Statistical Modelling

Léo Belzile

version of 2020-06-15

Contents

Preliminary remarks 5
- 0.1 Basics of R . . . 5
- 0.2 Tutorial 1 . . . 7

1 Introduction 11
- 1.1 Statistical inference versus predictions . . . 11
- 1.2 Reminders and Objectives . . . 11
- 1.3 Population and sample . . . 11
- 1.4 Testing statistical hypotheses . . . 12
- 1.5 Exploratory data analysis . . . 13

2 One-sample and two-sample location tests 15
- 2.1 One-sample and paired t-test . . . 15
- 2.2 Two-sample t-test . . . 17
- 2.3 Wilcoxon rank sum test . . . 24

3 Linear regression 31
- 3.1 The `lm` function . . . 33

Preliminary remarks

While we show how to implement statistical tests and models in SAS in the class note, the R programming language provides a free alternative. Visit the R-project website to download the program. The most popular graphical cross-platform front-end is RStudio Desktop.

R is an object-oriented interpreted language. It differs from usual programming languages in that it is designed for interactive analyses. Since R is not a conventional programming language, my teaching approach will be learning-by-doing. The benefit of using Rmarkdown is that you see the output directly and you can also copy the code.

You can find several introductions to R online. Have a look at the R manuals or better at contributed manuals. A nice official reference is An introduction to R. You may wish to look up the following chapters of the R language definition (Evaluation of expressions and part of the Objects chapter).

If you favor online courses, Data Camp offers a free introduction to R.

0.1 Basics of R

0.1.1 Help

Help can be accessed via `help` or simply `?`, e.g., `help("Normal")`. See R page about help files.

0.1.2 Basic commands

Basic R commands are fairly intuitive, especially if you want to use R as a calculator. Elementary functions such as `sum`, `min`, `max`, `sqrt`, `log`, `exp`, etc., are self-explanatory.

Some (unconventional) features of the language:

- R is case sensitive.
- Use `<-` for assignments to a variable, and `=` for matching arguments inside functions
- Indexing in R starts at 1, not 0.
- Most functions in R are vectorized and loops are typically inefficient.
- Integers are obtained by appending `L` to the number, so `2L` is an integer and `2` a double (`numerical`).

Besides integers and doubles, the common types are

- logical (`TRUE` and `FALSE`);
- null pointers (`NULL`), which can be assigned to arguments;
- missing values, namely `NA` or `NaN`. These can also be obtained a result of invalid mathematical operations such as `log(-2)`.

Beware! In R, invalid calls will often returns something rather than an error. It is therefore good practice to check that the output is sensical.

0.1.3 Linear algebra in R

R is an object oriented language, and the basic elements in R are (column) vector. Below is a glossary with some useful commands for performing basic manipulation of vectors and matrix operations:

- `c` concatenates elements to form a vector
- `cbind` (`rbind`) binds column (row) vectors
- `matrix` and `vector` are constructors
- `diag` creates a diagonal matrix (by default with ones)
- `t` is the function for transpose
- `rep` creates a vector of duplicates, `seq` a sequence. For integers i, j with $i < j$, `i:j` generates the sequence $i, i+1, \ldots, j-1, j$.

Subsetting is fairly intuitive and general; you can use vectors, logical statements. For example, if `x` is a vector, then

- `x[2]` returns the second element
- `x[-2]` returns all but the second element
- `x[1:5]` returns the first five elements
- `x[(length(x) - 5):length(x)]` returns the last five elements
- `x[c(1, 2, 4)]` returns the first, second and fourth element
- `x[x > 3]` return any element greater than 3. Possibly an empty vector of length zero!
- `x[ x < -2 | x > 2]` multiple logical conditions.
- `which(x == max(x))` index of elements satisfying a logical condition.

For a matrix `x`, subsetting now involves dimensions: `[1,2]` returns the element in the first row, second column. `x[,2]` will return all of the rows, but only the second column. For lists, you can use `[[` for subsetting by index or the `$` sign by names.

0.1.4 Packages

The great strength of R comes from its contributed libraries (called packages), which contain functions and datasets provided by third parties. Some of these (`base`, `stats`, `graphics`, etc.) are loaded by default whenever you open a session.

To install a package from CRAN, use `install.packages("package")`, replacing `package` by the package name. Once installed, packages can be loaded using `library(package)`; all the functions in `package` will be available in the environment.

– There are drawbacks to loading packages: if an object with the same name from another package is already present in your environment, it will be hidden. Use the double-colon operator `::` to access a single object from an installed package (`package::object`).

0.2 Tutorial 1

0.2.1 Datasets

- datasets are typically stored inside a `data.frame`, a matrix-like object whose columns contain the variables and the rows the observation vectors.
- The columns can be of different types (`integer`, `double`, `logical`, `character`), but all the column vectors must be of the same length.
- Variable names can be displayed by using `names(faithful)`.
- Individual columns can be accessed using the column name using the `$` operator. For example, `faithful$eruptions` will return the first column of the `faithful` dataset.
- To load a dataset from an (installed) R package, use the command `data` with the name of the `package` as an argument (must be a string). The package `datasets` is loaded by default whenever you open R, so these are always in the search path.

The following functions can be useful to get a quick glimpse of the data:

- `summary` provides descriptive statistics for the variable.
- `str` provides the first few elements with each variable, along with the dimension
- `head` (`tail`) prints the first (last) n lines of the object to the console (default is $n = 6$).

We start by loading a dataset of the Old Faithful Geyser of Yellowstone National park and looking at its entries.

```
# Load Old faithful dataset
data(faithful, package = "datasets")
# Query the database for documentation
?faithful
# look at first entries
head(faithful)
```

```
##   eruptions waiting
## 1       3.6      79
## 2       1.8      54
## 3       3.3      74
## 4       2.3      62
## 5       4.5      85
## 6       2.9      55
```

```
str(faithful)
```

```
## 'data.frame':    272 obs. of  2 variables:
##  $ eruptions: num  3.6 1.8 3.33 2.28 4.53 ...
##  $ waiting  : num  79 54 74 62 85 55 88 85 51 85 ...
```

```
# What kind of object is faithful?
class(faithful)
```

```
## [1] "data.frame"
```

Other common classes of objects:

- `matrix`: an object with attributes `dim`, `ncol` and `nrow` in addition to `length`, which gives the total number of elements.
- `array`: a higher dimensional extension of `matrix` with arguments `dim` and `dimnames`.
- `list`: an unstructured class whose elements are accessed using double indexing `[[ ]]` and elements are typically accessed using `$` symbol with names. To delete an element from a list, assign `NULL` to it.
- `data.frame` is a special type of list where all the elements are vectors of potentially different type, but of the same length.

0.2.2 Graphics

The `faithful` dataset consists of two variables: the regressand `waiting` and the regressor `eruptions`. One could postulate that the waiting time between eruptions will be smaller if the eruption time is small, since pressure needs to build up for the eruption to happen. We can look at the data to see if there is a linear relationship between the variables.

An image is worth a thousand words and in statistics, visualization is crucial. Scatterplots are produced using the function `plot`. You can control the graphic console options using `par` — see `?plot` and `?par` for a description of the basic and advanced options available.

Once `plot` has been called, you can add additional observations as points (lines) to the graph using `point` (`lines`) in place of `plot`. If you want to add a line (horizontal, vertical, or with known intercept and slope), use the function `abline`.

Other functions worth mentioning at this stage:

- `boxplot` creates a box-and-whiskers plot
- `hist` creates an histogram, either on frequency or probability scale (option `freq = FALSE`). `breaks` control the number of bins. `rug` adds lines below the graph indicating the value of the observations.
- `pairs` creates a matrix of scatterplots, akin to `plot` for data frame objects.

– There are two options for basic graphics: the base graphics package and the package `ggplot2`. The latter is a more recent proposal that builds on a modular approach and is more easily customizable — I suggest you stick to either and `ggplot2` is a good option if you don't know R already, as the learning curve will be about the same. Even if the display from `ggplot2` is nicer, this is no excuse for not making proper graphics. Always label the axis and include measurement units!

Chapter 1

Introduction

The goal of this course is to introduce the fundamental principles of statistical inference as well as advanced statistical methods used in business intelligence problems. We will also see applications of these methods using (mostly) simulated and real data applications using dedicated software. The SAS material is presented throughout the course slides, whereas R will be presented in this online book.

This introductory chapter touches many topics that may be covered in undergraduate statistics courses, including notions about hypothesis test (null and alternative hypothesis, statistics, null distributions, levels and power of test, confidence interval).

This chapter also includes a short description of population and samples, basic graphics (scatterplot, violinplots, histograms, box-and-whisker plots) and tests for one-sample location problems (signed test, signed rank test and one-sample and paired t-tests). Recommended reading are Chapter 1–4 of the OpenIntro book "Introductory Statistics with Randomization and Simulation".

1.1 Statistical inference versus predictions

Statistical inference deals with explanatory models — used for answering scientific questions, i.e. test research hypotheses concerning variable effects. However, these models can also be used to make predictions.

1.2 Reminders and Objectives

This section covers some basics about population and samples, hypothesis tests and duality between confidence intervals and p-values.

1.3 Population and sample

Statistics is about drawing inference from a large group given a representative subset of limited size; we cannot measure for practical and financial reasons a characteristic such as height to get the average

length of a human. Often, it is also unnecessary: conclusions drawn from a sample are good enough.

The population is the target group under study. A sample is a representative subset of the population for which we measure a characteristic. For example, if we are interested in voting intentions for the next federal election, the population would consist of all Canadian citizens aged 18 or more at the day the vote takes place. Pollsters may select a small group of eligible voters and ask them for their voting preferences: you can get pretty good estimates even when 17.5 millions people vote, as the margin of error of a probabilistic sample with 1000 participants is about 3% with 95% confidence level.

The population is often defined in the context of a study, but one should be careful in not making this definition too narrow. If you want to do a study of breast cancer, you do not want to define the population to be all patients at a given hospital who received a diagnostic of breast cancer; rather, you should aim to have a broad enough definition so that you can generalize your results. Choosing a sample is hard, and an entire branch of statistics, survey sampling, deals specifically with this. For opinion polls, pollster often have polls and databases of phone numbers to pick from. What makes it more challenging nowadays is that a large share of individuals do not own a landline anymore (and so their phone number is not public), but calling only cellphones leads to bias because it undersample elderly people; selecting exclusively one or the other leads to bad samples.

1.4 Testing statistical hypotheses

Hypothesis testing is ubiquitous in statistics, yet often poorly understood. The topic is thus presented in an intuitive manner through examples by making an analogy with trial for murder. We review the formulation of null and alternative hypotheses, the choice of test statistic, derivation of its null distribution and finally conclusions in light of p-value. As an illustrative example, we test for equal means for two groups using a permutation test.

The best analogy I have for hypothesis testing is a criminal trial: you are a jury member for a murder case and are asked to deliver a verdict, which will be either guilty or not guilty, based on evidence presented in court. By default, the presumption of innocence apply so your default position is that the accused is not guilty and you analyze evidences in this optic to avoid judicial errors (do not condemn an innocent to death). Only if the evidences presented by the prosecutor are overwhelming will you declare the person guilty.

We can make a parallel with hypothesis testing as follows:

- there are only two possible options considered, which correspond to the null hypothesis H_0 (the accused is innocent of the murder) and the alternative H_1 (the accused committed the murder).
- the test statistic comprises all evidences. There are many different choices, but not all of them are good (compare DNA samples with circumstantial evidences).
- rejecting the null hypothesis corresponds to the guilty verdict, i.e., reject H_0.
- sometimes, there are plenty of evidences presented even when the individual accused of murder is innocent. To minimize the risk of judicial error, we only reject if these are overwhelming, meaning that the probability of happening by chance for an innocent are equal to some tolerance level α, specified beforehand. The level of a test does not change.
- if we fail to reject the null hypothesis, corresponding to a verdict of not guilty, this means either of two things: the person is innocent or we have a reasonable doubt. The jury only weights the

evidence in light of the apriori assumption that the accused is innocent unless proven otherwise, so we cannot conclude anything about innocence.
- our ability to correctly declare a murderer guilty is called power. The choice of test statistic (evidences) determines how well we can confound murdererers. Power is a probability: the higher, the better. A test with no power is useless. If you have to choose a single evidence from the proof, you might select 'blood stains on the accused's clothes which matches the DNA victim' over 'he happened to be in the city as the victim when the murder was committed'.

The interplay between Type I and Type II errors are best displayed graphically. Nathaniel Stevens (University of Waterloo) made a nice Shiny app. In these plots, α is the significance level (usually 5%) corresponding to the type I error we are willing to commit. The effect size measures the difference between the null hypothesis and the true effect size (think about measuring height of people in two populations, with the null hypothesis being that there is no difference). If there was truly a difference, but the latter was small, say 0.01cm, it would be confounded with the measurement variability. On the other hand, if the true difference is 10cm, it would be pretty obvious. The main factors affecting power are sample size and variability in the population; the variability of the mean estimate goes down like $\sigma/\sqrt{n}$, so decreases when either (a) σ decreases or (b) n increases. With large enough samples, all differences will be detected. The type I error is the surface represented in red (α), whereas the type II error (false negative) is β in blue. The power is the area of the density in dashed blue; the larger, the more the test is capable of detecting when the null hypothesis is false.

1.5 Exploratory data analysis

Looking at the data is arguably the most important of an analysis. This step consists in looking at summary statistics (mean, quantiles, min and max) and to plot the data, one variable at a time and then pairs. This will allow you to detect outliers (for example, missing values were often encoded with -1 or 999 in the old days when only numerical values could be saved by devices). Real data is messy, and often artefacts (e.g., rounding, ties) are most easily viewed from scatterplot, histograms, box-and-whiskers plots, etc.

- a box-and-whiskers plot (or boxplot), consists of a box composed by the 25, 50 and 75 percentiles. The median separates the box, while the whiskers extend beyond 1.5 times the interquartile range (the length of the box). Values that fall outside of this range are often represented by dots and would be extreme points, if the sample was normally distributed.
- a histogram bins observations and can be useful to detect outliers and multimodality. The number of bins is often selected automatically.
- a kernel density estimator represents local fit of the density function and is a continuous analog of the histogram if the latter is on the density scale (as opposed to frequency). The kernel density is more sensitive to local deviations and may more easily capture discreteness. The bandwidth controls the amount of neighboring data used for the fit. Optimal (and default) choices are typically selected.

Chapter 2

One-sample and two-sample location tests

2.1 One-sample and paired t-test

We covered this example of Tech3Lab comparing the reaction time between individuals for two tasks, texting and speaking on the phone, while walking. The response variable is reaction time in seconds to the presence of a bicycle moving towards the participant.

The data can be found in `distraction.txt` and can be loaded in R in a `data.frame` object.

```
# get URL of the file
url <- "https://lbelzile.bitbucket.io/MATH60619A/distraction.txt"
# load data, stating header
data <- read.table(url, header = TRUE)
```

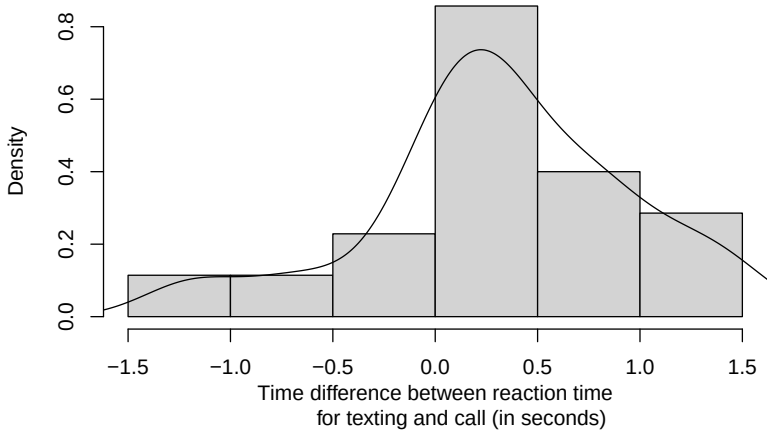
The columns of a `data.frame` object can be accessed using the `$` operator.

We can calculate descriptive statistics for the data by using the commands `summary`, or `mean`, `sd`, etc.

Our goal is to compare the average reaction time for texting `t` and for call `c`, say $\mu_C = \mu_T$. Since the individuals complete the two assignments (the order is random), we are looking at a paired t-test, which amounts to comparing the null $\mathcal{H}_0 : \mu_D = 0$, where μ_D is the difference (in seconds) between the reaction time for texting vs phone conversation in the population. We can assess this graphically through an histogram of the distribution.

```
# create new vector as difference in time
D <- data$t - data$c
# histogram, on the probability scale
hist(D, probability = TRUE,
     xlab = "Time difference between reaction time
         for texting and call (in seconds)",
     main = "")
```

```
# add density estimate
lines(density(D))
```



The two sample mean (standard error) are different, 1.39 (0.53) seconds for texting and 1.08 (0.4) seconds for call. To check if this is really the case, we perform a paired t-test.

```
ttest <- t.test(x = data$t, y = data$c,
                alternative = "two.sided", paired = TRUE)
print(ttest)
```

```
##
##  Paired t-test
##
## data:  data$t and data$c
## t = 3, df = 34, p-value = 0.006
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.094 0.532
## sample estimates:
## mean of the differences
##                    0.31
```

The output provides the value of the test statistic $t = \overline{D}/\mathsf{se}(\overline{D}) = 2.91$ with associated p-value . We strongly reject the null hypothesis that $\mu_C = \mu_D$ at level 5% (even 1%), meaning that the reaction time is significantly longer on average for texting with an estimated 95% confidence of [0.09, 0.53].

The null distribution of the test statistic is $\mathcal{T}_{34}$, so we can look at how extreme our test statistic is

relative to a typical value from the null distribution.

We can also do a one-sample t-test using the same function. The `hours` dataset consists of a sample of size $n = 100$ of workers who had attained at least a college degree. We could test whether the participants work 40 hours a week (two sided alternative), or if they work more than that. In the latter case, the null hypothesis is thus $\mathcal{H}_0 : \mu \geq 40$, where μ is the average number of weekly work hours, against the alternative $\mathcal{H}_1 : \mu < 40$.

```
url <- "https://lbelzile.bitbucket.io/MATH60619A/hours.txt"
data <- read.table(file = url, header = TRUE)
ttest_twosided <- t.test(data, alternative = "two.sided", mu = 40)
ttest_onesided <- t.test(data, alternative = "greater", mu = 40)
```

In both cases, we reject the null hypothesis at level 5%, but the one-sided alternative has a smaller p-value. Think about why it does not make sense to test for $\mathcal{H}_0 : \mu \leq 40$ versus $\mathcal{H}_1 : \mu > 40$ in view of the sample mean.

2.2 Two-sample t-test

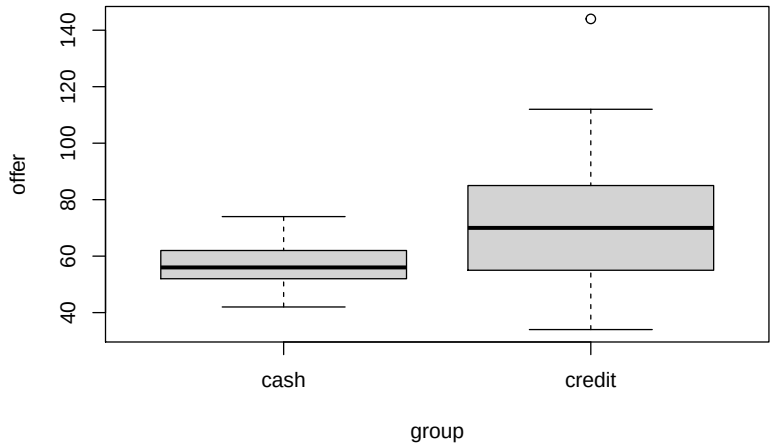
We use the same example covered in slides for the `tickets` data. Since the research question is "Does paying by credit card encourage consumers to pay more?", this calls for a one sided test, with $\mathcal{H}_1 : \mu_{\text{cred}} > \mu_{\text{cash}}$ and de facto the null hypothesis must be $\mathcal{H}_0 : \mu_{\text{cred}} \leq \mu_{\text{cash}}$. The alternative hypothesis is the one we would be interested in checking, but because we are the Devil's advocate, the null hypothesis is the opposite and includes all other possibilities. Note that we only need to consider the case $\mu_{\text{cred}} = \mu_{\text{cash}}$ (why?), so one often write this as the null hypothesis.

We can start by loading the data and preliminary visual inspection. In light of the boxplot, it seems that the variance in the two groups differs.

```
url <- "https://lbelzile.bitbucket.io/MATH60619A/tickets.txt"
tickets <- read.table(url, header = TRUE) #load data
head(tickets) #print first six lines
```

```
##   offer group
## 1    62  cash
## 2    44  cash
## 3    46  cash
## 4    48  cash
## 5    50  cash
## 6    58  cash
```

```
boxplot(offer ~ group, data = tickets)
```




```
tickets$group <- factor(x = tickets$group, #cast binary to categ
                        labels = c("cash","credit")) #assign meaningful
# summarize data by group (equivalent of PROC MEANS in SAS)
with(data = tickets, #use data
     expr = by(data = offer, FUN = summary, INDICES = group))
```

```
## group: cash
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      42      52      56      57      62      74
## ------------------------------------------------------------
## group: credit
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      34      55      70      72      85     144
```

```
with(data = tickets,
     expr = by(data = offer, FUN = sd, INDICES = group))
```

```
## group: cash
## [1] 7.5
## ------------------------------------------------------------
## group: credit
## [1] 24
```

```
# alternatively
# sd(tickets$offer[tickets$group == "cash"])
```

The option `header = TRUE` specifies that the first line of the file contains the name of the variables. While `read.table` is the default for command to read tables (with options `sep` for the separator, we will also encounter `read.csv` for comma-separated values. The data is stored in a `data.frame` object. The command `str` gives a description of the observations (with the type) and `head` prints the first lines. Other relevant commands include `ndim`, `nrow` and `ncol` that give the dimensions of the data, the number of rows and the number of columns, respectively.

Reference to the columns of a data frame is made using their column names (`colnames(tickets)`) using `$`, so `tickets$group` returns the second column with the binary variable. Alternatively, we can use the `attach` command to attach the dataset, in which case the variables are now accessible directly. Beware with this, as there is a risk of having multiple objects with the same name. A good practice is to `detach` the dataset after use.

When we load a dataset, the default option for strings is to cast them to factor (i.e., categorical variables). We do this likewise for the binary variables, even if in this case this makes no difference (but it is good practice).

Summary statistics showed that the mean amount offered for tickets is 56.61 (7.5) for the group paying by cash and 71.61 (23.6), but we need to perform a test in order to know whether such a difference, -15.01, is significative. Since our null hypothesis is that customers are willing to pay more by credit card, we perform a one-sided t-test.

```
ttest <- t.test(formula = offer ~ group,
       data = tickets,
       alternative = "less", #one-sided
       var.equal = TRUE) #by default FALSE
print(ttest)
```

```
##
##  Two Sample t-test
##
```

```
## data:  offer by group
## t = -3, df = 62, p-value = 0.0005
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##  -Inf -7.8
## sample estimates:
##   mean in group cash mean in group credit
##                   57                   72
```

In R, the default option for the function `t.test` is Welch's test (`var.equal = FALSE`), since the latter is valid whether or not the variance of the two groups are equal. The value of the test statistic is -3.48, which should follow a Student t distribution under H_0 with degrees of freedom, leading to a p-value smaller than 10×10^{-3}. We reject the null at level $\alpha = 0.05$ in favor of the alternative that people paying by credit card are willing to spend more than those paying by cash with the lower bound of the 95% confidence interval for this difference being -7.8.

The equality test for the variance is not calculated directly by `t.test` in R, but the function `var.test` implements an F test for this hypothesis (Levene's test is in `car::leveneTest`). We have overwhelming evidence that the variance for the two groups are unequal, so we can should Welch's test rather that the two-sample t-test. The conclusions are the same in this case, namely that such a difference is implausible if the true average were equal.

```
var.test(formula = offer ~ group,
       data = tickets)
```

```
##
##  F test to compare two variances
##
## data:  offer by group
## F = 0.1, num df = 32, denom df = 30, p-value = 6e-09
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.05 0.21
## sample estimates:
## ratio of variances
##                0.1
```

```
#Levene's test (same as SAS output)
car::leveneTest(offer ~ group,
       data = tickets)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value  Pr(>F)
## group  1    16.4 0.00014 ***
##       62
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
t.test(formula = offer ~ group,
       data = tickets,
       alternative = "less")
```

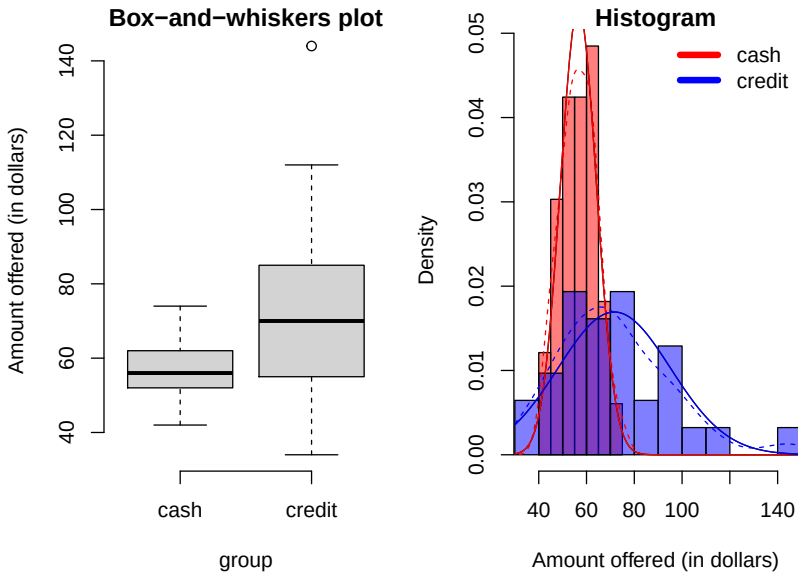
```
##
##  Welch Two Sample t-test
##
## data:  offer by group
## t = -3, df = 36, p-value = 0.0009
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##  -Inf -7.5
## sample estimates:
##   mean in group cash mean in group credit
##                   57                   72
```

It now remains to check the normality assumption graphically; SAS prints histograms with superimposed densities and box-and-whiskers plots. We can also add quantile-quantile plots; the basic function is `qqnorm` in R and `qqline` adds a line passing through the first and third quartile, but we use the `qqPlot` function from the `car` package instead, since the latter includes a grid in the background and simulated approximate 95% pointwise confidence intervals. The discreteness of the observations in the cash group is visible (corresponding to horizontal segments). Here, due to large enough sample sizes, we have no evidence against normality even if the variances are obviously different.

```
par(mfrow = c(1,2), mar = c(4,4,1,1)) #change margins,
# mfrow = c(1,2) gives two plots side by side
boxplot(offer ~ group, data = tickets,
        main = "Box-and-whiskers plot",
        ylab = "Amount offered (in dollars)",
        frame = FALSE)

# Histogram
# Attach dataset - variables (columns) are now visible
# avoids having tickets$group everywhere, now group
attach(tickets)
hist(x = offer[group == "cash"],
     breaks = 10,
     xlim = range(offer),
     freq = FALSE, # density scale
     xlab = "Amount offered (in dollars)",
     main = "Histogram",
     col = rgb(1, 0, 0, 0.5))
#Add the second group to the plot
```

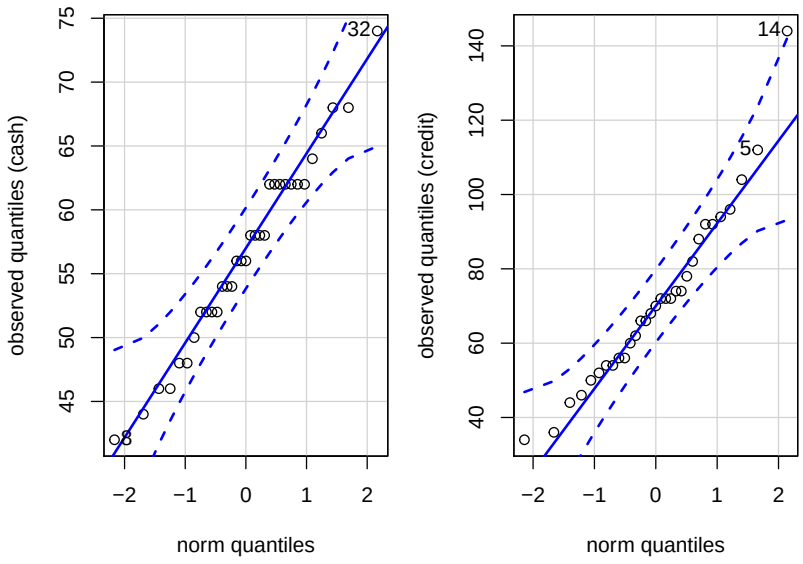
```
hist(x = offer[group == "credit"],
     breaks = 10,
     add = TRUE,
     freq = FALSE,
     col = rgb(0, 0, 1, 0.5))
legend(x = "topright",
       col = c("red", "blue"),
       legend = c("cash", "credit"),
       bty = "n",
       lwd = 5)
#superimpose density lines
lines(density(offer[group == "cash"]),
      lty = 2,
      col = "red")
lines(density(offer[group == "credit"]),
      lty = 2,
      col = "blue")
# Add normal density lines
lines(curve(dnorm(x,
            mean = mean(offer[group == "cash"]),
            sd = sd(offer[group == "cash"])),
            from = 30,
            to = 150,
            add = TRUE),
      col = "red")
lines(curve(dnorm(x,
            mean = mean(offer[group == "credit"]),
            sd = sd(offer[group == "credit"])),
            from = 30,
            to = 150,
            add = TRUE),
      col = "blue")
```




```
# Quantile-quantile plots
# install `car` package (only once)
# uncomment following line to install
# install.packages("car")
# default function is `qqnorm
car::qqPlot(offer[group == "cash"],
            ylab = "observed quantiles (cash)")
```

```
## [1] 32  8
```

```
car::qqPlot(offer[group == "credit"],
            ylab = "observed quantiles (credit)")
```




```
## [1] 14  5
```

```
# alternatively
# qqnorm(offer[group=="cash"], main = "cash")
# qqline(offer[group=="cash"])
# Detach dataset (don't forget)
detach(tickets)
```

2.3 Wilcoxon rank sum test

If the data are not normal, either because the distribution is skewed or heavy-tailed, or because the sample sizes are small enough that the asymptotic distribution of the Welch test or the two-sample t-test is unreliable, we may resort to non-parametric procedures. The Wilcoxon rank-sum test is a test for a shift in distribution; the drawback is that it requires the distribution of the samples to be the same, up to a change in location. If the mean is finite, then this amounts to a change in mean under the assumption that $F_1(x-\Delta) = F_0(x)$ for any x for F_0, F_1 the distribution function in group 0 and 1, respectively.

The syntax for the test is analogous to that of the other using the formula `y ~ x`, namely `wilcox.test(offer ~ groupe, data = tickets)`. You can also specify the values for the two samples with the arguments `x` and `y`. The Wilcoxon rank sum test works with ranks, which is the relative position of the observations in the pooled sample. Intuitively, if there is no difference and both samples come from the same distribution $F_0(x)$, the sum of the ranks in either group (relative to the number of observations in that group) should not be systematically too large or too small. Because ranks are bounded by n, the test statistic is less sensitive to outliers or extremes, even if its power is not as great in small samples, the loss of power is often not meaningful and the robustness is appealing, which is why Wilcoxon rank sum test is widely used in practice. Since ranks are discrete,

the distribution of the sum is tractable and we can use combinatorics to list all possibilities. For example, with eight observations split in two subgroups of equal size, the null distribution is

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

Probability (%)

1.4

1.4

2.9

4.3

7.1

7.1

10

10

11

10

10

7.1

7.1

4.3

2.9

1.4

1.4

Note that the value returned by R for the Wilcoxon rank sum test is not the same as the formula covered in class; it subtracts the minimum possible rank sum, here $m(m+1)/2$ for a first sample of size m so that the statistic is bounded below by zero, regardless of the sample size. As this is a shift by a deterministic constant, it has no impact on the asymptotic distribution.

In practice, there will often be ties (when two values are identical) and this means that the ranks are not uniquely defined (we could assign ranks at random for ties, or take the average, etc.); R will print a warning message to notify the user of the presence of ties. Formulae are adapted to deal with ties, but the software computes these automatically for us. While you will see multiple online references to the function `wilcox.test`, it does not return the correct p-value, point estimate or confidence interval as soon as there are ties — it is unfortunately is not doing the right thing. Outside of small samples, tabulating the null distribution is prohibitive if there are more than 50 observations in total, since the total number of possibilities explodes. In class, we saw that a normal approximation for the null distribution, i.e., the distribution of the rank sum test under the assumption that the two samples, of size n_1 and $n - n_1 = n_2$, respectively, both have distribution function F. The distribution can easily be approximated by simulation, by sampling uniformly integer values from 1 to n and summing the first n_1. We can repeat this a large number of times and get a good approximation. If there are ties, we proceed accordingly by resampling n_1 of the n ranks, without replacement. The normal approximation, whose formula was given in the slides, matches quite closely the empirical distribution of the simulated values; this is a consequence of the central limit theorem. Because of the normal approximation, we can do two-sided tests exactly as before by defining exteme values as deviation from the mean either side.

We illustrate this concept with the `tickets` data; bear in mind that the test is not valid here, because the data clearly do not come from the same distribution up to a shift (in particular, the variance are clearly not the same). We would normally need to test formally for equality of distribution up to a location shift, but this requires yet another test; most of the time, we will assess this hypothesis graphically.

```
n <- nrow(tickets)
# correct p-value, wrong confint and pe
exactRankTests::wilcox.exact(offer ~ group, data = tickets, conf.int = TRUE)
```

```
##
##  Exact Wilcoxon rank sum test
##
## data:  offer by group
## W = 288, p-value = 0.002
```

```
## alternative hypothesis: true mu is not equal to 0
## 95 percent confidence interval:
##  -22  -4
## sample estimates:
## difference in location
##                    -13
```

```
# wrong p-value b/c of ties, correct confint and pe
wilcox.test(offer ~ group, data = tickets, conf.int = TRUE)
```

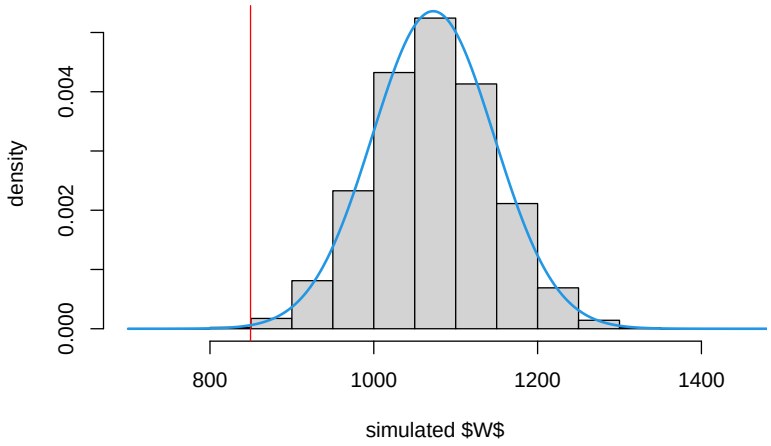
```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  offer by group
## W = 288, p-value = 0.003
## alternative hypothesis: true location shift is not equal to 0
## 95 percent confidence interval:
##  -20  -4
## sample estimates:
## difference in location
##                    -12
```

```
# Test statistic is sum of ranks
Wteststat <- sum(rank(tickets$offer,
                      ties.method = "average")[tickets$group == "cash"])

# Compute the p-value through simulation
prank <- rank(tickets$offer,
              ties.method = "average")
nrep <- 1e6L
W <- rep(0, nrep)
for(i in 1:nrep){
   #resample the ranks
   x <- sample(prank, size = 33, replace = FALSE)
   W[i] <- sum(x)
}
n1 <- 33;
n2 <- 31

hist(W, freq = FALSE,
     ylab = "density",
     xlab = "simulated $W$",
     main = "") #remove title
abline(v = Wteststat, col = "red")# add line for test stat
```

```
# Warning: this normal approximation is for
# data w/o ties, i.e., the formula presented
# in the course slides
mues <- n1*(n1+n2+1)/2 #mean of normal approx
sdes <- sqrt(n1*n2*(n1+n2+1)/12) #var of normal approx
#superimpose normal approx to empirical distribution
lines(seq(700, 1600, length = 1000L),
dnorm(x <- seq(700, 1600, length = 1000L),
      mean = mues, sd = sdes), lwd = 2, col = 4)
```




```
# Simulated one-sided p-value
# by defn, proportion of samples as extreme (i.e. smaller)
mean(W < Wteststat)
```

```
## [1] 0.0011
```

```
#p-value from normal approx
pnorm(q = Wteststat, mean = mues, sd = sdes)
```

```
## [1] 0.0014
```

```
# One sided confidence interval based on normal approx.
mues-qnorm(0.95)*sdes
```

```
## [1] 950
```

The value returned for the p-value by `wilcox.exact` would be roughly twice the one-sided p-value, in this case. The normal approximation is also convenient for obtaining confidence intervals, here $[950.06, \infty)$.

The `wilcox.test` function returns the so-called Hodges–Lehmann estimator (i.e., the median of all the pairs $Y_i^{(1)} - Y_j^{(2)}$ $(i = 1, \ldots, n_1;\ j = 1, \ldots, n_2)$. These point estimators, the associated confidence intervals and the exact coverage of the intervals (with additionally the subtleties surrounging from ties and zeros) are beyond the scope of the course, but Charles Geyer's course notes give additional information. The following function is adapted from his code and is distributed under CC BY-SA 3.0 licence. It can be used to compute the Hodges–Lehmann estimator with the associated confidence interval, while also returning the coverage (not exactly 95% because of the discreteness of the data).

```
source("https://lbelzile.bitbucket.io/MATH60619A/hl_wilcox.R")
hl <- hl.wilcox(tickets$offer[tickets$group == "credit"],
          tickets$offer[tickets$group == "cash"])
```

```
## Wilcoxon rank sum test
## Hodges-Lehmann estimator:  -12
## confidence interval:  -20 -4
## achieved confidence level: 0.95
```

Another potential nonparametric test is the Fligner–Policello test. Unlike Wilcoxon rank sum, it does not assume that the two distributions are equal under the null hypothesis, but require them to be symmetrical around their median (so the variance in each group could be different). SAS implements the test and uses the asymptotic normal approximation for p values and such like.

In R, the `NSM3` package provides an implementation of the Fligner–Policello test.

```
fptest <- NSM3::pFligPoli(tickets$offer[tickets$group == "credit"],
          tickets$offer[tickets$group == "cash"])
1-fptest$p.val
```

```
## [1] 0.0028
```

The estimated p-value for the one-sided test is 0.0017.

Chapter 3

Linear regression

In class, we conducted an analysis of the linear relation between buying intention for customers (`intention`) and the number of seconds the subject fixated a candy advertisement (`fixation`) using a simple linear regression. The following noes will show you how to do this using R. Normally, strings are cast to factor (unless you specify `stringsAsFactors = FALSE`), but the danger here is that some variables are encoded using integers (sex, revenue, level of education, marital status). It is okay if we keep binary variables, i.e., those encoded as 0-1, as is, but it is often better to cast them to factor to get more meaningful labels given the lack of obvious ordering. Note that response variable intention is bounded between $[2, 14]$ by construction, being the sum of two Likert scales ranging from 1 to 7. The normal approximation with a sample size this large probably means this won't impact the inference, but we must be careful with predictions (by truncating them lie within the range of possible values).

```
url <- "https://lbelzile.bitbucket.io/MATH60619A/intention.txt"
intention <- read.table(url, header = TRUE)
# number of observations (rows) and variables (cols)
dim(intention)
```

```
## [1] 120  10
```

```
summary(intention)
```

```
##     fixation      emotion          sex            age        revenue
##  Min.   :0.0   Min.   :0.05   Min.   :0.00   Min.   :19   Min.   :1.00
##  1st Qu.:0.8   1st Qu.:0.72   1st Qu.:0.00   1st Qu.:27   1st Qu.:1.00
##  Median :1.3   Median :0.93   Median :1.00   Median :30   Median :2.00
##  Mean   :1.6   Mean   :1.04   Mean   :0.52   Mean   :30   Mean   :2.07
##  3rd Qu.:2.1   3rd Qu.:1.38   3rd Qu.:1.00   3rd Qu.:33   3rd Qu.:3.00
##  Max.   :5.8   Max.   :2.80   Max.   :1.00   Max.   :45   Max.   :3.00
##       educ          marital       intention         buy           nitem
##  Min.   :1.00   Min.   :0.00   Min.   : 2.0   Min.   :0.00   Min.   : 0.0
```

```
##  1st Qu.:1.75   1st Qu.:0.00   1st Qu.: 6.0   1st Qu.:0.00   1st Qu.: 0.0
##  Median :2.00   Median :1.00   Median : 8.0   Median :0.00   Median : 0.0
##  Mean   :2.04   Mean   :0.54   Mean   : 8.3   Mean   :0.45   Mean   : 1.7
##  3rd Qu.:3.00   3rd Qu.:1.00   3rd Qu.:11.0   3rd Qu.:1.00   3rd Qu.: 2.0
##  Max.   :3.00   Max.   :1.00   Max.   :14.0   Max.   :1.00   Max.   :10.0
```

```
#variable names
colnames(intention)
```

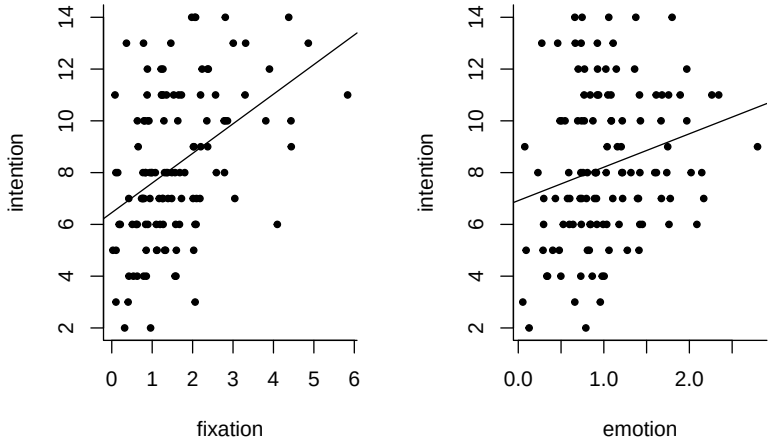
```
##  [1] "fixation"  "emotion"   "sex"       "age"       "revenue"   "educ"
##  [7] "marital"   "intention" "buy"       "nitem"
```

```
# cast categorical and ordinal variables to factors
intention$revenue <- factor(intention$revenue,
                            labels = c('low','medium','high'),
                             ordered = TRUE)
intention$educ <- factor(intention$educ,
                            labels = c(' <high school','high school','university'))
intention$marital <- factor(intention$marital,
                            labels = c("single","relationship"))
intention$sex <- factor(intention$sex,
                           labels = c("man","woman"))
```

By default, the baseline level for a factor is based on the alphabetical order, while SAS uses the first value it encounters (?). Once the variables are cast to factor, `summary` will print the counts for each respective categories; these could be likewise be obtained using `table`.

Before fitting a simple linear regression model, we should assess whether there is a linear relationship between the response and the explanatory (or not); a scatterplot can help to assess this. We superimpose the least square fit to the plot. In both cases, there is some positive correlation between intention and the explanatories, but this correlation is weak.

```
par(mfrow=c(1,2), pch = 20, bty = 'l')
plot(intention ~ fixation, data = intention)
abline(lm(intention ~ fixation, data = intention))
plot(intention ~ emotion, data = intention)
abline(lm(intention ~ emotion, data = intention))
```



We can compute the correlation coefficient using `cor`; we get 0.43 for `intention` and `fixation` and 0.23 for `intention` and `emotion`. If we specify a matrix with more than two columns or a data frame (with numeric values), we get a correlation matrix with ones on the diagonal. For factor variables, we would need to map these back to binary, but we saw that the R^2 value is equal to the squared correlation coefficient, so we can extract this from the output of the linear model fit.

3.1 The `lm` function

The function `lm` is the workshorse for fitting linear models. It takes as input a formula: suppose you have a data frame containing columns `x` (a regressor) and `y` (the regressand); you can then call `lm(y ~ x)` to fit the linear model $y = \beta_0 + \beta_1 x + \varepsilon$. The explanatory variable `y` is on the left hand side, while the right hand side should contain the predictors, separated by a `+` sign if there are more than one. If you provide the data frame name using `data`, then the shorthand `y ~ .` fits all the columns of the data frame (but `y`) as regressors.

To fit higher order polynomials or transformations, use the `I` function to tell R to interpret the input "as is". Thus, `lm(y~x+I(x^2))`, would fit a linear model with design matrix $(1_n, \mathrm{x}^\top, \mathrm{x}^2)^\top$. A constant is automatically included in the regression, but can be removed by writing `-1` or `+0` on the right hand side of the formula.

The `lm` function output will display ordinary least squares estimates along with standard errors, t values for the Wald test of the hypothesis $\mathrm{H}_0 : \beta_i = 0$ and the associated P-values. Other statistics and information about the sample size, the degrees of freedom, etc., are given at the bottom of the table.

Many methods allow you to extract specific objects. For example, the functions `coef`, `resid`, `fitted`, `model.matrix` will return $\widehat{\beta}$, e, $\widehat{y}$ and X, respectively.

```
linmod <- lm(intention ~ fixation, data = intention)
summary(linmod)
```

```
##
## Call:
## lm(formula = intention ~ fixation, data = intention)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.813 -1.828 -0.207  2.176  6.130
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    6.453      0.428   15.06  < 2e-16 ***
## fixation       1.144      0.224    5.12  1.2e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.7 on 118 degrees of freedom
## Multiple R-squared:  0.182,  Adjusted R-squared:  0.175
## F-statistic: 26.2 on 1 and 118 DF,  p-value: 1.21e-06
```

```
confint(linmod) #confidence intervals for model parameters
```

```
##             2.5 % 97.5 %
## (Intercept)   5.6    7.3
## fixation      0.7    1.6
```

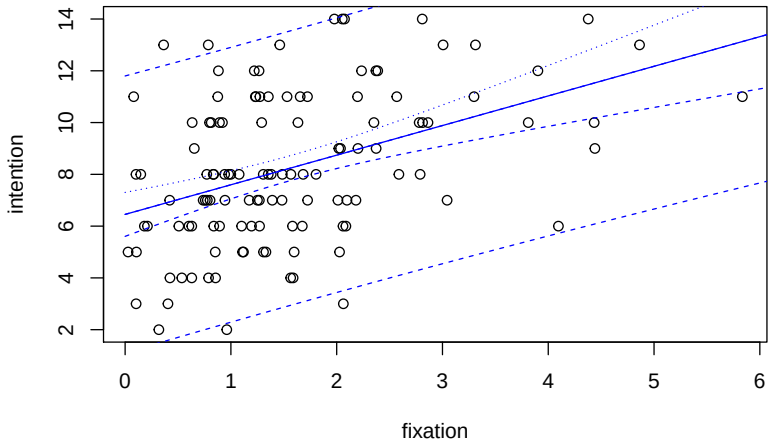
```
yhat <- fitted(linmod) #fitted values yhat
e <- resid(linmod) #ordinary residuals
jsr <- rstudent(linmod) #jackknife studentized resid.
```

The estimated intercept $\hat{\beta}_0$ is 6.92 and the estimated slope $\hat{\beta}_1$ is 1.29 The table gives the p-values for the null hypotheses $\beta_0 = 0$ and $\beta_1 = 0$, which are negligible. The estimated variance $\hat{\sigma}^2$, given by the `Residual standard error` is 2.86.

To obtain predicted values for combination of explanatories not present in the dataset, we need to first create a new data frame, with the same column names as those of the explanatory. In this case, the only X variable is `fixation`. We can get confidence intervals for the mean value with `interval = "conf"` or prediction intervals for the new observations with `interval = "pred"`.

```
newdata <- data.frame(fixation = seq(0, 6.5, length.out = 100L))
# Predictions with confidence interval for E(Y)
fitted_vals <- predict(object = linmod,
```

```
         newdata = newdata,
         interval = "conf")
#Same, but with prediction intervals for Y_new
predict_vals <- predict(object = linmod,
         newdata = newdata,
         interval = "pred")
plot(intention ~ fixation, data = intention)
matplot(x = newdata$fixation,
        y = fitted_vals,
        col = "blue",
        add = TRUE,
        type = "l")
matplot(x = newdata$fixation,
        y = predict_vals,
        lty = 2, #dashed lines,
        col = "blue",
        add = TRUE,
        type = "l")
```



The confidence interval for the mean value $\mathsf{E}(\widehat{Y}_i)$ is larger for values of X that are far from $\overline{\mathrm{X}}$; since there are less neighboring points, the model is extrapolating at this point and this translates in higher uncertainty, roughly speaking. The prediction intervals are much wider and account for the additional variability of ε_{new}; the intervals are pointwise, so for any given X, we would assume that 95% of the intervals in repeated samples would capture Y_{new}.

3.1.1 Graphical diagnostics and analysis of residuals

The hypothesis underlying the classical linear model are

1. independence
2. linearity (correct specification of the mean response)
3. homoscedasticity
4. normality

Underlying the linear model

$$Y_i = \beta_0 + \sum_{j=1}^{p} \mathrm{X}_{ij} + \varepsilon_i.$$

is the hypothesis that the errors $\{\varepsilon_i\}_{i=1}^n$ are independent and identically distributed $\mathcal{N}(0, \sigma^2)$ variables. We get to observe $y_i (i = 1, \dots, n)$, but $\beta_0, \beta_1, \dots \beta_p, \sigma^2, \varepsilon_i (i = 1, \dots, n)$ are all unknown. The best we can hope for is least square estimates for β's, say $\hat{\beta}_0, \dots, \hat{\beta}_p$. The fitted values are $\hat{y}_i = \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j \mathrm{X}_{i,j}$ from which we will deduce the ordinary residuals $e_i = Y_i - \widehat{Y_i}$.

Unfortunately, it turns out that e_i's are not independent; worst, they each have a different variance. However, the ordinary residuals and the fitted values are orthogonal, i.e. their linear correlation is zero. This is also true of X_j and e, and $\overline{e} = 0$ by construction. Thus, if we fit a linear model with mean structure $\mathsf{E}(e_i) = \beta_0 + \beta_1 \mathrm{X}_{ij}$ or $\mathsf{E}(e_i) = \beta_0 + \beta_1 \hat{y}_i$, the estimates for $\hat{\beta}_0$ and $\hat{\beta}_1$ will be exactly zero in both cases. We can verify this numerically.

```
#check whether coefs are zero
isTRUE(all.equal(coef(lm(e ~ yhat)), rep(0, 2), check.attributes = FALSE))
```

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

```
#due to machine precision, could be ~1e-17
coef(lm(e ~ fixation, data = intention))
```

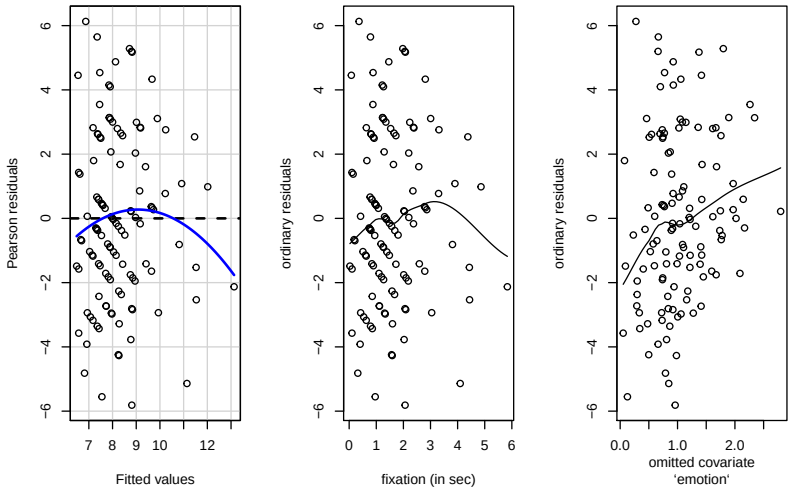
```
## (Intercept)    fixation
##    -6.3e-17     8.0e-18
```

```
mean(e)
```

```
## [1] -4.4e-17
```

If we plot e_i against $\hat{y}_i$ or X_i, we would expect to see no pattern. If we omitted important explanatory models for Y, these would not be accounted for in the mean model and their impact would be transferred to the residuals, which would be correlated with the omitted covariate. We can use local smoothing to check for residual nonlinear relationships or changepoints between e_i and $\hat{y}_i$ or e_i and X_i by using a local smoother (LOESS) that should capture these local effects, if present. Bear in mind that the model is not reliable.

```
par(mfrow = c(1, 3))
car::residualPlot(linmod)
scatter.smooth(e ~ intention$fixation,
     xlab = "fixation (in sec)",
     ylab = "ordinary residuals")
scatter.smooth(e ~ intention$emotion,
     xlab = "omitted covariate\n`emotion`",
     ylab = "ordinary residuals")
```



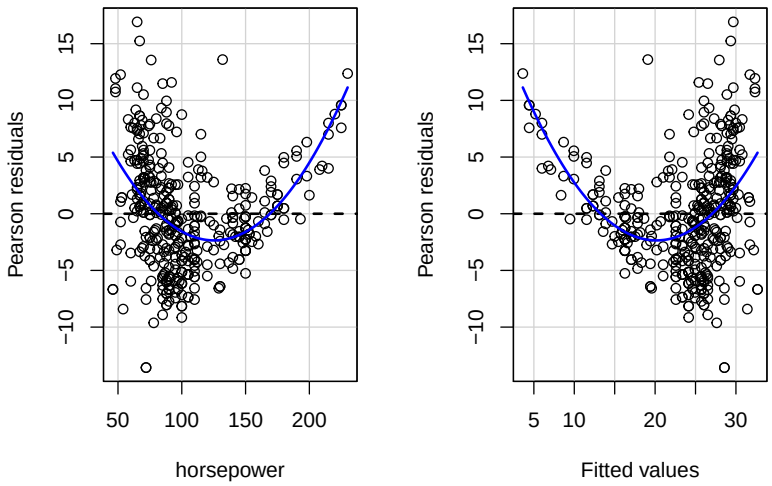
There is no graphical in our example that the relationship between fixation time and buying intention is nonlinear — the dip in fitted value and fixation is spurious and is due to the lack of long fixation time, which means the point has high leverage and pulls the smooth to itself.

If we plot the omitted covariate `emotion`, we see that the local trend is positive and possibly non-zero. This seems to imply that the effect of emotion has not been captured by the model, suggesting that our simple linear regression mean model is overly simple. The extension from simple to multiple linear regression models is straightforward and we can assess the significance of an added variable and test for their significance. In practice, we should use added-variable plots for new variables.

Contrast the lack of residual structure between `intention` and `fixation` with the relation between horsepower and fuel consumption in miles per gallon of the `auto` dataset: we can see a clear nonlinear (potentially quadratic) relationship between distance per gallon and fuel consumption.

```
url <- "https://lbelzile.bitbucket.io/MATH60619A/auto.csv"
auto <- read.csv(url, header = TRUE)
lmauto <- lm(mpg ~ horsepower, data = auto)
# residual plots (Pearson resid = ordinary resid)
```

```
car::residualPlots(lmauto, tests = FALSE)
```

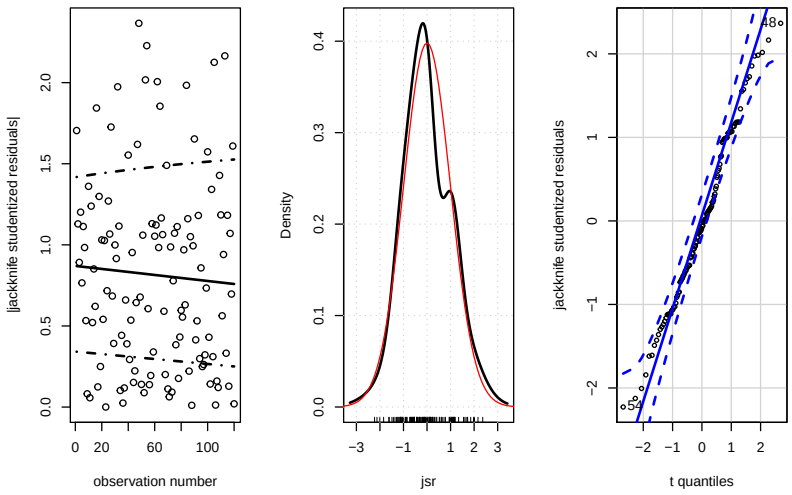


While `plot` method for `lm` objects return plots, the functions in the library `car` give nicer and often clearer output.

For heteroscedasticity, we can use the absolute jackknife studentized residuals rather than the ordinary residuals: the rationale is that the latter have different variance $\sigma^2(1-h_i)$, where h_i is a known constant that depends on X. The jackknife studentized residuals are thus standardized residuals, whereby each e_i is divided by an estimate of its standard deviation so that each has the same variance and we can make comparisons. The term jackknife comes from the estimation method. If the ε_i are truly normally distributed, then the jackknife studentized residuals should follow a Student with $n-k-1$ degrees of freedom, where k is the number of β parameters estimated. If $n-k$ is large, say larger than 25, we can use the normal distribution for comparison.

```
par(mfrow = c(1,3))
# check for heteroscedasticity with jsr
plot(1:length(jsr), abs(jsr),
     xlab="observation number",
     ylab="|jackknife studentized residuals|")
#a fancy smoother
car::gamLine(1:length(jsr), abs(jsr), spread = TRUE)
#density plot
car::densityPlot(jsr)
dfjsr <- linmod$df.residual - 1L
#superimpose density of student
curve(dt(x, df = dfjsr),
```

```
      from = -5,
      to = 5,
      add = TRUE,
      col = "red")
# student q-q plot of jackknife resid
car::qqPlot(jsr,
            distribution = "t",
            df = dfjsr,
            ylab = "jackknife studentized residuals")
```




```
## [1] 48 54
```

Here, we see no evidence of heteroscedasticity; the latter is more frequent in multiplicative models, when effects typically increase over time (like growth whose increase is, according to economic theory, exponential). The kernel density estimator (with rugs) and the quantile-quantile plots show that most points are in line with the postulated Student distribution, there is no outlier or extreme value and the residuals are symmetrically distributed around zero.

Interpretation of quantile-quantile plots requires experience; this post by Glen_b on StackOverflow nicely summarizes what can be detected (or not) from the Q-Q plot.

3.1.2 Binary explanatory variable

We have seen in class that the two-sample t-test is a special of linear regression and that we can use the same model. The default parametrization is the so-called contrasts, whereby the intercept β_0 correspond to the mean of the baseline, i.e., reference group and β_1 is the difference between the two groups. The summary table gives the t-test for the two-sided test $\beta_1 = 0$ with the alternative $\beta_1 \neq 0$.

For example, compare the output of the following two commands:

```
summary(lm(intention ~ sex, data = intention))
```

```
##
## Call:
## lm(formula = intention ~ sex, data = intention)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.919 -2.552  0.081  2.173  5.448
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    7.552      0.376   20.07   <2e-16 ***
## sexwoman       1.368      0.523    2.61     0.01 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.9 on 118 degrees of freedom
## Multiple R-squared:  0.0547, Adjusted R-squared:  0.0467
## F-statistic: 6.83 on 1 and 118 DF,  p-value: 0.0102
```

```
t.test(intention ~ sex, data = intention, var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  intention by sex
## t = -3, df = 118, p-value = 0.01
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.40 -0.33
## sample estimates:
##   mean in group man mean in group woman
##                 7.6                 8.9
```

It turns out that the value of the test statistic and the p-value for the two-sample t-test are the same as those of the linear regression for the coefficient β_1 corresponding to women. In fact, many tests we cover can be cast as linear models. The intercept parameter β_0 is the average of men, whereas $\beta_0 + \beta_1$ is the average of women. The advantage of doing a linear regression is that we can quantify the effect of sex while accounting for other potential explanatories.