

# Further Work – Producing an Atlas

**Objective – To learn how to create an Atlas.**

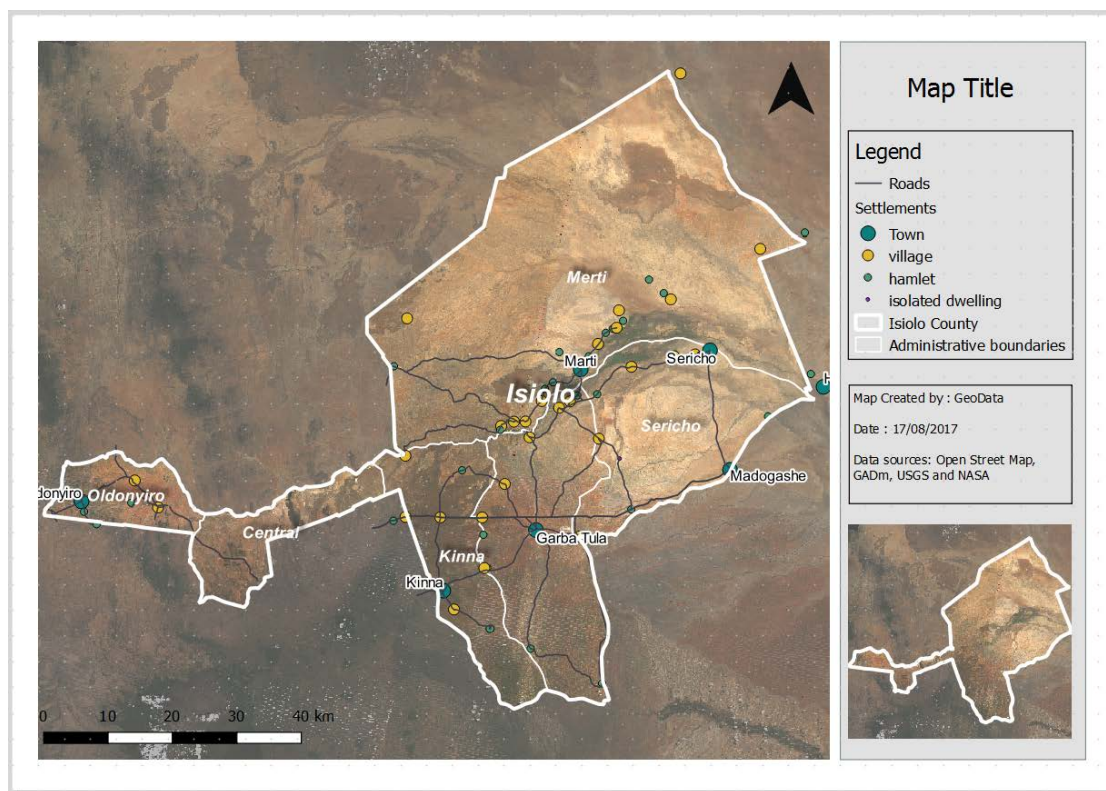
The data we are using for this exercise is for Isiolo county, Kenya.

- Open the following QGIS project:

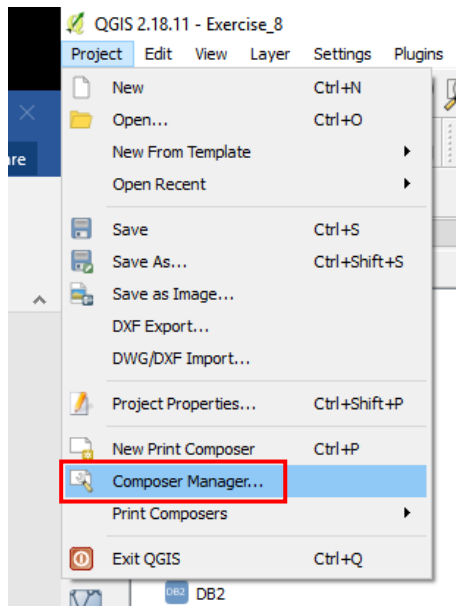
*C:\Intro\_Quantum\_GIS\Exercises\Further Work\FW\_Atlas.qgs*

## Step 1 – How to create an Atlas

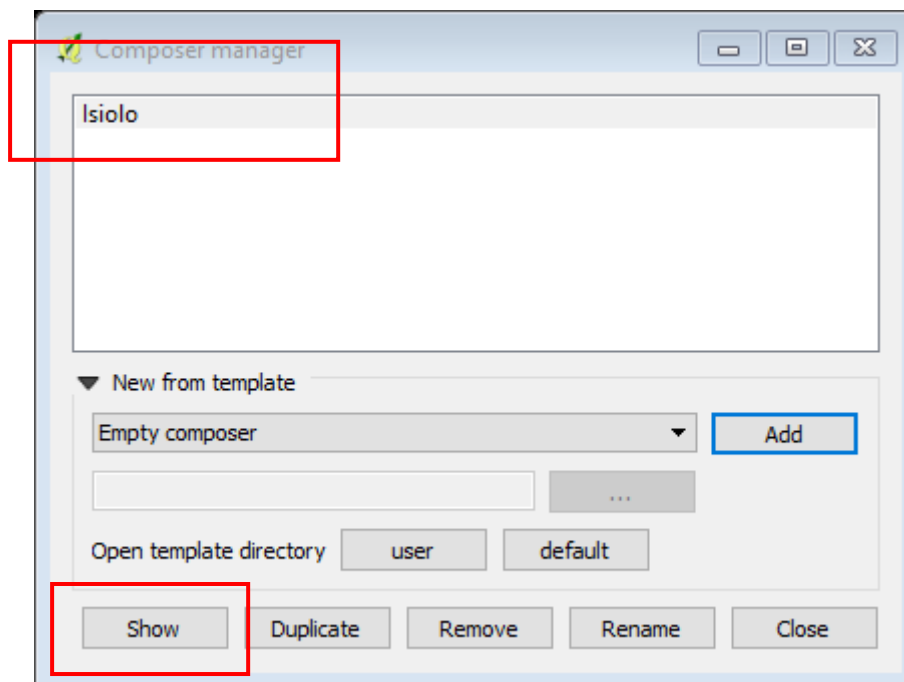
- The map project should appear similar to below

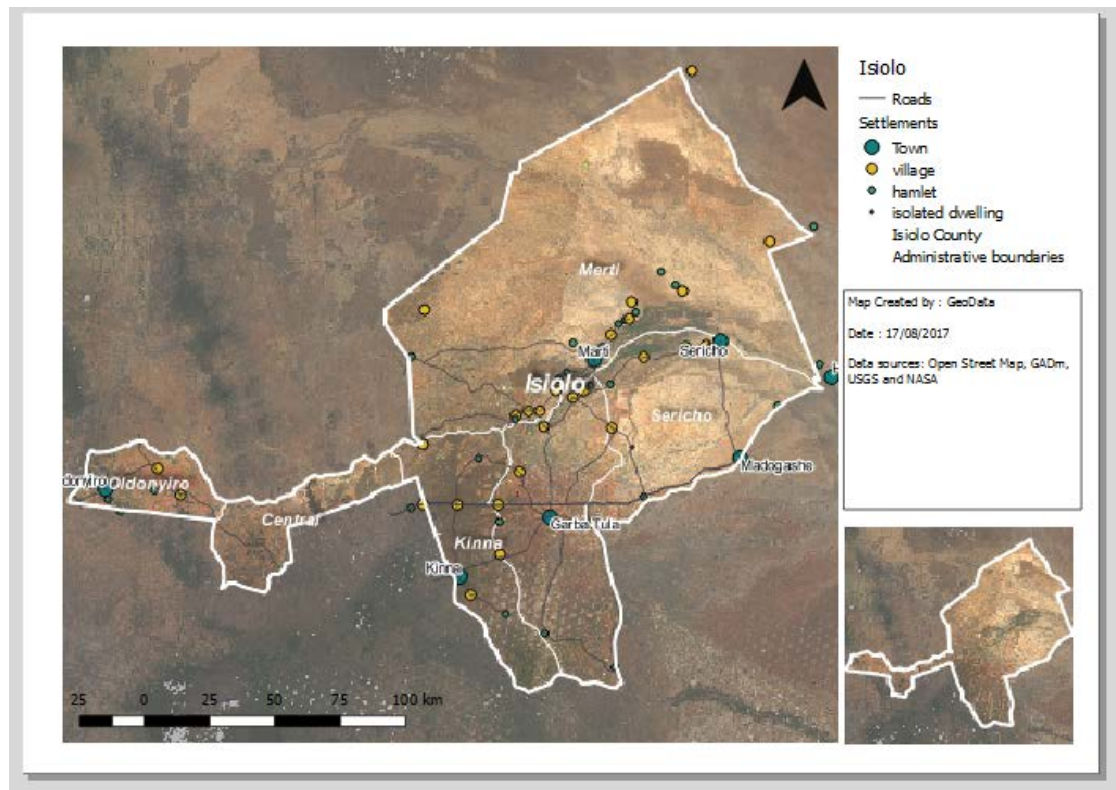


- Click on Project → Composer Manager as shown below:

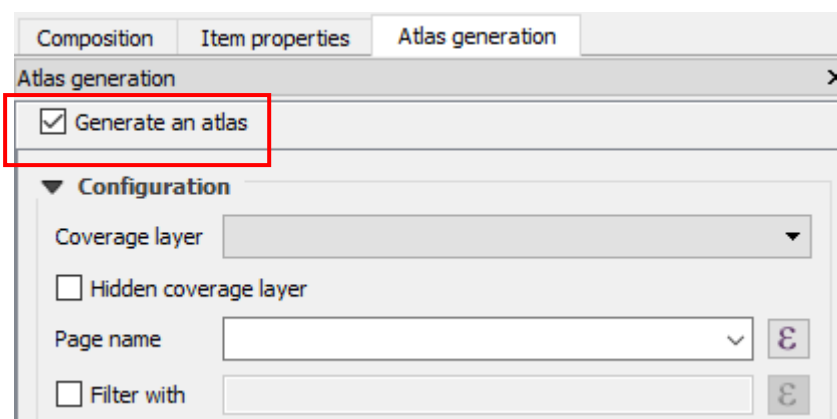


- Select **"Isiolo"** from the Composer manager dialog box and click **Show** as shown below:

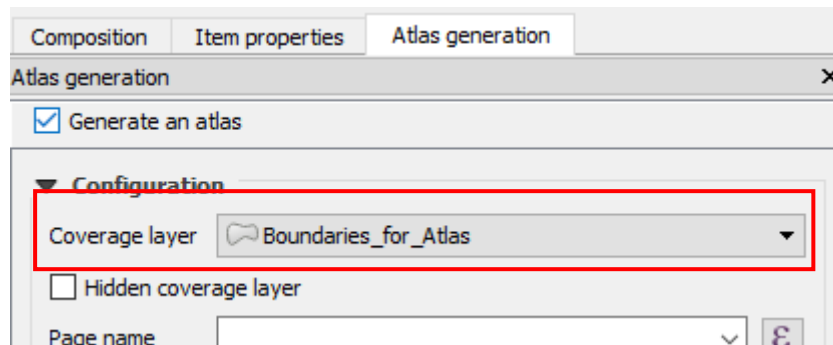




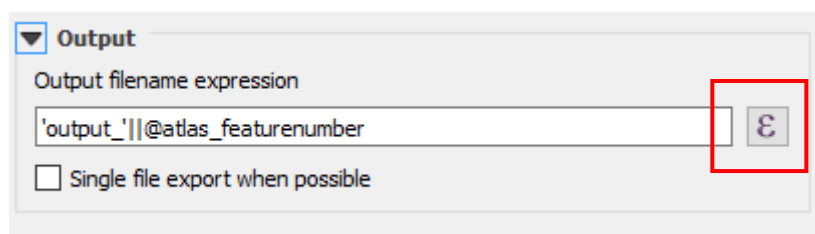
- In this exercise we are going to create an Atlas showing a different map extent for each of the Administrative areas in Isiolo. This will create 6 output maps: Oldonyiro, Central, Kinna, Garba Tulla, Sericho and Merti.
- Firstly we will set up the Atlas and then configure the legend and title.
- To set up the Atlas click onto the Atlas generation tab on the right hand side of the composer window as shown below:



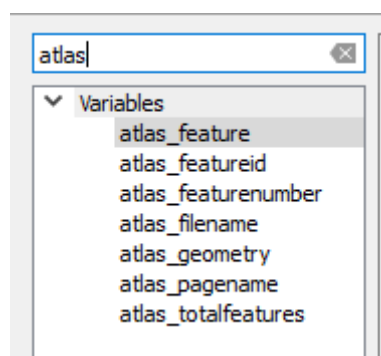
- Check “Generate an atlas” checkbox as highlighted above.
- Within the Configuration section we need to set the Coverage layer. This is the layer in the map that will drive the Atlas extents. In this exercise this will be **Boundaries\_for\_Atlas** polygon layer. From the Coverage layer drop down menu select **Boundaries\_for\_Atlas** as shown below:



- There are further options within the Configuration section to drive the atlas with a hidden layer and to filter the coverage layer. For example, if you have a settlements coverage layer where you only wish to create maps for just Towns and not Villages. An expression could be written to query out just the Towns.
- The Output section allows you to define names for your output filename. As default this is 'output\_' || \$feature. If this remained unchanged and we exported our 6 Administrative area maps the output names would be: output\_1.pdf, output\_2.pdf and output\_3.pdf. These names are not very useful as they do not describe the maps. A better output name would Sericho.pdf, Kinna.pdf etc. To achieve this we need to edit the expression.
- Click onto the expression button the right of the current expression box as highlighted below:



- An Expression based filename dialog box will pop up.
- Within the Functions tree search Atlas as shown below:



- Each of the functions in this list are prewritten specifically to hook into the Atlas features.

Function Name	Description
atlasfeature	Returns the current Atlas feature that is iterated. Can be used with the 'attribute' function to return attribute values from the current atlas feature.
atlasfeatureid	Returns the feature id of the current row while using atlas. Enables the use of features of the atlas in rules e.g to match with other layers in the map.
atlas_featurenumber	Returns the current feature number that is iterated over in the Atlas.
atlas_filename	Returns the current atlas file name
atlasgeometry	Returns the geometry of the current Atlas feature that is iterated. Enables the use of Atlas geometries in rules e.g only show geometries of other layers when their geometry intersects the iterated Atlas feature.
atlas_pagename	Returns the current feature number that is iterated over in the Atlas.
atlas_totalfeatures	Returns the total number of features within the coverage layer.

- Before we are able to write an expression we need to know which field in the Boundaries\_for\_atlas attribute table contains the name of the Administrative boundary. Go back into the main QGIS window and identify the field containing the names of the Administrative boundaries (It should be admin\_level or name – in this instance we will use **name**).
- From the previous table showing the list of Atlas functions we can identify the function we need to use to extract the value within the **name** field. This is the atlasfeature function.
- Navigate back into the Map Composer window.

- Click onto the expression button within the Output section within the Atlas generation as shown below:

- Within the pop up Expression based filename window, delete the current expression.
- The function we need to use is called **attribute**. This can be found within the **Record** section of the function tree.
- Find the **attribute** function either by searching for it or by expanding the **Record** section. Select the **attribute** function and explore functions help section.

**function attribute**

Returns the value of a specified attribute from a feature.

**Syntax**

```
attribute(feature, attribute_name)
```

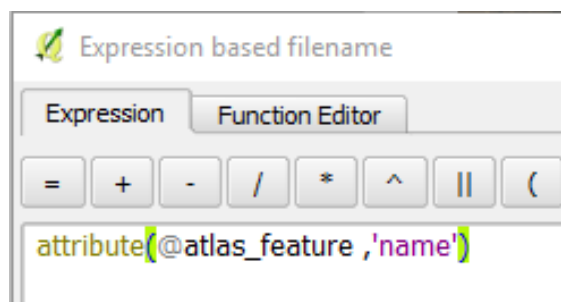
**Arguments**

*feature* a feature  
*attribute\_name* name of attribute to be returned

**Examples**

- `attribute( $currentfeature, 'name' )` → value stored in 'name' attribute for the current feature

- We can see that the **attribute** function requires two arguments to work. In this case we want the feature to come from our atlas
- Double click on the **attribute** function to add it into the expression builder.
- From the table above we know that we need to use the **atlas\_feature** function to return the feature (Boundaries\_for\_Atlas) that is being used to generate our atlas.
- Double click on the **atlas\_feature** function to add it into the expression builder.
- We want to use the 'name' attribute from the Boundaries\_for\_Atlas feature to be used in our output filename.
- This will need to be manually typed into the expression builder.
- The final, correct expression is as follows:

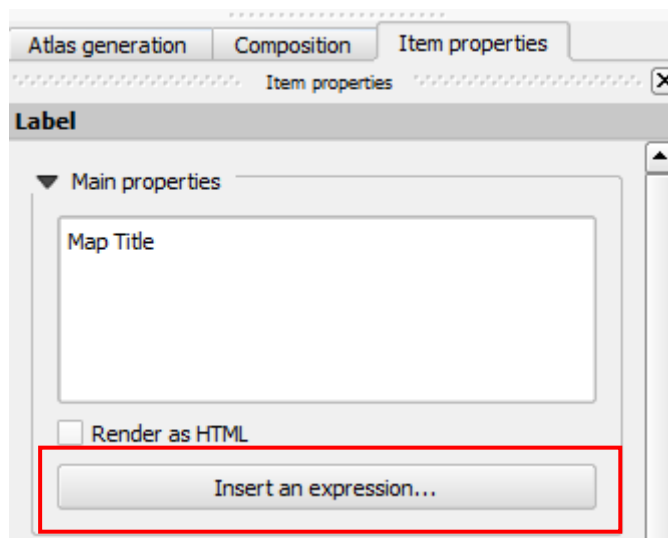


- The output maps will now be named after each of the respective administrative areas.
- This same function can be used to set up a title for the map that reflects the iterated administrative area
- Click onto the Map Title (top right hand corner of the map) as shown below;

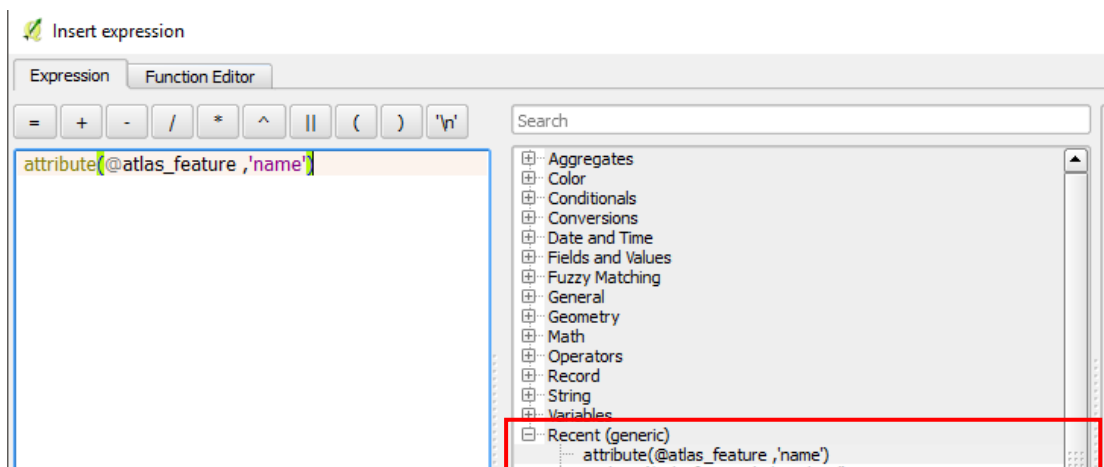


- Within the Item properties tab (on the left) click on the **Insert an expression** button as shown below:



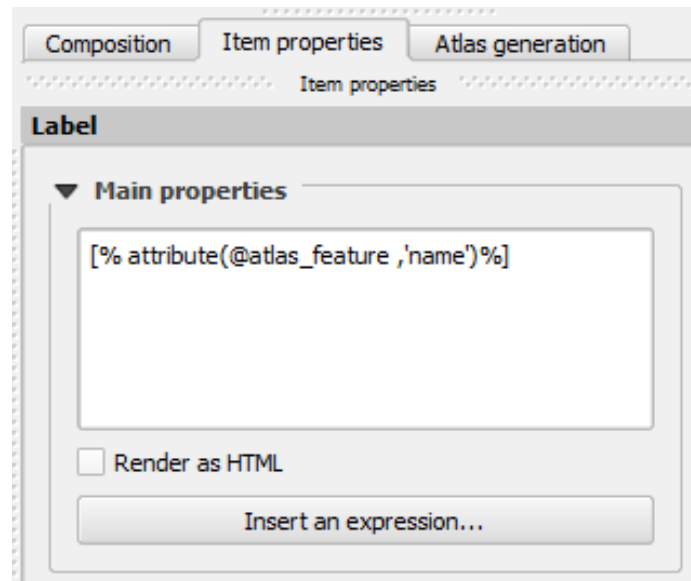


- The Insert expression dialog box should pop up.
- Within the Functions tree expand the Recent (generic) section. This shows the recent expression to alter the output file name of the maps. Double click on this expression to move it into the main expression window.

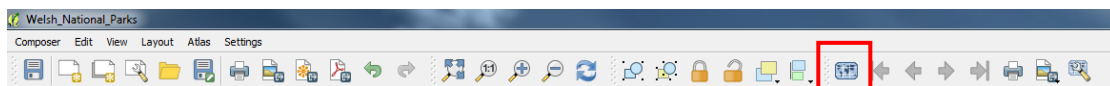


- Click OK to close the expression window.
- The expression should now appear in the box underneath the Main Properties window. The original Map Title text is also in the box. Delete the 'Map Title' so the box appears as shown below:





- The Atlas is ready for preview.
- Click onto the Preview Atlas button along the top tool button bar as shown below



- The first page of the Atlas should appear showing the title of the Kinna as shown below:



- You may notice that the map window does not change extent to zoom into the administrative area boundary.
- Using the arrows on the Atlas toolbar click forward to preview the next page of the Atlas as shown below:

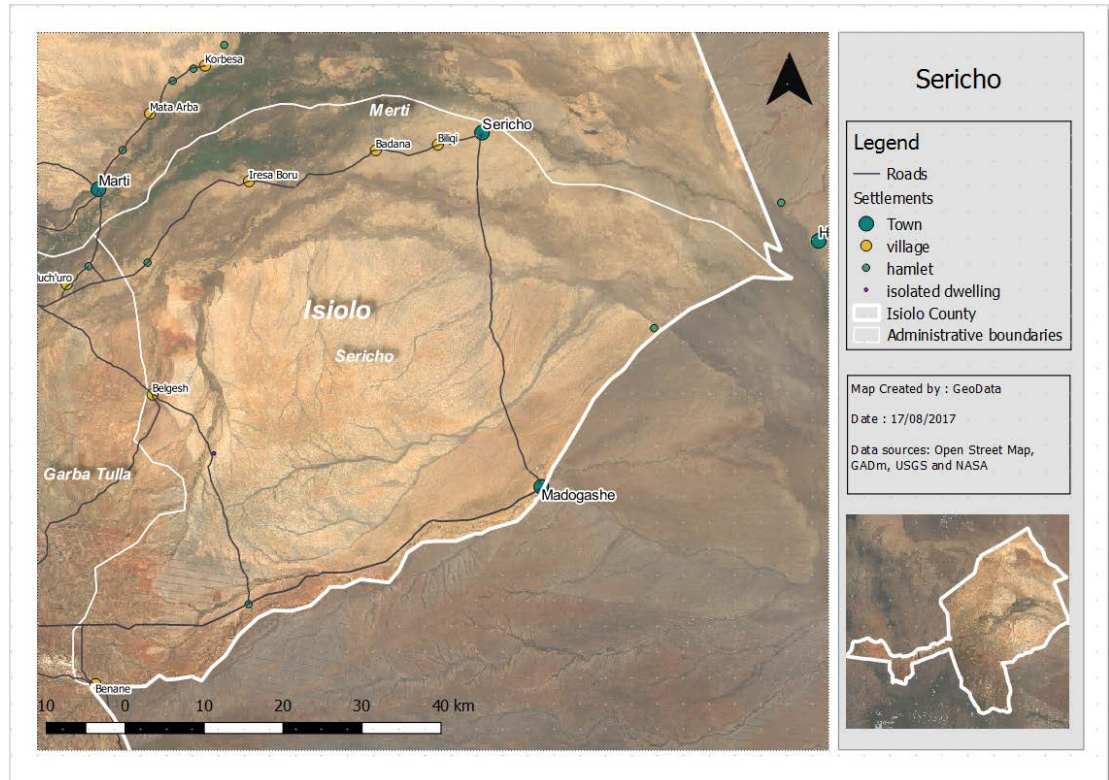


- The next page is Central. You will notice that the text changes but the extent does not change.

- Click onto the Map within the composer window to activate the Item Properties for the map. The Controlled by atlas checkbox needs to be ticked to allow the extent of the map to be driven by the Coverage layer as shown below:

The screenshot shows the 'Item properties' window for 'Map 0'. The 'Controlled by atlas' section at the bottom is highlighted with a red box. It contains a checked checkbox 'Controlled by atlas' and three radio button options: 'Margin around feature' (selected), 'Predefined scale (best fit)', and 'Fixed scale'. The 'Margin around feature' option has a value of '10%' next to it.

- As soon as this check box is ticked the extent of the map should change to show the extent of the current atlas feature.



- Return to preview the pages of the Atlas using the Atlas toolbar. The extents should change for each page to show the iterated Atlas National Park.
- You may notice that the size of the scale bar also now automatically adjusts, if you have left the scale bar segments settings to 'Fixed width' it may look inappropriate. Try changing the setting to 'Fit segment width' and experimenting with the sizes to find a scale bar that is appropriate for all pages of the atlas

▼ Segments

Segments

left 0

right 4

30000.000000 units

min 50 mm

max 100 mm

Height

3 mm

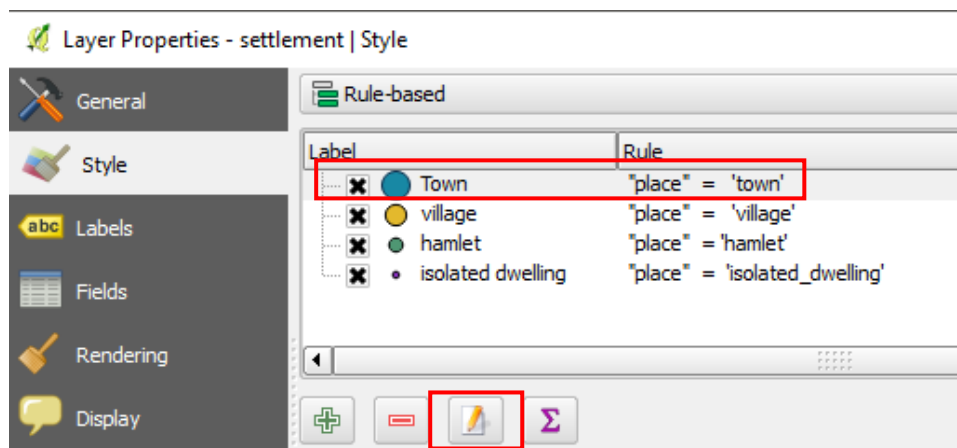
☐ Fixed width
 ☒ Fit segment width

### Further Atlas Features

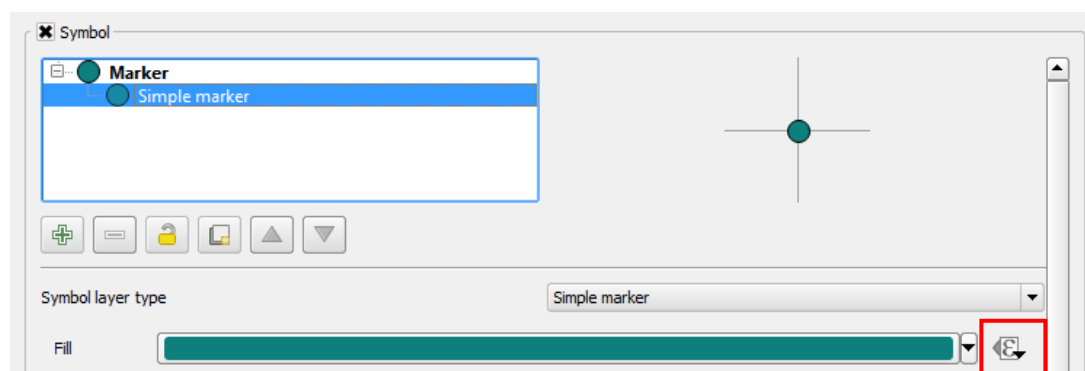
The Atlas functions can be used to link to features and labels in the layers within your map and alter their appearance based on the iterated Atlas feature.

The next example we are going to look at is changing the colour of those town that intersect administrative boundaries. Those towns that intersect an administrative boundary will be keep their current symbology whilst those outside will be coloured grey.

- Navigate back into the Main QGIS window.
- Open the Layer Properties for the settlement layer.
- Click onto the Style tab.
- Edit the rule for 'Town' by double clicking on Town or selecting town then clicking the edit rule button

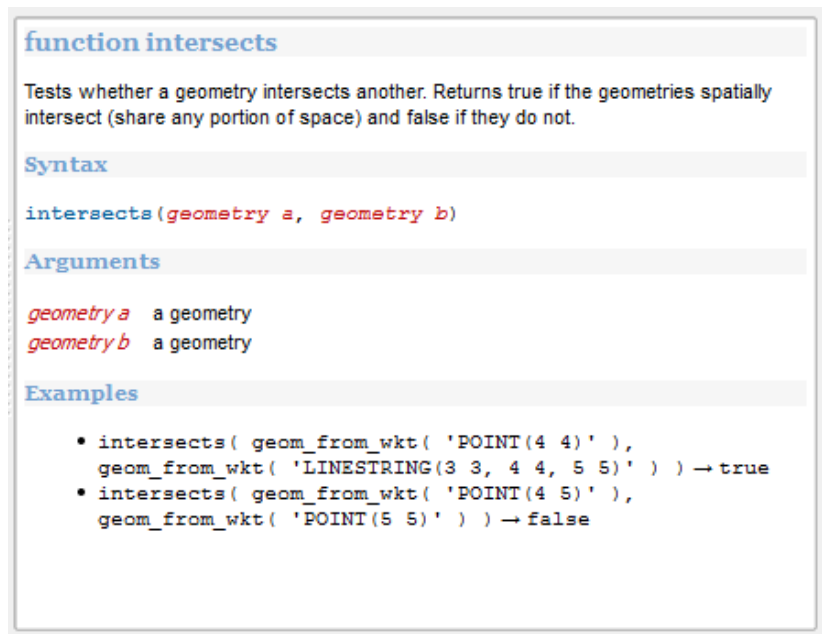


- From the subsequent menu highlight 'Simple marker' within the symbol window
- To the right of Fill click onto the data driven button as highlighted below and click onto **Edit** from the drop down menu:



- The Expression string builder should pop up.

- Within the Functions tree expand the **Geometry** section and click onto the **intersects** function. Explore the help as shown below:



- We can see that the **intersects** function takes two geometries as arguments.
- In this case we want to test whether the geometry of our **settlement feature** intersects the geometry of the feature that is being used to **construct our atlas (Boundaries\_for\_Atlas)**
- Add the **intersects** function into the expression builder.
- We now need to enter the two geometries into the expression as arguments for the **intersects** function.
- Explore the functions tree and use the function help to find the two required functions (Hint: the function for the atlas feature geometry can be found in the variable section; settlements geometry can be found within the Geometry section)
- Because we want to change the colour of the villages based on their location to either be red if they intersect or grey if they do not, we need to create a case statement using the **color\_rgb** function found within the **color** section of the function tree. We explored case statements in Exercise 6a.
- The rgb values for red are 255,0,0 and the rgb values for grey are 204,204,204.
- Try and build the case statement using the **intersects** function.

Answer:

```
case
when intersects(@atlas_geometry, $geometry )
then
color_rgb(255,0,0)
else
color_rgb(204,204,204)
end
```

Test the output in the composer window by scrolling through the Atlas preview.