Stochastic Lab Course II
– WiSe 2023/2024 –

Lecturer:
Tobias Kley

March 2024

Credits:
Earlier versions of these lecture notes were prepared by Tatyana Krivobokova. The first draft of the material on Git™ was prepared by Lukas Ungefug.

Contents

Chapter 1

Organisational Matters

1.1 Course Description

Key information

Time:	11/03/2024 – 22/03/2024, Mon–Fri daily 9.00 – 17.00
Submission Deadline:	Report is due on 30/04/2024
Location:	CIP room 4.101 at the IMS (Goldschmidtstr. 7)
Module:	M.Mat.0741: Advanced practical course in stochastics
Lecturers:	Tobias Kley
Intended Audience:	Advanced Bachelor and Master students in Mathematics

Prerequisites

It is assumed that participants have basic knowledge in stochastics, which is usually the case after attending at least one lecture in stochastics (probability theory and/or statistics). Previous familiarity with R is very helpful, but the *Stochastic Lab Course I* is not required. In the end of this section, resources to freshen up your skills are provided.

Daily Schedule

9^{00} – 11^{30}	Attend the lecture on the day's topic get started on the day's problem
11^{30} – 12^{30}	Question time with the lecturer
12^{30} – 13^{30}	Lunch break
13^{30} – 17^{00}	Work in groups on problems; help by the tutor available, on request

Description

This course is mostly learning by doing, with help available whenever needed. On each of the ten days, there is a short introductory lecture (approx. 1-1.5h) into an applied problem in stochastics. The problems cover diverse fields including applied probability and statistics. The lecture describes the problem, treats some basic theory and gives hints to get you started on the solution. Lecturer and tutor are available for general questions on problems and principles. In the afternoons, students work in groups and the tutor provides help on individual questions, on request. The students write a detailed report on their solutions.

Intended Learning Outcomes and Core Skills

In this course students deepen and expand their knowledge of a stochastical simulation and analysis software, and acquire advanced knowledge in project work in stochastics over the course of two weeks. In the course, students

- get to know a selection of ten, more complex stochastic problems;
- master some advanced theory, techniques, and methods of statistical data analysis and stochastical simulation to solve the problems;
- autonomously implement and interpret solutions to these problems using the widely popular software environment R;
- present their findings in form of a written report.

After having successfully completed the module, students will be able to

- handle practical problems with the aid of advanced stochastical methods and the suitable stochastical simulation and analysis software and present the obtained results well in form of a report;
- apply different algorithms to the suitable stochastical problem.

Assessment

1) In order to obtain credits for this course, students are required to submit a *detailed* lab report on *all* the problems. The written report is the basis for summative assessment.
The report should fully cover the following aspects:

- a description of the problem,
- a description of the methods used to solve the problem,
- a detailed discussion of the results,
- all graphics of interest, and
- the (well commented!) R code used to solve the problem (see Section 2.3).

Except for the R code, which will be submitted as a repository on GitLab, your report has to be a single PDF document (produced using LaTeX or R Markdown). The language of the report can be either English or German.

The deadline for the report submission is **30 April 2024** (no exceptions!).

2) Further, there will be a short oral examination (scheduled for **7 May and 8 May 2024**) where you will be asked questions about the material covered in the lectures (cf. Section 3 of this document).

Grading: the submitted report and oral examination will be graded separately. Both have to be passed (i. e., the grade for each has to be at least 4.0). The grade for the course is then obtained by averaging and rounded to the nearest grade (if the average is exactly between two grades the better will be chosen).

1.2 Getting started

(a) Download and install R on your computer: `https://ftp.gwdg.de/pub/misc/cran/`

(b) Download and install RStudio choosing the free RStudio Desktop version:
`https://rstudio.com/products/rstudio/download/`
This provides you with an integrated development environment (IDE) that will make your life with R a lot easier. It is not strictly required, but highly recommended.

(c) Download and install Git on your computer:
`https://git-scm.com/book/en/v2/Getting-Started-Installing-Git`

(d) Do the First-Time Git Setup:
`https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup`

(e) Check if you can log in to the GWDG GitLab: `https://gitlab.gwdg.de/`
Normally you should be able to log in with your student credentials which you use in eCampus/Stud.IP.

Each day, first attend the morning lectures where the day's topic will be introduced and then read the lecture notes and exercises. To work on the exercises also consider reading the help pages of the R functions mentioned after the exercises.

Chapter 2

Programming Basics

2.1 Why R?

First of all R is free. It is an open source project, which is similar to the S language environment which was developed at Bell Laboratories. Although there are some differences, much code written for S runs unaltered under R. Moreover R is growing at a rapid pace and there are many packages available covering a huge field of topics and methods.

2.2 References

There are numerous free resources for learning about R. Normally you should be fine by working through the material on Stud.IP, using the internal R-Help and searching the web for questions as you go along. In addition you should work together with your fellow students as much as possible and ask questions to the lecturers and the tutor, if necessary.
If you require a more comprehensive introduction to R, the following may be useful:

- An Introduction to R:
  `https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf`
  (the official document by the makers of R; maybe a bit dry)

- A number of other tutorials
  (satisfying various needs / levels of previous knowledge)
  `https://cran.r-project.org/other-docs.html`

- Introduction to R and RStudio:
  `https://www.youtube.com/watch?v=lL0s1coNtRk`
  (a comprehensive video about the very basics on R and RStudio, 91min)

- Hands-On Programming with R:
  `https://rstudio-education.github.io/hopr/`
  (a very gentle introduction; long, but quite entertaining)

Also, there will be an introduction to Git, which you will have to use to maintain your project. If you need more materials, the following may be useful:

- A website of the Git community, which contains a Reference Manual, a Book and a collection of other materials
  `https://git-scm.com/doc`

- Cheat Sheets from the GitHub Professional Services team
  `https://training.github.com/`

2.3 Programming Basics

The goal of this section is to provide you with some basic programming guidelines. In the heat of programming all the code you produce may seem perfectly understandable and clear to you. Anyway, if you don't follow some basic rules, you will run into trouble trying to understand your own code some time later. (There is a saying, that programs older than two weeks might be written by someone else...) In the special case of this practical course, you have to keep in mind, that your code will be corrected by a third person, who must be able to make sense of what you did.

Guidelines for writing code

- Keep it simple

  Think of the reader. Don't write just for yourself. Break down complexity into simpler chunks. Avoid implicit or obscure language features. Minimize scope, both logical and visual.

  Minimizing scope and breaking down complexity are not contradictory to each other. It may seem cool to do a very complex statement in one line, but if you ever tried to understand a program written by someone else, you surely have already cursed this particular programming style. When in doubt you should strive for clarity first, then efficiency.

- Keep it readable

  Use informative variable names. Good code can be read like a book. Make names clearly unique. Avoid abbreviations whenever possible. Name variables with noun or adjective noun combinations.

  It is quite tempting to use short names for variables and functions. This, however, is one of the main problems of many programs. Stories of programmers who used the name of their girlfriend as the name of a time variable might be known to you in the context of the year 2000 problem. In writing statistical programs people tend to name their variables 'x', 'xx', 'y0' and so on. Use more meaningful names. Especially abbreviations often seem totally clear at the moment but tend to lose their clarity very fast. 'predicted.value' is much more understandable than 'prdVl'. The usual way of separating two words in R is the use of a point like in 'linear.fit'.

- Comment your code

  Clearly comment necessary complexity. Be clear and concise. Say what is happening and why. Do not restate code. Keep code and comments visually separate.

  Comments are the most important part of a program, if you try to understand it later. Comment coherent units of code. Make it easy to look for a special functionality of your code. A good indicator whether you should comment or not is the amount of time you spend on producing the code. If you thought on some special lines of code for hours, they might be worth a short comment.

  Be aware, that comments may also decrease the readability of source code. They might even be misleading, if you fail to update them when changing your program. That is why you should make the code as clear as possible to reduce the need for comments.

  When using program packages like R it is sometimes very helpful to comment on the functions and the parameters of the functions you use. This is especially true for R since many names and parameters of functions do not meet the claim for clarity and intuitive comprehensibility.

Guidelines for commit messages in Git

Commit messages play an important role in a project, as they give you the means to understand what the purpose of each commit in the project history is. Therefore, it is useful to decide on a convention on how to formulate the commits, as a uniform structure makes it a lot easier to read them, in particular if you are collaborating with someone.
An example for a convention for this project could be a shortened version of Conventional Commits 1.0.0 (`https://www.conventionalcommits.org/en/v1.0.0/`):

⟨type⟩: ⟨description⟩

Here *type* stands for the type of the commit. For example:

- **fix** patches a bug in your code.
- **feat** adds a new feature.
- **refactor** is for changes in the code which neither fix a bug nor a feature (e.g. restructure to increase readability).
- **doc** adds documentation.
- **perf** improves the performance of the code (e.g. faster algorithms).
- **test** adds test cases.

The *description* is a short text which describes what the commit does with a consistent formatting (e.g. written in imperative mood, all lowercase and no point and no punctuation at the end). Some examples:

- feat: add function which creates a sample from the standard normal distribution
- refactor: improve readability of file.R
- perf: improve runtime by removing unnecessary loops

Chapter 3

Lecture Notes

3.1 Version Control with Git

This chapter is about Git, an open source version control system which allows users to track changes in files and to revert to previous versions of a project. Combining Git with a repository hosting service like GitHub or GitLab results to a useful tool for managing projects and collaborating in a team.
The goal of this chapter is to give an overview about the features and concepts of Git, which are needed to create and maintain a repository. Further details can be found in the Git documentation.

Terminology

This section contains a short explanation of the most important concepts of Git. The explanations are based on the 'Pro Git' book by Scott Chacon and Ben Straub.

- **Repository**: A collection of files with a revision history. Repositories are distinguished between local repositories, which are saved on your computer and remote repositories which are saved somewhere else (e.g. GitHub or GitLab).
- **Commit**: A specific point of the history, which contains the state of the files at that point and a reference to the previous commits (parents), if it is not the initial commit.
- **HEAD**: A pointer (reference) to the current branch which is worked on. The HEAD can also point to a commit (for example if you want to look at a specific one), then it is a detached HEAD, as it does not point to a branch.
- **Workspace**: Your project folder.
- **Staging area**: A collection of files whose changes you want to add to the next commit.
- **Committing**: Saving the changes of the files in the staging area as a new commit. The current commit is then a parent of the new commit, and the head is updated to point to the new commit.
- **Branch**: A pointer to a commit, the HEAD can point to a branch. When committing while the HEAD is pointing on a branch, the branch is pointing to the new commit. In most projects, the main branch is called `master`. Branches are useful, if you want to work on multiple independent features at the same time or if you have to add urgent changes while working on a feature which is not done yet.
- **Merging**: If you want to combine the progress of two branches, you have to merge them. For this, the HEAD has to point to a branch, and you have to choose a branch

which will merge into the current branch. As long as there are no conflicts (e.g. both commits are changing the same part of a file), a new commit is created, which has the other two commits as parents. Also, the branch, which the HEAD is pointing on, is then pointing on the new commit. In the case that a merge conflict arises, manual intervention is needed.

- **Pull and push**: In case you are working with a remote repository, you have to make sure, that your local repository has all the changes which are saved on the remote repository. To obtain the recent state of the remote repository, you have to pull from it. If you added commits to your local repository you have to push them to the remote repository to save them on it.
- **Stash**: Used to temporarily save changes. This is useful if you urgently need to fix something in another branch or if you added the changes to the wrong branch by accident and you want to move them to another one.

Important Commands

This section is a short list which contains the most important commands. For more details, consult the Git Reference, as most of the commands have optional parameters, which change the functionality of the command.
Note that some commands will change the workspace. This *can be dangerous* when modifications have not yet been committed. Anything that has been committed in Git can almost always be recovered.

Initializing the repository

- `git init`: Create a new git repository.
- `git clone` ⟨`URL`⟩: Copy the repository which is saved on the URL.

Commiting

- `git add` ⟨`file`⟩: Add file to the staging area.
- `git commit -m` ⟨`message`⟩: Create a commit.
- `git restore --staged` ⟨`file`⟩: Remove file from the staging area.
- `git restore` ⟨`file`⟩: Remove changes of an unstaged file, such that its state is equal to the state of the file in the most recent commit. (*Can be dangerous.*)
- `git status`: Show the current branch/commit, the unstaged changed files of your workspace and the staged files of your workspace.
- `git diff`: Show the differences between the unstaged, changed files of your workspace and the current commit.

Add a `.gitignore` to the repository to specify files that you would like to intentionally leave untracked and ignored by Git.

Branching and Merging

- `git branch` ⟨`name`⟩: Create branch at current commit.
- `git log --oneline --graph`: Show a graph of the history of the current branch.
- `git switch` ⟨`branch`⟩: Switch to the branch; your workspace has now the state of the chosen branch.
- `git checkout` ⟨`commit`⟩: Point the HEAD to the commit, your workspace has now the state of the chosen commit.
- `git merge` ⟨`branch`⟩: Merge the other branch into the current branch.

Undo and Redo
If changes are not yet pushed (note: commits are not deleted, only the head (and branch) point to a previous commit):

- `git reset --soft` ⟨`commit`⟩: 'Remove' the commits after the commit, changes will be put into the staging area, workspace stays the same.
- `git reset --mixed` ⟨`commit`⟩: 'Remove' the commits after the commit, workspace stays the same.
- `git reset --hard` ⟨`commit`⟩: 'Remove' the commits after the commit, workspace is changed to the chosen commit.
- `git reflog -n` N: Show a log of the latest N positions of the head.
- `git reset HEAD@{`N`}`: Point the head (and branch) to the position, where they pointed at N steps before.

If changes are already pushed:

- `git revert` ⟨`commit`⟩: Undo the commit, results in a commit, whose state is equal to the previous commit.
- `git revert -m 1` ⟨`merge commit`⟩: Undo the merged commit, results in a commit, whose status is equal to the previous commit, which was merged into.

Stash

- `git stash`: Put the unstaged changes on the staging area.
- `git stash pop`: Apply the changes in the stash to the current commit.

Push and Pull

- `git fetch`: Fetch and integrate updates from remote repository.
- `git pull`: Does the same as `git fetch`, additionally the current branch of the remote repository is merged into the current branch of your repository.
- `git push`: Push your changes of the current branch to the remote repository.

GitLab

GitLab is a service which can be used to host your repositories. By sharing the hosted repository with others, it is possible to collaborate in a project. GitLab also has many useful features, for example:

- Issues: Issues can be used to create tasks, report bugs, discuss features, etc. It is also possible to assign an issue to a person, the issue is then shown on your to-do list.
- Wiki: The wiki can be used to create a documentation for your project.
- Fork: Forks are copies of other repositories, which can be used to create an alternative version of a project.

3.2 Tidyverse

This chapter is about the **tidyverse**, a set of inter-compatible R packages that share a common philosophy. A list of the packages included in the **tidyverse** and resources to assist learning about it can be found on the "Packages" and "Learn" sections, respectively, of the website `https://www.tidyverse.org`.

The **tidyverse** has extensive functionality and it is not within the scope of one lecture to cover all of it. We therefore focus on the following three aspects:

- Building layered visualizations with **ggplot2**.
- Use of pipes `%>%` to improve clarity of your code.
- Data manipulations and transformations with **dplyr**.

Further topics and additional details can be found in Wickham and Grolemund (2017), which is also available online: `https://r4ds.had.co.nz`.

To load the **tidyverse** call

```
library(tidyverse)

## -- Attaching core tidyverse packages ----------------- tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ---------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

Take careful note of the conflicts message that is printed when you load the tidyverse. It tells you that **dplyr** overwrites some functions in base R. If you want to use the base version of these functions after loading **dplyr**, then you will have to use their full names: i. e. `stats::filter()` and `stats::lag()`.

3.2.1 Layered visualizations with ggplot2

We now discuss some examples of how to use **ggplot2** and comment on them. Many resources with additional information are available. For example, you may want to consider the following references:

- Section 3 in `https://r4ds.had.co.nz`,
- `https://ggplot2.tidyverse.org`,
- `https://r-graphics.org`,
- `https://ggplot2-book.org`.

The function `qplot` is a wrapper for `ggplot`. Its user interface is similar to that of the base function `plot` and can be called to produce some of the standard plots quickly. It is recommended, though, to learn `ggplot`, because this provides more flexibility in creating results. To this end, the function `qplot` has been depreciated in **ggplot2** since version 3.4.0.

Consider the following five calls of `qplot`:

```
qplot(x = displ, y = hwy, data = mpg)

## Warning:  `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning
## was generated.

qplot(x = displ, y = hwy, data = mpg, geom = "line")
qplot(x = displ, y = hwy, color = drv, data = mpg)
qplot(x = displ, data = mpg, binwidth = 0.25)
qplot(x = displ, data = mpg, geom = "density")
```

The first two calls provide the arguments `x` and `y`. Without the parameter `geom` set, the first call defaults to producing a point cloud. The second call, where the parameter `geom` is set to `"line"` produces a line plot. These two plots are shown in the first row of Figure 3.1.
The third call produces a point cloud as the first call, but with the points colored according to the values of the variable `drv`. This plot is shown in row two of Figure 3.1.
The fourth and fifth call, where only the argument `x` is set produces a histogram by default. In the fourth plot, we have set the parameter `binwidth` in response to a warning message that is received when using the default value. The fifth plot, where no parameter `y` is provided and the parameter `geom` is set to `"density"` shows a kernel density estimate. These two plots are shown in the third row of Figure 3.1.
Note that to create the plots in Figure 3.1, we have called `qplot` with the parameter `3/4` or `3/8` to achieve the non-square aspect ratios (not shown).

Next, we discuss how to obtain some of the plots in Figure 3.1 by regular calls to `ggplot` (i. e., without using the wrapper function `qplot`).
The following three calls illustrate how to add a layer to a `ggplot`:

```
ggplot(data = mpg)
ggplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy))
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy, color = drv))
```

The first call creates a coordinate system to which the data set `mpg` is attached in the background. Since no graphical layers have yet been added to it, the graph obtained from the first call is "empty". Therefore, in the second call, to obtain the scatterplot, we add a layer of points. A layer is added by the overloaded `+` operator with a `ggplot` object as the left argument and a geometric object as the right argument. geometric objects can be obtained by functions that follow a pattern such as

```
geom_LAYER(mapping = aes(MAPPINGS))
```

In the above examples' second call, we add a layer of points where the *aestetic mapping* indicates which variables to use as the x and y variables; `displ` is the engine displacement (in litres) and `hwy` is the highway miles per gallon. Further, in the third call, we also indicate which variable to use to obtain the color of the points. There are many other aestetics that can be mapped to variables. For examples consider the following calls:

```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy, alpha = cty))
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy, shape = drv))
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy, size = cyl), alpha = 0.2)
```

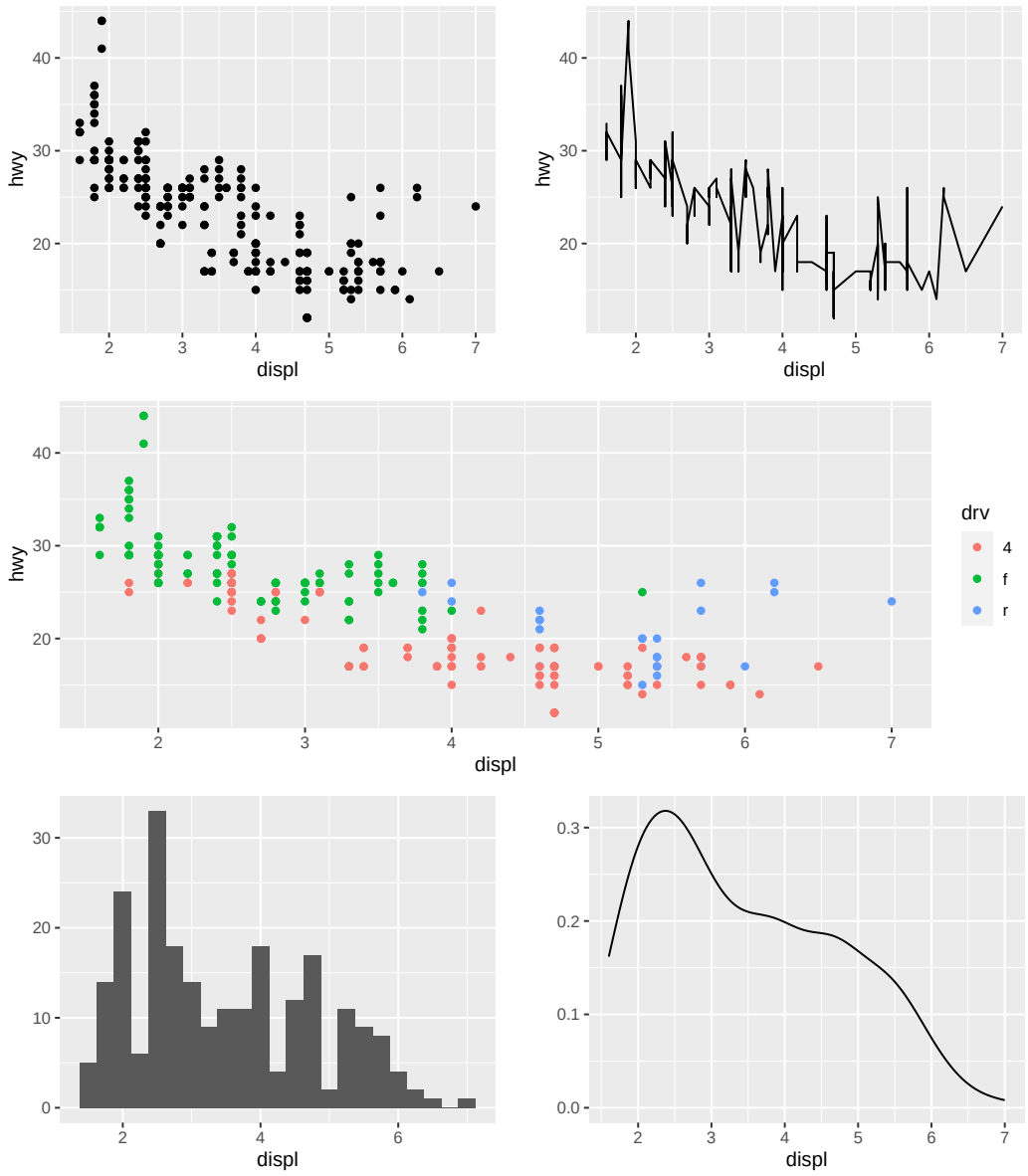

Figure 3.1: Plots produced with `qplot`.

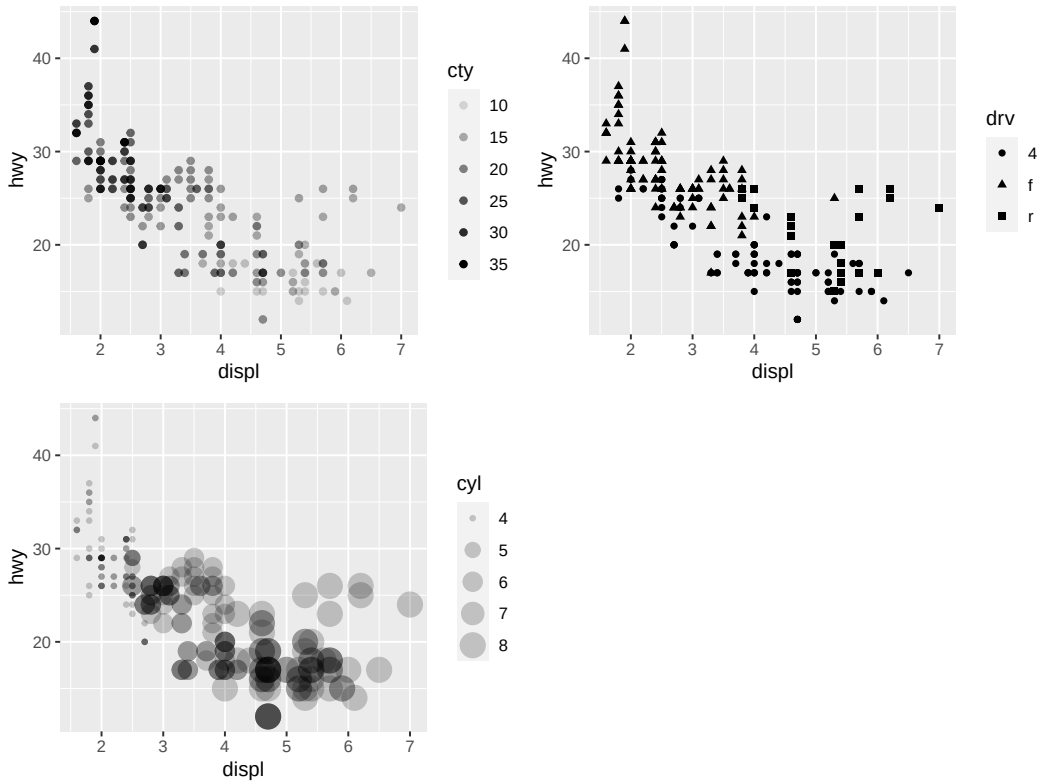

Figure 3.2: Scatterplots produced with `ggplot`.

In the first call we map the variable `cty` that encodes the city miles per gallon to the `alpha` aestetic that controls the transparency of the points. Note that using a discrete variable for an aestetic such as alpha (or size, below) is possible, but not advised and will therefore trigger a warning message. In the second call we map the `drv` variable that indicates whether a car has front-wheel drive, rear wheel drive, or 4 wheel drive to the `shape` aestetic. In the third call, we map the `cyl` variable (number of cylinders) to the `size` aestetic. Additionally, to allow for better visibilty of otherwise overlapping points, we set the `alpha` aestetic to a fixed value of `0.2`. Note that this is not part of the mapping and independent of the values that the variables take. For a list the aeastetics that can be used with point clouds see `?geom_point`. For a list of all aestetics in general see `vignette("ggplot2-specs")`. The three plots decribed above are shown in Figure 3.2.

Another way of using the variables available, instead of mapping them to an aesthetic within the scatter plot, is to separate the points into several panels. This is can be achieved by adding a `facet_wrap()` as illustrated in the following call:

```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy)) +
facet_wrap(~ drv, nrow = 1)
```

The resulting plot of three panels is shown in Figure 3.3. Noting that the first two lines are the same as the ones from which we obtained the first plot in Figure 3.1, another advantage of using layered plots can be observed: the plot is not rendered before all the layers have been added.

Another useful geometric object, `geom_smooth()` adds a smoothed conditional mean. The following calls illustrated how to use it:

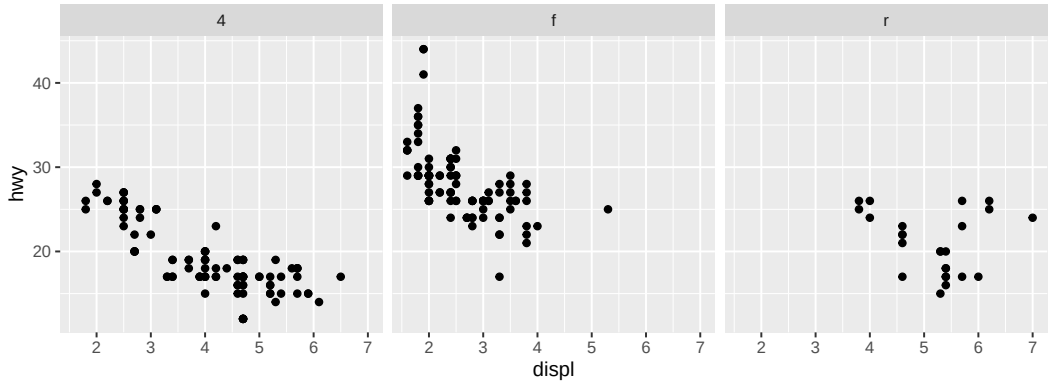

Figure 3.3: Scatterplots produced with `ggplot` and `facet_wrap()`.

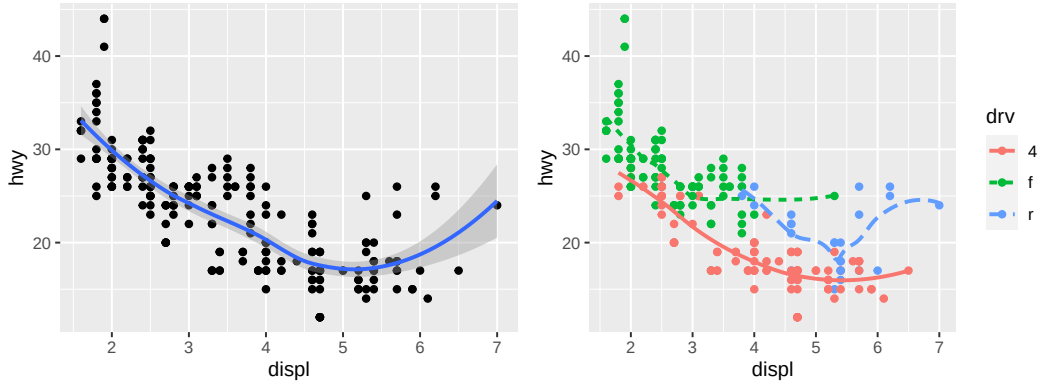

Figure 3.4: Scatterplots with smoothed conditional means, produced with `ggplot`.

```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy)) +
geom_smooth(mapping = aes(x = displ, y = hwy))
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy, color = drv)) +
geom_smooth(se = FALSE,
mapping = aes(x = displ, y = hwy, linetype = drv, color = drv))
```

In the first call we use, besides specifying the x and y variables, default values. Here, because there are less than 1000 values, the loess smoother is used. The second call illustrates that `geom_smooth()` is also compatible with mapping variables to aestetics. The output is shown in Figure 3.4.
Many other geometric objects are available. Information can be found in the help files under the entries of the form "`geom_*`".
Another geometric object that can, for example, be used to draw the outlines of maps is `geom_polygon()`. The following R code can be used to draw the map of New Zealand that is shown in Figure 3.5:

```
nz <- map_data("nz")
ggplot(nz, aes(x = long, y = lat, group = group)) +
geom_polygon(fill = "white", color = "black")
```

Call `?map_data` to obtain further information on the map data used.

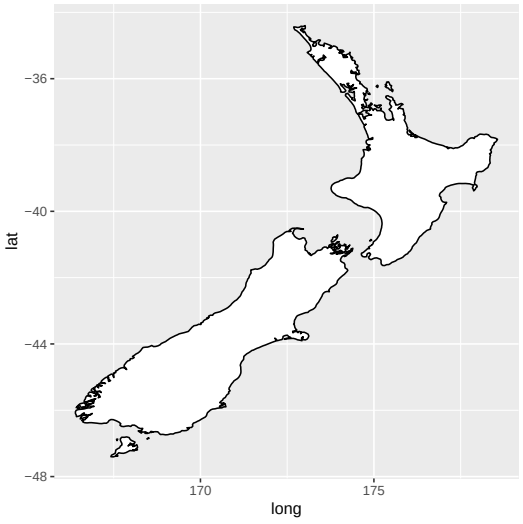

Figure 3.5: Map of New Zealand obtained with `ggplot` and `maps`.

Finally, before the other two topics (pipes and **dplr**) will be discussed, note that all the graphical objects, that were added to the plots, changed the plot only conceptually, but not visual details such as axis labels, fonts, or position of the plots elements. This can be achieved by adding a `theme()` to the plot, as is illustrated in the following R code:

```
## Changes axes, labels and add legends, themes
g <- ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy, color = drv))
g
## Sizes
g + theme(text = element_text(size = 20))
g + theme(axis.text = element_text(size = 25),
axis.title = element_text(size = 30),
legend.text = element_text(size = 20),
legend.title = element_text(size = 15))
## Labels
g2 <- g + theme(text = element_text(size = 20)) +
xlab("Engine size") +
ylab("Efficiency") +
ggtitle("Cars")
g2
## Legend (much more on position, background color is possible)
g2 + guides(color = guide_legend(title = "Drive")) +
theme(legend.position = c(0.8, 0.8))
## Changing xlim, ylim
g2 + xlim(c(0, 8)) +
ylim(c(0, 50)) +
theme(legend.position = c(0.8, 0.8))
## Change theme
g2 + theme_minimal()
g2 + theme_light()
```

In the above, we first assign the basic scatter plot to the variable `g`. It is shown in the top left corner of Figure 3.6.

Then, we change the size of the text in it to 20 and plot it again. We can also change the sizes of individual text elements differently and then plot it. The resulting plots are shown in panels (1,2) and (2,1) of Figure 3.6, respectively.
In panel (2,2) of Figure 3.6 we see the plot `g2`, generated by the R code above, where we have changed the text size and set x and y labels as well as the plot's title.
The remaining four plots are versions of `g2` where we have added further layers. The plot in panel (3,1) of Figure 3.6 has the legend's title and position changed. In the panel at position (3,2) the x and y limits have been changed and the position of the legend changed (but not the title). Finally, the two plots in the panels of the fourth row are versions of the plot with complete themes changed. See `?ggtheme` for information on what themes are available.
For further details, refer to `https://ggplot2.tidyverse.org/reference/#themes`.

3.2.2 Pipes

In this section the very basics of using pipes is explained. Further details can be found in Section 18 of Wickham and Grolemund (2017).
Pipes are used when a sequence of commands is to be applied to transform an object. Using pipes can improve the readability of the code, where "left to right" is more natural than "inside out" which comes naturally from nesting function calls. Further, fewer local variables need to be defined when using pipes and it becomes easier to insert steps into an already existing sequence of steps.
In R, the pipe operator is provided via the **magrittr** package. Packages from the **tidyverse** load the pipe operator automatically, so when using them **magrittr** needs not to be loaded explicitly.
For an example, consider the function

$$T_{\mu_0}(x_1, \ldots, x_n) := \frac{\sqrt{n}(\bar{x}_n - \mu_0)}{\sqrt{s_n}}, \quad \bar{x}_n := \frac{1}{n}\sum_{i=1}^{n} x_i, \quad s_n := \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x}_n)^2,$$

that computes the statistic to test $H_0 : \mu = \mu_0$. Implemented in R it could look as follows:

```
T <- function(x, mu0) {
return( sqrt(length(x)) * (mean(x) - mu0) / sqrt(sd(x)) )
}
```

Now, generating 10 independent standard normally distributed random variables, and applying it to this test with $\mu_0 = 2$ could look like this

```
set.seed(123)
T(rnorm(10), mu0 = 2)

## [1] -6.234335
```

Using the pipe operator we have

```
set.seed(123)
rnorm(10) %>% T(mu0 = 2)

## [1] -6.234335
```

To decide whether $H_0 : \mu = \mu_0 := 2$ should be rejected we also take absolute values and then compare the result with the $1 - \alpha/2$ quantile of the t_9-distribution. For $\alpha = 0.05$, for example, the following code yields that we have a significant difference:

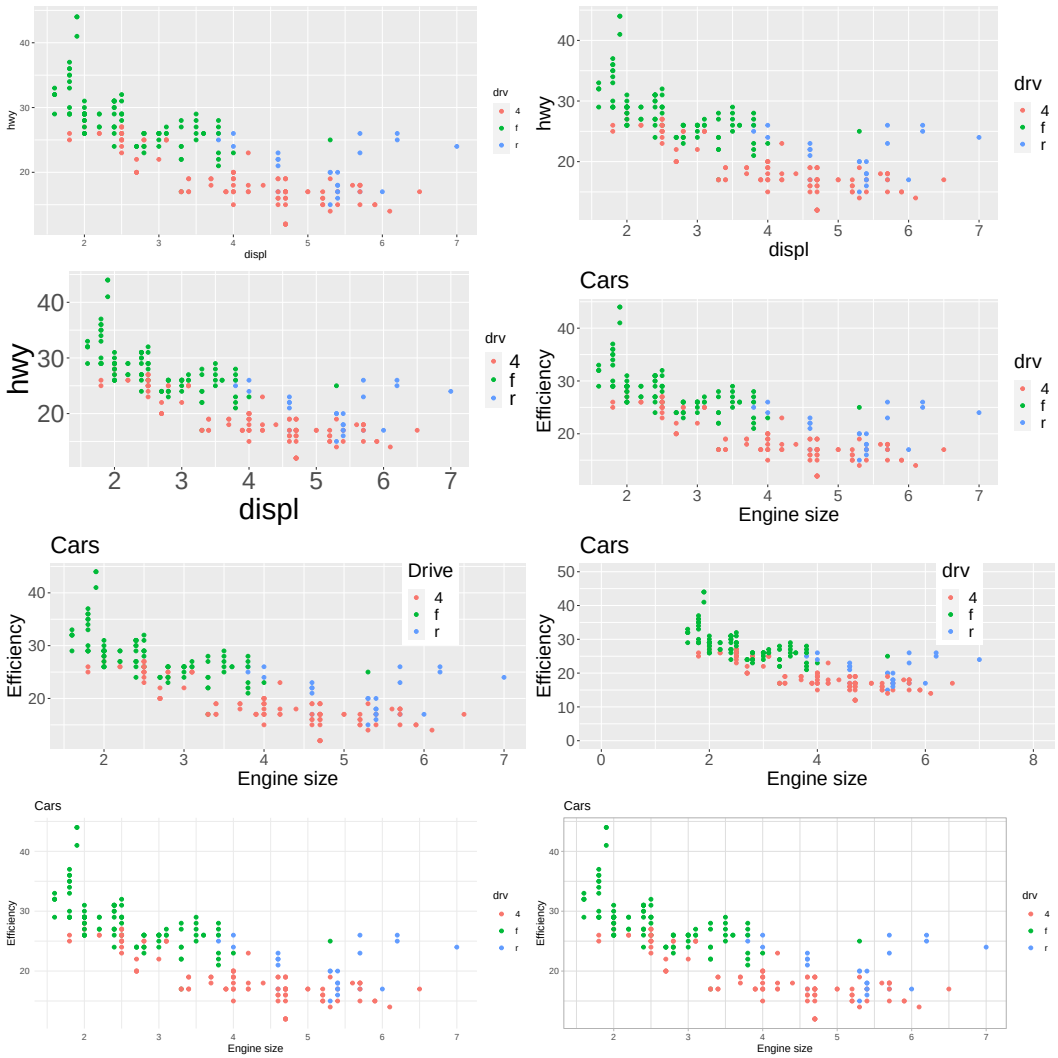

Figure 3.6: Illustration of using `theme()`s with `ggplot`.

```r
set.seed(123)
rnorm(10) %>% T(mu0 = 2) %>% abs

## [1] 6.234335

qt(0.975, df = 9)

## [1] 2.262157
```

In these toy examples, we obviously do not see the full advantage of using pipes. In the next section, where data transformations and manipulations within the tidyverse are discussed, the advantages will become clearer.
Some versions of the pipe operator exist that can be useful in certain situations. The assignment pipe `%<>%` can be used to forward the object on the left hand side and update the left hand side object with the resulting value. The tee pipe `%T>%` is useful when a function without a return value (e. g., `plot`) is part of the sequence of commands and the input to that function should also be piped forward to another function. The exposition pipe `%$%` exposes the names in the left hand side to the expression on the right hand side. Call `?magrittr` and read the documentation for further information.

3.2.3 Data Manipulations and Transformations

Now, consider the data set `flights` from the package **nycflights13**. The data about all flights that departed NYC in 2013 can be obtained by calling the following R code:

```r
library(nycflights13)
flights

## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # i 336,766 more rows
## # i 12 more variables: sched_arr_time <int>, arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>,
## #   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

We see that the data is saved in form of a tibble. A tibble is the tidyverse's version of a `data.frame`. Indeed they are data frames, but with their behaviour tweeked. You can find additional information in Section 10 of Wickham and Grolemund (2017) or in the vignette that you access by calling `vignette("tibble")`.
One difference that is obvious from the above is that a tibble also has information about the data types (`int` for integer, `dbl` for double, `chr` for character, `dttm` for data+tme, `lgl` for T/F, and `fctr` for factor).

We now discuss several commands from the **dplyr** package by giving examples. All of these commands have a verb as a name that, in combination with the pipe operator aims to obtain code that is clear, easy to read, write and maintain.

To begin with consider the `filter` command that can be used to filter rows in a data frame (or data frame extension, such as a tibble) according to expressions terms of variables from that data frame. For example, consider the following R code:

```
## filter = pick observations by their value
filter(flights, month == 1, day == 1)    # other: >, <, !=, ==, <=, >=
filter(flights, month == 11 | month==12) # logical: | (or), & (and), ! (not)
filter(flights, month %in% c(11, 12))    # same as previous
```

Instead of showing the output of these commands we briefly discuss what they do. The first command returns a data frame that contains only those rows from `flights` where the variable `month` equals 1 *and* the variable `day` equals 1, too. The second command returns the rows where `month` is 11 or 12. In the third command this condition is phrased equivalently using the `%in%` (is element of) operator.

Next, consider the `arrange` command that can be used to sort the rows according to some variables or functions of variables. By default, sorting is done in ascending order. To sort in descending order use `desc()`. To sort the `flights` data by date in ascending order or by arrival delay in descending order, call one of the following:

```
## reorder rows
arrange(flights, year, month, day)
arrange(flights, desc(arr_delay))
```

To subset variables (not rows), using a concise mini-language, use the `select` command. Besides of the three toy examples that follow, it is recommended to read `?select` to learn more about what can be expressed.

```
## select columns
select(flights, year, month, day)
#same as
select(flights, year:day)
select(flights, -(year:day))
```

To add new variables (while keeping existing one) use the `mutate` command. The command `transmute` works similarly, but drops existing variables. New variables overwrite existing variables of the same name. Variables can be removed by setting their value to NULL. Consider the following example, where also the use of the pipe operator is illustrated.

```
## add new variable
flights %>%
select(year:day, ends_with("delay"), distance, air_time) %>%
mutate(gain = arr_delay - dep_delay,
speed = distance / air_time * 60)
```

First we subset the variables endcoding the date, variables ending with the "delay" (e. g., `arr_delay`), the distance and air time. The result of the `select` command is then forwarded to the `mutate` command, where we add the variables `gain` and `speed` with definitions in terms of other variables (see the code example).
Another useful command is `summarize` that creates a new data frame with user specified variables containing the summary. This command is especially useful in combination with the `group_by` command that can be used to define groups by variables.

```
## summarize
flights %>%
summarize(delay = mean(dep_delay, na.rm = TRUE))
flights %>%
group_by(month) %>%
summarize(delay = mean(dep_delay, na.rm = TRUE))
```

As expected, the first call computes the mean of `dep_delay` for the whole data set, while the second call computes the means of the subsets where `month` takes one of the values 1 to 12.

We provide a second example, where we perform the following steps

- group flights by destination, then
- use `summarize` to compute (by group / destination)
  - the number of flights,
  - the average distances and
  - the average delays.
- Then we drop destinations with 20 flights or less, and flights to "Honolulu Intl", because this is about twice as far away from NYC than any of the other destinations.
- Finally, we use `ggplot` to obtain a scatterplot where we use the number of plots for different dot sizes, that are 1/3 transparent and a loess smoother.

The R code required to perform these steps is the following:

```
flights %>%
group_by(dest) %>%
summarize(count = n(),
dist = mean(distance, na.rm = TRUE),
delay = mean(arr_delay, na.rm = TRUE) ) %>%
filter(count > 20, dest != "HNL") %>%
ggplot(mapping = aes(x = dist, y = delay)) +
geom_point(aes(size = count), alpha = 1/3) +
geom_smooth(se = FALSE)
```

It provides a similar level of clarity and conciseness as the description in form of bullet points before. The resulting plot can be seen in Figure 3.7.

For the rest of this lecture, we are now discussing tibbles and how to work with them. Consider, for example, the `iris` data set, which comes as a `data.frame` object, as can be seen from

```
class(iris)

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

Use the function `as_tibble` to convert it and check its type:

```
iris_tl <- as_tibble(iris)
class(iris_tl)

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

To create a new tibble use the `tibble()` command, as seen in the following two examples

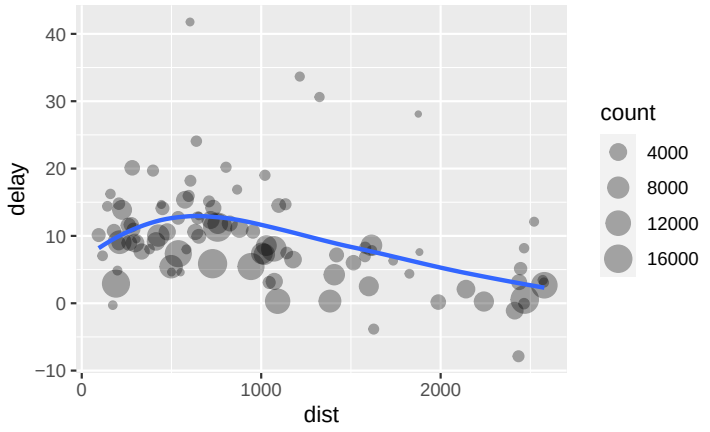

Figure 3.7: Scatterplot with smoother obtained from grouped and summarized flights data.

```
tibble(x = 1:5, y = 1, z = x^2 + y)
```

```
## # A tibble: 5 x 3
##       x     y     z
##   <int> <dbl> <dbl>
## 1     1     1     2
## 2     2     1     5
## 3     3     1    10
## 4     4     1    17
## 5     5     1    26
```

```
tibble(':)' = "smile", ' ' = "space",' 2000' = "number", no = 0)
```

```
## # A tibble: 1 x 4
##   `:)`  ` `   ` 2000`    no
##   <chr> <chr> <chr>   <dbl>
## 1 smile space number      0
```

You may want to familiarize yourself with this command by trying other possible data formats (dates, strings, etc) yourself.
By default 10 rows and not all columns (depends on the width of the screen) are printed. By calling the `print` command explicitly, you can customize this. For example, using the following code allows you to show more columns and only 5 rows:

```
flights %>% print(n = 5, width = Inf)
```

We are not showing the output here, because it makes sense to limit the number of columns, especially in a printed page.
To read data from a tab-delimited or comma-delimted file the functions `read_tsv()` and `read_csv()` are available. Alternatively, it may work better to load the data as a traditional data frame with `read.table()` and then use `as_tibble()` to convert it. To write a tibble to a file the functions `write_tsv()`, `write_csv()`, etc. are available. Call `?write_delim` to learn more.
Next, we discuss the concept of "tidy data" and give some examples from the **tidyr**. Tidy data refers to each variable having an own column, each observatios having its own row, and each value having its own cell. It is worth noting that **dplyr**, **ggplot2** are designed to work with tidy data.

In many data sets though, a variable can be spread across multiple columns or an observation can be scattered across multiple rows. To solve this, the functions `gather()`, `spread()`, `separate()`, and `pull()` are provided. Instead of the functions `gather()`, `spread()` and `separate()` the functions `pivot_longer()`, `pivot_wider()`, and `separate_wider_delim()` are now also available and users of the **tidyverse** are encouraged to use them instead. Regarding the first two groups of functions we note that `spread()` and `pivot_wider()` make long tables shorter and `gather()` and `pivot_longer()` make wide table narrower. Consider for example the example tibble `table4a` from **tidyr**:

```
table4a

## # A tibble: 3 x 3
##   country     `1999` `2000`
##   <chr>        <dbl>  <dbl>
## 1 Afghanistan    745   2666
## 2 Brazil       37737  80488
## 3 China       212258 213766
```

To make this a tidy tibble with country, year, cases we use `pivot_longer()`:

```
table4a %>% pivot_longer(cols = c(`1999`, `2000`),
names_to = "year", values_to = "cases")

## # A tibble: 6 x 3
##   country     year   cases
##   <chr>       <chr>  <dbl>
## 1 Afghanistan 1999     745
## 2 Afghanistan 2000    2666
## 3 Brazil      1999   37737
## 4 Brazil      2000   80488
## 5 China       1999  212258
## 6 China       2000  213766
```

To illustrate the use of `pivot_wider()`, which acts opposite to `pivot_longer()`, consider the example tibble `table2`, where each observation in country and year is spread across two rows:

```
table2

## # A tibble: 12 x 4
##    country      year type            count
##    <chr>       <dbl> <chr>           <dbl>
##  1 Afghanistan  1999 cases             745
##  2 Afghanistan  1999 population   19987071
##  3 Afghanistan  2000 cases            2666
##  4 Afghanistan  2000 population   20595360
##  5 Brazil       1999 cases           37737
##  6 Brazil       1999 population  172006362
##  7 Brazil       2000 cases           80488
##  8 Brazil       2000 population  174504898
##  9 China        1999 cases          212258
## 10 China        1999 population 1272915272
## 11 China        2000 cases          213766
## 12 China        2000 population 1280428583
```

We use `pivot_wider()` to transform it as follows:

```
#
table2 %>% pivot_wider(names_from = type, values_from = count)

## # A tibble: 6 x 4
##   country      year  cases population
##   <chr>       <dbl>  <dbl>      <dbl>
## 1 Afghanistan  1999    745   19987071
## 2 Afghanistan  2000   2666   20595360
## 3 Brazil       1999  37737  172006362
## 4 Brazil       2000  80488  174504898
## 5 China        1999 212258 1272915272
## 6 China        2000 213766 1280428583
```

Another commonly encountered issue is when the values for two variables are encoded in one cell. For example, in `table3` we see the number of cases and population of a year encoded as a rate that is formatted as "cases/population".

```
table3

## # A tibble: 6 x 3
##   country      year rate
##   <chr>       <dbl> <chr>
## 1 Afghanistan  1999 745/19987071
## 2 Afghanistan  2000 2666/20595360
## 3 Brazil       1999 37737/172006362
## 4 Brazil       2000 80488/174504898
## 5 China        1999 212258/1272915272
## 6 China        2000 213766/1280428583
```

The `separate()` command can be used to makes two variables from one, when a separator character is available:

```
table3 %>% separate(rate, into = c("cases", "population"),
sep = "/", convert = TRUE)

## # A tibble: 6 x 4
##   country      year  cases population
##   <chr>       <dbl>  <int>      <int>
## 1 Afghanistan  1999    745   19987071
## 2 Afghanistan  2000   2666   20595360
## 3 Brazil       1999  37737  172006362
## 4 Brazil       2000  80488  174504898
## 5 China        1999 212258 1272915272
## 6 China        2000 213766 1280428583
```

Note that we have specified `convert = TRUE`, as otherwise the variables cases and population would be encoded as characters.
In some situation we aim to achieve the opposite. Consider, for example, `table5` in which the year is encoded as century and year

```
table5

## # A tibble: 6 x 4
##   country     century year  rate
##   <chr>       <chr>   <chr> <chr>
```

```
## 1 Afghanistan 19      99    745/19987071
## 2 Afghanistan 20      00    2666/20595360
## 3 Brazil      19      99    37737/172006362
## 4 Brazil      20      00    80488/174504898
## 5 China       19      99    212258/1272915272
## 6 China       20      00    213766/1280428583
```

By calling `unite()` we can generate one variable, here `new`, by merging the existing ones.

```
table5 %>% unite(new, century, year, sep = "")

## # A tibble: 6 x 3
##   country     new   rate
##   <chr>       <chr> <chr>
## 1 Afghanistan 1999  745/19987071
## 2 Afghanistan 2000  2666/20595360
## 3 Brazil      1999  37737/172006362
## 4 Brazil      2000  80488/174504898
## 5 China       1999  212258/1272915272
## 6 China       2000  213766/1280428583
```

A function that can help to deal with missing values is the `complete()` command, as illustrated in the following example:

```
stocks <- tibble(year   = c(2015, 2015, 2015, 2015, 2016, 2016, 2016),
qtr    = c(1, 2, 3, 4, 2, 3, 4),
return = c(1.88, 0.59, 0.35, NA, 0.92, 0.17, 2.66))
stocks

## # A tibble: 7 x 3
##    year   qtr return
##   <dbl> <dbl>  <dbl>
## 1  2015     1   1.88
## 2  2015     2   0.59
## 3  2015     3   0.35
## 4  2015     4  NA
## 5  2016     2   0.92
## 6  2016     3   0.17
## 7  2016     4   2.66

stocks %>% complete(year, qtr)

## # A tibble: 8 x 3
##    year   qtr return
##   <dbl> <dbl>  <dbl>
## 1  2015     1   1.88
## 2  2015     2   0.59
## 3  2015     3   0.35
## 4  2015     4  NA
## 5  2016     1  NA
## 6  2016     2   0.92
## 7  2016     3   0.17
## 8  2016     4   2.66
```

Note that the return in Q4-2015 is explicitly missing (denoted by `NA`), while the return in

Q1-2016 is implicitly missing (not there at all). Calling `complete()` made the missing value explicit.
For an additional example see the `R` file for this topic that is also shown in the lecture.
Another important operation to deal with tibbles, or any kind of tables in fact, is to join them. Consider the following two tibbles for example:

```
base <- tibble(id = 1:3, age = seq(55, 60, length = 3))
base

## # A tibble: 3 x 2
##      id   age
##   <int> <dbl>
## 1     1  55
## 2     2  57.5
## 3     3  60

visits <- tibble(id      = c(rep(1:2, 2), 4),
visit   = c(rep(1:2, 2), 1),
outcome = rnorm(5))
visits

## # A tibble: 5 x 3
##      id visit outcome
##   <dbl> <dbl>   <dbl>
## 1     1     1   1.22
## 2     2     2   0.360
## 3     1     1   0.401
## 4     2     2   0.111
## 5     4     1  -0.556
```

The tibble `base` gives us the age to persons 1, 2 and 3. The tibble `visits` contains, in the first variable, a reference to the person who visited, as well as two additional variables concerning the visit. Note that the third person never visited, but that there is a 4th person, that we do not know about in the tibble `base` (i. e., we do not know the age).
Now, to join these two tibbles to one tibble containing all the information, we can perform an `inner_join()`. An `inner_join()` joins all the columns that have the same name. Note that column names are case sensitive so `id` and `ID` are not the same.

```
inner_join(base, visits)

## Joining with `by = join_by(id)`

## # A tibble: 4 x 4
##      id   age visit outcome
##   <dbl> <dbl> <dbl>   <dbl>
## 1     1  55       1   1.22
## 2     1  55       1   0.401
## 3     2  57.5     2   0.360
## 4     2  57.5     2   0.111
```

In some situations, it may be necessary to specify which columns are the keys to joining, which would looks similar to this

```
inner_join(base, visits, by = "id")
```

An alternative to the inner join is the `left_join()`. The resulting tibble will have everything what is in `base` and if something is missing in `visits` this will be indicated by `NA`.

```
left_join(base, visits)
```

```
## Joining with `by = join_by(id)`
```

```
## # A tibble: 5 x 4
##      id   age visit outcome
##   <dbl> <dbl> <dbl>   <dbl>
## 1     1  55       1   1.22
## 2     1  55       1   0.401
## 3     2  57.5     2   0.360
## 4     2  57.5     2   0.111
## 5     3  60      NA  NA
```

The other way a round, a `right_join()` will have everything from `visits` and where something is missing in `base` it will have `NA`. Note that this is the same as `left_join(visits, base)`, up to the order of the variables.

```
right_join(base,visits)
```

```
## Joining with `by = join_by(id)`
```

```
## # A tibble: 5 x 4
##      id   age visit outcome
##   <dbl> <dbl> <dbl>   <dbl>
## 1     1  55       1   1.22
## 2     1  55       1   0.401
## 3     2  57.5     2   0.360
## 4     2  57.5     2   0.111
## 5     4  NA       1  -0.556
```

Finally, we can also perform a `full_join()`, where the resulting tibble has everything from both input tibbles and `NA` if no matching id in the other one is found.

```
full_join(base, visits)
```

```
## Joining with `by = join_by(id)`
```

```
## # A tibble: 6 x 4
##      id   age visit outcome
##   <dbl> <dbl> <dbl>   <dbl>
## 1     1  55       1   1.22
## 2     1  55       1   0.401
## 3     2  57.5     2   0.360
## 4     2  57.5     2   0.111
## 5     3  60      NA  NA
## 6     4  NA       1  -0.556
```

3.3 Pseudo Random Number Generation

To generate a random variable from any distribution it is typically enough to be able to generate a uniform distributed random variable. How can one generate a sequence of numbers $u_1, u_2, \ldots$, such that they behave as independent realisations of $U \sim \mathcal{U}[0,1]$ (a random variable that is distributed uniformly over $[0,1]$)?

One of the first generators of uniform (pseudo) random variables are *linear congruent generators*. They were used e.g., in Visual Basic (up to 6) and Rand48 in Unix. Set $m > 0$ (modulus), $a > 0$ (multiplier), $r > 0$ (increment) and z_0 (seed, start value). Then, set

$$z_{n+1} = (a \cdot z_n + r) \bmod m, \quad n = 0, 1, 2, 3, \ldots$$

and define

$$u_n = \frac{z_n}{m}, \quad n = 0, 1, 2, \ldots.$$

Values $u_0, u_1, \ldots$ are in $\{0/m, 1/m, \ldots, (m-1)/m\} \subset [0, 1)$. Typically one chooses $m = 2^b$. As soon as a number from $\{0, \ldots, m-1\}$ appears for the second time, that is, as soon as $z_j = z_i$, $i < j$, one would get periodic numbers $z_i, \ldots, z_{j-1}$.

For example, set $m = 2^6 = 64$, $a = 4$, $r = 1$. Then, if $z_0 = 21$, then $z_1 = (4 \cdot 21 + 1) \bmod 64 = 85 \bmod 64 = 21 = z_0$, so that $z_0 = z_1 = \ldots = 21$ and the period is 1.

One can show that a linear congruent generator has a full period $m = 2^b$, $b \geq 2$ if and only if $r \in (0, m)$ is odd and $a \bmod 4 = 1$. However, even if m is large enough and the generator has a full period, it has a drawback to generate random variables that lay in hyperplanes in higher dimensions.

Another similar generator is a multiplicative congruent generator, which is defined for $m > 0$ (modulus), $a > 0$ (multiplier) and z_0 (seed) via $z_{n+1} = a \cdot z_n \bmod m$, $n = 0, 1, 2, \ldots$. This generator has similar drawback as a linear one.

In R, SAS, Matlab, Mathematica, Maple more modern generators are used. For example, the default in R is the Mersenne-Twister generator, see help to `Random` for details and other possible random variables generators.

Once one can generate a sequence of (pseudo) uniform random variables, there are several approaches to random variables generation from other distributions.

The first approach is the **inversion method**.
Let F be a distribution function on $\mathbb{R}$. Let the function $F^{-1} : (0, 1) \to \mathbb{R}$ be defined by

$$F^{-1}(u) = \inf\{x : F(x) \geq u\}, \ 0 < u < 1.$$

F^{-1} is called the quantile function. Note that if F is strictly increasing, then F^{-1} is just a regular inverse.

Theorem 1. *Let F be a distribution function. If $U \sim \mathcal{U}[0, 1]$, then $F^{-1}(U)$ has a distribution function F.*

Proof. It suffices to show that $\{U \leq F(x)\} \subseteq \{F^{-1}(U) \leq x\} \subseteq \{U \leq F(x)\}$, for all $x \in \mathbb{R}$. The assertion then follows from the monotonicity of P and the fact that $U \sim \mathcal{U}[0, 1]$ then implies that $P(F^{-1}(U) \leq x) = P(U \leq F(x)) = F(x)$ For the first inclusion, note that if $u \leq F(x)$, then $F^{-1}(u) = \inf\{t : F(t) \geq u\} \leq x$. For the second inclusion, assume that $x^* := F^{-1}(u) \leq x$. Then $u \leq F(x^*) \leq F(x)$, where the second inequality follows, because F is a cdf and thus non-decreasing. □

Several examples of continuous distributions:

(a) Exponential $F(x) = 1 - \exp(-\lambda x)$, $x > 0$, $F^{-1}(u) = \lambda^{-1} \log(1 - u)$, $u \in (0, 1)$

(b) Pareto $F(x) = 1 - (1 + x)^{-\alpha}$, $x > 0$, $F^{-1}(u) = (1 - u)^{-1/\alpha} - 1$, $u \in (0, 1)$

(c) Standard Cauchy $F(x) = 1/2 + \pi^{-1} \arctan(x)$, $F^{-1}(u) = \tan\{\pi(u - 1/2)\}$, $u \in (0, 1)$

Simulate U_i and set $X_i = F^{-1}(U_i)$. If F^{-1} can not be inverted analytically, appropriate numerical methods can be applied.

Let X be a discrete random variable with possible values $\{x_1 < x_2 < \ldots\}$, so that $F(x) = \sum_{i:x_i \le x} P(X = x_i)$ and

$$F^{-1}(u) = x_\ell, \quad \ell := \min \left\{ k \in \mathbb{N} : \sum_{j=1}^{k} P(X = x_j) = \sum_{j=1}^{k} p_j \ge u \right\}.$$

Then, the inversion method becomes: set $X = x_1$ if and only if $U_i \in [0, p_1)$ and $X = x_k$ if and only if $U_i \in \left[\sum_{j=1}^{k-1} p_j, \sum_{j=1}^{k} p_j\right)$, $k = 2, 3, \ldots$. Note that

$$P(X_i = x_k) = P\left(\sum_{j=1}^{k-1} p_j \le U_i < \sum_{j=1}^{k} p_j\right) = \sum_{j=1}^{k} p_j - \sum_{j=1}^{k-1} p_j = p_k.$$

For example, to simulate a Bernoulli random variable $Ber(p)$, generate $U \in U[0, 1]$ and set $X = 0$, if $U \le 1 - p$ and $X = 1$ if $U > 1 - p$.

Another general approach to random variables generation is the **rejection method** (Accept-Reject). Assume we would like to simulate random variables from a distribution with density f. The rejection method assumes the existence of a density g and the knowledge of a constant $c \ge 1$ (in practice we want to have c as close to 1 as possible), such that $f(x) \le cg(x)$, for all x. In contrast to f, we are able to generate random variables from a distribution with density g (e.g., with the inversion method).

Algorithm: Repeat the following steps:

(a) Generate a random variable X according to density g

(b) Generate a random variable $U \sim U[0, 1]$ (independent of X)

(c) If $Ucg(X) \le f(X)$, accept X, otherwise reject X.

Remarks

(a) $f(X)/\{cg(X)\}$ is a random variable independent of U.

(b) $f(X)/\{cg(X)\} \in (0, 1]$

(c) The number of iterations needed to successfully generate X is itself a random variable, which is geometrically distributed with the success probability $p = P(Ucg(X) \le f(X))$ (=acceptance probability), that is $P(N = n) = (1 - p)^{n-1}p$, $n = 1, 2, \ldots$. Hence, the expected number of iterations is $E(N) = 1/p$.

Calculating p we get

$$P\left(U \le \frac{f(X)}{cg(X)}\right) = \int_{-\infty}^{\infty} P\left(U \le \frac{f(X)}{cg(X)} \Big| X = x\right) g(x)\mathrm{d}x = \frac{1}{c}\int_{-\infty}^{\infty} f(x)\mathrm{d}x = \frac{1}{c}$$

Here we used that $P(U \le x) = x$. Hence, $E(N) = c$ and it makes sense to choose $c = \sup_x\{f(x)/g(x)\}$.

Theoretical justification of the accept-reject method is given in the following theorems.

Theorem 2. *Let $c > 0$ be an arbitrary constant, X be a random variable on $\mathbb{R}^d$, $d \geq 1$ with the c.d.f. $G(x)$ and p.d.f. $g(x)$ and $U \sim U[0,1]$, independent on X. Then, $(X, cUg(X)) \sim U[A]$, where $A = \{(x,u) : x \in \mathbb{R}^d, 0 \leq u \leq cg(x)\}$. Vice versa, If $(X,U) \sim U[A]$, then $X \sim G$.*

Proof. First note that since g is a density, the area under the graph of cg, that is, the Lebesgue measure of A equals to c. From this follows that the uniform distribution on A has the density $c^{-1}\mathbb{1}_{\{\cdot \in A\}}$. For the first statement, let B be a Borel set $B \subseteq A$ and denote $B_x = \{u : (x,u) \in B\}$. Since X and U are independent, by Tonelli's theorem,

$$P(\{X, cUg(X)\} \in B) = \int \int_{B_x} \frac{1}{cg(x)} \mathrm{d}u g(x) \mathrm{d}x = \frac{1}{c} \int_B \mathrm{d}u \mathrm{d}x.$$

Hence, $(X, cUg(X)) \sim U[A]$.
Now assume that $(X,U) \sim U[A]$. Then, the density of the marginal distribution of X is obtained by integrating out u

$$\int \frac{1}{c} \mathbb{1}_{\{(x,u) \in A)\}} \mathrm{d}u = \frac{1}{c} \int_0^{cg(x)} \mathrm{d}u = g(x).$$

Hence, $X \sim G$. □

Theorem 3. *Let $X_1, X_2, \ldots$ be a sequence of i.i.d. random variables on $\mathbb{R}^d$, $d \geq 1$ and let $A \subseteq \mathbb{R}^d$ be a Borel set such that $P(X_1 \in A) = p > 0$. Let Y be the first X_i taking values in A. Then*

$$P(Y \in B) = \frac{P(X_1 \in A \cap B)}{p}$$

for all Borel sets $B \subset \mathbb{R}^d$. In particular, if X_1 is uniformly distributed in A_0 for $A_0 \supseteq A$, then Y is uniformly distributed in A.

Proof. We have

$$\begin{aligned} P(Y \in B) &= \sum_{i=1}^{\infty} P(X_1 \notin A, \ldots, X_{i-1} \notin A, X_i \in B \cap A) \\ &= \sum_{i=1}^{\infty} (1-p)^{i-1} P(X_1 \in A \cap B) = \frac{1}{1-(1-p)} P(X_1 \in A \cap B). \end{aligned}$$

We have shown that

$$P(Y \in B) = \frac{P(X_1 \in A \cap B)}{P(X_1 \in A)}$$

Now, if X is uniformly distributed in some D, then by definition for any Borel set $C \supseteq D$

$$P(X \in C) = \frac{\int_{C \cap D} dx}{\int_D dx}.$$

Hence, since X_1 is uniformly distributed on A_0, we get

$$P(Y \in B) = \frac{P(X_1 \in A \cap B)}{P(X_1 \in A)} = \frac{\int_{A_0 \cap A \cap B} \mathrm{d}x}{\int_{A_0} \mathrm{d}x} \frac{\int_{A_0} \mathrm{d}x}{\int_{A \cap A_0} dx} = \frac{\int_{A \cap B} \mathrm{d}x}{\int_A \mathrm{d}x},$$

since $A \subseteq A_0$. That is, Y is uniformly distributed on A. □

From the first statement of Theorem 2 we have that $(X, Ucg(X))$ is uniformly distributed under the curve cg. By Theorem 3 we conclude that random variable $(X, Ucg(X))$ generated by the accept-reject algorithm (at the exit with X accepted) is uniformly distributed under the curve f. By the second statement of Theorem 2 we conclude that X must have density f.

3.4 Bootstrap

Let X be a real-valued random variable, that is a $(\mathcal{F}, \mathcal{B})$-measurable function from $(\Omega, \mathcal{F})$ to $(\mathbb{R}, \mathcal{B})$, where $(\Omega, \mathcal{F}, P)$ is the probability space associated with some random experiment. Consider n independent repetitions of this random experiment and denote $(x_1, \ldots, x_n)$ the resulting set of observations, called a **data set**. The corresponding random vector $\mathbf{X} = (X_1, \ldots, X_n) : (\Omega, \mathcal{F}) \to (\mathbb{R}^n, \mathcal{B}^n)$ is called a **sample of size n from a population with distribution** P. By construction, a sample $\mathbf{X}$ is a vector of n independent, identically distributed random variables, which will be denoted by $X_1, \ldots, X_n \overset{i.i.d}{\sim} P$. The corresponding distribution function will be denoted by F and p.d.f by f.

Any measurable function S of $\mathbf{X}$, $S(\mathbf{X})$ is called **statistic**, if $S(\mathbf{X})$ has a known value for known $\mathbf{X}$. A sample $\mathbf{X}$ is a trivial statistic. To make inference about $S(\mathbf{X})$ (confidence intervals, hypothesis tests) one has to know the distribution of $S(\mathbf{X})$. However, it is often impossible to derive the exact distribution of $S(\mathbf{X})$, either because S is complex or because P is unknown (even though in many cases an asymptotic distribution is available). In many cases **bootstrap** is an attractive way to approximate the distribution of $S(\mathbf{X})$.

A set of probability measures P_θ on $(\Omega, \mathcal{F})$ indexed by a parameter $\theta \in \Theta$ is said to be a **parametric family** if and only if $\Theta \subset \mathbb{R}^d$ for some fixed positive integer d and each P_θ is a known probability measure when θ is known. A **parametric statistical model** refers to the assumption that $\mathbf{X} = (X_1, \ldots, X_n)$ is a sample from the population with distribution $P_\theta \in \mathcal{P} = \{P_\theta : \theta \in \Theta \subset \mathbb{R}^d\}$ for a given parametric family. In the following our statistic $S(\mathbf{X})$ will be an estimator of a population characteristics θ (which might be a parameter of the distribution or e.g., median or moment).

If we knew P, we could sample many times from P to get many realisations of $S(\mathbf{X})$ and herewith the empirical distribution of $S(\mathbf{X})$. Recall that for random variables $Y_1, \ldots, Y_n$ the empirical distribution function is defined via $F_n(y) = n^{-1} \sum_{i=1}^n \mathbb{I}(Y_i \leq y)$. However, in practice P is unknown. The idea of the bootstrap is to sample from an empirical distribution function. Of course, there are many ways to estimate the distribution function and to sample from it. If an independent sample of size n is from a continuous distribution, then there are no ties, a. s., and each observation has a probability $1/n$ and sampling from F_n would be equivalent to draw with replacement from the sample.

Herewith is the bootstrap algorithm

(a) Draw n times with replacement from $\mathbf{X}$ to get a bootstrap sample $\mathbf{X}_1^*$ of size n. Repeat R times to get R bootstrap samples $\mathbf{X}_1^*, \ldots, \mathbf{X}_R^*$, each of size n.

(b) Compute bootstrap statistics $S(\mathbf{X}_1^*), \ldots, S(\mathbf{X}_R^*)$.

(c) Make inference about θ based on $S(\mathbf{X}_1^*), \ldots, S(\mathbf{X}_R^*)$.

How good are estimators (point or interval) based on the bootstrap sample? We consider bootstrap confidence intervals in detail. First recall the definition of a confidence interval.

Let $\mathbf{X} = (X_1, \ldots, X_n)$ be a sample from a population with distribution $P \in \mathcal{P} = \{P_\theta : \theta \in \Theta \subset \mathbb{R}^d\}$. Let $C(\mathbf{X})$ depend only on the sample $\mathbf{X}$ and $\theta \in \Theta$ be an unknown parameter of interest. If

$$\inf_{P \in \mathcal{P}} P(\theta \in C(\mathbf{X})) \geq 1 - \alpha$$

for a fixed constant $\alpha \in (0, 1)$, then $C(\mathbf{X})$ is called a **confidence set** for θ with **level of significance** $1 - \alpha$.

If the parameter θ is real-valued, then $C(\mathbf{X}) = \left[\underline{\theta}(\mathbf{X}), \bar{\theta}(\mathbf{X})\right]$, for a pair of real-valued statistics $\underline{\theta}$ and $\bar{\theta}$ is called a **confidence interval** for θ.

Example
Let $X_1, \ldots, X_n \overset{i.i.d.}{\sim} \mathbb{N}(\mu, \sigma^2)$ with unknown $\mu \in \mathbb{R}$ and known $\sigma^2 > 0$. Since $\overline{X} \sim \mathcal{N}(\mu, \sigma^2/n)$, we get

$$1 - \alpha = P\left(-z_{1-\alpha/2} \leq \frac{\overline{X} - \mu}{\sigma/\sqrt{n}} \leq z_{1-\alpha/2}\right) = P\left(\overline{X} - \frac{z_{1-\alpha/2}\sigma}{\sqrt{n}} \leq \mu \leq \overline{X} + \frac{z_{1-\alpha/2}\sigma}{\sqrt{n}}\right),$$

where z_α is the α-quantile of the standard normal distribution.
Hence, the confidence interval is given by $C(\mathbf{X}) = \left[\overline{X} - z_{1-\alpha/2}\sigma/\sqrt{n}, \overline{X} + z_{1-\alpha/2}\sigma/\sqrt{n}\right]$, which has length $2z_{1-\alpha/2}\sigma/\sqrt{n}$.

Note that since $\overline{X}$ is unbiased for μ and the transformation is linear, we can write this confidence band for μ as $[c_{\alpha/2}, c_{1-\alpha/2}]$, where c_α is the α-quantile of $\mathcal{N}(\overline{X}, \sigma^2/n)$, that is, $c_{\alpha/2} = \overline{X} - z_{1-\alpha/2}\sigma/\sqrt{n}$ and $c_{1-\alpha/2} = \overline{X} + z_{1-\alpha/2}\sigma/\sqrt{n},$.

Therefore, a natural way to construct the bootstrap confidence interval is to use empirical quantiles of the bootstrap distribution of $S(\mathbf{X})$: compute $\hat{\theta}_i^* = S(\mathbf{X}_i^*)$, $i = 1, \ldots, R$ bootstrap statistics and set the confidence interval for θ by $[\hat{\theta}_L^*, \hat{\theta}_U^*]$, where $\hat{\theta}_L^*$ and $\hat{\theta}_U^*$ are $\lfloor R\alpha/2 \rfloor$-th and $\lfloor R(1-\alpha/2) \rfloor$ value in the ordered list of $\hat{\theta}_i^*$. Such confidence intervals are called **bootstrap percentile** confidence intervals.

Let us consider more formally under which conditions such intervals work properly. Let $\hat{\theta}^* = S(\mathbf{X}^*)$ be a bootstrap estimator for θ and P^* denotes the distribution of $\mathbf{X}^*$ conditional on $\mathbf{X}$. Define $F_B(x) = P^*(\hat{\theta}^* \leq x)$. Let $\hat{\theta}_L^* = F_B^{-1}(\alpha/2)$ (above we used the same notation for the empirical version). Suppose there exists an increasing transformation ϕ_n such that

$$P\{\phi_n(\hat{\theta}) - \phi_n(\theta) \leq x\} = \Psi(x) \tag{3.1}$$

holds for all possible F (including empirical c.d.f), where $\Psi(x)$ is continuous, strictly increasing and symmetric about 0. Note that $\hat{\theta} = \hat{\theta}_n$ is a statistic (an estimator for θ) that depends on the sample of size n. When $\Psi = \Phi$ (c.d.f on $\mathcal{N}(0,1)$), then ϕ_n is called the normalizing and variance stabilizing transformation (standardisation as in the example with normal distribution is one possible ϕ). If ϕ_n and Ψ are known, then the lower confidence bound for θ has the form $\phi_n^{-1}\{\phi_n(\hat{\theta}) + \Psi^{-1}(\alpha/2)\}$. We show that this bound is the same as $\hat{\theta}_L^*$:

$$\Psi\{\phi_n(\hat{\theta}_L^*) - \phi_n(\hat{\theta})\} = P^*\left\{\phi_n(\hat{\theta}^*) - \phi_n(\hat{\theta}) \leq \phi_n(\hat{\theta}_L^*) - \phi_n(\hat{\theta})\right\} = P^*(\hat{\theta}^* \leq \hat{\theta}_L^*) = \alpha/2.$$

Hence, $\phi_n(\hat{\theta}_L^*) - \phi_n(\hat{\theta}) = \Psi^{-1}(\alpha/2)$ and $\hat{\theta}_L^* = \phi_n^{-1}\{\phi_n(\hat{\theta}) + \Psi^{-1}(\alpha/2)\}$.
Thus, the bootstrap percentile confidence intervals will have coverage probability of $1 - \alpha$ if assumption (3.1) holds exactly for all n. If (3.1) holds approximately for large n, then $\hat{\theta}_L^*$ is asymptotically correct and the confidence interval performance depends on how good the approximation is.

Efron (1981) considered a more general assumption

$$P\{\phi_n(\hat{\theta}) - \phi_n(\theta) + z_0 \leq x\} = \Psi(x), \tag{3.2}$$

where z_0 a constant that may depend on F and n. Since $\Psi(0) = 1/2$, z_0 is a kind of "bias" of $\phi_n(\hat{\theta})$. If ϕ_n, z_0 and Ψ are known, then the lower confidence bound for θ is $\phi_n^{-1}\{\phi_n(\hat{\theta}) + z_0 + \Psi^{-1}(\alpha/2)\}$. Under assumption (3.2), we obtain

$$F_B(\hat{\theta}) = P^*(\phi_n(\hat{\theta}^*) - \phi_n(\hat{\theta}) + z_0 \leq z_0) = \Psi(z_0),$$

so that $z_0 = \Psi^{-1}\{F_B(\hat{\theta})\}$. Also, from (3.2)

$$\begin{aligned} 1-\alpha/2 &= \Psi\{-\Psi^{-1}(\alpha/2)\} = P^*\left\{\phi_n(\hat{\theta}^*) - \phi_n(\hat{\theta}) + z_0 \leq -\Psi^{-1}(\alpha/2)\right\} \\ &= P^*\left(\hat{\theta}^* \leq \phi_n^{-1}\{\phi_n(\hat{\theta}) - \Psi^{-1}(\alpha/2) - z_0\}\right), \end{aligned}$$

which implies $\phi_n^{-1}\{\phi_n(\hat{\theta}) - \Psi^{-1}(\alpha/2) - z_0\} = F_B^{-1}(1-\alpha/2)$. Since this equation holds for any α, we get for any $x \in (0,1)$ that

$$F_B^{-1}(x) = \phi_n^{-1}\{\phi_n(\hat{\theta}) + \Psi^{-1}(x) - z_0\}.$$

Since the lower confidence bound is given by $\phi_n^{-1}\{\phi_n(\hat{\theta}) + z_0 + \Psi^{-1}(\alpha/2)\}$, using the last equation, allows us to rewrite the lower confidence bound as

$$F_B^{-1}\left[\Psi\{\Psi^{-1}(\alpha/2) + 2z_0)\}\right].$$

Assuming that Ψ is known (e.g., Φ) and using $z_0 = \Psi^{-1}\{F_B(\hat{\theta})\}$, Efron (1981) suggests the bias-corrected lower confidence bound for θ

$$\hat{\theta}^*_{LC} = F_B^{-1}\left(\Psi\left[\Psi^{-1}(\alpha/2) + 2\Psi^{-1}\{F_B(\hat{\theta})\}\right]\right).$$

Note that $\hat{\theta}^*_{LC}$ reduces to $\hat{\theta}^*_L$ if $F_B(\hat{\theta}) = 1/2$, i.e., $\hat{\theta}$ is the median of the bootstrap distribution F_B. Hence, $\hat{\theta}^*_{LC}$ is the bias corrected $\hat{\theta}^*_L$ and the bias correction is given by $2\Psi\{F_B(\hat{\theta})\}$. Again, if (3.2) holds exactly, then the corresponding bias corrected bootstrap confidence interval will have coverage probability $1-\alpha$. If (3.2) holds approximately, then the performance of the bias corrected bootstrap confidence interval will depend on how good the approximation is.

Since in practice we rather use the empirical version of F_B, to get the bias corrected bootstrap confidence intervals, set $\Psi = \Phi$ and calculate $\hat{z}_0 = \Phi^{-1}\left\{R^{-1}\sum_{i=1}^R \mathbb{I}(\hat{\theta}^*_i \leq \hat{\theta})\right\}$, then set $\alpha_1 = \Phi(z_{\alpha/2} + 2\hat{z}_0)$ and calculate $\hat{\theta}^*_{LC}$ as the $\lfloor R\alpha_1 \rfloor$ value in the ordered list of $\hat{\theta}^*_i$. Completely analogous, $\hat{\theta}^*_{UC}$ is the $\lfloor R\alpha_2 \rfloor$ value in the ordered list of $\hat{\theta}^*_i$, where $\alpha_2 = \Phi(z_{1-\alpha/2} + 2\hat{z}_0)$.

Bias corrected bootstrap confidence intervals improve bootstrap percentile confidence intervals by taking into account the bias. However, there are still many cases where assumption (3.2) is not fulfilled. Efron (1987) proposed a bootstrap accelerated bias-corrected confidence intervals, that further improves bias corrected bootstrap confidence intervals. One simple version of such confidence intervals is given by setting

$$\begin{aligned} \alpha_1 &= \Phi\left(\hat{z}_0 + \frac{\hat{z}_0 + z_{\alpha/2}}{1 - \hat{a}(\hat{z}_0 + z_{\alpha/2})}\right) \\ \alpha_2 &= \Phi\left(\hat{z}_0 + \frac{\hat{z}_0 + z_{1-\alpha/2}}{1 - \hat{a}(\hat{z}_0 + z_{1-\alpha/2})}\right), \end{aligned}$$

where

$$\hat{a} = \frac{\sum_{i=1}^n (\bar{\theta}_J - \hat{\theta}_{(i)})^3}{6\left\{\sum_{i=1}^n (\bar{\theta}_J - \hat{\theta}_{(i)})^2\right\}^{3/2}},$$

with $\bar{\theta}_J = n^{-1}\sum_{i=1}^n \hat{\theta}_{(i)}$, for $\hat{\theta}_{(i)}$ as the estimator of θ obtained without observation i, i.e., $\hat{\theta}_{(i)} = S(X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n)$.

3.5 Expectation Maximization

Assume that the random variable $X = (Y, Z)$ has distribution F_{θ_0} that belongs to a class of distributions $\mathcal{F} = \{F_\theta\}_{\theta\in\Theta}$, where $\Theta \subset \mathbb{R}^p$ is some parameter space. We intend to estimate θ_0 from *incomplete* data, i.e. we only observe independent realizations $y_1, \dots, y_n$ of Y and the observations $z_1, \dots, z_n$ for Z are latent (unobserved). The complete data would be $x_i = (y_i, z_i)$, $i = 1, ..., n$.
We shall further assume that each $F_\theta \in \mathcal{F}$ has a density $f_\theta(y, z)$ and that Y has a marginal density $g_\theta(y)$. Then, the conditional density of Z given Y takes the form

$$k_\theta(z|y) = \begin{cases} \frac{f_\theta(y,z)}{g_\theta(y)} & \text{if } g_\theta(y) \neq 0 \\ 0 & \text{else.} \end{cases}$$

To clarify the setting we study the following example.

Example 4 (Finite Mixture Distribution)**.** *Assume that Y is a finite mixture distribution with random weights Z, i.e. Y is a mixture of L distributions where $Z \in \{1, \dots, L\}$ encodes how Y is drawn. Assume that $\sum_{\ell=1}^L p_\ell = 1$. Then, (Y, Z) have the joint density*

$$f_\theta(y, z) = p_z \varphi_z(y, \delta_z),$$

where $\varphi_\ell(y, \delta_\ell)$ is the density of the ℓ-th distribution/cluster, depending on some parameter δ_ℓ. From this we find that $\theta = (p_1, \dots, p_L, \delta_1, \dots, \delta_L)$ and

$$g_\theta(y) = \sum_{\ell=1}^L p_\ell \varphi_\ell(y, \delta_\ell) \quad \text{and} \quad k_\theta(z|y) = \frac{p_z \varphi_z(y, \delta_z)}{\sum_{\ell=1}^L p_\ell \varphi_\ell(y, \delta_\ell)}$$

Given the observations $Y_1, ..., Y_n$ one ideally whishes to find a maximum likelihood estimator

$$\theta^* = \operatorname*{argmax}_{\theta\in\Theta} \sum_{i=1}^n \ln g_\theta(Y_i). \tag{3.3}$$

Since the marginal density g_θ usually is unknown, θ^* is not tractable. The *Expectation-Maximization (EM) algorithm* is an iterative method capable of computing θ^* from incomplete data $y_1, \dots, y_n$ using only the joint density $f_\theta(y, z)$. It was originally introduced in Dempster et al. (1977) and is defined as follows

(a) Choose an inital guess $\theta^{(0)} \in \Theta$.

(b) For $k = 0, 1, 2, \dots$ compute

$$Q(\theta, \theta^{(k)}) = \sum_{i=1}^n \mathbf{E}_{\theta^{(k)}} \left[\ln f_\theta(Y, Z)|Y = y_i\right] \qquad \text{E(xpectation)-step}$$

$$\theta^{(k+1)} = \operatorname*{argmax}_{\theta\in\Theta} Q(\theta, \theta^{(k)}) \qquad \text{M(aximization)-step}$$

The mapping $\theta \mapsto Q(\theta, \theta^{(k)})$ is the conditional expectation of the joint log-likelihood $\ln f_\theta(y, z)$ given the data $y_1, \dots, y_n$ under the hypothesis that $\theta^{(k)}$ is the true parameter. By maximizing $Q(\cdot|\theta^{(k)})$ over Θ one aims for replacing the parameter $\theta^{(k)}$ by an improvement $\theta^{(k+1)}$. If $\theta^{(k+1)} \approx \theta^{(k)}$, then the improvement in the maximization step is small and the algorithm is stopped.

Example 1 (Contd.)**.** *For two parameter vectors θ and $\tilde{\theta}$*

$$Q(\theta, \tilde{\theta}) = \sum_{i=1}^n \boldsymbol{E}_{\tilde{\theta}} \left[\ln f_\theta(Y, Z)|Y = y_i\right] = \sum_{i=1}^n \sum_{\ell=1}^L \ln\big(p_\ell \varphi_\ell(y_i, \delta_\ell)\big) k_{\tilde{\theta}}(\ell|y_i).$$

Maximization in $p = (p_1, \dots, p_L)$ subject to the constraint $\sum_{\ell=1}^{L} p_\ell = 1$ amounts to maximizing

$$\sum_{i=1}^{n} \sum_{\ell=1}^{L} \ln\big(p_\ell \varphi_\ell(y_i, \delta_\ell)\big) k_{\tilde{\theta}}(\ell|y_i) + \lambda \left(1 - \sum_{\ell=1}^{L} p_\ell\right)$$

for some suitable Lagrange multiplier $\lambda > 0$. Differentiating the above expression w.r.t. p_l and setting the resulting derivative to zero results in

$$p_\ell = \frac{1}{\lambda} \sum_{i=1}^{n} k_{\tilde{\theta}}(\ell|y_i).$$

Since $k(\cdot|y_i)$ is a discrete density function we further obtain

$$1 = \sum_{\ell=1}^{L} p_\ell = \frac{1}{\lambda} \sum_{\ell=1}^{L} \sum_{i=1}^{n} k_{\tilde{\theta}}(\ell|y_i) = \frac{1}{\lambda} \sum_{i=1}^{n} \sum_{\ell=1}^{L} k_{\tilde{\theta}}(\ell|y_i) = \frac{n}{\lambda}.$$

Hence, $\lambda = n$ and

$$p_l = \frac{1}{n} \sum_{i=1}^{n} k_{\tilde{\theta}}(\ell|y_i).$$

The update formulas for $\delta_1, \dots, \delta_L$ are obtained similarly, if a closed form expression of the densities $\varphi_\ell(\cdot, \delta_\ell)$ is at hand (cf. Exercises).

For a sequence $\theta^{(0)}, \theta^{(1)}, \dots$ generated by the EM-Algorithm, it is in general not true that $\theta^{(k)} \to \theta^*$ as $k \to \infty$. But at least one can prove that the likelihood is increasing.

Proposition 5. *Let $\theta^{(0)}, \theta^{(1)}, \dots$ be a sequence generated by the EM-algorithm. Then,*

$$\sum_{i=1}^{n} \ln g_{\theta^{(j+1)}}(y_i) \geq \sum_{i=1}^{n} \ln g_{\theta^{(j)}}(y_i).$$

Equality holds if and only if $\theta^{(j+1)} = \theta^{(j)}$.

Proof. Let $\theta \in \Theta$ and $y \in \mathbb{R}$ be arbitrary. Then,

$$\begin{aligned}
\int \ln\left(f_\theta(y,z)\right) k_{\theta^{(j)}}(z|y)\,\mathrm{d}z - \ln g_\theta(y) &= \int \ln\left(\frac{f_\theta(y,z)}{g_\theta(y)}\right) k_{\theta^{(j)}}(z|y)\,\mathrm{d}z \\
&= \int \ln\left(k_\theta(z|y)\right) k_{\theta^{(j)}}(z|y)\,\mathrm{d}z \\
&\leq \int \ln\left(k_{\theta^{(j)}}(z|y)\right) k_{\theta^{(j)}}(z|y)\,\mathrm{d}z \\
&= \int \ln\left(f_{\theta^{(j)}}(y,z)\right) k_{\theta^{(j)}}(z|y)\,\mathrm{d}z - \ln g_{\theta^{(j)}}(y)
\end{aligned}$$

Here we have used that by Jensen's inequality we have

$$\int \ln\left(\frac{k_\theta(z|y)}{k_{\theta^{(j)}}(z|y)}\right) k_{\theta^{(j)}}(z|y)\,\mathrm{d}z \leq \ln\left(\int \frac{k_\theta(z|y)}{k_{\theta^{(j)}}(z|y)} k_{\theta^{(j)}}(z|y)\,\mathrm{d}z\right) = 0.$$

Due to the strict concavity of $x \mapsto \ln x$, strict inequality holds unless $k_{\theta^{(k)}}(\cdot|y) = k_\theta(\cdot|y)$ a.e. Setting $y = y_i$ and summing over i now shows that

$$\begin{aligned}
Q(\theta, \theta^{(j)}) - \sum_{i=1}^{n} \ln g_\theta(y_i) &= \sum_{i=1}^{n} \int \ln\left(f_\theta(y_i,z)\right) k_{\theta^{(j)}}(z|y_i)\,\mathrm{d}z - \ln g_\theta(y_i) \\
&\leq \sum_{i=1}^{n} \int \ln\left(f_{\theta^{(j)}}(y_i,z)\right) k_{\theta^{(j)}}(z|y_i)\,\mathrm{d}z - \ln g_{\theta^{(j)}}(y_i) \\
&= Q(\theta^{(j)}, \theta^{(j)}) - \sum_{i=1}^{n} \ln g_{\theta^{(j)}}(y_i)
\end{aligned}$$

Setting $\theta = \theta^{(j+1)}$ finally shows

$$\sum_{i=1}^{n} \ln g_{\theta^{(j)}}(y_i) - \sum_{i=1}^{n} \ln g_{\theta^{(j+1)}}(y_i) \leq Q(\theta^{(j)}, \theta^{(j)}) - Q(\theta^{(j+1)}, \theta^{(j)})$$
$$= Q(\theta^{(j)}, \theta^{(j)}) - \max_{\theta \in \Theta} Q(\theta, \theta^{(j)}) \leq 0.$$

□

3.6 Extreme Value Theory

In $\mathbb{R}$ an extreme value is a (relatively) large or small data point. In $\mathbb{R}^d$, $d > 1$, an extreme is a data point with at least one coordinate large/small. The probability theory dealing with extremes or extreme values is the extreme value theory (EVT). The statistical theory based on the EVT is called the statistics of extremes. When extreme data points in a data set are outliers, that is, results of an error and can be regarded as unimportant, it is common practice to either ignore them or to try to limit their influence using different kinds of robust procedures. However, when these extreme data correspond to some significant real world rare events, such as a stock market crash or different kinds of weather or climate catastrophes, it is too costly and/or too dangerous to ignore them. Since the usual statistical tools are built to focus mainly on the central part of the data, a different approach is required to do the statistics for extremes. Here we consider only the large data, the right tail, in the $\mathbb{R}$-valued case. For the case of the left tail, note that $\min\{X_1, \ldots, X_n\} = -\max\{-X_1, \ldots, -X_n\}$.

EVT vs. CLT

CLT: sum of n iid random variables + appropriate normalization $\rightarrow$ standard normal distribution in the limit.

EVT: max of n iid random variables + appropriate normalization $\rightarrow$ extreme value distribution in the limit.

Domain of attraction condition Let $X_1, \ldots, X_n$ be i.i.d. random variables with distribution function F and $M_n := \max\{X_1, \ldots, X_n\}$. If there exist sequences $a_n > 0$ and $b_n \in \mathbb{R}$ such that

$$\frac{M_n - b_n}{a_n} \xrightarrow{d} Y, \text{ as } n \to \infty, \tag{3.4}$$

where Y is a random variable with a non-degenerate distribution function, say G, then F is in the domain of attraction (DoA) of G and G is called an extreme value distribution. The above domain of attraction can also be written as:

$$F^n(a_n x + b_n) \to G(x), \text{ for } n \to \infty, \tag{3.5}$$

for all x continuity points of G. The question now is, are there distribution functions F which satisfy this condition and what are the possible limiting distributions G?

Extreme value distributions and examples of functions in their DoA The family of extreme value distributions is a parametric family with a single parameter $\gamma \in \mathbb{R}$ called the extreme value index. It is given by

$$G_\gamma(x) = \begin{cases} \exp\left(-(1+\gamma x)^{-1/\gamma}\right) & 1 + \gamma x > 0 \\ 0 & x < -1/\gamma \text{ and } \gamma > 0 \\ 1 & x \geq -1/\gamma \text{ and } \gamma < 0. \end{cases}$$

For $\gamma = 0$ the term $(1+\gamma x)^{-1/\gamma}$ is to be understood as

$$\lim_{\gamma \to 0} (1+\gamma x)^{-1/\gamma} = \exp(-x).$$

Depending on the sign of the extreme value index, we get three different cases listed below.

- For $\gamma < 0$ the limiting extreme value distribution is of the reverse Weibull type, and the functions in their domain of attraction have finite right end-point[1], that is, they have finite right tails. Examples of distribution functions in the domain of attraction of a G_γ with a negative γ are the uniform and the beta distributions.

- For $\gamma = 0$ the limiting extreme value distribution is of the Gumbel or double-exponential type, $G_0(x) = \exp(-\exp(-x))$, and the functions in their domain of attraction have light tails. Examples of distribution functions in the domain of attraction of a G_γ with $\gamma = 0$ are the normal and the exponential distributions.

- For $\gamma > 0$ the limiting extreme value distribution is of the Fréchet type, and any distribution function in their domain of attraction has an infinite right end-point and heavy tails. Examples of such distributions are Cauchy, Pareto and t-distributions.

In particular, if a function is in the domain of attraction of an EVD with $\gamma > 0$, its moments of order greater than $1/\gamma$ do not exist.

Estimation of the extreme value index It can be shown that, if $\gamma > 0$, then (3.4) is equivalent to $x^* = \infty$ and

$$\frac{U(tx)}{U(t)} \to x^\gamma, \text{ as } t \to \infty,$$

where $U(x) := F^{-1}(1-1/x)$, with F^{-1} being the generalized inverse of F. Equivalently,

$$\log U(tx) - \log U(t) \to \gamma \log x, \text{ as } t \to \infty.$$

Now we proceed "heuristically": let $k \in \{1, \ldots, n\}$ such that $k = k_n \to \infty$ and $k/n \to 0$, as $n \to \infty$. Also write $t =: n/k$ and $x =: 1/y$. Then:

$$\begin{aligned} -\gamma \log y &\approx \log U\left(\frac{n}{ky}\right) - \log U\left(\frac{n}{k}\right) \\ &= \log F^{-1}\left(1 - \frac{ky}{n}\right) - \log F^{-1}\left(1 - \frac{k}{n}\right) \\ &\approx \log X_{\lceil n-ky \rceil :\, n} - \log X_{n-k:\, n}, \end{aligned}$$

where $X_{1:\, n} \le X_{2:\, n} \le \cdots \le X_{n:\, n} =: M_n$ are the order statistics of the sample. Integrating from 0 to 1 w.r.t. y both the left and the right hand side in the above (approximate) equality yields:

$$\gamma \approx \frac{1}{k} \sum_{i=0}^{k-1} \log X_{n-i:\, n} - \log X_{n-k:\, n} =: \hat{\gamma}_H.$$

The estimator $\hat{\gamma}_H$ is called the Hill estimator of the extreme value index γ, when $\gamma > 0$. The moment estimator from today's exercise (b), $\hat{\gamma}_m$, is an example of an estimator for a general $\gamma \in \mathbb{R}$.

For the next task, denote $a(t) := a_{\lfloor t \rfloor}$ and $b(t) := b_{\lfloor t \rfloor}$.

[1] The right end-point of a distribution function F is x^* defined as $x^* := \sup\{x\colon F(x) < 1\}$

Estimation of $a(n/k)$ and $b(n/k)$

- It can be argued that possible choices for a and b are: $b(t) = U(t)$ and $a(t) = tU'(t)$.
- A simple estimator of $b(n/k)$ thus is $\hat{b}(n/k) := X_{n-k:\,n}$;
- An estimator of $a(n/k)$ is

$$\hat{a}(n/k) := X_{n-k:\,n}\hat{\gamma}_H \left[1 - (\hat{\gamma}_H - \hat{\gamma}_m)\right].$$

If $\gamma > 0$, this simplifies to

$$\hat{a}(n/k) := X_{n-k:\,n}\hat{\gamma}_H.$$

Estimation of small probabilities and high quantiles Taking logarithms on both sides of (3.5) gives

$$n(1 - F(a_n x + b_n)) \to -\log G_\gamma(x), \text{ for } n \to \infty. \tag{3.6}$$

Replace n by n/k, with k as above, and $a_n x + b_n$ by y to obtain (for n "large"):

$$p_n := 1 - F(y) \approx \frac{k}{n}\left(1 + \gamma\frac{y - b(n/k)}{a(n/k)}\right)^{-1/\gamma}. \tag{3.7}$$

Replacing all the unknowns on the right-hand side with their estimators yields an estimator of a small probability $p = p_n$ of exceeding a prescribed high threshold y:

$$\hat{p} := \frac{k}{n}\left(\max\left\{0, 1 + \hat{\gamma}\frac{y - X_{n-k:\,n}}{\hat{a}(n/k)}\right\}\right)^{-1/\hat{\gamma}}.$$

Solving the above equation for y gives an estimator of a $(1-p)$-th quantile y, for a given value p:

$$\hat{y} := X_{n-k:\,n} + \hat{a}(n/k)\frac{\left(\frac{k}{np}\right)^{\hat{\gamma}} - 1}{\hat{\gamma}}.$$

In the above expressions, $\hat{\gamma}$ denotes any appropriate estimate of the extreme value index.

The choice of k For every different k, we obtain a different estimate of whatever quantity we are estimating. Heuristically, the choice of a good value k is often performed using the so-called Hill plot. The Hill plot is just a scatter-plot of the estimates (e.g. $\hat{\gamma}_H$ or p) as a function of k. Good values of k correspond to the first region where the plot looks stable: if k is too small, the estimation is based on too few observations, which is likely to lead to high variance; on the other hand, too large k violates the DoA conditions and leads to biased estimates.

3.7 Generalised Linear Models

Let $(Y_1, X_1), \ldots, (Y_n, X_n)$ be independent pairs of observations, where Y_i is real-valued and X_i are $\mathbb{R}^k$-valued random variables. Generalised linear models have the following three-part specification:

(a) The **random component** (=response from an overdispersed exponential family)
The data $Y_1, \ldots, Y_n$ are such that $Y_1|X_1, \ldots, Y_n|X_n$ are independent and $Y_i|X_i$ has the p.d.f.

$$f_{\eta,\psi}(y_i|x_i) = \exp\left\{\frac{\eta_i y_i - \kappa(\eta_i)}{\psi_i}\right\} h(y_i, \psi_i), \quad i = 1, \ldots, n,$$

where η_i is called **canonical parameter** and ψ_i is an unknown **scale or dispersion parameter**. Functions κ and h are known and $\kappa''(\eta) > 0$ is assumed. It is easy to see that

$$\mu(\eta_i) := \mathrm{E}(Y_i|X_i) = \kappa'(\eta_i) \quad \text{and} \quad \mathrm{Var}(Y_i|X_i) = \psi_i\kappa''(\eta_i), \quad i = 1, \ldots, n.$$

(b) The **systematic component** (=linear predictor)
Canonical parameter η_i is assumed to be related to X_i. The term $X_i^t\beta$ for unknown $\beta \in \mathbb{R}^d$ is called the linear predictor or systematic component.

(c) The **link function** between random and systematic components
The relationship between η_i and $X_i^t\beta$ is described through

$$g\{\mu(\eta_i)\} = X_i^t\beta, \quad i = 1, \ldots, n,$$

where g is called a link function. The link function g is assumed to be a known, one-to-one, third-order continuously differentiable function. If $g = \mu^{-1}$, then $\eta_i = X_i^t\beta$ and g is called the **canonical or natural link** function. If g is not canonical, then it is assumed that $d(g \circ \mu)(\eta)/d\eta \neq 0$ for all η.

In a GLM, the parameter of interest is β. Parameters ψ_i are considered to be *nuisance* parameters. It is often assumed that $\psi_i = \psi/t_i$, $i = 1, \ldots, n$ with an unknown ψ and known t_i's or, alternatively $\psi_i = a(\psi)$ for some known function a. Note that ψ_i enter $\mathrm{Var}(Y_i|X_i) = \psi_i\kappa^{''}(\eta_i)$, making it more flexible, that is allowing for *over- or under-dispersion*.

Example
Let $Y_i|X_i \sim Poi(\lambda_i)$. We can write the density

$$f_\eta(y_i) = \frac{\lambda_i^{y_i}}{y_i!}\exp(-\lambda_i)\mathbb{1}_{\{0,1,2,\ldots\}}(y_i) =: \exp\{y_i\log(\lambda_i) - \lambda_i\}\frac{1}{y_i!}\mathbb{1}_{\{0,1,2,\ldots\}}(y_i),$$

that is, the canonical parameter $\eta_i = \log(\lambda_i)$, $\kappa(\eta_i) = \lambda_i = \exp(\eta_i)$, $\psi_i = 1$ and $h(y_i) = (y_i)^{-1}\mathbb{1}_{\{0,1,2,\ldots\}}(y_i)$. Since $\mathrm{E}(Y_i|X_i) = \kappa^{'}(\eta_i) = \exp(\eta_i) =: \mu(\eta_i)$, the canonical link is $g(x) = \mu^{-1}(x) = \log(x)$, which is called the **log-link** ($g(\mu(\eta_i)) = \eta_i$). Hence,

$$\log\{\mathrm{E}(Y_i|X_i = x_i)\} = x_i^t\beta,$$

where $x_i \in \mathbb{R}^k$, $i = 1, \ldots, n$.

Estimation
Let $\theta = (\beta, \psi)$ and $(g \circ \mu)^{-1} = \zeta$ (for a canonical link $\zeta(x) \equiv x$). Then

$$\ell(\theta) = \sum_{i=1}^n \left[\frac{\zeta(X_i^t\beta)Y_i - \kappa\{\zeta(X_i^t\beta)\}}{a(\psi)} + \log h(Y_i, \psi)\right].$$

Further, consider the canonical link. Taking derivatives w.r.t. β and ψ we get the following score equations

$$\begin{aligned}
\frac{\partial\ell(\theta)}{\partial\beta} &= \frac{1}{a(\psi)}\sum_{i=1}^n \left\{Y_i - \mu(X_i^t\beta)\right\}X_i = 0 \\
\frac{\partial\ell(\theta)}{\partial\psi} &= \sum_{i=1}^n \left[\frac{\partial\log h(y_i,\psi)}{\partial\psi} + \{a^{-1}(\psi)\}^{'}\left\{X_i^t\beta Y_i - \kappa(X_i^t\beta)\right\}\right] = 0,
\end{aligned}$$

where $\kappa^{'}(X_i^t\beta) = \mu(X_i^t\beta)$ was used. If MLE of β exists, then it can be found from the first equation without estimating ψ. Estimation of ψ from the second equation in many cases is a difficult task and depends on a particular distribution.

To estimate β and study its properties we also need

$$-\frac{\partial^2\ell(\theta)}{\partial\beta\partial\beta^t} = \frac{1}{a(\psi)}\sum_{i=1}^n \kappa^{''}(X_i^t\beta)X_iX_i^t =: -\frac{F_n(\beta)}{a(\psi)}$$

With this, we can set up the Newton-Raphson algorithm as

$$\widehat{\beta}^{(j+1)} = \widehat{\beta}^{(j)} + \left\{F_n\left(\widehat{\beta}^{(j)}\right)\right\}^{-1} S_n\left(\widehat{\beta}^{(j)}\right), \quad j = 0, 1, 2 \ldots,$$

where $S_n(\beta) = a(\psi)\ell(\theta)/\partial\beta$.

The estimator $\widehat{\beta}$ of β, defined as a solution to $S_n(\beta) = 0$ can be shown to be consistent and asymptotically normal However, the case of non-canonical link has to be treated with care.

After the model is estimated, one would like to assess how good the model *fits* the data, i.e., to measure the discrepancy between the data $Y_i|X_i$ and estimated $\mathrm{E}(Y_i|X_i) = \mu_i$. Measures of discrepancy or *goodness-of-fit* can be formed in various ways, we will consider the deviance and generalised Pearson statistics.

The simplest model is the **null model**, it has only one parameter, representing a common mean μ, say, for all $Y_i|X_i$. At the other extreme is the **full model**, which has n parameters, one for each observation. The full model gives a baseline for measuring the discrepancy for an intermediate model with k parameters. Assume for the moment that ψ is known and denote $\ell(\widehat{\mu}, \psi)$ the log-likelihood with $\widehat{\mu} = g^{-1}(X\widehat{\beta})$. The maximum likelihood in the full model is then $\ell(Y, \psi)$ ($=\mu_i$ are replaced by Y_i). Then the **deviance of the fitted model** is defined as

$$D(Y, \widehat{\mu}) = a(\psi)\ 2\{\ell(Y, \psi) - \ell(\widehat{\mu}, \psi)\}.$$

Note that $D(Y, \widehat{\mu})/a(\psi)$ is called the **scaled deviance**. Some authors swap the definitions and call $2\{\ell(Y, \psi) - \ell(\widehat{\mu}, \psi)\}$ the deviance.

The **generalised Pearson statistic** is defined via

$$\chi^2 = \sum_{i=1}^{n} \frac{(Y_i - \widehat{\mu}_i)^2}{V(\widehat{\mu}_i)},$$

where $\widehat{\mu}_i$ and $V(\widehat{\mu}_i)$ are the estimated $\mathrm{E}(Y_i|X_i)$ and $\mathrm{Var}(Y_i|X_i)$, respectively.

Example
Let $(Y_1, X_1), \ldots, (Y_n, X_n)$ be independent, $Y_i|X_i \sim Poi(\lambda)$. Consider GLM with the canonical log-link. In this case $\mu_i = \exp(X_i^t\beta) = \exp(\eta_i) = \kappa(\eta_i) = \kappa^{'}(\eta_i)$ and we find

$$\begin{aligned} \ell(\widehat{\mu}) &= \sum_{i=1}^{n}\{Y_i \log(\widehat{\mu}_i) - \widehat{\mu}_i + \log h(Y_i)\}, \quad \ell(Y) = \sum_{i=1}^{n}\{Y_i \log(Y_i) - Y_i + \log h(Y_i)\} \\ D(Y, \widehat{\mu}) &= 2\sum_{i=1}^{n}\left\{Y_i \log\left(\frac{Y_i}{\widehat{\mu}_i}\right) - (Y_i - \widehat{\mu}_i)\right\}. \end{aligned}$$

Since $V(\mu_i) = \mu_i = \exp(X_i^t\beta)$, the Pearson statistics is given by

$$\chi^2 = \sum_{i=1}^{n} \frac{\{Y_i - \exp(X_i^t\widehat{\beta})\}^2}{\exp(X_i^t\widehat{\beta})}.$$

Scaled deviance can be used to compare two **nested models**, i.e. the parameter space under one model is a subspace of that under the second model. Assume $\eta_i = X_i^t\beta$, $\beta \in \mathbb{R}^k$ corresponds to a larger model M_k, say, and $\eta_i = \widetilde{X}_i^t\widetilde{\beta}$ with $\widetilde{\beta} \in \mathbb{R}^q$, $q < k$ corresponds to a smaller model M_q, say, where $\widetilde{X}$ is obtained from X by deleting $k-q$ columns of X. Models M_k and M_q are nested. If we would like to test the null hypothesis that a smaller model is as good as a larger one, this is equivalent to testing that $k - q$ parameters in M_k are zero. Thus, the difference between scaled deviances of M_k and M_q is twice the difference between

the log-likelihoods of the models M_k and M_q and is asymptotically χ^2_{k-q} distributed. The corresponding test is known as the **analysis of deviance**.

Since scaled deviance itself compares two nested models (the full one and a smaller one), it seems tempting to use the scaled deviance as a measure of goodness-of-fit. It is claimed that both the deviance and the Pearson statistics are approximately $a(\psi)\chi^2_{n-k}$ distributed, where k is the dimension of β in the model under consideration. Therefore, a widely used rule of thumb is that a good fit has the scaled deviance of about $n-k$, which is the expectation of a χ^2_{n-k} distributed random variable. Large values of the scaled deviance are considered to indicate a bad fit. However, this has to be treated with care. For Poisson data with large λ_i and Binomial data with large m_i the approximation to χ^2_{n-k} works reasonable, but not in many other cases.

To assess goodness-of-fit graphical tools such as **residual analysis** are used as well. There are several types of residuals, which are used for the residual analysis in the generalised linear models. These residuals are expected to behave approximately as zero-mean normally distributed variables.

Pearson residuals are defined by

$$r_i^p = \frac{Y_i - \widehat{\mu}_i}{\sqrt{V(\widehat{\mu}_i)}}, \quad i = 1, \ldots, n.$$

Pearson residuals have the disadvantage of being skewed for non-normal responses. Anscombe proposed a residual that uses a function $A(Y_i)$ instead of Y_i, where transformation A is chosen to make the distribution of $A(Y_i)$ as normal as possible. McCullagh and Nelder (1983) in Section 2.4.2 give **Anscombe residulas** for important cases. For example, for the Poisson distribution

$$r_i^a = \frac{3\left(Y_i^{2/3} - \widehat{\mu}_i^{2/3}\right)}{2\,\widehat{\mu}_i^{1/6}}, \quad i = 1, \ldots, n.$$

Another type of residuals, which is most widely used in practice, is based on the deviance. The **deviance residuals** are given by

$$r_i^d = \text{sign}(Y_i - \widehat{\mu}_i)\sqrt{2\{\ell_i(Y_i, \psi) - \ell_i(\widehat{\mu}_i, \psi)\}}, \quad i = 1, \ldots, n,$$

where ℓ_i is the log-likelihood corresponding to the i-th observation, so that $\sum_{i=1}^n (r_i^d)^2 = D(Y, \widehat{\mu})$.

Deviance residuals typically behave better than the Pearson ones and for most distributions are quite similar to the Anscombe residuals.
Similar to the normal response, a standardised version of the deviance (as well as Pearson) residuals are used:

$$\frac{r_i^d}{\sqrt{a(\widehat{\psi})(1-h_i)}}, \quad i = 1, \ldots, n,$$

where $h_i = H_{i,i}$ with the hat matrix H taking now the form $H = W^{1/2}X(X^tWX)^{-1}X^tW^{1/2}$, where W is the weight matrix from the Fisher scoring. In an adequate model the plot of standardised residuals against $\widehat{\eta} = X\widehat{\beta}$ should show no patterns. The so-called *null pattern* is a distribution of residuals with mean zero and constant variance.

To compare models with different subsets of parameters or even to compare two different models (e.g., with different link functions or a non-linear with a linear model), Akaike

information criterion (AIC) and Bayes information criterion (BIC) can be used. These two criteria are most popular examples of **penalised goodness-of-fit** criteria

$$\begin{aligned} \mathrm{AIC}(M) &= -2\ell(M) + 2|M| \\ \mathrm{BIC}(M) &= -2\ell(M) + \log(n)|M|, \end{aligned}$$

where $\ell(M)$ denotes the log-likelihood corresponding to a model M and $|M|$ is the number of parameters in that model M. The models, selected with these criteria are then

$$\begin{aligned} \widehat{M}_{\mathrm{AIC}} &= \arg\min_{M\in\mathcal{M}} \mathrm{AIC}(M) \\ \widehat{M}_{\mathrm{BIC}} &= \arg\min_{M\in\mathcal{M}} \mathrm{BIC}(M) \end{aligned}$$

In these criteria the term $-2\ell(M)$ describes the goodness-of-fit (e.g., residual sum of squares in the normal regression), while the second term penalises the number of the parameters in the model. Obviously, the larger the number of parameters in the model is, the smaller is the goodness-of-fit. Hence, the second term is the penalty for the large number of parameters (roughly speaking, the goal is a parsimonious fit: the best possible fit with the smallest possible number of parameters). Note that BIC has a larger penalty. One can show that BIC based model selection procedures are consistent, while AIC based procedures are conservative.

3.8 Kernel Density Estimation

Let $X_1, \ldots, X_n \overset{i.i.d.}{\sim} F$, where F is an unknown c.d.f. with a Lebesgue p.d.f. f.
First, choose a starting point x_0, a binwidth $h > 0$ and define bins (or classes) $I_j = [x_0 + jh - h, x_0 + jh)$, $j \in \mathbb{Z}$. W.l.o.g. we set $x_0 = 0$. Since $f(x) = F'(x)$ *a.e.*, a simple estimator for $f(x)$ at $x \in I_j$ would be

$$f_n(x;h) = \frac{F_n(jh) - F_n(jh-h)}{h} = \frac{1}{nh}\sum_{i=1}^{n} \mathbb{I}(jh - h < X_i \le jh) = \frac{1}{nh}\sum_{i=1}^{n} \mathbb{I}(X_i \in I_j).$$

This estimator is called the **regular histogram**.
In a histogram one fixes classes I_j and finds the number of observations that fall into each class, that is, $f_n(x;h) = h^{-1}\{F_n(jh) - F_n(jh-h)\}$. Recall again that

$$f(x) = F'(x) \approx \frac{F(x+h) - F(x-h)}{2h}$$

for some sufficiently small $h > 0$ and consider another approximation

$$\begin{aligned} \widehat{f}(x;h) &= \frac{F_n(x+h) - F_n(x-h)}{2h} = \frac{1}{2hn}\sum_{i=1}^{n}\{\mathbb{I}(X_i \le x+h) - \mathbb{I}(X_i \le x-h)\} \\ &= \frac{1}{2hn}\sum_{i=1}^{n} \mathbb{I}(x-h < X_i \le x+h) = \frac{1}{2hn}\sum_{i=1}^{n} \mathbb{I}\left(\frac{|X_i - x|}{h} \le 1\right) \\ &=: \frac{1}{nh}\sum_{i=1}^{n} K_u\left(\frac{X_i - x}{h}\right), \end{aligned}$$

where $h > 0$ and $h \to 0$ and

$$K_u(x) = \begin{cases} 1/2, & |x| \le 1 \\ 0, & |x| > 1 \end{cases}$$

is the density of the continuous uniform distribution on $[-1, 1]$.

Compared to a histogram, in $\widehat{f}(x;h)$ not the classes are fixed, but an interval around each X_i of length $2h$. Estimator $\widehat{f}(x;h)$ with K_u is known as the average shifted histogram or just the Rosenblatt estimator.

The Rosenblatt estimator, as well as a regular histogram, is piecewise constant (=not smooth), which is a clear drawback. A simple way out is to replace K_u by an appropriate smooth function K. Such a more general estimator is known as Parzen-Rosenblatt kernel density estimator or just kernel density estimator.

Definition
Let $X_1, \ldots, X_n \overset{i.i.d.}{\sim} F$ with a given density $F' = f$. A **kernel density estimator** for f is defined via

$$\widehat{f}(x;h) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right), \; x \in \mathbb{R}, \; h > 0.$$

Thereby $K : \mathbb{R} \to \mathbb{R}$, such that $\int_{-\infty}^{\infty} K(x)dx = 1$ is known as **kernel** and $h > 0$ is called **bandwidth**. A j**th moment** of a kernel K is defined as $\mu_j = \int_{-\infty}^{\infty} x^j K(x)dx$.

Some classical kernels:

(a) $K(x) = 0.5\, \mathbb{I}(|x| \leq 1)$ (the rectangular or uniform kernel)

(b) $K(x) = (1 - |x|)\, \mathbb{I}(|x| \leq 1)$ (the triangular kernel)

(c) $K(x) = 0.75(1 - x^2)\, \mathbb{I}(|x| \leq 1)$ (the Epanechnikov kernel)

(d) $K(x) = (2\pi)^{-1/2} \exp(-x^2/2)$ (the Gaussian kernel)

Let us consider the global risk of the kernel density estimator. Consider the mean integrated squared error

$$\begin{aligned} \mathrm{MISE}\left\{\widehat{f}(h)\right\} &= \mathrm{E} \int \left\{\widehat{f}(x;h) - f(x)\right\}^2 \mathrm{d}x = \int \mathrm{MSE}\left\{\widehat{f}(x;h)\right\} \mathrm{d}x \\ &= \int \left[\mathrm{bias}\left\{\widehat{f}(x;h)\right\}\right]^2 \mathrm{d}x + \int \mathrm{Var}\left\{\widehat{f}(x;h)\right\} \mathrm{d}x, \end{aligned}$$

where in the second equality Tonelli-Fubini theorem was used.

Since MISE is a risk corresponding to the $L_2(\mathbb{R})$-norm, it is natural to assume that f is smooth w.r.t. this norm. For example, we may assume that f belongs to a Nikolsky class.

Definition
Let $\beta > 0$ and $L > 0$. The **Nikolsky** class $N_2^{\beta}(L)$ is defined as the set of functions $f : \mathbb{R} \to \mathbb{R}$ whose derivatives $f^{(\ell)}$ of order $\ell = \lfloor \beta \rfloor$ exist and satisfy

$$\left[\int \left\{f^{(\ell)}(x+u) - f^{(\ell)}(x)\right\}^2 \mathrm{d}x\right]^{1/2} \leq L|u|^{\beta-\ell}, \quad \forall u \in \mathbb{R}.$$

Theorem
Let $X_1, \ldots, X_n \overset{i.i.d.}{\sim} F$, where c.d.f. F has a Lebesgue density f and $\widehat{f}(x;h) = (nh)^{-1} \sum_{i=1}^{n} K\{(X_i - x)/h\}$ be a kernel density estimator.

(i) Suppose that $K : \mathbb{R} \to \mathbb{R}$ is a function satisfying $\int \{K(x)\}^2 \mathrm{d}x < \infty$. Then for any $h > 0$, $n \geq 1$ and any density f

$$\int \mathrm{Var}\left\{\widehat{f}(x;h)\right\} \mathrm{d}x \leq \frac{1}{nh} \int \{K(x)\}^2 \mathrm{d}x.$$

(ii) Assume that $f \in \mathcal{F}_N = \{f : \ f \geq 0, \int f(x)dx = 1$ *and* $f \in N_2^\beta(L)\}$ *and let* K *be a kernel of order* $\ell = \lfloor \beta \rfloor$ *satisfying* $\int |x|^\beta |K(x)|dx < \infty$. *Then, for any* $h > 0$ *and* $n \geq 1$

$$\int \left[\text{bias}\left\{\widehat{f}(x;h)\right\}\right]^2 \mathrm{d}x \leq C_2^2 h^{2\beta},$$

where

$$C_2 = \frac{L}{\ell!} \int |x|^\beta |K(x)| \mathrm{d}x.$$

Hence, under assumptions of this Theorem we get

$$\text{MISE}\left\{\widehat{f}(h)\right\} \leq C_2^2 h^{2\beta} + \frac{1}{nh} \int \{K(x)\}^2 \mathrm{d}x.$$

The minimiser of the right hand side w.r.t. h is given by

$$h_{\text{MISE}} = \left[\frac{\int \{K(x)\}^2 dx}{2\beta C_2^2}\right]^{\frac{1}{2\beta+1}} n^{-\frac{1}{2\beta+1}},$$

so that $\text{MISE}\left\{\widehat{f}(h_{\text{MISE}})\right\} = \mathcal{O}(n^{-2\beta/(2\beta+1)})$.

So far we have not discussed the practical choice of K and h. First, we fix some kernel K and discuss how the bandwidth h can be chosen. One reasonable choice for h is a minimiser of $\text{MISE}\{\widehat{f}(h)\}$. However, $\text{MISE}\{\widehat{f}(h)\}$ (and hence the obtained h) depend on the unknown density f. Therefore, instead of minimising $\text{MISE}\{\widehat{f}(h)\}$, it is suggested to minimise an (approximately) unbiased estimator of $\text{MISE}\{\widehat{f}(h)\}$. First note that

$$\begin{aligned}
\text{MISE}\left\{\widehat{f}(h)\right\} &= \mathrm{E} \int \left\{\widehat{f}(x;h) - f(x)\right\}^2 \mathrm{d}x \\
&= \mathrm{E} \int \left\{\widehat{f}(x;h)\right\}^2 \mathrm{d}x - 2\mathrm{E} \int \widehat{f}(x;h) f(x) \mathrm{d}x + \int \{f(x)\}^2 \mathrm{d}x.
\end{aligned}$$

Since the last term is independent of h, minimisation of $\text{MISE}\{\widehat{f}(h)\}$ is equivalent to minimisation of

$$J(h) = \mathrm{E} \int \left\{\widehat{f}(x;h)\right\}^2 \mathrm{d}x - 2\mathrm{E} \int \widehat{f}(x;h) f(x) \mathrm{d}x.$$

Hence, it is sufficient to find an (approximately) unbiased estimator for each term of $J(h)$. Obviously, $\int \{\widehat{f}(x;h)\}^2 \mathrm{d}x$ is a trivial unbiased estimator for the first term. It remains to find an unbiased estimator for the second term. Let us show that

$$\frac{1}{n(n-1)h} \sum_{i=1}^n \sum_{j \neq i} K\left(\frac{X_j - X_i}{h}\right)$$

is an unbiased estimator for $\mathrm{E} \int \widehat{f}(x;h) f(x) dx$. Indeed, since $X_1, \ldots, X_n$ are i.i.d., we have

$$\begin{aligned}
\mathrm{E}\left\{\frac{1}{n(n-1)h} \sum_{i=1}^n \sum_{j \neq i} K\left(\frac{X_j - X_i}{h}\right)\right\} &= \mathrm{E}\left\{\frac{1}{(n-1)h} \sum_{j \neq 1} K\left(\frac{X_j - X_1}{h}\right)\right\} \\
&= \mathrm{E}\left\{\frac{1}{(n-1)h} \sum_{j \neq 1} \int K\left(\frac{X_j - u}{h}\right) f(u) \mathrm{d}u\right\} \\
&= \frac{1}{h} \int f(x) \int K\left(\frac{x-u}{h}\right) f(u) \mathrm{d}u \mathrm{d}x,
\end{aligned}$$

provided that the last expression is finite. On the other hand,

$$\begin{aligned} \mathrm{E}\int \widehat{f}(x;h)f(x)dx &= \mathrm{E}\left\{\frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{X_i-u}{h}\right) f(u)\mathrm{d}u\right\} \\ &= \frac{1}{h}\int f(x)\int K\left(\frac{x-u}{h}\right) f(u)\mathrm{d}u\mathrm{d}x, \end{aligned}$$

proving the claim. Putting all together, an unbiased estimator for $J(h)$ results in

$$\mathrm{CV}(h) = \int \left\{\widehat{f}(x;h)\right\}^2 dx - 2\frac{1}{n(n-1)h}\sum_{i=1}^{n}\sum_{j\neq i} K\left(\frac{X_j-X_i}{h}\right).$$

The function $\mathrm{CV}(\cdot)$ is called the **(leave-one-out) cross-validation criterion**. We have proved the following result.

Theorem
Assume that for a function $K : \mathbb{R} \to \mathbb{R}$ and for a density f satisfying $\int\{f(x)\}^2\mathrm{d}x < \infty$ and $h > 0$ we have

$$\int\int f(x)\left|K\left(\frac{x-u}{h}\right)\right| f(u)\mathrm{d}u\mathrm{d}x < \infty.$$

Then $\mathrm{E}\{\mathrm{CV}(h)\} = \mathrm{MISE}\{\widehat{f}(h)\} - \int\{f(x)\}^2\,\mathrm{d}x$.

Hence, functions $\mathrm{MISE}\{\widehat{f}(h)\}$ and $\mathrm{E}\{\mathrm{CV}(h)\}$ have the same minimisers. In turn, the minimizers of $\mathrm{E}\{\mathrm{CV}(h)\}$ can be approximated by those of the function $\mathrm{CV}(\cdot)$:

$$h_{\mathrm{CV}} = \arg\min_{h>0} \mathrm{CV}(h),$$

whenever the minimum is attained. Finally, we define the cross-validation kernel density estimator

$$\widehat{f}(x;h_{\mathrm{CV}}) = \frac{1}{nh_{\mathrm{CV}}}\sum_{i=1}^{n} K\left(\frac{X_i-x}{h_{\mathrm{CV}}}\right).$$

Note that h_{CV} also depends on the sample $X_1,\ldots,X_n$. It can be proved that under appropriate conditions the integrated squared error of $\widehat{f}(x;h_{\mathrm{CV}})$ is asymptotically equivalent to that of $\widehat{f}(x;h_M)$, where $h_M = \arg\min_{h>0}\mathrm{MISE}\{\widehat{f}(h)\}$ is unknown in practice (=oracle bandwidth).

Cross-validation is not the only way to obtain a bandwidth, which is optimal in some sense.

The choice of the kernel (among of the kernels of the same order) turns out to be less important than the choice of the bandwidth. First, let us state the theorem, which gives the asymptotic expression for $\mathrm{MISE}\{\widehat{f}(h)\}$ for a *fixed density* f.

Theorem
Assume that K is a kernel of order 1 satisfying the conditions

$$\int\{K(x)\}^2\mathrm{d}x < \infty, \quad \int x^2|K(x)|\mathrm{d}x < \infty, \quad \int x^2K(x)\mathrm{d}x \neq 0$$

and the density f is differentiable on $\mathbb{R}$, such that f' is absolutely continuous on $\mathbb{R}$ and $\int\{f''(x)\}^2dx < \infty$. Then for all $n \geq 1$

$$\mathrm{MISE}\{\widehat{f}(h)\} = \left[\frac{1}{nh}\int\{K(x)\}^2\mathrm{d}x + \frac{h^4}{4}\int x^2K(x)\mathrm{d}x\int\{f''(x)\}^2\mathrm{d}x\right]\{1+o(1)\},$$

where $o(1)$ is independent of n, but depends on f and tends to 0 as $h \to 0$.
Note that the $o(1)$ term depends on f, and hence this result holds for a fixed f, but not uniformly over a certain class of densities.

Obviously, one can scale the kernel K without violating assumptions on the kernel. We can take such a scaling parameter δ, that

$$\int \left\{\delta^{-1}K(x/\delta)\right\}^2 \mathrm{d}x = \left\{\int x^2\delta^{-1}K(x/\delta)\mathrm{d}x\right\}^2.$$

It is easy to see that choosing

$$\delta = \left[\frac{\int\{K(x)\}^2\mathrm{d}x}{\{\int x^2K(x)\mathrm{d}x\}^2}\right]^{1/5}$$

yields

$$\mathrm{MISE}\{\widehat{f}(h)\} = C(K)\left[\frac{1}{nh} + \frac{h^4}{4}\int_{-\infty}^{\infty} f^{''}(x)^2\mathrm{d}x\right]\{1 + o(1)\},$$

where

$$C(K) = \left[\int\{K(x)\}^2\mathrm{d}x\right]^{4/5}\left\{\int x^2K(x)\mathrm{d}x\right\}^{2/5}.$$

In particular, $C(K)$ is invariant to rescaling of K. A kernel scaled with δ is sometimes called *canonical kernel*. It permits "decoupling" of K and h (for a fixed f). For example, for a Gauss kernel $C(K) = (4\pi)^{-2/5}$. Canonical kernels are useful for (pictorial) comparison of density estimates based on different kernels of the same order, since they are defined in such a way that a particular single choice of bandwidth gives roughly the same amount of smoothing.

So far we considered kernel density estimators that are defined for densities on $\mathbb{R}$. However, there are many positive distributions with densities on $[0, \infty)$ or distributions with densities having a compact support.

Let $f(x) > 0$ for $x \in [0, \infty)$ and K is a kernel with the compact support $[-1, 1]$, so that $K\{(u - x)/h\}$ has support $[x - h, x + h]$. Then for $x < h$ $(x - h < 0)$

$$\begin{aligned}
\mathrm{E}\left\{\widehat{f}(x;h)\right\} &= \int_0^{x+h}\frac{1}{h}K\left(\frac{u-x}{h}\right)f(u)\mathrm{d}u = \int_{-x/h}^{1}K(v)f(x+hv)\mathrm{d}v \\
&= f(x)\int_{-x/h}^{1}K(v)\mathrm{d}v + hf^{'}(x)\int_{-x/h}^{1}vK(v)\mathrm{d}v + \ldots \\
\mathrm{Var}\left\{\widehat{f}(x;h)\right\} &= \frac{1}{nh}\int_{-x/h}^{1}\{K(v)\}^2\mathrm{d}v + \ldots
\end{aligned}$$

Since $x < h$, then for $x/h < 1$ we have in the bias term that $\int_{-x/h}^{1}K(v)\mathrm{d}v \neq 1$, as well as further possible terms in the bias $\int_{-x/h}^{1}v^jK(v)\mathrm{d}v \neq 0$, $j = 1, 2, \ldots$. The variance term is little influenced. Hence, $\widehat{f}(x;h)$ is not consistent for $x < h$.

There are several approaches to correct the behaviour of the kernel density estimators at the boundary.

3.9 Local Polynomial Regression

Let $(Y_1, X_1), \ldots, (Y_n, X_n)$ be i.i.d. as (Y, X) random variables, $Y \in \mathbb{R}$ and $X \in \mathbb{R}^d$. Consider the nonparametric regression model

$$Y_i = f(X_i) + \epsilon_i, \;\; \mathrm{E}(\epsilon_i | X_i) = 0, \;\; i = 1, \ldots, n$$

If f were a constant, then $\widehat{f}_n = n^{-1} \sum_{i=1}^n Y_i \to f$ a.s. (LLN).

If f is sufficiently smooth, then consider a finite (or countably infinite) partition $\{A_1, A_2, \ldots\}$ of $\mathbb{R}^d$, for Borel sets $A_j \subset \mathbb{R}^d$ and for all $x \in A_j$ estimate

$$\widehat{f}_n(x) = \frac{\sum_{i=1}^n \mathbb{I}\{X_i \in A_j\} Y_i}{\sum_{i=1}^n \mathbb{I}\{X_i \in A_j\}}, \; x \in A_j$$

(here and subsequently the convention $0/0 = 0$ is used).
This estimator is called **partitioning estimator** and in $d = 1$ is just a piecewise constant.

If instead of taking all $x \in A_j$, one estimates at each $x \in \mathbb{R}^d$ and generalizes the weight to some suitable $K : \mathbb{R}^d \to \mathbb{R}_+$, then

$$\widehat{f}_n(x; h) = \frac{\sum_{i=1}^n K\left(\frac{X_i - x}{h}\right) Y_i}{\sum_{i=1}^n K\left(\frac{X_i - x}{h}\right)} =: \sum_{i=1}^n W_i(x; h) Y_i, \; \forall x \in \mathbb{R}^d$$

for some $h > 0$. The function $W_i(x; h) = W_i(x; h, X_1, \ldots, X_n)$ is a weight function. A naive kernel would be $K(x) = \mathbb{I}\{\|x\| \leq 1\}$. This estimator is called **Nadaraya-Watson estimator**.

It is easy to see that the Nadaraya-Watson estimator can also be obtained as

$$\widehat{f}_n(x; h) = \arg\min_{c \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^n K\left(\frac{X_i - x}{h}\right) (Y_i - c)^2, \; \forall x \in \mathbb{R}^d.$$

This can be generalized as follows.

Let $g(\cdot; a) : \mathbb{R}^d \to \mathbb{R}$ be a parametric function of unknown parameters $a \in \mathbb{R}^{\ell+1}$, then define the estimator

$$\begin{aligned} \widehat{f}_n(x; h) &= g\left(x; \widehat{a}\right) \\ \widehat{a} &= \arg\min_{a \in \mathbb{R}^{\ell+1}} \frac{1}{n} \sum_{i=1}^n K\left(\frac{X_i - x}{h}\right) \left\{Y_i - g(X_i; a)\right\}^2. \end{aligned}$$

For $d = 1$ and $g(x; a) = \sum_{i=1}^{\ell+1} a_i x^{i-1}$, this estimator is referred to as a **local polynomial estimator** and is motivated by the Taylor expansion for some x_0 that is close to x

$$f(x) \approx f(x_0) + f^{'}(x_0)(x - x_0) + \cdots + \frac{f^{(\ell)}(x_0)}{\ell!}(x - x_0)^l =: \sum_{i=1}^{\ell+1} a_i x^{i-1}.$$

Consider now local polynomial estimators in more detail. Consider a random design nonparametric regression model

$$\begin{aligned} &Y_i = f(X_i) + \epsilon_i, \;\; i = 1, \ldots, n \\ &\mathrm{E}(\epsilon_i | X_i) = 0, \; \mathrm{E}(\epsilon_i^2 | X_i) = \sigma^2. \end{aligned}$$

For the regression function $f(x) = \mathrm{E}(Y|X = x)$ we assume that $f \in \Sigma(\beta, L)$ (a Hölder class with parameters β and L, $\lfloor\beta\rfloor = \ell$). If $f \in \Sigma(\beta, L)$ then for x_0 sufficiently close to some fixed $x \in [0, 1]$ we may write

$$f(x_0) \approx f(x) + f^{'}(x)(x_0 - x) + \ldots + \frac{f^{(\ell)}(x)}{\ell!}(x_0 - x)^{\ell} = A(x)^t P(x_0 - x) \in \mathcal{P}_{\ell+1},$$

where $A(x) = \left\{f(x), f^{'}(x), \ldots, f^{(\ell)}(x)/\ell!\right\}^t$ and $P(x_0 - x) = \left\{1, (x_0 - x), \ldots, (x_0 - x)^{\ell}\right\}^t$.

With this,

$$\widehat{A}_n(x) = \arg \min_{A \in \mathbb{R}^{\ell+1}} \sum_{i=1}^{n} \left\{Y_i - A^t P(X_i - x)\right\}^2 K\left(\frac{X_i - x}{h}\right)$$

is the local polynomial estimator of order $\ell + 1$ (degree ℓ) of $A(x)$.

Denote $e_k = (0, \ldots, 0, 1, 0, \ldots, 0) \in \mathbb{R}^{\ell+1}$ a unit vector with 1 at k-th position, $k = 1, \ldots, \ell + 1$. Then,

$$\widehat{f}^{(k-1)}(x) = (k-1)!\, e_k^t\, \widehat{A}_n(x)$$

is the local polynomial estimator of $f^{(k-1)}(x)$, $k = 1, \ldots, \ell + 1$.

In matrix notation

$$\begin{aligned} X &= \begin{pmatrix} 1 & (X_1 - x) & \ldots & (X_1 - x)^{\ell} \\ \vdots & \vdots & \ldots & \vdots \\ 1 & (X_n - x) & \ldots & (X_n - x)^{\ell} \end{pmatrix}, \quad Y = \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix} \\ V &= \operatorname{diag}\left\{K\left(\frac{X_1 - x}{h}\right), \ldots, K\left(\frac{X_n - x}{h}\right)\right\} \end{aligned}$$

we can write

$$\widehat{A}_n(x) = \arg \min_{A \in \mathbb{R}^{\ell+1}} \{Y - XA(x)\}^t\, V\, \{Y - XA(x)\} = (X^t V X)^{-1} X^t V Y,$$

which is unique, if $X^t V X$ is a positive definite matrix.

This representation makes obvious, that a local polynomial estimator of $f^{(k-1)}(x)$ is a linear estimator

$$\widehat{f}^{(k-1)}(x) = (k-1)!\, e_k^t\, (X^t V X)^{-1} X^t V Y = \sum_{i=1}^{n} W_{k,i}\,(x)\, Y_i,$$

with the weight function

$$W_{k,i}(x) = \frac{(k-1)!}{nh}\, e_k^t \left(\frac{1}{nh} X^t V X\right)^{-1} P(X_i - x) K\left(\frac{X_i - x}{h}\right).$$

Theorem
Let $\widehat{f}^{(k-1)}$, $k = 1, \ldots, \ell + 1$ be the degree $\ell \geq 0$ local polynomial estimator of $f^{(k-1)}$, where f is the regression function in a random design nonparametric regression model

$$Y_i = f(X_i) + \epsilon_i, \;\; \mathrm{E}(\epsilon_i|X_i) = 0, \; \mathrm{E}(\epsilon_i^2|X_i) = \sigma^2, \;\; i = 1, \ldots, n,$$

with unknown $\sigma^2 > 0$ and $f^{(\ell+1)}$ is bounded and continuous in a neighbourhood of x. Assume that

(i) kernel $K : [-1,1] \to [0,\infty)$ is a symmetric first order kernel with finite moments $\mu_j = \int_{-1}^{1} x^j K(x)\mathrm{d}x < \infty$, $j = 1,2,\ldots$ and $\int_{-1}^{1}\{K(x)\}^2\mathrm{d}x < \infty$;

(ii) the bandwidth h is such that $h = h(n) \to 0$ and $nh \to \infty$;

(iii) the marginal Lebesgue density of X_i, denoted by q, is assumed to be differentiable, bounded and bounded away from zero with $q^{'}$ being Lipschitz continuous.

Then, at $x \in [h, 1-h]$

$$\begin{aligned}
\mathrm{Var}\left\{\widehat{f}^{(k-1)}(x)\middle|\mathbf{X}\right\} &= \frac{\sigma^2\{(k-1)!\}^2}{nh^{2k-1}q(x)}\int_{-1}^{1}\{\mathcal{W}_k(u)\}^2\mathrm{d}u\{1+o_p(1)\} \\
\mathrm{bias}\left\{\widehat{f}^{(k-1)}(x)\middle|\mathbf{X}\right\} &= \begin{cases} \frac{h^{\ell+1-(k-1)}(k-1)!\,f^{(\ell+1)}(x)\kappa_{\ell+1}}{(\ell+1)!}\{1+o_p(1)\}, & (\ell+k-1)\ \textit{odd} \\ \frac{h^{\ell+2-(k-1)}(k-1)!\,f^{(\ell+1)}(x)\,q^{'}(x)\,\kappa_{\ell+2}}{q(x)(\ell+1)!}\{1+o_p(1)\}, & (\ell+k-1)\ \textit{even} \end{cases}
\end{aligned}$$

where $\kappa_\ell = \int_{-1}^{1} u^\ell \mathcal{W}_k(u)\mathrm{d}u$.

Remarks

(a) For $(\ell+k-1)$ odd, the asymptotic conditional bias is independent of $q(x)$ and is therefore **design-adaptive**.
For $(\ell+k-1)$ even, the asymptotic conditional bias depends on $q^{'}(x)/q(x)$.

(b) For $(\ell+k-1)$ even, the asymptotic conditional bias has the same asymptotic order $\mathcal{O}(h^{\ell+2-(k-1)})$ for $(\ell+k-1)$ and $(\ell+k)$. However, the constants are different.

(c) Similar to kernel density estimation, we observe the bias-variance trade-off: increasing h increases the bias, while reducing the variance (oversmoothing) and decreasing h decreases the bias, while increasing the variance (undersmoothing).

The following theorem gives the asymptotic conditional bias and variance at a left boundary point, that is $x \in [0,h)$. For the right boundary point the result is completely analogous.

Theorem

Under assumptions of previous Theorem, a local polynomial estimator $\widehat{f}^{(k-1)}$ of $f^{(k-1)}$ has the following asymptotic variance and bias at some $x \in [0,h)$:

$$\begin{aligned}
\mathrm{Var}\left\{\widehat{f}^{(k-1)}(x)|\mathbf{X}\right\} &= \frac{\sigma^2(0)\{(k-1)!\}^2}{nh^{2k-1}q(0)}\int_{-x/h}^{1}\{\mathcal{W}_k(u)\}^2\mathrm{d}u\{1+o_p(1)\} \\
\mathrm{bias}\left\{\widehat{f}^{(k-1)}(x)|\mathbf{X}\right\} &= \frac{h^{\ell+1-(k-1)}(k-1)!f^{(\ell+1)}(0)}{(\ell+1)!}\int_{-x/h}^{1}u^{\ell+1}\mathcal{W}_k(u)\mathrm{d}u\ \{1+o_p(1)\}.
\end{aligned}$$

Remarks

(a) For $(\ell+k-1)$ odd the rate of the bias $\mathcal{O}(h^{\ell+1-(k-1)})$ is the same for all $x \in [0,1]$, however, at the boundaries the constants are different and depend on x/h.

(b) For $(\ell+k-1)$ even, the rate of the bias at the boundary is larger, than in the interior (=boundary effect).

Under certain assumptions one can show that the estimator $\widehat{f}^{(k-1)}$ of $f^{(k-1)}$, $k = 1,\ldots,\ell+1$, $f \in \Sigma(\beta, L)$ satisfies

$$\limsup_{n\to\infty}\sup_{f\in\Sigma(\beta,L)}\mathrm{E}\left(n^{\frac{2\beta-2(k-1)}{2\beta+1}}\|\widehat{f}^{(k-1)} - f^{(k-1)}\|_2^2\right) \le C < \infty,$$

if $h = c\,n^{-1/(2\beta+1)}$, $c > 0$ is taken.

Remarks

(a) The convergence rate for derivatives is slower.

(b) The optimal bandwidth is independent on k.

Let us now discuss the choice of the bandwidth. Similar to kernel density estimation we are looking for an unbiased estimator of the L_2 risk of $\widehat{f}$ (=MISE$\{\widehat{f}(h)\}$). However, in regression models one can obtain, in general, only approximately unbiased estimators of MISE$\{\widehat{f}(h)\}$. In particular, we are able to find an unbiased estimator of a discretised version of the L_2 risk, that is of

$$\frac{1}{n}\sum_{i=1}^{n}\left\{f(X_i)-\widehat{f}_n(X_i)\right\}^2.$$

Consider the empirical L_2 risk $n^{-1}\sum_{i=1}^{n}\left\{Y_i-\widehat{f}_n(X_i;h)\right\}^2$. Obviously, minimizing this expression w.r.t. the smoothing parameter will result in an estimator $\widehat{f}_n$ which is closest to Y_i.

Let $\widehat{f}_n$ be an estimator, that can be written as

$$\widehat{f}_n(X_i;h)=\sum_{j=1}^{n}W_j(X_i;h)Y_j,$$

where $W_j(x;h)=W_j(x;h,X_1,\ldots,X_n)$ are some weight functions.
Assume $\mathrm{E}(\epsilon_i|X_1,\ldots,X_n)=0$ and $\mathrm{E}(\epsilon_i\epsilon_j|X_1,\ldots,X_n)=\sigma^2\delta_{ij}$, $\sigma\in(0,\infty)$.
Consider

$$\begin{aligned}
&\mathrm{E}\left[\frac{1}{n}\sum_{i=1}^{n}\left\{Y_i-\widehat{f}_n(X_i;h)\right\}^2\right]=\mathrm{E}\left[\frac{1}{n}\sum_{i=1}^{n}\left\{Y_i^2-2Y_i\widehat{f}_n(X_i;h)+\widehat{f}(X_i;h)^2\right\}\right]\\
=\;&\mathrm{E}\left[\frac{1}{n}\sum_{i=1}^{n}\left\{f(X_i)-\widehat{f}_n(X_i;h)\right\}^2\right]+\mathrm{E}\left[\frac{1}{n}\sum_{i=1}^{n}\left\{Y_i^2-f(X_i)^2\right\}\right]\\
&-\mathrm{E}\left(\frac{2}{n}\mathrm{E}\left[\sum_{i=1}^{n}\{Y_i-f(X_i)\}\widehat{f}(X_i;h)\middle|X_1,\ldots,X_n\right]\right)\\
=\;&\mathrm{E}\left[\frac{1}{n}\sum_{i=1}^{n}\left\{f(X_i)-\widehat{f}(X_i;h)\right\}^2\right]+\mathrm{E}\left(\frac{1}{n}\sum_{i=1}^{n}\epsilon_i^2\right)\\
&-\mathrm{E}\left[\frac{2}{n}\mathrm{E}\left\{\sum_{i=1}^{n}\epsilon_i\sum_{j=1}^{n}\epsilon_jW_j(X_i;h)|X_1,\ldots,X_n\right\}\right]\\
=\;&\mathrm{E}\left[\frac{1}{n}\sum_{i=1}^{n}\left\{f(X_i)-\widehat{f}_n(X_i;h)\right\}^2\right]+\sigma^2-\mathrm{E}\left\{2\sigma^2\frac{1}{n}\sum_{i=1}^{n}W_i(X_i;h)\right\},
\end{aligned}$$

so that the last term is "disturbing".

Mallows' C_p criterion is a simple way to correct for this term

$$C_p(h)=\frac{1}{n}\sum_{i=1}^{n}\left\{Y_i-\widehat{f}_n(X_i;h)\right\}^2+2\sigma^2\frac{1}{n}\sum_{i=1}^{n}W_i(X_i;h).$$

Apparently,

$$\mathrm{E}\left\{C_p(h)\right\}=\mathrm{E}\left[\frac{1}{n}\sum_{i=1}^{n}\left\{f(X_i)-\widehat{f}_n(X_i;h)\right\}^2\right]+\sigma^2$$

and h can be chosen as

$$\widehat{h} = \arg\min_{h>0} C_p(h)$$

Note that $C_p(h)$ criterion depends on an unknown σ^2, which needs to be estimated.

Other methods for smoothing parameter selection that (asymptotically) correct for the "disturbing" term include

$$\begin{aligned} \mathrm{AIC}(h) &= \log\left[\sum_{i=1}^{n}\left\{Y_i - \widehat{f}_n(X_i;h)\right\}^2\right] + \frac{2}{n}\sum_{i=1}^{n} W_i(X_i;h) \\ \mathrm{GCV}(h) &= \frac{\sum_{i=1}^{n}\left\{Y_i - \widehat{f}_n(X_i;h)\right\}^2}{\left\{1 - n^{-1}\sum_{i=1}^{n} W_i(X_i;h)\right\}^2}, \end{aligned}$$

3.10 Penalised Regression

Consider a linear regression model $y = X\beta + \varepsilon$; y is $\mathbb{R}^n$-valued, i.e. n observations, $\beta \in \mathbb{R}^p$ is an unknown parameters, and $X \in \mathbb{R}^{n\times p}$, $p > 0$, is a fixed design matrix of full rank. The random vector $\varepsilon \in \mathbb{R}^n$ has i. i. d. Gaussian entries with mean 0 and variance σ^2.
If $n \geq p$ and X has full column rank then

$$\hat{\beta} = \arg\min_{\beta} = \|y - X\beta\|_2^2 = (X^\top X)^{-1}X^\top y. \tag{3.8}$$

Then, the predicted values are

$$\hat{y} = X\hat{\beta} = X(X^\top X)^{-1}X^\top y = Hy,$$

where $H = X(X^\top X)^{-1}X^\top$ is called the hat matrix (it puts the hat on y). The map $v \mapsto Hv$ is the projection from $\mathbb{R}^n$ to the space spanned by the columns of X. The hat matrix is idempotent: $HH = H$.
We have $\operatorname{Var} y = I\sigma^2$ and $\operatorname{Var}\hat{y} = HIH\sigma^2 = H\sigma^2$, by the idempotence of H.
Using $\operatorname{tr}(AB) = \operatorname{tr}(BA)$, this implies that the overall variance of the predictions

$$\operatorname{tr}(\operatorname{Var}\hat{y}) = \operatorname{tr}(X(X^\top X)^{-1}X^\top)\sigma^2 = \operatorname{tr}((X^\top X)^{-1}X^\top X)\sigma^2 = p\sigma^2,$$

scales linearly as p varies between 1 and n. If $p > n$ the closed form solution in (3.8) is not valid anymore and the OLS estimate not unique.
Note that $v^\top X^\top Xv = \|Xv\|^2 \geq 0$, implying that $X^\top X$ is positive semi definite with non-negative eigenvalues. Weyl's inequality then implies that the eigenvalues of $X^\top X + \lambda I$ are at least λ. Thus, $X^\top X + \lambda I$ is positive definite and hence invertible. A simple approach to remedy rank deficiency therefore is to replace $(X^T X)^{-1}$ in (3.8) by $(X^T X + \lambda I)^{-1}$, for some $\lambda > 0$, yielding

$$\hat{\beta}_\lambda = (X^\top X + \lambda I)^{-1}X^\top y.$$

It can be shown that $\hat{\beta}_\lambda$ is the solution to a regularised least square problem:

$$\hat{\beta}_\lambda = \arg\min_{\beta} \|y - X\beta\|_2^2 + \lambda\|\beta\|^2, \tag{3.9}$$

where the L^2 penalty $\lambda\|\beta\|^2$ is sometimes called the ridge penalty.
A sensible procedure to choose the parameter λ would be to minimise the (discretised) MSE

$$\mathrm{MSE}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}(\mathrm{E}y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}\Big((X\beta)_i - (X\hat{\beta}_\lambda)_i\Big)^2.$$

In practice, because $\mathrm{E}y_i$ is unknown, we would have to use the empirical MSE:

$$\mathrm{MSE}_{\mathrm{emp}}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - (X\hat{\beta}_\lambda)_i\right)^2,$$

but this is minimised for $\lambda \to 0$ and would therefore lead to overfitting. To avoid this overfitting, we replace $\hat{y}_i$ by $\hat{y}_i^{[-i]}$, the prediction of y_i from the model fitted to all the data except y_i. The resulting leave-one-out criterion is

$$\mathrm{CV}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i^{[-i]})^2, \quad \lambda_{\mathrm{CV}} = \arg\min_{\lambda>0}\mathrm{CV}(\lambda).$$

For more efficient computations, the following representation can be used:

$$\mathrm{CV}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\frac{(y_i - \hat{y}_i)^2}{(1 - A_{ii})^2},$$

where A_{ii} are the diagonal elements of $A = X(X^\top X + \lambda I)^{-1}X^\top$. Note that this is more efficient, because only one fit is required, instead of n. Other, more computationally efficient variations are generalised cross-validation (GCV), where A_{ii} are replaced with their average in CV, or k-fold cross-validation.
We have seen that by adding the L^2 penalty to the regression problem, we can improve the predictive performance of the model (or for the case $p > n$ allow to use the model for prediction at all). The price is that, unlike the OLS estimate, the ridge regression estimate is not unbiased anymore:

$$\begin{aligned}
\mathrm{E}\hat{\beta}_\lambda - \beta &= (X^\top X + \lambda I)^{-1}X^\top \mathrm{E}y - \beta \\
&= (X^\top X + \lambda I)^{-1}X^\top X\beta - \beta \\
&= (X^\top X + \lambda I)^{-1}(X^\top X - (X^\top X + \lambda I))\beta \\
&= -\lambda(X^\top X + \lambda I)^{-1}\beta,
\end{aligned}$$

which increases as λ increases. Good predictive performance has been obtained by trading variance for bias.

In (3.9), we have considered the LS problem and added an L_2 penalty. More generaly, we can consider

$$\hat{\beta}_\lambda = \arg\min_{\beta} D(\beta) + \lambda P_q(\beta), \tag{3.10}$$

where $D(\beta)$ is a fitting function (previously the residual sum of squares, but could also be, e. g., the negative log likelihood) and $P_q(\beta) := \sum_{i=1}^{k}|\beta_i|^q$ is an L_q penalty. Note that if $q \geq 1$ then $P_q(\beta)^{1/q}$ defines a vector norm, but we can still consider $q \in (0, 1)$ and even define for $q = 0$, where $P_0(\beta) = \lim_{q\to 0} P_q(\beta) = \#\{i : \beta_i \neq 0\}$.
The minimiser $\hat{\beta}_\lambda$ in (3.10) lies on some contour $P_q(\beta) = c$ and it must be the minimiser along this contour:

$$\hat{\beta}_\lambda = \arg\min_{\beta} D(\beta) \text{ s. t. } P_q(\beta) = c, \tag{3.11}$$

because otherwise we could reduce $D(\beta)$ while leaving $P_q(\beta)$ unchanged and β_λ could not be the optimum.
As we reduce $q > 1$ towards 0, the contours $P_q(\beta) = c$ develop corners at $q = 1$. For $q \geq 1$ the contours are convex and for $q < 1$ they cease to be convex (allowing for multiple local minima). We have a unique solution if D and P_q are convex.
The L_2 penalty shrinks the coefficients $\hat{\beta}_i$ for which the data provides little evidence that $\beta_i \neq 0$ towards zero, but not exactly to zero. Often we would like those coefficients to be exactly zero, which would indicate that the corresponding effect is dropped from the model.

Parameter vectors with a large dimension where most of the components equal zero are called sparse. The case of $q = 1$ is special, because it entails convex contours, with corners that allow for a sparse solution.
Optimisation of non-differentialble functions, such as (3.11) with $q \leq 1$, is difficult, but again the case of $q = 1$ is special and $\hat{\beta}_\lambda$ can be computed simultaneously for all relevant λ at the cost of $\min\{\mathcal{O}(np^2), \mathcal{O}(n^3)\}$, the cost of a single least squares fit.

The L_1 penalised LS estimator was introduced under the name of LASSO (Least Absolute Shrinkage and Selection Operator) by Tibshirani (1996). From the preceding arguments we see that it performs model selection in addition to estimating parameters.
Here, model selection means that the design matrix X might be overly complex in the sense that some of its columns are not necessary to explain the data and could (and should) be omitted (along with the corresponding entries in β). The choice of which and how many columns of X should be omitted, if any, is difficult since the number of possible combinations of columns grows very fast with k. Traditionally, some criterion is introduced to specify the tradeoff between data fitting and model complexity. Possible methods are AIC- or BIC-type criteria. Direct application of either AIC or BIC is difficult. Note that, in our regression example, there are $2^p - 1$ models that can be considered. A naive implementation of model selection via AIC or BIC is therefore only possible when p is small. Otherwise, stepwise procedures in which covariates enter the model or are removed are common.
Omitting columns of X is equivalent to setting the corresponding entries of β to 0. To see the ability of the LASSO to do just this, consider the simple example of $p = 1$, so the matrix X degenerates to a column vector x and $y = x\beta + \epsilon$ with $\beta \in \mathbb{R}$. The OLS estimator of this model is $\hat{\beta}_{\text{OLS}} = \langle x, y\rangle / \langle x, x\rangle$. Clearly, $\hat{\beta}_{\text{OLS}} \neq 0$ almost surely, so this estimator will not be able to indicate directly that the model is overly complicated. The LASSO estimator, on the other hand, is given by

$$\hat{\beta}_{\text{LASSO}}(\lambda) = \text{argmin}_\beta \, \|y - \beta x\|^2 + \lambda|\beta|$$

Assume now that the solution of the optimization problem is > 0. Then it follows that it can be found by differentiating w.r.t. β and is given by

$$\hat{\beta}_{\text{LASSO}} = \frac{\langle x, y\rangle - \lambda/2}{\langle x, x\rangle},$$

hence this solution applies whenever $\langle y, x\rangle > \lambda/2$. Assuming, on the other hand, that the solution is < 0, it follows likewise that

$$\hat{\beta}_{\text{LASSO}} = \frac{\langle x, y\rangle + \lambda/2}{\langle x, x\rangle},$$

which holds whenever $\langle y, x\rangle < -\lambda/2$. In the remaining case $-\lambda/2 \leq \langle y, x\rangle \leq \lambda/2$, the solution is hence $\hat{\beta}_{\text{LASSO}} = 0$ (any other solution is ruled out by contradiction). This kind of behaviour generalizes to $p > 1$. Model selection is thus performed by shrinking the entries of the OLS towards 0, setting some entries to 0 exactly. The parameter λ controls the amount of shrinkage, with $\lambda = 0$ corresponding to the ordinary OLS estimator and $\lambda \to \infty$ resulting in $\hat{\beta}_{\text{LASSO}} = 0$. The problem of choosing a good value of λ will be discussed below.
A version of the LASSO estimator, the adaptive LASSO estimator $\hat{\beta}_{\text{AL}}$, was introduced by Zou (2006) and is defined by

$$\hat{\beta}_{\text{AL}}(\lambda) := \text{argmin}_{\beta \in \mathbb{R}^p} \, \|y - X\beta\|^2 + \lambda \sum_{i=1}^{p} |\beta_i| / |\hat{\beta}_i|,$$

where $\hat{\beta}$ is a ($\sqrt{n}$-consistent) "preliminary" estimator for β, usually the OLS-estimator, if available. The advantage of this estimator is that it employs a data-driven, component-specific shrinkage parameter $\lambda/|\hat{\beta}_i|$. Entries which are deemed "small" by the preliminary

estimator $\hat{\beta}$ get an increased shrinkage parameter compared to ones which are preestimated as being "large". In `R`, the (adaptive) LASSO estimator can be computed using (e.g.) the R-package `lars`. For further details on the package, see also Efron et al. (2004).
The problem of choosing a concrete value for the tuning parameter λ still remains. A sensible choice can be achieved by a BIC-type criterion Zou et al. (2007). Set

$$\mathrm{BIC}(\lambda) = \frac{\|y - X\hat{\beta}_{\mathrm{AL}}(\lambda)\|^2}{\sigma^2} + \log(n)|\{i : 1 \leq i \leq k, \hat{\beta}_{\mathrm{AL},i}(\lambda) \neq 0\}|$$

and choose the λ_n which minimizes this expression. Note that this is a fast operation (as opposed to using BIC for model selection directly) since the generation of the complete solution path for all λ is fast.
A different way of choosing λ is by cross validation. The basic idea is to split the sample randomly into two equally sized disjoint sets and calculate the LASSO solution path for one of them. Then use the value of λ which predicts the *other* subsample best. The package `lars` has cross validation built in so you don't need to implement it by hand.
Further information can, for example, be found in Chapter 3 of Hastie et al. (2017) that is available on `https://hastie.su.domains/ElemStatLearn/`.

Chapter 4

Exercises

4.1 Version Control with Git

Dataset

The data set `faithful` is available in base R.

Exercises

Getting familiar with Git

The goal of the exercises of this subsection is to get more familiar with Git, that is why each group member should do them on their own. You will be asked to create some branches. Use in that case your name as a prefix for the branch name, this way it is clear whose branch it is (e.g. if you are asked to create the branch named `base` you should name it `NAME-base` instead). Do NOT merge these branches into the Main branch. Make sure, that these branches are pushed into the remote repository, as we will check your results.

(a) Get access to your git repository and add your Name to the `README.txt` file.

(b) Create a branch named `base` and switch to the branch.

(c) Add the file `math.txt` from StudIP to your project. The text file contains three incorrect mathematical statements. Stage the file and commit the change.

(d) Create two branches: `branch-A` and `branch-B`. All three branches should now point to the same commit.

(e) On each branch, follow these instructions and commit them:

- `base`: Correct the first statement.
- `branch-A`: Correct the third statement.
- `branch-B`: Correct the second statement.

(f) Follow the instructions below in the given order. One instruction will cause a problem, the other one not, why? Find out how to resolve the problem and resolve it such that all changes are kept.

(i) Merge `branch-A` into `base`.

(ii) Merge `branch-B` into `base`.

Now `math.txt` on the branch `base` should contain three correct statements.

(g) Switch to `base`, if you are not on it. Delete `math.txt` and commit the change.

(h) Push the changes onto the remote repository. **Remember**: Always pull before you push to prevent problems!

(i) Undo the deletion of `math.txt` without changing the past history of the remote repository. Push the changes again.

Collaboration

In this subsection you are going to use Git to create a feature which implements a routine from machine learning as a team. Therefore, distribute the following tasks evenly to each group member. The resulting files will then be merged into your main branch.

(a) Create a branch called `fet-split_routine` and switch to it.

(b) Create 3 branches with meaningful names. Each branch should contain one of the following functions as a file:

- A function with the arguments $n \in \mathbb{N}$, $p \in (0, 1)$ which creates a random sample of size $\lceil np \rceil$ from $\{1 \dots n\}$ without replacement.
- A function which has a dataset and a list of indices as arguments which then creates two datasets: a dataset which contains all entries which correspond to the given indices (the train set) and a dataset which contains the rest (the test set).
- A function which has two datasets as arguments. The function applies a linear transformation to the first dataset (the train set), such that it has mean 0 and variance 1. The same transformation is then applied to the second dataset (the test set). The function then returns both normalized datasets.

(c) Merge all three branches into the `fet-split_routine` branch. Write a function which uses the previous functions to split and normalize a dataset into a train and test set.

(d) Apply the function to the variable `faithful$waiting` with $p = 0.5$ and plot the histograms of the train set and the test set. Compare both plots.

(e) Merge the branch into the main branch.

Project Management

In the end, the main branch should contain at least 11 folders: one for the code of each day and a folder which contains the written report of each member, otherwise you are free to choose on how to branch and merge the project, as long as it is appropriate.

(a) Write a short text where you explain your strategy of branching and managing the project. Are you satisfied with the resulting structure of your project, or would you have done something differently in hindsight?

R functions

You may find useful the following R functions: `sample.int`.

4.2 Tidyverse

Dataset

The dataset *childrenfinal.dta* is obtained from the **Kenyan Demographic and Health Survey 2003** and contains various variables sampled in 2003 on the Kenyan children of age between 0 and 5 years. The data are cross-sectional, there are no same children observed. There are 4686 observations on 177 variables, most of the variable names are self-explained.

Exercises

In this exercise the goal is to learn how to work with tools of the *tidyverse* package.

(a) The data `childrenfinal.dta` are given in the STATA format. Read the data into R using an appropriate function from the *tidyverse*. Next, remove all variables that start with "s", "v" and "m", followed by a number (avoid listing all of them). Check all the remaining variables. Do all variables have reasonable variable type (character, factor, double, integer, etc)? Convert the variables to a suitable type, if necessary.

(b) Make a smaller tibble that contains variables `hypage`, `ruralfacto`, `female`, `zstunt`, `zweight`, `zwast`, `adm2`. Variable `zstunt` is the so-called Z-score for stunting and is defined as the height of a child standardised with the median and standard deviation of heights of children at the same age from a healthy population. Children with Z-score less than -2 are defined to be stunted. Make a scatter plot of `zstunt` against `hypage`. Add a smooth line to the plot. Comment on the results. Now, make smooth plots of `zstunt` against `hypage` for females and males on one plot, add a suitable legend. Use different colors for males and females. Similarly, plot `zstunt` against `hypage` for urban and rural children. Comment on the results. Experiment with different aesthetics, themes and font sizes for the plots, report your favourite(s).

(c) Plot the map of Kenya with all counties listed in `adm2`. Colour the county areas according to the mean of `zstunt` in the corresponding county. Note that one county (Isiolo) is missing in the data. Make suitable legend and add county names (or corresponding labels) to the map. Comment on the results. In which counties are the children stunted?

(d) Finally, write the tibble from (b) into a text file. It will be used again in Exercise 9.

4.3 Pseudo Random Number Generation

Exercises

(a) Switch the default random number generator in R to `Wichmann-Hill`. Simulate $N = 1000$ geometrically distributed random variables $\text{Geo}(p = 0.4)$ using three approaches: inversion method, by using Bernoulli random variables that are simulated by inversion method and using R function `rgeom`. Plot the empirical probability density functions of all three samples on one panel. Comment on the results. Switch the random number generator back to its default.

(b) The aim is to simulate $N = 10\,000$ standard normal distributed random variables with density $f(x) = (2\pi)^{-1/2} \exp(-x^2/2)$ using accept-reject method and a generator for uniform random variables only. As a candidate density g use the density of the standard Laplace distribution $g(x) = \exp(-|x|)/2$.

- First determine the best value of the constant c, such that $f(x) \leq cg(x)$.
- Obtain N standard normal random variables using the accept-reject method, generating Laplace distributed random variables using inversion method. Compare estimated and theoretical acceptance probabilities. Plot a histogram of the obtained sample and add the standard normal density to the plot. Make a QQ-plot. Comment on the results.
- Show that it is not possible to simulate from the standard Laplace density using the accept-reject method with a standard normal candidate density.

R functions

You may find useful the following R functions: `apply, RNGkind, which`.

4.4 Bootstrap

Dataset

The dataset *shhs1.txt* has been obtained from **Sleep Heart Health Study**, where more information on the data can be found. We will be using only the following variable

`rdi4p`: respiratory disturbance index

Exercises

In the following set the sample size $n = 100$, the number of bootstrap replications $R = 1000$ and the number of Monte Carlo samples $M = 1000$.

(a) Simulate a sample $(x_1, \ldots, x_n)$ from the Weibull distribution with the scale parameter $\lambda = 13$ and shape parameter $k = 1$. The variance of a Weibull distributed random variable is given by $\sigma^2 = \lambda^2 \left[\Gamma(1 + 2/k) - \{\Gamma(1 + 1/k)\}^2\right]$, while the median $x_{med} = \lambda\{\log(2)\}^{1/k}$. We aim to build confidence intervals for σ based on a statistic $\hat{s}^2 = (n-1)^{-1} \sum_{i=1}^{n} (x_i - \bar{x})^2$ and for x_{med} based on the sample median.

- Build two-sided bootstrap percentile confidence intervals for σ and x_{med} at the significance level $\alpha = 0.95$. Use M Monte Carlo samples to estimate the coverage probability of both confidence intervals. Estimate also the average interval length. Comment on the results. How the results change if $n = R = 1000$ and $n = 100$, $R = 5000$? Comment on both accuracy of the coverage probabilities and the average length of confidence intervals in both cases.
- Use R function `bcanon` from the package *bootstrap* to build bootstrap accelerated bias-corrected confidence intervals both for σ and x_{med}. Use M Monte Carlo samples to assess the coverage probability and the average length of the confidence intervals. Comment on the results and differences to bootstrap percentile confidence intervals. Check estimated $\hat{z}_0$ and $\hat{a}$, comment on these values. Explain the performance of all obtained confidence bands.

(b) Consider now variable `rdi4p` from the dataset *shh1.txt*. Plot the histogram of this variable and describe the empirical distribution. Build bootstrap percentile and bootstrap accelerated bias-corrected confidence intervals for the standard deviation and median. Comment on the results.

R functions

You may find useful the following R functions: `apply, bcanon` (library *bootstrap*), `sample`

4.5 Expectation Maximization

Dataset

The data sets `quake` s$depth and and `faithful` $waiting are available in base R.

Exercises

(a) We use the EM-Algorithm to estimate the parameters of a Gaussian mixture distribution (with $L = 2$ components). Compute the explicit formulas for updating the means (μ_1, μ_2) and variances (σ_1^2, σ_2^2) of the mixture components as well as the weights $p_1 = p$ and $p_2 = 1 - p$.

$$\theta^{(0)} = (\mu_1^{(0)}, \mu_2^{(0)}, \sigma_1^{(0)}, \sigma_2^{(0)}, p^{(0)}) = (m - sd/2, m + sd/2, sd, sd, 0.5)$$

where m is the mean of all observation and sd the standard deviation, respectively. Comment on the results. Is the assumption of a Gaussian mixture realistic for both datasets?

Recall that the EM-algorithm has the property of generating a *non-decreasing* sequence of values of the (log-)likelihood function

$$\sum_{i=1}^{n} \log g_{\theta^{(k)}}(y_i).$$

which is strictly increasing unless $\theta^{(k)} = \theta^{(k-1)}$. Demonstrate this fact for the two data sets. How can this be used to formulate a stopping criterion for the EM-Algorithm?

(b) Find another data set of your choice and fit a finite mixture distribution.

4.6 Extreme Value Theory

Dataset

The dataset `39001_gdf.csv` is available at the **National River Flow Archive**. It contains the mean daily river flow of the river Thames, given in cubic meters per second (m^3/s), measured from 9.00 GMT to 8.59 GMT the following day. The measurements were taken over the time period from 1 Jan 1883 to 30 September 2021.

Exercises

(a) Define your own `R` function that for a given data set and the range of values of k calculates the Hill estimator of the extreme value index γ of the distribution of the data, assuming that this index is positive. Simulate 100 samples of size $n = 1000$ from each of the following distributions

(a) standard Cauchy,

(b) standard Fréchet,

(c) Student-(3) distribution

and estimate γ for each sample.

First, plot the Hill plot, that is plot the Hill estimator of γ as a function of k for a few samples from each distribution. Then, for each example, choose a good ("optimal") range of k values and define the final estimate of γ to be the average over those k. Plot the histogram of 100 estimates for each distribution.

(b) Read in the NRFA river flow dataset correctly and give the columns meaningful names. Propose a sensible model that accounts for the seasonal variation of the mean, use it to correct for the changing means and proceed with the residuals. Visually verify that there the residuals can be considered identically distributed.

Use the `R` code in file `Moment.function.R` available at Stud.IP to estimate the extreme value index for the residuals of the river flow levels. Select the "optimal" k visually, with the help of the Hill plot. Plot the density of the extreme value distribution corresponding to your estimated $\hat{\gamma}$ together with a histogram of the annual maxima of the residuals in the dataset. Shift and scale these datapoints appropriately, so that

it aligns as good as possible with the density. Does your value for $\hat{\gamma}$ seem plausible? The moment estimator of γ is defined as:

$$\begin{aligned} \hat{\gamma}_{\mathrm{m}} &:= \hat{\gamma}_{\mathrm{H}} + 1 - \frac{1}{2}\left(1 - \frac{\hat{\gamma}_{\mathrm{H}}^2}{M_{n,2}}\right)^{-1}, \quad \text{where,} \\ M_{n,2} &= \frac{1}{k}\sum_{i=0}^{k-1}\left(\log X_{(\lceil n-i \rceil)} - \log X_{(n-k)}\right)^2, \end{aligned}$$

and $\hat{\gamma}_{\mathrm{H}}$ is the Hill estimator of γ.

(c) Simulate 100 random samples of size $n = 2000$ from the distribution function F given by:

$$F(x) = \exp\left\{-\left(1 + 0.85(x-1)\right)^{-\frac{1}{0.85}}\right\},$$

for $1 + 0.85(x-1) > 0$. Recall that for $U \sim \mathrm{Unif}(0,1)$ we obtain $F^{-1}(U) \sim F$.

Fix $k = 200$ and for each of the simulated samples estimate

(a) the extreme value index γ,

(b) the $(1-p)$-th quantile, for $p = 0.0001$,

(c) $1 - F(c)$, for $c = 417.1$.

Present the results as boxplots in each of the three cases.

4.7 Generalised Linear Models

Dataset

The dataset *student-mat.csv* can be found on **Kaggle**. This page contains the full description of the data and all the variables. Variables `G1, G2, G3` are first, second and final grades in mathematics. The remaining variables are explanatory variables. We would like to identify variables that explain grades in mathematics.

Exercises

(a) First we need to identify the distribution of each of `G1, G2, G3`. Can each of these variables be assumed to follow a normal distribution? Justify your answer using suitable arguments and graphical tools. Can each of `G1, G2, G3` be assumed to follow a Poisson distribution? Are there signs for over-dispersion or any other anomalies in the distributions of any of `G1, G2, G3`? Support your answer using suitable arguments and graphical tools.

(b) Fit a suitable (generalised) linear model to explain `G1` including all explanatory variables (Model 1). Are all covariates significant? Comment on the goodness-of-fit of this model. Calculate the Pearson residuals and Anscombe residuals and assess how closely they follow a normal distribution. Pursue the residual analysis and comment if the fitted (generalised) linear model is adequate for the data.

(c) Take Model 1, but reduce the covariates to `sex`, `Fedu`, `studytime`, `failures`, `schoolsup`, `famsup`, `goout` (Model 2). Are all the covariates significant? Interpret the effect of each covariate on the grade. Assess the goodness-of-fit of this model. Perform an analysis of deviance test to compare Model 1 and Model 2. Comment on the results. In Model 2 replace `goout` by `Walc` to get Model 3. How can one compare Model 2 and Model 3? Which model delivers a better fit? Justify your answer.

R functions

You may find useful the following R functions: `glm`.

4.8 Kernel Density Estimation

Dataset

The dataset *StudentsPerformace.csv* can be found on **Kaggle datasets**. This page contains also the background information on the data. In our analysis we will only consider the following variables:

`test.preparation.course`: If a student took part at the preparation course

`math.score`: Score on the math exam (0–100)

`reading.score`: Score on the reading exam (0–100)

`writing.score`: Score on the writing exam (0–100)

Exercises

(a) Implement kernel density estimation in an R function that depends on the sample, bandwidth and a kernel. Read the data into R. Estimate and plot the density of `math.score` with the Epanechnikov kernel and 4 different choices of the bandwidth, putting them onto one plot. Next, fix the bandwidth you find most reasonable and plot kernel density estimators for 4 different choices of kernel functions (Gauss, Epanechnikov, uniform, tridiagonal), putting them onto one plot. Do not forget to put legends on both plots. Comment on the effect of the bandwidth and of the kernel function on kernel density estimators.

(b) Implement the cross-validation criterion to find the optimal bandwidth. Compare your resulting bandwidths with the ones obtained by built-in R functions `bw.ucv` and `bw.bcv` used in `density` for all three scores `math.score`, `reading.score` and `writing.score`.

(c) Use your implementation of the kernel density estimator with the cross-validated bandwidth to compare graphically densities of all three scores of the students that did not take part in the preparation course with the students who attended the preparation course. Comment on the results.

R functions

You may find useful the following R functions: `apply, density, integrate, optimize, outer, Vectorize`.

4.9 Local Polynomial Regression

Dataset

Consider again the dataset on Kenyan children from the exercise about the **tidyverse**. We are interested in the following two variables

`hypage`: Age of a child

`zwast`: Z-score for wasting

Z-score for wasting is defined as the weight of a child standardised with the median and standard deviation of children with the same height from the healthy population. We would like to investigate how the Z-score for wasting changes with age, that is we consider the model $zwast_i = f(hypage_i) + \epsilon_i$, for $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, $i = 1, \ldots, n$.

Exercises

(a) Write an R function that calculates a local polynomial fit and depends on the response, covariate, bandwidth, polynomial degree, kernel and the number of derivatives. First fix the polynomial degree to 1, estimate f with 4 different bandwidths and plot all resulting fits on one plot. Next, choose the most reasonable bandwidth, fix the polynomial degree to 1, estimate f with 4 kernel from the exercise on the kernel density estimation and put all fits on another plot. Comment on the results.

(b) Write a function that calculates the optimal bandwidth with Generalised Cross Validation (GCV). Use Epanechnikov kernel and GCV-bandwidth to estimate f using polynomial degrees from 1 to 4. List obtained GCV-bandwidths for each polynomial degree and comment on the results. Plot all four fits putting the curves on the same plot. Comment on the differences in the overall fits and at the boundaries. Do you find fits with the obtained GCV-bandwidths reasonable?

(c) Estimate the first derivative of the function of `zwast` with the GCV-bandwidth and polynomial degrees from 1 to 4. Plot all four derivative fits putting the curves on the same plot. Comment on the difference. Interpret the results. Are there indications that the Z-score is improving after 2 years? Do you find derivative fits with the obtained GCV-bandwidths reasonable?

R functions

You may find useful the following R functions: `lm, influence`.

4.10 Penalised Regression

Dataset

The first data set `prostate.data` has the log prostate specific antigen of 97 patients and 8 further predictors. This data and a full description of the data and all the variables are available on `https://hastie.su.domains/ElemStatLearn/data.html`. In the description, note the comment about the need to standardise the predictors.

The second data set is `diabetes` from the **lars** package.

Exercises

(a) Define your own R function for a given data set (y and X) and value of λ to compute and return β_λ and $\mathrm{CV}(\lambda)$. Apply it to the prostate dataset, present a plot $\lambda \mapsto \mathrm{CV}(\lambda)$ (for a reasonable range of λ) and report your findings.

(b) Make up a regression model with at least one large coefficient, one equal to zero and

one equal to $1/\sqrt{n}$, e.g.

$$X = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 0 \\ 1 & 3 & 1 \\ 1 & 4 & 0 \\ \vdots & \vdots & \vdots \\ 1 & n-1 & 1 \\ 1 & n & 0 \end{pmatrix}, \qquad \beta = (3, 0, 1/\sqrt{n})', \qquad \varepsilon \sim N(0, \mathbb{I}_n)$$

Estimate β via the Lasso and adaptive Lasso estimator. Plot the solution paths, pick an estimator using CV or BIC. Get used to how the package works.

(c) Repeat the above procedure for the adaptive Lasso estimator N times with $n = 100$ to estimate the average size of the model chosen by the estimator. Do this for both using CV and BIC. Also estimate the prediction error $\mathrm{E}[(y - X\hat{\beta})^T(y - X\hat{\beta})]$. Repeat for an "easier" parameter vector, such as $\beta = (3, 0, 5)^T$.

(d) Use the adaptive Lasso estimator for the `diabetes` data set of the `lars` package. Comment on your findings.

You do not have to include part (b) of the exercise in your report.

R functions

For the (adaptive) Lasso estimator, e. g. the **lars** or **glmnet** package in R can be used.

Bibliography

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc. Ser. B*, 39(1):1–38. With discussion.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *Annals of Statistics*, 32:407–499.

Hastie, T., Tibshirani, R., and Friedman, J. (2017). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B*, 58:267–288.

Wickham, H. and Grolemund, G. (2017). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly Media.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101:1418–1429.

Zou, H., Hastie, T., and Tibshirani, R. (2007). On the "degrees of freedom" of the lasso. *Annals of Statistics*, 35(5):2173–2192.