Assignment 1

Trisha Sukhtankar

25/01/2020

Importing the data

```
data <- read.csv("D:/Assignments/Sem 2/Data Visualization/week2_data_4cat(1).csv") #importing data
str(data) #Having a look at the type of variables, data container, number of variables and observations
```

```
## 'data.frame':    44 obs. of  3 variables:
##  $ var_a  : num  10.1 7.9 13 9 11 14 6 4 12 7 ...
##  $ var_b  : num  9.14 8.14 8.74 8.77 9.26 8.1 6.13 3.1 9.13 7.26 ...
##  $ country: Factor w/ 4 levels "england","ireland",..: 2 2 2 2 2 2 2 2 2 2 ...
```

```
head(data) #displays the first few rows of the data for better analysis
```

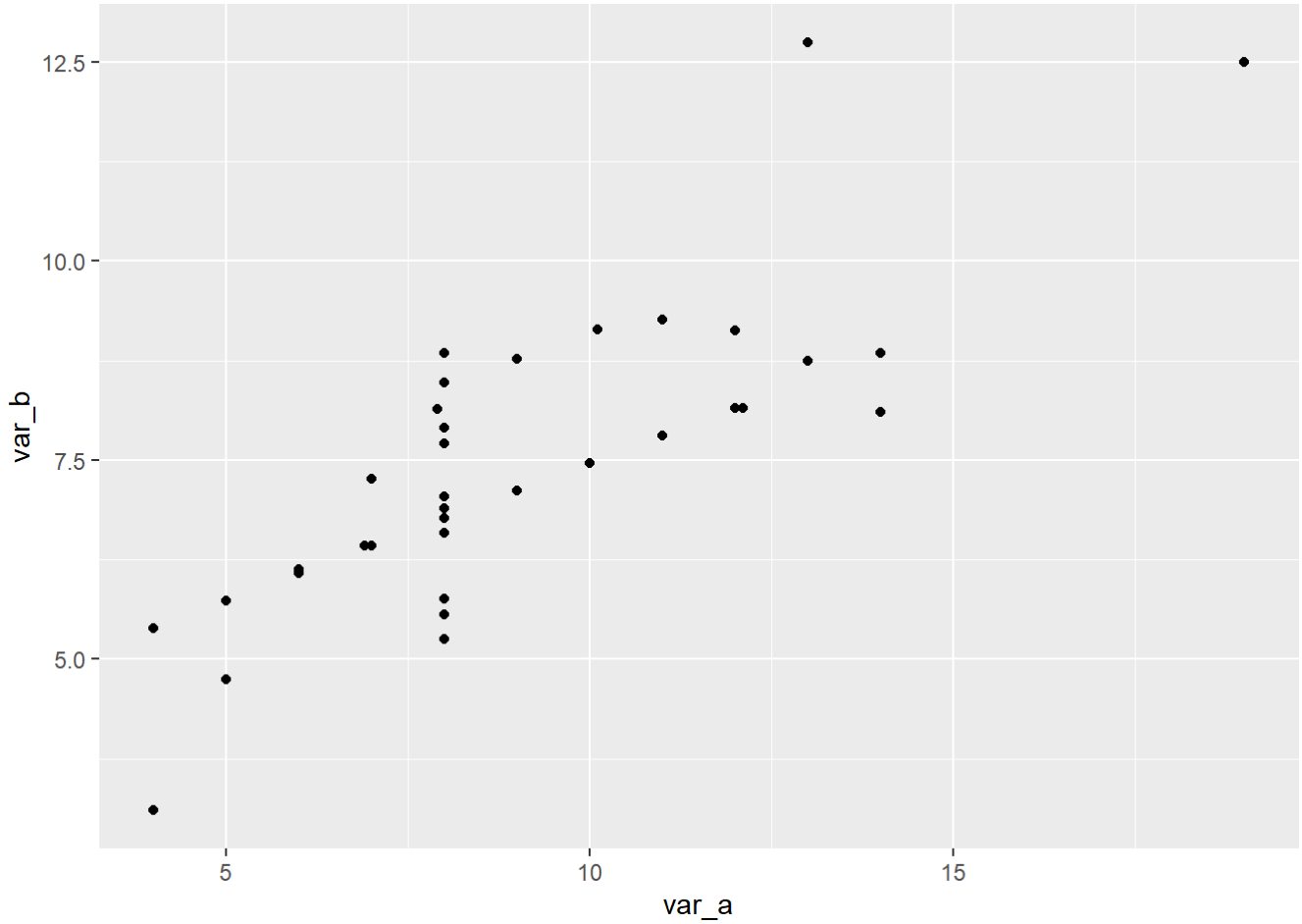
	var_a <dbl>	var_b <dbl>	country <fctr>
1	10.1	9.14	ireland
2	7.9	8.14	ireland
3	13.0	8.74	ireland
4	9.0	8.77	ireland
5	11.0	9.26	ireland
6	14.0	8.10	ireland

6 rows

Part 1: Plotting the data

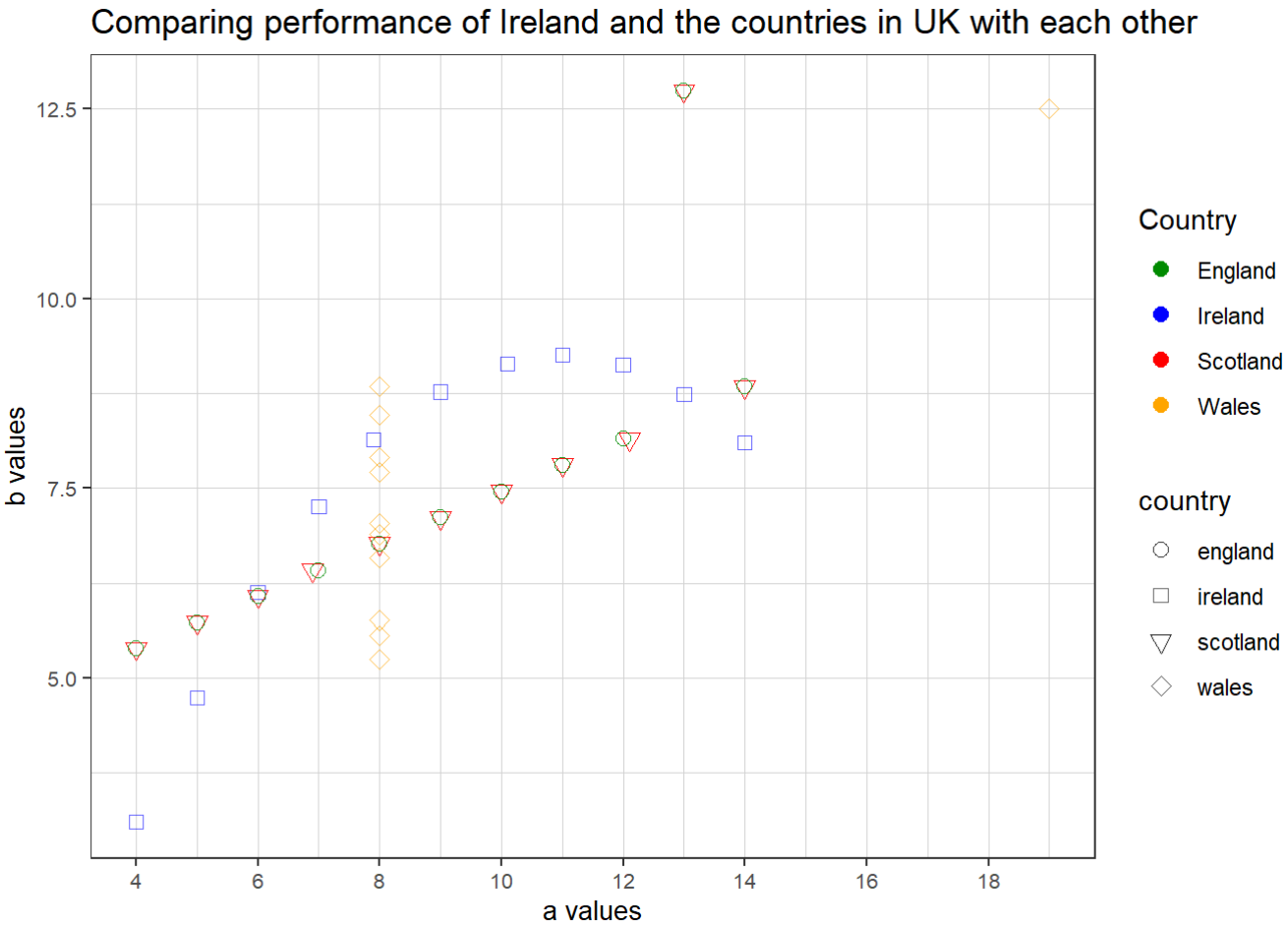
The first step for plotting the graph is to import ggplot2 (we could have achieved the same result using tidyverse package as well since ggplot2 is a part of tidyverse package). Upon having look, we can tell that 'country' is a factor and 'var_a', 'var_b' are merely values corresponding to the factor variable. The following is a basic scatterplot of the given dataset:

```
library(ggplot2)

plot <- ggplot(data = data, aes(x = var_a, y = var_b)) +
                geom_point() #adding geom layer and defining the type of plot
plot
```



In the above plot, a lot of data points are overlapping each other. Therefore, it is difficult to interpret the data from the above plot. Now, in order to make the datapoints clearly visible and distinguishable, we shall add custom color scheme based on the factors, reshape the points, adjust alpha values (to define opacity of each point based on'Country') and customize a suitable background to the plot.

```
plot <- ggplot(data = data, aes(x = var_a, y = var_b, colour = country, alpha = country, shape = country)) +
                geom_point(size = 2.5) +

                scale_alpha_manual(values = c(0.8, 0.6, 0.9, 0.7)) + #setting opacity of data points

                scale_colour_manual(values = c("green4", "blue", "red", "orange"), name = "Country", labels = c("England",
"Ireland", "Scotland", "Wales"))+ #setting colours for each country

                scale_shape_manual(values = c(21, 22, 25, 23)) + #setting shape as per the country

                ggtitle("Comparing performance of Ireland and the countries in UK with each other") +

                xlab("a values") + #new label for x axis
                ylab("b values") + #new label for y axis

                scale_x_continuous(breaks = c(4,6,8,10,12,14,16,18,20))+

                theme_bw()+
                theme(axis.title = element_text(size = 10),
                      axis.text = element_text(size = 8),
                      panel.background = element_rect(fill = "white"),
                      panel.grid.major = element_line(size = 0.25, linetype = 'solid', colour = "lightgrey"),
                      panel.grid.minor = element_line(size = 0.1, linetype = 'solid', colour = "lightgrey"))


plot
```



Part 2 : Calculating mean and standard deviation

Calculating mean and standard deviation of var_a and var_b when grouped by their country and the correlation between the 2 variables:

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
data_stats <- data %>%
  group_by(country) %>%
  summarise(cor_ab = cor(var_a, var_b), mean_a = mean(var_a), mean_b = mean(var_b), st_a = sd(var_a), st_b = sd(var_b))
data_stats
```

country <fctr>	cor_ab <dbl>	mean_a <dbl>	mean_b <dbl>	st_a <dbl>	st_b <dbl>
england	0.8162867	9	7.500000	3.316625	2.030424
ireland	0.8161639	9	7.500909	3.322950	2.031657
scotland	0.8150854	9	7.500000	3.331966	2.030424
wales	0.8165214	9	7.500909	3.316625	2.030579

4 rows

Formatting the table

We are using *kableExtra* package to format our tables. Here, **kable_styling()** enables the styling of a basic, crude-looking table and transforms into more readable and asthetically better format. The *bootstrap_options* lets us control the format into twitter-like tables by enabling us to use the various predefined classes such as ‘striped’, ‘hover’ etc.

```
library(knitr)
library(kableExtra)
```

```
## Warning: package 'kableExtra' was built under R version 3.6.2
```

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
kable(data) %>%
  kable_styling(bootstrap_options = "striped") %>%
  scroll_box()
```

var_a	var_b	country
10.1	9.14	ireland
7.9	8.14	ireland
13.0	8.74	ireland
9.0	8.77	ireland
11.0	9.26	ireland
14.0	8.10	ireland
6.0	6.13	ireland
4.0	3.10	ireland
12.0	9.13	ireland
7.0	7.26	ireland

var_a	var_b	country
5.0	4.74	ireland
10.0	7.46	scotland
8.0	6.77	scotland
13.0	12.74	scotland
9.0	7.11	scotland
11.0	7.81	scotland
14.0	8.84	scotland
6.0	6.08	scotland
4.0	5.39	scotland
12.1	8.15	scotland
6.9	6.42	scotland
5.0	5.73	scotland
10.0	7.46	england
8.0	6.77	england
13.0	12.74	england
9.0	7.11	england
11.0	7.81	england
14.0	8.84	england

var_a	var_b	country
6.0	6.08	england
4.0	5.39	england
12.0	8.15	england
7.0	6.42	england
5.0	5.73	england
8.0	6.58	wales
8.0	5.76	wales
8.0	7.71	wales
8.0	8.84	wales
8.0	8.47	wales
8.0	7.04	wales
8.0	5.25	wales
19.0	12.50	wales
8.0	5.56	wales
8.0	7.91	wales
8.0	6.89	wales