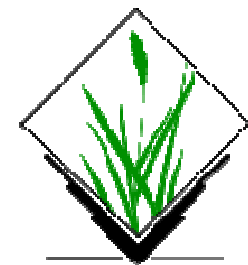




# GRASS GIS intro

Geographic Information Systems (GIS)

Sara Lucca  
Luana Valentini



# What's GRASS

- Free and Open source GIS
- **G**eographic **R**esources **A**nalysis **S**upport **S**ystem

- Portable:

- GNU/Linux
- Mac-OSX
- ...



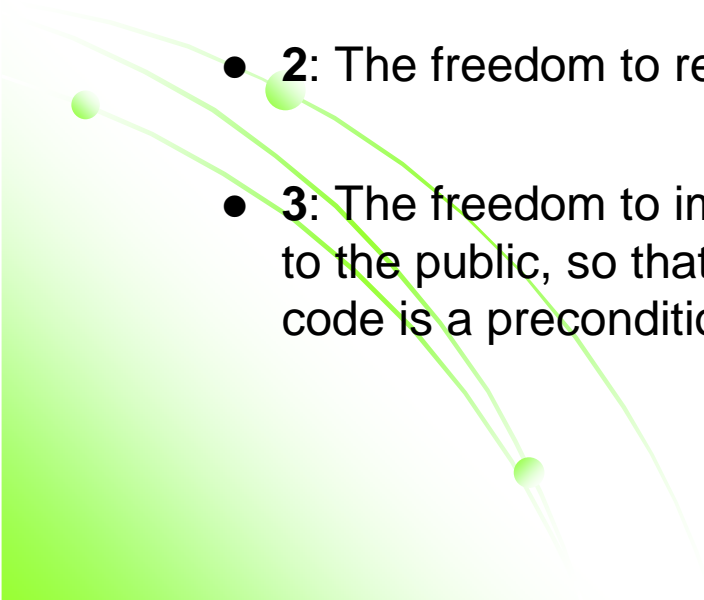
MS-Windows

- Written in: **C/C++**

Makes also use of Tcl/tk, Python

- GRASS main web site: <http://grass.osgeo.org>

# What's GRASS - 2

- **Free and Open Source Software (FOSS):**
    - **0:** The freedom to run the program, for any purpose.
    - **1:** The freedom to study how the program works, and adapt it to your needs. Access to the source code is a precondition for this.
    - **2:** The freedom to redistribute copies so you can help your neighbor
    - **3:** The freedom to improve the program, and release your improvements to the public, so that the whole community benefits. Access to the source code is a precondition for this.
- 
- A decorative graphic in the bottom-left corner consisting of several curved green lines and dots of varying opacity, creating a sense of motion or a stylized path.

# Interoperability

- Proj.4: Projections
- OGR/GDAL: data conversion libraries
- Database:

- PostgreSQL/PostGIS



- MySQL



- SQLite

- ODBC: PostgreSQL, MySQL, ORACLE, MS-Access, MS-SQL Server, ecc.



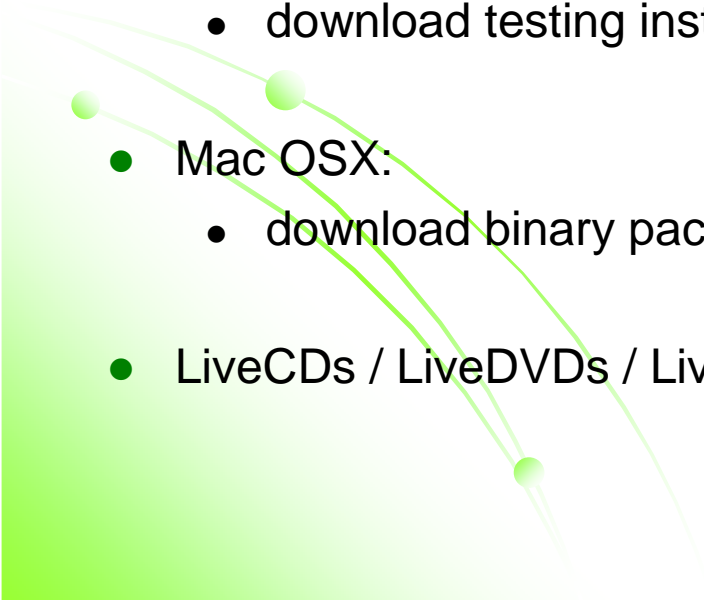
- QGIS

- R-stats



# How to obtain GRASS?

<http://grass.osgeo.org/download/index.php>

- LINUX:
    - download source code and compile it
    - download binary packages (generic or distribution-oriented)
  - WINDOWS:
    - download stable binaries (Cygwin)
    - download testing installer
  - Mac OSX:
    - download binary packages
  - LiveCDs / LiveDVDs / LiveUSBs
- 

# How to obtain GRASS? - 2

- As for LiveUSBs:

**slaxGIS**

<http://geomatica.como.polimi.it/software/slaxGIS/>



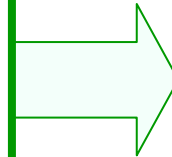
Complete Linux system bootable from a USB flash drive, containing (V090731B ):

- grass-6.4.svn\_2009\_10\_31
- - MapServer 5.4.2
- OpenLayers 2.8
- uDig 1.1.1
- OpenJUMP PIROL 1.2.F
- PostgreSQL 8.4.0
- PostGIS 1.3.6

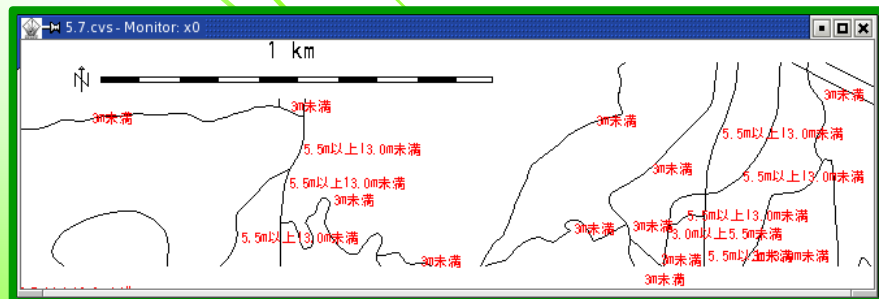
# GRASS i18N

“Official” language: *english*

Translations into **19 languages** for the user interface are currently ongoing.



```
neteler@s3ad197.media.osaka-cu.ac.jp: /home/neteler
GRASS 5.7.cvs:~ } echo "SELECT * FROM dourokukan" | db.select
cat|label|forward|backward|length
1|3m未満|50.000000|50.000000|
3|3.0m以上5.5m未満|50.000000|50.000000|
5|5.5m以上13.0m未満|50.000000|50.000000|
13|13.0m以上20.000000|50.000000|
15|真幅道路等|50.000000|50.000000|
GRASS 5.7.cvs:~ }
```

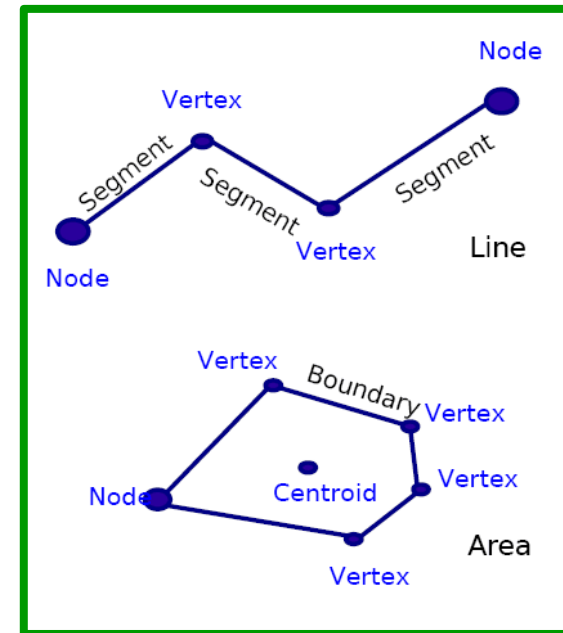


PO-Files		Translated messages	Fuzzy translations	Untranslated messages
Czech	grasstcl_cs.po	1429		
Thai	grasstcl_th.po	1424	6	1
Spanish	grasstcl_es.po	1422	2	5
Portuguese	grasstcl_pt_br.po	1361	39	31
Italian	grasstcl_it.po	1342	310	158
Portuguese	grasstcl_pt.po	1191	162	78
Turkish	grasstcl_tr.po	1126	3	302
French	grasstcl_fr.po	794	268	369
German	grasstcl_de.po	765	399	265
Polish	grasstcl_pl.po	732	375	324
Vietnamese	grasstcl_vi.po	726	420	285
Chinese	grasstcl_zh.po	221	15	1195
Amharic	grasstcl_am.po	174	80	1177
Slovenian	grasstcl_sl.po	0	1431	
Russian	grasstcl_ru.po	0	187	1244
Korean	grasstcl_ko.po	0	1431	
Greek, Modern (1453-)	grasstcl_el.po	0	1431	
Arabic	grasstcl_ar.po	0	1431	
Hindi	grasstcl_hi.po	0	1431	
Japanese	grasstcl_ja.po	0	667	764
Latvian	grasstcl_lv.po	0	125	1306
Marathi	grasstcl_mr.po	0	1431	

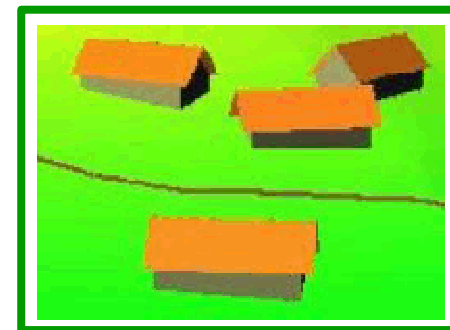
# Geometry and topology

- Geometry is true 3D (x, y, z):
  - Point
  - Centroid
  - Line
  - Boundary
  - $\text{Area}^* = \text{boundary} + \text{centroid}$
  - $\text{Face}^* = 3\text{D area}$
  - $\text{Kernel}^* = 3\text{D centroid}$
  - $\text{Volume}^* = \text{face} + \text{kernel}$

(\*) Not in all GIS!!



Face





# OGC simple features...

## Points, lines, polygons

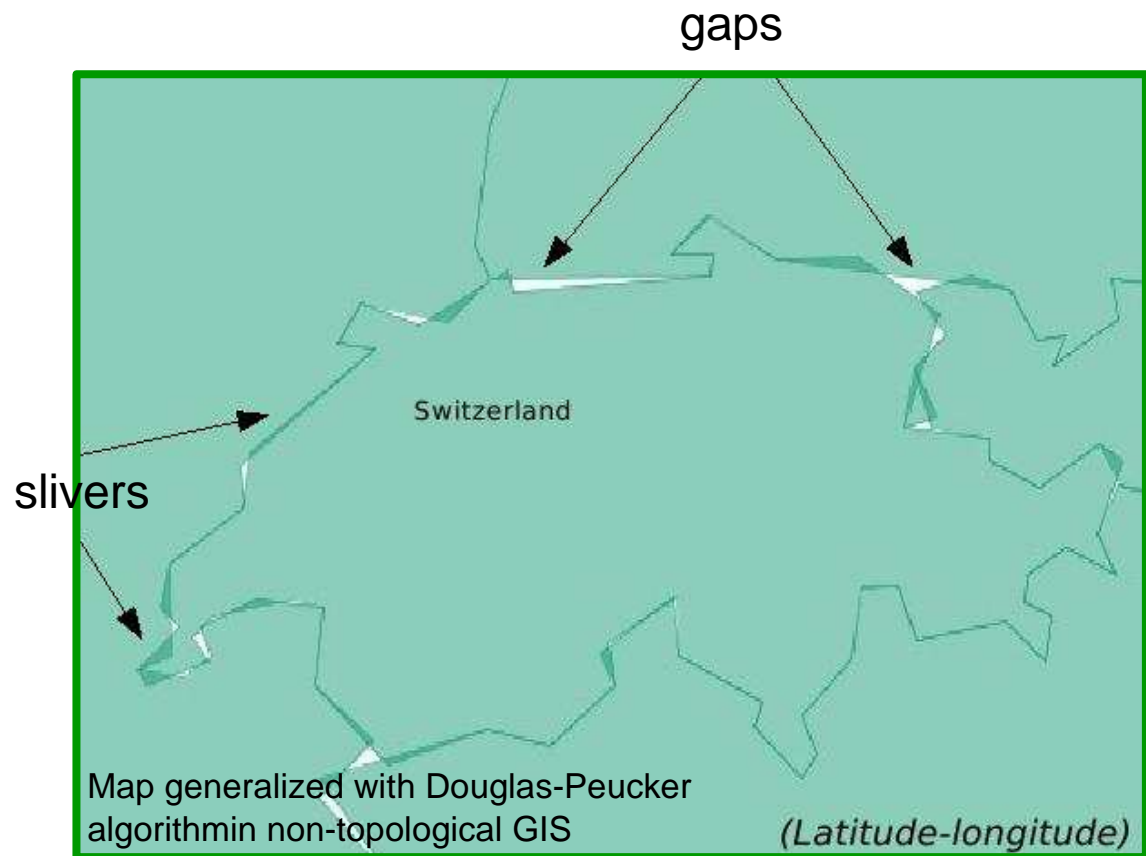
Replicated boundaries for adjacent areas

### Advantages:

- faster computations

### Disadvantages:

- Extra work for data maintenance
- Duplicated boundaries cause problems



# ... VS Vector topology

## Points, centroids, lines, boundaries

In topology centroid and boundary form an area

Single boundaries for adjacent areas

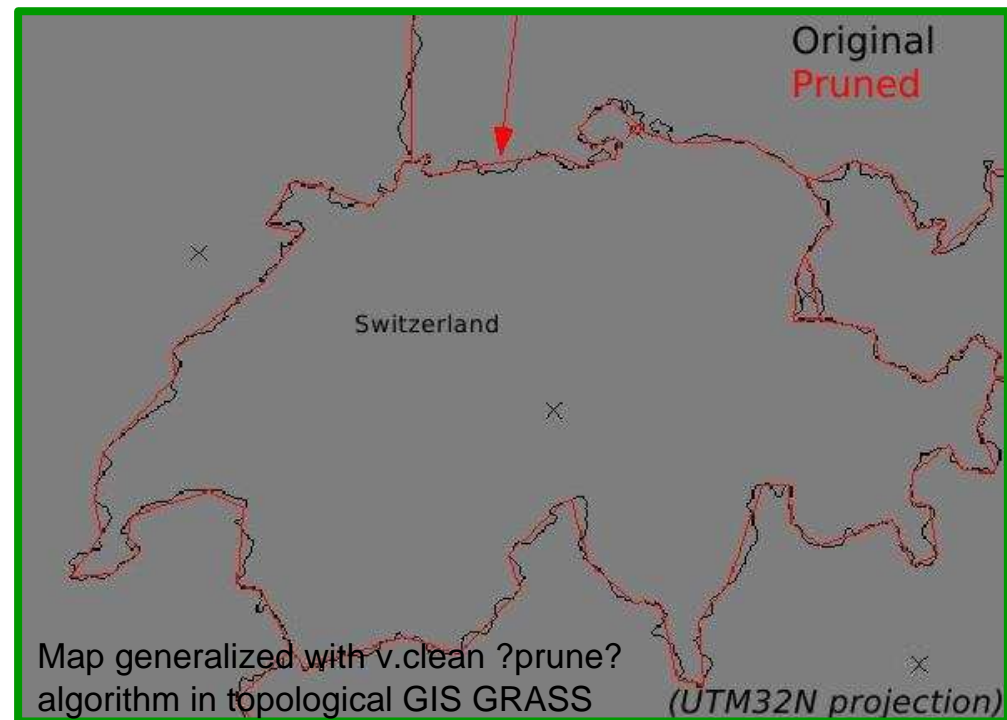
### Advantages:

- less maintenance, high quality

### Disadvantages:

- Slower computations

each boundary is a single line  
divided by two polygons

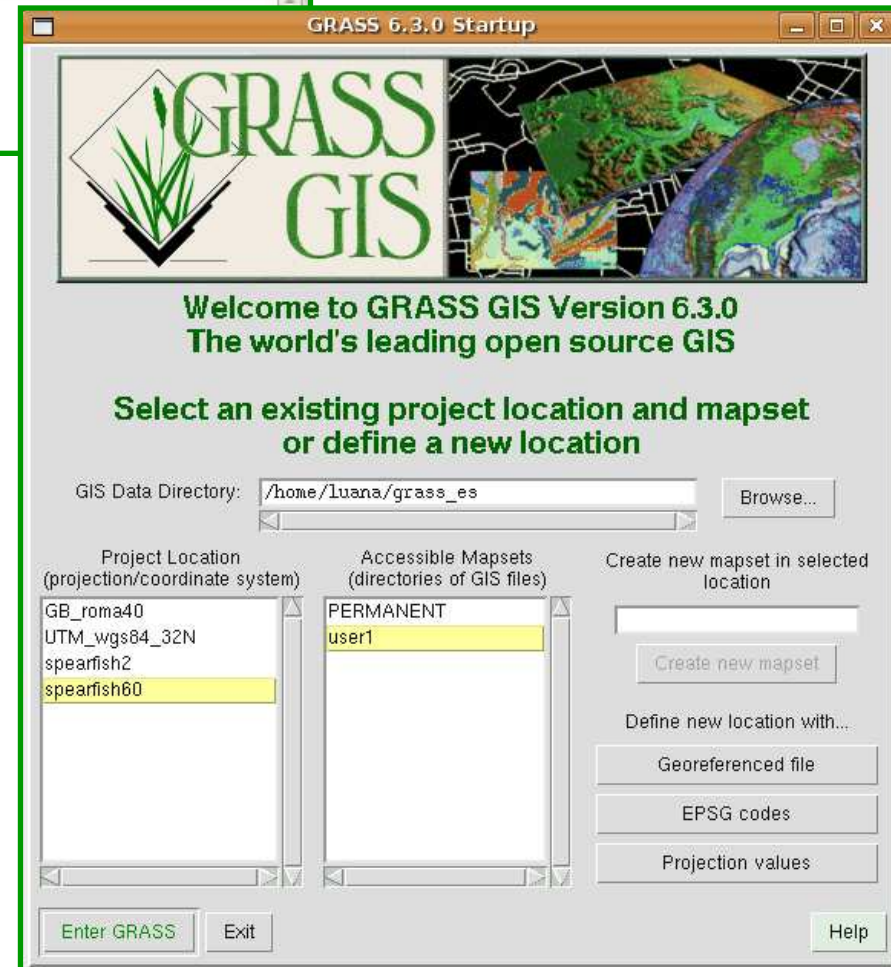


# How to start GRASS

```
luana@luana-laptop: ~  
File Edit View Terminal Tabs Help  
luana@luana-laptop:~$ grass63  
Cleaning up temporary files.....  
Starting GRASS ...  
█
```

You can start Grass typing 'grass63' in the shell.

In the startup window select the mapset and press the 'Enter GRASS' button.



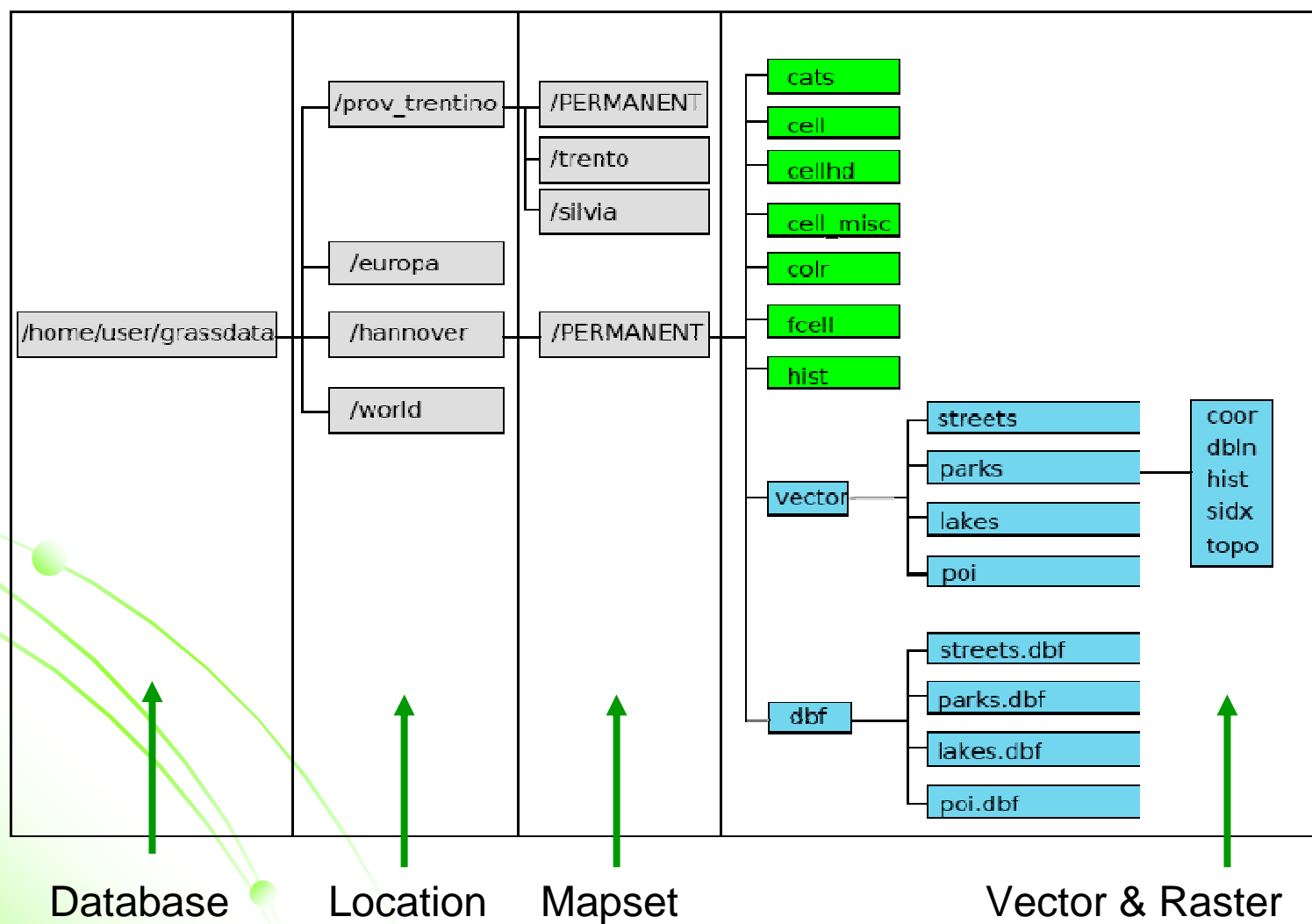
# Data structure

- **DATABASE:** Contains all GRASS data.

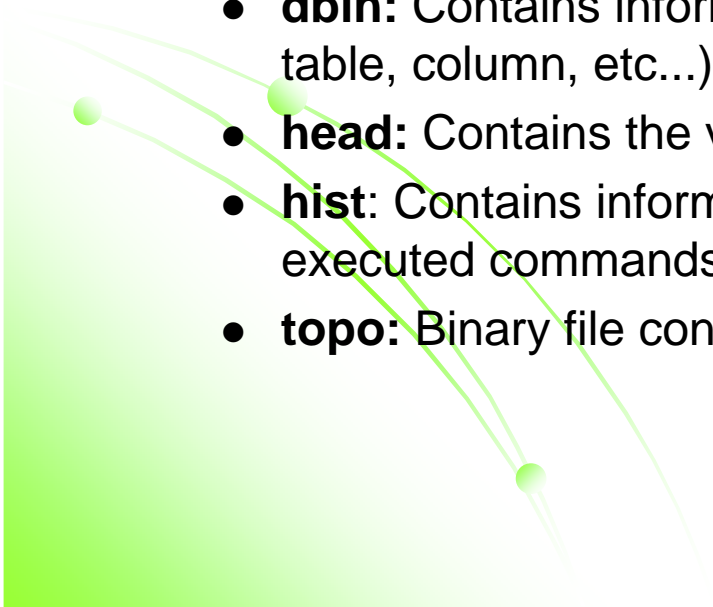
Each GRASS project is organized in a “Location” directory with subsequent “Mapset” directories

- **LOCATION:** Defines a coordinate system and a rectangular boundary for a project
- **MAPSET(s):** Used to subdivide data by user names or subregions or access rights.
  - **PERMANENT:** is a standard mapset that contains the definitions of the location. May also contain general cartography since it is visible to all the other mapsets.
- **Multi-User** support: multiple users can work in a single location using different mapsets. Access rights can be managed per user. No user can modify/delete data of other users.

# Data structure - 2



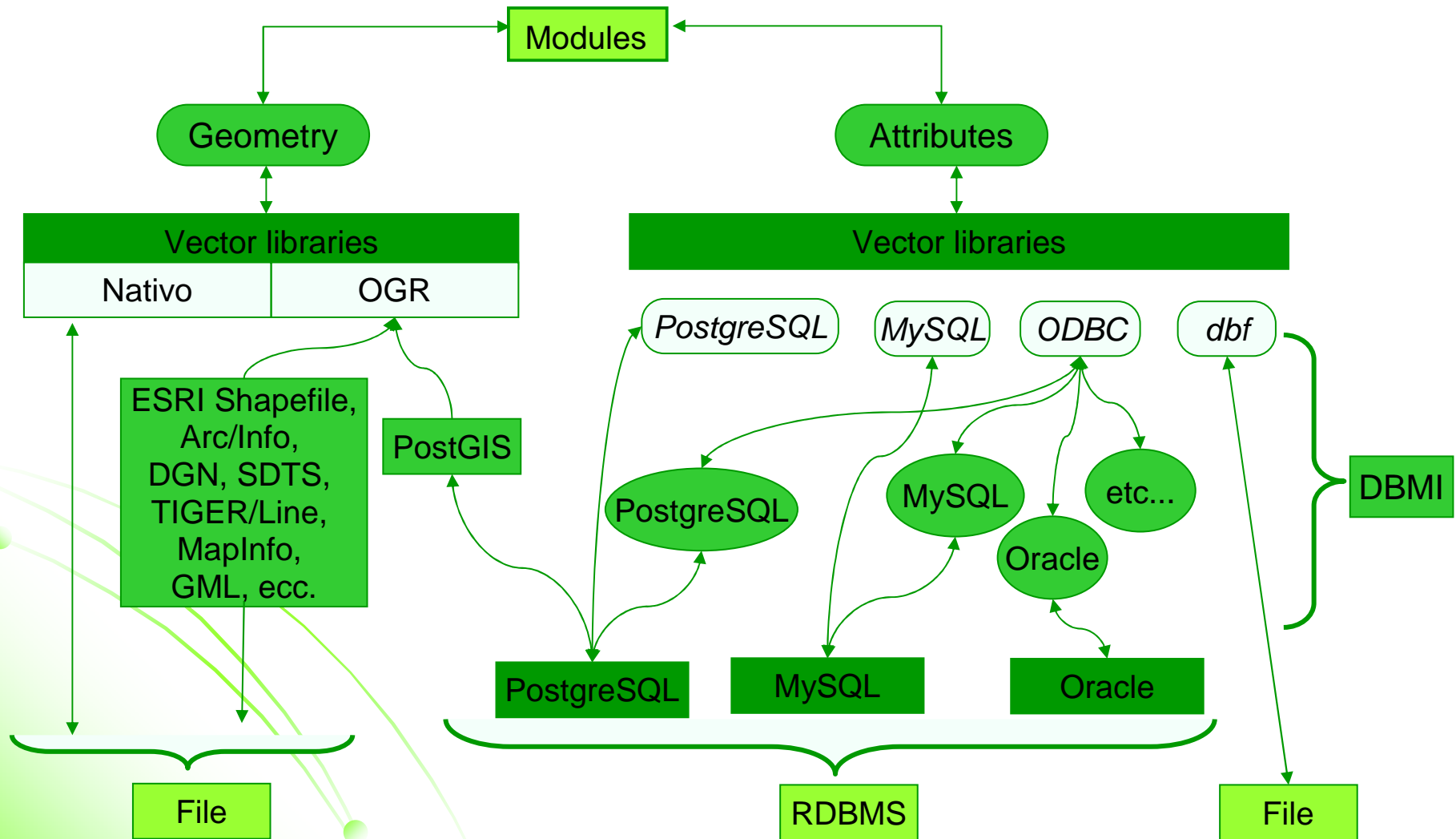
# Vector maps structure

- Each vector map is contained in a single directory (inside the '*vector*' directory) with the same name of the vector map.
  - Each vector directory contains the following files:
    - **cidx**: Contains the topological index
    - **coor**: Binary file that contains the features coordinates and categories
    - **dbln**: Contains information about database connection (driver, database, table, column, etc...)
    - **head**: Contains the vector header
    - **hist**: Contains information about the vector history: author, date, executed commands, etc...
    - **topo**: Binary file containing topology
- 

# Raster maps structure

- Raster maps are recorded in a matrix where each element represents a pixel with an **integer** or floating **point value**.
- Information about the raster maps is distributed in different files inside thematic subdirectories:
  - **cat**: Contains category information
  - **cell – fcell**: Contains binary files with the numerical matrix
  - **cellhd**: Contains maps headers
  - **colr**: Contains maps color informationon
  - **hist**: Contains information about raster map history: author, date, executed commands, etc...

# Databases with GRASS

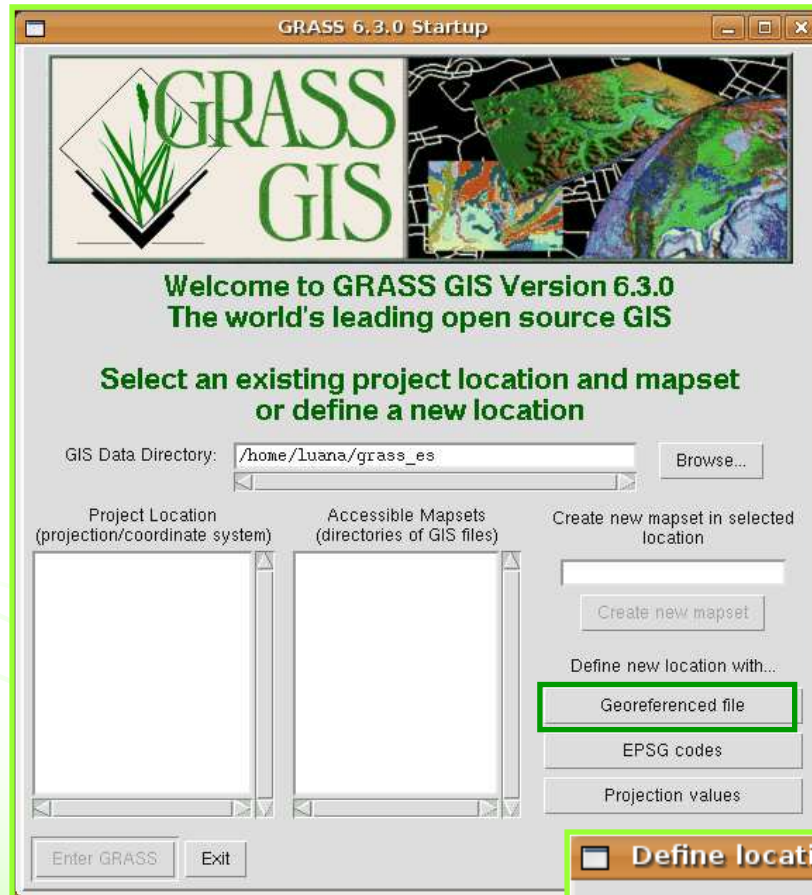




# Commands structure

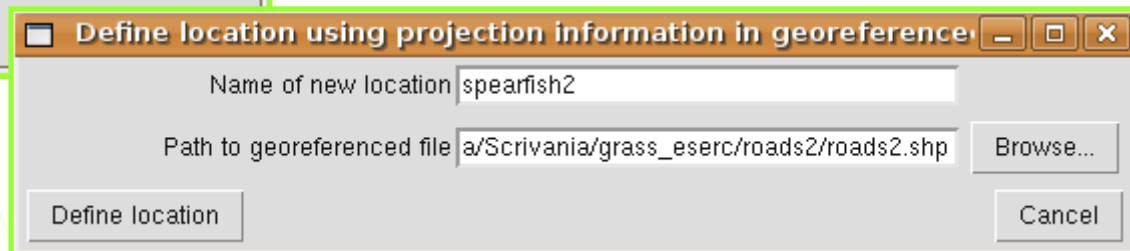
Prefix	Function class	Type of command	Example
d.*	display	graphical output	d.rast/d.vector: views raster / vector maps
db.*	database	database management	db.select: select value(s) from table
g.*	general	general file operations	g.rename: renames map
i.*	imagery	image processing	i.smap: image classifier
ps.*	postscripts	map creation in Postscript	ps.map: map creation
r.*	raster	raster data processing	r.buffer: buffer around raster features
r3.*	voxel	raster voxel data processing	r3.mapcalc: volume map algebra
v.*	vector	vector data processing	v.overlay: vector map intersection

# Create a new location



Georeferenced file:

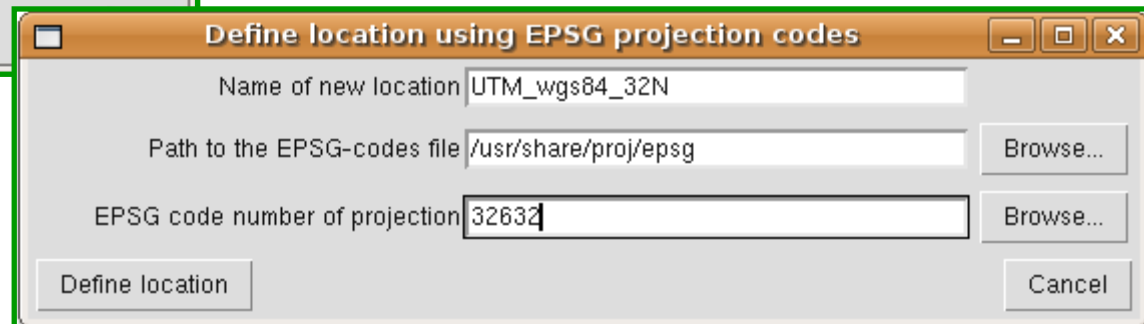
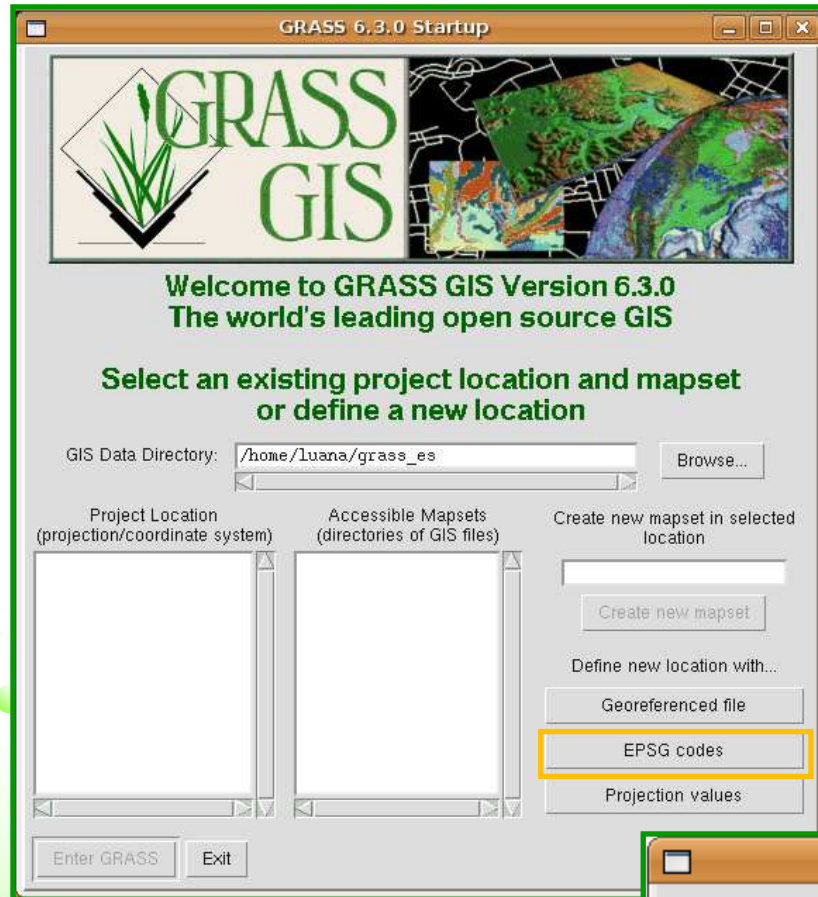
Insert the name of the new location to create (e.g. 'spearfish2') and the path to the georeferenced file (e.g. 'roads2.shp').



# Create a new location - 2

EPSG codes:

Insert the name of the new location to create (e.g. *'UTM\_wgs84\_32N'*) , the path to the EPSG-codes file (e.g. *'proj/epsg'*) and the EPSG code number of the chosen projection (e.g. *'32632'*).



# Create a new location - 3



```
EPSSG-codes
EPSSG CODES (from file: /usr/share/proj/epsg)
You can select EPSG code (in <brackets>) and copy it for later use

# WGS 84 / UTM zone 29N
<32629> +proj=utm +zone=29 +ellps=WGS84 +datum=WGS84 +units=m +no_defs <>

# WGS 84 / UTM zone 30N
<32630> +proj=utm +zone=30 +ellps=WGS84 +datum=WGS84 +units=m +no_defs <>

# WGS 84 / UTM zone 31N
<32631> +proj=utm +zone=31 +ellps=WGS84 +datum=WGS84 +units=m +no_defs <>

# WGS 84 / UTM zone 32N
<32632> +proj=utm +zone=32 +ellps=WGS84 +datum=WGS84 +units=m +no_defs <>

# WGS 84 / UTM zone 33N
<32633> +proj=utm +zone=33 +ellps=WGS84 +datum=WGS84 +units=m +no_defs <>

# WGS 84 / UTM zone 34N
<32634> +proj=utm +zone=34 +ellps=WGS84 +datum=WGS84 +units=m +no_defs <>

# WGS 84 / UTM zone 35N
<32635> +proj=utm +zone=35 +ellps=WGS84 +datum=WGS84 +units=m +no_defs <>

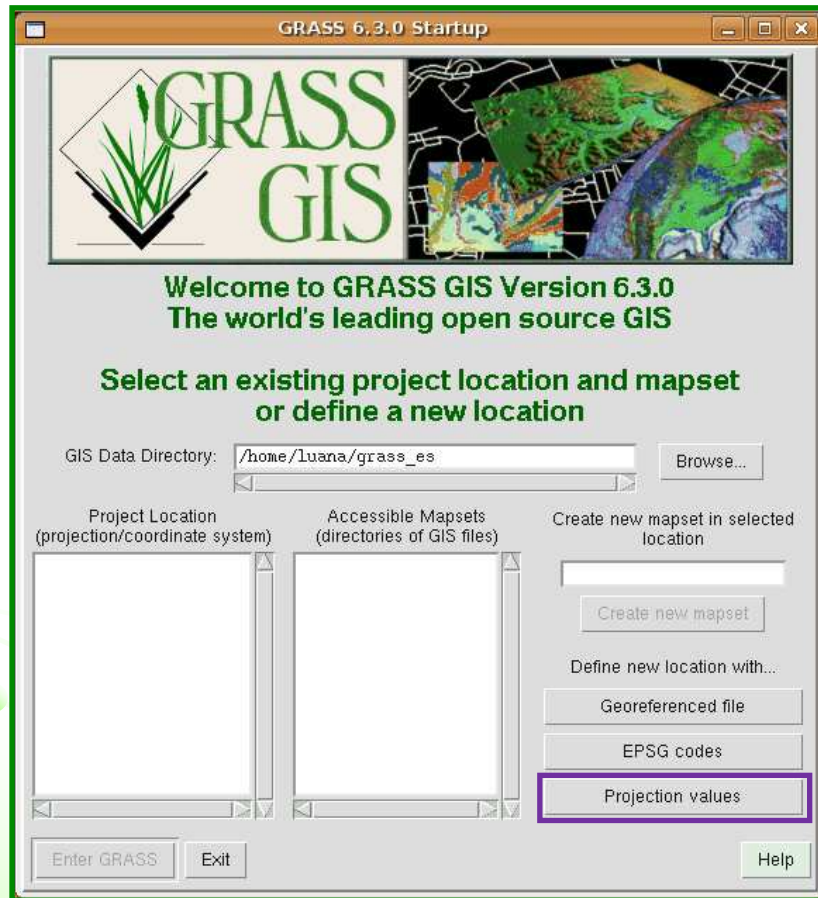
# WGS 84 / UTM zone 36N
<32636> +proj=utm +zone=36 +ellps=WGS84 +datum=WGS84 +units=m +no_defs <>

# WGS 84 / UTM zone 37N
<32637> +proj=utm +zone=37 +ellps=WGS84 +datum=WGS84 +units=m +no_defs <>

# WGS 84 / UTM zone 38N
<32638> +proj=utm +zone=38 +ellps=WGS84 +datum=WGS84 +units=m +no_defs <>
```

Search Grab code Close

# Create a new location - 4

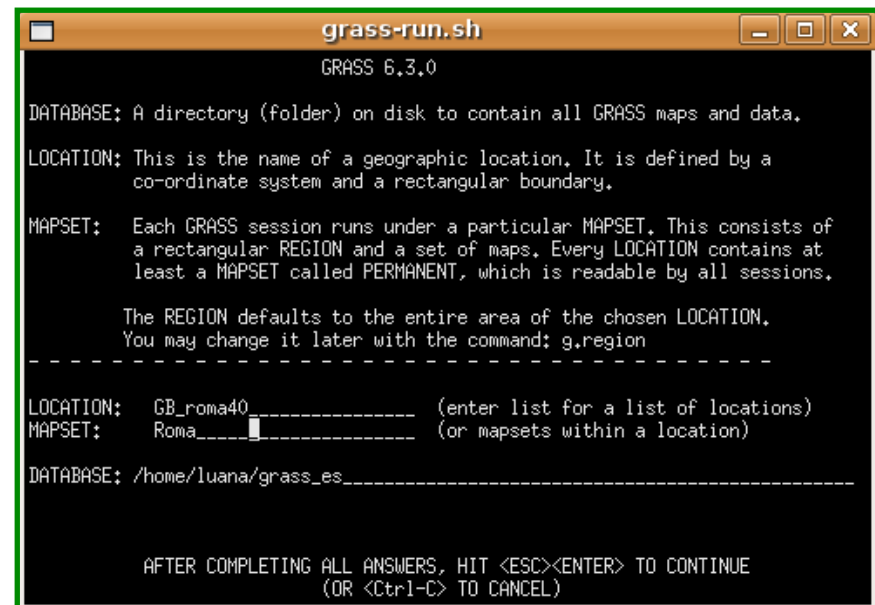


1

Name of location,  
mapset and  
database

Projection values:

Insert the all the parameters of the chosen projection (e.g. 'Gauss-Boaga Roma40').



# Create a new location - 5

2

Creation of the new location

```
grass-run.sh
LOCATION <GB_roma40> - doesn't exist

Available locations:
-----
UTM_wgs84_32N   spearfish2
-----

Would you like to create location <GB_roma40> ? (y/n) [y] y
```

```
grass-run.sh

To create a new LOCATION, you will need the following information:

1. The coordinate system for the database
   x,y (for imagery and other unreferenced data)
   Latitude-Longitude
   UTM
   Other Projection
2. The zone for the UTM database
   and all the necessary parameters for projections other than
   Latitude-Longitude, x,y, and UTM
3. The coordinates of the area to become the default region
   and the grid resolution of this region
4. A short, one-line description or title for the location

Do you have all this information? (y/n) [y] y
```

3

Checking availability of information

4

Insert type of the coordinate system

```
grass-run.sh

Please specify the coordinate system for location <GB_roma40>

A  x,y
B  Latitude-Longitude
C  UTM
D  Other Projection
RETURN to cancel

> D
```

# Create a new location - 6

```
grass-run.sh
Please specify the coordinate system for location <GB_roma40>
A  x,y
B  Latitude-Longitude
C  UTM
D  Other Projection
RETURN to cancel
> D
Other Projection coordinate system? (y/n) [y] y
```

5

Confirm the choice of  
'other projection'

6

Insert a  
description of the  
new location

```
grass-run.sh
Please enter a one line description for location <GB_roma40>
> Gauss-Boaga Roma40
=====
Gauss-Boaga Roma40
=====
ok? (y/n) [y] y
```

```
grass-run.sh
Please enter a one line description for location <GB_roma40>
> Gauss-Boaga Roma40
=====
Gauss-Boaga Roma40
=====
ok? (y/n) [y] y

Please specify projection name
Enter 'list' for the list of available projections
Hit RETURN to cancel request
>tmerc
```

7

Specify the  
projection name  
(e.g. '*tmerc*')



# Create a new location - 7

```
grass-run.sh
Please enter a one line description for location <GB_roma40>
> Gauss-Boaga Roma40
=====
Gauss-Boaga Roma40
=====
ok? (y/n) [y] y

Please specify projection name
Enter 'list' for the list of available projections
Hit RETURN to cancel request
>tmerc
Do you wish to specify a geodetic datum for this location?(y/n) [y] y

Please specify datum name
Enter 'list' for the list of available datums
or 'custom' if you wish to enter custom parameters
Hit RETURN to cancel request
>rome40
```

8

Specify the datum name (e.g. 'rome40')

9

Select datum transformation parameters (e.g. '1' for Italy)

```
grass-run.sh
Gauss-Boaga Roma40
=====
ok? (y/n) [y] y

Please specify projection name
Enter 'list' for the list of available projections
Hit RETURN to cancel request
>tmerc
Do you wish to specify a geodetic datum for this location?(y/n) [y] y

Please specify datum name
Enter 'list' for the list of available datums
or 'custom' if you wish to enter custom parameters
Hit RETURN to cancel request
>rome40

Now select Datum Transformation Parameters
Please think carefully about the area covered by your data
and the accuracy you require before making your selection.

Enter 'list' to see the list of available Parameter sets
Enter the corresponding number, or <RETURN> to cancel request
>1
```



# Create a new location - 8

```
grass-run.sh
ok? (y/n) [y] y

Please specify projection name
Enter 'list' for the list of available projections
Hit RETURN to cancel request
>tmerc
Do you wish to specify a geodetic datum for this location?(y/n) [y] y

Please specify datum name
Enter 'list' for the list of available datums
or 'custom' if you wish to enter custom parameters
Hit RETURN to cancel request
>rome40

Now select Datum Transformation Parameters
Please think carefully about the area covered by your data
and the accuracy you require before making your selection.

Enter 'list' to see the list of available Parameter sets
Enter the corresponding number, or <RETURN> to cancel request
>1

Enter Scale Factor at the Central Meridian [1.0000000000]: 0.9996
```

10

Insert the scale factor at the Central Meridian (e.g. '0.9996')

11

Enter the Central Parallel (e.g. '0')

```
grass-run.sh

Please specify projection name
Enter 'list' for the list of available projections
Hit RETURN to cancel request
>tmerc
Do you wish to specify a geodetic datum for this location?(y/n) [y] y

Please specify datum name
Enter 'list' for the list of available datums
or 'custom' if you wish to enter custom parameters
Hit RETURN to cancel request
>rome40

Now select Datum Transformation Parameters
Please think carefully about the area covered by your data
and the accuracy you require before making your selection.

Enter 'list' to see the list of available Parameter sets
Enter the corresponding number, or <RETURN> to cancel request
>1

Enter Scale Factor at the Central Meridian [1.0000000000]: 0.9996

Enter Central Parallel (23N): 0
```

# Create a new location - 9

```
Enter Scale Factor at the Central Meridian [1.0000000000]: 0.9996
Enter Central Parallel (23N) :0
Enter Central Meridian (96W) :9
```

12

Insert the Central Meridian (e.g. '9' for W zone)

13

Enter False Easting (e.g. '1500000' for W zone)

```
Enter Scale Factor at the Central Meridian [1.0000000000]: 0.9996
Enter Central Parallel (23N) :0
Enter Central Meridian (96W) :9
Enter False Easting [0.0000000000]: 1500000
```

```
Enter Scale Factor at the Central Meridian [1.0000000000]: 0.9996
Enter Central Parallel (23N) :0
Enter Central Meridian (96W) :9
Enter False Easting [0.0000000000]: 1500000
Enter False Northing [0.0000000000]: 0
```

14

Enter False Northing (e.g. '0')

# Create a new location - 10

```
Enter Scale Factor at the Central Meridian [1.0000000000]: 0.9996
Enter Central Parallel (23N) :0
Enter Central Meridian (96W) :9
Enter False Easting [0.0000000000]: 1500000
Enter False Northing [0.0000000000]: 0
Enter plural form of units [meters]: meters
```

15

Insert plural form of units (e.g. '*meters*')

16

Enter singular for unit (e.g. '*meter*')

```
Enter Scale Factor at the Central Meridian [1.0000000000]: 0.9996
Enter Central Parallel (23N) :0
Enter Central Meridian (96W) :9
Enter False Easting [0.0000000000]: 1500000
Enter False Northing [0.0000000000]: 0
Enter plural form of units [meters]: meters
Enter singular for unit: meter
```

17

Enter conversion factor (e.g. '*1*')

```
Enter Scale Factor at the Central Meridian [1.0000000000]: 0.9996
Enter Central Parallel (23N) :0
Enter Central Meridian (96W) :9
Enter False Easting [0.0000000000]: 1500000
Enter False Northing [0.0000000000]: 0
Enter plural form of units [meters]: meters
Enter singular for unit: meter
Enter conversion factor from meters to meters: 1
```

# Create a new location - 11

```
grass-run.sh
DEFINE THE DEFAULT REGION

===== DEFAULT REGION =====
| NORTH EDGE:4800000___ |
WEST EDGE |                | EAST EDGE
2200000___|                | 2800000___
| SOUTH EDGE:4200000___ |
=====

PROJECTION: 99 (Other Projection)      ZONE: 0

GRID RESOLUTION
  East-West: 10_____
  North-South: 10_____

AFTER COMPLETING ALL ANSWERS, HIT <ESC><ENTER> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)
```

18

Definition of the default region

```
projection: 99 (Other Projection)
zone: 0
north:      4800000
south:      4200000
east:       2800000
west:       2200000

e-w res:    10
n-s res:    10

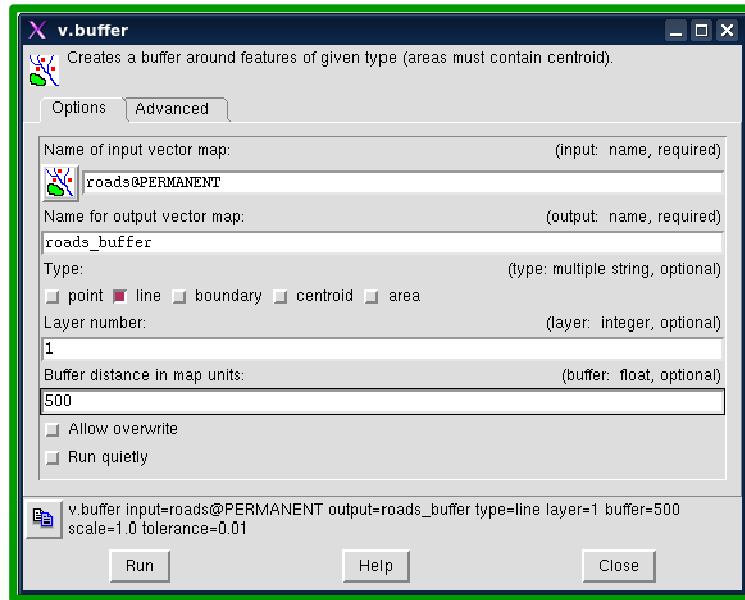
total rows: 60000
total cols: 60000
total cells: 3,600,000,000

Do you accept this region? (y/n) [y] > y
```

```
Do you accept this region? (y/n) [y] > y
LOCATION <GB_roma40> created!

Hit RETURN -->
```

# User interfaces



GIS manager  
(default)

and / or

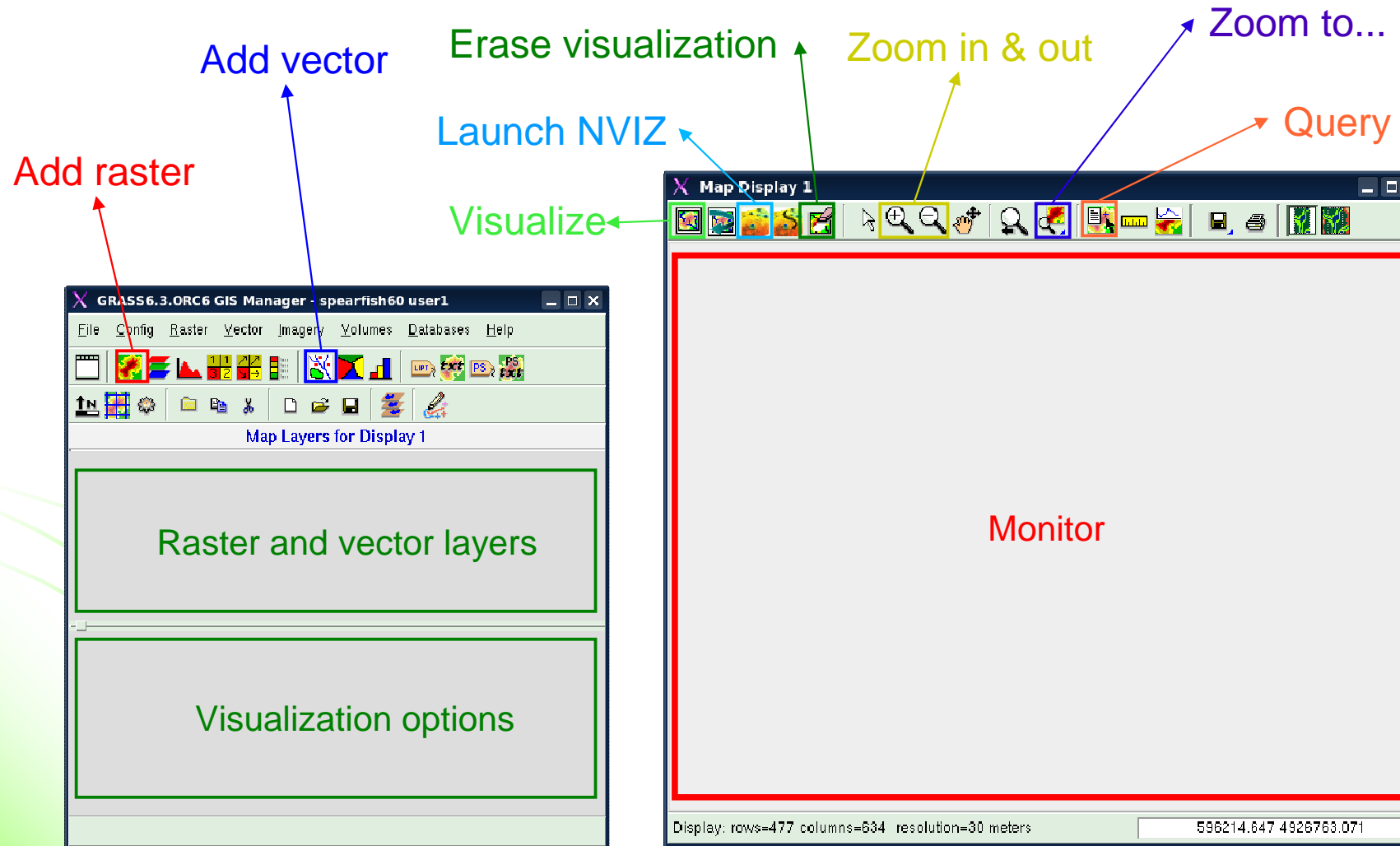
Command line

A screenshot of the 'Shell - GRASS' window. It shows the command 'v.buffer input=roads@PERMANENT output=roads\_buffer type=line layer=1 buffer=500 scale=1.0 tolerance=0.01' being executed. The output shows progress: 'Lines buffers... 100%', 'Building parts of topology...', 'Building topology ...', '825 primitives registered', 'Topology was built.', and a summary of statistics: 'Number of nodes : 825', 'Number of primitives : 825', 'Number of points : 0', 'Number of lines : 0', 'Number of boundaries : 825', 'Number of centroids : 0', 'Number of areas : -', 'Number of isles : -', 'Snapping boundaries...', 'All vertices : 37397', 'Registered points : 36572', 'Nodes marked as anchor : 36382', 'Nodes marked to be snapped: 190', 'Snapped vertices : 261', 'New vertices : 260', 'Breaking boundaries...', and 'Intersections: 141127 (line 184869)'.

additional ways of using GRASS:

- QGIS (with GRASS plug-in)
- JavaGRASS (JGRASS)

# GIS manager

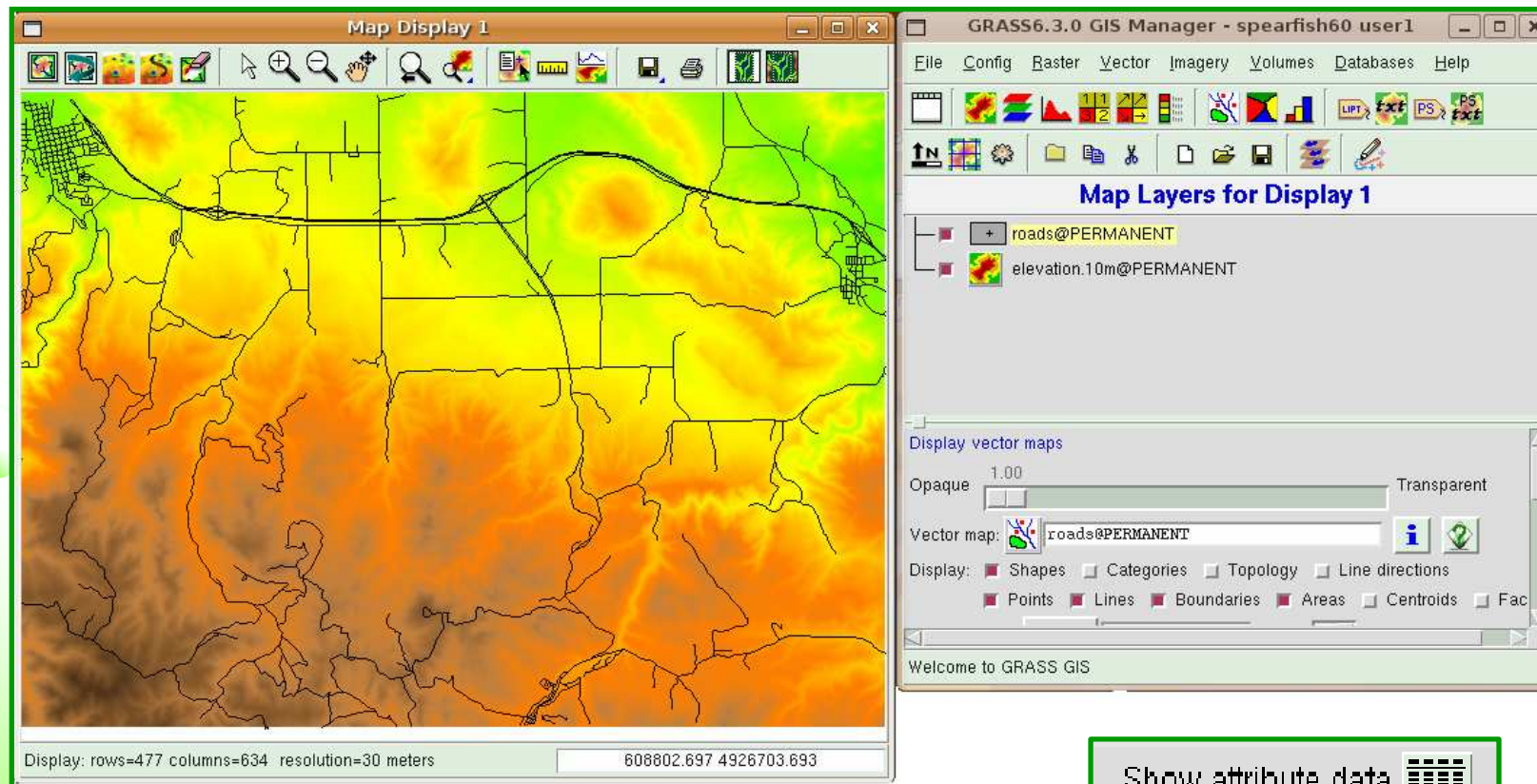


# Region in GRASS

- A *region* defines the resolution used for maps and the coordinates of boundaries (map extents)
    - **Default:** Settings of the working GRASS location
    - **Current:** Is the actual working area and it can be modified by the user
  - Vectorial modules work all over the vector map
  - Raster modules work with the current resolution/region
  - How to modify the region:
    - **g.region:** Changes the resolution and the coordinate boundaries of the current region.
- [ Config -> Region -> Change region settings ]

# Vector - Attribute query

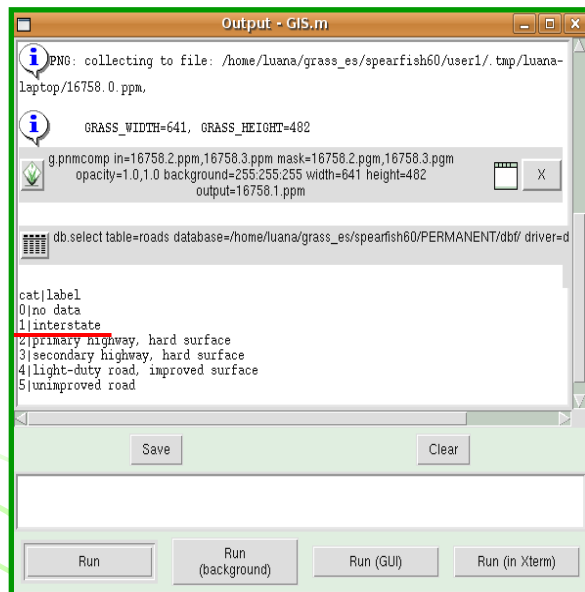
Extract interstate (highway) from roads vector map and buffer interstate for 3km in each direction.





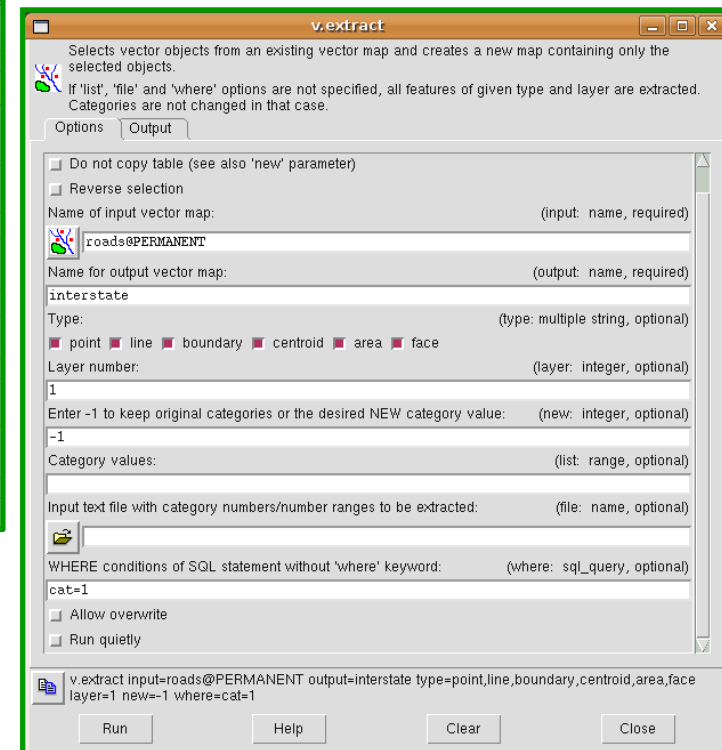
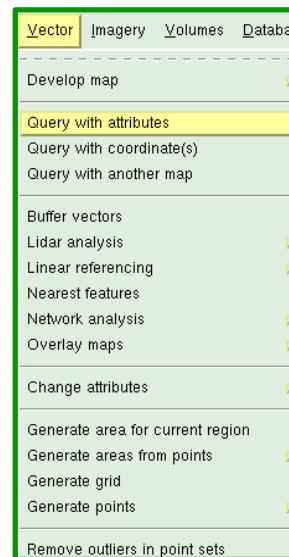
# Vector - Attribute query - 2

First take a look at the attribute table to get the column name and value for the interstate road:

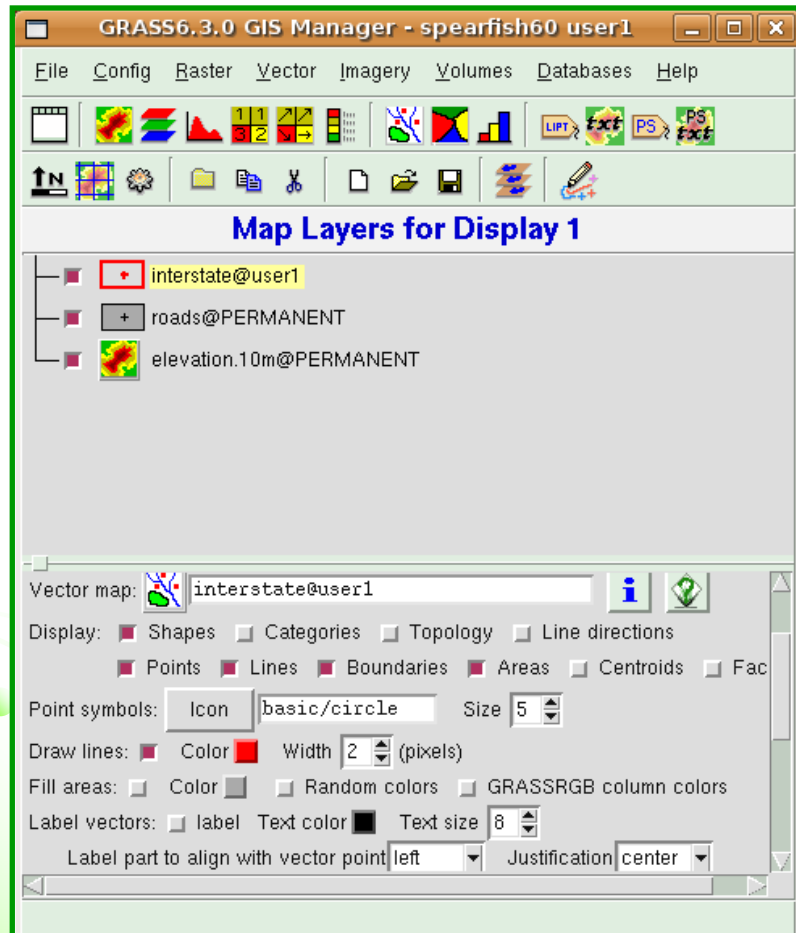


Extract only the **interstate** road:

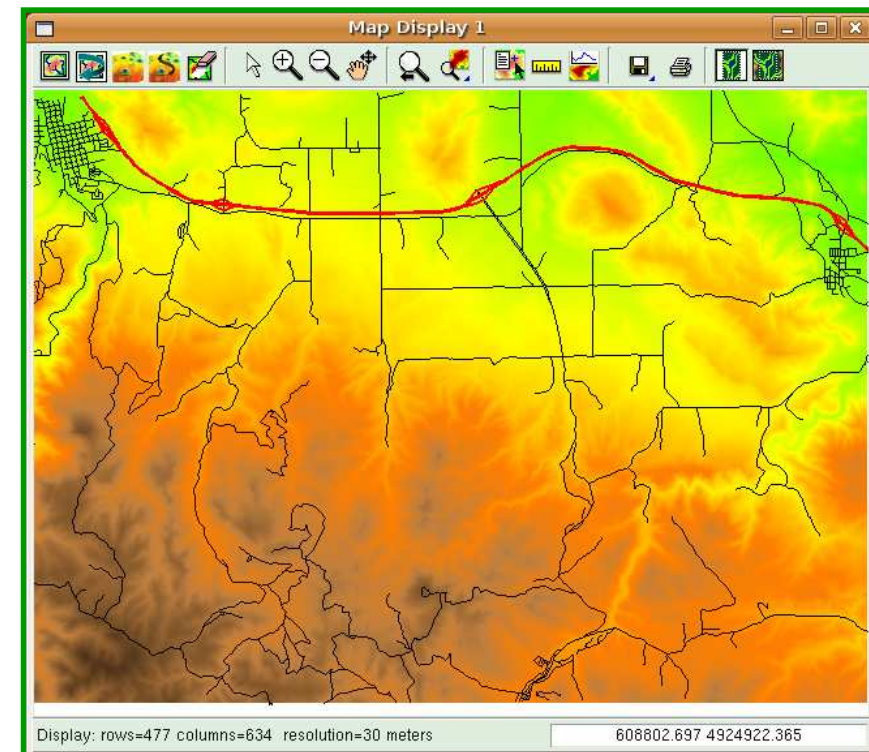
*v.extract*



# Vector - Attribute query - 3

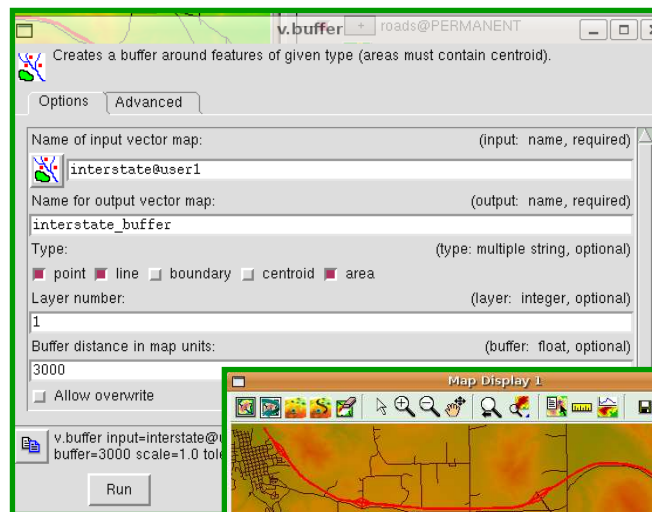
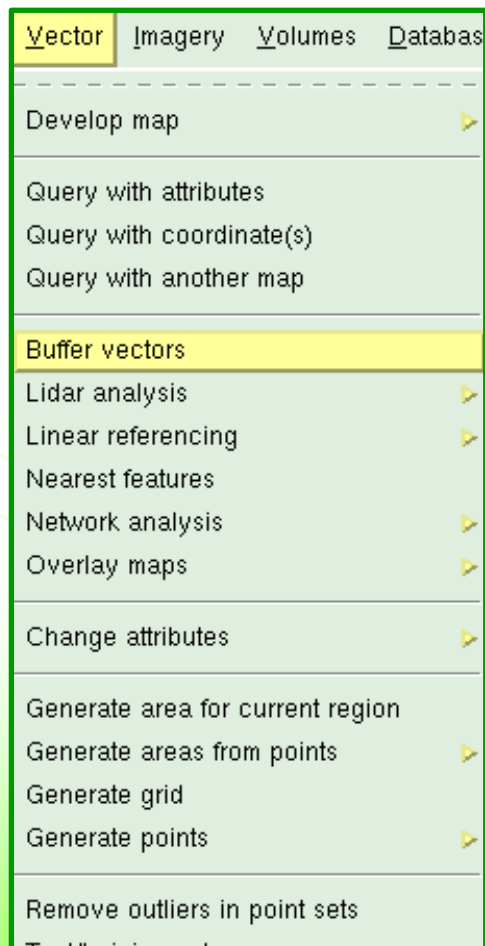


Assign a line width = 2 and a red color to the extracted road

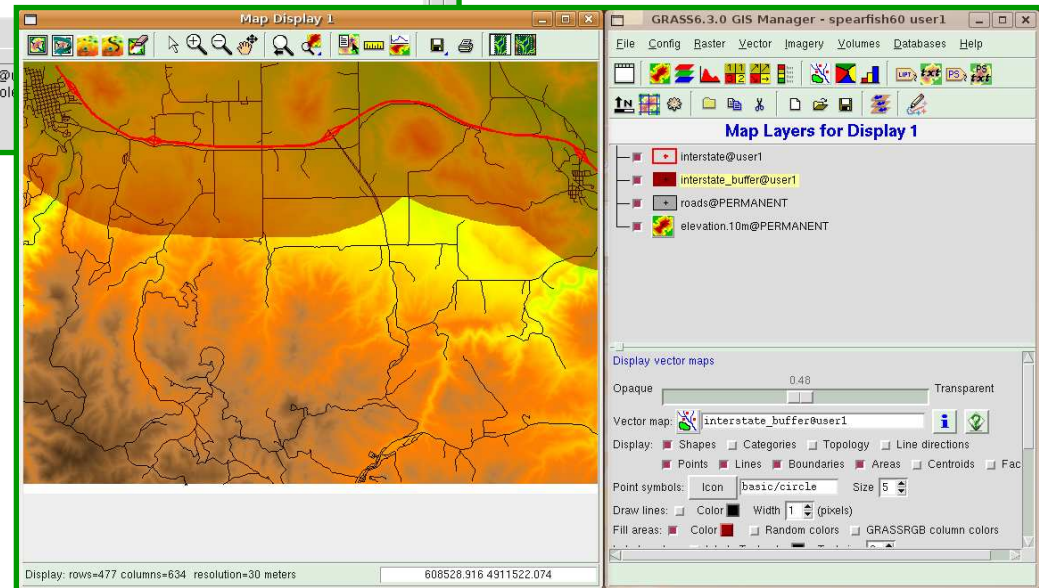


# Buffer – *noise impact example*

Buffer the interstate road (give buffer in map units which is meters here --> 3000)



*v.buffer*



# From vector to raster

Look at the attribute table of vector map 'landuse':

The screenshot displays the GRASS GIS 6.3.0 interface. The main window, 'Map Display 1', shows a map with a color-coded land use overlay. The 'GRASS6.3.0 GIS Manager - spearfish60 user1' window on the right lists the map layers for display 1:

- interstate@user1
- landuse@PERMANENT
- interstate\_buffer@user1
- roads@PERMANENT
- elevation.10m@PERMANENT

The 'Output - GIS.m' window at the bottom left shows the attribute table for the 'landuse' vector map. The table has two columns: 'cat' and 'label'. The data is as follows:

cat	label
1	residential
2	commercial and services
3	industrial
4	other urban
5	reservoirs
6	bare exposed rock
7	quarries, strip mines and gravel pits
8	transportation and utilities

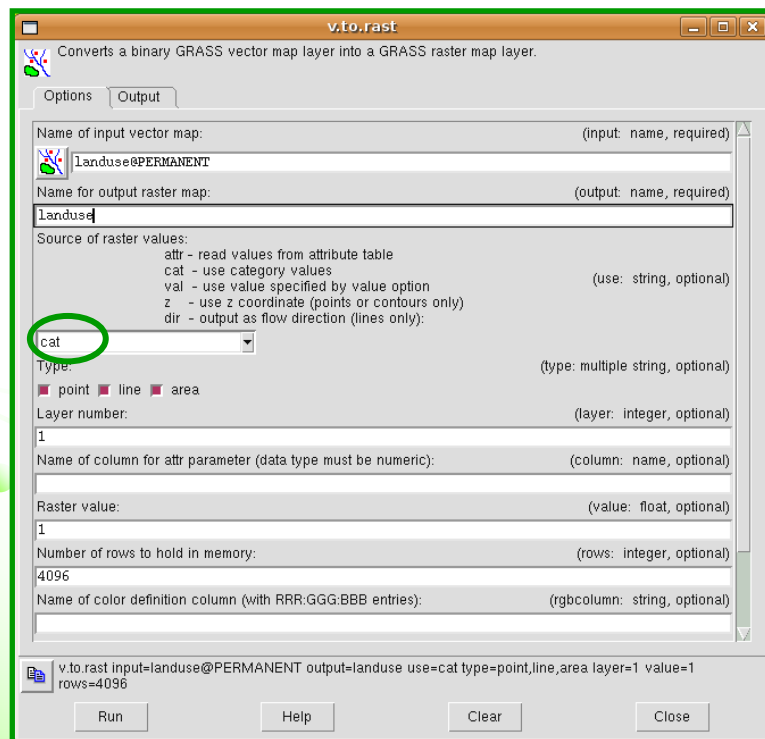
The 'GRASS6.3.0 GIS Manager' window also shows the 'Show attribute data' button, which is highlighted with a green box and a green arrow.



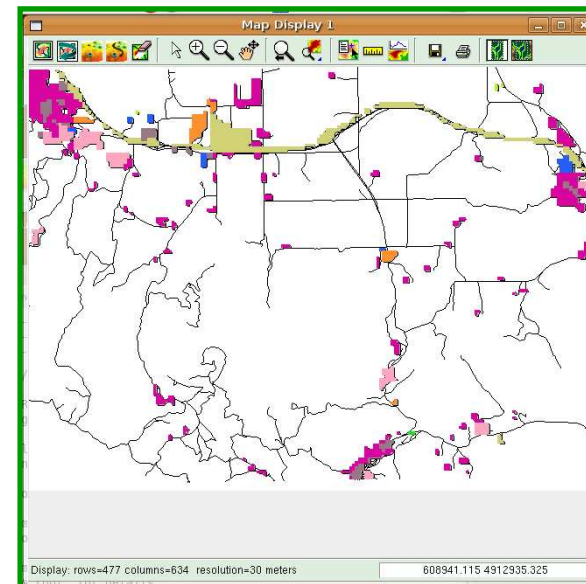
# From vector to raster - 2

We can create a raster map starting from 'landuse' vector map using the command **v.to.rast**:

**v.to.rast**



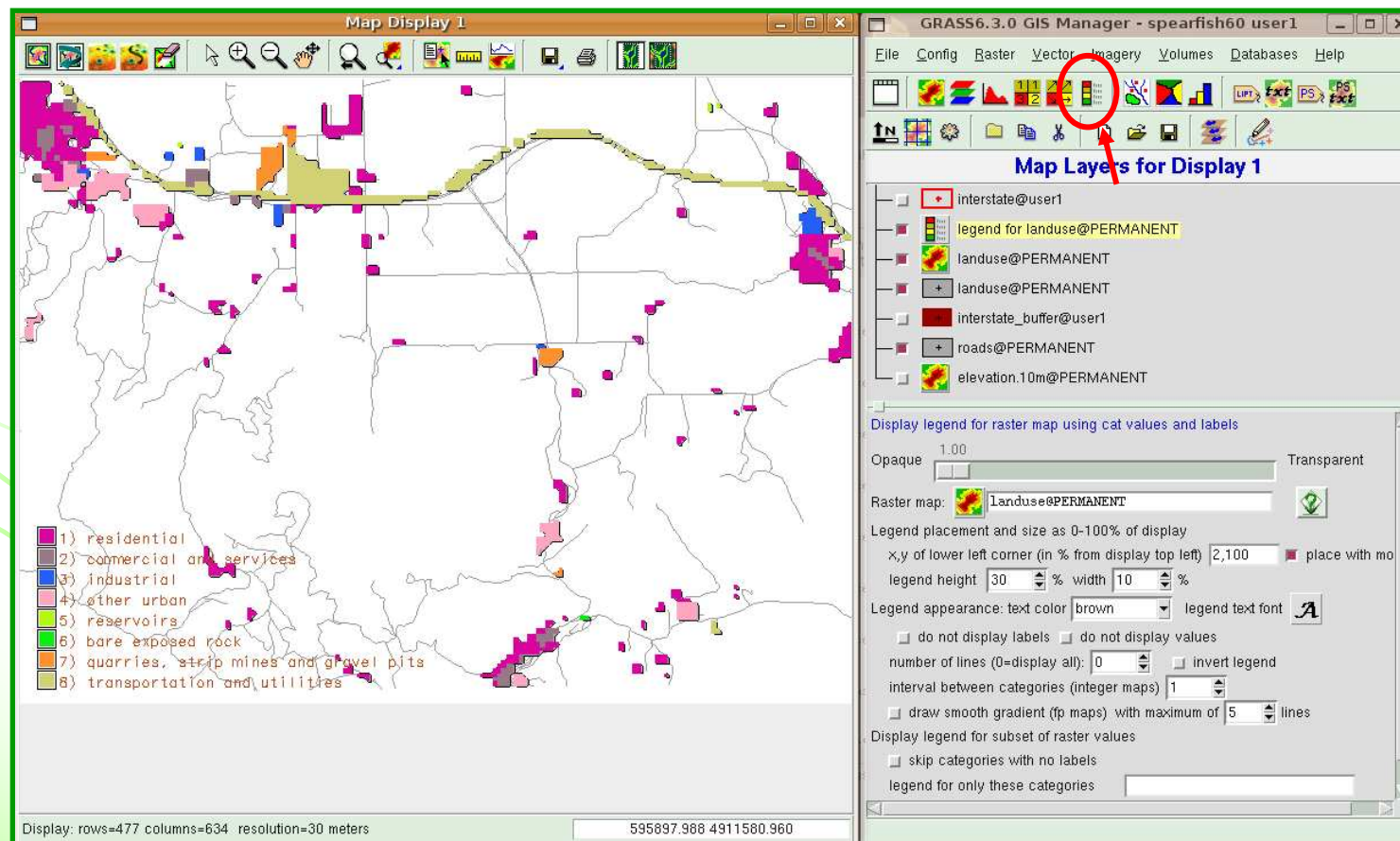
In this case we choose to use the categories values to create the new raster



**Remark:** the result depends on region settings, pay attention to the resolution!

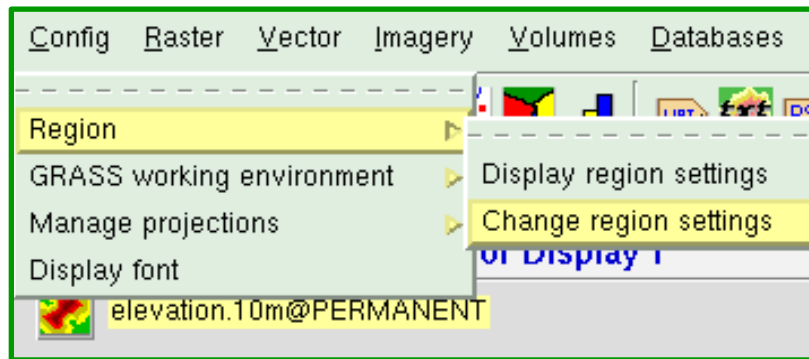
# From vector to raster - 3

To understand the meaning of the different colours we can add a legend to the map:

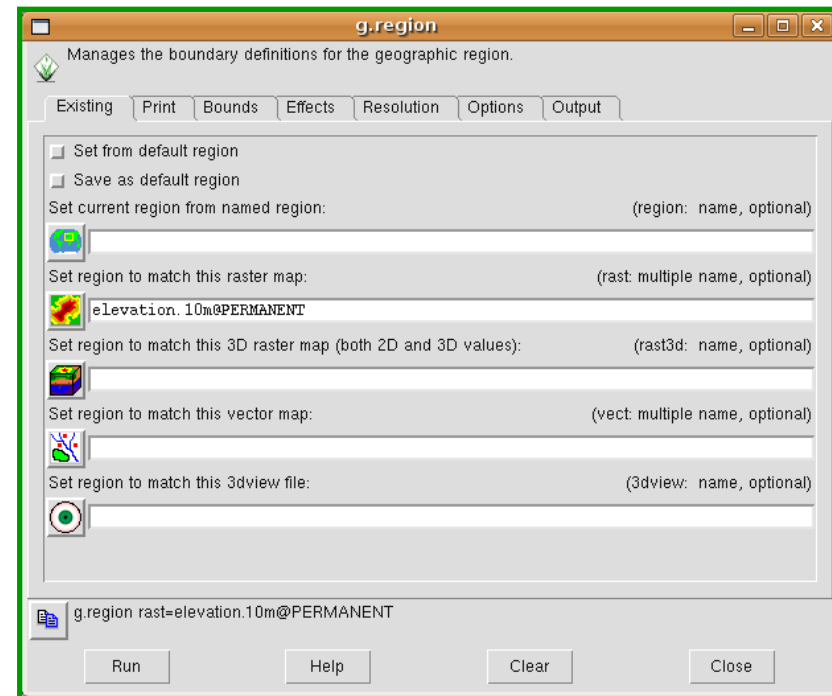


# Change region settings

Set current region to match elevation.10m map:

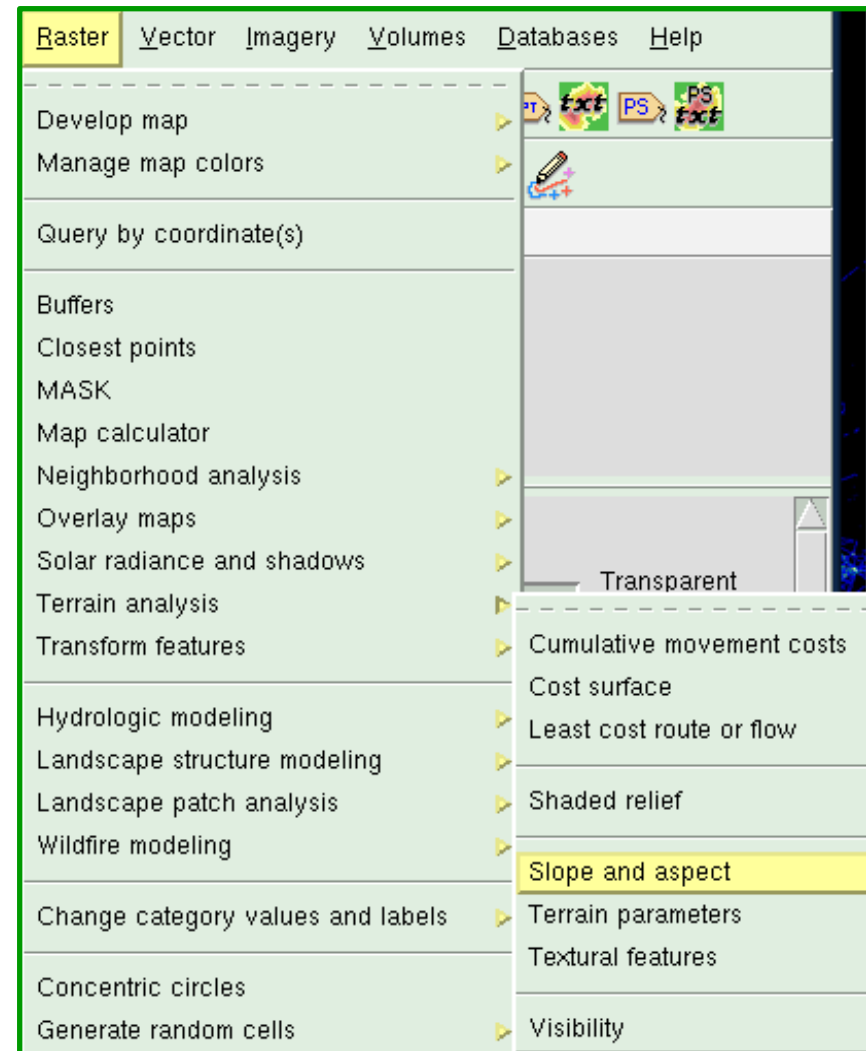


*g.region*



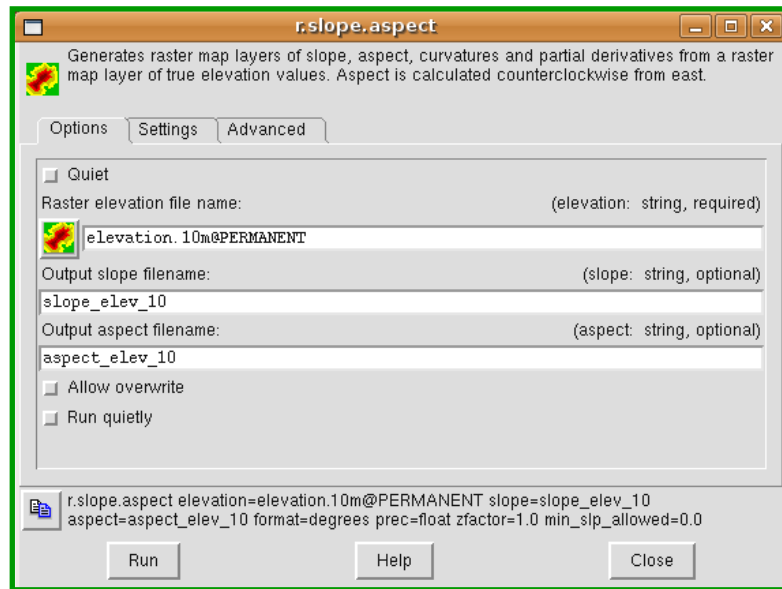
# Raster – Slope & Aspect - 2

- Calculate slope and aspect from a DEM
- Horizontal angles are counted counterclockwise from the East
- Slopes are calculated by default in degrees

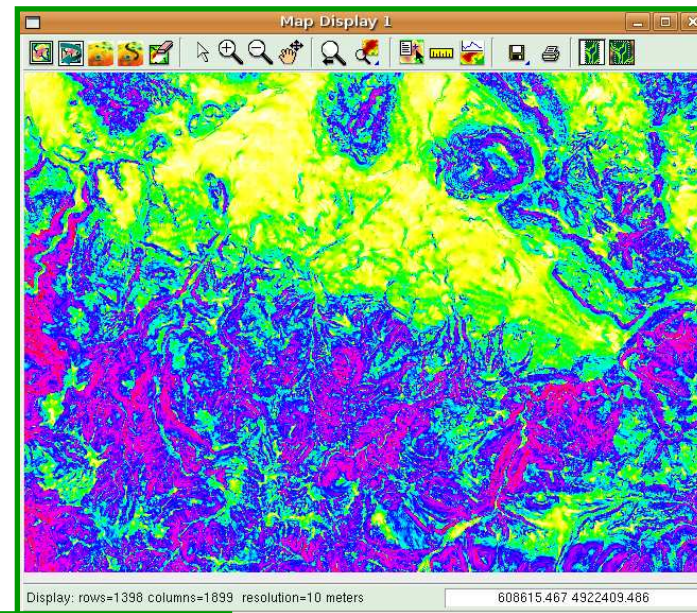




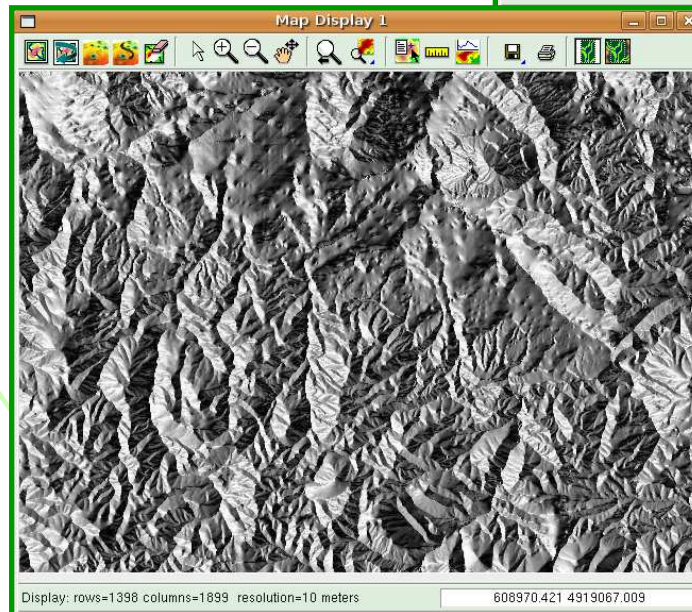
# Raster – Slope & Aspect - 3



*r.slope.aspect*

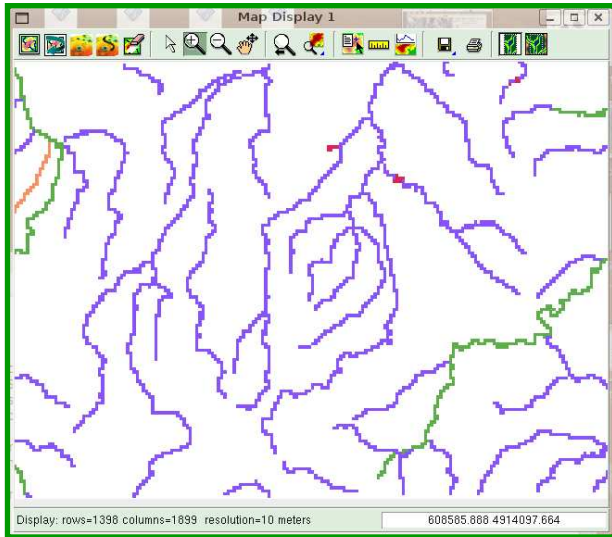


*slope*



*aspect*

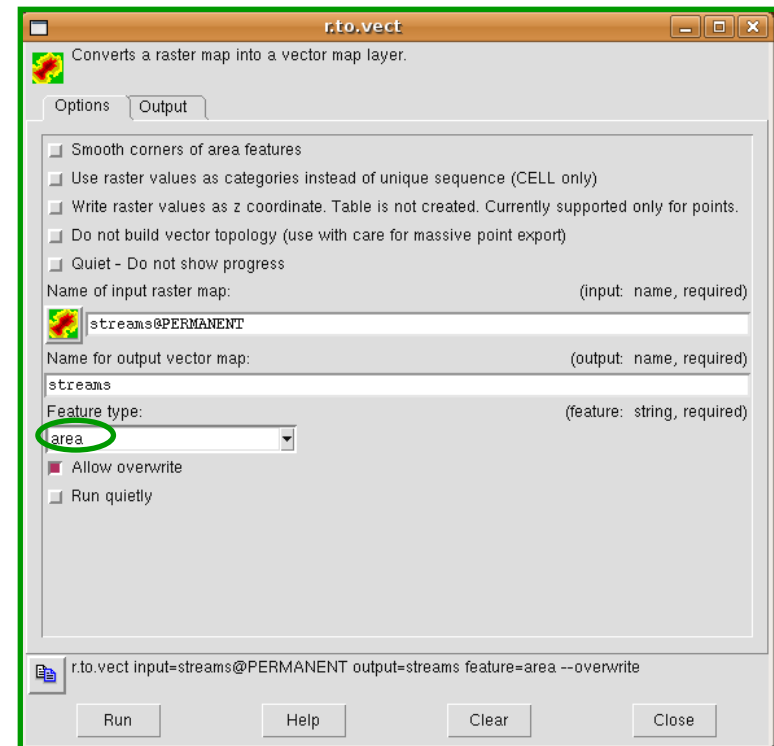
# From raster to vector



*streams raster map*

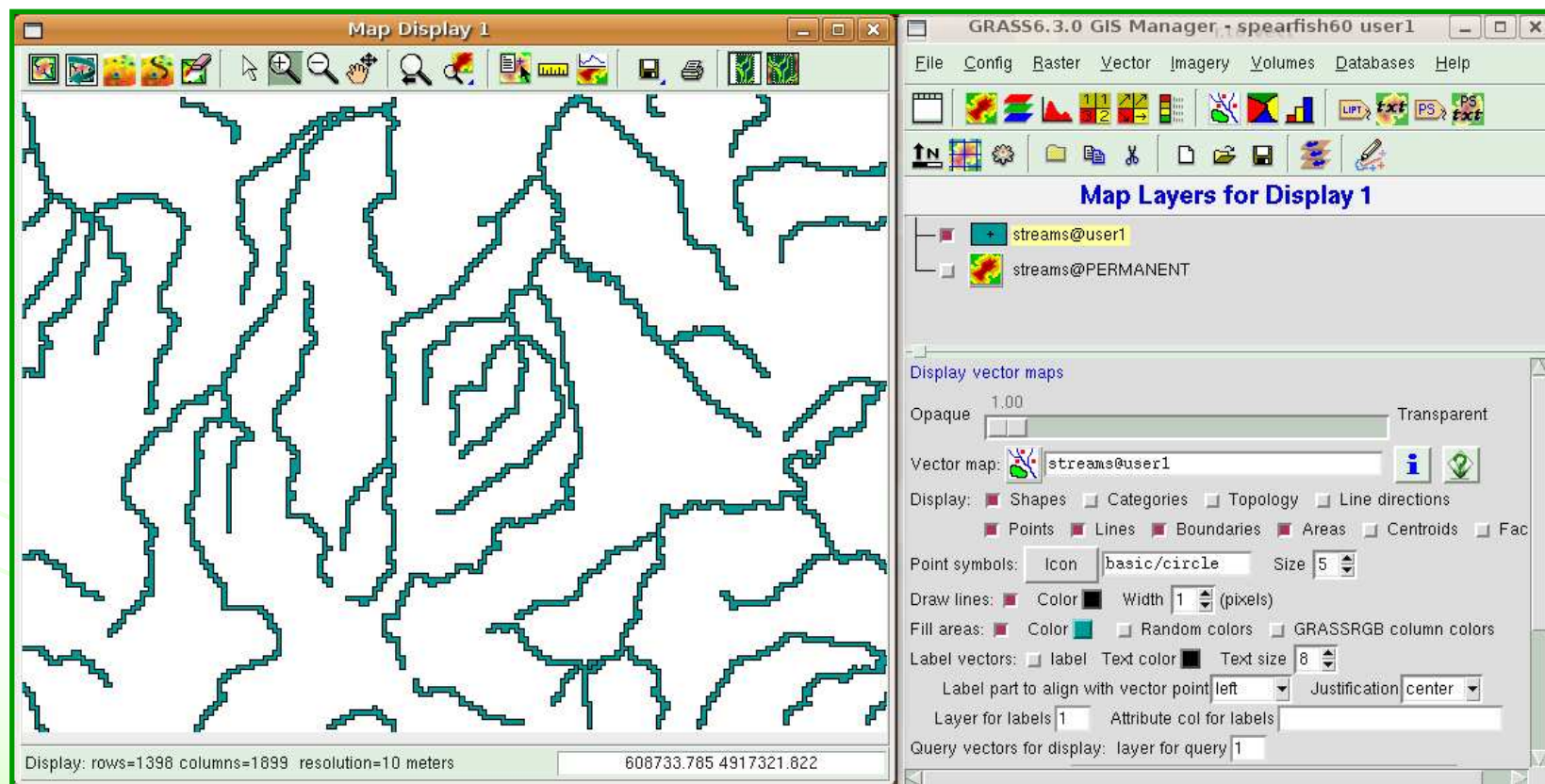
Also in this case you have  
to check region settings.  
??

We can create a vector map starting from  
'streams' raster map using the command  
**r.to.vect**:



*r.to.vect*

# From raster to vector - 2

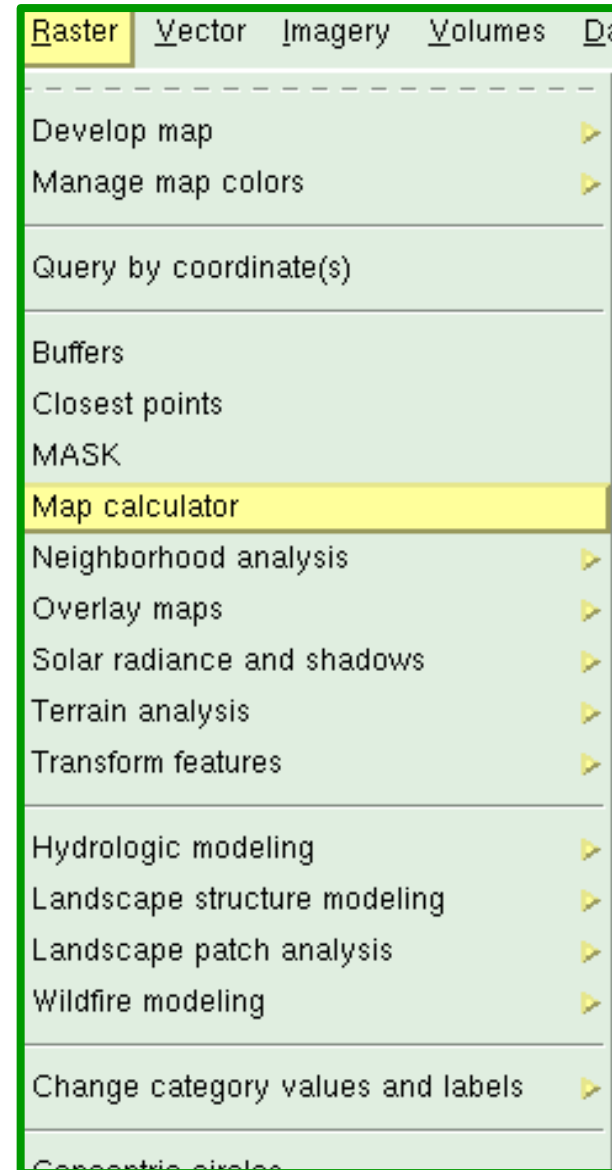


# Raster – Map algebra

GRASS does map algebra with  
**r.mapcalc**



Simplified version:  
**r.mapcalculator**

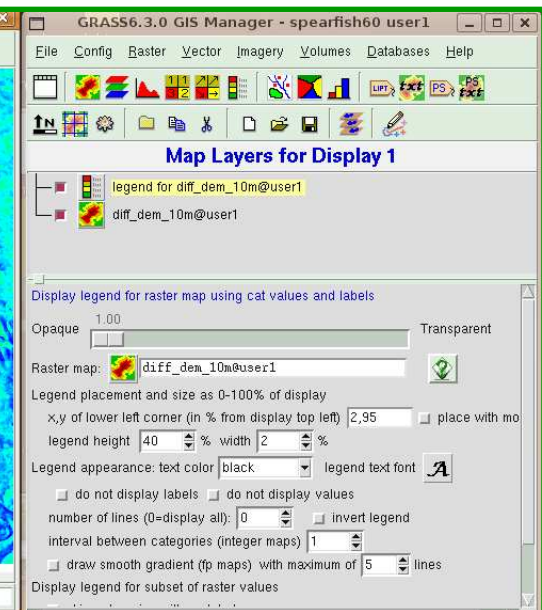
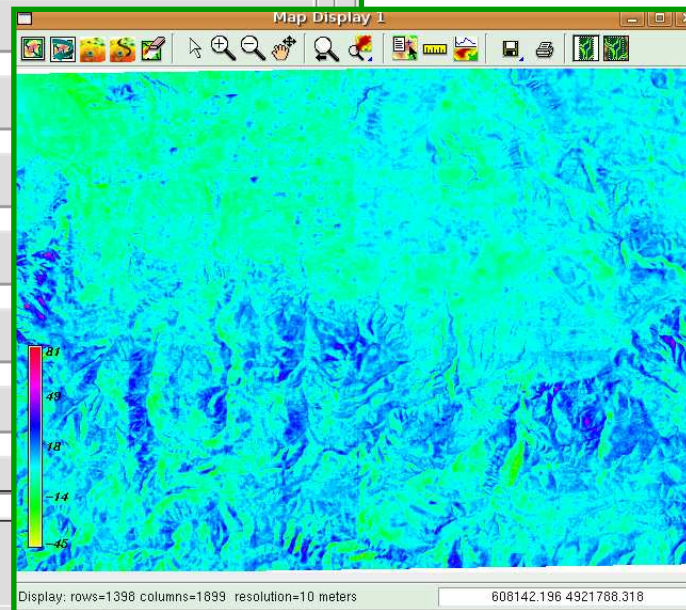
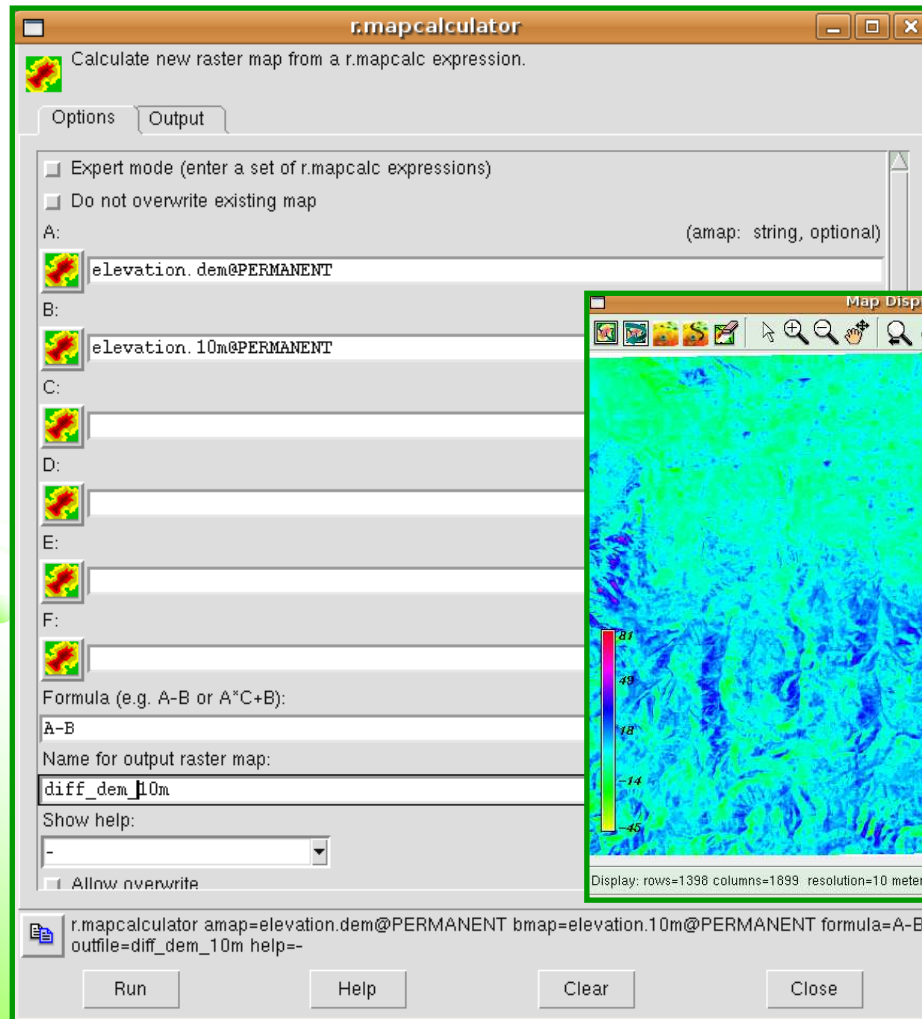




# Raster – Map algebra - 2

*r.mapcalculator*

Compute the difference between the two raster maps *elevation.dem* and *elevation.10m*



# Raster – Map algebra - 3

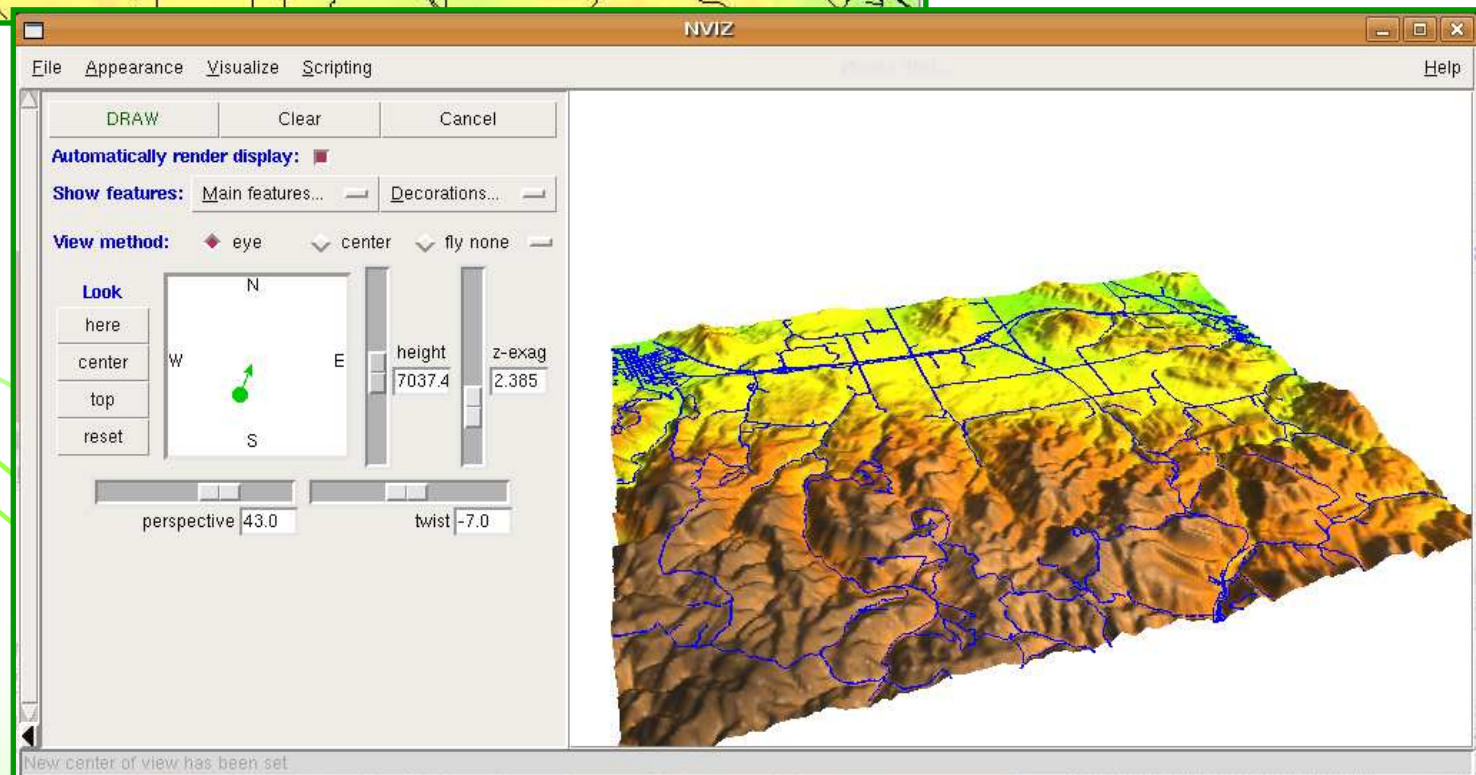
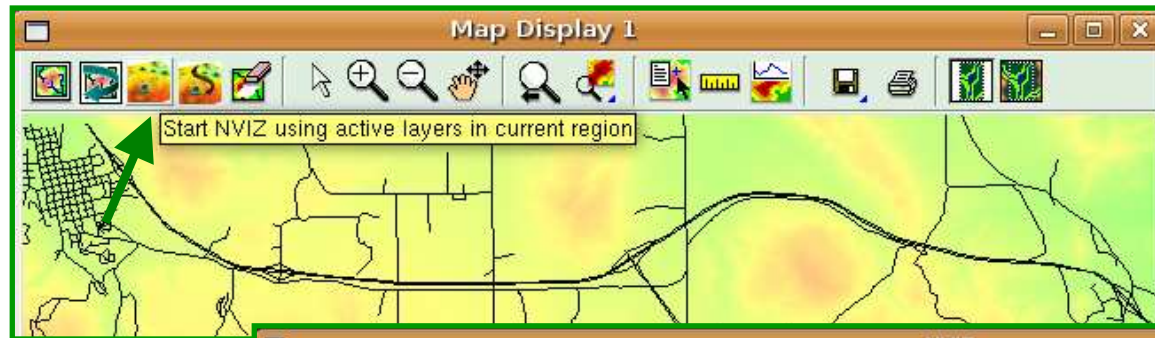
## Logical operators

Operator	Meaning	Type
!	not	Logical
^	exponentiation	Arithmetic
%	modulus	Arithmetic
/	division	Arithmetic
*	multiplication	Arithmetic
+	addition	Arithmetic
-	subtraction	Arithmetic
==	equal	Logical
!=	not equal	Logical
>	greater than	Logical
>=	greater than or equal	Logical
<	less than	Logical
<=	less than or equal	Logical
&&	and	Logical
	or	Logical
&&&	and[1]	Logical
	or[1]	Logical
#	color separator operator	Arithmetic

## Mathematical functions

function	description
abs(x)	return absolute value of x
acos(x)	inverse cosine of x (result is in degrees)
asin(x)	inverse sine of x (result is in degrees)
atan(x)	inverse tangent of x (result is in degrees)
atan(x,y)	inverse tangent of y/x (result is in degrees)
cos(x)	cosine of x (x is in degrees)
double(x)	convert x to double-precision floating point
eval([x,y,...],z)	evaluate values of listed expr, pass results to
exp(x)	exponential function of x
exp(x,y)	x to the power y
float(x)	convert x to single-precision floating point
graph(x,x1,y1[x2,y2..])	convert the x to a y based on points in a graph
if	decision options:
if(x)	1 if x not zero, 0 otherwise
if(x,a)	a if x not zero, 0 otherwise
if(x,a,b)	a if x not zero, b otherwise
if(x,a,b,c)	a if x > 0, b if x is zero, c if x < 0
int(x)	convert x to integer [ truncates ]
isnull(x)	check if x = NULL
log(x)	natural log of x
log(x,b)	log of x base b
max(x,y[,z...])	largest value of those listed
median(x,y[,z...])	median value of those listed
min(x,y[,z...])	smallest value of those listed
mode(x,y[,z...])	mode value of those listed
not(x)	1 if x is zero, 0 otherwise
pow(x,y)	x to the power y
rand(a,b)	random value x : a < x < b
round(x)	round x to nearest integer
sin(x)	sine of x (x is in degrees)
sqrt(x)	square root of x
tan(x)	tangent of x (x is in degrees)

# Visualization with NVIZ

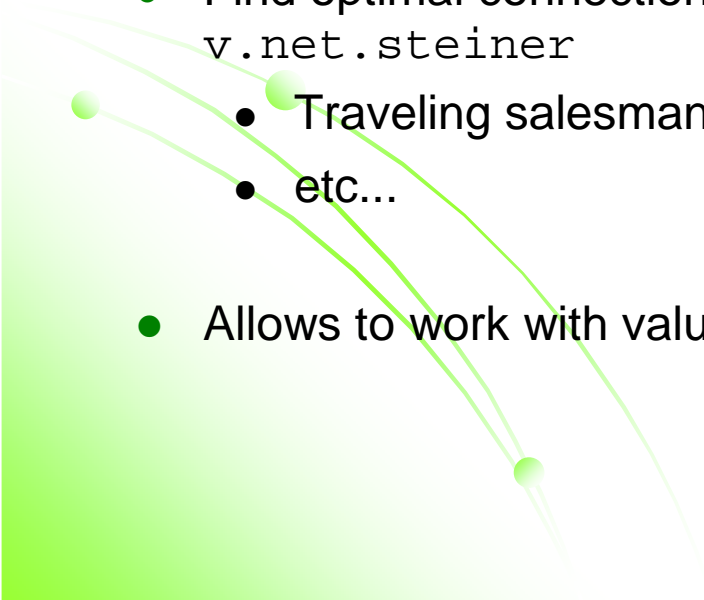


# Other GRASS functionalities

- Import/Export
  - Vector/raster analysis
  - Rasterization / vectorization
  - Overlay
  - Queries
  - Geometry management
  - Map algebra
  - Neighborhood analysis
  - Interpolation
  - Network analysis
  - Datum shifting
  - Solar radiance & shadows
  - Terrain analysis
  - Hydrologic modeling
  - Wildfire modeling
  - LiDAR analysis
  - Geostatistics
  - Optical image processing
  - 3D analysis
  - ....
- 
- A decorative graphic in the bottom-left corner consisting of a bright green square and several thin, curved green lines that sweep upwards and to the right, ending near the center of the slide.

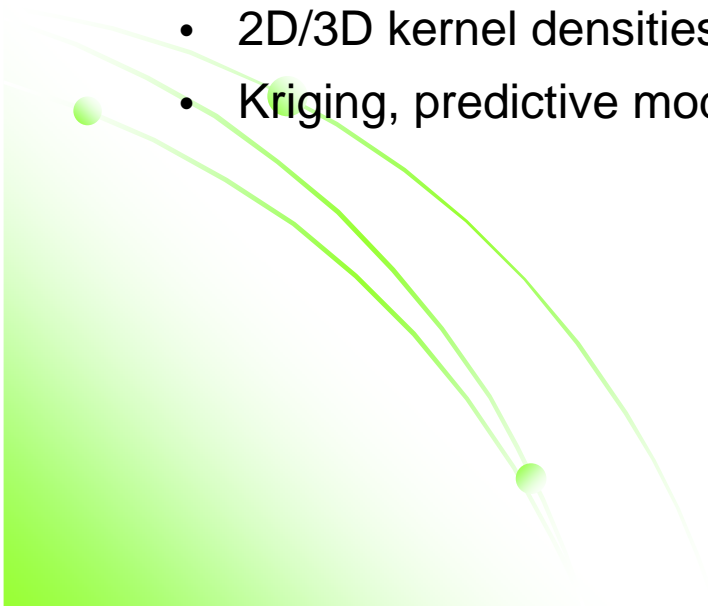


# Network analysis

- Methods:
    - Find shortest path along vector network – *road navigation* →  
`v.net.path`
    - Subdivide a network in subnetworks (iso-distances) – *how far can I go from a node in all directions* → `v.net.iso`
    - Find optimal connection between nodes (Minimum Steiner tree) →  
`v.net.steiner`
      - Traveling salesman problem → `v.net.salesman`
      - etc...
    - Allows to work with values in a attribute table
- 
- A decorative graphic in the bottom-left corner consisting of several overlapping, curved green lines of varying opacity, creating a sense of motion or a stylized path.

# Interpolation methods

- Interpolation of vector maps in order to create rasters  
[ Raster -> Interpolate surfaces ]:
  - 2D inverse distance weighted → `v surf.idw`
  - 2D regularized splines with tension (with cross-validation) → `v surf.rst`
  - 3D regularized splines with tension (with cross-validation) → `v vol.rst`
  - Bilinear and Bicubic splines with Tychonov regularizator → `v surf.bspline`
  - 2D/3D kernel densities → `v.kernel`
  - Kriging, predictive models (R-Stats)



# Volume processing

GRASS is able to process and display volumes (3D voxels)

Functionality:

- 3D Import/Export
- Interpolation with 3D regularized splines with tension
- 3D Algebra
- Volumes visualization with NVIZ: Iso-surfaces and profiles



# Import/Export data

- GRASS always import the complete map
- Exports raster maps within the **current region**
- Support a lot of vector and raster formats:

## Vectorial (`v.in.*`, `v.out.*`)

- **OGR:** ESRI Shapefile, UK .NTF, SDTS, TIGER, S57, MapInfo File, DGN, VRT, AVCBin, REC, Memory, CSV, GML, KML, ODBC, Pgeo, PostgreSQL/PostGIS, OGDl, ecc...
- ArcInfo E00
- ASCII
- DXF
- ...

## Raster (`r.in.*`, `r.out.*`)

- **GDAL:** ArcInfo, CEOS, DOQ, DTED, ENVI, Envisat, Erdas Img/LAN, FAST, (Geo)TIFF, HDF4, SAR, SDTS, ecc...
- ASCII
- Bin: binary, BIL, GMT, ecc...
- Matlab file
- SRTM
- ...

# Who's using GRASS?



# Document license

This work is released under a 'Creative Commons License'

<http://creativecommons.org/licenses/by-sa/3.0/>



**Attribution.** You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).



**Share Alike.** If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.

# Bibliography

“Introduction to GIS GRASS and QGIS” ---- 2007 Roberto Antolín, Spain

“Introduction to GRASS GIS” ---- 2008 Eugenio Realini

GRASS tutorial (in italian)

[http://www.ing.unitn.it/~grass/docs/tutorial\\_62/index.html](http://www.ing.unitn.it/~grass/docs/tutorial_62/index.html)

GRASS manual (in english)

[http://grass.itc.it/grass63/manuals/html63\\_user/full\\_index.html](http://grass.itc.it/grass63/manuals/html63_user/full_index.html)

