**Video Script: Section 5 Video 5 – customizing axis labels, legends and axes.**

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Description | Action on screen | Narration |
| 1 | Introduction  (Outcome and why it is desirable)  This should give the viewer an idea of the outcome of the task at the beginning of the videos and set the stage and expectations of the viewer. | Refer to PPT | In this video, we’ll see how to customize labels, legends and axes directions. |
| 2 | Context(Problem/Solution)  Present the viewer with a real-world solution and how the situation would pose as a challenge. It always helps to draw the viewer's attention using a use-case. Metadata template can be used here. |  | By default, axis labels and legend titles are given by the column names corresponding to the aesthetics. This is not always desirable because they can be cryptic to read or unsuitable for a publication. |
| 3 | Guidance (How to do it and how it works): | Switch to activity\_05\_05.R in RStudio.  Highlight and run:  library(ggplot2)  # Legends  p <- ggplot(head(mpg,50)) +  geom\_point(aes(x = cty, y = hwy,color = trans, shape = class), size = 5)  p + ggtitle('Default legends') | Open activity\_05\_05.R and run the first example |
| 4 |  | A description... | We have plotted the city and highway consumption for 50 cars, from the mpg dataset. |
| 5 |  |  | The legend title for the transmission type is ‘trans’ which is a bit cryptic, as well as the axis labels cty and hwy. |
| 6 |  | Highlight xlab('city miles per gallon')  Highlight ylab('highway miles per gallon') | Changing the axis labels is easily done with xlab and ylab.  Adding xlab('city miles per gallon') and ylab('highway miles per gallon') to the plot modifies the labels. |
| 7 |  | Highlight an run:  p + xlab('city miles per gallon') + ylab('highway miles per gallon') +  ggtitle('better axis labels')  point to labels  A description... |  |
| 8 |  | Highlight: guide =  Highlight: scale\_shape\_discrete | Changing the legend titles can be done with the parameter *guide* in the colour scale function used for generating the palette. |
| 9 |  | Highlight: colour = , shape =  Highlight: guides | Or by using the aesthetics name in the guides function. |
| 10 |  | Highlight and run:  p +  guides(colour = guide\_legend("Transmission"), shape = guide\_legend("Class")) +  ggtitle("Custom legends using guides parameters")  A description... | Both solutions will override the default values.  The function *Guides* actually gives you a lot of controls over the rendering of the palette, from its legends to its position. Follow the links in activity\_05\_05.R to see more examples. |
| 11 |  | Highlight:  + coord\_flip() | It is often useful to flip the x- and y- axes for better visualization. Rather than swapping the aesthetics *x* and *y*, we can simply use *coord\_flip* to achieve the same effect. |
| 12 |  | Highlight and run:  p <- ggplot(mpg, aes(x = manufacturer)) + geom\_histogram() + ylab("Number") + xlab('')  # flipping axes  p + coord\_flip() + ggtitle('Flipped axis') | Here is a usual bar chart, flipped over to have the car manufacturer’s names more readable. |
| 13 |  | A description... | Note that the effect is purely graphic: the list of car manufacturers goes from audi at the bottom to Volkswagen at the top, because it was originally going from audi on the left to Volkswagen on the right. |
| 14 |  | Highlight  mpg$manufacturer2 <- factor(mpg$manufacturer,rev(unique(mpg$manufacturer)))  ggplot(mpg) +  geom\_histogram(aes( x = manufacturer2)) +  ylab("Number") +  xlab('') +  coord\_flip() + ggtitle('') | To start with audi at the top, see video 02 in this section of or the last example in the current script. |
| 15 |  |  |  |
| 16 | Conclusion:The video concludes by showing the viewer that the goal has been achieved, and reminding them why they should be happy about that. A PowerPoint summary slide with the key points emphasized would make it easier for the viewer to remember what was covered in the video | Back to PPT | In this video, we saw how to override axis labels, legend titles and axes direction.  In the next section, we’ll see how to build interactive graphical webpages from R with the package Shiny. |