Introduction to Python for Social Science

Lecture 4 - Data Visualization

Musashi Harukawa, DPIR

4th Week Hilary 2020

Overview

Last Week

- Advanced Data Operations
  - Applying Functions to Vectors and Matrices
  - Grouped Summaries
  - Concatenating and Merging Data

This Week

This week we finally get to a fun topic: **data visualisation**.

There's more to data visualisation than I could possibly cover in 90 minutes, so I focus on *static*, *two-dimensional* visuals; these are the kind that you are most likely to use.

Theory & Motivation

Visual Summaries as an Aid

- Returning to the theme of this course, the aim of much of data science is to understand the whole picture of your data.
- If you can do this without reading your entire dataset, all the better!
- When making data visuals, I think it's helpful to remember that they are, in many ways, a form of summary.
- Visualising data is not just about communicating results; it is also a powerful tool for you to understand important features of your own data.

Motivating Example

Motivating Example

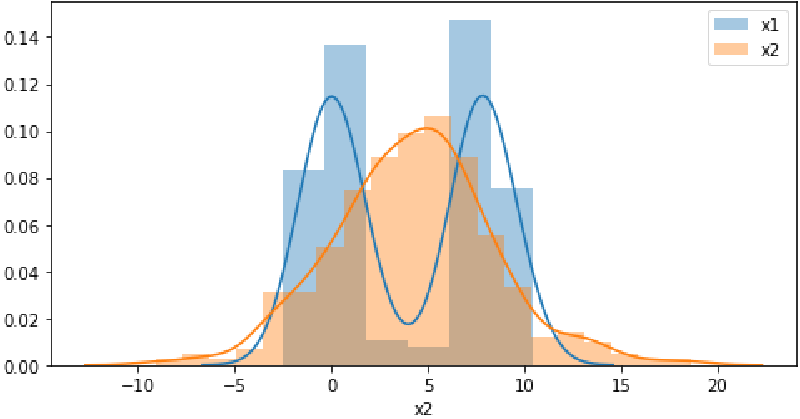


Figure 1: The same data

From Data Types and Structures to Visualisation

Data Types/Structures

The type and structure of your data tells you what *type* of figure you need:

- Number of Dimensions
- Ordered or Unordered?
- Discrete or Continuous?

Visuals on a Two-Dimensional Medium

Most figures are created on a two-dimensional plane, where the dimensions are usually referred to as X (width) and Y (height).

These axes are the most versatile; they can be used to plot any kind of variable. The only tradeoff is the overall size of the figure is determined by these two dimensions.

One-Dimension: Distributions

Visuals for one-dimensional data tend to be concerned with *distributions*; i.e. frequencies of values along some dimension.

Useful plots include:

- Histogram
- Box (and whiskers) plot
- Swarm plot
- Violin plot

Two-Dimensions: Relationships

Visuals for two-dimensional data often fulfil one of the following two purposes:

- Comparing distributions
- Plotting functions

In addition to all of the aforementioned plots, some examples of the latter include:

- Scatter plot
- Line plot
- Bar plot

Three-Dimensions and Higher: Levels

- While it is possible to draw plots that have a third, z axis, to show depth on a screen, I personally do not think it is very readable.
- There are many ways to vary visual elements to intuitively convey variation along further axes.

Showing Variation with Color

Colors can show variation along a multitude of data types.

- Discrete colors can differentiate unordered, discrete categories.
- Gradiated colors can represent ordered, continuous variation.
    - For example, see heatmaps

Showing Variation with Panels

Panelling is the use of multiple sub-plots within a single figure.

- Panelling can only show variation along a discrete variable.
- The order of the plots can be used to show variation along an ordered, discrete variable.

Other Ways of Showing Variation

Colors and panelling are not the only means.

- Shapes can be used to show categorical variation.
- Size/thickness and transparency can be used to show continuous variation.

Take-Away

When visualising data, ask yourself the following questions, then look through galleries to get an idea of what could work for you.

Are you:

- Making a comparison between groups?
- Trying to show conditional relationships between variables?
- Exploring your own data?

Implementation

Two Libraries

- `matplotlib` is the primary library for building data-based visuals in Python.
  - Requires a lot of explicit commands to get it to look good, but allows for nearly complete customisation of all aspects.
- `seaborn` is a more recent library, built on top of `matplotlib`.
  - Provides fast and convenient methods for most figures you will ever need.
- Both libraries can be used in conjunction.

The Anatomy of a Data Visual

On the back end, all `matplotlib`-based visuals adhere to a similar *tree-like* structure. By learning this structure, you can locate and customise any element of a figure.

The Matplotlib Hierarchy

Here is a truncated version of the matplotlib hierarchy:

Figure

The figure is essentially the “canvas” upon which all visuals are made. Some parameters/methods set at this level include:

- Total size (in pixels)
- Super-title
- Saving to file

Axes (Subplots)

Subplots are the frames within which individual visuals are contained.

Most drawing methods are called at the subplot level:

- Plotting (drawing the graphical objects)
- Individual plot labels
- Legends

Graphical Functions

`matplotlib` and `seaborn` provide an enormous number of plotting functions. These functions:

- Take one or more equal-length vectors as inputs (the data).
  - This data may be in long- or wide-format.
- Draw objects accordingly to the relevant subplot
  - If the function is a `matplotlib` function, you should call it as a method of the relevant subplot.
  - If the function is a `seaborn` function, and there is more than one subplot, then you should pass the relevant subplot as a parameter to the function.

Customisable Aspects

Graphical objects take a large number of customisable parameters, such as:

- Color
- Transparency
- Line/dot style

X and Y Axis

Subplots have `xaxis` and `yaxis` methods. Call these to customise the following aspects:

- Ticks (the little notches along the axes)
  - Spacing/interval
  - Labels
    - Text
    - Orientation
  - Top/Bottom, Left/Right
- Axis labels

Anatomy, Again, with Examples

Figure

```
import matplotlib.pyplot as plt
f = plt.figure(figsize=(15, 8))
```

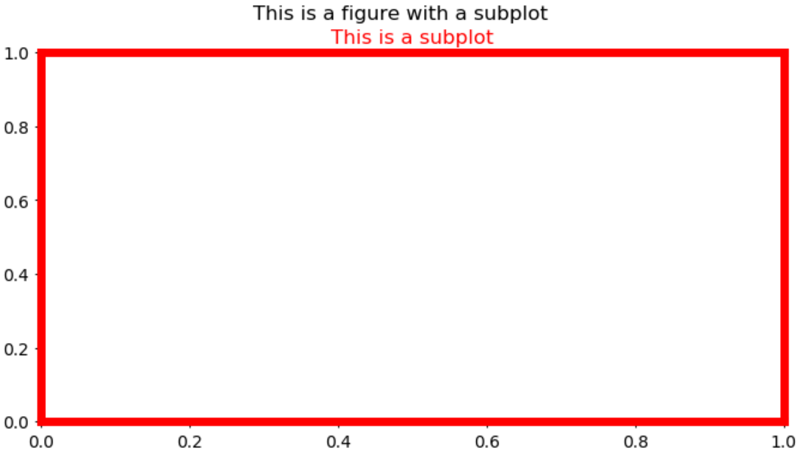
This does not create any visible objects, but it lays down the canvas that other things will go onto.

Note:

- Most of the plotting functionality is within the `pyplot` module of `matplotlib`.
- The output of `plt.figure` has been assigned to a variable, `f`. This will be our means of accessing the figure and its methods.
- The parameter `figsize=(15, 8)` has been passed to `plt.figure`. This tells `matplotlib` to create a canvas that is 1500x800 pixels.

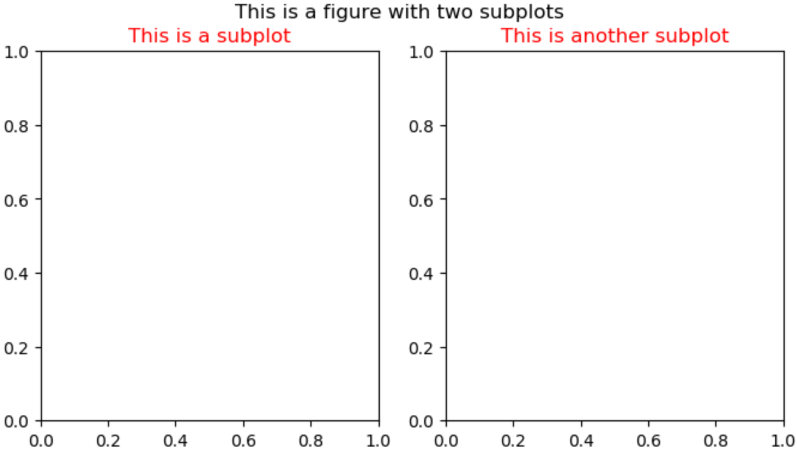
Axes (One Subplot)

```
f, ax = plt.subplots(1, 1, figsize=(15, 8))
f.suptitle("This is a figure with a subplot")
ax.set_title("This is a subplot", color="r")
```



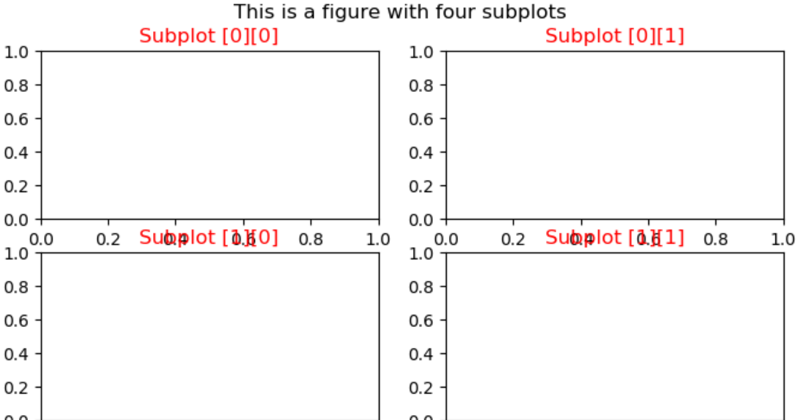
Axes (Two Subplots)

```
f, ax = plt.subplots(1, 2, figsize=(15, 8))
f.suptitle("This is a figure with two subplots")
ax[0].set_title("This is a subplot", color="r")
ax[1].set_title("This is another subplot", color="r")
```



Axes (Subplot Grid System)

```
f, ax = plt.subplots(2, 2, figsize=(15, 8))
f.suptitle("This is a figure with four subplots")
for i in range(2):
    for j in range(2):
        ax[i][j].set_title(f"Subplot [{i}][{j}]", color="r'
```



Graphical Functions (Scatter)

```
f, ax = plt.subplots(1, 1, figsize=(15, 8))
ax.scatter(data['x2'], data['x1'], color='r')
```

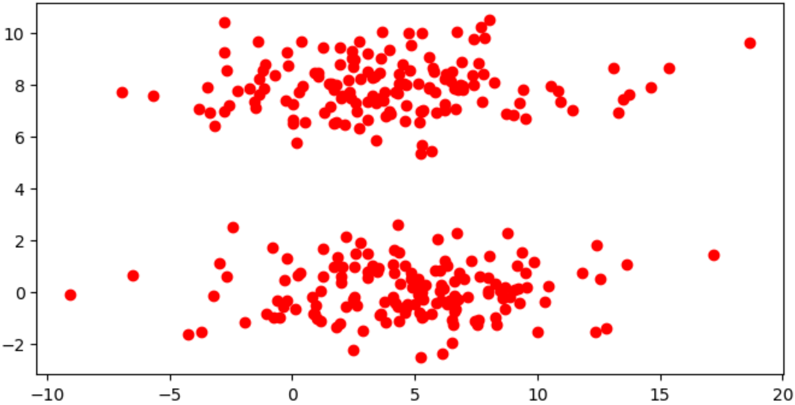


Figure 5: Scatter Plot

Graphical Functions (Line)

```
f, ax = plt.subplots(1, 1, figsize=(15, 8))
ax.plot(np.linspace(0, 10, 100), np.linspace(0, 5, 100), co
```

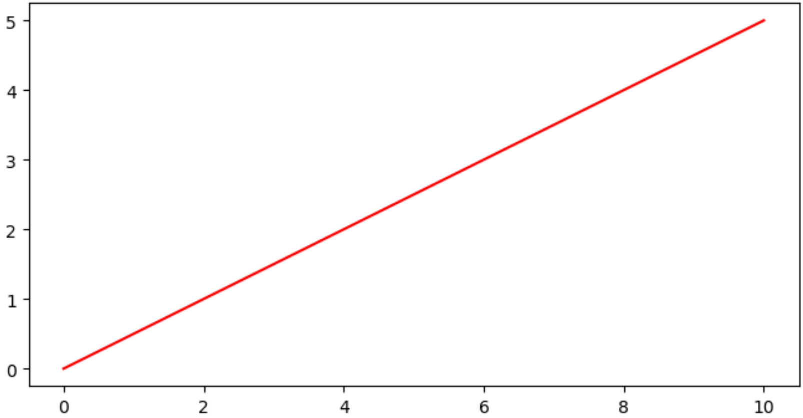


Figure 6: Line Plot

Combining Graphical Functions (Scatter + Line)

```
f, ax = plt.subplots(1, 1, figsize=(8, 4))
ax.scatter(data['x2'], data['x1'], color='r', s=3)
ax.plot(np.linspace(-10, 20, 150), np.linspace(-3.5, 4, 150
ax.axhline(0, color='k', alpha=0.5, ls="--")
ax.axvline(0, color='k', alpha=0.5, ls="--")
```

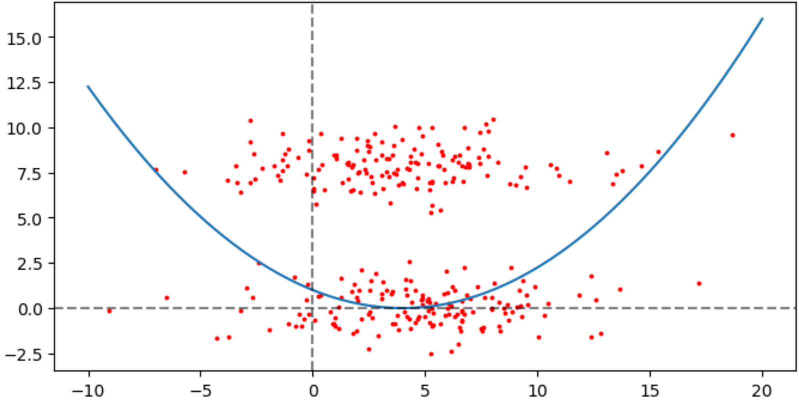


Figure 7: Line and Scatter Plot

Adding Axis Labels

```
[...]
ax.xaxis.set_label_text("X-Axis Label", color='r')
ax.yaxis.set_label_text("Y-Axis Label", color='r')
```

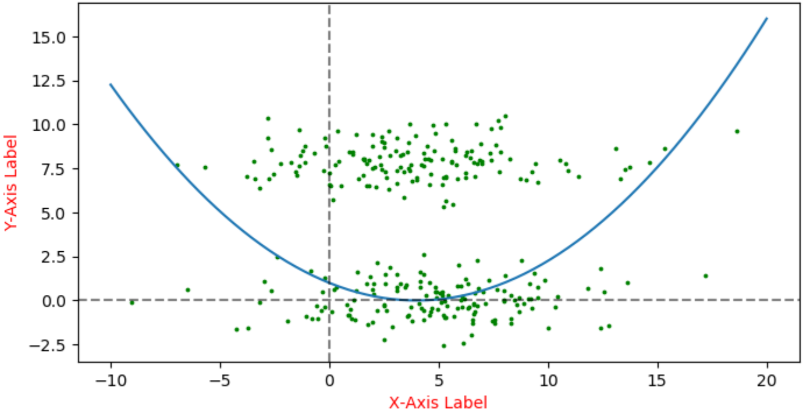


Figure 8: Custom Axis Labels

Customising Tick Locations (Manual)

```
[...]
ax.xaxis.set_ticks(range(-10, 40, 10))
ax.yaxis.set_ticks(range(-4, 25, 2))
```

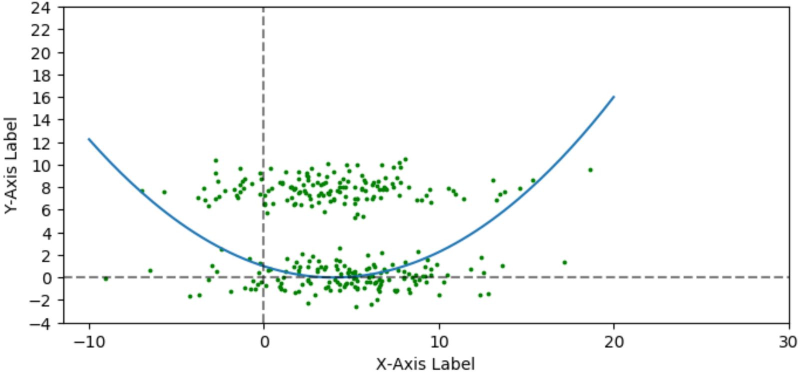


Figure 9: Manually Adjusted Ticks

Customising Tick Locations (Automatic)

```
ax.xaxis.set_major_locator(matplotlib.ticker.MultipleLocato
ax.yaxis.set_major_locator(matplotlib.ticker.MultipleLocato
```

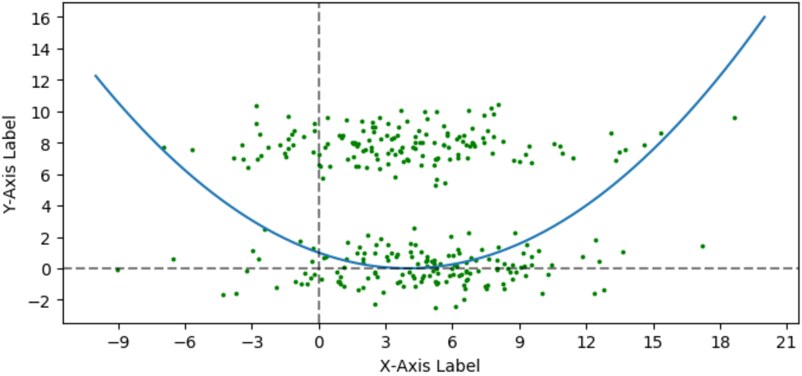


Figure 10: Manually Adjusted Ticks

Customising Tick Labels/Orientation

```
f, ax = plt.subplots(1, 1, figsize=(8, 4))
sns.boxenplot(bes_df['region'], bes_df['Age'], ax=ax)
ax.xaxis.set_ticklabels(ax.xaxis.get_ticklabels(), rotation
```

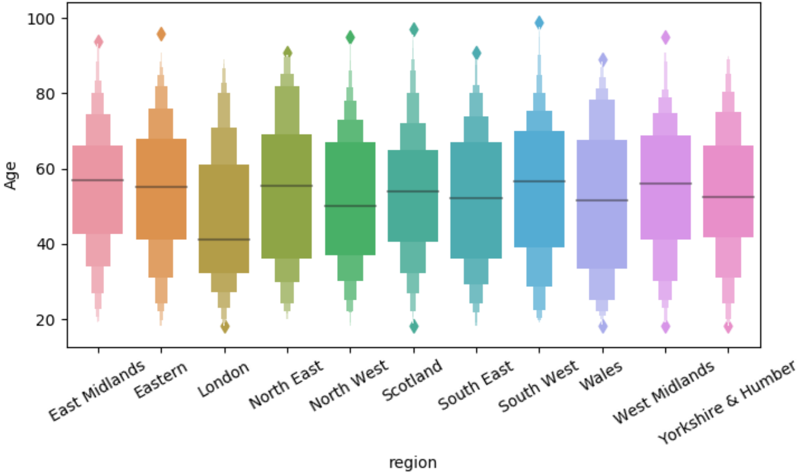
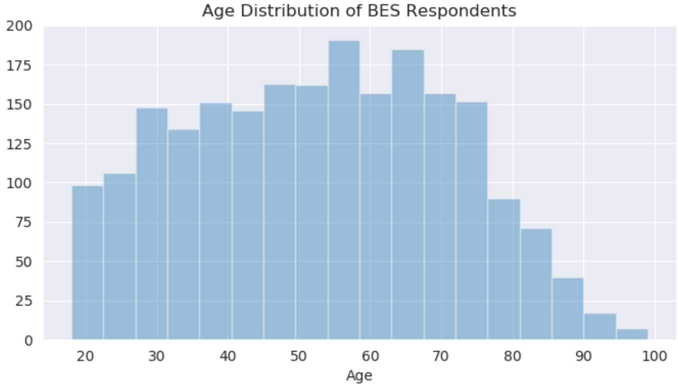


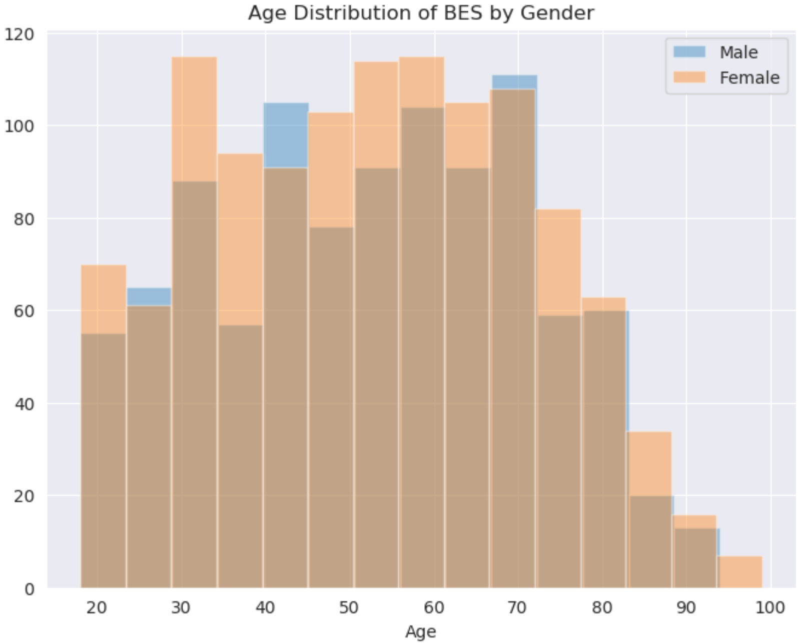
Figure 11: Rotated Tick Labels

Gallery

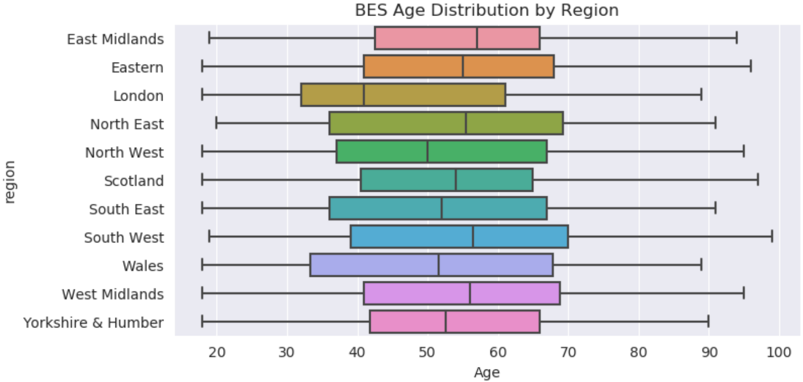
Histogram (One Category)



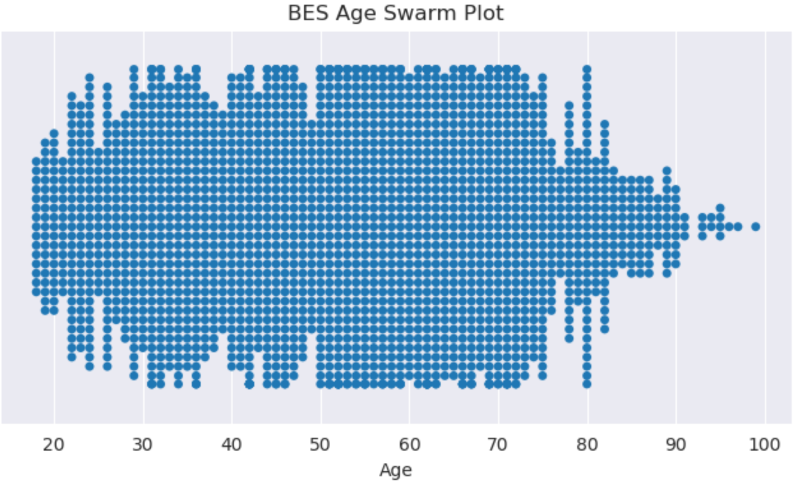
Histogram (Two Categories)



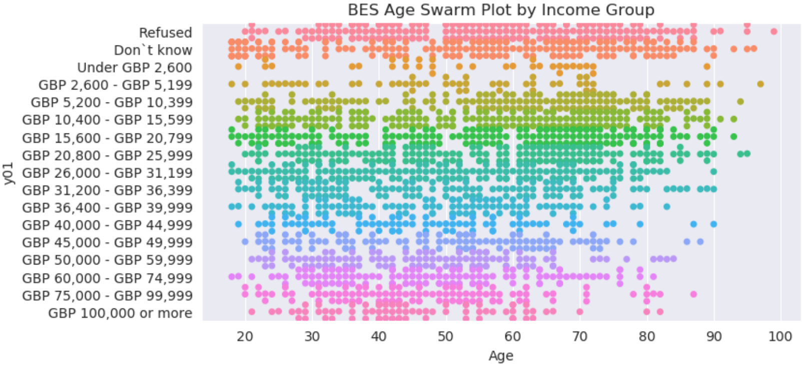
Box and Whisker Plot



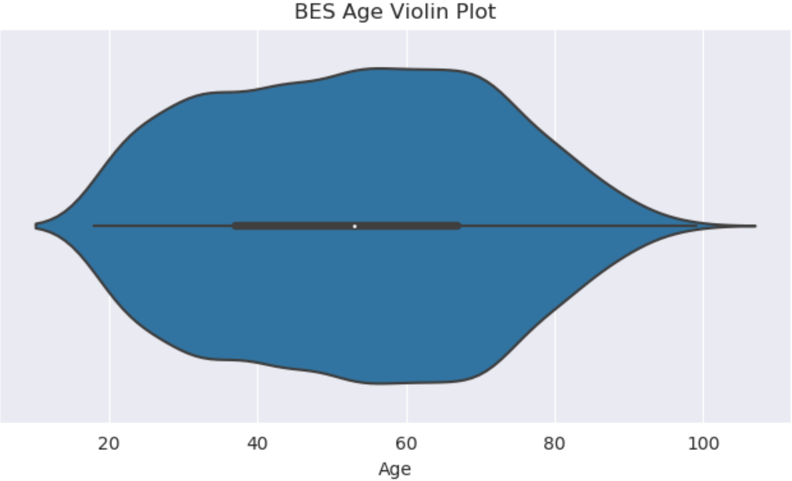
Swarm Plot (One Category)



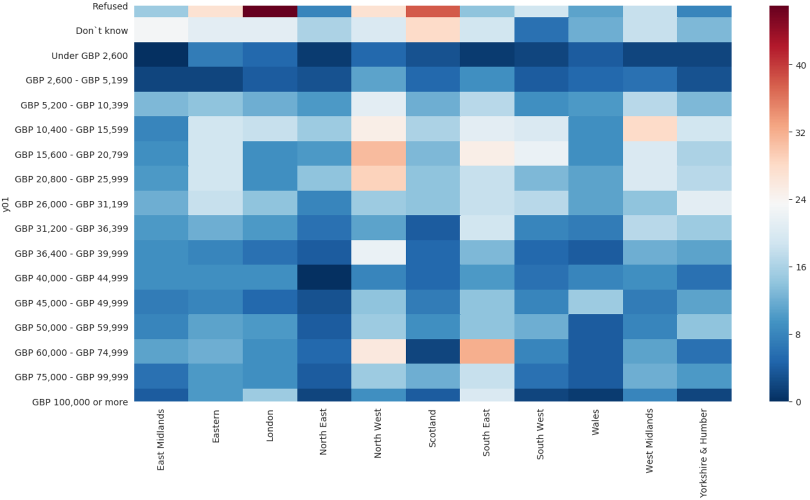
Swarm Plot (Multiple Categories)



Violin Plot (One Category)



Heatmap



Refused
Don`t know
Under GBP 2,600
GBP 2,600 - GBP 5,199
GBP 5,200 - GBP 10,399
GBP 10,400 - GBP 15,599
GBP 15,600 - GBP 20,799
GBP 20,800 - GBP 25,999
GBP 26,000 - GBP 31,199
GBP 31,200 - GBP 36,399
GBP 36,400 - GBP 39,999
GBP 40,000 - GBP 44,999
GBP 45,000 - GBP 49,999
GBP 50,000 - GBP 59,999
GBP 60,000 - GBP 74,999
GBP 75,000 - GBP 99,999
GBP 100,000 or more
y01
East Midlands
Eastern
London
North East
North West
Scotland
South East
South West
Wales
West Midlands
Yorkshire & Humber
region
40
32
24
16
8
0