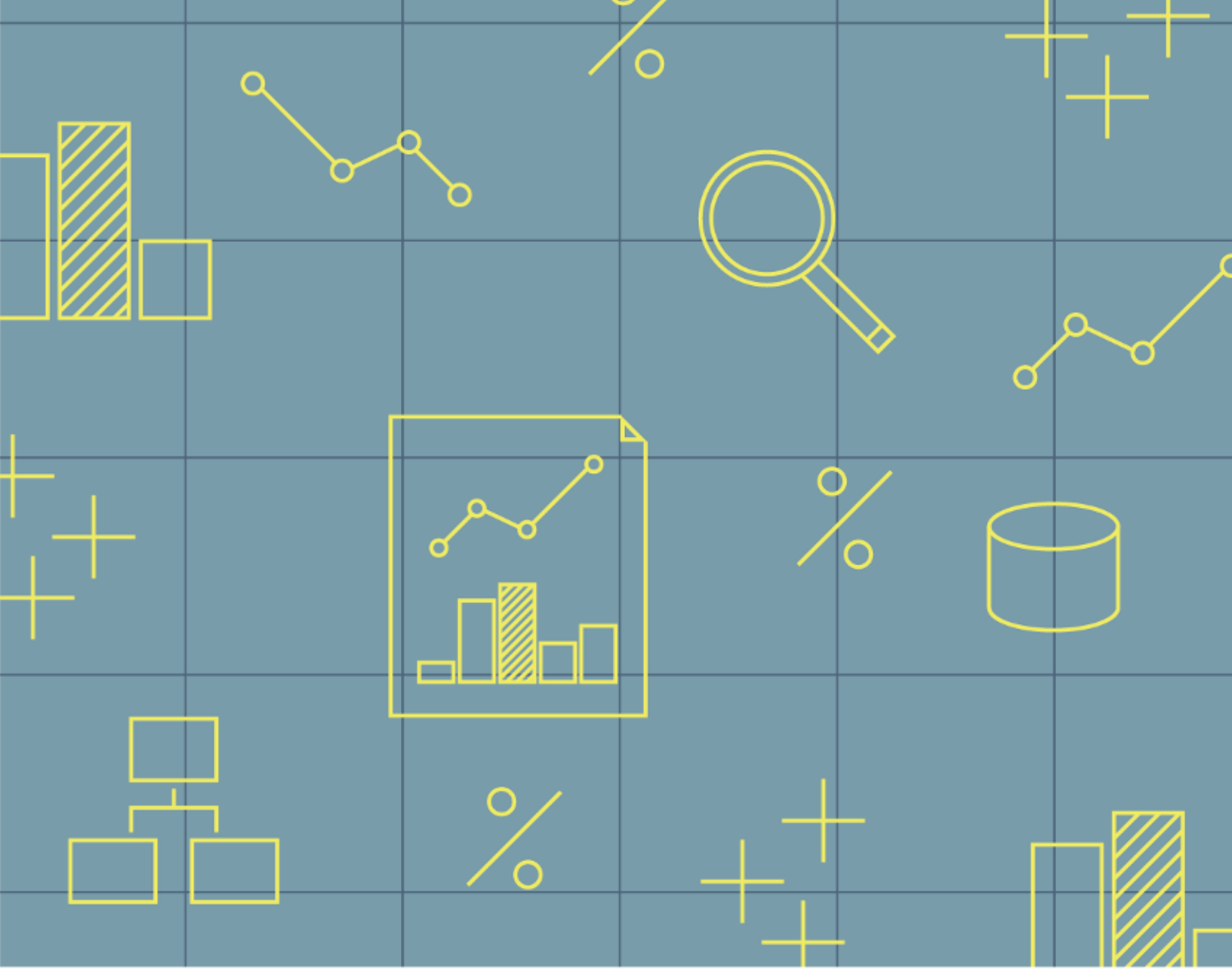


THE DATA PLAYBOOK

By Auth0's Data Team

Auth0

The Data Playbook

Auth0's Data Team

v1.0.0

Contents

Preface

Welcome to the data playbook! This book is aimed at sharing best practices we use when working with data at Auth0. In this book, you'll find various best practices and examples that will help you be as effective as possible when working with data.

This book is brought to you by Auth0's Data Team.

- Antonioli, Rodrigo
- Casas, Pablo
- Connell, Liam
- Crespo, Lucas
- Marasco, Hernan
- Otero, Federico
- Seibelt, Pablo

What we'll learn

You can expect to learn about:

- The essence of data
- Properly collecting data
- Information theory and why it is important
- Statistical functions
- Univariate/Bivariate/multivariate analysis
- Causation vs correlation
- Data visualization

Code examples

Sometimes in the book you'll see something like this:

```
filter(survey, !is.na(Pulse)) %>%
ggplot(aes(x=Pulse)) +
  geom_histogram(binwidth=5, fill="#333399")
```

This is R code that you can execute yourself. In most examples, we'll be using core R as well as the libraries ggplot2 and dplyr. While knowledge of them isn't required to read this book, they can be useful if you want to experiment yourself. If you want to experiment, we recommend that you download R Studio.

Data

For our examples, we'll use data from the responses to a number of questions from 237 Statistics I students at the University of Adelaide. If you want to import this dataset, first install the packages we'll use throughout the book:

```
install.packages(c('MASS', 'tibble', 'dplyr','ggplot2', 'tidyr', 'minerva'))
```

Each time you want to use the dataset in your R session:

```
library(tibble)
library(MASS)
data(survey)
survey = as.tibble(survey)
```

Table 1: Student survey dataset

Sex	Female	Male	Male	Male	Male	Female
Wr.Hnd	18.5	19.5	18.0	18.8	20.0	18.0
NW.Hnd	18.0	20.5	13.3	18.9	20.0	17.7
W.Hnd	Right	Left	Right	Right	Right	Right
Fold	R on L	R on L	L on R	R on L	Neither	L on R
Pulse	92	104	87	NA	35	64
Clap	Left	Left	Neither	Neither	Right	Right
Exer	Some	None	None	None	Some	Some
Smoke	Never	Regul	Occas	Never	Never	Never
Height	173.00	177.80	NA	160.00	165.00	172.72
M.I	Metric	Imperial	NA	Metric	Metric	Imperial
Age	18.250	17.583	16.917	20.333	23.667	21.000

The dataset components are:

Sex

The sex of the student ("Male" or "Female")

Wr.Hnd

The span (distance from tip of thumb to tip of little finger of spread hand) of the writing hand in centimeters.

NW.Hnd

The span of the non-writing hand

W.Hnd

The writing hand of student ("Left" or "Right")

Fold

"Fold your arms! Which is on top?" ("R on L," "L on R," "Neither")

Pulse

The pulse rate of the student (beats per minute)

Clap

'Clap your hands! Which hand is on top?' ("Right," "Left," "Neither")

Exer

How often the student exercises ("Freq" [frequently], "Some," "None")

Smoke

How much the student smokes ("Heavy,", "Regul" [regularly], "Occas" [occasionally], "Never") **Height**

The height of the student in centimeters

M.I

Whether the student expressed height in Imperial (feet/inches) or metric (centimeters/meters) units. ("Metric" or "Imperial")

Age

The age of the student in years

Chapter 1

The Essence of Data

1.1 What exactly is data?

The first thing is to define what we call **data**. According to the current Wikipedia definition, it is:

> Data [...] is a set of values of qualitative or quantitative variables. An example of qualitative data would be an anthropologist's handwritten notes about her interviews with people of an Indigenous tribe. Pieces of data are individual pieces of information. While the concept of data is commonly associated with scientific research, data is collected by a huge range of organizations and institutions, including businesses (e.g., sales data, revenue, profits, stock price), governments (e.g., crime rates, unemployment rates, literacy rates) and non-governmental organizations (e.g., censuses of the number of homeless people by non-profit organizations).
>
> Data is measured, collected and reported, and analyzed, whereupon it can be visualized using graphs, images or other analysis tools. Data as a general concept refers to the fact that some existing information or knowledge is represented or coded in some form suitable for better usage or processing. Raw data ("unprocessed data") is a collection of numbers or characters before it has been "cleaned" and corrected by researchers. Raw data needs to be corrected to remove outliers or obvious instrument or data entry errors (e.g., a thermometer reading from an outdoor Arctic location recording a tropical temperature). [...]

(Wikipedia, 2016b)

To put it simply, data is anything you can somehow *measure*, whether it is

`numerical`, `categorical`, `text`, `images`, or `sound`. All of this is data. In fact, the words written in this book can be considered data. What's more important is what we want to do with the data.

1.2 Where can we find data?

Everywhere! In your day-to-day life, you use data all the time. For example:

- How much money you have right now in the bank
- How much you spent last month
- How far your home is from work
- Your gender
- How old you are
- How fast your heart beats
- How many dogs/cats you have
- How many hairs you have
- Your favorite places to eat

Some of these datapoints can be static or slow-changing, while others can change by the minute or even by the second. You could capture most of these variables and "analyze yourself" or even theoretically capture it for family or friends. Now that wearables are all the rage, the act of capturing some of these variables can even be automated.

It's also interesting to realize how much of this information we give away to various providers (Google, Facebook, fitness apps, finance apps, etc.).

Chapter 2

Data Collection

The first step in doing anything with data is obviously obtaining that data. Sometimes, someone has already collected data for you, while other times we need to do it ourselves. In addition, we sometimes have to accept that for various reasons, we won't get the data we want. Those reasons can be technical, ethical, or otherwise.

This is a very important step to do well because it is the input for everything that follows. If this step has been done poorly, the data we collect will be useless or, *even worse*, **misleading**.

There are some guidelines we can follow for a better data collection **experience**. Here, we'll specifically talk about Auth0's case, a `Software as a Service` product.

2.1 Backend databases

Downloading production databases can be painful, but it can also be critical to success. Querying directly in production is usually not an option. If you do that, you risk disrupting the product itself, and the technology that was considered suitable for creating a service will most probably not be the ideal one for analytics.

There are many things to consider:

- The schema of the database
- The entities it contains
- How often the entities are updated (to know how often we need to come back to capture them)
- If you need to keep snapshots of the data

If we know all this, designing a proper data collection scheme should not be very difficult.

2.2 Backend tracking

Backend tracking is *almost* always better than frontend tracking. You'll always lose a percentage of events when tracking anything on the frontend. You should always try to track in the backend if possible because you own the infrastructure on the backend, but you don't own your user's computer.

Some issues with frontend tracking are:

- If you are using standard libraries like Google Analytics, Segment, Mixpanel, etc., anyone using tracking blockers like Ghostery and some adblockers will actively block all events sent this way.
- If the user has blocked cookies, local storage, etc. or if the user clears it regularly, you'll lose the association between different events.

So why bother with frontend tracking at all? Let's go on to the next section.

2.3 Frontend tracking

We need frontend tracking to know what pages users visit, the sliders they play with, whether they scroll down or don't even look, etc. Things that only happen in the browser have to be tracked there; there's no way around it.

Let's go on to the **Auth0** case.

Each browser gets a cookie that has what we call an *anonymous ID*. This ID lets us link all the activity from each user together to see their entire sequence of pageviews and events. If the user logs into the platform, we execute an **identify** event that links the anonymous ID with the internal user ID. This is our link between frontend data and backend data.

We use our own metrics library called auth0-metrics, with an API very similar to segment's. It sends to Segment and our own endpoint.

In Auth0, we track:

- Pageviews
- Tracked events
- Links between anonymous and user IDs

2.3.1 Pageviews

For every page visit, we track the following attributes:

- The date and time of the visit in UTC
- The URL visited
- The previous URL from which the user came (or empty if it was typed in the navigation bar)
- The UTM tags (Buffer, 2015)
- The IP address (can be used to infer the geolocation)
- The user agent (can be parsed for browser/OS)

2.3.2 Tracked events

We allow tracking of arbitrary events at any point, such as when the user clicks on a button, scrolls past a certain point, changes the value of a slider, etc.

We name the events like this:

`action:section:subsection:subsubsection:...`

For example, if a signup button is pressed on the website in the product section, the name could be:

`click:website:product:signup`

All the same attributes as a pageview are sent, but you can also send up to two optional strings as additional properties:

`trackData` and `value`

Why only two strings? What if we want to send a lot more properties? You should probably check why you need to send so much data in one event. Will you be able to analyze it later on? Will you be able to properly query it afterwards and extract meaningful insights from it? We've found that two values are more than enough for most cases, and it simplifies the required database maintenance.

Which brings us to…

2.4 Atomicity

It's important that no matter you measure, you try to be as atomic as possible. For example, if you are running a barbershop, don't just save how many clients you had, how many were men, and how many were women. Save each client and each service you do for them as different data points. You can't analyze summarized data the same way you can analyze detailed data.

2.5 Dates

For any deity's sake, or even for Richard Dawkins' sake, always save every datetime on UTC unless you want to suffer for no reason. Don't save data with any other timezone. It doesn't even matter if you think you'll never want something other than your current timezone. When you're interfacing with any other systems, they'll all probably be in UTC. So save yourself lots of trouble and store everything in UTC. You can always convert later when creating reports, but store your raw data in UTC.

2.6 Entities

When collecting data, understand the entity for which you are saving information. Is it an anonymous user, a user from your service, or a company? Understanding this is crucial because a badly designed data structure will come back to haunt you. Ensure that you have keys with which you can join all the datasets you collect.

Traversing the path from one entity through four or more entities to get to the final entity you want may sound ridiculous when you start collecting the data, and it may be business critical later on.

Chapter 3

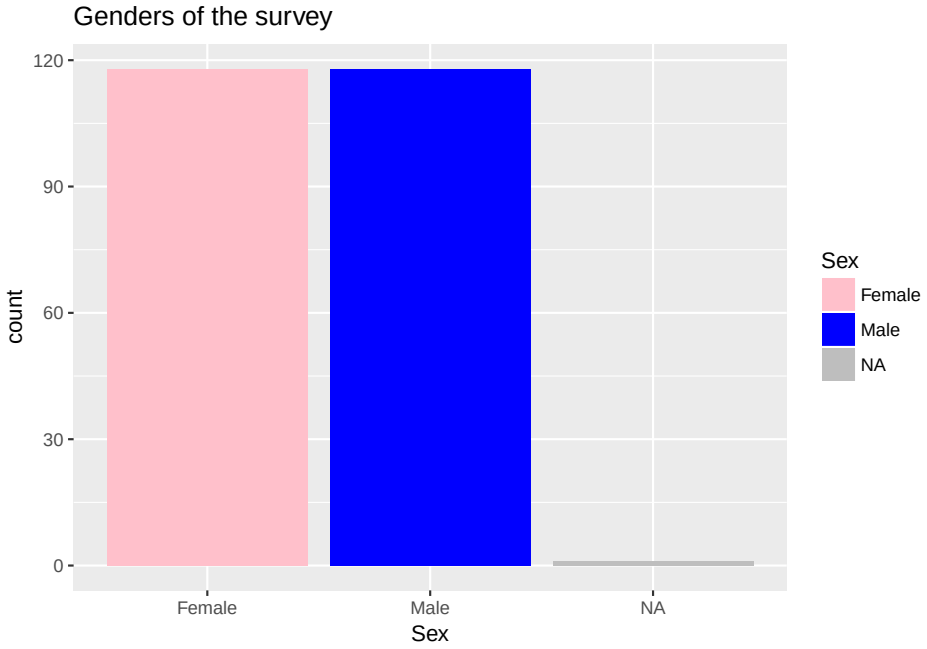
Information Theory

Information theory is a relatively new discipline that Claude E. Shannon founded in 1948. It studies the quantification, storage, and communication of information, mainly through measuring **entropy**. It is a fundamental theory with applications in communication, data compression, encryption, quantum physics, neurobiology, and more.

3.1 Let's start playing

Let's start playing around with the dataset mentioned in the preface, for example the variable `Sex` from the survey.

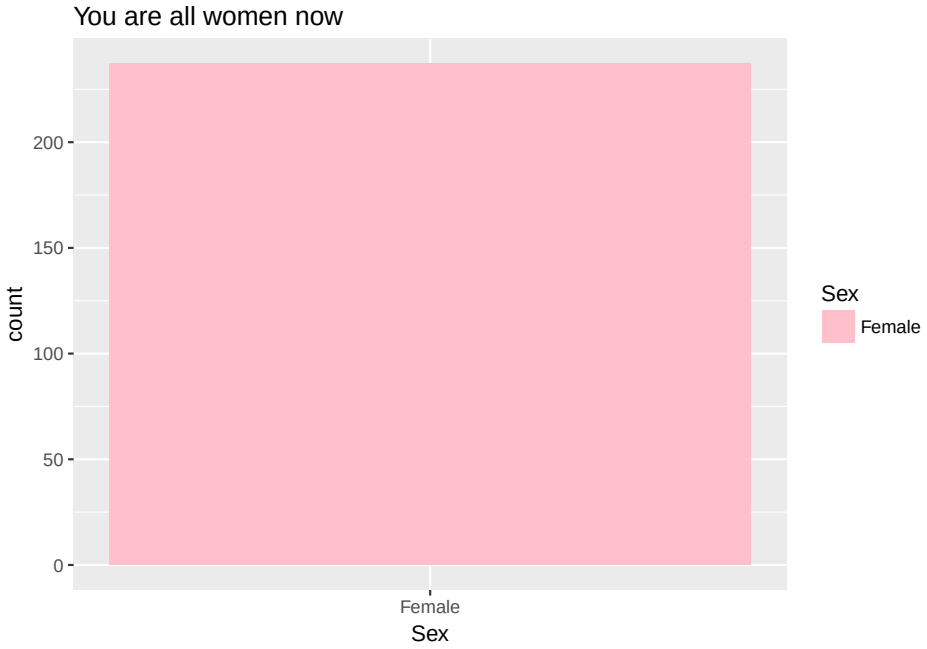
```
ggplot(survey, aes(Sex)) +
  geom_bar(aes(fill=Sex)) +
  scale_fill_manual(values = c("pink", "blue"), na.value="grey") +
  ggtitle("Genders of the survey")
```



We can see that we have a pretty much balanced variable; half of the respondents were men, and the other half were women. NA cases represent those where the answer was left empty; this is an R language annotation, and it stands for "Not available." In many databases, this is annotated as a `null` or `missing` value.

What can information theory tell us about this variable? Well, first let's think intuitively. What if instead of having both genders in our survey, all respondents were either *Female* or *Male*? How useful would a variable like that be?

```
allWomen = survey
allWomen$Sex = "Female"
ggplot(allWomen, aes(Sex)) +
  geom_bar(aes(fill=Sex)) +
  scale_fill_manual(values = c("pink"), na.value="grey") +
  ggtitle("You are all women now")
```



What does this variable tell you about row 1?

```
allWomen$Sex[1]
```

```
## [1] "Female"
```

What about row 17?

```
allWomen$Sex[17]
```

```
## [1] "Female"
```

3.2 Shannon's entropy

Ok, this is pointless, right? No matter what row I choose, the result will be the same because all rows have the same value. So what information do I gain from looking at this variable? According to Shannon's entropy, in such a variable we gain 0 bits of information. Calculating this in R is trivial.

```
freqs = table(allWomen$Sex) / length(allWomen$Sex)
paste("Shannon's entropy is",
      -sum(freqs * log2(freqs)), "bits")
```

```
## [1] "Shannon's entropy is 0 bits"
```

Going back to the original example, where each student can be female or male, we should gain information by looking at this variable because we have more

than one possible value. Let's calculate how much:

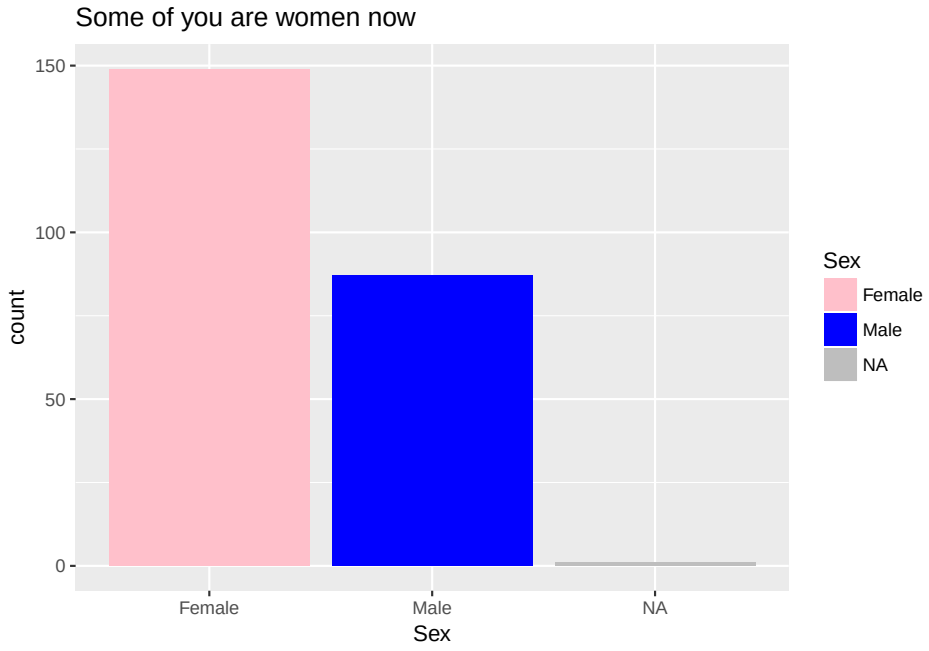
```
freqs = table(survey$Sex) / length(survey$Sex)
paste("Shannon's entropy is",
      round(-sum(freqs * log2(freqs)), 5), "bits")
```

```
## [1] "Shannon's entropy is 1.00186 bits"
```

Ok, this is interesting. It's telling us that knowing the sex of a student gains us a little bit more than 1 bit of information thanks to the missing values. This is intuitively sound because 1 bit is a binary variable: 1 or 0. Although Shannon's entropy is related to that, it's not exactly the same concept as a 0-1 computer bit. It refers to the amount of information gained by knowing this value.

Let's shake the dataset a little bit more to understand how the information gain works under different situations. Let's now bias the Sex variable so a random 25% of all rows are set to female whether they were male or female.

```
set.seed(14563) # For reproducibility we set the random seed
biased = survey
biased$Sex = ifelse(runif(nrow(biased)) > 0.75,
                    "Female", as.character(biased$Sex))
ggplot(biased, aes(Sex)) +
  geom_bar(aes(fill=Sex)) +
  scale_fill_manual(values = c("pink", "blue"),
                    na.value="grey") +
  ggtitle("Some of you are women now")
```



What's the new information gain?

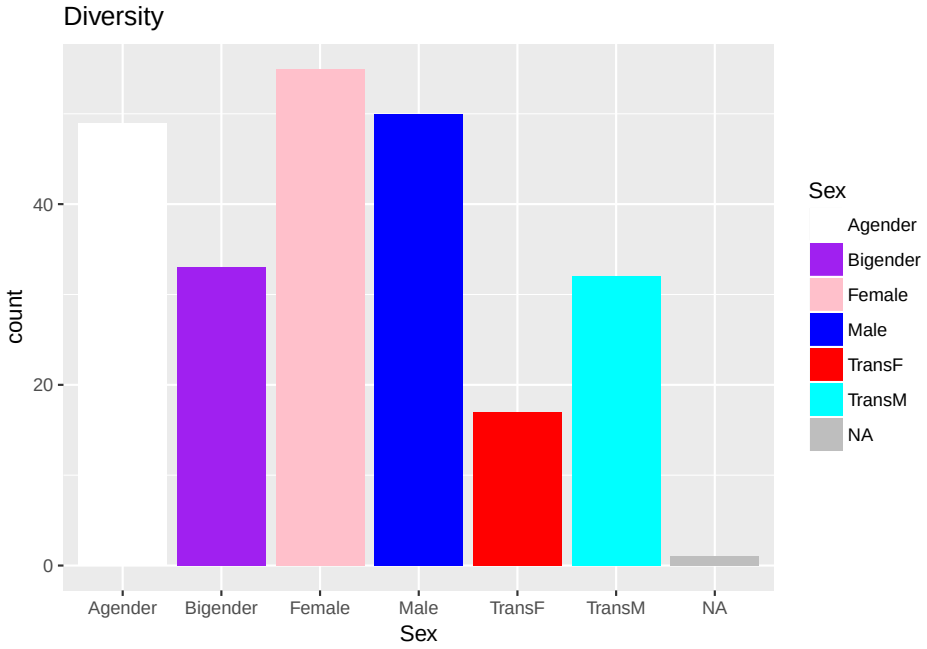
```
freqs = table(biased$Sex) / length(biased$Sex)
paste("Shannon's entropy is",
      round(-sum(freqs * log2(freqs)), 5), "bits")
```

```
## [1] "Shannon's entropy is 0.95169 bits"
```

The entropy was reduced to 0.95 bits, but why? Well, now that the percentage is biased towards women, knowing if a certain person is a man or a woman gives you less information because the dataset is now unbalanced. The extreme unbalancing example occurred when we transformed everyone into a woman. You could just assume that any row was a woman without looking at the individual row.

Let's make it a little more interesting by randomly introducing some non-binary genders.

```
set.seed(14563) # For reproducibility we set the random seed
nonbinary = survey
nonbinary$Sex = ifelse(runif(nrow(nonbinary)) > 0.80,
                       "TransF", as.character(nonbinary$Sex))
nonbinary$Sex = ifelse(runif(nrow(nonbinary)) > 0.80,
                       "TransM", as.character(nonbinary$Sex))
nonbinary$Sex = ifelse(runif(nrow(nonbinary)) > 0.80,
                       "Bigender", as.character(nonbinary$Sex))
nonbinary$Sex = ifelse(runif(nrow(nonbinary)) > 0.80,
                       "Agender", as.character(nonbinary$Sex))
ggplot(nonbinary, aes(Sex)) +
  geom_bar(aes(fill=Sex)) +
  scale_fill_manual(values = c("white", "purple", "pink", "blue",
                               "red", "cyan"), na.value="grey") +
  ggtitle("Diversity")
```




```
freqs = table(nonbinary$Sex) / length(nonbinary$Sex)
paste("Shannon's entropy is",
      round(-sum(freqs * log2(freqs)), 5), "bits")
```

```
## [1] "Shannon's entropy is 2.49157 bits"
```

Now that we have seven possible values, the information gain has increased a lot; we have almost 2.5 bits of information. What does this mean intuitively and in this context? We gain a lot more knowledge (information) about the person by knowing this variable, which makes a lot of sense, right?

Understanding how information theory works makes it possible to analyze huge datasets and select interesting variables based on the amount of information they possess. They might or might not be useful afterwards, but if they have little information, it is quite probable that they will not be meaningful to almost any analysis.

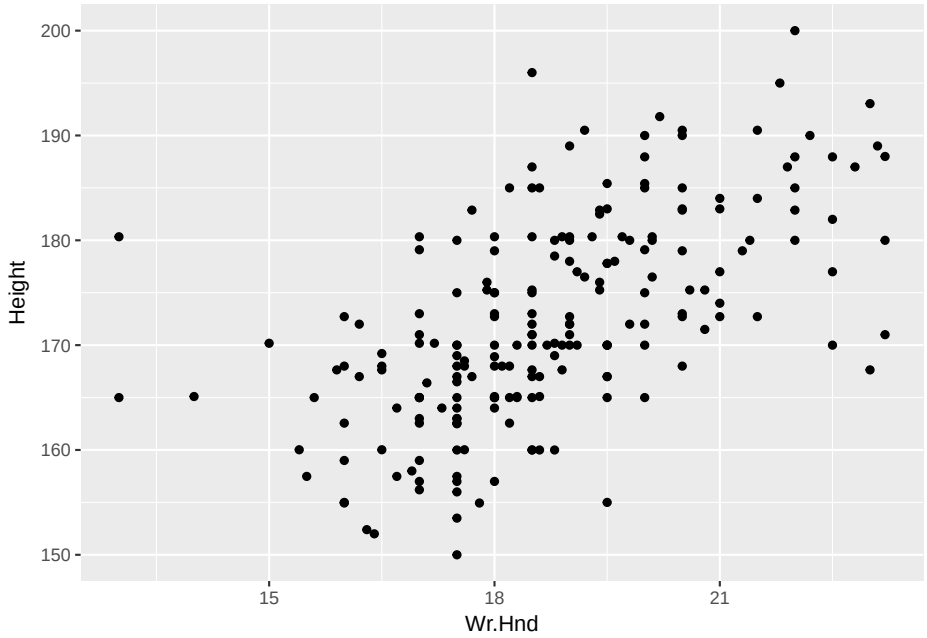
Let's try expanding this concept to look at the relationship between two variables.

3.3 Maximal information coefficient

We're going to try to understand whether there's a relationship between the students' height in cm (`Height`) and the distance from the tips of the students'

thumbs to the tips of their little fingers when they spread out their writing hand (`Wr.Hnd`).

```
ggplot(survey, aes(x=Wr.Hnd, y=Height)) +
  geom_point()
```



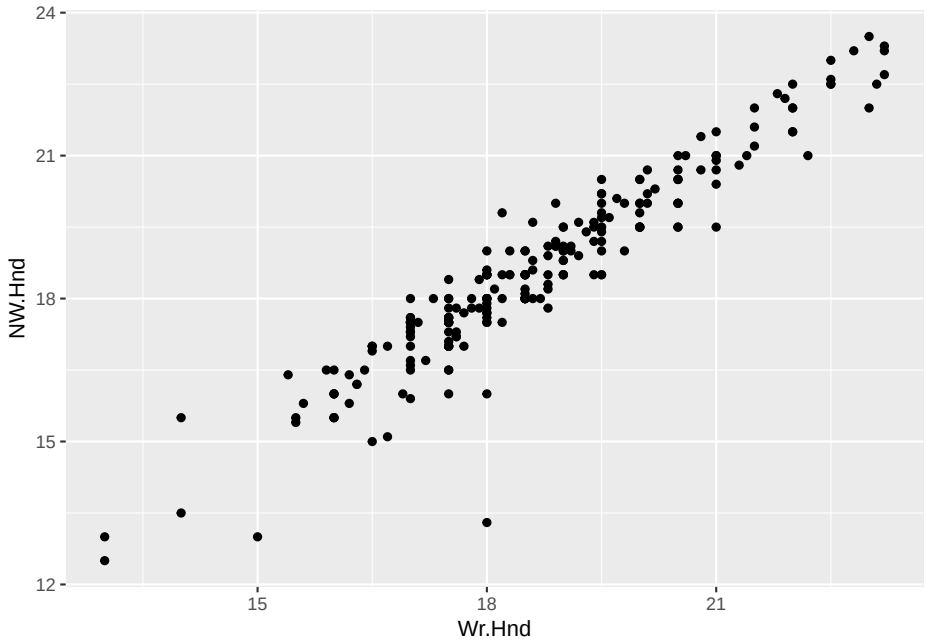
Looking at this chart intuitively, we can quickly see that there seems to be a relationship. When the span is higher, usually the height is too. There are various ways to analyze the relationship between two variables, but here we'll focus on the technique named "Maximal information coefficient", which captures not only lineal but also nonlineal relationships.

```
library(minerva)
res = mine(survey$Wr.Hnd, survey$Height,
           use = 'pairwise.complete.obs')
paste("The MIC is", round(res$MIC,3))
```

```
## [1] "The MIC is 0.398"
```

If we use a relationship that is more obviously related, we'll see that the MIC is higher, e.g., if we compare the span of the two hands:

```
ggplot(survey, aes(x=Wr.Hnd, y=NW.Hnd)) +
  geom_point()
```

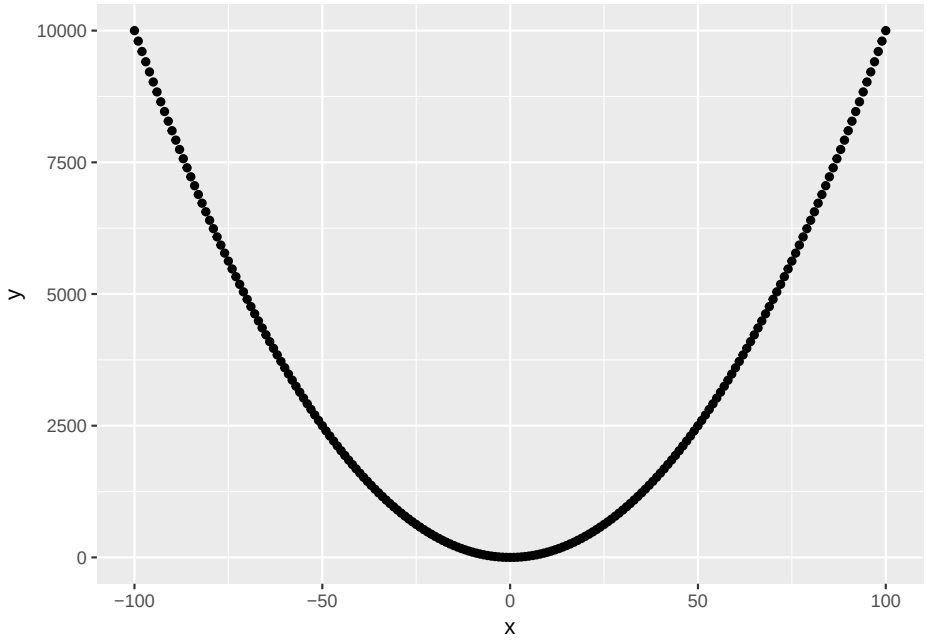



```
res = mine(survey$NW.Hnd, survey$Height,
           use = 'pairwise.complete.obs')
paste("The MIC is", round(res$MIC,3))
```

```
## [1] "The MIC is 0.4"
```

If we artifically produced a relationship of numbers that form a function, we get an interesting MIC:

```
x = seq(-100, 100)
parabole = data.frame(x=x, y=x^2)
ggplot(parabole, aes(x=x, y=y)) +
  geom_point()
```

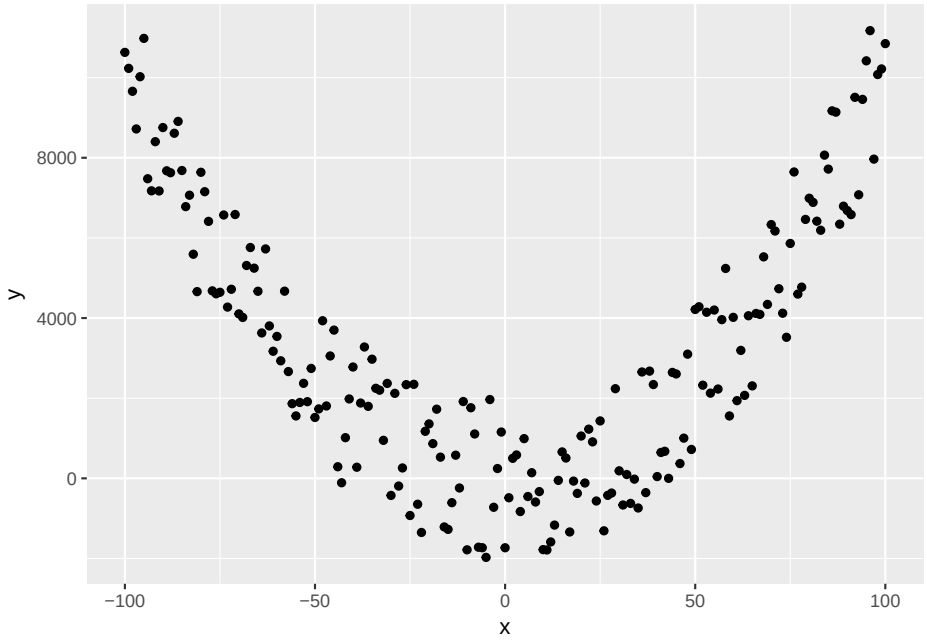



```
res = mine(parabole$x, parabole$y)
paste("The MIC is", round(res$MIC,3))
```

```
## [1] "The MIC is 1"
```

The MIC is 1, which is the maximum possible amount, so this metric has identified that these two variables are **very** related. It's right because we did $y = x^2$. But what if we purposefully introduce some noise?

```
x = seq(-100, 100)
set.seed(31332) # Reproducible always
noisyParabole = data.frame(x=x, y=x^2)
noisyParabole$y = jitter(noisyParabole$y, 1000)
ggplot(noisyParabole, aes(x=x, y=y)) +
  geom_point()
```

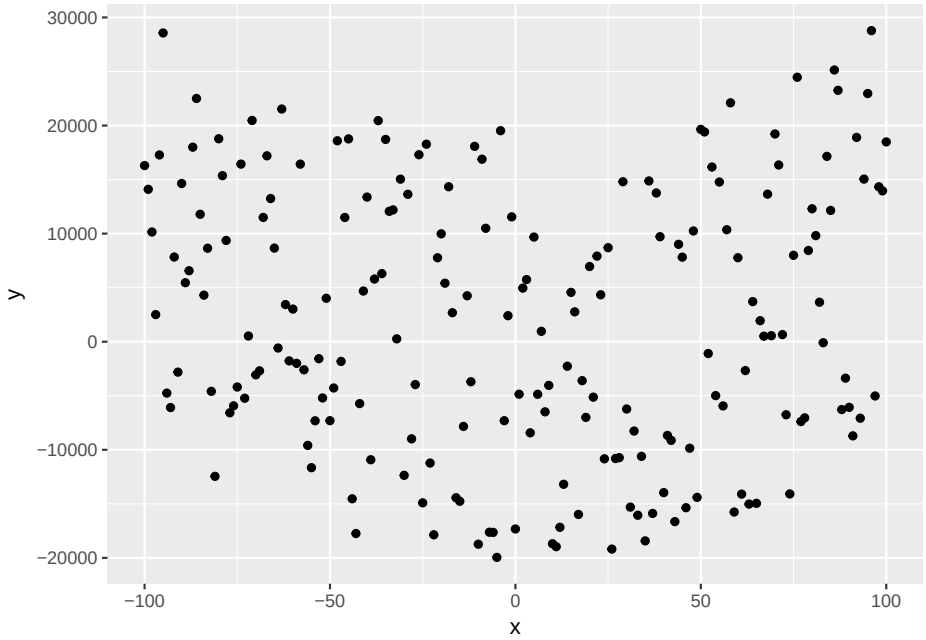



```
res = mine(noisyParabole$x, noisyParabole$y)
paste("The MIC is", round(res$MIC,3))
```

```
## [1] "The MIC is 0.824"
```

Even with some noise and a non-lineal shape, we still see a relationship! Way to go MIC! What if we break the dataset to the point of almost pure noise?

```
x = seq(-100, 100)
set.seed(31332) # Reproducible always
veryNoisyParabole = data.frame(x=x, y=x^2)
veryNoisyParabole$y = jitter(veryNoisyParabole$y, 10000)
ggplot(veryNoisyParabole, aes(x=x, y=y)) +
  geom_point()
```




```
res = mine(veryNoisyParabole$x, veryNoisyParabole$y)
paste("The MIC is", round(res$MIC,3))
```

```
## [1] "The MIC is 0.25"
```

Now the MIC has been lowered to just 0.25. With more noise we have a lower MIC, and with more information we have a higher one. Explaining the technical details of MIC is beyond the scope of this book, but you can read more in the paper "Detecting Novel Associations in Large Data Sets" (Reshef et al., 2011).

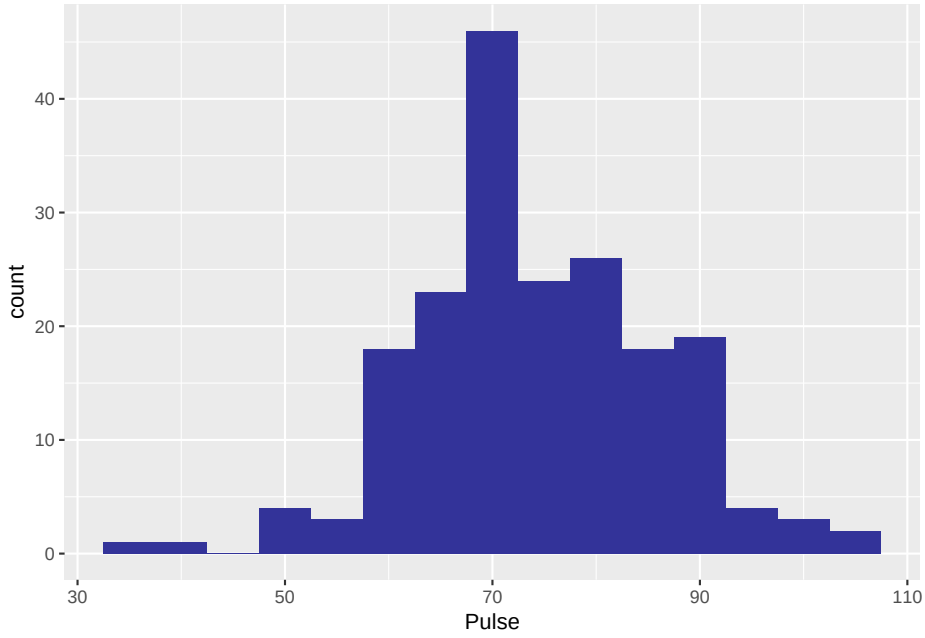
Chapter 4

Statistical functions

4.1 Analyzing the distribution

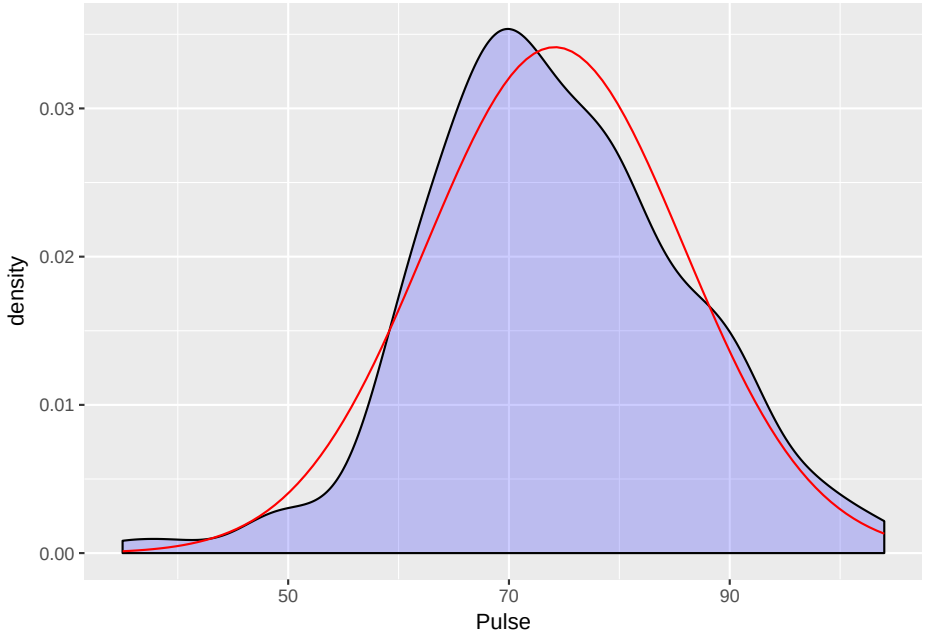
Let's continue playing around with the dataset mentioned in the preface. As an example, we'll take the variable `Pulse` from the survey.

```
filter(survey, !is.na(Pulse)) %>%
ggplot(aes(x=Pulse)) +
  geom_histogram(binwidth=5, fill="#333399")
```



The distribution of this variable looks pretty much like a normal distribution. We can also look at the density instead of looking at the histogram. We'll also plot in **red** a normal distribution to see how similar it is.

```
filter(survey, !is.na(Pulse)) %>%
ggplot(aes(x=Pulse)) +
  geom_density(fill="blue", alpha=0.2) +
  stat_function(fun = dnorm,
                color="red",
                args=list(
                  mean= mean(survey$Pulse, na.rm=T),
                  sd=sd(survey$Pulse, na.rm=T))
                )
```



We can see that this variable behaves almost like the normal distribution, with a mean just above 70. Well, we can study this variable like this because it has many values that are continuous. We know that a pulse of 90 is more than a pulse of 89, but that may not be true for some numeric variables, such as area codes, phone numbers, or any other kind of identifier that is non-sequential).

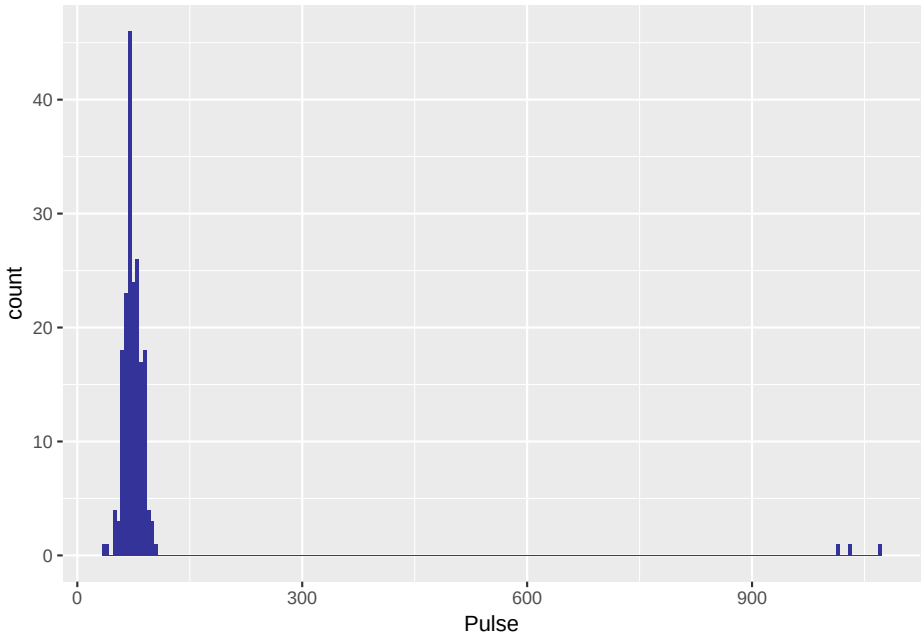
In addition, for most analysis we don't need continuous variables to be technically continuous in the mathematical sense, as in having infinite possible values between two numbers. We can analyze integers the same way we analyze decimal values for almost all, if not all, use cases.

4.2 What do we mean with the mean?

We'll change three pulses to be above 1000, even if it doesn't make sense for this particular example (although it could happen naturally through a measurement error).

```
statsSurvey = survey
statsSurvey[1,]$Pulse = 1017
statsSurvey[2,]$Pulse = 1032
statsSurvey[3,]$Pulse = 1071

filter(statsSurvey, !is.na(Pulse)) %>%
ggplot(aes(x=Pulse)) +
  geom_histogram(binwidth=5, fill="#333399")
```



So now if your boss asks you for the average pulse of the survey, what do you answer? Let's assume we just calculate the mean directly.

```
mean(statsSurvey$Pulse, na.rm=T)
```

```
## [1] 88.92708
```

Looking at the charts without the outliers, does this sound reasonable? It appears that the distribution is centered around 70, but the mean says almost 89. It sounds like calculating the mean directly is misleading in this case. Don't worry. There are ways to avoid this problem. Let's get some quartiles:

```
summary(statsSurvey$Pulse)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   35.00   66.00   72.50   88.93   80.00 1071.00      45
```

Ok, that sounds more useful. The first quartile is `66`, the median is `72.5`, and the third quartile is `80`. If we look at the original dataset:

```
summary(survey$Pulse)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   35.00   66.00   72.50   74.15   80.00  104.00      45
```

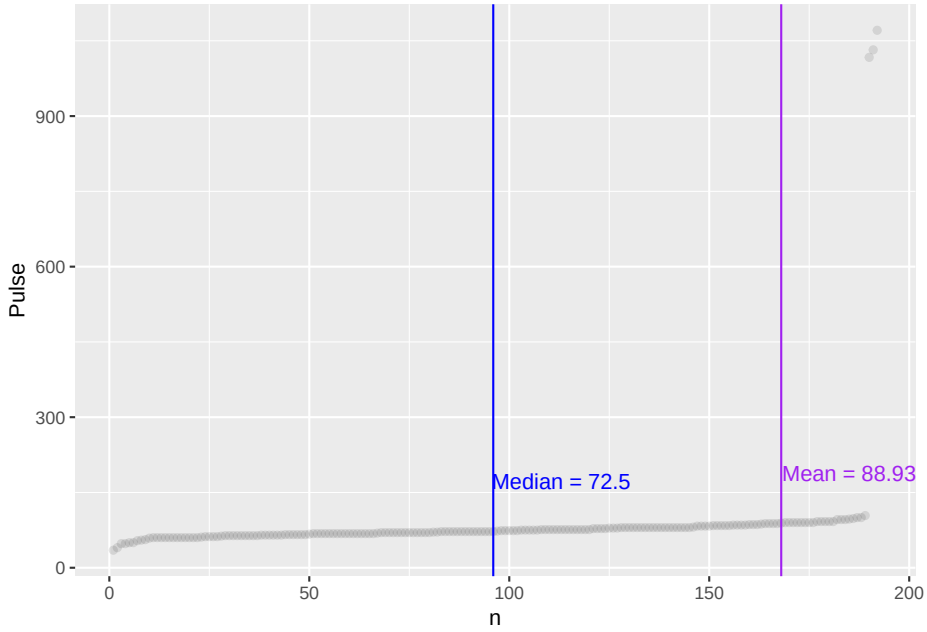
The mean is very near the median because there were no outliers. Notice that the three outliers we introduced didn't even move the median or the other quartiles. How is this possible?

The mean is just $mean(x) = sum(x)/length(x)$, but the median is calculated as the percentile 50 of the variable. This just involves ordering all the variables and getting the value at `length(x) / 2`. If the length is a multiple of 2, we'll just get the mean between the two values at the middle.

```
medianExample = filter(statsSurvey, !is.na(Pulse)) %>%
arrange(Pulse)
medianExample$n=seq(1,nrow(medianExample))

nearestToMean = which.min(
  abs(medianExample$Pulse-mean(medianExample$Pulse)))

ggplot(medianExample, aes(x=n, y=Pulse)) +
  geom_point(fill="blue", alpha=0.1) +
  annotate(geom="text",
           x=nrow(medianExample)/2 + 17,
           y = median(medianExample$Pulse)+100,
           label=paste("Median =",
                       median(medianExample$Pulse)),
           colour="blue"
           ) +
  annotate(geom="text",
           x=nearestToMean+17,
           y = mean(medianExample$Pulse)+100,
           label=paste("Mean =",
                       round(mean(medianExample$Pulse), 2)),
           colour="purple"
           ) +
  geom_vline(xintercept = nrow(medianExample)/2, colour="blue") +
  geom_vline(xintercept = nearestToMean, colour="purple")
```



Doing this, we can see that using the median is a much more stable metric that is much less sensitive to outliers, but that doesn't mean we should never look at the mean or something like that. It just means we have to understand the potential issues.

Another way to ensure that the outliers won't break our averages is by stripping out extreme values before we get the mean. How do we define extreme values? We can, for example, delete every value on the top 2% and the bottom 2%.

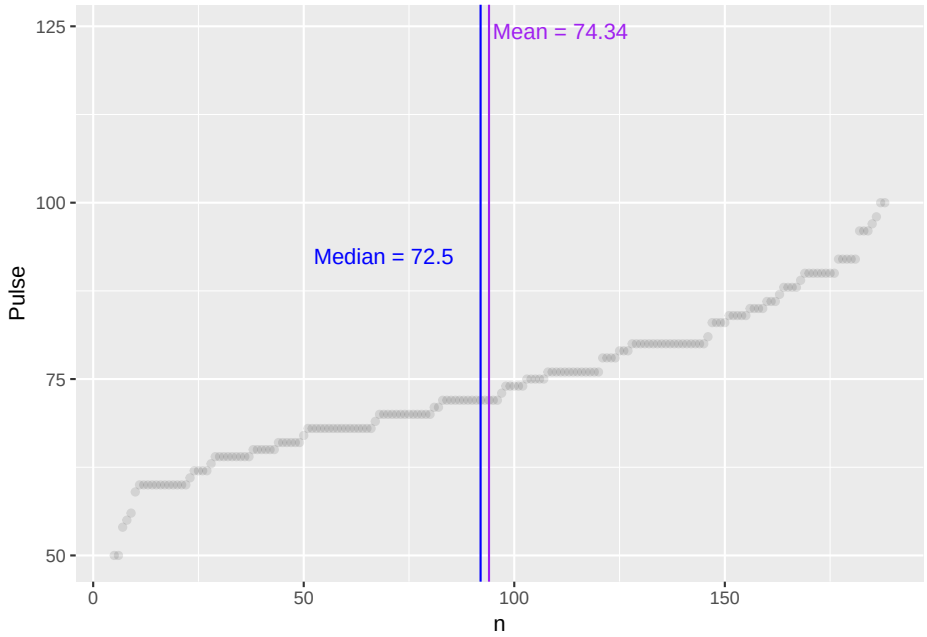
```
outliers = quantile(medianExample$Pulse,
                    probs=c(0.02, 0.98)) # percentiles 2 and 98

withoutOutliers = filter(medianExample,
                         Pulse > outliers[1], Pulse < outliers[2])

nearestToMean = which.min(
  abs(withoutOutliers$Pulse-mean(withoutOutliers$Pulse)))

ggplot(withoutOutliers, aes(x=n, y=Pulse)) +
  geom_point(fill="blue", alpha=0.1) +
  annotate(geom="text",
           x=nrow(withoutOutliers)/2 -23,
           y = median(withoutOutliers$Pulse)+20,
           label=paste("Median =",
                       median(withoutOutliers$Pulse)),
           colour="blue"
```

```
          ) +
annotate(geom="text",
          x=nearestToMean+17,
          y = mean(withoutOutliers$Pulse)+50,
          label=paste("Mean =",
                      round(mean(withoutOutliers$Pulse), 2)),
          colour="purple"
          ) +
geom_vline(xintercept = nrow(withoutOutliers)/2, colour="blue") +
geom_vline(xintercept = nearestToMean, colour="purple")
```



That sounds a lot more reasonable, right? In addition, the mean here can be a useful indicator in combination with the median. Here we can see that the mean is higher, and looking at the chart, you can easily see that the increase to the right is stronger than on the left side of the median, which indicates a negative skewness (the distribution is a little biased towards higher values).

Chapter 5

Univariate, bivariate, and multivariate analysis

In the last chapter, we learned how to study the distribution of a variable and how to relate one variable to another to try to find patterns in the data. The first study is univariate because you are studying just one variable at a time. The second one is a bivariate analysis because you are trying to find possible relationships between two variables. Before diving into a more advanced multivariate analysis, let's explore how the other ones work.

5.1 Another dataset

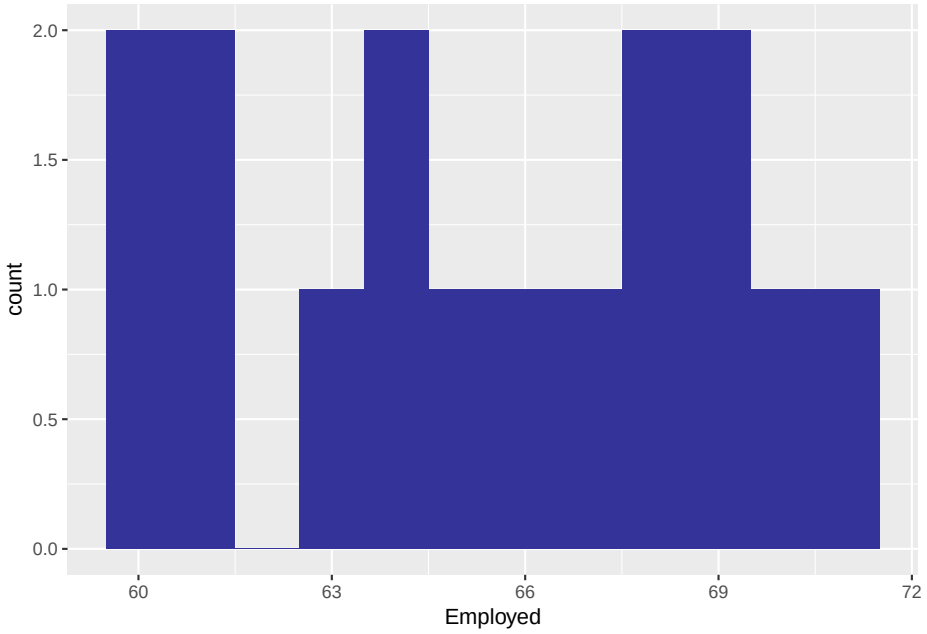
For this example, we'll use the dataset *Longley's economic regression data*. It's a macroeconomic dataset with seven economical variables observed from 1947 to 1962. We want to study the variable "Employed," which indicates the number of people employed in a year.

```
data(longley) #Include dataset
```

5.2 Univariate analysis

So, the first analysis would be how the variable is distributed.

```
ggplot(longley, aes(x=Employed)) +
  geom_histogram(binwidth=1, fill="#333399")
```




```
summary(longley$Employed)
```

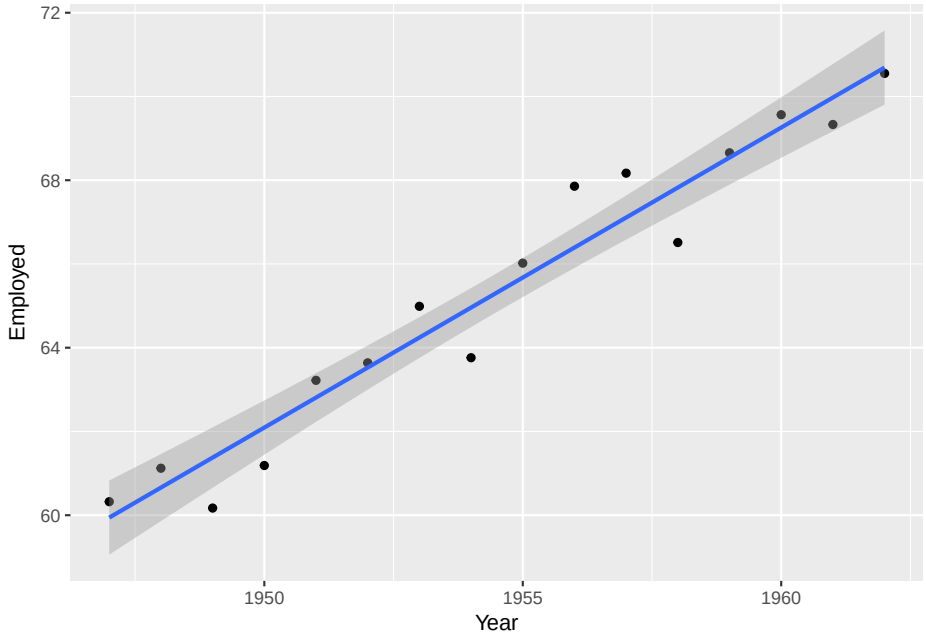
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   60.17   62.71   65.50   65.32   68.29   70.55
```

We have a pretty even distribution between 60 and 70. We can guess that these numbers are in some scale we don't know because 60 people employed obviously sounds ridiculous for a country. Maybe it's something like 60 million people.

5.3 Bivariate analysis

How can we continue? We can analyze whether they are related to any of the other variables. For example, we have the year, the population, and the amount of people in the armed forces.

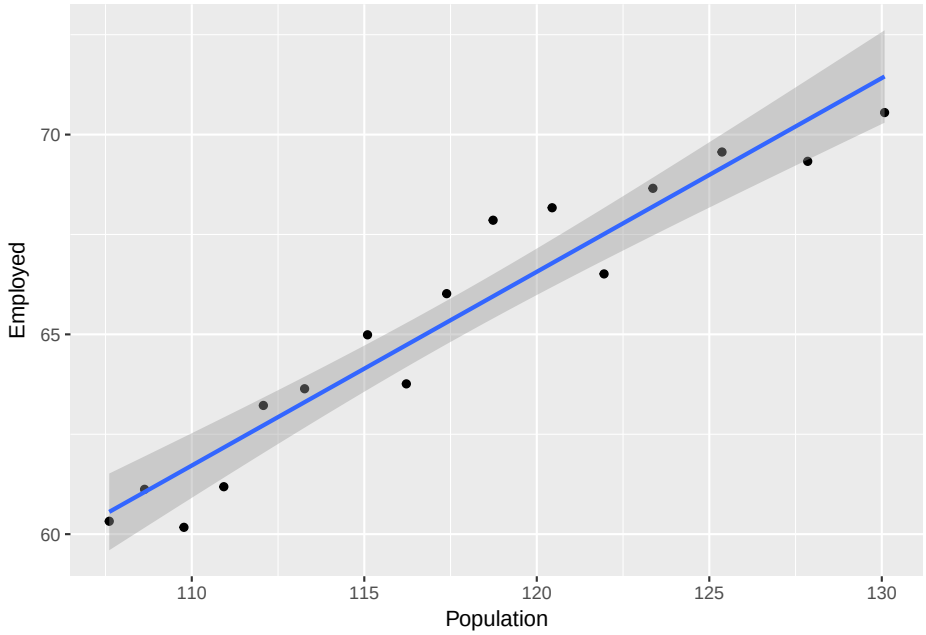
```
data(longley)
ggplot(longley, aes(x=Year, y=Employed)) +
  geom_point() +
  geom_smooth(method=lm)
```




```
m = mine(longley$Employed, longley$Year)
paste("This relationship's MIC is", m$MIC)
```

```
## [1] "This relationship's MIC is 1"
```

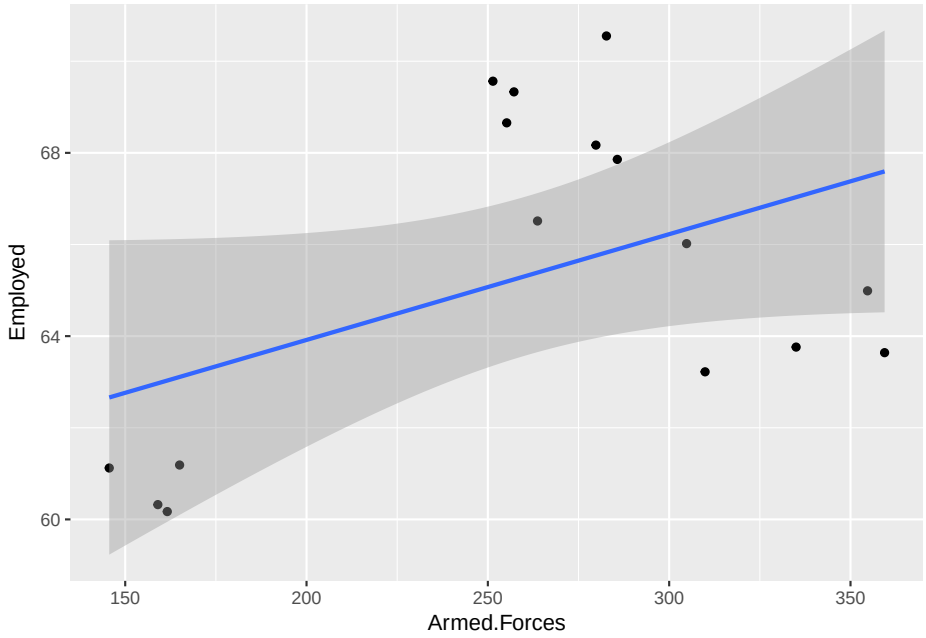
```
data(longley)
ggplot(longley, aes(x=Population, y=Employed)) +
  geom_point() +
  geom_smooth(method=lm)
```




```
m = mine(longley$Employed, longley$Population)
paste("This relationship's MIC is", m$MIC)
```

```
## [1] "This relationship's MIC is 1"
```

```
data(longley)
ggplot(longley, aes(x=Armed.Forces, y=Employed)) +
  geom_point() +
  geom_smooth(method=lm)
```

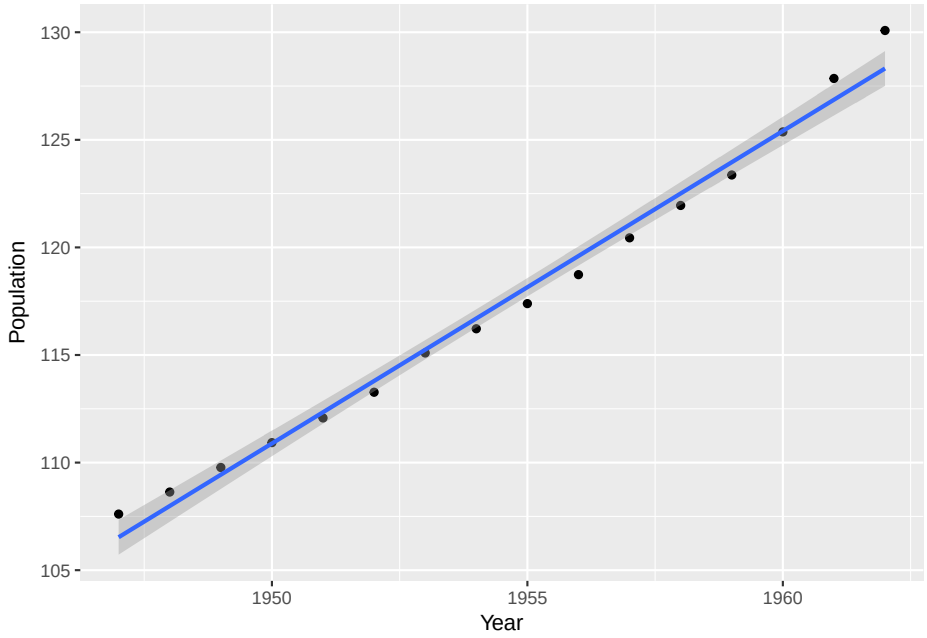



```
m = mine(longley$Employed, longley$Armed.Forces)
paste("This relationship's MIC is", m$MIC)
```

```
## [1] "This relationship's MIC is 0.311278124459133"
```

We can see a strong relationship with the population and the year, so either of them can "predict" the number of employed persons, and so can the amount of armed forces to a lesser degree. But what if we want to predict the population with the year, for example? Can we do that?

```
data(longley)
ggplot(longley, aes(x=Year, y=Population)) +
  geom_point() +
  geom_smooth(method=lm)
```




```
m = mine(longley$Year, longley$Population)
paste("This relationship's MIC is", m$MIC)
```

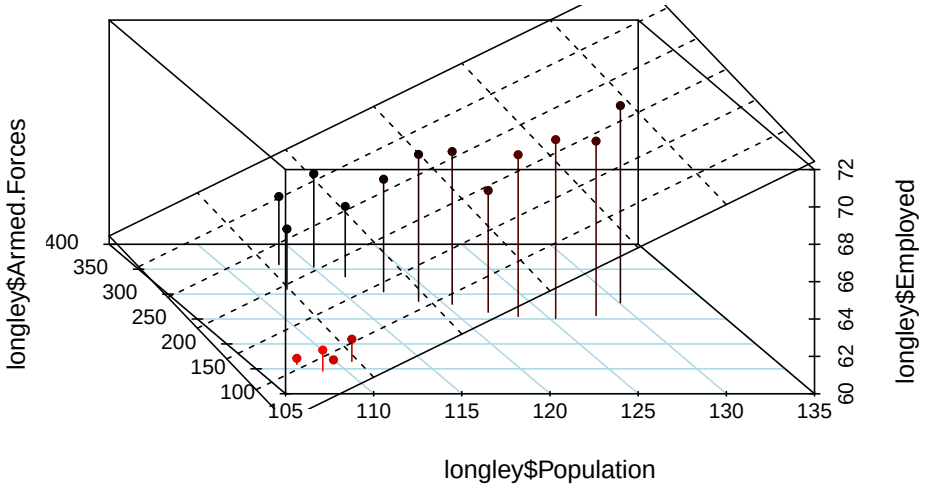
```
## [1] "This relationship's MIC is 1"
```

Yes, that works very well! But then what variable should I look at to "predict" the number of employed people?

5.4 Multivariate analysis

Actually, you should probably look at all the variables at once because most of the time, variables will form an n-dimensional shape that isn't captured by a bivariate analysis.

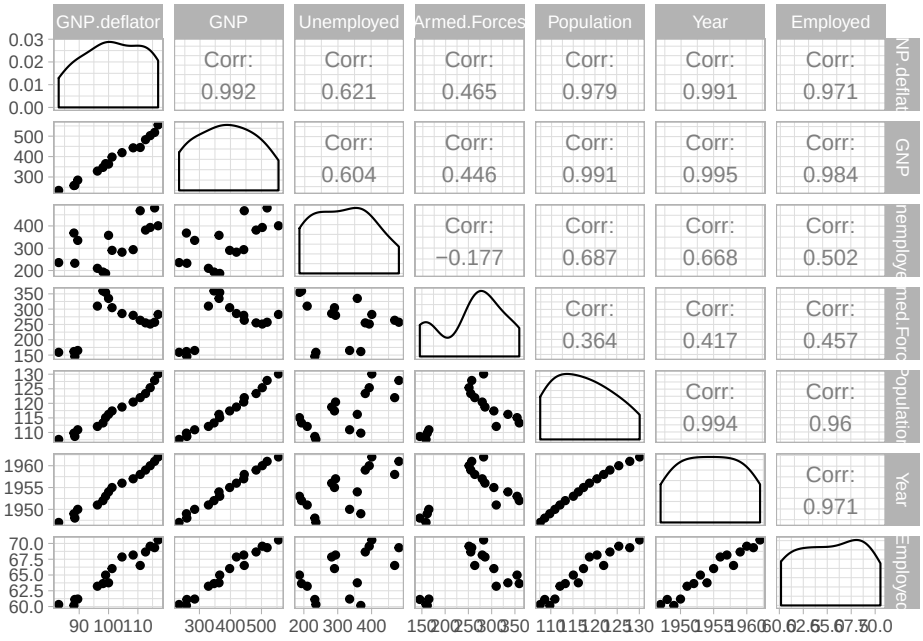
```
suppressMessages(library(scatterplot3d))
s3d = scatterplot3d(x = longley$Population,
                    y = longley$Armed.Forces,
                    z = longley$Employed, type='h',
                    angle=120, highlight.3d=T,
                    col.grid="lightblue",
                    main= "Population, Armed Forces & Employed",
                    pch=20)
attach(longley)
my.lm <- lm(Employed ~ Population + Armed.Forces)
s3d$plane3d(my.lm, lty.box = "solid")
```



However, with multivariate analysis comes a new challenge: **multicollinearity**.

This is what happens when two or more of the variables you are using to analyze your target variable are correlated. This is what happens in this example dataset; all the variables are very correlated.

```
suppressMessages(library(GGally))
ggpairs(longley) + theme_light()
```



This happens a lot in business when analyzing variables one by one. For example, suppose you see that a specific country brings leads to your website that convert at a 0.1% ratio while your base ratio is 4%. You might think, "This country's leads are low quality. I shouldn't try to bring those in."

But maybe a botnet is running on computers in that country and only specific IPs are generating all that bad traffic. If you had another variable that identified the IP addresses of compromised computers, you would see that the real variable that changes your conversion rate is `is_compromised` and not the country. In this case, supposing that the country has a lot of compromised computers, there is a collineality between the country and belonging to a botnet. The proper way of analyzing it is to view the two variables at once versus the conversion rate.

Machine-learning models are a good way to handle collinearity between variables. By choosing the right model, you can measure how good of a predictor each variable is. Another way is to use a dimensionality reduction technique, which reduces a big set of variables to a smaller set that can be easily visualized.

Chapter 6

Causation vs. Correlation

We could just write `Correlation is not causation` in this chapter, and the correct message would be delivered. But it is a good idea to explore what we are talking about.

The first argument focuses on why correlation is not causation. Look at these correlations and ask yourself if they make sense as cause-and-effect relationships.

6.1 Back to Wikipedia

First of all, we should revise our definition of causality:

> Causality [...] is the agency or efficacy that connects one process (the cause) with another process or state (the effect), where the first is understood to be partly responsible for the second, and the second is dependent on the first. In general, a process has many causes, which are said to be causal factors for it, and all lie in its past. An effect can in turn be a cause of many other effects. Although retrocausality is sometimes referred to in thought experiments and hypothetical analyses, causality is generally accepted to be temporally bound so that causes always precede their dependent effects.

(Wikipedia, 2016a)

So the main concept to take away here is that the cause happens **before** the effect. This sounds very simple and obvious, but is it? Is your dataset time constrained in such a way? When you are analyzing your data, are you sure one variable's value happened before the other value? Is your inference going that way or the opposite?

6.2 Machine learning help

A lot of times, we find relationships between variables that are useful to make predictions in machine learning. For example:

- We see a specific color or shape in a picture -> We think it's the sky.
- We track a user clicking on a button -> We predict that he'll pay.
- We find a specific curse word in some text -> We say it's text with a negative sentiment.

Does this mean any of these indicators are causes? Of course not! They are merely useful patterns that usually serve as a proxy to the real cause, which is not always visible to us because we don't possess all the universe's data. We have to settle for approximations, samples, and proxy variables.

6.3 A/B testing

So how can we test for cause and effect? How do I know if a specific change in a website will increase my conversion rates? How do I know if a new drug will be more effective than another one I'm giving or a placebo?

The solution has been extensively studied. It's called A/B testing. For example, you give one website or drug each to randomly assigned, equally distributed groups and see how they work out statistically. If your variation (B) works better than the original, you can conclude that you should implement the new website/drug. If variation (B) is equal or worse, you shouldn't.

6.4 Causal inference

In some situations, A/B testing can be unpractical or downright unethical. For example, if you need to test a new pricing model for your service, A/B testing a price change might lose *some customers* and start a scandal. In addition, measuring the effect of sending out a press release may be unpractical because you can't choose who reads it and who doesn't read it.

For these cases, the discipline of causal inference is very promising. We recommend that you check out these slides by coursera's Emily Glassberg Sands.

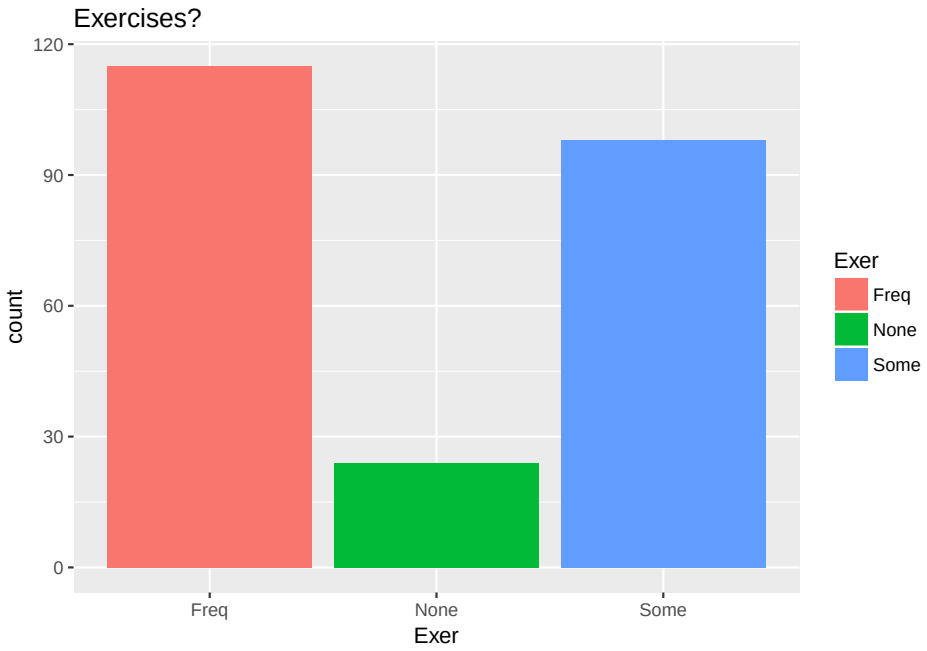
Chapter 7

Data Visualization

Data visualization is a very important discipline in data science. With the same dataset, a lot of wildly different visualizations can be made, some of which will be more useful than the rest. Let's try to make visualizations that make a real difference. We'll walk through some of the most common visualization tools and give you some tips.

7.1 Bar charts

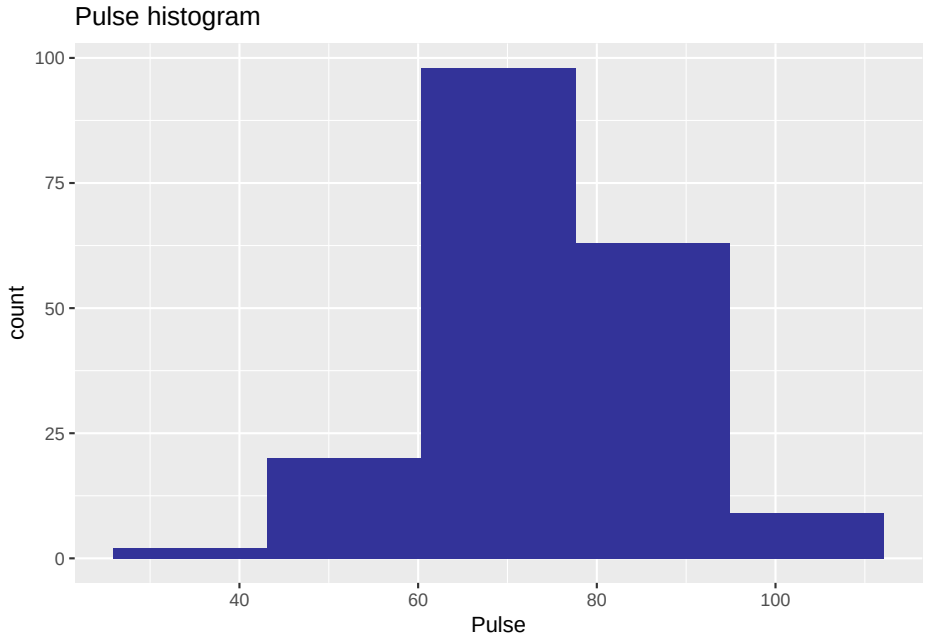
```
ggplot(survey, aes(Exer)) +
  geom_bar(aes(fill=Exer)) +
  ggtitle("Exercises?")
```



Bar charts are useful for quickly visualizing the different possible values of a categorical variable. In this case, the height of the bars are the amount of students, but you could have the sum of their heights or the tuition, if we had that value.

Continous variables are not directly usable with a bar chart, but you can discretize them by binning them into groups beforehand, which is effectively a histogram.

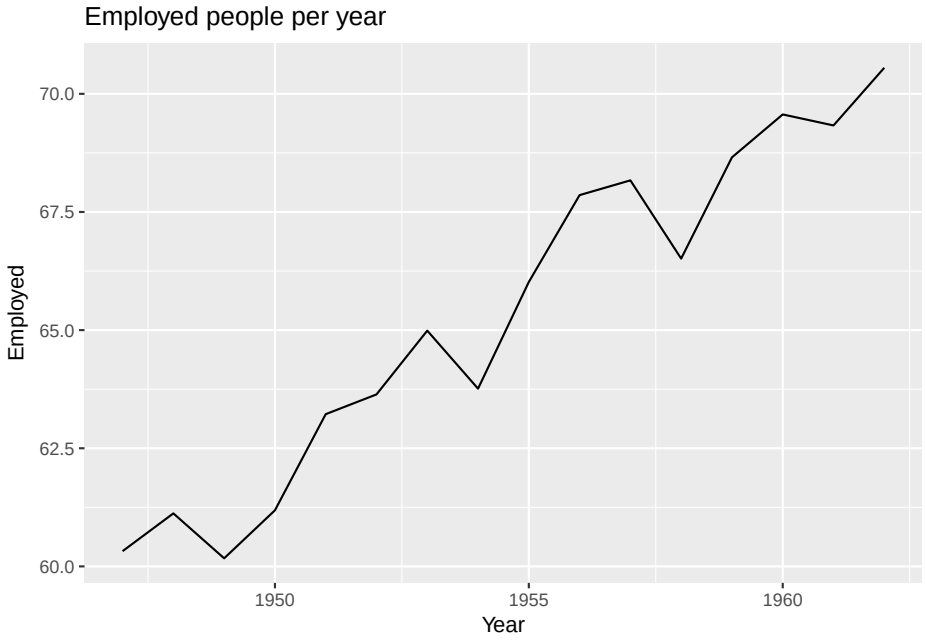
```
ggplot(survey, aes(Pulse)) +
  geom_histogram(bins=5, fill="#333399") +
  ggtitle("Pulse histogram")
```



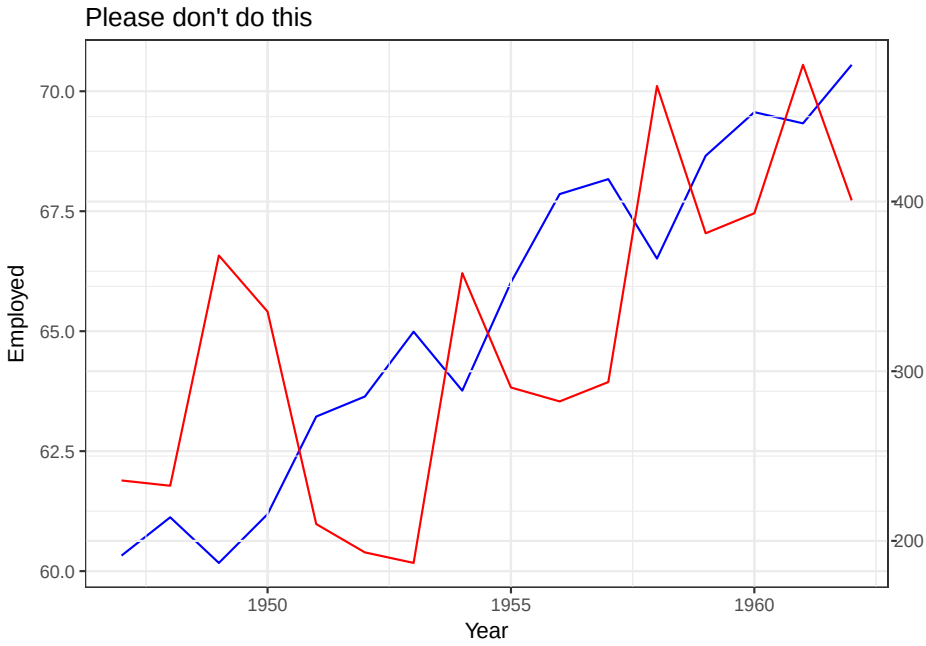
7.2 Line charts

Line charts are most useful for representing data when one of the variables is a time or is time dependant. An example of this could be found in the longley data.

```
ggplot(longley, aes(x=Year, y=Employed)) +
  geom_line() +
  ggtitle("Employed people per year")
```



One very bad practice we've seen occurs when people want to create line charts with another variable in a different scale. So they do something like this:



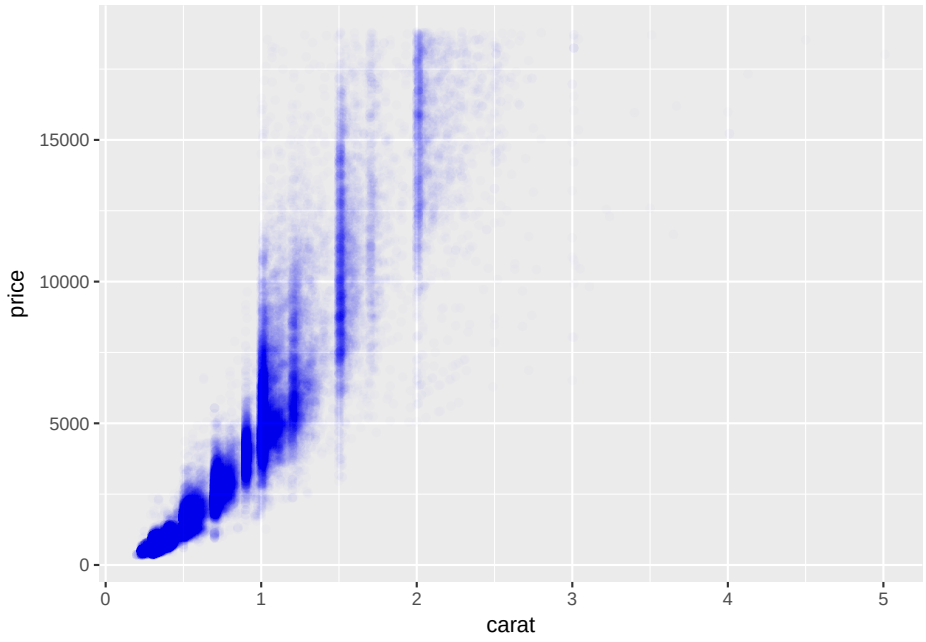
This is extremely unclear. I'll quote Hadley Wickham's arguments on StackOverflow:

> [...] I believe plots with separate y scales (not y-scales that are transformations of each other) are fundamentally flawed. Some problems:
>
> The are not invertible: given a point on the plot space, you can not uniquely map it back to a point in the data space.
>
> They are relatively hard to read correctly compared to other options. See A Study on Dual-Scale Data Charts by Petra Isenberg, Anastasia Bezerianos, Pierre Dragicevic, and Jean-Daniel Fekete for details.
>
> They are easily manipulated to mislead: there is no unique way to specify the relative scales of the axes, leaving them open to manipulation. Two examples from the Junkcharts blog: one, two
>
> They are arbitrary: why have only 2 scales, not 3, 4 or ten?
>
> You also might want to read Stephen Few's lengthy discussion on the topic Dual-Scaled Axes in Graphs Are They Ever the Best Solution?.

7.3 Scatter plots

Scatter plots can be very useful to get a first view of the relationship between two continuous variables. We've used it repeatedly throughout this book. One useful tip is that when you have a high volume of them, you can make each one a little more transparent. That'll help you see where you have more and less density.

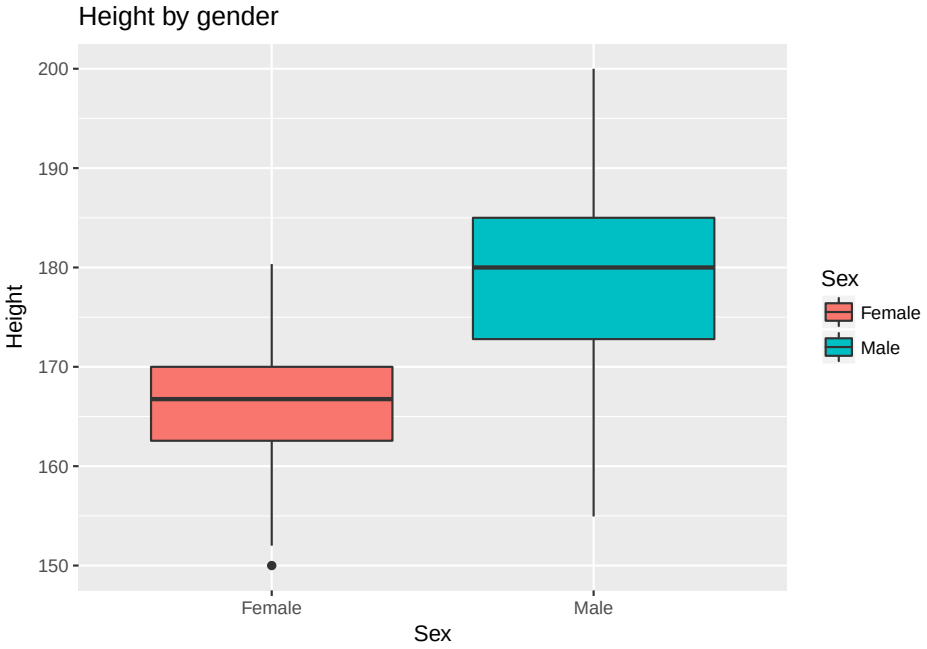
```
d <- ggplot(diamonds, aes(carat, price))
d + geom_point(alpha = 1/100, color="blue")
```



7.4 Box plots

Box plots are a very useful way to measure distributions across many groups. For example, let's assume that we want to understand how heights are distributed between men and women.

```
filter(survey, !is.na(Height) & !is.na(Sex)) %>%
ggplot(aes(x=Sex, y=Height)) +
  geom_boxplot(aes(fill=Sex)) +
  ggtitle("Height by gender")
```



Even if you don't know how a boxplot works, you can clearly see that the height variable looks smaller on women, right? Let's analyze each component and what it quickly tells us about the distribution:

- The box tells us the interquartile range, i.e., the percentile 25 and 75 of the distribution; so in the box, we already have 75% of all observations.
- The line crossing the box is the median, which is the value of the 50th percentile.
- The lines above and below the box are the "Whiskers." This varies by boxplot implementation. Sometimes it's the minimum and maximum values. Here in ggplot's implementation, they are 1.5 times the interquartile range, so if the range went from 165 to 170, the whiskers would be $(170 - 165) * 1.5 = 7.5$. They would go from 170 to 177.5 and from 157.5 to 165.
- If any point is outside the box and the whiskers, it is marked as a little dot that can be considered a potential outlier, allowing you to understand quickly if you may have outliers in that distribution.

7.5 Pie/donut charts

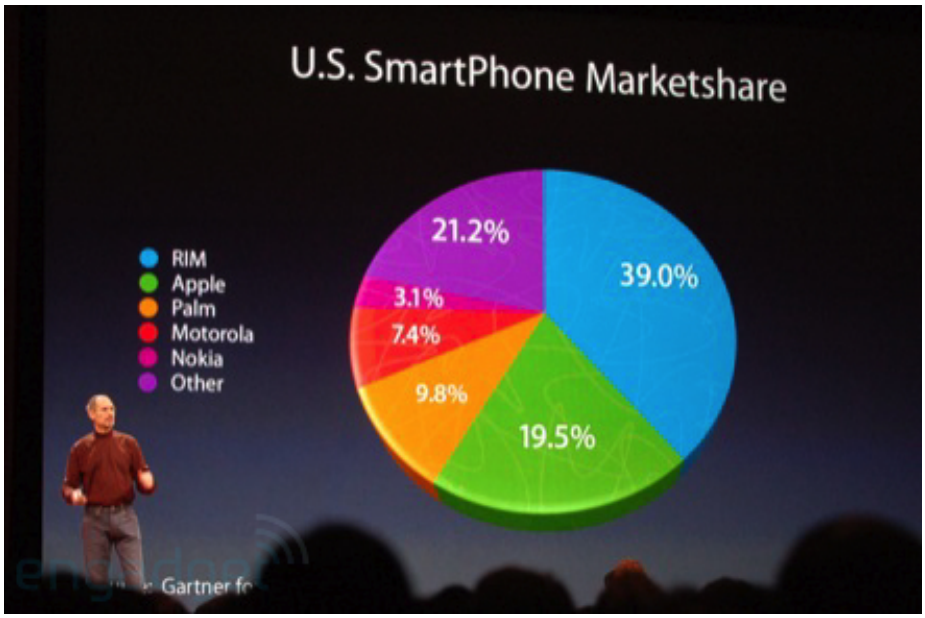
> "In a sense, it might be construed as an insult to a man's intelligence to show him a pie chart." K.G. Karsten, Charts and Graphs (1923)

Please, never do pie charts or donut charts. Even more important, never

do them in 3D. That's the worst combination. This is a rule almost as strong as *Correlation doesn't imply causation*. Whenever you are about to use a pie or donut chart, you should consider switching to a bar chart.

Let's go through some problems:

1. Pie chart categories are difficult to compare. We don't process irregular shapes in our brains as easily as we do bars in a barchart, for example, and there is no way to put two parts of a circle side by side to easily see which one is higher. Data visualization should make it easier and not more difficult to understand the data.
2. Comparing two pie charts is almost impossible.
3. Color-blind people will find it completely unusable depending on the color mix.
4. Pie charts are very easily to manipulate.



Which is bigger: `Apple` or `Other`? Now look at the percentages.

Chapter 8

Machine Learning

Any data you are looking at is information from the past. Even if it's real-time information, by the time the data has reached the computer, the data is already in the past. We usually look at it to somehow predict the future.

If we are looking at last year's sales for this month, a lot of the time we're likely trying to estimate how much we will sell this year. If we are looking at which browsers were the most used on a website for the last three months, we are likely to optimize for the top ones so future visits to the website are as pleasant as possible, and so on.

Some of these analyses can be done reasonably easily, but some require a more complex solution. This is where machine learning comes in.

8.1 Supervised learning

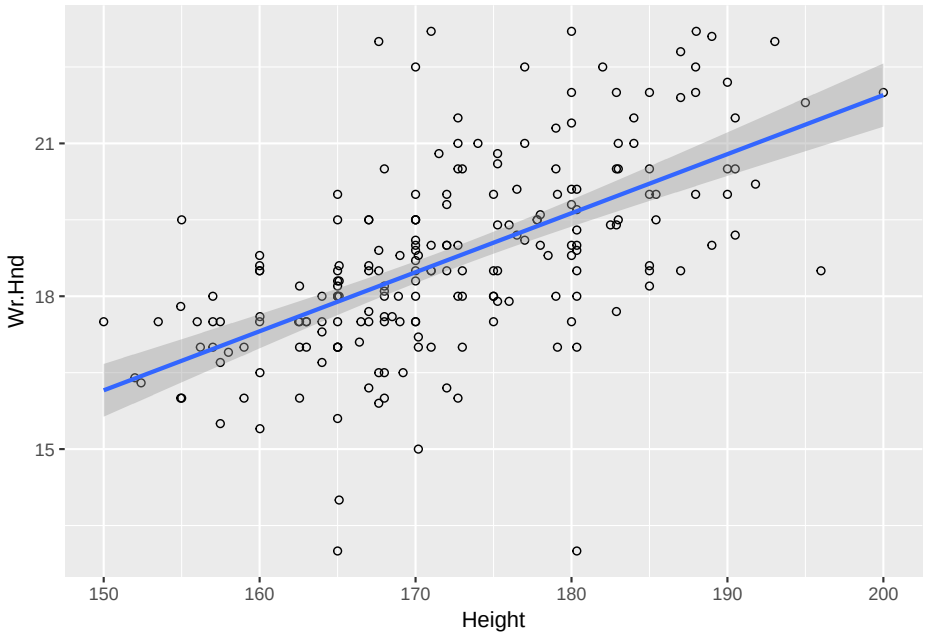
When we talk about *supervised learning*, we mean we are "guiding" the algorithm somehow with a target variable. Suppose with our original survey dataset that we want to infer the `Height` of a person using the other attributes. That's called a regression problem. We are trying to find a numeric value using one or more "regressors".

What if we wanted to guess whether a person exercises based on the other variables? That would be a multi-class classification problem in our dataset because we have `None`, `Some`, and `Frequently`.

8.1.1 Regression

For example, for the age we have:

```
model = lm(Height ~ Wr.Hnd, data=survey)
surveyH = filter(survey, !is.na(Height) & !is.na(Wr.Hnd))
ggplot(surveyH, aes(x=Height, y=Wr.Hnd)) +
    geom_point(shape=1) +
    geom_smooth(method=lm)
```



Now we have a model, but what does this mean? If I give the model data from Wr.Hnd (distance from tip of thumb to tip of little finger of spread writing hand in cm), I will get an estimate of the height (cm):

```
newData = data.frame(Wr.Hnd=20)
result = predict(model, newData)
result
```

```
##       1
## 176.286
```

With a span of 20cm, we can expect a height of around ~176cm. Cool right?. Well, there are a couple of issues:

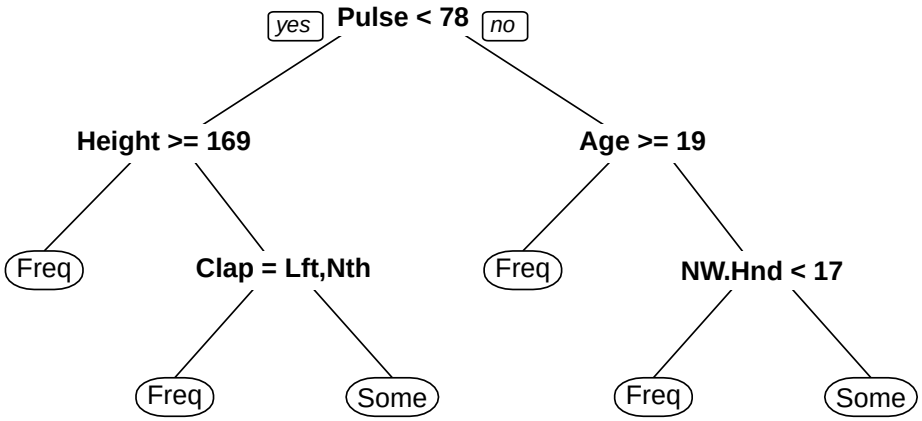
- We didn't separate a hold-out dataset, which the model doesn't see, so we could validate the accuracy of the model.
- We didn't consider more than one variable, so we probably missed out on information contained in other variables.
- We didn't check the accuracy so we don't know how much lower or higher the height can be than the predicted one.

All these pitfalls considered, let's continue with another example.

8.1.2 Classification

Let's try to classify exercisers by their other attributes using a decision tree.

```
library(rpart)
library(rpart.plot)
msmoke = rpart(Exer ~ ., data=na.omit(survey))
prp(msmoke)
```



This decision tree has the advantage of being easily interpretable, although it uses many variables. In reality, we usually use far more complex algorithms that are harder to interpret because the shape of the data is not linear (the decision tree is a linear classifier).

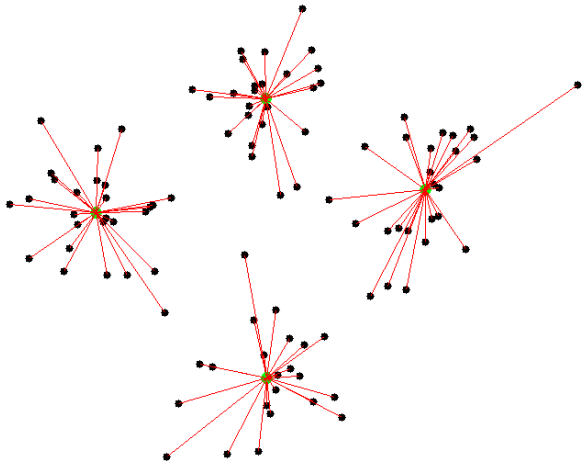
A good example of a hard-to-interpret classifier is the random forest, which is an ensemble of hundreds or thousands of small decision trees that vote for different outcomes. Like a democracy, the outcome with the most votes will be the output of the model.

8.2 Unsupervised learning

When we don't have a definite objective, we are just exploring the dataset, and we think there might be an underlying structure we are not seeing, unsupervised learning may be the way to go.

One of the most common techniques for this kind of learning is clustering. With clustering, we try to find groups in the data that may reveal useful information about our dataset (for example, this can be used to discover segments in a user base).

One of the simplest clustering algorithms is k-means, which finds clusters by creating "centroids", e.g., with only two dimensions:



Source: (Shabalin, 2016)

This same algorithm can be applied to highly dimensional spaces and it still works, but it can't be so easily visualized.

8.3 Learn more

Creating machine-learning algorithms is almost an art in itself that involves knowing how to prepare data for a model and avoiding pitfalls, which is beyond the scope of this book. If you are interested in machine learning, you may enjoy reading this book by Pablo Casas, our data scientist with the most experience of anyone on our team: http://livebook.datascienceheroes.com/.

Bibliography

Buffer (2015). The complete guide to utm codes: How to track every link and all the traffic from social media. [Online; accessed 2-December-2016].

Reshef, D. N., Reshef, Y. A., Finucane, H. K., Grossman, S. R., McVean, G., Turnbaugh, P. J., Lander, E. S., Mitzenmacher, M., and Sabeti, P. C. (2011). Detecting novel associations in large data sets. *Science*, 334(6062):1518–1524.

Shabalin, A. A. (2016). K-means clustering. — andrey's website. [Online; accessed 26-December-2016].

Wikipedia (2016a). Causality — wikipedia, the free encyclopedia. [Online; accessed 7-November-2016].

Wikipedia (2016b). Data — wikipedia, the free encyclopedia. [Online; accessed 2-December-2016].