lecture5

Marc Kaufmann

10/9/2019

Assignement 6

1. Please highlight the start of each exercise properly, as well as where your answer starts and ends.
2. Upload **only your exercises** and answers to Moodle, not all of the lecture notes.

Exercise 0: Manage your time. If you spend too much time on the assignment, it may be too long, and you may benefit from skipping an exercise. Since this lesson is lost on most people, exercise 0 requires you to skip one of the other exercises this week.[1] That is, write down which exercise you want to skip and why ("saves the most of time because it is the hardest" or "it's the most useless as you are quite confident in material"). Do not work on it. Of course, if you skip any additional exercise due to lack of time, highlight this, but that will cost some part of the grade, whereas skipping an exercise as part of exercise 0 will not.

Ans: I am skipping Excercise 6 because I feel that we covered that material (on how to load files) really well in class and there is no added benefit of repeating that again.

Exercise 1: Using the nycflights13 data. Note that it also contains a tibble called `airports` (as well as others). Use these two dataframes to find the answer to 3 of the following, and print them out in a separate chunk (i.e. the chunk should print the tibble, thus showing the first 10 lines of each):

- The number of flights (in the whole year) to each destination
- The number and list of distinct airports in the US
- The number and list of distinct airports that have at least one flight in the whole year from NYC
- The number and list of distinct airports that have at least one flight **per day (on average)** from NYC
- The number airports that are further south than NYC (Hint: look up longitude and latitude.)
- The top 5 carriers that have the lowest average delay times.
- What is the worst day of the year to fly in terms of arrival delay?
- What is the best day of the year to fly in terms of departure delay?

Reminder: Pick only 3 of the 8 possible ones.

Solution 1:

1. The number of flights (in the whole year) to each destination

```
library(nycflights13)
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(gapminder)
library(SportsAnalytics)

no_of_flights <- flights %>% group_by(dest) %>% summarise(no_of_flights <- n())
```

[1]If you want to be cute, skip exercise 0.

2. number and list of distinct airports in the US

```
list_of_airports <- airports %>%  group_by(name) %>% summarise(no_of_airports <- n_distinct(name)) %>% 
```

There are a total of 1440 airports because there are 1440 rows of the above table

The number and list of distinct airports that have at least one flight in the whole year from NYC

```
find_airports <- merge(flights, airports,
      by.x = 'dest', by.y='faa') %>% group_by(name) %>% summarise(distinct_airports_ny <- n_distinct(na
```

There are 101 distinct airports that have at least one flight in the whole year from NYC.

Exercise 2: Find all the rows with NA values in **the first two columns** for the following datasets:

- diamonds
- flights
- mtcars

Solution

```
diamonds %>% filter(is.na(carat), is.na(cut))
```

```
## # A tibble: 0 x 10
## # ... with 10 variables: carat <dbl>, cut <ord>, color <ord>,
## #   clarity <ord>, depth <dbl>, table <dbl>, price <int>, x <dbl>,
## #   y <dbl>, z <dbl>
```

```
flights %>% filter(is.na(year), is.na(month))
```

```
## # A tibble: 0 x 19
## # ... with 19 variables: year <int>, month <int>, day <int>,
## #   dep_time <int>, sched_dep_time <int>, dep_delay <dbl>, arr_time <int>,
## #   sched_arr_time <int>, arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
## #   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
mtcars %>% filter(is.na(mpg), is.na(cyl))
```

```
##  [1] mpg  cyl  disp hp   drat wt   qsec vs   am   gear carb
## <0 rows> (or 0-length row.names)
```

The next exercise asks you to check *all* columns, but you don't want to do that with `filter()`. Why do you not want to do that with filter, especially for a dataset with hundreds of columns?

Ans: Because that would be too cumbersome as we would have to name each column - it would be simpler use is.na() function over the whole dataframe.

If you can't find a dataset, do `??<dataset>` to find out more about it and what library you need to load.

Exercise 3: Look up `filter_all` and look at the examples at the end of the documentation. Use this (with some google-fu or discourse help) to find all the rows with NA values in *any* column for the following datasets:

- diamonds
- flights
- mtcars

Solution

```
diamonds_na <- diamonds %>% filter_all(any_vars(is.na(.)))

flights_na <- flights %>% filter_all(any_vars(is.na(.)))

mtcars_na <- mtcars %>% filter_all(any_vars(is.na(.)))
```

Thus, the output should be those rows that *do* contain NA values. Then look up `na.omit` (hat tip @kristof) and use that do achieve the same goal.

Solution using na.omit

```
diamonds_na2 <- setdiff(diamonds, na.omit(diamonds))
assertthat::are_equal(diamonds_na2, diamonds_na) #assert that the resulting dataframe from using filter
```

```
## [1] TRUE
```

```
flights_na2 <- setdiff(flights, na.omit(flights))
assertthat::are_equal(flights_na2, flights_na)
```

```
## [1] TRUE
```

```
mtcars_na2 <- setdiff(mtcars, na.omit(mtcars))
assertthat::are_equal(mtcars_na2, mtcars_na)
```

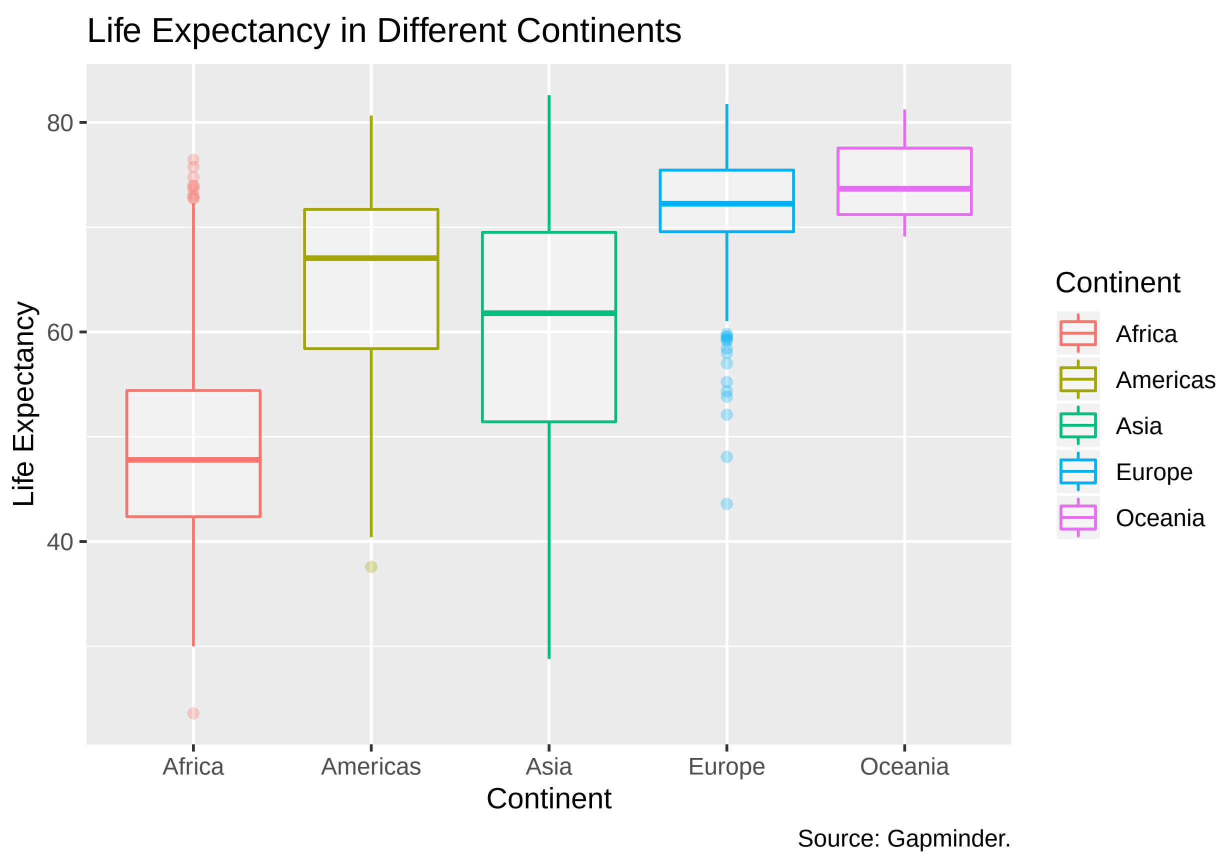
```
## [1] TRUE
```

Exercise 4: Pick your favourite dataset. Use it to illustrate the following types of plots, and describe briefly (2-3 sentences) what each plot means. I.e. for the boxplot, what do different lines mean in general, and thus what do they say about the specific data, such as the mean?

Solution: I will be using the gapminder dataset for this exercise.

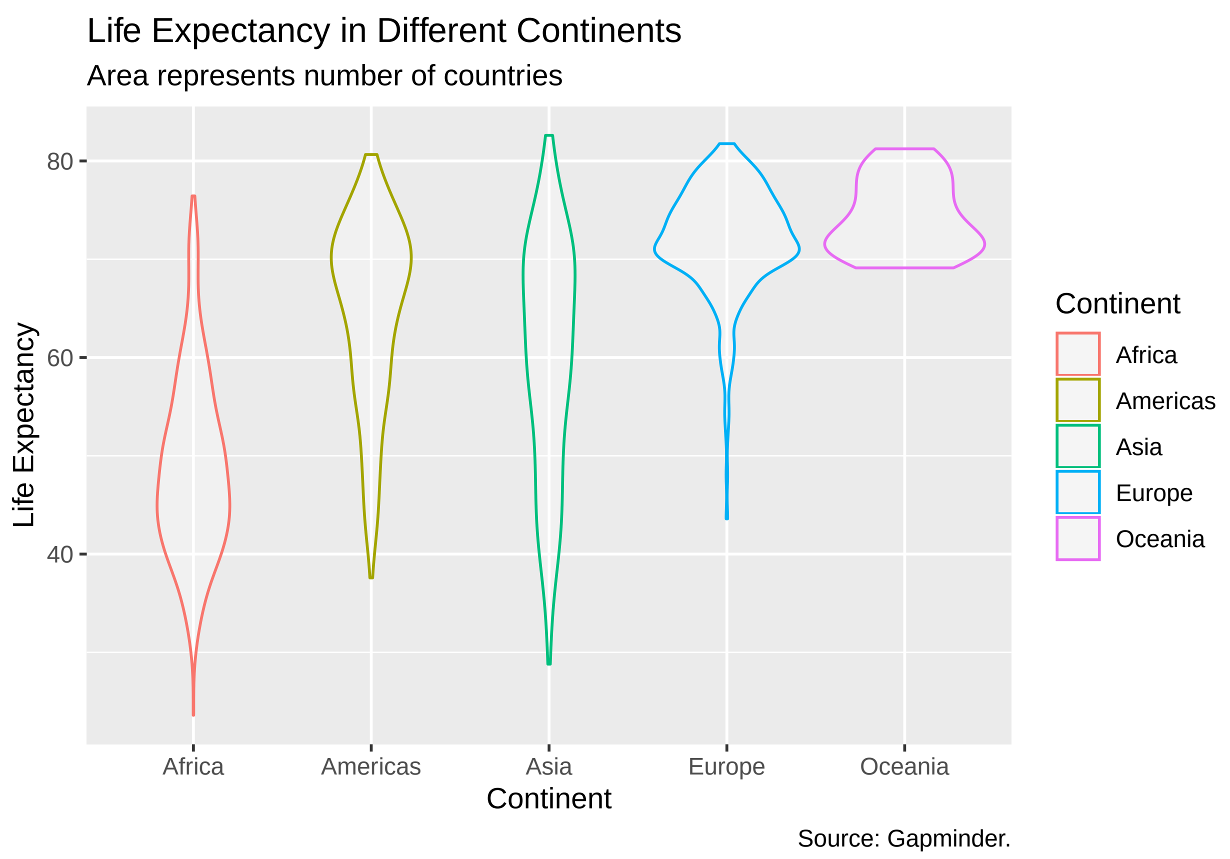
- boxplot

```
box_plot <- ggplot(gapminder, aes(continent, lifeExp, color = continent))

box_plot + geom_boxplot(alpha = 0.3) +

   labs(x = "Continent", y = "Life Expectancy",
        title = "Life Expectancy in Different Continents",
        caption = "Source: Gapminder.", color = "Continent")
```



- violin plot

```
violin_plt <- ggplot(gapminder, aes(continent, lifeExp, color = continent))

violin_plt + geom_violin(scale = "area", alpha = 0.3) +

   labs(x = "Continent", y = "Life Expectancy",
        title = "Life Expectancy in Different Continents",
        subtitle = "Area represents number of countries",
        caption = "Source: Gapminder.", color = "Continent")
```

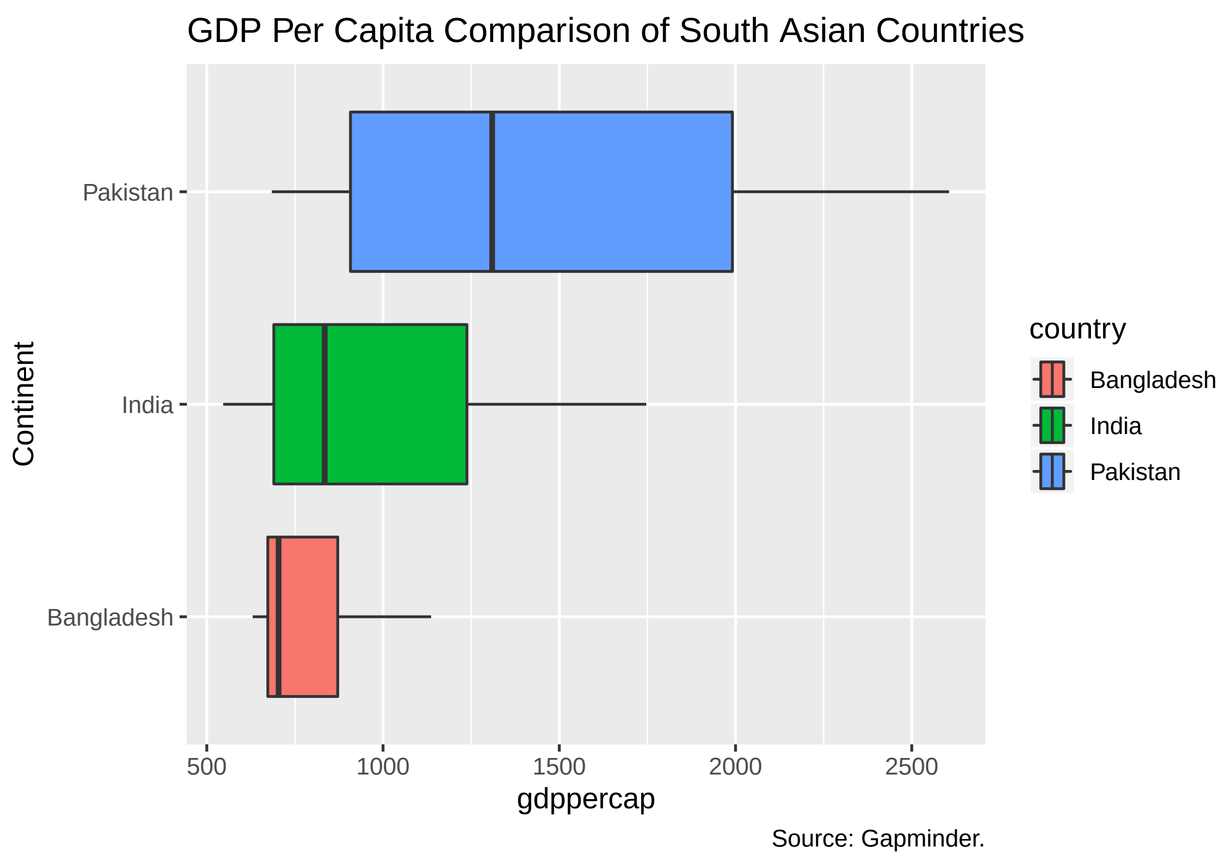


- boxploth (Hint: You need to import the right library for this)

```
library(ggstance)
```

```
##
## Attaching package: 'ggstance'

## The following objects are masked from 'package:ggplot2':
##
##     geom_errorbarh, GeomErrorbarh
```

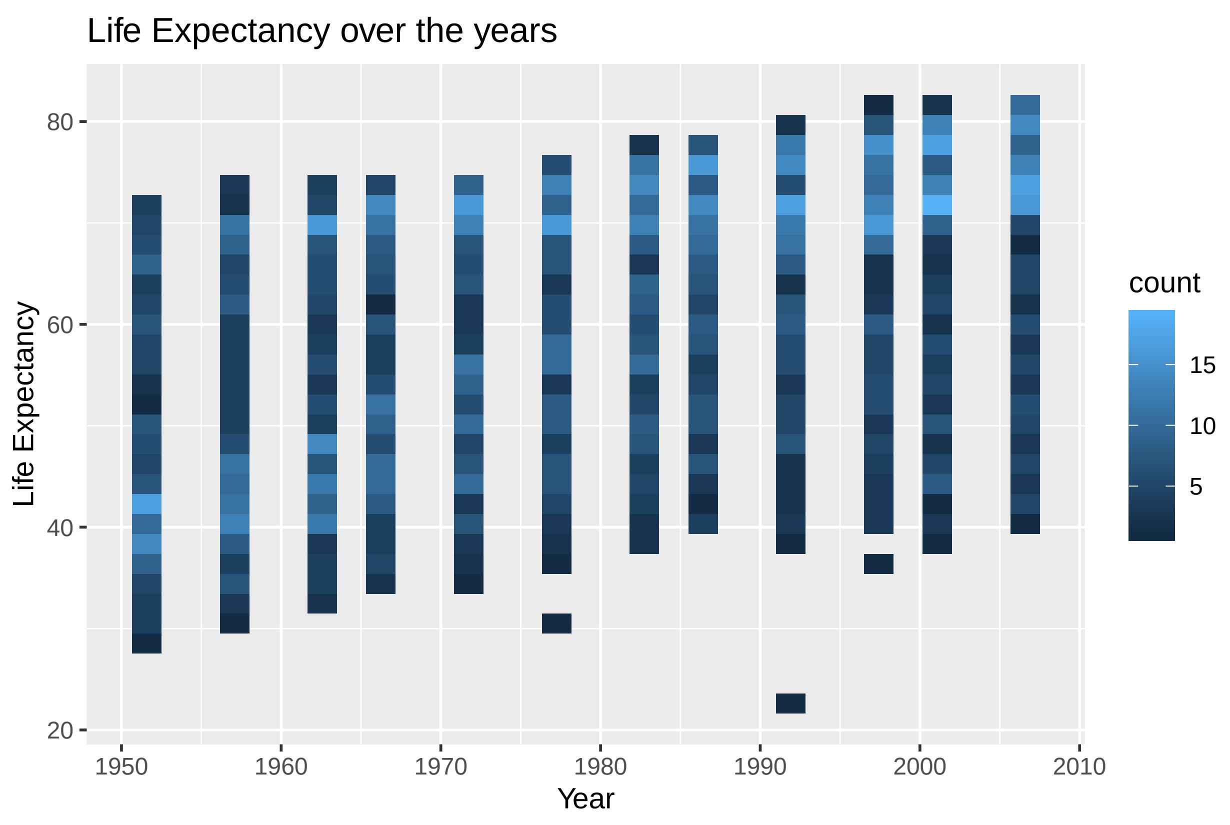
```
gapminder_south_asia <- gapminder %>% filter(country %in% c("India", "Pakistan", "Bangladesh"))

box_ploth <- ggplot(gapminder_south_asia, aes(gdpPercap, country, fill = country))

box_ploth + geom_boxploth(outlier.shape = NA) +

   labs(x = "gdppercap", y = "Continent",
        title = "GDP Per Capita Comparison of South Asian Countries",
        caption = "Source: Gapminder.", color = "Continent")
```



- bin2d

```
bin2d_plot <- ggplot(gapminder, aes(year, lifeExp))

bin2d_plot + geom_bin2d(shape = "Cirle") +

   labs(x = "Year", y = "Life Expectancy",
        title = "Life Expectancy over the years",
        caption = "Source: Gapminder.")
```

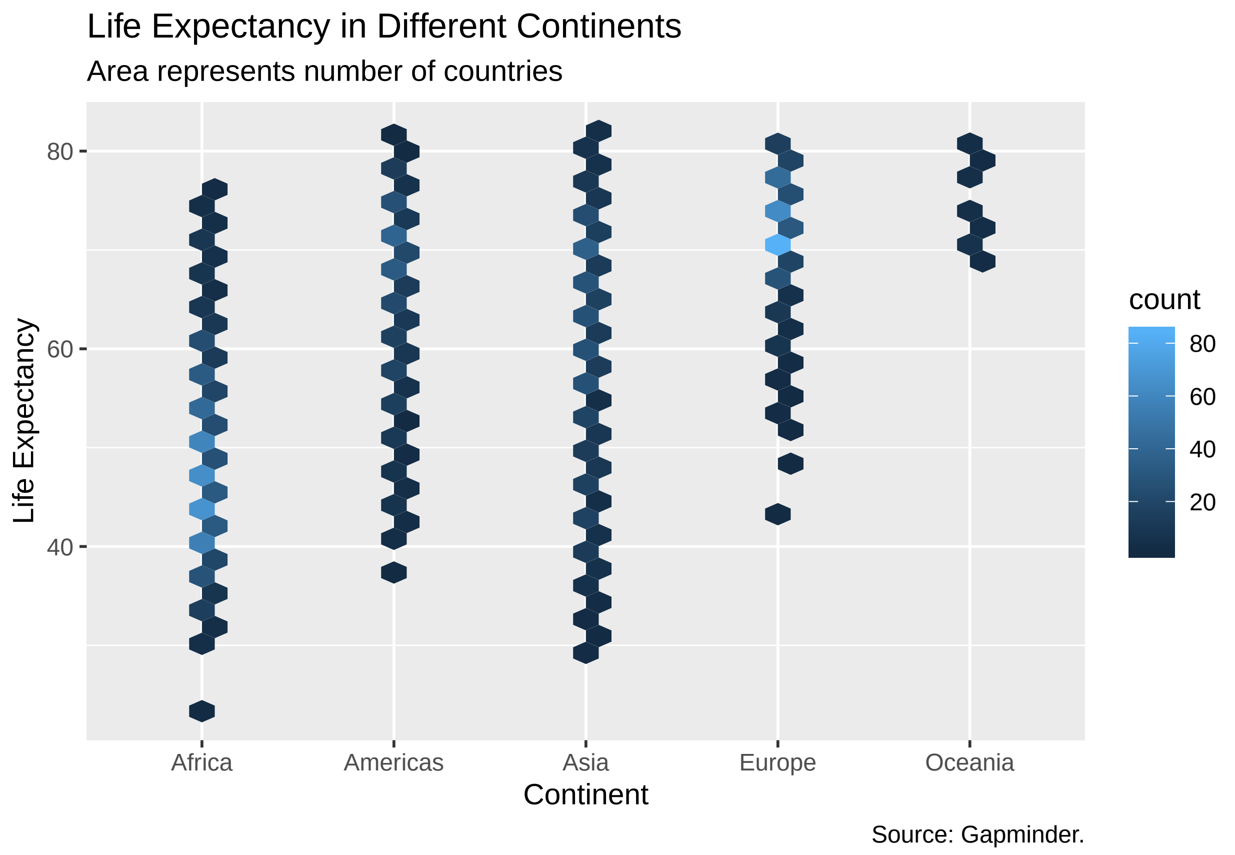
```
## Warning: Ignoring unknown parameters: shape
```



Source: Gapminder.

- hex

```
library(hexbin)

hex_plot <- ggplot(gapminder, aes(continent, lifeExp))

hex_plot + geom_hex() +

   labs(x = "Continent", y = "Life Expectancy",
        title = "Life Expectancy in Different Continents",
        subtitle = "Area represents number of countries",
        caption = "Source: Gapminder.")
```



Thus the code for each plot should be `ggplot(data = <your-data-set>, mapping = aes(...) + geom_...()` and is not the main challenge. The main part is for you to look at and understand the data.

Exercise 5: Come up with an exercise to help you – and others – learn `summarise` and `group_by` better. The more confused you are, the more you should simply try to come up with, or even copy, an example from somewhere and highlight what confuses you. Is it the order or arguments? Their role? If you are less confused, try to find a (non-obvious) use. Mention any resources used.

For this exercise, we are going to track the most dominant teams in the German Football League (Bundesliga) between 1990 till 2010. It's inspired by the Premier League Case Study in Professor Gabor's book.

```
#Loading the data - need to install packages first
#install.packages("SportsAnalytics")
#library(SportsAnalytics)
data(BundesligaFinalStandings)
standings <- BundesligaFinalStandings #named our data standings for convenience
```

Find the average points of each team over the whole time period.

```
standings %>%
  group_by(Team) %>%
  summarise(avg_points = mean(PointsScored, na.rm = TRUE)) %>%
  arrange(desc(avg_points))
```

```
## # A tibble: 36 x 2
##    Team                avg_points
##    <fct>                    <dbl>
##  1 Bayern München            63.2
##  2 Bayer Leverkusen          52.0
##   3 Werder Bremen           52.0
##   4 Borussia Dortmund       51.4
##   5 FC Schalke 04           50.1
##   6 VfB Stuttgart           49.4
##   7 1899 Hoffenheim         48.5
##   8 VfL Wolfsburg           47.4
##   9 Hertha BSC              47.1
##  10 Hamburger SV            46.4
## # ... with 26 more rows
```

Find the frequency for the distribution of the number of wins Borussia Dortmund had across all seasons.

```r
standings %>%
  filter(Team == "Borussia Dortmund") %>%
  group_by(Wins) %>%
  summarise(no_of_seasons = n())
```

```
## # A tibble: 10 x 2
##     Wins no_of_seasons
##    <dbl>         <int>
##  1     9             1
##  2    10             2
##  3    11             2
##  4    12             1
##  5    15             4
##  6    16             4
##  7    18             1
##  8    19             2
##  9    20             2
## 10    21             1
```

In which season and by which team were the highest number of points scored in our dataset?

```r
standings %>%
  group_by(Season, Team) %>%
  summarise(Max_points = max(PointsScored)) %>%
  arrange(desc(Max_points)) %>%
  head(1)
```

```
## # A tibble: 1 x 3
## # Groups:   Season [1]
##   Season    Team           Max_points
##   <fct>     <fct>               <dbl>
## 1 1998-1999 Bayern München         78
```

Exercise 6: Work through sections 11.1 and 11.2 (skip exercises). Skipped as part exercise 0.