WS27: Unleash the power of GRASS GIS 7

Session 6 – Temporal data analysis (TGRASS)

Markus Neteler – mundialis GmbH & Co KG, Germany
Luca Delucchi – Fondazione Edmund Mach, Italy
Martin Landa – Czech Technical University, Prague

FOSS4G 2016, Bonn
http://foss4g2016.org/ws27.html

Session Objectives

- Temporal GRASS framework
- Working with climate data
  - Starting with temporal framework
  - Aggregating data in new temporal dataset
  - Querying the temporal dataset
  - Visualizing temporal dataset

New Space-Time functionality in GRASS 7

Developer: Sören Gebbert

Temporal data processing in GRASS GIS

The temporal GIS framework in GRASS introduces three new datatypes that are designed to handle time series data:

- *Space time raster datasets* (strds) are designed to manage raster map time series. Modules that process strds have the naming prefix *t.rast*.
- *Space time 3D raster datasets* (str3ds) are designed to manage 3D raster map time series. Modules that process str3ds have the naming prefix *t.rast3d*.
- *Space time vector datasets* (stvds) are designed to manage vector map time series. Modules that process stvds have the naming prefix *t.vect*.

Temporal data management in general

List of general management modules:

- t.connect
- t.create
- t.remove
- t.register
- t.unregister
- t.info
- t.list
- t.rast3d.list
- t.vect.list
- t.vect.db.select
- t.sample
- t.support
- t.topology

Export/import conversion

- t.rast.export
- t.rast.import
- t.rast.out.vtk
- t.rast.to.rast3
- r3.out.netcdf
- t.vect.export

Statistics and gap filling

- t.rast.gapfill
- t.rast.univar

Querying and map calculation

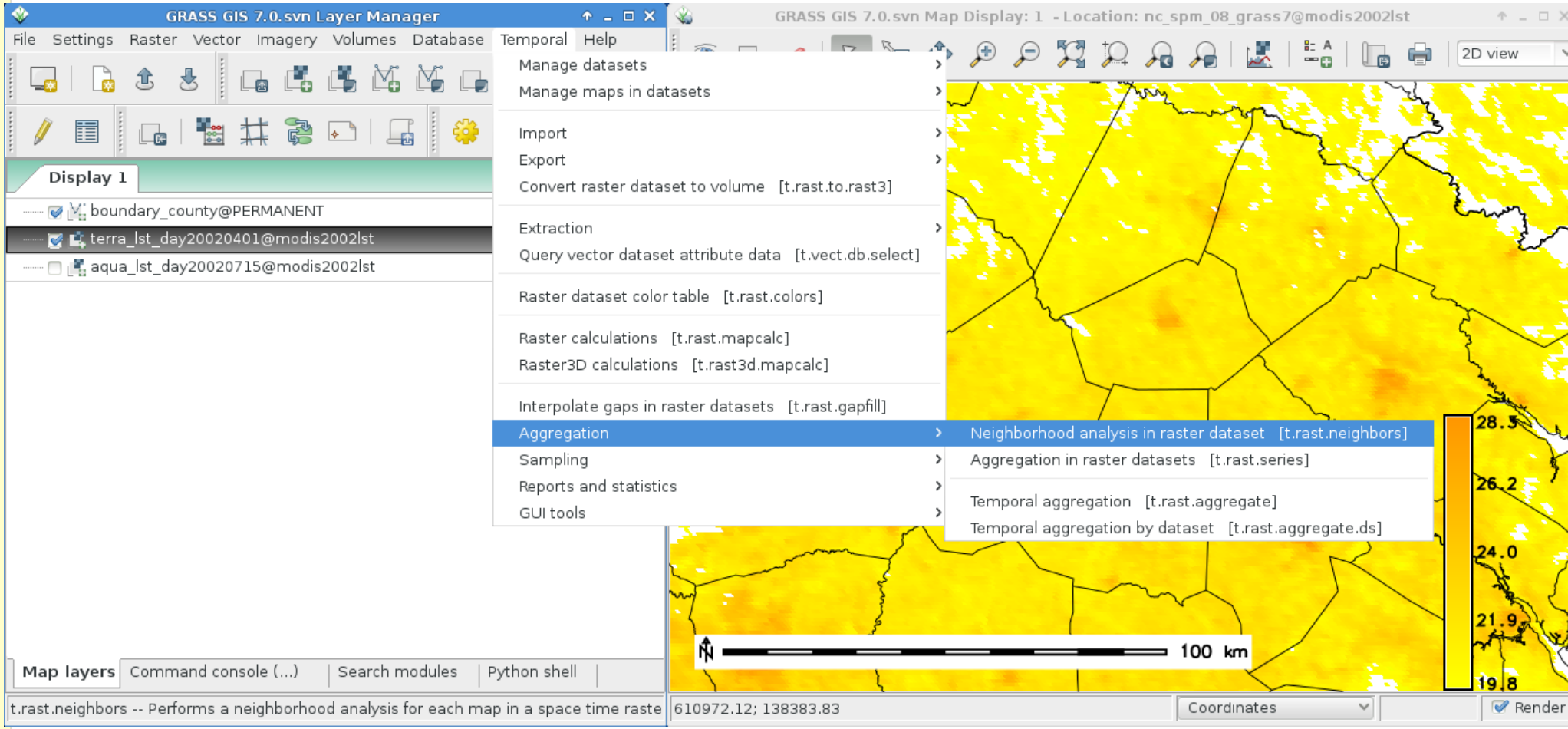
- t.rast.list
- t.rast.extract
- t.rast.gapfill
- t.rast.mapcalc
- t.rast3d.extract
- t.rast3d.mapcalc
- t.rast3d.univar
- t.vect.extract
- t.vect.import
- t.vect.observe.strds
- t.vect.univar
- t.vect.what.strds

Aggregation

- t.rast.aggregate.ds
- t.rast.aggregate
- t.rast.series

Space time datasets are stored in a temporal database. SQLite3 or PostgreSQL are supported as SQL database back end. Connection settings are performed with t.connect. As default a sqlite3 database will be created in the PERMANENT mapset that stores all space time datasets and registered time series maps from all mapsets in the location.

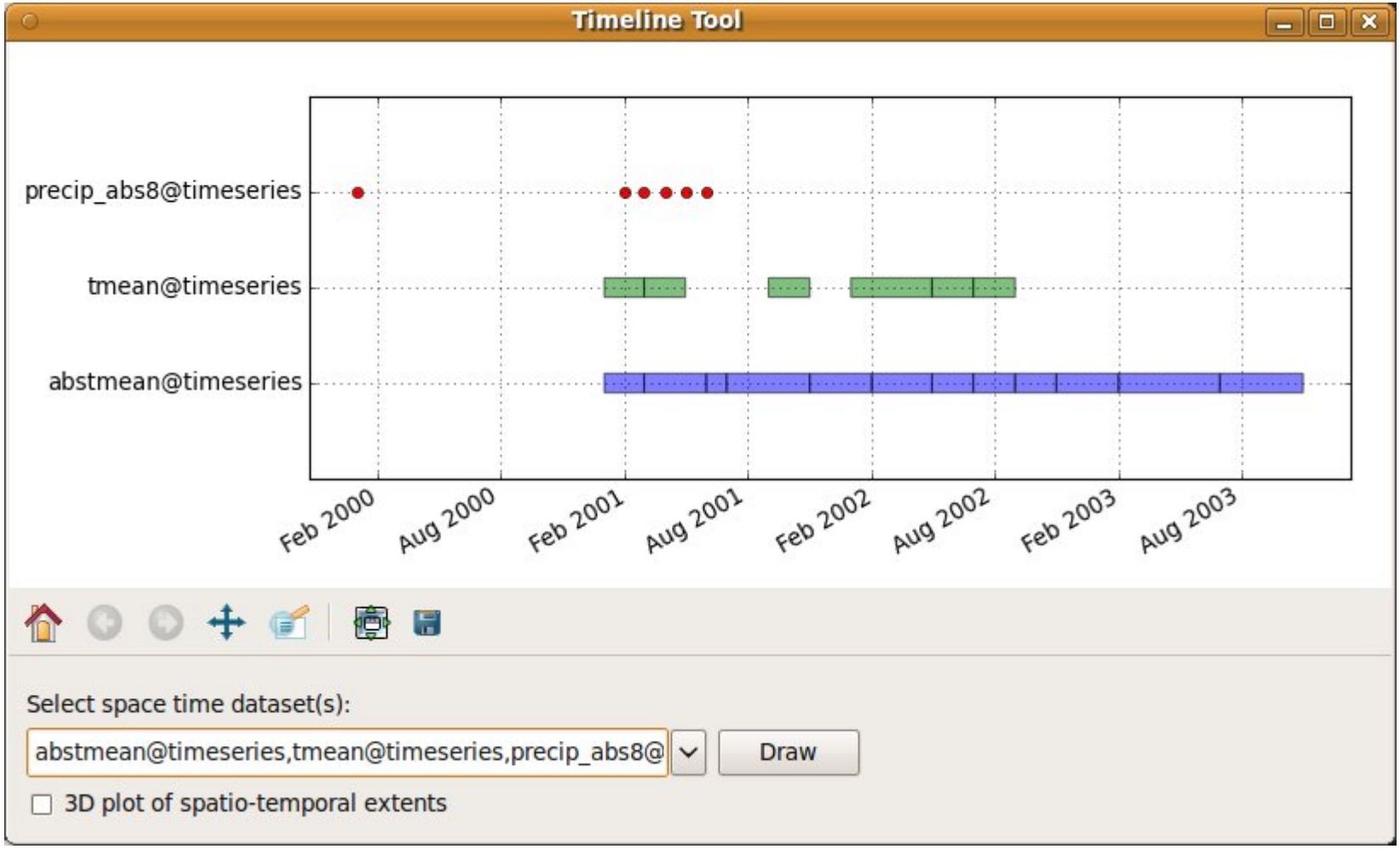
New Space-Time functionality in GRASS 7



Temporal GRASS (TGRASS) menus

Gebbert, S., Pebesma, E., 2014. *TGRASS: A temporal GIS for field based environmental modeling*. Environmental Modelling & Software 53, 1-12. (DOI)

New Space-Time functionality in GRASS 7

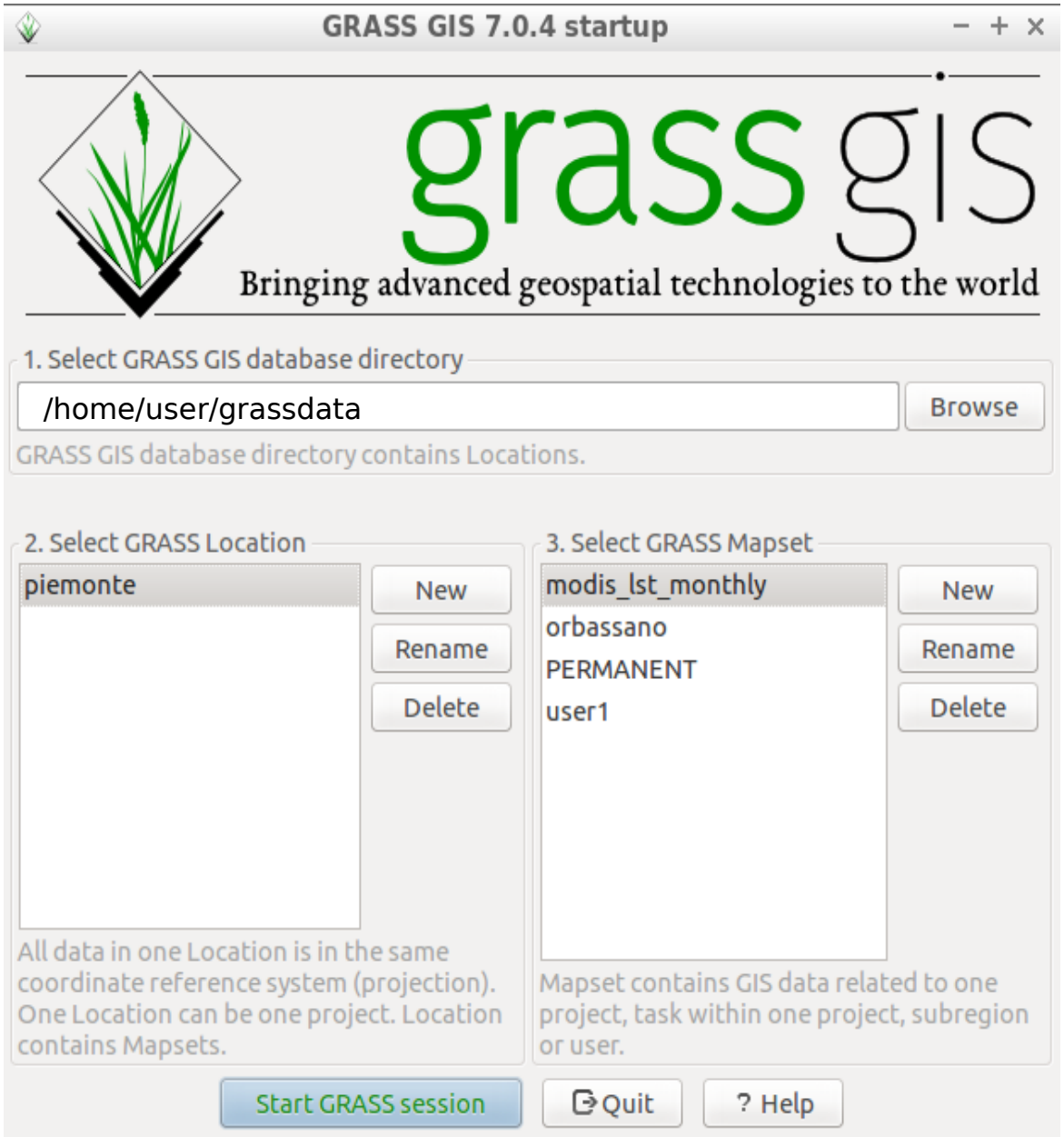


Screenshot: S Gebbert/A. Petrasova

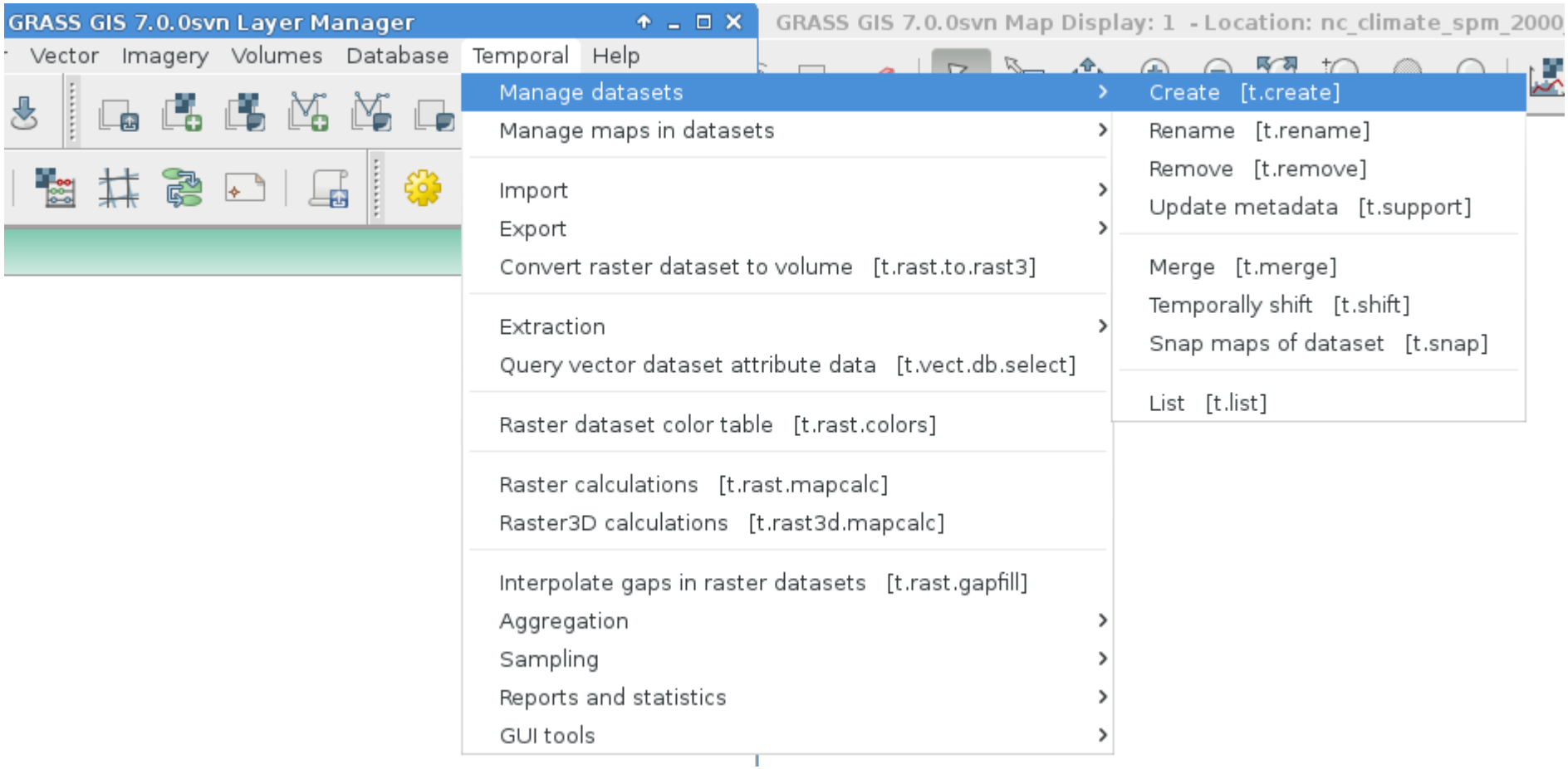
g.gui.timeline: show raster, vector and raster3D maps in a space time dataset

Gebbert, S., Pebesma, E., 2014. *TGRASS: A temporal GIS for field based environmental modeling*. Environmental Modelling & Software 53, 1-12. (DOI)

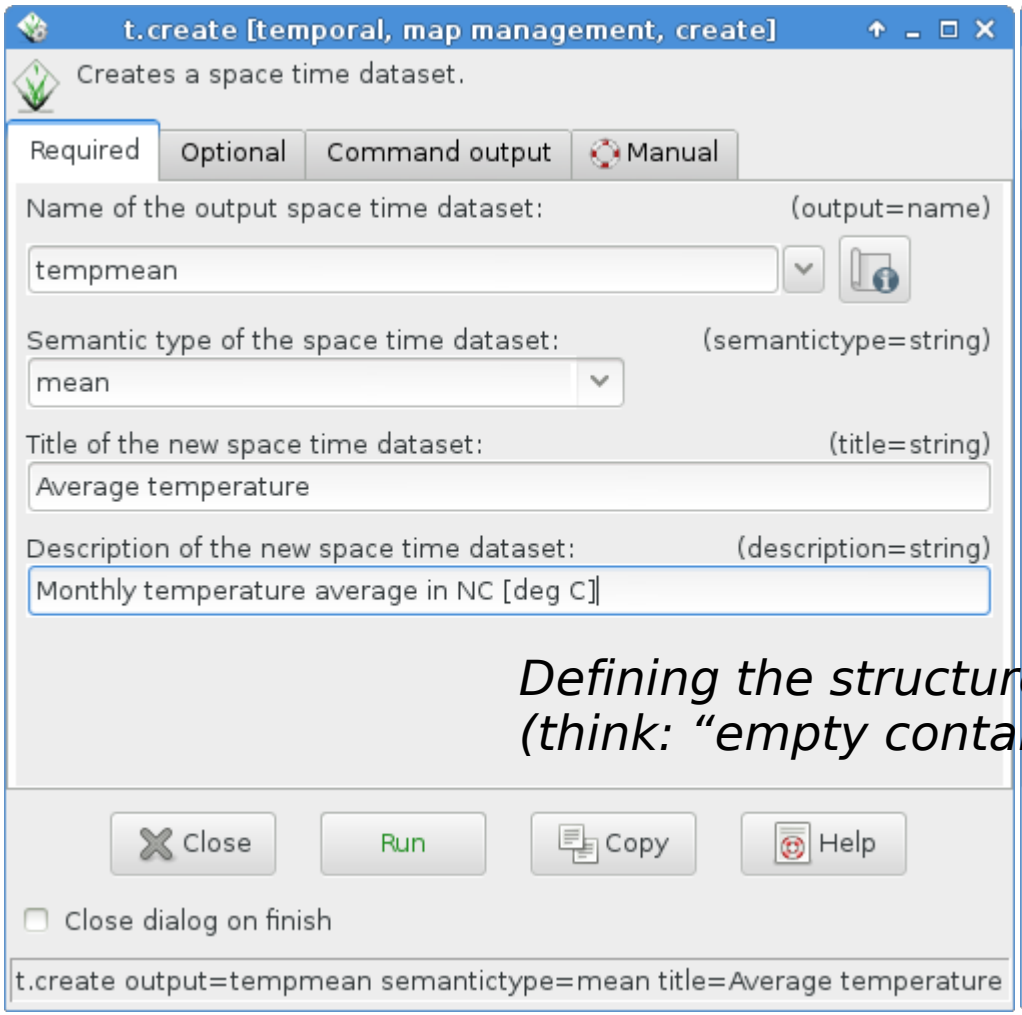
Exercise – GRASS GIS 7 with Piemonte climate data

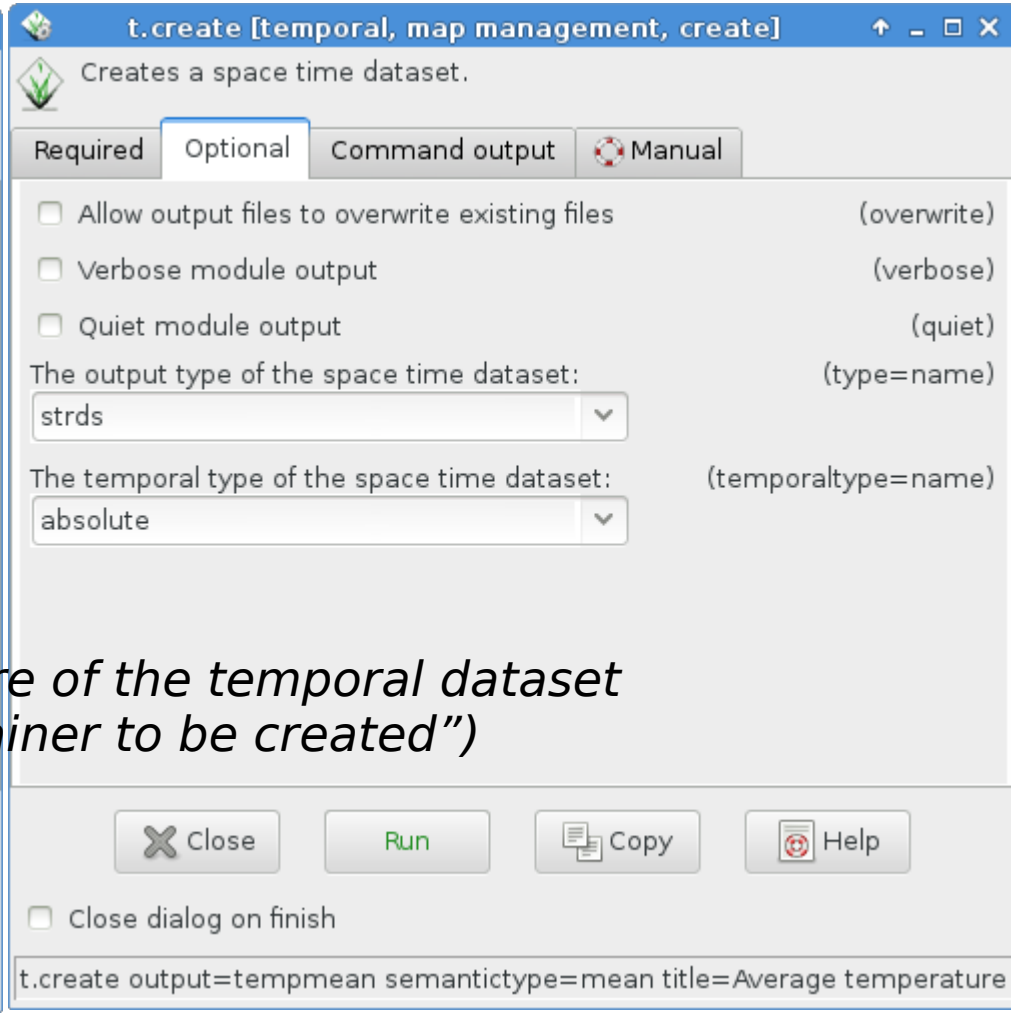


Exercise – Temporal support in GRASS GIS 7



Exercise – Temporal data creation: Temp+Precip





Defining the structure of the temporal dataset
(think: "empty container to be created")

```
t.create output=tempmean type=strds semantictype=mean temporaltype=absolute \
  title="Average temperature" description="Monthly temp. avg in Piemonte [deg C]"
```

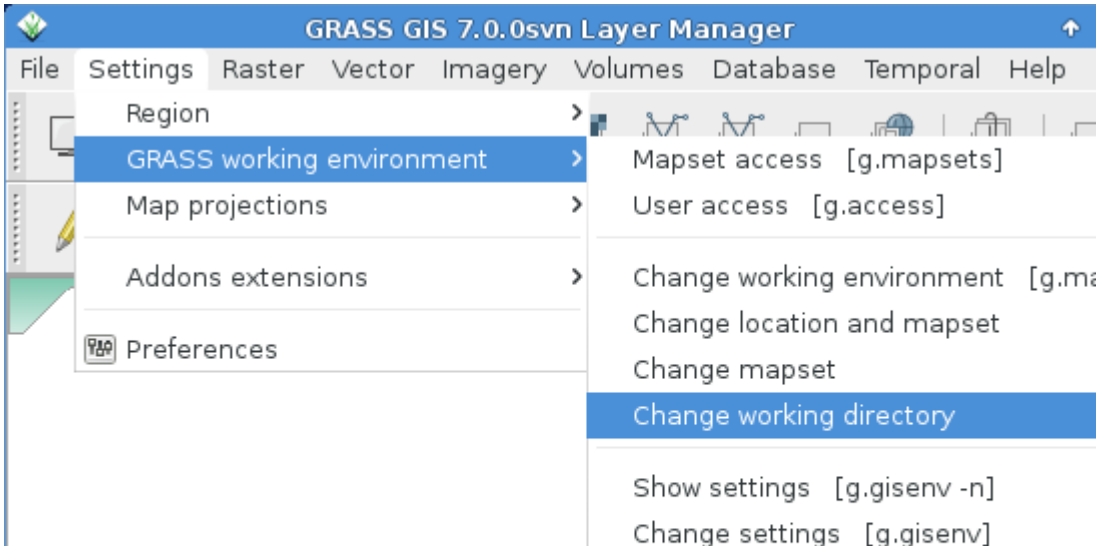
Exercise – Temporal data registration: Temp

Registering the existing raster maps into yet empty STRDS datasets:

1) Temperature data

```
# list TEMP raster maps with g.list
g.list type=raster mapset=. pattern=”lst*average”

# … if ok, then save to file for t.register
g.list type=raster mapset=. pattern=”lst*average” output=maps_tempmean.txt

t.register -i input=tempmean type=rast start=2004-01-01 increment="1 months" \
        file=maps_tempmean.txt

# get some info
t.rast.list input=tempmean

# more possibilities how to list maps
t.rast.list input=tempmean \
  columns=name,start_time,end_time \
  method=gran granule="1 years"
```

If g.list cannot write into “output”:



Exercise – Time series analysis

```
# univariate statistics (use where to limit output)
t.rast.univar tempmean where="start_time > '2010-01-01' " -h

# aggregate time series by years
t.rast.aggregate -s input=tempmean output=tempmean_year \
  base=lst_year granularity="1 year" method=average

# get information about the aggregate:
t.info tempmean_year
```

Exercise – Time series analysis

```
# extract June months and convert to degree Celsius from originally scaled °C * 100

# strftime('%m', start_time) returns the month of the map start timestamp
# note: it is not a GRASS GIS function but SQL function (here: SQLite)
# ... and it use multiple cores

t.rast.extract input=tempmean where="strftime('%m', start_time)='06'" \
  expression="tempmean  / 100.0" \
  output=tempmean_june base=tempmean_june nprocs=4
```

Exercise – Time series analysis

```
# get temperatures for the city of Torino, Italy
# either add UTM32N coordinates line to a file point.txt
# or add it 'interactively' in the v.in.ascii dialog
echo "395629,4991881" > point.txt
v.in.ascii -t input=point.txt output=torino separator=comma

# create a vector map and space-time vector dataset with values of temperature of Torino
t.vect.observe.strds input=torino strds=tempmean \
  output=torino_tempmean \
  vector_output=torino_tempmean_stvds column=tempmean

# list temperature values
t.vect.db.select input=torino_tempmean columns=tempmean \
  separator=, where="cat = 1"
```

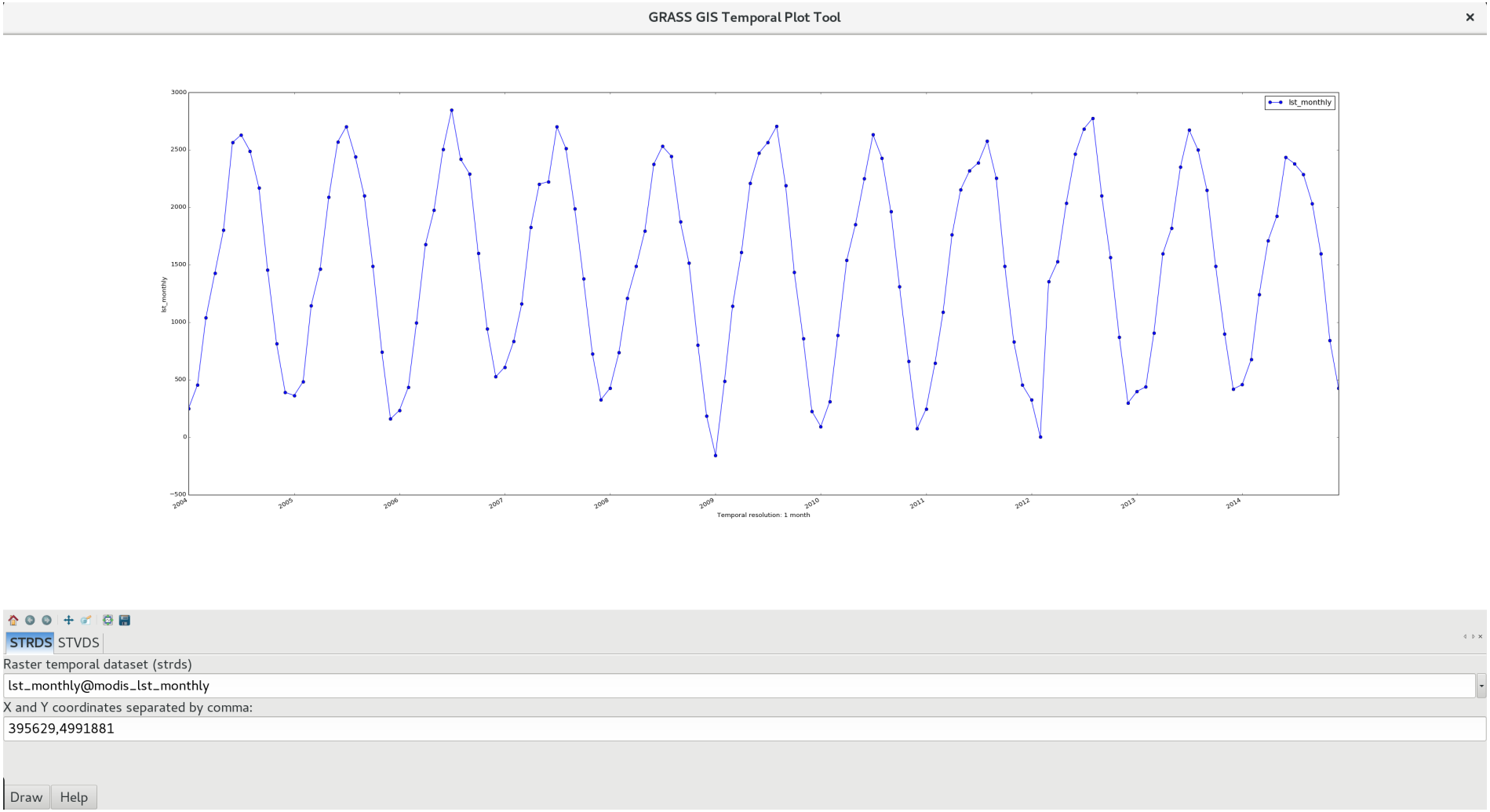
Exercise – Time series analysis

```
# create a vector map with values of summer temperature in Torino
v.what.stvds input=torino strds=tempmean_year output=torino_year

# list temperature values
db.select input=torino_years
```

Exercise – Time series analysis

show plot temporal datasets related to a point

g.gui.tplot strds=tempmean coordinates=395629,4991881



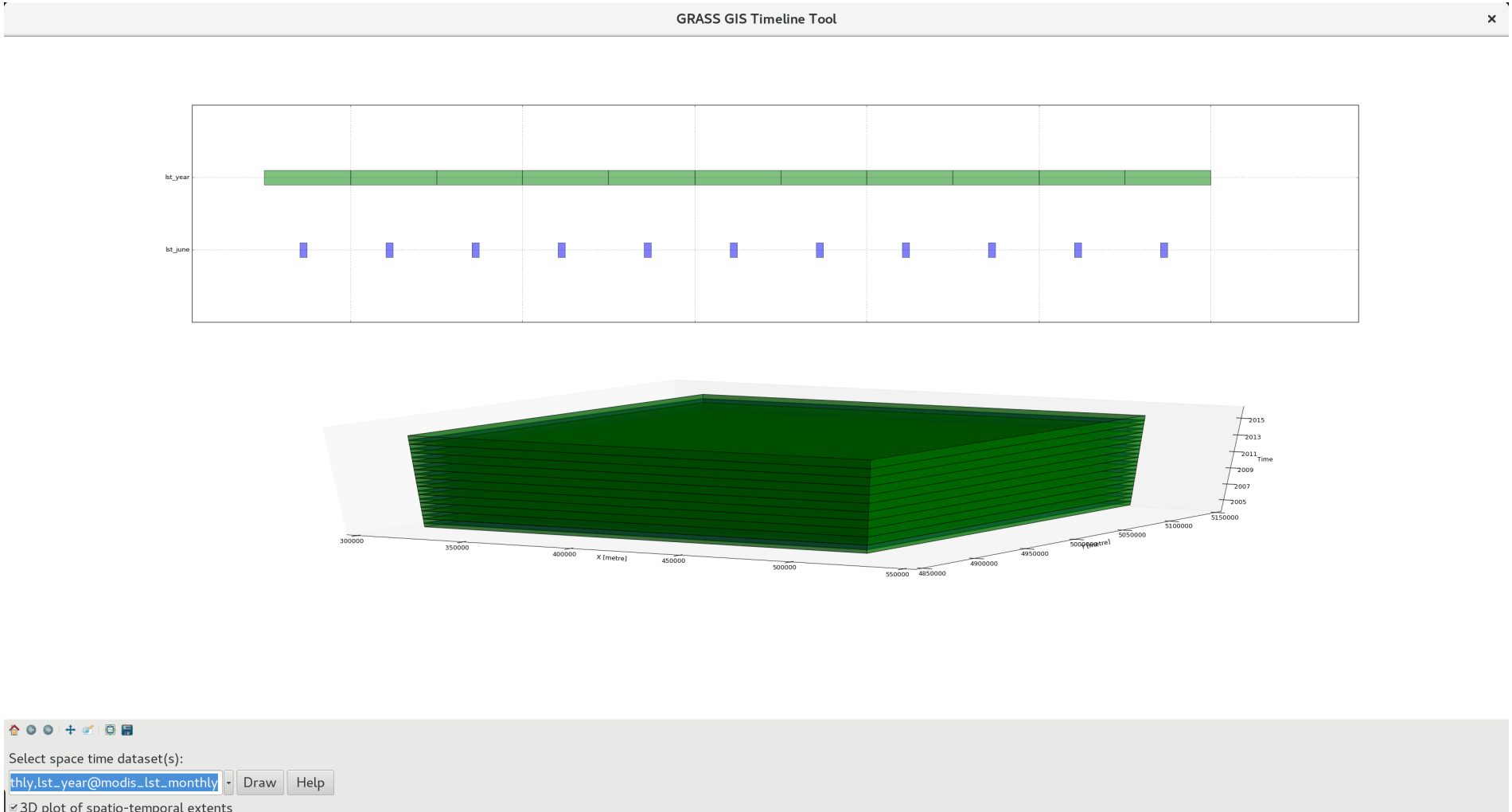
Exercise – Time series analysis

```
# show plot of min and max values
# first run t.rast.list and copy output to a file (you should set working
# directory from menu beforehand)
t.rast.list -h input=tempmean_year columns=start_time,min,max separator=comma

t.rast.list -h input=tempmean_year columns=start_time,min,max \
  separator=comma > output.txt


# now go to Python shell tab in the wxGUI and type
import matplotlib.pyplot as plt

plt.plotfile("output.txt", cols=(0,1,2), delimiter=',', subplots=False)

plt.show()
```

Exercise – Timeline of the time series

the 2D and 3D plot are zoomable (lense and right mouse button, respectively)

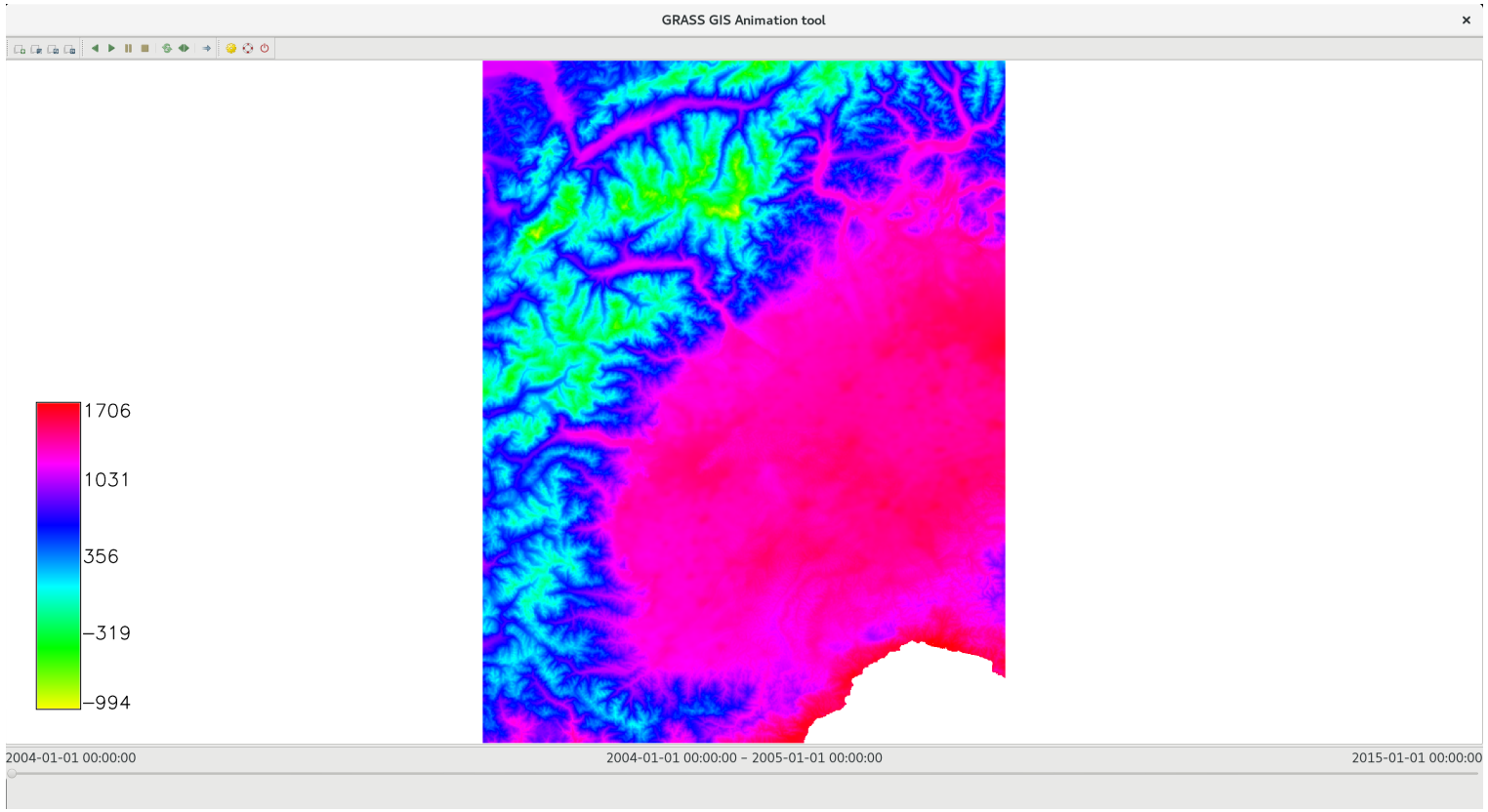
g.gui.timeline -3 inputs=lst_year,lst_june



Exercise – Visualizing the time series: Animation

display animation of temperatures
display legend, set min, max values in legend from t.info

t.info tempmean_year
g.gui.animation strds=tempmean_year



Exercise – Visualizing the time series: Animation

export animation to GIF file