Stochastisches Praktikum II
WiSe 2020/21

Anand Deshpande

July 25, 2021

Contents

1 Tidyverse

1.1 Z-score scatter plots

In this exercise we have a data of children between the 0 and 5 years in Kenya with the Z-score for stunting. In the first part we examine the development over time and based on gender of the children. And then based on whether they are living in rural or urban regions.

- Using the functions and methods provided by the **tidyverse** package we can do following tasks;
  - data loading
  - dropping irrelevant variables
  - checking the type of the remaining variables.
- Using the **haven** package the data in STATA format is loaded.

After data preparation we plot the age of children against the Z-score.

- In the first plot we consider the whole population (see Figure 1.1)
  1. We observe that the Z-score decreases after birth from an average level until it nearly reaches the critical threshold of -2 at an age about 18 months.
  2. After the age of 18 months the score increases slightly and seems to converge to a limit between 1 and -1.5.
  3. Therefore we observe that children in Kenyan are general are born with an average height.
  4. But, at least until the age of 5 years children's height is below the average .
  5. It's also important to note that, there is a substantial amount of points below the critical value which suggests that many children considered as stunted.

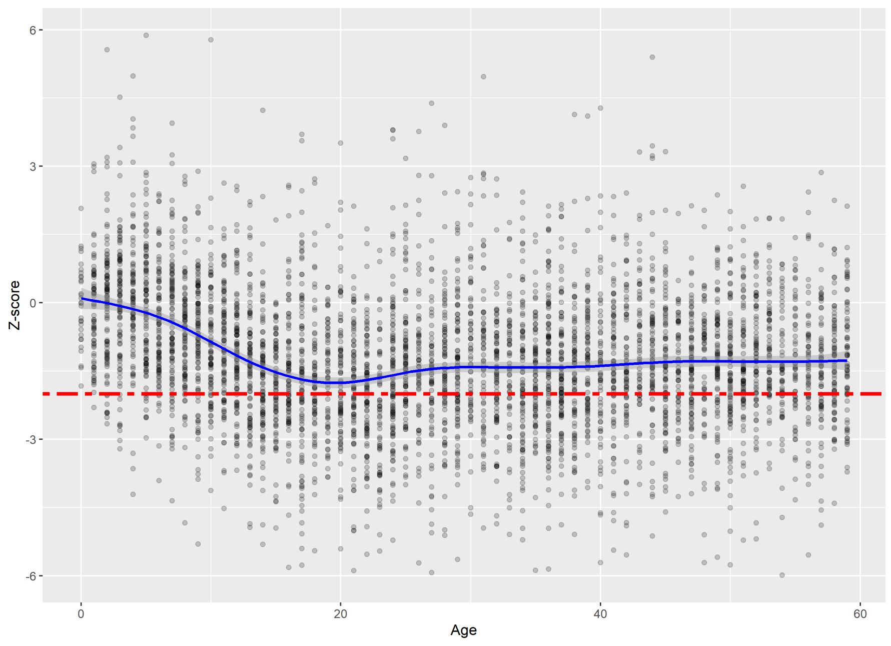


Figure 1.1: Scatter plot for age (in months) against Z-score, with a smooth line. The red line (at `zstunt` $= -2$) shows the critical value.

- We split the data first based on gender and then based on whether children are living in rural or urban regions of the country. (see Figure 1.2)
  1. We observe similar patter as we observed for the overall data
  2. We can also observe a slight but visible difference that the Z-score is lower in male children and those who live in rural areas of the country
  3. The difference between the genders in Z-score reduces with time
  4. The inequality caused by the environment seems to persist
  5. We see that for the urban part of the population the Z-score is increasing slowing and approaching an average level
  6. For the children living in rural areas the Z-score seems constant at a value of -1.5.

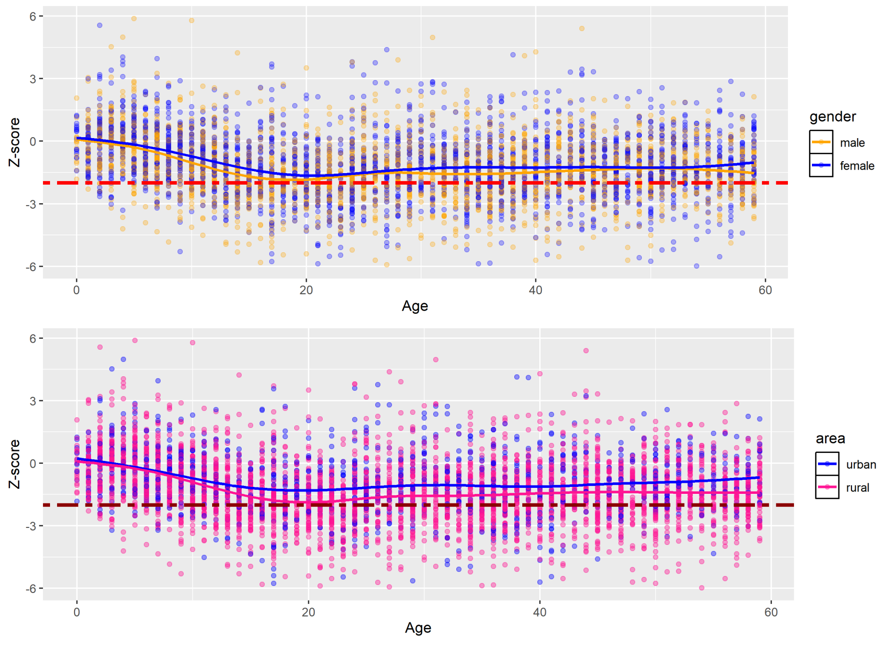


Figure 1.2: Scatter plots for age (in months) against Z-score split by gender (upper plot) and rural-urban factor (lower plot) with a smooth line. The red line (at `zstunt` $= -2$) shows the critical value.

1.2 County wise Z-score

In this exercise we use the functions of `tidyverse` and `ggplot2` to draw maps of counties. We take the data set provided by the GADM from the package `raster`. The task is to draw a map of Kenya's counties and fill the areas according to the mean Z-score (see Figure 1.3).

- For the county Isiolo there is no data, so it's left gray
- As its represented in the legend, towards the shades of blue, more is Z-score and towards the shades of red, less is Z-score.
- There is visible differences in means Z-score between different counties
- We can observe a general pattern that the western and southern parts of the country suffers more from stunting than the northern parts.
- Except Kajiado all the counties in the south have a lower mean Z-score
- In the county West Pokot children are more stunted.
- Counties with good Z-score are on the boundaries in different direction.

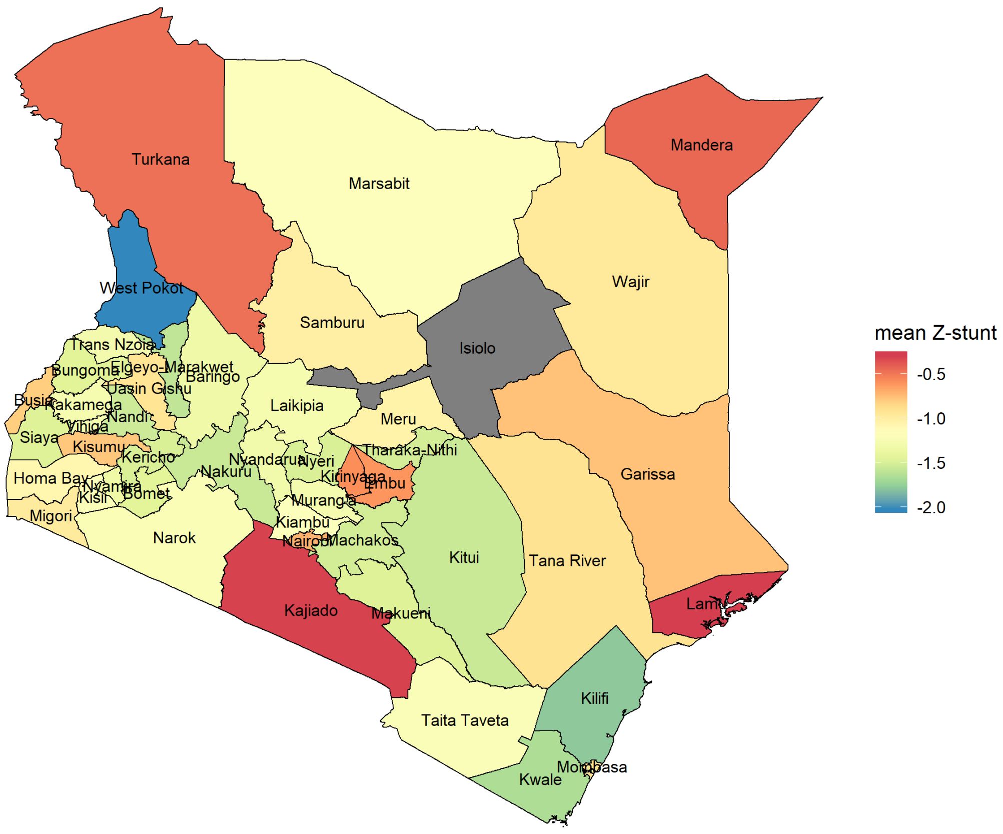


Figure 1.3: Map of Kenya with county-wise mean Z-score.

2 Pseudo Random Number Generation

2.1 Producing binomial random variables

The exercise is about different methods to produce pseudo random numbers with different distributions. One can do it in several ways to get pseudo random numbers. Most of methods use uniform distributed variables in some way or other. We don't consider the process of simulating uniformly distributed variables here. There are different methods, like Whichmann-Hill or the LCG, that are provided by R.
One easy way to get random simulations of a specific distribution is by using uniformly distributed variables and the quantile transformation of the desired distribution. In the first part of the exercise we simulate binomial random variables using this idea. The second part is about the accept-reject method to simulate standard normal random variables.

The inversion method is based on the generalised inverse of the distribution function (cdf) F which is considered to be defined on $\mathbb{R}$ here and its inverse defined on $t \in (0,1)$ as

$$F^{-1}(t) = inf\{x \in \mathbb{R} | F(x) \geq t\}.$$

This, requires that the cdf is in this way analytical invertible. Additionally, if $U \sim unif[0,1]$ holds, then $F^{-1}(U) \sim F$, i.e., it is possible to generate F-distributed variables using uniform variables.

In this exercise we were asked to simulate binomial random variables with parameters $n = 10$ and $p = 0.4$. The inverse is easily computed as a sum of indicator functions over the intervals $[0, B(n,p)(\{k\})]$ for $k = 0, \ldots 10$. So we produced 1000 binomials using the inverse of simulated uniforms. Another way to get binomial distributed variables is the summation of independent Bernoulli random variables with success probability p. Those can also be simulated using the inverse method with $F^{-1}(t) = \mathbb{1}_{[0,p)}(t)$. Therefore we used in another trial 10000 uniform distributed variables that we inverted to Bernoulli variables and added up in the end. Finally we compared those two approaches with the provided method `rbinom` which is based on a special accept- reject-method. In the visualisation we can see very clear that all approaches lead to a similar distribution (see Figure 2.1). Although the binomial distribution is discrete, I have added a smooth density estimate to compare the overall shape of the distributions.

So we see that the first two method, i.e. those using inverse method just differ in a slightly more centred distribution for method 1. The blue bins show a small shift of the distribution to the right most dominant at 5.

Drawing a histogram of three groups into one plot is sometimes problematic because the bins are drawn shifted in a graphical way. This shift may suggest a shift in distri-

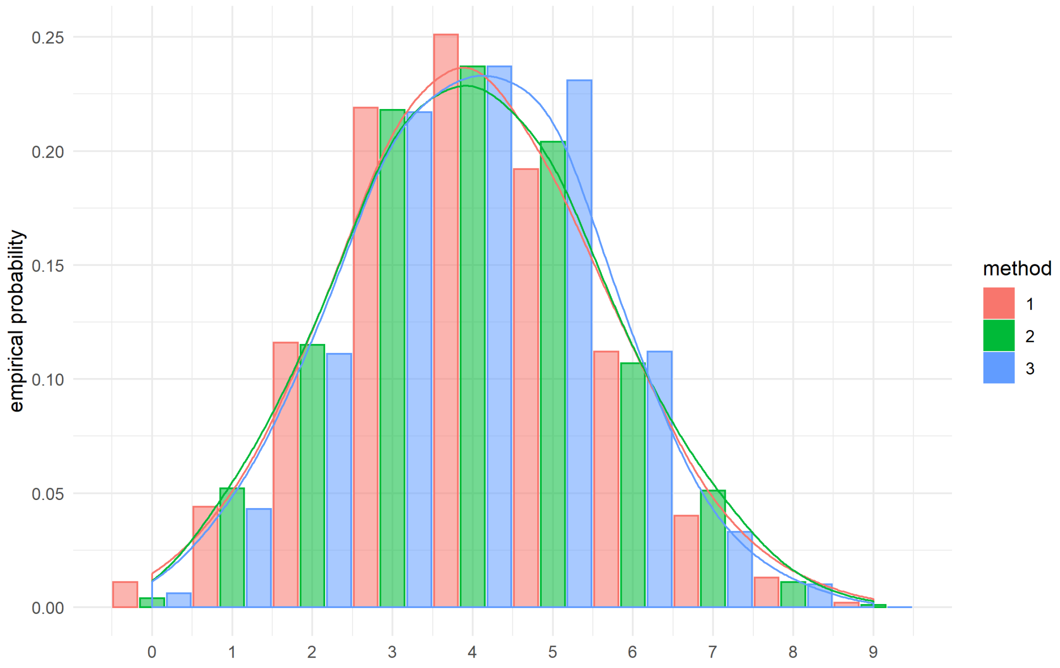


Figure 2.1: Empirical distribution of simulated binomial variables ($n = 10, p = 0.4$).
Method 1: inverse method with the binomial quantile function;
Method 2: Summation of Bernoulli variables (simulated with uniform variables and inverse method);
Method 3: simulated with `rbinom`.

bution that is not existent in the data. Therefore I have added another graphic (Figure 2.2) where I put the three histograms beneath each other. As the three distributions are very close, in general differences are hard to see, but one sees that method 1 generates a more centred distribution and that the first row shows a slightly more right skewed distribution, as one would expect from a binomial with $p = 0.4$. All in all the differences seem somehow negligible and we can assume that every method produced reasonable simulations.

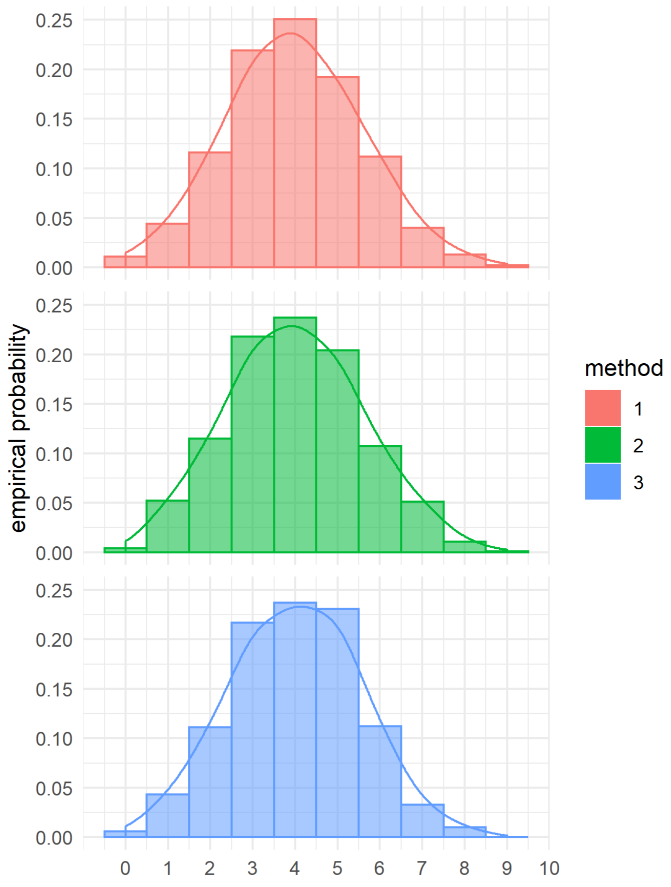


Figure 2.2: Histograms of simulated binomial variables with the three methods.

2.2 Accept-reject method for normal distribution

The next task is to produce standard normal random variables with the accept-reject method. Therefore one has to be able to simulate variables from another distribution with known density. Here we use the standard Cauchy distribution with the density $g(x) = [\pi(1 + x^2)]^{-1}$.

The general approach for generating variables with a density function f by variables with a distribution given by a density g demands a constant $c \geq 1$, s.t. $f(x) \leq cg(x)$ for all x and is given by following algorithm:

1. generate random variable X from density g
2. generate uniformly distributed $U \sim unif[0, 1]$ (independent from X)
3. Accept X as drawn from f, if $Ucg(X) \leq f(X)$, else reject.

The resulting variables X are then distributed with density f. In this exercise we produce the variables X with the inversion method. The cdf of the standard Cauchy distribution is given by

$$G(x) = (\arctan(x) - 0.5)/\pi$$

and therefore we get the inverse

$$G^{-1}(t) = \tan\{\pi(y - 0.5)\}.$$

To determine the best c for the rejection procedure, we search for the smallest possible constant - as we don't want to reject unnecessary many X so we have to draw less candidates and thus keep the algorithm fast - such that $c \geq \frac{f}{g}(x)$ for all x, i.e.

$$c := \sup_{x \in \mathbb{R}} \frac{f(x)}{g(x)} = \sup_{x \in \mathbb{R}} \frac{1}{\sqrt{2\pi}} \exp\{-x^2/2\}\pi(1 + x^2).$$

By differentiation we get three local extrema and the maxima at $x \in \{-1, 1\}$ resulting in $c = \sqrt{2\pi}e^{-1/2}$.

We simulated $10,000$ standard normals using this method after setting the random number generator back to default, i.e. using Mersenne-Twister. The result is shown in Figure 2.3. There we can see a very good fit of the data.

Additionally one can examine the normality of a sample for example with QQ- and PP- plots. The QQ-plot for the sample is shown in Figure 2.4. The points lie pretty much on the reference line. Just the few highest quantiles differ. Since on the tails are just a few data points this may happen by accident. I omit other plots since a very good fit of the data is already obvious in the histogram and the QQ- plot.

Because in this method we reject some samples of the distribution g, it is interesting to consider the acceptance probability. The theoretical overall acceptance probability should be $1/c = 0.658$. In this simulation we used 15 168 Cauchy samples resulting giving a similar empirical acceptance probability of 0.659. One can also look in more detail and

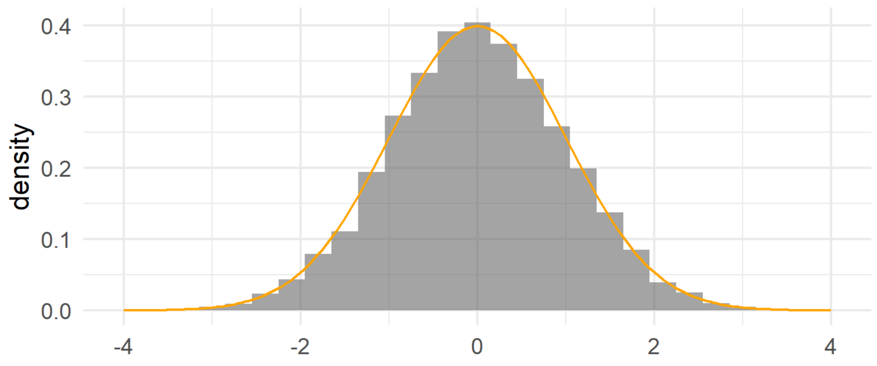


Figure 2.3: Histogram of simulated standard normals. The orange line shows the density of a standard normal variable

visualise more local acceptance probabilities. The probability that the uniform variable drawn in the second step of the algorithm accepts the sample X as drawn from f is

$$P(U \leq \frac{f(X)}{cg(X)}) = \frac{f(X)}{cg(X)}$$

and can be seen as a function in X. I estimated the local empirical acceptance probabilities by binning the data with the `hist` function in R (see Figure 2.5). Since the Cauchy distribution has very heavy tails compared to the standard normal, there are of course very high and low values in the Cauchy sample. I restricted the data for the visualisation to the area, where the empirical acceptance probability is positive (and not 0), i.e. where we have indeed data in the final normal sample. As expected we see in the plot that the empirical probabilities fluctuate around the theoretical ones over the whole range of values.

The last task of the exercise is to check, whether it is possible to simulate Cauchy distributed samples from a standard normal one. As mentioned the Cauchy distribution hat heavy tails compared to the normal. This can be also seen in the quotient of densities.

$$\frac{g(x)}{f(x)} = \sqrt{2\pi} exp\{x^2/2\}[\pi(1+x^2)]^{-1}$$

This is obviously unbounded as x tends to infinity, i.e. we can't find a suitable constant c for the algorithm. This can also be seen in Figure 2.5. As the limit of the function is 0 the inverse has to go to infinity.

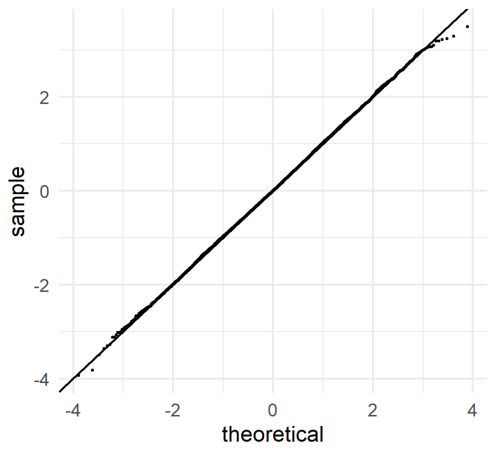


Figure 2.4: QQ-plot of a simulated sample of standard normals.

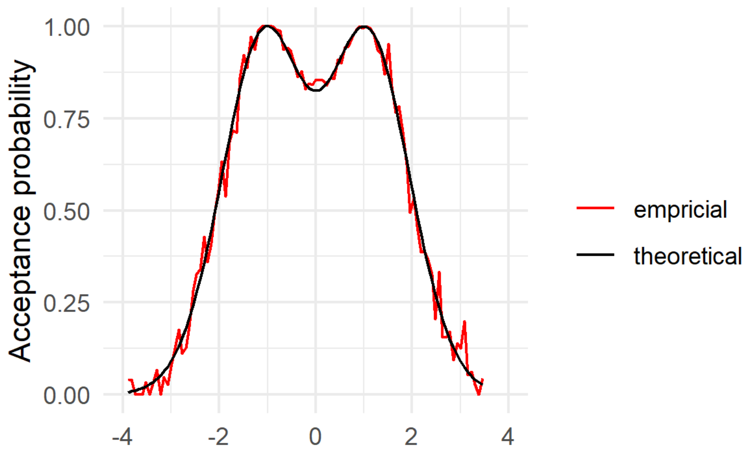


Figure 2.5: Theoretical and empirical acceptance probabilities of the normal sample generated with the accept-reject method.

3 Bootstrap

3.1 Simulation of bootstrap confidence intervals

The third exercise is about the bootstrap method to construct confidence intervals (CI) for different distribution measures, e.g. standard deviation (SD) and median here. The first part is a simulation study to examine coverage properties and length of the bootstrap percentile CI (bCI) for SD and median for a Weibull distribution (with parameters $\lambda = 13$ and $k = 1$ - so it is actually an exponential distribution). Therefore we drew M Monte Carlo samples of n Weibull distributed variables. For each sample we did R bootstrap repetitions where I used a n-out-of-n method with replacement like described in the lecture. The bCI at level 95% was then defined as the interval between the 0.25th and the 0.75th quantile of the bootstrap distribution for the respective statistic. For each MC sample I checked whether the true SD and median where covered and saved the length of the CIs. Where $M = 1000$ was fixed we compare the results for different choices of n and R, i.e. we varied the actual sample size and the number of bootstrap runs. The results are shown in Table 3.1. As we took the 95% bCI we would expect a corresponding

		Cover probability		Avg CI length	
R	n	med	SD	med	SD
1000	100	0.94	0.86	5.02	6.18
1000	1000	0.93	0.94	1.60	2.21
5000	100	0.95	0.86	5.18	6.06
`bcanon`					
1000	100	1.00	1.00	3.91	7.42

Table 3.1: Coverage probability and average CI length for median and SD for the MC simulation of 95% bootstrap percentile CIs with sample size n and R bootstrap repetitions and for the bootstrap accelerated bias-corrected CI using `bcanon`.

coverage probability. For the median this is almost achieved in every row. We also see that the differences between the first and the third row are rather small and that the cover probability lies beneath the desired level. On the other hand the coverage for SD increases notable when we increase the sample size n and is about 95%. Simultaneously the length of the bCI drastically decreased, showing that we get a higher precision for both statistic. This precision is not gained by using more bootstrap repetitions. This demonstrates that the amount of information we can pull from a fix sample size is bounded and can't be arbitrarily increased using more bootstrap samples. On the other hand, increasing the sample size, i.e. collecting more *independent* information helps estimating better CI.

To visualise the empirical distribution of the sample statistics I include histograms for the SD and median of the MC samples together with a bCI for an additional sample and histograms of the CI lengths with the length for this additional sample (see Figure 3.1). The bCIs of this example cover 0.81 of the samples' SD and 0.89 of the median. This is a bit lower than the ones shown in the first row of Table 3.1 since the empirical estimates vary independently of the CI around the true statistics.

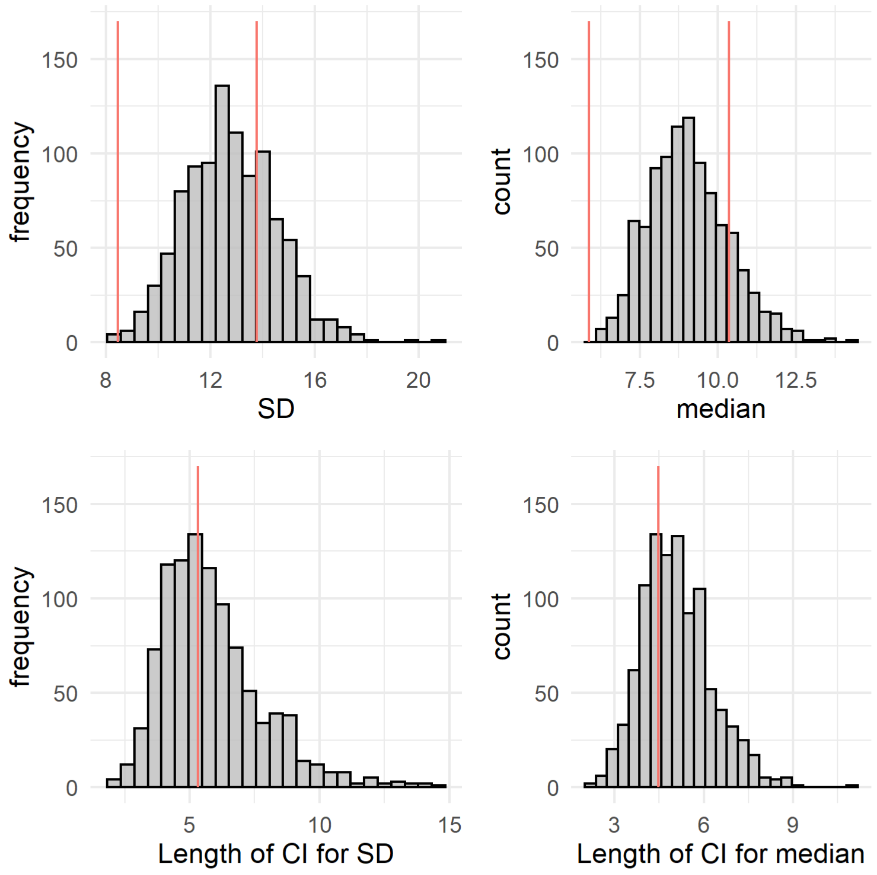


Figure 3.1: Histogram for 1000 MC samples of the two statistics (first row) and the length of the 95% bCI (second row). The red lines show the bCI for an independent sample (first row) and the respective length (second row).

Instead of using just the percentiles of the bootstrap distribution there is the approach of bias corrected bootstrap CIs. The function `bcanon` provided in the package *bootstrap* computes an accelerated and bias-corrected bootstrap CI. A MC simulation gives good results (see last row in Table 3.1) as the true statistics are always covered. The length of the CI for the median is smaller but for the SD it is greater than the ones in the first row. The mean value for the bias correction $\hat{z}_0$ is about -0.026 for the median and 0.124 for the standard deviation. Taking the value of the standard normal cdf of the values shows us that on average 0.49 of the bootstrap medians are smaller than the sample median. The same holds for 0.55 of the bootstrap standard deviations respectively. On average one would expect those values close to 0.5 and thus the average $\hat{z}_0$ close to 0. The acceleration constant is computed by a transformation of the Jackknife statistics and lies at a scale of 10^{-15} in this case and thus is negligible. Looking at the formula this is not surprising. The median and standard deviation should not change remarkable when leaving out one instance. Especially the median ranges between at most three values anyway. Together with the small sample size of 100 the sum of distances from the Jackknife statistics' mean is not very high. Therefore we can say that the CI is constructed by taking the $\lfloor R\alpha_1 \rfloor$th and the $\lfloor R\alpha_2 \rfloor$th instances of the ordered bootstrap statistics, where $\alpha_1 = \Phi(z_{\alpha/2} + 2\hat{z}_0)$, $\alpha_2 = \Phi(z_{1-\alpha/2} + 2\hat{z}_0)$ and Φ denotes the standard normal cdf.

3.2 Application for Sleep Heart Health Study data

In this section we use the bootstrap method to build CIs for a real data set. We are just concerning the variable `rdi4p` encoding the respiratory disturbance index. The histogram (Figure 3.2) shows the distribution of the data. As it looks like an exponential distribution (i.e. a Weibull distribution with shape parameter 1), I added the respective density with mean equal to the sample mean ($1/\lambda = 8.66$). The fit looks good and it would get better when refining the bin size, but for clearness is stay with this size.

Now we want to estimate confidence intervals for the median and standard deviation for `rdi4p`. We use $R = 1000$ bootstrap repetitions. The sample length is $n = 5804$ and

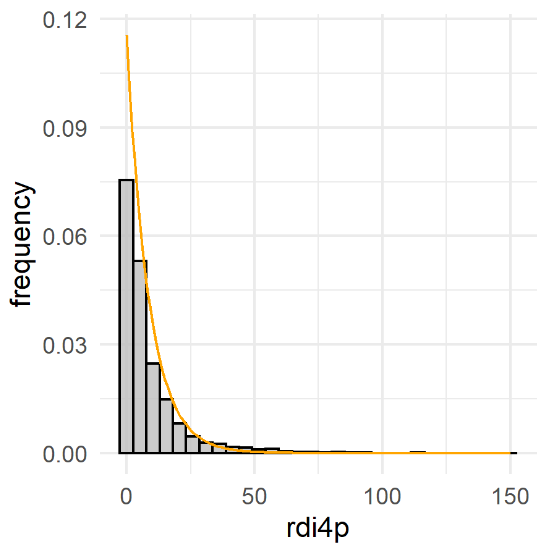


Figure 3.2: Histogram of the respiratory disturbance index (`rdi4p`) together with the density of a fitted exponential distribution (orange) with $\lambda = 0.12$.

the sample statistics are $\hat{x}_{med} = 4.19$ and $\hat{\sigma} = 12.43$. The results for the percentile CI and the accelerated bias-corrected CI are shown in Table 3.2. We see that the function `bcanon` produced **NaN** for the CI of the median. This is because is uses the variance of the Jackknife statistics in the definition of the acceleration coefficient. As the median occurs twice in the data the denominator is zero and the acceleration coefficient is not defined. For SD we see that the upper confidence bound is smaller with the corrected method, whereas the lower bound is roughly the same as in the percentile method. It is surprising that the sample value for the standard deviation is not included in the confidence interval for the second method. The percentile CIs include both statistics. Of course, we don't know the true parameters or distribution here, so we can't judge the real precision here.

	lower_med	upper_med	lower_sd	upper_sd
percentile CI	3.95	4.41	11.80	13.11
acc bias-corr CI	NaN	NaN	11.87	11.95

Table 3.2: Confidence intervals for median and SD using the percentile method (first row) and the accelerated bias-corrected CI using `bcanon` (second row).

4 Survival analysis

4.1 Survivor functions for whole sample

In the fifth exercise we do survival analysis with data from lung cancer patients. The three variables used here are `PRE30`, indicating whether the patient is a smoker, `AGE`, encoding the age at surgery, and `Risk1Y`, indicating whether the patient died within one year after the surgery. The first observations are shown in Table 4.1. Hence, this data can be understood as interval censored survival data. In this exercise we were asked to assume failure times at `AGE`+1, if the patient died and fit survivor functions to model $S(t) = P(T > t)$, where T is a random variable encoding the failure time of an object, here the time of death of a patient.

	PRE30	AGE	Risk1Y
1	TRUE	60	FALSE
2	TRUE	51	FALSE
3	TRUE	59	FALSE
4	FALSE	54	FALSE
5	TRUE	73	TRUE
6	FALSE	51	FALSE

Table 4.1: First observations of the lung cancer patient data set.

To assess which parametric model is reasonable for the data, we use non-parametric estimators for the survival curve. Therefore we draw the Kaplan-Meier estimator, given by

$$\hat{S}_{KM}(t) = \prod_{\{j:\tau_j<t\}} (1 - \frac{d_j}{r_j}),$$

where $0 \leq \tau_1 < \tau_2 < ...$ denote the ordered uncensored failure times (here `AGE`+1), r_i and d_i the number of units still at risk and failures at τ_i respectively. A similar estimator, especially in those areas where still many cases are at risk, is the Fleming-Harrington estimator

$$\hat{S}_{FH}(t) = \prod_{\{j:\tau_j<t\}} \exp(-\frac{d_j}{r_j}).$$

Both are shown in Figure 4.1 where we can see that they only differ obviously at ages > 80. In the area of lower risk both seem to be identical, why we can't see the orange line. We can also see that the confidence band spreads as the survival probability decreases.

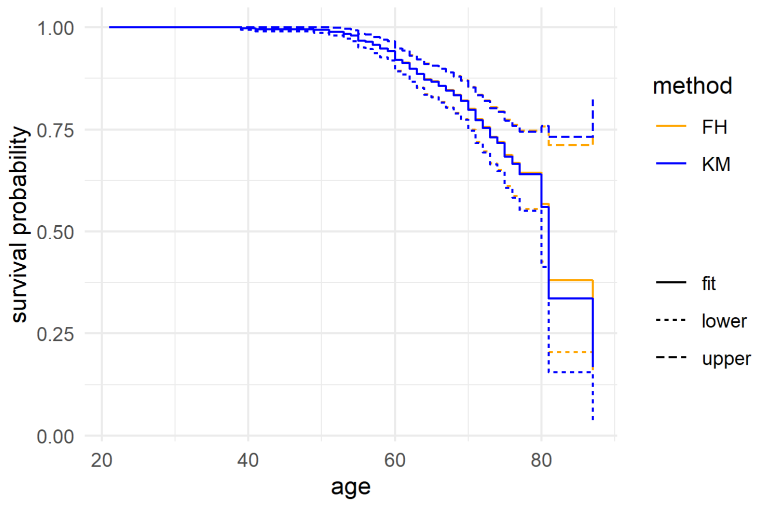


Figure 4.1: Nonparametric estimators for the survivor function. KM = Kaplan-Meier drawn in blue; FH=Fleming-Harrington in orange. Dashed lines show the confidence bands.

Both curves are fitted using `survfit` from the package *survival* which also computes pointwise confidence intervals using Greenwood's formula.

With the function `survreg` we can fit parametric models to the data. Typical models are the exponential, Weibull and log-logistic models. Because we were asked to fit the first two in this exercise, I only give more details on them. The exponential model

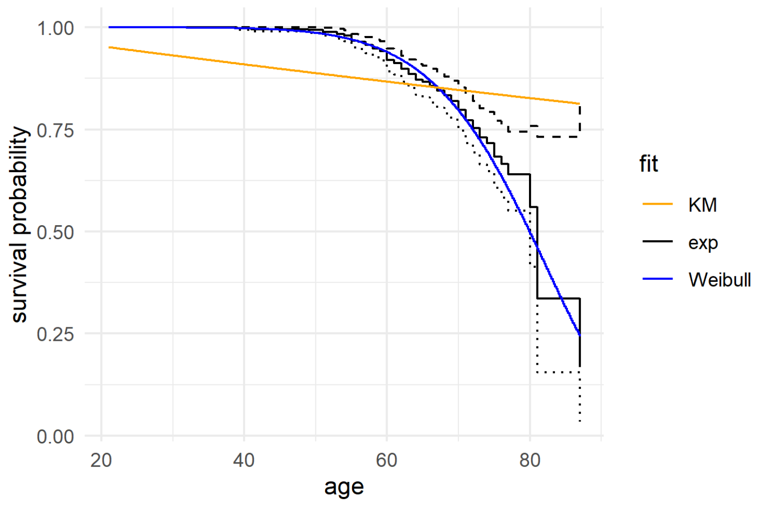


Figure 4.2: KM estimator and parametric fits for the survivor function. Dashed lines represent the confidence band of the KM estimator. Exponential fit in orange; Weibull fit in blue.

assumes a constant Hazard function, i.e. rate of failure, of λ over time resulting in an exponential survivor function $S_{exp}(t) = \exp(-\lambda t)$. With the Weibull distribution we can handle varying Hazard rates by taking $h(t) = \alpha\lambda^{\alpha}t^{\alpha-1}$ resulting in a survivor function $S(t) = \exp(-(\lambda t)^{\alpha})$. At $t = 1$ we have $h(1) = \alpha\lambda^{\alpha}$, so λ is still the parameter of failure rate, defined for the unit time and modified by the shape parameter. Obviously we get the exponential model for $\alpha = 1$, an increasing Hazard function for $\alpha > 1$ and an decreasing Hazard or $0 < \alpha < 1$. The Weibull distribution is often more appropriate when we can't assume constant Hazard rates. In clinical studies concerning relapses of psychological diseases for example a relapse, i.e. failure, gets often less probable with time since recovery. Here a decreasing Hazard function would be appropriate. But in general older living organisms have a smaller survivor probability over time than younger ones indicating the use of an increasing Hazard. In this example the fits (see Figure 4.2) compared to the nonparametric KM estimator suggest that an exponential model can be rejected quite obviously, since the shape does not at all represent the step function. The Weibull model on the other hand seems very good as it follows the KM estimation pretty directly. Although it lies beneath the lower confidence bound at an age around 78, the overall shape suits pretty well.

4.2 Comparison of smokers and non-smokers

Now we split the data set into two parts - smokers and non-smokers - according to the variable `PRE30`. Fitting a KM estimate for each group leads to Figure 4.3. We see very clear that the curve of smokers is stepper than the other. The curve for nonsmokers drops in the end since the only patient with age 81 in the nonsmokers group, who also was the oldest one, died. But all in all the survival probability of nonsmokers seems higher. In the plot we also see that there are more steps in the curve for the smokers. This is because of the unbalanced sample sizes. With 386 smokers and 84 nonsmokers the proportion of smokers is 82.12%. It is well known that smoking increases the probability to get cancer, so this is not a surprising observation here. An inferential comparison of the two groups to check whether the visual differences are statistically significant is done with a log-rank test. With the R function `survdiff` in the same package we get the results $\chi^2 = 2.7$ and $p = .1$. This indicates a difference between the groups, too, which most researchers won't interpret as significant. Since I am no expert in this area, I can't finally judge it, but would emphasize that the difference may be statistically relevant in this case.

Now we want to check whether the assumption of a Weibull model is still justified when we look at the groups separately. Therefore we fitted a Weibull curve for each of the groups (see Figure 4.4). For the group of smokers we see like above a very good fit. For nonsmokers the Weibull fit looks quite well but is probably distorted by the sudden drop down in the end. It seems like a less steep curve would fit better to the data in the beginning. This may be solved by an extended sample where especially more older patient are included. Of course, this is sometimes hard to accomplish in practice. Still the Weibull curve of the smokers group decreases faster than the other and I would

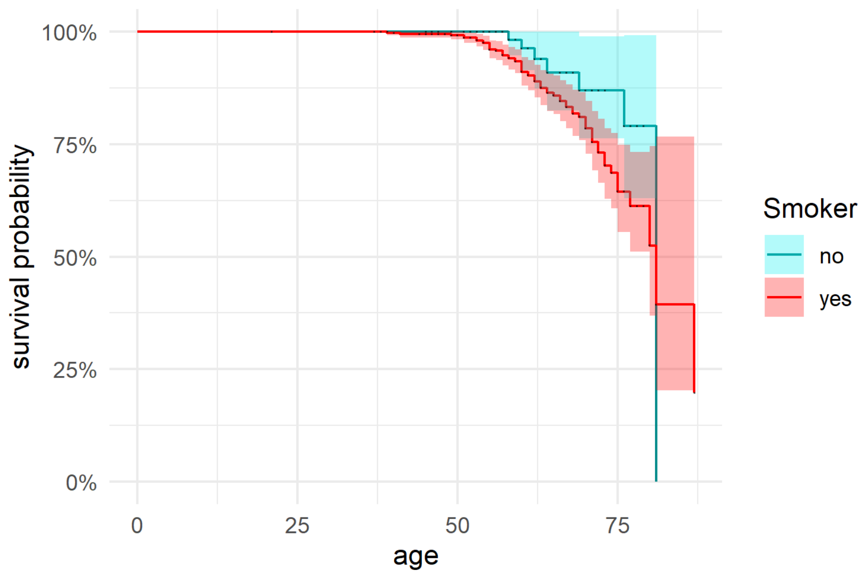


Figure 4.3: Kaplan-Meier estimate for smokers (red) and nonsmokers (blue) with confidence bands.

finally conclude, also because it seems quite reasonable (although a statistician should argue rather with data than with common sense), that smoking decreases the probability of surviving one year after the surgery.

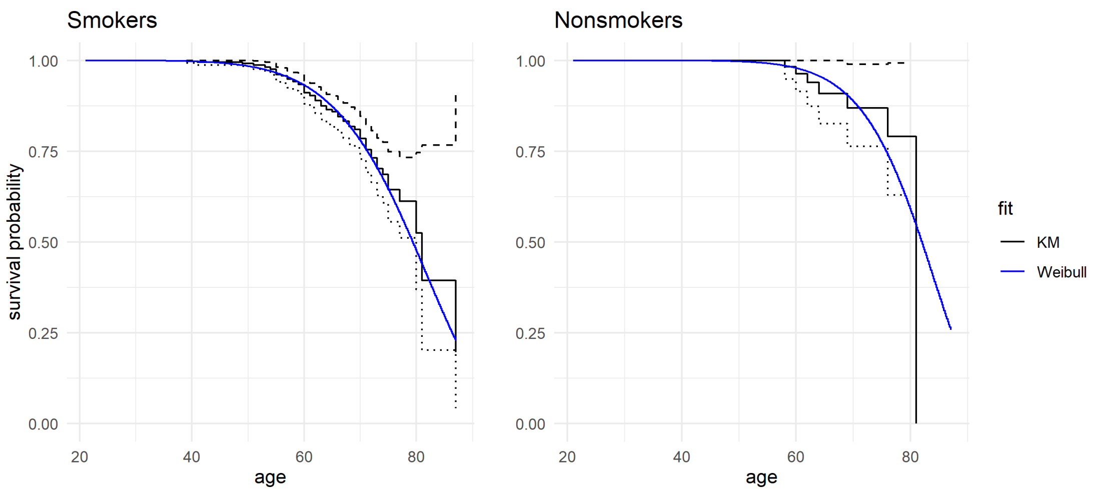


Figure 4.4: Kaplan-Meier estimate with dashed confidence bands (black) and Weibull fit (blue) for patient groups of smokers (left) and nonsmokers (right).

5 Expectation Maximization

5.1 EM-algorithm

Lets first explore the data and let's motivate the reason of why you would use a mixture model by using an example. Suppose we have the following density plot 5.1 and 5.2:

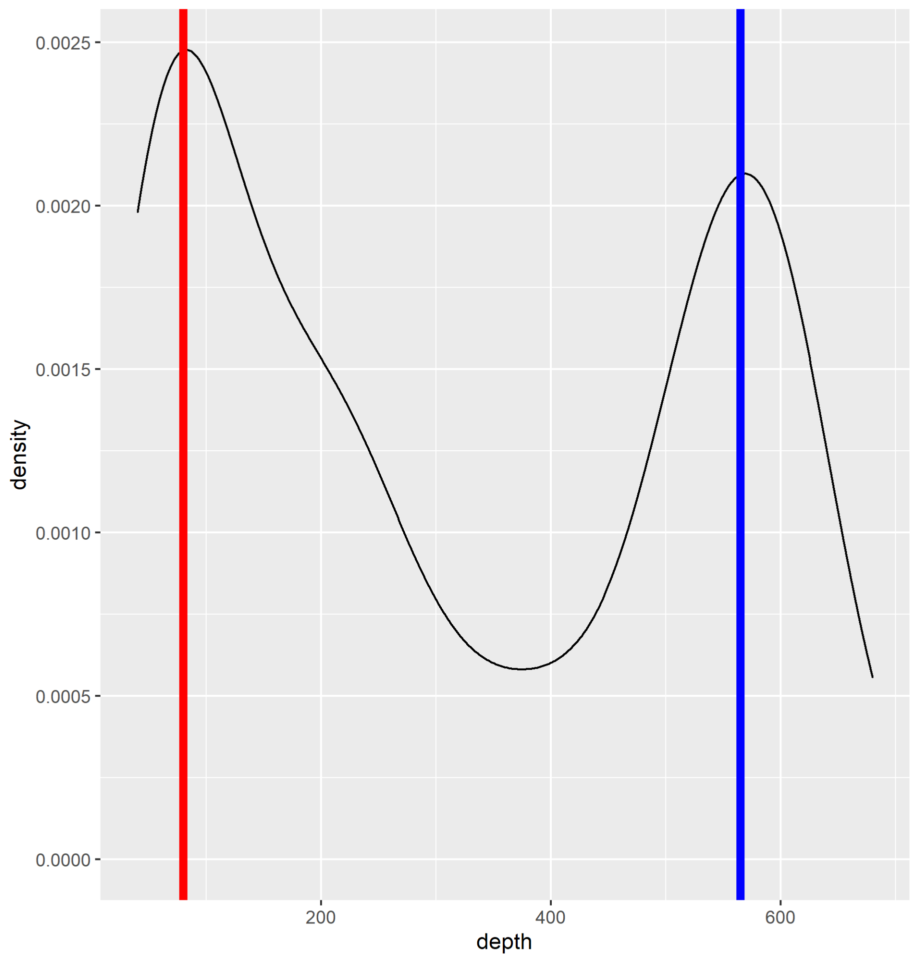


Figure 5.1: Density plot for variable `quakes$depth`

We can immediately see that the resulting distribution appears to be bi-modal (i.e. there are two bumps in both the plots) suggesting that these in case of both the data, they might be coming from two different sources.

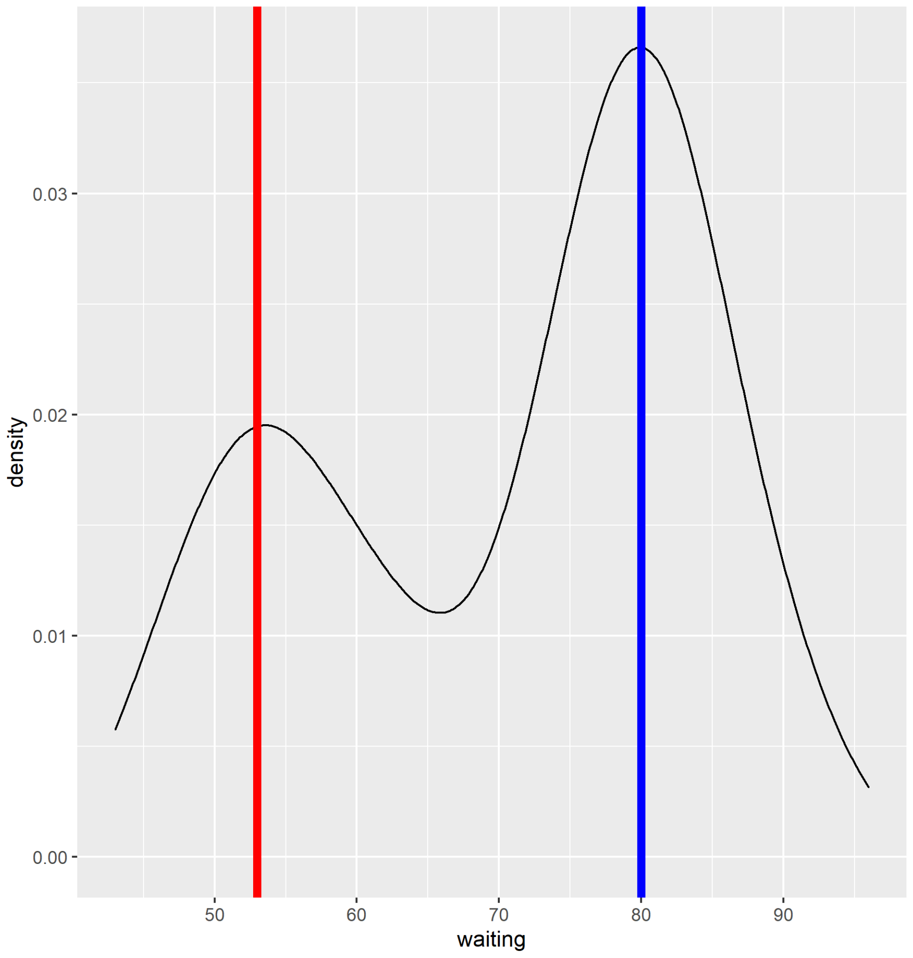


Figure 5.2: Density plot for variable `faith$waiting`

Putting the the `quakes` data into context suggests that the earthquakes in Fiji may be coming from two different sub-populations, depending upon the depth of the region. Similarly, putting the the `faithful` data into context suggests that the eruption times may be coming from two different sub-populations. There could be several reasons for this. For instance, maybe at different times of the year the geyser eruptions are more frequent. We can probably take an intuitive guess as to how we could split this data.

For instance, we can observe in Figure 5.1, there likely is a sub-population with a mean depth of ~ 80 with some variance around this mean (red vertical line in figure 5.1.) Another population with a mean depth of ~ 565 with again some variance around this mean (blue vertical line in figure 5.1). Similarly, as we can observe in Figure 5.2, there likely is a sub-population with a mean eruption of ~ 53 with some variance around this mean (red vertical line in figure 5.2.) Another population with a mean eruption of ~ 80 with again some variance around this mean (blue vertical line in figure 5.2).

5.2 Gaussian Mixture Model

What we have done is a naive attempt at trying to group the data into sub-populations or clusters. But surely there must be some objective and "automatic" way of defining these clusters? This is where mixture models come in by providing a "model-based approach" to clustering through the use of statistical distributions.

A mixture model consist of a mixture of distributions. The first thing we need to do when performing mixture model clustering is to determine what type of statistical distribution we want to use for the components.

In our exercise we will use one of the most common statistical distributions used for mixture model clustering which is the Gaussian/Normal Distribution. When Gaussian distributions are used for mixture model clustering, they are referred to as *Gaussian Mixture Models (GMM).* As it turns out, our earlier intuition on where the means and variance of the sub-population in the plot above is a perfect example of how we could apply a GMM. Specifically, we could try to represent each sub-population as its own distribution (aka. mixture component). The entire set of data could then be represented as a mixture of 2 Gaussian distributions (aka. 2-component GMM)

The procedures is based on the iterative *expectation maximization (EM) algorithm.* The following two points are important to note here. First, the EM algorithm is an iterative procedure, and the time required for it to reach convergence, if it converges at all depends strongly on the problem to which it is applied. The second key point is that because it is an iterative procedure, the EM algorithm requires starting values for the parameters, and algorithm performance can depend strongly on these initial values.

In our exercise, for both the data sets, we are going take initial values,

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

where m is the mean of all observation and sd the standard deviation.

Observe the following plots 5.3 and 5.4. We have built a 2-component GMM. So how do we interpret figures 5.3 and 5.4? It's actually quite simply, the red and blue lines simply indicate 2 different fitted Gaussian distributions.

5.3 Stopping criteria

In the EM-algorithm we have used the (log-)likelihood function, which is a non-decreasing sequence. Also, is strictly increasing unless $\theta^{(k)} = \theta^{(k-1)}$. We can observe this fact for the two data sets.

We can this be used to formulate a stopping criterion for the EM-Algorithm. We can decide a threshold on $\theta^{(k)} - \theta^{(k-1)}$, if it is below the threshold then, we can stop the algorithm. For eg. we have take the threshold as 0.00001. In the case of `quakes$depth` algorithm stopped after 12 iterations, where as in the case of `faith$waiting` it stopped after 9 iterations. This just assures that the improvement in the maximization step is small and the algorithm can be stopped.

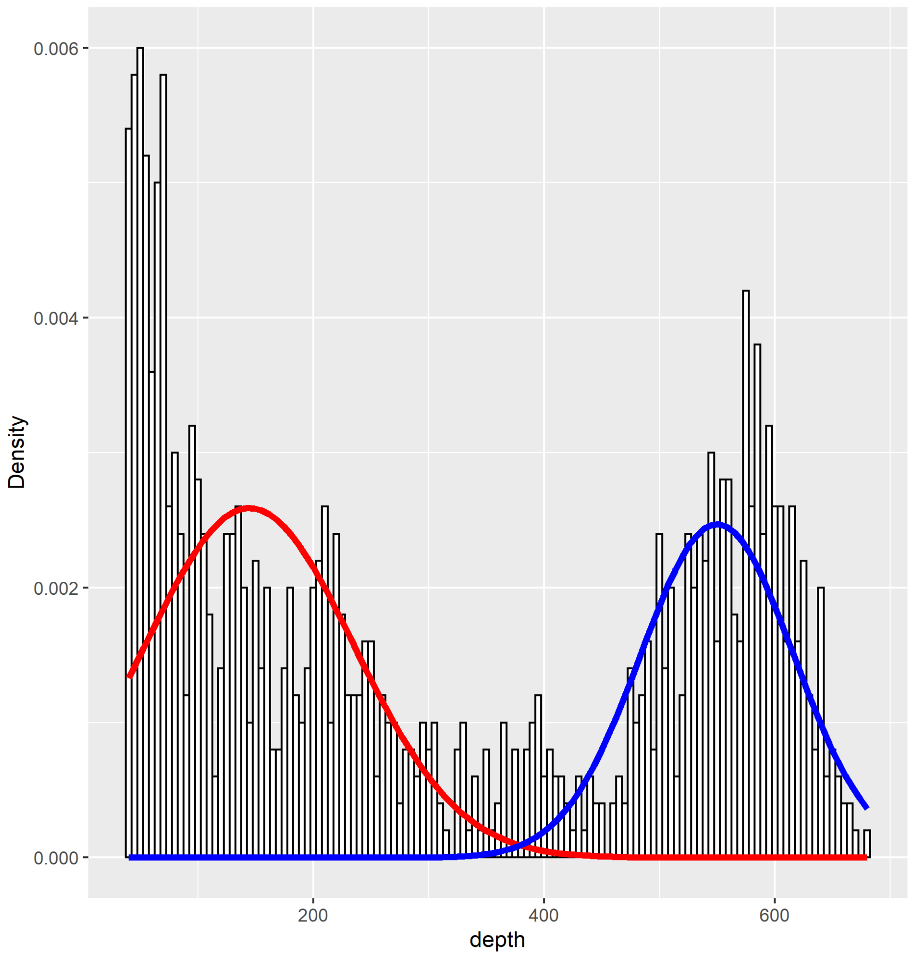


Figure 5.3: Density plot for GMM with barplot for `quakes$depth`

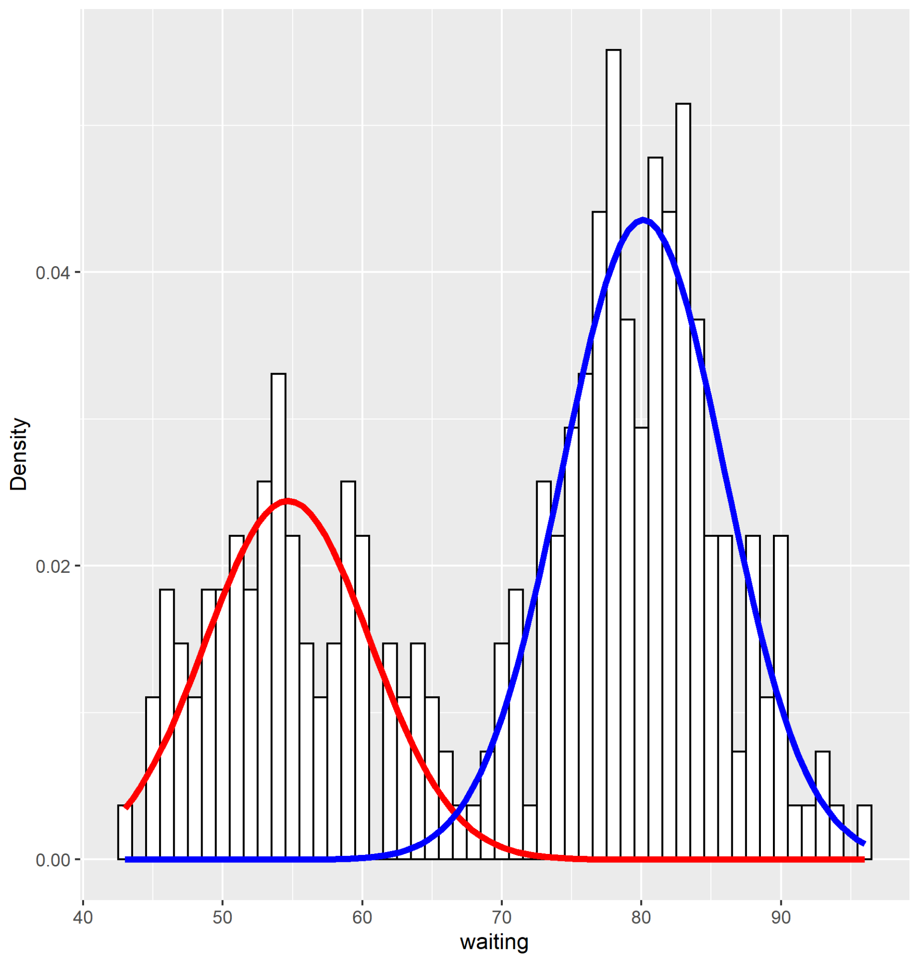


Figure 5.4: Density plot for GMM with barplot for `faith$waiting`

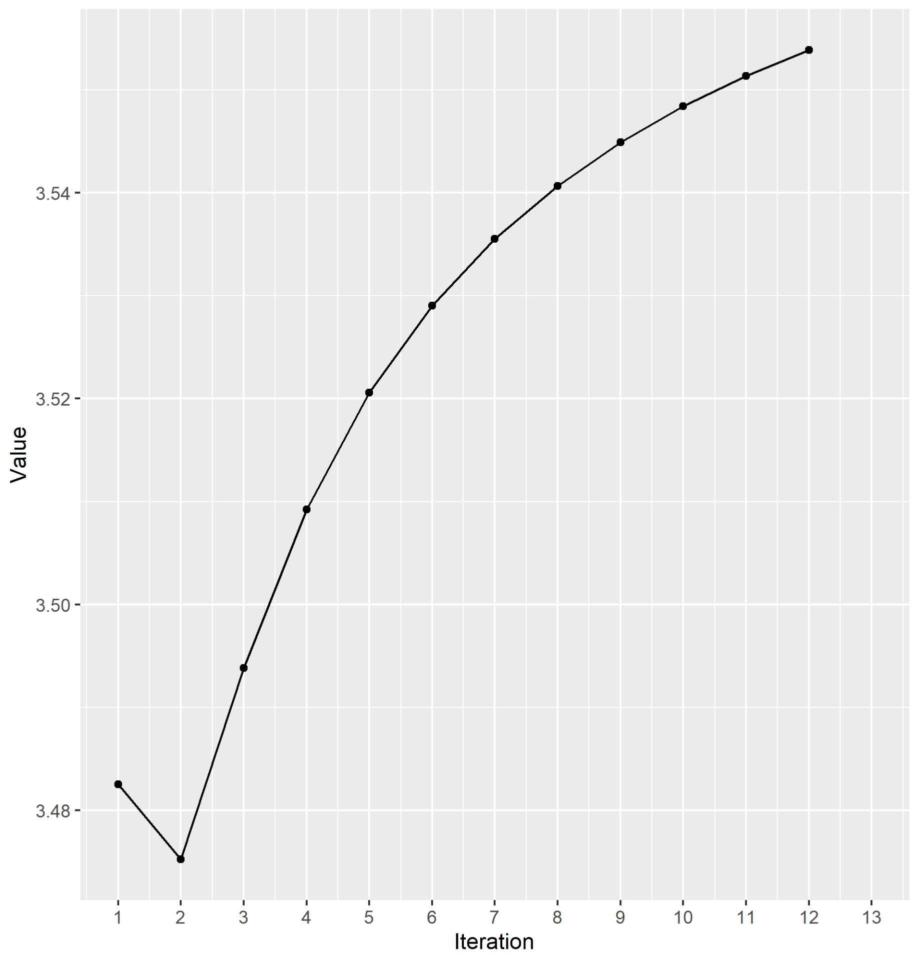


Figure 5.5: Values of the log-likelihood function per iteration for `quakes$depth`

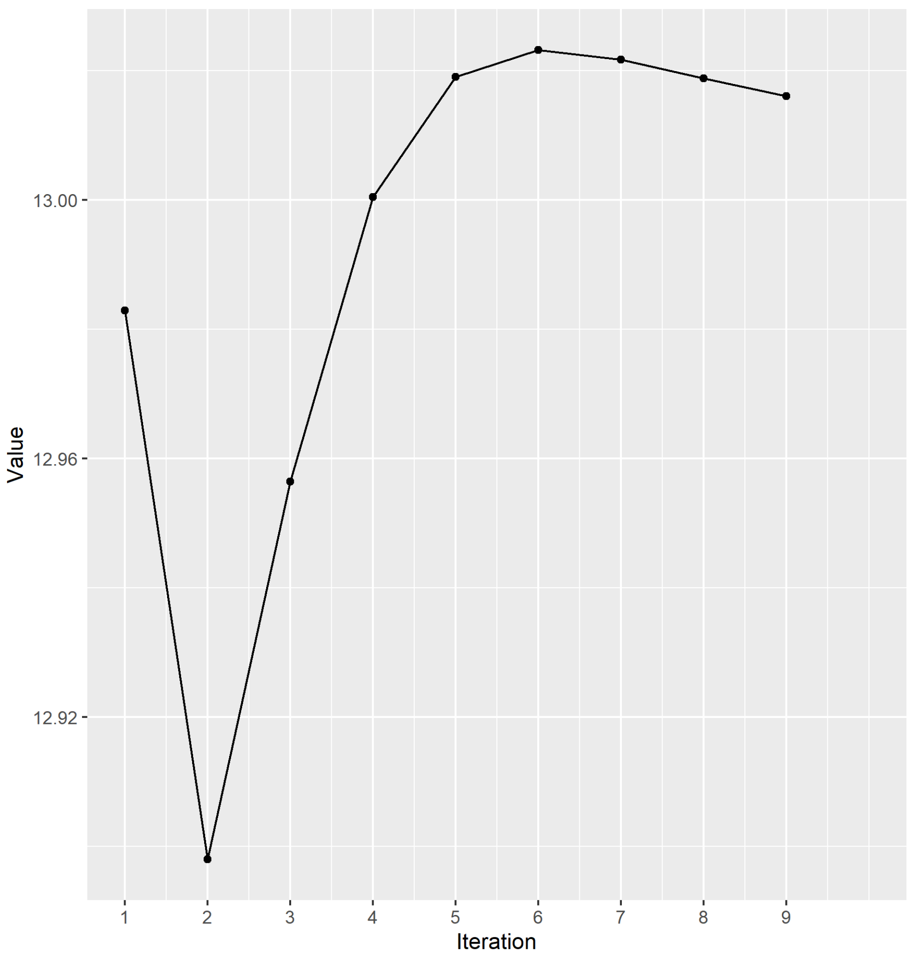


Figure 5.6: Values of the log-likelihood function per iteration for `faith$waiting`

6 Extreme Value Theory

7 Generalised Linear Models

7.1 Examine distribution of G1, G2 and G3

In this exercise we try to explain grades in mathematics of students by other predictive variables using linear models. First we have a look at the distribution of the dependent variables, here the first, second and third grade (G1, G2 and G3) during the year, to check whether a normal assumption is justified and we can use a standard linear model or whether we have to use a generalised linear model. Therefore we plot a histograms and QQ-Plots. Because the Poisson distribution seems suitable, too, for the data, we include a QQ-Plot for the normal distribution and the Poisson distribution (see Figure 7.1).

Although the distributions look very similar, especially for G1 and G2, there are already some theoretical arguments that speak for the assumption of a Poisson distribution rather than a normal. First, the data is discrete, whereas the normal distribution has a density. Additionally the normal distribution allows negative numbers, too, whereas the Poisson distribution - although the values range trough $\mathbb{N}$ - only realises positive integer values. Still the data ranges only from 0 to 20, which is difficult to model reasonable with generalised linear models anyway. In the concrete data we see, that - especially for G3 - the Pisson distribution fits better to the data in the histogram. For G1 we see a rather good fit for both distributions. In the QQ-Plot we see that the sample distribution has shorter tails than the normal visible in the slightly S-curved shape of the plot, showing us that the sample quantiles at the border don't increase as fast as the theoretical at the right tail of the distribution an don't decrease that fast at the lower tail respectively. This pattern is less distinct for the Poisson distribution. Otherwise this would be a sign for underdispersion of the sample. The Fano factor F is defined as the ratio of the variance over the mean of a distribution and equals 1 for a Poisson distribution. It is shown with other summary statistics in 7.1.

timepoint	min	q25	med	M	q75	max	IQR	SD	F
G1	3	8	11	10.91	13	19	5	3.32	1.01
G2	0	9	11	10.71	13	19	4	3.76	1.32
G3	0	8	11	10.42	14	20	6	4.58	2.02

Table 7.1: Summary statistics for the grade (G1, G2, G3) in mathematics of the data set. Here F is the Fano factor.

In the table we see that the Fano factor is near to one for G1 indicating that is there is no underdispersion. For G2 and G3 there is the problem of the high amount of 0 in the

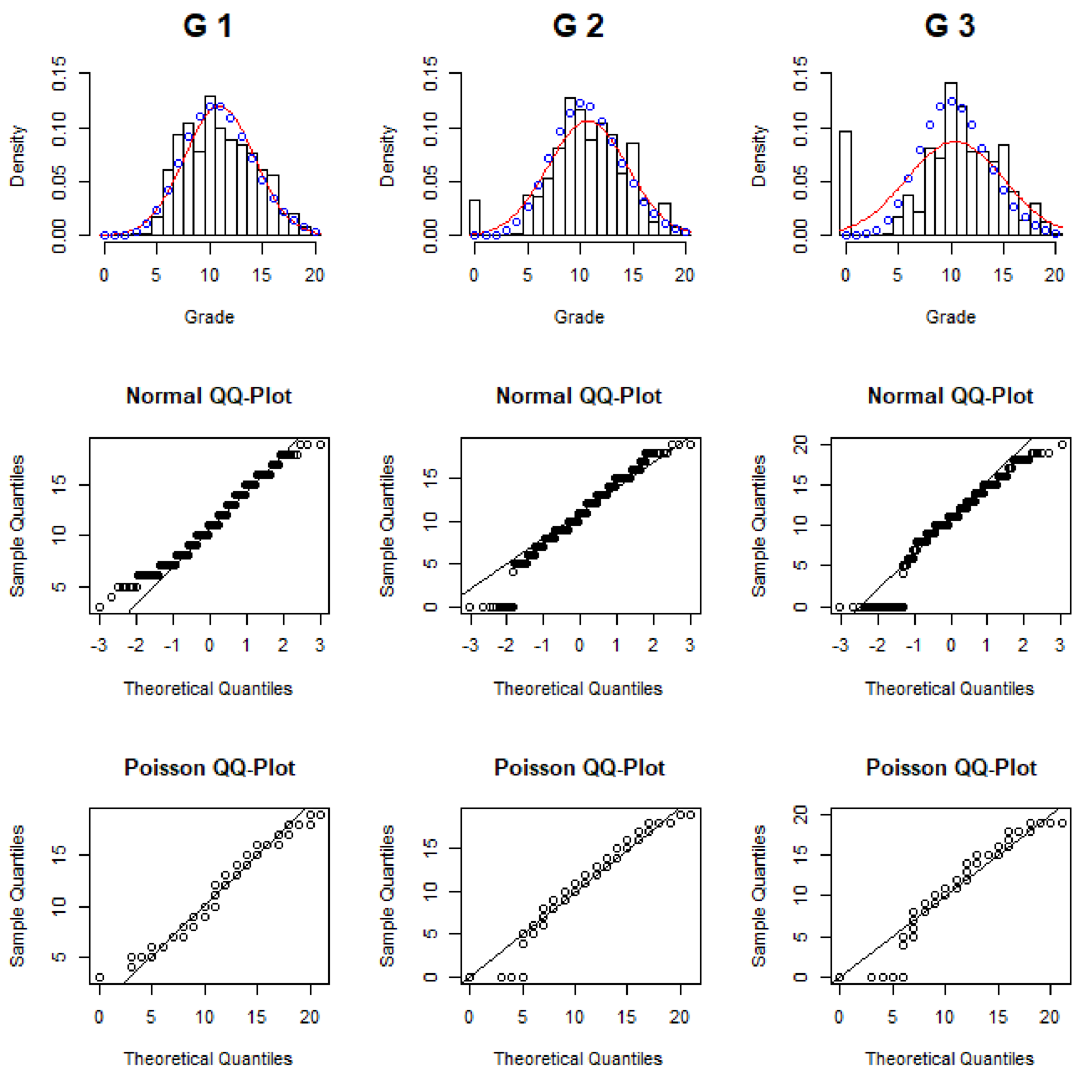


Figure 7.1: Histogram (first row) and QQ-Plots using Normal (second row) and Poisson (third row) distribution for theoretical quantiles for G1, G2 and G3. Red line represents the density of a fitted normal distribution and blue points a Poisson distribution, each with parameters estimated from the sample.

data, visible in the histogram and in the QQ-Plots as the first quantiles are all at 0. For G2 this is not that bad and the assumption of a Poisson distribution seems still justified because the rest of the data suits quite well. But also the normal distribution seems not far from the data. In the QQ-Plot we see that the points form a steeper curve compared to the reference line. This may be caused by the many zeros that don't suit to any of the both distributions. This is even more drastically for the third time point. Because the pattern is so much distorted, I decided to include the same plots for G3 where I dropped the zeros (see Figure 7.2). There we see a much better in the histogram as well as in the QQ-Plots. Now a similar pattern as for G1 is visible. An underdispersion is also visible in the QQ-Plot for the Poisson distribution. The Fano factor for the reduced sample is now 0.90 indicating a slight underdispersion compared to a Poisson distribution, too. Without dropping zeros the Fano factor was 2.02 indicating a high overdispersion, which is quite obvious from the visualisations. Also for G2 the overdispersion was confirmed by a smaller Fano factor of 1.32, which is still greater than the optimum at 1. Summing up I would conclude that a Poisson distribution is theoretical more suitable and that the data follow its shape well enough to use generalised linear models.

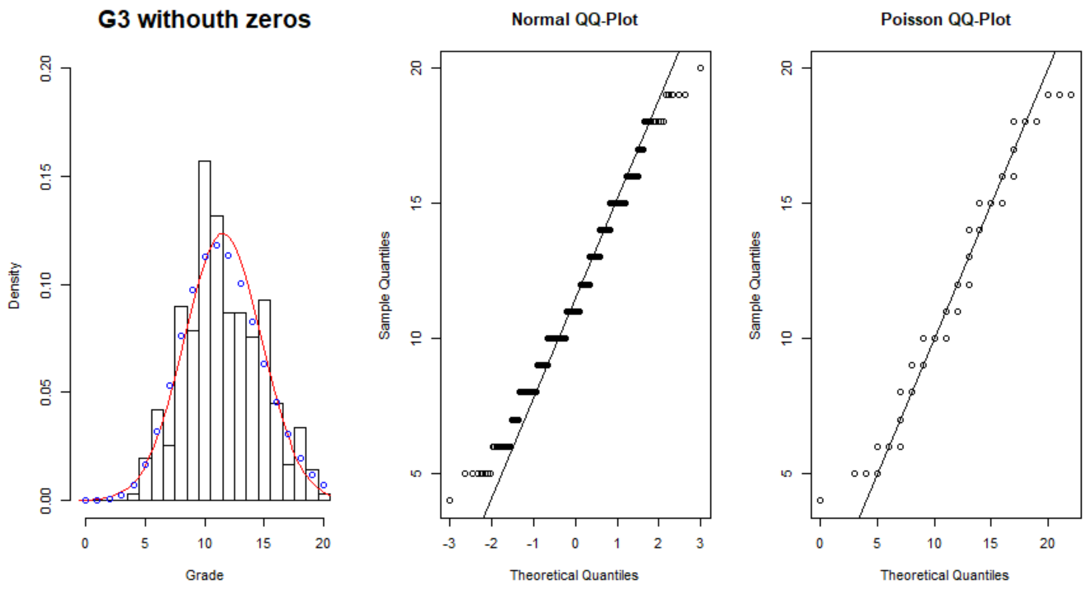


Figure 7.2: Histogram and QQ-Plots for normal and Poisson distribution with sample parameters for G3 without zeros. The red line represents the fitted normal curve and blue points the Poisson distribution with sample mean.

7.2 Model fit for G1

Now we try to fit a suitable model to predict he first grade G1 from other variables. Those include for example sex, age and education and job of father and mother. With the function `glm` and the argument `family='poisson'` we fit a generalised linear model with a Poisson distributed response. The design matrix is built with all available independent variables. The reduced (I dropped some not significant variable in the list) output of the `summary` for the resulting `glm` object looks as follows:

```
> summary(model1)

Call:
glm(formula = fmla, family = "poisson", data = dat)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.38057  -0.64611  -0.01502   0.50274   2.01602

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       2.418107   0.331755   7.289 3.13e-13 ***
schoolMS          0.007157   0.058426   0.122  0.90250
sexM              0.078237   0.036434   2.147  0.03176 *
age              -0.006480   0.015983  -0.405  0.68515
addressU          0.012823   0.043630   0.294  0.76882
famsizeLE3        0.036639   0.035590   1.029  0.30325
PstatusT          0.016047   0.053050   0.302  0.76227
Medu              0.010664   0.023797   0.448  0.65405
Fedu              0.014886   0.020349   0.732  0.46446
Mjobhealth        0.078652   0.081444   0.966  0.33419
[...]
traveltime       -0.003407   0.025788  -0.132  0.89488
studytime         0.053887   0.020846   2.585  0.00974 **
failures         -0.147644   0.028151  -5.245 1.56e-07 ***
schoolsupyes     -0.211886   0.052610  -4.027 5.64e-05 ***
famsupyes        -0.093311   0.035213  -2.650  0.00805 **
paidyes          -0.008676   0.035007  -0.248  0.80426
[...]
freetime          0.023120   0.017435   1.326  0.18482
goout            -0.037487   0.016743  -2.239  0.02516 *
Dalc             -0.002047   0.025074  -0.082  0.93494
Walc             -0.004639   0.018663  -0.249  0.80371
health           -0.015492   0.011868  -1.305  0.19178
absences          0.001520   0.002185   0.696  0.48662
---
Signif. codes:  0 ‘***‘ 0.001 ‘**‘ 0.01 ‘*‘ 0.05 ‘.‘ 0.1 ‘ ‘ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 402.47  on 394  degrees of freedom
Residual deviance: 263.08  on 355  degrees of freedom
AIC: 2000.2

Number of Fisher Scoring iterations: 4
```

There we see first some distributional measures for the deviance residuals. Let's have look at the coefficients. We see for each independent variable the estimate, standard error and z-value. In the last column there is the p value. This indicated whether we consider the predictive value of a variable as significant or not. R indicates the usual threshold with the significance codes. Using this suggestion we see that most of the variables don't provide much predictive information about the grade. Just sex (`sexM`), weekly study time (`studytime`), number of past class failures (`failures`), extra educational support in school (`schoolsupyes`) and in the family (`famsupyes`) as well as the time a student goes out with friends (`goout`) are significant. Here a negative estimate indicates a negative influence for higher values or the level of the factor respectively. Male students (`sexM`) for example get higher grades (estimate 0.08) than female. Beneath the table for the coefficients we see the deviances. Testing the full model against the null model is quite needless here, since the full model will have a better fit to the data. Although we can check this very fast using the `anova` function in R after fitting the null model. The difference of deviances is 139.39 and thus significant ($p < .01$). A more interesting comparison to a nested model will be discussed in the next section.

To examine the adequacy of the model we also have to look at the distribution of the residuals. There are many different approaches. The Pearson and Anscombe residuals are shown in Figure 7.3. An appropriate model will show normally distributed residuals. The Pearson residuals have to be taken with care, since for non-normal responses they are skewed. Therefore Anscombe resdiuals use a normalising transformation. Although both residual distribution look a little bit skewed in the histogram they show no obvious deviations in the QQ-Plot. The scatter plot on the left shows no systematic dependence from the height of the fitted values. All in all we may expect our model is a good choice for the data.

7.3 Reducing the model for G1

We saw in the last section that most of our variables were not significant for the prediction in our full model. Therefore we fitted a reduced model containing the variables that were significant in the first model. Additionally we included the factor `Fedu` encoding the educational level of the father. The summary looks like follows:

```
> summary(model2)

Call:
glm(formula = G1 ~ sex + Fedu + studytime + failures + schoolsup +
    famsup + goout, family = "poisson", data = dat)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.70146  -0.70149  -0.02238   0.58681   2.55092

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   2.34585    0.07578  30.955  < 2e-16 ***
```

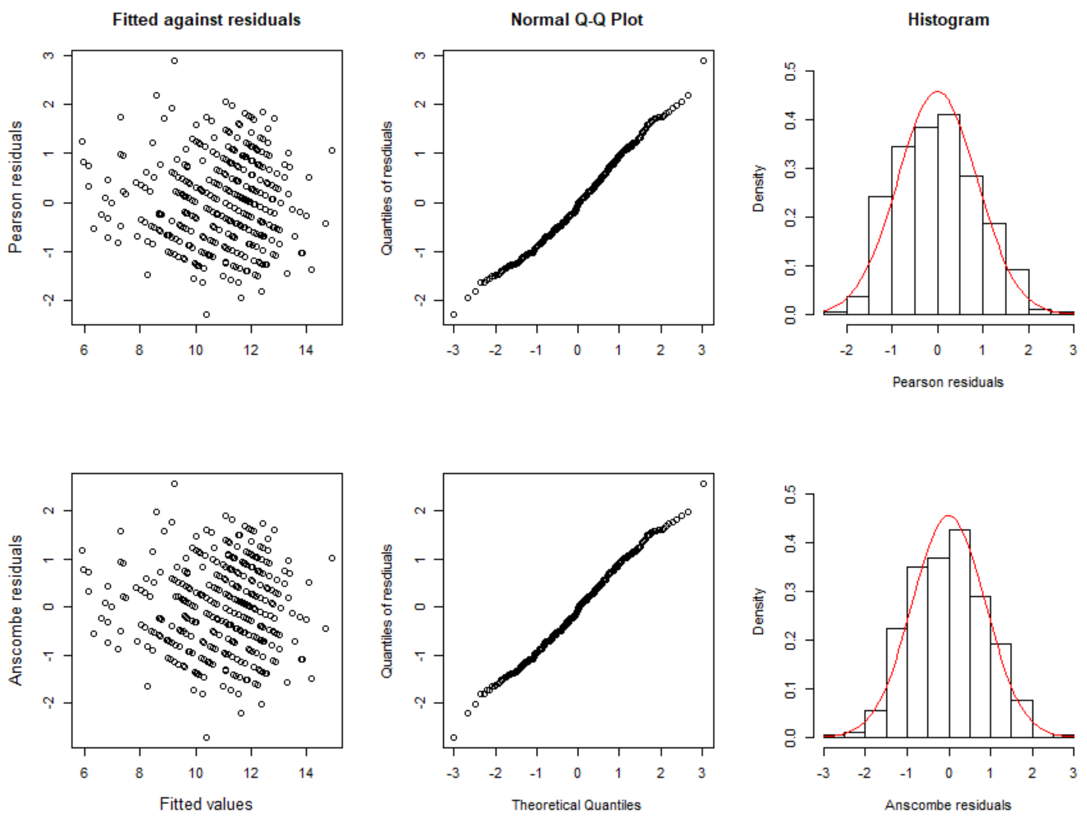


Figure 7.3: Scatter plot for fitted values against residuals, QQ-Plot and histogram with fitted normal curve in red for Pearson residuals (first row) and Anscombe residuals (second row).

```
sexM          0.06585    0.03237   2.034  0.04193 *
Fedu          0.04281    0.01482   2.888  0.00388 **
studytime     0.05828    0.01906   3.057  0.00223 **
failures     -0.13876    0.02495  -5.561 2.69e-08 ***
schoolsupyes -0.19834    0.04978  -3.984 6.78e-05 ***
famsupyes    -0.07330    0.03240  -2.263  0.02365 *
goout        -0.03525    0.01406  -2.506  0.01220 *
---
Signif. codes:  0 ‘***‘ 0.001 ‘**‘ 0.01 ‘*‘ 0.05 ‘.‘ 0.1 ‘ ‘ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 402.47  on 394  degrees of freedom
Residual deviance: 302.02  on 387  degrees of freedom
AIC: 1975.1

Number of Fisher Scoring iterations: 4
```

At first glance it may seem surprising that the variable `Fedu` is now significant. But since the whole model changes the estimates and standard errors for specific covariates change, too. Now we have a highly reduced model with only significant explaining variables. Compared to `model1` the residual deviance is of course higher. But the AIC, which is a criterion for the goodness-of-fit, which accounts for the size of the residuals as well as the number of coefficients to estimate in a model, decreased, indicating a better fit for `model2`. The residuals for `model2` follow a normal distribution, too. I omit the visualisation here since it is quite similar to the one in the last section, but it can be found in the supplementary code. Since the second model is nested in the full model we can compare them with an analysis of deviance. Like mentioned above this is easily done with the function `anova` and the argument `test='Chisq'`. With 32 degrees of freedom and a difference of 38.94 in the sum of residual deviances we get $p = .19$ indicating that the additional explaining value of the full model is not significant. Thus we can stick to the reduced model.

A third model was fitted by dropping the variable `goout` and adding the variable `Walc` again:

```
> summary(model3)

Call:
glm(formula = G1 ~ sex + Fedu + studytime + failures + schoolsup +
    famsup + Walc, family = "poisson", data = dat)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.64853  -0.69869  -0.02535   0.61090   2.51693

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   2.31120    0.07204  32.083  < 2e-16 ***
sexM          0.07558    0.03296   2.293  0.02183 *
Fedu          0.04067    0.01477   2.753  0.00591 **
studytime     0.05231    0.01935   2.704  0.00685 **
failures     -0.14110    0.02488  -5.671 1.42e-08 ***
schoolsupyes -0.20115    0.04985  -4.035 5.46e-05 ***
famsupyes    -0.07447    0.03239  -2.299  0.02148 *
Walc         -0.02614    0.01274  -2.052  0.04016 *
---
Signif. codes:  0 ‘***‘ 0.001 ‘**‘ 0.01 ‘*‘ 0.05 ‘.‘ 0.1 ‘ ‘ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 402.47  on 394  degrees of freedom
Residual deviance: 304.07  on 387  degrees of freedom
AIC: 1977.2

Number of Fisher Scoring iterations: 4
```

There we see again that all covariates are significant and the residuals don't show serious deviations from the normal distribution (see code). Now, how can we tell, whether `model2` or `model3` is better? We can't use analysis of deviances here, since the two models are not nested. Therefore we have to keep at other measures, like penalised goodness-of-fit criteria (e.g. the AIC and BIC). Like mentioned above, a smaller AIC indicates a better fit. Since the two models estimated the same amount of coefficients, we don't need to use those but can directly look at the residual deviance or the likelihood. With a sum of 302.02 the second model is a bit better than the third one with 304.07. This is also visible in the difference of the log-likelihood. It is -979.57 for the second and -980.60 for the third model respectively. So `model2` has the higher likelihood and should hence be our choice. Still we have to keep in mind that this comparison is not inferential but descriptive.

8 Kernel density estimation

8.1 Bandwidth and Kernel function

In this chapter we are dealing with marks secured by students in various subject, along with explanatory variables about the students viz., gender, ethnicity etc.. For our analysis for student's performance together with scores in mathematics, reading and writing tests, we consider students attendance for the test preparation course. We drop rest of the variables. Following is the sample for first few entries of the dataset;

	test preparation course	math score	reading score	writing score
1	none	72	72	74
2	completed	69	90	88
3	none	90	95	93
4	none	47	57	44
5	none	76	78	75
6	none	71	83	78
7	completed	88	95	92
8	none	40	43	39
9	completed	64	64	67
10	none	38	60	50

In this exercise we are suppose to implement kernel density estimation. The data entries i.e. math score are integers variables. Kernel density estimation help us visualize the "shape" of data, even though its discrete. Essentially it is sort of continuous replacement for the discrete histogram. Kernel estimation uses a weighted sum of observations dependent on their distance from the variable. The precise definition is as follows, for the given data $X_1, X_2, ..., X_n$, a kernel density estimator is

$$\widehat{f}_{n,h}(x) = \frac{1}{nh} \sum_{1=j}^{n} K\left(\frac{x - X_j}{h}\right),\ x \in \mathbb{R}.$$

where $K : \mathbb{R} \to \mathbb{R}$, such that $\int_{-\infty}^{\infty} K(x)dx = 1$ is known as kernel and $h > 0$ is called the bandwidth. h is smoothing parameter which basically governs how many distinct observations are taken into account around certain location. Thus it has a strong influence on the resulting estimate. Examples of some classical kernels is illustrated below in Figure 8.1.

To visualise and examine the influence of the choice of bandwidth and kernel we have considered the math score variable from the data. First we fit a kernel density

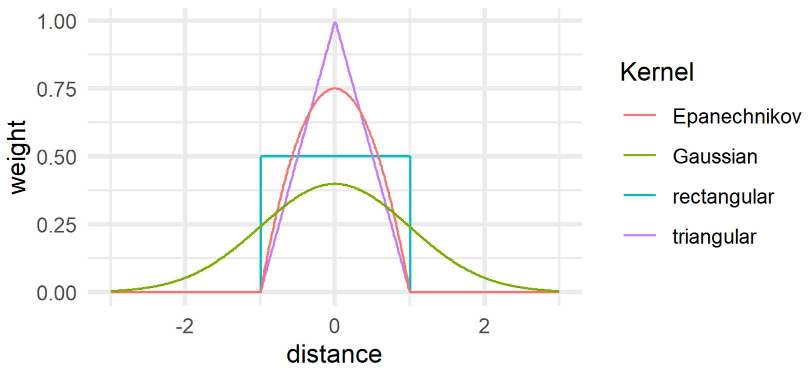


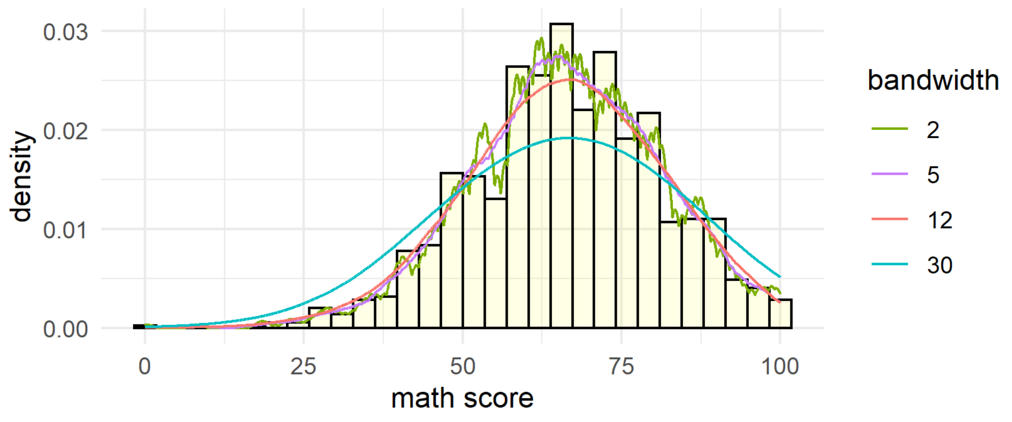
Figure 8.1: Classical examples of kernels often used for density estimation.

with different bandwidths using the Epanechnikov kernel (See Figure 8.2a) and then use different kernels with a fixed bandwidth Figure8.2b). We can clearly see the influence of the choice of bandwidth. The estimated curves get smoother with higher and higher h. For bandwidth $= 2$ it fluctuates highly. For the values of h between 5 and 12 seems reasonable where as for for higher values, like 30 density curve gets too smooth and it certainly does not represent the shape of the data distribution anymore. Since, visually it appears that the most reasonable bandwidth lies between 5 and 12, I have fixed $h = 8$ for the next analysis (See Figure 8.2b). We will now compare different kernels choices with a fix bandwidth $h = 8$.

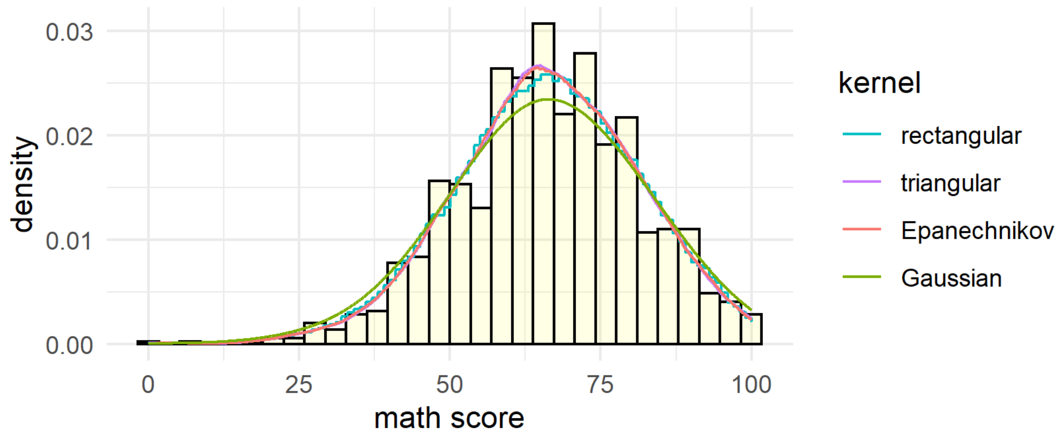
If we observe closely Figure 8.1, we can see that, the different shapes of the kernels shows how observations with certain weights at different distances influence the height of the density estimation at specific points. Therefore, the shape and some basic properties of the resulting density estimations depend highly on the choice of the kernel. The rectangular kernel for example will always result in a piece-wise constant estimation, which is not continuous. The triangular kernel will give us a piece-wise linear density and the Gaussian kernel will result in smooth estimations. In Figure 8.2b we can see that for the rectangular kernel the result is a step function, where as the rest of the kernels seem smoother. But as mentioned before the curve for the triangular kernel is piece-wise linear. Rectangular, triangular and Epanechnikov look quite similar with respect to the shape of the overall distribution. The Gaussian kernel produces a flatter curve that puts more mass to the tails. This might be because, of the differences in support of kernels. The support of the Gaussian kernel $\mathbb{R}$ and its $[-1, 1]$ for the other kernels. Thus, as we can also see in Figure 8.1 Gaussian kernel puts more mass to the tails.

8.2 Cross validation for optimal bandwidth

Now we want to consider a more formal approach to examine the best bandwidth. Above we just did a graphical examination. To get the best fitting curve a common approach is trying to minimise the mean integrated square error (MISE) for the estimator $\hat{f}_{n,h}(x)$



(a) Epanechnikov kernel with different bandwidths.



(b) Different kernels with fixed bandwidth $h = 8$.

Figure 8.2: Kernel density estimations for the math score (first plot) and kernels (second plot).

of $f(x)$, which is given by

$$MISE(\hat{f}_{n,h}) = E\left[\int \left(\hat{f}_{n,h}(x) - f(x)\right)^2 dx\right].$$

Because the real f is unknown one minimises an unbiased estimator of the h-dependent part of the MISE, called the cross-validation criterion:

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

Implementing a function computing the CV for a given estimation we can use the minimising $h_{CV} = \operatorname{argmin}_{h>0} CV(h)$. Afterwards we compared the results with the functions

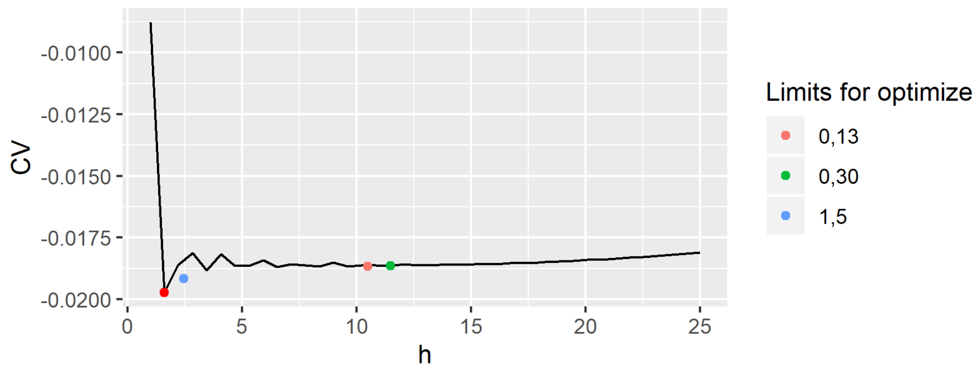


Figure 8.3: CV values for 40 different bandwidths. Red point shows minimum of the curve. Other points show the results for the `optimize` function in R with different limits (see legend).

`bw.ucv` and `bw.bcv` of the package `density`, which compute the optimal bandwidth for an unbiased (`ucv`) or biased (`bcv`) CV implementation for Gaussian kernels. Therefore I used the Epanechnikov kernel as well as the Gaussian kernel in my implementation to compare all results for the maths score. To find the minimizer of my function, I used the R function `optimize`. Before comparing the optimisation of my own CV function a little note about the disadvantages of using `optimize` here is important. To get a first idea of the shape of CV for different h as a curve I took 40 equidistant points in the interval $[1, 25]$ - because it seemed as a (maximal) reasonable region for bandwidths in this case - and computed CV there. Afterwards I took the minimum out of this observations as first reference point ($h = 1.62$). After using the R implemented optimisation with limits 0 and 30, I was surprised by the result ($h = 11.50$). As `optimize` just searches for local minima and starting points for the iteration are chosen as a golden section distance between the given limits, the result also depends highly on those borders. Obviously CV is a strongly fluctuating function as I got different results for every trial, none of them was near the first, say, 'scanned minimum'. Figure 8.3 shows the results. All in all, without an analytical examination of CV as a function one cannot be sure to really found the optimal bandwidth or even to be close to it. In the following I won't discuss this issue any more- I will use a combined strategy by first scanning the interval $[1, 20]$ with 40 points to get the area of small values visually and by taking the minimum. Afterwards I will use `optimize` limited to this region to get a local minimum. This should in total result in a somehow precise and at least acceptable minimum. The results for all three tests and the different methods are shown together in Table 8.1. For the math score and the best bandwidth with a Epanechnikov kernel we use the above optimal $h = 1.62$.

First, observe (see Figure 8.4) that for the Gaussian kernel the problem of fluctuating CV function is not that worse but the usage of the `optimize` function still improves the optimal bandwidth a bit. For the Epanechnikov kernel we have the same problem for every test. All in all the CV curves look very similar for math, reading and writing score. Therefore we get quite similar results for optimal bandwidths. As expected the values

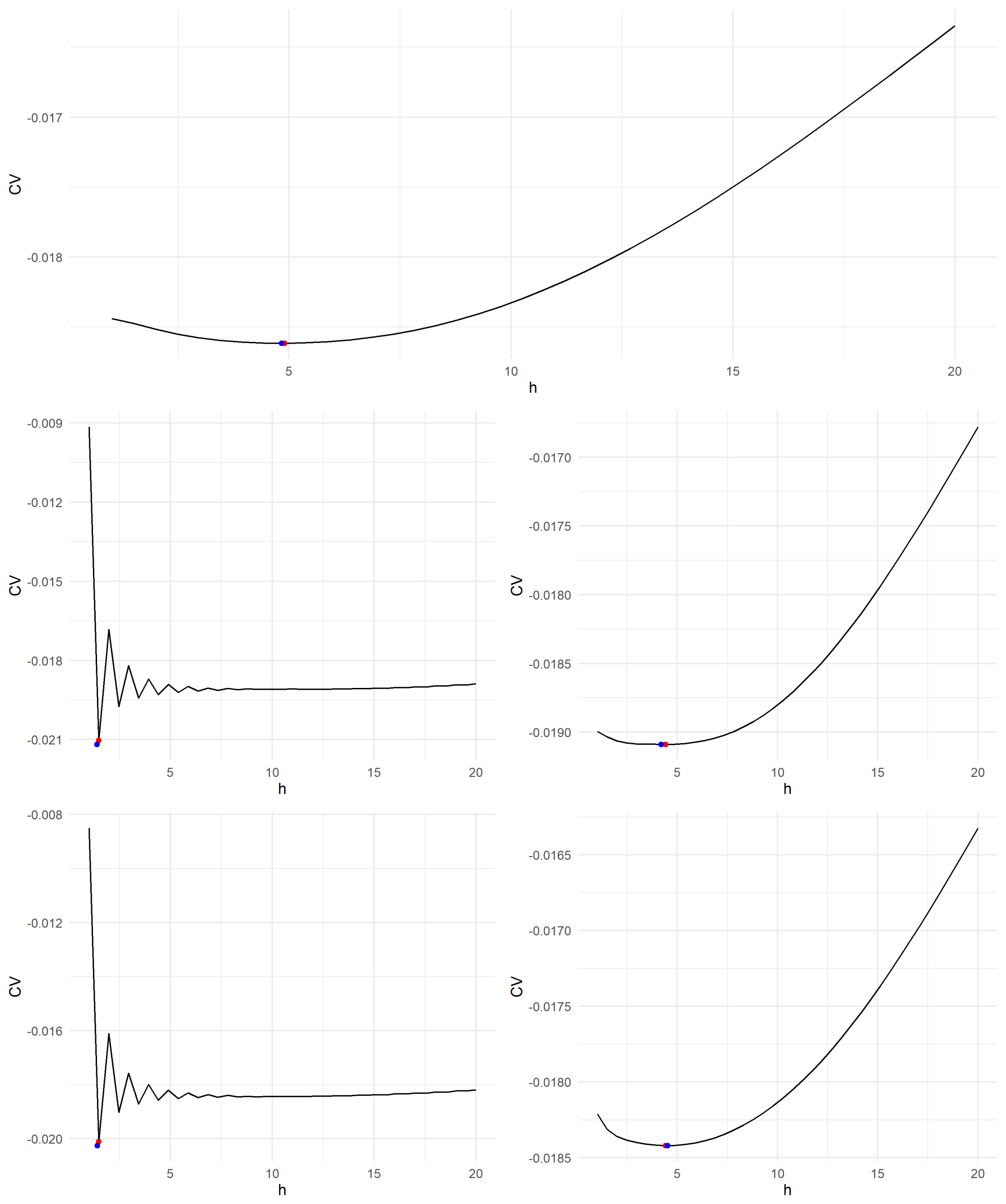


Figure 8.4: CV against h for math score and Gaussian kernel (first row), reading score (second row) and writing score (third row). Last two for Epanechnikov kernel (left) and Gaussian kernel (right)

	Epa		Gau		bw.bcv		bw.ucv	
	h	CV	h	CV	h	CV	h	CV
math	1.62	-0.0197	4.83	-0.0186	4.26	-0.0186	4.65	-0.0186
reading	1.41	-0.0212	4.19	-0.0191	4.40	-0.0191	3.77	-0.0191
writing	1.41	-0.0203	4.52	-0.0184	4.18	-0.0184	4.28	-0.0184

Table 8.1: Optimal bandwidths h and corresponding CV for math, reading and writing score and different methods. First two columns are optimum of self-written function with Epanechnikov kernel (Epa) and Gaussian kernel (Gau) respectively, last two columns with R function in the package `density`.

of my own optimisation for the Gauss kernel are similar to those of the R functions but differ from the optimal bandwidth for the Epanechnikov kernel. For every score the resulting CVs just differ on the presented scale between the kernels where always the Epanechnikov kernel gives a better result. Therefore we use this bandwidth an the Epanechnikov kernel for the last part of the exercise.

8.3 Comparison of groups

Now we use the kernel density estimation for a comparison of those students attending a preparation course and those that did not. Therefore we fit a curve for both groups using the optimal bandwidth we got in the previous examinations and the Epanechnikov kernel. For the bandwidth we use a common value equal to the mean of the three optimal bandwidths (this is just for a simplification of the code and does not change the results crucially).

The result is shown in Figure 8.5. I decided to plot the histograms above each other, so that there is no confusion about the positioning and because one can still distinguish the bars with just two groups. We see that for the math score the whole distribution of scores is not changed very much in the shape but just shifted to the right, i.e. to better grades. For the other two scores we can recognise that the distribution of students with the preparation course is a bit more left skewed. This is just slightly visible for the reading score and a bit more distinct for the writing score. This is probably due to the fact that the student get better in total but the limit of the score is at 100. Psychologists call this effect *ceiling effect*. This is maybe also the reason why the distribution has a smaller variation, especially for the writing score again. Summing up, we recognize better performance of students doing a preparation course. This is especially visible for the writing score. Apparently, one cannot train maths as easily as writing.

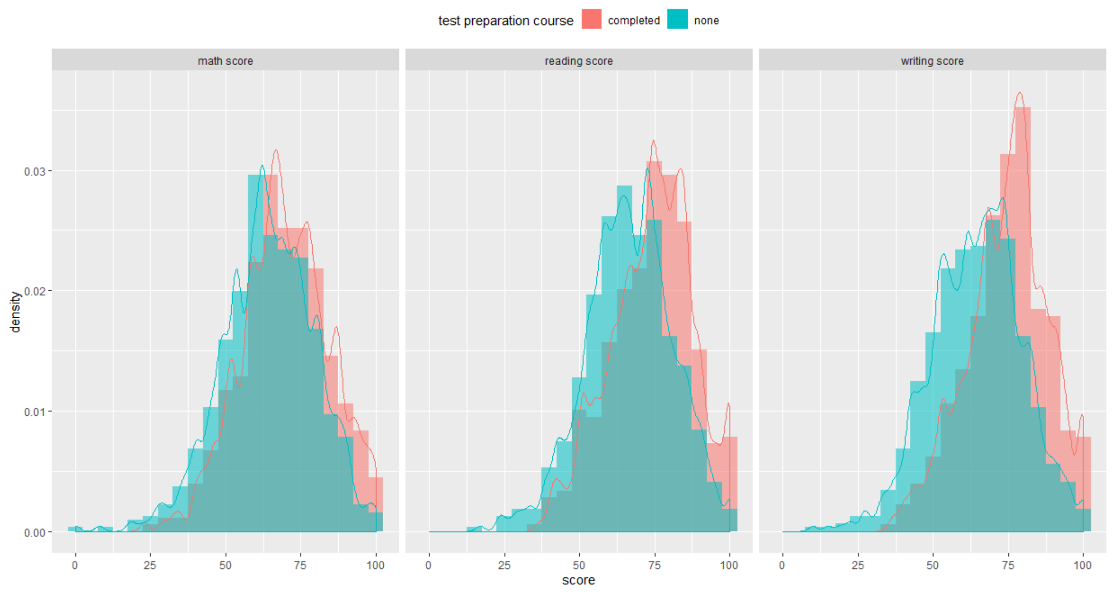


Figure 8.5: Histogram and kernel density estimate for two groups of students in different colors and the three scores, math (left), reading (middle) and writing (right).

9 Local polynomial regression

9.1 Different bandwidths and kernels

This exercise is about a method for non-parametric regression called local polynomial regression (LPR). It is a generalisation of the kernel based Nadaraya-Watson estimator, similar to the kernel density estimation in Exercise 6. The Nadaraya-Watson estimator is a local weighted mean estimation. The local polynomial regression estimation for $(X_1, Y_1), ..., (X_n, Y_n) \overset{iid}{\sim} (X, Y)$ and the model

$$Y_i = f(X_i) + \varepsilon_i,\ \varepsilon_i \overset{iid}{\sim} \left\langle 0, \sigma^2 \right\rangle$$

is derived by a Taylor approximation of the function f.

We apply this method to the data of Kenyan children already used in Exercise 1. We want to fit a curve for the Z-score for wasting (`zwast`) against age. This Z-score is a transformation of a child's weight standardised by the median and standard deviation of the weight of healthy children with the same height. We wrote our own function for fitting a local polynomial curve with data, bandwidth, kernel and polynomial order as arguments. To examine the influence of argument choice we again fitted a curve for different bandwidths and the same kernels as in Exercise 6 with a fixed order 1 polynomial regression (see Figure 9.1).

We see similar results as in Exercise 6. With increasing bandwidth the curve gets smoother. With a bandwidth of 1 the fit represents the data rather locally. With $h = 2$ some of the fluctuations are flattened out and with increasing bandwidth the curve converges to the constant mean of the data (see blue line). I took 40 as the highest bandwidth here to demonstrate this fact. A reasonable choice seems to be $h = 10$ since there the functions looks pretty smooth but still represents the first drop of the Z-score and the slight increase and convergence afterwards that we see in the scatter plot in Figure 9.1. Then using this bandwidth we were asked to fit the curve for different kernels. Like in Exercise 6 again we see that the Gaussian kernel produces the most flat curve. Indeed it looks quite similar to the fit for the Epanechnikov kernel with an astonishing high bandwidth compared to the one used here. The fits for the triangular and Epanechnikov kernel don't differ obviously. The red line representing the fit for the rectangular kernel seems piecewise linear which is quite natural as the kernel is piecewise constant and we used a LPR of order 1 (i.e. using linear polynomials) in all these plots.

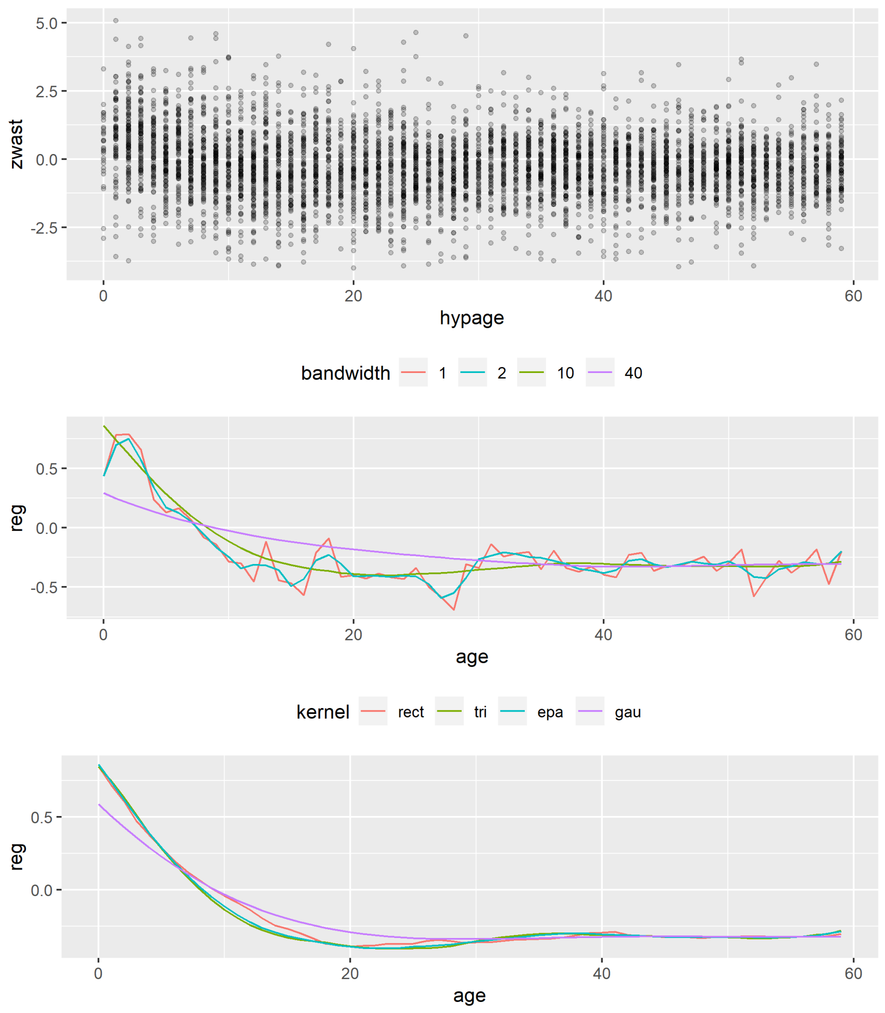


Figure 9.1: First: Age against Z-score in the Kenyan children data. Second row: LPR fit with different bandwidths and Epanechnikov kernel. Third: LPR fit with different kernels and fixed bandwidth $h = 10$.

9.2 Generalised cross-validation

Similar to the cross-validation method in Exercise 6 there is a generalised version for LPR we want to use here to find the best bandwidth for different polynomial degrees. The generalised cross-validation criterion (GCV) is given by

$$GCV(h) = \frac{\sum_{i=1}^{n} \left[Y_i - \hat{f}_{n,h}(X_i) \right]^2}{1 - n^{-1} \sum_{i=1}^{n} W_i(X_i, h)}$$

where W_i are the weights of the different observations for the linear generalised least square estimation used to compute the parameters. This time we used only the Epanechnikov kernel to search for the minimal GCV. Again we have the same problem - like in Exercise 6 - that the GCV curve is not unimodal and thus optimize won't find the global minimum (see Figure 9.2).

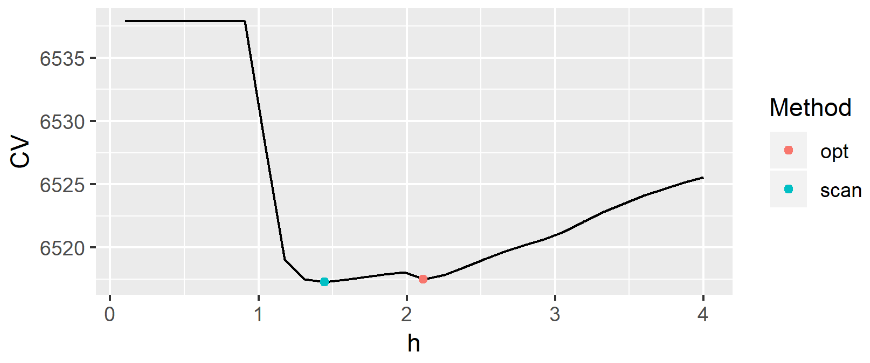


Figure 9.2: GCV against h. Points show optimal h for the scan over 30 points (blue) and the `optimize` function (red).

We therefore use a scanning strategy only to get the optimal bandwidths. The results are shown in Table 9.1. We see that the results for degree 2 and 3 are identical (rounded to two digits) and similar to the results for degree 4 whereas for degree 1 a lower bandwidths seems the best option. The degree 4 fit gives the highest GCV indicating a worse fit.

deg	h	GCV
1	1.44	6517.30
2	3.19	6518.00
3	3.19	6518.00
4	3.06	6522.78

Table 9.1: Optimal bandwidth h together with minimal GCV for different polynmial degrees (deg).

Now have a look at the fitted curves (see Figure 9.3). The curves don't differ but look very similar. This is especially surprising since the optimal bandwidth for degree 1 is much lower than for the other curves. But with higher polynomial degree the regression as more degrees of freedom (i.e. more parameters) to adapt to the data. This would explain why all three curves look like a very local fit. On the left border all three curves decrease as age tends to 0. This does not seem reasonable since the shape of the data suggests that the Z-score is very high at birth and decreases rapidly in the first months. This is one disadvantage of local non-parametric regressions, that in general at the border and outside of the observed data range the prediction is not valid and thus generalisations difficult. Finally note, that the curves in Figure 9.3 look piecewise linear. Indeed in this plot they are drawn since we approximated the regression curves at the data points. One could have generalised the function fitting the curve with an argument

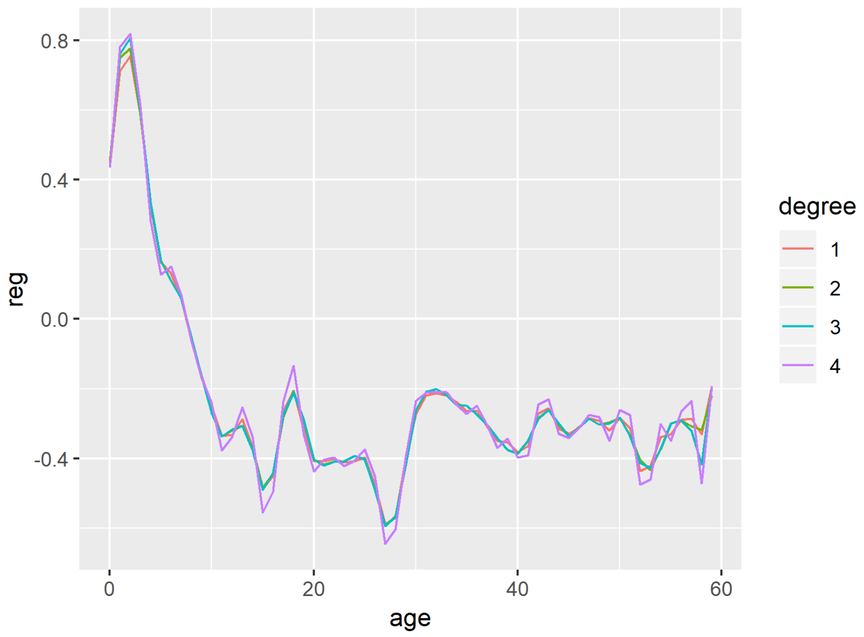


Figure 9.3: LPR fit for different polynomial degrees each with optimal bandwidth according to Table 9.1 with Epanechnikov kernel.

x to draw a better approximation with more points but note that one has to fit a linear model for every point then and this would lead to higher computational effort. Still one would recognizes difference in shape in the way the data is presented here. Summing up and looking back to the second plot in Figure 9.1 using the GCV may not be the best method to find a suitable bandwidth in every case but can result in a - in my opinion - too low bandwidth that results in a more local representation than a reasonable and useful trend analysis of the data.

9.3 Estimating derivatives

When considering the change of a variable over time it may be useful to have a look at the derivative of the curve. The method of LPR provides also an on-the-fly method to estimate the derivatives since it is based on a Taylor expansion of the function. For a regression of the derivatives of the curve we use the R function `localpoly.reg` in the package `NonpModelCheck`.

Indeed this function provides all functionality we need for this exercise. The result in the first row Figure 9.4 show the derivatives with the optimal bandwidths with respect to GCV we found in the last section. We recognize that the strong fluctuation of the regression function itself, shown in the previous section, is also visible for the derivative with is also highly fluctuating and thus in my opinion not really helpful for a reasonable examination of the change in the Z-score. Especially the values at the borders are

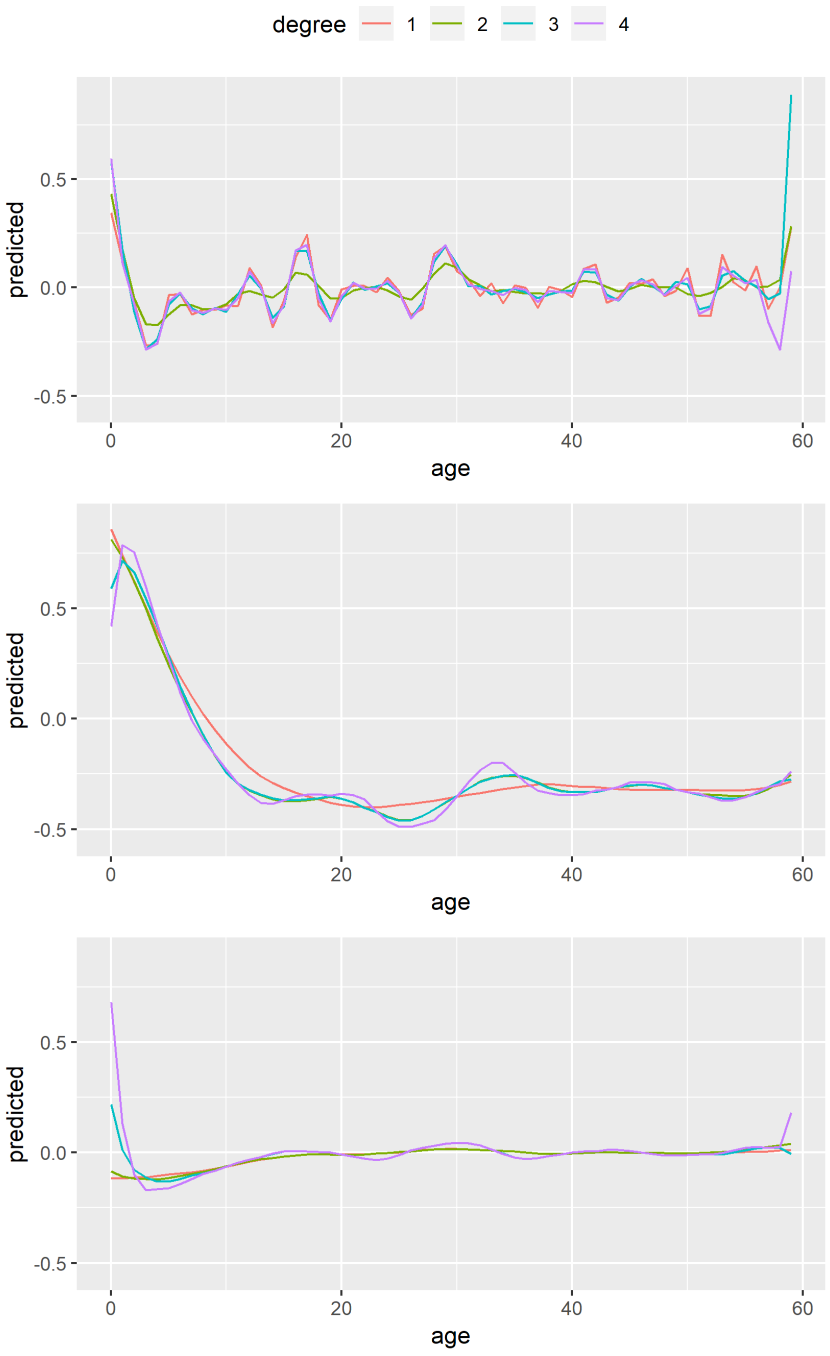


Figure 9.4: LPR fits with function `localpoly.reg` for different polynomial degrees. First plot: first derivative with optimal bandwidths according to 9.1; Second plot: regression curve with fixed bandwidth $h = 10$; Third plot: first derivative with fixed bandwidth $h = 10$.

very high and don't fit at all to the apparently converging behaviour of the data we see in Figure 9.1. Here we would maybe guess that the Z-score is improving since the derivative is positive but because of the high fluctuation we would expect it to decrease shortly afterwards. I decided to use a higher bandwidth again and fit the curve again (see second and third plot in Figure 9.4). I used $h = 10$ because it seemed already a reasonable bandwidth in the first section. I included the regression for the actual function and their derivative. Obviously the fluctuation is lower as expected. The fits for degree 3 and 4 still show a slight overfit, especially at the left border. The green and the red line are hard do distinct, so the fit for degree 1 and 2 are very similar. Both start with a negative derivative that tends to 0 and keeps a slightly positive value then. All in all I would conclude that there is a very small improvement of the Z-score at the age of 2 since the red and green curve are slightly above 0 on the right corner of Figure 9.4 in the last plot.

10 Penalised regression