Data Analysis in Ecology with R and Stan

Fränzi Korner-Nievergelt and Tobias Roth

2019-08-05

Table of Contents

[Preface 5](#_Toc15891376)

[Why this book? 5](#_Toc15891377)

[About this book 5](#_Toc15891378)

[How to contribute? 6](#_Toc15891379)

[Acknowledgments 6](#_Toc15891380)

[Introduction to PART I 6](#_Toc15891381)

[Further reading 7](#_Toc15891382)

[Prerequisits: Basic statistical terms 7](#_Toc15891383)

[Scale of measurement 7](#_Toc15891384)

[Correlations 7](#_Toc15891385)

[Basics of variances, covariances and correlations 7](#_Toc15891386)

[Pearson correlation coefficient 7](#_Toc15891387)

[Spearman correlation coefficient 7](#_Toc15891388)

[Kendall’s tau 8](#_Toc15891389)

[Principal components analyses PCA 8](#_Toc15891390)

[Inferential statistics 9](#_Toc15891391)

[Standard deviation and standard error 11](#_Toc15891392)

[Central limit theorem / law of large numbers 12](#_Toc15891393)

[Bayes theorem 12](#_Toc15891394)

[Bayes theorem for continuous parameters 13](#_Toc15891395)

[Single parameter model 13](#_Toc15891396)

[A model with two parameters 14](#_Toc15891397)

[t-distribution 15](#_Toc15891398)

[Frequentist one-sample t-test 15](#_Toc15891399)

[Nullhypothesis test 16](#_Toc15891400)

[Confidence interval 16](#_Toc15891401)

[Posterior distribution 17](#_Toc15891402)

[Posterior probability 17](#_Toc15891403)

[Monte Carlo simulation (parametric bootstrap) 18](#_Toc15891404)

[3 methods for getting the posterior distribution 18](#_Toc15891405)

[Grid approximation 18](#_Toc15891406)

[Monte Carlo simulations 19](#_Toc15891407)

[Comparison of the locations between two groups 19](#_Toc15891408)

[Difference between two means 20](#_Toc15891409)

[Two-sample t-test 20](#_Toc15891410)

[Wilxocon test 20](#_Toc15891411)

[Randomisation test 21](#_Toc15891412)

[Bootstrap 21](#_Toc15891413)

[F-distribution 22](#_Toc15891414)

[Analysis of variance ANOVA 23](#_Toc15891415)

[Chisquare test 26](#_Toc15891416)

[Bayesian way of analysing correlations between categorical variables 28](#_Toc15891417)

[Summary 29](#_Toc15891418)

[Data analysis step by step 29](#_Toc15891419)

[Plausibility of Data 30](#_Toc15891420)

[Relationships 30](#_Toc15891421)

[Data Distribution 30](#_Toc15891422)

[Preparation of Explanatory Variables 31](#_Toc15891423)

[Data Structure 32](#_Toc15891424)

[Define Prior Distributions 32](#_Toc15891425)

[Fit the Model 32](#_Toc15891426)

[Check Model 32](#_Toc15891427)

[Model Uncertainty 33](#_Toc15891428)

[Draw Conclusions 33](#_Toc15891429)

[Further reading 33](#_Toc15891430)

[Data preparation 33](#_Toc15891431)

[Basic operations 33](#_Toc15891432)

[Joining tables 33](#_Toc15891433)

[Further reading 33](#_Toc15891434)

[Probability distributions 34](#_Toc15891435)

[Introduction 34](#_Toc15891436)

[Normal distribution 34](#_Toc15891437)

[Poisson distribution 34](#_Toc15891438)

[Gamma distribution 34](#_Toc15891439)

[Cauchy distribution 34](#_Toc15891440)

[Visualizations 34](#_Toc15891441)

[Short Checklist for figures 34](#_Toc15891442)

[Further reading 35](#_Toc15891443)

[Spatial analyses and maps 36](#_Toc15891444)

[Data types 36](#_Toc15891445)

[Raster data 36](#_Toc15891446)

[Geometry data 37](#_Toc15891447)

[Basic functions 39](#_Toc15891448)

[Coordinate systems 39](#_Toc15891449)

[Joining spatial data 39](#_Toc15891450)

[Further reading 40](#_Toc15891451)

[Reproducible Research 40](#_Toc15891452)

[Introduction 40](#_Toc15891453)

[Further reading 40](#_Toc15891454)

[Further topics 40](#_Toc15891455)

[Bioacoustic analyse 40](#_Toc15891456)

[Python 41](#_Toc15891457)

[Introduction to PART II 41](#_Toc15891458)

[Further reading 41](#_Toc15891459)

[Prior distributions 42](#_Toc15891460)

[Introduction 42](#_Toc15891461)

[How to choose a prior 42](#_Toc15891462)

[Prior sensitivity 42](#_Toc15891463)

[MCMC using Stan 42](#_Toc15891464)

[Background 42](#_Toc15891465)

[Install rstan 42](#_Toc15891466)

[Writing a Stan model 42](#_Toc15891467)

[Run Stan from R 44](#_Toc15891468)

[Further reading 46](#_Toc15891469)

[Ridge regression 46](#_Toc15891470)

[Introduction 46](#_Toc15891471)

[Structural equation model 49](#_Toc15891472)

[Introduction 49](#_Toc15891473)

[Introduction to PART III 51](#_Toc15891474)

[Model notations 52](#_Toc15891475)

[Zero-inflated Poisson Mixed Model 52](#_Toc15891476)

[Introduction 52](#_Toc15891477)

[Example data 52](#_Toc15891478)

[Model 53](#_Toc15891479)

[Daily nest survival 54](#_Toc15891480)

[Background 54](#_Toc15891481)

[Models for estimating daily nest survival 55](#_Toc15891482)

[The Stan model 55](#_Toc15891483)

[Prepare data and run Stan 56](#_Toc15891484)

[Check convergence 59](#_Toc15891485)

[Look at results 59](#_Toc15891486)

[Further reading 60](#_Toc15891487)

[Capture-mark recapture model with a mixture structure to account for missing sex-variable for parts of the individuals 61](#_Toc15891488)

[Introduction 61](#_Toc15891489)

[Data description 61](#_Toc15891490)

[Model description 62](#_Toc15891491)

[The Stan code 63](#_Toc15891492)

[Call Stan from R, check convergence and look at results 68](#_Toc15891493)

[Referenzen 70](#_Toc15891494)

# Preface



## Why this book?

In 2015, we wrote a statistics book for Master/PhD level Bayesian data analyses in ecology (Korner-Nievergelt et al. [2015](#ref-KornerNievergelt2015)). You can order it [here](https://www.elsevier.com/books/bayesian-data-analysis-in-ecology-using-linear-models-with-r-bugs-and-stan/korner-nievergelt/978-0-12-801370-0). People seemed to like it (e.g. (Harju [2016](#ref-Harju2016))). Since then, two parallel processes happen. First, we learn more and we become more confident in what we do, or what we do not, and why we do what we do. Second, several really clever people develop software that broaden the spectrum of ecological models that now easily can be applied by ecologists used to work with R. With this e-book, we open the possibility to add new or substantially revised material. In most of the time, it should be in a state that it can be printed and used together with the book as handout for our stats courses.

## About this book

We do not copy text from the book into the e-book. Therefore, we refer to the book (Korner-Nievergelt et al. [2015](#ref-KornerNievergelt2015)) for reading about the basic theory on doing Bayesian data analyses using linear models. However, Chapters 1 to 17 of this dynamic e-book correspond to the book chapters. In each chapter, we may provide updated R-codes and/or additional material. The following chapters contain completely new material that we think may be useful for ecologists.

While we show the R-code behind most of the analyses, we sometimes choose not to show all the code in the html version of the book. This is particularly the case for some of the illustrations. An intrested reader can always consult the [public GitHub repository](https://github.com/TobiasRoth/BDAEcology) with the rmarkdown-files that were used to generate the book.

## How to contribute?

It is open so that everybody with a [GitHub](https://github.com) account can make comments and suggestions for improvement. Readers can contribute in two ways. One way is to add an [issue](https://github.com/TobiasRoth/BDAEcology/issues). The second way is to contribute content directly through the edit button at the top of the page (i.e. a symbol showing a pencil in a square). That button is linked to the rmarkdown source file of each page. You can correct typos or add new text and then submit a [GitHub pull request](https://help.github.com/articles/about-pull-requests/). We try to respond to you as quickly as possible. We are looking forward to your contribution!

## Acknowledgments

We thank *Yihui Xie* for providing [bookdown](bhttps://bookdown.org/yihui/bookdown/) which makes it much fun to write open books such as ours.

# Introduction to PART I

**Fehler! Linkreferenz ungültig.**

During our courses we are sometimes asked to give an introduction to some R-related stuff covering data analysis, presentation of results or rather specialist topics in ecology. In this part we present collected these introduction and try to keep them updated. This is also a commented collection of R-code that we documented for our own work. We hope this might be useful olso for other readers.

## Further reading

* [R for Data Science by Garrett Grolemund and Hadley Wickham](http://r4ds.had.co.nz): Introduces the tidyverse framwork. It explains how to get data into R, get it into the most useful structure, transform it, visualise it and model it.

# Prerequisits: Basic statistical terms

This chapter introduces some important terms useful for doing data analyses. It also introduces the essentials of the classical frequentist tests such as t- and Chisquare test. We will not use them later but we think it is important to know how to interpret the results in order to be able to understand 100 years of scientific literature. For each classical test, we provide a suggestion how to do it in a Bayesian way and we discuss some differences between the Bayesian and frequentist statistics.

## Scale of measurement

|  |  |  |  |
| --- | --- | --- | --- |
| Scale | Examples | Properties | Coding in R |
| Nominal | Sex, genotype, habitat | Identity (values have a unique meaning) | factor() |
| Ordinal | Elevational zones | Identity and magnitude (values have an ordered relationship) | ordered() |
| Numeric | Discrete: counts; continuous: body weight, wing length | Identity, magnitude, and equal intervals | intgeger() numeric() |

## Correlations

### Basics of variances, covariances and correlations

* variance   
  The term is called the degrees of freedom.
* standard deviation
* covariance

### Pearson correlation coefficient

standardized covariance

### Spearman correlation coefficient

rank correlation  
correlation between rank(x) and rank(y)

robust against outliers

### Kendall’s tau

rank correlation

I = number of pairs (i,k) for which & or viceversa

## Principal components analyses PCA

rotation of the coordinate system (see Fig. 1)

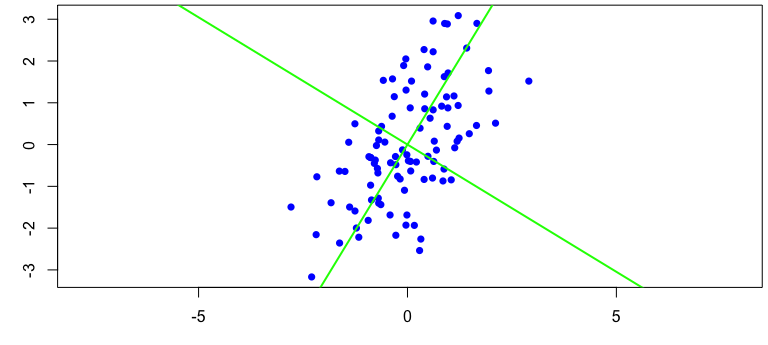


Figure 1: Principal components are eigenvectors of the covariance or correlation matrix

rotation of the coordinate system so that

* first component explains most variance
* second component explains most of the remaining variance and is perpendicular to the first one
* third component explains most of the remaining variance and is perpendicular to the first two
* …

becomes   
where  
 with being loadings

pca <- princomp(cbind(x,y), cor=TRUE)  
loadings(pca)

##   
## Loadings:  
## Comp.1 Comp.2  
## x 0.707 0.707  
## y 0.707 -0.707  
##   
## Comp.1 Comp.2  
## SS loadings 1.0 1.0  
## Proportion Var 0.5 0.5  
## Cumulative Var 0.5 1.0

loadings of a component can be multiplied by -1

proportion of variance explained by each component  
number of components = number of variables

summary(pca)

## Importance of components:  
## Comp.1 Comp.2  
## Standard deviation 1.2546821 0.6525127  
## Proportion of Variance 0.7871136 0.2128864  
## Cumulative Proportion 0.7871136 1.0000000

outlook: components with low variance are shrinked to a higher degree in Ridge regression

### Inferential statistics

there is never a “yes-or-no” answer  
there will always be uncertainty  
Amrhein (2017)[<https://peerj.com/preprints/26857>]

The decision whether an effect is important or not cannot not be done based on data alone. For a decision we should carefully consider the consequences of each decision, the aims we would like to achieve and the data. Consequences, needs and wishes of different stakeholders can be formally combined with the information in data by using methods of the decision theory. In most data analyses, particularly in basic research and when working on case studies, we normally do not consider consequences of decisions. In these cases, our job is extracting the information of data so that this information later can be used by other scientists, stakeholders and politicians to make decisions.

Therefore, statistics is describing pattern in data and quantifying the uncertainty of the described patterns that is due to the fact that the data is just a (small) random sample from the population we would like to know.

Quantification of uncertainty is only possible if

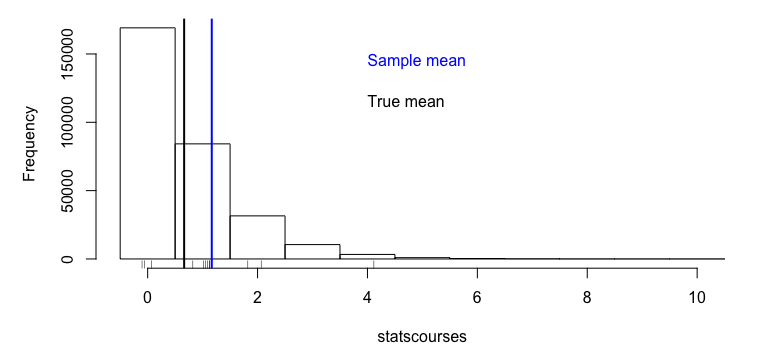
1. the mechanisms under study are known
2. the observations are a random sample from the population of interest

Solutions:  
to 1. working with models and reporting assumptions  
to 2. study design

reported uncertainties always are too small!

Example: Number of stats courses before starting a PhD among all PhD students

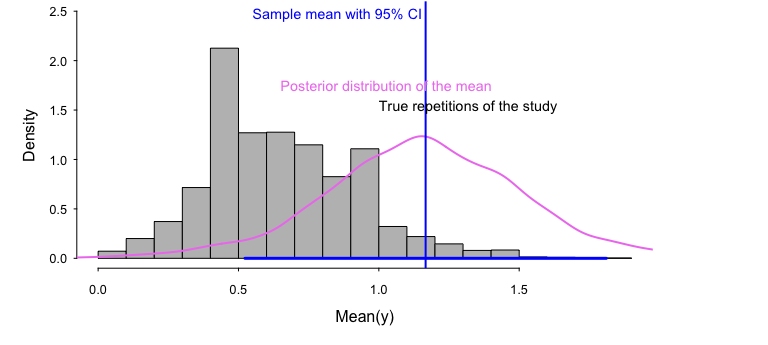
# simulate the virtual true population  
set.seed(235325) # set seed for random number generator  
  
# simulate fake data of the whole population  
statscourses <- rpois(300000, rgamma(300000, 2, 3))   
  
# draw a random sample from the population  
n <- 12 # sample size  
y <- sample(statscourses, 12, replace=FALSE)



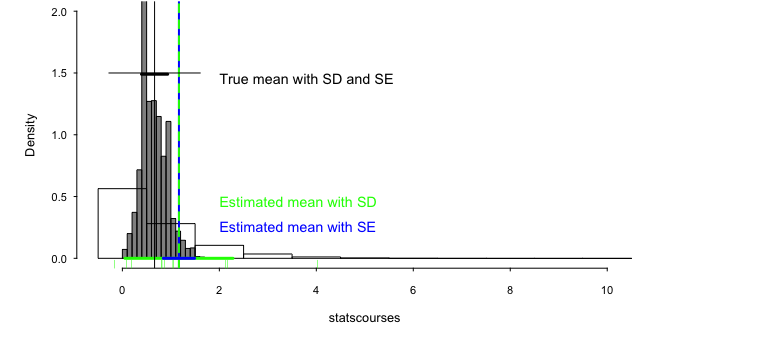
We observe the sample mean, what do we know about the population mean?

Frequentist solution: How would the sample mean scatter, if we repeat the study many times?

Bayesian solution: For any possible value, what is the probability that it is the true population mean?



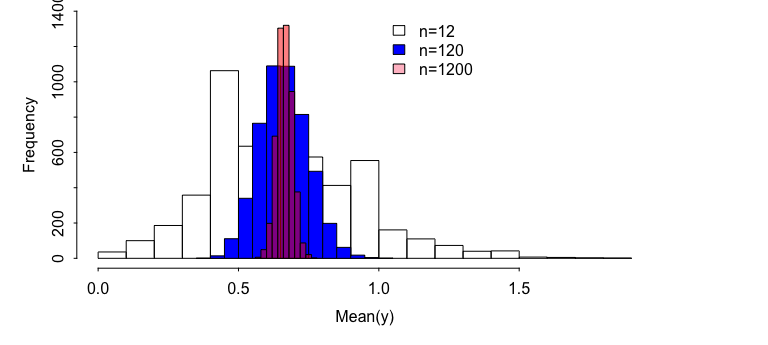
## Standard deviation and standard error



frequentist SE = SD/sqrt(n)

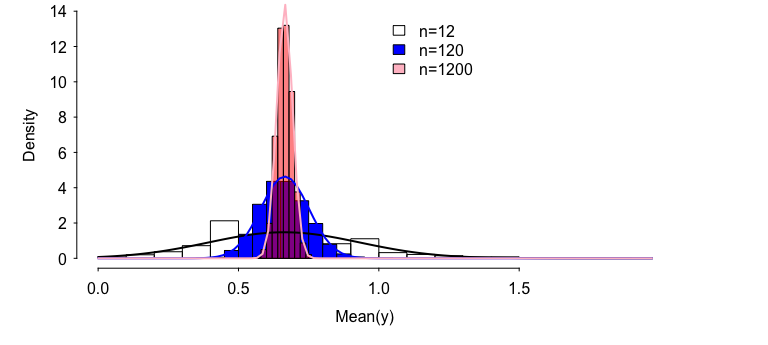
Bayesian SE = SD of posterior distribution

## Central limit theorem / law of large numbers



normal distribution = Gaussian distribution

, ,



## Bayes theorem

|  |  |  |  |
| --- | --- | --- | --- |
| car | flowers | wine | **sum** |
| no | 5 | 3 | **8** |
| yes | 1 | 4 | **5** |
| ——- | —————- | —————- | ——————- |
| **sum** | **6** | **7** | **13** |

What is the probability that the person likes wine given it has no car?  
 likes wine   
 no car

proportion car-free people among the wine liker

Knowing whether a persons owns a car increases the knowledge of the birthday preference.

## Bayes theorem for continuous parameters

: posterior distribution

: likelihood, data model

: prior distribution

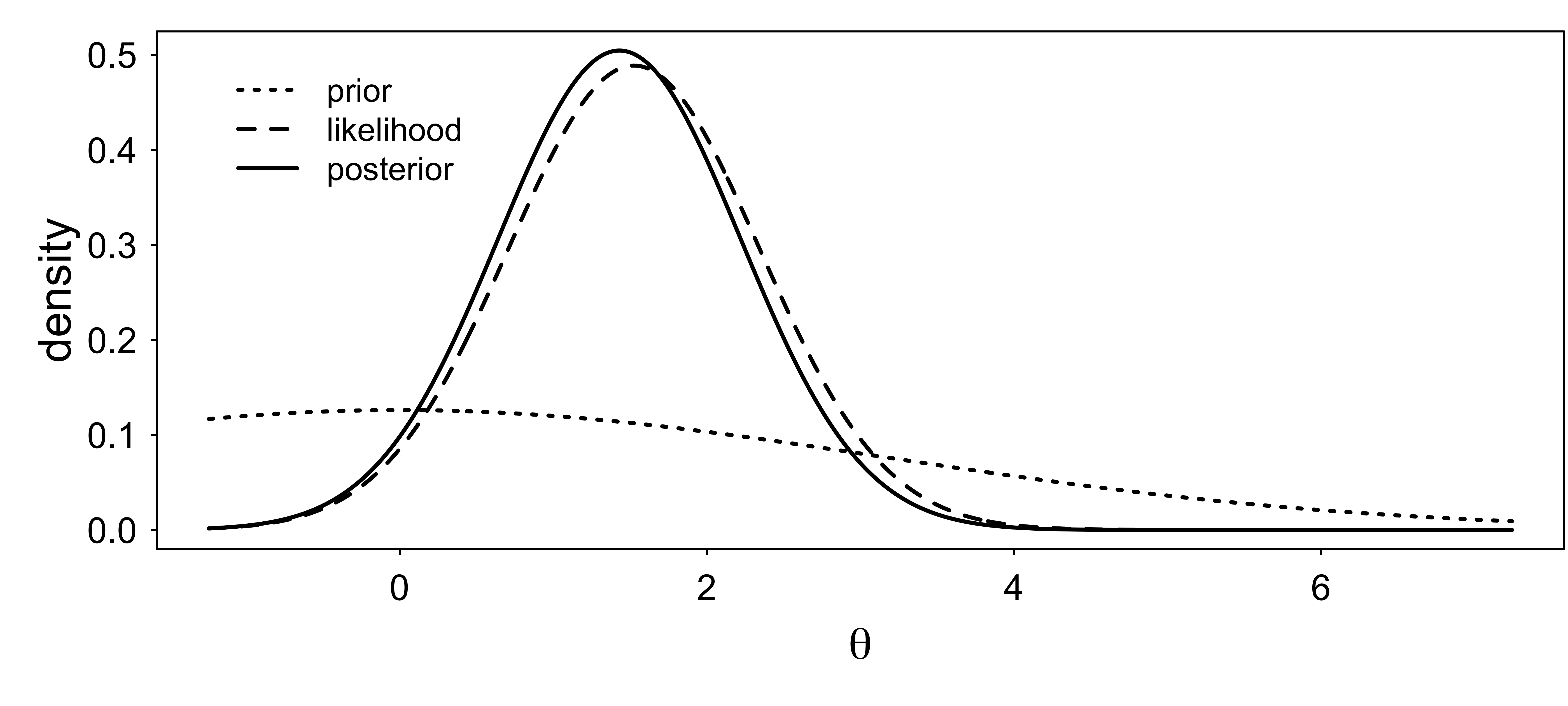
: scaling constant

## Single parameter model

, with known

, where and

is a sufficient statistics  
 is a conjugate prior for , with known.

Posterior mean = weighted average between prior mean and with weights equal to the precisions ( and ) 

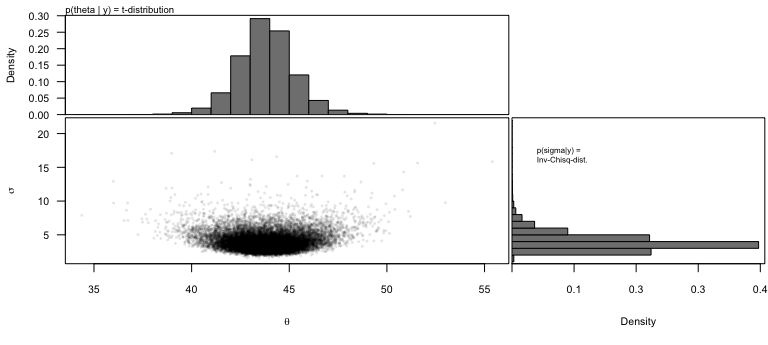
## A model with two parameters

# weight (g)  
y <- c(47.5, 43, 43, 44, 48.5, 37.5, 41.5, 45.5)  
n <- length(y)

conjugate prior

, with

and are sufficient statistics

Joint, marginal and conditional posterior distributions 

## t-distribution

marginal posterior distribution of a normal mean with unknown variance and conjugate prior distribution

degrees of freedom  
 location  
 scale

## Frequentist one-sample t-test

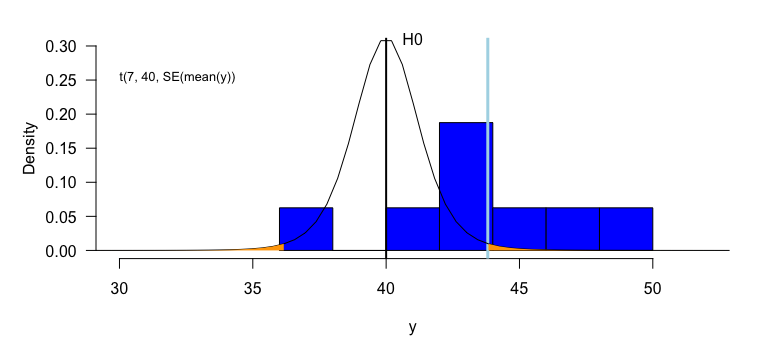
H0: the mean weight is equal to exactly 40g.

t.test(y, mu=40)

##   
## One Sample t-test  
##   
## data: y  
## t = 3.0951, df = 7, p-value = 0.01744  
## alternative hypothesis: true mean is not equal to 40  
## 95 percent confidence interval:  
## 40.89979 46.72521  
## sample estimates:  
## mean of x   
## 43.8125

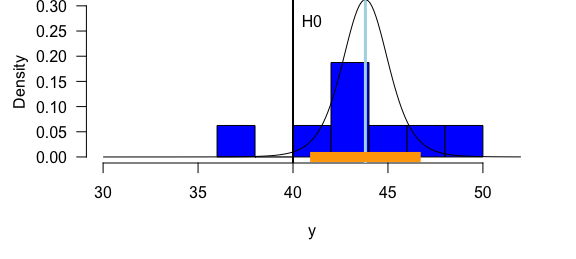
## Nullhypothesis test

p-value: Probability of the data or more extreme data given the null hypothesis is true.

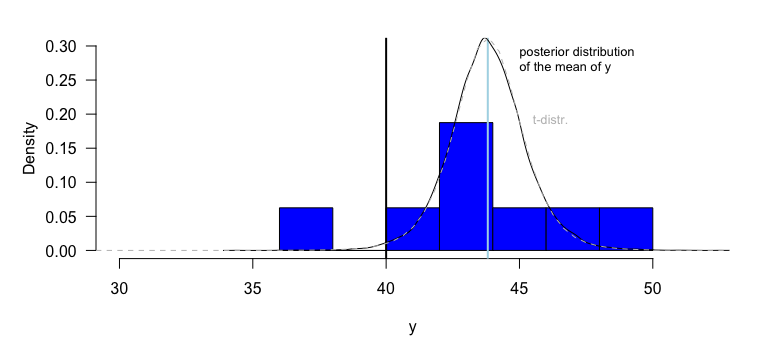


## Confidence interval

# lower limit of 95% CI  
mean(y) + qt(0.025, df=7)\*sd(y)/sqrt(n)   
# upper limit of 95% CI  
mean(y) + qt(0.975, df=7)\*sd(y)/sqrt(n)

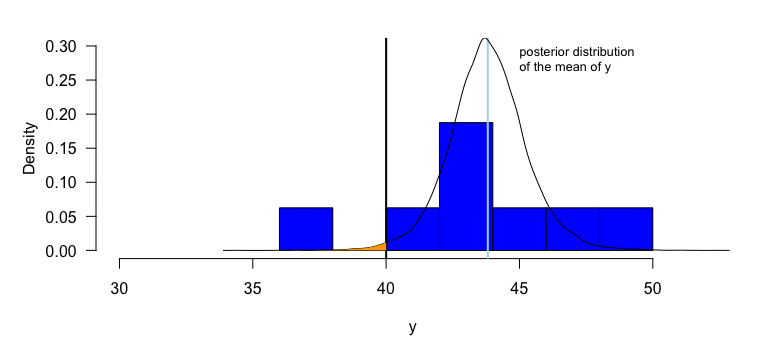


## Posterior distribution

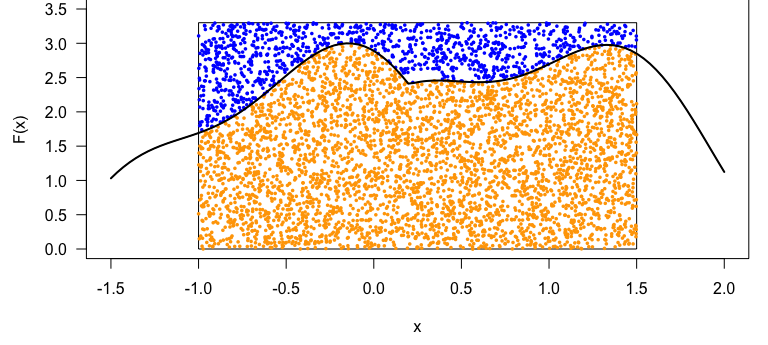


Two different theories - one single result!

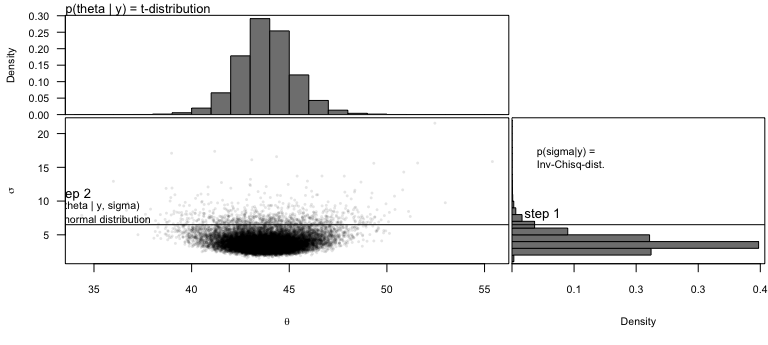
## Posterior probability

Probability 0.01 

## Monte Carlo simulation (parametric bootstrap)

Monte Carlo integration: numerical solution of 

sim is solving a mathematical problem by simulation How sim is simulating to get the marginal distribution of :



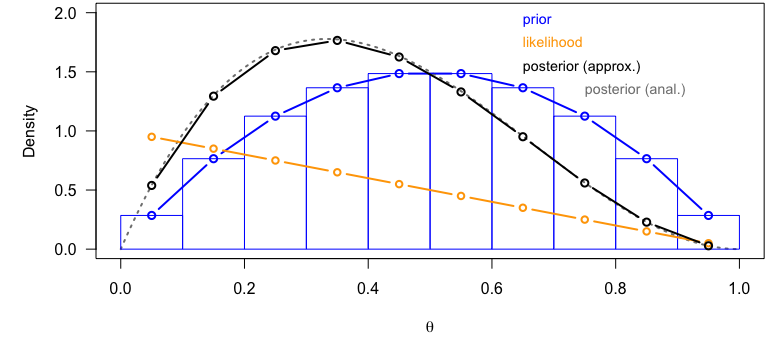
## 3 methods for getting the posterior distribution

* analytically
* approximation
* Monte Carlo simulation

## Grid approximation

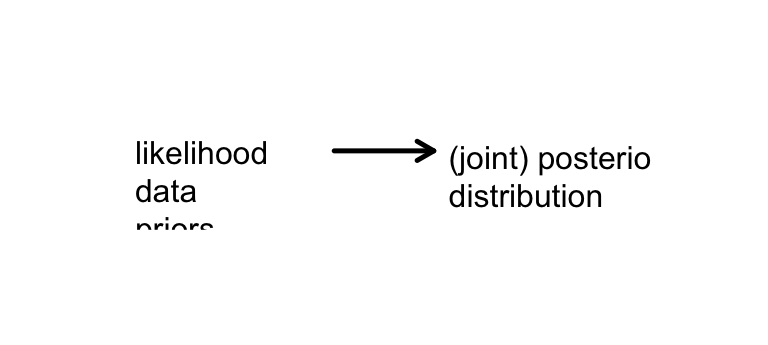
For example, one coin flip (Bernoulli model)

data: y=0 (a tail)  
likelihood:



## Monte Carlo simulations

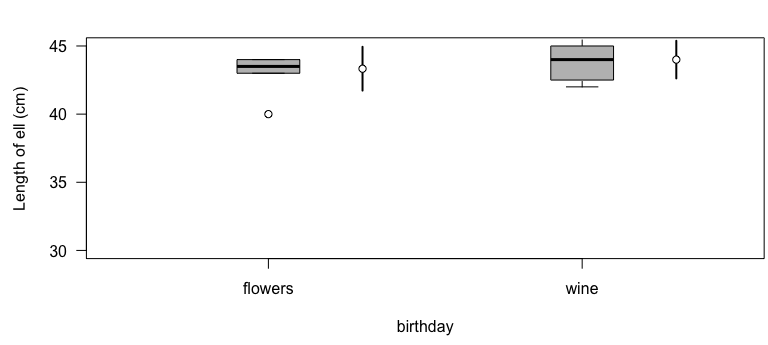
* Markov chain Monte Carlo simulation (BUGS, Jags)
* Hamiltonian Monte Carlo (Stan)



## Comparison of the locations between two groups

Boxplot:  
Median, 50% box, extremes observation within 1.5 times the interquartile range, outliers

The uncertainties of the means do not show the uncertainty of the difference between the means!



## Difference between two means

mod <- lm(ell~birthday, data=dat)  
mod

##   
## Call:  
## lm(formula = ell ~ birthday, data = dat)  
##   
## Coefficients:  
## (Intercept) birthdaywine   
## 43.3333 0.6667

bsim <- sim(mod, n.sim=nsim)  
quantile(bsim@coef[,2], prob=c(0.025, 0.5, 0.975))

## 2.5% 50% 97.5%   
## -1.6735535 0.6669283 2.9791876

## Two-sample t-test

t.test(ell~birthday, data=dat, var.equal=TRUE)

##   
## Two Sample t-test  
##   
## data: ell by birthday  
## t = -0.63369, df = 11, p-value = 0.5392  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -2.982185 1.648851  
## sample estimates:  
## mean in group flowers mean in group wine   
## 43.33333 44.00000

## Wilxocon test

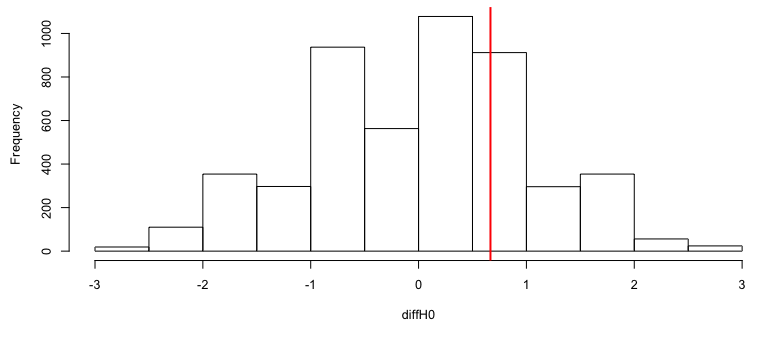
wilcox.test(ell~birthday, data=dat)

##   
## Wilcoxon rank sum test with continuity correction  
##   
## data: ell by birthday  
## W = 18, p-value = 0.7172  
## alternative hypothesis: true location shift is not equal to 0

## Randomisation test

diffH0 <- numeric(nsim)  
for(i in 1:nsim){  
 randbirthday <- sample(dat$birthday)  
 rmod <- lm(ell~randbirthday, data=dat)  
 diffH0[i] <- coef(rmod)[2]  
}  
mean(abs(diffH0)>abs(coef(mod)[2])) # p-value

## [1] 0.49



* Produces the distribution of a test statistics given the null hypothesis.
* assumption: all observations are independent
* becomes unfeasible when data is structured

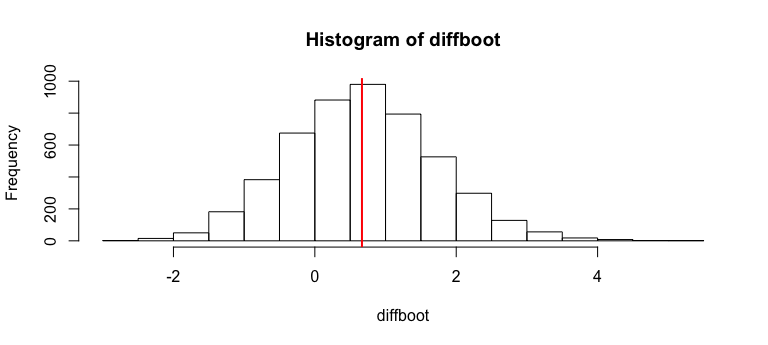
## Bootstrap

diffboot <- numeric(nsim)  
for(i in 1:nsim){  
 nbirthday <- 1  
 while(nbirthday==1){  
 bootrows <- sample(1:nrow(dat), replace=TRUE)  
 nbirthday <- length(unique(dat$birthday[bootrows]))  
 }  
 rmod <- lm(ell~birthday, data=dat[bootrows,])  
 diffboot[i] <- coef(rmod)[2]  
}  
quantile(diffboot, prob=c(0.025, 0.975))

## 2.5% 97.5%   
## -1.275069 2.833333

* result is a confidence interval
* assumption: all observations are independent!

hist(diffboot); abline(v=coef(mod)[2], lwd=2, col="red")



## F-distribution

Ratios of sample variances drawn from populations with equal variances follow an F-distribution. The density function of the F-distribution is even more complicated than the one of the t-distribution! We do not copy it here. Further, we have not yet met any Bayesian example where the F-distribution is used (that does not mean that there is no). It is used in frequentist analyses in order to compare variances, and, within the ANOVA, to compare means between groups. If two variances only differ because of natural variance in the data (nullhypothesis) then .

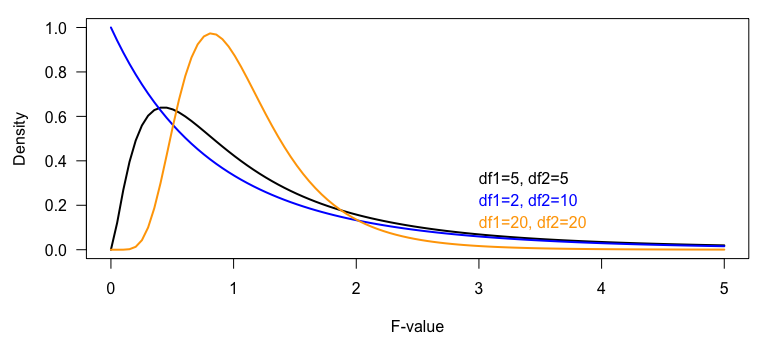


Figure 2: Different density functions of the F statistics

### Analysis of variance ANOVA

The aim of an ANOVA is to compare means of groups. In a frequentist analysis, this is done by comparing the between-group with the within-group variance. The result of a Bayesian analysis is the joint posterior distribution of the group means.

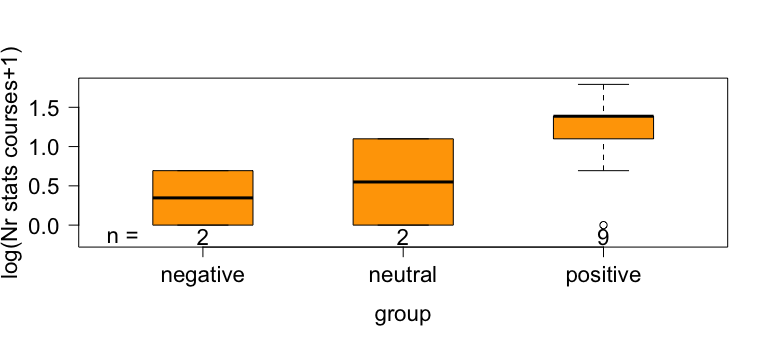


Figure 3: Number of stats courses students have taken before starting a PhD in relation to their feeling about statistics.

In the frequentist ANOVA, the following three sum of squared distances (SS) are used to calculate the total, the between- and within-group variances:  
Total sum of squares = SST =   
Within-group SS = SSW = : unexplained variance  
Between-group SS = SSB = : explained variance

The between-group and within-group SS sum to the total sum of squares: SST=SSB+SSW. Attention: this equation is only true in any case for a simple one-way ANOVA (just one grouping factor). If the data are grouped according to more than one factor (such as in a two- or three-way ANOVA), then there is one single solution for the equation only when the data is completely balanced, i.e. when there are the same number of observations in all combinations of factor levels. For non-balanced data with more than one grouping factor, there are different ways of calculating the SSBs, and the result of the F-test described below depends on the order of the predictors in the model.

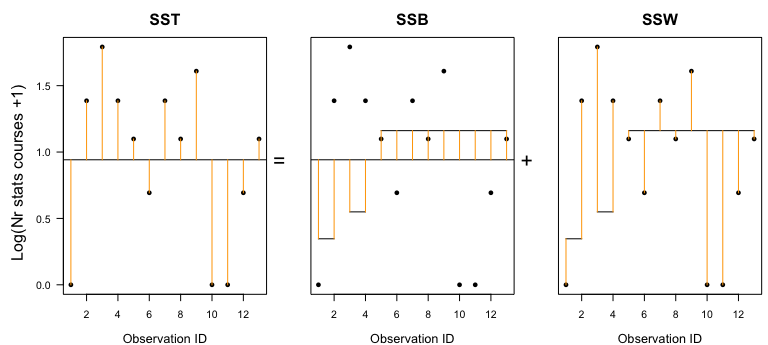


Figure 4: Visualisation of the total, between-group and within-group sum of squares. Points are observations; long horizontal line is the overall mean; short horizontal lines are group specific means.

In order to make SSB and SSW comparable, we have to divide them by their degrees of freedoms. For the within-group SS, SSW, the degrees of freedom is the number of obervations minus the number of groups (), because means have been estimated from the data. If the means are fixed and data points are known, then the last data points are defined, i.e., they cannot be chosen freely. For the between-group SS, SSB, the degrees of freedom is the number of groups minus 1 (the minus 1 stands for the overall mean).

* MSB = SSB/df\_between, MSW = SSW/df\_within

It can be shown (by mathematicians) that, given the nullhypothesis, the mean of all groups are equal , then the mean squared errors between groups (MSB) is expected to be equal to the mean squared errors within the groups (MSW). Therefore, the ration MSB/MSW is expected to follow an F-distribution given the nullhypothesis is true.

* MSB/MSW ~ F(df\_between, df\_within)

The Bayesian analysis for comparing group means consists of calculating the posterior distribution for each group mean and then drawing inference from these posterior distributions. A Bayesian one-way ANOVA involves the following steps:  
1. Decide for a data model: We, here, assume that the measurements are normally distributed around the group means. In this example here, we transform the outcome variable in order to better meet the normal assumption. Note: the frequentist ANOVA makes exactly the same assumptions. We can write the data model: with , where the -function is an indicator function taking on 1 if the expression is true and 0 otherwise. This model has 4 parameters: , , and .

# fit a normal model with 3 different means  
mod <- lm(log(nrcourses+1)~statsfeeling, data=dat)

1. Choose a prior distribution for each model parameter: In this example, we choose flat prior distributions for each parameter. By using these priors, the result should not remarkably be affected by the prior distributions but almost only reflect the information in the data. We choose so-called improper prior distributions. These are completely flat distributions that give all parameter values the same probability. Such distributions are called improper because the area under the curve is not summing to 1 and therefore, they cannot be considered to be proper probability distributions. However, they can still be used to solve the Bayesian theorem.
2. Solve the Bayes theorem: The solution of the Bayes theorem for the above priors and model is implemented in the function sim of the package arm.

# calculate numerically the posterior distributions of the model   
# parameters using flat prior distributions  
nsim <- 5000  
set.seed(346346)  
bsim <- sim(mod, n.sim=nsim)

1. Display the joint posterior distributions of the group means

# calculate group means from the model parameters  
newdat <- data.frame(statsfeeling=levels(dat$statsfeeling))  
X <- model.matrix(~statsfeeling, data=newdat)  
fitmat <- matrix(ncol=nsim, nrow=nrow(newdat))  
for(i in 1:nsim) fitmat[,i] <- X%\*%bsim@coef[i,]  
hist(fitmat[1,], freq=FALSE, breaks=seq(-2.5, 4.2, by=0.1), main=NA, xlab="Group mean of log(number of courses +1)", las=1, ylim=c(0, 2.2))  
hist(fitmat[2,], freq=FALSE, breaks=seq(-2.5, 4.2, by=0.1), main=NA, xlab="", las=1, add=TRUE, col=rgb(0,0,1,0.5))  
hist(fitmat[3,], freq=FALSE, breaks=seq(-2.5, 4.2, by=0.1), main=NA, xlab="", las=1, add=TRUE, col=rgb(1,0,0,0.5))  
legend(2,2, fill=c("white",rgb(0,0,1,0.5), rgb(1,0,0,0.5)), legend=levels(dat$statsfeeling))

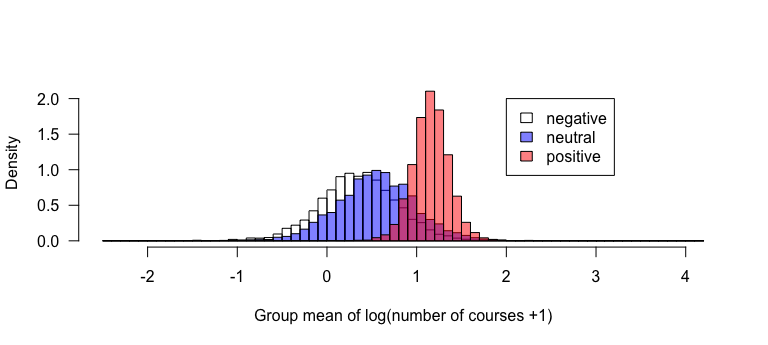


Figure 5: Posterior distributions of the mean number of stats courses PhD students visited before starting the PhD grouped according to their feelings about statistics.

Based on the posterior distributions of the group means, we can extract derived quantities depending on our interest and questions. Here, for example, we could extract the posterior probability of the hypothesis that students with a positive feeling about statistics have a better education in statistics than those with a neutral or negative feeling about statistics.

# P(mean(positive)>mean(neutral))  
mean(fitmat[3,]>fitmat[2,])

## [1] 0.9004

# P(mean(positive)>mean(negative))  
mean(fitmat[3,]>fitmat[1,])

## [1] 0.953

## Chisquare test

The chisquare test is used for two frequentist purposes.  
1. Testing for correlations between two categorical variables.  
2. Comparison of two distributions (goodness of fit test)

When testing for correlations between two categorical variables, then the nullhypothesis is “there is no correlation”. The data can be displayed in cross-tables.

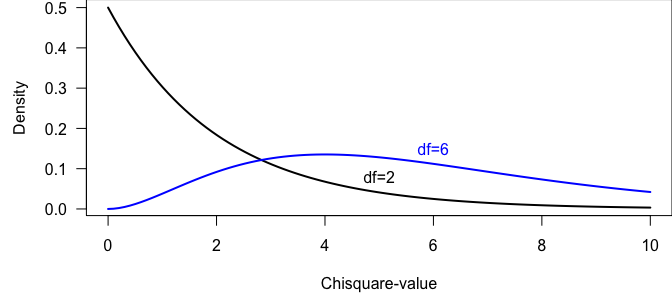
# Example: correlation between birthday preference and car ownership  
table(dat$birthday, dat$car)

##   
## N Y  
## flowers 5 1  
## wine 3 4

Given the nullhypothesis is true, we expect that the distribution of the data in each column of the cross-table is similar to the distribution of the row-sums. And, the distribution of the data in each row should be similar to the distribution of the column-sums. The chisquare test statistics measures the deviation of the data from this expected distribution of the data in the cross-table.

For calculating the chisquare test statistics , we first have to obtain for each cell in the cross-table the expected value = rowsum\*colsum/total.

measures the difference between the observed and expected values as:  
 where is the number of rows and is the number of columns. The -distribution has 1 parameter, the degrees of freedom = .



R is calculating the value for specific cross-tables, and it is also giving the p-values, i.e., the probability of obtaining the observed or a higher value given the nullhypothesis is true by comparing the observed with the corresponding chisquare distribution.

chisq.test(table(dat$birthday, dat$car))

##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: table(dat$birthday, dat$car)  
## X-squared = 0.85312, df = 1, p-value = 0.3557

The warning (that is suppressed in the rmarkdown version, but that you will see if you run the code on your own computer) is given, because in our example some cells have counts less than 5. In such cases, the Fisher’s exact test should be preferred. This test calculates the p-value analytically using probability theory, whereas the chisquare test relies on the assumption that the value follows a chisquare distribution. The latter assumption holds better for larger sample sizes.

fisher.test(table(dat$birthday, dat$car))

##   
## Fisher's Exact Test for Count Data  
##   
## data: table(dat$birthday, dat$car)  
## p-value = 0.2657  
## alternative hypothesis: true odds ratio is not equal to 1  
## 95 percent confidence interval:  
## 0.3369761 391.2320030  
## sample estimates:  
## odds ratio   
## 5.699654

## Bayesian way of analysing correlations between categorical variables

For a Bayesian analysis of cross-table data, a data model has to be found. There are several possibilities that could be used:

* a so-called log-linear model (Poisson model) for the counts in each cell of the cross-table.
* a binomial or a multinomial model for obtaining estimates of the proportions of data in each cell

These models provide possibilities to explore the patterns in the data in more details than a chisquare test.

# log-linear model  
mod <- glm(count~birthday+car + birthday:car,   
 data=datagg, family=poisson)  
bsim <- sim(mod, n.sim=nsim)  
round(t(apply(bsim@coef, 2, quantile,   
 prob=c(0.025, 0.5, 0.975))),2)

## 2.5% 50% 97.5%  
## (Intercept) 0.74 1.61 2.47  
## birthdaywine -1.91 -0.50 0.90  
## carY -3.71 -1.62 0.52  
## birthdaywine:carY -0.67 1.88 4.40

The interaction parameter measures the strength of the correlation. To quantitatively understand what a parameter value of 1.90 means, we have to look at the interpretation of all parameter values. We do that here quickly without a thorough explanation, because we already explained the Poisson model in chapter 8 of (Korner-Nievergelt et al. [2015](#ref-KornerNievergelt2015)).

The intercept 1.61 corresponds to the logarithm of the count in the cell “flowers” and “N” (number of students who prefer flowers as a birthday present and who do not have a car), i.e., = 5.00. The exponent of the second parameter corresponds to the multiplicative difference between the counts in the cells “flowers and N” and “wine and N”, i.e., count in the cell “wine and N” = = exp(1.61)exp(-0.51) = 3.00. The third parameter measures the multiplicative difference in the counts between the cells “flowers and N” and “flowers and Y”, i.e., count in the cell “flowers and Y” = = exp(1.61)exp(-1.61) = 1.00. Thus, the third parameter is the difference in the logarithm of the counts between the car owners and the car-free students for those who prefer flowers. The interaction parameter is the difference of this difference between the students who prefer wine and those who prefer flowers. This is difficult to intuitively understand. Here is another try to formulate it: The interaction parameter measures the difference in the logarithm of the counts in the cross-table between the row-differences between the columns. Maybe it becomes clear, when we extract the count in the cell “wine and Y” from the model parameters: = exp(1.61)exp(-0.51)exp(-1.61)exp(1.90) = 4.00.

Alternatively, we could estimate the proportions of flower and wine preferers within each group of car owners and car-free students using a binomial model. For an explanation of the binomial model, see chapter 8 of (Korner-Nievergelt et al. [2015](#ref-KornerNievergelt2015)).

# binomial model  
tab <- table(dat$car,dat$birthday)  
mod <- glm(tab~rownames(tab), family=binomial)  
bsim <- sim(mod, n.sim=nsim)

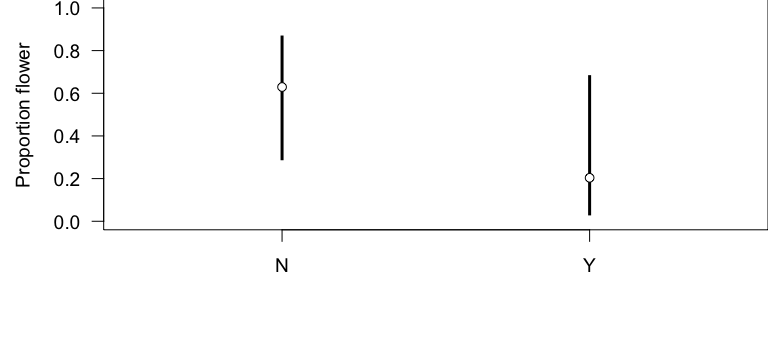


Figure 6: Estimated proportion of students that prefer flowers over wine as a birthday present among the car-free students (N) and the car owners (Y). Given are the median of the posterior distribution (circle). The bar extends between the 2.5% and 97.5% quantiles of the posterior distribution.

## Summary

Bayesian data analysis = applying the Bayes theorem for summarising knowledge based on data, priors and the model assumptions.

Frequentist statistics = quantifying uncertainty by hypothetical repetitions

# Data analysis step by step

In this chapter we provide a checklist with some guidance for data analysis. However, do not expect the list to be complete and for different studies, a different order of the steps may make more sense. We usually repeat steps 3.2 to 3.8 until we find a model that fit the data well and that is realistic enough to be useful for the intended purpose. Data analysis is always a lot of work and, often, the following steps have to be repeated many times until we find a useful model.

There is a chance and danger at the same time: we may find interesting results that answer different questions than we asked originally. They may be very exciting and important, however they may be biased. We can report such findings, but we should state that they appeared (more or less by chance) during the data exploration and model fitting phase, and we have to be aware that the estimates may be biased because the study was not optimally designed with respect to these findings. It is important to always keep the original aim of the study in mind. Do not adjust the study question according to the data. We also recommend reporting what the model started with at the first iteration and describing the strategy and reasoning behind the model development process.

## Plausibility of Data

Prepare the data and check graphically, or via summary statistics, whether all the data are plausible. Prepare the data so that errors (typos, etc.) are minimal, for example, by double-checking the entries. See chapter 4 for useful R-code that can be used for data preparation and to make plausibility controls.

## Relationships

Think about the direct and indirect relationships among the variables of the study. We normally start a data analysis by drawing a sketch of the model including all explanatory variables and interactions that may be biologically meaningful. We will most likely repeat this step after having looked at the model fit. To make the data analysis transparent we should report every model that was considered. A short note about why a specific model was considered and why it was discarded helps make the modeling process reproducible.

## Data Distribution

What is the nature of the variable of interest (outcome, dependent variable)? At this stage, there is no use of formally comparing the distribution of the outcome variable to a statistical distribution, because the rawdata is not required to follow a specific distribution. The models assume that conditional on the explanatory variables and the model structure, the outcome variable follows a specific distribution. Therefore, checking how well the chosen distribution fits to the data is done after the model fit 3.8. This first choice is solely done based on the nature of the data. Normally, our first choice is one of the classical distributions for which robust software for model fitting is available.

Here is a rough guideline for this first choice:

continuous measurements normal distribution  
> exceptions: time-to-event data see survival analysis

count Poisson or negative-binomial distribution

count with upper bound (proportion) binomial distribution

binary Bernoully distribution

rate (count by a reference) Poisson including an offset

nominal multinomial distribution

Chapter 5 gives an overview of the distributions that are most relevant for ecologists.

## Preparation of Explanatory Variables

1. Look at the distribution (histogram) of every explanatory variable: Linear models do not assume that the explanatory variables have any specific distribution. Thus there is no need to check for a normal distribution! However, very skewed distributions result in unequal weighting of the observations in the model. In extreme cases, the slope of a regression line is defined by one or a few observations only. We also need to check whether the variance is large enough, and to think about the shape of the expected effect. The following four questions may help with this step:
2. Is the variance (of the explanatory variable) big enough so that an effect of the variable can be measured?
3. Is the distribution skewed? If an explanatory variable is highly skewed, it may make sense to transform the variable (e.g., log, square-root).
4. Does it show a bimodal distribution? Consider making the variable binary.
5. Is it expected that a change of 1 at lower values for x has the same biological effect as a change of 1 at higher values of x? If not, a trans- formation (e.g., log) could linearize the relationship between x and y.
6. Centering: Centering () is a transformation that produces a variable with a mean of 0. Centering is optional. We have two reasons to center a predictor variable. First, it helps the model fitting algorithm to better converge because it reduces correlations among model parameters. Second, with centered predictors, the intercept and main effects in the linear model are better interpretable (they are measured at the center of the data instead of at the covariate value of 0 which may be far off).
7. Scaling: Scaling (, where is a constant) is a transformation that changes the unit of the variable. Also scaling is optional. We have three reasons to scale an predictor variable. First, to make the effect sizes better understandable. For example, a population change from one year to the next may be very small and hard to interpret. When we give the change for a 10-year period, its ecological meaning is better understandable. Second, to make the estimate of the effect sizes comparable between variables, we may use . The resulting variable has a unit of one standard deviation. A standard deviation may be comparable between variables that oritinally are measured in different units (meters, seconds etc). Gelman and Hill ([2007](#ref-Gelman2007)) (p. 55 f) propose to scale the variables by two times the standard deviation () to make effect sizes comparable between numeric and binary variables. Third, scaling can be important for model convergence, especially when polynomials are included. Also, consider the use of orthogonal polynomials, see Chapter 4.2.9 in Korner-Nievergelt et al. ([2015](#ref-KornerNievergelt2015)).
8. Collinearity: Look at the correlation among the explanatory variables (pairs plot or correlation matrix). If the explanatory variables are correlated, go back to step 2. Also, Chapter 4.2.7 in Korner-Nievergelt et al. ([2015](#ref-KornerNievergelt2015)) discusses collinearity.
9. Are interactions and polynomial terms needed in the model? If not already done in step 2, think about the relationship between each explanatory variable and the dependent variable. Is it linear or do polynomial terms have to be included in the model? If the relationship cannot be described appropriately by polynomial terms, think of a nonlinear model or a generalized additive model (GAM). May the effect of one explanatory variable depend on the value of another explanatory variable (interaction)?

## Data Structure

After having taken into account all of the (fixed effect) terms from step 4: are the observations independent or grouped/structured? What random factors are needed in the model? Are the data obviously temporally or spatially correlated? Or, are other correlation structures present, such as phylogenetic relationships? Our strategy is to start with a rather simple model that may not account for all correlation structures that in fact are present in the data. We first, only include those that are known to be important a priory. Only when residual analyses reveals important additional correlation structures, we include them in the model.

## Define Prior Distributions

Decide whether we would like to use informative prior distributions or whether we would like use priors that only have a negligible effect on the results. When the results are later used for informing authorities or for making a decision (as usual in applied sciences), then we would like to base the results on all information available. Information from the literature is then used to construct informative prior distributions. In contrast to applied sciences, in basic research we often would like to show only the information in the data that should not be influenced by earlier results. Therefore, in basic research we look for priors that do not influence the results.

## Fit the Model

Fit the model.

## Check Model

We assess model fit by graphical analyses of the residuals (Chapter 6 in Korner-Nievergelt et al. ([2015](#ref-KornerNievergelt2015))), by predictive model checking (Section 10.1 in Korner-Nievergelt et al. ([2015](#ref-KornerNievergelt2015))), or by sensitivity analysis (Chapter 15 in Korner-Nievergelt et al. ([2015](#ref-KornerNievergelt2015))).

For non-Gaussian models it is often easier to assess model fit using pos- terior predictive checks (Chapter 10 in Korner-Nievergelt et al. ([2015](#ref-KornerNievergelt2015))) rather than residual analyses. Posterior predictive checks usually show clearly in which aspect the model failed so we can go back to step 2 of the analysis. Recognizing in what aspect a model does not fit the data based on residual plots improves with experience. Therefore, we list in Chapter 16 of Korner-Nievergelt et al. ([2015](#ref-KornerNievergelt2015)) some patterns that can appear in residual plots together with what these patterns possibly indicate. We also indicate what could be done in the specific cases.

## Model Uncertainty

If, while working through steps 1 to 8, possibly repeatedly, we came up with one or more models that fit the data reasonably well, we then turn to the methods presented in Chapter 11 (Korner-Nievergelt et al. ([2015](#ref-KornerNievergelt2015))) to draw inference from more than one model. If we have only one model, we proceed to 3.10.

## Draw Conclusions

Simulate values from the joint posterior distribution of the model parameters (sim or Stan). Use these samples to present parameter uncertainty, to obtain posterior distributions for predictions, probabilities of specific hypotheses, and derived quantities.

## Further reading

* [R for Data Science by Garrett Grolemund and Hadley Wickham](http://r4ds.had.co.nz): Introduces the tidyverse framwork. It explains how to get data into R, get it into the most useful structure, transform it, visualise it and model it.

# Data preparation

## Basic operations

Alle Packete laden library(tidyverse) oder nur library(dplyr).

dat <- iris %>%   
 as.tibble() %>%   
 filter(Sepal.Length > 5) %>%   
 group\_by(Species) %>%   
 summarise(n = n(),  
 mittel = mean(Petal.Length))

## Joining tables

Beschreiben wie left\_join funktioniert.

## Further reading

* [R for Data Science by Garrett Grolemund and Hadley Wickham](http://r4ds.had.co.nz): Introduces the tidyverse framwork. It explains how to get data into R, get it into the most useful structure, transform it, visualise it and model it.

# Probability distributions

## Introduction

hist(rnorm(1000))

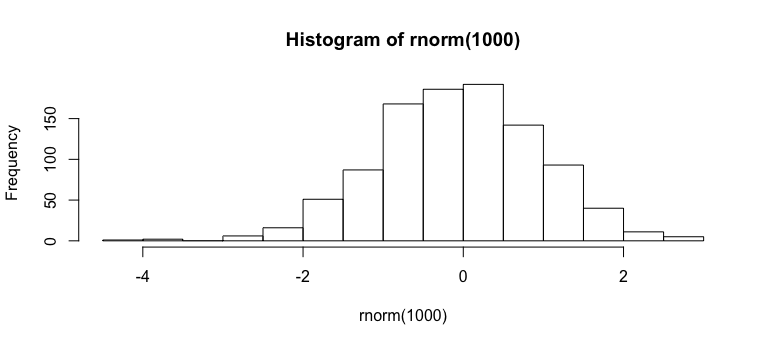


Figure 7: Das ist ein Versuch.

## Normal distribution

xxx

## Poisson distribution

xxx

## Gamma distribution

xxx

### Cauchy distribution

We sometimes use the Cauchy distiribution to specify the prior distirbution of the standart deviation and similar parameters in a model. An example for this is the regression model that we used to introduce Stan (Chapter 12.3).

# Visualizations

## Short Checklist for figures

1. The figure should represent the answer to the study question. Often, the classical types of plots such as a scatterplot, a bar plot, or an effects plot are sufficient. However, in many cases, adding a little bit of creativity can greatly improve readability or the message of the figure.

2.Label the x- and y-axes. Make sure that the units are indicated either within the title or in the figure legend.

1. Starty-axisatzeroifthereferencetozeroisimportantfortheinterpretationof effects. The argument ylim[c(0, max(dat$y)) in R is used for this purpose.
2. Scale the axes so that all data are shown. Make sure that sample size is indicated either in the figure or in the legend (or, at least, easy to find in the text).
3. Use interpretable units. That means, if the variable on the x-axis has been z-transformed to fit the model, back-transform the effects to the original scale.
4. Give the raw data whenever possible. Sometimes, a significant effect cannot be seen in the raw data because so many other variables have an influence on the outcome. Then, you may prefer showing the effect only (e.g., a regression line with a credible interval) and give the residual standard deviation in the figure legend. Even then, we think it is important to show the raw data graphically somewhere else in the paper or in the supplementary material. A scatterplot of the data can contain structures that are lost in summary statistics.
5. Draw the figures as simply as possible. Avoid 3D graphics. Delete all unnecessary elements.
6. Reduce the number of different colors to a minimum necessary. A color scale from orange to blue gives a gray scale in a black-and-white print. colorRampPalette(c("orange", "blue"))(5) produces five colors on a scale from orange to blue. Remember that around 8% of the northern European male population have difficulties distinguishing red from green but it is easier for them to distinguish orange from blue.

## Further reading

* [Data Visualization. A practical introduction](https://socviz.co): A practical introduction to data visulization in R. Good data visualizations also make it easier to communicate your ideas and findings to other people. Beyond that, producing effective plots from your own data is the best way to develop a good eye for reading and understanding graphs—good and bad—made by others.
* [Fundamentals of Data Visualization](https://serialmentor.com/dataviz/): A guide to making visualizations that accurately reflect the data, tell a story, and look professional. […] This is an online preview of the book “Fundamentals of Data Visualization” to be published with O’Reilly Media, Inc. Completed chapters will be posted here as they become available. The book is meant as a guide to making visualizations that accurately reflect the data, tell a story, and look professional. It has grown out of my experience of working with students and postdocs in my laboratory on thousands of data visualizations.

# Spatial analyses and maps

* Almost all the things that we can do with traditional geographic information system we can also do within R.
* If we do it in R we get the useful benefits of a script approach that allows for reproducible analyses (see Chapter 8) and that can be scaled to many more objects or larger data sets.
* Here we simply introduce the packages and functions that we most often use when working with spatial data.

## Data types

### Raster data

Very broadly speaking, we divide spatial data into two categories, raster data and all other types of data for points, lines or polygons. Raster data consists of a grid (i.e. a matrix) where each cell of the grid contains one or several values representing the spatial information. The R-package raster is very efficient for raster data.

We can use the function raster() to load raster data from files. Most of the common file formats for raster data such as .geotiff, .tif or .grd are supported. However, raster data can also be converted from tables with one column for the x-coordinates, one column for the y-coordinates and one column for the spatial information. The coordinates must be from a regular grid. For example the freely available [topographic data of Switzerland](https://www.bfs.admin.ch/bfs/de/home/dienstleistungen/geostat/geodaten-bundesstatistik/topografie.html) are collected on a 100m x 100m grid. In the following example the tibble elevations contains the elevation data from the canton of Aargau and is converted into raster data using the function rasterFromXYZ().

library(raster)  
ra <- rasterFromXYZ(elevation)  
plot(ra)

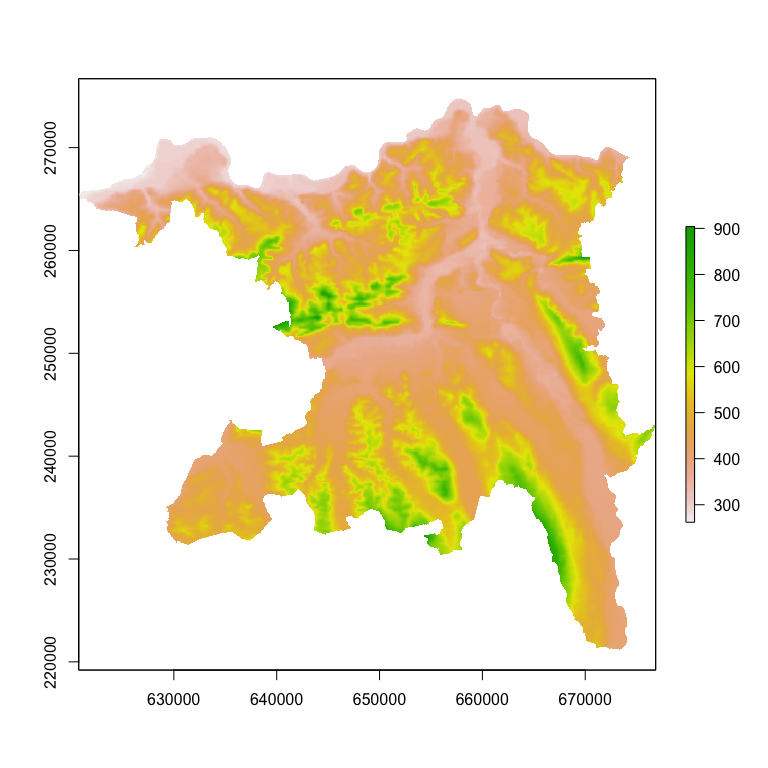


Figure 8: Meter above sea leavel (m) accross the canton Aargau in Switzerland.

### Geometry data

All geometry data types are composed of points. The spatial location of a point is defined by its x and y coordinates. Using several points one can then define lines (sequence of points connected by straight lines) and polygons (sequence of points that form a closed ring). Points, lines and plygons are the geometries we usually work with. We use the package sf to work with geometry data types. Its functions are very efficient to work with all spatial data other than raster data. It also links to GDAL (i.e. a computer software library for reading and writing raster and vector geospatial data formats) and proj.4 (i.e. a library for performing conversions between cartographic projections), which are important tools when woring with different sources of spatial data.

We can used the function st\_read() to read geometry data from file or database. In the following example, however, we convert the tibble frogs into a simple feature collection. The data file frogs, formatted as a tibble, contains different columns including the counts, variables that describe the ponds as well as the spatial coordinates of the counts. The simple feature collection looks rather similar to the original tibble, however instead of the x and y colomns it now contains the column geometry. With the simple feature collection we can work pretty much in the same way as we use to work with tibbles. For example we can filter only the data from 2011, select the geometries and plot them on the top of the raster with the elevation across the entire canton of Aargau (see 7.1.1 for the raster data).

library(sf)  
dat <- frogs %>% st\_as\_sf(coords = c("x", "y"), crs = 21781)  
plot(ra)  
dat %>% filter(year == 2011) %>%   
 st\_geometry() %>%   
 plot(add = TRUE, pch = "x")

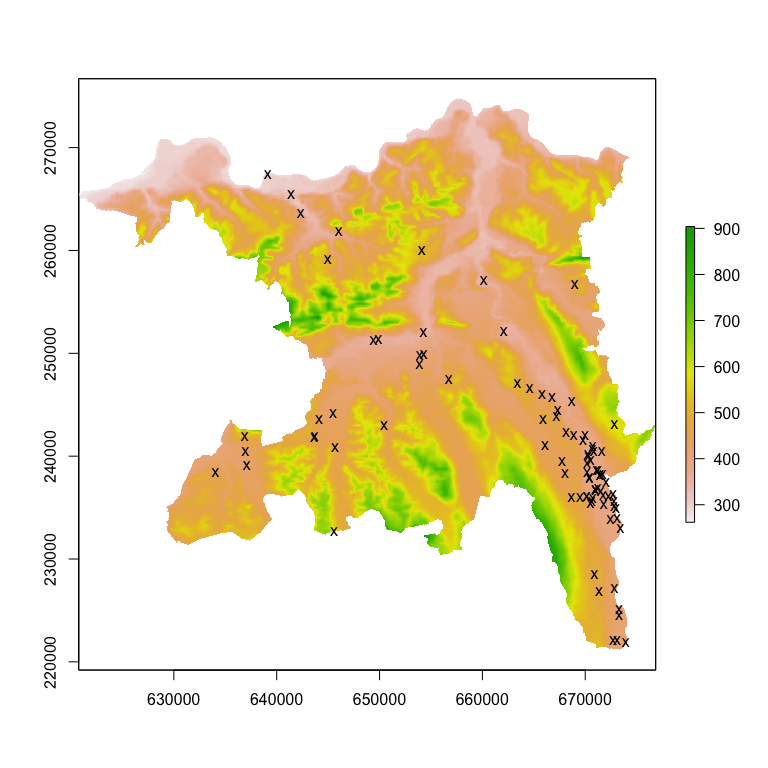


Figure 9: Locations of the ponds where frogs were counted in 2011. The background image shows the elevation (meter above sea level).

## Basic functions

In this chapter we shortly describe some functions that we often use when working with spatial data in R.

### Coordinate systems

An important aspect of spatial data is the coordinate reference system (CRS). A CRS determines for instance where the center of the map is, the units for the coordinates and others. [PROJ.4](https://proj4.org/) is an open source software library that is commonly used for CRS transformation. Most commonly used CRSs have been assigned a

HERE IS SOMETHING MISSING.

The EPSG (European Petroleum Survey Group) code is a unique ID that can be used to identify a CRS. Thus if we know the EPSG code it is rather simple to transform spatial data into other CRS. To search for the correct EPSG code we can use <https://www.epsg-registry.org> or <http://www.spatialreference.org>

The following code shows how to assign the CRS of existing data and how to transform the coordinate system for raster data and sf data, respectively.

# Assign CRS for raster data  
crs(ra) <- CRS("+init=epsg:21781")  
  
# Assign CRS for sf data  
st\_crs(dat) <- 21781  
  
# Transfrom raster data to WGS84  
projectRaster(ra, crs = CRS("+init=epsg:4326"))  
  
# Transfrom sf data to WGS84  
st\_transform(dat, crs = 4326)

### Joining spatial data

Joining two non-spatial datasets relies on a shared variable (key) using for instance the function left\_join(), as described in chapter 4.2. In a spatial context, we apply the exact same concept except for the key being a shared areas of geographic space. Note, however, that people with a background in geographic information systems may call these operations differently, such as spatial overlay or intersection.

For [geometry data](#geometrydata) we can use the function st\_join(). As an example, we aim to add the [biogegraphic Region](biogeographische%20regionen%20der%20schweiz) to the counts of the number of frogs in ponds of the Canton Aargau. We transform the frogs data into a spatial object (i.e. a simple feature collection). The polygons of the biogegraphic regions in Switzerland are available in the object bgr. To each of the polygons additional data are available and we aim to extract the information from the column BIOGREG\_R6, which contains the name of the biogeographic region. We add this column to the frogs data using the function st\_join() as following.

load("RData/bgr.RData")  
bgr <- bgr %>%  
 st\_transform(21781)  
dat <- frogs %>% st\_as\_sf(coords = c("x", "y"), crs = 21781)  
dat <- dat %>% st\_join(bgr["BIOGREG\_R6"])

## Further reading

* [Geocomputation with R](https://geocompr.robinlovelace.net): This online-book is aimed at people who want to do spatial data analysis, visualization and modeling using open source software and reproducible workflows.
* [Spatial Data Analysis and Modeling with R](http://rspatial.org/index.html): Online introduction to do spatial analyses with R. Good introduction to coordinate systems and projections.

# Reproducible Research

## Introduction

xxx

## Further reading

* [Rmarkdown](https://bookdown.org/yihui/rmarkdown/): The first official book authored by the core R Markdown developers that provides a comprehensive and accurate reference to the R Markdown ecosystem. With R Markdown, you can easily create reproducible data analysis reports, presentations, dashboards, interactive applications, books, dissertations, websites, and journal articles, while enjoying the simplicity of Markdown and the great power of R and other languages.
* [Bookdown by Yihui Xie](https://bookdown.org/yihui/bookdown/): A guide to authoring books with R Markdown, including how to generate figures and tables, and insert cross-references, citations, HTML widgets, and Shiny apps in R Markdown. The book can be exported to HTML, PDF, and e-books (e.g. EPUB). The book style is customizable. You can easily write and preview the book in RStudio IDE or other editors, and host the book wherever you want (e.g. bookdown.org). Our book is written using bookdown.

# Further topics

This is a collection of short introductions or links with commented R code that cover other topics that might be usful for ecologists.

## Bioacoustic analyse

Bioacoustic analyses are nicely covered in a blog by [Marcelo Araya-Salas](https://marce10.github.io).

## Python

Like R, python is a is a high-level programming language that is used by many ecologists. The [reticulate](https://rstudio.github.io/reticulate/index.html) package provides a comprehensive set of tools for interoperability between Python and R.

# Introduction to PART II

**Fehler! Linkreferenz ungültig.**

## Further reading

A really good introductory book to Bayesian data analyses is (McElreath [2016](#ref-McElreath2016)). This book starts with a thorough introduction to applying the Bayes theorem for drawing inference from data. In addition, it carefully discusses what can and what cannot be concluded from statistical results. We like this very much.

We like looking up statistical methods in papers and books written by Andrew Gelman (e.g. Gelman et al. [2014](#ref-Gelman2014)) and Trevor Hastie (e.g. (Hastie, Tibshirani, and Friedman [2009](#ref-Hastie2009), @Efron2016)) because both explain complicated things in a concise and understandable way.

# Prior distributions

## Introduction

## How to choose a prior

Tabelle von Fränzi (CourseIII\_glm\_glmmm/course2018/presentations\_handouts/presentations)

## Prior sensitivity

xxx

# MCMC using Stan

## Background

Markov chain Monte Carlo (MCMC) simulation techniques were developed in the mid-1950s by physicists (Metropolis et al., 1953). Later, statisticians discovered MCMC (Hastings, 1970; Geman & Geman, 1984; Tanner & Wong, 1987; Gelfand et al., 1990; Gelfand & Smith, 1990). MCMC methods make it possible to obtain posterior distributions for parameters and latent variables (unobserved variables) of complex models. In parallel, personal computer capacities increased in the 1990s and user-friendly software such as the different programs based on the programming language BUGS (Spiegelhalter et al., 2003) came out. These developments boosted the use of Bayesian data analyses, particularly in genetics and ecology.

## Install rstan

In this book we use the program [Stan](http://mc-stan.org) to draw random samples from the joint posterior distribution of the model parameters given a model, the data, prior distributions, and initial values. To do so, it uses the “no-U-turn sampler,” which is a type of Hamiltonian Monte Carlo simulation (Hoffman and Gelman [2014](#ref-Hoffman2014); Betancourt [2013](#ref-Betancourt2013_b)), and optimization-based point estimation. These algorithms are more efficient than the ones implemented in BUGS programs and they can handle larger data sets. Stan works particularly well for hierar- chical models (Betancourt and Girolami [2013](#ref-Betancourt2013)). Stan runs on Windows, Mac, and Linux and can be used via the R interface rstan. Stan is automatically installed when the R package rstan is installed. For [installing rstan](https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started), it is advised to follow closely the system-specific instructions.

## Writing a Stan model

The statistical model is written in the Stan language and saved in a text file. The Stan language is rather strict, forcing the user to write unambiguous models. Stan is very well documented and the [Stan Documentation](http://mc-stan.org/users/documentation/index.html) contains a comprehensive Language Manual, a Wiki documentation and various tutorials.

We here provide a normal regression with one predictor variable as a worked example. The entire Stan model is as following (saved as linreg.stan)

data {  
 int<lower=0> n;  
 vector[n] y;  
 vector[n] x;  
}  
  
parameters {  
 vector[2] beta;  
 real<lower=0> sigma;  
}  
  
model {  
 //priors  
 beta ~ normal(0,5);  
 sigma ~ cauchy(0,5);  
 // likelihood  
 y ~ normal(beta[1] + beta[2] \* x, sigma);  
}

A Stan model consists of different named blocks. These blocks are (from first to last): data, transformed data, parameters, trans- formed parameters, model, and generated quantities. The blocks must appear in this order. The model block is mandatory; all other blocks are optional.

In the *data* block, the type, dimension, and name of every variable has to be declared. Optionally, the range of possible values can be specified. For example, vector[N] y; means that y is a vector (type real) of length N, and int<lower=0> N; means that N is an integer with nonnegative values (the bounds, here 0, are included). Note that the restriction to a possible range of values is not strictly necessary but this will help specifying the correct model and it will improve speed. We also see that each line needs to be closed by a column sign. In the parameters block, all model parameters have to be defined. The coefficients of the linear predictor constitute a vector of length 2, vector[2] beta;. Alternatively, real beta[2]; could be used. The sigma parameter is a one-number parameter that has to be positive, therefore real<lower=0> sigma;.

The *model* block contains the model specification. Stan functions can handle vectors and we do not have to loop over all observations as typical for BUGS . Here, we use a [Cauchy distribution](#cauchydistri) as a prior distribution for sigma. This distribution can have negative values, but because we defined the lower limit of sigma to be 0 in the parameters block, the prior distribution actually used in the model is a truncated Cauchy distribution (truncated at zero). In Chapter 11.2 we explain how to choose prior distributions.

Further characteristics of the Stan language that are good to know include: The variance parameter for the normal distribution is specified as the standard deviation (like in R but different from BUGS, where the precision is used). If no prior is specified, Stan uses a uniform prior over the range of possible values as specified in the parameter block. Variable names must not contain periods, for example, x.z would not be allowed, but x\_z is allowed. To comment out a line, use double forward-slashes //.

## Run Stan from R

We fit the model to simulated data. Stan needs a vector containing the names of the data objects. In our case, x, y, and N are objects that exist in the R console.

The function stan() starts Stan and returns an object containing MCMCs for every model parameter. We have to specify the name of the file that contains the model specification, the data, the number of chains, and the number of iterations per chain we would like to have. The first half of the iterations of each chain is declared as the warm-up. During the warm-up, Stan is not simulating a Markov chain, because in every step the algorithm is adapted. After the warm-up the algorithm is fixed and Stan simulates Markov chains.

library(rstan)  
  
# Simulate fake data   
n <- 50 # sample size  
sigma <- 5 # standard deviation of the residuals  
b0 <- 2 # intercept  
b1 <- 0.7 # slope  
  
x <- runif(n, 10, 30) # random numbers of the covariate  
simresid <- rnorm(n, 0, sd=sigma) # residuals  
  
y <- b0 + b1\*x + simresid # calculate y, i.e. the data  
  
# Bundle data into a list   
datax <- list(n=length(y), y=y, x=x)  
  
# Run STAN  
fit <- stan(file = "stanmodels/linreg.stan", data=datax, verbose = FALSE)

##   
## SAMPLING FOR MODEL 'linreg' NOW (CHAIN 1).  
## Chain 1:   
## Chain 1: Gradient evaluation took 2e-05 seconds  
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.2 seconds.  
## Chain 1: Adjust your expectations accordingly!  
## Chain 1:   
## Chain 1:   
## Chain 1: Iteration: 1 / 2000 [ 0%] (Warmup)  
## Chain 1: Iteration: 200 / 2000 [ 10%] (Warmup)  
## Chain 1: Iteration: 400 / 2000 [ 20%] (Warmup)  
## Chain 1: Iteration: 600 / 2000 [ 30%] (Warmup)  
## Chain 1: Iteration: 800 / 2000 [ 40%] (Warmup)  
## Chain 1: Iteration: 1000 / 2000 [ 50%] (Warmup)  
## Chain 1: Iteration: 1001 / 2000 [ 50%] (Sampling)  
## Chain 1: Iteration: 1200 / 2000 [ 60%] (Sampling)  
## Chain 1: Iteration: 1400 / 2000 [ 70%] (Sampling)  
## Chain 1: Iteration: 1600 / 2000 [ 80%] (Sampling)  
## Chain 1: Iteration: 1800 / 2000 [ 90%] (Sampling)  
## Chain 1: Iteration: 2000 / 2000 [100%] (Sampling)  
## Chain 1:   
## Chain 1: Elapsed Time: 0.074881 seconds (Warm-up)  
## Chain 1: 0.06265 seconds (Sampling)  
## Chain 1: 0.137531 seconds (Total)  
## Chain 1:   
##   
## SAMPLING FOR MODEL 'linreg' NOW (CHAIN 2).  
## Chain 2:   
## Chain 2: Gradient evaluation took 7e-06 seconds  
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.07 seconds.  
## Chain 2: Adjust your expectations accordingly!  
## Chain 2:   
## Chain 2:   
## Chain 2: Iteration: 1 / 2000 [ 0%] (Warmup)  
## Chain 2: Iteration: 200 / 2000 [ 10%] (Warmup)  
## Chain 2: Iteration: 400 / 2000 [ 20%] (Warmup)  
## Chain 2: Iteration: 600 / 2000 [ 30%] (Warmup)  
## Chain 2: Iteration: 800 / 2000 [ 40%] (Warmup)  
## Chain 2: Iteration: 1000 / 2000 [ 50%] (Warmup)  
## Chain 2: Iteration: 1001 / 2000 [ 50%] (Sampling)  
## Chain 2: Iteration: 1200 / 2000 [ 60%] (Sampling)  
## Chain 2: Iteration: 1400 / 2000 [ 70%] (Sampling)  
## Chain 2: Iteration: 1600 / 2000 [ 80%] (Sampling)  
## Chain 2: Iteration: 1800 / 2000 [ 90%] (Sampling)  
## Chain 2: Iteration: 2000 / 2000 [100%] (Sampling)  
## Chain 2:   
## Chain 2: Elapsed Time: 0.075796 seconds (Warm-up)  
## Chain 2: 0.066709 seconds (Sampling)  
## Chain 2: 0.142505 seconds (Total)  
## Chain 2:   
##   
## SAMPLING FOR MODEL 'linreg' NOW (CHAIN 3).  
## Chain 3:   
## Chain 3: Gradient evaluation took 7e-06 seconds  
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.07 seconds.  
## Chain 3: Adjust your expectations accordingly!  
## Chain 3:   
## Chain 3:   
## Chain 3: Iteration: 1 / 2000 [ 0%] (Warmup)  
## Chain 3: Iteration: 200 / 2000 [ 10%] (Warmup)  
## Chain 3: Iteration: 400 / 2000 [ 20%] (Warmup)  
## Chain 3: Iteration: 600 / 2000 [ 30%] (Warmup)  
## Chain 3: Iteration: 800 / 2000 [ 40%] (Warmup)  
## Chain 3: Iteration: 1000 / 2000 [ 50%] (Warmup)  
## Chain 3: Iteration: 1001 / 2000 [ 50%] (Sampling)  
## Chain 3: Iteration: 1200 / 2000 [ 60%] (Sampling)  
## Chain 3: Iteration: 1400 / 2000 [ 70%] (Sampling)  
## Chain 3: Iteration: 1600 / 2000 [ 80%] (Sampling)  
## Chain 3: Iteration: 1800 / 2000 [ 90%] (Sampling)  
## Chain 3: Iteration: 2000 / 2000 [100%] (Sampling)  
## Chain 3:   
## Chain 3: Elapsed Time: 0.070117 seconds (Warm-up)  
## Chain 3: 0.059636 seconds (Sampling)  
## Chain 3: 0.129753 seconds (Total)  
## Chain 3:   
##   
## SAMPLING FOR MODEL 'linreg' NOW (CHAIN 4).  
## Chain 4:   
## Chain 4: Gradient evaluation took 7e-06 seconds  
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.07 seconds.  
## Chain 4: Adjust your expectations accordingly!  
## Chain 4:   
## Chain 4:   
## Chain 4: Iteration: 1 / 2000 [ 0%] (Warmup)  
## Chain 4: Iteration: 200 / 2000 [ 10%] (Warmup)  
## Chain 4: Iteration: 400 / 2000 [ 20%] (Warmup)  
## Chain 4: Iteration: 600 / 2000 [ 30%] (Warmup)  
## Chain 4: Iteration: 800 / 2000 [ 40%] (Warmup)  
## Chain 4: Iteration: 1000 / 2000 [ 50%] (Warmup)  
## Chain 4: Iteration: 1001 / 2000 [ 50%] (Sampling)  
## Chain 4: Iteration: 1200 / 2000 [ 60%] (Sampling)  
## Chain 4: Iteration: 1400 / 2000 [ 70%] (Sampling)  
## Chain 4: Iteration: 1600 / 2000 [ 80%] (Sampling)  
## Chain 4: Iteration: 1800 / 2000 [ 90%] (Sampling)  
## Chain 4: Iteration: 2000 / 2000 [100%] (Sampling)  
## Chain 4:   
## Chain 4: Elapsed Time: 0.080744 seconds (Warm-up)  
## Chain 4: 0.060983 seconds (Sampling)  
## Chain 4: 0.141727 seconds (Total)  
## Chain 4:

## Further reading

* [Stan-Homepage](http://mc-stan.org): It contains the documentation for Stand a a lot of tutorials.

# Ridge regression

We should provide an example in Stan.

## Introduction

# Settings   
library(R2OpenBUGS)  
bugslocation <- "C:/Program Files/OpenBUGS323/OpenBugs.exe" # location of OpenBUGS  
bugsworkingdir <- file.path(getwd(), "BUGS") # Bugs working directory  
  
#-------------------------------------------------------------------------------  
# Simulate fake data   
#-------------------------------------------------------------------------------  
library(MASS)  
n <- 50 # sample size  
b0 <- 1.2  
b <- rnorm(5, 0, 2)  
Sigma <- matrix(c(10,3,3,2,1,  
 3,2,3,2,1,  
 3,3,5,3,2,  
 2,2,3,10,3,  
 1,1,2,3,15),5,5)  
Sigma  
x <- mvrnorm(n = n, rep(0, 5), Sigma)  
  
simresid <- rnorm(n, 0, sd=3) # residuals  
  
x.z <- x  
for(i in 1:ncol(x)) x.z[,i] <- (x[,i]-mean(x[,i]))/sd(x[,i])   
  
y <- b0 + x.z%\*%b + simresid # calculate y, i.e. the data  
  
  
  
#-------------------------------------------------------------------------------  
# Function to generate initial values   
#-------------------------------------------------------------------------------  
inits <- function() {  
 list(b0=runif(1, -2, 2),  
 b=runif(5, -2, 2),  
 sigma=runif(1, 0.1, 2))  
}  
  
  
#-------------------------------------------------------------------------------  
# Run OpenBUGS  
#-------------------------------------------------------------------------------  
parameters <- c("b0", "b", "sigma")  
  
lambda <- c(1, 2, 10, 25, 50, 100, 500, 1000, 10000)  
bs <- matrix(ncol=length(lambda), nrow=length(b))  
bse <- matrix(ncol=length(lambda), nrow=length(b))  
  
for(j in 1:length(lambda)){  
datax <- list(y=as.numeric(y), x=x, n=n, mb=rep(0, 5), lambda=lambda[j])  
  
  
fit <- bugs(datax, inits, parameters, model.file="ridge\_regression.txt",  
 n.thin=1, n.chains=2, n.burnin=5000, n.iter=10000,  
 debug=FALSE, OpenBUGS.pgm = bugslocation,   
 working.directory=bugsworkingdir)  
  
bs[,j] <- fit$mean$b  
bse[,j] <- fit$sd$b  
  
}  
  
range(bs)  
plot(1:length(lambda), seq(-2, 1, length=length(lambda)), type="n")  
colkey <- rainbow(length(b))  
for(j in 1:nrow(bs)){  
 lines(1:length(lambda), bs[j,], col=colkey[j], lwd=2)  
 lines(1:length(lambda), bs[j,]-2\*bse[j,], col=colkey[j], lty=3)  
 lines(1:length(lambda), bs[j,]+2\*bse[j,], col=colkey[j], lty=3)  
}  
abline(h=0)  
round(fit$summary,2)  
  
#-------------------------------------------------------------------------------  
# Run WinBUGS  
#-------------------------------------------------------------------------------  
library(R2WinBUGS)  
bugsdir <- "C:/Users/fk/WinBUGS14" #   
mod <- bugs(datax, inits= inits, parameters,  
 model.file="normlinreg.txt", n.chains=2, n.iter=1000,   
 n.burnin=500, n.thin=1, debug=TRUE,  
 bugs.directory=bugsdir, program="WinBUGS", working.directory=bugsworkingdir)  
  
  
#-------------------------------------------------------------------------------  
# Test convergence and make inference  
#-------------------------------------------------------------------------------  
library(blmeco)  
  
# Make Figure 12.2  
par(mfrow=c(3,1))   
historyplot(fit, "beta0")  
historyplot(fit, "beta1")  
historyplot(fit, "sigmaRes")  
  
# Parameter estimates  
print(fit$summary, 3)  
  
# Make predictions for covariate values between 10 and 30  
newdat <- data.frame(x=seq(10, 30, length=100))  
Xmat <- model.matrix(~x, data=newdat)  
predmat <- matrix(ncol=fit$n.sim, nrow=nrow(newdat))  
for(i in 1:fit$n.sim) predmat[,i] <- Xmat%\*%c(fit$sims.list$beta0[i], fit$sims.list$beta1[i])  
newdat$lower.bugs <- apply(predmat, 1, quantile, prob=0.025)  
newdat$upper.bugs <- apply(predmat, 1, quantile, prob=0.975)  
  
plot(y~x, pch=16, las=1, cex.lab=1.4, cex.axis=1.2, type="n", main="")  
polygon(c(newdat$x, rev(newdat$x)), c(newdat$lower.bugs, rev(newdat$upper.bugs)), col=grey(0.7), border=NA)  
abline(c(fit$mean$beta0, fit$mean$beta1), lwd=2)  
box()  
points(x,y)

# Structural equation model

We should provide an example in Stan.

## Introduction

------------------------------------------------------------------------------------------------------  
# General settings  
#------------------------------------------------------------------------------------------------------  
library(MASS)  
library(rjags)  
library(MCMCpack)  
  
#------------------------------------------------------------------------------------------------------  
# Simulation  
#------------------------------------------------------------------------------------------------------  
n <- 100  
heffM <- 0.6 # effect of H on M  
heffCS <- 0.0 # effect of H on Clutch size  
meffCS <- 0.6 # effect of M on Clutch size  
  
SigmaM <- matrix(c(0.1,0.04,0.04,0.1),2,2)  
meffm1 <- 0.6  
meffm2 <- 0.7  
  
SigmaH <- matrix(c(0.1,0.04,0.04,0.1),2,2)  
meffh1 <- 0.6  
meffh2 <- -0.7  
  
# Latente Variablen  
H <- rnorm(n, 0, 1)  
M <- rnorm(n, heffM \* H, 0.1)  
  
# Clutch size  
CS <- rnorm(n, heffCS \* H + meffCS \* M, 0.1)  
  
# Indicators  
eM <- cbind(meffm1 \* M, meffm2 \* M)  
datM <- matrix(NA, ncol = 2, nrow = n)  
eH <- cbind(meffh1 \* H, meffh2 \* H)  
datH <- matrix(NA, ncol = 2, nrow = n)  
for(i in 1:n) {  
 datM[i,] <- mvrnorm(1, eM[i,], SigmaM)  
 datH[i,] <- mvrnorm(1, eH[i,], SigmaH)  
}  
  
#------------------------------------------------------------------------------  
# JAGS Model   
#------------------------------------------------------------------------------  
dat <- list(datM = datM, datH = datH, n = n, CS = CS, #H = H, M = M,   
 S3 = matrix(c(1,0,0,1),nrow=2)/1)  
  
# Function to create initial values  
inits <- function() {  
 list(  
 meffh = runif(2, 0, 0.1),  
 meffm = runif(2, 0, 0.1),  
 heffM = runif(1, 0, 0.1),  
 heffCS = runif(1, 0, 0.1),  
 meffCS = runif(1, 0, 0.1),  
 tauCS = runif(1, 0.1, 0.3),  
 tauMH = runif(1, 0.1, 0.3),  
 tauH = rwish(3,matrix(c(.02,0,0,.04),nrow=2)),  
 tauM = rwish(3,matrix(c(.02,0,0,.04),nrow=2))  
# M = as.numeric(rep(0, n))  
 )  
}  
  
t.n.thin <- 50  
t.n.chains <- 2  
t.n.burnin <- 20000  
t.n.iter <- 50000  
  
# Run JAGS  
jagres <- jags.model('JAGS/BUGSmod1.R',data = dat, n.chains = t.n.chains, inits = inits, n.adapt = t.n.burnin)  
  
params <- c("meffh", "meffm", "heffM", "heffCS", "meffCS")  
mod <- coda.samples(jagres, params, n.iter=t.n.iter, thin=t.n.thin)  
res <- round(data.frame(summary(mod)$quantiles[, c(3, 1, 5)]), 3)  
res$TRUEVALUE <- c(heffCS, heffM, meffCS, meffh1, meffh2, meffm1, meffm2)  
res  
  
# Traceplots  
post <- data.frame(rbind(mod[[1]], mod[[2]]))  
names(post) <- dimnames(mod[[1]])[[2]]  
par(mfrow = c(3,3))  
param <- c("meffh[1]", "meffh[2]", "meffm[1]", "meffm[2]", "heffM", "heffCS", "meffCS")  
traceplot(mod[, match(param, names(post))])

# Introduction to PART III

**Fehler! Linkreferenz ungültig.**

This part is a collection of more complicated ecological models to analyse data that may not be analysed with the traditional linear models that we covered in [PART I](#PART-I) of this book.

## Model notations

It is unavoidable that different authors use different notations for the same thing, or that the same notation is used for different things. We try to use, whenever possible, notations that is commonly used at the [International Statistical Ecology Congress ISEC](https://http://www.isec2018.org/home). Resulting from an earlier ISEC, Betancourt et al. ([2016](#ref-Thomson2009)) give guidelines on what letter should be used for which parameter in order to achieve a standard notation at least among people working with classical mark-recapture models. However, the alphabet has fewer letters compared to the number of ecological parameters. Therefore, the same letter cannot stand for the same parameter across all papers, books and chapters. Here, we try to use the same letter for the same parameter within the same chapter.

# Zero-inflated Poisson Mixed Model

## Introduction

Usually we describe the outcome variable with a single distribution, such as the normal distribution in the case of linear (mixed) models, and Poisson or binomial distributions in the case of generalized linear (mixed) models. In life sciences, however, quite often the data are actually generated by more than one process. In such cases the distribution of the data could be the result of two or more different distributions. If we do not account for these different processes our inferences are likely to be biased. In this chapter, we introduce a mixture model that explicitly include two processes that generated the data. The zero-inflated Poisson model is a mixture of a binomial and a Poisson distribution. We belief that two (or more)-level models are very useful tools in life sciences because they can help uncover the different processes that generate the data we observe.

## Example data

We used the blackstork data from the blmeco-package. They contain the breeding success of Black-stork in Latvia. The data was collected and kindly provided by Maris Stradz. The data contains the number of nestlings of more then 300 [Black-stork](https://en.wikipedia.org/wiki/Black_stork) nests in different years.

Counting animals or plants is a typical example of data that contain a lot of zero counts. For example, the number of nestlings produced by a breeding pair is often zero because the whole nest was depredated or because a catastrophic event occurred such as a flood. However, when the nest succeeds, the number of nestlings varies among the successful nests depending on how many eggs the female has laid, how much food the parents could bring to the nest, or other factors that affect the survival of a nestling in an intact nest. Thus the factors that determine how many zero counts there are in the data differ from the factors that determine how many nestlings there are, if a nest survives. Count data that are produced by two different processes–one produces the zero counts and the other the variance in the count for the ones that were not zero in the first process–are called zero-inflated data. Histograms of zero-inflated data look bimodal, with one peak at zero (Figure 10).

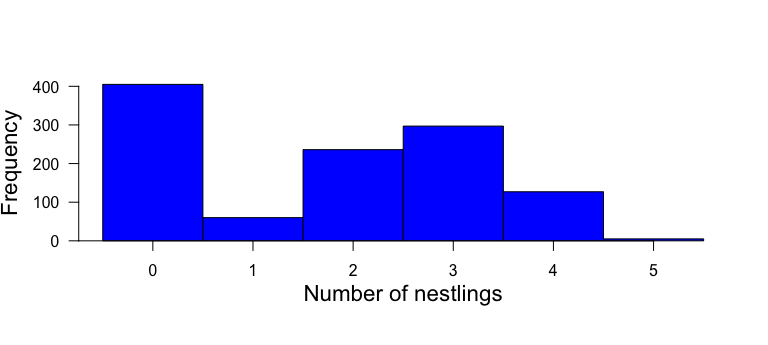


Figure 10: Histogram of the number of nestlings counted in black stork nests Ciconia nigra in Latvia (n = 1130 observations of 279 nests).

## Model

The Poisson distribution does not fit well to such data, because the data contain more zero counts than expected under the Poisson distribution. Mullahy (1986) and Lambert (1992) formulated two different types of models that combine the two processes in one model and therefore account for the zero excess in the data and allow the analysis of the two processes separately.

The hurdle model (Mullahy, 1986) combines a left-truncated count data model (Poisson or negative binomial distribution that only describes the distribution of data larger than zero) with a zero-hurdle model that describes the distribution of the data that are either zero or nonzero. In other words, the hurdle model divides the data into two data subsets, the zero counts and the nonzero counts, and fits two separate models to each subset of the data. To account for this division of the data, the two models assume left truncation (all measurements below 1 are missing in the data) and right censoring (all measurements larger than 1 have the value 1), respectively, in their error distributions. A hurdle model can be fitted in R using the function hurdle from the package pscl (Jackman, 2008). See the tutorial by Zeileis et al. (2008) for an introduction.

In contrast to the hurdle model, the zero-inflated models (Mullahy, 1986; Lambert, 1992) combine a Bernoulli model (zero vs. nonzero) with a conditional Poisson model; conditional on the Bernoulli process being nonzero. Thus this model allows for a mixture of zero counts: some zero counts are zero because the outcome of the Bernoulli process was zero (these zero counts are sometimes called structural zero values), and others are zero because their outcome from the Poisson process was zero. The function `zeroinfl from the package pscl fits zero-inflated models (Zeileis et al., 2008).

The zero-inflated model may seem to reflect the true process that has generated the data closer than the hurdle model. However, sometimes the fit of zero-inflated models is impeded because of high correlation of the model parameters between the zero model and the count model. In such cases, a hurdle model may cause less troubles.

Both functions (hurdle and zeroinfl) from the package pscl do not allow the inclusion of random factors. The functions MCMCglmm from the package MCMCglmm (Hadfield, 2010) and glmmadmb from the package glmmADMB (<http://glmmadmb.r-forge.r-project.org/>) provide the possibility to account for zero-inflation with a GLMM. However, these functions are not very flexible in the types of zero-inflated models they can fit; for example, glmmadmb only includes a constant proportion of zero values. A zero-inflation model using BUGS is described in Ke ́ry and Schaub (2012). Here we use Stan to fit a zero- inflated model. Once we understand the basic model code, it is easy to add predictors and/or random effects to both the zero and the count model.

The example data contain numbers of nestlings in black stork Ciconia nigra nests in Latvia collected by Maris Stradz and collaborators at 279 nests be- tween 1979 and 2010. Black storks build solid and large aeries on branches of large trees. The same aerie is used for up to 17 years until it collapses. The black stork population in Latvia has drastically declined over the last decades. Here, we use the nestling data as presented in Figure 14-2 to describe whether the number of black stork nestlings produced in Latvia decreased over time. We use a zero-inflated Poisson model to separately estimate temporal trends for nest survival and the number of nestlings in successful nests. Since the same nests have been measured repeatedly over 1 to 17 years, we add nest ID as a random factor to both models, the Bernoulli and the Poisson model. After the first model fit, we saw that the between-nest variance in the number of nest- lings for the successful nests was close to zero. Therefore, we decide to delete the random effect from the Poisson model. Here is our final model:

zit is a latent (unobserved) variable that takes the values 0 or 1 for each nest i during year t. It indicates a “structural zero”, that is, if zit 1⁄4 1 the number of nestlings yit always is zero, because the expected value in the Poisson model lit(1 zit) becomes zero. If zit 1⁄4 0, the expected value in the Poisson model becomes lit.

To fit this model in Stan, we first write the Stan model code and save it in a separated text-file with name “zeroinfl.stan”.

Here is a handy package: <https://cran.r-project.org/web/packages/GLMMadaptive/vignettes/ZeroInflated_and_TwoPart_Models.html>

# Daily nest survival

## Background

Analyses of nest survival is important for understanding the mechanisms of population dynamics. The life-span of a nest could be used as a measure of nest survival. However, this measure very often is biased towards nests that survived longer because these nests are detected by the ornithologists with higher probability (Mayfield [1975](#ref-Mayfield1975)). Therefore, instead of estimating overall nest survival it is better to estimate daily nest survival conditional on survival to the previous day.

## Models for estimating daily nest survival

Using a simple logistic regression with an indicator for whether a nest is alive or not at day is not appropriate for estimating nest survival because its result is directly dependent on the time point when the nest has been detected. This time point is not biologically relevant. The conditional logistic regression clogit from the survival package could be used to describe correlations of the mortality with environmental variables but it does not provide an estimate for the daily survival probability. A natural model that allows estimating daily nest survival is the known-fate survival model. It is a Markov model that models the state of a nest at day (whether it is alive, or not ) as a Bernoulli variable dependent on the state of the nest the day before.

The daily nest survival can be linearly related to predictor variables that are measured on the nest or on the day level.

It is also possible to add random effects if needed.

## The Stan model

The following Stan model code is saved as daily\_nest\_survival.stan.

data {  
 int<lower=0> Nnests; // number of nests  
 int<lower=0> last[Nnests]; // day of last observation (alive or dead)  
 int<lower=0> first[Nnests]; // day of first observation (alive or dead)  
 int<lower=0> maxage; // maximum of last  
 int<lower=0> y[Nnests, maxage]; // indicator of alive nests  
 real cover[Nnests]; // a covariate of the nest  
 real age[maxage]; // a covariate of the date  
}  
  
parameters {  
 vector[4] b; // coef of linear pred for S  
}  
  
model {  
 real S[Nnests, maxage-1]; // survival probability  
   
 for(i in 1:Nnests){   
 for(t in first[i]:(last[i]-1)){   
 S[i,t] = inv\_logit(b[1] + b[2]\*cover[i] + b[3]\*age[t] + b[4]\*pow(age[t], 2));   
 }  
 }  
  
 // priors  
 b[1]~normal(0,5);  
 b[2]~normal(0,3);  
 b[3]~normal(0,3);   
 b[4]~normal(0,3);  
  
 // likelihood  
 for (i in 1:Nnests) {  
 for(t in (first[i]+1):last[i]){  
 y[i,t]~bernoulli(y[i,t-1]\*S[i,t-1]);  
 }  
 }  
}

## Prepare data and run Stan

Read data DESCRIPTION OF ORIGIN INCL REFERENCE MISSING

load("RData/nest\_surv\_data.rda")  
str(datax)

## List of 7  
## $ y : int [1:157, 1:38] 1 NA 1 NA 1 NA NA 1 1 1 ...  
## $ Nnests: int 157  
## $ last : int [1:157] 26 30 32 27 31 30 31 31 33 31 ...  
## $ first : int [1:157] 1 14 1 3 1 24 18 1 1 1 ...  
## $ cover : num [1:157] -0.944 -0.226 0.133 0.133 -0.226 ...  
## $ age : num [1:38] -1.66 -1.57 -1.48 -1.39 -1.3 ...  
## $ maxage: int 38

datax$y[is.na(datax$y)] <- 0 # Stan does not allow for NA's in the outcome

# Run STAN  
library(rstan)  
  
mod <- stan(file = "stanmodels/daily\_nest\_survival.stan", data=datax,   
 chains=5, iter=2500, control=list(adapt\_delta=0.9), verbose = FALSE)

##   
## SAMPLING FOR MODEL 'daily\_nest\_survival' NOW (CHAIN 1).  
## Chain 1:   
## Chain 1: Gradient evaluation took 0.001052 seconds  
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 10.52 seconds.  
## Chain 1: Adjust your expectations accordingly!  
## Chain 1:   
## Chain 1:   
## Chain 1: Iteration: 1 / 2500 [ 0%] (Warmup)  
## Chain 1: Iteration: 250 / 2500 [ 10%] (Warmup)  
## Chain 1: Iteration: 500 / 2500 [ 20%] (Warmup)  
## Chain 1: Iteration: 750 / 2500 [ 30%] (Warmup)  
## Chain 1: Iteration: 1000 / 2500 [ 40%] (Warmup)  
## Chain 1: Iteration: 1250 / 2500 [ 50%] (Warmup)  
## Chain 1: Iteration: 1251 / 2500 [ 50%] (Sampling)  
## Chain 1: Iteration: 1500 / 2500 [ 60%] (Sampling)  
## Chain 1: Iteration: 1750 / 2500 [ 70%] (Sampling)  
## Chain 1: Iteration: 2000 / 2500 [ 80%] (Sampling)  
## Chain 1: Iteration: 2250 / 2500 [ 90%] (Sampling)  
## Chain 1: Iteration: 2500 / 2500 [100%] (Sampling)  
## Chain 1:   
## Chain 1: Elapsed Time: 5.3314 seconds (Warm-up)  
## Chain 1: 5.24442 seconds (Sampling)  
## Chain 1: 10.5758 seconds (Total)  
## Chain 1:   
##   
## SAMPLING FOR MODEL 'daily\_nest\_survival' NOW (CHAIN 2).  
## Chain 2:   
## Chain 2: Gradient evaluation took 0.000459 seconds  
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 4.59 seconds.  
## Chain 2: Adjust your expectations accordingly!  
## Chain 2:   
## Chain 2:   
## Chain 2: Iteration: 1 / 2500 [ 0%] (Warmup)  
## Chain 2: Iteration: 250 / 2500 [ 10%] (Warmup)  
## Chain 2: Iteration: 500 / 2500 [ 20%] (Warmup)  
## Chain 2: Iteration: 750 / 2500 [ 30%] (Warmup)  
## Chain 2: Iteration: 1000 / 2500 [ 40%] (Warmup)  
## Chain 2: Iteration: 1250 / 2500 [ 50%] (Warmup)  
## Chain 2: Iteration: 1251 / 2500 [ 50%] (Sampling)  
## Chain 2: Iteration: 1500 / 2500 [ 60%] (Sampling)  
## Chain 2: Iteration: 1750 / 2500 [ 70%] (Sampling)  
## Chain 2: Iteration: 2000 / 2500 [ 80%] (Sampling)  
## Chain 2: Iteration: 2250 / 2500 [ 90%] (Sampling)  
## Chain 2: Iteration: 2500 / 2500 [100%] (Sampling)  
## Chain 2:   
## Chain 2: Elapsed Time: 5.03361 seconds (Warm-up)  
## Chain 2: 5.414 seconds (Sampling)  
## Chain 2: 10.4476 seconds (Total)  
## Chain 2:   
##   
## SAMPLING FOR MODEL 'daily\_nest\_survival' NOW (CHAIN 3).  
## Chain 3:   
## Chain 3: Gradient evaluation took 0.000565 seconds  
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 5.65 seconds.  
## Chain 3: Adjust your expectations accordingly!  
## Chain 3:   
## Chain 3:   
## Chain 3: Iteration: 1 / 2500 [ 0%] (Warmup)  
## Chain 3: Iteration: 250 / 2500 [ 10%] (Warmup)  
## Chain 3: Iteration: 500 / 2500 [ 20%] (Warmup)  
## Chain 3: Iteration: 750 / 2500 [ 30%] (Warmup)  
## Chain 3: Iteration: 1000 / 2500 [ 40%] (Warmup)  
## Chain 3: Iteration: 1250 / 2500 [ 50%] (Warmup)  
## Chain 3: Iteration: 1251 / 2500 [ 50%] (Sampling)  
## Chain 3: Iteration: 1500 / 2500 [ 60%] (Sampling)  
## Chain 3: Iteration: 1750 / 2500 [ 70%] (Sampling)  
## Chain 3: Iteration: 2000 / 2500 [ 80%] (Sampling)  
## Chain 3: Iteration: 2250 / 2500 [ 90%] (Sampling)  
## Chain 3: Iteration: 2500 / 2500 [100%] (Sampling)  
## Chain 3:   
## Chain 3: Elapsed Time: 5.18304 seconds (Warm-up)  
## Chain 3: 5.00018 seconds (Sampling)  
## Chain 3: 10.1832 seconds (Total)  
## Chain 3:   
##   
## SAMPLING FOR MODEL 'daily\_nest\_survival' NOW (CHAIN 4).  
## Chain 4:   
## Chain 4: Gradient evaluation took 0.000484 seconds  
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 4.84 seconds.  
## Chain 4: Adjust your expectations accordingly!  
## Chain 4:   
## Chain 4:   
## Chain 4: Iteration: 1 / 2500 [ 0%] (Warmup)  
## Chain 4: Iteration: 250 / 2500 [ 10%] (Warmup)  
## Chain 4: Iteration: 500 / 2500 [ 20%] (Warmup)  
## Chain 4: Iteration: 750 / 2500 [ 30%] (Warmup)  
## Chain 4: Iteration: 1000 / 2500 [ 40%] (Warmup)  
## Chain 4: Iteration: 1250 / 2500 [ 50%] (Warmup)  
## Chain 4: Iteration: 1251 / 2500 [ 50%] (Sampling)  
## Chain 4: Iteration: 1500 / 2500 [ 60%] (Sampling)  
## Chain 4: Iteration: 1750 / 2500 [ 70%] (Sampling)  
## Chain 4: Iteration: 2000 / 2500 [ 80%] (Sampling)  
## Chain 4: Iteration: 2250 / 2500 [ 90%] (Sampling)  
## Chain 4: Iteration: 2500 / 2500 [100%] (Sampling)  
## Chain 4:   
## Chain 4: Elapsed Time: 4.88374 seconds (Warm-up)  
## Chain 4: 5.05651 seconds (Sampling)  
## Chain 4: 9.94025 seconds (Total)  
## Chain 4:   
##   
## SAMPLING FOR MODEL 'daily\_nest\_survival' NOW (CHAIN 5).  
## Chain 5:   
## Chain 5: Gradient evaluation took 0.000492 seconds  
## Chain 5: 1000 transitions using 10 leapfrog steps per transition would take 4.92 seconds.  
## Chain 5: Adjust your expectations accordingly!  
## Chain 5:   
## Chain 5:   
## Chain 5: Iteration: 1 / 2500 [ 0%] (Warmup)  
## Chain 5: Iteration: 250 / 2500 [ 10%] (Warmup)  
## Chain 5: Iteration: 500 / 2500 [ 20%] (Warmup)  
## Chain 5: Iteration: 750 / 2500 [ 30%] (Warmup)  
## Chain 5: Iteration: 1000 / 2500 [ 40%] (Warmup)  
## Chain 5: Iteration: 1250 / 2500 [ 50%] (Warmup)  
## Chain 5: Iteration: 1251 / 2500 [ 50%] (Sampling)  
## Chain 5: Iteration: 1500 / 2500 [ 60%] (Sampling)  
## Chain 5: Iteration: 1750 / 2500 [ 70%] (Sampling)  
## Chain 5: Iteration: 2000 / 2500 [ 80%] (Sampling)  
## Chain 5: Iteration: 2250 / 2500 [ 90%] (Sampling)  
## Chain 5: Iteration: 2500 / 2500 [100%] (Sampling)  
## Chain 5:   
## Chain 5: Elapsed Time: 5.07621 seconds (Warm-up)  
## Chain 5: 5.0219 seconds (Sampling)  
## Chain 5: 10.0981 seconds (Total)  
## Chain 5:

## Check convergence

We love exploring the performance of the Markov chains by using the function launch\_shinystan from the package shinystan.

## Look at results

It looks like cover does not affect daily nest survival, but daily nest survival decreases with the age of the nestlings.

print(mod)

## Inference for Stan model: daily\_nest\_survival.  
## 5 chains, each with iter=2500; warmup=1250; thin=1;   
## post-warmup draws per chain=1250, total post-warmup draws=6250.  
##   
## mean se\_mean sd 2.5% 25% 50% 75% 97.5% n\_eff  
## b[1] 3.89 0.00 0.18 3.54 3.76 3.88 4.01 4.25 3061  
## b[2] 0.00 0.00 0.13 -0.25 -0.09 0.00 0.08 0.25 5873  
## b[3] -0.90 0.00 0.19 -1.27 -1.02 -0.90 -0.78 -0.54 3988  
## b[4] -0.19 0.00 0.23 -0.62 -0.35 -0.20 -0.04 0.27 3257  
## lp\_\_ -306.85 0.03 1.41 -310.41 -307.56 -306.53 -305.81 -305.09 2853  
## Rhat  
## b[1] 1  
## b[2] 1  
## b[3] 1  
## b[4] 1  
## lp\_\_ 1  
##   
## Samples were drawn using NUTS(diag\_e) at Mon Aug 5 09:55:26 2019.  
## For each parameter, n\_eff is a crude measure of effective sample size,  
## and Rhat is the potential scale reduction factor on split chains (at   
## convergence, Rhat=1).

# effect plot  
bsim <- as.data.frame(mod)  
nsim <- nrow(bsim)  
  
newdat <- data.frame(age=seq(1, datax$maxage, length=100))  
newdat$age.z <- (newdat$age-mean(1:datax$maxage))/sd((1:datax$maxage))  
Xmat <- model.matrix(~age.z+I(age.z^2), data=newdat)  
fitmat <- matrix(ncol=nsim, nrow=nrow(newdat))  
for(i in 1:nsim) fitmat[,i] <- plogis(Xmat%\*%as.numeric(bsim[i,c(1,3,4)]))  
newdat$fit <- apply(fitmat, 1, median)  
newdat$lwr <- apply(fitmat, 1, quantile, prob=0.025)  
newdat$upr <- apply(fitmat, 1, quantile, prob=0.975)  
  
plot(newdat$age, newdat$fit, ylim=c(0.8,1), type="l",  
 las=1, ylab="Daily nest survival", xlab="Age [d]")  
lines(newdat$age, newdat$lwr, lty=3)  
lines(newdat$age, newdat$upr, lty=3)

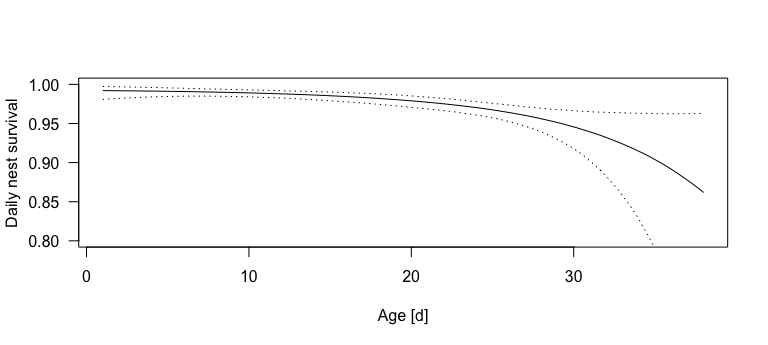


Figure 11: Estimated daily nest survival probability in relation to nest age. Dotted lines are 95% uncertainty intervals of the regression line.

## Further reading

ADD EXAMPLE STUDIES!!

# Capture-mark recapture model with a mixture structure to account for missing sex-variable for parts of the individuals

## Introduction

In some species the identification of the sex is not possible for all individuals without sampling DNA. For example, morphological dimorphism is absent or so weak that parts of the individuals cannot be assigned to one of the sexes. Particularly in ornithological long-term capture recapture data sets that typically are obtained by voluntary bird ringers who do normaly not have the possibilities to analyse DNA, often the sex identification is missing in parts of the individuals. For estimating survival, it would nevertheless be valuable to include data of all individuals, use the information on sex-specific effects on survival wherever possible but account for the fact that of parts of the individuals the sex is not known. We here explain how a Cormack-Jolly-Seber model can be integrated with a mixture model in oder to allow for a combined analyses of individuals with and without sex identified. An introduction to the Cormack-Jolly-Seber model we gave in Chapter 14.5 of the book Korner-Nievergelt et al. ([2015](#ref-KornerNievergelt2015)). We here expand this model by a mixture structure that allows including individuals with a missing categorical predictor variable, such as sex.

## Data description

## simulate data  
# true parameter values  
theta <- 0.6 # proportion of males  
nocc <- 15 # number of years in the data set  
b0 <- matrix(NA, ncol=nocc-1, nrow=2)  
b0[1,] <- rbeta((nocc-1), 3, 4) # capture probability of males  
b0[2,] <- rbeta((nocc-1), 2, 4) # capture probability of females   
a0 <- matrix(NA, ncol=2, nrow=2)  
a1 <- matrix(NA, ncol=2, nrow=2)  
a0[1,1]<- qlogis(0.7) # average annual survival for adult males  
a0[1,2]<- qlogis(0.3) # average annual survival for juveniles  
a0[2,1] <- qlogis(0.55) # average annual survival for adult females  
a0[2,2] <- a0[1,2]  
a1[1,1] <- 0  
a1[1,2] <- -0.5  
a1[2,1] <- -0.8  
a1[2,2] <- a1[1,2]  
  
nindi <- 1000 # number of individuals with identified sex  
nindni <- 1500 # number of individuals with non-identified sex  
nind <- nindi + nindni # total number of individuals  
y <- matrix(ncol=nocc, nrow=nind)  
z <- matrix(ncol=nocc, nrow=nind)  
first <- sample(1:(nocc-1), nind, replace=TRUE)  
sex <- sample(c(1,2), nind, prob=c(theta, 1-theta), replace=TRUE)  
juvfirst <- sample(c(0,1), nind, prob=c(0.5, 0.5), replace=TRUE)  
juv <- matrix(0, nrow=nind, ncol=nocc)  
for(i in 1:nind) juv[i,first[i]] <- juv[i]  
  
x <- runif(nocc-1, -2, 2) # a time dependent covariate covariate   
p <- b0 # recapture probability  
phi <- array(NA, dim=c(2, 2, nocc-1))  
# for ad males  
phi[1,1,] <- plogis(a0[1,1]+a1[1,1]\*x)  
# for ad females  
phi[2,1,] <- plogis(a0[2,1]+a1[2,1]\*x)  
# for juvs  
phi[1,2,] <- phi[2,2,] <- plogis(a0[2,2]+a1[2,2]\*x)  
for(i in 1:nind){  
 z[i,first[i]] <- 1  
 y[i, first[i]] <- 1  
 for(t in (first[i]+1):nocc){  
 z[i, t] <- rbinom(1, size=1, prob=z[i,t-1]\*phi[sex[i],juv[i,t-1]+1, t-1])  
 y[i, t] <- rbinom(1, size=1, prob=z[i,t]\*p[sex[i],t-1])  
 }  
}  
y[is.na(y)] <- 0

The mark-recapture data set consists of capture histories of 2500.00 individuals over 15.00 time periods. For each time period and individual the capture history matrix contains if the individual is captured during time period , or if the individual is not captured during time period . The marking time period varies between individuals from 1 to 14.00. At the marking time period, the age of the individuals was classified either as juvenile or as adult. Juveniles turn into adults after one time period, thus age is known for all individuals during all time periods after marking. For 1000.00 individuals of the 2500.00 individuals, the sex is identified, whereas for 1500.00 individuals, the sex is unknown. The example data contain one covariate that takes on one value for each time period.

# bundle the data for Stan  
i <- 1:nindi  
ni <- (nindi+1):nind  
datax <- list(yi=y[i,], nindi=nindi, sex=sex[i], nocc=nocc,   
 yni=y[ni,], nindni=nindni, firsti=first[i], firstni=first[ni],  
 juvi=juv[i,]+1, juvni=juv[ni,]+1, year=1:nocc, x=x)

## Model description

The observations , an indicator of whether individual i was recaptured during time period is modelled conditional on the latent true state of the individual birds (0 = dead or permanently emigrated, 1 = alive and at the study site) as a Bernoulli variable. The probability is the product of the probability that an alive individual is recaptured, , and the state of the bird (alive = 1, dead = 0). Thus, a dead bird cannot be recaptured, whereas for a bird alive during time period , the recapture probability equals :

The latent state variable is a Markovian variable with the state at time being dependent on the state at time and the apparent survival probability

:

We use the term apparent survival in order to indicate that the parameter is a product of site fidelity and survival. Thus, individuals that permanently emigrated from the study area cannot be distinguished from dead individuals. In both models, the parameters and were modelled as sex-specific. However, for parts of the individuals, sex could not be identified, i.e. sex was missing. Ignoring these missing values would most likely lead to a bias because they were not missing at random. The probability that sex can be identified is increasing with age and most likely differs between sexes. Therefore, we included a mixture model for the sex:

where is a vector of length 2, containing the probability of being a male and a female, respectively. In this way, the sex of the non-identified individuals was assumed to be male or female with probability and , respectively. This model corresponds to the finite mixture model introduced by Pledger, Pollock, and Norris ([2003](#ref-Pledger2003)) in order to account for unknown classes of birds (heterogeneity). However, in our case, for parts of the individuals the class (sex) was known.

In the example model, we constrain apparent survival to be linearly dependent on a covariate x with different slopes for males, females and juveniles using the logit link function.

Annual recapture probability was modelled for each year and age and sex class independently:

Uniform prior distributions were used for all parameters with a parameter space limited to values between 0 and 1 (probabilities) and a normal distribution with a mean of 0 and a standard deviation of 1.5 for the intercept , and a standard deviation of 5 was used for .

## The Stan code

The trick for coding the CMR-mixture model in Stan is to formulate the model 3 times:  
1. For the individuals with identified sex  
2. For the males that were not identified  
3. For the females that were not identified

Then for the non-identified individuals a mixture model is formulated that assigns a probability of being a female or a male to each individual.

data {  
 int<lower=2> nocc; // number of capture events  
 int<lower=0> nindi; // number of individuals with identified sex  
 int<lower=0> nindni; // number of individuals with non-identified sex  
 int<lower=0,upper=2> yi[nindi,nocc]; // CH[i,k]: individual i captured at k  
 int<lower=0,upper=nocc-1> firsti[nindi]; // year of first capture  
 int<lower=0,upper=2> yni[nindni,nocc]; // CH[i,k]: individual i captured at k  
 int<lower=0,upper=nocc-1> firstni[nindni]; // year of first capture  
 int<lower=1, upper=2> sex[nindi];  
 int<lower=1, upper=2> juvi[nindi, nocc];  
 int<lower=1, upper=2> juvni[nindni, nocc];  
 int<lower=1> year[nocc];  
 real x[nocc-1]; // a covariate   
}  
  
transformed data {  
 int<lower=0,upper=nocc+1> lasti[nindi]; // last[i]: ind i last capture  
 int<lower=0,upper=nocc+1> lastni[nindni]; // last[i]: ind i last capture  
 lasti = rep\_array(0,nindi);   
 lastni = rep\_array(0,nindni);  
 for (i in 1:nindi) {  
 for (k in firsti[i]:nocc) {  
 if (yi[i,k] == 1) {  
 if (k > lasti[i]) lasti[i] = k;  
 }  
 }  
 }  
 for (ii in 1:nindni) {  
 for (kk in firstni[ii]:nocc) {  
 if (yni[ii,kk] == 1) {  
 if (kk > lastni[ii]) lastni[ii] = kk;  
 }  
 }  
 }  
  
}  
  
  
parameters {  
 real<lower=0, upper=1> theta[nindni]; // probability of being male for non-identified individuals  
 real<lower=0, upper=1> b0[2,nocc-1]; // intercept of p  
 real a0[2,2]; // intercept for phi   
 real a1[2,2]; // coefficient for phi   
}  
  
transformed parameters {  
 real<lower=0,upper=1>p\_male[nindni,nocc]; // capture probability  
 real<lower=0,upper=1>p\_female[nindni,nocc]; // capture probability  
 real<lower=0,upper=1>p[nindi,nocc]; // capture probability  
  
 real<lower=0,upper=1>phi\_male[nindni,nocc-1]; // survival probability  
 real<lower=0,upper=1>chi\_male[nindni,nocc+1]; // probability that an individual   
 // is never recaptured after its  
 // last capture  
 real<lower=0,upper=1>phi\_female[nindni,nocc-1]; // survival probability  
 real<lower=0,upper=1>chi\_female[nindni,nocc+1]; // probability that an individual   
 // is never recaptured after its  
 // last capture  
 real<lower=0,upper=1>phi[nindi,nocc-1]; // survival probability  
 real<lower=0,upper=1>chi[nindi,nocc+1]; // probability that an individual   
 // is never recaptured after its  
 // last capture  
  
 {  
 int k;   
 int kk;   
 for(ii in 1:nindi){  
 if (firsti[ii]>1) {  
 for (z in 1:(firsti[ii]-1)){  
 phi[ii,z] = 1;  
 }  
 }  
 for(tt in firsti[ii]:(nocc-1)) {  
 // linear predictor for phi:  
 phi[ii,tt] = inv\_logit(a0[sex[ii], juvi[ii,tt]] + a1[sex[ii], juvi[ii,tt]]\*x[tt]);   
  
 }  
 }  
  
 for(ii in 1:nindni){  
 if (firstni[ii]>1) {  
 for (z in 1:(firstni[ii]-1)){  
 phi\_female[ii,z] = 1;  
 phi\_male[ii,z] = 1;  
 }  
 }  
 for(tt in firstni[ii]:(nocc-1)) {  
 // linear predictor for phi:  
 phi\_male[ii,tt] = inv\_logit(a0[1, juvni[ii,tt]] + a1[1, juvni[ii,tt]]\*x[tt]);   
 phi\_female[ii,tt] = inv\_logit(a0[2, juvni[ii,tt]]+ a1[2, juvni[ii,tt]]\*x[tt]);  
  
 }  
 }  
   
 for(i in 1:nindi) {  
 // linear predictor for p for identified individuals  
 for(w in 1:firsti[i]){  
 p[i,w] = 1;  
 }  
 for(kkk in (firsti[i]+1):nocc)  
 p[i,kkk] = b0[sex[i],year[kkk-1]];   
 chi[i,nocc+1] = 1.0;   
 k = nocc;  
 while (k > firsti[i]) {  
 chi[i,k] = (1 - phi[i,k-1]) + phi[i,k-1] \* (1 - p[i,k]) \* chi[i,k+1];   
 k = k - 1;  
 }  
 if (firsti[i]>1) {  
 for (u in 1:(firsti[i]-1)){  
 chi[i,u] = 0;  
 }  
 }  
 chi[i,firsti[i]] = (1 - p[i,firsti[i]]) \* chi[i,firsti[i]+1];  
 }// close definition of transformed parameters for identified individuals  
  
 for(i in 1:nindni) {  
 // linear predictor for p for non-identified individuals  
 for(w in 1:firstni[i]){  
 p\_male[i,w] = 1;  
 p\_female[i,w] = 1;  
 }  
 for(kkkk in (firstni[i]+1):nocc){  
 p\_male[i,kkkk] = b0[1,year[kkkk-1]];   
 p\_female[i,kkkk] = b0[2,year[kkkk-1]];  
 }  
 chi\_male[i,nocc+1] = 1.0;   
 chi\_female[i,nocc+1] = 1.0;   
 k = nocc;  
 while (k > firstni[i]) {  
 chi\_male[i,k] = (1 - phi\_male[i,k-1]) + phi\_male[i,k-1] \* (1 - p\_male[i,k]) \* chi\_male[i,k+1];   
 chi\_female[i,k] = (1 - phi\_female[i,k-1]) + phi\_female[i,k-1] \* (1 - p\_female[i,k]) \* chi\_female[i,k+1];   
 k = k - 1;  
 }  
 if (firstni[i]>1) {  
 for (u in 1:(firstni[i]-1)){  
 chi\_male[i,u] = 0;  
 chi\_female[i,u] = 0;  
 }  
 }  
 chi\_male[i,firstni[i]] = (1 - p\_male[i,firstni[i]]) \* chi\_male[i,firstni[i]+1];  
 chi\_female[i,firstni[i]] = (1 - p\_female[i,firstni[i]]) \* chi\_female[i,firstni[i]+1];  
 } // close definition of transformed parameters for non-identified individuals  
  
   
 } // close block of transformed parameters exclusive parameter declarations  
} // close transformed parameters  
  
model {  
 // priors  
 theta ~ beta(1, 1);  
 for (g in 1:(nocc-1)){  
 b0[1,g]~beta(1,1);  
 b0[2,g]~beta(1,1);  
 }  
 a0[1,1]~normal(0,1.5);  
 a0[1,2]~normal(0,1.5);  
 a1[1,1]~normal(0,3);  
 a1[1,2]~normal(0,3);  
  
 a0[2,1]~normal(0,1.5);  
 a0[2,2]~normal(a0[1,2],0.01); // for juveniles, we assume that the effect of the covariate is independet of sex  
 a1[2,1]~normal(0,3);  
 a1[2,2]~normal(a1[1,2],0.01);  
  
 // likelihood for identified individuals  
 for (i in 1:nindi) {  
 if (lasti[i]>0) {  
 for (k in firsti[i]:lasti[i]) {  
 if(k>1) target+= (log(phi[i, k-1]));   
 if (yi[i,k] == 1) target+=(log(p[i,k]));   
 else target+=(log1m(p[i,k]));   
 }  
 }   
 target+=(log(chi[i,lasti[i]+1]));  
 }  
   
 // likelihood for non-identified individuals  
 for (i in 1:nindni) {  
 real log\_like\_male = 0;  
 real log\_like\_female = 0;  
  
 if (lastni[i]>0) {  
 for (k in firstni[i]:lastni[i]) {  
 if(k>1){  
 log\_like\_male += (log(phi\_male[i, k-1]));  
 log\_like\_female += (log(phi\_female[i, k-1]));  
 }  
 if (yni[i,k] == 1){   
 log\_like\_male+=(log(p\_male[i,k]));  
 log\_like\_female+=(log(p\_female[i,k]));  
 }  
 else{  
 log\_like\_male+=(log1m(p\_male[i,k]));   
 log\_like\_female+=(log1m(p\_female[i,k]));   
 }  
  
 }  
 }   
 log\_like\_male += (log(chi\_male[i,lastni[i]+1]));  
 log\_like\_female += (log(chi\_female[i,lastni[i]+1]));  
   
 target += log\_mix(theta[i], log\_like\_male, log\_like\_female);  
 }  
  
}

## Call Stan from R, check convergence and look at results

# Run STAN  
library(rstan)  
fit <- stan(file = "stanmodels/cmr\_mixture\_model.stan", data=datax, verbose = FALSE)  
# for above simulated data (25000 individuals x 15 time periods)   
# computing time is around 48 hours on an intel corei7 laptop  
# for larger data sets, we recommed moving the transformed parameters block   
# to the model block in order to avoid monitoring of p\_male, p\_female,   
# phi\_male and phi\_female producing memory problems  
  
# launch\_shinystan(fit) # diagnostic plots  
summary(fit)

## mean se\_mean sd 2.5% 25%  
## b0[1,1] 0.60132367 0.0015709423 0.06173884 0.48042366 0.55922253  
## b0[1,2] 0.70098709 0.0012519948 0.04969428 0.60382019 0.66806698  
## b0[1,3] 0.50293513 0.0010904085 0.04517398 0.41491848 0.47220346  
## b0[1,4] 0.28118209 0.0008809447 0.03577334 0.21440931 0.25697691  
## b0[1,5] 0.34938289 0.0009901335 0.03647815 0.27819918 0.32351323  
## b0[1,6] 0.13158569 0.0006914740 0.02627423 0.08664129 0.11286629  
## b0[1,7] 0.61182981 0.0010463611 0.04129602 0.53187976 0.58387839  
## b0[1,8] 0.48535193 0.0010845951 0.04155762 0.40559440 0.45750793  
## b0[1,9] 0.52531291 0.0008790063 0.03704084 0.45247132 0.50064513  
## b0[1,10] 0.87174780 0.0007565552 0.03000936 0.80818138 0.85259573  
## b0[1,11] 0.80185454 0.0009425675 0.03518166 0.73173810 0.77865187  
## b0[1,12] 0.33152443 0.0008564381 0.03628505 0.26380840 0.30697293  
## b0[1,13] 0.42132288 0.0012174784 0.04140382 0.34062688 0.39305210  
## b0[1,14] 0.65180372 0.0015151039 0.05333953 0.55349105 0.61560493  
## b0[2,1] 0.34237039 0.0041467200 0.12925217 0.12002285 0.24717176  
## b0[2,2] 0.18534646 0.0023431250 0.07547704 0.05924694 0.12871584  
## b0[2,3] 0.61351083 0.0024140550 0.07679100 0.46647727 0.56242546  
## b0[2,4] 0.37140208 0.0024464965 0.06962399 0.24693888 0.32338093  
## b0[2,5] 0.19428215 0.0034618302 0.11214798 0.02800056 0.11146326  
## b0[2,6] 0.27371336 0.0026553769 0.09054020 0.11827243 0.20785316  
## b0[2,7] 0.18611173 0.0014387436 0.05328492 0.09122869 0.14789827  
## b0[2,8] 0.25648337 0.0018258589 0.05287800 0.16255769 0.21913271  
## b0[2,9] 0.20378754 0.0021367769 0.07380004 0.07777998 0.15215845  
## b0[2,10] 0.52679548 0.0024625568 0.08696008 0.36214334 0.46594844  
## b0[2,11] 0.47393354 0.0032593161 0.10555065 0.28843967 0.39781278  
## b0[2,12] 0.22289155 0.0017082729 0.05551514 0.12576797 0.18203335  
## b0[2,13] 0.26191486 0.0024159794 0.07016314 0.14106495 0.21234017  
## b0[2,14] 0.65111737 0.0055743944 0.18780555 0.29279480 0.50957591  
## a0[1,1] 0.95440670 0.0013771881 0.04808748 0.86301660 0.92146330  
## a0[1,2] 0.01529770 0.0469699511 1.46995922 -2.82218067 -0.95533706  
## a0[2,1] 0.16384995 0.0049928331 0.12634422 -0.06399631 0.07533962  
## a0[2,2] 0.01535679 0.0469634175 1.47006964 -2.81864060 -0.95515751  
## a1[1,1] 0.15937249 0.0028992587 0.08864790 -0.01288607 0.10017613  
## a1[1,2] 0.08055953 0.1007089857 3.02148727 -5.95525636 -1.96662599  
## a1[2,1] -0.83614134 0.0074143920 0.18655882 -1.21033848 -0.95698565  
## a1[2,2] 0.08071668 0.1006904255 3.02145647 -5.94617355 -1.96508733  
## 50% 75% 97.5% n\_eff Rhat  
## b0[1,1] 0.60206306 0.6431566 0.7206343 1544.5301 1.002331  
## b0[1,2] 0.70165494 0.7355204 0.7946280 1575.4617 1.001482  
## b0[1,3] 0.50367411 0.5330078 0.5898079 1716.3196 1.001183  
## b0[1,4] 0.27997512 0.3046483 0.3544592 1649.0040 1.000760  
## b0[1,5] 0.34936442 0.3751935 0.4191138 1357.3073 1.002072  
## b0[1,6] 0.12987449 0.1481661 0.1873982 1443.8040 1.003676  
## b0[1,7] 0.61203228 0.6397577 0.6933929 1557.5904 1.001458  
## b0[1,8] 0.48513822 0.5134314 0.5672066 1468.1355 1.002511  
## b0[1,9] 0.52534212 0.5501747 0.5994060 1775.7335 1.000824  
## b0[1,10] 0.87324112 0.8934047 0.9258033 1573.3747 1.000719  
## b0[1,11] 0.80300311 0.8261868 0.8675033 1393.1817 1.001172  
## b0[1,12] 0.33044476 0.3552199 0.4052902 1794.9956 1.000566  
## b0[1,13] 0.42116690 0.4492297 0.5026942 1156.5339 1.000289  
## b0[1,14] 0.64956850 0.6864706 0.7607107 1239.4056 1.004061  
## b0[2,1] 0.33493631 0.4251416 0.6150923 971.5524 1.004049  
## b0[2,2] 0.17981663 0.2358847 0.3446097 1037.6210 1.001474  
## b0[2,3] 0.61326419 0.6644156 0.7628427 1011.8737 1.005727  
## b0[2,4] 0.36837778 0.4158585 0.5190457 809.8949 1.003803  
## b0[2,5] 0.17910449 0.2591418 0.4533117 1049.4733 1.001499  
## b0[2,6] 0.26739172 0.3299594 0.4685139 1162.6006 1.001170  
## b0[2,7] 0.18254607 0.2198969 0.3003156 1371.6455 1.000878  
## b0[2,8] 0.25280556 0.2895585 0.3704113 838.7174 1.005624  
## b0[2,9] 0.19724053 0.2501298 0.3694806 1192.8747 1.003687  
## b0[2,10] 0.52587075 0.5845730 0.7061694 1247.0027 1.002851  
## b0[2,11] 0.46874445 0.5392302 0.7046892 1048.7425 0.999473  
## b0[2,12] 0.21961656 0.2580782 0.3397127 1056.1081 1.000907  
## b0[2,13] 0.25601959 0.3056204 0.4142888 843.3960 1.003130  
## b0[2,14] 0.65824835 0.7973674 0.9698829 1135.0669 1.003838  
## a0[1,1] 0.95368445 0.9862439 1.0515747 1219.2071 1.003898  
## a0[1,2] 0.01633534 0.9911055 2.9717839 979.4231 1.003726  
## a0[2,1] 0.15519648 0.2472483 0.4230776 640.3489 1.004625  
## a0[2,2] 0.01587281 0.9898084 2.9659552 979.8429 1.003744  
## a1[1,1] 0.15647489 0.2205720 0.3354845 934.8953 1.007190  
## a1[1,2] 0.06683287 2.1568781 6.0295208 900.1297 1.003701  
## a1[2,1] -0.83503982 -0.7075691 -0.4814539 633.1119 1.010568  
## a1[2,2] 0.06586905 2.1557247 6.0239735 900.4432 1.003704

# Referenzen

Betancourt, M.~J. 2013. “Generalizing the No-U-Turn Sampler to Riemannian Manifolds.” *ArXiv E-Prints*, April. <http://arxiv.org/abs/1304.1920>.

Betancourt, M.~J., and M. Girolami. 2013. “Hamiltonian Monte Carlo for Hierarchical Models.” *ArXiv E-Prints*. <http://arxiv.org/abs/1312.0906>.

Betancourt, M.~J., M Girolami, Paul-Christian Bürkner, K P Burnham, D R Anderson, Bradley Efron, Trevor Hastie, et al. 2016. “WinBUGS user manual.” Edited by D. L. Thomson, E. G. Cooch, and M. J. Conroy. *ArXiv E-Prints*, Environmental and ecological statistics 3, 80 (1): 771. <https://doi.org/10.1371/journal.pone.0082490>.

Efron, Bradley, and Trevor Hastie. 2016. *Computer age statistical inference: Algorithms, evidence, and data science*. Institute of Mathematical Statistics Monographs.

Gelman, A, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. 2014. *Bayesian Data Analysis*. Third. New York: CRC Press.

Gelman, A, and J Hill. 2007. *Data Analysis Using Regression and Multilevel / Hierarchical Models*. Cambridge: Cambridge Universtiy Press.

Harju, S. 2016. “Book review:~Bayesian Data Analysis in Ecology Using Linear Models with R, BUGS, and Stan.” *The Journal of Wildlife Management* 80: 771.

Hastie, T, R Tibshirani, and J Friedman. 2009. *The Elements of Statistical Learning, Data Mining, Inference, and Prediction*. New York: Springer.

Hoffman, Matthew D, and Andrew Gelman. 2014. “The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo.” *Journal of Machine Learning Research* 15 (1): 1593–1623.

Korner-Nievergelt, F, T Roth, Stefanie von Felten, J Guélat, B Almasi, and P Korner-Nievergelt. 2015. *Bayesian Data Analysis in Ecolog Using Linear Models with R, BUGS, and Stan*. New York: Elsevier.

Mayfield, Harold F. 1975. “Suggestions for Calculating Nest Success.” *Wilson Bulletin* 87: 456–66.

McElreath, Richard. 2016. *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*. New York: Max Planck Institute for Evolutionary Anthropology; CRC Press.

Pledger, S., K. H. Pollock, and James L. Norris. 2003. “Open Capture-Recapture Models with Heterogeneity: I. Cormack-Jolly-Seber Model.” *Biometrics* 59: 786–94.