Stochastics Lab Course II

Khwam Tabougua Trevor

March 2019

INTRODUCTION

The "Stochastics Lab course II" is an Introductory Course for statistics and stochastics applications with R programming language. The course lasted for two weeks in March 2019. The report written on LaTeX, contains results, interpretations and figures from the ten exercises that had to be solved. Along with this report, there is also the R codes, which are recommended to understand the result.

CONTENTS

CHAPTER

ONE

TIDYVERSE

1.1 Problem description

R base tools can accomplish "almost" every programming tasks. However, when using large datasets or when implementing complex tasks(like graphs, maps, tidying, etc), things get complicated. We want to enhance our algorithms fo batter results or productivity. To this aim, we will use the Tidyverse package.

1.2 Methods

Tidyverse is a collection of packages for data manipulation, exploration and visualization. The core packages are **ggplot2, dplyr, tidyr, readr, purrr, tibble, stringr, and forcats**, but we will only be using ggplot2, dplyr, tidyr, and tibble.

- **ggplot2** is a system for declaratively creating graphics. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.
- **dplyr** is a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges such as adding new variables (that are functions of existing variables), picking variables based on their names, selecting rows (based on their value), reducing multiple values down to a single summary, and changing the ordering of the rows.
- **tidyr** package goal is to help you create tidy data. Tidy data is data where each variable is in a column, each observation is a row, and Each value is a cell.
- **tibble** package goal is to use tibbles, which are modern take on data frames. They keep the features that have stood the test of time, and drop the features that used to be convenient but are now frustrating (i.e. converting character vectors to factors).

1.3 Results

(a) After loading and filtering the data childrenfinal.dta, we convert some variables (namely tetanusmother, breastfeeding, wantedchild, anetalvisits, and placedelivery) into double labeled *<dbl>* (doubles, or real numbers).

(b)

- The figure 1.1 indicates that the effect of zstunt is negatively affecting te hypage.

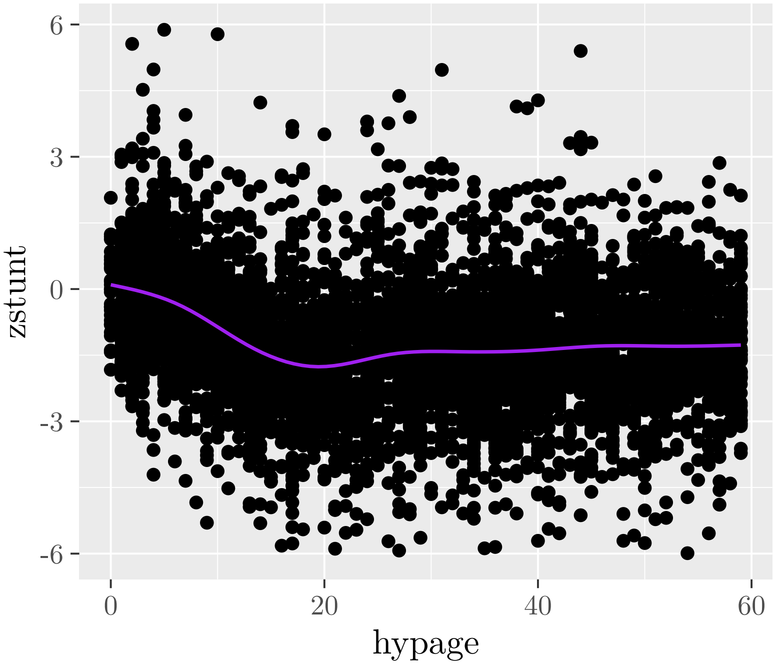

Figure 1.1: Scatter plot of zstunt against hypage with smooth line (in purple)

- gjdhgdhdg
- gjdhgdhdg

(c)

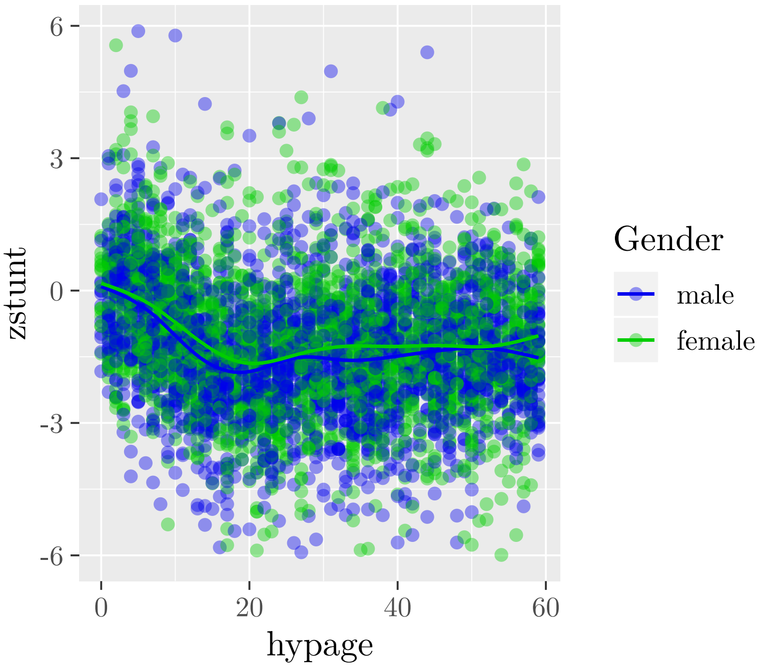

Figure 1.2: Some Meaningful Caption

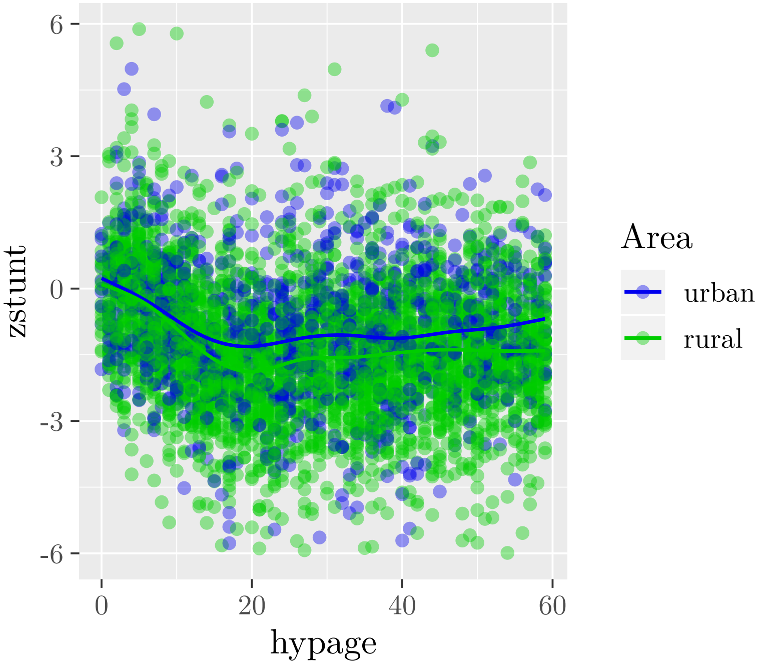

Figure 1.3: Some Meaningful Caption

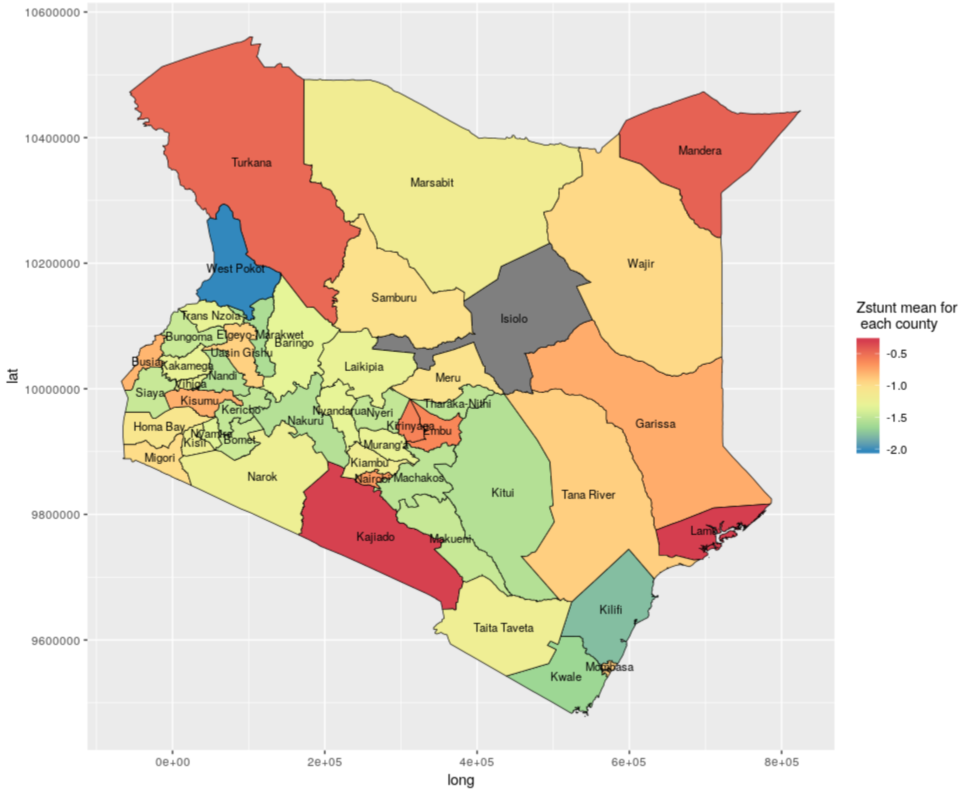

Figure 1.4: Scatter plot of zstunt against hypage with smooth line (in purple)

CHAPTER

TWO

RANDOM NUMBER GENERATION

2.1 Problem description

Generating a random variable from any distribution is very essential for stochastics studies. However, true random numbers are not always available, but we can use some algorithms that generate some pseudo-random numbers.

2.2 Methods

Pseudo-random numbers are a sequence of numbers that appear "random" or approximate properties of random numbers with the following properties:

- Good approximation of the properties of random numbers.
- Number can be easily and efficiently generated.
- Reproducibility (truly random numbers never satisfy this).

With the help of some probability properties, it is typically enough to be able to use a uniform distributed random variable, in order to generate any pseudo-random numbers from a given distribution. The simplest idea for generating uniformly distributed pseudo-random numbers is using a *linear congruent generators* (LCG):

1. Choose positive integer parameters a, c and m.
2. Choose an initial value $x_0 \in \{0, 1, \ldots, m-1\}$ (this value is called the seed).
3. For each $n \in \mathbb{N}$, compute

$$x_{n+1} := ax_n + c \mod m, \qquad \forall n \in \mathbb{N} \tag{2.2.1}$$

4. The psuedorandom numbers are the sequence $x_1; x_2; x_3; \ldots$.

We make some observations regarding the LCG algorithm.

- The LCG can generates at most m distinct numbers which are contained in $\{0, 1, \ldots, m-1\}$
- As soon as some number in the sequence, say x_{n0}, is repeated (i.e.,$\exists p$ such that $x_{n_0+p} = x_{n0}$), then the same is true for the entire sequence:

$$x_{n_0+p+j} = x_{n_0+j}, \qquad \forall j \geqslant 1 \tag{2.2.2}$$

  The number p is called the period of the sequence. This musst be less than or equal to m.
- Suppose that (under the right conditions) the LCG approximates a uniform distribution with parameters 0 and m. In this case, x_n/m would approximate a uniform distribution with parameters 0 and 1.

To generate uniformly random numbers, the period has to be maximal (i.e., $p = m$), so that we sample every value in the sequence before repeating any. One can show that LCG has a full period $m = 2^b$, $b \geqslant 2$ if and only if $c \in (0; m)$ is odd and $a \bmod 4 = 1$.

After generating uniform pseudo-random numbers, we can easily obtain random variables from other distributions. The **inversion method** is one way of doing so, with the help of the following theorem:

Theorem 1 *Let F be a distribution function on $\mathbb{R}$. The quantile function F^{-1} is defined by*

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

If $U \sim U_{[0;1]}$, then $F^{-1}(u)$ has a distribution function F.

Hence, for continuous distributions (exponential, Pareto, standard Cauchy, etc) where F^{-1}, we simply simulate U_i (with LCG) and set $X_i = F^{-1}(U_i)$. If F^{-1} cannot be inverted analytically, appropriate numerical methods can be applied.

Let X be a discrete random variable with ordered possible values $\{x_1, x2, \ldots\}$, so that $F(x) = \sum_{i:x_i \leqslant x} P(X = x_i)$ and

$$F^{-1}(r) = \min\{x_k \in \{x_1, x_2, \ldots\} : \sum_{j=1}^{k} P(X = x_j) = \sum_{j=1}^{k} p_j \geqslant r\}$$

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

$$P(X = x_k) = P\Big(\sum_{j=1}^{k-1} p_j \leqslant U_i < \sum_{j=1}^{k} p_j\Big) = \sum_{j=1}^{k} p_j - \sum_{j=1}^{k-1} p_j = p_k$$

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

Another general approach to pseudo-random variables generation is the **acceptance-rejection method**.

```
Data: Two probability densitiy functions: $f$ for $X$ and $g$ for $Y$
1. Find a constant $M > 0$ such that $sup_x \frac{f(x)}{g(x)} \leqslant c$ ;
2. Obtain a sample $y$ from $Y$ ;
3. Obtain a sample $u$ from the uniform distribution on $[0, 1]$;
4. if $u < \frac{f(y)}{cg(y)}$ then
5.     Accept $y$ as a sample drawn from $f$;
6. else
7.     Reject the value of $y$ and return to the sampling step (line 2);
Result: $y$, a sample drawn from $f$ (using $g$)
```

NB: Computing c could be difficult, but one can show that $c = sup_x \frac{f(x)}{g(x)}$

2.3 Results

(a) With the Wichmann-Hill pseudo-random number generator in R, we Simulate $N = 1000$ binomial random variables $B(n = 10, p = 0.4)$ using three approaches: inversion

method, by simulating corresponding Bernoulli random variables by inversion method and using R built-in function rbinom. From the figure 2.1, the histograms of the three samples present the same shape but are different. This proves that the "random" numbers generated by our methods are just approximations of true random numbers.

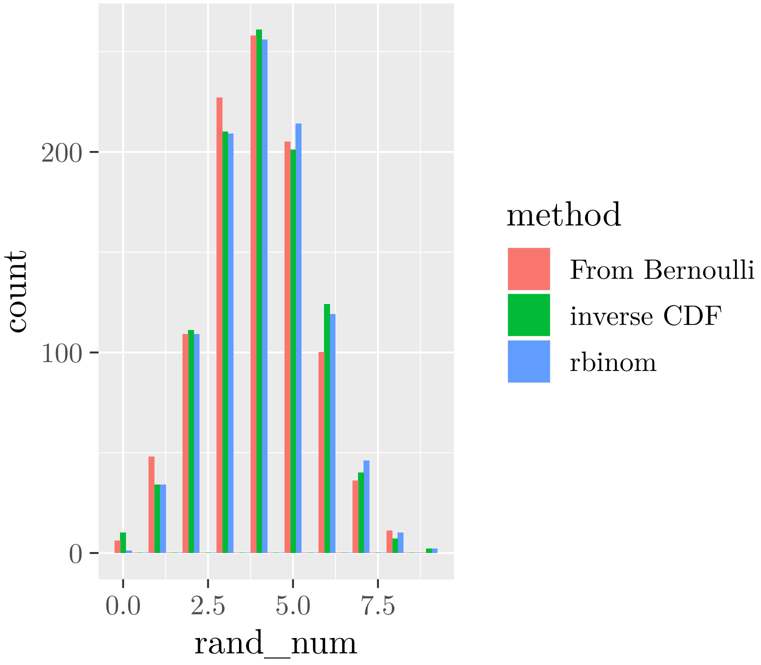

Figure 2.1: Histogram of the empirical CDF of all three samples

(b) To use accept-reject method (and a generator for uniform random variables only), of $N = 10000$ standard normal distributed random variables with density $f(x) = (2\pi)^{-\frac{1}{2}}e^{-\frac{x^2}{2}}$, the density of the standard Cauchy distribution is used: $g(x) = \{\pi(1+x^2)^{-1}\}$.

- ◦ The constant value c for this method is given by $sup_x \frac{f(x)}{g(x)} = 1.520347$
- ◦ Then after computing the N standard normal random variables, we notice that the estimated and theoretical acceptance probabilities are almost equal. This is well depicted with in the figure 2.2, where the histogram of the obtain sample is symmetric and has the same shape as the standard normal density curve.

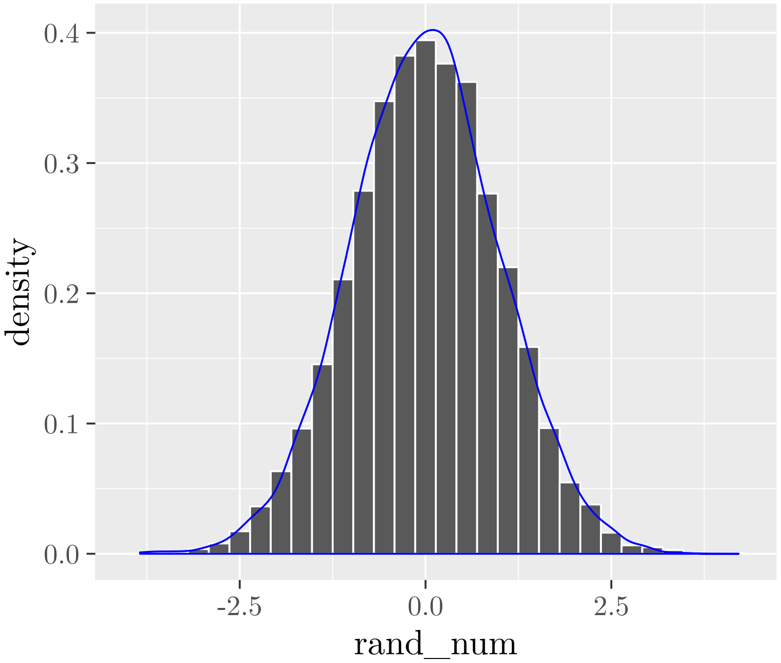

Figure 2.2: Histogram of the obtained sample and the standard normal density (in blue)

- The QQ-plot in figure 2.3 shows points following the identity line. Hence the accept-reject method used to simulate a standard normal distributed sample (using the the standard Cauchy density) is well accurate.

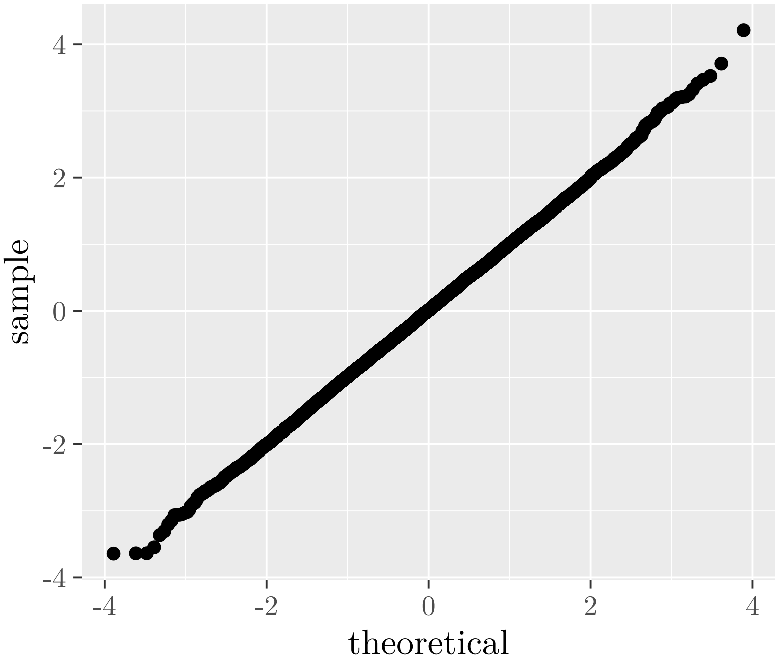

Figure 2.3: QQ-plot

- However, it is not possible to simulate ample distributed from the standard Cauchy density using the accept-reject method with a standard normal candidate density, simply because cannot find a c such that $g(x) \leqslant cf(x)$ is verified (because $sup_x \frac{g(x)}{f(x)} = \infty$).

CHAPTER

THREE

BOOTSTRAP

3.1 Problem description

Suppose that a sample $\mathbf{X} = \{X_1, \dots, X_n\}$ is used to estimate a parameter θ of the distribution P (which is unknown) and let $\hat{\theta} = S(\mathbf{X})$ be a statistic that estimates θ. For the purpose of statistical inference on θ, we are interested in the sampling distribution of $\hat{\theta}$ (or certain aspects of it) so as to assess the accuracy of our estimator or to set confidence intervals for our estimate of θ. If the true distribution P were known, we could draw samples $\mathbf{X}_l, l = 1, \dots, R \in \mathbb{N}$ from P and use Monte Carlo methods to estimate the sampling distribution of our estimate $\hat{\theta}$. The problem is that P is unknown and we cannot sample from it.
The following section explains how to use bootstrap to make the interference on $\hat{\theta}$.

3.2 Methods

The bootstrap is a computerintensive resampling method, which principle can be summarized by the following schematic diagram:

Real World **Bootstrap World**

Unknown probability distribution | Observed random sample | Empirical distribution | Bootstrap sample

$$\begin{tikzcd}
P \arrow[r] & \mathbf{X} = \{X_1, \dots, X_n\} \arrow[d] & \Longrightarrow & \hat{P} \arrow[r] & \mathbf{X}^* = \{{X_1}^*, \dots, {X_n}^*\} \arrow[d] \\
& \hat{\theta} = S(\mathbf{X}) & & & \hat{\theta}^* = S(\mathbf{X}^*)
\end{tikzcd}$$

Statistic of interest | Bootstrap replication

Then The idea is to sample from an empirical distribution function. Recall that for random variables $Y = \{Y_1, \dots, Y_n\}$, the empirical distribution function is defined via $F_n(y) = n^{-1} \sum_{i=1}^{n} \mathbb{I}(Y_i \leqslant y)$. If the sample of size n is from a continuous distribution, then each observation has a probability $1/n$ and sampling from F_n would be equivalent to draw with replacement from the sample. Hence the following algorithm:

1. Draw n times with replacement from $\mathbf{X}$ to get a bootstrap sample $\mathbf{X}_1^*$ of size n. Repeat R times to get R bootstrap samples $\mathbf{X}_1^*, \ldots, \mathbf{X}_R^*$, each of size n.
2. Compute bootstrap statistics $S(\mathbf{X}_1^*), \ldots, S(\mathbf{X}_R^*)$.
3. Make inference about θ based on $S(\mathbf{X}_1^*), \ldots, S(\mathbf{X}_R^*)$.

We can also evaluate the goodness of the estimators (point or interval) based on the bootstrap sample. We construct confidence intervals for θ from the bootstrap replications (see step 2 in the above algorithm). First recall the definition of a confidence interval.

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

$$\inf_{P \in \mathcal{P}} P(\theta \in C(\Theta)) \geqslant 1 - \alpha$$

for a fixed $\alpha \in (0, 1)$, then $C(\Theta)$ is a **confidence set** for θ with **level of significance** $1 - \alpha$. If the parameter θ is real-valued, then $C(\Theta) = [\underline{\theta}(\mathbf{X}), \bar{\theta}(\mathbf{X})]$, for a pair of real-valued statistics $\underline{\theta}$ and $\bar{\theta}$ is called a confidence interval for θ.

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

3.3 Results

CHAPTER

FOUR

GENERALISED LINEAR MODELS

4.1 Problem description

4.2 Methods

4.3 Results

CHAPTER

FIVE

SURVIVAL ANALYSIS

5.1 Problem description

We want to analyze data where the outcome variable is the time until the occurrence of an event of interest. The event can be death, occurrence of a disease, marriage, divorce, etc. The time to event or survival time can be measured in days, weeks, years, etc. For example, if the event of interest is heart attack, then the survival time can be the time in years until a person develops a heart attack. subjects are usually followed over a specified time period and the focus is on the time at which the event of interest occurs. Why not use linear regression to model the survival time as a function of a set of predictor variables? First, survival times are typically positive numbers; ordinary linear regression may not be the best choice unless these times are first transformed in a way that removes this restriction. Second, and more importantly, ordinary linear regression cannot effectively handle the censoring of observations. Why not compare proportion of events in your groups using risk/odds ratios or logistic regression? Simply because it ignores time.
To tackle these issues, we'll use some survival analysis methods.

5.2 Methods

5.3 Results

CHAPTER

SIX

KERNEL DENSITY ESTIMATION

6.1 Problem description

Consider observations which are realizations of univariate random variables, $X_1, \ldots, X_n \sim F$ where F denotes an unknown cumulative distribution function. The goal is to estimate the distribution F. In particular, we are interested in estimating the density $f = F^{'}$, assuming that it exists.
Instead of assuming a parametric model for the distribution (e.g. Normal distribution with unknown expectation and variance), we rather want to be "as general as possible": that is, we only assume that the density exists and is suitably smooth (e.g. differentiable). It is then possible to estimate the unknown density function $f(\cdot)$.

6.2 Methods

6.3 Results

CHAPTER

SEVEN

NONPARAMETRIC REGRESSION: LOCAL POLYNOMIALS

7.1 Problem description

To study the relation between a dependent variable Y and an independent variable X, the common method used is linear regression. When appropriate, this method is very useful as its suppose a simple model of the form

$$Y = \beta_0 + \beta_i x_i + \epsilon_i \tag{7.1.1}$$

This is advantageous since it is easy to interpret and to calculate. Moreover, when the assumptions on the residues ϵ_i are verified, we can run some tests on the parameters.
However, the restricted assumption of linearity is frequently not fulfilled, eventually when the data set is very large. In that case, we would like to find a complex model that will better highlight the relation between Y and X. A first approach for this aim would be to specify another parametric form for this relation, for example a transformation of the observations or a polynomial regression. Nonetheless it remains difficult to find the suitable relation since the form of the data does not really change after these transformations. That is why in this section, we opt for a non-parametric regression technique (local polynomials) in which data choose their own form of relation (the predictor does not take a predetermined form but is constructed according to information derived from the data) making things more flexible.

7.2 Methods

7.3 Results

CHAPTER

EIGHT

NONPARAMETRIC REGRESSION: SPLINES

8.1 Problem description

8.2 Methods

8.3 Results

CHAPTER

NINE

MIXED MODELS

9.1 Problem description

To illustrate the targeted problem ib this section, we use the following example. Let us consider the following linear model,

$$Y_{i,t} = \beta_0 + \beta_i t + \epsilon_{i,t} \tag{9.1.1}$$

Here, β_0 and β_i

9.2 Methods

9.3 Results

CHAPTER

TEN

PARTIAL LEAST SQUARES

10.1 Problem description

In a standard linear model, we have at our disposal (X_i, Y_i) supposed to be linked with,

$$Y_i = X_i^t \beta + \epsilon_i, \qquad 1 \leqslant i \leqslant n \tag{10.1.1}$$

In particular, each observation X_i is described by p variables $(X_1, \ldots, X_n)$ so that the former relation should be understood as

$$Y_i = \sum_{j=1}^{p} \beta_j X_i^j + \epsilon_i, \qquad 1 \leqslant i \leqslant n \tag{10.1.2}$$

From a matricial point of view, the linear model can we written as follows :

$$Y_i = X\beta_0 + \epsilon_i, \qquad Y \in \mathbb{R}^n, X \in \mathcal{M}_{n,p}, \beta_0 \in \mathbb{R}^p \tag{10.1.3}$$

A classical "optimal" estimator is the MLE :

$$\hat{\beta}_{MLE} := (X^t X)^{-1} X^t Y \tag{10.1.4}$$

This can be obtained while remarking that J is a convex function, that possesses a unique minimizer if and only if $X^t X$ has a full rank, meaning that J is indeed strongly convex :

$$D^2 J = X^t X \tag{10.1.5}$$

Which is a squared $p \times p$ symmetric and positive matrix. It is non degenerate if $X^t X$ has full rank, meaning that necessarily $p \leqslant n$.

In large dimensional case, we often have $p > n$, hence a problem when applying linear regression in this case:
$X^t X$ is an $p \times p$ matrix, but its rank is lower than n. If $n << p$, then

$$rk(X^t X) \leqslant n << p \tag{10.1.6}$$

Consequently, the the Gram matrix $X^t X$ is not invertible and even very ill-conditionned (most of the eigenvalues are 0 !). The linear model $\hat{\beta}_{MLE}$ completely fails.
As a remedy to this problem that occurs most of the time in big data analysis, we will make use of the partial least squares (PLS) method.

10.2 Methods

10.3 Results