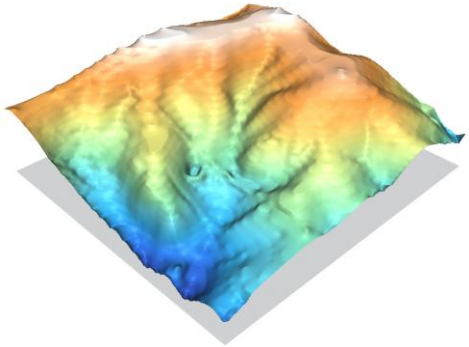
GRASS GIS 8.x
Crash course

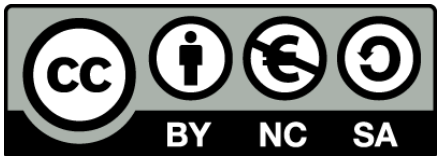


Giorgio Agugiaro

Last update: 15 July 2025

License

This presentation is licensed under the Creative Commons License CC BY-NC-SA 4.0. According to CC BY-NC-SA 4.0 permission is granted to share this document, i.e. copy and redistribute the material in any medium or format, and to adapt it, i.e. remix, transform, and build upon the material under the following conditions:



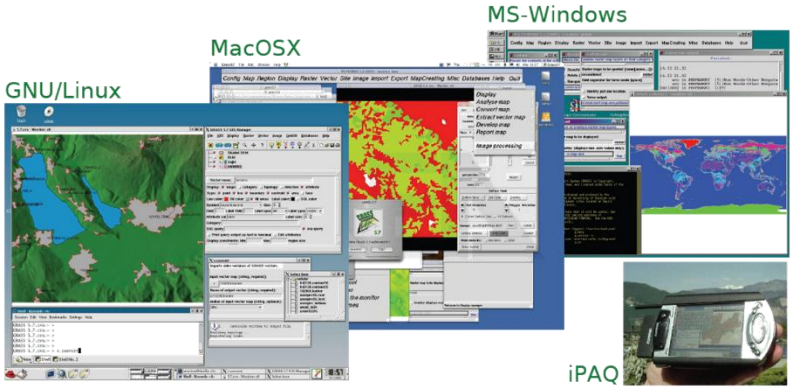
- **Attribution:** You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
- **NonCommercial:** You may not use the material for commercial purposes.
- **ShareAlike:** If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
- **No additional restrictions:** You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Brief overview of GRASS GIS

Some history

- Developed since 1982, first version as GRASS in 1985
- Licensed under GPL since 1999
- Written in C, but portable (and already ported) to many platforms
- Homepage: https://grass.osgeo.org
- Short movie about GRASS GIS (1987) narrated by William Shatner (Capt. Kirk!) https://www.youtube.com/watch?v=U3Hf0qI4JLc (14:27)

GRASS GIS



GRASS GIS modules

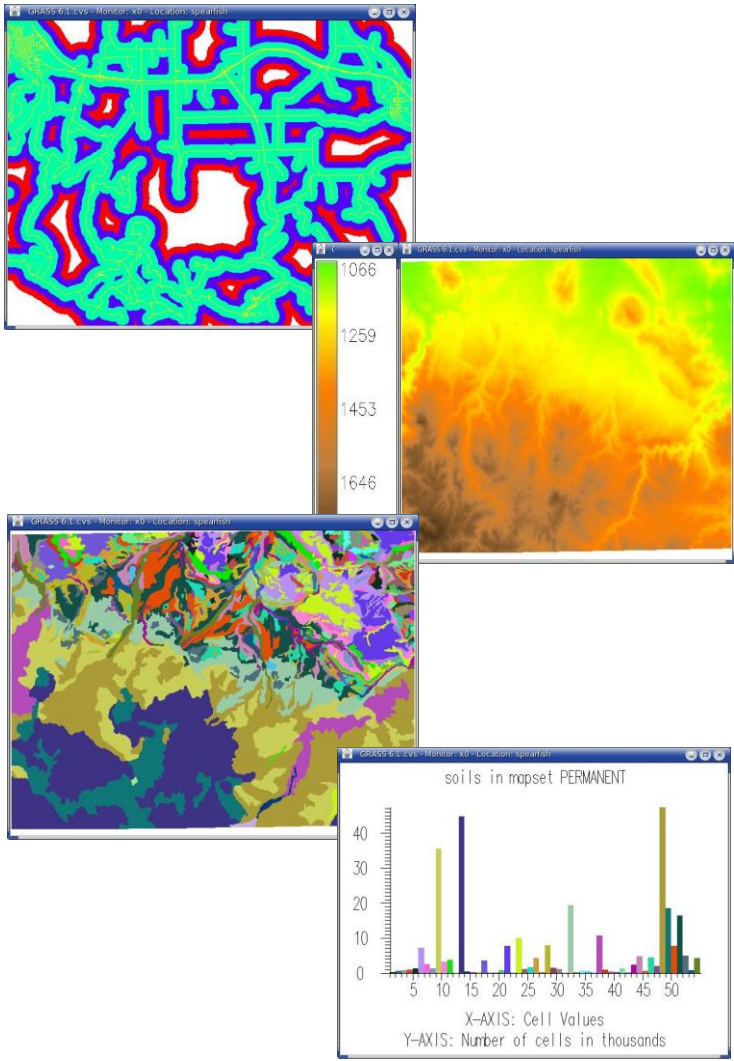
- GRASS GIS commands (or **modules**) are generally thematically grouped according to prefixes in the command name
- They can be used either from the command line, or the GRASS GUI, or from QGIS

GRASS Command Overview			
prefix	*function class*	*type of command*	*example*
d.*	display	graphical output	d.rast: views raster map d.vect: views vector map
db.*	database	database management	db.select: selects value(s) from table
g.*	general	general file operations	g.rename: renames map
i.*	imagery	image processing	i.smap: image classifier
ps.*	postscript	map creation in Postscript format	ps.map: map creation
r.*	raster	raster data processing	r.buffer: buffer around raster features r.mapcalc: map algebra
r3.*	voxel	raster voxel data processing	r3.mapcalc: volume map algebra
v.*	vector	vector data processing	v.overlay: vector map intersections

GRASS GIS modules

Typical **raster** operations

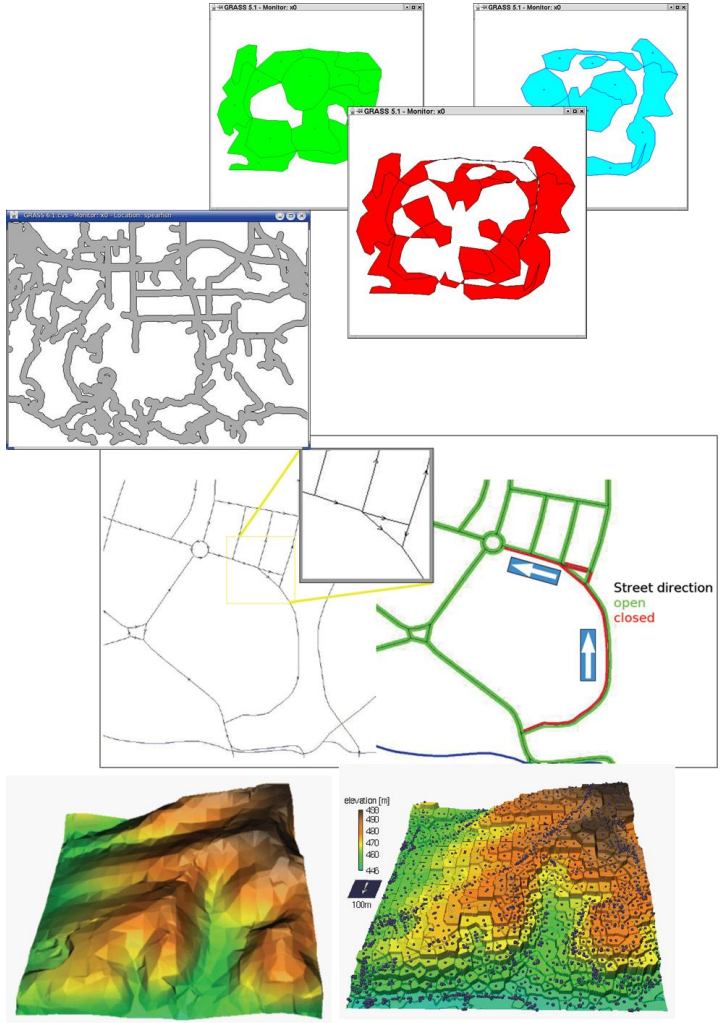
- Vectorization
  - r.to.vect
  - r.contour (creates contour lines)
- Digitalization
  - r.digit
- Creation of buffers
  - r.buffer
- Classification
  - r.reclass, r.recode
- Map algebra
  - r.mapcalc, r.stats, r.neighbors, …
- DEM creation and analysis
  - interpolation (IDW, spline, …)
  - slope, aspect, shaded relief maps, etc.
- Surface cost analysis
  - r.cost, r.drain
- Specific analyses (astronomy, hydrology, …)
  - r.sun, r.watershed, …



GRASS GIS modules

Typical **vector** operations

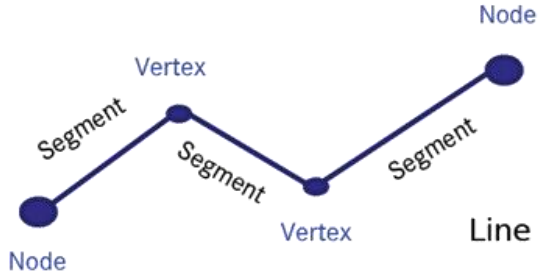
- Rasterization
  - v.to.rast, …
- Digitalization (with input of attributes)
  - v.digit
- Creation of buffers
  - v.buffer
- Classification and attribute management
  - v.reclass, v.category
- Overlay
  - v.overlay, v.patch
- Network analysis
  - v.net.xxx (several modules)
- Interpolation
  - v.voronoi, v.delaunay



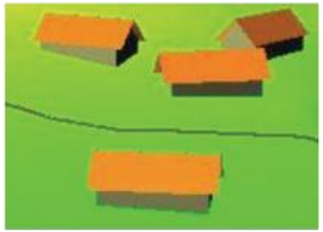
GRASS GIS vector data

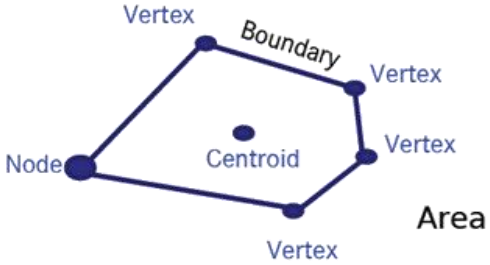
- Point – 3Dpoint
- Centroid
- Line
- Boundary
- Area (boundary + centroid)
- Face (3D area)
- Kernel (3D centroid)
- Volumes (faces + kernel)

NOTA BENE: Unlike shapefiles, in GRASS different vector entities can co-exist in the same layer!

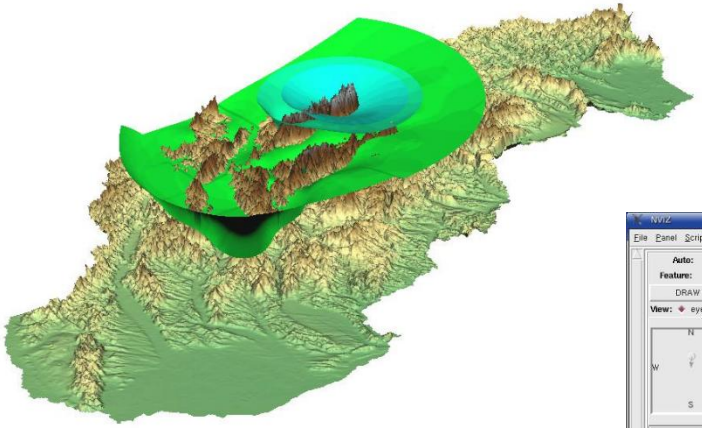


Faces

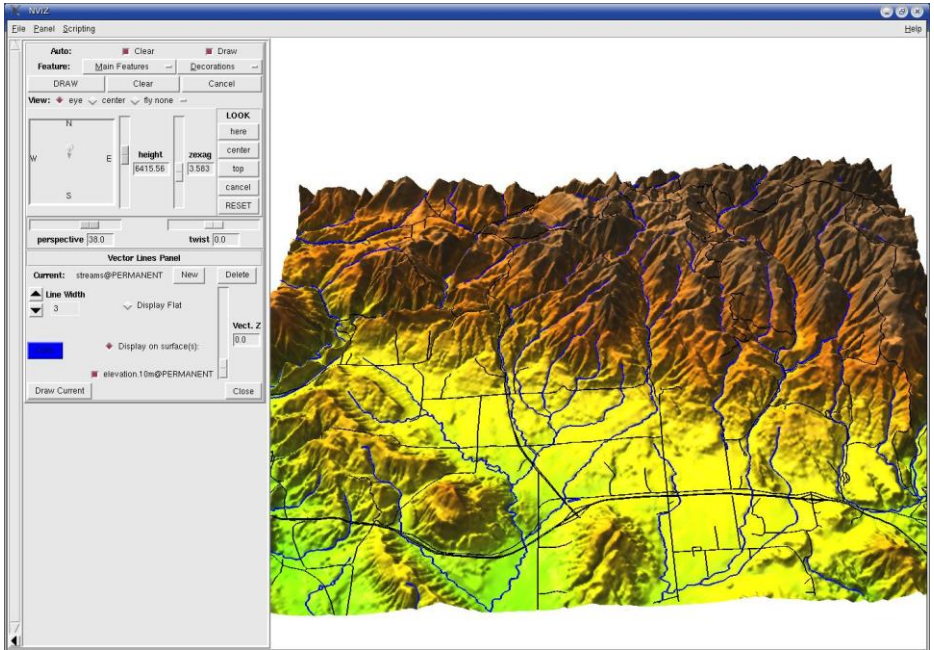




NVIZ



NVIZ is a 3D visualization module within GRASS GIS for 2D/3D vector, raster and voxel data



GRASS GIS “database” structure

In GRASS GIS, all datasets are stored in a **single directory** called the "**GRASS GIS Database**" and they are further divided into LOCATION(s) and MAPSET(s).

- A **LOCATION** is a directory containing all data having the same projection and – possibly – region (see later)

- A **MAPSET** is a subdirectory of LOCATION containing thematically similar data. Mapset PERMANENT is always present and read-only!
  - One LOCATION can have multiple MAPSETs.
  - Each MAPSET contain files and directories of maps in vector or raster format (2D and/or 3D)

- The **REGION** defines the size and the resolution (cell size) of the work area
  - First defined upon creation of the LOCATION, but can be changed later
  - Each MAPSET can have its own region

GRASS GIS “database” structure

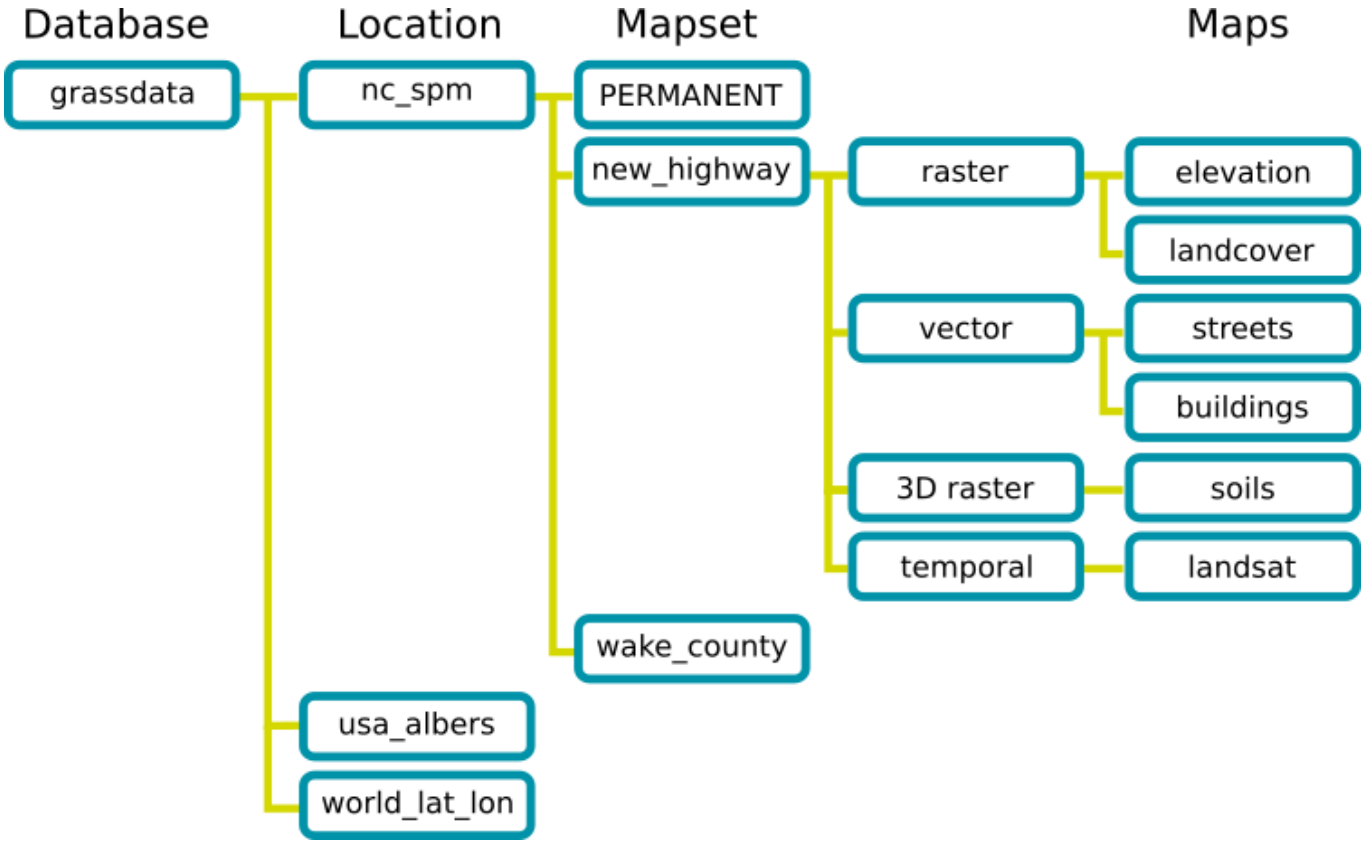
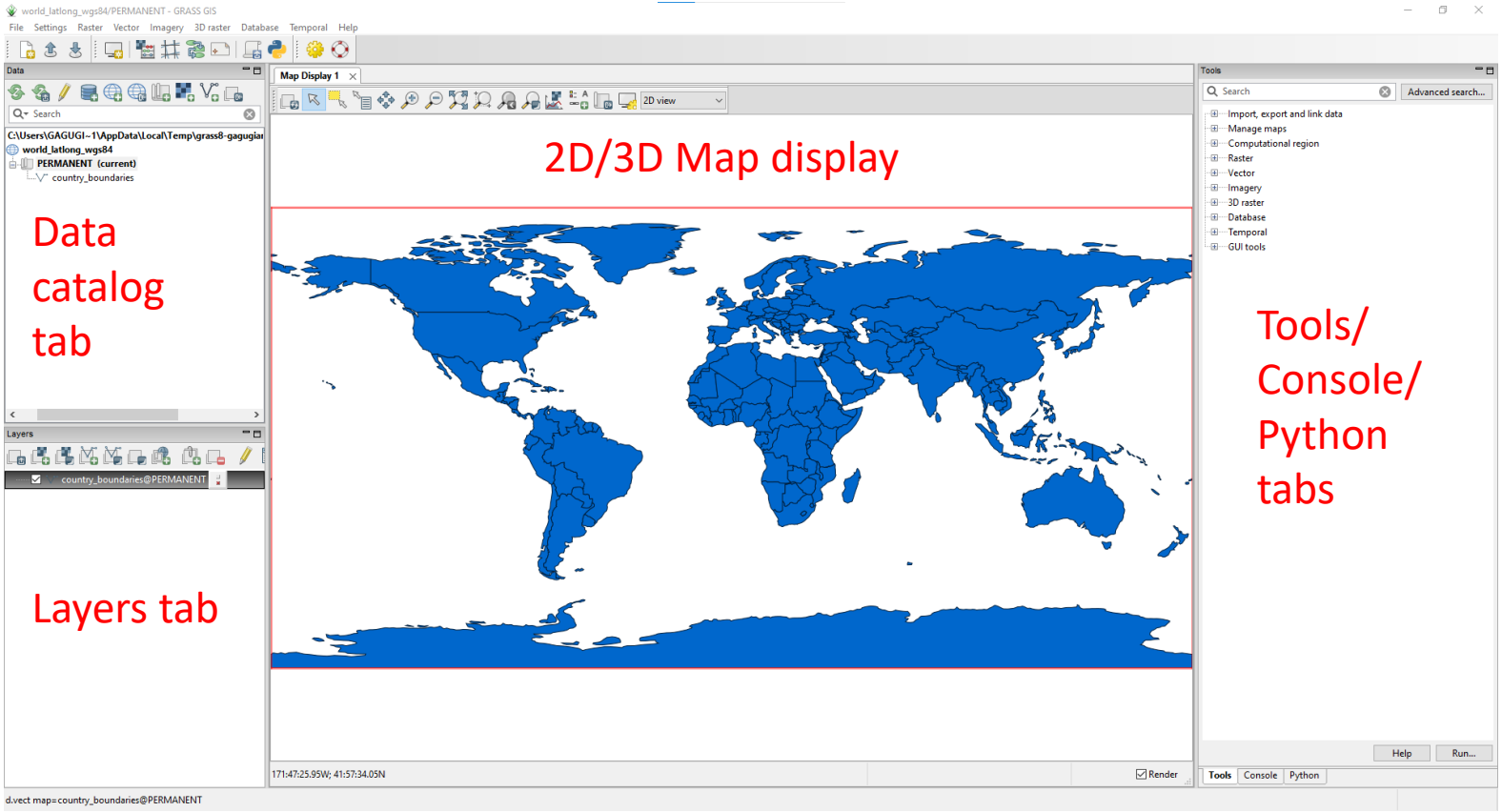


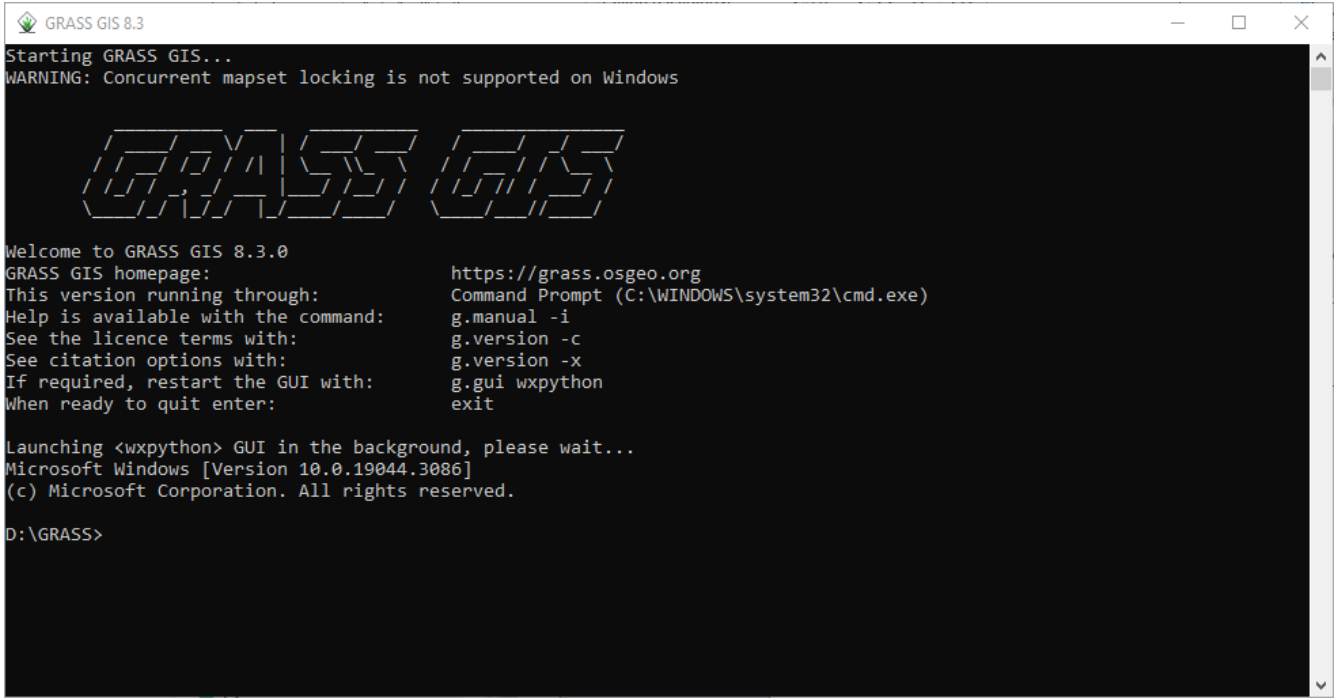
Image source: https://grass.osgeo.org/grass83/manuals/grass_database.png

Running GRASS GIS - GUI



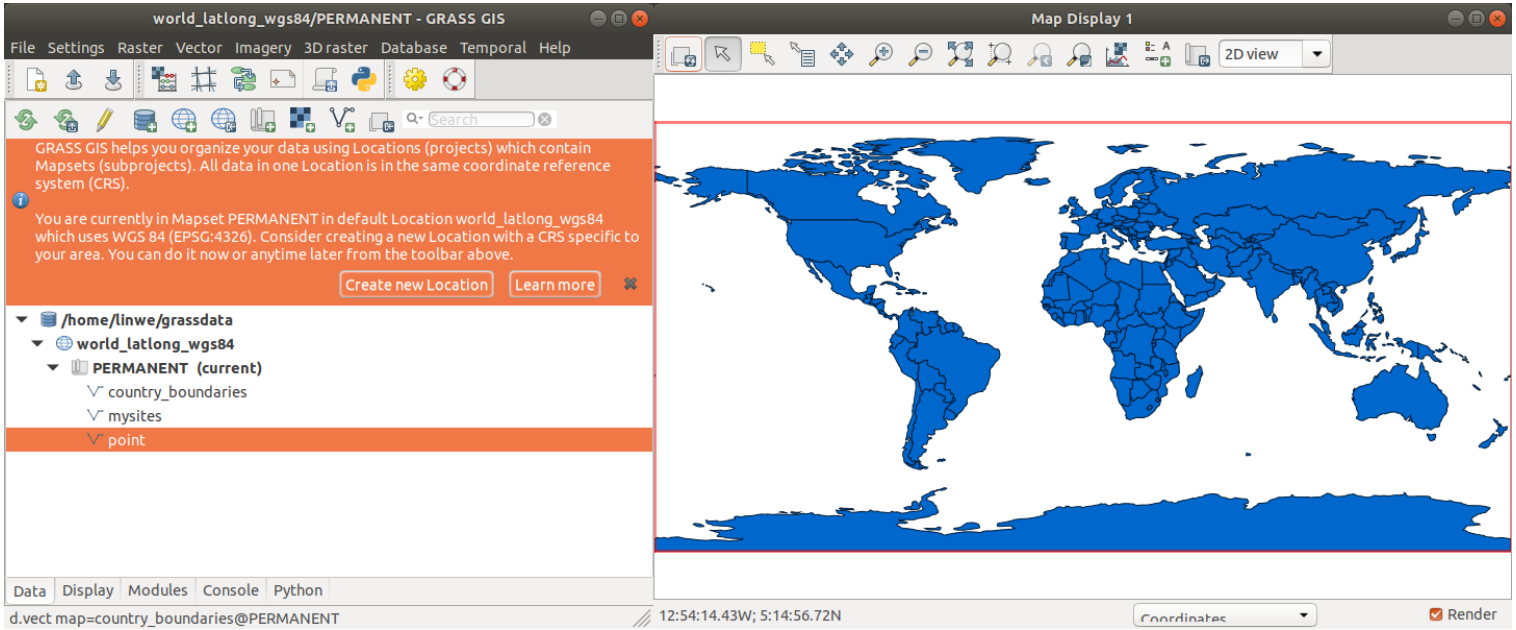
Running GRASS GIS – Command prompt

- In the background, minimized, there is aways a window with the command prompt.
- It can be used to type GRASS commands directly, too



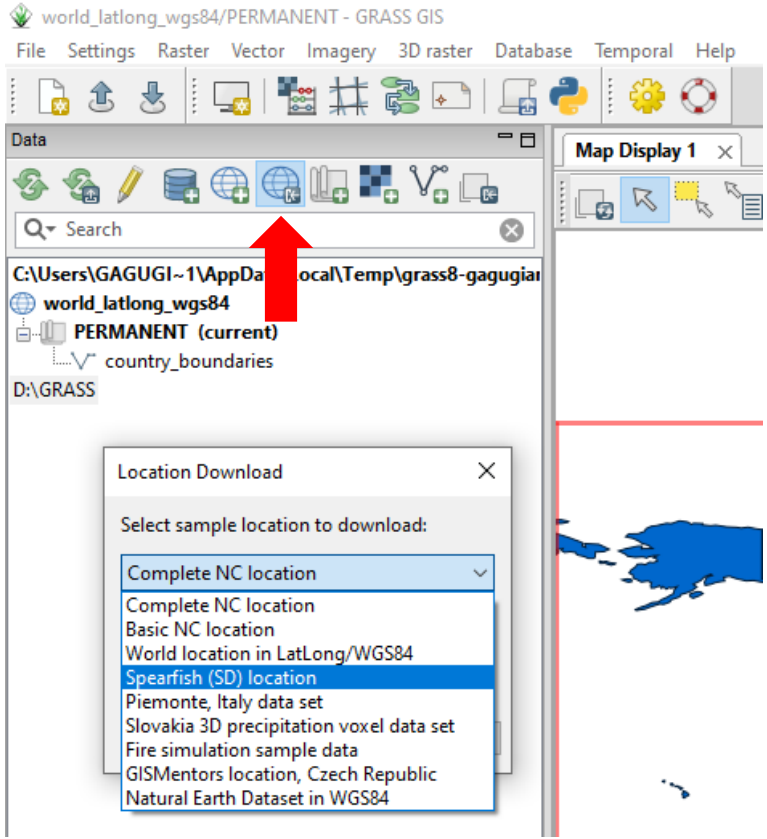
GRASS GIS

- When launching GRASS GIS for the first time, it will load the default Location "world_latlog_wgs84" where you can find a map layer called "country_boundaries" showing a world map in the WGS84 coordinate system
- If you know the CRS of your data or study area, you can fill **EPSG code** or description, and Location Wizard finds appropriate CRS from a predefined list of projections.



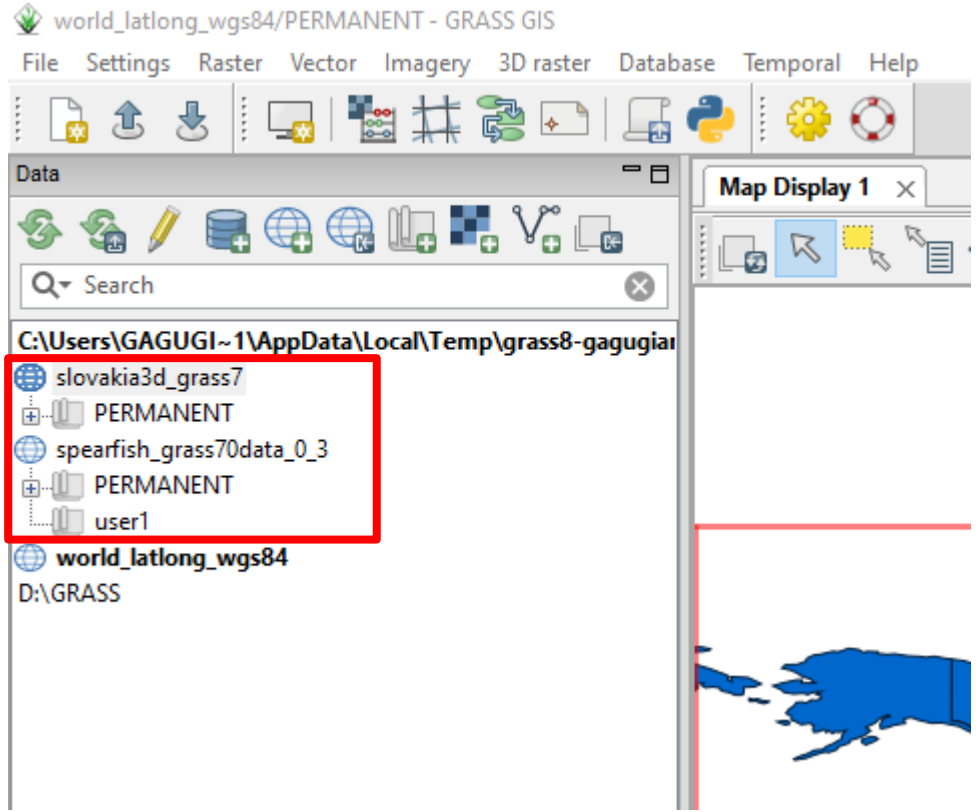
Adding sample datasets

- You can easily install sample datasets
- Click on the "Download sample location to current GRASS database" button and select the desired sample datasets. Here, for example, we choose Spearfish and Slovakia3D precipitation voxel data set



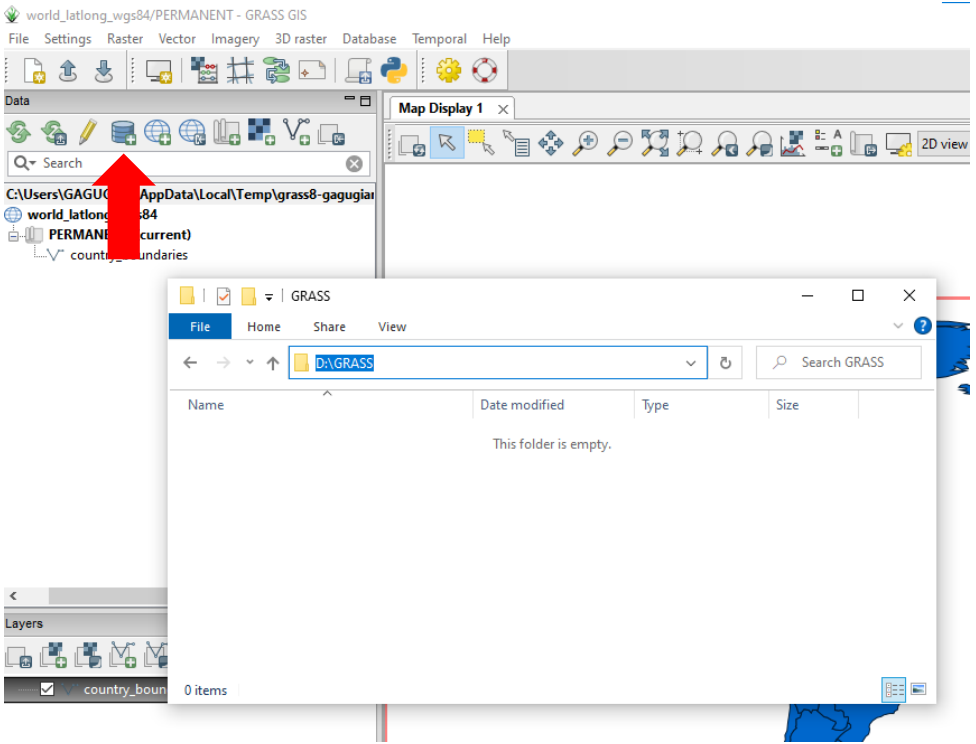
Adding sample datasets

- You can easily install sample datasets
- Click on the "Download sample location to current GRASS database" button and select the desired sample datasets. Here, for example, we choose Spearfish and Slovakia3D precipitation voxel data set
- Once installed, you will see a new locations called **slovakia3D_grass7** and **spearsish_grass70data_0_3**



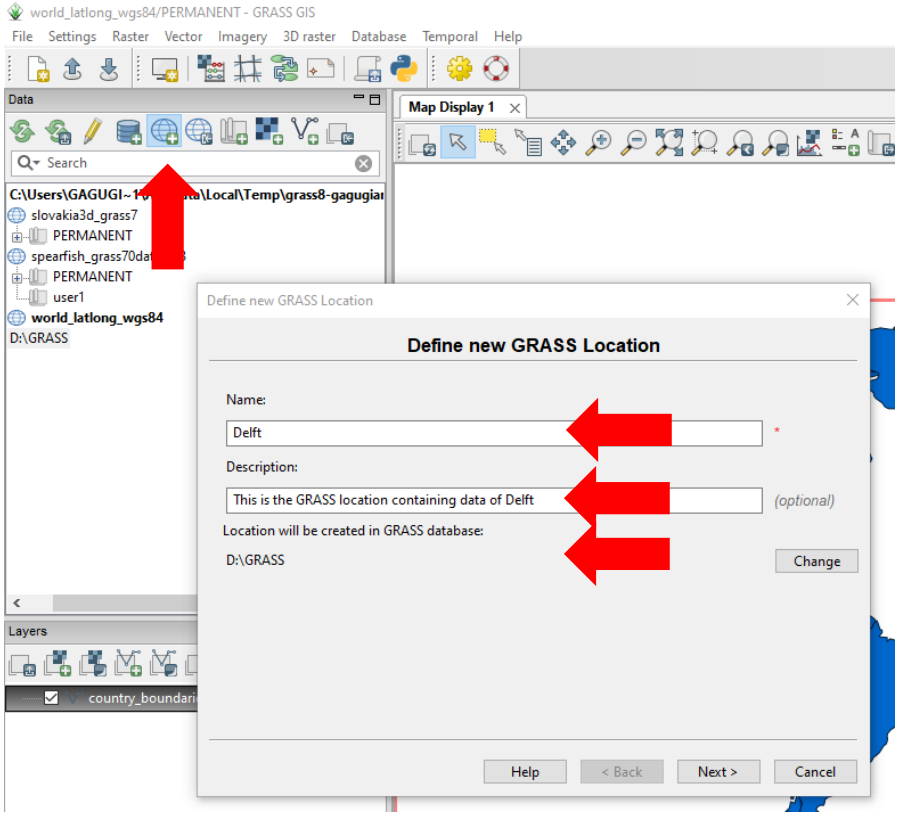
Defining a customized GRASS data folder

- It is a good idea to set up your own folder where to store your GRASS data.
- Click on the "Add existing or create new database" and choose a directory (here, for example, D:\GRASS)
- You will be immediately requested if you want to add a location
- If you have the necessary information, you can proceed
- Alternatively, you can also copy/paste an already existing location



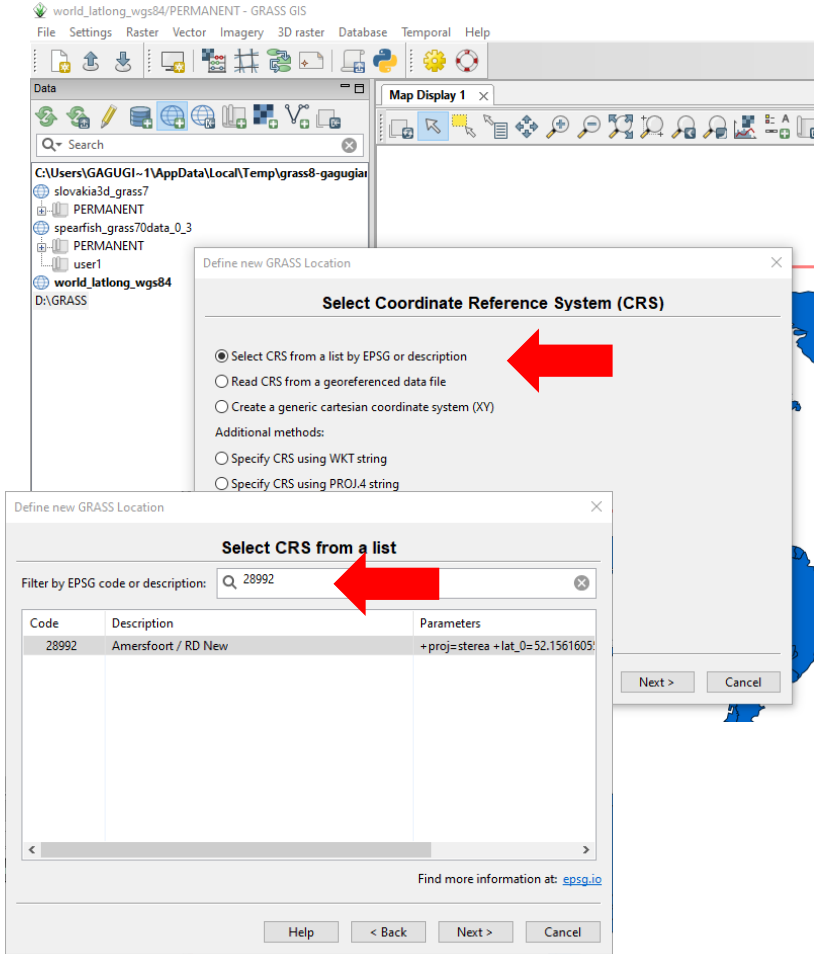
Adding a new GRASS location

- Click on the "Create new location in current GRASS database
- Choose a name (e.g. Delft)
- Optionally add a description
- Optionally select which GRASS database to use (e.g. The previously set D:\GRASS)



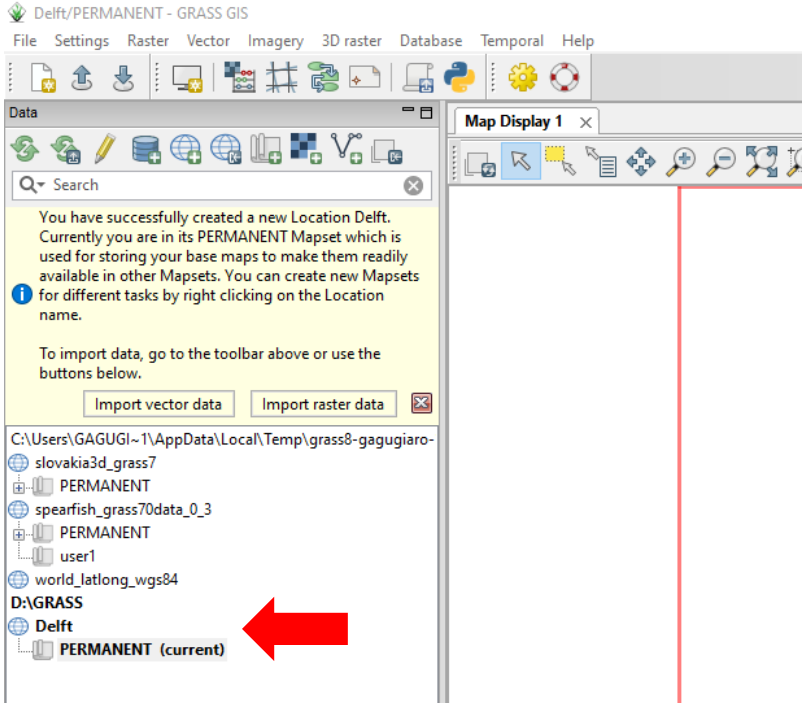
Adding a new GRASS location

- You will be asked to provide the needed parameters
- The easiest way is to choose the first optio and to select the CRS from a list of EPSG codes
- You can type the code or a string and you can then pick the desired CRS. Here, for example 28992 for the NL
- Accepts the standard settings and...



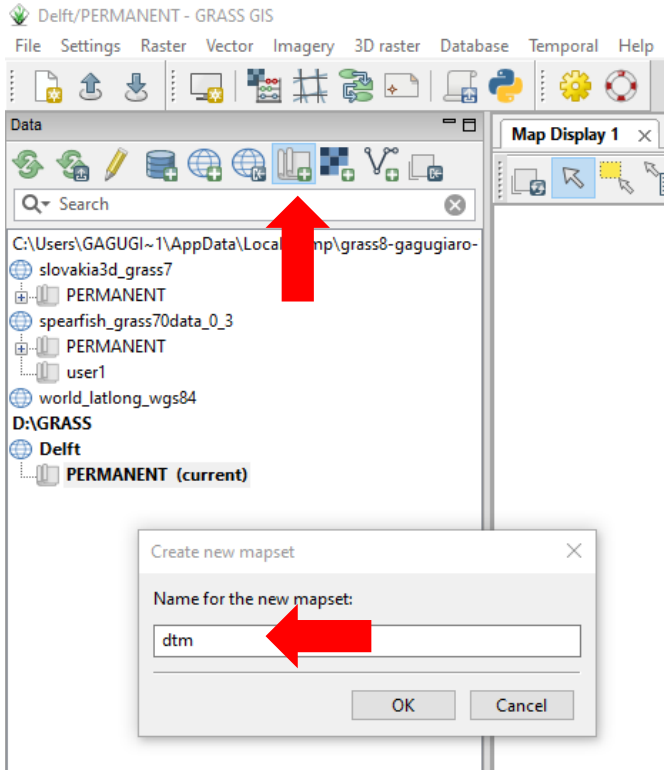
Adding a new GRASS location

- You will be asked to provide the needed parameters
- The easiest way is to choose the first optio and to select the CRS from a list of EPSG codes
- You can type the code or a string and you can then pick the desired CRS. Here, for example 28992 for the NL
- Accepts the standard settings and...
- ...the new location, with the defalt PERMANENT mapset will be created
- You are now offered the option to directly import new data (to the PERMANENT mapset), or you can alternatively create a your own mapset



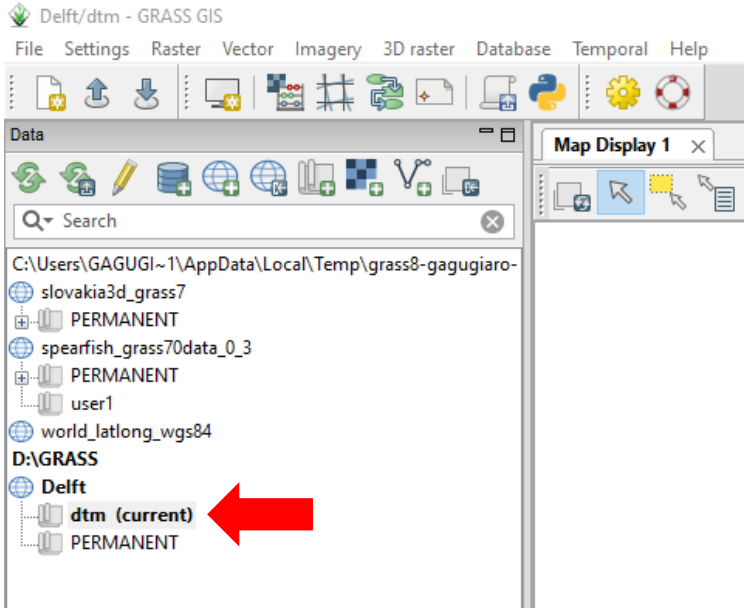
Adding a new GRASS mapset

- Click on the "Create new mapset in current location" button and set the name of the new mapset. Here, for example, "dtm"



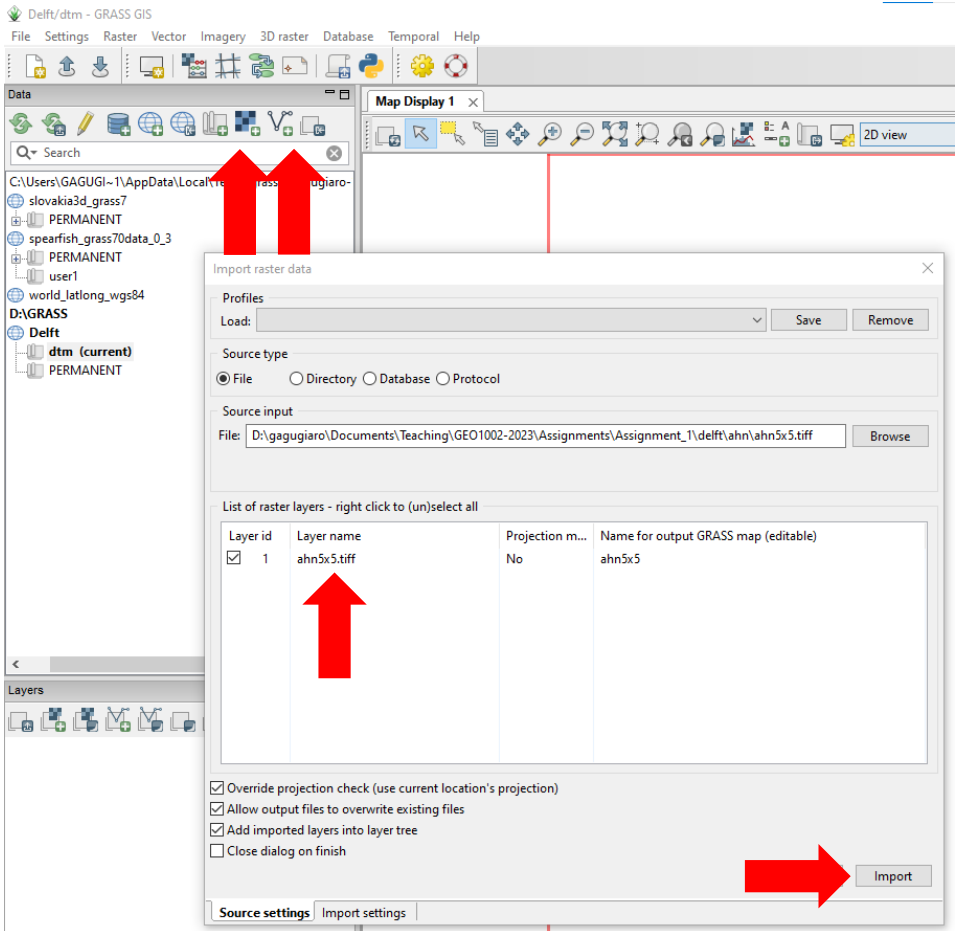
Adding a new GRASS mapset

- A new mapset is then added.
- Please note the **"current" suffix**. It reminds you of the mapset you are currently working in, i.e. All maps you will create will be written in the current mapset!



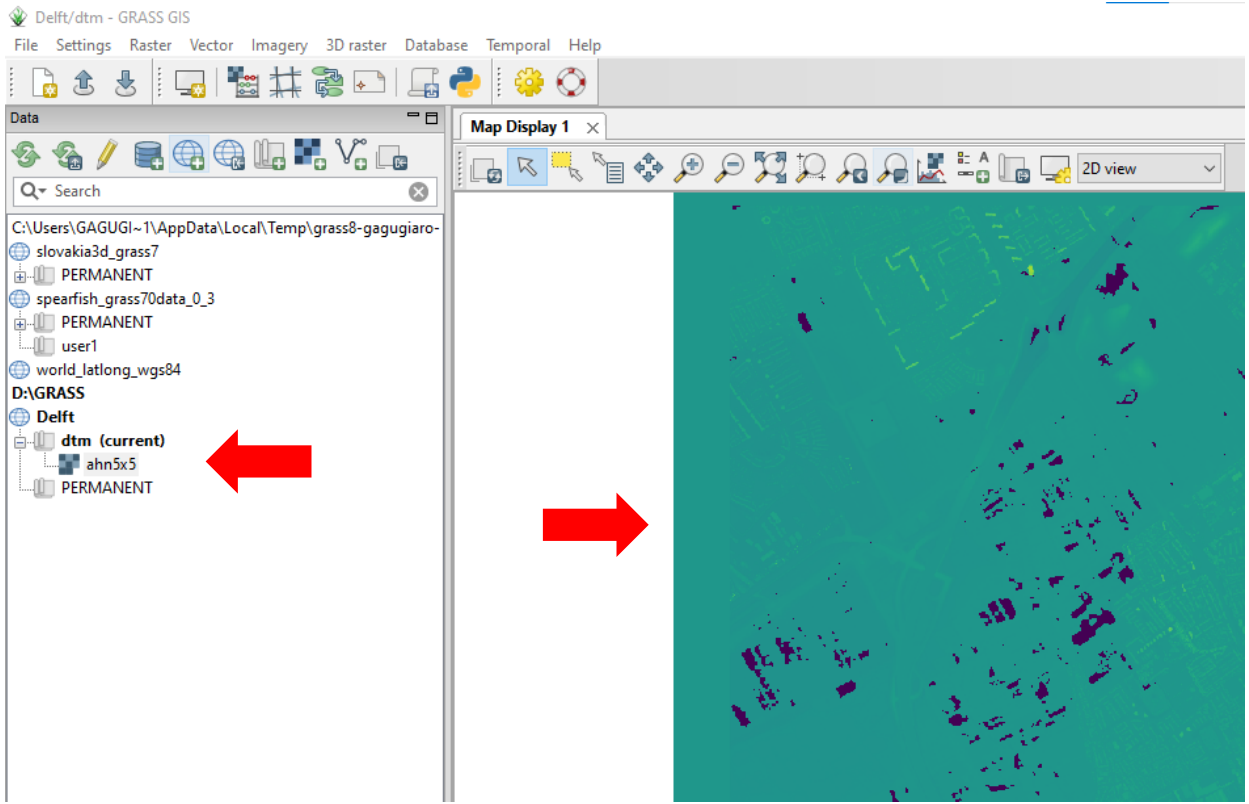
Importing data into the current mapset

- Once you have selected which mapset to work on, you can import data (vector or raster) by clicking on the proper "Import raster data" or "Import vector data" buttons
- Here, for example, we decide to import raster data. A window will open asking for the details. Here we import a till file containing the DTM of a portion of Delft
- Set the options and then click on "Import"



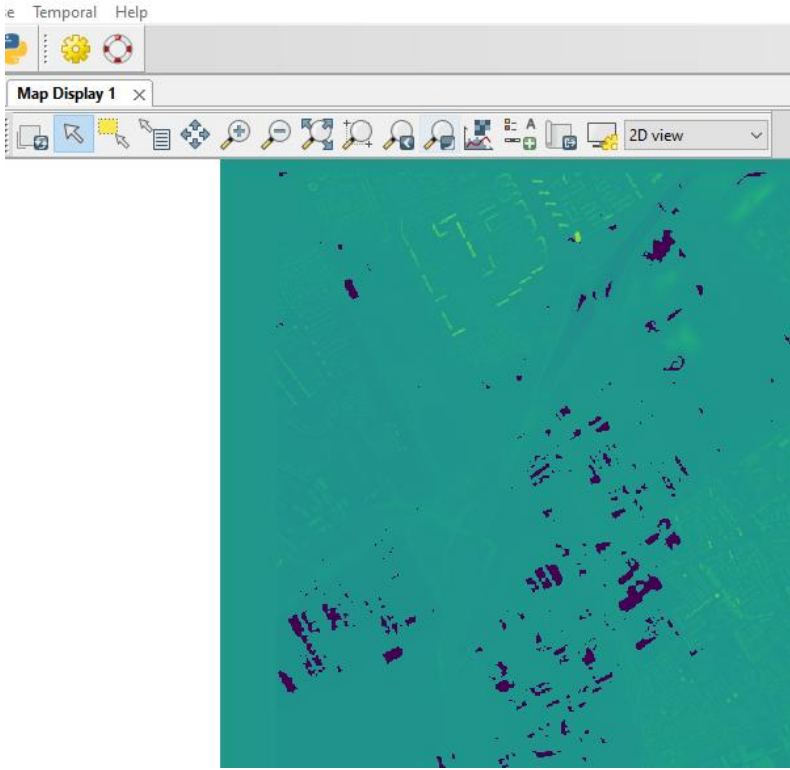
Importing data into the current mapset

- The file will be converted into a GRASS raster layer, added to the current mapset, and visualised in the Map Display



Setting the GRASS computational region

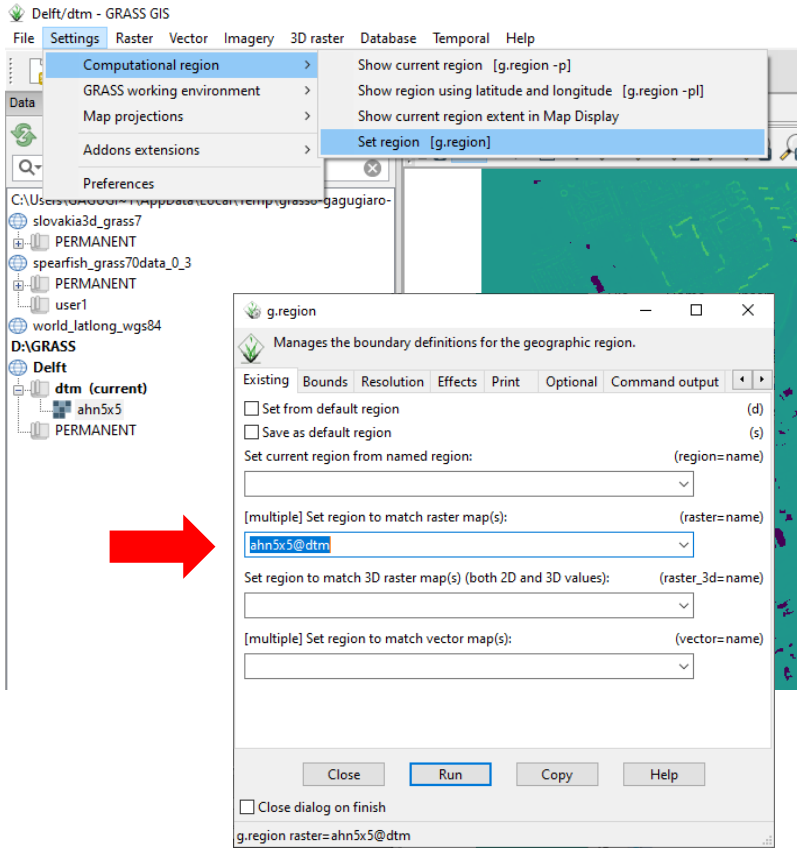
- When working in GRASS, it is FUNDAMENTAL to set the **computational region** correctly. Whatever operation you will carry out, it will be done **only within** the boundaries of such region.
- The region is displayed by means of a **red boundary in the map display**.
- It can be set and adapted according to your needs



No red boundary visible, the region is not set correctly!

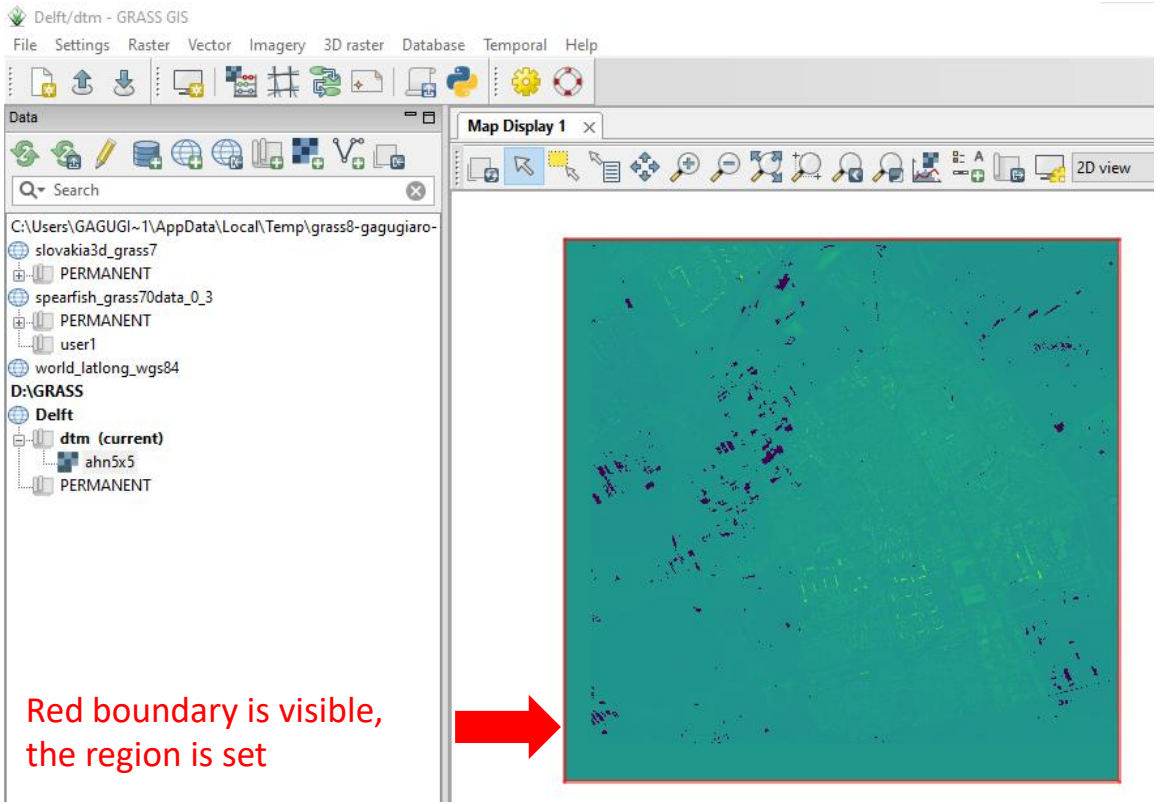
Setting the GRASS computational region

- You can set the region from the GUI via Settings\Computational region\Set region
- A command GUI will open. For example, you can set it to the extents of the previously imported DTM map (it is just an example!)



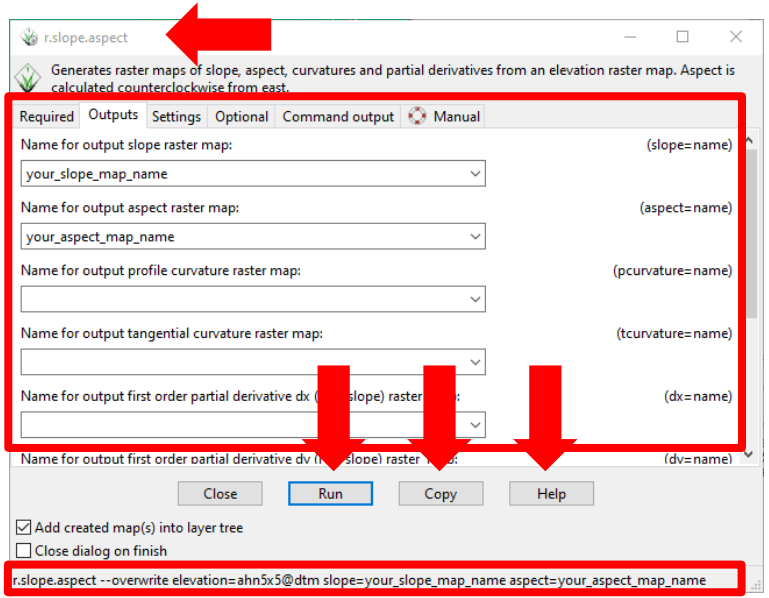
Setting the GRASS computational region

- Now you can see the current region as a red boundary around the selected raster layer.



Working with GRASS GIS: Module windows

- In general, nearly every GRASS module has an associated window GUI. The name of the module is written on the top of the window
- All user-defined parameters and flags can be set manually
- Once you are done, you simply press the "Run" button
- You can see (and copy) the full command with parameters and flags in the bottom part of the window. This is useful for learning how parameters are set, and to use the command in scripts to automate the process
- For help, simply press the "Help" button



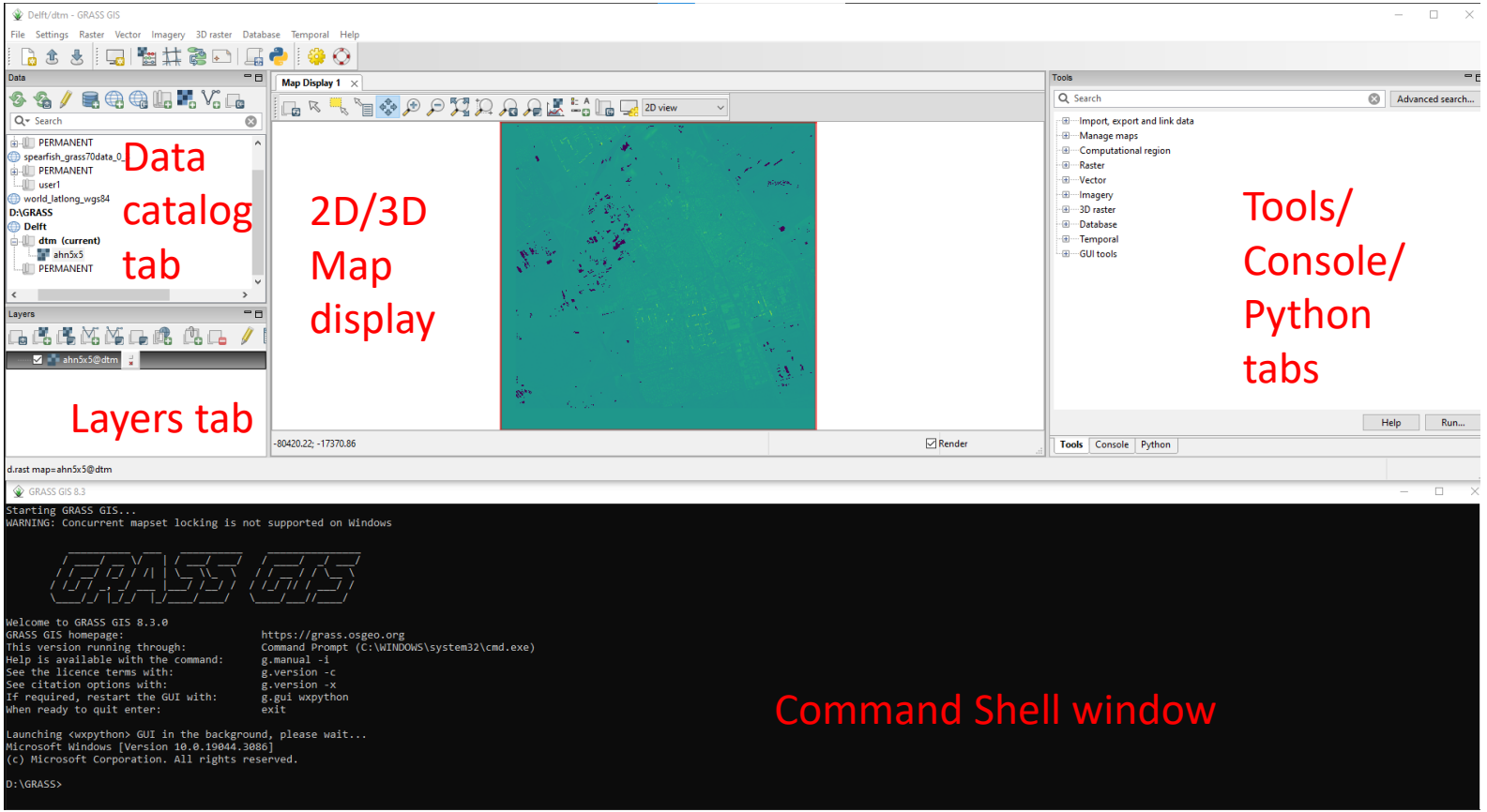
Working with GRASS GIS

There are **different, alternative ways** to launch and run the GRASS modules:

a) Using the **GUI drop-down menus** (the names of the modules are also suggested) with mouse and/or keyboard
b) Double clicking on the module name in the **Tools tab**
c) Writing the module name (with, optionally, parameters) in the **Console tab**
d) Writing the module name (with, optionally, parameters) in the **Command shell**
e) Writing Python code in the **Python tab**
f) Writing Python code in the **Python code editor**
g) From within **QGIS + GRASS Plugin** (not recommended)
h) A combination thereof

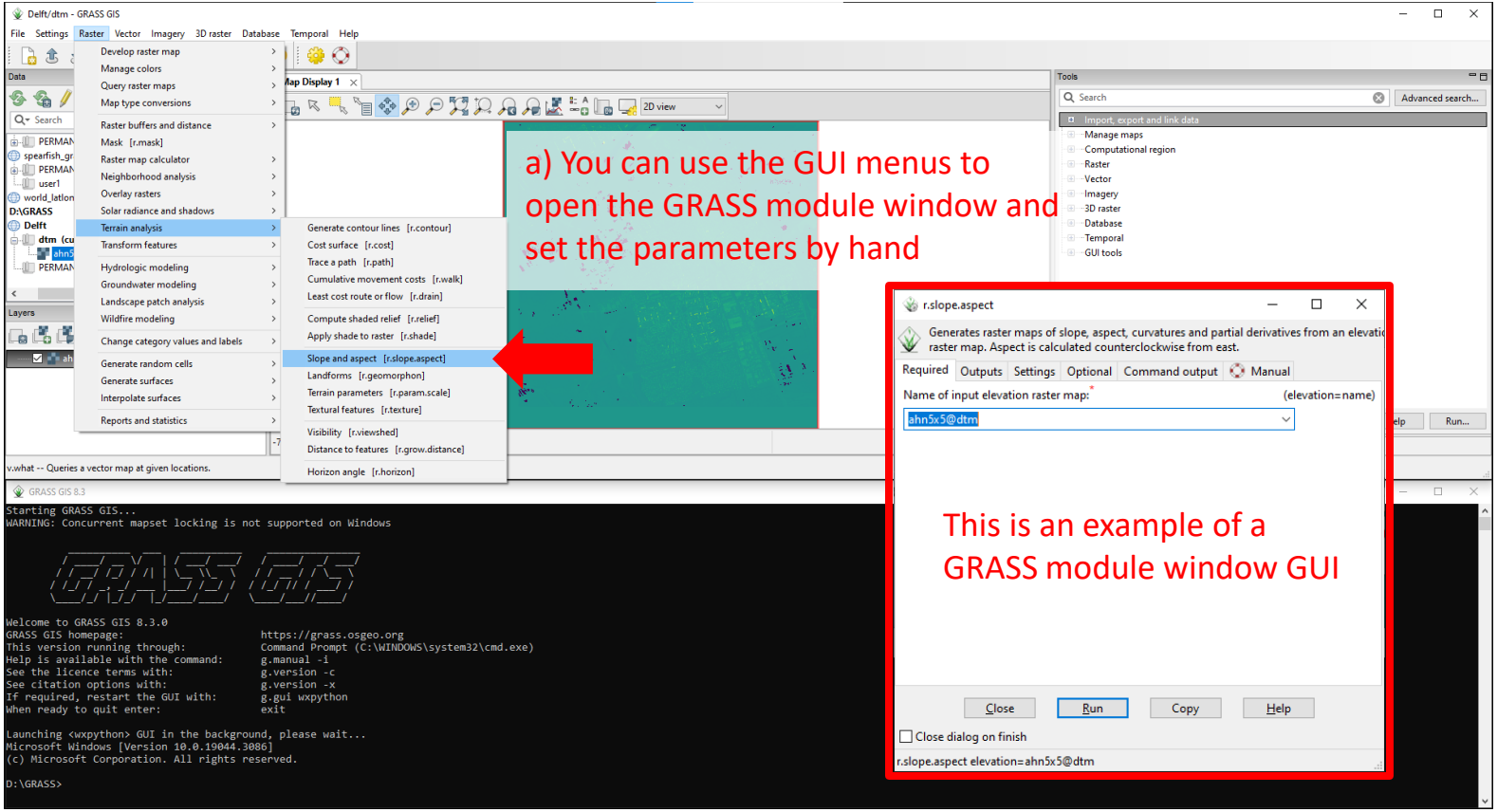
Working with GRASS GIS

- Intro
- GRASS modules
- GRASS GIS DB
- Running GRASS
- New setup
- **Running modules**
- Test data
- Help



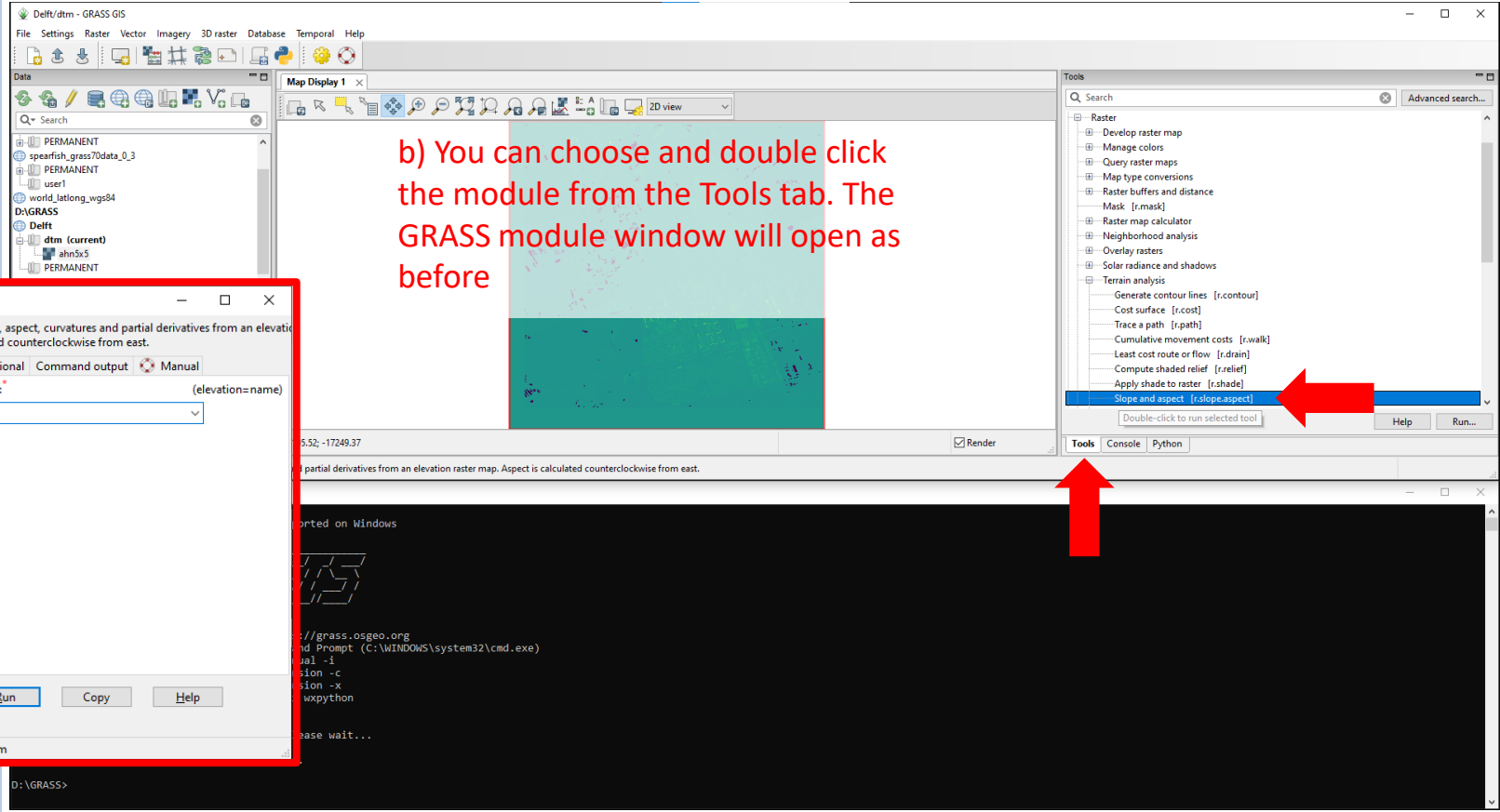
Working with GRASS GIS

- Intro
- GRASS modules
- GRASS GIS DB
- Running GRASS
- New setup
- **Running modules**
- Test data
- Help



Working with GRASS GIS

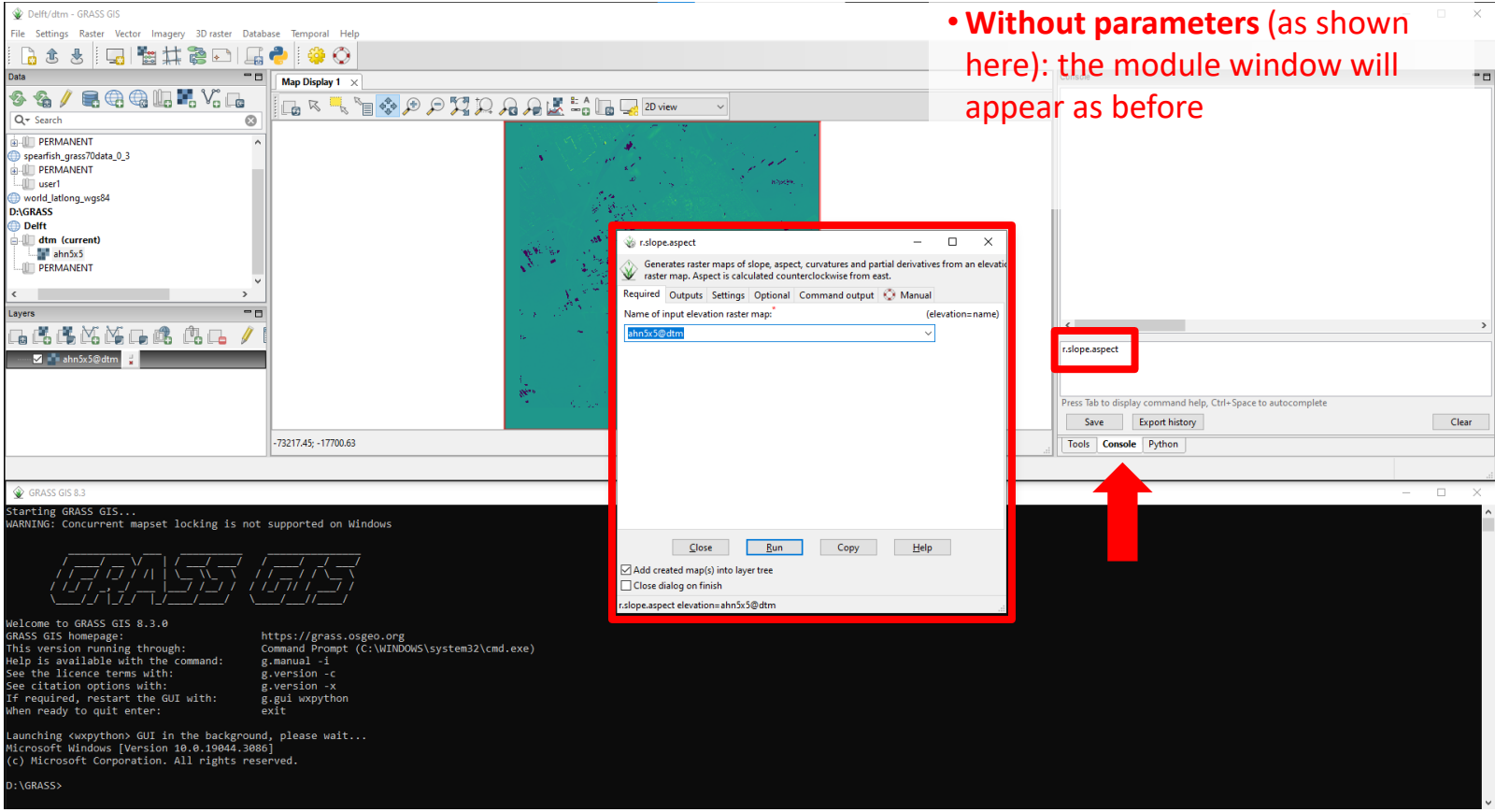
- Intro
- GRASS modules
- GRASS GIS DB
- Running GRASS
- New setup
- **Runn**
- Test d
- Help



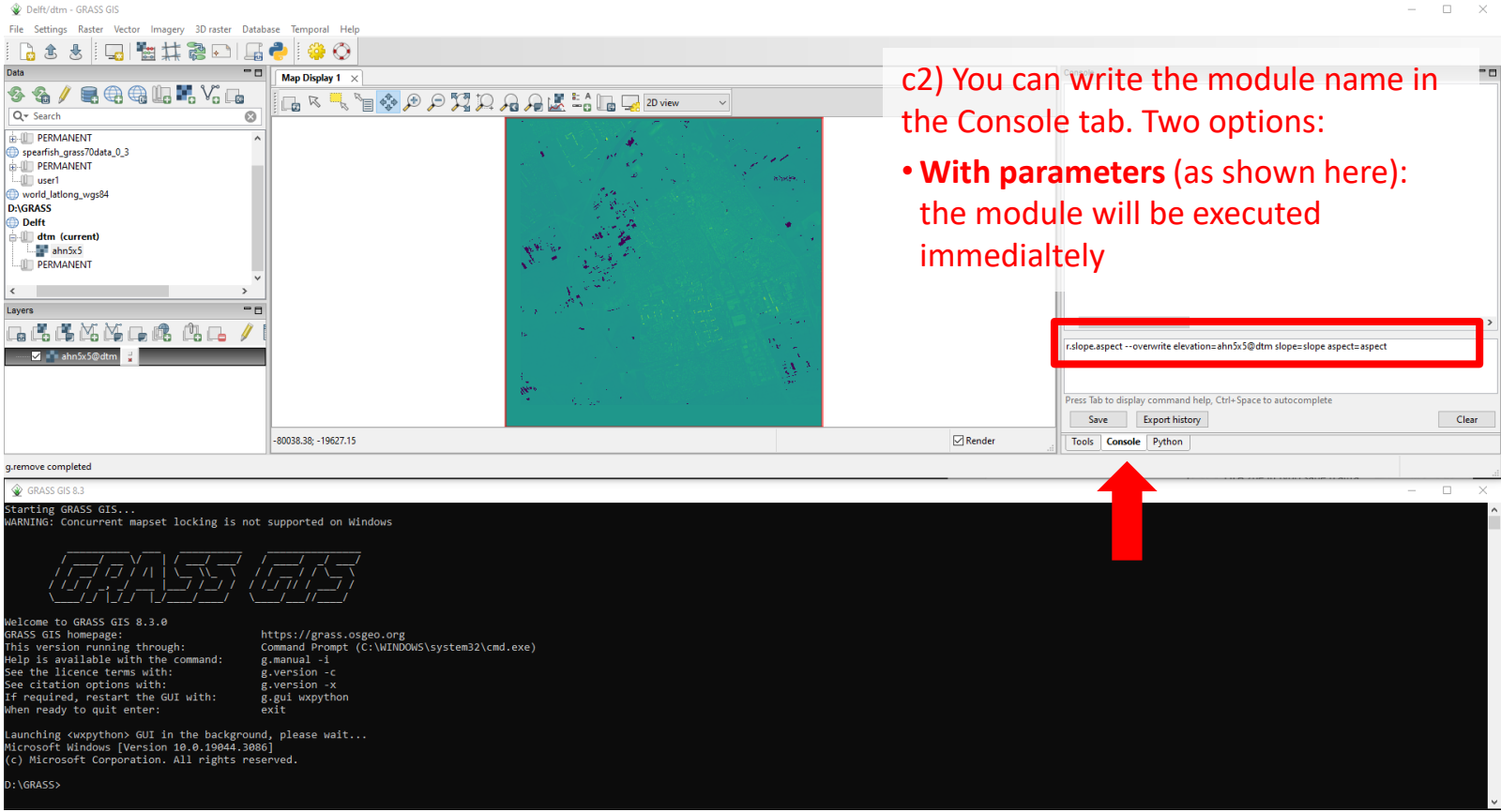
Working with GRASS GIS

c1) You can write the module name in the Console tab. Two options:

- **Without parameters** (as shown here): the module window will appear as before



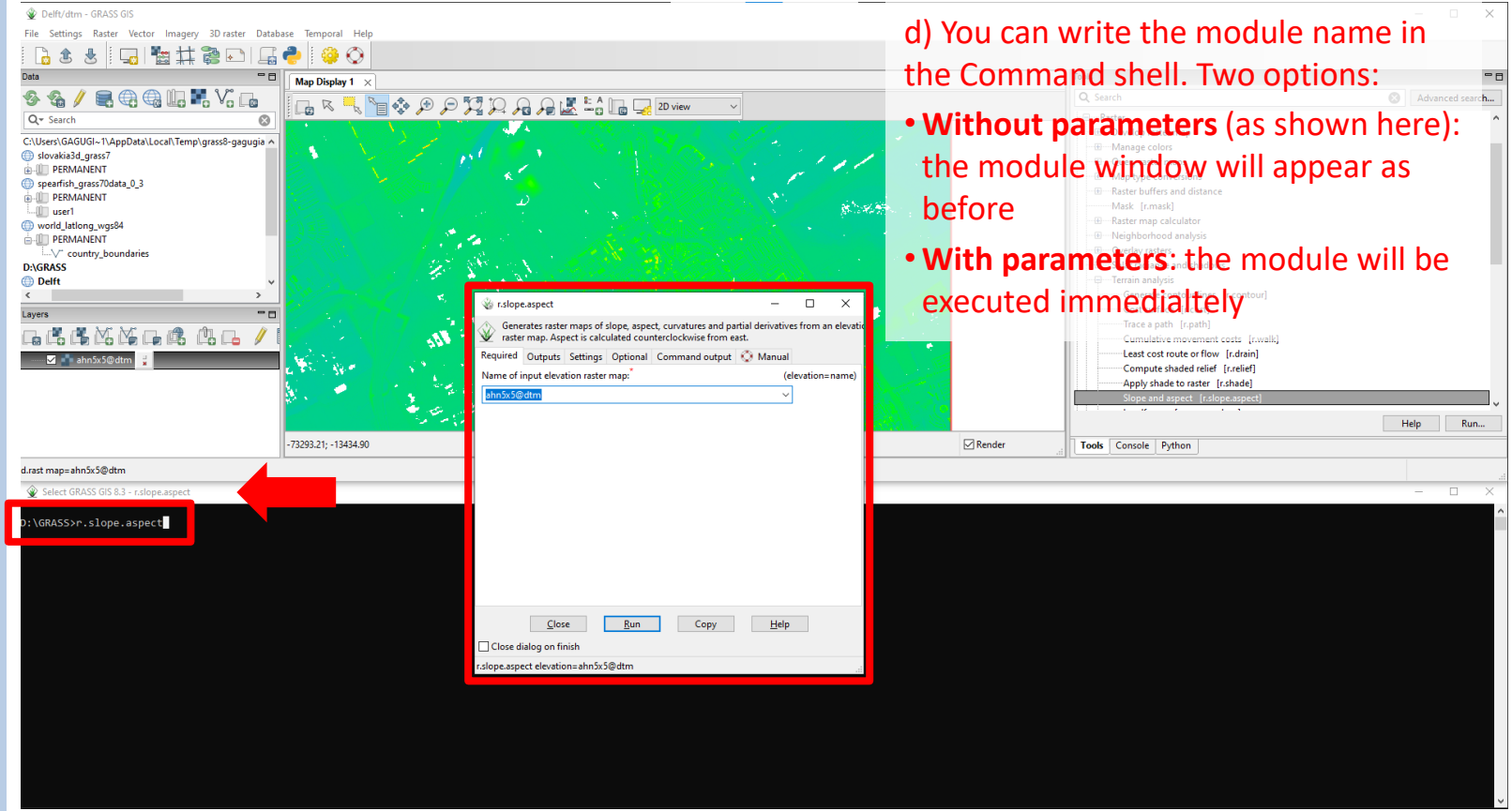
Working with GRASS GIS



c2) You can write the module name in the Console tab. Two options:

- **With parameters** (as shown here): the module will be executed immedialtely

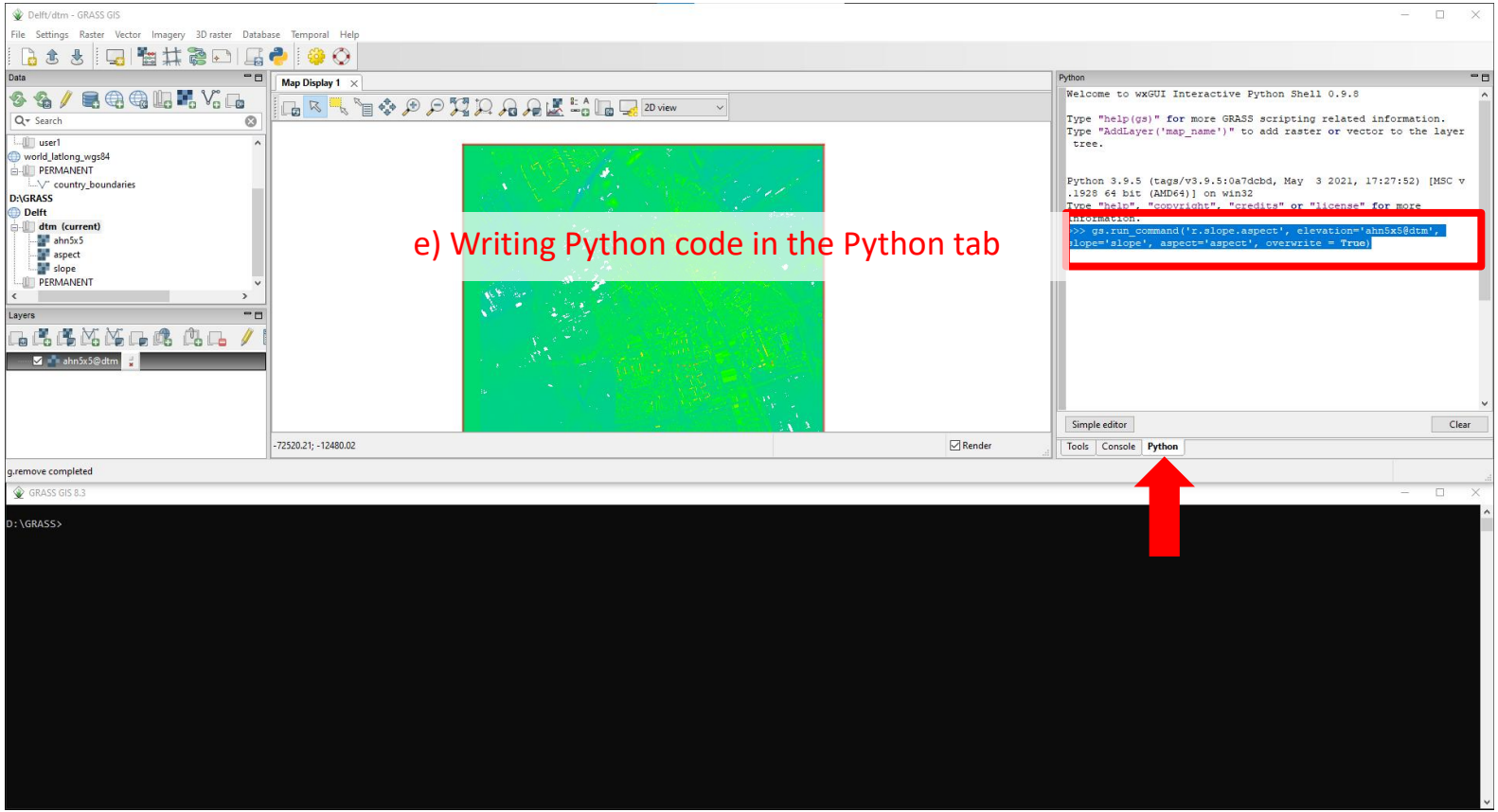
Working with GRASS GIS



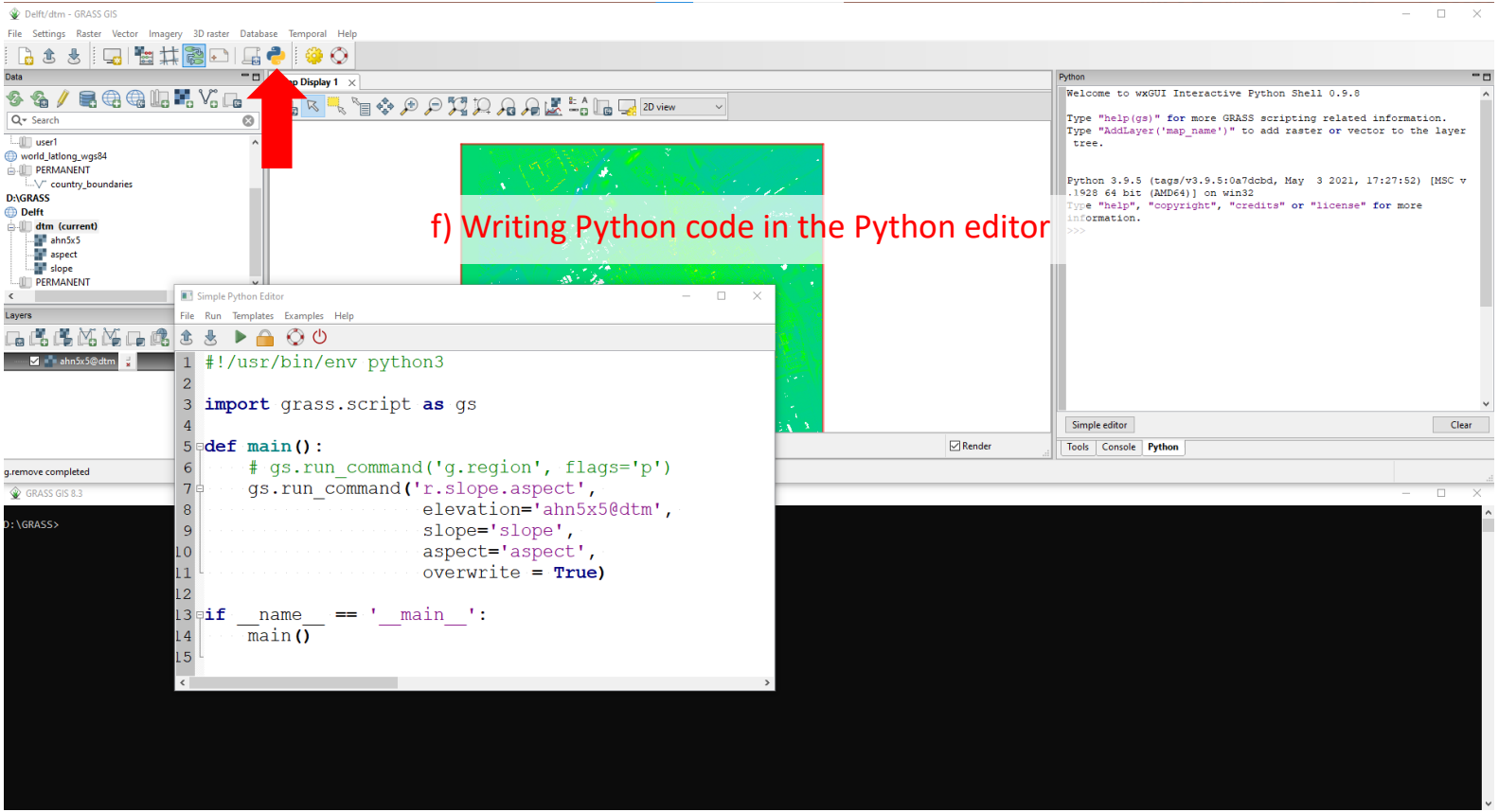
d) You can write the module name in the Command shell. Two options:

- **Without parameters** (as shown here): the module window will appear as before
- **With parameters**: the module will be executed immedialtely

Working with GRASS GIS

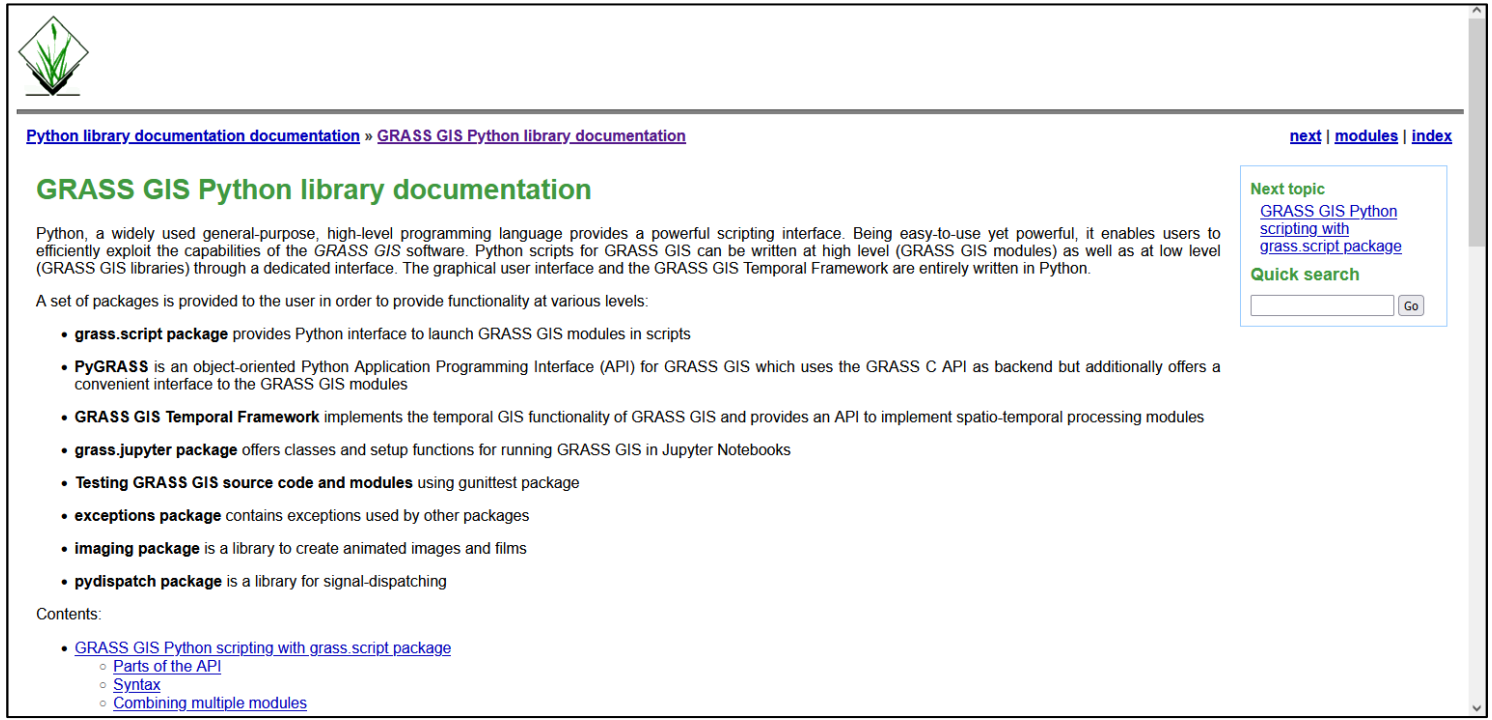


Working with GRASS GIS



GRASS and Python

- There is a useful manual providing all information on scripting GRASS with Python available at: https://grass.osgeo.org/grass84/manuals/libpython/index.html



Python library documentation documentation » GRASS GIS Python library documentation

next | modules | index

GRASS GIS Python library documentation

Python, a widely used general-purpose, high-level programming language provides a powerful scripting interface. Being easy-to-use yet powerful, it enables users to efficiently exploit the capabilities of the *GRASS GIS* software. Python scripts for GRASS GIS can be written at high level (GRASS GIS modules) as well as at low level (GRASS GIS libraries) through a dedicated interface. The graphical user interface and the GRASS GIS Temporal Framework are entirely written in Python.

A set of packages is provided to the user in order to provide functionality at various levels:

- **grass.script package** provides Python interface to launch GRASS GIS modules in scripts
- **PyGRASS** is an object-oriented Python Application Programming Interface (API) for GRASS GIS which uses the GRASS C API as backend but additionally offers a convenient interface to the GRASS GIS modules
- **GRASS GIS Temporal Framework** implements the temporal GIS functionality of GRASS GIS and provides an API to implement spatio-temporal processing modules
- **grass.jupyter package** offers classes and setup functions for running GRASS GIS in Jupyter Notebooks
- **Testing GRASS GIS source code and modules** using gunittest package
- **exceptions package** contains exceptions used by other packages
- **imaging package** is a library to create animated images and films
- **pydispatch package** is a library for signal-dispatching

Contents:

- GRASS GIS Python scripting with grass.script package
  - Parts of the API
  - Syntax
  - Combining multiple modules

Next topic

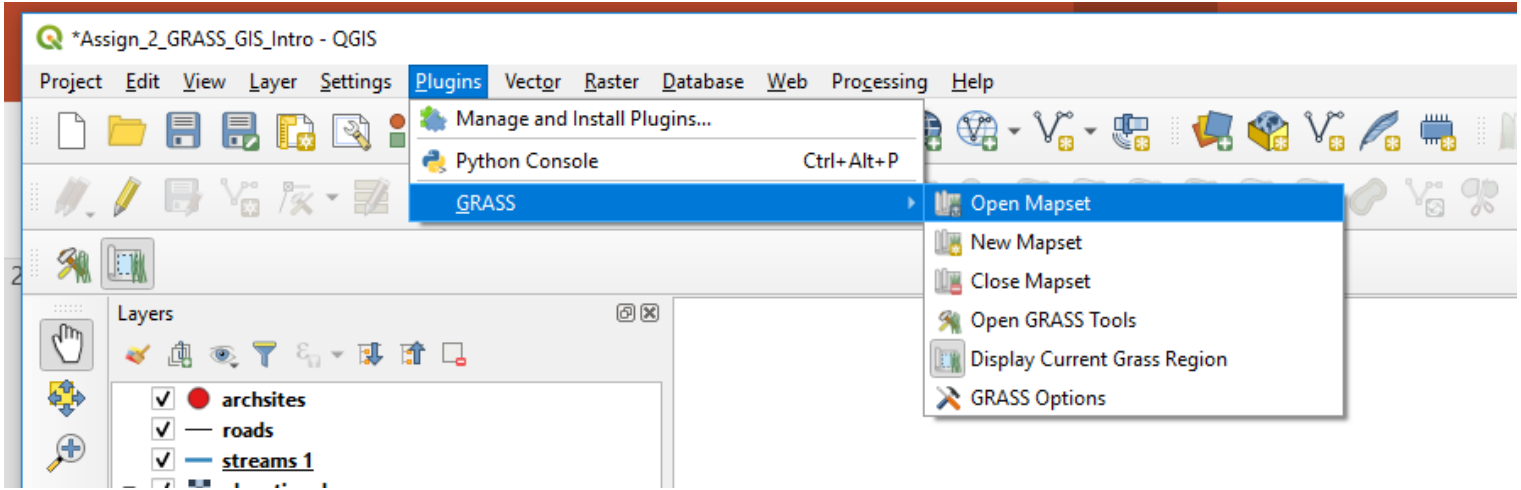
GRASS GIS Python scripting with grass.script package

Quick search

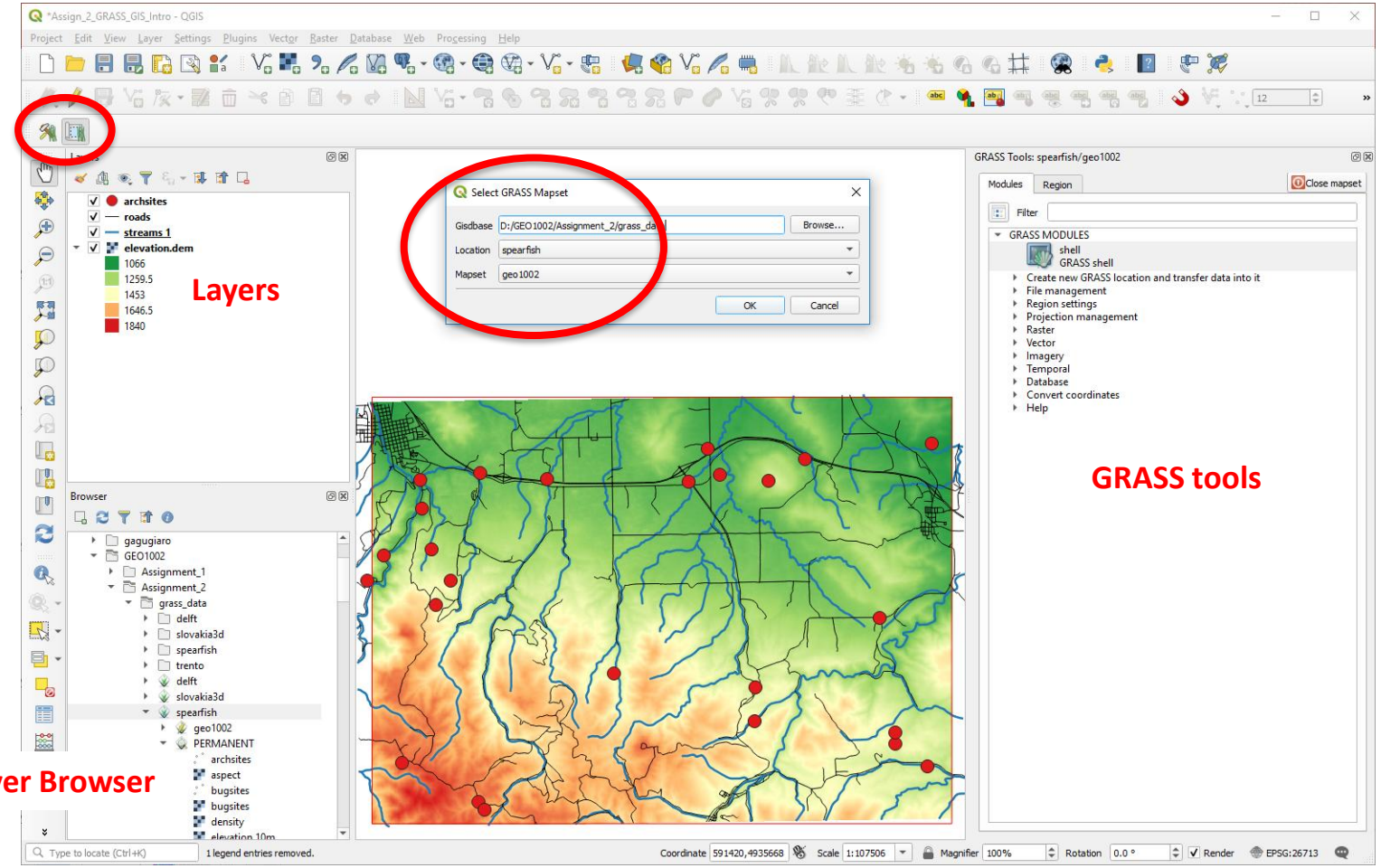
Go

Working with GRASS GIS: From QGIS

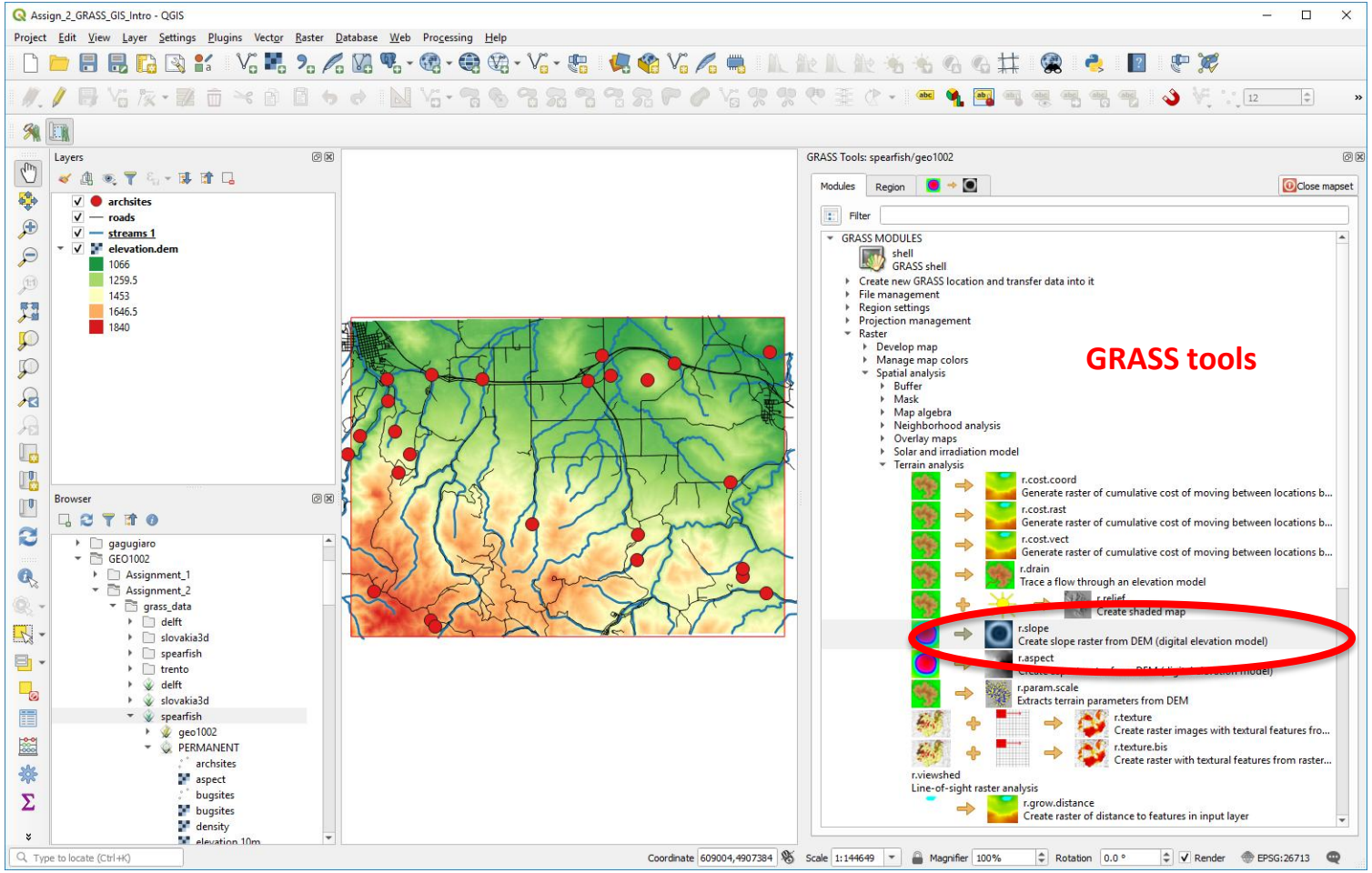
- Check that the GRASS plugin is installed and running!
- Select Plugins\GRASS\Open Mapset to configure GRASS in QGIS



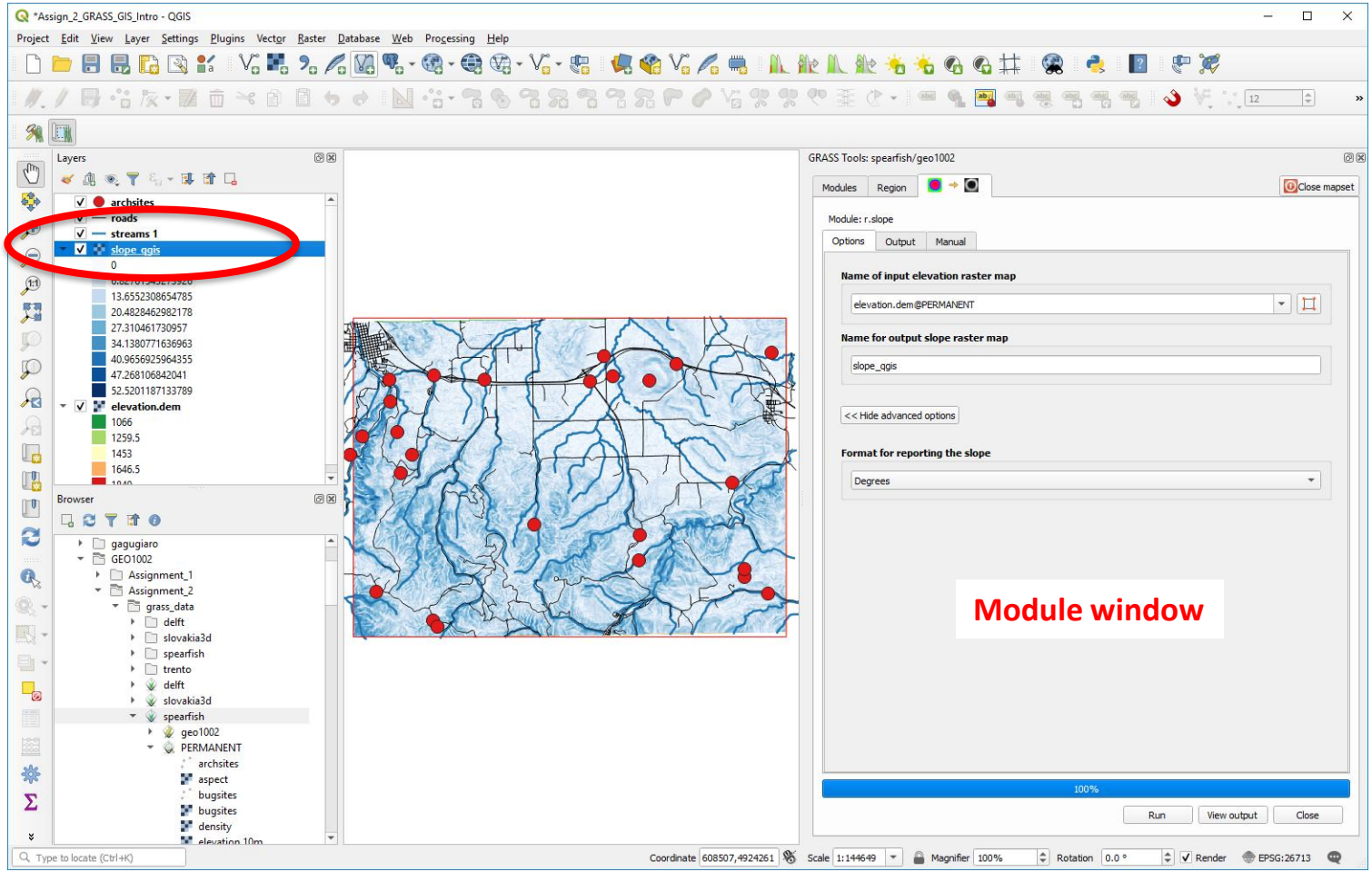
Working with GRASS GIS: From QGIS



Working with GRASS GIS: From QGIS



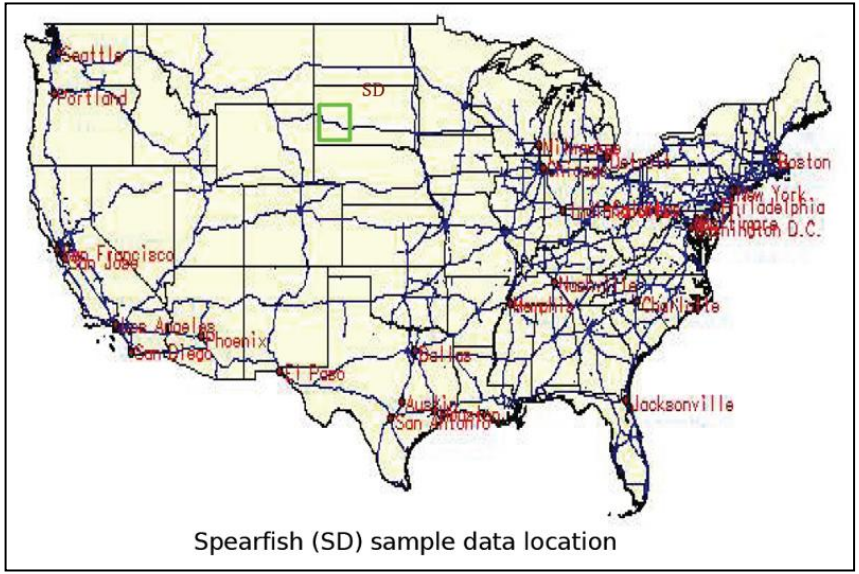
Working with GRASS GIS: From QGIS



GRASS GIS test data

There are many sample datasets to learn GRASS GIS, to “play around” and get used to how the program works. For example:

- **North Carolina**
  - LOTS of data in all formats
- **Spearfish (South Dakota)**
  - Contains raster e vector data
- **Spearfish imagery**
  - contains satellite imagery
- **Slovakia3d**
  - contains grid3d data (voxels)

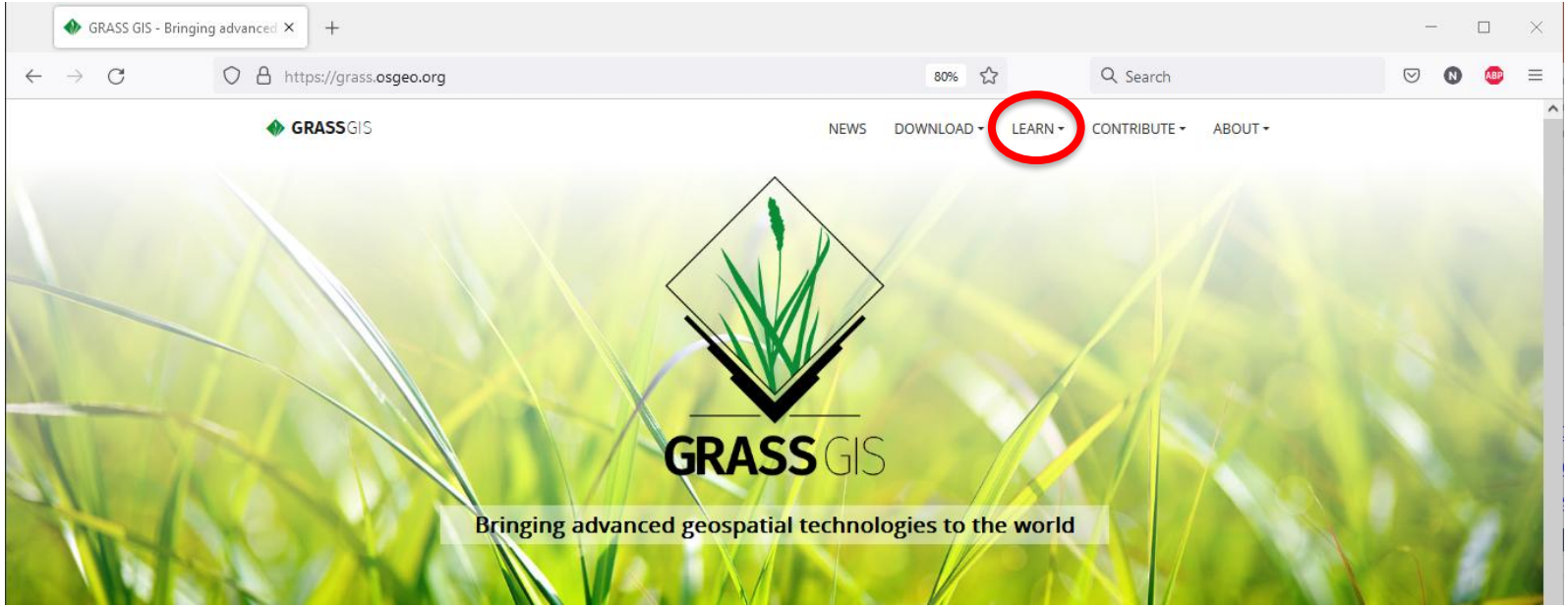


Spearfish (SD) sample data location

Download from https://grass.osgeo.org/download/data/

Help with GRASS GIS

- GRASS GIS **manual:** https://grass.osgeo.org/learn/manuals/
- GRASS GIS **wiki:** https://grasswiki.osgeo.org/wiki/GRASS-Wiki
- GRASS GIS **tutorials:** https://grass.osgeo.org/learn/tutorials/



Help with GRASS modules

- Run the module name with "**--help**" or "**--h**" in the Console window or in the Command shell
  - Example: r.slope.aspect --help
- From the module window, click on help button
- Refer to the GRASS online manual, where you find help for each module (see next slide)

Help with GRASS modules

Quick Introduction

- How to start with GRASS GIS
- Index of topics and keywords
- Intro: projections and spatial transformations
- FAQ - Frequently Asked Questions (Wiki)
- Graphical index of functionality

Graphical User Interface

- wxGUI
- wxGUI components
- wxGUI toolboxes
- GUI commands

Display

- Display commands manual
- Display drivers

General

- GRASS GIS startup manual
- General commands manual

Addons

- Addons manual pages

Programmer's Manual

- Programmer's Manual

Raster processing

- Intro: 2D raster map processing
- Raster commands manual

3D raster processing

- Intro: 3D raster map (voxel) processing
- 3D raster (voxel) commands manual

Image processing

- Intro: image processing
- Imagery commands manual

Vector processing

- Intro: vector map processing and network analysis
- Vector commands manual
- GRASS ASCII vector format specification

Database

- Intro: database management
- SQL support in GRASS GIS
- Database commands manual

Temporal processing

- Intro: temporal data processing
- Temporal commands manual

Cartography

- PostScript commands manual
- wxGUI Cartographic Composer

Miscellaneous & Variables

- Miscellaneous commands manual
- GRASS variables and environment variables

Python

- GRASS GIS Python library documentation
- PyGRASS documentation
- GRASS GIS in Jupyter Notebooks

Dr. Giorgio Agugiaro

g.agugiaro@tudelft.nl

3D Geoinformation Group

TU Delft

The Netherlands

https://3d.bk.tudelft.nl/gagugiaro