

GRASS GIS 7 workshop

Course Introduction
GRASS GIS 7 Overview

Markus Neteler

OSGeo Ireland – 1st National Symposium 2017

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



About the trainer

Markus Neteler: Germany – Italy – Germany

GRASS GIS since 1993, FOSSGIS.de, GFOSS.it, OSGeo.org, etc.

Worked in research from 1999-2016 (mainly in Italy)

Since 2016: partner and general manager at mundialis, Bonn (DE)

mundialis GmbH & Co. KG

- founded in 2015 in Bonn by T. Adams, H. Paulsen and M. Neteler
- at time 7 staff
- massive GIS data processing and Earth Observation
- offers decades of experience in Open Source GIS (especially GRASS GIS development)
- gained HPC experience through processing of MODIS Land Surface Temperature : “EuroLST”
 - 15 years of gap free daily data at 250m resolution

Dr. Markus Neteler

mundialis GmbH & Co. KG

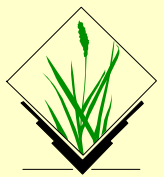
Kölnstraße 99

53111 Bonn, Germany



Email: neteler@mundialis.de

Web: <http://www.mundialis.de>



1) Introduction

Who is the presenter

Course overview: structure

What is GRASS GIS, OSGeolive

2) GRASS GIS Software first steps

- Intro QGIS-Processing-GRASS GIS
- Data: course data: North Carolina
- Using GRASS GIS in QGIS through "Processing"

3) GRASS GIS general introduction

- Database structure of GRASS GIS
- About the course data set
- First steps in using GRASS GIS 7
 - Graphical user interface (GUI)
 - GRASS command structure
 - Command line or GUI?
 - Creating a perspective view


4) GRASS GIS raster introduction

- raster processing concepts
- import of a GeoTIFF (DEM)
- Color tables, NULL masks etc
- hydrological modelling
- raster capabilities in GRASS GIS

5) GRASS GIS vector introduction

- Why a topological vector data model
- Vector feature extraction
- Vector geometry dissolving
- Geometry editing/digitizing
- Import/export
- Capabilities of the vector engine

6) Outlook and diskussion

An aerial satellite photograph of a coastal area, likely a bay or estuary, showing green land, blue water, and white sandy beaches. A semi-transparent white rectangular box is centered over the image, containing text.

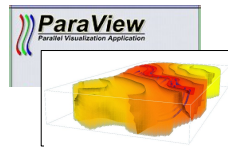
Short guide to

GRASS GIS

What's GRASS GIS?

<http://grass.osgeo.org>

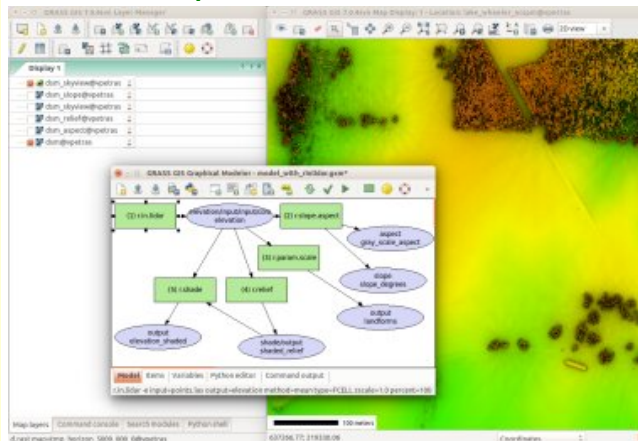
- **Geographic Resources Analysis Support System**
- **Open Source GIS**, developed since 1984, since 1999 GNU GPL
- **Portable code** (many operating systems, 32/64bit)
- Your **GIS backbone** – linkable to:



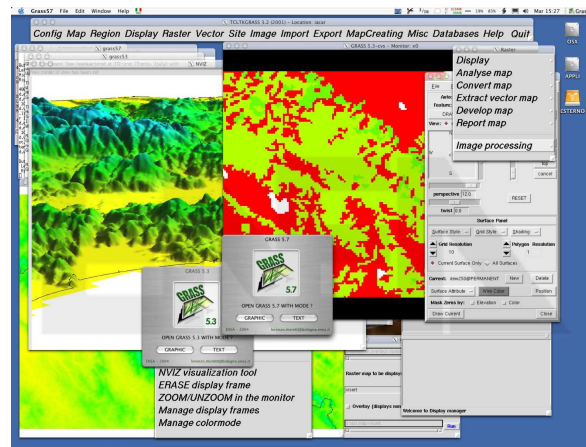
ZOO
<http://zoo-project.org>



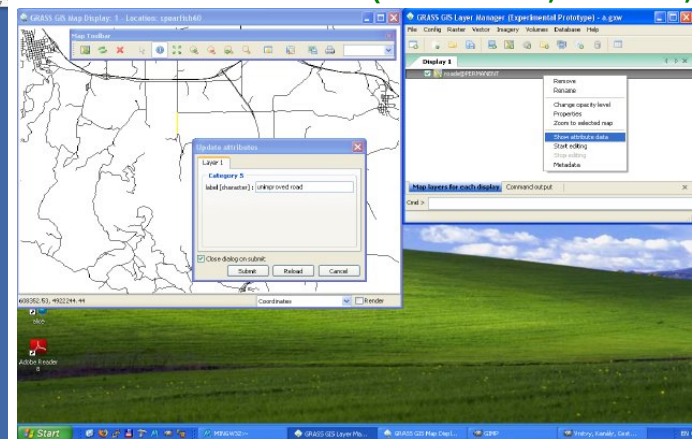
GNU/Linux



MacOSX



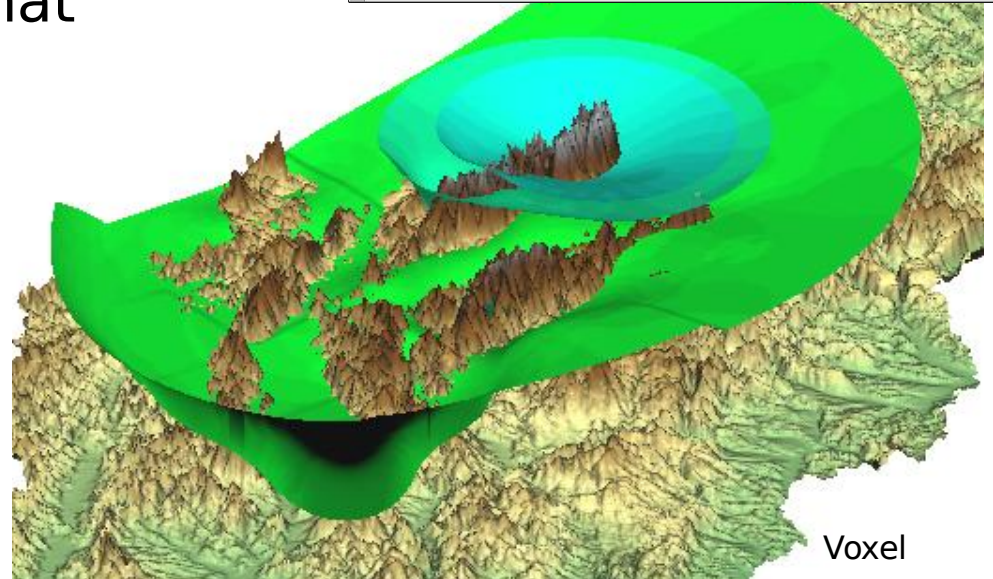
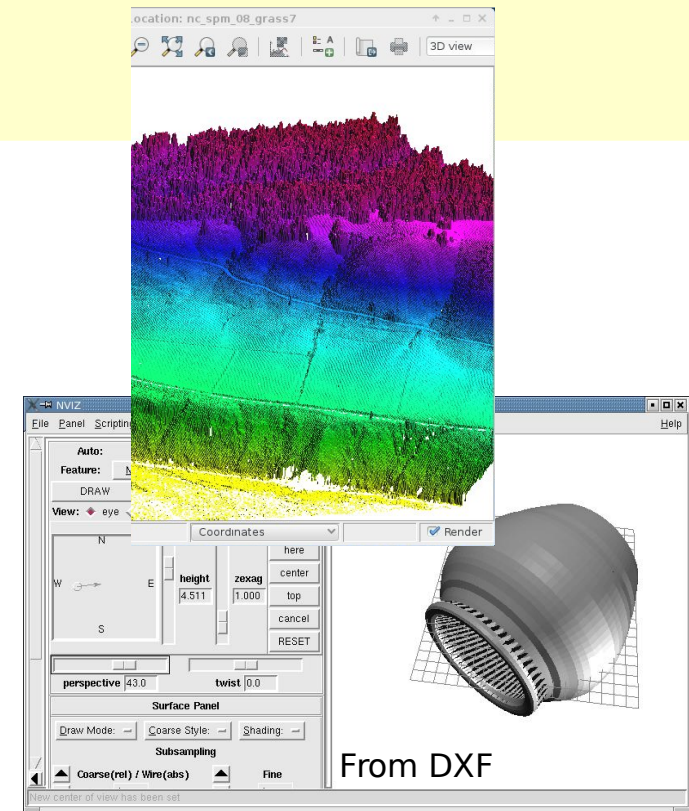
MS-Windows



(IBM AIX, *BSD, ...)

What's GRASS GIS?

- Raster 2D/3D (voxel) processing
- Vector 2D/3D topological processing
- Vector network analysis support
- Image processing system
- Space-time cubes, temporal GIS
- Native raster and vector format
- 3D Visualization system
- DBMS integrated (SQL) with SQLite, DBF, PostgreSQL, MySQL, and ODBC drivers



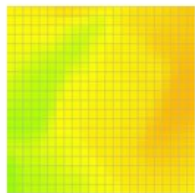
GRASS GIS 7 capabilities: a graphical overview:
<http://www.slideshare.net/markusN/grass-gis-7-capabilities-a-graphical-overview>

What's GRASS GIS?

Graphical index of GRASS GIS modules



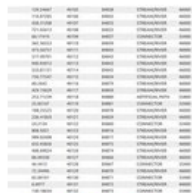
Graphical index of GRASS GIS modules



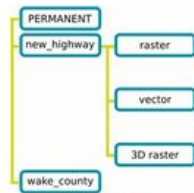
Raster



Vector



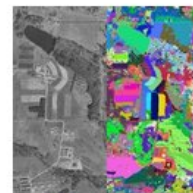
Database



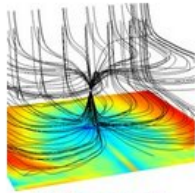
General



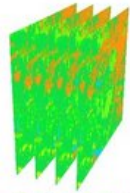
Display



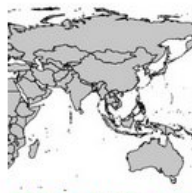
Imagery



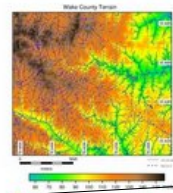
3D raster



Temporal



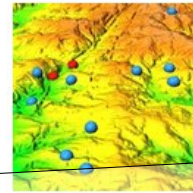
Miscellaneous



Cartography



GUI



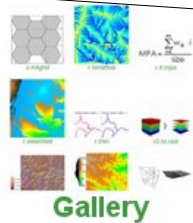
3D view



Python

C library
{for C/C++}

C library



Gallery



Graphical index of GRASS GIS modules

Go to [3D raster introduction](#) | [topics](#)

3d raster modules:

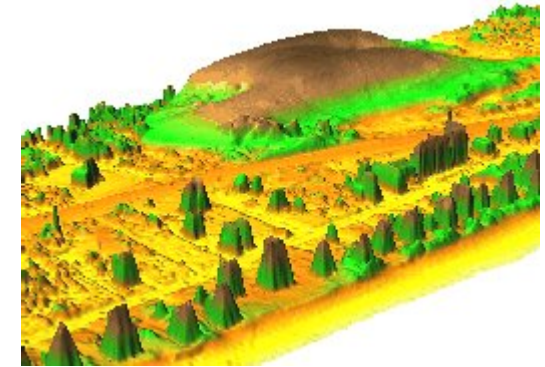
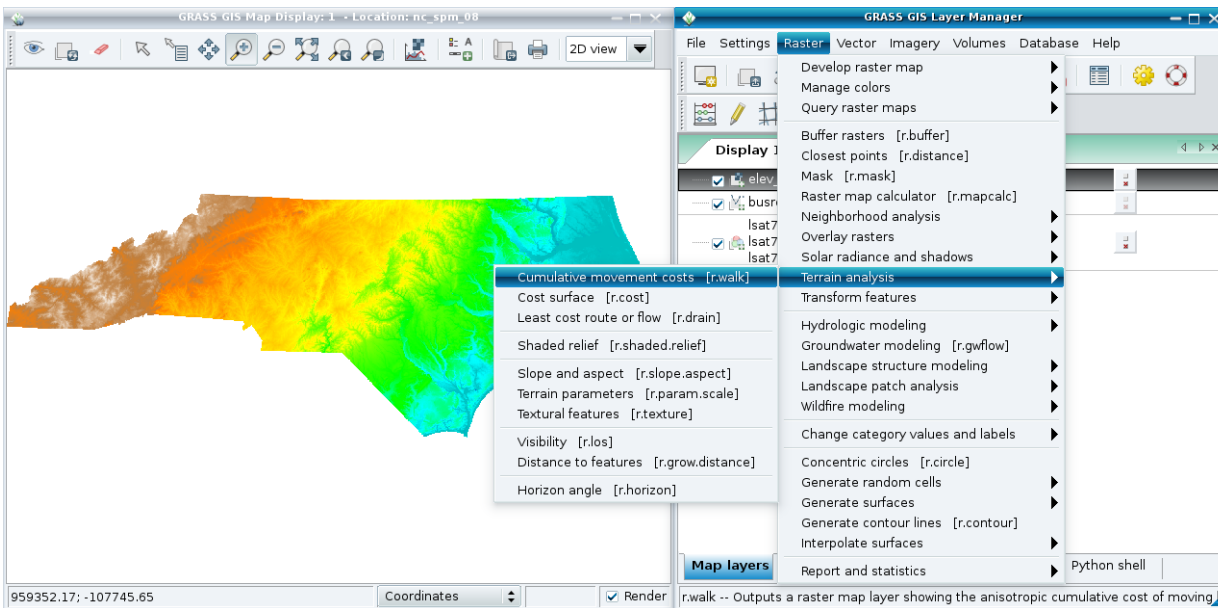
- r3.cross.rast**
Creates cross section 2D raster map from 3D raster map based on 2D elevation map
- r3.flow**
Computes 3D flow lines and 3D flow accumulation.
- r3.in.lidar**
Creates a 3D raster map from LAS LiDAR points
- r3.to.rast**
Converts 3D raster maps to 2D raster maps

[Main index](#) | [Topics index](#) | [Keywords index](#) | [Graphical index](#) | [Full index](#)

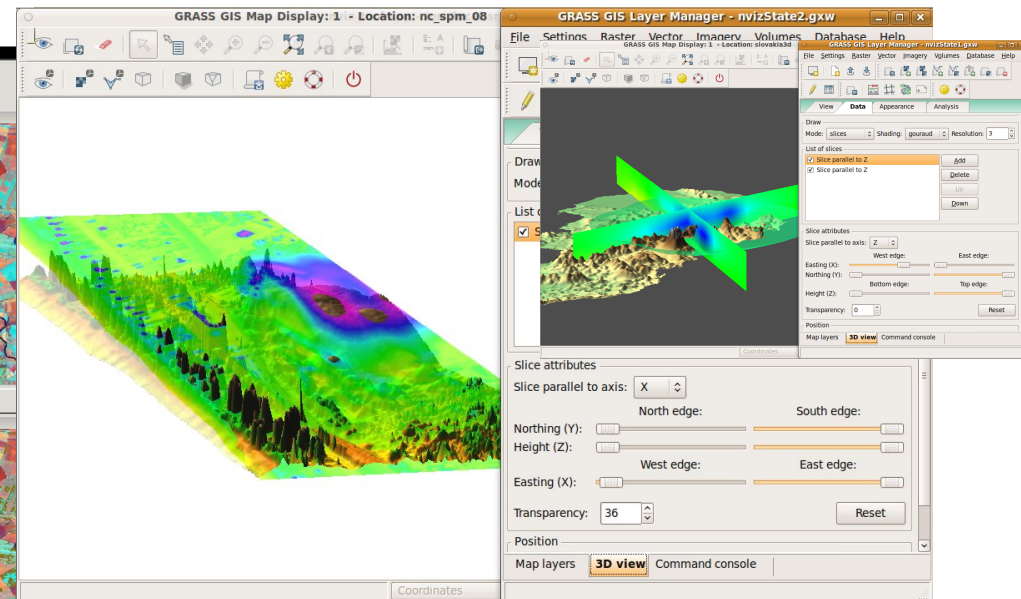
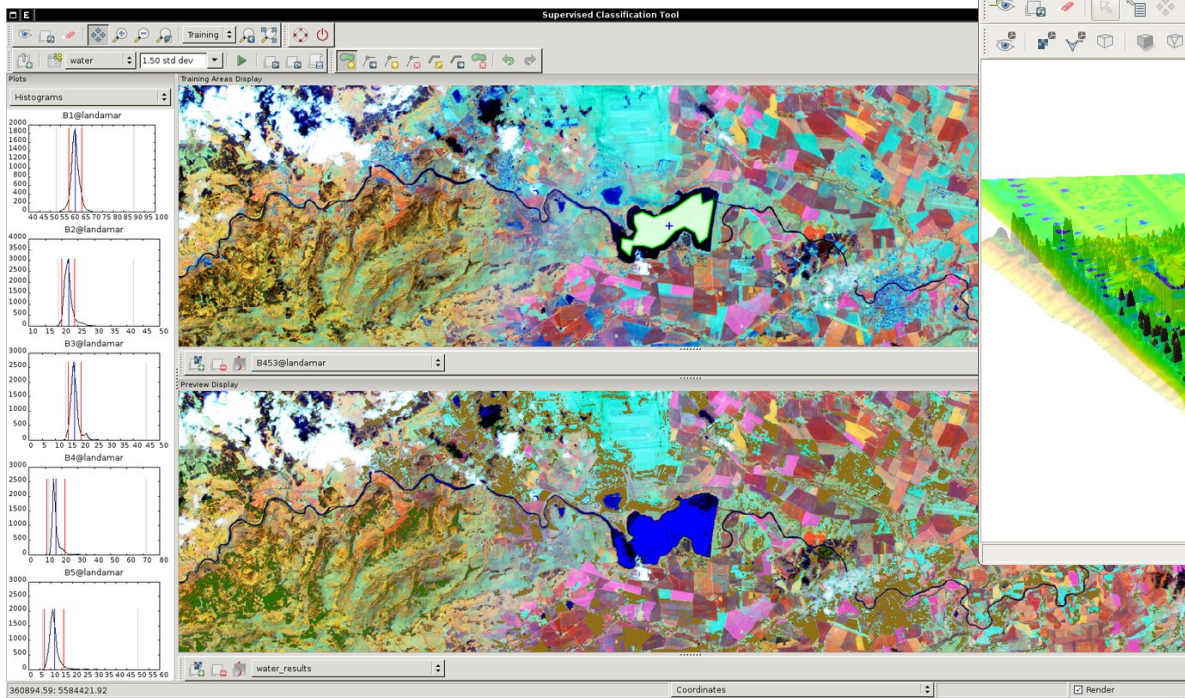
© 2003-2016 [GRASS Development Team](#), GRASS GIS 7.2.svn Reference Manual

https://grass.osgeo.org/grass72/manuals/graphical_index.html

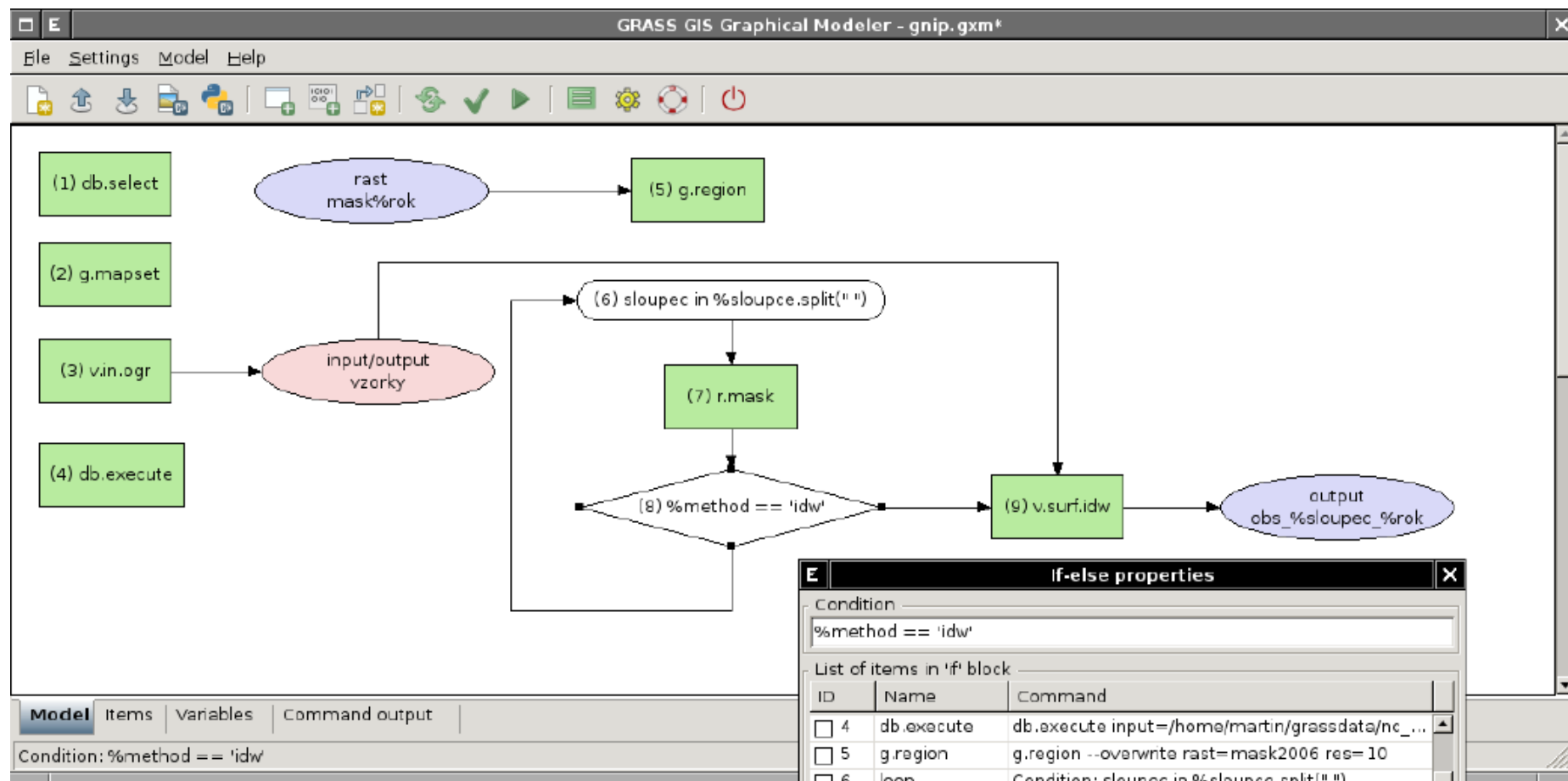
GRASS GIS 7 User interface



Nagshead LiDAR time series:
dune moving over 9 years
(NC, USA)



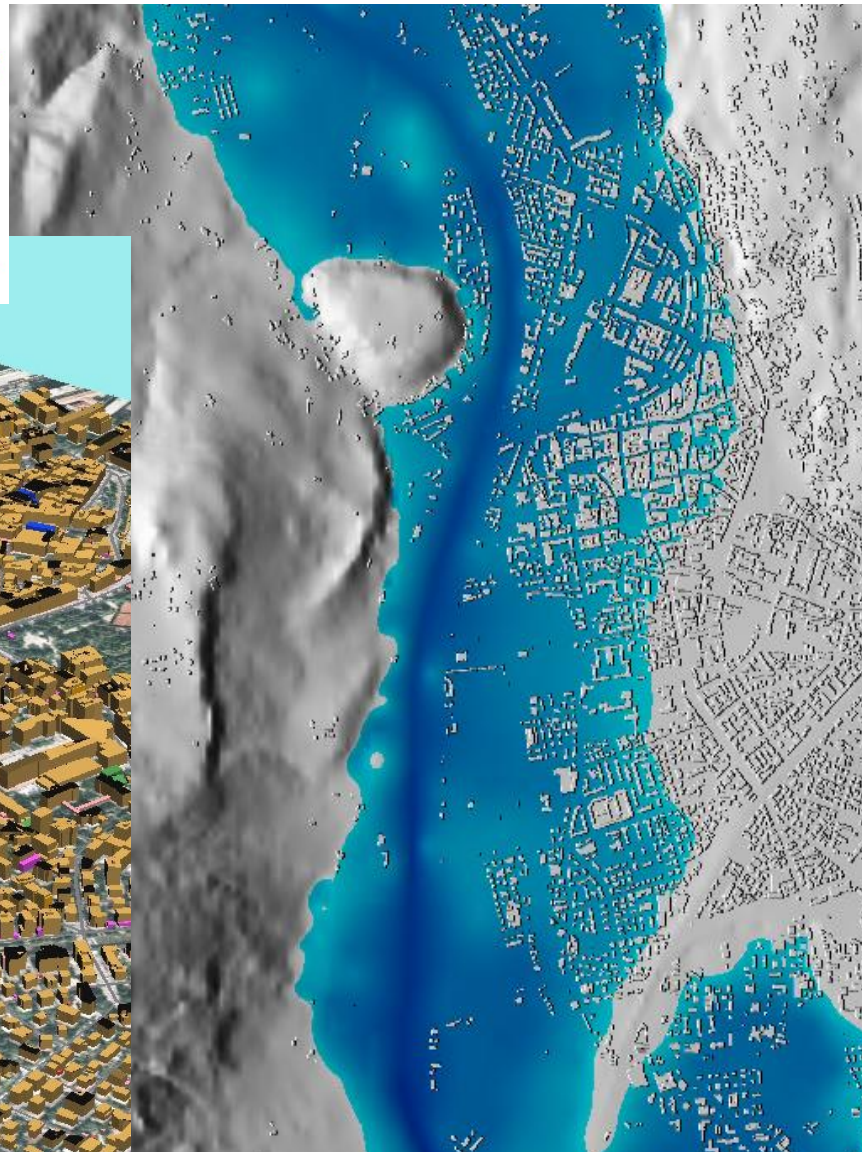
GRASS GIS 7: Geospatial Modeller



Extra bonus:
export to Python scripts

Raster and 3D vector

Elevation model combined with extruded 3D buildings; also true 3D vector supported



Trento, Italy

Optional: KML export for virtual globes

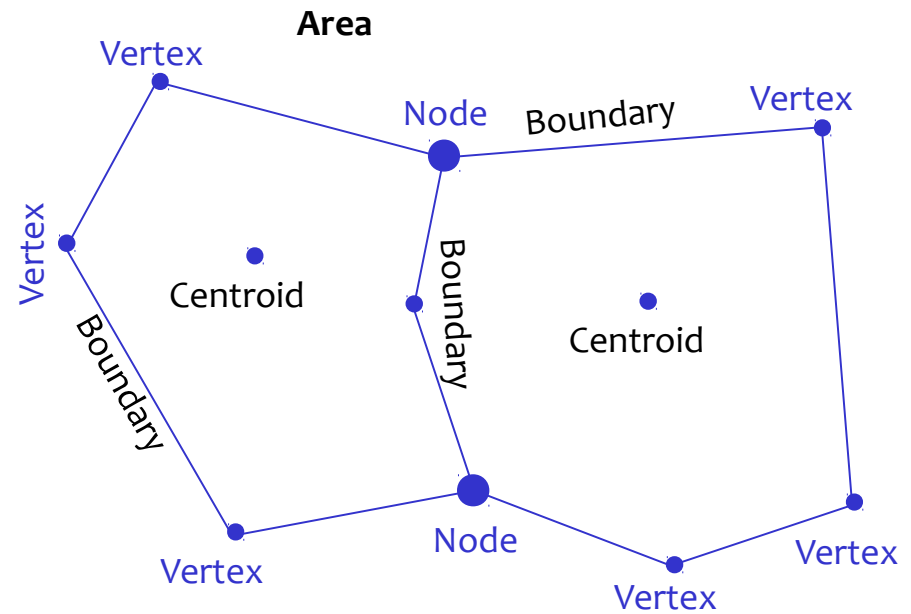
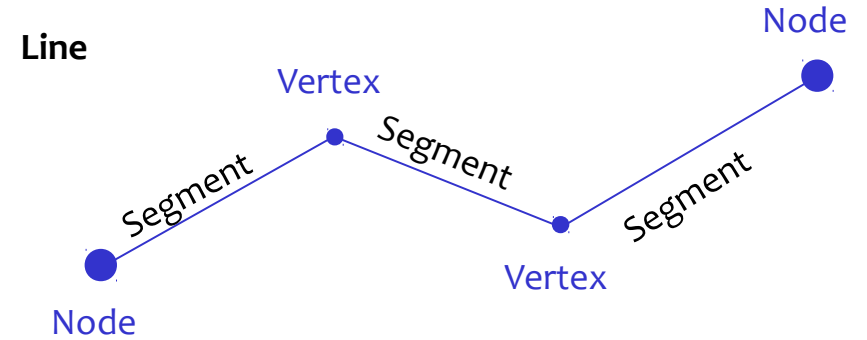
GRASS Topological 2D/3D Vector model

Vector geometry types

- Point
- Centroid
- Line
- Boundary
- Area (boundary + centroid)
- face (3D area)
- [kernel (3D centroid)]
- [volumes (faces + kernel)]

Geometry is **true** 3D when: x, y, z

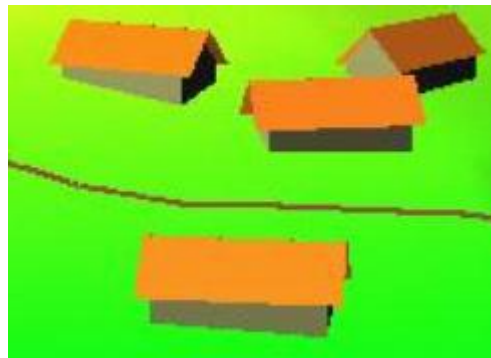
not in all GIS!



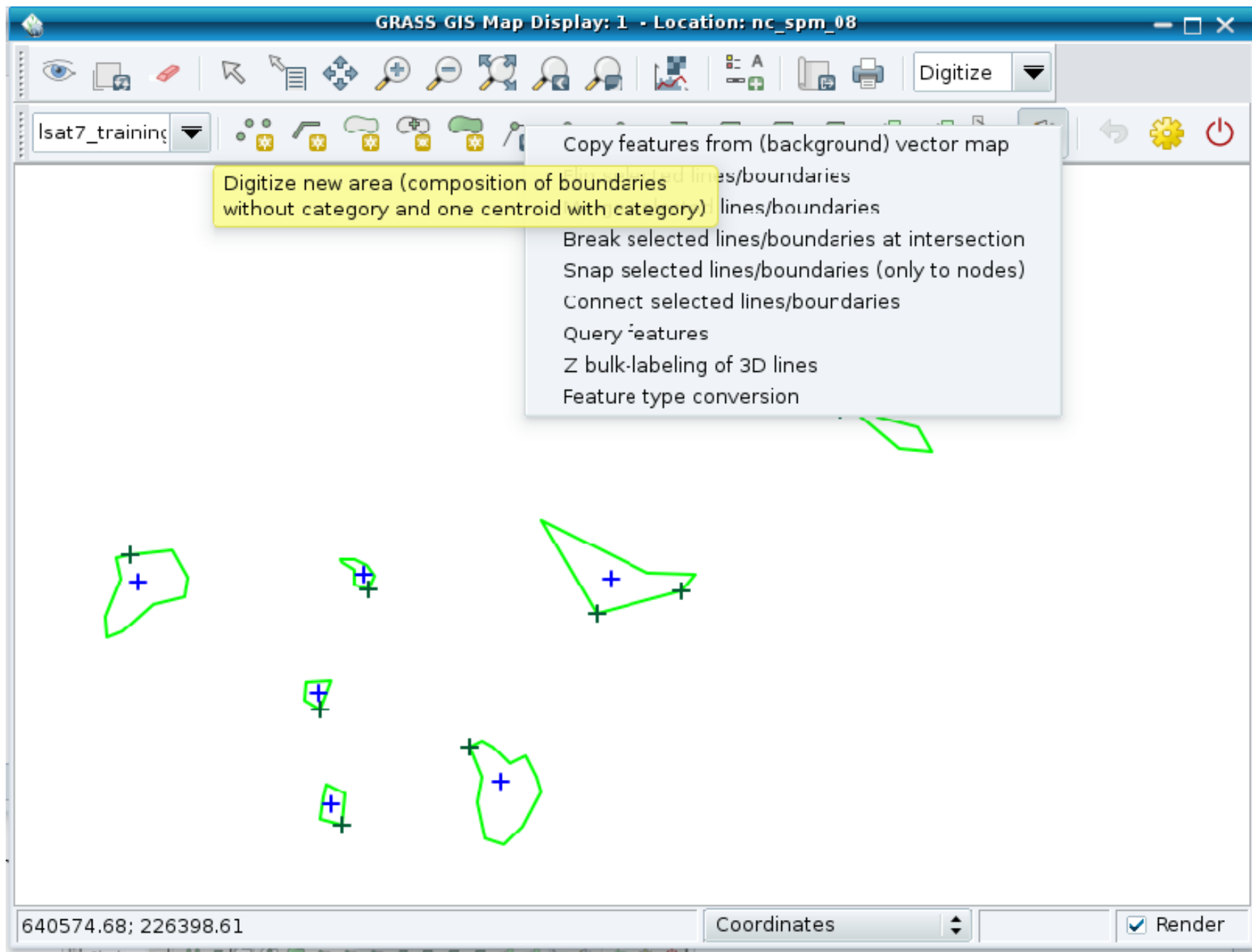
Use of Spatial Index



Faces



GRASS Topological Vector Digitizer



New Space-Time functionality in GRASS 7

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](#)

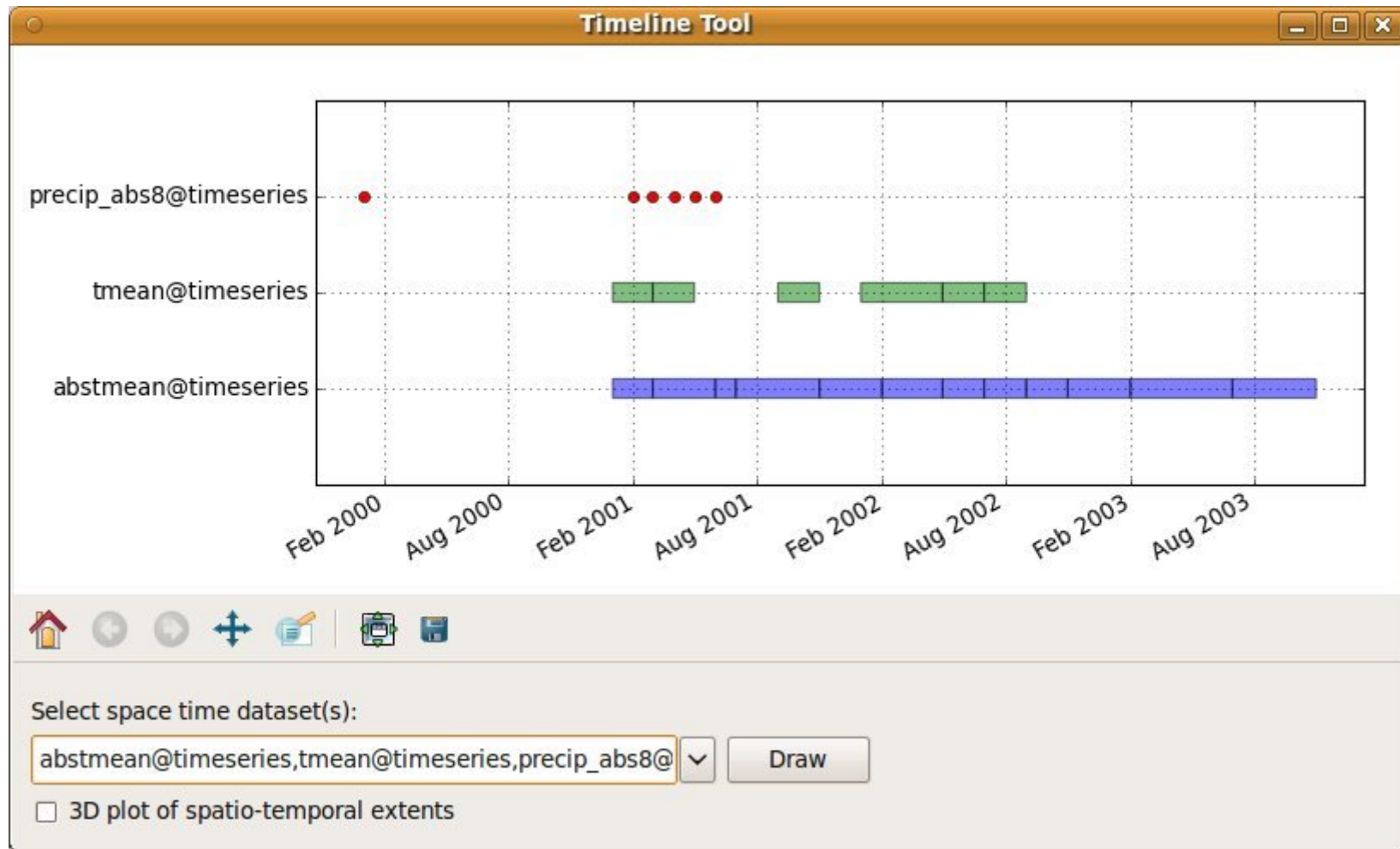
Aggregation

- [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 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>

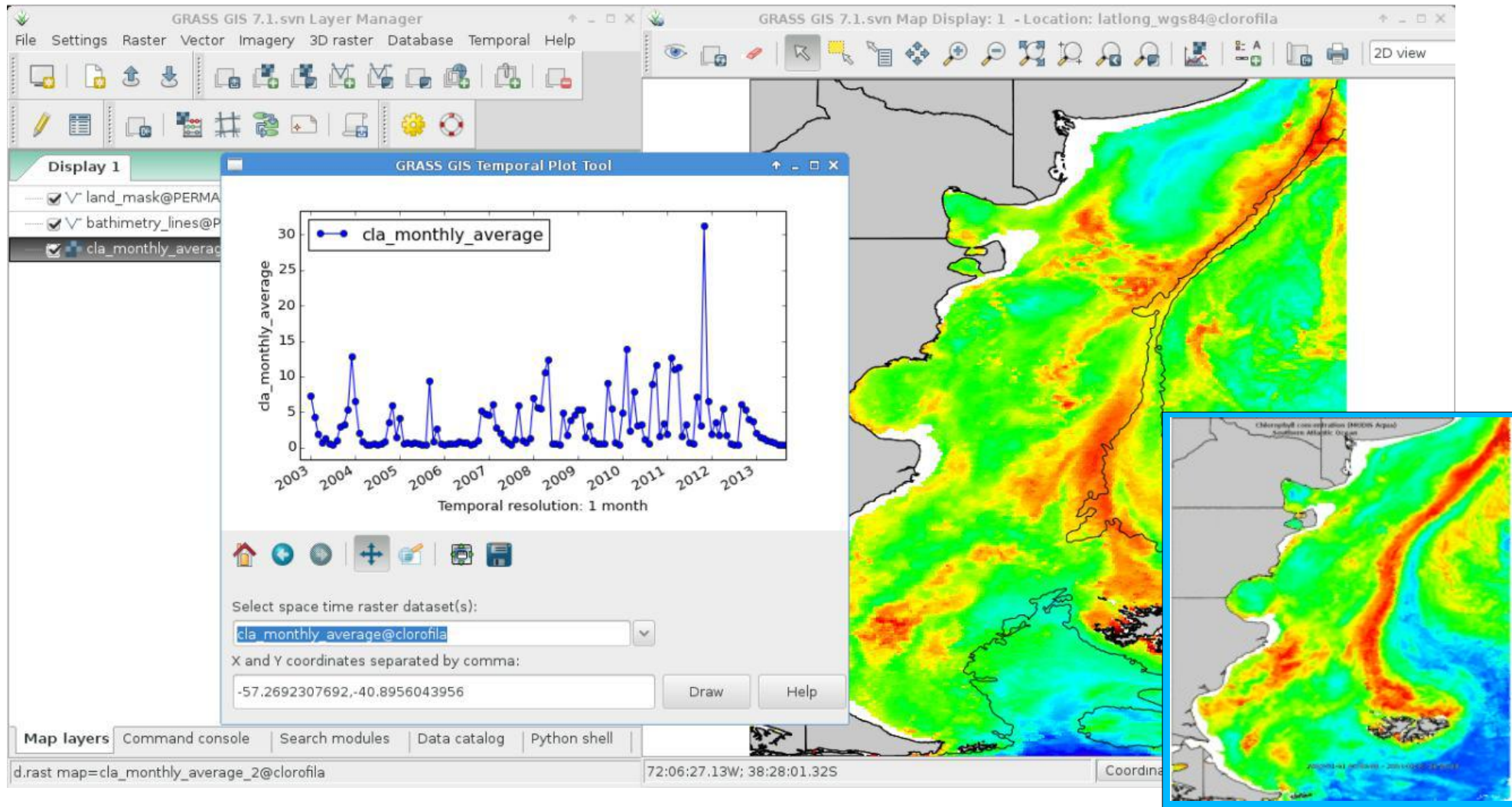
GRASS GIS 7: Space-time functionality



Screenshot: S Gebbert/A. Petrasova

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

GRASS GIS 7: Space-time functionality



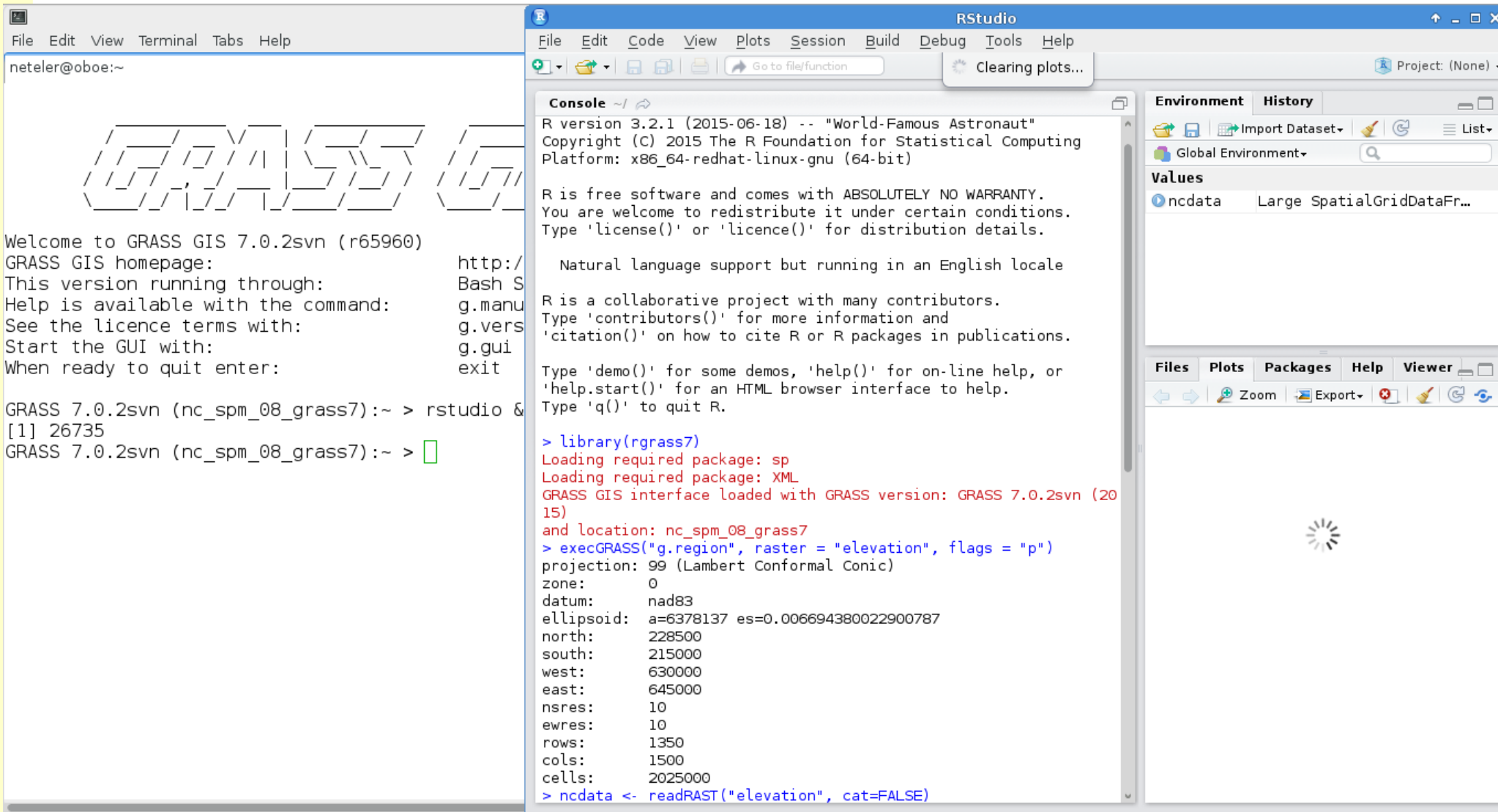
g.gui.tplot: plots the values of one or more temporal raster datasets for a queried point defined by a coordinate pair

(in PDF, click for [animation](#))

GRASS GIS 7 and R integration

There is a dedicated R packages “rgrass7” for GRASS GIS data exchange

https://grasswiki.osgeo.org/wiki/R_statistics/rgrass7



The image shows two side-by-side windows. The left window is a terminal window for GRASS GIS 7.0.2svn. It displays the GRASS logo and a welcome message. The user has entered the command `rstudio &` to launch RStudio. The right window is the RStudio interface. The console shows the R startup sequence, including the version (3.2.1) and platform (x86_64-redhat-linux-gnu). The user has loaded the `rgrass7` package, which reports the GRASS version (7.0.2svn) and location (`nc_spm_08_grass7`). The user has then executed `execGRASS("g.region", raster = "elevation", flags = "p")` to set the region, and `ncdata <- readRAST("elevation", cat=FALSE)` to read the elevation data into R.

```
neteler@ofoe:~  
  
Welcome to GRASS GIS 7.0.2svn (r65960)  
GRASS GIS homepage: http://grass.osgeo.org/  
This version running through: Bash Shell  
Help is available with the command: g.manual  
See the licence terms with: g.version  
Start the GUI with: g.gui  
When ready to quit enter: exit  
  
GRASS 7.0.2svn (nc_spm_08_grass7):~ > rstudio &  
[1] 26735  
GRASS 7.0.2svn (nc_spm_08_grass7):~ >   
  
R version 3.2.1 (2015-06-18) -- "World-Famous Astronaut"  
Copyright (C) 2015 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.  
  
> library(rgrass7)  
Loading required package: sp  
Loading required package: XML  
GRASS GIS interface loaded with GRASS version: GRASS 7.0.2svn (2015)  
and location: nc_spm_08_grass7  
> execGRASS("g.region", raster = "elevation", flags = "p")  
projection: 99 (Lambert Conformal Conic)  
zone: 0  
datum: nad83  
ellipsoid: a=6378137 es=0.006694380022900787  
north: 228500  
south: 215000  
west: 630000  
east: 645000  
nsres: 10  
ewres: 10  
rows: 1350  
cols: 1500  
cells: 2025000  
> ncdata <- readRAST("elevation", cat=FALSE)
```


Python API integration

http://grass.osgeo.org/wiki/GRASS_and_Python



Navigation

[Main Page](#)
[Community](#)
[Development](#)
[Documents](#)
[GRASS Help](#)
[Recent changes](#)
[Help](#)

Toolbox

[What links here](#)
[Related changes](#)
[Upload file](#)
[Special pages](#)
[Printable version](#)
[Permanent link](#)

Page **Discussion**

Read **Edit** **View history**

GRASS and Python

1 Python SIGs

2 Writing Python scripts

[2.1 Python script ed](#)

[2.2 Using the GRA](#)

[2.3 Creating Python](#)

[2.3.1 MS-Wind](#)

[2.3.2 Linux](#)

[2.4 Running extern](#)

[2.5 Testing and inst](#)

[2.5.1 Debuggin](#)

[2.5.2 Installatic](#)

3 Python extensions in C

[3.1 Python Scripting](#)

[3.2 Python Ctypes I](#)

[3.3 wxPython GUI c](#)

[3.4 Python-GRASS](#)

[3.5 Using GRASS g](#)

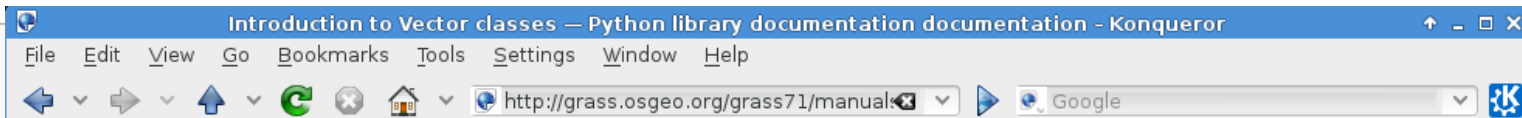
4 FAQ

5 Links

[5.1 General guides](#)

[5.2 Programming](#)

[5.3 Presentations](#)



[Python library documentation documentation](#) » [PyGRASS documentation](#) »

[previous](#) | [next](#) | [modules](#) | [index](#)

Introduction to Vector classes

Details about the architecture can be found in the [GRASS GIS 7 Programmer's Manual: GRASS Vector Library](#)

Instantiation and basic interaction.

```
>>> from pygrass.vector import VectTopo
>>> municip = VectTopo('boundary_municip_sqlite')
>>> municip.is_open()
False
>>> municip.mapset
''
>>> municip.exist() # check if exist, and if True set mapset
True
>>> municip.mapset
'user1'
```

Open the map with topology:

```
>>> municip.open()
```

get the number of primitive:

Previous topic

[Introduction to Raster classes](#)

Next topic

[Interface to GRASS GIS modules](#)

Quick search

Enter search terms or a module, class or function name.

Using Python and GRASS GIS 7 with ipython

An interactive (Web based!) shortcourse on writing GRASS scripts in Python

<https://github.com/wenzeslaus/python-grass-addon>

https://github.com/wenzeslaus/python-grass-addon/blob/master/01_scripting_library.ipynb

Introduction to the GRASS GIS 7 Python Scripting Library

The [GRASS GIS 7](#) Python Scripting Library provides functions to call GRASS modules within scripts as subprocesses. The most often used functions include:

- **run_command**: most often used with modules which output raster/vector data where text output is not expected
- **read_command**: used when we are interested in text output
- **parse_command**: used with modules producing text output as key=value pair
- **write_command**: for modules expecting text input from either standard input or file

Besides, this library provides several wrapper functions for often called modules.

Calling GRASS GIS modules

We start by importing GRASS GIS Python Scripting Library:

```
In [ ]: import grass.script as gscript
```

Before running any GRASS raster modules, you need to set the computational region using [g.region](#). In this example, we set the computational extent and resolution to the raster layer `elevation`.

```
In [ ]: gscript.run_command('g.region', raster='elevation')
```

The `run_command()` function is the most commonly used one. Here, we apply the focal operation *average* ([r.neighbors](#)) to smooth the elevation raster layer. Note that the syntax is similar to bash syntax, just the flags are specified in a parameter.

GRASS Addons: User contributed extensions

The Addons repository is SVN based:

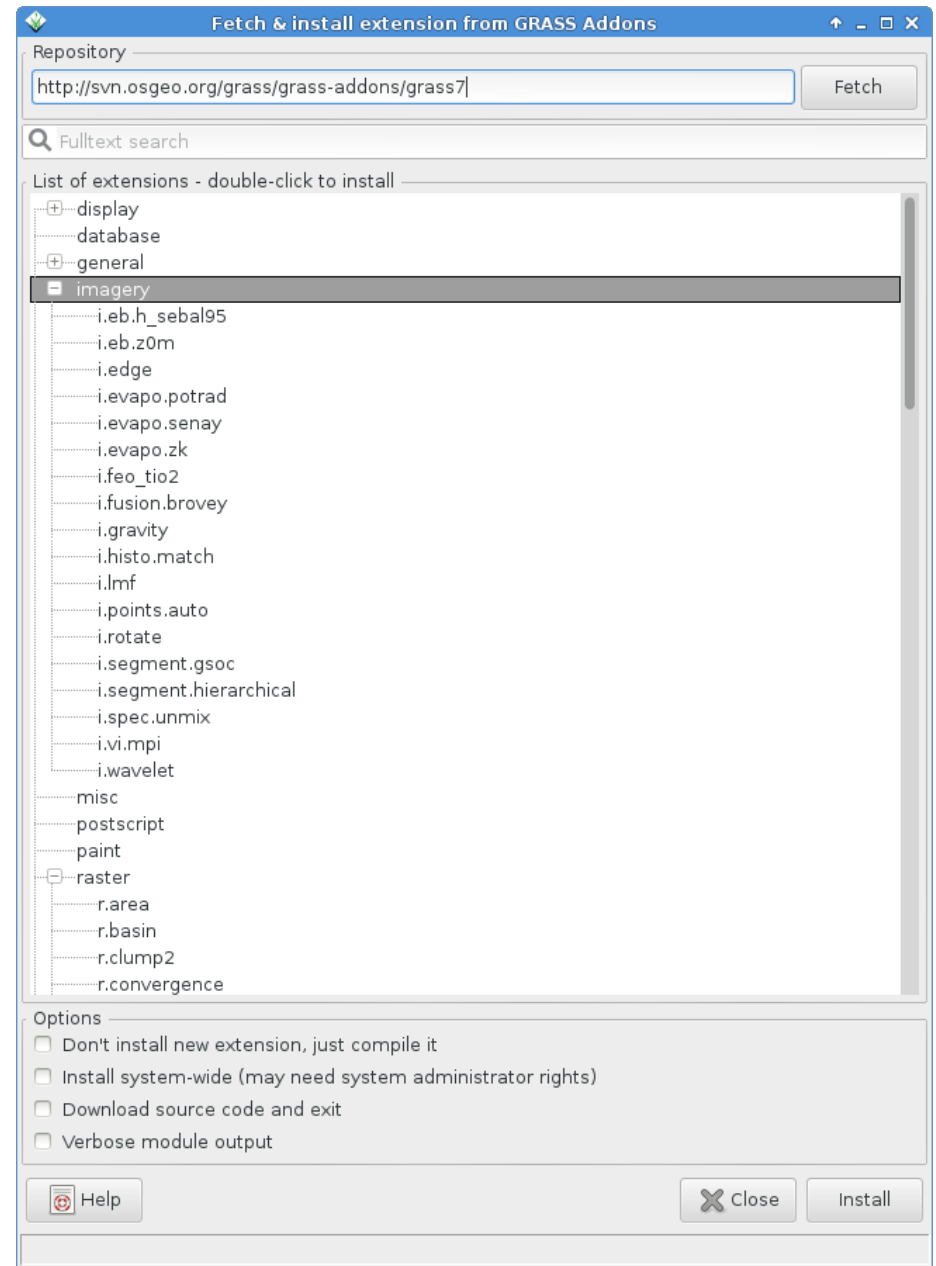
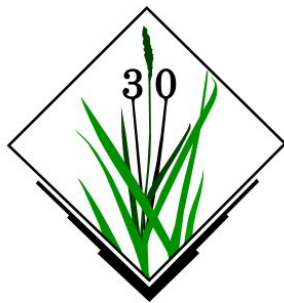
One-click installation with extension manager

Increasing inflow of Python scripts

Users can easily obtain **write access** to develop new functionality

Peer review through SVN commit email list

Also github, gitlab etc. now supported



<https://grass.osgeo.org/grass7/manuals/addons/>

Where's the stuff?



GRASS GIS 7 Software:

Free download for MS Windows, MacOSX, Linux and source code:

<https://grass.osgeo.org/download/>

Addons (user contributed extensions):

<https://grass.osgeo.org/grass7/manuals/addons/>

Free sample data:

Rich data set of North Carolina (NC)

... available as GRASS GIS location and in common GIS formats

<https://grass.osgeo.org/download/sample-data/>

User Help:

Mailing lists (also in different languages):

<https://grass.osgeo.org/support/>

Wiki:

<https://grasswiki.osgeo.org/wiki/>

https://grasswiki.osgeo.org/wiki/R_statistics/rgrass7

Manuals:

<https://grass.osgeo.org/documentation/manuals/>