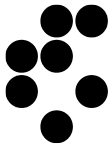
Institut "Jožef Stefan", Ljubljana, Slovenija
Department of Communication Systems (E-6)



IJS Delovno poročilo
DP-15291

GRASS-RaPlaT
Radio Planning Tool for GRASS

User Manual
V3.2

Igor Ozimek, Andrej Hrovat, Tomaž Javornik

Ljubljana, November 2025

Contents

1. GRASS-RAPLAT OVERVIEW 2

2. RAPLAT IN DETAILS 4

2.1. RUN A COMPLETE RADIO COVERAGE COMPUTATION – R.RAPLAT 5

2.1.1. Antenna types table 8

2.1.2. Computation region management 9

2.1.3. Parallel execution support 10

2.1.4. Reuse and Purge 11

2.1.5. Database support 11

2.1.6. Other parameters 12

2.2. RADIO PROPAGATION MODELS (ISOTROPIC ANTENNA) 12

2.2.1. r.fspl 12

2.2.2. r.hata 14

2.2.3. r.cost231 15

2.2.4. r.hataDEM 17

2.2.5. r.waik 20

2.3. ADD TRANSMISSION ANTENNA – R.SECTOR 23

2.4. CALCULATE COMPLETE COVERAGE – R.MAXPOWER 25

2.4.1. The input cell list file 27

2.4.2. The output data table 28

2.5. PREPARE CLUTTER MAP – R.CLUTCONVERT 29

3. GRASS AND RAPLAT INSTALLATION AND USAGE 32

3.1. GRASS INSTALLATION 32

3.2. GRASS GIS DATABASE 34

3.2.1. Creating a new GRASS location for RaPlaT 36

3.2.2. Using a pre-existing GRASS GIS database – demo database 40

3.2.3. GRASS maps for RaPlaT 44

3.3. RAPLAT INSTALLATION 45

3.3.1. Antennas 46

3.4. USING RAPLAT – RUNNING DEMOS 46

4. REFERENCES 50

Figures

Fig. 1: GRASS-RaPlaT block diagram 4
Fig. 2: Coverage by three antennas on one location (*r.hata*, 900 MHz) 7
Fig. 3: Path loss at 2 GHz computed with *r.fspl* 13
Fig. 4: Path loss at 900 MHz computed with *r.hata* 15
Fig. 5: Path loss at 2 GHz computed with *r.cost231* 16
Fig. 6: Basic concept of the hataDEM model 17
Fig. 7: Path loss at 2 GHz computed with *r.hataDEM* 19
Fig. 8: Path loss at 2 GHz computed with *r.waik* 22
Fig. 9: Path loss computed by *r.sector,* based on *r.hata* path loss (Fig. 4) 25
Fig. 10: Official land use map for the Ljubljana region 30
Fig. 11: The corresponding clutter map generated by *r.clutconvert* 31
Fig. 12: GRASS terminal window (text) interface 33
Fig. 13: GRASS GUI window – after installation 34
Fig. 14: Initial GRASS GIS database structure 35
Fig. 15: Initial GRASS GIS database directory structure 36
Fig. 16: Initial GRASS GIS database directory structure 36
Fig. 17: Creating location named Slovenia 37
Fig. 18: Setting of the cartographic projection 37
Fig. 19: Selecting a suitable cartographing projection for Slovenia 38
Fig. 20: Automatic creation of the *PERMANENT* mapset 39
Fig. 21: Creation of an additional (user) mapset 39
Fig. 22: GRASS session with the new "current" (working) mapset "igor" 40
Fig. 23: Adding demo GRASS GIS database (*grassdata_demo*) 41
Fig. 24: Automatically switching to the new GIS database (*grassdata_demo*) 42
Fig. 25: Radio signal coverage map (signal strength in dBm) with DEM in the background 43
Fig. 26: OpenStreetMap – Slovenia (https://www.openstreetmap.org/#map=9/46.1475/14.5486) 45
Fig. 27: Contens of the directory *raplat* 47
Fig. 28: New mapset *user1* 47
Fig. 29: Executing a demo RaPlaT computation 48

Tables

Table 1: An example of the cell list table 5
Table 2: Description of the cell list table columns 6
Table 3: An example of the antenna types table 9
Table 4: Parameters and their values for the Walfisch-Ikegami model 21
Table 5: Output data table format 28

Version 3.2 - What's new

Version 3.2 RaPlaT distribution supports and has been tested in the following Linux environment[1]:

- Kubuntu[2] 24.04 (as of October 30, 2025, fully updated, 24.04.3)
- GRASS GIS installed from the standard Ubuntu binary package distribution (repository) – version 8.3.2 (build date: 2024-03-31).

Compared to version 3.1, there are only some minor (more or less cosmetic) changes in two of the RaPlaT modules to accommodate for changes in the new (current) versions of the C compiler and Python 3 interpreter (stricter checking and warnings).

On the other hand, this User manual has been modified quite extensively, mostly to reflect the profound changes in the GRASS GUI version 8 design (compared to version 7).

Older versions (3.1 & 3.0):

Version 3.1 RaPlaT distribution supported and has been tested in the following environments:

1. Ubuntu 22.04 (as of March 30, 2023, fully updated, 22.04.2), with GRASS GIS installed from its standard Ubuntu binary package distribution (repository) – version 7.8.7.
2. Ubuntu 20.04 (as of March 31, 2023, fully updated, 20.04.6), with GRASS GIS installed from its standard Ubuntu binary package distribution (repository) – version 7.8.2.

RaPlaT version 3.1 is functionally identical to V3.0. The changes in V3.1 are:

- The *r.raplat* module has been converted to Python 3[3] (previously Python 2), with necessary corrections.
- The User Manual has been updated to correspond to the GRASS and RaPlaT installation on Ubuntu 22.04 and 20.04.

Version 3.1 distribution also includes the original V3.0 components (supporting Ubuntu 18.04 and 18.06) that have been changed in V3.1:

- The *r.raplat* module (v23apr2021, Python 2).
- User Manual V.3.0.

[1] Windows and MAC are not supported by RaPlaT.

[2] Kubuntu is a flavor of Ubuntu that uses the KDE desktop environment instead of the default GNOME. This choice is to a large degree a matter of personal (subjective) preference and should not influence the GRASS and RaPlaT functionality. In other words: everything described here should also work for Ubuntu 24.04 (with GNOME), but as of this moment it has not been tested yet.

[3] Python 2 is considered obsolete in Ubuntu 20.04, although it can still be installed from its standard Ubuntu repository (*sudo apt install python2*). Python 3 is officially supported in GRASS GIS starting from version 7.7.

1. GRASS-RaPlaT overview

GRASS GIS [1,2], shortly GRASS (*Geographic Resources Analysis Support System*), is a free Geographic Information System (GIS) software used for geospatial data management and analysis, image processing, graphics/maps production, spatial modeling, and visualization. It is available as prebuilt packets for various Linux distributions, MS Windows and MAC, as well as in source code.

RaPlaT (*Radio Planning Tool for GRASS*) [3,4,5] is an add-on for GRASS for radio signal coverage calculation. It uses the GRASS' support for geographic environment (terrain relief) and other GRASS functionalities (displaying, etc.) important for radio coverage computations and display.

RaPlaT comprises a set of C modules (small programs written in C, specifically for the GRASS environment) and Python modules (scripts, also written for the GRASS environment). They belong to the following groups:

1. A group of path loss model modules each calculating radio signal path loss according to a specific radio signal propagation model. The obtained raster map, which tells the path loss in [dB] at each point of the terrain surface, corresponds to a hypothetic isotropic transmission antenna with 0 dB gain. This group currently comprises the following modules:
    - *r.fspl* – Free Space Path Loss model,
    - *r.hata* – Okumura-Hata model,
    - *r.cost231* – COST 231 model,
    - *r.hataDEM* – modified Hata – Okumura-Hata DEM model,
    - *r.waik* – Walfish-Ikegami propagation model.
2. Module *r.sector*, which takes the isotropic path loss result calculated by a path loss model module, and modifies it according to the selected antenna characteristics (radiation pattern and gain), its position and orientation.
3. Module *r.MaxPower*, which calculates the received power at each raster point of the terrain surface for one or more transmission antennas (e.g. for a cellular communication network like GSM, UMTS or LTE). In case of multiple transmission antennas, it can compute various results, e.g. the maximum received power from any transmitter at each receive point (raster point on the terrain relief map). It can also build a data table (using e.g. MySQL or PostgreSQL) comprising the relevant data of a chosen number of strongest received signals at each receive point.
4. Script *r.raplat*, which can perform a complete computation by automatically calling the above modules. The user only uses *r.raplat* and does not need to deal directly with individual modules listed above.
5. Auxiliary module *r.clutconvert*.

The RaPlaT modules are distributed as source code for Linux environment only. They can be installed easily as add-ons to an existing GRASS installation using the GRASS command *g.extension*.

Most of the path loss model modules need only a DEM (*Digital Elevation Map*, i.e. a raster map describing the terrain profile) for their computations. Some modules (currently only *r.hataDEM*) need also a so-called *clutter map*, which describes the signal fading at each raster point due to the land use or type of vegetation (e.g. buildings, roads, forest, grass, rivers,

lakes, etc.). When using RaPlaT in a professional environment (e.g. by a mobile network operator), commercial DEM and clutter maps are normally available. For non-commercial use, various publicly available maps can be found on the internet, e.g. the NASA's SRTM (*Shuttle Radar Topography Mission*) DEM maps with global Earth coverage, which are based on radar measurements performed during a Space Shuttle mission in February 2000, [6,7].

2. RaPlaT in details

The main overall structure of the RaPlaT tools is depicted in Fig. 1. It consists of a number of path loss model modules (implementing different radio propagation models), the *r.sector* and *r.MaxPower* modules, and the *r.raplat* Python script that ties everything together. Input and output data are depicted in Fig. 1 as differently colored parallelograms – textual input and output files in orange, GRASS raster files in blue, and databases in yellow.

The user can call individual modules, however he/she would normally only call *r.raplat*, which in turn calls other modules as necessary.

The user defines the parameters of one or more radio transmitters together with the chosen path loss models in a cell list file, which is a simple data table in the CSV (*Comma-Separated Values*) format [8,9]. The list of all available antenna types is given in another CSV format file, which references the actual antenna data files written in the standard MSI text format [10]. The *r.raplat* script first executes the required path loss model modules for the given set of transmitters (as specified in the cell list file), continues with calling *r.sector* for all the transmission antennas and finishes with calling *r.MaxPower* for calculation of the overall radio signal coverage.

RaPLaT path loss model modules and *r.sector* require a DEM map (DEM – *Digital Elevation Map*), which describes the terrain relief. Some path loss models (*r.hataDEM* from the above modules) additionally need a clutter map that describes the signal loss due to the land-use (buildings, forests, lakes, etc.)

Fig. 1 depicts an additional module, *r.clutconvert*, which is used for creation of clutter maps (describing land-use-dependent signal loss) from general land-use GRASS raster maps.

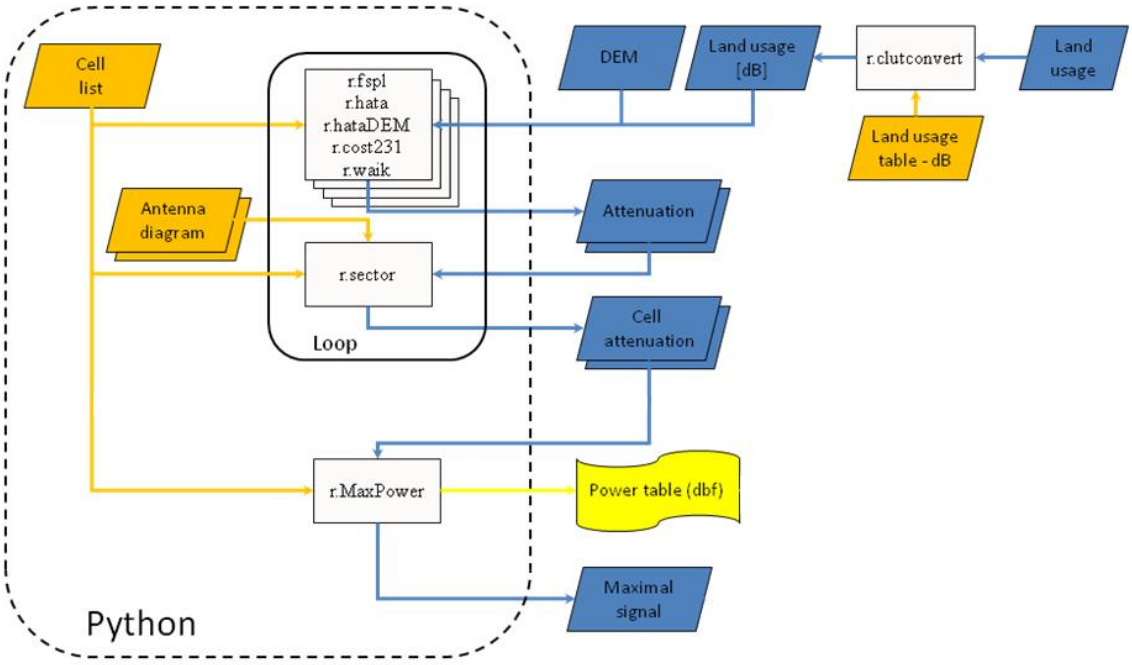


Fig. 1: GRASS-RaPlaT block diagram

GRASS modules generally work in the so called *current region*, which defines the geographic region extents and resolution (resampling of input maps, if necessary, is done automatically). The *r.raplat* script lets the user set the computation region independently of the current region (a temporal current region is established for the execution of the called modules). RaPlaT reduces the execution times of the path loss models and *r.sector* modules by additionally limiting computation to a circle with a given radius around each antenna. The points outside this area are assigned the *null* value (no signal received). The *r.raplat* script allows setting the radius independently for each antenna in the input cell list table (see below), or globally with the *radius_ovr* command line parameter.

Radio coverage computation requires a GRASS *location* with a cartographic projection with distances expressed in meters (e.g. the older Gauß-Krüger or the newer Transverse Mercator projection for Slovenia). It cannot work correctly in a *location* with the so called Latitude/Longitude pseudo projection, where locations and distances are expressed in angular degrees.

2.1. Run a complete radio coverage computation – r.raplat

A radio coverage computation could be accomplished by calling individual modules (described in details later in this document): isotropic path loss model modules, *r.sector*, and *r.MaxPower*. Such use would be quite awkward and demanding, therefore we created *r.raplat*, a Python script, which ties everything together and calls individual modules as necessary. The script gets the necessary information for radio coverage calculations from two tables written in the CSV text format, and from the *r.raplat* command line parameters.

RaPlaT can be used to calculate coverage by radio signals from multiple transmitters, as is the case with cellular networks (e.g. GSM radio network). The user describes the whole configuration in a cell list file ("cell list" here is actually a list of installed antennas with related data, as will be explained shortly). The file is in the CSV format and can be created with Apache OpenOffice or LibreOffice Calc (spreadsheet), but also with MS Excel in the MS Windows environment (*r.raplat* understands the peculiarities of the MS Excel CSV format including its European version; the RaPlaT tool itself is currently supported only on Linux).

The cell list file is specified with the *csv_file* command line parameter and contains a table, an example of which is shown in Table 1 (three transmit antennas on a single location).

Table 1: An example of the cell list table

cellName	antID	antType	antEast	antNorth	antHeightAG	antDirection	antElecTilt	antMechTilt	freq	power	radius	model	P1	P2	...	P11
IJS-A	1	COS-21	460697	99918	20	30	0	0	900	30	10	hata	urban			
IJS-B	2	COS-21	460697	99918	20	135	0	0	900	30	10	hata	urban			
IJS-C	3	COS-21	460697	99918	20	270	0	0	900	30	10	hata	urban			

The corresponding CSV file would be:

```
cellName,antID,antType,antEast,antNorth,antHeightAG,antDirection,antElecTilt,antMechTilt,freq,
power,radius,model,P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11
IJS-A,1,COS-21,460697,99918,20,30,0,0,900,30,10,hata,urban,,,,,,,,,
IJS-B,2,COS-21,460697,99918,20,135,0,0,900,30,10,hata,urban,,,,,,,,,
IJS-C,3,COS-21,460697,99918,20,270,0,0,900,30,10,hata,urban,,,,,,,,,
```

The first line contains the header (in the example above, it is split into two lines to fit on the page, but should actually be a single line). Each following line contains data for one transmission antenna. The *r.raplat* script parses this table according to a special data structure defined in the Python source code by the *cellTableDescrib* variable, which will not be explained here (see the source code, in Python). The data columns, their types and value constraints as defined by this structure are shown in Table 2.

Table 2: Description of the cell list table columns

Name	Type	Allowed values	Description
cellName	name	(see description)	Cell name (characters 'A'..'Z', 'a'..'z', numbers, '_', '-')
antID	id	1..999999	Antenna identification number
antType	antype	unconstrained	Antenna type
antEast	i	unconstrained	Antenna position – E-W in [m]
antNorth	i	unconstrained	Antenna position – N-S in [m]
antHeightAG	f	0.0..300.0	Antenna height above the terrain
antDirection	i	0..360	Antenna horizontal direction (0: northwards; positive: clockwise)
antElecTilt	i	0..10	Antenna electrical vertical tilt (downwards)
antMechTilt	i	-90..+90	Antenna vertical direction (positive: downwards)
freq	f	1..10000	Radio frequency in [MHz]
power	f	0.0..140.0	Transmission power in [dBm] (1mW..10kW)
radius	f	0.0..1000.0	Max. distance of the receiver in [km]
model	s	'hata', 'cost231', 'hataDEM', 'waik', 'fspl'	Radio signal path loss model
Parameters P1..P11 for the Hata model			
P1	s	'urban', 'suburban', 'open'	Area type for the Hata model
P2 .. P11	-	*(not used)*	
Parameters P1..P11 for the Cost231 model			
P1	s	'metropolitan', 'medium_cities'	Area type for the Cost231 model
P2 .. P11	-	*(not used)*	
Parameters P1..P11 for the hataDEM model			
P1	f	Unconstrained	Parameter A0 for the hataDEM model
P2	f	Unconstrained	Parameter A1 for the hataDEM model
P3	f	Unconstrained	Parameter A2 for the hataDEM model
P4	f	Unconstrained	Parameter A3 for the hataDEM model
P5 .. P11	-	*(not used)*	
Parameters P1..P11 for the Walfisch-Ikegami (waik) model			
P1	f	20..60	Parameter W0 (*Free space loss correction*)
P2	i	30..70	Parameter W1 (*Reduced base antenna height correction*)
P3	i	5..35	Parameter W2 (*Range correction*)
P4	i	3..15	Parameter W3 (*Street width correction*)
P5	i	3..25	Parameter W4 (*Frequency correction*)
P6	i	10..30	Parameter W5 (*Building height correction*)
P7	i	10..25	Parameter W6 (*Street width [m]*)
P8	i	20..50	Parameter W7 (*Distance between buildings [m]*)
P9	i	0..300	Parameter W8 (*Building height [m]*)
P10	i	0..180	Parameter PHI_Street (*Street orientation [deg]*)
P11	s	'metropolitan', 'medium_cities'	Area type
Parameters P1..P11 for the fspl (»free space«) model			
P1 .. P11	-	*(not used)*	

Type means:

- *name* : character string (see description in the table). Used for the cell names.
- *id* : integer value, similar to type *i* (see below) but values must be unique (the same value may not repeat). Used for antenna identification numbers in multi antenna systems (cellular networks).
- *antype* : character string, allowed characters: letters, numbers, ' ', '-', '/' and '.' in any order. Used to define the antenna types, according to antenna types defined in the antenna types table (see later).
- *i* : integer value with min and max limits. If the value of both min and max limits is 0, the value is unbounded. The decimal point/comma is not allowed.
- *f* : floating point value with min and max limits. If the value of both min and max limits is 0.0, the value is unbounded. The value can be written in the cell list file without the decimal point/comma.
- *s* : a word (character string) from a set of allowed words. The columns to the right can depend on this word (as defined by the *cellTableDescrib* variable; e.g. the P*n* columns depend on the word (model name) in the *model* column).
- - : arbitrary contents.

Empty lines are ignored. The character # as the first character of the cell name or the first character in a line has a special meaning: it marks the cell as a comment only, effectively disabling the cell. This is useful for simple and quick enabling/disabling of individual cells.

An example of radio coverage map (received signal strengths in [dBm]) for a system with three antennas on a single location is shown in Fig. 2. The antennas used here were not a real product but a mathematically created cosine type with half-power (-3 dB) beam width of about 30° and gain 0 dBd.

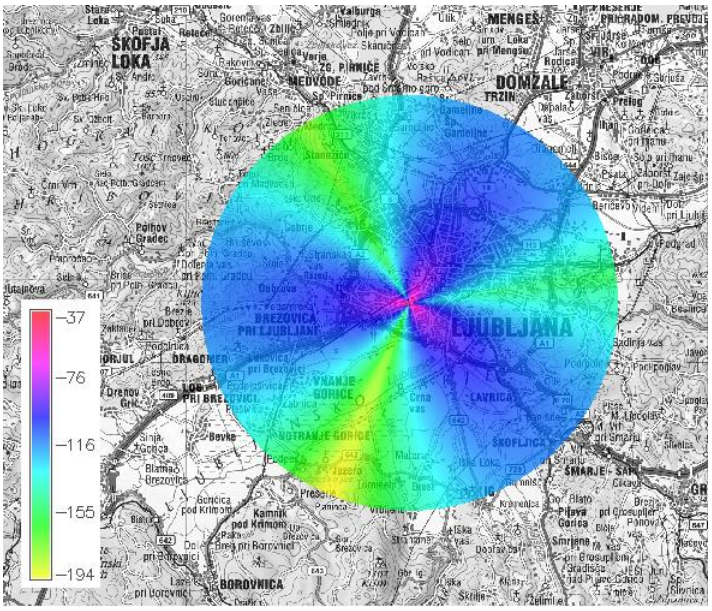


Fig. 2: Coverage by three antennas on one location (*r.hata*, 900 MHz)

- Description:

```
RaPlaT - raplat module (v04nov2025), RAdio PLanning Tool
```

- Usage:

```
r.raplat [-rpcx] csv_file=string [antmap_file=string] dem_map=string
  [clutter_map=string] [region=string] [rx_ant_height=value]
  [generate=string] [bandwidth=value] [rx_threshold=value] out_map=string
  [cellnum=value] [db_driver=string] [database=string] [out_table=string]
  [dbperf=value] [procnum=value] [freq_ovr=value] [radius_ovr=value]
  [model_ovr=string] [--overwrite] [--help] [--verbose] [--quiet]
  [--ui]
```

- Flags (-- flags are common GRASS command flags):

```
 -r   Reuse results from existing intermediate model/sector files
 -p   (purge) Delete all unused sector radio coverage files
 -c   (check) Test run without actually performing radio coverage computation
 -x   Write all maps (model/sector/output) also as xyz files (in the working directory)
```

- Parameters:

```
      csv_file   Radio cell/sector table in CSV format
   antmap_file   Antennas map file
                 default: $GISBASE/etc/radio_coverage/antenna_diagrams/antennamap.csv
       dem_map   DEM raster map for radio coverage simulation
   clutter_map   Clutter raster map (required for HataDEM model)
        region   Computation region (dem, current or region,raster,n,e,s,w,res (see
                 g.region))
                 default: current
 rx_ant_height   Receiver antenna height [m]
                 default: 1.5
      generate   Selection of the generated output contents
                 options: rss-max,coverage,rss-sum,rss-maxix,lte-rssi,
                          lte-rsrp,lte-rsrq,lte-cinr,lte-maxspecteff,
                          lte-maxthrput,lte-interfere
                 default: rss-max
     bandwidth   Bandwidth [MHz]
                 default: 5
  rx_threshold   Minimum received power [dBm] for radio signal coverage
       out_map   Simulated radio coverage - raster (output)
                 default: out_raster
       cellnum   Number of succesive path loss values to be written in the table
                 default: 5
     db_driver   Database driver
                 options: none,dbf,mysql,pg,sqlite,csv
                 default: none
      database   Database name
                 default: $GISDBASE/$LOCATION_NAME/$MAPSET/dbf
     out_table   Simulated radio coverage - db table (output)
                 default: out_db
        dbperf   Database insert performance (rows/INSERT; 99: special fast mode via CSV)
                 options: 1-99
                 default: 20
       procnum   Number of parallel processes (-1: automatic, 0: non-parallel)
                 default: -1
      freq_ovr   Radio frequency override [MHz]
    radius_ovr   Radius override [km]
     model_ovr   Model override (with parameters)
```

- Example (does not create a data table; single line command):

```
r.raplat csv_file=~/raplat/cell_list_ijs_hata.csv
dem_map=Slo_DEMsrtm_filled_100@PERMANENT
antmap_file=~/raplat/antenna_diagrams/antennamap.csv out_map=IJS_ABC_hata --o
```

2.1.1. Antenna types table

The list of available antenna types with corresponding parameters is given in a CSV format file. The *r.raplat* script reads the antenna type for each cell from the cell list table (CSV file, described above) and then uses the antenna types table to find the corresponding MSI file describing the antenna's characteristics. (The MSI format is described later in the *r.sector* chapter.)

The antenna types table file is specified with the *r.raplat*'s *antmap_file* command line parameter. The default path is *$GISBASE/etc/radio_coverage/antenna_diagrams/antennamap.csv*, where the GISBASE environment variable is set by GRASS and contains

the path to its program directory. An example of the antenna types table is shown in Table 3 (only one antenna is defined).

Table 3: An example of the antenna types table

antennaType	Frequency	frequencyLower	frequencyUpper	EDT	MSIfilename	technology
COS-21	1500	800	2200	0	COS_21	none

The CSV file must be in the standard format (the modified European MS Excel semicolon-separated-values format is not supported). The CSV file corresponding to Table 3 would be (generated by OpenOffice Spreadsheet):

```
"antennaType","frequency","frequencyLower","frequencyUpper","EDT","MSIfilename","technology"
"COS-21",1500,800,2200,0,"COS_21","none"
```

The first line contains the header; each following line contains the following data:

- *antennaType* – antenna type name, allowed character are letters, numbers, ' ', '-', '/' and '.' in any order,
- *frequency* – nominal frequency of the antenna in [MHz],
- *frequencyLower* – the lower frequency limit of the antenna in [MHz],
- *frequencyUpper* – the upper frequency limit of the antenna in [MHz],
- *EDT* – electrical tilt of the antenna in [°] (downwards), a non-negative integer value,
- *MSIfilename* – the name of the MSI file without the *.MSI* or *.msi* extension that describes the antenna characteristics for this particular combination of antenna type & frequency & electrical tilt,
- *technology* – used to describe the type of the radio communication technology the antenna is made for (e.g. GSM 900, GSM 1800, UMTS 2100); an arbitrary comment, not used for processing.

Empty lines are ignored. The character # as the first character of the antenna type or the first character in a line has a special meaning: it marks the line as a comment only.

An antenna type can support multiple frequency bands (nominal frequencies) and electrical tilts, with each combination having different characteristics (described by the corresponding MSI files). Therefore, the same antenna type can appear in the table multiple times. The *r.raplat* searches the table for rows with the required antenna type, electrical tilt and with the frequency range (defined by the lower and upper frequency limit) that includes the simulation radio frequency set by the *frequency* command line parameter of *r.raplat*. If multiple table rows fulfill these requirements, *r.raplat* takes the one with the antenna nominal frequency closest to the simulation radio frequency.

The MSI files must be located in the same directory with the antenna types table file or in any of its optional subdirectories. Subdirectories can be used for logically grouping MSI files and have no other meaning. The *r.raplat* automatically searches the whole directory subtree for the MSI files. The MSI filenames must be unique even if located in different subdirectories.

2.1.2. Computation region management

In general, GRASS modules (including the RaPlaT modules) perform computations in the so called current region, which can be set with the GRASS command *g.region*. A region is a rectangle defined by its geographic borders and resolution. The *r.raplat* script has a command

line parameter called *region* that allows user to specify a computational region for the radio coverage computation, and temporarily sets it as the current region during the computation. If signals can be received in this region that emanate from outside transmitters, the computational region is automatically enlarged to include those transmitters.

The *region* parameter allows setting the computation region in a few different ways:

- *region=current* – the existing current region is used as the computation region (this is the default setting),
- *region=dem* – the region of the DEM map (its extents and resolution) is used as the computation region,
- *region=region:saved_region_name* (or *region=region=saved_region_name*) – a prevoiously saved region is used as the computation region (the GRASS *g.region* command can be used to save a current region),
- *region=rast:raster_map* (or *region=rast=raster_map*) – a GRASS raster map region is used as the computation region (*region=dem* can be regarded as a special shorthand form of *region=rast:…*),
- *region=n:_,e:_,s:_,w:_,res:_* (or *'region=n=_ e=_ s=_ w=_ res=_'*), with _ standing for numerical values, sets the computation region by explicitly defining its extents and resolution; instead of all five values, any subset of them can be set (with the rest of them retaining the existing values).

The last three ways of computation region definition (i.e. with exception of *current* and *dem*) can be combined – e.g. the computation region can be defined with a raster map and then the resolution modified. The mechanism is the same as in the case of the GRASS *g.region* command because this command is actually used for region setting (after replacing ':' with '=' and ',' with ' '), so see *g.region* help for more detailed information.

After the computation region is defined with the above procedure, and before it is actually set and used, *r.raplat* performs some refinements:

- unifies north-south and east-west resolution by taking the latter (east-west) for both directions,
- rounds the resolution to the integer value (in [m]),
- rounds the region extents so that the values of pixel center coordinates are multiples of the resolution value, and that this new region exceeds the extents of the original one.

2.1.3. Parallel execution support

GRASS modules are generally single-thread processes that execute on a single processor core. This is also true for RaPLaT modules. However, with certain limitations it is possible to execute multiple modules in parallel on a multi-core processor [11,12]. To speed-up coverage computation for multiple-antenna communication networks, *r.raplat* is capable of calling and executing modules in parallel. The number of concurrently executing modules (path loss model modules in the first step, and *r.sector* in the second step) is defined by the value of the *procnum* command line parameter. By default (or setting *procnum*=-1, automatic mode) the number of concurrently executing modules equals the number of existing processor cores in the system. A positive value explicitly defines the number of modules to be executed in parallel. Even if it is set to 1, the underlying parallel scheduling and stdout/stderr buffering mechanisms are still active. The parallel mode of execution (and related supporting mechanisms) can be switched off completely by setting *procnum*=0.

2.1.4. Reuse and Purge

During the execution of *r.raplat*, intermediate GRASS maps are created by the path loss model modules and by *r.sector*. They are not deleted automatically and can optionally be reused for another similar coverage computation (these can be considered as a kind of caching of the intermediate results for future computations). By avoiding unnecessary and possibly lengthy re-computations of model path loss and *r.sector* intermediate results, the overall time required for a radio coverage computation can be reduced considerably.

The names of intermediate maps are generated automatically and contain important information that *r.raplat* needs to be able to reuse them automatically in another radio coverage computation. The names of the maps generated by the path loss model modules are built according to the following pattern:

```
_model_P1_..._Pmax_positionEast_positionNorth_heightAGL_radius_frequency
```

e.g.:

```
_hata_urban_460697_99918_20_10_900
```

The names of the maps generated by *r.sector* are based on the above pattern (*model* in the pattern below) extended with additional *r.sector* related information:

```
cellName-antennaID(model)_beamDirection_electricalTiltAngle_mechanicalAntennaTilt_antennaType
```

e.g.:

```
IJS-A-1_hata_urban_460697_99918_20_10_900_30_0_0_COS-21
```

By default, *r.raplat* ignores any existing intermediate maps and calculates everything anew. The user can request reusing existing maps by specifying flag *-r* ("reuse"), however this must be done with great caution. The intermediate map names do not contain information about the DEM and clutter maps and the computation region that were used during their creation. Therefore, the user must keep in mind that the cached maps may only be used if the DEM/clutter maps and the computation region have not changed.

The number of intermediate maps can become quite large, making a mess inside the user's mapset. When not needed any more, the user can request *r.raplat* to delete them by specifying flag *-p* (purge, all maps with the names following the above patterns will be deleted).

2.1.5. Database support

The *r.raplat* script supports saving the computed results into a database data table, which is performed by *r.MaxPower*. The related command line parameters *cellnum*, *db_driver*, *database*, *out_table* and *dbperf* are equivalent to those of *r.MaxPower* and are described there. By default (if *db_driver* is not defined or is set to *none*) no data table is created. The *r.raplat* script does not check which GRASS-supported database management systems are actually installed on the system. Instead, it has a fixed list of them: *none*, *dbf* (GRASS' own built-in database), *mysql* (MySQL), *pg* (PostgreSQL), *sqlite* (SQLite), and *csv* (does not save the data into a database data table but into a standard CSV-format file). Of course, MySQL and PostgresSQL can only be used if they are installed and the GRASS' support for them is also installed.

2.1.6. Other parameters

There are some more *r.raplat* command line parameters and flags that will be described here briefly.

The *freq_ovr*, *radius_ovr* and *model_ovr* parameters override per-cell settings in the previously described cell list table. While settings in the table are individual for each transmit antenna (cell), this parameters set the values globally, i.e. for all antennas. The *model_ovr* parameter expects a comma-delimited list (without spaces) consisting of the model name and its parameters as described in Table 1 (e.g. *model_ovr=hata,urban*).

The *rx_treshold*, *generate* and *bandwidth* parameters are equivalent to those of the *r.MaxPower* module and are used directly by that module. When *rx_treshold* is specified, the received signal is ignored at those raster points where its received strength (in [dBm]) falls below the threshold value. The *generate* parameter is used to compute other types of results instead of the default *rss-max* (the received signal strength of the strongest signal at each point on the map). Some of those computations need also the *bandwidth* parameter value. This is explained in details later in the *r.MaxPower* description.

The flag *-c* ("check") causes *r.raplat* to not call and execute modules. Instead, it only prints all the commands (module calls) that would be executed, and the contents of the related input cell list file generated by *r.raplat* for *r.MaxPower*. (The file itself is a temporary file and is automatically deleted when *r.raplat* completes its execution.)

2.2. Radio propagation models (isotropic antenna)

RaPLaT containes a number of modules that calculate radio signal path loss according to various path loss models. The result is a raster GRASS map with each point having the value of the signal fading in [dBm] at that point relative to the transmitter (no particular antenna is assumed yet, the situation corresponds to the isotropic radiation diagram with 0 dB gain).

2.2.1. r.fspl

The *r.fspl* module calculates the radio signal loss according to the free space model (FSPL – *Free Space Path Loss*), according to the equation (1), [13].

$$FSPL = 32.4 + 20\log R[km] + 20\log f[MHz] \quad (1)$$

where:

$FSPL$: loss in dB

R : distance between the transmitter and the receiver

f : transmission frequency in MHz

The model takes into account LOS (*Line of Sight*, i.e. the visibility between the transmitter and the receiver), but in general this is a very simplified theoretical model that works fine in empty space but does not give accurate results in real terrestrial propagation environments where the signal loss deviates from the free space "squared distance" law (it generally increases with distance with a higher exponent than 2).

The model allows changing of the default theoretical free-space exponent value of two by another explicitly specified value (usually > 2) which might better describe fading in a non-free-space environment. Fading offset (positive or negative) can also be specified.

An example of a path loss map obtained with *r.fspl* is shown in Fig. 3. (The transmitter is placed at the IJS location in Ljubljana, computation is limited to 10 km around the

transmitter, the actual command used is listed in the Example below. The same holds for the other models described in the next chapters.)

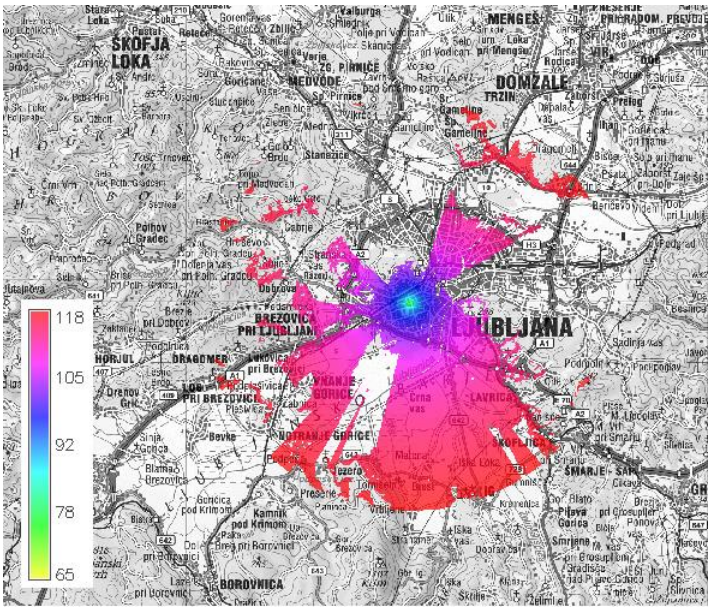


Fig. 3: Path loss at 2 GHz computed with *r.fspl*

- Description:

```
RaPlaT - Fspl module (v14aug2017)
```

- Usage:

```
r.fspl [-q] input_dem=name output=name [loss_exp=value]
  [loss_offset=value] coordinate=x,y [radius=value] [ant_height=value]
  [rx_ant_height=value] frequency=value [--overwrite] [--help]
  [--verbose] [--quiet] [--ui]
```

- Flags:

```
 -q   Quiet
```

- Parameters:

```
     input_dem   Name of input raster map
        output   Name for output raster map
      loss_exp   Exponent value (free-space-based fading model)
                 default: 2.0
   loss_offset   Offset value [dB] (free-space-based fading model)
                 default: 0.0
    coordinate   Base station coordinates
        radius   Computation radius [km]
                 default: 10
    ant_height   Transmitter antenna height [m]
                 default: 10
 rx_ant_height   Receiver antenna height [m]
                 default: 1.5
     frequency   Frequency [MHz]
```

- Example (single line command):

```
r.fspl input_dem=Slo_DEMsrtm_filled_100@PERMANENT output=fspl_ijs
coordinate=460697,99918 ant_height=20 frequency=2000 radius=10 --o
```

2.2.2. r.hata

The *r.hata* module implements the Okumura-Hata radio propagation empirical model [14]. It is one of the most widely used models for radio coverage estimation and is based on the empirically estimated rules (measured propagation data). It is valid for:

- carrier frequency: 150 – 1500 MHz,
- distance between transmitter and receiver: 1 – 20 km,
- effective BS (transmitter) antenna height: 30 – 200 m,
- effective MS (receiver) antenna height: 1 – 10 m.

It contains three sub-models, for urban, suburban and open geographic areas, as defined by the following equations:

$$L_U = 69.55 + 26.16 \log f[MHz] - 13.82 \log h[m] - C_H + (44.9 - 6.55 \log h[m]) \log R[km] \quad (2)$$

$$L_{SU} = L_U - 2\left(\log \frac{f[MHz]}{28}\right)^2 - 5.4 \quad (3)$$

$$L_O = L_U - 4.78(\log f[MHz])^2 + 18.33 \log f[MHz] - 40.94 \quad (4)$$

$$C_H = 0.8 + (1.1 \log f[MHz] - 0.7) h_M[m] - 1.56 \log f[MHz] \quad (5)$$

where:

L_U, L_{SU}, L_O : loss in dB for urban, suburban and open environments, respectively

h : difference between the transmitter and receiver antenna heights

h_M : receiver antenna height above the ground

C_H : correction factor related to the receiver antenna height

R : distance between the transmitter and the receiver

f : transmission frequency in MHz

The rate of the signal loss with the distance depends on the antenna height. For a very high antenna, it approximates the loss in empty space (the "squared distance" law, 20 dB/decade). The model ignores terrain configuration (relief, LOS), which is its main drawback, and the loss due to land use (clutter map). The model can give useful results if there are no major obstacles between the receiver and the transmitter.

The model supports also a special experimental *inverse* mode activated by the flag *-i*, which can be used for a transmitter localization application. In this mode, the receiver is placed at the specified location, and the resulting map shows signal fading for a transmitter placed at each point on the map.

An example of a path loss map obtained with *r.hata* is shown in Fig. 4.

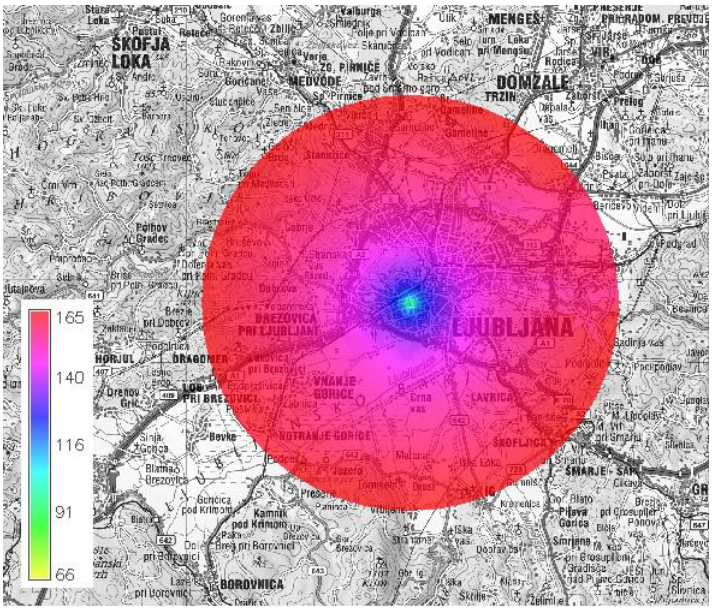


Fig. 4: Path loss at 900 MHz computed with *r.hata*

- Description:

```
RaPlaT - Hata module (v20jul2017)
```

- Usage:

```
r.hata [-qi] input_dem=name output=name [area_type=string]
  coordinate=x,y [radius=value] [ant_height=value] [rx_ant_height=value]
  frequency=value [--overwrite] [--help] [--verbose] [--quiet] [--ui]
```

- Flags:

```
 -q   Quiet
 -i   Inverse mode (RX and TX roles exchanged)
```

- Parameters:

```
     input_dem   Name of input raster map
        output   Name for output raster map
     area_type   Area type
                 options: urban,suburban,open
                 default: urban
    coordinate   Base station coordinates, or receiver location in inverse mode
        radius   Computation radius [km]
                 default: 10
    ant_height   Transmitter antenna height [m]
                 default: 10
 rx_ant_height   Receiver antenna height [m]
                 default: 1.5
     frequency   Frequency (MHz)
```

- Example (single line command):

```
r.hata input_dem=Slo_DEMsrtm_filled_100@PERMANENT output=hata_ijs
coordinate=460697,99918 ant_height=20 frequency=900 radius=10 --o
```

2.2.3. r.cost231

The *r.cost231* module implements the COST 231 empirical model, which extends the Okumura-Hata model to the 1500 – 2000 MHz band [15]. It is valid for:

- carrier frequency: 1500 – 2000 MHz,
- distance between transmitter and receiver: 1 – 20 km,

- effective BS (transmitter) antenna height: 30 – 200 m,
- effective MS (receiver) antenna height: 1 – 10 m.

The model is based on the Hata model for the suburban areas:

$$L[dB] = 46.33 + 33.9 \log f[MHz] - 13.82 \log h[m] - a(h_r) + \\ + (44.9 - 6.55 \log h[m]) \log d[km] + C \tag{6}$$

where:

C : =0 for medium-sized cities and suburban areas, =3 for large cities' centers

h : difference between the transmitter and receiver antenna heights

h_r : receiver antenna height above the ground

d : horizontal distance between the transmitter and the receiver

f : transmission frequency in MHz

The height correction factor $a(h_r)$ is given by:

$$a(h_r) = (1.1 \log f[MHz] - 0.7) h_r[m] - (1.56 \log f[MHz] - 0.8) \tag{7}$$

The model is adjusted for higher transmission frequencies. It is mostly suitable for medium-sized and large cities and assumes the transmit (base station) antenna to be positioned above the surrounding buildings. The model only partially takes into account the terrain configuration (the effective height h in the equation (6)) and ignores the signal loss behind large obstacles.

An example of a path loss map obtained with *r.cost231* is shown in Fig. 5.

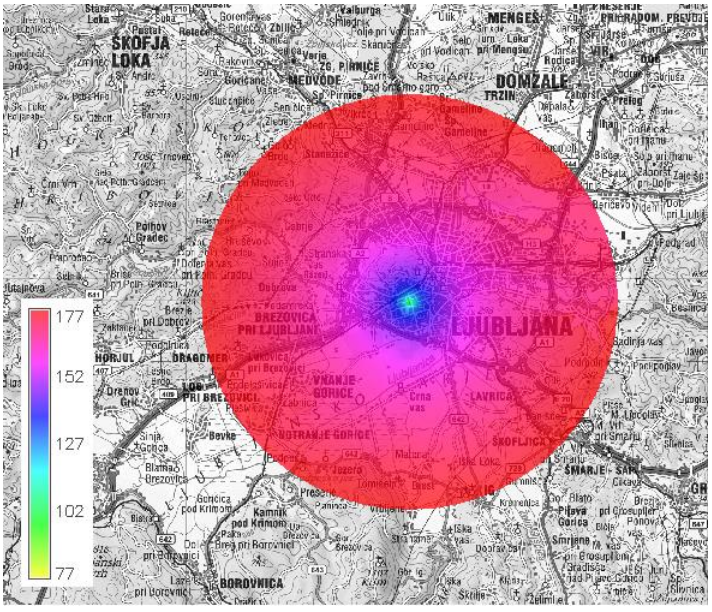


Fig. 5: Path loss at 2 GHz computed with *r.cost231*

- Description:

```
RaPlaT - Cost231 module (v01aug2017)
```

- Usage:

```
r.cost231 [-q] input_dem=name output=name coordinate=x,y
  [ant_height=value] [radius=value] [area_type=string] frequency=value
```

```
[--overwrite] [--help] [--verbose] [--quiet] [--ui]
```

- Flags:

```
-q   Quiet
```

- Parameters:

```
 input_dem   Name of input raster map
    output   Name for output raster map
coordinate   Base station coordinates
ant_height   Transmitter antenna height [m]
             default: 10
    radius   Computation radius [km]
             default: 10
 area_type   Area type
             options: metropolitan,medium_cities
             default: medium_cities
 frequency   Frequency [MHz]
```

- Example (single line command):

```
r.cost231 input_dem=Slo_DEMsrtm_filled_100@PERMANENT output=cost231_ijs
coordinate=460697,99918 ant_height=20 frequency=2000 radius=10 --o
```

2.2.4. r.hataDEM

The *r.hataDEM* module implements a modified/extended Okumura-Hata model. The radio signal loss depends on the transmission radio frequency, the distance between the transmitter and the receiver, the height of the transmit and receive antennas, and also on the terrain profile, land use and earth surface curvature. The model is valid for:

- carrier frequency: 150 MHz – 2 GHz (usable also for 2.1/2.4/2.6 GHz),
- distance between transmitter and receiver: 200 m – 100 km,
- effective BS (transmitter) antenna height: 20 – 200 m,
- effective MS (receiver) antenna height: 1 – 5 m.

The basic concept of the model is shown in Fig. 6.

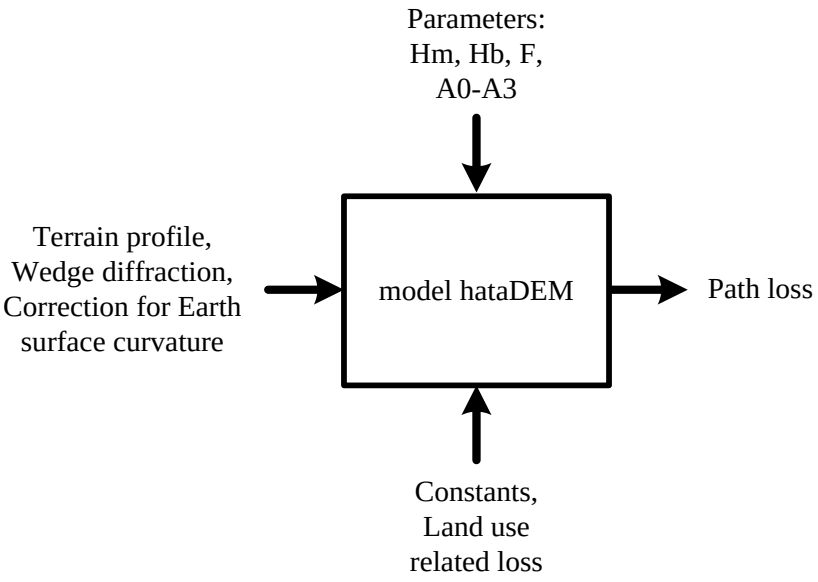


Fig. 6: Basic concept of the hataDEM model

The general path loss equation of the model is:

$$L[dB] = HOA[dB] + mk + \sqrt{(\alpha KDFR)^2 + (JDFR)^2} \tag{8}$$

where:

HOA : Okumura-Hata equation for "open" areas

mk : land-use related signal loss at the receiver location in [dB]

KDFR : contribution of wedge diffraction in [dB]

α : parameter related to KDFR

JDFR : diffraction loss due to the Earth surface curvature in [dB]

The Okumura-Hata path loss as defined by this model is:

$$\begin{aligned} HOA[dB] = A0 + A1 \cdot \log d[km] + A2 \cdot \log Heff[m] + A3 \cdot \log d[km] \cdot \log Heff[m] - \\ - 3.2[\log(11.75 \cdot Hm[m])]^2 + 44.49 \cdot \log f[MHz] - 4.78 \cdot (\log f[MHz])^2 \end{aligned} \tag{9}$$

where:

A0-A3 : model tuning parameter

Heff : difference between the transmitter and receiver antenna heights

Hm : receiver antenna height above the ground

d : horizontal distance between the transmitter and the receiver

f : transmission frequency in MHz

We implemented the single-wedge loss as:

$$L_{ke}[dB] = -20 \cdot \log \frac{1}{\pi \cdot \nu \cdot \sqrt{2}} \tag{10}$$

$$\nu = h \cdot \sqrt{\frac{2(d_1 + d_2)}{\lambda \cdot d_1 \cdot d_2}} \tag{11}$$

where:

h : height of the wedge above the direct line between transmitter and receiver

d_1, d_2 : the distances of the mobile and base stations from the wedge

Since *r.hataDEM* was originally intended for calculation on smaller geographic areas (cellular networks with transmitter-receiver distances of up to 35 km), it ignores the effect of the Earth surface curvature. Additionally, we fixed the value of parameter α to α=1.

The model supports also a special experimental *inverse* mode activated by the flag *-i*, which can be used for a transmitter localization application. In this mode, the receiver is placed at the specified location, and the resulting map shows signal fading for a transmitter placed at each point on the map.

By default the model uses a clutter map, which specifies additional fading at each point on the map (depending on the terrain type). However, by specifying *clut_mode=none*, this additional fading can be ignored and the clutter map need not be specified. There is also an additional experimental clutter mode *clut_mode=tx*, where the terrain-dependent additional fading is based on the terrain at the transmitter location (instead of at the receiver location), which might be useful in a non-standard reversed configuration of a low transmitter antenna and a high receiver antenna.

An example of a path loss map obtained with *r.hataDEM* is shown in Fig. 7.

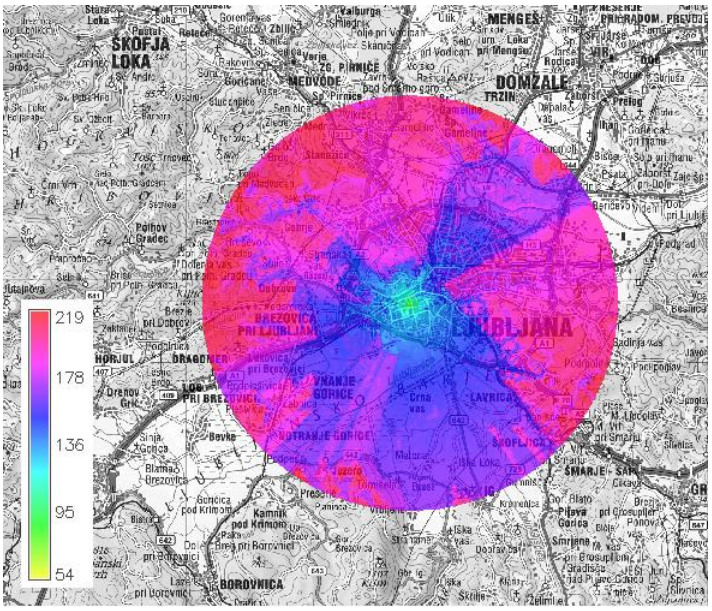


Fig. 7: Path loss at 2 GHz computed with *r.hataDEM*

- Description:

```
RaPlaT - HataDEM module (v07dec2018)
```

- Usage:

```
r.hataDEM [-qi] input_dem=name [clut_mode=string] [clutter=name]
  output=name a0=value a1=value a2=value a3=value coordinate=x,y
  [radius=value] [ant_height=value] [rx_ant_height=value] frequency=value
  [--overwrite] [--help] [--verbose] [--quiet] [--ui]
```

- Flgas:

```
 -q   Quiet
 -i   Inverse mode (RX and TX roles exchanged)
```

- Parameters:

```
     input_dem   Name of input raster map
     clut_mode   Clutter usage
                 options: rx,tx,none
                 default: rx
       clutter   Name of clutter raster map with path loss coefficients
                 default:
        output   Name for output raster map
            a0   Parameter a0
            a1   Parameter a1
            a2   Parameter a2
            a3   Parameter a3
    coordinate   Base station coordinates, or receiver location in inverse mode
        radius   Computation radius [km]
                 default: 10
    ant_height   Transmitter antenna height [m]
                 default: 10
 rx_ant_height   Receiver antenna height [m]
                 default: 1.5
     frequency   Frequency [MHz]
```

- Example (single line command):

```
r.hataDEM input_dem=Slo_DEMsrtm_filled_100@PERMANENT
clutter=Slo_Clut10dB_100@PERMANENT output=hataDEM_ijs coordinate=460697,99918
ant_height=20 frequency=2000 radius=10 a0=42 a1=42 a2=-12 a3=0.1 --o
```

2.2.5. r.waik

The *r.waik* module implements the Walfisch-Ikegami semi-deterministic model for path loss computation in microcells. It was developed in the framework of the COST 231 project [15] and is based on the Walfisch-Bertoni [16] and Ikegami [17] models. The model computes path loss in two different ways, based on LOS (*Line of Sight*). It is valid for:

- carrier frequency: 800 – 2000 MHz,
- distance between transmitter and receiver: 20 m – 5 km,
- the receiver and transmitter height constraints are different for LOS and NLOS cases – see below.

In the LOS case (transmitter-receiver visibility), the loss within the street canyon is defined as:

$$L[dB] = 42.64 + 26 \log d[km] + 20 \log f[MHz], \quad d[km] \geq 0{,}02 \tag{12}$$

The first constant corresponds to the empty space loss at the distance of 20 m. The transmitter antenna height must be at least 30 m, and there should be no obstacles in the first Fresnel zone. The signal loss is exponential with the distance, the exponent value is 2.6.

In the NLOS case (no direct visibility between the transmitter and the receiver), the model uses the following parameters:

- transmitter height: h_t (4 m to 50 m),
- receiver height: h_r (1 m to 3 m),
- buildings height: h_{roof} (3 m × number of floors plus 3 m for gabled roofs and 0 m for flat roofs),
- the transmitter antenna height above the roof height: $\Delta h_t = h_t - W8$,
- the receiver antenna height below the roof height: $\Delta h_r = W8 - h_r$,
- spacing between buildings: b (if no data is available, the recommended value is between 20 m and 50 m),
- street width: w (if no data is available, the recommended value is $b/2$),
- incident angle of radio rays: ϕ (if no data is available, the recommended value is 90°).

The path loss is:

$$L[\mathrm{dB}] = \begin{cases} L_0 + L_{rts} + L_{msd}, & L_{rts} + L_{msd} \geq 0 \\ L_0, & L_{rts} + L_{msd} < 0 \end{cases} \tag{13}$$

It consists of three components:

- the free space loss L_0,
- the rooftop-to-street diffraction loss L_{rts},
- the multiple screen diffraction loss L_{msd}.

The free space loss is:

$$L_0 = W0 + 20 \log(d) + 20 \log(f) \tag{14}$$

The rooftop-to-street diffraction loss is:

$$L_{rts} = -8.2 - W3 \log(W6) + W4 \log(f) + W5 \log(\Delta h_r) + L_{11} \tag{15}$$

where the orientation-related loss is:

$$L_{11} = \begin{cases} -10 + 0.354\ \phi, & 0 \le \phi < 35° \\ 2.5 + 0.075\ (\phi - 35°), & 35° \le \phi 55° \\ 4.5 - 0.11\ (\phi - 55°), & 55° \le \phi \le 90° \end{cases} \quad (16)$$

The multiple screen diffraction loss is:

$$L_{msd} = L_{21} + k_a + k_d\ \log(d) + k_f\ \log(f) - 9\log(W7) \quad (17)$$

where the shadowing gain is:

$$L_{21} = \begin{cases} -18\ \log(1 + \Delta h_t) & h_t > h_{roof} \\ 0 & h_t \le h_{roof} \end{cases} \quad (18)$$

The k_a and k_d parameters depend on the path length d and the transmitter height above the roofs:

$$k_a = \begin{cases} W1, & h_t \ge h_{roof} \\ W1 - 0.8\ (h_t - h_{roof}), & h_t < h_{roof} \wedge d \ge 0.5\ m \\ W1 - 0.4\ d(h_t - h_{roof}), & h_t < h_{roof} \wedge d < 0.5\ m \end{cases} \quad (19)$$

$$k_d = \begin{cases} W2, & h_t \ge h_{roof} \\ W2 - \dfrac{15\ (h_t - h_{roof})}{h_{roof}}, & h_t < h_{roof} \end{cases} \quad (20)$$

Parameter k_a represents the increase of path loss when the transmitter is located below the roof levels, while parameters k_d and k_f represent the path loss due to distance and frequency. The latter is defined by

$$k_f = -4 + k_{f1}\left(\frac{f}{925} - 1\right) \quad (21)$$

The value of k_{f1} is 1.5 for city centers, and 0.7 elsewhere.

Parameters W0-W8 should be set according to the recommended values given in Table 4.

Table 4: Parameters and their values for the Walfisch-Ikegami model

Parameter	Description	Value range	Default value
W0	Free space loss correction	20 – 60	32.5
W1	Reduced base antenna height correction	30 – 70	54
W2	Range correction	5 – 35	10
W3	Street width correction	3 – 15	10
W4	Frequency correction	3 – 25	10
W5	Building height correction	10 – 30	20
W6	Width of roads [m]	(rec. W7 / 2)	15
W7	Building separation [m]	(rec. 20 – 50)	30
W8	Height of buildings [m]	–	12

The COST 231 Walfish-Ikegami propagation model provides good path loss estimates if the transmission antenna is located above the roof level. If it is located near the ground level, the estimates are bad because the model does not take into account the waveguide effect of the large city street canyons.

An example of a path loss map obtained with *r.waik* is shown in Fig. 8.

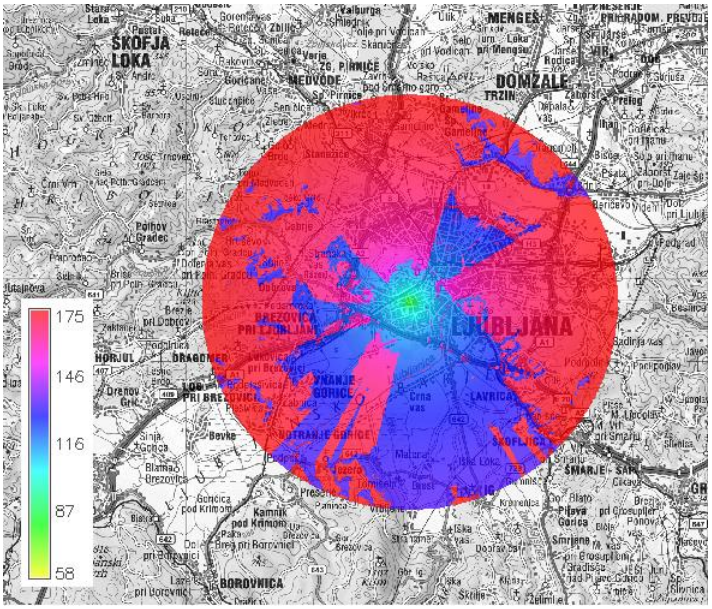


Fig. 8: Path loss at 2 GHz computed with *r.waik*

- Description:

```
RaPlaT - Walfish-Ikegami module (v07dec2018)
```

- Usage:

```
r.waik [-q] input_dem=name output=name coordinate=x,y
  [ant_height=value] frequency=value [radius=value]
  [free_space_loss_correction=value] [bs_correction=value]
  [range_correction=value] [street_width_correction=value]
  [frequency_correction=value] [building_height_correction=value]
  [street_width=value] [distance_between_buildings=value]
  [building_height=value] [phi_street=value] [area_type=string]
  [--overwrite] [--help] [--verbose] [--quiet] [--ui]
```

- Flags:

```
 -q   Quiet
```

- Parameters:

```
                 input_dem   Name of input raster map
                    output   Name for output raster map
                coordinate   Base station coordinates
                ant_height   Transmitter antenna height [m]
                             default: 10
                 frequency   Frequency [MHz]
                    radius   Computation radius [km]
                             default: 10
 free_space_loss_correction   Free space loss correction W0 (20-60)
                             default: 32.5
             bs_correction   Reduced base antenna height correction W1 (30-70)
                             default: 54
          range_correction   Range correction W2 (5-35)
                             default: 10
    street_width_correction   Street width correction W3 (3-15)
                             default: 10
       frequency_correction   Frequency correction W4 (3-25)
```

```
                            default: 10
 building_height_correction   Building Height Correction W5 (10-30)
                            default: 20
              street_width   Street width W6 [m]
                            default: 15
 distance_between_buildings   Distance between buildings W7 [m]
                            default: 30
           building_height   Building height W8 [m]
                            default: 12
                phi_street   Street orientation [deg]
                            default: 90
                 area_type   Area type
                            options: metropolitan,medium_cities
                            default: medium_cities
```

- Example (single line command):

```
r.waik input_dem=Slo_DEMsrtm_filled_100@PERMANENT output=waik_ijs
coordinate=460697,99918 ant_height=20 frequency=2000 radius=10 --o
```

2.3. Add transmission antenna – r.sector

The path loss model modules described so far compute path loss for the case of isotropic transmission (a hypothetical isotropic antenna with 0 dB gain in all directions), without considering actual transmission antenna characteristics, position and orientation. The next step is to apply the antenna radiation pattern, which is the task of the *r.sector* module. The antenna radiation pattern and other data must be given in the MSI Planet Antenna File Format [10]. This is a text format with the following structure:

```
NAME <name>
MAKE <make>
FREQUENCY <frequency>
H_WIDTH <h_width>
V_WIDTH <v_width>
FRONT_TO_BACK <front_to_back>
GAIN <gain>
TILT <tilt>
POLARIZATION <polarisation>
COMMENT <comment>
HORIZONTAL 360
0 <0H>
…
359 <359H>
VERTICAL 360
0 <0V>
…
359 <359V>
```

The variables are:

```
NAME            Name of the antenna
MAKE            Name of the manufacturer
FREQUENCY       Frequency in MHz
H_WIDTH         Opening angle in the horizontal plane between the -3 dB points
V_WIDTH         Opening angle in the vertical plane between the -3 dB points
FRONT_TO_BACK   Front to back ratio in dB
GAIN            Antenna gain in dBd; when in dBi, this must be specified
TILT            Electrical tilt of the main beam in degrees
POLARIZATION    Horizontal, vertical, +45 or -45
COMMENT         Comment
0H..359H        Horizontal gain data points per horizontal angle relative to
                maximum gain being zero. Minus sign is not used with these
                values, the gain values are assumed to be negative.
```

```
0V..359V        Vertical gain data points per vertical angle relative to maximum
                gain being zero. Minus sign is not used with these values, the
                gain values are assumed to be negative.
```

In practice, MSI files usually use only a subset of the parameters listed above, e.g. NAME, FREQUENCY, GAIN in dBd (default) or dBi, TILT, COMMENT, and of course the HORIZONAL 360 and VERTICAL 360 sections. Besides, the TILT parameters does not necessarily specify the actual electrical tilt value to which the radiation pattern corresponds, but is often assigned no value or the keyword 'ELECTRICAL' (for antennas not having or having electrical tilt option, respectively).

An actual MSI file could look like this (this particular file does not describe a real physical antenna but a mathematically generated one with the cosN radiation pattern):

```
NAME COSN21
FREQUENCY 2140
GAIN 19 dBd
TILT ELECTRICAL
COMMENT simple cos^4 antenna diagram
HORIZONTAL 360
0 0.0000
1 0.0139
2 0.0556
…
357 0.1251
358 0.0556
359 0.0139
VERTICAL 360
0 0.0000
1 0.0139
2 0.0556
…
357 0.1251
358 0.0556
359 0.0139
```

The *r.sector* module only reads and uses the GAIN parameter value expressed in dBd or dBi (i.e. relative to a dipole or isotropic antenna, respectively; dBd is default, x [dBd] = (x + 2.15) [dBi]), and the pattern definition specified in the HORIZONTAL and VERTICAL sections. It calculates the 3-D radiation pattern based on the antenna's given horizontal and vertical patterns, its gain, and its physical position and direction. It then generates an output path loss raster map by applying this pattern to the isotropic path loss raster map previously computed by a path loss model module.

Fig. 9 shows an example of a path loss map calculated by *r.sector*, using the path loss map previously computed by *r.hata* (shown in Fig. 4) and the just mentioned artificial cosN-type antenna. The antenna is directed 30° eastwards (north is the reference, positive values correspond to the clockwise rotation).

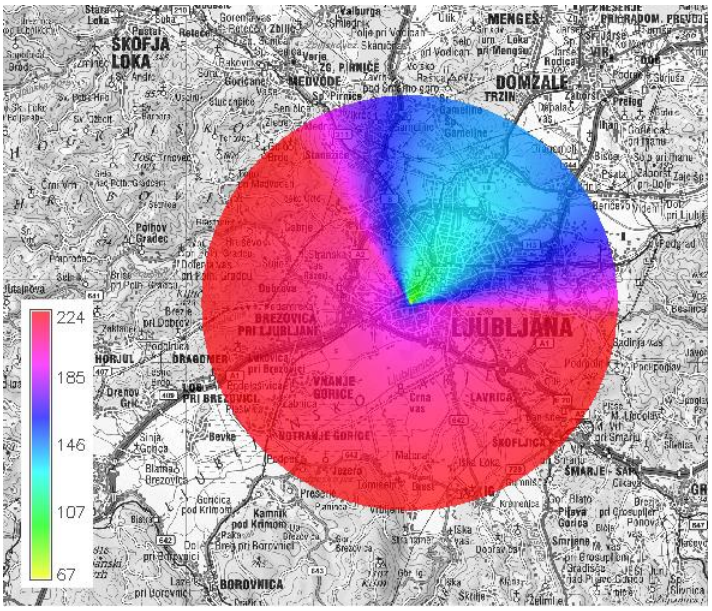


Fig. 9: Path loss computed by *r.sector*, based on *r.hata* path loss (Fig. 4)

- Description:

```
RaPlaT - Sector module (v06dec2018)
```

- Usage:

```
r.sector [-q] pathloss_raster=name input_dem=name output=name
  east=value north=value [radius=value] ant_data_file=string
  height_agl=value beam_direction=value mech_tilt=value
  [rx_ant_height=value] [--overwrite] [--help] [--verbose] [--quiet]
  [--ui]
```

- Flags:

```
 -q   Quiet
```

- Parameters:

```
 pathloss_raster   Name of isotropic antenna pathloss raster map
       input_dem   Name of elevation model raster map - required for transmitter
                   height determination
          output   Name for output raster map
            east   Easting coordinate
           north   Northing coordinate
          radius   Computation radius [km]
                   default: 10
    ant_data_file   Antenna data file
       height_agl   Above ground level height [m]
   beam_direction   Beam direction [deg]
        mech_tilt   Mechanical antenna tilt [deg]
    rx_ant_height   Receiver antenna height [m]
                   default: 1.5
```

- Example (single line command):

```
r.sector pathloss_raster=hata_ijs input_dem=Slo_DEMsrtm_100@PERMANENT
output=sector_hata_ijs ant_data_file=~/raplat/antenna_diagrams/_demo_/COS_21.MSI
beam_direction=30 mech_tilt=0 height_agl=20 radius=10 east=460697 north=99918 --o
```

2.4. Calculate complete coverage – r.MaxPower

The *r.MaxPower* module calculates the received radio signal strength(s) from one or more transmitters (transmit antennas). It does that by taking the path loss raster map(s) produced by

r.sector and applying the corresponding transmission power(s). It obtains the list of all input path loss raster maps with corresponding transmission powers (for all the transmission antennas, here also called *cells*) in a CSV format text file. By default (*generate=rss-max*) the module produces a raster map file containing the received strength of the strongest received signal for each raster point. If the *rx_treshold* parameter value is specified, the signals with lower received strengths are ignored.

The *generate* parameter can be used to specify different output results instead of the default maximum received signal strength (*rss-max*):

- *coverage* – a simple coverage area map is generated instead of the received signal strength map, with value of 1.0 for the received signal above a given threshold and 0.0 elsewhere; the *rx_threshold* parameter must be specified
- *rss-sum* – outputs the sum of the received signals strengths at each point of the map (instead of the strongest received signal strength)
- *rss-maxix* – generates a map containing the index of the strongest received transmit cell (antenna) at each point on the map; cells are indexed (numbered) sequentially according to their position in the cell list file (see the *r.raplat* description above)
- *lte-rssi* – LTE received signal strength
- *lte-rsrp* – LTE received signal representative power
- *lte-rsrq* – LTE received signal representative quality
- *lte-cinr* – LTE max CINR, interference free
- *lte-maxspecteff* – LTE max. spectral efficiency considering only AWGN
- *lte-maxthrput* – LTE max. throughput
- *lte-interfere* – LTE interference in dBm

LTE stands for Long-Term Evolution (4G) mobile radio network technology. All LTE computations require the *bandwidth* parameter to be specified.

In addition to the raster map, *r.MaxPower* can generate a data table containing data about a certain number (user selectable, parameter *cell_num*) of the strongest received signals at each raster point, suitable for further processing by other non-GRASS tools. It can be written to a CSV format file or stored in a standard database supported by GRASS (e.g. MySQL, PostgreSQL, SQLite and also the GRASS' own built-in DBF). The data table generation is activated by specifying the *driver* parameter with a value other than *none* (*dbf* for GRASS' DBF, *mysql* for MySQL, *pg* for PostgreSQL, *sqlite* for SQLite, *csv* for CSV file).

- Description:

```
RaPlaT - MaxPower module (v04nov2025)
```

- Usage:

```
r.MaxPower [-q] cell_input=string [generate=string]
  [rx_threshold=value] [chan_type=string] [bandwidth=value] output=name
  [table=string] [driver=string] [database=string] [cell_num=value]
  [dbperf=value] [--overwrite] [--help] [--verbose] [--quiet] [--ui]
```

- Flags:

```
 -q   Quiet
```

- Parameters:

```
   cell_input   Cells data table
     generate   Selection of the generated output contents
                options: rss-max,coverage,rss-sum,rss-maxix,lte-rssi,
                         lte-rsrp,lte-rsrq,lte-cinr,lte-maxspecteff,
                         lte-maxthrput,lte-interfere
                default: rss-max
```

```
  rx_threshold   Minimum received power [dBm] for radio signal coverage
                 default: -999
     chan_type   Channel type - Gaussian or Rayleigh (currently only Gaussian)
                 options: gaussian
                 default: gaussian
     bandwidth   Bandwidth [MHz] (required for LTE computations)
                 default: 5
        output   Name for output raster map
         table   Table name
        driver   Driver name
                 options: dbf,mysql,odbc,ogr,pg,sqlite,none,csv
                 default: none
      database   Database name
                 default: $GISDBASE/$LOCATION_NAME/$MAPSET/dbf/
      cell_num   Number of successive path loss values to be written in the table
                 default: 5
        dbperf   Database insert performance (rows/INSERT; 99: special fast mode via CSV)
                 options: 1-99
                 default: 20
```

- Examples (single line commands; the first one does not create a data table, but a dummy values for the *table* parameter must be specified anyway; the second one creates a DBF data table named *ijs_a_hata* in the default GRASS *dbf* folder within the user's mapset; the third one creates a SQLite data table named *ijs_a_hata* in the SQLite database (folder) *./sqlite*):

```
r.MaxPower cell_input=~/raplat/pwmx_cell_list_hata output=IJS_A_hata table=ijs_a_hata
cell_num=5 --o

r.MaxPower cell_input=~/raplat/pwmx_cell_list_hata output=IJS_A_hata table=ijs_a_hata
driver=dbf dbperf=1 cell_num=5 --o

r.MaxPower cell_input=~/raplat/pwmx_cell_list_hata output=IJS_A_hata database=./sqlite
table=ijs_a_hata driver=sqlite dbperf=1 cell_num=5 --o
```

2.4.1. The input cell list file

The cell list file ("Cells data table") is specified by the *cell_input* parameter. It is a text file in a CSV-like (actually "Semicolon-Separated Values") format, where each line contains data for a transmission antenna (here also called *cell*):

<cell_name>;<antenna_index>;<sector-raster-map_name>;<transmit-power>;<model-with-parameters>

There can be a single cell, or a number of them (e.g. in the case of a cellular radio network). No header line is used, and no empty lines are allowed (including at the end of the file).

The only important columns are <*sector-raster-map_name*> and <*transmit-power*>. Other columns are only informal and could be empty, they are written to the data table along with the calculated received powers. Their purpose is:

- <*cell_name*> : an (arbitrary) cell name that helps the user identify the cell identity/location,
- <*antenna_index*> : antenna index for uniquely identifying each antenna (cell) in the system; there can be more than one antenna in a cell (e.g. two antennas might be connected in parallel to a single transmission signal via a power splitter to obtain a required transmission pattern),
- <*model-with-parameters*> : contains information about the radio propagation model used and its parameters (independently for each antenna).

Following is an example of a cell list file (no empty lines at the beginning/end):

```
IJS-A;1;IJS-A-1_hata_urban_460697_99918_20_10_900_30_0_0_COS-21;30;hata;urban
IJS-B;2;IJS-B-2_hata_urban_460697_99918_20_10_900_135_0_0_COS-21;30;hata;urban
IJS-C;3;IJS-C-3_hata_urban_460697_99918_20_10_900_270_0_0_COS-21;30;hata;urban
```

2.4.2. The output data table

The optionally generated data table contains one row of data for each raster point of the output coverage raster map. Raster points with no coverage (no received radio signal from any transmitter) are not included in the map, nevertheless the table can be quite large and needs considerable time for creation. The row format of the table is shown in Table 5.

Table 5: Output data table format

Column name	**x**	**y**	**Resolution**	**cell1**	**id1**	**Pr1**	**model1**	**…**	**cellN**	**idN**	**PrN**	**modelN**	**E_c/N_0**
format	int 6	int 6	int 4	varchar 32	int 6	real 6	varchar 128	…	varchar 32	int 6	real 6	varchar 128	real 6

Description of columns:

1. **x**: x coordinate of geographic location (a map raster point), in [m] (*integer* – 4 bytes, default print length is 6).
2. **y**: y coordinate of geographic location (a map raster point), in [m] (*integer* – 4 bytes, default print length is 6).
3. **resolution**: resolution of the computed coverage map, in [m] (*integer* – 4 bytes, default print length is 4).
4. **cell*i***: cell name (*varchar*, max 32 chars (bytes)).
5. **id*i***: antenna index (*integer* – 4 bytes, default print length is 6).
6. **Pr*i***: calculated received signal power in [dBm], received at this geographic location (output map raster point) from *cell*i (*real* – 4 bytes floating point, default print length is 6). Valid values are > -999.0. The value of -999.0 has a special meaning – no signal received (replacing -∞).
7. **model*i***: Path loss model name with parameters, used for this *cell*i (*varchar*, max 128 chars (bytes)).
8. **E_c/N_0**: the received power od the strongest signal divided by the sum of the received powers of all signals, in [dB] (*real* – 4 bytes floating point, default print length is 6). Valid values are > -999.0. The value of -999.0 has a special meaning – no signal received. (This data is useful for single frequency band systems such as those using CDMA, e.g. UMTS.)

Columns 4, 5, 6 and 7 repeat *N*-times, as defined by the command line parameter *cell_num* but not exceeding the number of all cells specified in the input cell list file.

The command line parameters related to the output data table are:

- *driver* : defines the database management system used (called also simply the database). The default value *none* means that no database is used and no data table is generated (other parameters are ignored, but dummy values for the *table* and *cell_num* parameters must be specified anyway). Contrary to *r.raplat* (which does not check for actually installed and supported databases but have a fixed list of them), *r.MaxPower* uses GRASS's information about available databases (and lists them when called with flag *-help*). Value *dbf* selects GRASS' own built-in database, which GRASS uses for its own purposes. It is a relatively simple database with limited functionality and efficiency and is not really recommended for use with RaPlaT except for simple cases. Before writing to the disk it creates the whole data table in the main memory (which quickly runs out for large tables) and does not support fast writing modes (therefore it

can be very slow for large tables, see also the *dbperf* parameter). GRASS includes support for a number of external databases. When building GRASS from source, the GRASS support must be activated for each database to be used, and that database must already be installed. The two recommendable high-performance databases that are specifically supported and tested with *r.MaxPower* are MySQL (parameter value *msyql*) and PostgreSQL (parameter value *pg*). SQLite (a simple locally installed database library) is also supported (parameter value *sqlite*). Alternatively, the results can be written to a standard CSV format file (parameter value *csv*).

- *database* : defines the name of an existing database where the output data table will be created. The default value represents the location of the GRASS' built-in DBF database (which is always available, but is not recommendable for larger radio coverage projects; three environment variables represent the GRASS working environment – the basic GRASS database folder, the user-selected GRASS *location* and the user's GRASS *mapset*. The output data table is stored as a dbf-format file named *$GISDBASE/$LOCATION_NAME/$MAPSET/dbf/<table_name>.dbf*. In case of MySQL or PostgreSQL, an (empty) database must first be created with their own tools, with an arbitrary name (e.g., a reasonable database name could be *grass*).
- *table* : defines the output table name for a particular radio coverage computation.
- *cell_num* : defines the number of the strongest received signals at each point of the output coverage raster that are to be stored in the data table (see Table 5).
- *dbperf* : stands for "database performance" and selects faster modes of writing tables. For *dbperf* values between 2 and 98, the so called multiple-row inserts are used, where instead of a single data row, a group of data rows (2 to 98) is inserted with each SQL INSERT statement. For MySQL and PostgreSQL, a reasonable value is 20, which is the default. In our case it speeds up the data table creation around 2.5x for PostgreSQL and 3.8x for MySQL (relative to the basic single-row insert mode). For the GRASS' built-in DBF database, *dbperf* value should be set to 1, (the basic non-accelerated mode, because GRASS DBF does not support multiple-row inserts). The value of 99 selects a special very fast mode which is supported only for MySQL and PostgreSQL. It doesn't write data rows directly into the output data table but instead creates an intermediate CSV text file. This is at the end converted to the data table in a single step, using non-standard database-specific commands (for MySQL: *LOAD DATA LOCAL INFILE 'file.csv' INTO TABLE <table> FIELDS TERMINATED BY ',' ENCLOSED BY '"';* for PostgreSQL: *COPY <table> FROM 'file.csv' CSV QUOTE '"';*). In our case, by using this mode, speeds-up of data table creation of around 20x have been achieved.

2.5. Prepare clutter map – r.clutconvert

In addition to DEM, some radio signal propagation models (*hataDEM* in our case) need also information about the signal loss at the receiver location related to the land use (e.g. urban, agricultural, forest areas, etc.) This information is given in the form of a raster map called *clutter* map, where the value of each raster point specifies the received signal loss in [dB] at that point due to the land use. Land use is often given as a raster map with values specifying the use (e.g. 1 – irrigated agriculture, 2 – rangeland, 3 – coniferous forest, 4 – deciduous forest, 5 – mixed forest, 6 – disturbed). The *r.clutconvert* module converts such land use map to the corresponding clutter map. The conversion from land use types (integer numbers) to signal loss values in [dB] is specified by a table defined in a text file. The numbers representing land use depend on the particular land use map, and the radio signal

loss values depend on the radio frequency. Therefore, even if there is only one land use map, there will probably be a number of different conversion table files, each for a particular frequency band. The one to be used by *r.clutconvert* is specified with its *landuse_to_pathloss* command line parameter).

The conversion table file consists of a number of lines. Lines can be either comment or data lines, or empty. A comment line starts with '#' and is ignored. A data line specifies conversion from a land use value (an integer number) to the corresponding radio signal loss in [dB] (generally a floating point value), with ':' as a separator. The following is an example of this file:

```
# Terrain type loss factors - hataDEM model
#  1 - irrigate agriculture
#  2 - range land
#  3 - coniferous forest
#  4 - deciduous forest
#  5 - mixed forest
#  6 - disturbed

1:15.5
2:15
3:25
4:20
5:22
6:21
```

Fig. 10 shows the official (commercial) Slovenian land use map for the Ljubljana region (it includes 12 categories). The corresponding clutter map (terrain-related fading in [dBm]) generated by *r.clutconvert* is shown in Fig. 11.

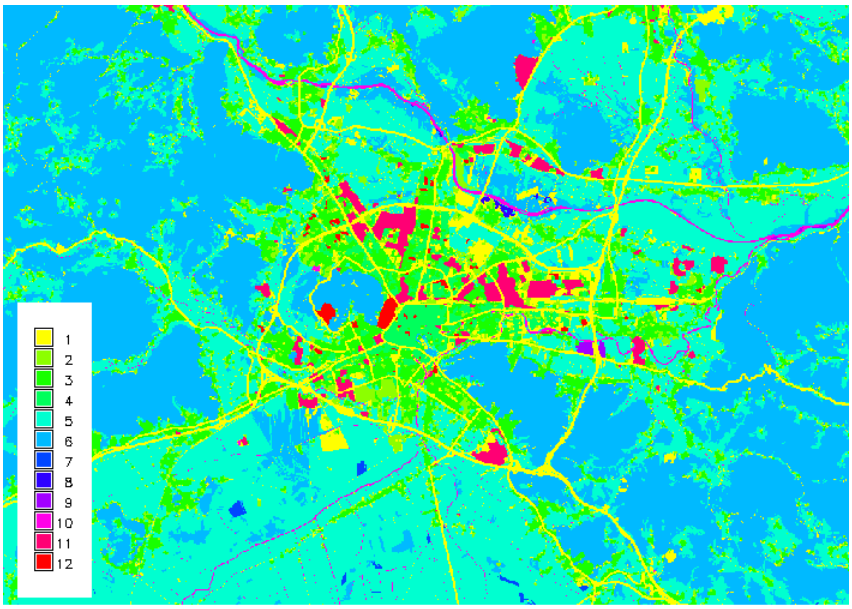


Fig. 10: Official land use map for the Ljubljana region

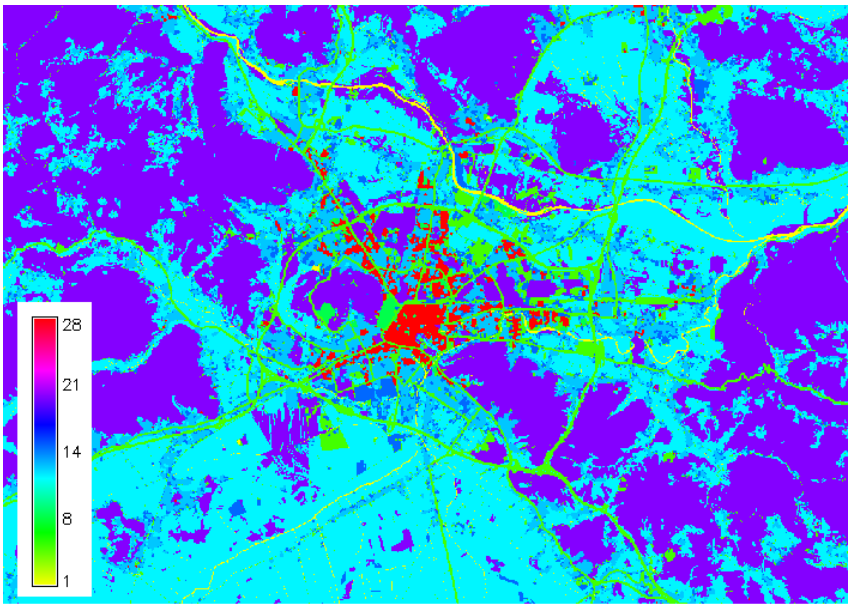


Fig. 11: The corresponding clutter map generated by *r.clutconvert*

- Description:

```
Clutter convert module (v18aug2017)
```

- Usage:

```
r.clutconvert input=name landuse_to_pathloss=name output=name
  [--overwrite] [--help] [--verbose] [--quiet] [--ui]
```

- Flags:
- Parameters:

```
                input   Input raster map - land usage categories
 landuse_to_pathloss    Input text file - mapping 'land usage' -> 'RaPlaT path loss' (e.g.
                        clutter map for hataDEM model)
               output   Output raster map - RaPlaT path loss (e.g. clutter map for hataDEM
                        model)
```

- Example (single line command):

```
r.clutconvert input=clut_category landuse_to_pathloss=/home/user1/convtable
output=clut_dBloss --o
```

3. GRASS and RaPlaT installation and usage

The GRASS and RaPlaT installation and usage described here correspond to the Kubuntu 24.04 operating system (version 24.04.3) and the binary GRASS 8.3.2 distribution from its standard Ubuntu repository.

On Kubuntu 24.04, there is a minor problems related to the installation of additional GRASS modules (like RaPlaT): the GRASS command *g.extension* needs the Linux *make* system to build (compile) additional modules such as RaPlaT modules, but the *make* system is no longer preinstalled by default on Ubuntu 24.04. Therefore it must be installed manually before using *g.extenstion*:

```
sudo apt install make
```

3.1. GRASS installation

RaPlaT requires the GRASS GIS application to be installed. The easiest way to install it is from the binary distribution located in the standard Ubuntu repository. The Ubuntu 24.04 repository contains GRASS GIS 8.3.2. There are six GRASS packages:

- *grass* – Geographic Resources Analysis Support System (GRASS GIS)
- *grass-doc* – GRASS GIS user documentation
- *grass-core* – GRASS GIS core components
- *grass-gui* – GRASS GIS graphical user interfaces
- *grass-dev* – GRASS GIS development files
- *grass-dev-doc* – GRASS GIS Programmers' Manual

Administrative (sudo) permission is required for installation. For normal work it is sufficient to install *grass* only, which automatically installs *grass-doc*, *grass-core* and *grass-gui*:

```
sudo apt update
sudo apt install grass
```

For installing additional (add-on) modules like RaPlaT (which means compiling them from their source code), the development package *grass-dev* also needs to be installed:

```
sudo apt install grass-dev
```

GRASS can be started either by typing the command “grass” in a terminal window, or with the mouse using the Kubuntu menu system. In the first case we can use a flag (“--text” or “--gui”) to explicitly select starting in the text-only mode or graphic (GUI) mode:

```
grass --text
```

or:

```
grass --gui
```

The mode selected this way is stored and used for the subsequent starts of GRASS without those flags.

When GRASS is started in GUI mode, its graphic windows interface is opened in addition to the basic text-only terminal window. This terminal windows (Fig. 12) has the GRASS environment defined and can be used to execute GRASS commands (in addition to the normal Linux commands).

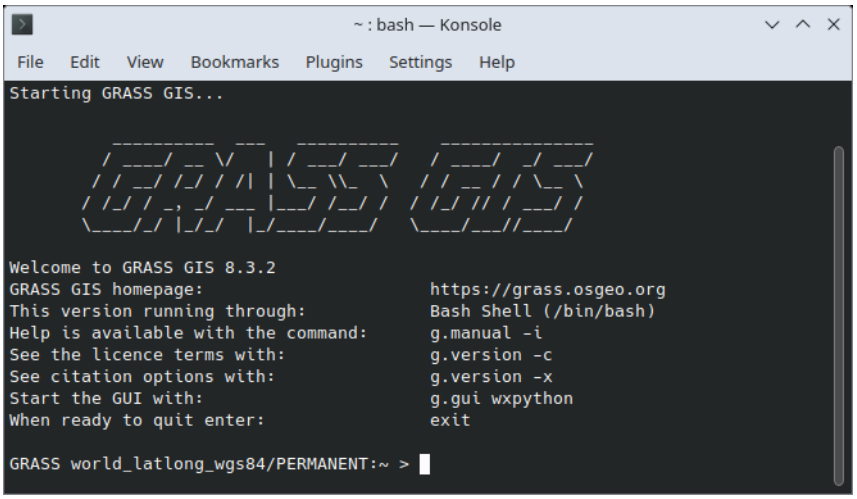
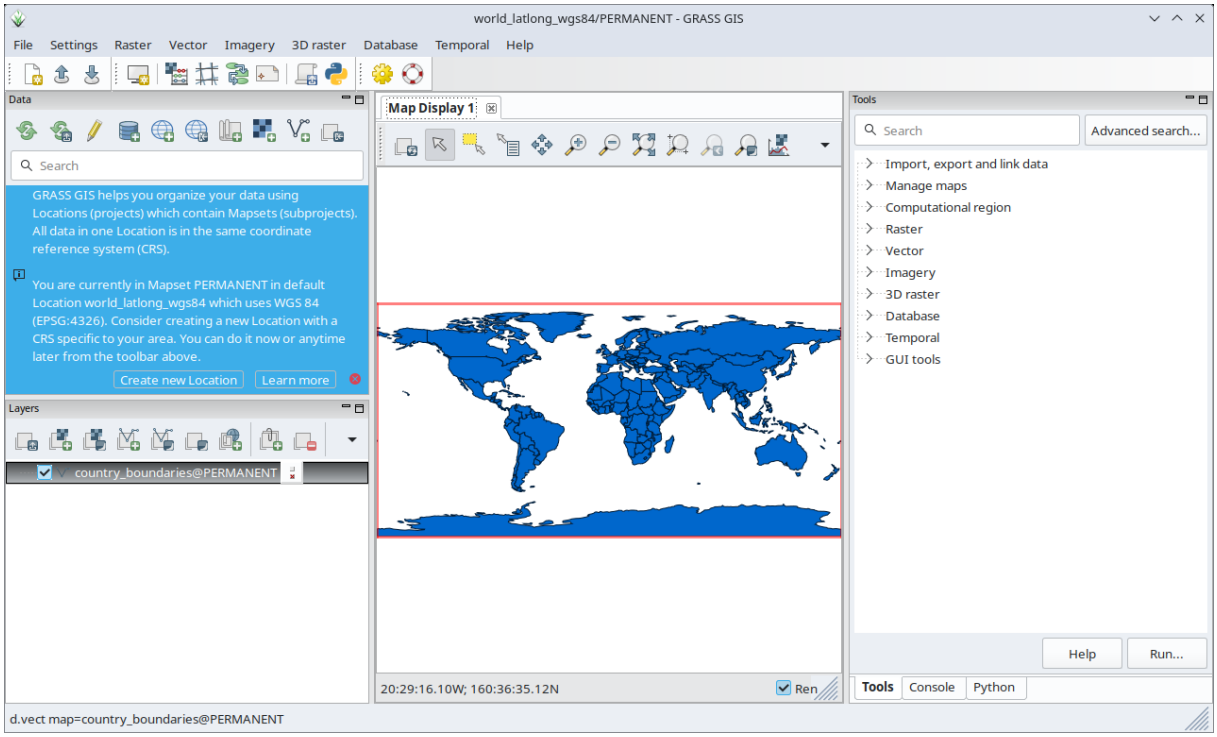
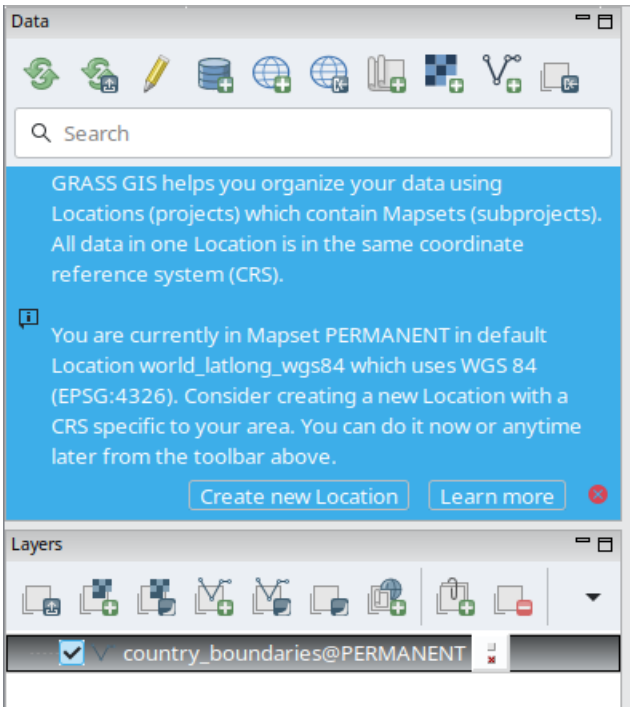


Fig. 12: GRASS terminal window (text) interface

Fig. 13 shows the GRASS GUI window after starting GRASS for the first time after installation. At that time, GRASS suggests to create a working mapset, in addition to the built-in *world_latlong_wgs84* location with *PERMANENT* mapset. This can be cancelled and done later.



(a - complete)



(b - detail)

Fig. 13: GRASS GUI window – after installation

3.2. GRASS GIS database

GRASS uses a database (*GRASS GIS database*) to store maps and other geographical data. It contains a directory tree hierarchically organized as *database → location(s) → mapset(s)*, with files representing maps and other data. Multiple GRASS GIS databases can exist, but only one is selected and used at a time.

When installed and run for the first time (without a pre-existing GRASS GIS database[4]), GRASS creates a default GIS database[5] named *grassdata* in the user's home directory[6] (i.e. *~/grassdata*) containing a location named *world_latlong_wgs84* with a mapset named *PERMANENT*. Each locations uses a particular geographical projection as defined during its creation. The predefined location *world_latlong_wgs84* (EPSG code: '4326', description; 'WSG 84') uses latitude/longitude pseudo projection with coordinates and distances in degrees. This is a global coordinate system, but is not useful for RaPlaT computations (which need distances in meters). However, it can be useful if one has DEM maps available in this coordinate system (like SRTM maps) and wants to project (convert) them to a specific target

[4] Even if GRASS has been uninstalled completely ("purged") and then reinstalled, it can detect a pre-existing GIS database (even if renamed) and use it instead of creating a new one. To start completely anew, GRASS GIS database should be deleted (or hid in another place), as well as the directory *~/.grass8*, which holds user settings including the location of the database, additionally installed modules (e.g. RaPlaT modules), etc.

[5] The template for this default database is stored as GRASS *demolocation* in */usr/lib/grass83/demolocation*, and is then copied to *~/grassdata*.

[6] If multiple users are going to use GRASS, a more proper location would be within the system space area, e.g. */var/local/grassdata*. In this case, administrative (*sudo*) privileges would be required to create the database folder and it subfolders (*locations*, *mapsets*) and to properly set their ownerships and permissions for individual users. We will not discuss the multiuser case in details here but will describe the simpler case for a single user configuration for both a newly created and a pre-existing GRASS database.

location with a suitable cartographic projection, with coordinates (and distances) in meters, which can be used for RaPlaT computations.

The idea of the special *PERMANENT* mapset is to hold permanent (non-changeable) information including DEM maps. In multiple user environment, this mapset is normally protected against (inadverte) changes by users[7]. In the single-user case described here, there is no such protection by default (but the user can protect this mapset manually by changing file/directory permissions).

The default intial GRASS GIS data base structure can be seen in Fig. 14. The permanent folder contains a single vector map containing the country boarders (as displayed in the Map Display 1 in Fig. 13 (a)).

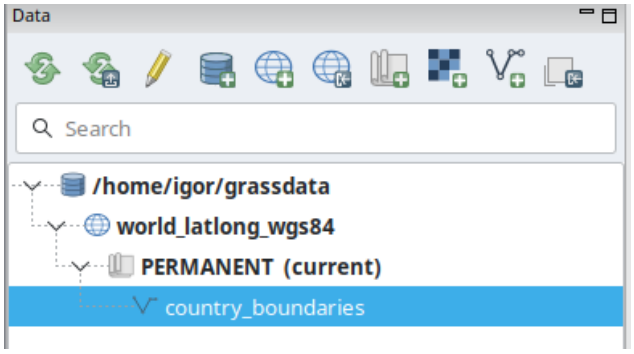


Fig. 14: Initial GRASS GIS database structure

The corresponding directory structure of this GRASS GIS database is shown in Fig. 15.

[7] This protection is based on OS (e.g. Linux) file and directory permissions, but also on GRASS' own protection - normally a user can only modify data in his/her *current* mapset.

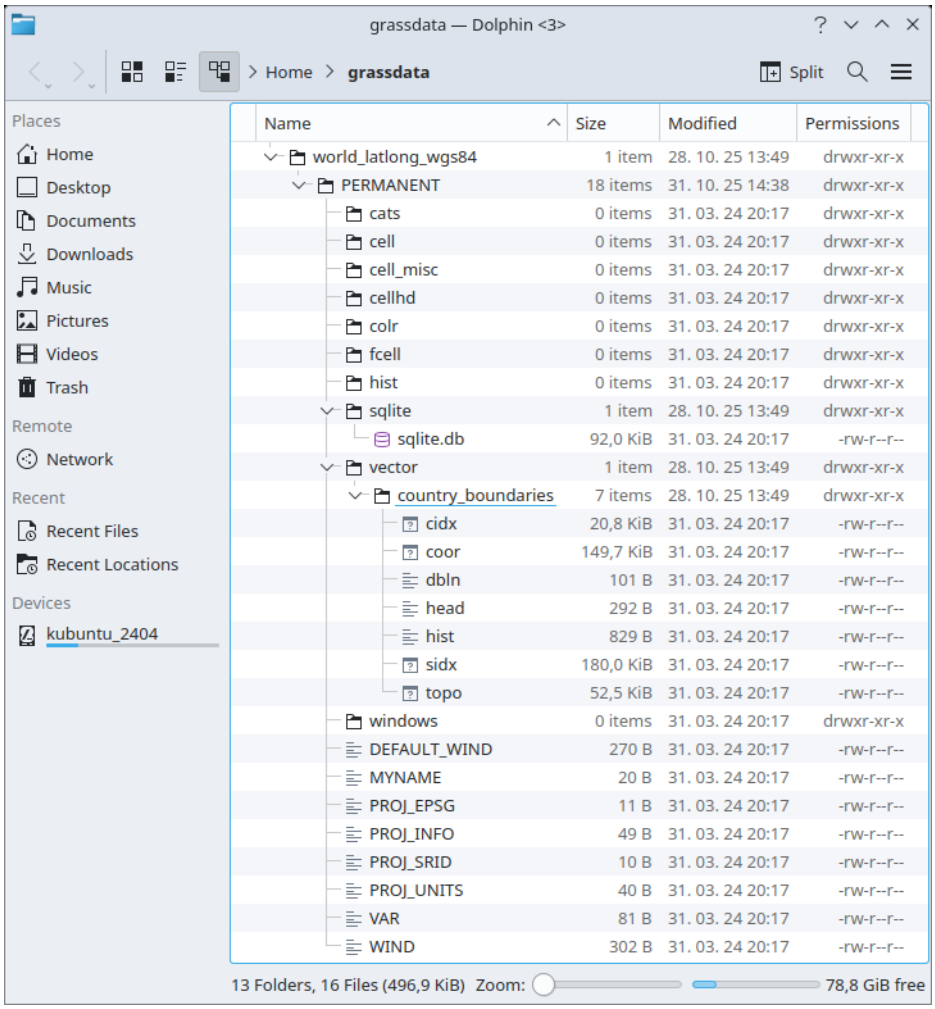


Fig. 15: Initial GRASS GIS database directory structure

3.2.1. Creating a new GRASS location for RaPlaT

We will illustrate creation of a new location suitable for processing by RaPlaT. The example shown is for the Slovenia geographic region. The user starts the process by clicking the icon *Create new location…*, Fig 16.

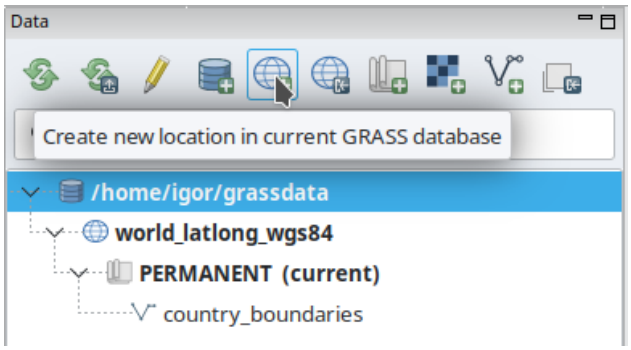


Fig. 16: Initial GRASS GIS database directory structure

In our case the name of the location will be Slovenia (Fig. 17). This will be the name of the location subfolder created in the GRASS GIS database folder.

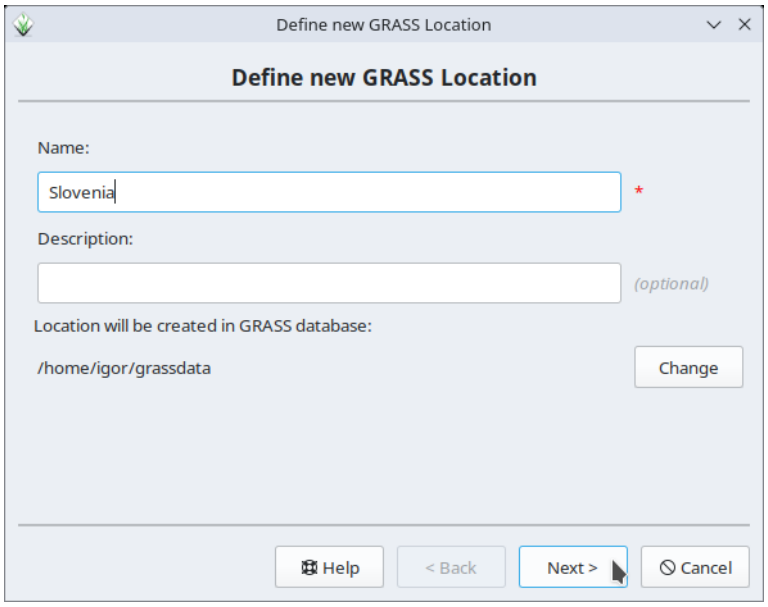


Fig. 17: Creating location named Slovenia

Now comes a more tricky part of properly defining a suitable cartographic projection for the geographic location being created (Fig. 18). This needs some knowledge about official cartographic projection(s) used in the particular country, e.g. [18] for Slovenia. Multiple choices are available to define the projection. The simplest one is probably to select an appropriate EPSG code for the particular country.

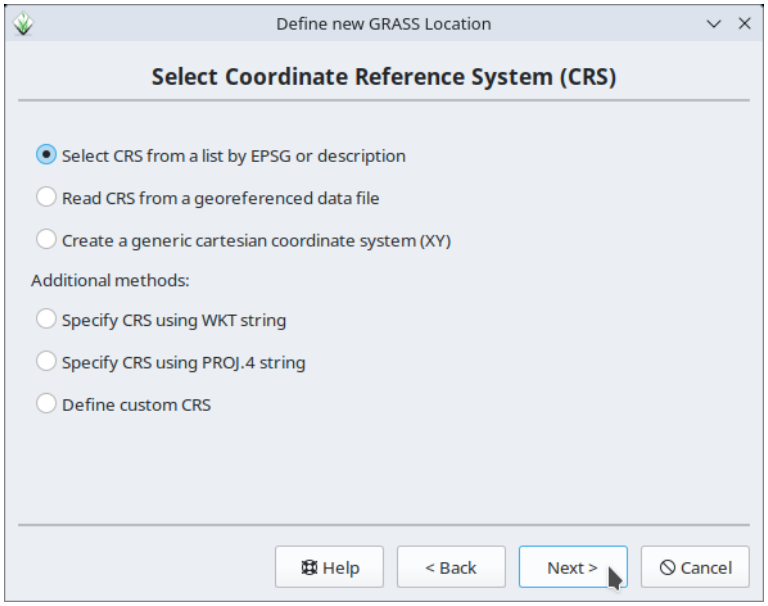
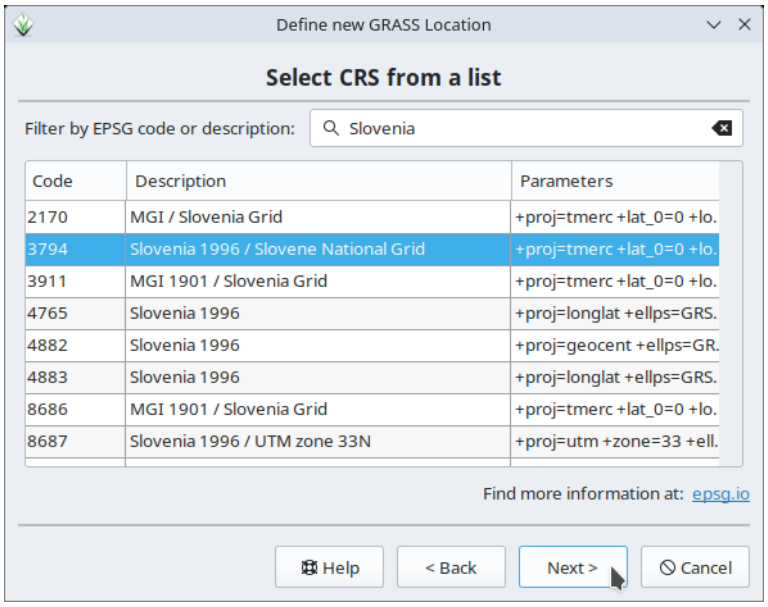


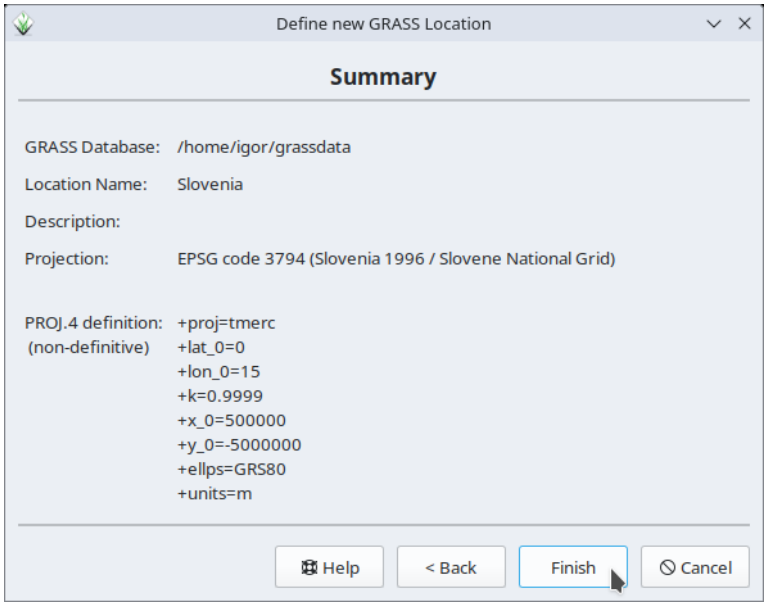
Fig. 18: Setting of the cartographic projection

GRASS contains a large list of EPSG codes. A search for Slovenia returns a short list of EPSG codes from which we select EPSG code 3794 [19,20,21] and confirm the choice (Fig. 19):

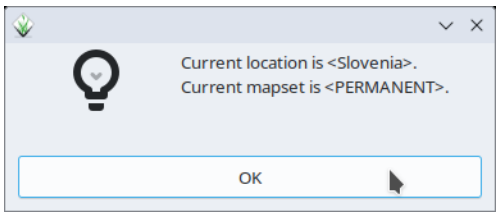
- The Slovenian new official coordinate system D96/TM, EPSG 3794 ("slovenski geodetski datum 1996, sistem dvorazsežnih kartezičnih koordinat – prečna Mercatorjeva projekcija").



(a - selecting the apropriate EPSG code for Slovenia)



(b - showing projection details)



(c - finishing of the cartographic projection setting)

Fig. 19: Selecting a suitable cartographing projection for Slovenia

The process of creating a new location also creates a special GRASS mapset named *PERMANENT* (Fig. 20). Its purpose is to contain data (maps) with read-only status accessible for common use by all GRASS users. In the case of RaPlaT it would contain a set of geographic maps used for RaPlaT computations and for position visualization (topographic, DEM and clutter maps for one or more geographic areas, possibly with different resolutions and extents).

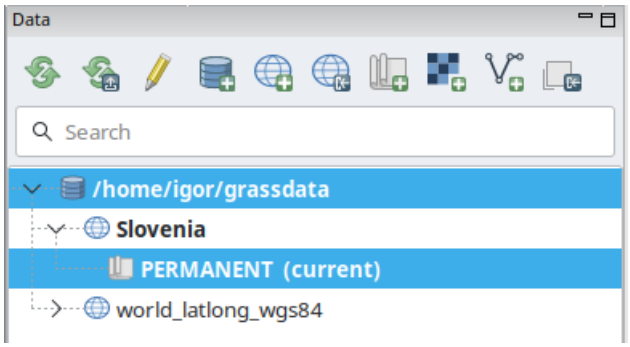
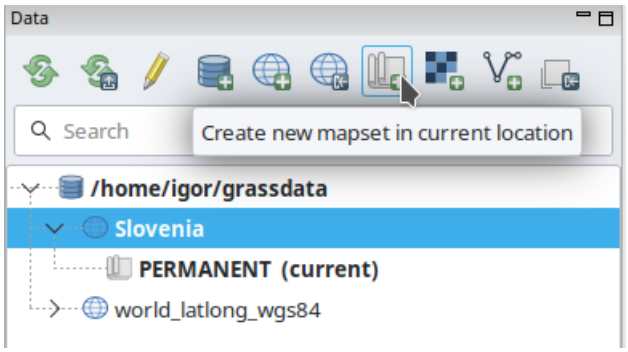
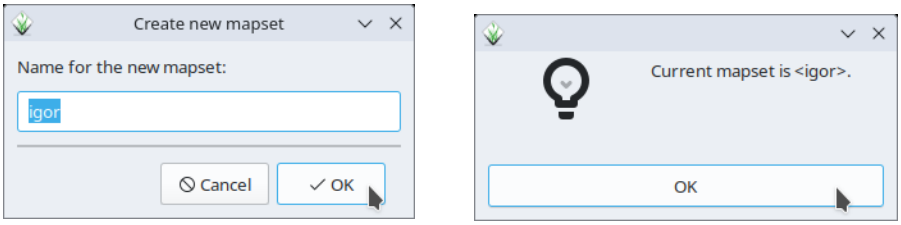


Fig. 20: Automatic creation of the *PERMANENT* mapset

Additional GRASS mapsets (user mapsets) can be created[8] by clicking the icon *Create new mapset…* (Fig 21). The suggested (default) name is the username.



(a - start creating a mapset)



(b - mapset name and confirmation)

Fig. 21: Creation of an additional (user) mapset

[8] In case of a multiuser system, individual mapsets can be assigned to individual users, and access to them protected by setting the Linux file/directory ownerships and permissions properly. The *PERMANENT* mapset can also be protected this way, limiting access to the read-only for standard GRASS users.

The newly created mapset is now set as the *current* (working) mapset (Fig. 22). The new location does not contains any maps yet (neither in the *PERMANENT* mapset nor in the new additional mapset created manually).

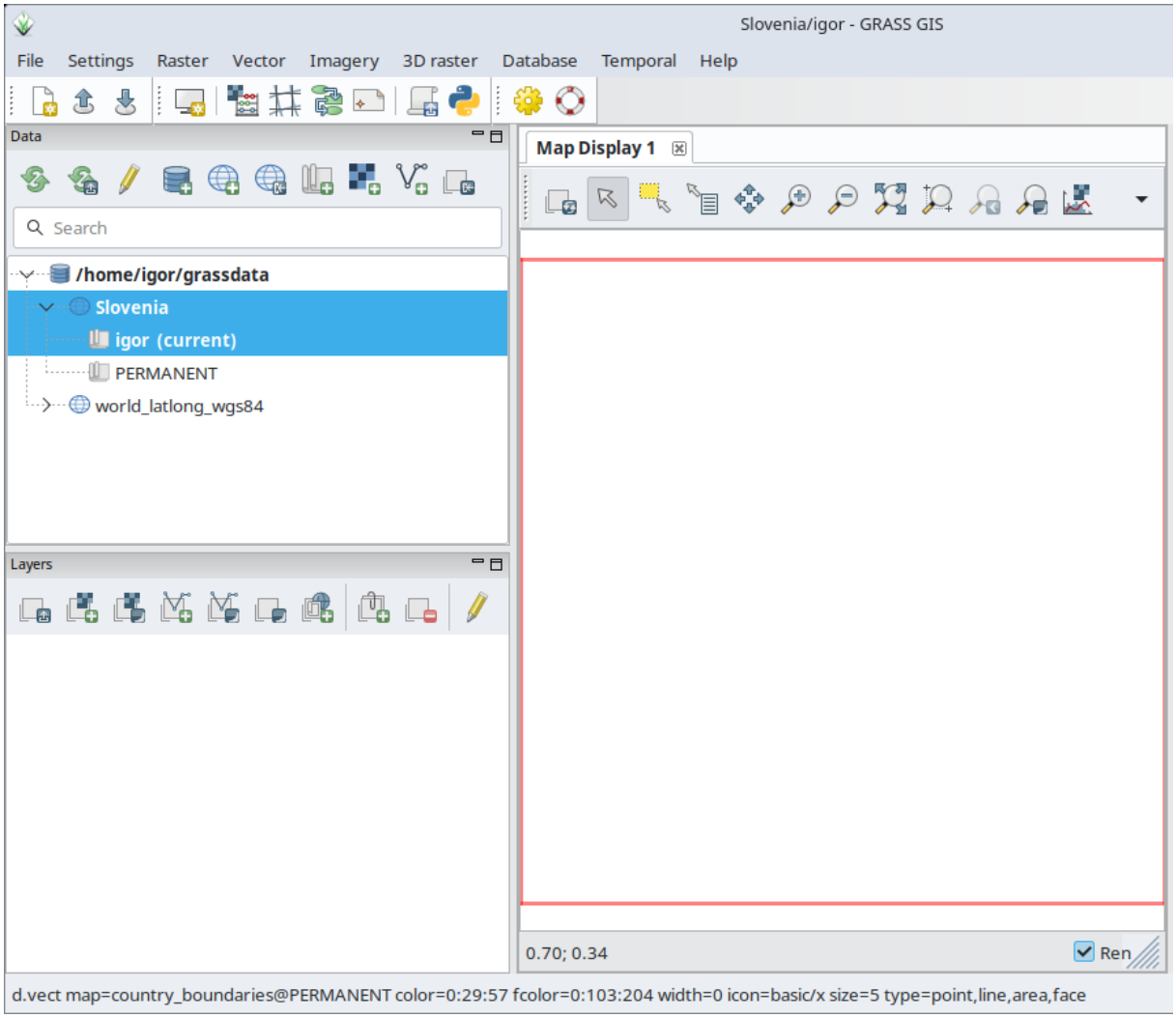


Fig. 22: GRASS session with the new *current* (working) mapset *igor*

GRASS itself grants write access only to the *current* mapset, protecting other mapsets from inadvertent changes.

3.2.2. Using a pre-existing GRASS GIS database – demo database

We can use a pre-existing GRASS GIS database, possibly from a different GRASS installation, which already includes the maps we need. (Keep in mind, however, that GIS databases from different GRASS versions might not be compatible).

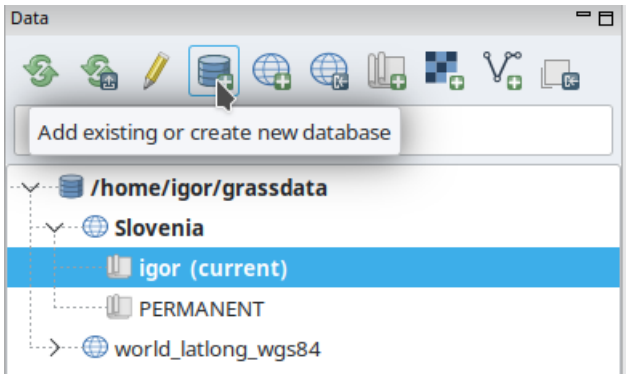
For demonstration purposes, a GRASS GIS database for RaPlaT has been created containing the minimal required maps for demo computations. It contains two locations:

- *demo_LatLong*:

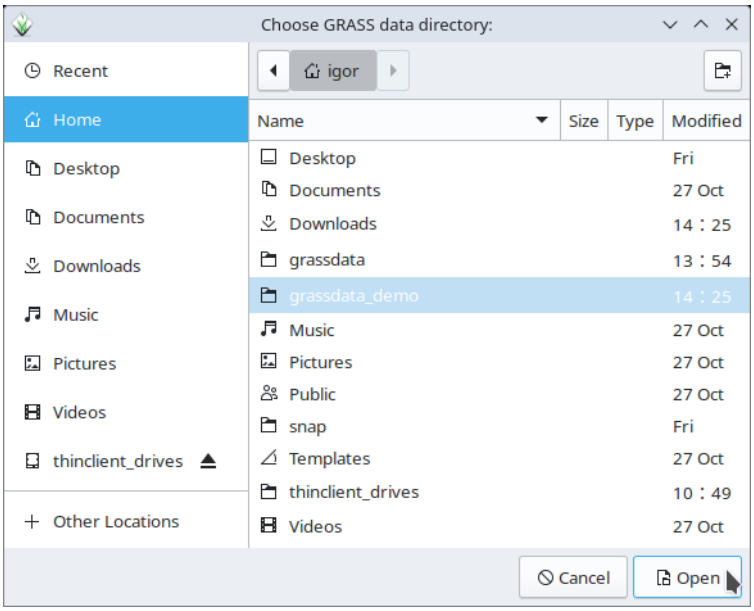
  This (empty) location can be used for processing maps given in this coordinate system. We can import such maps into this location and then project them to a final cartographic projection suitable for RaPlaT computations (e.g. *demo_Slovenia* in our case).

- *demo_Slovenia*:
  This location contains a working mapset named *user1* in addition to the PERMANENT mapset. Note that the mapset name (*user1*) is independent of the actual username (it is the name of the subdirectory) and can be changed (renamed) at will.

To use the demo GRASS GIS database, the user first extracts the folder *grassdata_demo* from *RaPlaT_demo_apr2023.zip* to a suitable directory e.g. */home/<user>* (i.e. */home/<user>/ grassdata_demo*). He/she then start adding it to GRASS by clicking the icon *Add existing or create…* (Fig 23).



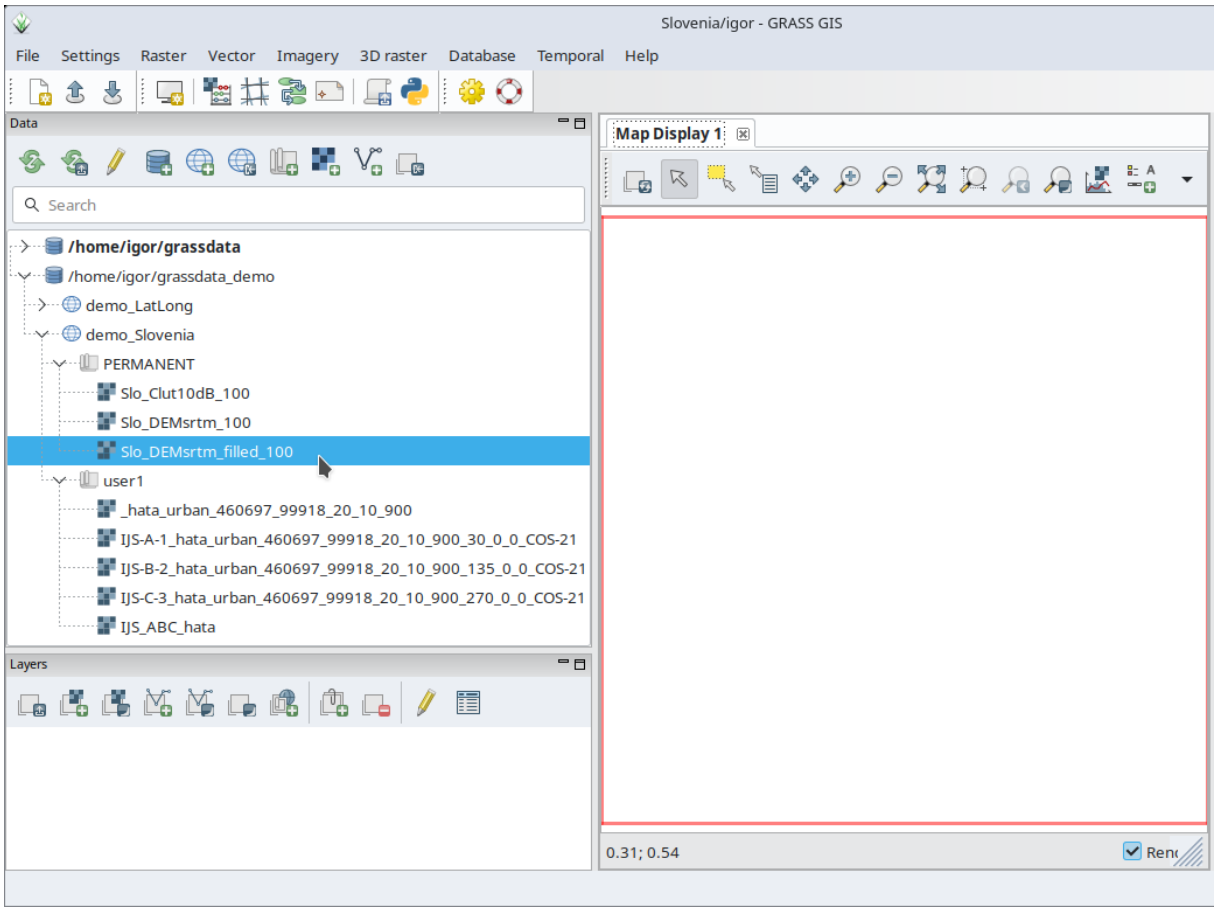
(a - clicking the icon)



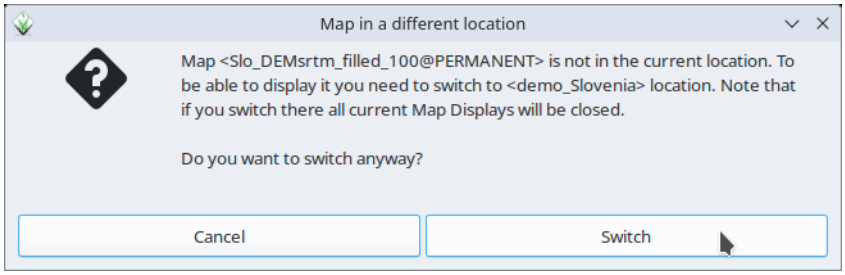
(b - selecting the extracted directory *grassdata_demo*)

Fig. 23: Adding demo GRASS GIS database (*grassdata_demo*)

The demo GRASS GIS database has been added now, but it is not selected as the active (working) database yet. GRASS can do this for the user automatically when he/she tries to display a map from the new database, e.g. by double-clicking on the map (Fig. 24).



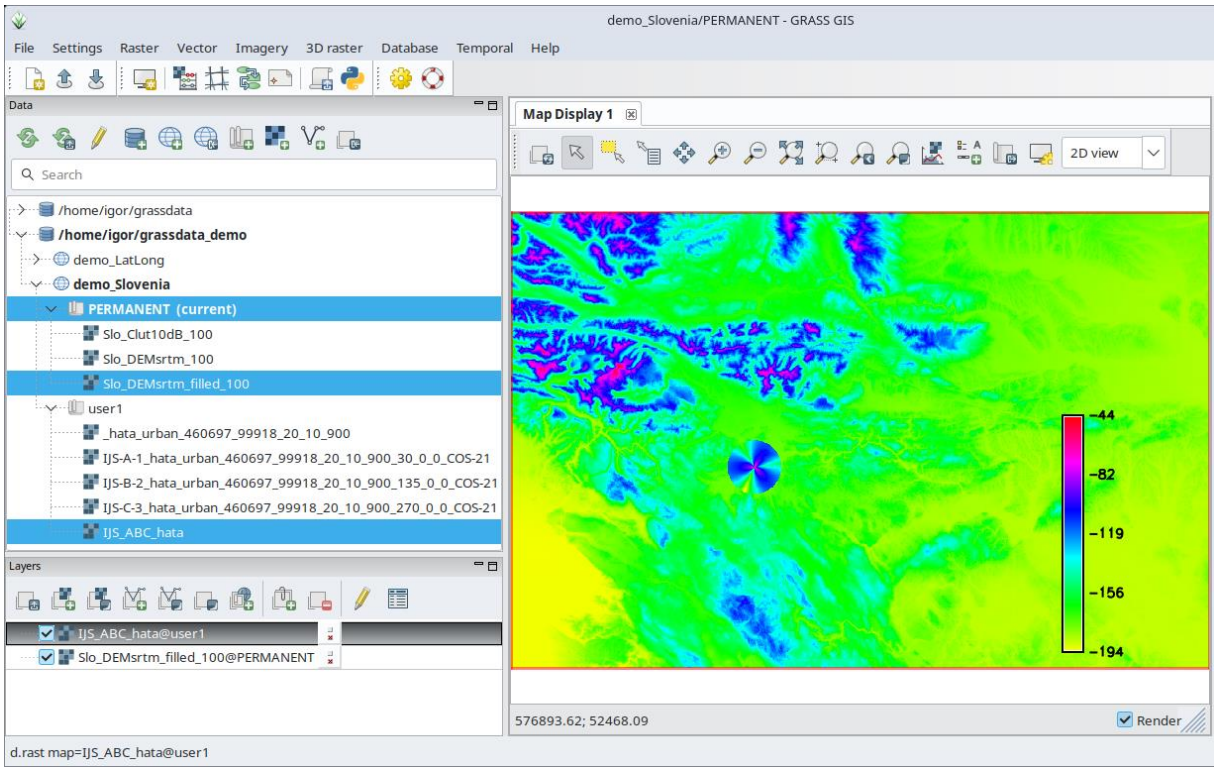
(a - a demo map selected)



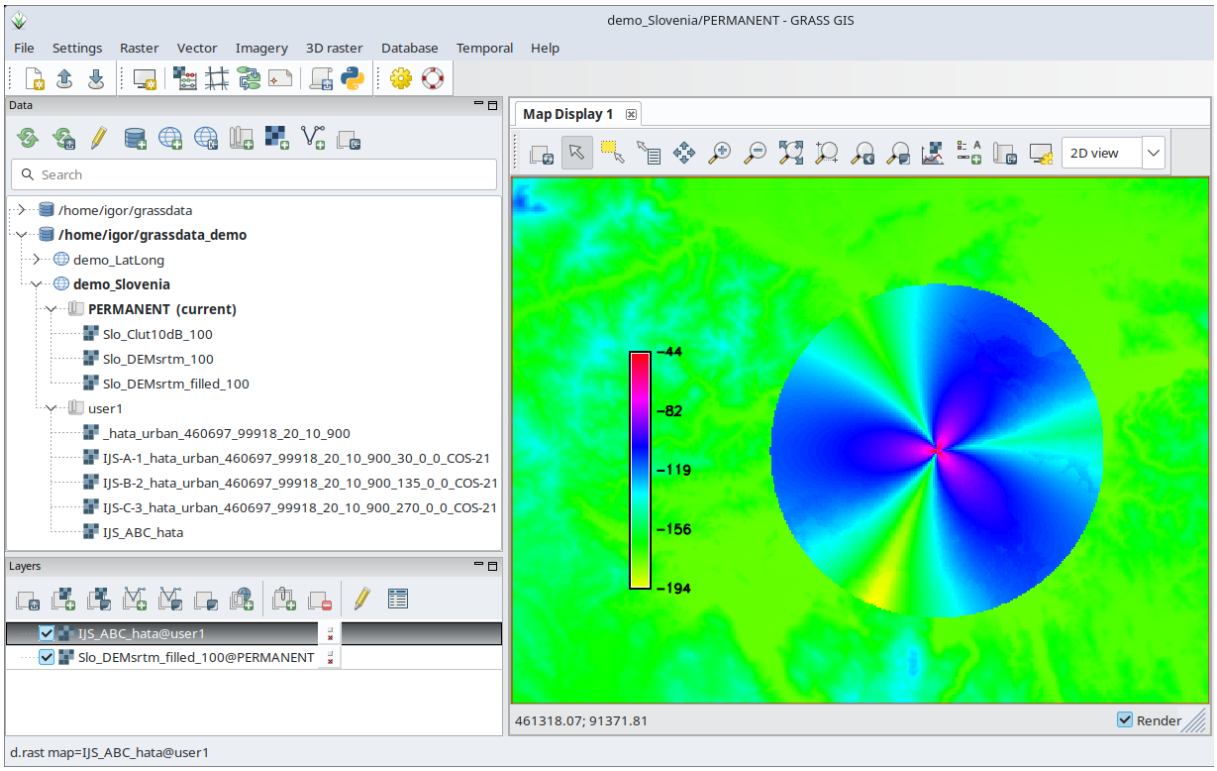
(b - switching the database after double clicking the map)

Fig. 24: Automatically switching to the new GIS database (*grassdata_demo*)

After that, the map is displayed and we can overly it by additional maps from the database. In Fig. 25, a computed radio coverage map (max. signal strength from any transmitter antenna in [dBm]) is displayed, with the DEM map in the background, and with the added legend for the radio signal strength.



(a - full map, with legend added)



(b - zoomed-in map)

Fig. 25: Radio signal coverage map (signal strength in dBm) with DEM in the background

The *PERMANENT* mapset in *demo_Slovenia* contains the following maps:

- *Slo_DEMstrm_100*: the DEM map of Slovenia obtained from the corresponding NASA SRTM Latitude/Longitude maps by projecting to a suitable Slovenian coordinate system (not really D96/TM but an older one, however this is not important for this demo). Resolution (pixel size) is 100 m. These map contain some small undefined regions.
- *Slo_DEMstrm_100_filled*: the above map with undefined areas filled in with interpolated values.
- *Slo_Clut10dB_100*: a dummy *clutter* map with fixed 10 dB fading. (The *clutter* map is required for the *hataDEM* radio propagation model.)

The *user1* mapset contains the following maps, resulting from the execution of the *r.raplat* RaPlaT module, as described later in chapter 3.4:

- *_hata_urban_460697_99918_20_10_900*: an intermediate map with omnidirectional fading computed for the *hata* propagation model at the given base station (transmitter) location. It was computed using the RaPlaT *r.hata* module (called automatically by the main *r.raplat* module).
- *IJS-A-**, *IJS-B-**, *IJS-C-**: three intermediate maps with computed signal fading for the signal transmitted from the three antennas at the given location. They were calculated from the above intermediate map using the RaPlaT *r.sector* module for each antenna (called automatically by the main *r.raplat* module).
- *IJS_ABC_hata*: the final result map, with the received strength in [dBm] of the strongest radio signal from any of the above three antennas at each point of the map. It was calculated from the above three intermediate maps using the RaPlaT *r.MaxPower* module (called automatically by the main *r.raplat* module).

3.2.3. GRASS maps for RaPlaT

A newly generated GRASS location (chapter 3.2.1) is empty, i.e. it has no maps. Three types of maps are important for RaPlaT:

- Topographic maps, showing roads, building, etc. which are not needed for radio signal computation but can be used as background maps, providing visual information about the location.
- DEM (*Digital Elevation Maps*), which are required by all RaPlaT radio signal propagation models and by the *r.sector* modul.
- *Clutter* maps, which provide information about the terrain type (usage) related radio signal fading. Only the *hataDEM* radio signal propagation model needs this information.

GRASS has functions to import maps written in various formats. Maps might be available from local institutions depending on a particular country (e.g. [22] for Slovenia).

There are some world-wide map projects providing free maps. Strictly speaking, the DEM maps are the only absolutely necessary maps for RaPlaT usage. One such source of publicly available DEM (*elevation*) maps are SRTM (*Shuttle Radar Topography Mission*) maps [6,7].

A well-known world-wide project providing free topographic maps is OpenStreetMap [23], Fig. 26.

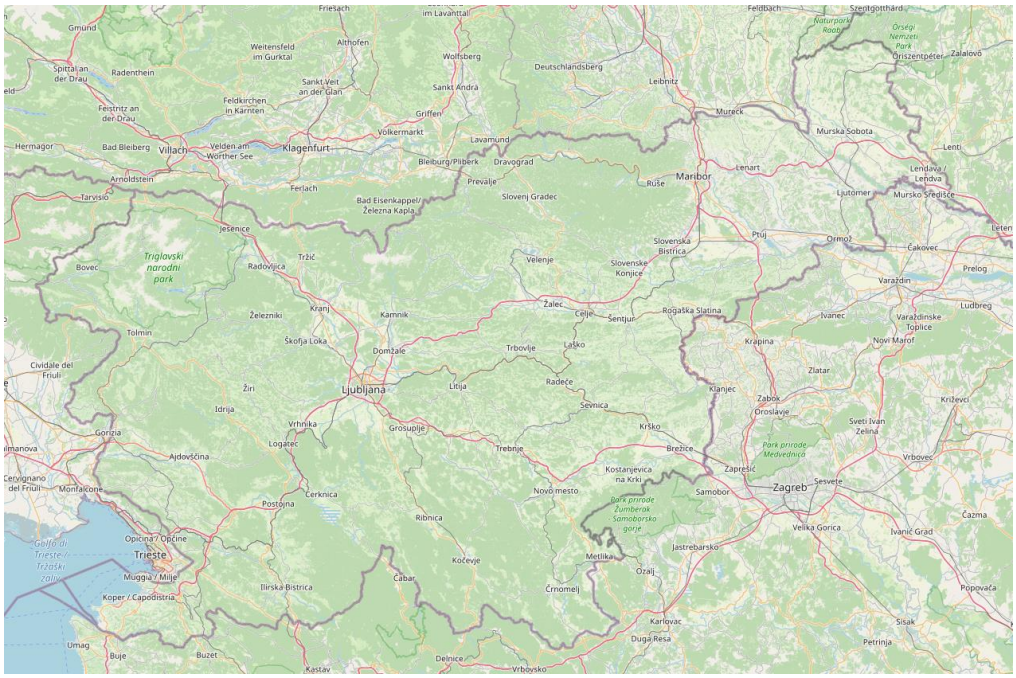


Fig. 26: OpenStreetMap – Slovenia
(https://www.openstreetmap.org/#map=9/46.1475/14.5486)

Clutter maps (land type / usage maps with areas classified as different kind of vegetation, water or urban areas) are more specific and are produced from satellite or airborne observations, e.g. by the Copernicus Programme [24,25].

Finding and importing maps for each particular location is beyond the scope of this manual and the RapLaT project, and left to the user.

3.3. RaPlaT installation

Provided that the development GRASS package has been installed as described earlier, individual RaPlaT modules can be installed easily using the GRASS command *g.extension*. For this to work, the Linux *make* system must be installed. On Ubuntu 24.04 it is not installed by default, so we must install it:

```
sudo apt install make
```

Suppose that the user has extracted the contents (i.e. the folder *RaPlaT_modules_public_nov2025*) form the distribution file *RaPlaT_modules_public_nov2025.zip to* his/her *Downloads* subdirectory (resulting in the directory */home/<user>/Downloads/RaPlaT_modules_public_nov2025*; the files and folders should have write permissions for the user, otherwise the *g.extension* command may fail.) All modules can then be installed by the following set of commands (executed from within the GRASS environment, i.e. in the terminal where GRASS has been started; the examples below are for the user *igor* and must be changed accordingly for a different user):

```
g.extension extension=r.raplat operation=add url=/home/igor/Downloads/RaPlaT_modules_public_nov2025/python3/r.raplat

g.extension extension=r.clutconvert operation=add url=/home/igor/Downloads/RaPlaT_modules_public_nov2025/raster/r.clutconvert

g.extension extension=r.cost231 operation=add url=/home/igor/Downloads/RaPlaT_modules_public_nov2025/raster/r.cost231
g.extension extension=r.fspl operation=add url=/home/igor/Downloads/RaPlaT_modules_public_nov2025/raster/r.fspl
g.extension extension=r.hata operation=add url=/home/igor/Downloads/RaPlaT_modules_public_nov2025/raster/r.hata
g.extension extension=r.hataDEM operation=add url=/home/igor/Downloads/RaPlaT_modules_public_nov2025/raster/r.hataDEM
g.extension extension=r.waik operation=add url=/home/igor/Downloads/RaPlaT_modules_public_nov2025/raster/r.waik
```

```
g.extension extension=r.sector operation=add url=/home/igor/Downloads/RaPlaT_modules_public_nov2025/raster/r.sector
g.extension extension=r.MaxPower operation=add url=/home/igor/Downloads/RaPlaT_modules_public_nov2025/raster/r.MaxPower
```

The modules are installed in the user's *.grass* directory, which is */home/igor/.grass7/addons/bin/* in the example above.

The modules can be uninstalled by the following commands:

```
g.extension -f extension=r.raplat operation=remove

g.extension -f extension=r.clutconvert operation=remove

g.extension -f extension=r.cost231 operation=remove
g.extension -f extension=r.fspl operation=remove
g.extension -f extension=r.hata operation=remove
g.extension -f extension=r.hataDEM operation=remove
g.extension -f extension=r.waik operation=remove

g.extension -f extension=r.sector operation=remove
g.extension -f extension=r.MaxPower operation=remove
```

3.3.1. Antennas

RaPlaT needs technical data about the antennas used by base stations (transmitters). This includes radiation patterns and some other parameters stored in text files in the MSI format. The *r.raplat* module uses a special CSV-format file containing the list of all antennas, with some additional data and the MSI file name for each antenna. The default path to this file is *$GISBASE/etc/radio_coverage/antenna_diagrams/antennamap.csv* (*GISBASE* is the top-level directory of the GRASS installation), but another path can be specified with the parameter *antmap_file* (see chapter 2.1 for details). The individual antennas' MSI files are stored in subdirectories of this file's location. The subdirectories can be named arbitrary, which provides a possibility to group the antennas suitably.

The RaPlaT demo distribution folder *RaPlaT_demo_apr2023* contains *antenna_diagrams* subfolder with only one artificial antenna named COS21 with a cosine radiation pattern, and the corresponding CSV file *antennamap.csv*. Putting the *antenna_diagrams* folder (complete with its contents) to the default location would require root (sudo) rights. Instead, the user can copy it inside his/her home directory, e.g. */home/<user>/raplat*.

3.4. Using RaPlaT – running demos

We assume that the user has done the following:

- Added the demo GRASS GIS database *grassdata_demo* distributed in *RaPlaT_demo_apr2023.zip*, as described in chapter 3.2.2.
- Installed the RaPlaT modules distributed in *RaPlaT_modules_public_nov2025.zip*, as described in chapter 3.3.
- Created directory *raplat* in his/her home directory (i.e. */home/<user>/raplat*) and copied there:
    a. The directory *antenna_diagrams* distributed in *RaPlaT_demo_apr2023.zip* (resulting in the directory */home/<user>/raplat/antenna_diagrams*).
    b. The files from the folder *rundemos* distributed in *RaPlaT_demo_apr2023.zip*. (This included the file *cmds.txt*.)

After that, the *raplat* directory should have the following structure (Fig. 27):

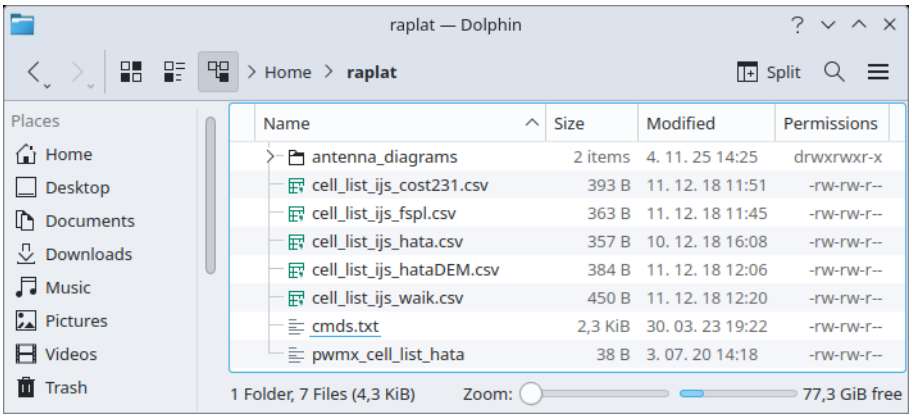


Fig. 27: Contens of the directory *raplat*

The user can then run computations using demo commands listed in the file *cmds.txt*. Computations are normally performed by calling the *r.raplat* module with a suitable input CSV-format file containing the list of base stations (transmitters) and propagation model parameters. A set of demo CSV files for all supported propagation models is included in the *rundemos* directory. Individual RaPlaT modules (propagation models, *r.sector* and *r.MaxPower*) can also be run manually and the corresponding demo commands are listed in *cmds.txt*.

To run *r.MaxPower*, a special input file is needed with a simplified list of base stations and some of their parameters. This file is normally prepared automatically as a temporary file during *r.raplat* execution. To make it possible to manually run *r.MaxPower* for demo or test purposes, this file is also included for the case of the *hata* propagation model computation (file *pwmx_cell_list_hata*).

In our example below, the user will first create a new empty mapset *user2* (see Fig. 21 in chapter 3.2.1) so as to not overwrite the existing maps in the demo *user1* mapset (Fig. 28).

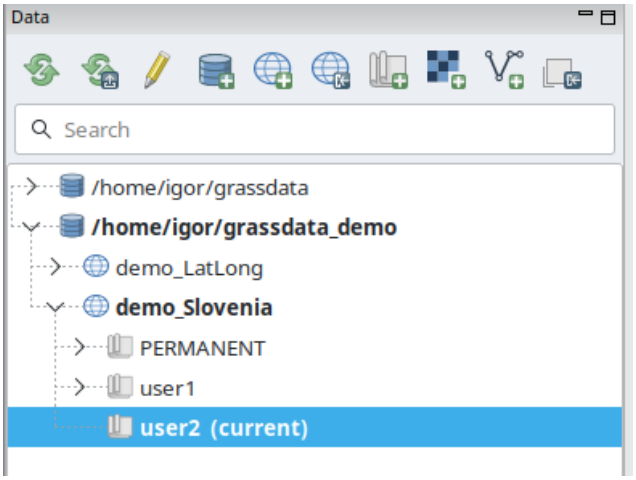


Fig. 28: New mapset *user1*

Next, the user will execute the following *r.raplat* command from *cmds.txt* (Fig. 29).

```
r.raplat csv_file=~/raplat/cell_list_ijs_hata.csv dem_map=Slo_DEMsrtm_filled_100@PERMANENT
antmap_file=~/raplat/antenna_diagrams/antennamap.csv out_map=IJS_ABC_hata --o
```

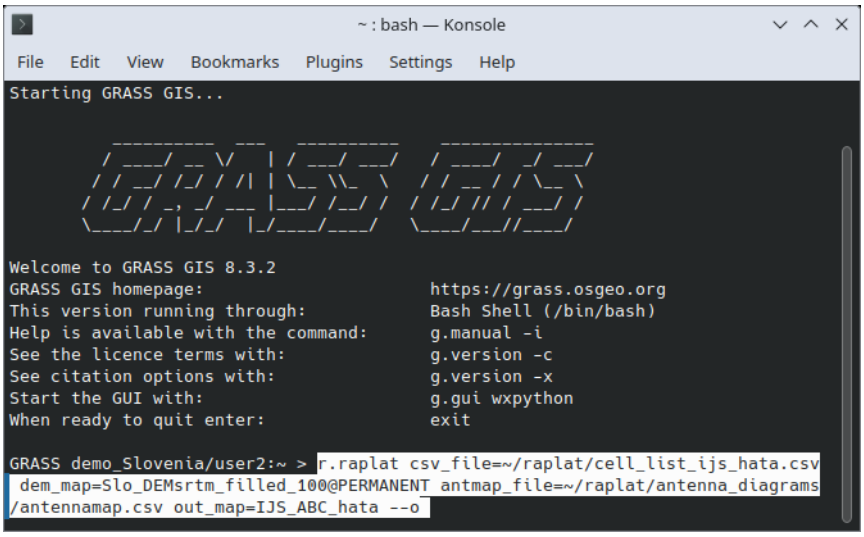


Fig. 29: Executing a demo RaPlaT computation

This command will perform the computation of RSS (*Received Signal Strength*) in [dBm] (more precisely: max signal strength form any transmit antenna) for a system with three antennas at a single geographical location, using the *hata* propagation model, as described in *cell_list_ijs_hata.csv*. When finished, the *user2* mapset should contain computed signal coverage maps identical to those already existing in the demo *user1* mapset.

During the *r.raplat* execution, the details of the individual RaPlaT modules' execution are displayed in the terminal window:

```
GRASS demo_Slovenia/user2:~ > r.raplat csv_file=~/raplat/cell_list_ijs_hata.csv
dem_map=Slo_DEMsrtm_filled_100@PERMANENT antmap_file=~/raplat/antenna_diagrams/antennamap.csv
out_map=IJS_ABC_hata --o

Number of detected processors = 4

READING AND CHECKING THE ANTENNAS-MAPPING CSV TABLE ...
Number of .MSI files found: 1

READING AND CHECKING THE RADIO SECTOR CSV TABLE ...
Standard CSV file format detected

GETTING THE LIST OF EXISTING MODEL AND SECTOR FILES
FROM PREVIOUS SIMULATION RUNS IN THE CURRENT MAPSET...

DELETING UNNEEDED MODEL AND SECTOR FILES FROM
PREVIOUS SIMULATION RUNS IN THE CURRENT MAPSET...

STARTING RADIO COVERAGE COMPUTATION...

----- PROCESSING MODELS -----
Temporarily changing the existing current region:
N = 195000.0
W E = 370000.0 630000.0
S = 25000.0
res E-W/N-S = 100.0/100.0
to the extended computation region:
N = 194950.0
W E = 370050.0 629950.0
S = 25050.0
res E-W/N-S = 100.0/100.0
> IJS-A-1 (1./1)
r.hata input_dem=Slo_DEMsrtm_filled_100@PERMANENT
output=_hata_urban_460697_99918_20_10_900 area_type=urban
coordinate=460697,99918 ant_height=20 radius=10 rx_ant_height=1.5
frequency=900 --overwrite
< (1./_)
Color table for raster map <_hata_urban_460697_99918_20_10_900> set to
```

```
'rainbow'

----- PROCESSING SECTORS -----
> IJS-A-1 (1./3)
r.sector pathloss_raster=_hata_urban_460697_99918_20_10_900@user2
input_dem=Slo_DEMsrtm_filled_100@PERMANENT
output=IJS-A-1_hata_urban_460697_99918_20_10_900_30_0_0_COS-21 east=460697
north=99918 radius=10
ant_data_file=/home/igor/raplat/antenna_diagrams/_demo_/COS_21.MSI
beam_direction=30 mech_tilt=0 height_agl=20 rx_ant_height=1.5 --overwrite
> IJS-B-2 (2./3)
r.sector pathloss_raster=_hata_urban_460697_99918_20_10_900@user2
input_dem=Slo_DEMsrtm_filled_100@PERMANENT
output=IJS-B-2_hata_urban_460697_99918_20_10_900_135_0_0_COS-21 east=460697
north=99918 radius=10
ant_data_file=/home/igor/raplat/antenna_diagrams/_demo_/COS_21.MSI
beam_direction=135 mech_tilt=0 height_agl=20 rx_ant_height=1.5 --overwrite
> IJS-C-3 (3./3)
r.sector pathloss_raster=_hata_urban_460697_99918_20_10_900@user2
input_dem=Slo_DEMsrtm_filled_100@PERMANENT
output=IJS-C-3_hata_urban_460697_99918_20_10_900_270_0_0_COS-21 east=460697
north=99918 radius=10
ant_data_file=/home/igor/raplat/antenna_diagrams/_demo_/COS_21.MSI
beam_direction=270 mech_tilt=0 height_agl=20 rx_ant_height=1.5 --overwrite
< (1./_)
< (3./_)
< (2./_)
Color table for raster map
<IJS-A-1_hata_urban_460697_99918_20_10_900_30_0_0_COS-21> set to 'rainbow'
Color table for raster map
<IJS-B-2_hata_urban_460697_99918_20_10_900_135_0_0_COS-21> set to 'rainbow'
Color table for raster map
<IJS-C-3_hata_urban_460697_99918_20_10_900_270_0_0_COS-21> set to 'rainbow'

----- GENERATING FINAL RESULTS - RASTER MAP AND DB (OPTIONALLY) -----
Temporarily changing current region to the computation region:
N = 194950.0
W E = 370050.0 629950.0
S = 25050.0
res E-W/N-S = 100.0/100.0
> WRITE RASTER MAP ONLY
r.MaxPower 'cell_input=/tmp/grass8-igor-1357324/tmp_t3n2r4f'
output=IJS_ABC_hata generate=rss-max bandwidth=5 table=out_db driver=none
'database=$GISDBASE/$LOCATION_NAME/$MAPSET/dbf' dbperf=1 cell_num=5
--overwrite
Processing 3 cells...
Sorting receive power values
 100%
Finished sorting receive power values
Color table for raster map <IJS_ABC_hata> set to 'rainbow'
Processing of 1 model(s) took 0.32[s], i.e. 0.32[s/model]
Processing of 3 sector(s) took 0.42[s], i.e. 0.14[s/sector]
Writing of output raster map took 0.56[s]
Processing finished
GRASS demo_Slovenia/user2:~ >
```

After the computation has finished, the resulting map(s) can be displayed in GRASS. They should be identical to those in the *user1* demo mapset, shown previously in Fig. 25 in chapter 3.2.2.

4. References

[1] GRASS GIS, Wikipedia, http://en.wikipedia.org/wiki/GRASS_GIS

[2] GRASS GIS, home page, http://grass.osgeo.org/

[3] GRASS-RaPlaT, The Radio Planning Tool for GRASS GIS system, Home page, http://www-e6.ijs.si/RaPlaT/GRASS-RaPlaT_main_page.htm

[4] Andrej Hrovat, Igor Ozimek, Andrej Vilhar, Tine Celcer, Iztok Saje, Tomaž Javornik, Radio coverage calculations of terrestrial wireless networks using an open-source GRASS system. WSEAS Transactions on Communications, 2010, vol. 9, no. 10, pp. 646-657.

[5] Igor Ozimek, Andrej Hrovat, Andrej Vilhar, Tine Celcer, Iztok Saje, Tomaž Javornik, GRASS-RaPlaT - an open-source tool for radio coverage calculations, V: Joint Workshop on Wireless Communications, 1-2 March 2011, Paris, France, JNCW 2011. [S. l.]: IEEE, France section, 2011, 6 pp.

[6] Shuttle Radar Topography Mission, Wikipedia, http://en.wikipedia.org/wiki/Shuttle_Radar_Topography_Mission

[7] (a) Shuttle Radar Topography Mission, NASA - Jet Propulsion Laboratory, http://www2.jpl.nasa.gov/srtm/ (currently not available),
(b) USGS EROS Archive - Digital Elevation - Shuttle Radar Topography Mission (SRTM), https://www.usgs.gov/centers/eros/science/usgs-eros-archive-digital-elevation-shuttle-radar-topography-mission-srtm-1,
(c) USGS EarthExplorer, https://earthexplorer.usgs.gov/,
(d) OpenTopography, https://opentopography.org/start

[8] Comma-separated values, Wikipedia, http://en.wikipedia.org/wiki/Comma-separated_values

[9] Common Format and MIME Type for Comma-Separated Values (CSV) Files, RFC 4180, October 2005, http://www.rfc-editor.org/rfc/rfc4180.txt

[10] MSI Planet Antenna File Format, http://radiomobile.pe1mew.nl/?The_program___Definitions___MSI

[11] GRASS-Wiki, Category:Parallelization, http://grasswiki.osgeo.org/wiki/Category:Parallelization

[12] GRASS-Wiki, Parallel GRASS jobs, http://grasswiki.osgeo.org/wiki/Parallel_GRASS_jobs

[13] S. R. Saunders, A. Aragón-Zavala, Antennas and Propagation for Wireless communication systems.

[14] M. Hata, Empirical formula for propagation loss in Land Mobile radio services, IEEE Transactions on Vehicular Technology, Vol. 29, no. 3, August 1980.

[15] D. J. Cichon, T. Kurner, Propagation prediction models, COST 231 Final Rep, http://www.lx.it.pt/cost231/final_report.htm

[16] J. Walfisch, H. L. Bertoni, A Theoretical Model of UHF Propagation in Urban Environments, IEEE Trans. Antennas Propagat., Vol. 36, pp. 1788–1796, December 1988.

[17] F. Ikegami, S. Yoshoida, T. Takeuchi, M. Umehira, Propagation Factors Controlling Mean Field Strenght on Urban Streets, IEEE Trans. Antennas Propagat., Vol. 32, pp. 822-829, December 1984.

[18] Geodetska uprava Republike Slovenije (Surveying and Mapping Authority of the Republic of Slovenia, https://www.e-prostor.gov.si/podrocja/drzavni-topografski-sistem/drugo/razno/epsg-kode-za-slovenijo/

[19] Sandi Berk, Stari in novi državni horizontalni koordinatni sistem ter stara in nova državna kartografska projekcijaGeodetska uprava Republike Slovenije, 20. 03. 2008, https://www.e-prostor.gov.si/fileadmin/DPKS/Navodila/Stari_in_novi_DKS_in_DKP_2008_GURS.pdf

[20] Sandi Berk, Danijel Boldin, Slovenski referenčni koordinatni sistemi v okolju GIS (Slovenian coordinate reference systems in GIS environment), Geodetski Vestnik 61(1), pp. 91–101, March 2017, http://www.geodetski-vestnik.com/61/1/gv61-1_berk.pdf , http://www.geodetski-vestnik.com/62/4/gv62-4_vugrin.pdf

[21] epsg.io, Coordinate Systems Worldwide, https://epsg.io/

[22] Geodetska uprava Republike Slovenije (Surveying and Mapping Authority of the Republic of Slovenia), https://www.e-prostor.gov.si/podrocja/drzavni-topografski-sistem/topografski-podatki/, https://ipi.eprostor.gov.si/jgp/data

[23] OpenStreetMap, https://www.openstreetmap.org

[24] Wikipedia: Copernicus Programme, https://en.wikipedia.org/wiki/Copernicus_Programme

[25] Copernicus Global Land Service, https://land.copernicus.eu/global/, https://land.copernicus.eu/en/map-viewer