

Semi-automatic processing chain of VHR satellite imagery



Présentation
disponible sous
licence

MAUPP





grass gis
Bringing advanced geospatial technologies to the world

- SIG gratuit et open-source
- Développement permanent depuis 1980s.
- Fait de centaines de petits programmes
- La plupart des traitements géospatiaux
- Traitement image
 - Initialement pixel
 - De plus en plus de possibilités orientées objet depuis 2012
- Automatisable via Python

Automatisation dans GRASS

Niveau de complexité
croissant



- Ligne de commande
- Script Bash
- Modeleur graphique
- Script Python
- Module GRASS (interface graphique)
- Interaction avec d'autres logiciels (R)
- PyGRASS / Language C

Automatisation dans GRASS

→ INFO

- Chaque module est autonome et peut être vu comme une fonction
- Interface graphique ou ligne de commande

g.region [général, paramètres, computational region, emprise, resolution, level1] — + ×

Manages the boundary definitions for the geographic region.

Existant ☐ Définir à partir de la région par défaut (d)

Limites ☐ Save as default region (s)

Résolution Définir la région actuelle à partir de la région spécifiée: (region=name)

Effets [multiple] Set region to match raster map(s): (raster=name)

Afficher elevation

Optionnel Set region to match 3D raster map(s) (both 2D and 3D values): (raster_3d=name)

Messages de la commande

Manuel [multiple] Set region to match vector map(s): (vector=name)

Fermer Exécuter Copier Aide

☐ Fermer la boîte de dialogue après exécution

g.region -p raster=elevation

Automatisation dans GRASS

→ INFO

- Chaque module est autonome et peut être vu comme une fonction
- Interface graphique ou ligne de commande

g.region [général, paramètres, computational region, emprise, resolution, level1]

Manages the boundary definitions for the geographic region.

Existant ☐ Définir à partir de la région par défaut (d)

Limites ☐ Save as default region (s)

Résolution Définir la région actuelle à partir de la région spécifiée: (region=name)

Effets [multiple] Set region to match raster map(s): (raster=name)

Afficher elevation

Optionnel Set region to match 3D raster map(s) (both 2D and 3D values): (raster_3d=name)

Messages de la commande

Manuel [multiple] Set region to match vector map(s): (vector=name)

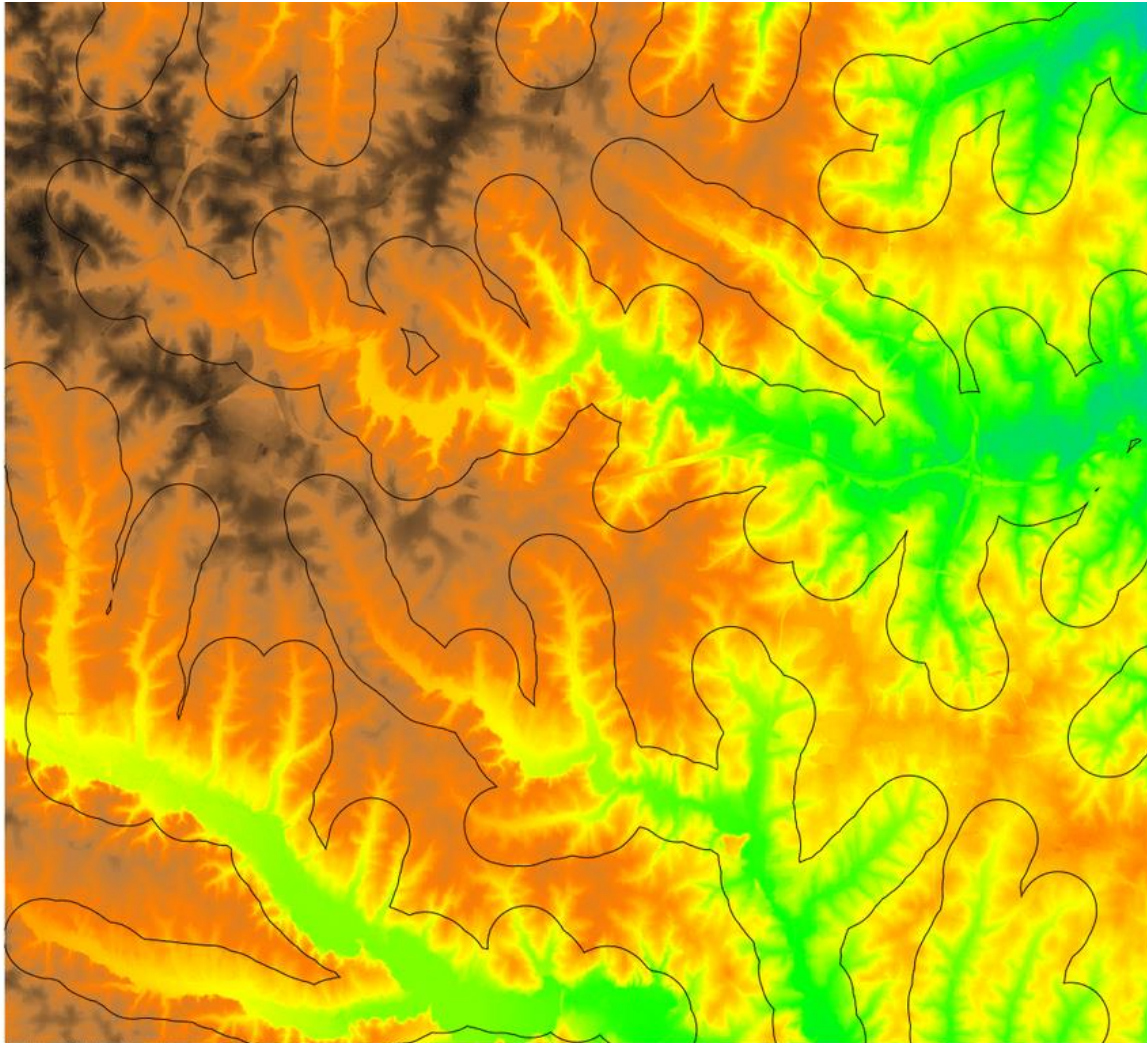
Fermer Exécuter Copier Aide

☐ Fermer la boîte de dialogue après exécution

g.region -p raster=elevation

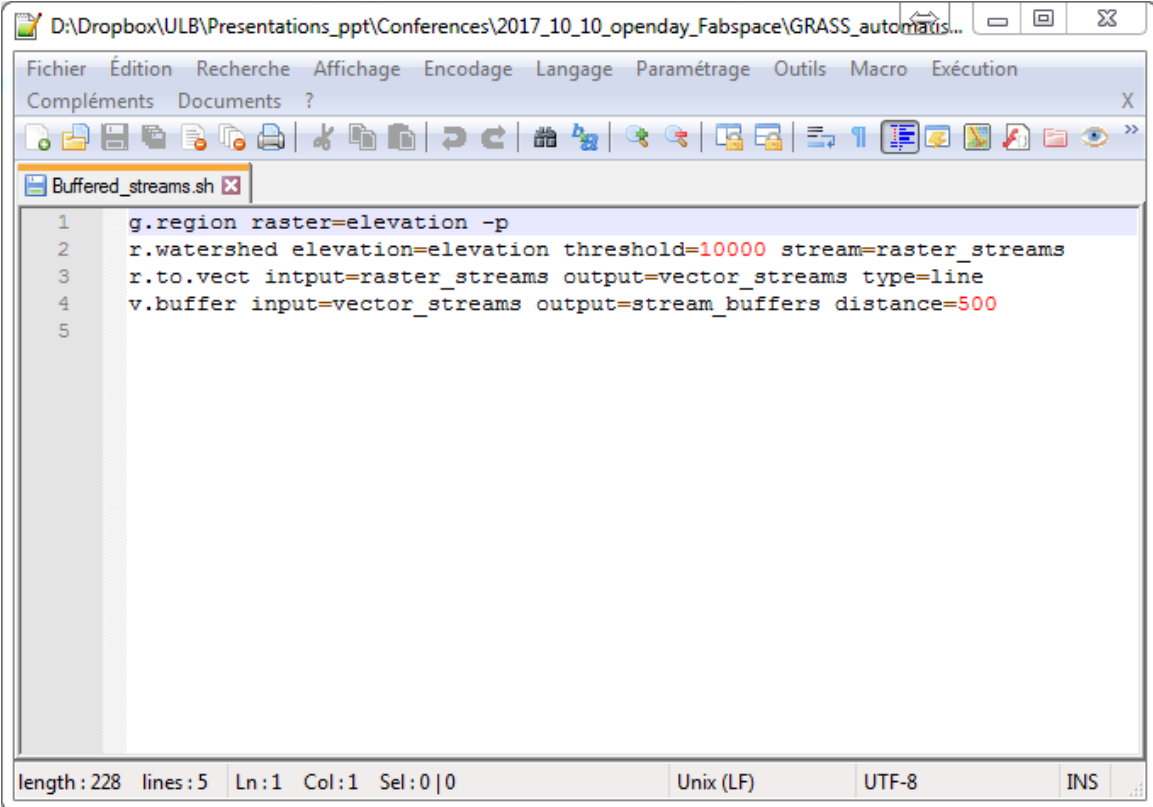
g.region -p raster=elevation

Exemple simple



Script bash

- Créer un fichier .sh avec une succession de commandes GRASS et le lancer depuis le terminal



The screenshot shows a text editor window titled "Buffered_streams.sh". The script contains the following commands:

```

1  g.region raster=elevation -p
2  r.watershed elevation=elevation threshold=10000 stream=raster_streams
3  r.to.vect input=raster_streams output=vector_streams type=line
4  v.buffer input=vector_streams output=stream_buffers distance=500
5

```

The status bar at the bottom indicates: length: 228 lines: 5 Ln: 1 Col: 1 Sel: 0 | 0 Unix (LF) UTF-8 INS

Script bash

- Peut déjà être considéré comme étant de l'automatisation
 - Peut être réappliqué à d'autres données juste en changeant le nom du raster
- De nombreux avantages comparés à l'utilisation via l'interface graphique:
 - Archive fidèle des commandes exécutées
 - Rendre le traitement facilement reproductible
 - Pouvoir le partager avec des collègues...

Modeleur graphique

→ INFO

The main Graphical Modeler menu contains options which enable the user to fully control the model. Directly under the main menu one can find toolbar with buttons (see figure below). There are options including (1) Create new model, (2) Load model from file, (3) Save current model to file, (4) Export model to image, (5) Export model to Python script, (6) Add command (GRASS modul) to model, (7) Add data to model, (8) Manually define relation between data and commands, (9) Add loop/series to model, (10) Add comment to model, (11) Redraw model canvas, (12) Validate model, (13) Run model, (14) Manage model variables, (15) Model settings, (16) Show manual, (17) Quit Graphical Modeler.

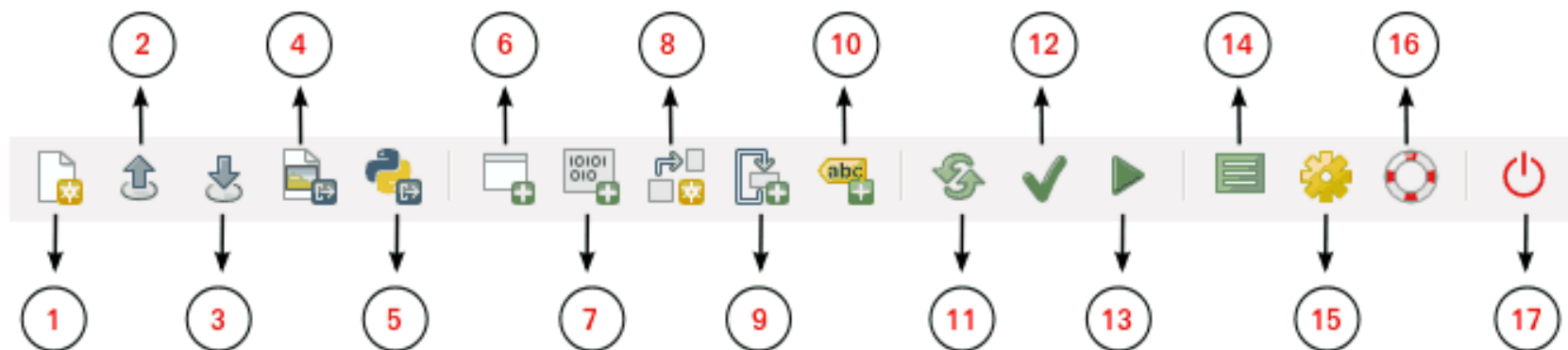
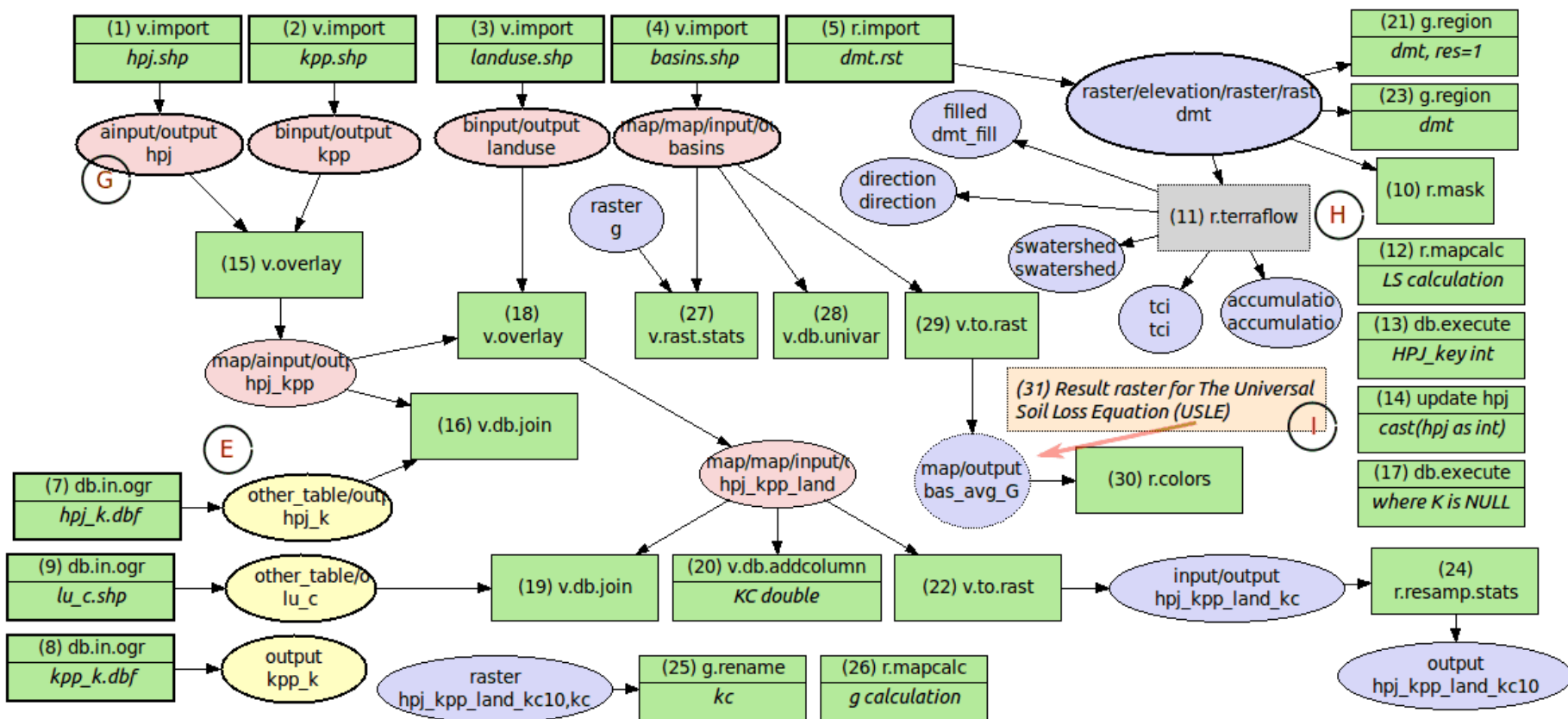


Figure: Components of Graphical Modeler menu toolbar.

Modeleur graphique

→ INFO



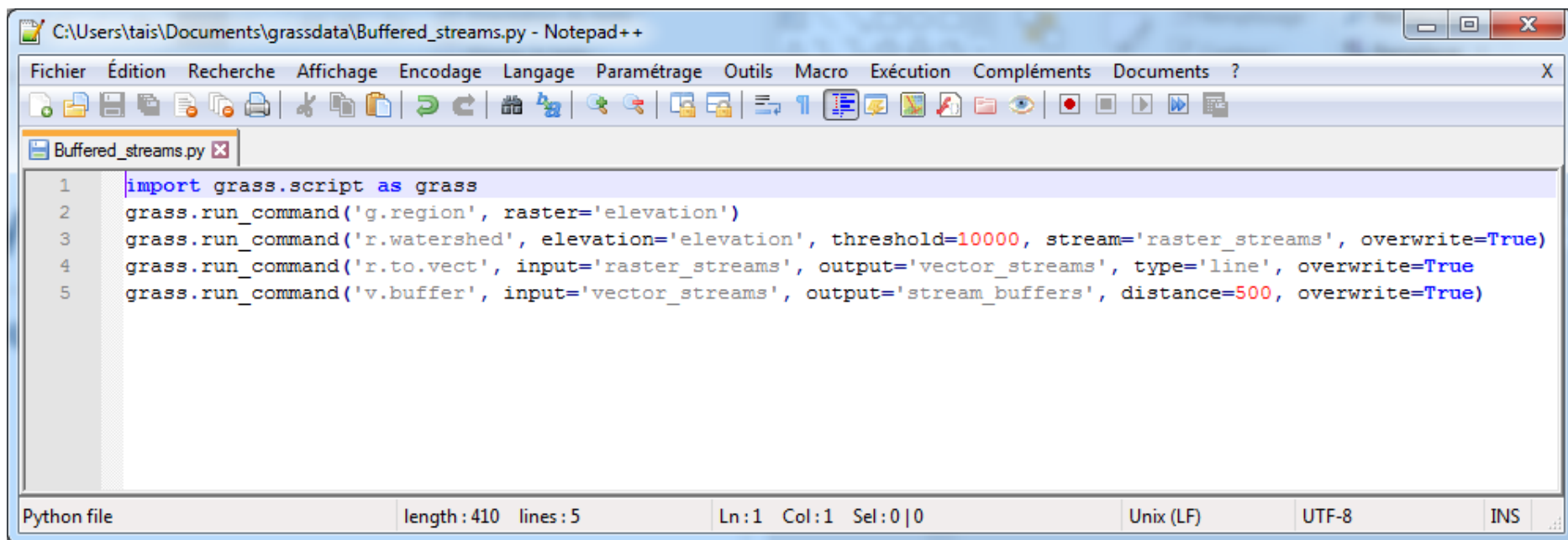
GRASS Python Scripting Library → [INFO](#)

- Librairie Python permettant d'appeler de façon simple n'importe quel module GRASS
- Point de départ parfait pour les débutants qui veulent simplement écrire un script sans pour autant entrer profondément dans le Python de GRASS

GRASS Python Scripting Library → *INFO*

- Permet de passer des paramètres et d'enregistrer l'output des modules GRASS comme des variables
- La puissance de Python:
 - Variables, listes, dictionnaires
 - Boucles, conditions
 - Des packages pour tout faire:
 - Pandas
 - NumPy
 - Matplotlib
 - Scikit-Learn
 - Scikit-Learn
 - Scikit-image
 - Rpy
 - Scipy

GRASS Python Scripting Library → [INFO](#)



C:\Users\tais\Documents\grassdata\Buffered_streams.py - Notepad++

Fichier Édition Recherche Affichage Encodage Langage Paramétrage Outils Macro Exécution Compléments Documents ?

Buffered_streams.py

```

1 import grass.script as grass
2 grass.run_command('g.region', raster='elevation')
3 grass.run_command('r.watershed', elevation='elevation', threshold=10000, stream='raster_streams', overwrite=True)
4 grass.run_command('r.to.vect', input='raster_streams', output='vector_streams', type='line', overwrite=True)
5 grass.run_command('v.buffer', input='vector_streams', output='stream_buffers', distance=500, overwrite=True)

```

Python file length: 410 lines: 5 Ln: 1 Col: 1 Sel: 0 | 0 Unix (LF) UTF-8 INS

GRASS Python Scripting Library

- Différents appels aux modules GRASS en fonction de l'output attendu → *INFO*

```
>>> # get the current region settings as a list of parameters one per line (can then be treated
with the Python splitlines() function
>>> grass.read_command('g.region', flags='g')
'n=228500\ns=215000\nw=630000\nne=645000\nnsres=10\nnewres=10\nrows=1350\ncols=1500\ncells=2025000\n'

>>> # get current region settings as a python dictionary:
>>> grass.parse_command('g.region', flags='g')
{'rows': '1350', 'e': '645000', 'cells': '2025000', 'cols': '1500', 'n': '228500', 's':
'215000', 'w': '630000', 'ewres': '10', 'nsres': '10'}
>>> # or using a dedicated function as a shortcut (which also directly transforms all numbers
from strings to numeric format)
>>> grass.region()
{'rows': 1350, 'e': 645000.0, 'cells': 2025000, 'cols': 1500, 'n': 228500.0, 's': 215000.0,
'w': 630000.0, 'ewres': 10.0, 'nsres': 10.0}
```

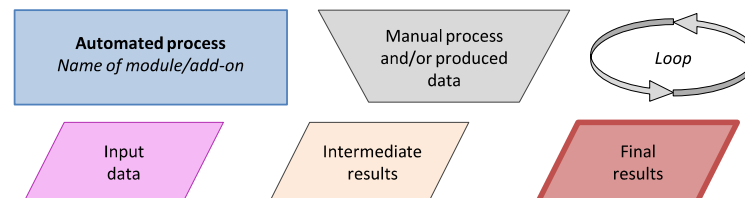
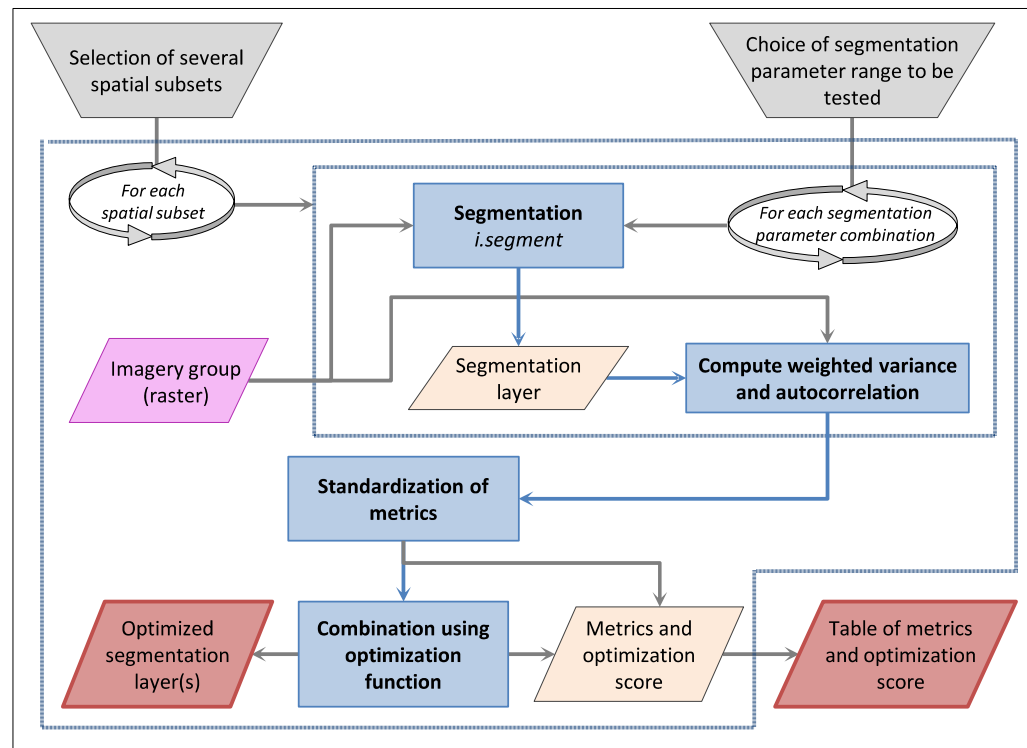
Traitement image orienté objet

- `i.segment`
– Segmentation (region growing / mean shift) → INFO
- `i.segment.uspo`
– Optimisation des paramètres de segmentation → INFO
- `i.segment.stats`
– Calcul des statistiques des segments → INFO
- `i.superpixels.slic`
– Segmentation en superpixels (SLIC et SLIC0) → INFO
- `v.class.mlr`
– Machine Learning dans R (classification supervisée) → INFO

Exemple de création d'un module GRASS

→ INFO

Unsupervised
segmentation
parameter
optimization
=> i.segment.uspo



Exemple de création d'un module GRASS



Projection: WGS 1984 / UTM zone 30N (EPSG: 32630)
© DigitalGlobe, Inc. All Rights Reserved

0 5 10 m

Exemple de création d'un module GRASS



Projection: WGS 1984 / UTM zone 30N (EPSG: 32630)

© DigitalGlobe, Inc. All Rights Reserved

0 5 10 m

Exemple de création d'un module GRASS



Projection: WGS 1984 / UTM zone 30N (EPSG: 32630)

© DigitalGlobe, Inc. All Rights Reserved

0 5 10 m

Exemple de création d'un module GRASS



Projection: WGS 1984 / UTM zone 30N (EPSG: 32630)

© DigitalGlobe, Inc. All Rights Reserved

0 5 10 m

Exemple de création d'un module GRASS

→ INFO

Script Python

```

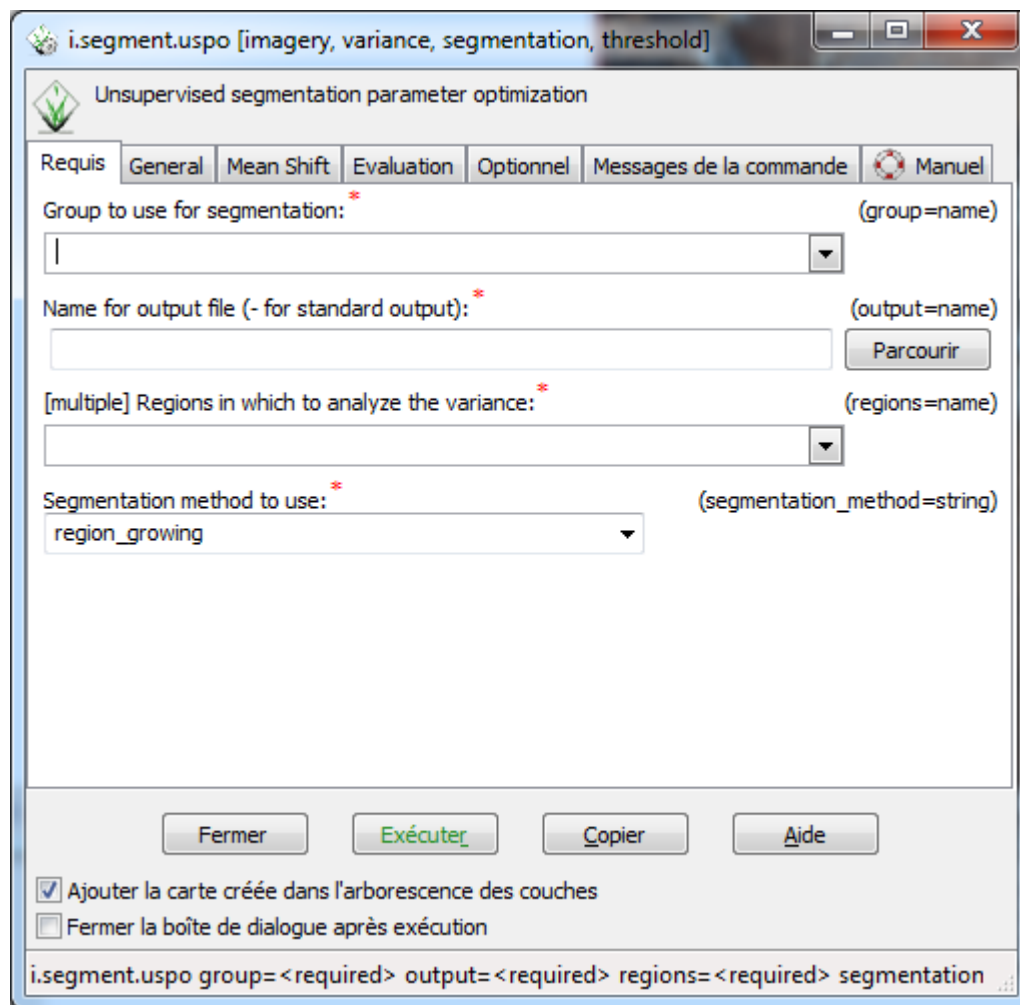
689 if options['thresholds']:
690     thresholds = [float(x) for x in options['thresholds'].split(',')]
691 else:
692     step = float(options['threshold_step'])
693     start = float(options['threshold_start'])
694     stop = float(options['threshold_stop'])
695     iter_thresh = drange(start, stop, step)
696     # We want to keep a specific precision, so we go through string
697     # representation and back to float
698     thresholds = [float(y) for y in ["%.4f" % x for x in iter_thresh]]
699
700 if options['minsizes']:
701     minsizes = [int(x) for x in options['minsizes'].split(',')]
702 else:
703     step = int(options['minsize_step'])
704     start = int(options['minsize_start'])
705     stop = int(options['minsize_stop'])
706     minsizes = range(start, stop, step)

```

Exemple de création d'un module GRASS

→ INFO

Interface
graphique =>



Exemple de création d'un module GRASS

→ INFO

- Pas besoin d'être informaticien
- Interface graphique créée automatiquement via les commentaires du script

```

32  %%Module
33  %% description: CREER UN MODULE GRASS C EST FACILE
34  %% keyword: imagery
35  %% keyword: variance
36  %% keyword: segmentation
37  %% keyword: threshold
38  %%end
39  #
40  %%option G_OPT_I_GROUP
41  %% description: Group to use for segmentation
42  %% required : yes
43  %%end
44  #
45  %%option G_OPT_R_MAPS
46  %% key: maps
47  %% description: Raster band(s) for  which to calculate variance (default: all group members)
48  %% required : no
49  %%end

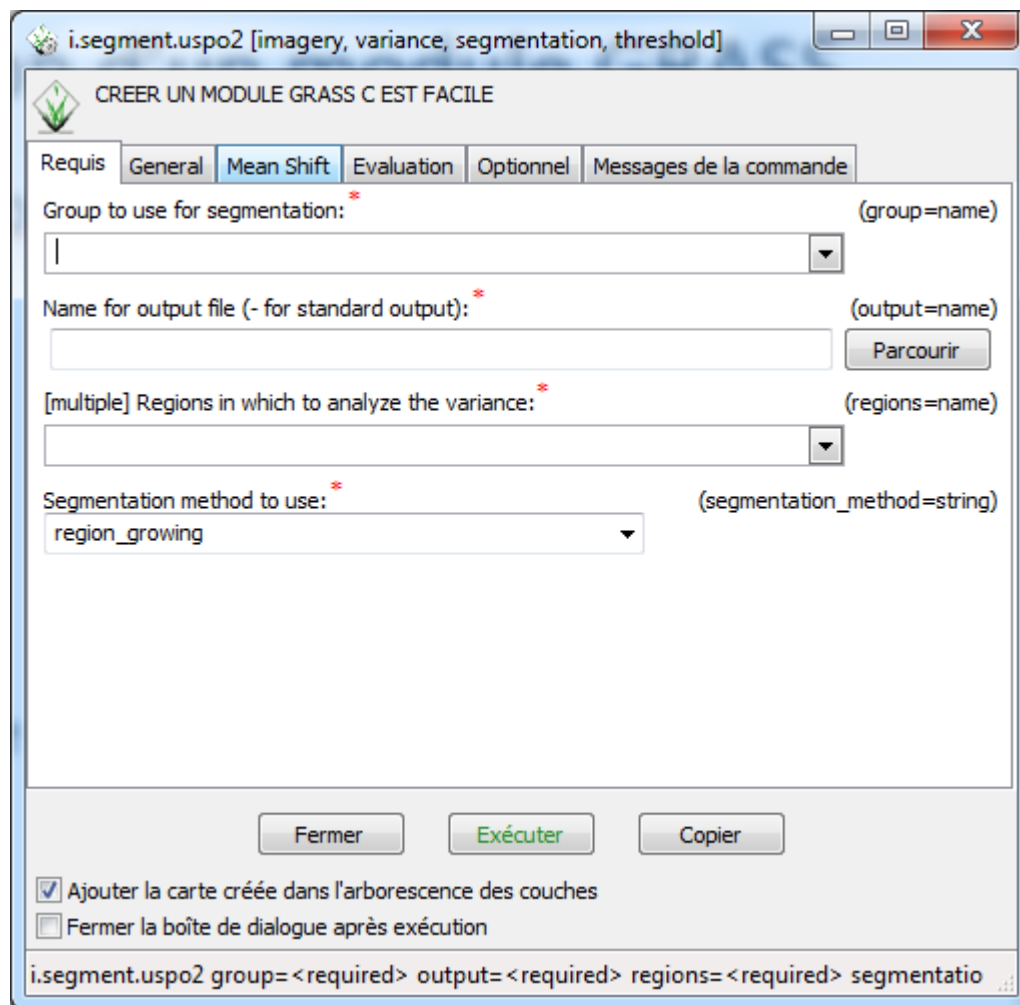
```

→ INFO

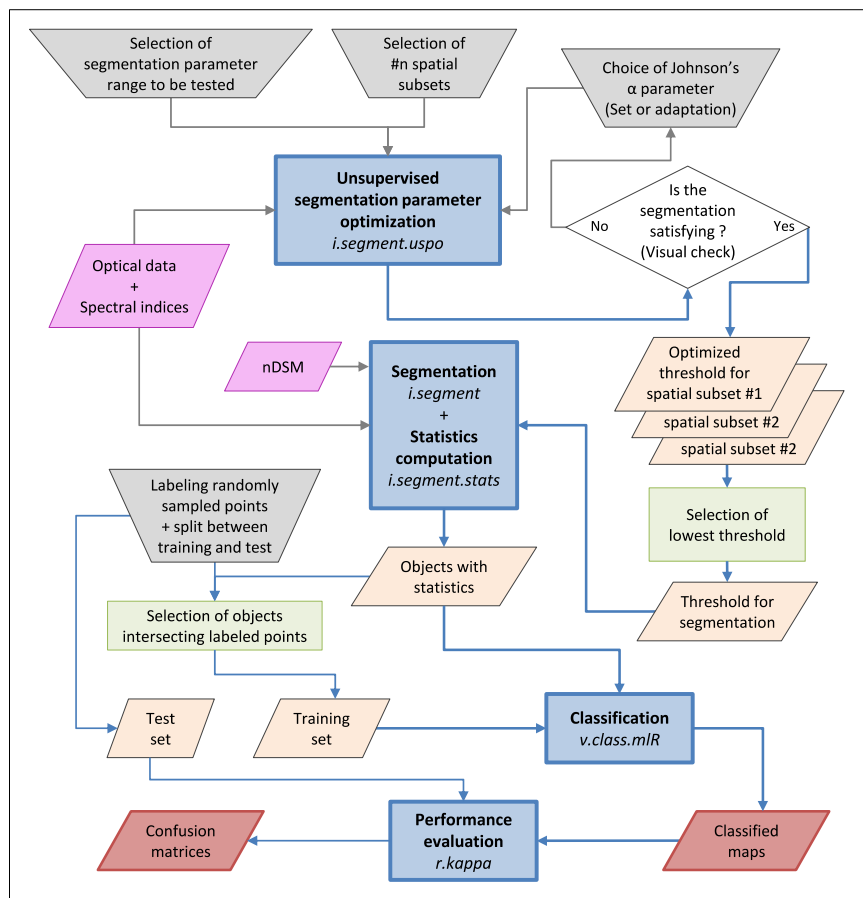
Exemple de création d'un module GRASS

→ INFO

Interface
graphique =>



Développement d'une solution semi-automatisée



grassgis
Bringing advanced geospatial technologies to the world

```
In [ ]: ## Define computational region to match the extent of segmentation raster
grass.run_command('g.region', overwrite=True, raster="segments@CLASSIFICATION")

## Saving current time for processing time management
print ("Start computing statistics for training segments, using i.segment.stats on " + time.ctime())
begin_time_i_segment_stats=time.time()

## Compute statistics of objets using i.segment.stats only with .csv output (no vectormap output)
grass.run_command('i.segment.stats', overwrite=True, map="segments_training@CLASSIFICATION",
rasters=inputsstats,
raster_statistics="min,max,range,mean,stddev,sum,coeff_var,first_quart,median,third_
quart,perc_90",
area_measures="area,perimeter,compact_circle",
csvfile="F:\\.....\\Classification\\i.segment.stats\\stats_training_sample.csv")

## Compute processing time and print it
print_processing_time(begin_time_i_segment_stats, "Segment statistics computed in :")
```



GitHub



remote sensing



Article

An Open-Source Semi-Automated Processing Chain for Urban Object-Based Classification

Taïs Grippa^{1,*}, Moritz Lennert¹, Benjamin Beaumont^{1,2}, Sabine Vanhuyse¹,
Nathalie Stephenne² and Eléonore Wolff¹

Classification orientée objet



Projection: WGS 1984 / UTM zone 30N (EPSG: 32630) ©
DigitalGlobe, Inc. All Rights Reserved

0 20 40 m

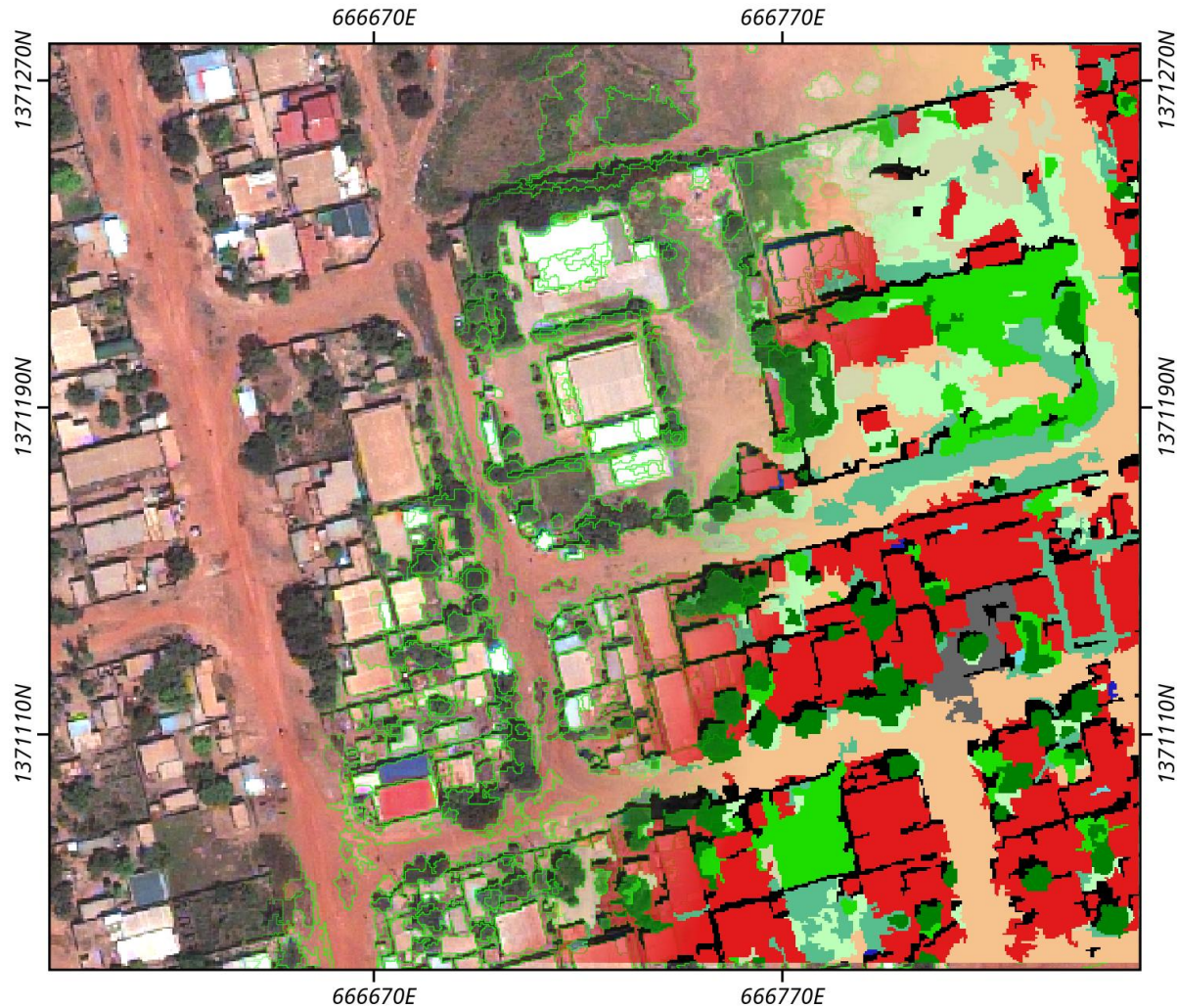
Classification orientée objet



Projection: WGS 1984 / UTM zone 30N (EPSG: 32630) ©
DigitalGlobe, Inc. All Rights Reserved

0 20 40 m

Classification orientée objet



Projection: WGS 1984 / UTM zone 30N (EPSG: 32630) ©
DigitalGlobe, Inc. All Rights Reserved

0 20 40 m

Thank you for your attention

Any questions ?

Contact: tgrippa@ulb.ac.be

MAUPP project website: <http://maupp.ulb.ac.be/>

Processing chain available at: <https://github.com/tgrippa>

Présentation disponible sous licence CC-BY-SA. © Taïs Grippa

