Summary of R course by IME

Disclaimer: This is not a very good summary of the course provided by IME, but rather served as way for me to get familiar with ‘rmarkdown‘ :)

Basics

- **Creating and modifying a vector**

```
x <- seq(1,4,by = 1)
x[c(1,2,3)] = 5
x
```

```
## [1] 5 5 5 4
```

```
#or alternatively

x <- 1:4
x[1:3] = 5
x
```

```
## [1] 5 5 5 4
```

- **Vector multiplication**

```
x <- 1:5

y <- 6:10

a <-  t(x) %*% y
```

- **order() and sort()**

```
#order() creates a permutation vector and can be used to sort data.frames
sort(x) == x[order(x)]
```

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

```
rev(sort(x)) == x[order(-x)]
```

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

- **Factors**

```
gender = factor(c("male", "female", "female", "male"))

# Look at it and make a summary table
gender
```

```
## [1] male   female female male
## Levels: female male
```

```
table(gender)
```

```
## gender
## female   male
##      2      2
```

```
#Find number of males
sum(gender == "male")
```

```
## [1] 2
```

- **Matrices**

```
#Creating matrices
A <- matrix(1:6, nrow = 2, ncol = 3, byrow = TRUE)

x1 <- 1:3
x2 <-  c(7, 6, 6)
x3 <-  c(12, 19, 21)

# Bind vectors x1, x2, and x3 column-wise into a matrix
C <-  cbind(x1, x2, x3) # Bind vectors x1, x2, and x3 column-wise into a matrix


# Bind vectors x1, x2, and x3 row-wise into a matrix.
R = rbind(x1, x2, x3) # Bind vectors x1, x2, and x3 row-wise into a matrix.
```

```
# Here are some other useful  matrix commands

dim(A) # get the dimensions of a matrix
nrow(A) # number of rows
ncol(A) # number of columns
apply(A, 1, sum) # apply the sum function to the rows of A
apply(A, 2, sum) # apply the sum function to the columns of A
sum(diag(A)) # trace of A

A = diag(1:3) # a 3 by 3 diagonal matrix with entries 1, 2, 3
solve(A) # inverse of A, in general solve(A,b) solves Ax=b wrt x
det(A) # determinant of A
```

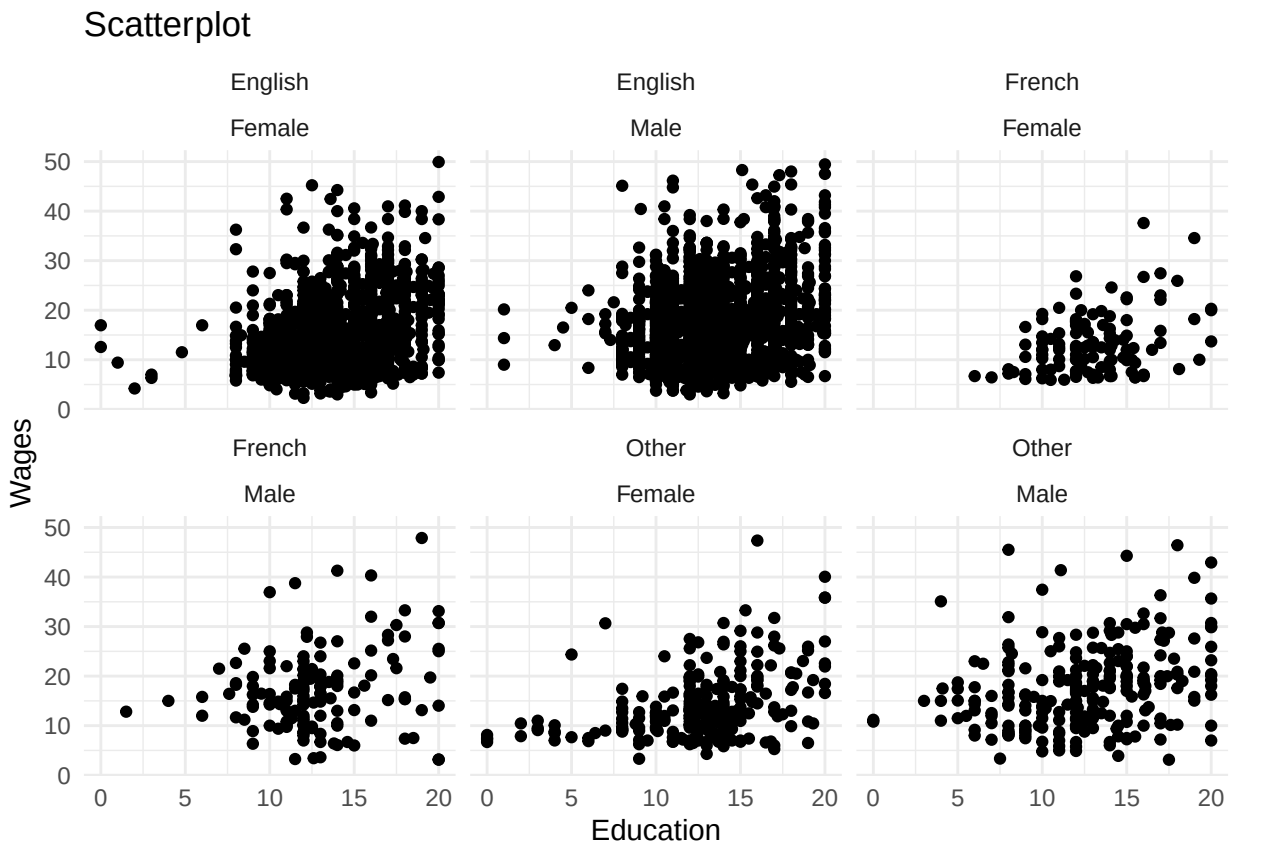
Plotting (ggplot)

ggplot2 is a package which offers an alternative way to plot data as opposed to the standard `plot()` function. What follows is an example of using ggplot with the dataset `SLID`.

```
#Setup
library(ggplot2)
library(car) #Contain SLID
SLID = na.omit(SLID) # We only use rows without missing values
```

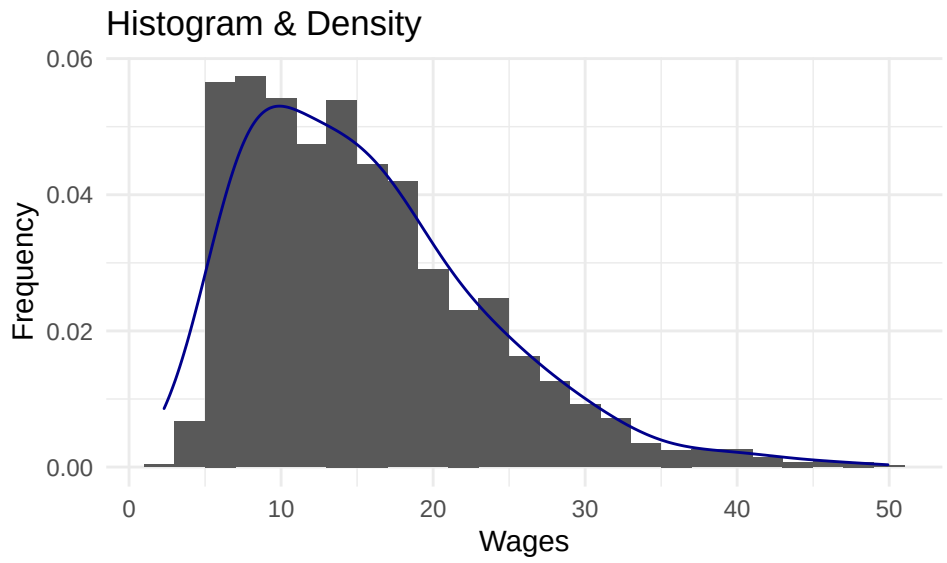
- **Scatterplots**

```
ggplot(SLID, aes(education, wages)) + #aes(x,y) plots y against x.
  geom_point() +
  labs(title = "Scatterplot") +
  xlab("Education") +
  ylab("Wages") +
  theme_bw() +
  facet_wrap(~language + sex) + #facet_wrap divides the dataset into langauge/sex pairs.
  theme_minimal()
```



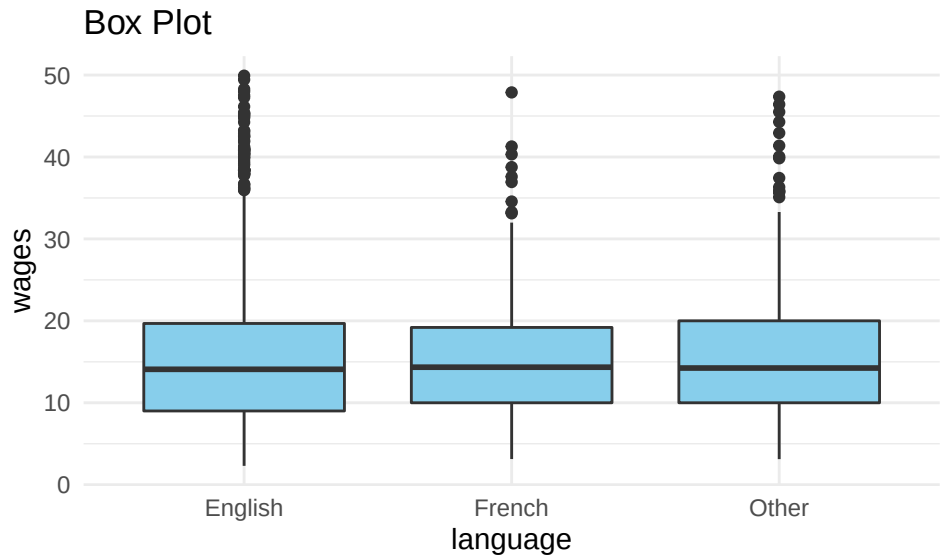
- **Histogram and density plots**

```
ggplot(SLID, aes(wages)) +
  geom_histogram(binwidth = 2,aes(y=..density..)) +
  geom_density(color="darkblue",adjust = 2) +
  labs(title = "Histogram & Density") +
  xlab("Wages") +
  ylab("Frequency") +
  theme_minimal()
```



- **Boxplot**

```
ggplot(SLID, aes(x=language, y=wages)) +
  geom_boxplot(fill = "skyblue") +
  labs(title = "Box Plot") +
  theme_minimal()
```



- **Pairs plot**
  As opposed to the standard `pairs()` function, ggplot offers a more powerful version, which requires the library `GGally`:

```
library(GGally)
ggpairs(SLID) + theme_minimal()
```

Data manipualtion with `dplyr`

`dplyr` is a grammar of data manipulation which is a part of the tidyverse.

```r
#Setup
library(tidyverse)
library(nycflights13)
```

- **glimpse() and selelct()**

```r
data(storms) # load data

# The storms data is a so-called tibble (https://tibble.tidyverse.org/),
# which we can quickly summarise by using the function glimpse()
glimpse(storms)

# select the two columns called storm and pressure.
# The first argument in the select() function is always the data
# (note that the storm column is called "name" in the dataset we use)
select(storms,name,pressure)

# select all colums except name
select(storms,-name)

# select all columns between the column "name" and "hour"
select(storms,name:hour)
```

- **filter()**
  Chooses all rows that fulfill the prescribed conditions:

```r
# We can filter out all the rows that fulfill the condition of wind>50.
# Again, the dataset is the first argument in the filter() function;
# this leaves 4163 rows in the dataset
filter(storms,wind>50)

# Filtering for more than one condition;
# Here we filter for wind speet >50 and three storms;
# This leaves only 105 rows:
filter(storms,wind>50,name %in% c("Alberto","Alex","Allison"))
```

- **mutate()**
  Adds new columns to the existing dataframe:

```r
# Here we add a new column to the strom data, called "ratio".
# It contains the ratio between the pressure and the wind columns:
mutate(storms,ratio=pressure/wind)

# We can even add another column at the same time,
# which is a transformation of a new column generated earlier in the same call:
mutate(storms,ratio=pressure/wind, inverse = 1/ratio)
```

- **summarise()**
  Summaraises the data in some way:

```r
# We can for example calculate median and variance of all wind speeds
summarise(storms, median=median(wind),variance=var(wind))
```

```r
# Alternatively, we can "pipe" the dataset as follows
storms %>% summarise(median=median(wind),variance=var(wind))
```

- `arrange()`
  Rearranges the order of the rows depending on some values of the columns.

```r
# To sort the storms data in increasing order of wind speed:
arrange(storms,wind)

# To sort the storms data in decreasing order of wind speed:
arrange(storms,desc(wind))

# To sort the storms data in increasing order of wind speed, and within the same
# wind speed, order according to year (it's like a hierarchicay way of ordering)
arrange(storms,wind,year)
```

- **The pipe operator %>%**

```r
# As an example, the following two ways of using select are equivalent
select(storms,name,pressure)
storms %>% select(name,pressure)
# So storms is taken as the first argument in the following command select()

# The same holds for all the other examples, e.g.
filter(storms,wind>50) # is the same as
storms %>% filter(wind>50)

# Piping becomes most useful if we pipe sequentially several times, for example
storms %>%
  filter(wind>50) %>%
  select(name,pressure)

# The above command first filters the wind speeds >50 and from that filtered version
# of the data, selects the columns name and pressure. We could do the same by
# nesting the commands into each other:

select(filter(storms,wind>50),name,pressure)

# but that quickly becomes very confusing...


# Another example where we first add the new column ratio, and then select the name
# and ratio columns from that dataset
storms %>%
  mutate(ratio=pressure/wind) %>%
  select(name,ratio)
```

- `group_by()` + `summarise()`

```r
# For better illustration we reduce our storms data to only three storms
storms.small <- storms %>% filter(name %in% c("Alicia","Barry","Ernesto"))
```

```r
# The following code gives the mean and variance of wind speed measured in each
# storm, and the number of measurements taken (n)

storms.small %>%
  group_by(name) %>%
  summarise(mean=mean(wind),variance=var(wind),n=n())

# Example where we group for two variables, name of the storm and month
storms.small %>%
  group_by(name,month) %>%
  summarise(mean.wind=mean(wind))

# Each time we summarize we remove one layer of our grouping
storms.small %>%
  group_by(name,month) %>%
  summarise(mean.wind=mean(wind)) %>%
  summarise(mean.wind=mean(mean.wind))
```