Exercise Sheet - Data Manipulation

1. Flights - delay vs. time.

Load the package `nycflights13`, which contains the on-time data `flights`, using the command `require(nycflights13)`. The `flights` data set is about all the flights that departed from New Yord City (i.e. airports JFK, LGA or EWR) in 2013. In particular, the interest lies in the following variables:

- `hour`, `minute`: the hour and minute of the departure
- `arr_delay`: the arrival delay of the incoming plane (in minutes)
- `dest`: the destination.

Load the data set and take a look at it using the commands `str()`, `head()` and `summary()`.

a) Create a new variable which encodes a given `hour` and `minute` as one decimal number, i.e. time in hours. Save this new variable as variable `time` in the data frame `flights`.

b) Calculate the average arrival delay and the number of flights per value of the variable `time` and save them as variables `arr_delay` and `n` in a new data frame named `delay.per.hour`. You can use one of the following possibilities:

- the `dplyr` package
- the "built-in function" `aggregate()` of R
- optional: the `plyr` package, which is not content of the course. You have to search online for hints.

Note that the packages `dplyr` and `plyr` need to be installed first! Also, one can access functions without loading a package: for example, use `dplyr::summarise(...)` instead of `require(dplyr); summarise(...)`.

c) Plot the average arrival delay against the time. Scale the size of the points according to the number of flights. What do you notice?

2. Flights - Dealing with missing values.

Let's find out about an important difference of the function `aggregate()` and the function `summarise()` of the R-package `dplyr` (`dplyr::summarise()`).

a) Redo Exercise 1.b) for the function `aggregate()` and the function `dplyr::summarise()`. Save the two objects as `delay.1` and `delay.2`, respectively. What are the differences?

Hint: Compare the dimensions as well as the first few rows of the two objects.

b) Try to find **two solutions** how one could change the code in order to obtain the same result. Think about how to change the function call of `aggregate()` and how to change the function call of `dplyr::summarise()`. Look at the help file of the functions and study the default arguments.

c) Check that both of your solutions work, i.e. that the corresponding data frames are really equal.

Hint: Use the function `identical()` for each column separately.

3. Flights - Plotting the destinations.

The goal is to explore if there are large differences between destinations regarding arrival delay and number of flights.

We work again with the flights data set in the package `nycflights13` from Exercise 1. If you need to load the data set, use the command `require(nycflights13)`.

a) Calculate the average value of the arrival delay (`arr_delay`) and the number of departing flights (`n`) for each destination (`dest`) and name the resulting data frame `delay.per.dest`. Do the calculation in two different ways:

- by omitting the missing values first:
  - ▷ using `dplyr`
  - ▷ using the "built-in function" `aggregate()`
- by keeping the missing values in the data set:
  - ▷ using `dplyr`
  - ▷ using the "built-in function" `aggregate()`

b) Merge the data frames `delay.per.dest` and `airports` in order to add the coordinates (`lon`, `lat`) of the airports to `delay.per.dest`. The data frame `airports` is included in the package `nycflights13`.

Hint: The hints below describe two different ways of merging two data frames. You can try both or just pick one.

- Using the function `left_join()` from the package `dplyr`

```
delay.per.dest <- dplyr::left_join(x = delay.per.dest, y = ...,
                                   by = c("dest" = "faa"))
```

- Using `merge()`, a "built-in function" in R:

```
merge(x = delay.per.dest, y = ..., by.x = "dest", by.y = "faa",
      all.x = T, all.y = F)
```

c) Create a scatter plot of the latitude against the longitude and scale the points according to the number of departing planes. You can use traditional `graphics` or the package `ggplot2`.

Hint:

- Using traditional `graphics`:

```
plot(... ~ ..., data = ..., pch =  19, cex = n / 6000)
```

- Using `ggplot2`:

```
ggplot(data = ..., aes(..., ..., size = n)) + geom_point()
```

d) BONUS: We continue to work on the plot from part c) which was obtained by using `ggplot2`. Add a map of the US to this plot. Only consider data points with longitude greater than -140. This omits Hawaii and Alaska, which is convenient because they are too far away on the map.
Proceed step by step.

- First install and/or load the package `maps`.

```
require(maps)
```

- Define a subset of the data `delay.per.dest` for which the longitude is larger than -140.

```
delay.per.dest2 <- subset(delay.per.dest, subset = ...)
```

  Alternatively you could use indexing or use the function `dplyr::filter()`
- Define a basic plotting object

```
g <- ggplot(data = ...., aes(..., ..., size = n))
```

- Finally add the points and the US map via following command:

```
g + borders(database = "state", size = 0.5) + geom_point()
```