Analyzing space-time satellite data with GRASS GIS for environmental monitoring

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 GIS is **the first FOSS GIS** that incorporated capabilities to **manage, analyze, process and visualize spatio-temporal data**, as well as the temporal relationships among time series.

The TGRASS framework

The TGRASS framework

- TGRASS is the temporal enabled GRASS GIS designed to easily handle time series data

The TGRASS framework

- TGRASS is the temporal enabled GRASS GIS designed to easily handle time series data
- TGRASS is fully based on metadata and does not duplicate any dataset

The TGRASS framework

- TGRASS is the temporal enabled GRASS GIS designed to easily handle time series data
- TGRASS is fully based on metadata and does not duplicate any dataset
- Snapshot approach, i.e., adds time stamps to maps

The TGRASS framework

- TGRASS is the temporal enabled GRASS GIS designed to easily handle time series data
- TGRASS is fully based on metadata and does not duplicate any dataset
- Snapshot approach, i.e., adds time stamps to maps
- A collection of time stamped maps (snapshots) of the same variable are called space-time datasets or STDS

The TGRASS framework

- TGRASS is the temporal enabled GRASS GIS designed to easily handle time series data
- TGRASS is fully based on metadata and does not duplicate any dataset
- Snapshot approach, i.e., adds time stamps to maps
- A collection of time stamped maps (snapshots) of the same variable are called space-time datasets or STDS
- Maps in a STDS can have different spatial and temporal extents

Space-time datasets

- Space time raster datasets (**STRDS**)
- Space time 3D raster datasets (**STR3DS**)
- Space time vector datasets (**STVDS**)

📢 Suppot for **image collections** is on the way!

Other TGRASS notions

Other TGRASS notions

- Time can be defined as intervals (start and end time) or instances (only start time)

Other TGRASS notions

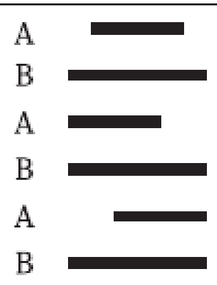
- Time can be defined as intervals (start and end time) or instances (only start time)
- Time can be absolute (e.g., 2017-04-06 22:39:49) or relative (e.g., 4 years, 90 days)

Other TGRASS notions

- Time can be defined as intervals (start and end time) or instances (only start time)
- Time can be absolute (e.g., 2017-04-06 22:39:49) or relative (e.g., 4 years, 90 days)
- Granularity is the greatest common divisor of the temporal extents (and possible gaps) of all maps in the space-time cube

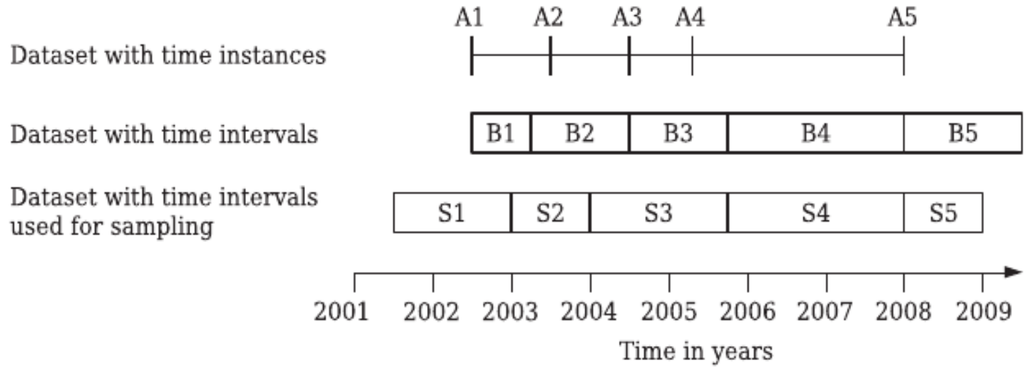
Other TGRASS notions

- Topology refers to temporal relations between time intervals in a STDS.

	A in relation to B	B in relation to A
	equivalent	equivalent
	overlap	overlap
	during	contain
	follows	precedes

Other TGRASS notions

- Temporal sampling is used to determine the state of one process during a second process.



Sampling time instances

	start
S1	A1
S2	A2
S3	A3,A4
S4	—
S5	A5

Sampling time intervals

	start	during	contain	overlap	equal
S1	B1	—	—	B1	—
S2	B2	—	—	B1,B2	—
S3	B3	B3	—	B2	—
S4	B4	—	—	—	B4
S5	B5	—	B5	—	—

Spatio-temporal modules

- **t.***: General modules to handle STDS of all types
- **t.rast.***: Modules that deal with STRDS
- **t.rast3d.***: Modules that deal with STR3DS
- **t.vect.***: Modules that deal with STVDS

TGRASS framework and workflow

SPATIO-TEMPORAL GIS MODULES

GRASS DATABASE

3D rasters | Rasters | Vectors

Create STDS
t.create

Assign time stamps
t.register

t.rast.to.vect

STR3DS **STRDS** **STVDS**

TEMPORAL DATABASE

General utility: t.list, t.info, t.remove, t.select, t.sample, t.topology, etc.

Other useful modules and Add-ons

g.gui.mapswipe

Gap-fill:
r.hants,
r.series.lwr

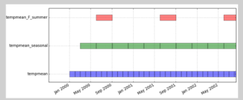
Nulls:
t.rast.nulls

Data extraction:
v.what.strds,
v.strds.stats,
t.rast.what.aggr

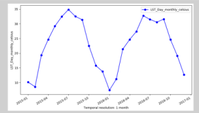
Others:
r.seasons,
r.regression.series,
r.bioclim

t.rast.out.xyz

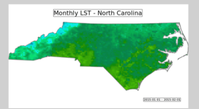
Visualization



g.gui.timeline



g.gui.tplot



g.gui.animation

t.rast.list,
t.rast.extract,
t.rast.univar

List, extract, and statistics

t.vect.list,
t.vect.extract,
t.vect.univar

t.rast.colors,
t.rast.gapfill

Pre-processing

t.rast.series

t.rast.aggregate

Processing and analysis

t.vect.what.strds
t.vect.observe.strds

t.rast.algebra

```
t.rast.algebra "C = A {+, contains} B"
t.rast.algebra "C = A + map(b)"
t.rast.algebra "C = (A[-1] + A[0] + A[1])/3"
```

t.rast.accumulate + t.rast.accdetect

ST algebra

t.vect.algebra

```
t.vect.algebra \
"C = A {|,equal|overlap,u} B"
t.vect.algebra \
"C = buff_p(A,1) {&,overlap,i} buff_p(B,1)"
```

t.rast.import
t.rast.export

Import/Export

t.vect.import
t.vect.export

Hands-on to raster time series in GRASS GIS

Overview

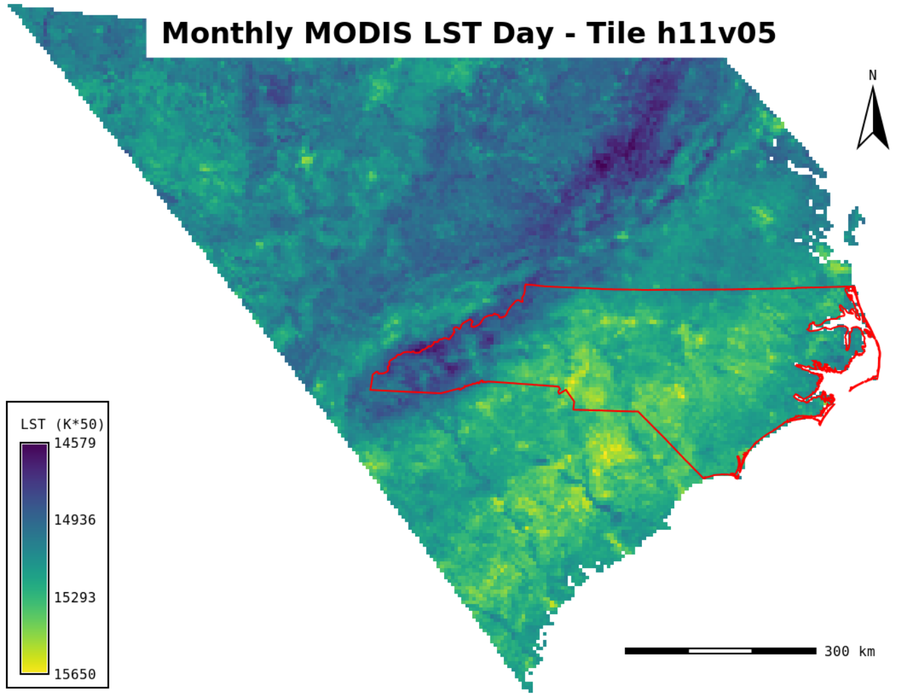
- Data for the session
- Create time series and register maps
- Convert from K to Celsius degrees
- Temporal operations
- Temporal aggregations
- Climatologies and anomalies
- Zonal statistics and SUHI

Sample location: North Carolina

- Download the **North Carolina location**
- Create a folder in your `$HOME` directory (or Documents) and name it `grassdata`
- Unzip the file `nc_basic_ogh_2019.zip` within `grassdata`
- Download the GRASS script to follow the session

Data for the session

- LST Day from MOD11B3 Collection 6
- Tiled monthly composites (h11v05)
- Spatial resolution: 5600m
- Selected time period: 2015 - 2017



Let's start GRASS GIS!

Set computational region and apply MASK

```
#!/bin/bash
############################################################################
# Commands for the TGRASS lecture at GEOSTAT Summer School in Prague
# Author: Veronica Andreo
# Date: July - August, 2018 - Edited October, 2018 and July, 2019
############################################################################


########### Before the workshop (done for you in advance) #############

#~ # Install i.modis add-on (requires pymodis library - www.pymodis.org)
#~ g.extension extension=i.modis

#~ # Download and import MODIS LST data
#~ # Note: User needs to be registered at Earthdata:
#~ # https://urs.earthdata.nasa.gov/users/new
#~ i.modis.download settings=$HOME/gisdata/SETTING \
  #~ product=lst_terra_monthly_5600 \
  #~ tile=h11v05 \
  #~ startday=2015-01-01 endday=2017-12-31 \
  #~ folder=/tmp
```

Set computational region and apply MASK

```
#~ i.modis.import files=/tmp/listfileMOD11B3.006.txt \
  #~ spectral="( 1 0 0 0 0 0 0 0 0 0 0 0 )"



############## For the workshop (what you have to do) ################

## Download the ready to use mapset 'modis_lst' from:
## https://gitlab.com/veroandreo/grass-gis-geostat-2018
## and unzip it into North Carolina full LOCATION 'nc_spm_08_grass7'

# Get list of raster maps in the 'modis_lst' mapset
g.list type=raster

# Get info from one of the raster maps
r.info map=MOD11B3.A2015060.h11v05.single_LST_Day_6km



## Region settings and MASK

# Set region to NC state boundary with LST maps' resolution
g.region -p vector=nc_state \
```

Set computational region and apply MASK

```
## Region settings and MASK

# Set region to NC state boundary with LST maps' resolution
g.region -p vector=nc_state \
  align=MOD11B3.A2015060.h11v05.single_LST_Day_6km

#~ projection: 99 (Lambert Conformal Conic)
#~ zone:       0
#~ datum:      nad83
#~ ellipsoid:  a=6378137 es=0.006694380022900787
#~ north:      323380.12411493
#~ south:      9780.12411493
#~ west:       122934.46411531
#~ east:       934934.46411531
#~ nsres:      5600
#~ ewres:      5600
#~ rows:       56
#~ cols:       145
```

Set computational region and apply MASK

```
#~ south:          9780.12411493
#~ west:           122934.46411531
#~ east:           934934.46411531
#~ nsres:          5600
#~ ewres:          5600
#~ rows:           56
#~ cols:           145

#~ cells:          8120


# Set a MASK to NC state boundary
r.mask vector=nc_state

# you should see this statement in the terminal from now on
#~ [Raster MASK present]


## Time series


# Create the STRDS
t.create type=strds temporaltype=absolute output=LST_Day_monthly \
```

Create a temporal dataset (STDS)

t.create

- Creates an SQLite container table in the temporal database
- Handles huge amounts of maps by using the STDS as input
- We need to specify:
  - *type of maps* (raster, raster3d or vector)
  - *type of time* (absolute or relative)

Create a raster time series (STRDS)

```
#!/bin/bash
############################################################################
# Commands for the TGRASS lecture at GEOSTAT Summer School in Prague
# Author: Veronica Andreo
# Date: July - August, 2018 - Edited October, 2018 and July, 2019
############################################################################


########### Before the workshop (done for you in advance) #############

#~ # Install i.modis add-on (requires pymodis library - www.pymodis.org)
#~ g.extension extension=i.modis

#~ # Download and import MODIS LST data
#~ # Note: User needs to be registered at Earthdata:
#~ # https://urs.earthdata.nasa.gov/users/new
#~ i.modis.download settings=$HOME/gisdata/SETTING \
  #~ product=lst_terra_monthly_5600 \
  #~ tile=h11v05 \
  #~ startday=2015-01-01 endday=2017-12-31 \
  #~ folder=/tmp
```

Create a raster time series (STRDS)

```
# Set a MASK to NC state boundary
r.mask vector=nc_state

# you should see this statement in the terminal from now on
#~ [Raster MASK present]


## Time series

# Create the STRDS
t.create type=strds temporaltype=absolute output=LST_Day_monthly \
  title="Monthly LST Day 5.6 km" \
  description="Monthly LST Day 5.6 km MOD11B3.006, 2015-2017"

# Check if the STRDS is created
t.list type=strds

# Get info about the STRDS
t.info input=LST_Day_monthly
```

Create a raster time series (STRDS)

```
# you should see this statement in the terminal from now on
#~ [Raster MASK present]


## Time series

# Create the STRDS
t.create type=strds temporaltype=absolute output=LST_Day_monthly \
  title="Monthly LST Day 5.6 km" \
  description="Monthly LST Day 5.6 km MOD11B3.006, 2015-2017"

# Check if the STRDS is created
t.list type=strds

# Get info about the STRDS
t.info input=LST_Day_monthly


## Add time stamps to maps (i.e., register maps)

# in Unix systems
```

Create a raster time series (STRDS)

```
## Time series

# Create the STRDS
t.create type=strds temporaltype=absolute output=LST_Day_monthly \
  title="Monthly LST Day 5.6 km" \
  description="Monthly LST Day 5.6 km MOD11B3.006, 2015-2017"

# Check if the STRDS is created
t.list type=strds

# Get info about the STRDS
t.info input=LST_Day_monthly


## Add time stamps to maps (i.e., register maps)

# in Unix systems
t.register -i input=LST_Day_monthly \
 maps=$(g.list type=raster pattern="MOD11B3*LST_Day*" separator=comma) \
 start="2015-01-01" increment="1 months"
```

Register maps into the STRDS

t.register

- Assigns time stamps to maps
- We need:
  - the *empty STDS* as input, i.e., the container table,
  - the *list of maps* to be registered,
  - the *start date*,
  - *increment* option along with the *-i* flag for interval creation

For more details, check the t.register manual and related map registration wiki page.

Register maps in STRDS (assign time stamps)

```
#!/bin/bash
############################################################################
# Commands for the TGRASS lecture at GEOSTAT Summer School in Prague
# Author: Veronica Andreo
# Date: July - August, 2018 - Edited October, 2018 and July, 2019
############################################################################


########### Before the workshop (done for you in advance) #############

#~ # Install i.modis add-on (requires pymodis library - www.pymodis.org)
#~ g.extension extension=i.modis

#~ # Download and import MODIS LST data
#~ # Note: User needs to be registered at Earthdata:
#~ # https://urs.earthdata.nasa.gov/users/new
#~ i.modis.download settings=$HOME/gisdata/SETTING \
  #~ product=lst_terra_monthly_5600 \
  #~ tile=h11v05 \
  #~ startday=2015-01-01 endday=2017-12-31 \
  #~ folder=/tmp
```

Register maps in STRDS (assign time stamps)

```
  description="Monthly LST Day 5.6 km MOD11B3.006, 2015-2017"

# Check if the STRDS is created
t.list type=strds

# Get info about the STRDS
t.info input=LST_Day_monthly


## Add time stamps to maps (i.e., register maps)

# in Unix systems
t.register -i input=LST_Day_monthly \
 maps=$(g.list type=raster pattern="MOD11B3*LST_Day*" separator=comma) \
 start="2015-01-01" increment="1 months"

# in MS Windows, first create the list of maps
g.list type=raster pattern="MOD11B3*LST_Day*" output=map_list.txt
t.register -i input=LST_Day_monthly \

 file=map_list.txt start="2015-01-01" increment="1 months"
```

Register maps in STRDS (assign time stamps)

```
t.info input=LST_Day_monthly


## Add time stamps to maps (i.e., register maps)

# in Unix systems
t.register -i input=LST_Day_monthly \
 maps=$(g.list type=raster pattern="MOD11B3*LST_Day*" separator=comma) \
 start="2015-01-01" increment="1 months"

# in MS Windows, first create the list of maps
g.list type=raster pattern="MOD11B3*LST_Day*" output=map_list.txt
t.register -i input=LST_Day_monthly \
 file=map_list.txt start="2015-01-01" increment="1 months"

# Check info again
t.info input=LST_Day_monthly

# Check the list of maps in the STRDS
t.rast.list input=LST_Day_monthly
```

Register maps in STRDS (assign time stamps)

```
# in Unix systems
t.register -i input=LST_Day_monthly \
 maps=$(g.list type=raster pattern="MOD11B3*LST_Day*" separator=comma) \
 start="2015-01-01" increment="1 months"

# in MS Windows, first create the list of maps
g.list type=raster pattern="MOD11B3*LST_Day*" output=map_list.txt
t.register -i input=LST_Day_monthly \
 file=map_list.txt start="2015-01-01" increment="1 months"

# Check info again
t.info input=LST_Day_monthly

# Check the list of maps in the STRDS
t.rast.list input=LST_Day_monthly

# Check min and max per map
t.rast.list input=LST_Day_monthly columns=name,min,max
```

Register maps in STRDS (assign time stamps)

```
 maps=$(g.list type=raster pattern="MOD11B3*LST_Day*" separator=comma) \
 start="2015-01-01" increment="1 months"

# in MS Windows, first create the list of maps
g.list type=raster pattern="MOD11B3*LST_Day*" output=map_list.txt
t.register -i input=LST_Day_monthly \

 file=map_list.txt start="2015-01-01" increment="1 months"

# Check info again
t.info input=LST_Day_monthly

# Check the list of maps in the STRDS
t.rast.list input=LST_Day_monthly

# Check min and max per map
t.rast.list input=LST_Day_monthly columns=name,min,max


## Let's see a graphical representation of our STRDS
g.gui.timeline inputs=LST_Day_monthly
```

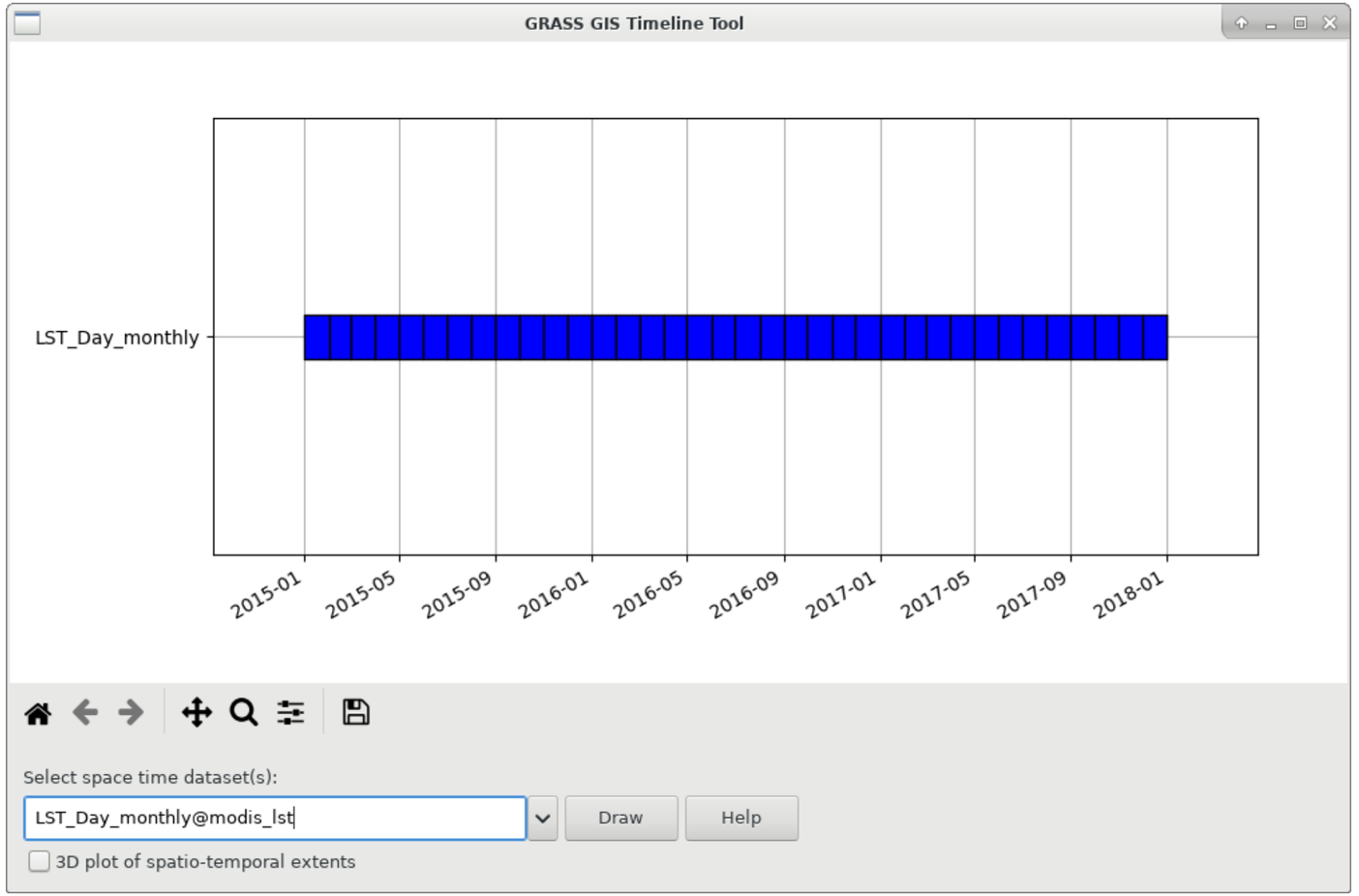
Register maps in STRDS (assign time stamps)

```
# in MS Windows, first create the list of maps
g.list type=raster pattern="MOD11B3*LST_Day*" output=map_list.txt
t.register -i input=LST_Day_monthly \
 file=map_list.txt start="2015-01-01" increment="1 months"

# Check info again
t.info input=LST_Day_monthly

# Check the list of maps in the STRDS
t.rast.list input=LST_Day_monthly

# Check min and max per map
t.rast.list input=LST_Day_monthly columns=name,min,max


## Let's see a graphical representation of our STRDS
g.gui.timeline inputs=LST_Day_monthly


## Temporal calculations: K*50 to Celsius
```

Register maps in STRDS (assign time stamps)

```
 file=map_list.txt start="2015-01-01" increment="1 months"

# Check info again
t.info input=LST_Day_monthly

# Check the list of maps in the STRDS
t.rast.list input=LST_Day_monthly

# Check min and max per map
t.rast.list input=LST_Day_monthly columns=name,min,max


## Let's see a graphical representation of our STRDS
g.gui.timeline inputs=LST_Day_monthly


## Temporal calculations: K*50 to Celsius

# Re-scale data to degrees Celsius
t.rast.algebra basename=LST_Day_monthly_celsius suffix=gran\
  expression="LST_Day_monthly_celsius = LST_Day_monthly * 0.02 - 273.15"
```

Monthly LST for the period 2015-2017



See g.gui.timeline manual page

Operations with temporal algebra

t.rast.algebra

- Performs a wide range of temporal and spatial operations based on map's spatial and temporal topology
  - Temporal operators: union, intersection, etc.
  - Temporal functions: *start_time()*, *start_doy()*, etc.
  - Spatial operators (subset of r.mapcalc)
  - Temporal neighbourhood modifier: *[x,y,t]*
  - Other temporal functions like *tsnap()*, *buff_t()* or *tshift()*

they can all be combined in complex expressions!! 💣

From K*50 to Celsius using the temporal calculator

```
#!/bin/bash
############################################################################
# Commands for the TGRASS lecture at GEOSTAT Summer School in Prague
# Author: Veronica Andreo
# Date: July - August, 2018 - Edited October, 2018 and July, 2019
############################################################################


########### Before the workshop (done for you in advance) #############

#~ # Install i.modis add-on (requires pymodis library - www.pymodis.org)
#~ g.extension extension=i.modis

#~ # Download and import MODIS LST data
#~ # Note: User needs to be registered at Earthdata:
#~ # https://urs.earthdata.nasa.gov/users/new
#~ i.modis.download settings=$HOME/gisdata/SETTING \
  #~ product=lst_terra_monthly_5600 \
  #~ tile=h11v05 \
  #~ startday=2015-01-01 endday=2017-12-31 \
  #~ folder=/tmp
```

From K*50 to Celsius using the temporal calculator

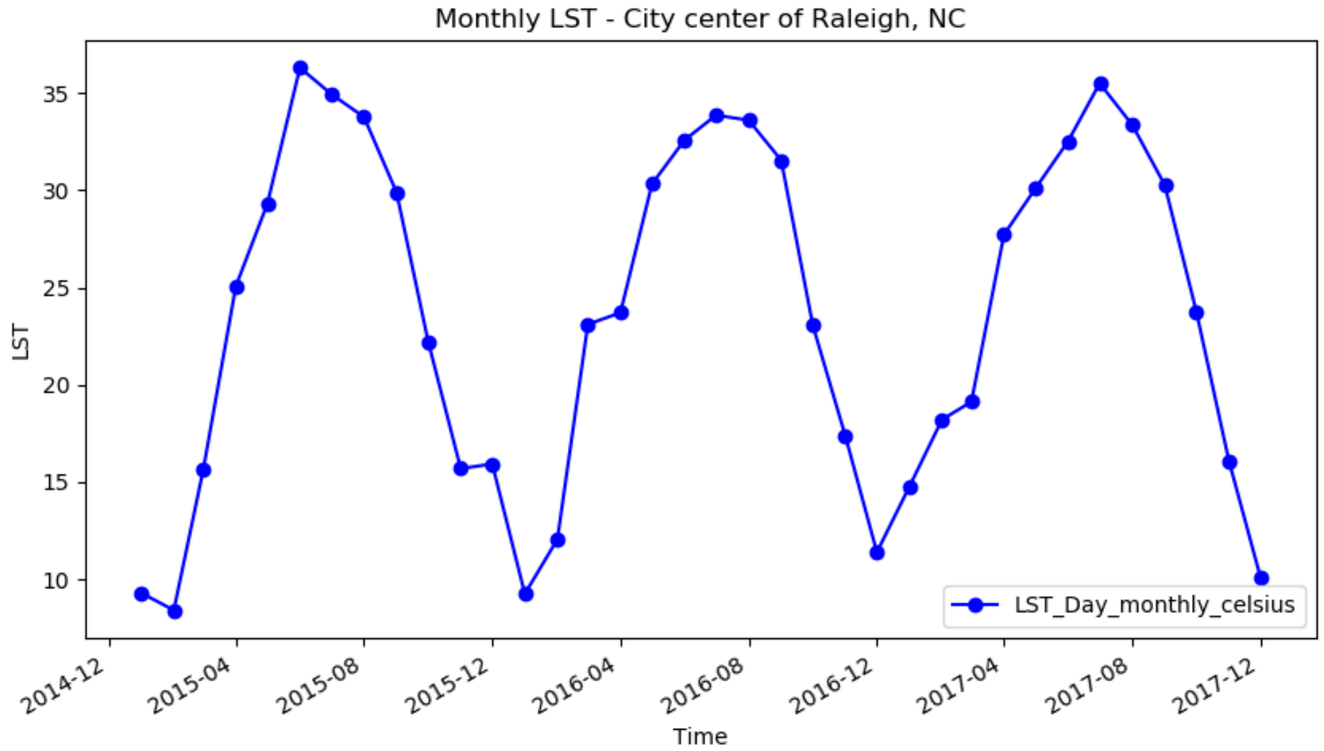
```
t.rast.list input=LST_Day_monthly

# Check min and max per map
t.rast.list input=LST_Day_monthly columns=name,min,max


## Let's see a graphical representation of our STRDS
g.gui.timeline inputs=LST_Day_monthly


## Temporal calculations: K*50 to Celsius

# Re-scale data to degrees Celsius
t.rast.algebra basename=LST_Day_monthly_celsius suffix=gran\
  expression="LST_Day_monthly_celsius = LST_Day_monthly * 0.02 - 273.15"

# Check info
t.info LST_Day_monthly_celsius


## Time series plots
```

From K*50 to Celsius using the temporal calculator

```
## Let's see a graphical representation of our STRDS
g.gui.timeline inputs=LST_Day_monthly


## Temporal calculations: K*50 to Celsius

# Re-scale data to degrees Celsius
t.rast.algebra basename=LST_Day_monthly_celsius suffix=gran\
  expression="LST_Day_monthly_celsius = LST_Day_monthly * 0.02 - 273.15"

# Check info
t.info LST_Day_monthly_celsius


## Time series plots

# LST time series plot for the city center of Raleigh
g.gui.tplot strds=LST_Day_monthly_celsius \
  coordinates=641428.783478,229901.400746 \
```

From K*50 to Celsius using the temporal calculator

```
# Re-scale data to degrees Celsius
t.rast.algebra basename=LST_Day_monthly_celsius suffix=gran\
  expression="LST_Day_monthly_celsius = LST_Day_monthly * 0.02 - 273.15"

# Check info
t.info LST_Day_monthly_celsius


## Time series plots

# LST time series plot for the city center of Raleigh
g.gui.tplot strds=LST_Day_monthly_celsius \
  coordinates=641428.783478,229901.400746 \
  title="Monthly LST. City center of Raleigh, NC " \
  xlabel="Time" ylabel="LST" \

  csv=raleigh_monthly_LST.csv


## Get specific lists of maps
```



Point coordinates can be typed directly, copied from the map display or directly chosen from the main map display.

For a single point, see g.gui.tplot. For a vector of points, see t.rast.what.

Lists and selections

- **t.list** for listing STDS and maps registered in the temporal database,
- **t.rast.list** for maps in raster time series, and
- **t.vect.list** for maps in vector time series.

Listing examples

```
#!/bin/bash
############################################################################
# Commands for the TGRASS lecture at GEOSTAT Summer School in Prague
# Author: Veronica Andreo
# Date: July - August, 2018 - Edited October, 2018 and July, 2019
############################################################################


########### Before the workshop (done for you in advance) #############

#~ # Install i.modis add-on (requires pymodis library - www.pymodis.org)
#~ g.extension extension=i.modis

#~ # Download and import MODIS LST data
#~ # Note: User needs to be registered at Earthdata:
#~ # https://urs.earthdata.nasa.gov/users/new
#~ i.modis.download settings=$HOME/gisdata/SETTING \
  #~ product=lst_terra_monthly_5600 \
  #~ tile=h11v05 \
  #~ startday=2015-01-01 endday=2017-12-31 \
  #~ folder=/tmp
```

Listing examples

```
  xlabel="Time" ylabel="LST" \

  csv=raleigh_monthly_LST.csv



## Get specific lists of maps


# Maps with minimum value lower than or equal to 5
t.rast.list input=LST_Day_monthly_celsius order=min \
 columns=name,start_time,min where="min <= '5.0'"

#~ name|start_time|min
#~ LST_Day_monthly_celsius_2015_02|2015-02-01 00:00:00|-1.31
#~ LST_Day_monthly_celsius_2017_01|2017-01-01 00:00:00|-0.89
#~ LST_Day_monthly_celsius_2015_01|2015-01-01 00:00:00|-0.25
#~ LST_Day_monthly_celsius_2016_01|2016-01-01 00:00:00|-0.17
#~ LST_Day_monthly_celsius_2016_02|2016-02-01 00:00:00|0.73
#~ LST_Day_monthly_celsius_2017_12|2017-12-01 00:00:00|1.69
#~ LST_Day_monthly_celsius_2016_12|2016-12-01 00:00:00|3.45


# Maps with maximum value higher than 30
```

Listing examples

```
#~ LST_Day_monthly_celsius_2017_01|2017-01-01 00:00:00|-0.89
#~ LST_Day_monthly_celsius_2015_01|2015-01-01 00:00:00|-0.25
#~ LST_Day_monthly_celsius_2016_01|2016-01-01 00:00:00|-0.17
#~ LST_Day_monthly_celsius_2016_02|2016-02-01 00:00:00|0.73
#~ LST_Day_monthly_celsius_2017_12|2017-12-01 00:00:00|1.69
#~ LST_Day_monthly_celsius_2016_12|2016-12-01 00:00:00|3.45


# Maps with maximum value higher than 30
t.rast.list input=LST_Day_monthly_celsius order=max \
 columns=name,start_time,max where="max > '30.0'"

#~ name|start_time|max
#~ LST_Day_monthly_celsius_2017_04|2017-04-01 00:00:00|30.85
#~ LST_Day_monthly_celsius_2017_09|2017-09-01 00:00:00|32.45
#~ LST_Day_monthly_celsius_2016_05|2016-05-01 00:00:00|32.97
#~ LST_Day_monthly_celsius_2015_09|2015-09-01 00:00:00|33.49
#~ LST_Day_monthly_celsius_2017_05|2017-05-01 00:00:00|34.35
#~ LST_Day_monthly_celsius_2015_05|2015-05-01 00:00:00|34.53
#~ LST_Day_monthly_celsius_2017_08|2017-08-01 00:00:00|35.81
#~ LST_Day_monthly_celsius_2016_09|2016-09-01 00:00:00|36.33
#~ LST_Day_monthly_celsius_2016_08|2016-08-01 00:00:00|36.43
```

Listing examples

```
#~ LST_Day_monthly_celsius_2017_09|2017-09-01 00:00:00|32.45
#~ LST_Day_monthly_celsius_2016_05|2016-05-01 00:00:00|32.97
#~ LST_Day_monthly_celsius_2015_09|2015-09-01 00:00:00|33.49
#~ LST_Day_monthly_celsius_2017_05|2017-05-01 00:00:00|34.35
#~ LST_Day_monthly_celsius_2015_05|2015-05-01 00:00:00|34.53
#~ LST_Day_monthly_celsius_2017_08|2017-08-01 00:00:00|35.81
#~ LST_Day_monthly_celsius_2016_09|2016-09-01 00:00:00|36.33
#~ LST_Day_monthly_celsius_2016_08|2016-08-01 00:00:00|36.43


# Maps between two given dates
t.rast.list input=LST_Day_monthly_celsius columns=name,start_time \
 where="start_time >= '2015-05' and start_time <= '2015-08-01 00:00:00'"

#~ name|start_time
#~ LST_Day_monthly_celsius_2015_05|2015-05-01 00:00:00
#~ LST_Day_monthly_celsius_2015_06|2015-06-01 00:00:00
#~ LST_Day_monthly_celsius_2015_07|2015-07-01 00:00:00
#~ LST_Day_monthly_celsius_2015_08|2015-08-01 00:00:00

# Maps from January
```

Listing examples

```
# Maps between two given dates
t.rast.list input=LST_Day_monthly_celsius columns=name,start_time \
 where="start_time >= '2015-05' and start_time <= '2015-08-01 00:00:00'"

#~ name|start_time
#~ LST_Day_monthly_celsius_2015_05|2015-05-01 00:00:00
#~ LST_Day_monthly_celsius_2015_06|2015-06-01 00:00:00
#~ LST_Day_monthly_celsius_2015_07|2015-07-01 00:00:00
#~ LST_Day_monthly_celsius_2015_08|2015-08-01 00:00:00


# Maps from January
t.rast.list input=LST_Day_monthly_celsius columns=name,start_time \
 where="strftime('%m', start_time)='01'"

#~ name|start_time
#~ LST_Day_monthly_celsius_2015_01|2015-01-01 00:00:00
#~ LST_Day_monthly_celsius_2016_01|2016-01-01 00:00:00
#~ LST_Day_monthly_celsius_2017_01|2017-01-01 00:00:00
```

Temporal aggregation 1: Using the full time series

t.rast.series

- Aggregates full STRDS or parts of it using the *where* option
- Different methods available: average, minimum, maximum, median, mode, etc.

Maximum and minimum LST in the past 3 years

```
#!/bin/bash
############################################################################
# Commands for the TGRASS lecture at GEOSTAT Summer School in Prague
# Author: Veronica Andreo
# Date: July - August, 2018 - Edited October, 2018 and July, 2019
############################################################################


########### Before the workshop (done for you in advance) #############

#~ # Install i.modis add-on (requires pymodis library - www.pymodis.org)
#~ g.extension extension=i.modis

#~ # Download and import MODIS LST data
#~ # Note: User needs to be registered at Earthdata:
#~ # https://urs.earthdata.nasa.gov/users/new
#~ i.modis.download settings=$HOME/gisdata/SETTING \
  #~ product=lst_terra_monthly_5600 \
  #~ tile=h11v05 \
  #~ startday=2015-01-01 endday=2017-12-31 \
  #~ folder=/tmp
```

Maximum and minimum LST in the past 3 years

```
#~ LST_Day_monthly_celsius_2015_05@modis_lst|2015-05-01 00:00:00|2015-06-01 00:00:0

#~ # Get extended statistics

#~ t.rast.univar -e input=LST_Day_monthly_celsius

#~ # Write the univariate stats output to a csv file
#~ t.rast.univar input=LST_Day_monthly_celsius separator=comma \
  #~ output=stats_LST_Day_monthly_celsius.csv



## Temporal aggregations (full series)

# Get maximum LST in the STRDS
t.rast.series input=LST_Day_monthly_celsius \
  output=LST_Day_max method=maximum

# Get minimum LST in the STRDS
t.rast.series input=LST_Day_monthly_celsius \
  output=LST_Day_min method=minimum
```

Maximum and minimum LST in the past 3 years

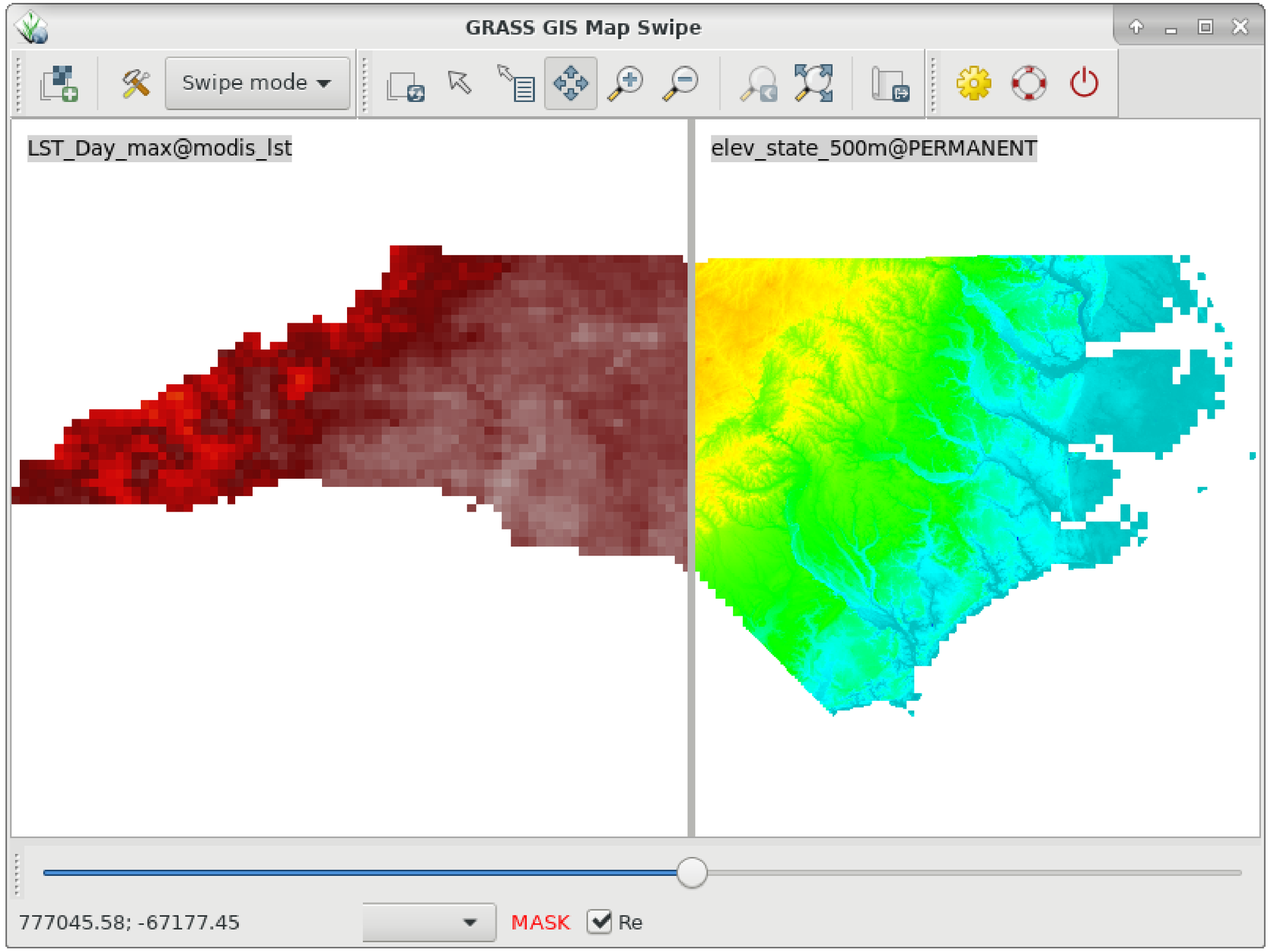
```
#~ t.rast.univar -e input=LST_Day_monthly_celsius

#~ # Write the univariate stats output to a csv file
#~ t.rast.univar input=LST_Day_monthly_celsius separator=comma \
  #~ output=stats_LST_Day_monthly_celsius.csv


## Temporal aggregations (full series)

# Get maximum LST in the STRDS
t.rast.series input=LST_Day_monthly_celsius \
  output=LST_Day_max method=maximum

# Get minimum LST in the STRDS
t.rast.series input=LST_Day_monthly_celsius \
  output=LST_Day_min method=minimum

# Change color pallete to celsius
r.colors map=LST_Day_min,LST_Day_max color=celsius
```

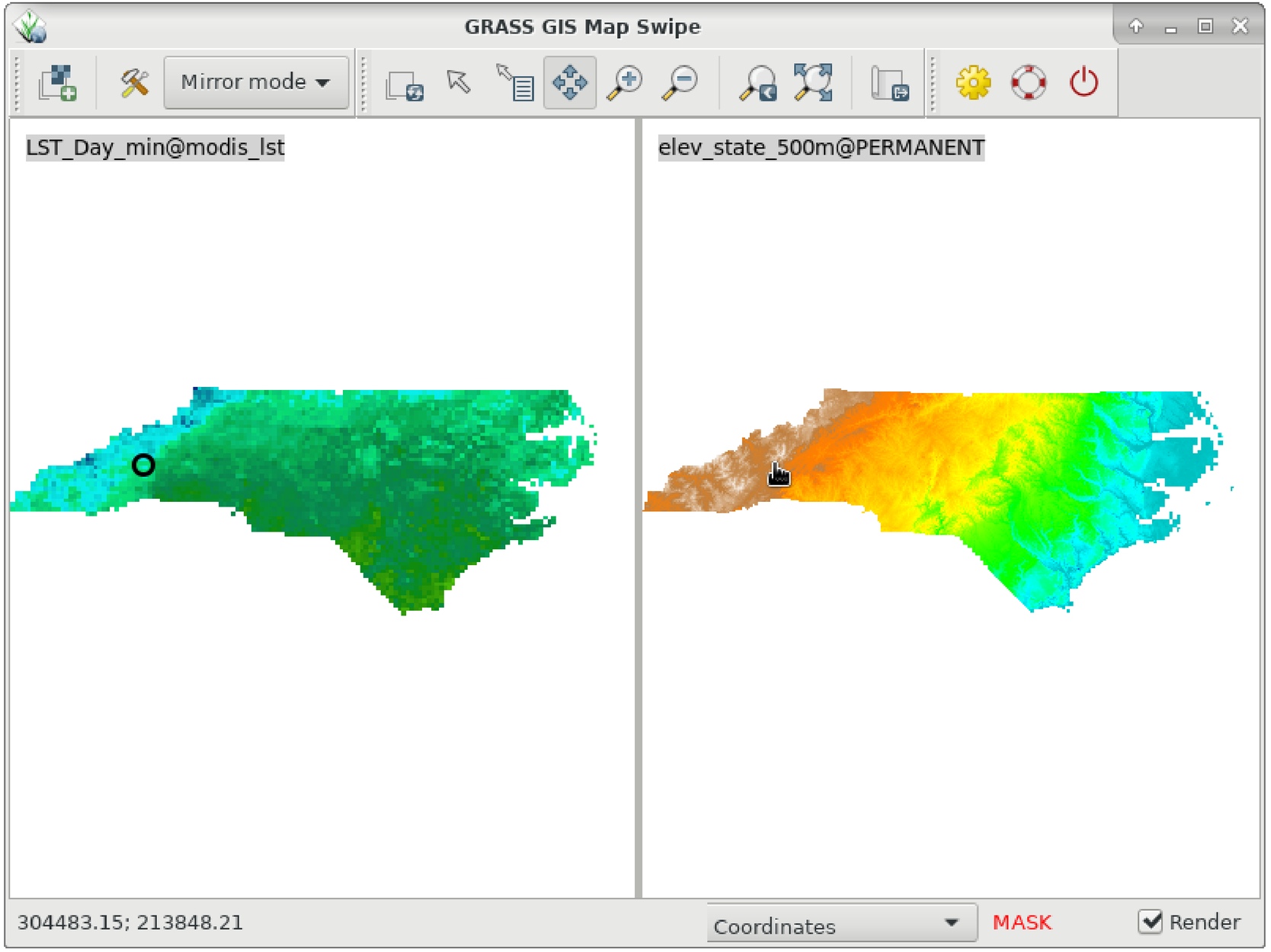
Maximum and minimum LST in the past 3 years

```
#~ t.rast.univar input=LST_Day_monthly_celsius separator=comma \
  #~ output=stats_LST_Day_monthly_celsius.csv



## Temporal aggregations (full series)

# Get maximum LST in the STRDS
t.rast.series input=LST_Day_monthly_celsius \
  output=LST_Day_max method=maximum

# Get minimum LST in the STRDS
t.rast.series input=LST_Day_monthly_celsius \
  output=LST_Day_min method=minimum

# Change color pallete to celsius
r.colors map=LST_Day_min,LST_Day_max color=celsius



## Display the new maps with mapswipe and compare them to elevation

# LST_Day_max & elevation
```

Maximum and minimum LST in the past 3 years

```
# Get maximum LST in the STRDS
t.rast.series input=LST_Day_monthly_celsius \
  output=LST_Day_max method=maximum

# Get minimum LST in the STRDS
t.rast.series input=LST_Day_monthly_celsius \
  output=LST_Day_min method=minimum

# Change color pallete to celsius
r.colors map=LST_Day_min,LST_Day_max color=celsius


## Display the new maps with mapswipe and compare them to elevation

# LST_Day_max & elevation
g.gui.mapswipe first=LST_Day_max second=elev_state_500m

# LST_Day_min & elevation
g.gui.mapswipe first=LST_Day_min second=elev_state_500m
```


GRASS GIS Map Swipe
Swipe mode
LST_Day_max@modis_lst
elev_state_500m@PERMANENT
777045.58; -67177.45
MASK
Re


GRASS GIS Map Swipe
Mirror mode
LST_Day_min@modis_lst
elev_state_500m@PERMANENT
304483.15; 213848.21
Coordinates
MASK
Render

Temporal operations using time variables

t.rast.mapcalc

- Performs spatio-temporal mapcalc expressions
- It allows for *spatial and temporal operators*, as well as *internal variables* in the expression string
- The temporal variables include: *start_time(), end_time(), start_month(), start_doy()*, etc.

Which is the month of the maximum LST?

```
#!/bin/bash
############################################################################
# Commands for the TGRASS lecture at GEOSTAT Summer School in Prague
# Author: Veronica Andreo
# Date: July - August, 2018 - Edited October, 2018 and July, 2019
############################################################################


########### Before the workshop (done for you in advance) #############

#~ # Install i.modis add-on (requires pymodis library - www.pymodis.org)
#~ g.extension extension=i.modis

#~ # Download and import MODIS LST data
#~ # Note: User needs to be registered at Earthdata:
#~ # https://urs.earthdata.nasa.gov/users/new
#~ i.modis.download settings=$HOME/gisdata/SETTING \
  #~ product=lst_terra_monthly_5600 \
  #~ tile=h11v05 \
  #~ startday=2015-01-01 endday=2017-12-31 \
  #~ folder=/tmp
```

Which is the month of the maximum LST?

```
## Display the new maps with mapswipe and compare them to elevation

# LST_Day_max & elevation
g.gui.mapswipe first=LST_Day_max second=elev_state_500m

# LST_Day_min & elevation
g.gui.mapswipe first=LST_Day_min second=elev_state_500m



## Temporal operations with time variables


# Get month of maximum LST

t.rast.mapcalc -n inputs=LST_Day_monthly_celsius output=month_max_lst \
  expression="if(LST_Day_monthly_celsius == LST_Day_max, start_month(), null())" \
  basename=month_max_lst

# Get basic info
t.info month_max_lst
```

Which is the month of the maximum LST?

```
# LST_Day_max & elevation
g.gui.mapswipe first=LST_Day_max second=elev_state_500m

# LST_Day_min & elevation
g.gui.mapswipe first=LST_Day_min second=elev_state_500m



## Temporal operations with time variables


# Get month of maximum LST

t.rast.mapcalc -n inputs=LST_Day_monthly_celsius output=month_max_lst \
  expression="if(LST_Day_monthly_celsius == LST_Day_max, start_month(), null())" \
  basename=month_max_lst

# Get basic info
t.info month_max_lst

# Get the earliest month in which the maximum appeared (method minimum)
t.rast.series input=month_max_lst method=minimum output=max_lst_date
```

Which is the month of the maximum LST?

```
# LST_Day_min & elevation
g.gui.mapswipe first=LST_Day_min second=elev_state_500m



## Temporal operations with time variables

# Get month of maximum LST

t.rast.mapcalc -n inputs=LST_Day_monthly_celsius output=month_max_lst \
  expression="if(LST_Day_monthly_celsius == LST_Day_max, start_month(), null())" \
  basename=month_max_lst

# Get basic info
t.info month_max_lst

# Get the earliest month in which the maximum appeared (method minimum)
t.rast.series input=month_max_lst method=minimum output=max_lst_date

# Remove month_max_lst strds
# we were only interested in the resulting aggregated map
t.remove -rf inputs=month_max_lst
```

Which is the month of the maximum LST?

```
# Get month of maximum LST

t.rast.mapcalc -n inputs=LST_Day_monthly_celsius output=month_max_lst \
  expression="if(LST_Day_monthly_celsius == LST_Day_max, start_month(), null())" \
  basename=month_max_lst

# Get basic info
t.info month_max_lst

# Get the earliest month in which the maximum appeared (method minimum)
t.rast.series input=month_max_lst method=minimum output=max_lst_date

# Remove month_max_lst strds
# we were only interested in the resulting aggregated map
t.remove -rf inputs=month_max_lst

# Note that the flags "-rf" force (immediate) removal of both
# the STRDS and the maps registered in it.
```

Which is the month of the maximum LST?

```
# Get basic info
t.info month_max_lst

# Get the earliest month in which the maximum appeared (method minimum)
t.rast.series input=month_max_lst method=minimum output=max_lst_date

# Remove month_max_lst strds
# we were only interested in the resulting aggregated map
t.remove -rf inputs=month_max_lst

# Note that the flags "-rf" force (immediate) removal of both
# the STRDS and the maps registered in it.



## Display maps in a wx monitor

# Open a monitor
d.mon wx0

# Display the raster map
d.rast map=max_lst_date
```

Which is the month of the maximum LST?

```
t.rast.series input=month_max_lst method=minimum output=max_lst_date

# Remove month_max_lst strds
# we were only interested in the resulting aggregated map
t.remove -rf inputs=month_max_lst

# Note that the flags "-rf" force (immediate) removal of both
# the STRDS and the maps registered in it.


## Display maps in a wx monitor

# Open a monitor
d.mon wx0

# Display the raster map
d.rast map=max_lst_date

# Display boundary vector map
d.vect map=nc_state type=boundary color=#4D4D4D width=2
```

Which is the month of the maximum LST?

```
# we were only interested in the resulting aggregated map
t.remove -rf inputs=month_max_lst

# Note that the flags "-rf" force (immediate) removal of both
# the STRDS and the maps registered in it.



## Display maps in a wx monitor

# Open a monitor
d.mon wx0

# Display the raster map
d.rast map=max_lst_date

# Display boundary vector map
d.vect map=nc_state type=boundary color=#4D4D4D width=2

# Add raster legend
d.legend -t raster=max_lst_date title="Month" \
  labelnum=6 title_fontsize=20 font=sans fontsize=18
```

Which is the month of the maximum LST?

```
# the STRDS and the maps registered in it.


## Display maps in a wx monitor

# Open a monitor
d.mon wx0

# Display the raster map
d.rast map=max_lst_date

# Display boundary vector map
d.vect map=nc_state type=boundary color=#4D4D4D width=2

# Add raster legend
d.legend -t raster=max_lst_date title="Month" \
  labelnum=6 title_fontsize=20 font=sans fontsize=18


# Add scale bar
d.barscale length=200 units=kilometers segment=4 fontsize=14
```

Which is the month of the maximum LST?

```
## Display maps in a wx monitor

# Open a monitor
d.mon wx0

# Display the raster map
d.rast map=max_lst_date

# Display boundary vector map
d.vect map=nc_state type=boundary color=#4D4D4D width=2

# Add raster legend
d.legend -t raster=max_lst_date title="Month" \
  labelnum=6 title_fontsize=20 font=sans fontsize=18


# Add scale bar
d.barscale length=200 units=kilometers segment=4 fontsize=14

# Add North arrow
d.northarrow style=1b text_color=black
```

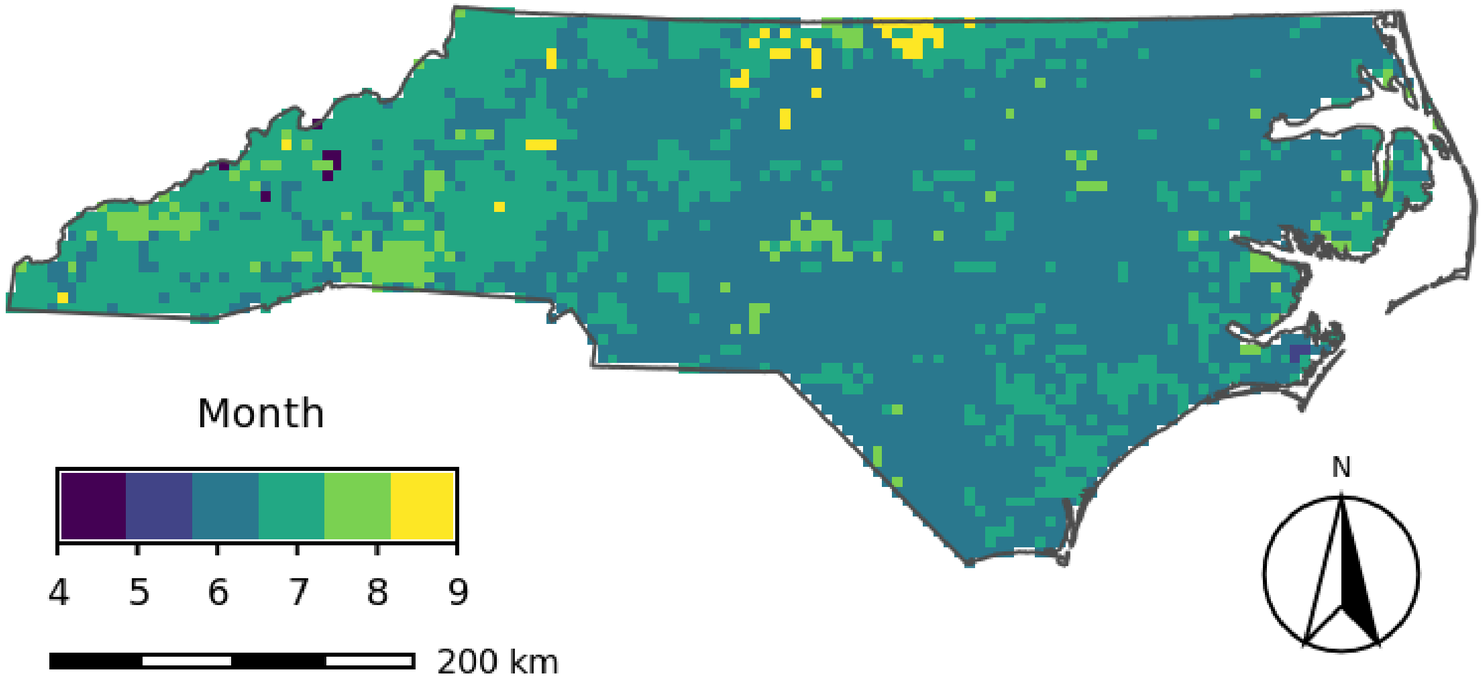
Which is the month of the maximum LST?

```
d.mon wx0

# Display the raster map
d.rast map=max_lst_date

# Display boundary vector map
d.vect map=nc_state type=boundary color=#4D4D4D width=2

# Add raster legend
d.legend -t raster=max_lst_date title="Month" \
  labelnum=6 title_fontsize=20 font=sans fontsize=18


# Add scale bar
d.barscale length=200 units=kilometers segment=4 fontsize=14

# Add North arrow
d.northarrow style=1b text_color=black

# Add text
d.text -b text="Month of maximum LST 2015-2017" \
```

Which is the month of the maximum LST?

```
d.rast map=max_lst_date

# Display boundary vector map
d.vect map=nc_state type=boundary color=#4D4D4D width=2

# Add raster legend
d.legend -t raster=max_lst_date title="Month" \
  labelnum=6 title_fontsize=20 font=sans fontsize=18


# Add scale bar
d.barscale length=200 units=kilometers segment=4 fontsize=14

# Add North arrow
d.northarrow style=1b text_color=black

# Add text
d.text -b text="Month of maximum LST 2015-2017" \
  color=black align=cc font=sans size=12


## Temporal aggregation (with granularity)
```

Month of maximum LST 2015-2017



Temporal aggregation 2: using granularity

t.rast.aggregate

- Aggregates raster maps in STRDS with different **granularities**
- *where* option allows to set specific dates for the aggregation
- Different methods available: average, minimum, maximum, median, mode, etc.

From monthly to seasonal LST

```
#!/bin/bash
############################################################################
# Commands for the TGRASS lecture at GEOSTAT Summer School in Prague
# Author: Veronica Andreo
# Date: July - August, 2018 - Edited October, 2018 and July, 2019
############################################################################


########### Before the workshop (done for you in advance) #############

#~ # Install i.modis add-on (requires pymodis library - www.pymodis.org)
#~ g.extension extension=i.modis

#~ # Download and import MODIS LST data
#~ # Note: User needs to be registered at Earthdata:
#~ # https://urs.earthdata.nasa.gov/users/new
#~ i.modis.download settings=$HOME/gisdata/SETTING \
  #~ product=lst_terra_monthly_5600 \
  #~ tile=h11v05 \
  #~ startday=2015-01-01 endday=2017-12-31 \
  #~ folder=/tmp
```

From monthly to seasonal LST

```
# Add scale bar
d.barscale length=200 units=kilometers segment=4 fontsize=14

# Add North arrow
d.northarrow style=1b text_color=black

# Add text
d.text -b text="Month of maximum LST 2015-2017" \
  color=black align=cc font=sans size=12


## Temporal aggregation (with granularity)

# 3-month mean LST
t.rast.aggregate input=LST_Day_monthly_celsius \
  output=LST_Day_mean_3month \
  basename=LST_Day_mean_3month suffix=gran \
  method=average granularity="3 months"
```

From monthly to seasonal LST

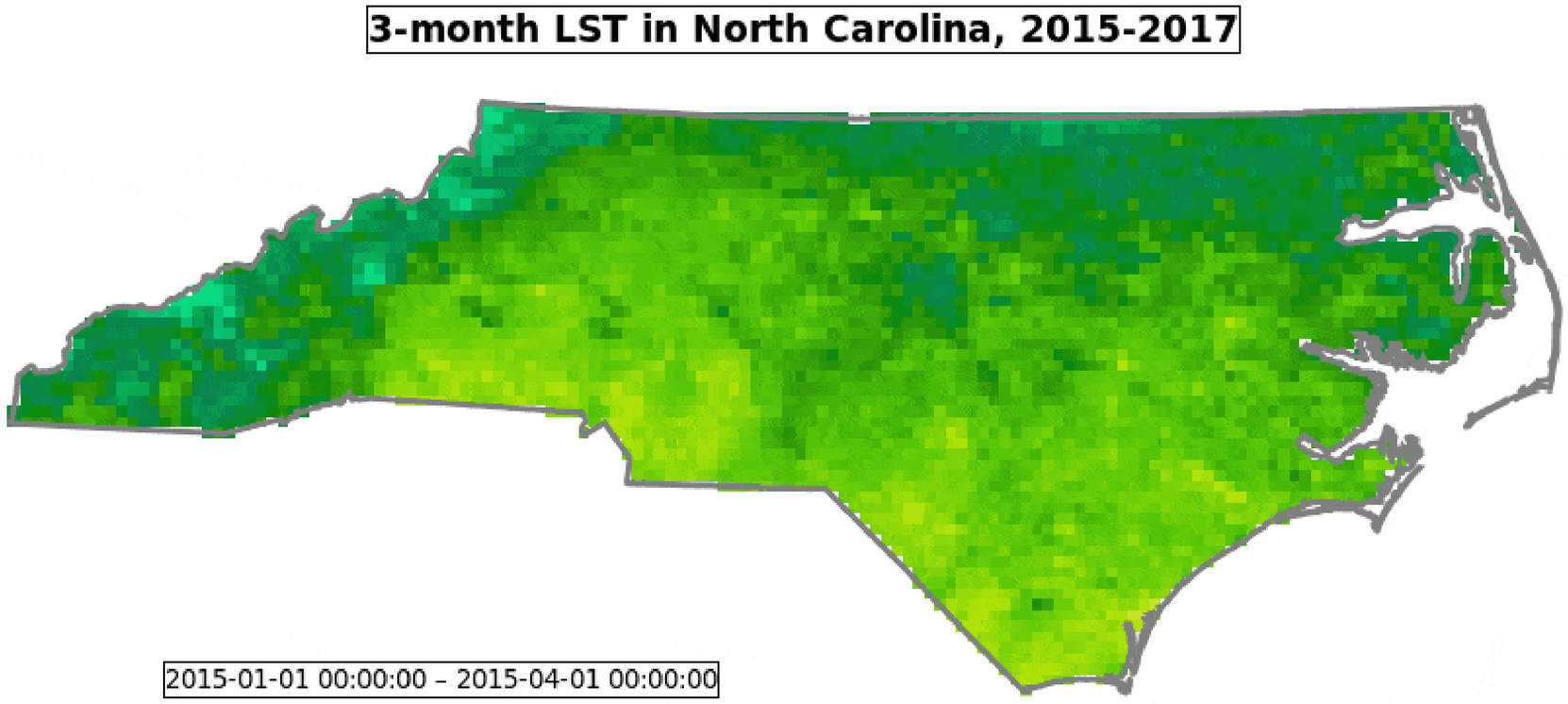
```
# Add North arrow
d.northarrow style=1b text_color=black

# Add text
d.text -b text="Month of maximum LST 2015-2017" \
  color=black align=cc font=sans size=12



## Temporal aggregation (with granularity)

# 3-month mean LST
t.rast.aggregate input=LST_Day_monthly_celsius \
  output=LST_Day_mean_3month \
  basename=LST_Day_mean_3month suffix=gran \
  method=average granularity="3 months"

# Check info
t.info input=LST_Day_mean_3month

# Check map list
t.rast.list input=LST_Day_mean_3month
```

From monthly to seasonal LST

```
# 3-month mean LST
t.rast.aggregate input=LST_Day_monthly_celsius \
  output=LST_Day_mean_3month \
  basename=LST_Day_mean_3month suffix=gran \
  method=average granularity="3 months"

# Check info
t.info input=LST_Day_mean_3month

# Check map list
t.rast.list input=LST_Day_mean_3month

#~ name|mapset|start_time|end_time
#~ LST_Day_mean_3month_2015_01|modis_lst|2015-01-01 00:00:00|2015-04-01 00:00:00
#~ LST_Day_mean_3month_2015_04|modis_lst|2015-04-01 00:00:00|2015-07-01 00:00:00
#~ LST_Day_mean_3month_2015_07|modis_lst|2015-07-01 00:00:00|2015-10-01 00:00:00
#~ LST_Day_mean_3month_2015_10|modis_lst|2015-10-01 00:00:00|2016-01-01 00:00:00
#~ LST_Day_mean_3month_2016_01|modis_lst|2016-01-01 00:00:00|2016-04-01 00:00:00
#~ LST_Day_mean_3month_2016_04|modis_lst|2016-04-01 00:00:00|2016-07-01 00:00:00
#~ LST_Day_mean_3month_2016_07|modis_lst|2016-07-01 00:00:00|2016-10-01 00:00:00

#~ LST_Day_mean_3month_2016_10|modis_lst|2016-10-01 00:00:00|2017-01-01 00:00:00
```

From monthly to seasonal LST

```
#~ d.frame -c frame=third at=50,100,0,50
#~ d.rast map=LST_Day_mean_3month_2015_01
#~ d.vect map=nc_state type=boundary color=#4D4D4D width=2
#~ d.text text='Jan-Mar 2015' color=black font=sans size=10


#~ # create a fourth frame
#~ d.frame -c frame=fourth at=50,100,50,100
#~ d.rast map=LST_Day_mean_3month_2015_04
#~ d.vect map=nc_state type=boundary color=#4D4D4D width=2
#~ d.text text='Apr-Jun 2015' color=black font=sans size=10

#~ # release monitor
#~ d.mon -r


## Time series animation

# Animation of seasonal LST
g.gui.animation strds=LST_Day_mean_3month
```

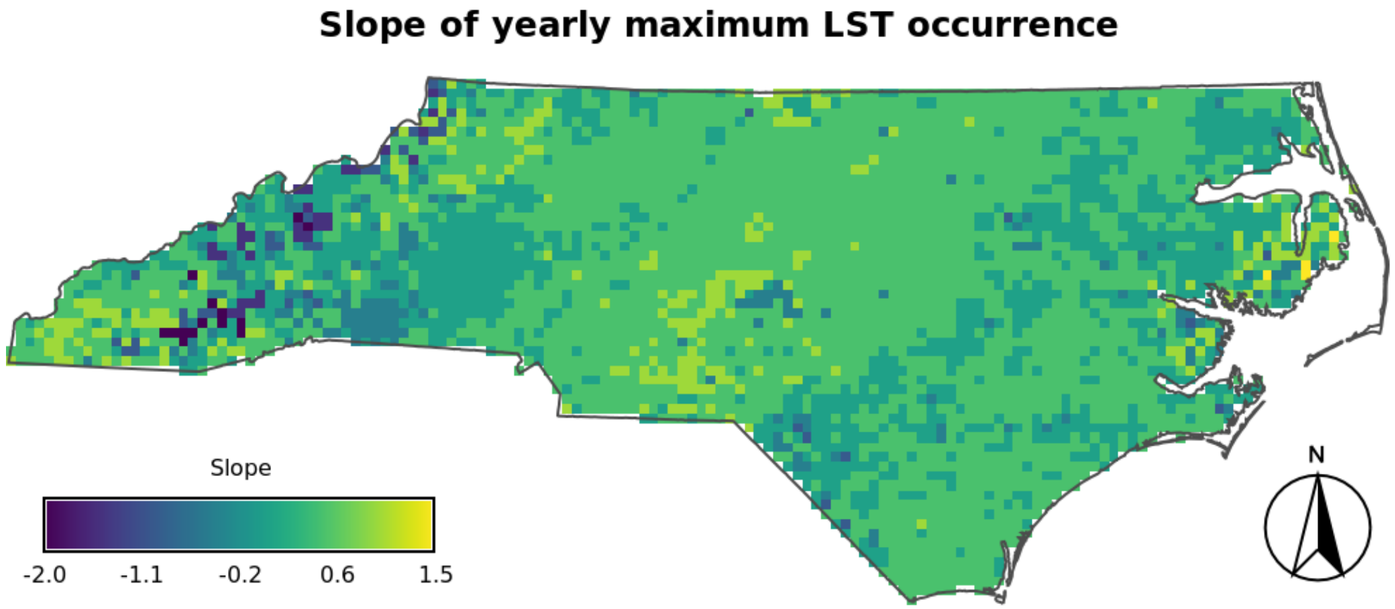


See g.gui.animation manual for further options and tweaks

Task: *Now that you know t.rast.aggregate, extract the month of maximum LST per year and then test if there's any positive or negative trend, i.e., if maximum LST values are observed later or earlier with time (years)*

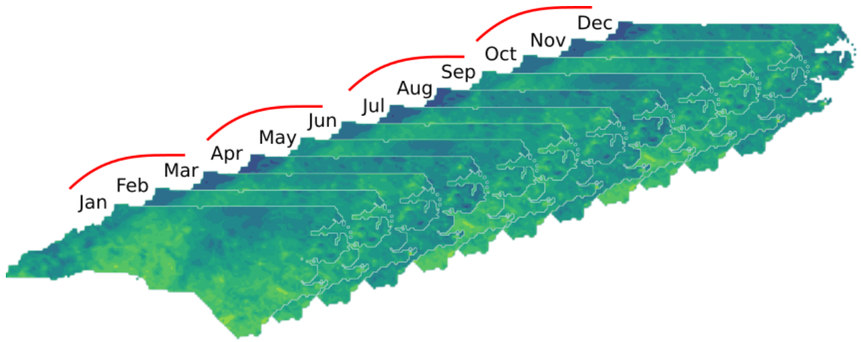
One solution could be...

```
t.rast.aggregate input=LST_Day_monthly_celsius \
  output=month_max_LST_per_year \
  basename=month_max_LST suffix=gran \
  method=max_raster granularity="1 year"

t.rast.series input=month_max_LST_per_year \
  output=slope_month_max_LST \
  method=slope
```


Slope of yearly maximum LST occurrence
Slope
-2.0
-1.1
-0.2
0.6
1.5
N

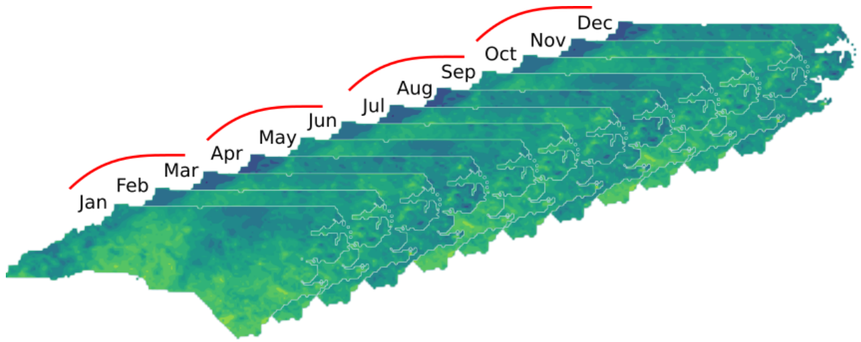
Aggregation vs Climatology

Aggregation vs Climatology

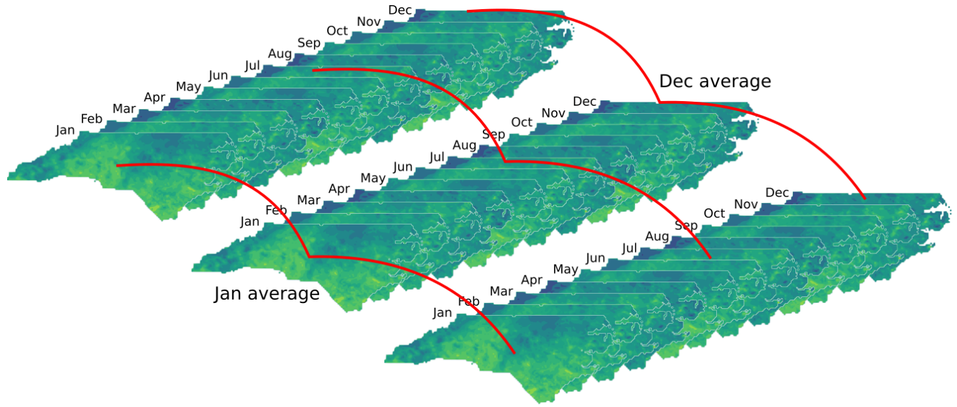


Granularity aggregation

Aggregation vs Climatology



Granularity aggregation



Climatology-type aggregation

Monthly climatologies

```
#!/bin/bash
#############################################################################
# Commands for the TGRASS lecture at GEOSTAT Summer School in Prague
# Author: Veronica Andreo
# Date: July - August, 2018 - Edited October, 2018 and July, 2019
#############################################################################


########### Before the workshop (done for you in advance) #############

#~ # Install i.modis add-on (requires pymodis library - www.pymodis.org)
#~ g.extension extension=i.modis

#~ # Download and import MODIS LST data
#~ # Note: User needs to be registered at Earthdata:
#~ # https://urs.earthdata.nasa.gov/users/new
#~ i.modis.download settings=$HOME/gisdata/SETTING \
  #~ product=lst_terra_monthly_5600 \
  #~ tile=h11v05 \
  #~ startday=2015-01-01 endday=2017-12-31 \
  #~ folder=/tmp
```

Monthly climatologies

```
#~ d.rast map=LST_Day_mean_3month_2015_04
#~ d.vect map=nc_state type=boundary color=#4D4D4D width=2
#~ d.text text='Apr-Jun 2015' color=black font=sans size=10

#~ # release monitor
#~ d.mon -r



## Time series animation

# Animation of seasonal LST
g.gui.animation strds=LST_Day_mean_3month



## Long term monthly averages (Monthly climatoligies)


# January average LST
t.rast.series input=LST_Day_monthly_celsius method=average \
  where="strftime('%m', start_time)='01'" \
  output=LST_average_jan
```

Monthly climatologies

```
## Time series animation

# Animation of seasonal LST
g.gui.animation strds=LST_Day_mean_3month


## Long term monthly averages (Monthly climatoligies)

# January average LST
t.rast.series input=LST_Day_monthly_celsius method=average \
  where="strftime('%m', start_time)='01'" \
  output=LST_average_jan

# for all months - *nix
for MONTH in $(seq -w 1 12) ; do
 t.rast.series input=LST_Day_monthly_celsius method=average \
  where="strftime('%m', start_time)='${MONTH}'" \
  output=LST_average_${MONTH}
done
```

Monthly climatologies

```
## Long term monthly averages (Monthly climatoligies)

# January average LST
t.rast.series input=LST_Day_monthly_celsius method=average \
  where="strftime('%m', start_time)='01'" \
  output=LST_average_jan

# for all months - *nix
for MONTH in $(seq -w 1 12) ; do
 t.rast.series input=LST_Day_monthly_celsius method=average \
  where="strftime('%m', start_time)='${MONTH}'" \
  output=LST_average_${MONTH}
done

# for all months - windows
FOR %c IN (01,02,03,04,05,06,07,08,09,10,11,12) DO (

 t.rast.series input=LST_Day_monthly_celsius method=average \
  where="strftime('%m', start_time)='%c'" \
  output=LST_average_%c
```

Annual standardized anomalies

$$StdAnomaly_i = \frac{Average_i - Average}{SD}$$

We need:

- overall average and standard deviation
- annual averages

Annual anomalies

```
#!/bin/bash
############################################################################
# Commands for the TGRASS lecture at GEOSTAT Summer School in Prague
# Author: Veronica Andreo
# Date: July - August, 2018 - Edited October, 2018 and July, 2019
############################################################################


########### Before the workshop (done for you in advance) #############

#~ # Install i.modis add-on (requires pymodis library - www.pymodis.org)
#~ g.extension extension=i.modis

#~ # Download and import MODIS LST data
#~ # Note: User needs to be registered at Earthdata:
#~ # https://urs.earthdata.nasa.gov/users/new
#~ i.modis.download settings=$HOME/gisdata/SETTING \
  #~ product=lst_terra_monthly_5600 \
  #~ tile=h11v05 \
  #~ startday=2015-01-01 endday=2017-12-31 \
  #~ folder=/tmp
```

Annual anomalies

```
for MONTH in $(seq -w 1 12) ; do
 t.rast.series input=LST_Day_monthly_celsius method=average \
  where="strftime('%m', start_time)='${MONTH}'" \
  output=LST_average_${MONTH}
done

# for all months - windows
FOR %c IN (01,02,03,04,05,06,07,08,09,10,11,12) DO (

 t.rast.series input=LST_Day_monthly_celsius method=average \
  where="strftime('%m', start_time)='%c'" \
  output=LST_average_%c
)



## Anomalies


# Get general average
t.rast.series input=LST_Day_monthly_celsius \
 method=average output=LST_average
```

Annual anomalies

```
  output=LST_average_${MONTH}
done

# for all months - windows
FOR %c IN (01,02,03,04,05,06,07,08,09,10,11,12) DO (

 t.rast.series input=LST_Day_monthly_celsius method=average \
  where="strftime('%m', start_time)='%c'" \
  output=LST_average_%c
)


## Anomalies

# Get general average
t.rast.series input=LST_Day_monthly_celsius \
 method=average output=LST_average

# Get general SD
t.rast.series input=LST_Day_monthly_celsius \
 method=stddev output=LST_sd
```

Annual anomalies

```
 t.rast.series input=LST_Day_monthly_celsius method=average \
  where="strftime('%m', start_time)='%c'" \
  output=LST_average_%c
)


## Anomalies


# Get general average
t.rast.series input=LST_Day_monthly_celsius \
 method=average output=LST_average

# Get general SD
t.rast.series input=LST_Day_monthly_celsius \
 method=stddev output=LST_sd

# Get annual averages
t.rast.aggregate input=LST_Day_monthly_celsius \
 method=average granularity="1 years" \
 output=LST_yearly_average basename=LST_yearly_average
```

Annual anomalies

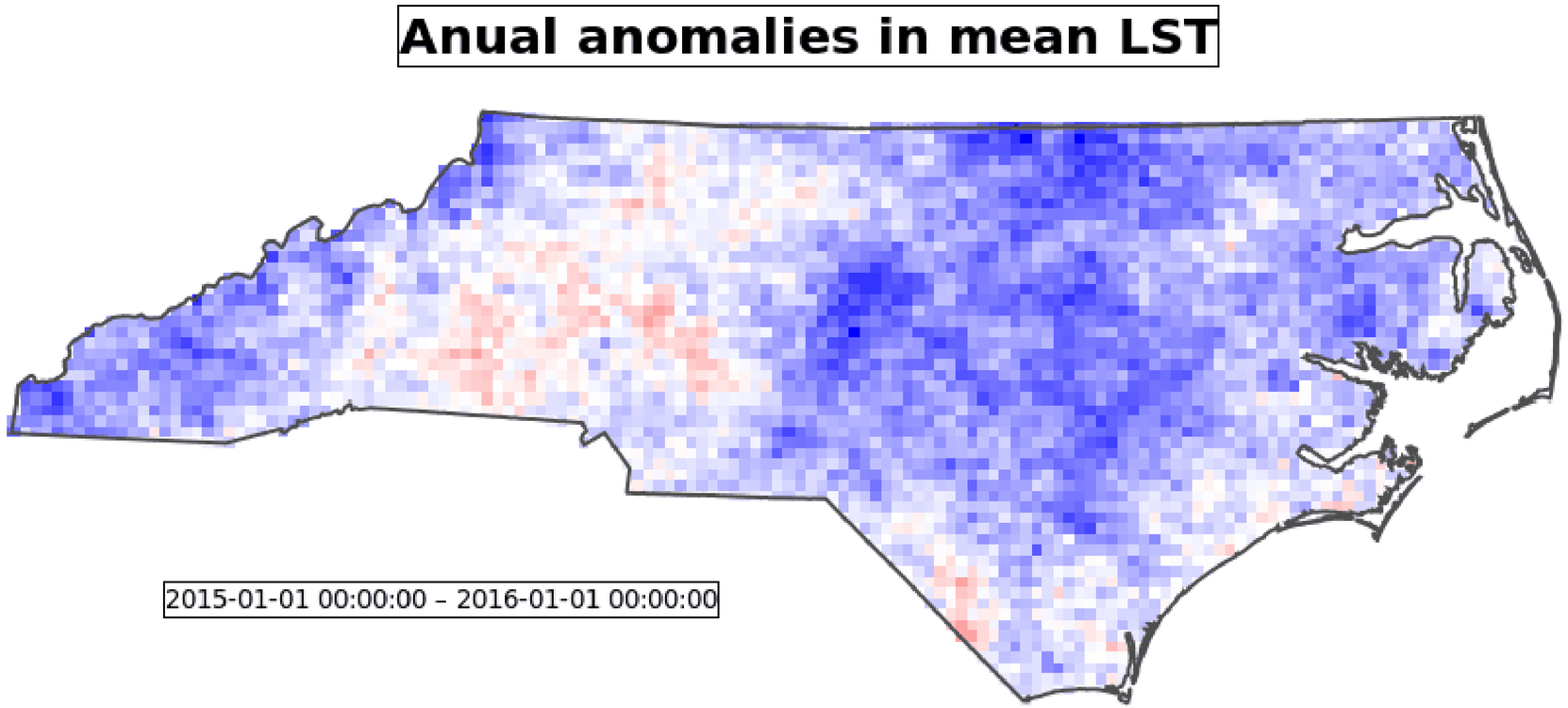
```
## Anomalies

# Get general average
t.rast.series input=LST_Day_monthly_celsius \
 method=average output=LST_average

# Get general SD
t.rast.series input=LST_Day_monthly_celsius \
 method=stddev output=LST_sd

# Get annual averages
t.rast.aggregate input=LST_Day_monthly_celsius \
 method=average granularity="1 years" \
 output=LST_yearly_average basename=LST_yearly_average

# Estimate annual anomalies
t.rast.algebra basename=LST_year_anomaly suffix=gran \
 expression="LST_year_anomaly = (LST_yearly_average - map(LST_average)) / map(LST_s
```

Annual anomalies

```
# Get general average
t.rast.series input=LST_Day_monthly_celsius \
 method=average output=LST_average

# Get general SD
t.rast.series input=LST_Day_monthly_celsius \
 method=stddev output=LST_sd

# Get annual averages
t.rast.aggregate input=LST_Day_monthly_celsius \
 method=average granularity="1 years" \
 output=LST_yearly_average basename=LST_yearly_average

# Estimate annual anomalies
t.rast.algebra basename=LST_year_anomaly suffix=gran \
 expression="LST_year_anomaly = (LST_yearly_average - map(LST_average)) / map(LST_s

# Set differences color table
t.rast.colors input=LST_year_anomaly color=differences
```

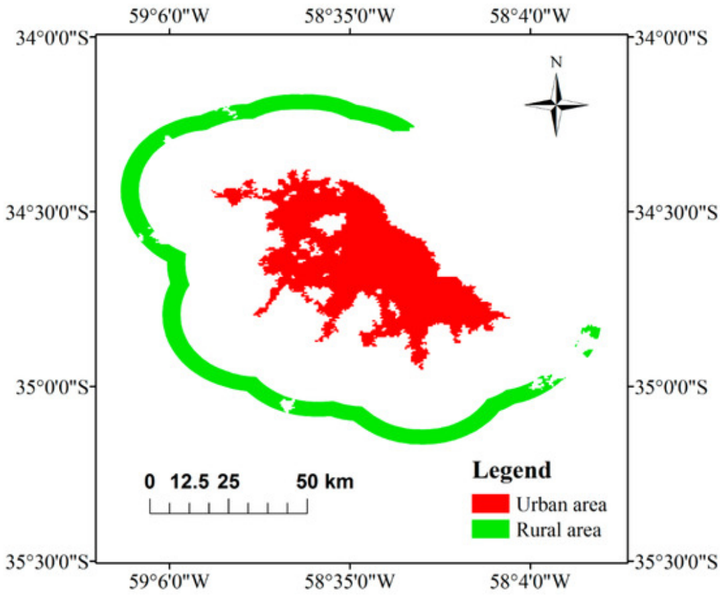
Annual anomalies

```
 method=average output=LST_average

# Get general SD
t.rast.series input=LST_Day_monthly_celsius \
 method=stddev output=LST_sd

# Get annual averages
t.rast.aggregate input=LST_Day_monthly_celsius \
 method=average granularity="1 years" \
 output=LST_yearly_average basename=LST_yearly_average

# Estimate annual anomalies
t.rast.algebra basename=LST_year_anomaly suffix=gran \
 expression="LST_year_anomaly = (LST_yearly_average - map(LST_average)) / map(LST_s

# Set differences color table
t.rast.colors input=LST_year_anomaly color=differences

# Animation of annual anomalies
g.gui.animation strds=LST_year_anomaly
```


Anual anomalies in mean LST
2015-01-01 00:00:00 – 2016-01-01 00:00:00

Surface Urban Heat Island (SUHI)

- Air temperature of an urban area is higher than that in nearby areas
- UHI has negative effects on water and air quality, biodiversity, human health, and climate
- SUHI is also highly related to health, since it influences UHI



SUHI and surrounding rural area for Buenos Aires city. Source Wu et al, 2019.

Zonal statistics in raster time series

v.strds.stats

- Allows to obtain spatially aggregated time series data for polygons in a vector map

Summer SUHI for the city of Raleigh and surroundings

```
#!/bin/bash
##############################################################################
# Commands for the TGRASS lecture at GEOSTAT Summer School in Prague
# Author: Veronica Andreo
# Date: July - August, 2018 - Edited October, 2018 and July, 2019
##############################################################################


########### Before the workshop (done for you in advance) #############

#~ # Install i.modis add-on (requires pymodis library - www.pymodis.org)
#~ g.extension extension=i.modis

#~ # Download and import MODIS LST data
#~ # Note: User needs to be registered at Earthdata:
#~ # https://urs.earthdata.nasa.gov/users/new
#~ i.modis.download settings=$HOME/gisdata/SETTING \
  #~ product=lst_terra_monthly_5600 \
  #~ tile=h11v05 \
  #~ startday=2015-01-01 endday=2017-12-31 \
  #~ folder=/tmp
```

Summer SUHI for the city of Raleigh and surroundings

```
t.rast.aggregate input=LST_Day_monthly_celsius \
 method=average granularity="1 years" \
 output=LST_yearly_average basename=LST_yearly_average

# Estimate annual anomalies
t.rast.algebra basename=LST_year_anomaly suffix=gran \
 expression="LST_year_anomaly = (LST_yearly_average - map(LST_average)) / map(LST_s

# Set differences color table
t.rast.colors input=LST_year_anomaly color=differences

# Animation of annual anomalies
g.gui.animation strds=LST_year_anomaly


## Extract zonal statistics for areas


# Install v.strds.stats add-on
g.extension extension=v.strds.stats
```

Summer SUHI for the city of Raleigh and surroundings

```
# Estimate annual anomalies
t.rast.algebra basename=LST_year_anomaly suffix=gran \
 expression="LST_year_anomaly = (LST_yearly_average - map(LST_average)) / map(LST_s

# Set differences color table
t.rast.colors input=LST_year_anomaly color=differences

# Animation of annual anomalies
g.gui.animation strds=LST_year_anomaly



## Extract zonal statistics for areas



# Install v.strds.stats add-on
g.extension extension=v.strds.stats

# List maps in seasonal time series
t.rast.list LST_Day_mean_3month
```

Summer SUHI for the city of Raleigh and surroundings

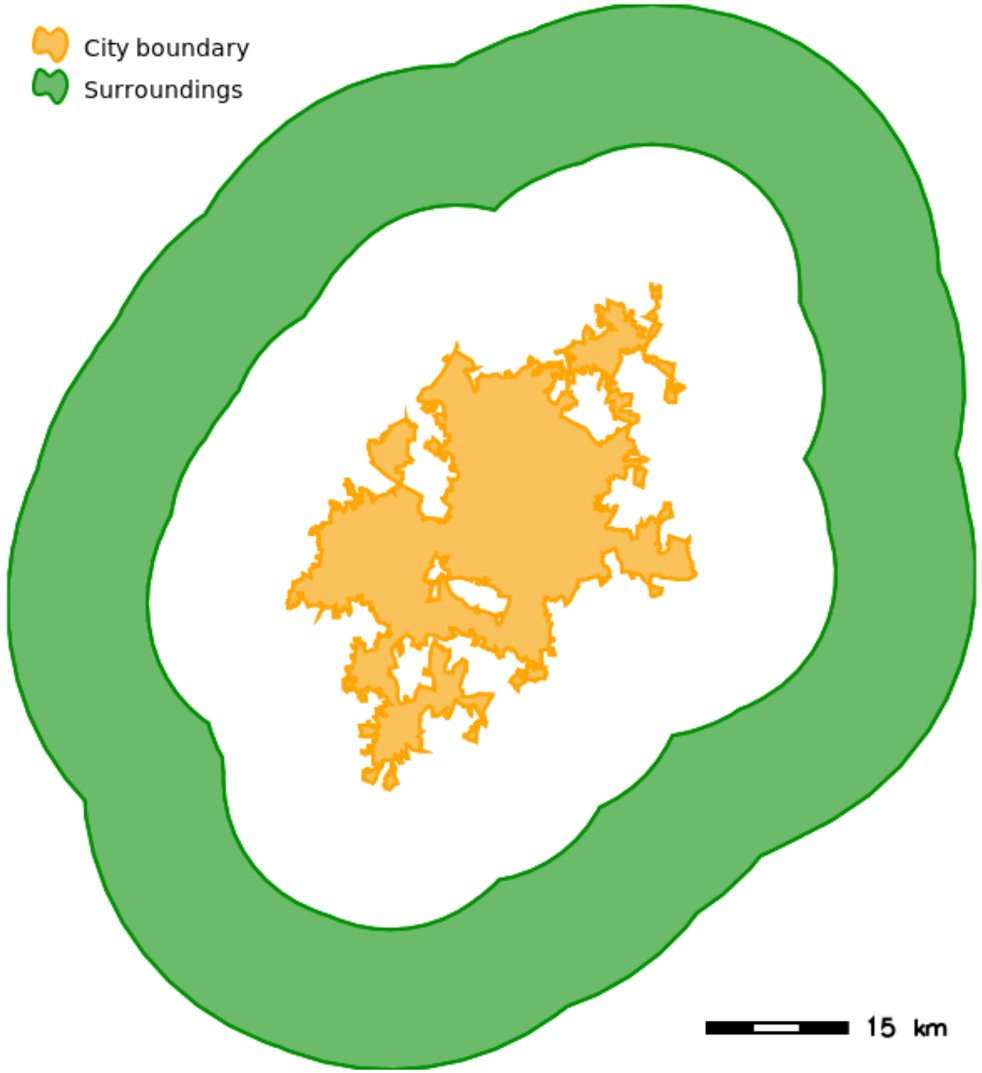
```
# Install v.strds.stats add-on
g.extension extension=v.strds.stats

# List maps in seasonal time series
t.rast.list LST_Day_mean_3month

# Extract summer average LST for Raleigh urban area
v.strds.stats input=urbanarea \
  strds=LST_Day_mean_3month \
  where="NAME == 'Raleigh'" \
  t_where="strftime('%m', start_time)='07'" \
  method=average \
  output=raleigh_summer_lst


## SUHI vs surroundings from 2015 to 2017

# Create outside buffer - 30km
v.buffer input=raleigh_summer_lst \
  distance=30000 \
```

Summer SUHI for the city of Raleigh and surroundings

```
r.rast.list LST_Day_mean_3month

# Extract summer average LST for Raleigh urban area
v.strds.stats input=urbanarea \
  strds=LST_Day_mean_3month \
  where="NAME == 'Raleigh'" \
  t_where="strftime('%m', start_time)='07'" \
  method=average \
  output=raleigh_summer_lst



## SUHI vs surroundings from 2015 to 2017


# Create outside buffer - 30km
v.buffer input=raleigh_summer_lst \
  distance=30000 \
  output=raleigh_summer_lst_buf30

# Create inside buffer - 15km
v.buffer input=raleigh_summer_lst \
  distance=15000 \
```

Summer SUHI for the city of Raleigh and surroundings

```
  t_where="strftime('%m', start_time)='07'" \
  method=average \
  output=raleigh_summer_lst


## SUHI vs surroundings from 2015 to 2017

# Create outside buffer - 30km
v.buffer input=raleigh_summer_lst \
  distance=30000 \
  output=raleigh_summer_lst_buf30

# Create inside buffer - 15km
v.buffer input=raleigh_summer_lst \
  distance=15000 \
  output=raleigh_summer_lst_buf15

# Remove 15km buffer area from the 30km buffer area
v.overlay ainput=raleigh_summer_lst_buf15 \
  binput=raleigh_summer_lst_buf30 \
  operator=xor \
```



Raleigh city boundary and surrounding rural area

Summer SUHI for the city of Raleigh and surroundings

```
#!/bin/bash
##############################################################################
# Commands for the TGRASS lecture at GEOSTAT Summer School in Prague
# Author: Veronica Andreo
# Date: July - August, 2018 - Edited October, 2018 and July, 2019
##############################################################################


########### Before the workshop (done for you in advance) #############

#~ # Install i.modis add-on (requires pymodis library - www.pymodis.org)
#~ g.extension extension=i.modis

#~ # Download and import MODIS LST data
#~ # Note: User needs to be registered at Earthdata:
#~ # https://urs.earthdata.nasa.gov/users/new
#~ i.modis.download settings=$HOME/gisdata/SETTING \
  #~ product=lst_terra_monthly_5600 \
  #~ tile=h11v05 \
  #~ startday=2015-01-01 endday=2017-12-31 \
  #~ folder=/tmp
```

Summer SUHI for the city of Raleigh and surroundings

```
## SUHI vs surroundings from 2015 to 2017

# Create outside buffer - 30km
v.buffer input=raleigh_summer_lst \
  distance=30000 \
  output=raleigh_summer_lst_buf30

# Create inside buffer - 15km
v.buffer input=raleigh_summer_lst \
  distance=15000 \
  output=raleigh_summer_lst_buf15

# Remove 15km buffer area from the 30km buffer area
v.overlay ainput=raleigh_summer_lst_buf15 \
  binput=raleigh_summer_lst_buf30 \
  operator=xor \
  output=raleigh_surr


# Extract zonal stats for Raleigh surroundings
```

Summer SUHI for the city of Raleigh and surroundings

```
## SUHI vs surroundings from 2015 to 2017

# Create outside buffer - 30km
v.buffer input=raleigh_summer_lst \
  distance=30000 \
  output=raleigh_summer_lst_buf30

# Create inside buffer - 15km
v.buffer input=raleigh_summer_lst \
  distance=15000 \
  output=raleigh_summer_lst_buf15

# Remove 15km buffer area from the 30km buffer area
v.overlay ainput=raleigh_summer_lst_buf15 \
  binput=raleigh_summer_lst_buf30 \
  operator=xor \
  output=raleigh_surr
```

We will use **R** and **RStudio** to create a nice and easy plot with the resulting vector maps

Download the R code for this part

In the GRASS GIS terminal type:

```
rstudio &
```

Plotting GRASS GIS maps in R

```
#####################################################################
# Commands for the FOSS4G workshop in Bucharest
# Author: Veronica Andreo
# Date: July, 2019
#####################################################################

# Load rgrass library
library(rgrass7)
library(sf)

# List available vectors
execGRASS("g.list", parameters = list(type="vector", mapset="."))

# Read in GRASS vector maps as sf
use_sf()
raleigh_summer_lst <- readVECT("raleigh_summer_lst")
raleigh_surr_summer_lst <- readVECT("raleigh_surr_summer_lst")

# Remove columns we don't need
raleigh_summer_lst <- raleigh_summer_lst[,-c(2:6)]
raleigh_surr_summer_lst <- raleigh_surr_summer_lst[,-c(2:3)]
```

Plotting GRASS GIS maps in R

```
##############################################################################
# Commands for the FOSS4G workshop in Bucharest
# Author: Veronica Andreo
# Date: July, 2019
##############################################################################


# Load rgrass library
library(rgrass7)
library(sf)

# List available vectors
execGRASS("g.list", parameters = list(type="vector", mapset="."))

# Read in GRASS vector maps as sf
use_sf()
raleigh_summer_lst <- readVECT("raleigh_summer_lst")
raleigh_surr_summer_lst <- readVECT("raleigh_surr_summer_lst")

# Remove columns we don't need
raleigh_summer_lst <- raleigh_summer_lst[,-c(2:6)]
raleigh_surr_summer_lst <- raleigh_surr_summer_lst[,-c(2:3)]
```

Plotting GRASS GIS maps in R

```
# Commands for the FOSS4G workshop in Bucharest
# Author: Veronica Andreo
# Date: July, 2019
##############################################################################


# Load rgrass library
library(rgrass7)
library(sf)

# List available vectors
execGRASS("g.list", parameters = list(type="vector", mapset="."))

# Read in GRASS vector maps as sf
use_sf()
raleigh_summer_lst <- readVECT("raleigh_summer_lst")
raleigh_surr_summer_lst <- readVECT("raleigh_surr_summer_lst")

# Remove columns we don't need
raleigh_summer_lst <- raleigh_summer_lst[,-c(2:6)]
raleigh_surr_summer_lst <- raleigh_surr_summer_lst[,-c(2:3)]
```

Plotting GRASS GIS maps in R

```
# Load rgrass library
library(rgrass7)
library(sf)

# List available vectors
execGRASS("g.list", parameters = list(type="vector", mapset="."))

# Read in GRASS vector maps as sf
use_sf()
raleigh_summer_lst <- readVECT("raleigh_summer_lst")
raleigh_surr_summer_lst <- readVECT("raleigh_surr_summer_lst")

# Remove columns we don't need
raleigh_summer_lst <- raleigh_summer_lst[,-c(2:6)]
raleigh_surr_summer_lst <- raleigh_surr_summer_lst[,-c(2:3)]

# Paste the 2 vectors together
raleigh <- rbind(raleigh_summer_lst,raleigh_surr_summer_lst)
```

Plotting GRASS GIS maps in R

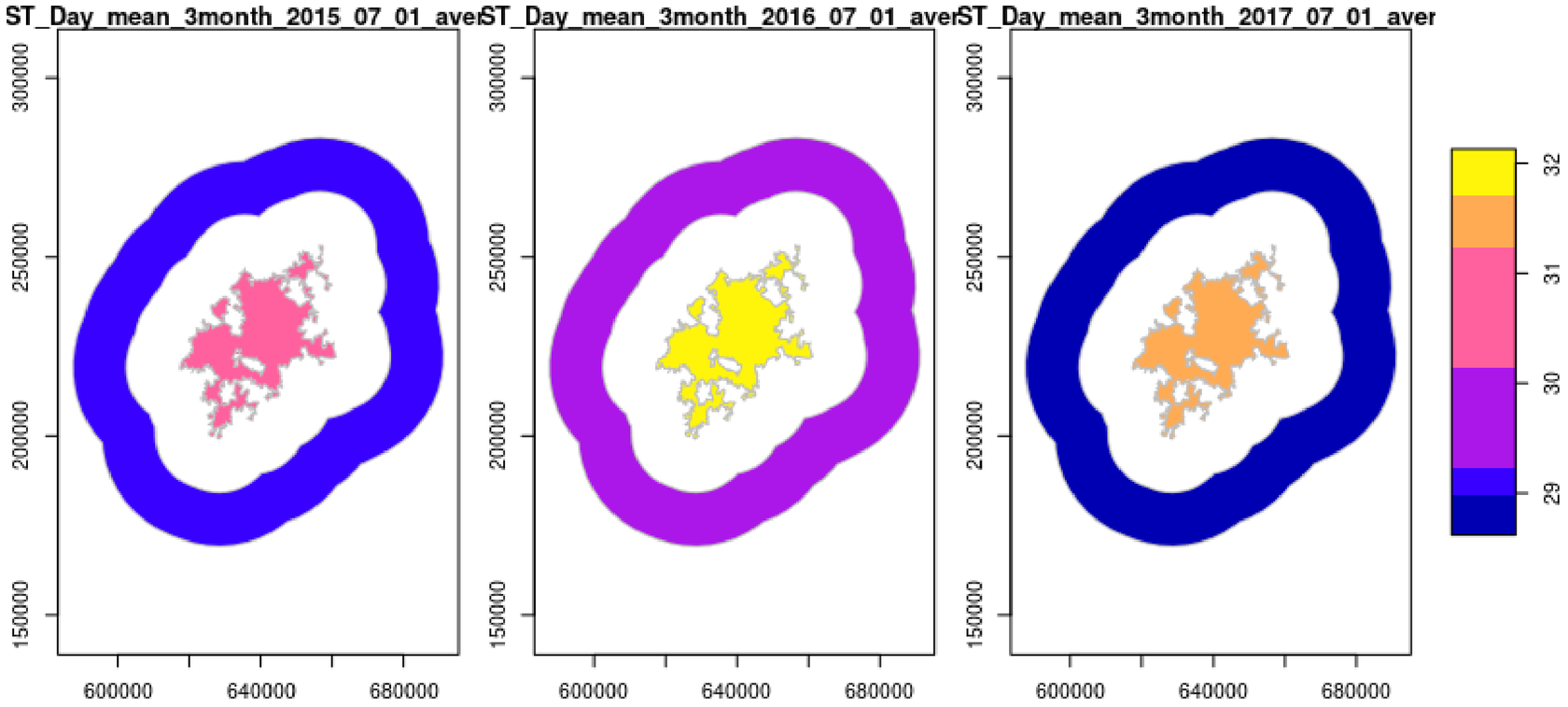
```
# List available vectors
execGRASS("g.list", parameters = list(type="vector", mapset="."))

# Read in GRASS vector maps as sf
use_sf()
raleigh_summer_lst <- readVECT("raleigh_summer_lst")
raleigh_surr_summer_lst <- readVECT("raleigh_surr_summer_lst")

# Remove columns we don't need
raleigh_summer_lst <- raleigh_summer_lst[,-c(2:6)]
raleigh_surr_summer_lst <- raleigh_surr_summer_lst[,-c(2:3)]

# Paste the 2 vectors together
raleigh <- rbind(raleigh_summer_lst,raleigh_surr_summer_lst)

# Quick sf plot

plot(raleigh[c(2:4)], border = 'grey', axes = TRUE, key.pos = 4)
```

Plotting GRASS GIS maps in R

```
# Read in GRASS vector maps as sf
use_sf()
raleigh_summer_lst <- readVECT("raleigh_summer_lst")
raleigh_surr_summer_lst <- readVECT("raleigh_surr_summer_lst")

# Remove columns we don't need
raleigh_summer_lst <- raleigh_summer_lst[,-c(2:6)]
raleigh_surr_summer_lst <- raleigh_surr_summer_lst[,-c(2:3)]

# Paste the 2 vectors together
raleigh <- rbind(raleigh_summer_lst,raleigh_surr_summer_lst)

# Quick sf plot

plot(raleigh[c(2:4)], border = 'grey', axes = TRUE, key.pos = 4)



# Let's try with ggplot library
library(ggplot2)
library(dplyr)
library(tidyr)
```

Plotting GRASS GIS maps in R

```
raleigh_surr_summer_lst <- readVECT("raleigh_surr_summer_lst")

# Remove columns we don't need
raleigh_summer_lst <- raleigh_summer_lst[,-c(2:6)]
raleigh_surr_summer_lst <- raleigh_surr_summer_lst[,-c(2:3)]

# Paste the 2 vectors together
raleigh <- rbind(raleigh_summer_lst,raleigh_surr_summer_lst)

# Quick sf plot

plot(raleigh[c(2:4)], border = 'grey', axes = TRUE, key.pos = 4)


# Let's try with ggplot library
library(ggplot2)
library(dplyr)
library(tidyr)

# Arrange data from wide to long format
```


ST_Day_mean_3month_2015_07_01_aver
ST_Day_mean_3month_2016_07_01_aver
ST_Day_mean_3month_2017_07_01_aver
300000
250000
200000
150000
600000
640000
680000
32
31
30
29

Plotting GRASS GIS maps in R

```
#########################################################################
# Commands for the FOSS4G workshop in Bucharest
# Author: Veronica Andreo
# Date: July, 2019
#########################################################################

# Load rgrass library
library(rgrass7)
library(sf)

# List available vectors
execGRASS("g.list", parameters = list(type="vector", mapset="."))

# Read in GRASS vector maps as sf
use_sf()
raleigh_summer_lst <- readVECT("raleigh_summer_lst")
raleigh_surr_summer_lst <- readVECT("raleigh_surr_summer_lst")

# Remove columns we don't need
raleigh_summer_lst <- raleigh_summer_lst[,-c(2:6)]
raleigh_surr_summer_lst <- raleigh_surr_summer_lst[,-c(2:3)]
```

Plotting GRASS GIS maps in R

```
raleigh_surr_summer_lst <- raleigh_surr_summer_lst[, c(2:3)]

# Paste the 2 vectors together
raleigh <- rbind(raleigh_summer_lst,raleigh_surr_summer_lst)

# Quick sf plot
plot(raleigh[c(2:4)], border = 'grey', axes = TRUE, key.pos = 4)


# Let's try with ggplot library
library(ggplot2)
library(dplyr)
library(tidyr)

# Arrange data from wide to long format
raleigh2 <-
  raleigh %>%
  select(LST_Day_mean_3month_2015_07_01_average,
         LST_Day_mean_3month_2016_07_01_average,
         LST_Day_mean_3month_2017_07_01_average,
         geom) %>%
```

Plotting GRASS GIS maps in R

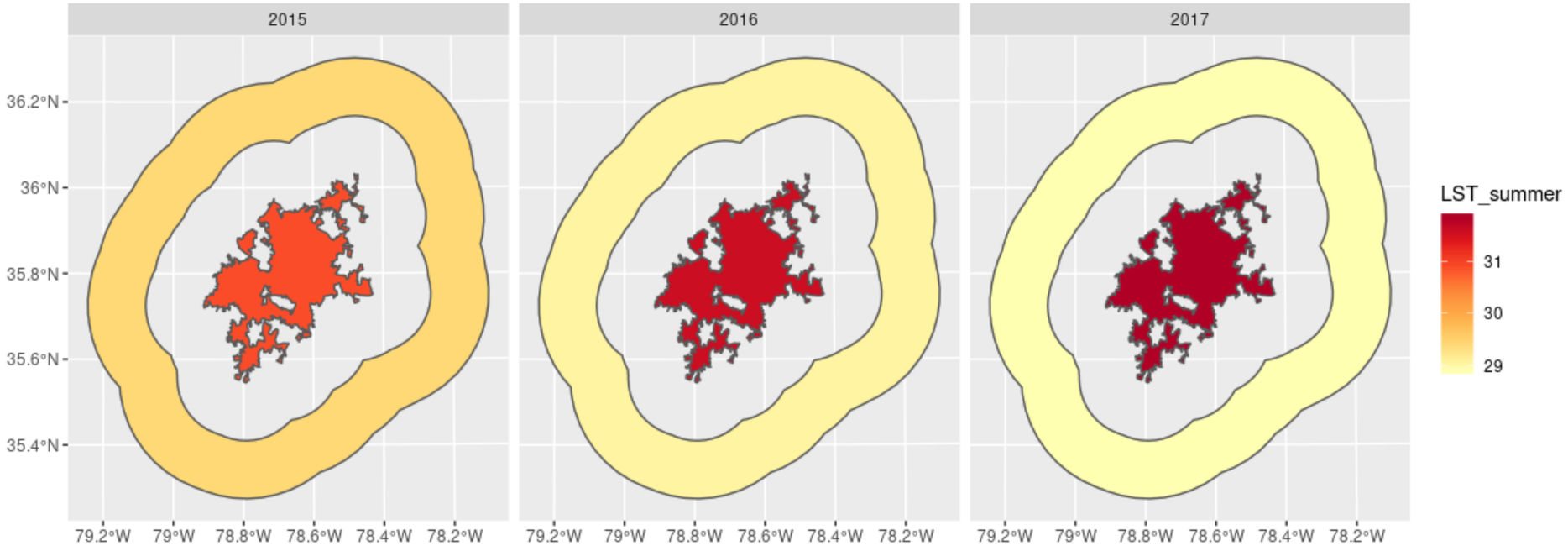
```
# Let's try with ggplot library
library(ggplot2)
library(dplyr)
library(tidyr)

# Arrange data from wide to long format
raleigh2 <-
  raleigh %>%
  select(LST_Day_mean_3month_2015_07_01_average,
         LST_Day_mean_3month_2016_07_01_average,
         LST_Day_mean_3month_2017_07_01_average,
         geom) %>%
  gather(YEAR, LST_summer, -geom)

# Replace values in YEAR column
raleigh2$YEAR <- rep(c(2015:2017),2)

# Plot
ggplot() +
```

Plotting GRASS GIS maps in R

```
library(tidyr)

# Arrange data from wide to long format
raleigh2 <-
  raleigh %>%
  select(LST_Day_mean_3month_2015_07_01_average,
         LST_Day_mean_3month_2016_07_01_average,
         LST_Day_mean_3month_2017_07_01_average,
         geom) %>%
  gather(YEAR, LST_summer, -geom)

# Replace values in YEAR column
raleigh2$YEAR <- rep(c(2015:2017),2)

# Plot
ggplot() +
  geom_sf(data = raleigh2, aes(fill = LST_summer)) +
  facet_wrap(~YEAR, ncol = 3) +
  scale_fill_distiller(palette = "YlOrRd",
                       direction = 1) +
  scale_y_continuous()
```

Plotting GRASS GIS maps in R

```
          LST_Day_mean_3month_2016_07_01_average,
          LST_Day_mean_3month_2017_07_01_average,
          geom) %>%
  gather(YEAR, LST_summer, -geom)

# Replace values in YEAR column
raleigh2$YEAR <- rep(c(2015:2017),2)

# Plot
ggplot() +
  geom_sf(data = raleigh2, aes(fill = LST_summer)) +
  facet_wrap(~YEAR, ncol = 3) +
  scale_fill_distiller(palette = "YlOrRd",
                       direction = 1) +
  scale_y_continuous()


# Let's try also with tmap
library(tmap)

# Plot
```


2015
2016
2017
36.2°N
36°N
35.8°N
35.6°N
35.4°N
79.2°W
79°W
78.8°W
78.6°W
78.4°W
78.2°W
LST_summer
31
30
29

Plotting GRASS GIS maps in R

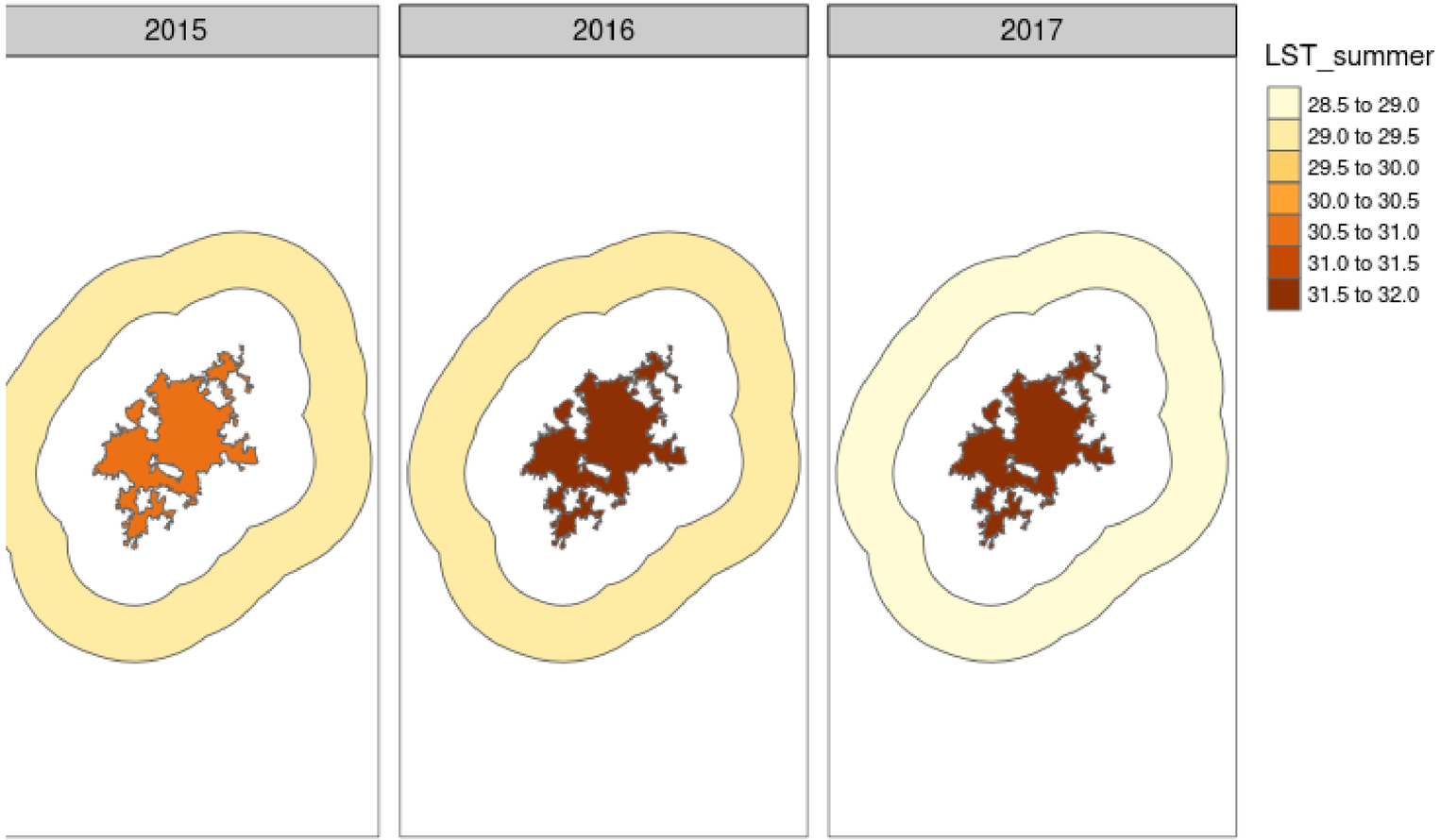
```
#####################################################################
# Commands for the FOSS4G workshop in Bucharest
# Author: Veronica Andreo
# Date: July, 2019
#####################################################################

# Load rgrass library
library(rgrass7)
library(sf)

# List available vectors
execGRASS("g.list", parameters = list(type="vector", mapset="."))

# Read in GRASS vector maps as sf
use_sf()
raleigh_summer_lst <- readVECT("raleigh_summer_lst")
raleigh_surr_summer_lst <- readVECT("raleigh_surr_summer_lst")

# Remove columns we don't need
raleigh_summer_lst <- raleigh_summer_lst[,-c(2:6)]
raleigh_surr_summer_lst <- raleigh_surr_summer_lst[,-c(2:3)]
```

Plotting GRASS GIS maps in R

```
# Plot
ggplot() +
  geom_sf(data = raleigh2, aes(fill = LST_summer)) +
  facet_wrap(~YEAR, ncol = 3) +
  scale_fill_distiller(palette = "YlOrRd",
                       direction = 1) +
  scale_y_continuous()


# Let's try also with tmap
library(tmap)

# Plot

tm_shape(raleigh2) +
  tm_polygons(col = "LST_summer") +
  tm_facets(by = "YEAR", nrow = 1, free.coords = FALSE)



# mapview for quick visualizations with basemaps is really cool!
```

Plotting GRASS GIS maps in R

```
# Plot
ggplot() +
  geom_sf(data = raleigh2, aes(fill = LST_summer)) +
  facet_wrap(~YEAR, ncol = 3) +
  scale_fill_distiller(palette = "YlOrRd",
                       direction = 1) +
  scale_y_continuous()



# Let's try also with tmap
library(tmap)

# Plot

tm_shape(raleigh2) +
  tm_polygons(col = "LST_summer") +
  tm_facets(by = "YEAR", nrow = 1, free.coords = FALSE)



# mapview for quick visualizations with basemaps is really cool!
```

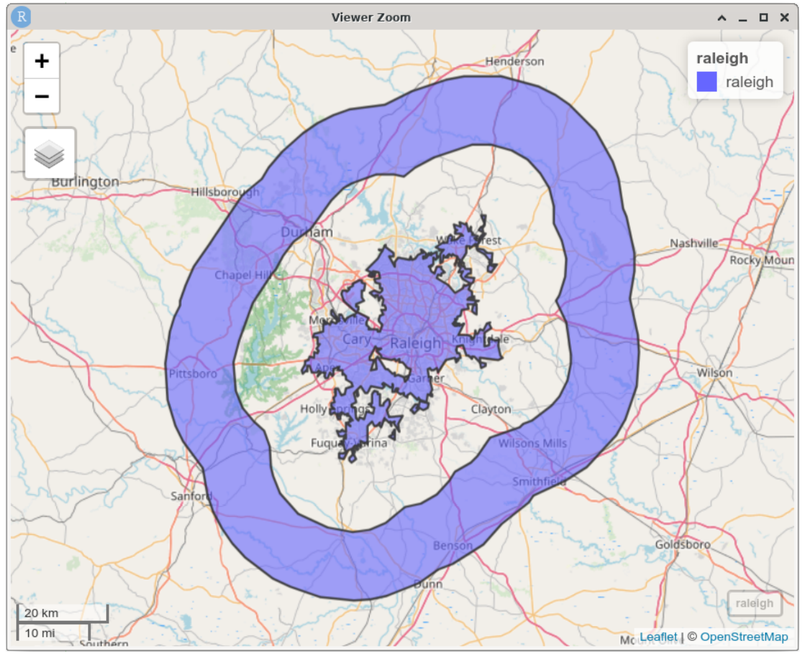

2015
2016
2017
LST_summer
28.5 to 29.0
29.0 to 29.5
29.5 to 30.0
30.0 to 30.5
30.5 to 31.0
31.0 to 31.5
31.5 to 32.0

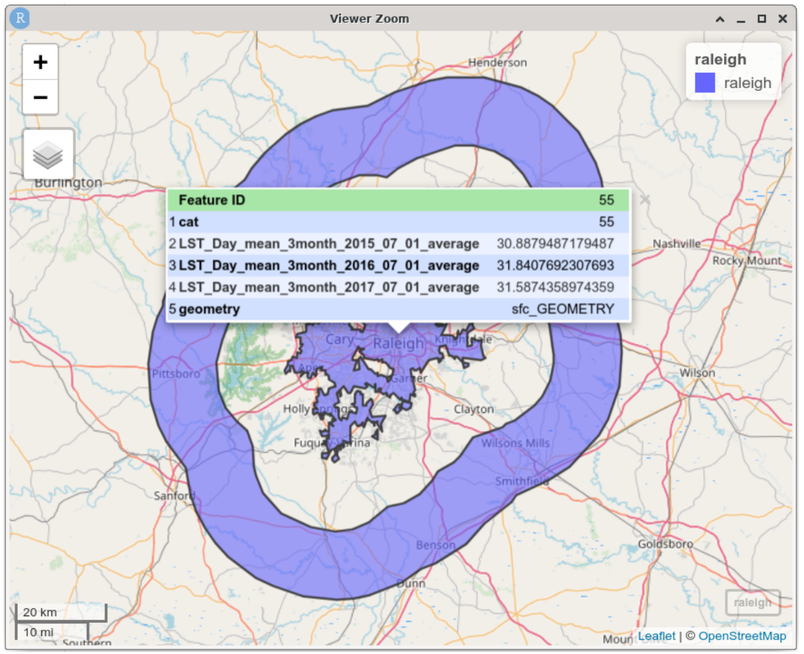
Plotting GRASS GIS maps in R

```
#############################################################################
# Commands for the FOSS4G workshop in Bucharest
# Author: Veronica Andreo
# Date: July, 2019
#############################################################################

# Load rgrass library
library(rgrass7)
library(sf)

# List available vectors
execGRASS("g.list", parameters = list(type="vector", mapset="."))

# Read in GRASS vector maps as sf
use_sf()
raleigh_summer_lst <- readVECT("raleigh_summer_lst")
raleigh_surr_summer_lst <- readVECT("raleigh_surr_summer_lst")

# Remove columns we don't need
raleigh_summer_lst <- raleigh_summer_lst[,-c(2:6)]
raleigh_surr_summer_lst <- raleigh_surr_summer_lst[,-c(2:3)]
```

Plotting GRASS GIS maps in R

```
# Plot
ggplot() +
  geom_sf(data = raleigh2, aes(fill = LST_summer)) +
  facet_wrap(~YEAR, ncol = 3) +
  scale_fill_distiller(palette = "YlOrRd",
                       direction = 1) +
  scale_y_continuous()


# Let's try also with tmap
library(tmap)

# Plot

tm_shape(raleigh2) +
  tm_polygons(col = "LST_summer") +
  tm_facets(by = "YEAR", nrow = 1, free.coords = FALSE)


# mapview for quick visualizations with basemaps is really cool!
library(mapview)
```


Viewer Zoom
raleigh
raleigh
Henderson
Burlington
Hillsborough
Durham
Chapel Hill
Cary
Raleigh
Pittsboro
Holly Springs
Fuquay-Varina
Clayton
Wilsons Mills
Smithfield
Sanford
Benson
Goldsboro
Wilson
Nashville
Rocky Mount
Dunn
20 km
10 mi
Leaflet | © OpenStreetMap


Viewer Zoom
raleigh
raleigh
Feature ID 55
1 cat 55
2 LST_Day_mean_3month_2015_07_01_average 30.8879487179487
3 LST_Day_mean_3month_2016_07_01_average 31.8407692307693
4 LST_Day_mean_3month_2017_07_01_average 31.5874358974359
5 geometry sfc_GEOMETRY
Henderson
Burlington
Cary
Raleigh
Pittsboro
Clayton
Wilsons Mills
Smithfield
Sanford
Benson
Goldsboro
Wilson
Nashville
Rocky Mount
Dunn
20 km
10 mi
Leaflet | © OpenStreetMap

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


Join and enjoy GRASS GIS!!

Thanks for your attention!!

Verónica Andreo

veroandreo

@VeronicaAndreo

veroandreo@gmail.com

Presentation powered by

GitPitch