Lab

Building plots layer by layer with ggplot2

Lab objectives:

- To consolidate knowledge about how to create plots layer by layer
- To learn more about geom_bar
- To learn how to construct pie charts (not so easy with ggplot!)
- To learn to add titles and labels to the plot.

Type all the examples in this lab and check the results to ensure you understand the concepts being introduced. Exercises are at the end of this lab.

ggplot2

In this lab, we are using ggplot2. Ensure that you import this library as explained in lab 3.

Data

Data must be represented by a data frame.

Note: in this lab, as in previous labs, we will assume that we have perfect data which we do not need to transform. The datasets to be downloaded have already been manipulated to give the perfect data for our purposes.

Download the following files from CampusMoodle **weeks 3 and 4**:

- *consumptionNoOil.csv:* It contains data regarding the consumption of various energy sources since 1990. Some of the data are predictions (e.g. for year 2035). The dataset contains the year, the Million Ton of Oil Equivalent (MTOE) and the energy source.
- *projectTeam.csv:* contains details of the job title and site at which members of a project team work.
- *importsExportsAmerica.csv:* contains details of the crude and product imports and exports for various countries in America.
- *provenOilReservesT.csv:* it contains data regarding proven oil reserves for European countries.
- *provenOilReserveWEurope.csv*: contains data regarding proven oil reserves for western European countries.
- *channels*.csv: contains perceived sensation vs stimuli for various channels.
- *uni*.csv: contains the data regardingthe number of staff, student and "both" for 4 universities.

Load these datasets.

Geoms

We have seen the point, the bar and the line geoms. Other geoms which you may use (the is is not exhaustive) include:

- geom_area(): creates an area plot which is filled to the value of the y axis
- geom_polygon(): creates polygons where each point which makes the polygon represents one line of data.
- geom_text(): includes labels for points
- geom_tile():uses tiles.

Ensure data *file importExportsAmerica.csv* is loaded.

Try the following

```
p <- ggplot(importsExportsAmerica, aes(crude.imports,crude.exports,
label=country))
p <- p+geom_text()
p
```

Try the code above but replace geom_text() with each of the following

- geom_area()
- geom_polygon()
- geom_tile()

Barplots, including bar position

You can tweak the position of the (discrete) data which you are plotting. The following positions can be used:

- Identity – no position adjustment (useful for lines but not for bars)
- Dodge – adjust the position to the side to avoid overlaps
- Fill – stack overlapping objects. Standardise heights
- Jitter – points are jittered to avoid over-plotting
- Stack – put overlapping objects on top of each other

Compare the following.

Stacking

```
p <- ggplot(projectTeam, aes(site, fill = experience))
p <- p + geom_bar(position = "stack")
p
```

Dodging

```
p <- ggplot(projectTeam, aes(site, fill = experience))
p <- p + geom_bar(position = "dodge")
p
```

Filling

```
p <- ggplot(projectTeam, aes(site, fill = experience))
p <- p + geom_bar(position = "fill")
p
```

Identity

```
p <- ggplot(projectTeam, aes(site, fill = experience))
p <- p + geom_bar(position = "identity")
p
```

Note that even if the plot appears, not all plots make sense. Note also how identity is not useful as some of the bars are hidden behind others.

Adjusting the x and y axis

You saw the following example in the previous lab.

```
p <- ggplot(consumptionNoOil, aes(Year,MTOE, colour=Source))
p <- p+geom_point()
p
```

You may adjust the x and y axis using xlim and ylim to see more clearly the data from 2010 onwards. Try

```
p <- ggplot(consumptionNoOil, aes(Year,MTOE, colour=Source))
p <- p + xlim(2010,2035)
p <- p+geom_point()
p
```

R will issue a warning to let you know that some points are outside the plotting region (all the ones for years 1990, 1995, 2000 and 2005).

To adjust the y axis, use ylim().

Adding titles, subtitles and labels

You can use labs to add labels and titles to your plot. For example, try

```
p <- ggplot(consumptionNoOil, aes(Year,MTOE, colour=Source))
p <- p+geom_point()
p<- p+ labs(x="Year", y="Millions of tons of oil equivalent",
            title ="Energy consumption",
            subtitle = "Oil is excluded")
p
```

But this does not change the label for the legend. The legend is changed by adding "colour" in the labs function. Try

```
p <- ggplot(consumptionNoOil, aes(Year,MTOE, colour=Source))
p <- p+geom_point()
p<- p+ labs(x="Year","Millions of tons of oil equivalent",
```

```
       title ="Energy consumption by source",
       subtitle = "Oil is excluded",
       colour = "Energy source")
p
```

Changing the colours

You may wish to change the colours used in your plots. For example, in the previous plot, you may wish to colour coal black, gas yellow and renewables green. You can do the following:

```
colours <-c(coal="black", gas="yellow2", renewables="green")
p <- ggplot(consumptionNoOil, aes(Year, MTOE, colour=Source))
p <- p +geom_point()
p<-p+labs(x= "Year",
          y= "Millions of tons of oil equivalent",
          title = "Energy consumed",
          colour="Energy source")
p<- p + scale_colour_manual(values=colours)
p
```

Adding lines

You may wish to add some lines to make it easier to gauge values from the plot.

For example try

```
p <- ggplot(consumptionNoOil, aes(Year,MTOE, colour=Source))
p <- p+geom_point()
p <- p + geom_vline(aes(xintercept = Year), linetype = "longdash",
                        colour="green")
p
```

A vertical line will appear at every year. Compare this with

```
p <- ggplot(consumptionNoOil, aes(Year,MTOE, colour=Source))
p <- p+geom_point()
p <- p + geom_vline(aes(xintercept = 2015),  linetype = "longdash",
                            colour="green")
p
```

where there is only one vertical line, in 2015.

If we wanted a horizontal line we could try

```
p <- ggplot(consumptionNoOil, aes(Year,MTOE))
p <- p+geom_point()
```

```
p <- p + geom_hline(aes(yintercept = 100), linetype = "dotdash",
colour="green")
p
```

And if we want both

```
p <- ggplot(consumptionNoOil, aes(Year,MTOE, colour=Source))
p <- p+geom_point()
p <- p + geom_hline(aes(yintercept = 100), linetype = "dotted",
                          colour="darkgreen")
p <- p + geom_vline(aes(xintercept = 2015),  linetype = "twodash",
                          colour="blue")
p
```

Note: types are 0 = blank, 1 = solid, 2 = dashed, 3 = dotted, 4 = dotdash, 5 = longdash, 6 = twodash. The line types can also be specified using a number, e.g. `linetype = 4`.

Pie Charts

Producing pie charts in ggplot2 is not straightforward. They are made with the `geom_bar` (same as bar charts), and then it is made into a circle using `coord_polar("y", start=0)`. The following is an example using the project team data to plot the number of staff per site.

First, we create a new data frame which contains the sites and the number of staff per site, which is obtained by counting the number of times each site name appears in the data.

```
# counting the number of staff per site.
pieData <- projectTeam |> count(site, name="Number")
```

Next we add the column names

```
names(pieData) <- c("site","numberStaff")
```

In order to the pie to be presented properly, data needs to be ordered in descending order. We can do this as follows.

```
# order data according to the site (important for placing labels)
pieData <- arrange(pieData,desc(site))
```

We also need to create a position of each amount (i.e. number of staff), if we want them to appear in the pie chart. The following code does this.

```
# create new column with position for label
pieData <- pieData |>
     mutate(positionLabel = cumsum(numberStaff) - 0.5* numberStaff)
```

Note that function cumsum returns the cumulative sum. For example, column numberStaff has values 3, 5 and 6. Therefore, `cumsum(numberStaff)` will return values 3, 8 and 14, where 8 is 3+5 and 14 is 3 + 5 + 6. As labels are positioned in the middle of each slice, half the size of a slice is removed to calculate the position of the label.

We are now ready to create the plot.
We need to use the geom_bar() to get the right proportions in the data with a `stat="identity".` We then make it into a circle using `coord_polar("y", start=0).`

```
# creating plot
p <- ggplot(pieData,  aes(x="", y= numberStaff, fill = site))
p <- p + geom_bar(width = 1, stat="identity")
p <- p + coord_polar("y", start=0)
```

We can add the actual amounts (number of staff) to the pie chart using text labels as follows.

```
# adding text labels
p <- p + geom_text(aes(y = positionLabel, label = numberStaff))
```

If we want to change the colour scheme, we can give new values using the `scale_new_values()` function.

```
# Changing the colour scheme
p <- p +   scale_fill_manual(values = c("red","blue", "lightblue"))
```

Finally, the above code produces an outer ring on the chart. To remove it use

```
# removing outer "ring"
p <- p + theme_void()
```

We are now ready to visualise our pie chart so we execute `p.`

```
p
```

Exercises

You have done some of these exercises in lab 3, but your plots were incomplete. For example, they had no title.

1. Produce a scatterplot of yearly proven oil vs Denmark. Ensure that your plot contains a title and appropriate labels for both the x and y axis.
2. Update your scatterplot to contain a vertical lines.
3. Produce a line plot of yearly proven oil vs Italy. Your line should be of size 2, of orange colour. Ensure your title and labels are appropriate and that you have at least one horizontal line.
4. Produce a scatterplot of yearly proven oil vs United Kingdom. Use colour to emphasize the size of values. Ensure your legend has an appropriate label. Your plot must have a title and suitable x and y axis labels.

5. Load the provenOilReserveWEurope.csv dataset. Produce a line plot of year versus Million Tons of Barrels by country. Add titles and labels to your plot.
6. Using the channels dataset, plot length against area using a scatterplot. Use size to indicate "n" (the size of the stimuli).
7. Repeat the exercise above, but this time change the y axis to go from 0 to 5, so that it has the same scale as the x axis. Compare your plot with your previous plot, paying particular attention to which one is more effective to compare length against area. Ensure appropriate titles and labels.
8. Use the universities.csv dataset to plot status. Use the school as fill. Experiment with various positions such as stacking , doging, filling and identity.
9. Use the universities data to produce a pie chart of number of students per school. You will first need to select only the data related to students, i.e. where the status is either "student" or "both". You can do this as follows (leaving the result in universities2):

```
universities2 <- universities |>
     filter(status %in% c("student", "both"))
```