Chapter 1

ggplot2

1.1 ggplot2

ggplot2 is a very popular package that offers enhanced plotting functionality. It uses a clever way to map numbers to visual representations in plots, ("Grammar of Graphcis"). Visual representations include for example lines, bars, dots. By varying attributes such as colour, shape or size, even more information can be visualized. *ggplot2* uses the following conceptual framework:

- *Data*: is passed as a dataframe to the *ggplot()* function.
- *Geoms*: points, lines, bars etc are the types of geometric objects that are added in layers to the plot to represent the data
- *Aesthetics*: these serve as the "connection" between the geoms and the data. They map visual properties of the geom, such as (x,y) coordinates, line colour, shapes etc to the variables represented by columns of the dataframe. For example, the y coordinate of lines points may be mapped to a variable in the dataframe.

ggplot2 plots always start with the `ggplot()` function Then, geoms can be added to the plot by putting a '+' after the () brackets, followed by the geom functions. Geom functions start with `geom_`, for example `geom_line`. The data and the aesthetics are specified as arguments of either the `ggplot()` or the `geom_` functions. The data is provided using the 'data=' argument, and the aesthetics using the 'mapping=' argument followed by the `aes()` function.

If the data and mappings are provided in the `ggplot()` function, they are globally available in all the following `geom_` functions. If provided in a `geom_` function, the data and the aesthetic are defined only for the respective geom. The data for ggplot should be formatted in long format.

ggplot2 way of plotting enables to quickly construct complex plots because of flexibility and modularity.

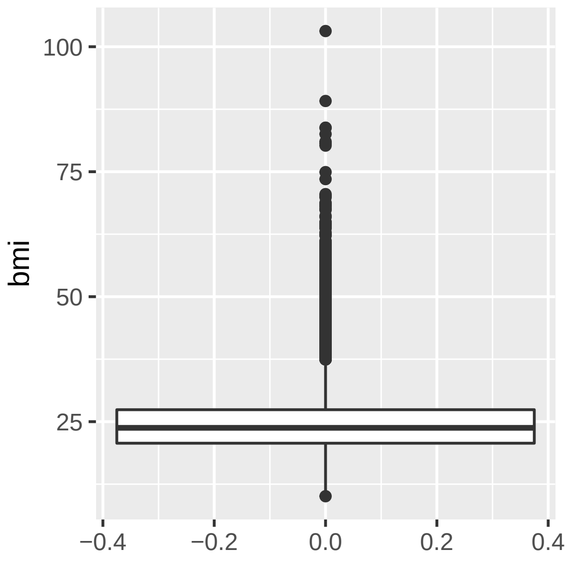
1.2 Plotting distribution of single variables

At the beginning of a statistical analysis it is useful to get an idea of what values are present how hoften in the data for the different variables. This for example could help to identify outliers, to check assumptions about normally distributed data, or simply to verify if the data has been loaded correctly.

In a first example we create boxplots. Note the `aes()` function that designates *bmi* to be the value that is visualized on the y axis.

```
library(ggplot2)
d <- read.csv("../data/BackPain.csv")

ggplot(data = d, mapping= aes(y = bmi))+
  geom_boxplot(na.rm = T) # boxplot, remove bmi values that are NA
```

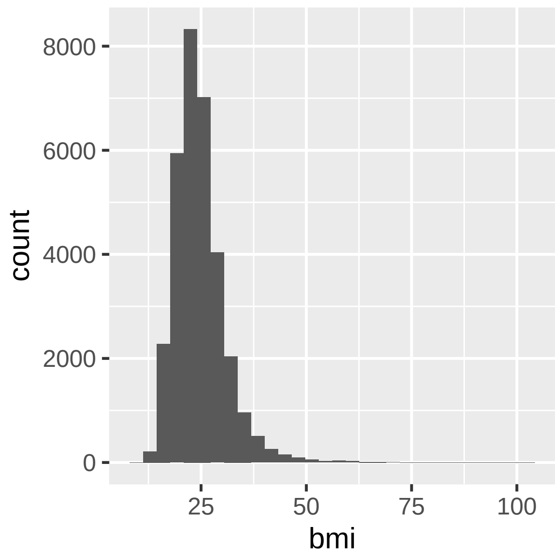


The data and the aesthetic mappings are globally defined in the function `ggplot()`. The `geom_boxplot()` function automatically applies this mapping to the provided data. In addition, NAs are are removed

Next, consider a histogram that may offer an even better picture of a distribution.

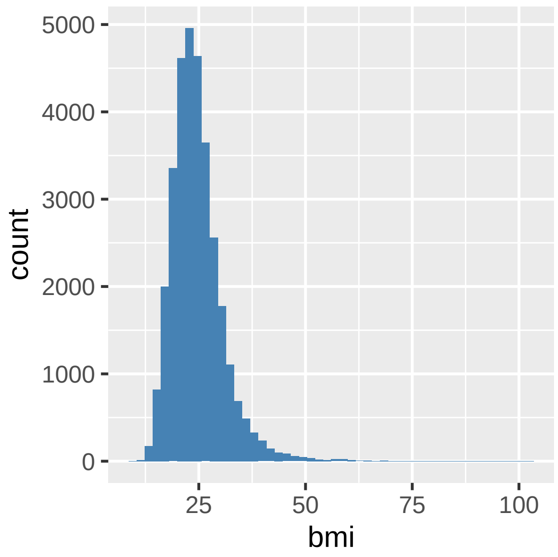
```
# complains about binwidth
ggplot(data = d, mapping= aes(x = bmi))+
  geom_histogram(na.rm = T)

'stat_bin()' using 'bins = 30'.  Pick better value with 'binwidth'.
```



If you try to run this code, you would see that `ggplot` complains that you did not pick a good value for 'bindwidth'. Lets correct that and manually set a fill color. If you manually set visual properties such as the color, that is without mappings, this should generally happen directly in the `geom` functions outside the `aesthetic()` functions:

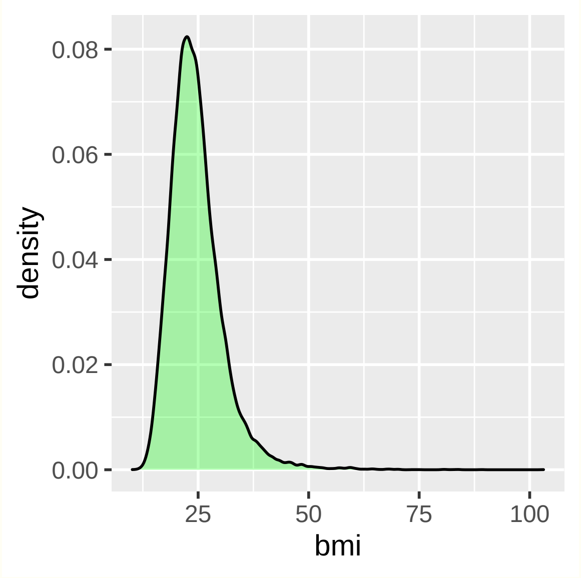
```
# manually define color and set number of bins
ggplot(data = d, mapping= aes(x = bmi))+
  geom_histogram(na.rm = T, bins = 50, fill = "steelblue")
```



`geom_density()` works similarly as histograms and displays "smooth" histograms.

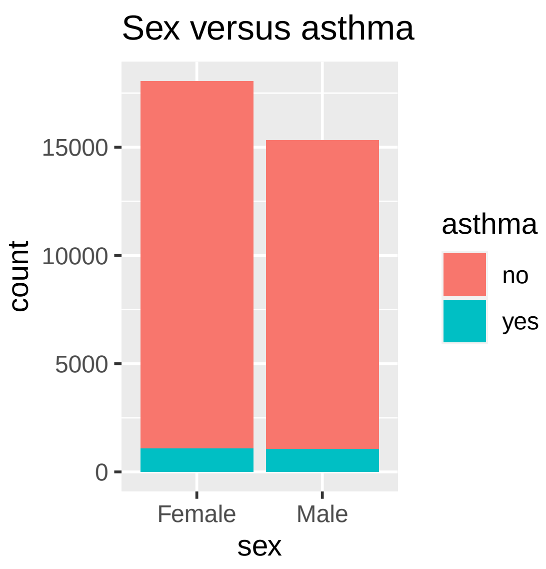
Own experimentation

Try to use `geom_density()` to make a density plot of bmi, like so (Hint: set the *alpha* argument):



Now we use barplots to demonstrate how to group sex by colors and how to manually set a title:

```
# A barplot and how to set a plot title
ggplot(data = d[d$asthma != "",], mapping= aes(x = sex, fill = asthma))+
  geom_bar(na.rm = T) +
  ggtitle("Sex versus asthma")
```



1.3 Relationship of variables

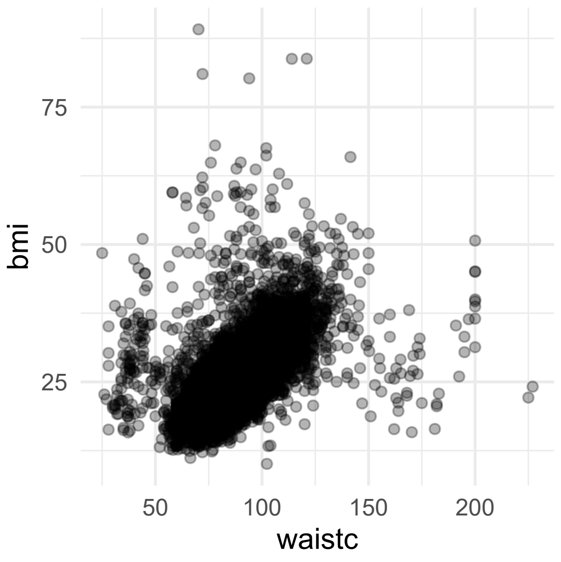
Relationship between different variables are often at the heart of statistical analyses. Visualization of these relationships is crucial to communicate results in an intuitive, understandable manner.

First we prepare a subset of the data to avoid having to deal with NA, and we infer the weight from `bmi` and `height`:

```
ds = d[,c("age", "bmi", "waistc", "height", "sex")] # the columns we want to work with
ds$weight <- ds$bmi * (ds$height/100)^2 # recover weight value using the BMI formula
ds <- ds[complete.cases(ds),] # get rid of rows with any NA
```

We start with a simple scatterplot. GGplot also provides several themes to alter the general look of plots. Here we use the minimal theme, which provides a white background. We set *a*lpha to 0.3 to better handle overlapping dots by making them transparent:

```
# A points or scatterplot with the 'minimal' theme
ggplot(aes(x = waistc, y = bmi), data = ds)+
  geom_point(alpha = 0.3)+
  theme_minimal()
```

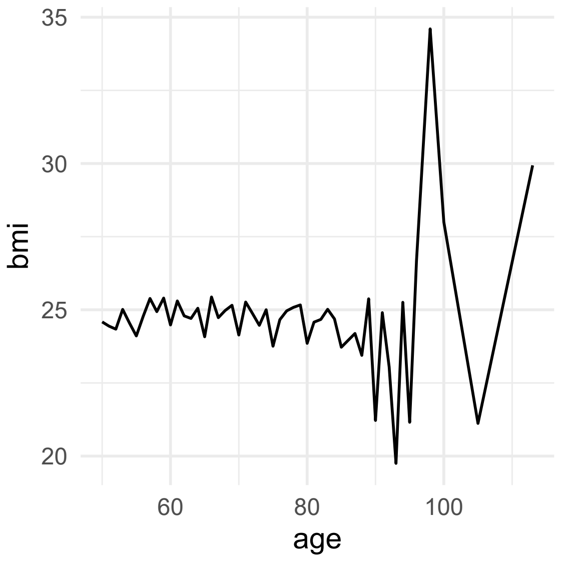


Bmi and waist circumference seem to correlate pretty well.

If we want to plot the mean bmi by age, we first need to calculate it. For that we use the *a*ggregate() function that works similararly to *x*tabs(). It creates stratified data according to the formula provided and then applies the provided function to each cell. Here, we categorize the data by age, and then apply the *m*ean() function for every strata. The stratification here uses that age is an integer variable. The plot is then stored in the gg object before displaying.

```
da = aggregate(bmi ~ age, ds, mean) # mean of bmi by age (mean function passed as object)
gg = ggplot(aes(x=age, y = bmi), data = da)+
  geom_line() # line plot

# only now plot the plot and apply the theme
gg +  theme_minimal()
```

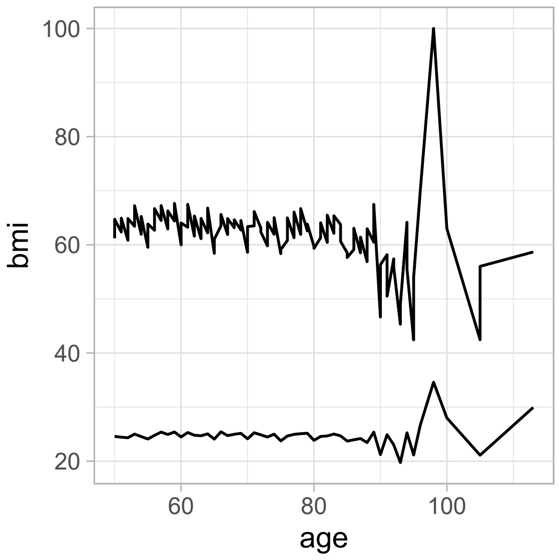


Above an age of 90, the data becomes scarce and the mean is influenced by stark variation.

Thanks to the modular nature of ggplot, it is easy to add things to existing plot. Lets add a line for the weight and change the theme. Note that the new data needs to be provided to the `geom_line()` seperately.

```
# calculate mean weight for every age
da2 = aggregate(weight ~ age+sex, ds, mean)

# add a new line based on the new data to the old plot
gg+geom_line(aes(x = age, y = weight), data = da2)+
  theme_light()
```



1.4 Scales and legends

ggplot2 of course also offers large flexibility in customizing how x and y axes, as how well as the plot legend look like. To do that, ggplot2 uses *scales*. Scales define how the data is mapped to aesthetics and how the axes of the plot look like.

There are different type of scales in ggplot2. The main distinction is into continuous and discrete scales.

1.4.1 Continous scales

`scale_x_continous()` and `scale_x_continous()` are the principal functions to modify continuous scales, for x and y axes respectively. Here a list of useful arguments:

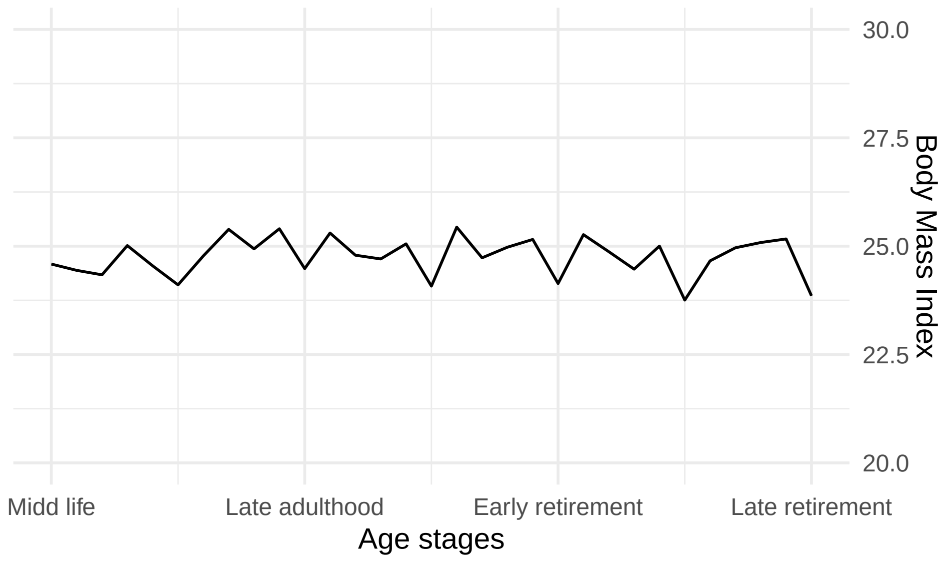
- name: Specifies the axis title
- breaks: Specifies which values are denoted with axis tickmarks in the form of a vector.
- labels: The names for the breaks.
- limits: Takes a vector of length 2 whose values indicate the beginning and the end of the axis
- trans: Sets a transformation for this axis. Valid values are for example "log", "sqrt", ""reverse", "logit". (See the documentation for a full list)
- position: Defines the position of the axis. For y, it is either "left" or "right, for y either "top" or "bottom".

The full list of arguments is accessible in the documentation.

For an example, consider again the plot of average weight by age. Lets restrict the plot to ages <= 80 years by customizing the axis. Also set titles and restrict the y axes to a bmi of 20 to 30.

```r
# restrict x axis and set manual axis labels using scales
gg + scale_x_continuous(breaks = c(50,60,70,80),
                        labels = c("Midd life", "Late adulthood",
                                   "Early retirement", "Late retirement"),
                        limits = c(50, 80),
                        name = "Age stages")+
     scale_y_continuous(limits = c(20,30),
                        position = "right",
                        name = "Body Mass Index") +
     theme_minimal()

Warning:  Removed 20 row(s) containing missing values (geom_path).
```



ggplot displays a warnign for the values that did not fit within the axis limits.

Note that this way of denoting the x axis may not result in a useful plot, it is likely that direct age numbers are more informative.

Alternatively, the `ylab()`, `xlab()` and `lims()` function can be used to specify axes names and limits.

Own experimentation

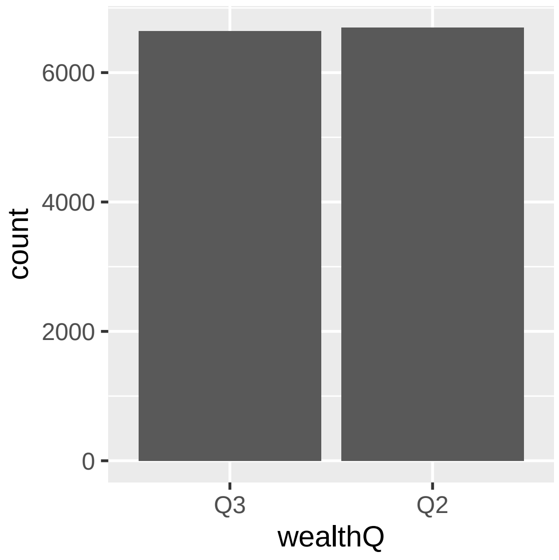
Use the `labs()` and `xlims()` functions to create a plot of average waist circumference for the ages 55 to 75 with a title and a subtitle.

1.4.2 Discrete scales

Axis manipulation of categorical plots is somewhat different. Instead of modifying the tickmarks, ggplot2 allows to manipulate which categories are displayed and how they are displayed. For example, lets create a barplot limited to *Q2* and *Q3* of *wealthQ*:

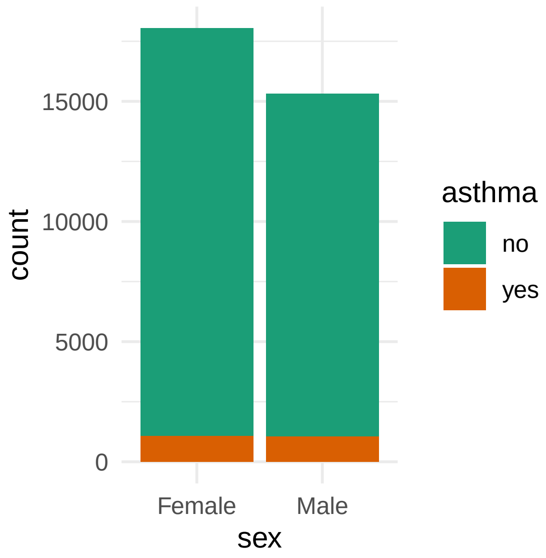
```r
# only display two levels of categorical data
ggplot(data=d, mapping=aes(x=wealthQ))+
  geom_bar()+
  scale_x_discrete(limits=c("Q3","Q2"))

Warning:  Removed 20782 rows containing non-finite values (stat_count).
```



Consider again the barplot asthma and sex. In that case, ggplot uses discrete scales to map sex and asthma to the aesthetics of the bar geom. To modify the color, we need to change this discrete scale:

```
# use a different color palette by altering the fill scale
ggplot(data = d[d$asthma != "",], mapping= aes(x = sex, fill = asthma))+
  geom_bar(na.rm = T)+
  scale_fill_brewer(type="qual", palette = 2)+
  theme_minimal()
```



We specifically use `scale_fill_brewer` specify a predefined color palette. This function provides access to predefined color palettes from ColowBrewer. See for a list.

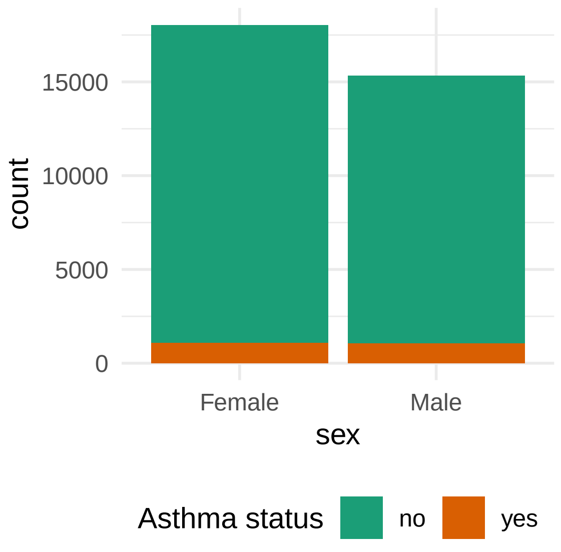
Own experimentation

Use the `scale_color_manual()` function to define two custom colors for the above plot. Hint: You need to use the *values* argument and set the *aesthetics* argument to "fill" to select the corect aesthetic.

1.4.3 Legends

Another concept in ggplot2 are *guides*. Guides are are the legends for the different aesthetic mappings. They can be specified either in the scale functions or as a seperate part of a ggplot using the `guides()` function. In the following example we change the legend title from "asthma" to "Asthma status" and change the direction of the labels in the legend from vertical to horizontal, in the same barplot example as above. To make space for the vertical legend, we also need to position the legend below the plot, for this we need to use a theme option:

```r
# an example using guides to change the legend title
ggplot(data = d[d$asthma != "",], mapping= aes(x = sex, fill = asthma))+
  geom_bar(na.rm = T)+
  scale_fill_brewer(type="qual", palette = 2)+
  guides(fill = guide_legend(title = "Asthma status", direction = "horizontal"))+
  theme_minimal()+
  theme(legend.position = "bottom")
```



Many more options are available to customize and style legends and set legend types. See the ggplot2 documentation for details.