Minilab 4c Worksheet

Building Plots in ggplot

In this minilab, we take a more systematic look at plotting using the R tidyverse package ggplot2. Make sure you have the (green) "Data Visualization with ggplot2" cheat sheet handy (https://rstudio.com/resources/cheatsheets/).

Exploratory Data Analysis (EDA) forms a core part of the "explore" phase of the data science lifecycle, but is also a key part of communicating/interpreting the results of models and further statistical analysis.

OBTAIN
SCRUB
EXPLORE
MODEL
INTERPRET

1. Structure of a Graphical Plot in ggplot

The R tidyverse package ggplot2 is designed to produce beautiful graphics. We can use ggplot2 to draw different types of plots including bar charts, boxplots, line graphs, scatterplots, histograms, heatmaps, surface plots, contour plots, and many more.

(1) Notice that there is a common pattern to each plot.

```
library(tidyverse)
library(ISLR)    # you might also need install.packages("ISLR")
credit = as_tibble(Credit)
ggplot(credit, aes(x=Income,y=Balance)) +
  geom_point()
```

1. The ggplot() function is passed that **data** (as a tibble) that goes into a plot, i.e., a data table of entities (rows) and attributes (columns). Calling this function creates the blank canvas on which the visualisation will be created.

2. You specify the type of **geometric** object (sometimes called a "geom") to draw by calling one of the many geom_ functions. There are geom functions for point (for a scatterplot), line (for a line graph), smooth (for regression lines), col (for a bar chart) and many more (see the green "Data Visualization with ggplot2" cheat sheet).

3. You add layers of geometric objects to the plot by using the addition (**+**) operator.

4. In each geom_ function, you must specify the **aesthetic mapping** which specifies how data from the data tibble will be mapped to the visual aspects of the geometry. These mappings are defined using the aes() function which take named arguments, specifying what variable goes on the x-axis, the colour of the plotting symbol, etc. The mapping can appear in the ggplot() function (in which case it applies to all the following geom_ functions) or to each geom_ function individually.

(2) It is good practice to specify a title and axis labels to a plot.

```
ggplot(credit) +
  geom_point(aes(x=Income,y=Balance,colour=Student)) +
  scale_colour_discrete() +
  labs(title="Balance vs Income ",
       x = "Income ($000)",
       y = "Balance ($)")
```

(3) We also often want to set the scales of a plot, e.g., when we want to draw several plots with the same scales.

```
ggplot(credit) +
  geom_point(aes(x=Income,y=Balance)) +
  scale_x_continuous(limits=c(-100,250))
```

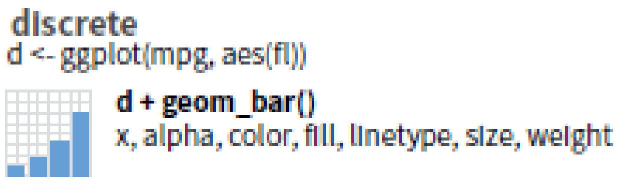
The R package ggplot2 implements a *grammar of graphics*, i.e., a framework which follows a layered approach to describe and construct visualisations or graphics in a structured manner.

2. Building Different Types of Plots

Now we will consider the different functions on the “Data Visualization with ggplot2” Cheat Sheet (tick them off as you try them out). Note that the cheat sheet is both a summary and includes lots of examples. It is worth explaining how to follow the examples presented on the cheat sheet. *At first look, it looks like some kind of strange secret code.*

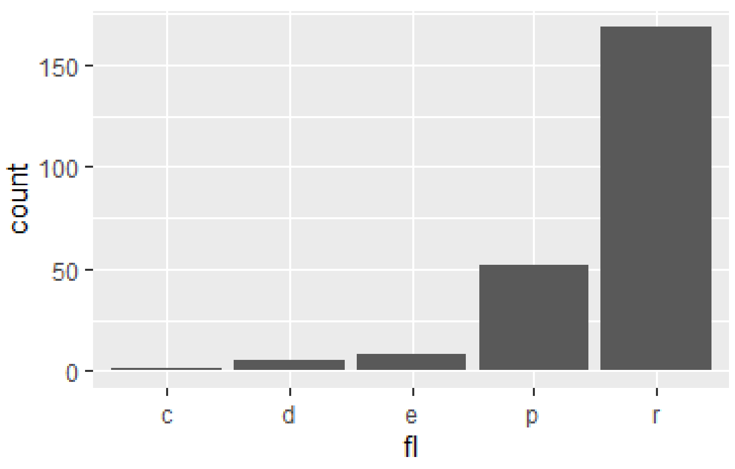
At the top of each small section, there is an initial example with a variable (a,b,c…). The *Grammar of Graphics* means that we can “add” further details. Note that categorical variables are called “discrete” on the cheat sheet, and quantitative variables are called “continuous” on the cheat sheet. All the datasets used (mpg, etc) are built into the tidyverse packages so immediately available to you.

For example, consider adapting the example for the one variable discrete case (bottom of the first page, second column).

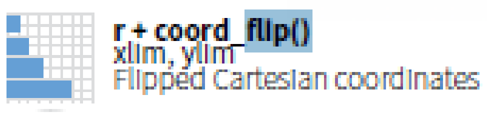


discrete
d <- ggplot(mpg, aes(fl))

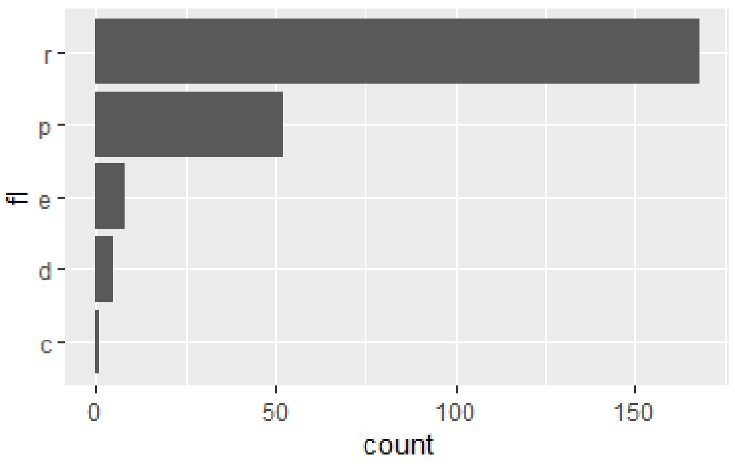
d + geom_bar()
x, alpha, color, fill, linetype, size, weight



Here we add coord_flip() to draw the bars horizontally rather than vertically (second page, column 3 near the top under Coordinate Systems).



r + coord_flip()
xlim, ylim
Flipped Cartesian coordinates



So overall, we have the following R code to build the plot in a variable called *d*. *The final line draws the plot.*

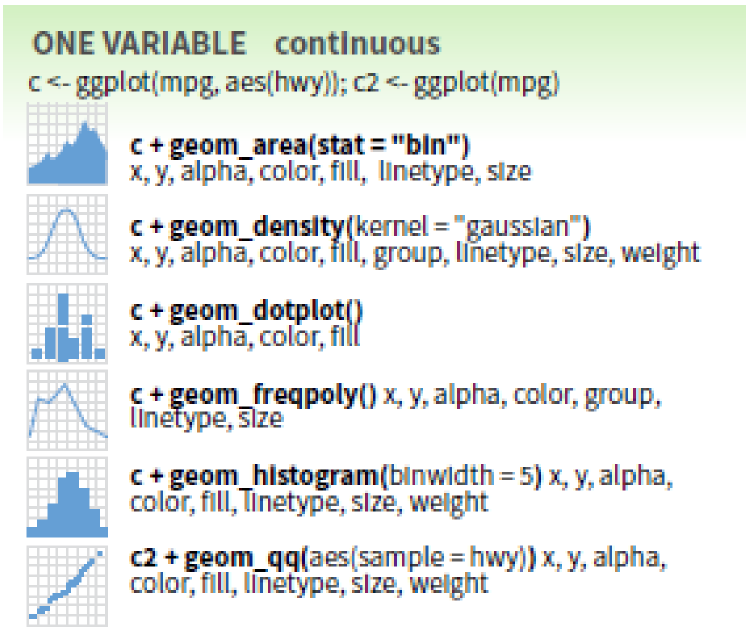
```
d = ggplot(mpg, aes(fl))
d = d + geom_bar()
d = d + coord_flip()
d
```

In practice, we condense this to the following.

```
ggplot(mpg, aes(fl)) +
  geom_bar() +
  coord_flip()
```

Exercise.

(a) The cheat sheet (extract shown below) shows how to draw a **histogram** (for one continuous/quantitative variable).



ONE VARIABLE continuous
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot()
x, y, alpha, color, fill

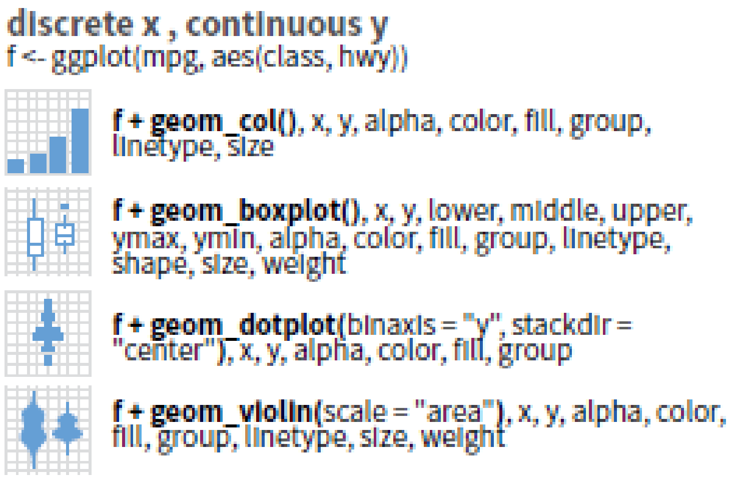
c + geom_freqpoly() x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5) x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy)) x, y, alpha, color, fill, linetype, size, weight

Construct the example plot. Check your understanding using the R Graph Gallery (https://www.r-graph-gallery.com/). Try constructing your own histogram, for example, you could use a variable from the gapminder dataset.

(b) The cheat sheet (extract shown below) shows how to draw side-by-side **boxplots** (for one continuous/quantitative variable and one discrete/categorical variable).



discrete x , continuous y
f <- ggplot(mpg, aes(class, hwy))

f + geom_col(), x, y, alpha, color, fill, group, linetype, size

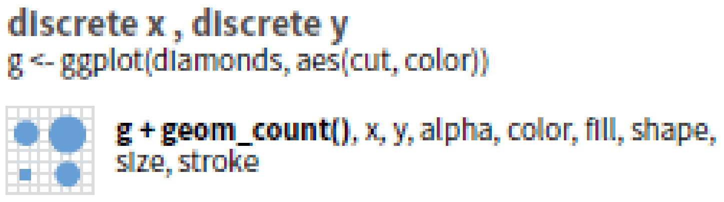
f + geom_boxplot(), x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

f + geom_dotplot(binaxis = "y", stackdir = "center"), x, y, alpha, color, fill, group

f + geom_violin(scale = "area"), x, y, alpha, color, fill, group, linetype, size, weight

Construct the example plot. Check your understand using the R Graph Gallery. For example, you could use a variable from the gapminder dataset.

(c) Try to construct the example plot for two discrete variables from the following extract from the cheat sheet.



discrete x , discrete y
g <- ggplot(diamonds, aes(cut, color))

g + geom_count(), x, y, alpha, color, fill, shape, size, stroke

Summary

In this minilab, we have looked at the structure of plot in the R tidyverse package ggplot2, i.e., data, mapping and geom. The “Data Visualization with ggplot2” cheat sheet provides examples of many different types of plots, so you can’t forget how to build the one you want. Don’t forget to add a title and axis labels (e.g. when you know the units of each variable).