Analyzing space-time satellite data for disease ecology applications with GRASS GIS and R stats

Verónica Andreo

OpenGeoHub Summer School 2019, Münster

ABOUT ME

PhD in Biology

MSc in Remote Sensing and GIS

Researcher @ Argentinean Space Agency

Applications of RS & GIS for disease ecology

Keywords: RS, GIS, Time series, SDM,
Disease Ecology, Rodents, Hantavirus

GRASS GIS Dev Team

OSGeo Charter member

FOSS4G enthusiast and advocate

veroandreo.gitlab.io

veroandreo @VeronicaAndreo

veroandreo@gmail.com

GRASS and R for disease ecology

rgrass7

- `initGRASS()`: starts a GRASS GIS session from R
- `execGRASS()`: executes GRASS GIS commands
- `gmeta()`: shows GRASS location metadata
- `readVECT()` and `readRAST()`: read vector and raster maps from GRASS into *sf/sp* objects
- `writeVECT()` and `writeRAST()`: write *sf/sp* objects into GRASS GIS database

An update to support stars for GRASS raster maps is on the way! Kudos to Roger Bivand!

GRASS GIS and R can be used together in two ways:

- Using R within a GRASS GIS session
- Using GRASS GIS within an R session

Details and examples at the GRASS and R wiki

- Using **R within a GRASS GIS session**, i.e. starting R (or RStudio) from the GRASS GIS terminal

  - we do not need to initialize GRASS with `initGRASS()`, just type `R` or `rstudio &` in the GRASS GIS terminal
  - we access GRASS GIS funtionalities and database through `execGRASS()`
  - we use `readVECT()`, `readRAST()` to read data from GRASS DB to do analysis and plots in R
  - we write data (back) to GRASS GIS database with `writeVECT()` and `writeRAST()`

- Using **GRASS GIS within an R session**, i.e. we connect to GRASS GIS database from within R (or RStudio).

  - we need to start GRASS GIS with `initGRASS()` (usually by creating a throw away location)
  - we access GRASS GIS funtionalities through `execGRASS()` (usually to apply them on data outside GRASS DB)

For more detailed examples check:

- R and GRASS presentation
- Example of GRASS - R for raster time series wiki

link2GI

See the vignette on how to set GRASS database with link2GI for further details

Hands-on to space-time analysis for disease ecology with GRASS GIS and R

Overview

- Import species records
- Create random background points
- Create environmental layers
- Read data into R
- Model species distribution
- Model evaluation
- Visualization of results

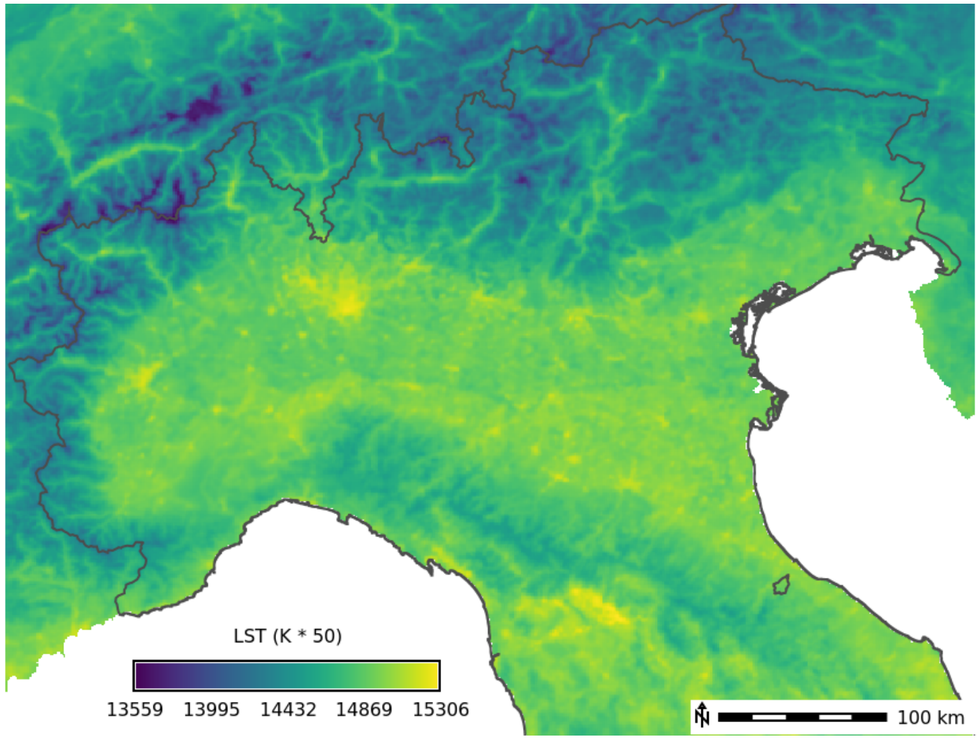
Check Analyzing space-time satellite data with GRASS GIS for environmental monitoring presentation for an intro to time series in GRASS GIS.

Data for the session

- Records of *Aedes albopictus* (Asian tiger mosquito) in Northern Italy from GBIF
- Environmental layers derived from reconstructed daily MODIS LST products

Data for the session: LST

- MODIS LST reconstructed by mundialis based on Metz et al. 2017
- Daily average LST
- 1 km spatial resolution
- Converted to Celsius degrees



LST, July 2018 - Northern Italy.

get sample location and code

- eu_laea location with LST mapset: download and unzip within your `grassdata` folder
- GRASS script
- R script

Let's start GRASS GIS! 

Importing species records and creating random background points

```
#!/bin/bash
############################################################################
# GRASS GIS commands for the session:
# "Analysis of space-time satellite data for disease ecology
# applications with GRASS GIS and R stats" at OpenGeoHub Summer School,
# Muenster (Germany)
#
# Modelling Aedes albopictus potential distribution in Northern Italy
#
# Author: Veronica Andreo
#
# Date: August, 2019
############################################################################



#
# Download data from GBIF for Aedes albopictus in Northern Italy
#



# Set computational region
```

Importing species records and creating random background points

```
# Date: August, 2019
##############################################################################



#
# Download data from GBIF for Aedes albopictus in Northern Italy
#



# Set computational region
g.region -p raster=lst_2014.001_avg

# Install extension (requires pygbif: pip install pygbif)
g.extension extension=v.in.pygbif


# Import data from GBIF
v.in.pygbif output=aedes_albopictus \
  taxa="Aedes albopictus" \
  date_from="2014-01-01" \
  date_to="2018-12-31"
```

Importing species records and creating random background points

```
#
# Download data from GBIF for Aedes albopictus in Northern Italy
#



# Set computational region
g.region -p raster=lst_2014.001_avg

# Install extension (requires pygbif: pip install pygbif)
g.extension extension=v.in.pygbif

# Import data from GBIF
v.in.pygbif output=aedes_albopictus \
  taxa="Aedes albopictus" \
  date_from="2014-01-01" \
  date_to="2018-12-31"


#
```

Importing species records and creating random background points

```
# Set computational region
g.region -p raster=lst_2014.001_avg

# Install extension (requires pygbif: pip install pygbif)
g.extension extension=v.in.pygbif

# Import data from GBIF
v.in.pygbif output=aedes_albopictus \
  taxa="Aedes albopictus" \
  date_from="2014-01-01" \
  date_to="2018-12-31"


#
# Create background points
#


# Create buffer around Aedes albopictus records
```

Importing species records and creating random background points

```
  date_from="2014-01-01" \
  date_to="2018-12-31"



#
# Create background points
#



# Create buffer around Aedes albopictus records
v.buffer input=aedes_albopictus \
  output=aedes_buffer \
  distance=1000

# Create a vector mask to limit background points
r.mapcalc \
  expression="rast_mask = if(lst_2014.001_avg, 1, null())"

r.to.vect input=rast_mask \
  output=vect_mask \
  type=area
```

Importing species records and creating random background points

```
# Create buffer around Aedes albopictus records
v.buffer input=aedes_albopictus \
  output=aedes_buffer \
  distance=1000

# Create a vector mask to limit background points
r.mapcalc \
  expression="rast_mask = if(lst_2014.001_avg, 1, null())"

r.to.vect input=rast_mask \
  output=vect_mask \
  type=area

# Subtract buffers from vector mask
v.overlay ainput=vect_mask \
  binput=aedes_buffer \
  operator=xor \
  output=mask_background
```

Importing species records and creating random background points

```
# Create a vector mask to limit background points
r.mapcalc \
  expression="rast_mask = if(lst_2014.001_avg, 1, null())"

r.to.vect input=rast_mask \
  output=vect_mask \
  type=area

# Subtract buffers from vector mask
v.overlay ainput=vect_mask \
  binput=aedes_buffer \
  operator=xor \
  output=mask_background

# Generate random background points
v.random output=background_points \

  npoints=1000 \
  restrict=mask_background \
  seed=3749
```

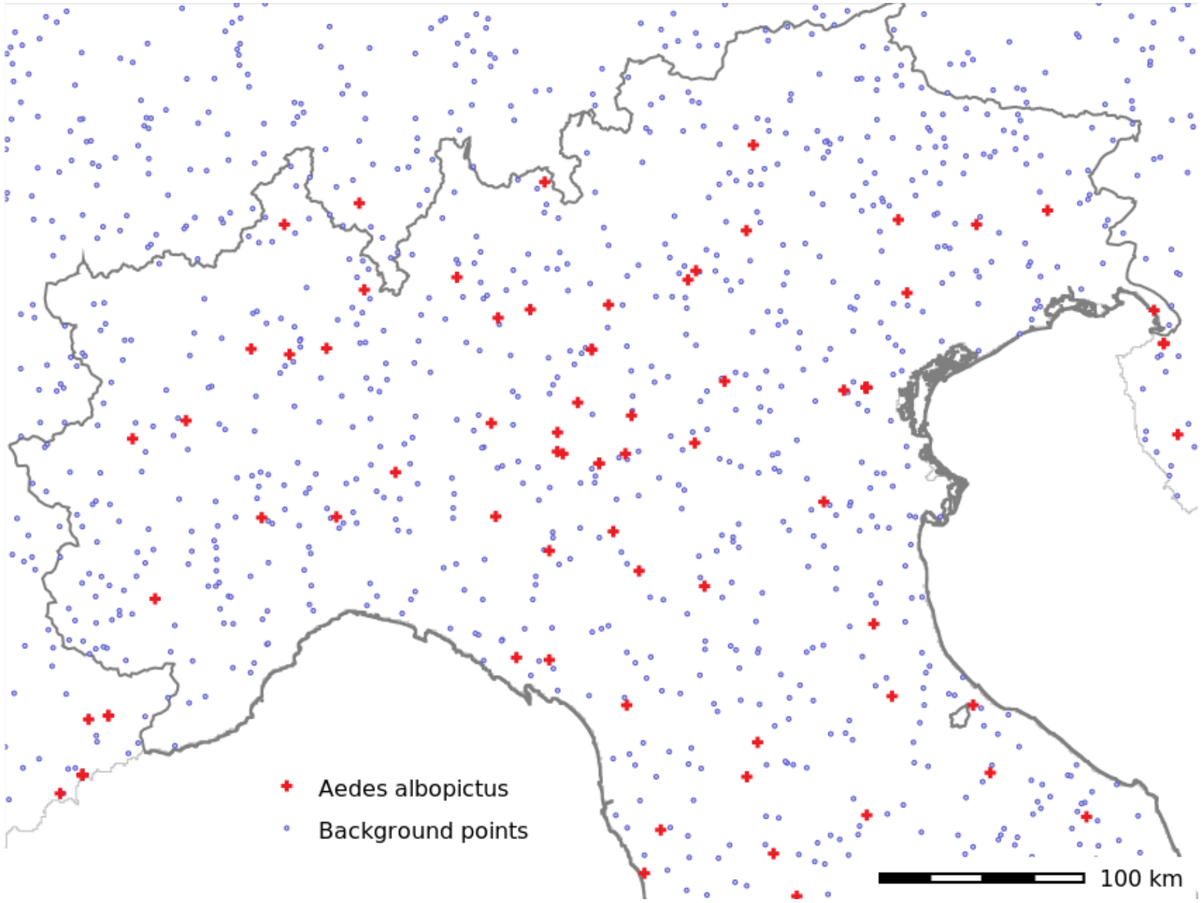
Importing species records and creating random background points

```
  type=area

# Subtract buffers from vector mask
v.overlay ainput=vect_mask \
  binput=aedes_buffer \
  operator=xor \
  output=mask_background

# Generate random background points
v.random output=background_points \
  npoints=1000 \
  restrict=mask_background \
  seed=3749


#
# Create daily LST STRDS
#
```


Aedes albopictus
Background points
100 km

Create daily LST STRDS

```
#!/bin/bash
############################################################################
# GRASS GIS commands for the session:
# "Analysis of space-time satellite data for disease ecology
# applications with GRASS GIS and R stats" at OpenGeoHub Summer School,
# Muenster (Germany)
#
# Modelling Aedes albopictus potential distribution in Northern Italy
#
# Author: Veronica Andreo
#
# Date: August, 2019
############################################################################



#
# Download data from GBIF for Aedes albopictus in Northern Italy
#



# Set computational region
```

Create daily LST STRDS

```
  restrict=mask_background \
  seed=3749



#
# Create daily LST STRDS
#



# Create time series
t.create type=strds \
  temporaltype=absolute \
  output=lst_daily \
  title="Average Daily LST" \
  description="Average daily LST in degree C - 2014-2018"

# Get list of maps
g.list type=raster \
  pattern="lst_201*" \
  output=list_lst.csv
```

Create daily LST STRDS

```
#


# Create time series
t.create type=strds \
  temporaltype=absolute \
  output=lst_daily \
  title="Average Daily LST" \
  description="Average daily LST in degree C - 2014-2018"

# Get list of maps
g.list type=raster \
  pattern="lst_201*" \
  output=list_lst.csv

# Register maps in strds
t.register -i input=lst_daily \
  file=list_lst.csv \
  increment="1 days" \
  start="2014-01-01"
```

Create daily LST STRDS

```
  temporaltype=absolute \
  output=lst_daily \
  title="Average Daily LST" \
  description="Average daily LST in degree C - 2014-2018"

# Get list of maps
g.list type=raster \
  pattern="lst_201*" \
  output=list_lst.csv

# Register maps in strds
t.register -i input=lst_daily \
  file=list_lst.csv \
  increment="1 days" \
  start="2014-01-01"

# Get info about the strds
t.info input=lst_daily


#
```

Create daily LST STRDS

```
# Get list of maps
g.list type=raster \
  pattern="lst_201*" \
  output=list_lst.csv

# Register maps in strds
t.register -i input=lst_daily \
  file=list_lst.csv \
  increment="1 days" \
  start="2014-01-01"

# Get info about the strds
t.info input=lst_daily


#

# Generate environmental variables from LST STRDS
#
```

Generate environmental variables from LST STRDS

```
#!/bin/bash
############################################################
# GRASS GIS commands for the session:
# "Analysis of space-time satellite data for disease ecology
# applications with GRASS GIS and R stats" at OpenGeoHub Summer School,
# Muenster (Germany)
#
# Modelling Aedes albopictus potential distribution in Northern Italy
#
# Author: Veronica Andreo
#
# Date: August, 2019
############################################################



#
# Download data from GBIF for Aedes albopictus in Northern Italy
#



# Set computational region
```

Generate environmental variables from LST STRDS

```
# Long term monthly avg, min and max LST
for i in $(seq -w 1 12) ; do

  # average
  t.rast.series input=lst_daily \
    method=average \
    where="strftime('%m', start_time)='${i}'" \
    output=lst_average_${i}

  # minimum
  t.rast.series input=lst_daily \
    method=minimum \
    where="strftime('%m', start_time)='${i}'" \
    output=lst_minimum_${i}

  # maximum
  t.rast.series input=lst_daily \
    method=maximum \
    where="strftime('%m', start_time)='${i}'" \
    output=lst_maximum_${i}
```

Generate environmental variables from LST STRDS

```
    method=minimum \
    where="strftime('%m', start_time)='${i}'" \
    output=lst_minimum_${i}

  # maximum
  t.rast.series input=lst_daily \
    method=maximum \
    where="strftime('%m', start_time)='${i}'" \
    output=lst_maximum_${i}

done

# Install extension
g.extension extension=r.bioclim

# Estimate temperature related bioclimatic variables

r.bioclim \
  tmin=$(g.list type=raster pattern="lst_minimum_??" separator=",") \
  tmax=$(g.list type=raster pattern="lst_maximum_??" separator=",") \
  tavg=$(g.list type=raster pattern="lst_average_??" separator=",") \
```

Generate environmental variables from LST STRDS

```
  t.rast.series input=lst_daily \
    method=maximum \
    where="strftime('%m', start_time)='${i}'" \
    output=lst_maximum_${i}

done

# Install extension
g.extension extension=r.bioclim

# Estimate temperature related bioclimatic variables

r.bioclim \
  tmin=$(g.list type=raster pattern="lst_minimum_??" separator=",") \
  tmax=$(g.list type=raster pattern="lst_maximum_??" separator=",") \
  tavg=$(g.list type=raster pattern="lst_average_??" separator=",") \
  output=worldclim_


## Spring warming
```

Generate environmental variables from LST STRDS

```
r.bioclim \
  tmin=$(g.list type=raster pattern="lst_minimum_??" separator=",") \
  tmax=$(g.list type=raster pattern="lst_maximum_??" separator=",") \
  tavg=$(g.list type=raster pattern="lst_average_??" separator=",") \
  output=worldclim_


## Spring warming

# Annual spring warming: slope(daily Tmean february-march-april)
t.rast.aggregate input=lst_daily \
  output=annual_spring_warming \
  basename=spring_warming \
  suffix=gran \
  method=slope \
  granularity="1 years" \
  where="strftime('%m',start_time)='02' or \
         strftime('%m',start_time)='03' or \
         strftime('%m', start_time)='04'"
```

Generate environmental variables from LST STRDS

```
# Annual spring warming: slope(daily Tmean february=march=april)
t.rast.aggregate input=lst_daily \
  output=annual_spring_warming \
  basename=spring_warming \
  suffix=gran \
  method=slope \
  granularity="1 years" \
  where="strftime('%m',start_time)='02' or \
         strftime('%m',start_time)='03' or \
         strftime('%m', start_time)='04'"

# Average spring warming
t.rast.series input=annual_spring_warming \
  output=avg_spring_warming \
  method=average


## Autumnal cooling

# Annual autumnal cooling: slope(daily Tmean august=september=october)
```

Generate environmental variables from LST STRDS

```
          strftime('%m', start_time)='04'"

# Average spring warming
t.rast.series input=annual_spring_warming \
  output=avg_spring_warming \
  method=average


## Autumnal cooling

# Annual autumnal cooling: slope(daily Tmean august-september-october)
t.rast.aggregate input=lst_daily \
  output=annual_autumnal_cooling \
  basename=autumnal_cooling \
  suffix=gran \

  method=slope \
  granularity="1 years" \
  where="strftime('%m',start_time)='08' or \
         strftime('%m',start_time)='09' or \
         strftime('%m', start_time)='10'"
```

Generate environmental variables from LST STRDS

```
# Annual autumnal cooling: slope(daily Tmean august=september=october)
t.rast.aggregate input=lst_daily \
  output=annual_autumnal_cooling \
  basename=autumnal_cooling \
  suffix=gran \

  method=slope \
  granularity="1 years" \
  where="strftime('%m',start_time)='08' or \
         strftime('%m',start_time)='09' or \
         strftime('%m', start_time)='10'"


# Average autumnal cooling
t.rast.series input=annual_autumnal_cooling \
  output=avg_autumnal_cooling \
  method=average



## Start/end and length of mosquito growing season
```

Generate environmental variables from LST STRDS

```
  method=slope \
  granularity="1 years" \
  where="strftime('%m',start_time)='08' or \
         strftime('%m',start_time)='09' or \
         strftime('%m', start_time)='10'"

# Average autumnal cooling
t.rast.series input=annual_autumnal_cooling \
  output=avg_autumnal_cooling \
  method=average


## Start/end and length of mosquito growing season

# Install extension
g.extension extension=r.seasons

# Detect seasons
for YEAR in $(seq 2014 2018) ; do
```

Generate environmental variables from LST STRDS

```
t.rast.series input=annual_autumnal_cooling \
  output=avg_autumnal_cooling \
  method=average



## Start/end and length of mosquito growing season


# Install extension
g.extension extension=r.seasons

# Detect seasons
for YEAR in $(seq 2014 2018) ; do

  # Get map list per year
  t.rast.list -u lst_daily \
    columns=name \
    where="strftime('%Y',start_time)='${YEAR}'" \
    output=list_${YEAR}.txt

  # Mosquito season (threshold: 10C, min duration: 150 days)
  r.seasons file=list_${YEAR}.txt \
```

Generate environmental variables from LST STRDS

```
g.extension extension=r.seasons

# Detect seasons
for YEAR in $(seq 2014 2018) ; do

  # Get map list per year
  t.rast.list -u lst_daily \
    columns=name \
    where="strftime('%Y',start_time)='${YEAR}'" \
    output=list_${YEAR}.txt

  # Mosquito season (threshold: 10C, min duration: 150 days)
  r.seasons file=list_${YEAR}.txt \
    prefix=mosq_season_${YEAR}_ \
    n=1 \
    nout=mosq_season_${YEAR} \
    threshold_value=10 \
    min_length=150 \

    max_gap=12

  # Length core season
```

Generate environmental variables from LST STRDS

```
  # Mosquito season (threshold: 10C, min duration: 150 days)
  r.seasons file=list_${YEAR}.txt \
    prefix=mosq_season_${YEAR}_ \
    n=1 \
    nout=mosq_season_${YEAR} \
    threshold_value=10 \
    min_length=150 \

    max_gap=12


  # Length core season
  r.mapcalc expression="mosq_season_length_${YEAR}_1 = \
    mosq_season_${YEAR}_1_end1 - mosq_season_${YEAR}_1_start1 + 1"

  # Length extended season
  r.mapcalc expression="mosq_season_length_${YEAR}_2 = \
    mosq_season_${YEAR}_1_end2 - mosq_season_${YEAR}_1_start2 + 1"

done
```

Generate environmental variables from LST STRDS

```
  # Length core season
  r.mapcalc expression="mosq_season_length_${YEAR}_1 = \
    mosq_season_${YEAR}_1_end1 - mosq_season_${YEAR}_1_start1 + 1"

  # Length extended season
  r.mapcalc expression="mosq_season_length_${YEAR}_2 = \
    mosq_season_${YEAR}_1_end2 - mosq_season_${YEAR}_1_start2 + 1"

done

# Average length of mosquito season
r.series \
  input=$(g.list type=raster pattern=mosq_season_length*1 separator=",") \
  output=avg_mosq_season_1_length \
  method=average

r.series \
  input=$(g.list type=raster pattern=mosq_season_length*2 separator=",") \
  output=avg_mosq_season_2_length \
  method=average
```

Generate environmental variables from LST STRDS

```
r.series \
  input=$(g.list type=raster pattern=mosq_season_length*1 separator=",") \
  output=avg_mosq_season_1_length \
  method=average

r.series \
  input=$(g.list type=raster pattern=mosq_season_length*2 separator=",") \
  output=avg_mosq_season_2_length \
  method=average


## Number of days with LSTmean >= 20 and <= 30

# Keep only pixels meeting the condition
t.rast.algebra -n \
  expression="tmean_higher20_lower30 = if(lst_daily >= 20.0 && \
  lst_daily <= 30.0, 1, null())" \
  basename=tmean_higher20_lower30 \
  suffix=gran
```

Generate environmental variables from LST STRDS

```
  method=average


## Number of days with LSTmean >= 20 and <= 30

# Keep only pixels meeting the condition
t.rast.algebra -n \
  expression="tmean_higher20_lower30 = if(lst_daily >= 20.0 && \
  lst_daily <= 30.0, 1, null())" \
  basename=tmean_higher20_lower30 \
  suffix=gran


# Count how many times per year the condition is met
t.rast.aggregate input=tmean_higher20_lower30 \
  output=count_tmean_higher20_lower30 \
  basename=tmean_higher20_lower30 \
  suffix=gran \
  method=count \
  granularity="1 years"

  t.rast.colors
```

Generate environmental variables from LST STRDS

```
  expression="tmean_higher20_lower30 = if(lst_daily >= 20.0 && \
  lst_daily <= 30.0, 1, null())" \
  basename=tmean_higher20_lower30 \
  suffix=gran


# Count how many times per year the condition is met
t.rast.aggregate input=tmean_higher20_lower30 \
  output=count_tmean_higher20_lower30 \
  basename=tmean_higher20_lower30 \
  suffix=gran \
  method=count \
  granularity="1 years"

# Inspect values
t.rast.list \
  count_tmean_higher20_lower30 \
  columns=name,start_time,end_time,min,max

# Average number of days with LSTmean >= 20 and <= 30
t.rast.series input=count_tmean_higher20_lower30 \
```

Generate environmental variables from LST STRDS

```
# Count how many times per year the condition is met
t.rast.aggregate input=tmean_higher20_lower30 \
  output=count_tmean_higher20_lower30 \
  basename=tmean_higher20_lower30 \
  suffix=gran \
  method=count \
  granularity="1 years"

# Inspect values
t.rast.list \
  count_tmean_higher20_lower30 \
  columns=name,start_time,end_time,min,max

# Average number of days with LSTmean >= 20 and <= 30
t.rast.series input=count_tmean_higher20_lower30 \
  output=avg_count_tmean_higher20_lower30 \
  method=average


## Number of consecutive days with LSTmean >= 33
```

Generate environmental variables from LST STRDS

```
# Inspect values
t.rast.list \
  count_tmean_higher20_lower30 \
  columns=name,start_time,end_time,min,max

# Average number of days with LSTmean >= 20 and <= 30
t.rast.series input=count_tmean_higher20_lower30 \
  output=avg_count_tmean_higher20_lower30 \
  method=average


## Number of consecutive days with LSTmean >= 33

# Create annual mask
t.rast.aggregate input=lst_daily \
  output=annual_mask \
  basename=annual_mask \
  suffix=gran \
  granularity="1 year" \
  method=count
```

Generate environmental variables from LST STRDS

```
  method=average


## Number of consecutive days with LSTmean >= 33

# Create annual mask
t.rast.aggregate input=lst_daily \
  output=annual_mask \
  basename=annual_mask \
  suffix=gran \
  granularity="1 year" \
  method=count

# Replace values by zero
t.rast.mapcalc \
  input=annual_mask \
  output=annual_mask_0 \
  expression="if(annual_mask, 0)" \
  basename=annual_mask_0
```

Generate environmental variables from LST STRDS

```
  output=annual_mask \
  basename=annual_mask \
  suffix=gran \
  granularity="1 year" \
  method=count

# Replace values by zero
t.rast.mapcalc \
  input=annual_mask \
  output=annual_mask_0 \

  expression="if(annual_mask, 0)" \
  basename=annual_mask_0

# Calculate consecutive days with LSTmean >= 33
t.rast.algebra \
  expression="higher35_consec_days = annual_mask_0 {+,contains,l} \
  if(lst_daily >= 33.0 && lst_daily[-1] >= 33.0 || \
  lst_daily[1] >= 33.0 && lst_daily >= 33.0, 1, 0)" \
  basename=higher35_days \
  suffix=gran \
  nproc=7
```

Generate environmental variables from LST STRDS

```
t.rast.mapcalc \
  input=annual_mask \
  output=annual_mask_0 \

  expression="if(annual_mask, 0)" \
  basename=annual_mask_0

# Calculate consecutive days with LSTmean >= 33
t.rast.algebra \
  expression="higher35_consec_days = annual_mask_0 {+,contains,l} \
  if(lst_daily >= 33.0 && lst_daily[-1] >= 33.0 || \
  lst_daily[1] >= 33.0 && lst_daily >= 33.0, 1, 0)" \
  basename=higher35_days \
  suffix=gran \
  nproc=7

# Inspect values
t.rast.list input=higher35_consec_days \
  columns=name,start_time,end_time,min,max

# Median number of consecutive days with LSTmean >= 33
```

Generate environmental variables from LST STRDS

```
  expression="if(annual_mask, 0)" \
  basename=annual_mask_0

# Calculate consecutive days with LSTmean >= 33
t.rast.algebra \
  expression="higher35_consec_days = annual_mask_0 {+,contains,l} \
  if(lst_daily >= 33.0 && lst_daily[-1] >= 33.0 || \
  lst_daily[1] >= 33.0 && lst_daily >= 33.0, 1, 0)" \
  basename=higher35_days \
  suffix=gran \
  nproc=7

# Inspect values
t.rast.list input=higher35_consec_days \
  columns=name,start_time,end_time,min,max

# Median number of consecutive days with LSTmean >= 33
t.rast.series input=higher35_consec_days \
  output=median_higher35_consec_days \
  method=median
```

Generate environmental variables from LST STRDS

```
  suffix=gran \
  nproc=7

# Inspect values
t.rast.list input=higher35_consec_days \
  columns=name,start_time,end_time,min,max

# Median number of consecutive days with LSTmean >= 33
t.rast.series input=higher35_consec_days \
  output=median_higher35_consec_days \
  method=median


## Growing Degree Days

# Accumulation of degree days
t.rast.accumulate -n input=lst_daily \
  output=mosq_daily_bedd \
  basename=mosq_daily_bedd \
  suffix=gran \
  start="2014-03-01" stop="2018-10-01" \
```

Generate environmental variables from LST STRDS

```
  output=median_higher35_consec_days \
  method=median


## Growing Degree Days

# Accumulation of degree days
t.rast.accumulate -n input=lst_daily \
  output=mosq_daily_bedd \
  basename=mosq_daily_bedd \
  suffix=gran \
  start="2014-03-01" stop="2018-10-01" \
  cycle="7 months" offset="5 months" \
  method=bedd limits=11,30


# Get basic info
t.info input=mosq_daily_bedd

# Inspect values
t.rast.list input=mosq_daily_bedd \
  columns=name,start_time,end_time,min,max
```

Generate environmental variables from LST STRDS

```
  basename=mosq_daily_bedd \
  suffix=gran \
  start="2014-03-01" stop="2018-10-01" \
  cycle="7 months" offset="5 months" \
  method=bedd limits=11,30


# Get basic info
t.info input=mosq_daily_bedd

# Inspect values
t.rast.list input=mosq_daily_bedd \
  columns=name,start_time,end_time,min,max


## Detection of mosquito generations

# Arrays
cycle=($(seq 1 9))
cycle_beg=($(seq 1 300 2700))
cycle_end=($(seq 300 300 2700))
# Length of the array
```

Generate environmental variables from LST STRDS

```
  columns=name,start_time,end_time,min,max


## Detection of mosquito generations


# Arrays
cycle=($(seq 1 9))
cycle_beg=($(seq 1 300 2700))
cycle_end=($(seq 300 300 2700))
# Length of the array
count=${#cycle[@]}

for i in $(seq 1 $count) ; do

  echo "cycle: "${cycle[$i-1]}" - "${cycle_beg[$i-1]} ${cycle_end[$i-1]}

  # Identify generations
  t.rast.accdetect input=mosq_daily_bedd \
    occurrence=mosq_occurrence_cycle_${cycle[$i-1]} \
    indicator=mosq_indicator_cycle_${cycle[$i-1]} \
```

Generate environmental variables from LST STRDS

```
# Identify generations
t.rast.accdetect input=mosq_daily_bedd \
  occurrence=mosq_occurrence_cycle_${cycle[$i-1]} \
  indicator=mosq_indicator_cycle_${cycle[$i-1]} \
  basename=mosq_cycle_${cycle[$i-1]} \
  start="2014-03-01" stop="2018-10-01" \
  cycle="7 months" offset="5 months" \
  range=${cycle_beg[$i-1]},${cycle_end[$i-1]}

# Extract areas that have full generations

# Indicator takes values 1, 2 or 3 for start, middle and end of cycle

t.rast.aggregate input=mosq_indicator_cycle_${cycle[$i-1]} \
  output=mosq_cycle${cycle[$i-1]}_yearly \
  basename=mosq_c${cycle[$i-1]}_yearly \
  granularity="1 year" \
  method=maximum \
  suffix=gran
```

Generate environmental variables from LST STRDS

```
  granularity="1 year" \
  method=maximum \
  suffix=gran

# Keep only complete generations
t.rast.mapcalc input=mosq_cycle${cycle[$i-1]}_yearly \
  output=mosq_cycle${cycle[$i-1]}_yearly_clean \
  basename=mosq_clean_c${cycle[$i-1]} \
  expression="if(mosq_cycle${cycle[$i-1]}_yearly == 3, \
  ${cycle[$i-1]}, null())"

# Duration of each mosquito generation

# Beginning
t.rast.aggregate input=mosq_occurrence_cycle_${cycle[$i-1]} \
  output=mosq_min_day_cycle${cycle[$i-1]} \
  basename=occ_min_day_c${cycle[$i-1]} \
  method=minimum \
  granularity="1 year" \
  suffix=gran

# End
```

Generate environmental variables from LST STRDS

```
  output=mosq_min_day_cycle${cycle[$i-1]} \
  basename=occ_min_day_c${cycle[$i-1]} \
  method=minimum \
  granularity="1 year" \
  suffix=gran


# End
t.rast.aggregate input=mosq_occurrence_cycle_${cycle[$i-1]} \
  output=mosq_max_day_cycle${cycle[$i-1]} \
  basename=occ_max_day_c${cycle[$i-1]} \
  method=maximum \
  granularity="1 year" \
  suffix=gran


# Difference

t.rast.mapcalc \
  input=mosq_min_day_cycle${cycle[$i-1]},mosq_max_day_cycle${cycle[$i-1]} \
  output=mosq_duration_cycle${cycle[$i-1]} \
  basename=mosq_duration_cycle${cycle[$i-1]} \
```

Generate environmental variables from LST STRDS

```
    method=maximum \
    granularity="1 year" \
    suffix=gran

  # Difference

  t.rast.mapcalc \
    input=mosq_min_day_cycle${cycle[$i-1]},mosq_max_day_cycle${cycle[$i-1]} \
    output=mosq_duration_cycle${cycle[$i-1]} \
    basename=mosq_duration_cycle${cycle[$i-1]} \
    expression="mosq_max_day_cycle${cycle[$i-1]} - \
    mosq_min_day_cycle${cycle[$i-1]} + 1"

done


# Maximum number of generations per pixel per year
for i in $(seq 1 5) ; do
  g.list type=raster separator="," pattern="mosq_clean_c*_${i}"
  r.series \
    input=$(g.list type=raster pattern="mosq_clean_c*_${i}" separator=",") \
```

Generate environmental variables from LST STRDS

```
    input=mosq_min_day_cycle${cycle[$i-1]},mosq_max_day_cycle${cycle[$i-1]} \
    output=mosq_duration_cycle${cycle[$i-1]} \
    basename=mosq_duration_cycle${cycle[$i-1]} \
    expression="mosq_max_day_cycle${cycle[$i-1]} - \
    mosq_min_day_cycle${cycle[$i-1]} + 1"

done



# Maximum number of generations per pixel per year
for i in $(seq 1 5) ; do
  g.list type=raster separator="," pattern="mosq_clean_c*_${i}"
  r.series \
    input=$(g.list type=raster pattern="mosq_clean_c*_${i}" separator=",") \
    output=mosq_generations_${i} \
    method=maximum
done

# Median number of generations per pixel
r.series \
  input=$(g.list type=raster pattern="mosq_generations_*" separator=",") \
  output=median_mosq_generations \
```

Generate environmental variables from LST STRDS

```
# Maximum number of generations per pixel per year
for i in $(seq 1 5) ; do
  g.list type=raster separator="," pattern="mosq_clean_c*_${i}"
  r.series \
    input=$(g.list type=raster pattern="mosq_clean_c*_${i}" separator=",") \
    output=mosq_generations_${i} \
    method=maximum
done

# Median number of generations per pixel
r.series \
  input=$(g.list type=raster pattern="mosq_generations_*" separator=",") \
  output=median_mosq_generations \
  method=median

# Median duration of generations per year per pixel
for i in $(seq 1 5) ; do
  g.list type=raster separator="," pattern="mosq_duration_cycle*_${i}"
  r.series \
    input=$(g.list type="raster pattern=mosq_duration_cycle*_${i}" separator=",") \
```

Generate environmental variables from LST STRDS

```
    method=maximum
done

# Median number of generations per pixel
r.series \
  input=$(g.list type=raster pattern="mosq_generations_*" separator=",") \
  output=median_mosq_generations \
  method=median

# Median duration of generations per year per pixel
for i in $(seq 1 5) ; do
  g.list type=raster separator="," pattern="mosq_duration_cycle*_${i}"
  r.series \
    input=$(g.list type="raster pattern=mosq_duration_cycle*_${i}" separator=",") \
    output=mosq_generation_median_duration_${i} \
    method=median
done


# Median duration of generations per pixel
r.series \
  input=$(g.list type=raster pattern="mosq_generation_median_duration_*" separator=
```

Generate environmental variables from LST STRDS

```
done

# Median number of generations per pixel
r.series \
  input=$(g.list type=raster pattern="mosq_generations_*" separator=",") \
  output=median_mosq_generations \
  method=median

# Median duration of generations per year per pixel
for i in $(seq 1 5) ; do
  g.list type=raster separator="," pattern="mosq_duration_cycle*_${i}"
  r.series \
    input=$(g.list type="raster pattern=mosq_duration_cycle*_${i}" separator=",") \
    output=mosq_generation_median_duration_${i} \
    method=median
done


# Median duration of generations per pixel
r.series \
  input=$(g.list type=raster pattern="mosq_generation_median_duration_*" separator=
```

Within R...

```
#####################################################################
# R commands for the session:
# "Analysis of space-time satellite data for disease ecology
# applications with GRASS GIS and R stats" at OpenGeoHub Summer School,
# Muenster (Germany)
#
# Modelling Aedes albopictus potential distribution in Northern Italy
#
# Original example contributed by Carol Garzon Lopez
# Adapted by Veronica Andreo
#
# Date: October, 2018
# Updated: August, 2019
#####################################################################



#
# Install and load required packages
#
```

Within R...

```
#
# Install and load required packages
#


# Install packages
install.packages("rgrass7")
install.packages("raster")
install.packages("sf")
install.packages("mapview")
install.packages("biomod2")

# Load libraries
library(rgrass7)
library(raster)
library(sf)
library(mapview)
library(biomod2)
```

Within R...

```
# Install packages
install.packages("rgrass7")
install.packages("raster")
install.packages("sf")
install.packages("mapview")

install.packages("biomod2")

# Load libraries
library(rgrass7)
library(raster)
library(sf)
library(mapview)
library(biomod2)



#
# Read vector data
#
```

Within R...

```
library(raster)
library(sf)
library(mapview)
library(biomod2)


#
# Read vector data
#


# Use sf for vectors
use_sf()

# Read vector layers
Aa_pres <- readVECT("aedes_albopictus")
background <- readVECT("background_points")


#
# Read raster data
```

Within R...

```
#
# Read vector data
#



# Use sf for vectors
use_sf()

# Read vector layers
Aa_pres <- readVECT("aedes_albopictus")
background <- readVECT("background_points")


#
# Read raster data
#


# Use sp for rasters (stars support on the way)
```

Within R...

```
# Read vector layers
Aa_pres <- readVECT("aedes_albopictus")
background <- readVECT("background_points")


#
# Read raster data
#


# Use sp for rasters (stars support on the way)
use_sp()

# List rasters by pattern
worldclim <- execGRASS("g.list",
                       parameters = list(
                       type = "raster",
                       pattern = "worldclim*"))

avg <- execGRASS("g.list",
```

Within R...

```
# Use sp for rasters (stars support on the way)
use_sp()

# List rasters by pattern
worldclim <- execGRASS("g.list",
                       parameters = list(
                       type = "raster",
                       pattern = "worldclim*"))

avg <- execGRASS("g.list",
                 parameters = list(
                 type = "raster",
                 pattern = "avg*"))

median <- execGRASS("g.list",
                    parameters = list(
                    type = "raster",
                    pattern = "median*",
                    exclude = "*[1-5]"))
```

Within R...

```
                parameters = list(
                type = "raster",
                pattern = "avg*"))

median <- execGRASS("g.list",
                    parameters = list(
                    type = "raster",
                    pattern = "median*",
                    exclude = "*[1-5]"))


# Concatenate map lists
to_import <- c(attributes(worldclim)$resOut,
               attributes(avg)$resOut,
               attributes(median)$resOut)

# Read raster layers
predictors <- list()
for (i in to_import){
  predictors[i] <- raster(readRAST(i))
}
```

Within R...

```
                    parameters = list(
                    type = "raster",
                    pattern = "median*",
                    exclude = "*[1-5]"))

# Concatenate map lists
to_import <- c(attributes(worldclim)$resOut,
               attributes(avg)$resOut,
               attributes(median)$resOut)

# Read raster layers
predictors <- list()
for (i in to_import){
  predictors[i] <- raster(readRAST(i))
}

# Quick visualization in mapview
mapview(predictors[[3]]) + Aa_pres

#
```

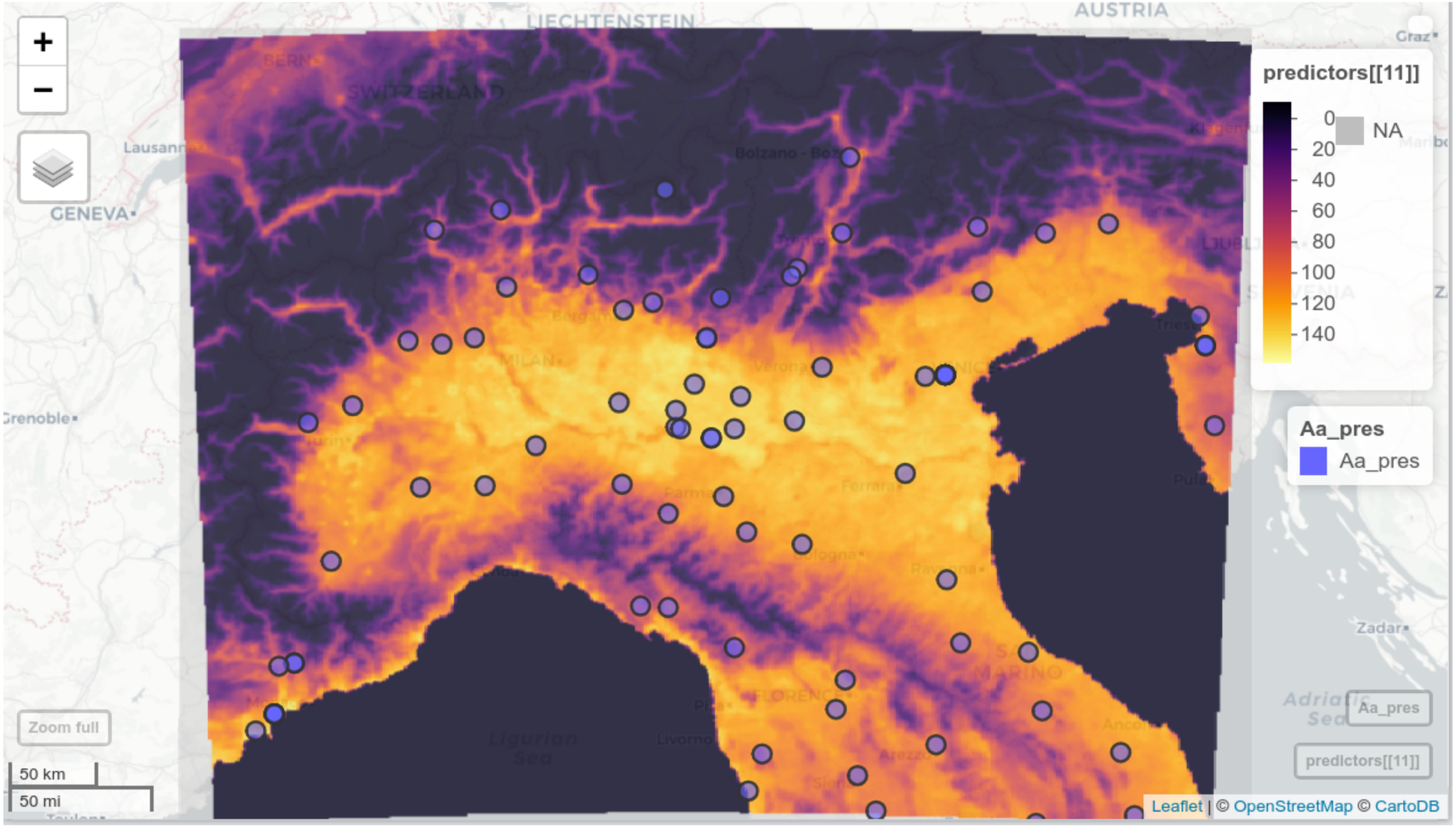
Within R...

```
# Concatenate map lists
to_import <- c(attributes(worldclim)$resOut,
               attributes(avg)$resOut,
               attributes(median)$resOut)

# Read raster layers
predictors <- list()
for (i in to_import){
  predictors[i] <- raster(readRAST(i))
}

# Quick visualization in mapview
mapview(predictors[[3]]) + Aa_pres


#
# Data preparation and formatting
#
```


predictors[[11]]
0
20
40
60
80
100
120
140
NA
Aa_pres
Aa_pres
Zoom full
50 km
50 mi
Aa_pres
predictors[[11]]
Leaflet | © OpenStreetMap © CartoDB

Data formatting

```
##############################################################################
# R commands for the session:
# "Analysis of space-time satellite data for disease ecology
# applications with GRASS GIS and R stats" at OpenGeoHub Summer School,
# Muenster (Germany)
#
# Modelling Aedes albopictus potential distribution in Northern Italy
#
# Original example contributed by Carol Garzon Lopez
# Adapted by Veronica Andreo
#
# Date: October, 2018
# Updated: August, 2019
##############################################################################



#
# Install and load required packages
#
```

Data formatting

```
#
# Data preparation and formatting
#



# Response variable
n_pres <- dim(Aa_pres)[1]
n_backg <- dim(background)[1]

myRespName <- 'Aedes.albopictus'

pres <- rep(1, n_pres)
backg <- rep(0, n_backg)
myResp <- c(pres, backg)

myRespXY <- rbind(st_coordinates(Aa_pres),
                  st_coordinates(background))


# Explanatory variables
```

Data formatting

```
myRespName <- 'Aedes.albopictus'

pres <- rep(1, n_pres)
backg <- rep(0, n_backg)
myResp <- c(pres, backg)

myRespXY <- rbind(st_coordinates(Aa_pres),
                  st_coordinates(background))


# Explanatory variables

# Stack raster layers
myExpl <- raster::stack(predictors)


# All together
myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                     expl.var = myExpl,
                                     resp.xy = myRespXY,
```

Data formatting

```
myRespXY <- rbind(st_coordinates(Aa_pres),
                  st_coordinates(background))


# Explanatory variables

# Stack raster layers
myExpl <- raster::stack(predictors)


# All together
myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                     expl.var = myExpl,
                                     resp.xy = myRespXY,
                                     resp.name = myRespName)


#
# Run model
#
```

MaxEnt model

```
##########################################################################
# R commands for the session:
# "Analysis of space-time satellite data for disease ecology
# applications with GRASS GIS and R stats" at OpenGeoHub Summer School,
# Muenster (Germany)
#
# Modelling Aedes albopictus potential distribution in Northern Italy
#
# Original example contributed by Carol Garzon Lopez
# Adapted by Veronica Andreo
#
# Date: October, 2018
# Updated: August, 2019
##########################################################################



#
# Install and load required packages
#
```

MaxEnt model

```
                                    resp.xy = myRespXY,
                                    resp.name = myRespName)



#
# Run model
#



# Set options
myBiomodOption <- BIOMOD_ModelingOptions(
  MAXENT.Phillips =
    list( path_to_maxent.jar = "/home/veroandreo/software/maxent/maxent.jar",
          lq2lqptthreshold = 100,
          l2lqthreshold = 100))

# Run model
myBiomodModelOut <- BIOMOD_Modeling(
  myBiomodData,
  models = c('MAXENT.Phillips'),
```

MaxEnt model

```
# Set options
myBiomodOption <- BIOMOD_ModelingOptions(
  MAXENT.Phillips =
    list( path_to_maxent.jar = "/home/veroandreo/software/maxent/maxent.jar",
          lq2lqptthreshold = 100,
          l2lqthreshold = 100))

# Run model
myBiomodModelOut <- BIOMOD_Modeling(
  myBiomodData,
  models = c('MAXENT.Phillips'),
  models.options = myBiomodOption,
  NbRunEval=5,
  DataSplit=80,
  VarImport=10,
  models.eval.meth = c('ROC','ACCURACY'),
  SaveObj = TRUE,
  rescal.all.models = FALSE,
  do.full.models = FALSE,
  modeling.id = paste(myRespName,"FirstModeling",sep="_"))
```

MaxEnt model

```
myBiomodModelOut <- BIOMOD_Modeling(
  myBiomodData,
  models = c('MAXENT.Phillips'),
  models.options = myBiomodOption,
  NbRunEval=5,
  DataSplit=80,
  VarImport=10,
  models.eval.meth = c('ROC','ACCURACY'),
  SaveObj = TRUE,
  rescal.all.models = FALSE,
  do.full.models = FALSE,
  modeling.id = paste(myRespName,"FirstModeling",sep="_"))

# Inspect the model
myBiomodModelOut


#
# Model evaluation
#
```

MaxEnt model

```
  models.eval.meth = c('ROC','ACCURACY'),
  SaveObj = TRUE,
  rescal.all.models = FALSE,
  do.full.models = FALSE,
  modeling.id = paste(myRespName,"FirstModeling",sep="_"))

# Inspect the model
myBiomodModelOut



#
# Model evaluation
#


# Extract evaluation data
myBiomodModelEval <- get_evaluations(myBiomodModelOut)


# Accuracy
myBiomodModelEval["ACCURACY","Testing.data",,,]
```

MaxEnt model

```
# Inspect the model
myBiomodModelOut



#
# Model evaluation
#



# Extract evaluation data
myBiomodModelEval <- get_evaluations(myBiomodModelOut)


# Accuracy
myBiomodModelEval["ACCURACY","Testing.data",,,]

# ROC: Receiver-operator curve
myBiomodModelEval["ROC","Testing.data",,,]

# Variable importance
get_variables_importance(myBiomodModelOut)
```

MaxEnt model

```
#
# Model evaluation
#


# Extract evaluation data
myBiomodModelEval <- get_evaluations(myBiomodModelOut)


# Accuracy
myBiomodModelEval["ACCURACY","Testing.data",,]

# ROC: Receiver-operator curve
myBiomodModelEval["ROC","Testing.data",,]

# Variable importance
get_variables_importance(myBiomodModelOut)
```

MaxEnt model

```
# Model evaluation
#


# Extract evaluation data
myBiomodModelEval <- get_evaluations(myBiomodModelOut)


# Accuracy
myBiomodModelEval["ACCURACY","Testing.data",,,]

# ROC: Receiver-operator curve
myBiomodModelEval["ROC","Testing.data",,,]

# Variable importance
get_variables_importance(myBiomodModelOut)


#
# Model predictions
#
```

Model predictions

```
#####################################################################
# R commands for the session:
# "Analysis of space-time satellite data for disease ecology
# applications with GRASS GIS and R stats" at OpenGeoHub Summer School,
# Muenster (Germany)
#
# Modelling Aedes albopictus potential distribution in Northern Italy
#
# Original example contributed by Carol Garzon Lopez
# Adapted by Veronica Andreo
#
# Date: October, 2018
# Updated: August, 2019
#####################################################################



#
# Install and load required packages
#
```

Model predictions

```
# Variable importance
get_variables_importance(myBiomodModelOut)


#
# Model predictions
#


# Set parameters for model projection
myBiomodProj <- BIOMOD_Projection(
                modeling.output = myBiomodModelOut,
                new.env = myExpl,
                proj.name = "current",
                selected.models = "all",
                compress = FALSE,
                build.clamping.mask = FALSE)

# Obtain predictions
mod_proj <- get_predictions(myBiomodProj)
```

Model predictions

```
# Model predictions
#



# Set parameters for model projection
myBiomodProj <- BIOMOD_Projection(
                modeling.output = myBiomodModelOut,
                new.env = myExpl,
                proj.name = "current",
                selected.models = "all",
                compress = FALSE,
                build.clamping.mask = FALSE)


# Obtain predictions
mod_proj <- get_predictions(myBiomodProj)
mod_proj

# Plot predicted model
plot(mod_proj)
plot(mod_proj[[2]], main = "Predicted potential distribution")
```

Model predictions

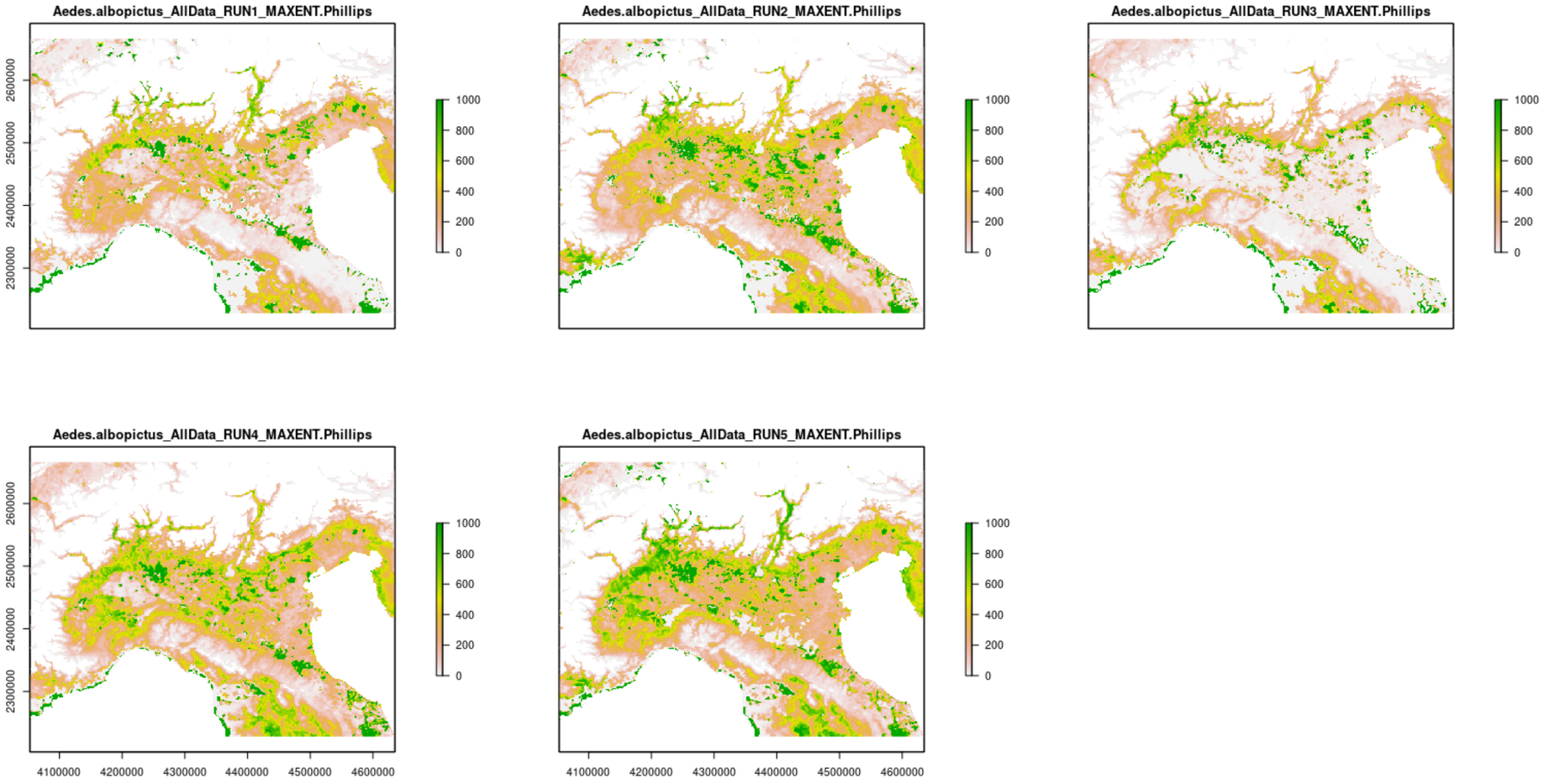
```
# Set parameters for model projection
myBiomodProj <- BIOMOD_Projection(
                modeling.output = myBiomodModelOut,
                new.env = myExpl,
                proj.name = "current",
                selected.models = "all",
                compress = FALSE,
                build.clamping.mask = FALSE)


# Obtain predictions
mod_proj <- get_predictions(myBiomodProj)
mod_proj

# Plot predicted model
plot(mod_proj)
plot(mod_proj[[2]], main = "Predicted potential distribution")



#
# Optionally, write data back to GRASS GIS
```


Aedes.albopictus_AllData_RUN1_MAXENT.Phillips
Aedes.albopictus_AllData_RUN2_MAXENT.Phillips
Aedes.albopictus_AllData_RUN3_MAXENT.Phillips
Aedes.albopictus_AllData_RUN4_MAXENT.Phillips
Aedes.albopictus_AllData_RUN5_MAXENT.Phillips
2300000
2400000
2500000
2600000
4100000
4200000
4300000
4400000
4500000
4600000
1000
800
600
400
200
0

***Task**: Explore the algorithms available in `BIOMOD_Modeling()` and try with a different one. Compare assumptions and results.*

Write model predictions into GRASS GIS DB

```
#####################################################################
# R commands for the session:
# "Analysis of space-time satellite data for disease ecology
# applications with GRASS GIS and R stats" at OpenGeoHub Summer School,
# Muenster (Germany)
#
# Modelling Aedes albopictus potential distribution in Northern Italy
#
# Original example contributed by Carol Garzon Lopez
# Adapted by Veronica Andreo
#
# Date: October, 2018
# Updated: August, 2019
#####################################################################



#
# Install and load required packages
#
```

Write model predictions into GRASS GIS DB

```
# Obtain predictions
mod_proj <- get_predictions(myBiomodProj)
mod_proj

# Plot predicted model
plot(mod_proj)
plot(mod_proj[[2]], main = "Predicted potential distribution")




#
# Optionally, write data back to GRASS GIS
#


# Only one layer
g <- as(mod_proj[[4]], 'SpatialGridDataFrame')
writeRAST(g, "maxent_albopictus", flags = "overwrite")

# Export all MaxEnt runs
```

Write model predictions into GRASS GIS DB

```
# Obtain predictions
mod_proj <- get_predictions(myBiomodProj)
mod_proj

# Plot predicted model
plot(mod_proj)
plot(mod_proj[[2]], main = "Predicted potential distribution")




#
# Optionally, write data back to GRASS GIS
#



# Only one layer
g <- as(mod_proj[[4]], 'SpatialGridDataFrame')
writeRAST(g, "maxent_albopictus", flags = "overwrite")

# Export all MaxEnt runs
for(i in seq_along(1:length(mod_proj@layers))){
```

QUESTIONS?

Other (very) useful resources

- Temporal data processing wiki
- GRASS GIS and R for time series processing wiki
- GRASS GIS temporal workshop at NCSU
- GRASS GIS workshop held in Jena 2018
- GRASS GIS course IRSAE 2018
- GRASS GIS course in Argentina 2018

References

- Gebbert, S., Pebesma, E. (2014). *A temporal GIS for field based environmental modeling*. Environmental Modelling & Software, 53, 1-12. DOI
- Gebbert, S., Pebesma, E. (2017). *The GRASS GIS temporal framework*. International Journal of Geographical Information Science 31, 1273-1292. DOI
- Gebbert, S., Leppelt, T. and Pebesma, E. (2019). *A Topology Based Spatio-Temporal Map Algebra for Big Data Analysis*. Data, 4, 86. DOI

Thanks for your attention!!

Verónica Andreo

veroandreo

@VeronicaAndreo

veroandreo@gmail.com