FDP – R for Data Science

Session 3: Data Visualisation

Introduction

Data visualization is the graphical representation of information and data. It is useful to gain insights and understand what happened in the past. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. Usually any analytics project starts with data visualisation.

R programming provides comprehensive sets of tools such as in-built functions and a wide range of packages to perform data analysis, represent data and build visualizations.

R provides many packages for data visualization, each with its advantages and disadvantages. Data visualization in R can be performed using:

- Base Graphics
- Grid Graphics
- Lattice Graphics
- ggplot2

Note: When graphing for the first time with R, most people use base graphics and then move on to ggplot2 when their needs become more complex. In order to save time, we will straight away learn visualisation with ggplot2 (by Leland Wilkinson).

ggplot2

ggplot2 is an R package for producing statistical, or data graphics. It is the most modern of the plotting systems. The gg stands for grammar of graphics which aims to break down graphs into component chunks. ggplot2 is designed to work iteratively. You can start with a layer showing the raw data then add layers of annotations and statistical summaries. The ggplot2 graphics is composed of the following:

- Data
- Layers
- Scales
- Coordinates
- Faceting
- Themes

Every ggplot2 plot has three key components:

1. Data
2. A set of aesthetic mappings between variables in the data and visual properties
3. At least one layer which describes how to render each observation. Layers are usually created with a geom function.

A geom is the geometrical object that a plot uses to represent data. We often describe plots by the type of geom that the plot uses. For example, bar charts use bar geoms, line charts use line geoms, boxplots use boxplot geoms, and so on. ggplot2 provides over 40 geoms, and extension packages provide even more. The best way to get a comprehensive overview is the ggplot2 cheatsheet.

The basic syntax for ggplot is given below:
ggplot(data = , mapping = aes(x = , y =)) + geom_function()

The data and aesthetic mappings are supplied in ggplot(), then layers are added on with + sign. To add additional variables to a plot, we can use other aesthetics like colour, shape, and size. An aesthetic is a visual property of the objects in the plot. Not every aesthetic works with every geom.
Tip: Use the Data Visualisation with ggplo2t cheatsheet as reference

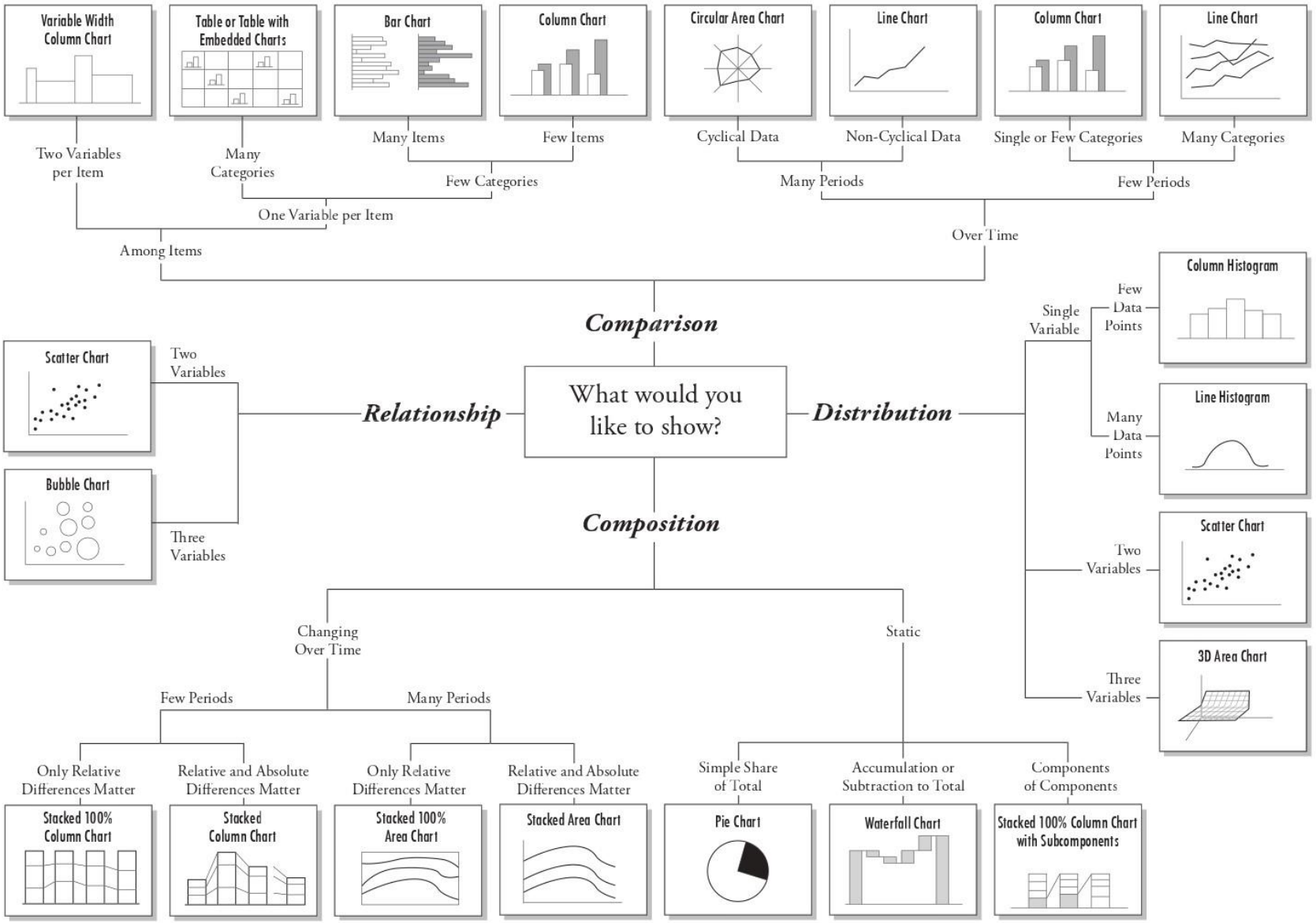
We will be using the placement dataset to understand most of the plots.

```
# Installing & Loading the package
library(ggplot2)

# read the data
location4 <- "C:/Users/Admin/Desktop/FDP_R/Placement_Data_FDP.csv" # copy applicable path
placementgg <- read.csv(location4, stringsAsFactors = T)
head(placementgg)
colnames(placementgg)
str(placementgg)
```

Source: https://extremepresentation.com/

Chart Suggestions—A Thought-Starter



Scatterplot

The most frequently used plot for data analysis is the scatterplot. Whenever we want to understand the nature of relationship between two variables, invariably the first choice is the scatterplot. By displaying a variable in each axis, we can detect if a relationship or correlation between the two variables exists. It can be drawn using geom_point().

Tip: The + sign has to come at the end of the line in case the code is being split over multi lines.

Note: ggplot2 will treat the x and y mappings as global mappings that apply to each geom in the graph. If you place mappings in a geom function, ggplot2 will treat them as local mappings for the layer. It will use these mappings to extend or overwrite the global mappings for that layer only. This makes it possible to display different aesthetics in different layers.

```
# Scatter Plot
base1 <- ggplot(placementgg, aes(x= degree_p, y = mba_p))
base1 + geom_point()
base1 + geom_point(shape = 22, color = "blue", fill = "red", size = 2) # Static
# https://cran.r-project.org/web/packages/ggplot2/vignettes/ggplot2-specs.html
```

In the above, the aesthetic is set by name as an argument of geom function; i.e. it goes outside of aes()

We can convey information about our data by mapping the aesthetics in our plot to the variables in our dataset. To map an aesthetic to a variable, associate the name of the aesthetic to the name of the variable inside aes().

```
# Dynamic - Make the aesthetics vary based on a variable
base1 + geom_point(aes(color = status))
base1 + geom_point(aes(shape = status))
base1 + geom_point(aes(size = status))
base1 + geom_point(aes(alpha = status))
# add additional varaibales with aesthetics
base1 + geom_point(aes(color = status, shape = gender))
base1 + geom_point(aes(color = status, shape = gender, size = workex))
```

Bubble Chart: A bubble chart is a variation of a scatter chart in which the data points are replaced with bubbles, and an additional dimension of the data is represented in the size of the bubbles. Just like a scatter chart, a bubble chart does not use a category axis — both horizontal and vertical axes are value axes. In addition to the x values and y values that are plotted in a scatter chart, a bubble chart plots z (size) values.

```
# add labels
base1 + geom_point()
gg1 <- base1 + geom_point() +
  labs(title = "Scatterplot", x = "% in Degree", y = "% in MBA")
gg1
gg2 <- gg1 +
  theme(plot.title = element_text(face = 'bold.italic', colour = "blue", size = 12, hjust = 0.5),
        axis.title.x = element_text(face = 'bold', colour = "red", size = 10),
        axis.title.y = element_text(face = 'bold', colour = "red", size = 10))
gg2

# make the plot interactive
# install.packages("plotly")
library(plotly)
ggplotly(gg2)

# flip x and y axis
gg2 + coord_flip()
```

Adding a Smoother to a Plot: Lines or curves are fitted within the graph to aid in analysis and are drawn as close to all the points as possible and to show how all the points were condensed into a single line would look. We can add a smooth line to our existing scatter plot using geom_smooth(). The default line can be tweaked by specifying method argument. For example:- method = "lm" fits a linear model, giving the line of best fit.
Note: We can place multiple geoms in the same plot.

```
# using geom_smooth()
base1 + geom_point() + geom_smooth()
base1 + geom_point() + geom_smooth(method = "lm", se = F) # Don't add shaded confidence region
base1 + geom_smooth()
base1 + geom_smooth(aes(colour = status))
base1 + geom_point(aes(colour = status)) + geom_smooth(aes(colour = status))
```

Bar Chart
Bar chart is a frequency chart for qualitative variable (or categorical variable). Bar chart can be used to assess the most-occurring and least-occurring categories within a data set. Bar chart answers the question of 'how many?' in each category. geom_bar() is used to draw a bar chart.
Column Chart: Vertical bar charts are commonly called column charts. A stacked column chart displays the contribution of each value to the total by stacking the

rectangles and a 100% stacked column chart compares the percentage that each value contributes to a total.
Note: By default, geom_bar() has the stat set to count.

```
# Bar Chart
base2 <- ggplot(placementgg, aes(degree_t))
base2 + geom_bar()
base2 + geom_bar(aes(y= stat(prop), group=1))
base2 + geom_bar(aes(fill = degree_t)) # color the bar chart (use same variable)
base2 + geom_bar(aes(color = status))
base2 + geom_bar(aes(fill = status)) # the bars are stacked
base2 + geom_bar(aes(fill = status), position = "dodge") #side-by-side
base2 + geom_bar(aes(fill = status), position = "fill") # percenatge bar chart
base2 + geom_bar(fill = "yellow") + coord_flip()
# Beautifying
base2 + geom_bar(aes(fill = status)) + theme(legend.position = "bottom") +
  ggtitle("Bar Chart")
base2 + geom_bar(aes(fill = status), width = 0.75) +
  geom_text(stat = "count", aes(label=stat(count)), vjust = 1)
base2 + geom_bar(aes(fill = status)) + theme_bw() #theme function changes appearance
```

Dot Plots

It is a relatively simple statistical chart that is generally used to display continuous, quantitative data. In a dot plot, each data value is plotted along the horizontal axis and is represented by a dot on the chart. If multiple data points have the same values, the dots will stack up vertically. Dot plots are especially useful for observing the overall shape of the distribution of the data points along with identical data values or intervals for which there are grouping and gaps in the data.

```
# Dot Plot
ggplot(placementgg, aes(degree_p)) + geom_dotplot(binwidth = 1)
```

Histogram

A histogram is a plot to explore the distribution of a continuous variable. It is a frequency distribution of data arranged in consecutive and non-overlapping classes. Histograms give an insight into the underlying distribution (eg. normal distribution) of the variable, outliers, skewness etc.
Histogram on a continuous variable can be accomplished using geom_histogram(). When using geom_histogram(), we can control the number of bars using the bins option. Else, we can set the range covered by each bin using binwidth. It is

very important to experiment with the bin width. The default just splits the data into 30 bins, which is unlikely to be the best choice. We should always try many bin widths, and we may find that we need multiple bin widths to tell the full story of our data.

```
# Histogram
base3 <- ggplot(placementgg, aes(degree_p))
base3 + geom_histogram()
base3 + geom_histogram(bins = 10)
base3 + geom_histogram(binwidth = 5, fill = "brown")
base3 + geom_freqpoly(binwidth = 5) #  frequency polygons use lines instead of bars
base3 + geom_histogram(binwidth = 5, fill = "brown") + geom_freqpoly(binwidth = 5)
base3 + geom_histogram(aes(fill = status), binwidth = 5)
base3 + geom_histogram(aes(fill = status), binwidth = 5) +
  facet_wrap(~gender)
```

Density Plot

(Kernel Density Plot / Density Trace Graph)

A Density Plot visualises the distribution of data over a continuous interval or time period. This chart is a variation of a Histogram that uses kernel smoothing to plot values, allowing for smoother distributions by smoothing out the noise. The peaks of a Density Plot help display where values are concentrated over the interval.

An advantage Density Plots have over Histograms is that they're better at determining the distribution shape because they're not affected by the number of bins used.

```
# Density Plot
ggplot(placementgg) + geom_density(aes(etest_p), fill = "pink") + theme_classic()
ggplot(placementgg) + geom_density(aes(etest_p, fill = degree_t))
ggplot(placementgg) + geom_density(aes(etest_p, fill = degree_t), alpha = 0.5)
```

Box Plot

(Box and Whisker Plot)

A box plot is a graphical representation of numerical data that can be used to understand the variability of the data and the existence of outliers. Box plot is designed by identifying the following descriptive statistics:

1. Lower Quartile, Median, and Upper Quartile
2. Lowest and highest value
3. Inter-quartile range (IQR)

The length of the box is equivalent to IQR. The dark line inside the box represents the median. The whisker of the box plot extends till Q_1 – 1.5 IQR (or minimum value) and

Q_3 + 1.5 IQR (or maximum value); observations beyond these two limits are marked as dots and are potential outliers.

Faceting: Faceting is used to further drill down data and split the data by one or more variables, and then plot the subsets of the data altogether for optimum data visualization in R. To facet your plot by a single variable, use facet_wrap(). To facet your plot on the combination of two variables, add facet_grid() to your plot call.

```
# Box Plot
base4 <- ggplot(placementgg, aes(y = mba_p))
base4 + geom_boxplot(fill = "yellow")
base4 + geom_boxplot(aes(x = gender))
base4 + geom_boxplot(aes(fill = status))
base4 + geom_boxplot(aes(fill = status)) +
  facet_wrap(~gender)
base4 + geom_boxplot(aes(y = mba_p, fill = status)) +
  facet_grid(workex~gender)
```

Violin Plot

This chart is a combination of a Box Plot and a Density Plot that is rotated and placed on each side, to show the distribution shape of the data. geom_violin(), show a compact representation of the density of the distribution, highlighting the areas where more points are found.

```
#  Violin Plot
base4 + geom_violin(aes(x = status), fill = "green")
```

Pie Chart

Pie chart is mainly used for categorical data. Pie Charts help show proportions and percentages between categories, by dividing a circle into proportional segments. Each arc length represents a proportion of each category, while the full circle represents the total sum of all the data, equal to 100%.

Pie charts are not recommended in the R documentation. A bar is recommended because people are able to measure length more accurately than volume.

```
# Pie Chart
piebar <- ggplot(placementgg, aes(x = " ", fill = degree_t)) + geom_bar(width = 1)
piebar
piechart <- piebar + coord_polar("y") + theme_void()
piechart
pie(table(placementgg$degree_t)) # using base graphics
```

Scatterplot Matrix

A scatterplot matrix is a collection of scatterplots organized into a grid (or matrix). Each scatterplot shows the relationship between a pair of variables.

We will be using the GGally R package which is an extension to ggplot2. It creates a matrix with scatterplots in the lower diagonal, densities on the diagonal and correlations written in the upper diagonal.

```
# Pairwise scatterplot matrix
#install.packages("GGally")
library("GGally")
ggcorr(placementgg)
ggpairs(placementgg, columns = c(3,5,8,11,13)) # correlogram
ggscatmat(placementgg, columns = c(3,5,8,11,13), color = "status") # alternative
```

Correlogram: A correlogram or correlation matrix allows to analyse the relationship between each pair of numeric variables in a dataset. It gives a quick overview of the whole dataset. It is more used for exploratory purpose than explanatory.

Quick Plots

In some cases, we will want to create a quick plot with a minimum of typing. In these cases we may prefer to use qplot() over ggplot(). qplot() lets us define a plot in a single call, picking a geom by default if we don't supply one. To use it, we have to provide a set of aesthetics and a data set.

```
# quick plot
qplot(degree_p, mba_p, data = placementgg)
qplot(degree_p, mba_p, data = placementgg, colour = status)
qplot(degree_t, data = placementgg)
qplot(degree_p, data = placementgg)
qplot(y= degree_p, data = placementgg)
```

Saving a plot

ggsave() is a convenient function for saving a plot. It defaults to saving the last plot that you displayed, using the size of the current graphics device. It also guesses the type of graphics device from the extension. You can also control the width, height and dpi using the additional arguments.

Alternatively, use the Export option available under Plots tab.

```
# saving a plot
setwd("C:/Users/Admin/Desktop/FDP_R") # copy applicable path
ggsave("myplot.pdf")
ggsave("myplot.png")
```

Other charts:

Line Chart

Line charts are often used when one variable has a certain continuity. A Line Graph is most frequently used to show trends and analyse how the data has changed over time. Typically, the y-axis has a quantitative value, while the x-axis is a timescale or a sequence of intervals. When grouped with other lines (other data series), individual lines can be compared to one another.

Treemap

Treemap is a hierarchical map made up of nested rectangles frequently used as a part of business intelligence reports. Each category is assigned a rectangle area with their subcategory rectangles nested inside of it. The size of rectangle and colours are used for describing/differentiating the characteristics of the data.

Heatmap

Heatmaps visualise data through variations in colouring. When applied to a tabular format, Heatmaps are useful for cross-examining multivariate data, through placing variables in the rows and columns and colouring the cells within the table. They use color to communicate relationships between data values that would be much harder to understand if presented numerically in a spreadsheet. Heatmaps are good for detecting if any correlation exists between the variables.

Mosaic Plot

A mosaic plot (also known as a Marimekko diagram) is a graphical method for visualizing data from two or more qualitative variables. The mosaic plot starts as a square with length one. The square is divided first into horizontal bars whose widths are proportional to the probabilities associated with the first categorical variable. Then each bar is split vertically into bars that are proportional to the conditional probabilities of the second categorical variable. Additional splits can be made if required. Mosaic plots are constructed hierarchically, so the ordering of the variables is very important.