Exploring data with summary statistics

Video lecture 1

Now that we have seen how to explore a dataset with graphs, we will focus on the summary statistics. Summary statistics are simple measurements of an entire dataset used to overview its characteristics. There are no tests conducted when reporting summary statistics, and no application of probability theory. Instead, we report basic traits such as:

- the minimum and maximum
  - the lowest and highest values observed, respectively
- the range
  - the difference between the highest and lowest values
- the mean
  - the sum of the values divided by the total number of values
- the standard deviation
  - a measure of dispersion of a set of data from its mean
- the median
  - the middle value, in a sorted, ascending or descending list of values
- the xth percentiles, for any value of x between 0 and 100
  - the value at which x% of the ordered values are below
  - the median is the 50th percentile.
- the interquartile range
  - the difference between the 75th and 25th percentiles
- the frequency of certain values
  - the number of samples which correspond to a specific value of a variable.
  - frequencies are only used for categorical or discrete values

Load the packages

This is where we load the packages we need before we start analysing data. Here we will use the **library()** function to load the tidyverse package, and the NHANES health data set we used in the previous session.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.0     v forcats 0.5.1

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

```
library(NHANES)
```

The NHANES dataset

This is survey data collected by the US National Center for Health Statistics (NCHS) which has conducted a series of health and nutrition surveys since the early 1960's. Since 1999, approximately 5,000 individuals of all ages are interviewed in their homes every year and complete the health examination component of the survey. Find out more here, in particular check the NHANES.pdf reference manual.

First glance at the dataset

When we look at an R object, it's very useful to look at it. We can use RStudio to View the object either by clicking on the object name in the Environment tab in the top-right pane. Or we can use the **View()** function in the console. This opens a new tab in the top-left pane where you can see the whole object. NHANES is a data frame with many rows (observations / individuals) and columns (variables).

```
View(NHANES)
```

We can also look at the first or last few elements of an R object by using the **head()** and **tail()** functions, respectively.

```
head(NHANES)
```

```
## # A tibble: 6 x 76
##      ID SurveyYr Gender   Age AgeDecade AgeMonths Race1 Race3 Education
##   <int> <fct>    <fct>  <int> <fct>         <int> <fct> <fct> <fct>
## 1 51624 2009_10  male      34 " 30-39"        409 White <NA>  High School
## 2 51624 2009_10  male      34 " 30-39"        409 White <NA>  High School
## 3 51624 2009_10  male      34 " 30-39"        409 White <NA>  High School
## 4 51625 2009_10  male       4 " 0-9"           49 Other <NA>  <NA>
## 5 51630 2009_10  female    49 " 40-49"        596 White <NA>  Some College
## 6 51638 2009_10  male       9 " 0-9"          115 White <NA>  <NA>
## # ... with 67 more variables: MaritalStatus <fct>, HHIncome <fct>,
## #   HHIncomeMid <int>, Poverty <dbl>, HomeRooms <int>, HomeOwn <fct>,
## #   Work <fct>, Weight <dbl>, Length <dbl>, HeadCirc <dbl>, Height <dbl>,
## #   BMI <dbl>, BMICatUnder20yrs <fct>, BMI_WHO <fct>, Pulse <int>,
## #   BPSysAve <int>, BPDiaAve <int>, BPSys1 <int>, BPDia1 <int>, BPSys2 <int>,
## #   BPDia2 <int>, BPSys3 <int>, BPDia3 <int>, Testosterone <dbl>,
## #   DirectChol <dbl>, TotChol <dbl>, UrineVol1 <int>, UrineFlow1 <dbl>, ...
```

Another useful function from the dplyr package is **glimpse()**. This function displays the number of observations and the number of variables, and then for each variable, displays its name, data type (double, integer, character, factor…) and the first few values.

From the following code, you can see that there are 10,000 individuals and 76 variables. The variable data types include integer (e.g. Age), factor (e.g. Race1) and double (e.g. Poverty). You can also see that are some missing values for a few variables (e.g. Race3). The NHANES dataset is quite tidy but this output could also highlight issues, such as spaces instead of NA for missing values, or variable values stored as column names.

```
NHANES %>%
  glimpse()
```

```
## Rows: 10,000
## Columns: 76
## $ ID               <int> 51624, 51624, 51624, 51625, 51630, 51638, 51646, 5164~
## $ SurveyYr         <fct> 2009_10, 2009_10, 2009_10, 2009_10, 2009_10, 2009_10,~
## $ Gender           <fct> male, male, male, male, female, male, male, female, f~
## $ Age              <int> 34, 34, 34, 4, 49, 9, 8, 45, 45, 45, 66, 58, 54, 10, ~
## $ AgeDecade        <fct>  30-39,  30-39,  30-39,  0-9,  40-49,  0-9,  0-9,  40~
## $ AgeMonths        <int> 409, 409, 409, 49, 596, 115, 101, 541, 541, 541, 795,~
## $ Race1            <fct> White, White, White, Other, White, White, White, Whit~
## $ Race3            <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ Education        <fct> High School, High School, High School, NA, Some Colle~
## $ MaritalStatus    <fct> Married, Married, Married, NA, LivePartner, NA, NA, M~
## $ HHIncome         <fct> 25000-34999, 25000-34999, 25000-34999, 20000-24999, 3~
## $ HHIncomeMid      <int> 30000, 30000, 30000, 22500, 40000, 87500, 60000, 8750~
## $ Poverty          <dbl> 1.36, 1.36, 1.36, 1.07, 1.91, 1.84, 2.33, 5.00, 5.00,~
## $ HomeRooms        <int> 6, 6, 6, 9, 5, 6, 7, 6, 6, 6, 5, 10, 6, 10, 10, 4, 3,~
## $ HomeOwn          <fct> Own, Own, Own, Own, Rent, Rent, Own, Own, Own, Own, O~
## $ Work             <fct> NotWorking, NotWorking, NotWorking, NA, NotWorking, N~
## $ Weight           <dbl> 87.4, 87.4, 87.4, 17.0, 86.7, 29.8, 35.2, 75.7, 75.7,~
## $ Length           <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ HeadCirc         <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ Height           <dbl> 164.7, 164.7, 164.7, 105.4, 168.4, 133.1, 130.6, 166.~
## $ BMI              <dbl> 32.22, 32.22, 32.22, 15.30, 30.57, 16.82, 20.64, 27.2~
## $ BMICatUnder20yrs <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ BMI_WHO          <fct> 30.0_plus, 30.0_plus, 30.0_plus, 12.0_18.5, 30.0_plus~
## $ Pulse            <int> 70, 70, 70, NA, 86, 82, 72, 62, 62, 62, 60, 62, 76, 8~
## $ BPSysAve         <int> 113, 113, 113, NA, 112, 86, 107, 118, 118, 118, 111, ~
## $ BPDiaAve         <int> 85, 85, 85, NA, 75, 47, 37, 64, 64, 64, 63, 74, 85, 6~
## $ BPSys1           <int> 114, 114, 114, NA, 118, 84, 114, 106, 106, 106, 124, ~
## $ BPDia1           <int> 88, 88, 88, NA, 82, 50, 46, 62, 62, 62, 64, 76, 86, 6~
## $ BPSys2           <int> 114, 114, 114, NA, 108, 84, 108, 118, 118, 118, 108, ~
## $ BPDia2           <int> 88, 88, 88, NA, 74, 50, 36, 68, 68, 68, 62, 72, 88, 6~
## $ BPSys3           <int> 112, 112, 112, NA, 116, 88, 106, 118, 118, 118, 114, ~
## $ BPDia3           <int> 82, 82, 82, NA, 76, 44, 38, 60, 60, 60, 64, 76, 82, 7~
## $ Testosterone     <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ DirectChol       <dbl> 1.29, 1.29, 1.29, NA, 1.16, 1.34, 1.55, 2.12, 2.12, 2~
## $ TotChol          <dbl> 3.49, 3.49, 3.49, NA, 6.70, 4.86, 4.09, 5.82, 5.82, 5~
## $ UrineVol1        <int> 352, 352, 352, NA, 77, 123, 238, 106, 106, 106, 113, ~
## $ UrineFlow1       <dbl> NA, NA, NA, NA, 0.094, 1.538, 1.322, 1.116, 1.116, 1.~
## $ UrineVol2        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ UrineFlow2       <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ Diabetes         <fct> No, No, No, No, No, No, No, No, No, No, No, No, No, N~
## $ DiabetesAge      <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ HealthGen        <fct> Good, Good, Good, NA, Good, NA, NA, Vgood, Vgood, Vgo~
## $ DaysPhysHlthBad  <int> 0, 0, 0, NA, 0, NA, NA, 0, 0, 0, 10, 0, 4, NA, NA, 0,~
## $ DaysMentHlthBad  <int> 15, 15, 15, NA, 10, NA, NA, 3, 3, 3, 0, 0, 0, NA, NA,~
## $ LittleInterest   <fct> Most, Most, Most, NA, Several, NA, NA, None, None, No~
## $ Depressed        <fct> Several, Several, Several, NA, Several, NA, NA, None,~
## $ nPregnancies     <int> NA, NA, NA, NA, 2, NA, NA, 1, 1, 1, NA, NA, NA, NA, N~
## $ nBabies          <int> NA, NA, NA, NA, 2, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ Age1stBaby       <int> NA, NA, NA, NA, 27, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ SleepHrsNight    <int> 4, 4, 4, NA, 8, NA, NA, 8, 8, 8, 7, 5, 4, NA, 5, 7, N~
## $ SleepTrouble     <fct> Yes, Yes, Yes, NA, Yes, NA, NA, No, No, No, No, No, Y~
## $ PhysActive       <fct> No, No, No, NA, No, NA, NA, Yes, Yes, Yes, Yes, Yes, ~
## $ PhysActiveDays   <int> NA, NA, NA, NA, NA, NA, NA, 5, 5, 5, 7, 5, 1, NA, 2, ~
## $ TVHrsDay         <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ CompHrsDay       <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ TVHrsDayChild    <int> NA, NA, NA, 4, NA, 5, 1, NA, NA, NA, NA, NA, NA, 4, N~
## $ CompHrsDayChild  <int> NA, NA, NA, 1, NA, 0, 6, NA, NA, NA, NA, NA, NA, 3, N~
## $ Alcohol12PlusYr  <fct> Yes, Yes, Yes, NA, Yes, NA, NA, Yes, Yes, Yes, Yes, Y~
## $ AlcoholDay       <int> NA, NA, NA, NA, 2, NA, NA, 3, 3, 3, 1, 2, 6, NA, NA, ~
## $ AlcoholYear      <int> 0, 0, 0, NA, 20, NA, NA, 52, 52, 52, 100, 104, 364, N~
## $ SmokeNow         <fct> No, No, No, NA, Yes, NA, NA, NA, NA, NA, No, NA, NA, ~
## $ Smoke100         <fct> Yes, Yes, Yes, NA, Yes, NA, NA, No, No, No, Yes, No, ~
## $ Smoke100n        <fct> Smoker, Smoker, Smoker, NA, Smoker, NA, NA, Non-Smoke~
## $ SmokeAge         <int> 18, 18, 18, NA, 38, NA, NA, NA, NA, NA, 13, NA, NA, N~
## $ Marijuana        <fct> Yes, Yes, Yes, NA, Yes, NA, NA, Yes, Yes, Yes, NA, Ye~
## $ AgeFirstMarij    <int> 17, 17, 17, NA, 18, NA, NA, 13, 13, 13, NA, 19, 15, N~
## $ RegularMarij     <fct> No, No, No, NA, No, NA, NA, No, No, No, NA, Yes, Yes,~
## $ AgeRegMarij      <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 20, 15, N~
## $ HardDrugs        <fct> Yes, Yes, Yes, NA, Yes, NA, NA, No, No, No, No, Yes, ~
## $ SexEver          <fct> Yes, Yes, Yes, NA, Yes, NA, NA, Yes, Yes, Yes, Yes, Y~
## $ SexAge           <int> 16, 16, 16, NA, 12, NA, NA, 13, 13, 13, 17, 22, 12, N~
## $ SexNumPartnLife  <int> 8, 8, 8, NA, 10, NA, NA, 20, 20, 20, 15, 7, 100, NA, ~
## $ SexNumPartYear   <int> 1, 1, 1, NA, 1, NA, NA, 0, 0, 0, NA, 1, 1, NA, NA, 1,~
## $ SameSex          <fct> No, No, No, NA, Yes, NA, NA, Yes, Yes, Yes, No, No, N~
## $ SexOrientation   <fct> Heterosexual, Heterosexual, Heterosexual, NA, Heteros~
## $ PregnantNow      <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
```

Summary statistics

To calculate summary statistics on a dataset, we can use several functions. Base R has a **summary()** function, which is available without loading any packages, and Tidyverse has a nice **summarise()** function too.

A **function** is a set of instructions that are saved with a shorthand name. We can pass arguments to functions, such as datasets and options, and the functions can return outputs. We've already seen many functions in our first session (e.g. **ggplot()**, **geom_boxplot()**, **library()**, etc.). They are always written as: functionName(argument,…).

Before we look at the functions, it's important to understand three common types of variables in R, which need to be handled in different ways.

- **Numeric** variables are numbers! We can apply mathematical functions and operations to them.

- **Character** variables are text, like names. They can be sorted alphabetically, but no value is inherently larger than another.
- **Factors** are variables that can only take a finite number of values. They can be ordered (like low, medium, high) or not (like eye colour). They typically have text labels, like “high” or “green”, but there is an underlying ranking, even if this is arbitrarily assigned alphabetically, such as with eye colour.

Base R

Let’s look at the **summary()** function first. This takes either a whole dataset of variables and returns for each variable a list of summary statistics, which will depend on the type of variable.

Let’s use the **select()** method to get summary statistics for Age, Height and Weight from the NHANES dataset.

These variables are numeric, so for each variable a simple list of summary statistics is presented: the minimum, the 25th percentile (1st quartile), the median, the mean, the 75th percentile (3rd quartile), and the maximum. You can see that there is also a number of NA’s for Height and Weight. This is the number of individuals for which the value of those variables are missing. We have encountered missing values in our previous session, and removed them from the plots using the **drop_na()** function.

```
NHANES %>%
  select(Age,Height,Weight) %>%
  summary()
```

```
##       Age            Height          Weight
##  Min.   : 0.00   Min.   : 83.6   Min.   :  2.80
##  1st Qu.:17.00   1st Qu.:156.8   1st Qu.: 56.10
##  Median :36.00   Median :166.0   Median : 72.70
##  Mean   :36.74   Mean   :161.9   Mean   : 70.98
##  3rd Qu.:54.00   3rd Qu.:174.5   3rd Qu.: 88.90
##  Max.   :80.00   Max.   :200.4   Max.   :230.70
##                  NA’s   :353     NA’s   :78
```

Now let’s look at the Gender variable. This is a factor variable, and so the output shows the counts of each value for that variable.

```
NHANES %>%
  select(Gender) %>%
  summary()
```

```
##     Gender
##  female:5020
##  male  :4980
```

There is no character variable in the NHANES dataset, but the **summary()** function doesn’t summarise those. For example, in the built-in dataset of US State Abbreviations, the output only tells us the length of this character variable.

```
state.abb %>%
  summary()
```

```
##    Length     Class      Mode
##        50 character character
```

The Tidyverse

Now let's look at how Tidyverse does summary statistics, specifically the dplyr package. The **summarize()** function is a lot more flexible than the **summary()** function, you can completely customise your output. And the output is also a tibble, the Tidyverse equivalent of a base R data frame, so you can use it for further analysis too.

Let's first get the mean and standard deviation for the Height variable. First we will remove any individual for which the Height value is missing. Then we calculate the mean by calling the base R **mean()** function, and the standard deviation by calling the **sd()** function.

```
NHANES %>%
  drop_na(Height) %>%
  summarize(meanHeight = mean(Height), stdevHeight = sd(Height))
```

```
## # A tibble: 1 x 2
##   meanHeight stdevHeight
##        <dbl>       <dbl>
## 1       162.        20.2
```

If we didn't remove the missing values with **drop_na()** first, we would have to add the na.rm = T argument to each of these functions to make sure the missing values are excluded from the calculations.

```
NHANES %>%
  summarize(meanHeight = mean(Height, na.rm=T), stdevHeight = sd(Height, na.rm=T))
```

```
## # A tibble: 1 x 2
##   meanHeight stdevHeight
##        <dbl>       <dbl>
## 1       162.        20.2
```

We can also use a combination of **group_by()** and **summarize()** to obtain summary statistics of a variable, stratified by another variable.

For example, we can get the mean and standard deviation of the Height variable according to the Gender variable. This time, the tibble has two rows, one for each value of Gender.

```
NHANES %>%
  drop_na(Height) %>%
  group_by(Gender) %>%
  summarize(meanHeight = mean(Height), stdevHeight = sd(Height))
```

```
## # A tibble: 2 x 3
##   Gender meanHeight stdevHeight
##   <fct>       <dbl>       <dbl>
## 1 female       157.        16.8
## 2 male         167.        21.9
```

Similarly, we can also get other summary statistics, still grouped by Gender. Here we will use the **min()**, **max()** and **quantile()** functions to get the minimum, maximum, 25th percentile, median, 75th percentile, range and interquartile range (IQR). The probs argument in the **quantile()** function gives the probability corresponding to the quantile we want. For example, 0.25 will give the 25th percentile.

```
NHANES %>%
  drop_na(Height) %>%
  group_by(Gender) %>%
  summarize(minHeight = min(Height),
            maxHeight = max(Height),
            q25Height = quantile(Height, probs = .25),
            medianHeight = quantile(Height, probs = .5),
            q75Height = quantile(Height, probs = .75),
            rangeHeight = maxHeight - minHeight,
            IQRheight = q75Height - q25Height)
```

```
## # A tibble: 2 x 8
##   Gender minHeight maxHeight q25Height medianHeight q75Height rangeHeight
##   <fct>      <dbl>     <dbl>     <dbl>        <dbl>     <dbl>       <dbl>
## 1 female      83.8      184.      154.         161.      166.        101.
## 2 male        83.6      200.      166.         174.      179.        117.
## # ... with 1 more variable: IQRheight <dbl>
```

Frequencies

To get the number of observations for each value of a categorical variable, we can use the base R **table()** function to create a frequency table. For example, we can see how many individuals there are in each age category.

```
table(NHANES$AgeDecade)
```

```
##
##    0-9   10-19   20-29   30-39   40-49   50-59   60-69     70+
##   1391    1374    1356    1338    1398    1304     919     587
```

We can do the same with the Tidyverse, but the output will be a tibble we can keep using, and we can add a proportion for each age category.

```
NHANES %>%
  drop_na(AgeDecade) %>%
  count(AgeDecade) %>%
  mutate(prop = prop.table(n))
```

```
## # A tibble: 8 x 3
##   AgeDecade     n   prop
##   <fct>     <int>  <dbl>
## 1 " 0-9"     1391 0.144
## 2 " 10-19"   1374 0.142
## 3 " 20-29"   1356 0.140
## 4 " 30-39"   1338 0.138
## 5 " 40-49"   1398 0.145
## 6 " 50-59"   1304 0.135
## 7 " 60-69"    919 0.0951
## 8 " 70+"      587 0.0607
```

If we want to get the number of observations for each combination of two factor variables, we can also use the **table()** function to create a contingency table. Looking at the Gender and AgeDecade variables, we create a table counting the number of male and female individuals for each value of AgeDecade.

```
table(NHANES$Gender,NHANES$AgeDecade)
```

```
##
##          0-9  10-19  20-29  30-39  40-49  50-59  60-69  70+
##   female 653    684    681    677    681    623    480  348
##   male   738    690    675    661    717    681    439  239
```

You will note that here, we use the base R notation **$** to call the variables from the NHANES data frame.

We can do the same with the Tidyverse, but the output will be a tibble we can keep using, and we've added a proportion for each combination.

```
NHANES %>%
  drop_na(Gender,AgeDecade) %>%
  count(Gender,AgeDecade) %>%
  group_by(Gender) %>%
  mutate(prop = prop.table(n))
```

```
## # A tibble: 16 x 4
## # Groups:   Gender [2]
##    Gender AgeDecade     n   prop
##    <fct>  <fct>     <int>  <dbl>
##  1 female " 0-9"      653 0.135
##  2 female " 10-19"    684 0.142
##  3 female " 20-29"    681 0.141
##  4 female " 30-39"    677 0.140
##  5 female " 40-49"    681 0.141
##  6 female " 50-59"    623 0.129
##  7 female " 60-69"    480 0.0994
##  8 female " 70+"      348 0.0721
##  9 male   " 0-9"      738 0.152
## 10 male   " 10-19"    690 0.143
## 11 male   " 20-29"    675 0.139
## 12 male   " 30-39"    661 0.137
## 13 male   " 40-49"    717 0.148
## 14 male   " 50-59"    681 0.141
## 15 male   " 60-69"    439 0.0907
## 16 male   " 70+"      239 0.0494
```

Video lecture 2

Outliers

Outliers are extreme values, which can influence our data analysis. They can either be due to errors, in which case it might be a good idea to remove them, or just extremely rare values observed in the population. It is important to spot these outliers when exploring a new dataset. There are several ways to identify them:

- Extreme minimum or maximum values

- Values that are three standard deviations from the mean
- Values that are more than 1.5 interquartile range below the 25th percentile or more than 1.5 interquartile range above the 75th percentile

You might remember from the graphical exploratory data analysis session, that histograms and boxplots are a good way to visualise outliers. We've already seen the minimum and maximum values, so let's look at the other two methods.

Here we remove the individuals with a missing Weight value, we use the **select()** function to select the ID and Weight variables, and we only keep individuals for which the Weight value is either less than the mean minus three times the standard deviation or greater than the mean plus three times the standard deviation. In the **filter()** function, we've used two conditions, and we separated them by the | "or" operator, so any individual that fulfills either of these conditions will be kept. This gives us a tibble with the 46 potential outliers.

```
NHANES %>%
  drop_na(Weight) %>%
  select(ID,Weight) %>%
  filter(Weight < mean(Weight)-3*sd(Weight) | Weight > mean(Weight)+3*sd(Weight))
```

```
## # A tibble: 46 x 2
##       ID Weight
##    <int>  <dbl>
##  1 52134   169.
##  2 52134   169.
##  3 52134   169.
##  4 52315   231.
##  5 52315   231.
##  6 52365   161.
##  7 53124   161.
##  8 53834   161.
##  9 54491   203
## 10 54732   162.
## # ... with 36 more rows
```

Now we will look at the IQR method. First we create an IQR variable, we remove the individuals with a missing Weight value, then we use the **summarize()** function to calculate the interquartile range by subtracting the 25th percentile from the 75th percentile and finally we use the **pull()** function to extract the value as a vector (as opposed to a tibble). There are several data structures of R objects.

- Atomic vectors just group some values of the same type together.
  - Use the **length()** function to get their size.
  - Create them by using the **c()** combine function.
  - Check whether a variable is a vector with the **is.vector()** function.
- Arrays are made with a vector arranged in a given number of rows and columns
  - They can have 1 (vector), 2 (matrix) or more dimensions
  - Only one data type
- Matrices are two-dimensional arrays
  - They have rows and columns
  - Only one data type
  - Use the **dim()** function to get their size

- Lists are one-dimensional groups of R objects
- Data frames (base R) or tibbles (Tidyverse dplyr) are two-dimensional versions of a list
    - Tables with rows and columns, equivalent to Excel spreadsheets
    - Created as collection of columns, which are vectors

```
IQR <- NHANES %>%
  drop_na(Weight) %>%
  summarize(value=quantile(Weight, probs = 0.75)-quantile(Weight, probs = 0.25)) %>%
  pull(value)
```

Then we take NHANES again, we remove the individuals with a missing Weight value, we use the **select()** function to select the ID and Weight variables, and we only keep individuals for which the Weight value is either more than 1.5 interquartile range below the 25th percentile or more than 1.5 interquartile range above the 75th percentile. This time, we get a tibble with the 172 potential outliers.

```
NHANES %>%
  drop_na(Weight) %>%
  select(ID,Weight) %>%
  filter(Weight < quantile(Weight, probs = 0.25)-1.5*IQR | Weight > quantile(Weight, probs = 0.75)+1.5*I
```

```
## # A tibble: 172 x 2
##       ID Weight
##    <int>  <dbl>
##  1 51962    4.5
##  2 52054  157.
##  3 52134  169.
##  4 52134  169.
##  5 52134  169.
##  6 52297  150
##  7 52315  231.
##  8 52315  231.
##  9 52365  161.
## 10 52442    6
## # ... with 162 more rows
```

Missing values

Another important thing to examine in a new dataset, is how much missing data there is. We've already mentioned missing values, which should be indicated by **NA**. We've seen how to remove missing values by using the **drop_na()** function or the na.rm argument in individual functions.

If there are any missing values indicated by any other value (e.g. na, N/A, spaces or empty quotes), they should be replaced by **NA**. The **glimpse()** function showed us some missing values at the start of our table, but it's more useful to look at the whole dataset.

Using the summarise() function we have seen earlier, we can count the number of NA values by combining the **sum()** and **is.na()** functions. Here we count how many missing values the Weight variable has.

```
NHANES %>%
  summarise(countNA = sum(is.na(Weight)))
```

```
## # A tibble: 1 x 1
##   countNA
##     <int>
## 1      78
```

OPTIONAL – Number of missing values for all variables

This is similar to the code we use to visualise missing data, and you are not expected to understand this. Here we create a tibble with four columns: the variable name (key), the total number of observations (total) and the number of missing observations in that category (num.isna).

```
NHANES %>%
  select(Age, Weight, Height, BMI, PhysActive, DirectChol, BMI_WHO, AgeDecade, Education, Gender) %>%
  gather(key = "key", value = "val") %>%
  mutate(isna = is.na(val)) %>%
  group_by(key) %>%
  mutate(total = n()) %>%
  group_by(key, total, isna) %>%
  summarise(num.isna = n()) %>%
  filter(isna==TRUE) %>%
  select(num.isna)
```

```
## Warning: attributes are not identical across measure variables;
## they will be dropped
```

```
## `summarise()` has grouped output by 'key', 'total'. You can override using the `.groups` argument.
```

```
## Adding missing grouping variables: `key`, `total`
```

```
## # A tibble: 8 x 3
## # Groups:   key, total [8]
##   key        total num.isna
##   <chr>      <int>    <int>
## 1 AgeDecade  10000      333
## 2 BMI        10000      366
## 3 BMI_WHO    10000      397
## 4 DirectChol 10000     1526
## 5 Education  10000     2779
## 6 Height     10000      353
## 7 PhysActive 10000     1674
## 8 Weight     10000       78
```