# Find Neighbor Polygons in a Layer

QGIS Tutorials and Tips



# Author Ujaval Gandhi

http://google.com/+UjavalGandhi

Translations by

Marina Pavlova Ilya Trofimov Fayçal Fatihi

# Нахождение соседних полигонов в слое

Существует ряд случаев, когда вы хотите, найти все соседние полигоны для каждого полигона в слое. С помощью небольшого сценария Python мы можем сделать в QGIS это и многое другое. Вот пример сценария, который вы можете использовать, чтобы найти все полигоны, которые имеют общую границу с каждым из полигонов в слое, а также добавить их названия в таблицу атрибутов. В качестве дополнительного бонуса, сценарий также суммирует атрибут на ваш выбор по всем соседним полигонам.

#### Обзор задачи

Чтобы продемонстрировать, как работает скрипт, мы будем использовать слой полигонов стран и найдем страны, которые имеют общие границы. Мы также хотим вычислить общее население соседних стран для каждой страны.

### Получение данных

Мы будем использовать набор данных Admin 0 - Countries из Natural Earth.

Загрузите shape-файл Admin 0 - countries...

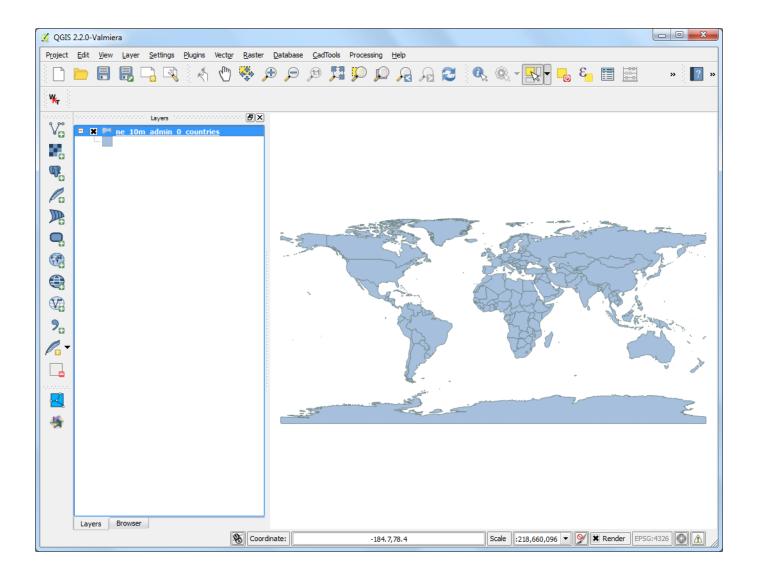
Источник данных: [NATURALEARTH]

## Получение сценария

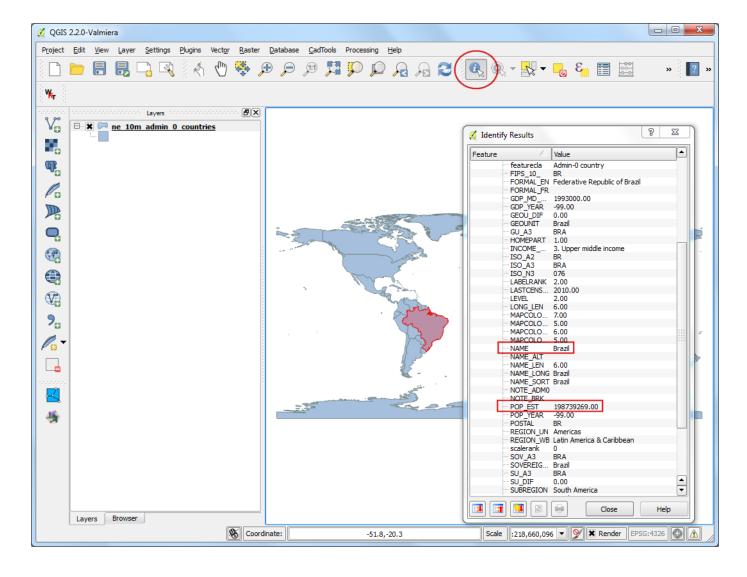
Загрузите **вышьшь neighbors.py** и сохраните его на диск.

#### Методика

1. Загрузите слой *ne\_10m\_admin\_0\_countries* перейдя к меню Layer → Add Vector Layer.



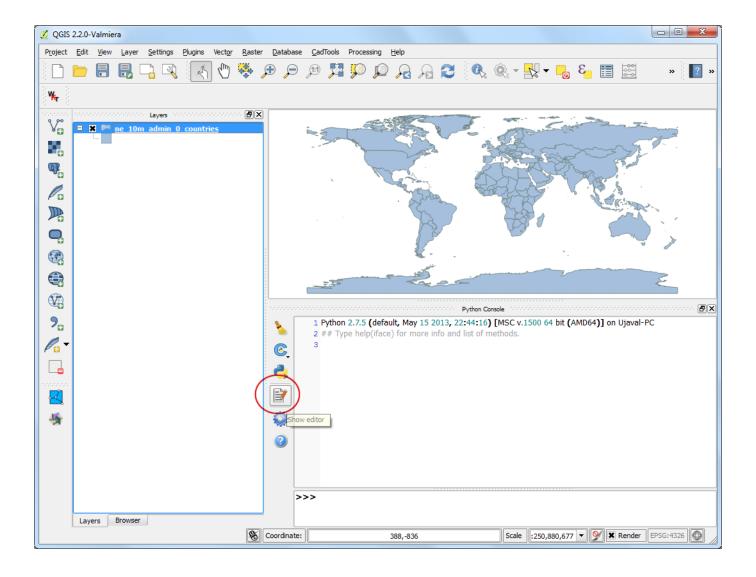
2. Скрипт использует 2 поля для выполнения действия. Название поля и поле, которое вы хотите просуммировать. Используйте инструмент Identify, чтобы нажать на любом объекте и просмотреть его атрибуты. В данном случае название поля NAME, и мы хотим просуммировать оценку численности народонаселения из поля POP\_EST.



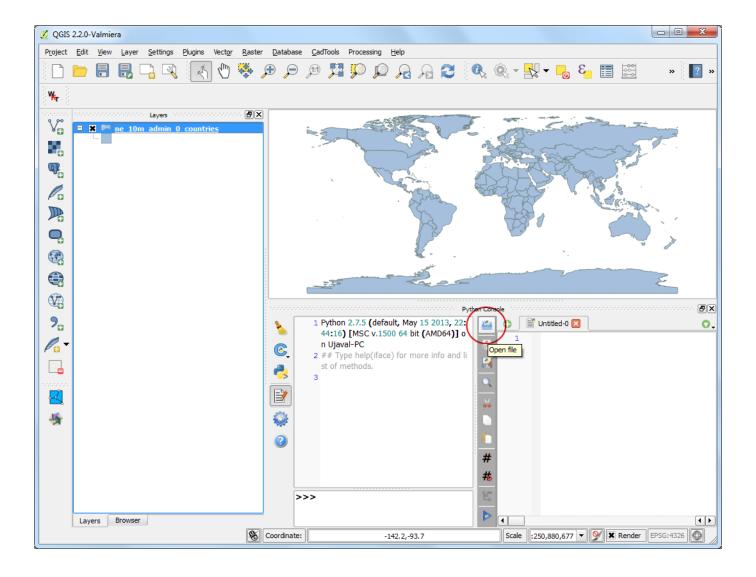
3. Перейдите к меню Plugins • Python Console.



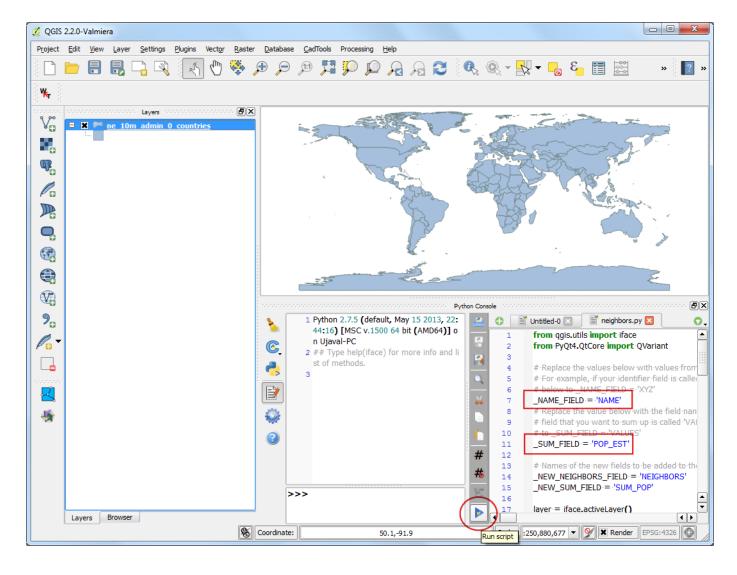
4. В окне Python Console нажмите на кнопку Show Editor.



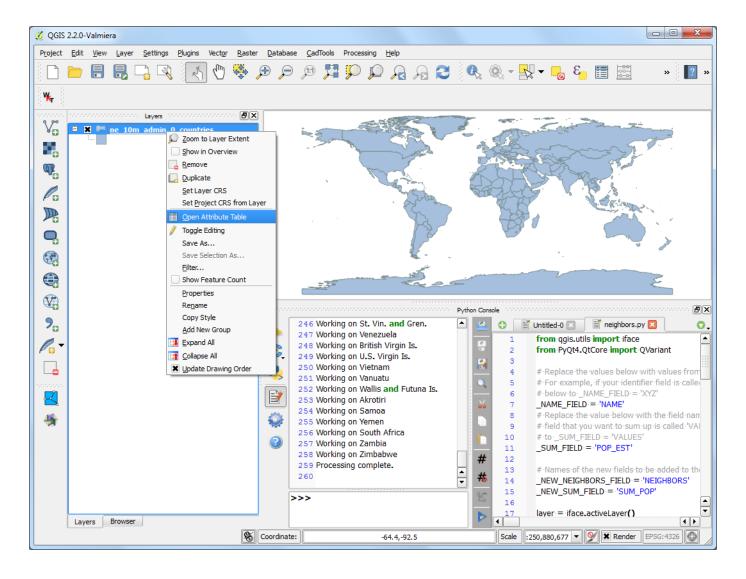
5. В панели Editor нажмите на кнопку Open file и перейдите к загруженному сценарию neighbors.py, после чего нажмите Open.



6. После того, как скрипт загружен, вы можете изменить значения \_NAME\_FIELD и \_SUM\_FIELD в соответствии с атрибутами вашего собственного слоя. Если вы работаете со слоем ne\_10m\_admin\_0\_countries, вы можете оставить всё, как есть. Нажмите: guilabel: save кнопку в: guilabel: панели Editor`, если вы сделали какие-либо изменения. Теперь нажмите: guilabel: кнопку Run script, для выполнения сценария.



7. После того, как скрипт закончит работу, щелкните правой кнопкой мыши на слое ne\_10m\_admin\_0\_countries и выберите Open Aтрибут Table.



8. Вы заметите, 2 новых атрибута под названием **NEIGHBORS** и **SUM**. Они были добавлены сценарием.

		E   E		<b>)</b> [9]					
	ION	REGION_WB	NAME_LEN	LONG_LEN	ABBREV_LEN	TINY	HOMEPART	NEIGHBORS	SUM
)		Latin America &	5.00	5.00	5.00	4.00	-99.00	NULL	0
1	sia	South Asia	11.00	11.00	4.00	-99.00	1.00	Iran,Turkmenista	1621125240
2	a	Sub-Saharan Africa	6.00	6.00	4.00	-99.00	1.00	Namibia,Zambia,	86676756
3		Latin America &	8.00	8.00	4.00	-99.00	-99.00	NULL	0
4	ırope	Europe & Central	7.00	7.00	4.00	-99.00	1.00	Macedonia, Greec	15281164
5	irope	Europe & Central	5.00	13.00	5.00	5.00	-99.00	NULL	0
6	ırope	Europe & Central	7.00	7.00	4.00	5.00	1.00	France,Spain	104582794
7	ia	Middle East & No	20.00	20.00	6.00	-99.00	1.00	Saudi Arabia,Oman	32104718
8	ica	Latin America &	9.00	9.00	4.00	-99.00	1.00	Bolivia,Paraguay,	235606259
9	ia	Europe & Central	7.00	7.00	4.00	-99.00	1.00	Georgia,Turkey,I	156089287
10		East Asia & Pacific	14.00	14.00	9.00	3.00	-99.00	NULL	0
11		Antarctica	10.00	10.00	4.00	-99.00	1.00	NULL	0
12	d Ne	East Asia & Pacific	23.00	27.00	7.00	-99.00	-99.00	NULL	0
13	(ope	Sub-Saharan Africa	22.00	35.00	10.00	2.00	-99.00	NULL	0
14		Latin America &	17.00	19.00	6.00	4.00	1.00	NULL	0
15	d Ne	East Asia & Pacific	9.00	9.00	4.00	-99.00	1.00	NULL	0
16	rope	Europe & Central	7.00	7.00	5.00	-99.00	1.00	Italy,Hungary,Slo	175681436
17	ia	Europe & Central	10.00	10.00	4.00	-99.00	1.00	Georgia,Turkey,R	290858866
18	ica	Sub-Saharan Africa	7.00	7.00	4.00	-99.00	1.00	Rwanda, Tanzani	120214356
19	rope	Europe & Central	7.00	7.00	5.00	-99.00	1.00	France,Netherla	163595324
20	rica	Sub-Saharan Africa	5.00	5.00	5.00	-99.00	1.00	Nigeria,Niger,Bur	186301451
21	rica	Sub-Saharan Africa	12.00	12.00	4.00	-99.00	1.00	Mali,Niger,Ghana	87234511
22	sia	South Asia	10.00	10.00	5.00	-99.00	1.00	India,Myanmar	1214216958
4									::::: [ <b>1</b> ]

Ниже для справки приведен полный сценарий. Вы можете изменить его в соответствии с вашими потребностями.

```
# Copyright 2014 Ujaval Gandhi
#This program is free software; you can redistribute it and/or
#modify it under the terms of the GNU General Public License
#as published by the Free Software Foundation; either version 2
#of the License, or (at your option) any later version.
#This program is distributed in the hope that it will be useful,
#but WITHOUT ANY WARRANTY; without even the implied warranty of
#MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
#GNU General Public License for more details.
#You should have received a copy of the GNU General Public License
#along with this program; if not, write to the Free Software
#Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.
from qgis.utils import iface
from PyQt4.QtCore import QVariant
# Replace the values below with values from your layer.
# For example, if your identifier field is called 'XYZ', then change the line
# below to _NAME_FIELD = 'XYZ'
NAME FIELD = 'NAME'
# Replace the value below with the field name that you want to sum up.
# For example, if the # field that you want to sum up is called 'VALUES', then
# change the line below to _SUM_FIELD = 'VALUES'
```

```
_SUM_FIELD = 'POP_EST'
# Names of the new fields to be added to the layer
_NEW_NEIGHBORS_FIELD = 'NEIGHBORS'
_NEW_SUM_FIELD = 'SUM'
layer = iface.activeLayer()
# Create 2 new fields in the layer that will hold the list of neighbors and sum
# of the chosen field.
layer.startEditing()
layer.dataProvider().addAttributes(
        [QgsField(_NEW_NEIGHBORS_FIELD, QVariant.String),
         QgsField(_NEW_SUM_FIELD, QVariant.Int)])
layer.updateFields()
# Create a dictionary of all features
feature_dict = {f.id(): f for f in layer.getFeatures()}
# Build a spatial index
index = QgsSpatialIndex()
for f in feature_dict.values():
    index.insertFeature(f)
# Loop through all features and find features that touch each feature
for f in feature_dict.values():
   print 'Working on %s' % f[_NAME_FIELD]
    geom = f.geometry()
    # Find all features that intersect the bounding box of the current feature.
    # We use spatial index to find the features intersecting the bounding box
    # of the current feature. This will narrow down the features that we need
    # to check neighboring features.
    intersecting_ids = index.intersects(geom.boundingBox())
    # Initalize neighbors list and sum
   neighbors = []
    neighbors_sum = 0
    for intersecting_id in intersecting_ids:
        # Look up the feature from the dictionary
        intersecting_f = feature_dict[intersecting_id]
        # For our purpose we consider a feature as 'neighbor' if it touches or
        # intersects a feature. We use the 'disjoint' predicate to satisfy
        # these conditions. So if a feature is not disjoint, it is a neighbor.
        if (f != intersecting_f and
            not intersecting_f.geometry().disjoint(geom)):
            neighbors.append(intersecting_f[_NAME_FIELD])
            neighbors_sum += intersecting_f[_SUM_FIELD]
    f[_NEW_NEIGHBORS_FIELD] = ','.join(neighbors)
    f[_NEW_SUM_FIELD] = neighbors_sum
    # Update the layer with new attribute values.
    layer.updateFeature(f)
layer.commitChanges()
print 'Processing complete.'
```