

Подсчёт уникальных значений пикселей в растре при помощи SEXTANTE и QGIS

[Обсудить в форуме](#) Комментариев — 6

Эта страница опубликована в основном списке статей сайта по адресу <http://gis-lab.info/qa/qgis-sextante-raster-unique-values-count.html>

В данной статье описан процесс подсчёта уникальных значений пикселей растра при помощи скрипта, интегрируемого в Processing - модуль геопроессинга QGIS.

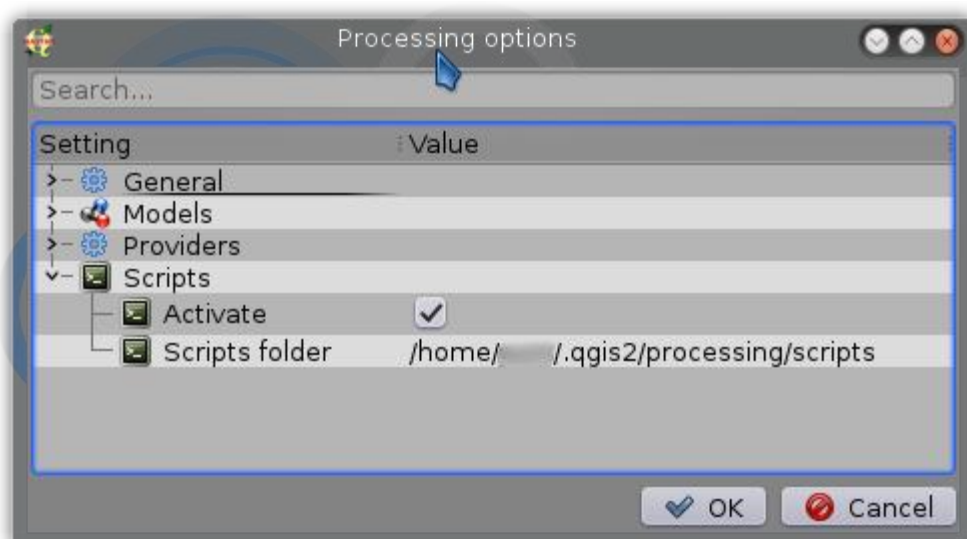
[via Misanthrope's Thoughts](#)

Содержание

- [1 Установка скрипта](#)
- [2 Работа со скриптом](#)
- [3 Как это работает](#)
- [4 Вместо заключения](#)
- [5 Ссылки по теме](#)

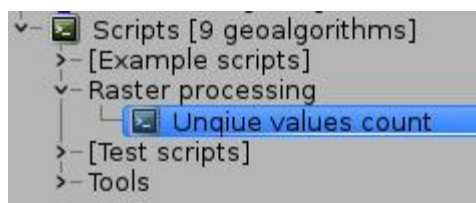
Установка скрипта

[Загрузите архив](#) со скриптом и help-файлом. Распакуйте архив в директорию, предназначенную для Python-скриптов SEXTANTE в QGIS (например, ~/.qgis2/processing/scripts, если вы используете Linux). Если вы не знаете, где находится нужная папка перейдите в Processing -> Options and configuration -> Scripts -> Scripts folder см. скриншот:

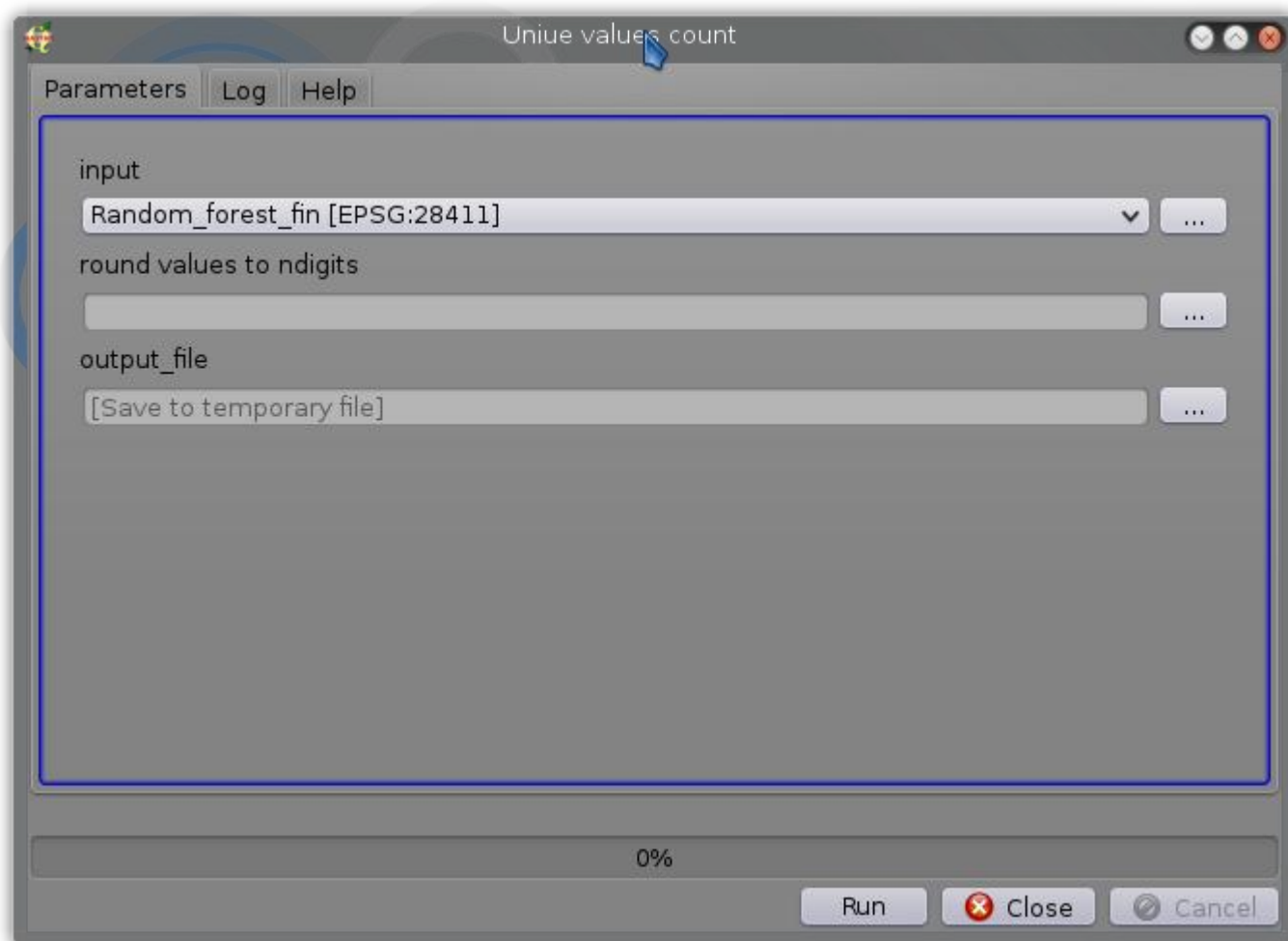


Работа со скриптом

Перезапустите QGIS, если она была запущена. Откройте Processing Toolbox. Во вкладке Scripts вы увидите новый раздел Raster processing, а в нём - скрипт Unique values count:



Запустив его, вы увидите диалоговое окно:

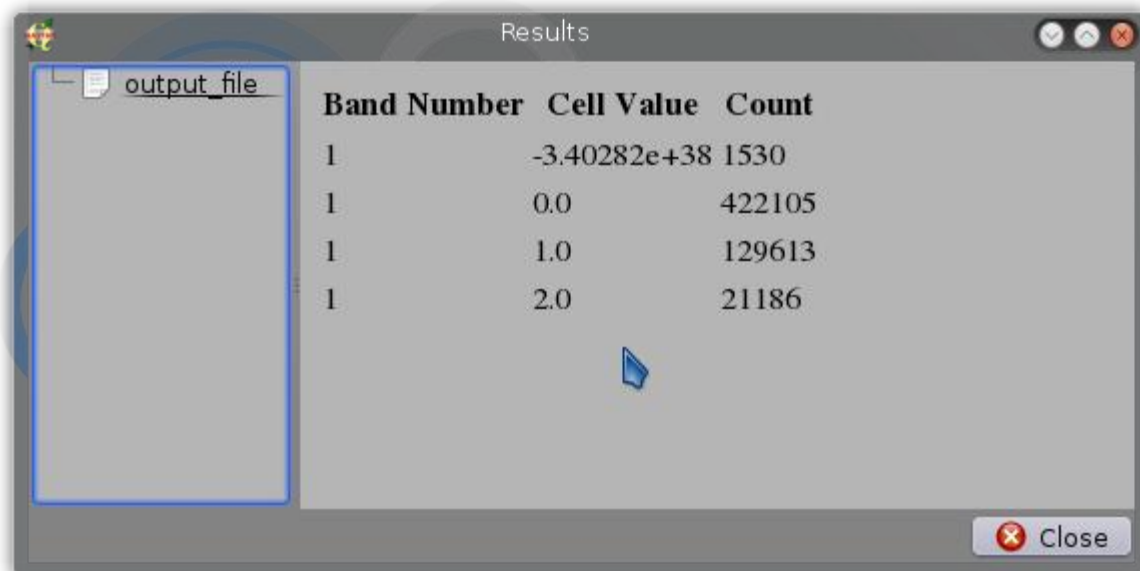


Обратите внимание, что во вкладке Help находится описание скрипта (на английском языке).

В поле 'input' укажите растр, уникальные значения пикселей которого надо посчитать. Подходят как однослойные, так и многослойные растры.

После того, как растр выбран, следует определиться с тем, необходимо ли округлять значения пикселей перед подсчётом. Если нет - оставляете поле 'round values to ndigits' пустым. Если же округление необходимо - вставьте в это поле номер знака после запятой до которого следует округлять значения. Обратите внимание, что значения можно делать отрицательными: в этом случае округление будет проводится до указанного знака ПЕРЕД запятой.

Когда все необходимые поля заполнены можно нажимать кнопку 'Run'. По окончании работы скрипта появится окно с результатами подсчётов в виде таблицы:



| Band Number | Cell Value | Count |
|-------------|--------------|--------|
| 1 | -3.40282e+38 | 1530 |
| 1 | 0.0 | 422105 |
| 1 | 1.0 | 129613 |
| 1 | 2.0 | 21186 |

В графе **Band Number** указывается номер слоя раstra для которого подсчитано количество пикселей с данным значением. В графе **Cell Value** указывается уникальное значение пикселя. В графе **Count** - количество пикселей с данным значением в данном слое раstra.

Как это работает

Ниже представлен исходный код скрипта с комментариями. Обратите внимание, что в скриптах, использующихся в SEXTANTE двойная решётка '##' предназначена для задания специальных параметров ввода данных и вывода результата.

```
##Raster processing=group                # задаём имя группы скриптов SEXTANTE, где будет
находиться скрипт                        # задаём поле выбора раstra
##input=raster                           # задаём поле ввода значения для округления
##round_values_to_ndigits=string          # задаём вывод результатов в виде html-файла (путь
##output_file=output html                к файлу)

# импортируем необходимые библиотеки
from osgeo import gdal
import sys
import math

# загружаем растр
gdalData = gdal.Open(str(input))

# получаем размеры раstra в пикселях
xsize = gdalData.RasterXSize
ysize = gdalData.RasterYSize

# получаем количество слоёв раstra
bands = gdalData.RasterCount

# начинаем запись результирующего html-файла
f = open(output_file, 'a')
f.write('<TABLE>\n<TH>Band Number</TH> <TH>Cell Value</TH> <TH>Count</TH>\n')

# подсчёт пикселей и запись результата
for i in xrange(1, bands + 1):
    band_i = gdalData.GetRasterBand(i)
    raster = band_i.ReadAsArray()

    # создаём словарь в который будем записывать результат
    count = {}
```

```

# подсчитываем уникальные значения пикселей для данного слоя
for col in range(xsize):
    for row in range(ysize):
        cell_value = raster[row, col]

        # проверяем, не соответствует ли cell_value тип NaN
        if math.isnan(cell_value):
            cell_value = 'Null' # если да, то считаем этот пиксель как 'Null'

        # округляем значения пикселя если это необходимо
        elif round_values_to_ndigits:
            try:
                cell_value = round(cell_value, int(round_values_to_ndigits))
            except:
                # если в поле 'round_values_to_ndigits' было введено нечто непонятное,
                # будем считать, что пользователь хотел округлить до целых чисел
                cell_value = round(cell_value)

        # наконец, добавляем / обновляем счётчик для данного значения пикселя
        try:
            count[cell_value] += 1
        except:
            count[cell_value] = 1

# записываем отсортированные результаты обработки данного слоя раstra в таблицу
for key in sorted(count.iterkeys()):
    line = "<TD>%s</TD> <TD>%s</TD> <TD>%s</TD>" %(i, key, count[key])
    f.write('<TR>'+ line + '</TR>' + '\n')

# заканчиваем запись таблицы и закрываем файл
f.write('</TABLE>')
f.close

```

Вместо заключения

Пожелания и предложения по работе скрипта можно оставлять в соответствующей теме форума.

Ссылки по теме

- [Работа с растрами при помощи GDAL и Python](#)
- [Геопроессинг с SEXTANTE для QGIS](#)

[Обсудить в форуме](#) Комментариев — 6

Последнее обновление: 2014-05-15 01:56

Дата создания:

Автор(ы): [SS_Rebelious](#)