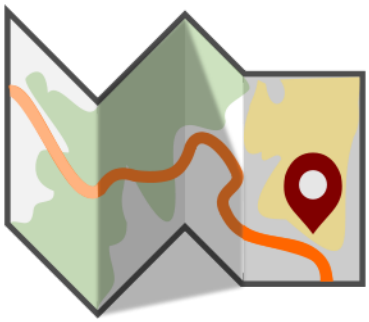
<?py # This templating language is pyTenjin ?> <?py # http://www.kuwata-lab.com/tenjin/pytenjin-users-guide.html ?>

<?py if title: ?>

# #{title}

<?py #end ?>

## QGIS Tutorials and Tips




Author

<?py for author in authors: ?> .. cssclass:: author

#{author}

http://www.spatialthoughts.com


<?py #end ?>

# Writing Python Scripts for Processing Framework

> ***Warning***
>
> A new version of this tutorial is available at Writing Python Scripts for Processing Framework (QGIS3)

One can write standalone pyqgis scripts that can be run via the Python Console in QGIS. With a few tweaks, you can make your standalone scripts run via the Processing Framework. This has several advantages. First, taking user input and writing output files is far easier because Processing Framework offers standardized user interface for these. Second, having your script in the Processing Toolbox also allows it to be part of any Processing Model or be run as a Batch job with multiple inputs. This tutorial will show how to write a custom python script that can be part of the Processing Framework in QGIS.

## Overview of the task

Our script will perform a dissolve operation based on a field chosen by the user. It will also sum up values of another field for the dissolved features. In the example, we will dissolve a world shapefile based on a `SUBREGION` attribute and sum up `POP_EST` field to calculate total population in the dissolved region.

> ***Note***
>
> If you are looking to do a Dissolve operation along with Statistics, you can use the excellent `DissolveWithStats` plugin. This script is a demonstration how to implement similar functionality via a Processing script.

## Get the data

We will use the Admin 0 - Countries dataset from Natural Earth.

Download the Admin 0 - countries shapefile..

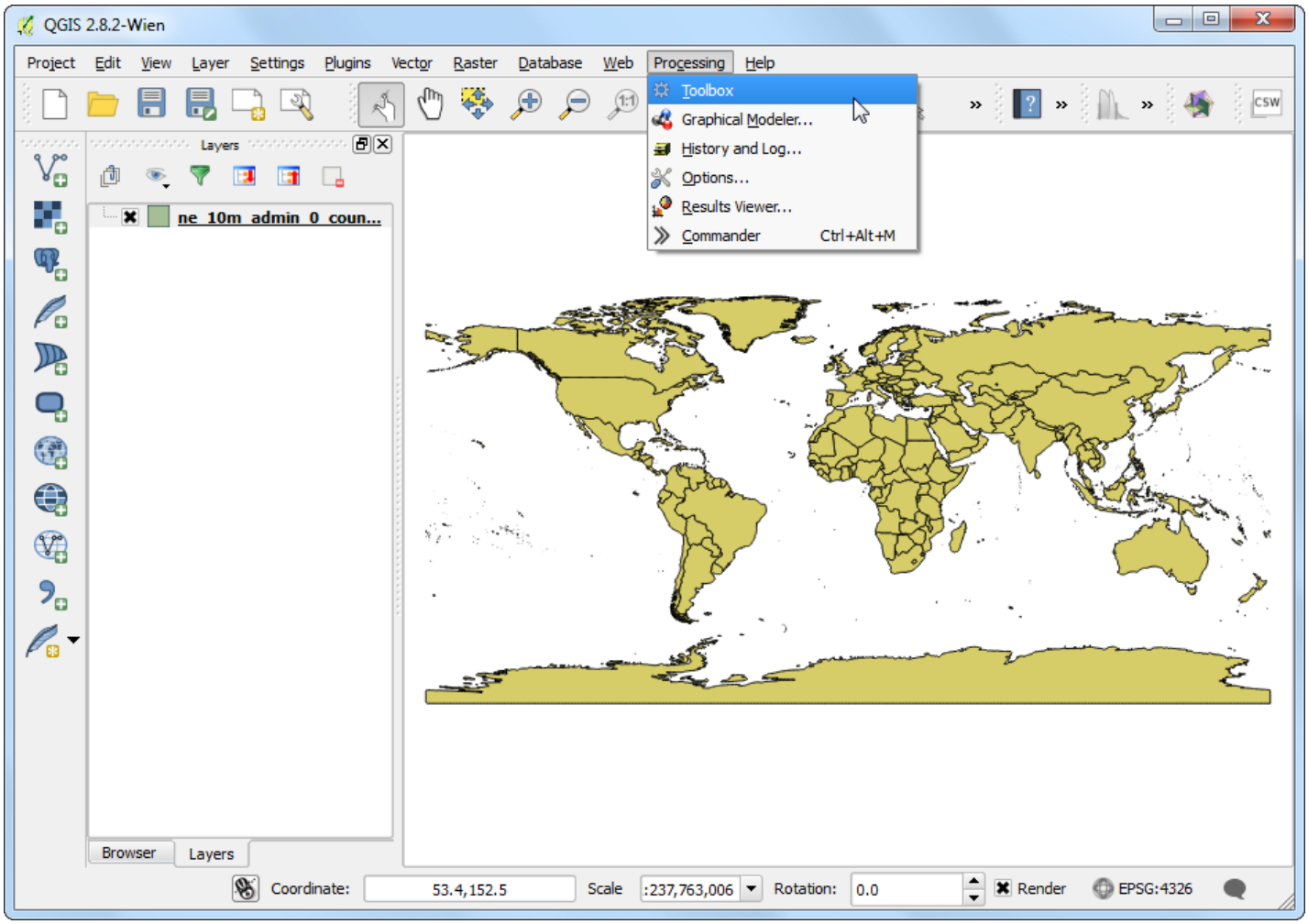
Data Source [NATURALEARTH]

For convenience, you may directly download a copy of the dataset from the links below:

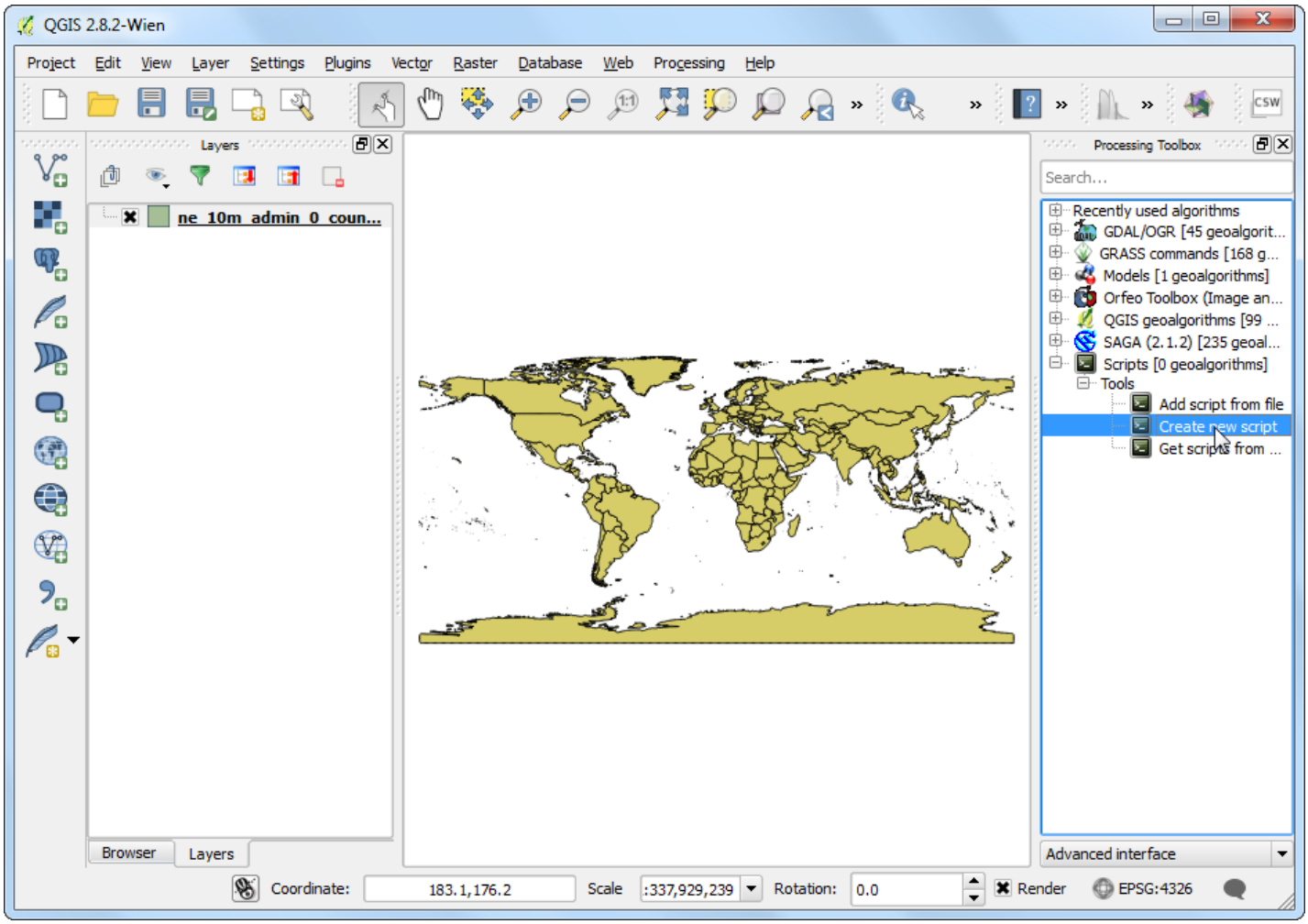
ne_10_admin_0_countries.zip

## Procedure

1. Open QGIS and go to Layers ▸ Add Vector ▸ Add Vector Layer. Browse to the downloaded `ne_10_admin_0_countries.zip` file and load the `ne_10_admin_0_countries` layer. Go to Processing ▸ Toolbox.

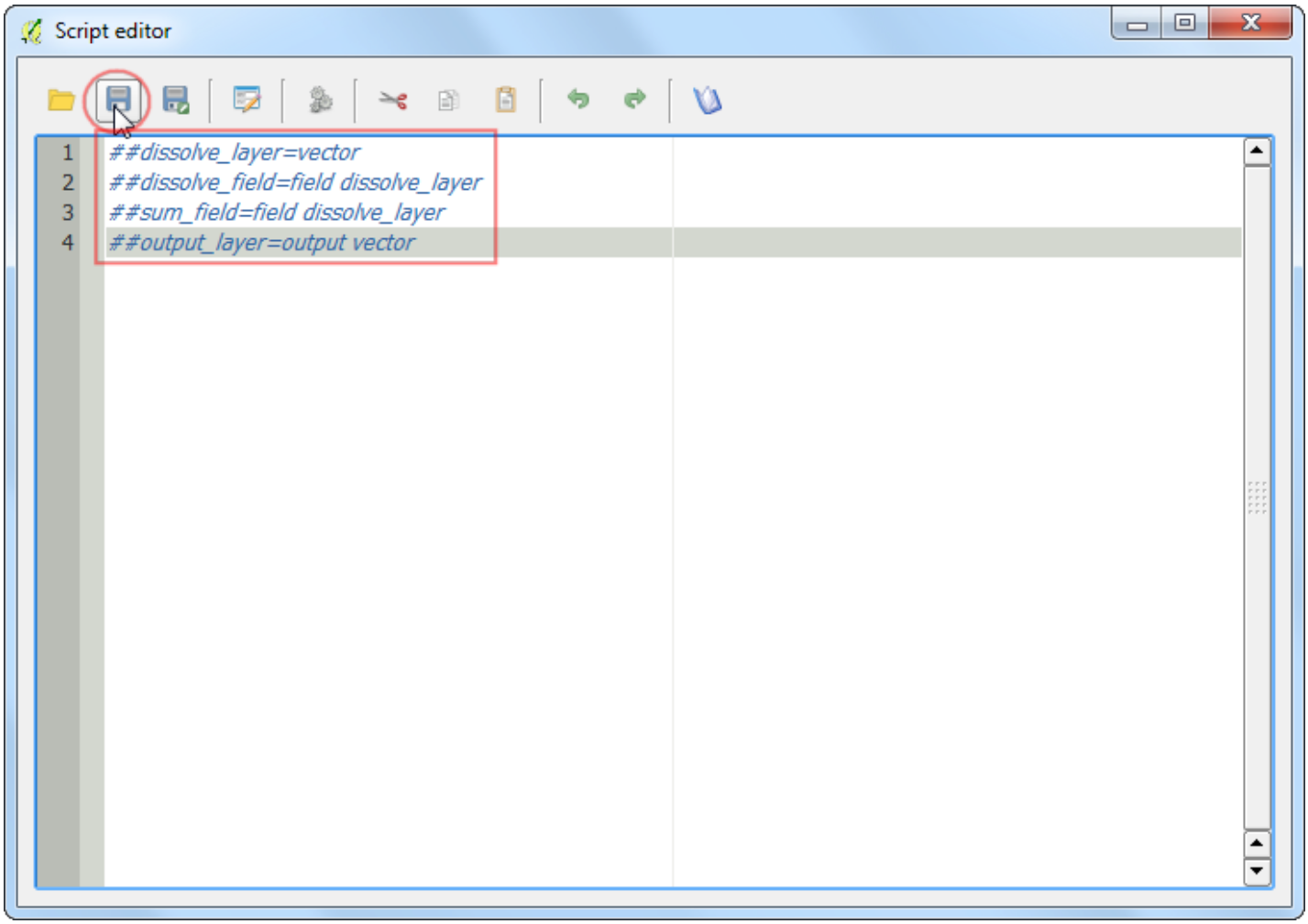



2. Expand the Scripts group in the Processing Toolbox and select Create a new script.

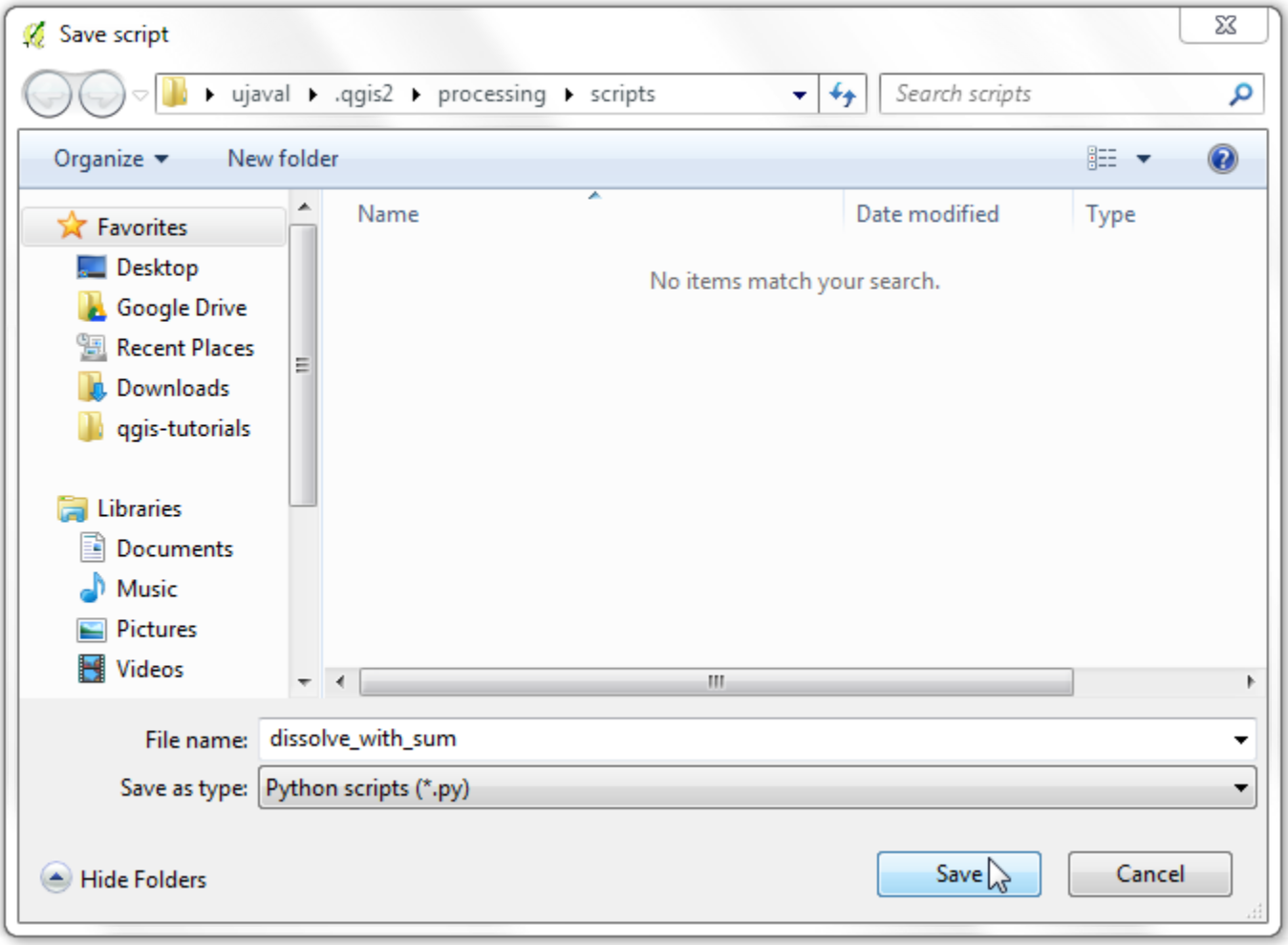



3. For a python script to be recognized as a Processing script, the beginning of the script must be the specifications of the input and outputs. This will be used to construct the user interface to run the script. You can learn more about the format of these lines from QGIS Processing Documentation. Enter the following lines in the Script editor. Here we are specifying 3 user inputs: `dissolve_layer`, `dissolve_field` and `sum_field`. Note that we are adding `dissolve_layer` after both the field input definitions. This means that input will be pre-populated with choice of fields from the `dissolve_layer`. We also specify the `output_layer` as the output vector layer. Click Save button.

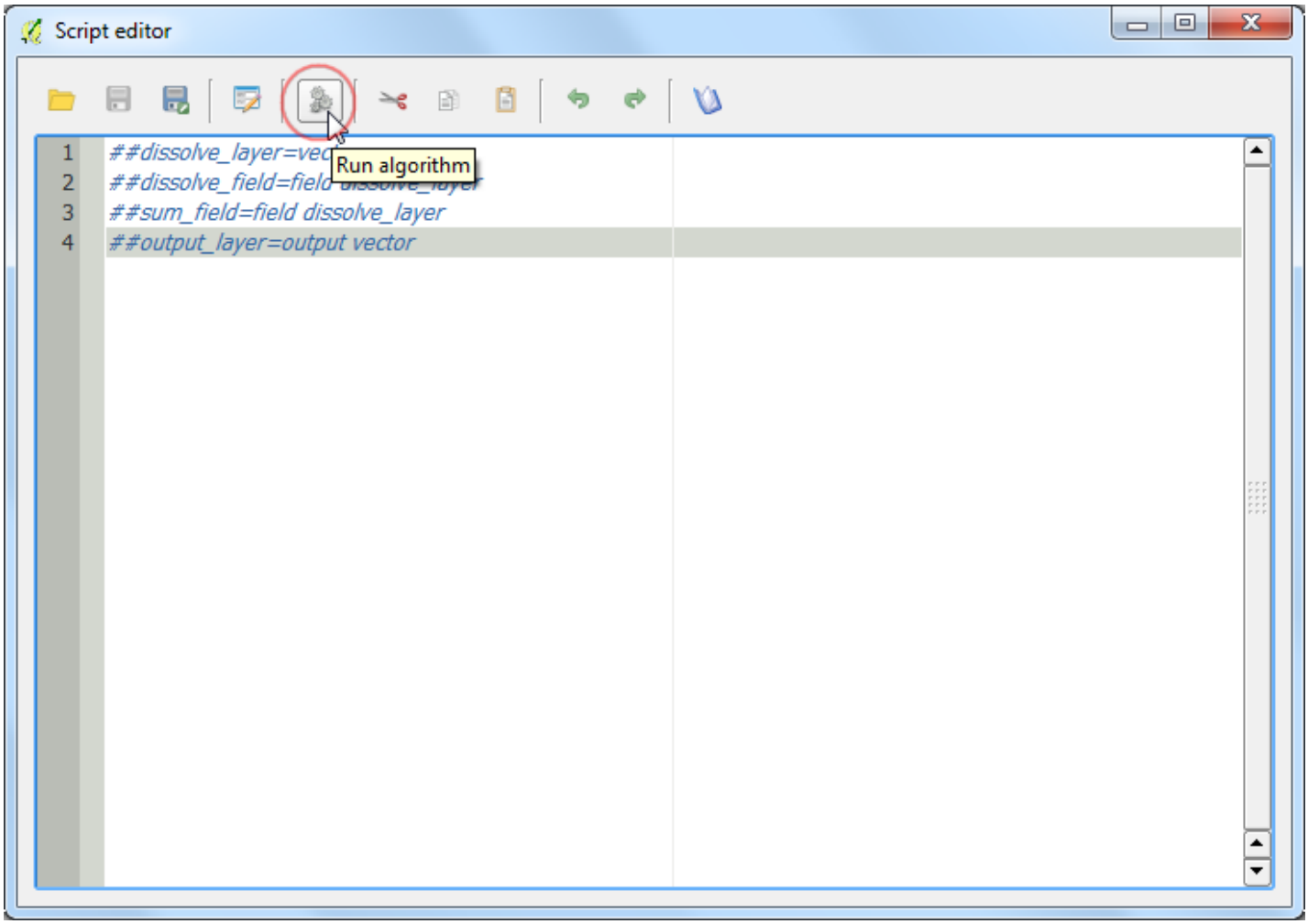
```
##dissolve_layer=vector
##dissolve_field=field dissolve_layer
##sum_field=field dissolve_layer
##output_layer=output vector
```



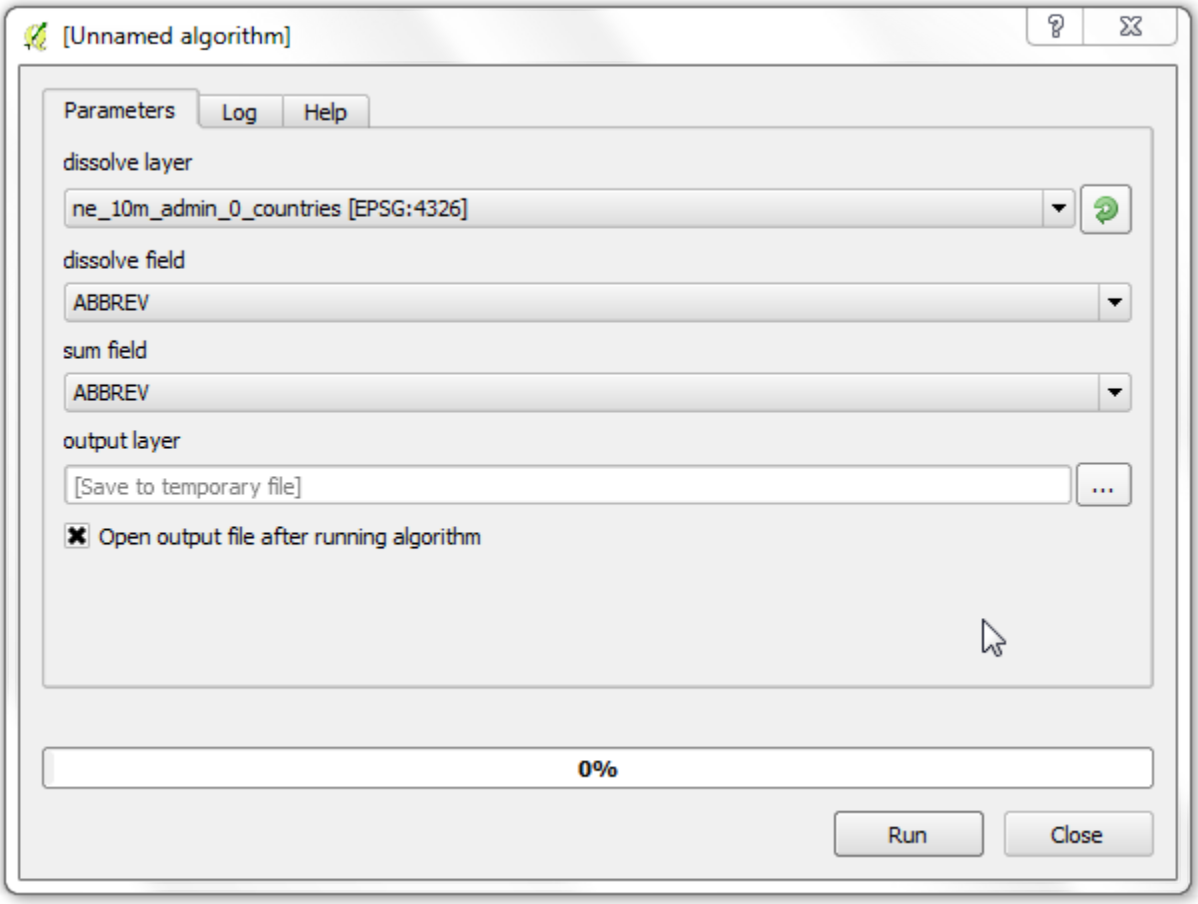

4. Name the script `dissolve_with_sum` and save it at the default location under .qgis2 ▸ processing ▸ scripts folder.




5. Back in the Script editor, click Run algorithm button to preview the user interface.




6. You can see that just by adding a few lines, we have a nice user interface for the user to specify the inputs. It is also consistent with all other Processing algorithms, so there is no learning curve involved in using your custom algorithm.



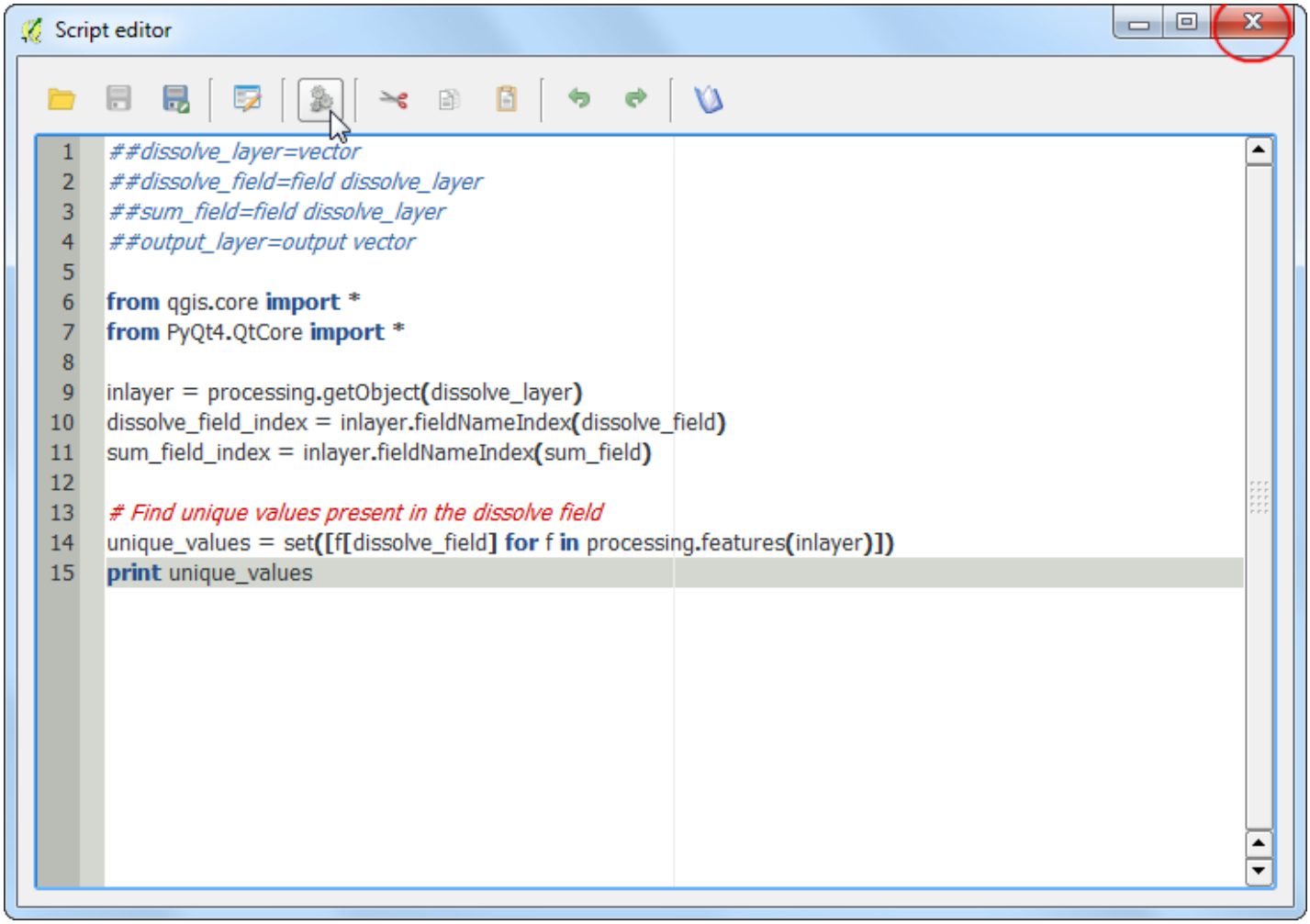

7. In the Script editor, enter the following code. You will notice that we are using some special methods such as `processing.getObject()` and `processing.features()`. These are convenience wrappers that make it easy to work with data. You can learn more about these from Additional functions for handling data section of QGIS Processing Documentation. Click Save to save the newly entered code and then the X button to close the editor.

### *Note*

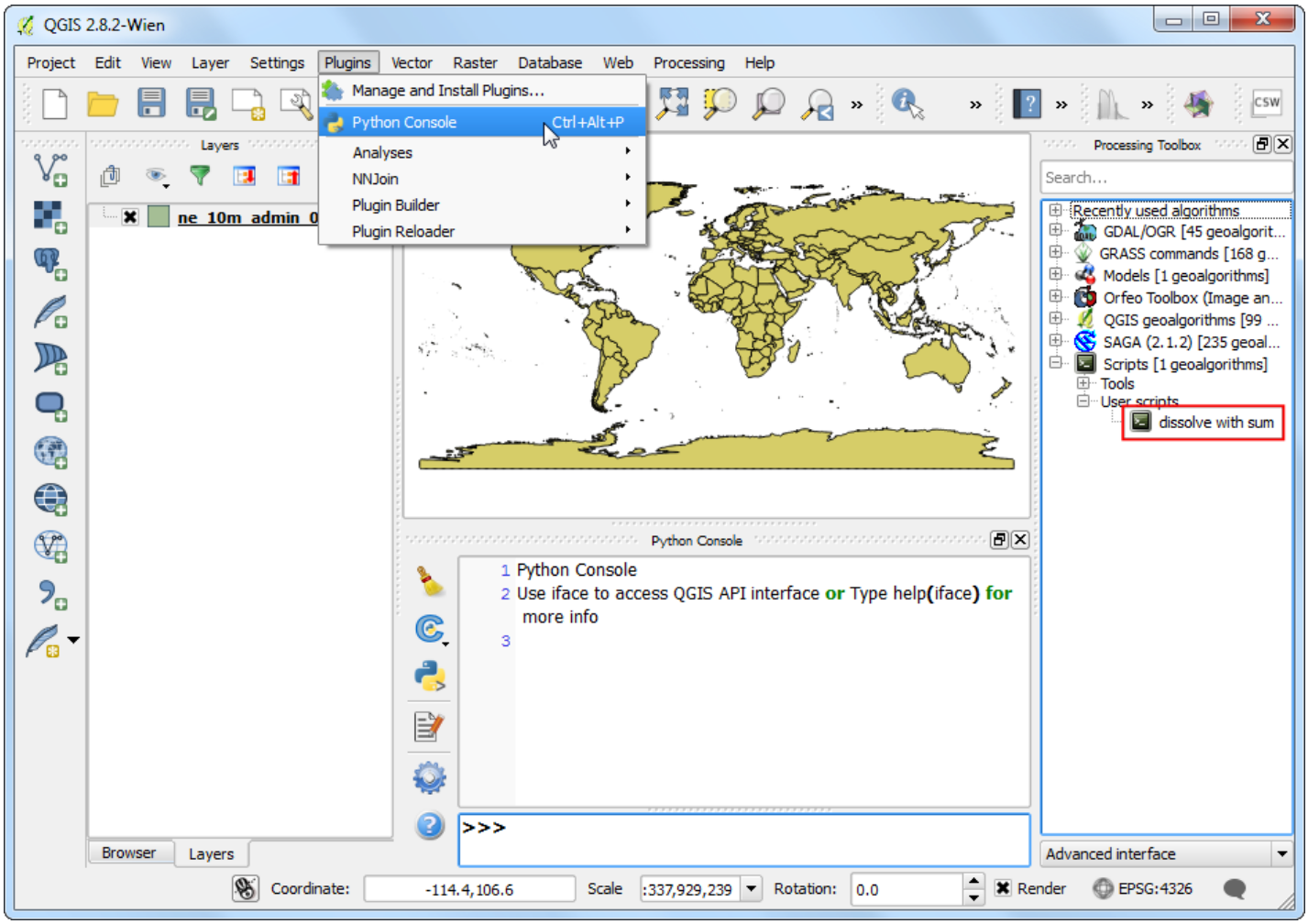
This script uses python list comprehensions extensively. Take a look at this list comprehension tutorial to get familiar with the syntax.

```
from qgis.core import *
from PyQt4.QtCore import *

inlayer = processing.getObject(dissolve_layer)
dissolve_field_index = inlayer.fieldNameIndex(dissolve_field)
sum_field_index = inlayer.fieldNameIndex(sum_field)

# Find unique values present in the dissolve field
unique_values = set([f[dissolve_field] for f in
processing.features(inlayer)])
```

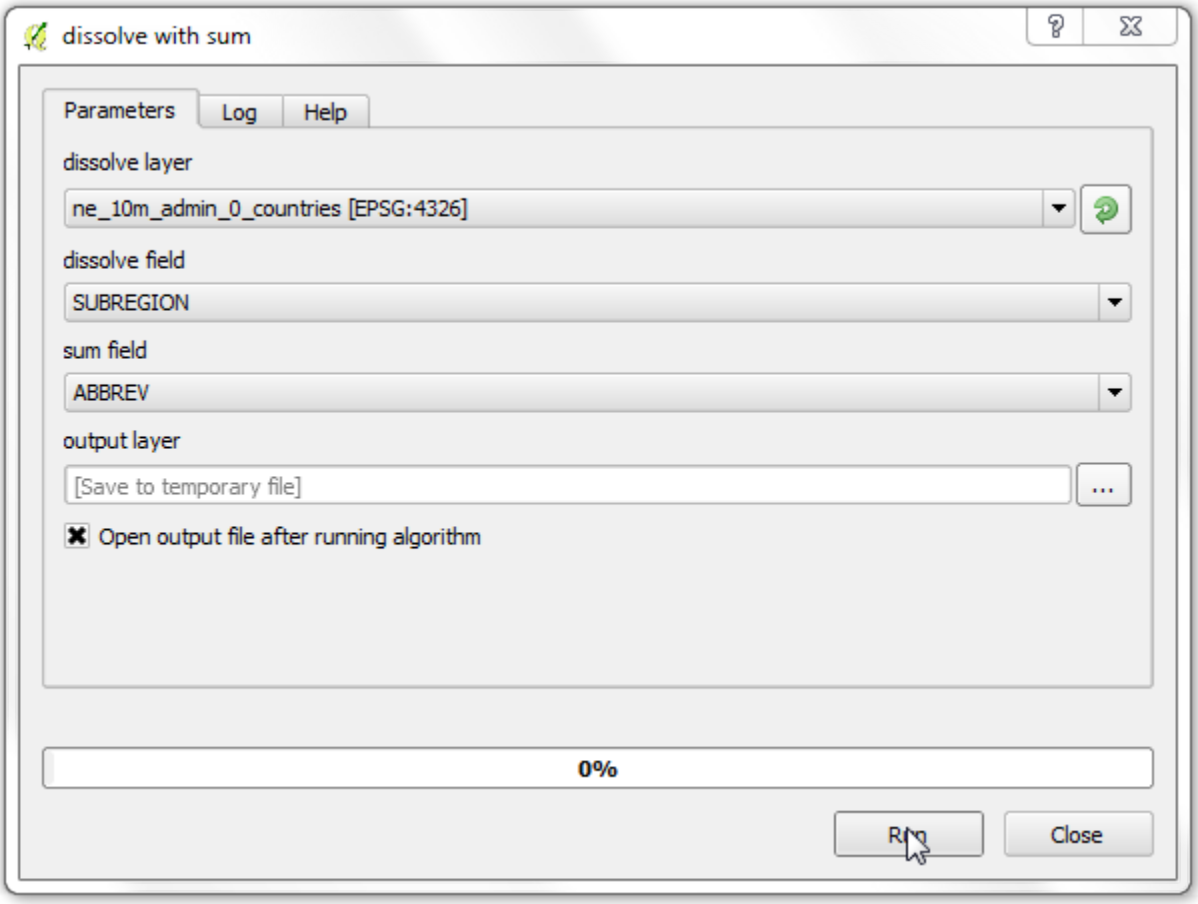
```
print unique_values
```



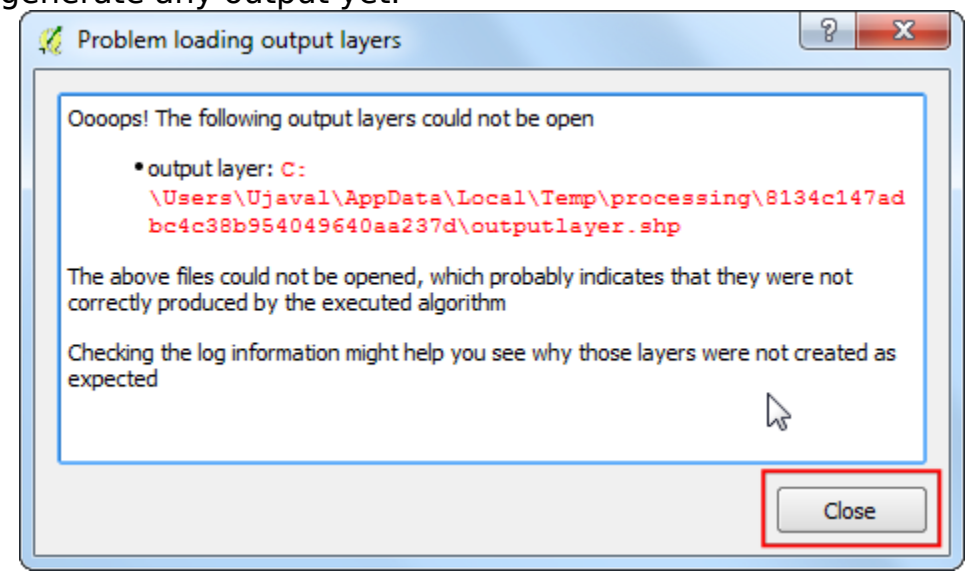

8. While writing code, it is important to be able to see errors and debug your code. Your processing scripts can be debugged easily via the built-in Python Console. In the main QGIS window, go to Plugins ▸ Python Console. Once the console is open, find your script in the Processing Toolbox and double-click it to launch it.



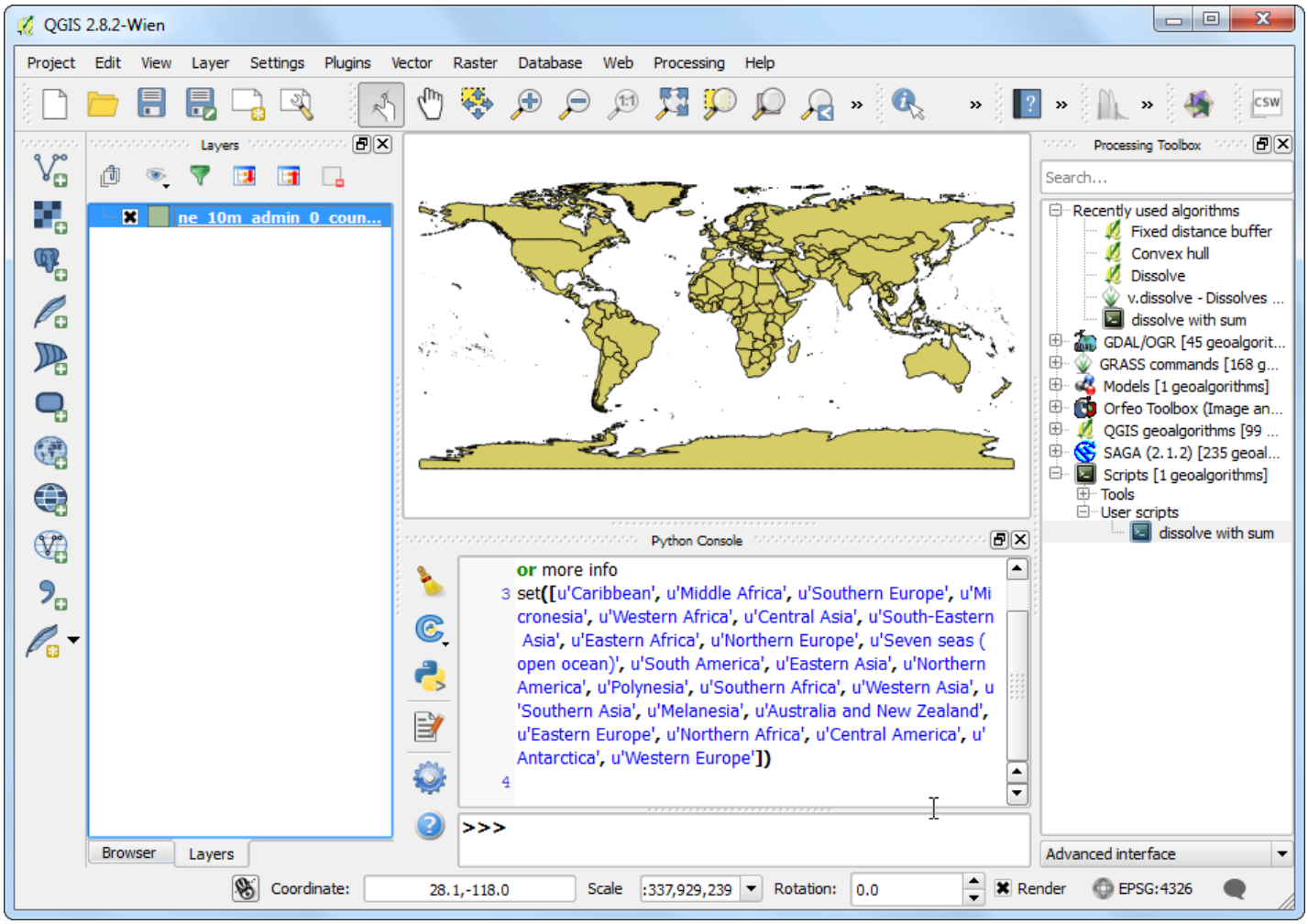

9. Select `SUBREGION` as the dissolve field. You may choose any field as the sum field as the script doesn't have any code yet to deal with it. Click Run.



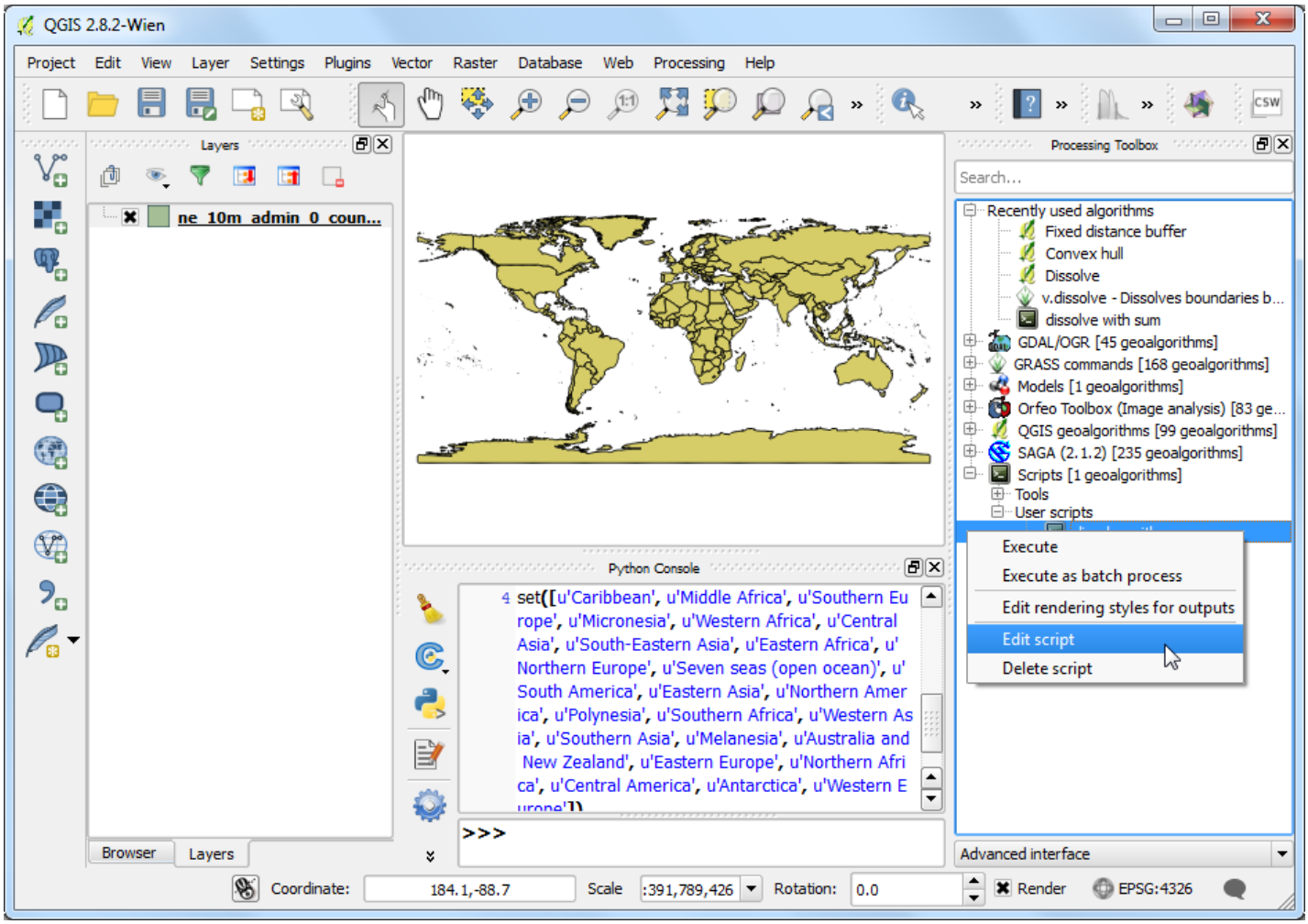

10. You will see an error dialog. This is expected since the script is incomplete and doesn't generate any output yet.




11. In the main QGIS windows, you will see the debug output from the script printed in the console. This is useful way to add print statements and see intermediate variable values.



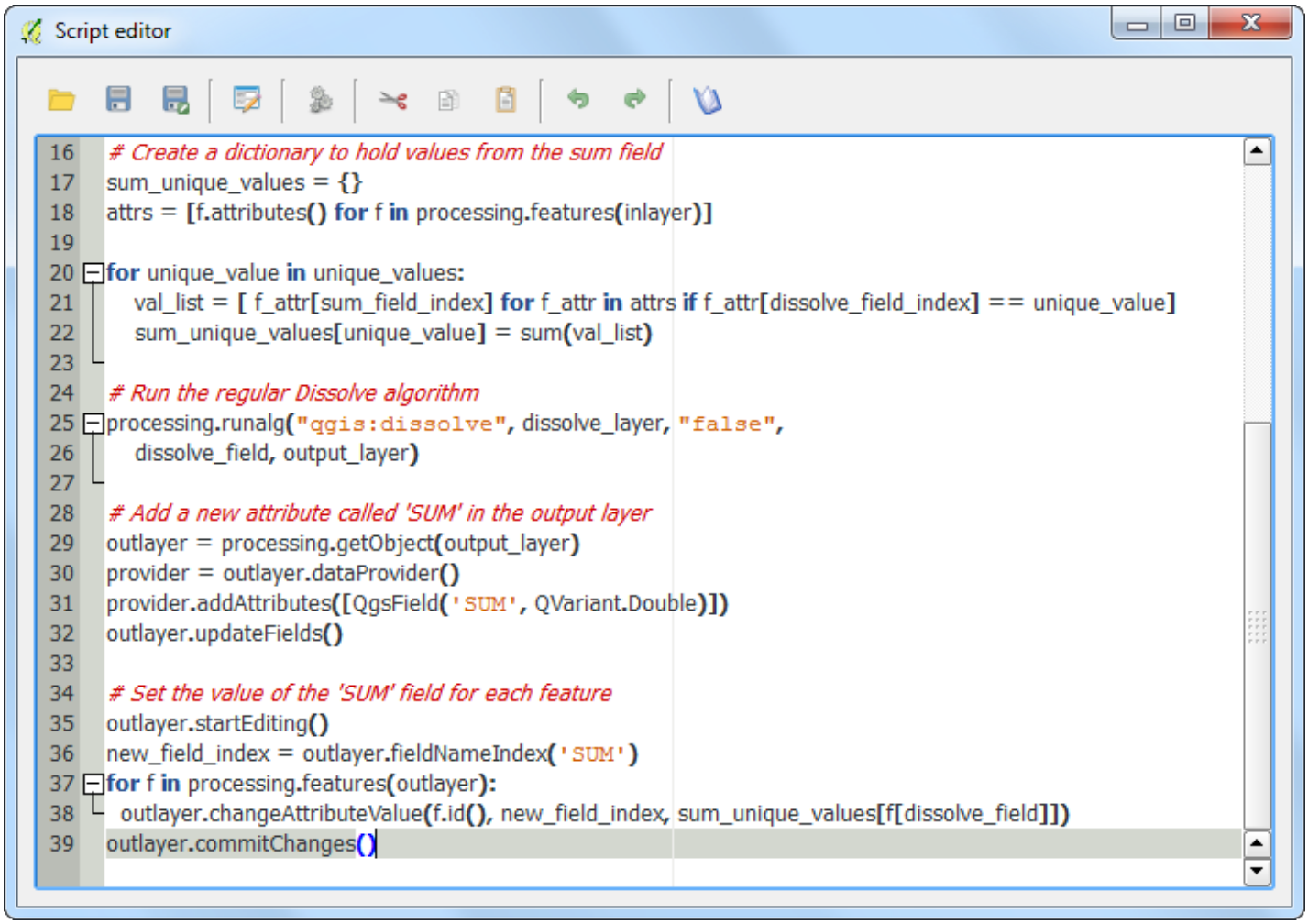

12. Let’s go back to editing the script by right-clicking the script and select Edit script.




13. Enter the following code to complete the script. Note that we are using the existing dissolve algorithm in QGIS via processing using `processing.runalg()` method.

```
# Create a dictionary to hold values from the sum field
sum_unique_values = {}
attrs = [f.attributes() for f in processing.features(inlayer)]

for unique_value in unique_values:
    val_list = [ f_attr[sum_field_index] for f_attr in attrs if f_attr[dissolve_field_index] =
    sum_unique_values[unique_value] = sum(val_list)

# Run the regular Dissolve algorithm
processing.runalg("qgis:dissolve", dissolve_layer, "false",
    dissolve_field, output_layer)

# Add a new attribute called 'SUM' in the output layer
outlayer = processing.getObject(output_layer)
provider = outlayer.dataProvider()
provider.addAttributes([QgsField('SUM', QVariant.Double)])
outlayer.updateFields()

# Set the value of the 'SUM' field for each feature
```

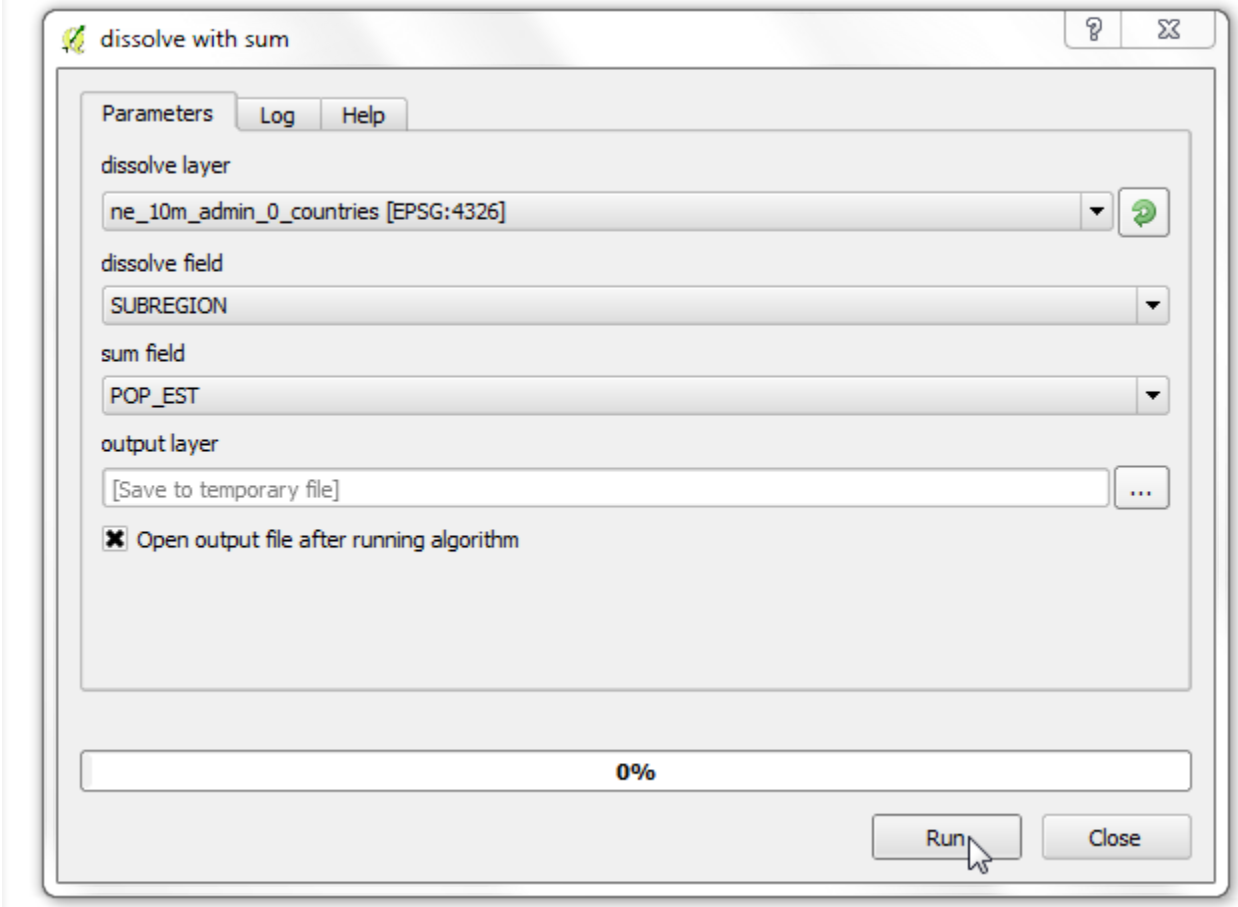
```
outlayer.startEditing()
new_field_index = outlayer.fieldNameIndex('SUM')
for f in processing.features(outlayer):
  outlayer.changeAttributeValue(f.id(), new_field_index, sum_unique_values[f[dissolve_field]])
outlayer.commitChanges()
```



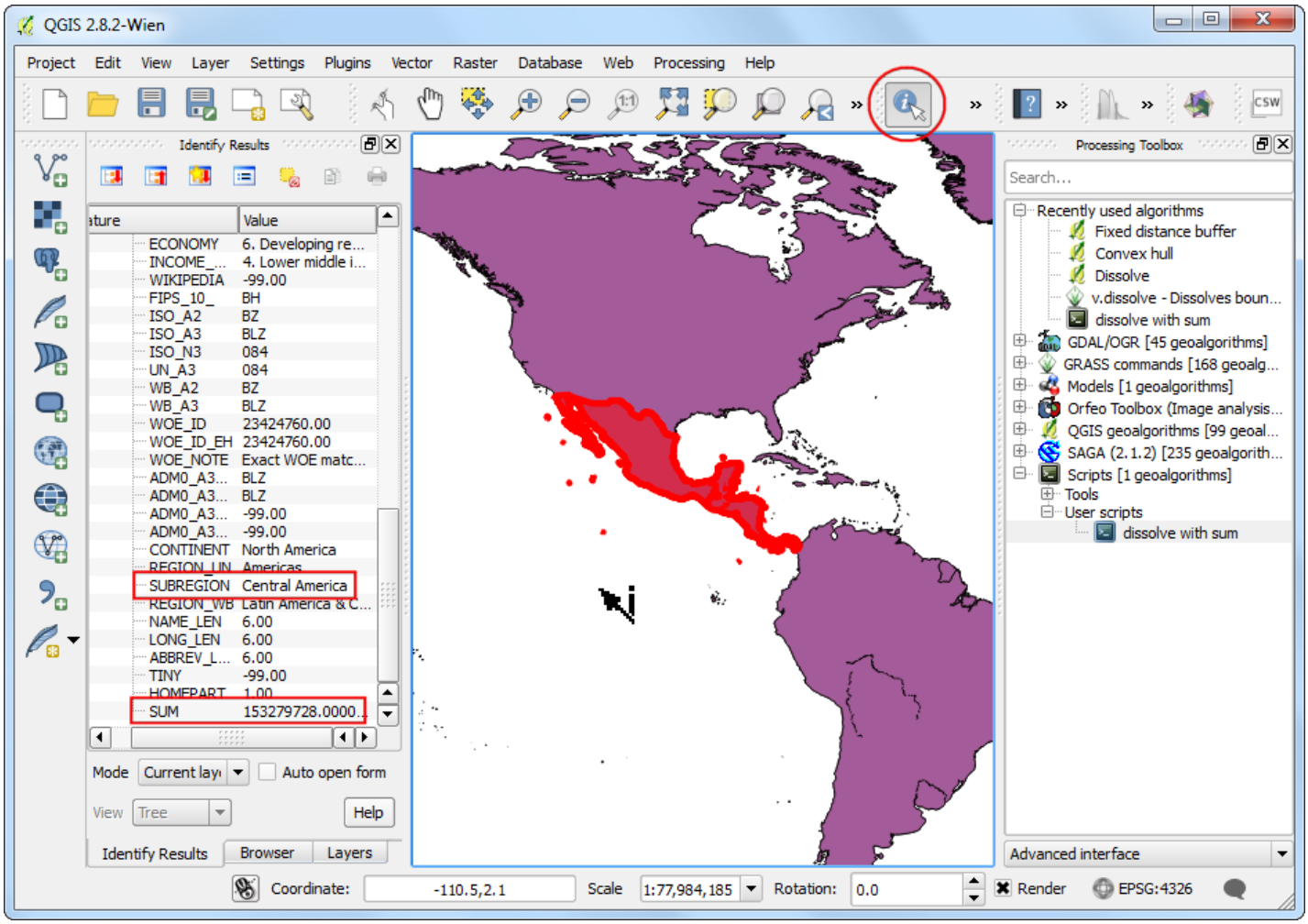

14. Run the algorithm by selecting `SUBREGION` as the dissolve field and `POP_EST` as the sum field. Click Run.

### *Note*

The processing algorithm may take upto 10 minutes to finish depending on your system.

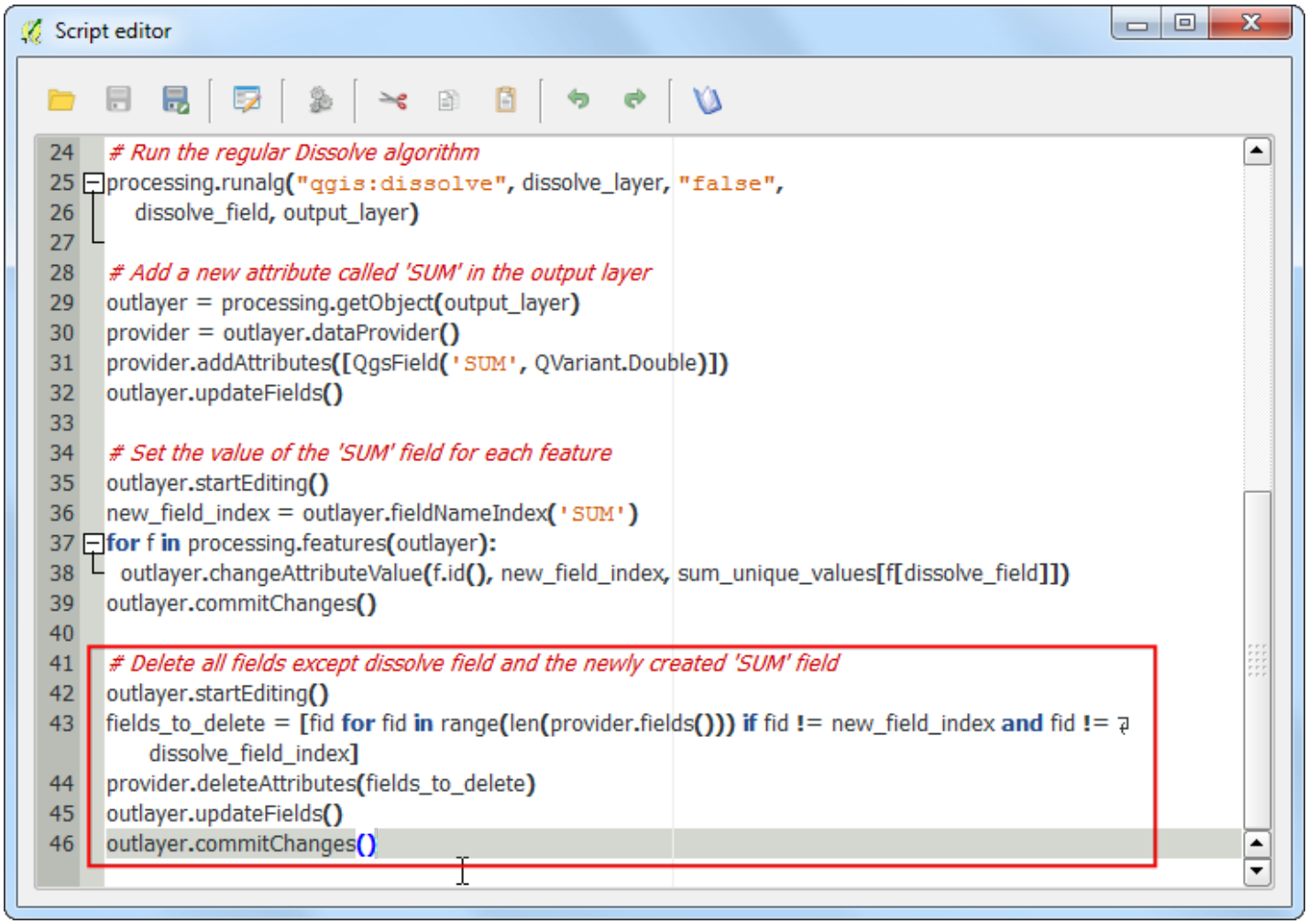



15. Once the processing finishes, you can use the Identify tool and click on any polygon. You will see the newly added `SUM` field with the `POP_EST` values from all original polygons added up.

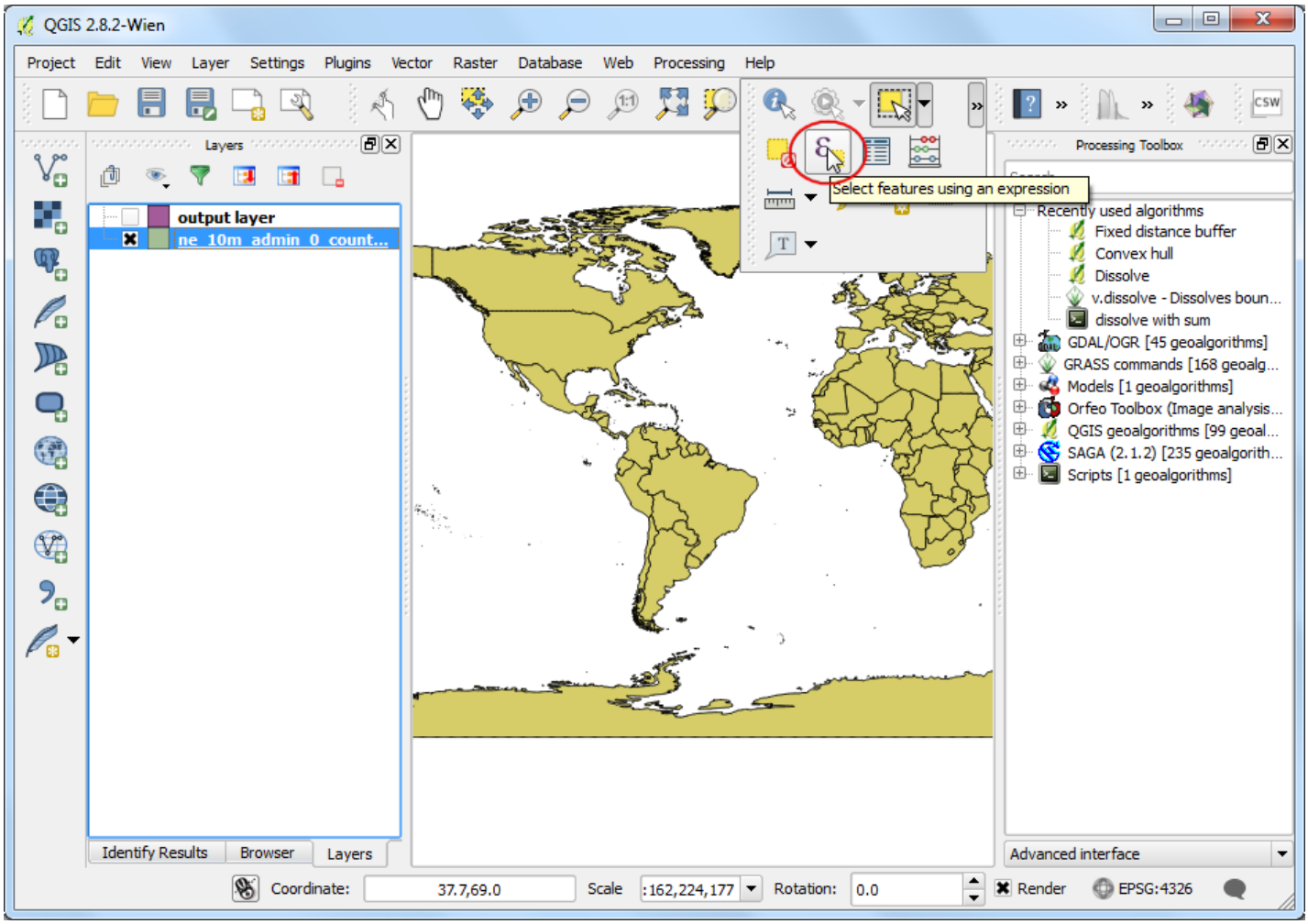



16. You will note that all other fields in the output are still present. When you dissolve many features to create a single feature, it doesn't make sense to keep the original fields in the output. Go back to the Script editor and add the following code to delete all fields except the SUM field and the field that was used to dissolve the original layer. Click Save button and close the window.

```
# Delete all fields except dissolve field and the newly created 'SUM' field.
outlayer.startEditing()

fields_to_delete = [fid for fid in range(len(provider.fields())) if fid != new_field_index and
provider.deleteAttributes(fields_to_delete)
outlayer.updateFields()

outlayer.commitChanges()
```

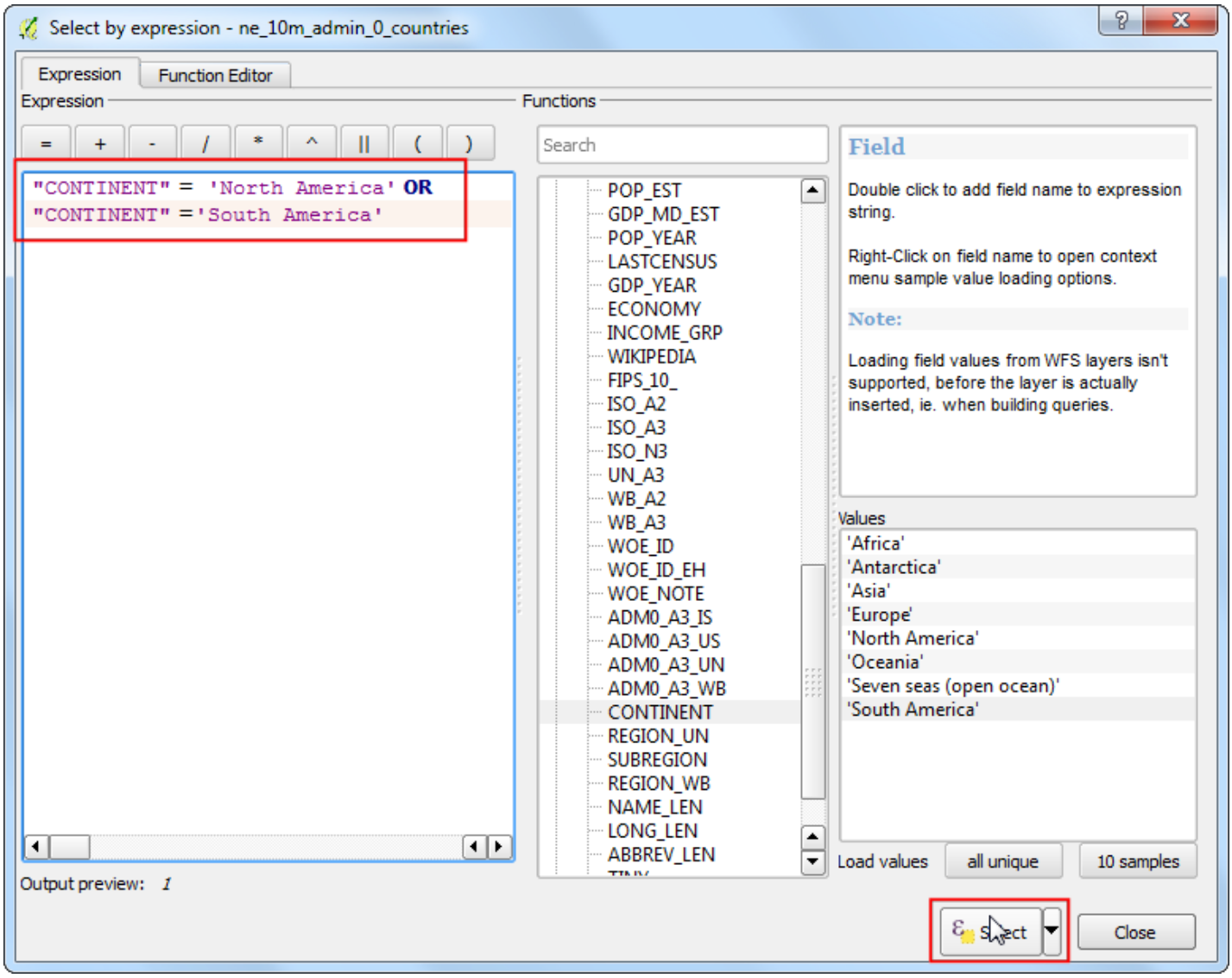



17. One of the hidden features of the Processing Framework is that all algorithms can work on selected features of a layer. This is very helpful when you want to run an algorithm on the subset of a layer. As our script uses `processing.features()` method to read features, it will respect the current selection. To demonstrate that, let's make a selection first. Click on the Select features using an expression button.

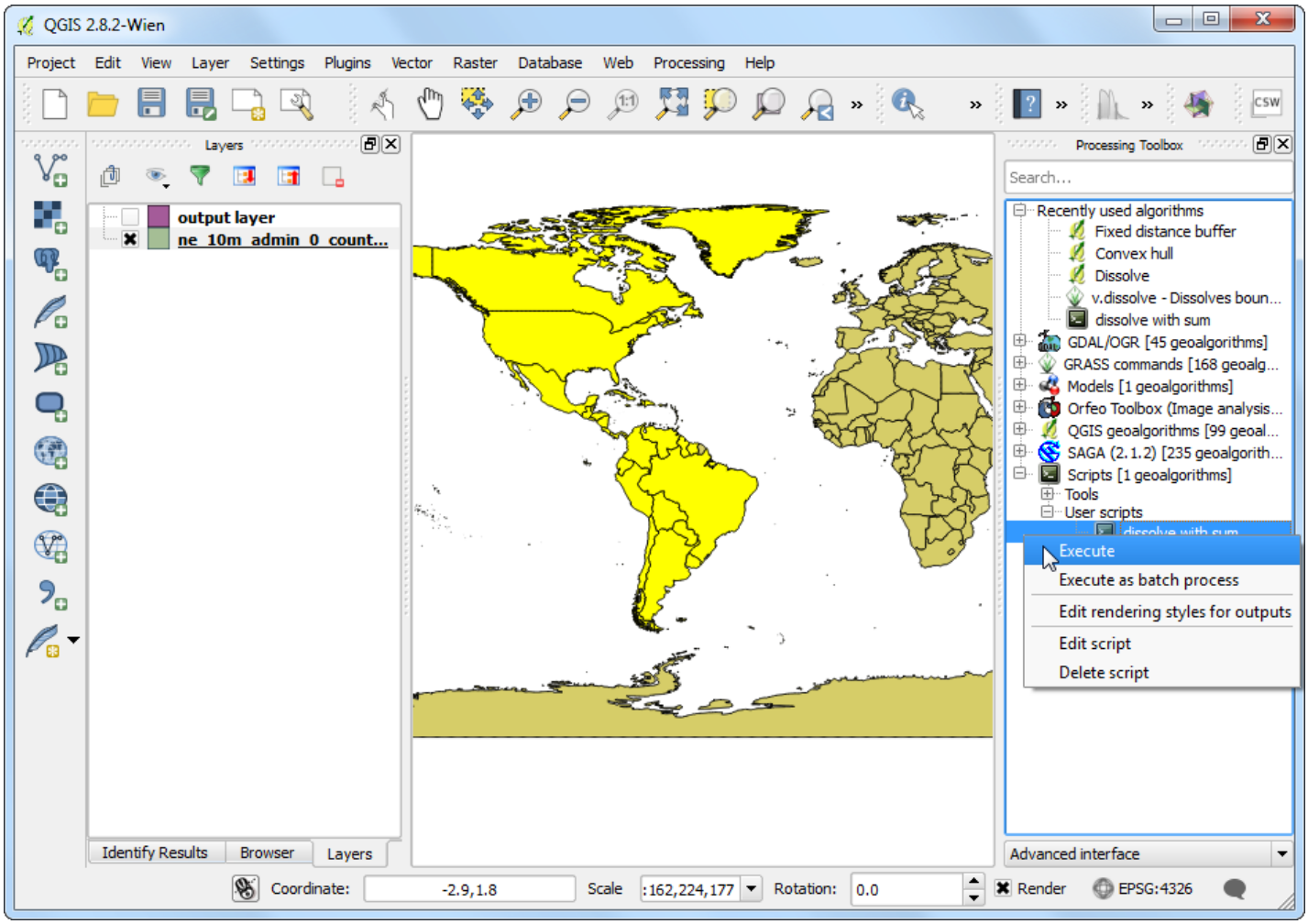



18. Enter the following expression to select features from North and South America and click Select.

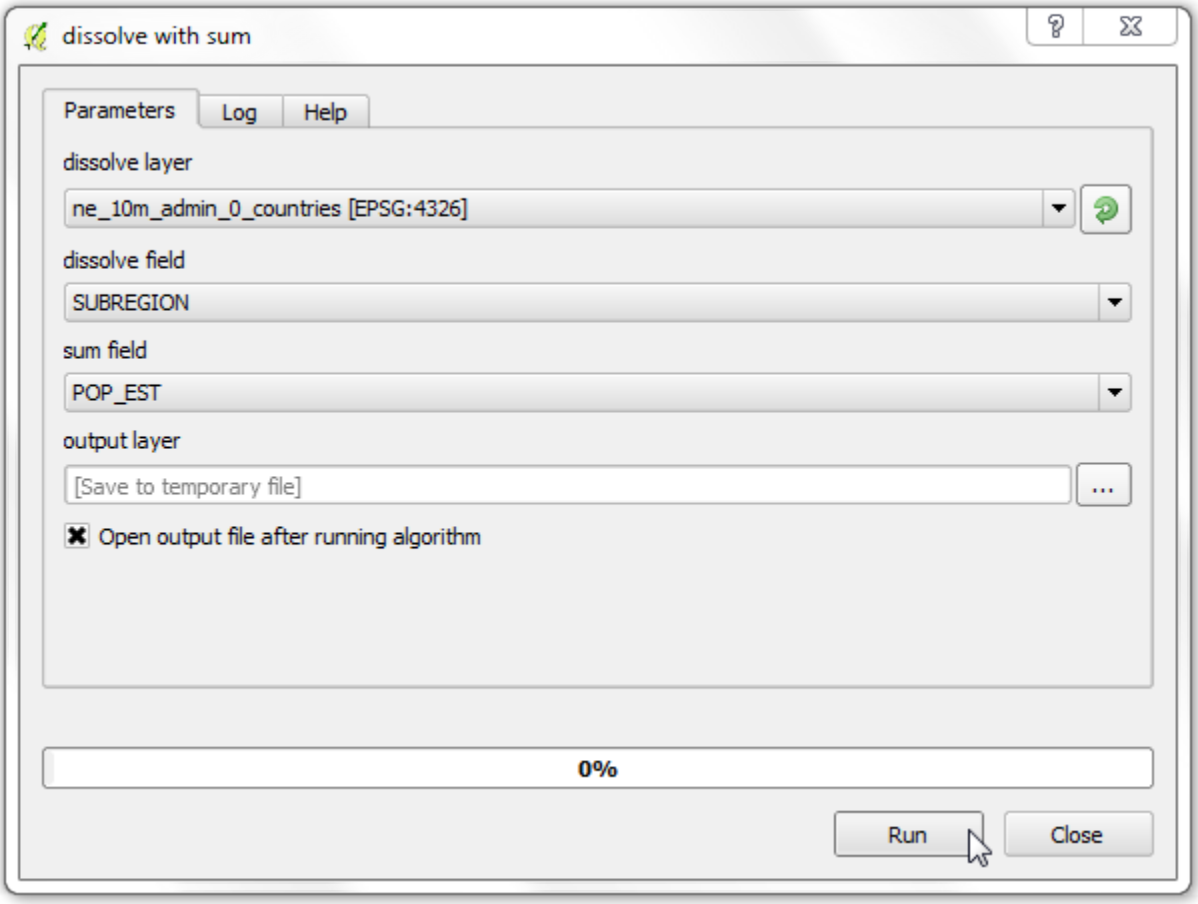
```
"CONTINENT" = 'North America' OR "CONTINENT" = 'South America'
```



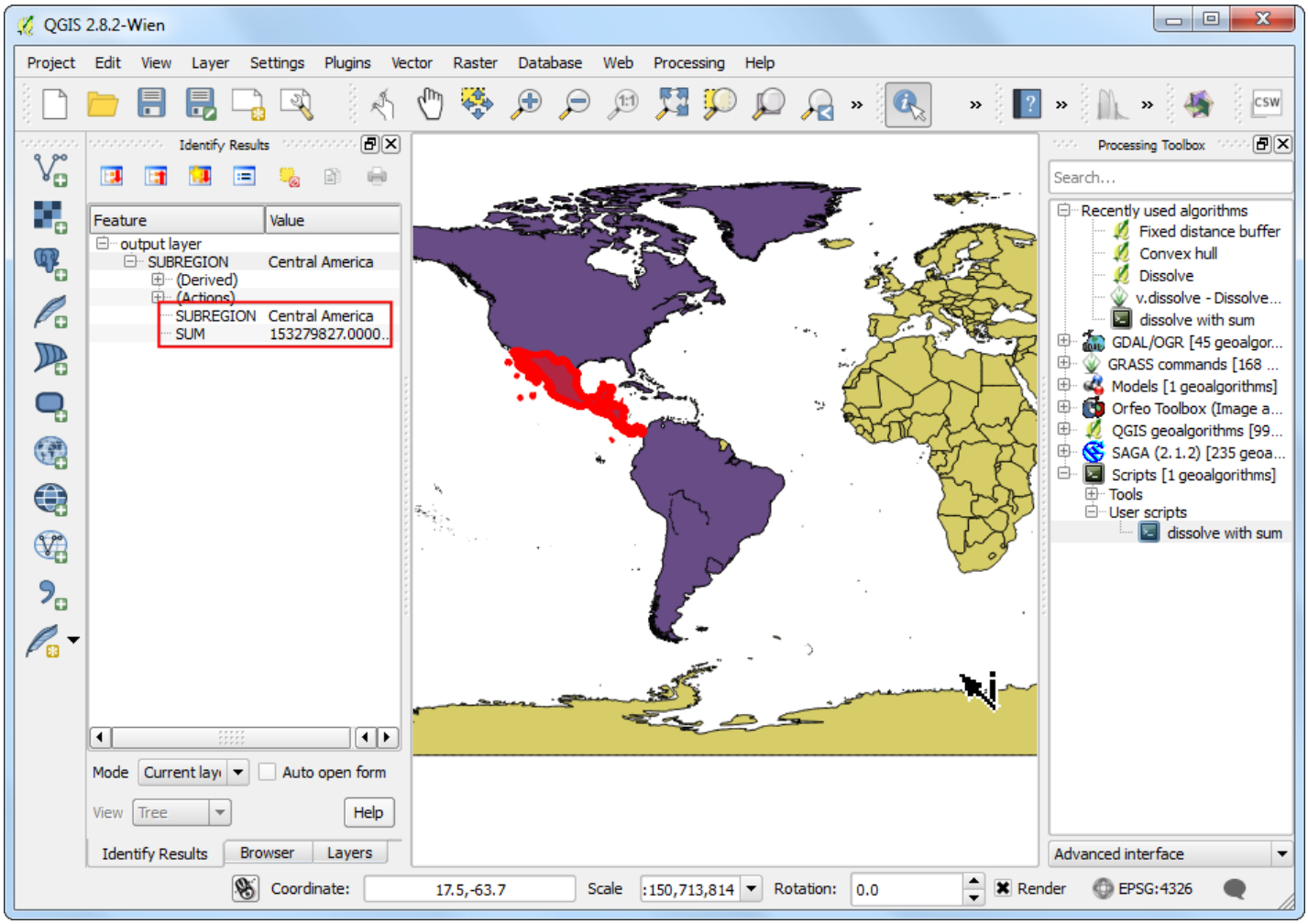

19. You will see the selected features highlighted in yellow. Right-click the `dissolve_with_sum` script and select Execute.



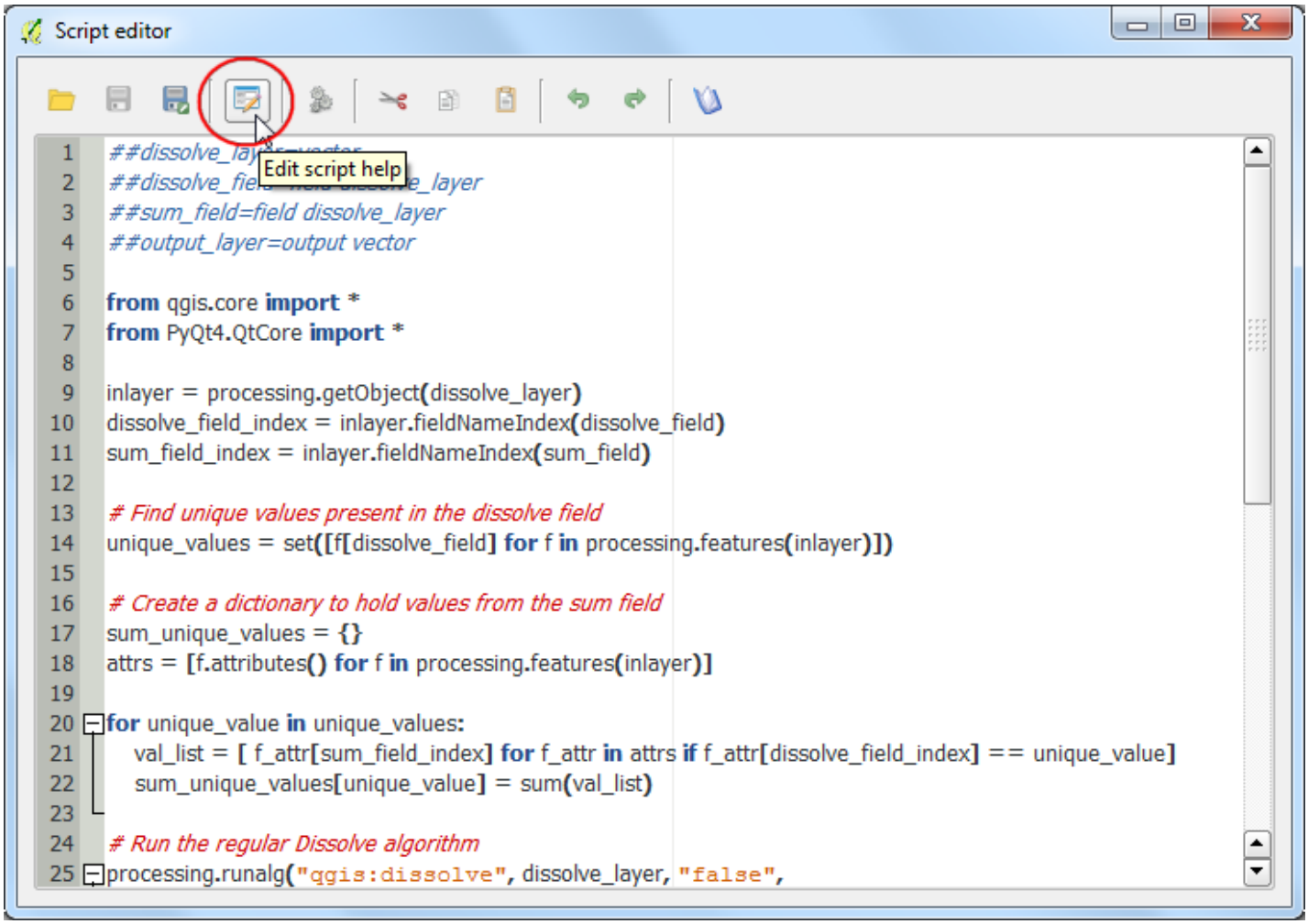

20. Select the inputs as before and click Run.



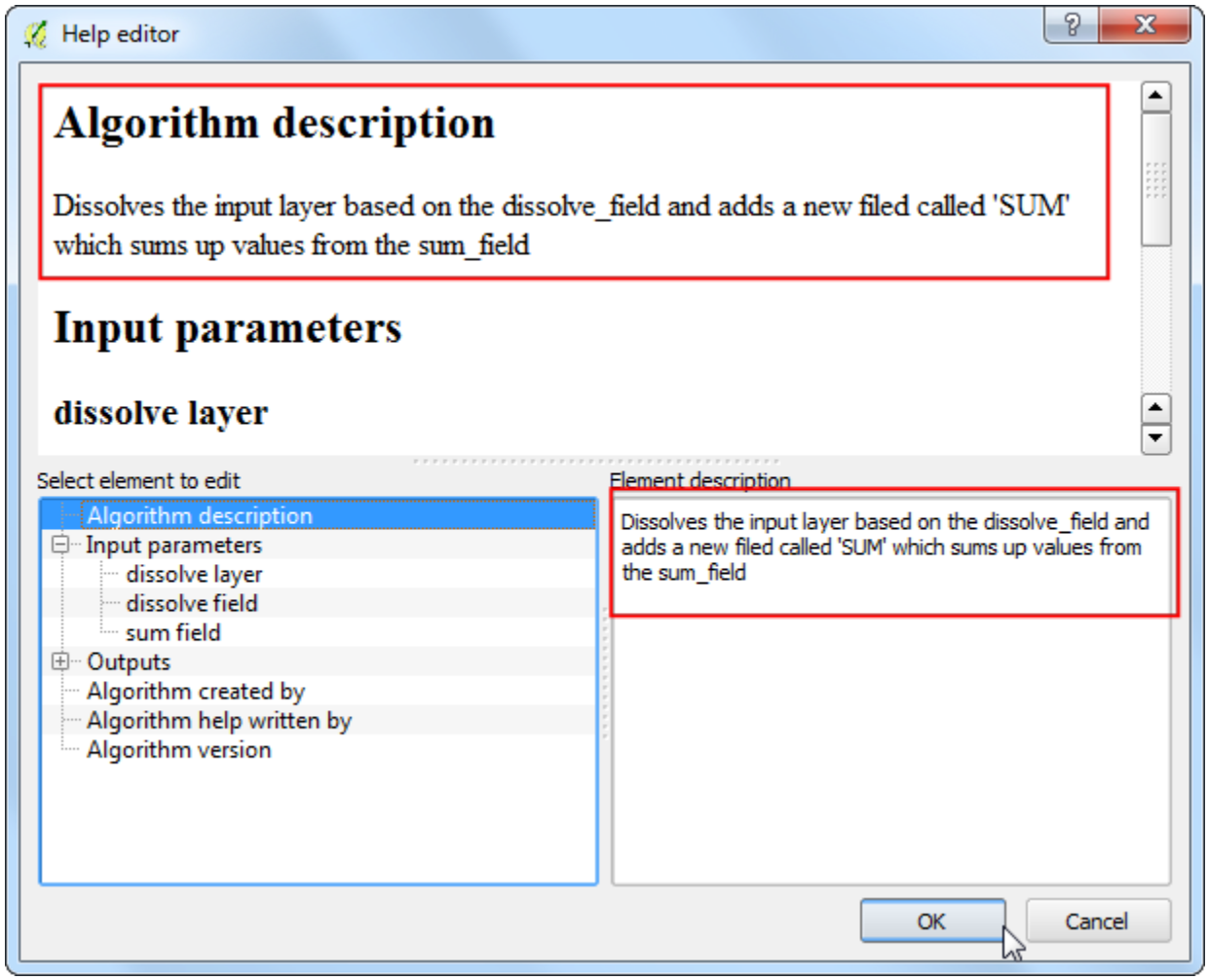

22. A new `output layer` will be added to QGIS. This will contain dissolved geometries only from the selected features in the input layer. You will also note that the `output layer` will contain only 2 fields as expected.




23. One last but important remmaining work is to document our algorithm. The Processing Framework has nice tools to write and access help. Go to the Script editor and click the Edit script help button.




24. Fill in the details for different elements and click OK. Now a detailed help will be available to all users of your script in the Help tab when they launch the algorithm.




Below is the complete script for reference. You may modify it to suit your needs.

```
##dissolve_layer=vector
##dissolve_field=field dissolve_layer
##sum_field=field dissolve_layer
##output_layer=output vector

from qgis.core import *
from PyQt4.QtCore import *

inlayer = processing.getObject(dissolve_layer)
dissolve_field_index = inlayer.fieldNameIndex(dissolve_field)
sum_field_index = inlayer.fieldNameIndex(sum_field)

# Find unique values present in the dissolve field
unique_values = set([f[dissolve_field] for f in processing.features(inlayer)])

# Create a dictionary to hold values from the sum field
sum_unique_values = {}
attrs = [f.attributes() for f in processing.features(inlayer)]

for unique_value in unique_values:
    val_list = [ f_attr[sum_field_index] for f_attr in attrs if f_attr[dissolve_field_index] =
    sum_unique_values[unique_value] = sum(val_list)
```

```
# Run the regular Dissolve algorithm
processing.runalg("qgis:dissolve", dissolve_layer, "false",
    dissolve_field, output_layer)

# Add a new attribute called 'SUM' in the output layer
outlayer = processing.getObject(output_layer)
provider = outlayer.dataProvider()
provider.addAttributes([QgsField('SUM', QVariant.Double)])
outlayer.updateFields()

# Set the value of the 'SUM' field for each feature
outlayer.startEditing()
new_field_index = outlayer.fieldNameIndex('SUM')
for f in processing.features(outlayer):
  outlayer.changeAttributeValue(f.id(), new_field_index, sum_unique_values[f[dissolve_field]])
outlayer.commitChanges()

# Delete all fields except dissolve field and the newly created 'SUM' field
outlayer.startEditing()
fields_to_delete = [fid for fid in range(len(provider.fields())) if fid != new_field_index and
provider.deleteAttributes(fields_to_delete)
outlayer.updateFields()
outlayer.commitChanges()
```