Computational Physics

Marcus Petschlies and Carsten Urbach

11th December 2019

This is a script of the lecture *Computational Physics* given at Bonn University in the years from 2010 to 2016. It is based on notes taken in a lecture by Prof. Dr. Volkard Linke at FU Berlin and the references given in each chapter.
The theme of this lecture is Monte-Carlo methods in physics and other natural sciences, with particular emphasis on Markov chain Monte-Carlo (MCMC). But it also introduces the needed theory on probability and statistical tools any physicist should be expert in.

Please send typos or mistakes to urbach@hiskp.uni-bonn.de.

Contents

List of Algorithms

1. Introduction

In physics we often face problems which cannot be solved analytically. Many of such problems can be attacked using perturbative approaches, which basically means that one expands around a solvable problem in a small coupling parameter. Such an approach works well as long as there is a small parameter for the expansion. However, perturbative approaches fail for strongly coupled system, i.e. when the expansion parameter becomes large.

In such situations one reverts to numerical methods in order to solve such problems. There is a vast amount of numerical methods providing systematically improvable approximate solutions e.g. for integrals or differential equations. For integrals, for instance, such approximate solutions mostly come from discretising the integral. Such a solution can be improved systematically by making the discretisation finer and finer. Analysis then provides one also with worst case estimates for the discretisation error.

The Curse of Dimensionality

For the problems we are going to be interested in here, such approaches become unfeasible. In order to see why, we consider grid-based methods to solve integrals numerically. The convergence rate for such grid-based quadrature methods is $\mathcal{O}(N^{-k/d})$ for an order k method in d dimensions. The reason for this scaling is that the grid with N points in the unit cube has spacing $N^{-1/d}$. As an example we recall Simpsons rule for an integral in one dimension for even N

$$\int_a^b f(x)\,\mathrm{d}x \;=\; \frac{h}{3}(f_0 + 4f_1 + 2f_2 + 4f_3 + \cdots + 4f_{N-1} + f_N) + \mathcal{O}(h^4)\,, \tag{1.0.1}$$

with $h = (b-a)/N$ and $f_i = f(x_i)$. This is a quadrature formula of order $k = 4$ and, more precisely, the absolute value of the error is strictly bounded by

$$\frac{b-a}{180}h^4 \max_{x\in[a,b]} |f^{(4)}(x)|\,,$$

if the fourth derivative of f can be computed. Of course, quadrature formulae for any order k can in principle be derived.

For large d, the main bottleneck (apart from code complexity, precision issues, etc.) of such problems is on the one hand the scaling $N^{-k/d}$, but much more that a refinement at given number of points N requires an increase by 2^d. For $d = 20$ this amounts to $2^{20} = 1048576$, and $d = 20$ is not particularly big. Imagine for instance a statistical

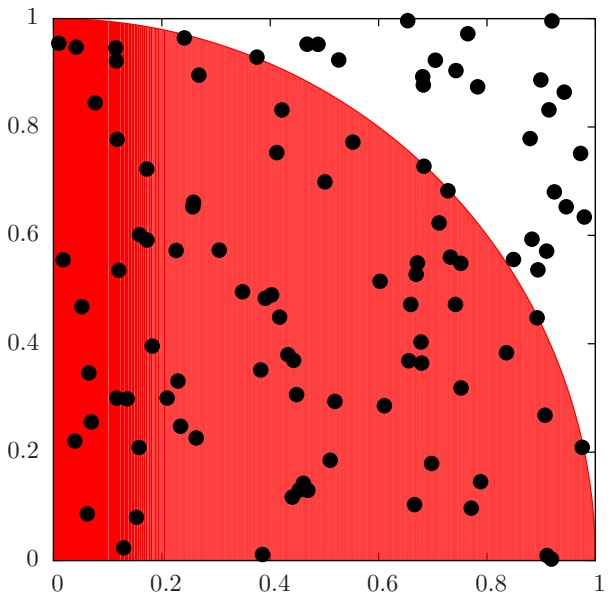


Figure 1.0.1.:
Illustration of example 1.0.1 on how to determine π using a stochastic method.

description of the interaction of the approximately $N = 4 \times 10^{22}$ molecules in one cubic centimetre of nitrogen with a phase space integral dimension of $6N$.

This is why so-called *Monte-Carlo* methods play a very important role in solving such problems. Monte-Carlo methods are stochastic methods. Such methods have, as we will see in chapter 2, by the virtue of the central limit theorem an error scaling of $o(N^{-1/2})$ under very mild restrictions, independent of the dimension and the complexity of the problem for large enough N. Thus, Monte-Carlo beats a grid-based method of order k if

$$k/d \; < \; 1/2\,.$$

However, the price is that the convergence is now to be understood in a probabilistic sense, i.e. there is no strict error bound, as we will see later on.

Example 1.0.1

One well known example of a stochastic method is the computation of π. Consider a circle with radius R and enclose it in a square with side length $2R$. The ratio of areas of square and circle is then given by

$$\frac{A_{\text{circle}}}{A_{\text{square}}} = \frac{\pi R^2}{4R^2} \; = \; \frac{\pi}{4}$$

independent of R. Solving for π gives

$$\pi \;=\; \frac{4A_{\text{circle}}}{A_{\text{square}}}\,.$$

Let us now place N random points uniformly on the square and count how many of those lie in the circle, which we denote with N_{circle}. This is illustrated in Figure 1.0.1 for the upper right quadrant. If the points are really drawn uniformly on the square, it is easy to see that the ratio N_{circle}/N should approximate the ratio of areas, namely $A_{\text{circle}}/A_{\text{square}}$. Hence, we get an estimate for π from

$$\pi \;\approx\; \frac{4N_{\text{circle}}}{N}\,.$$

It is also intuitively clear that with increasing N this approximation must become better and better, because the points become denser and denser over the square.

It will be part of this lecture to formalise this relation and, in particular, to derive the result that the scaling in N is like $N^{-1/2}$.

As said before the $N^{-1/2}$ scaling is independent of the complexity of the problem. This means also that there is no benefit if more restrictions can be put on the problem, i.e. if more properties of the problem are known. One approach to circumvent this and to obtain better scaling in N are so-called *Quasi Monte-Carlo* methods, which we will discuss in chapter 5.

Problems in Statistical Physics and Quantum Fieldtheory

The independence of Monte-Carlo method on the knowledge about the problem can also be turned in favour of such methods. Therefore, chapter 6 introducing Markov–chains can be see as the summit of this lecture. Let us again motivate these by means of an example: consider the following Hamiltonian

$$\mathcal{H}(s_1, ..., s_N) \;\stackrel{\text{def}}{=}\; -J \sum_{\langle i,j \rangle} s_i s_j + h \sum_{i=1}^{N} s_i\,. \tag{1.0.2}$$

Here, $s_i \in \{-1, 1\}$ denote so-called spins which are distributed on a regular 2–dimensional grid with $N = n^2$ sites and $\langle i, j \rangle$ denote neighbouring grid points i and j. $J \in \mathbb{R}$ is called the interaction energy and h is an external (magnetic) field. The model is called Ising–model, and there are more general versions of it including e.g. more neighbours in the interaction. The Ising–model is valuable in the description of (anti)-ferromagnets.

Many ferromagnets have the interesting property that there exists a phase of spontaneous magnetisation below a certain temperature. It is actually easy to see that the lowest energetic state of the Hamiltonian above for $h = 0$ is given by all spin equal, if $J > 0$. If one is interested in the thermal properties of this Ising–model and, therefore,

its phase structure, one needs to study the so-called *partition function*

$$\mathcal{Z} = \sum_{s_1,\ldots,s_N} e^{-\frac{\mathcal{H}(s_1,\ldots,s_N)}{k_B T}} = \sum_{s_1,\ldots,s_N} e^{-\beta\mathcal{H}(s_1,\ldots,s_N)}\,, \tag{1.0.3}$$

where we defined $\beta = 1/k_b T$. For computing the partition function Eq. 1.0.3, we have to sum over all 2^N possible states of the Ising–model, which even for moderate values of N is not feasible. However, even if we could achieve this sum, it is clear that we would sum a lot of very small numbers, and only a few relevant ones, simply because the Boltzmann–weight is suppressing most of the states in the sum. In fact, we know from statistical physics that the fluctuations in the energy of the system are given as follows

$$\frac{\langle\mathcal{H}^2\rangle - \langle\mathcal{H}\rangle^2}{\langle\mathcal{H}\rangle^2} \quad \propto \quad \frac{1}{N}\,.$$

Therefore, only states with energy close to the mean value of the energy contribute and, with increasing system size N, the number of such states is decreasing rapidly. Markov–chains implement a solution to this weight problem by generating preferably such states of the system with a large weight, which we will see in chapter 6.

Even though we motivated Markov–chains by means of an example from statistical physics, they find applications in many fields of physics, most notably also in Quantum Fieldtheory and in particular in Quantum Chromodynamics (QCD), the theory of strong interactions.

Data Analysis and Programming

Since we aim at practical simulations, a not so small part of this lecture is devoted to random number generation and to data analysis. Random number generation of arbitrary distributions is mandatory for anyone interested in performing Monte-Carlo analyses and simulations. This topic will be discussed in chapter 3. The tools for data analysis will be introduced in chapter 4. They are needed to analyse Monte-Carlo data. But they will also be useful for any physicist interested in performing state-of-the-art data analysis.

Apart from examples, we also provide practical material to implement the concepts introduced during the lecture. For this purpose we use the programming language `R`, which is very powerful when it comes to statistical analyses. However, it takes little effort to implement the same code in e.g. `python`. The practical parts are enclosed in separate sections, which are part of the relevant chapters. The source files with only the `R`–code are provided in addition to this script. A very short introduction to `R` can be found in the appendix.

References will be provided at the end of each chapter.

2. Probability Theory

In this chapter we summarise basic facts about probability theory. In particular, we will introduce distribution functions and the notion of random variables. We will introduce important characteristics of distributions like expectation value and variance. The final goal of this chapter is the proof of the central limit theorem, which will be the basis of all stochastic methods introduced later on.

Before going to the details, let us shortly repeat some notation and wording from set theory. The empty set is denoted as $\emptyset$ and the power set of a set $\Omega \neq \emptyset$ is denoted as 2^Ω. 2^Ω is the set of all subsets of Ω.

Let now A, B, C be elements of $\mathcal{E} = 2^\Omega$, i.e. subsets of Ω. Then one writes

- $C = A \cup B = A + B$: union or set union, say A or B
- $C = A \cap B = AB$: intersection or intersecting set, say A and B
- $C = \bar{A}$: not A, the complement of A: $\Omega = A + \bar{A}$
- $C = A \setminus B = A\bar{B}$: difference of A and B

2.1. Probability

We all have a very intuitive idea about probability. If you ask someone on the street they will very likely agree to define probability in the following way:

- perform some experiment X N-times in the exact same way
- measure the N outcomes
- if outcome (or *event*) A occurs N_A-times we associate it with probability $P_A = N_A/N$:
- the larger N the more precise we know about P_A

In the following we assume $\Omega \neq \emptyset$ to be a set and $\mathcal{E} = 2^\Omega$ the powerset of Ω. The set Ω will be interpreted as the space of elementary events and $\mathcal{E}$ as a system of observable events. Hence, $A \in \mathcal{E}$ is called an event. We define a map $\mathbb{P} : \mathcal{E} \to \mathbb{R}$ which we call *probability*, if the following axioms (Kolmogorov) hold:

K1. $1 \geq \mathbb{P}(A) \geq 0 \quad \forall A \in \mathcal{E}$
Every event must have some probability. Any event cannot be more likely than the certain event ($\mathbb{P} = 1$) and it cannot be less likely than impossible ($\mathbb{P} = 0$).

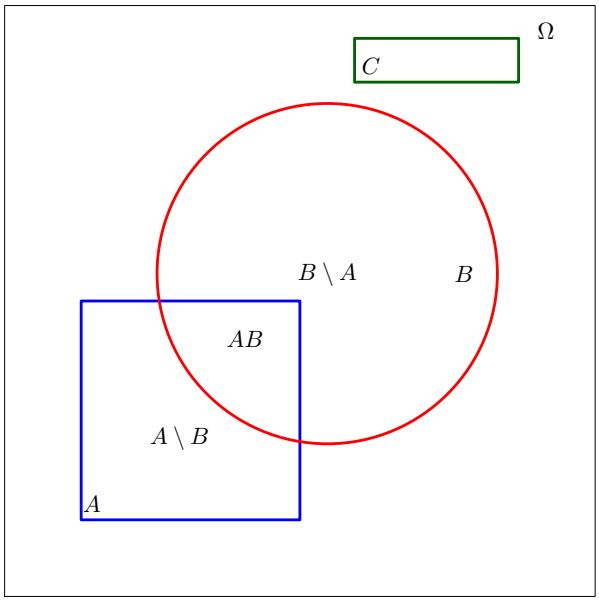


Figure 2.1.1.: Illustration of an event space and some operations on sets.

K2. $\mathbb{P}(\Omega) = 1$
There must not be missing probability, i.e. some event, and let it be the null-event must happen. And, hence, the probability of Ω must be one.

K3. for $A, B \in \mathcal{E}$ with $A \cap B = 0$: $\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B)$
The probability of the union of two events which are exclusive ($A \cap B = 0$) must be equal to the sum of the probabilities of the single events.

Again, KOLMOGOROVs axioms should come natural given our intuition about probability. From the axioms stated above it follows immediately:

1. $\forall A \in \mathcal{E}: \quad \mathbb{P}(\overline{A}) = 1 - \mathbb{P}(A)$
2. $\mathbb{P}(\emptyset) = 0$
3. $\forall A, B \in \mathcal{E}: \quad \mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(AB)$

The three axioms introduced above actually define a LEBESGUE–measure on $\mathcal{E}$ with the additional property of $\mathbb{P}(\Omega) = 1$. This is why the modern way of approaching probability is from measure theory. For this one introduces a probabilityspace

Definition 2.1.1: Probabilityspace

Let Ω be a non-empty set, $\mathcal{E} \subset 2^\Omega$ a σ–algebra and $\mathbb{P}$ a measure on $\mathcal{E}$. The triple $(\Omega, \mathcal{E}, \mathbb{P})$ is called a ***probabilityspace*** *if $\mathbb{P}(\Omega) = 1$.*

For the definition of a σ–algebra see exercise 1 of this chapter. We have already seen that we needed exclusive events, i.e. two elements of 2^Ω which do not have overlap. This is an important classification of events and we define it as follows

Definition 2.1.2: mutually exclusive events

Let $A_1, A_2, \ldots \in \mathcal{E}$ be events for $i = 1, 2, \ldots$. They are called ***mutually exclusive*** *if*

$$A_i \cap A_j \;=\; \emptyset \qquad \forall\; i \neq j\,. \tag{2.1.1}$$

With this definition we could have also formulated the third axiom of KOLMOGOROV as follows:

K3′. For $A_1, \ldots A_n \in \mathcal{E}$ mutually exclusive

$$\mathbb{P}\left(A_1 + A_2 + \ldots + A_n\right) \;=\; \mathbb{P}\left(A_1\right) + \mathbb{P}\left(A_2\right) + \ldots + \mathbb{P}\left(A_n\right)\,. \tag{2.1.2}$$

Another important classification is whether the probability of event A depends in some way on whether or not event B has happened or not.

Definition 2.1.3: independent events

Let $A, B \in \mathcal{E}$ be two events. A and B are called ***independent*** *if*

$$\mathbb{P}(A \cap B) \;=\; \mathbb{P}\left(AB\right) \;=\; \mathbb{P}(A) \cdot \mathbb{P}(B)\,. \tag{2.1.3}$$

Events are not always independent, of course. In case an event is conditional on another event one speaks about conditional probability:

Definition 2.1.4: conditional probability

We define the probability of A conditional on B by the relation

$$\mathbb{P}\left(A|B\right) \;=\; \frac{\mathbb{P}\left(A \cap B\right)}{\mathbb{P}(B)} \;=\; \frac{\mathbb{P}\left(AB\right)}{\mathbb{P}(B)}\,, \tag{2.1.4}$$

if $\mathbb{P}\left(B\right) \neq 0$ and $\mathbb{P}\left(A|B\right) = 0$, if $\mathbb{P}\left(B\right) = 0$. Conditional probabilities have the following properties that can be checked using the definition:

1. $\mathbb{P}(A|B)\mathbb{P}(B) = \mathbb{P}(A \cap B) = \mathbb{P}(B|A) \cdot \mathbb{P}(A)$
2. $\mathbb{P}(\bar{A}|B) = 1 - \mathbb{P}(A|B)$
3. $\mathbb{P}(A|\bar{B}) = \frac{\mathbb{P}(A) - \mathbb{P}(B)\mathbb{P}(A|B)}{1 - \mathbb{P}(B)}$
4. if A and B are independent, then

$$\begin{aligned} \mathbb{P}(A|B) &= \mathbb{P}(A) \\ \mathbb{P}(B|A) &= \mathbb{P}(B)\,. \end{aligned}$$

Example 2.1.1

Consider three classical spins with possible directions up $\uparrow$ and down $\downarrow$. The state space Ω is given by

$$\Omega \;=\; \{\uparrow\uparrow\uparrow,\; \uparrow\uparrow\downarrow,\; \uparrow\downarrow\uparrow,\; \downarrow\uparrow\uparrow,\; \uparrow\downarrow\downarrow,\; \downarrow\uparrow\downarrow,\; \downarrow\downarrow\uparrow,\; \downarrow\downarrow\downarrow\}$$

containing $2^3 = 8$ states. Therefore, every of these states has probability 1/8. The conditional probability of states in Ω with A two or more spins pointing up given B the first spin is pointing up is given by

$$\mathbb{P}\left(A|B\right) \;=\; \frac{\mathbb{P}\left(A \cap B\right)}{\mathbb{P}\left(B\right)} \;=\; \frac{3/8}{4/8} \;=\; \frac{3}{4}\,.$$

This allows one to formulate the following theorem, which formalises the intuitive fact that if B did not happen, then $\bar{B}$ (not-B) must have happened.

Theorem 2.1.1: Theorem of the total probability

With $A, B \in \mathcal{E}$ we can write the probability of A as

$$\mathbb{P}(A) = \mathbb{P}(A|B)\mathbb{P}(B) + \mathbb{P}(A|\bar{B})\mathbb{P}(\bar{B}) \tag{2.1.5}$$

or more generally, for $A_1, \ldots, A_n \in \mathcal{E}$ events with properties:

1. $A_i \cap A_j = \emptyset \qquad \forall i \neq j \;\in 1, ..., n\,.$
2. $\bigcup_{i=1}^{n} A_i = \Omega\,.$
3. $\mathbb{P}(A_i) > 0 \;\forall i \in 1, ..., n$

then

$$\mathbb{P}(B) \;=\; \sum_{i=1}^{n} \mathbb{P}\left(B|A_i\right) \mathbb{P}\left(A_i\right)\,. \tag{2.1.6}$$

Proof:

$$\begin{aligned} \mathbb{P}(B) \;&=\; \mathbb{P}\left(B\Omega\right) \;=\; \mathbb{P}\left(B \sum_i A_i\right) = \mathbb{P}\left(\sum_i BA_i\right) \;=\; \sum_i \mathbb{P}(BA_i) \\ &= \sum_i \mathbb{P}\left(B|A_i\right) \mathbb{P}\left(A_i\right) \end{aligned} \tag{2.1.7}$$

□

Example 2.1.2

Continuing example 2.1.1 with the same events A and B as defined there. We have

$$\mathbb{P}(A|B)\,\mathbb{P}(B) \;=\; 3/4 \cdot 4/8 \;=\; 3/8$$

and

$$\mathbb{P}\left(A|\bar{B}\right)\mathbb{P}(\bar{B}) \;=\; 1/8$$

which sum to $4/8$. Four of the states in Ω have two or more spins pointing up, hence $\mathbb{P}(A) = 4/8$.

With the definition of conditional probability at hand we can also proof the following famous result of Bayes:

Theorem 2.1.2: Theorem of Bayes

$$\mathbb{P}(A_j|B) \;=\; \mathbb{P}(B|A_j)\,\frac{\mathbb{P}(A_j)}{\mathbb{P}(B)}\,. \tag{2.1.8}$$

Proof:
We have from the definition of conditional probability

$$\mathbb{P}(A_j|B) \;=\; \frac{\mathbb{P}(A_j \cap B)}{\mathbb{P}(B)} \;=\; \frac{\mathbb{P}(B \cap A_j)}{\mathbb{P}(B)}\,. \tag{2.1.9}$$

Using the definition of conditional probability again for $\mathbb{P}(B|A_j) = \mathbb{P}(BA_j)/\mathbb{P}(A_j)$, we find

$$\mathbb{P}(A_j|B) \;=\; \frac{\mathbb{P}(B|A_j)\,\mathbb{P}(A_j)}{\mathbb{P}(B)}\,. \tag{2.1.10}$$

Finally we could also apply theorem 2.1.1 for $\mathbb{P}(B) = \sum_{i=1}^{n} \mathbb{P}(B|A_i)\,\mathbb{P}(A_i)$ in the denominator.

□

In the context of Bayesian-statistics $\mathbb{P}(A_j)$ is called *prior probability*, $\mathbb{P}(A_j|B)$ *posterior probability*, $\mathbb{P}(B)$ evidence and $\mathbb{P}(B|A_j)$ *likelihood*.

2.2. Random Variables and Distribution Functions

We consider now pairs $(\Omega, \mathcal{E})$ and specifically $(\Omega' = \mathbb{R}, \mathcal{E}' = \mathcal{B}(\mathbb{R}))$ with $\mathcal{B}(\mathbb{R})$ the Borel σ-algebra on $\mathbb{R}$. The Borel σ-algebra on $\mathbb{R}$ is Lebesgue–measurable and contains all open, closed and half-open intervals, as well as all finite and countable unions of intervals in $\mathbb{R}$. $(\mathbb{R}, \mathcal{B}(\mathbb{R}), \mu)$ define a standard probability space. In this way we will map the (arbitrary) event space Ω to the real numbers $\mathbb{R}$, which will allow us to define random variables. Formally, we define a

Definition 2.2.1: Measurable Mapping

A mapping $X : \Omega \to \Omega' = \mathbb{R}$ is called $\mathcal{B}(\mathbb{R})$-measurable or simply measurable, if $X^{-1}(\mathcal{E}') \stackrel{def}{=} \{X^{-1}(A') : A' \in \mathcal{E}'\} \subset \mathcal{E}$, or

$$X^{-1}(A') \in \mathcal{E} \qquad \forall\ A' \in \mathcal{B}(\mathbb{R})\,. \tag{2.2.1}$$

Such a measurable map will be what we will call a random variable. We will concentrate on real, uni-variate random variables here. The generalisation to the multi-variate case is straightforward. In what follows we will always assume a probability space $(\Omega, \mathcal{E}, \mathbb{P})$. $\mathbb{P}$ will be interpreted as probability and $A \in \mathcal{E}$ is called an event. First we define a random variable:

Definition 2.2.2: Random Variable

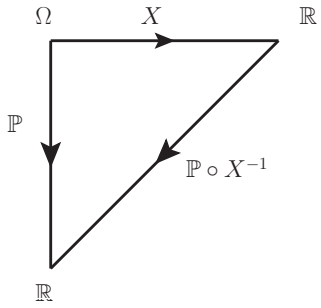
Let $X : \Omega \to \mathbb{R}$ be a measurable map. X is called a real random variable with values in $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$.

The cumulative distribution function of such a random variable is then defined as follows:

Definition 2.2.3: cumulative distribution function (cdf)

Let X be a real random variable.

- *$\mathbb{P}_X \stackrel{def}{=} \mathbb{P} \circ X^{-1}$ is called distribution of X.*



- *The map $F_X : x \mapsto \mathbb{P}(X \leq x)$ is called the cumulative distribution function (cdf) of X (more precisely of $\mathbb{P}_X$).*
- *Two random variables X_1, X_2 are identically distributed if $\mathbb{P}_{X_1} = \mathbb{P}_{X_2}$.*

F_X is representing the probability for a random variable X to assume values smaller or equal to x. Every cdf is monotonous, non-decreasing and right continuous. From the definition it follows immediately

$$\begin{aligned} \mathbb{P}\left(a \leq X \leq b\right) &= F_X(b) - F_X(a-0)\,, \\ \mathbb{P}\left(a < X \leq b\right) &= F_X(b) - F_X(a)\,, \\ F_X(-\infty) &= 0\,, \\ F_X(+\infty) &= 1\,. \end{aligned} \tag{2.2.2}$$

Here we indicate with $F_X(a-0)$ the limit of $x \nearrow a$. In the continuous case we may differentiate $F_X(x)$. This leads to the following definition

Definition 2.2.4: probability density function

For continuous and differentiable cdf F_X

$$f_X(x) \stackrel{def}{=} \frac{\mathrm{d}F_X(x)}{\mathrm{d}x} \tag{2.2.3}$$

is called the probability density function (pdf) of X. Written in integral form we have

$$F_X(x) = \int_{-\infty}^{x} f_X(x')\,\mathrm{d}x', \quad \Leftrightarrow \quad F_X(x) = \mathbb{P}_X(X \leq x). \tag{2.2.4}$$

In the discrete case we set

$$f_X(x) \stackrel{def}{=} \mathbb{P}_X(X = x), \tag{2.2.5}$$

which is sometimes also called probability mass function (pmf).

Per definition, every pdf must fulfil $f_X(x) \geq 0$ for all $x \in \mathbb{R}$ and

$$\int_{-\infty}^{+\infty} f_X(x)\,\mathrm{d}x = 1. \tag{2.2.6}$$

These concepts can be generalised to the multi-variate (multi-dimensional) case with vectors of random variables $\vec{X} = (X_1, X_2, X_3, \ldots, X_n)$. In this case the distribution functions are defined as follows:

$$\begin{aligned} \text{cdf:} \quad & F_{\vec{X}}(\vec{x}) = \mathbb{P}_{\vec{X}}(X_1 \leq x_1,\ X_2 \leq x_2,\ \ldots,\ X_n \leq x_n), \\ \text{pdf:} \quad & f_{\vec{X}}(\vec{x}) = \frac{\partial^n F_{\vec{X}}(\vec{x})}{\partial x_1 \ldots \partial x_n}. \end{aligned}$$

Integrating out one or several of the random variables of some multi-variate cdf $F_{\vec{X}}$ leads to what is called a *marginal distribution*. For instance in the two-dimensional case the marginal distribution in the first random variable is obtained by

$$F_1(x_1) = \int_{-\infty}^{x_1} \mathrm{d}x_1' \int_{-\infty}^{\infty} \mathrm{d}x_2' f_{(X_1,X_2)}(x_1', x_2').$$

Correspondingly, for the pdf one obtains

$$f_1(x_1) = \int_{-\infty}^{\infty} \mathrm{d}x_2' f_{(X_1,X_2)}(x_1', x_2').$$

In general, we cannot reconstruct $f_{\vec{X}}$ from a marginal distribution, because by integrating out one or more of the random variables we decided to ignore the corresponding information. More details for the multi-variate case can be found in the literature, see for instance Ref. [3].

Let us come back to the notion of independence. We had already seen that two events are called *independent* if $\mathbb{P}(AB) = \mathbb{P}(A)\mathbb{P}(B)$. We transfer this definition to random variables as follows

Definition 2.2.5: independent random variables

Let I be an index set. The family $(X_i)_{i\in I}$ of random variables is called independent if for every finite $J \subset I$ and for every $A_j \in \mathcal{E}_j$

$$\mathbb{P}\left(\bigcap_{j\in J}\{X^{-1}(A_j)\}\right) \;=\; \prod_{j\in J}\mathbb{P}\left(X^{-1}(A_j)\right)\,, \tag{2.2.7}$$

i.e. the probability factorises.

As a consequence, the cdf of independent random variables $\vec{X} = (X_1, X_2, ..., X_n)$ factorises as well

$$F_{\vec{X}}(\vec{x}) \;=\; \prod_{j=1,\dots,n} F_{X_j}(x_j) \tag{2.2.8}$$

or for the pdfs

$$f_{\vec{X}}(\vec{x}) \;=\; \prod_{j=1,\dots,n} f_{X_j}(x_j)\,. \tag{2.2.9}$$

We say that a family $(X_i)_{i\in I}$ of independent random variables is *independent and identically distributed* (**iid**) if $\mathbb{P}_{X_j} = \mathbb{P}_{X_i}$ for all i, j in I.

Example 2.2.1

Consider an experiment with an electron beam. We measure the spin of the electrons by some technique. Lets assume the probability for finding spin up is p and for spin down $q = 1 - p$ with $0 < p < 1$. (For an unbiased electron source we would expect $p = q$.) We are interested in the number of spin ups k in n measurements. Let us now model spin up in the ith measurement with $X_i = 1$ and spin down in the ith measurement with $X_i = 0$. X_i represents a random variable for which we have

$$\mathbb{P}(X_i = 1) \;=\; p \;=\; 1 - \mathbb{P}(X_i = 0) = 1 - q\,, \qquad i = 1, \dots, n\,.$$

And, of course, all X_i $i = 1, \dots, n$ are independent. Set

$$X = \sum_{i=1}^{n} X_i$$

the total number of spins pointing up. X has values in $0, 1, \dots, n$. Then we obtain

$$\mathbb{P}(X = k) \;=\; \binom{n}{k} p^k (1-p)^{n-k}$$

for $k = 0, 1, \dots, n$. This is because an n measurements with k–times spin up and $n - k$–times spin down has probability

$$p^k (1-p)^{n-k}$$

and there are n choose k possible combinations of n spins. In other words

$$X \sim \text{Bin}(n,p)$$

with $\text{Bin}(n,p)$ the binomial distribution.

We do obtain the same result if we consider a free gas of n identical, distinguishable and non-interacting particles, which can be in state one with probability p and in state two with probability $1-p$. In quantum mechanics, however, indentical particles are not distinguishable. If we assume the particles to be bosons, the total state of the n particle system must be symmetric under exchange of any two particle (labels). Hence, there are $n+1$ ways to distribute the n particles in the two possible states. The probability of a state with k bosons in state one is given by

$$\mathbb{P}(X=k) = \frac{1}{\sum_{l=0}^{n} p^l (1-p)^{n-l}} \, p^k (1-p)^{n-k} .$$

For $p = 1-p = 1/2$ this reduces to

$$\mathbb{P}(X=k) = \frac{1}{\sum_l p^n} \, p^n = \frac{1}{n+1}$$

independent of k. In this case we find

$$\mathbb{E}(X) = \frac{1}{\sum_l p^n} \sum_k k \, p^n = \frac{1}{n+1} \frac{n(n+1)}{2} = \frac{n}{2} .$$

2.3. Functions of Random Variables

We consider now a continuous, differentiable function $h : \mathbb{R} \to \mathbb{R}$ and interpret $Y = h(X)$ as a new random variable. In order to learn about the distribution of Y we have to consider several cases:

(i) let $h'(x) > 0 \; \forall \, x \in \mathbb{R}$. In this case the inverse of h exists and one finds

$$\mathbb{P}(Y \le y) = \mathbb{P}(X \le x) = \mathbb{P}\left(X \le h^{-1}(y)\right) . \tag{2.3.1}$$

Therefore, the cdf of Y is given as

$$F_Y(y) = F_X\left(h^{-1}(y)\right) . \tag{2.3.2}$$

The pdf is obtained by differentiating F_Y with respect to y

$$\begin{aligned} f_Y(y) &= \frac{\mathrm{d}x}{\mathrm{d}y} \frac{\mathrm{d}}{\mathrm{d}x} F_X(x) \Big|_{x=h^{-1}(y)} = f_X\left(h^{-1}(y)\right) \frac{1}{h'(h^{-1}(y))} \\ &= \left(\frac{\mathrm{d}}{\mathrm{d}y} h^{-1}(y)\right) \, f_X\left(h^{-1}(y)\right) . \end{aligned} \tag{2.3.3}$$

(ii) let $h'(x) < 0 \; \forall \, x \in \mathbb{R}$. In this case we make use of $\mathbb{P}_Y\left(Y \leq y\right) = \mathbb{P}\left(X \geq x\right) = 1 - \mathbb{P}\left(X < x\right)$ which leads to the cdf

$$F_Y(y) \;=\; 1 - F_X\left(h^{-1}(y)\right) \tag{2.3.4}$$

and the pdf

$$f_Y(y) \;=\; -\frac{\mathrm{d}}{\mathrm{d}y}h^{-1}(y)\, f_X\left(h^{-1}(y)\right) \;=\; \left|\frac{\mathrm{d}}{\mathrm{d}y}h^{-1}(y)\right| \; f_X\left(h^{-1}(y)\right) . \tag{2.3.5}$$

The absolute value comes from the fact that $h'(x) < 0$. Hence, the latter equation combines cases (i) and (ii).

(iii) in case h is non-monotonous one needs to consider several branches with definite sign of $h'(x)$ for which the cases discussed above can be applied.

The generalisation to the multi-variate case, i.e. for $\vec{Y} = \vec{h}(\vec{X})$ under the assumption that h is invertible reads

$$f_{\vec{Y}}(\vec{y}) \;=\; \left|D\vec{h}^{-1}(\vec{y})\right| \; f_{\vec{X}}\left(\vec{h}^{-1}(\vec{y})\right) \tag{2.3.6}$$

with Df the Jacobi–matrix of f.

It is useful to apply these developments to the case of $Y = h(X_1, X_2, ..., X_n)$. For this we define the following vector valued function

$$\vec{h} : \vec{X} \mapsto \vec{Y} = \vec{h}(\vec{X}) \tag{2.3.7}$$

with $Y_1 = h(X_1, X_2, ..., X_n)$ and $Y_i = X_i$ for $n \geq i > 1$. The pdf of Y is then obtained by computing the marginal distribution

$$\begin{aligned} f_Y(y) \;&=\; \int \mathrm{d}y'_1 ... \mathrm{d}y'_n \; f_{\vec{X}}(h^{-1}(\vec{y}), y'_2, ..., y'_n) \left|D\vec{h}^{-1}(\vec{y})\right| \; \delta(y'_1 - y) \\ &=\; \int \mathrm{d}x'_1 ... \mathrm{d}x'_n \; f_{\vec{X}}(x'_1, ..., x'_n)\, \delta(y - h(x'_1, ..., x'_n)) \, . \end{aligned} \tag{2.3.8}$$

Here, we have made use of the fact that the determinant of $D\vec{h}^{-1}$ can be computed easily to

$$\left|D\vec{h}^{-1}(\vec{y})\right| \;=\; \frac{\partial h^{-1}(\vec{y})}{\partial y_1} \tag{2.3.9}$$

where $h^{-1}(\vec{y})$ denotes the solution of $y_1 = h(x_1, y_2, ..., y_n)$ with respect to x_1. Moreover, we have fixed $x_1 = y$ using the δ-distribution.

Example 2.3.1

Let X be a random variable distributed uniformly in $[0, 1] \subset \mathbb{R}$, i.e.

$$f_X(x) = \mathcal{U}_{[0,1]} \;=\; \begin{cases} 1 & 0 \leq x \leq 1 \, , \\ 0 & \text{else} \, . \end{cases} \tag{2.3.10}$$

We also write $X \sim \mathcal{U}_{[0,1]}$. Define $Y = h(X) = X^2$. Then we have with $x = \sqrt{y}$ $(x > 0)$

$$\frac{\partial h^{-1}}{\partial y} = \frac{1}{2\sqrt{y}} \tag{2.3.11}$$

and, hence,

$$f_Y(y) = \frac{1}{2\sqrt{y}} f_X(x) = \begin{cases} \frac{1}{2\sqrt{y}} & 0 \le y \le 1\,, \\ 0 & \text{else}\,. \end{cases} \tag{2.3.12}$$

Of particular relevance is the case where a new random variable is constructed as the sum of other ones. We can directly apply the formalism developed above as follows: let X_1, X_2 be independent random variables and define $Y = h(X_1, X_2) = X_1 + X_2$. Then, $\partial h/\partial X_1 = \partial h/\partial X_2 = 1$ and we obtain the pdf by the convolution integral Eq. 2.3.8

$$\begin{aligned} f_Y(y) &= \int \mathrm{d}x_1' \, \mathrm{d}x_2' \; f_{X_1}(x_1') f_{X_2}(x_2') \; \delta(y - x_1' - x_2') \\ &= \int \mathrm{d}x_1' \; f_{X_1}(x_1') f_{X_2}(y - x_1')\,. \end{aligned} \tag{2.3.13}$$

Example 2.3.2

Take $X_i \sim U_{[0,1]}$ **iid** , then we obtain

$$\begin{aligned} f_Y(y) &= \int \mathrm{d}x_1' \; f_{X_1}(x_1') f_{X_2}(y - x_1') = \int_0^1 \mathrm{d}x' f_X(y - x') \\ &= \int_{y-1}^{y} \mathrm{d}\xi \; \Theta\left(\xi(1-\xi)\right) \end{aligned} \tag{2.3.14}$$

with $\xi = y - x'$ and the HEAVISIDE-function Θ. This can be further resolved to

$$f_Y(y) = \begin{cases} y & 0 \le y \le 1\,, \\ 2 - y & 1 \le y \le 2\,, \\ 0 & \text{else}\,. \end{cases} \tag{2.3.15}$$

2.4. Characteristic Measures of Distributions

Next we define measures to characterise distributions. We first define what we call an outcome of a (infinitely often) repeated random experiment, i.e. a random variable X. We are interested in the average outcome, but also in the fluctuation. First we define the expectation value:

Definition 2.4.1: Expectation Value $\mathbb{E}$

Let X be a random variable with pdf $f_X(x)$. The expectation value or mean of X: $\mathbb{E}(x) \equiv \mu$ is a linear mapping $\mathbb{E} : \mathbb{R} \to \mathbb{R}$ defined via

- *in the continuous case as*

$$\mathbb{E}(X) \stackrel{def}{=} \int x \, \mathrm{d}F_X = \int x \, f_X(x) \, \mathrm{d}x \,. \tag{2.4.1}$$

- *in the discrete case as*

$$\mathbb{E}(X) \stackrel{def}{=} \sum_x x \, f_X(x) \,. \tag{2.4.2}$$

We will also write $\langle X \rangle$ for the expectation value of X.

This definition extends to functions of random variables h, namely e.g. in the continuous case to

$$\mathbb{E}(h(X)) = \langle h(X) \rangle = \int h(x) \, f_X(x) \, \mathrm{d}x \,. \tag{2.4.3}$$

Example 2.4.1

In physics the expectation value is used frequently in quantum mechanics (QM), and then in any quantum theory. A quantum state is described by a wave function, a complex valued, square integrable function

$$\psi(x) \in L^2 \,.$$

$f(x) = \psi(x)\psi^\star(x) \geq 0$ is interpreted as probability density leading to the normalisation

$$\int \psi(x)\psi^\star(x) \, \mathrm{d}x = 1 \,.$$

The expectation value of a physical operator $O(x)$ is then given by Eq. 2.4.1

$$\mathbb{E}(O) = \int O(x) \; \psi(x)\psi^\star(x) \, \mathrm{d}x \,.$$

Using the definition of the expectation value, we can also measure the fluctuation, which we call variance:

Definition 2.4.2: Variance and Standard Deviation

The variance of a random variable X is defined as:

$$\mathrm{Var}(X) \stackrel{def}{=} \mathbb{E}\left((X - \mathbb{E}(X))^2\right) = \mathbb{E}\left(X^2\right) - \mathbb{E}(X)^2 \stackrel{def}{=} \sigma^2 \tag{2.4.4}$$

$\mathrm{sd}(X) \overset{def}{=} \sigma = +\sqrt{\sigma^2}$ *is called the standard deviation.*

Expectation value and variance are the most commonly used characteristics of distributions. But they are certainly not sufficient to uniquely characterise all distributions. Moments can be used to characterise a distribution

Definition 2.4.3: n-th moment

The n-th moment $\mu^{(n)}$ *of a distribution* f_X *is defined as*

$$\mu^{(n)} \;=\; \langle X^n \rangle = \int \mathrm{d}x\, x^n f_X(x)\,. \tag{2.4.5}$$

The first few moments are given by

$$\mu^{(0)} \;=\; 1\,; \quad \mu^{(1)} \;=\; \mu\,; \quad \mu^{(2)} \;=\; \sigma^2 + \mu^2 \ldots \tag{2.4.6}$$

The median $\mu_{1/2}$ is the x-value, which cuts the area underneath the distribution f_X in halfs. For the interval $I = [a, b]$ it is defined by

$$\int_a^{\mu_{1/2}} f_X(x)\mathrm{d}x \;=\; \int_{\mu_{1/2}}^{b} f(x)\mathrm{d}x = \frac{1}{2}\,. \tag{2.4.7}$$

This is a special case of the so-called α–quantile

Definition 2.4.4: α–quantile

The α*–quantile* μ_α *of a distribution* f_X *is defined by*

$$F\left(\mu_\alpha\right) \;=\; \int_a^{\mu_\alpha} f_X(x)\mathrm{d}x = \alpha\,. \tag{2.4.8}$$

For a multi-variate distribution $f_{\vec{X}}$ of random variables $\vec{X}$ one defines analogously

$$\mu_i = \langle X_i \rangle = \int \mathrm{d}^n x\, x_i f\left(x_1, \ldots, x_n\right)\,,$$
$$\langle X_i X_j \rangle = \int \mathrm{d}^n x\, x_i x_j f\left(x_1, \ldots, x_n\right)\,.$$

For the case of multi-variate distributions there appear mixed moments. They will let one conclude on possible correlation between separate random variables, as we will see. We fist define the *covariance*:

Definition 2.4.5: Covariance

The covariance of two random variables X_1 and X_2 is defined as

$$\mathrm{Cov}\,(X_1, X_2) \stackrel{def}{=} \langle (X_1 - \mu_1)(X_2 - \mu_2) \rangle \,. \tag{2.4.9}$$

The covariance has the following decomposition

$$\mathrm{Cov}(X, Y) = \mathbb{E}(XY) - \mathbb{E}(X)\mathbb{E}(Y) \tag{2.4.10}$$

as one can show easily and, per definition,

$$\mathrm{Cov}(X, X) = \mathrm{Var}(X)\,. \tag{2.4.11}$$

Theorem 2.4.1:

The matrix $\mathrm{Cov}(X_i, X_j) = \mathrm{Cov}_{i,j}$ is symmetric $n \times n$ and positive semi–definite.

Proof:
Symmetric $n \times n$ is clear. Positive semi–definite can be seen by writing the covariance matrix in vectorial form using the usual matrix multiplication

$$\mathrm{Cov} \;=\; \langle (X - \mu) \cdot (X - \mu)^t \rangle \;\equiv\; \langle Z \cdot Z^t \rangle$$

with $X = (X_1, \ldots, X_n)^t$, $\mu = (\mu_1, \ldots, \mu_n)^t$ and $Z = X - \mu$. An $n \times n$ matrix Σ is positive semi–definite if and only if $w^t \Sigma w \geq 0$ for all $w \in \mathbb{R}^n$. Compute

$$w^t\,\mathrm{Cov}\,w \;=\; \langle w^t Z Z^t w \rangle \;=\; \langle (w^t Z)(Z^t w) \rangle \;=\; \langle (Z^t w)^t (Z^t w) \rangle \;=\; \langle v^2 \rangle \;\geq 0$$

with $v = Z^t w \in \mathbb{R}$.

□

A scaled version of the covariance matrix is given by the correlation matrix

$$\mathrm{Cor}\,(X_i, X_j) \;=\; \frac{\mathrm{Cov}\,(X_i, X_j)}{\sigma_i \sigma_j}\,. \tag{2.4.12}$$

For independent random variables X_1, X_2 the joint pdf factorises and, hence,

$$\mathrm{Cov}(X_1, X_2) = 0\,,$$

which is, in turn, a necessary condition for independence, but not a sufficient one.

A useful concept are the so-called characteristic functions, which we will need to proof the central limit theorem. They are defined as follows

Definition 2.4.6: Characteristic Function G_X

The characteristic function $G_X(k)$ of a distribution f_X is defined as

$$G_X(k) \;=\; \left\langle \mathrm{e}^{\mathrm{i}kX} \right\rangle = \int \mathrm{d}x\, \mathrm{e}^{\mathrm{i}kx}\, f_X(x)\,. \tag{2.4.13}$$

With this definition we can re-write the moments if all $\mu^{(n)}$ do exist as follows

$$\langle X^n \rangle \;=\; \frac{1}{\mathrm{i}^n} \frac{\mathrm{d}^n G(k)}{\mathrm{d}k^n}\Big|_{k=0} \tag{2.4.14}$$

using the series representation of the exponential function

$$G_X(k) \;=\; \sum_{n=0}^{\infty} \frac{\mathrm{i}^n}{n!} k^n \langle X^n \rangle \tag{2.4.15}$$

Taking the logarithm of $G(k)$ leads to

$$\ln\left(G_X(k)\right) = \sum_{n=1}^{\infty} \frac{(\mathrm{i}k)^n}{n!} \kappa_n$$

with so-called *cumulants* κ_n, with the first two being identified with

$$\kappa_1 \;=\; \mu\,; \quad \kappa_2 \;=\; \sigma^2\,. \tag{2.4.16}$$

Cumulants represent an alternative, but equivalent way to characterise distributions. With the cumulants we can re-write the characteristic function as follows:

$$G_X(k) \;=\; \mathrm{e}^{\mathrm{i}k\mu - \frac{k^2}{2}\sigma^2 + \mathcal{O}\left(k^3\right)}\,. \tag{2.4.17}$$

The following two theorems for characteristic functions will be needed in the proof in the next section:

Theorem 2.4.2:

Let $\vec{X} = (X_1, X_2, X_3, ..., X_n)$ be a random vector with independent components. Then the characteristic function factorises:

$$G_{\vec{X}}\left(\vec{k}\right) \;=\; \prod_i G_{X_i}\left(k_i\right)\,. \tag{2.4.18}$$

Proof:
Since the components are independent, the pdf factorises and, therefore,

$$\begin{aligned} G_{\vec{X}}\left(\vec{k}\right) &= \int \mathrm{d}^n x\, \mathrm{e}^{\mathrm{i}\vec{k}\vec{x}} f_{\vec{X}}\left(x_1, \ldots, x_n\right) \\ &= \int \mathrm{d}^n x\, \mathrm{e}^{\mathrm{i}k_1 x_1} \mathrm{e}^{\mathrm{i}k_2 x_2} \ldots f_{X_1}\left(x_1\right) f_{X_2}\left(x_2\right) \ldots \\ &= G_{X_1}(k_1) \ldots G_{X_n}(k_n)\,. \end{aligned}$$

□

Moreover,

Theorem 2.4.3:

Let X, Y be two independent random variables and $Z = X + Y$. Then

$$G_Z(k) \;=\; G_X(k) \;\cdot\; G_Y(k) \tag{2.4.19}$$

Proof:
Since X, Y are independent, $f_{X,Y}(x,y) = f_X(x) \times f_Y(y)$ and f_{X+Y} is given by the convolution of f_X and f_Y

$$\begin{aligned} f_{X+Y}(z) &= \int f_X(x)\, f_Y(y)\, \delta(z-x-y)\, \mathrm{d}x\, \mathrm{d}y \\ &= \int f_X(z-y)\, f_Y(y)\, \mathrm{d}y\,. \end{aligned}$$

The characteristic function for distribution with pdf f_{X+Y} is then given by

$$\begin{aligned} G_{X+Y}(k) &= \mathbb{E}_{X+Y}\left(\exp(ikZ)\right) \\ &= \int f_X(z-y)\, f_Y(y)\, \mathrm{d}y\, \exp(ikz)\, \mathrm{d}z \\ &= \int f_X(x)\, f_Y(y)\, \mathrm{d}y\, \exp(ik(x+y))\, \mathrm{d}x \\ &= \mathbb{E}_X\left(\exp(ikX)\right)\, \mathbb{E}_Y\left(\exp(ikY)\right) \\ &= G_X(k)\, G_Y(k)\,. \end{aligned}$$

Since the factorization holds for two independent variables, it also holds for any finite set of independent variables.

□

Now we have all the prerequisites available to proof the important central limit theorem.

2.5. Law of Large Numbers

The law of large numbers puts our intuition that the arithmetic average converges to the expectation value on firm grounds. To see this, we first introduce the important concept of stochastic convergence, which is fundamentally different from what we usual understand under convergence.

Definition 2.5.1: Stochastic Convergence

A series Z_n of real random variables is said to ***almost–surely*** *to $c \in \mathbb{R}$ if*

$$\mathbb{P}\left(\lim_{n\to\infty} Z_n = c\right) = 1\,. \tag{2.5.1}$$

We will also write

$$\lim_{n\to\infty} Z_n \overset{a.s.}{=} c\,, \quad or \quad Z_n \overset{a.s.}{\to} c \tag{2.5.2}$$

for short. A series Z_n of real random variables is said to ***converge in probability*** *to $c \in \mathbb{R}$ if*

$$\lim_{n\to\infty} \mathbb{P}\left(|Z_n - c| > \epsilon\right) \;=\; 0\,. \tag{2.5.3}$$

Using characteristic functions we may proof the following results, known as the weak law of large numbers.

Theorem 2.5.1: Weak law of large numbers

Let $X_1, X_2, ...$ be a sequence of **iid** random variables with finite $\mathbb{E}(X_1) = \mu$ and

$$S_N \;=\; \frac{X_1 + ... + X_N}{N} \;=\; \frac{1}{N}\sum_{n=1}^{N} X_n\,. \tag{2.5.4}$$

Then

$$\lim_{N\to\infty} \mathbb{P}\left(|S_N - \mu| > \epsilon\right) \;=\; 0\,. \tag{2.5.5}$$

We say it converges in probability.

Proof:

The characteristic function G_X of any random variable can be written using a Taylor expansion as

$$G_X(k) = 1 + ik\mu + \mathcal{O}(k^2)$$

for $k \to 0$. Now we have

$$G_{\bar{S}_N}(k) \;=\; \left[G_X\left(\frac{k}{N}\right)\right]^N \;=\; \left[1 + i\mu\frac{k}{N} + \mathcal{O}\left(\frac{k^2}{N^2}\right)\right]^N \;\to\; e^{ik\mu}\,,$$

which is the characteristic function of a constant random variable.

□

Next, we quote a stronger version of this theorem without proof. The proof can be found in the literature.

Theorem 2.5.2: Strong law of large numbers

Let $X_1, X_2, ...$ be a sequence of **iid** random variables with $\mathbb{E}(X_1) = \mu$. Then

$$S_N = \frac{X_1 + ... + X_N}{N} = \frac{1}{N}\sum_{n=1}^{N} X_n \tag{2.5.6}$$

converges almost–surely towards μ.

Of course, the strong law of large numbers implies the weak law. Depending on the assumptions made, there are different versions of the laws of large numbers known. The weak law can be proven for cases with less assumptions.

2.6. Central Limit Theorem

Next we define the *normal distribution*:

Definition 2.6.1: Normal Distribution $\mathcal{N}_{\mu,\sigma}$

The normal distribution $\mathcal{N}_{\mu,\sigma}$ is defined as

$$f_{\mathcal{N}_{\mu,\sigma}}(x) \stackrel{def}{=} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right). \tag{2.6.1}$$

It has mean μ and variance σ^2. $\mathcal{N}_{0,1}$ is called the standard normal distribution.

$$F_{\mathcal{N}_{\mu,\sigma}}(x) \stackrel{def}{=} \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{x-\mu}{\sqrt{2\sigma^2}}\right)\right) \tag{2.6.2}$$

with the so-called error function

$$\mathrm{erf}(y) = \frac{2}{\sqrt{\pi}} \int_0^y \exp(-z^2)\,\mathrm{d}z\,.$$

The standard normal distribution has the following property

Theorem 2.6.1:

The characteristic function of the standard normal distribution is given by

$$G_{\mathcal{N}_{0,1}}(k) = \exp\left(-\frac{k^2}{2}\right). \tag{2.6.3}$$

Proof:
It is easy to show that all cumulants of $\mathcal{N}_{0,1}$ vanish apart from $\kappa_2 = \sigma^2 = 1$.

□

Now we turn towards the central limit theorem using the concepts introduced in the last section to proof the central limit theorem:

Theorem 2.6.2: Central Limit Theorem

Let $X_1, X_2, \ldots, X_n$ be independent random variables with mean μ_i, variance $\sigma_i^2 < \infty$ and all higher cumulants bounded by a constant $C < \infty$, then for all $x \in \mathbb{R}$

$$\lim_{n\to\infty} \mathbb{P}\left(\frac{1}{\sqrt{n}}\sum_{i=1}^{n}\frac{X_i-\mu_i}{\sigma_i} \leq x\right) \;=\; F_{\mathcal{N}_{0,1}}(x)\,.$$

Proof:
The characteristic function of X_i is given as

$$G_{X_i}(k) \;=\; \mathrm{e}^{\mathrm{i}k\mu_i - \frac{k^2}{2}\sigma_i^2 + \mathcal{O}\left(k^3\right)}\,. \tag{2.6.4}$$

Next we define a standardised variable

$$S_i \;\stackrel{\text{def}}{=}\; \frac{X_i-\mu_i}{\sqrt{n}\sigma_i}\,, \tag{2.6.5}$$

which has characteristic function

$$G_{S_i}(k) \;=\; \mathrm{e}^{-\frac{k^2}{2n}+\mathcal{O}\left(\frac{k^3}{n^{\frac{3}{2}}}\right)}\,, \tag{2.6.6}$$

since the mean is zero and the variance is one. Now let

$$Z_n \;=\; \sum_{i=1}^{n} S_i\,. \tag{2.6.7}$$

The characteristic function of Z_n reads

$$\begin{aligned} G_{Z_n}(k) = \prod_{i=1}^{n} G_{S_i}(k) &= \mathrm{e}^{n\left(-\frac{k^2}{2n}+\mathcal{O}\left(\frac{k^3}{n^{\frac{3}{2}}}\right)\right)} \\ &= \mathrm{e}^{-\frac{k^2}{2}+\mathcal{O}\left(\frac{k^3}{\sqrt{n}}\right)}\,, \end{aligned} \tag{2.6.8}$$

which converges in probability (to be more precise, in distribution) to $G_{\mathcal{N}_{0,1}}(k) = \exp\left(-\frac{k^2}{2}\right)$ in the limit of $n \to \infty$ for all k.

□

Note that we can lift the constraint on the higher cumulants, which we imposed to keep the proof simple.

Example 2.6.1

An immediate application of the central limit theorem is the generation of standard normal distributed random numbers by adding up uniform random numbers $X \sim U_{[0,1]}$. For $n > 1$ we define

$$Z_n \;=\; \frac{1}{\sqrt{n}}\sqrt{12}\left(X_1 + X_2 + \ldots + X_n - \frac{n}{2}\right) \tag{2.6.9}$$

Usually for $n = 12$ this is a good approximation.

2.7. Some important distributions

In this section we provide the details of some important probability distributions. More can be found in any textbook or on the internet, see. e.g. [6].

2.7.1. Discrete Distributions

1. discrete *uniform* distribution for possible values $0, 1, ..., m-1$. The pdf is the following
$$L_m(x) = \frac{1}{m}\,, \qquad x \in \{0, 1, ..., m-1\}\,.$$
Mean and variance are given by
$$\mu \;=\; \frac{m-1}{2}\,, \qquad \sigma^2 \;=\; \frac{m^2-1}{12}\,.$$
And the characteristic function is
$$g(k) \;=\; \frac{1}{m}\sum_{j=0}^{m-1} \mathrm{e}^{\mathrm{i}jk} = \frac{1}{m}\frac{1-\mathrm{e}^{\mathrm{i}lm}}{1-\mathrm{e}^{\mathrm{i}k}}$$

2. The *binomial*–distribution $\mathrm{Bin}(n, p)$ is one of the most important discrete distributions. It describes the probability for success in a sequence of n independent yes/no experiments, each of which yields success with probability p. Its pdf is given as
$$f(k) \;=\; \binom{n}{k} p^k\,(1-p)^{n-k}\;.$$
where k is the number of successes with support on $\{0, 1, ..., n\}$. The mean and variance are given by
$$\mu \;=\; np\,, \qquad \sigma^2 \;=\; n\cdot p\cdot(1-p)\;.$$

3. The Poisson–distribution $\mathrm{Poi}(\lambda)$ describes an arrival counting process with rate $\lambda > 0$ with the number of events appearing in a fixed period of time being independent. The pdf is given by
$$f(k) \;=\; \frac{\lambda^k}{k!}\mathrm{e}^{-\lambda}\,, \qquad k \in \mathbb{N}\,, \lambda > 0\,.$$
Mean and variance are given by
$$\mu \;=\; \lambda\,, \quad \sigma^2 \;=\; \lambda\,.$$

2.7.2. Continuous Distributions

1. uniform distribution $\mathcal{U}_{[\alpha,\beta]}$, $\alpha < \beta$ in the interval $[\alpha, \beta] \subset \mathbb{R}$:

$$f(x) = \frac{1}{\beta - \alpha} \qquad x \in [\alpha, \beta]$$
$$\mu = \frac{\alpha + \beta}{2} \qquad \sigma^2 = \frac{(\beta - \alpha)^2}{12}$$

2. The exponential distribution $\mathrm{Exp}\,(\lambda)$ describes the time between events in a Poisson process, i.e. events occur continuously and independently at a constant rate $\lambda > 0$. Its pdf is given by

$$f(x) \;=\; \lambda \mathrm{e}^{-\lambda x}\,, \qquad x \in \mathbb{R}_+\,. \tag{2.7.1}$$

Mean and variance are given by

$$\mu \;=\; \frac{1}{\lambda}\,, \quad \sigma^2 \;=\; \frac{1}{\lambda^2}\,. \tag{2.7.2}$$

Its characteristic functions reads

$$G(k) \;=\; \frac{\lambda}{\lambda - \mathrm{i}k}\,. \tag{2.7.3}$$

3. Gamma distribution $\mathrm{Gamma}(a, b)$. The pdf is given by

$$\mathrm{Gamma}(\lambda; a, b) \;=\; \frac{b^{-a}\lambda^{a-1}\mathrm{e}^{-b\lambda}}{\Gamma(x)}\,, \tag{2.7.4}$$

with $a, b > 0$. a is called the *shape* and b is called the rate. It has mean and variance

$$\mu \;=\; \frac{a}{b}\,, \qquad \sigma^2 \;=\; \frac{a}{b^2}\,. \tag{2.7.5}$$

2.8. Exercises

1. Let Ω be a non-empty set. A system of sets $\mathcal{E} \subset 2^\Omega$ is called a σ-algebra, if the following three conditions are fulfilled.
   (i) $\Omega \in \mathcal{E}$.
   (ii) $\mathcal{E}$ is stable under complement $(\forall A \in \mathcal{E} \,:\, \bar{A} \in \mathcal{E})$.
   (iii) $\mathcal{E}$ is $\sigma - \cup$ stable. (For any countably infinite set $A_1, A_2, \ldots \in \mathcal{E}$ it holds $\bigcup_{i=1}^{\infty} A_i \in \mathcal{E}$; analogously for $\sigma - \cap$ stable.)

   Show that if $\mathcal{E} \subset 2^\Omega$ is stable under complement, then $\mathcal{E}$ is $\sigma - \cap$ stable $\Leftrightarrow$ $\mathcal{E}$ $\sigma - \cup$ stable.

2. Show that for a non-empty set Ω the set $\mathcal{E} = \{A \subset \Omega \,|\, A \text{ finite or } \bar{A} \text{ finite}\}$ is a σ-algebra for finite Ω. For infinite Ω, $\mathcal{E}$ is not a σ-algebra.

3. Show $\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(AB)$.

4. Show $\mathbb{P}(\bar{A}|B) = 1 - \mathbb{P}(A|B)$

5. Show $\mathbb{P}(A|\bar{B}) = (\mathbb{P}(A) - \mathbb{P}(B)\mathbb{P}(A|B)) \,/\, (1 - \mathbb{P}(B))$

6. Let $X \sim \mathcal{N}_{\mu,\sigma^2}$, $a \in \mathbb{R}\backslash\{0\}$ and $b \in \mathbb{R}$. Show that $Y = (aX + b)$ is $\mathcal{N}_{a\mu+b,a^2\sigma^2}$ distributed.

7. If $X \sim \text{Exp}(\theta)$ and $a > 0$, then aX is $\text{Exp}(\theta/a)$ distributed.

8. Let X_1, X_2 be **iid** normal distributed random variables with $\mathbb{E}(X_1) = 0$ and variance $\text{Var}(X_1) = \sigma^2$. Show that $Y = X_1 + X_2$ is a normal distributed random variable with $\mathbb{E}(Y) = 0$ and $\text{Var}(Y) = 2\sigma^2$.

9. Let X_1, X_2 be independent random variables with pdfs f_{X_1} and f_{X_2} and $Y = X_1 X_2$. Compute the pdf of Y.

10. Let X and Y be independent random variables with $X \sim \text{Exp}(\theta)$ and $Y \sim \text{Exp}(\rho)$ for θ, $\rho > 0$. Show that $\mathbb{P}[X < Y] = \frac{\theta}{\theta+\rho}$.

11. Let Z be a real $p \times n$ matrix, $n, p > 0$ and define $\Sigma = Z \cdot Z^t$ an $n \times n$ matrix. Show that Σ is positive semi–definite.

References

[1] K Bosch. *Elementare Einführung in die angewandte Statistik*. Vieweg+Teubner, 2010.

[2] K Bosch. *Elementare Einführung in die Wahrscheinlichkeitsrechnung*. Vieweg+Teubner, 2010.

[3] A Klenke. *Probability Theory: A Comprehensive Course*. 2nd ed. Springer, 2014. DOI: 10.1007/978-1-4471-5361-0.

[4] A Klenke. *Wahscheinlichkeitstheorie*. 3rd ed. Springer Spektrum, 2013. DOI: 10.1007/978-3-642-36018-3.

[5] RY Rubinstein and DP Kroese. *Simulation and the Monte Carlo Method*. 2nd ed. Wiley Series in Probability and Statistics. Wiley-Interscience, 2008. DOI: 10.1002/978-0-4702-3038-1.

[6] Wikipedia. *Probability distribution*. [Online; accessed 04-October-2016]. 2016. URL: https://en.wikipedia.org/wiki/Probability_distribution.

3. Random variable generation

After having defined probability distributions in the previous chapter, we will now discuss how to generate random numbers drawn from a given probability distribution on a computer. The reason is that simulation methods heavily rely on the possibility to generate random variables that are distributed according to some pdf f.

It is important to realise that random number generators on modern computers do generate only pseudo random numbers: the sample is generated using a *deterministic* algorithm, but still almost has the properties of a truely random sample.

We will first show how to generate uniform random numbers $\mathcal{U}_{[0,1]}$ and then, how to transform them into (almost) any other distribution. However, the most important result in view of the coming chapters is the *fundamental theorem of simulation*.

3.1. Pseudo Random Number Generator

We shall use a deterministic algorithm to generate **pseudo random numbers**, which mimic properties of **iid** random numbers.

Definition 3.1.1: Uniform Pseudo Random Number Generator (pRNG)

*A Uniform Pseudo Random Number Generator (pRNG) is an algorithm which, starting from an initial value u_0 and a transformation D, produces a sequence $u_i = D^i(u_0)$ of values in $[0,1]$. For all n, the sequence $(u_0, \ldots, u_{n-1})$ should reproduce the behaviour of an **iid** sample $(V_1, \ldots, V_n)$ of uniform random numbers when compared through a suite of standard tests (e.g. Kolmogorov-Smirnov-test, χ^2-test, Spectral test).*

In other words, random numbers are generated as a sequence

$$x_{n+1} \;=\; Dx_n\,.$$

Definition 3.1.2: Period

The period T_0 of a pRNG is the smallest T, such that $x_{i+T} = x_i$ for all i.

Modern random number generators are based on two techniques, or combinations thereof. They, in fact, generate random integers in some range $\{0, \ldots, M\}$, which are then scaled to the interval $[a, b]$. The first one is a so-called congruential generator, defined as follows:

Definition 3.1.3: Linear Congruential RNG (LCG)

A linear congruential random number generator generator is defined via the mapping

$$D(x) \;=\; (a \cdot x + b) \;\; mod \;\; (M+1)\;, \qquad x, a, b, M \in \mathbb{Z}_+^0\,, \tag{3.1.1}$$

with

- M*: the modulus,* $M > 0$
- a*: the multiplier,* $0 \leq a < M$
- b*: the increment,* $0 \leq b < M$

And, of course, it requires an initial value x_0*.*

For a linear-congruential generator we have $T_0\,(\mathrm{LCG}) \leq M$.

Example 3.1.1

The infamous **RANDU** LCG is given by the parameters: $M = 2^{31}$, $a = 65539$, $b = 0$. It is infamous, because it was extensively used while having severe problems with correlations, see section 3.4.

The second type of generators is called *shift-register* generator. It relies on the fact that the k binary components of an integer represented on a computer as $x_n \in \{0, \ldots, M\}$ can be modified independently.

Definition 3.1.4: Shift-Register Generator (SRG)

An integer is represented in binary form as

$$x_n = \sum_{i=0}^{k-1} e_{ni}\, 2^i\,, \quad e_{n,i} \in \{0,1\}\;\forall\, i\,.$$

For the shift-register generator the transformation D *is given by the application of a matrix* $T : T_{i,j} \in \{0,1\}$ *combined with a modulo-2 operation as follows*

$$e_{n+1,i} \;=\; T_{ij} \;\mathrm{XOR}\; e_{n,j}\,,$$

from which the new x_{n+1} *is defined by taking* $e_{n+1,i}$ *as binary components,* $x_{n+1} = \sum_{i=0}^{k-1} e_{n+1,i}\, 2^i$.

3.2. Random variable generation

Given the ability to generate uniform random numbers in some interval, we now discuss methods to transform them to other distributions.

3.2.1. Inverse transform method

In the previous chapter we have seen already how to use inverse functions to transform one distribution into another one. We will generalise this concept here. For this we first generalise the concept of an inverse function for the case of probability distributions.

Definition 3.2.1: Generalised Inverse

For a real valued, non-decreasing function F (the cdf of some distribution) the generalised inverse F^{-1} is defined by

$$\begin{aligned} F &: \mathbb{R} \to [0,\,1] \\ F^{-1} &: [0,\,1] \to \mathbb{R}\,, \quad u \mapsto F^{-1}(u) = \inf\{x \,|\, F(x) \geq u\}\ . \end{aligned} \tag{3.2.1}$$

This generalised inverse is needed for the case that F is not bijectiv and, hence, the usual inverse does not exist. With the generalised inverse we can proof the following

Theorem 3.2.1:

Let $U \sim \mathcal{U}_{[0,1]}$ and $F_X : \mathbb{R} \to [0,1]$ be a cdf. Then the random variable $F_X^{-1}(U)$ has cdf F_X.

Proof:
For any $u \in [0,1]$ we have from the definition

$$F_X\left(F_X^{-1}(u)\right) \;\geq\; u \tag{3.2.2}$$

and also for any $x \in F_X^{-1}([0,1])$

$$F_X^{-1}\left(F_X(x)\right) \;\leq\; x\,. \tag{3.2.3}$$

Consider now the following equality of sets

$$\left\{(x,u):\ F_X^{-1}(u) \leq x\right\} = \{(u,x) \,|\, F_X(x) \geq u\}$$

which follows from Eqs. 3.2.2 and 3.2.3 by using that F_X is non-decreasing, so whenever $x \leq y$ we have $F_X(x) \leq F_X(y)$ and by the analogous property for F_X^{-1}: since for $u \leq u'$ we have $F_X^{-1}(u) \leq F_X^{-1}(u')$.

This identity of sets translates to the following equality of probabilities

$$\mathbb{P}\left(F_X^{-1}(u) \leq x\right) \;=\; \mathbb{P}\left(u \leq F_X(x)\right)$$

which is nothing but $F_X(x)$.

□

The inverse transform method is quite general and always applicable, if F_X^{-1} can be computed. It might, however, be not very convenient in case F_X^{-1} is difficult and / or costly to evaluate. One may need to use an approximation to F_X^{-1}, e.g. a discretisation.

Example 3.2.1

We generate exponential random variables from uniform ones using the inverse transform. The cdf of Exp(1) is given by

$$F(x) \;=\; 1 - \mathrm{e}^{-x} \;=\; u\,,$$

which can be inverted, such that

$$x \;=\; -\ln(1-u)$$

has exponential distribution, if $u \sim \mathcal{U}_{[0,1]}$.

Example 3.2.2

We consider the triangle pdf

$$f(x) = \begin{cases} 1+x & -1 \le x < 0\,, \\ 1-x & 0 \ge x \le 1\,. \end{cases}$$

By piece-wise integration we get the cdf

$$F(x) = \begin{cases} \frac{1}{2}\,(1+x)^2 & -1 \le x < 0\,, \\ 1 - \frac{1}{2}\,(1-x)^2 & 0 \le x \le 1\,. \end{cases}$$

The cdf is monotonously non-decreasing and can be inverted to give the inverse F^{-1}

$$F^{-1}(u) \;=\; \begin{cases} 1 - \sqrt{2\,(1-u)} & \frac{1}{2} \le u <\le 1\,, \\ \sqrt{2\,u} - 1 & 0 \le u < \frac{1}{2}\,. \end{cases} \tag{3.2.4}$$

3.2.2. General Transformation methods

This method is used in cases, when the target distribution is linked to another distribution, which is easier to simulate. We will only show two examples here, other similar examples have already been shown in the previous chapter.

Example 3.2.3

If $X_j \sim Exp(1)$ for $k = 1, \ldots, \nu$, then $Y = \sum_{j=1}^{\nu} X_j \sim \chi^2_\nu$ is χ^2-distributed with 2ν

degrees of freedom. We use ascending variables

$$\begin{aligned} z_1 &= x_1 \\ z_2 &= z_1 + x_2 \\ &\vdots \\ z_n &= z_{n-1} + x_n = y/2 \end{aligned}$$

with $0 \leq z_1 \leq z_2 \leq \ldots \leq z_n = y/2$ and $\mathrm{d}x_n = \mathrm{d}z_n$.

$$\begin{aligned} F_Y(y) &= \int_0^{y/2} \exp(-z_n) \left[\int_0^{z_n} \ldots \int_0^{z_2} \mathrm{d}z_1 \ldots \mathrm{d}z_{n-1} \right] \mathrm{d}z_n \\ &= \int_0^{y/2} \exp(-z_n) \, \frac{z_n^{n-1}}{(n-1)!} \, \mathrm{d}z_n \,. \end{aligned}$$

With $t = 2z_n$ this becomes

$$F_Y(y) = \int_0^{y} \exp(-t/2) \, \frac{t^{n-1}}{2^n \, \Gamma(n)} \, \mathrm{d}t$$

and finally rewriting $n = \nu = (2\nu)/2$ we have

$$F_Y(y) = \int_0^{y} \exp(-t/2) \, \frac{t^{(2\nu)/2-1}}{2^{(2\nu)/2} \, \Gamma((2\nu)/2)} \, \mathrm{d}t = F_{\chi^2_{2\nu}}(y) \,.$$

Example 3.2.4

The Box-Muller method where from

$$U, V \sim \mathcal{U}(0, 1) \,, \quad \text{iid}$$

standard normal random variables X, Y are generated from the transformation

$$\begin{pmatrix} X \\ Y \end{pmatrix} = G\left(\begin{pmatrix} U \\ V \end{pmatrix} \right) = \sqrt{-2 \log(U)} \begin{pmatrix} \cos(2\pi V) \\ \sin(2\pi V) \end{pmatrix} .$$

The transformation G is a diffeomorphism on $(0, 1)^2$ and with $f_{U,V}(u, v) = f_U(u) \cdot$

$f_V(v) = \mathbb{1}_{(0,1)}(u) \cdot \mathbb{1}_{(0,1)}(v)$ the joint pdf for U, V using polar coordinates we have

$$\begin{pmatrix} R \\ \Phi \end{pmatrix} = G\left(\begin{pmatrix} \sqrt{-2\log(U)} \\ 2\pi V \end{pmatrix}\right) \quad \Leftrightarrow \quad G^{-1}\left(\begin{pmatrix} R \\ \Phi \end{pmatrix}\right) \quad = \begin{pmatrix} \exp\left(-R^2/2\right) \\ \Phi/2\pi \end{pmatrix}$$

Now we need to compute the JACOBI-matrix:

$$DG = \begin{pmatrix} -\frac{1}{u\sqrt{2\log(u)}} & 0 \\ 0 & 2\pi \end{pmatrix}$$
$$DG \circ G^{-1}(r, \phi) = \begin{pmatrix} -\exp\left(r^2/2\right)/r & 0 \\ 0 & 2\pi \end{pmatrix}$$

and hence

$$\left|\det\left(DG \circ G^{-1}(r, \phi)\right)\right| = 2\pi \, \exp\left(r^2/2\right)/r \, .$$

Now, the joint distribution is given as

$$\begin{aligned} f_{X,Y}(x, y)\,\mathrm{d}x\,\mathrm{d}y = f_{R,\Phi}(r, \phi)\, r\,\mathrm{d}r\,\phi = f_{U,V} \circ G^{-1}(r, \phi)\, \frac{1}{\left|\det\left(DG \circ G^{-1}(r, \phi)\right)\right|}\,\mathrm{d}r\,\mathrm{d}\phi \\ = \frac{1}{\sqrt{2\pi}^2} \exp\left(-r^2/2\right)\, r\,\mathrm{d}r\,\mathrm{d}\phi \\ = \frac{1}{\sqrt{2\pi}^2} \exp\left(-(x^2+y^2)/2\right)\,\mathrm{d}x\,\mathrm{d}y \\ = \left(\frac{1}{\sqrt{2\pi}} \exp\left(-x^2/2\right)\,\mathrm{d}x\right) \times \left(\frac{1}{\sqrt{2\pi}} \exp\left(-y^2/2\right)\,\mathrm{d}y\right) \\ = f_X(x)\,\mathrm{d}x \times f_Y(y)\,\mathrm{d}y \, . \end{aligned}$$

Hence X, Y are **iid** standard-normally distributed. The variation of the Box-Muller method (polar method) works analogously.

In general some ingenuity will be required to represent the target distribution by a general transformation.

3.2.3. Accept-reject methods

Now we come – in view of the topics to be covered in the future – to the core of this chapter, namely the *fundamental theorem of simulation*. It will not only define new means to sample from probability distributions, it will also introduce a very first simulation method and the first stochastic integration procedure. Accept-reject methods require only minimal assumptions on the distribution to be sampled from: one basically needs to be only able to evaluate the corresponding pdf f_X.

The whole method is based on the following identity

$$f_X(x) \;=\; \int_0^{f_X(x)} \mathrm{d}u \;=\; \int_0^{f_X(x)} \mathbb{1}\,\mathrm{d}u\,, \tag{3.2.5}$$

which, when $\mathbb{1}$ is interpreted as the joint distribution $f_{U,X} = \mathbf{1}$ of the independent random variables X and U (U is called *auxiliary* variable) leads to the interpretation of f_X as the marginal distribution of $f_{U,X}$ with the auxiliary variable U integrated out. With $f_{U,X} = \mathbb{1}$, we have to sample uniformly from the constrained set

$$(U, X) \;\sim\; \mathcal{U}\{(x,u) : 0 \le u < f(x)\}\,. \tag{3.2.6}$$

This proofs already the following central theorem:

Theorem 3.2.2: Fundamental Theorem of Simulation

Simulating $X \sim f(x)$ is equivalent to simulating the constrained set

$$(X, U) \;\sim\; \mathcal{U}\{(x,u) : 0 < u < f(x))\}\,.$$

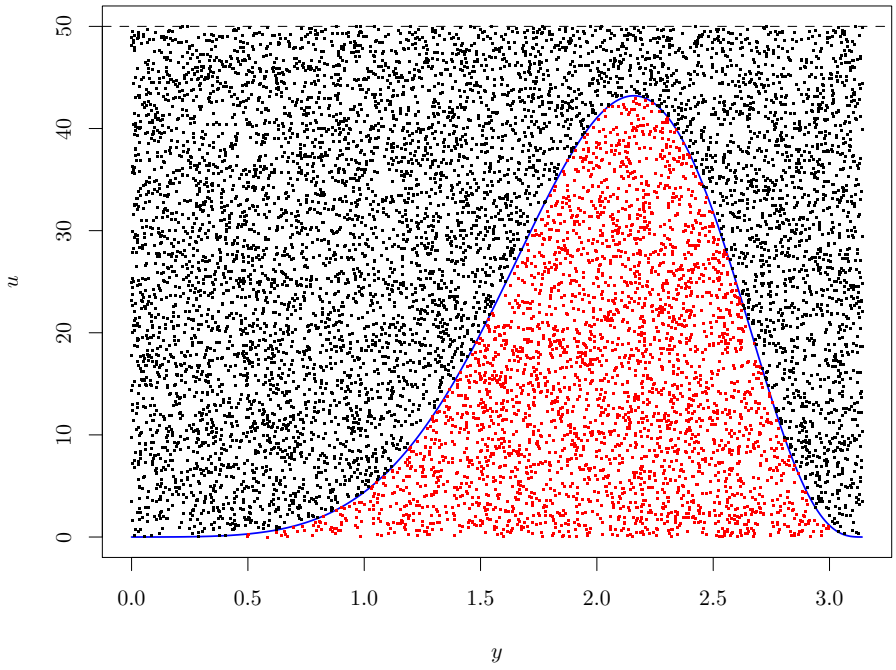


Figure 3.2.1.:
Accept-reject approach with constant upper bound as discussed in section 3.4. The red points correspond to the accepted ones, and the black to the not accepted ones. The code to generate this figure can be found in section 3.4

In this form the theorem is not yet very useful, because simulating on this constrained set is usually impractical. In order to make it useful in practice we have to define a set that is simple to simulate uniformly on and then constrain it to the set we are interested in. Such a bigger set could for instance be a rectangular area, which we can simulate on uniformly easily.

Consider a one dimensional target pdf f_X with compact support in $[a, b]$, $a, b \in R$, which has the property that for an $m \in \mathbb{R}$

$$f_X(x) \;\leq\; m\,, \quad \forall x \in [a, b]\,.$$

Then we uniformly simulate the set

$$(X, U) \;\sim\; \mathcal{U}\{[a, b] \times [0, m]\}$$

by simulating independently $Y \sim \mathcal{U}([a, b])$ and independently $U \sim \mathcal{U}([0, m])$ and accept $x = y$ if

$$u \;\leq f(y)\,.$$

Then, X has the distribution

$$\begin{aligned}\mathbb{P}\left(X \leq x\right) = \mathbb{P}\left(Y \leq x | U < f_X(Y)\right) &= \frac{\mathbb{P}\left(Y \leq x \cap U < f_X(Y)\right)}{\mathbb{P}\left(U \leq f_X(Y)\right)} \\ &= \int_a^x \int_0^{f_X(y)} \frac{\mathrm{d}u}{m}\,\frac{\mathrm{d}y}{b-a} \left[\int_a^b \int_0^{f_X(y)} \frac{\mathrm{d}u}{m}\,\frac{\mathrm{d}y}{b-a}\right]^{-1} = \int_a^x f_X(y)\mathrm{d}y\,.\end{aligned}$$

Moreover, the *acceptance* probability is given by

$$\mathbb{P}_{\text{accept}} = \mathbb{P}(U < f_X(Y)) = \int_a^b \int_0^{f_X(y)} \frac{\mathrm{d}u}{m}\,\frac{\mathrm{d}y}{b-a} = \frac{1}{(b-a)m}\,, \tag{3.2.7}$$

which is – remember that f_X is normalised to one – equal to

$$\frac{\text{area}(f_X)}{\text{area(box)}}\,.$$

At this point we have – without intention – designed the first stochastic integration algorithm: if f_X was not normalised to one, we could compute it from

$$\begin{aligned}\text{area}(f_X) \;&=\; \text{area(box)} \cdot \mathbb{P}_{\text{accept}} \\ &=\; \text{area(box)} \frac{N_{\text{accept}}}{N_{\text{total}}}\,,\end{aligned} \tag{3.2.8}$$

which will converge stochastically against the integral of f_X over the interval $[a, b]$. The algorithm is visualised in Figure 3.2.1, which was produced with the code presented in section 3.4.

This simple version of the accept-reject method will not work in more complicated cases, e.g. when f_X is unbound or the support is not compact. Still, we can generalise

Algorithm 3.2.1 Accept-Reject

1: generate $X \sim g(X)$
2: generate $Y \sim \mathcal{U}(0, Cg(x))$
3: if $Y \leq f(X)$ return X, otherwise return to step 1.

the method as follows: let g be an arbitrary instrumental density with the property $f_X(x) \leq Cg(x)$ for all x and some constant $C \in \mathbb{R}$. g is also called the proposal pdf and we assume that it is easy to simulate from g. Then, the algorithm can be written as in 3.2.1

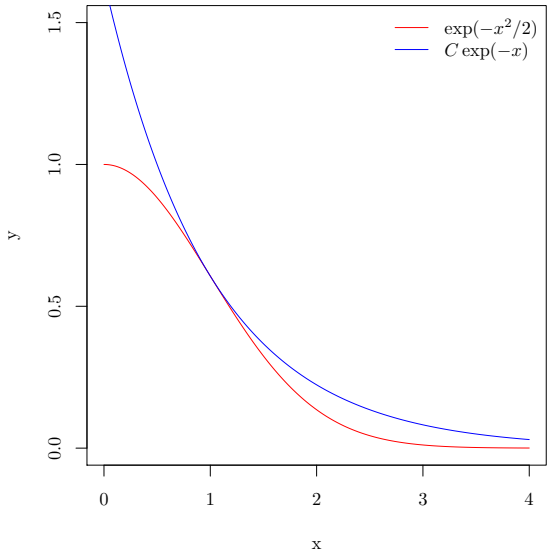


Figure 3.2.2.: Illustration of example 3.2.5.

Theorem 3.2.3: Fundamental Theorem of Simulation II

The random variable generated according to algorithm 3.2.1 has pdf f_X.

Proof:
Define two sets A, B as follows

$$A = \{(x, y) : 0 \leq y \leq Cg(x)\}, \qquad B = \{(x, y) : 0 \leq y \leq f_X(x)\}.$$

with $B \subset A$, since A represents the area underneath $Cg(x)$ and B the one underneath $f_X(x)$. The random vector (X, Y) is uniformly distributed on A. Actually, the joint pdf of (X, Y) is given by $f_{X,Y} = C^{-1}$ for all $(x, y) \in A$ (the area of A equals C due to g being a pdf). This implies that the accepted vectors $(X^\star, Y^\star)$ are uniformly distributed on B. Due to the normalisation of f_X as a pdf, the area of B is unity and, therefore, the

joint pdf of $(X^\star, Y^\star)$ is the pdf for the uniform distribution as well. Thus, the marginal pdf of $X^\star$ is

$$\int_0^{f_X(x)} \mathbf{1}\,\mathrm{d}y = f_X(x)\,.$$

□

The efficiency or acceptance probability of algorithm 3.2.1 is again given by the ratio of the two areas, hence,

$$\mathbb{P}\left((X,Y) \text{ accepted}\right) \;=\; \frac{\text{area}\,B}{\text{area}\,A} = \frac{1}{C}\,.$$

Example 3.2.5

We want to generate from a normal distribution with pdf

$$f(x) = A\,\mathrm{e}^{-\frac{x^2}{2}}$$

using an exponential distribution as instrumental density. First we explore the symmetry of f by simulating only $x \geq 0$ and assigning a random sign. Next we chose as the proposal pdf the exponential pdf for $\lambda = 1$

$$g(x) = \mathrm{e}^{-x}\,.$$

Now we chose $C = A\sqrt{\mathrm{e}}$ for which $f(x) \leq Cg(x)$ for all x. Now apply algorithm 3.2.1

3.3. Generating Correlated Random Numbers

It is sometimes useful to generate correlated instead of independent random numbers. We will discuss here the case of normally distributed random numbers, because this is a commonly needed case when data with error and correlation needs to be analysed further. In case of X_1, X_2 both normal distributed with mean zero and variance σ^2 the correlation matrix reads

$$\mathrm{Cor}(X_1, X_2) \;=\; \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}\,, \tag{3.3.1}$$

with correlation coefficient ρ. If we start from *uncorrelated* normally distributed random variables Y_1, Y_2 with mean zero and variance σ^2, we would like to generate X_1, X_2 from Y_1, Y_2. Making the ansatz

$$X \;=\; \phi(Y) \;=\; A \cdot Y\,, \tag{3.3.2}$$

with $A : \mathbb{R}^2 \to \mathbb{R}^2$ linear and invertible. Let us first compute the corresponding cdf

$$\begin{aligned} F_{Y_1,Y_2}(y_1, y_2) \;&=\; \int_{-\infty}^{y_1} \mathrm{d}y_1' \int_{-\infty}^{y_2} \mathrm{d}y_2' \,\frac{1}{2\pi\sigma^2}\mathrm{e}^{-\frac{(y_1')^2+(y_2')^2}{2\sigma^2}} \\ &=\; \int_{-\infty}^{y_1} \mathrm{d}y_1' \int_{-\infty}^{y_2} \mathrm{d}y_2' \,\frac{1}{2\pi\sigma^2}\mathrm{e}^{-\frac{1}{2}(y')^t\Sigma^{-2}y'}\,, \end{aligned} \tag{3.3.3}$$

with $\Sigma = \mathrm{diag}\,(\sigma, \sigma)$ and $x' = (x'_1, x'_2)^t$. Now we perform a change of variables to $x = A \cdot y$ and apply Eq. 2.3.6

$$\begin{aligned} \mathrm{d}y'_1 \mathrm{d}y'_2 \, \frac{1}{2\pi\sigma^2} \mathrm{e}^{-\frac{(y'_1)^2+(y'_2)^2}{2\sigma^2}} &= \mathrm{d}x'_1 \, \mathrm{d}x'_2 \, |D\phi^{-1}| \, \frac{1}{2\pi\sigma^2} \mathrm{e}^{-\frac{1}{2}(x')^t (A\Sigma^2 A^t)^{-1} x'} \\ &= \mathrm{d}x'_1 \, \mathrm{d}x'_2 \, |D\phi^{-1}| \, \frac{1}{2\pi\sigma^2} \mathrm{e}^{-\frac{1}{2\sigma^2}(x')^t (A \cdot A^t)^{-1} x'} \,. \end{aligned} \tag{3.3.4}$$

Since ϕ is linear, the Jacobi–determinant is given by $|D\phi^{-1}| = (\det A)^{-1}$, leading to the cdf of X_1, X_2

$$F_{X_1,X_2}(x_1, x_2) \;=\; \int_{-\infty}^{x_1} \int_{-\infty}^{x_2} \mathrm{d}x'_1 \, \mathrm{d}x'_2 \, \frac{1}{2\pi\sigma^2 \det A} \mathrm{e}^{-\frac{1}{2\sigma^2}(x')^t (A \cdot A^t)^{-1} x'} \,. \tag{3.3.5}$$

In order for F_{X_1,X_2} to be appropriately normalised, we require A to be positive definite. The expression Eq. 3.3.5 corresponds actually to the cdf of a bivariate normally distribution, if the matrix $\sigma^2 A \cdot A^t$ is the covariance matrix, which turns out to be the case

$$\mathrm{Cov}\,(X_1, X_2) \;=\; \langle X \cdot X^t \rangle \;=\; A \, \langle Y \cdot Y^t \rangle \, A^t \;=\; \sigma^2 A \cdot A^t \;\equiv C \,. \tag{3.3.6}$$

From this follows that $A \cdot A^t$ must be real and symmetric, positive semi–definite. From requiring that X_1 and X_2 both have variance σ^2, the squares of elements of each row must sum to σ^2, from which follows

$$\mathrm{Cor}\,(X_1, X_2) \;=\; A \cdot A^t \;=\; \frac{1}{\sigma^2} C \,. \tag{3.3.7}$$

We use the special ansatz $A = A^t$. Hence, we have to find the matrix A such that

$$A^2 \stackrel{!}{=} \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} .$$

A possible solution for A is given by

$$A \;=\; \begin{pmatrix} \sin\phi & \cos\phi \\ \cos\phi & \sin\phi \end{pmatrix}$$

which also fulfils the condition on the sum of elements in each row, namely $\sin^2\phi + \cos^2\phi = 1$. A^2 reads

$$A^2 \;=\; \begin{pmatrix} \sin^2\phi + \cos^2\phi & 2\sin\phi\cos\phi \\ 2\sin\phi\cos\phi & \sin^2\phi + \cos^2\phi \end{pmatrix} \;=\; \begin{pmatrix} 1 & \sin 2\phi \\ \sin 2\phi & 1 \end{pmatrix}$$

leading to

$$\rho \;\stackrel{!}{=}\; \sin 2\phi \qquad \Leftrightarrow \qquad \phi = \frac{1}{2} \sin^{-1} \rho \,.$$

The derivation given here for two correlated normally distributed random numbers generalises to n correlated normally distributed random numbers. In this case the matrix A can be found by a Cholesky decomposition of the correlation matrix.

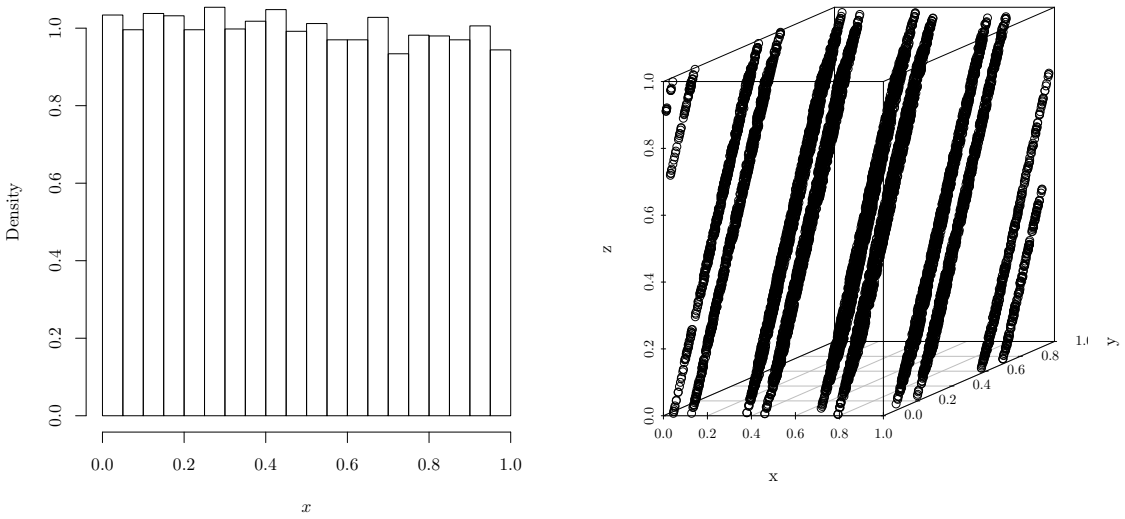


Figure 3.4.1.:
Left: histogram of random numbers in $[0, 1]$ produced by RANDU. Right: threedimensional plot of (x_i, x_{i+1}, x_{i+2}) of random numbers generated by RANDU. One clearly observes the unwanted correlation RANDU generates.

Finally, making use of $A = A^t$ we can rewrite the joint cdf as follows

$$F_{X_1,X_2}(x_1, x_2) = \int_{-\infty}^{x_1} \mathrm{d}x_1' \int_{-\infty}^{x_2} \mathrm{d}x_2' \, \frac{1}{2\pi \det C^{1/2}} \mathrm{e}^{-\frac{1}{2}(x')^t C^{-1} x'}$$

which is the expression for the cdf of a bivariate normal distribution with covariance matrix C.

3.4. Random Numbers with R

Let us start by looking at the RANDU random number generator and see, why it is so bad. Here is an R function implementing it

```
rrandu <- function(n) {
  x0 <- 5
  M <- 2^21
  a <- 65539
  x <- x0
  for(i in c(1:n)) {
    x[i+1] <- (a*x[i]) %% M
```

```
  }
  return(x/(M+1))
}
```

When looking at the histogram of a random sample generated with RANDU, everything seems fine.

```
x <- rrandu(10000)
idx <- c(1:9998)
hist(x, main="", freq=FALSE)
```

The histogram is shown in the left panel of Figure 3.4.1. However, when plotted in three dimensions with points $\vec{x}_i = (x_i, x_{i+1}, x_{i+2})$, one observes that the points are not uniformly distributed in the unit cube.

```
library(scatterplot3d)
scatterplot3d(x=x[idx], y=x[idx+1], z=x[idx+2], angle=20,
              xlab="x", ylab="y", zlab="z")
```

This is shown in the right panel of Figure 3.4.1.

R has of course many distributions implemented already. Typically, R provides functions for the pdf, the cdf, the quantiles and one function to generate random deviates, i.e. correspondingly distributed pseudo random numbers. An example is the uniform distribution, for which `dunif`, `punif`, `qunif` and `runif` are provided for the density, the cdf, the quantiles and the random number generation. See the corresponding help pages using e.g. `?runif`.

We consider again example 3.2.2 for the triangle pdf. First, we define a function for the generlised inverse Eq. 3.2.4 as follows

```
Finv <- function(x) {
  if(any(x<0) ||  any(x>1)) stop("Error, x must be in [0,1]")
  res <- numeric(length(x))
  res[x >= 0.5] <- 1 - sqrt(2*(1-x[x >= 0.5]))
  res[x < 0.5] <- sqrt(2 * x[x < 0.5]) - 1
  return(res)
}
```

Here we see the ability of R to work with index sets. It allows to use conditions on vectors to be used as index sets. With this function at hand we have to first generate a random sample of size N of uniformly distributed random numbers, which is done via

```
N <- 10000
u <- runif(n=N)
```

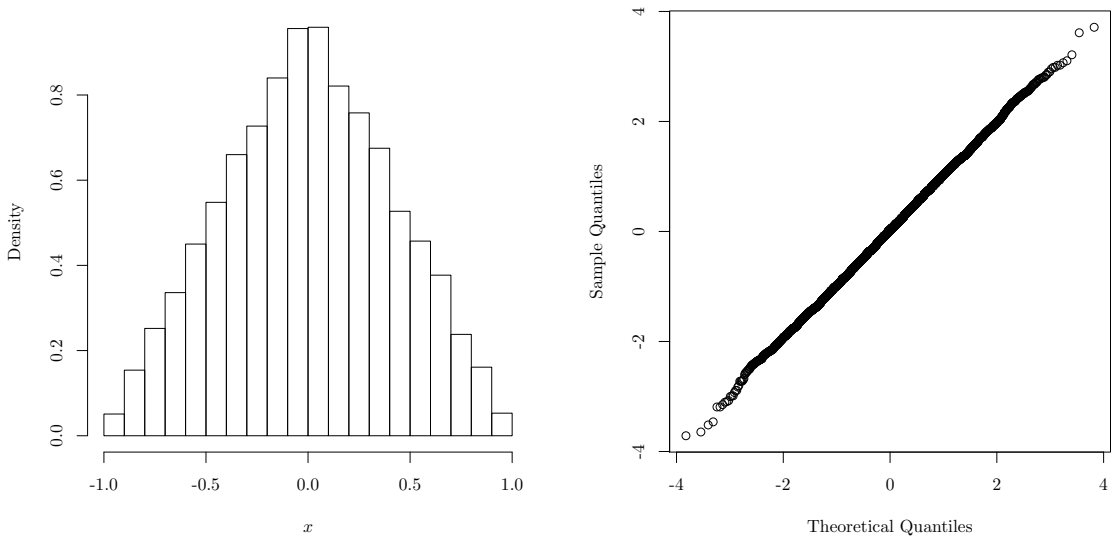


Figure 3.4.2.:
Left: histogram for the triangle pdf generated with the inverse transform method. Right: QQ-plot for the normal random numbers generated with the accept-reject method from exponential random numbers.

They are stored in the vector `u` of length $N = 10^4$. Now, we can generate random numbers using the generalised inverse as

```
x <- Finv(u)
hist(x, freq=FALSE, main="")
```

The `hist` function generates a histogram of the data. The corresponding plot is shown in the left panel of Figure 3.4.2. This was predicted by Marsaglia [1] and is known as spectral test. RANDU was used for an extensive number of scientific results, which must all be taken with a lot of care now.

Let's come to a second example, this time for the accept-reject algorithm. Consider the example

$$f(x) \propto \exp(2x) \cdot \sin(x)^3 , \quad 0 \leq x \leq \pi .$$

which we define in `R` as follows

```
f <- function(x) exp(2*x)*sin(x)^3
```

An upper bound for this function is provided by $m = 50$. Now we generate y uniformly in $[0, \pi]$ and u uniformly in $[0, 1]$ and implement the accept-reject step by determining the indices of the accepted points

```
N <- 10000                ## number of samples
m <- 50                   ## upper bound
y <- runif(n=N)*pi        ## uniform in [0,pi]
u <- runif(n=N)*m         ## uniform in [0, m]
idx <- which(u < f(y))    ## indices of accepted points
```

Finally, we plot the function and the random points as follows.

```
plot(NA, xlim=c(0,pi), ylim=c(0,50), main="", xlab="y", ylab="u")
abline(h=50, lty=2) ## horizontal line
x <- seq(0,pi,0.01)
fy <- f(x)
lines(x=x, y=fy, col="blue", lwd=2.5) ## the function line
points(x=y[idx], y=u[idx], col="red", pch=".")      ## accepted points
points(x=y[-idx], y=u[-idx], col="black", pch=".") ## not accepted points
```

In this code chunk we see another nice ability of R, namely the use of negative index arrays: those are interpreted as the complement index set. The corresponding plot can be found in Figure 3.2.1.

Next we have a look at example 3.2.5, where we have proposed to generate normal random numbers from exponential ones. So, here is an R implementation of algorithm 3.2.1 for this case including the random sign generation:

```
C <- sqrt(exp(1))
x <- rexp(n=N)
y <- runif(n=N)*exp(-x)*C
idx <- which(y <= exp(-x^2/2))     ## the indices of accepted values
sign <- rep(1, times=length(idx)) ## take care of the random sign
sign[runif(n=length(idx)) <= 0.5] <- -1
length(idx)/N                      ## accepted points: "efficiency"

## [1] 0.7587

qqnorm(x[idx]*sign, main="")       ## compare to theoretical quantiles
```

In order to check whether or not the generated random number are normally distributed, a QQ-plot is generated using `qqnorm` and shown in the left panel of Figure 3.4.2. All the points are located on a straight line with slope one indicating good agreement between theoretical quantiles and sample quantiles.

Finally, it is instructive to compare the normally distributed random numbers we just generated against normally distributed random numbers generated using the central limit theorem like in example 2.6.1 Eq. 2.6.9

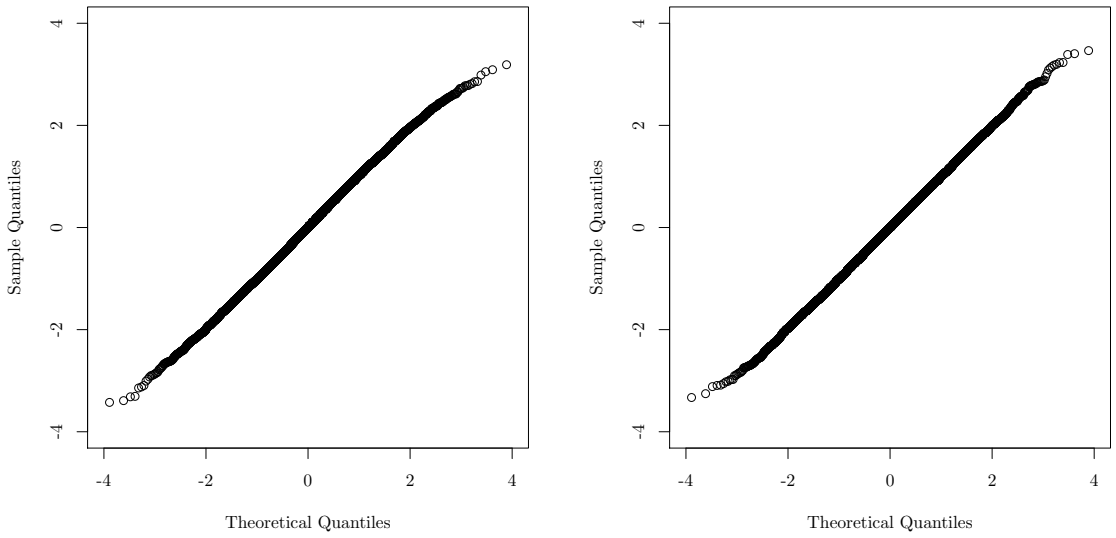


Figure 3.4.3.:
QQ-plot comparing the quantiles of normally distributed random numbers generated with the central limit theorem to the theoretical expectation. Left: $n = 6$ in the sum. Right: $n = 12$.

```
cltnormal <- function(n=1, N=12) {
  u <- array(runif(n=N*n), dim=c(N,n))
  return((apply(u, MARGIN=2, FUN=sum) - N/2)*sqrt(12/N))
}
qqnorm(cltnormal(n=length(idx)), main="")
```

The corresponding QQ-plot is shown in Figure 3.4.3 with $n = 6$ in the left and $n = 12$ in the right panel (recall, n is the number of terms in the sum). It is visible that the implementation using the central limit theorem has some problems to correctly sample in the tails of the normal distribution. The problem is worse for $n = 6$. The previously discussed accept-reject method has less of this problem, but also there it is still visible.

Finally, here is an implementation of correlated normal distributed random numbers as described in section 3.3.

```
N <- 500
rho <- 0.95    ### correlation coefficient
phi <- 0.5*asin(rho)
A <- matrix(c(sin(phi), cos(phi), cos(phi), sin(phi)), nrow=2, ncol=2)
y <- array(rnorm(n=2*N), dim=c(2, N))
x <- A %*% y
```

It makes use of R's matrix multiplication capability. A scatterplot of the new variable x shows the correlation introduced by the procedure. But we can also compute the correlation using `cor(x[1,], x[2,])`, resulting in a correlation coefficient of $\rho = 0.9520411$.

References

[1] G Marsaglia. "Random Numbers fall mainly in the Planes". In: *PNAS* 61.1 (1968), pp. 25–28.

[2] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2016. URL: https://www.R-project.org/.

[3] CP Robert and G Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005. ISBN: 0387212396.

[4] RY Rubinstein and DP Kroese. *Simulation and the Monte Carlo Method*. 2nd ed. Wiley Series in Probability and Statistics. Wiley-Interscience, 2008. DOI: 10.1002/978-0-4702-3038-1.

[5] C Sharpsteen and C Bracken. *tikzDevice: R Graphics Output in LaTeX Format*. R package version 0.6.3/r49. 2012. URL: https://R-Forge.R-project.org/projects/tikzdevice/.

4. Analysis of Stationary Data

In this chapter we basically want to determine properties of samples of random numbers, i.e. we want to measure moments of distributions given a random sample of (finite) size N. In practice this often means to estimate the mean μ and the variance σ^2.

Since we will base this on a finite sample of realisation of a random variable X (or a random vector $\vec{X}$), the moments can be computed only with a certain reliability, which is why we talk about *estimation*. This is why a major fraction of this chapter is about estimating also the uncertainties.

4.1. The Plug-in Principle

We are interested in computing arbitrary observables

$$\theta \;=\; t(f)\,. \tag{4.1.1}$$

Here, we emphasise the dependence of θ on the distribution f. The dependence on f might be complicated and we will not specify it further here. The question is how to obtain θ from a finite sample. In general, the procedure is now to draw a random sample of size N from f

$$f \quad\to\quad (x_1, ..., x_N)\,. \tag{4.1.2}$$

Now, we estimate θ from the random sample, i.e.

$$\hat{\theta} \;=\; t(x_1, ..., x_N)\,. \tag{4.1.3}$$

Strictly speaking, we have replaced f by an estimate $\hat{f}$ derived from the sample $(x_1, ..., x_N)$. $\hat{f}$ is called the empirical distribution function and the procedure is also called the *plug-in principle* . This leads to the following definition

Definition 4.1.1: plug-in estimator

$t(f)$ can be estimated from the empirical distribution $\hat{f}$ by means of the plug-in estimator

$$\hat{\theta} \;=\; t(\hat{f})\,. \tag{4.1.4}$$

The plug-in principle works quite well, if $\hat{f}$ is representative for the distribution. Actually, it is the only available choice in most of the cases. And it is what we have used implicitly during the previous chapters.

4.2. Estimators and Statistical Uncertainties

Let us first assume that we have N independent realisations of a random variable X, which we call a *random sample* of size N. Again, for the forthcoming analysis to make sense we need to assume that the random sample is representative, which means that the properties of the underlying, unknown distribution are properly represented in $\hat{f}$. On such a random sample we want to estimate e.g. $\langle x^n \rangle$ of the underlying distribution f_X. Let θ be some unknown parameter of the distribution and $x = (x_1, \ldots, x_\mathrm{N})$ the random sample. Let t_N be a function of x which estimates θ from x, i.e.

$$\bar{\theta} \;=\; t_\mathrm{N}\,(x_1, \ldots, x_\mathrm{N}) \;\equiv\; t(\hat{f})\,. \tag{4.2.1}$$

We will use the bar ($\bar{\theta}$) to indicate the estimated quantity versus the true value that is written without the bar (θ). t is called statistics. The value of t_N can be considered as a random variable again:

$$T_\mathrm{N} \;=\; t_\mathrm{N}\,(X_1, \ldots, X_\mathrm{N}) \tag{4.2.2}$$

which we call *estimator*. Here, the X_i are now **iid** random variables. Such an estimator should fulfil certain conditions, namely that is is *consistent* and *unbiased*.

Definition 4.2.1: unbiased

The estimator T_N for parameter θ is called unbiased if

$$\mathbb{E}\,(T_N) \;=\; \mathbb{E}\,(t_N\,(X_1, \ldots, X_N)) = \theta\,. \tag{4.2.3}$$

The bias of an estimator T_N of θ is consequently defined as

$$\mathrm{Bias}(T_\mathrm{N}) \;=\; \mathbb{E}\,(T_N) - \theta\,. \tag{4.2.4}$$

Definition 4.2.2: consistent

The estimator T_N is called consistent if $\forall \epsilon > 0$:

$$\lim_{N\to\infty} \mathbb{P}\,(|T_N\,(x_1, \ldots, x_N) - \theta| > \epsilon) \;=\; 0\,, \tag{4.2.5}$$

i.e., the estimator converges stochastically against the true value.

Example 4.2.1

The function

$$t_{\bar{x}_\mathrm{N}}(x_1, ..., x_N) \;=\; \bar{x}_\mathrm{N} \;\stackrel{\mathrm{def}}{=}\; \frac{1}{N}\sum_i x_i \tag{4.2.6}$$

can be used to estimate the mean μ of a random sample. It represents the plug-in estimator for the expectation value Eq. 2.4.1. $\bar{x}_\mathrm{N}$ (later we will simply write $\bar{x}$) is unbiased

$$\mathbb{E}(T_{\bar{x}_\mathrm{N}}) \;=\; \mathbb{E}\left(\frac{1}{N}\sum_i X_i\right) \;=\; \frac{1}{N}\sum_i \mathbb{E}(X_i) \;=\; \mu\,. \tag{4.2.7}$$

Moreover,

$$\begin{aligned} \mathrm{Var}(T_{\bar{x}_\mathrm{N}}) &= \mathbb{E}[(T_{\bar{x}_\mathrm{N}} - \mathbb{E}(T_{\bar{x}_\mathrm{N}}))] \\ &= \mathrm{Var}\left(\frac{1}{N}\sum_i X_i\right) = \frac{1}{N^2}\sum_i \mathrm{Var}(X_i) = \frac{1}{N}\mathrm{Var}(X)\,. \end{aligned} \tag{4.2.8}$$

Therefore, the estimator converges stochastically against μ and it is consistent.

By the central limit theorem, $T_{\bar{x}_\mathrm{N}}$ is standard normal distributed for sufficiently large N. The width of this distribution is given by

$$\Delta_N \overset{\mathrm{def}}{=} \sqrt{\mathrm{var}\,(T_{\bar{x}_\mathrm{N}})} = \frac{\sigma_x}{\sqrt{N}}\,. \tag{4.2.9}$$

Δ_N defines the *standard error* of the estimator with σ_x the standard deviation of the distribution of X or simply of X. The meaning of Δ_N is that with 68% probability the true value μ lies in the interval $[\bar{x}_\mathrm{N} - \Delta_N, \bar{x}_\mathrm{N} + \Delta_N]$. $2\Delta_N$ represents 95% and $3\Delta_N$ already 99.7% probability. The intervals are also called *confidence intervals*. For general estimators T_N we define the standard error as

$$\mathrm{se}\,(T_N) = \mathrm{sd}\,(T_N) = \sqrt{\mathrm{Var}\,(T_N)}\,.$$

This definition for the standard error and the confidence intervals is meaningful only for normally distributed errors. However, with the notion of α–quantiles the definition can be extended to arbitrary, not necessarily symmetric probability distributions: the 68% confidence interval is then defined as the interval $[\mu_{16}, \mu_{84}]$ given by the 16– and the 84–quantile of the distribution. This interval is then no longer symmetric around the median $\mu_{\frac{1}{2}}$.

An unbiased estimator for the variance reads

$$\bar{\sigma}_N^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2\,. \tag{4.2.10}$$

It is obtained from Eq. 2.4.4 by the plug-in principle, apart from the factor $(N-1)^{-1}$. Also for $\bar{\sigma}_N$ we will drop the subscript soon, if it is clear from the context what the sample size is. The reason for the factor $(N-1)^{-1}$ is that we use $\bar{x}$ instead of the true mean value μ. So the N observations satisfy

$$\bar{x} - \frac{1}{N}\sum_i x_i = 0\,,$$

and, hence, they possess only $N-1$ degrees of freedom.

Proof:

Compute first

$$\begin{aligned}\left\langle (X_i - T_{\bar{x}_\mathrm{N}})^2 \right\rangle &= \left\langle (X_i - \mu - T_{\bar{x}_\mathrm{N}} + \mu)^2 \right\rangle \\ &= \langle (X_i - \mu)(T_{\bar{x}_\mathrm{N}} - \mu) \rangle - \frac{1}{N}\sigma_x^2 \\ &= \frac{1}{N}\sum_i \langle (X_i - \mu)(X_j - \mu) \rangle - \frac{1}{N}\sigma_x^2\end{aligned} \tag{4.2.11}$$

X_i and X_j are **iid** random variables, hence, in the sum only the terms with $i = j$ survive and evaluate to σ_x^2. Therefore,

$$\left\langle (X_i - T_{\bar{x}_\mathrm{N}})^2 \right\rangle = \frac{N-1}{N}\sigma_x^2 . \tag{4.2.12}$$

□

All this can be generalised to the multivariate case with independent random vectors $\vec{X} \in \mathbb{R}^n$ with the vector of means

$$\overline{\vec{X}}_N = \frac{1}{N}\sum_i \vec{X}_i \tag{4.2.13}$$

and the covariance matrix

$$\Sigma_N^2 = \frac{1}{N-1}\left(\vec{x} - \overline{\vec{x}}_N\right) \cdot \left(\vec{x} - \overline{\vec{x}}_N\right)^t . \tag{4.2.14}$$

Here we have dropped the bar over the Σ for better readability.

We add to this notion of error for general estimators T_N of a parameter θ by defining the *mean square error* (MSE) as follows

$$\mathrm{MSE}(T_\mathrm{N}) = \mathbb{E}[(T_\mathrm{N} - \theta)^2] \tag{4.2.15}$$

Theorem 4.2.1:

The mean square error of estimator T_N is given as

$$\mathrm{MSE}(T_\mathrm{N}) = \mathrm{Var}\,(T_\mathrm{N}) + \mathrm{Bias}\,(T_\mathrm{N})^2 . \tag{4.2.16}$$

Hence, for an unbiased estimator, MSE and variance are identical.

The proof follows from the definitions of variance and bias and from the properties of the expectation value. In particular, one needs $\mathbb{E}(\mathbb{E}(T_N)) = \mathbb{E}(T_N)$ or, more generally, $\mathbb{E}(\mathrm{const}) = \mathrm{const}$ and completing the square. In terms of the MSE, the *root mean square error* is defined as

$$\mathrm{RMSE}\,(T_\mathrm{N}) = \sqrt{\mathrm{MSE}(T_\mathrm{N})} . \tag{4.2.17}$$

In practice the RMSE differs from the standard error by the squared Bias. We will discuss in section 4.5 how to estimate Δ_{T_N} for general estimators T_N.

4.3. Linear Regressions

We consider the case of linear regressions. Let $Y_1, \ldots, Y_n$ be random variables. Suppose for $i = 1, \ldots, n$ and $k = 1, \ldots, p$

$$\mathbb{E}(Y_i) \;=\; t_{ik}\,\beta_k \tag{4.3.1}$$

and covariance C

$$\mathbb{E}\left[(Y_i - \mathbb{E}(Y_i))(Y_j - \mathbb{E}(Y_j))\right] \;=\; \mathbb{E}\left[(Y_i - t_{ik}\beta_k)(Y_j - t_{jk}\beta_k)\right] \;=\; C_{ij} \tag{4.3.2}$$

Here, $\beta = (\beta_1, ..., \beta_p)^t$ is the vector of linear parameters, t_{ki} are the independent (or explanatory) and $Y = (Y_1, ..., Y_n)^t$ the dependent variables. We allow the dependent variables to be correlated. Now we are interested in estimating β from the data. This is usually an overdetermined problem.

In matrix notation the equations from above read

$$\mathbb{E}(Y) \;=\; Z\beta\,, \tag{4.3.3}$$

$$\mathbb{E}\left[(Y - Z\beta)(Y - Z\beta)^t\right] \;=\; C\,, \tag{4.3.4}$$

with matrix notation $Z = t_{ik}$ and covariance C known. We now chose a basis in which the transformed variables are independent. Let D diagonalise C, i.e.

$$D\,C\,D^t = \mathbb{1}\,.$$

Write $X = DY$ and $W = DZ$. Multiply Eq. 4.3.3 and Eq. 4.3.4 from the left with D and Eq. 4.3.4 from the right with D^t

$$\mathbb{E}(X) \;=\; W\beta\,, \tag{4.3.5}$$

$$\mathbb{E}\left((X - W\beta)(X - W\beta)^t\right) \;=\; \mathbb{1}\,. \tag{4.3.6}$$

Thus, the new random variables $X = DY$ are statistically independent and homoscedastic. In this basis we can proceed and formulate an equivalent minimisation problem for the residuals, i.e. a minimisation problem for the quadratic form

$$F(\beta) \;=\; \mathbb{E}(X)\mathbb{E}(X)^t + \beta^t W^t W\beta - 2(W\beta)^t\mathbb{E}(X)\,. \tag{4.3.7}$$

The neccessary condition for a minimum reads

$$\mathrm{grad}_\beta\, F(\beta) \;=\; 0\,, \tag{4.3.8}$$

which leads to the normal equation

$$A\,\beta \;=\; c \tag{4.3.9}$$

with $A = W^tW$ and $c = W^t\,\mathbb{E}(X)$. We assume that W has maximal rank. As a consequence A is positive definit. Therefore, since the second derivative of F with

respect to β equals A, also the sufficient condition for a minimum is fulfilled. The estimator for β – denoted as T_β – is the solution of the normal equation

$$A\,T_\beta \;=\; W^t\,X\,, \tag{4.3.10}$$

cf. Eq. 4.3.9. Inserting all the definitions again one obtains (equivalent to Eq. 4.3.5)

$$Z^t D^t D Z T_\beta \;=\; Z^t D^t D\,Y\,.$$

Since $C^{-1} = D^t D$, the solution of the normal equation defines the estimator T_β as

$$T_\beta \;=\; (Z^t C^{-1} Z)^{-1} Z^t C^{-1} Y\,. \tag{4.3.11}$$

One can easily verify that

$$\mathbb{E}(T_\beta) \;=\; \beta$$

and, hence, T_β is an unbiased estimator. Moreover,

$$\mathrm{Var}\,(T_\beta) \;=\; \mathbb{E}\left[(T_\beta - \beta)(T_\beta - \beta)^t\right] \;=\; (Z^t C^{-1} Z)^{-1}\,. \tag{4.3.12}$$

The usual least–squares estimate (i.e. without correlation taken into account)

$$T_L \;=\; (Z^t Z)^{-1} Z^t Y \tag{4.3.13}$$

is unbiased as well. It can be obtained from Eq. 4.3.11 by setting $C = \mathbb{1}$. It not only neglects correlation, but also puts equal weight on every $\mathbb{E}(Y_i)$. However, it usually possesses larger variance. T_β minimises the quadratic form

$$\chi^2 \;\stackrel{\mathrm{def}}{=}\; (\bar{y} - Z\tilde{\beta})^t\, C^{-1}\, (\bar{y} - Z\tilde{\beta})\,,$$

with respect to $\tilde{\beta}$ and in case $Y_1, \ldots, Y_n$ have a joint normal distribution, T_β is the maximum–likelihood estimate of β, which we describe in the following section.

In practice, following the *plug-in principle*, the estimator T_β (or the estimator T_L) is used by replacing the expectation values of the random variables Y and the covariance matrix $\sigma^2 C$ by their corresponding estimates from the data. Hence, for instance the standard error of the parameters β can be computed from Eq. 4.3.12 by replacing C by its estimates $\hat{C}$ from the observations for Y.

To give a concrete example, lets have a look at the case with $p = 2$ and a single explanatory variable x, i.e.

$$\mathbb{E}\,(Y_i) \;=\; \beta_1 + \beta_2 x_i \;=\; (Z\beta)_i\,,$$

with

$$\beta = \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}\,, \qquad \text{and} \qquad Z = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}\,.$$

To compute the variance of T_β, we need to compute

$$(Z^t C^{-1} Z)^{-1} ,$$

which is not so simple for general C. Therefore, lets assume here that Y_i and Y_j are not correlated for $i \neq j$, i.e. $C = \mathrm{diag}\,(\sigma_1^2, ..., \sigma_n^2)$. In this case we have

$$Z^t C^{-1} Z \;=\; \begin{pmatrix} \sum \sigma_i^{-2} & \sum x_i/\sigma_i^2 \\ \sum x_i/\sigma_i^2 & \sum x_i^2/\sigma_i^2 \end{pmatrix} .$$

Inverting this 2×2 matrix leads to

$$(Z^t C^{-1} Z)^{-1} \;=\; \frac{1}{(\sum \sigma_i^{-2})(\sum x_i^2/\sigma_i^2) - (\sum x_i/\sigma_i^2)^2} \begin{pmatrix} \sum x_i^2/\sigma_i^2 & -\sum x_i/\sigma_i^2 \\ -\sum x_i/\sigma_i^2 & \sum \sigma_i^{-2} \end{pmatrix} .$$

Hence, the standard error of the estimate T_β of e.g. the estimated slope β_2 is given by

$$\mathrm{se}(\hat\beta_2) \;=\; \sqrt{[(Z^t C^{-1} Z)^{-1}]_{22}} \;=\; \sqrt{\frac{\sum \hat\sigma_i^{-2}}{(\sum \hat\sigma_i^{-2})(\sum x_i^2/\hat\sigma_i^2) - (\sum x_i/\hat\sigma_i^2)^2}} .$$

For the *homoscedastic* case with $\sigma_i = \sigma$ for all $i = 1, ..., n$, this reduces to

$$\mathrm{se}(\hat\beta_2) \;=\; \sqrt{\frac{n\,\hat\sigma^2}{n \sum x_i^2 - (\sum x_i)^2}} .$$

As mentioned before, we will later introduce the *bootstrap*, which allows to compute errors in a very general way.

For the example of a linear regression in the *heteroscedastic* but with **iid** data points we can also show what the MSE is. Assume we have found best fit parameters $\hat\beta$ from solving the normal equation from above. Then, the *(weighted) mean square error* of the fit is given by

$$\mathrm{MSE} \;=\; \frac{1}{n} \sum_{i=1}^{n} \frac{(\bar y_i - t_{ik} \hat\beta_k)^2}{\hat\sigma_i^2} \;\equiv\; \chi^2 , \tag{4.3.14}$$

with the sum over k to be understood. What is called the *(weighted) residual sum of squares* (RSS) is given by

$$\mathrm{RSS} \;=\; \sum_{i=1}^{n} \frac{(\bar y_i - t_{ik} \hat\beta_k)^2}{\hat\sigma_i^2} .$$

With the RSS we can write the MSE as follows

$$\mathrm{MSE} \;=\; \frac{1}{n}\,\mathrm{RSS} .$$

The *(weighted) residual standard error* (RSE) is also in terms of the RSS defined as

$$\mathrm{RSE} \;=\; \sqrt{\frac{\mathrm{RSS}}{n-p}} .$$

It differs from the RMSE by the fact that one devides by the degrees of freedom $n - p$ instead of the sample size n. RSE represents the unbiased estimate of error variance. We remark that sometimes the term MSE is used for the RSE.

4.4. Maximum–Likelihood–method

Consider some explanatory variable at discrete points t_i $(i = 1, \ldots, n)$ (think about time) and for each i we have measured an independent random variable Y_i N-times. For each i we can give an estimate for the mean $\overline{y}_i$ and the standard error Δ_i. For this we assume that N is large enough that the central limit theorem can be applied. We assume a functional form for Y as

$$\mathbb{E}(Y_i) \;=\; f\,(t_i;\,\lambda)$$

depending on a set of parameters $\lambda = (\lambda_1, \ldots)$. The task is to estimate the parameters λ from our set of measurements. We call t the explanatory variable and f is also called the model.

The posterior probability for the function f given a set of parameters λ' and for a single t_i is then

$$\mathbb{P}\left(\mu_i = f\,(t_i, \lambda')\right) \;=\; \frac{1}{\Delta_i\sqrt{2\pi}}\exp\left(-\frac{\left(f\,(t_i;\,\lambda') - \overline{y}_i\right)^2}{2\Delta_i^2}\right) \tag{4.4.1}$$

and for the set t_i with $i = 1, ..., n$

$$L\,(\lambda') \;\propto\; \prod_{i=1}^{n}\exp\left(-\frac{\left(f\,(t_i;\,\lambda') - \overline{y}_i\right)^2}{2\Delta_i^2}\right). \tag{4.4.2}$$

$L(\lambda)$ is called *Likelihood function*. Finding the most probable parameter set $\overline{\lambda}$ means maximising $L(\lambda)$ with respect to the parameters λ

$$\bar{\lambda} \;=\; \mathrm{argmax}_\lambda\, L(\lambda)\,. \tag{4.4.3}$$

Equivalently, we can use $l = -\ln\left(L(\lambda)\right)$ and minimise instead:

$$\chi^2(\lambda) \;=\; \sum_i \frac{\left(f\,(t_i, \lambda) - \overline{y}_i\right)^2}{\Delta_i^2}\,. \tag{4.4.4}$$

The Likelihood function can also be defined in a more general case, as the following conditional probability:

$$L(\lambda) \;=\; \prod_i \mathbb{P}\,(y_i|\lambda) \tag{4.4.5}$$

$\chi^2(\lambda)$ follows a χ^2–distribution for dof $= n - m$ degrees of freedom if there are m parameters. For the special case we are looking at here this procedure is also called *method of least squares*.

The χ^2 random variable follows a χ^2–distribution as a sum of squared normally distributed random variables.

Definition 4.4.1: χ^2–distribution

The sum of squares

$$\chi^2 \;=\; \sum_{i=1}^{k} \left(\frac{X_i - \mu_i}{\sigma_i} \right)^2$$

of independent, normally distributed random variables $X_1, ..., X_k$ with means μ_i and variances σ_i^2 is chi^2–distributed with k degrees of freedom. The χ^2–distribution for k degrees of freedom has the following cdf

$$\begin{aligned} F_{\chi^2}(k,x) \;&=\; \int_0^x \frac{1}{\Gamma\left(\frac{k}{2}\right) 2^{k/2}} y^{k/2-1} \mathrm{e}^{-y/2}\, \mathrm{d}y\,, \qquad x > 0 \\ &=\; \Gamma\left(\frac{k}{2}, \frac{x}{2}\right)\,, \end{aligned} \tag{4.4.6}$$

where $\Gamma(n,x)$ is the incomplete Γ–function. Mean and variance are given by

$$\mu = k\,, \qquad \sigma^2 = 2k\,.$$

Given $\Gamma(n,x)$ we can compute the $(1-\alpha)$ quantile $\chi^2_{\mathrm{dof},1-\alpha}$. This leads to an estimate of the quality of the fit via the p–value

$$p = \; 1 - F_{\chi^2}\left(\mathrm{dof}, \chi^2\right) \;=\; 1 - \Gamma\left(\frac{\mathrm{dof}}{2}, \frac{\chi^2}{2}\right)\,. \tag{4.4.7}$$

The p–value represents the probability for finding a larger χ^2 value in a next, independent experiment under the assumption that the model is correct. It is easy to see that we expect a value of $p = 0.5$ for a statistically meaningful fit. However, looking at the χ^2 distribution all values in between 0.01 and 0.99 are acceptable. The p–value can also be used to reject the hypothesis that the variance in the data is explained by the model.

4.4.1. Correlated χ^2

So far we assumed the data to be independent. Now we consider the situation that the X_i are correlated, but still Gaussian. In contrast to Eq. 4.4.2, we now need a general Gaussian surface. Let us start with the joint pdf of a multivariate Gaussian distribution. Denoting with $X = (X_1, X_2, ..., X_n)^t$ the column vector, the joint pdf has the form

$$f_{X_1,X_2}(x_1,x_2) = \frac{1}{2\pi(\det C)^{1/2}} \exp\left(-\frac{1}{2} X^t\, C^{-1}\, X\right)\,. \tag{4.4.8}$$

with $C = \mathrm{Cor}(X_1, ..., X_n)$. Therefore, the likelyhood takes now the form

$$L(\lambda) = \mathcal{N} \exp\left(-\frac{1}{2} \sum_{i,j} (f(t_i,\lambda) - \bar{y}_i) M_{ij} (f(t_j,\lambda) - \bar{y}_j)\right) \tag{4.4.9}$$

with

$$M \stackrel{\text{def}}{=} C^{-1} .$$

C is the variance–covariance matrix defined as

$$C = \frac{1}{N} \Sigma_N^2 , \tag{4.4.10}$$

with Σ_N defined in Eq. 4.2.14. This normalisation guarantees that on the diagonals of C_{ij} we have Δ_i^2. Again for N large enough, we have to minimise

$$\chi^2(\lambda) = \sum_{ij} (\bar{y}_i - f(t_i, \lambda)) \; M_{ij} \; (\bar{y}_j - f(t_j, \lambda)) , \tag{4.4.11}$$

to find the best fit parameters. χ^2 is again χ^2 distributed with $(n-m)$ degrees of freedom and a corresponding p-value can be computed as introduced above.

Care needs to be taken in treating the inversion of C. This might be a numerically rather unstable problem, because second moments of distributions require already a significant amount of statistics, i.e. large enough N to be estimated properly. If that is not the case, the matrix C might have artificially small eigenvalues leading to numerical instabilities. See for instance Ref. [12] for possible solutions to this problem.

4.4.2. χ^2 minimisation with errors on both axis

So far we have assumed that only the x_i are random variables but the explanatory variable t_i is fixed to certain values. This is clearly not the most general case and for many problems it is clearly not the case. Hence, we assume now to have likely not independent random variables Y_i and X_i for $i = 1, ..., n$ and we join them into one random vector by defining $\vec{Z} = (\vec{X}, \vec{Y})$. Now we extend our parameters by n new parameters $t_1, ..., t_n$

$$\hat{\lambda} \stackrel{\text{def}}{=} (t_1, ..., t_n, \lambda)$$

and define a new function

$$\vec{f}(\hat{\lambda}) \stackrel{\text{def}}{=} (t_1, ..., t_n, f(t_1; \lambda), ..., f(t_n; \lambda)) .$$

We will assume a linear model for the X_i, which leads to the definition of a new χ^2 function

$$\chi^2(\hat{\lambda}) \stackrel{\text{def}}{=} \left(\vec{z} - \vec{f}(\hat{\lambda})\right) \; M \; \left(\vec{z} - \vec{f}(\hat{\lambda}\right) . \tag{4.4.12}$$

For the minimisation, the parameters λ and the parameters t_i can be adjusted. Note that the number of degrees of freedom did not change when the number of parameters was increased, because the number of data points increased by the same amount. The matrix M is defined like above, but of course for the random vector $\vec{Z}$. Therefore, M is now a $2n \times 2n$ matrix.

4.4.3. Incorporating prior knowledge in a fit

In what we discussed so far, we have always assumed that we do not have any knowledge on the parameters and, thus, they are completely determined from the data. This is also known as the so-called *frequentists approach*. This leads to the maximum likelihood minimisation, which represents the probability of the data given the model (or the parameters). However, it occurs frequently that we do have prior knowledge for one or more of the parameters, which we would like to include in our statistical inference. An example would be that e.g. parameter λ_1 is known from other experiments/investigations to take values $\tilde{\lambda}_1 \pm \Delta\tilde{\lambda}_1$ with a GAUSSIAN error. But it could also be some less specific knowledge.

To include such prior knowledge in the analysis we recall the theorem 2.1.2 of BAYES stating

$$\mathbb{P}(A|B) \;=\; \frac{\mathbb{P}(B|A)}{\mathbb{P}(B)}\,\mathbb{P}(A)\,. \tag{4.4.13}$$

In words: the probability of A given B is equal to the probability of B given A times the probability of A devided by the probability of B. $\mathbb{P}(A|B)$ was called *posterior probability*, $\mathbb{P}(B|A)$ was called *likelihood* and $\mathbb{P}(A)$ *prior probability*.

In our case, A would correspond to the parameters or the model, and B to the data. Thus, in the BAYESIAN *approach* it is actually possible to compute the probability for the parameters given the data. And it is given by the probability of the data given the parameters, which we also call likelihood and which we used previously, times the prior probability of the data. The unconditional probability for the data is to be understood as fixed and, therefore, a normalisation constant.

We can connect this to our original maximum likelyhood method by assuming a flat prior probability containing no information. In this case the posterior probability is up to constants equal to the likelihood. The alternative is an informative prior, like the example $\tilde{\lambda}_1 \pm \Delta\tilde{\lambda}_1$ given above, for which we need to specify the prior probability.

Example 4.4.1

For the example of a GAUSSIAN error for $\tilde{\lambda}_1$ this can be done

$$\mathbb{P}(\lambda_1) \;\propto\; \exp\left(-\frac{(\lambda_1-\tilde{\lambda}_1)^2}{2(\Delta\tilde{\lambda}_1)^2}\right)\,.$$

So, in fact, for data with GAUSSIAN errors we need to minimise the following augmented χ^2–function

$$\chi^2_{\text{aug}} \;=\; \chi^2(\lambda) + \frac{(\lambda_1-\tilde{\lambda}_1)^2}{(\Delta\tilde{\lambda}_1)^2} \tag{4.4.14}$$

with $\chi^2(\lambda)$ the original χ^2 function. Hence, in this case one can also think of $\tilde{\lambda}_1$ as an additional data point, which is included in the likelihood function as any of the

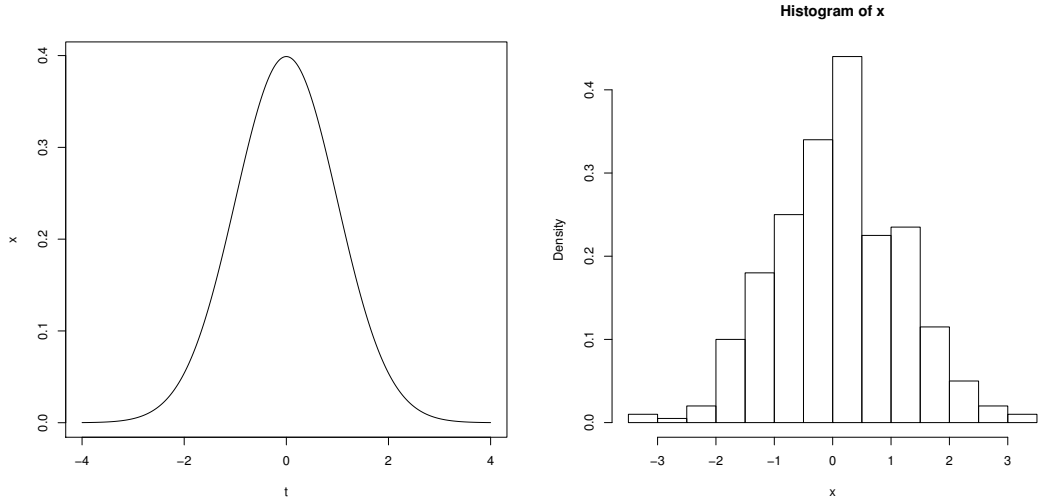


Figure 4.5.1.:
The pdf of $f = \mathcal{N}_{0,1}$ is plotted in the left panel, while one particular realisation of the empirical distribution function $\hat{f}$ is shown in the right panel.

other data points.

The Bayesian approach is applicable in a wider range of problems which we cannot discuss here.

4.5. The Bootstrap

We have discussed in the previous sections how to estimate the standard error Δ_N based on a finite random sample $x_1, ..., x_N$ of size N. Implicitly, we have discussed only simple observables and on purpose we did not yet say anything yet e.g. on computing errors for parameters of a χ^2 fit. This is still usually done by error propagation using – often unaware – certain assumptions.

In this section we will discuss a very general method for determining the errors based on random samples $x_1, ..., x_N$, the so-called *bootstrap* method [5]. Here, we do not specify whether we deal with the uni- or the multi-variate case. So x_i for given i could be an n-tuple of possibly correlated random variables.

The bootstrap is very general because – if used properly – it does require only minimal assumptions on the random sample. In particular, it is applicable for Gaussian and non-Gaussian distributions. It works for primary and derived quantities, and automatically takes correlation into account. The bootstrap can be applied in an automated way. However, it comes for the price that it is computationally demanding, but with nowadays computing power this is no longer a serious issue.

Coming back to the question of estimating uncertainty on $\hat{\theta}$, we would proceed as follows if we had infinite resources (including time) generate R random samples of length

N drawn from f

$$f \quad \rightarrow \quad x^1, x^2, ..., x^R$$

and compute $\bar{\theta}^j$ on each random sample x^j, $j = 1, ..., R$, using the plug-in-principle. Now the standard error of $\hat{\theta}$ can be estimated from the standard deviation $\mathrm{sd}(\bar{\theta}^1, ..., \bar{\theta}^R)$ over the R replications of $\bar{\theta}$. However, this is for the really interesting problems usually not feasible.

The bootstrap circumvents this problem by replacing the pdf f by the *empirical probability distribution* $\hat{f}$, which is estimated from the random sample $x_1, ..., x_N$. Lets first define the *indicator* function

Definition 4.5.1: indicator function

$$\mathbb{1}_{\text{condition}} \;=\; \begin{cases} 1 & \textit{condition fulfilled} \\ 0 & \textit{else} \end{cases} \tag{4.5.1}$$

The empirical distribution $\hat{f}$ is obtained from the data by assigning the probability

$$\mathbb{P}(A) \;=\; \frac{\#\{x_i \in A\}}{N} \;=\; \sum_{i=1,...,N} \mathbb{1}_{x_i \in A} \tag{4.5.2}$$

to a set A in sample space. Eq. 4.5.2 represents nothing but a histogram, as also illustrated in Figure 4.5.1. The corresponding cdf is then given as

$$\hat{F}(x) \;=\; \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{x_i \leq x} \,. \tag{4.5.3}$$

From the introduction it should now be clear how to proceed to estimate the error for $\hat{\theta}$. We generate R bootstrap samples $x^{\star 1}, ..., x^{\star R}$ from $\hat{f}$, compute R bootstrap estimates $\hat{\theta}^{\star 1}, ..., \hat{\theta}^{\star R}$ and estimate the standard error from the standard deviation over these R bootstrap samples. In single steps this amounts to

1. draw R random samples from $\hat{f}$ by randomly sampling from $x_1, ..., x_N$

$$\hat{f} \quad \rightarrow \quad (x_1^{\star b}, ..., x_N^{\star b}) = (x_{\sigma(1)}, ..., x_{\sigma(N)}) \,. \tag{4.5.4}$$

   Here $\sigma(i)$, $i = 1, ..., N$ is a random permutation drawn from a discrete uniform distribution L_N with replacement. So, for instance this could be implemented by setting for all $b = 1, ..., R$ and all $i = 1, ..., N$

$$\begin{aligned} u \;\sim\; \mathcal{U}_{[0,1]} \,, \qquad & k = \lfloor u \cdot N \rfloor + 1 \,, \\ & \Rightarrow x_i^{\star b} \;=\; x_k \,, \end{aligned} \tag{4.5.5}$$

   where $\mathcal{U}_{[0,1]}$ is the continuous uniform distribution over the interval $[0, 1]$. Each $x^{\star b}$ for $b = 1, ..., R$ we call a bootstrap sample.

2. evaluate
$$\hat{\theta}^{\star b} = t(x^{\star b}) \,, \qquad b = 1, ..., m \,.$$
Each $\hat{\theta}^{\star b}$ we call a bootstrap replication of $\hat{\theta}$.

3. the estimate of the standard error is then given by the standard deviation over the bootstrap replications
$$\Delta^{\star}_{R} \;=\; \mathrm{sd}_R(\theta^{\star b}) \;=\; \sqrt{\frac{\sum_b (\hat{\theta}^{\star b} - \bar{\theta}^{\star})^2}{(R-1)}} \,, \tag{4.5.6}$$
with the bootstrap mean
$$\bar{\theta}^{\star} \;=\; \frac{1}{R} \sum_{b=1}^{R} \hat{\theta}^{\star b} \,. \tag{4.5.7}$$

4. the bootstrap bias is given by
$$\mathrm{Bias}^{\star}_{\theta} \;=\; \bar{\theta}^{\star} - \bar{\theta}$$
which should fulfil $\mathrm{Bias}^{\star}_{\theta} \ll \Delta^{\star}_{R}$ in practice.

So far we have implicitly assumed that all errors are normally distributed. However, this does not need to be the case. If the errors are not normally distributed or the distribution is not known the standard deviation over the bootstrap samples needs only be replaced by the 68% confidence interval of the empirical distribution of $\hat{\theta}$, i.e. $\hat{f}_{\hat{\theta}}$. The α–quantile of $\hat{f}_{\hat{\theta}}$ can be estimated from
$$x_{\alpha} \;=\; \sup_{x_1,...,x_N} \{x_i : \hat{F}(x_i) < \alpha\} \,. \tag{4.5.8}$$

Alternatives could be to interpolate between the infimum of the set $\{x_i : \hat{F}(x_i) < \alpha\}$ and the supremum of the set $\{x_i : \hat{F}(x_i) > \alpha\}$ over $x_1, ..., x_N$. In particular for smallish N the estimate of x_α is not very precise. Of course, x_α as a function of α it is non-continuous per definition.

Having introduced this, it is important to always review the distribution of $\hat{\theta}^{\star}$. If the distribution is normal, this is most conveniently done by producing a so-called *QQ-plot*: the empirical quantiles of $\hat{\theta}^{\star}$ standardised by the bootstrap mean and error are plotted versus the theoretical quantiles of the standard normal distribution $\mathcal{N}_{0,1}$. The resulting points should be close to the bisecting line. Depending on N, the tails of the empirical distribution are often not sampled very well, but around the mean the agreement with the theoretical quantiles should be good. See section 4.8.1 for an example.

Finally, which value of R is reasonable? Since the bootstrap is stochastic method, the uncertainty on the bootstrap error decreases like $1/\sqrt{R}$. In general it is said that $R = 4N$ is a save choice, however, nowadays much larger R can be afforded without any problem.

4.5.1. Parametric Bootstrap

So far we have estimated the empirical distribution function directly from the data by sampling with replacement. There is another approach, called *parametric bootstrap*, for which one first determines the parameters of the distribution from the data and uses those to generate bootstrap samples.

To be precise, lets assume it is known what the distribution of the random sample $(x_1, ..., x_N)$ is. Lets denote it by $f_\kappa(x)$, where κ is the set of parameters of the distribution. For the normal distribution these are the mean and the variance. Next we fit $f_\kappa(x)$ to the empirical distribution of $x_1, ..., x_N$, possibly by a χ^2–fit to determine the best fit parameters $\hat{\kappa}$. Now we replace step one from above by

1′. draw R random samples from $f_{\hat{\kappa}}$

$$f_{\hat{\kappa}} \quad \to \quad (x_1^{\star b}, ..., x_N^{\star b}) \,, \qquad b = 1, ..., R \,. \tag{4.5.9}$$

Of course, for this to be useful the distribution needs to be known and one needs to be able to draw random samples from this distribution. Parametric bootstrap can be for instance applied when the data sample is not available, but only the mean μ and the standard error σ, under the assumption that the errors are normally distributed. Then, bootstrap samples are drawn from

$$x_i^{\star b} \quad \sim \quad \mathcal{N}_{\mu,\sigma} \,, \qquad b = 1, ..., R \,;\; i = 1, ..., N \,.$$

These bootstrap samples may then be used in further analysis.

4.5.2. Bootstrapping a χ^2–fit

We will now discuss the application of the bootstrap to a general χ^2–fit. As before, t will be the explanatory variable. Data has been measured for several t-values $t_1, ..., t_n$ and will be denoted by $y(t_1) \equiv y_1, ..., y(t_n) \equiv y_n$. We will understand $\vec{y} = y_1, ..., y_n$ as a multi-variate random vector of size n. Each y_i is assumed to have Gaussian error distribution. The means $\bar{y}_1, ..., \bar{y}_n$ can be estimated using the standard estimator Eq. 4.2.6. With the tools provided in this section we can estimate the standard error for each y_i using the bootstrap which will lead to standard errors $\Delta_1, ..., \Delta_n$ determined from R bootstrap samples $y_1^{\star b}, ..., y_n^{\star b}$ with $b = 1, ..., R$. It is important to have identical number of bootstrap replications for all i–values for the following to work. So, while in principle all y_i might stemm from a samples of different sizes N_i, the number of bootstrap samples must be chosen to be identical.

Alternatively, we have given estimates of means $\bar{y}_1, ..., \bar{y}_n$ and standard errors $\Delta_1, ..., \Delta_n$. Using the parametric bootstrap one generates now the bootstrap samples $y_1^{\star b}, ..., y_n^{\star b}$ with $b = 1, ..., R$ from normal distributions $\mathcal{N}_{\bar{y}_i, \Delta_i}$. This corresponds to the parametric bootstrap approach. The big advantage of using the original random sample instead of a parametric bootstrap is that possible correlations between Y_i and Y_j are maintained automatically.

Note that the case where every Y_i has its own uncertainty and, therefore, weight, is called *heteroscedastic* case, while *homoscedasticity* means that all Y_i have identical uncertainties. In the latter case a weighted χ^2 minimisation leads to identical results as an unweighted one.

The goal is to fit a general, non-linear model $f(\lambda, t)$ to our data via parameters $\lambda = (\lambda_1, ..., \lambda_m)$ and estimate the best fit parameters λ' and their standard errors using a χ^2 minimisation. The fist step in this procedure consists in estimating the variance-covariance matrix of $\vec{y} = y_1, ..., y_n$ in case of correlation. This can be done using the bootstrap estimator

$$C^\star \;=\; \frac{1}{R-1}\left(\vec{y}^\star - \overline{\vec{y}}^\star_R\right)\cdot\left(\vec{y}^\star - \overline{\vec{y}}^\star_R\right)^t . \tag{4.5.10}$$

One can easily verify that with this definition in the case of no correlation between different t–values the matrix $M = (C^\star)^{-1}$ has the form $M = \mathrm{diag}\,(1/\Delta_1^2, ..., 1/\Delta_n^2)$, as expected.

The next step is to perform a χ^2 fit to the what we call *original data* with variance-covariance matrix $C^\star$. The original data are the means estimated on the not-bootstrapped data. This fit will give best fit parameters λ' and one χ^2–value, lets denote it as Q'. Next we perform a χ^2 fit on all bootstrap samples $y_1^{\star b}, ..., y_n^{\star b}$ with $b = 1, ..., R$. One question here is which covariance matrix to use. There are basically two option: the so-called *frozen covariance* approach, where the covariance matrix is kept fixed equal to $C^\star$ for all fits on the bootstrap samples. The second one is to perform a double bootstrap to determine $C^{b\star}$ on each sample. We follow here the frozen covariance matrix approach, which usually works quite well.

From the R fits we obtain parameters $\lambda^{\star b}$ and χ^2 values $Q^{\star b}$. The bootstrap estimate of the errors of the best fit parameters is then given by (independently of the frozen covariance approach)

$$\Delta^\star_{\lambda_i} \;=\; \mathrm{sd}_R\left(\lambda_i^{\star b}\right) , \qquad i = 1, ..., m$$

assuming normally distributed errors, which needs to be checked. If it is not the case, the confidence intervals can be estimated as discussed above.

What is the distribution of the bootstrapped χ^2–values $Q^\star$? For the frozen covariance approach they are not χ^2 distributed as one may think intuitively. They are distributed according to a *non-central* χ^2–distribution.

Definition 4.5.2: non-central χ^2–distribution

Let $X_1, ..., X_n$ be independent normally distributed random variables with means μ_i and variances σ_i^2 for $i = 1, ..., n$. Then the sum

$$\chi^2_{\mathrm{nc}} \;=\; \sum_{i=1}^{n}\left(\frac{X_i}{\sigma_i}\right)^2$$

is distributed according to the the non-central χ^2 distribution with k degrees of freedom, non-centrality parameter $\mathrm{ncp} = \sum_i \mu_i^2$ *with pdf given by*

$$f_{\chi^2_{\mathrm{nc}}}(k, \mathrm{ncp}, x) \;=\; \frac{1}{2}\mathrm{e}^{-(x+\mathrm{ncp})2}\left(\frac{x}{\mathrm{ncp}}\right)^{k/4-1/2} \mathcal{I}_{k/2-1}\left(\sqrt{\mathrm{ncp}\cdot x}\right) \tag{4.5.11}$$

with $\mathcal{I}_k$ the modified Bessel*–function of the first kind of order k. Mean and variance are given by*

$$\mu = k + \mathrm{ncp}\,, \qquad \sigma^2 = 2(k + 2\mathrm{ncp})\,.$$

The reason is that the bootstrap samples for each data point vary around the estimated mean $\bar{x}_i$ and not the true mean μ_i. Therefore, the expected distribution is a non-central χ^2 distribution with non-centrality parameter given by the original χ^2–value Q', but the same number of degrees of freedom $\mathrm{dof} = n - m$.

We remark that in the literature there are other ways discussed to bootstrap (linear) regressions: for linear regressions usually the residuals are bootstrapped. The closest to what we have described here is called *wild bootstrap* [20].

4.6. Estimating the error of the error

We have discussed so far how to estimate the statistical uncertainty of some statistics $t(x_1, ..., x_N)$. Since this statistical error is itself estimated from the data it is useful to also estimate the error of the error. Given the bootstrap method, there is a natural way to determine the error on the error. Namely by performing a bootstrap analysis on the bootstrap samples itself. This is called *double bootstrap*. However, the double bootstrap can be rather computer time intensive, because it involves a full bootstrap analysis on each of the R bootstrap samples, as we will see next.

A less resource intensive method is called *jackknife-after-bootstrap*, which we will introduce afterwards.

4.6.1. Double bootstrap

In order to estimate the error of the bootstrap error one needs to estimate the standard deviation on each bootstrap sample separately. This can be achieved by a *double bootstrap* procedure: for every bootstrap sample $x^{\star b}$, $b = 1, \ldots, R$ one generates R' bootstrap samples $x^{\star\star bb'}$, $b' = 1, \ldots, R'$ by uniformly sampling from $x^{\star b}$ with replacement.

For each $b = 1, \ldots, R$ one can now estimate $\Delta^{\star\star b}$ from

$$\Delta^{\star\star b}_{R'} \;=\; \mathrm{sd}_{R'}(x^{\star\star bb'})\,.$$

The double bootstrap estimate of standard error on $\Delta^\star$ is then given as

$$\Delta^{\star\star}_{RR'}(\Delta^\star) \;\overset{\mathrm{def}}{=}\; \mathrm{sd}_R(\Delta^{\star\star b}) \;=\; \sqrt{\frac{\sum_{b=1}^{R}(\Delta^{\star\star b}_{R'} - \bar{\Delta}^{\star\star}_{R'})^2}{R-1}}\,. \tag{4.6.1}$$

This procedure can be generalised directly for functions other than the standard deviation. One obvious example is the covariance matrix needed for χ^2–fits as discussed previously. It can be computed using the double bootstrap procedure and the frozen covariance matrix can be replaced by the covariance matrix computed on each bootstrap sample.

Clearly, the computing resources required for the double bootstrap scale like $R \cdot R'$, which is the major downside of this method. This is even more true when a tripple bootstrap is attempted. However, it is certainly the most direct way to estimate statistical uncertainties of error estimates.

4.6.2. Jackknife-after-bootstrap

The *jackknife* [15, 18] is an alternative resampling method predating the bootstrap. Suppose we have a sample $(x_1, \ldots, x_N)$ and an estimator $\hat{\theta} = t(x_1, \ldots, x_N)$. The ith jackknife sample is then generated as

$$x_{(i)} \;=\; (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N)\,, \qquad i = 1, \ldots, N\,. \tag{4.6.2}$$

The jackknife replication of θ is defined as

$$\hat{\theta}_{(i)} \;\stackrel{\text{def}}{=}\; t(x_{(i)}) \tag{4.6.3}$$

and we define the jackknife mean as

$$\bar{\theta}_{(\cdot)} \;\stackrel{\text{def}}{=}\; \frac{1}{N}\sum_{i=1}^{N} \hat{\theta}_{(i)}\,. \tag{4.6.4}$$

Given these definitions, the jackknife estimate of standard error is usually written as

$$\Delta_N^{\text{jack}} \;\stackrel{\text{def}}{=}\; \sqrt{\frac{N-1}{N}\sum_{i=1}^{N}\left(\hat{\theta}_{(i)} - \bar{\theta}_{(\cdot)}\right)^2}\,. \tag{4.6.5}$$

The somewhat unexpected inflation factor $\sqrt{N-1}$ comes from the fact that on average $\hat{\theta}_{(i)} - \bar{\theta}_{(\cdot)}$ will be much smaller than its bootstrap counterpart $\hat{\theta}^{\star,b} - \bar{\theta}^{\star}$ used in Eq. 4.5.6. Of course, for N large enough we may equally well define

$$\Delta_N^{\text{jack}\prime} \;\stackrel{\text{def}}{=}\; \sqrt{\sum_{i=1}^{N}\left(\hat{\theta}_{(i)} - \bar{\theta}_{(\cdot)}\right)^2}\,, \tag{4.6.6}$$

with no material difference for the estimate. In fact, we define the jackknife estimate of covariance of two estimators $\hat{\theta}$ and $\hat{\eta}$

$$\text{Cov}^{\text{jack}}(\theta, \eta) \;\stackrel{\text{def}}{=}\; \sum_{i=1}^{N}(\hat{\theta}_{(i)} - \bar{\theta}_{(\cdot)}) \cdot (\hat{\eta}_{(i)} - \bar{\eta}_{(\cdot)})\,. \tag{4.6.7}$$

The factor $\sqrt{(N-1)/N}$ can be understood by considering the usual estimator for the mean

$$\hat{\theta} = \frac{1}{N}\sum x_i \,.$$

In this case one verifies easily that the jackknife mean is identical to the usual, unbiased and consistent estimator of mean

$$\bar{\theta}_{(\cdot)} \;=\; \bar{\theta}\,.$$

Using this, we can write

$$\begin{aligned}\left(\hat{\theta}_{(i)} - \bar{\theta}_{(\cdot)}\right) &= \left(\frac{1}{N-1}\sum_{j=1,j\neq i}^{N} x_j - \frac{1}{N}\sum_{j=1}^{N} x_j\right) \\ &= \frac{1}{N(N-1)}\left(x_1 + x_2 + \ldots + x_{i-1} + x_{i+1} + \ldots x_N - (N-1)x_i\right) \\ &= \frac{1}{N-1}\left(\bar{x} - x_i\right)\,.\end{aligned} \tag{4.6.8}$$

Inserting this in Eq. 4.6.5, we find for the jackknife estimate of standard error

$$\begin{aligned}\Delta_N^{\text{jack}} &= \sqrt{\frac{N-1}{N}\sum_{i=1}^{N}\left[\left(\frac{1}{N-1}\left(\bar{x} - x_i\right)\right)\right]^2} \\ &= \sqrt{\frac{1}{N(N-1)}\sum_{i=1}^{N}\left(x_i - \bar{x}\right)^2} \\ &= \Delta_N\,,\end{aligned} \tag{4.6.9}$$

which is the unbiased and consistent estimator of standard error for the mean. Similarly, one can show if one defines the jackknife estimate of bias as

$$\text{bias}^{\text{jack}}(\theta) \;\stackrel{\text{def}}{=}\; (N-1)\left(\bar{\theta}_{(\cdot)} - \hat{\theta}\right)\,, \tag{4.6.10}$$

then the jackknife estimator of mean is unbiased, i.e. $\text{bias}^{\text{jack}}(\bar{x}) = 0$. The jackknife can be viewed as an approximation to the bootstrap, see e.g. Ref. [6]. Its main advantage is that there are N jackknife samples or replications for a sample of size N, which makes it usually much less computer resource intensive than the bootstrap, for which $R = 4N$ can be used as a rule of thumb. The jackknife can be also, under certain circumstances, numerically more stable than the bootstrap, because the jackknife deviations $\hat{\theta}_{(i)} - \bar{\theta}$ are on average much smaller than the bootstrap equivalents. This can be helpful for smallish N. On the other hand, the jackknife might wrongly estimate variances, in particular for highly non-linear estimators or even those with a discontinuity.

For the purpose of estimating the error of the bootstrap error $\Delta^\star$ Eq. 4.5.6 we could apply the jackknife as follows: for each $i = 1, \ldots, N$, leave out data point i and recompute the error $\Delta^\star_{(i)}$. Then determine the jackknife error as outlined above. This requires for each i a new, full bootstrap simulation and it is, therefore, no advantage compared to the double bootstrap. However, we can make the following observation [7]

Theorem 4.6.1: jackknife-after-bootstrap

A bootstrap sample drawn with replacement from $x_{(i)}$ has the same distribution as the bootstrap sample drawn from $x^{\star}_{(i)} = x_1, x_2, \ldots, x_N$ with none of the x_j, $j = 1, ..., N$ equal to x_i.

Proof:
Each of the N components of either $x_{(i)}$ or $x^{\star}_{(i)}$ independently equal x_j, $j \neq i$ with probability $\mathbb{P} = 1/(N-1)$.

□

This observation allows one to estimate $\Delta^{\star}_{(i)}$ from all bootstrap samples not including x_i using

$$\Delta^{\star}_{(i)} = \sqrt{\sum_{b \in C_i} \frac{\left[\mathrm{sd}\left(x^{\star b}\right) - \overline{\mathrm{sd}}_i\right]^2}{B_i - 1}} \tag{4.6.11}$$

with

$$\overline{\mathrm{sd}}_i = \sum_{b \in C_i} \frac{\mathrm{sd}\left(x^{\star b}\right)}{B_i} .$$

Here, we let C_i denote the index set of the bootstrap samples not containing data point i and $B_i = |C_i|$ the number of indices in C_i. The *jackknife-after-bootstrap* estimate of standard error on $\Delta^{\star}$ is then given by

$$\Delta^{\mathrm{jack}}(\Delta^{\star}) = \sqrt{\frac{N-1}{N} \sum_{i=1}^{N} \left(\Delta^{\star}_{(i)} - \bar{\Delta}^{\star}_{(\cdot)}\right)^2} . \tag{4.6.12}$$

jackknife-after-bootstrap might provide a dramatic reduction in required computing resources to obtain an estimate of the error on the error compared to double bootstrap.

Moreover, Eq. 4.6.7 can be used to estimate the covariance matrix for each bootstrap sample $x^{\star b}$ using the jackknife proceedure. Compared to the double bootstrap, this can again provide a less resource intensive way to replace the frozen covariance matrix approach discussed in section 4.5.2 for the correlated χ^2–fit.

4.7. Autocorrelated Data

4.7.1. Autocorrelation Times and Error Correction

So far we assumed that the x_i are independent realisations of a random variable X. This leads to the definition of the standard error in Eq. 4.2.9. Recall that in the deviation we have explicitly made use of the independence. Now we will consider so called time series or a stochastic process:

Definition 4.7.1: Stochastic Process

A stochastic process is a collection of random variables indexed by a totally ordered index set T (time)

$$\{X_t : t \in T\}\,. \tag{4.7.1}$$

Hence, X_t does no longer need to be independent of $X_{t-1}, X_{t-2}, ...$ But we will assume in what follows that the actual value of t is not important. Later we will name this a *stationary* stochastic process. Thus, the correlation in $\{X_t\}$ depends only on the time difference. While the mean μ can still be estimated without bias from our standard estimator Eq. 4.2.6, the standard error cannot, as we will see now. Let us look at the variance of $T_{\bar{x}}$ again for such a stochastic process. For this we first define the autocorrelation function:

Definition 4.7.2: (Normalised) Autocorrelation Function

The autocorrelation (also autocovariance) function $C_X(t)$ of a stochastic process X_t is defined as

$$C_X(t) \overset{def}{=} \langle (X_i - \mu)(X_{i+t} - \mu) \rangle \tag{4.7.2}$$

and its normalised version Γ_X

$$\Gamma_X(t) \overset{def}{=} \frac{C_X(t)}{C_X(0)}\,. \tag{4.7.3}$$

As it is apparent from this definition, C_X does not depend on i, but only on (time) difference t. Typically, the autocorrelation function decays exponentially like

$$C_X(t) \propto \sum_{n=0}^{\infty} A_n \, \mathrm{e}^{-|t|/\tau_n}\,, \quad \tau_0 > \tau_1 > ... \in \mathbb{R}^+\,. \tag{4.7.4}$$

This does not necessarily need to be the case, but in the following we will always assume this to be the case. Then, for large enough $|t|$ only the largest correlation time τ_0 survives. Hence, one defines the so-called *exponential autocorrelation time* as follows:

$$\tau_{\exp} \overset{\text{def}}{=} \limsup_{t\to\infty} \frac{t}{-\ln|\Gamma_X(t)|}\,. \tag{4.7.5}$$

In terms of C_X we can compute the variance

$$\begin{aligned} \mathrm{Var}\,(T_{\bar{x}}) &= \frac{1}{N}\,\mathrm{Var}\,(X_1 + X_2 + ... + X_N) \\ &= \frac{1}{N^2}\sum_{r,s=1}^{N} C_X\,(r-s)\,. \end{aligned} \tag{4.7.6}$$

At this point we had previously argued that terms $\langle X_i, X_{i+t}\rangle$ are identically zero for $t \neq 0$. This is no longer the case now. Therefore, we have to proceed by including all

terms in the sum and counting multiplicities for identical $t = r - s$: t can take integer values in the range from $[-N+1, N-1]$. $t = -N+1$ appears once, like $t = N-1$. $-N+2$ appears twice, much like $N-2$. It is easy to see that this generalises to: t appears with a multiplicity of $N - |t|$. Thus, we can write

$$\begin{aligned} \mathrm{Var}\left(T_{\bar{x}}\right) &= \frac{1}{N^2} \sum_{t=-(N-1)}^{N-1} (N - |t|)\, C_X(t) \\ &= \frac{1}{N} \sum_{t=-(N-1)}^{N-1} \left(1 - \frac{|t|}{N}\right) C_X(t) \\ &= \frac{1}{N} C_X(0) \sum_{t=-(N-1)}^{N-1} \left(1 - \frac{|t|}{N}\right) \Gamma_X(t)\,. \end{aligned}$$

If we assume now that N is much larger than the autocorrelation time τ_0 in X_t and that $\Gamma(t) \propto \exp(-|t|/\tau)$, we can make the following approximation

$$\mathrm{Var}\left(T_{\bar{x}}\right) \overset{N \gg \tau}{\approx} \frac{2}{N} C_X(0)\ \tau_{\mathrm{int},X}\,, \tag{4.7.7}$$

with the following definition:

Definition 4.7.3: Integrated Autocorrelation Time

$$\tau_{\mathrm{int},X} \overset{def}{=} \frac{1}{2} \sum_{t=-\infty}^{\infty} \Gamma_x(t)\,. \tag{4.7.8}$$

The factor $1/2$ is conventionally inserted to obtain $\tau_{\mathrm{int}} \approx \tau_{\mathrm{exp}}$ for $\Gamma(t) \propto \exp(-|t|/\tau)$ with $\tau \gg 1$. The error we make in this approximation is of the order τ/N. At this point one notes that $C_X(0) = \mathrm{Var}\,(X)$, and, therefore, in case of autocorrelation the variance of $\bar{x}$ is roughly a factor $2\tau_{\mathrm{int},X}$ larger than it would be if the X_t were independent. Stated differently, in case of autocorrelation the effective number of measurements is $N/(2\tau_{\mathrm{int},X})$.

An estimator for the autocorrelation function $C(t)$ is given as follows

$$\bar{C}_X(t) = \frac{1}{N - |t|} \sum_{i=1}^{N-|t|} (x_i - \bar{x}_N)\left(x_{i+|t|} - \bar{x}_N\right) \tag{4.7.9}$$

which has a bias of the order $1/N$ and for $\Gamma(t)$

$$\bar{\Gamma}(t) = \frac{\bar{C}(t)}{\bar{C}(0)}\,. \tag{4.7.10}$$

The variance of this estimator (see e.g. Refs. [14, 1]) can be computed to

$$\mathrm{Var}\left(\bar{C}_X(t)\right) = \frac{1}{N} \sum_{i=-\infty}^{\infty} \left(C^2(i) + C(i+t)C(i-t) + \kappa(t,i,i+t)\right) + o\left(\frac{1}{N}\right) \tag{4.7.11}$$

with κ the connected only 4–point autocorrelation function. The integrated autocorrelation time should then be estimated using the following (biased) estimator

$$\overline{\tau}_{\text{int}} \;=\; \frac{1}{2}\sum_{t=-W}^{W}\frac{\overline{C}(t)}{\overline{C}(0)}\,, \qquad \tau \ll W \ll N \tag{4.7.12}$$

because the variance of this estimator grows like

$$\text{Var}\left(\overline{\tau}_{\text{int}}\right) \;\approx\; \frac{2\left(2W+1\right)}{N}\overline{\tau}_{\text{int}}^{2}\,.$$

This windowing procedure introduces a bias

$$\text{bias}\left(\overline{\tau}_{\text{int}}\right) \;=\; -\frac{1}{2}\sum_{|t|>W}\Gamma(t) + o\left(\frac{1}{N}\right)\,.$$

However, we have to live with this bias because otherwise the relative variance of the estimator becomes order 1. A practical way to chose the cutoff W is to chose it as the first t–value where $\Gamma(t)$ becomes zero within errors. The variance of $\bar{\Gamma}$ can be estimated using Eq. 4.7.11 and one finds

$$\text{Var}\left(\bar{\Gamma}(t)\right) \;\approx\; \frac{1}{N}\sum_{i=1}^{t+\Lambda}\left[\bar{\Gamma}(i+t)+\bar{\Gamma}(i-t)-2\bar{\Gamma}(i)\bar{\Gamma}(t)\right]^{2}\,, \tag{4.7.13}$$

where it was used that the dominant contribution to the variance comes from the disconnected contributions [11]. So, only those were kept and again a suitable windowing procedure was adopted [10]. We refer to appendix C of Ref. [11] and to Ref. [17] for more details.

4.7.2. Data Blocking

An alternative way to deal with autocorrelation in the data is to block the time series in consecutive blocks of length ℓ. Hence, we define the i-th block as

$$x_i^B \;=\; \frac{1}{\ell}\sum_{k=1}^{\ell} x_{(i-1)\cdot\ell+k} \tag{4.7.14}$$

creating a new random sample $x_1^B, ..., x_{N^B}^B$ of size $N^B = \lfloor N/\ell \rfloor$. In case ℓ does not divide N without rest we currently simply drop the reminder. Now we apply the statistical tools to the new, blocked random sample x^B. This is called *non-overlapping blocking*.

Lets look at the covariance of two consecutive blocks

$$\begin{aligned}\text{Cov}\left(X_1^B, X_2^B\right) \;&=\left\langle\left[\frac{1}{\ell}\sum_{i=1}^{\ell}(X_i-\mu)\right]\cdot\left[\frac{1}{\ell}\sum_{i=1}^{\ell}(X_{i+\ell}-\mu)\right]\right\rangle \\ &=\frac{1}{\ell^2}C_X(0)\sum_{r=\ell+1}^{2\ell}\sum_{s=1}^{\ell}\Gamma(r-s)\,.\end{aligned}$$

Now we count multiplicities again: $r-s=1$ appears once, $r-s=2$ twice, ..., $r-s=\ell$ appears ℓ–times, $r-s=\ell+1$ appears $\ell-1$–times, ..., $r-s=2\ell-1$ again once. Thus, assuming $\Gamma(t)=\exp\left(-1/\tau_0\right)^t \stackrel{\text{def}}{=} \xi^t$, this can be re-expressed as

$$\begin{aligned}\mathrm{Cov}\left(X_1^B, X_2^B\right) &= \frac{1}{\ell^2} C_X(0) \left[\sum_{t=1}^{\ell} t\,\xi^t + \sum_{t=\ell+1}^{2\ell-1} (2\ell-t)\xi^t\right], \\ &= \frac{1}{\ell^2} C_X(0) \left[\ell\xi^\ell + \sum_{t=1}^{\ell-1} t\,(\xi^t + \xi^{2\ell-t})\right].\end{aligned}$$

Since $0<\xi<1$ and $\xi^t > \xi^{2\ell-t}$ for all $t \in 1, ..., \ell-1$, this is bounded by

$$\mathrm{Cov}\left(X_1^B, X_2^B\right) < \frac{1}{\ell^2} C_X(0)\, 2 \sum_{t=1}^{\ell} t\,\xi^t .$$

The right hand side can be summed up to

$$\mathrm{Cov}\left(X_1^B, X_2^B\right) < \frac{1}{\ell^2} C_X(0)\; 2\,\frac{\ell\xi^{\ell+2} - (\ell+1)\xi^{\ell+1} + \xi}{(\xi-1)^2} .$$

Hence, $\mathrm{Cov}\left(X_1^B, X_2^B\right)$ vanishes in the limit of $\ell \to \infty$. This shows already that data blocking indeed leads to an uncorrelated blocked random sample for large enough ℓ. Moreover, for large enough ℓ the only important term is

$$\mathrm{Cov}\left(X_1^B, X_2^B\right) \approx \frac{1}{\ell^2}\ \mathrm{Var}(X)\ \frac{2\,\xi}{(\xi-1)^2} \tag{4.7.15}$$

and we recall that $0<\xi<1$. This gives a theoretical criterion

$$\frac{1}{\ell^2}\ \frac{2\,\xi}{(\xi-1)^2} \ll 1$$

for how to chose ℓ. However, in practice it is not possible to chose ℓ arbitrarily large, nor might τ_0 be known very precisely. A more practice oriented way to determine ℓ is to estimate the standard error for a row of ℓ–values, say $\ell = 1, 2, 4, 8, ...$ as the corresponding blocked data. Next one plots the so computed standard error versus ℓ. If the time series is autocorrelated, the error will increase with increasing ℓ. As soon as ℓ is large enough the standard error will reach a plateau, which indicates the value of ℓ to chose. **show plot**.

This procedure is complicated by the fact that the standard error has a statistical error itself. Therefore, the plateau will show up only within error bars. It is difficult in practice to estimate the error of the error reliably. Still, one could assign a relative error of size $\sqrt{\ell/N}$ as a guideline.

The bootstrap procedure can be applied to the blocked data directly, so the combination of the bootstrap and blocking is very natural. The blocking method we described

above then leads to what is called *simple block bootstrap* or *non-overlapping block bootstrap*.

We remark that there are alternative, more sophisticated ways to block the data, for which one also does not need to drop a part of the data. For instance, one can use moving overlapping blocks leading to what is called *moving block bootstrap*, or even extend it to the *circular block bootstrap* where the timeseries is put on a torus with periodic boundary conditions. Alternatively, one can sample the block length from a geometric distribution with mean ℓ. This leads to the *stationary block bootstrap* method.

4.8. Data Analysis with R

Let's generate some sample data using R's capability of generating pseudo random numbers

```
N <- 100
data <- data.frame(V1=rnorm(n=N, mean=0, sd=1),
                   V2=rnorm(n=N, mean=2, sd=1),
                   V3=rnorm(n=N, mean=4, sd=1))
```

Each collumn of the the `data.frame` is now a random sample of size $N = 10000$ from a normal distribution with standard deviation equal to one and different means. R provides function for mean, variance and standard deviation, which can be applied to the collumns separaterly

```
mean(data$V1)

## [1] 0.04101957

var(data$V3)

## [1] 0.6833162

sd(data$V2)

## [1] 1.072879
```

or directly to all of them as follows

```
xbar <- apply(X=data, MARGIN=2, FUN=mean)
xbar

##          V1          V2          V3
## 0.04101957 1.93314195 3.98352653
```

```
apply(X=data, MARGIN=2, FUN=sd)

##        V1        V2        V3
## 1.0480091 1.0728790 0.8266294
```

for which the `data.frame` is treated as a 10000×3 matrix and the function is applied to the index specified by `MARGIN`. Let's define a function for the standard error Δ_N

```
se <- function(x) {
  return(sd(x)/sqrt(length(x)))
}
```

which gives applied to our data the following

```
errors <- apply(X=data, MARGIN=2, FUN=se)
errors

##         V1         V2         V3
## 0.10480091 0.10728790 0.08266294
```

stored in the new object `errors`. Before we fit a linear model to our data, we first pack the data into a new `data.frame` we call `fdata`

```
fdata <- data.frame(t=c(1,2,3), x=xbar, err=errors)
```

`R` has the ability to perform weighted linear regressions. We refer to the documentation of `lm`, which can be obtained (like for any `R` function) using `?lm` for more details. However, we will do it manually applying the bootstrap procedure.

4.8.1. Bootstrapping

Let us first do the bootstrap by hand for a simple example of a sample of real valued observations of size N. We use the `sample` function of `R` to generate the bootstrap samples as follows

```
boot.R <- 1000
x <- rnorm(n=N, mean=0, sd=0.1)
xstar <- array(sample(x=x, size=N*boot.R, replace=TRUE),
               dim=c(boot.R, N))
```

Next we compute the bootstrap replications on every sample and obtain the estimate of standard error

```
xbarstar <- apply(xstar, 1, mean)
sd(xbarstar)

## [1] 0.01044123
```

Note that we could also have used `sample.int` to generate index arrays instead of `sample`. For comparison, the usual estimate of standard error gives

```
sd(x)/sqrt(N)

## [1] 0.01023689
```

which is very similar to the bootstrap estimate. The error on the error we estimate using the jackknife-after-bootstrap method. For this purpose we need to find all bootstrap samples not containing the ith data point. This we can for instance do by applying `duplicated`. We use it to write a function finding the duplicates in a given bootstrap sample

```
find.duplicates <- function(xstar, x) {
  duplicated(c(xstar, x))[(length(x) + 1):(2 * length(x))]
}
duplicates <- t(apply(xstar, 1, find.duplicates, x))
```

which we directly applied to all bootstrap samples using the `apply` function. Note that we transposed the result for later puposes. The array `duplicates` contains booleans for whether or not the corresponding value of the orignal data appeared in the bootstrap sample. Hence, due to transposing every row corresponds to one data point in x and every column to one bootstrap sample.

For computing the jackknife replications, we define yet another function taking one row of the `duplicates` index array as argument. It applies a function `f` (in our case the standard deviation) to all columns of `xstar`. But only to those rows which for which the index array is `FALSE`, i.e. those which do not contain the data point encoded by the column.

```
jack.boot <- function(indices, xstar, f) {
  f(xstar[!indices])
}
jack.boot.values <- apply(duplicates, 2, jack.boot, xstar,
                          f=sd)
```

Here, the notation `xstar[!indices]` returns a single vector, which are concatenations of `f` applied to `xstar` as explained above. The boolean vector is used for the first index of the array `xstar`. Hence, `jack.boot.values` is a vector containing $\Delta^\star_{(i)}$ from Eq. 4.6.11 for $i = 1, \ldots, N$. Now, the jackknife-after-boot estimate of standard error of the standard error reads

```
jack.boot.se <- sqrt(((N - 1)/N) *
                     sum((jack.boot.values - mean(jack.boot.values))^2))
jack.boot.se

## [1] 0.007006397
```

following Eq. 4.6.12. It is, as not unexpected for $N = 100$, quite sizable.

In the following we use for convenience the package `boot` to perform the bootstrap on our data. The relevant function is called `boot`, and it takes the data, the statistics function and the number of bootstrap samples as arguments (and more, see the help page for details). The statistics functions requires at least two arguments, the first one being the original data and the second one an index array for which to apply the statistics. We start with the mean and, therefore, define a function which meets these criteria as follows

```
library(boot)
mean.boot <- function(x, ii) {
  if(length(dim(x)) > 1) {
    return(apply(x[ii,], 2, mean))
  }
  return (mean(x[ii]))
}
boot.res <- boot(data=data, statistic=mean.boot, R=boot.R)
boot.res

##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = data, statistic = mean.boot, R = boot.R)
##
##
## Bootstrap Statistics :
##       original      bias    std. error
## t1* 0.04101957 0.001603431  0.10628785
## t2* 1.93314195 0.001942262  0.10351188
## t3* 3.98352653 0.001593587  0.08206857
```

As we defined the function, it is directly able to deal also with two-dimensional structures, like our data frame. In the function we assume the first index to be the one for the observations, and the second to be the one for the different observables.

You can see from the output that the important information is comoputed directly for the three observables (called `t1, t2, t3`). Compared to the errors we computed

previously 0.1048009, 0.1072879, 0.0826629, we find good agreement. Also the bias is small, as it should be.

The `plot` function is overloaded for objects of type `boot`. You can access the corresponding help page via `?plot.boot`. The corresponding plot gives us further hints on the quality of the bootstrap procedure. In particular, the QQ-plot helps us to understand whether or not the standard-deviation is a good estimator for the error.

```
plot(boot.res, index=1)
```

The corresponding plot is displayed in Figure 4.8.1. The `index` argument to `plot.boot` allows one to chose which of the possibly several observables to look at.

Let's now bootstrap also the fit. We will use a χ^2–fit, even if we look at a linear model. For this purpose we need to provide a χ^2–function to the `R` function `optim`.

```
par <- c(1, 1)
boot.err <- apply(boot.res$t, 2, sd)
chisqr <- function(par, x, y, dy){
  return(sum( (par[1] + par[2]*x - y)^2/dy^2))
}
fit.res <- optim(par=par, fn=chisqr, x=fdata$t, y=boot.res$t0, dy=boot.err)
fit.res

## $par
## [1] -1.965646  1.977227
##
## $value
## [1] 0.4113397
##
## $counts
## function gradient
##       85       NA
##
## $convergence
## [1] 0
##
## $message
## NULL

pvalue <- 1-pchisq(fit.res$value, 1)
pvalue

## [1] 0.5212903
```

The p–value of the fit looks quite reasonable, which is not surprising. Now we introduce another, more general data container of `R`, namely a `list`. And we will store the whole

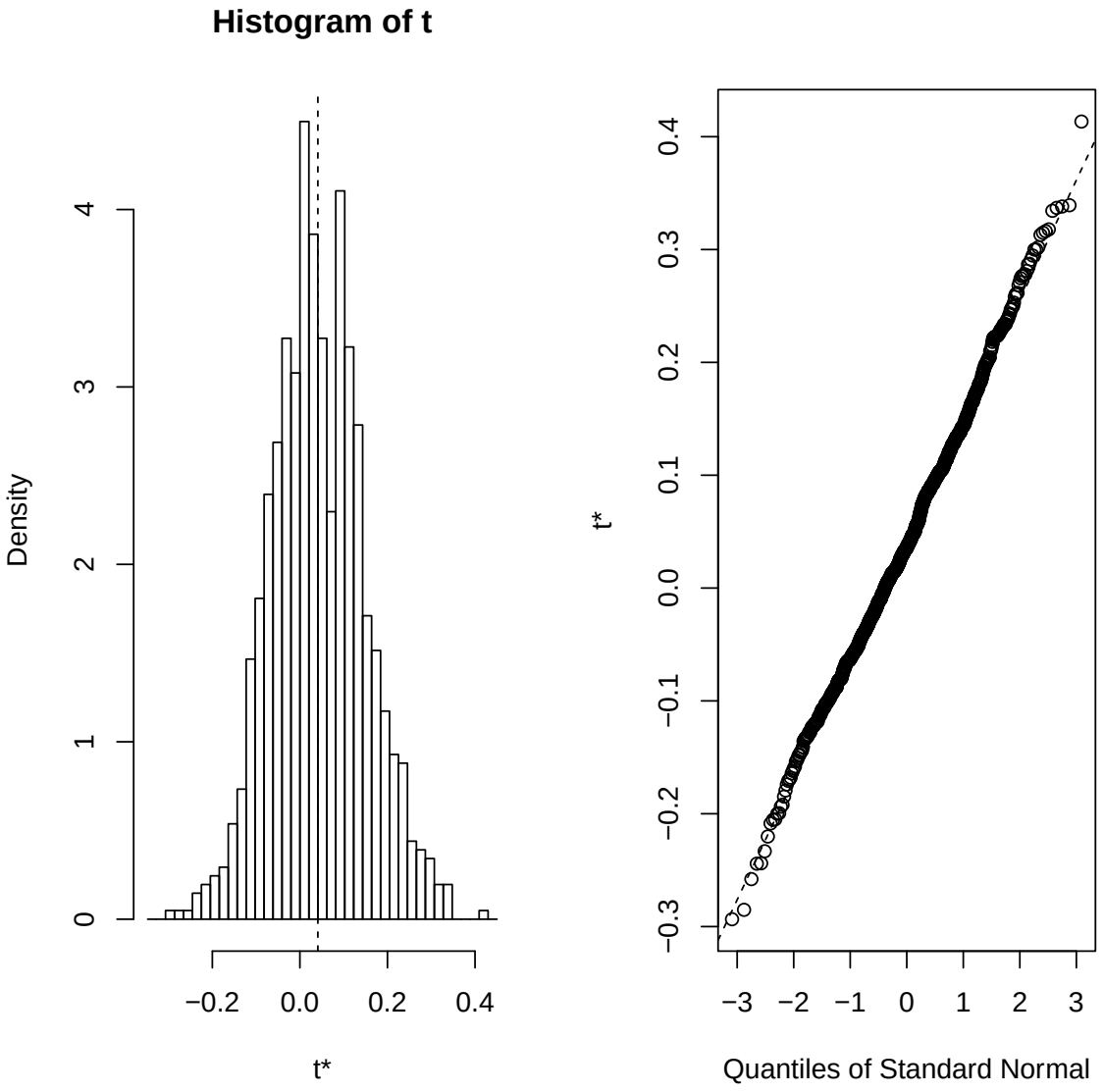


Figure 4.8.1.: Plot of the bootstrap result.

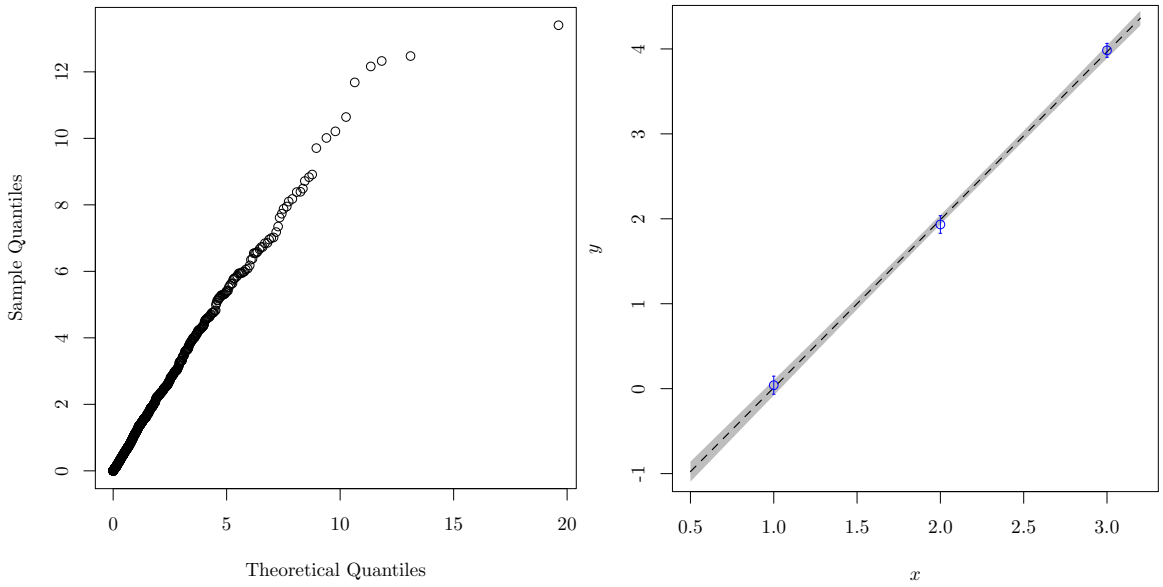


Figure 4.8.2.:
QQ-plot and plot of the data for the χ^2–fit for a linear model to sample data.

fit result in such a container. Moreover, we will now repeat the fit for every bootstrap sample.

```
fit.boot <- list(R=boot.R)
fit.boot$t <- array(NA, dim=c(boot.R, length(par)))
fit.boot$t0 <- fit.res$par
fit.boot$value0 <- fit.res$value
fit.boot$value <- array(NA, dim=c(boot.R))
fit.boot$data <- list(x=fdata$t, y=boot.res$t0, boot.res=boot.res, dy=boot.err)
fit.boot$dof <- 1
fit.boot$pvalue <- 1-pchisq(fit.boot$value0, fit.boot$dof)
for(r in c(1:boot.R)) {
 tmp <- optim(par=par, fn=chisqr, x=fdata$t, y=boot.res$t[r,], dy=boot.err)
  fit.boot$t[r,] <- tmp$par
  fit.boot$value[r] <- tmp$value
}
class(fit.boot) <- c("myfit", "list")
```

From that we compute the errors and biases of the two fit parameters. We have given the object a new class in order to demonstrate the possibility to overload standard `R` functions for new classes. Here, we overload `summary` and then we also make a QQ-plot of the bootstrap samples of the χ^2–values versus the theoretical non-central χ^2 distribution.

```
summary.myfit <- function(fit.boot) {
  cat("parameters:", fit.boot$t0, "\n")
  cat("errors    :", apply(fit.boot$t, 2, sd), "\n")
  cat("biases    :", fit.boot$t0 - apply(fit.boot$t, 2, mean), "\n")
}
summary(fit.boot)

## parameters: -1.965646 1.977227
## errors    : 0.1469021 0.06405679
## biases    : -0.001891319 7.970157e-05

qqplot(y=fit.boot$value,
       x=rchisq(n=fit.boot$R*10, df =1 , ncp=fit.boot$value0),
       main="", xlab="Theoretical Quantiles", ylab="Sample Quantiles")
```

The corresponding QQ-plot can be found in the left panel of Figure 4.8.2, which shows good agreement with our expectation of a non-central χ^2–distribution. Note the different scales on x– and y–axis.

Finally, we would also like to plot our fit result. For an error band we can use the bootstrap samples for the fit parameters and compute for every x–value the corresponding y–value and its error. The plot is then very similar to the function `plotwitherror` discussed in appendix A.1.

```
x <- seq(.5, 3.2, 0.01)
y <- fit.boot$t0[1] + fit.boot$t0[2]*x
Y <- array(NA, dim=c(boot.R, length(x)))
for(r in c(1:boot.R)) Y[r,] <- fit.boot$t[r,1] + fit.boot$t[r,2]*x
dy <- apply(Y, 2, sd)

plot(NA, xlim=c(0.5,3.2), ylim=c(-1,4.3), main="", xlab="x", ylab="y")
polygon(x=c(x, rev(x)), y=c(y+dy, rev(y-dy)),
        col="gray", lty=0, lwd=0.001, border="gray")
lines(x=x, y=y, lty=2, lwd=1.5)
points(x=fit.boot$data$x, y=fit.boot$data$y, col="blue", pch=21)
arrows(x0=fit.boot$data$x, y0=fit.boot$data$y-fit.boot$data$dy,
       x1=fit.boot$data$x, y1=fit.boot$data$y+fit.boot$data$dy,
       length=0.01, angle=90, code=3, col="blue")
```

The corresponding plot can be found in the right panel of Figure 4.8.2.

4.8.2. Time Series Analysis

Lets obtain some sample data. The NASA provides the global temperature averages for all year from 1881 till 2016 on their webpage (`http://data.giss.nasa.gov/`

gistemp/) [9, 16]. They provide an CSV file which can be accessed as follows

```
Data <- read.csv(
    "http://data.giss.nasa.gov/gistemp/tabledata_v3/GLB.Ts+dSST.csv",
    header=TRUE, na.strings="***", skip=1)
```

for which one needs to be able to access the internet. The file contains one column per month and in the last columns averages over three successive month. Note that the meteorological year counts from December till November, thus, the average of the last four columns gives the mean over the meteorological year

```
Data$means <- rowMeans(Data[,c("DJF","MAM","JJA","SON")],
    na.rm=TRUE)
YoYChanges <- diff(Data$means, 1)
sd(YoYChanges)

## [1] 0.1138618
```

Note that the file contains not the absolute temperatures, but normalised to the base period 1951-1980. Units are degrees Celsius. In `YoYChanges` we save the year-over-year changes in the temperature, using the `diff` function.

Now, we can ask the question whether or not there is any correlation in the year-over-year changes. The easiest visual impression is obtained by computing the autocorrelation function. For this purpose `R` provides the function `acf`

```
N <- length(YoYChanges)
W.max <- 10
Lambda <- 5
Gamma <- acf(YoYChanges, main="", lag.max=W.max, xlab="t")
```

The `acf` function als plots the autocorrelation function directly. The correponding plot can be found in Figure 4.8.3. The blue dashed lines represent the 95% confidence interval for the data being consistent with zero. In this data we have a case where there is an anti-correlation at lag equal to one, which seems to be significant. To confirm this, we can implement Eq. 4.7.13 to estimate the error of the autocorrelation function as follows

```
Gamma$acf[(W.max+2):(2*W.max+W.max+1)] <- 0 ## pad with zeros
dGamma <- numeric()
for(t in c(0:W.max)) {
  k <- c(max(1,(t-Lambda)):(t+Lambda))
  dGamma[t+1] <- sum((Gamma$acf[(k+t+1)]+Gamma$acf[(abs(k-t)+1)]-
                      2*Gamma$acf[t+1]*Gamma$acf[(k+1)])^2);
  dGamma[t+1] <- sqrt(dGamma[t+1]/N)
```

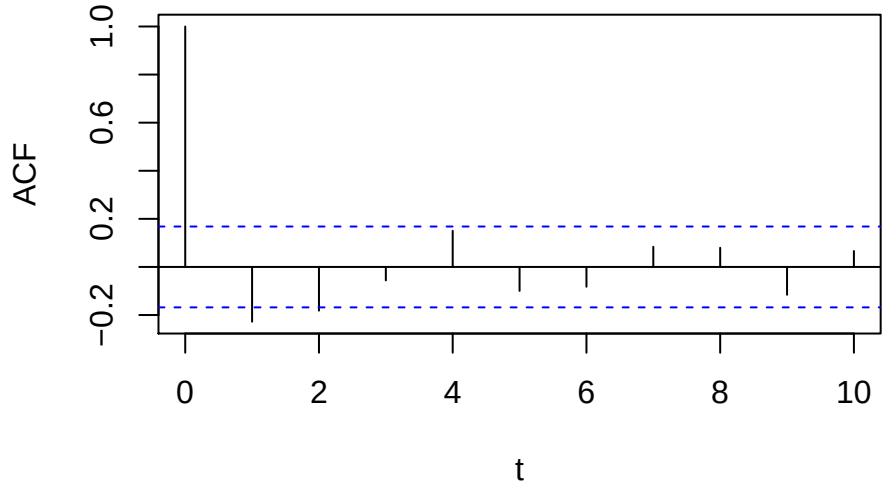


Figure 4.8.3.:
Autocorrelation function of mean year-over-year change in the global temperature data.

```
}
ii <- c(2:(W.max+1))
plotwitherror(ii-1, Gamma$acf[ii], dGamma[ii],
              xlim=c(0,10), ylim=c(-0.4,1.1),
              ylab="ACF", xlab="t", col="blue")
points(0, 1, col="blue")
abline(h=0, lty=2)
```

So, in fact, $\Gamma(t)$ is only significantly different from zero for lag $t = 1$. Note that we have plotted the value at $t = 0$ separately, because the error of this point is zero, of course.

References

[1] T Anderson. *The Statistical Analysis of Time Series*. Wiley, 2011. DOI: 10.1002/9781118186428.

[2] BA Berg. "Monte Carlo calculation of confidence limits for realistic least square fitting". In: *Computer Physics Communications* 69.1 (1992), pp. 65–72. ISSN: 0010-4655. DOI: http://dx.doi.org/10.1016/0010-4655(92)90129-M. URL: http://www.sciencedirect.com/science/article/pii/001046559290129M.

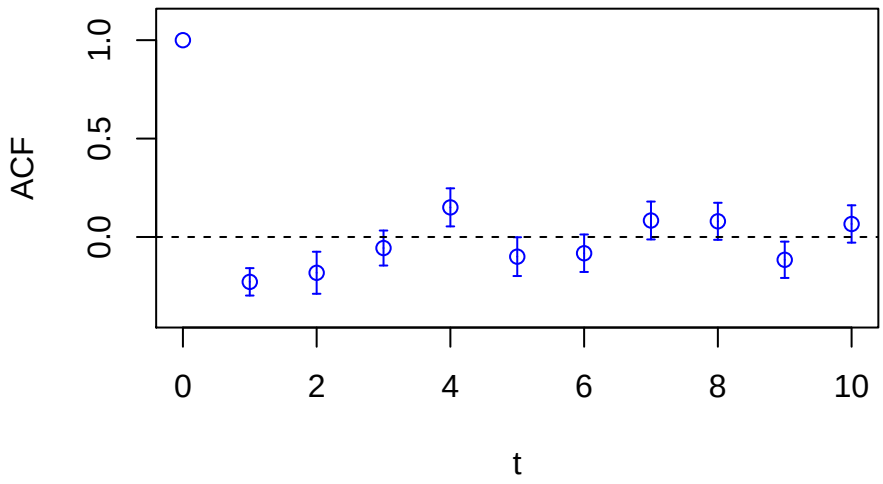


Figure 4.8.4.:
Autocorrelation function of mean year-over-year change in the global temperature data with errors.

[3] A Canty and B Ripley. *boot: Bootstrap R (S-Plus) Functions*. R package version 1.3-17. 2015.

[4] A Davison and D Hinkley. *Bootstrap Methods and their Application*. Cambridge Series in Statistical and Probabilistic Mathematics. Camebridge University Press, 1997.

[5] B Efron. "Bootstrap Methods: Another Look at the Jackknife". In: *Ann. Statist.* 7.1 (Jan. 1979), pp. 1–26. DOI: 10.1214/aos/1176344552. URL: http://dx.doi.org/10.1214/aos/1176344552.

[6] B Efron and R Tibshirani. *An introduction to the bootstrap*. Chapman and Hall/CRC, 1994.

[7] B Efron. "Jackknife-After-Bootstrap Standard Errors and Influence Functions". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 54.1 (1992), pp. 83–127. ISSN: 00359246. URL: http://www.jstor.org/stable/2345949.

[8] D Freedman and S Peters. "Bootstrapping a Regression Equation: Some Empirical Results". In: *J. Am. Stat. Assoc.* 79.385 (1984), pp. 97–106.

[9] GISTEMP-Team. *GISS Surface Temperature Analysis (GISTEMP)*. [Online; accessed 03-November-2016]. 2016. URL: http://data.giss.nasa.gov/gistemp/.

[10] M Lüscher. "Schwarz-preconditioned HMC algorithm for two-flavour lattice QCD". In: *Comput. Phys. Commun.* 165 (2005), pp. 199–220. DOI: 10.1016/j.cpc.2004.10.004. arXiv: hep-lat/0409106 [hep-lat].

[11] N Madras and A Sokal. "The pivot algorithm: A highly efficient Monte Carlo method for the self-avoiding walk". In: *Journal of Stat. Phys.* 50 (1988), pp. 109–186.

[12] C Michael and A McKerrell. "Fitting correlated hadron mass spectrum data". In: *Phys. Rev.* D51 (1995), pp. 3745–3750. DOI: 10.1103/PhysRevD.51.3745. arXiv: hep-lat/9412087 [hep-lat].

[13] S original, from StatLib, and by Rob Tibshirani. R port by Friedrich Leisch. *bootstrap: Functions for the Book "An Introduction to the Bootstrap"*. R package version 2015.2. 2015. URL: https://CRAN.R-project.org/package=bootstrap.

[14] M Priestley. *Spectral analysis and time series*. Vol. I and II. Academic Press, 1981. DOI: 10.1002/for.3980010411.

[15] MH Quenouille. "Problems in Plane Sampling". In: *Ann. Math. Statist.* 20.3 (Sept. 1949), pp. 355–375. DOI: 10.1214/aoms/1177729989. URL: http://dx.doi.org/10.1214/aoms/1177729989.

[16] J Hansen, R Ruedy, M Sato, and K Lo. "Global surface temperature change". In: *Rev. Geophys.* 48 (2010). DOI: doi:10.1029/2010RG000345.

[17] A Sokal. "Monte Carlo Methods in Statistical Mechanics: Foundations and New Algorithms". Lecture notes at Cargese Summer School. 1996. URL: www.stat.unc.edu/faculty/cji/Sokal.pdf.

[18] J Turkey. "Bias and Confidence in Not-quite Large Samples (abstract)". In: *Ann. Math. Statist.* 29.2 (June 1958), pp. 614–623. DOI: 10.1214/aoms/1177706647. URL: http://dx.doi.org/10.1214/aoms/1177706647.

[19] U Wolff. "Monte Carlo errors with less errors". In: *Comput. Phys. Commun.* 156 (2004). [Erratum: Comput. Phys. Commun.176,383(2007)], pp. 143–153. DOI: 10.1016/S0010-4655(03)00467-3,10.1016/j.cpc.2006.12.001. arXiv: hep-lat/0306017 [hep-lat].

[20] C Wu. "Jackknife, Bootstrap and other resampling methods in regression analysis". In: *Annals of Statistics* 14.4 (1986), pp. 1261–1295.

5. Monte-Carlo Integration and Optimisation

We have seen in chapter 3 that the accept-reject method basically computes an integral. In this chapter we will formalise this a bit and we will discuss refined methods with reduced variance. Next, we will discuss so-called *quasi Monte-Carlo* methods, which have superior scaling properties in the number of samples. Finally, we will also discuss a different class of problems, namely optimisation problems.

5.1. Classical Monte-Carlo Integration

Consider the generic problem of solving

$$\theta \;=\; \mathbb{E}_f\left[h(X)\right] \;\stackrel{\text{def}}{=}\; \int_{\chi} h(x)f(x)\mathrm{d}x \tag{5.1.1}$$

with $f(x) > 0$ for all x in the support of f. It appears rather natural from the developments we did in the previous lectures to generate a random sample $(x_1, \ldots, x_N)$ from pdf $f(x)$ and estimate

$$\overline{\theta}_N \;=\; \frac{1}{N}\sum_{i=1}^{N} h\left(x_i\right) . \tag{5.1.2}$$

This converges almost–surely to θ with N going to infinity using the strong law of large numbers. The estimator for the variance of $\bar{\theta}_N$, as discussed before, is given by

$$\Delta_N^2 \;=\; \frac{1}{N(N-1)}\sum_{i=1}^{N}\left(h\left(x_i\right) - \overline{\theta}_N\right)^2 . \tag{5.1.3}$$

From the central limit theorem we then know that

$$\frac{\overline{\theta}_N - \theta}{\Delta_N} \tag{5.1.4}$$

is standard normally distributed, i.e. from $\mathcal{N}_{0,1}$. Therefore we know that the statistical error of the estimator scales as

$$\Delta_N \;\propto\; \frac{1}{\sqrt{N}} \tag{5.1.5}$$

for N large enough.

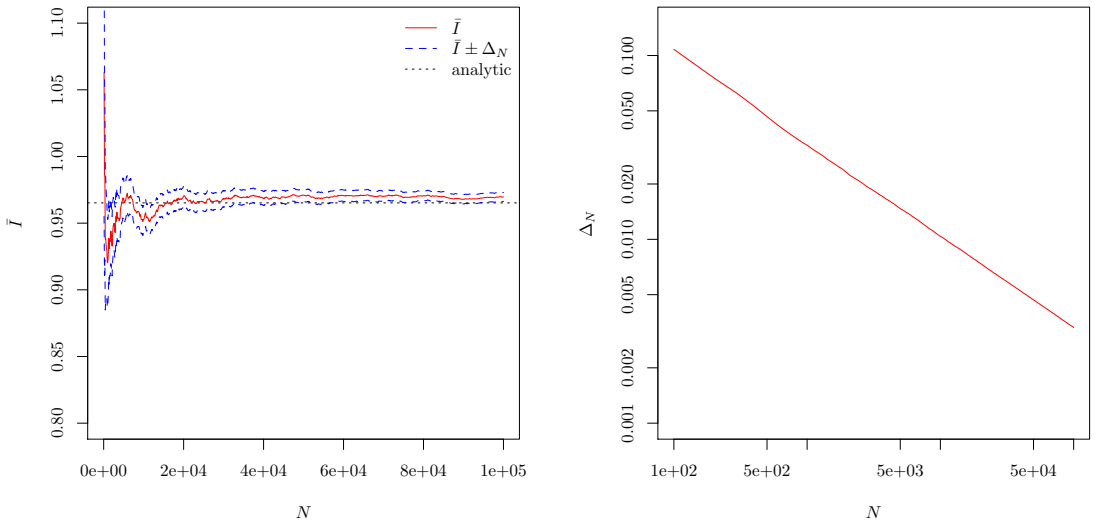


Figure 5.1.1.:
Results for example 5.1.1: in the left panel we who $\bar{I}$ and in the right panel Δ_N as functions of N.

Example 5.1.1

Consider

$$I \;=\; \int_0^1 \left(\cos\left(50x\right) + \sin\left(20x\right)\right)^2 \,\mathrm{d}x\,.$$

The analytic solution is given by

$$I \;=\; 1 + \frac{\cos(30)-1}{30} - \frac{\cos(70)-1}{70} - \frac{40}{80} + \frac{100}{200} \;\approx\; 0.9651733\,.$$

We interpret this as for $f(x) = 1$ a uniform distribution in $[0,1]$ and

$$h(x) \;=\; \left(\cos\left(50x\right) + \sin\left(20x\right)\right)^2\,.$$

This means, we compute

$$I \;=\; \mathbb{E}_{\mathcal{U}_{[0,1]}}\left[h(x)\right]\,.$$

Hence, we sample $u_1, \ldots, u_N \sim U_{[0,1]}$ and estimate

$$\bar{I} \;=\; \frac{1}{N}\sum_n h(u_n)\,.$$

The convergence of $\bar{I}$ with increasing N is shown in the left panel of Figure 5.1.1. In the right panel the standard error is shown again as a function of N. One nicely observes the $1/\sqrt{N}$ scaling.

This was a very straightforward example. In the next one it gets a bit more complicated:

Example 5.1.2

Consider the integral

$$\theta \;=\; \int_{-2}^{2} \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{y^2}{2}}\, \mathrm{d}y\,.$$

We have the choice what to call $h(x)$ and what $f(x)$ in Eq. 5.1.1. Consider the following two possibilities

1. choose $h(x) = 1$ and $f(x) = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{x^2}{2}}$

   Hence, we generate $x_1, \ldots, x_N \sim \mathcal{N}_{0,1}$ and estimate

   $$\overline{\theta} \;=\; \frac{1}{N} \sum_j \mathbb{1}_{-2 \le x_j \le 2}$$

   using the indicator function.

2. choose $h(x) = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{x^2}{2}}$ and $f(x) = 1$

   Now we have to draw $U_1, \ldots, U_N \sim U_{[-2,2]}$

5.2. Accept-Reject

We re-formulate the problem now slightly and adopt what we learned in chapter 3. Consider the integral

$$I \;=\; \int_0^1 h(x)\, f(x)\, \mathrm{d}x\,,$$

with $0 \le f(x) \le 1$ and for simplicity we have constrained the integral to the interval $[0, 1]$. We rewrite this integral as follows

$$I \;=\; \int_0^1 \int_0^1 h(x)\, \mathbb{1}_{y \le f(x)}\, \mathrm{d}y\, \mathrm{d}x\,. \tag{5.2.1}$$

This leads to the following algorithmic steps:

1. sample (x, y) uniformly on $[0, 1] \times [0, 1]$
2. set the weight $w = \mathbb{1}_{y \le f(x)}$

Algorithm 5.2.1 Accept-Reject for Integration

$M = 0$
for $i = 1, 1, 2, 3, ..., N$ **do**
    independently sample $x \sim \mathcal{U}_{[0,1]}, y \sim \mathcal{U}_{[0,1]}$
    compute

$$w_i = \begin{cases} 1 & y \leq f(x) \\ 0 & y > f(x) \end{cases}.$$

    if $w_i = 1$ **then**
        $M = M + 1$
    end if
end for
return

$$\frac{1}{M} \sum_{i=1}^{N} w_i \, h(x_i)$$

3. repeat from the first step N–times until number of accepted points is M

This algorithm is a slight reformulation of algorithm 3.2.1, which we know from chapter 3. The Monte-Carlo estimator for the integral is then given by

$$\bar{I} \;=\; \frac{1}{M} \sum_{i=1}^{N} w_i \, h(x_i) \,, \tag{5.2.2}$$

with

$$M \;=\; \sum_{i=1}^{N} w_i$$

the sum of weights, or in this case the total number of accepted points. Inspecting the algorithm again, one realises that one can drop the condition $0 \leq f(x) \leq 1$, if one also generalises the uniform sampling for (x, y). The algorithm is summarised in algorithm 5.2.1.

5.3. Importance Sampling

It became clear from what we discussed above that we have a large freedom in what we call instrumental distribution. Different choices will lead to different estimators for the same integral. The different estimators will differ with respect to:

1. variance
2. conceptional strength
3. computational effort

and we can optimise our method by choosing an appropriate estimator.

In this section we will formalise the idea of optimising the Monte-Carlo representation of an integral in a systematic way. The discussion is based on the *fundamental importance sampling identity*:

$$\mathcal{I} \;=\; \mathbb{E}_f\left[h(X)\right] \;=\; \int h(x)\frac{f(x)}{g(x)}g(x)\,\mathrm{d}x \;=\; \mathbb{E}_g\left[h(X)\frac{f(X)}{g(X)}\right] \tag{5.3.1}$$

for a distribution $g(x) > 0$ suitably chosen. g is called *instrumental distribution* and we require for the supports of f and g that $\operatorname{supp} f \subset \operatorname{supp} g$.

Definition 5.3.1: Importance Sampling

Evaluate Eq. 5.3.1 by sampling $X \sim g$ to obtain a random sample $x_1, ..., x_N$ and approximate the integral $\mathcal{I}$ by

$$\overline{\mathcal{I}} \;=\; \frac{1}{N}\sum_{j=1}^{N} h\left(x_j\right)\frac{f\left(x_j\right)}{g\left(x_j\right)}\,. \tag{5.3.2}$$

We can calculate the (exact) variance of the integral by taking

$$\mathrm{Var}_g\left(h(X)\,\frac{f(X)}{g(X)}\right) = \mathbb{E}_g\left[h(X)^2\,\frac{f(X)^2}{g(X)^2}\right] - \left(\mathbb{E}_g\left[h(X)\,\frac{f(X)}{g(X)}\right]\right)^2\,. \tag{5.3.3}$$

The second term on the right-hand side is just the integral value $\mathcal{I}$ and independent of g. Thus, the variance of the integral is finite, only if

$$\mathbb{E}_g\left[h(X)^2\,\frac{f(X)^2}{g(X)^2}\right] = \mathbb{E}_f\left[h(X)^2\,\frac{f(X)}{g(X)}\right] = \int h(x)^2\,\frac{f(x)^2}{g(x)}\,\mathrm{d}x < \infty\,. \tag{5.3.4}$$

From Eq. 5.3.4 we infer that g should preferably be chosen

- such that the ratio $f(x)/g(x)$ is bounded, i.e. $f(x)/g(x) < M$ for all x, since an unbounded ratio leads to strongly fluctuating weights and, therefore, also fluctuating estimators;
- for x–values, where $f(x)$ becomes small, $g(x)$ should be larger than $f(x)$ to ensure that the ratio $f(x)/g(x)$ does not become too large. One also says that g should have thicker tails than f.

These conditions give some guidance on how to choose the instrumental density. Mathematical theory provides a solution to the question of how to optimally chose the instrumental distribution g by the following theorem:

Theorem 5.3.1: Rubinstein, 1981

Algorithm 5.3.1 Importance Sampling

1: **for** $i = 1, ..., N$ **do**
2: sample $x_i \sim g$
3: compute $y_i = h(x_i)/g(x_i)$
4: **end for**
5: **return**

$$\frac{1}{N}\sum_{i=1}^{N} y_i$$

The choice of g, which minimises the g-variance Var_g of

$$\mathcal{I} = \mathbb{E}_g\left[h(X)\,\frac{f(X)}{g(X)}\right] = \int h(x)\,\frac{f(x)}{g(x)}\,g(x)\,\mathrm{d}x$$

is given by

$$g^*(x) = \frac{|h(x)|\ f(x)}{\int |h(x)|\ f(x)\,\mathrm{d}x}\,. \tag{5.3.5}$$

Proof:
To prove this relation we go back to Eq. 5.3.3 and the fact, that minimising the g-variance means minimising the first term on the right-hand side. The application of Jensen's inequality then gives

$$\mathbb{E}_g\left[h(X)^2\,\frac{f(X)^2}{g(X)^2}\right] \geq \left(\mathbb{E}_g\left[|h(X)|\ \frac{f(X)}{g(X)}\right]\right)^2 = \left(\int |h(x)|\ f(x)\,\mathrm{d}x\right)^2\,. \tag{5.3.6}$$

The lower bound is realised by the g^* given in Eq. 5.3.5.

□

But this theorem is only a formal statement, since to construct g^* we would have to know the integral $\int |h|\ f$ from the beginning. Still, even if we do not have enough information to construct g, the theorem tells us that we should choose g such that $|h|\,f/g \approx$ const and with finite variance. The performance of importance sampling is low on the other hand, if

$$\int \frac{f(x)^2}{g(x)}\,\mathrm{d}x = \infty\,.$$

The algorithm for importance sampling with instrumental density $g(x)$ can be found in algorithm 5.3.1.

Example 5.3.1

We aim to compute the integral

$$P = \int_5^\infty \frac{1}{\sqrt{2\pi}} \exp\left(-x^2/2\right) , \tag{5.3.7}$$

which corresponds to the probability of a standard normally distributed random variable x to be larger than five. As instrumental density we use the truncated exponential distribution

$$\begin{aligned} g(x) &= \exp(5-x) \\ G(x) &= 1 - \exp(5-x) \qquad (5.3.8) \\ G^-(x) &= -\log(1-u) + 5 \qquad (5.3.9) \end{aligned}$$

We added the cumulative distribution function and the generalised inverse. We use the generalised inverse method to sample $X \sim g$ as usual $U \sim \mathcal{U}(0,1)$ with $X = G^-(U) \sim g$.

An alternative estimator for $\mathcal{I}$ (5.3.1) is given by:

$$\overline{\mathcal{I}} \;=\; \frac{\sum h(x_j)\, w(x_j)}{\sum w(x_j)}, \quad w(x) \stackrel{\text{def}}{=} f(x)/g(x), \quad x_j \sim g\,. \tag{5.3.10}$$

Here, compared to the estimator Eq. 5.3.2, N was replaced by the sum of weights $\sum_j f(x_j)/g(x_j) = \sum_j w(x_j)$. This is possible since

$$\frac{1}{N}\sum_j \frac{f(X_j)}{g(X_j)} \;=\; \frac{1}{N}\sum_j w(X_j) \;\stackrel{\text{a.s.}}{=}\; 1\,,$$

since f is a distribution and $X_j \sim g$. This alternative estimator has a small bias. But its performance might be more stable in practice. And, it allows for a more practical solution to the optimality question: choose g such that we can estimate $E_f[h(x)]$ from

$$\frac{\sum h(x_j)\,|h(x_j)|^{-1}}{\sum |h(x_j)|^{-1}}$$

However, the optimality does not translate directly and the estimator might have severe instabilities. The ratio

$$W(x) = \frac{f(x)}{g(x)}$$

is called the **likelihood ratio** or simply **weight** and the sampling with

$$\frac{\mathbb{E}_g[h(x)\cdot W(x)]}{\mathbb{E}_g[W(x)]}$$

is called **weighted sampling**. Compare also to the accept/reject algorithm 5.2.1.

Example 5.3.2

An example for weighted sampling can be given by considering self-avoiding random walks (SAWs). An SAW is a random walk (RW) that is non-reversal and does not intersect itself. SAWs are for instance used to describe polymer chains. A very simple algorithm to generate an SAW of N steps in d–dimensions on a regular lattice with lattice spacing a is the following:

1: start with initial coordinate $\vec{r}_0 = 0$ and $\vec{\mu}_0 = 0$
2: **for** $k = 1, ..., N$ **do**
3: choose a random direction: $\vec{\mu}_{k+1} = \pm a\vec{e}_i \neq -\vec{\mu}_k$ with $i \sim \mathcal{U}_{1,...,d}$
4: $\vec{r}_{k+1} = \vec{r}_k + \vec{\mu}_{k+1}$, $k = k + 1$
5: **if** $\vec{r}_k$ visited already **then**
6: terminate and start all over
7: **end if**
8: **end for**

The problem with this algorithm is that it will perform rather poorly in generating SAWs for large N. The probability of generating a SAW decreases exponentially with N.

The termination step is needed to guarantee uniform sampling over all RWs. A solution to this "problem" is to choose $\vec{r}_{k+1}$ only among the lattice sites which are still free. Thus, termination appears only if there is no "free" site left. This means we have modified the sampling, because certain chains are prefered over others. It is no longer uniform.

It can be cured by giving the single SAWs an according weight. The weight is given as follows:

$$W = \prod_{i=1}^{N} \frac{l_i}{2d-1},$$

where l_i is the number of "free" sites at step $i = 1, ..., N$.

Note that for this example we are not solving an integral, but a sum, because the state space of all SAWs with N steps is finite.

5.3.1. Acceleration methods

Before closing the section on Monte-Carlo integration, we want to look at two examples of how to accelerate the convergence of the simulation.

Antithetic variables

In simulations we usually considered the production of samples of independent and identically distributed (**iid**) random variables. But there are cases, where it can be advantageous to have correlated random variables. Remember that the primary goal

is to construct estimators with minimal variance and highest efficiency. If introducing correlation between random numbers can serve this purpose, we use it.

An example, which makes the potential use of correlation plausible is the comparison of two quantities. Suppose we have

$$\mathcal{I}_j = \int g_j(x)\, f_j(x)\, \mathrm{d}x\,, \quad j = 1,\, 2 \tag{5.3.11}$$

and we are interested in the difference $\delta\,\mathcal{I} = \mathcal{I}_1 - \mathcal{I}_2$. We know that, if $\delta\,\mathcal{I}$ is a small difference of two large quantities and both have independent noise, then it is difficult to get a significant precision from an estimate. Say we have two estimators θ_1 and θ_2, one for each of the two integrals, then if they are independent

$$\mathrm{var}\,(\theta_1 - \theta_2) = \mathrm{var}\,(\theta_1) + \mathrm{var}\,(\theta_2)\ . \tag{5.3.12}$$

If $\theta_{1,2}$ are correlated, the variance of the difference is modified by the covariance term:

$$\mathrm{var}\,(\theta_1 - \theta_2) = \mathrm{var}\,(\theta_1) + \mathrm{var}\,(\theta_2) - 2\,\mathrm{Cov}\,(\theta_1,\, \theta_2)\ . \tag{5.3.13}$$

Thus the variance of the estimator for the difference can be reduced, if the estimators $\theta_{1,2}$ can be chosen and the simulation carried out, such that they have a strong positive correlation.

The concept of *antithetic variables* exploits this idea. Suppose we have two samples of random variables, $(X_1, \ldots, X_N)$ and $(Y_1, \ldots, Y_N)$. With both we can estimate the target integral

$$\mathcal{I} = \int h(x)\, f(x)\, \mathrm{d}x\,.$$

The crucial point is,that the estimator

$$\hat{\theta}_{XY} = \frac{1}{2N} \sum_{j=1}^{N} \left(h(X_j) + h(Y_j)\right) \tag{5.3.14}$$

is more efficient than the estimator

$$\hat{\theta}_{2X} = \frac{1}{2N} \sum_{j=1}^{2N} h(X_j) \tag{5.3.15}$$

based on an **iid** sample of size $2N$, if $h(X_j)$ and $h(Y_j)$ entering the definition Eq. 5.3.14 are negatively correlated. Using this concept leaves the task of suitably introducing correlation between the random variables and in general this is not a straightforward task.

Rubinstein (1981) makes a practical proposal, which is applicable when sampling with the generalised inverse method with $X = F^{-1}\,(U)$ and $U \sim \mathcal{U}\,(0,\, 1)$. One uses both U to generate $X = F^{-1}\,(U)$ and $1 - U$ to generate $Y = F^{-1}\,(1 - U)$. Then $h(X_j)$ and $h(Y_j)$ are negatively correlated, if $H = h \circ F^{-1}$ is a monotone function.

Control variates

Suppose we want to estimate

$$\mathcal{I} = \int h(x)\, f(x)\, \mathrm{d}x \tag{5.3.16}$$

by an estimator θ_1. If we have some exact knowledge of expectation values of other quantities, e.g. $\mathcal{I}_0 = \mathbb{E}_f\left[h_0(X)\right]$, then this information can be used to reduce the variance of the estimator for the target integral Eq. 5.3.16. To that end we use an *unbiased estimator* θ_3 for $\mathcal{I}_0$ and define the new estimator

$$\theta_2 = \theta_1 + \beta\, \left(\theta_3 - \mathbb{E}_f\left[h_0(X)\right]\right) \tag{5.3.17}$$

$$\mathbb{E}_f\left[\theta_1\right] = \mathbb{E}_f\left[\theta_2\right] \tag{5.3.18}$$

which has the variance

$$\mathrm{Var}_f\left(\theta_2\right) = \mathrm{Var}_f\left(\theta_1\right) + \beta^2\, \mathrm{Var}_f\left(\theta_3\right) + 2\,\beta\, \mathrm{Cov}_f\left(\theta_1,\, \theta_3\right)\,. \tag{5.3.19}$$

For the optimal choice of β

$$\beta^* = -\frac{\mathrm{Cov}_f\left(\theta_1,\, \theta_3\right)}{\mathrm{Var}\left(\theta_3\right)} \tag{5.3.20}$$

the variance becomes

$$\mathrm{Var}_f\left(\theta_2\right) = \mathrm{Var}_f\left(\theta_1\right)\, \left(1 - \rho_{13}^2\right) \tag{5.3.21}$$

$$\rho_{13} = \frac{\mathrm{Cov}_f\left(\theta_1,\, \theta_3\right)}{\sqrt{\mathrm{Var}_f\left(\theta_1\right)\, \mathrm{Var}_f\left(\theta_3\right)}}$$

Example 5.3.3

Assume we want to calculate the following integral

$$\mathbb{P}\left(X > a\right) \;=\; \int_a^\infty f(x)\, \mathrm{d}x$$

and suppose we know $\mathbb{P}\left(X > \mu\right)$ for some $\mu < a$. With the indicator function and $X_i \sim f$ we then have

$$\theta_2 = \frac{1}{N}\sum_{j=1}^{N} 1_{(a,\infty)}(X_j) + \beta\, \left(\frac{1}{N}\sum_{j=1}^{N} 1_{(\mu,\infty)}(X_j) - \mathbb{P}\left(X > \mu\right)\right)\,.$$

We use the relation

$$1_{(a,\,\infty)}\, 1_{(\mu,\,\infty)} = 1_{(a,\,\infty)}$$

for $\mu < a$ and get the variance and covariance

$$\begin{aligned}\mathrm{Var}\left(\theta_3\right) &= \mathbb{P}\left(X > \mu\right)\left(1 - \mathbb{P}\left(X > \mu\right)\right)/N \\ \mathrm{Cov}\left(\theta_1, \theta_3\right) &= \mathbb{P}\left(X > a\right)\left(1 - \mathbb{P}\left(X > \mu\right)\right)/N\,.\end{aligned}$$

We conclude, that $\beta < 0$ and

$$|\beta| < 2\,\frac{\mathrm{Cov}\left(\theta_1, \theta_3\right)}{\mathrm{Var}\left(\theta_3\right)} = 2\,\frac{\mathbb{P}\left(X > a\right)}{\mathbb{P}\left(X > \mu\right)}\,.$$

5.4. Monte-Carlo Integration with R

Let us start by looking at example 5.1.1. We first implement the function to be evaluated

```
f <- function(x) {
  return ( (cos(50*x) + sin(20*x))^2 )
}
```

Next we implement the naïve Monte-Carlo estimator discussed in example 5.1.1. For this we first generate uniform random numbers from $\mathcal{U}_{[0,1]}$ and evaluate $f(x)$ on those. Next, we compute the estimator $\bar{I}$ and its standard error.

```
analytic <- 1+(cos(30)-1)/30 - (cos(70)-1)/70 - sin(40)/80 + sin(100)/200
N <- 100000
u <- runif(N)        ### uniform random sample
y <- f(u)            ### evaluate f on u
I <- mean(y)         ### compute the mean
sdI <- sd(y)
seI <- sdI/sqrt(N)   ### and the standard error
```

In order to obtain the estimator as a function of the sample size we repeat this calculation for various sample sizes using a `for`–loop as follows

```
I.running <- numeric(0)
I.running.se <- numeric(0)
step <- 100
nseq <- seq(step, N, step)
for(n in nseq) {
  I.running[n/step] <- mean(y[1:n])
  I.running.se[n/step] <- sd(y[1:n])/sqrt(n)
}
### next we plot
```

```
plot(NA, xlim=c(0,N), ylim=c(0.8,1.1), xlab=c("N"), ylab=c("I"))
abline(h=analytic, col="black", lty=c(3))
lines(nseq, I.running, col="red")
lines(nseq, I.running-I.running.se, col="blue", lty=c(2))
lines(nseq, I.running+I.running.se, col="blue", lty=c(2))
```

Finally, we also plot the standard error as a function of the sample size.

```
plot(NA, xlim=c(step,N), ylim=c(0.001,.15), xlab=c("N"),
     ylab=c("se(I)"), log=c("xy"))
lines(nseq, I.running.se, col="red")
```

The corresponding plots can be found in Figure 5.1.1.

We now turn towards example 5.3.1 for importance sampling. The neccessary functions are defined as follows

```
h <- function(x) {                    ### target function
  return ( exp(-x^2/2)/sqrt(2*pi) )
}
g <- function(x) {                    ### instrumental density
  return( exp(5-x))
}
Ginv <- function(x) {                 ### its generlised inverse
  return(5 - log(x))
}
```

Now we implement algorithm 5.3.1 as follows

```
u <- runif(n=N)                       ### sample uniformly
x <- Ginv(x=u)                        ### apply generalised inverse
y <- h(x)/g(x)                        ### generate monte carlo samples
```

The result is $\bar{P} = 2.8868439 \times 10^{-7}$ with a standard error of $\Delta_N = 1.2616816 \times 10^{-9}$. For comparison, the expected value is $P = 2.8665157 \times 10^{-7}$. We can compare this to the estimate using classical Monte-Carlo integration with $X \sim \mathcal{N}_{0,1}$.

```
x <- rnorm(n=N)
ii <- which(x > 5)
P <- length(ii)/N
```

The result is $\bar{P} = 0$, which is a very bad estimate because there are simply too few $x > 5$. From the expected result we can estimate that we need something like $100/P = 3.4885558 \times 10^8$ for having at least 100 samples with value larger than five. This is more than three orders of magnitudes larger than the N–value we used. The classical estimator works better, if we only ask for the probability of $x > 2$:

```
ii <- which(x > 2)
P <- length(ii)/N
w <- rep(0, times=N)
w[ii] <- 1
seP <- sd(w)/sqrt(N)
```

Now the result is $\bar{P} = 0.0228$ with error $\Delta_N = 4.72021 \times 10^{-4}$, which is compatible within error with the actual value $P = 0.0227501$.

5.5. Quasi Monte-Carlo Integration

We have seen that Monte-Carlo integration provides one with methods to approximately compute integrals with a statistical error decreasing like $N^{-1/2}$. This scaling is guaranteed by the central limit theorem as long as the variance of the estimator is finite. The methods could be improved by inventing estimators with smaller variance, which, however, does not improve on the general $N^{-1/2}$ scaling law.

The idea of quasi Monte-Carlo is to use quasi-random numbers (qRN) instead of pseudo random numbers (pRN). qRN are designed to provide better uniformity than pRN and, thus, faster convergence. For this we first have to define what we mean by uniformity, or provide means to measure it. This will be done using discrepancies and, hence, quasi-random sequences are also called low discrepancy sequences.

Quasi-random sequences do not try to mimic random sequences. To the contrary, they have correlations which prevents clumping of points. This improves the scaling to $\mathcal{O}\left((\log N)^k N^{-1}\right)$.

Let us first define discrepancies:

Definition 5.5.1: Discrepancies

For a sequence of points $x_1, ..., x_N$ in the unit cube $x_n \in I^d$ in d dimensions we define for any subset $J \subset I^d$

$$R_N(J) \overset{def}{=} \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}_{x_n \in J} - \mu(J)\,,$$

with $\mu(J)$ the Lebesgue*–measure of J. $R_N(J)$ is just the Monte-Carlo quadrature error in measuring the volume of J. Next we define a norm on R_N, such as to measure discrepancies. This is most conveniently done by restricting the sets J to rectangular ones (those are actually sufficient to uniquely define the* Lebesgue*-measure on I^d). We define*

$$D_N \overset{def}{=} \sup_{J \in E} |R_N(J)| \tag{5.5.1}$$

with E the set of all rectangular subsets of I^d. D_N defines a L^∞ norm on J. We also define the $D^\star$–discrepancy over the set $E^\star$ containing all rectangular sets in I^d with one vertex at 0

$$D_N^\star \overset{def}{=} \sup_{J \in E^\star} |R_N(J)|\,. \tag{5.5.2}$$

Note that a rectangular set $J(x, y)$ is uniquely defined by two antipodal vertices, x and y. $D_N^\star$ represents an L^∞ norm on $J(0, x) \equiv J(x)$. D_N and $D_N^\star$ define two different discrepancies. There are other definitions possible.

In principle any measure of discrepancy could be used. The $D_N^\star$ discrepancy represents one particular choice which coincides with the Kolmogorov-Smirnov distance between the empirical cumulative distribution function of the points $x_1, \ldots, x_N$ and the theoretical uniform distribution on I^d.

The idea is now to chose the points $x_1, \ldots, x_N$ in a way that the $D_N^\star$ discrepancy is minimised or at least small. Consider the integral over the unit cube

$$I[f] \;=\; \int_{I^d} f(x)\,\mathrm{d}x\,.$$

Its Monte-Carlo approximation is given by

$$I_N[f] \;=\; \frac{1}{N}\sum_{n=1}^{N} f(x_n)$$

with a random sequence $x_1, ..., x_N$. The error of this quadrature formula is naturally given by

$$\epsilon[f] \;\stackrel{\text{def}}{=}\; |I[f] - I_N[f]|\,. \tag{5.5.3}$$

Now, indeed, chosing the points $x_1, \ldots, x_N$ by minimising the $D_N^\star$ discrepancy leads to small errors in the quadrature formula Eq. 5.5.3 if the function f is smooth enough and if it has smooth partial derivatives. Smoothness is measured using $V(f)$, a total variation:

Definition 5.5.2: Variation in the Hardy-Krause sense

Let f be a function of a single variable. Then we define the variation in the Hardy-Krause *sense*

$$V[f] \;\stackrel{def}{=}\; \int_0^1 \left|\frac{\mathrm{d}f}{\mathrm{d}t}\right| \mathrm{d}t\,.$$

The d dimensional generalisation is given by

$$V[f] \;\stackrel{def}{=}\; \int_{I^d} \left|\frac{\partial^d f}{\partial t_1 \ldots \partial t_d}\right| \mathrm{d}t_1 \ldots \mathrm{d}t_d + \sum_{i=1}^{d} V[f_1^{(i)}]\,. \tag{5.5.4}$$

Here, $f_1^{(i)}$ is the restriction of f to the boundary $x_i = 1$. The restrictions are functions of $d-1$ variables. Thus, the definition above is recursive.

$V(f)$ represents intuitively the length of the monotone segments of f. The next theorem [5, 4] shows that a bound on the integration error can be established in terms of the discrepancy and the measure of the smoothness of the function.

Theorem 5.5.1: Koksma-Hlawka

Let f be a function with bounded variation $V[f]$. For any sequence $x_1, ..., x_N \in I^d$ the integration error $\epsilon[f]$ is bounded as

$$\epsilon[f] \;\leq\; V[f]\, D_N^\star \,. \tag{5.5.5}$$

Proof:
Let f be a function with bounded variation $V[f]$ and $x_1, ..., x_N$ a sequence in I^d. We write R_N with δ–functions:

$$R(x) \;=\; \int_{J(x)} \left[\frac{1}{N}\sum_n \delta(x' - x_n) - 1\right] \mathrm{d}x' \;=\; \frac{1}{N}\sum_n \mathbb{1}_{x_n \in J} - \int_{J(x)} \mathrm{d}x' \,.$$

Then, $R(x) = R_N(J(x)) \equiv R_N(J(0,x))$ with $J \in E$ and x the one vertex that is not equal to 0. Consider first a function f that vanishes on the boundaries of the unit cube. This allows one to perform an integration by part in the following derivation and drop the boundary related term:

$$\begin{aligned}
\epsilon[f] &= \left| \int_{I^d} f(x)\mathrm{d}x - \frac{1}{N}\sum_n f(x_n) \right| \\
&= \left| \int_{I^d} \left[1 - \frac{1}{N}\sum_n \delta(x - x_n)\right] f(x)\, \mathrm{d}x \right| \\
&\overset{\text{P.I.}}{=} \left| \int_{I^d} R(x)\, f'(x) \mathrm{d}x \right| \\
&\leq \left[\sup_x R(x)\right] \int_{I^d} |\mathrm{d}f(x)| \\
&= D_N^\star\, V[f] \,.
\end{aligned}$$

For f nonzero on the boundary of the unit cube, the terms from integration by parts are bounded by the boundary terms in $V[f]$.

□

In practice it turns out that it is not so much the variation $V[f]$ that is important, but mostly the discrepancy.

5.5.1. Low Discrepancy Sequences

Definition 5.5.3: quasi-random sequence

An infinite sequence $x_1, x_2, \ldots$ is uniformly distributed if

$$\lim_{N\to\infty} D_N = 0 \,. \tag{5.5.6}$$

It is quasi-random if

$$D_N \;\leq\; c(\log N)^k\, N^{-1} \tag{5.5.7}$$

with c and k constants independent of N. It is common to say a sequence is quasi-random only if $k = d$ with d the dimension of the integral to be solved.

In $d = 1$ dimension, the simplest such sequence is the Van der Corput sequence. x_n is obtained by writing n in dual base

$$n \;=\; a_m a_{m-1} \ldots a_0$$

with a_i the digits in base 2. Then x_n is given in dual base

$$x_n = 0.a_0 a_1 \ldots a_m \,.$$

So-called Halton sequences are the generalisation of Van der Corput sequences to arbitrary dimensions. The discrepancy of a Halton sequence is bounded by

$$D_N(\text{Halton}) \;\leq\; c_d(\log N)^d N^{-1}\,,$$

with c_d a constant depending on d. There are other low discrepancy sequences known, for instance Sobol sequences and others. Algorithms to generate those can be found in the literature.

For comparison we show in Figure 5.5.1 in the left panel uniformly distributed random points and in the left panel a two-dimensional Halton quasi-random sequence. The clumping in the pseudo-random points is clearly visible, while the Halton sequence appears to be more uniformly distributed in the unit square.

As an example we show in Figure 5.6.1 again for example 5.1.1 the convergence in the length of the sequence in the left panel, both for quasi Monte-Carlo with a Van der Corput sequence and for Monte-Carlo. In the right panel of this figure the error scaling is shown, determined from 100 independent sequences. The N^{-1} scaling of quasi Monte-Carlo can nicely be observed.

5.5.2. Limitations

Quasi Monte-Carlo seems to be a big improvement compared to ordinary Monte-Carlo: quasi Monte-Carlo converges much faster than ordinary Monte-Carlo. However, there are limitations, some of which we are going to mention here.

First of all, there is no theoretical basis for empirical estimates of accuracy of quasi Monte-Carlo methods. In other words, the central limit theorem does not hold for quasi Monte-Carlo and, hence, it cannot be used to access the error scaling. In practice the error will be accessed by understanding quasi Monte-Carlo as an iterative refinement method: the change in the estimate from N to N' will serve as an error estimate.

Second, for large dimensions d the $(\log N)^d$ dominates the error bound as long as $N < 2^d$. And indeed, it is found that quasi Monte-Carlo looses efficiency in large dimensions.

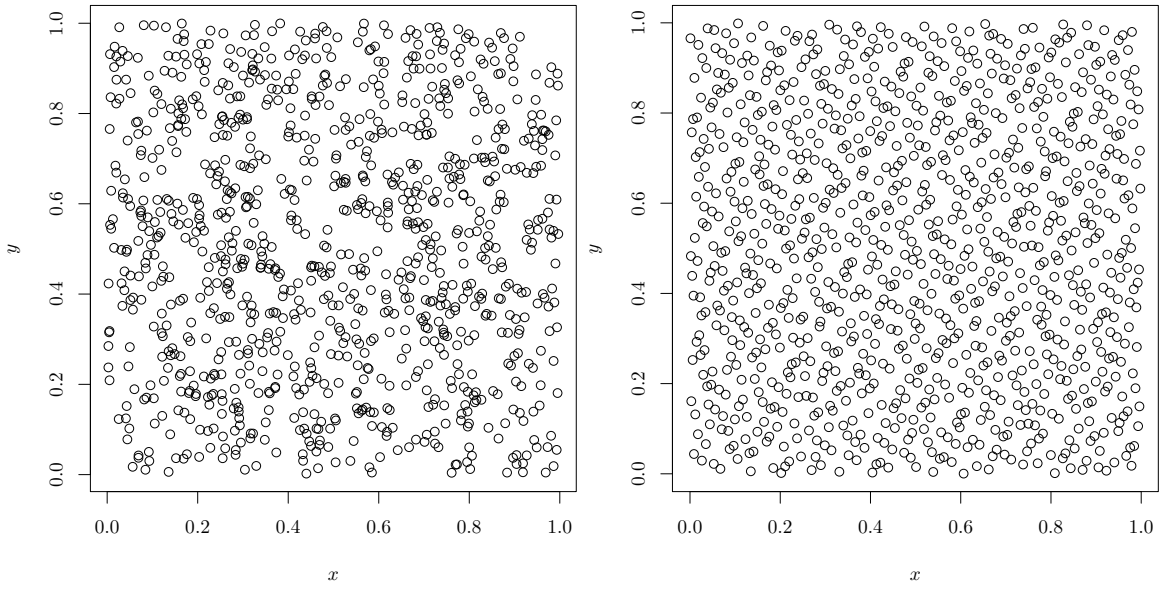


Figure 5.5.1.:
In the left panel we show uniformly distributed random points in I^2, whereas in the right panel we show HALTONS quasi-random sequence in the unit square. For both, pseudo-random and quasi-random sequences the number of points was $N = 1000$.

Third, the price for the improved scaling are additional conditions we assume for the integrand. In the case of quasi Monte-Carlo it is smoothness of the function. This comes from the variation being part of the KOKSMA-HLAWKA inequality which becomes important for non-smooth functions. This is indeed relevant if one thinks of accept-reject methods, which involve decisions and, thus, integration over discontinuous step functions..

5.5.3. Smoothed Accept-Reject

This can be nicely illustrated by the accept-reject method algorithm 5.2.1. It involves the discontinuous indicator function. In this situation quasi Monte-Carlo does not perform well. The solution [13, 7] is to replace the indicator function by a smooth function $q(x, y)$ satisfying

$$\int_0^1 q(x, y)\,\mathrm{d}y \;=\; f(x)\,.$$

q could for instance be chosen as

$$q(x,y) = \begin{cases} 1 & y \le f(x) - \epsilon \\ 0 & y > f(x) + \epsilon \\ \text{linear} & f(x) - \epsilon < y \le f(x) + \epsilon \end{cases} .$$

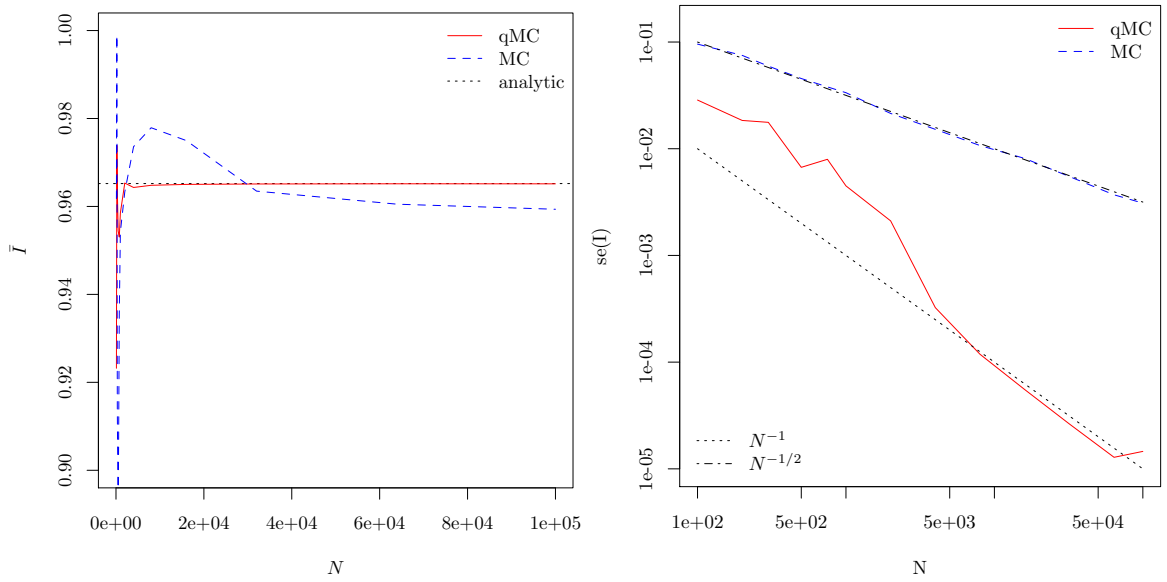


Figure 5.6.1.:
MC and qMC result for example 5.1.1. In the left panel we show the estimate for the integral as a function of the length of the sequence, in the right panel the error scaling.

Having defined q we can rewrite the integral as

$$\int_0^1 h(x)\, f(x)\, \mathrm{d}x \;=\; \int_0^1 \int_0^1 h(x)\, q(x,y)\, \mathrm{d}x\, \mathrm{d}y\,.$$

The algorithm would then proceed as follows

1. sample (x, y) uniformly **iid**
2. compute $w = q(x, y)$
3. repeat

and the estimator for the integral is given by

$$\bar{I} \;=\; \frac{1}{N} \sum_{i=1}^{N} q(x_i, y_i)\, h(x_i)\,.$$

5.6. Quasi Monte-Carlo with R

This time we have to work a bit more to get our hands on the error scaling: we cannot use the statistical tools anymore. Therefore, we generate 100 quai-random sequences

of total length 10^5 and estimate the standard deviation using Eq. 4.2.10 over the 100 sequences. We do the same also with pseudo-random sequences, for comparison.

We use HALTON sequences which we generate using the library `randtoolbox`, which you may need to install using `install.packages`. The library provides for instance HALTON and SOBOL sequences, see the corresponding help pages. We initialise the library as follows

```
library(randtoolbox)

## Loading required package: rngWELL
## This is randtoolbox. For overview, type 'help("randtoolbox")'.

x <- halton(1)
```

Here, we have also initialised the function `halton` once in order to generate the 100 following sequences without initialisation, but still different from each other. Next, we apply f from example 5.1.1 to the qRN and the pRN

```
N <- 100000
X <- array(0, dim=c(N, 100))
Y <- array(0, dim=c(N, 100))
for(i in c(1:100)) {
  x <- halton(n=N, dim=1, init=FALSE)
  X[,i] <- f(x)           ### evaluate f on qRN
  Y[,i] <- f(runif(n=N))  ### same for pRN
}
```

Next we compute the running mean and standard deviation for several N–values.

```
qI.running <- numeric(0)
qI.running.se <- numeric(0)
I.running <- numeric(0)
I.running.se <- numeric(0)
nseq <- c(100, 200, 300, 500, 750, 1000, 2000,
          4000, 8000, 16000, 32000, 64000, 100000)
for(n in c(1:length(nseq))) {
  qI.running[n] <- mean(X[1:nseq[n],1])
  qI.running.se[n] <- sd(apply(X[1:nseq[n],], 2, mean))
  I.running[n] <- mean(Y[1:nseq[n],1])
  I.running.se[n] <- sd(apply(Y[1:nseq[n],], 2, mean))
}
```

In the right panel of Figure 5.6.1 we show the expectation value of the integral as a function of the length of the sequence, both for qRN and qRN sequences. One observes already from this plot that the quasi-random sequence outperforms crude Monte-Carlo.

In the left panel of the same figure we show the integration error, estimated from the standard deviation over the 100 sequences, again as a function of N. Indeed, for quasi MC we observe N^{-1} scaling, whereas for MC we observe $N^{-1/2}$ scaling.

Another example is the integral

$$\mathcal{I} = \int_{I^3} f(x) \mathrm{d}^3 x \tag{5.6.1}$$

with

$$f(x) = \begin{cases} 1 & x_1 < x_2 < x_3 \\ 0 & \text{otherwise} \end{cases} .$$

The corresponding R–code looks as follows

```
N <- 100000
f <- function(x) {
  if(x[1] < x[2] && x[2] < x[3]) return(1)
  return(0)
}
## with 3d Halton sequence
x <- halton(n=N, dim=3, init=TRUE)
y <- apply(x, 1, f)
## with uniform random numbers
u <- array(runif(n=3*N), dim=c(N, 3))
yu <- apply(u, 1, f)

mean(y)

## [1] 0.1666

mean(yu)

## [1] 0.16574

sd(yu)/sqrt(N)

## [1] 0.00117589
```

and we could study the convergence in the same way as we did previously, which is left to the exercises. The analytic result, however, is $\mathcal{I} = 1/6$.

5.7. Monte-Carlo Optimisation

In physics, we are often concerned with the problem of finding the maximum of some function h over a given set of arguments:

$$x^\star \;=\; \max_{x\in\chi}\; h\left(x_1,\ldots,x_n\right)$$

with χ is the search space. We will talk here about the maximum implicitly including also the case for finding the minimum. In this case the function needs to be bounded from below. The problem of finding the minimum of h can be mapped to finding the maximum of $-h$. Typical examples from physics are:

1. variational methods in Quantum Mechanics.
   Let $|\varphi\rangle \in \mathcal{H}$ be a state in a HILBERT space and
   $$\langle A\rangle_\varphi = \frac{\langle\varphi|A|\varphi\rangle}{\langle\varphi|\varphi\rangle}$$
   the expectation value of some operator A. Then, every state $|\varphi\rangle$ for which $\langle A\rangle_\varphi$ becomes extremal is an Eigenstate of the operator A.
2. χ^2 minimisation.
3. gauge fixing in field theories which amounts to finding A^μ such that
   $$\partial_\mu A^\mu \;=\; 0\,.$$

In some cases there are analytic solutions available, like for instance for linear regressions. However, very often such solutions do not exist, and one way to obtain results in particular in high dimensional search spaces is using stochastic methods.

Moreover, a problem one hits often in optimisation problems is the one of local extrema. One may interprete the function to maximise as height in some high dimensional landscape. Therefore, it is easy to imagine that there are *local* maxima in addition to the one or more *global* maxima. Deterministic algorithm often have problems with local extrema and stochastic methods can help to overcome those.

5.7.1. Basic Solution

Let $h : \mathbb{R}^n \to \mathbb{R}$ be a function which is bounded from above on some compact support $\chi \subset \mathbb{R}^n$. Assume we are interested in finding

$$h^\star \;=\; \max_{x\in\chi} h(x)\,. \tag{5.7.1}$$

A first approach is to simulate, if possible, uniformly on $\chi \subset \mathbb{R}^n$, i.e. $x_1,\ldots,x_N \sim \mathcal{U}_\chi$. and use the estimator

$$\bar{h}^\star_N \;=\; \max\left(h\left(x_1\right),\ldots,h\left(x_N\right)\right) \tag{5.7.2}$$

This method will converge to the true maximum of h on χ for $N \to \infty$. On the other hand it is clear that depending on the properties of h this method might be very inefficient, because we do not use any of the properties of h.

In case that we relate the function h to a probability distribution with properties:

$$\int_\chi h(x)\,\mathrm{d}x \;<\; +\infty\,, \qquad h(x) \;\geq\; 0 \tag{5.7.3}$$

we can apply the tools developed so far: we may simulate $X \sim h$ and the solution of our problem is then given by the *modes* of the density h.

Definition 5.7.1: Mode

The mode is the value that appears most often in a set of data.

If h does not fulfil the conditions of a probability density function, we may still be able to define a new function $H(x)$ with the following properties

1. $H(x) \geq 0$ and $\int_\chi H(x)\,\mathrm{d}x < +\infty$
2. The maximum of $H(x)$ on χ agrees with the maximum of $h(x)$ on χ.

One possible choice for the function $H(x)$ could be for instance:

$$H(x) \;=\; \exp\left(\frac{h(x)}{T}\right) \tag{5.7.4}$$

which clearly fulfils the conditions. The free parameter T can be adjusted to accelerate the particular method under consideration.

5.7.2. Gradient Method

The gradient method is in principle a deterministic method, which iteratively finds a solution to the problem we are after to solve. The iteration can be written as

$$x_{i+1} \;=\; x_i + \alpha_i \cdot \nabla h\,(x_i) \tag{5.7.5}$$

which involves the gradient of the function $h(x)$, which is utilised to decrease the function value in each iteration. Examples for such gradient methods are the Newton– or Newton–Raphson methods.

For certain choices of the coefficient α_i the methods are guaranteed to find the correct solution. This guarantee depends, however, on the choice of the initial point x_0: x_0 must be close enough to the desired solution, which is not always easy to fulfil.

One possibility to overcome this shortcoming is to add some stochasticity to the method and modify equation (5.7.5) as follows:

$$x_{i+1} = x_i + \frac{\alpha_i}{2\beta_i}\Delta h\,(x_i, \beta_i\xi_i)\,\xi_i$$

The variables ξ_i are uniformly distributed random variables on the unit sphere, i.e. $\|\xi\| = 1$, and we define

$$\Delta h\,(x, y) = h\,(x + y) - h\,(x - y) \approx 2\|y\|\nabla h(x)$$

This method will not necessarily travel along the steepest slope, but this is a feature of this method to help to overcome the problem of local extrema.

5.7.3. Simulated Annealing

This method is actually motivated by physics: consider a system at temperature T which is assumed to be in equilibrium. It will have a mean energy and the actual energy of the system will fluctuate around this mean value

$$\overline{h} = \int \mathrm{d}x\, h(x)\, \mathrm{e}^{-\frac{h(x)}{T}}$$

When one slowly decreases the temperature towards $T = 0$, the system will slowly approach its ground state, the state with the smallest energy.

Now we abstract from the physical idea to some function $h(x)$ we want to minimise. The algorithm goes as follows: start with some value x_i. Generate ξ from an instrumental distribution $g\,(|\xi - x_i|)$ and generate the next value x_{i+1} as follows:

$$x_{i+1} = \begin{cases} \xi & \text{with probability } \mathbb{P}_{\text{acc}} = \min\left(\exp\left(-\frac{\Delta h}{T}\right), 1\right), \\ x_i & \text{with probability } 1 - \mathbb{P}_{\text{acc}}\,, \end{cases} \tag{5.7.6}$$

with $\Delta h = h(\xi) - h\,(x_i)$. For the instrumental distribution there are other choices than $g\,(|\xi - x_i|)$ possible, but in the form given above the update at fixed temperature corresponds to the original Metropolis update, which will be discussed in chapter 6 in detail. The acceptance step Eq. 5.7.6 distinguishes two cases

1. if $h\,(\xi) < h\,(x_0)$, ξ is always accepted, because $\exp\left(-\frac{\Delta h}{T}\right)$ is always larger than 1.
2. if $h\,(x_0) < h\,(\xi)$, there is still non-zero probability for the proposed ξ to be accepted. This property helps to be not get trapped in local minima.

After a certain number of so-called updates the temperature T is lowered towards $T = 0$ and the procedure is repeated. The algorithm for this so-called simulated annealing (SA) method is summarised in algorithm 5.7.1.

Unfortunately, there is only a very limited number of exact results available on how to lower the free parameter T exactly. Unfortunately, this choice (together with the choice of the instrumental distribution g) crucially influences the efficiency of the algorithm. It determines how fast the algorithm will move on the surface of h. A higher acceptance rate leads to more movement on the surface of h, which is beneficial, since we seek to scan the surface for the location of the global extremum. But a high acceptance rate should not come at the expense of suggesting new candidate states, which differ only very

Algorithm 5.7.1 Simulated Annealing

1: Initialise with some x_0 and T_0
2: **for** j=1,2,...,M **do**
3: **for** i=1,2,...,N **do**
4: Generate $\xi \sim g(|\xi - x_i|)$
5: Accept $x_{i+1} = \xi$ with probability

$$P_{\text{acc}} = \min\left\{1, \exp\left(-\frac{\Delta h}{T}\right)\right\}$$

6: If ξ is not accepted keep $x_{i+1} = x_i$
7: **end for**
8: Update $T \to T_j < T_{j-1}$
9: **end for**

little from the current one, since then moves away from the current configuration maybe likely, but they proceed with very small distance. This again makes the movement on the surface of h inefficient on a global scale. So a balance has to be found.

Note, that since T is step wise lowered to zero at a certain rate (so the scale $1/T$ multiplying the change in h increases), the algorithm will with high probability gradually be confined to move in a shrinking area, be attracted by some local extremum and stay there, because due to the large scale, the acceptance probability for a change with $\Delta h > 0$ becomes tiny. How fast this happens depends on the rate at which T is lowered. So we may want to lower T at a small enough rate. It turns out, a polynomial choice like $T_j \propto 1/(j+1)$ or similar is often decaying too fast for small j. And in general a polynomial decrease is not optimal.

Instead, one typically reduces T for instance logarithmically or geometrically like

$$T_j = \frac{\Gamma}{\log(j)} \quad \text{or} \quad T_j = \alpha^j T_0\,, \quad (0 < \alpha < 1)\,. \tag{5.7.7}$$

The parameters Γ and α may then be tuned such as to obtain maximal efficiency, i.e. to tune the acceptance rate to a sufficient level. For a finite search space, the logarithmic rate can be shown to lead to convergence for certain values of Γ.

Example 5.7.1

The Hamiltonian of the Ising–model was introduced in chapter 1 as

$$H(s_1, ..., s_N) = -J\sum_{\langle i,j\rangle} s_i s_j + h\sum_i s_i\,, \qquad s_i \in \{-1, 1\}\,.$$

Lets assume we want to find the ground state configuration of the model on a square discrete lattice. Metropolis et al. invented SA in 1953 [6] for this purpose. The

SA algorithm for the ISING–model reads as follows

1. start with a random configuration of spins

$$s^{(k)} = \{s_1, \ldots, s_n\} \qquad\qquad k = 0$$

   Set $T_k = T > 0$

2. For all sites i of the lattice do
   a) flip spin $s_i \to s_i' = -s_i$ at site i
   b) compute the energy difference $\Delta H \stackrel{\text{def}}{=} H(\ldots, s_i', \ldots) - H(\ldots, s_i, \ldots)$ (see Eq. 7.8.1 for an explicit formula)
   c) accept the spin flip with probability

$$\mathbb{P}_{\text{acc}} = \min\left\{1, \exp\left(-\frac{\Delta H}{T}\right)\right\}$$

3. Update T, e.g.

$$T_{k+1} = \frac{\Gamma}{\log(k+1)}$$

4. $k = k + 1$ and restart from 2.

For the practical use of this algorithm some remarks are in order:

- It is useful to restart the algorithm with different initial random configurations (Monte-Carlo on the initial configuration).
- One sequence is stopped if the change in energy becomes smaller than some ϵ.
- if the spin s_i of site i is flipped, the ΔH can be computed as follows:

$$\begin{aligned}\Delta H &= H(s_1, \ldots, s_{i-1}, s_i, s_{i+1}, s_n) - H(s_1, \ldots, s_{i-1}, -s_i, s_{i+1}, s_n) \\ &= -J \sum_{\mu} 2 s_i \left(s_{i+\hat{\mu}} + s_{i-\hat{\mu}}\right) - 2 h s_i \,,\end{aligned}$$

  where μ runs over all directions and $\hat{\mu}$ is a unit vector in direction μ.

5.8. Monte-Carlo Optimisation with R

We start again with example 5.1.1 with the function to be integrated over reading in `R`

```
f <- function(x) {
  return( (cos(50*x) + sin(20*x) )^2 )
}
```

First we try a simple uniform search, implemented by the following function

```
simple.search <- function(N=1000, h) {
  res <- array(NA, dim=c(N,2)) ## array to store the search history
  res[,1] <- runif(n=N)
  res[,2] <- h(res[,1])
  return(invisible(res))
}
```

which we may call as follows

```
max(simple.search(N=100, h=f)[,2])

## [1] 3.816355

max(simple.search(N=500, h=f)[,2])

## [1] 3.832541

max(simple.search(N=1000, h=f)[,2])

## [1] 3.832314
```

Next, we define a function implementing the simulated annealing algorithm in R

```
simulated.annealing <- function(N=1000, h, r=0.5, x0=0) {
  res <- array(NA, dim=c(N+1,2)) ## array to store the search history
  res[1,] <- c(x0, h(x0))        ## initial values
  xold <- x0
  hold <- h(xold)
  for(j in 1:N) {
    T <- 1. / log(j+1)
    a <- max( c(xold - r, 0))
    b <- min( c(xold + r, 1))
    xnew <- runif(1) * (b-a) + a ## proposal x
    hnew <- h(xnew)              ## h at x
    ## accept-reject step
    p <- min( c(exp( (hnew-hold)/T ), 1) )
    u <- runif(1)
    if( u < p ) {                ## if accepted
```

```
      xold <- xnew
      hold <- hnew
    }
    res[(j+1),] <- c(xold, hold) ## store next point
  }
  return(invisible(res))
}
```

This function can be called, using the default values for `r` and `x0`, as follows

```
res <- simulated.annealing(N=100, h=f)
res[101,2]

## [1] 3.618042

max(res[,2])

## [1] 3.828477

res <- simulated.annealing(N=500, h=f)
res[501,2]

## [1] 3.760275

max(res[,2])

## [1] 3.831014

res <- simulated.annealing(N=1000, h=f)
res[1001,2]

## [1] 3.830259

max(res[,2])

## [1] 3.83216
```

So, without tuning of parameters, the simple uniform search works as good as SA for this particular problem. At least if we take the `max` operation over the full history. Taking only the last point from SA is slightly worse than the simple search result. This is mainly because of the very particular, one-dimensional function we are working with. In higher dimensions, SA will quickly outperform the simple search approach.

References

[1] RE Caflisch. "Monte Carlo and quasi-Monte Carlo methods". In: *Acta Numerica* (1998), pp. 1–49.

[2] D Christophe and S Petr. *randtoolbox: Generating and Testing Random Numbers*. R package version 1.17. 2015.

[3] JH Halton. "Algorithm 247: Radical-inverse Quasi-random Point Sequence". In: *Commun. ACM* 7.12 (Dec. 1964), pp. 701–702. ISSN: 0001-0782. DOI: 10.1145/355588.365104. URL: http://doi.acm.org/10.1145/355588.365104.

[4] E Hlawka. "Funktionen von beschränker Variation in der Theorie der Gleichverteillung". In: *Ann. Mat. Pura Appl.* 54 (1961), pp. 325–333.

[5] J Koksma. "Een algemeene stelling inuit de theorie der gelijkmatige verdeeling modulo 1". In: *Mathematica (Zutphen B)* 11 (1942-43), pp. 7–11.

[6] N Metropolis, A Rosenbluth, MN Rosenbluth, AH Teller, and E Teller. "Equation of State Calculations by Fast Computing Machines". In: *J. Chem. Phys.* 21 (1953), p. 1087. DOI: 10.1063/1.1699114.

[7] B Moskowitz and R Caflisch. "Smoothness and dimension reduction in Quasi-Monte Carlo methods". In: *Mathematical and Computer Modelling* 23.8 (1996), pp. 37–54. ISSN: 0895-7177. DOI: http://dx.doi.org/10.1016/0895-7177(96)00038-6. URL: http://www.sciencedirect.com/science/article/pii/0895717796000386.

[8] H Niederreiter. "Quasi-Monte Carlo Methods and Pseudo-Random Numbers". In: *AMS* 84.6 (Nov. 1978).

[9] CP Robert and G Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005. ISBN: 0387212396.

[10] RY Rubinstein and DP Kroese. *Simulation and the Monte Carlo Method*. 2nd ed. Wiley Series in Probability and Statistics. Wiley-Interscience, 2008. DOI: 10.1002/978-0-4702-3038-1.

[11] IM Sobol. "The distribution of points in a cube and the accurate evaluation of integrals". In: *Zh. Vychisl. Mat. Mat. Fiz.* 7 (1967). in Russian, pp. 784–802.

[12] IM Sobol. "Uniformly distributed sequences with additional uniformity property". In: *USSR Comput. Math. Math. Phys.* 16 (1976), pp. 1332–1337.

[13] J Spanier and EH Maize. "Quasi-random methods for estimating integrals using relatively small samples". In: *SIAM Review* 36.1 (1994), pp. 18–44.

6. Markov Chains

As discussed previously, we would like to generate directly from a given distribution, say π, and we have by now seen several ways to achieve this. In this chapter we will discuss another approach, which is based on stochastic processes. The idea is the following: we generate a stochastic process in the state space Ω (or event space)

$$\mu_0 \quad \xrightarrow{P} \quad \mu_1 \quad \xrightarrow{P} \quad \mu_2 \quad \xrightarrow{P} \quad \dots \qquad \mu_i \in \Omega$$

by applying a transition matrix P. P defines the transition probabilities for any state $\mu \in \Omega$ to any other state $\nu \in \Omega$. Typically, in physics every state of a system has some finite, non-zero probability. This is why we require the transition matrix to have the property of *irreducibility*, which basically says that every state $\mu \in \Omega$ can be reached from any other state $\nu \in \Omega$ in a final number of steps.

Now, as the stochastic process goes along, every state $\mu \in \Omega$ should be visited corresponding to some probability π_μ. This probability distribution π is an input, e.g. the Boltzmann distribution, and we would like to design P such as to produce π. In order for this to work, we need that every state $\mu \in \Omega$ is visited over and over again, otherwise it will be left forever by the process at some point. This will be formulated as the notion of *recurrence* and *positive recurrence*.

Finally, if we consider P as a linear map from the state of distributions to itself, it turns out that the distribution π needs to be the invariant distribution of P. In a formula

$$\pi \;=\; \pi P$$

by interpreting π as a row vector. In practice, this is ensured almost always by the so-called *detailed balance* condition, which is why we follow this route here as well.

We will consider only discrete time stochastic processes here. For the continuous case we refer to Ref. [5]. Moreover, in this chapter we will put a bit less emphasis on mathematical rigour in favour of a less technical presentation. All the details and rigorous proofs can be found in the excellent book by Norris [5].

6.1. Stochastic Matrices and Markov–Chains

We first define the transition matrix. It needs to have certain properties, which are summarised in the following definition:

Definition 6.1.1: Stochastic Matrix

*A matrix $P = (p_{\mu\nu} : \mu, \nu \in \Omega)$ is **stochastic** if every row $(p_{\mu\nu} : \nu \in \Omega)$ of P is a distribution as defined in definition 2.2.4.*

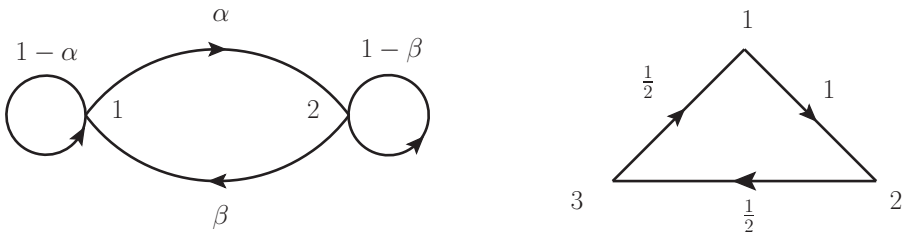


Figure 6.1.1.: Examples of stochastic matrices

Lets look at two examples first:

Example 6.1.1

Consider the most general 2-dimensional stochastic matrix

$$P = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix}, \quad 0 \le \alpha, \beta \le 1\,. \tag{6.1.1}$$

A diagram can be associated with a stochastic matrix showing matrix elements as links as in the left panel of figure Figure 6.1.1 for the 2-dimensional case. The right panel of the figure shows the diagram for the matrix

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1/2 & 1/2 \\ 1/2 & 0 & 1/2 \end{pmatrix}. \tag{6.1.2}$$

Recall the definition 4.7.1 of a stochastic process, which we need now to define what we call a Markov–chain:

Definition 6.1.2: Markov–Chain

A stochastic process $(X_n)_{n\ge 0}$ is a Markov *chain with initial distribution f and transition matrix P if*

1. *X_0 has distribution f.*
2. *for $n \ge 0$, conditional on $X_n = \mu$, X_{n+1} has distribution $(p_{\mu\nu} : \nu \in \Omega)$ and is independent of $X_0, \ldots, X_{n-1}$.*

We say that $(X_n)_{n\ge 0}$ is Markov (f, P). *Also, for $(X_n)_{0\le n\le N}$ we say it is* Markov (f, P) *if the two conditions are fulfilled. Expressed in probabilities, this means*

1. $\mathbb{P}(X_i = \mu_0) = f_{\mu_0}$.
2. $\mathbb{P}(X_{n+1} = \mu_{n+1} \,|\, X_0 = \mu_0, \ldots, X_n = \mu_n) = \mathbb{P}(X_{n+1} = \mu_{n+1} \,|\, X_n = \mu_n) = p_{\mu_n, \mu_{n+1}}$.

In particular, we will only consider transition matrices P independent of time n. In that case we speak of a *time homogeneous* chain. In the following we will need n-step transition probabilities, i.e. n-times applying P, which we will denote as P^n. For the n-step transition probability from state μ to ν we will use $p^{(n)}_{\mu\nu}$, not to be confused with the nth power of $p^n_{\mu\nu}$.

Example 6.1.2

A simple, one dimensional random walk (RW) over $\mathbb{Z}$ is defined at step n by a state $X_n \in \mathbb{Z}$. Transitions to $X_{n+1} = X_n + 1$ (up) occur with probability $0 < p < 1$ and to $X_{n+1} = X_n - 1$ (down) with probability $q = 1 - p$. The state space is given by $\Omega = \mathbb{Z}$ and it is countably infinite. It is easily verified that the simple random walk represents an example for a Markov–chain.

A particular chain $X_0, X_1, \ldots, X_N$ of length N with $X_N - X_0 = r \in \mathbb{Z}$ with $r < N/2$ has probability

$$\mathbb{P}(X_N = X_0 + r | X_0, \ldots, X_N) = \begin{cases} p^{(N+r)/2}(1-p)^{(N-r)/2} & N + r \text{ even}, \\ 0 & \text{otherwise.} \end{cases}$$

It is not possible to reach even (odd) X_N with odd (even) number of steps N. Assume $M = N + r$ to be even. Then, the probability for $X_N - X_0 = r$ is obtained by counting all possible random walks with N steps. It requires $N/2$ steps up in N total steps, hence

$$\mathbb{P}(X_N = X_0 + r | X_0) = \binom{N}{(N+r)/2} p^{(N+r)/2}(1-p)^{(N-r)/2}.$$

We will now introduce and proof some of the very basic properties of Markov–chains. Most importantly, we proof the so-called Markov property, which states that Markov–chains have no memory and restart afresh under certain conditions. We first need the following theorem:

Theorem 6.1.1:

A discrete time random process $(X_n)_{0 \le n \le N}$ is Markov(f, P) if and only if for all $\mu_0, \ldots, \mu_N \in \Omega$

$$\mathbb{P}(X_0 = \mu_0, \ldots, X_N = \mu_N) = f_{\mu_0} p_{\mu_0, \mu_1} \ldots p_{\mu_{N-1}, \mu_N}. \tag{6.1.3}$$

Proof:

This is shown in two steps: first, let $(X_n)_{0\leq n\geq N}$ be Markov $(f,\, P)$. Then it follows by making repeated use of the definition of conditional probability (the so-called sum rule of probabilities)

$$\begin{aligned}&\mathbb{P}\left(X_0=\mu_0,\ldots,X_N=\mu_N\right)\\&\quad=\mathbb{P}\left(X_0=\mu_0\right)\,\mathbb{P}\left(X_1=\mu_1\,|\,X_0=\mu_0\right)\cdot\ldots\cdot\mathbb{P}\left(X_N=\mu_N\,|\,X_0=\mu_0,\ldots,X_{N-1}=\mu_{N-1}\right)\\&\quad=f_{\mu_0}\,p_{\mu_0,\mu_1}\ldots p_{\mu_{N-1},\mu_N}\,.\end{aligned}\tag{6.1.4}$$

Second, let $(X_n)_{0\leq n\geq N}$ fulfil Eq. 6.1.3. We can then sum over $\mu_N\in\Omega$ and use $\sum\limits_{\nu\in\Omega}p_{\mu\nu}=1$ to get

$$\mathbb{P}\left(X_0=\mu_0,\ldots,X_{N-1}=\mu_{N-1}\right)=f_{\mu_0}\,p_{\mu_0,\mu_1}\ldots p_{\mu_{N-2},\mu_{N-1}}\,.\tag{6.1.5}$$

We can repeat this reduction iteratively and find that for all $n=0,\ldots,N$ we have

$$\mathbb{P}\left(X_0=\mu_0,\ldots,X_n=\mu_n\right)=f_{\mu_0}\,p_{\mu_0,\mu_1}\ldots p_{\mu_{n-1},\mu_n}\,.\tag{6.1.6}$$

This means in particular $\mathbb{P}\left(X_0=\mu_0\right)=f_{\mu_0}$ and for all $n=0,\ldots,N-1$ we have by definition of the conditional probability and using Eq. 6.1.6

$$\begin{aligned}\mathbb{P}\left(X_{n+1}=\mu_{n+1}\,|\,X_0=\mu_0,\ldots,X_n=\mu_n\right)&=\frac{\mathbb{P}\left(X_0=\mu_0,\ldots,X_n=\mu_n,X_{n+1}=\mu_{n+1}\right)}{\mathbb{P}\left(X_0=\mu_0,\ldots,X_n=\mu_n\right)}\\&=p_{\mu_n,\mu_{n+1}}\,.\end{aligned}\tag{6.1.7}$$

□

In the following, $\delta_\mu=(\delta_{\mu\nu}\,|\,\mu\in\Omega)$ denotes the distribution which is one for state $\mu=\nu$ and zero otherwise. In case $f_\mu\neq 0$ we will also use the notation $\mathbb{P}_\mu\left(A\right)$ for the conditional probability $\mathbb{P}(A|X_0=\mu)$.

Theorem 6.1.2: Markov Property

If $(X_n)_{n\geq 0}$ is Markov $(f,\, P)$, then, conditional on $X_m=\mu$, $(X_{m+n})_{n\geq 0}$ is Markov $(\delta_\mu,\, P)$ and independent of $X_0,\ldots,X_m$.

Proof:

To see this, one needs to show that the chain after time m is determined by the transition matrix P, and that it is independent of any event that happened before time m. As we recall, independence in the probabilistic sense means the probability for the intersection of two events "factorises". So what is to be shown is that for any event A, which is determined by $X_0,\ldots,X_m$ it holds that

$$\begin{aligned}&\mathbb{P}\left(\{X_m=\mu_m,\ldots,X_{m+n}=\mu_{m+n}\}\cap A\,|\,X_m=\mu\right)\\&\quad=\delta_{\mu,\mu_m}\,p_{\mu_m,\mu_{m+1}}\ldots p_{\mu_{m+n-1},\mu_{m+n}}\,\mathbb{P}\left(A\,|\,X_m=\mu\right)\,.\end{aligned}\tag{6.1.8}$$

This is again done in two steps: first we look at *elementary* events of the form $A = \{X_0 = \mu_1, \ldots, X_m = \mu_m\}$, i.e. events characterised by a specific sequence of states $\mu_0, \ldots, \mu_m \in \Omega$. In such a case the left-hand side of eq. Eq. 6.1.8 becomes

$$\frac{\mathbb{P}\left(X_0 = \mu_1, \ldots, X_{m+n} = \mu_{m+n} \text{ and } \mu = \mu_m\right)}{\mathbb{P}\left(X_m = \mu\right)}$$
$$= \delta_{\mu,\mu_m}\, p_{\mu_m,\mu_{m+1}} \cdots p_{\mu_{m+n-1},\mu_{m+n}}\, \frac{\mathbb{P}\left(X_0 = \mu_1, \ldots, X_m = \mu_m \text{ and } \mu = \mu_m\right)}{\mathbb{P}\left(X_m = \mu\right)}. \quad (6.1.9)$$

This equality follows from Eq. 6.1.3 shown above. The fraction on the right-hand side is precisely $\mathbb{P}\left(A \,|\, X_m = \mu\right)$ in this case.

The case for any event determined by $X_0, \ldots, X_m$ follows from writing A as an at most *countable disjoint union* of elementary events, $A = \bigcup_k A_k$ and sum the relations Eq. 6.1.9 for the individual A_k.

□

The interpretation of this result is that a Markov–chain has no memory. Or, in other words, it starts afresh from $X_m = \mu$ independent of $X_m, X_{m-1}, ..., X_0$. We will make frequent use of this property in the following.

6.2. Irreducibility

With these important properties of Markov–chains at hand, we turn now to the question which state can be reached from a given state in a chain. As mentioned in the introduction to this chapter, the crucial aspect here for us is that there should be a non-zero probability to reach each state in Ω from any given state in a finite number of steps. To discuss this one uses the concept of *Communicating classes* of the chain. We say

(i) "μ leads to ν" or $\mu \to \nu$, if $\mathbb{P}_\mu\left(X_n = \nu\right) > 0$ for some $n \geq 0$

(i) "μ communicates with ν" or $\mu \leftrightarrow \nu$, if $\mu \to \nu$ and $\nu \to \mu$

The relation "communicates with" is an *equivalence relation*, as one can show easily. Under this equivalence relation, the state space Ω can be partitioned into *communicating classes*. Such a partition C is called *irreducible* if all $\mu, \nu \in C$ are communicating.

Definition 6.2.1: Irreducible Transition Matrix

If the complete state space Ω forms a single communicating class we call the transition matrix P ***irreducible****.*

This is one of the major conditions we will have to put on the stochastic matrices we use to generate a Markov–chain.

Example 6.2.1

Consider the simple random walk introduced in example 6.1.2.. In order to understand whether or not it is an irreducible MARKOV–chain, we have to compute the probility to reach state $b \in \mathbb{Z}$ from state $a \in \mathbb{Z}$. Every state b can be reached from state a in $N = |b - a|$ steps with probability p^N for $b > a$ and $(1-p)^N$ for $a > b$. Therefore, the simple random walk is an irreducible MARKOV–chain.

Of course, there are more ways to reach b from a. But it is sufficient to show that the probability is non-zero.

6.3. Strong Markov Property

We have already shown the MARKOV property. We are now going to proof a stronger version of it: the MARKOV property means that a MARKOV–chain starts afresh for each time m, conditional on $X_m = \mu$. Now we will formulate it such that we wait to hit the state μ at some random time $T = m$ conditional on $X_1, \ldots, X_m$ and $X_m = \mu$. And we show that the MARKOV–chain then starts afresh from μ.

Definition 6.3.1: Stopping Time

A random variable $T : \Omega \to \{0, 1, 2, 3, \ldots\} \cup \{\infty\}$ *is called a* ***stopping time*** *if the event* $\{T = n\}$ *depends only on* $X_0, X_1, \ldots, X_n$ *for* $n = 1, 2, 3, \ldots$.

It should be intuitively clear what a stopping time is.

Example 6.3.1

The first passage time (or *return time*) of a MARKOV–chain Markov (f, P)

$$T_\mu \overset{\text{def}}{=} \inf\{n \geq 1 : X_n = \mu\}$$

is a stopping time, because it depends only on $X_1, \ldots, X_n$, namely by the condition $X_1 \neq \mu, X_2 \neq \mu, \ldots, X_{n-1} \neq \mu, X_n = \mu$. The last exit time is, on the other hand, not a stopping time.

We now show that the MARKOV property holds at stopping times. At first sight this looks like nothing new compared to the MARKOV property. The difference comes from the fact that in contrast to m, $T = m$ depends on all $X_0, X_1, \ldots, X_m$ and $X_m = \mu$. Hence, the *strong* MARKOV *property* is an additional property going beyond the MARKOV property.

Theorem 6.3.1: Strong Markov Property

If $(X_n)_{n\geq 0}$ is Markov $(f,\, P)$ and T a stopping time of $(X_n)_{n\geq 0}$, then, conditional on $T<\infty$ and $X_T=\mu$, $(X_{T+n})_{n\geq 0}$ is Markov $(\delta_\mu,\, P)$ and independent of $X_0,\ldots,X_T$.

Proof:

The statement is about independence of the chain after stopping time T from any X_n with $n<T$. We will again proof this by showing that the probability factorises. Let B be any event determined by $X_0,\ldots,X_T$. Then it follows that $B\cap\{T=m\}$ is determined by $X_0,\ldots,X_m$. Now we can use the MARKOV property at time m, since m is only conditional on $X_m=\mu$, which is precisely the realm of the MARKOV property. Moreover, we assumed that B is determined by X_k up to $k\leq m$.

$$\begin{aligned}&\mathbb{P}\left(\{X_T=\nu_0,\ldots,X_{T+n}=\nu_n\}\cap B\cap\{T=m\}\cap\{X_T=\mu\}\right)\\&\quad=\mathbb{P}_\mu\left(X_0=\nu_0,\ldots X_n=\nu_n\right)\,\mathbb{P}\left(B\cap\{T=m\}\cap\{X_T=\mu\}\right)\,.\end{aligned}\tag{6.3.1}$$

The events $\{T=m\}$ for different values of m are mutually exclusive, so now the relation Eq. 6.3.1 can be summed over $m=0,1,2,\ldots$ and divided by $\mathbb{P}\left(T<\infty, X_T=\mu\right)$, which will give the conditional probabilities

$$\begin{aligned}&\mathbb{P}\left(\{X_T=\nu_0,\ldots X_{T+n}=\nu_n\}\cap B\,|\,\{T<\infty, X_T=\mu\}\right)\\&\quad=\mathbb{P}_\mu\left(X_0=\nu_0,\ldots X_n=\nu_n\right)\,\mathbb{P}\left(B\,|\,T<\infty, X_T=\mu\right)\,.\end{aligned}\tag{6.3.2}$$

This is the factorisation showing the independence.

□

6.4. Positive Recurrent

Next, we need to introduce the notion of recurrence. This is motivated by the fact that we will generate a long MARKOV–chain and have to make sure that every state will appear with the correct probability. This requires that the chain keeps coming back to all the states. Since a chain has an infinite number of elements, every state must be visited infinitely often. A state μ is said to be *recurrent* if

$$\mathbb{P}_\mu\left(X_n=\mu \text{ for infinitely many } n\right)\;=\;1\tag{6.4.1}$$

and *transient* if

$$\mathbb{P}_\mu\left(X_n=\mu \text{ for infinitely many } n\right)\;=\;0\,.\tag{6.4.2}$$

A transient state is one that the chain will eventually leave forever. This motivates the definition of *return time* for a given state μ:

Definition 6.4.1: Return Time

$$\tau_\mu\;=\;\inf\left(n\geq 1:\;X_n=\mu|X_0=\mu\right)\tag{6.4.3}$$

is called the return time of state μ given $X_0 = \mu$. We set $\tau_\mu = \infty$ if $X_n \neq \mu$ for all $n \geq 1$. The return time is very similar to the first passage time and it is also a stopping time.

Note that the first passage time can be extended to the rth passage time recursively. Also the rth passage time is a stopping time.

From this we define the probability of ever returning to μ when started from state μ, the so-called *return probability* as follows

$$r_\mu \stackrel{\text{def}}{=} \mathbb{P}\left(\tau_\mu < \infty\right) . \tag{6.4.4}$$

Thus, a recurrent state has $r_\mu = 1$ and a transient state $r_\mu < 1$. The strong Markov property tells us that each time the state μ is visited the probability to visit μ again is given by r_μ, because τ_μ is a *stopping time*. And the chain starts afresh independent of the past. The visits to state μ can be counted by

$$N_\mu \stackrel{\text{def}}{=} \sum_{n=0}^{\infty} \mathbb{1}_{\{X_n=\mu | X_0=\mu\}} \tag{6.4.5}$$

which has a geometric distribution including the initial visit in the counting

$$\mathbb{P}\left(N_\mu = n\right) = r_\mu^{n-1}(1 - r_\mu) , \quad n \geq 1 .$$

Clearly, a state μ is recurrent if $\mathbb{E}(N_\mu) = \infty$. This expectation value reads

$$\mathbb{E}\left(N_\mu\right) = \sum_{n=0}^{\infty} p_{\mu\mu}^{(n)} .$$

which leads to the following result. A state μ is recurrent if and only if the n–step transition probability fulfils

$$\sum_{n=0}^{\infty} p_{\mu\mu}^{(n)} = \infty \tag{6.4.6}$$

and transient otherwise.

Example 6.4.1

For the simple random walk introduced in example 6.1.2 we can discuss whether or not it is recurrent or transient using the property spelled out in Eq. 6.4.6. Without loosing generality we start at $X_0 = 0$ and, hence, we are interested in the N-step transition probability $p_{00}^{(N)}$. This is clearly zero if N is odd and, therefore, we can restrict ourselves to computing $p_{00}^{(2N)}$ with $N \in \mathbb{N}_0$. With the result from example 6.1.2 we know

$$p_{00}^{(2N)} = \binom{2N}{N} p^N (1-p)^N .$$

For large n, $n!$ can be approximated by STIRLING'S formula reading

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{\mathrm{e}}\right)^n \quad \text{as} \quad n \to \infty\,.$$

To be precise, the ratio of left hand and right hand side tends to one as $n \to \infty$. We can use it to write

$$p_{00}^{(2N)} = \frac{(4p(1-p))^N}{\sqrt{\pi N}} \quad \text{as} \quad N \to \infty\,.$$

We consider two cases:

- symmetric case with $p = q = (1-p) = 1/2$:

  In this case we have $(4pq)^n = 1$. With the asymptotic behaviour at hand, we can find some $M > 0$, such that for all $N \geq M$ we have

  $$p_{00}^{(2N)} \geq \frac{1}{2\sqrt{\pi N}}$$

  from which we conclude

  $$\sum_{N=M}^{\infty} p_{00}^{(2N)} \geq \frac{1}{2\sqrt{\pi}} \sum_{N=M}^{\infty} \frac{1}{\sqrt{N}} = \infty\,.$$

  Therefore, the simple symmetric random walk with $p = q$ is recurrent.

- the asymmetric case with $p \neq q$ can be shown to be transient because we find with the same arguments

  $$\sum_{N=M}^{\infty} p_{00}^{(2N)} \leq \frac{1}{\sqrt{\pi}} \sum_{N=M}^{\infty} (4pq)^N < \infty$$

  since $0 < 4pq < 1$.

Theorem 6.4.1:

Let C be a communicating class, then all states in C are either recurrent or transient.

Proof:

Let $\mu \neq \nu$ be communicating states and n chosen such that $p = p_{\mu\nu}^{(n)} > 0$. If μ is recurrent it will be visited an infinite number of times. Every time it is visited there is the same probability $p > 0$ that ν will be visited n steps later. Thus, also ν will be visited an infinite number of times.

□

This means in particular that an irreducible MARKOV–chain has either only recurrent or only transient states.

Definition 6.4.2: Recurrent Markov–Chain

An irreducible Markov*–chain is called recurrent if all states in* Ω *are recurrent.*

Definition 6.4.3: Positive Recurrent

A recurrent state μ *is called positive recurrent if the expected return time is finite, i.e.*

$$m_\mu \;=\; \mathbb{E}\left(\tau_\mu\right) \;<\; \infty\,. \tag{6.4.7}$$

Otherwise, μ *is called null recurrent.*

One can show the following result (for the proof see e.g. Ref. [5])

Theorem 6.4.2:

For an irreducible Markov–chain all states are either positive recurrent, null recurrent or transient.

An irreducible Markov–chain with all states (positive) recurrent, is called a (positive) recurrent Markov–chain.

Example 6.4.2

Consider again the simple and symmetric random walk over $\mathbb{Z}$ from example 6.1.2 with $p = q = 1/2$. We know already that it is irreducible and recurrent. As we have shown in the previous example, the chain can return only after an even number of steps to a given visited state μ. The probability to first return to state μ after $\tau_\mu = 2N$ steps is given by

$$\mathbb{P}_{\text{fr}}^{2N} \;=\; \frac{1}{2N-1}\binom{2N}{N}\cdot\frac{1}{2^{2N}}\,,$$

since there are $2N$ choose N walks to return to a given state, we have to divide by $2N-1$ to choose only those walks which are first returns and we have to divide by the total number of walks of length $2N$ given by 2^{2N}. We see again that the simple symmetric random walk is recurrent, since the return probability is

$$\mathbb{P}_{\text{return}} \;=\; \sum_{N=1}^{\infty}\mathbb{P}_{\text{fr}}^{2N} \;=\; 1\,.$$

But, the expectation value of $\tau_\mu = 2N$ is given as follows

$$\mathbb{E}\left(\tau_\mu\right) \;=\; \sum_{N=1}^{\infty} 2N\;\mathbb{P}_{\text{fr}}^{2N} \;=\; \infty\,.$$

Therefore, the simple symmetric random walk is null–recurrent.

6.5. Limiting Distributions

Let us introduce the following notation for the long run proportion of time a MARKOV–chain spends in state μ by the stochastic limit

$$\pi_\mu \overset{\text{def}}{=} \lim_{n\to\infty} \frac{1}{n} \sum_{m=1}^{n} \mathbb{1}_{\{X_m=\mu | X_0=\nu\}} \,, \tag{6.5.1}$$

i.e. almost surely. This requires of course the assumption that the limit exists, which is not always the case. If π_μ exists, it can hence be computed via

$$\begin{aligned} \pi_\mu &\overset{\text{a.s.}}{=} \lim_{n\to\infty} \frac{1}{n} \sum_{m=1}^{n} \mathbb{P}\left(X_m = \mu | X_0 = \nu\right) \\ &= \lim_{n\to\infty} \frac{1}{n} \sum_{m=1}^{n} p_{\nu\mu}^{(m)} \end{aligned} \tag{6.5.2}$$

for all initial states ν and again the limit is to be understood in the stochastic sense.

Definition 6.5.1: Stationary Distribution

*$\pi = (\pi_{\mu_0}, \pi_{\mu_1}, \ldots)$ is called **stationary distribution** of the* MARKOV*–chain if π_μ exists for every $\mu \in \Omega$, if it is independent of the initial state ν and $\sum_\mu \pi_\mu = 1$.*

In other words, if we average the m-step transition matrix P^m like in Eq. 6.5.2, each row converges to the stationary probabilities π:

$$\lim_{n\to\infty} \frac{1}{n} \sum_{m=1}^{n} P^m \overset{\text{a.s.}}{=} \begin{pmatrix} \pi_{\mu_0} & \pi_{\mu_1} & \cdots \\ \pi_{\mu_0} & \pi_{\mu_1} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} . \tag{6.5.3}$$

Theorem 6.5.1:

Let $(X_n)_{n\geq 0}$ be Markov $(\delta_\nu,\, P)$ with P positive recurrent and irreducible. Then a unique stationary distribution π exists and is given for all states $\mu \in \Omega$ as

$$\pi_\mu \overset{\text{a.s.}}{=} \frac{1}{\mathbb{E}(\tau_\mu)} > 0 \tag{6.5.4}$$

independent on ν.

Proof:

Since P is irreducible and positive recurrent, the probability to reach state μ from ν is non-zero and μ will be reached in a finite number of steps. Once μ is visited for the first

time the MARKOV–chain starts afresh and, hence, we can set $X_0 = \mu$ without loosing generality.

Let $t_0 = 0, t_1 = \tau_\mu, t_2 = \min\{k > t_1 | X_k = \mu\}, \ldots$ denote the subsequent return times to state μ ($t_n = \min\{k > t_{n-1} | X_k = \mu\}$), i.e. the nth passage time. Define the random variable $Y_n = t_n - t_{n-1}$ for $n > 0$. Hence, Y_n is the time to wait for the next visit of state μ after then $n-1$st visit. Now the proof comes from the following ingenious argument: the strong MARKOV property ensures that the chain starts afresh every time the chain hits μ, independently of the past of the chain. Therefore, the Y_n are **iid** with $\mathbb{E}(Y_n) = \mathbb{E}(\tau_\mu)$ for all $n > 0$. The number of visits to state μ at time $t_n = Y_1 + \ldots + Y_n$ is just equal to n and, therefore, for $t_n \leq m < t_{n+1}$ we have

$$\sum_{k=1}^{m} \mathbb{1}_{\{X_k = \mu | X_0 = \mu\}} \;=\; n\,.$$

Thus, we find

$$\begin{aligned} \pi_\mu \;&=\; \lim_{m\to\infty} \frac{1}{m} \sum_{k=1}^{m} \mathbb{1}_{\{X_k=\mu|X_0=\mu\}} \;=\; \lim_{m\to\infty} \frac{n}{m} \\ &=\; \lim_{n\to\infty} \frac{n}{m} \;=\; \lim_{n\to\infty} \frac{n}{\sum_{i=1}^{n} Y_i} \\ &\overset{\text{a.s.}}{=}\; \frac{1}{\mathbb{E}(\tau_\mu)} \;> 0 \end{aligned} \tag{6.5.5}$$

for all $\mu \in \Omega$. Here, we have used the strong law of large numbers theorem 2.5.2. For the null recurrent case one sees from above that $\pi_\mu \overset{\text{a.s.}}{=} 0$ for all $\mu \in \Omega$. In the transient case $\pi_\mu \overset{\text{a.s.}}{=} 0$ by definition.

□

This result shows that a chain of single states

$$\mu_0 \quad \xrightarrow{P} \quad \mu_1 \quad \xrightarrow{P} \quad \mu_2 \quad \xrightarrow{P} \quad \ldots$$

will produce states distributed like π under the conditions of the theorem. But so far we have only shown the existence and uniqueness of the stationary distribution π. We still need means to construct π.

6.6. Invariant Distribution

As discussed in the introduction to this chapter, we are interested in invariant distributions of MARKOV–chains. The condition to show that a MARKOV–chain has an invariant distribution used in practice is the so-called *detailed balance* condition:

Definition 6.6.1: Detailed Balance

A stochastic matrix P and a distribution f are said to be in ***detailed balance*** *if*

$$f_\mu \, p_{\mu\nu} \;=\; f_\nu \, p_{\nu\mu} \,, \qquad \forall \mu, \nu \,\in \Omega \,. \tag{6.6.1}$$

Written differently, detailed balance states that

$$\frac{f_\mu}{f_\nu} \;=\; \frac{p_{\nu\mu}}{p_{\mu\nu}} \,,$$

telling that the flow from $\mu \to \nu$ equals to the reverse one $\nu \to \mu$. With detailed balance one can immediately formulate the following result:

Theorem 6.6.1:

If P and π are in detailed balance, then $\pi = \pi P$ and π is called *invariant distribution* of P.

Proof:
We have

$$(\pi \, P)_\mu \;=\; \sum_{\nu \in \Omega} \pi_\nu \, p_{\nu\mu} \;=\; \sum_{\nu \in \Omega} \pi_\mu \, p_{\mu\nu} \;=\; \pi_\mu \,,$$

since $\sum_\nu p_{\mu\nu} = 1$ for all $\mu \in \Omega$.

□

Example 6.6.1

Consider the update procedure applied in the SA algorithm presented in the last chapter at fixed temperature T. It actually defines the Metropolis algorithm. This procedure fulfils detailed balance:

For the Metropolis algorithm, the transition matrix element $P\,(x \to y)$ for a transition from x to y has two elements:

1. the probability to propose a new state y conditional on x; this conditional probability distribution is symmetric in x and y, i.e. the probability to propose y conditional on x is the same as the one to propose x conditional on y; we call this probability $g\,(|y - x|)$, why will become clear later on; note that e.g. for the 2-dim. Ising model using single-spin-flip dynamics $g(|y - x|)$ is zero for any configuration y, unless it differs from x in a single spin;

2. the probability to accept the proposed configuration and keep the previous one in case of rejection.

With the invariant distribution we $\pi(x) \propto \exp -S(x)$ we then have

$$\mathbb{P}(x \to y) = g(|y-x|) \begin{cases} \frac{\pi(y)}{\pi(x)} & \pi(y) < \pi(x) \\ 1 & \text{otherwise} \end{cases} = g(|y-x|) \min\left\{\frac{\pi(y)}{\pi(x)}, 1\right\}$$

$$\frac{\pi(y)}{\pi(x)} = \exp\left(-\left(S(y) - S(x)\right)\right) = \exp(-\Delta S)\,.$$

One then has

$$\begin{aligned} \pi(x)\,\mathbb{P}(x \to y) &= \pi(x)\, g(|y-x|) \begin{cases} \frac{\pi(y)}{\pi(x)} & \pi(y) < \pi(x) \\ 1 & \pi(y) \geq \pi(x) \end{cases} \\ &= \pi(y)\, g(|x-y|) \begin{cases} 1 & \pi(x) > \pi(y) \\ \frac{\pi(x)}{\pi(y)} & \pi(x) \leq \pi(y) \end{cases} = \pi(y)\,\mathbb{P}(y \to x)\,. \end{aligned} \tag{6.6.2}$$

Theorem 6.6.2:

Let $(X_n)_{n\geq 0}$ be an irreducible Markov–chain with transition matrix P. Let π be the invariant distribution of P, i.e. $\pi = \pi P$. Then $(X_n)_{n\geq 0}$ is positive recurrent and π is the unique invariant distribution of P.

Proof:

Assume the chain is either transient or null recurrent. Then

$$\lim_{n\to\infty} \frac{1}{n} \sum_{m=1}^{n} P^m \overset{\text{a.s.}}{=} 0 \tag{6.6.3}$$

with probability one. We have $\pi = \pi P$ from which follows

$$\pi \;=\; \pi P^m\,, \qquad m > 0\,.$$

We multiply Eq. 6.6.3 from the left with π and obtain

$$\pi \frac{1}{n} \sum_{m=1}^{n} P^m = \frac{1}{n} \sum_{m=1}^{n} \pi P^m = \pi\,,$$

which would yield $\pi = 0$. This is a contradiction and, hence, the chain must be positive recurrent.

Now assume there exists a stationary solution π' given by

$$\pi'_\mu \;=\; \frac{1}{\mathbb{E}(\tau_\mu)}\,.$$

We multiply both sides of Eq. 6.5.3 with π leading to

$$\pi \;=\; \pi \begin{pmatrix} \pi'_{\mu_0} & \pi'_{\mu_1} & \cdots \\ \pi'_{\mu_0} & \pi'_{\mu_1} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}\,.$$

Since $\sum_\mu \pi_\mu = 1$ it follows $\pi = \pi'$.

□

Note that one can also proof the other direction.

Now we have shown that every irreducible MARKOV–chain with a stochastic matrix P in detailed balance with distribution π is positive recurrent and it has π as its unique stationary distribution.

One can proof stronger convergence. Instead of the average convergence in Eq. 6.5.3, one can also show

$$\pi_\mu \;=\; \lim_{n\to\infty} \mathbb{P}\left(X_n = \mu\right) \,. \tag{6.6.4}$$

For this one needs the notion of aperiodicity.

Definition 6.6.2: Aperiodic

*A state μ is called **aperiodic** if $p_{\mu\mu}^{(n)} > 0$ for all sufficiently large n.*

In particular, a state is aperiodic if $p_{\mu\mu} > 0$. Again, one can show that if one state μ of an irreducible transition matrix P is aperiodic, then all states are aperiodic. In this case we call P irreducible and aperiodic. For this case one can proof the following theorem

Theorem 6.6.3:

Let P be irreducible and aperiodic and let π be an invariant distribution of P. Let f be any distribution and suppose $(X_n)_{n\geq 0}$ is Markov $(f,\, P)$. Then

$$\mathbb{P}\left(X_n = \mu\right) \;\to\; \pi_\mu \tag{6.6.5}$$

for $n \to \infty$ and for all $\mu \in \Omega$. In particular,

$$p_{\mu\nu}^{(n)} \;\to\; \pi_\mu \tag{6.6.6}$$

as $n \to \infty$ and for all $\mu, \nu \in \Omega$.

The proof is rather lengthy and it can be found in Ref. [5]. As a counter example for a not aperiodic and, hence, periodic P recall example 6.6.2.

Example 6.6.2

Consider the transition matrix

$$P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \,. \tag{6.6.7}$$

One can easily verify that $P^{2m} = \mathbb{1}$ and $P^{2m+1} = P$, $m \in \mathbb{N}$. This is a counter example where the limit Eq. 6.6.4 does not exist. In this particular case one speaks of a *limit cycle*.

6.7. Time Reversal

The detailed balance condition implies even more. Consider a MARKOV–chain with time reversed. What can we say about this chain?

Definition 6.7.1: reversible

Let $(X_n)_{n\geq 0}$ *be* Markov $(f,\, P)$ *with* P *irreducible. We say that* $(X_n)_{n\geq 0}$ *is* ***reversible*** *if, for all* $N \geq 1$, $(X_{N-n})_{0\leq n\leq N}$ *is also* Markov $(f,\, P)$.

Theorem 6.7.1:

Let P be an irreducible stochastic matrix in detailed balance with distribution π and let $(X_n)_{n\geq 0}$ be Markov $(\pi,\, P)$. Then $(X_n)_{n\geq 0}$ is reversible.

Proof:
From detailed balance it follows that π is the invariant distribution of P, so we have

$$\pi_\nu p_{\nu\mu} \;=\; \pi_\mu p_{\mu\nu}\,.$$

Let $N \geq 1$ and set $Y_n = X_{N-n}$ the time reversal of $(X_n)_{0\leq n\leq N}$. Then we have

$$\begin{aligned}\mathbb{P}(Y_0=\mu_0, Y_1=\mu_1,\ldots,Y_N=\mu_N) &= \mathbb{P}\left(X_0=\mu_N,\ldots,X_N=\mu_0\right)\\ &= \pi_{\mu_N} p_{\mu_N\mu_{N-1}}\cdots p_{\mu_1\mu_0}\\ &= \pi_{\mu_0} p_{\mu_0\mu_1}\cdots p_{\mu_{N-1}\mu_N}\,,\end{aligned}$$

where we used the detailed balance condition in the last step. By theorem 6.1.1 $(Y_n)_{0\leq n\leq N}$ is Markov $(\pi,\, P)$ for all $N \geq 1$ and, hence, $(X_n)_{n\geq 0}$ is reversible.

□

6.8. Exercises

1. Show that $\mu \leftrightarrow \nu$ defines an equivalence relation.
2. Show that an irreducible MARKOV–chain with finite state space Ω is always recurrent.
3. Show that for a finite state space every irreducible MARKOV–chain is positive recurrent. (Hint: use that in the finite case the $\lim_{n\to\infty}$ can be interchanged with sums.)

References

[1] MEJ Barkema and GT Newman. *Monte Carlo Methods in Statistical Physics*. Oxford University Press, 1999.

[2] FJ Fritz, B Huppert, and W Willems. *Stochastische Matrizen*. Springer-Verlag, 1979.

[3] WK Hastings. "Monte Carlo sampling methods using Markov chains and their applications". In: *Biometrika* 57.1 (1970), pp. 97–109. DOI: 10.1093/biomet/57.1.97. eprint: http://biomet.oxfordjournals.org/content/57/1/97.full.pdf+html. URL: http://biomet.oxfordjournals.org/content/57/1/97.abstract.

[4] N Metropolis, AW Rosenbluth, MN Rosenbluth, AH Teller, and E Teller. "Equation of state calculations by fast computing machines". In: *J. Chem. Phys.* 21 (1953), pp. 1087–1092. DOI: 10.1063/1.1699114.

[5] JR Norris. *Markov Chains*. Cambridge Books Online. Cambridge University Press, 1997. ISBN: 9780511810633. DOI: 10.1017/CBO9780511810633. URL: http://dx.doi.org/10.1017/CBO9780511810633.

[6] CP Robert and G Casella. *Introducing Monte Carlo Methods with R*. Use R! Springer, 2010. ISBN: 978-1-4419-1575-7. DOI: 10.1007/978-1-4419-1576-4.

[7] CP Robert and G Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005. ISBN: 0387212396.

7. Markov–Chain Monte-Carlo

Being backed up by the theoretical results on Markov–chains, we are now left with some questions concerning applying the Markov–chains in a simulation

1. How to *implement* a Markov–chain simulation? Remember our goal is to sample from the invariant distribution π and estimate $\mathbb{E}_\pi[Q]$ for observable quantities Q.
2. How to *monitor* convergence
   - of the distribution of states in chain to π;
   - of estimators $\hat{f}$ to $\mathbb{E}_\pi[f(X)]$;
   - to iid sampling?
3. How to *analyse* Markov–chain Monte-Carlo data (once at equilibrium) and get reliable error estimates?

We already know the answer to (3) from chapter 4. In the discussion of the remaining questions we largely follow the discussion in Ref. [6].

Concerning "convergence to **iid** ", of course we know that we are not generating **iid** samples with the chain. There will be some correlation between the states, or in the sequence of measurements $\ldots, Q(X_n), Q(X_{n+1}), \ldots$ for some observable Q, because X_n is generated from X_{n-1}. But we also know means to quantify this correlation, see chapter 4.

The amount of autocorrelation in the chain is something that can be influenced by the algorithm, i.e. by the choice of the transition matrix P. The comparison of the Metropolis algorithm and the worm cluster algorithm in the critical region gives an example, how this can work, as we will see later in this chapter.

First we define Markov–Chain Monte-Carlo (MCMC)

Definition 7.0.1: Markov Chain Monte Carlo (MCMC)

An MCMC method for the simulation of a distribution π is any method producing an irreducible and aperiodic Markov–*chain $(X_n)_{n\geq 0}$, that has π as its stationary distribution.*

There are several general classes of algorithms which implement this principle and for which the convergence properties can be investigated in a more general setup. We remark once more that an irreducible and aperiodic Markov Chain is often called *ergodic* in abuse of the term.

Algorithm 7.1.1 METROPOLIS-HASTINGS algorithm

1: generate proposal $Y_n \sim q\,(y \,|\, x_n)$

2: $X_{n+1} = \begin{cases} Y_n & \text{with probability } \rho(x_n, Y_n) \\ x_n & \text{with probability } 1 - \rho(x_n, Y_n) \end{cases}$ with

$$\rho(x,y) = \min\left\{ \frac{\pi(y)}{\pi(x)} \frac{q(x \,|\, y)}{q(y \,|\, x)}, 1 \right\} . \tag{7.1.1}$$

7.1. METROPOLIS-HASTINGS algorithm

In the following we will denote states also with x, y. We start with our target density π and additionally choose a conditional density $q = q\,(y \,|\, x)$. The transition probability is then split into two parts as already alluded to when the detailed balance condition for the METROPOLIS algorithm was discussed. The conditional density q is called the *instrumental* or proposal distribution and ρ is the METROPOLIS-HASTINGS *acceptance probability*. The algorithm is summarised in algorithm 7.1.1. The METROPOLIS-HASTINGS algorithm is very versatile concerning the target density π and the choice of instrumental density given a target density. But there is a minimal necessary condition to ensure, that all state space can be visited by the chain

$$\bigcup_{x \in \sup(f)} \sup\left(q\,(\cdot, x)\right) \supset \sup\left(f\right) . \tag{7.1.2}$$

Similarly to the case of the METROPOLIS algorithm encountered when looking at the Ising model, but more general, we now have the following result:

Theorem 7.1.1:

If $(X_n)_{n\geq 0}$ is a chain produced by the METROPOLIS-HASTINGS algorithm, then for every conditional density q fulfilling Eq. 7.1.2

(i) the transition matrix P of the chain is in detailed balance with π;

(ii) P is irreducible and aperiodic, i.e. π is the stationary distribution of the chain.

Proof:

To see this, let us look at the probability for a transition from a given state x to a subset $A \subset \Omega$ of (in general continuous) state-space. There are two possibilities for such a transition: (1) the proposed candidate state Y is in A and it is accepted or the current state x itself is in A and the proposed candidate is rejected, thus we keep $X_{n+1} = x \in A$.

$$\begin{aligned} \mathbb{P}\,(x \to A) &= \mathbb{P}\,(X_{n+1} \in A \,|\, X_n = x) \\ &= \mathbb{P}\,(Y \in A,\, X_{n+1} = Y,\, |\, X_n = x) + \mathbb{P}\,(X_n \in A,\, X_{n+1} = x \,|\, X_n = x) \\ &= \int_A q\,(y' \,|\, x)\; \rho\,(x, y')\; \mathrm{d}y' + \int_I \mathbb{1}_{\{x \in A\}}\; (1 - \rho\,(x, y'))\; q\,(y' \,|\, x)\; \mathrm{d}y' . \end{aligned} \tag{7.1.3}$$

If Eq. 7.1.3 is applied to the set $A = \{y\}$ with one element, one has

$$\mathbb{P}(x \to \{y\}) = q(y \,|\, x)\ \rho(x, y) + (1 - r(x))\ \delta_x(y) \tag{7.1.4}$$
$$r(x) = \int_I \rho(x, y)\ q(y' \,|\, x)\ \mathrm{d}y' .$$

Checking, that

$$\pi(x)\, q(y \,|\, x)\, \rho(x, y) = \pi(y)\, q(x \,|\, y)\, \rho(y, x) \tag{7.1.5}$$

one then proceeds analogously to the steps in Eq. 6.6.2. Moreover,

$$\pi(x)\ (1 - r(x))\ \delta_x(y) = \pi(y)\ (1 - r(y))\ \delta_y(x) \tag{7.1.6}$$

by the properties of the Dirac mass, which is 1, if $x = y$ and zero otherwise.

The irreducibility of the chain follows from sufficient conditions on the conditional density q, e.g. that q be positive, i.e. $q(y \,|\, x) > 0$ for all $x, y \in \mathrm{supp}(\pi)$. Note that for the Ising model using the Metropolis algorithm with single-spin-flip dynamics that was not the case, since from a given configuration, one can only reach "neighbouring" configurations, which differ by at most one spin flip. However, given one spin configuration, it is possible to reach any other configuration in maximally N steps with non-zero probability, where N is the total number of spins on the lattice.

□

Given that the chain is in detailed balance with the invariant distribution, it is also reversible. Moreover, it is aperiodic, since the the transition with $\{X_{n+1} = X_n)$ has non-zero probability.

The Metropolis-Hastings algorithm is versatile in the possible target distribution π and the choice of the conditional density q. We look at two specific forms in what follows.

Independent Metropolis-Hastings Algorithm

This special variant generalises the accept-reject method. It is a special case of the Metropolis-Hastings algorithm in the sense that the instrumental distribution q is *independent* of X_n, that is $q(\cdot, \,|\, x) = g(\cdot)$. It is formulated in algorithm 7.1.2.

Random Walk Metropolis-Hastings

This special variant aims at a *local exploration* of the neighbourhood of the current state in the chain and generates a new state while taking into account the previously simulated one: the new candidate state Y_n is proposed as $Y_n = X_n + \epsilon_n$, where now ϵ_n is considered a *random perturbation* of the current state, distributed according to the chosen instrumental density $\epsilon_n \sim g$ and *independent* of X_n. Thus $q(y \,|\, x) = g(y - x)$.

Algorithm 7.1.2 Independent Metropolis-Hastings algorithm

1: Input x_n
2: generate $y_n \sim g(y)$;
3: set x_{n+1} according to

$$x_{n+1} = \begin{cases} y_n & \text{with probability } \min\left\{\frac{\pi(y_n)}{\pi(x_n)}\frac{g(x_n)}{g(y_n)}, 1\right\}, \\ x_n & \text{otherwise}. \end{cases} \tag{7.1.7}$$

Algorithm 7.1.3 Random Walk Metropolis-Hastings algorithm

1: Input: X_n
2: generate $Y_n \sim g(|y - x_n|)$,
3: set X_{n+1} according to

$$X_{n+1} = \begin{cases} Y_n & \text{with probability } \min\left\{\frac{\pi(Y_n)}{\pi(x_n)}, 1\right\} \\ x_n & \text{otherwise}. \end{cases} \tag{7.1.8}$$

The associated Markov–chain is then a random walk in state space. Usual choices for the instrumental density g are the uniform distribution on the sphere (giving a random direction) or the normal distribution.

If g is defined *symmetric* $g(x) = g(-x)$, then this gives the original Metropolis algorithm introduced in Ref. [3] and given in algorithm 7.1.3. The norm $|\cdot|$ must be chosen suitably for the elements of state space.

Example 7.1.1

A nice example is the random walk generator for the normal distribution was given in Ref. [2]. To generate a chain with invariant distribution $\mathcal{N}(0,1)$ one fixes a proposal interval width $\delta > 0$ and given x_n

1. generate $Y_n \sim \mathcal{U}([-\delta, \delta])$;
2. accept with probability

$$\rho(x_n, Y_n) = \min\left\{\exp\left(\left(x_n^2 - Y_n^2\right)/2, 1\right\}\right. . \tag{7.1.9}$$

Example 7.1.2

Another example for an Independent Metropolis-Hastings algorithm is the *Heat bath algorithm* for the Ising model. As usual we consider a spin configuration for

the whole lattice as $s = (s_1, \ldots, s_N)$ if there are in total N spins on the lattice and the energy for the spin configuration is given by

$$E(s) = -J \sum_{\langle i,j \rangle} s_i \, s_j \,. \tag{7.1.10}$$

Then we use the algorithm:
Given $s^{(n)} = \left(s_1^{(n)}, \ldots, s_N^{(n)}, \right)$

(i) choose $s_i^{(n)}$ at random (choose $1 \leq i \leq N$ at random)

(ii) generate

$$\sigma_i \sim \exp\left(-E\left(s_1^{(n)}, \ldots, s_{i-1}^{(n)}, \sigma_i, s_{i+1}^{(n)}, \ldots, s_N^{(n)}\right)/T\right) = g(\sigma_i)\,. \tag{7.1.11}$$

When we write $g(\sigma_i)$ for the exponential in the middle, it is meant as a function of σ_i alone, while all the other $s_k^{(n)}$ with $k \neq i$ are kept fixed at this step. Physically, this has the interpretation of the spin i being coupled to a *heat bath*, which is formed by its neighbouring spins.

(iii) accept with probability

$$p = \min\left\{1, \frac{\exp\left(-E\left(s_1^{(n)}, \ldots, s_{i-1}^{(n)}, \sigma_i, s_{i+1}^{(n)}, \ldots, s_N^{(n)}\right)/T\right)}{\exp\left(-E\left(s_1^{(n)}, \ldots, s_{i-1}^{(n)}, s_i^{(n)}, s_{i+1}^{(n)}, \ldots, s_N^{(n)}\right)/T\right)} \frac{g\left(s_i^{(n)}\right)}{g\left(\sigma_i\right)}\right\} \tag{7.1.12}$$

$$= 1\,.$$

The new configuration is thus always accepted. The algorithm as written above is again based on single-spin-flip dynamics. It is irreducible, since after at most N cycles any configuration can be reached from any other with non-zero probability. Moreover, the transition kernel is in detailed balance with the distribution given by the Boltzmann weight $\pi(s) \propto \exp(-E(s)/T)$.

7.2. Optimisation and Control

It is difficult to make general statements about how fast a chain converges to equilibrium. We will thus only make some comments for the two more specialised variants of the Metropolis-Hastings algorithms (Independent and Random Walk).

The Metropolis-Hastings method allows one to produce a positive recurrent and aperiodic Markov–chain for many choices of target and instrumental densities. Despite it being a theoretically valid approach in these many cases, the convergence, which is tied to the chain's exploration of the complexity of the invariant density π, can be slow.

A potent measure of this is the *acceptance rate*.

7.2.1. Independent Metropolis-Hastings

For the Independent Metropolis-Hastings algorithm one generically wants to reach a high acceptance rate to speed up the exploration of the surface of π. We look at the special case, that the target density π is *uniformly bounded* by the instrumental density g, i.e. there is a constant M, thus that $\pi(x) \leq M\, g(x)$ for all $x \in \operatorname{supp}(\pi)$. In that case

$$||P^n(x \to \cdot) - \pi||_{TV} \leq 2\left(1 - \frac{1}{M}\right)^n . \tag{7.2.1}$$

$P(x \to \cdot)$ denotes the probability distribution conditional on x. Here $||\cdot||_{TV}$ is the *total variational norm*; for two measures it is given by

$$||\mu_1 - \mu_2||_{TV} = \sup_{A \subset \Omega} \{||\mu_1(A) - \mu_2(A)\} \tag{7.2.2}$$

and the supremum is taken over all subsets A of state space. Moreover, the expected acceptance probability is at least $1/M$.

Within this setup one can find the above mentioned bounds by

$$\begin{aligned}
\rho &= \mathbb{E}\left[\min\left\{\frac{\pi(Y_n)}{\pi(x_n)}\,\frac{g(x_n)}{g(Y_n)}\right\}\right] \\
&= \int \mathbb{1}_{\left\{\frac{\pi(y)}{\pi(x)}\,\frac{g(x)}{g(y)} > 1\right\}} \pi(x)\, g(y)\, \mathrm{d}x\, \mathrm{d}y + \int \mathbb{1}_{\left\{\frac{\pi(y)}{\pi(x)}\,\frac{g(x)}{g(y)} \leq 1\right\}} \frac{\pi(y)\, g(x)}{\pi(x)\, g(y)}\, \pi(x)\, g(y)\, \mathrm{d}x\, \mathrm{d}y \\
&= 2\int \mathbb{1}_{\left\{\frac{\pi(y)}{\pi(x)}\,\frac{g(x)}{g(y)} \geq 1\right\}} \pi(x)\, g(y)\, \mathrm{d}x\, \mathrm{d}y \\
&= 2\,\mathbb{P}\left(\frac{\pi(Y)}{g(Y)} \geq \frac{\pi(X)}{g(X)}\right) \quad [Y \sim g,\, X \sim \pi] \\
&\geq 2\int \mathbb{1}_{\left\{\frac{\pi(y)}{g(y)} \geq \frac{\pi(x)}{g(x)}\right\}} \pi(x)\, \frac{\pi(y)}{M}\, \mathrm{d}x\, \mathrm{d}y \\
&= \frac{2}{M}\,\mathbb{P}\left(\frac{\pi(X_1)}{g(X_1)} \geq \frac{\pi(X_2)}{g(X_2)}\right) \\
&= \frac{1}{M}\,,
\end{aligned} \tag{7.2.3}$$

where the last equality follows from X_1 and X_2 being independent random variables, which means in half of all cases the inequality in the penultimate line will be true. There are then two comments here.

(a) Given the similarity with the accept-reject method, for *generic optimisation* the instrumental density g should reproduce the target density f as closely as possible. This will drive M towards 1 from above and maximise ρ.

(b) One can use an *empirical approach* by introducing an additional tunable parameter θ into the instrumental density $g(\cdot) \to g(\cdot \,|\, \theta)$ and consider this parametrised instrumental density. Having chosen an initial value θ_0 one can then iteratively adjust the parameter by monitoring the *empirical acceptance rate*

$$\hat{\rho}(\theta) = \frac{2}{m} \sum_{i=1}^{m} \mathbb{1}_{\{\pi(y_i)\, g(x_i \,|\, \theta) > \pi(x_i)\, g(y_i \,|\, \theta)\}} \tag{7.2.4}$$

in a series of shorter simulation runs of length m.

7.2.2. Random Walk Metropolis-Hastings

This variant of the Metropolis-Hastings algorithm requires a different approach, since the instrumental conditional density g depends on the current state. The random walk variant is based on the *local exploration* of π. Then a high acceptance rate can signal a slow movement on the surface of π: if x_n is close to y_n, then $\pi(x_n) \approx \pi(y_n)$ and it follows that

$$\min\left\{\frac{\pi(y_n)}{\pi(x_n)}, 1\right\} \approx 1\,.$$

In extreme cases of multimodal densities (i.e. with several maxima) and deep valleys between modes, the probability to jump from one mode to another can be arbitrarily small for a high acceptance rate. If on the other hand, $\pi(y_n)$ is small compared to $\pi(x_n)$, this will result in a lower acceptance rate, but for farther jumps are possible.

There are more methods for optimisation, e.g. adaptive schemes, which use information on the chain's past for optimisation. But care needs to be taken here, since using past information *can violate the Markov property*. This kind of adaption should only be used during an initial, so-called "burn-in" phase, when no measurements are taken. i.e. before the production phase.

7.3. Slice Sampler

The Metropolis-Hastings algorithms are rather universal and put minimal conditions on the target and instrumental densities. We now turn to algorithms, which exploit the local features of the target, invariant density π and whose properties and performance are tied to this target distribution.

To start we recall the Fundamental Theorem of Simulation (cf. theorem 3.2.2): it states, that simulating $X \sim f_X$ is equivalent to simulating $(X, U) \sim \mathcal{U}(\mathcal{S}(f_X))$ uniformly on the sub-graph of f_X,

$$\mathcal{S}(f_X) = \{(x, u) \,:\, 0 \le u \le f_X(x)\}\,. \tag{7.3.1}$$

The idea is now to construct a Markov chain on $\mathcal{S}(f_X)$ with stationary distribution the uniform distribution $\mathcal{U}(\mathcal{S}(f_X))$. This is done by alternately moving along the u and x axis, respectively, as follows:

Algorithm 7.3.1 2D-Slice-Sampler

1: generate $U_{n+1} \sim \mathcal{U}([0,\ f(x_n)])$
2: generate $X_{n+1} \sim \mathcal{A}_{n+1}$ with

$$A_{n+1} = \{x\ :\ f(x) \geq u_{n+1}\}\ . \tag{7.3.2}$$

- generate $U \mid \{X = x\} \sim \mathcal{U}(\{u\ :\ u \leq f_X(x)\})$, which results in a value u'
- generate $X \mid \{U = u'\} \sim \mathcal{U}(\{x\ :\ u' \leq f_X(x)\})$, which results in a value x'

and we have the next link the chain $(x, u) \to (x', u')$. We sum this up in algorithm 7.3.1 calling it the $2d.$–slice sampler. Note, that $x_n \in \{x\ :\ u' \leq f_X(x)\}$ by construction, so $\{x\ :\ u' \leq f_X(x)\}$ is not the empty set. Moreover, f_X is required to be known only up to a multiplicative constant.

7.4. General Slice Sampler

The scheme in the previous section 7.3 can be generalised. Suppose the target density $\pi(x)$ can be decomposed into several factors according to

$$\pi(x) = \prod_{i=1}^{k} \pi_i(x)\,, \tag{7.4.1}$$

where the π_i are positive for all $1 \leq i \leq k$, but not necessarily densities.

One then introduces k auxiliary variables $\omega_1, \ldots, \omega_k$, such that π_i can be written as

$$\pi_i(x) = \int \mathbb{1}_{\{0 \leq \omega_i \leq \pi_i(x)\}}\, d\omega_i\,. \tag{7.4.2}$$

With Eq. 7.4.2 for all $1 \leq i \leq k$ the target density π becomes the *marginal density of the joint distribution* for $(X, \omega_1, \ldots, \omega_k)$

$$(X, \omega_1, \ldots, \omega_k) \sim p(x, \omega_1, \ldots, \omega_k) \propto \prod_{i=1}^{k} \mathbb{1}_{\{0 \leq \omega_i \leq \pi(x)\}}\,. \tag{7.4.3}$$

The step leading to eq. Eq. 7.4.3 is called *demarginalization* of π and is a generalisation of the random walk employed by the 2D slice sampler. We sum up with algorithm 7.4.1

Example 7.4.1

We consider the target density

$$\pi(x) = \left(1 + \sin^2(3x)\right)\left(1 + \cos^2(5x)\right)\,\mathrm{e}^{-x^2/2} \tag{7.4.6}$$

Algorithm 7.4.1 General Slice-Sampler

1: generate

$$\begin{aligned} \omega_1^{(n+1)} &\sim \mathcal{U}\left([0,\, \pi_1(x_n)]\right) \\ &\vdots \\ \omega_k^{(n+1)} &\sim \mathcal{U}\left([0,\, \pi_k(x_n)]\right) ; \end{aligned} \tag{7.4.4}$$

2: generate $X_{n+1} \sim \mathcal{U}\left(A_{n+1}\right)$, where

$$A_{n+1} \sim \left\{ y \,:\, \pi_i(y) \geq \omega_i^{(n+1)} \,\forall\, i = 1, \ldots, k \right\} . \tag{7.4.5}$$

for $x \in \mathbb{R}$. For this example the factorisation required in eq. Eq. 7.4.1 is already given and can be read off from eq. Eq. 7.4.6

$$\begin{aligned} \pi_1(x) &= \left(1 + \sin^2(3x)\right) \\ \pi_2(x) &= \left(1 + \cos^2(5x)\right) \\ \pi_3(x) &= \mathrm{e}^{-x^2/2} . \end{aligned} \tag{7.4.7}$$

Note, that as required, the π_i are positive, but not densities (not normalised) - and not even normalisable on $\mathbb{R}$ in case of π_1 and π_2.

Figure Figure 7.4.1 shows the result of an exemplary simulation run using the `R`-code discussed in section 7.8 below. The histogram shows the empirical distribution, while for comparison the blue curve gives the analytic density. The normalisation to obtain the analytic density was calculated with Mathematica.

The potential difficulty of the Slice sampler is the realisation of simulating uniformly from the set A_{n+1}. The solution in the `R`-script above is a brute force hit-and-miss. The Gibbs *Sampler* discussed in the next part overcomes this difficulty.

7.5. The Gibbs Sampler

In the preceding sections we looked at two general classes of algorithms that can produce positive recurrent and aperiodic Markov–chains: firstly, the Metropolis-Hastings algorithm. It suggests a new, candidate state using some chosen conditional distribution and then does an accept-reject step (cf. Eq. 7.1.1). Secondly, the slice sampler. In contrast to the Metropolis-Hastings algorithm, the slice sampler does not rely on an accept-reject step. Instead it uses the *true conditional distribution* to sample from the target distribution (cf. Eqs. 7.4.4, 7.4.5).

Another major class of algorithms, which are based on the principle on using the true conditional distribution, are Gibbs samplers. Following [6] we first look at the 2-stage Gibbs sampler, which can be derived from the $2d$ slice sampler.

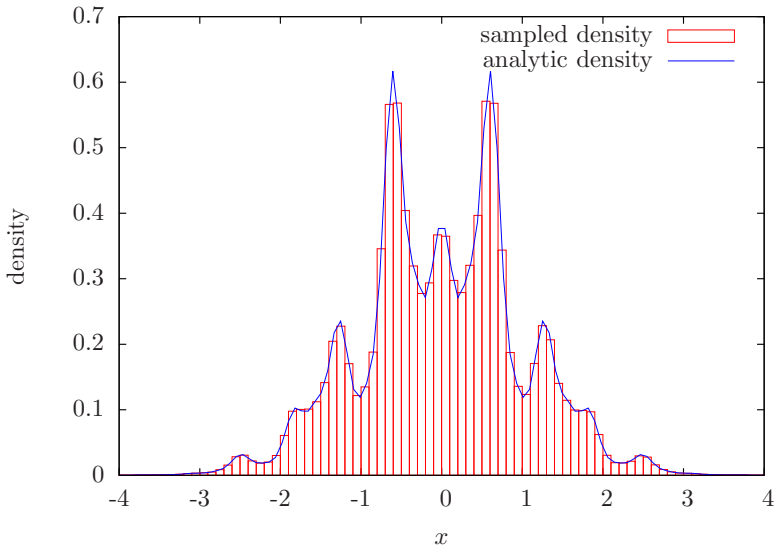


Figure 7.4.1.:
Example simulation for the 3D slice sampler for the density Eq. 7.4.6 with factorisation Eq. 7.4.7.

For the $2d$ slice sampler we had a target density $f_X(x)$. We now generalise this setup to two random variables $(x, y) \in \mathcal{X} \times \mathcal{Y}$ from the product space of $\mathcal{X}$ and $\mathcal{Y}$ with target density the *joint density* $f_{XY}(x, y)$.

From here we again continue using the *Fundamental Theorem of Simulation*: simulation $(X, Y) \sim f_{XY}$ is equivalent to simulating uniformly on the set

$$\mathcal{S}(f_{XY}) = \{(x, y, u) \,:\, 0 \leq u \leq f_{XY}(x, y)\} \,. \tag{7.5.1}$$

This 3-component setting can be naturally implemented with the random walk principle by moving uniformly in one component at a time. Thus given (x, y, u), generate

(i) X along x-axis, $X \sim \mathcal{U}(\{x \,:\, u \leq f_{XY}(x, y)\})$, which gives x';

(ii) Y along y-axis, $Y \sim \mathcal{U}(\{y \,:\, u \leq f_{XY}(x', y)\})$, which gives y';

(iii) U along u-axis, $U \sim \mathcal{U}([0,\ f_{XY}(x', y')])$, which gives u'

and we register a new state $(x, y, u) \to (x', y', u')$.

There are two crucial observations here:

1. Simulating $X \sim \mathcal{U}(\{x \,:\, u \leq f_{XY}(x, y)\})$ is equivalent to simulating

$$X \sim \mathcal{U}\left(\left\{x \,:\, f_{X\,|\,Y}(x \,|\, y) \geq u / f_Y(y)\right\}\right) \,,$$

where

$$f_{X\,|\,Y}(x \,|\, y) = \frac{f_{XY}(x, y)}{f_Y(y)} \tag{7.5.2}$$

Algorithm 7.5.1 GIBBS Sampler

1: $X_0 = x_0$;
2: **for** $n = 1, 2, \ldots$ **do**
3: generate $Y_n \sim f_{Y\,|\,X}(\cdot\,|\,x_{n-1})$;
4: generate $X_n \sim f_{X\,|\,Y}(\cdot\,|\,y_n)$.
5: **end for**

is the *conditional distribution* of X given Y and

$$f_Y(y) = \int_{\mathcal{X}} f_{XY}(x, y)\,\mathrm{d}x \tag{7.5.3}$$

is the *marginal distribution* of Y.

2. The sequence (i) → (ii)→ (iii) does not have to be in that order all the time, it can be altered while still keeping the chain stationary with the the uniform distribution on $\mathcal{S}(f_{XY})$ its stationary distribution.

Since the ordering, in which the updates along the individual axes are performed, is irrelevant, one can repeat (i) and (iii) in alternation. One can even go to the limiting case of *infinitely many* alternating repetitions of (i) and (iii). But in doing that, one is actually applying the 2D slice sampling algorithm to the pair (X, U) conditional on the given Y, which means with this procedure one simulates $X \sim f_{X\,|\,Y}(x\,|\,y)$ according to the conditional distribution of X given Y.

In the next step this procedure is repeated with the pair (Y, U) conditional on X. We again consider the limiting case of infinitely many repetitions, which means simulating $Y \sim f_{Y\,|\,X}(y\,|\,x)$ (2D slice sampling).

Thus the idea for the 2-stage GIBBS sampler is: assuming the conditional distributions $f_{X\,|\,Y}(x\,|\,y)$ and $f_{Y\,|\,X}(y\,|\,x)$ can be simulated, one implements the limiting case of the slice sampler for the pairs $(X, U)\,|\,Y$ and $(Y, U)\,|\,X$, which maintains the stationarity of the uniform distribution on $\mathcal{S}(f_{XY})$.

The auxiliary variable U is not needed in the process except for the fictitious slice sampling sequence. It can thus be removed from the algorithm. This step is referred to as "data augmentation".

The so obtained algorithm uses more knowledge on f_{XY} than the METROPOLIS-HASTINGS algorithm. This becomes clear in view of the required true conditional densities $f_{X\,|\,Y}$ and $f_{Y\,|\,X}$, which are at the heart of the updating procedure. We summarise the procedure in algorithm 7.5.1. $f_{X\,|\,Y}$ and $f_{Y\,|\,X}$ are the conditional distributions obtained from f_{XY}.

An important observation at this point is, that the individual sequences $(X_n)_{n\geq 0}$ and $(Y_n)_{n\geq 0}$ are Markov chains by themselves. For instance the transition probability for $x \to x'$ given by

$$\mathbb{P}(x \to x') = \int f_{Y\,|\,X}(y\,|\,x)\; f_{X\,|\,Y}(x\,|\,y)\;\mathrm{d}y \tag{7.5.4}$$

depends only on $X_n = x$. Moreover, the marginal distribution $f_X(x)$ is the invariant distribution for the transition kernel in Eq. 7.5.4.

$$\begin{aligned} f_X(x') &= \int f_{X\,|\,Y}(x'\,|\,y)\, f_Y(y)\,\mathrm{d}y \\ &= \int f_{X\,|\,Y}(x'\,|\,y) \left(\int f_{XY}(x,y)\,\mathrm{d}x \right) \mathrm{d}y \\ &= \int f_{X\,|\,Y}(x'\,|\,y) \int f_{Y\,|\,X}(y\,|\,x)\, f_X(x)\,\mathrm{d}x\,\mathrm{d}y \\ &= \int \mathbb{P}\left(x \to x'\right)\, f_X(x)\,\mathrm{d}x\,. \end{aligned} \tag{7.5.5}$$

Slice Sampler as special case of a Gibbs Sampler The motivating derivation of the Gibbs sampler above used the concept of the Gibbs sampler as a limiting case of two 2D slice samplers to simulate the 3-coordinate problem. On the other hand, the slice sampler can be understood as special case of a 2-stage Gibbs sampler when the joint distribution is the uniform distribution on the sub-graph $\mathcal{S}\left(f_X\right)$:

So one starts with the target distribution $f_X(x)$ and extends that to the joint distribution $f_{XU}\left(x,u\right) \propto \mathbb{1}_{\{0\le u\le f_X(x)\}}$, i.e. the uniform distribution on $\mathcal{S}\left(f\right)$ with the conditional densities

$$f_{X\,|\,U}(x\,|\,u) = \frac{\mathbb{1}_{\{0\le u\le f_X(x)\}}}{\int \mathbb{1}_{\{0\le u\le f_X(x)\}}\,\mathrm{d}x} \tag{7.5.6}$$

$$f_{X\,|\,U}(x\,|\,u) = \frac{\mathbb{1}_{\{0\le u\le f_X(x)\}}}{\int \mathbb{1}_{\{0\le u\le f_X(x)\}}\,\mathrm{d}u}\,. \tag{7.5.7}$$

From the observation above it then follows, that $\left(X_n\right)_{n\ge 0}$ is a Markov chain with stationary distribution f_X.

An important property, which enables the Gibbs sampling described above, is that the conditional distributions *have sufficient information* to sample from the joint distribution. This is similar to maximising a criterion function successively in all individual directions in a given basis, while keeping the dependence in the other directions fixed. In summary, the joint distribution f_{XY} can be derived from the conditional distributions $f_{X\,|\,Y}$ and $f_{Y\,|\,X}$. This is the statement of the following theorem:

Theorem 7.5.1: Hammersley-Clifford

The joint distribution associated with the conditional densities $f_{X\,|\,Y}$ and $f_{Y\,|\,X}$ has the density[1]

$$f_{XY}(x,y) = \frac{f_{Y\,|\,X}\left(y\,|\,x\right)}{\int f_{Y\,|\,X}(y\,|\,x)/f_{X\,|\,Y}(x\,|y)\,\mathrm{d}y}\,. \tag{7.5.8}$$

Proof:

[1] assuming the joint density exists

Algorithm 7.6.1 Multi-Stage Gibbs-Sampler

1: input: $(x_{n,1}, \ldots, x_{n,p})$
2: generate

$$\begin{aligned}
&1) \quad X_{n+1,1} && \sim f_1\left(x \,|\, x_{n,2}, \ldots, x_{n,p}\right) \\
&2) \quad X_{n+1,2} && \sim f_1\left(x \,|\, x_{n+1,1}, x_{n,3}, \ldots, x_{n,p}\right) \\
&\quad \vdots \\
&p) \quad X_{n+1,p} && \sim f_1\left(x \,|\, x_{n+1,1}, \ldots, x_{n+1,p-1}\right) ,
\end{aligned}$$

where $f_1, \ldots, f_p$ are the full conditional densities and the only densities used for the simulation.

This follows from the relations

$$\begin{aligned}
f_{XY}(x,y) &= f_{Y\,|\,X}(y\,|\,x)\, f_X(x) = f_{X\,|\,Y}(x\,|y)\, f_Y(y) \\
\int \frac{f_{Y\,|\,X}(y\,|\,x)}{f_{X\,|\,Y}(x\,|y)}\,\mathrm{d}y &= \int \frac{f_Y(y)}{f_X(x)}\,\mathrm{d}y = \frac{1}{f_X(x)} .
\end{aligned} \tag{7.5.9}$$

□

7.6. The Multi-Stage Gibbs Sampler

The multi-stage Gibbs sampler is a natural extension of the 2-stage Gibbs sampler introduced in the previous section. It is defined as follows:
For some $p > 1$ let $X = (X_1, \ldots, X_p) \in \mathcal{X}$ be a p-tuple of random variables, where each X_i can be uni- or multidimensional. Suppose one can simulate from the *univariate conditional densities* $f_1, \ldots, f_p$, i.e.

$$\begin{aligned}
X_i \,|\, &\{X_1 = x_1, \ldots, X_{i-1} = x_{i-1}, X_{i+1} = x_{i+1}, \ldots, X_p = x_p\} \\
&\sim f_i\left(x_i \,|\, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_p\right)
\end{aligned} \tag{7.6.1}$$

for all $i = 1, \ldots, p$. Then the sampler is given as in algorithm 7.6.1.

Example 7.6.1

A very nice example is the Ising model with spin configuration denoted as usual by $s = (s_1, \ldots s_N)$ for a lattice of N spins. Here we have the familiar target density

$$f(s) = \exp\left(-H \sum_{i=1}^{N} s_i - J \sum_{\langle i,j \rangle} s_i\, s_j\right) \tag{7.6.2}$$

and the conditional density f_i for each spin

$$f_i\left(s_i \,|\, s_j,\, j \neq i\right) = \frac{\exp\left(-H\, s_i - J\, s_i \sum\limits_{j\in\langle i,j\rangle} s_j\right)}{\exp\left(-H - J \sum\limits_{j\in\langle i,j\rangle} s_j\right) + \exp\left(+H + J \sum\limits_{j\in\langle i,j\rangle} s_j\right)}\,. \tag{7.6.3}$$

This multi-stage GIBBS sampler version of the simulation algorithm for the Ising model reminds of the heat bath algorithm (cf. Eqs. 7.1.11, 7.1.12), but now we do not choose the spin to be updated at random, but can use some predefined sequence, with which we go through the lattice.

Comments:

(i) for the GIBBS sampler the acceptance rate for the newly generate state is uniformly equal to unity.

(ii) using the GIBBS sampler puts limitations on the target density $f - X$. It requires prior knowledge of some analyticallity properties of the target density .

(iii) the sampling is by construction multidimensional/multivariate.

7.7. Critical Slowing Down and Collective Sampling

From a physics point of view interesting statistical problems are often connected to phase transitions. Phase transitions are typically characterised by diverging correlation lengths ξ. It should be clear from intuition that in such a situation local updates like in the sampling algorithms presented above cannot work very efficiently.

We will now try to understand how the efficiency of Monte-Carlo methods behaves in the case of $\xi \to \infty$. We remark that in any Monte-Carlo simulation the correlation length will of course not diverge, due to the finite volume, but it will become very large in the region around the phase transition point. The question is whether or not there is any relation between the correlation length of the system under investigation and the algorithm efficiency. The efficiency could be defined by the ability of the algorithm to generate independent states for a given amount of computational resources N_r. From what we discussed before, it is natural to chose the integrated autocorrelation time Eq. 4.7.8 as a measure for generating independent states. Hence, we define efficiency of an algorithm as

Definition 7.7.1: Algorithm Efficiency

$$E \overset{def}{=} \frac{C}{\tau_{\text{int}}}\,. \tag{7.7.1}$$

Here C is the computational cost per update step.

It turns out that there is a connection between the correlation length ξ and the autocorrelation, namely of the form

$$\tau_{\text{int}} \quad \propto \quad \xi^z \tag{7.7.2}$$

with some exponent $z \geq 0$ depending on the sampling algorithm in use. In a finite volume characterised through some length L this would read

$$\tau_{\text{int}} \quad \propto \quad L^{d+z}\,, \tag{7.7.3}$$

with d the dimension of the problem. And, or course, we have to take the thermodynamic limit $L \to \infty$ in order to obtain meaningful results.

The sampling algorithms discussed so far typically have values of $z \geq 2$, which comes from the fact that they are not updating the system at the physical correlation length, but only locally. This is what is called *critical slowing down*. The solution to this phenomenon is to update at typical length scales of the physical system, which is known as *collective Monte-Carlo updating*.

7.7.1. Swendsen-Wang Algorithm

As a first collective Monte-Carlo algorithm we discuss here the algorithm proposed by Swendsen and Wang [7]. We take the Ising model at vanishing external field $h = 0$ as the example, i.e. the Hamiltonian

$$\mathcal{H} \;=\; -\frac{1}{k_B T} J \sum_{\langle i,j \rangle} s_i s_j\,. \tag{7.7.4}$$

The product $s_i s_j$ can either be $+1$ or -1, so we can re-write the Hamiltonian as follows

$$\mathcal{H} \;=\; -\frac{1}{k_B T} J \left[+N_{\text{pairs}} + \sum_{\langle i,j \rangle} (s_i s_j - 1) \right]\,, \tag{7.7.5}$$

where N_{pairs} is the number of pairs with interaction. The constant shift proportional to N_{pairs} can be dropped, as it just represents a re-definition of the zero-point, and we write

$$\mathcal{H} \;=\; -\frac{2J}{k_B T} \sum_{\langle i,j \rangle} \frac{1}{2}(s_i s_j - 1)\,. \tag{7.7.6}$$

Absorbing all factors in $K = -2J/k_B T$ we write this as

$$\mathcal{H} \;=\; K \sum_{\langle i,j \rangle} \left(\delta_{s_i s_j} - 1 \right)\,. \tag{7.7.7}$$

Here, we have made use of the fact that

$$\frac{1}{2}(s_i s_j - 1) = \begin{cases} -1 & s_i \neq s_j\,, \\ 0 & s_i = s_j\,. \end{cases}$$

The partition function of the system is given as the trace over all states $S = (s_1, ..., s_N)$

$$\mathcal{Z} \;=\; \mathrm{Tr}_S\, \mathrm{e}^{\mathcal{H}(S)}\,. \tag{7.7.8}$$

Next, consider two sides l and m and remove their interaction from the HAMILTONIAN

$$\mathcal{H}_{\langle l,m\rangle} \;=\; K \sum_{\langle i,j\rangle \neq \langle l,m\rangle} (\delta_{s_i s_j} - 1)\,. \tag{7.7.9}$$

The partition function can be re-written as follows

$$\mathcal{Z} \;=\; \mathcal{Z}^{\mathrm{same}}_{\langle l,m\rangle} + e^{-K}\mathcal{Z}^{\mathrm{diff}}_{\langle l,m\rangle}\,, \tag{7.7.10}$$

if one defines

$$\begin{aligned} \mathcal{Z}^{\mathrm{same}}_{\langle l,m\rangle} &= \mathrm{Tr}_S \exp\left(\mathcal{H}_{\langle l,m\rangle}\right) \delta_{s_l s_m} \\ \mathcal{Z}^{\mathrm{diff}}_{\langle l,m\rangle} &= \mathrm{Tr}_S \exp\left(\mathcal{H}_{\langle l,m\rangle}\right) (1 - \delta_{s_l s_m})\,. \end{aligned} \tag{7.7.11}$$

The equality Eq. 7.7.10 can be understood as follows: $\mathcal{Z}^{\mathrm{same}}_{\langle l,m\rangle}$ includes everything from the partition function when $s_l = s_m$ apart from the interaction between s_l and s_m, and $\mathcal{Z}^{\mathrm{diff}}_{\langle l,m\rangle}$ so for $s_l \neq s_m$. The missing piece is the interaction between s_l and s_m, which amounts to

$$\mathrm{e}^{K(\delta_{s_l s_m} - 1)}$$

which is equal to 1 for $s_l = s_m$ and e^{-K} for $s_l \neq s_m$. By explicitly multiplying in e^{-K} to $\mathcal{Z}^{\mathrm{diff}}_{\langle l,m\rangle}$ one arrives at Eq. 7.7.10. Defining also

$$\mathcal{Z}^{\mathrm{ind}}_{\langle l,m\rangle} \;=\; \mathrm{Tr}_S\, \mathrm{e}^{\mathcal{H}_{\langle l,m\rangle}} \;=\; \mathcal{Z}^{\mathrm{same}}_{\langle l,m\rangle} + \mathcal{Z}^{\mathrm{diff}}_{\langle l,m\rangle}\,, \tag{7.7.12}$$

we can write the partition function as follows

$$\begin{aligned} \mathcal{Z} \;&=\; \mathcal{Z}^{\mathrm{same}}_{\langle l,m\rangle} + e^{-K}\mathcal{Z}^{\mathrm{diff}}_{\langle l,m\rangle} \\ &=\; \mathcal{Z}^{\mathrm{same}}_{\langle l,m\rangle} + e^{-K}\left(\mathcal{Z}^{\mathrm{ind}}_{\langle l,m\rangle} - \mathcal{Z}^{\mathrm{same}}_{\langle l,m\rangle}\right) \\ &=\; (1 - \mathrm{e}^{-K})\mathcal{Z}^{\mathrm{same}}_{\langle l,m\rangle} + \mathrm{e}^{-K}\mathcal{Z}^{\mathrm{ind}}_{\langle l,m\rangle}\,. \end{aligned} \tag{7.7.13}$$

Thus, we can re-write $\mathcal{Z}$ as follows

$$\mathcal{Z} \;=\; p\,\mathcal{Z}^{\mathrm{same}}_{\langle l,m\rangle} + (1-p)\,\mathcal{Z}^{\mathrm{ind}}_{\langle l,m\rangle}\,, \tag{7.7.14}$$

where we have set $p = 1 - \mathrm{e}^{-K}$ and note that $0 \leq p < 1$ can be interpreted as a probability. Since the first term $\mathcal{Z}^{\mathrm{same}}_{\langle l,m\rangle}$ is restricted to states where the spins s_m and s_l are equal, and the second term is not restricted in this sense, we can interpret p as the probability for a *bond* between the two sites l and m.

The very same can be repeated all interactions in the HAMILTONIAN, which leads to the SWENDSEN–WANG algorithm formulated in algorithm 7.7.1. A cluster is defined as all sites which are interconnected by bonds. And here we see why the SWENDSEN–WANG

Algorithm 7.7.1 SWENDSEN–WANG algorithm

1: Input: arbitrary spin configuration S
2: Create bonds with probability p between all interacting spins s_l, s_m with $s_l = s_m$. There are no bonds between spins with not-equal values.
3: erase the information on the spins and just keep the bond configuration with weights p and $1-p$
4: assign randomly spin values to each cluster of bonds as a whole to obtain a new spin configuration S'

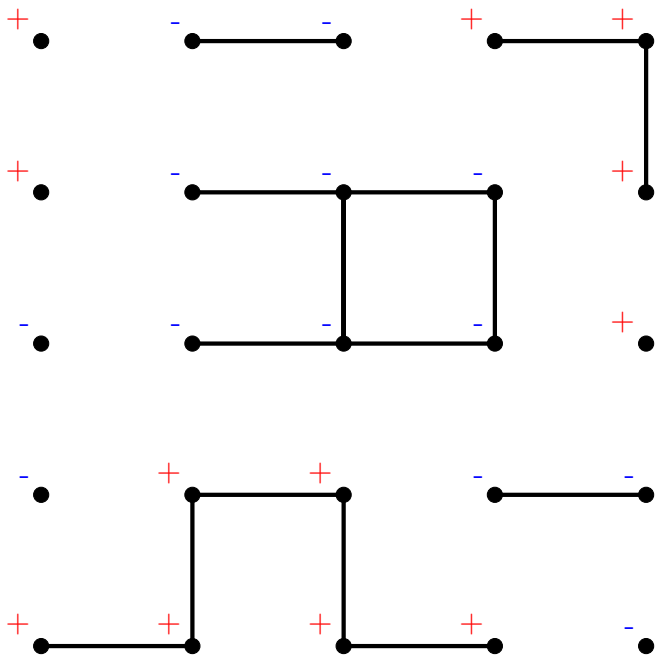


Figure 7.7.1.: Example of a bond configuration overlaid to a ISING spin configuration.

algorithm is superior to the METROPOLIS algorithm: at large values of the correlation length ξ, large clusters will form and the SWENDSEN–WANG algorithm will update them as clusters. Instead, the METROPOLIS algorithm updates spin by spin.

An example for the mapping from a spin state to a bond configuration is shown in Figure 7.7.1. The bond system is called a *bond correlation model* known from *bond percolation*. In the example of Figure 7.7.1 there are $b = 15$ bonds, $N_c = 5$ clusters and $n = 7 + 3$ interactions, which did not form a bond. Since every spin has two possible orientations, we may rewrite the partition function as

$$\mathcal{Z} \;=\; \mathrm{Tr}_{\mathrm{bonds}}\; p^b\,(1-p)^n\,2^{N_c}\,.$$

Instead of the spin system we now simulate the bond configurations. The SWENDSEN–WANG algorithm fulfils detailed balance:

Proof:

Every transition from a spin state S to another spin state S' must proceed through a

bond configuration. One observes the following

- given a bond configuration B with b bonds and N_c clusters, every spin configuration S consistent with B following the rules of the algorithm is reached with equal probability.
- going from S to S' through a particular bond configuration B with b bonds and N_c clusters happens with probability

$$p^b\,(1-p)^n\,2^{N_c} = p^b\,(\mathrm{e}^{-K})^n\,2^{N_c}\,,$$

  where p^b and N_c are independent of the two spin configurations.
- Two spin configurations compatible with a bond configuration B can only differ in n.

Therefore, the probabilities of passing through a particular bond configuration B, again with b bonds and N_c clusters differ only by the number of e^{-K} factors:

$$\frac{\mathbb{P}\,(S_i \to S_j)}{\mathbb{P}\,(S_j \to S_i)} \;=\; \frac{\mathrm{e}^{-n_i K}}{\mathrm{e}^{-n_j K}} \;=\; \exp\left(\mathcal{H}(S_i) - \mathcal{H}(S_j)\right)\,.$$

This is detailed balance for the particular bond configuration B. But, since it is fulfilled for every bond configuration B, detailed balance follows for the Swendsen–Wang algorithm.

□

7.7.2. Worm Algorithm

The Swendsen–Wang algorithm is based on cleverly rewriting the partition function. In this way the Ising model could be mapped to a bond percolation model. The worm algorithm follows a different ansatz: we re-write the partition function as follows

$$\begin{aligned}\mathcal{Z} \;&=\; \sum_S \mathrm{e}^{-\beta\mathcal{H}(S)} \;=\; \sum_S \mathrm{e}^{-\sum_{\langle i,j\rangle} K s_i s_j} \\ &= \sum_S \prod_{p=\langle i,j\rangle} \mathrm{e}^{-K s_i s_j} \;=\; \sum_S \prod_{p=\langle i,j\rangle} \sum_{N_b=0}^{\infty} \frac{K^{N_b}}{N_b!}(s_i s_j)^{N_b}\,.\end{aligned} \tag{7.7.15}$$

We can visualise the sum for a particular interaction $\langle i,j\rangle$ as follows

$\sum_{N_b=0}^{\infty} \frac{K^{N_b}}{N_b!}(s_i s_j)^{N_b} =$ 

Every line connecting the sides i and j in this pictorial representation corresponds to one factor of $K s_i s_j$. We can quickly tabularise the first few contributing factors, again for two spins s_i and s_j:

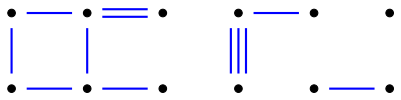


Figure 7.7.2.: One particular bond configuration for the worm algorithm.

s_i	s_j	1	Ks_is_j	$(Ks_is_j)^2$	$(Ks_is_j)^3$	$(Ks_is_j)^4$
$+$	$+$	$+1$	$+1$	$+1$	$+1$	$+1$
$+$	$-$	$+1$	-1	$+1$	-1	$+1$
$-$	$+$	$+1$	-1	$+1$	-1	$+1$
$-$	$-$	$+1$	$+1$	$+1$	$+1$	$+1$
$\sum$		4	0	$4K^2$	0	$4K^4$

It emerges that only the even N_b contribute and, hence, we can write for the two spin system

$$\mathcal{Z} \;=\; \sum_{\text{even } N_b} 2^2\,\frac{K^{N_b}}{N_b!}\,. \tag{7.7.16}$$

For the generalisation we make a guess for a sum over bond configurations $\{N_b\}$

$$\mathcal{Z} \;=\; 2^N \sum_{\{N_b\}} \prod_b \frac{K^{N_b}}{N_b!} \tag{7.7.17}$$

where b indexes the single bonds in a bond configuration $\{N_b\}$. A particular bond configuration is visualised in Figure 7.7.2.

In the two site example we had the constraint that N_b had to be even. What is the constraint we have to face in the general case? The expression Eq. 7.7.15 looks as follows

$$\sum_S \left(\sum_{N_b=0}^{\infty} \frac{K^{N_b}}{N_b!}(s_is_j)^{N_b}\right)\left(\sum_{N_b=0}^{\infty} \frac{K^{N_b}}{N_b!}(s_is_j)^{N_b}\right)\cdot\ldots\cdot\left(\sum_{N_b=0}^{\infty} \frac{K^{N_b}}{N_b!}(s_is_j)^{N_b}\right)\,.$$

Let multiply this out and look at the following terms in the (large) sum

$$\begin{aligned}\prod_b (s_{i_b}s_{j_b})^{N_b} \;&=\; (s_{i_1}s_{j_2})^{N_1}\cdot(s_{i_2}s_{j_2})^{N_2}\cdot\ldots\cdot(s_{i_B}s_{j_B})^{N_B}\\ &=\; s_1^{L_1}\cdot s_2^{L_2}\cdot\ldots\cdot s_N^{L_N}\cdot\end{aligned} \tag{7.7.18}$$

for a total of N sites and

$$L_i \;=\; \sum_{p=\langle i,j\rangle} N_b\,. \tag{7.7.19}$$

By recalling that $s_i \in \{\pm 1\}$, we will obtain a positive contribution if all L_i are even. As soon as one of the L_i is odd, however, we get zero, because

$$\sum_{s=\pm 1} s^L = \begin{cases} 1 & L \text{ even}\,,\\ 0 & L \text{ odd}\,.\end{cases} \tag{7.7.20}$$

Figure 7.7.3.:
Left: bond configuration with only closed loops. Right: a single worm configuration with two sites i_1, i_2 with an odd number of bonds connected to it.

Therefore, the constraint in the general case is that L_i must be even for all $i = 1, ..., N$. In the pictorial representation this is equivalent that an even number of bonds connected to every site i, drawn in the left panel of Figure 7.7.3. This can only be fulfilled if the bonds form only closed loops, where every bond must be used only once. Finally, we write the partition function

$$\mathcal{Z} \;=\; 2^N \sum_{\{N_b\}}^{\text{closed loops}} \prod_b \frac{K^{N_b}}{N_b!} \;\overset{\text{def}}{=}\; \sum_{\{N_b\}}^{\text{closed loops}} W_z\left(\{N_b\}\right) . \tag{7.7.21}$$

In conclusion, we can also simulate the Ising model by simulating bond configurations with the constraint of only closed loops.

The arguments presented above can be generalised for correlation functions

$$\mathcal{G}(i_1 - i_2) \;\overset{\text{def}}{=}\; \frac{1}{\mathcal{Z}} \sum_S s_{i_1} s_{i_2} \mathrm{e}^{-\beta\mathcal{H}} \;=\; \frac{g(i_1 - i_2)}{\mathcal{Z}} . \tag{7.7.22}$$

For $\mathcal{G}$ one can show [4] that the bond configurations need to be constrained to those with exactly the two sites i_1 and i_2 with an odd number of bonds connected to it. This is easy to understand, because we get exactly one additional factor $s_{i_1} s_{i_2}$ in Eq. 7.7.22 compared to Eq. 7.7.18. Therefore, L_{i_1} and L_{i_2} need to be odd to get a non-zero contribution. Such a worm is visualised in the right panel of Figure 7.7.3. One speaks about $\mathcal{Z}$–paths with all closed loops and $\mathcal{G}$–paths containing exactly one *worm* for $i_1 \neq i_2$. We call $\mathcal{P}_g$ the set of all bond configurations with zero or one worm.

Suppose we have a Monte-Carlo algorithm producing a set G of N_G bond configurations from $\mathcal{P}_g$ with proper weight W_z. Since $g(0) = \mathcal{Z}$ have identical bond elements, we can estimate both $\mathcal{Z}$ and $\mathcal{G}$ from $\mathcal{P}_g$, the correlation function from

$$g(i) \;=\; \frac{1}{N_G} \sum_G \mathbb{1}_{i=i_1-i_2}$$

and the partition function from

$$\mathcal{Z} \;=\; \frac{1}{N_G} \sum_G \mathbb{1}_{i_1=i_2} .$$

The idea is now to design an Metropolis algorithm that samples from $\mathcal{P}_g$ with the proper weight W_z. The algorithm is defined in algorithm 7.7.2 as a sequence of *move* and

Algorithm 7.7.2 Worm Algorithm

1: Input: an arbitrary bond configuration from $\mathcal{P}_g$ with i_1, i_2 given
2: **if** $i_1 = i_2$ **then**
3: *Move* i_1 and i_2 both to a point j with probability $0 < p_0 < 1$ or shift (see below) with probability $1 - p_0$.
4: **else**
5: pick one of the $2d$ neighbours j of i_2 and the connecting bond $b = \langle i_2, j\rangle$. *Shift* $i_2 \to j$ with $N_b \to N_b' = N_b \pm 1$ and accept the shift with probability

$$\mathbb{P}_{\text{acc}} \;=\; \min\left(1, \frac{K^{N_b'-N_b}}{N_b'!} N_b!\right) . \tag{7.7.23}$$

Increase or decrease in N_b is chosen with probability $1/2$.
6: **end if**

	Metropolis	Swendsen-Wang	Wolff	Worm
2d-Ising	2.167(1)	0.25(1)	0.25(1)	0.25
3d-Ising	2.02(2)	0.54(2)	0.33(1)	
4d-Ising	–	0.86(2)	0.25(1)	

Table 7.1.: Exponents z for various algorithms and the 2d- to 4d–Ising model.

shift steps. As one sees from Eq. 7.7.23, the worm algorithm is just a local Metropolis algorithm producing bonds with weight

$$W_z \;=\; \frac{K^{N_b}}{N_b!} .$$

As such, the worm algorithm fulfils detailed balance. Surprisingly, one finds for the Ising model an exponent $z = 0.25$, even though it is a fully local Metropolis algorithm. But compared to the Swendsen-Wang algorithm, the worm algorithm is much cheaper, because it does not involve the cluster search.

In Table 7.1 we have summarised the values for z for different algorithms and models. For the Wolff cluster algorithm see the exercises and Ref. [8]. For more details on the worm algorithm we refer to the original work by Prokof'ev and Svistunov Ref. [4].

7.7.3. Hybrid Monte-Carlo Algorithm

So far we have always discussed the Ising model with local interaction, i.e. only neighbouring sites interact. For such a case the Metropolis algorithm is very well suited, even though one can invent better algorithms, as we have seen. In particular, since a local change means for the Ising model case that the Hamiltonian difference can be computed cheaply, because only order $2d$ terms need to be re-computed.

Consider now a system with degrees of freedom $\phi \in \Omega$, where we keep this very general, but we require Ω to be a differentiable manifold for which we can define a *tangent* space

$T(\Omega)$. Assume we are interested in the partition function of the form

$$\mathcal{Z} \;=\; \int \mathcal{D}\phi\, \mathrm{e}^{-\mathcal{S}(\phi)}\,. \tag{7.7.24}$$

The integration is to be understood as a path integral, which on a discrete lattice takes the form

$$\int \mathcal{D}\phi \;\equiv\; \int \prod_i \mathrm{d}\phi_i \tag{7.7.25}$$

for all sites i in the lattice. But, we will assume that $\mathcal{S}(\phi)$ is *non-local* and *expensive* to evaluate. As a consequence, for every local change the whole function $\mathcal{S}$ needs to be re-computed for the proposal ϕ'. For this cases a collective updating procedure is needed, which also makes not too small steps in sample space possible at guaranteed acceptance.

We will first discuss the algorithm using a model, which still has a local interaction. But it should become clear that it circumvents the problems alluded to above. The model we are going to use is very closely related to the Ising model. It is the discretised non-linear $O(n)$ σ-model in $d = 2$ dimension for which $\mathcal{S}$ reads

$$\mathcal{S}_\sigma(\phi) \;=\; -2\beta \sum_i \sum_\mu (\phi_i \cdot \phi_{i+\hat{\mu}})\,, \qquad (\phi_i \cdot \phi_i) = 1\,, \tag{7.7.26}$$

where $\phi_i \in \mathbb{R}^n$ and $(\cdot\cdot)$ the usual scalar product. $\mathcal{S}$ is called the action of this model and $\beta = 1/g$ is the inverse coupling constant. μ encodes the directions, so in two dimensions $\mu = 0, 1$ and $\hat{\mu}$ is the corresponding unit vector in direction μ. With the constraint every ϕ_i is a vector on the $n-1$ sphere.

From now on we concentrate on $n = 4$ for which it is more convenient to write the model in terms of $SU(2, \mathbb{C})$ matrices (unitary 2×2 matrices with determinant equal 1, $SU(2)$ for short). For $U, V \in SU(2)$ one can define a scalar product as

$$(U \cdot V) \;\stackrel{\text{def}}{=}\; \frac{1}{2} \operatorname{Re} \operatorname{Tr}\left(UV^\dagger\right)\,. \tag{7.7.27}$$

The generators of $SU(2)$ are the Pauli matrices $\sigma_1, \sigma_2, \sigma_3$ with the usual properties. The Pauli matrices are a basis of the Lie algebra $su(2)$. The mapping

$$T(x) \;\stackrel{\text{def}}{=}\; x_0 \mathbb{1} + \mathrm{i} \sum_k x_k \sigma_k\,, \quad x = (x_0, x_1, x_2, x_3) \in S_3 \tag{7.7.28}$$

is a homeomorphism from the 3-sphere S_3 into $SU(2)$ and one can show that it leaves the scalar product invariant

$$(x \cdot x) \;=\; (T(x) \cdot T(x))\,. \tag{7.7.29}$$

Hence, we write the action as follows

$$\mathcal{S}_\sigma(U) \;=\; -2\beta \sum_i \sum_\mu (U_i \cdot U_{i+\hat{\mu}}) \;=\; -\beta \operatorname{Re} \left(\sum_{i,\mu} \operatorname{Tr}(U_i U_{i+\hat{\mu}}^\dagger) \right)\,. \tag{7.7.30}$$

Now, introduce new, auxiliary variables $P \in T(\Omega)$ and an artificial HAMILTONIAN

$$\mathcal{H}(P,U) \;=\; \frac{1}{2}\sum_i (P_i \cdot P_i) + \mathcal{S}(U)\,, \tag{7.7.31}$$

where we denote with $(P_i \cdot P_i)$ the scalar product on $T(\Omega)$. P can be interpreted as the canonical momenta, conjugate to U. Hence, the momenta P_i are elements of the tangent space of $SU(2)$ which is $SU(2)$ again. The artificial HAMILTONIAN $\mathcal{H}$ is invariant under the equations of motion

$$\dot{q} = \frac{\partial \mathcal{H}}{\partial P}\,, \qquad \dot{P} = -\frac{\partial \mathcal{S}}{\partial q}\,. \tag{7.7.32}$$

We define the derivative of a function $f(U)$, $U \in SU(2)$ in direction j as follows

$$D_j f(U) \;\stackrel{\text{def}}{=}\; \frac{\partial}{\partial \alpha} f(\mathrm{e}^{\mathrm{i}\alpha\sigma_j} U)_{\alpha=0}\,. \tag{7.7.33}$$

This definition is more general and can be used for $SU(N)$ as well. It can be used to show that

$$D_j(i)\mathcal{S}(U) \;=\; \frac{\beta}{2}\,\mathrm{Im}\sum_\mu \mathrm{Tr}\left[\sigma_j\left(U_i U^\dagger_{i+\hat{\mu}} - U_{i-\hat{\mu}} U^\dagger_i\right)\right]\,. \tag{7.7.34}$$

There are a few more words in order for the representation of the momenta P_i. It is most convenient to chose three real, standard normal distributed random variables p_i^k $k = 1,2,3$ for each site i. Then one generates

$$P_i = \sum_{k=1}^{3} \frac{1}{\sqrt{2}} p_i^k \sigma_k \;\in\; SU(2)\,, \tag{7.7.35}$$

for which we have

$$(P_i \cdot P_i) = \mathrm{Tr}\, P_i P_i^\dagger = \sum_k (p_i^k)^2\,. \tag{7.7.36}$$

With this we have

$$\mathcal{H}(P,U) \;=\; \sum_i \frac{1}{2}\,\mathrm{Tr}\, P_i P_i^\dagger + \mathcal{S}(U) = \frac{1}{2}\sum_{i,k}(p_i^k)^2 + \mathcal{S}(U)\,. \tag{7.7.37}$$

With the choice Eq. 7.7.35 the P_i have the property that

$$\exp\left(\mathrm{i}\alpha P_i\right) \;\in\; SU(2)\,,\; \alpha \in \mathbb{R}\,. \tag{7.7.38}$$

Unitarity follows because P_i is hermitian. And, because P_i is traceless, we also have

$$\det \exp\left(P_i\right) \;=\; \exp \mathrm{Tr}\left(P_i\right) = \exp(0) = 1\,. \tag{7.7.39}$$

Note that one can verify with elementary steps that

$$\exp(\mathrm{i}\alpha P_i) = \mathbb{1}_2 \cos(\beta) + \mathrm{i}(\hat{n}\cdot\vec{\sigma})\sin(\beta) \tag{7.7.40}$$

Algorithm 7.7.3 The Hybrid Monte-Carlo Algorithm

1: Input: arbitrary $U = \{U_i\}$
2: **for** $t = 1, 2, \ldots$ **do**
3: generate p_i^k standard normal distributed
4: integrate EOM numerically for trajectory of length τ and find proposal $U' = U(\tau)$ and momenta P' using an integration scheme which leaves the measure in phase space invariant and which is reversible
5: accept U' with probability

$$P_{\text{acc}} \stackrel{\text{def}}{=} \min\left\{1, \exp\left(\mathcal{H}(P,U) - \mathcal{H}(P',U')\right)\right\} \tag{7.7.42}$$

6: **end for**

with

$$\beta\hat{n} = \alpha P_i\,, \qquad |\hat{n}| = 1\,. \tag{7.7.41}$$

With this at hand we can define the so-called *Hybrid Monte-Carlo* algorithm [1] as follows

Definition 7.7.2: Hybrid Monte-Carlo (HMC) algorithm

1. *start with an arbitrary set of $SU(2)$ matrices $U = \{U_i\}$*
2. *generate p_i^k standard normal distributed.*
3. *integrate* Hamiltons *equations of motion Eq. 7.7.32 for a trajectory of length τ with initial conditions $p_i^k(0) = p_i^k$ and $U(0) = U$ and generate a new $U = U(\tau)$.*
4. *restart from 2.*

The integration part is also called the molecular dynamics (MD) part. In this form the algorithm is not yet very useful: the reason is that in general the equations of motion cannot be integrated analytically. For this reason one resorts to numerical integration methods. As we will see in the proof of detailed balance for the HMC, it is important to chose an integration scheme which is *symplectic* and, therefore, leaves the integration measure in phase space (P, U) invariant and which is reversible. Such a scheme is for instance defined by the *leap-frog* integration scheme.

As a consequence of numerical integration, the Hamiltonian is no longer conserved exactly, but only up to discretisation errors. For symplectic integration schemes one can show that the deviation from 0 does not depend on the trajectory length, but only on the length of the chosen time step. Symplectic integrators conserve a so-called shadow Hamiltonian exactly. However, the violation of energy conservation leads to an additional step in the HMC algorithm, namely an accept/reject step. The HMC algorithm as used in practice is summarised in algorithm 7.7.3.

The acceptance probability Eq. 7.7.42 is chosen such that the new U' is always accepted if the new HAMILTONIAN is smaller than the old one. It should be noted that the HMC algorithm defined here does not depend on the special properties of the non-linear σ–model under consideration. The HMC algorithm as formulated in algorithm 7.7.3 fulfils detailed balance:

Proof:

The transition probability from $U \to U'$ is the product of several probabilities, namely

$$\mathbb{P}\left(U \to U'\right) \;=\; \int \mathrm{d}P\, \mathrm{d}P'\; \mathbb{P}_G(P)\mathbb{P}_{\mathrm{MD}}\left([P,U] \to [P',U']\right) \mathbb{P}_{\mathrm{acc}}\left([P,U] \to [P',U']\right) . \tag{7.7.43}$$

Here, $\mathbb{P}_G$ is the GAUSSIAN distribution of the canonical momenta. The acceptance probability is given in Eq. 7.7.42. The integration scheme is reversible, hence we have for the MD transition probability

$$\mathbb{P}_{\mathrm{MD}}\left([P,U] \to [P',U']\right) \;=\; \mathbb{P}_{\mathrm{MD}}\left([-P',U'] \to [-P,U]\right) . \tag{7.7.44}$$

Moreover, for given (P,U) the integration is deterministic. Therefore, from (P,U) we will always generate identical (P',U') after trajectory length τ. But the integration is reversible and it leaves the measure invariant. For the accept/reject step we know from the METROPOLIS algorithm that detailed balance in the form

$$\mathrm{e}^{-\mathcal{H}(P,U)}\; \mathbb{P}_{\mathrm{acc}}\left([P,U] \to [P',U']\right) \;=\; \mathrm{e}^{-\mathcal{H}(P',U')}\; \mathbb{P}_{\mathrm{acc}}\left([P',U'] \to [P,U]\right) \tag{7.7.45}$$

holds. Hence, we have

$$\begin{aligned} \mathrm{e}^{-\mathcal{S}(U)}\, \mathbb{P}\left(U \to U'\right) \;=\; \mathcal{N} \int \mathrm{d}P\, \mathrm{d}P'\; \mathrm{e}^{-\mathcal{H}(P,U)} \mathbb{P}_{\mathrm{MD}}\left([P,U] \to [P',U']\right) \\ \times\, \mathbb{P}_{\mathrm{acc}}\left([P,U] \to [P',U']\right) . \end{aligned} \tag{7.7.46}$$

with $\mathcal{N}$ the normalisation constant from $\mathbb{P}_G$. In this step we have used that

$$\mathrm{e}^{-\mathcal{S}(U)} \mathbb{P}_G(P) \;=\; \mathcal{N} \mathrm{e}^{-\mathcal{H}(P,U)} . \tag{7.7.47}$$

Using Eq. 7.7.45 and $\mathcal{H}(P,U) = \mathcal{H}(-P,U)$ one obtains

$$\begin{aligned} \mathrm{e}^{-\mathcal{S}(U)}\, \mathbb{P}\left(U \to U'\right) \;=\; \mathcal{N} \int \mathrm{d}P\, \mathrm{d}P'\; \mathrm{e}^{-\mathcal{H}(-P',U')} \mathbb{P}_{\mathrm{MD}}\left([P,U] \to [P',U']\right) \\ \times\, \mathbb{P}_{\mathrm{acc}}\left([-P',U'] \to [-P,U]\right) . \end{aligned} \tag{7.7.48}$$

Using reversibility, one obtains

$$\begin{aligned} \mathrm{e}^{-\mathcal{S}(U)}\, \mathbb{P}\left(U \to U'\right) \;=\; \mathcal{N} \int \mathrm{d}P\, \mathrm{d}P'\; \mathrm{e}^{-\mathcal{H}(-P',U')} \mathbb{P}_{\mathrm{MD}}\left([-P',U'] \to [-P,U]\right) \\ \times\, \mathbb{P}_{\mathrm{acc}}\left([-P',U'] \to [-P,U]\right) . \end{aligned} \tag{7.7.49}$$

Finally, one performs a change of variables $-P \to P$ and $-P' \to P'$ to arrive at

$$\mathrm{e}^{-\mathcal{S}(U)}\, \mathbb{P}\left(U \to U'\right) \;= \mathrm{e}^{-\mathcal{S}(U')}\, \mathbb{P}\left(U' \to U\right) , \tag{7.7.50}$$

which is possible because the measure is left invariant by the integration scheme.

□

We close the discussion of the HMC algorithm by explicitly defining the *leap-frog* algorithm for the $SU(2)$ case:

1. perform a *half*-step in the momenta according to

$$P_i(\delta\tau/2) \;=\; P_i(0) - \frac{\delta\tau}{2}\vec{D}_i\mathcal{S}(U(0))\vec{\sigma}\,. \tag{7.7.51}$$

2. iterate for $k = 1, ..., n-1$

$$\begin{aligned} U_i(k\delta\tau) \;&=\; \exp\left[\mathrm{i}\delta\tau P_i((k-1/2)\delta\tau)\right] U_i((k-1)\delta\tau)\,, \\ P_i(k\delta\tau+\delta\tau/2) \;&=\; P_i(0) - \frac{\delta\tau}{2}\vec{D}_i\mathcal{S}(U(k\delta\tau))\vec{\sigma}\,. \end{aligned} \tag{7.7.52}$$

3. perform a last half step in the momenta and one full step in the U fields

$$\begin{aligned} U_i(n\delta\tau) \;&=\; \exp\left[\mathrm{i}\delta\tau P_i((n-1/2)\delta\tau)\right] U_i((n-1)\delta\tau)\,, \\ P_i(n\delta\tau) \;&=\; P_i((n-1/2)\delta\tau) - \frac{\delta\tau}{2}\vec{D}_i\mathcal{S}(U(n\delta\tau))\vec{\sigma}\,. \end{aligned} \tag{7.7.53}$$

Here, we have used the notation

$$\vec{D}_i f(U)\vec{\sigma} \;\stackrel{\text{def}}{=}\; \sum_{j=1}^{3} D_j(i) f(U)\sigma_j\,.$$

For the evaluation of the exponential function it is sufficient to use a fixed but finite number of terms in the series representation.

7.8. Markov Chain Monte-Carlo with R

We first write a simple Metropolis algorithm for the Ising model. For this we write an initialisation function that creates a $2d$ spin-lattice and fills it randomly with -1 and $+1$

```
N <- 20
IsingInit <- function(N) {
  S <- array(runif(N^2), dim=c(N,N))
  S[which(S>=0.5)] <- 1
  S[which(S<0.5)] <- -1
  return(invisible(S))
}
beta <- 0.1  ## large temperature
```

Next, we need a function to compute the difference in the HAMILTONIAN if one spin is flipped at site i, hence, $s_i \to -s_i$. The corresponding formula for the energy difference of the new and the old energy value reads

$$\begin{aligned}\Delta\mathcal{H} &= \mathcal{H}(...,-s_i,...) - \mathcal{H}(...,s_i,...) \\ &= -J\sum_{j=i\pm\hat{\mu}}(-s_i s_j - s_i s_j) = 2Js_i\sum_{j=i\pm\hat{\mu}} s_j \\ &= 2J\, s_i \sum_{\mu}(s_{i+\hat{\mu}} + s_{i-\hat{\mu}})\,.\end{aligned} \tag{7.8.1}$$

This can be conveniently programmed as follows

```
deltaH <- function(x, S, J=1) {
  N <- dim(S)[1]
  z <- x-1    ### in C index style its easier with modulus
  y <- t(array((z+c(+c(1,0), -c(1,0), +c(0,1), -c(0,1)))
               %% (N), dim=c(2,4)) + 1)
  return(2*J*S[x[1], x[2]]*sum((S[y])) )
}
```

We implement periodic boundary conditions using the modulo operation. For this purpose we first change to C–index style by subtracting 1 and add the 1 again in the end. The METROPOLIS update can then be written in a few lines of code. First we implement a socalled *sweep* through all sites of the $N \times N$ grid

```
IsingSweep <- function(S, J=1, beta=1, N) {
  N <- dim(S)[1]
  AR <- 0
  for(x in c(1:N)) {
    for(y in c(1:N)) {
      dH <- deltaH(x=c(x,y), S=S, J=J)
      Accept <- (runif(n=1) < exp(-beta*dH))
      if(Accept) {
        S[x,y] <- -S[x,y]  ## flip at (x,y)
        AR <- AR+1
      }
    }
  }
  return(invisible(list(S=S, AcceptanceRate=AR/N^2)))
}
```

To visualise the results its convenient to also have corresponding plot function available

```
IsingPlot <- function(S) {
  cc=c("red", "yellow")
 heatmap(x=S, Rowv = NA, Colv = NA, scale="none", col=cc, xlab="", ylab="",
          labRow=NA, labCol=NA, margins=c(0.5,0.5))
}
```

Moreover, we would like to measure for instance the magentisation on each spin configuration. The following function measures the magnetisation naïvly

```
IsingMag <- function(S) {
  return(sum(S)/dim(S)[1]^2)
}
```

A certain number of sweeps can be run as follows

```
M <- numeric()
S <- IsingInit(N=N)
for(i in c(1:100)) {
  S <- IsingSweep(S=S, beta=beta, N=N)$S
  M[i] <- IsingMag(S)
}
IsingPlot(S)
```

where the example plot is shown in Figure 7.8.1 using the colour *red* and *yellow* for positive, spin up or negative spin down.

7.8.1. Slice Sampler

Consider example 7.4.1. We define the target density and its factorisation as follows

```
ftarget <- function(x) {
  return( (1 + sin(3*x)^2) * (1 + cos(5*x)^4) * exp(-x^2/2.) )
}

f1 <- function(x) return( 1 + sin(3*x)^2)
f2 <- function(x) return(1 + cos(5*x)^4)
f3 <- function(x) return(exp(-x^2/2.))
```

and a function for the 3d-slice sampler

```
SliceSampler3d <- function(N = 10^5, x0 = 0,
    kmax = 10000) {
```

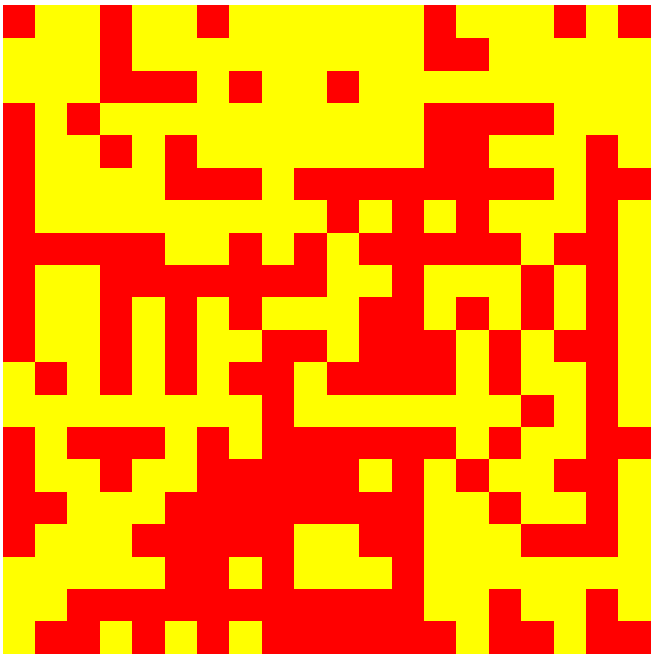


Figure 7.8.1.: Example Ising model spin configuration.

```
  s <- numeric(N)
  x <- x0
  for(t in 1:N) {
    u <- runif(3)
    w1 <- u[1] * f1(x)
    w2 <- u[2] * f2(x)
    w3 <- u[3] * f3(x)

    xlim <- sqrt( -2*log(w3))
    y <- (2 * runif(1) - 1 ) * xlim
    k <- 0
    if( ( (w1 > 1) || (w2 > 1) ) && k < kmax) {
      while( (f1(y) < w1) || (f2(y) < w2) ) {
        y <- (2 * runif(1) - 1 ) * xlim
        k <- k + 1
      }
      if(k >= kmax) stop("reached kmax")
    }
    s[t] <- y
    x <- y
  }
  return(invisible(s=s))
}
```

7.8.2. Gibbs Sampler and Bayesian Learning

As an example for the Gibbs sampler we will discuss here Bayesian learning. We recall the theorem of Bayes 2.1.2 from chapter 2. It states

$$\mathbb{P}(x|y) \;=\; \frac{\mathbb{P}(y|x)\,\mathbb{P}(x)}{\mathbb{P}(y)}\,.$$

We have discussed already in section 4.4.3 the application of this theorem for parameter estimation incorporating prior knowledge. Here, we will apply it to sampling from the posterior probability using a Gibbs sampler. For this purpose consider the so-called Poisson disruption problem: Let $X_1, \ldots, X_n$ be random variables counting the number of disasters per year in n consecutive years. Suppose the rate of disasters changes in year K from rate λ_1 to rate λ_2. K is called a *change point*. We assume that we have prior knowledge for the rates λ_1 and λ_2, namely

$$\lambda_i \;\sim\; \text{Gamma}\,(a_i, b_i)$$

with shape parameters a_i and rate b_i, see Eq. 2.7.4. We assume the parameters a_i and b_i to be known. We encounter the following hirarchical structure

1. K has some discrete pdf on $1, \ldots n$. Here, we will assume that it is uniformly distributed, so any year might be the change point with equal probability.
2. Given K, the λ_i are independent and from a Gamma distribution.
3. Given K, λ_1 and λ_2, the X_i are independent and have a Poi(λ_1) distribution for years $i = 1, \ldots, K$ and a Poi(λ_2) distribution for years $i = K+1, \ldots, n$.

One also calls this a hirarchical model. Using BAYES theorem we can write BAYES:

$$f(\lambda_1, \lambda_2, K|x) \;\propto\; f(x|\lambda_1, \lambda_2, K)\, f(K)\, f(\lambda_1)\, f(\lambda_2)\,. \tag{7.8.2}$$

First, we need to find the likelihood $f(x|\lambda_1, \lambda_2, K)$. It is given as follows

$$\begin{aligned} f(x|\lambda_1, \lambda_2, K) \;&=\; \prod_{i=1}^{K} \frac{\lambda_1^{x_i}}{x_i!} \mathrm{e}^{-\lambda_1} \;\; \prod_{i=K+1}^{n} \frac{\lambda_2^{x_i}}{x_i!} \mathrm{e}^{-\lambda_2} \\ &= e^{-K\lambda_1} e^{-(n-K)\lambda_2} \lambda_1^{\sum_{i=1}^{K} x_i} \lambda_2^{\sum_{i=K+1}^{n} x_i} \prod_{i=1}^{n} \frac{1}{x_i!}\,, \end{aligned} \tag{7.8.3}$$

because every of the X_i follows a POISSON distribution. With this we obtain for Eq. 7.8.2

$$\begin{aligned} f(\lambda_1, \lambda_2, K|x) \;\propto\; &\prod_{i=1}^{K} \frac{\lambda_1^{x_i}}{x_i!} \mathrm{e}^{-\lambda_1} \;\; \prod_{i=K+1}^{n} \frac{\lambda_2^{x_i}}{x_i!} \mathrm{e}^{-\lambda_2} \times \\ &\times f(K) \;\; \mathrm{Gamma}(a_1, b_1) \;\; \mathrm{Gamma}(a_2, b_2)\,. \end{aligned} \tag{7.8.4}$$

From this expression we can read off the various conditional posterior distributions needed for GIBBS sampling. Recall, we assume that

$$f(K) \;=\; \mathcal{U}_{[1,2,3,\ldots,n]}\,.$$

For instance, the distribution for λ_1 with all other parameters fixed is given as

$$f(\lambda_1|\lambda_2, K, x) \;\propto\; \mathrm{Gamma}(a_1 + \sum_{i=1}^{K} x_i, b_1 + K)\,, \tag{7.8.5}$$

which can be derived by simple algebraical rearrangements of Eq. 7.8.4. In the same way one finds

$$f(\lambda_2|\lambda_1, K, x) \;\propto\; \mathrm{Gamma}(a_2 + \sum_{i=K+1}^{n} x_i, b_2 + n - K)\,. \tag{7.8.6}$$

It is well known that for the combination of Poisson and Gamma distributions, the conditional distributions are again Gamma distributions. The last conditional posterior

distribution we need to find is $f(K|\lambda_1, \lambda_2, x)$. Arranging all terms depending on K leads to

$$f(K|\lambda_1, \lambda_2, x) \propto e^{-K(\lambda_1-\lambda_2)} \lambda_1^{\sum_{i=1}^{K} x_i} \lambda_2^{\sum_{i=K+1}^{n} x_i} \tag{7.8.7}$$

We can still rewrite this by noticing that we only need the conditional distribution up to constant factors. Hence, we may rewrite

$$\lambda_2^{\sum_{i=K+1}^{n} x_i} = \lambda_2^{\sum_{i=1}^{n} x_i - \sum_{i=1}^{K} x_i}$$

to obtain

$$\begin{aligned} f(K|\lambda_1, \lambda_2, x) &\propto e^{-K(\lambda_1-\lambda_2)} \lambda_1^{\sum_{i=1}^{K} x_i} \lambda_2^{-\sum_{i=1}^{K} x_i} \\ &= e^{-K(\lambda_1-\lambda_2)} \left(\frac{\lambda_1}{\lambda_2}\right)^{\sum_{i=1}^{K} x_i} . \end{aligned} \tag{7.8.8}$$

This distribution is not known in closed form. However, we can still simulate from it by computing $f(K|\lambda_1, \lambda_2, x)$ for all K. This represents the probability for each possible value of K with the other parameters fixed. From this distribution we can draw using a multinomial distribution.

In a practical implementation we first fix the external parameters

```
n <- 50
a1 <- 1; a2 <- 1
b1 <- 2; b2 <- 1
K <- 21
lambda1 <- 1; lambda2 <- 3
niter <- 10000
```

Next, we generate sample data from a Poisson distribution

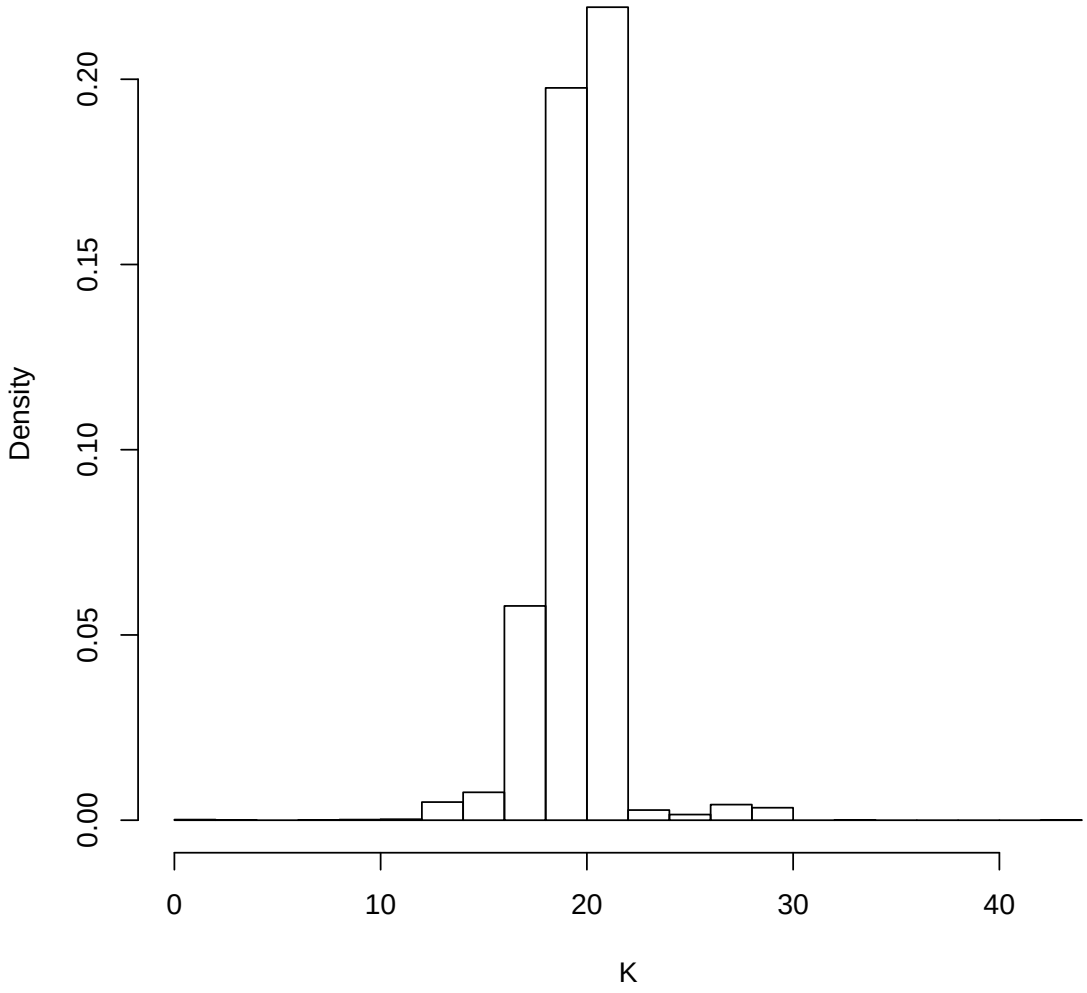
```
set.seed(1234567)
x <- rep(NA, times=n)
x[1:K] <- rpois(n=K, lambda=lambda1)
x[(K+1):n] <- rpois(n=n-K, lambda=lambda2)
```

In the following step we allocate memory for the Markov chain and find inital values for all three parameters by drawing from the corresponding distributions

```
chain <- array(NA, dim=c(niter,3))
## initial values
chain[1,1] <- sample.int(n=n, size=1)
chain[1,2] <- rgamma(n=1, shape=a1, rate=b1)
chain[1,3] <- rgamma(n=1, shape=a2, rate=b2)
```

Now we can run the Gibbs sampler as follows

```
for(i in c(2:niter)) {
  chain[i, 2] <- rgamma(n=1, shape=a1+sum(x[1:chain[i-1,1]]),
                        rate=b1+chain[i-1,1])
  chain[i, 3] <- rgamma(n=1, shape=a2+sum(x[(chain[i-1,1]+1):n]),
                        rate=b2+n-chain[i-1,1])
  probs <- rep(NA, times=n)
  s <- 0
  for(r in c(1:n)) {
    s <- s + x[r]
    probs[r] <- -r*chain[i, 2] + r*chain[i, 3] +
      (log(chain[i, 2])-log(chain[i, 3]))*s
  }
  probs <- exp(probs - max(probs))
  chain[i, 1] <- which(rmultinom(n=1, size=1, prob=probs)
                       == 1)
}
rr <- c(500:niter)
hist(chain[rr, 1], main="", freq=FALSE, xlab=c("K"))
```



The prediction of the GIBBS sampler for K given the data is $\hat{K} = 20.2597621$, compared to the input value $K = 21$. The standard error reads $\Delta = 0.0207363$. Here, we have dropped the first 500 measurements for equilibration and neglected any autocorrelation in the data.

Note that we used the log–likelihood to compute the conditional posterior distribution for K and exponentiated then. Note also that the outcome of this simulation depends strongly on the choice of the parameters a_i and b_i. Here we assumed that they are known and the GIBBS sampler works well for actual values of λ_1 and λ_2 close the modes of the corresponding gamma distributions. The performance is naturally much poorer if the actual values of λ_1 and λ_2 are located in the tails of the gamma distributions.

7.8.3. Hybrid Monte Carlo and Maximum Likelihood

We have discussed χ^2 fits at length in previous chapters. Here, we will shed a different light on it by use the HMC algorithm to sample the maximum likelihood distribution including prior knowledge for some of the parameters. We use the following data that we already used in section 4.8.

```
kable(fdata)
```

	t	x	err
V1	1	-0.133250	0.1037669
V2	2	1.995190	0.0900265
V3	3	3.983042	0.1004825

The single data points have errors which are normally distributed. The model is linear plus intercept, i.e.

$$F(t) \;=\; at + b\,,$$

and we will assume that we have prior knowledge for the intercept, namely $\bar{b} = -1.983(12)$. Again, the error $\delta\bar{b} = 0.012$ is assumed to be normally distributed. Using Bayes theorem reads here

$$f(a,b|x) \;\propto\; f(x|a,b)f(a)f(b)\,. \tag{7.8.9}$$

From the prior knowledge we know that

$$f(b) \;=\; \mathcal{N}_{\bar{b},\Delta\bar{b}} \;\propto \mathrm{e}^{-(\bar{b}-b)^2/\Delta\bar{b}}\,.$$

For the slope a we do not have prior knowledge, so we set $f(b)$ to the uniform distribution over $\mathbb{R}$. With this we can write

$$\begin{aligned} f(x|a,b)f(a)f(b) \;&\propto\; \exp\left(-\sum_i \frac{(x_i - F(t_i))^2}{2\Delta x_i^2}\right)\; \exp\left(\frac{-(\bar{b}-b)^2}{2\cdot\Delta\bar{b}^2}\right) \\ &\propto \exp\left(-\frac{1}{2}\left[\sum_i \frac{(x_i - F(t_i))^2}{\Delta x_i^2} + \frac{(\bar{b}-b)^2}{\Delta\bar{b}^2}\right]\right)\,, \end{aligned} \tag{7.8.10}$$

which is a function of the parameters a and b, which we want to determine. Now, lets use the HMC. For this we need to define the artificial Hamiltonian

$$\begin{aligned} \mathcal{H}(p_a,p_b,a,b) \;&=\; \frac{p_a^2}{2} + \frac{p_b^2}{2} - \log\left[\, f(x|a,b)f(a)f(b)\,\right] \\ &=\; \frac{p_a^2}{2} + \frac{p_b^2}{2} + \frac{1}{2}\left[\sum_i \frac{(x_i - F(t_i))^2}{\Delta x_i^2} + \frac{(\bar{b}-b)^2}{\Delta\bar{b}^2}\right]\,. \end{aligned} \tag{7.8.11}$$

Hamiltonians equations of motion read for this Hamiltonian

$$\begin{aligned} \dot{a} \;&=\; p_a\,, \qquad & \dot{p}_a \;&=\; -\frac{\partial\mathcal{H}}{\partial a}\,, \\ \dot{b} \;&=\; p_b\,, \qquad & \dot{p}_b \;&=\; -\frac{\partial\mathcal{H}}{\partial b}\,. \end{aligned} \tag{7.8.12}$$

The partial derivatives of $\mathcal{H}$ with respect to a and b can be computed easily. The derivative can be implemented as the following function

```
derivH <- function(p, par, data, pb, dpb) {
  c(sum(-(data$x - par[1]*data$t - par[2])*data$t/data$err^2),
    sum(-(data$x - par[1]*data$t - par[2])/data$err^2)
    - (pb - par[2])/dpb^2)
}
```

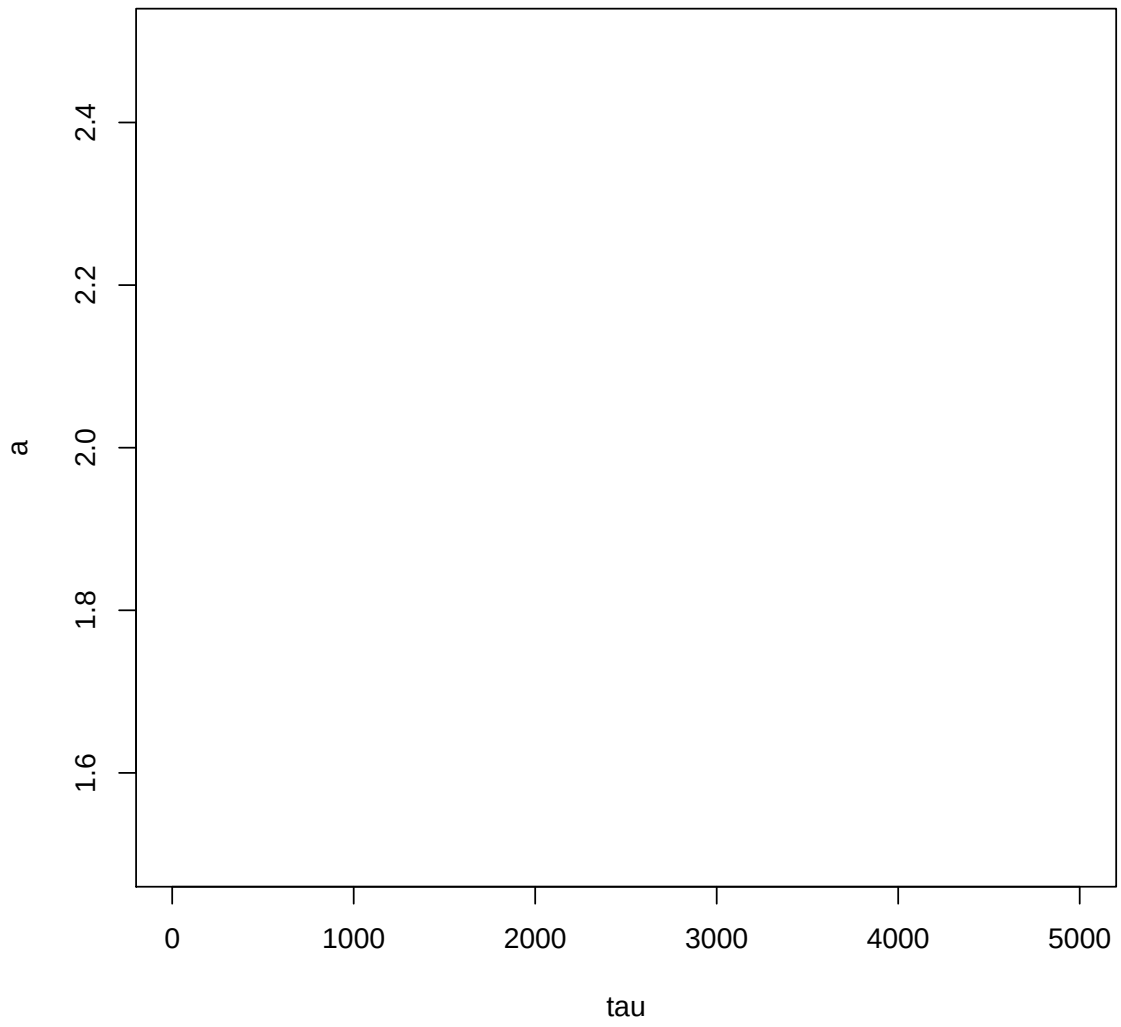
The MD evolution using the leapfrog integration scheme reads

```
md.integ <- function(tau=1, nsteps=10, p, par, data, pb, dpb) {
  dtau <- tau/nsteps
  ## half step in momenta
  tmp <- derivH(p=p, par=par, data=data, pb=pb, dpb=dpb)
  p <- p - dtau/2*tmp
  for(i in c(1:(nsteps-1))) {
    par <- par + dtau*p
    tmp <- derivH(p=p, par=par, data=data, pb=pb, dpb=dpb)
    p <- p - dtau*tmp
  }
  par <- par + dtau*p
  ## final half step in momenta
  tmp <- derivH(p=p, par=par, data=data, pb=pb, dpb=dpb)
  p <- p - dtau/2*tmp
  return(list(par=par, p=p))
}
```

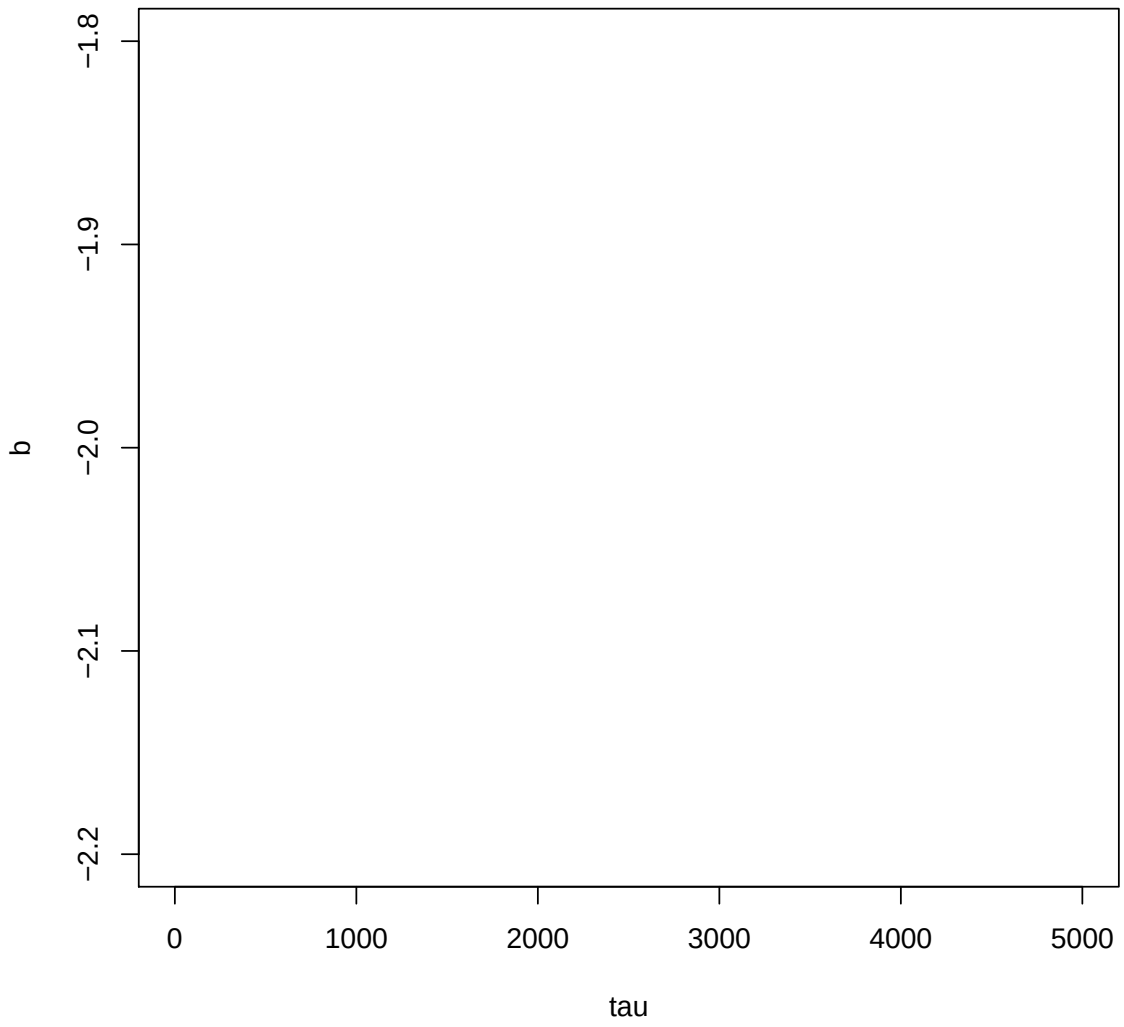
And the still messy HMC implementation for this problem looks like

```
N <- 5000
## initial values
a <- c(0)
b <- c(1.9)
dh <- c(0)
edh <- c(0)
## prior values for b
pb <- -1.983
dpb <- 0.012
## Acceptance cumulated
CumAcc <- 0
for(i in c(1:N)) {
  par <- c(a[i], b[i])
  p <- rnorm(n=2, mean=0, sd=1)
```

```
  Hold <- 0.5*sum(p^2/2)
    + 0.5*sum((fdata$x - par[1]*data$t - par[2])^2/fdata$err^2)
    + 0.5*(par[2]-pb)^2/dpb^2
  res <- md.integ(tau=1, nsteps=80, p=p, par=par,
                  data=fdata, pb=pb, dpb=dpb)
  Hnew <- 0.5*sum(res$p^2/2)
    + 0.5*sum((fdata$x - res$par[1]*data$t - res$par[2])^2/fdata$err^2)
    + 0.5*(res$par[2]-pb)^2/dpb^2
  deltaH <- Hnew - Hold
  r <- runif(n=1)
  dh[i] <- deltaH
  edh[i] <- exp(-deltaH)
  accept <- (r < edh[i])
  if(accept) {
    a[i+1] <- res$par[1]
    b[i+1] <- res$par[2]
    CumAcc <- CumAcc + 1
  }
  else {
    a[i+1] <- a[i]
    b[i+1] <- b[i]
  }
}
plot(a, xlab="tau", ylab="a", main="", t="l", ylim=c(1.5, 2.5))
```




```
plot(b, xlab="tau", ylab="b", main="", t="l", ylim=c(-2.2,-1.8))
```



And it gives $b = 1.9$ with a naïve error of $\Delta b = 0$ and $a = 0$ with a naïve error of $\Delta a = 0$, where we skipped the first 200 trajectories for equilibration. We did not take autocorrelation into account here, so real errors are certainly a bit larger.

We also find $\langle \exp(-\Delta\mathcal{H}) \rangle = 0$ with error $\Delta \exp(-\Delta\mathcal{H}) = 0$. The acceptace rate was 0%.

References

[1] S Duane, AD Kennedy, BJ Pendleton, and D Roweth. “Hybrid Monte Carlo”. In: *Phys. Lett.* B195 (1987), pp. 216–222. DOI: 10.1016/0370-2693(87)91197-X.

[2] WK Hastings. “Monte Carlo sampling methods using Markov chains and their applications”. In: *Biometrika* 57.1 (1970), pp. 97–109. DOI: 10.1093/biomet/57.1.97. eprint: http://biomet.oxfordjournals.org/content/57/1/97.full.pdf+html. URL: http://biomet.oxfordjournals.org/content/57/1/97.abstract.

[3] N Metropolis, AW Rosenbluth, MN Rosenbluth, AH Teller, and E Teller. “Equation of state calculations by fast computing machines”. In: *J. Chem. Phys.* 21 (1953), pp. 1087–1092. DOI: 10.1063/1.1699114.

[4] N Prokof’ev and B Svistunov. “Worm Algorithms for Classical Statistical Models”. In: *Phys. Rev. Lett.* 87 (16 Sept. 2001), p. 160601. DOI: 10.1103/PhysRevLett.87.160601. URL: http://link.aps.org/doi/10.1103/PhysRevLett.87.160601.

[5] CP Robert and G Casella. *Introducing Monte Carlo Methods with R*. Use R! Springer, 2010. ISBN: 978-1-4419-1575-7. DOI: 10.1007/978-1-4419-1576-4.

[6] CP Robert and G Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005. ISBN: 0387212396.

[7] RH Swendsen and JS Wang. “Nonuniversal critical dynamics in Monte Carlo simulations”. In: *Phys. Rev. Lett.* 58 (2 Jan. 1987), pp. 86–88. DOI: 10.1103/PhysRevLett.58.86. URL: http://link.aps.org/doi/10.1103/PhysRevLett.58.86.

[8] U Wolff. “Collective Monte Carlo Updating for Spin Systems”. In: *Phys. Rev. Lett.* 62 (1989), p. 361. DOI: 10.1103/PhysRevLett.62.361.

[9] U Wolff. “Simulating the All-Order Strong Coupling Expansion I: Ising Model Demo”. In: *Nucl. Phys.* B810 (2009), pp. 491–502. DOI: 10.1016/j.nuclphysb.2008.09.033. arXiv: 0808.3934 [hep-lat].

A. A very short Introduction to R

A.1. Data Analysis with R

There are plenty of introductions to R available online or in form of books. You can find an incomplete collection in the bibliography. There is a large collection of addon packages available for R, which can be found on the CRAN [3] archive. Packages can be installed using `install.packages` from the R command line, or via the systems package manager.

The intention of this introduction is to provide you with the abilities to follow the R–examples in this lecture and to solve the exercises with R. It goes without saying that the same can be achived with any other programming language more or less easily.

After R is started, you may type commands at the prompt. Most importantly, how to quit R again? Well, R is based on functions and the function to quit is `q()`. Help is provided for any core R function `f` by typing `?f`. Topics can be searched using `??topic`. R can be used as a calculator, so you may type things like

```
3+5

## [1] 8

exp(17)

## [1] 24154953

3^32

## [1] 1.85302e+15

pi

## [1] 3.141593

3 < 1

## [1] FALSE
```

However, R can, of course, do much more, as an object oriented programming language. First, R is very good in handling vectors: you can create one by typing

```
x <- c(1,2,3)
```

Note the non-standard assignment symbol `<-`, which however, can also be replaced by a `=`. You can access elements of `x` with indices and index vectors

```
x[2]

## [1] 2

x[c(1,3)]

## [1] 1 3
```

We remark here that `R` starts counting indices at `1`. Index-vectors can be quite useful. You can sum the elements of `x`, or square or multiply the whole vector

```
sum(x)

## [1] 6

x^2

## [1] 1 4 9

3*x

## [1] 3 6 9
```

The length of a vector can be accessed using

```
length(x)

## [1] 3
```

You can also access a certain class of elements of your vector, e.g. with

```
which(x > 1)

## [1] 2 3

ii <- which(x>2)
x[ii]

## [1] 3

x[x<3]

## [1] 1 2
```

There are also special functions to create sequences:

```
seq(from=1, to=10, by=3)

## [1]  1  4  7 10

rep(0, times=5)

## [1] 0 0 0 0 0
```

where you also seen the ability of R to use named function parameters. In case the parameters are given named, the order is not important. If they are not named, R tries to use order and similarity of names to match parameters. Matrices are created as follows

```
M <- matrix(data=c(0:3), nrow=2, ncol=2)
M

##      [,1] [,2]
## [1,]    0    2
## [2,]    1    3
```

and its elements or rows and columns are accessed like

```
M[2,1]

## [1] 1

M[1,]

## [1] 0 2
```

An important remark here is that R follows the FORTRAN convention for index ordering, so the first index is fastest.

R allows you to define functions. In its simplest version it looks like this

```
add.vector <- function(x,y) { return(x+y)}
add.vector(x,x)

## [1] 2 4 6
```

Note that this function now works for vectors, but also for scalars, which are treated in R as a vector of length 1.

A.1.1. Conditionals and Loops

Of course, R also knows conditional statements like while or if. if has the following syntax

```
add.vector <- function(x,y) { return(x+y)}
add.vector(x,x)

## [1] 2 4 6
```

where we have also used the any function, which should have a quite intuitive functionality. Its complement is all. while is used similarly

```
x <- numeric()
i <- 1
while(i < 5) {
  x[2*i] <- i
  i <- i+1
}
x

## [1] NA  1 NA  2 NA  3 NA  4
```

In this example we made use of the automatic extension of vectors by R. Not directly set elements will be set to NA for not available. R is used to handle such missing values. For instance, the sum function has explicit means to handle those

```
sum(x)

## [1] NA

sum(x, na.rm=TRUE)

## [1] 10
```

As you can see there is also an explicit boolean type with possible values TRUE or FALSE. There are also the usual boolean operators available, like && or ||. More often than while one uses for loops

```
ii <- which(!is.na(x))
y <- numeric()
for(i in ii) {
  y[i] <- 2*x[i]+1
}
y[-ii] <- 0
y
```

```
## [1] 0 3 0 5 0 7 0 9
```

Of course, in R you would rather write instead by avoiding the `for` loop

```
ii <- which(!is.na(x))
y[ii] <- 2*x[ii]+1
y[-ii] <- 0
y

## [1] 0 3 0 5 0 7 0 9
```

which is much more efficient in R. So, whenever possible avoid loops and use index arrays and, most importantly, `apply`. We remind you again that indices start with `1` in contrast to C/C++.

A.1.2. Plotting Data

R has strong graphical capabilities. We will exemplify this here by writing a function that plots data with error bars, which is not available per default in R. So, lets assume we have data available as follows

```
x <- c(0,1,2,3,4)
y <- c(0, 1.1, 3.8, 9.05, 15.5)
dy <- c(0.2, 0.24, 0.22, 0.30, 0.4)
```

The standard R plot routine without errorbars works as follows

```
plot(x=x, y=y)
```

with the result displayed in Figure A.1.1. Now lets set-up a function doing this

```
plotwitherror <- function(x, y, dy) {
  plot(x=x, y=y)
}
```

To add the error bars we use the `arrows` function

```
plotwitherror <- function(x, y, dy) {
  plot(x=x, y=y)
  arrows(x0=x, y0=y-dy, x1=x, y1=y+dy, length=0.01,
         angle=90, code=3)
}
plotwitherror(x=x, y=y, dy=dy)
```

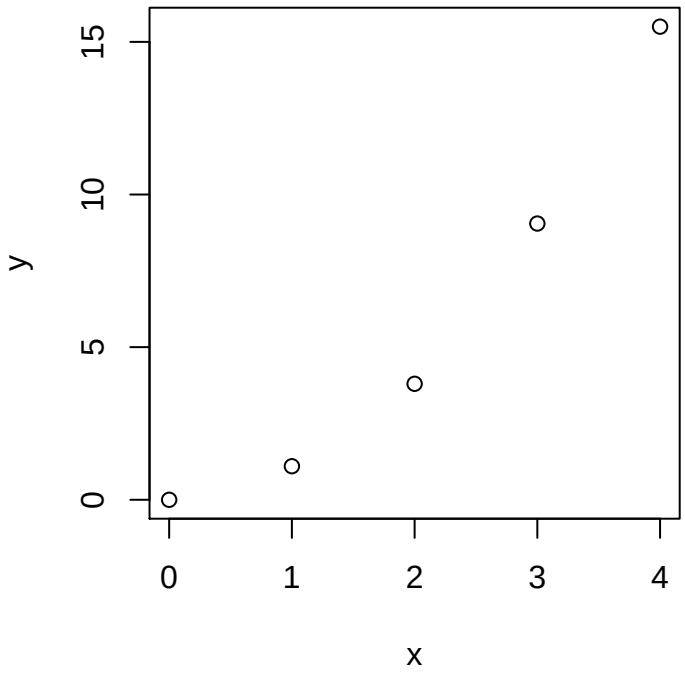


Figure A.1.1.: Simple R plot of the sample data.

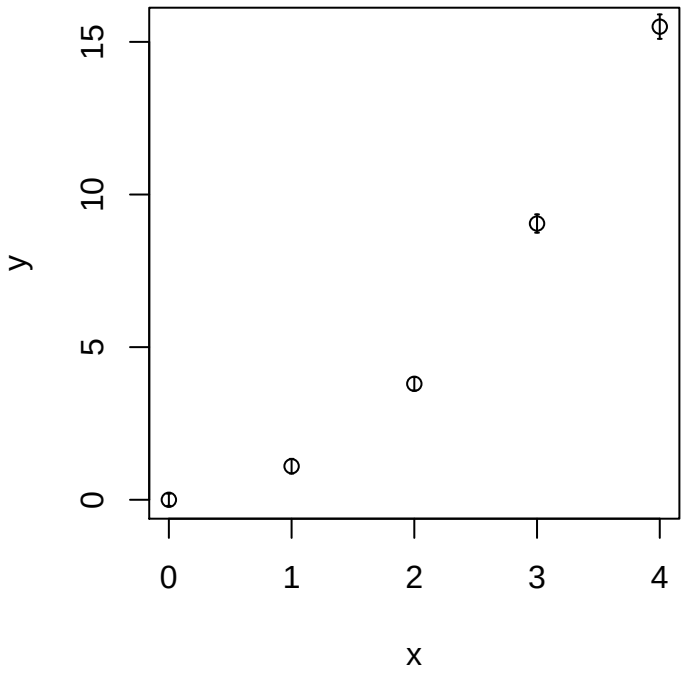


Figure A.1.2.: Plot with error bars.

with the result displayed in Figure A.1.2. Now, we would actually like to set the limits on x– and y–axis by ourselves. For this R provides the special ... argument for functions

```
plotwitherror <- function(x, y, dy, col="black", ...) {
  plot(x=x, y=y, col=col, ...)
  arrows(x0=x, y0=y-dy, x1=x, y1=y+dy, length=0.01,
         angle=90, code=3, col=col)
}
```

where we have also added the possibility for coloured points. Instead of using the ... argument, we have made `col` explicit, because we want to pass this to the `arrows` function as well. We have given it the default value `"black"`. There are other useful graphical parameters to pass to plot, like e.g. `pch` to change the point type. So, here a nicer version of the plot with error bars

```
plotwitherror(x=x, y=y, dy=dy, col="blue",
              xlim=c(-0.5, 4.5), ylim=c(0,16),
              pch=21, bg="blue", main="", xlab="x", ylab="y")
legend("topleft", legend=c("my data with errors"),
       bty="n", col="blue", pt.bg="blue", pch=21)
```

shown in Figure A.1.3. We have also added a legend using `legend`, try `?legend` to find out about this function. The plots in the main text of this lecture have been generated using the package `tikzDevice` for nicer look and LaTeX fonts.

Lets now finally add a curve to the plot, which is not so handy in R without addon packages, but one gets used to it. We first need to create a sequence of x–values, next the corresponding y–values from the x–values and finally use `lines` to plot it as a curve

```
plotwitherror(x=x, y=y, dy=dy, col="blue",
              xlim=c(-0.5, 4.5), ylim=c(0,16),
              pch=21, bg="blue", main="", xlab="x", ylab="y")
t <- seq(x[1], x[5], by=0.05)
ty <- t^2
lines(x=t, y=ty, lty=c(2), col="red")
legend("topleft", legend=c("my data with errors"),
       bty="n", col="blue", pt.bg="blue", pch=21)
```

as shown in Figure A.1.4. Please check the help page of the function `dev.copy2pdf` for creating a PDF of a plot.

Finally, note that R also provides the function `curve` to plot functions in a given interval. Please see `?curve` for details.

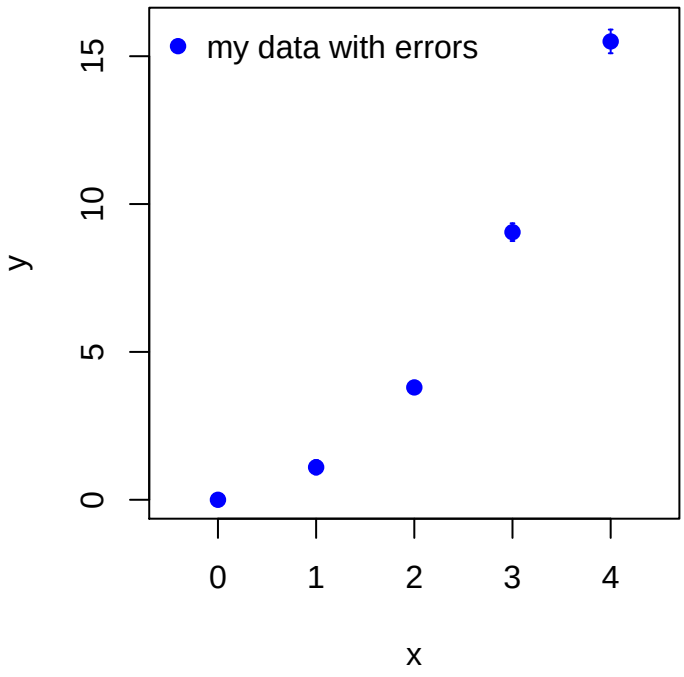


Figure A.1.3.: Nicer plot with error bars.

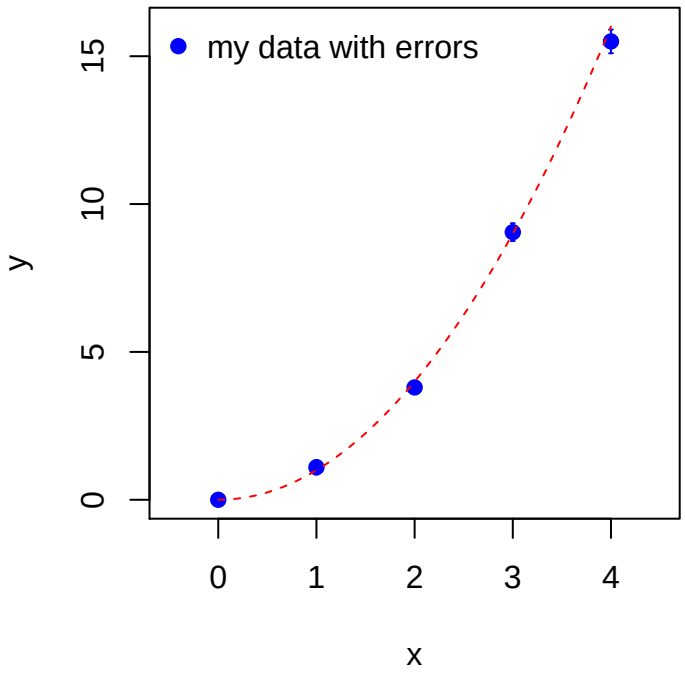


Figure A.1.4.: Nicer plot with error bars and extra curve.

A.1.3. Handling multi-dimensional data structures

As mentioned previously, R is very good in handling vectors and in general multi-dimensional arrays. We can create a two dimensional array as follows

```
X <- array(c(1:20), dim=c(5,4))
X

##      [,1] [,2] [,3] [,4]
## [1,]    1    6   11   16
## [2,]    2    7   12   17
## [3,]    3    8   13   18
## [4,]    4    9   14   19
## [5,]    5   10   15   20
```

which we filled from a vector. Here, you can see the index ordering or R directly. We can act with functions on all rows or colums in a very comfortable way using the `apply` function as follows

```
apply(X, MARGIN=1, FUN=sum)

## [1] 34 38 42 46 50

apply(X, MARGIN=2, FUN=sum)

## [1] 15 40 65 90
```

The first command applies the function `sum` to all rows simultaneously, the second one to all columns. Using `array` you can create data structures of any dimension and `apply` is very powerful in contracting those. We will make use of it in many circumstances.

It often appears that you have to add data to an array. This can be done in R using `cbind` and `rbind`

```
X <- cbind(X, c(21:25))
X

##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    6   11   16   21
## [2,]    2    7   12   17   22
## [3,]    3    8   13   18   23
## [4,]    4    9   14   19   24
## [5,]    5   10   15   20   25
```

A special container for data of the same length, but not neccessary of the same type is called `data.frame`

```
data <- data.frame(x=x, y=y, dy=dy)
data$x

## [1] 0 1 2 3 4

data[1,3]

## [1] 0.2
```

whose rows can be accessed by the names and the `$` operator, or by the index operator. You can also directly pass the `data.frame` to the plot function, however, then again without error bars or you have to modify our `plotwitherror` function accordingly.

A.1.4. pRN in R

Finally, we give references for creating uniform and normal distributed random numbers. The corresponding functions are `runif` and `rnorm`.

```
u <- runif(n=1000)                  ## uniform random numbers
x <- rnorm(n=1000, mean=0, sd=2)    ## normal distr. random numbers
hist(x, probability=TRUE, ylim=c(0,0.2),
     main="")
z <- seq(-8,8,0.1)
d <- dnorm(z, mean=0, sd=2)         ## the density function
lines(x=z, y=d, col="red", lwd=2)
```

A histogram of the random sample together with the density function generated with the function `dnorm` is displayed in Figure A.1.5.

A.1.5. Scripting R and Data Handling

It is often convenient to not type all commands at the `R`–comand prompt. For this purpose, `R` offers several possibilities. The first is to source scripts from the prompt using the `source("file.R")` command. The file `file.R` can contain definitions of functions and/or variables as well as commands to execute.

`source` can be rather useful when developing code. However, one needs to be careful with the variables defined in the evironment of the command line interpreter. You can list these as follows

```
ls()

##  [1] "add.vector"    "d"             "data"          "dy"
##  [5] "i"             "ii"            "M"             "old"
##  [9] "plotwitherror" "t"             "ty"            "u"
## [13] "x"             "X"             "y"             "z"
```

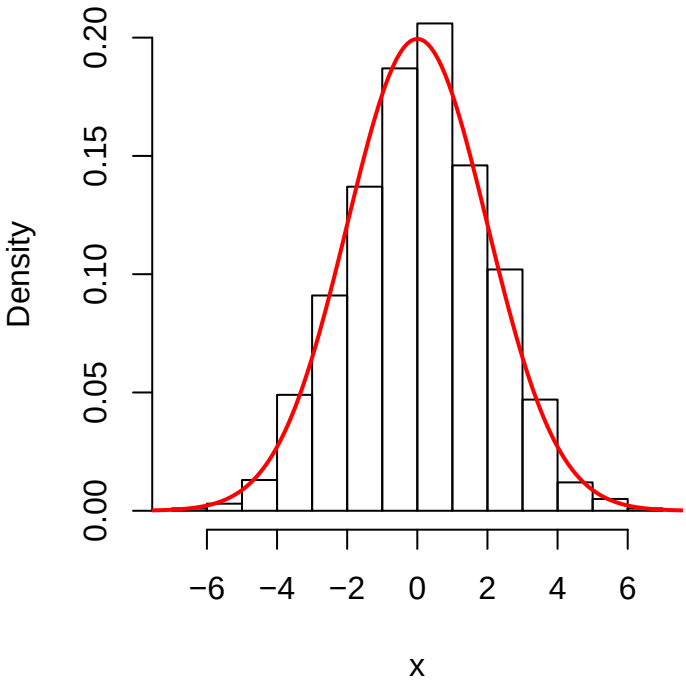


Figure A.1.5.: Histogram of normal distributed pRN.

Here you see the variables and functions we have defined in this introduction so far. You may remove some of them from the environment explicitly using

```
rm(y, z)
```

If not removed they will stay in the environment and can lead, sometimes, to unexpected behaviour. In addition, R is trying to find not defined objects in the next higher environment. For example, the following works

```
f <- function(y) return(ii)
f(3)

## [1] 2 4 6 8
```

if ii is defined not in the function environment, but in the one from which the function was called. Here, one needs to be careful.

One way to avoid such behaviour is to use Rscript which runs an R–script in a fresh environment, everytime it is called. From the Linux command line this is invoked as
`$ Rscript file.R`
We refer to the manual page for more details.

R also allows you to store parts or all of your object on hard disk. For this purpose the commands save and load are provided. save expects a named list of objects to save and a filename, while load only needs a filename

```
save(ty=ty, file="ty.Rdata")
rm(ty)
load("ty.Rdata")
ls()

##  [1] "add.vector"    "d"             "data"          "dy"
##  [5] "f"             "i"             "ii"            "M"
##  [9] "old"           "plotwitherror" "t"             "ty"
## [13] "u"             "x"             "X"
```

The function save.image stores an image of the current session per default to the file .Rdata. This is called by q(), if wanted. But it can also be used in scripts to store the progress of a calculation.

For reading data from disk, R provides a variety of functonality. Maybe the most used ones read.table and read.csv. They allow one to read data from an ASCII file into an R data.frame. We refer to the manual page of these for further details.

References

[1] J Cornelissen. *Introduction to R*. [Online; accessed 06-October-2016]. 2016. URL: https://www.datacamp.com/courses/free-introduction-to-r.

[2] C Sharpsteen and C Bracken. *tikzDevice: R Graphics Output in LaTeX Format*. R package version 0.6.3/r49. 2012. URL: `https://R-Forge.R-project.org/projects/tikzdevice/`.

[3] *The Comprehensive R Archive Network*. [Online; accessed 06-October-2016].

[4] RCT W. N. Venables D. M. Smith. *An Introduction to R*. [Online; Version 3.3.1, accessed 06-October-2016]. 2016. URL: `cran.us.r-project.org/doc/manuals/R-intro.pdf`.

[5] C Yau. *R Introduction*. [Online; accessed 06-October-2016]. 2016. URL: `http://www.r-tutor.com/r-introduction`.

Index