České vysoké učení technické v Praze
Fakulta stavební

FREE SOFTWARE GIS

Semestrální projekt

Testy GRASS modulů

Skupina C

Lucie Děkanová, Josef Pudil

Letní semestr 2019/2020

Obsah

1. Úvod

1.1 Zadání

Zadáním bylo vytvořit testy GRASS modulů pro funkce *v.random, r.stats, r.region, r.resample, r.surf.random, r.buffer*.

1.2 Cíl projektu

Cílem projektu bylo vytvořit testy, které otestují správnou funkčnost GRASS modulů v případě jakékoliv změny.

2. Použitý software

Pro vytváření testů byl využit Simple Python Editor přímo v programu GRASS GIS 7.9.dev, tedy ve vývojové verzi GRASS. Tento editor umožnil vytvoření testů v jazyku Python.

3. Data

Pro testování byla využita volně dostupná data GRASS GIS Sample Data. Jedná se o dataset, který je balíčkem obsahujícím geoprostorová data ze Severní Karolíny. Nabízí rastrová, vektorová, LiDAR a družicová data. Balíček je primárně určen pro testování modulů v GRASS GIS.

Pro účel této semestrální práce byla využita data rastrová a vektorová.

4. Popis testovaných modulů

V této kapitole jsou stručně popsány moduly, ke kterým byly vytvořeny testy.

4.1 Modul v.random

V.random generuje náhodně body ve vektorovém formátu vně vybraného regionu pro vybraný počet náhodných bodů.

U *v.random* můžeme k vymezení regionu využít i vektorovou vrstvu, kterou můžeme definovat pomocí parametru modulu *restrict*.

V.random může také generovat 3D body nebo napsat náhodnou hodnotu do atributové tabulky. Rozsah výšky bodů nebo rozsah atributové hodnoty můžeme kontrolovat pomocí definování minimální a maximální hodnoty výšky (zmin, zmax).[1]

4.2 Modul r.surf.random

R.surf.random vytváří rastrovou mapovou vrstvu vně námi zvoleného rozmezí. Tato funkce používá lineární číselný generátor. [2]

4.3 Modul r.resample

Tento modul převzorkuje hodnoty dat podle rastru, který specifikuje uživatel. Rastr musí být ohraničený aktuálním výpočetním regionem. Tato funkce vytváří nový rastrový výstup, který obsahuje výsledky převzorkování.

Kategorie hodnot v nové rastrové mapě budou stejné jako v originální mapě, kromě rozlišení a rozsahu nového výstupu, které budou odpovídat těm z aktuálního nastavení regionu (*g.region*).

Modul je určený k převzorkování diskrétních dat rastru (jako je geologie, typ povrchu) do jiného rozlišení. [3]

4.4 Modul r.buffer

Tento modu vytváří novou rastrovou mapu s *„buffer“* zónami kolem buněk které obsahují nenulové hodnoty v existující rastrové mapě. Vzdálenosti zón jsou definovány uživatelem a musí mít kladnou hodnotu. [4]

4.5 Modul r.stats

R.stats počítá oblast v každé z kategorií nebo intervalech v zvolené vstupní rastrové mapě. Statistika oblasti je dána v jednotkách metrů čtverečních nebo počtem buněk. Tato analýza používá aktuální region (*r.region*) a nastavení masky *(r.mask*). Výstupní statistika může být uložena do výstupního souboru.

Pokud chceme statistiku oblasti v metrech čtverečních, využijeme *„flag“* -a a pro statistiku v počtu buněk využijeme *„flag“* -c. [5]

4.6 Modul r.region

Tento modul umožňuje uživateli určit meze rastrové mapy. Tyto meze můžou být vymezeny uživatelem přímo nebo pomocí rastrové nebo vektorové mapy.[6]

5. Testy modulů

Tato kapitola se zabývá popsáním a ukázkami jednotlivých testů.

Základem testů GRASS modulů je balíček gunittest, ze kterého je třeba importovat *TestCase* a *Test* pro vytvoření testů. *TestCase* je třída, ve které jsou obsaženy metody testování (*test methods*). *Test* slouží k spuštění testu.

Dále jsou ve všech funkcích použity příkazy *SetUpClass* a *TearDownClass*. Příkaz *SetUpClass* slouží k přípravě testu a *TearDownClass* k vymazání po proběhnutí testu.

Metody končící „*Class*“ můžou být definované využitím *@classmethod* a *cls* je použito jako první argument. Tyto metody jsou prováděny jednou pro celou třídu, zatímco metody bez „*Class*“ jsou prováděny pro každou testovací metodu.

Každý testovací soubor by měl být schopen běžet sám a přijímat určitou sadu parametrů příkazového řádku. To se provádí pomocí funkce *if __name__ == '__main__' a test ()*.

6. Ukázky testů

6.1 Test v.random

```python
#!/usr/bin/env python3

from grass.gunittest.case import TestCase
from grass.gunittest.main import test

output = 'test01'
output2 = 'test02'
npoints = 100
state = 'boundary_state'
zmin=10
zmax=120

class Test_v_random(TestCase):

    @classmethod
    def setUpClass(cls):
        cls.use_temp_region()
        cls.runModule('g.region', vector=state)

    @classmethod
    def tearDownClass(cls):
        cls.del_temp_region()


    #Checking if number of points equals 100
    def test_num_points(self):
        self.assertModule('v.random', output=output, npoints=npoints,
                          overwrite=True)

        topology = dict(points=npoints)
        self.assertVectorFitsTopoInfo(vector=output, reference=topology)
```

```python

    #Checking if the map is 3D and number of points is 100
    def test_num_points_3D(self):
        self.assertModule('v.random', output=output, npoints=npoints,
                          zmin=zmin, zmax=zmax,
                          overwrite=True, flags='z')

        topology = dict(points=npoints, map3d=1)
        self.assertVectorFitsTopoInfo(vector=output, reference=topology)

    #Checking if all points are in the polygon boundary state
    def test_restrict(self):
        self.assertModule('v.random', output=output, npoints=npoints,
                          restrict=state, overwrite=True)
        self.assertModule('v.clip', input=output, clip=state,
                          overwrite=True, output=output2)

        self.assertVectorInfoEqualsVectorInfo(output,output2,precision=0.01)


if __name__ == '__main__':
    test()
```

6.2 Test r.surf.random

```python
#!/usr/bin/env python3

from grass.gunittest.case import TestCase
from grass.gunittest.main import test

output = 'test01'
n=100
s=-100
w=-100
e=100
res=1
min=10
max=40

class Test_v_surf_random(TestCase):

# setting up computational region
    @classmethod
    def setUpClass(cls):
        cls.use_temp_region()
        cls.runModule('g.region', n=n, s=s, e=e, w=w, res=res)

    @classmethod
    def tearDownClass(cls):
        cls.del_temp_region()

# testing if output parameters are between input parameters
    def test_min_max(self):
         self.assertModule('r.surf.random', output=output, min=min, max=max, overwrite=True)
         self.assertRasterMinMax(map=output, refmin=min, refmax=max)

if __name__ == '__main__':
    test()
```

6.3 Test r.resample

```python
#!/usr/bin/env python3

from grass.gunittest.case import TestCase
from grass.gunittest.main import test

output = 'test01'
input = 'basins'
resolution = 20
north = 228500
south = 215000
east = 645000
west = 630000
min=2
max=30


class Test_r_resample(TestCase):

    @classmethod

    def setUpClass(cls):
        cls.use_temp_region()

        # change of resolution from 10 to 20
        cls.runModule('g.region', raster=input, res=resolution)

    @classmethod
    def tearDownClass(cls):
        cls.del_temp_region()

    def test_check_resample(self):
        # running r.resample
        self.assertModule('r.resample', input=input, output=output, overwrite=True)
```

```python

        # testing resolution
        ref_res="nsres=20\newres=20"
        self.assertRasterFitsInfo(raster=output, reference=ref_res)

        # testing coordinates
        ref_extent="north=228500\nsouth=215000\neast=645000\nwest=630000"
        self.assertRasterFitsInfo(raster=output, reference=ref_extent)

        # testing data values
        ref_values="min=2\nmax=30"
        self.assertRasterFitsInfo(raster=output, reference=ref_values)

if __name__ == '__main__':
    test()

```

6.4 Test r.buffer

```python
#!/usr/bin/env python3
from grass.gunittest.case import TestCase
from grass.gunittest.main import test
from grass.gunittest.gmodules import call_module
import grass.script as gscript

output = 'test05'
buffer = 'basins'
distance1=100
distance2=200
ref_stats = '1 1554947\n2 58014\n* 412039'

class Test_r_buffer(TestCase):
    @classmethod
    def setUpClass(cls):
        cls.use_temp_region()
#         cls.runModule('g.region', raster=buffer)


    @classmethod
    def tearDownClass(cls):
        cls.del_temp_region()

#create buffer with distance 100 metres
    def test_check_distance(self):
        self.assertModule('r.buffer', output=output, input=buffer, overwrite=True, distances=distance1)
        # self.assertRastersDifference(actual='buffer', reference='output', precision=0.5, statistics=None)

        stats = call_module('r.stats', input=output, flags='c').rstrip('\n')
        self.assertEqual(stats, ref_stats)

if __name__ == '__main__':
    test()
```

6.5 Test r.stats

```python
#!/usr/bin/env python3

from grass.gunittest.case import TestCase
from grass.gunittest.main import test
from grass.gunittest.gmodules import call_module

input = 'lakes'
#count of pixels with different ID
ref_stats = '34300 442\n39000 35099\n43600 470\n* 1988989'
#area of pixels with different ID

ref_stats2 = '34300 44200.000000\n39000 3509900.000000\n43600 47000.000000'

class Test_r_stats(TestCase):

    @classmethod
    def setUpClass(cls):
        cls.use_temp_region()
        cls.runModule('g.region', raster = input)

    @classmethod
    def tearDownClass(cls):
        cls.del_temp_region()

    # Testing total area
    def test_number_of_pixels(self):
        stats = call_module('r.stats', input=input, flags='c').rstrip('\n')
        self.assertEqual(stats, ref_stats)

    def test_area(self):
        stats = call_module('r.stats', input=input, flags='aN').rstrip('\n')
        self.assertEqual(stats, ref_stats2)


if __name__ == '__main__':
    test()
```

6.6 Test r.region

```
#!/usr/bin/env python3

import ...

output = 'test03'
input = 'basins'
north = 228600
south = 215100
ref_extent="north=228600\nsouth=215100\neast=645000\nwest=630000"

class Test_r_region(TestCase):

    @classmethod
    def setUpClass(cls):
        cls.use_temp_region()
        cls.runModule('g.region', raster=input)

    @classmethod
    def tearDownClass(cls):
        cls.del_temp_region()

    def test_r_region(self):
        self.assertModule('g.copy', raster=(input, output), overwrite=True)
        self.assertModule('r.region', map=output, n=north, s=south)
        self.assertRasterFitsInfo(raster=output, reference=ref_extent)

if __name__ == '__main__':
    test()

```

Závěr

Cílem projektu bylo vytvořit testy GRASS GIS moduly. Celkem byly vytvořeny testy pro 6 modulů. V tvorbě testů by se dalo pokračovat i pro další moduly, ovšem z časových důvodů nebylo možné pokračovat jejich další tvorbou.

Testy jsou připraveny k reálnému využití v GRASS GIS.

Zdroje

[1] v.random: https://grass.osgeo.org/grass78/manuals/v.random.html

[2]r.surf.random: https://grass.osgeo.org/grass78/manuals/r.surf.random.html

[3]r.resample: https://grass.osgeo.org/grass78/manuals/r.resample.html

[4]r.buffer: https://grass.osgeo.org/grass78/manuals/r.buffer.html

[5]r.stats: https://grass.osgeo.org/grass78/manuals/r.stats.html

[6]r.region: https://grass.osgeo.org/grass78/manuals/r.region.html