Introduction to GRASS GIS

Time Series Analysis of Remote Sensing Data

Micha Silver
Remote Sensing Laboratory
Ben Gurion University, Sde Boker Campus

11/06/2018

email: micha@arava.co.il

Goals

Some background

Initial setup

Examples of batch processing

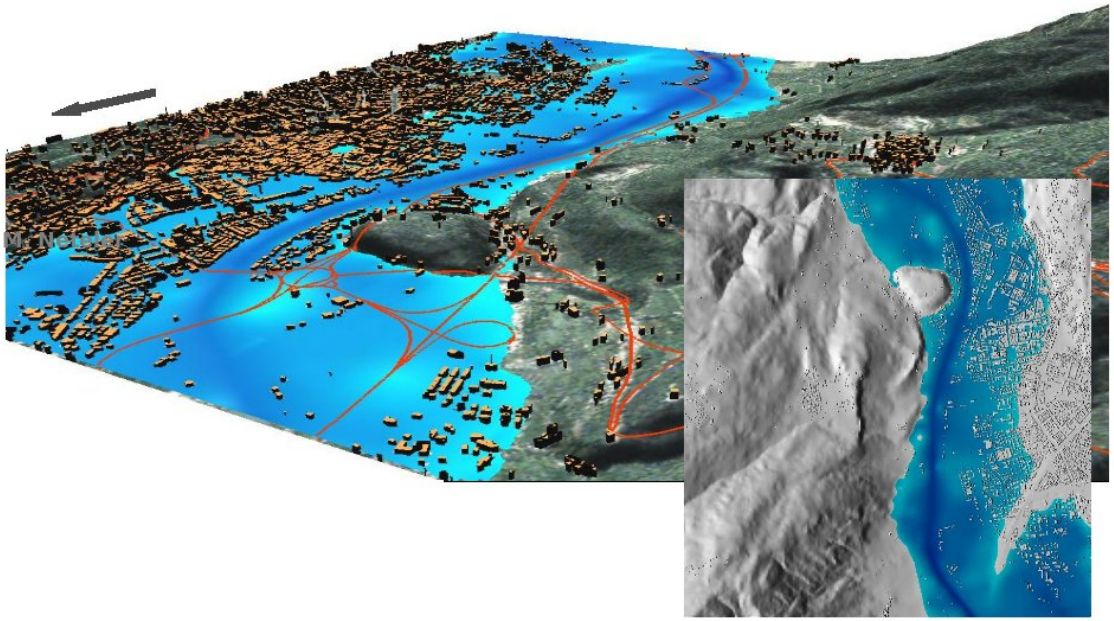
Define and analyze GRASS Spatial-Temporal Dataset

Overview

- Among the most experienced GIS in the world
- Actively developed, current stable version: 7.4.0
- License: GPL
- About 350 modules - raster, vector, database, space-time
- Over 130 addons
- Topologically clean vector format
- 3D visualizations
- Fully script-able in bash or python
- *GRASS Development Team, 2018. Geographic Resources Analysis Support System (GRASS) Software, Version 7.4. Open Source Geospatial Foundation. https://grass.osgeo.org*

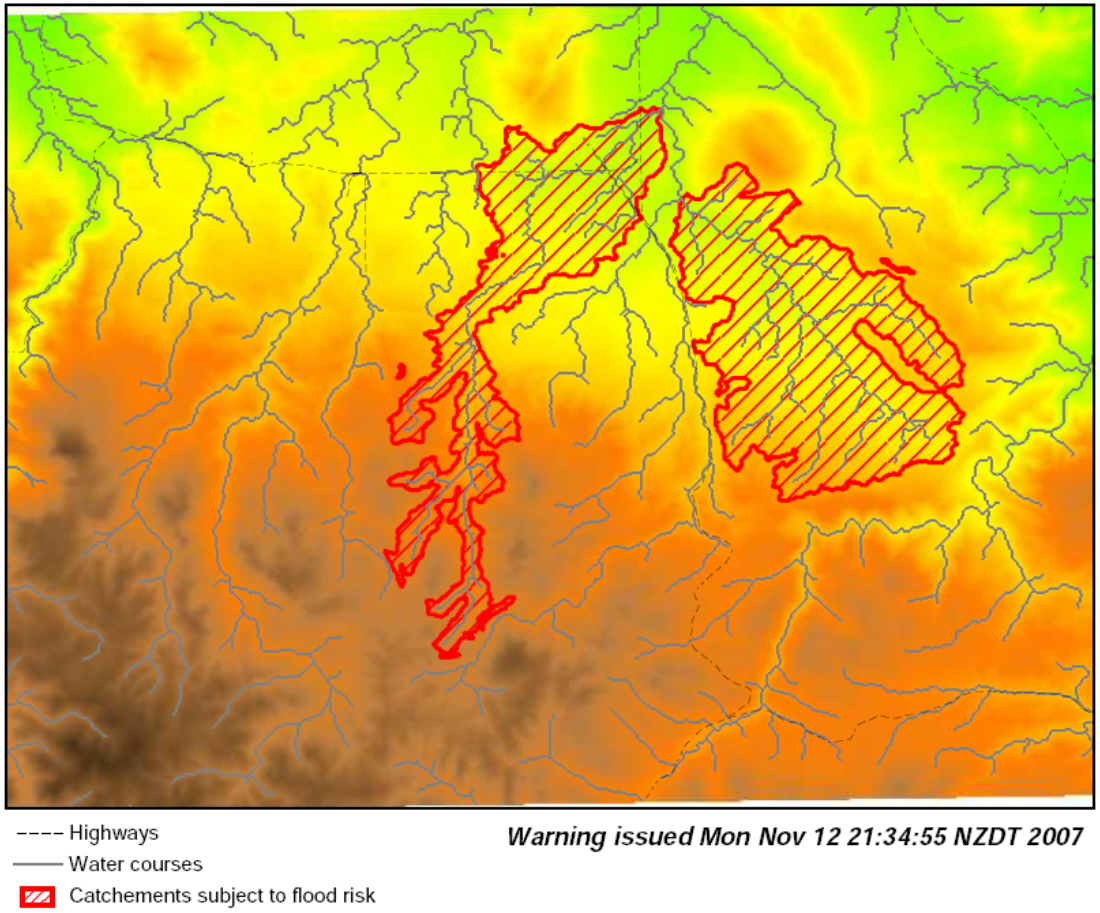
Use cases

- Flooding simulation



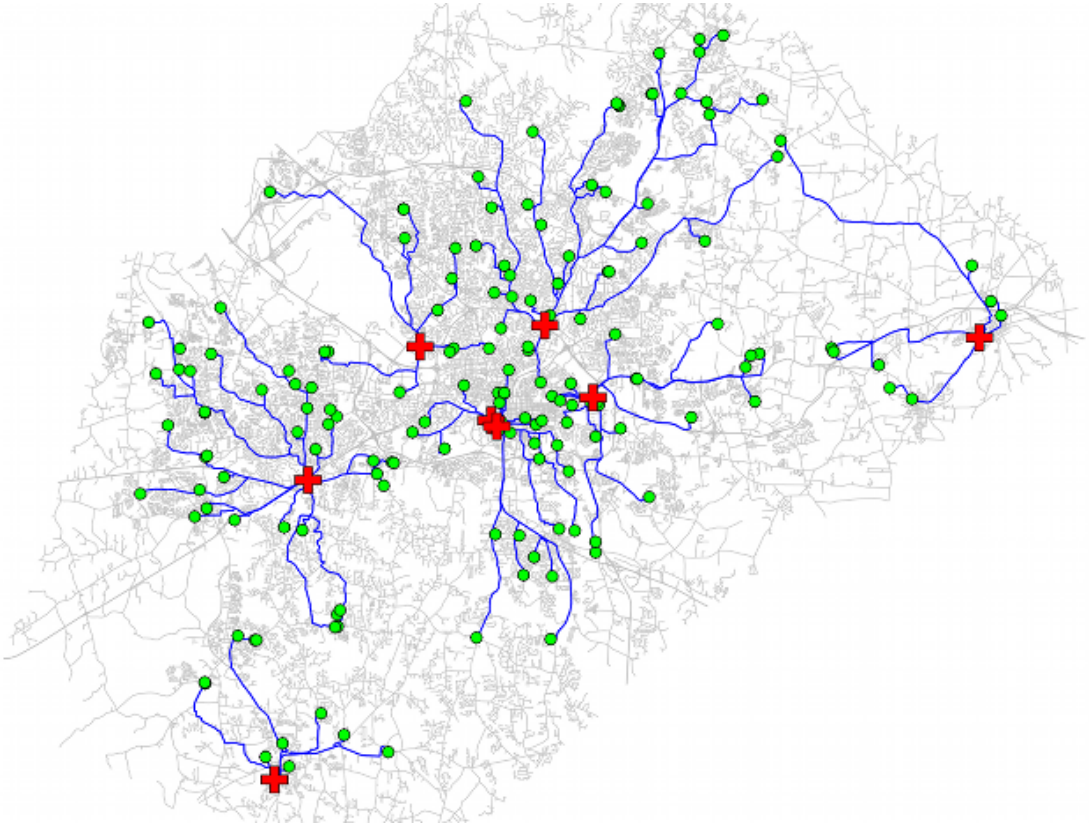
Use cases

- Flooding simulation
- Flood hazards



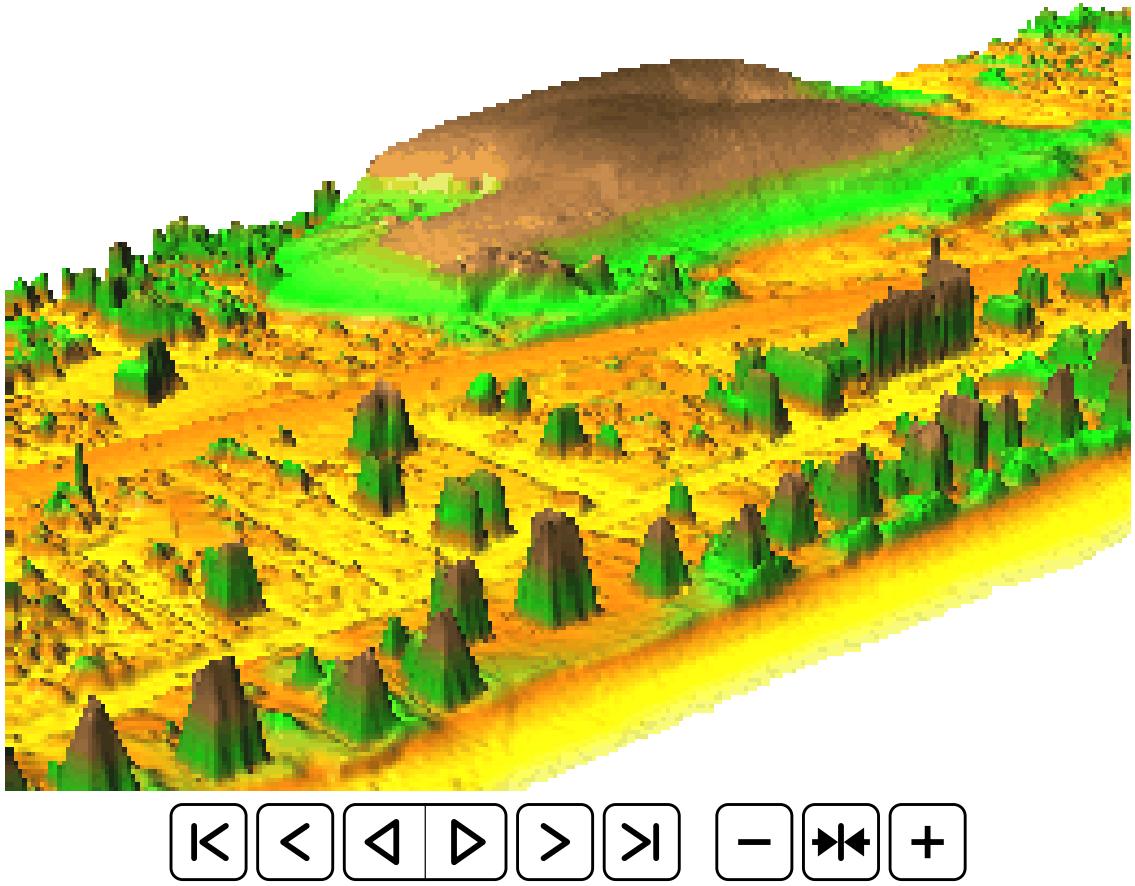
Use cases

- Flooding simulation
- Flood hazards
- Shortest distance



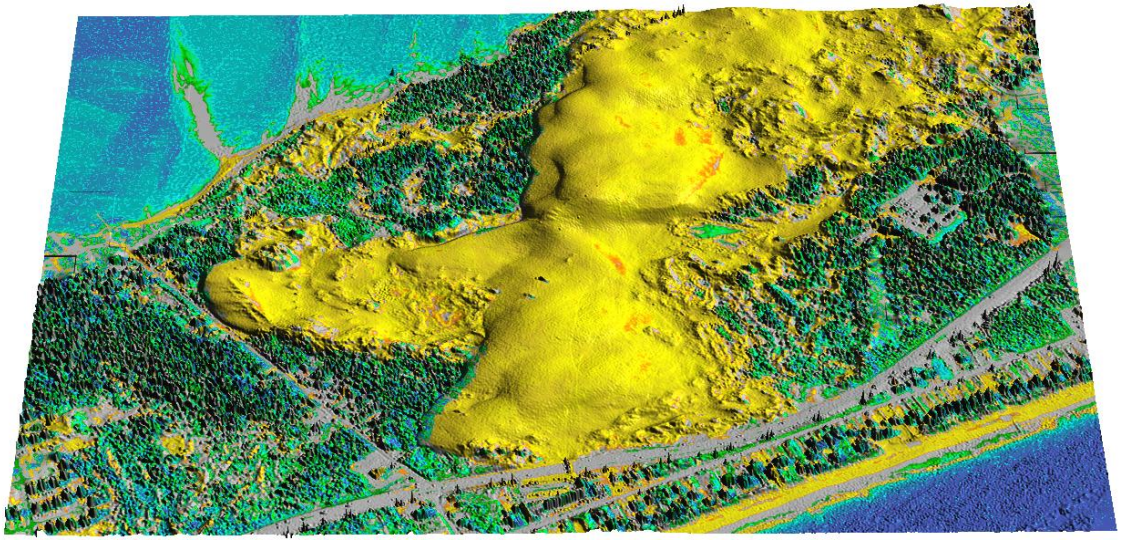
Use cases

- Flooding simulation
- Flood hazards
- Shortest distance
- Landscape changes

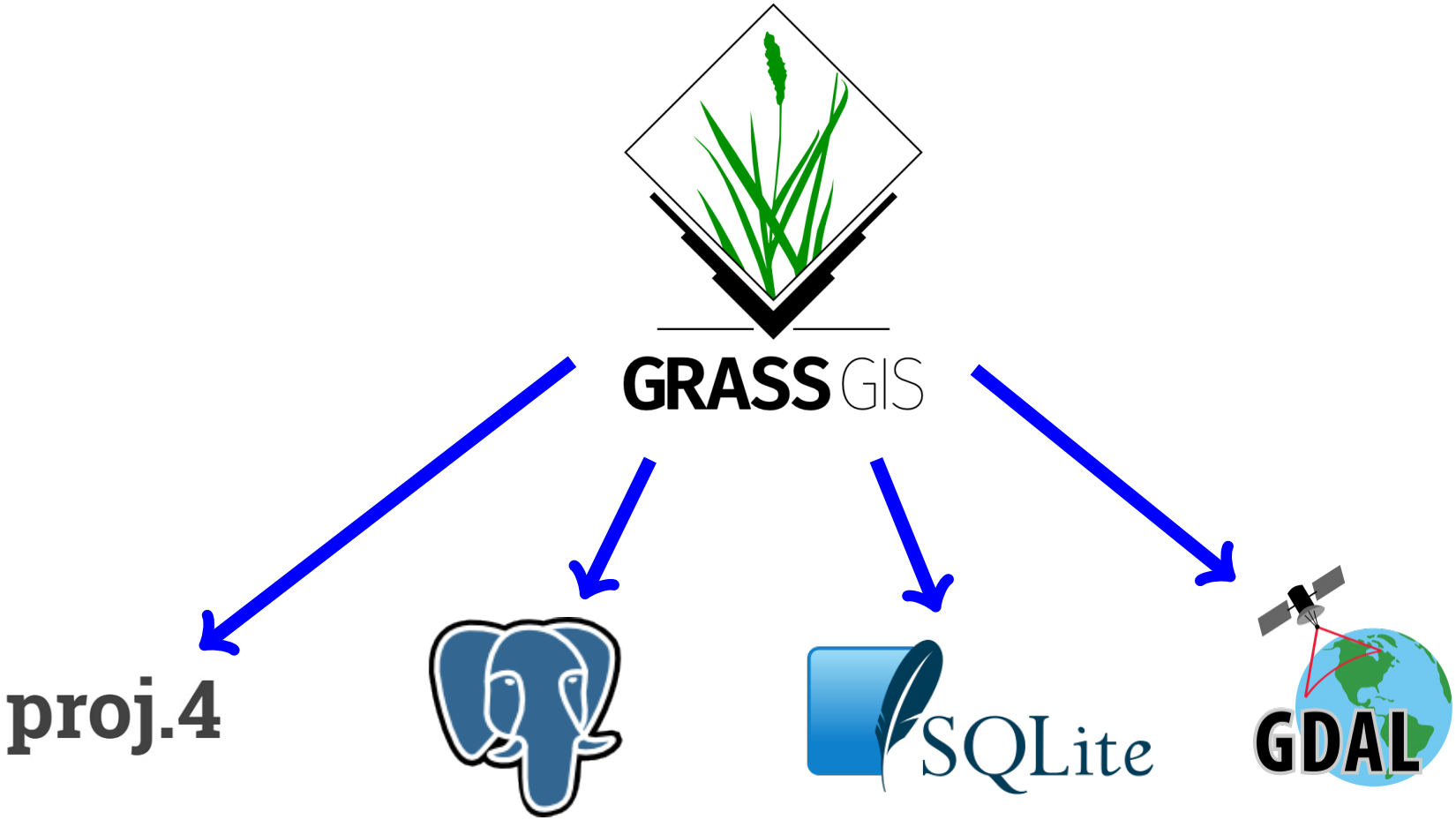


Use cases

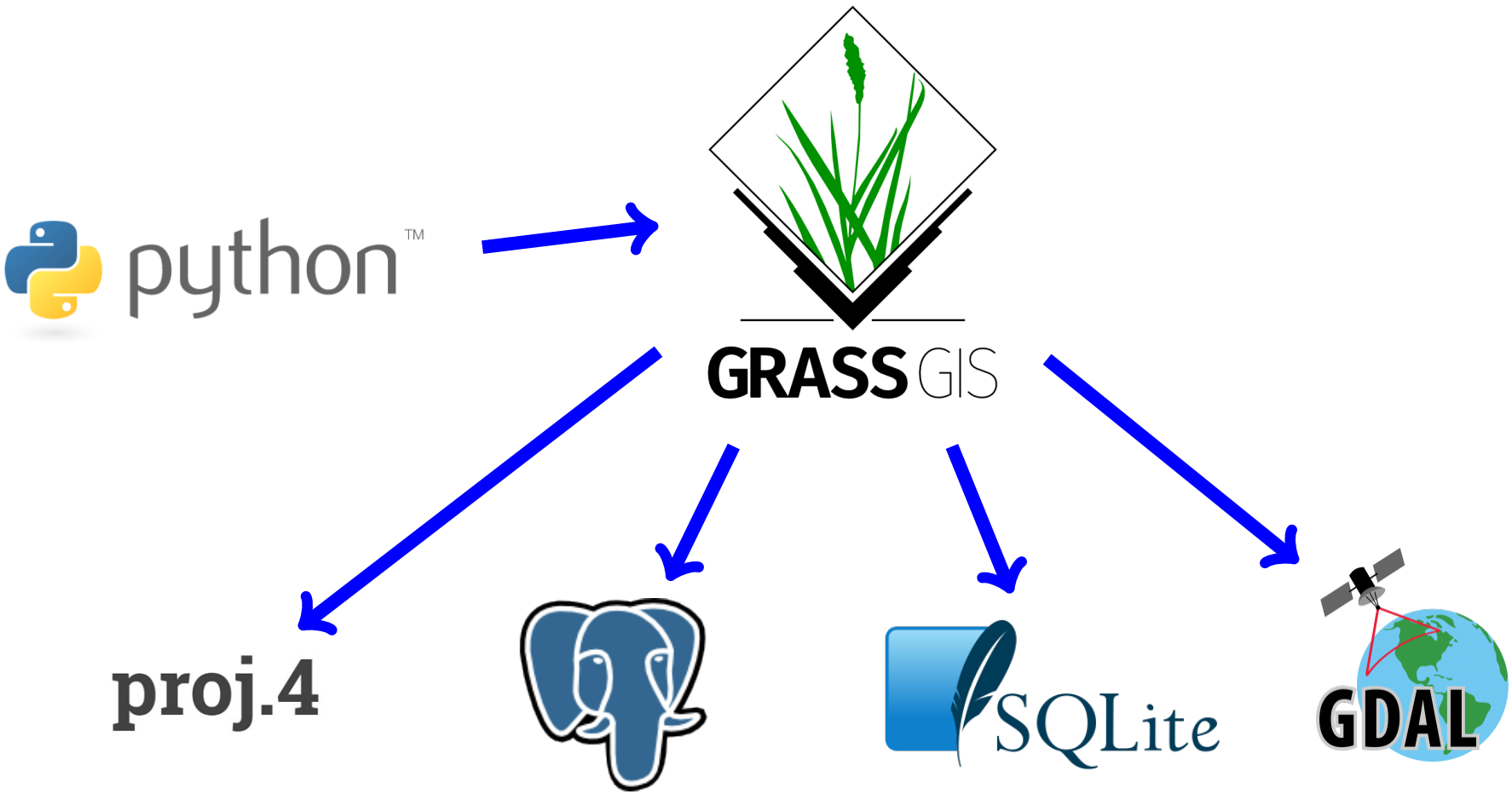
- Flooding simulation
- Flood hazards
- Shortest distance
- Landscape changes
- LIDAR point clouds



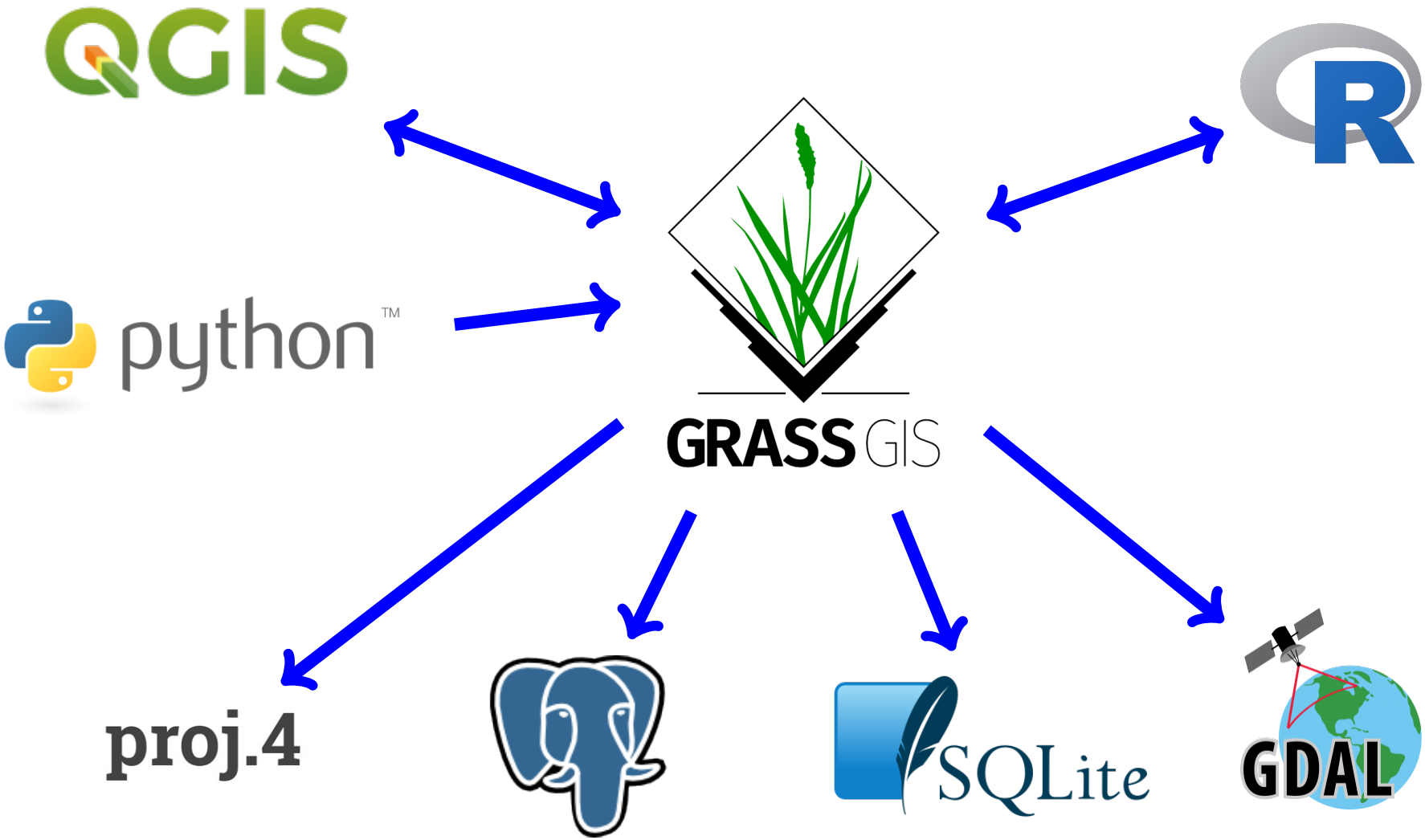
Interaction with Open Source GIS Ecosystem



Interaction with Open Source GIS Ecosystem



Interaction with Open Source GIS Ecosystem

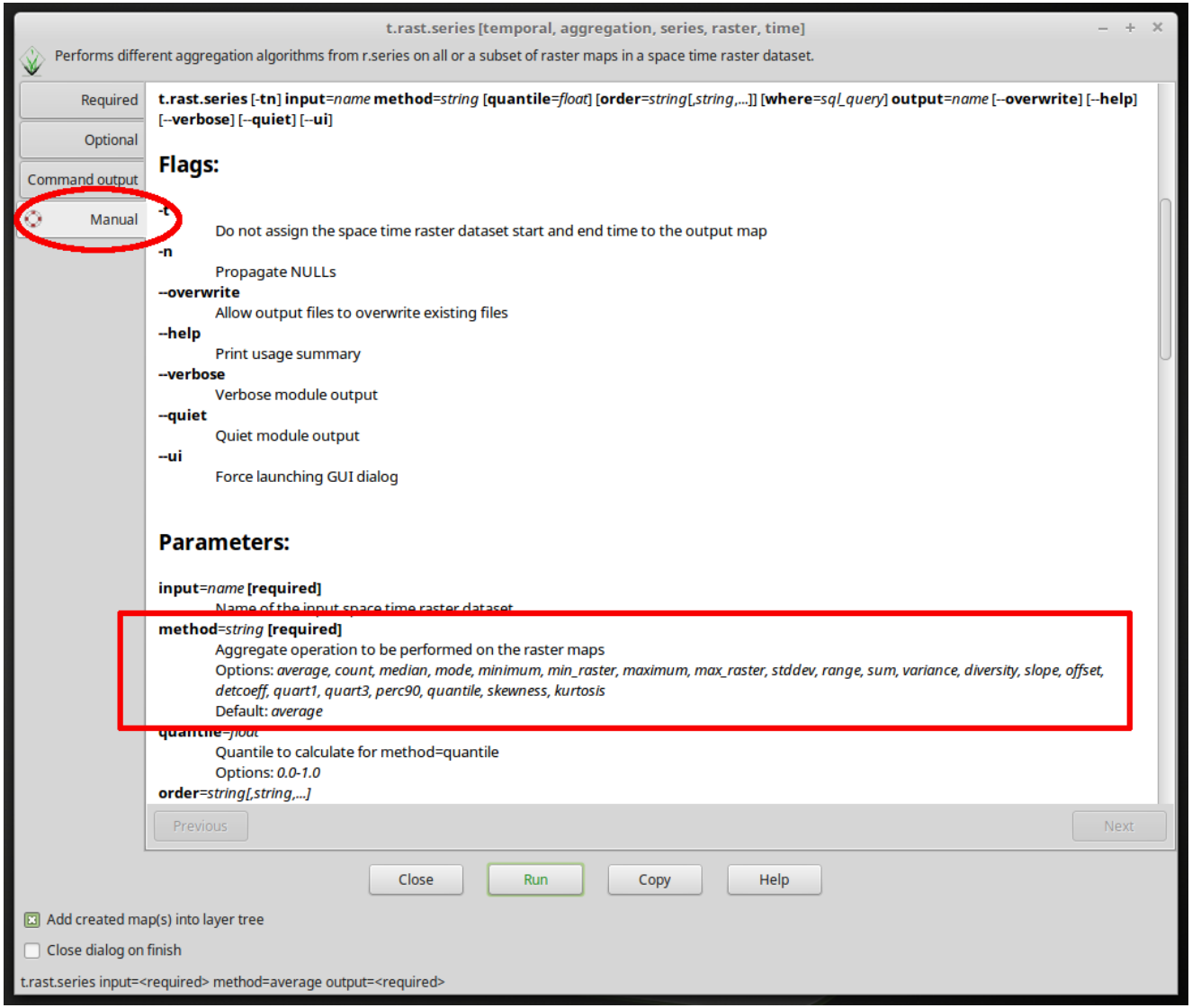


Online Documentation

- Tutorial:
  http://www.training.gismentors.eu/grass-gis-workshop-jena-2018/units/01.html
- Wiki:
  https://grasswiki.osgeo.org/wiki/From_GRASS_GIS_novice_to_power_user_(workshop_at_FOSS4G_Boston_2017)
- The GRASS book:
  https://grassbook.org/
- User's maillist:
  http://lists.osgeo.org/mailman/listinfo/grass-user
- NCSU Lab:
  http://ncsu-geoforall-lab.github.io/geospatial-modeling-course/grass/
- Spatial-Temporal workshop:
  http://ncsu-geoforall-lab.github.io/grass-temporal-workshop/

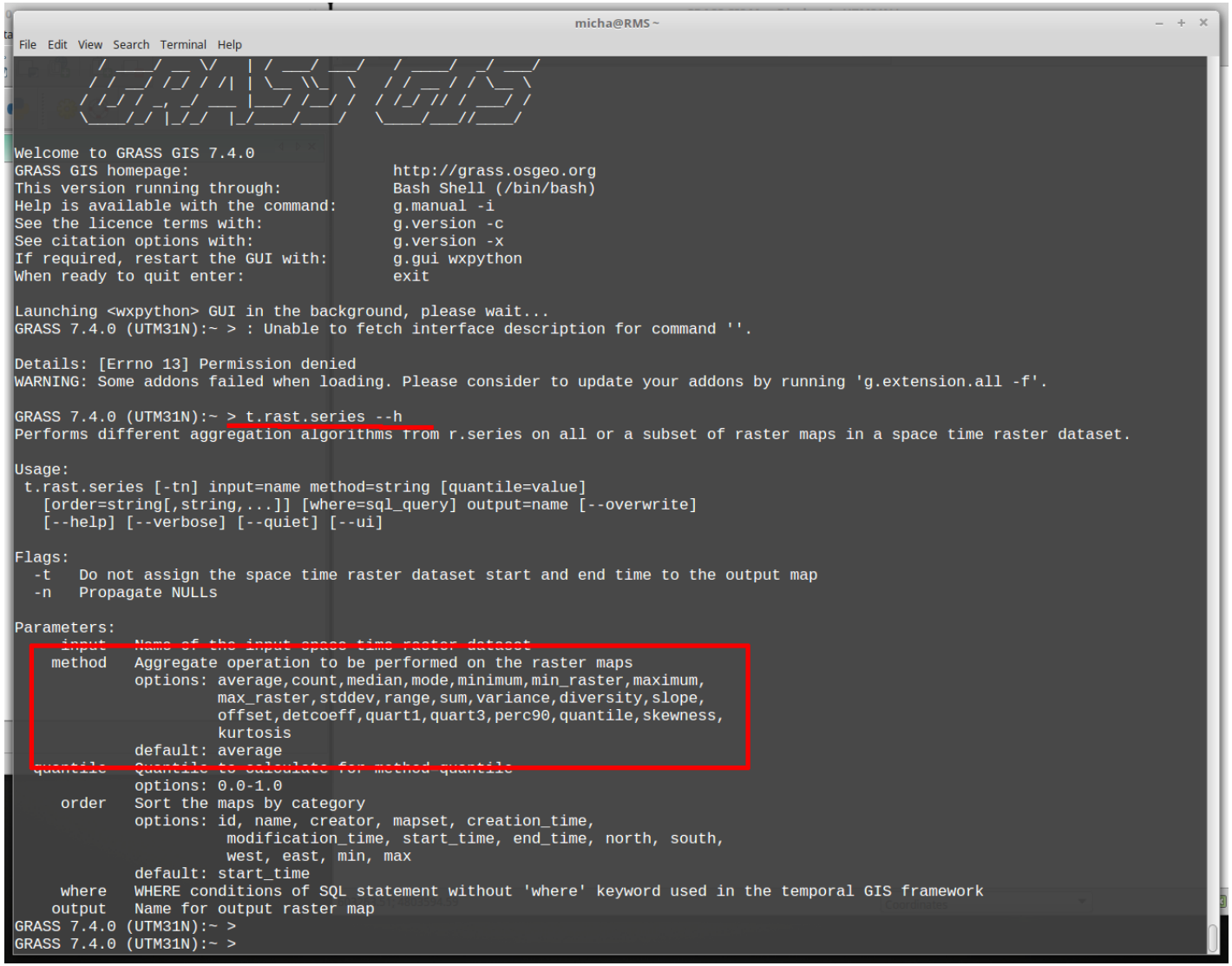
Getting help

Built-in manuals and help



Getting help

Built-in manuals and help

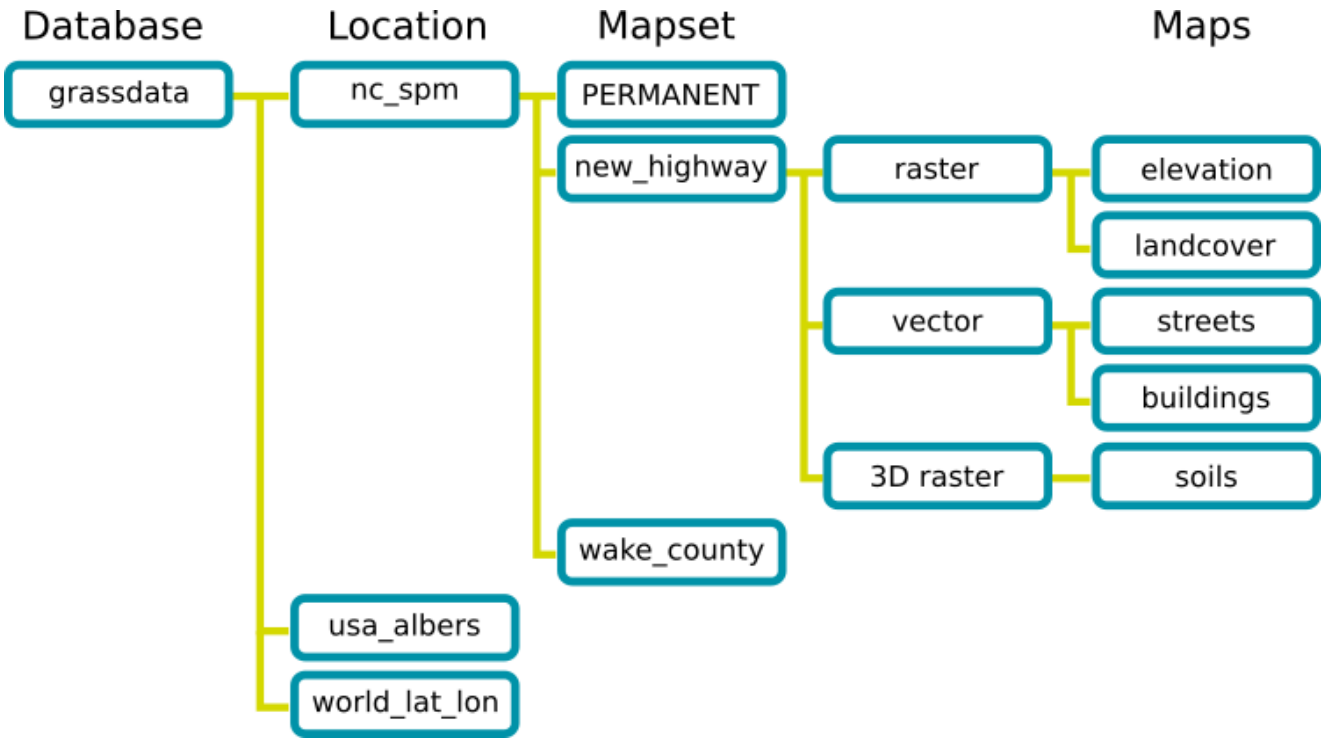


GRASS - Initial setup

- GRASS **Database**
  - **Location** defined by coordinate reference system
  - **Mapset** contains data for a project
- Always check and set **computational region**

GRASS - Initial setup

- GRASS **Database**
  - **Location** defined by coordinate reference system
  - **Mapset** contains data for a project
- Always check and set **computational region**



GRASS - Initial setup

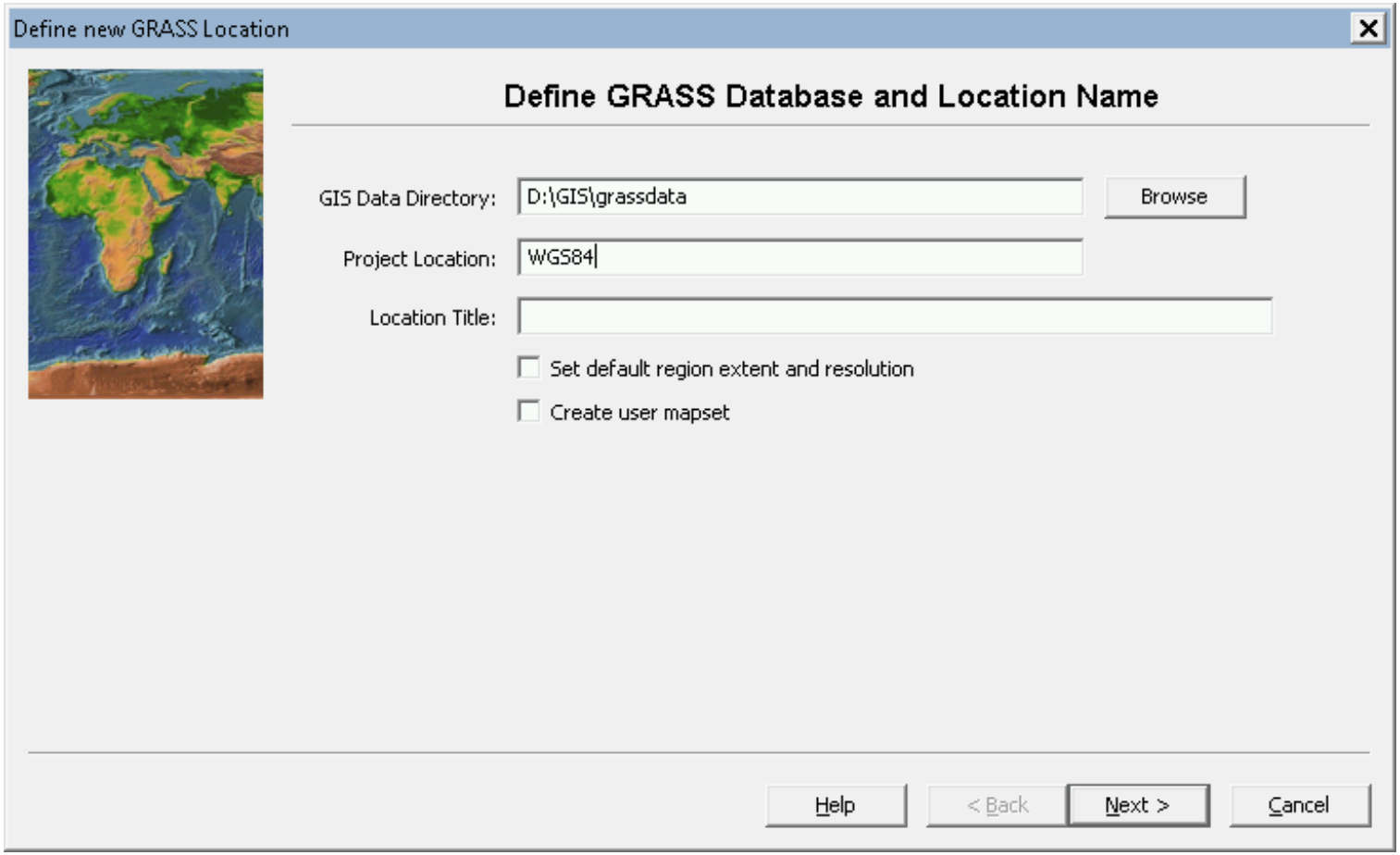
- GRASS **Database**
  - **Location** defined by coordinate reference system
  - **Mapset** contains data for a project
- Always check and set **computational region**

Two take-home messages:

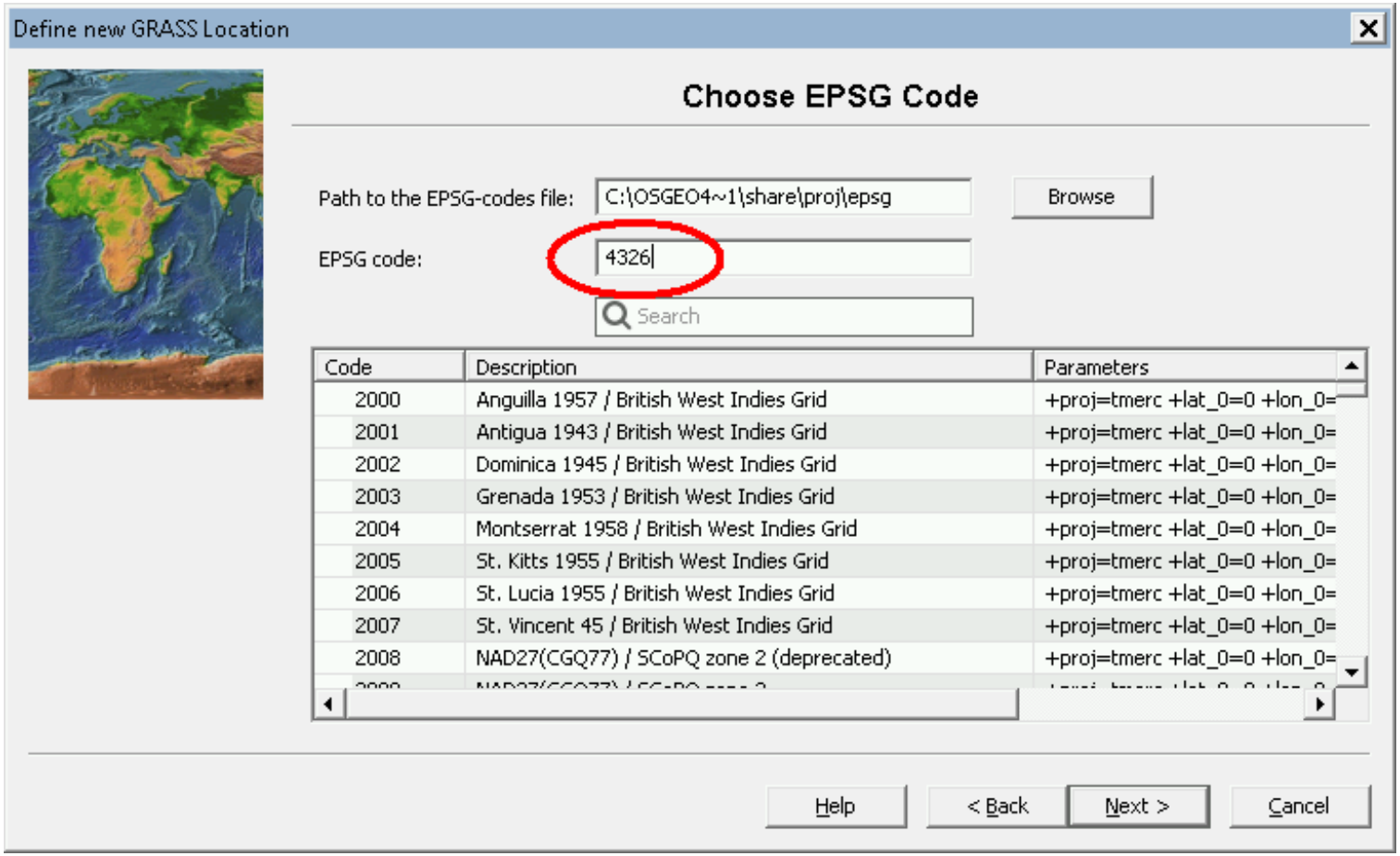
1. **Location** = Coordinate System
2. Set GRASS **region**

GRASS - Initial setup

GRASS - Initial setup

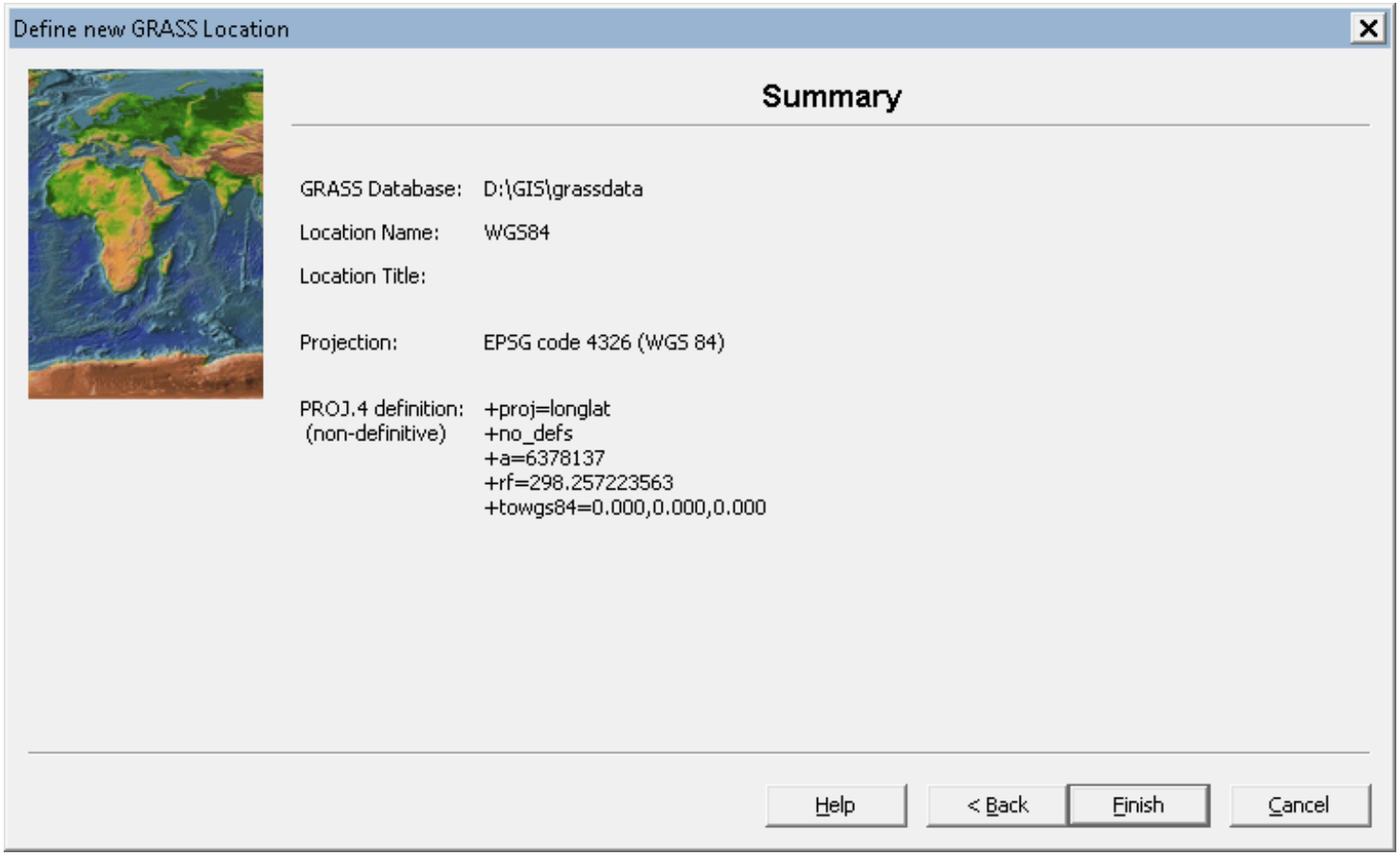


GRASS - Initial setup



Find EPSG codes here: http://epsg.io/

GRASS - Initial setup

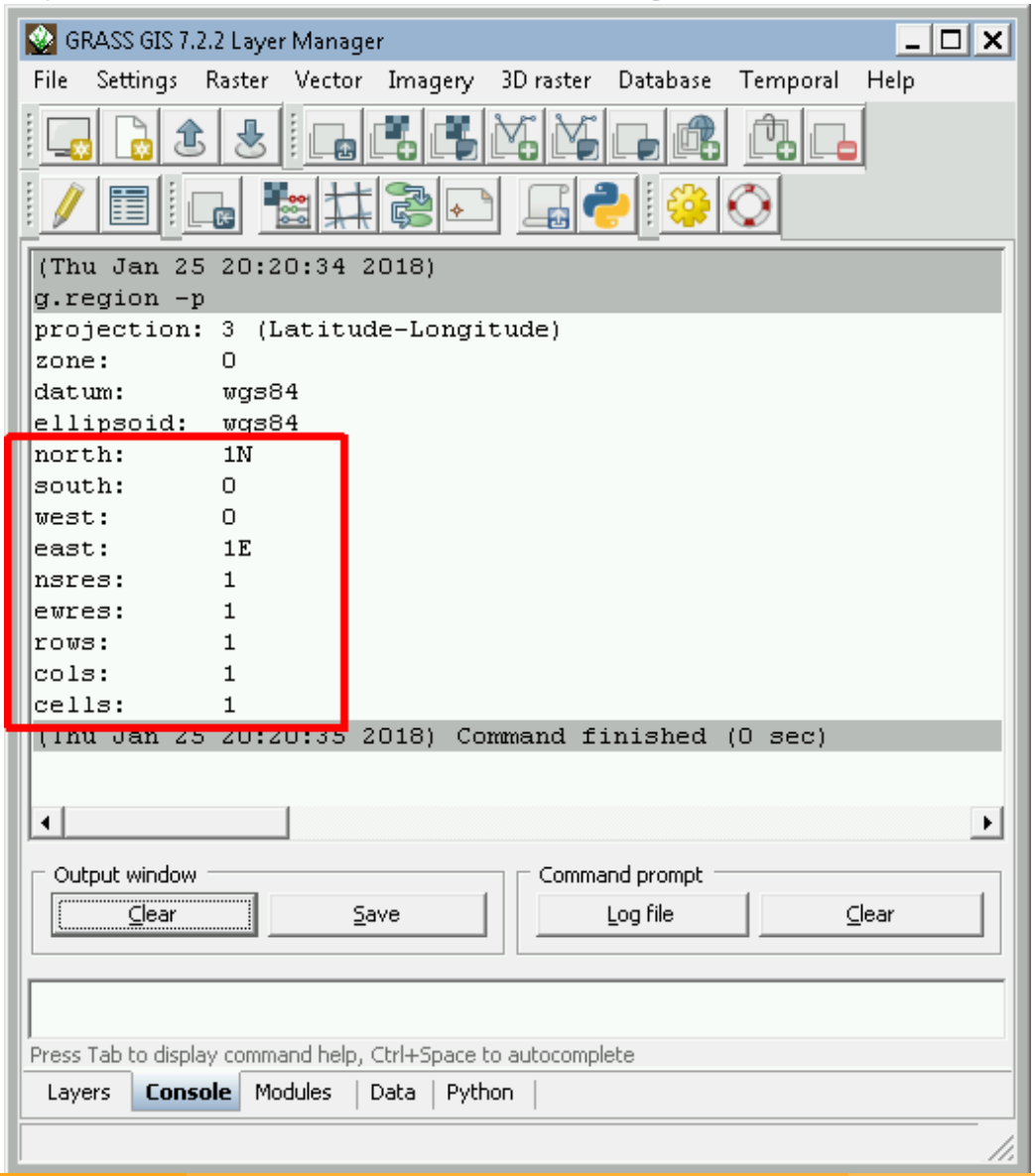


GRASS - Initial setup

GRASS - Initial setup

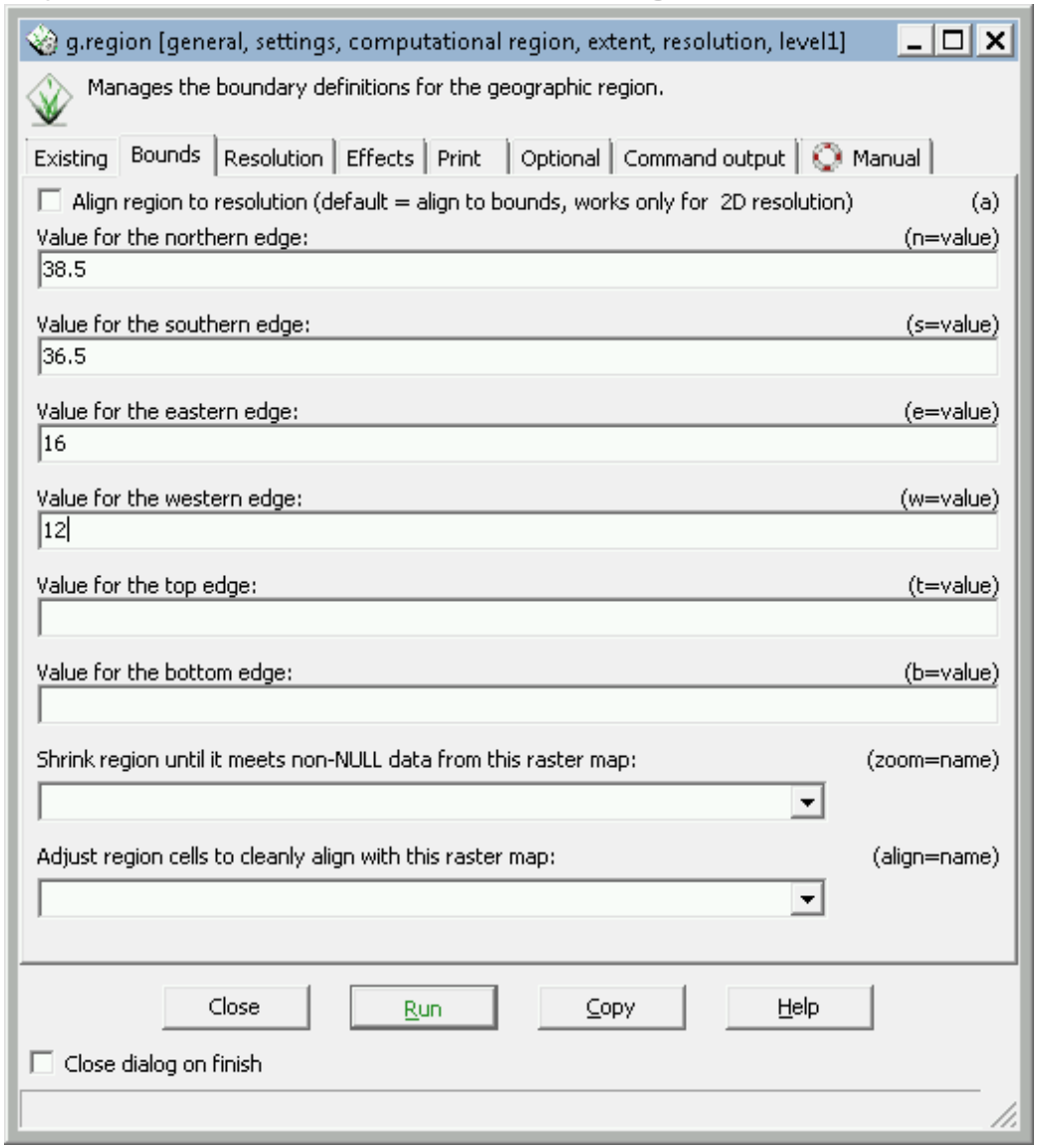
GRASS - Initial setup

Always set the **computational region**



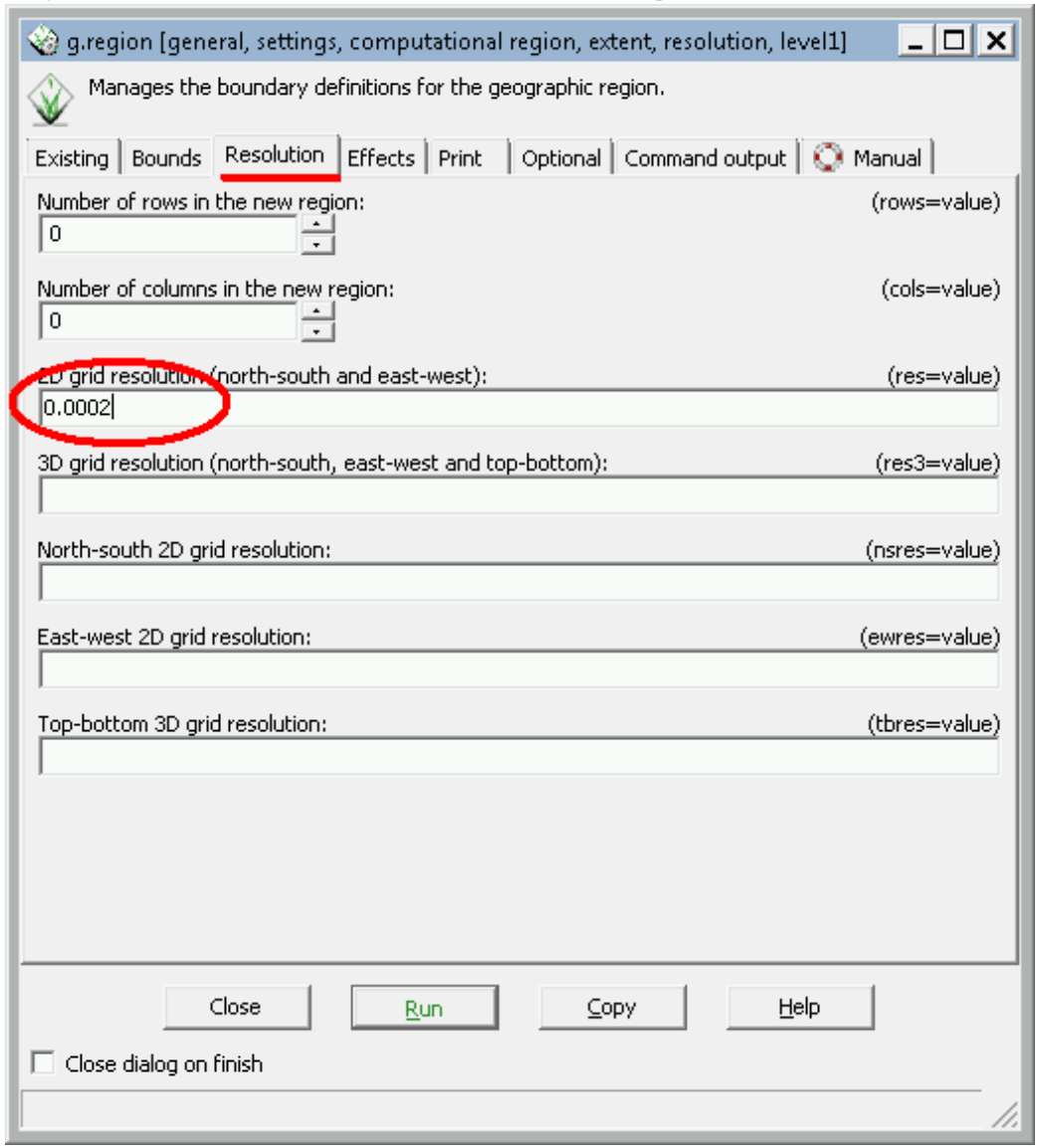
GRASS - Initial setup

Always set the **computational region**

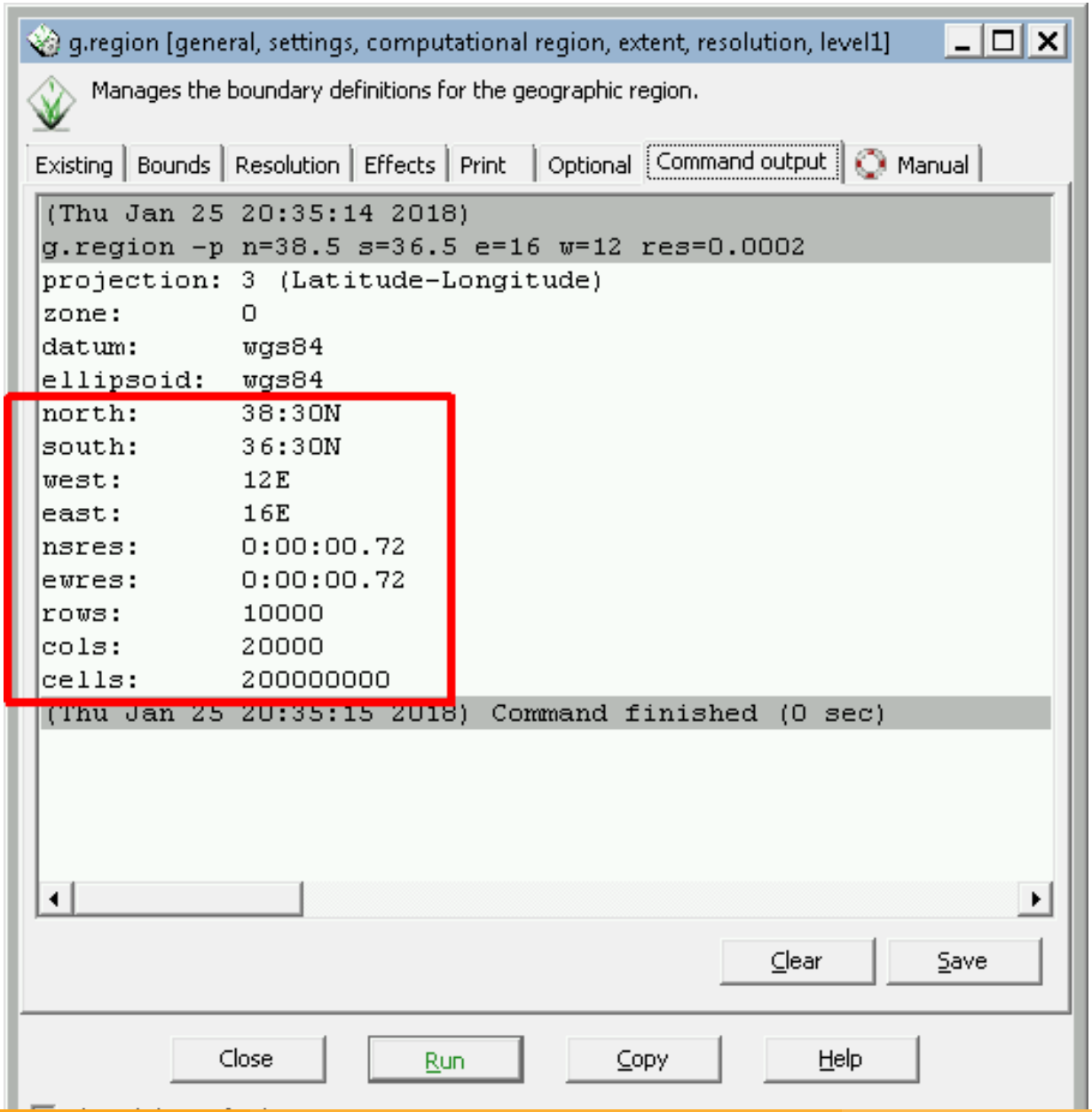


GRASS - Initial setup

Always set the **computational region**



GRASS - Initial setup



Batch Processing in GRASS

Batch Processing

Time series analysis **requires** processing large batches of input

Batch Processing

Language choices:

- Windows command prompt `cmd.exe`
- Linux bash shell script (also on MacOS)

Batch Processing

Language choices:

- Windows command prompt `cmd.exe`
- Linux bash shell script (also on MacOS)
- python (any OS)

Batch Processing

Language choices:

- Windows command prompt `cmd.exe`
- Linux bash shell script (also on MacOS)
- python (any OS)
  - Python scripting in GRASS session:

```
import grass.script as gscript
```

GRASS - Batch import

GRASS command names:

- general commands: **g.***
- raster commands: **r.***
- vector commands **v.***
- temporal commands **t.***

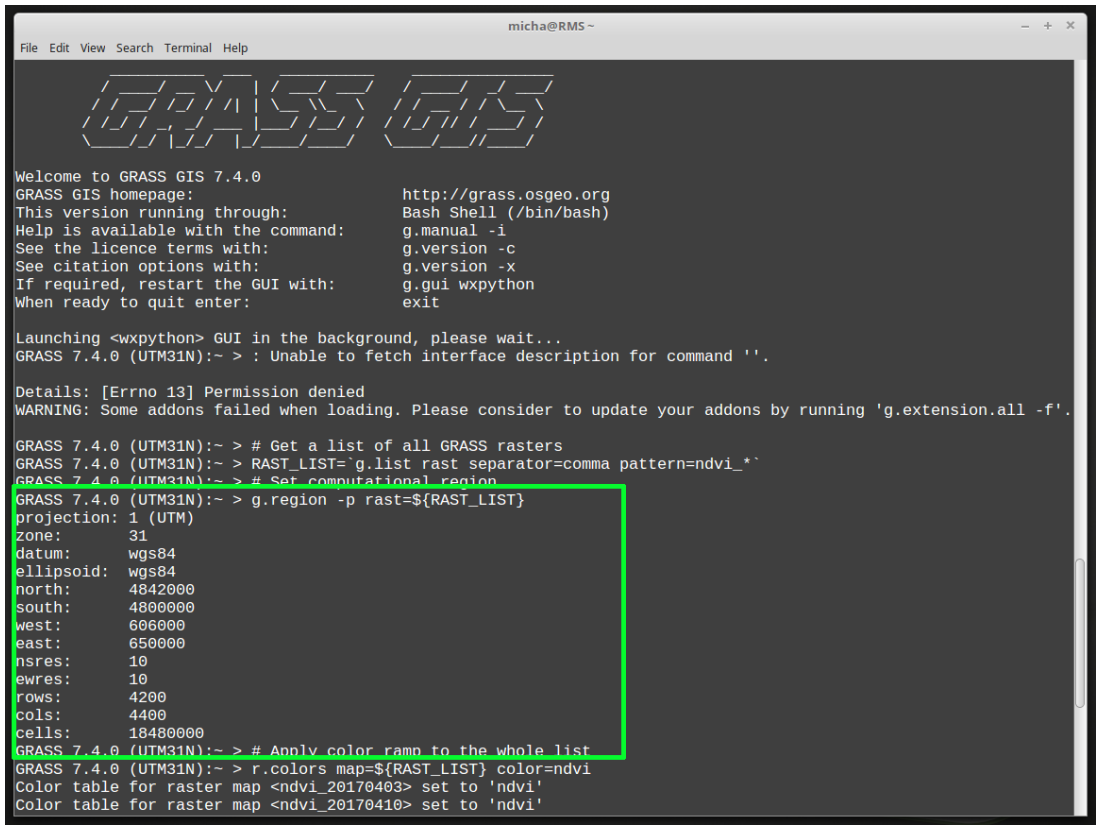
Start GRASS pointing to the LOCATION/MAPSET that we created:

```
# The location/mapset was already created
GRASS_DIR="${BASE_DIR}/grassdata/UTM31N/camargue"
cd ${TARGET_DIR}
# Start grass
grass $GRASS_DIR

# All following commands are entered at GRASS terminal
# Import NDVI rasters
for f in 20*.tif; do
        r=`basename $f .tif`
        r.in.gdal input=$f output="ndvi_${r}" band=1
done
```

GRASS - Set Computational Region

```
# Get a list of all GRASS rasters
RAST_LIST=`g.list rast separator=comma pattern=ndvi_*`
# Set computational region
g.region -p rast=${RAST_LIST}
# Apply color ramp to the whole list
r.colors map=${RAST_LIST} color=ndvi
```



Spatial-Temporal Raster Datasets

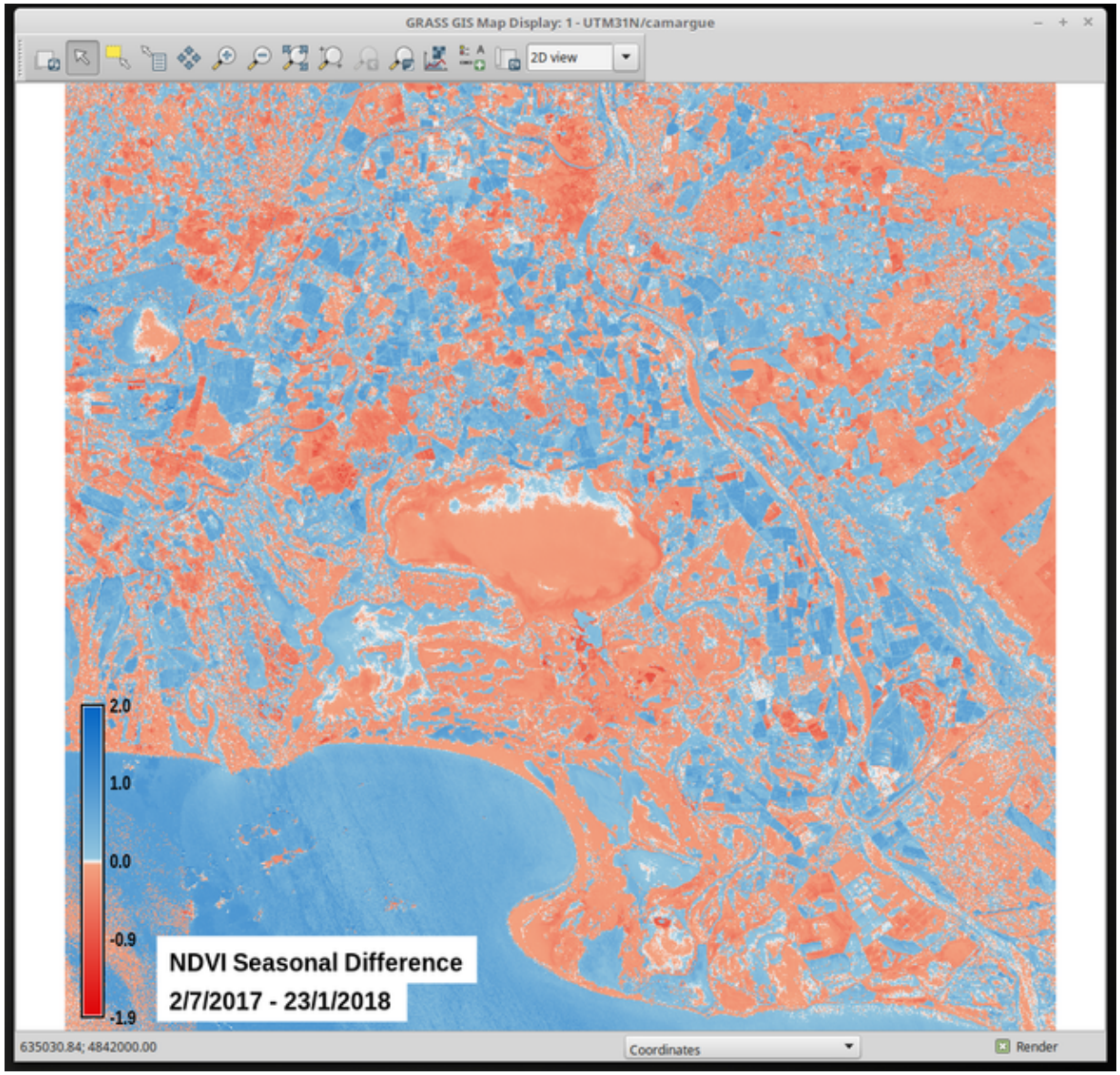
GRASS - Seasonal Difference in NDVI

Use map calculator to get diff between NDVI from two dates

```
# Compare Winter/Summer difference
r.mapcalc "season_diff = 1.0*(ndvi_20170702 - ndvi_20180123)"
r.colors season_diff color=${CODE_DIR}/diff_ramp.txt
```

Note: multiply by **1.0** to force REAL number, otherwise an INTEGER raster is created

GRASS - Seasonal Difference in NDVI



GRASS - Spatial Temporal Raster DataSet

Create a new, empty Spatial-Temporal Raster Dataset (**strds**)

```
# Create and populate space-time raster dataset
t.create output=camargue_ndvi type=strds
  temporaltype=absolute title="camargue NDVI" \
  description="camargue NDVI time series"
```

and populate with all NDVI rasters, using bash **date** manipulation to set start and end times

```
for r in `g.list rast pattern=ndvi_*`; do
        y=${r:5:4}
        m=${r:9:2}
        d=${r:11:2}
        s_dt=$y-$m-$d
        e_dt=`date -d "$s_dt + 1 days" +"%Y-%m-%d"`
        t.register input=camargue_ndvi maps=$r \
        start=$s_dt end=$e_dt --overwrite
done
```

GRASS - Check STRDS

Set color ramp, and check info

```
t.rast.colors camargue_ndvi color=ndvi
t.info camargue_ndvi
t.rast.list camargue_ndvi colu=id,start_time,end_time
```

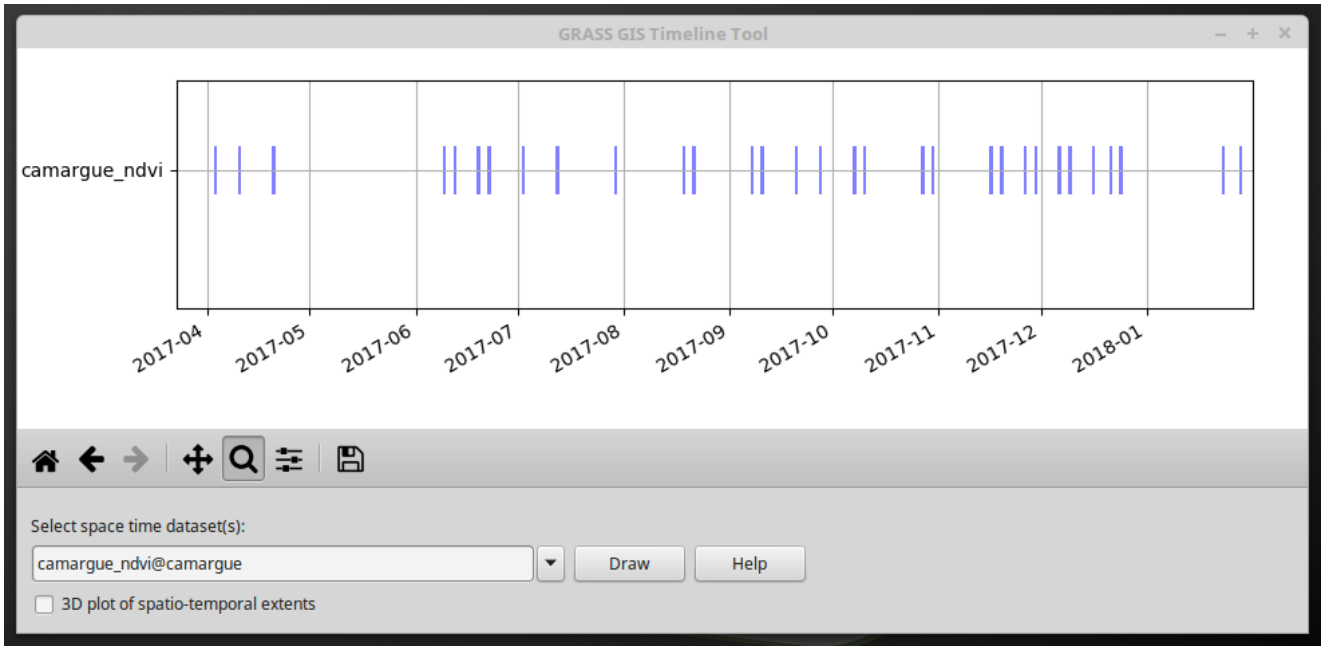
GRASS - Check STRDS

Set color ramp, and check info

```
t.rast.colors camargue_ndvi color=ndvi
t.info camargue_ndvi
t.rast.list camargue_ndvi colu=id,start_time,end_time
```

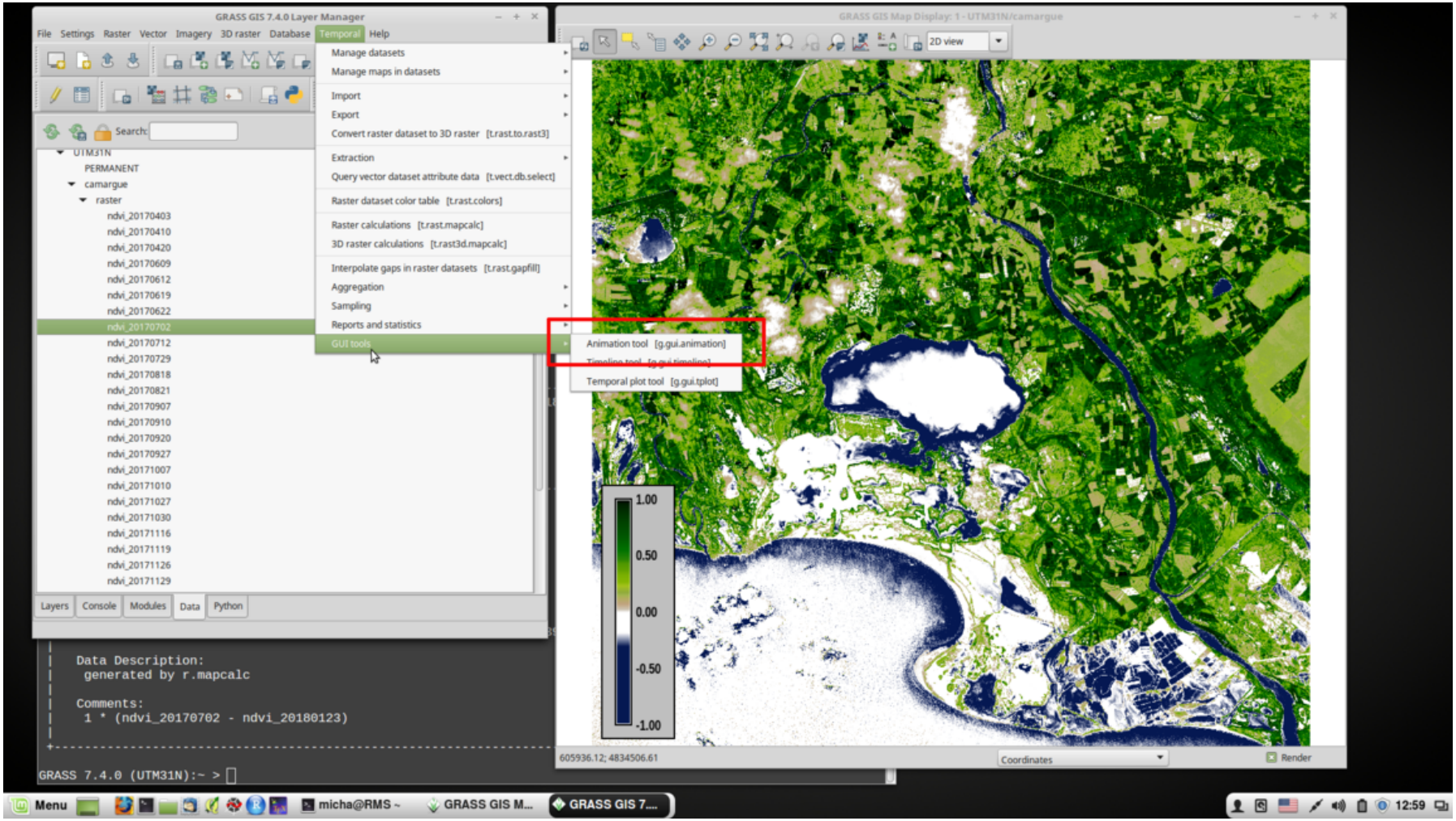
View the Timeline

```
g.gui.timeline strds=camargue_ndvi
g.gui.animation strds=camargue_ndvi
```



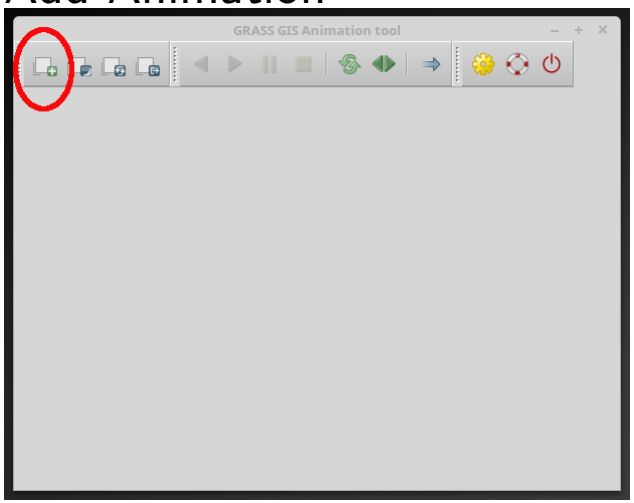
GRASS - STRDS Animation

Show STRDS as animation

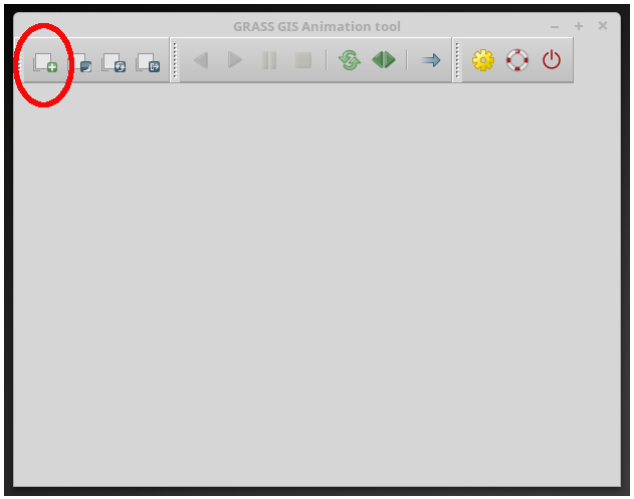


GRASS - STRDS Animation

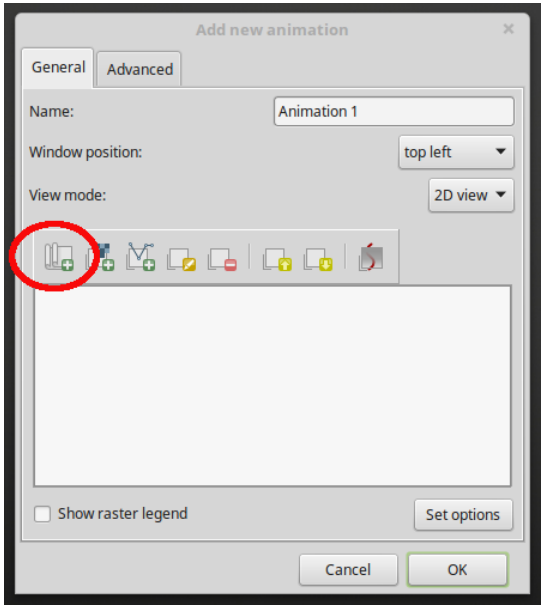
Add Animation



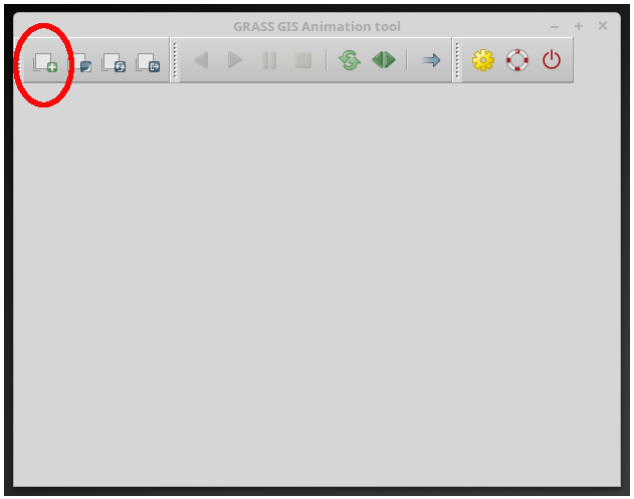
GRASS - STRDS Animation

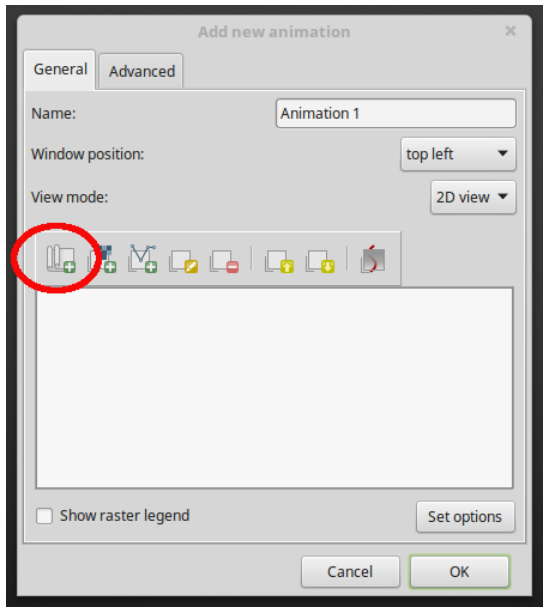


Add STRDS to animation

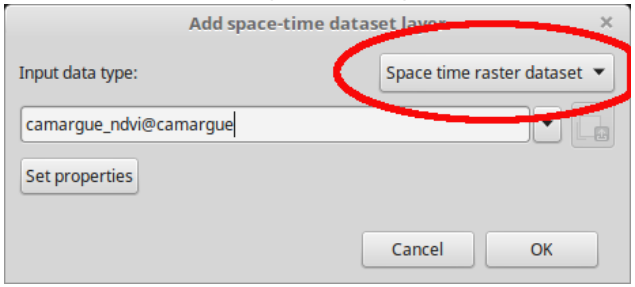


GRASS - STRDS Animation

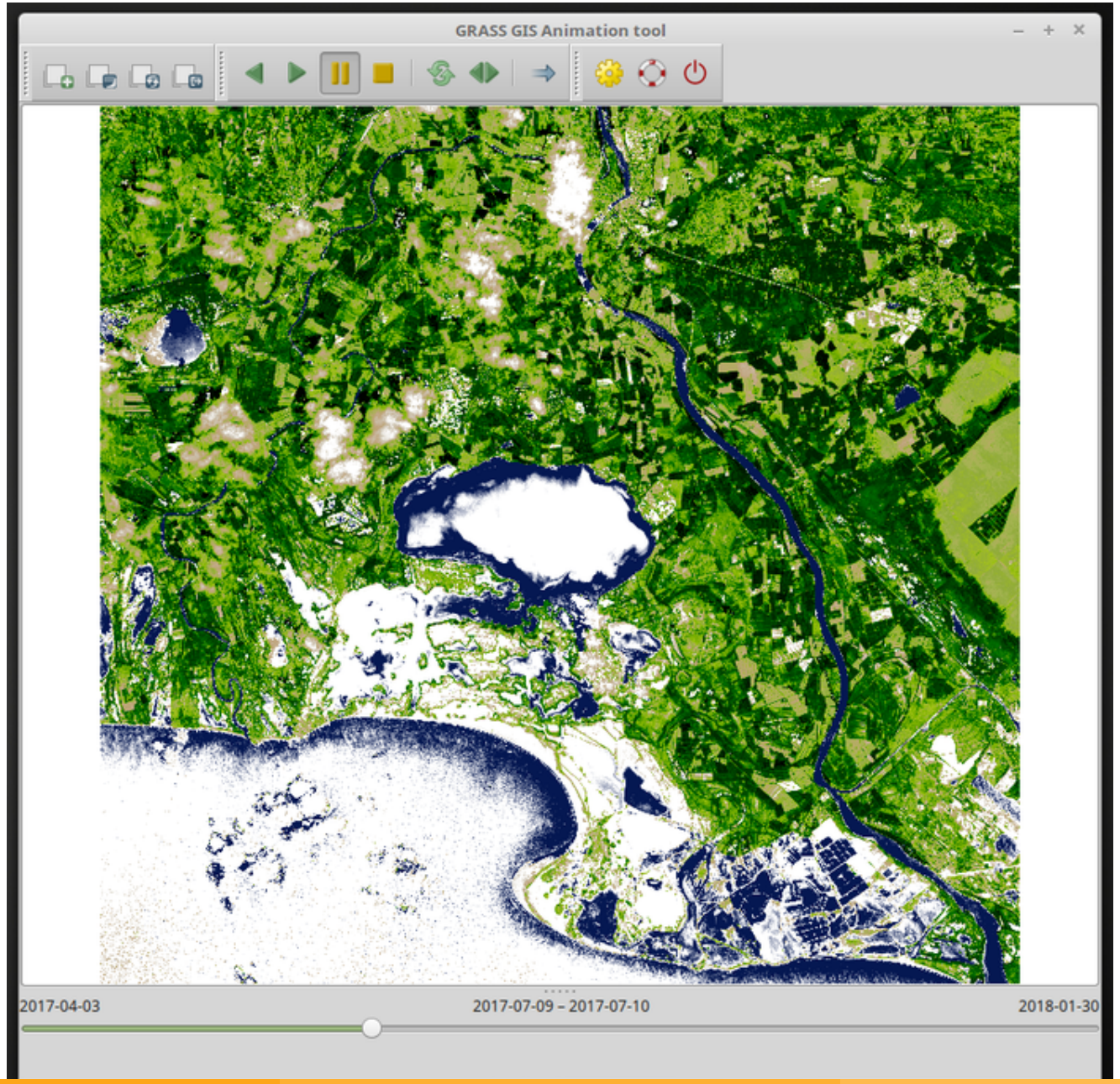




Choose the STRDS

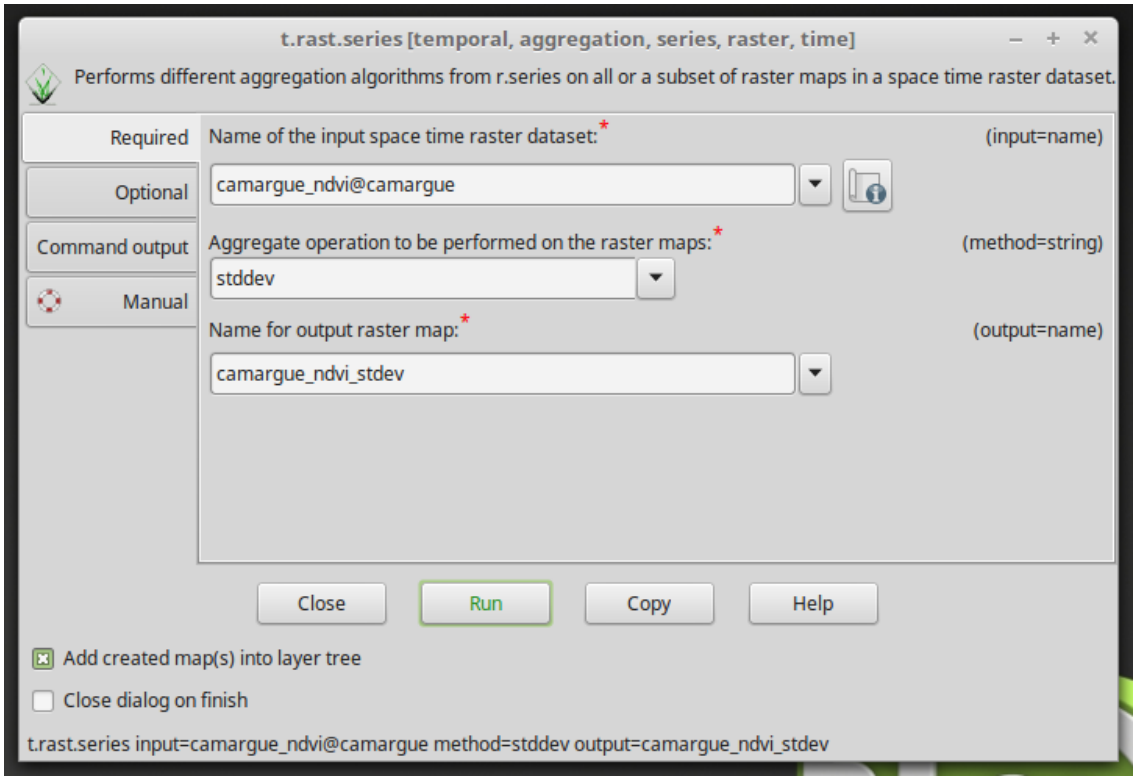


GRASS - Animation Result



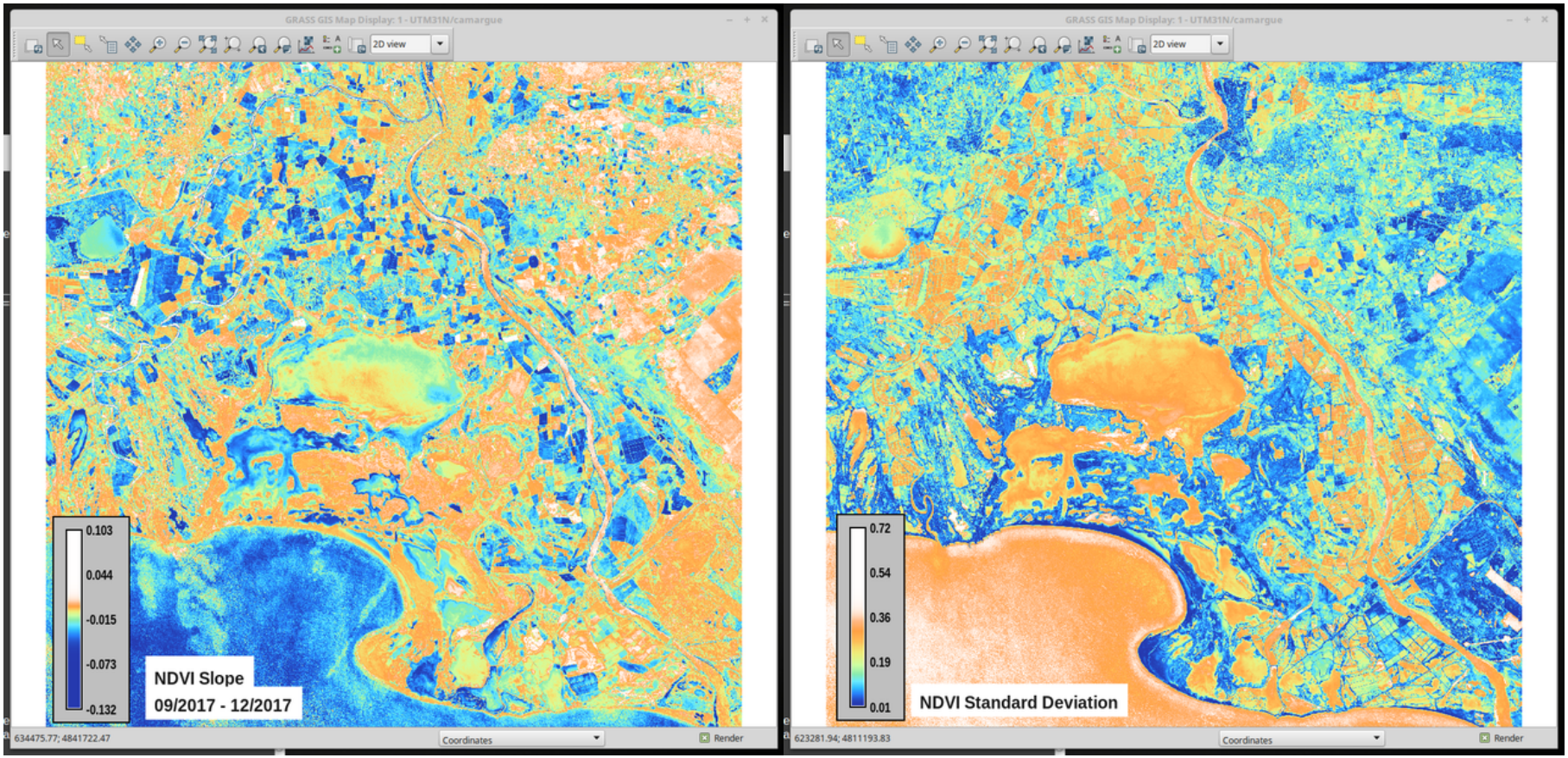
GRASS - STRDS Aggregation

Module: `t.rast.series`




```
# t.rast.series example
t.rast.series input=camargue_ndvi@camargue method=slope \
  where="start_time>'2017-09-01 00:00' and \
  start_time<'2017-12-31" output=camargue_ndvi_slope_fall
```

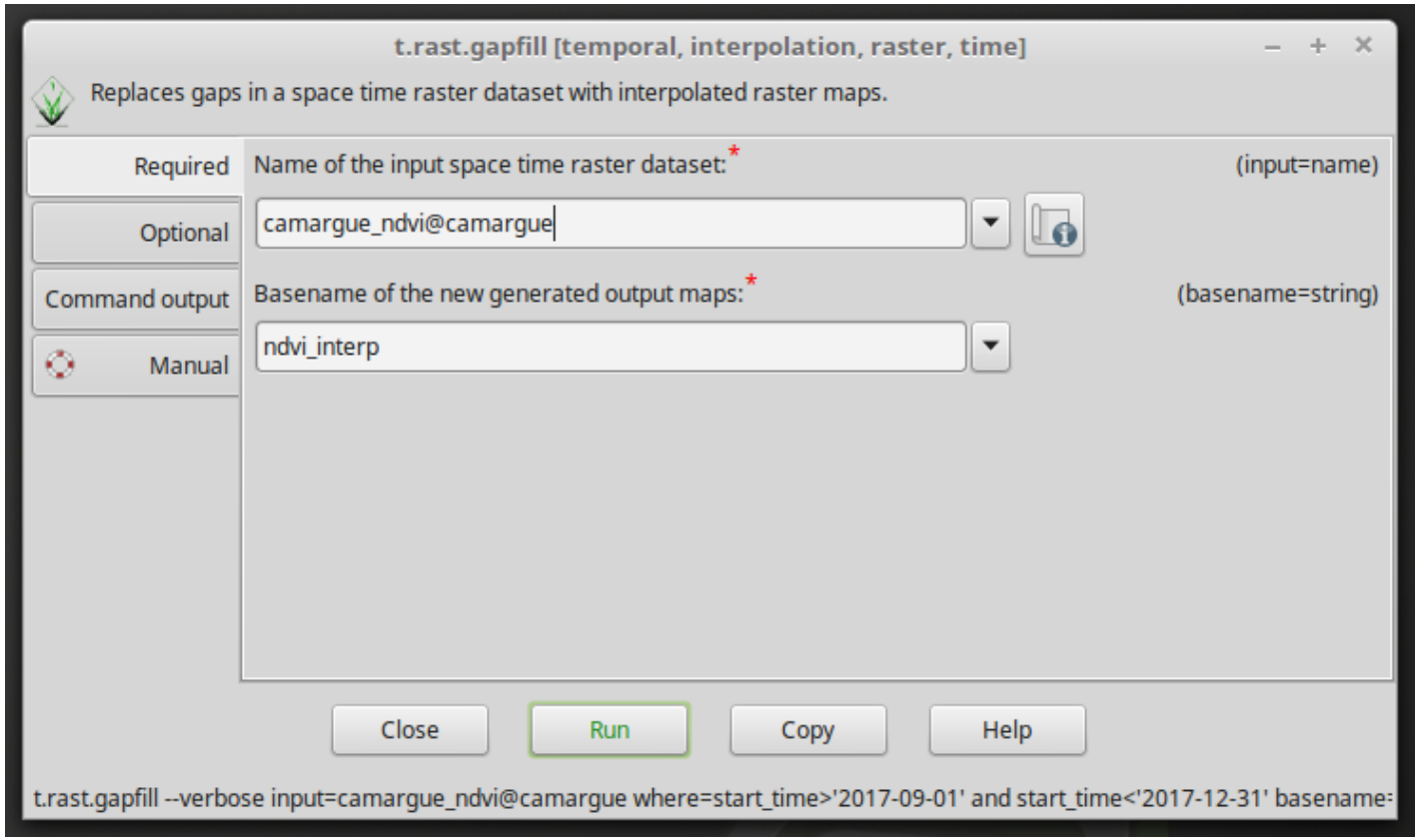
GRASS - STRDS Aggregation



GRASS Interpolate Missing Time slots

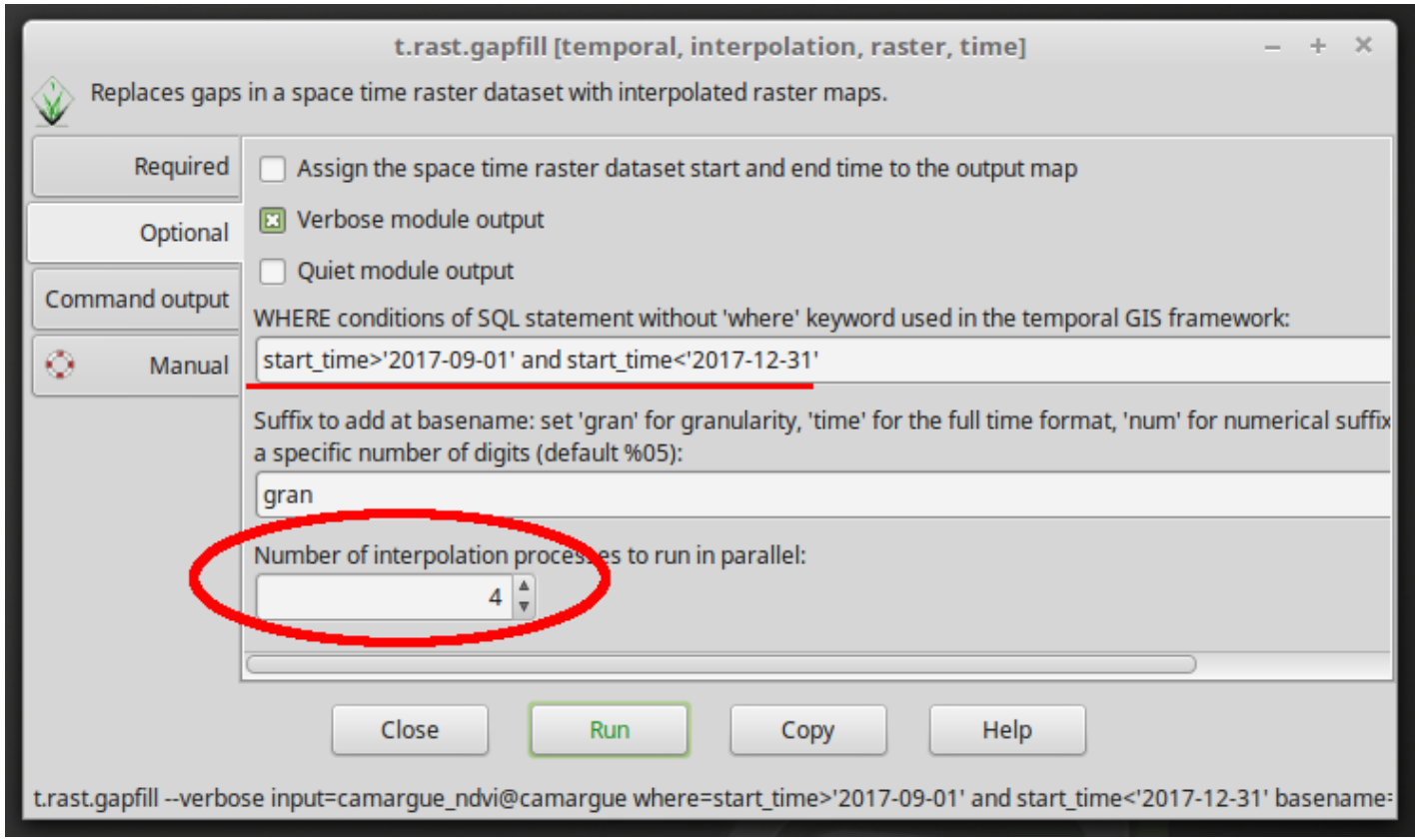
Linear interpolation between dates

Module: `t.rast.gapfill`



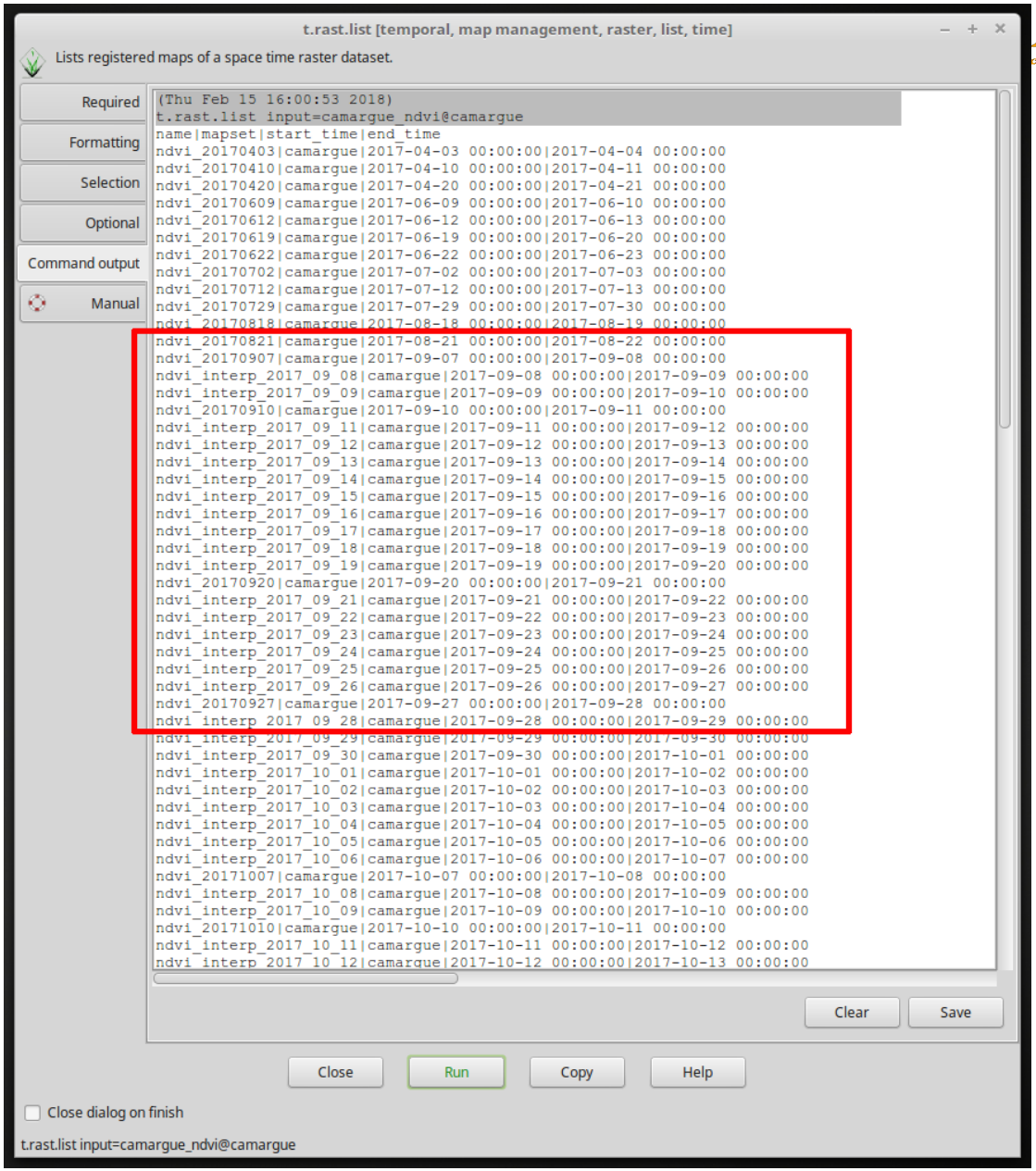
GRASS Interpolate Missing Time slots

Linear interpolation between dates
Module: `t.rast.gapfill`



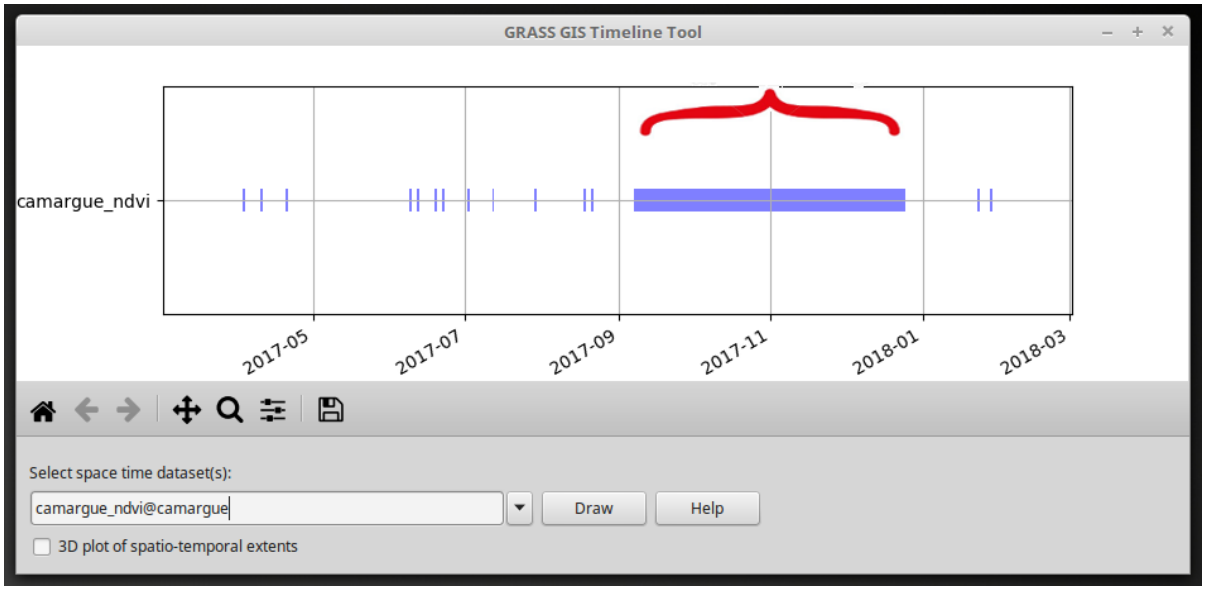
GRASS Interpolate Missing Time slots

List of maps in STRDS, including interpolated



GRASS Interpolate Missing Time slots

List of maps in STRDS, including interpolated



Thanks