

"SOLAR ELECTRICITY" INDICATOR

thanks to

GRASS, PYTHON & LEAFLET

Jean Pierre Huart - FOSS4G - 2016-09-22

THE LEADING PROJECT

(where ideas come from...)

ENERGIZAIR

The screenshot shows the homepage of the EnergizAIR website. At the top, there's a banner featuring the EnergizAIR logo (three yellow circles) and a photograph of industrial equipment near water. Below the banner is a navigation bar with links: Home, About, Media integration, Your weather forecast, Contact, and Saturday, 27 August. A small European Union flag icon is also present.

Welcome to the EnergizAIR project!

EnergizAIR adds positive indicators about the part of the energy needs that were covered thanks to renewable energy sources in the weather forecast.

Belgium, France, Italy, Portugal and Slovenia are part of this Intelligent Energy Europe project, which has now welcomed Germany, Hungary, Spain, Sweden and the UK. The new countries are on the starting block to get to their weather forecasts with EnergizAIR indicators.

EnergizAIR's indicators (last seven days)

A map of Europe showing the locations of renewable energy installations. The countries highlighted in yellow are: United Kingdom, Sweden, Belgium, France, Germany, Hungary, Portugal, Spain, and Italy. To the left of the map, there are three icons with labels: Solar PV (blue house with sun), Wind (green wind turbine), and Solar Thermal (red building with sun).

WATCH THIS

ENERGIZAIR ON YOUR WEBSITE

In the spotlight

Battery assets lead winning projects in National Grid tender
<https://t.co/oxPmbhZhZK>

@ Weather Energy

RT @lgsolaruk: What a story - #solar and wind 'cheaper than new nuclear' by the time Hinkley is built <https://t.co/K5D5A3aw5W> #solarwin

@ Weather Energy

<http://energizair.eu>

La météo des énergies renouvelables

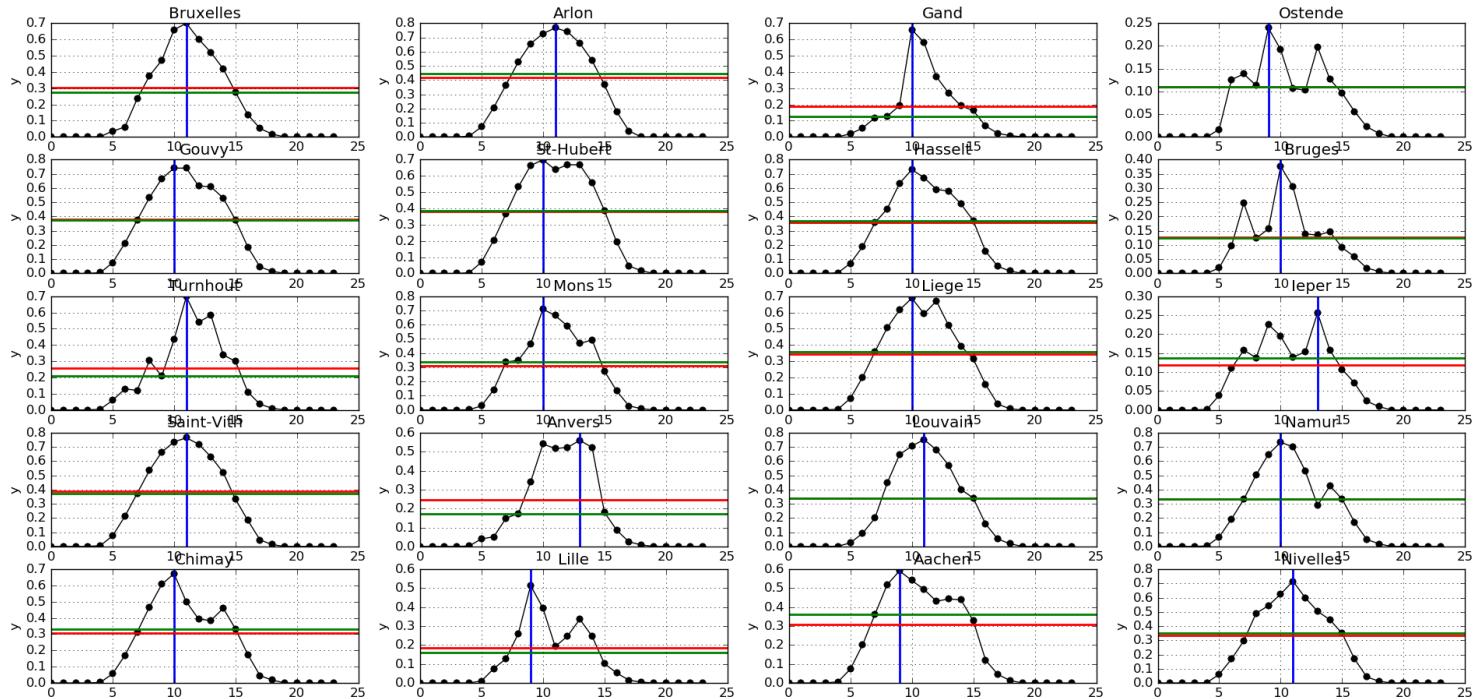


<http://www.meteo-renouvelable.be>

DATA SOURCES

The french partner of EnergizAIR (Hespul) provides hourly forecasts of PV production for 20 points in Belgium (or very near to it).

map_daily_2015_08_11



How can we summarize this to be presented in 20" on TV?

Can we build a map about it?

GRASS

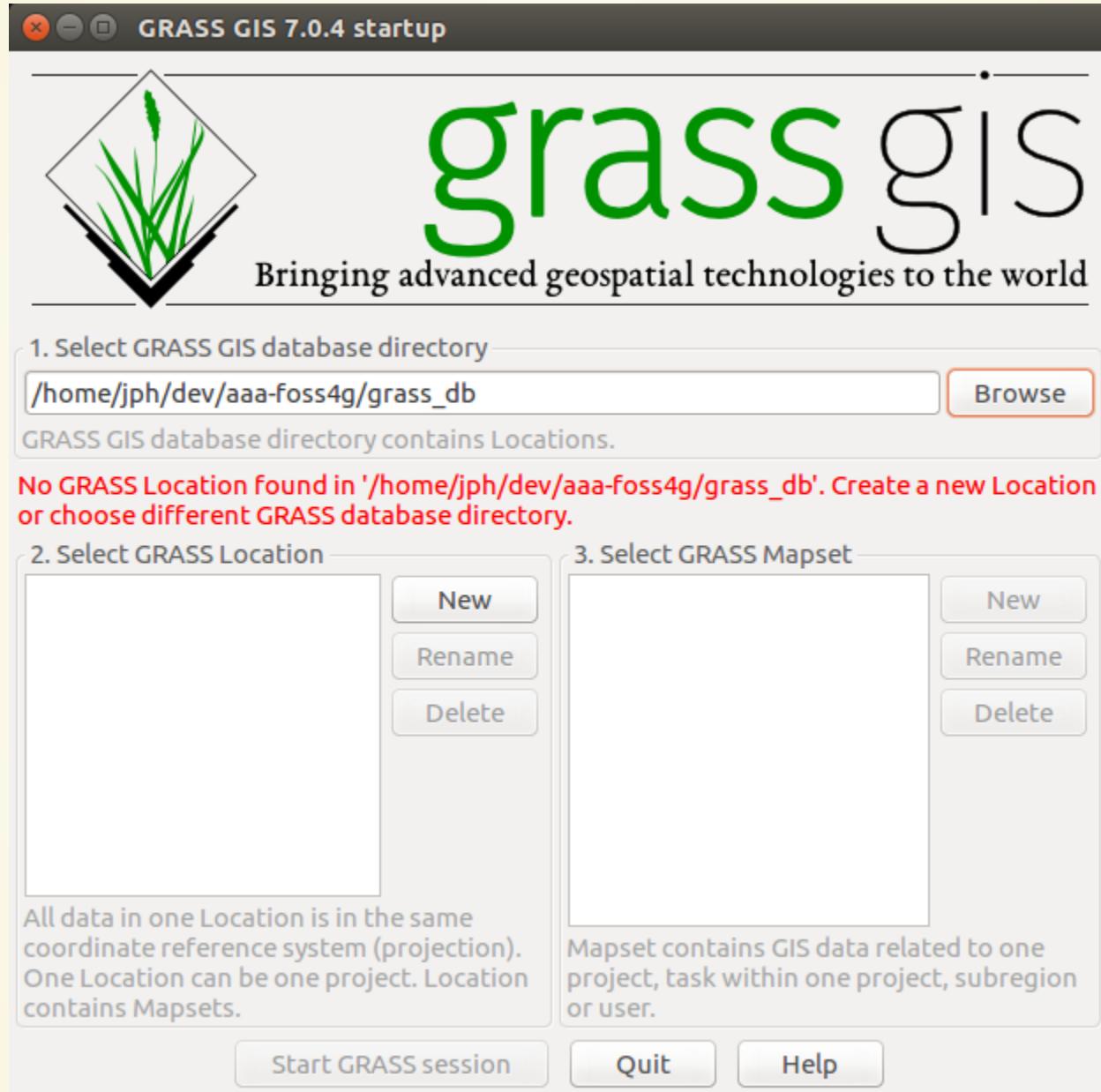
The help structure disappointed me, but it's well done.

Subscribe to the mailing list:

<http://lists.osgeo.org/mailman/listinfo/grass-user>

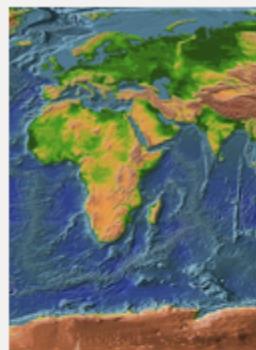
HOW TO FIND YOUR WAY IN GRASS

- Launch the GUI
- Selecting the GIS Database directory
- Create your LOCATION
- Create your MAPSET
- Start the GRASS session





Define new GRASS Location



Define GRASS Database and Location Name

GIS Data Directory: [Browse](#)

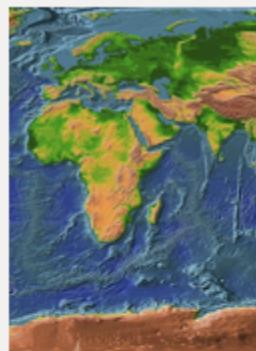
Project Location:

Location Title:

[Help](#)[< Back](#)[Next >](#)[Cancel](#)



Define new GRASS Location



Choose method for creating a new location

- Select EPSG code of spatial reference system
- Read projection and datum terms from a georeferenced data file
- Read projection and datum terms from a Well Known Text (WKT) .prj file
- Select coordinate system parameters from a list
- Specify projection and datum terms using custom PROJ.4 parameters
- Create a generic Cartesian coordinate system (XY)

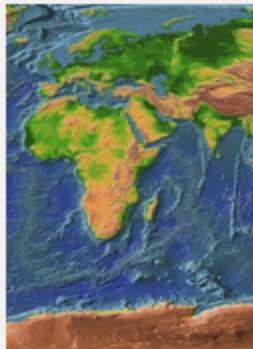
Help

< Back

Next >

Cancel

Define new GRASS Location



Choose EPSG Code

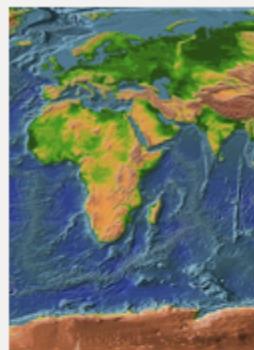
Path to the EPSG-codes file:

EPSG code:

Code	Description	Parameters
4215	Belge 1950	+proj=longlat +ellps...
4313	Belge 1972	+proj=longlat +ellps...
4809	Belge 1950 (Brussels)	+proj=longlat +ellps...
6190	Belge 1972 / Belgian Lambert 72 + Osten...	+proj=lcc +lat_1=51.1...
21500	Belge 1950 (Brussels) / Belge Lambert 50	+proj=lcc +lat_1=49.8...
31300	Belge 1972 / Belge Lambert 72	+proj=lcc +lat_1=49.8...
31370	Belge 1972 / Belgian Lambert 72	+proj=lcc +lat_1=51.1...



Define new GRASS Location



Summary

GRASS Database: /home/jph/dev/aaa-foss4g/grass_db

Location Name: foss4g

Location Title: foss4g

Projection: EPSG code 31370 (Belge 1972 / Belgian Lambert 72)

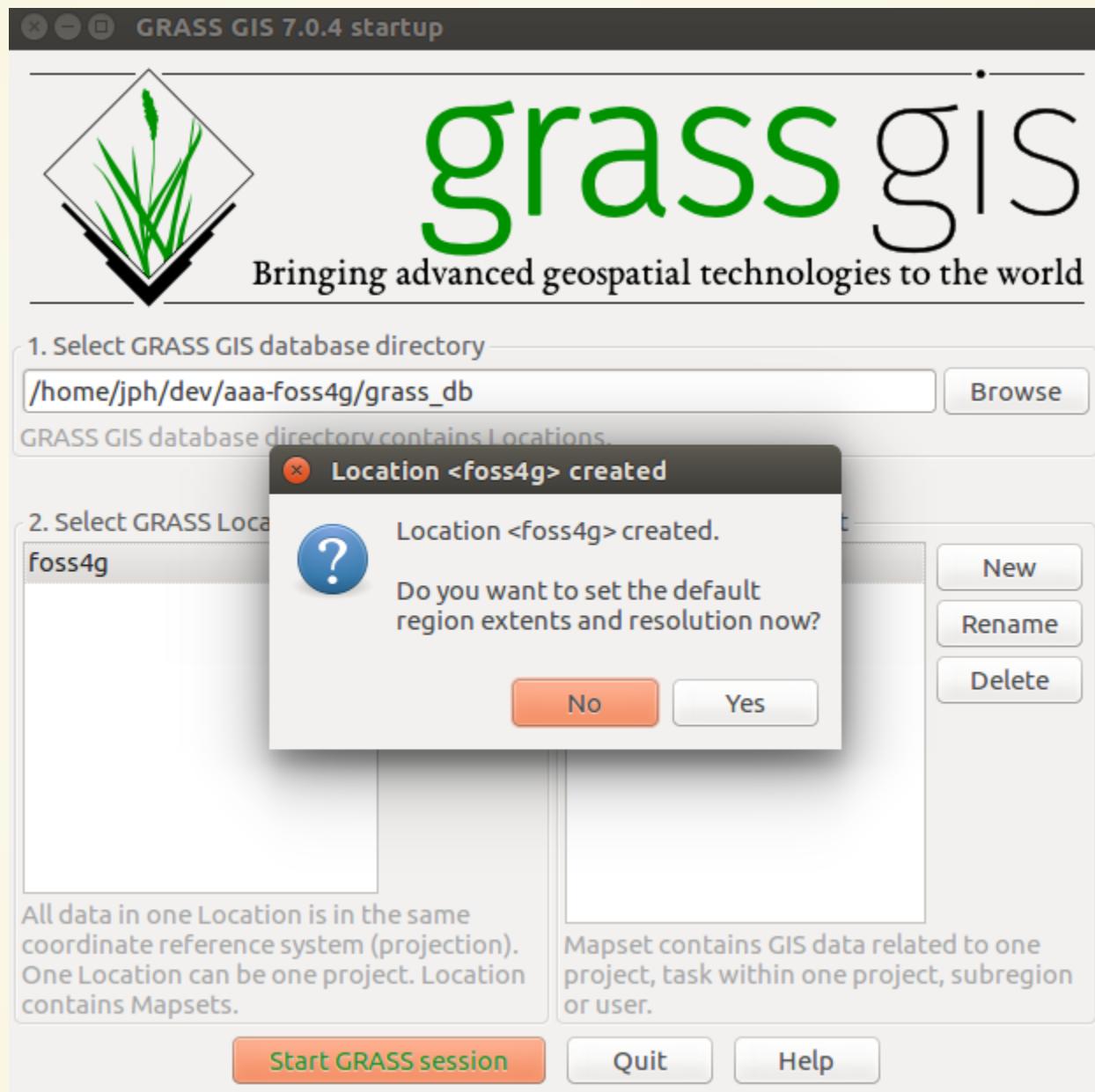
PROJ.4 definition: +proj=lcc
(non-definitive) +lat_1=51.16666723333333
+lat_2=49.8333339
+lat_0=90
+lon_0=4.367486666666666
+x_0=150000.013
+y_0=5400088.438
+no_defs

Help

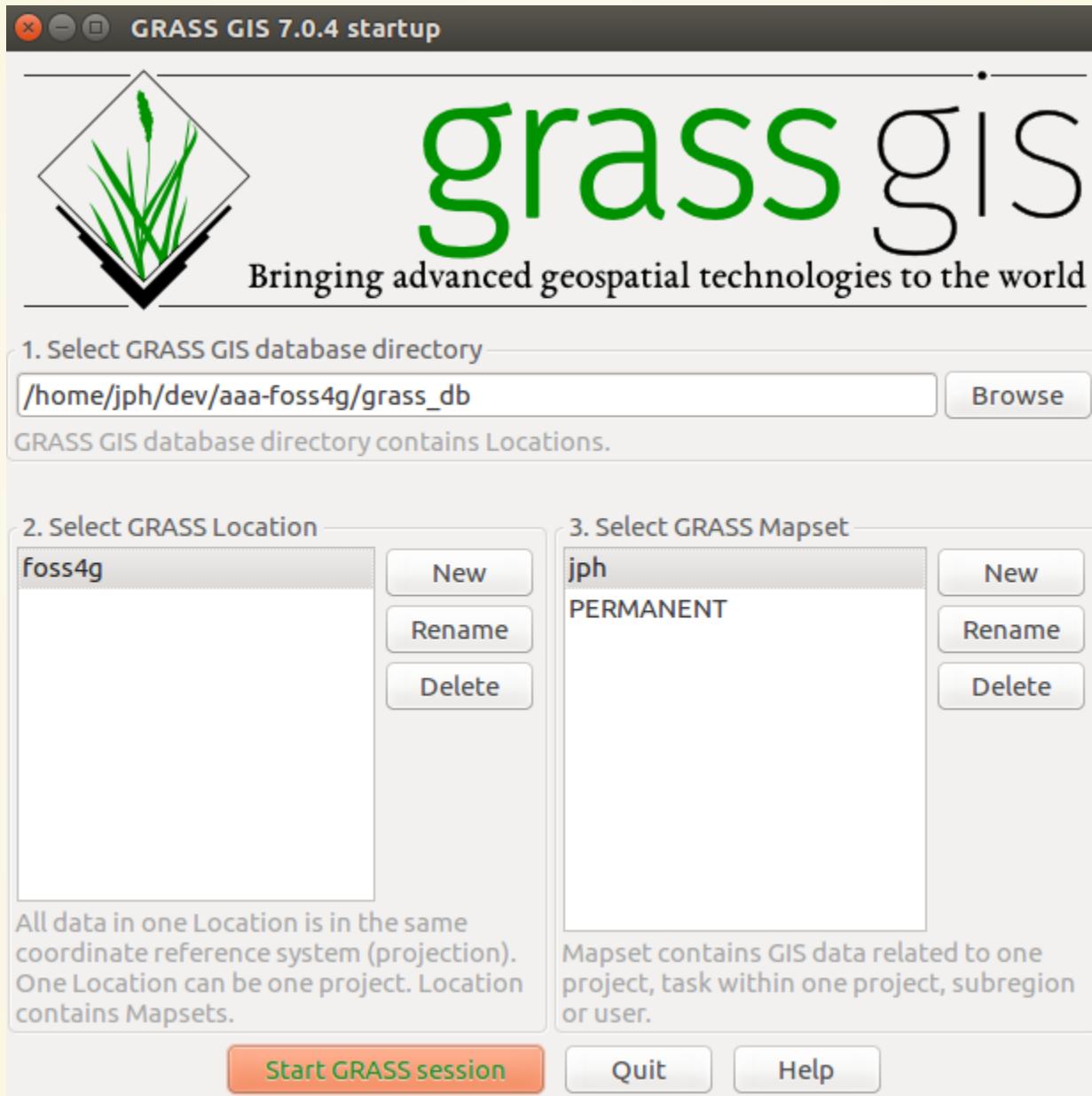
< Back

Finish

Cancel







ADD THE VECTOR MAP OF BELGIUM

Import vector data

Settings
Load settings: Save Remove

Source type
 File Directory Database Protocol

Source settings
File:

List of vector layers - right click to (un)select all

Layer id	Layer name	Feature type	Projection	Name for output GRASS map
<input checked="" type="checkbox"/> 1	be_adm0	polygon	Yes	be_adm0

Options

Do not clean polygons (not recommended)
 Extend region extents based on new dataset
 Override dataset projection (use location's projection)
 Limit import to the current region
 Do not create attribute table
 Change column names to lowercase characters

Encoding:

Allow output files to overwrite existing files
 Add imported layers into layer tree
 Close dialog on finish

GRASS GIS 7.0.4 Layer Manager

Number of isles: 3

Finding centroids for OGR layer <be_adm0>...

Writing centroids...

3 input polygons

Total area: 3.07058E+10 (3 areas)

Copying features...

Building topology for vector map <be_adm0@jph>...

Registering primitives...

6 primitives registered

5751 vertices registered

Building areas...

3 areas built

3 isles built

Attaching islands...

Attaching centroids...

Number of nodes: 3

Number of primitives: 6

Number of points: 0

Number of lines: 0

Number of boundaries: 3

Number of centroids: 3

Number of areas: 3

Number of isles: 3

(Fri Sep 16 21:44:44 2016) Command finished (0 sec)

Output window

Clear Save Log file

Press Tab to display command help, Ctrl+Space to autocomplete

Map layers Command console Search modules Python shell

GRASS GIS 7.0.4 Map Display: 1 - Location: foss4g@jph

Import vector data

Settings

Load settings: Save Remove

Source type

File Directory Database Protocol

Source settings

File: /home/jph/dev/aaa-foss4g/gis_data/be_31370/be_adm0.shp Browse

List of vector layers - right click to (un)select all

Layer id	Layer name	Feature type	Projection	Name for output GRASS map
1	be_adm0	polygon	Yes	be_adm0

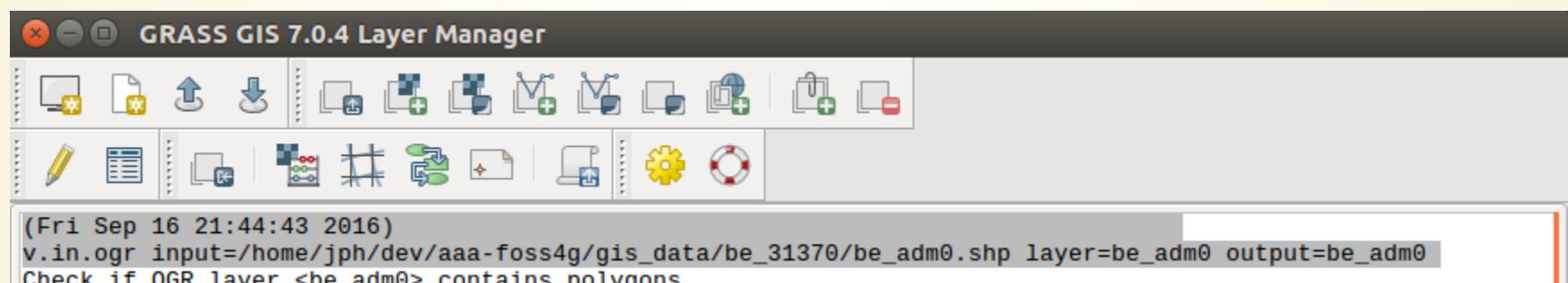
Options

- Do not clean polygons (not recommended)
- Extend region extents based on new dataset
- Override dataset projection (use location's projection)
- Limit import to the current region
- Do not create attribute table
- Change column names to lowercase characters

Encoding:

- Allow output files to overwrite existing files
- Add imported layers into layer tree





Output window**Command prompt**

```
v.in.ogr input=/home/jph/dev/aaa-foss4g/gis_data/be_31370/be_adm2.shp layer=be_adm2 output=be_adm2
```

Press Tab to display command help, Ctrl+Space to autocomplete

```
d.vect map=be_adm0@jph fcolor=none color=black width=1 size=5 type=point,line,boundary,area,face icon=basic/x
```

GRASS GIS 7.0.4 Layer Manager

Total area: 3.07058E+10 (21 areas)
Overlapping area: 3.14456E-05 (5 areas)

Copying features...
Building topology for vector map <be_adm2@jph>...
Registering primitives...
74 primitives registered
8228 vertices registered
Building areas...
21 areas built
4 isles built
Attaching islands...
Attaching centroids...
Number of nodes: 37
Number of primitives: 74
Number of points: 0
Number of lines: 0
Number of boundaries: 54
Number of centroids: 20
Number of areas: 21
Number of isles: 4

Some input polygons are overlapping each other.
If overlapping is not desired, the data need to be cleaned.
The input could be cleaned by snapping vertices to each other.
Estimated range of snapping threshold: [1e-10, 0.1]
Try to import again, snapping with at least 1e-10: 'snap=1e-10'
(Fri Sep 16 21:47:59 2016) Command finished (0 sec)

GRASS GIS 7.0.4 Map Display: 1 - Location: foss4g@jph

2D view

Output window

Command prompt

Clear Save Log file Clear

Press Tab to display command help, Ctrl+Space to autocomplete

Map layers Command console Search modules Python shell

Map layers Command console Search modules Python shell

d.vect map=be_adm2@jph fcolor=none color=black width=1 size=5 type=point,line,boundary,area,face id 35918.13; 97849.74

Coordinates ▲▼

Render

Belgium:

```
v.in.ogr  
input=/home/jph/dev/aaa-foss4g/gis_data/be_31370/be_adm0.shp  
layer=be_adm0 output=be_adm0
```

Belgian provinces:

```
v.in.ogr  
input=/home/jph/dev/aaa-foss4g/gis_data/be_31370/be_adm2.shp  
layer=be_adm2 output=be_adm2
```

SETUP YOUR PROJECT REGION

The geographic area in which GRASS should work:

- geographical projection (e.g. Belgian Lambert 72, etc)
- geographical extension, i.e. the North/South/East/West limits of the area covered
- number of columns and number of rows for the data
- resolution, i.e. the extension divided by the number of rows (N-S resolution), respectively columns (E-W resolution).

In other words, *you need a good friend* to set the displayed region:

```
g.region n=243900 s=21200 e=295950 w=23700 rows=4454 cols=5445 nsres=50
```

ADD THE PV FORECASTS AS VECTOR POINT MAP

v.in.ascii [vector, import, ASCII]

Creates a vector map from an ASCII points file or ASCII vector file.

Required Name of input file to be imported: * (input=name)
`/home/jph/dev/aaa-foss4g/gis_data/data/test_map_2015-08-11_10.csv`

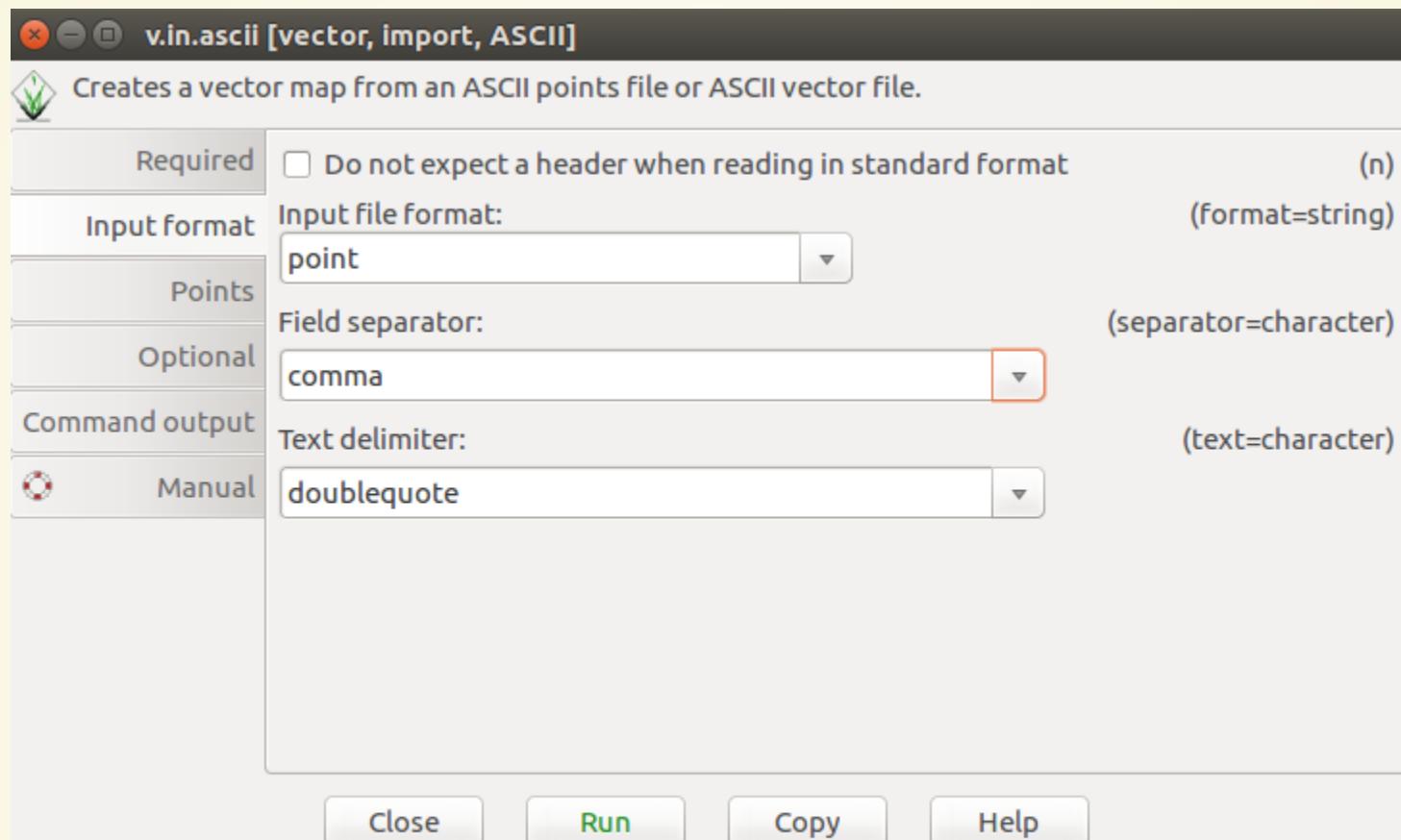
or enter values directly:

Input format Points
Optional
Command output
Manual

Name for output vector map: * (output=name)
`data_map_2015_08_11_10`

Add created map(s) into layer tree
 Close dialog on finish

`v.in.ascii input=/home/jph/dev/aaa-foss4g/gis_data/data/test_map_2015-08-11_10.csv output=data_map_`



```
v.in.ascii input=/home/jph/dev/aaa-foss4g/gis_data/data/test_map_2015-08-11_10.csv output=data_map_1
```

v.in.ascii [vector, import, ASCII]

Creates a vector map from an ASCII points file or ASCII vector file.

Required Do not create table in points mode (t)

Input format Do not build topology in points mode (b)

Points Only import points falling within current region (points mode) (r)

Optional Ignore broken line(s) in points mode (i)

Number of header lines to skip at top of input file (points mode): (skip=integer)

Command output (columns:string)

Manual (x=integer)

Column definition in SQL style (points mode):

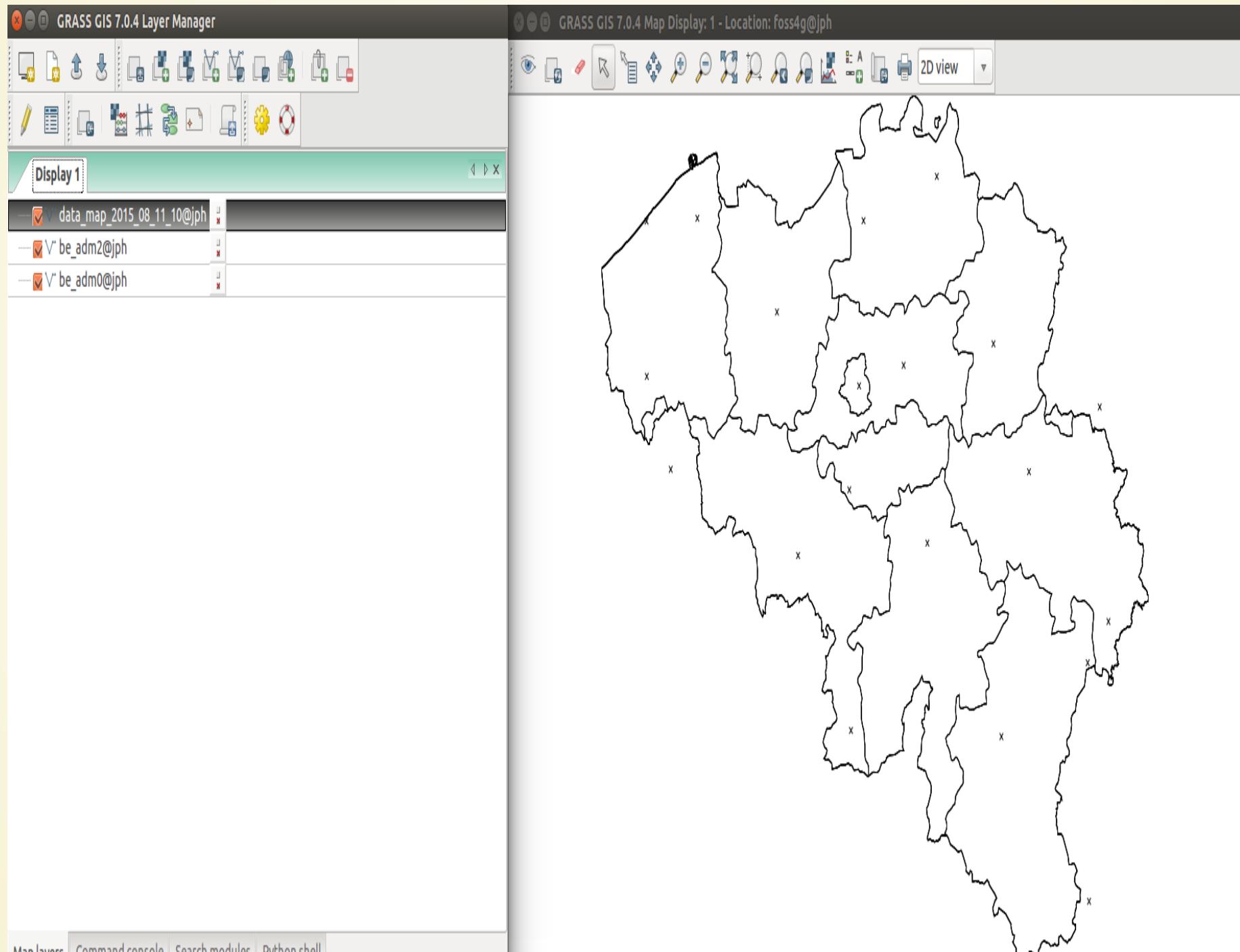
Number of column used as x coordinate (points mode): (y=integer)

Number of column used as y coordinate (points mode): (y=integer)

Add created map(s) into layer tree

Close dialog on finish

v.in.ascii input=/home/jph/dev/aaa-foss4g/gis_data/data/test_map_2015-08-11_10.csv output=data_map_2015_08_11_10 separator=comma skip=1



Map layers

Command console

Search modules

Python shell

75462.32; 69058.80

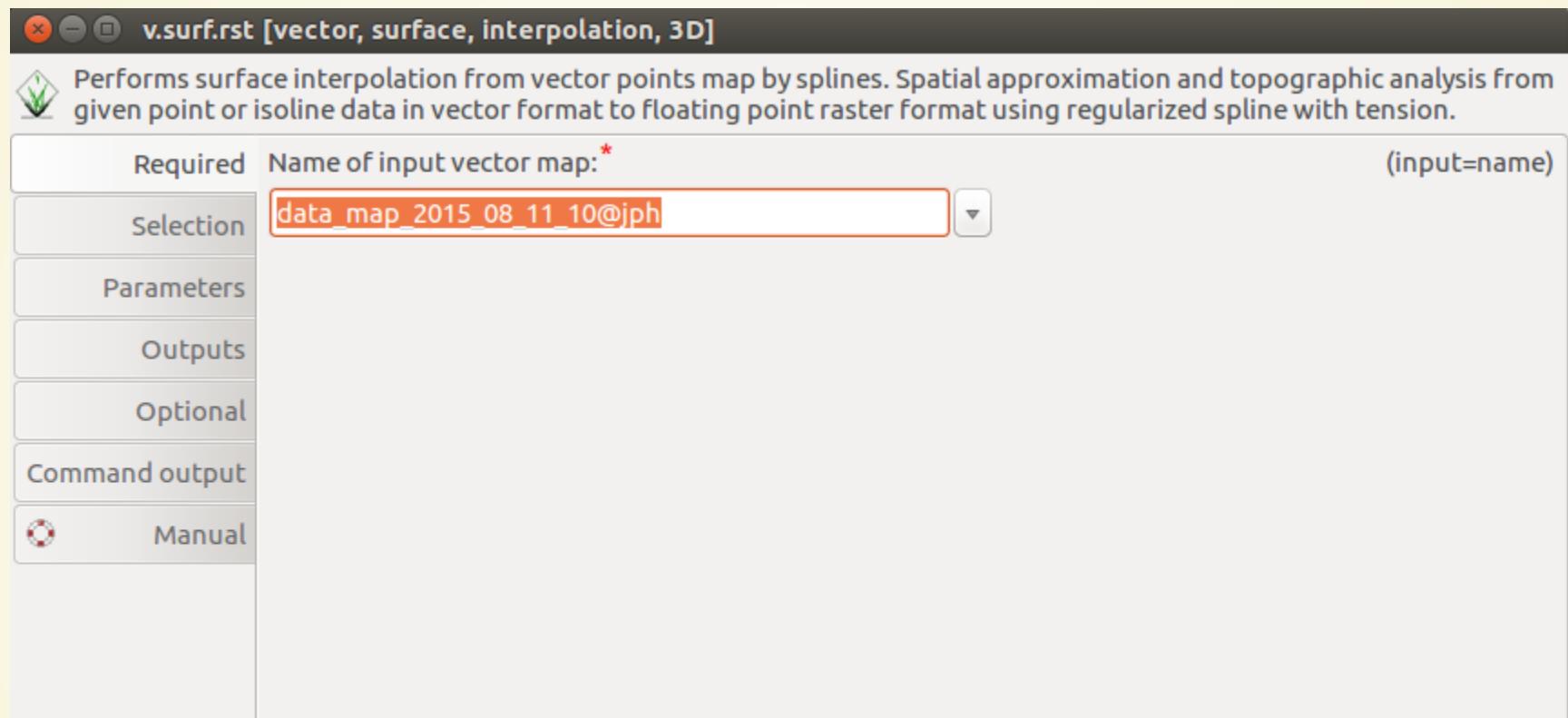
Coordinates

 Render

All this is summarized in this command:

```
v.in.ascii --overwrite  
input=/home/jph/dev/aaa-foss4g/gis_data/data/test_map_2015-08-11_10.csv  
output=data_map_2015_08_11_10  
separator=comma  
skip=1
```

BUILD A RASTER FROM THE VECTOR POINT MAP



Add created map(s) into layer tree

Close dialog on finish

`v.surf.rst input=data_map_2015_08_11_10@jph`

v.surf.rst [vector, surface, interpolation, 3D]

Performs surface interpolation from vector points map by splines. Spatial approximation and topographic analysis from given point or isoline data in vector format to floating point raster format using regularized spline with tension.

Required

Perform cross-validation procedure without raster approximation (c)

Use scale dependent tension (t)

Selection

Parameters

Name of the attribute column with values to be used for approximation: (zcolumn=name)
dbl_3

Outputs

Name of raster map used as mask: (mask=name)

Optional

Tension parameter: (tension=float)
40.

Command output

Manual

Smoothing parameter: (smooth=float)
1

Name of the attribute column with smoothing parameters: (smooth_column=string)

Add created map(s) into layer tree

Close dialog on finish

v.surf.rst input=data_map_2015_08_11_10@jph zcolumn=dbl_3 smooth=1

v.surf.rst [vector, surface, interpolation, 3D]

Performs surface interpolation from vector points map by splines. Spatial approximation and topographic analysis from given point or isoline data in vector format to floating point raster format using regularized spline with tension.

Required

Output partial derivatives instead of topographic parameters (d)

Selection

Name for output surface elevation raster map: (elevation=name)
rst_20150811H10

Parameters

Outputs

Name for output slope raster map: (slope=name)

Optional

Name for output aspect raster map: (aspect=name)

Command output

Manual

Name for output profile curvature raster map: (pcurvature=name)

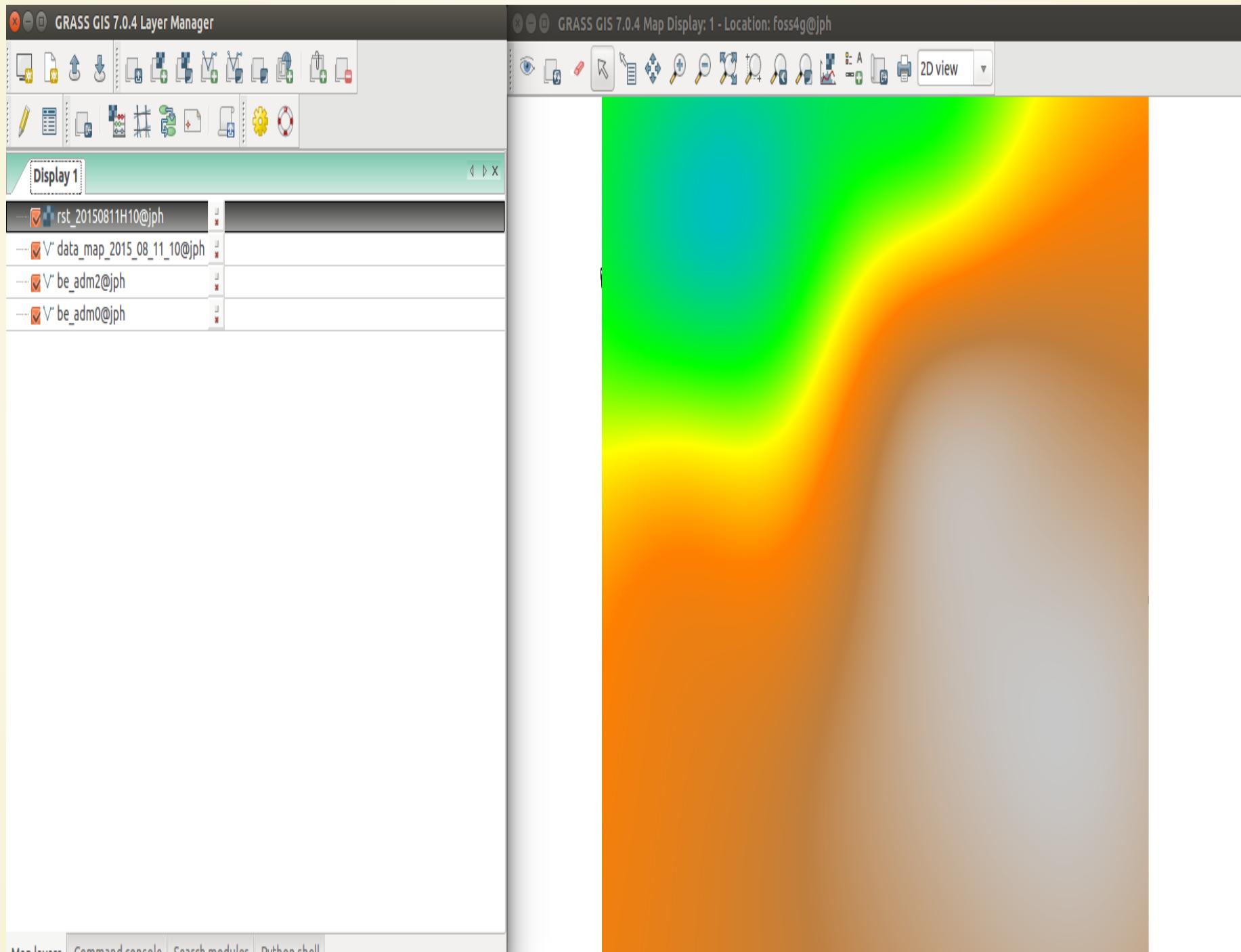
Name for output tangential curvature raster map: (tcurvature=name)

Close Run Copy Help

Add created map(s) into layer tree

Close dialog on finish

```
v.surf.rst input=data_map_2015_08_11_10@jph zcolumn=dbl_3 elevation=rst_20150811H10 smooth=1
```



Map layers

Command console

Search modules

Python shell

d.rast map=rst_20150811H10@jph

56730.86; 72874.46

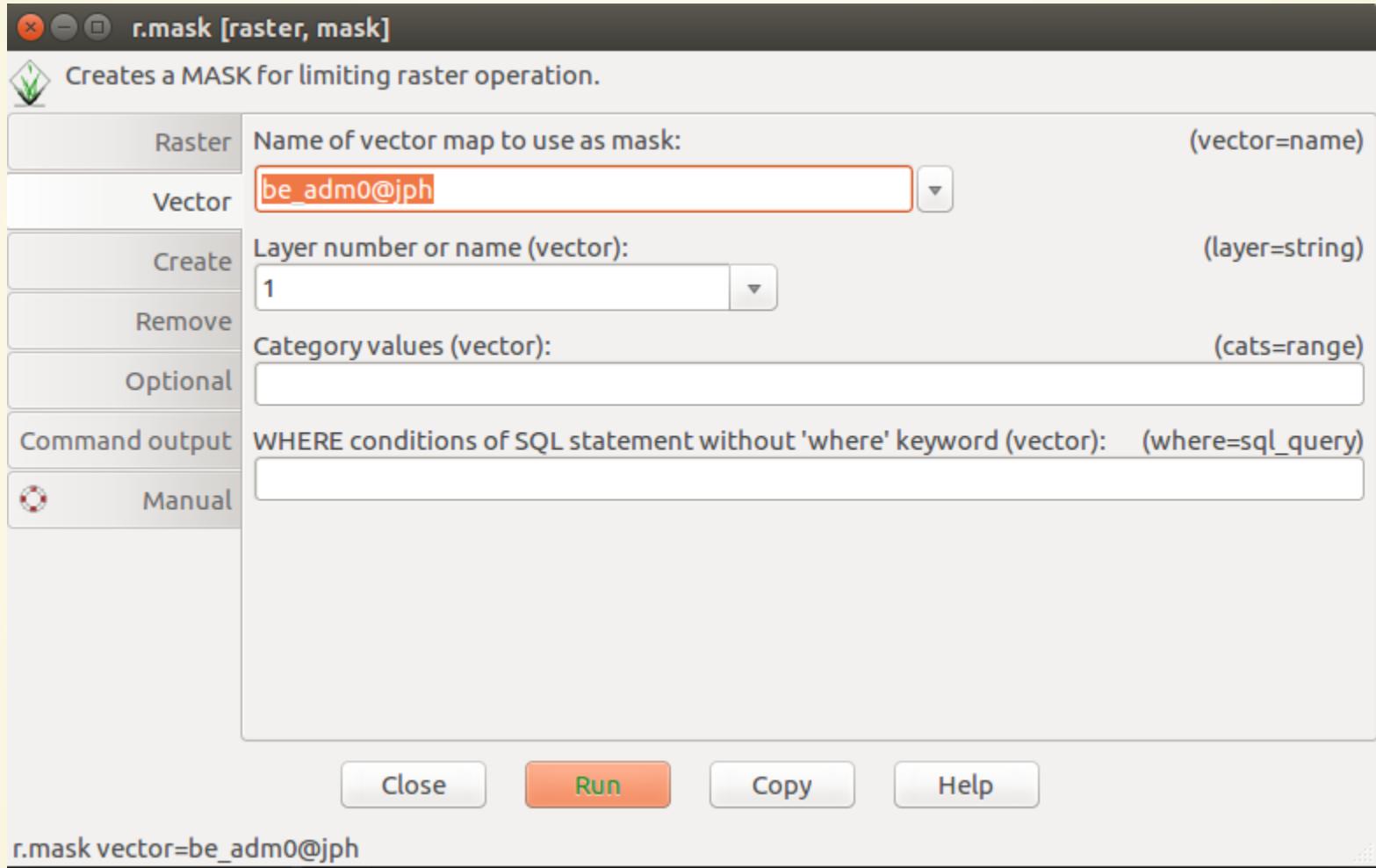
Coordinates

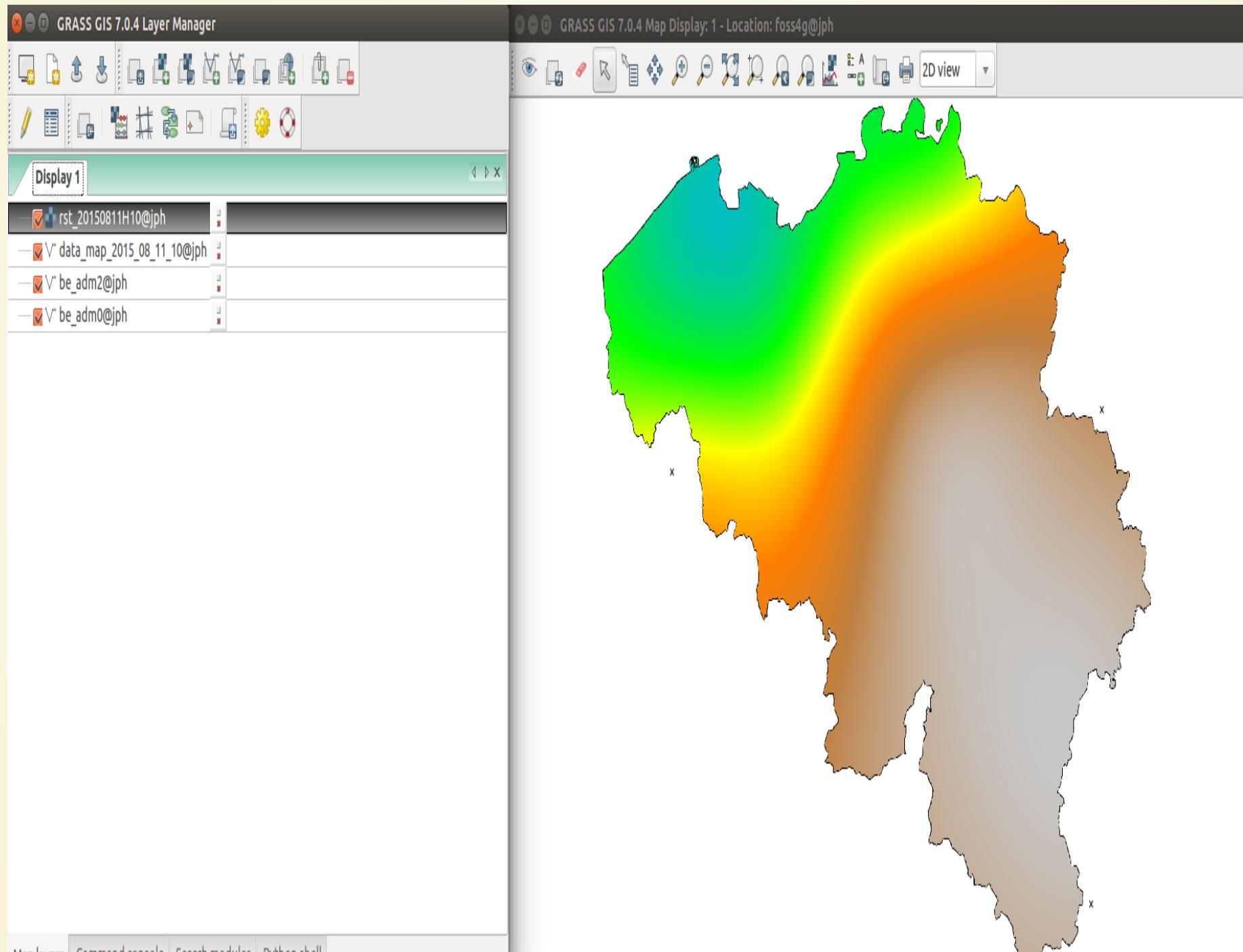
Render

We have chosen the Regularized Spline Tension (rst) interpolation:

```
v.surf.rst --overwrite  
input=data_map_2015_08_11_10@jph  
zcolumn=dbl_3  
elevation=rst_20150811H10  
smooth=1
```

ADD A MASK





Map layers Command console Search modules Python shell

d.rast map=rst_20150811H10@jph

-11604.26; 141209.59

Coordinates

Render

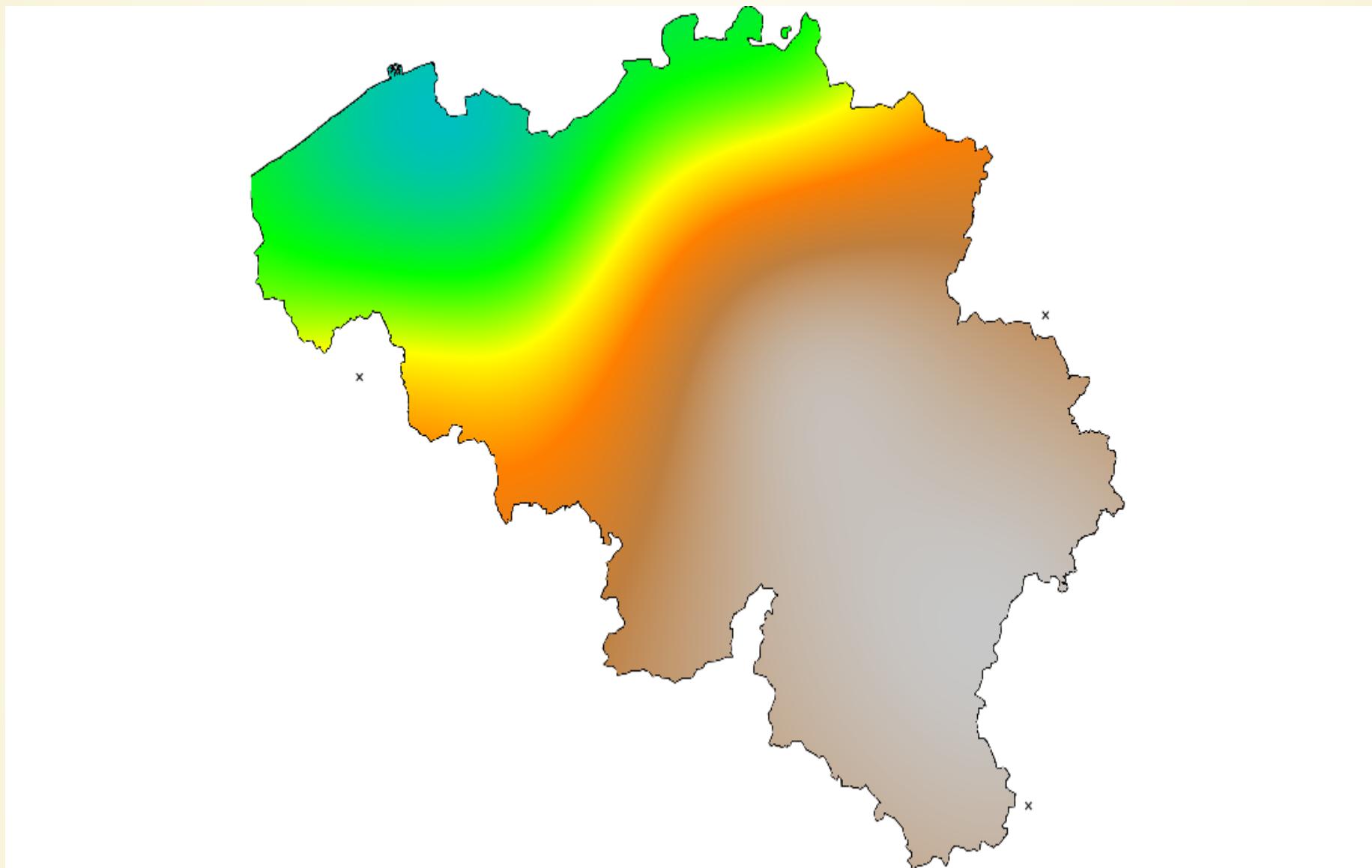
To limit my raster to the country limits:

```
r.mask --overwrite vector=be_adm0@jph
```

To directly build the raster including the mask:

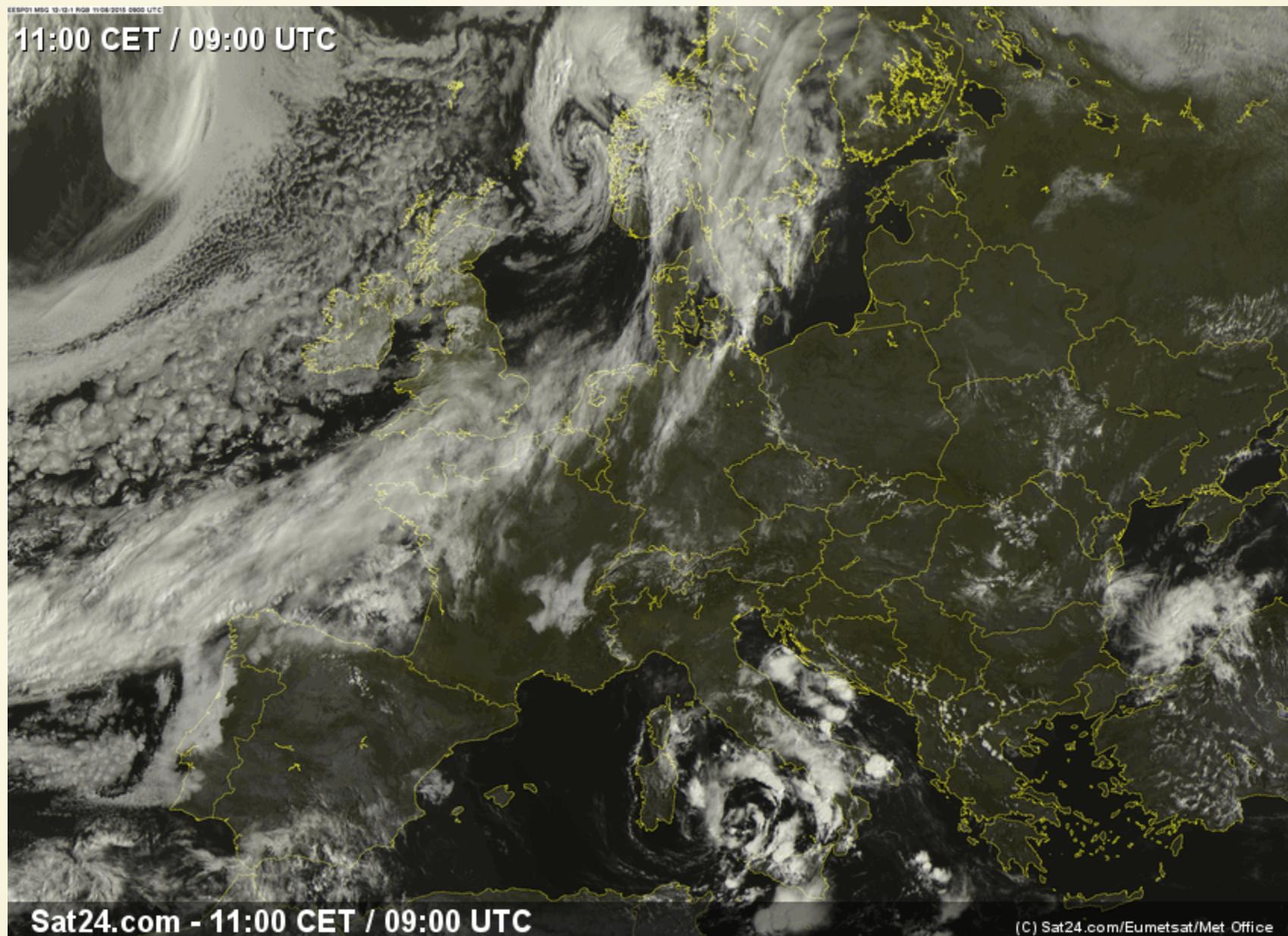
```
v.surf.rst --overwrite  
input=test_map_2015_08_11_10@jph  
zcolumn=dbl_3  
elevation=rst_20150811H10  
smooth=1  
mask=MASK@jph
```

SAVE THE RESULT AS AN IMAGE



EERSPO1 MSG 12/12/1 RGB 11/08/2015 0900 UTC

11:00 CET / 09:00 UTC



Sat24.com - 11:00 CET / 09:00 UTC

(C) Sat24.com/Eumetsat/Met Office

We would like to use an indicator that can be easily interpreted by anybody whatever installation he is using and something valid across seasons.

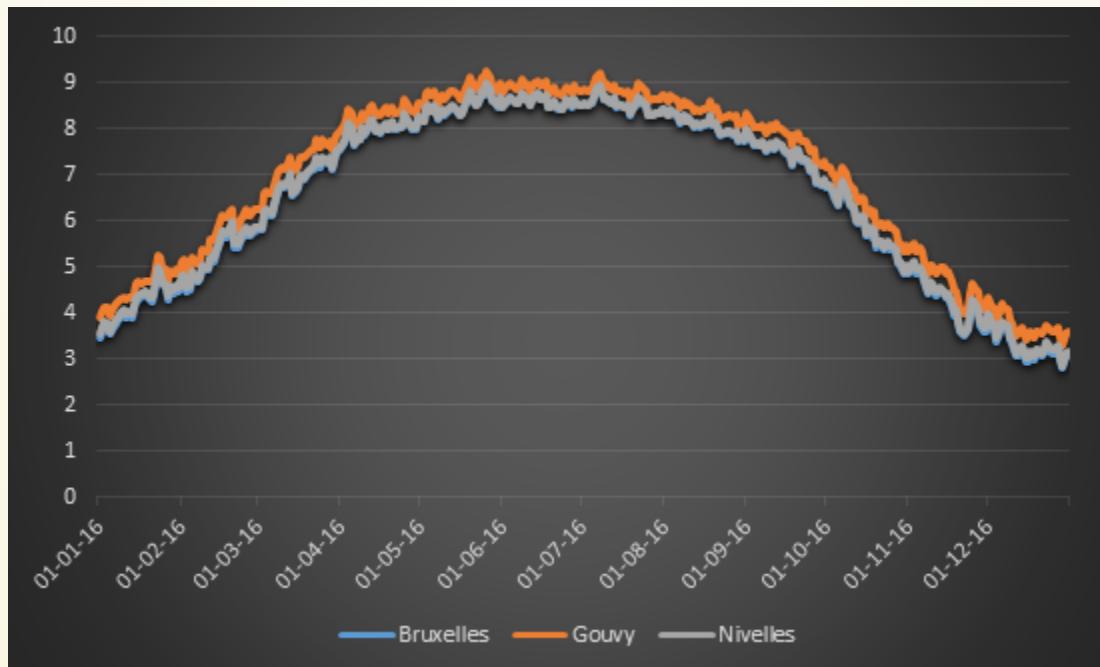
We decided to display a percentage of the **clear sky PV production**.

In other words, **is it a bright day for the season?**

TMY CLEAR SKY

Clear sky solar energy received on an inclined plane during a normal year.

This is an amount of solar energy expressed in kWh/m²*day
(source météonorm).



A	B	C	D	E	F	G
Year	Month	Day	Hour	Extraterrestrial Horizontal Radiation [Wh/m^2]	Extraterrestrial Direct Normal Radiation [Wh/m^2]	Global Horizontal Radiation [Wh/m^2]
5	1	1	1	0	0	0
5	1	1	2	0	0	0
5	1	1	3	0	0	0
5	1	1	4	0	0	0
5	1	1	5	0	0	0
5	1	1	6	0	0	0
5	1	1	7	0	0	0
5	1	1	8	0	0	0
5	1	1	9	2	1412	0
5	1	1	10	111	1412	23
5	1	1	11	253	1412	58
5	1	1	12	348	1412	89
5	1	1	13	391	1412	112
5	1	1	14	377	1412	104
5	1	1	15	309	1412	78
5	1	1	16	190	1412	42
5	1	1	17	37	1412	2
5	1	1	18	0	0	0
5	1	1	19	0	0	0
5	1	1	20	0	0	0
5	1	1	21	0	0	0
5	1	1	22	0	0	0
5	1	1	23	0	0	0
5	1	1	24	0	0	0
5	1	2	1	0	0	0
5	1	2	2	0	0	0
5	1	2	3	0	0	0
5	1	2	4	0	0	0
5	1	2	5	0	0	0
5	1	2	6	0	0	0
5	1	2	7	0	0	0
5	1	2	8	0	0	0
5	1	2	9	2	1412	0
5	1	2	10	112	1412	43
5	1	2	11	254	1412	110
5	1	2	12	350	1412	172
5	1	2	13	392	1412	187
5	1	2	14	379	1412	161
5	1	2	15	311	1412	119
5	1	2	16	193	1412	64

5	1	2	17	39	1412	2
5	1	2	18	0	0	0
5	1	2	19	0	0	0
5	1	2	20	0	0	0
5	1	2	21	0	0	0
5	1	2	22	0	0	0
5	1	2	23	0	0	0
5	1	2	24	0	0	0
...						

▶ ▷ + Uccle-horizon

CSV DATA FILE

Result daily power is expressed in percentage of clear sky.
Result hourly power is expressed in Wh/Wp.

A	B	C	D	E	F	G	H	I
SOLPOINT	DATE	CLEARSKY	PEAK	PEAKPOWER	MEAN	MEDIAN	OPTIMUM	GRAPHOPTIMUM
Bruxelles	2016-02-14	11.3316441441	11	0.0683333333333	0.0447222222222	0.0580555555556	[10-11 13-15]	oM_oMo
Arlon	2016-02-14	22.2972972973	9	153		88	0.0901666666666	[9-9 11-12 14-15]
Gand	2016-02-14	17.6858108108	13		118	0.0698	0.0748333333333	[11-14 16-16]
Ostende	2016-02-14	33.5810810811	12	0.400666666667	0.132533333333	0.0876666666667	[11-14]	oMoo
Gouvy	2016-02-14	19.3918918919	9	0.105333333333	0.0765333333333	0.0793333333333	[9-9 11-14 16-16]	M_oooo o
St-Hubert	2016-02-14	14.6199324324	14	0.105333333333	0.0577	0.0656666666666	[11-15]	oooMo
Hasselt	2016-02-14	12.1199324324	14	0.0766666666667	0.047833333333		57 [10-14]	ooooM
Bruges	2016-02-14	34.2736486486	13	251	0.135266666667		127 [12-15]	oMoo
Turnhout	2016-02-14	15.152027027	12	0.097333333333	0.0598	0.0701666666667	[11-15]	oMooo
Mons	2016-02-14	14.2652027027	12	0.089333333333	0.0563	0.0601666666666	[11-14 16-16]	oMoo o
Liege	2016-02-14	11.2922297297	11	69	0.0445666666667		48 [10-14]	oMooo
Ieper	2016-02-14	35.0337837838	13	373	0.138266666667	0.122166666667	[11-15]	ooMoo
Saint-Vith	2016-02-14	18.4121621622	9	0.130333333333	0.0726666666666	0.0791666666667	[9-9 11-14]	M_oooo
Anvers	2016-02-14	13.1418918919	14	0.077333333333	0.0518666666667	0.0623333333333	[10-15]	ooooMo
Louvain	2016-02-14	13.5557432432	11	0.1206666666667	0.0535	0.0555	[10-12 14-15]	oMo oo
Namur	2016-02-14	11.6891891892	12	78	0.0461333333333	0.0493333333334	[11-15]	oMooo
Chimay	2016-02-14	11.7803350225	12	0.0715277777778	0.0464930555556	0.0513888888889	[10-14]	ooMoo
Lille	2016-02-14	25.0823302294	12	0.183613445378	0.0989915966386	0.12268907563	[12-16]	Moooo
Aachen	2016-02-14	11.2837837838	9	67	0.0445333333333	0.0518333333333	[9-10 12-14]	Mo_ooo
Nivelles	2016-02-14	13.7162162162	12	0.107333333333	0.0541333333333		53 [11-15]	oMooo

PYTHON SCRIPTS

SCRIPTING GRASS

https://grasswiki.osgeo.org/wiki/Working_with_GRASS_without_a_terminal

The EnergizAIR GRASS session class.

```
mysession = Grass7Session()
mysession.start_grass()
mysession.gscript.message('Current GRASS GIS 7 environment:')
print mysession.gscript.gisenv()
""" Do what you want """
```

Now you have access to **gscript.run_command()** to run the following functions.

GENERATE MAPS

ADD THE PV FORECASTS AS VECTOR POINT MAP

```
def insert_vector_csv(self, input, output):
    columns = 'long double precision , lat double precision, power double precision'
    result = self.gscript.run_command(
        'v.in.ascii'
        , overwrite=True
        , input=input
        , separator='comma'
        , skip=1
        , output=output
        , columns=columns
        , quiet=not self.debug)
```

BUILD A RASTER FROM THE VECTOR POINT MAP

```
def create_rst_interpolation_raster(self, input, elevation, zcolumn,
                                     smooth=0, tension=40, npmin=20, segmax=40):

    result = self.gscript.run_command(
        'v.surf.rst'
        , overwrite=True
        , input='{0}@{1}'.format(input, self.mapset)
        , zcolumn=zcolumn
        , elevation=elevation
        , mask='MASK@{0}'.format(self.mapset)
        , smooth=smooth
        , tension=tension
        , npmin=npmin
        , segmax=segmax
        , quiet=not self.debug)
```

DEFINE THE COLOR PALETTE

Define the palette rules and colors in a separate file:

```
0 78:148:228
10.0 78:198:228
20.0 21:177:98
30.0 37:211:24
40.0 122:237:20
50.0 218:255:10
60.0 255:229:10
70.0 255:177:10
80.0 255:99:5
90.0 188:38:13
150.0 188:38:13
```

Pass it to the python script (rules):

```
def set_color_palette(self, map, rules):
    result = self.gscript.run_command(
        'r.colors'
        , map='{0}@{1}'.format(map, self.mapset)
        , rules=rules
        , quiet=not self.debug)
```

ADD A MASK

```
def set_mask(self, name):
    result = self.gsclient.run_command(
        'r.mask',
        , overwrite=True,
        , vector='{0}@{1}'.format(name, self.mapset)
        , quiet=not self.debug)
```

SAVE THE MAP IN A PNG FILE WITH TRANSPARENT BACKGROUND

```
def save_png(self, filename, raster, map, height=960, width=1280):
    # SETTINGS for PNG DRIVER
    os.system('rm {0}'.format(filename))
    os.environ['GRASS_RENDER_IMMEDIATE'] = 'png'
    os.environ['GRASS_RENDER_FILE'] = filename
    os.environ['GRASS_RENDER_FILE_READ'] = 'TRUE'
    os.environ['GRASS_RENDER_TRANSPARENT'] = 'TRUE'
    os.environ['GRASS_RENDER_HEIGHT'] = str(height)
    os.environ['GRASS_RENDER_WIDTH'] = str(width)

    self.gscript.run_command('d.rast'
                           , map='{0}@{1}'.format(raster, self.mapset)
                           , quiet=not self.debug)

    self.gscript.run_command('d.vect'
                           , map='{0}@{1}'.format(map, self.mapset)
                           , color='white'
                           , fill_color='none'
```

SAVE ALL VALUES AS XML AND JSON ARRAY

- To be able to build friendly reports for the radio stations.
- To pass values to the leaflet Iframe for popups.

BUILD A PDF FRIENDLY REPORT

```
""" Load the xml summary """
content, fc_date = self._generate_daily_xml_content(datafile, True)

""" Transform it in HTML format with XSL stylesheet """
htmlcontent = self.apply_xsl_transformation('radio_report.xsl', content)
with open('report.html', 'w') as f:
    f.write(htmlcontent.encode('latin1'))

""" Transform HTML in PDF """
os.system("wkhtmltopdf -q {0} {1}".format('report.html', 'report.pdf'))
```

NB to insert the image into the xml, it is encoded in base64

```
with open('mymap.png'), "rb") as f:
    myimage = f.read()
    image_io = myimage.encode("base64")
```

Bonjour à tous,

Voici la carte et les chiffres du 22/06/2016 calculés le 22/06/2016 à 11:35.



Médiocre Faible Moyen Bon Excellent

Localités Viva Namur

Localité	Où aussi	Appréciation Pic	Visualisation
Namur	Profondenville, Jambes, Champion, Wépion, Beez, Malonne, Bouge	Bon 13h	10 11 12 13 14 15 16 17
Chimay	Saint-Rémy, Virelle, Vaulx, Lompret, Forges, Robechies	Moyen 14h	10 11 12 13 14 15 16 17
St-Hubert	Sartay, Libin, Lorcy, Vesqueville, Poix-Saint-Hubert, Jenneville	Moyen 14h	10 11 12 13 14 15 16 17
Nivelles	Monstreux, Thines, Baulers, Witterzée, Rossignies, Croiseau	Bon 13h	10 11 12 13 14 15 16 17

LEAFLET

Used libraries:

- jquery-2.2.3.min.js
- leaflet.js
- proj4-compressed.js
- proj4leaflet.js

GLOBAL SETTINGS

Windows bounds:

```
beBounds : [[ 51.49, 2.54], [49.49, 6.42]],
```

Projection used by the map:

```
belProjection : "+proj=lcc +lat_1=51.16666723333333 +lat_2=49.8333339 +l  
EPSG31370 : new L.Proj.CRS("urn:ogc:def:crs:EPSG::31370", belProjection,  
    {  
        resolutions: [8192, 4096, 2048, 1024, 512, 256, 128],  
        origin: [0, 0]  
    }),
```

Solar image bounds in EPSG31370:

```
beimgBounds : L.bounds([12500.00, 244000.00], [308600.00, 21000.00]),
```

LOAD THE COUNTRY MAP

buildMap :

```
map = L.map('map-container',
            {center: [50.50, 4.50], zoom: 2.7, crs: EPSG31370});
addPane('beadm2', 400);
beadm2 = L.Proj.geoJson(map_be,
                        {weight:0.5,
                         color:"#fff",
                         fill: false,
                         pane: 'beadm2'}
                        ).addTo(map);

buildPointsLayer();
""" Add map event for responsiveness """
map.on('popupclose', function(){
    resizeMap();
});
});
```

LOAD PV POINTS AND MANAGE PROJECTION SETTINGS WITH PROJ4

buildPointsLayer :

```
solpv_points = [];
$.each(summaryData.zpoints, function(key, point){

    myIconPv = L.divIcon({className: 'geopoint-icon'
        , html: '<a tooltip="'+ point.solptname + '" href="#" class="geo-link"'
        , iconAnchor: icon_anchor
    });

    myPopup = L.popup({className:"pv_popup"})
        .setContent(buildPopupContent(point));

    belCoords = proj4(belProjection)
        .inverse([ point.longitude, point.latitude]);

    L.marker([belCoords[1], belCoords[0]],{icon: myIconPv})
        .bindPopup(myPopup).openPopup().addTo(map);

});
```

ADD EACH HOURLY IMAGE AND THE DAILY SUMMARY AS A PANE

What are panes?

In Leaflet, map panes group layers together implicitly. This grouping allows web browsers to work with several layers at once in a more efficient way than working with layers individually.

*Map panes use the **z-index** CSS property to always show some layers on top of others. This is why, popups always show “on top” of other layers, markers always show on top of tile layers, etc.*

Add the solar image as background pane:

```
sl = [];
for (i = 0; i < (endHour - startHour); i++) {
    hr = startHour + i;
    sl[i] = {id: 'sl_' + hr,
              layer : addSolarLayer('map_hourly' + hr + '.png', 'sl_' + hr)};
}
```

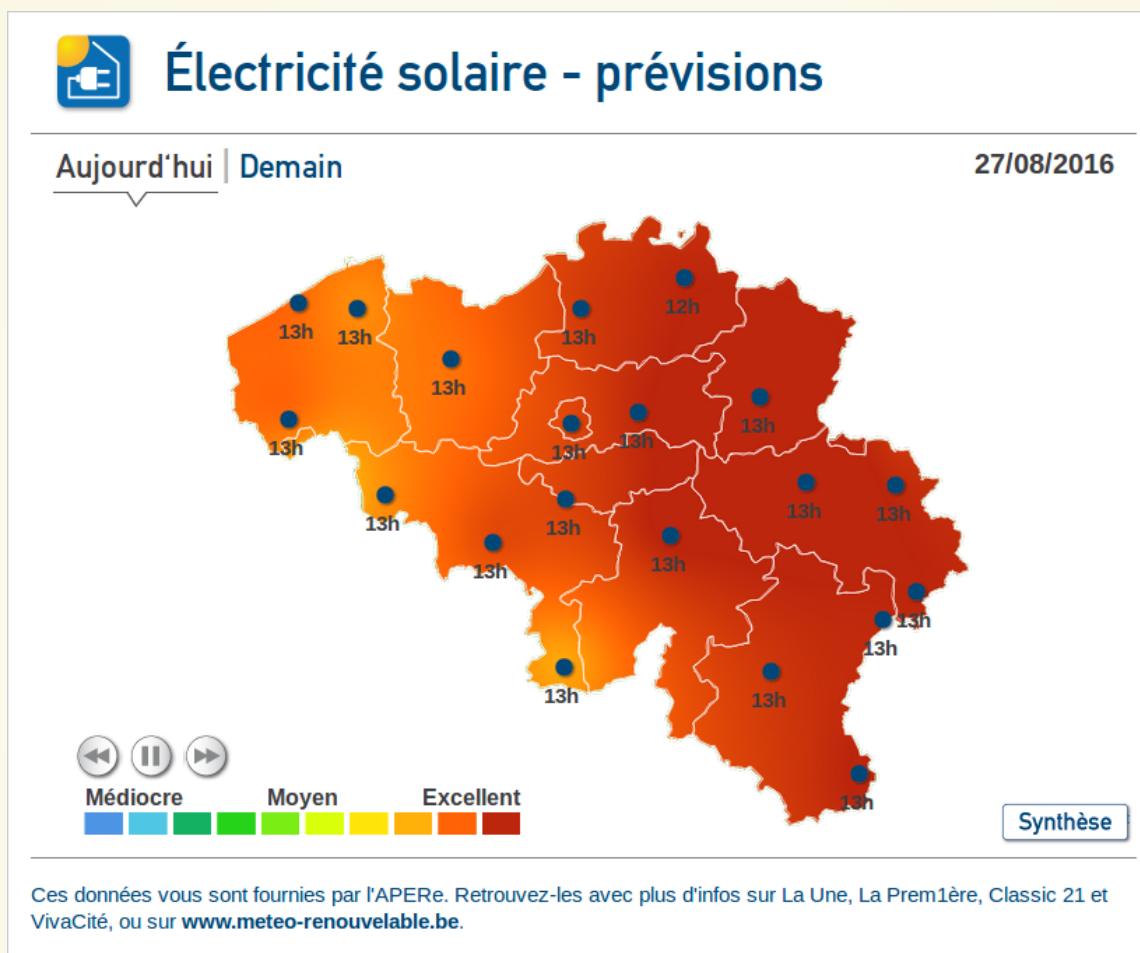
Add the daily synthesis map:

```
sl[i] = {id: 'sl_day',
          layer : addSolarLayer('map_daily.png', 'sl_day')};
```

Add pane functions:

```
addSolarLayer : function(imageUrl, id){
    var SL;
    addPane(id, 300);
    SL = L.Proj.imageOverlay(imageUrl,
                            bounds,
                            {pane: id}).addTo(map);
    return SL
},
addPane : function(id, startZi){
    map.createPane(id);
    map.getPane(id).style.zIndex = startZi;
},
```

NOW YOU CAN PLAY WITH THE Z-INDEX OF THE DIFFERENT PANES



#VaVersLeSoleil IFrame

@jphuart

jph@openjph.be

Thank you for your kind attention.

This presentation is available at

<https://github.com/jphuart/openjph-presentations/tree/master/FOSS4G-2016>

Thanks to reveal.js <https://github.com/hakimel/reveal.js> for the presentation framework.

Jean Pierre Huart - FOSS4G - 2016-09-22

