Spatio-temporal data processing & visualization in GRASS GIS

Who are we?

veroandreo

landam

pesekon2

neteler

lucadelu

mlennert

Brief intro to GRASS GIS

GRASS GIS: Brief history

- **GRASS GIS** (Geographic Resources Analysis Support System), is a FOSS GIS software suite used for geospatial data management and analysis, image processing, graphics and maps production, spatial modeling, and visualization.
- Used in academic and commercial settings around the world, as well as by many governmental agencies and consulting companies.
- Originally developed by the U.S. Army Construction Engineering Research Laboratories as a tool for land management and environmental planning by the military (USA-CERL, 1982-1995).

A bit of (geek) GRASS GIS history...

Advantages

- open source, you can use, modify, improve, share
- strong user community, commercial support
- large amount of tools
- both GUI and CLI (easy for scripting) interface
- Python API and libraries

Disadvantages

- complicated startup for newcomers
- native format (requires importing data, possibility of linking external formats)
- vector topology (confusing for beginners, sometimes tricky to import broken GIS data)

When to use GRASS GIS?

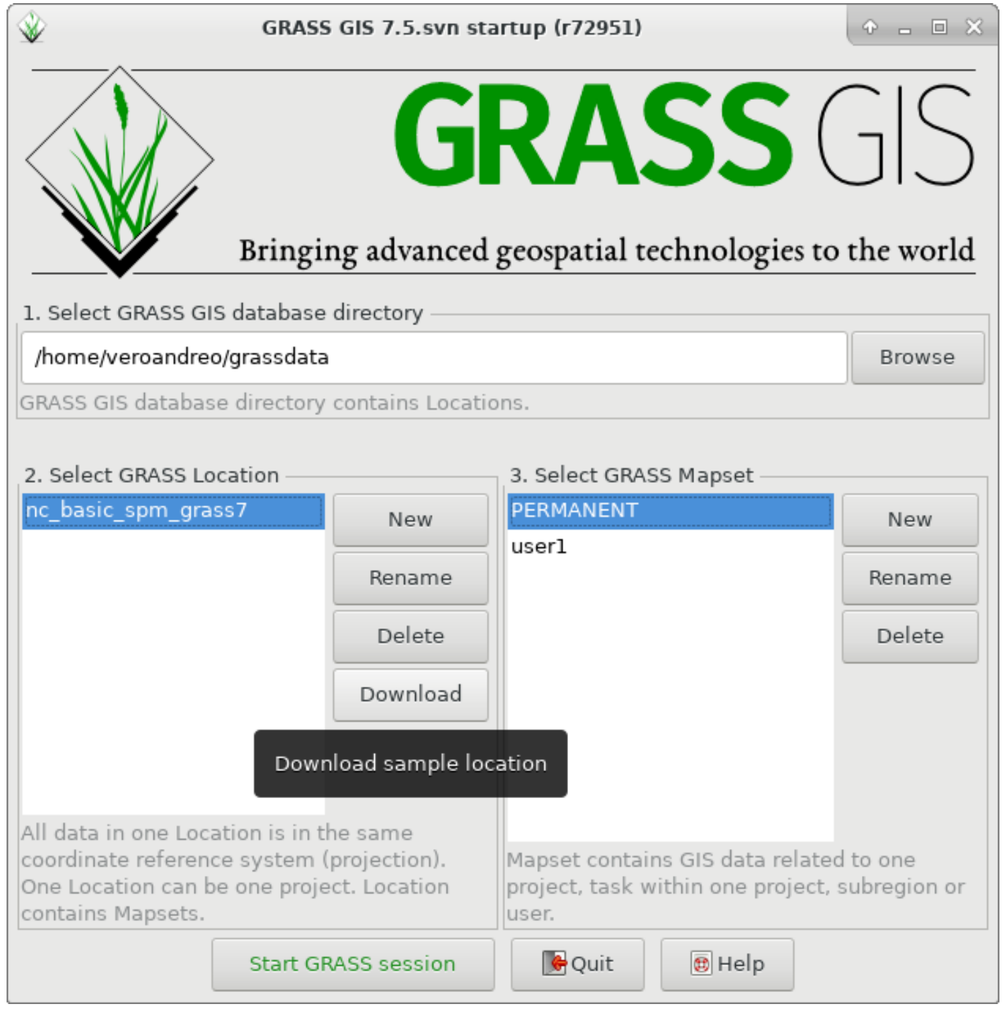
- doing (heavy) geospatial data analysis
- working with topological vector data
- analysing space-time datasets
- doing Python scripting
- deploying server-side applications (e.g. as WPS process)

When to use rather something else?

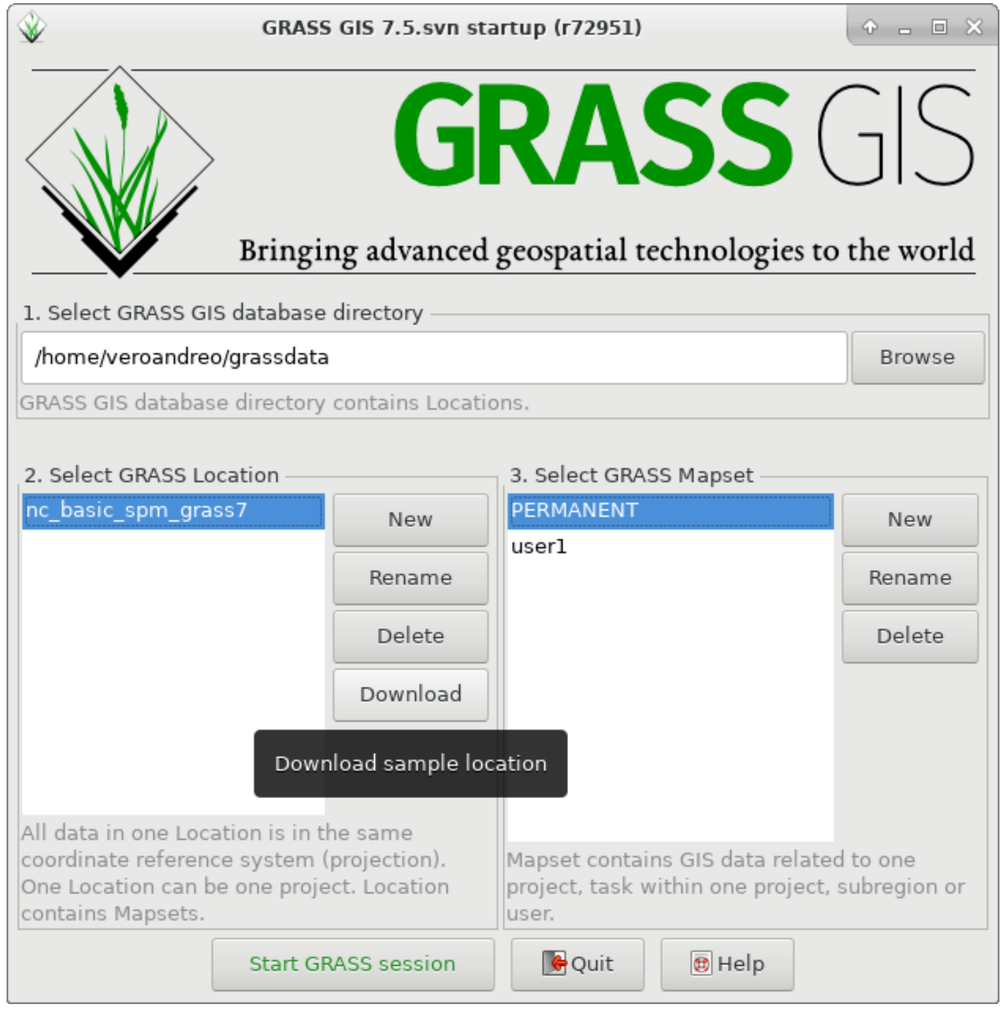
- want to vizualize geodata in easy and quick way (use QGIS instead)
- scared of location and mapsets ☺

Working with GRASS GIS is not much different than any other GIS...

Well, except for this...



Well, except for this...



Basic notions

Basic notions

- The **GRASS DATABASE** (or "GISDBASE") is an existing directory containing all GRASS GIS LOCATIONs.

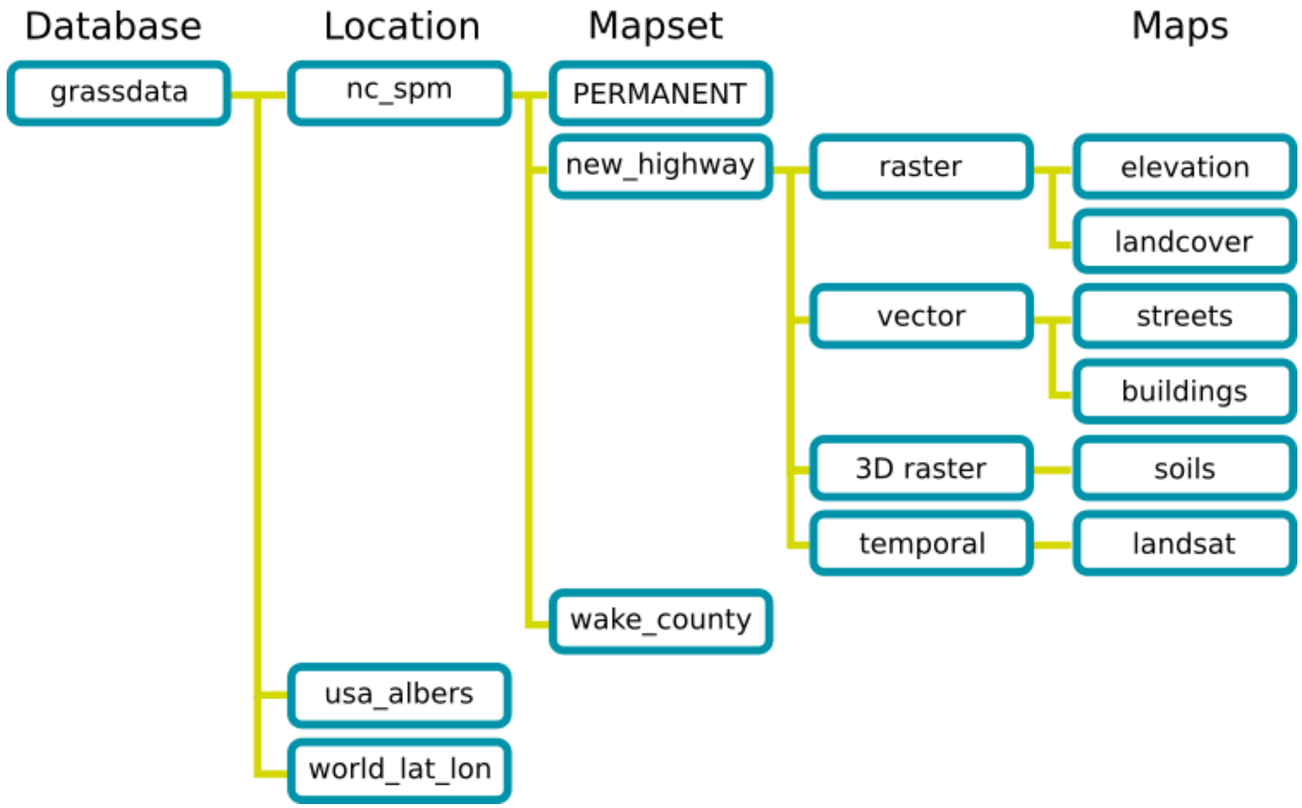
Basic notions

- The **GRASS DATABASE** (or "GISDBASE") is an existing directory containing all GRASS GIS LOCATIONs.
- A **LOCATION** is defined by its coordinate system, map projection and geographical boundaries.

Basic notions

- The **GRASS DATABASE** (or "GISDBASE") is an existing directory containing all GRASS GIS LOCATIONs.
- A **LOCATION** is defined by its coordinate system, map projection and geographical boundaries.
- **MAPSET** is a subdirectory within Locations. In a **MAPSET** you can organize GIS maps thematically, geographically, by project or however you prefer.

When GRASS GIS is started, it connects to the Database, Location and Mapset specified by the user



GRASS database

- **Why this structure?**
  - GRASS GIS has a *native format* for raster and vector data, therefore they must be *imported* or *linked* into a GRASS Location/Mapset (see r.external for example).

- **What are the advantages?**
  - GRASS DATABASE, LOCATIONs and MAPSETs are folders that *can be easily shared with other users*.
  - The GRASS DATABASE can be *local or remote*, and *special permissions* can be set to specific mapsets in a LOCATION.
  - All data in a LOCATION have necessarily the same CRS.

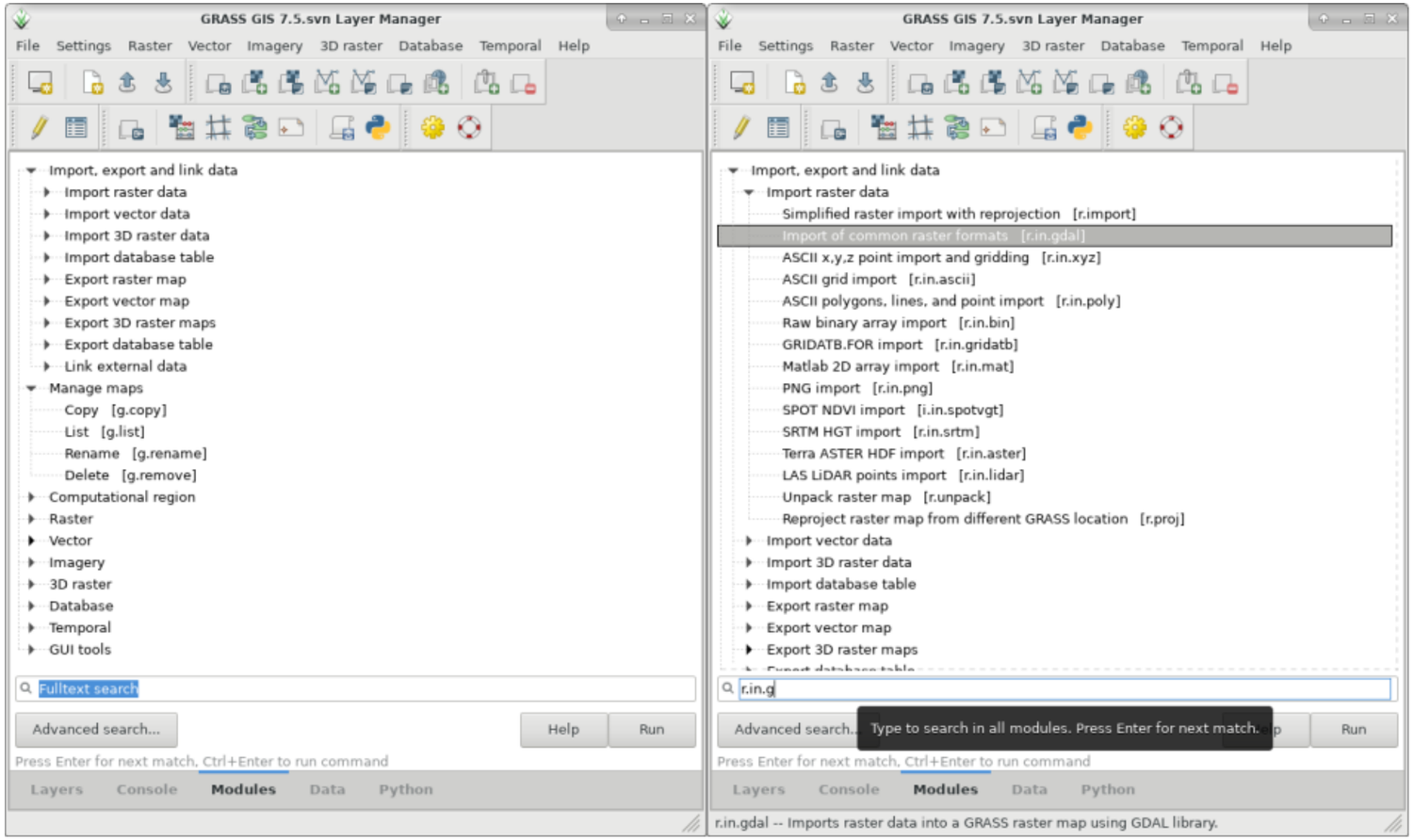
Data types in GRASS GIS

- Raster (including satellite imagery)
- 3D raster or voxel
- 2D and 3D Vector: point, line, centroid, boundary, area, face, volume
- Space-time datasets: collections of raster (**STRDS**), raster 3D (**STR3DS**) or vector (**STVDS**) maps

Modules

More than 500 modules but well structured:

Prefix	Function class	Type of command	Example
g.*	general	general data management	g.rename: renames map
d.*	display	graphical output	d.rast: display raster map
r.*	raster	raster processing	r.mapcalc: map algebra
v.*	vector	vector processing	v.clean: topological cleaning
i.*	imagery	imagery processing	i.pca: Principal Components Analysis on imagery group
r3.*	voxel	3D raster processing	r3.stats: voxel statistics
db.*	database	database management	db.select: select value(s) from table
ps.*	postscript	PostScript map creation	ps.map: PostScript map creation
t.*	temporal	space-time datasets	t.rast.aggregate: raster time series aggregation



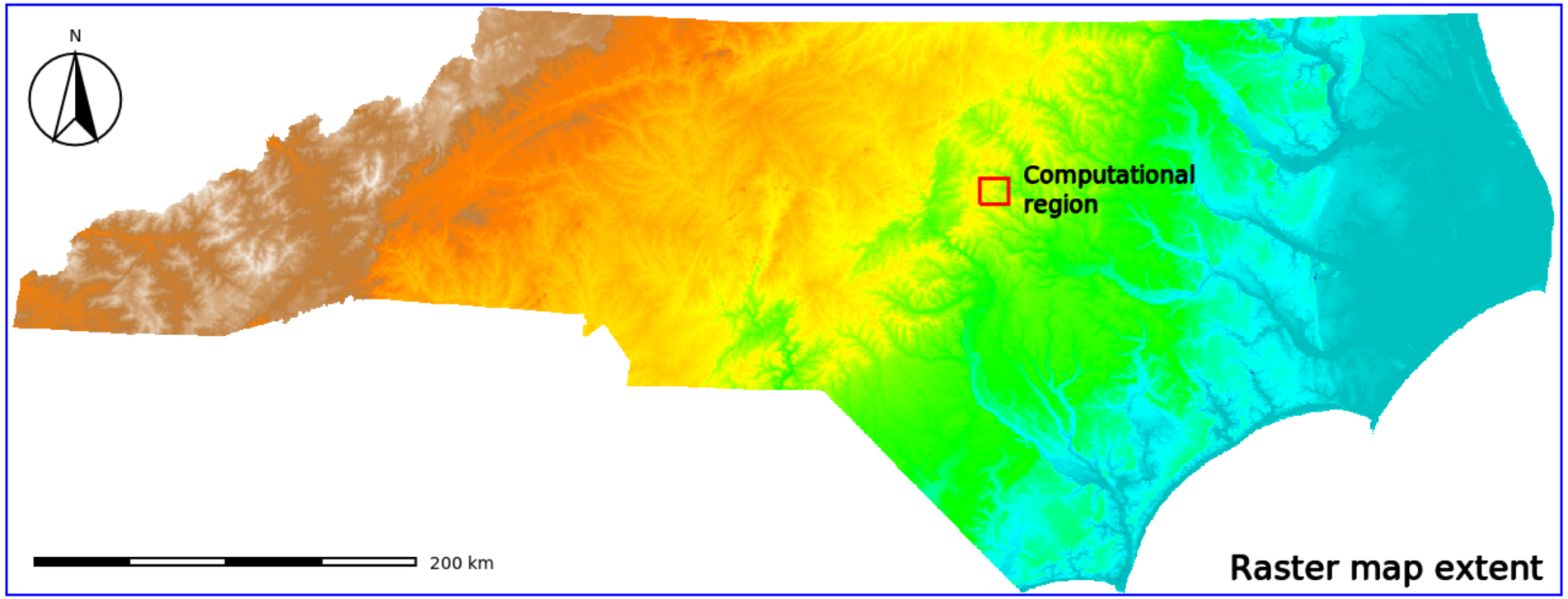
Module tree and module search engine

Add-ons

Plugins or **Add-ons** can be installed from a centralized OSGeo repository or from github (or similar repositories) using g.extension command.

```
# install extension from GRASS GIS Add-on repository
g.extension extension=r.hants

# install extension from github repository
g.extension extension=r3.slice \
  url=https://github.com/petrasovaa/r3.slice
```

Computational region



The **computational region** is the *actual setting of the region boundaries and the actual raster resolution*.
The **computational region** can be set and changed by means of g.region to the extent of a vector map, a raster map or manually to some area of interest.
Output raster maps will have their extent and resolution equal to those of the current computational region, while vector maps are always considered in their original extent.

Computational region

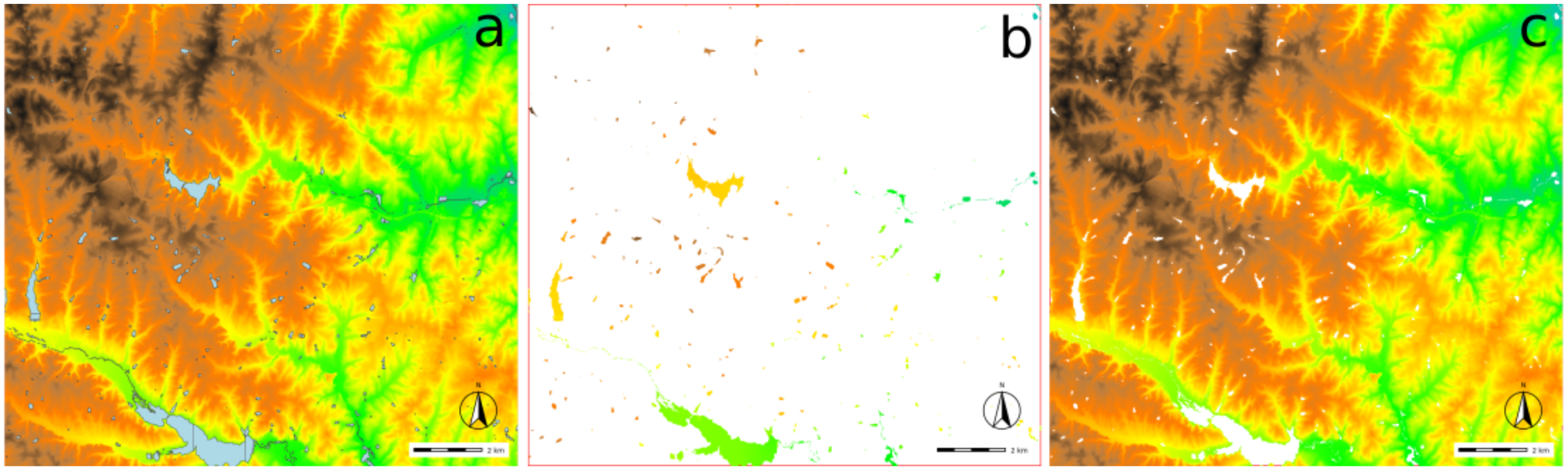
- **Which are the advantages?**
  - Keep your results consistent
  - Avoid clipping maps prior to subarea analysis
  - Test an algorithm or computationally demanding process in small areas
  - Fine-tune the settings of a certain module
  - Run different processes in different areas

More details at the Computational region wiki

MASK

- A raster map named MASK can be created to mask out areas
- All cells that are *NULL* in the MASK map will be ignored (also all areas outside the computational region).
- Masks are set with r.mask or creating a raster map called **MASK**.

Vector maps can be also used as masks



a- Elevation raster and lakes vector maps. b- Only the raster data inside the masked area are used for further analysis. c- Inverse mask.

MASK examples

```
# use vector as mask
r.mask vector=lakes

# use vector as mask, set inverse mask
r.mask -i vector=lakes

# mask categories of a raster map
r.mask raster=landclass96 maskcats="5 thru 7"

# create a raster named MASK
r.mapcalc expression="MASK = if(elevation < 100, 1, null())"

# remove mask
r.mask -r
```

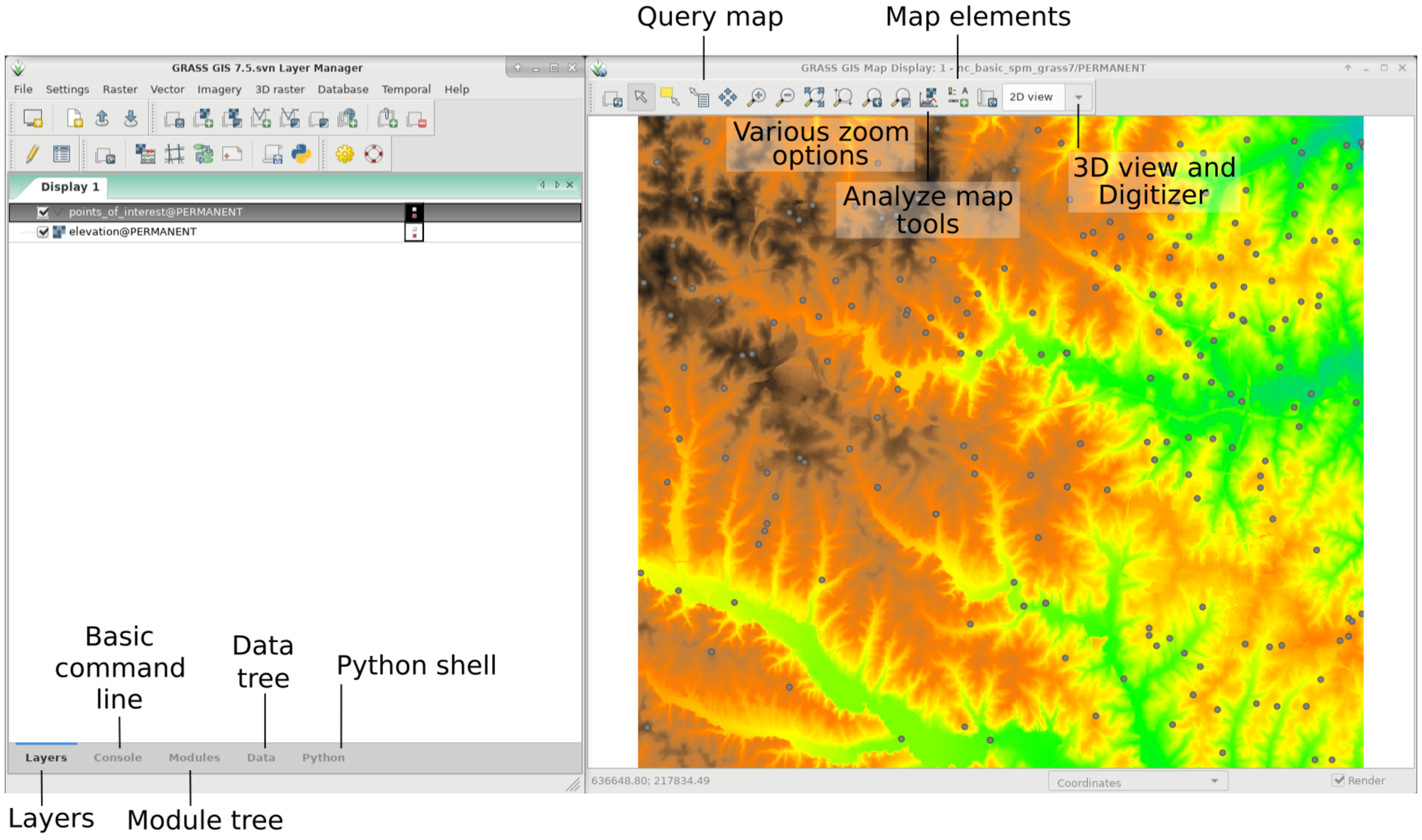
Note: A mask is only actually applied when reading a GRASS raster map, i.e., when used as input in a module.

Interfaces

GRASS GIS offers different interfaces for the interaction between user and software.

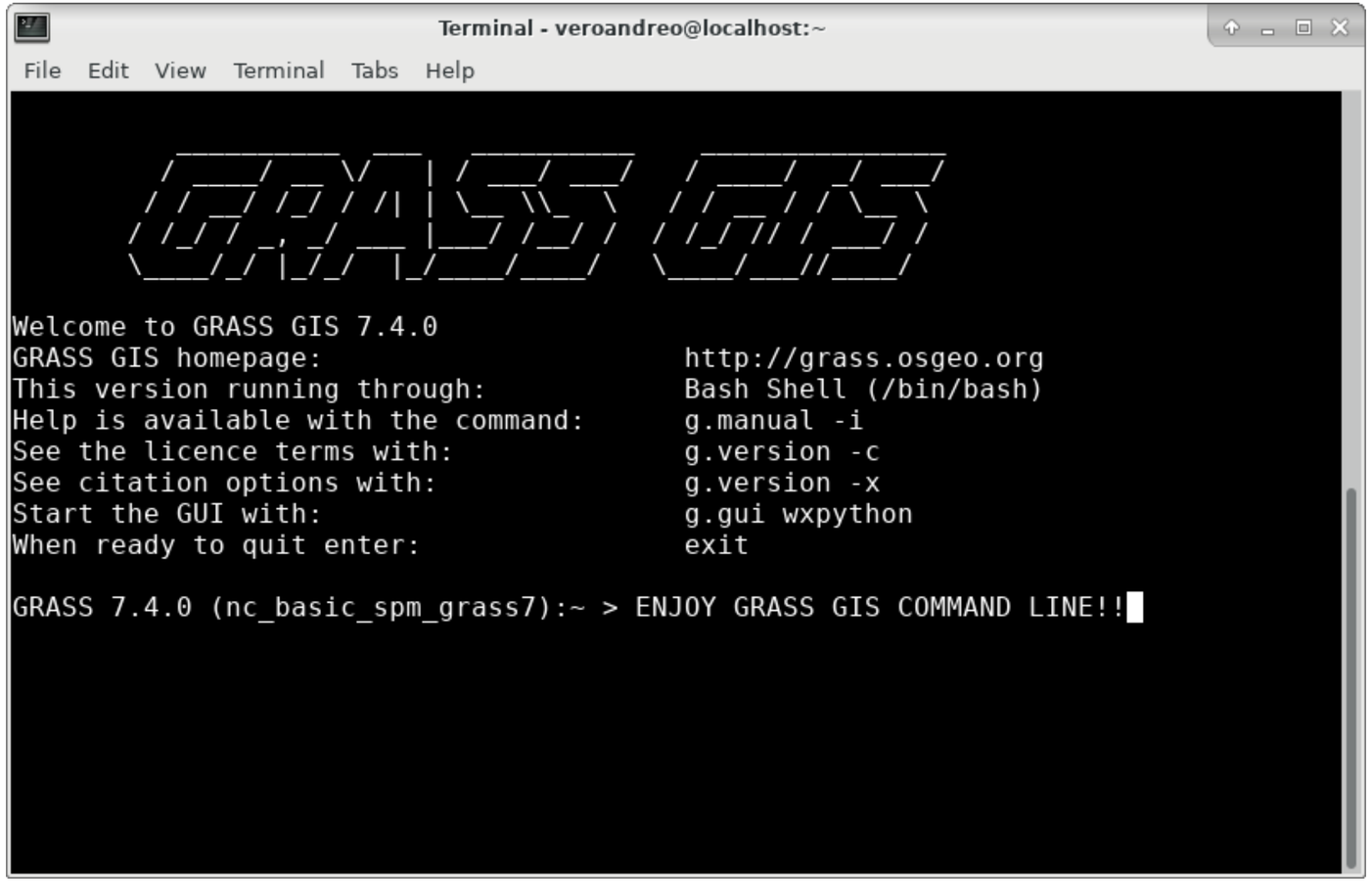
Let's see them!

Graphical User Interface (GUI)



>_ Command line

The most powerful way to use GRASS GIS!!



Advantages of the command line

Advantages of the command line

- Run `history` to see all your previous commands

Advantages of the command line

- Run `history` to see all your previous commands
- History is stored individually per MAPSET

Advantages of the command line

- Run `history` to see all your previous commands
- History is stored individually per MAPSET
- Search in history with `CTRL-R`

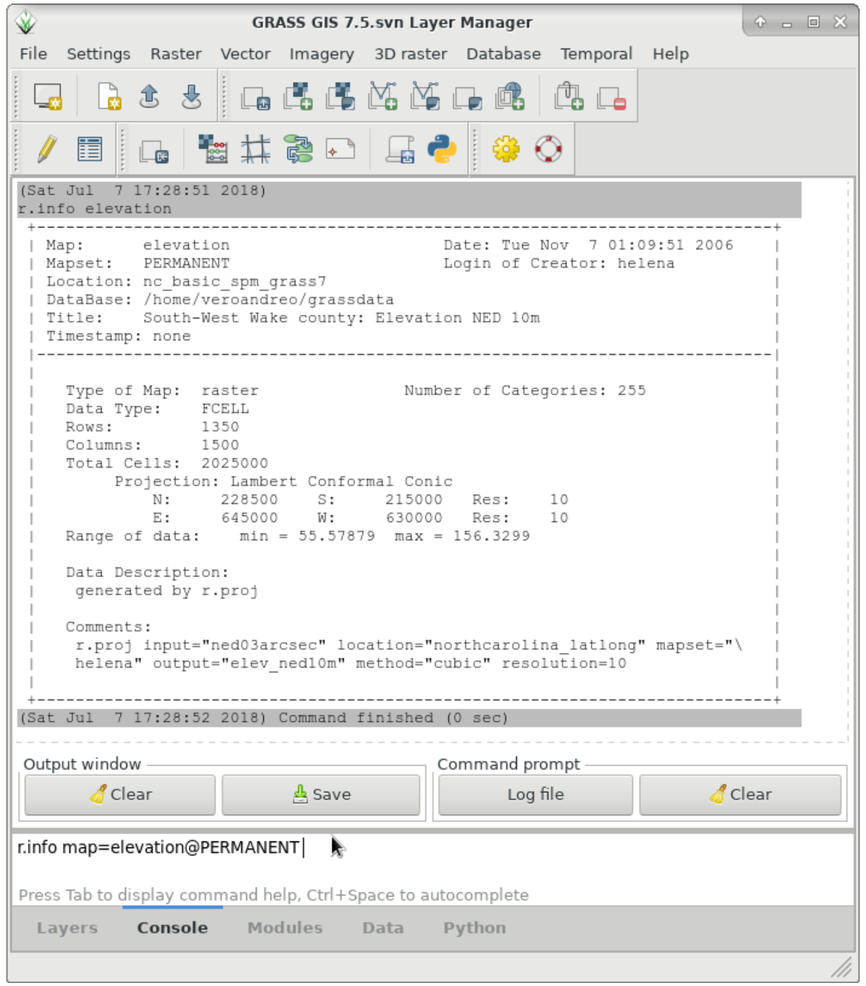
Advantages of the command line

- Run `history` to see all your previous commands
- History is stored individually per MAPSET
- Search in history with `CTRL-R`
- Save the commands to a file: `history > my_protocol.sh`, polish/annotate the protocol and re-run with: `sh my_protocol.sh`

Advantages of the command line

- Run `history` to see all your previous commands
- History is stored individually per MAPSET
- Search in history with `CTRL-R`
- Save the commands to a file: `history > my_protocol.sh`, polish/annotate the protocol and re-run with: `sh my_protocol.sh`
- Call module's GUI and "Copy" the command for further replication

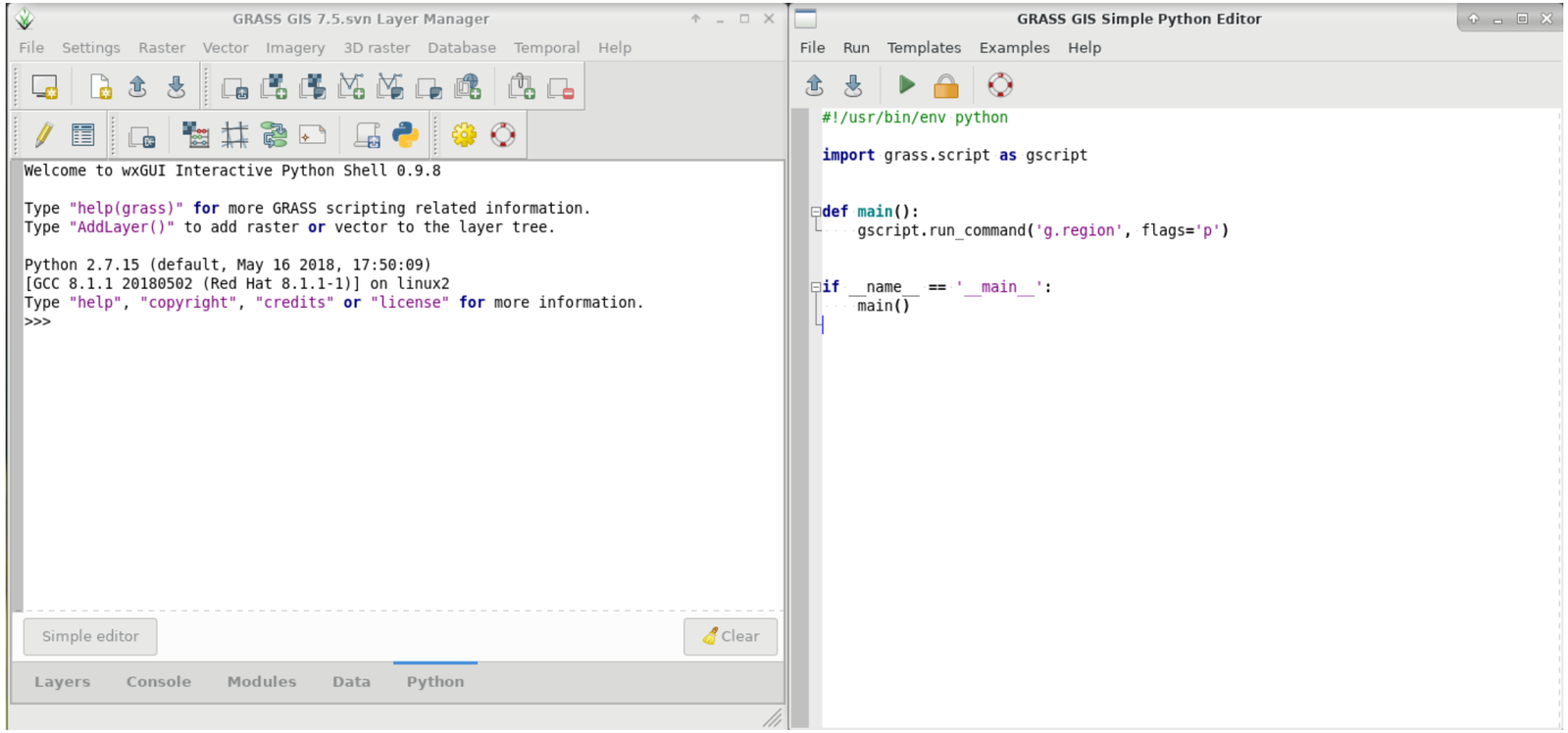
The GUI's simplified command line offers a *Log file* button to save the history to a file



Python

- 2 libraries delivered with GRASS GIS: **grass** and **pygrass**, that provide access to modules and internal C functions
- 1 external library: **grass-session** (install with `pip`)
- They can be combined and they integrate well with Jupyter notebooks

The simplest way to execute a Python script is through the Simple Python editor



... or write your Python script in your favorite editor and run it:

```
#!/usr/bin/env python

# simple example for pyGRASS usage: raster processing via modules approach
from grass.pygrass.modules.shortcuts import general as g
from grass.pygrass.modules.shortcuts import raster as r
g.message("Filter elevation map by a threshold...")

# set computational region
input = 'elevation'
g.region(raster=input)
output = 'elev_100m'
thresh = 100.0

r.mapcalc("%s = if(%s > %d, %s, null())" % (output, input, thresh, input), overwri
r.colors(map=output, color="elevation")
```

... or use the **grass-session** Python library

```
#!/usr/bin/env python

# Markus Neteler, 2018
#
# based on
# https://grasswiki.osgeo.org/wiki/Working_with_GRASS_without_starting_it_explicitl
#
# Requirements:
#               'sudo pip install grass-session'


# define where to process the data in the temporary grass-session
mygisdb = '/tmp/grassdata'
mylocation = 'world'
mymapset = 'user'

# the next line starts the GRASS GIS session
from grass_session import Session

# import some convenient GRASS GIS Python API parts
from grass.script import core as gcore
```

Credits: Pietro Zambelli. See grass-session GitHub for further details.

... or use the **grass-session** Python library

```
mygisdb = '/tmp/grassdata'
mylocation = 'world'
mymapset = 'user'

# the next line starts the GRASS GIS session
from grass_session import Session

# import some convenient GRASS GIS Python API parts
from grass.script import core as gcore
import grass.script as gscript
import grass.script.setup as gsetup
# import grass python libraries
from grass.pygrass.modules.shortcuts import general as g
from grass.pygrass.modules.shortcuts import raster as r
from grass.pygrass.modules.shortcuts import vector as v
from grass.pygrass.modules.shortcuts import temporal as t


# set some common environmental variables, like for raster compression settings:
import os
os.environ.update(dict(GRASS_COMPRESS_NULLS='1'))
```

Credits: Pietro Zambelli. See grass-session GitHub for further details.

... or use the **grass-session** Python library

```
import os
os.environ.update(dict(GRASS_COMPRESS_NULLS='1'))
#  needs G75:              GRASS_COMPRESSOR='ZSTD'))



# create a PERMANENT mapset; create a Session instance
PERMANENT = Session()
# hint: EPSG code lookup: https://epsg.io
PERMANENT.open(gisdb=mygisdb, location=mylocation,
               create_opts='EPSG:4326')

# exit from PERMANENT right away in order to perform analysis in our own mapset
PERMANENT.close()

# create a new mapset in the same location
user = Session()
user.open(gisdb=mygisdb, location=mylocation, mapset=mymapset,
               create_opts='')

# execute some command inside user mapset
```

Credits: Pietro Zambelli. See grass-session GitHub for further details.

... or use the **grass-session** Python library

```
user.open(gisdb=mygisdb, location=mylocation, mapset=mymapset,
                create_opts='')

# execute some command inside user mapset

# import admin0 vector data - it downloads and imports including topological cleani
#   Data source: https://www.naturalearthdata.com/downloads/10m-cultural-vectors/
#   VSI driver for remote access: http://www.gdal.org/gdal_virtual_file_systems.htm
inputfile = "/vsizip/vsicurl/https://www.naturalearthdata.com/http//www.naturaleart

# note the pythonic underscore
v.in_ogr(input=inputfile, output="countries", overwrite = True)

# show the attribute column names
v.info(map="countries", flags="c")


# list vector maps available in the current mapset
g.list(type="vector", flags="m")

# now do anaylsis with the map...
```

Credits: Pietro Zambelli. See grass-session GitHub for further details.

... or use the **grass-session** Python library

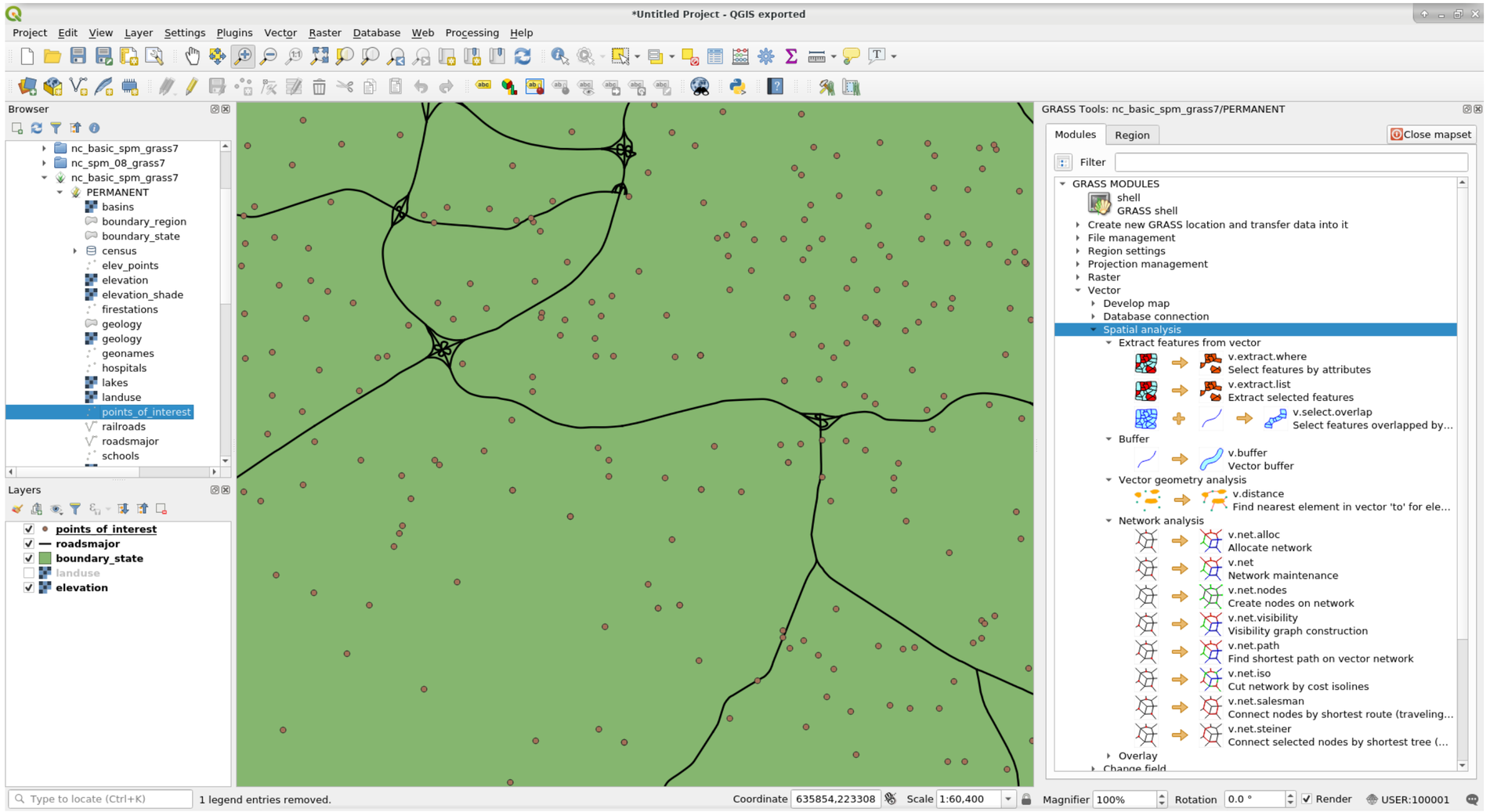
```
                create_opts='')

# execute some command inside user mapset

# import admin0 vector data - it downloads and imports including topological clean
#   Data source: https://www.naturalearthdata.com/downloads/10m-cultural-vectors/
#   VSI driver for remote access: http://www.gdal.org/gdal_virtual_file_systems.htm
inputfile = "/vsizip/vsicurl/https://www.naturalearthdata.com/http//www.naturaleart

# note the pythonic underscore
v.in_ogr(input=inputfile, output="countries", overwrite = True)

# show the attribute column names
v.info(map="countries", flags="c")


# list vector maps available in the current mapset
g.list(type="vector", flags="m")

# now do anaylsis with the map...

# exit from user mapset (the data will remain)
```

Credits: Pietro Zambelli. See grass-session GitHub for further details.

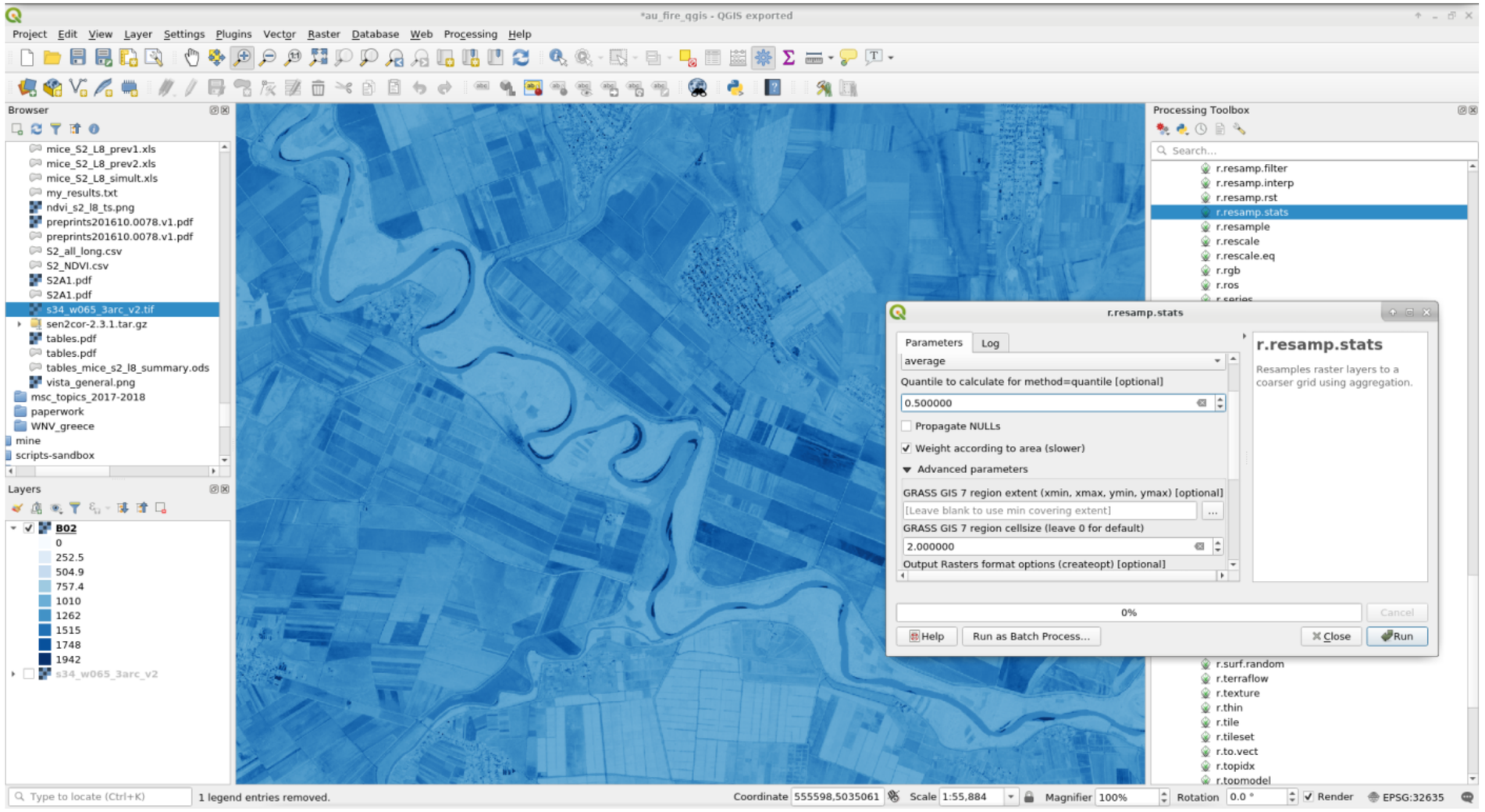
QGIS

There are two ways to use GRASS GIS functionalities within QGIS:

- GRASS GIS plugin
- Processing toolbox



Using GRASS GIS modules through the GRASS Plugin in QGIS

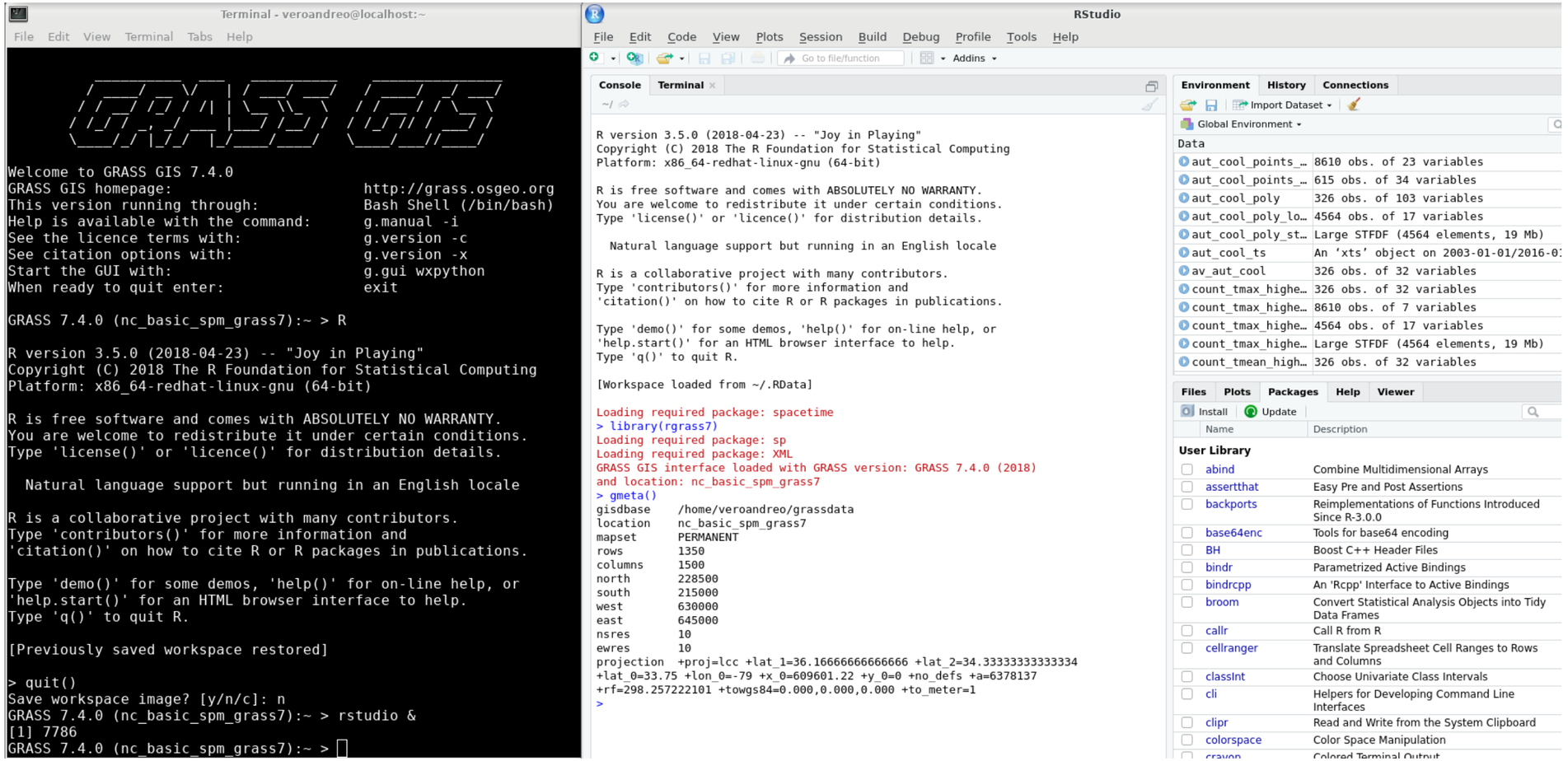


Using GRASS GIS modules through the Processing Toolbox

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

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

Details and examples at the GRASS and R wiki


Terminal - veroandreo@localhost:~
File Edit View Terminal Tabs Help
Welcome to GRASS GIS 7.4.0
GRASS GIS homepage: http://grass.osgeo.org
This version running through: Bash Shell (/bin/bash)
Help is available with the command: g.manual -i
See the licence terms with: g.version -c
See citation options with: g.version -x
Start the GUI with: g.gui wxpython
When ready to quit enter: exit
GRASS 7.4.0 (nc_basic_spm_grass7):~ > R
R version 3.5.0 (2018-04-23) -- "Joy in Playing"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-redhat-linux-gnu (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Previously saved workspace restored]
> quit()
Save workspace image? [y/n/c]: n
GRASS 7.4.0 (nc_basic_spm_grass7):~ > rstudio &
[1] 7786
GRASS 7.4.0 (nc_basic_spm_grass7):~ >
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Console Terminal
R version 3.5.0 (2018-04-23) -- "Joy in Playing"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-redhat-linux-gnu (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Workspace loaded from ~/.RData]
Loading required package: spacetime
> library(rgrass7)
Loading required package: sp
Loading required package: XML
GRASS GIS interface loaded with GRASS version: GRASS 7.4.0 (2018)
and location: nc_basic_spm_grass7
> gmeta()
gisdbase /home/veroandreo/grassdata
location nc_basic_spm_grass7
mapset PERMANENT
rows 1350
columns 1500
north 228500
south 215000
west 630000
east 645000
nsres 10
ewres 10
projection +proj=lcc +lat_1=36.16666666666666 +lat_2=34.33333333333334
+lat_0=33.75 +lon_0=-79 +x_0=609601.22 +y_0=0 +no_defs +a=6378137
+rf=298.257222101 +towgs84=0.000,0.000,0.000 +to_meter=1
>
Environment History Connections
Import Dataset
Global Environment
Data
aut_cool_points_... 8610 obs. of 23 variables
aut_cool_points_... 615 obs. of 34 variables
aut_cool_poly 326 obs. of 103 variables
aut_cool_poly_lo... 4564 obs. of 17 variables
aut_cool_poly_st... Large STFDF (4564 elements, 19 Mb)
aut_cool_ts An 'xts' object on 2003-01-01/2016-0
av_aut_cool 326 obs. of 32 variables
count_tmax_highe... 326 obs. of 32 variables
count_tmax_highe... 8610 obs. of 7 variables
count_tmax_highe... 4564 obs. of 17 variables
count_tmax_highe... Large STFDF (4564 elements, 19 Mb)
count_tmean_high... 326 obs. of 32 variables
Files Plots Packages Help Viewer
Install Update
Name Description
User Library
abind Combine Multidimensional Arrays
assertthat Easy Pre and Post Assertions
backports Reimplementations of Functions Introduced Since R-3.0.0
base64enc Tools for base64 encoding
BH Boost C++ Header Files
bindr Parametrized Active Bindings
bindrcpp An 'Rcpp' Interface to Active Bindings
broom Convert Statistical Analysis Objects into Tidy Data Frames
callr Call R from R
cellranger Translate Spreadsheet Cell Ranges to Rows and Columns
classInt Choose Univariate Class Intervals
cli Helpers for Developing Command Line Interfaces
clipr Read and Write from the System Clipboard
colorspace Color Space Manipulation

WPS - OGC Web Processing Service

- Web Processing Service is an OGC standard.
- ZOO-Project and PyWPS allow the user to run GRASS GIS commands in a simple way through the web.
- See GRASS GIS Workshop in Jena for PyWPS-based examples.

actinia: The GRASS GIS REST API

- Open source REST API for scalable, distributed, high performance processing of geo data using GRASS GIS
- Core functionality: processing of geo-data and time series of satellite images, raster and vector data
- OSGeo Community Project since 2019

Don't miss *actinia* talks at FOSS4G 2019!

- GRASS GIS in the cloud: actinia geoprocessing
- How digging in the earth for the fibre roll-out took GRASS to the cloud

https://actinia.mundialis.de/
DOI:10.5281/zenodo.2631917

Some useful commands and cool stuff

- r.import and v.import: import of raster and vector maps with reprojection, subsetting and resampling on the fly.

```
## IMPORT RASTER DATA: SRTM V3 data for NC

# set computational region to e.g. 10m elevation model:
g.region raster=elevation -p

# import with reprojection on the fly
r.import input=n35_w079_1arc_v3.tif output=srtmv3_resamp10m \
 resample=bilinear extent=region resolution=region \
 title="SRTM V3 resampled to 10m resolution"

## IMPORT VECTOR DATA

# import SHAPE file, clip to region extent and reproject to
# current location projection
v.import input=research_area.shp output=research_area extent=region
```

- g.list: Lists available GRASS data base files of the user-specified data type (i.e., raster, vector, 3D raster, region, label) optionally using the search pattern.

```
g.list type=vector pattern="r*"
g.list type=vector pattern="[ra]*"
g.list type=raster pattern="{soil,landuse}*"
```

- g.remove, g.rename and g.copy:

  These modules remove maps from the GRASSDBASE, rename maps and copy maps either in the same mapset or from other mapset.

Always perform these tasks from within GRASS

- g.region: Manages the boundary definitions and resolution for the computational region.

```
## Subset a raster map
# 1. Check region settings
g.region -p
# 2. Change region relative to current N and W values
g.region n=n-3000 w=w+4000
# 3. Subset map
r.mapcalc "new_elev = elevation"
r.colors new_elev color=viridis
# 4. Display maps
d.mon wx0
d.rast elevation
d.rast new_elev
```

- r.info and v.info: useful to get basic info about maps as well as their history.

```
# info for raster map
r.info elevation
# info for vector map
v.info nc_state
# history of vector map
v.info nc_state -h
```

- --exec in the grass76 startup command: This flag allows to run modules or complete workflows written in Bash shell or Python without starting GRASS GIS.

```
# running a module
grass76 $HOME/grassdata/nc_spm_08_grass7/PERMANENT/ --exec r.info elevation

# running a script
grass76 $HOME/grassdata/nc_spm_08_grass7/PERMANENT/ --exec sh test.sh

## test.sh might be as simple as:

#!/bin/bash

g.region -p
g.list type=raster
r.info elevation
```

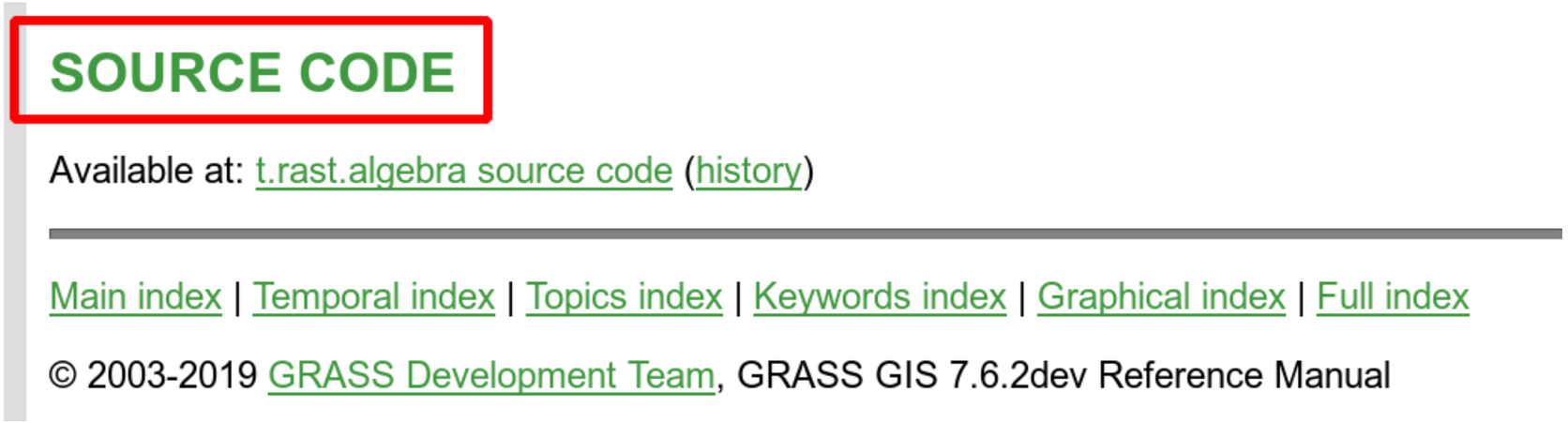
HELP!!!

KEEP CALM and GRASS GIS

- g.manual: in the main GUI under Help or just pressing *F1*
- `--help` or `--h` flag after the module name
- GRASS wiki: examples, explanations and help on particular modules or tasks, tutorials, applications, news, etc.
- Jupyter/IPython notebooks with example workflows for different applications
- grass-user mailing list: Just subscribe and post or check the archives.
- Facebook group

If you are more curious...

- Link to source code and history in each module manual page, eg., t.rast.algebra



SOURCE CODE

Available at: t.rast.algebra source code (history)

Main index | Temporal index | Topics index | Keywords index | Graphical index | Full index

References

- Neteler, M., Mitasova, H. (2008): *Open Source GIS: A GRASS GIS Approach*. Third edition. ed. Springer, New York. Book site
- Neteler, M., Bowman, M.H., Landa, M. and Metz, M. (2012): *GRASS GIS: a multi-purpose Open Source GIS*. Environmental Modelling & Software, 31: 124-130 DOI

Other (very) useful links

- GRASS intro workshop held at NCSU
- Unleash the power of GRASS GIS at US-IALE 2017
- GRASS GIS course in Jena 2018
- GRASS GIS course IRSAE 2018
- GRASS GIS course in Argentina 2018

QUESTIONS?

Let's move on to:

TGRASS presentation