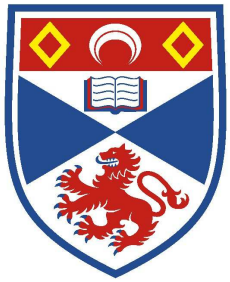
Supplementary data

R code for:

Modelling the Spatial Dynamics of Non-State Terrorism: World Study, 2002-2013

André Python



University of
St Andrews

Appendices

Appendix A

Extraction of World Terrorism Data (2002-2009)

A.1 Introduction

This Appendix provides the code used in R generated in order to extract data from GTD (Appendix A.2), GDELT (Appendix A.3), and RDWTI (Appendix A.4). In addition, the code used to attribute latitude and longitude to cities with Google Earth™ (Appendix A.5) and the code used to aggregate the databases all together (Appendix A.6) are provided as well.

A.2 Global Terrorism Database (GTD)

The main goal of this code is to extract, structure and merge GTD with other terrorism databases. The original GTD data cover the world from 1970 to 2013 without 1993 (events in 1993 are not geolocalised). The 'Entire GTD dataset' has been downloaded from the database provider (http://www.start.umd.edu/gtd/contact/). The extraction of data of GTD is provided below:

```
#GOAL: I extract GTD events from 1970 to 2013 worldwide and select
    accurate events.
#GTDsource is the original GTD file and covers the world from 1970 to
    2013 without 1993 (events in 1993 are not geolocalised).

#Initialisation
```

```
setwd("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/PhD_R/
    GTD")
require("lattice")
require("rgl")
require("vcd")
require("ggplot2")
require("foreign")
require("dplyr")
GTDoriginal<- read.csv("C:/Users/apython/Documents/Andre/Education/
    StAndrews/PhD/PhD_R/GTD/GTDsource.csv",
header= TRUE, dec=".", na.strings=c("","NA",".","-9","-99","-999")) #na.
    strings transform missing values into "NA"
#GTD original source contains 125,087 events

#we identify useful variables
a<-which(colnames(GTDoriginal)=="eventid") #save the position number of
    the variable to be kept
b<-which(colnames(GTDoriginal)=="iyear") #save the position number of
    the variable to be kept
c<-which(colnames(GTDoriginal)=="imonth") #save the position number of
    the variable to be kept
d<-which(colnames(GTDoriginal)=="iday") #save the position number of the
     variable to be kept
e<-which(colnames(GTDoriginal)=="country_txt") #save the position number
     of the variable to be kept
f<-which(colnames(GTDoriginal)=="city") #save the position number of the
     variable to be kept
g<-which(colnames(GTDoriginal)=="latitude") #save the position number of
     the variable to be kept
h<-which(colnames(GTDoriginal)=="longitude") #save the position number
    of the variable to be kept
i<-which(colnames(GTDoriginal)=="success") #save the position number of
    the variable to be kept
j<-which(colnames(GTDoriginal)=="suicide") #save the position number of
    the variable to be kept
k<-which(colnames(GTDoriginal)=="attacktype1") #save the position number
     of the variable to be kept
l<-which(colnames(GTDoriginal)=="attacktype1_txt") #save the position
    number of the variable to be kept
```

```
m<-which(colnames(GTDoriginal)=="weaptype1") #save the position number
    of the variable to be kept
n<-which(colnames(GTDoriginal)=="weaptype1_txt") #save the position
    number of the variable to be kept
o<-which(colnames(GTDoriginal)=="nkill") #save the position number of
    the variable to be kept
p<-which(colnames(GTDoriginal)=="specificity") #save the position number
     of the variable to be kept
q<-which(colnames(GTDoriginal)=="gname") #save the position number of
    the variable to be kept
r<-which(colnames(GTDoriginal)=="region") #save the position number of
    the variable to be kept
s<-which(colnames(GTDoriginal)=="targtype1") #save the position number
    of the variable to be kept
t<-which(colnames(GTDoriginal)=="targtype1_txt") #save the position
    number of the variable to be kept
u<-which(colnames(GTDoriginal)=="vicinity") #save the position number of
     the variable to be kept

#we keep useful variables of GTD
GTDoriginal<- GTDoriginal[c(a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u)]#
    keep the useful variables
#renaming "country" variable
names(GTDoriginal)[names(GTDoriginal)=="country_txt"] <- "country"
#summary(GTDoriginal$specificity)#to indicate the number of NA values
#summary(GTDoriginal$vicinity)
#selecting events which are geolocalised at a city level (specificity <
    3 ) or with no specificity: 65,245 events
#first change NA to 0 for specificity and vicinity that is not provided
    (NA)
GTDoriginal$specificity[is.na(GTDoriginal$specificity)] <- 0
GTDoriginal$vicinity[is.na(GTDoriginal$vicinity)] <- 0
# keep specifictiy 1 and 2 which are events geolocalised at a city/town/
    village level and specifity not clearly mentioned (0)
GTD <- subset(GTDoriginal, specificity<3)#in total 114,364 events
#if specifity is not informed (specificity =0), keep only events
    occurred in the immediate vicinity of the city mentioned (vicinity=1)
ind <- with(GTD, (specificity == 0 & vicinity==0))#events to be deleted:
     not specified and not localised the immediate vicinity of the city
    in question
```

```
#ind#view the logic result of the index ind
GTD <- GTD[!ind, ]#keep all events except thos which are not clearly
    geolocalised and not and the immediate vicinity of the citiy
    mentioned.
#we keep 70,823 events

####OPTIONAL#################################################
#selecting only lethal events  / selected years           #
#GTD <- subset(GTD, nkill > 0)#at least one fatality      #
GTD <- subset(GTD, iyear > 1983 & iyear<2014)#time period#
############################################################

#check type of target
summary(GTD$targtype1_txt)
#most hitted targets: private citizens and property, business,
    government, police, military
summary(GTD$latitude)#approx 5000 cities not geolocalised
#change class of latitude
GTD$latitude<-as.numeric(levels(GTD$latitude))[GTD$latitude]
# Optional step: we delete events (0) which were not geolocalised
#(unprecised geolocalisation (regional level, NA, Unknown, etc) - not
    founded with Google earth)
# the number of events does not change here, because all events are
    geolocalised
require(DataCombine)
GTD <- DropNA(GTD, Var="city")#from 57,511 events to 57,449 events (62
    events which have no city provided)
GTD <- DropNA(GTD, Var="country")#from 57,449 events to 57,449 events (0
     event which have no country provided)
#Adding the missing latitude and longitude of cities from GoogEcities
    into GTD.
require(DataCombine)
#importing the .csv file which contains cities geolocalised with Google
    earth
GoogEcities<- read.csv("C:/Users/apython/Documents/Andre/Education/
    StAndrews/PhD/PhD_R/GoogleEarth/GoogE_cities.csv", header= TRUE, dec=
    ".", na.strings=c("NA"))
#FillIn fills the values (variables "lat" then "long") from GoogE_cities
     to GTD if "latitude" (then "longitude") of GTD contains missing
    values (NA)
```

```
GTD <- FillIn(GTD, GoogEcities,Var1="latitude",Var2="lat", KeyVar=c("
    city","country"),allow.cartesian=T)
GTD <- FillIn(GTD, GoogEcities,Var1="longitude",Var2="long",KeyVar=c("
    city","country"),allow.cartesian=T)
#60,773 observations, so need to delete duplicates that have been
    generated by the Fillin (with allow.cartesian=T) process.
GTD<-GTD[!duplicated(GTD[,3]),]
GTD <- DropNA(GTD, Var="latitude")#from 57,449 events to 57,100 events
    (349 events which cannot be identified with google earth:
#due to misreported, mispelling issues, unprecise locations (between
    city A and city B) etc)
###########OPTIONAL STEP (no influence on this study (only use year)##
#we add the date (month and day) to missing values.                  #
#5 events have month and day=0; 148 events have day=0 only.          #
#We add month=6 and day=15 if the day and month are not provided.    #
#An alternative way: remove observations not temporally defined.     #
#replace values of month below 1 or above 12 by 6                    #
GTD$imonth[GTD$imonth>12 | GTD$imonth< 1]<-6                         #
#replace NA values by 6                                              #
GTD$imonth[is.na(GTD$imonth)]<-6                                     #
#replace values of day below 1 or above 31 by 15                     #
GTD$iday[GTD$iday>31 | GTD$iday< 1]<-15                              #
GTD$iday[is.na(GTD$iday)]<-6 #replace NA values by 6                 #
######################################################################
#Creating a POSIXIt time variable
#we concatenate year, month and day variables in gtd in one POSIXIt
    variable date
date<-as.Date(ISOdate(GTD$iyear,GTD$imonth, GTD$iday))
GTD$Date<-date #to put the new date vector in the dataframe
Rdate<-strptime(GTD$Date, format="%Y-%m-%d") #it create a new variable
    in a POSIXIt format, useful for further analysis in R.
GTD$Rdate<-Rdate#to put the new date vector in the dataframe
#Some variables were not classified in the right measure level.
GTD$eventid<-as.factor(GTD$eventid)
GTD$suicide<-as.logical(GTD$suicide)
GTD$success<-as.logical(GTD$success)
GTD$specificity<-as.factor(GTD$specificity)
#delete unuseful objects
rm(Rdate);rm(GTDoriginal);rm(GoogEcities);rm(a);rm(b);rm(c);rm(d);rm(
    date);rm(e);rm(f);rm(g);rm(h);rm(i);
```

```
rm(j);rm(k);rm(l);rm(m);rm(n);rm(o);rm(p);rm(q);rm(r);rm(s);rm(t)
#to delete possible duplicates in GTD
GTD<-unique(GTD)#in our case, this should not change GTD because we did
    not observe identical rows in GTD.
# #counting the number of events per terrorist group and per country
# Tgroup<-as.data.frame(table(GTD$gname, GTD$country))
# Tgroup <- Tgroup[order(-Tgroup$Freq),]
# write.table (Tgroup, "TgroupWorld.csv", sep=",",row.names=FALSE)
#
# #counting the number of fatalities per terrorist group and per country
# Tfat<-GTD
# Tfat <-aggregate(Tfat$nkill, by=list(Tfat$gname,Tfat$country),FUN=sum,
    na.rm=TRUE)
# Tfat <- Tfat[order(-Tfat$x),]
# write.table (Tfat, "Tfatworld.csv", sep=",",row.names=FALSE)
#dropping no more useful levels (very useful to do this for plots
    essentially)
GTD <- droplevels(GTD)
#export the dataframe GTD in a readable file, GTD.csv.
write.table (GTD, "GTDworld.csv", sep=",",row.names=FALSE)
#END#######################################################
```

A.3 Global Database of Events, Language, and Tone (GDELT)

The main goal of this code is to extract, structure and merge GDELT with other terrorism databases. In order to download GDELT, there is a package in R called "GetGDELT", which provides tools to extract data from GDELT. The time period has been selected from 2002 to 2014. Note that there is no data before 1979. Data have been extracted from GDELT year by year and with a server machine (OS: Linux; RAM: 64 Gb, cores: 16) because R could become unstable if one tries to extract data for the whole period at once and would require high level of RAM. Then, the yearly data frames are joined into one data frame, which includes data from 1979 to 2011.

The events in GDELT are classified according to CAMEO coding (more detail see: http://eventdata.psu.edu/cameo.dir/CAMEO.CDB.09b5.pdf). First, events in which terrorist attacks could be included have been selected. These are events within categories starting with 18, 19, or 20. In order to include only events perpetrated by terrorists, actors classified

as 'terrorists' have been selected. The date variable has been converted into a `POSIXIt` format, which is used for time series analysis. The extraction code is provided below:

```
#GOAL: I will read and explore the GDELT data. GDELT provides real-time
    data (not only terrorism data) from January 1, 1979.
# An additional objective is to structure GDELT database in order to
    merge it with other terrorism database.
#Initialisation
setwd("/home/andre/R/GDELT")#flipper (server) home file
require("vcd")
require("foreign")
require("plyr")
require("GDELTtools")
require("TimeWarp")
require("ff")
#Extract GDELT data year by year (for memory issues).The time process is
     approx. 6.5h.
#########################################################################
gdelt.02 <- GetGDELT(start.date="2002-01-01", end.date="2002-12-31",
                     local.folder = "C:/Users/apython/Documents/Andre/
                         Education/StAndrews/PhD/PhD_R/GDELT",
                     max.local.mb = 20000,
                     allow.wildcards=T,
                     filter=list(QuadClass=4,
                                 EventCode=c("18*","19*","20*"),
                                 Actor1Geo_Type=4,
                                 Actor1Name="TERRORIST"))
#########################################################################
gdelt.03 <- GetGDELT(start.date="2003-01-01", end.date="2003-12-31",
                     local.folder = "C:/Users/apython/Documents/Andre/
                         Education/StAndrews/PhD/PhD_R/GDELT",
                     max.local.mb = 20000,
                     allow.wildcards=T,
                     filter=list(QuadClass=4,
                                 EventCode=c("18*","19*","20*"),
                                 Actor1Geo_Type=4,
                                 Actor1Name="TERRORIST"))
#########################################################################
gdelt.04 <- GetGDELT(start.date="2004-01-01", end.date="2004-12-31",
                     local.folder = "C:/Users/apython/Documents/Andre/
                         Education/StAndrews/PhD/PhD_R/GDELT",
```

```
                    max.local.mb = 20000,
                    allow.wildcards=T,
                    filter=list(QuadClass=4,
                                EventCode=c("18*","19*","20*"),
                                Actor1Geo_Type=4,
                                Actor1Name="TERRORIST"))
#########################################################################
gdelt.05 <- GetGDELT(start.date="2005-01-01", end.date="2005-12-31",
                    local.folder = "C:/Users/apython/Documents/Andre/
                        Education/StAndrews/PhD/PhD_R/GDELT",
                    max.local.mb = 20000,
                    allow.wildcards=T,
                    filter=list(QuadClass=4,
                                EventCode=c("18*","19*","20*"),
                                Actor1Geo_Type=4,
                                Actor1Name="TERRORIST"))
#########################################################################
gdelt.06 <- GetGDELT(start.date="2006-01-01", end.date="2006-12-31",
                    local.folder = "C:/Users/apython/Documents/Andre/
                        Education/StAndrews/PhD/PhD_R/GDELT",
                    max.local.mb = 20000,
                    allow.wildcards=T,
                    filter=list(QuadClass=4,
                                EventCode=c("18*","19*","20*"),
                                Actor1Geo_Type=4,
                                Actor1Name="TERRORIST"))
#########################################################################
gdelt.07 <- GetGDELT(start.date="2007-01-01", end.date="2007-12-31",
                    local.folder = "C:/Users/apython/Documents/Andre/
                        Education/StAndrews/PhD/PhD_R/GDELT",
                    max.local.mb = 20000,
                    allow.wildcards=T,
                    filter=list(QuadClass=4,
                                EventCode=c("18*","19*","20*"),
                                Actor1Geo_Type=4,
                                Actor1Name="TERRORIST"))
#########################################################################
gdelt.08 <- GetGDELT(start.date="2008-01-01", end.date="2008-12-31",
                    local.folder = "C:/Users/apython/Documents/Andre/
                        Education/StAndrews/PhD/PhD_R/GDELT",
```

```
                    max.local.mb = 20000,
                    allow.wildcards=T,
                    filter=list(QuadClass=4,
                                EventCode=c("18*","19*","20*"),
                                Actor1Geo_Type=4,
                                Actor1Name="TERRORIST"))
#########################################################################
gdelt.09 <- GetGDELT(start.date="2009-01-01", end.date="2009-12-31",
                    local.folder = "C:/Users/apython/Documents/Andre/
                        Education/StAndrews/PhD/PhD_R/GDELT",
                    max.local.mb = 20000,
                    allow.wildcards=T,
                    filter=list(QuadClass=4,
                                EventCode=c("18*","19*","20*"),
                                Actor1Geo_Type=4,
                                Actor1Name="TERRORIST"))
#########################################################################
gdelt.10 <- GetGDELT(start.date="2010-01-01", end.date="2010-12-31",
                    local.folder = "C:/Users/apython/Documents/Andre/
                        Education/StAndrews/PhD/PhD_R/GDELT",
                    max.local.mb = 20000,
                    allow.wildcards=T,
                    filter=list(QuadClass=4,
                                EventCode=c("18*","19*","20*"),
                                Actor1Geo_Type=4,
                                Actor1Name="TERRORIST"))
#########################################################################
gdelt.11 <- GetGDELT(start.date="2011-01-01", end.date="2011-12-31",
                    local.folder = "C:/Users/apython/Documents/Andre/
                        Education/StAndrews/PhD/PhD_R/GDELT",
                    max.local.mb = 20000,
                    allow.wildcards=T,
                    filter=list(QuadClass=4,
                                EventCode=c("18*","19*","20*"),
                                Actor1Geo_Type=4,
                                Actor1Name="TERRORIST"))
#########################################################################
gdelt.12 <- GetGDELT(start.date="2012-01-01", end.date="2012-12-31",
                    local.folder = "C:/Users/apython/Documents/Andre/
                        Education/StAndrews/PhD/PhD_R/GDELT",
```

```
                        max.local.mb = 20000,
                        allow.wildcards=T,
                        filter=list(QuadClass=4,
                                    EventCode=c("18*","19*","20*"),
                                    Actor1Geo_Type=4,
                                    Actor1Name="TERRORIST"))
#########################################################################
gdelt.13 <- GetGDELT(start.date="2013-01-01", end.date="2013-12-31",
                        local.folder = "C:/Users/apython/Documents/Andre/
                            Education/StAndrews/PhD/PhD_R/GDELT",
                        max.local.mb = 20000,
                        allow.wildcards=T,
                        filter=list(QuadClass=4,
                                    EventCode=c("18*","19*","20*"),
                                    Actor1Geo_Type=4,
                                    Actor1Name="TERRORIST"))
#########################################################################
#Append gdelt.yy with rbind function
#if gdelt.1 and gdelt.2 have not the same #of variables, use "smartbind"
    function from gtools package
gdelt<-rbind(gdelt.02,gdelt.03,gdelt.04,gdelt.05,gdelt.06,gdelt.07,gdelt
    .08,gdelt.09,gdelt.10,gdelt.11,gdelt.11,gdelt.13)
#########################################################################
#Add Date in POSIXIt format for gdelt
Date<-dateParse(gdelt$SQLDATE, format=NULL, stop.on.error = F, quick.try
    = F,
                dross.remove = FALSE, na.strings = c("NA", ""), ymd8 = T
                    )
Date<-strptime(Date, format="%Y-%m-%d") #it creates a new variable
    considered as a time variable in R.
#class(Date1)#to see the class of the new vector, should be POSIXIt
#Add the new date (DateR) considered as a vector, into gdelt dataframe
gdelt$DateR <- Date
#Clean gdelt of possible duplicates
gdelt<-unique(gdelt, by="GLOBALEVENTID")# clean gdelt of possible
    duplicates
#########################################################################
#Delete intermediate data (not useful anymore)
rm(Date);rm(gdelt.00);rm(gdelt.01);rm(gdelt.02);rm(gdelt.03);rm(gdelt
    .04);
```

```
rm(gdelt.05);rm(gdelt.06);rm(gdelt.07);rm(gdelt.08);rm(gdelt.09);rm(
    gdelt.10);
rm(gdelt.11);rm(gdelt.12);rm(gdelt.13);rm(gdelt.14);rm(gdelt.15);
rm(gdelt.79);rm(gdelt.80);rm(gdelt.81);rm(gdelt.82);rm(gdelt.83);rm(
    gdelt.84);
rm(gdelt.85);rm(gdelt.86);rm(gdelt.87);rm(gdelt.88);rm(gdelt.89);rm(
    gdelt.90);
rm(gdelt.91);rm(gdelt.92);rm(gdelt.93);rm(gdelt.94);rm(gdelt.95);rm(
    gdelt.96);
rm(gdelt.97);rm(gdelt.98);rm(gdelt.99);rm(start)
# OPTIONAL STEP AFTER IMPORTING GDELT####################
#NORMALIZE THE NUMBER OF EVENTS FOR TEMPORAL COMPARISONS)#
# #normalising GDELT (to be checked how to do it)        #
# #gdelt.normed <- NormEventCounts(gdelt.subset,         #
# #unit.analysis="country.month",                        #
# #var.name="protest")                                   #
##########################################################
#Export the dataframe in a readable file
gdeltworld<-gdelt
write.table (gdeltworld, "gdeltworld.csv", sep=",",row.names=FALSE)
save.image("/home/andre/R/GDELT/gdelt.RData")
#END#####################################
```

A.4 RAND Database of Worldwide Terrorism Incidents (RDWTI)

The goal is to extract, structure and merge the RAND Database of Worldwide Terrorism Incidents (RDWTI) with other data on terrorism. RDWTI data have been downloaded Dec 17th 2013 from http://smapp.rand.org/rwtid/search_form.php. RDWTI data are available until 2009 only. The format of the date has been changed ("UK date": dd/mm/yyyy) in order to facilitate its import in R.

The source file has been imported in R. Only events which were provided a city name are kept. If no city were provided, the observation was removed. The latitude and longitude of cities without coordinate have been completed with Google Earth™ (Section A.5). Some cities which could not be geolocalised with Google Earth™ (unprecised geolocalisation) have been excluded. The extraction code for RDWTI is provided below:

```
#The Goal is to import RDWTI and to add the geolocalision (latitude,
    longitude) of the events of RDWTI using Google earth
```

```
#and to set up a structured dataframe which will be merged with other
    terrorism databases.
#The original source is RANDtotal.csv; we keep only the events
    attributed at a city level.
#initialisation
setwd("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/PhD_R/
    RDWTI")
require("lattice")
require("rgl")
require("vcd")
require("ggplot2")
require("foreign")
require("DataCombine")
require("ggmap")
require("devtools")
require("data.table")
#Reading the original file saved in .csv which includes 40,129 events
RDWTItotal<- read.csv("C:/Users/apython/Documents/Andre/Education/
    StAndrews/PhD/PhD_R/RDWTI/RANDtotal.csv", header= TRUE, dec=".", na.
    strings=c("","NA"))
#Deleting some unuseful variables, which can create problem in the
    import because of special string character
j<-which(colnames(RDWTItotal)=="Description")#save the position number
    of the variable to be deleted
i<-which(colnames(RDWTItotal)=="Perpetrator")#save the position number
    of the variable to be deleted
RDWTItotal<- RDWTItotal[c(-j)]#delete the unuseful variables "
    Description" . Finally keep "Perpetrator", cause useful
#Changing one country name(Germany) in order to optimize the merging
    step with Google Earth city localisation
RDWTItotal$Country[RDWTItotal$Country == "Federal␣Republic␣of␣Germany"]
    <-"Germany"
#Keeping only events, which are localised at a city level, so if no city
     is provided, the observation (entire row) is deleted.
RDWTInet<-subset(RDWTItotal, !is.na(RDWTItotal$City))
# We keep therefore 35,155 localised events
#Renaming the variable "City" as "city" (useful for the merging step)
require(reshape)
RDWTInet<-rename(RDWTInet, c(City="city"))
RDWTInet<-rename(RDWTInet, c(Country="country"))
```

```
#Adding the missing latitude and longitude of cities from GoogEcities (
    lat, long of cities provided by Google Earth) into RDWTIGTD.
#we read the cities geolocalised with Google Earth (city, lat, lon)
GoogEarth<- read.csv("C:/Users/apython/Documents/Andre/Education/
    StAndrews/PhD/PhD_R/GoogleEarth/GoogE_cities.csv", header= TRUE, dec=
    ".", na.strings="")
GoogEarth<-GoogEarth[,-5]#delete the last variable comment(not used)
#Merging by city (to add the geolocalisation variables latitude and
    longitude of Google Earth localisation into RDWTI)
RDWTInet <- merge(RDWTInet,GoogEarth,by=c("city","country"), all.x=T) #
    the variable "city" and country in RDWTI and GoogEarth should be
    identically written.
#Deleting non RDWTI events and non-geolocalised events
RDWTInet <- DropNA(RDWTInet, Var="city")#: DropNA drops rows from a data
     frame in which missing (NA) values are present in the variable "city
    " (of RDWTI events)
RDWTInet <- DropNA(RDWTInet, Var="lat")#: DropNA drops rows from a data
    frame in which missing (NA) values are present in the variable "lat"
    (of RDWTI events)
#from the 35,297 events have been localised by Google Earth. Some
    locations have not been found.
#Renaming useful variables
RDWTInet<-rename(RDWTInet, c(lat="latitude"))
RDWTInet<-rename(RDWTInet, c(long="longitude"))
RDWTInet<-rename(RDWTInet, c(country.x="country"))
RDWTI<-RDWTInet
#Changing the date into a POSIXIt format
RDWTIdate<-strptime(RDWTI$Date, format="%d/%m/%Y") #it creates a new
    variable considered as a time variable in R.
as.Date(RDWTIdate)# put in the same format than the other databases "%Y
    -%m-%d"
#Adding the new date (RDWTIdate) considered as a vector, into RDWTI
    dataframe
RDWTI$RDWTIdate <- RDWTIdate
#extracting the year variable
year<-factor(format(RDWTI$RDWTIdate,'%Y'))
#Adding the "year" variable into RDWTI dataframe
RDWTI$year <- year
#Deleting the intermediate and no more useful data
```

```
rm(GoogEarth);rm(RDWTItotal); rm(RDWTInet);rm(RDWTIdate);rm(i);rm(j);rm(
    year)
#Exporting the final result as a table
write.table (RDWTI, "RDWTIworld.csv", sep=",",row.names=FALSE)
#END#########################################################
```

In R, missing latitude and longitude of cities have been obtained with Google Earth API extraction™. The code is provided below:

A.5 Google Earth™ API extraction of missing coordinates

```
#initialisation
setwd("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/PhD_R/
    GoogleEarth")
require("lattice")
require("rgl")
require("vcd")
require("ggplot2")
require("foreign")
require("DataCombine")
require("ggmap")
require("devtools")
require("data.table")
#especially needed for Google API
require(RCurl)
require(RJSONIO)
# Code from Tony Breyal on: http://stackoverflow.com/questions/3257441/
    geocoding-in-r-with-google-maps Used to extract latitude, longitude
    of a city
construct.geocode.url <- function(address, return.call = "json", sensor
    = "false") {
  root <- "http://maps.google.com/maps/api/geocode/"
  u <- paste(root, return.call, "?address=", address, "&sensor=", sensor
      , sep = "")
  return(URLencode(u))
}
gGeoCode <- function(address,verbose=FALSE) {
  if(verbose) cat(address,"\n")
  u <- construct.geocode.url(address)
  doc <- getURL(u)
```

```
  x <- fromJSON(doc, simplify = FALSE)
  if(x$status=="OK") {
    lat <- x$results[[1]]$geometry$location$lat
    lng <- x$results[[1]]$geometry$location$lng
    return(c(lat, lng))
  } else {
    return(c(NA,NA))
  }
}
#new lines added to read one file of multiple locations instead of one
    location
#a .csv file should be created and named "addresses.csv" with two
    columns, headers are 'addressid' and 'address'.
addresses <- read.csv("addresses.csv",header=T)
newaddresses <- addresses
for(i in 1:nrow(addresses)) {
  row <- addresses[i,]
  row.to.update <- which(addresses$addressid==row[1,1])
  x <- gGeoCode(row[1,2])
  newaddresses[row.to.update, 3] = paste(c(x), collapse = "␣")
}
#export the .csv with a new column (lat lon)
write.csv(newaddresses, file = "geocodes.csv")
#end#########################################
```

A.6 Merging GTD, GDELT, and RDWTI

The goal is to merge GTD, GDELT, and RDWTI together. This will allow us to compare them and plot events from different terrorist databases together based on common variables: latitude, longitude, number of death(s), and time variables.

The code "ALL_data.R" use the objects created in the previous operations run to extract GTD, GDELT, and GTD. These are saved in the workspace "All_Data.RData". It contains all databases into one dataframe. In this process — mainly a cosmetic one — the variables are renamed in order to be identical. With regard to RDWTI, a new variable 'year' has been created from a time variable in a `POSIXit` format. Only the main variables from the dataframes are extracted, while unused variables are deleted. With regard to GDELT, which does not contain the number of death (in contrast to the other dataframes), an empty vector has been created. This latter operation is required to merge the dataframes together.

Finally, the dataframes are merged into one dataframe exported into a .csv file: “ALL.csv”. The code is provided below:

```
#Goal: to put in form all terrorist databases (DTV,GTD,RDWTI, GDELT) in
    order to be able to compare and to plot them.
#There are different names for variables lat/long/#death/year/POSIXIt
    date in the databases:
#DTV: latitude, longitude, deathtoll, year, DTVdate
#GTD: latitude, longitude, nkill, iyear, Rdate
#RDWTI: latitude, longitude, Fatalities, (!no year!), RDWTIdate
#GDELT: ActionGeo_Lat, ActionGeo_Long, (!no#kill!), Year, DateR.
#The number of deaths is not present in GDELT, however another file GKG
    could provide them (to be looked at)
#DTV,GTD,RDWTI could be compared regarding 4 variables: latitude,
    longitude,time, and #death
#DTV,GTD,RDWTI and GDELT could be compared regarding 3 variables:
    latitude,longitude and time.
#initialisation
setwd("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/PhD_R/
    ALL")
#Reading dataframe from the global environment (if saved) or reading the
     .csv output of each individual process
GTD<- read.csv("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD
    /PhD_R/GTD/GTDworld.csv", header= TRUE, dec=".", na.strings=c("","NA"
    ))
RDWTI<-read.csv("C:/Users/apython/Documents/Andre/Education/StAndrews/
    PhD/PhD_R/RDWTI/RDWTIworld.csv", header= TRUE, dec=".", na.strings=c(
    "","NA"))
gdelt<-read.csv("C:/Users/apython/Documents/Andre/Education/StAndrews/
    PhD/PhD_R/GDELT/gdeltworld.csv", header= TRUE, dec=".", na.strings=c(
    "","NA"))
#First step: When needed, rename variables in order to have the same
    names in all dataframes: "latitude", "longitude", "death", "year", "
    time".
require(reshape)
require(lattice)
require(sp)
require(RgoogleMaps)
require(rgdal)
require(ggplot2)
require(maptools)
```

```
GTD <- rename(GTD, c(nkill="death"));GTD <- rename(GTD, c(Rdate="time"))
    ;GTD <- rename(GTD, c(iyear="year"));
RDWTI <- rename(RDWTI, c(Fatalities="death"));RDWTI <- rename(RDWTI, c(
    RDWTIdate="time"));
gdelt<-rename(gdelt,c(ActionGeo_Lat="latitude"));gdelt<-rename(gdelt,c(
    ActionGeo_Long="longitude"));gdelt<-rename(gdelt,c(Year="year"));
    gdelt<-rename(gdelt,c(DateR="time"))#no death variable!
GDELT<-gdelt#changing the name of the dataframe gdelt in GDELT for
    avoiding confusion in next steps
rm(gdelt)#deleting the old dataframe gdelt
RDWTI$year<-as.integer(RDWTI$year)# put "year" in the same format (
    integer) than the other dataframes
RDWTI$death<-as.numeric(RDWTI$death)# put "death" in the same format (
    numeric) than the other dataframes
#check class of variables year in all dataframes
# class(RDWTI$year)
# class(GTD$year)
# class(GDELT$year)
#Second step: to keep only the needed variables (latitude,longitude,
    death,year,time) in all dataframes
GTD<-subset(GTD, select=c("latitude","longitude","death","year","time"))
DTV<-subset(DTV, select=c("latitude","longitude","death","year","time"))
RDWTI<-subset(RDWTI, select=c("latitude","longitude","death","year","
    time"))
GDELT<-subset(GDELT, select=c("latitude","longitude","year","time"))#
    GDELT has not "death" variable!
#Third step: to add an identification variable to recognise the values
    of each dataframe
GTD$id<-(GTD$id="GTD")
GDELT$id<-(GDELT$id="GDELT")
GDELT$death<-(GDELT$death="NA")#create a variable "death" with missing
    elements in order to merge it with the other df.
GDELT$death<-as.numeric(GDELT$death)#put "death" in the same format (
    numeric) than the other dataframes
RDWTI$id<-(RDWTI$id="RDWTI")
#Fourth step: to merge all dataframes in one the so called "ALL"
    dataframe.
require(gtools)
ALL<-rbind(GTD,RDWTI)#add the resulting df to RDWTI
ALL<- rbind(ALL,GDELT)#add the resulting df to GDELT.
```

```
#str(ALL)#to check the database ALL
#exporting "ALL" df as a .csv file
write.table (ALL, "ALL.csv", sep=",",row.names=FALSE)
#exporting "GDELT" df as a .csv file
write.table (GDELT, "GDELT.csv", sep=",",row.names=FALSE)
#exporting "RDWTI" df as a .csv file
write.table (RDWTI, "RDWTI.csv", sep=",",row.names=FALSE)
#exporting "GTD" df as a .csv file
write.table (GTD, "GTD.csv", sep=",",row.names=FALSE)
#plot preparation
rm(list=setdiff(ls(), "ALL"))# it keeps only the dataframe "ALL"
world.map <- readShapeSpatial("C:/Users/apython/Documents/Andre/
    Education/StAndrews/PhD/PhD_ArcGIS/World_map/world.shp")
proj4string(world.map)<-CRS("+proj=longlat_+datum=WGS84_+no_defs+towgs84
    =0,0,0+ellps=WGS84")
studyarea <- world.map[world.map$F_AREA > 9000000000,]#delete small
    polygons
A <- subset(ALL, year > 2001 & year < 2015)#we create a subset 1979-2005
     in order to be able to compare all databases
A2 <- subset(ALL, year > 2001 & year < 2010)#we create a subset
    1979-2005 in order to be able to compare all databases
GTD<-subset(A2, id=="GTD")#we could create a subset of GTD from all (
    could be useful to do plot only on GTD or another dataframe)
RDWTI<-subset(A2, id=="RDWTI")#we could create a subset of GTD from all
    (could be useful to do plot only on GTD or another dataframe)
GDELT<-subset(A2, id=="GDELT")#we could create a subset of GTD from all
    (could be useful to do plot only on GTD or another dataframe)
#plot
pdf("ALL.pdf",width=14,height=5)
par(mar=c(0, 0, 0, 0),oma=c(0,0,0,0))
plot(studyarea,xlim = c(-180, 180), ylim = c(-45, 75),cex=1.3)
points(GDELT$longitude,GDELT$latitude,cex=0.3,pch=3,col="green")
points(RDWTI$longitude,RDWTI$latitude,cex=0.3,pch=2,col="red")
points(GTD$longitude,GTD$latitude,cex=0.3,pch=1,col="black")
dev.off()
#END#################################################################
```

Appendix B

Exploratory Analysis of World Terrorism Data (2002-2009)

B.1 Introduction

This Appendix provides the code which has been generated to explore temporal and spatial patterns of GDELT, GTD, and RDWTI. First, Appendix B.2 shows the code used to calculate cross-correlations between the databases in time (2002-2009) and space. Second, Appendix B.4 provides the code in `R` which has been used to calculate intensity, density, and the F-function. Third, Appendix B.5 provides a code generated in `R` which has been used to calculate the pcf function. Fourth, Appendix C.2 provides the code used to compute correlation between the covariates.

B.2 Time Series: monthly average and trends (2002-2009)

```
#The goal of this process is to plot the different data from the
    dataframe ALL, which contains data of DTV, GTD, GDELT, and RDWTI.
#It uses All.csv, the output of All_data.R, which should be executed
    prior this operation process.
#1: initialisation
setwd("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/PhD_R/
    ALL")
require("plyr")
require("data.table")
require("ggplot2")
```

```
ALL<-read.csv("ALL.csv", header= TRUE, dec=".", na.strings=c("","NA"))
#we create a subset 2002-2009 in order to be able to compare all
    databases on a same temporal framework
A <- subset(ALL, year > 2001 & year < 2010)
#Databases time period: GTD: 1970-2013; GDELT: 1979-2013 (downloaded but
     can have until today);RDWTI: 1968-2009. The maximum common period to
     all databases is: 2002-2009

# STATISTICS (deaths, number of events) for 2002-2009
#mean and standard deviation of number of death (including non-lethal)
A0<-A[complete.cases(A),]#exclude NA
deathmean <- ddply(A0, "id", summarise, dmean=mean(death), na.rm = TRUE)
deathsd <- ddply(A0, "id", summarise, dsd=sd(death), na.rm = TRUE)
# Subsetting the Nb. of non-lethal events
GTD1<-subset(A, id=="GTD")
RDWTI1<-subset(A, id=="RDWTI")
#Counting the number of events with no casualties
GTDd<-as.data.frame(table(GTD1$death))
RDWTId<-as.data.frame(table(RDWTI1$death))
GTDd;RDWTId#proportion of non-lethal events/total events:GTD 5401/
    12031=0.45 ; RDWTI: 12360/23873=0.52
##################################################
#TEMPORAL ANALYSIS of number events
#create a vector with months from 2002 to 2009
#creating all months from 2002 to 2009 (96 months for 8 years)
timeformatted <- as.Date(A$time,"%Y-%m-%d")
A$YYmm<-strftime(timeformatted,"%Y-%m")
s <- as.Date("2002-01-01")
e <- as.Date("2009-12-31")
months<-seq(from=s, to=e, by="month")
as.factor(months)
monthformatted <- as.Date(months,"%Y-%m-%d")
M<-as.data.frame(strftime(monthformatted,"%Y-%m"))#extract year and
    month
colnames(M) <- "YYmm"

#TEMPORAL ANALYSIS of number events in GTD
#we look at GTD so we subset the sample
GTD2<-subset(A, id=="GTD")
#merge the dataframe with all days with each terrorist events data
```

```
#note that it is possible that no terrorist events have been recorded in
    some days, so
#all data should be kept (all=TRUE)
B<-merge(M, GTD2, by="YYmm", all=TRUE)
#sum the number of events per day and per database (id)
Nb.evt2<-ddply(B,.(YYmm, id),nrow)
Nb.evt2<-rename(Nb.evt2, c(V1="nbevents"))
#put 0 in the number of events if NA is present in the "id" column
Nb.evt2$nbevents[is.na(Nb.evt2$id)]<-0
myvector<-as.numeric(Nb.evt2$nbevents)
#creating a time series
GTDts <- ts(myvector, start=c(2002,1,1), end=c(2009,12,31), frequency
    =12)
plot(GTDts, ylab="Nb.monthly␣events␣in␣GTD")
GTDtsdecomp <- decompose(GTDts)
plot(GTDtsdecomp)
plot(GTDtsdecomp$trend, ylab="GTD␣trend")
GTDtrend<-as.data.frame(GTDtsdecomp$trend)
GTDdf<-cbind(GTDtrend, Nb.evt2)
GTDdf$id[is.na(GTDdf$id)]<-"GTD"
GTDdf$season<-as.numeric(GTDtsdecomp$seasonal)
GTDdf$random<-as.numeric(GTDtsdecomp$random)
names(GTDdf)<-c("month","YYmm","id","trend","season","random")

#TEMPORAL ANALYSIS of number of deaths
#we look at GTD so we subset the sample
#we put 0 if NA is present in number of death
B$death[is.na(B$death)]<-0
#now we sum the number of death per day and per database (id)
death2<-ddply(B, "YYmm", transform, totdeaths = sum(death))#calculate
    the sum of deaths per month
death2<-death2[!duplicated(death2[,1]),]#delete replicates
death2<-death2[c(1,8)]#keep useful columns only
myvector<-as.numeric(death2$totdeaths)#create a numeric vector
#creating a time series
GTDts2 <- ts(myvector, start=c(2002,1,1), end=c(2009,12,31), frequency
    =12)
GTDts2decomp <- decompose(GTDts2)
GTD2trend<-as.data.frame(GTDts2decomp$trend)
GTD2df<-cbind(GTD2trend, death2)
```

```
GTD2df$season<-as.numeric(GTDts2decomp$seasonal)
GTD2df$random<-as.numeric(GTDts2decomp$random)
names(GTD2df)<-c("month","YYmm","trend","season","random")
GTD2df$id<-"GTD" #adding a new variable id

#TEMPORAL ANALYSIS of number of events in RDWTI
#we look at RDWTI so we subset the sample
RDWTI2<-subset(A, id=="RDWTI")
B<-merge(M, RDWTI2, by="YYmm", all=TRUE)
#sum the number of events per day and per database (id)
Nb.evt4<-ddply(B,.(YYmm, id),nrow)
Nb.evt4<-rename(Nb.evt4, c(V1="nbevents"))
#put 0 in the number of events if NA is present in the "id" column
Nb.evt4$nbevents[is.na(Nb.evt4$id)]<-0
myvector<-as.numeric(Nb.evt4$nbevents)
#creating a time series
RDWTIts <- ts(myvector, start=c(2002,1,1), end=c(2009,12,31), frequency
    =12)
plot(RDWTIts, ylab="Nb.monthly␣events␣in␣RDWTI")
RDWTItsdecomp <- decompose(RDWTIts)
plot(RDWTItsdecomp)
plot(RDWTItsdecomp$trend, ylab="RDWTI␣trend")
RDWTItrend<-as.data.frame(RDWTItsdecomp$trend)
RDWTIdf<-cbind(RDWTItrend, Nb.evt4)
RDWTIdf$season<-as.numeric(RDWTItsdecomp$seasonal)
RDWTIdf$random<-as.numeric(RDWTItsdecomp$random)
names(RDWTIdf)<-c("month","YYmm","id","trend","season","random")

#TEMPORAL ANALYSIS of number of deaths in RDWTI
#we look at RDWTI so we subset the sample
#we put 0 if NA is present in number of death
B$death[is.na(B$death)]<-0
#sum the number of death per day and per database (id)
death2<-ddply(B, "YYmm", transform, totdeaths = sum(death))#calculate
    the sum of deaths per day
death2<-death2[!duplicated(death2[,1]),]#delete replicates
death2<-death2[c(1,8)]#keep useful columns only
myvector<-as.numeric(death2$totdeaths)#create a numeric vector
#creating a time series
```

```
RDWTIts2 <- ts(myvector, start=c(2002,1,1), end=c(2009,12,31),
    frequency=12)
RDWTIts2decomp <- decompose(RDWTIts2)
RDWTI2trend<-as.data.frame(RDWTIts2decomp$trend)
RDWTI2df<-cbind(RDWTI2trend, death2)
RDWTI2df$season<-as.numeric(RDWTIts2decomp$seasonal)
RDWTI2df$random<-as.numeric(RDWTIts2decomp$random)
names(RDWTI2df)<-c("month","YYmm","trend","season","random")
RDWTI2df$id<-"RDWTI" #adding a new variable id

#TEMPORAL ANALYSIS of number of events in GDELT
#we look at GDELT so we subset the sample
GDELT2<-subset(A, id=="GDELT")
#merge the dataframe with all days with each terrorist events data
#note that it is possible that no terrorist events have been recorded in
    some days, so
#all data should be kept (all=TRUE)
B<-merge(M, GDELT2, by="YYmm", all=TRUE)
#sum the number of events per day and per database (id)
Nb.evt5<-ddply(B,.(YYmm, id),nrow)
Nb.evt5<-rename(Nb.evt5, c(V1="nbevents"))
#put 0 in the number of events if NA is present in the "id" column
Nb.evt5$nbevents[is.na(Nb.evt5$id)]<-0
myvector<-as.numeric(Nb.evt5$nbevents)
#creating a time series
GDELTts <- ts(myvector, start=c(2002,1,1), end=c(2009,12,31), frequency
    =12)
plot(GDELTts, ylab="Nb.monthly␣events␣in␣GDELT")
GDELTtsdecomp <- decompose(GDELTts)
plot(GDELTtsdecomp)
plot(GDELTtsdecomp$trend, ylab="GDELT␣trend")
GDELTtrend<-as.data.frame(GDELTtsdecomp$trend)
GDELTdf<-cbind(GDELTtrend, Nb.evt5)
GDELTdf$season<-as.numeric(GDELTtsdecomp$seasonal)
GDELTdf$random<-as.numeric(GDELTtsdecomp$random)
names(GDELTdf)<-c("month","YYmm","id","trend","season","random")

#plots (time unit: month)
#putting variable in numeric
GTDdf$month<-as.numeric(GTDdf$month)
```

```
GDELTdf$month<-as.numeric(GDELTdf$month)
RDWTIdf$month<-as.numeric(RDWTIdf$month)
#merging time series
trend<-rbind(GTDdf,RDWTIdf,GDELTdf)
#color settings
my.settings <- list(
  strip.background=list(col="transparent"),
  strip.border=list(col="black")
)
#plotting nb. of monthly events using lattice
require("lattice")
plot.new()
mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    Latex/phd-thesis-template-phd-latex-template-latest-stable/Chapter4/
    Figs/PDF/monthandtrend.pdf")
pdf(file=mypath,onefile=FALSE,width=16,height=8,paper="special")
par(oma=c(0, 0.2, 0, 0), mar=c(4.1, 4.5, 2.1, 0.5),cex=8)
xyplot(month + trend ~ YYmm | id, between = list(x = 0.5),
data = trend, type = "l",xlab=list(label="Year",cex=2),ylab=list(label="
    Nb.terrorist events per month and trend",cex=2),
lty = c(1,1),lwd = c(7,2),col.line = c("red","black"), layout = c(3, 1),
scales=list( x=list(at=seq(1,96,24),#in total 96 days are included and
    each mark should be placed after one year=12 months
labels=c("2002","2004","2006","2008","2010"),cex=2),
y=list(at=seq(0,600,200),labels=c("0","200","400","600"),cex=2)),
par.strip.text=list(cex=2),stack=TRUE,
par.settings = list(strip.background=list(col=NA),cex=4,list(
                    )))
dev.off();dev.off()

#seasonality and randomness
#plotting nb.monthly events using lattice
require("lattice")
plot.new()
mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    Latex/phd-thesis-template-phd-latex-template-latest-stable/Chapter4/
    Figs/PDF/monthandseason.pdf")
pdf(file=mypath,onefile=FALSE,width=16,height=8,paper="special")
par(oma=c(0, 0.2, 0, 0), mar=c(4.1, 4.5, 2.1, 0.5),cex=8)
xyplot(season+random  ~ YYmm | id, between = list(x = 0.5),
```

```
        data = trend, type = "l",xlab=list(label="Year",cex=2),ylab=list(
            label="Seasonal and random components",cex=2),
        lty = c(1,1),lwd = c(7,2),col.line = c("red","black"), layout = c
            (3, 1),
        scales=list( x=list(at=seq(1,96,24),#in total 96 days are
            included and each mark should be placed after one year=12
            months
                                labels=c("2002","2004","2006","2008","2010"),
                                    cex=2),
                        y=list(at=seq(0,600,200),labels=c("0","200","400","
                            600"),cex=2)),
        par.strip.text=list(cex=2),stack=TRUE,
        par.settings = list(strip.background=list(col=NA),cex=4,list(
        )))
dev.off();dev.off()

#plot nb. monthly fatalities
#putting variable in numeric
GTD2df$month<-as.numeric(GTD2df$month)
RDWTI2df$month<-as.numeric(RDWTI2df$month)
#merging time series
trend2<-rbind(GTD2df, RDWTI2df)
#plotting nb.monthly fatalities using lattice
require("lattice")
xyplot(month + trend ~ YYmm | id, between = list(x = 0.5),
data = trend2, type = "l",xlab=list(label="Year", cex=1.3),ylab=list(
    label="Nb. fatalities per month and trend", cex=1.3),
lty = c(1,1),lwd = c(7,2),col.line = c("red","black"), layout = c(3, 1),
scales=list( x=list(at=seq(1,96,12),
labels=c("2002","2003","2004","2005","2006","2007","2008","2009","2010")
    )))
#plotting nb.monthly events using lattice
require("lattice")
plot.new()
mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    Latex/phd-thesis-template-phd-latex-template-latest-stable/Chapter4/
    Figs/PDF/monthandtrendfat.pdf")
pdf(file=mypath,onefile=FALSE,width=16,height=8,paper="special")
par(oma=c(0, 0.2, 0, 0), mar=c(4.1, 4.5, 2.1, 0.5),cex=8)
xyplot(month + trend ~ YYmm | id, between = list(x = 0.5),
```

```
        data = trend2, type = "l",xlab=list(label="Year",cex=2),ylab=list
            (label="Nb.fatalities␣per␣month␣and␣trend",cex=2),
        lty = c(1,1),lwd = c(7,2),col.line = c("red","black"), layout = c
            (3, 1),
        scales=list( x=list(at=seq(1,96,24),#in total 96 days are
            included and each mark should be placed after one year=12
            months
                                labels=c("2002","2004","2006","2008","2010"),
                                    cex=2),
                        y=list(at=seq(0,1250,250),labels=c("0","250","500","
                            750","1000","1250"),cex=2)),
        par.strip.text=list(cex=2),stack=TRUE,
        par.settings = list(strip.background=list(col=NA),cex=4,list(
        )))
dev.off();dev.off()

#seasonality and random components
require("lattice")
plot.new()
mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    Latex/phd-thesis-template-phd-latex-template-latest-stable/Chapter4/
    Figs/PDF/monthandseasonfat.pdf")
pdf(file=mypath,onefile=FALSE,width=16,height=8,paper="special")
par(oma=c(0, 0.2, 0, 0), mar=c(4.1, 4.5, 2.1, 0.5),cex=8)
xyplot(season+random  ~ YYmm | id, between = list(x = 0.5),
        data = trend2, type = "l",xlab=list(label="Year",cex=2),ylab=list
            (label="Seasonal␣and␣random␣components",cex=2),
        lty = c(1,1),lwd = c(7,2),col.line = c("red","black"), layout = c
            (3, 1),
        scales=list( x=list(at=seq(1,96,24),#in total 96 days are
            included and each mark should be placed after one year=12
            months
                                labels=c("2002","2004","2006","2008","2010"),
                                    cex=2),
                        y=list(at=seq(0,1250,250),labels=c("0","250","500","
                            750","1000","1250"),cex=2)),
        par.strip.text=list(cex=2),stack=TRUE,
        par.settings = list(strip.background=list(col=NA),cex=4,list(
        )))
dev.off();dev.off()
```

```
#checking correlation between time series data in terms of nb. events
    per month
#first put databases year, DTV, GTD, GDELT, RDWTI in column and values
    rows
require("reshape2")
corr<-trend
corr$index<-as.numeric(as.factor(corr$YYmm))#create a new column for
    indexing the time
corr<-corr[complete.cases(corr),]#delete if NA is present
corr<-corr[c(-4,-2)]#delete trend
corr<-dcast(corr, index~ id, value.var="month")#month gives the nb.
    events per month
a<-cor.test(corr$GTD, corr$RDWTI)
c<-cor.test(corr$GTD, corr$GDELT)
e<-cor.test(corr$RDWTI, corr$GDELT)
#correlation's test results
a;c;e

#checking correlation between time series data in terms of nb. deaths
#first put databases year, DTV, GDELT, RDWTI in column and values rows
require("reshape2")
#using time2, GDELT does not provide nb. deaths.
corr2<-trend2
corr2$index<-as.numeric(as.factor(corr2$YYmm))#create a new column for
    indexing the time
corr2<-corr2[complete.cases(corr2),]#delete if NA is present
corr2<-corr2[c(-2,-3)]#delete trend
corr2<-dcast(corr2, index~ id, value.var="month")#month gives the nb.
    events per month
#correlation's test results
g<-cor.test(corr2$GTD, corr2$RDWTI);g
#checking of time series (number of deaths and number of events) are
    random using 'randtests'
#Wald-Wolfowitz runs test of randomness for continuous data
require("randtests")
runGTD<-subset(time, id=="GTD")
runGDELT<-subset(time, id=="GDELT")
runRDWTI<-subset(time, id=="RDWTI")
```

```
bartels.rank.test(runGTD$nbevents)
bartels.rank.test(runGDELT$nbevents)
bartels.rank.test(runRDWTI$nbevents)
bartels.rank.test(runGTD$nbdeath)
bartels.rank.test(runRDWTI$nbdeath)
#END ##############################
```

B.3 Global and Local Spatial Autocorrelation

This Appendix provides the codes which have been generated to estimate the global (Moran's I) and local (Getis and Ord $G^*_{\mathbf{s}_i}$) autocorrelation on the density of terrorist attacks in the study area.

```
#The goal of this process is to expore the data using the dataframe ALL,
     which contains data of DTV, GTD, GDELT, and RDWTI
#It uses All.csv, the output of All_data.R, which should be executed
    prior this operation process.
#initialisation
setwd("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/PhD_R/
    ALL")
require("lattice")
require("sp")
require("RgoogleMaps")
require("rgdal")
require("ggplot2")
require("rgl")
require("vcd")
require("foreign")
require("plyr")
require("pwr")
require("rgdal")
require("maptools")
require("ape")
require("geosphere")
require("spdep")
require("GISTools")
ALL<-read.csv("ALL.csv", header= TRUE, dec=".", na.strings=c("","NA"))
world.map <- readShapeSpatial("C:/Users/apython/Documents/Andre/
    Education/StAndrews/PhD/PhD_ArcGIS/World_map/world.shp")
proj4string(world.map)<-CRS("+proj=longlat +datum=WGS84 +no_defs+towgs84
    =0,0,0+ellps=WGS84")
```

```
studyarea <- world.map[world.map$F_AREA > 9000000000,]#delete small
    polygons
#looking at events from 2002 to 2009 (common time period)
A <- subset(ALL, year > 2001 & year < 2010)#we create a subset 2002-2009
     in order to be able to compare all databases on a same temporal
    framework
GTD1<-subset(A, id=="GTD")
RDWTI1<-subset(A, id=="RDWTI")
GDELT1<-subset(A, id=="GDELT")
#Global statistics:Moran's I on the number of terrorist attacks per
    location ("density")
#In order to calculate Moran's I on the density of terrorist attacks (
    based on terrorist events only so it does not take into account
    observations with no terrorist attacks)
#First: creating anew variable which counts the number of events with
    similar latitude and longitude
GTDdens<-ddply(GTD1,.(latitude,longitude),nrow)
GDELTdens<-ddply(GDELT1,.(latitude,longitude),nrow)
RDWTIdens<-ddply(RDWTI1,.(latitude,longitude),nrow)
#compute spatial weight and Moran's I for GTD1 (inverse distance method)
#distance on sphere (Haversine method)
GTDpt<-cbind(GTDdens$longitude, GTDdens$latitude)
GTDd <- distm(GTDpt, GTDpt, fun=distHaversine)#provides distance in
    meter.
GTDd.inv <- 1/GTDd
diag(GTDd.inv) <- 0
wGTD<-GTDd.inv
#Moran's I for density
IGTD<-Moran.I(GTDdens$V1, wGTD, scaled = TRUE)
#compute spatial weight and Moran's I for RDWTI1
RDWTIpt<-cbind(RDWTIdens$longitude, RDWTIdens$latitude)
RDWTId <- distm(RDWTIpt, RDWTIpt, fun=distHaversine)#provides distance
    in meter.
RDWTId.inv <- 1/RDWTId
diag(RDWTId.inv) <- 0
wRDWTI<-RDWTId.inv
#Moran's I for density
IRDWTI<-Moran.I(RDWTIdens$V1, wRDWTI, scaled = TRUE)
#compute spatial weight and Moran's I for GDELT1
GDELTpt<-cbind(GDELTdens$longitude, GDELTdens$latitude)
```

```
GDELTd <- distm(GDELTpt, GDELTpt, fun=distHaversine)#provides distance
    in meter.
GDELTd.inv <- 1/GDELTd
diag(GDELTd.inv) <- 0
wGDELT<-GDELTd.inv
#Moran's I for density
IGDELT<-Moran.I(GDELTdens$V1, wGDELT, scaled = TRUE)
IGTD;IRDWTI;IGDELT
#Second approach: threshold distance: disregarding points further than
    1000 km with a binary distance matrix
GTDd2<- (GTDd > 0 & GTDd <= 1000000)
I2GTD<-Moran.I(GTDdens$V1, GTDd2)
RDWTId2<- (RDWTId > 0 & RDWTId <= 1000000)
I2RDWTI<-Moran.I(RDWTIdens$V1, RDWTId2)
GDELTd2<- (GDELTd > 0 & GDELTd <= 1000000)
I2GDELT<-Moran.I(GDELTdens$V1, GDELTd2)
#Moran's I for density with limit distance of 1000 km
I2GTD;I2RDWTI;I2GDELT

#robustness test
# #not reported in thesis (results are not affected)
# Moran's I for density with limit distance of 500 km
# GTDd3<- (GTDd > 0 & GTDd <= 500000)
# I3GTD<-Moran.I(GTDdens$V1, GTDd3)
# RDWTId3<- (RDWTId > 0 & RDWTId <= 500000)
# I3RDWTI<-Moran.I(RDWTIdens$V1, RDWTId3)
# GDELTd3<- (GDELTd > 0 & GDELTd <= 500000)
# I3GDELT<-Moran.I(GDELTdens$V1, GDELTd3)
# Moran's I for density with limit distance of 200 km
# I3GTD;I3RDWTI;I3GDELT
# GTDd4<- (GTDd > 0 & GTDd <= 200000)
# I4GTD<-Moran.I(GTDdens$V1, GTDd4)
# RDWTId4<- (RDWTId > 0 & RDWTId <= 200000)
# I4RDWTI<-Moran.I(RDWTIdens$V1, RDWTId4)
# GDELTd4<- (GDELTd > 0 & GDELTd <= 200000)
# I4GDELT<-Moran.I(GDELTdens$V1, GDELTd4)
#I4GTD;I4RDWTI;I4GDELT

#Local statistics: local Geary index G*i based on PRIO unit of
    observation
```

```
#first intersect PRIO with the study area
#read PRIOworld which is the intersection of PRIO with the studyarea +
    area calculated in QGIS
PRIO <- readShapeSpatial("C:/Users/apython/Documents/Andre/Education/
    StAndrews/PhD/PhD_ArcGIS/World␣map/PRIOworld.shp")
proj4string(PRIO)<-CRS("+proj=longlat␣+datum=WGS84␣+no_defs+towgs84
    =0,0,0+ellps=WGS84")
PRIO <- PRIO[PRIO@data$area > 0.025,]#delete small polygons
#local Geary index G*i for GDELT
#create spatial points from GDELT
GDELTpts<-SpatialPoints(cbind(GDELT1$longitude,GDELT1$latitude))
proj4string(GDELTpts)<-CRS("+proj=longlat␣+datum=WGS84␣+no_defs+towgs84
    =0,0,0+ellps=WGS84")
#keep PRIO polygons that intersect with GDELT points
PRIOGDELT<-PRIO[GDELTpts,]
#calculate nb. points in each polygon
GDELTPRIO<-poly.counts(GDELTpts, PRIOGDELT)
#add the new variable to spatial polygon PRIOGDELT
PRIOGDELT$nb<-GDELTPRIO
#create and add density variable to spatial polygon PRIOGDELT based on
    area of polygon
PRIOGDELT$density<-PRIOGDELT$nb/PRIOGDELT$area
#transform as a dataframe
GDELTdf<-as.data.frame(PRIOGDELT)
#extract coordinates GDELT in PRIO
GDELTxy <- cbind(GDELTdf$xcoord, GDELTdf$ycoord)
#create first neighbour to obtain max distance (see Bivand Jan 2015)
GDELTnb <- knn2nb(knearneigh(GDELTxy, k = 1))
#find min distance to have one neighbour
dsts <- unlist(nbdists(GDELTnb, GDELTxy))
maxGDELT<-max(dsts)#max:31.38 (dec.degree) so take max of GDELT
#calculate distance-based neighbour based on 1 * min distance to have
    one neighbour (in order that it works for GDELT and GDELT as well)
GDELTnb <- dnearneigh(GDELTxy, d1 = 0, d2 = 1 * maxGDELT)
#calculate the localG
GDELTw<-nb2listw(include.self(GDELTnb))#include.self to obtain G*
Gstar_GDELT<-localG(GDELTdf$density, GDELTw, zero.policy=NULL, spChk=
    NULL)
#positive values indicate clustering of high values and negative are
    clustering of low values (here the density)
```

```
#Under the null hypothesis that there is no association, the expectation
    value is 0, the variance is 1. If the underlying data are normally
    distributed, we can consider their values as standard variates.

#local Geary index G*i for GTD
#create spatial points from GTD
GTDpts<-SpatialPoints(cbind(GTD1$longitude,GTD1$latitude))
proj4string(GTDpts)<-CRS("+proj=longlat +datum=WGS84 +no_defs+towgs84
    =0,0,0+ellps=WGS84")
#keep PRIO polygons that intersect with GTD points
PRIOGTD<-PRIO[GTDpts,]
#calculate nb. points in each polygon
GTDPRIO<-poly.counts(GTDpts, PRIOGTD)
#add the new variable to spatial polygon PRIOGTD
PRIOGTD$nb<-GTDPRIO
#create and add density variable to spatial polygon PRIOGTD based on
    area of polygon
PRIOGTD$density<-PRIOGTD$nb/PRIOGTD$area
#transform as a dataframe
GTDdf<-as.data.frame(PRIOGTD)
#extract coordinates GTD in PRIO
GTDxy <- cbind(GTDdf$xcoord, GTDdf$ycoord)
#create first neighbour to obtain max distance (see Bivand Jan 2015)
GTDnb <- knn2nb(knearneigh(GTDxy, k = 1))
#find min distance to have one neighbour
dsts <- unlist(nbdists(GTDnb, GTDxy))
maxGTD<-max(dsts)
if(maxGDELT > maxGTD) maxGTD <- maxGDELT
#calculate distance-based neighbour based on 1 * min distance to have
    one neighbour (in order that it works for RDWTi and GDELT as well)
GTDnb <- dnearneigh(GTDxy, d1 = 0, d2 = 1 * maxGTD)
#calculate the localG
GTDw<-nb2listw(include.self(GTDnb))#include.self to obtain G*
Gstar_GTD<-localG(GTDdf$density, GTDw, zero.policy=NULL, spChk=NULL)

#local Geary index G*i for RDWTI
#create spatial points from RDWTI
RDWTIpts<-SpatialPoints(cbind(RDWTI1$longitude,RDWTI1$latitude))
proj4string(RDWTIpts)<-CRS("+proj=longlat +datum=WGS84 +no_defs+towgs84
    =0,0,0+ellps=WGS84")
```

```
#keep PRIO polygons that intersect with RDWTI points
PRIORDWTI<-PRIO[RDWTIpts,]
#calculate nb. points in each polygon
RDWTIPRIO<-poly.counts(RDWTIpts, PRIORDWTI)
#add the new variable to spatial polygon PRIORDWTI
PRIORDWTI$nb<-RDWTIPRIO
#create and add density variable to spatial polygon PRIORDWTI based on
    area of polygon
PRIORDWTI$density<-PRIORDWTI$nb/PRIORDWTI$area
#transform as a dataframe
RDWTIdf<-as.data.frame(PRIORDWTI)
#extract coordinates RDWTI in PRIO
RDWTIxy <- cbind(RDWTIdf$xcoord, RDWTIdf$ycoord)
#create first neighbour to obtain max distance (see Bivand Jan 2015)
RDWTInb <- knn2nb(knearneigh(RDWTIxy, k = 1))
#find min distance to have one neighbour
dsts <- unlist(nbdists(RDWTInb, RDWTIxy))
maxRDWTI<-max(dsts)#max:23.6 (dec.degree) so take max of GTD
if(maxGDELT > maxRDWTI) maxRDWTI <- maxGDELT
#calculate distance-based neighbour based on 1 * min distance to have
    one neighbour (in order that it works for RDWTi and GDELT as well)
RDWTInb <- dnearneigh(RDWTIxy, d1 = 0, d2 = 1 * maxRDWTI)
#calculate the localG
RDWTIw<-nb2listw(include.self(RDWTInb))#include.self to obtain G*
Gstar_RDWTI<-localG(RDWTIdf$density, RDWTIw, zero.policy=NULL, spChk=
    NULL)
#z-score outside +/-1.96 signify cluster with 5% alpha
#z-score outside +/-2.576 signify cluster with 1% alpha

#Getis tests
#add G-star values to data.frame
RDWTIdf$Gstar<-matrix(Gstar_RDWTI)
GDELTdf$Gstar<-matrix(Gstar_GDELT)
GTDdf$Gstar<-matrix(Gstar_GTD)
#extract only if values are above or below critical values with alpha=5%
     and 1%
RDWTIgsH <- RDWTIdf[RDWTIdf$Gstar>2.576,]
GTDgsH   <- GTDdf[GTDdf$Gstar>2.576,]
GDELTgsH <- GDELTdf[GDELTdf$Gstar>2.576,]
 GTDgsl   <- GTDdf[which(GTDdf$Gstar<-1.96 & GTDdf$Gstar>-2.576), ]
```

```
#plot
plot(studyarea)
points(GDELTgsH$xcoord,GDELTgsH$ycoord,cex=0.7,pch=3,col="green")
points(RDWTIgsH$xcoord,RDWTIgsH$ycoord,cex=0.7,pch=2,col="red")
points(GTDgsH$xcoord,GTDgsH$ycoord,cex=0.7,pch=1,col="black")
#look at min/max latitude/longitude of high value clusters
summary(GDELTgsH$xcoord);summary(RDWTIgsH$xcoord);summary(GTDgsH$xcoord)
summary(GTDgsH$ycoord);summary(RDWTIgsH$ycoord);summary(GDELTgsH$ycoord)
#########################################################################
save.image("~/Andre/Education/StAndrews/PhD/PhD_R/ALL/Moran.RData")
#plot for thesis
pdf("cluster.pdf",width=7,height=7)
op <- par(mar = c(4.2,1,0.0,4.4) + 0.2)#mar: bottom,left,top,right
a <- matrix(0,nrow=105,ncol=105)
image(seq(-20,85,by=1),seq(-20,85,by=1),a,col="white",ylab="",xlab="",
    yaxt = "n",xaxt = "n")
plot(studyarea, add=TRUE,xlim=c(-20,85),ylim=c(-20,85),yaxt = "n")
axis(4, col = "black", lty = 1, lwd = 1)
axis(1, col = "black", lty = 1, lwd = 1)
points(GDELTgsH$xcoord,GDELTgsH$ycoord,cex=0.7,pch=3,col="green")
points(RDWTIgsH$xcoord,RDWTIgsH$ycoord,cex=0.7,pch=2,col="red")
points(GTDgsH$xcoord,GTDgsH$ycoord,cex=0.7,pch=1,col="black")
mtext("latitude", side = 4, line = 3, cex = par("cex.lab"))
mtext("longitude", side = 1, line = 3, cex = par("cex.lab"))
par(op)
dev.off()
```

B.4 Point Process: First-Order Spatial Statistics

This code generates first-order spatial statistics in R.

```
#The goal of this process is to generate first order measures based on
    databases on terrorism
#1: initialisation
setwd("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/PhD_R")
require(rgeos)
require(mapdata)
require(maptools)
require(rgdal)
```

```
require(spatstat)
require(rgdal)
ALL<-read.csv("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/All/ALL.csv", header= TRUE, dec=".", na.strings=c("","NA"))
#Llinpal col
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal/Llinpal
    .RData")#load custom palette
#STUDY AREA & OBSERVATION WINDOW
#It reads an OGR data source and layer into a suitable Spatial vector
    object. This is the aggregate map of the study area (simplified
    polygon).
world.map <- readShapeSpatial("C:/Users/apython/Documents/Andre/
    Education/StAndrews/PhD/PhD_ArcGIS/World␣map/world.shp")
proj4string(world.map)<-CRS("+proj=longlat␣+datum=WGS84␣+no_defs+towgs84
    =0,0,0+ellps=WGS84")
studyarea <- world.map[world.map$F_AREA > 9000000000,]#delete small
    polygons
rectang<- rbind( c(-20,80),c(-40,40))
#change its projection in order to have km as unit
studyarea <- spTransform(studyarea,CRS=CRS("+proj=merc␣+ellps=WGS84␣+
    units=km"))
#Visual check if there is no duplicates vertices before analysis
# spatstat.options(checkpolygons=T)
# #The class "owin" is a way of specifying the obs. window for a point
    pattern
# W.owin<-as.owin(studyarea)# W.owin becomes the obs.window
# #polygon with duplicated vertices to be identified
# library(rgeos)
# gIsValid(studyarea)
# plot(studyarea, col = "grey")
# points(-14773.21050087, 7143.1259033899996, pch = 3, cex = 3, col = "
    red")
# axis(1);axis(2);box()
#delete polygon num50
studyarea <- studyarea[-50,]
#end check: polygon num50 is deleted
W.owin<-as.owin(studyarea)
summary(W.owin)
#TERRORISM DATA PREPARATION
require("foreign")
```

```
#Creating a subset 2002-2009 to compare all databases
#transforming projection: degree to km
coordinates(ALL) <- c("longitude", "latitude")
proj4string(ALL) <- CRS("+proj=longlat")
ALL<-spTransform(ALL, CRS("+proj=merc +ellps=WGS84 +units=km"))
A <- subset(ALL, year > 2001 & year < 2010)
#splittin A into the four original databases
GTD<-subset(A, id=="GTD")
RDWTI<-subset(A, id=="RDWTI")
GDELT<-subset(A, id=="GDELT")
#extracting latitude, longitude
GTDla<-GTD$latitude
GTDlo<-GTD$longitude
RDWTIla<-RDWTI$latitude
RDWTIlo<-RDWTI$longitude
GDELTla<-GDELT$latitude
GDELTlo<-GDELT$longitude
#creating a two-dimensional point pattern object
#(marks could be optionally added)
GTDp<-ppp(GTDlo,GTDla,window=W.owin)
RDWTIp<-ppp(RDWTIlo,RDWTIla,window=W.owin)
GDELTp<-ppp(GDELTlo,GDELTla,window=W.owin)
#keep points inside the study area (delete those not precisely
    geolocalised)
GTDp<-GTDp[W.owin]
RDWTIp<-RDWTIp[W.owin]
GDELTp<-GDELTp[W.owin]
#plot point patterns (visual check)
#   plot(GTDp, pch=20,cex=0.3)
#   plot(RDWTIp, pch=20,cex=0.3)
#   plot(GDELTp, pch=20,cex=0.3)

#FIRST-ORDER CALCULATION
summary(GTDp)
summary(RDWTIp)
summary(GDELTp)
#window= 325.709 mio km2

#OPTIONAL (not done here)
# #scale the ppp in order to obtain PRIO-GRID unit (one cell~55kmX55km)
```

```
# RDWTIp <- rescale(RDWTIp, 55, unitname = "PRIO")#from 1km to 55km
# GDELTp <- rescale(GDELTp, 55, unitname = "PRIO")#from 1km to 55km
# GTDp <- rescale(GTDp, 55, unitname = "PRIO")#from 1km to 55km
d1<-density.ppp(RDWTIp,sigma=150,eps=50)#use this if rescaled sigma=10,
    eps=1
d2<-density.ppp(GDELTp,sigma=150,eps=50)
d3<-density.ppp(GTDp,sigma=150,eps=50)
d1$v<-log(d1$v)#for plotting
d2$v<-log(d2$v)#for plotting (use log)
d3$v<-log(d3$v)#for plotting (use log)
#compute density with other bandwidth (sigma change)
d1b<-density.ppp(RDWTIp,sigma=300,eps=50)
d2b<-density.ppp(GDELTp,sigma=300,eps=50)
d3b<-density.ppp(GTDp,sigma=300,eps=50)
d1b$v<-log(d1b$v)#for plotting
d2b$v<-log(d2b$v)#for plotting (use log)
d3b$v<-log(d3b$v)#for plotting (use log)

#find min/max of all density maps
mind<-mind<-min(min(d3$v,na.rm=TRUE),min(d2$v,na.rm=TRUE),min(d1$v,na.rm
    =TRUE),
          min(d3b$v,na.rm=TRUE),min(d2b$v,na.rm=TRUE),min(d1b$v,na.rm=
              TRUE))
maxd<-max(max(d3$v,na.rm=TRUE),max(d2$v,na.rm=TRUE),max(d1$v,na.rm=TRUE)
    ,
          max(d3b$v,na.rm=TRUE),max(d2b$v,na.rm=TRUE),max(d1b$v,na.rm=
              TRUE))
require(fields)
library("colorspace")
breaks<-seq(mind,maxd+2,by=2)
n<-length(breaks)-1
dev.off()
plot.new()
mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    revision/revisedPhDthesis/Chapter4/Figs/PDF/DensityRDWTI.pdf")
pdf(file=mypath,onefile=FALSE,width=21,height=8,paper="special")
par(cex=3,mar=c(0,0,0,2) + 0.1)#mar: c(bottom,left,top,right)
image(d1,col = rev(heat_hcl(n,c = c(300, 0),power = c(35, 8))),main="",
    zlim=c(mind,maxd),breaks=breaks)
plot(studyarea,lwd=1,add=TRUE)
```

```
dev.off()
#plot
dev.off()
plot.new()
mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    revision/revisedPhDthesis/Chapter4/Figs/PDF/DensityRDWTI2.pdf")
pdf(file=mypath,onefile=FALSE,width=21,height=8,paper="special")
par(cex=3,mar=c(0,0,0,2) + 0.1)#mar: c(bottom,left,top,right)
#image(d1,col="black",main="",axes=FALSE)
image(d1b,col = rev(heat_hcl(n,c = c(300, 0),power = c(35, 8))),main="",
    zlim=c(mind,maxd),breaks=breaks)
plot(studyarea,lwd=1,add=TRUE)
dev.off()
#plot
dev.off()
plot.new()
mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    revision/revisedPhDthesis/Chapter4/Figs/PDF/DensityGDELT.pdf")
pdf(file=mypath,onefile=FALSE,width=21,height=8,paper="special")
par(cex=3,mar=c(0,0,0,2) + 0.1)#mar: c(bottom,left,top,right)
image(d2,col = rev(heat_hcl(n,c = c(300, 0),power = c(35, 8))),main="",
    zlim=c(mind,maxd),breaks=breaks)
plot(studyarea,lwd=1,add=TRUE)
dev.off()
#plot
dev.off()
plot.new()
mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    revision/revisedPhDthesis/Chapter4/Figs/PDF/DensityGDELT2.pdf")
pdf(file=mypath,onefile=FALSE,width=21,height=8,paper="special")
par(cex=3,mar=c(0,0,0,2) + 0.1)#mar: c(bottom,left,top,right)
image(d2b,col = rev(heat_hcl(n,c = c(300, 0),power = c(35, 8))),main="",
    zlim=c(mind,maxd),breaks=breaks)
plot(studyarea,lwd=1,add=TRUE)
dev.off()
#plot
dev.off()
plot.new()
mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    revision/revisedPhDthesis/Chapter4/Figs/PDF/DensityGTD.pdf")
```

```
pdf(file=mypath,onefile=FALSE,width=21,height=8,paper="special")
par(cex=3,mar=c(0,0,0,2) + 0.1)#mar: c(bottom,left,top,right)
image(d3,col = rev(heat_hcl(n,c = c(300, 0),power = c(35, 8))),main="",
    zlim=c(mind,maxd),breaks=breaks)
plot(studyarea,lwd=1,add=TRUE)
dev.off()
#plot
dev.off()
plot.new()
mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    revision/revisedPhDthesis/Chapter4/Figs/PDF/DensityGTD2.pdf")
pdf(file=mypath,onefile=FALSE,width=21,height=8,paper="special")
par(cex=3,mar=c(0,0,0,2) + 0.1)#mar: c(bottom,left,top,right)
image(d3b,col = rev(heat_hcl(n,c = c(300, 0),power = c(35, 8))),main="",
    zlim=c(mind,maxd),breaks=breaks)
plot(studyarea,lwd=1,add=TRUE)
dev.off();dev.off()
#plot
```

B.5 Point Process: Second-Order Spatial Statistics

This code generates the *pcf* function and produces plots in R. However, it requires the output generated in Appendix B.4.

```
#The goal of this process is to generate second order measures based on
    databases on terrorism
#the code is run in the server flipper (high memory is required to
    calculate pcf in particular)
#1: initialisation
setwd("/home/andre/R/secondorder")#flipper home file
#already pre-installed
library(spatstat,lib.loc="/home/andre/R/x86_64-redhat-linux-gnu-library/
    3.1")
library(foreign,lib.loc="/home/andre/R/x86_64-redhat-linux-gnu-library/
    3.1")
library(maptools,lib.loc="/home/andre/R/x86_64-redhat-linux-gnu-library/
    3.1")
library(sp,lib.loc="/home/andre/R/x86_64-redhat-linux-gnu-library/3.1")
#require(RandomFields)#does not work
```

```
load("/home/andre/R/SPDE/SPDE.RData")#contains study area, mesh,
    covariates
ALL<-read.csv("ALL.csv", header= TRUE, dec=".", na.strings=c("","NA"))
load("/home/andre/R/secondorder/W.owin.RData")#contains study area with
    transformed projection

#STUDY AREA & OBSERVATION WINDOW
proj4string(world.map)<-CRS("+proj=longlat +datum=WGS84 +no_defs+towgs84
    =0,0,0+ellps=WGS84")
studyarea <- world.map[world.map$F_AREA > 9000000000,]#delete small
    polygons
#Check if there is no duplicates vertices before analysis
# spatstat.options(checkpolygons=T)
# #The class "owin" is a way of specifying the obs. window for a point
    pattern
# W.owin<-as.owin(studyarea)# W.owin becomes the obs.window
# #polygon with duplicated vertices to be identified
# library(rgeos)
# gIsValid(studyarea)
# plot(studyarea, col = "grey")
# points(-14773.21050087, 7143.1259033899996, pch = 3, cex = 3, col = "
    red")
# axis(1);axis(2);box()
#delete polygon num50
studyarea <- studyarea[-50,]
# end check: polygon num50 is deleted#
studyarea <- spTransform(studyarea, CRS=CRS('+proj=merc +ellps=WGS84 +
    units=km'))
spatstat.options(checkpolygons=T)
#create a buffer zone around the coastlines in order to include all
    accurately geolocalised points in the process
studyarea <-gBuffer(studyarea,width=50)
#The class "owin" is a way of specifying the obs. window for a point
    pattern
W.owin<-as.owin(studyarea)# W.owin becomes the obs.window

#TERRORISM DATA PREPARATION
#we create a subset 2002-2009 to compare all databases
#transforming projection: degree to km
coordinates(ALL) <- c("longitude", "latitude")
```

```
proj4string(ALL) <- CRS("+proj=longlat")
ALL<-spTransform(ALL, CRS("+proj=merc +ellps=WGS84 +units=km"))
A <- subset(ALL, year > 2001 & year < 2010)
#we split A into the four original databases
GTD<-subset(A, id=="GTD")
RDWTI<-subset(A, id=="RDWTI")
GDELT<-subset(A, id=="GDELT")
#Extracting latitude, longitude
GTDla<-GTD$latitude
GTDlo<-GTD$longitude
RDWTIla<-RDWTI$latitude
RDWTIlo<-RDWTI$longitude
GDELTla<-GDELT$latitude
GDELTlo<-GDELT$longitude
#Creating a two-dimensional point pattern object (marks could be
    optionally added)
GTDp<-ppp(GTDlo,GTDla,window=W.owin)
RDWTIp<-ppp(RDWTIlo,RDWTIla,window=W.owin)
GDELTp<-ppp(GDELTlo,GDELTla,window=W.owin)
#some points are rejected because there are outside the study area (not
    well localised or on too small islands which are not taken into
    account)
#visualising points outside the window
# isin<-inside.owin(x=GTD@coords[,1],y=GTD@coords[,2],w=W.owin)
# point_in <- GTD[isin,]
# point_out <- GTD[!isin,]
# plot(W.owin)
# #points(GTDp)
# points(point_out,col="red",cex = 1 )
# points(point_in,col="green",cex = 1 )
# visualising points outside the window

#K-function (not reported - less informative than the pcf)
# KGDELT<-Kest(GDELTp, correction="border")#if nb.points is large ->
    faster
# KGTD<-Kest(GTDp, correction="border")#if nb.points is large ->faster
# KRDWTI<-Kest(RDWTIp, correction="border")#if nb.points is large ->
    faster
#L-function
```

```
# LGDELT<-Lest(GDELTp, correction="border")#if nb.points is large ->
    faster
# LGTD<-Lest(GTDp, correction="border")#if nb.points is large ->faster
# LRDWTI<-Lest(RDWTIp, correction="border")#if nb.points is large ->
    faster

#pcf-function (zoom from 0 to 5000 km)
#creating vector r in which pcf is evaluated
r1<-as.vector(seq(0, 5000, 500))
#creating the function pcf
GDELTpcf1<-pcf.ppp(GDELTp, bw= 1, r=r1, correction="translate")
GTDpcf1<-pcf.ppp(GTDp, bw= 1, r=r1, correction="translate")
RDWTIpcf1<-pcf.ppp(RDWTIp, bw= 1, r=r1, correction="translate")
#pcf-function (zoom from 0 to 1000 km)
#creating vector r in which pcf is evaluated
r2<-as.vector(seq(0, 1000, 10))
#creating the function pcf
GDELTpcf2<-pcf.ppp(GDELTp, bw= 1,r=r2, correction="translate")
GTDpcf2<-pcf.ppp(GTDp, bw= 1, r=r2, correction="translate")
RDWTIpcf2<-pcf.ppp(RDWTIp, bw= 1, r=r2, correction="translate")
#pcf-function (zoom from 0 to 100 km)
r3<-as.vector(seq(0, 100, 5))
#creating the function pcf
GDELTpcf3<-pcf.ppp(GDELTp, bw= 1,r=r3, correction="translate")
GTDpcf3<-pcf.ppp(GTDp, bw= 1, r=r3, correction="translate")
RDWTIpcf3<-pcf.ppp(RDWTIp, bw= 1, r=r3, correction="translate")
#optional 95% confidence intervals for K, L, pcf (only pcf is reported)
#time consuming! (total for K,L, pcf: 4.7 hours)
# #K
# KDTVl<-lohboot(DTVp, fun="Kest",nsim=99, confidence=0.95, global=FALSE
    , type=7)
# KGTDl<-lohboot(GTDp, fun="Kest",nsim=99, confidence=0.95, global=FALSE
    , type=7)
# KGDELTl<-lohboot(GDELTp, fun="Kest",nsim=99, confidence=0.95, global=
    FALSE, type=7)
# KRDWTIl<-lohboot(RDWTIp, fun="Kest",nsim=99, confidence=0.95, global=
    FALSE, type=7)
# #L
# LDTVl<-lohboot(DTVp, fun="Lest",nsim=99, confidence=0.95, global=FALSE
    , type=7)
```

```
# LGTDl<-lohboot(GTDp, fun="Lest",nsim=99, confidence=0.95, global=FALSE
    , type=7)
# LGDELTl<-lohboot(GDELTp, fun="Lest",nsim=99, confidence=0.95, global=
    FALSE, type=7)
# LRDWTIl<-lohboot(RDWTIp, fun="Lest",nsim=99, confidence=0.95, global=
    FALSE, type=7)
#pcf
pcfGTDl<-lohboot(GTDp, fun="pcf",nsim=30, confidence=0.95, global=TRUE,
    type=7)
pcfGDELTl<-lohboot(GDELTp, fun="pcf",nsim=30, confidence=0.95, global=
    TRUE, type=7)
pcfRDWTIl<-lohboot(RDWTIp, fun="pcf",nsim=30, confidence=0.95, global=
    TRUE, type=7)
rm(list=setdiff(ls(), c("pcfGTDl","pcfGDELTl","pcfRDWTIl","GTDp","GDELTp
    ","RDWTIp",
"GDELTpcf3","GTDpcf3","RDWTIpcf3","GDELTpcf2","GTDpcf2","RDWTIpcf2","
    GDELTpcf1","GTDpcf1","RDWTIpcf1")))
#Second-order
#K-function
# KGDELT<-Kest(GDELTp, correction="border")#if nb.points is large ->
    faster
# KGTD<-Kest(GTDp, correction="border")#if nb.points is large ->faster
# KRDWTI<-Kest(RDWTIp, correction="border")#if nb.points is large ->
    faster
# #plot K-function
# cbPalette <- c("#666666","#999999","#CCCCCC")#colors defined for GDELT
    ,GTD,RDWTI respectively.
#
# plot.new()
# par(oma=c(0, 0.2, 0, 0), mar=c(4.1, 4.5, 2.1, 0.5))
# plot(KGDELT, col=c("#666666", "red"),main="K-function", legend=NA,
# cex.title=1.5, cex.axis=1.1, cex.lab=1.3, lwd=c(2,2), xlim=c(0, 3000),
    ylim=c(0,120000000))
# plot(KGTD, col=c("#999999","#999999"), lwd=c(2,NA),xlim=c(0, 3000),
    add=T)
# plot(KRDWTI, col=c("#CCCCCC","#CCCCCC"), lwd=c(2,NA), xlim=c(0, 3000),
    add=T)
# legend("topleft", inset=.05, title="",box.lwd = 0,box.col = "white",bg
    = "white",
```

```
# c("GDELT","GTD","RDWTI"), fill=c("#666666","#999999","#CCCCCC"), horiz
    =TRUE)
# #
# # #L-function
# LGDELT<-Lest(GDELTp, correction="border")#if nb.points is large ->
    faster
# LGTD<-Lest(GTDp, correction="border")#if nb.points is large ->faster
# LRDWTI<-Lest(RDWTIp, correction="border")#if nb.points is large ->
    faster
# # #plot L-function
# plot.new()
# par(oma=c(0, 0.2, 0, 0), mar=c(4.1, 4.5, 2.1, 0.5))
# plot(main="L-function", legend=NA, cex.title=1.5, cex.axis=1.1, cex.
    lab=1.3,
# LGDELT, col=c("#666666", "red"), lwd=c(2,2), xlim=c(0, 3000),ylim=c
    (0,6000))
# plot(LGTD, col=c("#999999","#999999"), lwd=c(2,NA),xlim=c(0, 3000),
    add=T)
# plot(LRDWTI, col=c("#CCCCCC","#CCCCCC"), lwd=c(2,NA), xlim=c(0, 3000),
    add=T)
# legend("topleft", inset=.05, title="",box.lwd = 0,box.col = "white",bg
    = "white",
# c("GDELT","GTD","RDWTI"), fill=c("#666666","#999999","#CCCCCC"), horiz
    =TRUE)

# #plotting pcf-function: r1<-as.vector(seq(0, 5000, 500))
plot.new()
par(oma=c(0, 0.2, 0, 0), mar=c(4.1, 4.5, 2.1, 0.5))
plot(GDELTpcf1$r,GDELTpcf1$trans, col=c("green"), lwd=2,type="l" , ylim=
    c(0,65),ylab="g(r)",xlab="r")
lines(GTDpcf1$r,GTDpcf1$trans, col=c("black"), lwd=2)
lines(RDWTIpcf1$r,RDWTIpcf1$trans, col=c("red"), lwd=2)
lines(GTDpcf1$r,GTDpcf1$theo, col="grey", lwd=2,lty=2 )
legend("topright", inset=.05, title="",box.lwd = 0,box.col = "white",bg
    = "white",
       c("GDELT","GTD","RDWTI"), fill=c("green","black","red"), horiz=
           TRUE)

#plotting pcf-function v.2 zoom medium: r2<-as.vector(seq(0, 1000, 10))
plot.new()
```

```
par(oma=c(0, 0.2, 0, 0), mar=c(4.1, 4.5, 2.1, 0.5))
plot(GDELTpcf2$r,GDELTpcf2$trans, col=c("green"), lwd=2,type="l" , ylim=
    c(0,3000),ylab="g(r)",xlab="r")
lines(GTDpcf2$r,GTDpcf2$trans, col=c("black"), lwd=2)
lines(RDWTIpcf2$r,RDWTIpcf2$trans, col=c("red"), lwd=2)
lines(GTDpcf2$r,GTDpcf2$theo, col="grey", lwd=2,lty=2 )
legend("topright", inset=.05, title="",box.lwd = 0,box.col = "white",bg
    = "white",
       c("GDELT","GTD","RDWTI"), fill=c("green","black","red"), horiz=
           TRUE)
#plotting pcf-function v.3 zoom high: r3<-as.vector(seq(0, 100, 5))
plot.new()
par(oma=c(0, 0.2, 0, 0), mar=c(4.1, 4.5, 2.1, 0.5))
plot(GDELTpcf3$r,GDELTpcf3$trans, col=c("green"), lwd=2,type="l",ylab="g
    (r)",xlab="r",ylim=c(0,3000))
lines(GTDpcf3$r,GTDpcf3$trans, col=c("black"), lwd=2)
lines(RDWTIpcf3$r,RDWTIpcf3$trans, col=c("red"), lwd=2)
lines(GTDpcf3$r,GTDpcf3$theo, col="grey", lwd=2,lty=2 )
legend("topright", inset=.01, title="",box.lwd = 0,box.col = "white",bg
    = "white",
       c("GDELT","GTD","RDWTI"), fill=c("green","black","red"), horiz=
           TRUE)
#plot (reported)
#95% confidence intervals for pcf (only pcf is reported) 99 simulations
#time consuming! (total for K,L, pcf: 4.7 hours)
#plot pcf (zoom) with confidence interval bands
plot.new()
mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    Latex/phd-thesis-template-phd-latex-template-latest-stable/Chapter4/
    Figs/PDF/pcf1.pdf")
pdf(file=mypath,onefile=FALSE,width=20,height=20,paper="special")
par(cex=5,oma=c(0, 0.2, 0, 0), mar=c(4.1, 4.5, 2.1, 0.5))
plot(pcfGDELTl$r,pcfGDELTl$bord,ylim=c(-50,1000), xlim=c(0, 2000),lty=1,
     lwd=6,col=c("NA"),
      type="l",main="", legend=NA, cex.title=1.5, cex.axis=1.1, cex.lab
          =1.3,
      ylab="pcf",xlab="r")
polygon(c(pcfGDELTl$r,rev(pcfGDELTl$r)),c(pcfGDELTl$lo,rev(pcfGDELTl$hi)
    ),col = "grey40", border = "grey40")
lines(pcfGDELTl$r,pcfGDELTl$bord,lwd=6,lty=1,col=c("green"))
```

```
lines(pcfGDELTl$r,pcfGDELTl$theo,lwd=6,lty=2,col=c("black"))
polygon(c(pcfGTDl$r,rev(pcfGTDl$r)),c(pcfGTDl$lo,rev(pcfGTDl$hi)),col =
    "grey40", border = "grey40")
lines(pcfGTDl$r,pcfGTDl$bord,lwd=6,lty=1,col=c("black"))
lines(pcfGTDl$r,pcfGTDl$theo,lwd=6,lty=2,col=c("black"))
polygon(c(pcfRDWTIl$r,rev(pcfRDWTIl$r)),c(pcfRDWTIl$lo,rev(pcfRDWTIl$hi)
    ),col = "grey40", border = "grey40")
lines(pcfRDWTIl$r,pcfRDWTIl$bord,lwd=6,lty=1,col=c("red"))
lines(pcfRDWTIl$r,pcfRDWTIl$theo,lwd=6,lty=2,col=c("black"))
legend("topright", inset=.01,cex=0.8, title="",box.lwd = 0,box.col = "
    white",bg = "white",
        c("GDELT","GTD","RDWTI"), fill=c("green","black","red"), horiz=
            TRUE)
dev.off()
dev.off()
#zoom 0-500 km
plot.new()
mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    Latex/phd-thesis-template-phd-latex-template-latest-stable/Chapter4/
    Figs/PDF/pcf2.pdf")
pdf(file=mypath,onefile=FALSE,width=20,height=20,paper="special")
par(cex=5,oma=c(0, 0.2, 0, 0), mar=c(4.1, 4.5, 2.1, 0.5))
plot(pcfGDELTl$r,pcfGDELTl$bord,ylim=c(-50,1000), xlim=c(0, 500),lty=1,
    lwd=6,col=c("NA"),
      type="l",main="", legend=NA, cex.title=1.5, cex.axis=1.1, cex.lab
          =1.3,
      ylab="pcf",xlab="r")
polygon(c(pcfGDELTl$r,rev(pcfGDELTl$r)),c(pcfGDELTl$lo,rev(pcfGDELTl$hi)
    ),col = "grey40", border = "grey40")
lines(pcfGDELTl$r,pcfGDELTl$bord,lwd=6,lty=1,col=c("green"))
lines(pcfGDELTl$r,pcfGDELTl$theo,lwd=6,lty=2,col=c("black"))
polygon(c(pcfGTDl$r,rev(pcfGTDl$r)),c(pcfGTDl$lo,rev(pcfGTDl$hi)),col =
    "grey40", border = "grey40")
lines(pcfGTDl$r,pcfGTDl$bord,lwd=6,lty=1,col=c("black"))
lines(pcfGTDl$r,pcfGTDl$theo,lwd=6,lty=2,col=c("black"))
polygon(c(pcfRDWTIl$r,rev(pcfRDWTIl$r)),c(pcfRDWTIl$lo,rev(pcfRDWTIl$hi)
    ),col = "grey40", border = "grey40")
lines(pcfRDWTIl$r,pcfRDWTIl$bord,lwd=6,lty=1,col=c("red"))
lines(pcfRDWTIl$r,pcfRDWTIl$theo,lwd=6,lty=2,col=c("black"))
```

```
legend("topright", inset=.01, cex=0.8,title="",box.lwd = 0,box.col = "
    white",bg = "white",
        c("GDELT","GTD","RDWTI"), fill=c("green","black","red"), horiz=
            TRUE)
dev.off()
dev.off()
save.image("/home/andre/R/secondorder/secondorder.RData")
#END#####################################################
```

Appendix C

Modelling Lethal Terrorism in Space and Time

C.1 Study area, Mesh, Terrorism Data and Covariates

This code is used to define the study area, generate the mesh, select GTD events, and extract the covariates.

```
#Creating data used for different models (space) or (spatio-temporal) of
    terrorism based on SPDE/INLA
#including GTD extraction, mesh, covariates, and study area. Time
    selected is 2002-2013 (12 years)
setwd("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/PhD_R/
    SPDE")
require(foreign)
require(sp)
require(RandomFields)
library(INLA)
library(spatstat)
library(fields)
library(maptools)
library(fields)
library(lattice)
library(latticeExtra)
library(spdep)
library(plyr)
#if one does not want to run the entire code, uncommment the following
    line.
```

```
#load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/SPDE.RData")
#1.Import terrorist events worldwide from GTD
GTD<- read.csv("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD
    /PhD_R/GTD/GTDworld.csv")
#OPTIONAL: add random variable to lat/long in order to avoid identical
    locations of the points (in line with spatial accuracy of the
    database)
nGTD<-length(GTD$latitude)#length of vector GTD
#random values created
v<-runif(nGTD,-0.01, 0.01)
w<-runif(nGTD,-0.01, 0.01)
# adding to GTD coordinates
GTD$latitude<-GTD$latitude+v
GTD$longitude<-GTD$longitude+w
# GTD from 2002 to 2013 is called GTD1
GTD1 <- subset(GTD, iyear > 2001 & iyear < 2014)#selecting 12 years of
    observations
#2.Put points onto the unit sphere
loc<-GTD1[,7:8]#select latitude and longitude
loc<-loc*pi/180
lat<-loc[,1]
lon<-loc[,2]
loc = cbind(cos(lat)*cos(lon), cos(lat)*sin(lon), sin(lat))#change as
    spherical coordinates (r,phi,theta)
#3.Generate meshes on the sphere (to be adjusted for better results)
#select study area and delete small polygons
world.map <- readShapeSpatial("C:/Users/apython/Documents/Andre/
    Education/StAndrews/PhD/PhD_ArcGIS/World␣map/world.shp")
proj4string(world.map)<-CRS("+proj=longlat␣+datum=WGS84␣+no_defs+towgs84
    =0,0,0+ellps=WGS84")
studyarea <- world.map[world.map$F_AREA > 9000000000,]#delete small
    polygons
#plot(studyarea)
#in order to have a mesh on a sphere (S2)
bdry<-inla.sp2segment(studyarea)
bdry$loc <- inla.mesh.map(bdry$loc, projection="longlat", inverse=TRUE)

#Mesh
#mesh version to be used for final version of PhD
#high detailed mesh
```

```
mesh<-inla.mesh.2d(boundary=bdry, max.edge=c(3,1000)/180,cutoff=3/180)#
    simple: max.edge=c(6,10000)/180,cutoff=10.5/180)
plot(mesh, rgl=TRUE, draw.segments=FALSE,edge.color=rgb(0.25, 0.25,
    0.25))
interior<-inla.mesh.interior(mesh)
for(i in 1:length(interior)) {
  lines(interior[[i]],col=1,rgl=TRUE,lwd=2)
}

#Covariates#
#population density 2000
#Gridded Population of the World (GPW) 2000 (which has been reworked in
    arcgis in order to put 0 if no values + extent to -90;90 latitude)
#with arcgis "raster calculator" function: Con(IsNull("inputraster),0,"
    inputraster") and processing environment add lat -90 to 90 and lon
    -180 to 180)
#then exported as ASCII with arcgis "sample" function
require(raster)
pop00<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/flipper/glds00ag_0add")
projection(pop00)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
#create lon lat in matrix form
xy<-cbind(GTD1[,8],GTD1[,7])#select the 3rd and 2nd variable of GTD
#extract coordinates from population at the location of the points
    coordinates
#a bilinearl method + buffer zone is used if data is not provided (long
    duration but ensure no NA values)
pop<-extract(pop00,xy,method='bilinear')#extract covariate value at
    point locations
summary(pop)#check if NA:
pop<-as.vector(pop)
sd(pop)#sd
mean(pop)#mean
#replace NA values by 0: not do it-this may cause issues if large cities
     are not provide with values and replaced by 0. Should be avoided by
    using bilateral + buffer zone extract.
#pop[is.na(pop)] <- 0
pop<-scale(pop)#normalising data (mean=0, sd=1) for better INLA process
```

```
#luminosity (spatio-temporal covariate)
# adding luminosity data as spatio-temporal covariate
# satellite night light (which has been reworked in arcgis in order to
    put 0 if no values + extent to -90;90 latitude)
# with arcgis "raster calculator" function: Con(IsNull("inputraster")
    ,0,"inputraster") and processing environment add lat -90 to 90 and
    lon -180 to 180)
# then exported as ASCII with arcgis "sample" function
# require(raster)
lum02<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum02.tif")
projection(lum02)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum03<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum03.tif")
projection(lum03)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum04<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum04.tif")
projection(lum04)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum05<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum05.tif")
projection(lum05)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum06<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum06.tif")
projection(lum06)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum07<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum07.tif")
projection(lum07)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum08<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum08.tif")
projection(lum08)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum09<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum09.tif")
```

```
projection(lum09)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum10<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum10.tif")
projection(lum10)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum11<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum11.tif")
projection(lum11)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum12<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum12.tif")
projection(lum12)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum13<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_ArcGIS/luminosity/F182013.v4c_web.stable_lights.avg_vis.tif")
projection(lum13)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"

#extract luminosity values at locations in space and time defined by GTD
    space-time locations
lum02<-as.data.frame(lum02<-extract(lum02,xy,method='bilinear'))
lum03<-as.data.frame(lum03<-extract(lum03,xy,method='bilinear'))
lum04<-as.data.frame(lum04<-extract(lum04,xy,method='bilinear'))
lum05<-as.data.frame(lum05<-extract(lum05,xy,method='bilinear'))
lum06<-as.data.frame(lum06<-extract(lum06,xy,method='bilinear'))
lum07<-as.data.frame(lum07<-extract(lum07,xy,method='bilinear'))
lum08<-as.data.frame(lum08<-extract(lum08,xy,method='bilinear'))
lum09<-as.data.frame(lum09<-extract(lum09,xy,method='bilinear'))
lum10<-as.data.frame(lum10<-extract(lum10,xy,method='bilinear'))
lum11<-as.data.frame(lum11<-extract(lum11,xy,method='bilinear'))
lum12<-as.data.frame(lum12<-extract(lum12,xy,method='bilinear'))
lum13<-as.data.frame(lum13<-extract(lum13,xy,method='bilinear'))
# intercalibration based on Elvidge2013
# Process: 1) calculate: Y = C0 + C1X + C2X^2 // 2) Values > 63 are
    truncated at 63 // 3) values = 0 stay zero.
# Sat   Year      C0          C1       C2
# F15   2002     0.0491   0.9568   0.0010
# F15    2003    0.2217   1.5122   -0.0080
# F16   2004     0.2853   1.1955   -0.0034
```

```
# F16    2005    -0.0001  1.4159   -0.0063
# F16    2006    0.1065   1.1371   -0.0016
# F16    2007    0.6394   0.9114   0.0014
# F16  2008      0.5564   0.9931   0.0000
# F16    2009    0.9492   1.0683   -0.0016
# F18  2010      2.3430   0.5102   0.0065
# F18    2011    1.8956   0.7345   0.0030
# F18    2012    1.8750   0.6203   0.0052
f02 <- function(x){ifelse(x>0, 0.0491+0.9568*x+0.0010*x^2, 0)}#if the
    values are above 0, execute intercalibration
f03 <- function(x){ifelse(x>0, 0.2217+1.5122*x+0.0010*x^2, 0)}
f04 <- function(x){ifelse(x>0, 0.2853+1.1955*x+0.0010*x^2, 0)}
f05 <- function(x){ifelse(x>0, -0.0001+1.4159*x+0.0010*x^2, 0)}
f06 <- function(x){ifelse(x>0, 0.1065+1.1371*x+0.0010*x^2, 0)}
f07 <- function(x){ifelse(x>0, 0.6394+0.9114*x+0.0010*x^2, 0)}
f08 <- function(x){ifelse(x>0, 0.5564+0.9931*x+0.0010*x^2, 0)}
f09 <- function(x){ifelse(x>0, 0.9492+1.0683*x+0.0010*x^2, 0)}
f10 <- function(x){ifelse(x>0, 2.3430+0.5102*x+0.0010*x^2, 0)}
f11 <- function(x){ifelse(x>0, 1.8956+0.7345*x+0.0010*x^2, 0)}
f12 <- function(x){ifelse(x>0, 1.8750+0.6203*x+0.0010*x^2, 0)}
f13 <- function(x){ifelse(x>0, 1.8750+0.6203*x+0.0010*x^2, 0)}

lum02<-as.data.frame(sapply(lum02,f02))
lum02[lum02 > 63] <- 63 #truncate if values are higher than 63
lum03<-as.data.frame(sapply(lum03,f03))
lum03[lum03 > 63] <- 63 #truncate if values are higher than 63
lum04<-as.data.frame(sapply(lum04,f04))
lum04[lum04 > 63] <- 63 #truncate if values are higher than 63
lum05<-as.data.frame(sapply(lum05,f05))
lum05[lum05 > 63] <- 63 #truncate if values are higher than 63
lum06<-as.data.frame(sapply(lum06,f06))
lum06[lum06 > 63] <- 63 #truncate if values are higher than 63
lum07<-as.data.frame(sapply(lum07,f07))
lum07[lum07 > 63] <- 63 #truncate if values are higher than 63
lum08<-as.data.frame(sapply(lum08,f08))
lum08[lum08 > 63] <- 63 #truncate if values are higher than 63
lum09<-as.data.frame(sapply(lum09,f09))
lum09[lum09 > 63] <- 63 #truncate if values are higher than 63
lum10<-as.data.frame(sapply(lum10,f10))
lum10[lum10 > 63] <- 63 #truncate if values are higher than 63
```

```
lum11<-as.data.frame(sapply(lum11,f11))
lum11[lum11 > 63] <- 63 #truncate if values are higher than 63
lum12<-as.data.frame(sapply(lum12,f12))
lum12[lum12 > 63] <- 63 #truncate if values are higher than 63
lum13<-as.data.frame(sapply(lum13,f13))
lum13[lum13 > 63] <- 63 #truncate if values are higher than 63
#create time variable for lumYY data.frame
lum02$time<-2002;lum08$time<-2008
lum03$time<-2003;lum09$time<-2009
lum04$time<-2004;lum10$time<-2010
lum05$time<-2005;lum11$time<-2011
lum06$time<-2006;lum12$time<-2012
lum07$time<-2007;lum13$time<-2013
#bind luminosity and coordinates values together
lum02<-cbind(xy,lum02);lum08<-cbind(xy,lum08)
lum03<-cbind(xy,lum03);lum09<-cbind(xy,lum09)
lum04<-cbind(xy,lum04);lum10<-cbind(xy,lum10)
lum05<-cbind(xy,lum05);lum11<-cbind(xy,lum11)
lum06<-cbind(xy,lum06);lum12<-cbind(xy,lum12)
lum07<-cbind(xy,lum07);lum13<-cbind(xy,lum13)
#renaming variables
names(lum02) <- c("lon","lat","lum", "time");names(lum09) <- c("lon","
    lat","lum", "time")
names(lum03) <- c("lon","lat","lum", "time");names(lum10) <- c("lon","
    lat","lum", "time")
names(lum04) <- c("lon","lat","lum", "time");names(lum11) <- c("lon","
    lat","lum", "time")
names(lum05) <- c("lon","lat","lum", "time");names(lum12) <- c("lon","
    lat","lum", "time")
names(lum06) <- c("lon","lat","lum", "time");names(lum13) <- c("lon","
    lat","lum", "time")
names(lum07) <- c("lon","lat","lum", "time")
names(lum08) <- c("lon","lat","lum", "time")
#merging vertically all databases: lat,lon,lum,time (4 columns)
lum<-rbind(lum02,lum03,lum04,lum05,lum06,lum07,lum08,lum09,lum10,lum11,
    lum12,lum13)
lum<-as.data.frame(lum)
sd(lum$lum)#sd
mean(lum$lum)#mean
```

```
# #normalising data (for better INLA process): luminosity mean=0, sd=1
    and rearrange data frame
lum$lum<-scale(lum$lum)
lum <- lum[c(4,1,2,3)]#order columns as: time, lon, lat, lum
lum <- lum[order(lum$time,lum$lon,lum$lat),]
#extract lat, lon, and year from GTD and create a database which
    represent the response vector
GTDresp<-GTD1[,c(4,7,8)]#extract year,lat, long
#add id unique values in GTDresp to sort it later
id <- c(1:nrow(GTDresp))
GTDresp <- cbind(id=id, GTDresp)
names(GTDresp)<-c("id","time","lat","lon")
GTDresp$time<-as.numeric(GTDresp$time)
GTDresp <- GTDresp[c(1,2,4,3)]#order columns as: id, time, lon, lat
#match luminosity with the response vector based on GTD locations
GTDlum <- merge(GTDresp,lum,by.x=c("time","lon","lat"), by.y=c("time","
    lon","lat"),all.x=TRUE)
GTDlum <- subset(GTDlum, !duplicated(GTDlum[,4])) #remove duplicate
    observations from id (unique cannot be used since we can have
    identical lat,lon,lum!)
GTDlum<- GTDlum[order(GTDlum$id),]
#putting lum as a vector
lum<-as.vector(GTDlum$lum)
#test plot to check distance values
#create dataframe
# GTD1lum<-cbind(GTD1,lum)
# lum08<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/
    PhD/PhD_R/luminosity/lum08.tif")
# projection(lum08)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
# plot.new()
# rbPal <- colorRampPalette(c('blue','green','yellow','orange','red','
    brown'))
# GTD1lum$Col <- rbPal(6)[as.numeric(cut(GTD1lum$lum,breaks = 6))]
# par(mfrow=c(2,1))
# plot(GTD1lum$longitude,GTD1lum$latitude,pch = 20,cex=0.3,col = GTD1lum
    $Col)
# plot(world,add=T)
# plot(lum08,ylim=c(-45,65),xlim=c(-180,180))
# plot(world,add=T)
```

```
#END luminosity (spatio-temporal covariate)

#travel time covariate (spatial not temporal)
ttime <- raster("C:/Users/apython/Documents/Andre/Education/StAndrews/
    PhD/PhD_ArcGIS/traveltime/access_50k/acc_50k")
projection(ttime)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
#create lon lat in matrix form
xy<-cbind(GTD1[,8],GTD1[,7])#select the 3rd and 2nd variable of GTD
#extract coordinates from population at the location of the points
    coordinates
tt<-extract(ttime,xy,method='bilinear')#extract covariate value at point
     locations
tt<-as.vector(tt)
#replace NA values by 0
tt[is.na(tt)] <- 0
tt<-scale(tt)#normalising data (mean=0, sd=1) for better INLA process
#test plot to check distance values
#create dataframe
# GTDtt<-cbind(GTD1,tt)
# plot.new()
# rbPal <- colorRampPalette(c('blue','green','yellow','orange','red','
    brown'))
# GTDtt$Col <- rbPal(6)[as.numeric(cut(GTDtt$tt,breaks = 6))]
# plot(GTDtt$longitude,GTDtt$latitude,pch = 20,cex=0.3,col = GTDtt$Col)
# plot(world,add=T)
# plot(ttime,ymin=c(-45,65))
# points(GTD1[,8],GTD1[,7],add=T,cex=0.1)
#END travel time (spatial not temporal)

#Polity IV covariate (country-level + temporal)
#read the original file (saved in .csv) from http://www.systemicpeace.
    org/inscrdata.html
polity<- read.csv("C:/Users/apython/Documents/Andre/Education/StAndrews/
    PhD/PhD_R/PolityIV/p4v2013.csv",header = TRUE)
polity<-subset(polity, select=c("country","year","polity2"))
polity <-subset(polity, year > 2001 & year < 2014)#selecting the same
    time period than GTD
#rename year as iyear for compatibility with GTD
require(reshape)
```

```
polity<-rename(polity, c(year="iyear"))
#add polity IV variable (polity2 has been adjusted for time-series
    analysis) to GTD data
#note that Bahamas, Belize, Iceland, Maldives,West Bank and Gaza Strip,
    are not present in polity
#change some country names in polity before merging with GTD
polity$country <- as.character(polity$country)
polity$country[polity$country == "Bosnia"] <-"Bosnia-Herzegovina"
polity$country[polity$country == "Congo␣Brazzaville"] <-"Congo␣(
    Brazzaville)"
polity$country[polity$country == "Congo␣Kinshasa"] <-"Congo␣(Kinshasa)"
polity$country[polity$country == "Dominican␣Rep"] <-"Dominican␣Republic"
polity$country[polity$country == "Serbia␣and␣Montenegro"] <-"Serbia-
    Montenegro"
#polity$country[polity$country == "Korea North"] <-"North Korea"
polity$country[polity$country == "Korea␣South"] <-"South␣Korea"
polity$country[polity$country == "East␣Timor"] <-"Timor-Leste"
polity$country[polity$country == "UAE"] <-"United␣Arab␣Emirates"
polity$country[polity$country == "Myanmar␣(Burma)"] <-"Myanmar"
#create new observations based on equivalent countries and add to polity
     dataframe
uk<-subset(polity, country == "United␣Kingdom")
uk$country[uk$country == "United␣Kingdom"] <-"Great␣Britain"
polity<-rbind(polity,uk)
ni<-subset(polity, country == "Great␣Britain")
ni$country[ni$country == "Great␣Britain"] <-"Northern␣Ireland"
polity<-rbind(polity,ni)
rm(uk);rm(ni)
ch<-subset(polity, country == "China")
ch$country[ch$country == "China"] <-"Hong␣Kong"
polity<-rbind(polity,ch)
rm(ch)
is<-subset(polity, country == "Israel")
is$country[is$country == "Israel"] <-"West␣Bank␣and␣Gaza␣Strip"
polity<-rbind(polity,is)
rm(is)
cs<-subset(polity, country == "France")
cs$country[cs$country == "France"] <-"Corsica"
polity<-rbind(polity,cs)
rm(cs)
```

```
#merging with GTD
GTDpolity<-merge(GTD1, polity ,by=c("country","iyear"),all.x=TRUE)
pol<-GTDpolity[,c(21)]#extract polity
#replace NA values by 0 (warning polityIV=0 means neither demo nor auto
    -cratic)
pol[is.na(pol)] <- 0
pol<-scale(pol)#normalising data (mean=0, sd=1) for better INLA process
pol<-as.vector(pol)
#end Polity IV covariate (country-level + temporal)

#GREG ethnic group covariate (spatial not temporal)
#import GREG file from Weidmann ETHZ: http://www.icr.ethz.ch/data/other/
    greg
GREG <- readShapeSpatial("C:/Users/apython/Documents/Andre/Education/
    StAndrews/PhD/PhD_R/GREG/GREG.shp")
proj4string(GREG)<-CRS("+proj=longlat +datum=WGS84 +no_defs+towgs84
    =0,0,0+ellps=WGS84")
#create vector shape with GTD1
GTDpt<- SpatialPoints(GTD1[, c("longitude", "latitude")])
proj4string(GTDpt)<-CRS("+proj=longlat +datum=WGS84 +no_defs+towgs84
    =0,0,0+ellps=WGS84")
#intersect values of GREG with points of GTD
o <- over(GTDpt,GREG)
#recombine points with attributes from GRM
GTDpt<-cbind(GTDpt,o)
#add id unique values in GTDresp to sort it later
id <- c(1:nrow(GTDpt))
GTDGREG <- cbind(id=id, GTDpt)
#keep only id and number of ethnic groups
GTDGREG<-GTDGREG[,c(1,8,9,10)]
#if no ethnic group (0) put NA in order that the absence of ethnic group
     will not be counted
GTDGREG[GTDGREG == 0] <- NA
GTDGREG<-ddply(GTDGREG, .(id), mutate, count = length(unique(na.omit(c(
    G1ID,G2ID,G3ID)))))
#sort(already done but for double check)
GTDGREG <- GTDGREG[order(id),]
#scale
GTDGREG$count<-scale(GTDGREG$count)
#keep a vector
```

```
greg<-as.vector(GTDGREG$count)
#EndGREG ethnic group covariate (spatial not temporal

#Altitude covariate (spatial not temporal)
#source: DEM map (ETOPO1): https://www.ngdc.noaa.gov/mgg/global/relief/
    ETOPO1/image/
altitude <- raster("C:/Users/apython/Documents/Andre/Education/StAndrews
    /PhD/PhD_ArcGIS/DEM/ETOPO1/color_etopo1_ice_full.tif")
projection(altitude)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84
    =0,0,0+ellps=WGS84"
#plot(altitude)
#create lon lat in matrix form
xy<-cbind(GTD1[,8],GTD1[,7])#select the 3rd and 2nd variable of GTD
#extract coordinates from population at the location of the points
    coordinates
alt<-extract(altitude,xy,method='bilinear')#extract covariate value at
    point locations
alt<-as.vector(alt)
#replace NA values by 0
alt[is.na(alt)] <- 0
alt<-scale(alt)#normalising data (mean=0, sd=1) for better INLA process
#test plot to check distance values
#create dataframe
# GTDalt<-cbind(GTD1,alt)
# plot.new()
# rbPal <- colorRampPalette(c('blue','green','yellow','orange','red','
    brown'))
# GTDalt$Col <- rbPal(6)[as.numeric(cut(GTDalt$alt,breaks = 6))]
# plot(GTDalt$longitude,GTDalt$latitude,pch = 20,cex=0.3,col = GTDalt$
    Col)
# plot(world,add=T)
#End altitude covariate (spatial not temporal)

#Slope covariate (spatial not temporal)
require(insol)
slope<-slope(cgrad(altitude), degrees = TRUE)
slope=raster(slope,crs=projection(altitude))
extent(slope)=extent(altitude)
#create lon lat in matrix form
xy<-cbind(GTD1[,8],GTD1[,7])#select the 3rd and 2nd variable of GTD
```

```
#extract coordinates from population at the location of the points
    coordinates
slo<-extract(slope,xy,method='bilinear')#extract covariate value at
    point locations
slo<-as.vector(slo)
#replace NA values by 0
slo[is.na(slo)] <- 0
slo<-scale(slo)#normalising data (mean=0, sd=1) for better INLA process
#test plot to check distance values
#create dataframe
# GTDslo<-cbind(GTD1,slo)
# plot.new()
# rbPal <- colorRampPalette(c('blue','green','yellow','orange','red','
    brown'))
# GTDslo$Col <- rbPal(6)[as.numeric(cut(GTDslo$slo,breaks = 6))]
# plot(GTDslo$longitude,GTDslo$latitude,pch = 20,cex=0.3,col = GTDslo$
    Col)
# plot(world,add=T)
# slopew <- mask(slope, world.map)
# plot.new()
# plot(slopew,col= terrain.colors(12))
#End slope covariate (spatial not temporal)

#Distance to country border (spatial)
GTDp<-ppp(GTD1[,8],GTD1[,7],c(-180,180), c(-90,90))
#download simple map with national borders from natural earth data
    (1:110) instead of Digital chart of the world 2000 (country borders)-
    too complex
world.map <- readShapeSpatial("C:/Users/apython/Documents/Andre/
    Education/StAndrews/PhD/PhD_ArcGIS/World␣map/ne_110m_admin_0_
    countries.shp")
proj4string(world.map)<-CRS("+proj=longlat␣+datum=WGS84␣+no_defs+towgs84
    =0,0,0+ellps=WGS84")
world<-as.owin(world.map)
world = edges(world.map)
distb<-nncross(GTDp,world)
distb<-as.vector(distb$dist)
#distb[is.na(distb)] <- 0
distb<-scale(distb)#normalising data (mean=0, sd=1) for better INLA
    process
```

```
#test plot to check distance values
#create dataframe
# GTDdistb<-cbind(GTD1,distb)
# plot.new()
# rbPal <- colorRampPalette(c('blue','green','yellow','orange','red','
    brown'))
# GTDdistb$Col <- rbPal(6)[as.numeric(cut(GTDdistb$distb,breaks = 6))]
# plot(GTDdistb$longitude,GTDdistb$latitude,pch = 20,cex=0.3,col =
    GTDdistb$Col)
# plot(world,add=T)
#END distance to country border (spatial not temporal)

# #delete unuseful data
rm(GTDpolity);rm(polity);rm(xy);rm(lat);rm(lon);rm(pop00);rm(lum02);rm(
    lum03);rm(lum04);rm(ttime)
rm(lum05);rm(lum06);rm(lum07);rm(lum08);rm(lum09);rm(lum10);rm(lum11);rm
    (lum12);rm(lum13);
rm(v);rm(w);rm(nGTD);rm(id);rm(bdry);rm(world.map);rm(GTDresp);rm(GTDlum
    );rm(f02,f03,f04,f05,f06,f07,f08,f09,f10,f11,f12,f13)
rm(GTDGREG);rm(GTDpt);rm(o);rm(GREG);rm(altitude);rm(GTDdistb);rm(GTDalt
    );rm(GTDp);rm(world);rm(rbPal);rm(GTDtt);rm(GTD1lum);
rm(slope)
#save all data into
save.image("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/SPDE.RData")
```

C.2 Covariate Selection

This code has been generated to investigate correlation between the covariates).

```
#The goal of this process is to look at correlation between covariates
    and GTD events (2002-2013) in the study area (World)
#initialisation
setwd("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/PhD_R/
    Covariates")
#load SPDE values (include covariate values at each GTD coordinate and
    year for temporal variable)
load("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/PhD_R/
    SPDE/SPDE.RData")#contains study area, mesh, covariates
#create a dataframe with GTD variables and covariate values
pop<-as.vector(pop)
```

```
GTD1$pop<-pop
lum<-as.vector(lum)
GTD1$lum<-lum
greg<-as.vector(greg)
GTD1$greg<-greg
pol<-as.vector(pol)
GTD1$pol<-pol
tt<-as.vector(tt)
GTD1$tt<-tt
alt<-as.vector(alt)
GTD1$alt<-alt
slo<-as.vector(slo)
GTD1$slo<-slo
distb<-as.vector(distb)
GTD1$distb<-distb
#covariates and lethality
#delete if NA in nb. kill (fatalities)
GTD1<-GTD1[complete.cases(GTD1[,15]),]
#create lethality from number of killing
GTD1$lethal<-ifelse(GTD1$nkill==0,0,1)#4th column is nb. fatalities:
    change to binary: lethal(1),non-lethal(0)
#delete matrix elements
rm(tt);rm(pol);rm(pop);rm(greg);rm(lum);rm(alt);rm(slo);rm(distb)
#looking at the relationship between lethality and covariate values
#construct corstars function from http://myowelt.blogspot.co.uk/2008/04/
    beautiful-correlation-tables-in-r.html
corstarsl <- function(x){
  require(Hmisc)
  x <- as.matrix(x)
  R <- rcorr(x)$r
  p <- rcorr(x)$P
  ## define notions for significance levels; spacing is important.
  mystars <- ifelse(p < .001, "***", ifelse(p < .01, "** ", ifelse(p <
      .05, "* ", " ")))
  ## trunctuate the matrix that holds the correlations to two decimal
  R <- format(round(cbind(rep(-1.11, ncol(x)), R), 2))[,-1]
  ## build a new matrix that includes the correlations with their
      apropriate stars
  Rnew <- matrix(paste(R, mystars, sep=""), ncol=ncol(x))
  diag(Rnew) <- paste(diag(R), " ", sep="")
```

```
  rownames(Rnew) <- colnames(x)
  colnames(Rnew) <- paste(colnames(x), "", sep="")
  ## remove upper triangle
  Rnew <- as.matrix(Rnew)
  Rnew[upper.tri(Rnew, diag = TRUE)] <- ""
  Rnew <- as.data.frame(Rnew)
  ## remove last column and return the matrix (which is now a data frame
      )
  Rnew <- cbind(Rnew[1:length(Rnew)-1])
  return(Rnew)
}
#correlation between covariates and lethality of terrorism (lethal=1,
    non-lethal=0)
require(xtable)
xtable(corstarsl(GTD1[,24:32]))
#not in the thesis but could be investigated if necessary
#correlation between covariates and number of fatalities of terrorism (
    fatalites is integer)
# require(xtable)
# xtable(corstarsl(GTD1[,c(15,24:31)]))
```

C.3 Binomial Models: Specification

This code includes all investigated spatio-temporal Bernoulli models of terrorism's lethality.

```
#SPATIO-TEMPORAL MODELLING / binomial models of lethal events worldwide
    2002-2013 constructed from various covariates (from 0 to 5 covariates
    )
#The code has been run in NTNU (server)
rm(list=ls())
setwd("/home/ap215/SPDE")#NTNU home file
load("SPDEfinal.RData")
library(INLA)
bdry <- inla.sp2segment(studyarea)
bdry$loc <- inla.mesh.map(bdry$loc, projection = "longlat",inverse =
    TRUE)
#mesh for robusntess tests
mesh3<-inla.mesh.2d(boundary=bdry, max.edge=c(9,6000)/180,cutoff=9/180)#
    nv=1,341
```

```
mesh2<-inla.mesh.2d(boundary=bdry, max.edge=c(7,1000)/180,cutoff=7/180)#
    nv=2,157
#selecting events from 2002 to 2013 (12 years). For computational issues
    , only post 9/11 events are selected
data<-cbind(loc,GTD1[,15])
data[,4]<-ifelse(data[,4]==0,0,1)#4th column is nb. fatalities: change
    to binary: lethal(1),non-lethal(0)
y<-data[,4]
time<-GTD1[,4]-2001#create time index (transform real time (year) into
    index: required)
#generate the mesh
k<-12#the number of years
nv<-mesh$n
spde <- inla.spde2.matern(mesh, alpha=2)
iset<-inla.spde.make.index('i',n.spde=spde$n.spde,n.group=k)
A<-inla.spde.make.A(mesh,loc=loc, group=time)
#Producing several models with different number of covariates
#0 covariate
stk <- inla.stack(data=list(y=y),A=list(A,1), tag='dat',
                  effects=list(i=iset,b0=rep(1,length(y))))
res <- inla(y ~ 0 + b0 +f(i, model=spde,group=i.group,control.group=list
    (model="ar1")),
            family= 'binomial',data=inla.stack.data(stk), control.
                compute=list(dic=TRUE),
            control.predictor=list(A=inla.stack.A(stk), compute=TRUE),
                verbose=TRUE)
# #1 spatial covariate distb
stk <- inla.stack(data=list(y=y),A=list(A,1,1), tag='dat',#if two
    covariates use: A=list(A,1,1,1)
                  effects=list(i=iset,distb=distb,b0=rep(1,length(y))))#
                      add after pop=pop, lum=lum to add the second
                     covariate. If problem:try this : list(list(i=iset),
                      list(cov=pop), list(b0=rep(1,length(y)))
res1 <- inla(y ~ 0+ b0 +distb +f(i, model=spde,group=i.group,control.
    group=list(model="ar1")),##if two covariates use after "+ pop,": +lum
    +f(...)
            family= 'binomial',data=inla.stack.data(stk), control.
                compute=list(dic=TRUE),
            control.predictor=list(A=inla.stack.A(stk), compute=TRUE),
                verbose=TRUE)
```

```
# #2 spatial covariates ethnic + pop
stk <- inla.stack(data=list(y=y),A=list(A,1,1,1), tag='dat',#if two
    covariates use: A=list(A,1,1,1)
                  effects=list(i=iset,greg=greg, pop=pop,b0=rep(1,length
                      (y))))# add after pop=pop, lum=lum to add the
                      second covariate. If problem:try this : list(list(i
                      =iset), list(cov=pop), list(b0=rep(1,length(y)))
res2 <- inla(y ~ 0+ b0 +greg +pop +f(i, model=spde,group=i.group,control
    .group=list(model="ar1")),##if two covariates use after "+ pop,": +
    lum+f(...)
             family= 'binomial',data=inla.stack.data(stk), control.
                 compute=list(dic=TRUE),
             control.predictor=list(A=inla.stack.A(stk), compute=TRUE),
                 verbose=TRUE)
# #3 spatial covariates tt + pop + greg
stk <- inla.stack(data=list(y=y),A=list(A,1,1,1,1), tag='dat',
                  effects=list(i=iset,tt=tt,pop=pop,greg=greg,b0=rep(1,
                      length(y))))# If problem:try this : list(list(i=
                      iset), list(cov=pop), list(b0=rep(1,length(y)))
res3 <- inla(y ~ 0+b0 + tt + pop + greg +f(i, model=spde,group=i.group,
    control.group=list(model="ar1")),##if two covariates use after "+ pop
    ,": +lum+f(...)
              family= 'binomial',data=inla.stack.data(stk), control.
                  compute=list(dic=TRUE),
              control.predictor=list(A=inla.stack.A(stk), compute=TRUE),
                  verbose=TRUE)
# #4 spatial covariates lum + pop + greg + tt
stk <- inla.stack(data=list(y=y),A=list(A,1,1,1,1,1), tag='dat',
                  effects=list(i=iset,lum=lum,pop=pop,greg=greg,tt=tt,b0
                      =rep(1,length(y))))# If problem:try this : list(
                      list(i=iset), list(cov=pop), list(b0=rep(1,length(y
                      )))
res4 <- inla(y ~ 0+b0 +lum+pop+greg+tt+f(i, model=spde,group=i.group,
    control.group=list(model="ar1")),##if two covariates use after "+ pop
    ,": +lum+f(...)
             family= 'binomial',data=inla.stack.data(stk), control.
                 compute=list(dic=TRUE),
             control.predictor=list(A=inla.stack.A(stk), compute=TRUE),
                 verbose=TRUE)
# #5 spatial covariates lum + pop + greg + tt + altitude
```

```
stk <- inla.stack(data=list(y=y),A=list(A,1,1,1,1,1,1), tag='dat',
effects=list(i=iset,lum=lum,pop=pop,tt=tt,greg=greg,alt=alt,b0=rep(1,
    length(y))))# If problem:try this : list(list(i=iset), list(cov=pop),
     list(b0=rep(1,length(y)))
res5 <- inla(y ~ 0+b0+pop+lum+tt+greg+alt+f(i, model=spde,group=i.group,
    control.group=list(model="ar1")),##if two covariates use after "+ pop
    ,": +lum+f(...)
family= 'binomial',data=inla.stack.data(stk), control.compute=list(dic=
    TRUE),
control.predictor=list(A=inla.stack.A(stk), compute=TRUE),verbose=TRUE)

#4 bis: 4 spatial covariates lum + greg + tt + altitude
stk <- inla.stack(data=list(y=y),A=list(A,1,1,1,1,1), tag='dat',
                  effects=list(i=iset,lum=lum,tt=tt,greg=greg,alt=alt,b0
                     =rep(1,length(y))))# If problem:try this : list(
                     list(i=iset), list(cov=pop), list(b0=rep(1,length(y
                     )))
res4bis <- inla(y ~ 0+b0+lum+tt+greg+alt+f(i, model=spde,group=i.group,
    control.group=list(model="ar1")),##if two covariates use after "+ pop
    ,": +lum+f(...)
              family= 'binomial',data=inla.stack.data(stk), control.
                 compute=list(dic=TRUE),
              control.predictor=list(A=inla.stack.A(stk), compute=TRUE),
                 verbose=TRUE)
save.image("/home/andre/R/SPDE/STbintotal.RData")
#robustness test (alternative meshes)
#mesh2
nv<-mesh2$n
spde <- inla.spde2.matern(mesh2, alpha=2)
iset<-inla.spde.make.index('i',n.spde=spde$n.spde,n.group=k)
A<-inla.spde.make.A(mesh2,loc=loc, group=time)

stk <- inla.stack(data=list(y=y),A=list(A,1,1,1,1,1), tag='dat',
                   effects=list(i=iset,lum=lum,tt=tt,greg=greg,alt=alt,
                      b0=rep(1,length(y))))# If problem:try this : list(
                      list(i=iset), list(cov=pop), list(b0=rep(1,length(
                      y)))
res4mesh2 <- inla(y ~ 0+b0+lum+tt+greg+alt+f(i, model=spde,group=i.group
    ,control.group=list(model="ar1")),##if two covariates use after "+
    pop,": +lum+f(...)
```

```
                  family= 'binomial',data=inla.stack.data(stk), control.
                      compute=list(dic=TRUE),
                  control.predictor=list(A=inla.stack.A(stk), compute=TRUE
                      ),verbose=TRUE)
save(res4mesh2 ,file="res4mesh2.RData")
#mesh3
nv<-mesh3$n
spde <- inla.spde2.matern(mesh3, alpha=2)
iset<-inla.spde.make.index('i',n.spde=spde$n.spde,n.group=k)
A<-inla.spde.make.A(mesh3,loc=loc, group=time)
stk <- inla.stack(data=list(y=y),A=list(A,1,1,1,1,1), tag='dat',
                    effects=list(i=iset,lum=lum,tt=tt,greg=greg,alt=alt,
                        b0=rep(1,length(y))))# If problem:try this : list(
                        list(i=iset), list(cov=pop), list(b0=rep(1,length(
                        y)))
res4mesh3 <- inla(y ~ 0+b0+lum+tt+greg+alt+f(i, model=spde,group=i.group
    ,control.group=list(model="ar1")),##if two covariates use after "+
    pop,": +lum+f(...)
                  family= 'binomial',data=inla.stack.data(stk), control.
                      compute=list(dic=TRUE),
                  control.predictor=list(A=inla.stack.A(stk), compute=
                      TRUE),verbose=TRUE)
save(res4mesh3 ,file="res4mesh3.RData")
#END
```

C.4 Binomial Models: Model Selection and Graphics

This code is used to generate and compare the goodness-of-fit of the models, and provide a summary of the posterior densities of the main parameters of the selected models.

```
#SPATIO-TEMPORAL MODELLING / Bernoulli model of lethal events worldwide
    2002-2013
#This codes generate graphics of the random field (mean + std.dev) and
    the probability of lethal attacks based on the output of STbin_total_
    flipperv2.R
#The model to be loaded needs to be the selected one.
setwd("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE")
#load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal/
    STbintotal.RData")
```

```
load("C:/Users/apython/Documents/Andre/Education/StAndrews/arXiv/
    RSSstylefile/R_code/STbintotal.RData")#data 2002-2013
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal/Llinpal
    .RData")#load custom palette
require(foreign)
require(sp)
library(INLA)
library(spatstat)
library(fields)
library(maptools)
library(fields)
library(lattice)
#Model selection (DIC values comparison)
summary(res);summary(res1);summary(res2);summary(res3);
summary(resfinal);summary(res5);summary(res4)
#Model 4bis selected (note that distb is not kept for model 5. The
    variable is not "significant": not same sign between interquantile)
#load data?
#load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/SPDE.RData")
#res4<-res4bis
GTD1<-GTD
#model with lower DIC
summary(resfinal)
res$cpu.used;res1$cpu.used;res2$cpu.used;res3$cpu.used
resfinal$cpu.used;res4$cpu.used;res5$cpu.used
#comparison with alternative meshes
round(resfinal$summary.fixed,3)#B0-,lum-,tt+,greg+,alt+
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal/
    resfinalmesh2.RData")#3 var model
round(resfinalmesh2$summary.fixed,3)#B0-,lum-,tt+,greg+,alt/
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal/
    resfinalmesh3.RData")#3
round(resfinalmesh3$summary.fixed,3)#B0-,lum-,tt+,greg+,alt/
#prior check
#spde$f$hyper.default
#create points in location of mesh vertices
meshvert<-rbind(c(mesh$loc[,1],mesh$loc[,2],mesh$loc[,3]))
#convert from Cartesian xyz to latitude-longitude
lat = asin(mesh$loc[,3])
lon = atan2(mesh$loc[,2], mesh$loc[,1])
```

```
lat<-lat*180/pi
lon<-lon*180/pi #from radian to degree
coord=cbind(lon,lat)
sp = SpatialPoints(coord)
longlat<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+ellps=WGS84"
proj4string(sp) = CRS(longlat)
#plot(studyarea,col="red")
#points(coord,cex=0.4,pch="+")
bdry<-inla.sp2segment(studyarea)
bdry$loc <- inla.mesh.map(bdry$loc, projection="longlat", inverse=TRUE)
#mesh version to be used for 4 version of PhD
#high detailed mesh
mesh<-inla.mesh.2d(boundary=bdry, max.edge=c(3,1000)/180,cutoff=3/180)#
    simple: max.edge=c(6,10000)/180,cutoff=10.5/180)
#plot on 2d (projection)
proj = inla.mesh.projector(mesh, projection = "longlat",dims=c(1444,724)
    )
win<-as.owin(studyarea)
library(mgcv)
e<-expand.grid(proj$x,proj$y)
ins<-inside.owin(e[,1],e[,2],win)
ins<-matrix(ins,nrow=length(proj$y))
#goodness-of-fit
GTD1$lum<-as.numeric(lum)
GTD1$alt<-as.numeric(alt)
GTD1$distb<-as.numeric(distb)
GTD1$lum<-as.numeric(lum)
GTD1$pop<-as.numeric(pop)
GTD1$tt<-as.numeric(tt)
GTD1$greg<-as.numeric(greg)
GTD1$time<-GTD1$iyear-2001
locdf<-as.data.frame(loc)
GTD1$loc1<-as.numeric(locdf[,1])
GTD1$loc2<-as.numeric(locdf[,2])
GTD1$loc3<-as.numeric(locdf[,3])
meanfinal<-list();covfinal<-list();probfinal<-list()
mean0<-list();cov0<-list();prob0<-list()
mean3<-list();cov3<-list();prob3<-list()
mean5<-list();cov5<-list();prob5<-list()
loc<-list()
```

```
for (j in 1:k){
loc[[j]]<-as.matrix(cbind(GTD1$loc1[GTD1$time==j],GTD1$loc2[GTD1$time==j
    ],GTD1$loc3[GTD1$time==j]))
#final model
covfinal[[j]]<-resfinal$summary.fix[1,1] +resfinal$summary.fix[2,1]*GTD1
    $lum[GTD1$time==j]+resfinal$summary.fix[3,1]*GTD1$tt[GTD1$time==j] +
    resfinal$summary.fix[4,1]*GTD1$greg[GTD1$time==j] + resfinal$summary.
    fix[5,1]*GTD1$alt[GTD1$time==j]
meanfinal[[j]]<-inla.mesh.project(inla.mesh.projector(mesh, loc=loc[[j
    ]]),resfinal$summary.random$i$mean[iset$i.group==j])
probfinal[[j]]<-binomial(link='logit')$linkinv(meanfinal[[j]]+covfinal[[
    j]])
#Zero cov model
cov0[[j]]<-res$summary.fix[1,1]
mean0[[j]]<-inla.mesh.project(inla.mesh.projector(mesh, loc=loc[[j]]),
    res$summary.random$i$mean[iset$i.group==j])
prob0[[j]]<-binomial(link='logit')$linkinv(mean0[[j]]+cov0[[j]])
#3 cov model
cov3[[j]]<-res3$summary.fix[1,1] +res3$summary.fix[2,1]*GTD1$tt[GTD1$
    time==j]+res3$summary.fix[3,1]*GTD1$pop[GTD1$time==j] + res3$summary.
    fix[4,1]*GTD1$greg[GTD1$time==j]
mean3[[j]]<-inla.mesh.project(inla.mesh.projector(mesh, loc=loc[[j]]),
    res3$summary.random$i$mean[iset$i.group==j])
prob3[[j]]<-binomial(link='logit')$linkinv(mean3[[j]]+cov3[[j]])
#5 cov model
cov5[[j]]<-res5$summary.fix[1,1] +res5$summary.fix[2,1]*GTD1$pop[GTD1$
    time==j]+res5$summary.fix[3,1]*GTD1$lum[GTD1$time==j] + res5$summary.
    fix[4,1]*GTD1$tt[GTD1$time==j] + res5$summary.fix[5,1]*GTD1$greg[GTD1
    $time==j]+res5$summary.fix[6,1]*GTD1$alt[GTD1$time==j]
mean5[[j]]<-inla.mesh.project(inla.mesh.projector(mesh, loc=loc[[j]]),
    res5$summary.random$i$mean[iset$i.group==j])
prob5[[j]]<-binomial(link='logit')$linkinv(mean5[[j]]+cov5[[j]])
}
#unlist
GTD1$probfinal<-unlist(probfinal)
GTD1$prob5<-unlist(prob5)
GTD1$prob3<-unlist(prob3)
GTD1$prob0<-unlist(prob0)
rmsedf<-GTD1
```

```
rmsedf$nkill<-ifelse(rmsedf$nkill==0,0,1)#4th column is nb. fatalities:
    change to binary: lethal(1),non-lethal(0)
rmsedf<-rmsedf[c("latitude","longitude","nkill","probfinal","prob5","
    prob3","prob0")]
colnames(rmsedf) <- c("lat","lon","obs","probfinal","prob5","prob3","
    prob0")
#remove if NA
rmsedf<-rmsedf[complete.cases(rmsedf),]
rmseprobfinal<-sqrt(mean((rmsedf$obs-rmsedf$probfinal)^2,na.rm=T)) ;
    rmseprobfinal
rmseprob5<-sqrt(mean((rmsedf$obs-rmsedf$prob5)^2,na.rm=T));rmseprob5
rmseprob3<-sqrt(mean((rmsedf$obs-rmsedf$prob3)^2,na.rm=T));rmseprob3
rmseprob0<-sqrt(mean((rmsedf$obs-rmsedf$prob0)^2,na.rm=T));rmseprob0
maeprobfinal<-mean(abs(rmsedf$obs-rmsedf$probfinal),na.rm=T) ;
    maeprobfinal
maeprob5<-mean(abs(rmsedf$obs-rmsedf$prob5),na.rm=T) ;maeprob5
maeprob3<-mean(abs(rmsedf$obs-rmsedf$prob3),na.rm=T) ;maeprob3
maeprob0<-mean(abs(rmsedf$obs-rmsedf$prob0),na.rm=T) ;maeprob0

xmean<-list()
for (j in 1:k){
  xmean[[j]]<-inla.mesh.project(proj,resfinal$summary.random$i$mean[iset
      $i.group==j])#update file location according to model selection
}
for (j in 1:k){
  xmean[[j]][!ins]<-NA
}
probsurf<-list()
for(j in 1:k){
  probsurf[[j]]<-binomial(link='logit')$linkinv(resfinal$summary.fix
      [1,1] +resfinal$summary.fix[2,1]*GTD1$lum[GTD1$time==j]+
    resfinal$summary.fix[3,1]*GTD1$tt[GTD1$time==j] + resfinal$summary.
        fix[4,1]*GTD1$greg[GTD1$time==j] + resfinal$summary.fix[5,1]*GTD1$
        alt[GTD1$time==j]+xmean[[j]])

}
#plot prob. surface for year 2002
dev.off()
plot.new()
breaks<-seq(0,1,by=0.001)
```

```
n<-length(breaks)-1
par(cex=3,mar=c(5, 4, 2, 4.5) + 0.1)#mar: c(bottom,left,top,right)
image.plot(proj$x,proj$y,probsurf[[1]],col=Llinpal(n),xlab="Longitude",
           xlim = c(-180, 180), ylim = c(-60, 90),zlim=c(0,1),#zlim
               creates identical scale
           main= 2001+1,sub="Prob surf" ,ylab="Latitude",
           breaks=breaks)#should have one more break than color
#plot all years from 2002 to 2013 and save in file all pictures (30
    pictures)
for(i in 1:12){
  mypath=paste("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/
      STbinomialtotal/probsurf/probsurf",i,".png",sep="")
  png(file=mypath,width=1444,height=724)
  par(cex=3,mar=c(5, 4, 2, 3.5) + 0.1)#mar: c(bottom,left,top,right)
  image.plot(proj$x,proj$y,probsurf[[i]],col=Llinpal(n),xlab="Longitude"
      ,
   main="Probability of lethal attack",sub=2001+i,ylab="Latitude",
   xlim = c(-180, 180), ylim = c(-60, 90),zlim=c(0,1),
   breaks=breaks)
  dev.off()
}
dev.off()
plot.new()
for(i in 1:12){
  mypath=paste("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/
      STbinomialtotal/probsurf/probsurf",i,".eps",sep="")
  postscript(file=mypath,horiz=FALSE,onefile=FALSE,width=21,height=8,
      paper="a4")
  par(cex=3,mar=c(3.75, 0.1, 1.75, 3.4) + 0.1)#mar: c(bottom,left,top,
      right) changed 'mar' (original: mar=c(1.75, 0.1, 1.75, 3.4)) to be
      in line with random mean and sd pictures
  image.plot(proj$x,proj$y,probsurf[[i]],col=Llinpal(n),xlab=NA,ylab=NA,
             main=NA,axes=FALSE,
             xlim = c(-180, 180), ylim = c(-60, 90),zlim=c(0,1),
             breaks=breaks,legend.shrink=1.1)
  dev.off()
}

#put everything into one file only
library(reshape2)
```

```
probsurfdf<-list()
for(i in 1:12){
probsurfdf[[i]]<-melt(probsurf[[i]])
probsurfdf[[i]]$year<-as.factor(2001+i)
probsurfdf[[i]]<-probsurfdf[[i]][c("value","year")]
probsurfdf[[i]]$longitude<-e$Var1
probsurfdf[[i]]$latitude<-e$Var2
probsurfdf[[i]]<-probsurfdf[[i]][which(probsurfdf[[i]]$latitude > -57 &
    probsurfdf[[i]]$latitude < 85),]#optional for plotting
}
#mean of the random field
#plot all years from 2002 to 2013 and save in file all pictures (30
   pictures)
minrf<-min(sapply(xmean, min,na.rm=TRUE))
maxrf<-max(sapply(xmean, max,na.rm=TRUE))
brk<-seq(minrf,maxrf,by=0.01)
n<-length(brk)-1
for(i in 1:12){
  mypath=paste("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/
      STbinomialtotal/birf/birf",i,".png",sep="")
  png(file=mypath,width=1444,height=724)
  par(cex=3,mar=c(5, 4, 2, 3.5) + 0.1)#mar: c(bottom,left,top,right)
  image.plot(proj$x,proj$y,xmean[[i]],col=Llinpal(n),xlab="Longitude",
             main="Mean␣of␣the␣Random␣field",sub=2001+i,ylab="Latitude",
             xlim = c(-180, 180), ylim = c(-60, 90),zlim=c(minrf,maxrf),
             breaks=brk)
  dev.off()
}

minrf<-min(sapply(xmean, min,na.rm=TRUE))
maxrf<-max(sapply(xmean, max,na.rm=TRUE))
brk<-seq(minrf,maxrf,by=0.01)
n<-length(brk)-1
for(i in 1:12){
  mypath=paste("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/
      STbinomialtotal/birf/birf",i,".eps",sep="")
  postscript(file=mypath,horiz=FALSE,onefile=FALSE,width=21,height=8,
      paper="a4")
  par(cex=3,mar=c(1.75, 0.1, 1.75, 3.4) + 0.1)#mar: c(bottom,left,top,
      right)
```

```
  image.plot(proj$x, proj$y, xmean[[i]], col=Llinpal(n), xlab=NA, ylab=NA,
             main=NA, axes=FALSE, zlim=c(minrf, maxrf),
             breaks=brk)
  dev.off()
}
#standard deviation of the random field
xsd<-list()
for (j in 1:k){
  xsd[[j]]<-inla.mesh.project(proj, res4$summary.random$i$sd[iset$i.group
      ==j])
}
for (j in 1:k){
  xsd[[j]][!ins]<-NA
}

minsd<-min(sapply(xsd, min, na.rm=TRUE))
maxsd<-max(sapply(xsd, max, na.rm=TRUE))
brk<-seq(minsd, maxsd, by=0.01)
n<-length(brk)-1
for(i in 1:12){
  mypath=paste("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/
      STbinomialtotal/bisdrf/bisdrf",i,".png",sep="")
  png(file=mypath, width=1444, height=724)
  par(cex=3,mar=c(5, 4, 2, 3.5) + 0.1)#mar: c(bottom, left, top, right)
  image.plot(proj$x, proj$y, xsd[[i]], col=Llinpal(n), xlab="Longitude",
             main="Standard␣deviation␣of␣the␣Random␣field", sub=2001+i,
                 ylab="Latitude",
             xlim = c(-180, 180), ylim = c(-60, 90), zlim=c(minsd, maxsd),
             breaks=brk)
  dev.off()
}

minsd<-min(sapply(xsd, min, na.rm=TRUE))
maxsd<-max(sapply(xsd, max, na.rm=TRUE))
brk<-seq(minsd, maxsd, by=0.01)
n<-length(brk)-1
dev.off()
plot.new()
for(i in 1:12){
```

```
  mypath=paste("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/
      STbinomialtotal/bisdrf/bisdrf",i,".eps",sep="")
  postscript(file=mypath,horiz=FALSE,onefile=FALSE,width=21,height=8,
      paper="a4")
  par(cex=3,mar=c(1.75, 0.1, 1.75, 3.4) + 0.1)#mar: c(bottom,left,top,
      right)
  image.plot(proj$x,proj$y,xsd[[i]],col=Llinpal(n),xlab=NA,ylab=NA,
             main=NA,axes=FALSE,zlim=c(minsd,maxsd),
             breaks=brk)
  dev.off()
}
#graph (parameter posterior densities)
#intercept
par(mfrow=c(1,2), mar=c(3,3.5,0,0), mgp=c(1.5, .5, 0), las=0)
plot(resfinal$marginals.fix[[1]], type='l', xlab='Intercept', ylab='
    Posterior␣density')#update file location according to model selection
rfsd<-density(resfinal$summary.random$i[[3]])#, xlab='sd of the random
    field', ylab='Posterior density')
plot(rfsd,main='',xlab='standard␣deviation␣of␣the␣random␣field', ylab='
    Posterior␣density')
#Beta coefficients
round(resfinal$summary.fixed[,1:5],3)
#temporal and spatial parameters
summary(resfinal)
#Spatial parameters with nominal scale (time is aggregated)
spde.resfinal <- inla.spde2.result(resfinal, name="i",spde,do.transform=
    TRUE)#do.transform put in correct scale
#kappa /computed using the approach from Blangiardo & Cameletti (2015)
Kappa<-inla.emarginal(function(x) x, spde.resfinal$marginals.kappa[[1]])
    #kappa (mean)
Kappahpd<-inla.hpdmarginal(0.95, spde.resfinal$marginals.kappa[[1]])#
    kappa (hpd 95%)
#variance of the random field
variance<-inla.emarginal(function(x) x, spde.resfinal$marginals.variance
    .nominal[[1]])#variance (mean)
variancehpd<-inla.hpdmarginal(0.95, spde.resfinal$marginals.variance.
    nominal[[1]])#variance (hpd 95%)
#range in radian
range<-inla.emarginal(function(x) x, spde.resfinal$marginals.range.
    nominal[[1]])#range (mean)
```

```
rangehpd<-inla.hpdmarginal(0.95, spde.resfinal$marginals.range.nominal
    [[1]])#range (hpd 95%)
#Matrn
matern <- function(lambda,kappa,dist)
  2^(1-lambda)/gamma(lambda) * (kappa*dist)^lambda* besselK(x=dist*kappa
      , nu=lambda)
dist.x = seq(0,0.31,l=100)
lambda<-1
kappa<-c(Kappahpd[,1],Kappa,Kappahpd[,2])

new.plot()
dev.off()
mypath<-"~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal/
    Matern.eps"
postscript(file=mypath,horiz=FALSE,onefile=FALSE,width=0,height=0,paper=
    "default")
par(cex=3.0,mar=c(4, 4, 1.0, 1.0) + 0.1,mgp=c(2.3,1,0))#mar: c(bottom,
    left,top,right)
plot(dist.x,matern(lambda,kappa=kappa[2],dist.x),lwd=3,type="l",ylab="
    Matrn␣covariance␣function",xlab="distance␣[km]",lty=1,xaxt = 'n')
lines(dist.x,matern(lambda,kappa=kappa[1],dist.x),lwd=3,type="l",ylab=NA
    ,xlab=NA,lty=2)
lines(dist.x,matern(lambda,kappa=kappa[3],dist.x),lwd=3,type="l",ylab=NA
    ,xlab=NA,lty=2)
abline( a = 0.1,b=0, v = NULL, reg = NULL,lty=3,lwd=2)#for vertical
    value v=range
abline( v =range, lty=3,lwd=2)#for vertical value v=range
axis(1, at=seq(0,2000/6371,500/6371), labels=c(0,500,1000,1500,2000))#1
    rad=6371km, 2000 km=2000/6371rad
dev.off()

#for thesis
plot.new()
dev.off()
mypath<-"C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/Latex/
    phd-thesis-template-phd-latex-template-latest-stable/Chapter5/Figs/
    PDF/Matern.pdf"
pdf(file=mypath,onefile=FALSE,paper="special")
par(cex=2.2,mar=c(4, 4, 1.0, 1.0) + 0.1,mgp=c(2.3,1,0))#mar: c(bottom,
    left,top,right)
```

```
plot(dist.x,matern(lambda,kappa=kappa[2],dist.x),lwd=3,type="l",ylab="
    Matrn covariance function",xlab="distance [km]",lty=1,xaxt = 'n',
    yaxt='n')
lines(dist.x,matern(lambda,kappa=kappa[1],dist.x),lwd=3,type="l",ylab=NA
    ,xlab=NA,lty=2)
lines(dist.x,matern(lambda,kappa=kappa[3],dist.x),lwd=3,type="l",ylab=NA
    ,xlab=NA,lty=2)
abline( a = 0.1,b=0, v = NULL, reg = NULL,lty=3,lwd=2)#for vertical
    value v=range
abline( v =range, lty=3,lwd=2)#for vertical value v=range
axis(1, at=seq(0,2000/6371,500/6371), labels=c(0,500,1000,1500,2000))#1
    rad=6371km, 2000 km=2000/6371rad
axis(2, at=seq(0,0.8,0.2), labels=c(0.0,0.2,0.4,0.6,0.8))#1rad=6371km,
    2000 km=2000/6371rad
dev.off()

#computational time
#MCMC (N^3) in R2
MCMC<-(9697*12)^3
#INLA (N^(3/2)) in R2
INLA<-(9697*12)^(3/2)
#ration INLA/MCMC
MCMC/INLA

#plot comparing binomial with default priors and manual priors
#loading first binomial with default priors
coeff<-as.data.frame(resfinal$summary.fixed[3:5])#keep 0.025Q,0.5Q,0.975
    Q
#kappa /computed using Blangiardo & Cameletti (2015)
variance<-inla.emarginal(function(x) x, spde.resfinal$marginals.variance
    .nominal[[1]])#variance (mean)
variancehpd<-inla.hpdmarginal(0.95, spde.resfinal$marginals.variance.
    nominal[[1]])#variance (hpd 95%)
range<-inla.emarginal(function(x) x, spde.resfinal$marginals.range.
    nominal[[1]])#range (mean)
rangehpd<-inla.hpdmarginal(0.95, spde.resfinal$marginals.range.nominal
    [[1]])#range (hpd 95%)
r<-as.vector(c(rangehpd[1],range,rangehpd[2]))#keep 0.025Q,0.5Q,0.975Q
t<-as.vector(c(1/variancehpd[2],1/variance,1/variancehpd[1]))#keep 0.025
    Q,0.5Q,0.975Q
```

```
def<-rbind(coeff,r,t)
def$model<-1

#with manual prior (link to be adjusted once the model is run)
coeff2<-as.data.frame(resprior$summary.fixed[3:5])#keep 0.025Q,0.5Q
    ,0.975Q
variance<-inla.emarginal(function(x) x, spde.resprior$marginals.variance
    .nominal[[1]])#variance (mean)
variancehpd<-inla.hpdmarginal(0.95, spde.resprior$marginals.variance.
    nominal[[1]])#variance (hpd 95%)
range<-inla.emarginal(function(x) x, spde.resprior$marginals.range.
    nominal[[1]])#range (mean)
rangehpd<-inla.hpdmarginal(0.95, spde.resprior$marginals.range.nominal
    [[1]])#range (hpd 95%)
r2<-as.vector(c(rangehpd[1],range,rangehpd[2]))#keep 0.025Q,0.5Q,0.975Q
t2<-as.vector(c(1/variancehpd[2],1/variance,1/variancehpd[1]))#keep
    0.025Q,0.5Q,0.975Q
def2<-rbind(coeff2,r2,t2)
def2$model<-2

alldef<-rbind(def,def2)
alldef$model<-as.factor(alldef$model)
alldef$name<-c("b0","lum","tt","greg","alt","range","tau","b0","lum","tt
    ","greg","alt","range","tau")
alldef <- alldef[order(alldef$name, alldef$model),]
alldef$number<-as.numeric(c("3","4","1","2","7","8","5","6","11","12","
    13","14","9","10"))
alldef<-alldef[order(alldef$number),]

plot.new()
dev.off()
mypath<-"~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal/
    binpriorcompa.eps"
postscript(file=mypath,horiz=FALSE,onefile=FALSE,width=0,height=0,paper=
    "default")
par(cex=2,mar=c(2.2, 4.3, 0.3, 0.8) + 0.1)#mar: c(bottom,left,top,right)
dotchart(alldef[,2], labels=c(expression(beta[0]),"",expression(beta[alt
    ]),"",expression(beta[lum]),"",expression(beta[greg]),"",expression(
    beta[tt]),"",expression(italic(range)),"",expression(tau),""),
        pch=c(19,17),#pch=c(21,17),#bg=c("black","white"),
```

```
          xlim=c(min(alldef[,1]), max(alldef[,3])))
segments(alldef[,2]-(alldef[,2]-alldef[,1]),1:14,alldef[,2]+(alldef[,3]-
    alldef[,2]),1:14,lwd=2)
dev.off()
```

C.5 Poisson Models: Data Aggregation

This code generate data used for the Poisson models (including GTD extraction, mesh, covariates, and study area). The time period is 2002-2013 (12 years).

```
#Creating data used for different models (space) or (spatio-temporal) of
    terrorism based on SPDE/INLA
#including GTD extraction, mesh, covariates, and study area. Time
    selected is 2002-2013 (12 years)
setwd("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/PhD_R/
    SPDE")
require(foreign)
require(sp)
require(RandomFields)
library(INLA)
library(spatstat)
library(fields)
library(maptools)
library(fields)
library(lattice)
library(latticeExtra)
library(spdep)
library(plyr)
#if one does not want to run the entire code: loading data (quicker than
    running the entire code..)
#load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/SPDE_nbevents_final.
    RData")
#1.Import terrorist events worldwide from GTD
GTD<- read.csv("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD
    /PhD_R/GTD/GTDworld.csv")
# GTD from 2002 to 2013 is called GTD1
GTD1 <- subset(GTD, iyear > 2001 & iyear < 2014)#selecting 12 years of
    observations
#keep only nkill variable and city,country,year, latitude,longitude
GTD1<-GTD1[c(1,2,4,7,8,15)]
```

```
#create a new variable that distinguish lethal from non-lethal attack
GTD1$lethal <- ifelse(GTD1$nkill >0, 1, 0)
#create a new variable that will be used to sum the total number of
    attack
GTD1$total <- 1
#Aggregate to SRt deg. level (in order to be consistent with the
    resolution of the model)
#spatioal resolution threshold SRt
SRt<-0.5#could be modified according to desired resolution
GTD1$latitude<-round(GTD1$latitude/SRt)*SRt
GTD1$longitude<-round(GTD1$longitude/SRt)*SRt
GTD1<-GTD1[c(2,3,4,5,6,7,8)]
lethal<-aggregate(lethal~latitude+longitude+iyear,GTD1,sum)#keep 1790
    observations (locations within a radius of 0.25 deg)
total<-aggregate(total~latitude+longitude+iyear,GTD1,sum)
GTDlethal<-merge(total,lethal,by=c("latitude","longitude","iyear"))
GTDcountry<-GTD1[c(1,3,4)]
GTDcountry<-GTDcountry[!duplicated(GTDcountry), ]
GTDcountry<-GTDcountry[!duplicated(GTDcountry[,2:3]),]#keep for each lat
    , lon one country
GTDlethal<-merge(GTDlethal,GTDcountry,by=c("latitude","longitude"))
any(is.na(GTDlethal))#check if NA present (should not be the case)
#2.Put points onto the unit sphere
loc<-GTDlethal[,1:2]#select latitude and longitude
loc<-loc*pi/180
lat<-loc[,1]
lon<-loc[,2]
loc = cbind(cos(lat)*cos(lon), cos(lat)*sin(lon), sin(lat))#change as
    spherical coordinates (r,phi,theta)

#3.Generate meshes on the sphere (to be adjusted for better results)
#select study area and delete small polygons
world.map <- readShapeSpatial("C:/Users/apython/Documents/Andre/
    Education/StAndrews/PhD/PhD_ArcGIS/World␣map/world.shp")
proj4string(world.map)<-CRS("+proj=longlat␣+datum=WGS84␣+no_defs+towgs84
    =0,0,0+ellps=WGS84")
studyarea <- world.map[world.map$F_AREA > 9000000000,]#delete small
    polygons
#plot(studyarea)
#in order to have a mesh on a sphere (S2)
```

```
bdry<-inla.sp2segment(studyarea)
bdry$loc <- inla.mesh.map(bdry$loc, projection="longlat", inverse=TRUE)
#Mesh
#high detailed mesh
mesh<-inla.mesh.2d(boundary=bdry, max.edge=c(3,1000)/180,cutoff=3/180)#
    simple: max.edge=c(6,10000)/180,cutoff=10.5/180)
# plot(mesh, rgl=TRUE, draw.segments=FALSE,edge.color=rgb(0.25, 0.25,
    0.25))
# interior<-inla.mesh.interior(mesh)
# for(i in 1:length(interior)) {
#   lines(interior[[i]],col=1,rgl=TRUE,lwd=2)
# }

#Covariates
#Population density 2000
#Gridded Population of the World (GPW) 2000 (which has been reworked in
    arcgis in order to put 0 if no values + extent to -90;90 latitude)
#with arcgis "raster calculator" function: Con(IsNull("inputraster),0,"
    inputraster") and processing environment add lat -90 to 90 and lon
    -180 to 180)
#then exported as ASCII with arcgis "sample" function
require(raster)
pop00<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/flipper/glds00ag_0add")
projection(pop00)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
#create lon lat in matrix form
xy<-cbind(GTDlethal[,2],GTDlethal[,1])#select the 3rd (lon) and 2nd (lat
    ) variable of GTD
#extract coordinates from population at the location of the points
    coordinates
#a bilinearl method + buffer zone is used if data is not provided (long
    duration but ensure no NA values)
pop<-extract(pop00,xy,method='bilinear')#extract covariate value at
    point locations
summary(pop)#check if NA:
pop<-as.vector(pop)
#replace NA values by 0: not do it-this may cause issues if large cities
     are not provide with values and replaced by 0. Should be avoided by
    using bilateral + buffer zone extract.
```

```
#pop[is.na(pop)] <- 0
 pop<-scale(pop)#normalising data (mean=0, sd=1) for better INLA process
#End population density 2000

#Luminosity (spatio-temporal covariate)
#adding luminosity data as spatio-temporal covariate
#satellite night light (which has been reworked in arcgis in order to
    put 0 if no values + extent to -90;90 latitude)
#with arcgis "raster calculator" function: Con(IsNull("inputraster"),0,"
    inputraster") and processing environment add lat -90 to 90 and lon
    -180 to 180)
#then exported as ASCII with arcgis "sample" function
lum02<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum02.tif")
projection(lum02)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum03<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum03.tif")
projection(lum03)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum04<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum04.tif")
projection(lum04)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum05<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum05.tif")
projection(lum05)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum06<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum06.tif")
projection(lum06)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum07<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum07.tif")
projection(lum07)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum08<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum08.tif")
projection(lum08)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
```

```
lum09<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum09.tif")
projection(lum09)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum10<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum10.tif")
projection(lum10)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum11<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum11.tif")
projection(lum11)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum12<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum12.tif")
projection(lum12)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum13<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_ArcGIS/luminosity/F182013.v4c_web.stable_lights.avg_vis.tif")
projection(lum13)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
#Extract luminosity values at locations in space and time defined by GTD
    space-time locations
lum02<-as.data.frame(lum02<-extract(lum02,xy,method='bilinear'))
lum03<-as.data.frame(lum03<-extract(lum03,xy,method='bilinear'))
lum04<-as.data.frame(lum04<-extract(lum04,xy,method='bilinear'))
lum05<-as.data.frame(lum05<-extract(lum05,xy,method='bilinear'))
lum06<-as.data.frame(lum06<-extract(lum06,xy,method='bilinear'))
lum07<-as.data.frame(lum07<-extract(lum07,xy,method='bilinear'))
lum08<-as.data.frame(lum08<-extract(lum08,xy,method='bilinear'))
lum09<-as.data.frame(lum09<-extract(lum09,xy,method='bilinear'))
lum10<-as.data.frame(lum10<-extract(lum10,xy,method='bilinear'))
lum11<-as.data.frame(lum11<-extract(lum11,xy,method='bilinear'))
lum12<-as.data.frame(lum12<-extract(lum12,xy,method='bilinear'))
lum13<-as.data.frame(lum13<-extract(lum13,xy,method='bilinear'))
# intercalibration based on Elvidge2013
# Process: 1) calculate: Y = C0 + C1X + C2X^2 // 2) Values > 63 are
    truncated at 63 // 3) values = 0 stay zero.
# Sat   Year      C0          C1       C2
# F15   2002     0.0491   0.9568   0.0010
# F15    2003    0.2217   1.5122   -0.0080
```

```
# F16   2004      0.2853   1.1955   -0.0034
# F16     2005    -0.0001  1.4159   -0.0063
# F16     2006    0.1065   1.1371   -0.0016
# F16     2007    0.6394   0.9114   0.0014
# F16   2008      0.5564   0.9931   0.0000
# F16     2009    0.9492   1.0683   -0.0016
# F18   2010      2.3430   0.5102   0.0065
# F18     2011    1.8956   0.7345   0.0030
# F18     2012    1.8750   0.6203   0.0052
f02 <- function(x){ifelse(x>0, 0.0491+0.9568*x+0.0010*x^2, 0)}#if the
    values are above 0, execute intercalibration
f03 <- function(x){ifelse(x>0, 0.2217+1.5122*x+0.0010*x^2, 0)}
f04 <- function(x){ifelse(x>0, 0.2853+1.1955*x+0.0010*x^2, 0)}
f05 <- function(x){ifelse(x>0, -0.0001+1.4159*x+0.0010*x^2, 0)}
f06 <- function(x){ifelse(x>0, 0.1065+1.1371*x+0.0010*x^2, 0)}
f07 <- function(x){ifelse(x>0, 0.6394+0.9114*x+0.0010*x^2, 0)}
f08 <- function(x){ifelse(x>0, 0.5564+0.9931*x+0.0010*x^2, 0)}
f09 <- function(x){ifelse(x>0, 0.9492+1.0683*x+0.0010*x^2, 0)}
f10 <- function(x){ifelse(x>0, 2.3430+0.5102*x+0.0010*x^2, 0)}
f11 <- function(x){ifelse(x>0, 1.8956+0.7345*x+0.0010*x^2, 0)}
f12 <- function(x){ifelse(x>0, 1.8750+0.6203*x+0.0010*x^2, 0)}
f13 <- function(x){ifelse(x>0, 1.8750+0.6203*x+0.0010*x^2, 0)}

lum02<-as.data.frame(sapply(lum02,f02))
lum02[lum02 > 63] <- 63 #truncate if values are higher than 63
lum03<-as.data.frame(sapply(lum03,f03))
lum03[lum03 > 63] <- 63 #truncate if values are higher than 63
lum04<-as.data.frame(sapply(lum04,f04))
lum04[lum04 > 63] <- 63 #truncate if values are higher than 63
lum05<-as.data.frame(sapply(lum05,f05))
lum05[lum05 > 63] <- 63 #truncate if values are higher than 63
lum06<-as.data.frame(sapply(lum06,f06))
lum06[lum06 > 63] <- 63 #truncate if values are higher than 63
lum07<-as.data.frame(sapply(lum07,f07))
lum07[lum07 > 63] <- 63 #truncate if values are higher than 63
lum08<-as.data.frame(sapply(lum08,f08))
lum08[lum08 > 63] <- 63 #truncate if values are higher than 63
lum09<-as.data.frame(sapply(lum09,f09))
lum09[lum09 > 63] <- 63 #truncate if values are higher than 63
lum10<-as.data.frame(sapply(lum10,f10))
```

```
lum10[lum10 > 63] <- 63 #truncate if values are higher than 63
lum11<-as.data.frame(sapply(lum11,f11))
lum11[lum11 > 63] <- 63 #truncate if values are higher than 63
lum12<-as.data.frame(sapply(lum12,f12))
lum12[lum12 > 63] <- 63 #truncate if values are higher than 63
lum13<-as.data.frame(sapply(lum13,f13))
lum13[lum13 > 63] <- 63 #truncate if values are higher than 63
#Create time variable for lumYY data.frame
lum02$time<-2002;lum08$time<-2008
lum03$time<-2003;lum09$time<-2009
lum04$time<-2004;lum10$time<-2010
lum05$time<-2005;lum11$time<-2011
lum06$time<-2006;lum12$time<-2012
lum07$time<-2007;lum13$time<-2013
#Bind luminosity and coordinates values together
lum02<-cbind(xy,lum02);lum08<-cbind(xy,lum08)
lum03<-cbind(xy,lum03);lum09<-cbind(xy,lum09)
lum04<-cbind(xy,lum04);lum10<-cbind(xy,lum10)
lum05<-cbind(xy,lum05);lum11<-cbind(xy,lum11)
lum06<-cbind(xy,lum06);lum12<-cbind(xy,lum12)
lum07<-cbind(xy,lum07);lum13<-cbind(xy,lum13)
#Renaming variables
names(lum02) <- c("lon","lat","lum", "time");names(lum09) <- c("lon","
    lat","lum", "time")
names(lum03) <- c("lon","lat","lum", "time");names(lum10) <- c("lon","
    lat","lum", "time")
names(lum04) <- c("lon","lat","lum", "time");names(lum11) <- c("lon","
    lat","lum", "time")
names(lum05) <- c("lon","lat","lum", "time");names(lum12) <- c("lon","
    lat","lum", "time")
names(lum06) <- c("lon","lat","lum", "time");names(lum13) <- c("lon","
    lat","lum", "time")
names(lum07) <- c("lon","lat","lum", "time")
names(lum08) <- c("lon","lat","lum", "time")
#Merging vertically all databases: lat,lon,lum,time (4 columns)
lum<-rbind(lum02,lum03,lum04,lum05,lum06,lum07,lum08,lum09,lum10,lum11,
    lum12,lum13)
lum<-as.data.frame(lum)
sd(lum$lum)#sd
mean(lum$lum)#mean
```

```
#Normalising data (for better INLA process): luminosity mean=0, sd=1 and
    rearrange data frame
lum$lum<-scale(lum$lum)
lum <- lum[c(4,1,2,3)]#order columns as: time, lon, lat, lum
lum <- lum[order(lum$time,lum$lon,lum$lat),]
#Extract lat, lon, and year from GTD and create a database which
    represent the response vector
GTDresp<-GTDlethal[,c(3,1,2)]#extract year,lat, long
#Add id unique values in GTDresp to sort it later
id <- c(1:nrow(GTDresp))
GTDresp <- cbind(id=id, GTDresp)
names(GTDresp)<-c("id","time","lat","lon")
GTDresp$time<-as.numeric(GTDresp$time)
GTDresp <- GTDresp[c(1,2,4,3)]#order columns as: id, time, lon, lat
#Match luminosity with the response vector based on GTD locations
GTDlum <- merge(GTDresp,lum,by.x=c("time","lon","lat"), by.y=c("time","
    lon","lat"),all.x=TRUE)
GTDlum <- subset(GTDlum, !duplicated(GTDlum[,4])) #remove duplicate
    observations from id (unique cannot be used since we can have
    identical lat,lon,lum!)
GTDlum<- GTDlum[order(GTDlum$id),]
#Putting lum as a vector
lum<-as.vector(GTDlum$lum)
#test plot to check distance values
#create dataframe
# GTDlethallum<-cbind(GTDlethal,lum)
# lum08<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/
    PhD/PhD_R/luminosity/lum08.tif")
# projection(lum08)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
# plot.new()
# rbPal <- colorRampPalette(c('blue','green','yellow','orange','red','
    brown'))
# GTDlethallum$Col <- rbPal(6)[as.numeric(cut(GTDlethallum$lum,breaks =
    6))]
# par(mfrow=c(2,1))
# plot(GTDlethallum$longitude,GTDlethallum$latitude,pch = 20,cex=0.3,col
    = GTDlethallum$Col)
# plot(world,add=T)
# plot(lum08,ylim=c(-45,65),xlim=c(-180,180))
```

```
# plot(world,add=T)
#End luminosity (spatio-temporal covariate)

#Travel time covariate (spatial not temporal)
ttime <- raster("C:/Users/apython/Documents/Andre/Education/StAndrews/
    PhD/PhD_ArcGIS/traveltime/access_50k/acc_50k")
projection(ttime)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
#create lon lat in matrix form
xy<-cbind(GTDlethal[,2],GTDlethal[,1])#select the 3rd and 2nd variable
    of GTD
#extract coordinates from population at the location of the points
    coordinates
tt<-extract(ttime,xy,method='bilinear')#extract covariate value at point
     locations
tt<-as.vector(tt)
#replace NA values by 0
tt[is.na(tt)] <- 0
tt<-scale(tt)#normalising data (mean=0, sd=1) for better INLA process

#test plot to check distance values
#create dataframe
# GTDtt<-cbind(GTD1,tt)
# plot.new()
# rbPal <- colorRampPalette(c('blue','green','yellow','orange','red','
    brown'))
# GTDtt$Col <- rbPal(6)[as.numeric(cut(GTDtt$tt,breaks = 6))]
# plot(GTDtt$longitude,GTDtt$latitude,pch = 20,cex=0.3,col = GTDtt$Col)
# plot(world,add=T)
# plot(ttime,ymin=c(-45,65))
# points(GTD1[,8],GTD1[,7],add=T,cex=0.1)
#End travel time covariate (spatial not temporal)

#Polity IV covariate (country-level + temporal)
#read the original file (saved in .csv) from http://www.systemicpeace.
    org/inscrdata.html
polity<- read.csv("C:/Users/apython/Documents/Andre/Education/StAndrews/
    PhD/PhD_R/PolityIV/p4v2013.csv",header = TRUE)
polity<-subset(polity, select=c("country","year","polity2"))
```

```
polity <-subset(polity, year > 2001 & year < 2014)#selecting the same
    time period than GTD
#rename year as iyear for compatibility with GTD
require(reshape)
polity<-rename(polity, c(year="iyear"))
#add polity IV variable (polity2 has been adjusted for time-series
    analysis) to GTD data
#note that Bahamas, Belize, Iceland, Maldives,West Bank and Gaza Strip,
    are not present in polity
#change some country names in polity before merging with GTD
polity$country <- as.character(polity$country)
polity$country[polity$country == "Bosnia"] <-"Bosnia-Herzegovina"
polity$country[polity$country == "Congo␣Brazzaville"] <-"Congo␣(
    Brazzaville)"
polity$country[polity$country == "Congo␣Kinshasa"] <-"Congo␣(Kinshasa)"
polity$country[polity$country == "Dominican␣Rep"] <-"Dominican␣Republic"
polity$country[polity$country == "Serbia␣and␣Montenegro"] <-"Serbia-
    Montenegro"
#polity$country[polity$country == "Korea North"] <-"North Korea"
polity$country[polity$country == "Korea␣South"] <-"South␣Korea"
polity$country[polity$country == "East␣Timor"] <-"Timor-Leste"
polity$country[polity$country == "UAE"] <-"United␣Arab␣Emirates"
polity$country[polity$country == "Myanmar␣(Burma)"] <-"Myanmar"
#create new observations based on equivalent countries and add to polity
     dataframe
uk<-subset(polity, country == "United␣Kingdom")
uk$country[uk$country == "United␣Kingdom"] <-"Great␣Britain"
polity<-rbind(polity,uk)
ni<-subset(polity, country == "Great␣Britain")
ni$country[ni$country == "Great␣Britain"] <-"Northern␣Ireland"
polity<-rbind(polity,ni)
rm(uk);rm(ni)
ch<-subset(polity, country == "China")
ch$country[ch$country == "China"] <-"Hong␣Kong"
polity<-rbind(polity,ch)
rm(ch)
is<-subset(polity, country == "Israel")
is$country[is$country == "Israel"] <-"West␣Bank␣and␣Gaza␣Strip"
polity<-rbind(polity,is)
rm(is)
```

```
cs<-subset(polity, country == "France")
cs$country[cs$country == "France"] <-"Corsica"
polity<-rbind(polity,cs)
rm(cs)
#merging with GTD
GTDpolity<-merge(GTDlethal,polity,by=c("country","iyear"),all.x=TRUE)
pol<-GTDpolity[,c(7)]#extract polity
#replace NA values by 0 (warning polityIV=0 means neither demo nor auto
    -cratic)
pol[is.na(pol)] <- 0
pol<-scale(pol)#normalising data (mean=0, sd=1) for better INLA process
pol<-as.vector(pol)
#END Polity IV covariate (country-level + temporal)

#GREG ethnic group covariate (spatial not temporal)
#import GREG file from Weidmann ETHZ: http://www.icr.ethz.ch/data/other/
    greg
GREG <- readShapeSpatial("C:/Users/apython/Documents/Andre/Education/
    StAndrews/PhD/PhD_R/GREG/GREG.shp")
proj4string(GREG)<-CRS("+proj=longlat␣+datum=WGS84␣+no_defs+towgs84
    =0,0,0+ellps=WGS84")
#create vector shape with GTD1
GTDpt<- SpatialPoints(GTDlethal[, c("longitude", "latitude")])
proj4string(GTDpt)<-CRS("+proj=longlat␣+datum=WGS84␣+no_defs+towgs84
    =0,0,0+ellps=WGS84")
#intersect values of GREG with points of GTD
o <- over(GTDpt,GREG)
#recombine points with attributes from GRM
GTDpt<-cbind(GTDpt,o)
#add id unique values in GTDresp to sort it later
id <- c(1:nrow(GTDpt))
GTDGREG <- cbind(id=id, GTDpt)
#keep only id and number of ethnic groups
GTDGREG<-GTDGREG[,c(1,8,9,10)]
#if no ethnic group (0) put NA in order that the absence of ethnic group
     will not be counted
GTDGREG[GTDGREG == 0] <- NA
GTDGREG<-ddply(GTDGREG, .(id), mutate, count = length(unique(na.omit(c(
    G1ID,G2ID,G3ID)))))
#sort(already done but for double check)
```

```
GTDGREG <- GTDGREG[order(id),]
#scale
GTDGREG$count<-scale(GTDGREG$count)
#keep a vector
greg<-as.vector(GTDGREG$count)
#End GREG ethnic group covariate (spatial not temporal)

#Altitude covariate (spatial not temporal)
#download lDEM map (ETOPO1): https://www.ngdc.noaa.gov/mgg/global/relief
    /ETOPO1/image/
altitude <- raster("C:/Users/apython/Documents/Andre/Education/StAndrews
    /PhD/PhD_ArcGIS/DEM/ETOPO1/color_etopo1_ice_full.tif")
projection(altitude)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84
    =0,0,0+ellps=WGS84"
#plot(altitude)
#extract coordinates from population at the location of the points
    coordinates
alt<-extract(altitude,xy,method='bilinear')#extract covariate value at
    point locations
alt<-as.vector(alt)
#replace NA values by 0
alt[is.na(alt)] <- 0
alt<-scale(alt)#normalising data (mean=0, sd=1) for better INLA process
#test plot to check distance values
#create dataframe
# GTDalt<-cbind(GTD1,alt)
# plot.new()
# rbPal <- colorRampPalette(c('blue','green','yellow','orange','red','
    brown'))
# GTDalt$Col <- rbPal(6)[as.numeric(cut(GTDalt$alt,breaks = 6))]
# plot(GTDalt$longitude,GTDalt$latitude,pch = 20,cex=0.3,col = GTDalt$
    Col)
# plot(world,add=T)
#End altitude covariate (spatial not temporal)

#Slope covariate (spatial not temporal)
require(insol)
slope<-slope(cgrad(altitude), degrees = TRUE)
slope=raster(slope,crs=projection(altitude))
extent(slope)=extent(altitude)
```

```
#extract coordinates from population at the location of the points
    coordinates
slo<-extract(slope,xy,method='bilinear')#extract covariate value at
    point locations
slo<-as.vector(slo)
#replace NA values by 0
slo[is.na(slo)] <- 0
slo<-scale(slo)#normalising data (mean=0, sd=1) for better INLA process
#test plot to check distance values
#create dataframe
# GTDslo<-cbind(GTD1,slo)
# plot.new()
# rbPal <- colorRampPalette(c('blue','green','yellow','orange','red','
    brown'))
# GTDslo$Col <- rbPal(6)[as.numeric(cut(GTDslo$slo,breaks = 6))]
# plot(GTDslo$longitude,GTDslo$latitude,pch = 20,cex=0.3,col = GTDslo$
    Col)
# plot(world,add=T)
# slopew <- mask(slope, world.map)
# plot.new()
# plot(slopew,col= terrain.colors(12))
#End slope covariate (spatial not temporal)

#Distance to country border (spatial)
GTDp<-ppp(GTDlethal[,2],GTDlethal[,1],c(-180,180), c(-90,90))
#download simple map with national borders from natural earth data
    (1:110) instead of Digital chart of the world 2000 (country borders)-
    too complex
world.map <- readShapeSpatial("C:/Users/apython/Documents/Andre/
    Education/StAndrews/PhD/PhD_ArcGIS/World␣map/ne_110m_admin_0_
    countries.shp")
proj4string(world.map)<-CRS("+proj=longlat␣+datum=WGS84␣+no_defs+towgs84
    =0,0,0+ellps=WGS84")
world<-as.owin(world.map)
world = edges(world.map)
distb<-nncross(GTDp,world)
distb<-as.vector(distb$dist)
#distb[is.na(distb)] <- 0
distb<-scale(distb)#normalising data (mean=0, sd=1) for better INLA
    process
```

```
#End distance to country border (spatial)

#Delete unuseful data
rm(GTDpolity);rm(polity);rm(xy);rm(lat);rm(lon);rm(pop00);rm(lum02);rm(
    lum03);rm(lum04);rm(ttime)
rm(lum05);rm(lum06);rm(lum07);rm(lum08);rm(lum09);rm(lum10);rm(lum11);rm
    (lum12);rm(lum13);
rm(v);rm(w);rm(nGTD);rm(id);rm(bdry);rm(world.map);rm(GTDresp);rm(GTDlum
    );rm(f02,f03,f04,f05,f06,f07,f08,f09,f10,f11,f12,f13)
rm(GTDGREG);rm(GTDpt);rm(o);rm(GREG);rm(altitude);rm(GTDdistb);rm(GTDalt
    );rm(GTDp);rm(world);rm(rbPal);rm(GTDtt);rm(GTD1lum);
rm(GTDcountry)
rm(slope)
#Save all data into
save.image("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/SPDE_nbevents_
    final.RData")
```

C.6 Poisson Models: Specification

This code generate all investigated Poisson models of the number of lethal attacks.

```
#Goal is to generate several Poisons models which will be compared.
#Preliminary note: a linux configuration with high available RAM is
    recommended
#to be able to run the models with R-INLA.
#define working directory
setwd("/home/ap215/SPDE")#NTNU home file
library(INLA)
#load data generated in R-code (3/5): study area, mesh, covariates, GTD
load("/home/ap215/SPDE/SPDE_nbevents_final.RData")
#delete the country variable (not needed)
GTDlethal<-GTDlethal[,1:5]
#keep coordinates and number of lethal attacks (variable lethal)
data<-cbind(loc,GTDlethal[,5])
y<-data[,4]#response as the number of lethal attacks
time<-GTDlethal[,3]-2001#create time index
k<-12#the number of years
#SPDE space-time model
nv<-mesh$n#number of mesh vertices
spde <- inla.spde2.matern(mesh, alpha=2)
```

```
iset<-inla.spde.make.index('i',n.spde=spde$n.spde,n.group=k)
A<-inla.spde.make.A(mesh,loc=loc, group=time)
#0 spatial covariate
stk <- inla.stack(data=list(y=y), A=list(A,1), tag='dat',
                  effects=list(i=iset,b0=rep(1,length(y))))
res0 <- inla(y ~ 0+b0+f(i, model=spde,group=i.group,control.group=list(
    model="ar1")),
             family= 'poisson',data=inla.stack.data(stk),control.compute
                =list(dic=TRUE,waic=TRUE),
             control.predictor=list(A=inla.stack.A(stk), compute=TRUE),
                verbose=TRUE
             ,num.threads=1
)
#1 spatial covariate (lum)
stk <- inla.stack(data=list(y=y), A=list(A,1,1), tag='dat',
                  effects=list(i=iset,lum=lum,b0=rep(1,length(y))))
res1 <- inla(y ~ 0+b0+lum+f(i, model=spde,group=i.group,control.group=
    list(model="ar1")),
             family= 'poisson',data=inla.stack.data(stk),control.compute
                =list(dic=TRUE,waic=TRUE),
             control.predictor=list(A=inla.stack.A(stk), compute=TRUE),
                verbose=TRUE
             ,num.threads=1
)
#2 spatial covariates (tt + lum)
stk <- inla.stack(data=list(y=y), A=list(A,1,1,1), tag='dat',
                  effects=list(i=iset,lum=lum,tt=tt,b0=rep(1,length(y)))
                      )
res2 <- inla(y ~ 0+b0+lum+tt+f(i, model=spde,group=i.group,control.group
    =list(model="ar1")),
             family= 'poisson',data=inla.stack.data(stk),control.compute
                =list(dic=TRUE,waic=TRUE),
             control.predictor=list(A=inla.stack.A(stk), compute=TRUE),
                verbose=TRUE
             ,num.threads=1
)
#3 spatial covariates (pol + tt + lum)
stk <- inla.stack(data=list(y=y), A=list(A,1,1,1,1), tag='dat',
                  effects=list(i=iset,lum=lum,tt=tt,pol=pol,b0=rep(1,
                      length(y))))
```

```
res3 <- inla(y ~ 0+b0+lum+tt+pol+f(i, model=spde,group=i.group,control.
    group=list(model="ar1")),
                  family= 'poisson',data=inla.stack.data(stk),control.
                      compute=list(dic=TRUE,waic=TRUE),
                  control.predictor=list(A=inla.stack.A(stk), compute=
                      TRUE),verbose=TRUE
                  ,num.threads=1
)
#4 spatial covariates (pol + tt + lum  + alt)
stk <- inla.stack(data=list(y=y), A=list(A,1,1,1,1,1), tag='dat',
                   effects=list(i=iset,lum=lum,tt=tt,pol=pol,alt=alt,b0=
                       rep(1,length(y))))
res4 <- inla(y ~ 0+b0+lum+tt+pol+alt+f(i, model=spde,group=i.group,
    control.group=list(model="ar1")),
                  family= 'poisson',data=inla.stack.data(stk),control.
                      compute=list(dic=TRUE,waic=TRUE),
                  control.predictor=list(A=inla.stack.A(stk), compute=
                      TRUE),verbose=TRUE
                  ,num.threads=1
)
#5 spatial covariates (pol + tt + alt+ lum + pop)
stk <- inla.stack(data=list(y=y), A=list(A,1,1,1,1,1,1), tag='dat',
                   effects=list(i=iset,lum=lum,tt=tt,pop=pop,alt=alt,pol=
                       pol,b0=rep(1,length(y))))
resfinal <- inla(y ~ 0+b0+lum+tt+pop+alt+pol+f(i, model=spde,group=i.
    group,control.group=list(model="ar1")),
                  family= 'poisson',data=inla.stack.data(stk),control.
                      compute=list(dic=TRUE,waic=TRUE),
                  control.predictor=list(A=inla.stack.A(stk), compute=
                      TRUE),verbose=TRUE
                  ,num.threads=1#(adjusted nb. cpu involved if necessary
                      to avoid crash)
)
save.image("STpoitotal.RData")#adjust directory
#robustness test (using alternative meshes)
bdry <- inla.sp2segment(studyarea)
bdry$loc <- inla.mesh.map(bdry$loc, projection = "longlat",inverse =
    TRUE)
#mesh for robusntess tests
```

```
mesh3<-inla.mesh.2d(boundary=bdry, max.edge=c(9,6000)/180,cutoff=9/180)#
    nv=1,341
mesh2<-inla.mesh.2d(boundary=bdry, max.edge=c(7,1000)/180,cutoff=7/180)#
    nv=2,157
#mesh2
nv<-mesh2$n#number of mesh vertices
spde <- inla.spde2.matern(mesh2, alpha=2)
iset<-inla.spde.make.index('i',n.spde=spde$n.spde,n.group=k)
A<-inla.spde.make.A(mesh2,loc=loc, group=time)
#5 spatial covariates (pol + tt + alt+ lum + pop)
stk <- inla.stack(data=list(y=y), A=list(A,1,1,1,1,1,1), tag='dat',
                  effects=list(i=iset,lum=lum,tt=tt,pop=pop,alt=alt,pol=
                      pol,b0=rep(1,length(y))))
poimesh2 <- inla(y ~ 0+b0+lum+tt+pop+alt+pol+f(i, model=spde,group=i.
    group,control.group=list(model="ar1")),
                 family= 'poisson',data=inla.stack.data(stk),control.
                     compute=list(dic=TRUE),
                 control.predictor=list(A=inla.stack.A(stk), compute=
                     TRUE),verbose=TRUE)
               #    ,num.threads=1#(adjusted nb. cpu involved if necessary
                     to avoid crash)
#)
#save models output
save.image("poimesh2.RData")#adjust directory
#mesh3
nv<-mesh3$n#number of mesh vertices
spde <- inla.spde2.matern(mesh3, alpha=2)
iset<-inla.spde.make.index('i',n.spde=spde$n.spde,n.group=k)
A<-inla.spde.make.A(mesh3,loc=loc, group=time)
#5 spatial covariates (pol + tt + alt+ lum + pop)
stk <- inla.stack(data=list(y=y), A=list(A,1,1,1,1,1,1), tag='dat',
                  effects=list(i=iset,lum=lum,tt=tt,pop=pop,alt=alt,pol=
                      pol,b0=rep(1,length(y))))
poimesh3 <- inla(y ~ 0+b0+lum+tt+pop+alt+pol+f(i, model=spde,group=i.
    group,control.group=list(model="ar1")),
                 family= 'poisson',data=inla.stack.data(stk),control.
                     compute=list(dic=TRUE),
                 control.predictor=list(A=inla.stack.A(stk), compute=
                     TRUE),verbose=TRUE)
```

```
#    ,num.threads=1#(adjusted nb. cpu involved if necessary to avoid
    crash)
#)
#save models output
save.image("poimesh3.RData")#adjust directory
#robustness test (using alternative priors for the GMRF)
#note: we use informative prior based on range = 100 km (i.e. 100/6371
    rad) and GMRF standard deviation (std) 50 km (i.e. 50/6371 rad).
#From Lindren and Rue (2015) we link theta with std and range through
    the process described in Blangiardo 2015 p. 214-5.
range0<-100/6371
sigma0<-50/6371
kappa0<-sqrt(8)/range0
tau0<-1/(sqrt(4*pi)*kappa0*sigma0)
#SPDE space-time model
spde <- inla.spde2.matern(mesh, alpha=2
                          ,B.tau=matrix(c(log(tau0),-1,+1),nrow=1,ncol
                              =3)
                          ,B.kappa=matrix(c(log(kappa0),0,-1),nrow=1,
                              ncol=3)
                          ,theta.prior.mean=c(0,0)
                          ,theta.prior.prec=c(0.1,0.1))

iset<-inla.spde.make.index('i',n.spde=spde$n.spde,n.group=k)
A<-inla.spde.make.A(mesh,loc=loc, group=time)
#Using 5 spatial covariates (pol + tt + alt+ lum + pop)
stk <- inla.stack(data=list(y=y), A=list(A,1,1,1,1,1,1), tag='dat',
                  effects=list(i=iset,lum=lum,tt=tt,pop=pop,alt=alt,pol=
                      pol,b0=rep(1,length(y))))# If problem:try this :
                      list(list(i=iset), list(cov=pop), list(b0=rep(1,
                      length(y)))
resprior <- inla(y ~ 0+b0+lum+tt+pop+alt+pol+f(i, model=spde,group=i.
    group,control.group=list(model="ar1")),
                 family= 'poisson',data=inla.stack.data(stk),control.
                     compute=list(dic=TRUE,waic=TRUE),
                 control.predictor=list(A=inla.stack.A(stk), compute=
                     TRUE),verbose=TRUE
                 ,num.threads=1#to reduce computational time
)
#save output from model with informative prior
```

```
save.image("STpoitotal.RData")#adjust directory
```

C.7 Poisson Models: Model Selection and Graphics

This code is used to select the Poisson models and produce figures.

```
#SPATIO-TEMPORAL MODELLING / Poisson model of the number of lethal
    events worldwide 2002-2013
#This codes generate graphics of the random field (mean + std.dev) and
    the expected number of lethal attacks based on the output of STpoi_
    totalv2.R
#The model to be loaded needs to be the selected one.
setwd("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson")
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/SPDE_nbevents
    _final.RData")
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal/Llinpal
    .RData")#load custom palette
require(foreign)
require(sp)
library(INLA)
library(spatstat)
library(fields)
library(maptools)
library(fields)
library(lattice)
#Model selection (DIC values comparison)
#load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/STpoitotal.
    RData")
load("C:/Users/apython/Documents/Andre/Education/StAndrews/arXiv/
    RSSstylefile/R_code/STpoitotal.RData")
load("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/PhD_R/
    SPDE/STpoisson/STpoi0.RData")
summary(res0)#0 var model: DIC 28,395 /WAIC: 32637
summary(res3)#3 var model: DIC 24,203 /WAIC: 26,466
summary(res4)#4 var model: DIC 24,189 /WAIC: 26,438
#model with lower DIC
summary(resfinal);#full model 5 variables: DIC 24,164 / WAIC: 26,418
resfinal$cpu.used
res3$cpu.used
res4$cpu.used
```

```
res0$cpu.used
#comparison with alternative meshes
round(resfinal$summary.fixed,3)#B0-,lum+,tt-,pop/,alt/,pol+
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/poimesh2.
    RData")#3 var model
round(poimesh2$summary.fixed,3)#B0-,lum+,tt-,pop+,alt-,pol+
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/poimesh3.
    RData")#3 var model
round(poimesh3$summary.fixed,3)#B0-,lum+,tt-,pop+,alt/,pol+
#with manual prior (elicited)
summary(resprior)
#convert from Cartesian xyz to latitude-longitude
lat = asin(mesh$loc[,3])
lon = atan2(mesh$loc[,2], mesh$loc[,1])
lat<-lat*180/pi
lon<-lon*180/pi #from radian to degree
coord=cbind(lon,lat)
sp = SpatialPoints(coord)
longlat<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+ellps=WGS84"
proj4string(sp) = CRS(longlat)
#plot(studyarea,col="red")
#points(coord,cex=0.4,pch="+")
#rmse
GTDlethal$lum<-as.numeric(lum)
GTDlethal$alt<-as.numeric(alt)
GTDlethal$distb<-as.numeric(distb)
GTDlethal$pop<-as.numeric(pop)
GTDlethal$pol<-as.numeric(pol)
GTDlethal$tt<-as.numeric(tt)
GTDlethal$greg<-as.numeric(greg)
GTDlethal$time<-GTDlethal$iyear-2001
locdf<-as.data.frame(loc)
GTDlethal$loc1<-as.numeric(locdf[,1])
GTDlethal$loc2<-as.numeric(locdf[,2])
GTDlethal$loc3<-as.numeric(locdf[,3])
mean4<-list();cov4<-list();prob4<-list()
mean0<-list();cov0<-list();prob0<-list()
mean3<-list();cov3<-list();prob3<-list()
mean5<-list();cov5<-list();prob5<-list()
loc<-list()
```

```
for (j in 1:k){
  loc[[j]]<-as.matrix(cbind(GTDlethal$loc1[GTDlethal$time==j],GTDlethal$
      loc2[GTDlethal$time==j],GTDlethal$loc3[GTDlethal$time==j]))
    #final model
  cov4[[j]]<-res4$summary.fix[1,1] +res4$summary.fix[2,1]*GTDlethal$lum[
      GTDlethal$time==j]+res4$summary.fix[3,1]*GTDlethal$tt[GTDlethal$
      time==j] + res4$summary.fix[4,1]*GTDlethal$pol[GTDlethal$time==j] +
       res4$summary.fix[5,1]*GTDlethal$alt[GTDlethal$time==j]
  mean4[[j]]<-inla.mesh.project(inla.mesh.projector(mesh, loc=loc[[j]]),
      res4$summary.random$i$mean[iset$i.group==j])
  prob4[[j]]<-poisson(link='log')$linkinv(mean4[[j]]+cov4[[j]])
  #Zero cov model
  cov0[[j]]<-res0$summary.fix[1,1]
  mean0[[j]]<-inla.mesh.project(inla.mesh.projector(mesh, loc=loc[[j]]),
      res0$summary.random$i$mean[iset$i.group==j])
  prob0[[j]]<-poisson(link='log')$linkinv(mean0[[j]]+cov0[[j]])
  #3 cov model
  cov3[[j]]<-res3$summary.fix[1,1] +res3$summary.fix[2,1]*GTDlethal$lum[
      GTDlethal$time==j]+res3$summary.fix[3,1]*GTDlethal$tt[GTDlethal$
      time==j] + res3$summary.fix[4,1]*GTDlethal$pol[GTDlethal$time==j]
  mean3[[j]]<-inla.mesh.project(inla.mesh.projector(mesh, loc=loc[[j]]),
      res3$summary.random$i$mean[iset$i.group==j])
  prob3[[j]]<-poisson(link='log')$linkinv(mean3[[j]]+cov3[[j]])
  #5 cov model
  cov5[[j]]<-resfinal$summary.fix[1,1] +resfinal$summary.fix[2,1]*
      GTDlethal$lum[GTDlethal$time==j]+resfinal$summary.fix[3,1]*
      GTDlethal$tt[GTDlethal$time==j] + resfinal$summary.fix[4,1]*
      GTDlethal$pop[GTDlethal$time==j] + resfinal$summary.fix[5,1]*
      GTDlethal$alt[GTDlethal$time==j]+ resfinal$summary.fix[6,1]*
      GTDlethal$pol[GTDlethal$time==j]
  mean5[[j]]<-inla.mesh.project(inla.mesh.projector(mesh, loc=loc[[j]]),
      resfinal$summary.random$i$mean[iset$i.group==j])
  prob5[[j]]<-poisson(link='log')$linkinv(mean5[[j]]+cov5[[j]])
}
#unlist
GTDlethal$prob4<-unlist(prob4)
GTDlethal$prob5<-unlist(prob5)
GTDlethal$prob3<-unlist(prob3)
GTDlethal$prob0<-unlist(prob0)
rmsedf<-GTDlethal
```

```
rmsedf<-rmsedf[c("latitude","longitude","total","prob4","prob5","prob3",
    "prob0")]
colnames(rmsedf) <- c("lat","lon","obs","prob4","prob5","prob3","prob0")
#remove if NA
rmsedf<-rmsedf[complete.cases(rmsedf),]
rmseprob4<-sqrt(mean((rmsedf$obs-rmsedf$prob4)^2,na.rm=T)) ;rmseprob4
rmseprob5<-sqrt(mean((rmsedf$obs-rmsedf$prob5)^2,na.rm=T));rmseprob5
rmseprob3<-sqrt(mean((rmsedf$obs-rmsedf$prob3)^2,na.rm=T));rmseprob3
rmseprob0<-sqrt(mean((rmsedf$obs-rmsedf$prob0)^2,na.rm=T));rmseprob0
maeprob4<-mean(abs(rmsedf$obs-rmsedf$prob4),na.rm=T) ;maeprob4
maeprob5<-mean(abs(rmsedf$obs-rmsedf$prob5),na.rm=T) ;maeprob5
maeprob3<-mean(abs(rmsedf$obs-rmsedf$prob3),na.rm=T) ;maeprob3
maeprob0<-mean(abs(rmsedf$obs-rmsedf$prob0),na.rm=T) ;maeprob0
```

```
#plot on 2d (projection)
proj = inla.mesh.projector(mesh, projection = "longlat",dims=c(1444,724)
    )
win<-as.owin(studyarea)
library(mgcv)
e<-expand.grid(proj$x,proj$y)
ins<-inside.owin(e[,1],e[,2],win)
ins<-matrix(ins,nrow=length(proj$y))

xmean<-list()
for (j in 1:k){
  xmean[[j]]<-inla.mesh.project(proj,resfinal$summary.random$i$mean[iset
      $i.group==j])#update file location according to model selection
}
for (j in 1:k){
  xmean[[j]][!ins]<-NA
```

```
}
#RECHECK THIS!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
probsurf<-list()
for(j in 1:k){
  probsurf[[j]]<-log(poisson(link='log')$linkinv(resfinal$summary.fix
      [1,1]+resfinal$summary.fix[2,1]
   +resfinal$summary.fix[3,1]#using log before (poisson(link=...to have
       better scale
  +resfinal$summary.fix[4,1]+resfinal$summary.fix[5,1]+resfinal$summary.
      fix[6,1]
   +xmean[[j]])#update file location according to model selection
  )
}
#plot prob. surface for year 2002
dev.off()
plot.new()
breaks<-seq(min(probsurf[[12]],na.rm=TRUE),max(probsurf[[12]],na.rm=TRUE
    ),by=0.1)
n<-length(breaks)-1
par(cex=3,mar=c(5, 4, 2, 4.5) + 0.1)#mar: c(bottom,left,top,right)
image.plot(proj$x,proj$y,probsurf[[1]],col=Llinpal(n),xlab="Longitude",
           main= 2002,sub="Number␣of␣lethal␣attack␣(log)" ,ylab="
               Latitude",
           xlim = c(-180, 180), ylim = c(-60, 90),zlim=c(-8,8),#zlim
               creates identical scale
           breaks=breaks)#should have one more break than color
#plot all years from 2002 to 2013 and save in file all pictures (30
    pictures)

for(i in 1:12){
  mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD
      /Latex/phd-thesis-template-phd-latex-template-latest-stable/
      Chapter5/Figs/Raster/pprobsurf",i,".png",sep="")
  png(file=mypath,width=1444,height=724)
  par(cex=3,mar=c(5, 4, 2, 3.5) + 0.1)#mar: c(bottom,left,top,right)
  image.plot(proj$x,proj$y,probsurf[[i]],col=Llinpal(n),xlab="Longitude"
      ,
   main="Number␣of␣lethal␣attack␣(log)",sub=2001+i,ylab="Latitude",
   xlim = c(-180, 180), ylim = c(-60, 90),zlim=c(-8,8),
   breaks=breaks)
```

```
  dev.off()
}

dev.off()
plot.new()
for(i in 1:12){
  mypath=paste("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/
      probsurf/pprobsurf",i,".eps",sep="")
  postscript(file=mypath, horiz=FALSE, onefile=FALSE, width=21, height=8,
      paper="a4")
  par(cex=3,mar=c(3.75, 0.1, 1.75, 3.4) + 0.1)#mar: c(bottom, left, top,
      right) changed 'mar' (original: mar=c(1.75, 0.1, 1.75, 3.4)) to be
      in line with random mean and sd pictures
  image.plot(proj$x, proj$y, probsurf[[i]], col=Llinpal(n), xlab=NA, ylab=NA,
             main=NA, axes=FALSE,
             xlim = c(-180, 180), ylim = c(-60, 90), zlim=c(min(probsurf
                 [[12]], na.rm=TRUE), max(probsurf[[12]], na.rm=TRUE)),
             breaks=breaks, legend.shrink=1.1)
  dev.off()
}
#mean of the random field
#plot all years from 2002 to 2013 and save in file all pictures (30
    pictures)
minrf<-min(sapply(xmean, min, na.rm=TRUE))
maxrf<-max(sapply(xmean, max, na.rm=TRUE))
brk<-seq(minrf, maxrf, by=0.1)
n<-length(brk)-1
for(i in 1:12){
  mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD
      /Latex/phd-thesis-template-phd-latex-template-latest-stable/
      Chapter5/Figs/Raster/pbirf",i,".png",sep="")
  png(file=mypath, width=1444, height=724)
  par(cex=3,mar=c(5, 4, 2, 3.5) + 0.1)#mar: c(bottom, left, top, right)
  image.plot(proj$x, proj$y, xmean[[i]], col=Llinpal(n), xlab="Longitude",
             main="Mean␣of␣the␣Random␣field", sub=2001+i, ylab="Latitude",
             xlim = c(-180, 180), ylim = c(-60, 90), zlim=c(minrf, maxrf),
             breaks=brk)
  dev.off()
}
```

```
minrf<-min(sapply(xmean, min,na.rm=TRUE))
maxrf<-max(sapply(xmean, max,na.rm=TRUE))
brk<-seq(minrf,maxrf,by=0.01)
n<-length(brk)-1

for(i in 1:12){
  mypath=paste("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/
      birf/pbirf",i,".eps",sep="")
  postscript(file=mypath,horiz=FALSE,onefile=FALSE,width=21,height=8,
      paper="a4")
  par(cex=3,mar=c(1.75, 0.1, 1.75, 3.4) + 0.1)#mar: c(bottom,left,top,
      right)
  image.plot(proj$x,proj$y,xmean[[i]],col=Llinpal(n),xlab=NA,ylab=NA,
             main=NA,axes=FALSE,zlim=c(minrf,maxrf),
             breaks=brk)
  dev.off()
}
#standard deviation of the random field
xsd<-list()
for (j in 1:k){
  xsd[[j]]<-inla.mesh.project(proj,resfinal$summary.random$i$sd[iset$i.
      group==j])
}
for (j in 1:k){
  xsd[[j]][!ins]<-NA
}
minsd<-min(sapply(xsd, min,na.rm=TRUE))
maxsd<-max(sapply(xsd, max,na.rm=TRUE))
brk<-seq(minsd,maxsd,by=0.05)
n<-length(brk)-1
dev.off()
plot.new()
for(i in 1:12){
  mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD
      /Latex/phd-thesis-template-phd-latex-template-latest-stable/
      Chapter5/Figs/Raster/pbisdrf",i,".png",sep="")
  png(file=mypath,width=1444,height=724)
  par(cex=3,mar=c(5, 4, 2, 3.5) + 0.1)#mar: c(bottom,left,top,right)
  image.plot(proj$x,proj$y,xsd[[i]],col=Llinpal(n),xlab="Longitude",
```

```
                main="Standard␣deviation␣of␣the␣Random␣field",sub=2001+i,
                    ylab="Latitude",
                xlim = c(-180, 180), ylim = c(-60, 90),zlim=c(minsd,maxsd),
                breaks=brk)
  dev.off()
}

minsd<-min(sapply(xsd, min,na.rm=TRUE))
maxsd<-max(sapply(xsd, max,na.rm=TRUE))
brk<-seq(minsd,maxsd,by=0.01)
n<-length(brk)-1
for(i in 1:12){
  mypath=paste("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/
      bisdrf/pbisdrf",i,".eps",sep="")
  postscript(file=mypath,horiz=FALSE,onefile=FALSE,width=21,height=8,
      paper="a4")
  par(cex=3,mar=c(1.75, 0.1, 1.75, 3.4) + 0.1)#mar: c(bottom,left,top,
      right)
  image.plot(proj$x,proj$y,xsd[[i]],col=Llinpal(n),xlab=NA,ylab=NA,
                main=NA,axes=FALSE,zlim=c(minsd,maxsd),
                breaks=brk)
  dev.off()
}
#graph (parameter posterior densities)
#intercept
plot.new()
par(mfrow=c(1,2), mar=c(3,3.5,0,0), mgp=c(1.5, .5, 0), las=0)
plot(resfinal$marginals.fix[[1]], type='l', xlab='Intercept', ylab='
    Posterior␣density')#update file location according to model selection
rfsd<-density(resfinal$summary.random$i[[3]])#, xlab='sd of the random
    field', ylab='Posterior density')
plot(rfsd,main='',xlab='standard␣deviation␣of␣the␣random␣field', ylab='
    Posterior␣density')
#Beta coefficients
round(resfinal$summary.fixed[,1:5],3)
#temporal and spatial parameters
summary(resfinal)
#Spatial parameters with nominal scale (time is aggregated)
spde.resfinal <- inla.spde2.result(resfinal, name="i",spde,do.transform=
    TRUE)#do.transform put in correct scale
```

```
#kappa /computed using Blangiardo & Cameletti (2015)
Kappa<-inla.emarginal(function(x) x, spde.resfinal$marginals.kappa[[1]])
    #kappa (mean)
Kappahpd<-inla.hpdmarginal(0.95, spde.resfinal$marginals.kappa[[1]])#
    kappa (hpd 95%)
#variance of the random field
variance<-inla.emarginal(function(x) x, spde.resfinal$marginals.variance
    .nominal[[1]])#variance (mean)
variancehpd<-inla.hpdmarginal(0.95, spde.resfinal$marginals.variance.
    nominal[[1]])#variance (hpd 95%)
#range in radian
range<-inla.emarginal(function(x) x, spde.resfinal$marginals.range.
    nominal[[1]])#range (mean)
rangehpd<-inla.hpdmarginal(0.95, spde.resfinal$marginals.range.nominal
    [[1]])#range (hpd 95%)
#Matern
matern <- function(lambda,kappa,dist)
  2^(1-lambda)/gamma(lambda) * (kappa*dist)^lambda* besselK(x=dist*kappa
      , nu=lambda)
  dist.x = seq(0,0.31,l=100)
lambda<-1
kappa<-c(Kappahpd[,1],Kappa,Kappahpd[,2])

plot.new()
dev.off()
mypath<-"~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/Matern2.
    eps"
postscript(file=mypath,horiz=FALSE,onefile=FALSE,width=0,height=0,paper=
    "default")
par(cex=3.0,mar=c(4, 4, 1.0, 1.0) + 0.1,mgp=c(2.3,1,0))#mar: c(bottom,
    left,top,right)
plot(dist.x,matern(lambda,kappa=kappa[2],dist.x),lwd=3,type="l",ylab="
    Matrn␣covariance␣function",xlab="distance␣[km]",lty=1,xaxt = 'n')
lines(dist.x,matern(lambda,kappa=kappa[1],dist.x),lwd=3,type="l",ylab=NA
    ,xlab=NA,lty=2)
lines(dist.x,matern(lambda,kappa=kappa[3],dist.x),lwd=3,type="l",ylab=NA
    ,xlab=NA,lty=2)
abline( a = 0.1,b=0, v = NULL, reg = NULL,lty=3,lwd=2)#for vertical
    value v=range
abline( v =range, lty=3,lwd=2)#for vertical value v=range
```

```
axis(1, at=seq(0,2000/6371,500/6371), labels=c(0,500,1000,1500,2000))#1
    rad=6371km, 2000 km=2000/6371rad
dev.off()

#for thesis
plot.new()
dev.off()
mypath<-"C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/Latex/
    phd-thesis-template-phd-latex-template-latest-stable/Chapter5/Figs/
    PDF/Matern2.pdf"
pdf(file=mypath,onefile=FALSE,paper="special")
par(cex=2.2,mar=c(4, 4, 1.0, 1.0) + 0.1,mgp=c(2.3,1,0))#mar: c(bottom,
    left,top,right)
plot(dist.x,matern(lambda,kappa=kappa[2],dist.x),lwd=3,type="l",ylab="
    Matrn covariance function",xlab="distance [km]",lty=1,xaxt = 'n',
    yaxt = 'n')
lines(dist.x,matern(lambda,kappa=kappa[1],dist.x),lwd=3,type="l",ylab=NA
    ,xlab=NA,lty=2)
lines(dist.x,matern(lambda,kappa=kappa[3],dist.x),lwd=3,type="l",ylab=NA
    ,xlab=NA,lty=2)
abline( a = 0.1,b=0, v = NULL, reg = NULL,lty=3,lwd=2)#for vertical
    value v=range
abline( v =range, lty=3,lwd=2)#for vertical value v=range
axis(1, at=seq(0,2000/6371,500/6371), labels=c(0,500,1000,1500,2000))#1
    rad=6371km, 2000 km=2000/6371rad
axis(2, at=seq(0,0.8,0.2), labels=c(0.0,0.2,0.4,0.6,0.8))#1rad=6371km,
    2000 km=2000/6371rad
dev.off()
#computational time
#MCMC (N^3) in R2
MCMC<-(9697*12)^3
#INLA (N^(3/2)) in R2
INLA<-(9697*12)^(3/2)
#ration INLA/MCMC
MCMC/INLA

#plot comparing poisson with default priors and manual priors
coeff<-as.data.frame(resfinal$summary.fixed[3:5])#keep 0.025Q,0.5Q,0.975
    Q
#kappa /computed using Blangiardo & Cameletti (2015)
```

```
variance<-inla.emarginal(function(x) x, spde.resfinal$marginals.variance
    .nominal[[1]])#variance (mean)
variancehpd<-inla.hpdmarginal(0.95, spde.resfinal$marginals.variance.
    nominal[[1]])#variance (hpd 95%)
range<-inla.emarginal(function(x) x, spde.resfinal$marginals.range.
    nominal[[1]])#range (mean)
rangehpd<-inla.hpdmarginal(0.95, spde.resfinal$marginals.range.nominal
    [[1]])#range (hpd 95%)
r<-as.vector(c(rangehpd[1],range,rangehpd[2]))#keep 0.025Q,0.5Q,0.975Q
t<-as.vector(c(1/variancehpd[2],1/variance,1/variancehpd[1]))#keep 0.025
    Q,0.5Q,0.975Q
def<-rbind(coeff,r,t)
def$model<-1

#with manual prior (elicited)
coeff2<-as.data.frame(resprior$summary.fixed[3:5])#keep 0.025Q,0.5Q
    ,0.975Q
variance<-inla.emarginal(function(x) x, spde.resprior$marginals.variance
    .nominal[[1]])#variance (mean)
variancehpd<-inla.hpdmarginal(0.95, spde.resprior$marginals.variance.
    nominal[[1]])#variance (hpd 95%)
range<-inla.emarginal(function(x) x, spde.resprior$marginals.range.
    nominal[[1]])#range (mean)
rangehpd<-inla.hpdmarginal(0.95, spde.resprior$marginals.range.nominal
    [[1]])#range (hpd 95%)
r2<-as.vector(c(rangehpd[1],range,rangehpd[2]))#keep 0.025Q,0.5Q,0.975Q
t2<-as.vector(c(1/variancehpd[2],1/variance,1/variancehpd[1]))#keep
    0.025Q,0.5Q,0.975Q
def2<-rbind(coeff2,r2,t2)
def2$model<-2

alldef<-rbind(def,def2)
alldef$model<-as.factor(alldef$model)
alldef$name<-c("b0","lum","tt","pop","alt","pol","range","tau","b0","lum
    ","tt","pop","alt","pol","range","tau")
alldef <- alldef[order(alldef$name, alldef$model),]
alldef$number<-as.numeric(c("3","4","1","2","5","6","7","8","9","10","13
    ","14","15","16","11","12"))
alldef<-alldef[order(alldef$number),]
```

```
plot.new()
dev.off()
mypath<-"~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/
    poissspriorcompa.eps"
postscript(file=mypath,horiz=FALSE,onefile=FALSE,width=0,height=0,paper=
    "default")
par(cex=2,mar=c(2.2, 4.3, 0.3, 0.3) + 0.1)#mar: c(bottom,left,top,right)
dotchart(alldef[,2], labels=c(expression(beta[0]),"",expression(beta[alt
    ]),"",expression(beta[lum]),"",expression(beta[pol]),"",expression(
    beta[pop]),"",expression(beta[tt]),"",expression(italic(range)),"",
    expression(tau),""),
        pch=c(19,17),#pch=c(21,17),#bg=c("black","white"),
        xlim=c(min(alldef[,1]), max(alldef[,3])))
segments(alldef[,2]-(alldef[,2]-alldef[,1]),1:16,alldef[,2]+(alldef[,3]-
    alldef[,2]),1:16,lwd=2)
dev.off()
#End#####
```

Appendix D

Hotspots and Diffusion Processes of Lethal Terrorism

D.1 Definition of Hotspot, Escalation, and Diffusion Processes

This is the code used to define hotspots, escalation, and diffusion of the lethality of terrorism and creates graphics for visualisation.

```
#SPATIO-TEMPORAL MODELLING / Bernoulli model of the lethality of
    terrorism worldwide 2002-2013
#The code used to define hotspots, escalation, and diffusion and to
    generate graphics for visualisation.
#Before running the code, the selected model should be used.
setwd("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal")
rm(list=ls())
require(foreign)
require(sp)
library(INLA)
library(spatstat)
library(fields)
library(maptools)
library(fields)
library(lattice)
require(rgdal)#required for buffer
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/SPDE.RData")
```

```
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal/
    STbintotal.RData")#update file location according to selected model
    and the other ones
#load("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/PhD_R/
    NTNU/STbinfinal.RData")#data 2002-2013
rm(res,res1,res2,res3,res4,res5,resprior)#remove unselected models
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal/Llinpal
    .RData")#load custom palette
country<- readShapeSpatial("C:/Users/apython/Documents/Andre/Education/
    StAndrews/PhD/PhD_ArcGIS/World␣map/country.shp")#load country borders
     with same characteristics than study area (for plot)
proj4string(country)<-CRS("+proj=longlat␣+datum=WGS84␣+no_defs+towgs84
    =0,0,0+ellps=WGS84")
#projection of the mesh and restriction of analysis in the studyarea (
    ocean excluded)
proj = inla.mesh.projector(mesh, projection = "longlat", dims=c
    (1440,720))#resolution identical to PRIO-Grid
win<-as.owin(studyarea)
library(mgcv)
e<-expand.grid(proj$x,proj$y)
ins<-inside.owin(e[,1],e[,2],win)
ins<-matrix(ins,nrow=length(proj$y))
#get linear predictor prediction Quant 0.025
#covariate value at mesh location
require(raster)
#create lon lat in matrix form
loc2d<-inla.mesh.map(mesh$loc, projection = "longlat",inverse = FALSE)
loc2d<-as.data.frame(loc2d)
loc2d$position<-rownames(loc2d)
xy = SpatialPoints(cbind(loc2d[,1],loc2d[,2]))
xy = SpatialPointsDataFrame(cbind(loc2d[,1],loc2d[,2]), loc2d)
proj4string(xy)<-CRS("+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84")
#xy<-xy[studyarea,]

lum02<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum02.tif")
WGS84 = "+proj=longlat␣+datum=WGS84␣+ellps=WGS84␣+towgs84=0,0,0"
projection(lum02)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
```

```
lum03<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum03.tif")
projection(lum03)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum04<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum04.tif")
projection(lum04)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum05<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum05.tif")
projection(lum05)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum06<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum06.tif")
projection(lum06)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum07<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum07.tif")
projection(lum07)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum08<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum08.tif")
projection(lum08)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum09<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum09.tif")
projection(lum09)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum10<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum10.tif")
projection(lum10)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum11<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum11.tif")
projection(lum11)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum12<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum12.tif")
projection(lum12)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
```

```
lum13<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_ArcGIS/luminosity/F182013.v4c_web.stable_lights.avg_vis.tif")
projection(lum13)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"

#extract luminosity values at locations in space and time defined by GTD
     space-time locations (time-consuming!)
lum02<-as.data.frame(lum02<-extract(lum02,xy,method='simple'))
lum03<-as.data.frame(lum03<-extract(lum03,xy,method='simple'))
lum04<-as.data.frame(lum04<-extract(lum04,xy,method='simple'))
lum05<-as.data.frame(lum05<-extract(lum05,xy,method='simple'))
lum06<-as.data.frame(lum06<-extract(lum06,xy,method='simple'))
lum07<-as.data.frame(lum07<-extract(lum07,xy,method='simple'))
lum08<-as.data.frame(lum08<-extract(lum08,xy,method='simple'))
lum09<-as.data.frame(lum09<-extract(lum09,xy,method='simple'))
lum10<-as.data.frame(lum10<-extract(lum10,xy,method='simple'))
lum11<-as.data.frame(lum11<-extract(lum11,xy,method='simple'))
lum12<-as.data.frame(lum12<-extract(lum12,xy,method='simple'))
lum13<-as.data.frame(lum13<-extract(lum13,xy,method='simple'))
# # intercalibration based on Elvidge2013
# # Process: 1) calculate: Y = C0 + C1X + C2X^2 // 2) Values > 63 are
    truncated at 63 // 3) values = 0 stay zero.
# Sat   Year      C0          C1       C2
# F15   2002     0.0491   0.9568   0.0010
# F15     2003   0.2217   1.5122   -0.0080
# F16   2004     0.2853   1.1955   -0.0034
# F16     2005   -0.0001  1.4159   -0.0063
# F16     2006   0.1065   1.1371   -0.0016
# F16     2007   0.6394   0.9114   0.0014
# F16   2008     0.5564   0.9931   0.0000
# F16     2009   0.9492   1.0683   -0.0016
# F18   2010     2.3430   0.5102   0.0065
# F18     2011   1.8956   0.7345   0.0030
# F18     2012   1.8750   0.6203   0.0052
f02 <- function(x){ifelse(x>0, 0.0491+0.9568*x+0.0010*x^2, 0)}#if the
    values are above 0, execute intercalibration
f03 <- function(x){ifelse(x>0, 0.2217+1.5122*x+0.0010*x^2, 0)}
f04 <- function(x){ifelse(x>0, 0.2853+1.1955*x+0.0010*x^2, 0)}
f05 <- function(x){ifelse(x>0, -0.0001+1.4159*x+0.0010*x^2, 0)}
f06 <- function(x){ifelse(x>0, 0.1065+1.1371*x+0.0010*x^2, 0)}
```

```
f07 <- function(x){ifelse(x>0, 0.6394+0.9114*x+0.0010*x^2, 0)}
f08 <- function(x){ifelse(x>0, 0.5564+0.9931*x+0.0010*x^2, 0)}
f09 <- function(x){ifelse(x>0, 0.9492+1.0683*x+0.0010*x^2, 0)}
f10 <- function(x){ifelse(x>0, 2.3430+0.5102*x+0.0010*x^2, 0)}
f11 <- function(x){ifelse(x>0, 1.8956+0.7345*x+0.0010*x^2, 0)}
f12 <- function(x){ifelse(x>0, 1.8750+0.6203*x+0.0010*x^2, 0)}
f13 <- function(x){ifelse(x>0, 1.8750+0.6203*x+0.0010*x^2, 0)}

lum02<-as.data.frame(sapply(lum02,f02))
lum02[lum02 > 63] <- 63 #truncate if values are higher than 63
lum03<-as.data.frame(sapply(lum03,f03))
lum03[lum03 > 63] <- 63 #truncate if values are higher than 63
lum04<-as.data.frame(sapply(lum04,f04))
lum04[lum04 > 63] <- 63 #truncate if values are higher than 63
lum05<-as.data.frame(sapply(lum05,f05))
lum05[lum05 > 63] <- 63 #truncate if values are higher than 63
lum06<-as.data.frame(sapply(lum06,f06))
lum06[lum06 > 63] <- 63 #truncate if values are higher than 63
lum07<-as.data.frame(sapply(lum07,f07))
lum07[lum07 > 63] <- 63 #truncate if values are higher than 63
lum08<-as.data.frame(sapply(lum08,f08))
lum08[lum08 > 63] <- 63 #truncate if values are higher than 63
lum09<-as.data.frame(sapply(lum09,f09))
lum09[lum09 > 63] <- 63 #truncate if values are higher than 63
lum10<-as.data.frame(sapply(lum10,f10))
lum10[lum10 > 63] <- 63 #truncate if values are higher than 63
lum11<-as.data.frame(sapply(lum11,f11))
lum11[lum11 > 63] <- 63 #truncate if values are higher than 63
lum12<-as.data.frame(sapply(lum12,f12))
lum12[lum12 > 63] <- 63 #truncate if values are higher than 63
lum13<-as.data.frame(sapply(lum13,f13))
lum13[lum13 > 63] <- 63 #truncate if values are higher than 63

lumls<-list(lum02,lum03,lum04,lum05,lum06,lum07,lum08,lum09,lum10,lum11,
    lum12,lum13)
# #normalising data
for (j in 1:k){
lumls[[j]][,1]<-scale(lumls[[j]][,1])
names(lumls[[j]])<-"lum"
}
```

```
xydf<-list()
for (j in 1:k){
xydf[[j]]<-as.data.frame(xy)
xydf[[j]]$time<-2001+j
}
#travel time covariate (spatial not temporal)
ttime <- raster("C:/Users/apython/Documents/Andre/Education/StAndrews/
    PhD/PhD_ArcGIS/traveltime/access_50k/acc_50k")
#ttime<-projectRaster(ttime,crs=WGS84)
#projection(ttime)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
tt<-extract(ttime,xy,method='simple')#extract covariate value at point
    locations
tt<-as.vector(tt)
#replace NA values by 0
tt[is.na(tt)] <- 0
tt<-scale(tt)#normalising data (mean=0, sd=1) for better INLA process
GREG <- readShapeSpatial("C:/Users/apython/Documents/Andre/Education/
    StAndrews/PhD/PhD_R/GREG/GREG.shp")
proj4string(GREG)<-CRS("+proj=longlat +datum=WGS84 +no_defs+towgs84
    =0,0,0+ellps=WGS84")
#intersect values of GREG with points of xy
o <- over(xy,GREG)
#if no ethnic group (0) put NA in order that the absence of ethnic group
     will not be counted
#keep only id and number of ethnic groups
xyGREG<-o[,c(5,6,7)]
#if no ethnic group (0) put NA in order that the absence of ethnic group
     will not be counted
xyGREG[xyGREG == 0] <- NA
xyGREG$id <- c(1:nrow(xyGREG))
xyGREG<-ddply(xyGREG, .(id), mutate, count = length(unique(na.omit(c(
    G1ID,G2ID,G3ID)))))
#scale
xyGREG$count<-scale(xyGREG$count)
#keep a vector
greg<-as.vector(xyGREG$count)
#Altitude covariate (spatial not temporal)
#source: DEM map (ETOPO1): https://www.ngdc.noaa.gov/mgg/global/relief/
    ETOPO1/image/
```

```
altitude <- raster("C:/Users/apython/Documents/Andre/Education/StAndrews
    /PhD/PhD_ArcGIS/DEM/ETOPO1/color_etopo1_ice_full.tif")
projection(altitude)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84
    =0,0,0+ellps=WGS84"
#extract coordinates from population at the location of the points
    coordinates
alt<-extract(altitude,xy,method='simple')#extract covariate value at
    point locations
alt<-as.vector(alt)
#replace NA values by 0
alt[is.na(alt)] <- 0
alt<-scale(alt)#normalising data (mean=0, sd=1) for better INLA process
for (j in 1:k){
  xydf[[j]]$lum<-lumls[[j]]$lum
  xydf[[j]]$tt<- tt
  xydf[[j]]$greg<- greg
  xydf[[j]]$alt<- alt
  xydf[[j]]$time<- xydf[[j]]$time-2001
}
rm(GTD,GTDGREG,GTDpt,loc2d,lum02,lum03,lum04,lum05,lum06,lum07,lum08,
    lum09,lum10,lum11,lum12,lum13)
rm(alt,data,distb,lum,o,pop,slo,test,tt,xy,xyDpt,xyGREG,altitude,GREG,i,
    id,interior,j,k,kappa0,lumls)
rm(greg,nv,pol,range0,res4bis,sigma0,sigma0,stk,studyarea,tau0,time,
    ttime,xypt,f02,f03,f04,f05,f06,f07)
rm(f08,f09,f10,f11,f12,f13)
#covariates at the mesh vertices in 12 years
xyall<-do.call("rbind", xydf)
xyall<-xyall[c("V1","V2","time","lum","tt","greg","alt")]
attr(xyall$lum, "dimnames") <- NULL
attr(xyall$alt, "dimnames") <- NULL
attr(xyall$tt, "dimnames") <- NULL
xyall$lum<-as.numeric(xyall$lum)
xyall$tt<-as.numeric(xyall$tt)
xyall$alt<-as.numeric(xyall$alt)
#load old data
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal/
    STbintotal.RData")#update file location according to selected model
    and the other ones
olddf<-as.data.frame(cbind(y,lum,tt,greg,alt,time,loc))
```

```
names(olddf)<-c("y","lum","tt","greg","alt","time","V1","V2","V3")
olddf<-olddf[complete.cases(olddf$y),]
#to save time, start from here...
#load("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/PhD_R/
    SPDE/STbinomialtotal/minfile.RData")
k<-12
spde <- inla.spde2.matern(mesh, alpha=2)
iset<-inla.spde.make.index('i',n.spde=spde$n.spde,n.group=k)
iset2<-inla.spde.make.index('i',n.spde=nrow(xyall)/12,n.group=k)
A<-inla.spde.make.A(mesh,loc=cbind(olddf$V1,olddf$V2,olddf$V3), group=
    olddf$time)
stk <- inla.stack(data=list(y=olddf$y),A=list(A,1,1,1,1,1), tag='dat',
                  effects=list(i=iset,lum=olddf$lum,tt=olddf$tt,greg=
                      olddf$greg,alt=olddf$alt,b0=rep(1,length(olddf$y)))
                      )
stk.mesh <- inla.stack(data=list(y=NA), tag='mesh',A=list(1,1,1,1,1,1),
                    #  stack2 <- inla.stack(stk, stk.mesh),
                       effects=list(i=iset2,lum=xyall$lum,tt=xyall$tt,
                           greg=xyall$greg,alt=xyall$alt,b0=rep(1,length(
                           iset2$i))))
idx.mesh <- inla.stack.index(stk.mesh, tag='mesh')$data
                  #     idx.dat <- inla.stack.index(stk, tag='dat')$data
                  #     and them use 'idx.mesh' to collect as
                       #random field is ok but the fitted values do not
                           work! So strange!!!
lowprob<-as.data.frame(binomial(link='logit')$linkinv(resfinal$summary.
    linear.predictor[idx.mesh, "0.025quant"]))
highprob<-as.data.frame(binomial(link='logit')$linkinv(resfinal$summary.
    linear.predictor[idx.mesh,'0.975quant']))

lowprob$group<-xyall$time
names(lowprob)<-c("prob","group")
highprob$group<-xyall$time
names(highprob)<-c("prob","group")
bdq25prob<-list()
bdq975prob<-list()
for (j in 1:k){
  bdq25prob[[j]]<-lowprob$prob[lowprob$group==j]
  bdq975prob[[j]]<-highprob$prob[highprob$group==j]
```

```
  bdq25prob[[j]]<-inla.mesh.project(proj,bdq25prob[[j]])#adapt to the
      selected model
  bdq25prob[[j]][!ins]<-NA
  bdq975prob[[j]]<-inla.mesh.project(proj,bdq975prob[[j]])#adapt to the
      selected model
  bdq975prob[[j]][!ins]<-NA
}
#plot lethal hotspots year after year (with country borders)
#define values for hotspot
H<-0.5
maxH<-max(unlist(bdq25prob),na.rm=TRUE)
require(fields)
library("colorspace")
breaks<-seq(H,maxH,by=0.01)
n<-length(breaks)-1
# rm(i)#check
# plot.new()
# image.plot(proj$x,proj$y,bdq25prob[[2]],
#             xlab="",ylab="",main=""
#          )
# plot(country,yaxt = "n",add=TRUE)

for(i in 1:12){
  mypath<-paste("C:/Users/apython/Documents/Andre/Education/StAndrews/
      PhD/revision/revisedPhDthesis/Chapter6/Figs/Raster/hotspot",i,".png
      ",sep="")
   png(file=mypath)
  par(cex=1.4, cex.lab=0.7)#mar: c(bottom,left,top,right)
  image.plot(proj$x,proj$y,bdq25prob[[i]],
             col=rev(heat_hcl(n, c = c(120, 0.5), l = c(30, 90), power =
                 c(1/10, 3))),
             xlab="",ylab="",main="",
             xlim = c(-20, 80), ylim = c(-40, 40),zlim=c(H,maxH))#zlim
                 creates identical scale
  title(main=2001+i, line = 0.5,cex.main=2.2)
  plot(country,yaxt = "n",add=TRUE)
  mtext("Latitude", side = 2, line = 2.8,adj =0,cex = 1.4,at=-6.5)
  mtext("Longitude", side = 1, line = 3,adj =0,cex = 1.4,at=20)
  dev.off()
}
```

```
#Compute diffusion for each year (2003-2012): 2002 excluded since it
    requires j+1 year
#However, hotspots of lethality are identified from 2002 (j) to 2012 (k
    -1) and keep only locations with hotspot (with value to be put to 1)
    and put NA else
hot<-list()
for(j in 1:k){
  hot[[j]]<-ifelse(bdq25prob[[j]]<H,NA,1) #put NA everywhere except
     where there are hotspots (useful for next steps)
}
require(raster)
#create vectors with projection values (x,y) and hotspot values (z)
hotdf<-list()
hotr<-list()
for(j in 1:k){
hotdf[[j]]=list()
hotdf[[j]]$x=proj$x#longitude
hotdf[[j]]$y=proj$y#latitude
hotdf[[j]]$z=hot[[j]]
#convert vectors to raster
hotr[[j]]=raster(hotdf[[j]])
}

#create dataframe from raster
hotspotdf=list()
for(j in 1:k){
hotspotdf[[j]]=as.data.frame(hotr[[j]],xy=TRUE)#add xy option to extract
    lon and lat
names(hotspotdf[[j]])<-c("lon","lat","hot")
}
#delete hotpsot values if NA
for(j in 1:k){
hotspotdf[[j]]=hotspotdf[[j]][complete.cases(hotspotdf[[j]]),]
}

#idenfiy locations of hotspots cells (cell number of hotspots)
hotloc<-list()
for(j in 1:k){
hotloc[[j]]<-which(!is.na(values(hotr[[j]])))
}
```

```
#NEIGHBOURHOOD: OPTIONAL CHOICE OF THE WAY NEIGHBOURHOOD IS DEFINED
#identify the neighbours of hotspots for each j time period
#we chose buffer of approx. 100 km at the equator (could be adjusted)
#create polygon of hotspots from raster cells
hotrpoly<-list()
for(j in 1:k){
  hotrpoly[[j]]<-rasterToPolygons(hotr[[j]], fun=NULL, n=8, na.rm=TRUE,
      digits=5, dissolve=TRUE)
  hotrpoly[[j]]<-unionSpatialPolygons(hotrpoly[[j]], rep(1, length(
      hotrpoly[[j]])))#put adjacent polygons together
  hotrpoly[[j]]<-disaggregate(hotrpoly[[j]])#disaggregate into multiple
      polygons
}
#some visual checks
# plot(hotrpoly[[11]],col="red",xlim=c(25,45),ylim=c(30,40))
# plot(country,add=TRUE)

#Create buffer
require(rgeos)
hotbuff<-list()
hotadj<-list()
diffnet<-list()#the final result clipped within studyarea (if option
    below not done)
for(j in 1:k){
  hotbuff[[j]]<-gBuffer(hotrpoly[[j]], byid=FALSE, id=NULL, width=0.9,
      quadsegs=5, capStyle="ROUND",joinStyle="ROUND")
  hotbuff[[j]]<-gDifference(hotbuff[[j]], hotrpoly[[j]])
  proj4string(hotbuff[[j]])<-CRS(proj4string(studyarea))#gives reference
  hotadj[[j]]<-gIntersection(hotbuff[[j]],studyarea)
  hotadj[[j]]<-disaggregate(hotadj[[j]])#from one to multiple polygons
  diffnet[[j]]<-hotadj[[j]]
}
#OPTIONAL: delete too small polygon (here we keep polgyons only if the
    area=0.5 (same as PRIO-Grid))
# tau<-0.25#(0.5 corresponds to  PRIO-grid cell area)
# areas<-list()
# maxareas<-list()
# indexarea<-list()
# for(j in 1:k){
```

```
# #calculate areas of all polygons included in hotspots neighbourhood
    for each year
# areas[[j]]<- sapply(slot(hotadj[[j]], "polygons"), function(x) sapply(
    slot(x,"Polygons"), slot, "area"))
# areas[[j]]<- gArea(hotadj[[j]], byid=TRUE)
# maxareas[[j]]<-sapply(areas[[j]], max)#in case of possible divided
    polygons keep the one with highest area (is required for following
    steps)
# indexarea[[j]]<- which(maxareas[[j]]>tau)
# }
# #
# # #keep polygon with area>tau
# diffnet<-list()
# for(j in 1:k){#length rpos idem rneg...
#   diffnet[[j]]<-hotadj[[j]][indexarea[[j]],]#keep polygon with area>
    tau
# }

#delete the first year because we look at diffusion from 2003 to 2013.
    Indeed, a diffusion from 2002 to 2003 is labelled '2003'.
diffnet[[1]] <- NULL
# #some checks
# plot.new()
#  plot(hotadj[[2]],col="yellow",xlim=c(0,20),ylim=c(30,45))#year 2003
#  plot(diffnet[[1]],col="red",add=TRUE)#year 2003
#  plot(studyarea,add=TRUE)
#END NEIGHBOURHOOD CHOICE
#count nb. of hotspots (from the model, including true and not true
    hostspots)
nbhot<-list()
for(j in 1:11){
  nbhot[[j]]<-length(disaggregate(diffnet[[j]]))}
Reduce("+",nbhot)#nb. of hotspots 169

#create dataframe with projection values (x,y) and lethal probability
    values (z) for lower and higher bounds of CI.
problowdf<-list()
problowr<-list()
for(j in 1:k){
  problowdf[[j]]=list()
```

```
  problowdf[[j]]$x=proj$x
  problowdf[[j]]$y=proj$y
  problowdf[[j]]$z=bdq25prob[[j]]#using lower bound CI of probability of
      lethal attack
  #convert dataframe to raster
  problowr[[j]]<-raster(problowdf[[j]])
  proj4string(problowr[[j]])<-CRS(proj4string(studyarea))#required
}

probhighdf<-list()
probhighr<-list()
for(j in 1:k){
  probhighdf[[j]]=list()
  probhighdf[[j]]$x=proj$x
  probhighdf[[j]]$y=proj$y
  probhighdf[[j]]$z=bdq975prob[[j]]#using lower bound CI of probability
     of lethal attack
  #convert dataframe to raster
  probhighr[[j]]<-raster(probhighdf[[j]])
  proj4string(probhighr[[j]])<-CRS(proj4string(studyarea))#required
}

#extract delta probability of lethal attack at time j and j+1 for
    adacent cells to hotspots
#define kappa (threshold) for enlarging CI (min 0 to max 1) to compute
    signifcant differences between CI in t and t-1
#kappa is between ]0;1]. Values close to 0 indicate high overlap between
     CI and values close to 1 indicate low overlap (more restrictive)
kappa<-0.75 #(0.75: indicate that we tolerate an overlap of CI with 75%
    of the related value)
#for positive diffusion: low p(t+1)-kappa*high p(t)>0
#for negative diffusion: high p(t+1)-kappa*low p(t)<0
#OPTION: WE EXTRACT min and max of prob. values within adjacent polygons
     of hotspots.If there is at least one diffusion or negative diffusion
     within the polygon, we identify it. This is done by using min and
    max of the CI lower and higher bounds.
diffpos<-list()
diffneg<-list()
for(j in 1:(k-1)){  #diffnet starts in 2003 (j: 1 to 11)
```

```
diffpos[[j]]<-extract(problowr[[j+1]],diffnet[[j]],fun=max,na.rm=TRUE)-
    extract(kappa*probhighr[[j]],diffnet[[j]],fun=min,na.rm=TRUE)
diffneg[[j]]<-extract(kappa*probhighr[[j+1]],diffnet[[j]],fun=min,na.rm=
    TRUE)-extract(problowr[[j]],diffnet[[j]],fun=max,na.rm=TRUE)
}
#put the result as dataframe
diffposdf<-list()
diffnegdf<-list()
for(j in 1:(k-1)){  #starts in 2003 (j: 1 to 11)
  diffposdf[[j]]<-as.data.frame(diffpos[[j]])
  diffnegdf[[j]]<-as.data.frame(diffneg[[j]])
}
#create ID column for each df (in order to merge dataframe)
for(j in 1:(k-1)){
  diffposdf[[j]]$ID<-1:nrow(diffposdf[[j]])
  diffnegdf[[j]]$ID<-1:nrow(diffnegdf[[j]])
  names(diffposdf[[j]]) <- c("diffpos","ID")
  names(diffnegdf[[j]]) <- c("diffneg","ID")
}
#create  spatial poly df from spatial polygons.
diffnetdf<-list()
for(j in 1:(k-1)){
  diffnetdf[[j]]<-SpatialPolygonsDataFrame(diffnet[[j]], data.frame(N =
      1:length(diffnet[[j]]), row.names = 1:length(diffnet[[j]])))
}
#put diffusion values (positive and negative) into sp. poly dfs
for(j in 1:(k-1)){
diffnetdf[[j]]@data <- data.frame(diffnetdf[[j]]@data, diffposdf[[j]][
    match(diffnetdf[[j]]@data[,'N'], diffposdf[[j]][,'ID']),])#N is name
    of variable of spdf and ID name of variable of df which are the
    common variable on which the merge is executed.
}
for(j in 1:(k-1)){
diffnetdf[[j]]@data <- data.frame(diffnetdf[[j]]@data, diffnegdf[[j]][
    match(diffnetdf[[j]]@data[,'N'], diffnegdf[[j]][,'ID']),])#N is name
    of variable of spdf and ID name of variable of df which are the
    common variable on which the merge is executed.
}
#plot annual diffusion
#color definition
```

```
coldiffpos<-list()
for(j in 1:(k-1)){coldiffpos[[j]]<-diffnetdf[[j]]@data$diffpos
}
mindiffpos<-0
maxdiffpos<-max(sapply(coldiffpos, max,na.rm=TRUE))
breaks<-seq(mindiffpos,maxdiffpos,by=0.002)
n<-length(breaks)-1

coldiffneg<-list()
for(j in 1:(k-1)){coldiffneg[[j]]<-diffnetdf[[j]]@data$diffneg
}
maxdiffneg<-0
mindiffneg<-min(sapply(coldiffneg, min,na.rm=TRUE))
breaks<-seq(mindiffneg,maxdiffneg,by=0.002)
n<-length(breaks)-1

#plot annual diffusion
dev.off()
plot.new()
#restricted lat and lon: xlim = c(-20, 60), ylim = c(-40, 40),#Africa
    and Middle East for better visualisation
diffnetcrop<-list()
for(j in 1:(k-1)){diffnetcrop[[j]] <- crop(diffnetdf[[j]], extent(-20,
    80, -40, 40))}
plot.new()
for(j in 1:(k-1)){
  mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD
      /revision/revisedPhDthesis/Chapter6/Figs/Raster/diffusion",j,".png"
      ,sep="")
  png(file=mypath)
  par(cex=1.4, cex.lab=0.7)
  plot(diffnetcrop[[j]],xlim=c(-20,80),ylim=c(-40,40),
       col=ifelse(diffnetcrop[[j]]@data$diffpos>0 & diffnetcrop[[j]]
           @data$diffneg>=0 ,"red",ifelse(diffnetcrop[[j]]@data$diffneg<0
            & diffnetcrop[[j]]@data$diffpos<=0,"blue","grey25")),main=""
       ,xlab="",ylab="",axes=F)
  box()
  axis(1, col.axis="black",at=seq(-20,80,by=20),labels=seq(-20,80,by=20)
      , las=2)
```

```
  axis(2, col.axis="black", at=seq(-40,40,by=20),labels=seq(-40,40,by
      =20),las=0)#
  plot(country,add=TRUE)
  title(main=2002+j, line = 0.5,cex.main=1.5)
  mtext("Latitude", side = 2, line = 2,adj =0,cex = 1.4,at=-7.5)
  mtext("Longitude", side = 1, line = 2,adj =0,cex = 1.4,at=20)
    legend("bottomleft",title="",inset=.005,cex=1,bty="n",
         c("diffusion","dissipation","not␣significant"),fill=c("red","
            blue","grey25"))
  dev.off();dev.off()
}

#Escalation (change in terrorism intensity within hotspot (at the cell
    level))
#annual escalation
#extract delta probability of lethal attack at time j and j+1 within
    hotspots
#define kappa (threshold) for enlarging CI (min 0 to max 1) to compute
    signifcant differences between CI in t and t-1
kappa<-0.75 #(0.75: indicate that we tolerate an overlap of CI with 75%
    of the related value)
#for positive escalation: low p(t+1)-kappa*high p(t)>0
#for negative escalation: high p(t+1)-kappa*low p(t)<0
require(raster)
escpos<-list()
escneg<-list()
for(j in 1:(k-1)){
  escpos[[j]]<-extract(problowr[[j+1]],hotloc[[j]])-extract(kappa*
      probhighr[[j]],hotloc[[j]])
  escneg[[j]]<-extract(kappa*probhighr[[j+1]],hotloc[[j]])-extract(
      problowr[[j]],hotloc[[j]])
}

#join escalation values with cell number of hotspotscells
#create dataframe with projection values (x,y) and lethal probability
    values (z)
escdf<-list()
for(j in 1:(k-1)){
  escdf[[j]]<-as.data.frame(cbind(escpos[[j]],escneg[[j]],hotloc[[j]]))
}
```

```
#extract raster cells with values of prob of lethal at the location of
    hotspot cells
#Done for each year from 2002 to 2013
escxyz<-list()
for(j in 1:k){
  escxyz[[j]]<-xyFromCell(probhighr[[j]],hotloc[[j]])
}
#Add xy location and delta values from j to k-1 to the dataframe for eac
    j year. Note that delta have 11 values (k-1), no values calculated
    for time k (2013)
#Done for each year from 2003 to 2012
for(j in 1:(k-1)){
  escdf[[j]]<-cbind(escdf[[j]],escxyz[[j]])
}
#Rename variables in dataframe: positive, negative escalation, cell
    number, longitude, and latitude,
for(j in 1:(k-1)){
  colnames(escdf[[j]]) <- c("posesc","negesc","cellnum","lon","lat")
}

#delete escdf observations if NA is present (if no esc values are
    provided. For example, when adjacent cell lie in ocean)
for(j in 1:(k-1)){
  escdf[[j]]=escdf[[j]][complete.cases(escdf[[j]]),]
}

#Escalation
#create spatial points dataframe from escdf dataframe
cooresc<-list()
escpt<-list()
for(i in 1:(k-1))
  {
#escalation data
cooresc[[i]]<-cbind(escdf[[i]]$lon,escdf[[i]]$lat)
escpt[[i]]<-SpatialPointsDataFrame(cooresc[[i]], escdf[[i]],proj4string=
    CRS("+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+ellps=WGS84"))
escpt[[i]]<-escpt[[i]][studyarea,]#problem!!!!CRS
escpt[[i]]<-remove.duplicates(escpt[[i]])
    }
```

```
#plot annual escalation
#color definition
colescpos<-list()
for(j in 1:(k-1)){colescpos[[j]]<-escpt[[j]]$posesc}
minescpos<-0
maxescpos<-max(sapply(colescpos, max,na.rm=TRUE))
breaks<-seq(minescpos,maxescpos,by=0.002)
n<-length(breaks)-1

colescneg<-list()
for(j in 1:(k-1)){colescneg[[j]]<-escpt[[j]]$negesc}
maxescneg<-0
minescneg<-min(sapply(colescneg, min,na.rm=TRUE))
breaks<-seq(minescneg,maxescneg,by=0.002)
n<-length(breaks)-1

#plot annual escalation
plot.new()
dev.off()
for(j in 1:(k-1))
{
  mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD
      /revision/revisedPhDthesis/Chapter6/Figs/Raster/escalation",j,".png
      ",sep="")
  png(file=mypath)
  par(cex=1.4, cex.lab=0.7)
  plot(escpt[[j]]$lon,escpt[[j]]$lat,pch=15,cex=0.4,col="NA",main=""
       ,xlim = c(-20, 80), ylim = c(-40, 40),
       xlab="",ylab="")
  points(escpt[[j]]$lon,escpt[[j]]$lat,pch=15,cex=0.2,col=ifelse(escpt[[
      j]]$posesc>0,"red",ifelse(escpt[[j]]$negesc<0 , "blue", "NA")))
  plot(country,add=TRUE)
  title(main=2002+j, line = 0.5,cex.main=1.5)
  mtext("Latitude", side = 2, line = 1.9,adj =0,cex = 1.4,at=-7.5)
  mtext("Longitude", side = 1, line = 1.9,adj =0,cex = 1.4,at=20)
    legend("bottomleft",title="",inset=.005,cex=1,bty="n",
         c("escalation","de-escalation"),fill=c("red","blue"))
    dev.off()
}
```

```
# Not reported: global escalation from 2002 to 2013: E= (p_{2013}-p_{
    j2002}), with j time index (for positive and negative escalation)
# t<-1;T<-12
# #calculate cumulated escalation from 2002-2003 and put as a dataframe
    with coordinates of hotspot in 2002
# globposesc<-extract(problowr[[12]],hotloc[[1]])-extract(kappa*
    probhighr[[1]],hotloc[[1]])
# globnegesc<-extract(kappa*probhighr[[12]],hotloc[[1]])-extract(
    problowr[[1]],hotloc[[1]])
# globescdf<-as.data.frame(cbind(globposesc,globnegesc,hotloc[[1]],
    escxyz[[1]]))
# colnames(globescdf) <- c("posesc","negesc","cellnum","lon","lat")
# #delete deltadf observations if NA is present (if no delta values are
    provided. For example, when adjacent cell lie in ocean)
# globescdf=globescdf[complete.cases(globescdf),]
#
# #create spatial points dataframe from globalescdf and hotspot
    dataframe
# #hotspot data
# coorescal<-cbind(globescdf$lon,globescdf$lat)
# globescpt<-SpatialPointsDataFrame(coorescal, globescdf,proj4string=CRS
    ("+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+ellps=WGS84"))
# globescpt<-globescpt[studyarea,]#keep observations within the study
    area exclusively
# globescpt<-remove.duplicates(globescpt)#remove possible spatial
    duplicates (if points on same location)
#
# #plot
# path="~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal/
    escalation/escalation02_13.png"
# png(file=path,width=1440,height=720)
# par(cex=2,mar=c(5.5, 4.5, 2.5, 0.5) + 0.1)#mar: c(bottom,left,top,
    right)
# plot(globescpt$lon,globescpt$lat,pch=15,cex=0.4,col="NA",main="Global
    escalation of lethal terrorism 2002-2013",sub=2002+j
#       ,xlim=c(-180, 180), ylim=c(-65, 65),xlab="longitude",ylab="
    latitude")
#
# points(globescpt$lon,globescpt$lat,pch=15,cex=0.2,col=ifelse(globescpt
    $posesc>0,"red",ifelse(globescpt$negesc<0 , "blue", "NA")))
```

```
# #points(hotpt[[1]]$lon,hotpt[[1]]$lat,pch=15,cex=0.5,col="grey25")
# plot(country,add=TRUE)
# dev.off()

save.image("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal/
    STbin_diffusion.RData")
#load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal/STbin_
    diffusion.RData")
#END
```

This is the code used to define hotspots, escalation, and diffusion of the number of lethal terrorist attacks and creates graphics for visualisation.

```
#SPATIO-TEMPORAL MODELLING / Poisson model of the number of lethal
    events worldwide 2002-2013
#The code used to define hotspots, escalation, and diffusion and to
    generate graphics for visualisation.
#Before running the code, the selected model should be used.
setwd("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson")
rm(list=ls())
require(foreign)
require(sp)
library(INLA)
library(spatstat)
library(fields)
library(maptools)
library(fields)
library(lattice)
require(rgdal)#required for buffer
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/SPDE_nbevents
    _final.RData")
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/STpoitotal.
    RData")#update file location according to model selection
rm(res3,res4,resprior)#remove unselected models
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal/Llinpal
    .RData")#load custom palette
country<- readShapeSpatial("C:/Users/apython/Documents/Andre/Education/
    StAndrews/PhD/PhD_ArcGIS/World␣map/country.shp")#load country borders
     with same characteristics than study area (for plot)
proj4string(country)<-CRS("+proj=longlat␣+datum=WGS84␣+no_defs+towgs84
    =0,0,0+ellps=WGS84")
```

```
#projection of the mesh and restriction of analysis in the studyarea (
    ocean excluded)
proj = inla.mesh.projector(mesh, projection = "longlat", dims=c
    (1440,720))#resolution identical to PRIO-Grid
#proj = inla.mesh.projector(mesh, projection = "longlat", dims=c(144,72)
    )#resolution identical to PRIO-Grid
win<-as.owin(studyarea)
library(mgcv)
e<-expand.grid(proj$x,proj$y)
ins<-inside.owin(e[,1],e[,2],win)
ins<-matrix(ins,nrow=length(proj$y))
require(raster)
#create lon lat in matrix form
loc2d<-inla.mesh.map(mesh$loc, projection = "longlat",inverse = FALSE)
loc2d<-as.data.frame(loc2d)
loc2d$position<-rownames(loc2d)
xy = SpatialPoints(cbind(loc2d[,1],loc2d[,2]))
xy = SpatialPointsDataFrame(cbind(loc2d[,1],loc2d[,2]), loc2d)
proj4string(xy)<-CRS("+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84")
#xy<-xy[studyarea,]
lum02<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum02.tif")
projection(lum02)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
#plot(lum02);plot(country,border="red",col="transparent",add=T)
lum03<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum03.tif")
projection(lum03)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum04<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum04.tif")
projection(lum04)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum05<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum05.tif")
projection(lum05)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum06<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum06.tif")
```

```
projection(lum06)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum07<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum07.tif")
projection(lum07)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum08<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum08.tif")
projection(lum08)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum09<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum09.tif")
projection(lum09)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum10<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum10.tif")
projection(lum10)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum11<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum11.tif")
projection(lum11)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum12<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_R/luminosity/lum12.tif")
projection(lum12)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
lum13<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_ArcGIS/luminosity/F182013.v4c_web.stable_lights.avg_vis.tif")
projection(lum13)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
#extract luminosity values at locations in space and time defined by GTD
    space-time locations (time-consuming!)
lum02<-as.data.frame(lum02<-extract(lum02,xy,method='simple'))
lum03<-as.data.frame(lum03<-extract(lum03,xy,method='simple'))
lum04<-as.data.frame(lum04<-extract(lum04,xy,method='simple'))
lum05<-as.data.frame(lum05<-extract(lum05,xy,method='simple'))
lum06<-as.data.frame(lum06<-extract(lum06,xy,method='simple'))
lum07<-as.data.frame(lum07<-extract(lum07,xy,method='simple'))
lum08<-as.data.frame(lum08<-extract(lum08,xy,method='simple'))
lum09<-as.data.frame(lum09<-extract(lum09,xy,method='simple'))
```

```
lum10<-as.data.frame(lum10<-extract(lum10,xy,method='simple'))
lum11<-as.data.frame(lum11<-extract(lum11,xy,method='simple'))
lum12<-as.data.frame(lum12<-extract(lum12,xy,method='simple'))
lum13<-as.data.frame(lum13<-extract(lum13,xy,method='simple'))
## intercalibration based on Elvidge2013
## Process: 1) calculate: Y = C0 + C1X + C2X^2 // 2) Values > 63 are
   truncated at 63 // 3) values = 0 stay zero.
# Sat   Year      C0          C1       C2
# F15   2002     0.0491   0.9568   0.0010
# F15     2003   0.2217   1.5122   -0.0080
# F16   2004     0.2853   1.1955   -0.0034
# F16     2005   -0.0001  1.4159   -0.0063
# F16     2006   0.1065   1.1371   -0.0016
# F16     2007   0.6394   0.9114   0.0014
# F16   2008     0.5564   0.9931   0.0000
# F16     2009   0.9492   1.0683   -0.0016
# F18   2010     2.3430   0.5102   0.0065
# F18     2011   1.8956   0.7345   0.0030
# F18     2012   1.8750   0.6203   0.0052
f02 <- function(x){ifelse(x>0, 0.0491+0.9568*x+0.0010*x^2, 0)}#if the
   values are above 0, execute intercalibration
f03 <- function(x){ifelse(x>0, 0.2217+1.5122*x+0.0010*x^2, 0)}
f04 <- function(x){ifelse(x>0, 0.2853+1.1955*x+0.0010*x^2, 0)}
f05 <- function(x){ifelse(x>0, -0.0001+1.4159*x+0.0010*x^2, 0)}
f06 <- function(x){ifelse(x>0, 0.1065+1.1371*x+0.0010*x^2, 0)}
f07 <- function(x){ifelse(x>0, 0.6394+0.9114*x+0.0010*x^2, 0)}
f08 <- function(x){ifelse(x>0, 0.5564+0.9931*x+0.0010*x^2, 0)}
f09 <- function(x){ifelse(x>0, 0.9492+1.0683*x+0.0010*x^2, 0)}
f10 <- function(x){ifelse(x>0, 2.3430+0.5102*x+0.0010*x^2, 0)}
f11 <- function(x){ifelse(x>0, 1.8956+0.7345*x+0.0010*x^2, 0)}
f12 <- function(x){ifelse(x>0, 1.8750+0.6203*x+0.0010*x^2, 0)}
f13 <- function(x){ifelse(x>0, 1.8750+0.6203*x+0.0010*x^2, 0)}
lum02<-as.data.frame(sapply(lum02,f02))
lum02[lum02 > 63] <- 63 #truncate if values are higher than 63
lum03<-as.data.frame(sapply(lum03,f03))
lum03[lum03 > 63] <- 63 #truncate if values are higher than 63
lum04<-as.data.frame(sapply(lum04,f04))
lum04[lum04 > 63] <- 63 #truncate if values are higher than 63
lum05<-as.data.frame(sapply(lum05,f05))
lum05[lum05 > 63] <- 63 #truncate if values are higher than 63
```

```
lum06<-as.data.frame(sapply(lum06,f06))
lum06[lum06 > 63] <- 63 #truncate if values are higher than 63
lum07<-as.data.frame(sapply(lum07,f07))
lum07[lum07 > 63] <- 63 #truncate if values are higher than 63
lum08<-as.data.frame(sapply(lum08,f08))
lum08[lum08 > 63] <- 63 #truncate if values are higher than 63
lum09<-as.data.frame(sapply(lum09,f09))
lum09[lum09 > 63] <- 63 #truncate if values are higher than 63
lum10<-as.data.frame(sapply(lum10,f10))
lum10[lum10 > 63] <- 63 #truncate if values are higher than 63
lum11<-as.data.frame(sapply(lum11,f11))
lum11[lum11 > 63] <- 63 #truncate if values are higher than 63
lum12<-as.data.frame(sapply(lum12,f12))
lum12[lum12 > 63] <- 63 #truncate if values are higher than 63
lum13<-as.data.frame(sapply(lum13,f13))
lum13[lum13 > 63] <- 63 #truncate if values are higher than 63
lumls<-list(lum02,lum03,lum04,lum05,lum06,lum07,lum08,lum09,lum10,lum11,
    lum12,lum13)
#normalising data
for (j in 1:k){
lumls[[j]][,1]<-scale(lumls[[j]][,1])
names(lumls[[j]])<-"lum"
}
xydf<-list()
for (j in 1:k){
xydf[[j]]<-as.data.frame(xy)
xydf[[j]]$time<-2001+j
}
#travel time covariate (spatial not temporal)
ttime <- raster("C:/Users/apython/Documents/Andre/Education/StAndrews/
    PhD/PhD_ArcGIS/traveltime/access_50k/acc_50k")
#ttime<-projectRaster(ttime,crs=WGS84)
#projection(ttime)<-"+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+
    ellps=WGS84"
tt<-extract(ttime,xy,method='simple')#extract covariate value at point
    locations
tt<-as.vector(tt)
#replace NA values by 0
tt[is.na(tt)] <- 0
tt<-scale(tt)#normalising data (mean=0, sd=1) for better INLA process
```

```
#Altitude covariate (spatial not temporal)
#source: DEM map (ETOPO1): https://www.ngdc.noaa.gov/mgg/global/relief/
    ETOPO1/image/
altitude <- raster("C:/Users/apython/Documents/Andre/Education/StAndrews
    /PhD/PhD_ArcGIS/DEM/ETOPO1/color_etopo1_ice_full.tif")
projection(altitude)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84
    =0,0,0+ellps=WGS84"
#extract coordinates from population at the location of the points
    coordinates
alt<-extract(altitude,xy,method='simple')#extract covariate value at
    point locations
alt<-as.vector(alt)
#replace NA values by 0
alt[is.na(alt)] <- 0
alt<-scale(alt)#normalising data (mean=0, sd=1) for better INLA process
require(raster)
pop00<-raster("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    PhD_ArcGIS/GPWv3/gldens00/glds00ag_0add")
projection(pop00)<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+
    ellps=WGS84"
#create lon lat in matrix form
#extract coordinates from population at the location of the points
    coordinates
#a bilinearl method + buffer zone is used if data is not provided (long
    duration but ensure no NA values)
pop<-extract(pop00,xy,method='bilinear')#extract covariate value at
    point locations
summary(pop)#check if NA:
pop<-as.vector(pop)
#replace NA values by 0: not do it-this may cause issues if large cities
     are not provide with values and replaced by 0. Should be avoided by
    using bilateral + buffer zone extract.
#pop[is.na(pop)] <- 0
pop<-scale(pop)#normalising data (mean=0, sd=1) for better INLA process
polity<- read.csv("C:/Users/apython/Documents/Andre/Education/StAndrews/
    PhD/PhD_R/PolityIV/p4v2013.csv",header = TRUE)
polity<-subset(polity, select=c("scode","year","polity2"))
polity <-subset(polity, year > 2001 & year < 2014)#selecting the same
    time period than GTD
#rename year as iyear for compatibility with GTD
```

```
require(reshape)
polity<-rename(polity, c(year="iyear"))
country<- readShapeSpatial("C:/Users/apython/Documents/Andre/Education/
    StAndrews/PhD/PhD_ArcGIS/World map/country.shp")#load country borders
     with same characteristics than study area (for plot)
proj4string(country)<-CRS("+proj=longlat +datum=WGS84 +no_defs+towgs84
    =0,0,0+ellps=WGS84")
countrydf<-as.data.frame(country)
polity$time<-polity$iyear-2001
countls<-list()
polpt<-list()
polptdf<-list()
for (j in 1:k){
  countryls<-merge(countrydf,polity[polity$time==j,],by.x="adm0_a3",by.y
     ="scode")
  countls[[j]]<-country
  countls[[j]]@data = data.frame(countls[[j]]@data, countryls[match(
      countls[[j]]@data[,"adm0_a3"],countryls[,"adm0_a3"]),])
  polpt[[j]]<-over(xy, countls[[j]])
  polptdf[[j]]<-as.data.frame(polpt[[j]])
  polptdf[[j]]<-polptdf[[j]][c("polity2","time")]
  polptdf[[j]][,1][is.na(polptdf[[j]][,1])] <- 0
  polptdf[[j]][,1]<-scale(polptdf[[j]][,1])
  names(polptdf[[j]])<-"pol"
}
#end Polity IV covariate (country-level + temporal)
for (j in 1:k){
  xydf[[j]]$lum<-lumls[[j]]$lum
  xydf[[j]]$tt<- tt
  xydf[[j]]$alt<- alt
  xydf[[j]]$pop<- pop
  xydf[[j]]$pol<-polptdf[[j]]$pol
  xydf[[j]]$time<- xydf[[j]]$time-2001
}
#covariates at the mesh vertices in 12 years
xyall<-do.call("rbind", xydf)
xyall<-xyall[c("V1","V2","time","lum","tt","alt","pop","pol")]
attr(xyall$lum, "dimnames") <- NULL
attr(xyall$alt, "dimnames") <- NULL
attr(xyall$tt, "dimnames") <- NULL
```

```
attr(xyall$pop, "dimnames") <- NULL
attr(xyall$pol, "dimnames") <- NULL
xyall$lum<-as.numeric(xyall$lum)
xyall$tt<-as.numeric(xyall$tt)
xyall$alt<-as.numeric(xyall$alt)
xyall$pop<-as.numeric(xyall$pop)
xyall$pol<-as.numeric(xyall$pol)
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/STpoitotal.
    RData")#update file location according to model selection
olddf<-as.data.frame(cbind(y,lum,tt,pop,alt,pol,time,loc))
names(olddf)<-c("y","lum","tt","pop","alt","pol","time","V1","V2","V3")
olddf<-olddf[complete.cases(olddf$y),]
k<-12
spde <- inla.spde2.matern(mesh, alpha=2)
iset<-inla.spde.make.index('i',n.spde=spde$n.spde,n.group=k)
iset2<-inla.spde.make.index('i',n.spde=nrow(xyall)/12,n.group=k)
A<-inla.spde.make.A(mesh,loc=cbind(olddf$V1,olddf$V2,olddf$V3), group=
    olddf$time)
stk <- inla.stack(data=list(y=olddf$y),A=list(A,1,1,1,1,1,1), tag='dat',
                  effects=list(i=iset,lum=olddf$lum,tt=olddf$tt,pop=
                      olddf$pop,alt=olddf$alt,pol=olddf$pol,b0=rep(1,
                      length(olddf$y))))
stk.mesh <- inla.stack(data=list(y=NA), tag='mesh',A=list(1,1,1,1,1,1,1)
    ,
                #         stack2 <- inla.stack(stk, stk.mesh),
                       effects=list(i=iset2,lum=xyall$lum,tt=xyall$tt,
                          pop=xyall$pop,alt=xyall$alt,pol=xyall$pol,b0=
                          rep(1,length(iset2$i))))
idx.mesh <- inla.stack.index(stk.mesh, tag='mesh')$data
lowprob<-as.data.frame(poisson(link='log')$linkinv(resfinal$summary.
    linear.predictor[idx.mesh, "0.025quant"]))
highprob<-as.data.frame(poisson(link='log')$linkinv(resfinal$summary.
    linear.predictor[idx.mesh,'0.975quant']))
lowprob$group<-xyall$time
names(lowprob)<-c("prob","group")
highprob$group<-xyall$time
names(highprob)<-c("prob","group")
bdq25prob<-list()
bdq975prob<-list()
for (j in 1:k){
```

```
  bdq25prob[[j]]<-lowprob$prob[lowprob$group==j]
  bdq975prob[[j]]<-highprob$prob[highprob$group==j]
  bdq25prob[[j]]<-inla.mesh.project(proj,bdq25prob[[j]])#adapt to the
      selected model
  bdq25prob[[j]][!ins]<-NA
  bdq975prob[[j]]<-inla.mesh.project(proj,bdq975prob[[j]])#adapt to the
      selected model
  bdq975prob[[j]][!ins]<-NA
}
#plot nb. lethal events hotspots year after year (with country borders)
#define values for hotspot
require(fields)
library("colorspace")
breaks<-seq(H,maxH,by=0.01)
n<-length(breaks)-1
rm(i)
plot.new()
image.plot(proj$x,proj$y,log(bdq25prob[[2]]),
           col=rev(heat_hcl(n, c = c(120, 0.5), l = c(30, 90), power = c
               (1/10, 5))),
           xlab="",ylab="",main="",xlim = c(-20, 80), ylim = c(-40, 40),
               zlim=c(H-0.5,maxH))#zlim creates identical scale
plot(country,yaxt = "n",add=TRUE)
H<-log(5)#in log scale for better visualisation
maxH<-log(max(unlist(bdq25prob),na.rm=TRUE))
require(fields)
library("colorspace")
breaks<-seq(H,maxH,by=0.1)
n<-length(breaks)-1
dev.off()
plot.new()
for(i in 1:12){
  mypath<-paste("C:/Users/apython/Documents/Andre/Education/StAndrews/
      PhD/revision/revisedPhDthesis/Chapter6/Figs/Raster/photspot",i,".
      png",sep="")
  png(file=mypath)
  par(cex=1.4, cex.lab=0.7)#mar: c(bottom,left,top,right)
  image.plot(proj$x,proj$y,log(bdq25prob[[i]]),
             col=rev(heat_hcl(n, c = c(120, 0.5), l = c(30, 90), power =
                 c(1/10, 5))),
```

```
                xlab="",ylab="",main="",
                xlim = c(-20, 80), ylim = c(-40, 40),zlim=c(H-0.5,maxH))#
                    zlim creates identical scale
  #should have one more break than color
  title(main=2001+i, line = 0.5,cex.main=2.2)
  plot(country,yaxt = "n",add=TRUE)
  mtext("Latitude", side = 2, line = 2.8,adj =0,cex = 1.4,at=-6.5)
  mtext("Longitude", side = 1, line = 3,adj =0,cex = 1.4,at=20)
  dev.off()
}
#Compute diffusion for each year (2003-2012): 2002 excluded since it
    requires j+1 year
#However, hotspots of lethality are identified from 2002 (j) to 2012 (k
    -1) and keep only locations with hotspot (with value to be put to 1)
    and put NA else
hot<-list()
for(j in 1:k){
  hot[[j]]<-ifelse(bdq25prob[[j]]<H,NA,1) #put NA everywhere except
      where there are hotspots (useful for next steps)
}
require(raster)
#create vectors with projection values (x,y) and hotspot values (z)
hotdf<-list()
hotr<-list()
for(j in 1:k){
hotdf[[j]]=list()
hotdf[[j]]$x=proj$x#longitude
hotdf[[j]]$y=proj$y#latitude
hotdf[[j]]$z=hot[[j]]
#convert vectors to raster
hotr[[j]]=raster(hotdf[[j]])
}
#create dataframe from raster
hotspotdf=list()
for(j in 1:k){
hotspotdf[[j]]=as.data.frame(hotr[[j]],xy=TRUE)#add xy option to extract
    lon and lat
names(hotspotdf[[j]])<-c("lon","lat","hot")
}
#delete hotpsot values if NA
```

```
for(j in 1:k){
hotspotdf[[j]]=hotspotdf[[j]][complete.cases(hotspotdf[[j]]),]
}
#idenfiy locations of hotspots cells (cell number of hotspots)
hotloc<-list()
for(j in 1:k){
hotloc[[j]]<-which(!is.na(values(hotr[[j]])))
}
#NEIGHBOURHOOD: OPTIONAL CHOICE OF THE WAY NEIGHBOURHOOD IS DEFINED
#identify the neighbours of hotspots for each j time period
#we chose buffer of approx. 100 km at the equator (could be adjusted)
#create polygon of hotspots from raster cells
hotrpoly<-list()
for(j in 1:k){
  hotrpoly[[j]]<-rasterToPolygons(hotr[[j]], fun=NULL, n=16, na.rm=TRUE,
       digits=5, dissolve=TRUE)
  hotrpoly[[j]]<-unionSpatialPolygons(hotrpoly[[j]], rep(1, length(
      hotrpoly[[j]])))#put adjacent polygons together
  hotrpoly[[j]]<-disaggregate(hotrpoly[[j]])#disaggregate into multiple
      polygons
}
tau<-0.25#(0.25 correspons to PRIO-grid cell area)
areas<-list()
maxareas<-list()
index<-list()
for(j in 1:length(hotrpoly)){
  #calculate areas of all polygons included in hotspots for each year
  areas[[j]]<- sapply(slot(hotrpoly[[j]], "polygons"), function(x)
      sapply(slot(x,"Polygons"), slot, "area"))
  maxareas[[j]]<-sapply(areas[[j]], max)#in case of possible divided
      polygons keep the one with highest area (is required for following
      steps)
  index[[j]] <- which(maxareas[[j]]>tau)#find position where area >tau
  hotrpoly[[j]]<-hotrpoly[[j]][index[[j]],]#keep polygon with area>tau
  }
#some visual checks
# plot(hotrpoly[[1]],col="red",xlim=c(25,45),ylim=c(30,40))
# plot(country,add=TRUE)
#Create buffer
require(rgeos)
```

```
hotbuff<-list()
hotadj<-list()
diffnet<-list()#the final result clipped within studyarea (if option
    below not done)
for(j in 1:k){
  hotbuff[[j]]<-gBuffer(hotrpoly[[j]], byid=FALSE, id=NULL, width=0.9,
      quadsegs=5, capStyle="ROUND",joinStyle="ROUND")
  hotbuff[[j]]<-gDifference(hotbuff[[j]], hotrpoly[[j]])
  proj4string(hotbuff[[j]])<-CRS(proj4string(studyarea))#gives reference
  hotadj[[j]]<-gIntersection(hotbuff[[j]],studyarea)
  hotadj[[j]]<-disaggregate(hotadj[[j]])#from one to multiple polygons
  diffnet[[j]]<-hotadj[[j]]
}
# some checks
# plot.new()
# plot(hotadj[[1]],col="yellow",xlim=c(0,20),ylim=c(30,45))#year 2003
# plot(diffnet[[1]],col="red",add=TRUE)#year 2003
# plot(studyarea,add=TRUE)
#END NEIGHBOURHOOD CHOICE
#create dataframe with projection values (x,y) and lethal probability
    values (z) for lower and higher bounds of CI.
problowdf<-list()
problowr<-list()
for(j in 1:k){
  problowdf[[j]]=list()
  problowdf[[j]]$x=proj$x
  problowdf[[j]]$y=proj$y
  problowdf[[j]]$z=bdq25prob[[j]]#using lower bound CI of probability of
      lethal attack
  #convert dataframe to raster
  problowr[[j]]<-raster(problowdf[[j]])
  proj4string(problowr[[j]])<-CRS(proj4string(studyarea))#required
}
probhighdf<-list()
probhighr<-list()
for(j in 1:k){
  probhighdf[[j]]=list()
  probhighdf[[j]]$x=proj$x
  probhighdf[[j]]$y=proj$y
```

```
  probhighdf[[j]]$z=bdq975prob[[j]]#using lower bound CI of probability
     of lethal attack
  #convert dataframe to raster
  probhighr[[j]]<-raster(probhighdf[[j]])
  proj4string(probhighr[[j]])<-CRS(proj4string(studyarea))#required
}
#extract delta probability of lethal attack at time j and j+1 for
   adacent cells to hotspots
#define kappa (threshold) for enlarging CI (min 0 to max 1) to compute
   signifcant differences between CI in t and t-1
#kappa is between ]0;1]. Values close to 0 indicate high overlap between
    CI and values close to 1 indicate low overlap (more restrictive)
kappa<-0.75 #(0.75: indicate that we tolerate an overlap of CI with 75%
   of the related value)
#for positive diffusion: low p(t+1)-kappa*high p(t)>0
#for negative diffusion: high p(t+1)-kappa*low p(t)<0
#OPTION: WE EXTRACT min and max of prob. values within adjacent polygons
    of hotspots.If there is at least one diffusion or negative diffusion
    within the polygon, we identify it. This is done by using min and
   max of the CI lower and higher bounds.
diffpos<-list()
diffneg<-list()
for(j in 1:(k-1)){  #diffnet starts in 2003 (j: 1 to 11)
diffpos[[j]]<-extract(problowr[[j+1]],diffnet[[j]],fun=max,na.rm=TRUE)-
   extract(kappa*probhighr[[j]],diffnet[[j]],fun=min,na.rm=TRUE)
diffneg[[j]]<-extract(kappa*probhighr[[j+1]],diffnet[[j]],fun=min,na.rm=
   TRUE)-extract(problowr[[j]],diffnet[[j]],fun=max,na.rm=TRUE)
}
#put the result as dataframe
diffposdf<-list()
diffnegdf<-list()
for(j in 1:(k-1)){  #starts in 2003 (j: 1 to 11)
  diffposdf[[j]]<-as.data.frame(diffpos[[j]])
  diffnegdf[[j]]<-as.data.frame(diffneg[[j]])
}
#create ID column for each df (in order to merge dataframe)
for(j in 1:(k-1)){
  diffposdf[[j]]$ID<-1:nrow(diffposdf[[j]])
  diffnegdf[[j]]$ID<-1:nrow(diffnegdf[[j]])
  names(diffposdf[[j]]) <- c("diffpos","ID")
```

```
  names(diffnegdf[[j]]) <- c("diffneg","ID")
}
#create  spatial poly df from spatial polygons.
diffnetdf<-list()
  for(j in 1:(k-1)){
  diffnetdf[[j]]<-SpatialPolygonsDataFrame(diffnet[[j]], data.frame(N =
      1:length(diffnet[[j]]), row.names = 1:length(diffnet[[j]])))
}
#put diffusion values (positive and negative) into sp. poly dfs
for(j in 1:(k-1)){
diffnetdf[[j]]@data <- data.frame(diffnetdf[[j]]@data, diffposdf[[j]][
    match(diffnetdf[[j]]@data[,'N'], diffposdf[[j]][,'ID']),])#N is name
    of variable of spdf and ID name of variable of df which are the
    common variable on which the merge is executed.
}
for(j in 1:(k-1)){
diffnetdf[[j]]@data <- data.frame(diffnetdf[[j]]@data, diffnegdf[[j]][
    match(diffnetdf[[j]]@data[,'N'], diffnegdf[[j]][,'ID']),])#N is name
    of variable of spdf and ID name of variable of df which are the
    common variable on which the merge is executed.
}
#OPTIONAL: delete too small polygon (here we keep polgyons only if the
    area=0.5 (same as PRIO-Grid))
tau<-0.25#(0.25 corresponds to  PRIO-grid cell area)
areas<-list()
maxareas<-list()
indexarea<-list()
diffnetdfnet<-list()
for(j in 1:(k-1)){
  #calculate areas of all polygons included in hotspots neighbourhood
      for each year
  areas[[j]]<- sapply(slot(diffnetdf[[j]], "polygons"), function(x)
      sapply(slot(x,"Polygons"), slot, "area"))
  maxareas[[j]]<-sapply(areas[[j]], max)#in case of possible divided
      polygons keep the one with highest area (is required for following
      steps)
  indexarea[[j]]<- which(maxareas[[j]]>tau)
  diffnetdfnet[[j]]<-diffnetdf[[j]][indexarea[[j]],]#keep polygon with
      area>tau
}
```

```
nbhot<-list()
for(j in 1:(k-1)){
  nbhot[[j]]<-length(disaggregate(diffnetdfnet[[j]]))}
#nbhot<-nbhot[c(-1,-2)]
Reduce("+",nbhot)#nb. of hotspots
#delete the first year because we look at diffusion from 2003 to 2013.
    Indeed, a diffusion from 2002 to 2003 is labelled '2003'.
#diffnetdfnet[[1]] <- NULL
#plot annual diffusion
#color definition
coldiffpos<-list()
for(j in 1:(k-1)){coldiffpos[[j]]<-diffnetdfnet[[j]]@data$diffpos
}
mindiffpos<-0
maxdiffpos<-max(sapply(coldiffpos, max,na.rm=TRUE))
breaks<-seq(mindiffpos,maxdiffpos,by=1)
n<-length(breaks)-1
coldiffneg<-list()
for(j in 1:(k-1)){coldiffneg[[j]]<-diffnetdfnet[[j]]@data$diffneg
}
maxdiffneg<-0
mindiffneg<-min(sapply(coldiffneg, min,na.rm=TRUE))
breaks<-seq(mindiffneg,maxdiffneg,by=0.5)
n<-length(breaks)-1
#plot annual diffusion
dev.off()
plot.new()
#restricted lat and lon: xlim = c(-20, 60), ylim = c(-40, 40),#Africa
    and Middle East for better visualisation
diffnetcrop<-list()
for(j in 1:(k-1)){diffnetcrop[[j]] <- crop(diffnetdfnet[[j]], extent
    (-20, 80, -40, 40))}
plot.new()
for(j in 1:(k-1)){
  mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD
      /revision/revisedPhDthesis/Chapter6/Figs/Raster/pdiffusion",j,".png
      ",sep="")
  png(file=mypath)
  par(cex=1.4, cex.lab=0.7)
  plot(diffnetcrop[[j]],xlim=c(-20,80),ylim=c(-40,40),
```

```
        col=ifelse(diffnetcrop[[j]]@data$diffpos>0 & diffnetcrop[[j]]
            @data$diffneg>=0 ,"red",ifelse(diffnetcrop[[j]]@data$diffneg<0
             & diffnetcrop[[j]]@data$diffpos<=0,"blue","grey25")),main=""
        ,xlab="",ylab="",axes=F)
  box()
  axis(1, col.axis="black",at=seq(-20,80,by=20),labels=seq(-20,80,by=20)
      , las=2)
  axis(2, col.axis="black", at=seq(-40,40,by=20),labels=seq(-40,40,by
     =20),las=0)#
  plot(country,add=TRUE)
  title(main=2002+j, line = 0.5,cex.main=1.5)
  mtext("Latitude", side = 2, line = 2,adj =0,cex = 1.4,at=-7.5)
  mtext("Longitude", side = 1, line = 2,adj =0,cex = 1.4,at=20)
    legend("bottomleft",title="",inset=.005,cex=1,bty="n",
         c("diffusion","dissipation","not␣significant"),fill=c("red","
            blue","grey25"))
  dev.off()
}
#Escalation (change in terrorism intensity within hotspot (at the cell
    level))
#annual escalation
#extract delta probability of lethal attack at time j and j+1 within
    hotspots
#define kappa (threshold) for enlarging CI (min 0 to max 1) to compute
    signifcant differences between CI in t and t-1
kappa<-0.75 #(0.75: indicate that we tolerate an overlap of CI with 75%
    of the related value)
#for positive escalation: low p(t+1)-kappa*high p(t)>0
#for negative escalation: high p(t+1)-kappa*low p(t)<0
require(raster)
escpos<-list()
escneg<-list()
for(j in 1:(k-1)){
  escpos[[j]]<-extract(problowr[[j+1]],hotloc[[j]])-extract(kappa*
      probhighr[[j]],hotloc[[j]])
  escneg[[j]]<-extract(kappa*probhighr[[j+1]],hotloc[[j]])-extract(
      problowr[[j]],hotloc[[j]])
}
#join escalation values with cell number of hotspotscells
```

```
#create dataframe with projection values (x,y) and lethal probability
    values (z)
escdf<-list()
for(j in 1:(k-1)){
  escdf[[j]]<-as.data.frame(cbind(escpos[[j]],escneg[[j]],hotloc[[j]]))
}
#extract raster cells with values of prob of lethal at the location of
    hotspot cells
#Done for each year from 2002 to 2013
escxyz<-list()
for(j in 1:k){
  escxyz[[j]]<-xyFromCell(probhighr[[j]],hotloc[[j]])
}
#Add xy location and delta values from j to k-1 to the dataframe for eac
    j year. Note that delta have 11 values (k-1), no values calculated
    for time k (2013)
#Done for each year from 2003 to 2012
for(j in 1:(k-1)){
  escdf[[j]]<-cbind(escdf[[j]],escxyz[[j]])
}
#Rename variables in dataframe: positive, negative escalation, cell
    number, longitude, and latitude,
for(j in 1:(k-1)){
  colnames(escdf[[j]]) <- c("posesc","negesc","cellnum","lon","lat")
}
#delete escdf observations if NA is present (if no esc values are
    provided. For example, when adjacent cell lie in ocean)
for(j in 1:(k-1)){
  escdf[[j]]=escdf[[j]][complete.cases(escdf[[j]]),]
}
#Escalation
#create spatial points dataframe from escdf dataframe
cooresc<-list()
escpt<-list()
for(i in 1:(k-1))
  {
#escalation data
cooresc[[i]]<-cbind(escdf[[i]]$lon,escdf[[i]]$lat)
escpt[[i]]<-SpatialPointsDataFrame(cooresc[[i]], escdf[[i]],proj4string=
    CRS("+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+ellps=WGS84"))
```

```
escpt[[i]]<-escpt[[i]][studyarea,]#problem!!!!CRS
escpt[[i]]<-remove.duplicates(escpt[[i]])
   }
#plot annual escalation
#color definition
colescpos<-list()
for(j in 1:(k-1)){colescpos[[j]]<-escpt[[j]]$posesc}
minescpos<-0
maxescpos<-max(sapply(colescpos, max,na.rm=TRUE))
breaks<-seq(minescpos,maxescpos,by=1)
n<-length(breaks)-1
colescneg<-list()
for(j in 1:(k-1)){colescneg[[j]]<-escpt[[j]]$negesc}
maxescneg<-0
minescneg<-min(sapply(colescneg, min,na.rm=TRUE))
breaks<-seq(minescneg,maxescneg,by=1)
n<-length(breaks)-1
#plot annual escalation
plot.new()
dev.off()
for(j in 1:(k-1))
{
  mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD
      /revision/revisedPhDthesis/Chapter6/Figs/Raster/pescalation",j,".
      png",sep="")
  png(file=mypath)
  par(cex=1.4, cex.lab=0.7)
  plot(escpt[[j]]$lon,escpt[[j]]$lat,pch=15,cex=0.4,col="NA",main=""
       ,xlim = c(-20, 80), ylim = c(-40, 40),
       xlab="",ylab="")
  points(escpt[[j]]$lon,escpt[[j]]$lat,pch=15,cex=0.2,col=ifelse(escpt[[
      j]]$posesc>0,"red",ifelse(escpt[[j]]$negesc<0 , "blue", "NA")))
  plot(country,add=TRUE)
  title(main=2002+j, line = 0.5,cex.main=1.5)
  mtext("Latitude", side = 2, line = 1.9,adj =0,cex = 1.4,at=-7.5)
  mtext("Longitude", side = 1, line = 1.9,adj =0,cex = 1.4,at=20)
    legend("bottomleft",title="",inset=.005,cex=1,bty="n",
         c("escalation","de-escalation"),fill=c("red","blue"))
    dev.off()
}
```

```
save.image("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/STpoi_
    diffusion.RData")
#END
```

D.2 Detection of Hotspot Processes

This is the code used to assess the performance of the model with regard to the identified hotspots of the probability of lethality of terrorism and real observations.

```
#SPATIO-TEMPORAL MODELLING / Bernoulli model of the probability of
    lethal events worldwide 2002-2013
#The code is used to assess the performance of the model with regard to
    the identified hotspots of lethal terrorism and real observations.
setwd("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal")
rm(list=ls())
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/SPDEfinal.RData")
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal/STbin_
    diffusion.RData")
require(foreign)
require(sp)
library(INLA)
library(spatstat)
library(fields)
library(maptools)
library(fields)
library(lattice)
require(foreign)
k<-12
spde <- inla.spde2.matern(mesh, alpha=2)
iset<-inla.spde.make.index('i',n.spde=spde$n.spde,n.group=k)
iset2<-inla.spde.make.index('i',n.spde=nrow(xyall)/12,n.group=k)
A<-inla.spde.make.A(mesh,loc=cbind(olddf$V1,olddf$V2,olddf$V3), group=
    olddf$time)
stk <- inla.stack(data=list(y=olddf$y),A=list(A,1,1,1,1,1), tag='dat',
                  effects=list(i=iset,lum=olddf$lum,tt=olddf$tt,greg=
                      olddf$greg,alt=olddf$alt,b0=rep(1,length(olddf$y)))
                      )
stk.mesh <- inla.stack(data=list(y=NA), tag='mesh',A=list(1,1,1,1,1,1),
                       #  stack2 <- inla.stack(stk, stk.mesh),
```

```
                        effects=list(i=iset2,lum=xyall$lum,tt=xyall$tt,
                            greg=xyall$greg,alt=xyall$alt,b0=rep(1,length(
                            iset2$i))))
idx.mesh <- inla.stack.index(stk.mesh, tag='mesh')$data
lowprob<-as.data.frame(binomial(link='logit')$linkinv(resfinal$summary.
    linear.predictor[idx.mesh, "0.025quant"]))
highprob<-as.data.frame(binomial(link='logit')$linkinv(resfinal$summary.
    linear.predictor[idx.mesh,'0.975quant']))
lowprob$group<-xyall$time
names(lowprob)<-c("prob","group")
highprob$group<-xyall$time
names(highprob)<-c("prob","group")
bdq25prob<-list()
bdq975prob<-list()
for (j in 1:k){
  bdq25prob[[j]]<-lowprob$prob[lowprob$group==j]
  bdq975prob[[j]]<-highprob$prob[highprob$group==j]
  bdq25prob[[j]]<-inla.mesh.project(proj,bdq25prob[[j]])#adapt to the
      selected model
  bdq25prob[[j]][!ins]<-NA
  bdq975prob[[j]]<-inla.mesh.project(proj,bdq975prob[[j]])#adapt to the
      selected model
  bdq975prob[[j]][!ins]<-NA
}
#lethality hotspots are identified from 2002 (j) to 2013 (k) and keep
    only locations with hotspot (with value to be put to 1) and put NA
    else
H<-0.5#define lethal hotspot threshold probability
hot<-list()
for(j in 1:k){
  hot[[j]]<-ifelse(bdq25prob[[j]]<H,NA,1) #put NA everywhere except
      where there are hotspots (useful for next steps)
}
require(raster)
#create vectors with projection values (x,y) and hotspot values (z)
hotdf<-list()
hotr<-list()
for(j in 1:k){
hotdf[[j]]=list()
hotdf[[j]]$x=proj$x#longitude
```

```
hotdf[[j]]$y=proj$y#latitude
hotdf[[j]]$z=hot[[j]]
#convert vectors to raster
hotr[[j]]=raster(hotdf[[j]])
}
#create dataframe from raster
hotspotdf=list()
for(j in 1:k){
hotspotdf[[j]]=as.data.frame(hotr[[j]],xy=TRUE)#add xy option to extract
    lon and lat
names(hotspotdf[[j]])<-c("lon","lat","hot")
}
#delete hotpsot values if NA
for(j in 1:k){
hotspotdf[[j]]=hotspotdf[[j]][complete.cases(hotspotdf[[j]]),]
}
#idenfiy locations of hotspots cells (cell number of hotspots)
hotloc<-list()
for(j in 1:k){
hotloc[[j]]<-which(!is.na(values(hotr[[j]])))
}
#create polygons from hotspots
hotpoly<-list()
hotinter<-list()
hotdis<-list()
WGS84="+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+ellps=WGS84"
for(i in 1:12){
hotpoly[[i]]<-rasterToPolygons(hotr[[i]], fun=NULL, n=8, na.rm=TRUE,
    digits=5, dissolve=TRUE)
proj4string(hotpoly[[i]]) = CRS(WGS84)
hotinter[[i]]<-gIntersection(studyarea,hotpoly[[i]])#optional (may
    provide better results): intersect with studyarea (to be reviewed)
hotdis[[i]]<-disaggregate(hotpoly[[i]])#disaggregate into multiple
    polygons
#hotdis[[i]]<-gSimplify(hotdis[[i]], 0.05, topologyPreserve=TRUE)#
    OptionalL: slighly simplify polygons with Douglas Peucker method (if
    some polygons are too complex)
}
#OPTION (recommended): remove polygons with area < tau in deg.
```

```
#if the area of the hotspot is very small, the probability of finding
    observations is close to 0 too. Therefore, we suggest to remove very
    small polygons.
#threshold area: tau#
tau<-0.25#(0.25 corresponds to PRIO-grid cell area)
areas<-list()
maxareas<-list()
index<-list()
for(i in 1:length(hotdis)){
#calculate areas of all polygons included in hotspots for each year
areas[[i]]<- sapply(slot(hotdis[[i]], "polygons"), function(x) sapply(
    slot(x,"Polygons"), slot, "area"))
maxareas[[i]]<-sapply(areas[[i]], max)#in case of possible divided
    polygons keep the one with highest area (is required for following
    steps)
index[[i]] <- which(maxareas[[i]]>tau)#find position where area >tau
}
hotnet<-list()
for(i in 1:length(hotdis)){
hotnet[[i]]<-hotdis[[i]][index[[i]],]
}
#count nb. of hotspots (from the model, including true and not true
    hostspots) with area > tau
nbhot<-list()
for(j in 1:k){
  nbhot[[j]]<-length(disaggregate(hotnet[[j]]))}
#nbhot<-nbhot[c(-1,-2)]
Reduce("+",nbhot)#nb. of hotspots 166
#extract lethal and non-lethal events for each year from 2002-2013
let<-list()
for(i in 1:12){
  let[[i]]<-GTD1[which(GTD1$nkill > 0 & GTD1$iyear==2001+i), ]
}
nlet<-list()
for(i in 1:12){
  nlet[[i]]<-GTD1[which(GTD1$nkill == 0 & GTD1$iyear==2001+i), ]
}
#create spatial points for each year (required to count points in
    polygon)
WGS84="+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+ellps=WGS84"
```

```
plet<-list()
pnlet<-list()
for(i in 1:12){
  plet[[i]]<-SpatialPoints(cbind(let[[i]]$longitude,let[[i]]$latitude))#
     create spatial points
  pnlet[[i]]<-SpatialPoints(cbind(nlet[[i]]$longitude,nlet[[i]]$latitude
     ))
  proj4string(plet[[i]]) = CRS(WGS84)
  proj4string(pnlet[[i]]) = CRS(WGS84)
}
#counting observed proportion of lethal attacks in hotspots from the
   model
require(GISTools)
nblet<-list()
nbnlet<-list()
proplet<-list()
for(i in 1:12){
 nblet[[i]]<-poly.counts(plet[[i]], hotnet[[i]])#count lethal attacks in
     hotspot
 nbnlet[[i]]<-poly.counts(pnlet[[i]], hotnet[[i]])#count non lethal
     attacks in hotspot
 proplet[[i]]<-nblet[[i]]/(nblet[[i]]+nbnlet[[i]])#compute the
     proportion of lethal attack in hotspot
 #OPTION to be discussed
 #proplet[[i]]<-proplet[[i]][!is.na(proplet[[i]])]#delete when no
     observation (this is an alternative approach to deleting small
     polygons)
 }
#calculate the proportion of hotspots that have a proportion of lethal
   event >= tau.
proplet<-unlist(proplet)
sum(proplet >= H,na.rm=TRUE)#number of hotspots that contain >= H of
   lethal event.
length(proplet)#number of hotspots
sum((proplet >= H)/length(proplet),na.rm=TRUE)#proportion of hotspots
   that contain >= H of lethal event.
summary(proplet,na.rm=TRUE)
sd(proplet,na.rm=TRUE)
save.image("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal/
   hotspot_modelvsobs2.RData")
```

This is the code used to assess the performance of the model with regard to the identified hotspots of the number of lethal terrorist events and real observations.

```
#SPATIO-TEMPORAL MODELLING / Poisson model of the number of lethal
    events worldwide 2002-2013
#The code assesses the performance of the model with regard to the
    identified hotspots of lethal terrorism and real observations.
setwd("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson")
rm(list=ls())
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/SPDE_nbevents
    _final.RData")
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/STpoi_
    diffusion.RData")
require(foreign)
require(sp)
library(INLA)
library(spatstat)
library(fields)
library(maptools)
library(fields)
library(lattice)
require(foreign)
require(rgeos)
k<-12
spde <- inla.spde2.matern(mesh, alpha=2)
iset<-inla.spde.make.index('i',n.spde=spde$n.spde,n.group=k)
iset2<-inla.spde.make.index('i',n.spde=nrow(xyall)/12,n.group=k)
A<-inla.spde.make.A(mesh,loc=cbind(olddf$V1,olddf$V2,olddf$V3), group=
    olddf$time)
stk <- inla.stack(data=list(y=olddf$y),A=list(A,1,1,1,1,1,1), tag='dat',
                  effects=list(i=iset,lum=olddf$lum,tt=olddf$tt,pop=
                      olddf$pop,alt=olddf$alt,pol=olddf$pol,b0=rep(1,
                      length(olddf$y))))
stk.mesh <- inla.stack(data=list(y=NA), tag='mesh',A=list(1,1,1,1,1,1,1)
    ,
                       #         stack2 <- inla.stack(stk, stk.mesh),
                       effects=list(i=iset2,lum=xyall$lum,tt=xyall$tt,
                           pop=xyall$pop,alt=xyall$alt,pol=xyall$pol,b0=
                           rep(1,length(iset2$i))))
idx.mesh <- inla.stack.index(stk.mesh, tag='mesh')$data
```

```
lowprob<-as.data.frame(poisson(link='log')$linkinv(resfinal$summary.
    linear.predictor[idx.mesh, "0.025quant"]))
highprob<-as.data.frame(poisson(link='log')$linkinv(resfinal$summary.
    linear.predictor[idx.mesh,'0.975quant']))
lowprob$group<-xyall$time
names(lowprob)<-c("prob","group")
highprob$group<-xyall$time
names(highprob)<-c("prob","group")
bdq25prob<-list()
bdq975prob<-list()
for (j in 1:k){
  bdq25prob[[j]]<-lowprob$prob[lowprob$group==j]
  bdq975prob[[j]]<-highprob$prob[highprob$group==j]
  bdq25prob[[j]]<-inla.mesh.project(proj,bdq25prob[[j]])#adapt to the
      selected model
  bdq25prob[[j]][!ins]<-NA
  bdq975prob[[j]]<-inla.mesh.project(proj,bdq975prob[[j]])#adapt to the
      selected model
  bdq975prob[[j]][!ins]<-NA
}
#lethality hotspots are identified from 2002 (j) to 2013 (k) and keep
    only locations with hotspot (with value to be put to 1) and put NA
    else
##################################################
H<-5#define lethal hotspot threshold probability#
##################################################
hot<-list()
for(j in 1:k){
  hot[[j]]<-ifelse(bdq25prob[[j]]<H,NA,1) #put NA everywhere except
      where there are hotspots (useful for next steps)
}
require(raster)
#create vectors with projection values (x,y) and hotspot values (z)
hotdf<-list()
hotr<-list()
for(j in 1:k){
hotdf[[j]]=list()
hotdf[[j]]$x=proj$x#longitude
hotdf[[j]]$y=proj$y#latitude
hotdf[[j]]$z=hot[[j]]
```

```
#convert vectors to raster
hotr[[j]]=raster(hotdf[[j]])
}
#create dataframe from raster
hotspotdf=list()
for(j in 1:k){
hotspotdf[[j]]=as.data.frame(hotr[[j]],xy=TRUE)#add xy option to extract
    lon and lat
names(hotspotdf[[j]])<-c("lon","lat","hot")
}
#delete hotpsot values if NA
for(j in 1:k){
hotspotdf[[j]]=hotspotdf[[j]][complete.cases(hotspotdf[[j]]),]
}
#idenfiy locations of hotspots cells (cell number of hotspots)
hotloc<-list()
for(j in 1:k){
hotloc[[j]]<-which(!is.na(values(hotr[[j]])))
}
#create polygons from hotspots
hotpoly<-list()
hotinter<-list()
hotdis<-list()
WGS84="+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+ellps=WGS84"
for(i in 1:12){
  hotpoly[[i]]<-rasterToPolygons(hotr[[i]], fun=NULL, n=8, na.rm=TRUE,
      digits=5, dissolve=TRUE)
  proj4string(hotpoly[[i]]) = CRS(WGS84)
  hotinter[[i]]<-gIntersection(studyarea,hotpoly[[i]])#optional (may
      provide better results): intersect with studyarea (to be reviewed)
  hotdis[[i]]<-disaggregate(hotpoly[[i]])#disaggregate into multiple
      polygons
  #hotdis[[i]]<-gSimplify(hotdis[[i]], 0.05, topologyPreserve=TRUE)#
      OptionalL: slighly simplify polygons with Douglas Peucker method (
      if some polygons are too complex)
}
########if warning message no values in selection#######################
#check if there are years with no hotspot (very likely) and remove them#
#CAREFUL!!!!! SEE WHICH YEAR THERE IS NO HOTSPOT (HERE YEAR 1)          #
```

```
#hotpoly<-hotpoly[!sapply(hotpoly, is.null)]
    #
#end careful check ######################################################
hotinter<-list()
hotdis<-list()
for(i in 1:length(hotpoly)){
proj4string(hotpoly[[i]]) = CRS(WGS84)
hotinter[[i]]<-gIntersection(studyarea,hotpoly[[i]])#optional (may
    provide better results): intersect with studyarea (to be reviewed)
hotdis[[i]]<-disaggregate(hotpoly[[i]])#disaggregate into multiple
    polygons
#hotdis[[i]]<-gSimplify(hotdis[[i]], 0.05, topologyPreserve=TRUE)#
    OptionalL: slighly implify polygons with Douglas Peucker method (some
     could be too complex)
}
#1 OPTION (recommended): remove polygons with area < tau in deg.
#if the area of the hotspot is very small, the probability of finding
    observations is close to 0 too. Therefore, we suggest to remove very
    small polygons.
#threshold area: tau#
tau<-0.25#(0.25 correspons to PRIO-grid cell area)
areas<-list()
maxareas<-list()
index<-list()
for(i in 1:length(hotdis)){
#calculate areas of all polygons included in hotspots for each year
areas[[i]]<- sapply(slot(hotdis[[i]], "polygons"), function(x) sapply(
    slot(x,"Polygons"), slot, "area"))
maxareas[[i]]<-sapply(areas[[i]], max)#in case of possible divided
    polygons keep the one with highest area (is required for following
    steps)
index[[i]] <- which(maxareas[[i]]>tau)#find position where area >tau
}
#########################################################################
#check if there are years with no hotspot (very likely) and remove them#
#CAREFUL!!!!! SEE WHICH YEAR THERE IS NO HOTSPOT, here year 2 too small#
hotnet<-list()
for(i in 1:length(hotdis)){
hotnet[[i]]<-hotdis[[i]][index[[i]],]
}
```

```
nbhot<-list()
for(j in 1:k){
  nbhot[[j]]<-length(disaggregate(hotnet[[j]]))}
#nbhot<-nbhot[c(-1,-2)]
Reduce("+",nbhot)#nb. of hotspots 189 with area > tau
plot(hotnet[[1]])
#########################################################################
#check if there are years with no hotspot (very likely) and remove them#
#CAREFUL!!!!! SEE WHICH YEAR THERE IS NO HOTSPOT (HERE YEAR 1)          #
summary(hotnet)                                  #
#delete hotnet[[1]] (year 2003): has no element
#hotnet[[1]]<-NULL
########CAREFUL CHECK WITH correspondance year hotspot and year GTD
#CAREFUL WITH extraction of years (since hotspot have missing years
    (2002,2003))
#extract lethal and non-lethal events for each year from 2003-2013
let<-list()
for(i in 1:12){
  let[[i]]<-GTD1[which(GTD1$nkill > 0 & GTD1$iyear==2001+i), ]
}
#create spatial points for each year (required to count points in
    polygon)
WGS84="+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+ellps=WGS84"
plet<-list()
for(i in 1:12){
  plet[[i]]<-SpatialPoints(cbind(let[[i]]$longitude,let[[i]]$latitude))#
      create spatial points
  proj4string(plet[[i]]) = CRS(WGS84)
}
#counting observed propoortion of lethal attacks in hotspots from the
    model
require(GISTools)
nblet<-list()
for(i in 1:12){
 nblet[[i]]<-poly.counts(plet[[i]], hotnet[[i]])#count lethal attacks in
      hotspot
 }
#calculate the proportion of hotspots that have number of lethal event
    >= H.
nblet<-unlist(nblet)
```

```
sum(nblet >= H,na.rm=TRUE)#number of hotspots that contain >= H of
    lethal event.
length(nblet)#number of hotspots
sum((nblet >= H)/length(nblet),na.rm=TRUE)#proportion of hotspots that
    contain >= H of lethal event.
summary(nblet,na.rm=TRUE)
sd(nblet,na.rm=TRUE)
save.image("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/hotspot
    _modelvsobs.RData")
```

D.3 Detection of Escalation Processes

This is the code used to assess the performance of the model with regard to escalation and de-escalation of the lethality of terrorism and real observations.

```
#SPATIO-TEMPORAL MODELLING / Bernoulli model of probability of lethal
    events worldwide 2002-2013
#The code assesses the performance of the model with regard to the
    escalation of the lethality of terrorism and real observations.
setwd("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal")
rm(list=ls())
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/SPDE.RData")
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal/STbin_
    diffusion.RData")
require(foreign)
require(sp)
library(INLA)
library(spatstat)
library(fields)
library(maptools)
library(fields)
library(lattice)
require(rgeos)
require(raster)
require(rgdal)
#Identify escalation areas and construct polygon from 2003 to 2013
#create polygons from escalation and de-escalation areas separately
#the polygons are geometrically the same but contain different values
rpos<-list()#escalation polygons
xypos<-list()#coordinates of escalation spatial points
```

```
raspos<-list()#escalation raster
rneg<-list()#de-escalation
xyneg<-list()
rasneg<-list()
rbase<-raster(ncols=1440, nrows=720)
for(i in 1:length(escpt)){
  xypos[[i]] <- cbind(escpt[[i]]$lon, escpt[[i]]$lat)
  raspos[[i]] <- rasterize(xypos[[i]], rbase, field=escpt[[i]]$posesc)
  rpos[[i]]<-rasterToPolygons(raspos[[i]], fun=NULL, n=8, na.rm=TRUE,
      digits=5, dissolve=TRUE)
   rpos[[i]]<-unionSpatialPolygons(rpos[[i]], rep(1, length(rpos[[i]])))
      #put adjacent polygons together
  rpos[[i]]<-disaggregate(rpos[[i]])#disaggregate into multiple polygons
  xyneg[[i]] <- cbind(escpt[[i]]$lon, escpt[[i]]$lat)
  rasneg[[i]] <- rasterize(xyneg[[i]], rbase, field=escpt[[i]]$negesc)
  rneg[[i]]<-rasterToPolygons(rasneg[[i]], fun=NULL, n=8, na.rm=TRUE,
      digits=5, dissolve=TRUE)
  rneg[[i]]<-unionSpatialPolygons(rneg[[i]], rep(1, length(rneg[[i]])))#
      put adjacent polygons together
  rneg[[i]]<-disaggregate(rneg[[i]])#disaggregate into multiple polygons
}
#some checks
#plot(rpos[[11]],col="red")

#OPTION (recommended): remove polygons with area < tau deg
#if the area of the escalation is very small, the probability of finding
     observations is close to 0 too. Therefore, we suggest to remove very
     small polygons.
#threshold area: tau
tau<-0.25#(0.25 corresponds to PRIO-grid cell area)
areaspos<-list()
maxareaspos<-list()
indexpos<-list()
areasneg<-list()
maxareasneg<-list()
indexneg<-list()#length rpos idem rneg
for(i in 1:length(rpos)){
#calculate areas of all polygons included in escalation areas for each
    year
```

```
areaspos[[i]]<- sapply(slot(rpos[[i]], "polygons"), function(x) sapply(
    slot(x,"Polygons"), slot, "area"))
maxareaspos[[i]]<-sapply(areaspos[[i]], max)#in case of possible divided
     polygons keep the one with highest area (is required for following
    steps)
indexpos[[i]] <- which(maxareaspos[[i]]>tau)#find position where area >
    tau
areasneg[[i]]<- sapply(slot(rneg[[i]], "polygons"), function(x) sapply(
    slot(x,"Polygons"), slot, "area"))
maxareasneg[[i]]<-sapply(areasneg[[i]], max)#in case of possible divided
     polygons keep the one with highest area (is required for following
    steps)
indexneg[[i]] <- which(maxareasneg[[i]]>tau)#find position where area >
    tau
}
#keep polygon with area>tau
escposnet<-list()
escnegnet<-list()
for(i in 1: length(rpos)){#length rpos idem rneg...
escposnet[[i]]<-rpos[[i]][indexpos[[i]],]#keep polygon with area>tau
escnegnet[[i]]<-rneg[[i]][indexneg[[i]],]#keep polygon with area>tau
}
#extract escalation /de-escalation values from raster to polygons
posvalue<-list()
negvalue<-list()
for(i in 1: length(escposnet)){#length rpos idem rneg...
posvalue[[i]]<-extract(raspos[[i]],escposnet[[i]],fun=max,na.rm=FALSE,df
    =TRUE)
negvalue[[i]]<-extract(rasneg[[i]],escnegnet[[i]],fun=min,na.rm=FALSE,df
    =TRUE)
}
#extract lethal and non-lethal events for each year from 2002-2013
let<-list()
for(i in 1:12){
  let[[i]]<-GTD1[which(GTD1$nkill > 0 & GTD1$iyear==2001+i), ]
}
nlet<-list()
for(i in 1:12){
  nlet[[i]]<-GTD1[which(GTD1$nkill == 0 & GTD1$iyear==2001+i), ]
}
```

```
#create spatial points for each year (required to count points in
    polygon)
WGS84="+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+ellps=WGS84"
plet<-list()
pnlet<-list()
for(i in 1:12){
  plet[[i]]<-SpatialPoints(cbind(let[[i]]$longitude,let[[i]]$latitude))#
      create spatial points
  pnlet[[i]]<-SpatialPoints(cbind(nlet[[i]]$longitude,nlet[[i]]$latitude
      ))
  proj4string(plet[[i]]) = CRS(WGS84)
  proj4string(pnlet[[i]]) = CRS(WGS84)
}
#counting observed proportion of lethal attacks in escalation areas in
    TIME T+1
require(GISTools)
nblet<-list()
nbnlet<-list()
proplet<-list()
for(i in 1:11){#TIME T+1: 11 observations of lethal attacks from 2003 to
     2013
   nblet[[i]]<-poly.counts(plet[[i+1]], escposnet[[i]])#count lethal
       attacks in hotspot
  nbnlet[[i]]<-poly.counts(pnlet[[i+1]], escposnet[[i]])#count non
      lethal attacks in hotspot
  proplet[[i]]<-nblet[[i]]/(nblet[[i]]+nbnlet[[i]])#compute the
      proportion of lethal attack in hotspot
}
#counting observed propoortion of lethal attacks in escalation areas in
    TIME T
 nblet2<-list()
 nbnlet2<-list()
 proplet2<-list()
 for(i in 1:11){#TIME T-1: 11 observations of lethal attacks from 2002
     to 2012
   nblet2[[i]]<-poly.counts(plet[[i]], escposnet[[i]])#count lethal
       attacks in hotspot
  nbnlet2[[i]]<-poly.counts(pnlet[[i]], escposnet[[i]])#count non lethal
       attacks in hotspot
```

```
  proplet2[[i]]<-nblet2[[i]]/(nblet2[[i]]+nbnlet2[[i]])#compute the
      proportion of lethal attack in hotspot
    }
#compute the ratio in prop. lethal events from 2003 to 2013 (For ex.
   2003 indexes ratio. 2003/2002)
 deltaesc<-list()
  for(i in 1:11){
  deltaesc[[i]]<-proplet[[i]]/proplet2[[i]]#compute the ratio between
      proportion of lethal attack.
 }
#put escalation and de-escalation dataframe together
escaldf<-list()
for(i in 1:11){
escaldf[[i]]<-merge(posvalue[[i]],negvalue[[i]], by='ID')
colnames(escaldf[[i]]) <- c("ID","posesc","negesc")
}
#create dataframe with delta values of prop. lethal attack
deltaescdf<-list()
for(i in 1:11){
  deltaescdf[[i]]<-as.data.frame((deltaesc[[i]]))
  deltaescdf[[i]]$ID<-seq.int(nrow(deltaescdf[[i]]))#add id column
  colnames(deltaescdf[[i]]) <- c("deltaesc","ID")
}
#merge both dataframes
escalation<-list()
for(i in 1:11){
escalation[[i]]<-merge(escaldf[[i]],deltaescdf[[i]],by="ID")
}
#putting all together
escalation<-rbind(escalation[[1]],escalation[[2]],escalation[[3]],
    escalation[[4]],escalation[[5]],escalation[[6]],
            escalation[[7]],escalation[[8]],escalation[[9]],escalation
                [[10]],escalation[[11]])
#define threshold of the ratio for an escalation to be significant
escalt<-1.1
#define threshold of the ratio for an escalation to be significant
descalt<-0.9
#creating score variables with regard to prediciton of escalation / de-
    escalation
escalation$totescpred<-ifelse(escalation$posesc>0,1,0)
```

```
escalation$escpred<-ifelse(escalation$posesc>0 & escalation$deltaesc>=
    escalt,1,0)
escalation$totdeescpred<-ifelse(escalation$negesc<0,1,0)
escalation$deescpred<-ifelse(escalation$negesc<0 & escalation$deltaesc<=
    descalt,1,0)
#calculate the proportion of good escalation prediction
sum(escalation$escpred,na.rm=TRUE)
sum(escalation$totescpred,na.rm=TRUE)
sum(escalation$escpred,na.rm=TRUE)/sum(escalation$totescpred,na.rm=TRUE)
#calculate the proportion of good de-escalation prediction
sum(escalation$deescpred,na.rm=TRUE)
sum(escalation$totdeescpred,na.rm=TRUE)
sum(escalation$deescpred,na.rm=TRUE)/sum(escalation$totdeescpred,na.rm=
    TRUE)
#calculate sum of original escalation areas (before sample reduction
    with areas > tau)
len<-vector()
for(i in 1:11){
  len[[i]]<-length(rpos[[i]])
}
sum(len)
#calculate sum of escalation areas (after sample reduction with areas >
    tau)
lenesc<-vector()
for(i in 1:11){
  lenesc[[i]]<-length(escposnet[[i]])
}
sum(lenesc)
#end
```

This is the code used to assess the performance of the model with regard to escalation and de-escalation of the number of lethal terrorist events and real observations.

```
#SPATIO-TEMPORAL MODELLING / Poisson model of the number of lethal events
    worldwide 2002-2013
#The code assesses the performance of the model with regard to the
    escalation of lethal terrorism and real observations.
setwd("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson")
rm(list=ls())
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/SPDE_nbevents
    _final.RData")
```

```
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/STpoi_
    diffusion.RData")
require(foreign)
require(sp)
library(INLA)
library(spatstat)
library(fields)
library(maptools)
library(fields)
library(lattice)
require(rgeos)
require(raster)
require(rgdal)
#Identify escalation areas and construct polygon from 2003 to 2013

#create polygons from escalation and de-escalation areas separately
#the polygons are geometrically the same but contain different values
rpos<-list()#escalation polygons
xypos<-list()#coordinates of escalation spatial points
raspos<-list()#escalation raster
rneg<-list()#de-escalation
xyneg<-list()
rasneg<-list()
rbase<-raster(ncols=1440, nrows=720)
for(i in 1:length(escpt)){
  xypos[[i]] <- cbind(escpt[[i]]$lon, escpt[[i]]$lat)
  raspos[[i]] <- rasterize(xypos[[i]], rbase, field=escpt[[i]]$posesc)
  rpos[[i]]<-rasterToPolygons(raspos[[i]], fun=NULL, n=8, na.rm=TRUE,
      digits=5, dissolve=TRUE)
   rpos[[i]]<-unionSpatialPolygons(rpos[[i]], rep(1, length(rpos[[i]])))
      #put adjacent polygons together
  rpos[[i]]<-disaggregate(rpos[[i]])#disaggregate into multiple polygons
   xyneg[[i]] <- cbind(escpt[[i]]$lon, escpt[[i]]$lat)
  rasneg[[i]] <- rasterize(xyneg[[i]], rbase, field=escpt[[i]]$negesc)
  rneg[[i]]<-rasterToPolygons(rasneg[[i]], fun=NULL, n=8, na.rm=TRUE,
      digits=5, dissolve=TRUE)
  rneg[[i]]<-unionSpatialPolygons(rneg[[i]], rep(1, length(rneg[[i]])))#
      put adjacent polygons together
  rneg[[i]]<-disaggregate(rneg[[i]])#disaggregate into multiple polygons
}
```

```
#some visual checks
#plot(rpos[[11]],col="red")

#OPTION (recommended): remove polygons with area < tau deg
#if the area of the escalation is very small, the probability of finding
    observations is close to 0 too. Therefore, we suggest to remove very
    small polygons.
#threshold area: tau
tau<-0.25#(0.25 corresponds to PRIO-grid cell area)
areaspos<-list()
maxareaspos<-list()
indexpos<-list()
areasneg<-list()
maxareasneg<-list()
indexneg<-list()#length rpos idem rneg
for(i in 1:length(rpos)){
#calculate areas of all polygons included in escalation areas for each
    year
areaspos[[i]]<- sapply(slot(rpos[[i]], "polygons"), function(x) sapply(
    slot(x,"Polygons"), slot, "area"))
maxareaspos[[i]]<-sapply(areaspos[[i]], max)#in case of possible divided
    polygons keep the one with highest area (is required for following
    steps)
indexpos[[i]] <- which(maxareaspos[[i]]>tau)#find position where area >
    tau
areasneg[[i]]<- sapply(slot(rneg[[i]], "polygons"), function(x) sapply(
    slot(x,"Polygons"), slot, "area"))
maxareasneg[[i]]<-sapply(areasneg[[i]], max)#in case of possible divided
    polygons keep the one with highest area (is required for following
    steps)
indexneg[[i]] <- which(maxareasneg[[i]]>tau)#find position where area >
    tau
}
#keep polygon with area>tau
escposnet<-list()
escnegnet<-list()
for(i in 1: length(rpos)){#length rpos idem rneg...
escposnet[[i]]<-rpos[[i]][indexpos[[i]],]#keep polygon with area>tau
escnegnet[[i]]<-rneg[[i]][indexneg[[i]],]#keep polygon with area>tau
}
```

```
#extract escalation /de-escalation values from raster to polygons
#OPTION: take max for escalation and min for de-escalation to capture
    esclation/de-escalation that occur in areas within the entire polygon
     without the requirement to occur everywhere within the polygon.
posvalue<-list()
negvalue<-list()
for(i in 1:length(escposnet)){#length rpos idem rneg...
posvalue[[i]]<-extract(raspos[[i]],escposnet[[i]],fun=max,df=TRUE)
negvalue[[i]]<-extract(rasneg[[i]],escnegnet[[i]],fun=min,df=TRUE)
}
#extract lethal events for each year from 2002-2013
let<-list()
for(i in 1:12){
  let[[i]]<-GTD1[which(GTD1$nkill > 0 & GTD1$iyear==2001+i), ]
}
#create spatial points for each year (required to count points in
    polygon)
WGS84="+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+ellps=WGS84"
plet<-list()
for(i in 1:12){
  plet[[i]]<-SpatialPoints(cbind(let[[i]]$longitude,let[[i]]$latitude))#
     create spatial points
  proj4string(plet[[i]]) = CRS(WGS84)
}
#counting observed number of lethal attacks in escalation areas in TIME
    T+1
require(GISTools)
nblet<-list()
for(i in 1:11){#TIME T+1: 11 observations of lethal attacks from 2003 to
     2013
    nblet[[i]]<-poly.counts(plet[[i+1]], escposnet[[i]])#count lethal
       attacks in hotspot
  }
#counting observed number of lethal attacks in escalation areas in TIME
    T
 nblet2<-list()
 for(i in 1:11){#TIME T-1: 11 observations of lethal attacks from 2002
     to 2012
    nblet2[[i]]<-poly.counts(plet[[i]], escposnet[[i]])#count lethal
       attacks in hotspot
```

```
    }
#compute the raio between nb lethal events from 2003 to 2013 (For ex.
    2003 indexes ratio 2003/2002)
 deltaesc<-list()
  for(i in 1:11){
  deltaesc[[i]]<-nblet[[i]]/nblet2[[i]]#compute the difference between
      proportion of lethal attack.
 }
#put escalation and de-escalation dataframe together
escaldf<-list()
for(i in 1:11){
escaldf[[i]]<-merge(posvalue[[i]],negvalue[[i]], by='ID')
colnames(escaldf[[i]]) <- c("ID","posesc","negesc")
}
#create dataframe with delta values of ratio lethal attack
deltaescdf<-list()
for(i in 1:11){
  deltaescdf[[i]]<-as.data.frame((deltaesc[[i]]))
  deltaescdf[[i]]$ID<-seq.int(nrow(deltaescdf[[i]]))#add id column
  colnames(deltaescdf[[i]]) <- c("deltaesc","ID")
}
#merge both dataframes
escalation<-list()
for(i in 1:11){
escalation[[i]]<-merge(escaldf[[i]],deltaescdf[[i]],by="ID")
}
#putting all together
escalation<-rbind(escalation[[1]],escalation[[2]],escalation[[3]],
    escalation[[4]],escalation[[5]],escalation[[6]],
            escalation[[7]],escalation[[8]],escalation[[9]],escalation
                [[10]],escalation[[11]])
#define threshold of the ratio for an escalation to be significant
escalt<-1.1
#define threshold of the ratio for an escalation to be significant
descalt<-0.9
#creating score variables with regard to prediciton of escalation / de-
    escalation
escalation$totescpred<-ifelse(escalation$posesc>0,1,0)
escalation$escpred<-ifelse(escalation$posesc>0 & escalation$deltaesc>
    escalt,1,0)
```

```
escalation$totdeescpred<-ifelse(escalation$negesc<0,1,0)
escalation$deescpred<-ifelse(escalation$negesc<0 & escalation$deltaesc<
    descalt,1,0)

#calculate the proportion of good escalation prediction
sum(escalation$escpred,na.rm=TRUE)
sum(escalation$totescpred,na.rm=TRUE)
sum(escalation$escpred,na.rm=TRUE)/sum(escalation$totescpred,na.rm=TRUE)
#calculate the proportion of good de-escalation prediction
sum(escalation$deescpred,na.rm=TRUE)
sum(escalation$totdeescpred,na.rm=TRUE)
sum(escalation$deescpred,na.rm=TRUE)/sum(escalation$totdeescpred,na.rm=
    TRUE)

#calculate sum of original escalation areas (before sample reduction
    with areas > tau)
len<-vector()
for(i in 1:11){
  len[[i]]<-length(rpos[[i]])
}
sum(len)
#calculate sum of escalation areas (after sample reduction with areas >
    tau)
lenesc<-vector()
for(i in 1:11){
  lenesc[[i]]<-length(escposnet[[i]])
}
sum(lenesc)
#end
```

D.4 Detection of Diffusion Processes

This is the code used to assess the performance of the model with regard to the diffusion and dissipation of the lethality of terrorism and real observations.

```
#SPATIO-TEMPORAL MODELLING / Bernoulli model of lthe probability of
    lethal events worldwide 2002-2013
#The code assesses the performance of the model with regard to the
    contagious diffusion of the lethality of terrorism and real
    observations.
```

```
setwd("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal")
rm(list=ls())
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/SPDE.RData")
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal/STbin_
    diffusion.RData")
require(foreign)
require(sp)
library(INLA)
library(spatstat)
library(fields)
library(maptools)
library(fields)
library(lattice)
require(rgeos)
require(raster)
require(rgdal)
require(plyr)
require(ggplot2)
#Identify (contagious) diffusion areas from 2003 to 2013
#content of dataframes: diffpos: if positive means contagious diff;
    diffneg: if negative, means retraction.

#OPTION delete buffer areas smaller than tau deg.: remove polygons with
    area < tau deg.
#threshold area: tau
tau<-0.5#(0.5 corresponds to one-half of PRIO-grid cell area)
areasdiff<-list()
maxareasdiff<-list()
indexdiff<-list()
diffdisag<-list()
for(j in 1:length(diffnetdf)){
#calculate areas of all polygons included in buffer areas for each year
areasdiff[[j]]<- gArea(diffnetdf[[j]], byid=TRUE)
indexdiff[[j]] <- which(areasdiff[[j]]>tau)#find position where area >
    tau
}
#keep polygon with area>tau
diffpolynetdf<-list()
for(j in 1: length(diffnetdf)){
```

```
diffpolynetdf[[j]]<-diffnetdf[[j]][indexdiff[[j]],]#keep polygon with
    area>tau
}
#extract lethal and non-lethal events for each year from 2002-2013
let<-list()
for(i in 1:12){
  let[[i]]<-GTD1[which(GTD1$nkill > 0 & GTD1$iyear==2001+i), ]
}
nlet<-list()
for(i in 1:12){
  nlet[[i]]<-GTD1[which(GTD1$nkill == 0 & GTD1$iyear==2001+i), ]
}
#create spatial points for each year (required to count points in
    polygon)
WGS84="+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+ellps=WGS84"
plet<-list()
pnlet<-list()
for(i in 1:12){
  plet[[i]]<-SpatialPoints(cbind(let[[i]]$longitude,let[[i]]$latitude))#
      create spatial points
  pnlet[[i]]<-SpatialPoints(cbind(nlet[[i]]$longitude,nlet[[i]]$latitude
      ))
  proj4string(plet[[i]]) = CRS(WGS84)
  proj4string(pnlet[[i]]) = CRS(WGS84)
}
#counting observed proportion of lethal and non-lethal attacks in
    escalation areas in TIME T+1
require(GISTools)
nblet<-list()
nbnlet<-list()
proplet<-list()
for(i in 1:11){#TIME T+1: 11 observations of lethal attacks from 2003 to
     2013
   nblet[[i]]<-poly.counts(plet[[i+1]], diffpolynetdf[[i]])#count lethal
       attacks in hotspot
  nbnlet[[i]]<-poly.counts(pnlet[[i+1]], diffpolynetdf[[i]])#count non
      lethal attacks in hotspot
  proplet[[i]]<-nblet[[i]]/(nblet[[i]]+nbnlet[[i]])#compute the
      proportion of lethal attack in hotspot
  }
```

```
#counting observed proportion of lethal and non-lethal attacks in
    escalation areas in TIME T-1
nblet2<-list()
nbnlet2<-list()
proplet2<-list()
for(i in 1:11){#TIME T-1: 11 observations of lethal attacks from 2002 to
     2012
  nblet2[[i]]<-poly.counts(plet[[i]], diffpolynetdf[[i]])#count lethal
      attacks in hotspot
  nbnlet2[[i]]<-poly.counts(pnlet[[i]], diffpolynetdf[[i]])#count non
      lethal attacks in hotspot
  proplet2[[i]]<-nblet2[[i]]/(nblet2[[i]]+nbnlet2[[i]])#compute the
      proportion of lethal attack in hotspot
}
#compute the ratio in prop. lethal events from 2003 to 2013 (For ex.
    2003 indexes diff. 2003-2002)
deltadiff<-list()
for(i in 1:11){
  deltadiff[[i]]<-proplet[[i]]/(proplet2[[i]])#compute the ratio between
       proportion of lethal attack
}
#create dataframe with delta values of prop. lethal attack
deltadiffdf<-list()
for(i in 1:11){
  deltadiffdf[[i]]<-as.data.frame((deltadiff[[i]]))
  deltadiffdf[[i]]$ID<-seq.int(nrow(deltadiffdf[[i]]))#add id column
  colnames(deltadiffdf[[i]]) <- c("deltadiff","ID")
}
#merge both dataframes
for(j in 1:11){
       diffpolynetdf[[j]]@data <- data.frame(diffpolynetdf[[j]]@data,
           deltadiffdf[[j]][match(diffpolynetdf[[j]]@data[,'N'],
           deltadiffdf[[j]][,'ID']),])#N is name of variable of spdf and
           ID name of variable of df which are the common variable on
           which the merge is executed.
  }
#putting all together
diffusion<-rbind(diffpolynetdf[[1]]@data,diffpolynetdf[[2]]@data,
    diffpolynetdf[[3]]@data,diffpolynetdf[[4]]@data,diffpolynetdf[[5]]
    @data,diffpolynetdf[[6]]@data,
```

```
                    diffpolynetdf[[7]]@data, diffpolynetdf[[8]]@data,
                        diffpolynetdf[[9]]@data, diffpolynetdf[[10]]@data,
                        diffpolynetdf[[11]]@data)
#define threshold of the ratio for a diffusion to be significant
difft<-1.1
#define threshold of the ratio for a dissipation to be significant
disst<-0.9
#creating score variables with regard to prediciton of diffusion / de-
    diffusion
diffusion$totdiffpred<-ifelse(diffusion$diffpos>0,1,0)
diffusion$diffpred<-ifelse(diffusion$diffpos>0 & diffusion$deltadiff>=
    difft,1,0)
diffusion$totnegdiffpred<-ifelse(diffusion$diffneg<0,1,0)
diffusion$negdiffpred<-ifelse(diffusion$diffneg<0 & diffusion$deltadiff
    <=disst,1,0)

#calculate the proportion of good diffusion prediction
sum(diffusion$diffpred,na.rm=TRUE)
sum(diffusion$totdiffpred,na.rm=TRUE)
sum(diffusion$diffpred,na.rm=TRUE)/sum(diffusion$totdiffpred,na.rm=TRUE)
#calculate the proportion of good dissipatiom (or negative diffusion)
    prediction
sum(diffusion$negdiffpred,na.rm=TRUE)
sum(diffusion$totnegdiffpred,na.rm=TRUE)
sum(diffusion$negdiffpred,na.rm=TRUE)/sum(diffusion$totnegdiffpred,na.rm
    =TRUE)
#calculate sum of original buffer areas (before sample reduction with
    areas > tau)
len<-vector()
for(i in 1:11){
  len[[i]]<-length(diffnet[[i]])
}
sum(len)
#calculate sum of buffer areas (after sample reduction with areas > tau)
lenpoly<-vector()
for(i in 1:11){
  lenpoly[[i]]<-length(diffpolynetdf[[i]])
}
sum(lenpoly)
#OBSERVED DIFFUSION:
```

```
#look at observed diffusion and keep polygons only where diffusion is
    observed.
obsdiff<-list()
keep<-list()
for(i in 1:11){
 keep[[i]] <- !is.na(diffpolynetdf[[i]]@data$deltadiff)  #index polygon
     that do not contain NA in the observed difference in the prop. of
     lethal attack in buffer areas
 obsdiff[[i]]<-diffpolynetdf[[i]][keep[[i]],]#keep the desired polygons
     (which contain no NA)
}
keep<-list()
for(i in 1:11){
  keep[[i]] <-   obsdiff[[i]]@data$deltadiff==0 #index polygon that do
      not contain 0 in the observed difference in the prop. of lethal
      attack in buffer areas
   obsdiff[[i]]<- obsdiff[[i]][!keep[[i]],]#keep the desired polygons (
       which contain no 0)
}
#keep hotspots that are related to observed diffusion
#delete first hotspot year to keep 2003 to 2013 (rather than 2002 to
    2013)
hotrpoly[[1]] <- NULL
obshot<-list()#the hotspots to be kept
negbuff<-list()
for(i in 1:11){
negbuff[[i]]<-gDifference(studyarea,obsdiff[[i]])#locate areas in the
    study area except diffusion polygons which includes original hotspots
proj4string(hotrpoly[[i]])<-CRS(proj4string(negbuff[[i]]))#projection of
     hotspots to run spatial operation
obshot[[i]]<-gIntersection(negbuff[[i]],hotrpoly[[i]])#keep hotspots
    that are within orginal hotspots (note that hotspots which do not
    lead to significant contagious diffusion or retraction are also
    included)
}
#count nb. of "true" hotspots
nbhot<-list()
for(j in 1:11){
  nbhot[[j]]<-length(disaggregate(obshot[[j]]))}
#nbhot<-nbhot[c(-1,-2)]
```

```
Reduce("+",nbhot)#nb. of hotspots 213
#count nb. of sign.neighbourhood areas (total of 53)
nbdiff<-list()
for(j in 1:11){
  nbdiff[[j]]<-length(disaggregate(obsdiff[[j]]))}
#nbdiff<-nbdiff[c(-1,-2)]
Reduce("+",nbdiff)#nb. of hotspots with diff: 60
#count nb. of neighbourhood areas with sidgnificant diffusion/
    dissipation (>0.1)

#some visual checks
# plot(hotrpoly[[2]],col="red",xlim=c(30,70),ylim=c(22,27))
# plot(studyarea,add=TRUE)
# plot(obsdiff[[2]],col="green",add=TRUE)
# plot(obshot[[2]],col="yellow",add=TRUE)

#visual plot of an example of diffusion in Iraq
# require(shape)
# plot.new()
# dev.off()
# mypath="~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal/
    diffusion/schema.png"
# png(file=mypath,width=980,height=720)
# par(cex=2.5,mar=c(4.9, 4.2,1.4, 0.1) )#mar: c(bottom,left,top,right)
# plot(hotrpoly[[6]],col="green",xlim=c(44.5,45),ylim=c(31,37),yaxt="n",
    xaxt="n",xlab="",ylab="")
# plot(obsdiff[[6]][4,],col="grey25",add=TRUE)
# plot(country,add=TRUE)
# axis(1, at=c(38,40,42,44,46,48),labels=c(38,40,42,44,46,48),cex=2, col
    .axis="black",pos=30)
# axis(2, at=c(30,31,32,33,34,35,36,37,38),labels=c(30,NA,32,NA,34,NA
    ,36,NA,38),pos=38,cex=2, col.axis="black")
# mtext(side = 2, cex=2,text = "Latitude", line = 2.5)
# mtext(side = 1, cex=2,text = "Longitude", line = 3.6,adj=0.35)
# text(40, 31, labels = "Saudi Arabia",cex =1)
# text(42.5, 33.2, labels = "Iraq",cex =1)
# text(39.1, 36.2, labels = "Syria",cex =1)
# Arrows(43.2,34.2,43.5,35, code = 2, arr.col = "red",lwd=5,arr.length
    =0.7,lcol="red",arr.type = "curved")
```

```
# Arrows(41.3,34.4,40.6,34.9, code = 2, arr.col = "red",lwd=5,arr.length
    =0.7,lcol="red",arr.type = "curved")
# Arrows(42.7,32.1,42.4,31.3, code = 2, arr.col = "red",lwd=5,arr.length
    =0.7,lcol="red",arr.type = "curved")
# Arrows(46,33.3,46.9,33.5, code = 2, arr.col = "red",lwd=5,arr.length
    =0.7,lcol="red",arr.type = "curved")
# dev.off()
save.image("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/diffusion_
    modelvsobs.RData")
#end
#load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/diffusion_modelvsobs.
    RData")
```

This is the code used to assess the performance of the model with regard to the diffusion and dissipation of the number of lethal attacks and real observations.

```
#SPATIO-TEMPORAL MODELLING / Poisson model of the number of lethal
    events worldwide 2002-2013
#The code assesses the performance of the model with regard to the
    contagious diffusion of lethal terrorism and real observations.
setwd("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson")
rm(list=ls())
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/SPDE_nbevents
    _final.RData")
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/STpoi_
    diffusion.RData")
require(foreign)
require(sp)
library(INLA)
library(spatstat)
library(fields)
library(maptools)
library(fields)
library(lattice)
require(rgeos)
require(raster)
require(rgdal)
require(plyr)
require(ggplot2)
#Identify (contagious) diffusion areas from 2003 to 2013
```

```
#content of dataframes: diffpos: if positive means contagious diff;
    diffneg: if negative, means retraction.
#extract lethal events for each year from 2002-2013
let<-list()
for(i in 1:12){
  let[[i]]<-GTD1[which(GTD1$nkill > 0 & GTD1$iyear==2001+i), ]
}
#create spatial points for each year (required to count points in
    polygon)
WGS84="+proj=longlat +datum=WGS84 +no_defs+towgs84=0,0,0+ellps=WGS84"
plet<-list()
for(i in 1:12){
  plet[[i]]<-SpatialPoints(cbind(let[[i]]$longitude,let[[i]]$latitude))#
      create spatial points
  proj4string(plet[[i]]) = CRS(WGS84)
}
#counting observed nb of lethal attacks in escalation areas in TIME T+1
require(GISTools)
diffpolynetdf<-list()
nblet<-list()
for(i in 1:11){#TIME T+1: 11 observations of lethal attacks from 2003 to
     2013
  diffpolynetdf[[i]]<-diffnetdf[[i]]
   nblet[[i]]<-poly.counts(plet[[i+1]], diffpolynetdf[[i]])#count lethal
       attacks in hotspot
  }
#counting observed nb of lethal attacks in escalation areas in TIME T-1
nblet2<-list()
for(i in 1:11){#TIME T-1: 11 observations of lethal attacks from 2002 to
     2012
  nblet2[[i]]<-poly.counts(plet[[i]], diffpolynetdf[[i]])#count lethal
      attacks in hotspot
}
#compute the ratio in nb. lethal events from 2003 to 2013 (For ex. 2003
    indexes diff. 2003-2002)
deltadiff<-list()
for(i in 1:11){
  deltadiff[[i]]<-nblet[[i]]/(nblet2[[i]])#compute the ratio between nb
      of lethal attack
}
```

```
#create dataframe with delta values of nb. lethal attack
deltadiffdf<-list()
for(i in 1:11){
  deltadiffdf[[i]]<-as.data.frame((deltadiff[[i]]))
  deltadiffdf[[i]]$ID<-seq.int(nrow(deltadiffdf[[i]]))#add id column
  colnames(deltadiffdf[[i]]) <- c("deltadiff","ID")
}
#merge both dataframes
for(j in 1:11){
      diffpolynetdf[[j]]@data <- data.frame(diffpolynetdf[[j]]@data,
          deltadiffdf[[j]][match(diffpolynetdf[[j]]@data[,'N'],
          deltadiffdf[[j]][,'ID']),])#N is name of variable of spdf and
          ID name of variable of df which are the common variable on
          which the merge is executed.
  }
#putting all together
diffusion<-rbind(diffpolynetdf[[1]]@data,diffpolynetdf[[2]]@data,
    diffpolynetdf[[3]]@data,diffpolynetdf[[4]]@data,diffpolynetdf[[5]]
    @data,diffpolynetdf[[6]]@data,
                  diffpolynetdf[[7]]@data,diffpolynetdf[[8]]@data,
                      diffpolynetdf[[9]]@data,diffpolynetdf[[10]]@data,
                      diffpolynetdf[[11]]@data)
#define threshold of the ratio for a diffusion to be significant
difft<-1.1
#define threshold of the ratio for a dissipation to be significant
disst<-0.9
#creating score variables with regard to prediciton of diffusion / de-
    diffusion
diffusion$totdiffpred<-ifelse(diffusion$diffpos>0,1,0)
diffusion$totdiffpred<-ifelse(diffusion$diffpos>0,1,0)
diffusion$diffpred<-ifelse(diffusion$diffpos>0 & diffusion$deltadiff>=
    difft,1,0)
diffusion$totnegdiffpred<-ifelse(diffusion$diffneg<0,1,0)
diffusion$negdiffpred<-ifelse(diffusion$diffneg<0 & diffusion$deltadiff
    <=disst,1,0)
#calculate the proportion of good diffusion prediction
sum(diffusion$diffpred,na.rm=TRUE)
sum(diffusion$totdiffpred,na.rm=TRUE)
sum(diffusion$diffpred,na.rm=TRUE)/sum(diffusion$totdiffpred,na.rm=TRUE)
```

```
#calculate the proportion of good dissipatiom (or negative diffusion)
    prediction
sum(diffusion$negdiffpred,na.rm=TRUE)
sum(diffusion$totnegdiffpred,na.rm=TRUE)
sum(diffusion$negdiffpred,na.rm=TRUE)/sum(diffusion$totnegdiffpred,na.rm
    =TRUE)
#calculate sum of original buffer areas (before sample reduction with
    areas > tau)
len<-vector()
for(i in 1:11){
  len[[i]]<-length(diffnet[[i]])
}
sum(len)
#calculate sum of buffer areas (after sample reduction with areas > tau)
lenpoly<-vector()
for(i in 1:11){
  lenpoly[[i]]<-length(diffpolynetdf[[i]])
}
sum(lenpoly)
#OBSERVED DIFFUSION:
#look at observed diffusion and keep polygons only where diffusion is
    observed.
obsdiff<-list()
keep<-list()
for(i in 1:11){
 keep[[i]] <- !is.na(diffpolynetdf[[i]]@data$deltadiff)  #index polygon
     that do not contain NA in the observed difference in the prop. of
     lethal attack in buffer areas
 obsdiff[[i]]<-diffpolynetdf[[i]][keep[[i]],]#keep the desired polygons
     (which contain no NA)
}
keep<-list()
for(i in 1:11){
  keep[[i]] <-   obsdiff[[i]]@data$deltadiff==0 #index polygon that do
      not contain 0 in the observed difference in the prop. of lethal
      attack in buffer areas
   obsdiff[[i]]<- obsdiff[[i]][!keep[[i]],]#keep the desired polygons (
       which contain no 0)
}
#keep hotspots that are related to observed diffusion
```

```
#delete first hotspot year to keep 2003 to 2013 (rather than 2002 to
    2013)
hotrpoly[[1]] <- NULL
obshot<-list()#the hotspots to be kept
negbuff<-list()
for(i in 1:11){
negbuff[[i]]<-gDifference(studyarea,obsdiff[[i]])#locate areas in the
    study area except diffusion polygons which includes original hotspots
proj4string(hotrpoly[[i]])<-CRS(proj4string(negbuff[[i]]))#projection of
     hotspots to run spatial operation
negbuff[[i]] <- gBuffer(negbuff[[i]], byid=TRUE, width=0)
obshot[[i]]<-gIntersection(negbuff[[i]],hotrpoly[[i]])#keep hotspots
    that are within orginal hotspots (note that hotspots which do not
    lead to significant contagious diffusion or retraction are also
    included)
}
#some visual checks
# plot(hotrpoly[[11]],col="red",xlim=c(30,70),ylim=c(22,27))
# plot(studyarea,add=TRUE)
# plot(obsdiff[[11]],col="green",add=TRUE)
# plot(obshot[[11]],col="yellow",add=TRUE)
save.image("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/
    pdiffusion_modelvsobs.RData")
#END
```

D.5 Assessment of Failed State and Hierarchical Theories

Table D.1 indicates the country of origin (*Origin*) of the centroid of the hotspots and the name of each country towards which terrorism diffuses (*Destin*). For each year, it provides the average of the Fragile States Index (FSI) of both *Origin* and *Destin*. Moreover, the worldwide annual average FSI ($FSIavg$) is given based on all observations taken in the study area within the investigated time period. Observations that indicate diffusion or dissipation absolute values ($Osbdfiff$) smaller than 0.1 have been excluded. The first column (N) provides an index used to identify each diffusion area for each year.

This is the code used to empirically assess the failed states thery with regard to diffusion and dissipation of lethality and real observations.

Table D.1 Diffusion of lethality and FSI (GTD 2005-2013)

N	Origin	Destin	Obsdiff	Diffusion	FSIorigin	FSIdestin	FSIavg	Year
1	Myanmar	China	0.8	retraction	93.4	72.3	88.7	2005
1	Myanmar	Laos	0.8	retraction	93.4	91.5	88.7	2005
1	Myanmar	Myanmar	0.8	retraction	93.4	93.4	88.7	2005
1	Myanmar	Thailand	0.8	retraction	93.4	77.4	88.7	2005
2	Russia	Georgia	0.9	retraction	83.5		88.7	2005
2	Russia	Russia	0.9	retraction	83.5	83.5	88.7	2005
3	Afghanistan	Afghanistan	0.8	retraction	99.0	99.0	88.7	2005
4	Pakistan	Pakistan	0.9	N/D	89.4	89.4	88.7	2005
4	Pakistan	India	0.9	N/D	89.4	69.5	88.7	2005
1	Sri Lanka	Georgia	0.6	retraction	92.4	82.2	70.8	2006
1	Sri Lanka	Russia	0.6	retraction	92.4	87.1	70.8	2006
2	Russia	Afghanistan	1.7	diffusion	87.1	99.8	70.8	2006
2	Russia	India	1.7	diffusion	87.1	70.4	70.8	2006
2	Russia	Iran	1.7	diffusion	87.1	84.0	70.8	2006
2	Russia	Pakistan	1.7	diffusion	87.1	103.1	70.8	2006
2	Russia	Turkmenistan	1.7	diffusion	87.1	86.1	70.8	2006
3	Afghanistan	Sri Lanka	1.6	diffusion	99.8	92.4	70.8	2006
1	Philippines	Uganda	0.5	retraction	83.2	96.4	70.6	2007
2	Uganda	Kyrgyzstan	1.0	N/D	96.4	88.2	70.6	2007
2	Uganda	Kazakhstan	1.0	N/D	96.4	72.3	70.6	2007
2	Uganda	Tajikistan	1.0	N/D	96.4	88.7	70.6	2007
2	Uganda	Uzbekistan	1.0	N/D	96.4	93.5	70.6	2007
3	Uzbekistan	Georgia	1.0	N/D	93.5	82.3	70.6	2007
3	Uzbekistan	Russia	1.0	N/D	93.5	81.2	70.6	2007
4	Russia	Afghanistan	1.2	diffusion	81.2	102.3	70.6	2007
4	Russia	India	1.2	diffusion	81.2	70.8	70.6	2007
4	Russia	Iran	1.2	diffusion	81.2	82.8	70.6	2007
4	Russia	Pakistan	1.2	diffusion	81.2	100.1	70.6	2007
4	Russia	Turkmenistan	1.2	diffusion	81.2	87.5	70.6	2007
5	Afghanistan	Philippines	0.9	retraction	102.3	83.2	70.6	2007
1	Afghanistan	Afghanistan	0.9	N/D	105.4	105.4	70.9	2008
1	Afghanistan	India	0.9	N/D	105.4	72.9	70.9	2008
1	Afghanistan	Iran	0.9	N/D	105.4	85.7	70.9	2008
1	Afghanistan	Pakistan	0.9	N/D	105.4	103.8	70.9	2008
1	Afghanistan	Turkmenistan	0.9	N/D	105.4	86.2	70.9	2008
2	Thailand	Malaysia	0.4	retraction	75.6	67.2	70.9	2008
2	Thailand	Thailand	0.4	retraction	75.6	75.6	70.9	2008
1	Afghanistan	Afghanistan	1.3	diffusion	108.2	108.2	72.1	2009
1	Afghanistan	India	1.3	diffusion	108.2	77.8	72.1	2009
1	Afghanistan	Iran	1.3	diffusion	108.2	90.0	72.1	2009
1	Afghanistan	Pakistan	1.3	diffusion	108.2	104.1	72.1	2009
1	Afghanistan	Tajikistan	1.3	diffusion	108.2	90.3	72.1	2009
1	Afghanistan	Turkmenistan	1.3	diffusion	108.2	84.3	72.1	2009
1	DRC	Burundi	0.5	retraction	109.9	96.7	71.9	2010
1	DRC	Uganda	0.5	retraction	109.9	97.5	71.9	2010
1	DRC	DRC	0.5	retraction	109.9	109.9	71.9	2010
1	DRC	Rwanda	0.5	retraction	109.9	88.7	71.9	2010
1	DRC	Tanzania	0.5	retraction	109.9	81.2	71.9	2010
2	Pakistan	Afghanistan	0.7	retraction	102.5	109.3	71.9	2010
2	Pakistan	Pakistan	0.7	retraction	102.5	102.5	71.9	2010
2	Pakistan	India	0.7	retraction	102.5	79.2	71.9	2010
2	Pakistan	Iran	0.7	retraction	102.5	92.2	71.9	2010
3	Thailand	Thailand	1.2	diffusion	78.8	78.8	71.9	2010
3	Thailand	Malaysia	1.2	diffusion	78.8	69.2	71.9	2010
1	DRC	Burundi	1.0	N/D	108.2	98.6	71.1	2011
1	DRC	DRC	1.0	N/D	108.2	108.2	71.1	2011
1	DRC	Uganda	1.0	N/D	108.2	96.3	71.1	2011
1	DRC	Rwanda	1.0	N/D	108.2	91.0	71.1	2011
1	DRC	Tanzania	1.0	N/D	108.2	81.3	71.1	2011
2	Sri Lanka	India	Inf	diffusion	93.1	79.3	71.1	2011
2	Sri Lanka	Bangladesh	Inf	diffusion	93.1	94.4	71.1	2011
3	Philippines	Afghanistan	1.0	N/D	85.0	107.5	71.1	2011
3	Philippines	India	1.0	N/D	85.0	79.3	71.1	2011
3	Philippines	Pakistan	1.0	N/D	85.0	102.3	71.1	2011
4	Philippines	Algeria	0.5	retraction	85.0	78.0	71.1	2011
4	Philippines	Tunisia	0.5	retraction	85.0	70.1	71.1	2011
5	India	Iran	1.1	diffusion	79.3	90.2	71.1	2011
5	India	Turkey	1.1	diffusion	79.3	71.5	71.1	2011
5	India	Iraq	1.1	diffusion	79.3	104.8	71.1	2011
5	India	Syria	1.1	diffusion	79.3	85.9	71.1	2011
5	India	Saudi Arabia	1.1	diffusion	79.3	75.2	71.1	2011
6	Pakistan	Malaysia	1.0	N/D	102.3	68.7	71.1	2011
6	Pakistan	Thailand	1.0	N/D	102.3	78.3	71.1	2011
7	Algeria	Sri Lanka	0.8	retraction	78.0	93.1	71.1	2011

```
#SPATIO-TEMPORAL MODELLING / binomial model of lethal events worldwide
    2002-2013
#The code assesses theories that explain the diffusion of lethal
    terrorism.
setwd("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal")
rm(list=ls())
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/diffusion_modelvsobs.
    RData")
require(foreign)
require(sp)
library(INLA)
library(spatstat)
library(fields)
library(maptools)
library(fields)
library(lattice)
require(rgeos)
require(raster)
require(rgdal)
require(plyr)
require(ggplot2)
#work on country maps
#delete unmatched countries
country<-country[!(country@data$admin=="Falkland Islands"),]
country<-country[!(country@data$admin=="French Southern and Antarctic
    Lands"),]
country<-country[!(country@data$admin=="Greenland"),]
country<-country[!(country@data$admin=="Kosovo"),]
country<-country[!(country@data$admin=="New Caledonia"),]
country<-country[!(country@data$admin=="Northern Cyprus"),]
country<-country[!(country@data$admin=="Palestine"),]
country<-country[!(country@data$admin=="Puerto Rico"),]
country<-country[!(country@data$admin=="Vanuatu"),]
country<-country[!(country@data$admin=="Taiwan"),]
country<-country[!(country@data$admin=="Somaliland"),]
country<-country[!(country@data$admin=="Western Sahara"),]
country<-country[!(country@data$admin=="Antigua and Barbuda"),]
country<-country[!(country@data$admin=="Antigua & Barbuda"),]
country<-country[!(country@data$admin=="Samoa"),]
country<-country[!(country@data$admin=="Sao Tome and Principe"),]
```

```
country<-country[!(country@data$admin=="Sao Tome"),]
country<-country[!(country@data$admin=="Micronesia"),]
country<-country[!(country@data$admin=="Comoros"),]
country<-country[!(country@data$admin=="Barbados"),]
country<-country[!(country@data$admin=="Maldives"),]
country<-country[!(country@data$admin=="Bhutan"),]
country<-country[!(country@data$admin=="Seychelles"),]
country<-country[!(country@data$admin=="Mauritius"),]
country<-country[!(country@data$admin=="Malta"),]
country<-country[!(country@data$admin=="South Sudan"),]
country<-country[!(country@data$admin=="Grenada"),]
country<-country[!(country@data$admin=="Cape Verde"),]
country<-country[!(country@data$admin=="Bahrain"),]
country<-country[!(country@data$admin=="Singapore"),]
country@data$admin<-droplevels(country@data$admin)
#create 9 copies (2005-2013) of country polygons to add temporal
    dimension and name them states[[i]]
states<-list()
for(i in 1:9){
states[[i]]<-country
}
#create list of spatial polygon df with country map by adding failed
    state index (FSI) variable
#2005 (separated from the rest) download 23.6.15: http://fsi.
    fundforpeace.org/rankings-2005-sortable
FSI1<- read.csv("C:/Users/apython/Documents/Andre/Education/StAndrews/
    PhD/PhD_R/FSI/FSI2005.csv",header = TRUE)
FSI1<-subset(FSI1, select=c("Country","Total"))
FSI<-list()#2005-2013
for(i in 1:1){
  FSI[[i]]<-FSI1
  for(i in 2:9){
FSI[[i]]<- read.csv(paste("C:/Users/apython/Documents/Andre/Education/
    StAndrews/PhD/PhD_R/FSI/FSI",2004+i,".csv",sep=""),header = TRUE)
FSI[[i]]<-subset(FSI[[i]], select=c(paste("Failed.States.Index.",2004+i,
    sep=""),"Total"))
names(FSI[[i]])<-c("Country","Total")
  }}
for(i in 1:9){
```

```
levels(FSI[[i]]$Country) <- c(levels(FSI[[i]]$Country), c("Ivory Coast",
    "Democratic Republic of the Congo","Central African Republic",
   "Bosnia and Herzegovina","Serbia and Montenegro","Guinea Bissau","
       East Timor","Republic of Congo","Israel","Republic of Serbia",
   "Trinidad and Tobago","United States of America" ,"The Bahamas","
       United Republic of Tanzania","Bosnia"))
FSI[[i]]$Country[FSI[[i]]$Country=="Cote d'Ivoire"] <- "Ivory Coast"
FSI[[i]]$Country[FSI[[i]]$Country=="Congo (D. R.)"] <- "Democratic
    Republic of the Congo"
FSI[[i]]$Country[FSI[[i]]$Country=="Congo, D.R."] <- "Democratic
    Republic of the Congo"
FSI[[i]]$Country[FSI[[i]]$Country=="Central African Rep."] <- "Central
    African Republic"
FSI[[i]]$Country[FSI[[i]]$Country=="Bosnia & Herz."] <- "Bosnia and
    Herzegovina"
FSI[[i]]$Country[FSI[[i]]$Country=="Bosnia"] <- "Bosnia and Herzegovina"
FSI[[i]]$Country[FSI[[i]]$Country=="Serbia & Mont."] <- "Serbia and
    Montenegro"
FSI[[i]]$Country[FSI[[i]]$Country=="Serbia and Montenegro"] <- "Serbia
    and Montenegro"
FSI[[i]]$Country[FSI[[i]]$Country=="Serbia/Kosovo"] <- "Serbia and
    Montenegro"
FSI[[i]]$Country[FSI[[i]]$Country=="Guinea-Bissau"] <- "Guinea Bissau"
FSI[[i]]$Country[FSI[[i]]$Country=="Timor-Leste"] <- "East Timor"
FSI[[i]]$Country[FSI[[i]]$Country=="Congo (Republic)"] <- "Republic of
    Congo"
FSI[[i]]$Country[FSI[[i]]$Country=="Congo, Republic"] <- "Republic of
    Congo"
FSI[[i]]$Country[FSI[[i]]$Country=="Israel/West Bank"] <- "Israel"
FSI[[i]]$Country[FSI[[i]]$Country=="Serbia"] <- "Republic of Serbia"
FSI[[i]]$Country[FSI[[i]]$Country=="Trinidad"] <- "Trinidad and Tobago"
FSI[[i]]$Country[FSI[[i]]$Country=="Trinadad"] <- "Trinidad and Tobago"
FSI[[i]]$Country[FSI[[i]]$Country=="United States"] <- "United States of
    America"
FSI[[i]]$Country[FSI[[i]]$Country=="Bahamas"] <- "The Bahamas"
FSI[[i]]$Country[FSI[[i]]$Country=="Tanzania"] <- "United Republic of
    Tanzania"
FSI[[i]]$Country[FSI[[i]]$Country=="Dijbouti"] <- "Djibouti"
}
#delete data that cannot be merged
```

```
for(i in 1:9){
#FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Djibouti"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Falkland Islands"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="French Southern and Antarctic
    Lands"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Greenland"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Kosovo"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="New Caledonia"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Northern Cyprus"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Palestine"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Puerto Rico"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Vanuatu"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Taiwan"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Somaliland"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Western Sahara"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Antigua and Barbuda"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Antigua & Barbuda"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="South Sudan"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Samoa"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Sao Tome and Principe"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Sao Tome"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Barbados"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Micronesia"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Comoros"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Barbados"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Maldives"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Bhutan"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Mauritius"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Malta"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Seychelles"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="South Sudan"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Grenada"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Singapore"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Bahrain"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Cape Verde"),]
}
for(i in 1:9){
  FSI[[i]]$Country<-droplevels(FSI[[i]]$Country)
}
#merge data of FSI with states
```

```
countries<-list()
FSIstate<-list()
for(j in 1:9){
  countries[[j]]<-as.data.frame(states[[j]]@data$admin)
  names(countries[[j]])<-"Country"
FSIstate[[j]]<-merge(countries[[j]],FSI[[j]],by="Country",all.x=TRUE)
FSIstate[[j]]<-FSIstate[[j]][order(FSIstate[[j]]$Country),]#order for
    proper matching
states[[j]]<-states[[j]][order(states[[j]]$admin),]#order for proper
    matching
 }
#check if country order is identical to destination df. (should be TRUE
    everywhere)
for(j in 1:9){
  check<-identical(FSIstate[[j]]$Country,states[[j]]$admin)}
check#the value should be TRUE
#put FSI values into country map (states) for each year from 2005 to
    2013
for(j in 1:9){
  states[[j]]@data <- data.frame(states[[j]]@data, FSIstate[[j]][match(
      states[[j]]@data[,'admin'], FSIstate[[j]][,'Country']),])#
      sovereignt is name of variable of spdf and Country name of variable
       of df which are the common variable on which the merge is executed
      .
}
#plot annual observed diffusion
minFSindex<-list()
maxFSindex<-list()
for(j in 1:9){
minFSindex[[j]]<-min(states[[j]]@data$Total,na.rm=TRUE)
maxFSindex[[j]]<-max(states[[j]]@data$Total,na.rm=TRUE)
}
minFSI<-min(unlist(minFSindex))
maxFSI<-max(unlist(maxFSindex))
#minFSI<-0
#maxFSI<-130
brk<-seq(minFSI,maxFSI,by=20)
n<-length(brk)-1
cols <- c("grey20","grey35","grey50","grey65","grey80")
#cols <-Llinpal(n)
```

```
#define threshold of the ratio for a diffusion to be significant
difft<-1.1
#define threshold of the ratio for a dissipation to be significant
disst<-0.9
require(Hmisc)
plot.new()
dev.off()
for(j in 1:9)#FSI indices in country are from 2005 to 2013 but the
    observed diffusion has been computed from 2003 to 2013
 {
  mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD
      /revision/revisedPhDthesis/Chapter6/Figs/Raster/obsdiffusion",j,".
      png",sep="")
  png(file=mypath,width=1440,height=720)
  par(cex=2.9,mar=c(3.5, 1.5, 1.2, 0.1) )#mar: c(bottom,left,top,right)
  plot(states[[j]],col=cols[findInterval(states[[j]]@data$Total,brk,all.
      inside=TRUE)],xlim=c(-180, 180), ylim=c(-60, 85),yaxt="n",xaxt="n",
      xlab="",ylab=""
       ,cex.main=1  ,main="")
  plot(obsdiff[[j+2]],border=NA,col=ifelse(obsdiff[[j+2]]@data$deltadiff
      >difft,"red",ifelse(obsdiff[[j+2]]@data$deltadiff<disst,"blue",NA))
      ,add=TRUE)
  plot(obshot[[j+2]],col=c("yellow"),add=TRUE)
    axis(2, at=c(-60,-40,-20,0,20,40,60,80),labels=c(-60,NA,-20,NA,20,NA
        ,60,NA), cex = 1.4,pos=-180,col.axis="black")
    axis(1, at=c(-180,-150,-100,-50,0,50,100,150,180),pos=-60,labels=c(
        NA,-150,-100,-50,0,50,100,150,NA), cex = 1.4,col.axis="black")
      mtext(side = 1, cex=2.9,text = "Longitude", line = 1.3)
       #mtext(side = 1, cex=2.9,text = 2004+j, line = 2.5)
  title(main=2004+j, line = 0.0,cex.main=1.4)
    mtext(side = 2, cex=2.9,text = "Latitude", line = -0.6)
    legend(list(x = -175,y = 37),title="FSI",inset=.12,cex=1,bty="n",
        title.adj = 0.2,
   c("17-37","37-57","57-77","77-97","97-115"),fill=c("grey20","grey35",
      "grey50","grey65","grey80"))
  legend(list(x = -55,y = -40),cex=1,bty="n",ncol=3,text.width=c
      (0,34,34),
          c("hotspot","diffusion","dissipation"),fill=c("yellow","red","
              blue"))
    dev.off()
```

```
}
#other plot for check
#spplot(states[[7]], zcol = "Total")
#count nb. of hotspots
nbhot<-list()
for(j in 1:9){
  nbhot[[j+2]]<-length(disaggregate(diffpolynetdf[[j+2]]))}
nbhot<-nbhot[c(-1,-2)]
Reduce("+",nbhot)#nb. of diff areas

#count nb. of sign.neighbourhood areas (total of 53)
nbdiff<-list()
for(j in 1:9){
  nbdiff[[j+2]]<-length(disaggregate(obsdiff[[j+2]]))}
nbdiff<-nbdiff[c(-1,-2)]
Reduce("+",nbdiff)#nb. of diff areas
#count nb. of neighbourhood areas with sidgnificant diffusion/
    dissipation (>0.1)


#summary(hotnet)                                    #
#delete hotnet[[1]] (year 2003): has no element
#hotnet[[1]]<-NULL
########CAREFUL CHECK WITH correspondance year hotspot and year GTD
#CAREFUL WITH extraction of years (since hotspot have missing years
    (2002,2003))
#extract lethal and non-lethal events for each year from 2003-2013
let<-list()
for(i in 1:11){
  let[[i]]<-GTD1[which(GTD1$nkill > 0 & GTD1$iyear==2002+i), ]
}

#create spatial points for each year (required to count points in
    polygon)
WGS84="+proj=longlat␣+datum=WGS84␣+no_defs+towgs84=0,0,0+ellps=WGS84"
plet<-list()
for(i in 1:11){
  plet[[i]]<-SpatialPoints(cbind(let[[i]]$longitude,let[[i]]$latitude))#
     create spatial points
  proj4string(plet[[i]]) = CRS(WGS84)
```

```
}

#counting observed propoortion of lethal attacks in hotspots from the
    model
require(GISTools)
nblet<-list()
for(i in 1:11){
  nblet[[i]]<-poly.counts(plet[[i]], hotnet[[i]])#count lethal attacks
     in hotspot
}
#calculate the proportion of hotspots that have number of lethal event
    >= H.
nblet<-unlist(nblet)
sum(nblet >= H,na.rm=TRUE)#number of hotspots that contain >= H of
    lethal event.
length(nblet)#number of hotspots
sum((nblet >= H)/length(nblet),na.rm=TRUE)#proportion of hotspots that
    contain >= H of lethal event.
summary(nblet,na.rm=TRUE)
sd(nblet,na.rm=TRUE)


#find origin /centroid of diffusion areas
centdiff<-list()
origin<-list()
for(j in 1:9){
obsdiff[[j]]<-spTransform(obsdiff[[j]],CRS("+init=epsg:3857"))#diffusion
     polygon
states[[j]]<-spTransform(states[[j]],CRS("+init=epsg:3857"))#country
    polygon
centdiff[[j]]<-gCentroid(obsdiff[[j]],byid=TRUE,id=obsdiff[[j]]@data$N)#
    centroid of diffusion areas
#extract country name related to the position
origin[[j]]<-over(centdiff[[j]],states[[j]])
origin[[j]]<-origin[[j]][c(64,65)]
colnames(origin[[j]])<-c("origin","FSIorigin")
origin[[j]]$ID<-rownames(origin[[j]])
#join attributes of hotspot origin to diffusion polygons
obsdiff[[j]]@data = data.frame(obsdiff[[j]]@data, origin[[j]][match(
    obsdiff[[j]]@data[,"N"], origin[[j]][,"ID"]),])
```

```
}
# put states as dataframe
countrydf<-list()
for(j in 1:9){
  countrydf[[j]]<-as.data.frame(states[[j]]@data$Country)#all world
      country
  countrydf[[j]]$ID<-rownames(countrydf[[j]])#countries and their ID
  colnames(countrydf[[j]])<-c("country","ID")
}
#find countries intersecting with diffusion areas (could be simplified?)
step1<-list()
diffstates1<-list()
  for(i in 1:length(obsdiff[[1]])){#from 1 to the numb of diffusion
      areas in each year
  step1[[i]]<-as.data.frame(states[[1]][obsdiff[[1]][i,],])
  diffstates1[[i]]<-as.data.frame(step1[[i]]$Country)
  diffstates1[[i]]$FSIdestin<-step1[[i]]$Total
  diffstates1[[i]]$N<-as.integer(i)#should put identical numbers: 1 for
      first elements of the list, 2 for second elements of the list,...
      for further merging
colnames(diffstates1[[i]])<-c("destin","FSIdestin","N")
}
step2<-list()
diffstates2<-list()
for(i in 1:length(obsdiff[[2]])){#from 2 to the numb of diffusion areas
    in each year
  step2[[i]]<-as.data.frame(states[[2]][obsdiff[[2]][i,],])
  diffstates2[[i]]<-as.data.frame(step2[[i]]$Country)
  diffstates2[[i]]$FSIdestin<-step2[[i]]$Total
  diffstates2[[i]]$N<-as.integer(i)#should put identical numbers: 1 for
      first elements of the list, 2 for second elements of the list,...
      for further merging
  colnames(diffstates2[[i]])<-c("destin","FSIdestin","N")
 }
step3<-list()
diffstates3<-list()
for(i in 1:length(obsdiff[[3]])){#from 3 to the numb of diffusion areas
    in each year
  step3[[i]]<-as.data.frame(states[[3]][obsdiff[[3]][i,],])
  diffstates3[[i]]<-as.data.frame(step3[[i]]$Country)
```

```
  diffstates3[[i]]$FSIdestin<-step3[[i]]$Total
  diffstates3[[i]]$N<-as.integer(i)#should put identical numbers: 1 for
     first elements of the list, 2 for second elements of the list,...
     for further merging
  colnames(diffstates3[[i]])<-c("destin","FSIdestin","N")
}
step4<-list()
diffstates4<-list()
for(i in 1:length(obsdiff[[4]])){#from 4 to the numb of diffusion areas
   in each year
  step4[[i]]<-as.data.frame(states[[4]][obsdiff[[4]][i,],])
  diffstates4[[i]]<-as.data.frame(step4[[i]]$Country)
  diffstates4[[i]]$FSIdestin<-step4[[i]]$Total
  diffstates4[[i]]$N<-as.integer(i)#should put identical numbers: 1 for
     first elements of the list, 2 for second elements of the list,...
     for further merging
  colnames(diffstates4[[i]])<-c("destin","FSIdestin","N")
  }
step5<-list()
diffstates5<-list()
for(i in 1:length(obsdiff[[5]])){#from 5 to the numb of diffusion areas
   in each year
  step5[[i]]<-as.data.frame(states[[5]][obsdiff[[5]][i,],])
  diffstates5[[i]]<-as.data.frame(step5[[i]]$Country)
  diffstates5[[i]]$FSIdestin<-step5[[i]]$Total
  diffstates5[[i]]$N<-as.integer(i)#should put identical numbers: 1 for
     first elements of the list, 2 for second elements of the list,...
     for further merging
  colnames(diffstates5[[i]])<-c("destin","FSIdestin","N")
  }
step6<-list()
diffstates6<-list()
for(i in 1:length(obsdiff[[6]])){#from 6 to the numb of diffusion areas
   in each year
  step6[[i]]<-as.data.frame(states[[6]][obsdiff[[6]][i,],])
  diffstates6[[i]]<-as.data.frame(step6[[i]]$Country)
  diffstates6[[i]]$FSIdestin<-step6[[i]]$Total
  diffstates6[[i]]$N<-as.integer(i)#should put identical numbers: 1 for
     first elements of the list, 2 for second elements of the list,...
     for further merging
```

```
  colnames(diffstates6[[i]])<-c("destin","FSIdestin","N")
  }
step7<-list()
diffstates7<-list()
for(i in 1:length(obsdiff[[7]])){#from 7 to the numb of diffusion areas
    in each year
  step7[[i]]<-as.data.frame(states[[7]][obsdiff[[7]][i,],])
  diffstates7[[i]]<-as.data.frame(step7[[i]]$Country)
  diffstates7[[i]]$FSIdestin<-step7[[i]]$Total
  diffstates7[[i]]$N<-as.integer(i)#should put identical numbers: 1 for
      first elements of the list, 2 for second elements of the list,...
      for further merging
  colnames(diffstates7[[i]])<-c("destin","FSIdestin","N")
  }
step8<-list()
diffstates8<-list()
for(i in 1:length(obsdiff[[8]])){#from 8 to the numb of diffusion areas
    in each year
  step8[[i]]<-as.data.frame(states[[8]][obsdiff[[8]][i,],])
  diffstates8[[i]]<-as.data.frame(step8[[i]]$Country)
  diffstates8[[i]]$FSIdestin<-step8[[i]]$Total
  diffstates8[[i]]$N<-as.integer(i)#should put identical numbers: 1 for
      first elements of the list, 2 for second elements of the list,...
      for further merging
  colnames(diffstates8[[i]])<-c("destin","FSIdestin","N")
}
step9<-list()
diffstates9<-list()
for(i in 1:length(obsdiff[[9]])){#from 9 to the numb of diffusion areas
    in each year
  step9[[i]]<-as.data.frame(states[[9]][obsdiff[[9]][i,],])
  diffstates9[[i]]<-as.data.frame(step9[[i]]$Country)
  diffstates9[[i]]$FSIdestin<-step9[[i]]$Total
  diffstates9[[i]]$N<-as.integer(i)#should put identical numbers: 1 for
      first elements of the list, 2 for second elements of the list,...
      for further merging
  colnames(diffstates9[[i]])<-c("destin","FSIdestin","N")
  }
#put df together into a list
```

```
diffstates<-list(diffstates1,diffstates2,diffstates3,diffstates4,
    diffstates5, diffstates6,diffstates7,diffstates8,diffstates9)
for(j in 1:9){#from 9 to the numb of diffusion areas in each year
diffstates[[j]] = Reduce(function(...) merge(..., all=T), diffstates[[j
    ]])#merge df within each year
}
#join attributes of hotspot origin to diffusion polygons
 difftable<-list()
 for(i in 1:9){#from 1 to the numb of diffusion areas in each year
 obsdiff[[i]]<-as.data.frame(obsdiff[[i]])#put as a dataframe for
     further merging
 obsdiff[[i]]$N <- as.numeric(as.factor(with(obsdiff[[i]], paste(ID, sep
     ="_"))))#the variable "N" give a unique identifier for each centroid
      of diffusion areas (required for further merging)
  difftable[[i]]<-merge(diffstates[[i]],obsdiff[[i]],by='N',all.x=TRUE)#
     merging destination states with diffusion areas and origin
 keeps <- c("N","destin","FSIdestin","diffpos","diffneg","deltadiff","
     origin","FSIorigin")
 difftable[[i]]<-difftable[[i]][keeps]#keep important variables only
 difftable[[i]]$year<-2004+i#add year of diffusion as new variable
}
#put all df together
diffusiontab<-rbind(difftable[[1]],difftable[[2]],difftable[[3]],
    difftable[[4]],difftable[[5]],
                    difftable[[6]],difftable[[7]],difftable[[8]],
                       difftable[[9]])
#define threshold of the ratio for a diffusion to be significant
difft<-1.1
#define threshold of the ratio for a dissipation to be significant
disst<-0.9
#keep important variables and produce new variables for publication in
    thesis
diffusiontab$diffusion<-ifelse(diffusiontab$deltadiff>=difft,"diffusion"
    ,ifelse(diffusiontab$deltadiff<= 0.9,"retraction","N/D"))
diffusiontab<-diffusiontab[c(-4,-5)]
diffusiontab<-diffusiontab[,c("N","origin","FSIorigin","destin","
    FSIdestin","deltadiff","diffusion","year")]
colnames(diffusiontab)<-c("N","Origin","FSIorigin","Destin","FSIdestin",
    "Obsdiff","Diffusion","Year")
```

```
diffusiontab$Origin<-as.character(diffusiontab$Origin);diffusiontab$
    Destin<-as.character(diffusiontab$Destin)
diffusiontab[diffusiontab=="Democratic␣Republic␣of␣the␣Congo"] <- "DRC"
diffusiontab[diffusiontab=="United␣Republic␣of␣Tanzania"] <- "Tanzania"
#get average FSI from original files of FSI years 2005 to 2013
FSIavg<-c(88.68,70.79,70.56,70.86,72.07,71.87,71.08,70.88,70.5)
Year<-c(2005,2006,2007,2008,2009,2010,2011,2012,2013)
FSIavg<-as.data.frame(cbind(FSIavg,Year))
#merge with df
diffusiontab<-merge(diffusiontab,FSIavg,by="Year")
#reorder df
diffusiontab<-diffusiontab[,c("N","Origin","Destin","Obsdiff","Diffusion
    ","FSIorigin","FSIdestin","FSIavg","Year")]
#rounding
diffusiontab$Year<-as.integer(diffusiontab$Year)
diffusiontab
#OPTION:KEEP ONLY DIFFUSION OR RETRACTION IF ABSOLUTE VALUE > 10%
#diffusiontab<-diffusiontab[- grep("N/D", diffusiontab$Diffusion),]
#histogram which relates the difference in FSI according to areas of
    diffusion and areas of retraction
Diffpos<-diffusiontab[diffusiontab$Obsdiff>=difft,]
Diffneg<-diffusiontab[diffusiontab$Obsdiff<=disst,]
#basic statistics
difffsistat<-aggregate(diffusiontab, by=list(diffusiontab$Diffusion,
    diffusiontab$Year), FUN=mean, na.rm=TRUE)
fsimax<-aggregate(diffusiontab, by=list(diffusiontab$Diffusion,
    diffusiontab$Year), FUN=max, na.rm=TRUE)
fsimin<-aggregate(diffusiontab, by=list(diffusiontab$Diffusion,
    diffusiontab$Year), FUN=min, na.rm=TRUE)
difffsistat<-difffsistat[c(-3,-4,-5,-7)]
fsimax<-fsimax[c(-3,-4,-5,-6,-7,-10,-11)]
names(fsimax)<-c("Group.1","Group.2","FSIoriginmax","FSIdestinmax")
fsimin<-fsimin[c(-3,-4,-5,-6,-7,-10,-11)]
names(fsimin)<-c("Group.1","Group.2","FSIoriginmin","FSIdestinmin")
difffsistat<-merge(difffsistat,fsimax,by=c("Group.1","Group.2"))
difffsistat<-merge(difffsistat,fsimin,by=c("Group.1","Group.2"))
#look at diffusion observation in origin country
diffonly<-difffsistat[difffsistat$Group.1=="diffusion",]
retraconly<-difffsistat[difffsistat$Group.1=="retraction",]
```

```
#stat. test to check if FSI in hotspots different from FSI in areas of
    contagion/retraction + comparision with FSI world avg.
wilcox.test(diffusiontab$FSIorigin, diffusiontab$FSIavg,alternative = c(
    "greater"))
wilcox.test(diffusiontab$FSIdestin, diffusiontab$FSIavg,alternative = c(
    "greater"))
wilcox.test(diffonly$FSIorigin, diffonly$FSIdestin,alternative = c("
    greater"))
wilcox.test(diffonly$FSIorigin, diffonly$FSIavg,alternative = c("greater
    "))#FSI hot with diff > world avg p=0.00029
wilcox.test(diffonly$FSIdestin, diffonly$FSIavg,alternative = c("greater
    "))
wilcox.test(retraconly$FSIorigin, retraconly$FSIdestin,alternative = c("
    greater"))
wilcox.test(retraconly$FSIorigin, retraconly$FSIavg,alternative = c("
    greater"))
wilcox.test(retraconly$FSIdestin, retraconly$FSIavg,alternative = c("
    greater"))
hotanddiff<-subset(diffusiontab, Diffusion=='diffusion')
hotandnodiff<-subset(diffusiontab, Diffusion!='diffusion')
#stat. test to check if FSI in hotspots that diffuse different from
    their diffusion areas
wilcox.test(hotanddiff$FSIorigin, hotanddiff$FSIdestin,alternative = c("
    greater"),exact=FALSE)
wilcox.test(hotanddiff$FSIorigin, hotanddiff$FSIdestin,alternative = c("
    less"),exact=FALSE)
#stat. test to check if FSI in hotspots that diffuse different from FSI
    in hotspots that do not diffuse
wilcox.test(hotanddiff$FSIorigin, hotandnodiff$FSIorigin,alternative = c
    ("greater"))
wilcox.test(hotanddiff$FSIorigin, hotandnodiff$FSIorigin,alternative = c
    ("less"))
#stat. test to check if FSI area of diffusion different from FSI in
    hotspots that do not diffuse
wilcox.test(hotanddiff$FSIdestin, hotandnodiff$FSIorigin,alternative = c
    ("less"))
wilcox.test(hotanddiff$FSIdestin, hotandnodiff$FSIorigin,alternative = c
    ("greater"))
#stat. test to check if  FSI in hotspot that do not diffuse is equal
    with FSi in its neighbourhood
```

```
wilcox.test(hotandnodiff$FSIdestin, hotandnodiff$FSIorigin,alternative =
    c("less"))
wilcox.test(hotandnodiff$FSIdestin, hotandnodiff$FSIorigin,alternative =
    c("greater"))
#table for LateX
require(xtable)
print(xtable(diffusiontab,digits=1), include.rownames=FALSE)
#look at the FSI differences between origin and neighbourhood for each
    pair of hotspot/contagious diffusion areas
#average values for each polygon in the origin and destination
#aggregate by polygon ID and by year (and by diffusion to differentiate
    diffusion from retraction) in order to have couple of origin and its
    related areas
#basic statistics
deltaFSImean<-aggregate(diffusiontab, by=list(diffusiontab$Diffusion,
    diffusiontab$Year,diffusiontab$N), FUN=mean, na.rm=TRUE)
deltaFSImax<-aggregate(diffusiontab, by=list(diffusiontab$Diffusion,
    diffusiontab$Year,diffusiontab$N), FUN=max, na.rm=TRUE)
deltaFSImin<-aggregate(diffusiontab, by=list(diffusiontab$Diffusion,
    diffusiontab$Year,diffusiontab$N), FUN=min, na.rm=TRUE)
#End

a=diffusiontab[diffusiontab$Diffusion=="diffusion",]$FSIdestin
b=diffusiontab[diffusiontab$Diffusion=="diffusion",]$FSIorigin
c=diffusiontab[diffusiontab$Diffusion=="N/D",]$FSIorigin

# Make a list of these 2 vectors
C=list(a,b,c)
names(C)=c(paste("Diffusion\n " , "area" , sep=""), paste("Hotspot\n " ,
    "with diffusion" , sep=""), paste("Hotspot\n " , "without diffusion"
    , sep=""))
#I change the mgp argument to avoid the text overlays the x axis

plot.new()
mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    revision/revisedPhDthesis/Chapter6/Figs/Raster/FSIschema.png",sep="")
png(file=mypath,width=900,height=720)
par(cex=2.4,mar=c(3, 4, 0.2,1),mgp=c(3,2,0))#mar: c(bottom,left,top,
    right)
boxplot(C, main=NA,outcex=1,
```

```
        xlab=NA, ylab="FSI",ylim=c(60,120))
abline(h = mean(diffusiontab$FSIavg,na.rm=T), col = "black",lwd=2,lty=2)
text(x=3.0, y = 70, labels = "World average FSI",
     pos = NULL, offset = 0.5, vfont = NULL,
     cex = 0.75, col = NULL, font = NULL)
text(x=3.0, y = 119, labels = "Bernoulli model",
     pos = NULL, offset = 0.5, vfont = NULL,
     cex = 0.75, col = NULL, font = NULL)
dev.off();dev.off()

save.image("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/diffusion_FSI.
    RData")
#load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/diffusion_FSI.RData")
```

Table D.2 indicates the country of origin (*Origin*) of the centroid of the source of the diffusion (hotspot) and the countries towards which terrorism diffuses (*Destin*). For each year, it provides the values of the Fragile States Index (FSI) averaged within both *Origin* and *Destin* polygons. Moreover, the worldwide annual average FSI ($FSIavg$) is computed based on values taken at all locations of terrorist events worldwide during the investigated time period. Observations that indicate absolute values of diffusion or dissipation ($Osbdfiff$) smaller than 0.1 have been excluded. The first column (N) identifies each diffusion area for each year.

This is the code used to empirically assess the failed states theory with regard to diffusion and dissipation of the number of lethal terrorist events and real observations.

```
#SPATIO-TEMPORAL MODELLING / Poisson model of the number of lethal
    events worldwide 2002-2013 (X covariates)
#The code assesses the performance of the model with regard to the
    contagious diffusion of lethal terrorism and real observations.
setwd("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson")
rm(list=ls())
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/pdiffusion_
    modelvsobs.RData")
require(foreign)
require(sp)
library(INLA)
library(spatstat)
library(fields)
library(maptools)
library(fields)
```

Table D.2 Diffusion of lethal attacks and FSI (GTD 2005-2013)

1	Thailand	Myanmar	Inf	diffusion	77.4	93.4	88.7	2005
1	Thailand	China	Inf	diffusion	77.4	72.3	88.7	2005
1	Thailand	Laos	Inf	diffusion	77.4	91.5	88.7	2005
1	Thailand	Thailand	Inf	diffusion	77.4	77.4	88.7	2005
2		Pakistan	0.3	retraction		89.4	88.7	2005
2		India	0.3	retraction		69.5	88.7	2005
3	Algeria	Afghanistan	0.6	retraction	81.2	99.0	88.7	2005
3	Algeria	Pakistan	0.6	retraction	81.2	89.4	88.7	2005
4	India	Russia	0.6	retraction	69.5	83.5	88.7	2005
4	India	Georgia	0.6	retraction	69.5		88.7	2005
4	India	Azerbaijan	0.6	retraction	69.5	85.7	88.7	2005
5	Afghanistan	Syria	5.0	diffusion	99.0	91.5	88.7	2005
5	Afghanistan	Iraq	5.0	diffusion	99.0	103.2	88.7	2005
5	Afghanistan	Iran	5.0	diffusion	99.0	83.8	88.7	2005
5	Afghanistan	Turkey	5.0	diffusion	99.0	86.1	88.7	2005
6	Russia	Iran	0.7	retraction	83.5	83.8	88.7	2005
6	Russia	Kuwait	0.7	retraction	83.5		88.7	2005
6	Russia	Iraq	0.7	retraction	83.5	103.2	88.7	2005
7	Iraq	Egypt	Inf	diffusion	103.2	88.8	88.7	2005
7	Iraq	Syria	Inf	diffusion	103.2	91.5	88.7	2005
7	Iraq	Israel	Inf	diffusion	103.2		88.7	2005
7	Iraq	Jordan	Inf	diffusion	103.2		88.7	2005
7	Iraq	Lebanon	Inf	diffusion	103.2	88.9	88.7	2005
8	Iran	Algeria	2.0	diffusion	83.8	81.2	88.7	2005
1	Sri Lanka	India	1.4	diffusion	92.4	70.4	70.8	2006
1	Sri Lanka	Pakistan	1.4	diffusion	92.4	103.1	70.8	2006
2	Pakistan	Afghanistan	2.5	diffusion	103.1	99.8	70.8	2006
2	Pakistan	Iran	2.5	diffusion	103.1	84.0	70.8	2006
2	Pakistan	Turkmenistan	2.5	diffusion	103.1	86.1	70.8	2006
2	Pakistan	Pakistan	2.5	diffusion	103.1	103.1	70.8	2006
3	Iran	Azerbaijan	1.0	N/D	84.0	81.9	70.8	2006
3	Iran	Georgia	1.0	N/D	84.0	82.2	70.8	2006
3	Iran	Russia	1.0	N/D	84.0	87.1	70.8	2006
4	India	Turkey	0.8	retraction	70.4	74.4	70.8	2006
4	India	Iraq	0.8	retraction	70.4	109.0	70.8	2006
4	India	Iran	0.8	retraction	70.4	84.0	70.8	2006
4	India	Syria	0.8	retraction	70.4	88.6	70.8	2006
5	Afghanistan	Indonesia	1.3	diffusion	99.8	89.2	70.8	2006
6	Russia	Sri Lanka	0.2	retraction	87.1	92.4	70.8	2006
7	Iraq	Pakistan	2.8	diffusion	109.0	103.1	70.8	2006
8	Indonesia	Iran	Inf	diffusion	89.2	84.0	70.8	2006
8	Indonesia	Iraq	Inf	diffusion	89.2	109.0	70.8	2006
8	Indonesia	Kuwait	Inf	diffusion	89.2	60.8	70.8	2006
1	Afghanistan	Afghanistan	1.2	diffusion	102.3	102.3	70.6	2007
1	Afghanistan	Turkmenistan	1.2	diffusion	102.3	87.5	70.6	2007
1	Afghanistan	Iran	1.2	diffusion	102.3	82.8	70.6	2007
1	Afghanistan	India	1.2	diffusion	102.3	70.8	70.6	2007
1	Afghanistan	Tajikistan	1.2	diffusion	102.3	88.7	70.6	2007
1	Afghanistan	Pakistan	1.2	diffusion	102.3	100.1	70.6	2007
2	Iraq	Iran	2.5	diffusion	111.4	82.8	70.6	2007
2	Iraq	Iraq	2.5	diffusion	111.4	111.4	70.6	2007
2	Iraq	Kuwait	2.5	diffusion	111.4	62.1	70.6	2007
2	Iraq	Turkey	2.5	diffusion	111.4	74.9	70.6	2007
2	Iraq	Syria	2.5	diffusion	111.4	88.6	70.6	2007
3	Russia	Russia	7.0	diffusion	81.2	81.2	70.6	2007
3	Russia	Azerbaijan	7.0	diffusion	81.2	81.2	70.6	2007
3	Russia	Georgia	7.0	diffusion	81.2	82.3	70.6	2007
4	Indonesia	Indonesia	0.5	retraction	84.4	84.4	70.6	2007
5	Indonesia	Indonesia	1.0	N/D	84.4	84.4	70.6	2007
6	Thailand	Malaysia	Inf	diffusion	76.0	65.9	70.6	2007
6	Thailand	Thailand	Inf	diffusion	76.0	76.0	70.6	2007
1	Afghanistan	Afghanistan	1.5	diffusion	105.4	105.4	70.9	2008
1	Afghanistan	Iran	1.5	diffusion	105.4	85.7	70.9	2008
1	Afghanistan	Turkmenistan	1.5	diffusion	105.4	86.2	70.9	2008
1	Afghanistan	India	1.5	diffusion	105.4	72.9	70.9	2008
1	Afghanistan	Pakistan	1.5	diffusion	105.4	103.8	70.9	2008
1	Afghanistan	Tajikistan	1.5	diffusion	105.4	88.9	70.9	2008
2	Iraq	Kuwait	0.6	retraction	110.6	62.0	70.9	2008
2	Iraq	Iran	0.6	retraction	110.6	85.7	70.9	2008
2	Iraq	Iraq	0.6	retraction	110.6	110.6	70.9	2008
2	Iraq	Syria	0.6	retraction	110.6	90.1	70.9	2008
2	Iraq	Turkey	0.6	retraction	110.6	75.4	70.9	2008
3	Russia	Azerbaijan	0.8	retraction	79.7	81.0	70.9	2008
3	Russia	Russia	0.8	retraction	79.7	79.7	70.9	2008
3	Russia	Georgia	0.8	retraction	79.7	83.8	70.9	2008

```
library(lattice)
require(rgeos)
require(raster)
require(rgdal)
require(plyr)
require(ggplot2)
#work on country maps
#delete unmatched countries
country<-country[!(country@data$admin=="Falkland Islands"),]
country<-country[!(country@data$admin=="French Southern and Antarctic
    Lands"),]
country<-country[!(country@data$admin=="Greenland"),]
country<-country[!(country@data$admin=="Kosovo"),]
country<-country[!(country@data$admin=="New Caledonia"),]
country<-country[!(country@data$admin=="Northern Cyprus"),]
country<-country[!(country@data$admin=="Palestine"),]
country<-country[!(country@data$admin=="Puerto Rico"),]
country<-country[!(country@data$admin=="Vanuatu"),]
country<-country[!(country@data$admin=="Taiwan"),]
country<-country[!(country@data$admin=="Somaliland"),]
country<-country[!(country@data$admin=="Western Sahara"),]
country<-country[!(country@data$admin=="Antigua and Barbuda"),]
country<-country[!(country@data$admin=="Antigua & Barbuda"),]
country<-country[!(country@data$admin=="Samoa"),]
country<-country[!(country@data$admin=="Sao Tome and Principe"),]
country<-country[!(country@data$admin=="Sao Tome"),]
country<-country[!(country@data$admin=="Micronesia"),]
country<-country[!(country@data$admin=="Comoros"),]
country<-country[!(country@data$admin=="Barbados"),]
country<-country[!(country@data$admin=="Maldives"),]
country<-country[!(country@data$admin=="Bhutan"),]
country<-country[!(country@data$admin=="Seychelles"),]
country<-country[!(country@data$admin=="Mauritius"),]
country<-country[!(country@data$admin=="Malta"),]
country<-country[!(country@data$admin=="South Sudan"),]
country<-country[!(country@data$admin=="Grenada"),]
country<-country[!(country@data$admin=="Cape Verde"),]
country<-country[!(country@data$admin=="Bahrain"),]
country<-country[!(country@data$admin=="Singapore"),]
country@data$admin<-droplevels(country@data$admin)
```

```
#create 9 copies (2005-2013) of country polygons to add temporal
    dimension and name them states[[i]]
states<-list()
for(i in 1:9){
states[[i]]<-country
}
#create list of spatial polygon df with country map by adding failed
    state index (FSI) variable
#2005 (separated from the rest) download 23.6.15: http://fsi.
    fundforpeace.org/rankings-2005-sortable
FSI1<- read.csv("C:/Users/apython/Documents/Andre/Education/StAndrews/
    PhD/PhD_R/FSI/FSI2005.csv",header = TRUE)
FSI1<-subset(FSI1, select=c("Country","Total"))
FSI<-list()#2005-2013
for(i in 1:1){
  FSI[[i]]<-FSI1
  for(i in 2:9){
FSI[[i]]<- read.csv(paste("C:/Users/apython/Documents/Andre/Education/
    StAndrews/PhD/PhD_R/FSI/FSI",2004+i,".csv",sep=""),header = TRUE)
FSI[[i]]<-subset(FSI[[i]], select=c(paste("Failed.States.Index.",2004+i,
    sep=""),"Total"))
names(FSI[[i]])<-c("Country","Total")
  }}
for(i in 1:9){
levels(FSI[[i]]$Country) <- c(levels(FSI[[i]]$Country), c("Ivory␣Coast",
     "Democratic␣Republic␣of␣the␣Congo","Central␣African␣Republic",
    "Bosnia␣and␣Herzegovina","Serbia␣and␣Montenegro","Guinea␣Bissau","
        East␣Timor","Republic␣of␣Congo","Israel","Republic␣of␣Serbia",
    "Trinidad␣and␣Tobago","United␣States␣of␣America" ,"The␣Bahamas","
        United␣Republic␣of␣Tanzania","Bosnia"))
FSI[[i]]$Country[FSI[[i]]$Country=="Cote␣d'Ivoire"] <- "Ivory␣Coast"
FSI[[i]]$Country[FSI[[i]]$Country=="Congo␣(D.␣R.)"] <- "Democratic␣
    Republic␣of␣the␣Congo"
FSI[[i]]$Country[FSI[[i]]$Country=="Congo,␣D.R."] <- "Democratic␣
    Republic␣of␣the␣Congo"
FSI[[i]]$Country[FSI[[i]]$Country=="Central␣African␣Rep."] <- "Central␣
    African␣Republic"
FSI[[i]]$Country[FSI[[i]]$Country=="Bosnia␣&␣Herz."] <- "Bosnia␣and␣
    Herzegovina"
FSI[[i]]$Country[FSI[[i]]$Country=="Bosnia"] <- "Bosnia␣and␣Herzegovina"
```

```
FSI[[i]]$Country[FSI[[i]]$Country=="Serbia & Mont."] <- "Serbia and
    Montenegro"
FSI[[i]]$Country[FSI[[i]]$Country=="Serbia and Montenegro"] <- "Serbia
    and Montenegro"
FSI[[i]]$Country[FSI[[i]]$Country=="Serbia/Kosovo"] <- "Serbia and
    Montenegro"
FSI[[i]]$Country[FSI[[i]]$Country=="Guinea-Bissau"] <- "Guinea Bissau"
FSI[[i]]$Country[FSI[[i]]$Country=="Timor-Leste"] <- "East Timor"
FSI[[i]]$Country[FSI[[i]]$Country=="Congo (Republic)"] <- "Republic of
    Congo"
FSI[[i]]$Country[FSI[[i]]$Country=="Congo, Republic"] <- "Republic of
    Congo"
FSI[[i]]$Country[FSI[[i]]$Country=="Israel/West Bank"] <- "Israel"
FSI[[i]]$Country[FSI[[i]]$Country=="Serbia"] <- "Republic of Serbia"
FSI[[i]]$Country[FSI[[i]]$Country=="Trinidad"] <- "Trinidad and Tobago"
FSI[[i]]$Country[FSI[[i]]$Country=="Trinadad"] <- "Trinidad and Tobago"
FSI[[i]]$Country[FSI[[i]]$Country=="United States"] <- "United States of
    America"
FSI[[i]]$Country[FSI[[i]]$Country=="Bahamas"] <- "The Bahamas"
FSI[[i]]$Country[FSI[[i]]$Country=="Tanzania"] <- "United Republic of
    Tanzania"
FSI[[i]]$Country[FSI[[i]]$Country=="Dijbouti"] <- "Djibouti"
}
#delete data that cannot be merged
for(i in 1:9){
#FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Djibouti"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Falkland Islands"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="French Southern and Antarctic
    Lands"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Greenland"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Kosovo"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="New Caledonia"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Northern Cyprus"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Palestine"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Puerto Rico"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Vanuatu"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Taiwan"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Somaliland"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Western Sahara"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Antigua and Barbuda"),]
```

```
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Antigua␣&␣Barbuda"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="South␣Sudan"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Samoa"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Sao␣Tome␣and␣Principe"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Sao␣Tome"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Barbados"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Micronesia"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Comoros"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Barbados"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Maldives"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Bhutan"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Mauritius"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Malta"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Seychelles"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="South␣Sudan"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Grenada"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Singapore"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Bahrain"),]
FSI[[i]]<-FSI[[i]][!(FSI[[i]]$Country=="Cape␣Verde"),]
}
for(i in 1:9){
  FSI[[i]]$Country<-droplevels(FSI[[i]]$Country)
}
#merge data of FSI with states
countries<-list()
FSIstate<-list()
for(j in 1:9){
  countries[[j]]<-as.data.frame(states[[j]]@data$admin)
  names(countries[[j]])<-"Country"
FSIstate[[j]]<-merge(countries[[j]],FSI[[j]],by="Country",all.x=TRUE)
FSIstate[[j]]<-FSIstate[[j]][order(FSIstate[[j]]$Country),]#order for
    proper matching
states[[j]]<-states[[j]][order(states[[j]]$admin),]#order for proper
    matching
 }
#check if country order is identical to destination df. (should be TRUE
    everywhere)
for(j in 1:9){
  check<-identical(FSIstate[[j]]$Country,states[[j]]$admin)}
check#the value should be TRUE
```

```
#put FSI values into country map (states) for each year from 2005 to
    2013
for(j in 1:9){
  states[[j]]@data <- data.frame(states[[j]]@data, FSIstate[[j]][match(
      states[[j]]@data[,'admin'], FSIstate[[j]][,'Country']),])#
      sovereignt is name of variable of spdf and Country name of variable
       of df which are the common variable on which the merge is executed
      .
}
#plot annual observed diffusion plot.new()
minFSindex<-list()
maxFSindex<-list()
for(j in 1:9){
minFSindex[[j]]<-min(states[[j]]@data$Total,na.rm=TRUE)
maxFSindex[[j]]<-max(states[[j]]@data$Total,na.rm=TRUE)
}
minFSI<-min(unlist(minFSindex))
maxFSI<-max(unlist(maxFSindex))
#minFSI<-0
#maxFSI<-130
brk<-seq(minFSI,maxFSI,by=20)
n<-length(brk)-1
cols <- c("grey20","grey35","grey50","grey65","grey80")
#cols <-Llinpal(n)
#define threshold of the ratio for a diffusion to be significant
difft<-1.1
#define threshold of the ratio for a dissipation to be significant
disst<-0.9
require(Hmisc)
plot.new()
dev.off()
for(j in 1:9)#FSI indices in country are from 2005 to 2013 but the
    observed diffusion has been computed from 2003 to 2013
{
  mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD
      /revision/revisedPhDthesis/Chapter6/Figs/Raster/pobsdiffusion",j,".
      png",sep="")
  png(file=mypath,width=1440,height=720)
  par(cex=2.9,mar=c(3.5, 1.5, 1.2, 0.1) )#mar: c(bottom,left,top,right)
```

```
  plot(states[[j]],col=cols[findInterval(states[[j]]@data$Total,brk,all.
      inside=TRUE)],xlim=c(-180, 180), ylim=c(-60, 85),yaxt="n",xaxt="n",
      xlab="",ylab=""
        ,cex.main=1 ,main="")
  plot(obsdiff[[j+2]],border=NA,col=ifelse(obsdiff[[j+2]]@data$deltadiff
      >difft,"red",ifelse(obsdiff[[j+2]]@data$deltadiff<disst,"blue",NA))
      ,add=TRUE)
  plot(obshot[[j+2]],col=c("yellow"),add=TRUE)
  axis(2, at=c(-60,-40,-20,0,20,40,60,80),labels=c(-60,NA,-20,NA,20,NA
      ,60,NA), cex = 1.4,pos=-180,col.axis="black")
  axis(1, at=c(-180,-150,-100,-50,0,50,100,150,180),pos=-60,labels=c(NA
      ,-150,-100,-50,0,50,100,150,NA), cex = 1.4,col.axis="black")
  mtext(side = 1, cex=2.9,text = "Longitude", line = 1.3)
  #mtext(side = 1, cex=2.9,text = 2004+j, line = 2.5)
  title(main=2004+j, line = 0.0,cex.main=1.4)
  mtext(side = 2, cex=2.9,text = "Latitude", line = -0.6)
  legend(list(x = -175,y = 37),title="FSI",inset=.12,cex=1,bty="n",title
      .adj = 0.2,
          c("17-37","37-57","57-77","77-97","97-115"),fill=c("grey20","
              grey35","grey50","grey65","grey80"))
  legend(list(x = -55,y = -40),cex=1,bty="n",ncol=3,text.width=c
      (0,34,34),
          c("hotspot","diffusion","dissipation"),fill=c("yellow","red","
              blue"))
  dev.off()
}
#other plot for check
#spplot(states[[7]], zcol = "Total")
#count nb. of hotspots
tau<-0.25#(0.25 corresponds to one-half of PRIO-grid cell area)
areasdiff<-list()
maxareasdiff<-list()
indexdiff<-list()
diffdisag<-list()
for(j in 1:length(diffnetdf)){
  #calculate areas of all polygons included in buffer areas for each
      year
  areasdiff[[j]]<- gArea(diffnetdf[[j]], byid=TRUE)
  indexdiff[[j]] <- which(areasdiff[[j]]>tau)#find position where area >
      tau
```

```
}
#keep polygon with area>tau
diffpolynetdf<-list()
for(j in 1: length(diffnetdf)){
  diffpolynetdf[[j]]<-diffnetdf[[j]][indexdiff[[j]],]#keep polygon with
     area>tau
}
nbhot<-list()
for(j in 1:9){
  nbhot[[j+2]]<-length(disaggregate(diffpolynetdf[[j+2]]))}
nbhot<-nbhot[c(-1,-2)]
Reduce("+",nbhot)#nb. of diff areas
#count nb. of sign.neighbourhood areas (total of 53)
nbdiff<-list()
for(j in 1:9){
  nbdiff[[j+2]]<-length(disaggregate(obsdiff[[j+2]]))}
nbdiff<-nbdiff[c(-1,-2)]
Reduce("+",nbdiff)#nb. of diff areas
#count nb. of neighbourhood areas with sidgnificant diffusion/
   dissipation (>0.1)
#find origin /centroid of diffusion areas
centdiff<-list()
origin<-list()
for(j in 1:9){
obsdiff[[j]]<-spTransform(obsdiff[[j]],CRS("+init=epsg:3857"))#diffusion
    polygon
states[[j]]<-spTransform(states[[j]],CRS("+init=epsg:3857"))#country
   polygon
centdiff[[j]]<-gCentroid(obsdiff[[j]],byid=TRUE,id=obsdiff[[j]]@data$N)#
   centroid of diffusion areas
#extract country name related to the position
origin[[j]]<-over(centdiff[[j]],states[[j]])
origin[[j]]<-origin[[j]][c(64,65)]
colnames(origin[[j]])<-c("origin","FSIorigin")
origin[[j]]$ID<-rownames(origin[[j]])
#join attributes of hotspot origin to diffusion polygons
obsdiff[[j]]@data = data.frame(obsdiff[[j]]@data, origin[[j]][match(
   obsdiff[[j]]@data[,"N"], origin[[j]][,"ID"]),])
}
# put states as dataframe
```

```
countrydf<-list()
for(j in 1:9){
  countrydf[[j]]<-as.data.frame(states[[j]]@data$Country)#all world
      country
  countrydf[[j]]$ID<-rownames(countrydf[[j]])#countries and their ID
  colnames(countrydf[[j]])<-c("country","ID")
}
#find countries intersecting with diffusion areas (could be simplified?)
step1<-list()
diffstates1<-list()
  for(i in 1:length(obsdiff[[1]])){#from 1 to the numb of diffusion
      areas in each year
  step1[[i]]<-as.data.frame(states[[1]][obsdiff[[1]][i,],])
  diffstates1[[i]]<-as.data.frame(step1[[i]]$Country)
  diffstates1[[i]]$FSIdestin<-step1[[i]]$Total
  diffstates1[[i]]$N<-as.integer(i)#should put identical numbers: 1 for
      first elements of the list, 2 for second elements of the list,...
      for further merging
colnames(diffstates1[[i]])<-c("destin","FSIdestin","N")
}
step2<-list()
diffstates2<-list()
for(i in 1:length(obsdiff[[2]])){#from 2 to the numb of diffusion areas
    in each year
  step2[[i]]<-as.data.frame(states[[2]][obsdiff[[2]][i,],])
  diffstates2[[i]]<-as.data.frame(step2[[i]]$Country)
  diffstates2[[i]]$FSIdestin<-step2[[i]]$Total
  diffstates2[[i]]$N<-as.integer(i)#should put identical numbers: 1 for
      first elements of the list, 2 for second elements of the list,...
      for further merging
  colnames(diffstates2[[i]])<-c("destin","FSIdestin","N")
 }
step3<-list()
diffstates3<-list()
for(i in 1:length(obsdiff[[3]])){#from 3 to the numb of diffusion areas
    in each year
  step3[[i]]<-as.data.frame(states[[3]][obsdiff[[3]][i,],])
  diffstates3[[i]]<-as.data.frame(step3[[i]]$Country)
  diffstates3[[i]]$FSIdestin<-step3[[i]]$Total
```

```
  diffstates3[[i]]$N<-as.integer(i)#should put identical numbers: 1 for
      first elements of the list, 2 for second elements of the list,...
      for further merging
  colnames(diffstates3[[i]])<-c("destin","FSIdestin","N")
}
step4<-list()
diffstates4<-list()
for(i in 1:length(obsdiff[[4]])){#from 4 to the numb of diffusion areas
    in each year
  step4[[i]]<-as.data.frame(states[[4]][obsdiff[[4]][i,],])
  diffstates4[[i]]<-as.data.frame(step4[[i]]$Country)
  diffstates4[[i]]$FSIdestin<-step4[[i]]$Total
  diffstates4[[i]]$N<-as.integer(i)#should put identical numbers: 1 for
      first elements of the list, 2 for second elements of the list,...
      for further merging
  colnames(diffstates4[[i]])<-c("destin","FSIdestin","N")
  }
step5<-list()
diffstates5<-list()
for(i in 1:length(obsdiff[[5]])){#from 5 to the numb of diffusion areas
    in each year
  step5[[i]]<-as.data.frame(states[[5]][obsdiff[[5]][i,],])
  diffstates5[[i]]<-as.data.frame(step5[[i]]$Country)
  diffstates5[[i]]$FSIdestin<-step5[[i]]$Total
  diffstates5[[i]]$N<-as.integer(i)#should put identical numbers: 1 for
      first elements of the list, 2 for second elements of the list,...
      for further merging
  colnames(diffstates5[[i]])<-c("destin","FSIdestin","N")
  }
step6<-list()
diffstates6<-list()
for(i in 1:length(obsdiff[[6]])){#from 6 to the numb of diffusion areas
    in each year
  step6[[i]]<-as.data.frame(states[[6]][obsdiff[[6]][i,],])
  diffstates6[[i]]<-as.data.frame(step6[[i]]$Country)
  diffstates6[[i]]$FSIdestin<-step6[[i]]$Total
  diffstates6[[i]]$N<-as.integer(i)#should put identical numbers: 1 for
      first elements of the list, 2 for second elements of the list,...
      for further merging
  colnames(diffstates6[[i]])<-c("destin","FSIdestin","N")
```

```
  }
step7<-list()
diffstates7<-list()
for(i in 1:length(obsdiff[[7]])){#from 7 to the numb of diffusion areas
    in each year
  step7[[i]]<-as.data.frame(states[[7]][obsdiff[[7]][i,],])
  diffstates7[[i]]<-as.data.frame(step7[[i]]$Country)
  diffstates7[[i]]$FSIdestin<-step7[[i]]$Total
  diffstates7[[i]]$N<-as.integer(i)#should put identical numbers: 1 for
      first elements of the list, 2 for second elements of the list,...
      for further merging
  colnames(diffstates7[[i]])<-c("destin","FSIdestin","N")
  }
step8<-list()
diffstates8<-list()
for(i in 1:length(obsdiff[[8]])){#from 8 to the numb of diffusion areas
    in each year
  step8[[i]]<-as.data.frame(states[[8]][obsdiff[[8]][i,],])
  diffstates8[[i]]<-as.data.frame(step8[[i]]$Country)
  diffstates8[[i]]$FSIdestin<-step8[[i]]$Total
  diffstates8[[i]]$N<-as.integer(i)#should put identical numbers: 1 for
      first elements of the list, 2 for second elements of the list,...
      for further merging
  colnames(diffstates8[[i]])<-c("destin","FSIdestin","N")
}
step9<-list()
diffstates9<-list()
for(i in 1:length(obsdiff[[9]])){#from 9 to the numb of diffusion areas
    in each year
  step9[[i]]<-as.data.frame(states[[9]][obsdiff[[9]][i,],])
  diffstates9[[i]]<-as.data.frame(step9[[i]]$Country)
  diffstates9[[i]]$FSIdestin<-step9[[i]]$Total
  diffstates9[[i]]$N<-as.integer(i)#should put identical numbers: 1 for
      first elements of the list, 2 for second elements of the list,...
      for further merging
  colnames(diffstates9[[i]])<-c("destin","FSIdestin","N")
  }
#put df together into a list
diffstates<-list(diffstates1,diffstates2,diffstates3,diffstates4,
    diffstates5, diffstates6,diffstates7,diffstates8,diffstates9)
```

```
for(j in 1:9){#from 9 to the numb of diffusion areas in each year
diffstates[[j]] = Reduce(function(...) merge(..., all=T), diffstates[[j
    ]])#merge df within each year
}

#join attributes of hotspot origin to diffusion polygons
 difftable<-list()
 for(i in 1:9){#from 1 to the numb of diffusion areas in each year
 obsdiff[[i]]<-as.data.frame(obsdiff[[i]])#put as a dataframe for
     further merging
 obsdiff[[i]]$N <- as.numeric(as.factor(with(obsdiff[[i]], paste(ID, sep
     ="_"))))#the variable "N" give a unique identifier for each centroid
      of diffusion areas (required for further merging)
  difftable[[i]]<-merge(diffstates[[i]],obsdiff[[i]],by='N',all.x=TRUE)#
     merging destination states with diffusion areas and origin
 keeps <- c("N","destin","FSIdestin","diffpos","diffneg","deltadiff","
     origin","FSIorigin")
 difftable[[i]]<-difftable[[i]][keeps]#keep important variables only
 difftable[[i]]$year<-2004+i#add year of diffusion as new variable
}
#put all df together
diffusiontab<-rbind(difftable[[1]],difftable[[2]],difftable[[3]],
    difftable[[4]],difftable[[5]],
                    difftable[[6]],difftable[[7]],difftable[[8]],
                        difftable[[9]])
#define threshold of the ratio for a diffusion to be significant
difft<-1.1
#define threshold of the ratio for a dissipation to be significant
disst<-0.9
#keep important variables and produce new variables for publication in
    thesis
diffusiontab$diffusion<-ifelse(diffusiontab$deltadiff>=difft,"diffusion"
    ,ifelse(diffusiontab$deltadiff<= 0.9,"retraction","N/D"))
diffusiontab<-diffusiontab[c(-4,-5)]
diffusiontab<-diffusiontab[,c("N","origin","FSIorigin","destin","
    FSIdestin","deltadiff","diffusion","year")]
colnames(diffusiontab)<-c("N","Origin","FSIorigin","Destin","FSIdestin",
    "Obsdiff","Diffusion","Year")
diffusiontab$Origin<-as.character(diffusiontab$Origin);diffusiontab$
    Destin<-as.character(diffusiontab$Destin)
```

```
diffusiontab[diffusiontab=="Democratic␣Republic␣of␣the␣Congo"] <- "DRC"
diffusiontab[diffusiontab=="United␣Republic␣of␣Tanzania"] <- "Tanzania"
#get average FSI from original files of FSI years 2005 to 2013
FSIavg<-c(88.68,70.79,70.56,70.86,72.07,71.87,71.08,70.88,70.5)
Year<-c(2005,2006,2007,2008,2009,2010,2011,2012,2013)
FSIavg<-as.data.frame(cbind(FSIavg,Year))
#merge with df
diffusiontab<-merge(diffusiontab,FSIavg,by="Year")
#reorder df
diffusiontab<-diffusiontab[,c("N","Origin","Destin","Obsdiff","Diffusion
    ","FSIorigin","FSIdestin","FSIavg","Year")]
#rounding
diffusiontab$Year<-as.integer(diffusiontab$Year)
diffusiontab
#OPTION:KEEP ONLY DIFFUSION OR RETRACTION IF ABSOLUTE VALUE > 10%
#diffusiontab<-diffusiontab[-grep("N/D", diffusiontab$Diffusion),]

#histogram which relates the difference in FSI according to areas of
    diffusion and areas of retraction
Diffpos<-diffusiontab[diffusiontab$Obsdiff>=difft,]
Diffneg<-diffusiontab[diffusiontab$Obsdiff<=disst,]
#basic statistics
difffsistat<-aggregate(diffusiontab, by=list(diffusiontab$Diffusion,
    diffusiontab$Year), FUN=mean, na.rm=TRUE)
fsimax<-aggregate(diffusiontab, by=list(diffusiontab$Diffusion,
    diffusiontab$Year), FUN=max, na.rm=TRUE)
fsimin<-aggregate(diffusiontab, by=list(diffusiontab$Diffusion,
    diffusiontab$Year), FUN=min, na.rm=TRUE)

difffsistat<-difffsistat[c(-3,-4,-5,-7)]
fsimax<-fsimax[c(-3,-4,-5,-6,-7,-10,-11)]
names(fsimax)<-c("Group.1","Group.2","FSIoriginmax","FSIdestinmax")
fsimin<-fsimin[c(-3,-4,-5,-6,-7,-10,-11)]
names(fsimin)<-c("Group.1","Group.2","FSIoriginmin","FSIdestinmin")
difffsistat<-merge(difffsistat,fsimax,by=c("Group.1","Group.2"))
difffsistat<-merge(difffsistat,fsimin,by=c("Group.1","Group.2"))
#boxplot for LateX thesis file
#look at diffusion observation in origin country
diffonly<-difffsistat[difffsistat$Group.1=="diffusion",]
retraconly<-difffsistat[difffsistat$Group.1=="retraction",]
```

```
#stat. test to check if FSI in hotspots different from FSI in areas of
    contagion/retraction + comparision with FSI world avg.
wilcox.test(diffusiontab$FSIorigin, diffusiontab$FSIavg,alternative = c(
    "greater"))
wilcox.test(diffusiontab$FSIdestin, diffusiontab$FSIavg,alternative = c(
    "greater"))
wilcox.test(diffonly$FSIorigin, diffonly$FSIdestin,alternative = c("
    greater"))
wilcox.test(diffonly$FSIorigin, diffonly$FSIavg,alternative = c("greater
    "))
wilcox.test(diffonly$FSIdestin, diffonly$FSIavg,alternative = c("greater
    "))
wilcox.test(retraconly$FSIorigin, retraconly$FSIdestin,alternative = c("
    greater"),exact = FALSE, correct = TRUE)
wilcox.test(retraconly$FSIorigin, retraconly$FSIavg,alternative = c("
    greater"))
wilcox.test(retraconly$FSIdestin, retraconly$FSIavg,alternative = c("
    greater"))
hotanddiff<-subset(diffusiontab, Diffusion=='diffusion')
hotandnodiff<-subset(diffusiontab, Diffusion!='diffusion')
#stat. test to check if FSI in hotspots that diffuse different from
    their diffusion areas
wilcox.test(hotanddiff$FSIorigin, hotanddiff$FSIdestin,alternative = c("
    greater"))
wilcox.test(hotanddiff$FSIorigin, hotanddiff$FSIdestin,alternative = c("
    less"))
mean(hotanddiff$FSIorigin);mean(hotanddiff$FSIdestin,na.rm=T)
#stat. test to check if FSI in hotspots that diffuse different from FSI
    in hotspots that do not diffuse
wilcox.test(hotanddiff$FSIorigin, hotandnodiff$FSIorigin,alternative = c
    ("greater"))
wilcox.test(hotanddiff$FSIorigin, hotandnodiff$FSIorigin,alternative = c
    ("less"))
#stat. test to check if FSI area of diffusion different from FSI in
    hotspots that do not diffuse
wilcox.test(hotanddiff$FSIdestin, hotandnodiff$FSIorigin,alternative = c
    ("less"))
wilcox.test(hotanddiff$FSIdestin, hotandnodiff$FSIorigin,alternative = c
    ("greater"))
```

```
#stat. test to check if  FSI in hotspot that do not diffuse is equal
    with FSi in its neighbourhood
wilcox.test(hotandnodiff$FSIdestin, hotandnodiff$FSIorigin,alternative =
    c("less"))
wilcox.test(hotandnodiff$FSIdestin, hotandnodiff$FSIorigin,alternative =
    c("greater"))
#table for LateX
require(xtable)
print(xtable(diffusiontab,digits=1), include.rownames=FALSE)
#box plot for the thesis
a=diffusiontab[diffusiontab$Diffusion=="diffusion",]$FSIdestin
b=diffusiontab[diffusiontab$Diffusion=="diffusion",]$FSIorigin
c=diffusiontab[diffusiontab$Diffusion=="N/D",]$FSIorigin
# Make a list of these 2 vectors
C=list(a,b,c)
names(C)=c(paste("Diffusion\n " , "area" , sep=""), paste("Hotspot\n " ,
    "with diffusion" , sep=""), paste("Hotspot\n " , "without diffusion"
    , sep=""))
#I change the mgp argument to avoid the text overlays the x axis
plot.new()
mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    revision/revisedPhDthesis/Chapter6/Figs/Raster/pFSIschema.png",sep=""
    )
png(file=mypath,width=900,height=720)
par(cex=2.4,mar=c(3, 4, 0.2,1),mgp=c(3,2,0))#mar: c(bottom,left,top,
    right)
boxplot(C, main=NA,outcex=1,
        xlab=NA, ylab="FSI",ylim=c(60,120))
abline(h = mean(diffusiontab$FSIavg,na.rm=T), col = "black",lwd=2,lty=2)
text(x=2.0, y = 71, labels = "World average FSI",
     pos = NULL, offset = 0.5, vfont = NULL,
     cex = 0.75, col = NULL, font = NULL)
text(x=3.0, y = 119, labels = "Poisson model",
     pos = NULL, offset = 0.5, vfont = NULL,
     cex = 0.75, col = NULL, font = NULL)
dev.off();dev.off()
#look at the FSI differences between origin and neighbourhood for each
    pair of hotspot/contagious diffusion areas
#average values for each polygon in the origin and destination
```

```
#aggregate by polygon ID and by year (and by diffusion to differentiate
    diffusion from retraction) in order to have couple of origin and its
    related areas
#basic statistics
deltaFSImean<-aggregate(diffusiontab, by=list(diffusiontab$Diffusion,
    diffusiontab$Year,diffusiontab$N), FUN=mean, na.rm=TRUE)
deltaFSImax<-aggregate(diffusiontab, by=list(diffusiontab$Diffusion,
    diffusiontab$Year,diffusiontab$N), FUN=max, na.rm=TRUE)
deltaFSImin<-aggregate(diffusiontab, by=list(diffusiontab$Diffusion,
    diffusiontab$Year,diffusiontab$N), FUN=min, na.rm=TRUE)
#Option: plot observation with "significant" diffusion or retraction
deltaFSImean<-deltaFSImean[c(1,2,3,9,10,11)]
deltaFSImax<-deltaFSImax[c(1,2,3,9,10)]
names(deltaFSImax)<-c("Group.1","Group.2","Group.3","FSIdestinmax","
    FSIoriginmax")
deltaFSImin<-deltaFSImin[c(1,2,3,9,10)]
names(deltaFSImin)<-c("Group.1","Group.2","Group.3","FSIdestinmin","
    FSIoriginmin")
deltaFSI<-merge(deltaFSImean,deltaFSImax,by=c("Group.1","Group.2","Group
    .3"))
deltaFSI<-merge(deltaFSI,deltaFSImin,by=c("Group.1","Group.2","Group.3")
    )
deltaFSI$deltamean<-deltaFSI$FSIorigin-deltaFSI$FSIdestin#create
    variable that expressses the difference in the mean of FSI between
    origin and destination
#separate diffusion and retraction observations
diffdeltaFSIonly<-deltaFSI[deltaFSI$Group.1=="diffusion",]
retracdeltaFSIonly<-deltaFSI[deltaFSI$Group.1=="retraction",]
#diffusion FSI
#World FSI from 2006 to 2013:
    70.79,70.56,70.86,72.07,71.87,71.08,70.88,70.5
FSIworld0613<-mean(70.79,70.56,70.86,72.07,71.87,71.08,70.88,70.5)
png(filename="~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/
    diffusion/deltaFSIdiff.png",width=1444,height=724)
par(cex=2.5,mar=c(4.5, 5.1, 0.8, 0.5) + 0.1)#mar: c(bottom,left,top,
    right)
plot(diffdeltaFSIonly$Group.2,diffdeltaFSIonly$FSIorigin,
      ylim=c(0,120),pch=19, xlab="years", ylab="",main="",axes = FALSE,
          bty = "n")
axis(1, at=2006:2013, labels=c(2006,2007,2008,2009,2010,2011,2012,2013))
```

```
axis(side=2, at = c(60,80,100,120))
mtext("FSI", side=2, line=3,cex=2.5,adj=0.74)
points(diffdeltaFSIonly$Group.2, diffdeltaFSIonly$FSIdestin,pch=8,lwd=1,
    cex=1)
clip(2006,2013,0,120)
abline(a=FSIworld0613,b=0,lwd=2,lty=3)
text(2007,70,"World␣avg.␣FSI␣2006-13",srt=0.2,pos=3)
par(new = TRUE,cex=2.5)
plot(diffdeltaFSIonly$Group.2,diffdeltaFSIonly$deltamean, pch=17,lwd=2,
    axes = FALSE, bty = "n", xlab = "", ylab = "",ylim=c(-10,90))
axis(side=2, at = c(-10,0,10,20))
mtext("delta␣FSI", side=2, line=3,cex=2.5,adj=0.12)
clip(2006,2013,-25,15)
abline(a=0,b=0,lwd=2,lty=2)
dev.off()
#retraction FSI
png(filename="~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/
    deltaFSIretrac.png",width=1444,height=724)
par(cex=2.5,mar=c(4.5, 5.1, 0.8, 0.5) + 0.1)#mar: c(bottom,left,top,
    right)
plot(retracdeltaFSIonly$Group.2,retracdeltaFSIonly$FSIorigin,
      ylim=c(0,120),pch=19, xlab="years", ylab="",main="",axes = FALSE,
          bty = "n")
axis(1, at=2006:2013, labels=c(2006,2007,2008,2009,2010,2011,2012,2013))
axis(side=2, at = c(60,80,100,120))
mtext("FSI", side=2, line=3,cex=2.5,adj=0.74)
points(retracdeltaFSIonly$Group.2, retracdeltaFSIonly$FSIdestin,pch=8,
    lwd=1,cex=1)
clip(2006,2013,0,120)
abline(a=FSIworld0613,b=0,lwd=2,lty=3)
text(2007,70,"World␣avg.␣FSI␣2006-13",srt=0.2,pos=3)
par(new = TRUE,cex=2.5)
plot(retracdeltaFSIonly$Group.2,retracdeltaFSIonly$deltamean, pch=17,lwd
    =2, axes = FALSE, bty = "n", xlab = "", ylab = "",ylim=c(-5,100))
axis(side=2, at = c(-5,0,5,10,15,20,25))
mtext("delta␣FSI", side=2, line=3,cex=2.5,adj=0.12)
clip(2006,2013,-25,15)
abline(a=0,b=0,lwd=2,lty=2)
dev.off()
#end
```

```
save.image("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/
    pdiffusion_FSI.RData")
#load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/pdiffusion_
    FSI.RData")
```

D.6 Assessment of Economic Theory

This is the code used to empirically assess the theories that relate economic development and the diffusion and dissipation of lethality and real observations.

```
#SPATIO-TEMPORAL MODELLING / Bernoulli model of lethal events worldwide
    2002-2013
#The code assesses economic theories with regard to the contagious
    diffusion of lethal terrorism and real observations.
setwd("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STbinomialtotal")
rm(list=ls())
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/diffusion_modelvsobs.
    RData")
#since some operations are highly time consuming, load this data to have
     the results directly:
#load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/diffusion_GDP.RData")
require(foreign)
require(sp)
library(INLA)
library(spatstat)
library(fields)
library(maptools)
library(fields)
library(lattice)
require(rgeos)
require(raster)
require(rgdal)
require(plyr)
require(ggplot2)
#use same projection
for(j in 1:11){
obsdiff[[j]]<-spTransform(obsdiff[[j]],CRS("+proj=longlat␣+datum=WGS84␣+
    no_defs+towgs84=0,0,0+ellps=WGS84"))#diffusion polygon
```

```
hotrpoly[[j]]<-spTransform(hotrpoly[[j]],CRS("+proj=longlat␣+datum=WGS84
    ␣+no_defs+towgs84=0,0,0+ellps=WGS84"))#hotspots that are within
    diffusion polygon or that do not have associated diffusion polygon
}
#download luminosity values from 2003 to 2013
#Luminosity (spatio-temporal covariate)
#adding luminosity data as spatio-temporal covariate
#satellite night light (which has been reworked in arcgis in order to
    put 0 if no values + extent to -90;90 latitude)
#with arcgis "raster calculator" function: Con(IsNull("inputraster"),0,"
    inputraster") and processing environment add lat -90 to 90 and lon
    -180 to 180)
#then exported as ASCII with arcgis "sample" function
lum<-list()
for(j in 1:7){#extraction for 2003 to 2009
  lum[[j]]<-raster(paste("C:/Users/apython/Documents/Andre/Education/
      StAndrews/PhD/PhD_R/luminosity/lum0",2+j,".tif",sep=""))
   projection(lum[[j]])<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84
      =0,0,0+ellps=WGS84"
    for(j in 8:11){#extraction for 2010 to 2013
    lum[[j]]<-raster(paste("C:/Users/apython/Documents/Andre/Education/
       StAndrews/PhD/PhD_R/luminosity/lum",2+j,".tif",sep=""))
    projection(lum[[j]])<-"+proj=longlat␣+datum=WGS84␣+no_defs+towgs84
       =0,0,0+ellps=WGS84"
         }}
#extract values of luminosity values within hotspot of terrorism
meanlumhot<-list()
for(j in 1:11){#computationally intense, careful, this operation could
    take a lot of time.
meanlumhot[[j]]<-extract(lum[[j]],hotrpoly[[j]])
}
#correction for luminosity values in order to be able to compare them
    through time (according to NOAA)
#intercalibration based on Elvidge2013
#Process: 1) calculate: Y = C0 + C1X + C2X^2 // 2) Values > 63 are
    truncated at 63 // 3) values = 0 stay zero.
#f0 <- function(x){ifelse(x>0, 0.0491+0.9568*x+0.0010*x^2, 0)}#2002
    values: if the values are above 0, execute intercalibration
f1 <- function(x){ifelse(x>0, 0.2217+1.5122*x+0.0010*x^2, 0)}#2003
    values
```

```
f2 <- function(x){ifelse(x>0, 0.2853+1.1955*x+0.0010*x^2, 0)}
f3 <- function(x){ifelse(x>0, -0.0001+1.4159*x+0.0010*x^2, 0)}
f4 <- function(x){ifelse(x>0, 0.1065+1.1371*x+0.0010*x^2, 0)}
f5 <- function(x){ifelse(x>0, 0.6394+0.9114*x+0.0010*x^2, 0)}
f6 <- function(x){ifelse(x>0, 0.5564+0.9931*x+0.0010*x^2, 0)}
f7 <- function(x){ifelse(x>0, 0.9492+1.0683*x+0.0010*x^2, 0)}
f8 <- function(x){ifelse(x>0, 2.3430+0.5102*x+0.0010*x^2, 0)}
f9 <- function(x){ifelse(x>0, 1.8956+0.7345*x+0.0010*x^2, 0)}
f10 <- function(x){ifelse(x>0, 1.8750+0.6203*x+0.0010*x^2, 0)}
f11 <- function(x){ifelse(x>0, 1.8750+0.6203*x+0.0010*x^2, 0)}#2013
    values
for(j in 1:11){#calibrated values instead of original values
  meanlumhot[[j]]<-sapply(meanlumhot[[j]],mean,na.rm=TRUE)
  meanlumhot[[j]]<-as.data.frame(sapply(meanlumhot[[j]],paste("f",j,sep=
      "")))
  meanlumhot[[j]][meanlumhot[[j]] > 63] <- 63 #truncate if values are
      higher than 63
  meanlumhot[[j]]<-as.data.frame(meanlumhot[[j]])
  meanlumhot[[j]]$N<-rownames(meanlumhot[[j]])
  meanlumhot[[j]]$Year<-2002+j
  colnames(meanlumhot[[j]])<-c("meanlum","N","Year")
}
#look at diffusion areas
#extract values of luminosity values within hotspot of terrorism
meanlumdiff<-list()
rm(j)
for(j in 1:11){#computationally intense...
  meanlumdiff[[j]]<-extract(lum[[j]],obsdiff[[j]])
}

for(j in 1:11){#calibrated values instead of original values
  meanlumdiff[[j]]<-sapply(meanlumdiff[[j]],mean,na.rm=TRUE)
  meanlumdiff[[j]]<-as.data.frame(sapply(meanlumdiff[[j]],paste("f",j,
      sep="")))
  meanlumdiff[[j]][meanlumdiff[[j]] > 63] <- 63 #truncate if values are
      higher than 63
  meanlumdiff[[j]]<-as.data.frame(meanlumdiff[[j]])
  meanlumdiff[[j]]$N<-rownames(meanlumdiff[[j]])
  meanlumdiff[[j]]$Year<-2002+j
  colnames(meanlumdiff[[j]])<-c("meanlum","N","Year")
```

```
}
# #OPTION: partial merge (all.x=T): keeping only hotspots that are
    showing significant diffusion or retraction processes
# #To use in order to focus the analysis of the relationshiop between
    terrorism's diffusion (and retraction) in areas where diffusion/
    retraction originates and spreads.
# lumdiffusion<-list()
# for(j in 1:11){#put data into dataframe and add variables for merging
    issues
# lumdiffusion[[j]]<-merge(lumdiff[[j]],lumhot[[j]],by=c("Year","N"),all
    .x=TRUE)
# colnames(lumdiffusion[[j]])<-c("Year","N","meanlumdiff","sdlumdiff","
    meanlumhot","sdlumhot")
# }
#OPTION2: total merge: keeping all hotspots
lumdiffusion<-list()
for(j in 1:11){#put data into dataframe and add variables for merging
    issues
lumdiffusion[[j]]<-merge(meanlumdiff[[j]],meanlumhot[[j]],by=c("Year","N
    "))
colnames(lumdiffusion[[j]])<-c("Year","N","meanlumdiff","meanlumhot")
}
#merge lum values with observed diffusion values
lumfinal<-list()
for(j in 1:11){#put data into dataframe and add variables for merging
    issues
  lumfinal[[j]]<-merge(lumdiffusion[[j]],obsdiff[[j]],by=c("N"),all.x=
      TRUE)
}
#put all df together
lumfinal<-rbind(lumfinal[[1]],lumfinal[[2]],lumfinal[[3]],lumfinal[[4]],
    lumfinal[[5]],lumfinal[[6]],
                lumfinal[[7]],lumfinal[[8]],lumfinal[[9]],lumfinal
                    [[10]],lumfinal[[11]])
lumfinal$Year<-as.integer(lumfinal$Year)
lumfinal<-lumfinal[c(-6,-8,-10)]#keep important variables
lumfinal<-lumfinal[complete.cases(lumfinal),]#delete possible NA values
#separate diffusion from retraction observations
lumfinal$diffusion<-ifelse(lumfinal$deltadiff>=1.1,"diffusion",ifelse(
    lumfinal$deltadiff<= 0.9,"retraction","N/D"))
```

```
#OPTION:KEEP ONLY DIFFUSION OR RETRACTION IF ABSOLUTE VALUE > 10%
#lumfinal<-lumfinal[- grep("N/D", lumfinal$diffusion),]
#testing difference in lum between hotspots that diffuse and their
    neighbourhood (not sig)
wilcox.test(lumfinal[lumfinal$diffusion=="diffusion",]$meanlumdiff,
    lumfinal[lumfinal$diffusion=="diffusion",]$meanlumhot,alternative = c
    ("less"),paired=TRUE)#less means x<y
wilcox.test(lumfinal[lumfinal$diffusion=="diffusion",]$meanlumdiff,
    lumfinal[lumfinal$diffusion=="diffusion",]$meanlumhot,alternative = c
    ("greater"),paired=TRUE)#greater means x>y
#testing difference in lum from hotspots with diffusion vs hotspots
    without diffusion areas (lum in hot with diff > hot without diff)
wilcox.test(lumfinal[lumfinal$diffusion=="diffusion",]$meanlumhot,
    lumfinal[lumfinal$diffusion!="diffusion",]$meanlumhot,alternative = c
    ("less"),exact=FALSE)#less means x<y
wilcox.test(lumfinal[lumfinal$diffusion=="diffusion",]$meanlumhot,
    lumfinal[lumfinal$diffusion!="diffusion",]$meanlumhot,alternative = c
    ("greater"),exact=FALSE)#greater means x>y
#testing difference in lum between diffusion areas in hotspots and
    hotspots without diffusion (lum in diff area > hot without diff)
wilcox.test(lumfinal[lumfinal$diffusion=="diffusion",]$meanlumdiff,
    lumfinal[lumfinal$diffusion!="diffusion",]$meanlumhot,alternative = c
    ("less"),exact=FALSE)#less means x<y
wilcox.test(lumfinal[lumfinal$diffusion=="diffusion",]$meanlumdiff,
    lumfinal[lumfinal$diffusion!="diffusion",]$meanlumhot,alternative = c
    ("greater"),exact=FALSE)#greater means x>y
#testing difference in lum between hotspots that do not diffuse and
    their neighbourhood (lum in hot without diff > their neighbourhood)
wilcox.test(lumfinal[lumfinal$diffusion!="diffusion",]$meanlumdiff,
    lumfinal[lumfinal$diffusion!="diffusion",]$meanlumhot,alternative = c
    ("less"),paired=TRUE)#less means x<y
wilcox.test(lumfinal[lumfinal$diffusion!="diffusion",]$meanlumdiff,
    lumfinal[lumfinal$diffusion!="diffusion",]$meanlumhot,alternative = c
    ("greater"),paired=TRUE)#greater means x>y
#comparing hotspots mean lum with other locations where terrorism
    occurred
GTDlummean<- -0.1218624#from SPDE.R (average of standardised lum where
    GTD taken)
#one sample student t.test
```

```
t.test(lumfinal$meanlumhot, mu=GTDlummean)#lum in hot > lum everywhere
    else
t.test(lumfinal$meanlumdiff, mu=GTDlummean)#lum in diff area > lum
    everywhere else

a=lumfinal[lumfinal$diffusion=="diffusion",]$meanlumdiff
b=lumfinal[lumfinal$diffusion=="diffusion",]$meanlumhot
c=lumfinal[lumfinal$diffusion=="N/D",]$meanlumhot
# Make a list of these 2 vectors
C=list(a,b,c)
names(C)=c(paste("Diffusion\n " , "area" , sep=""), paste("Hotspot\n " ,
     "with diffusion" , sep=""), paste("Hotspot\n " , "without diffusion"
     , sep=""))
#I change the mgp argument to avoid the text overlays the x axis
plot.new()
mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    revision/revisedPhDthesis/Chapter6/Figs/Raster/lumschema.png",sep="")
png(file=mypath,width=900,height=720)
par(cex=2.4,mar=c(3, 4, 0.2,1),mgp=c(3,2,0))#mar: c(bottom,left,top,
    right)
boxplot(C, main=NA,outcex=1,
        xlab=NA, ylab="(standardised) luminosity ",ylim=c(-1,8))
abline(h = GTDlummean, col = "black",lwd=2,lty=2)
text(x=2.7, y = -0.4, labels = "World average (standardised) luminosity"
    ,
      pos = NULL, offset = 0.5, vfont = NULL,
      cex = 0.75, col = NULL, font = NULL)
text(x=3.0, y = 7.9, labels = "Bernoulli model",
      pos = NULL, offset = 0.5, vfont = NULL,
      cex = 0.75, col = NULL, font = NULL)
dev.off();dev.off()

save.image("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/diffusion_GDP.
    RData")
#load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/diffusion_GDP.RData")
#end
```

This is the code used to empirically assess the theories relate economic development and the diffusion and dissipation of the number of lethal terrorist events and real observations.

```
#SPATIO-TEMPORAL MODELLING / Poisson model of the number of lethal
    events worldwide 2002-2013
#The code assesses the performance of the model with regard to the
    contagious diffusion of lethal terrorism and real observations.
setwd("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson")
rm(list=ls())
load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/diffusion_
    modelvsobs.RData")
#since some operations are highly time consuming, load this data to get
    the results directly:
#load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/pdiffusion_
    GDP.RData")
require(foreign)
require(sp)
library(INLA)
library(spatstat)
library(fields)
library(maptools)
library(fields)
library(lattice)
require(rgeos)
require(raster)
require(rgdal)
require(plyr)
require(ggplot2)
#same projection
for(j in 1:11){
obsdiff[[j]]<-spTransform(obsdiff[[j]],CRS("+proj=longlat␣+datum=WGS84␣+
    no_defs+towgs84=0,0,0+ellps=WGS84"))#diffusion polygon
hotrpoly[[j]]<-spTransform(hotrpoly[[j]],CRS("+proj=longlat␣+datum=WGS84
    ␣+no_defs+towgs84=0,0,0+ellps=WGS84"))#hotspots that are within
    diffusion polygon or that do not have associated diffusion polygon
}
#download luminosity values from 2003 to 2013
#Luminosity (spatio-temporal covariate)
#adding luminosity data as spatio-temporal covariate
#satellite night light (which has been reworked in arcgis in order to
    put 0 if no values + extent to -90;90 latitude)
```

```
#with arcgis "raster calculator" function: Con(IsNull("inputraster"),0,"
    inputraster") and processing environment add lat -90 to 90 and lon
    -180 to 180)
#then exported as ASCII with arcgis "sample" function
lum<-list()
for(j in 1:7){#extraction for 2003 to 2009
  lum[[j]]<-raster(paste("C:/Users/apython/Documents/Andre/Education/
      StAndrews/PhD/PhD_R/luminosity/lum0",2+j,".tif",sep=""))
   projection(lum[[j]])<-"+proj=longlat +datum=WGS84 +no_defs+towgs84
      =0,0,0+ellps=WGS84"
    for(j in 8:11){#extraction for 2010 to 2013
    lum[[j]]<-raster(paste("C:/Users/apython/Documents/Andre/Education/
       StAndrews/PhD/PhD_R/luminosity/lum",2+j,".tif",sep=""))
    projection(lum[[j]])<-"+proj=longlat +datum=WGS84 +no_defs+towgs84
       =0,0,0+ellps=WGS84"
         }}
#extract values of luminosity values within hotspot of terrorism
meanlumhot<-list()
for(j in 1:11){#computationally intense, careful, this operation could
    take a lot of time.
  meanlumhot[[j]]<-extract(lum[[j]],hotrpoly[[j]])
}
#correction for luminosity values in order to be able to compare them
    through time (according to NOAA)
# # intercalibration based on Elvidge2013
# # Process: 1) calculate: Y = C0 + C1X + C2X^2 // 2) Values > 63 are
    truncated at 63 // 3) values = 0 stay zero.
#f0 <- function(x){ifelse(x>0, 0.0491+0.9568*x+0.0010*x^2, 0)}#2002
    values: if the values are above 0, execute intercalibration
f1 <- function(x){ifelse(x>0, 0.2217+1.5122*x+0.0010*x^2, 0)}#2003
    values
f2 <- function(x){ifelse(x>0, 0.2853+1.1955*x+0.0010*x^2, 0)}
f3 <- function(x){ifelse(x>0, -0.0001+1.4159*x+0.0010*x^2, 0)}
f4 <- function(x){ifelse(x>0, 0.1065+1.1371*x+0.0010*x^2, 0)}
f5 <- function(x){ifelse(x>0, 0.6394+0.9114*x+0.0010*x^2, 0)}
f6 <- function(x){ifelse(x>0, 0.5564+0.9931*x+0.0010*x^2, 0)}
f7 <- function(x){ifelse(x>0, 0.9492+1.0683*x+0.0010*x^2, 0)}
f8 <- function(x){ifelse(x>0, 2.3430+0.5102*x+0.0010*x^2, 0)}
f9 <- function(x){ifelse(x>0, 1.8956+0.7345*x+0.0010*x^2, 0)}
f10 <- function(x){ifelse(x>0, 1.8750+0.6203*x+0.0010*x^2, 0)}
```

```
f11 <- function(x){ifelse(x>0, 1.8750+0.6203*x+0.0010*x^2, 0)}#2013
    values
for(j in 1:11){#calibrated values instead of original values
  meanlumhot[[j]]<-sapply(meanlumhot[[j]],mean,na.rm=TRUE)
  meanlumhot[[j]]<-as.data.frame(sapply(meanlumhot[[j]],paste("f",j,sep=
      "")))
  meanlumhot[[j]][meanlumhot[[j]] > 63] <- 63 #truncate if values are
      higher than 63
  meanlumhot[[j]]<-as.data.frame(meanlumhot[[j]])
  meanlumhot[[j]]$N<-rownames(meanlumhot[[j]])
  meanlumhot[[j]]$Year<-2002+j
  colnames(meanlumhot[[j]])<-c("meanlum","N","Year")
}
#diffusion areas
#extract values of luminosity values within hotspot of terrorism
meanlumdiff<-list()
rm(j)
for(j in 1:11){#computationally intense...
  meanlumdiff[[j]]<-extract(lum[[j]],obsdiff[[j]])
}

for(j in 1:11){#calibrated values instead of original values
  meanlumdiff[[j]]<-sapply(meanlumdiff[[j]],mean,na.rm=TRUE)
  meanlumdiff[[j]]<-as.data.frame(sapply(meanlumdiff[[j]],paste("f",j,
      sep="")))
  meanlumdiff[[j]][meanlumdiff[[j]] > 63] <- 63 #truncate if values are
      higher than 63
  meanlumdiff[[j]]<-as.data.frame(meanlumdiff[[j]])
  meanlumdiff[[j]]$N<-rownames(meanlumdiff[[j]])
  meanlumdiff[[j]]$Year<-2002+j
  colnames(meanlumdiff[[j]])<-c("meanlum","N","Year")
}
#merging mean and sd values in hotspot and diffusion areas
lumdiffusion<-list()
for(j in 1:11){#put data into dataframe and add variables for merging
    issues
  lumdiffusion[[j]]<-merge(meanlumdiff[[j]],meanlumhot[[j]],by=c("Year",
      "N"))
  colnames(lumdiffusion[[j]])<-c("Year","N","meanlumdiff","meanlumhot")
}
```

```
#merge lum values with observed diffusion values
lumfinal<-list()
for(j in 1:11){#put data into dataframe and add variables for merging
    issues
  lumfinal[[j]]<-merge(lumdiffusion[[j]],obsdiff[[j]],by=c("N"),all.x=
      TRUE)
}
#put all df together
lumfinal<-rbind(lumfinal[[1]],lumfinal[[2]],lumfinal[[3]],lumfinal[[4]],
    lumfinal[[5]],lumfinal[[6]],
                lumfinal[[7]],lumfinal[[8]],lumfinal[[9]],lumfinal
                    [[10]],lumfinal[[11]])
lumfinal$Year<-as.integer(lumfinal$Year)
lumfinal<-lumfinal[c(-6,-8,-10)]#keep important variables
lumfinal<-lumfinal[complete.cases(lumfinal),]#delete possible NA values
#separate diffusion from retraction observations
lumfinal$diffusion<-ifelse(lumfinal$deltadiff>=1.1,"diffusion",ifelse(
    lumfinal$deltadiff<= 0.9,"retraction","N/D"))
#OPTION:KEEP ONLY DIFFUSION OR RETRACTION IF ABSOLUTE VALUE > 10%
#lumfinal<-lumfinal[- grep("N/D", lumfinal$diffusion),]

#testing difference in lum between hotspots that diffuse and their
    neighbourhood (lum hot that diff. > their neigb p-value = 0.006061)
wilcox.test(lumfinal[lumfinal$diffusion=="diffusion",]$meanlumdiff,
    lumfinal[lumfinal$diffusion=="diffusion",]$meanlumhot,alternative = c
    ("less"),paired=TRUE)#less means x<y
wilcox.test(lumfinal[lumfinal$diffusion=="diffusion",]$meanlumdiff,
    lumfinal[lumfinal$diffusion=="diffusion",]$meanlumhot,alternative = c
    ("greater"),paired=TRUE)#greater means x>y
#testing difference in lum from hotspots with diffusion vs hotspots
    without diffusion areas (not sig.)
wilcox.test(lumfinal[lumfinal$diffusion=="diffusion",]$meanlumhot,
    lumfinal[lumfinal$diffusion!="diffusion",]$meanlumhot,alternative = c
    ("less"),exact=FALSE)#less means x<y
wilcox.test(lumfinal[lumfinal$diffusion=="diffusion",]$meanlumhot,
    lumfinal[lumfinal$diffusion!="diffusion",]$meanlumhot,alternative = c
    ("greater"),exact=FALSE)#less means x<y
#testing difference in lum between diffusion areas in hotspots and
    hotspots without diffusion
```

```
wilcox.test(lumfinal[lumfinal$diffusion=="diffusion",]$meanlumdiff,
    lumfinal[lumfinal$diffusion!="diffusion",]$meanlumhot,alternative = c
    ("less"),exact=FALSE)#less means x<y
wilcox.test(lumfinal[lumfinal$diffusion=="diffusion",]$meanlumdiff,
    lumfinal[lumfinal$diffusion!="diffusion",]$meanlumhot,alternative = c
    ("greater"),exact=FALSE)#less means x<y
#testing difference in lum between hotspots that do not diffuse and
    their neighbourhood (lum hot with diff > their neigb p-value =
    0.006061)
wilcox.test(lumfinal[lumfinal$diffusion!="diffusion",]$meanlumdiff,
    lumfinal[lumfinal$diffusion!="diffusion",]$meanlumhot,alternative = c
    ("less"),paired=TRUE)#less means x<y
wilcox.test(lumfinal[lumfinal$diffusion!="diffusion",]$meanlumdiff,
    lumfinal[lumfinal$diffusion!="diffusion",]$meanlumhot,alternative = c
    ("greater"),paired=TRUE)#greater means x>y
#comparing hotspots mean lum with other locations where terrorism
    occurred
GTDlummean<- -0.005436653#from nb events Poisson (average where GTD
    taken)
#one sample student t.test
t.test(lumfinal$meanlumhot, mu=GTDlummean) #p-value = 0.0004399
#comparing diffusion mean lum with world mean lum
#one sample student t.test
t.test(lumfinal$meanlumdiff, mu=GTDlummean)

a=lumfinal[lumfinal$diffusion=="diffusion",]$meanlumdiff
b=lumfinal[lumfinal$diffusion=="diffusion",]$meanlumhot
c=lumfinal[lumfinal$diffusion=="N/D",]$meanlumhot
# Make a list of these 2 vectors
C=list(a,b,c)
names(C)=c(paste("Diffusion\n " , "area" , sep=""), paste("Hotspot\n " ,
     "with diffusion" , sep=""), paste("Hotspot\n " , "without diffusion"
     , sep=""))
#I change the mgp argument to avoid the text overlays the x axis
plot.new()
mypath=paste("C:/Users/apython/Documents/Andre/Education/StAndrews/PhD/
    revision/revisedPhDthesis/Chapter6/Figs/Raster/plumschema.png",sep=""
    )
png(file=mypath,width=900,height=720)
```

```
par(cex=2.4,mar=c(3, 4, 0.2,1),mgp=c(3,2,0))#mar: c(bottom,left,top,
    right)
boxplot(C, main=NA,outcex=1,
        xlab=NA, ylab="(standardised) luminosity ",ylim=c(-1,8))
abline(h = GTDlummean, col = "black",lwd=2,lty=2)
text(x=2.7, y = -0.3, labels = "World average (standardised) luminosity"
    ,
      pos = NULL, offset = 0.5, vfont = NULL,
      cex = 0.75, col = NULL, font = NULL)
text(x=3.0, y = 7.9, labels = "Poisson model",
      pos = NULL, offset = 0.5, vfont = NULL,
      cex = 0.75, col = NULL, font = NULL)
dev.off();dev.off()

save.image("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/
    pdiffusion_GDP.RData")
#load("~/Andre/Education/StAndrews/PhD/PhD_R/SPDE/STpoisson/pdiffusion_
    GDP.RData")

#end
```