

GRASS GIS 7: A theoretical and practical course

Session 1 – Temporal data analysis (TGRASS)

Markus Neteler

NINA 2016, Oslo, Norway

mundialis GmbH & Co. KG
<http://www.mundialis.de>





Session Objectives

- Temporal GRASS framework
- Registration of maps
- Time series analysis



New Space-Time functionality in GRASS 7

Temporal data processing in GRASS GIS

Developer: Sören Gebbert

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](#)

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](#)

Statistics and gap filling

- [t.rast.gapfill](#)
- [t.rast.univar](#)

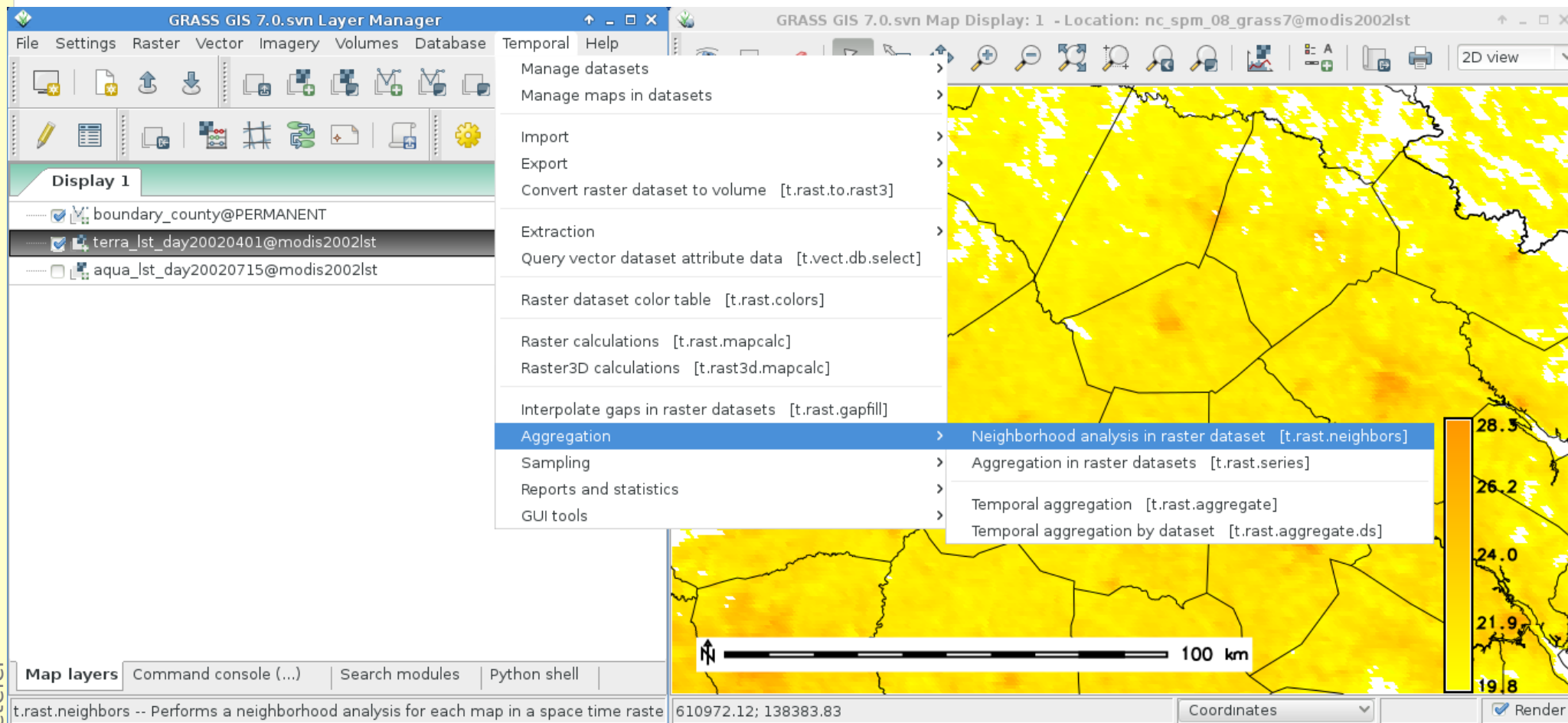
Aggregation

- [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 backend. 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.

<https://grass.osgeo.org/grass72/manuals/temporalintro.html>

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

Example: daily MODIS Land Surface Temperature time series

The screenshot shows the GRASS GIS interface with the **t.register** tool and the **Timeline Tool** open.

t.register [temporal, map management, register]
Registers raster, vector and raster3d maps in a space time datasets.

Input
Name of the input space time dataset: `modis_lst2002@modis2002lst`

Optional
[multiple] Name of the input maps:

Command output

Manual
Type of the input map: `rast`

Input file with map names, one per line. Additionally the start time and the
`/home/neteler/tgrass.aqua_lst_day_list.csv`

or enter values interactively

Field separator:

Buttons: Close, Run, Copy, Help

Timeline Tool

Timeline plot showing the time series of `modis_lst2002` from July 2002 to December 2002.

3D plot of spatio-temporal extents showing the spatial distribution of the time series data.

Select space time dataset(s): `modis_lst2002@modis2002lst`

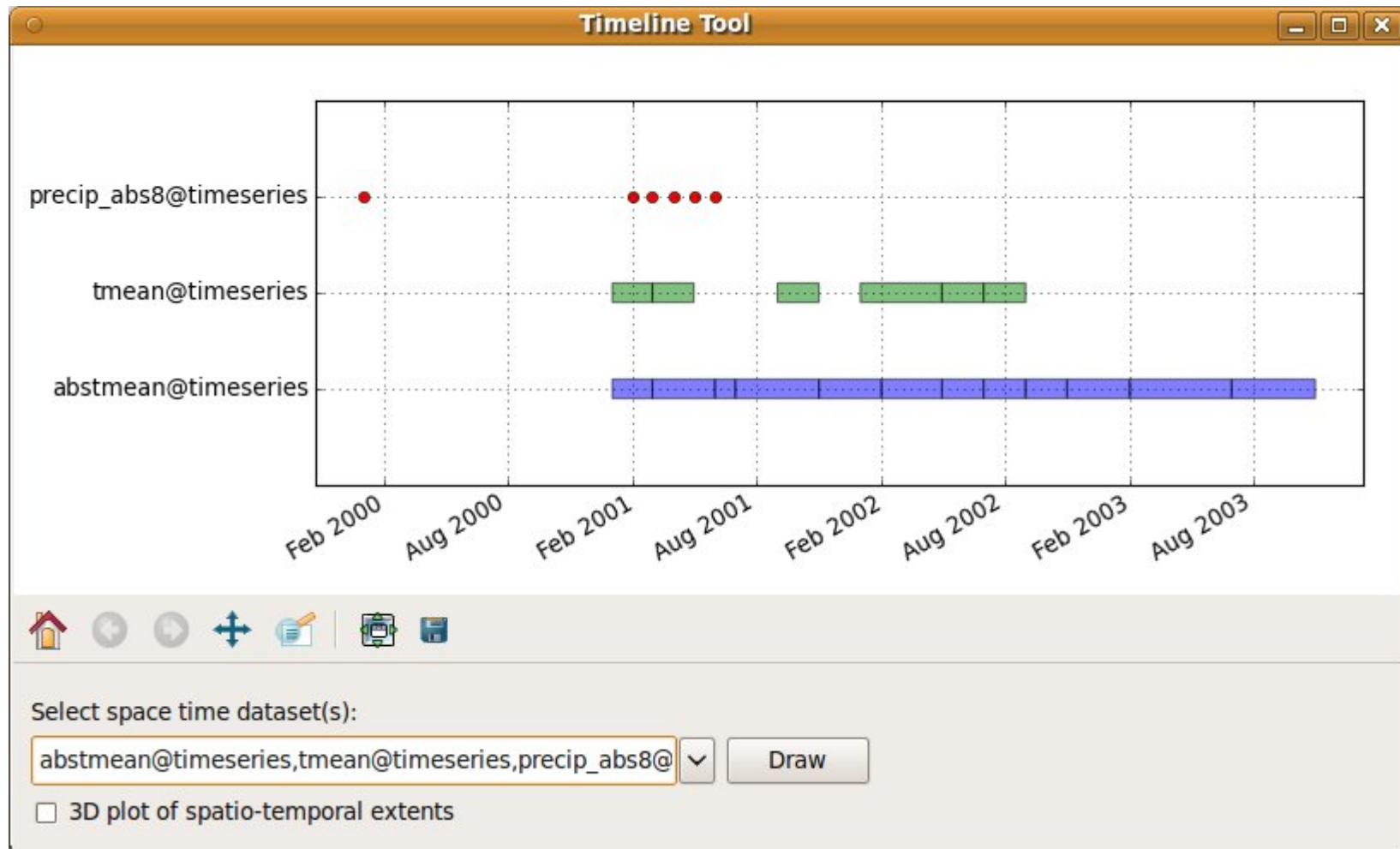
Buttons: Draw, Help

☒ 3D plot of spatio-temporal extents

t.register input=modis_lst2002@modis2002lst file=/home/neteler/tgrass.aqua_lst_day_list.csv

t.register: Registers raster, vector and raster3d maps in a space time dataset

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](#))



Data: time series

Course data download address:

http://courses.ncsu.edu/mea592/common/media/02/nc_climate_spm_2000_2012.zip

Get the package and save it on your computer.

Unpack the GRASS GIS location into your HOME/grassdata/

Content for exercises:

- Aggregated gridded data for NC (PRISM)
- Elevation model at 500m

Exercise – GRASS GIS 7 startup with NC climate data



GRASS GIS 7.2.svn startup (r69482M)

 **GRASS GIS**
Bringing advanced geospatial technologies to the world

1. Select GRASS GIS database directory

GRASS GIS database directory contains Locations.

2. Select GRASS Location

demolocation	New
ecad5_grassdata_ll	Rename
grassdata_piemonte	Delete
nc_climate_spm_2000_2012	
nc_spm_08_grass7	
spearfish70	

All data in one Location is in the same coordinate reference system (projection). One Location can be one project. Location contains Mapsets.

3. Select GRASS Mapset

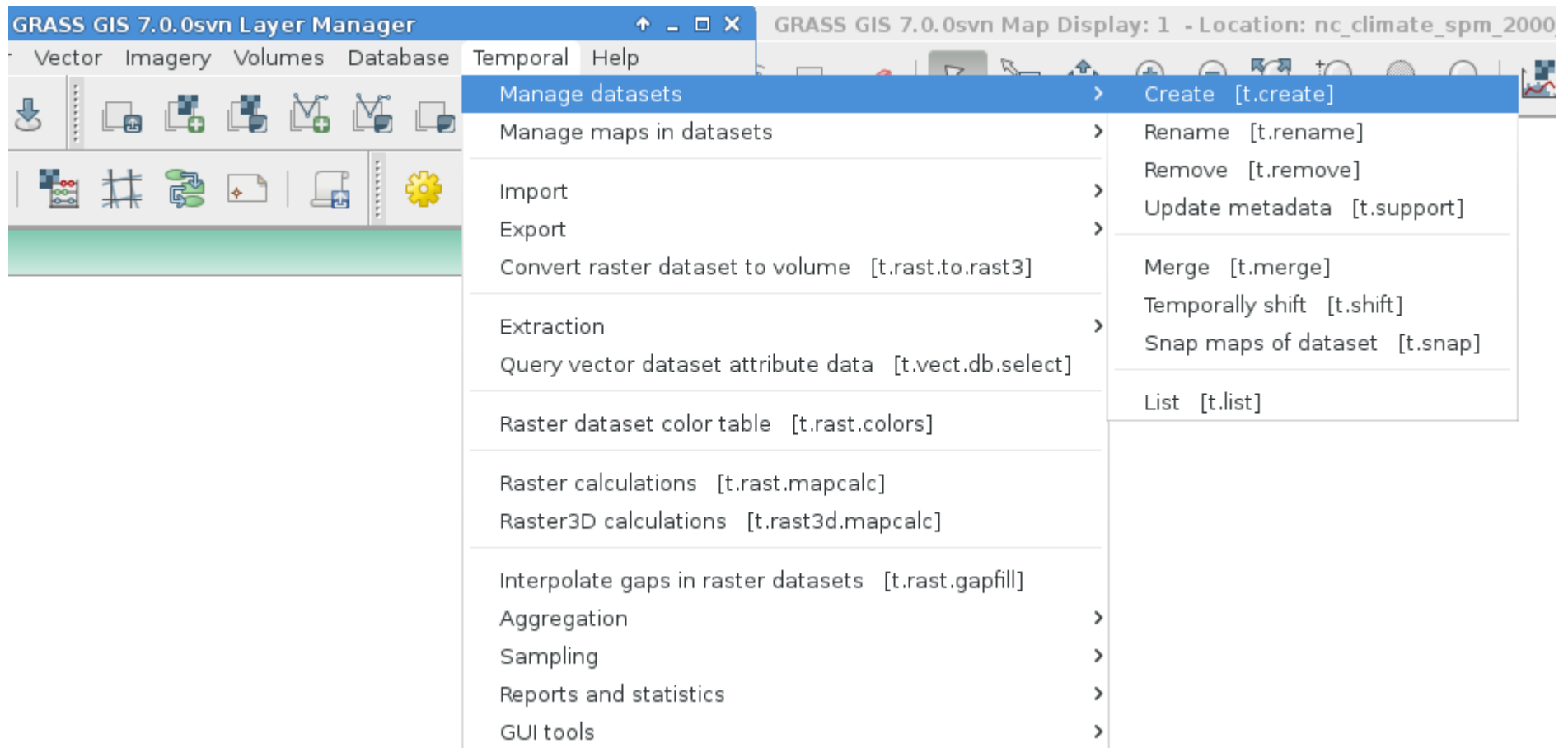
climate_1970_2012	New
PERMANENT	Rename
	Delete

Mapset contains GIS data related to one project, task within one project, subregion or user.

Exercise based on [material](#) developed by Anna Petrášová, NCSU (group led by Dr. Helena Mitasova)



Exercise – Temporal support in GRASS GIS 7



Exercise – Temporal data creation: Temp+Precip



t.create [temporal, map management, create] ↑ _ □ ×

Creates a space time dataset.

Required Optional Command output Manual

Name of the output space time dataset: (output=name)
tempmean

Semantic type of the space time dataset: (semantictype=string)
mean

Title of the new space time dataset: (title=string)
Average temperature

Description of the new space time dataset: (description=string)
Monthly temperature average in NC [deg C]

Close Run Copy Help

☐ Close dialog on finish

t.create output=tempmean semantictype=mean title=Average temperature

Defining the structure of the temporal dataset (think: “empty container”)

t.create [temporal, map management, create] ↑ _ □ ×

Creates a space time dataset.

Required Optional Command output Manual

☐ Allow output files to overwrite existing files (overwrite)
☐ Verbose module output (verbose)
☐ Quiet module output (quiet)

The output type of the space time dataset: (type=name)
strds

The temporal type of the space time dataset: (temporaltype=name)
absolute

Close Run Copy Help

☐ Close dialog on finish

t.create output=tempmean semantictype=mean title=Average temperature

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

```
t.create output=precipsum type=strds semantictype=mean temporaltype=absolute \  
title="Precipitation" description="Monthly precip. sum in NC [mm]"
```



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`

(note: maps must be in search path, manage with `g.mapsets -s`)

```
g.list type=raster pattern="*tempmean"
```

```
# ... if ok, then save to file for t.register
```

```
g.list type=raster pattern="*tempmean" output=maps_tempmean.txt
```

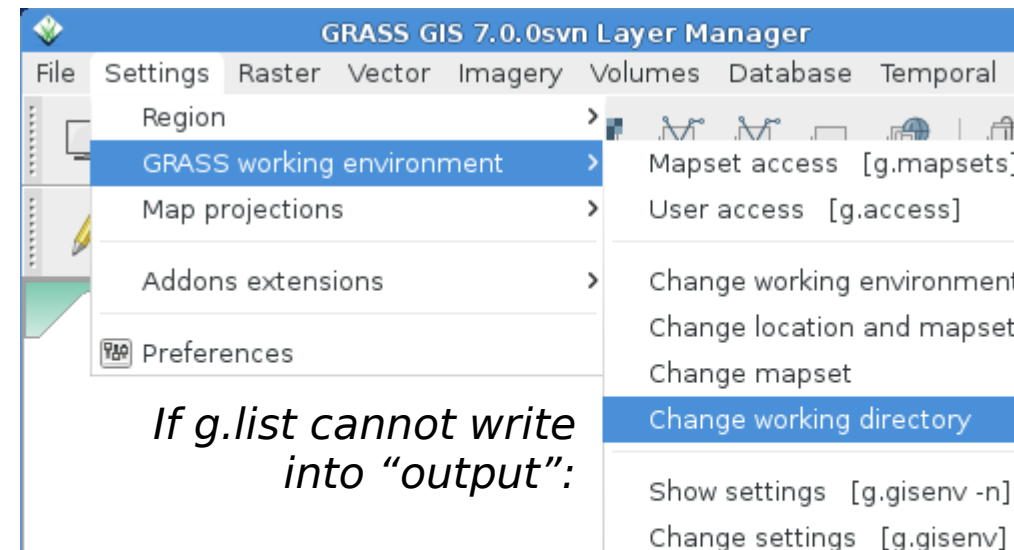
```
t.register -i input=tempmean type=rast start=2000-01-01 increment="1 months" \
          file=maps_tempmean.txt
```

```
# 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 – Temporal data registration: Precip

Registering the existing raster maps into yet empty STRDS datasets:

2) Precipitation data

List PRECIP raster maps with g.list

```
g.list type=raster pattern="*precip*"

# ... if ok, then save to file for t.register
g.list type=raster pattern="*precip*" output=maps_precipsum.txt
t.register -i input=precipsum type=rast start=2000-01-01 increment="1 months" \
          file=maps_precipsum.txt

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

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



Exercise – Time series analysis

Univariate statistics

... can be combined with “where” conditions

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

# aggregate time series by seasons
t.rast.aggregate input=tempmean output=tempmean_seasonal \
  base=tempmean_seasonal \
  granularity="3 months" method=average \
  where="start_time >= '2000-03-01' and start_time < '2012-11-01'"

# get information about the new aggregate:
t.info tempmean_seasonal
```



Exercise – Time series analysis

Extract summer periods and convert to degrees Fahrenheit

- we do extraction with conversion on the fly
- the `strftime()` function is also used in R and SQL/SQLite, used for timestamp parsing

```
# strftime('%m', start_time) returns the month of the map start timestamp  
# It is not a GRASS GIS function but SQL (here: SQLite)  
# ... our job uses multiple cores, here: 4
```

```
t.rast.extract input=tempmean_seasonal \  
  where="strftime('%m', start_time)='06'" \  
  expression="(tempmean_seasonal * 9.0/5.0) + 32" \  
  output=tempmean_F_summer base=tempmean_F_summer nprocs=4
```




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_F_summer columns=start_time,min,max
separator=comma

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

# now go to Python shell tab in the wxGUI and enter:
import matplotlib.pyplot as plt
plt.plotfile("output.txt", cols=(0,1,2), delimiter=',', subplots=False)
plt.show()
```



Exercise – Time series analysis

A similar exercise with precipitation dataset using a different conversion

```
# aggregate data using time intervals of tempmean_F_summer
# convert millimeters to inch
t.rast.aggregate.ds input=precipsum sample=tempmean_F_summer \
  output=precip_summer \
  base=precip_summer method=average

t.rast.mapcalc inputs=precip_summer expression="precip_summer / 25.4" \
  output=precip_inch_summer base=precip_inch_summer nprocs=4
```



Exercise – Time series analysis

Are precipitation and temperature correlated?

```
# download extension r.regression.series
g.extension extension=r.regression.series

# run g.list and save output into a file
g.list type=raster pattern="tempmean_F_summer*" separator=comma
g.list type=raster pattern="tempmean_F_summer*" separator=comma \
  output=tempmean_F_summer.txt

g.list type=raster pattern="precip_inch_summer*" separator=comma \
  output=precip_IN_summer.txt

# commands between backticks `...` are executed first (here: print list):
r.regression.series xseries=`cat tempmean_F_summer.txt` \
  yseries=`cat precip_IN_summer.txt` output=corr method=corcoef

r.colors map=corr color=differences
```



Exercise – Time series analysis

Display temperatures in Raleigh, NC

```
# add this line to a file point.txt or add it 'interactively' in the  
v.in.ascii dialog
```

```
echo "638863,225446" > point.txt
```

```
v.in.ascii -t input=point.txt output=raleigh separator=comma
```

```
# create a vector map and space-time vector dataset with values of summer  
temperature in Raleigh
```

```
t.vect.observe.strds input=raleigh strds=tempmean_F_summer \  
  output=raleigh_tempmean_summer \  
  vector_output=raleigh_summer column=tempmean
```

```
# list temperature values
```

```
t.vect.db.select input=raleigh_tempmean_summer columns=tempmean \  
  separator=comma where="cat = 1"
```



Exercise – Time series analysis: plotting

Plot the values using matplotlib commands as before

Extract one year and animate it (a few slides later)

```
t.rast.extract input=tempmean output=tempmean_2012 \  
  base=tempmean_2012 where="start_time >= '2012-01-01'"
```

```
t.rast.extract input=precipsum output=precip_2012 \  
  base=precip_2012 where="start_time >= '2012-01-01'"
```

```
# change color table if desired (to precipitation_monthly or create your own)
```

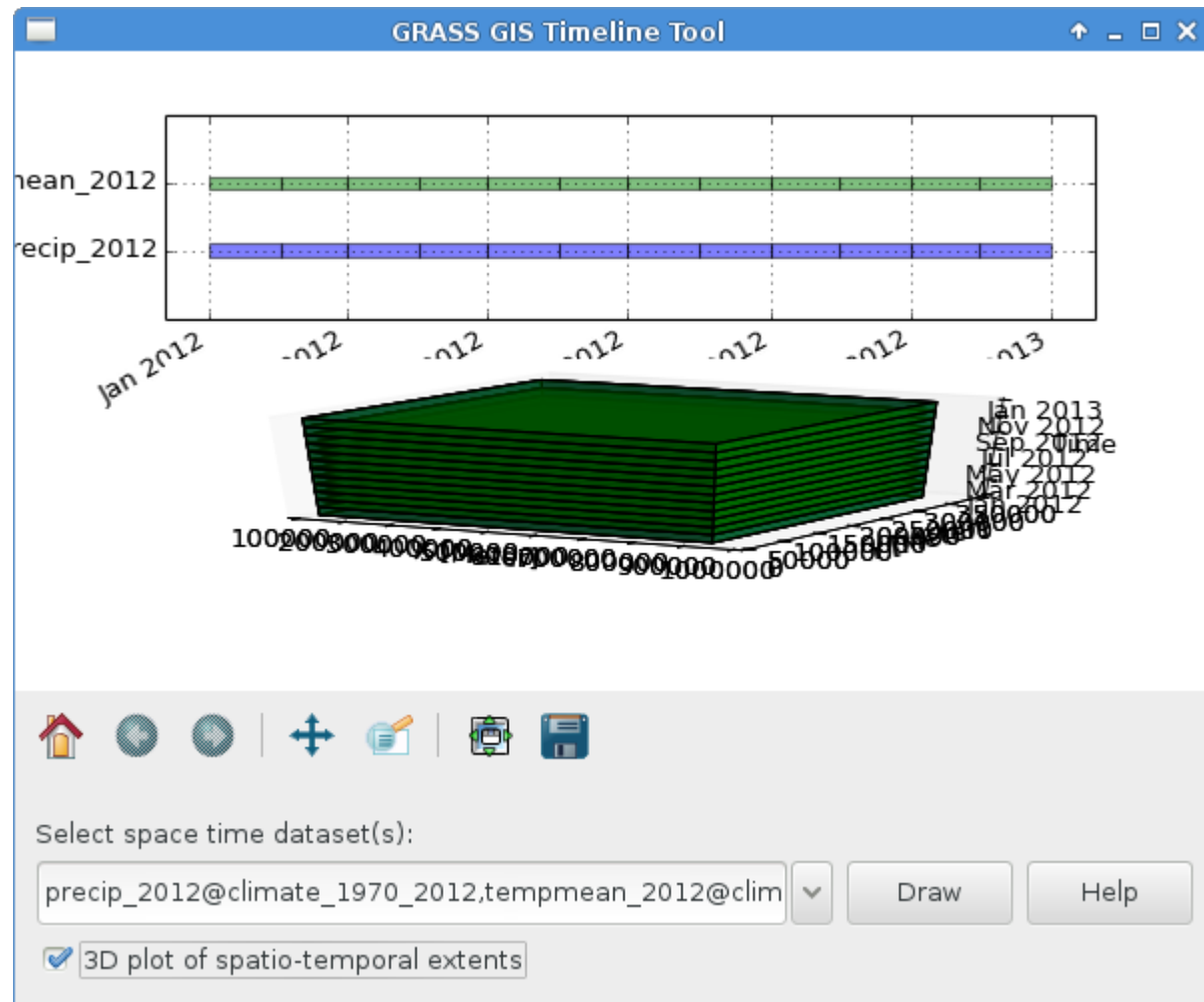
```
t.rast.colors input=precip_2012 rules=precipitation_monthly
```



Exercise – Timeline of the time series

Display timeline of precipitation and temperatures observations or aggregates

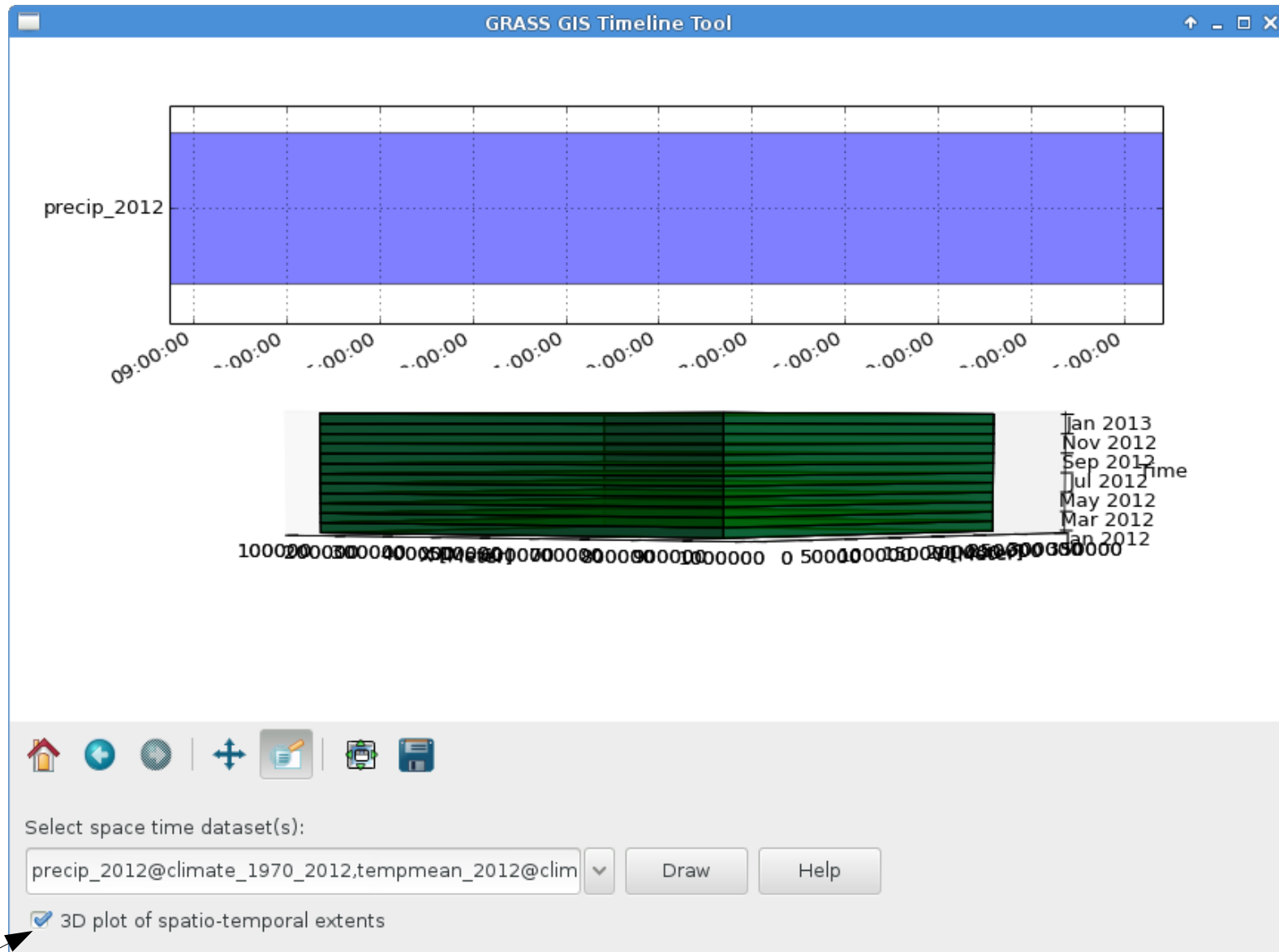
```
t.info tempmean_2012  
t.info precip_2012  
g.gui.timeline input=tempmean_2012,precip_2012
```





Exercise – Timeline of the time series

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



Exercise – Visualizing the time series: Animation



Display animation of precipitation and temperatures

```
# display legend, set min, max values in legend from t.info
t.info tempmean_2012
g.gui.animation strds=tempmean_2012

t.info precip_2012
g.gui.animation strds=precip_2012
```

