Exercises - Graphics with ggplot2

1. Cocaine seizures.

The goal is to explore the data set `cocaine` which reports measurements about 3380 seizures of cocaine in the US from 2007.

Load the data `cocaine` using the command `data("cocaine", package = "ggvis")` in R. The data set contains information about

- `state`: the state in which the seizure occured
- `potency`: the purity of cocaine (as percentage)
- `weight`: the weight (in grams) of the seized cocaine
- `month`: the months in which the seizure occured
- `price`: the estimated value (in USD).

Type `str(cocaine)` and `summary(cocaine)` to gain some overview of the data.

a) We will only consider here the seizures with weight below 200 grams. Define a new data frame `cocaine2` accordingly.

Hint: You can use the command `subset()` to consider only part of the data set.

b) In which three states do seizures happen most frequently? Investigate this graphically.

Hint: Create a barplot of the variable `state`.

c) BONUS: Visualize the counts of the seizures by states in increasing order of the counts.

d) Draw a histogram of the variable `weight`. Is there a tendency that seizures with smaller weight happen more often?

e) What does influence the price of cocaine?

Hint: Explore the relation between the variables `potency`, `weight` and `price` using simple scatterplots.

f) BONUS: We can even visualize the three variables together in one plot. Create a scatterplot of the variable `price` against the variable `potency`. Color the points according to the variable `weight`. Beautify the plot by making the points half-transparent. Finally, it might be useful to log-transform the color scale.

Hint: Use the argument `color = ...` of the function `aes(...)` to color the points. Further try to use the functions

```
...+ geom_point(alpha = ...) + scale_color_gradient(transf = ...)
```

for a nicer visualization.

g) Plot `potency` against `weight`. Add a smoother to the scatterplot. What do you notice?

Hint: Use the function `...+ geom_smooth()` for smoothing.

2. Flights data.

The aim of this exercise is to check if there is a relation between the average arrival delay and the time of departure of planes.

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 departing from one of the airport of New York 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.

a) Let's look at the average arrival delay for a given departure time of the day (`hour` and `minute`). For this purpose create a new variable which encodes a given `hour` and `minute` as one decimal number and call this new variable `time`. Thereafter calculate the average arrival delay per value of the variable `time` and save it in a new data frame named `delay.per.hour`.

Hint: First create the new variable `time` with the command

```
flights$time <- flights$hour + flights$minute / 60
```

Use the following function call in order to calculate the average delay per value of `time`:

```
aggregate(formula = arr_delay ~ time, data = flights, FUN = ..., na.rm = ...)
```

b) Plot the average arrival delay against time. What do you conclude?

c) Scale the points in the plot by the number of planes `n` which departed at a particular time of the day. The variable `n` needs to be calculated and added to the data set `delay.per.hour` which you defined in task a). Why is this plot more informative then the one of the previous subtask?

Hint: The variable `n` can be calculated as in task a) if you slightly change the function call of `aggregate()`. Use the argument `FUN = length`.

d) BONUS: Plot only the observations with value of `n` larger than 50 and scale the points according to their area instead of their diameter.

Hint: Look at the function `scale_size_area()`.

e) BONUS: Redo task a) and c) using the package `dplyr` or `plyr` instead of using the function `aggregate()`. That's not content of the course and you have to search online for hints if you are not familiar with the two packages.

3. Flights data, continued.

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

We work again with the `flights` data set in the package `nycflights13` from Exercise 2. If you need to reload the data set, use the command `require(nycflights13)`.

a) Calculate the average value of the arrival delay `arr_delay` for each destination (`dest`). Omit all the missing values in the calculation.

Hint: Use the function `aggregate()`. The argument `na.rm = TRUE` of the function `mean()` allows to omit missing values in the calculation of the mean. Note that the function `aggregate()` creates a dataframe with first column corresponding to the grouping variable (here `dest`). Save the output of the function `aggregate()` as a new data frame `delay.per.dest`.

b) Calculate the number of planes departing to each destination. Add those counts as variable `n` to the data frame `delay.per.dest`.

Hint: Use again `aggregate()` but only save the second column of the output.

c) 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: Use the function

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

Look at the help file of the function `merge()` by typing `?merge` to understand what the different arguments mean.

d) Create a scatterplot of the latitude against the longitude and scale the points according to the number of departing planes.

Hint: Use the argument `size = ...` in the function `aes()`.

e) Moreover, color the points by the value of the average arrival delay. What do you notice?

f) BONUS: Now scale the area of points to be proportional to the number of departing planes. The diameter of the points is by default proportional to `size` argument.

Hint: Look at the function `scale_size_area()`.

g) BONUS: Add a map of the US to the plot from part d). We only consider data points with longitude greater than -140. This omits Hawai and Alaska which is convenient because they are too far away on the map.

Hint: The database `states` can be found in the packages `maps` and can be used to add a map of the US to the plot. Use the function

```
...+ borders(database = ..., size = 0.5)
```

to add the map. The argument `size` changes the width of the border lines.

4. Gapminder: Fact-based world view.

Gapminder Foundation wants to give access to a fact-based world view in order to promote a sustainable global development. For more information and entertaining videos; see `http://www.gapminder.org/`. The aim of this exercise is to obtain a nice visualization of the life expectancy vs. the GDP per capita. This is achieved by successively adding more functions to a basic function call.

Load the package `gapminder` using the command

```
require(gapminder, quietly = TRUE)
```

which contains the data set `gapminder` and the vector `country_colors`. Take a first look at the data by looking at the help files by typing the commands `?gapminder`, `?country_colors` and `str()`. Consider first the data set `gapminder`.

a) Let's pick one particular year of the data set `gapminder`. Use the function `subset()` in order to extract the observations of the year 2002. Create a scatterplot of `LifeExp` against `gdpPercap`.

b) Use the variable `country` in order to color the points of the plot. Scale the points by the square root of the `pop` and omit the legend.

Hint: Use the function

```
...+ geom_point(aes(size = ...), pch = 21, show_guide = ...)
```

c) Reproduce the same plot using a log-scale for the x-axis.

d) Now make the size of the points a bit larger. The size of the points should range from 1 to 40.

Hint: Use the function

```
...+ scale_size_continuous(range = ...)
```

e) Color the points in a way that you can distinguish between the different continents. Consider the vector `country_colors` which provides a color encoding for the continents.

Hint: Use the function

```
...+ scale_fill_manual(values = ...)
```

f) Use facetting to create a separate plot for each continent.

Hint: Use the function

```
... + facet_grid( . ~ ...)
```

5. Napoleon

The goal is to reproduce the Russian campaign of Napoleon in 1812 as did Charles Minard on his chart in 1869. In particular, we will be able to visualize the decreasing number of soldiers and their movements. You can find the Charles Minard's chart under the following link: `https://en.wikipedia.org/wiki/File:Minard.png`.

The data can be found in the package `HistData` in R. The data sets of interest are called `Minard.troops` and `Minard.cities`. Load the data and take a look at the structure of the two data sets using the command `str()`.

a) Plot the x- and y- coordinates from the data set `Minard.troops` and add a path with adjusted `size` according to the number of survivors and with `color` according to the direction. Since there are three groups of soldiers at the start, indicate this with the argument `group = ...` . Save the plot for further use in the object `plot_troops`.

Hint: Use the function

```
...+ geom_path(aes(...), lineend = "round")
```

b) Add the cities as labels to the plot `plot_troops` from part a). Store the new plot as `plot_both`.

Hint: The cities can be found in the data set `Minard.cities`. Use the function

```
... + geom_text(aes(labels = ...), size = ..., data = ...)
```

to add some text to the plot. Adjust the `size` to 4.

c) Change the colors of the two directions of the path and store the plot in the object `plot_polished`. The forward path should be red and the return path should be grey. Moreover, remove the x- and y- labels of the plot.

Hint: Use the function

```
...+ scale_color_manual(values = ...) + xlab(NULL)
```

to set colors manually and to remove the x-labels, respectively.

This lab is based entirely on Nina Roth's and Claude Renaux's
Seminar for Statistics originally developed at ETH Zurich