Stochastic Modeling:

An Excursion

Reinhard Furrer
and the Applied Statistics Group

Version July 10, 2018

Contents

Acknowledgment

This document accompanies the lectures *STA111 Stochastic Modeling* and *MAT919 Stochastic Simulation* in the spring semester of 2018. The lecture has been designed in the framework of the minor in *Applied Probability and Statistics* (www.math.uzh.ch/aws) and is open to anyone with interest in stochastic modeling and moderate background in probability theory and statistics.

The first version of the document consisted of a chapter summaries from the participants of the Fall 2016 edition of the lecture. It was very interesting to see how they have perceived and digested my lecture and there are still flavors of this document that carries their names: thanks to Damian Durrer, Philipp Nikolaus, Leander Mösinger, Cristina Bozzo, Beatrice Ehmann, Roman Flury, Alessandra Urbinati and Jonas Bächinger.

A huge thanks to the assistants of the lecture, Florian Gerber and Roman Flury. The discussions had a significant effect on the content of this script and structure of the lecture. While we made a significant step forward this document needs more than a polishing. Please let me know of any necessary improvements and I highly appreciate all forms of contributions in form of errata, examples, or text blocks. Contributions can be deposited directly in the following Google Doc sheet.

Reinhard Furrer
June 2018

Preface

Stochastic simulation is a powerful tool used in many different scientific domains. For several decades, scientists have realized that adding stochastic elements, in some view uncertainty, has advantages ranging from

1. Better understanding of a complex phenomena
2. Extending deterministic models
3. Tool for uncertainty assessment
4. Data compression

Not all of these points will be discussed with equal depth but we link to these points whenever appropriate.

The main goal of the lecture is to expose the audience to a series of questions or problems of various depth and to give stochastic modeling and simulation based approaches to solve these. Based on such a problem-solving oriented approach, we discuss theoretical aspects only as necessary. This implies that we do not follow the classical "Stochastic Modeling" textbook setup, which typically start with the chapters: Introduction, Probability theory, Markov chains, Point processes, ... (see, e.g., Pinsky and Karlin, 2011; Taylor and Karlin, 1998; Beichelt, 2006; Guttorp, 1995). Here, we develop part of the theory as required by the application and thus almost have the reverse approach. There are many textbooks available, none is fully convincing and thus we use sections and chapters of several of them.

All the datasets that are not part of regular CRAN packages are available via the url user.math.uzh.ch/furrer/download/sta111/.

We try to use as consistently as possible the "classical" notation: uppercase for random quantities, lower case for realizations, boldface for vectors (lower case) and matrices (upper case). Inherently, a matrix can not be differentiated from a random vector based on notation only. The context and, if necessary, explanations will clarify.

Chapter 1

Drawing random variables

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta111/chapter01.R.

At the core of stochastic simulation is the mechanism of generating "random numbers". We should use quotes because in practice we never have genuinely random numbers but only a sequence of numbers that *seem* random, i.e., they give the impression of being a random sample from a given distribution.

Such a simple mechanism is required because it is often impractical to perform experiments delivering a required sequence of numbers.

If truly random numbers are needed nevertheless, quite complex mechanisms are required. An example is the lava lamp wall, as illustrated in Figure 1.1, which is translated to a sequence of numbers by image processing tools.

Figure 1.1: A wall full of lava lamps, used to generate random numbers. Screenshot taken from `www.youtube.com/watch?v=1cUUfMeOijg` at 2:53.

1.1 How to Draw Random Variables

A realization of a random variable X is the "observed value" of X seen as a function: $x = X(\omega)$. In practice we often work with a random sample, i.e., $X_1, \ldots, X_n$ independent and identically distributed and we use the term realization of a random sample (or in case of a single value a realization of a random variable).

In practice, we often start drawing a realization of a uniform distribution and derive from that sequence a new sequence according to certain "rules" such that we can assume the final sequence a realization of our desired random variable.

For the moment, we assume that we have an efficient uniform random number generator and discuss two fundamentally different approaches to generate specific sequences. We discuss uniform random number generator later.

1.1.1 Inverse Function

For all continuous random variables with distribution function $F(x)$, for which a closed form of the quantile function $F^{-1}(x)$ exists, we can directly draw random samples using the following approach.

Property 1.1. *Let $U \sim \mathcal{U}(0,1)$ and $g(u) = F^{-1}(u)$. Then $X = g(U)$ has distribution function $F(x)$.*

This result is due to

$$\mathrm{P}(X \leq x) = \mathrm{P}(g(U) \leq x) = \mathrm{P}(U \leq g^{-1}(x)) = \mathrm{P}(U \leq F(x)) = F(x). \tag{1.1}$$

Example 1.1. We have some well-known and quite simple examples. Let $U \sim \mathcal{U}(0,1)$.

1. Let $X = g(U) = -\log(1-U)/\lambda$, $\lambda > 0$. Then X is distributed according to an exponential distribution with rate λ: $X \sim \mathcal{E}xp(\lambda)$.

2. Let $X = g(U) = \tan\big(\pi(U - 1/2)\big)$. Then X is distributed according to a Cauchy random variable, i.e., a random variable having density $f(x) = (1 + x^2)^{-1}/\pi$. The Cauchy distribution is used to model random variables with very heavy tails, as its density decays so slowly that $\mathrm{E}(|X|) = \infty$. ♣

Remark 1.1. Naturally, Proposition 1.1 still holds even if we do not have a closed form of the quantile function. Moreover, defining

$$F^{-1}(u) = \inf\{x : F(x) \geq u\}, \qquad 0 < u < 1, \tag{1.2}$$

the proposition holds for any distribution. The set specified in the right hand side of (1.2) is never empty due to the right-continuity of the cumulative distribution function.

The cases $F^{-1}(0)$ and $F^{-1}(1)$ are defined by limit arguments. ♣

Discrete random variables can be obtained by constructing a partition of the interval $(0,1)$ according to the probabilities of the random variable, i.e., the kth break-point is $\sum_{i=1}^{k} p_i$. We draw u from $U \sim \mathcal{U}(0,1)$ and set the variable to its kth value if u is between the $(k-1)$th and kth break-point.

Example 1.2. We have some well-known and quite simple examples. Let $U \sim \mathcal{U}(0,1)$.

1. Let $X = I_{\{U<p\}}$. Then X is distributed according to a Bernoulli random variable with success probability p.

2. Let $X = \lfloor 6U \rfloor + 1$, where $\lfloor x \rfloor$ denotes the integer part of x. Then X is distributed uniformly over $\{1,2,3,4,5,6\}$. ♣

This partitioning approach is the discrete case of the inversion approach. More formally, let X be a discrete random variable with values in $\{1,2,\ldots,n\}$ (possibly $n=\infty$) and cumulative distribution function $F_X(x)$ and define the intervals $I_k = [F_X(k-1), F_X(k))$, $k = 1,2,\ldots$. Then $X = k$ if $U \in I_k$, $k = 1,2,\ldots$.

Gaussian random variables are omnipresent and it is no surprise that several methods to draw realizations thereof have been published. There are even two simple algorithms that approximate the quantile function astonishingly well. The most well-known algorithm is probably the Box–Muller.

For many random variables, special methods can be derived. Examples include convolution, product, minimum, ... approaches. Examples thereof and more approaches are discussed in Leemis and McQueston (2008).

Figure 1.2 shows the dependency of many well-known and some quite exotic random variables. The network can be exploited to simulate random variables. Albeit, in practice, more efficient algorithms are used.

1.1.2 Rejection Sampling

When no method exists to directly sample from a continuous random variable X with density $f_X(x)$, rejection sampling can be used. In this method values from a known density $f_Y(y)$ are drawn and through rejection of unsuitable values, realizations of X are generated.

The procedure is as follows:

Step 0 Find an $m < \infty$, so that $f_X(y) \le m \cdot f_Y(y)$.

Step 1 Simulate a realization y from a random variable with density $f_Y(y)$ and simulate a realization u from $U \sim \mathcal{U}(0,1)$.

Step 2 If $u \le f_X(y)/\big(m \cdot f_Y(y)\big)$ then $x = y$ is accepted as a simulated value from $f_Y(y)$, otherwise return to Step 1.

Of course we cannot use an arbitrary density $f_Y(y)$. The support of X has to be in Y, otherwise m in **Step 0** does not exist. For efficiency reasons the constant m should be chosen to be as small as possible.

This method can also be used when the normalizing constant of $f_X(x)$ is unknown and is thus well suited in the framework of unknown normalizing constant.

Rejection sample can also be used for discrete random variables, with only minor modifications in the procedure.

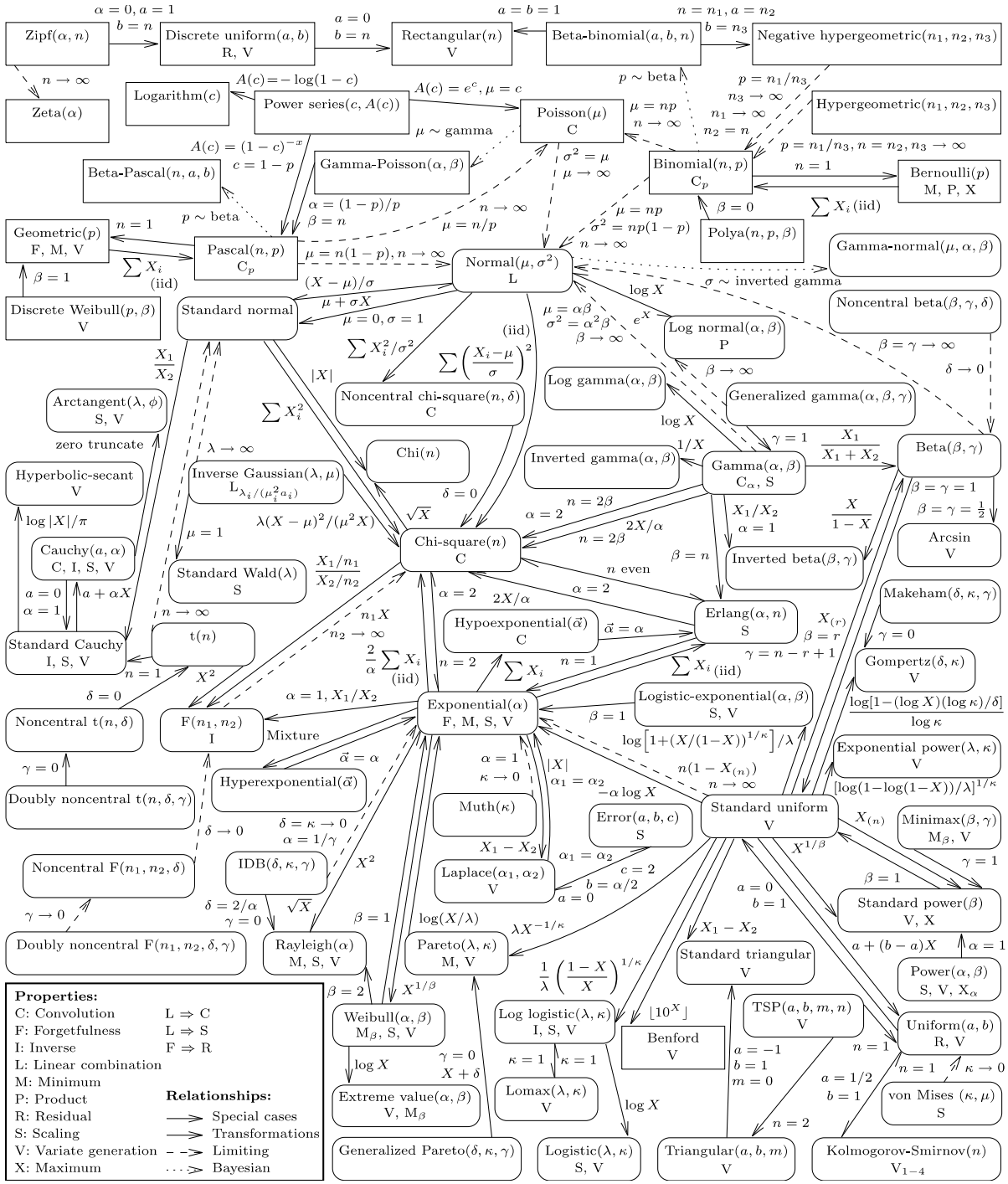

Figure 1.2: Univariate distriubtion relationships. Figure 1 of Leemis and McQueston (2008).

Example 1.3. We draw values from a $X \sim \mathcal{B}eta(6, 3)$ distribution with the rejection sampling method using the uniform distribution only. That means, $f_X(y) = c \cdot y^{6-1}(1-y)^{3-1}$ (the normalizing constant c is $1/\beta(6, 3) = 1/168$) and $f_Y(y) = 1_{0 \le y \le 1}(y)$. We select $m = 0.02$, which fulfills the condition $f_X(y) \le m \cdot f_Y(y)$.

An implementation of the example is given in R-Code 1.1. The R-Code can be optimized with respect to speed. It would then, however, be more difficult to read. Figure 1.3 shows a histogram and the density of the simulated values. ♣

R-Code 1.1 Rejection sampling (See Figure 1.3)

```
set.seed( 14)
n.sim <- 1000
m <- 0.02
f_X<- function(y) y^( 6-1) * (1-y)^(3-1)
f_Y <- function(y) ifelse( y >= 0 & y <= 1, 1, 0)
result <- sample <- rep( NA, n.sim)
for (i in 1:n.sim){
    sample[i] <- runif(1)
    u <- runif(1)
    if( u < f_X( sample[i]) /( m * f_Y( sample[i])) ) # if accepted ...
        result[i] <- sample[i]
}
mean( !is.na(result))  # proportion of accepted samples
## [1] 0.285
result <- result[ !is.na(result)]
length( result)
## [1] 285
hist( sample, xlab="y", main="", col="LightBlue")
hist( result, add=TRUE, col=4)
curve( dbeta(x, 6, 3), frame =FALSE, ylab="", xlab='y', yaxt="n")
lines( density( result), lty=2, col=4)
legend( "topleft", legend=c("Theoretical", "Simulated"),
       lty=1:2, col=c(1,4), bty='n')
```

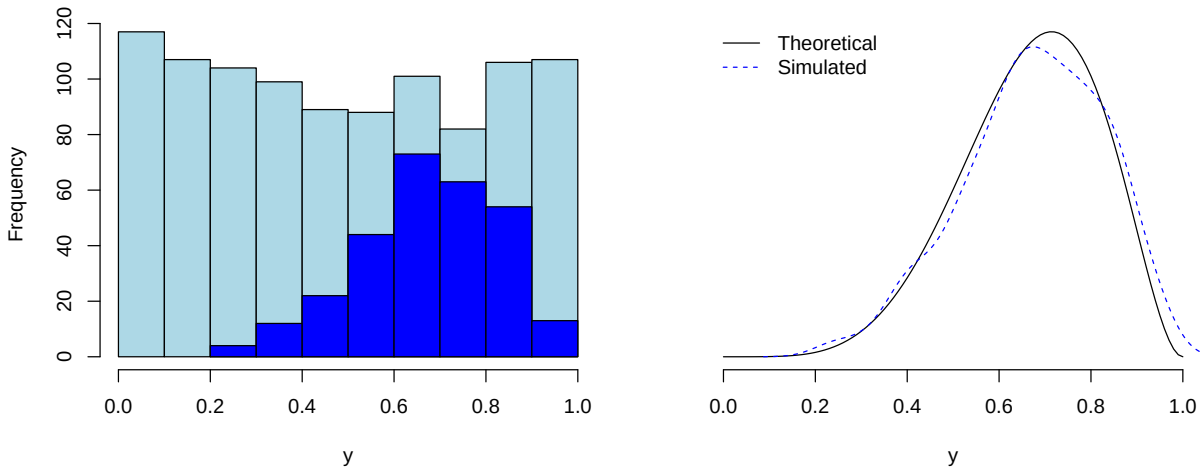

Figure 1.3: On the left we have a histogram of the simulated values of $f_Z(y)$ (light blue) and $f_Y(y)$ (dark blue). On the right the theoretical and simulated densities are drawn. (See R-code 1.1 and Example 1.3.)

Example 1.4. In analogy to the Box–Muller approach, the polar method to sample $Z_1, Z_2 \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$ is as follows:

Step 1: Draw u_1, u_2 from $U_1, U_2 \sim \mathcal{U}(-1,1)$.

Step 2: If $r = u_1^2 + u_2^2 > 1$ return to Step 1.

Step 3: Return the $z_i = \sqrt{-2\log(r)/r} \cdot u_i$, $i = 1, 2$.

See Example 2d, Chapter 10 of Ross (2010). ♣

More advanced techniques will be revisited in later chapters.

1.2 How to test for randomness

Here we only cover some keywords. Most of the tools are situated in statistics, ranging from visual tools, like QQ-plots (quantile-quantile plots), probability plots, to more quantitative tools like formal statistical tests (Anderson–Darling test, Kologorov–Smirnov test, ...).

1.3 Bibliographic Remarks

The content of this chapter is covered in many text books. The online chapters by Art Owen cover in much more details what we have discussed here: http://statweb.stanford.edu/~owen/mc/Ch-nonunifrng.pdf and http://statweb.stanford.edu/~owen/mc/Ch-unifrng.pdf.

A detailed story about Cloudflare's lava lamp wall is given in www.youtube.com/watch?v=1cUUfMeOijg, en.wikipedia.org/wiki/Cloudflare). A completely alternative approach to use random numbers is through www.amazon.com/Million-Random-Digits-Normal-Deviates/dp/0833030477.

In R it is possible to set random number generator for the uniform and for the Gaussian distribution. The current choices can be queried by *RNGkind()*. By default, the Mersenne–Twister (en.wikipedia.org/wiki/Mersenne_Twister and the inversion method are used, see also *?Random*. Note that setting a seed, internal random seeds are set. See lines 350ff from Marius Hofert's page github.com/qrmtutorial/qrm/blob/master/R/01_An_Introduction_to_R_Programming/01_An_introduction_to_R_programming.R. In R, the actual implemented code is given in, e.g., github.com/wch/r-source/blob/trunk/src/main/RNG.c.

Naturally, more efficient methods than the ones seen here are implemented. See for example the approaches for the exponential github.com/wch/r-source/blob/trunk/src/nmath/sexp.c or binomial github.com/wch/r-source/blob/trunk/src/nmath/rbinom.c case.

Other alternative (random) facts about random number generation are available at https://www.iro.umontreal.ca/~lecuyer/myftp/slides/rng-history-talk-X2017.pdf

Lab Content

1. Seeds, States and periods;
2. linear congruential generator (LCG), multiplicative congruential generator (MCG) and variants thereof;
3. R implementation of an LCG;
4. More about uniformity tests.

Worksheet 1

1. Simulate a discrete distribution: geometric random variable.
2. Simulate a continuous distribution: Gaussian random variable using the rejection method.
3. Explore the 'RANDU' random generator.

Chapter 2

Runs and Ruins

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta111/chapter02.R.

Mathematica file: www.math.uzh.ch/furrer/download/sta111/chapter02.nb.

In this chapter we consider a sequence of independent and identically distributed Bernoulli random variables, termed a sequence of trials. In the past, you may have modeled such a sequence of such experiments according to a fixed number of trials (binomial random variable) or the number of successes (negative binomial random variable with geometric random variable as a special case). Here we look at more specific and more interesting questions that arise from such sequences.

2.1 Runs

Surprisingly, random sequences are much more "regular" than one may naively assume.

We consider n (fixed, possibly large) iid Bernoulli random variable with success probability p. We encode the result with H/T, where H signifies "success".

Definition 2.1. *String*: a consecutive series of one kind.
Run: a consecutive series of one kind, bounded by the alternative kind. ♢

Let L_n the length of the longest run in a sequence of length n. We are interested in quantities of type $\mathrm{P}(L_n = k)$ or approximate bound such that $b_l \leq L_n \leq b_u$ with high probability.

There are different approaches to calculate or bound $\mathrm{P}(L_n = k)$. Two natural ones are as follows.

1. For large k, $\mathrm{P}(L_n = k)$ is small and can be seen as a rare event. Hence, use a Poisson approximation.

2. Derive the exact distribution of the run lengths, and derive from that the longest run. For example, Ross (2010), Example 7d, gives a formula for the exact distribution.

2.2 Ruins

We now investigate the setting if one observes first an excess of say i failures or of say $N - i$ successes. This question is wrapped in the following gambling situation.

Ben proposes a game to Sam, his younger brother: Sam should toss a biased coin, with probability for heads $p > 1/2$. If heads turns up, Sam receives one unit (from Ben), if tails is observed, Sam pays one unit to Ben. The game continuous until Sam or Ben lost all their money. When proposing the game, Sam arrives with a i units. Should Sam agree to play? Or rather, under what circumstances (values of p, i, ...) should he agree?

The answer to the question above depends on how many units Ben is putting on the table, i.e., what is Ben's starting amount. To illustrate this, we use the following formalism.

Table 2.1: Gambler's ruin setup.

Player	Coins (state)	Success probability
Sam	i	p
Ben	$N - i$	$1 - p$

Assume that Ben brings $N - i$ units to the game, see Table 2.1. Therefore at any moment in time, there is a total of N units involved, and if either has either 0 or N units, the game is over. It is convenient to illustrate the status of the game using states, here, the amount of one of the players (see Figure 2.1).

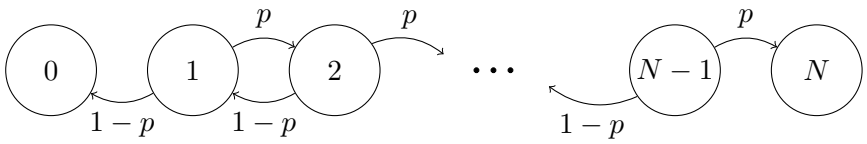

Figure 2.1: Status of the game using states $N + 1$ different states and possible movements. States 0 and N are absorbing.

To calculate the probability to win, P_i (i.e., that Sam wins when starting with i units), we start as follows:

$$\begin{aligned} P_i &= \mathrm{P}(\text{Sam reaches state } N, \text{ starting with at } i) && (2.1)\\ &= p \cdot \mathrm{P}(\text{Sam wins, starting at } i+1 \mid \text{success in the first round}) && (2.2)\\ &\quad + (1-p) \cdot \mathrm{P}(\text{Sam wins, starting at } i-1 \mid \text{no success in the first round}). && (2.3) \end{aligned}$$

Because the tosses are independent, we can simplify to

$$P_i = p \cdot P_{i+1} + (1-p) \cdot P_{i-1}, \qquad \text{for } i = 1, \ldots, N-1, \tag{2.4}$$

and "boundary" values $P_0 = 0$, $P_N = 1$. The problem is thus framed by a difference equation with constant coefficients and boundary values.

There are four different approaches solving the problem:

1. unrolling the iteration, illustrated below,
2. software oriented, illustrated in the accompanying Mathematica script,
3. based on the Ansatz $P_i = \theta^i$ (see end of this section),
4. using harmonic equations, i.e., which is essentially solving differential equations.

Here, we use the first approach for solving for P_i. Starting from (2.4) with left hand side $P_i = (p + (1-p))P_i$ we collect the terms as follows

$$p \cdot (P_{i+1}) - p \cdot (P_i) = -(1-p) \cdot P_{i-1} + (1-p) \cdot P_i \tag{2.5}$$

$$P_{i+1} - P_i = \frac{1-p}{p}(P_i - P_{i-1}). \tag{2.6}$$

We assume for the moment that $p \neq 1/2$ and write the equation for $i = 1$ with the boundary condition $P_0 = 0$. We recursively write it for larger i

$$P_i - P_{i-1} = \left(\frac{1-p}{p}\right)^{i-1} \cdot P_1 \tag{2.7}$$

to then use a "telescopic sum" argument. After summing the a geometric series, we use $P_N = 1$ and get, after back substitution:

$$P_i = \frac{1 - \left(\frac{1-p}{p}\right)^i}{1 - \left(\frac{1-p}{p}\right)^N}, \qquad p \neq \frac{1}{2} \tag{2.8}$$

R-Code 2.1 calculates and "visualizes" the probability $P_i = f(i, N, p)$, i.e., as a function of the parameters p, N and i. For example, for the setting $i = 8$ and $N - i = 28$, p needs to exceed 0.52 for $P_8 > 0.5$. Reversibly, if Ben brings only four units to the table, Sam's wins with higher probability when $p > 0.469$.

R-Code 2.1: Winning probability. (See Figure 2.2.)

```
p <- 0.52       # success probability
i <- 8          # value A brings
N <- 28+i       # total amount
# Winning probability for different i, p and N:
p_i <- function(i, p, N)   ( 1- ( (1-p)/p )^i)/( 1- ( (1-p)/p )^N)

# Given i = 8 and N = 28 + i, what is the probablilty of Sam  winning?
pseq <- seq(0.4, to=0.8, length=100)
plot(pseq, p_i( i, pseq, N), type='l', ylab=expression(P[i]),
     xlab="p=P(success)", ylim=c(0,1))
abline( h=0.5, v=c(0.5, p), col=c("gray", "gray", 4))
```

```
legend("bottomright", legend=c("i = 8", "N-i = 28"), bty='n', cex=1.2)
# If Ben comes with just a tiny bit, 4 units, p can be much smaller!
plot(pseq, p_i( i, pseq, i+4), type='l', ylab=expression(P[i]),
     xlab="p=P(success)", ylim=c(0,1))
abline( h=0.5, v=c(0.5, 0.469), col=c("gray", "gray", 4))
legend("bottomright", legend=c("i = 8", "N-i = 4"), bty='n', cex=1.2)
# Probablilty of Sam winning the game at different values of N?
Nseq <- seq( i+4, to=80, by=1)
plot( Nseq, p_i( i, 0.52, Nseq), type='l', ylab=expression(P[i]), xlab="N")
abline( h=1-((1-p)/p)^i, col=2) # asymptotes off
abline( h=.50, v=N, col='gray')
legend("topright", legend=c("i=8", "p=0.52"), bty='n', cex=1.2)
# Given p = 0.52, what is the probablilty of
# Sam winning the game at different values of i and N?
Niseq <- iseq <- 1:50
grid <- expand.grid( i=iseq, Ni=Niseq)
grid$N <- grid$i+grid$Ni
PE <- apply( grid, 1, function(x) { p_i(x[1], p, x[3])})
PE <- matrix( PE, 50,50)
PE[ abs(PE-.5)<.025] <- NA
fields::image.plot(iseq, Niseq, PE, xlab="i", ylab="N-i")
```

The bottom left panel of Figure 2.2 illustrates another interesting fact. Gambling houses can stay in business because of their original capital is much larger than that of individual players.

As an alternative view of the same result, consider an isolated island which can provide a sustainable environment for N individuals only. Suppose a mutant is born and in every generation he gains one position with probability p and looses one with probability $1-p$. Then with probability $(2p-1)/p$ the mutants take over. Because of the unrealistically long absorption time, this model is not adequate for practical settings.

Only for reference: using the Ansatz $P_i = \theta^i$, $\theta \neq 0$, in (2.4) leads to

$$\theta^i = p\theta^{i+1} + (1-p)\theta^{i-1} \tag{2.9}$$

$$\Rightarrow \quad 0 = p\theta^2 - \theta + (1-p) \tag{2.10}$$

$$\Rightarrow \quad \theta_1 = 1, \quad \theta_2 = (1-p)/p \tag{2.11}$$

$$\Rightarrow \quad p_i = \begin{cases} P_i = A_1\theta_1^i + A_2\theta_2^i = 1 + \left(\frac{1-p}{p}\right)^i, & \text{if } p \neq 1/2, \\ P_i = A_1 + A_2 i, & \text{if } p = 1/2. \end{cases} \tag{2.12}$$

The constants A_1 and A_2 are found by using the boundary conditions, here $P_0 = 0$, $P_N = 1$.

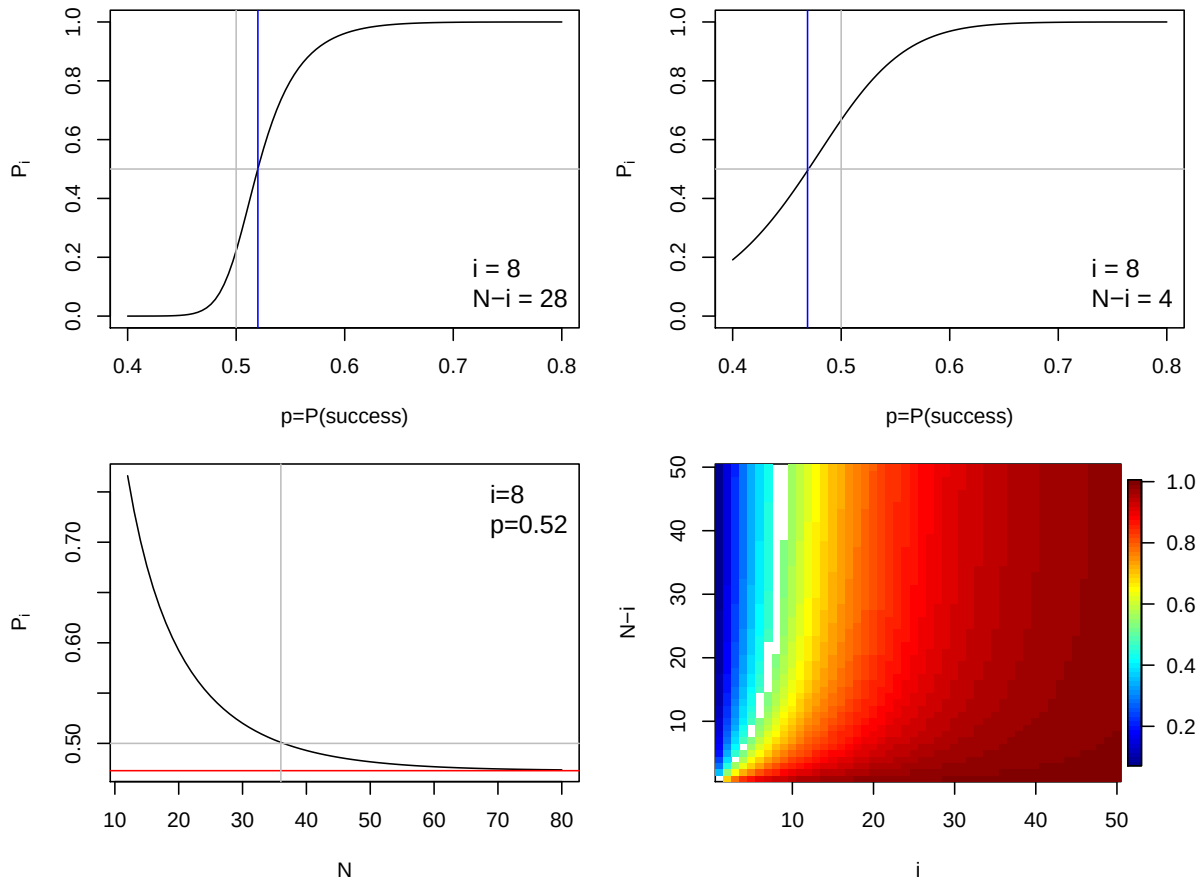

Figure 2.2: Winning probabilities. Top row: probability that Ben wins with starting amount $i = 8$ as a function of p (left for $N - i = 28$, right for $N - i = 4$. Bottom left: winning probability with increasing N ($p = 0.52$ and $i = 8$). Bottom right: winning probability for different i and Ni ($p = 0.52$). (See R-Code 2.1.)

2.2.1 Duration of the Game

An alternative question is how long a game lasts on average. We have to calculate expectations along the number of steps in the game.

Denote with D_i the expected number steps until reaching 0 or N when starting in state i. Similarly to the first step analysis for the winning probability $P_i = p \cdot P_{i+1} + (1-p) \cdot P_{i-1}$, we have

$$D_i = p \cdot (1 + D_{i+1}) + (1-p) \cdot (1 + D_{i-1}), \qquad \text{for } i = 1, \ldots, N-1, \tag{2.13}$$

with boundary conditions $D_0 = D_N = 0$. The solution of this difference equation is somewhat more evolved compared to the winning probability because we have an *inhomogeneous difference equation*. It can be shown that

$$D_i = \begin{cases} \dfrac{1}{2p-1}(NP_i - i), & \text{if } p \neq 1/2, \\ i(N-i), & \text{if } p = 1/2. \end{cases} \tag{2.14}$$

with P_i given by 2.8. The accompanied Mathematica script illustrates the solution as well.

R-Code 2.2 illustrates the calculation of the duration of the game as a function of N for different probabilities. For $p = 1/2$ the time is linear in n, for other probabilities it is virtually linear in N, shown in Figure 2.3.

R-Code 2.2 Duration of the game (See Figure 2.3.)

```
D_i <- function( i, p, N) {
        if (length(p)>1) {
     ifelse(p==1/2,  (N-i)*i,  (i-N * p_i(i, p, N))/(1-2*p))
        } else {  # single p, many i and N
                if (p==1/2) (N-i)*i else  (i-N * p_i(i, p, N))/(1-2*p)
        }
}

print( i)
## [1] 8
plot( Nseq, D_i( i, 0.5, Nseq), type='l', ylab="Expected duration", xlab="N")
lines( Nseq, D_i( i, 0.52, Nseq), col=2)
lines( Nseq, D_i( i, 0.6, Nseq), col=4)
legend("topleft", lty=1, col=c(1,2,4), legend=c("p=1/2", "p=0.52", "p=0.6"),
       bty='n')
lines( c(i, 60), D_i( i, 0.52, c(i,60)), col='gray')
lines( c(i, 60), D_i( i, 0.6, c(i, 60)), col='gray')
pseq <- seq(0.01, 0.99, length=200)
plot( pseq, D_i( i, pseq, 60), type='l', ylab="Expected duration", xlab='p')
abline( v=c(0.5, 0.52, 0.6), col='gray')
```

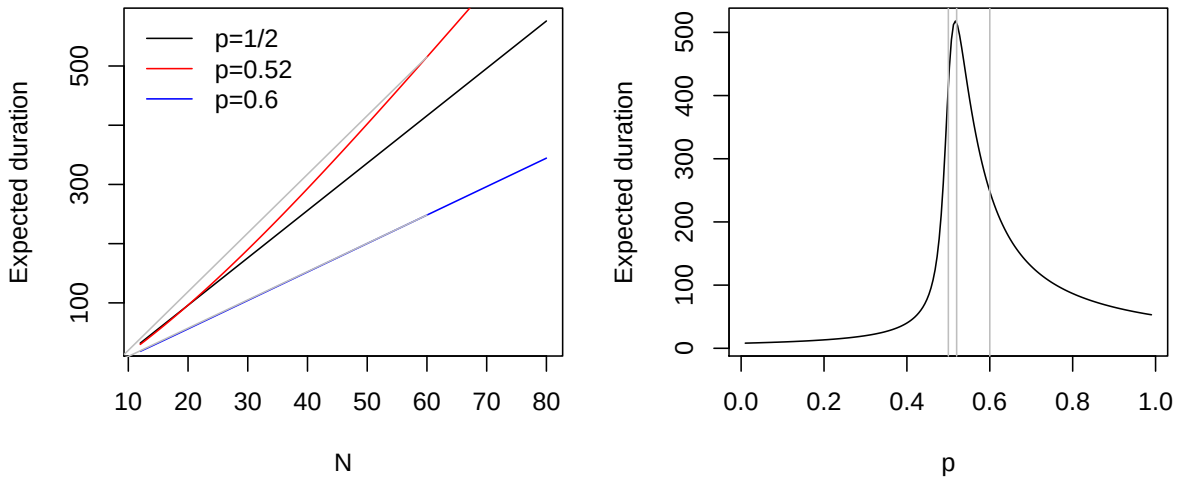

Figure 2.3: Top: duration of the game as a function of the amount N for three different success probabilities ($i = 8$). Bottom duration of the game for fixed $N = 60$ as a function of success probability ($i = 8$). (See R-Code 2.2.)

2.2.2 Alternations of the Game

There are many alternations of "Gambler's ruin" problem. For example, one of the players has a rich uncle that covers all his losses (assume that Sam's losses are covered). Hence the game does not end if he has lost all his money but (using state notation)

$$\mathrm{P}(S_{k+1} = 1 \mid S_k = 0) = p = 1 - \mathrm{P}(S_{k+1} = 0 \mid S_k = 0). \tag{2.15}$$

The state "0" is now a reflecting barrier, also sketched in Figure 2.4. The duration of the game can also be calculated by using the boundary conditions $D_N = 0$ and $pD_0 = 1 + pD_1$.

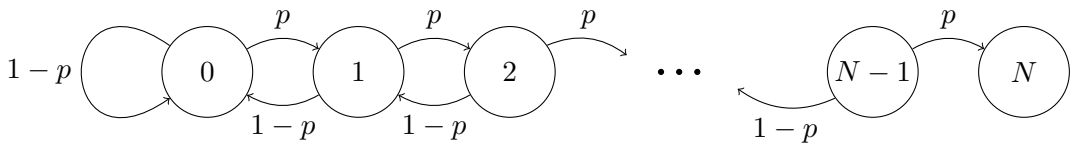

Figure 2.4: Status of the game using states. State 0 is reflecting and state N is absorbing.

2.3 Bibliographic Remarks

Exact distribution of the length of longest run is given in Ross (2010); Sen (1991) gives an alternative approach. See also the accessible account of Schilling (1990), explaining the heuristic of $\mathrm{E}(L_n) \approx \log_2(n) - 2/3$ as well as $\mathrm{sd}(L_n) \approx 1.83$. Note that Embrechts *et al.* (1997) give a much more precise interval, discussed in Worksheet 2.

The problem of ruins is covered in many classical books, here are some specific links: Ross (2010), Chapter 3, Example 4l; Grimmett and Stirzaker (2001), Section 1.7.4 and 3.9.6; Kemeny *et al.* (1960), Section 7.1.

Lab Content

1. Length of the longest run: Assumed a coin is flipped n times independentely, what is the probability that there is a run of k consecutive heads? An approximative solution can be found with the help of the Poisson paradigm, see Ross (2010) for a complete derivation.

2. Simulation study of longest runs

Worksheet 2

4. Moments of number of runs.

5. Simulation of the problem of ruins (one player game).

6. Precise approximate confidence interval of the longest run.

Chapter 3

Random Walks

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta111/chapter03.R.

Mathematica file: www.math.uzh.ch/furrer/download/sta111/chapter03.nb.

We use again Bernoulli trials but consider sequential sums thereof. This extends the concept of ruins to several dimensions.

3.1 Airplane Evacuation

We try to study an unstructured, chaotic airplane evacuation problem as given by Guttorp (1995), page 87. Due to the heavy smoke in the cabin, inexperienced travelers lost orientation and arbitrarily moved in the cabin in desperate search for an (emergency) exit. The movements have been reported to the post accident investigation as "random", passengers seemed to have moved arbitrarily left or right, even crawled across seating rows. We will simulate the evacuation, albeit not a simultaneous simulation of all passengers, but individual traveler movements.

We start with the strong assumption that, in case of such an emergency, when a passenger reaches the aisle before the window he survives. Otherwise, he dies. This simplifying assumption is based on the idea that hitting a window will lead to a delay that is too long to evacuate on time. In other words, we want to find the probability to reach the aisle before the window (bottom and left border) when sitting in the bottom 3x20 block of a Boeing 737 (Figure 3.1).

We thus imagine the sitting area (denoted with D, domain) to have boundaries around (denoted with B), all being absorbing. As we are interested in survival probabilities we set B_1 for the aisle and B_0 for the window and are interested in $P($ reach aisle before window $)$.

Figure 3.1: Seating arrangement for a Boeing 737 (Guttorp, 1995, p. 87)

R-Code 3.1 illustrates the simulation. In the later part of the chapter we see techniques to formally solve the problem.

R-Code 3.1: Evacuation problem. (See Figure 3.2.)

```
require( spam)    # precmat.GMRFreglat
require( fields)  # tim.colors

nrow <- 20     # defining the length of the sitting grid
ncol <- 3

# setting the window (0) and aisle (1) boundaries
fb <- c( rep(1,ncol), # left to right
         rep(1, nrow), # top-down
         rep(0, ncol), # top-down
         rep(0, nrow)) # left to right

nbou <- length( fb)
nint <- nrow*ncol

# Constructing Q  (D to D steps of the matrix)
Q <- -precmat.GMRFreglat(ncol, nrow, par=1/4)
diag( Q) <- 0
#  rowSums(Q)

# Constructing R  (D to B steps of the matrix)
tmp0 <- diag.spam(ncol)/4
pad( tmp0) <- c(nint,ncol)
tmp1 <- tmp2 <- spam(0,nint,nrow)
tmp1[ cbind( (1:nrow)*3, 1:nrow)] <- 1/4
tmp2[ cbind( (1:nrow)*3-2, 1:nrow)] <- 1/4
tmp3 <- spam(0, nint,ncol)
tmp3[ nrow*ncol-2:0] <- diag.spam( 1/4, 3)

R <- cbind( tmp0, tmp1, tmp2, tmp3)
# rowSums( cbind(R,Q))

# calculating the D to B part of P^n and
# then the probabilities to reach B_1 before B_0
B <- solve(diag.spam(nint)-Q) %*% R
fD <- B %*% fb       #*#
fDmat <- matrix( fD, nrow, ncol, byrow=TRUE)
```

```
head( fDmat, 4)
##         [,1]    [,2]    [,3]
## [1,] 0.53304 0.76589 0.89907
## [2,] 0.36628 0.63146 0.83038
## [3,] 0.30060 0.56330 0.79098
## [4,] 0.27282 0.53017 0.77024
plot_seats <- function(seats, ...) {
  image.plot(1:nrow(seats)-1, 1:ncol(seats)-1, seats, xlab='',ylab='',...)
  abline( h=1:ncol(seats) - 0.5, v=1:nrow(seats) - 0.5)
  rect(.5, .5, nrow(seats) - 1.5, ncol(seats) - 1.5, border="white", lwd=4)
}

pall <- matrix( NA, nrow+2, ncol+2)
pall[1+1:nrow,1+1:ncol] <- fDmat
plot_seats( pall, zlim=c(0, 1), col=tim.colors(20)[20:1])
```

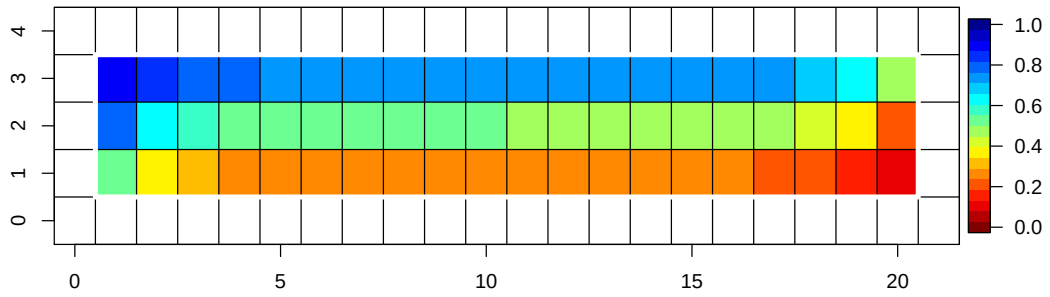

Figure 3.2: Survival probabilities of simple airplane evacuation simulation. (See R-Code 3.1.)

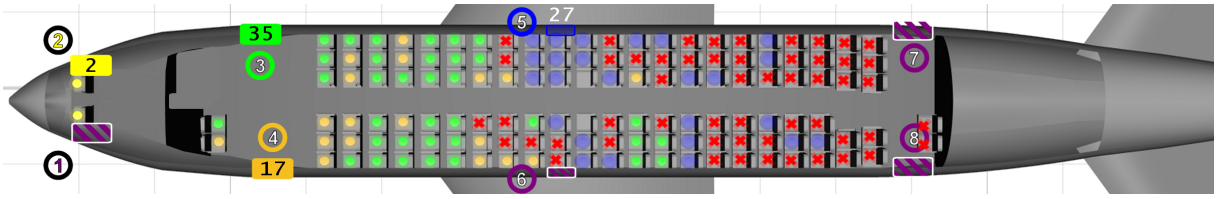

Figure 3.3: Ultimate survivor list. (Source: en.wikipedia.org/wiki/British_Airtours_Flight_28M.

We leave it to the interested reader to play with the simulation setup. Alternative setups may be:

1. Window seats are reflecting boundaries. This would entail that survival should be calculated by P(reach aisle within a certain time).

2. The aisle is not an absorbing state, but movement is mainly forth and possibly back. In case of blockage some revert their path.

The simulation setup does not take other passengers into account. A fully interactive setup would require a tremendous effort as it is not sufficient to follow one single "particle" on a lattice. A simulation that simulates the individual paths by iterating over passengers and over time might mimic interaction to a certain degree.

3.2 Random Walks on the Integers

Let $X_1, X_2 \ldots$, iid random variables with $\mathrm{P}(X_i = -1) = \mathrm{P}(X_i = -1) = 1/2$. We define the (symmetric) random walk (on the natural numbers) as

$$S_n = \sum_{i=1}^{n} X_i. \tag{3.1}$$

Notice, that we have discrete "fixed" steps. We typically set $S_0 = 0$.

We can "visualize" the random walk as a particle making unit jumps on a regular lattice. If n is large enough we can (theoretically) walk n units in one direction. Gambler's ruin for $p = 1/2$ is a simple symmetric random walk with a finite lattice.

We are interested in calculating the probability of being at a certain state at a certain time. To answer this, we denote $p_j(n) = \mathrm{P}($ particle is in state j at time n $)$. By independence we have

$$\begin{aligned} p_j(n) &= \frac{1}{2}p_{j-1}(n-1) + \frac{1}{2}p_{j+1}(n-1) && (3.2)\\ &= \frac{1}{2}\Big(\frac{1}{2}p_{j-2}(n-2) + \frac{1}{2}p_j(n-2)\Big) + \frac{1}{2}\Big(\frac{1}{2}p_j(n-2) + \frac{1}{2}p_{j+2}(n-2)\Big). && (3.3)\end{aligned}$$

Ignoring the "boundaries", we write the set of $p_i(n)$ in form of a row vector $\boldsymbol{p}(n)$. We can write

$$\boldsymbol{p}(n) = \boldsymbol{p}(n-1)\tilde{\mathbf{Q}} = \boldsymbol{p}(n-2)\tilde{\mathbf{Q}}\tilde{\mathbf{Q}} = \cdots = \boldsymbol{p}(n-m)\tilde{\mathbf{Q}}^m, \qquad 0 < m < n, \tag{3.4}$$

where $\tilde{\mathbf{Q}}$ contains $\pm 1/2$ on the elements just next to the diagonal. We call these elements transition probabilities, and the matrix with these probabilities the transition matrix. The non-zero elements of the matrix $\tilde{\mathbf{Q}}^m$ have an alternating checkerboard pattern for m odd/even. These elements can be read as "probability to move from point i to j in m steps".

If we do not have absorbing or reflecting states the size of the matrix $\mathbf{Q}$ depends on n. More specifically, in such a case we have a random walk on $\mathbb{Z}$ and for n fixed $\tilde{\mathbf{Q}} \in \mathbb{R}^{2n+1\times 2n+1}$. For the moment, we will restrict ourselves to finite lattices and thus to "classical" matrices.

Hence assume that we have a lattice of size K, i.e., the system has K different states. Thus $\boldsymbol{p}(n)$ is an K-dimensional row vector.

In case state j is an absorbing state, then once entered, we never leave: $p_j(n+k) = 1$ for all $k \geq 1$. If, for example, $b > 0$ is a reflecting state, then in the next step it will be in state $b-1$ with probability one. Hence, for absorbing or reflecting states, we can write similar equations as (3.2).

We write $\mathbf{P} = (p_{ij})$ the matrix containing the transition probabilities, where we (possibly) reorder the states, starting with all absorbing ones denoted with B and remaining ones, denoted with D. Notice that transition matrix of the set of absorbing states are in form of an identity

matrix. The transition matrix describing the movements within non-absorbing corresponds (essentially) to the matrix $\tilde{\mathbf{Q}}$. We also need to elaborate the transitions from interior to absorbing, which we denote $\mathbf{R}$. Finally, the transition matrix is given by

$$\mathbf{P} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{R} & \mathbf{Q} \end{pmatrix}. \tag{3.5}$$

We are interested in calculating the probabilities b_{ij} that the process starts in an interior state of D and ends in an absorbing state of B. This is

$$b_{ij} = p_{ij} + \sum_{k \in D} p_{ik} b_{kj}. \tag{3.6}$$

In matrix form $\mathbf{B} = \mathbf{R} + \mathbf{QB}$. Hence, $\mathbf{B} = (\mathbf{I} - \mathbf{Q})^{-1}\mathbf{R}$. For the ultimate probability in question, we can post multiply $\mathbf{B}$ with a vector containing 0 and 1 only, according to the boundary state value, see line (#*#). (We admit that we have used quite some hand-waiving here, more formal arguments follow soon).

3.2.1 Random Walks on Lattices and in Higher Dimensions

We now extend the (symmetric) random walk from $\mathbb{Z}$ to $\mathbb{Z}^d$, often $d = 2$ or 3.

Definition 3.1. Let $X_1, X_2, \ldots$ be iid with

$$\mathrm{P}(X_j = -\boldsymbol{e}_k) = \mathrm{P}(X_j = \boldsymbol{e}_k) = \frac{1}{2d}, \tag{3.7}$$

where $\boldsymbol{e}_k$ with $k = 1, \ldots, d$ represents the standard basis $\boldsymbol{e}_1 = (1, 0, \ldots, 0)^\top$, $\ldots$, $\boldsymbol{e}_d = (0, \ldots, 0, 1)^\top$ in $\mathbb{Z}^d$. A symmetric random walk S_n in d dimensions is given by

$$S_n = X_1 + \cdots + X_n. \tag{3.8}$$

♢

The definition does not restrict the random walk. As in the last chapter we may have states that are absorbing. Often we have finite lattices, where the boundary consists of absorbing states. Figure 3.4 shows a small 2-dimensional random walk setup on a finite lattice.

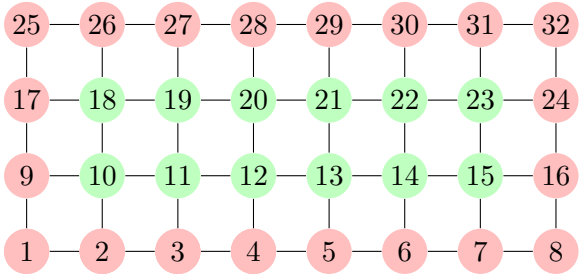

Figure 3.4: Two-dimensional lattice random walk, with boundary states B and interior states D in red and green shading.

We denote B the boundary states and D are the interior states. The quantity of all potential states of the random walk is defined as $S = B \cup D$. As in the one-dimensional setting, we write the probability to be at state $i = \{k, \ell\}$ at time n as $p_{\{k,\ell\}}(n)$ recursively as

$$p_{\{k,\ell\}}(n) = \frac{1}{4} \cdot \big(p_{\{k-1,\ell\}}(n-1) + p_{\{k+1,\ell\}} + p_{\{k,\ell-1\}}(n-1) + p_{\{k,\ell+1\}}(n-1)\big). \tag{3.9}$$

The problem is said to be harmonic if an averaging property such as in equations (3.2) and (3.9) is satisfied. Finding solutions to a harmonic function given boundary conditions is a Dirichlet problem.

Harmonic problems can also be solved using a *method of relaxation*: an iterative approach that adjust interior points of the lattice according to the average over its neighbors. We then iterate over a predefined sequence of interior points until convergence.

3.2.2 Transition Matrix Approach

In the transition matrix approach, we write all potential values of $S = B \cup D$ with cardinality of S being K into a matrix $\mathbf{P} \in \mathbb{R}^{K \times K}$ that corresponds to a 1-step random walk. In the matrix $\mathbf{P} = (p_{ij}) = \mathrm{P}(S_{n+1} = j \mid S_n = i) \in \mathbb{R}^{K \times K}$ are the transition probabilities. We order the states such that we first have the absorbing (boundary) states then the interior ones. The matrix $\mathbf{P}$ is then written in terms of a 2×2 block matrix. We denote the submatrix corresponding to steps from D to B as $\mathbf{R}$ and the submatrix corresponding to steps from D to D as $\mathbf{Q}$. Since the boundaries are absorbing, the B to B submatrix is the identity matrix and the B to D submatrix is equal to $\mathbf{0}$ (see (3.5)).

Because we have an harmonic function, we have $\boldsymbol{f} = \mathbf{P}\boldsymbol{f}$, and thus $\boldsymbol{f} = \mathbf{P}^n\boldsymbol{f}$, $n = 1, 2, \ldots$. Straightforward arguments show that

$$\lim_{n\to\infty} \mathbf{Q}^n = \mathbf{0}, \tag{3.10}$$

$$\sum_{i=0}^{\infty} \mathbf{Q}^i = (\mathbf{I} - \mathbf{Q})^{-1}, \tag{3.11}$$

$$\lim_{n\to\infty} \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{R} & \mathbf{Q} \end{pmatrix}^n = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ (\mathbf{I} - \mathbf{Q})^{-1}\mathbf{R} & \mathbf{0} \end{pmatrix}. \tag{3.12}$$

Thus we have again $\mathbf{B} = (\mathbf{I} - \mathbf{Q})^{-1}\mathbf{R}$ as derived in the last section.

Classify all boundary states according to two a zero and one scheme ($B = B_0 \cup B_1$) and let $\boldsymbol{b}$ be a column vector containing zero and ones only according to the same scheme. Hence, the probability of reaching B_1 before B_0 when starting from an interior point i is the i element of $\mathbf{B}\boldsymbol{b}$.

3.3 Bibliographic Remarks

Details about the airplane accident used for the illustration can be found at en.wikipedia.org/wiki/British_Airtours_Flight_28M. As a side note, the was a similar accident with same aircraft type, en.wikipedia.org/wiki/Pacific_Western_Airlines_Flight_501, however no lives were lost: “Almost all the passengers were frequent air travellers familiar with the Boeing

737. This contributed to the success of the evacuation. [...] The evacuation was completed with little time to spare." (https://www.webcitation.org/6aBxvzm0W).

The corresponding Mathematica script illustrates that it is in general not possible to solve the two dimensional difference equation in a similar fashion as in the one dimensional case.

Lab Content

1. Fundamental matrix, properties of the matrices;
2. Expectation of hitting time;
3. Mathemtica script.

Worksheet 3

7. Describe a tennis game using a random walk of order 1.
8. Determine the number of airplane emergency exits with the help of a random walk of order 2.
9. Show that a random walk of order 1 with infinite locations is not expected to return to its origin.

Chapter 4

Discrete Time Markov Processes

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta111/chapter04.R.

Mathematica file: www.math.uzh.ch/furrer/download/sta111/chapter04.nb.

In the first few chapters we have extensively used iid random sequences. When working with random walks we have summed such sequences. The resulting sequence $\{S_i, i = 1, \ldots, n\}$, is of course not independent, conditionally $\{S_i \mid S_{i-1}, i = 2, \ldots, n\}$ it is. In this chapter, we will formalize this concept further.

4.1 Weather Prediction

By "nature", weather, or more general the atmospheric state, cannot change arbitrarily within short time frames. Current condition have an effect on future states, but far past states are not relevant to predict the future, provided current state is known.

We look at a very simple weather prediction system with four states. We classify each day according to the total (relative) amount of sun shine and the total amount of precipitation. Hence we have the four states (sunny and dry), (sunny and not dry) (not sunny and dry) and (not sunny and not dry). Figure 4.1 illustrates the four states in scatterplot of precipitation versus sunshine (relative to the max possible duration) for weather data in 2013 at Zurich Fluntern. R-Code 4.1 illustrates the analysis, including a couple simple weather forecasts.

R-Code 4.1: Simple weather prediction (See Figure 4.1.)

```
sma <- read.csv("data/bodenstationsdaten_dbformat.csv", header=TRUE)
dry <- (sma$rre150d0 <= .1)     # classical definition
sunny <- (sma$sremaxdv > 50)    # arbitrary threshold
with( sma,  plot( rre150d0~sremaxdv, col=1+dry+2*sunny))
table( data.frame(sunny,dry))
```

```
##        dry
## sunny   FALSE TRUE
##   FALSE   152   91
##   TRUE     25   97
legend( 'topright', col=1:4,legend=c('wet','cloudy','mostlysunny',
                                     'sunnydry'), pch='o', bty='n')
state <- cbind(wet=(!sunny & !dry), cloudy=(!sunny & dry),
               mostlysunny=(sunny & !dry), sunnydry=(sunny & dry))
head( state, n=3)
##        wet cloudy mostlysunny sunnydry
## [1,]  TRUE  FALSE       FALSE    FALSE
## [2,] FALSE   TRUE       FALSE    FALSE
## [3,]  TRUE  FALSE       FALSE    FALSE

# P( A and B ), i.e., first day in B then in A
P <- ( t( state[-365,]) %*% state[-1,] ) # B = "-365"
sum(P)/364   # one day missing, is ok!
## [1] 1
P <- (t( state[-365,]) %*% state[-1,]/apply( state[-365,],2,sum)) # P(A|B)
P      # Proper transition matrix
##                 wet  cloudy mostlysunny sunnydry
## wet         0.60526 0.21711    0.065789  0.11184
## cloudy      0.31868 0.41758    0.032967  0.23077
## mostlysunny 0.66667 0.16667    0.041667  0.12500
## sunnydry    0.14433 0.16495    0.113402  0.57732
# What is the weather prediction?
Pm <- P %*% P  %*% P  %*% P  %*% P   # forecast for 5 days for a wet day
Pm['wet',]
##         wet      cloudy mostlysunny    sunnydry
##    0.421414    0.251335    0.067812    0.259439
p <- runif(4)           # We generate some random weather
(p <- p/sum(p))         #     ... need to normalize the vector.
## [1] 0.301488 0.410973 0.072427 0.215111
p %*% P                 # Prediction for next day
##          wet  cloudy mostlysunny sunnydry
## [1,] 0.39278 0.28462    0.060795   0.2618
p %*% Pm                # 5-days
##          wet  cloudy mostlysunny sunnydry
## [1,] 0.41452 0.24982    0.068663    0.267
```

```
p %*% Pm %*% Pm %*% Pm  # 15 days = "climatology"
##          wet  cloudy mostlysunny sunnydry
## [1,] 0.41504 0.24983    0.068623  0.26651
p <- eigen(t(P))$vectors[,1]
(p <- p/sum(p))  # "climatology" again...
## [1] 0.415037 0.249829 0.068623 0.266512
#  naturally, does not change when post multiply with P or Pm
```

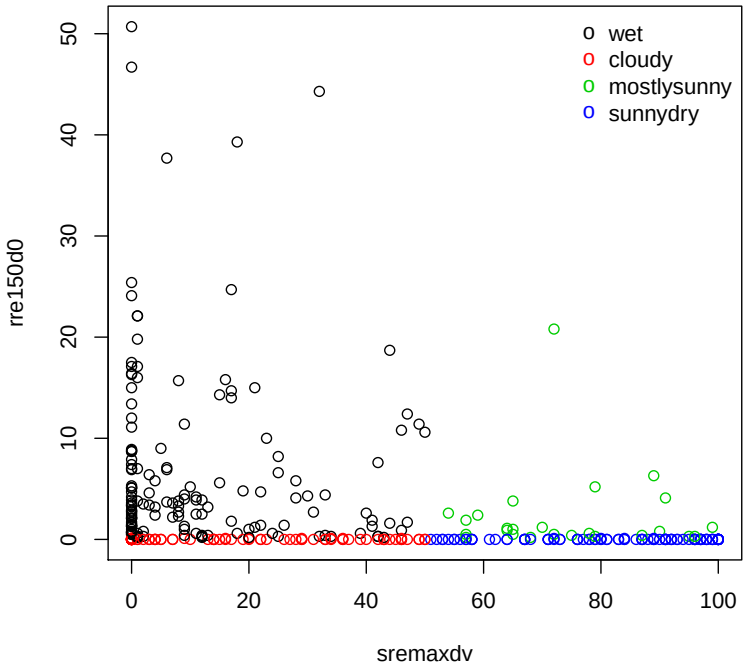

Figure 4.1: Daily precipitation (mm) versus relative sunshine duration in 2013 at SMA Zurich/Fluntern 8°34'/47°23'. (See R-Code 4.1.)

The R-Code shows that with a simple transition model, we can better predict compared to baseline models like random choice, most frequent state or even today's state.

There is strong doubt that wetter pattern changes behave constantly over the year. A natural extension would be the inclusion of non-homogeneous transition probabilities, for example. Such models can be fitted using a likelihood approach. More formally, starting from a particular state, we use a multinomial model with parameterized transition probabilities.

As illustrated in the script, more complex models are not necessary here (based on AIC or likelihood ratio tests). Further improvement would include different classification of the states, including modeling different types of precipitation.

4.2 Markov Property

We now formally define the Markov property for a discrete state space, e.g., $S = \{1, \dots, K\}$, K possibly infinite.

Definition 4.1. Let $\{X_n, n = 0, 1, \dots\}$ a stochastic process where each of X_n is a discrete random variable. We say that X_n satisfies the Markov assumption (Markov property) if

$$\mathrm{P}(X_{n+1} = j \mid X_n = i_n, X_{n-1} = i_{n-1}, \dots, X_0 = i_0) = \mathrm{P}(X_{n+1} = j \mid X_n = i_n) \tag{4.1}$$

♢

In other words the probability that at time $n + 1$ we are in state j depends solely on the state at time n.

If the probability does not depend on n, i.e., only depends on i and j, the Markov chain is said to be homogeneous.

4.2.1 Transition Probabilities

Because of the Markov assumption, it is intuitive that the description of a Markov process is based on transition probabilities, i.e., $p_{ij}(n) := \mathrm{P}(X_{n+1} = j \mid X_n = i)$. For a homogeneous chain we omit the the functional dependence on n and simply write $p_{ij} := \mathrm{P}(X_{n+1} = j \mid X_n = i)$.

In case we have a finite state space S, i.e., cardinality of S is $K < \infty$, we typically write the transition probabilities in form of a matrix: $\mathbf{P}(n) = \big(p_{ij}(n)\big) \in \mathbb{R}^{K \times K}$. For homogeneous chain we also use the term stationary transition probabilities $\mathbf{P} = \big(p_{ij}\big) \in \mathbb{R}^{K \times K}$.

The transition matix $\mathbf{P}$ is a stochastic matrix,

- all entries are non-negative,
- row sums are equal to 1,

as all entries are probabilities.

Note that stochastic matrices have at least one (right) eigenvalue equal to one ($\mathbf{P1} = \mathbf{1}$) and positive natural powers of such matrices remain stochastic.

4.2.2 n-Step Transition Probabilities

Now lets look at probabilities if we take multiple steps. The basic idea is that we use (again) fist step analysis: The probability that we change from i to j in exactly n steps is the same as the product of the probability to fist advance one step from i to k and then the probability to change from k at j in $n - 1$ steps:

$$P_{ij}^{(n)} = \sum_{k=1}^{K} P_{ik}^{(1)} P_{kj}^{(n-1)} \tag{4.2}$$

As shown in Worksheet 4, the following is also true (Chapman–Kolmogorov equations):

$$P_{ij}^{(n)} = \left(P_{ij}^{(1)}\right)^n = \left(P_{ij}^{(k)}\right)\left(P_{ij}^{(n-k)}\right) \tag{4.3}$$

$$\mathbf{P}^n = \mathbf{P}^k \mathbf{P}^{n-k} \qquad k = 1, 2, \dots, n-1. \tag{4.4}$$

More specifically, the n-step transition matrix is equivalent to the nth power of a 1-step transition matrix.

Starting from an initial distribution, $\boldsymbol{p}(0) = \big(\mathrm{P}(X_0 = 1), \mathrm{P}(X_0 = 2), \ldots, \mathrm{P}(X_0 = K)\big)$ (row vector!), Using this, we can also calculate the state distribution after any number of steps

$$\boldsymbol{p}(n) = \boldsymbol{p}(0)\mathbf{P}^n = \boldsymbol{p}(0)\mathbf{P}^{(n)}. \tag{4.5}$$

4.3 Classification of States

Many properties of the chain are set if we know properties of individual states. For example, if we have a simple random walk in one dimension with at least one absorbing state, then sooner or later, we are trapped in that state.

Definition 4.2. We say that a state j is accessible from state i and write $i \to j$, if $(\mathbf{P}^\ell)_{ij} > 0$ for some $\ell > 0$.

If $i \to j$ and $j \to i$ then the two states communicate and write $i \leftrightarrow j$. ♢

The accessibility relation divides states into different classes. Within each class, all states communicate to each other, but no pair of states in different classes communicates.

A chain is irreducible if there is one single class. If the chain has K states, irreducibility means that the entries of the matrix $(\mathbf{I} + \mathbf{P} + \cdots + \mathbf{P}^K)$ are all nonzero.

Let $f_i = P(\text{ever reenter to state } i \mid \text{chain started at } i)$.

Definition 4.3. We call the a state i recurrent if $f_i = 1$, and transient if $f_i < 1$. ♢

In other words, transient states are states within a set that are reached by any other state in the set but the set can be possibly left (and never reached again). A recurrent state will be revisited as time moves on.

It can be shown that for a recurrent state $\sum_{n=1}^{\infty}(\mathbf{P}^n)_{ii} = \infty$ (if and only if).

4.4 Stationarity

We have seen stationary transition probabilities (i.e., homogeneous Markov chains). There is another type of stationarity, i.e., stationarity with respect to the distribution of X_n over n.

Definition 4.4. The distribution $\boldsymbol{\pi}$ is said to be stationary if $\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P}$ and all states of the chain are recurrent ♢

In other words, a stationary distribution is a left eigenvector of the transition matrix. A stationary distribution exists if the transition matrix has an eigenvalue 1 associated with a left eigenvector.

A priori, it is not clear that such a stationary distribution exists. Consider the stochastic system as illustrated in Figure 4.2, with the transition matrix

$$\mathbf{P} = \begin{pmatrix} 0 & 1 & 0 \\ p & 0 & 1-p \\ 0 & 0 & 1 \end{pmatrix}. \tag{4.6}$$

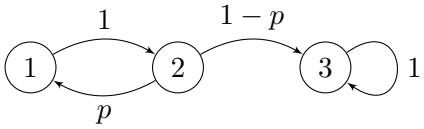

Figure 4.2: Simple three state system.

It is possible to go back and forth between the states 1 and 2, but once you are in state 3, you remains in that state forever (absorbing). In other words, 1 and 2 communicate with each other, but 3 does not communicate with neither 1 nor 2. Let $\boldsymbol{\pi} = (a, b, c)$, then

$$(a, b, c) = \boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P} = (bp, a, (1-p)b + c), \tag{4.7}$$

which has multiple solutions:

- $a = b$, $p = 1$. This implies that there is no movement between 2 and 3, the chain is reducible and we have two separated systems.
- $b = a = 0$. This is a trivial solution where we always start at 3, where we obviously stay.
- $p = 0$, $a \neq b$. Although we will end up in state 3 after at most 2 steps, $\boldsymbol{\pi}$ is not stationary. $p = b = a = 0$ is.

If $0 < p < 1$ we do not have a stationary solution but we can further describe the states. However, we give here only a very coarse classification of states.

R-Code 4.2 gives a framework to study the limiting distribution(s) for random walks on $\{1, \ldots, K\}$, with reflecting (*type=1*) and absorbing (*type==2* end points. Only the former has a unique stationary distribution for $0 < p < 1$. The setting *type==3* represents a reducible chain, that could be analyzed by considering the chains $\{1, \ldots, K-1\}$ and $\{K\}$ separately.

R-Code 4.2: Different transition matrices and limiting distributions.

```
Ptypes <- function(p, K=4, type=1) {
  P <- matrix( 0, K, K)
  P[ cbind(1:(K-1), 2:K)] <- p
  P[ cbind(2:K, 1:(K-1))] <- 1-p
  if (type==1) {            # both reflecting
      P[ cbind(c(1, K), c(1, K))] <- c( 1-p, p)
  } else if (type==2){      # both absorbing
      P[1, 1] <- P[K, K] <- 1 ;      P[1, 2] <- P[K, K-1] <- 0
  } else {                  # reducible
      P[ cbind(c(1, K-1), c(1, K-1))] <- c( 1-p, p)
      P[ K, K] <- 1 :                P[ K-1, K] <- P[ K, K-1] <- 0
  }
  P
```

```
}

print( P <- Ptypes(p=1/4, type=1))
##      [,1] [,2] [,3] [,4]
## [1,] 0.75 0.25 0.00 0.00
## [2,] 0.75 0.00 0.25 0.00
## [3,] 0.00 0.75 0.00 0.25
## [4,] 0.00 0.00 0.75 0.25
eigP <- eigen( t(P))
cat( 'Eigenvalues:', round( eigP$val, 5))
## Eigenvalues: 1 0.61237 -0.61237 0
unnorm <- eigP$vect[, abs( eigP$val-1) < 1e-14]
print( unnorm/sum( unnorm))
## [1] 0.675 0.225 0.075 0.025
Pn <- P %*% P
for (i in 1:100)     Pn <- Pn %*% P
round( Pn, 5)
##       [,1]  [,2]  [,3]  [,4]
## [1,] 0.675 0.225 0.075 0.025
## [2,] 0.675 0.225 0.075 0.025
## [3,] 0.675 0.225 0.075 0.025
## [4,] 0.675 0.225 0.075 0.025
```

4.5 Bibliographic Remarks

The weather data used has been downloaded from https://shop.meteoswiss.ch/productView.html?type=psc&id=17

If the weather forecast constructed here is not sufficiently precise, you may be interested in learning more about "read" meteorologists https://de.wikipedia.org/wiki/Innerschwyzer_Meteorologen and in following "real" forecast https://wetterpropheten.ch/prognosen.html.

Lab Content

1. Classification of states (Definitions are given below).
2. Recurrence in higher dimensions, result only.
3. Some results about regular chains: (i) $\mathbf{P}$ transition matrix: $\lim_{n\to\infty} \mathbf{P}^n = \mathbf{A} = \mathbf{1}\boldsymbol{a}^\top$. (ii) Fundamental matrix for regular chains.

The following is a detailed list of important definitions, taken virtually verbatim from Kemeny *et al.* (1960). The sets are to be understood as maximal sets, i.e., equivalence classes.

Definition 4.5. 1. A *finite homogeneous Markov chain* is a stochastic process which moves through a finite number of states, and for which the probability of entering a certain state depends only on the last state occupied.

2. An *ergodic set* of states is a set in which every state can be reached from every state, and which cannot be left once it is entered.

3. An *ergodic state* is an element of an ergodic class.

4. A *transient set* of states is a set in which every state can be reached from every other state, and which can be left.

5. A *transient state* is an element of a transient set.

6. An *absorbing state* is a state which once entered is never left.

7. An *absorbing chain* is a chain which has at least one absorbing state, and such that an absorbing state can be reached from every state.

8. An *ergoic chain* is a chain in which every state can be reached from every state.

9. A *cyclic chain* is an ergodic chain in which each state can only be entered a certain periodic intervals.

10. A *regular chain* is an ergodic chain that is not cyclic.

Worksheet 4

10. Chapmann–Kolmogorov equations;

11. Weather prediction with the help of a state diagram;

12. Classification of Markov chains and states.

Chapter 5

Queues

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta111/chapter05.R.

Mathematica file: www.math.uzh.ch/furrer/download/sta111/chapter05.nb.

Queuing theory has widespread applications and is an important tool for management and optimization purposes. We introduce general concepts of queuing theory which provides (theoretical) tools for the design and for the quantitative analysis of service systems.

5.1 Airport Management

Within an airport there are many different queues and serving systems set in place. The study of a subsystem thereof is often quite complicated due to missing data (not recorded or not disclosed). Individual stakeholders have different objectives which drive optimization procedures. Hence, there is a vast playground for statistical analysis and stochastic simulation.

R-Code 5.1 analyzes the number of passengers processed on flights arriving in Denver International Airport (DIA, IATA airport code) each hour based on how long it took for those passengers to clear Passport Control in 2017 (the link to the chapter script gives a more detailed analysis and many more figures). Much of the understanding is based on a careful EDA, followed by a careful choice of specific sub-models.

In the setting here, we can assume that the (mean) waiting time is somewhat independent of the number of open booths (provided the number is between 4 to 10). The max waiting time is increasing with number of booths and total passengers arriving. The time is the total time, spend in the queue and at the booth. We cannot not discriminate the two times reliably.

For simplicity, a potential analysis can restriction to hours with a fixed number of open booths. The total service capacity is very close to incoming frequency. Possibly the time of empty queue can be neglected.

R-Code 5.1: Waiting times at Denver International Airport (See Figure 5.1.)

```
# Data available from awt.cbp.gov:
load( file="data/DIA_waiting_awt_cbp_gov.RData") # Data from awt.cbp.gov

# Total travelers and sum of boots over time.
plot(total~day, data=dia.daily, type='l', ylab='total passengers', xaxt='n', xlab='')
abline(v=dia.daily$day[c(106,149,185,247,327)], col=3)
# Indication of Easter, Memorial day, 4. July, Labor day and Thanksgiving.
par(new=T)
plot(booths~day, data=dia.daily, type='l',col=4,yaxt='n',ylab='',xaxt='n',xlab='')
axis(4,col=4, col.axis=4)
mtext(side=4, line=2.6, col=4, "total booth hours")
mtext(side=1, line=1, "Day of 2017") # => weekly cycle, annual cycle visible.
# Follow-up: did waiting time increase over time?
plot(mean~day, data=dia.daily, type='l', ylab='sum of mean waiting', xaxt='n', xlab='')
abline(v=dia.daily$day[c(106,149,185,247,327)], col=3)
par(new=T)
plot(max~day, data=dia.daily, type='l',col=4,yaxt='n',ylab='',xaxt='n',xlab='')
axis(4,col=4, col.axis=4)
mtext(side=4, line=2.3, col=4, "sum of max waiting")
mtext(side=1, line=1, "Day of 2017")
# virtually constant (of course significant but not relevant)

boxplot( total~tod, data=DIA, ylab="Total passengers")
boxplot( mean~tod, data=DIA, ylab="Mean waiting time")
# Waiting times vers booths
boxplot( max~booths, data=DIA, ylab="Max waiting time", xlab="Open booths")
boxplot( DIA$mean~weekdays( as.Date(DIA$Doy), abbr=TRUE),
                                  ylab="Mean waiting time")

plot( maxNonUS~maxUS, data=DIA, col=4, cex=.6)
abline( 0,1)
points( meanNonUS~meanUS, data=DIA, col=7, cex=.5)
abline( lm(meanNonUS~meanUS, data=DIA), col=7)
abline( lm(maxNonUS~maxUS, data=DIA), col=4)
# Waiting times as a function of open booths:
waitingtime <- DIA$meanUS[DIA$booths==5]
hist( waitingtime, prob=TRUE, br=25, xlim=c(0,max(waitingtime)), main='')
for (i in 3:10)
  lines( density( DIA$meanUS[DIA$booths==i & !is.na(DIA$meanUS)]), col=i-2)
```

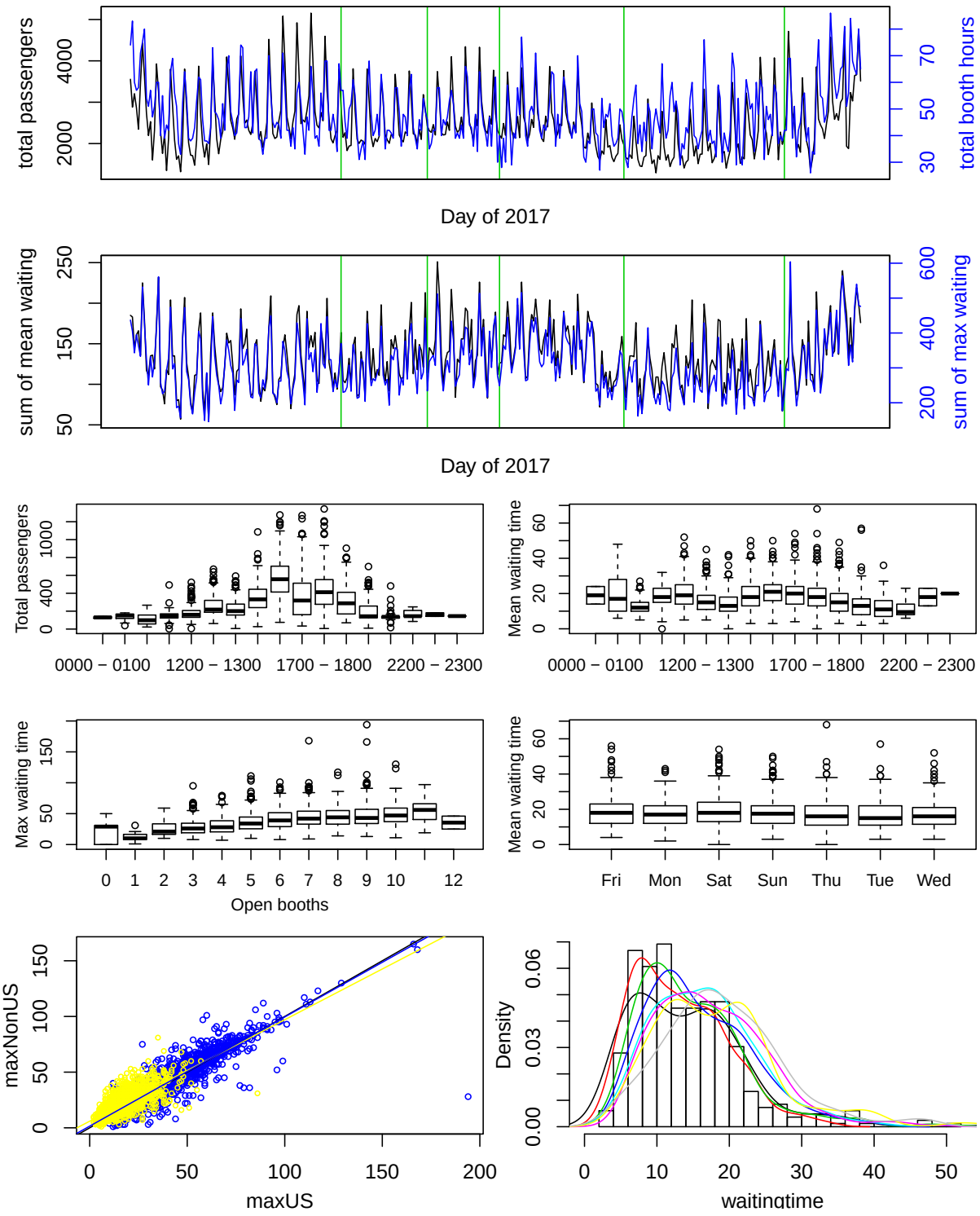

Figure 5.1: Top two panels: daily summaries over the year 2017. Center: boxplots for different hourly values over time of day, number of open booths or weekdays. Bottom left: waiting times (average yellow and max blue) for non-US and US citizens. Bottom right: waiting times stratified according to open booths. (See R-Code 5.1.)

5.2 Basic Terminology

We consider the situation where a customer arrives at a service system where he has to (possibly) wait in a queue until he gets served. Such a simple queuing system consists of an arrival process (modeling the arrival times of the customers), serving facilities (modeling serving time) and a queuing discipline (describing how customers within the queue are selected for service).

For simplicity, we talk about customers, service desks and task even if we do not have explicit customers, desks/servers and tasks.

We use Kendall notation to specify arrival process, serving facilities components of the system. A queuing system is denoted A/B/c/d, where A is the distribution of the inter arrival times (specifies the arrival process), B is the distribution of the serving time, c number of servers and d is the capacity of the queue.

If we have a queuing capacity, we need to specify if we serve according to first in-first out (FIFO), last in-first out (LIFO) or some random order. In a priority system, the customers are served according to their priorities (possibly with interruptions of current customers (*preemptive priority discipline*). Kendall's notation does not specify the queuing discipline.

We denote the arrival time the time a customer arrives at the queue. The waiting time is the time between arrival and start of the processing at the desk. The serving time is the time spent at the desk. The time between two consecutive arrivals is called the inter-arrival time. We assume that two task will never arrive at the same time (almost surely). Often the waiting time and serving time is denoted as the sojourn time. Figure 5.2 illustrates one possibility of visualizing the system. Trivially, increasing the number of service desks decreases the total waiting time. Moreover, the single desk setup does not have any capacities (empty desks), the double desk setup has for more than half the time capacities.

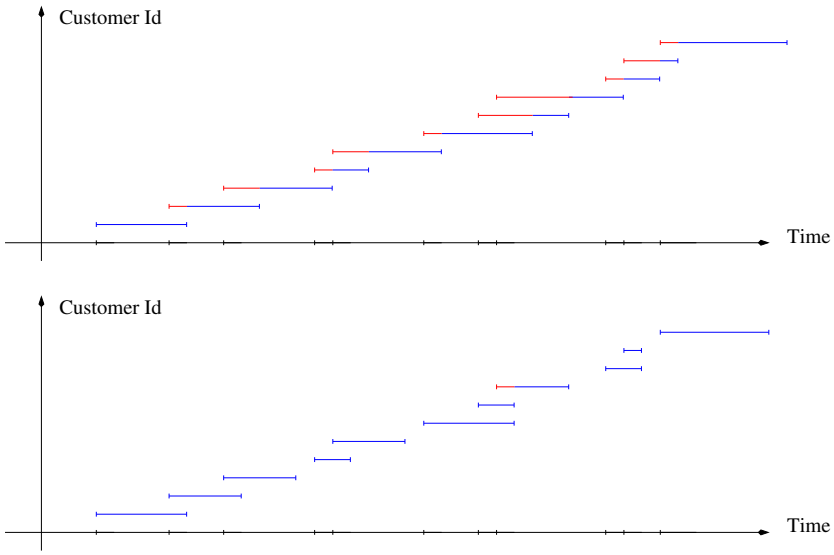

Figure 5.2: Single (top) and double (bottom) desk setup. The customers are ordered according to arrival times (which are identical in both settings). Waiting and serving times are indicated in read and blue, respectively.

We often assume that the inter-arrival time is exponentially distributed (Kendall's notation: A=M, M for Markov). Similarly, we assume that we have exponential service times (Kendall's notation: B=M). In a M/M/$\cdot$/$\cdot$ system, the customers are arriving independently with rate λ and the rate per desk is the constant μ. We further assume that arrival and service times are independent.

In analogy to the previous chapters, we can summarize the system with the number of customers in the system: upon an arrival of a task we move from state k to $k+1$ and upon finishing a service, we move from state k to $k-1$. The movements are not at integer times but according to some distribution spread in time. Assuming inter-arrival times according to an exponential distribution makes the calculation of relevant summaries tractable.

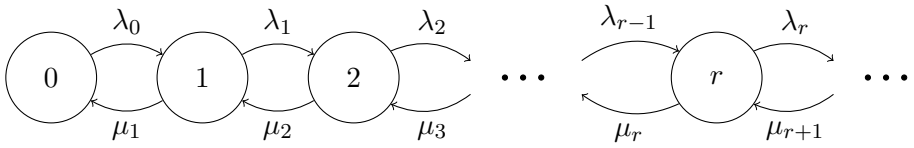

Figure 5.3: Transition graph with rates between the individual states.

Here is a first glance at a solution to a queuing problem (we will generalize and formalize it later). As above, we assume stationary (homogeneous) arrival and service processes, that means the rates λ and μ are constant over time. If a stationary solution exists, then the rate of incoming is equal to the rate of outgoing. We illustrate the situation in Figure 5.3.

We now derive a stationary solution and for the sake of understanding but also completeness, we will denote the rates individually for each state, i.e., λ_k and μ_k for state k. We assume steady state and define $p_k = p_k(t)$ as the probability that the system is in state k. Thus for state zero we have

$$\lambda_0 p_0 = \mu_1 p_1 \tag{5.1}$$

and for an arbitrary state $r = 1, 2, \ldots$

$$\lambda_{r-1} p_{r-1} + \mu_{r+1} p_{r+1} = \mu_r p_r + \lambda_r p_r \,. \tag{5.2}$$

Thus

$$p_1 = \frac{\lambda_0}{\mu_1} p_0 \,, \tag{5.3}$$

$$p_2 = \frac{\lambda_1}{\mu_2} p_1 = \frac{\lambda_1\, \lambda_0}{\mu_2\, \mu_1} p_0 \,, \tag{5.4}$$

$$\vdots \tag{5.5}$$

$$p_k = \frac{\lambda_{k-1} \ldots \lambda_1\, \lambda_0}{\mu_k \ldots \mu_2\, \mu_1} p_0 \,. \tag{5.6}$$

$$\vdots \tag{5.7}$$

As p_k are probabilities, we have the additional condition $\sum_{k=0}^{\infty} p_k = 1$ and thus we

$$p_0 = \Big(1 + \sum_{j=1}^{\infty} \prod_{i=1}^{j} \frac{\lambda_{i-1}}{\mu_i}\Big)^{-1}. \tag{5.8}$$

Note that we require additional constraints on the rates in order that the limit exists. (More specifically there exists a finite N, such that $\lambda_k/\mu_{k+1} < 1$, for all $k > N$.) For a more detailed discussion on necessary and sufficient conditions for the existence of a stationary solution see Karlin and Taylor (1981).

5.3 Some Simple Queuing Systems

Let $X(t)$ be the total number of customers in the system (either waiting in the queue or being served at a desk).

We denote $\rho = \lambda/\mu$ the traffic intensity which measures the ratio between the input rate and output rate. Of course if the input rate is larger than the output rate, the queue starts to grow. Here, we will assume that the traffic intensity is smaller than the number of desks, i.e., $\rho < s$.

5.3.1 M/M/s/0

An example of a system with no queue is a telephone system with s servers. Formally, we have the rates

$$\lambda_i = \lambda, \qquad i = 0, \dots, s-1, \qquad\qquad \lambda_i = 0, \qquad i \geq s, \tag{5.9}$$

$$\mu_i = i\mu, \qquad i = 1, \dots, s. \tag{5.10}$$

By using these rates with the above formulas we have

$$\text{vacant probability: } \mathrm{P}(X(t) = 0) = \Big(\sum_{i=0}^{s} \frac{\rho^i}{i!}\Big)^{-1} =: p_0\,, \tag{5.11}$$

$$\mathrm{P}(X(t) = k) = \frac{\rho^k}{k!} p_0 =: p_k, \qquad k = 1, \dots, s, \tag{5.12}$$

$$\text{loss probability: } \mathrm{P}(X(t) = s) = \frac{\rho^s}{s!}\,\mathrm{P}(X(t) = 0) =: p_s\,, \tag{5.13}$$

and, of course, $\mathrm{P}(X(t) > s) = 0$. Using these identities, it is straightforward to calculate the *mean number of busy servers*, $\mathrm{E}\big(S(t)\big)$:

$$\mathrm{E}\big(S(t)\big) = \sum_{i=0}^{s} i\,\mathrm{P}(X(t) = i) = \sum_{i=1}^{s} i \frac{\rho^i}{i!} p_0 = \rho \sum_{j=0}^{s-1} \frac{\rho^j}{j!} p_0 = \rho \sum_{j=0}^{s-1} p_j = \rho(1 - p_s)\,. \tag{5.14}$$

The server utilization is then $\mathrm{E}\big(S(t)\big)/s = \rho(1 - p_s)/s$. In the setting of an M/M/s/0 queue, $\mathrm{E}\big(S(t)\big) = \mathrm{E}\big(X(t)\big)$.

Of course with $s = 1$, much of the formulas simplify, specifically

$$p_0 = \frac{1}{1+\rho}\,, \qquad p_1 = \frac{\rho}{1+\rho}\,. \tag{5.15}$$

It is important to know that if the service times are iid according an "arbitrary" distribution, B = G, the results of this section remain valid. That is, if the traffic intensities in a M/M/s/0 and M/G/s/0 are ρ then the stationary distribution remains the same.

5.3.2 M/M/1/∞

We now introduce an "infinite" queuing system and thus change the processing rates from (5.10) to

$$\lambda_i = \lambda, \qquad i = 0, 1, \ldots, \tag{5.16}$$

$$\mu_i = \mu, \qquad i = 1, 2, \ldots. \tag{5.17}$$

and the stationary distribution $p_k = \big(1 - \frac{\lambda}{\mu}\big)\big(\frac{\lambda}{\mu}\big)^k = (1-\rho)\rho^k$, i.e., an geometric random variable with parameter ρ.

$$\text{E(number of customers in system)} = \mathrm{E}\big(X(t)\big) = \frac{\lambda}{\mu - \lambda} = \frac{\rho}{1-\rho}\,, \tag{5.18}$$

$$\text{E(number of customers waiting)} = \frac{\lambda}{\mu - \lambda}\,, \tag{5.19}$$

$$\text{P(server idle)} = p_0 = 1 - \rho\,. \tag{5.20}$$

The waiting time for a customer consists of the sum of exponential random variables that are in the queue and thus a gamma random variable.

5.3.3 M/M/s/∞

Analogue to the last section we change the rates from (5.9) and (5.10) to

$$\lambda_i = \lambda, \qquad i = 0, 1, \ldots, \tag{5.21}$$

$$\mu_i = i\mu, \qquad i = 1, \ldots, s. \qquad\qquad \mu_i = s\mu, \qquad i \geq s. \tag{5.22}$$

To guarantee manageable queues, we assume $\rho = \lambda/\mu < s$. Replacing these rates into equations (5.3) to (5.6) leads to

$$p_i = \frac{\rho^i}{i!} p_0, \qquad\qquad i = 0, \ldots, s-1, \tag{5.23}$$

$$p_i = \frac{\rho^i}{s! s^{i-s}} p_0, \qquad i \geq s. \tag{5.24}$$

Summing the probabilities and solving for p_0 (geometric series) yields

$$\text{vacant probability:} \qquad \mathrm{P}(X(t) = 0) = p_0 = \Big(\sum_{i=0}^{s-1} \frac{\rho^i}{i!} + \frac{\rho^s}{(s-1)!(s-\rho)} \Big)^{-1}. \tag{5.25}$$

The probability that an arriving customer has to wait is

$$\text{waiting probability:} \qquad \sum_{i=s}^{\infty} \mathrm{P}(X(t) = s) = \frac{p_s}{1 - \rho/s} =: p_w. \tag{5.26}$$

Other interesting summaries are *mean number of busy servers*, $\mathrm{E}\big(S(t)\big)$, and a *mean queue length*, $\mathrm{E}\big(L(t)\big)$, given by

$$\mathrm{E}\big(S(t)\big) = \sum_{i=0}^{\infty} \min(i, s)\, \mathrm{P}(X(t) = i) = \sum_{i=0}^{s-1} i p_i + s \cdot p_w = \rho\,. \tag{5.27}$$

$$\mathrm{E}\big(L(t)\big) = \sum_{i=s}^{\infty} (i - s)\, \mathrm{P}(X(t) = i) = \sum_{i=s}^{\infty} i p_i - s \frac{p_s}{1 - \rho/s} = \frac{\rho s\, p_s}{(s-\rho)^2}\,. \tag{5.28}$$

It is even possible to derive the *waiting time distribution* $F_W(t)$ (see, e.g., Beichelt, 2006) and the mean waiting time:

$$F_W(t) = 1 - \frac{p_s}{1-\rho/s}\exp(-\mu(s-\rho)t) = 1 - p_w \exp(-\mu(s-\rho)t), \qquad t \geq 0. \tag{5.29}$$

Note that $F_W(0) = 1 - \frac{p_s}{1-\rho/s} = 1 - p_w$. Hence $W(t)$ has a point mass at zero. Conditional on the fact that we need to wait, the waiting time is exponential. Using the law of total expectation, we can directly calculate the mean waiting time:

$$\begin{aligned} \mathrm{E}\big(W(t)\big) &= \mathrm{E}\big(W(t) \mid W(t) > 0\big) \cdot \mathrm{P}\big(W(t) > 0\big) && (5.30)\\ &= \frac{1}{\mu(s-\rho)} \cdot p_w = \frac{\lambda\rho}{(s-\rho)} \cdot \frac{p_s}{1-\rho/s} = \frac{\rho s\, p_s}{\lambda(s-\rho)^2}. && (5.31) \end{aligned}$$

Equations (5.28) and (5.31) also bring forth *Little's law*:

$$\mathrm{E}\big(L(t)\big) = \lambda\, \mathrm{E}\big(W(t)\big). \tag{5.32}$$

which is illustrated in Figure 5.4 based on total customers and total time in the system

$$\mathrm{E}\big(X(t)\big) = \mathrm{E}\big(L(t)\big) + \mathrm{E}\big(S(t)\big) = \lambda\, \mathrm{E}\big(W(t)\big) + \rho = \lambda\Big(\mathrm{E}\big(W(t)\big) + \frac{1}{\mu}\Big) = \lambda\, \mathrm{E}\big(T(t)\big), \tag{5.33}$$

where $T(t)$ is the total sojourn time.

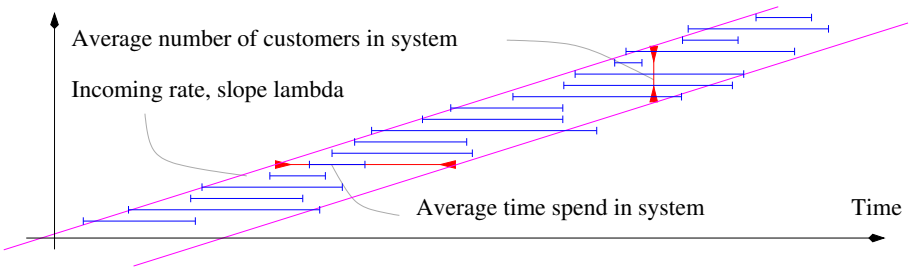

Figure 5.4: Sketch of Little's law.

5.4 Bibliographic Remarks

Browsing the web, one finds quite a few documents illustrating queuing theory at airports. Wang *et al.* (2009); Mueller and Chatterji (2002) and Sankaranarayanan *et al.* (2016) are accessible and cover much of the underlying concepts. As for data, the pages `https://awt.cbp.gov/` and `https://pqt.cbp.gov/` give waiting times at major US airports and waiting times at preclearance airport locations.

Beichelt (2006) gives an accessible overview of queuing theory, including more specific examples of queuing systems.

Lab Content

Interactive R session visualizing waiting times at JFK.

Worksheet 5

13. Simulation of a M/M/1 system;

14. Memoryless property of exponential random variables;

15. Simple post office example.

Chapter 6

Birth and Death Processes

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta111/chapter06.R.

6.1 Modeling populations

In this chapter we model populations through increase (birth or immigration) or decrease (death or emigration). As in the last chapter, events may happen at any time (continuous time) but an event leads a discrete change (± 1 individual). After some introductory examples, we first look at a simple birth process and then extend to more complex models.

6.1.1 Baboon Population

We are interested in modeling the population of a baboon troop. R-Code 6.1 loads the data and gives a simple analysis in terms of birth and death rates. The dataset is quite small and consists of 28 events over a period of roughly one year. Different modeling approaches are possible, we can pool the events birth and immigration as well as death and emigration. Alternatively we can consider the four events individually. In the latter case we only have three events to estimate the corresponding parameters.

R-Code 6.1: Baboon example (See Figure 6.1.)

```
baboon <- read.csv("data/baboon.csv", header=FALSE)
names(baboon) <- c("delta t","x(t+)","type")
str(baboon)
## 'data.frame': 28 obs. of  3 variables:
##  $ delta t: int  41 5 22 2 17 26 0 55 35 20 ...
##  $ x(t+)  : int  40 41 42 43 42 41 42 43 44 45 ...
##  $ type   : Factor w/ 4 levels "B","D","E","I": 1 1 1 2 2 4 4 1 4 3 ...
table( baboon$type)
```

```
##
##  B  D  E  I
## 10 12  3  3
Ttotal <- sum( baboon$delta)
nmin <- min(baboon$x)
nmax <- max(baboon$x)

baboon$up <- baboon$type %in% c('B','I')
nEvents <- length(baboon$type)

#  Visualization:
plot(cumsum(c( 0, baboon$delta)), c(baboon$x,baboon$x[nEvents]+1), type='s',
     ylab='troop size', xlab='Time [days]')
points( cumsum(baboon$delta), baboon$x, col=baboon$type)
legend('bottomleft', col=c(1,4,2,3), pch='o', legend=c('Birth', 'Immigration',
             'Death', 'Emigration'), bty='n')
#  Total time spend in each state:
AT <- aggregate( baboon$delta, by=list( baboon$x), sum)$x

# Sum of transistions
tmp <- aggregate( as.numeric( baboon$up), by=list( baboon$x, baboon$up), sum)
NTiup <- tmp$x[ tmp$Group.2==TRUE]
tmp <- aggregate( as.numeric(!baboon$up), by=list( baboon$x, baboon$up), sum)
NTidown <- tmp$x[ tmp$Group.2==FALSE]
Qup <- NTiup/AT[-10]      # Qij = N_T(i,j)/A_T(i), see later
Qdown <- NTidown/AT[-1]

plot( nmin:nmax, c(Qup, NA), ylim=c(0,.25), ylab='Transition rates')
points( nmin:nmax, c(NA, Qdown), col=4, pch=3)
legend( 'topright', pch=c(1,3), col=c(1,4), legend=c("BI","DE"), bty='n')
summary(lmup <- lm(Qup~c(36:44)))$coef
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  0.748651  0.1954372  3.8306 0.0064524
## c(36:44)    -0.017136  0.0048758 -3.5145 0.0097998
summary(lmdown <- lm(Qdown~c(37:45)))$coef
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.2523573  0.0872898   2.891 0.023282
## c(37:45)    -0.0050337  0.0021248  -2.369 0.049677
abline( lmup, col=1)
abline( lmdown, col=4)
```

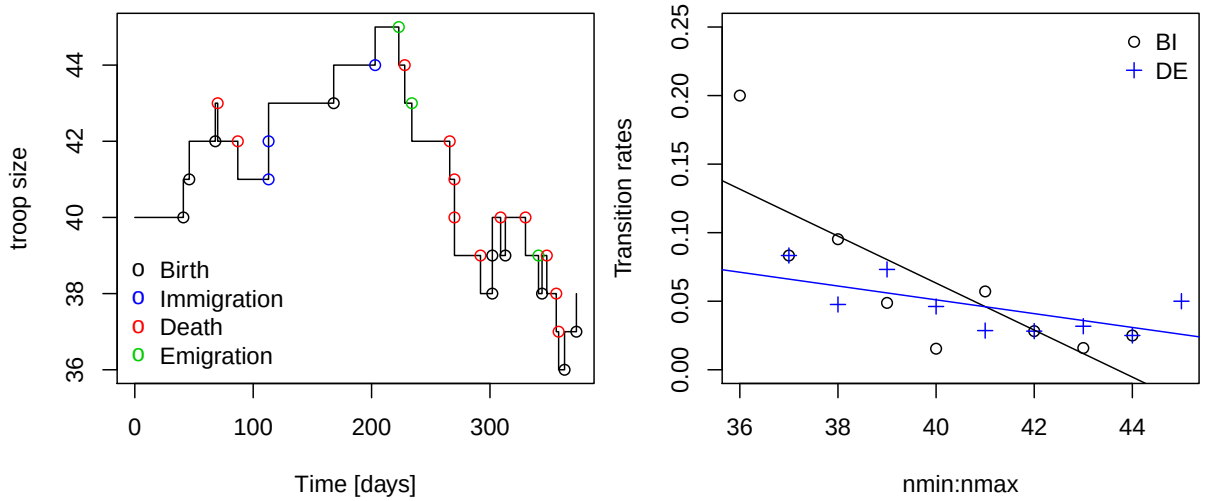

Figure 6.1: Baboon example. (See R-Code 6.1.)

6.1.2 Sterile Insect Technique

The sterile insect technique consists of dispersing sterile male insects to reduce the total insect population. There have been several very successful applications of this technique, e.g., screw-worm fly (*Cochliomyia hominivorax*) from North and Central America for a complete eradication.

Let θ the reproduction rate, n_i the male population size at generation i and S the number of sterile males put into the population. The reduction is from

$$n_1 = \theta n_0 \qquad \text{to} \qquad n_1 = \theta \frac{n_0}{n_0 + S} n_0 \,. \tag{6.1}$$

R-Code 6.2 illustrates the sterile insect technique with toy numbers. Notice that even for moderate reproduction rates θ, the amount of sterile males have to out-weight the initial population: $\theta n_0/(n_0 + S) < 1$, of course.

R-Code 6.2 Sterile insect technique.

```
n0 <- 20       # initial population
theta <- 4    # reproduction rate (male offsprings)
S <- 100       # number of sterile males put into the population

repeat {
        n0 <- theta * n0/(n0+S) * n0
        cat( round( n0,2), "  ")
        if( n0<1) break
}
## 13.33   6.27   1.48   0.09
```

We can generalize the discrete "generation"-type approach to a continuous time by denoting the population size at time t by $n(t)$. We assume that the population change is proportional to

the population and for small time steps, linear in time. A more formal justification of the last assumption follows soon.

More specifically, we are looking for a function $n(t)$, such that $n(t+h) = n(t) + h\lambda n(t)$, h small, which can be solved by

$$\lambda n(t) = \frac{n(t+h) - n(t)}{h} \approx n'(t) \text{ for small h} \tag{6.2}$$

$$\Rightarrow \frac{n'(t)}{n(t)} = \frac{d}{dt} \log n(t) = \lambda \qquad \Rightarrow n(t) = n_0 e^{\lambda t}. \tag{6.3}$$

In what follows, we consider a stochastic version of $n(t)$.

6.2 Birth Process

To start, we look at a simple birth process where new individuals are born at specific times. As we do not model death, a mother cell splitting into two new cells which again splits is probably a better model. With every birth the population size increases from k to $k+1$. As has been done in the past we do not model explicitly individuals but rather a population.

6.2.1 The Poisson Process

As state, we consider the size of the population, with independent births. Hence, the status of the population can be summarized by a counting process, i.e., a process that counts the number of events up to time t. More formally:

Definition 6.1. A counting process $\{N(t), t \geq 0\}$ is a stochastic process with

1. $N(0) = 0$;
2. $N(t) \in \{0, 1, 2, \dots\}$, for all $t \geq 0$;
3. if $s < t$, $N(s) \leq N(t)$. ♢

Definition 6.2. A counting process, or any general stochastic process $X(t)$, is said to have independent increments if the random variables

$$X(t_2) - X(t_1), X(t_3) - X(t_2), \dots, X(t_n) - X(t_{n-1}) \tag{6.4}$$

are independent, for all $0 \leq t_1 < t_2 < \cdots < t_n$. ♢

In the last chapter, we assumed independent interarrival times. Moreover, we even assumed a stationary setting:

Definition 6.3. A counting process, or any general stochastic process $X(t)$, is said to have stationary increments if the random variables

$$X(t_2) - X(t_1) \quad \text{and} \quad X(t_2 + h) - X(t_1 + h) \tag{6.5}$$

have the same distribution, for all $0 \leq t_1 < t_2$ and $r > 0$. ♢

Because a counting process counts the number of events, the following two facts are evident. (i) A counting process has independent increments if the numbers of arrivals in non-overlapping, i.e., disjoint, intervals are independent. (ii) A counting process has stationary increments if $N(t_2) - N(t_1)$ has the same distribution as $N(t_2 - t_1)$.

We now introduce a very convenient and often used counting process.

Definition 6.4. A counting process $\{N(t), t \geq 0\}$ is a Poisson process with rate λ if:

1. $N(0) = 0$;
2. $N(t)$ has independent increments;
3. $\mathrm{P}(N(t+h) - N(t) = n) = \exp(-\lambda h)\dfrac{(\lambda h)^n}{n!}$. ♢

Starting from Definition 6.4 and considering short intervals $h = t_2 - t_1$ we can develop $\exp(-\lambda h)$ as a Taylor series to get

$$\mathrm{P}\big(N(h) = 0\big) = \exp(-\lambda h) = 1 - \lambda h + \frac{\lambda^2 h^2}{2} - \cdots \approx 1 - \lambda h + o(h), \tag{6.6}$$

$$\mathrm{P}\big(N(h) = 1\big) = \lambda h \exp(-\lambda h) = \lambda h\Big(1 - \lambda h + \frac{\lambda^2 h^2}{2} - \ldots\Big) \approx \lambda h + o(h), \tag{6.7}$$

$$\mathrm{P}\big(N(h) \geq 2\big) = 1 - \mathrm{P}\big(N(h) = 0\big) - \mathrm{P}\big(N(h) = 1\big) \approx o(h). \tag{6.8}$$

These results can be formalized and extended as follows.

Property 6.1. *For a Poisson process $\{N(t), t \geq 0\}$ with rate λ:*

1. - $\mathrm{P}\big(N(h) = 0\big) = 1 - \lambda h + o(h)$,
   - $\mathrm{P}\big(N(h) = 1\big) = \lambda h + o(h)$,
   - $\mathrm{P}\big(N(h) \geq 2\big) = o(h)$;
2. *the interarrival times X_t are exponential random variables with rate λ.*

There is an alternative definition of a Poisson process that can be often found in the literature. The definition is essentially using the Property 6.1.1.

Definition 6.5. A counting process $\{N(t), t \geq 0\}$ is a Poisson process with rate λ if:

1. $N(0) = 0$
2. $N(t)$ has independent and stationary increments
3. - $\mathrm{P}\big(N(h) = 0\big) = 1 - \lambda h + o(h)$;
   - $\mathrm{P}\big(N(h) = 1\big) = \lambda h + o(h)$;
   - $\mathrm{P}\big(N(h) \geq 2\big) = o(h)$. ♢

We now use a Poisson process and construct associated transition probabilities.

6.2.2 Transition Probabilities for a Poisson Process

We denote $p_{ij}(h)$ as the probability that the population size $N(t)$ increases from i to j withing an (infinitesimal) time step h. By stationarity of the increments, this probability does not depend on t.

$$p_{ij}(h) = \mathrm{P}\big(N(t+h) = j \mid N(t) = i\big) = \begin{cases} \lambda_i h + o(h), & \text{for } j = i+1, \\ 1 - \lambda_i h + o(h), & \text{for } j = i, \\ o(h), & \text{otherwise.} \end{cases} \tag{6.9}$$

Furthermore, we have the initial condition:

$$p_{ij}(0) = \delta_{i,j} = \begin{cases} 1, & \text{for } i = j, \\ 0, & \text{otherwise.} \end{cases} \tag{6.10}$$

Here and in many settings we consider proportional birth rates:

$$\lambda_i = i\lambda. \tag{6.11}$$

Such a birth process is called a Yule process. This means, the rate at which the population grows increases with every new birth.

We now elaborate $p_{1n}(t)$, for arbitrary $t > 0$ and $n \geq 1$, as opposed to small increments h in (6.9):

$$p_{1n}(t+h) = \big(1 - n\lambda h + o(h)\big)p_{1n}(t) + \big((n-1)\lambda h + o(h)\big)p_{1,n-1}(t), \tag{6.12}$$

$$p_{1n}(t+h) - p_{1n}(t) = -n\lambda h p_{1n}(t) + (n-1)\lambda h p_{1,n-1}(t) + o(h), \tag{6.13}$$

$$\frac{d}{dt}p_{1n}(t) \approx \frac{p_{1n}(t+h) - p_{1n}(t)}{h} = -n\lambda p_{1n}(t) + (n-1)\lambda p_{1,n-1}(t). \tag{6.14}$$

Using the initial conditions $p_{10}(0) = 0$ and $p_{11}(0) = 1$, we solve recursively to obtain

$$p_{1n}(t) = \exp(-\lambda t)\big(1 - \exp(-\lambda t)\big)^{n-1}. \tag{6.15}$$

Hence our process is distributed as a geometric random variable with mean $\exp(\lambda t)$.

Remark 6.1. When starting with more than one individual at the beginning, the resulting distribution is a negative Binomial distribution, with the same parameter ♣

6.3 Death Process

A pure death process can be modeled as removing individuals from a population or reversing the time of a birth process. Formally, we can show that a birth process, as introduced above, is time reversible.

6.4 Birth-Death Processes

If we consider birth and deaths independent, we can "superimpose" the two processes. This works because the interarrival times of both subprocesses are exponential random variable and because of the memoryless property of exponential random variables.

We denote λ_i the birth rate and μ_i the death rate at population size i. If we assume again two Yule processes, i.e., $\lambda_i = i\lambda$ and $\mu_i = i\mu$, we have similarly to the previous derivations

$$\frac{d}{dt}p_{1n}(t) = (n+1)\mu p_{1,n+1}(t) - n(\lambda+\mu)p_{1n}(t) + (n-1)\lambda p_{1,n-1}(t). \tag{6.16}$$

Of course we may consider $p_{ij}(t)$, for arbitrary i and j. If we assume the general setting, with "arbitrary" rates, we have

$$\frac{d}{dt}p_{ij}(t) = \mu_{j+1}p_{i,j+1}(t) - (\lambda_i+\mu_j)p_{ij}(t) + \lambda_{j-1}p_{i,j-1}(t), \tag{6.17}$$

with initial condition $p_{ij}(0) = \delta_{ij}$.

In case we have a stationary population, we can solve (6.17) by setting $\frac{d}{dt}p_{ij}(t) = 0$ and realizing that we have the same equations as in the Chapter 5.

Remark 6.2. Nonstationary solutions can be obtained by the use of Laplace transformations. ♣

6.5 Generator Matrix

In the framework of discrete time Markov chains, we have constructed the transition probability matrix, giving us the probability to move from i to j in the next step. In a continuous setting we do not have discrete steps and work with rates.

Let $\mathbf{P}(t) = (p_{ij}(t))$.

Property 6.2. *Assume a finite state space S and assume that all transition probabilities $p_{ij}(t)$ are continuous at $t = 0$.*

1. $\mathbf{P}(0) = \mathbf{I}$;
2. $\mathbf{P}(s+t) = \mathbf{P}(s)\mathbf{P}(t)$;
3. $\sum_{j=1}^{K} \frac{d}{dt}p_{ij}(t)\Big|_{t=0} = \sum_{j=1}^{K} p'_{ij}(0) = 0$, *for all i, which can be written equivalently as* $\mathbf{P}'(0)\mathbf{1} = \mathbf{0}$.

Definition 6.6. The matrix $\mathbf{G} = \mathbf{P}'(0) = \frac{d}{dt}\mathbf{P}(t)\Big|_{t=0} = \left(\frac{d}{dt}p_{ij}(0)\right)$ is called the generator of $\mathbf{P}(t)$ or simply the generator matrix. ♢

Let $q_i = \sum_{j=1, j\neq i}^{K} g_{ij}$ and thus $q_i = -g_{ii}$. By definition of $\mathbf{G}$, we have by a simple Taylor argument

$$p_{ij}(h) = p_{ij}(0) + p'_{ij}(0)(h-0) + o(h) \approx g_{ij}h, \qquad i \neq j \tag{6.18}$$

$$p_{ii}(h) \approx 1 - q_i h, \tag{6.19}$$

and thus

$$\mathrm{P}(\text{ transition from } i \text{ to } j \text{ in } [t, t+h] \mid \text{ transition occurred }) = \frac{p_{ij}(h)}{1-p_{ii}} \approx \frac{g_{ij}}{q_i}\,. \tag{6.20}$$

Property 6.3. *1.* $\mathbf{P}(t) = \exp(-t\mathbf{G}) = \sum_{k=0}^{\infty} \frac{t^k}{k!}\mathbf{G}^k$;

2. $\frac{d}{dt}\mathbf{P}(t) = \mathbf{P}(t)\mathbf{G}$ *(Kolmogorov's forward equations);*

3. $\frac{d}{dt}\mathbf{P}(t) = \mathbf{G}\mathbf{P}(t)$ *(Kolmogorov's backward equations).*

It is possible to show that the process given in Property 6.3.1 is the unique solution satisfying equations given in Property 6.2.1 and 2.

6.6 Bibliographic Remarks

The baboon data analysis is from Guttorp (1995), with original data from Altmann and Altmann (1970). See also Cohen (1969) and Keiding (1977).

For the sterile insect technique (SIT), Alphey *et al.* (2010) give an accessible overview, also in forms of FAQs. Scott and Benedict (2016) give a nice list of reasons why the eradication of the Screwworm Fly worked well. Side note, the current use of drones simplifies the dispersion of reared and sterile compared to last century techniques. Karlin and Taylor (1981), pages 388ff, use the SIT as a BD process, with 20 wild and 100 sterile males. In practice more than 1000 sterile males per hectare may be needed. See, e.g., Chapter 9 of Enkerlin (2007).

Lab Content

1. Review of JFK and DIA data (Chapter 05);
2. Laplace transforms to solve difference equations with non-constant coefficients;
3. Very short excursion of probability of extinction and mean survival time.

Worksheet 6

16. Simulation of a Birth-Death process with immigration.
17. Poisson properties shown with the help of probability generating functions.

Chapter 7

Continuous Time Markov Processes

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta111/chapter07.R.

We now formalize the concepts of the last two chapters in the general setting of continuous time Markov chains.

7.1 Application

Retake the Baboon example, but consider a stationary setting, i.e., the population size of the troop remains the same up to stochastic fluctuations. In other words, the number of births and immigration balance deaths and emigration.

We are interested in modeling the population of a baboon troop. R-Code 7.1 loads the data and gives a simple analysis in terms of birth and death rates. The dataset is quite small and consists of 28 events over a period of roughly one year. Different modeling approaches are possible, we can pool the events birth and immigration as well as death and emigration. Alternatively we can consider the four events individually. In the latter case we only have three events to estimate the corresponding parameters.

R-Code 7.1: Baboon example (See Figure 7.1.)

```
baboon <- read.csv("data/baboon.csv", header=FALSE)
names(baboon) <- c("delta t","x(t+)","type")
str(baboon)
## 'data.frame': 28 obs. of  3 variables:
##  $ delta t: int  41 5 22 2 17 26 0 55 35 20 ...
##  $ x(t+)  : int  40 41 42 43 42 41 42 43 44 45 ...
##  $ type   : Factor w/ 4 levels "B","D","E","I": 1 1 1 2 2 4 4 1 4 3 ...
table( baboon$type)
```

```
##
##  B  D  E  I
## 10 12  3  3
Ttotal <- sum( baboon$delta)
nmin <- min(baboon$x)
nmax <- max(baboon$x)
baboon$up <- baboon$type %in% c('B','I')
nEvents <- length(baboon$type)

#  Visualization:
plot(cumsum(c( 0, baboon$delta)), c(baboon$x,baboon$x[nEvents]+1), type='s',
     ylab='Troop size', xlab='Time [days]')
points( cumsum(baboon$delta), baboon$x, col=baboon$type)
legend('bottomleft', col=c(1,4,2,3), pch=1, legend=c('Birth', 'Immigration',
             'Death', 'Emigration'), bty='n')
#  Total time spend in each state:
A <- aggregate( baboon$delta, by=list( baboon$x), sum)$x

# Sum of transistions
tmp <- aggregate( as.numeric( baboon$up), by=list( baboon$x, baboon$up), sum)
Niup <- tmp$x[ tmp$Group.2==TRUE]
tmp <- aggregate( as.numeric(!baboon$up), by=list( baboon$x, baboon$up), sum)
Nidown <- tmp$x[ tmp$Group.2==FALSE]
Gup <- Niup/A[-10]      # estimate of gij  is N(i,j)/A(i), see later
Gdown <- Nidown/A[-1]

plot( nmin:nmax, c(Gup, NA), ylim=c(0,.25), xlim=c(35,47),
     xlab='Troop size', ylab='Transition rates')
points( nmin:nmax, c(NA, Gdown), col=2, pch=3)
legend( 'topright', pch=c(1,3), col=c(1,2), legend=c("BI","DE  "), bty='n')
summary(lmup <- lm(Gup~c(36:44)))$coef
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  0.748651  0.1954372  3.8306 0.0064524
## c(36:44)    -0.017136  0.0048758 -3.5145 0.0097998
summary(lmdown <- lm(Gdown~c(37:45)))$coef
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.2523573  0.0872898   2.891 0.023282
## c(37:45)    -0.0050337  0.0021248  -2.369 0.049677
abline( lmup, col=1)
abline( lmdown, col=2)
```

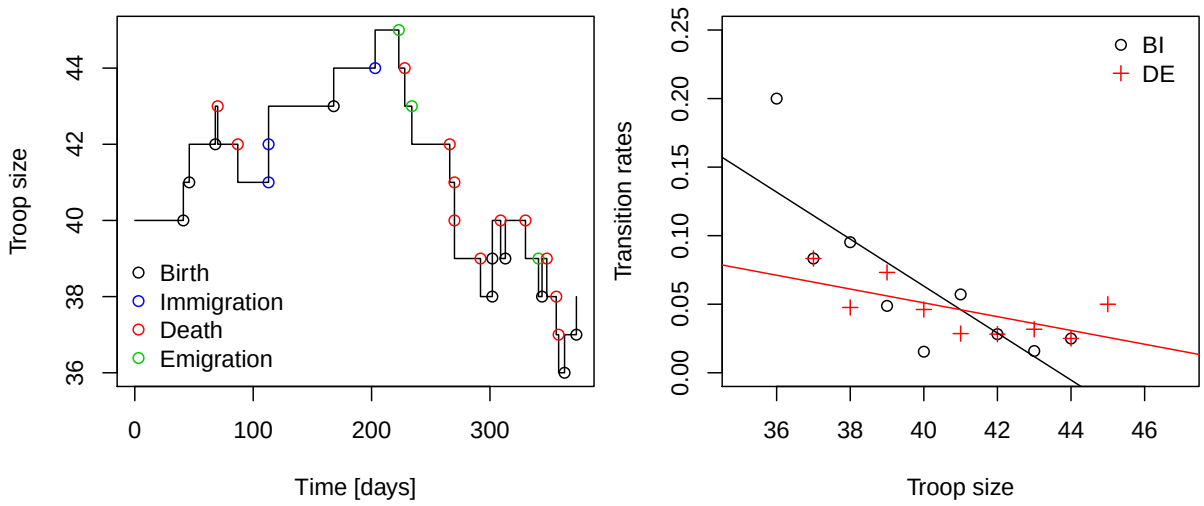

Figure 7.1: Baboon example. (See R-Code 7.1.)

Later in this chapter, we recognize the estimates of the transition rates (more specifically of the generator matrix) as maximum likelihood estimates. Therefore, we can also resort to classical likelihood theory to compare differently complex nested models (via a likelihood ratio test, for example).

7.2 Continuous Time Markov Chains

Much of the chapter follows the same structure as Chapter 4. As in the discrete time setting, we assume that besides "current" state knowing the past does not add additional information about the state of the chain in the future.

Definition 7.1. A stochastic process $\{X(t); t \geq 0\}$ with a discrete state space S is called a continuous time Markov chain (CTMC) if for all $s, t \geq 0$ and all states $i, j \in S$

$$\mathrm{P}\big(X(t+s) = j \mid X(s) = i,\ \{X(u),\ 0 \leq u \leq s\}\big) = P\big(X(t+s) = j \mid X(s) = i\big) =: p_{ij}(s, t+s), \tag{7.1}$$

i.e., the transition probability $p_{ij}(s, t+s)$ is the probability to be in j at time t given that the chain is in state i at time s. ♢

Definition 7.2. A continuous time Markov chain is time homogeneous if the transition probabilities do not depend on the times s and $s+t$ but only on the elapsed time t, i.e., $p_{ij}(s, t+s) = p_{ij}(t)$. ♢

We assume time homogeneous chains only. As in the discrete setting, the chain moves from state to state and then remains in the state for some time. We denote this time as *holding time* H_i, i.e., the time at state i before jumping to state $j \neq i$.

Because of the Markov property and time homogeneity, the distribution of the holding times is imposed.

Property 7.1. *In a CTMC, the holding times H_i are exponentially distributed $H_i \sim \mathcal{E}xp(q_i)$.*

Definition 7.3. Let $\{X(t), t \geq 0\}$ a CTMC.

1. $p_i(t) = \mathrm{P}\big(X(t) = i\big)$ is called the absolute state probability at time t.
2. $\{p_i(t), i \in S\}$ is called the absolute probability distribution of the Markov chain at time t.
3. $\{p_i(0), i \in S\}$ is the initial probability distribution of the Markov chain. ♢

As in the past, we write – for any finite state space S – the matrix $\mathbf{P}(t) = \big(p_{ij}(t)\big) \in \mathbb{R}^{K\times K}$ and the row vector $\boldsymbol{p}(t) = \big(p_1(t), \ldots, p_K(t)\big)$. Note that $\mathbf{P}(0) = \mathbf{I}$.

Remark 7.1. It is theoretically possible that there are states i for which $\sum_j p_{ij}(t) < 1$, which translates to a setting where in a finite time interval unboundedly many transitions between the states occur. We do not consider this situation here and thus assume that $\sum_j p_{ij}(t) = 1$, for all states $i \in S$. ♣

Recall that in the setting of a discrete time Markov chain, we required the corresponding transition matrix, and the initial distribution to fully describe the process. In the continuous time setting we need the transition matrix with coresponding holding rates and its initial distribution.

Similarly to the construction of the generator matrix of Section 6.5 based on counting processes, we can construct the generator matrix (or transition *rate* matrix) based on the exponential holding times.

We proceed as follows. We assume that for all states the transition probabilities sum to one. In other words that for infinitesimal time we do not have movements, i.e., $p_{ij}(0) = \delta_{ij}$. We define

$$q_i = \lim_{h\to 0} \frac{1 - p_{ii}(h)}{h}, \qquad g_{ij} = \lim_{h\to 0} \frac{p_{ij}(h)}{h}, i \neq j. \tag{7.2}$$

Both limits exist by the former assumption. The parameters q_i are the unconditional transition rates of leaving state i for any other state and the parameters g_{ij} are the conditional transition rates of making a transition from state i to state j.

Thus

$$\mathrm{P}(\text{ transition from } i \text{ to } j \text{ in } [t, t+h] \mid \text{ transition occurred }) = \frac{p_{ij}(h)}{1 - p_{ii}} \approx \frac{g_{ij}}{q_i}\,. \tag{7.3}$$

These transition rates define the generator matrix $\mathbf{G} = (g_{ij})$, with $g_{ii} = -q_i$.

Similar as in the context of Poisson processes, we can show that the Markov property implies holding times that are exponential. Or a system with exponential holding times satisfies the Markov property. Similarly, we can imagine we have individual holding times $H_{ij} \sim \mathcal{E}xp(\lambda_{ij})$, $i \neq j$, representing random times when to move from state i to j. Of course,

$$H_i = \min(H_{i1}, \ldots, H_{iK}) \sim \mathcal{E}xp\Big(\sum_j \lambda_{ij}\Big) = \mathcal{E}xp(q_i), \tag{7.4}$$

because equation (7.2) implies $\lambda_{ij} = g_{ij}$.

Again by homogeneity, as soon as the chain leaves the state i, it has a fixed number of states that it can jump to among (the fixed) states and it has a time invariant probability to jump to any of these. Hence, conditional on an "event", we can reuse these transition probabilities and "neglect" the holding times, as illustrated in Figure 7.2.

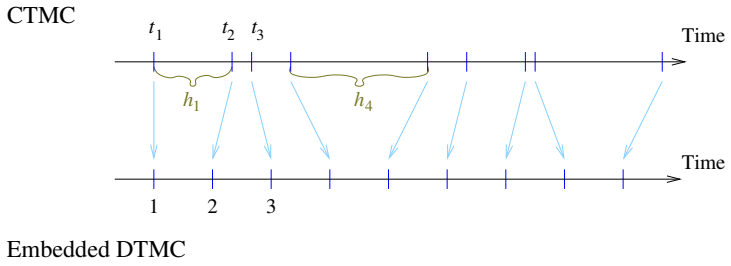

Figure 7.2: Illustration of CTMC embedding.

Definition 7.4. For a CTMC with generator matrix $\mathbf{G}$, we define $p_{ij} = g_{ij}/q_i$. The DTMC with transition matrix $\mathbf{P} = (p_{ij})$ is the associated embedded DTMC. ♢

Remark 7.2. Based on Remark 7.1, we directly have $p_{ii} = 0$. ♣

Whereas the embedded DTMC is based on the transition probabilities p_{ij}, we often visualize directly the transition rates g_{ij} when graphically representing the state. Examples thereof are, for example, Figure 5.3.

Example 7.1. In the setting of a M/M/1/∞ queue, the embedded DTMC is based on the transition probabilities

$$p_{ij} = \begin{cases} 1 & \text{if } i = 1, j = 2, \\ \dfrac{\lambda}{\lambda + \mu} & \text{if } j = i + 1 \\ \dfrac{\mu}{\lambda + \mu} & \text{if } j = i - 1 \\ 0 & \text{otherwise.} \end{cases} \tag{7.5}$$

Similarly, Figure 7.3 represents a M/M/1/4 queue with $\lambda = p/(1-p)\mu$. ♣

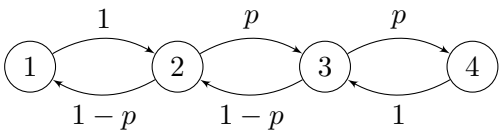

Figure 7.3: Simple four state system.

7.3 Classification

As with DTMC, the states i and j are said to communicate if state j is reachable from state i, and state i is reachable from state j. All states that communicate with each other form a class and thus the state space S is decomposed into disjoint classes. Similarly, irreducible chain is a chain for which all states communicate, i.e., the S consists of a single class.

Definition 7.5. A CTMC is irreducible if and only if its embedded chain is irreducible. ♢

Definition 7.6. A state is recurrent/transient for a CTMC if and only if it is recurrent/transient for its embedded chain. ♢

For some theoretical results, we need to further specify recurrence, namely positive recurrence, the expected amount to return is finite. Moreover, an irreducible Markov chain with finite state space is always positive recurrent.

Definition 7.7. An irreducible aperiodic positive chain is called ergodic. ♢

Remark 7.3. For a CTMC with transition probabilities $p_{ij}(t)$, a state i is recurrent or transient if

$$\int_0^\infty p_{ii}(t)\,\mathsf{d}t = \infty \qquad \text{or} \qquad \int_0^\infty p_{ii}(t)\,\mathsf{d}t < \infty, \tag{7.6}$$

respectively. However, the concept of a period does not make sense. ♣

7.4 Stationarity

Under stationarity, the mean intensity (per unit time) of leaving and of entering the states are equal.

Definition 7.8. A absolute state distribution is called stationary if for a state the distribution is time invariant, i.e., for $i \in S$, $\pi_i = p_i(t)$ for any t. ♢

For a irreducible Markov chain with finite state space (or an irreducible positive recurrent), a *unique* stationary solution $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$ exists and it can be shown that

$$\pi_j = \lim_{t\to\infty} p_{ij}(t). \tag{7.7}$$

Moreover, $\lim_{t\to\infty} p_j'(t) = 0$, otherwise $p_j(t)$ would (unboundedly) increase as $t \to \infty$ and contradict the requirement that $p_j(t) \leq 1$.

We conclude this section by summarizing some well-known results (see Property 6.3) and some new results from this chapter.

Property 7.2. *Let $\{X(t), t \geq 0\}$ a irreducible CTMC with finite state space.*

1. $\mathbf{G1} = \mathbf{0}$;
2. $\mathbf{G} = \dfrac{d}{dt}\mathrm{P}(0)$;
3. $\mathbf{P}(t) = \exp(-t\mathbf{G}) = \displaystyle\sum_{k=0}^{\infty} \frac{t^k}{k!}\mathbf{G}^k$;
4. $\dfrac{d}{dt}\mathbf{P}(t) = \mathbf{P}(t)\mathbf{G}$ *(Kolmogorov's forward equations);*
5. $\dfrac{d}{dt}\mathbf{P}(t) = \mathbf{G}\mathbf{P}(t)$ *(Kolmogorov's backward equations);*
6. *In case of stationarity:* $\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P}$;
7. *In case of stationarity:* $\mathbf{0} = \boldsymbol{\pi}\mathbf{G}$ *(balance equations).*

For irreducible, positive recurrent CTMC with infinite state space, we can write the property in terms of sums for individual states. For example, $\sum_{j\in S} g_{ij} = 0$, for all $i \in S$.

7.5 Estimation

In a CTMC framework, the generator matrix $\mathbf{G}$ determines "uniquely" the chain. Given data we have to estimate the parameters g_{ij} and q_i, i.e., the elements of $\mathbf{G}$.

Since the holding times are exponential, it is possible to setup a likelihood function.

A natural estimation procedure is to setup a likelihood function based on the number of jumps, the holding times and the jump chain (this information provides a sufficient statistic). That means we write the likelihood as a product of starting position, holding time, move, holding time, etc.

More formally, let $v = \{n, x_0, t_1, x_1, t_2, x_2, \ldots, t_n, x_n\}$ represent the observed chain, i.e., starting at state x_0, n transitions via x_1, x_2, etc. staying $t_{r+1} - t_r$ at x_r. For simplicity, assume $n > 0$ and that all the states in S have been visited during the observation time $[0, T]$. Also write $t_0 = 0$ and $t_{n+1} = T$, then

$$L(\mathbf{G}; v) = p_{x_0}(0) \times q_{x_0} \exp(-q_{x_0} t_1) \times \tag{7.8}$$

$$\prod_{j=1}^{n-1} \frac{g_{x_{j-1},x_j}}{q_{x_{j-1}}} \cdot q_{x_j} \exp\big(-q_{x_j}(t_{j+1} - t_j)\big) \times \frac{g_{x_{n-1},x_n}}{q_{x_{n-1}}} \cdot \exp\big(-q_{x_j}(T - t_n)\big) \tag{7.9}$$

$$= p_{x_0}(0) \times \Big(\prod_{j=1}^{n} g_{x_{j-1},x_j}\Big) \times \prod_{j=0}^{n} \exp\big(-q_{x_j}(t_{j+1} - t_j)\big) \tag{7.10}$$

$$= p_{x_0}(0) \Big(\prod_{\substack{i,j \\ i \neq j}}^{K} g_{i,j}^{N(i,j)}\Big) \exp\Big(-\sum_{i=1}^{K} A(i) q_i\Big), \tag{7.11}$$

where $N(i, j)$ is the number of transitions from i to j and $A(i)$ is the total time spend in state i.

In general, we cannot estimate the initial probability distribution of the Markov chain with a single realization. Hence, we can formulate the likelihood conditional on $p_{x_0}(0)$. A more complex approach would be to include a stationarity assumption and thus including the initial state.

Remark 7.4. Note that we have set up the notation fairly simple. Ideally and to answer more general questions, we have to include time as well, e.g., $A_t(i)$ is the total time spend in state i up to time t. ♣

Intuitive estimates (or estimators) are

$$\widehat{g}_{ij} = \frac{N(i,j)}{A(i)}, \qquad \widehat{q}_i = \frac{\sum_{j \neq i} N(i,j)}{A(i)}. \tag{7.12}$$

As a matter of fact, these are not only the intuitive estimates/estimators but also the maximum likelihood (ML) ones. By classical ML theory we have

$$\sqrt{A(i)\widehat{g}_{ij}}\,(\widehat{g}_{ij} - g_{ij}) \xrightarrow{t\to\infty} \mathcal{N}(0,1) \qquad \text{in distribution.} \tag{7.13}$$

If we have an ergodic chain, it is possible to show that

$$\frac{A(i)}{t} = \frac{A_t(i)}{t} \xrightarrow{t\to\infty} \pi_i \qquad \text{in probability,} \tag{7.14}$$

$$\frac{N(i,j)}{t} = \frac{N_t(i,j)}{t} \xrightarrow{t\to\infty} \pi_i g_{ij} \qquad \text{in probability.} \tag{7.15}$$

Quite often, we do impose parametric models on the generator matrix $\mathbf{G}$, i.e., $\mathbf{G} = \mathbf{G}(\boldsymbol{\theta})$. The individual functions $g_{ij}(\boldsymbol{\theta})$ need to satisfy Fisher's regularity condition, of course.

Example 7.2. R-Code 7.1 calculates the individual values $N(i,j)$ and $A(i)$.

A parameterized likelihood is, e.g., based on (λ, ν, κ) with "birth" rates $i\lambda + \nu$ (birth and immigration) and "death" rates $i\kappa$ (sum of death and emigration). R-Code 7.2 calculates the ML estimates based on the likelihood. Notice that the values differ from the ones given by Guttorp (1995), page 160.

R-Code 7.2 also calculates the ML estimates for a four parameter setting BIDE setting with rates $i\lambda$, ν, $i\mu$, and $i\eta$, respectively.

Uncertainties can be derived explicitly or using the argument *hessian=TRUE* in the optimization. ♣

R-Code 7.2 BD and BIDE fitting of the baboon troop data.

```
ll.BD <- function(theta) {      # Reducing to BD model:
    nTii1 <- colSums(table(baboon[baboon$up,1:2]))  # # i->i+1
    iseq <- min(baboon$x):(max(baboon$x)-1)
    ST <- sum(baboon$x*baboon$`delta t`)
    sum(log( theta[1]*iseq+theta[2])*nTii1)+log( theta[3])*sum(!baboon$up)-
        theta[2]*Ttotal - (theta[1]+theta[3])*ST
}
res <- optim(c(1e-9,1e-2,1e-5), ll.BD, control=list(fnscale=-1),
                    method="L-BFGS-B", lower=c(1e-12,1e-12,1e-12) )
print( res$par)     # estimated parameters
## [1] 0.00050957 0.00999982 0.00122703
print( res$value)  # Value of loglikelihood
## [1] -176.46
# Guttorp's solution:
ll.BD(c(1.65e-9,3.49e-2,9.74e-4))
## [1] -175.65
#   Using all information for a BIDE model:
ST <- sum(baboon$x*baboon$`delta t`)
print( mle <- c( sum( baboon$type %in% 'B')/ST,
                 sum( baboon$type %in% 'I')/Ttotal,
                 sum( baboon$type %in% 'D')/ST,
                 sum( baboon$type %in% 'E')/ST) )
## [1] 0.00064906 0.00804290 0.00077887 0.00019472
```

7.6 Bibliographic Remarks

The content here is a consolidation of Guttorp (1995), Karlin and Taylor (1981) and Beichelt (2006), but plenty of other books and resources at virtually arbitrary mathematical level exist. Estimation within CTMC is particularly well presented in Guttorp (1995).

Lab Content

1. On Markov Chains: from discrete state space to continous state space. Sketch of basic properties.

Worksheet 7

18. A Gibbs sampler on $\{0,1\}^2$.
19. Develop and implement a Metropolis Hastings algorithm on a discrete state space.
20. Continuous state space Markov chain.

Chapter 8

Deterministic Compartmental Models

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta111/chapter08.R.

Until now we have considered the state of a system or, equivalently, a description of a single population. We now extend our modeling approach to several populations or rather subpopulations in order to model settings like, predator-prey systems, chemical reactions etc. We start considering a deterministic setting first and revisit stochastic models in Chapter 10.

8.1 Lynx–Hare Population Model

The Canadian lynx and snowshoe hare population has been well documented through pelt-trading records of the Hudson Bay Company, starting in 1845. Here we look at a subset thereof and try to model the interaction between both populations. R-Code 8.1 contains the data, visualized in Figure 8.1, top row. In a second step we fit a "classical" Lotka–Volterra model to the data, which is shown as function in time and in phase space in the bottom panels of Figure 8.1.

R-Code 8.1: Lynx–hare population model (See Figure 8.1.)

```
require(deSolve)
Time <- 1900:1920   # years 1900 to 1920
Hare <- c(30,47.2,70.2,77.4,36.3,20.6,18.1,21.4,22,25.4,27.1,40.3,57,
          76.6,52.3,19.5,11.2,7.6,14.6,16.2,24.7)
Lynx <- c(4,6.1,9.8,35.2,59.4,41.7,19,13,8.3,9.1,7.4,8,12.3,19.5,45.7,
          51.1,29.7,15.8,9.7,10.1,8.6);
plot( Time, Hare, type='b', col=3, ylim=c(0, 100), ylab="Counts")
lines( Time, Lynx, type='b', col=2)
```

```
legend( "topright", c("Hare", "Lynx"), lty=1, col=3:2, bty="n")
plot( Hare, Lynx, type='b')
# Lotka-Voltera model, 4 parameter model!
LV <- function(time, state, parameters) {
        with(as.list(c(state, parameters)), {
                dX <- a1 * X - a2 * X * Y
                dY <- b2 * X * Y - b1 * Y
                return(list(c(dX, dY)))
        })
}
pars <- c(X=34.8, Y=3.7, a1=.48, a2=.025, b1=.93, b2=0.028)
fcn <- function( pars) {
    out <- ode(y=pars[1:2], times=Time-Time[1], func=LV, parms=pars[3:6],
               method='rk4')
    sum( (out[,2]-Hare)^2 + (out[,3]-Lynx)^2)
}
res <- optim( pars, fcn, control=list(maxit=1000))
print( res[c(1,2,4)])
## $par
##         X         Y        a1        a2        b1        b2
## 34.788183  3.727313  0.480287  0.024883  0.933364  0.027826
##
## $value
## [1] 586.02
##
## $convergence
## [1] 0
Time <- seq(0,to=20,by=.1)
out <- ode(y=res$par[1:2], times=Time, func=LV, parms=res$par[3:6])

matplot( x=Time+1900, y=out[,-1], type="l", xlab="Time",
        ylab="Counts", lty=1, col=3:2, ylim=c(0, 100))
legend( "topright", c("Hare", "Lynx"), lty=1, col=3:2, bty="n")
points( 1900:1920, Hare, col=3)
points( 1900:1920, Lynx, col=2)
# Phase space:
plot( Hare, Lynx,  xlab="Prey", ylab="Predator")
lines( out[,2], out[,3])
```

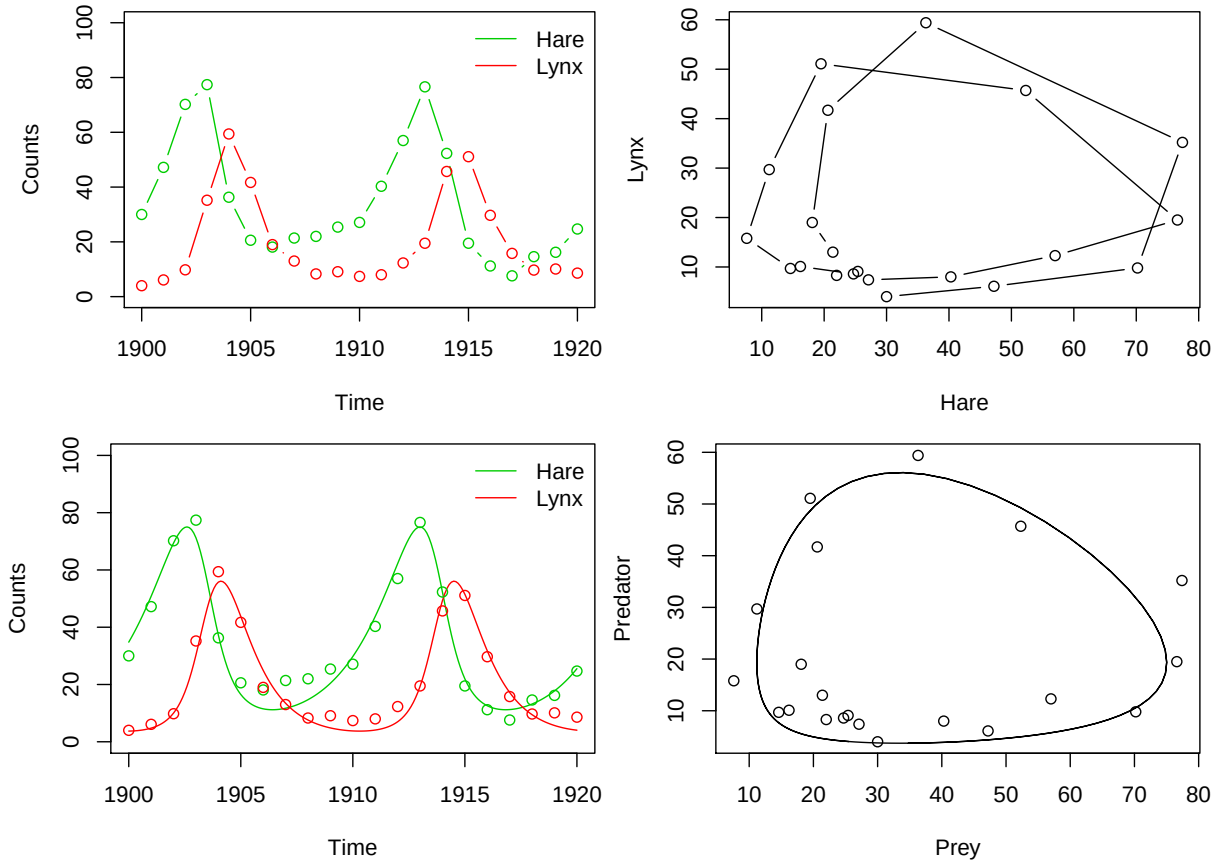

Figure 8.1: Lynx–hare population, top row data, bottom row fitted model with data points. (See R-Code 8.1.)

8.2 Simple Compartmental Models

A compartment model contains a (small) number of compartments or containers and the model describes transports of material or mass from one compartment to another compartment. Often the transport (rate) depends on the amount of material in the compartment.

In the previous section, we had two compartments: lynx and hare. Each compartment has an inflow (population increase) and drain (population decrease), as well as a transport (predator feeds on prey) and natural death. Other systems include, for example, containers for organs (liver, blood, other tissues) describing the dynamics of cholesterol. Sometimes, the compartments are not physically separated. A classical example is the modeling of a chemical reaction, where the compartments are the individual reactants (educts) and products. Moreover, in such a setting we even assume a homogeneous mixture of the reactants and products.

The basic approach to solve compartmental model is to consider the quantity of interest for each compartment and model the changes thereof. This idea is closely linked to the chapters modeling birth-death processes, see Chapter 6 and 7. Without restarting the development from a discrete setting towards a continuous framework using limit arguments we jump here directly to "continuous" equations. Let $y(t)$ denote the material (mass, proportion or "count") of one container at time t. The setup defines the changes for $y(t)$ as well as for further compartments. That means that we need to solve a system of differential equations.

From such a continuous setting, there is a natural link back towards a discrete description. If we solve the systems numerically, based on the discretization

$$y'(t) \approx \frac{y(t+h) - y(t)}{h} \tag{8.1}$$

we have the forward difference scheme

$$y'(t) = F(t, y(t)), \qquad y_{i+1} \approx y_i + hF(t_i, y_i), \quad t_i = t_0 + nh. \tag{8.2}$$

In practice we may use more complex discretization schemes, not having such intuitive forward "propagation" rules.

Remark 8.1. In R the systems are solved with the command `ode` from the package `deSolve`. It is possible to specify explicitly what solver to use. We recommend to start with the default setting. In case the function issues warnings, `method='rk4'` often solves the problem. ♣

The idea of compartmental models is quite intuitive and we illustrate these with more examples.

8.2.1 Predator-Prey Model

Consider two populations, y denoting predators and x denoting preys, varying over time t. The classical Lotka–Volterra model has the following four rates: (i) intrinsic rate of prey population increase, (ii) predation rate coefficient, (iii) reproduction rate of predators per 1 prey eaten and (iv) predator mortality rate. Of course, this is a very simplistic setting, as it relies on the following strict assumptions:

- rates of changes of the populations is proportional to its size;
- unlimited food for prey population;
- food supply of the predator population depends solely on the size of the prey population;
- limitless appetite of predators;
- stable environment.

The following equations are a special case of the classical *Lotka–Volterra* model:

$$\frac{dx(t)}{dt} = k_1 x(t) - k_2 x(t) y(t) \tag{8.3}$$

$$\frac{dy(t)}{dt} = k_2 x(t) y(t) - k_3 y(t) \tag{8.4}$$

The R-Code 8.2 implements a three parameter version, including an illustration of the steady state. The resulting plots are very similar to Figure 8.1 and are not shown. We illustrate the dynamics for 30 time units with increments of 0.1. We start with 10 prey and 1 predator.

Remark 8.2. Of course we could launch a discussion the existence of a periodic versus asymptotic solution. In the setting of a 2×2 system of first order linear equations, the type of solution is determined by the eigenvalues of the matrix describing the coefficients of system. ♣

R-Code 8.2 Lotka–Volterra

```
require(deSolve)
LV <- function(time, state, parameters) {  # Lotka--Voltera model
  with(as.list(c(state, parameters)), {
    dX <- k1 * X - k2 * X * Y
    dY <- k3 * X * Y - k4 * Y
    return(list(c(dX, dY)))
  })
}
init       <- c(X=10, Y=1)                    # Size of populations
parameters <- c(k1=.8, k2=.1, k3=.1, k4=.5)   # Rate parameters
Time       <- seq(0, 30, by=.1)               # Time frame
## Solve using ode (General Solver for Ordinary Differential Equations)
out <- ode(y=init, times=Time, func=LV, parms=parameters)

head( out, n=2)
##      time      X     Y
## [1,]  0.0 10.000 1.000
## [2,]  0.1 10.722 1.055
matplot( x=Time, y=out[,-1], type="l", ylab="x(t), y(t)", lty=1, col=2:3)
legend( "topright", c("Prey", "Predator"), lty=1, col=2:3, bty="n")
# Phase space:
plot( out[,2], out[,3], type='l', xlab="Prey", ylab="Predator")
points(out[,2], out[,3], cex=.3, col=4)
init    <- c(X=5, Y=8)
out <- ode(y=init, times=Time, func=LV, parms=parameters)
tail( out, n=2)
##        time X Y
## [300,] 29.9 5 8
## [301,] 30.0 5 8
```

8.2.2 Chemical Reaction

In chemical reaction the "number" of reactants in a mixture is large and we typically work with proportions instead. As in simple chemical reactions, we have actio and re-actio.

The illustration here consists of an enzymatic conversion of a single substrate, S, to a single product, P. In a first step, a binding of the educt to the enzyme, E, is achieved, which forms a substrate-enzyme complex, ES. The binding is followed by a reaction-release event, which yields the product and the enzyme.

As given by R-Code 8.3, the three curves converge to a stable solution (as opposed to cyclic solutions shown in the Lotka–Volterra setting). **Buggy for the moment!!**

R-Code 8.3 Reaction model (See Figure 8.2.)

```
react <- function(time, state, parameters, Etot=10) {
        with(as.list(c(state, parameters)), {
        # E <- max(0, Etot - ES)
            E <- Etot - ES
                dS <- k3 * ES - k1 * S * E
                dES <- k1 * S * E - k3 * ES - k4 * ES + k2 * P * E
                dP <- + k4 * ES - k2 * P * E
                return(list(c(dS, dP, dES)))
        })
}
init       <- c(S=10, P=0, ES=0)
parameters <- c(k1=.2, k2=.4, k3=.1, k4=.1)
Time       <- seq(0, 10, by=.05)
out <- ode(y=init, times=Time, func=react, parms=parameters, method='rk4')
matplot( x=Time, y=out[,-1], type="l", ylab="Proportion", lty=1, col=2:4)
```

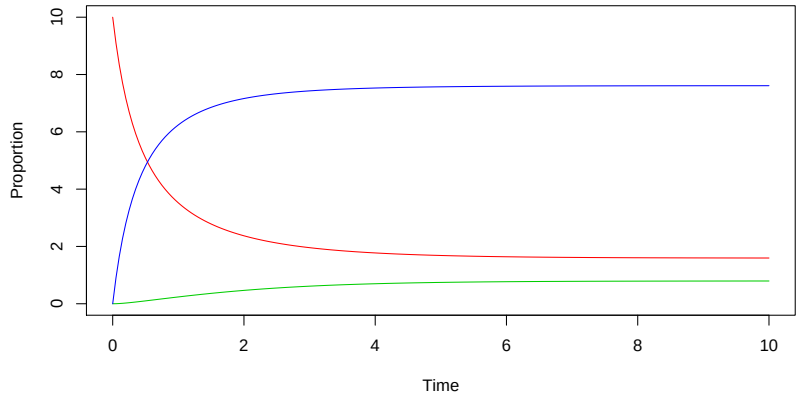

Figure 8.2: Solution of reaction model. (See R-Code 8.3.)

8.3 Epidemic Compartmental Models

The classical examples of compartmental models reside in epidemiological modeling, where healthy individuals (may) get infected and (may) recover. In the next two sections, we look at two specific examples.

8.3.1 SIR Model

We assume that an individual is either susceptible for a disease, infectious or recovered therefrom. Hence we can model the population by the three subpopulations S: susceptible, I: infectious and R: recovered. More formal assumptions of this so-called SIR model are as follows

- We have a closed population: an individual is either susceptible, infectious or recovered.
- The population is large enough to work with proportions: $S(t) + R(t) + I(t) = 1$, for all t.

- Chain of the process: $S \longrightarrow I \longrightarrow R$
- The parameters of the model are: βI: disease transmission rate $S \longrightarrow I$, γ: recovery rate $I \longrightarrow R$. Note that β is the contact rate taking into account the probability of getting the disease in a contact between a susceptible and an infectious subject.
- An infection is present in the population: $I(0) > 0$.

We formalize the model:

$$\frac{dS(t)}{dt} = -\beta I(t)S(t), \tag{8.5}$$
$$\frac{dI(t)}{dt} = \beta I(t)S(t) - \gamma I(t), \tag{8.6}$$
$$\frac{dR(t)}{dt} = \gamma I(t). \tag{8.7}$$

The term $\beta I(t)$ is also called the force of infection and $1/\gamma$ is often referred to as the mean infection period.

R-Code 8.4 Basic SIR model, in similar setup as previously. (See Figure 8.3.)

```
sir <- function(time, state, parameters) {
  with(as.list(c(state, parameters)), {
      dS <- -beta * S * I
      dI <-  beta * S * I - gamma * I
      dR <-                 gamma * I
    return(list(c(dS, dI, dR)))
  })
}

init       <- c(S=1-1e-3, I=1e-3, R=0.0)
parameters <- c(beta=1.4, gamma=.14)
Time       <- seq(0, 50, by=1)

out <- ode(y=init, times=Time, func=sir, parms=parameters)
matplot( x=Time, y=out[,-1], type="l", ylab="Proportion", lty=1, col=2:4)
legend( "topright", c("Susceptible", "Infected", "Recovered"), lty=1,
       col=2:4, bty="n")
```

R-Code 8.4 illustrates the SIR model with initial values are: $S = 0.999$, $I = 0.001$, $R = 0$. The recovered class is an absorbing state, so that ultimately the entire population is recovered. If we have a very infectious disease, that has a very high β, the shapes of the curves essentially remain the same, but they squash toward the origin. If the value of β is lower, then the curve $I(t)$ flattens out and it takes longer until the population has recovered. For very low values of β, "nothing" happens"

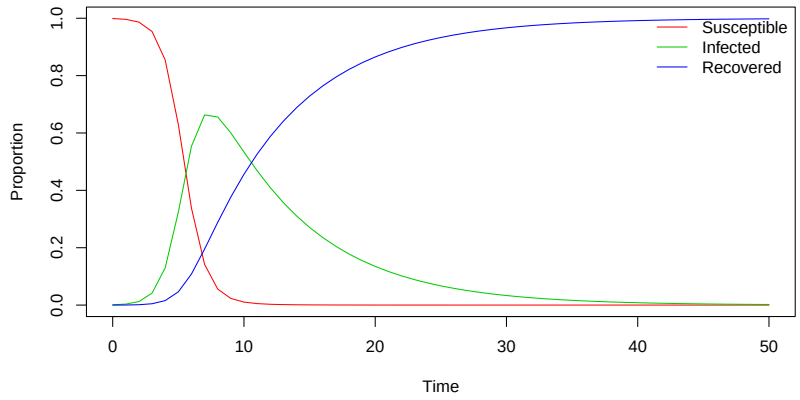

Figure 8.3: Basic SIR model. (See R-Code 8.4.)

Definition 8.1. The *basic reproduction number*, R_0, is the average number of the new infections caused by one infected individual when introduced into a population consisting only of susceptible individuals. ♢

We have the following situations:

- $R_0 < 1$: the number of infected individuals decrease monotonically and the disease fades out.
- $R_0 > 1$: there will be an epidemic outbreak.
- $R_0 = 1$: sharp bound between disease dying out and epidemic outbreak.

For a SIR model, we have $\mathsf{R}_0 = \beta/\gamma$, defining formally what is meant by the last statement before the definition.

8.3.2 SEIR Model and Other Extensions

The classical SIR model has several intuitive generalization:

- Often an infected individual is not directly infectious. This latency period is often model with an additional compartment, E. Thus the chain of the process is $S \longrightarrow E \longrightarrow I \longrightarrow R$, with E for Exposed and I for Infectious.
- Not closed population: birth, death and possibly immigration and emigration terms are added accordingly.
- Carrier: someone carrying disease without showing symptoms but possibly infecting others.
- Not-homogeneous case: for example flue depends on the seasonality, $\beta(t)$.
- External factor of the infections: for example mosquito for malaria.

8.4 Limitations of Compartment Models

As with any (statistical) model limitations exist and – depending on the situation – the model may not be capable of sufficiently well describe the observed data or reality. We list some of the compartment model limitations.

- The system is not closed:

  Additional (unknown) sources or sinks exist. Working with proportions typically helps at a price of transport inaccuracies.

- Homogeneity assumption not satisfied:

  Especially in chemical reaction models the reactants may not be sufficiently well mixed. The contact rate β may depend on the age of the individual.

- Inaccurate balance equation:

  Transports from one compartment to the other may only be known up to a first order approximation. Especially for very small masses parameters behave different.

- Balance of mass not relevant:

  In many real world systems some of the compartments do not really describe the system. A classical example would be a hare/grass example, where the amount of grass cannot be accurately be described by a system.

8.5 Bibliographic Remarks

The lynx/hare dataset is quite common and different versions exists, including the *lynx* dataset from the base package *datasets* or the extensive lynx/hare data available from https://github.com/bblais/Systems-Modeling-Spring-2015-Notebooks/tree/master/data/Lynx%20and%20Hare%20Data. The causes of the cycles have been discussed extensively in the literature, see, e.g., Moran (1949), Krebs *et al.* (2018).

Although of limited quality, the page https://en.wikipedia.org/wiki/Compartmental_models_in_epidemiology gives an accessible overview of the approach.

We can can check some basic production numbers on: https://en.wikipedia.org/wiki/Basic_reproduction_number.

Lab Content

1. Linear 2×2 homogeneous differential equations and their phase plots
2. Stability of linearized 2×2 homogeneous differential equations
3. (Example of phase plot, link to Lorenz model).

Worksheet 8

21. Implementation of a deterministic SIR model.

22. Implementation of a SIR model consisting of time varying coefficients.

23. Equilibrium of Lotka Volterra's equations.

Chapter 9

Cellular Automata

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta111/chapter09.R.

Mathematica file: www.math.uzh.ch/furrer/download/sta111/chapter09.nb.

In this chapter, we study a deterministic population model based on very simple rules. The resulting population or pattern can be classified according to its "complexity". Surprisingly, more complex model do not lead to more complex structures. However, such simple rules can be used to model traffic, wild fires, shell patterns, or even universal computing devices.

In a later chapter we will add a stochastic component and recover SIR-type models, for example.

9.1 Conway's Game of Life

Conway's Game of Life is a two-dimensional cellular automaton that creates intriguing structures. It is one of the best known two-dimensional cellular automata, invented by John H. Conway. Originally played by hand with counters, the coming of computers greatly helped to understand its structure. It is run by placing a number of filled cells on a two-dimensional grid. Each generation then switches cells on or off depending on the state of the cells that surround it. The rules are defined as follows. All eight of the cells surrounding the current one are checked to see if they are on or not. Any cells that are on are counted, and this count is then used to determine what will happen to the current cell.

1. Death/turn off: if the count is less than 2 or greater than 3, the current cell is switched off;

2. Survival/remain on: if either the count is exactly 2, or the count is exactly 3, the current cell is on, the current cell is left unchanged;

3. Birth/turn on: if the current cell is off and the count is exactly 3, the current cell is switched on.

9.2 Cellular Automata

Cellular Automata (CA) are discrete models that are studied in many fields as for example physics, complexity theory or theoretical biology. Such a model consists of

- a discrete space R, e.g., $\mathbb{Z}$, $\mathbb{Z}/n\mathbb{Z}$;
- a neighborhood structure N describing which elements of R determine the states of the subsequent time unit, i.e., of the subsequent generation;
- a state space Q, e.g., $\{0, 1\}$;
- a transition function $\delta : Q^N \to Q$ defining the next generation based upon the current state. t

The transition functions, specifying the state of a cell at the next generation based on the its and its neighbors values, are typically given as a table. Each generation evolves all cells simultaneously. The following example illustrates a simple CA.

Example 9.1. Suppose we consider a sequence of cells, each having either a state 0 (white) or 1 (black). The value of the cell in the next generation depends on its current value and the value of the cell next to it. For example if both have the same state, then the result is zero, otherwise one. Figure 9.1 illustrates the four necessary transition functions and eight generations obtained from the circular starting sequence $\{1, 0, 0, 0, 1, 0, 1, 0, 0, 0\}$. ♣

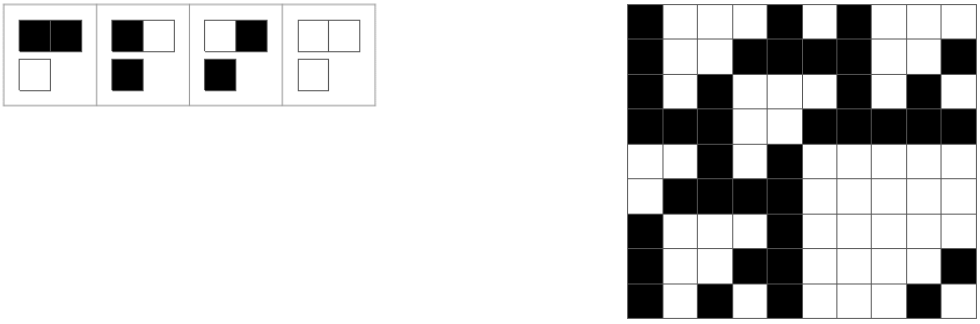

Figure 9.1: Simple cellular automaton. Transition functions (left) and eight generations derived from the circular starting sequence $\{1, 0, 0, 0, 1, 0, 1, 0, 0, 0\}$ (right).

Note that the number of possible transition functions and moreover the number of different possible CA explodes as we complexify the neighborhood structures. For Example 9.1 we have a neighborhood size $k = 2$ with two states and thus four transition functions. We would be able to construct $2^{2^2} = 16$ different CA. We can enumerate these different CA by encoding the binary result of the transition table.

We now extend the neighborhood structure to three cells, L, C, R (left, central, right) determining the central cell of the next generation. In this case with $Q = \{0, 1\}$, the CA are known as an elementary cellular automaton. For elementary CA, we have to specify $2^k = 8$ transition function and there are $2^{(2^k)} = 256$ different CA available. These are again numbered

according to the binary values of the transition table. For example the transition function δ, that only maps to 0 is rule 0, R0 (or that only maps to 1 is rule R255). There are of course more interesting and less trivial rules. and we should distinguish between more interesting and trivial rules. By symmetry, similar patterns will appear several times. We will see later, that we sometimes observe interesting oscillatory patterns.

Figure 9.2 shows the transition functions for four specific CA. The decimal number 30 in binary representation is 11110, which corresponds to the resulting pattern of Rule 30 (top left). Figure 9.3 shows 75 generations

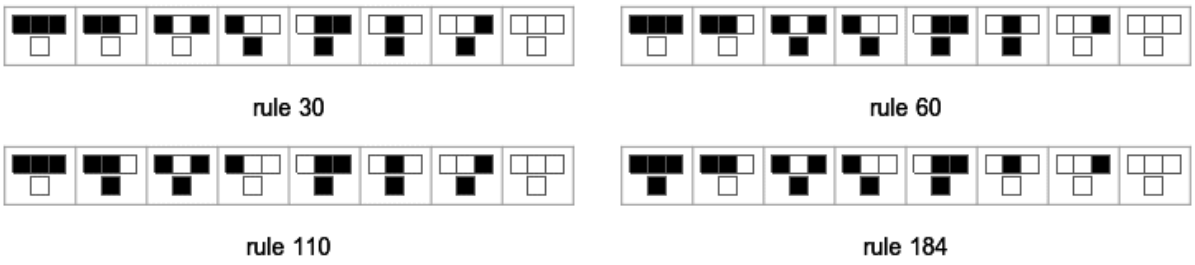

Figure 9.2: Example of elementary cellular automata: tables of transition functions for Rules 30, 60, 110 and 184.

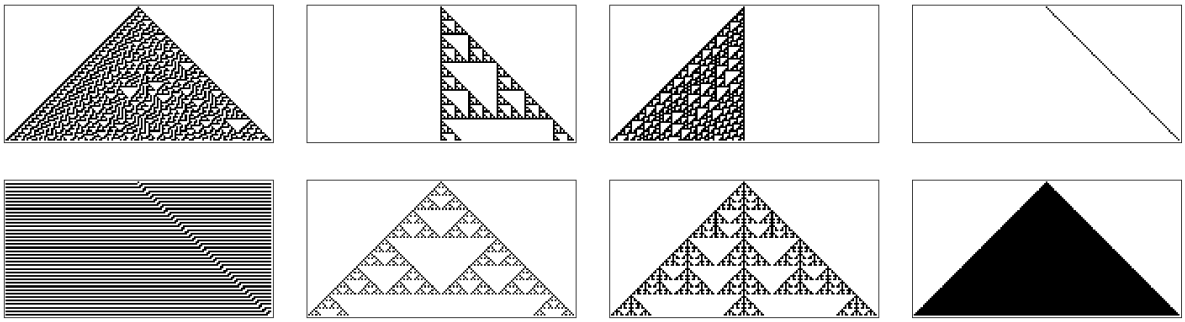

Figure 9.3: 75 generations of different elementary CA, starting with a single cell. Top row are Rules 30, 60, 110 and 184; bottom row represent Rules 11, 90, 150 and 222.

9.2.1 Classification

For elementary CA, nearly all random initial patterns evolve into

1. stable homogeneous state, e.g., Rule 0, Rule 222.

2. stable or oscillating pattern, such that local changes remain local, e.g., Rule 6, Rule 11

3. pseudo random structure or chaotic behavior, such that small local changes can have huge effects, e.g., Rule 30

4. complex structures, e.g., Rule 110.

The third and fourth group are particularly interesting. For example, Rule 30 has been used as one of the pseudo random number generators in Mathematica. Rule 110 is of particular interest, because it is capable of universal computing. That means, in a nutshell, all its structure is rich enough to simulate (emulate) Turing machine computer programs.

Of course an individual, simple starting sequence may not reveal the full breath of a rule. For example, Rule 150 is similar to Rule 30 yet the corresponding panel in Figure 9.3 would rather indicate a rather simple structure. Visually, it is very hard to distinguish patterns from the third and fourth class, compare Figures 9.3 and 9.4.

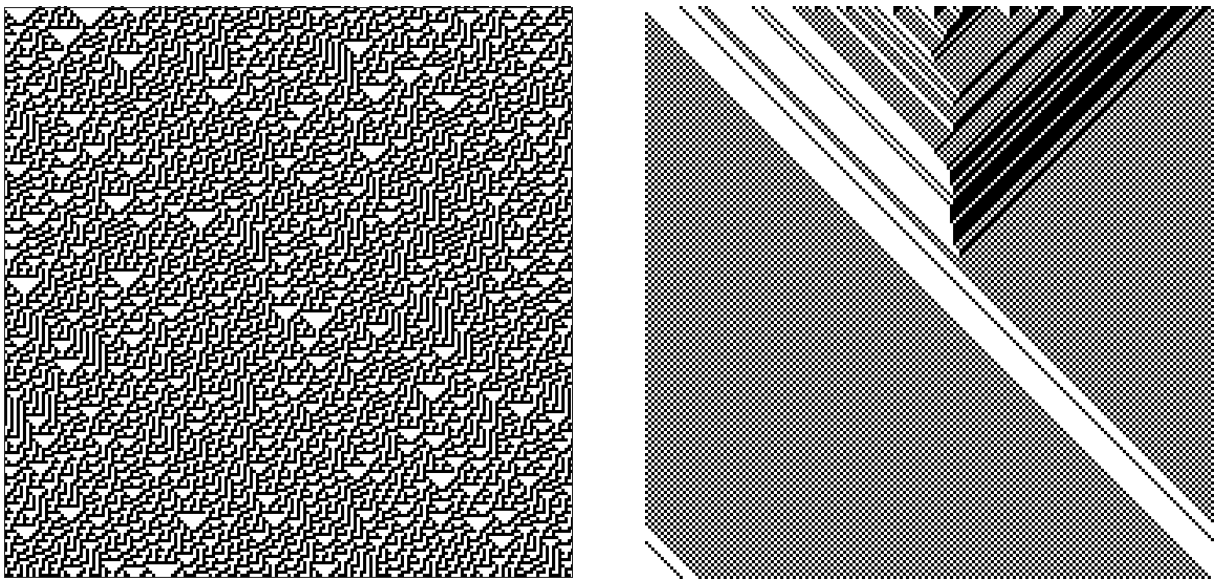

Figure 9.4: Rule 110 (left) and Rule 184 (right) for 200 generations with a random starting sequence, also of length 200.

Nature, however, often seems to produce patterns similar to those from elementary CA. A classical example shown in Figure 9.5 is *conus textile*, a venomous sea snail species.

Figure 9.5: The color pattern of its shell resembles a CA Rule 30. Source https://en.wikipedia.org/wiki/Conus_textile.

9.3 More Advanced Automata

When adding further states to the system, the number of transition functions (and the number of possibly different automata) necessary grows very quickly. One simplification is to "average" the value of the neighbors. For elementary CA, instead of the four possibilities $\{(1,\cdot,1),(1,\cdot,0),(0,\cdot,1),(0,\cdot,0)\}$ we have the averages 1,1/2,0. The simplifications are more pronounced for CA with more states or more complex neighborhood structures. For example, with three states, we reduce the number of transition functions from $3^3 = 27$ to 7 and thus the number of different CA from $3^{3^3} = 7.63 \cdot 10^{12}$ to $3^7 = 2187$. CA depending on the average of the neighbors only are called totalistic CA.

Interestingly, working with three states does not lead to more "complex" systems. Often, periods or burnin-time are longer. Such an example is also given in the Mathematica notebook.

9.3.1 Moving Automata

Moving automata are very similar to CA but they have a single active cell which is the only cell that is updated from one generation to the next generation.

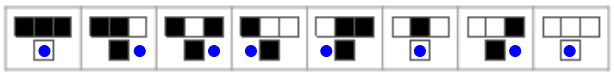

Figure 9.6: Subset of a moving automata rules. The active cell is indicated by the blue dot.

9.3.2 Turing Machine

In the context of CA, we see a Turing machine as a moving automata where the active cell may have different states. Figure 9.7 illustrates the transitions functions in the setting of a "zero" neighborhood and a simple example.

From a mathematical or computer science point-of-view more formal definitions of Turing machines are available, essentially as here but with a "Halt" state and a blank symbol. Turing machines as introduced here can emulate the CA seen above or substitution systems (shown next).

9.3.3 Substitution Schemes

All automata presented above had a fixed number of cells where the state thereof possibly changed. In a substitution system, individual cells are replaced by a sequence of cells. For a one dimensional automata, an insertion can be envisioned as moving all other cells further off (the space is growing). Figure 9.8 illustrates such a simple rule that essentially increases the space between marked cells (left). The second example shown in Figure 9.8 constructs a similar shape as Rule 90, a Sierpiński triangle type shape.

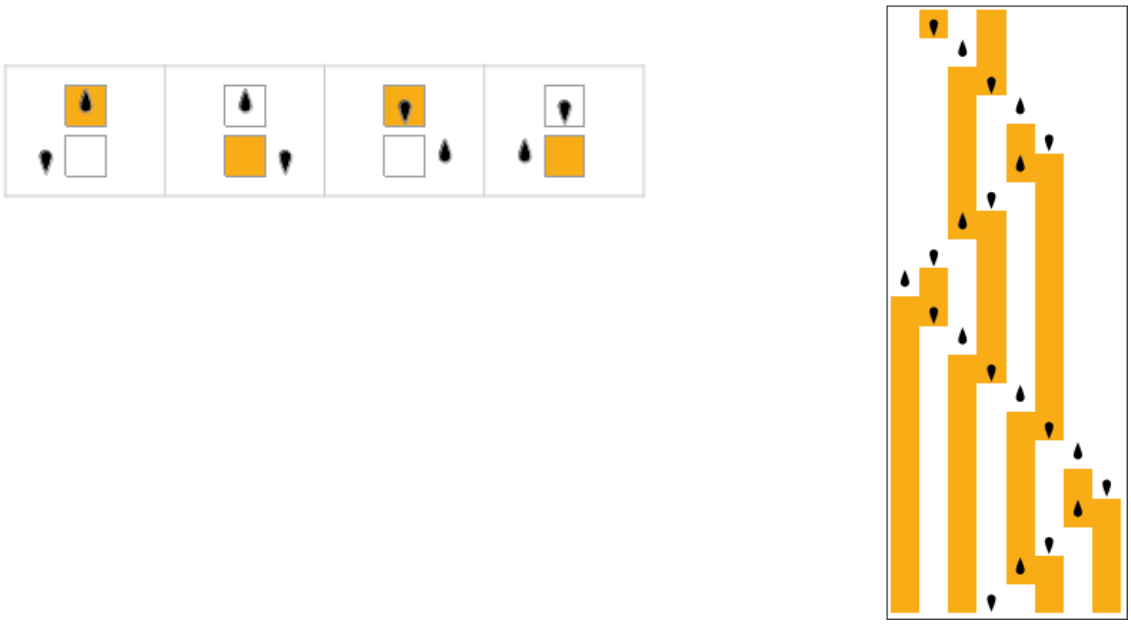

Figure 9.7: Turing machine rule (left) and a example (right).

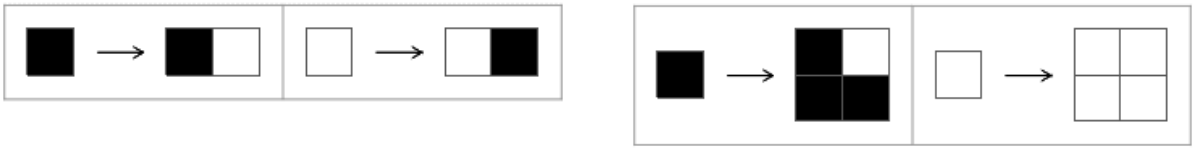

Figure 9.8: One-dimensional SS (left), two-dimensional SS (right).

Remark 9.1. The Sierpińsky triangle can be constructed via a simple substitution rule: starting from an an equilateral triangle, every new generation subdivides each triangle into four (smaller) congruent equilateral triangles and removing the central one.

An alternative construction with the Sierpińsky triangle is the limiting shape is as follows. Start with a single line segment and for each generation, replace each line segment of the curve with three shorter segments, forming 120° angles at each junction between two consecutive segments. The first and last segments of the curve either parallel to the original line segment or forming a 60° angle with it.

Substitution schemes are closely related to iterative function systems or L-systems, mimicking growth of plants. Figure 9.9. We come back to such type of objects in Chapter 11. ♣

9.4 Two-dimensional Domains

Last section presented CA in one dimension. We now extend the domain in two-dimensions. For visual illustrations, we often consider a grid wrapped on a torus. Elements leaving one face, reappear on the opposite one. Typical neighborhood structures are the eight surrounding cells, or the four nearest neighbors.

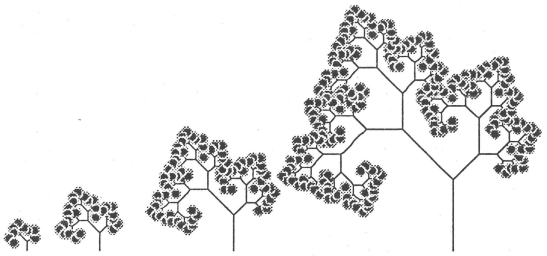

Figure 9.9: Four generations of a substitution system mimicking growth of a tree. Source Amherd *et al.* (1992).

9.4.1 Conway's Game of Life

As introduced in the first section, Conway's Game of Life is a two-dimensional totalistic cellular automaton. In the setting of an eight neighborhood structure and two states, the average may take 9 values: 0,1/8,2/8,...,1. Thus we have to define 18 transitions, shown in Figure 9.10. These transitions are equivalent to the "rules" given in Section 9.1.

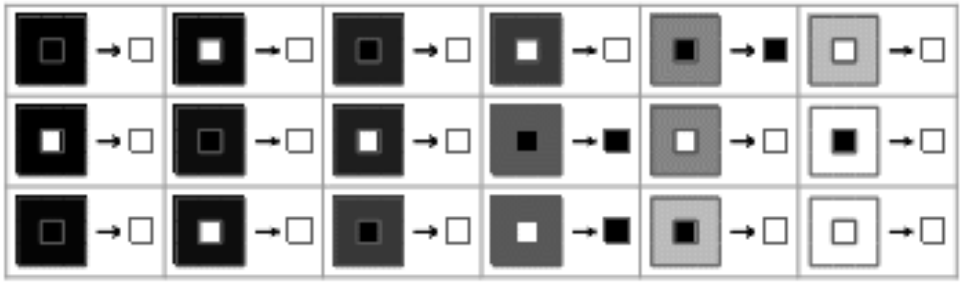

Figure 9.10: Totalistic rules of Conway's game of life.

Similar to Rule 110, the game of life exhibits complex structures and is capable of universal computing. It is possible to study small pattern that oscillate or glide.

9.4.2 Biham–Middleton–Levine Traffic Model

The Biham–Middleton–Levine traffic model (Biham *et al.*, 1992) is a self-organizing cellular automaton traffic flow model that can be interpreted as the two-dimensional Rule 184 model. Cars are randomly placed on a two-dimensional grid, wrapped on a torus. Two types of cars are modeled: On the one hand cars going from left to right and on the other hand cars going from up to down. Cars advance one step, if the corresponding cell is not blocked by another car. For low densities the free flow establishes. For higher densities, congestion start to occur, even at a level to full blocking. It is remarkable that even though the model is completely deterministic, we recognize self-organizing features (minor congested areas free up).

Figure 9.11 illustrates a high (38%) and low density (27%) snapshot of the model. Blue cars move from top to bottom, red from left to right.

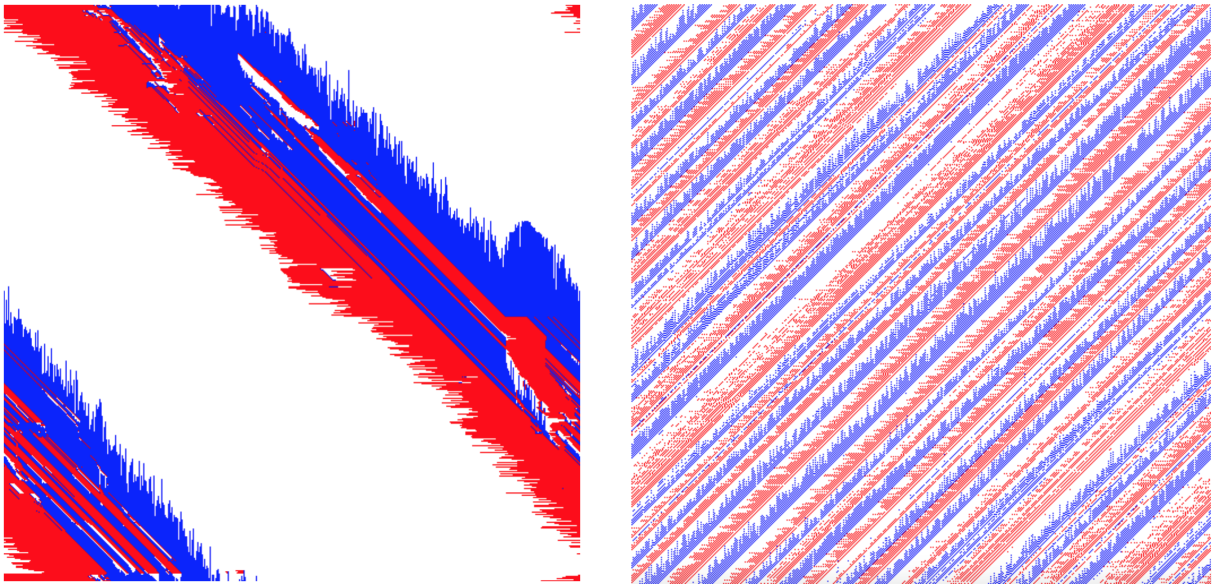

Figure 9.11: Biham–Middleton–Levine Traffic model: congestion (left) and self-organizing in stripes (right). Blue cars move from top to bottom, red from left to right, 512×512 lattice wrapped to a torus.. Source https://en.wikipedia.org/wiki/Biham-Middleton-Levine_traffic_model.

9.5 Self-organization and Artificial Intelligence

(The following paragraphs are adapted (some almost verbatim) from Wikipedia.)

Self-organization is a process where some form of overall order arises from local interactions between parts of an initially disordered system. The process is spontaneous, not needing control by any external agent. It is often triggered by random fluctuations, amplified by positive feedback. The resulting organization is wholly decentralized, distributed over all the components of the system. As such, the organization is typically robust and able to survive or self-repair substantial perturbation.

Intelligence can been loosely defined as the capacity for logic, understanding, self-awareness, learning, emotional knowledge, reasoning, planning, creativity, and problem solving. It can be more generally described as the ability to perceive or infer information, and to retain it as knowledge to be applied towards adaptive behaviors within an environment or context.

Intelligence displayed by humans or animals is typically termed natural intelligence. Artificial intelligence denotes (elements of) intelligence demonstrated by algorithms or machines.

Therefore, self-organization is a necessary process of an intelligent system. Rule 184 exhibits self-organization, much more visible in the two-dimensional traffic models.

From the list of capacities, only 'logic' is clearly visible in cellular automata (see Example 9.1 for XOR operation). Not surprisingly, elementary CA exhibit AND operations similarly. Of the 256 elementary cellular automata eight are additive: $\{0, 60, 90, 102, 150, 170, 204, 240\}$. All of these are either trivial or essentially equivalent to rules 90 or 150 (Wolfram, 2002).

9.6 Bibliographic Remarks

The book "A New Kind of Science" by Wolfram (2002) is a very accessible treaty of CA and goes much beyond. The online version is accessible here http://www.wolframscience.com/nks/. However, the scientific reviews have been quite mixed.

Additional accessible references are from Wikipedia or Mathworld. A selection thereof is as follows. Cellular automaton https://en.wikipedia.org/wiki/Elementary_cellular_automaton, https://en.wikipedia.org/wiki/Cellular_automaton and http://mathworld.wolfram.com/ElementaryCellularAutomaton.html. Conway's Game of Life: https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life and life in life: https://www.youtube.com/watch?v=xP5-iIeKXE8

Several rules are discussed on details on individual pages. For example Rule 30: https://en.wikipedia.org/wiki/Rule_30 (and similarly http://mathworld.wolfram.com/Rule30.html), Rule 110: https://en.wikipedia.org/wiki/Rule_110 (http://mathworld.wolfram.com/Rule110.html) or Rule 184: https://en.wikipedia.org/wiki/Rule_184 .

The "Biham Middleton Levine traffic model" is discussed in details at https://en.wikipedia.org/wiki/Biham-Middleton-Levine_traffic_model.

Lab Content

1. Odds and ends.

Worksheet 10

24. Generate pseudo random numbers with a cellular automaton.

25. Implement Conway's Game of Life.

26. Game of Life Spaceship speed.

Chapter 10

Stochastic Compartmental Models and Stochastic CA

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta111/chapter10.R.

Mathematica file: www.math.uzh.ch/furrer/download/sta111/chapter10.nb.

Although cellular automata exhibit stochastic like behavior, they are fully deterministic. We now add a stochastic component to compartmental models and cellular automata.

10.1 Wildfire modeling

Modeling wildfires can be done by considering a grid of cells where a successive change of the cell properties occur, from, say, changing a cell with available fuel to ignite and burned. Possibly, we have different amount of fuel available and ignition might not be fully deterministic.

To model the spread of a wildfire, we can set up a model with the following states

- V_1: low burning mass
- V_2: high burning mass
- F_1: fire started
- F_2: fire can spread
- B_1: burned/no fire

Initially, all states are in either V_1 or V_2, and a small subset in, typically one cell in F_1. It still remains to define a neighborhood structure N, as well as the transition function δ, which is possibly stochastic. In such settings, it is possible to obtain states resembling images as shown in Figure 10.1.

Of course it is possible to define more states, often the images only marginally improve.

Figure 10.1: Satellite image of a wild fire. Source: https://cdn.satellitetoday.com/wp-content/uploads/2017/10/631_crop_chip1.jpg.

10.2 Compartmental Models

There are two natural ways to extend deterministic compartmental models to stochastic setting. The simpler one is keeping time discrete and modeling infection by a stochastic process. The second is modeling continuous time framework and time to infection is stochastic. We illustrate the two settings in the following two sections.

10.2.1 Reed–Frost Model

We first extension starts with a so-called *chain binomial model*, were the dynamics is governed by a simple binomial process, i.e., an infection spreads through direct contact (independently and with constant probability). An infectious individual spreads the disease via a chain of infections. A prototype of such a model is called the Reed–Frost epidemic model.

The Reed–Frost model can be characterized as follows.

- It is a compartment model where individuals are either susceptible, infectious or recovered.
- The population size is closed (constant) with initially $S_0 = s_0 \in \mathbb{N}$ susceptibles and $I_0 = i_0 \in \mathbb{N}$ infectious individuals.
- The dynamic is described by the discrete time Markov chain

$$\begin{cases} I_{t+1} \mid S_t = s_t, I_t = i_t \sim \mathcal{B}in\left(s_t, 1-(1-\omega)^{i_t}\right), \\ S_{t+1} = S_t - I_{t+1}, \end{cases} \tag{10.1}$$

  where $t = 1, 2, \ldots$ are time steps and ω is the probability of an infectious contact between an infected (thus infectious) individual and a susceptible individual during one time step.
- The final size of the epidemic is $Z = \sum_{i=0}^{\infty} I_i$.

Formally, if ω is the probability of infection, $1-\omega$ the probability of escape and thus $(1-\omega)^i$ is the probability to escape all i contacts. Finally, $1-(1-\omega)^i$ the probability of infection.

The Reed–Frost model can be seen as a SIR model, where the incubation period as well as the recovery time is (roughly) one time unit. The basic reproduction number is essentially ωs_0.

Natural questions in such frameworks are: What is the distribution of the final sizes? What is the expected final size of the epidemic? Example 10.1 and associated R-Code 10.1 illustrate the model and give a glimpse to answers of these questions.

As the dynamics are given by a simple form, it is possible to write the likelihood of the model as

$$L\big(\omega; \{i_0, i_1, \ldots, i_T, s_0\}\big) = \prod_{t=0}^{T-1} \theta_t^{i_{t+1}} (1-\theta_t)^{s_t - i_{t+1}}, \qquad \theta_t = 1 - (1-\omega)^{i_t}, \tag{10.2}$$

where T is the number time-steps with observations, often the smallest time t for which $i_t = 0$. That means, there is no infectious individual left, that can further spread the disease.

As we have a closed population, we can construct the sequence of susceptibles based on the infections. The second part of R-Code 10.1 illustrates the estimation of ω using a simple set of artificial data.

Example 10.1. Consider a population with 20 individuals where initially one individual is infectious. We simulate the epidemic for different probabilities of infections $\omega = 0.03, 0.07, 0.17$. R-Code 10.1 empirically determines the expected final size of the epidemic and the distribution of the final sizes. We repeat each setting 1000 times and plot a histogram of the different final sizes of the epidemic.

Using maximum likelihood techniques we estimate ω together with a 95% confidence interval if the following data is observed: $s_t = \{19, 18, 12, 4, 2, 0\}$, for $t = 0, \ldots, 5$ and one infected individual at $t = 0$. Thus, $i_t = \{1, 1, 6, 8, 2, 2\}$. For a more stable estimation and for better confidence intervals, the approach uses a logit transformation.

R-Code 10.1: Illustration of Reed–Frost model (See Figure 10.2.)

```
reed_frost_1step <- function(w, s, i){
    i_new <- rbinom(1, s, 1 - (1 - w)^i)
    s_new <- s - i_new
    c(s_new, i_new)
}

reed_frost <- function(w, s0, i0){
    current <- c(s0, i0)
    out <- matrix(current, ncol=2)
    while(current[2] > 0){
        current <- reed_frost_1step(w, current[1], current[2])
```

```
        out <- rbind(out, current)
    }
    dimnames(out) <- list(paste(1:nrow(out)), c("S","I"))
    out
}

w <- c(0.03, 0.07, 0.17)
N_sim <- 1000
for(i in seq_along(w)){
    sim <- replicate(N_sim, {
        final_size <-  sum(reed_frost(w=w[i], 19, 1)[,2])})   # 19+1=20
    hist(sim, breaks=0:21-0.5, freq=FALSE, main=paste("w =", w[i]),
         xlab="# infected")
    abline(v=mean(sim), lwd=2, lty=2, col="darkblue")
}
# Likelihood calculation
log_likihood_reed_frost <- function(w_logit, s, i){
    stopifnot(identical(length(s), length(i)))
    w <- plogis(w_logit)
    theta <- 1 - (1 - w)^i
    k <- length(s)           # k=T-1
    sum(dbinom(x=i[-1], size=s[-k], prob=theta[-k], log=TRUE))
}

s <- c(19, 18, 12, 4, 2, 0)
i <- c( 1,  1,  6, 8, 2, 2)
log_likihood_reed_frost(0, s, i)
## [1] -33.312
mle <- optim(par=0, fn=log_likihood_reed_frost, method="BFGS", s=s, i=i,
             control=list(fnscale=-1), hessian=TRUE)
print( unlist( mle[ c(1,2,4,6)]))
##         par       value convergence     hessian
##     -1.5741    -10.6891      0.0000    -14.8033
(w_hat <- plogis(mle$par))
## [1] 0.17163
(w_ci <- plogis(mle$par + c(-1, 1) * qnorm(.975) * sqrt(-1/mle$hess)))
## [1] 0.11071 0.25642
```

The Reed–Frost model can be easily extended to more complicated models.

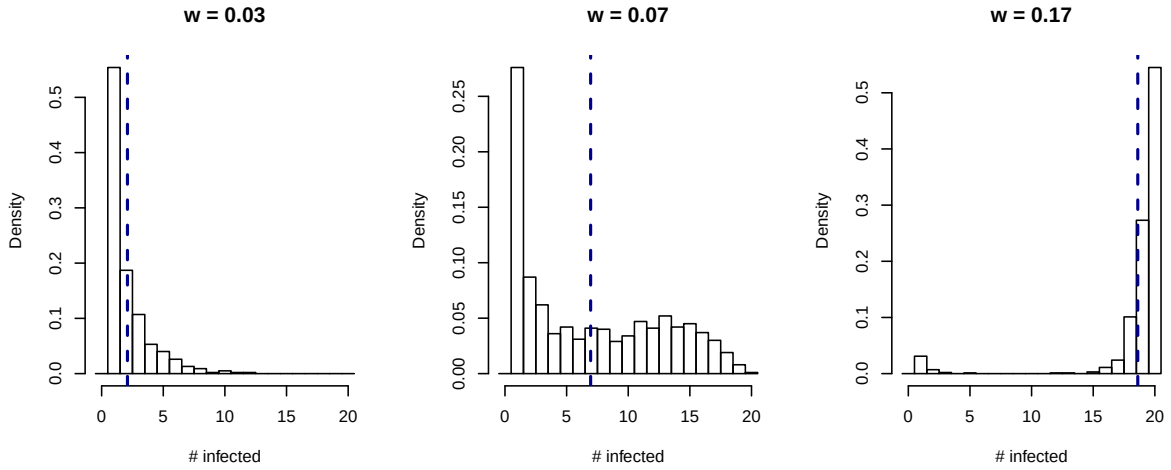

Figure 10.2: Histograms of final sizes of epidemic for a Reed–Frost model with different infection probabilities. Population size 20. (See R-Code 10.1.)

Remark 10.1. There is a direct link to birth-death processes. Assume we have a large population of size n with a small infection probability p (formally, $n > 30$, $p < 0.05$ or $n > 100$, $np < 10$). Then $\mathcal{B}in(n, p) \approx \mathcal{P}oisson(np)$. Thus, we are back to Poisson processes as seen in Chapter 6. ♣

10.2.2 Gillespie Algorithm

We now consider the second approach to extend deterministic compartmental models to stochastic setting.

In the classical setting of deterministic compartmental models, time and state is continuous. We now use a similar construct as for CTMC and model individual changes, although we do not follow explicitly each individual. Similar as for CTMC we use exponential "holding" times (i.e., the times individual stay in the corresponding compartments). After an event, all "clocks" are set back (memoryless property of the exponential random variable). Then we wait until the next event happens, which is an exponential random variable with rate being the sum of all individual rates (minimum of a sequence of independent exponential random variable is an exponential random variable with rate being the sum of all individual rates). In a second step, we determine which event occurred.

The implementation of this algorithm is termed the Gillespie algorithm. R-Code 10.2 illustrates the output of such a simulation.

R-Code 10.2: Gillespie algorithm for a simple SIR model. (See Figure 10.3.)

```
source(file="data/gillespie_fct_R.R")   # Helping functions by G. Kratzer
# set parameters
parms <- c(m=1e-4, b=0.02, v=0.1, r=0.3)
```

```
X0 <- c(S=97, I=3, R=0)
time.window <- c(0, 100)

# define how state variables S, I and R change for each process
processes <- matrix( c(1,1,1, -1,0,0,  0,-1,0,  0,0,-1, -1,1,0,  0,-1,0,
   0,-1,1), nrow=7, ncol=3, byrow=TRUE,  dimnames=list(c("birth",
   "death.S", "death.I", "death.R", "infection", "death.infec", "recovery"),
   c( "dS","dI","dR")))

res <- gillespie( parms, X0, time.window, processes)
matplot( x=res[,1], y=res[,-1], type="l", lty=1, xlab="Time", ylab="Number")

restt <- apply( res, 2, diff)[,2:4]
ttt <- apply( restt, 1, paste0, collapse='')
table( ttt)                      # Number of transitions.
## ttt
## 0-10 0-11 -110
##   19   79   95
NbTraj <- 50
dres <- dim(res)[1]
resarr <- array(0, c(dres, 4, NbTraj))
for (i in 1:NbTraj) {
    res <-  gillespie( parms, X0, time.window, processes)
    tres <- dim(res)[1]
    matlines( x=res[,1], y=res[,-1], lty=1)
    resarr[ 1:min(dres,tres),,i] <-  as.matrix( res)[1:min(dres,tres),]
}
library(RColorBrewer)
rf <- colorRampPalette( rev( brewer.pal( 11, 'Spectral')))
library(hexbin)
h <- hexbin( x=resarr[,1,], y=resarr[,3,], xbins=30)
plot( h, colramp=rf, xlab="Time", ylab="Number", main="Infected")
```

Remark 10.2.

- For huge populations the Gillespie algorithm does not work well as it lacks variability.

- The basic reproduction number is $R_0 = bN/(m+v+r)$.

♣

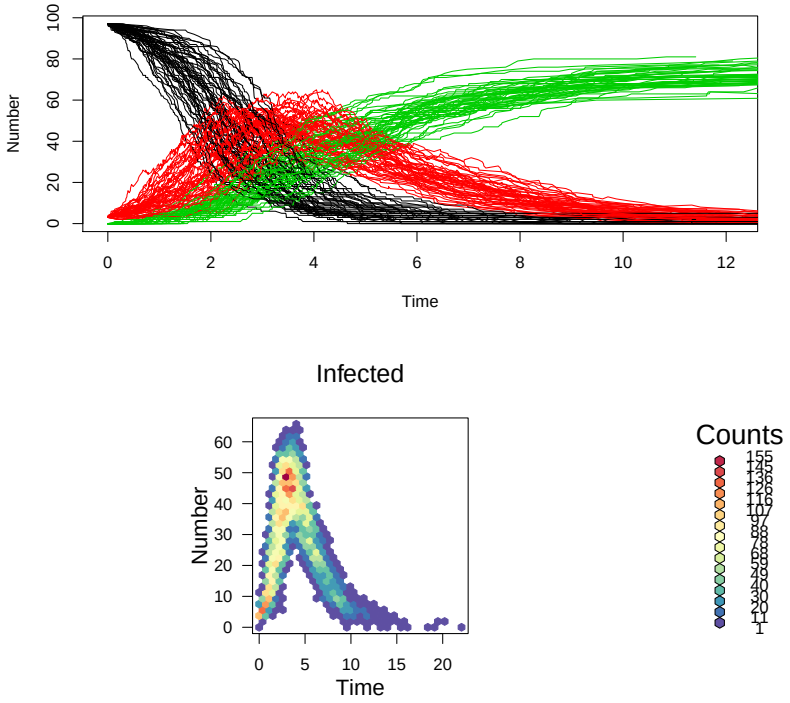

Figure 10.3: Some words (See R-Code 10.2.)

10.3 Probabilistic Cellular Automata

Probabilistic cellular automata or stochastic cellular automata are CA whose updating rule is stochastic: the new states are chosen according to some probability.

As in the setting of deterministic CA, probabilistic CA are capable of self-organization and depending on the setup show flocking behavior, pattern formation, and self-maintenance.

It is possible to show that probabilistic are Markovian. Examples include the Ising model, which we revisit in Chapter 12. Another common area to apply probabilistic CA is modeling of protein folding or membrane building.

10.3.1 SIR modeling

The SIR model seen in Chapter 8, reduced to a SI model can be considered as a cellular automaton with state space $\{S, I\}$. We can define the neighborhood N as L, C, R interpreted as left neighbor (L), the person itself (C) and the right neighborhood (R). The transition function δ could be for example a function that assigns I in the next time step if the present state is I. This can be interpreted in the sense that once you are infected, you stay infected. If δ is stochastic instead of deterministic (i.e. δ is a random variable), we enter the field of *Probabilistic Cellular Automata* (PCA).

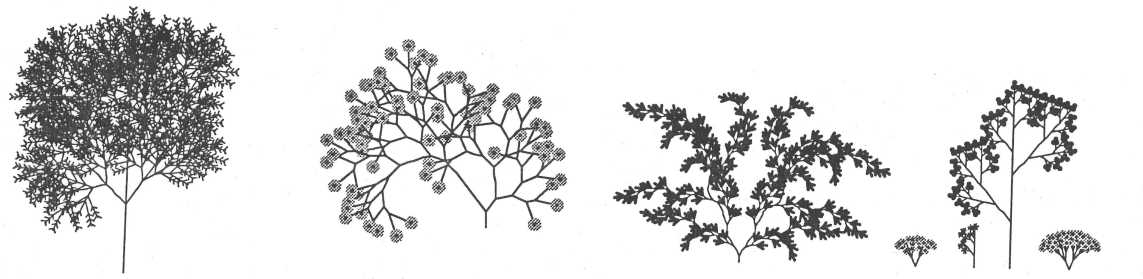

Figure 10.4: Substitution systems with random branch length and angles. Source Amherd *et al.* (1992).

10.3.2 Substitution Systems

Adding random branch length or angles, results in very realistic graphical trees. Albeit slightly different substitution rules, Figure 10.4 illustrates some simple illustrations.

10.3.3 Wildfire modeling

See exercises.

10.3.4 Predator-Prey Type Models

There are several two-dimensional cellular automata that mimic predator-prey relations. A classical example is Wa-Tor. Visualization of such automata have gained substantial popularity through screen savers.

10.4 Bibliographic Remarks

Wa-Tor: https://en.wikipedia.org/wiki/Wa-Tor and to play with parameters: https://www.cheesygames.com/wator/ Dewdney (1984) "Sharks and fish Wage an ecological War on the toroidal planet Wa-Tor", Scientific American December pg I422. http://home.cc.gatech.edu/biocs1/uploads/2/wator_dewdney.pdf.

Lab Content

To be determined by attendees.

Worksheet 10

27. A continuous time Birth-Death process in a stochastic framework.

28. Model wideland fire dynamics using a stochastic cellular automaton.

29. Linearization of Lorenz model and identification and simulation of stability points.

Chapter 11

Brownian Motion and Self-similarity

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta111/chapter11.R.

Self-similarity is visual in nature in, for example, farns, Romanesco broccoli. In mathematics, self-similar objects such as the Shierpinski triangle, the Koch curve or Julia sets have been popularized and resulting visualizations are known to more than specialists.

In this chapter we extend the concept of random walks to Brownian motions and link Brownian motions to self-similarity. Of course, we can only sketch certain aspects thereof.

11.1 Length of Switzerland's Boundary

According to Wikipedia, Switzerland as a total area of 41 285 km^2 (en.wikipedia.org/wiki/Switzerland) but what is the length of its boundary (not given in on the same page)? Of course it is considerably larger than 720 km (lower bound matching area of disc and circumference). Using basic *maps* data, the boundary is approximately 1 250 km, whereas is over 1 600 km for higher resolution data from the package *mapdata*.

The resolution determines the length, see also R-Code 11.1 which is based on official boundaries, determining the boundary to virtually any precision. Other countries like England or more general the island Great Britain would rather ask what is their coastal length. In such setting, it is possible to virtually zoom in to an "arbitrary" level, ending at molecular level only. Boundaries at these resolutions are of course not recorded but suggest that we actually have "arbitrary" long coastal lengths.

R-Code 11.1: Boundary of Switzerland at different resolutions and estimate of the Hausdorff dimension. (See Figure 11.1.)

```
load('data/swissBoundaries.RData')
plot( CHcoords, type='l',xaxt='n', yaxt='n')
```

```
colnames( CHcoords) <- c('x','y')
sum( sqrt( diff(CHcoords[,'x'])^2 + diff(CHcoords[,'y'])^2 ), na.rm=TRUE )
## [1] 1933360
lseq <- c(2^c(0:9))
nlseq <- length( lseq)
res <- data.frame(Skip=lseq, Length=0, Resolution=0)
for (i in 1:nlseq) {
    out <- CHcoords[seq(from=1,to=44415, by=lseq[i]),]
    tmp <- sqrt( diff(out[,'x'])^2 + diff(out[,'y'])^2 )
    res[i,2:3] <- c(sum( tmp)/1000, median( tmp))
    lines( out, col=i)
  }
plot( Length~Resolution, log='xy', data=res)
all <- lm(log(Length)~log(Resolution),  data=res)
lines( res$Resolution, exp(all$fitted))
fir <- lm(log(Length)~log(Resolution),  data=res, subset=1:5)
lines( exp(fir$model[[2]]), exp(fir$fitted))
las <- lm(log(Length)~log(Resolution),  data=res, subset=6:10)
lines( exp(las$model[[2]]), exp(las$fitted))
# Not covered in class:  Fractal dimension D, given by
#       Length( Resolulution) = M * Resolution^(1-D)
# is somewhere between
1-c(coef(fir)[2], coef(las)[2])
## log(Resolution) log(Resolution)
##          1.0503          1.1252
#   ... but significant ;-)
```

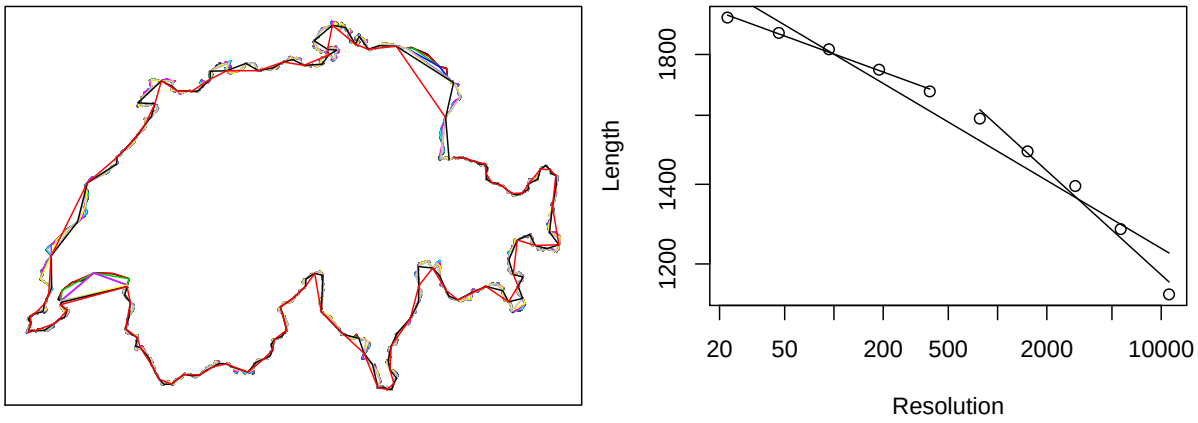

Figure 11.1: Boundary of Switzerland with different resolutions. (See R-Code 11.1.)

11.2 Brownian Motion (BM)

In previous chapters, we have considered random walks on integers or on regular lattices, where, by construction, we performed unit steps. We now extend this discrete walk to "continuous jumps".

We start in one dimension and set up again the classical random walk

$$X_i = \begin{cases} +1, & \text{with probability } p, \\ -1, & \text{with probability } 1-p. \end{cases} \tag{11.1}$$

For simplicity we assume $p = 1/2$ throughout the chapter. Instead of such unit moves in time and in space, we define two (arbitrarily) small values Δt and Δx respectively. We define

$$X(t) = (X_1 + X_2 + \cdots + X_{\lfloor \frac{t}{\Delta t} \rfloor}) \times \Delta x, \tag{11.2}$$

which is a stochastic process, continuous in time, having moments $\mathrm{E}\big(X(t)\big) = 0$ and

$$\mathrm{Var}\big(X(t)\big) = \Delta x^2 \left\lfloor \frac{t}{\Delta t} \right\rfloor \mathrm{Var}(X_1) \approx \Delta x^2 \Big(\frac{t}{\Delta t}\Big) \mathrm{Var}(X_1) = \Delta x^2 \Big(\frac{t}{1/\sigma^2 \Delta x^2}\Big) = \sigma^2 t, \tag{11.3}$$

where we used $\Delta x = \sigma\sqrt{\Delta t}$ for the last equality. We now use a limit argument and let $\Delta t \to 0$ to formally define the process $B(t)$.

Definition 11.1. The process $\{B(t); t \geq 0\}$ is a Brownian motion or a Wiener process if

1. $B(0) = 0$ almost surely,
2. $B(t)$ has homogeneous and independent increments,
3. $B(t) \sim \mathcal{N}(0, \sigma^2 t)$.

If $\sigma = 1$, then $\{B(t); t \geq 0\}$ is a standard Brownian motion. ♢

The following property states that the increments are also Gaussian (not so surprising) and gives expressions for the covariance and correlation (slightly more surprising).

Property 11.1. *Let $\{B(t); t \geq 0\}$ a Brownian motion, then $B(t) - B(s) \sim \mathcal{N}(0, \sigma^2|s-t|)$ and*

$$\mathrm{Cov}\big(B(t), B(s)\big) = \sigma^2 \min(s,t)\,, \qquad \mathrm{Corr}\big(B(t), B(s)\big) = \sqrt{\frac{\min(s,t)}{\max(s,t)}}\,. \tag{11.4}$$

Based on the independent increment assumption and based on (11.4) it is possible to show that for any set $t_1 < \cdots < t_n$ the vector $\big(B(t_1), \ldots, B(t_n)\big)^\top$ is has a multivariate Gaussian distribution.

Brownian motions are widely used in economics, quantitative finance, physics and – of course – statistics. Without going into the details, the following gives a glimpse of the versatility.

Property 11.2. *1.* $\{B(t); t \geq 0\}$ *is a Markov process.*

2. $\{B(t); t \geq 0\}$ *is a Gaussian process.*

3. $\{B(t); t \geq 0\}$ *is mean-square continuous, i.e.,* $\mathrm{E}\big((B(t+h) - B(t))^2\big) \xrightarrow{h \to 0} 0$.

4. $\{B(t); t \geq 0\}$ *is a martingale.*

Since a Brownian motion is mean-square continuous, it is not surprising that its sample paths $b(t)$ are continuous functions in t. However, the sample paths $b(t)$ are nowhere differentiable, which can be seen, heuristically, by

$$b'(t) \approx \frac{b(t+\Delta t) - b(t)}{\Delta t} \approx \frac{c \cdot \sigma\sqrt{\Delta t}}{\Delta t} = \frac{\sigma}{\sqrt{\Delta t}} \xrightarrow{\Delta t \to 0} \infty, \tag{11.5}$$

where we have linked the step sizes with the corresponding standard deviation of the Brownian motion.

Remark 11.1. The continuity of sample paths should be phrased rigorously by the sample paths are continuous with probability one.

The fact that a Brownian motion has 'unbounded variation', i.e., sum of arbitrary small increments of $b(t_i)$ is unbounded, implies that the 'length' of the sample path between $b(s)$ and $b(t)$, $s < t$, is infinite. ♣

In a similar fashion as we have extended random walks from one to several dimensions (i.e., from $\mathbb{Z}$ to $\mathbb{Z}^d$, $d > 1$), we can define Brownian motions in higher dimensions. R-Code 11.2 illustrates the construction and visualization of a Brownian motion in two dimensions. By construction, the sample path is continuous.

R-Code 11.2: Brownian motion in two dimensions. (See Figure 11.2.)

```
N <- 2^18                                # total number of segments
set.seed( 214)
Xt <- matrix( rnorm(2*N), ncol=2)        # individual increments
Bt <- rbind(0, apply( Xt, 2, cumsum)) # summed to form the process
lim <- range( Bt)
plot( Bt, type='l', xlab='', ylab='', xlim=lim, ylim=lim)
points( Bt[c(1,N),1], Bt[c(1,N),2], col=c(2,3), pch=19)
```

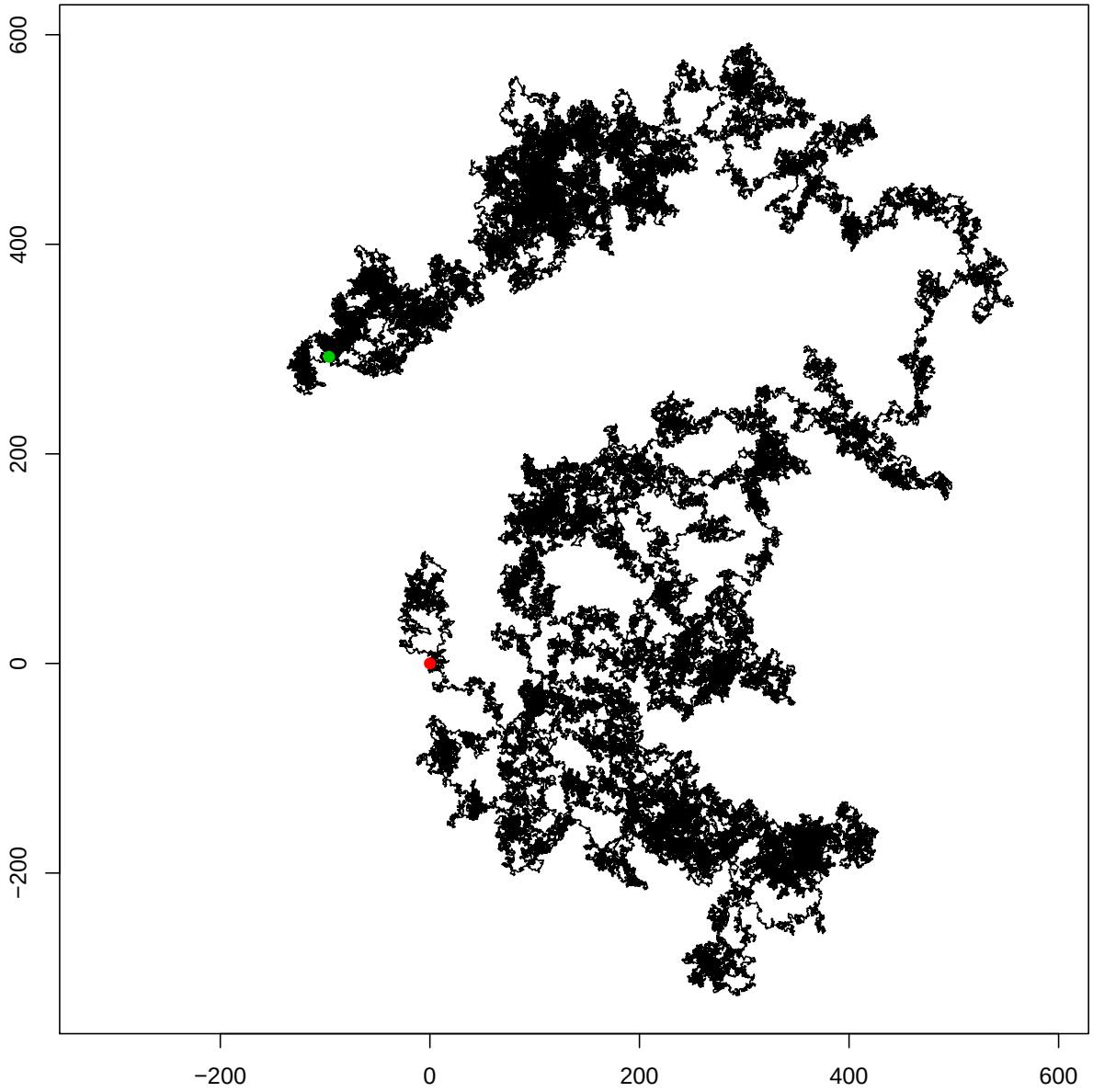

Figure 11.2: Brownian motion in the plane. There are $2^{18} = 262\,144$ segments between the red starting point and green end point. (See R-Code 11.2.)

11.3 Self-similarity

A geometric object is said to be self-similar if parts of it look roughly like the entire object.

Self-similar objects may have fractional dimensions.

For instance, the Koch curve summarized earlier is constructed from an equilateral triangle; in each iteration, its component line segments are divided into 3 segments of unit length, the newly created middle segment is used as the base of a new equilateral triangle that points outward, and this base segment is then deleted to leave a final object from the iteration of unit length of 4. That is, after the first iteration, each original line segment has been replaced with $N = 4$, where each self-similar copy is $1/S = 1/3$ as long as the original. Stated another way, we have taken an object with Euclidean dimension, D, and reduced its linear scale by $1/3$ in

each direction, so that its length increases to $N = S^D$.

Example 11.1. The graphical trees of Figures 9.9 and 10.4 are self-similar. ♣

Property 11.3. *For a one-dimensional Brownian motion* $\{B(t); t \geq 0\}$*, we have*

1. $\frac{1}{\sqrt{c}}B(ct)$ *is a Brownian motion, hence a Brownian motion is self similar.*
2. $tB(1/t)$*, for* $t > 0$ *(and 0 for* $t = 0$*) is a Brownian motion, denoted* time inversion *property.*
3. *We have fractal dimension* $D = 2 - H = 2 - \frac{1}{2} = 1.5$*, where* H *is the Hurst exponent. (Values of the Hurst exponent vary between 0 and 1, with higher values indicating a smoother trend, less volatility, and less roughness.)*

Remark 11.2. The path or graph of Brownian motions in two and higher dimensions, have Hausdorff dimension 2. Notice the similarity with recurrence of random walks in one, two and higher dimensions.

As side notes, the Hausdorff dimension of the boundary of a two-dimensional Brownian motion has Hausdorff dimension 4/3.

♣

R-Code 11.3 illustrates the self-similarity of the Brownian motion by zooming into the sample path shown in Figure 11.2.

R-Code 11.3: Self-similarity of the Brownian motion. (See Figure 11.3.)

```
len <- 2^9           # Length of individual paths
for (i in 8:0) {
   plot( Bt[(0:len)*2^i+1, ], type='l', xlab='', ylab='', xaxt='n', yaxt='n',
        xlim=range(Bt[(0:len)*2^i+1,]), ylim=range(Bt[(0:len)*2^i+1,]))
   iN <- c( 1, (len)*2^i+1, (len)*2^(i-1)+1)
   points( Bt[iN,1], Bt[iN, 2], col=2:4, pch=19)
}
```

11.4 Excursions of Brownian Motion

We consider excursions in the literal and figurative meaning.

11.4.1 Random Variables Linked to Brownian Motions

Starting from a Brownian motion, there are several interesting resulting random variables that can be defined. As illustration we consider the first passage time and the maximum of the Brownian motion.

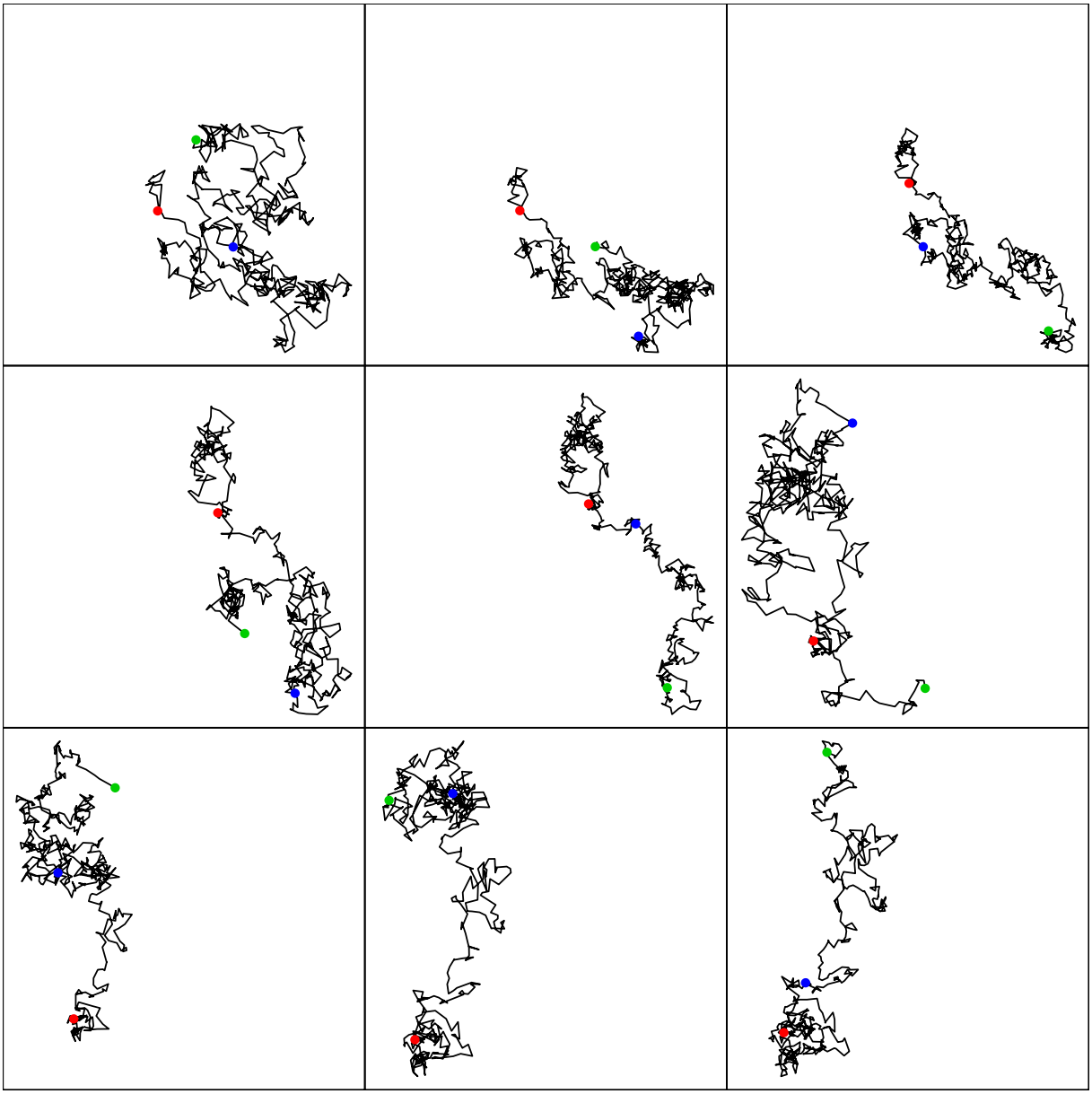

Figure 11.3: Self-similarity of a two-dimensional Brownian motion. The sample path of Figure 11.2 have been subsampled. Each panel consists of $2^9 = 512$ segments (red to green dot) and each following is doubling the resolution (from red to blue dot). The panels have different scales but respect the aspect ratio. (See R-Code 11.3.)

More specifically, we denote the first passage time as the (random) time point $L(x)$ as the time at which $\{B(t); t \geq 0\}$ reaches the level x for the first time, i.e., $L(x) = \min_t\{B(t) = x\}$. Similarly, we let $M(t)$ be the maximum of the Brownian motion in the interval $[0, t]$, i.e., $M(t) = \max_{0\leq s\leq t} B(s)$.

For both random variables, $L(x)$ and $M(t)$, closed form expressions of the distribution and

thus densities exist:

$$F_{L(x)}(t) = 2\Big(1 - \Phi\Big(\frac{|x|}{\sigma\sqrt{t}}\Big)\Big), \qquad f_{L(x)}(t) = \frac{|x|}{\sqrt{2\pi}\sigma\, t^{3/2}} \exp\Big(-\frac{x^2}{2\sigma^2\, t}\Big), \qquad t \geq 0, \tag{11.6}$$

$$F_{M(t)}(x) = 2\Phi\Big(\frac{x}{\sigma\sqrt{t}}\Big) - 1, \qquad f_{M(t)}(x) = \frac{2}{\sqrt{2\pi\, t}\,\sigma} \exp\Big(-\frac{x^2}{2\sigma^2\, t}\Big), \qquad x \geq 0, \tag{11.7}$$

with $\Phi(\cdot)$ the cdf of a standard normal random variable. (The derivation of the former relies on the *reflection principle*.)

11.4.2 Itô Calculus

Starting from an ordinary differential equation

$$\frac{\mathsf{d}\, y(t)}{\mathsf{d}t} = f(t, y(t)) \qquad \mathsf{d}\, y(t) = f(t, y(t))\, \mathsf{d}t \tag{11.8}$$

we can generalize to

1. random differential equations:
   random coefficients or random initial values
2. stochastic differential equations:
   random inhomogeneous part (and possibly random coefficients)

the former is solved "sample path by sample path" for the latter we need special tools, as illustrated in this section.

Itô calculus, named after Kiyoshi Itô, extends the methods of calculus to stochastic processes such as Brownian motion. It has important applications in mathematical finance and in stochastic differential equations.

The sample paths $b(t)$ of a Brownian motion are continuous functions in t. Hence, integrals of the form

$$u(t) := \int_0^t b(y)\, \mathsf{d}y \tag{11.9}$$

exist. We can consider $u(t)$ as an realization of a stochastic process $U(t)$, defined as

$$U(t) := \int_0^t B(y)\, \mathsf{d}y. \tag{11.10}$$

The process $U(t)$ is called integrated Brownian motion. Of course, the integral in (11.10) is not to be understood in the classical sense, but it can be defined in a "Riemann" sense and relying on mean-square convergence. Similarly, for a process $X(t)$ satisfying some quite weak properties, the integral $\int_0^t X(y)B(y)\, \mathsf{d}y$ makes sense and is often written as $\int_0^t X(y)\, \mathsf{d}B(y)$.

However, as the sample paths are nowhere differentiable, the stochastic process $\{X(t); t \geq 0\}$ defined by

$$X(t) := \frac{d}{dt}B(t) = B'(t) \tag{11.11}$$

cannot be introduced analogously. A definition via an integral is possible though.

11.5 Bibliographic Remarks

Swiss boundary data is freely available at https://shop.swisstopo.admin.ch/en/products/landscape/boundaries3D.

Kim (2017) argues that cross-sections of broccoli and of cauliflower have approximate Hausdorff dimensions 1.78 ± 0.02 and 1.88 ± 0.02 respectively. https://en.wikipedia.org/wiki/List_of_fractals_by_Hausdorff_dimension gives many more examples of better known and less known fractals.

There a lot of very technical literature on Itô calculus. Chapter 7 of Beichelt (2006) is fairly accessible. The chapter gives a few links to economics but does not cover stochastic differential and partial differential equations. Inference for stochastic differential equations is given in Guttorp (1995), Sections 6.6 and 6.7.

Lab Content

1. Using the Hurst exponent to link the range of a Brownian motion to the average run length and its variance. See https://www.sciencedirect.com/science/article/pii/096007799600032X?via%3Dihub.

Worksheet 11

30. The definition of a Browninan motion and a Gaussian process.

31. Simulation of a Brownian motion and a Brownian bridge.

32. Stirling approximation.

Chapter 12

Markov Random Fields

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta111/chapter12.R.

In this chapter we extend the Markovian idea from a temporal to a spatial setting. We will focus on Gaussian processes mainly, although very popular non-Gaussian settings exist. For example, the Ising model is a mathematical model of ferromagnetism in statistical mechanics. As the magnetic dipole moments of atomic spins take two states (+1 or 1) the model consists of a discrete random variable, typically arranged on a lattice, where the variables interact with its neighbors.

12.1 Sudden Infant Death Syndrome Data

Sudden infant death syndrome (SIDS) is the sudden unexplained death of a child less than one year of age. The death typically occurs during sleep, without any signs of struggle.

We work with well studied dataset containing birth counts as well as SIDS cases for two periods 1974–1978 and 1979–1984, aggregated over two ethnicities and the 100 counties of North Carolina (NC). The dataset has been analyzed quite extensively in the literature, most prominently in Cressie (1993), further details to references are given in the vignette of the package *spdep*.

We investigate the spatial dependency using a Gaussian Markov random fields approach, i.e., the mean and the variance of an individual county depends conditionally on its neighbors. R-Code 12.1 fits such a model using a first order neighbor structure. As seen in Figure 12.1, the data contains a possible outlier that we eliminate for the subsequent modeling.

We use the function *spautolm* fitting to fit a spatial conditional autoregressive model. The resulting fit is a spatially smoothed surface. (The degree of spatial dependency is given by the coefficient named *lambda*.) As the data are quite noisy the range of the residuals seem substantial.

R-Code 12.1: SIDS rates visualization and analysis for the period 1974 to 1978. (See Figure 12.1.)

```
library(spdep)
library(maptools)
library(spData)
path <- system.file( "shapes/sids.shp", package="spData")[1]
nc <- rgdal::readOGR( path, verbose=FALSE)
names(nc)
##  [1] "CNTY_ID"   "AREA"      "PERIMETER" "CNTY_"     "NAME"
##  [6] "FIPS"      "FIPSNO"    "CRESS_ID"  "BIR74"     "SID74"
## [11] "NWBIR74"   "BIR79"     "SID79"     "NWBIR79"   "east"
## [16] "north"     "x"         "y"         "lon"       "lat"
## [21] "L_id"      "M_id"
nc$rates <- nc$SID74 / nc$BIR74
spplot( nc, "rates", col.regions=tim.colors(16))
index <- which.max( nc$rates)            # where is the outlier?
as.character( nc$NAME[index])
## [1] "Anson"
nc$ratesNO <- nc$rates
nc$ratesNO[index] <- nc$rates[index]/10  # "remove" the outlier

ncW <- nb2listw( poly2nb(nc), style = 'B')
# display( as.spam.listw( ncW), cex=1)
# summary( ncW, zero.policy=TRUE)

plot( nc)                                # plot neighborhood structre
plot(ncW, coordinates(nc), add=TRUE, col='green', points=FALSE)
carfit <- spautolm( ratesNO ~ 1, data=nc, listw=ncW, family="CAR")
# summary(carfit)

nc$fit <- fitted(carfit)              # For nice plotting, add to data.frame
nc$resid <- resid( carfit)

c(coef(carfit), s2=carfit$fit$s2, range.resid=range( nc$resid))
##  (Intercept)       lambda           s2 range.resid1 range.resid2
##    1.846e-03    1.506e-01    1.477e-06   -2.298e-03    3.326e-03
spplot( nc, c("ratesNO", "fit"),  col.regions=tim.colors(16))
spplot( nc, "resid", col.regions=tim.colors(16))
```

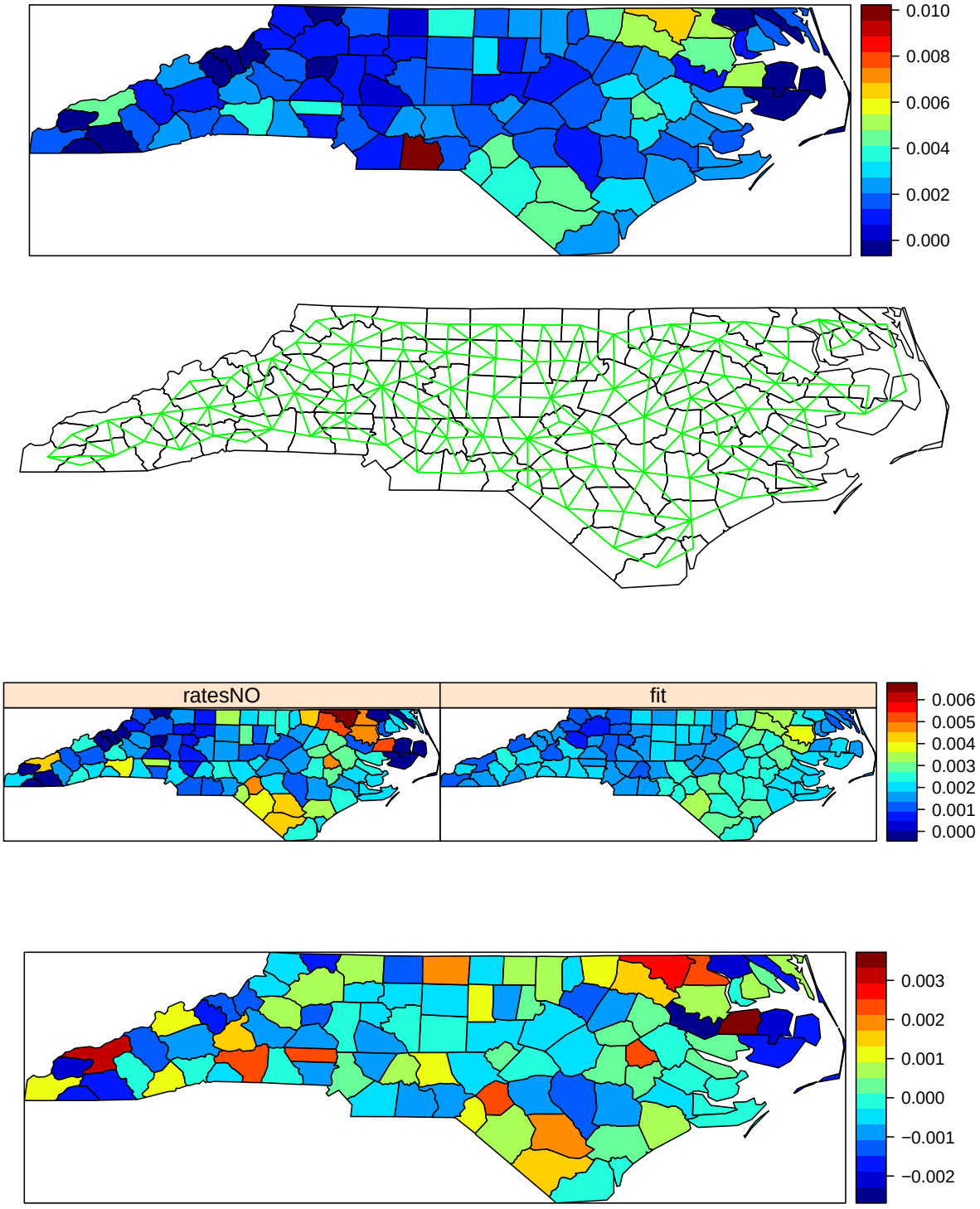

Figure 12.1: Top to bottom: Visualization of crude SIDS rates for the period 1974 to 1978. Counties of North Carolina with first order neighborhood structure of the model (green line indicates neighbors). SIDS rates without outlier and model fit and finally residuals. (See R-Code 12.1.)

12.2 Models for Lattice Data

In several previous chapters we had a Markovian property based on (discrete or continuous) time. We now extend the concept to space (which is conceptually continuous), but use natural discretization thereof. Such spatial data is typically called lattice data or areal data.

An example of such a discretiziation is the partition of the lower 48 US states into the counties (shown in Figure 12.2). The set of a representative location of the block defines a lattice. This specific lattice has a varying density through the differences in the county sizes. Hence a pure geographic distance between the counties does not necessary make sense.

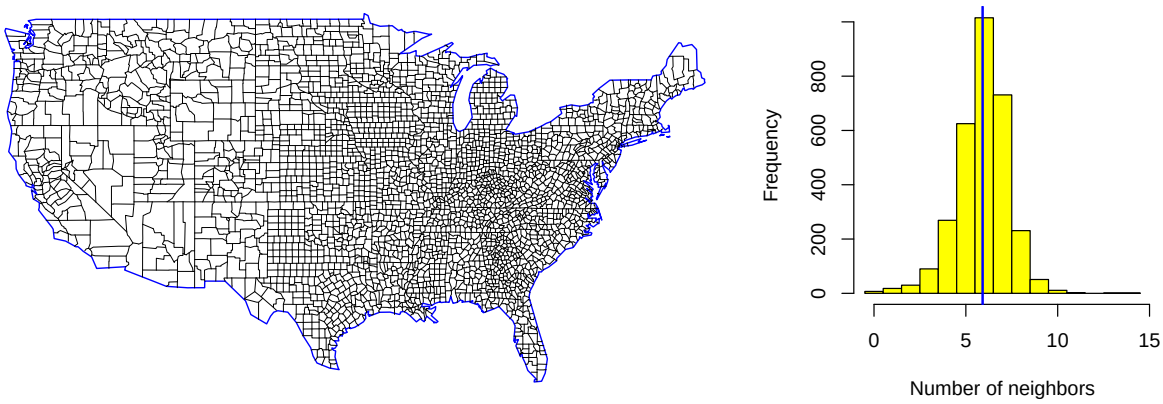

Figure 12.2: The 3082 counties of the lower 48 US states (left panel) and the histograms of the number of neighbors (right). The vertical line represents the average of 5.9. Seven counties do not share any boundaries with others.

There is a significant difference between time series data and lattice data. In time series we have a temporal "direction" and forecasting (prediction) is a very natural concept. In the case of lattice data, forecasting in the sense of interpolation or extrapolation is only used in very rare cases. Task are much more centered around the concept of "smoothing" (i.e., filtering in the time series context). Smoothing can be used to separate a/the signal from the noise or to fill in missing values.

12.2.1 Conditionally Autoregressive Models

Consider a random vector $\mathbf{Y}$ and its density $f_{\mathbf{Y}}(\boldsymbol{y})$. The latter determines the full conditionals $\big\{f(y_i \mid y_j, j \neq i), i = 1, \ldots, n\big\}$. Now, for simplicity, take a regular grid as shown in the left panel of Figure 12.3. The basic idea behind conditionally autoregressive (CAR) models is to construct conditional densities based on the neighbors, i.e., $\big\{f(y_i \mid y_j, j \neq i), i = 1, \ldots, n\big\}$. Naturally, these conditional densities cannot be arbitrary as the joint density $f_{\mathbf{Y}}(\boldsymbol{y})$ might not exist.

As often, assuming Gaussian full conditionals we are likely to get a valid joint density. More specifically, a simple model for location i is given by

$$Y_i \mid \boldsymbol{y}_j, j \neq i \sim \mathcal{N}\Big(\sum_{j, j \neq i} b_{ij} y_j, \tau_i^2\Big), \qquad i = 1, \ldots, n. \tag{12.1}$$

There are some additional conditions (mainly on $\{b_{ij}\}$ and $\{\tau_i^2\}$) but *Brook's lemma* allows us to conclude that the joint distribution of $\mathbf{Y}$ is also Gaussian

$$\mathbf{Y} \sim \mathcal{N}_n(\mathbf{0}, (\mathbf{I} - \mathbf{B})^{-1}\mathbf{T}), \tag{12.2}$$

where $\mathbf{B} = (b_{ij})$ and $b_{ii} = 0$, $\mathbf{T} = \text{diag}(\tau_i^2)$.

Through the joint distribution, we can directly derive a few conditions on the model. Because a covariance matrix is symmetric, it is required that

$$\frac{b_{ij}}{\tau_i^2} = \frac{b_{ji}}{\tau_j^2} \tag{12.3}$$

Further, the matrix $\mathbf{I} - \mathbf{B}$ has to be positive definite. As typically done, a low-dimensional parameterization will be used (see Section 12.4).

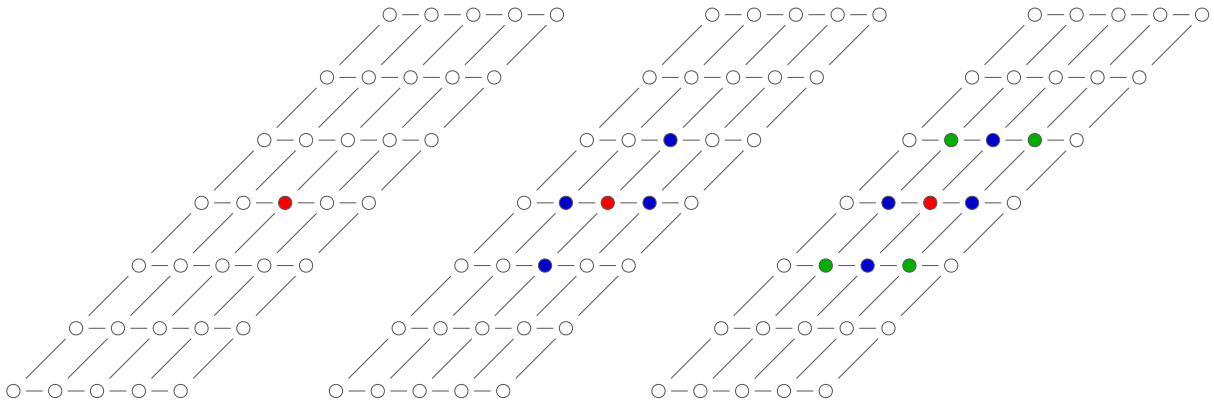

Figure 12.3: Grid locations (middle panel: von Neumann neighborhood, right panel: Moore neighborhood).

The CAR model approach, specifying the joint density through n full conditional densities $\{f(y_i \mid \boldsymbol{y}_{-i})\}$ has been pioneered by Besag (1974). Note here the use of "negative" indices. More specifically, let $\boldsymbol{x}$ be an arbitrary n-vector. Then $\boldsymbol{x}_{-i}$ is the vector $(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)^\top$, that means the vector $(n-1)$-vector obtained by deleting the ith component of $\boldsymbol{x}$.

We impose now a Markov property on the conditional dependence structure and assume that we have conditional dependence on a few neighbor sites only. In a temporal setting, *neighbor* is interpreted as the previous value (and possibly next value); in lattice models, *neighbor* typically signifies sharing a common edge or boundary. We denote the neighbor relation with $i \sim j$ for sites $i \neq j$. The relation is symmetric, i.e., if $i \sim j$ then $j \sim i$. In the middle panel of Figure 12.3 the blue locations are *first order* neighbors of the red location. In terms of (12.10), only four of the $\{b_{ij}\}$ are non-zero. In the right panel of Figure 12.3 the four green *second order* neighbors of the red location have been added, implying four more non-zero $\{b_{ij}\}$s.

12.2.2 Graphs and Gaussian Markov Random Fields

It is convenient to represent the dependence structure of conditionally autoregressive models with an undirected, labeled graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes in a graph and $\mathcal{E}$ the set of edges $\{i, j\}$, $i \neq j \in \mathcal{V}$. As example, Figure 12.4 shows the 5 counties of the US state Rhodes Island ($\mathcal{V} = \{1, 2, 3, 4, 5\}$) and the first-order neighbor structure ($\mathcal{E} = \big\{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{2,5\},\{3,5\}\big\}$) and the associated *adjacency matrix* $\mathbf{A}$, with $a_{ij} = a_{ji} = 1$ if $\{i, j\} \in \mathcal{E}$, $\forall i \neq j \in \mathcal{V}$ and $a_{ij} = a_{ji} = 0$ otherwise.

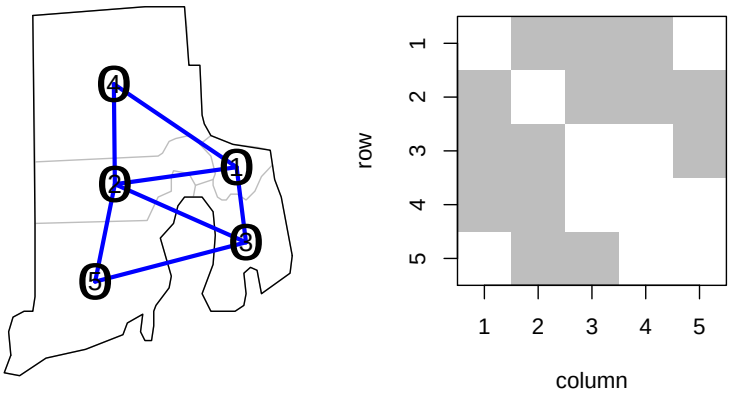

Figure 12.4: The 5 counties of Rhodes Island (left panel) and the associated adjacency matrix with non-zero values represented in gray (right panel).

Definition 12.1. The precision of a random variable is the inverse of the variance.

In the case of random vectors, the inverse of a covariance function is called the precision matrix. ♢

The precision and the diagonal precision matrix is an intuitive concept: a high precision (low variance) implies a lot of knowledge about the variable.

Definition 12.2. A random vector $\mathbf{Y} = (Y_1, \ldots, Y_n)^\top$ is a GMRF with respect to a labelled graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with mean $\boldsymbol{\mu}$ and (symmetric positive definite) precision matrix $\mathbf{Q}$ if its density is given by

$$f(\boldsymbol{y}) = (2\pi)^{-n/2} \det(\mathbf{Q})^{1/2} \exp\Big(-\frac{1}{2}(\boldsymbol{y} - \boldsymbol{\mu})^\top \mathbf{Q} (\boldsymbol{y} - \boldsymbol{\mu}) \Big) \tag{12.4}$$

and $Q_{ij} \neq 0 \iff \{i, j\} \in \mathcal{E}$, $\forall i \neq j$. ♢

We will denote independence between two random variables X and Y by $X \perp Y$ and conditional independence between X and Y given $Z = z$ by $X \perp Y \mid Z = z$.

Property 12.1. *Let* $\mathbf{Y}$ *be a GMRF with respect to* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *with mean* $\boldsymbol{\mu}$ *and (symmetric, positive definite) precision matrix* $\mathbf{Q}$*. Then the following hold.*

1. $Y_i \perp Y_j \mid \mathbf{Y}_{-ij} = y_{-ij} \iff Q_{ij} = 0$.

2. $\mathrm{E}(Y_i \mid \mathbf{Y}_{-i} = \boldsymbol{y}_{-i}) = \mu_i - \dfrac{1}{Q_{ii}} \sum_{j:j\sim i} Q_{ij}(y_j - \mu_j)$,

3. $\mathrm{Prec}(Y_i \mid \mathbf{Y}_{-i} = \boldsymbol{y}_{-i}) = Q_{ii}$,

4. $\mathrm{Corr}(Y_i, Y_j \mid \mathbf{Y}_{-ij} = \boldsymbol{y}_{-ij}) = -\dfrac{Q_{ij}}{\sqrt{Q_{ii}Q_{jj}}}$, $i \neq j$.

12.3 Random Walk Models: Intrinsic GMRFs

A density that does not integrate to one is called an improper density. Such densities are typically used as priors in a Bayesian framework because the resulting posterior may be proper.

In many modeling approaches, we assume that a series consisting of $Y_1, \ldots, Y_n$ has constant mean μ or a mean that is parameterized (in a regression setting). This is often very restrictive and it is natural to relax this condition by considering that $Y_{i+1} - Y_i \sim \mathcal{N}(0, \kappa^{-1})$ instead of $Y_{i+1} - Y_i \equiv 0$.

More specifically, assume that the locations of n random variables are $i = 1, \ldots, n$, e.g., referred to as equispaced observations on the transect. In such a case we also often consider i as the time. We define $\Delta Y_i = Y_{i+1} - Y_i$, $i = 1, \ldots, n-1$. Assuming that the $\Delta Y_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \kappa^{-1})$, we have

$$f(\Delta \boldsymbol{y}) \propto \kappa^{(n-1)/2} \exp\Big(-\frac{\kappa}{2} \sum_{i=1}^{n-1} (\Delta y_i)^2 \Big) = \kappa^{(n-1)/2} \exp\Big(-\frac{1}{2} \boldsymbol{y}^\top \mathbf{Q} \boldsymbol{y} \Big), \tag{12.5}$$

where the $n \times n$ precision matrix is given by

$$\mathbf{Q} = \kappa \begin{pmatrix} 1 & -1 & 0 & \ldots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \ldots & 0 & -1 & 1 \end{pmatrix} = \kappa \mathbf{D}^\top \mathbf{D}, \quad \mathbf{D} = \begin{pmatrix} -1 & 1 & 0 & \ldots \\ 0 & \ddots & \ddots & \ddots \\ \vdots & \ddots & -1 & 1 \end{pmatrix} \in \mathbb{R}^{n-1 \times n}. \tag{12.6}$$

Note that (12.5) is not a proper density because $\mathbf{Q}\mathbf{1} = \mathbf{0}$. In other words, the $\mathbf{Q}$ has rank $n-1$, thus, is rank deficient (here symmetric positive-semidefinite) and does not fit in our "classical" framework for multivariate normal variables.

Definition 12.3. Let $\mathbf{Q}$ be an $n \times n$ symmetric positive-semidefinite matrix of rank $n - k > 0$. $\mathbf{Y}$ is an improper GMRF of rank $n-k$ and parameters $(\boldsymbol{\mu}, \mathbf{Q})$ with respect to the labelled graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ if its density is

$$f^*(\boldsymbol{y}) = (2\pi)^{-(n-k)/2} \det{}^*(\mathbf{Q})^{1/2} \exp\Big(-\frac{1}{2} (\boldsymbol{y} - \boldsymbol{\mu})^\top \mathbf{Q} (\boldsymbol{y} - \boldsymbol{\mu}) \Big) \tag{12.7}$$

and $Q_{ij} \neq 0 \iff \{i, j\} \in \mathcal{E}$, $\forall i \neq j$. The term $\det^*(\mathbf{Q})$ is the generalized determinant, i.e., the product of all the nonzero eigenvalues of $\mathbf{Q}$. ♢

The parameters $(\boldsymbol{\mu}, \mathbf{Q})$ do no longer represent the mean and the precision since, formally, they no longer exist.

Definition 12.4. An intrinsic GMRF (IGMRF) of first-order (or order $n-1$) is an improper GMRF of rank $n-1$ where $\mathbf{Q1} = \mathbf{0}$. ♢

The model given by (12.5) is thus an IGMRF of first-order, also referred to as *Random Walk* of first-order, RW1.

Prediction in a RW1 model is a meaningful and we have, for example, the following results:

$$Y_i \mid \mathbf{Y}_{-i} = \boldsymbol{y}_{-i} \sim \mathcal{N}\Big(\frac{1}{2}(y_{i+1} + y_{i-1}), \frac{1}{2\kappa}\Big), \tag{12.8}$$

$$Y_{i+p} \mid Y_i = y_i, Y_{i-1} = y_{i-1}, \ldots \sim \mathcal{N}\Big(y_i, \frac{p}{\kappa}\Big), \qquad 0 < i < i+p \le n. \tag{12.9}$$

Notice that there is no shrinkage towards the mean, as typically seen in prediction.

Similar to a RW1 model, we can define a RW2 by defining $\Delta^2 Y_i = \Delta Y_{i+1} - \Delta Y_i = Y_{i+2} - 2Y_{i+1} + Y_i$, $i = 1, \ldots, n-2$. We assume again that the $\Delta^2 Y_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \kappa^{-1})$.

12.4 Specific models for GMRF

As already discussed several times, low-dimensional parametrizations are needed. For GMRFs, we can choose $\{b_{ij}\}$ and τ_i^2 (under the constraint of symmetry and positive definiteness of the precision matrix).

We often assume that $\tau_i = \tau$, for all i. That means that the conditional precision is constant. Another simplification is that the coefficients b_{ij} do not depend on j, i.e., $b_{ij} = b_i$, no preference is given to "neighboring" information. More specifically, we have

$$Y_i \mid \boldsymbol{y}_{-i} \sim \mathcal{N}\Big(\sum_{j, j\sim i} \theta_1 y_j, \tau^2\Big), \quad \tau > 0, \qquad i = 1, \ldots, n, \tag{12.10}$$

where $j \sim i$ indicates a first order neighbor. The neighbor structure can be encoded in a matrix, often denoted with $\mathbf{A}$ (adjacency matrix) or $\mathbf{W}$ (spatial weight matrix). In both cases, we have a $w_{ij} = 1$ if $i \sim j$ and zero otherwise. To link with Section 12.2.1 and literature elsewhere, we often write $\mathbf{B} = \lambda \mathbf{W}$, for some λ.

For a legitimate precision matrix, we need $\mathbf{Q} = \mathbf{I} - \lambda \mathbf{W}$ to be positive definite. Hence, we can choose $\lambda \in (\lambda_{\max}, \lambda_{\min})$, where $\lambda_{\min}$ is the smallest and $\lambda_{\max}$ the largest eigenvalue of the matrix $\mathbf{W}$.

Example 12.1. R-Code 12.2 illustrates an example based on the oral cavity cancer data. We use a simple CAR model with binary weight matrix. The estimated value of λ is quite close to at the boundary of the valid range. This might be an indication, that the proposed CAR model is not sufficiently flexible. This might also be seen as the drawn realizations seem more speckled (see Figure 12.5). Notice that only the marginal precision is constant, not the standard deviation. ♣

R-Code 12.2: Oral cavity cancer example. (See Figure 12.5.)

```
require( spdep)
require( spam)
data( Oral, package="spam")
hist( Oral$SMR, main="", xlab="SMR")
abline( v=mean( Oral$SMR), col=4, lwd=2)
path <- system.file("demodata/germany.adjacency", package="spam")
W <- adjacency.landkreis(path)
barplot( table( diff( W@rowpointers)), xlab="# of neighbors")
Dlistw <- mat2listw( as.matrix(W))
# summary( Dlistw)

# We can now start spatial modeling. For example a simple CAR
carfit <- spautolm( SMR ~ 1, data=Oral, listw=Dlistw, family="CAR")
coefs <- c( coef( carfit), sigma2=carfit$fit$s2)
print(  c( coefs, logLik=carfit$LL))
## (Intercept)      lambda      sigma2      logLik
##     1.00103     0.15219     0.08888  -141.58270
# Valid parameter range for lambda:
1/range( eigen( as.matrix(W), only.values=TRUE)$values)
## [1] -0.2959  0.1591
# Hence the estimate is quite to the upper boundary!
germany.plot( Oral$SMR, main='SMR',border=NA)
germany.plot( rnorm( 544), main="White noise",border=NA)
##### simulations with similar parameters:
N <- 1000
set.seed(16)
Q <- (diag.spam( 544)- coefs[2]*W)/coefs[3]
ex <- rmvnorm.prec( N, mu=coefs[1], Q=Q)
# Plot and fit for specific examples:
lambda.est <- numeric(4)
for (i in 1:4) {
  germany.plot( ex[i,], main=paste("Sample",i), border=NA)
  lambda.est[i] <- spautolm( ex[i,] ~ 1, listw=Dlistw, family="CAR")$lambda
}
lambda.est
## [1] 0.1581 0.1478 0.1528 0.1528
germany.plot( colMeans( ex), main='Means',border=NA)
germany.plot( apply( ex, 2, sd), main='SD',border=NA)
germany.plot( diag( solve( cov( ex))), main="Precision",border=NA)
```

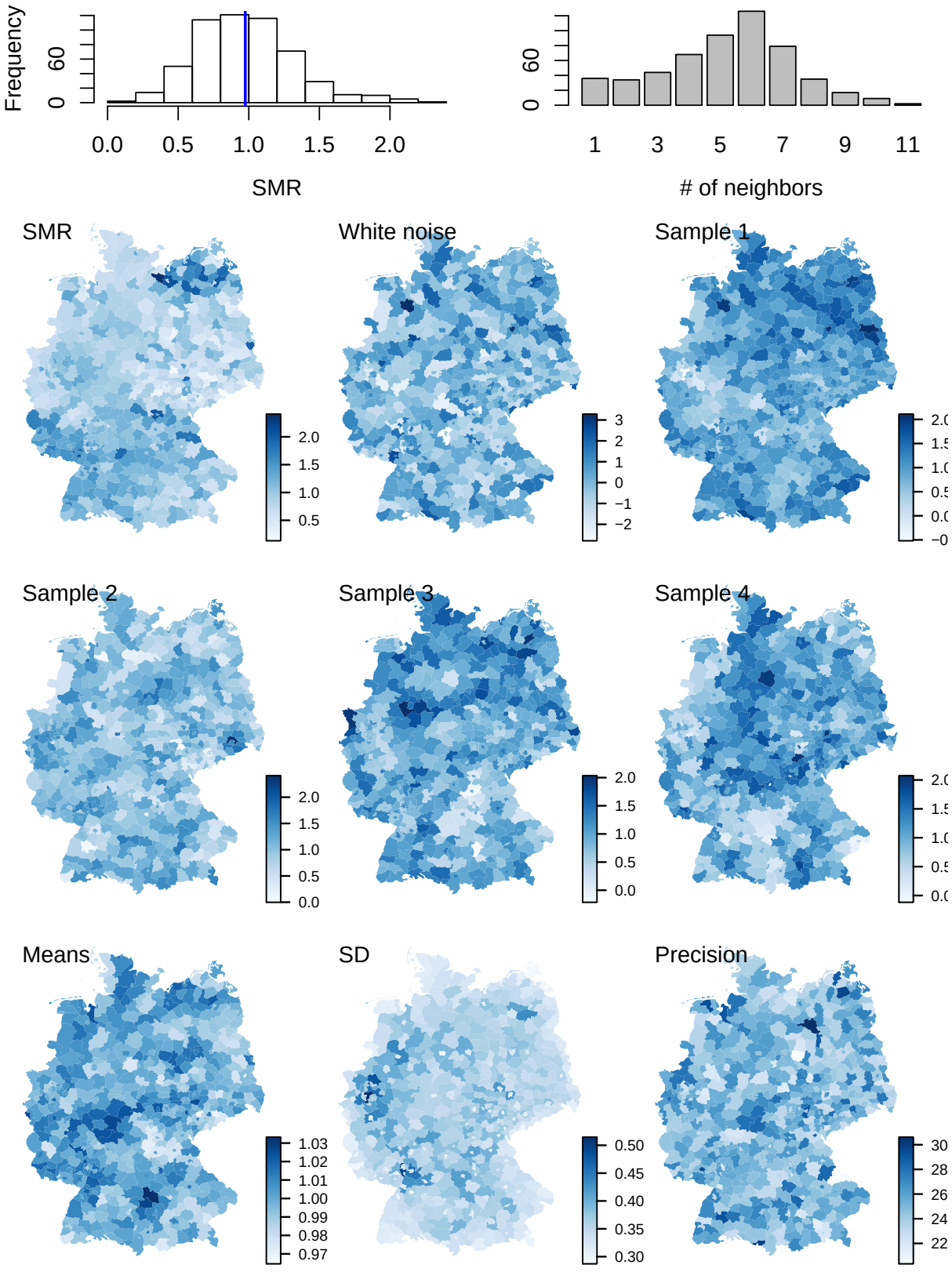

Figure 12.5: Top row: histogram of SMR of oral cavity cancer (left) and number of neighbors of each of the districts (right). Bottom panels: SMR, white noise comparison, four samples having the same mean and precision matrix (unconditional simulation), means, marginal standard deviation and precision of 1000 samples. (See R-Code 12.2.)

In Example 12.2 and corresponding R-Code 12.3 we use a more complex parameterization by letting the coefficients b_{ij} depending on the degree of neighbor: $b_{ij} = b_1$ for adjacent cells (first order neighbors) and $b_{ij} = b_2$ for adjacent cells of adjacent cells (second order neighbors).

Example 12.2. Illustration of numerically determine the valid parameter space. Consider a CAR model with a first- and a second-order neighbor structure. More specifically, we consider

$$Y_i \mid \boldsymbol{y}_{-i} \sim \mathcal{N}\Big(\sum_{j,j\sim i} \theta_1 y_j + \sum_{j,j\approx i} \theta_2 y_j, \tau^2\Big), \quad \tau > 0, \qquad i = 1, \ldots, n, \tag{12.11}$$

where $j \sim i$ and $j \approx i$ indicate a first order neighbor and a second order neighbor, respectively. That means, the resulting precision matrix $\mathbf{Q}$ has the structure $\mathbf{Q} = \tau^{-2}(\theta_1 \mathbf{W}_1 + \theta_2 \mathbf{W}_2)$ where $\mathbf{W}_i$ are (binary) spatial weight matrices. We have to impose constraints on (θ_1, θ_2), such that $\mathbf{Q}$ is positive definite, i.e., we need to determine the valid parameter space $\boldsymbol{\Theta} \subset \mathbb{R}^2$.

We first construct an (arbitrary, but valid) precision matrix based on the first- and second-order structure to exploit the `spam` options. Then, we cycle over a specified fine grid of `theta1` and `theta2` and we verify if the precision matrix is positive definite. If the matrix passed to `update` is not symmetric positive definite, which means that value of `tmp` is `NULL`, the pair (`theta1[i]`,`theta2[2]`) lies not within $\boldsymbol{\Theta}$. Figure 12.6 shows $\boldsymbol{\Theta}$. The valid range is color coded according to the value of $\log(\det \mathbf{Q})$. Notice the asymmetry of the domain and the very small values of the determinant.

Based on the fine grid, the loop takes several minutes. Naturally, the convexity of the domain could easily be exploited. ♣

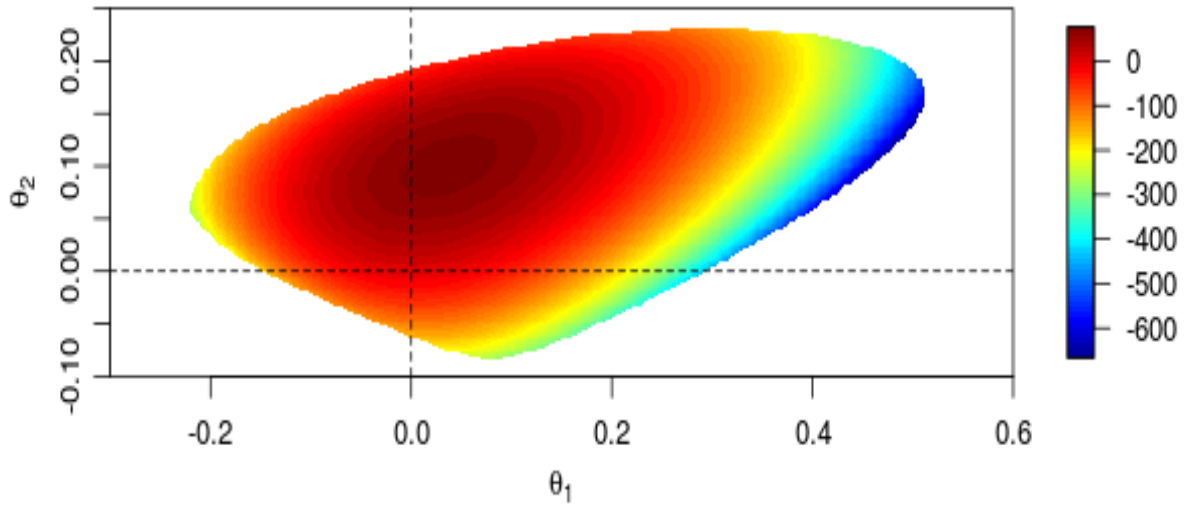

Figure 12.6: The domain $\boldsymbol{\Theta}$ for the US counties with a second-order neighbor structure. The values represent $\log(\det \mathbf{Q})$ and the white area represents the complement of $\boldsymbol{\Theta}$. (See R-Code 12.3.)

R-Code 12.3 Determining the valid parameter space Θ through a numerical assessment of the precision matrix. (See Figure 12.6.)

```
require( spam)
In <- diag.spam( nrow( UScounties.storder))
struct <- chol( In + .2 * UScounties.storder + .1 * UScounties.ndorder)
#  which is a valid, but (arbitrary) precision matrix.
options('spam.cholupdatesingular'='null')      # We want to avoid errors.
len1 <- 300   # even
len2 <- 150
theta1 <- seq( -.225, .515, len=len1)
theta2 <- seq( -.085, .235, len=len2)
grid <- array( NA, c( len1, len2))
for (i in 1:len1) {
    for(j in 1:len2) {
      tmp <- update( struct, In + theta1[i]*UScounties.storder
                          + theta2[j]* UScounties.ndorder)
      if(!is.null(tmp))   grid[i,j] <- determinant(tmp)$modulus
    }
}
image.plot( theta1, theta2, grid, xlab=expression(theta[1]),
            ylab=expression(theta[2]), xlim=c(-.3,.6), ylim=c(-.1,.25))
abline( v=0, h=0, lty=2)
```

12.5 To Go Beyond

In this section we give some general remarks concerning computational and statistical aspects linked to GMRFs.

1. In the case of GMRF, the off-diagonal non-zero elements of the precision matrix $\mathbf{Q}$ correspond to the conditional dependence structure. As by the Markovian property, the number of neighbors is small, the precision matrix $\mathbf{Q}$ is *sparse*, i.e., contains only $\mathcal{O}(n)$ non-zero elements compared to $\mathcal{O}(n^2)$ for a regular, *full* matrix. To take advantage of the few non-zero elements, special structures to represent the matrix are required, i.e., only the positions of the non-zeros and their values are kept in memory. Because of these special structures, tailored algorithms are required to fully exploit the sparsity structure. The package `spam` provides this functionality, see Furrer and Sain (2009) for a detailed exposition.

   It is important to note that the labeling (order of the variables) has an influence on the structures that result from relevant computations. Reordering is performed through a so-called permutation. In matrix notation, a permutation is a matrix having exactly one element 1 per row and column. All other elements are zero. Hence, choosing a proper permutation is crucial for a modeling of huge datasets.

2. In the case of (arbitrary) lattice data, *Moran's I* and *Geary's C* are often used metrics to

assess spatial dependenies. Let $\mathbf{Y}$ be a multivariate random n-vector and $\mathbf{W} = (w_{ij})$ a matrix of spatial weights. Moran's I is defined as

$$I = \frac{n}{\sum_{i,j} w_{ij}} \frac{\sum_{i,j} w_{ij}(Y_i - \overline{Y})(Y_j - \overline{Y})}{\sum_i (Y_i - \overline{Y})^2} \quad \in [-1, 1]. \tag{12.12}$$

Positive (negative) values of the observed statistic indicate positive (negative) spatial autocorrelation Note that under no spatial dependency $\mathrm{E}(I) = -1/(n-1)$. Closed form expressions for the variance exist.

Geary's C is defined as

$$C = \frac{n-1}{2\sum_{i,j} w_{ij}} \frac{\sum_{i,j} w_{ij}(Y_i - Y_j)^2}{\sum_i (Y_i - \overline{Y})^2} \quad \in [0, 2] \tag{12.13}$$

Smaller values indicate stronger positive spatial dependency. Moran's I and Geary's C are measures of global spatial autocorrelation. However, Geary's C is more sensitive to local spatial autocorrelation. The local spatial autocorrelation can further be exploited with functions `moran.plot` and `localmoran` (Bivand *et al.*, 2013, Section 9.3.2).

3. As an alternative to a CAR specification, an autoregressive approach that is closer to the time series one is as follows:

$$Y_i = \sum_{j, j \neq i} b_{ij} Y_j + \varepsilon_i, \qquad \varepsilon_i \sim \mathcal{N}(0, \sigma_i^2), \qquad i = 1, \ldots, n. \tag{12.14}$$

To see the link to the time series, suppose a univariate grid and set $b_{i,i-1} = \phi$ and zero for all others.

Model (12.14) is called a simultaneous autoregressive (SAR) model. The condition that the sum is over $i \neq j$ implies that $b_{ii} \equiv 0$ for all i. Further conditions on $\{b_{ij}\}$ exist and for the moment we assume that $\mathbf{I} - \mathbf{B}$ is invertible. Then

$$\mathbf{Y} \sim \mathcal{N}\big(\mathbf{0}, (\mathbf{I} - \mathbf{B})^{-1} \mathbf{V} ((\mathbf{I} - \mathbf{B})^{-1})^\top\big), \tag{12.15}$$

where $\mathbf{V} = \mathrm{diag}(\sigma_i^2)$.

There is a link between CAR and SAR models, visible when we write both models as

$$\mathbf{Y} = \mathbf{B}\mathbf{Y} + \boldsymbol{\varepsilon}. \tag{12.16}$$

The models as well as the (dis)similarities are summarized in Table 12.1. However, every SAR model can be written as a CAR but not vice-versa.

4. In its simplest form, the Ising Model consists of a $N \times N$ lattice of binary variables $x_i \in \{-1, +1\}$ that are locally connected horizontally and vertically with pairwise potentials. There can also be an external field applied to the variables that biases them toward a particular state. The total energy of a simple Ising model we consider here is defined as

$$E = -J \sum_{(i,j) \in E} x_i x_j - J_b \sum_{i \in V} b_i x_i, \tag{12.17}$$

where the first sum is over all edges of the lattice and the second over all nodes. Here, J, J_b, b_i are the strength of pairwise interactions, strength of external field, and per-pixel binary desired values. The corresponding un-normalized probability distribution over states of the lattice is

$$f(\boldsymbol{x}) \propto \exp\Big(J \sum_{(i,j)\in E} x_i x_j + J_b \sum_{i\in V} b_i x_i\Big). \tag{12.18}$$

When J is positive, the low energy states are smooth regions, as this minimizes the number of edges that connect nodes of different values. When J is negative, the reverse happens as the model assigns higher probabilities to states with many crossing edges.

As we do not have closed form of the density (for reasonable lattice sizes) a Gibbs sampling procedure is used to sample from an Ising model. In a nutshell, for a given sample $\boldsymbol{x}$ we propose a new sample $\boldsymbol{x}'$ where we flipped a single component, say, $x_i' = -x_i$. The acceptance probability is given by

$$\alpha(\boldsymbol{x}'|\boldsymbol{x}) = \min(1, \frac{f(\boldsymbol{x}')}{f(\boldsymbol{x})}). \tag{12.19}$$

Specifically, in the case of $f(\boldsymbol{x}') > f(\boldsymbol{x})$, the state will transition to $\boldsymbol{x}'$ (i.e., the proposal is definitely accepted). In the case of $f(\boldsymbol{x}') \leq f(\boldsymbol{x})$, the proposal is only accepted with probability $\alpha(\boldsymbol{x}'|\boldsymbol{x})$.

The Ising Model is a special case of a pairwise Markov Random Field, which is a special case of a Markov Random Field.

Table 12.1: (Dis)similarities between a CAR and a SAR model.

CAR	SAR
$\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{T}(\mathbf{I}-\mathbf{B})^\top)$	$\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{V})$
$\mathbf{Y} \sim \mathcal{N}(\mathbf{0}, (\mathbf{I}-\mathbf{B})^{-1}\mathbf{T})$	$\mathbf{Y} \sim \mathcal{N}\big(\mathbf{0}, (\mathbf{I}-\mathbf{B})^{-1}\mathbf{V}((\mathbf{I}-\mathbf{B})^{-1})^\top\big)$
$\mathrm{Cov}(\mathbf{Y}, \boldsymbol{\varepsilon}) = \mathbf{T}$	$\mathrm{Cov}(\mathbf{Y}, \boldsymbol{\varepsilon}) = (\mathbf{I}-\mathbf{B})^{-1}\mathbf{V}$

12.6 Bibliographic Remarks

Virtually all books about spatial data have at least one chapter dedicated to lattice data or areal data. We typically recommend Cressie (1993); Schabenberger and Gotway (2005); Banerjee *et al.* (2003). Handling of the data with appropriate packages is extensively discussed in Bivand *et al.* (2008).

The CAR models approach was pioneered by Besag (1974). However, it took a couple decades until its full power was recognized and through the nowadays computational capabilities exploited.

The page https://cs.stanford.edu/people/karpathy/visml/ising_example.html provides an interactive Ising simulation. Further details are given https://en.wikipedia.org/wiki/Square-lattice_Ising_model.

The vignette *vignette('nb', package='spdep')* with direct link https://cran.r-project.org/web/packages/spdep/vignettes/nb.pdf gives further insights in building neighborhood structures, see also *vignette('sids', package='spdep')*.

Lab Content

- Formal definition of a GMRF using graph theory.
- Positivity Condition G6,7
- Brooks lemma
- CAR model (Thm 4.1, G206)

Worksheet 12

33. GMRF on line and circle.
34. Simulate GMRFs with spam

Bibliography

Alphey, L., Benedict, M., Bellini, R., Clark, G. G., Dame, D. A., Service, M. W., and Dobson, S. L. (2010). Sterile-insect methods for control of mosquito-borne diseases: An analysis. *Vector-Borne and Zoonotic Diseases*, **10**, 295–311. 50

Altmann, S. A. and Altmann, J. (1970). *Baboon ecology. African field research*. S. Karger Basel, Munchen, New York. 50

Amherd, E., Furrer, R., Oswald, O., and Zeiter, D. (1992). Graphtal plants and varieties of trees. Working Report. 77, 88

Banerjee, S., Carlin, B. P., and Gelfand, A. E. (2003). *Hierarchical Modeling and Analysis for Spatial Data*. Chapman & Hall/CRC, London. 112

Beichelt, F. (2006). *Stochastic Processes in Science, Engineering and Finance*. CRC Press. vii, 40, 59, 97

Besag, J. (1974). Spatial Interaction and the Statistical Analysis of Lattice Systems (with discussion). *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **36**, 192–225. 103, 112

Biham, O., Middleton, A. A., and Levine, D. (1992). Self-organization and a dynamical transition in traffic-flow models. *Physical Review A*, **46**, R6124–R6127. 77

Bivand, R. S., Pebesma, E. J., and Gómez-Rubio, V. (2008). *Applied Spatial Data Analysis with R (Use R)*. Springer Verlag. 112

Bivand, R. S., Pebesma, E. J., and Gómez-Rubio, V. (2013). *Applied Spatial Data Analysis with R (Use R)*. Springer Verlag, second edition. 111

Cohen, J. E. (1969). Natural primate troops and a stochastic population model. *The American Naturalist*, **103**, 455–477. 50

Cressie, N. A. C. (1993). *Statistics for Spatial Data*. Wiley, revised edition. 99, 112

Embrechts, P., Klüppelberg, C., and Mikosch, T. (1997). *Modelling Extremal Events for Insurance and Finance*. Springer. 15

Enkerlin, W., editor (2007). *Guidance for Packing, Shipping, Holding and Release of Sterile Flies in Area-Wide Fruit Fly Control Programmes*. Joint FAO/IAEA Programme of Nuclear Techniques in Food and Agriculture. 50

Furrer, R. and Sain, S. R. (2009). Spatial Model Fitting for Large Datasets with Applications to Climate and Microarray Problems. *Statistics and Computing*, **19**, 113–128. 110

Grimmett, G. and Stirzaker, D. (2001). *Probability and random processes*. Oxford university press. 15

Guttorp, P. (1995). *Stochastic Modeling of Scientific Data*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis. vii, 17, 50, 58, 59, 97

Karlin, S. and Taylor, H. M. (1981). *A Second Course in Stochastic Processes*. Academic Press. 38, 50, 59

Keiding, N. (1977). Statistical comments on cohen's application of a simple stochastic population model to natural primate troops. *The American Naturalist*, **111**, 1211–1219. 50

Kemeny, J. G., Snell, J. L.,*et al.* (1960). *Finite markov chains*. University series in undergraduate mathematics. Van Nostrand. 15, 31

Kim, S.-H. (2017). Fractal dimensions of a green broccoli and a white cauliflower. "arXiv preprint http://arxiv.org/abs/cond-mat/0411597v1". 97

Krebs, C. J., Boonstra, R., and Boutin, S. (2018). Using experimentation to understand the 10year snowshoe hare cycle in the boreal forest of North America. *Journal of Animal Ecology*, **87**, 87–100. 69

Leemis, L. M. and McQueston, J. T. (2008). Univariate distribution relationships. *The American Statistician*, **62**, 45–53. 3, 4

Moran, P. A. P. (1949). The statistical analysis of the sunspot and lynx cycles. *Journal of Animal Ecology*, **18**, 115–116. 69

Mueller, E. R. and Chatterji, G. B. (2002). Analysis of aircraft arrival and departure delay characteristics. In *http://web.mit.edu/~feron/Public/www/aiaa99.pdf*. 40

Pinsky, M. and Karlin, S. (2011). *An Introduction to Stochastic Modeling*. Academic Press, fourth edition. vii

Ross, S. M. (2010). *A First Course in Probability*. Pearson Prentice Hall. 6, 9, 15

Sankaranarayanan, H. B., Agarwal, G., and Rathod, V. (2016). An exploratory data analysis of airport wait times using big data visualisation techniques. In *2016 International Conference on Computational Systems and Information Systems for Sustainable Solutions*, 324–329. IEEE. 40

Schabenberger, O. and Gotway, C. A. (2005). *Statistical Methods for Spatial Data Analysis*. Chapman & Hall/CRC. 112

Schilling, M. F. (1990). The longest run of heads. *The College Mathematics Journal*, **21**, 196–207. 15

Scott, M. J. and Benedict, M. Q. (2016). Concept and history of genetic control. In *Genetic Control of Malaria and Dengue*, 31–54. Elsevier. 50

Sen, Z. (1991). On the probability of the longest run length in an independent series. *Journal of Hydrology*, **125**, 37 – 46. 15

Taylor, H. and Karlin, S. (1998). *An Introduction to Stochastic Modeling*. Academic Press, third edition. vii

Wang, J., Shortle, J. F., Wang, J., and Sherry, L. (2009). Analysis of gate-waiting delays at major us airports. In *9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO)*, 1–20. 40

Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media Inc., Champaign, Ilinois, US, United States. 78, 79