Böing-Messing, F., Kaptein, M.C., & van den Heuvel, E.R.

Statistics for Data Scientists

Solutions to the end-of-chapter problems.

December 31, 2020

Springer

Preface

This document is the solutions manual for the end-of-chapter problems in the book *Statistics for Data Scientists* by Maurits Kaptein and Edwin van den Heuvel. In this manual, we present guided solutions that include important intermediate steps and additional explanations. A large part of the problems is solved in [R] and the corresponding code is presented in this document as well. Furthermore, a number of problems concern analyzing data sets containing real as well as simulated data. These data sets can be found at `http://www.nth-iteration.com/statistics-for-data-scientist`. On this website, we also provide [R] scripts that contain all the code that is shown in this manual. Thus, the interested reader can download the scripts and easily reproduce the solutions in [R] without copying and pasting any code from this PDF document.

Tilburg, Nijmegen, Eindhoven, Den Bosch,
January 2019

Florian Böing-Messing
Maurits Kaptein
Edwin van den Heuvel

Contents

Chapter 1
Solutions for Chapter 1

1.1

1. To compute the mean, mode, and median of `Age`, execute the following [R] code:

```
> demographics <- read.csv("demographics-synthetic.csv")
>
> getmode <- function(v) {
+  uniqv <- unique(v)
+  uniqv[which.max(tabulate(match(v, uniqv)))]
+ }
>
> mean(demographics$Age)
[1] 28.4066
> median(demographics$Age)
[1] 26.85
> getmode(demographics$Age)
[1] 23.8
```

   We proceed similarly for `Weight` and `Voting`.
2. The following are some errors that are easy to find:

   - The 333^{rd} person has an age of 622, which seems impossible.
   - A missing value (`NA`) appears in the variable `Weight` in the 400^{th} row.
   - Further missing values (`99`) appear in the variable `Voting` in rows 217, 242, 259, 327, and 337.

   These units can be deleted using

```
> demographics <- demographics[-c(333, 400, 217, 242, 259,
    327, 337), ]
```

   After the deletion, we are left with 493 rows (or units):

```
> nrow(demographics)
[1] 493
```

3.
```
> mean(demographics$Age)
[1] 27.19838
> median(demographics$Age)
[1] 26.8
> getmode(demographics$Age)
[1] 23.8
```

   The mean and median became smaller since we removed the person with an age of 622. The mode stayed the same.

4.
```
> var(demographics$Height)
[1] 269.3064
> var(demographics$Weight)
[1] 94.90481
```

   The variance of `Height` is greater than the variance of `Weight`. This means that the average squared deviation from the sample mean is greater for `Height` than it is for `Weight`.

5.
```
> quantile(demographics$Age, probs=0.18)
 18%
24.2
```

   This means that 18% of the values in `Age` are smaller than 24.2. In other words, 18% of the persons in the data set are younger than 24.2.

6.
```
> plot(demographics$Weight, demographics$Age)
```

   The plot is shown in Figure 1.1a. There is no clear relationship between the two variables.

7.
```
> plot(demographics$Weight, demographics$Age, xlab="Weight",
    ylab="Age",
+     col="red", main="Scatterplot␣of␣Weight␣and␣Age")
```

   The plot is shown in Figure 1.1b. The horizontal and vertical black lines will be added in the next part.

8. Lines can be added using the `abline()` function (see `?abline` for its documentation):

```
> abline(h=30)
> abline(v=90)
```

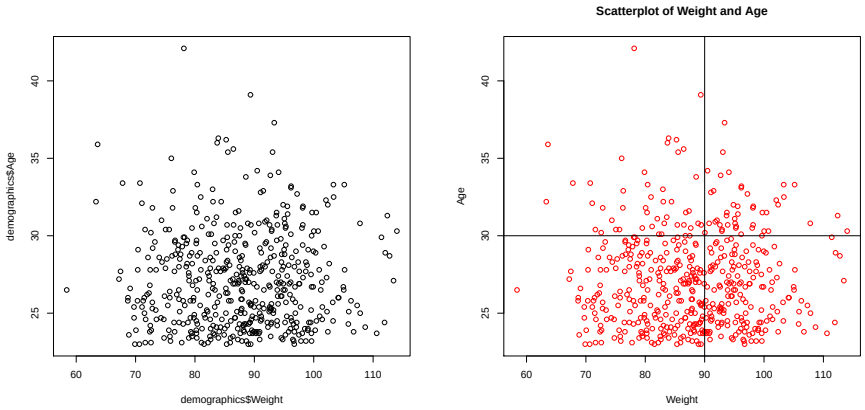
9. Since `Gender` is a factor and `Age` is numeric, [R]'s default `plot()` function will produce the desired boxplot:

```
> plot(demographics$Gender, demographics$Age)
```

   The plot is shown in Figure 1.1c.

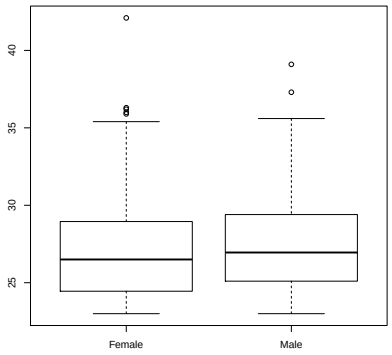
10. The code below uses the `par()` function to set up a figure with two panels:

```
> par(mfrow=c(1, 2))
> plot(demographics$Weight, demographics$Age, xlab="Weight",
    ylab="Age",
+     col="red", main="Scatterplot␣of␣Weight␣and␣Age")
> plot(demographics$Gender, demographics$Age)
```

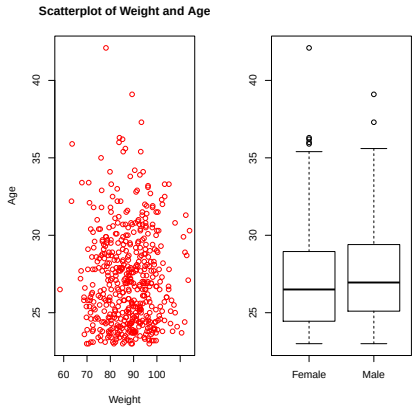


(a)

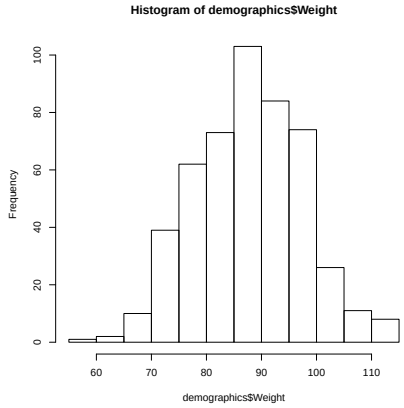
(b)



(c)



(d)



(e)

Fig. 1.1: Plots for Problem 1.1.

The plot is shown in Figure 1.1d.

11. One rule for determining the number of bins is the Freedman-Diaconis rule. According to this rule, the bin width is calculated as $2 \cdot \text{IQR} \cdot n^{-1/3}$, where n is the number of observations and IQR is the interquartile range (i.e., the distance between the 25^{th} and the 75^{th} percentile). The number of bins is then set to $(\max - \min)/\text{bin width}$, where $\max - \min$ is the range of the data. In [R] this can be obtained as follows:

```
> hist(demographics$Weight, breaks="FD")
```

Here `"FD"` stands for Freedman-Diaconis. The histogram is shown in Figure 1.1e.

1.2

1.
```
> voting <- read.csv("voting-demo.csv")
> nrow(voting)
[1] 750
> ncol(voting)
[1] 7
```

2. The measurement levels are as follows:

   - `X`: nominal
   - `Vote`: nominal
   - `Age`: ratio
   - `Church`: nominal
   - `Choice`: nominal
   - `Educ`: ordinal
   - `agegr`: ordinal

3. A quick overview can be obtained using the `summary()` function:

```
> summary(voting)
```

Note that [R] considers `X` a continuous variable even though it actually is a nominal variable that assigns an index to each unit in the data set. From the summary it can be seen that there is one missing value (NA) on the variable `Age` and two missing values on each of the two variables `Vote` and `Church`. An inspection of the data shows that the missing values appear in rows 94, 239, 335, 602, and 649:

```
> which(is.na(voting$Vote))
[1] 602 649
> which(is.na(voting$Age))
[1] 335
> which(is.na(voting$Church))
[1] 94 239
```

We remove these rows from the data set and compute the summary again on the data set without the missing values:

```
> voting <- voting[-c(94, 239, 335, 602, 649), ]
> summary(voting)
```

In addition to the statistics from the `summary()` function, we can compute the mode for all variables in the data set. The following code computes the mode for the variable `Age`:

```
> getmode <- function(v) {
+  uniqv <- unique(v)
+  uniqv[which.max(tabulate(match(v, uniqv)))]
+ }
> getmode(voting$Age)
[1] 50
```

For the nominal and ordinal variables `Vote`, `Church`, `Choice`, `Educ`, and `agegr` we can also produce a frequency table. The following code does so for the variable `Choice`:

```
> t1 <- table(voting$Choice)
> t2 <- transform(t1, cumulative=cumsum(Freq), relative=prop.
    table(Freq))
> t2
    Var1 Freq cumulative relative
1 CDU/CSU 330    330 0.4429530
2    FDP 131     461 0.1758389
3    SPD 284     745 0.3812081
```

Finally, for the ratio variable `Age` we can compute all the measures of central tendency and dispersion that the `summary()` function does not return: range, interquartile range (IQR), mean absolute deviation (MAD), mean squared deviation (MSD), variance, and standard deviation (SD). Furthermore, we can compute the skewness and kurtosis of `Age`. Note that [R] does not have built-in functions for computing the skewness and kurtosis. This is not a problem since we can use the functions `skewness()` and `kurtosis()` from the package `e1071` to carry out the computations. The following [R] code computes all these measures:

```
> library(e1071)
> n <- length(voting$Age) # We first compute the number of
    observations in Age.
> max(voting$Age) - min(voting$Age) # Range
[1] 52
> quantile(voting$Age, probs=0.75) - quantile(voting$Age,
    probs=0.25) # IQR
75%
 26
> sum(abs(voting$Age - mean(voting$Age))) / n # MAD
[1] 13.14503
> (n - 1)*var(voting$Age) / n # MSD
[1] 230.175
> var(voting$Age) # Variance
[1] 230.4844
> sd(voting$Age) # SD
[1] 15.18171
> skewness(voting$Age) # Skewness
[1] 0.01954586
> kurtosis(voting$Age) # Kurtosis
```

```
[1] -1.180987
```

4. This is more of a trial and error question, and there is no definitive method. However, if you plot some of the variables you can see that the artificial data has unrealistic values and distributions, which is not true for this real data set.

1.3

1. The following function creates a frequency table using the material discussed in Chapter 1:

```
> create.table <- function(x) {
+  tab <- table(x)
+  tab <- transform(tab, cumFreq=cumsum(Freq), relative=prop.
    table(Freq))
+  colnames(tab) <- c("Value", "Frequency", "Cum.Freq", "Rel.
    Freq")
+  return(tab)
+ }
```

   The function can be used by making a call to `create.table()` and passing it a vector as an argument. A possible alternative implementation to count the frequency of different factor levels that does not use the built-in [R] functions for manipulating tables is shown in the following code:

```
> count_frequency <- function(x) {
+  counts <- rep(0, times=length(levels(x)))
+  names(counts) <- levels(x)
+  for (value in x) {
+    counts[value] <- counts[value] + 1
+  }
+  return(counts)
+ }
```

   However, note that many alternative ways of computing this exist in [R].
2. This can be implemented as follows:

```
> compute.mean.1 <- function(x) {
+  s <- sum(x)
+  l <- length(x)
+  return(s/l)
+ }
```

3. This is a bit trickier since we cannot use the `length()` and `sum()` functions:

```
> compute.mean.2 <- function(x) {
+  l <- 0
+  s <- 0
+  for (i in x) {
+    l <- l+1
+    s <- s+i
+  }
+  return(s/l)
+ }
```

4. The `compute.mean.2()` function turns out to be slower than the `compute.mean.1()` function. We can see this with the following code:

```
> # Simulate data:
> set.seed(631584)
> x <- rnorm(10^7, mean=0, sd=1)
>
> # Time compute.mean.1() function:
> start.time <- Sys.time()
> compute.mean.1(x)
[1] 0.0001126156
> end.time <- Sys.time()
> end.time - start.time
Time difference of 0.01697183 secs
>
> # Time compute.mean.2() function:
> start.time <- Sys.time()
> compute.mean.2(x)
[1] 0.0001126156
> end.time <- Sys.time()
> end.time - start.time
Time difference of 0.512516 secs
```

   This is a big difference! It is caused by the fact that `for` loops in [R] are actually really slow. [R] is fully vectorized and you should avoid writing manual `for` loops as much as possible.

5. We just used this command above as well. With these arguments it creates a new vector of length 100:

```
> set.seed(631584)
> x <- rnorm(100, mean=0, sd=1)
> length(x)
[1] 100
```

6. First, note that `x` is a continuous variable. Some useful descriptive statistics for the values in `x` can be obtained using the `summary()` function and the `sd()` function (where the latter computes the sample standard deviation):

```
> summary(x)
   Min. 1st Qu. Median Mean 3rd Qu. Max.
-2.84259 -0.70978 0.02160 -0.06938 0.51702 1.99635
> sd(x)
[1] 0.9950925
```

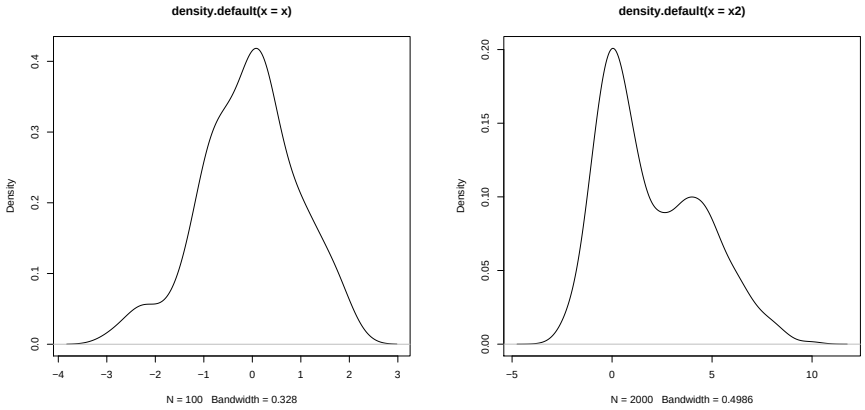
7. Since `x` is a continuous variable, a density plot is a suitable way of visualizing the data in `x`:

```
> plot(density(x))
```

   The plot is shown in Figure 1.2a. It can be seen that the distribution of the data in `x` is (approximately) bell-shaped.

8. The following [R] code performs the required computations:

```
> set.seed(631584)
> x2 <- c(rnorm(1000, mean=0, sd=1), rnorm(1000, mean=4, sd
    =2))
```



(a) (b)

Fig. 1.2: Plots for Problem 1.3.

```
> summary(x2)
   Min. 1st Qu. Median Mean 3rd Qu.  Max.
-3.25713 -0.08936 1.26592 1.95538 3.92013 10.26143
> sd(x2)
[1] 2.533682
> plot(density(x2))
```

The variable `x2` is actually a combination of two (bell-shaped) normal variables that are located (the position of their mean) at different places: One is centered around a mean of 0 and the other around a mean of 4. As a result, the distribution of the data in `x2` has two modes at (approximately) 0 and 4, respectively. This can also be seen in the density plot, which is shown in Figure 1.2b.

1.4

1. The required statistics can be calculated as follows:
   - Arithmetic mean: $\bar{x} = \frac{1}{8}\sum_{i=1}^{8} x_i = (30+23+29+36+68+32+32+23)/8 = 34.125$
   - Median: $(30+32)/2 = 31$
   - Mode: There are two modes, 23 and 32.
   - Range: $68-23 = 45$
   - Mean absolute deviation: $MAD = \frac{1}{8}\sum_{i=1}^{8}|x_i-\bar{x}| = (4.125+11.125+5.125+1.875+33.875+2.125+2.125+11.125)/8 = 8.938$
   - Variance: $s^2 = \frac{1}{8-1}\sum_{i=1}^{8}(x_i-\bar{x})^2 = (17.016+123.766+26.266+3.516+1147.516+4.516+4.516+123.766)/7 = 207.268$
   - Standard deviation: $s = \sqrt{s^2} = \sqrt{207.268} = 14.397$
2. After removing the value of 68 from the data set, we obtain the following descriptive statistics based on the remaining 7 data points:

- Arithmetic mean: 29.286
- Median: 30
- Mode: 23 and 32
- Range: 13
- Mean absolute deviation: 3.673
- Variance: 23.238
- Standard deviation: 4.821

It can be seen that the arithmetic mean is more affected by the removal of the outlier than the median and the mode. Furthermore, removing the outlier has a strong effect on the measures of dispersion (range, mean absolute deviation, variance, and standard deviation).

3. The z-scores are calculated according to $z_i = (x_i - \bar{x})/s = (x_i - 34.125)/14.397$, for $i = 1, \ldots, 8$. The mean of the z-scores is then given by $\frac{1}{8}\sum_{i=1}^{8} z_i = 0$ and the variance is given by $\frac{1}{8-1}\sum_{i=1}^{8} z_i^2 = 1$.

1.5

1. We have that $\sum_{k=1}^{K} f_k = n$.
2. This can be done using first the fact that the sample mean is given by $\bar{x} = \sum_{k=1}^{K} v_k f_k / n$ and subsequently the sample variance can be calculated as $s^2 = \sum_{k=1}^{K} f_k (v_k - \bar{x})^2/(n-1)$.
3. It is possible to compute the median: You can construct all the unique values by taking each value v_k and repeating it f_k times (using for example the `rep()` function in [R]). Subsequently, you can compute the median as you would normally do.

1.6

1. The `rbinom(n, size, prob)` function can be used to generate random numbers with a specific pattern. It will generate `n` numbers (also called draws) that can be thought as the number of heads when tossing a coin `size` times, where the probability that the coin lands heads up (= 1) is equal to `prob`. Check if you understand the following examples (the exact outcomes depend on the seed you select):

```
> set.seed(26078)
> rbinom(10, size=1, prob=0.5)
 [1] 1 0 0 0 0 1 1 0 1 0
> rbinom(10, size=1, prob=0.9)
 [1] 1 0 1 1 1 0 1 1 0 1
> rbinom(10, size=1, prob=0)
 [1] 0 0 0 0 0 0 0 0 0 0
> rbinom(10, size=5, prob=0)
 [1] 0 0 0 0 0 0 0 0 0 0
> rbinom(10, size=5, prob=1)
 [1] 5 5 5 5 5 5 5 5 5 5
> rbinom(10, size=5, prob=0.5)
 [1] 2 0 4 4 4 3 0 3 0 2
```

2. We have already inspected this function in Problem 1.3. Just like `rbinom()`, it generates random numbers with a specific pattern. Formally, `rnorm(n, mean=m, sd=s)` generates `n` random draws from a normal distribution with mean `m` and standard deviation `s`. We will cover the normal distribution in more detail in later chapters.
3. The seed initializes the random number generator and thus changing the seed will change the (pseudo-)randomly drawn numbers. Random number generation and seeds are discussed in more detail in the additional material of Chapter 2.
4. The `ifelse(test, yes, no)` function performs a test of a condition—which is carried out element-wise for vectors that are passed to it—and returns the value of the argument `yes` if the test is `TRUE`, and the value of the argument `no` if the test is `FALSE`. Here is an example:

```
> x <- c(0, 1, 1, 1)
> y <- c(1, 0, 1, 0)
> ifelse(x==y, yes="Match", no="No match")
[1] "No match" "No match" "Match" "No match"
```

Chapter 2
Solutions for Chapter 2

??

1. We can form $\binom{9}{3} = 9!/(3!(9-3)!) = 84$ subsets of three units when ignoring the time structure.
2. We can form 3 systematic subsets of size 3 when considering the time structure: $\{1,4,7\}$, $\{2,5,8\}$, and $\{3,6,9\}$.
3. We can form $\binom{3}{1} \times \binom{3}{1} \times \binom{3}{1} = (3!/(1!(3-1)!))^3 = 27$ stratified subsets of size 3 when considering the time structure.
4. The following [R] code selects one random subset according to the three methods described above:

```
> # Method 1:
> K_1 <- choose(9, 3)
> subset_1 <- sample(1:K_1, size=1)
>
> # Method 2:
> K_2 <- 3
> subset_2 <- sample(1:K_2, size=1)
>
> # Method 3:
> K_3 <- choose(3, 1)*choose(3, 1)*choose(3, 1)
> subset_3 <- sample(1:K_3, size=1)
```

5. The following [R] code selects directly from the population:

```
> # Method 1:
> population <- 1:9
> sample_1 <- sample(population, size=3, replace=FALSE)
> sample_1
[1] 5 8 4
>
> # Method 2:
> group_1 <- c(1, 2, 3)
> group_2 <- c(4, 5, 6)
> group_3 <- c(7, 8, 9)
> sample_2 <- rep(NA, 3)
> sample_2[1] <- sample(group_1, size=1)
```

```
> sample_2[2] <- sample_2[1] + 3
> sample_2[3] <- sample_2[1] + 6 # or sample_2[2] + 3
> sample_2
[1] 1 4 7
>
> # Method 3:
> sample_3 <- rep(NA, 3)
> sample_3[1] <- sample(group_1, size=1)
> sample_3[2] <- sample(group_2, size=1)
> sample_3[3] <- sample(group_3, size=1)
> sample_3
[1] 2 6 7
```

??

1. There are $\binom{6}{3} = 6!/(3!(6-3)!) = 20$ possible simple random samples of 3 schools. The following [R] code recreates the data set, constructs a collection of possible samples, and subsequently computes the bias, standard error, and mean square error:

```
> # Recreate data:
> school <- 1:6
> students <- c(59, 28, 90, 44, 36, 57)
> count <- c(4, 5, 3, 3, 7, 8)
> data <- as.data.frame(cbind(school, students, count))
>
> # Compute estimators for all possible samples of 3 schools:
> samples <- combn(1:6, 3) # Creates a matrix with 20 columns
    containing all possible samples.
> K <- ncol(samples)
> theta <- 30/314
> estimators <- rep(NA, K)
> for (k in 1:K) {
+  sample_school <- samples[, k]
+  sample_data <- data[sample_school, ]
+  estimators[k] <- sum(sample_data$count)/sum(sample_data$
    students)
+ }
>
> # Compute bias, mean square error, and standard error:
> bias <- mean(estimators) - theta
> MSE <- mean((estimators - theta)^2)
> SE <- sqrt(mean((estimators - mean(estimators))^2))
> data.frame(bias=bias, MSE=MSE, SE=SE)
       bias       MSE      SE
1 0.004047035 0.0008348662 0.02860922
```

2. There are $\binom{2}{1} \times \binom{2}{1} \times \binom{2}{1} = 8$ possible stratified samples of 3 schools. To implement stratified sampling, we can make use of the [R] code above. We only need to change the `samples` object such that it contains all possible samples given the strata above. To achieve this, we can make use of the `expand.grid()` function as follows:

```
> stratum_1 <- c(1, 3)
```

```
> stratum_2 <- c(4, 6)
> stratum_3 <- c(2, 5)
> samples <- t(unname(expand.grid(stratum_1, stratum_2,
    stratum_3))) # Creates a matrix with 8 columns containing
    all possible samples.
```

Using this stratified sampling approach results in the following bias, MSE, and SE:

```
> data.frame(bias=bias, MSE=MSE, SE=SE)
        bias       MSE      SE
1 0.0006509842 0.0003211946 0.01791008
```

3. Again, we make use of the `expand.grid()` function:

```
> stratum_1 <- c(1, 2)
> stratum_2 <- c(3, 5)
> stratum_3 <- c(4, 6)
> samples <- t(unname(expand.grid(stratum_1, stratum_2,
    stratum_3)))
```

The results are then as follows:

```
> data.frame(bias=bias, MSE=MSE, SE=SE)
       bias      MSE     SE
1 0.006190646 0.001355985 0.0362996
```

4. Option 2 is to be preferred because this stratified sampling procedure results in the smallest bias, mean square error, and standard error.
5. The following [R] code first draws a simple random sample of 3 schools and subsequently uses the sample data to estimate the bias, standard error, and mean square error:

```
> # Draw sample:
> set.seed(19670912)
> sample_school <- sample(data$school, size=3, replace=FALSE)
> sample_data <- data[sample_school, ]
> sample_imm <- sample_data$count/sample_data$students
>
> # Estimate bias, mean square error, and standard error:
> N <- 6
> n <- 3
> f <- n/N
> bias <- mean(sample_imm) - theta
> SE <- sqrt(1-f)*sqrt(var(sample_imm))/sqrt(n)
> MSE <- SE^2 + bias^2
> data.frame(bias=bias, MSE=MSE, SE=SE)
      bias      MSE      SE
1 0.03865591 0.002167332 0.02594327
```

??

1.
```
> high_school <- read.csv("high-school.csv")
>
> mean(high_school$AGE)
```

```
[1] 12.67477
> mean(high_school$HEIGHT)
[1] 163.3794
> mean(high_school$SPORTS)
[1] 4.430406
> mean(high_school$TV)
[1] 14.22914
> mean(high_school$COMPUTER)
[1] 6.131658
> mean(high_school$ALLOWANCE)
[1] 10.99485
> mean(high_school$WORK)
[1] 10.39444
>
> N <- nrow(high_school)
> var(high_school$AGE)*(N-1)/N
[1] 0.5847917
> var(high_school$HEIGHT)*(N-1)/N
[1] 72.54536
> var(high_school$SPORTS)*(N-1)/N
[1] 21.58447
> var(high_school$TV)*(N-1)/N
[1] 108.9036
> var(high_school$COMPUTER)*(N-1)/N
[1] 61.0186
> var(high_school$ALLOWANCE)*(N-1)/N
[1] 391.9201
> var(high_school$WORK)*(N-1)/N
[1] 950.1984
```

2.
```
> high_school$NOSPRT <- as.numeric(high_school$SPORTS==0)
> high_school$NOTV <- as.numeric(high_school$TV==0)
> high_school$NOCOMPUTER <- as.numeric(high_school$COMPUTER
    ==0)
> mean(high_school$NOSPRT)
[1] 0.10867
> mean(high_school$NOTV)
[1] 0.01625756
> mean(high_school$NOCOMPUTER)
[1] 0.1236693
```

3.
```
> freq_gender <- table(high_school$GENDER)
> freq_subject <- table(high_school$SUBJECT)
> freq_break <- table(high_school$BREAKFAST)
> prop.table(freq_gender)

     Boy    Girl
0.4887256 0.5112744
> prop.table(freq_subject)

           Art        Biology Computer Science      Dance
     0.048053686   0.040104656   0.061994448   0.008528231
         Drama  Dutch language    Economics   Engineering
     0.018095029   0.024626016   0.009247239   0.078671433
```

```
 English language French language  Geography        German
      0.046975174    0.036569534   0.015418722   0.016676986
         History     Mathematics         Music       Nursing
      0.037528211    0.092931754   0.074157662   0.031556452
           Other Physics & Chemisty    Sports
      0.061175578    0.018314726   0.279374463
> prop.table(freq_break)

Brought something        No           Yes
      0.06974375    0.05722103    0.87303521
```

4. a.
```
> # Draw a simple random sample:
> set.seed(19670912)
> n <- 1200
> sample_Nr <- sample(high_school$Nr, size=n, replace=
    FALSE)
> new_data <- high_school[sample_Nr, ]
>
> # Estimate the means and standard errors (run ?apply for
     information on the apply() function):
> apply(new_data[, c(4:8, 10:11)], 2, mean)
      AGE   HEIGHT   SPORTS      TV  COMPUTER ALLOWANCE  WORK
 12.643333 163.472500 4.553333 14.302500 6.277500
     11.329167 10.229167
> vars <- apply(new_data[, c(4:8, 10:11)], 2, var)
> N <- nrow(high_school)
> n <- nrow(new_data)
> f <- n/N
> SE_hat <- sqrt(1-f)*sqrt(vars)/sqrt(n)
> SE_hat
      AGE   HEIGHT   SPORTS      TV  COMPUTER ALLOWANCE  WORK
0.02263629 0.24652424 0.14450088 0.30661293 0.22867944
    0.76764904 0.93288210
```

   b. The standard error of a proportion can be estimated using the following formula:

$$\hat{SE}(\hat{\pi}) = \sqrt{(1-f)}\sqrt{\hat{\pi}(1-\hat{\pi})}/\sqrt{n},$$

   where $\hat{\pi}$ is the sample proportion of successes. In [R] we can carry out the calculations as follows:

```
> prop_nosport <- mean(new_data$NOSPRT)
> prop_noTV <- mean(new_data$NOTV)
> prop_nocomp <- mean(new_data$NOCOMPUTER)
> prop_nosport
[1] 0.1025
> prop_noTV
[1] 0.01666667
> prop_nocomp
[1] 0.1308333
>
> SE_nosport <- sqrt(1-f)*sqrt(prop_nosport*(1-prop_
    nosport))/sqrt(n)
> SE_noTV <- sqrt(1-f)*sqrt(prop_noTV*(1-prop_noTV))/sqrt(
    n)
```

```
> SE_nocomp <- sqrt(1-f)*sqrt(prop_nocomp*(1-prop_nocomp))
    /sqrt(n)
> SE_nosport
[1] 0.008650094
> SE_noTV
[1] 0.003651038
> SE_nocomp
[1] 0.009617283
```

c.
```
> freq_SexMath <- table(new_data$GENDER, new_data$SUBJECT)
>
> prop_boy <- prop.table(freq_SexMath)[1, 14]
> prop_girl <- prop.table(freq_SexMath)[2, 14]
> prop_boy
[1] 0.0525
> prop_girl
[1] 0.04833333
>
> SE_boy <- sqrt(1-f)*sqrt(prop_boy*(1-prop_boy))/sqrt(n)
> SE_girl <- sqrt(1-f)*sqrt(prop_girl*(1-prop_girl))/sqrt(
    n)
> SE_boy
[1] 0.006360788
> SE_girl
[1] 0.006116563
```

5. a. We determine the sample size in province k as $n_k = N_k/N \times 1200$, for $k = 1,\ldots,12$, where N_k is the number of high school children in province k, N is the total number of high school children, and 1200 is the size of the simple random sample in part 4 of this exercise. If necessary, the resulting sample size n_k is rounded to the nearest integer. With this choice the sample sizes are proportional to the size of the provinces and the sum of the sample sizes $n_1 + \cdots + n_{12}$ is about equal to 1200 (due to the rounding).

b.
```
> # Note that high_school$PROVINCE is coded from 20 to 31,
     so we recode it from 1 to 12:
> high_school$PROV <- high_school$PROVINCE - 19
>
> # Create vectors and matrices for storing the results
    for 12 provinces and 7 numerical variables:
> W_k <- rep(NA, 12)
> n_k <- rep(NA, 12)
> N_k <- rep(NA, 12)
> mean_k <- matrix(NA, nrow=12, ncol=7, dimnames=list(c(),
    colnames(high_school)[c(4:8, 10:11)]))
> var_k <- matrix(NA, nrow=12, ncol=7, dimnames=list(c(),
    colnames(high_school)[c(4:8, 10:11)]))
>
> # Draw a stratified sample and compute the sample means
    and variances in each stratum (i.e., province):
> N <- nrow(high_school)
> n <- 1200 # 1200 is the size of the simple random sample
    from part (4) above.
> set.seed(19670912)
```

```
> for (i in 1:12) {
+  PROVINCE <- high_school[high_school$PROV==i, ]
+  N_k[i] <- nrow(PROVINCE)
+  W_k[i] <- N_k[i]/N
+  n_k[i] <- round(W_k[i]*n)
+
+  # Draw random sample from province i:
+  PROVINCE$newid <- 1:N_k[i]
+  strat_prov <- sample(PROVINCE$newid, size=n_k[i],
    replace=FALSE)
+  sample_prov <- PROVINCE[strat_prov, ]
+
+  # Compute means and variances for 7 variables using the
     sample data from province i:
+  mean_k[i, ] <- apply(sample_prov[, c(4:8, 10:11)], 2,
    mean)
+  var_k[i, ] <- apply(sample_prov[, c(4:8, 10:11)], 2,
    var)
+ }
>
> means <- apply(W_k*mean_k, 2, sum)
> means
     AGE   HEIGHT   SPORTS     TV  COMPUTER ALLOWANCE  WORK
 12.688329 163.432034 4.471823 13.802539 6.214184
     11.529400 12.335856
>
> var_stra <- (1 - n/N)*W_k*var_k/n
> SE_stra <- sqrt(colSums(var_stra))
> SE_stra
     AGE   HEIGHT   SPORTS     TV  COMPUTER ALLOWANCE  WORK
0.02217014 0.23821966 0.15140780 0.27624249 0.22625084
    0.74666866 1.10461847
```

6. We prefer sampling approaches for which the (estimated) standard error is smaller. Hence, simple random sampling would be preferred for the variables SPORTS and WORK, whereas our stratified sampling approach is to be preferred for the variables AGE, HEIGHT, TV, COMPUTER, and ALLOWANCE.

??

1.
```
> # Histogram:
> hist(estimators, breaks=30)
> abline(v=theta)
>
> # Density plot:
> plot(density(estimators))
> abline(v=theta)
```

   The histogram is shown in Figure 2.1a and the density plot in Figure 2.1b.
2. For $n = 10$ we get the following results:

```
> print_summary(theta=theta, estimators=estimators)
        bias    mse      se
1 -0.0001635409 0.7963648 0.8923927
```

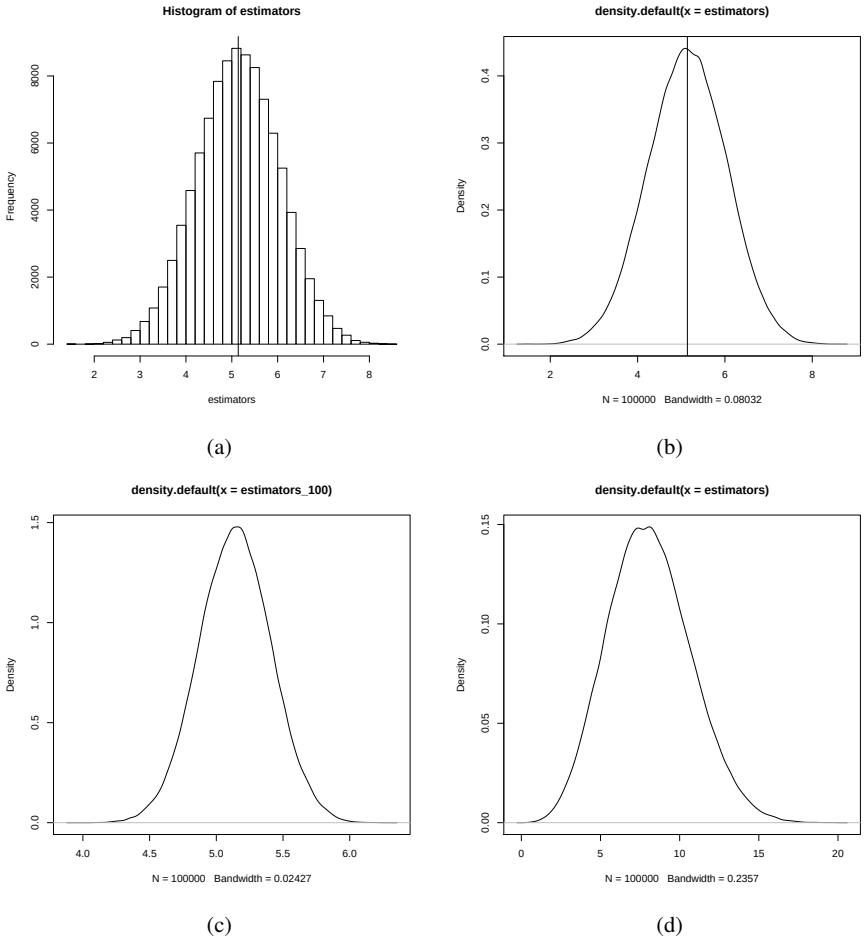


Fig. 2.1: Plots for Problem **??**.

A sample size of $n = 100$ yields the following results:

```
> print_summary(theta=theta, estimators=estimators)
       bias      mse       se
1 0.0004709154 0.07273873 0.2697008
```

It can be seen that the bias remains close to 0, whereas the mean square error and the standard error are smaller for $n = 100$. Figure 2.1c shows a plot of the distribution of the estimators for $n = 100$. This distribution is narrower than the one for $n = 10$ shown in Figure 2.1b, which is in line with the smaller standard error for $n = 100$.

3. Here, we need to change the `compute_estimator()` function as follows (all other code remains the same):

```
> compute_estimator <- function(x){
+  est <- median(x)
+  return(est)
+ }
```

Using the sample median as the estimator for the population mean yields the following bias, mean square error, and standard error:

```
> print_summary(theta=theta, estimators=estimators)
      bias    mse     se
1 0.08441917 1.813524 1.344023
```

Using the sample mean as the estimator gave the following results:

```
> print_summary(theta=theta, estimators=estimators)
        bias    mse      se
1 -0.0001635409 0.7963648 0.8923927
```

Hence, the sample mean is to be preferred as an estimator for the population mean because it has smaller bias, MSE, and SE than the sample median.

4. Here, we need to make two changes to the code. First, we need to change the `compute_estimator()` function as follows:

```
> compute_estimator <- function(x) {
+  est <- sum((x - mean(x))^2) / (length(x) - 1) # Computes
    the sample variance using n - 1 in the denominator.
+  return(est)
+ }
```

Next, we also need to change the line that computes the population parameter `theta` such that it computes the population variance:

```
> theta <- sum((P$x - mean(P$x))^2) / length(P$x) # Computes
    the population variance using N in the denominator.
```

We then obtain the following results:

```
> print_summary(theta=theta, estimators=estimators)
     bias    mse     se
1 0.0172433 6.856162 2.618371
```

Figure 2.1d shows a plot of the distribution of the estimators. It can be seen that the distribution is slightly skewed to the right. This is due to the fact that the variance cannot be negative.

5. In this exercise we need to change the `sample_from_population()` function as follows:

```
> sample_from_population <- function(P, n) {
+  selected_groups <- sample(sort(unique(P$group)), size=n) #
     Draw simple random sample of n clusters.
+  in_selected_groups <- P$group %in% selected_groups #
    Determine which units are in the selected groups.
+  selected <- P$i[in_selected_groups] # Select the units
    that are in the selected groups.
+  return(selected)
+ }
```

For more information on the `%in%` operator run `?match` and try running `1:10 %in% c(1, 3, 5, 9)`. Note that in our new `sample_from_population()` function `n` is the number of clusters to be sampled (instead of the number of units). Next, we need to change the beginning of our simulation as follows:

```
> # Set seed, N, groups, n, and the number of simulations
> set.seed(12345)
> N <- 1000
> groups <- 50
> n <- 5
> sims <- 100000
>
> # Generate the population data:
> P <- generate_population_data(N=N, groups=groups)
```

We then obtain the following simulation results:

```
> print_summary(theta=theta, estimators=estimators)
       bias    mse      se
1 0.001663936 0.0856343 0.2926287
```

6. We do not need the π_k's when computing the bias, MSE, and SE because in our simulation samples that are more likely (i.e., have a larger π_k) will be drawn more often. In fact, sample k will be drawn about $\pi_k \times 100\%$ of the time. As a result, the estimates of more likely samples appear more often in our simulation. This allows us to compute the bias, MSE, and SE without explicitly using the π_k's. This can best be illustrated with a simple example: Suppose our population contains units 1, 2, and 3, and our sample function gives the probabilities $\pi_1 = 1/4$ to sample $S_1 = \{1,2\}$, $\pi_2 = 1/4$ to sample $S_2 = \{1,3\}$, and $\pi_3 = 2/4$ to sample $S_3 = \{2,3\}$ (and that is all our possible samples). Then, if we repeat our sampling scheme, say, $m = 10^5$ times, samples S_1 and S_2 will be drawn about $m/4$ times whereas sample S_3 will be drawn about $m/2$ times. Hence, in the computation of the bias, MSE, and SE, the estimate from sample S_3 will appear about twice as often as the estimates from S_1 and S_2. We therefore do not need the probabilities π_1, π_2, and π_3.

Chapter 3
Solutions for Chapter 3

3.1

1. A fair die means that each side has the same probability $1/6$ of turning up in each throw. The result of the second die is irrelevant, since the question relates only to the first die. An odd number is either 1, 3, or 5. The total number of possible outcomes is 6. The probability is $3/6 = 1/2$.
2. If two dice are thrown there are 36 possible outcomes, namely $(1,1)$, $(1,2)$, $(1,3)$, $(1,4)$, $(1,5)$, $(1,6)$, $(2,1)$, $(2,2)$, $(2,3)$, $(2,4)$, $(2,5)$, $(2,6)$, $(3,1)$, $(3,2)$, $(3,3)$, $(3,4)$, $(3,5)$, $(3,6)$, $(4,1)$, $(4,2)$, $(4,3)$, $(4,4)$, $(4,5)$, $(4,6)$, $(5,1)$, $(5,2)$, $(5,3)$, $(5,4)$, $(5,5)$, $(5,6)$, $(6,1)$, $(6,2)$, $(6,3)$, $(6,4)$, $(6,5)$, $(6,6)$. The number of combinations that lead to a sum of eleven is 2, only $(5,6)$ and $(6,5)$. Hence the answer is $2/36 = 1/18$.

3.2

1. There are 51 cards in the deck of which 13 are clubs. Thus the probability is $13/51$.
2. There are four queens in the deck. Hence the probability is $4/51$.
3. If the events are independent, we need to show that the probability of both events simultaneously is equal to the product of the events separately (see calculation rule 3 in Section 3.3). The probability that the card is both a queen and clubs is equal to $1/51$. The product of the probabilities in parts 1 and 2 is equal to $13/51 \times 4/51 = 52/2601$, which is unequal to $1/51 = 51/2601$. Thus the events are not independent.

3.3

1. There are 33 children of which 15 children have had measles. The probability is thus $15/33 = 5/11$.
2. The probability is $9/18 = 1/2$.
3. The probability that the randomly collected child is a boy is equal to $15/33 = 5/11$. The probability that the child is both a boy and has had measles is equal to $6/33 = 2/11 = 22/121$. This probability is not equal to the product $5/11 \times 5/11 = 25/121$ of the events separately. Thus the events are not independent.

3.4

1. The probability is $16/5322 = 8/2661$.
2. The probability is $77/7019$.
3. There are 12341 males in total and 93 males with lung cancer. The probability that a randomly collected male is both a smoker and has lung cancer is $77/12341 = 11/1763$. The probability that a male has lung cancer is $93/12341$. The probability that the male is a smoker is equal to $7019/12341$. The product of the last two probabilities is $652767/154529761 \approx 0.0043$ and this is unequal to $11/1763 \approx 0.0062$. Thus the events are not independent.
4. If A is the event "smoking" and B is the event "lung cancer", then the requested conditional probability is $P(A|B)$. This probability is equal to $P(A \cap B)/P(B)$. The probability of being both a smoker and having lung cancer is $77/12341$. The probability of having lung cancer is $93/12341$. Thus the conditional probability is $P(A|B) = 77/93 \approx 0.8280$.

3.5 We use the probability calculation rules in Section 3.3. Let $A \perp B$, so that $P(A \cap B) = P(A)P(B)$. In the first case (i.e., $A \perp B^c$) we need to show that $P(A \cap B^c) = P(A)P(B^c)$. We proceed as follows:

$$\begin{aligned} P(A \cap B^c) &= P(A) - P(A \cap B) && \text{(by additional rule 2)} \\ &= P(A) - P(A)P(B) && \text{(by assumption)} \\ &= P(A)[1 - P(B)] \\ &= P(A)P(B^c) && \text{(by rule 1)}. \end{aligned}$$

Thus we have that $A \perp B^c$ if $A \perp B$. In the same way we can show that $A^c \perp B$ if $A \perp B$: $P(A^c \cap B) = P(B) - P(A \cap B) = P(B) - P(A)P(B) = P(B)[1 - P(A)] = P(B)P(A^c)$. In the third case (i.e., $A^c \perp B^c$) we proceed as follows:

$$\begin{aligned} P(A^c \cap B^c) &= P((A \cup B)^c) = 1 - P(A \cup B) = 1 - [P(A) + P(B) - P(A \cap B)] \\ &= 1 - P(A) - P(B) + P(A)P(B) = P(A^c) - P(B) + P(A)P(B) \\ &= P(A^c) - P(B)[1 - P(A)] = P(A^c) - P(B)P(A^c) \\ &= P(A^c)[1 - P(B)] = P(A^c)P(B^c). \end{aligned}$$

3.6 Define the event A as being alive at the age of 50 years and event B as being alive the age of 70 years. The probability of interest is the conditional probability $P(B|A)$. The probability is $P(B|A) = P(A \cap B)/P(A)$. The probability that both A and B occur is the same as that event B occurs. A 70 year old female has of course also passed an age of 50 years. The conditional probability is now $P(B|A) = 0.571/0.898 \approx 0.636$.

3.7

1. Suppose A is the event of a positive test result and B is the event of having the disease. The information in the exercise has specified $P(A|B) = 0.9$, $P(A^c|B^c) = 0.9$, and $P(B) = 0.6$. The answer to the question requires Bayes

theorem as given in Equation (3.4). The requested probability is equal to $P(B|A) = (0.9 \times 0.6) / (0.9 \times 0.6 + 0.1 \times 0.4) \approx 0.931$.

2. The exercise asks for the minimum value of $P(A^c|B^c)$, such that $P(B|A)$ is at least 0.95, under the assumption that $P(A|B) = 0.9$ and $P(B) = 0.6$. This requires some manipulation of the Bayes formula in Equation (3.4):

$$0.95 = \frac{0.9 \times 0.6}{0.9 \times 0.6 + [1 - P(A^c|B^c)] \times (1 - 0.6)}.$$

Solving this equality for $P(A^c|B^c)$ gives us the minimum required specificity such that $P(B|A)$ is at least 0.95. After manipulation we find that the specificity $P(A^c|B^c)$ must be at least 0.929.

3. This exercise is similar to part 2, but now the minimum value of $P(A|B)$ is requested to obtain a probability $P(B^c|A^c)$ of at least 0.95, when $P(A^c|B^c) = 0.9$ and $P(B) = 0.6$. Here we need to manipulate the Bayes formula in Equation (3.5):

$$0.95 = \frac{0.9 \times (1 - 0.6)}{0.9 \times (1 - 0.6) + [1 - P(A|B)] \times 0.6}.$$

Solving this equality for $P(A|B)$ gives us the minimum required sensitivity such that $P(B^c|A^c)$ is at least 0.95. After manipulation we find that the sensitivity $P(A|B)$ must be at least 0.968.

3.8 The calculations can be carried out using the following [R] code:

```
> # Create matrix from contingency table in Table 3.2:
> con_tab <- matrix(c(92/763, 256/763, 348/763,
+                77/763, 338/763, 415/763,
+                169/763, 594/763, 1),
+               nrow = 3,
+               ncol = 3,
+               byrow = TRUE,
+               dimnames = list(c("E", "E^c", "Total"),
+                               c("D", "D^c", "Total")))
>
> # Calculate conditional probabilities:
> P_D_given_E <- con_tab[1, 1] / con_tab[1, 3]
> P_D_given_notE <- con_tab[2, 1] / con_tab[2, 3]
>
> # Calculate risk difference:
> P_D_given_E - P_D_given_notE
[1] 0.07882565
>
> # Calculate relative risk:
> P_D_given_E / P_D_given_notE
[1] 1.42484
>
> # Calculate odds ratio:
> O_E <- P_D_given_E / (1 - P_D_given_E)
> O_notE <- P_D_given_notE / (1 - P_D_given_notE)
> O_E / O_notE
[1] 1.577516
```

3.9

1. The fact that the total number of open surgeries and small incisions is equal suggests that the researchers used a cohort study with a sample size of 350 in each treatment group.
2. The risk difference, relative risk, and odds ratio can be calculated using the [R] code from Problem 3.8 as follows:

```
> # Create matrix from contingency table:
> con_tab <- matrix(c(273/700, 77/700, 350/700,
+                289/700, 61/700, 350/700,
+                552/700, 148/700, 1),
+              nrow = 3,
+              ncol = 3,
+              byrow = TRUE,
+              dimnames = list(c("E", "E^c", "Total"),
+                              c("D", "D^c", "Total")))
>
> # Calculate conditional probabilities:
> P_D_given_E <- con_tab[1, 1] / con_tab[1, 3]
> P_D_given_notE <- con_tab[2, 1] / con_tab[2, 3]
>
> # Calculate risk difference:
> P_D_given_E - P_D_given_notE
[1] -0.04571429
>
> # Calculate relative risk:
> P_D_given_E / P_D_given_notE
[1] 0.9446367
>
> # Calculate odds ratio:
> O_E <- P_D_given_E / (1 - P_D_given_E)
> O_notE <- P_D_given_notE / (1 - P_D_given_notE)
> O_E / O_notE
[1] 0.7483485
```

   The results indicate that the probability of successful kidney stone removal is greater for the small incision treatment as compared to open surgery.
3. This can be shown as follows:

$$
\begin{aligned}
OR_{EE^c} &= \frac{O_E}{O_{E^c}} = \frac{P(D|E)[1-P(D|E^c)]}{P(D|E^c)[1-P(D|E)]} = \frac{P(D|E)P(D^c|E^c)}{P(D|E^c)P(D^c|E)} \\
&= \frac{P(D\cap E)/P(E)\times P(D^c\cap E^c)/P(E^c)}{P(D\cap E^c)/P(E^c)\times P(D^c\cap E)/P(E)} = \frac{P(D\cap E)P(D^c\cap E^c)}{P(D\cap E^c)P(D^c\cap E)} \\
&= \frac{P(E\cap D)P(E^c\cap D^c)}{P(E^c\cap D)P(E\cap D^c)} = \frac{P(E|D)P(D)\times P(E^c|D^c)P(D^c)}{P(E^c|D)P(D)\times P(E|D^c)P(D^c)} \\
&= \frac{P(E|D)P(E^c|D^c)}{P(E|D^c)P(E^c|D)} = \frac{P(E|D)[1-P(E|D^c)]}{P(E|D^c)[1-P(E|D)]} = \frac{O_D}{O_{D^c}} = OR_{DD^c}.
\end{aligned}
$$

Chapter 4
Solutions for Chapter 4

4.1

1. Using the binomial probability, the calculation of probability $P(S_{20} = 10)$ results in $P(S_{20} = 10) = \binom{20}{10} \cdot (0.5)^{10}(1-0.5)^{20-10} = 184756 \cdot (0.5)^{20} \approx 0.176$.
2. Here we need to determine $P(S_{20} \leq 9)$. This can be done by calculating $P(S_{20} = 0) + P(S_{20} = 1) + \cdots + P(S_{20} = 9)$. In this case there is a trick. When the parameter $p = 0.5$, the binomial distribution is symmetric around np. The value np is 10, which implies that $P(S_{20} = 0) + P(S_{20} = 1) + \cdots + P(S_{20} = 9)$ is equal to $P(S_{20} = 11) + P(S_{20} = 12) + \cdots + P(S_{20} = 20)$, or in other words, $P(S_{20} \leq 9)$ is equal to $P(S_{20} \geq 11)$. Furthermore, $P(S_{20} \leq 9) + P(S_{20} = 10) + P(S_{20} \geq 11) = 1$. Thus $P(S_{20} \leq 9)$ is now equal to $(1 - 0.176)/2 \approx 0.412$.

4.2

1. Let X be Bernoulli distributed with success probability $p = P(X = 1)$, that is, $X \sim \mathscr{B}(p)$. Then the cumulative distribution function of X is given by

$$F(x) = P(X \leq x) = \begin{cases} 0 & \text{if } x < 0, \\ 1-p & \text{if } 0 \leq x < 1, \\ 1 & \text{if } x \geq 1. \end{cases}$$

   Note that the Bernoulli random variable X can only take on the values 0 and 1, whereas the cumulative distribution function $F(x)$ is defined for all $x \in \mathbb{R}$.
2. The plot can be created using the following [R] code:

```
> k <- 0:15 # Number of events k.
> PXleqk <- ppois(k, lambda = 5) # Poisson CDF with lambda =
    5 evaluated at each k.
> plot(k, PXleqk, ylab=expression(P(X <= k)), pch=19) #
    Create plot.
> segments(x0 = k, y0 = PXleqk, x1 = k+1, y1 = PXleqk) # Add
    line segments because the Poisson CDF is a step function.
```

   The resulting plot is shown in Figure 4.1.

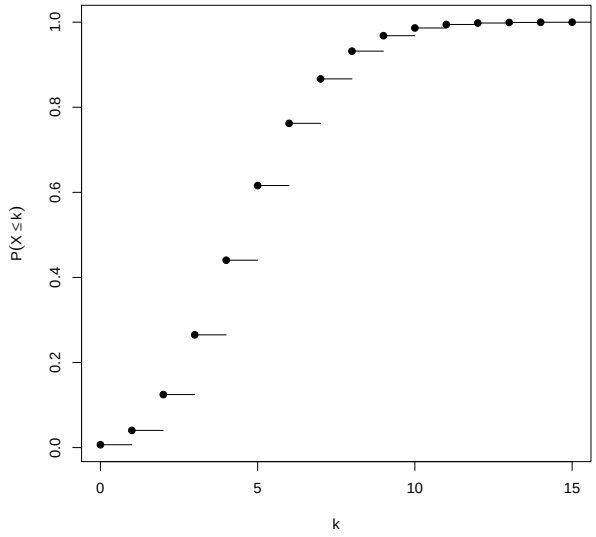


Fig. 4.1: Plot for part 2 of Problem 4.2.

4.3 Let S_{20} be the number of left handed persons. It is binomial distributed with parameters $n = 20$ and $p = 0.1$. The requested probability is $P(S_{20} \geq 3)$. This probability is equal to $P(S_{20} \geq 3) = 1 - P(S_{20} < 3) = 1 - P(S_{20} \leq 2)$. The last probability is equal to $P(S_{20} \leq 2) = P(S_{20} = 0) + P(S_{20} = 1) + P(S_{20} = 2)$. The probabilities are $P(S_{20} = 0) = (0.9)^{20}$, $P(S_{20} = 1) = 20 \cdot (0.1) \cdot (0.9)^{19}$, $P(S_{20} = 2) = 190 \cdot (0.1)^2 \cdot (0.9)^{18}$. Thus $P(S_{20} \leq 2) \approx 0.677$, which implies that $P(S_{20} \geq 3) \approx 0.323$.

4.4 The probabilities can be calculated with the binomial probabilities, using $n = 5$ and $p = 0.9$. This results in $P(S_5 = 0) = (0.1)^5 = 0.00001$, $P(S_5 = 1) = 5 \cdot 0.9 \cdot (0.1)^4 = 0.00045$, $P(S_5 = 2) = 10 \cdot (0.9)^2 \cdot (0.1)^3 = 0.0081$, $P(S_5 = 3) = 10 \cdot (0.9)^3 \cdot (0.1)^2 = 0.0729$, $P(S_5 = 4) = 5 \cdot (0.9)^4 \cdot 0.1 = 0.32805$, and $P(S_5 = 5) = (0.9)^5 = 0.59049$.

4.5

1. First of all, note that the density $f(x)$ of the exponential distribution is given by $f(x) = \lambda \exp(-\lambda x)$ for $x > 0$ and zero otherwise. Using integration by parts, the mean of X is then given by

$$\begin{aligned}\mathbb{E}(X) &= \int_0^\infty x\lambda \exp(-\lambda x)dx \\ &= \left[-\frac{1}{\lambda}\exp(-\lambda x)x\lambda\right]_0^\infty - \int_0^\infty -\frac{1}{\lambda}\exp(-\lambda x)\lambda dx \\ &= \left[-x\exp(-\lambda x)\right]_0^\infty + \frac{1}{\lambda}\int_0^\infty \lambda\exp(-\lambda x)dx \\ &= 0 + \frac{1}{\lambda}\cdot 1 = \frac{1}{\lambda},\end{aligned}$$

where the integrand in the third line is the density of the exponential distribution whose integral over its domain $(0,\infty)$ is equal to 1. For the calculation of the variance of X we first note that $\mathrm{Var}(X) = \mathbb{E}(X^2) - [\mathbb{E}(X)]^2$. The second moment $\mathbb{E}(X^2)$ is now given by

$$\begin{aligned}\mathbb{E}(X^2) &= \int_0^\infty x^2\lambda \exp(-\lambda x)dx \\ &= \left[-\frac{1}{\lambda}\exp(-\lambda x)x^2\lambda\right]_0^\infty - \int_0^\infty -\frac{1}{\lambda}\exp(-\lambda x)2\lambda x dx \\ &= \left[-x^2\exp(-\lambda x)\right]_0^\infty + \frac{2}{\lambda}\int_0^\infty x\lambda\exp(-\lambda x)dx \\ &= 0 + \frac{2}{\lambda}\cdot\frac{1}{\lambda} = \frac{2}{\lambda^2},\end{aligned}$$

where the integral in the third line is the mean $\mathbb{E}(X) = 1/\lambda$ as determined above. Thus the variance becomes $\mathrm{Var}(X) = \mathbb{E}(X^2) - [\mathbb{E}(X)]^2 = 2/\lambda^2 - 1/\lambda^2 = 1/\lambda^2$.

2. First, note that the cumulative distribution function $F(x)$ of the exponential distribution is given by $F(x) = 1 - \exp(-\lambda x)$ for $x > 0$ and zero otherwise. Now, the median value is a value of $x = x_{0.5}(f)$ such that the probability below this value is 0.5 and the probability above this value is 0.5. Thus $F(x_{0.5}(f)) = 0.5$. Solving this equation results in $1 - \exp(-\lambda x_{0.5}(f)) = 0.5$, which means that $-\lambda x_{0.5}(f) = \ln(0.5)$. Thus we obtain that $x_{0.5}(f) = -\ln(0.5)/\lambda \approx 0.693/\lambda$.

4.6

1. First, note that the function $f(x)$ is zero outside the interval $(0,1]$. We have that $f(x) \geq 0$ and the integral is $\int_{\mathbb{R}} f(x)dx = \int_0^1 3x^2 dx = \left[x^3\right]_0^1 = 1$. Thus f is a density.
2. The mean is $\mathbb{E}(X) = \int_0^1 xf(x)\,dx = \int_0^1 3x^3 dx = \left[\frac{3}{4}x^4\right]_0^1 = \frac{3}{4}$. The second moment is $\mathbb{E}(X^2) = \int_0^1 x^2 f(x)dx = \int_0^1 3x^4 dx = \left[\frac{3}{5}x^5\right]_0^1 = \frac{3}{5}$. The variance is then $\mathrm{Var}(X) = \mathbb{E}(X^2) - [\mathbb{E}(X)]^2 = \frac{3}{5} - \frac{9}{16} = \frac{3}{80}$ and the standard deviation is given by $\sqrt{\frac{3}{80}}$.
3. The cumulative distribution function $F(x)$ is given by $F(x) = x^3$ for $x \in (0,1]$. The probability then becomes $F(0.75) - F(0.25) = \left(\frac{3}{4}\right)^3 - \left(\frac{1}{4}\right)^3 = \frac{27}{64} - \frac{1}{64} = \frac{26}{64} = \frac{13}{32}$.

4.7 First, recall the Taylor series for the exponential function: $\exp(x) = \sum_{k=0}^{\infty} \left(x^k/k!\right)$. From this it follows that $\sum_{x=0}^{\infty} (\lambda^x/x!)\exp(-\lambda) = 1$. The mean of the Poisson distribution is then equal to

$$\begin{aligned}\mathbb{E}(X) &= \sum_{x=0}^{\infty} x\frac{\lambda^x}{x!}\exp(-\lambda) = \sum_{x=1}^{\infty} x\frac{\lambda^x}{x!}\exp(-\lambda) = \sum_{x=1}^{\infty} \frac{\lambda^x}{(x-1)!}\exp(-\lambda) \\ &= \lambda\sum_{x=1}^{\infty} \frac{\lambda^{(x-1)}}{(x-1)!}\exp(-\lambda) = \lambda\sum_{x=0}^{\infty} \frac{\lambda^x}{x!}\exp(-\lambda) = \lambda.\end{aligned}$$

For the variance, it is easier to first calculate $\mathbb{E}[X(X-1)] = \mathbb{E}(X^2) - \mathbb{E}(X) = \mathbb{E}(X^2) - \lambda$. This expectation is equal to

$$\begin{aligned}\mathbb{E}[X(X-1)] &= \sum_{x=0}^{\infty} x(x-1)\frac{\lambda^x}{x!}\exp(-\lambda) = \sum_{x=2}^{\infty} x(x-1)\frac{\lambda^x}{x!}\exp(-\lambda) \\ &= \sum_{x=2}^{\infty} \frac{\lambda^x}{(x-2)!}\exp(-\lambda) = \lambda^2\sum_{x=2}^{\infty} \frac{\lambda^{(x-2)}}{(x-2)!}\exp(-\lambda) \\ &= \lambda^2\sum_{x=0}^{\infty} \frac{\lambda^x}{x!}\exp(-\lambda) = \lambda^2.\end{aligned}$$

From this it follows that $\mathbb{E}(X^2) = \lambda^2 + \lambda$. The variance is now given by $\text{Var}(X) = \mathbb{E}(X^2) - [\mathbb{E}(X)]^2 = \lambda^2 + \lambda - \lambda^2 = \lambda$.

4.8

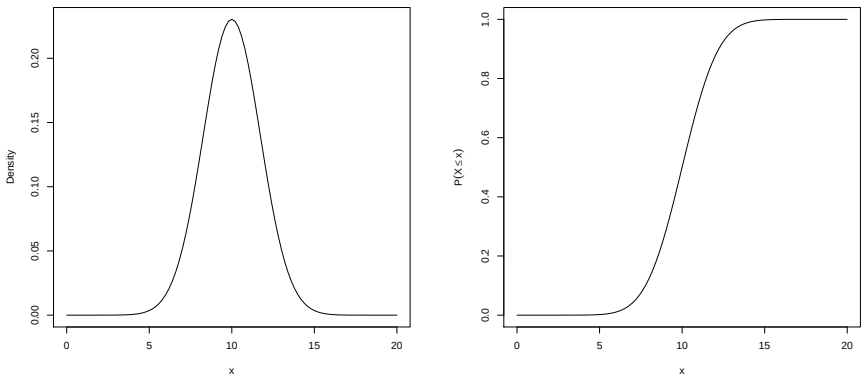
1. The plots can be produced using the following [R] code:

```
> mu <- 10
> sigma2 <- 3
> curve(dnorm(x, mean=mu, sd=sqrt(sigma2)), from=0, to=20,
    ylab="Density")
> curve(pnorm(x, mean=mu, sd=sqrt(sigma2)), from=0, to=20,
    ylab=expression(P(X <= x)))
```

   The PDF is shown in Figure 4.2a and the CDF in Figure 4.2b.
2. The following [R] code approximates the expected value and variance of the normal distribution $\mathcal{N}(\mu = 10, \sigma^2 = 3)$ using $10^6 = 1{,}000{,}000$ simulated draws:

```
> set.seed(274)
> x <- rnorm(10^6, mean=mu, sd=sqrt(sigma2))
> mean(x)
[1] 9.999347
> var(x)
[1] 3.002463
```

4.9 First, note that the cumulate distribution function is given by $F(x) = \frac{1}{4}x^2$ for $x \in (0,2]$. In order to be able to perform inverse transform sampling, we need to determine the inverse of this CDF. Denote $F(x) = p$. Then $p = \frac{1}{4}x^2$, which implies that



(a) (b)

Fig. 4.2: Plots for part 1 of Problem 4.8.

$4p = x^2$, giving us $x = \sqrt{4p} = 2\sqrt{p}$. Thus, the inverse CDF (or quantile function) is given by $F^{-1}(p) = 2\sqrt{p}$. We can now implement inverse transform sampling for the distribution $f(x) = \frac{1}{2}x$ using the [R] code from Section 4.8.3:

```
> cdf_inverse <- function(p) {
+  2*sqrt(p)
+ }
>
> rfx <- function(n) {
+  p <- runif(n, min=0, max=1)
+  cdf_inverse(p)
+ }
>
> set.seed(274)
> n <- 10^6
> draws <- rfx(n)
> hist(draws, freq=FALSE, breaks=40)
> curve(1/2*x, col="red", add=TRUE)
```

The histogram of the draws with the superimposed red curve of the PDF in Figure 4.3 shows that our inverse transform sampling implementation worked properly.

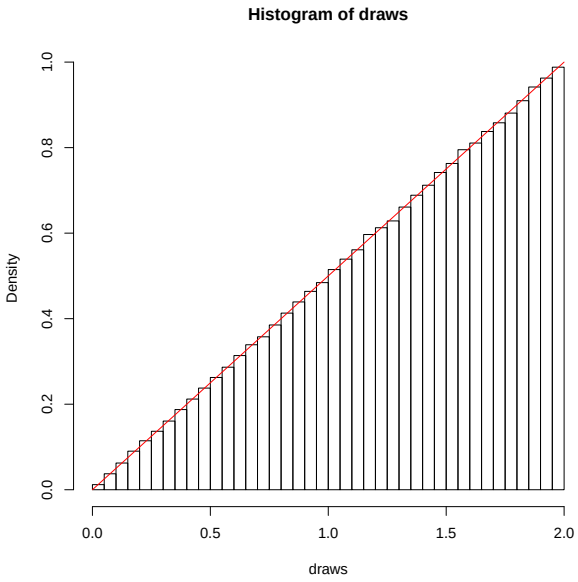


Fig. 4.3: Plot for Problem 4.9.

Chapter 5
Solutions for Chapter 5

5.1

1. The relative standard deviation is given by the ratio of the standard deviation and the mean, i.e., $RSD = \sqrt{\text{Var}(X)}/\mathbb{E}(X)$. The mean and variance can be calculated using the appropriate integrals as in the solutions to Problem 4.5. There it was found that the mean is given by $\mathbb{E}(X) = 1/\lambda$ and the variance by $\text{Var}(X) = 1/\lambda^2$. Thus, the relative standard deviation is equal to $RSD = \sqrt{1/\lambda^2}/(1/\lambda) = 1$.
2. The pth quantile is given by $F^{-1}(p)$, with F^{-1} the inverse distribution function. In other words, we are looking for a value q such that $F(q) = p$. Substituting these values in the exponential distribution function leads to $p = 1 - \exp(-\lambda q)$. Solving this equation for q results in $q = -\ln(1-p)/\lambda$.
3. The skewness is given by $\gamma_1 = \mathbb{E}(X-\mu)^3/\sigma^3$, with $\mu = \mathbb{E}(X)$ and $\sigma^2 = \mathbb{E}(X - \mathbb{E}(X))^2$. Removing the brackets from the third central moment leads to

$$\begin{aligned}\mathbb{E}(X-\mu)^3 &= \mathbb{E}\left(X^3 - 3X^2\mu + 3X\mu^2 - \mu^3\right)\\ &= \mathbb{E}(X^3) - 3\mu\mathbb{E}(X^2) + 3\mu^2\mathbb{E}(X) - \mu^3\\ &= \mathbb{E}(X^3) - 3\mu\mathbb{E}(X^2) + 3\mu^3 - \mu^3\\ &= \mathbb{E}(X^3) - 3\mu\mathbb{E}(X^2) + 2\mu^3.\end{aligned}$$

Now, the third moment is given by

$$\begin{aligned}\mathbb{E}(X^3) &= \int_0^\infty x^3\lambda\exp(-\lambda x)dx\\ &= \left[-\frac{1}{\lambda}\exp(-\lambda x)x^3\lambda\right]_0^\infty - \int_0^\infty -\frac{1}{\lambda}\exp(-\lambda x)3\lambda x^2 dx\\ &= \left[-x^3\exp(-\lambda x)\right]_0^\infty + \frac{3}{\lambda}\int_0^\infty x^2\lambda\exp(-\lambda x)dx\\ &= 0 + \frac{3}{\lambda}\cdot\frac{2}{\lambda^2} = \frac{6}{\lambda^3},\end{aligned}$$

where the integral in the third line is the second moment $\mathbb{E}(X^2) = 2/\lambda^2$. Thus, the third central moment is equal to $\mathbb{E}(X-\mu)^3 = 6/\lambda^3 - 6/\lambda^3 + 2/\lambda^3 = 2/\lambda^3$. Then the skewness becomes $\gamma_1 = \left[2/\lambda^3\right] / \left[(1/\lambda)^3\right] = 2$. Hence the skewness is independent of the parameter λ.
The excessive kurtosis is given by $\gamma_2 = \mathbb{E}(X-\mu)^4/\sigma^4 - 3$. Again removing the bracket, but now for the fourth central moment, we obtain

$$\mathbb{E}(X-\mu)^4 = \mathbb{E}(X^4) - 4\mu\mathbb{E}(X^3) + 6\mu^2\mathbb{E}(X^2) - 3\mu^4.$$

The fourth moment is given by

$$\mathbb{E}(X^4) = \left[-x^4\exp(-\lambda x)\right]_0^\infty + \frac{4}{\lambda}\int_0^\infty x^3\lambda\exp(-\lambda x)dx = 24/\lambda^4.$$

The fourth central moment is now $\mathbb{E}(X-\mu)^4 = 24/\lambda^4 - 24/\lambda^4 + 12/\lambda^4 - 3/\lambda^4 = 9/\lambda^4$. The excessive kurtosis then is $\gamma_2 = \left[9/\lambda^4\right] / \left[(1/\lambda)^4\right] - 3 = 6$, which is also independent of the parameter λ.

4. The maximum likelihood estimator for λ is given by $\hat{\lambda} = 1/\bar{X} = n/\sum_{i=1}^n X_i$, where $\bar{X} = \sum_{i=1}^n X_i/n$ is the sample mean. The maximum likelihood estimate $\hat{\lambda}$ for the 10 observations can be computed in [R] as follows:

```
> x <- c(0.05, 0.20, 1.72, 0.61, 0.24, 0.79, 0.13, 0.59,
    0.26, 0.54)
> xbar <- mean(x)
> lambdahat <- 1/xbar
> lambdahat
[1] 1.949318
```

5.2 Below we provide the [R] code to compute the requested quantities for the variable `AGE`. The quantities for the other numerical variables can be computed similarly.

1. Note that [R] does not have built-in functions for computing the skewness and kurtosis, hence in the following code we use the functions `skewness()` and `kurtosis()` from the package `e1071`:

```
> library(e1071)
> high_school <- read.csv("high-school.csv")
> colnames(high_school) # Print a list of the variable names
    in high_school.
 [1] "Nr"    "CLASS"  "GENDER" "AGE"    "HEIGHT" "SPORTS" "TV"
          "COMPUTER" "SUBJECT"
[10] "ALLOWANCE" "WORK" "PROVINCE" "BREAKFAST"
> hs_variable <- high_school$AGE # Extract variable of
    interest and store in hs_variable.
>
> n <- nrow(high_school)
> mean(hs_variable)
[1] 12.67477
> var(hs_variable)
[1] 0.5848034
```

```
> skewness(hs_variable, type=3)
[1] 1.107057
> kurtosis(hs_variable, type=3)
[1] 5.387464
```

2. Note that the 95% confidence interval for the mean is calculated as

$$\left(\bar{X} - z_{0.975}\sqrt{S^2/n},\ \bar{X} + z_{0.975}\sqrt{S^2/n}\right],$$

with $\bar{X}$ the sample average, S^2 the sample variance, and $z_{0.975}$ the 0.975th quantile of the standard normal distribution. These calculations can be carried out in [R] as follows:

```
> lower <- mean(hs_variable) - qnorm(1-0.025, mean=0, sd=1)*
    sqrt(var(hs_variable)/n)
> upper <- mean(hs_variable) + qnorm(1-0.025, mean=0, sd=1)*
    sqrt(var(hs_variable)/n)
> CI_mean <- c(lower, upper)
> CI_mean
[1] 12.66807 12.68147
```

A 95% confidence interval for the variance is given by

$$\left((n-1)S^2/x_{0.025}\left(f_{\chi^2}\right),\ (n-1)S^2/x_{0.975}\left(f_{\chi^2}\right)\right],$$

where $x_p\left(f_{\chi^2}\right)$ is the pth quantile of the chi-square distribution with $n-1$ degrees of freedom. That is, if X is a random variable having the chi-square distribution with $n-1$ degrees of freedom, then $P\left(X \geq x_p\left(f_{\chi^2}\right)\right) = p$. In [R] the confidence interval can be calculated using the following code:

```
> lower <- (n-1)*var(hs_variable)/qchisq(0.025, df=n-1, lower
    .tail=FALSE)
> upper <- (n-1)*var(hs_variable)/qchisq(1-0.025, df=n-1,
    lower.tail=FALSE)
> CI_var <- c(lower, upper)
> CI_var
[1] 0.5776262 0.5921158
```

Note that we use `lower.tail=FALSE` because $P\left(X \geq x_p\left(f_{\chi^2}\right)\right)$ is an upper tail probability.

3.
```
> quantile(hs_variable, probs=0.2)
20%
 12
```

4.
```
> high_school$NOSPORT <- as.numeric(high_school$SPORT==0) #
    Create dichotomous variable indicating whether a child
    sports (NOSPORT = 0) or does not sport (NOSPORT = 1).
> prop_nosport <- mean(high_school$NOSPORT)
> var_nosport <- prop_nosport*(1-prop_nosport)
> lower <- prop_nosport - qnorm(1-0.025, mean=0, sd=1)*sqrt(
    var_nosport/n)
```

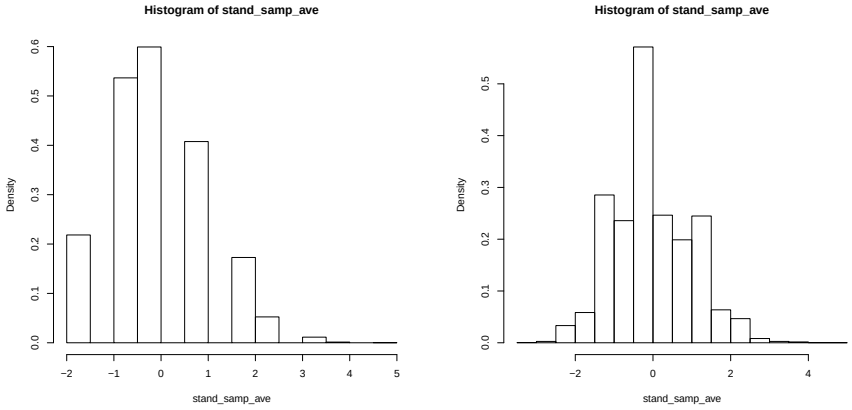
```
> upper <- prop_nosport + qnorm(1-0.025, mean=0, sd=1)*sqrt(
    var_nosport/n)
> CI_prop <- c(lower, upper)
> CI_prop
[1] 0.1059440 0.1113961
```

5.3

1.
```
> set.seed(19680912)
> n <- 10
> p <- 0.2
> SE <- sqrt((p*(1-p)/n))
> nsim <- 10000 # Number of simulations.
> stand_samp_ave <- rep(NA, nsim)
> for (i in 1:nsim){
+  x <- rbinom(n, size=1, prob=p)
+  xbar <- mean(x)
+  stand_samp_ave[i] <- (xbar-p)/SE
+ }
> hist(stand_samp_ave, freq=FALSE)
> mean(stand_samp_ave)
[1] -0.003636619
> var(stand_samp_ave)
[1] 1.001712
```

   The average is close to 0 and the standard deviation is close to 1. The histogram in Figure 5.1a, however, does not look much like a standard normal distribution.

2.
```
> set.seed(19680912)
> n <- 10
> p <- 0.2
```



(a) (b)

Fig. 5.1: Plots for Problem 5.3.

```
> nsim <- 10000
> counter <- 0
> for (i in 1:nsim) {
+  x <- rbinom(n, size=1, prob=p)
+  xbar <- mean(x)
+  var_x <- xbar*(1-xbar)
+  CI_lower <- xbar - qnorm(1-0.025, mean=0, sd=1)*sqrt(var_x
    /n)
+  CI_upper <- xbar + qnorm(1-0.025, mean=0, sd=1)*sqrt(var_x
    /n)
+  if (CI_lower<p & CI_upper>p) counter <- counter+1
+ }
> counter/nsim
[1] 0.8843
```

The asymptotic confidence interval is substantially lower than 0.95. The confidence level is close to 0.884. The reason is that the asymptotic normality does not yet hold properly when only 10 binary variables are averaged.

3. We can use the [R] code from the solution to part 1 of Problem 5.3. We only need to replace `n <- 10` with `n <- 50` in the code. This gives the following results:

```
> mean(stand_samp_ave)
[1] -0.006151829
> var(stand_samp_ave)
[1] 0.9949366
```

Thus, the mean is again close to zero and the variance is again close to 1. From the histogram in Figure 5.1b it can be seen that for the average of 50 binary variables the normality seems to be much better.

5.4

1. We can make use of the [R] code from the solution to part 2 of Problem 5.3. Note that we need to adjust the code in a few places to account for the fact that we now sample from the exponential distribution:

```
> set.seed(19680912)
> n <- 3
> lambda <- 0.5
> nsim <- 10000
> counter <- 0
> for (i in 1:nsim) {
+  x <- rexp(n, rate=0.5)
+  xbar <- mean(x)
+  var_x <- var(x)
+  CI_lower <- xbar - qnorm(1-0.025, mean=0, sd=1)*sqrt(var_x
    /n)
+  CI_upper <- xbar + qnorm(1-0.025, mean=0, sd=1)*sqrt(var_x
    /n)
+  if (CI_lower<(1/lambda) & CI_upper>(1/lambda)) counter <-
    counter+1
+ }
> counter/nsim
[1] 0.7367
```

The asymptotic confidence interval for the average of just 3 exponentially distributed data points is approximately 0.736. Thus the asymptotic normality for $n = 3$ is not good.

2. Replacing `n <- 3` with `n <- 20` in the [R] code above yields the following results:

```
> counter/nsim
[1] 0.9086
```

In case the sample size is $n = 20$, the confidence level is getting closer to the nominal value 0.95, since it is 0.909.

3. The larger sample size in part 2 makes the normality approximation better, which explains the increase in confidence level.
4. There are two possible ways of calculating a 95% confidence interval for the variance $\mathrm{Var}(X) = 1/\lambda^2$. The first is to assume that the data $X_1, X_2, \ldots, X_{20}$ come from a normal distribution (even though they do not). The 95% confidence interval is then calculated by

$$\left((n-1)S^2/x_{0.025}\left(f_{\chi^2}\right),\ (n-1)S^2/x_{0.975}\left(f_{\chi^2}\right)\right],$$

with $x_p\left(f_{\chi^2}\right)$ the pth quantile of the chi-square distribution with $n-1 = 19$ degrees of freedom. This approach to computing a confidence interval for the variance can be implemented in [R] as follows:

```
> set.seed(19680912)
> n <- 20
> lambda <- 0.5
> nsim <- 10000
> counter <- 0
> for (i in 1:nsim) {
+  x <- rexp(n, rate=0.5)
+  var_x <- var(x)
+  CI_lower <- (n-1)*var_x/qchisq(0.025, df=n-1, lower.tail=
    FALSE)
+  CI_upper <- (n-1)*var_x/qchisq(1-0.025, df=n-1, lower.tail
    =FALSE)
+  if (CI_lower<(1/lambda^2) & CI_upper>(1/lambda^2)) counter
     <- counter+1
+ }
> counter/nsim
[1] 0.7383
```

Thus, the confidence interval based on the chi-square distribution has a coverage probability of about 0.738.

The second approach to computing a confidence interval for the variance is to use an asymptotic confidence interval that is based on asymptotic normality. The sample variance also converges to a normal distribution. The approximate 95% confidence interval is then calculated by

$$\left(S^2 - z_{0.975} \cdot S^2\sqrt{(b_2+2)/n},\ S^2 + z_{0.975} \cdot S^2\sqrt{(b_2+2)/n}\right],$$

with b_2 the sample excessive kurtosis. The following [R] code implements this confidence interval approach:

```
> library(e1071) # We use the kurtosis() function from
    package e1071.
> set.seed(19680912)
> n <- 20
> lambda <- 0.5
> nsim <- 10000
> counter <- 0
> for (i in 1:nsim) {
+  x <- rexp(n, rate=0.5)
+  var_x <- var(x)
+  b2 <- kurtosis(x, type=3)
+  SE <- var_x*sqrt((b2+2)/n)
+  CI_lower <- var_x - qnorm(1-0.025, mean=0, sd=1)*SE
+  CI_upper <- var_x + qnorm(1-0.025, mean=0, sd=1)*SE
+  if (CI_lower<(1/lambda^2) & CI_upper>(1/lambda^2)) counter
     <- counter+1
+ }
> counter/nsim
[1] 0.6986
```

The result shows that the confidence interval based on asymptotic normality has a coverage probability of about 0.699.

5. In the simulations above we counted how often the two confidence intervals contained the true variance $1/\lambda^2 = 4$. The confidence levels from the simulations are 0.738 for the interval based on the chi-square distribution and 0.699 for the interval based on asymptotic normality. Both procedures do worse than the confidence interval on the mean $1/\lambda$ since the distributional approximations for the variance are less precise. It should be noted that when the sample size n increases, the confidence interval for the variance based on asymptotic normality would do much better than the chi-square based interval.

5.5 In this problem the random variable X has an exponential distribution $F_E(x) = 1 - \exp(-\lambda(x-\eta))$ for $x > \eta$ and otherwise equal to zero. Note that the probability density function is $dF_E(x)/dx = \lambda \exp(-\lambda(x-\eta))I(x > \eta)$, with $I(x > \eta)$ an indicator variable that is equal to 1 if $x > \eta$ and zero otherwise.

1. For $\eta = 0$ we have already determined the first moment $\mathbb{E}(X)$ in the solution to part 1 of Problem 4.5. The first moment is $\mathbb{E}(X) = 1/\lambda$. Setting this first moment equal to the sample mean $\bar{X}$ yields $\bar{X} = 1/\lambda$, which results in the moment estimator $\tilde{\lambda} = 1/\bar{X}$ for the parameter λ.
2. For $\eta = 0$ the likelihood function is $L(\lambda) = \prod_{i=1}^{n} \lambda \exp(-\lambda X_i)$ and the log likelihood function is

$$\begin{aligned}\ell(\lambda) = \ln\big(L(\lambda)\big) &= \sum_{i=1}^{n} \ln\big[\lambda \exp(-\lambda X_i)\big] = \sum_{i=1}^{n} \big[\ln(\lambda) - \lambda X_i\big] \\ &= n\ln(\lambda) - \lambda \sum_{i=1}^{n} X_i = n\ln(\lambda) - \lambda n\bar{X},\end{aligned}$$

where $\bar{X} = \sum_{i=1}^{n} X_i/n$ is the sample mean. Setting the derivative of the log likelihood function with respect to λ equal to zero results in $d\ell(\lambda)/d\lambda = n/\lambda - n\bar{X} = 0$. Solving this equation for λ yields the maximum likelihood estimator $\hat{\lambda} = 1/\bar{X}$. Thus, the maximum likelihood estimator is the same as the moment estimator.

3. To obtain the moment estimators for λ and η we need to determine the first two moments. The first moment is

$$\begin{aligned}\mathbb{E}(X) &= \int_{\eta}^{\infty} x\lambda \exp\big[-\lambda(x-\eta)\big]dx = \int_{0}^{\infty} (x+\eta)\lambda \exp(-\lambda x)dx \\ &= \int_{0}^{\infty} x\lambda \exp(-\lambda x)dx + \eta \int_{0}^{\infty} \lambda \exp(-\lambda x)dx = \frac{1}{\lambda} + \eta.\end{aligned}$$

The second moment (using some of the results from the solution to part 1 of Problem 4.5) is given by

$$\begin{aligned}\mathbb{E}(X^2) &= \int_{\eta}^{\infty} x^2\lambda \exp\big[-\lambda(x-\eta)\big]dx \\ &= \int_{0}^{\infty} (x+\eta)^2\lambda \exp(-\lambda x)dx = \int_{0}^{\infty} \big(x^2+2x\eta+\eta^2\big)\lambda \exp(-\lambda x)dx \\ &= \int_{0}^{\infty} x^2\lambda \exp(-\lambda x)dx + 2\eta\int_{0}^{\infty} x\lambda \exp(-\lambda x)dx + \eta^2\int_{0}^{\infty} \lambda \exp(-\lambda x)dx \\ &= \frac{2}{\lambda^2} + \frac{2\eta}{\lambda} + \eta^2 = \frac{1}{\lambda^2} + \left(\frac{1}{\lambda}+\eta\right)^2.\end{aligned}$$

Thus, the variance of X is now $\text{Var}(X) = \mathbb{E}(X^2) - [\mathbb{E}(X)]^2 = 1/\lambda^2$. Equating this second population moment to the second sample moment S^2 (i.e., the sample variance) yields $1/\lambda^2 = S^2$. Solving this equation for λ gives us the moment estimator $\tilde{\lambda} = 1/S$. The moment estimator for η can now be determined by equating the first population moment to the first sample moment (i.e., the sample mean): $1/\lambda + \eta = S + \eta = \bar{X}$, where we substituted λ with its estimator $1/S$. Solving for η then gives us $\tilde{\eta} = \bar{X} - S$ as the moment estimator.

4. First of all, note that all variables $X_1, X_2, \ldots, X_n$ should be larger than η, and thus the minimum value $X_{(1)}$ should be larger than η. The likelihood function is now

$$\begin{aligned}L(\lambda,\eta) &= \prod_{i=1}^{n} \lambda \exp\big[-\lambda(X_i-\eta)\big] I(X_i > \eta) \\ &= \prod_{i=1}^{n} \lambda \exp\big[-\lambda(X_i - X_{(1)} + X_{(1)} - \eta)\big] I(X_{(1)} > \eta).\end{aligned}$$

The log likelihood is

$$\ell(\lambda,\eta) = \sum_{i=1}^{n} \big[\ln(\lambda) - \lambda(X_i - X_{(1)}) - \lambda(X_{(1)} - \eta) + \ln(I(X_{(1)} > \eta))\big].$$

Even though $\eta \leq X_{(1)}$, it should be noted that relaxing this boundary does not help, since the log likelihood becomes $-\infty$ when $\eta > X_{(1)}$. Furthermore, if we fix the $\lambda > 0$, the log likelihood will become maximal when $\eta = X_{(1)}$, since values for $\eta = X_{(1)} - \varepsilon$, with $\varepsilon > 0$, will have a log likelihood that is $\lambda\varepsilon$ smaller than when we use $\eta = X_{(1)}$. Thus the maximum likelihood estimator for η is now $\hat{\eta} = X_{(1)}$. Substituting this into the log likelihood gives us a new log likelihood for $\lambda > 0$ only:

$$\ell(\lambda, X_{(1)}) = \sum_{i=1}^{n} \big[\ln(\lambda) - \lambda(X_i - X_{(1)})\big].$$

Setting the derivative of this log likelihood with respect to λ equal to zero, we find $\sum_{i=1}^{n}(1/\lambda - X_i + X_{(1)}) = n/\lambda - n\bar{X} + nX_{(1)} = 0$, where $\bar{X} = \sum_{i=1}^{n} X_i/n$ is the sample mean. Solving for λ results in the maximum likelihood estimator $\hat{\lambda} = 1/(\bar{X} - X_{(1)})$. Note that the maximum likelihood estimators are now truly different from the moment estimators.

5.6

1. The observations are assumed to come from a gamma distribution $G(\alpha, \beta)$. Note that the gamma function $\Gamma(\alpha) = \int_0^\infty x^{\alpha-1}\exp(-x)dx$ and that $\Gamma(\alpha+1) = \alpha\Gamma(\alpha)$. To determine the moment estimators, we first need to calculate the first and second moment. The first moment is equal to

$$\begin{aligned}\mathbb{E}(X) &= \int_0^\infty x\frac{\beta^\alpha}{\Gamma(\alpha)}x^{\alpha-1}\exp(-\beta x)dx = \frac{\beta^\alpha}{\Gamma(\alpha)}\int_0^\infty x^\alpha\exp(-\beta x)dx \\ &\overset{z=\beta x}{=} \frac{\beta^\alpha}{\Gamma(\alpha)}\int_0^\infty \left(\frac{z}{\beta}\right)^\alpha \exp(-z)\frac{1}{\beta}dz = \frac{1}{\beta\Gamma(\alpha)}\int_0^\infty z^\alpha\exp(-z)dz \\ &= \frac{1}{\beta\Gamma(\alpha)}\int_0^\infty z^{(\alpha+1)-1}\exp(-z)dz = \frac{\Gamma(\alpha+1)}{\beta\Gamma(\alpha)} = \frac{\alpha\Gamma(\alpha)}{\beta\Gamma(\alpha)} = \frac{\alpha}{\beta},\end{aligned}$$

where in the second line we use the change of variable $z = \beta x$. The second moment can now be determined as

$$\begin{aligned}\mathbb{E}(X^2) &= \int_0^\infty x^2\frac{\beta^\alpha}{\Gamma(\alpha)}x^{\alpha-1}\exp(-\beta x)dx \overset{z=\beta x}{=} \frac{\beta^\alpha}{\Gamma(\alpha)}\int_0^\infty \left(\frac{z}{\beta}\right)^{\alpha+1}\exp(-z)\frac{1}{\beta}dz \\ &= \frac{1}{\beta^2\Gamma(\alpha)}\int_0^\infty z^{(\alpha+2)-1}\exp(-z)dz = \frac{\Gamma(\alpha+2)}{\beta^2\Gamma(\alpha)} = \frac{\alpha(\alpha+1)}{\beta^2}.\end{aligned}$$

The variance of a gamma distributed random variable X is now $\text{Var}(X) = \mathbb{E}(X^2) - [\mathbb{E}(X)]^2 = (\alpha^2+\alpha)/\beta^2 - \alpha^2/\beta^2 = \alpha/\beta^2$. This means that $\beta = (\alpha/\beta)/(\alpha/\beta^2) = \mathbb{E}(X)/\text{Var}(X)$. This means that the moment estimator for β is given by $\tilde{\beta} = \bar{X}/S^2$. Equating the first population moment to the first sample moment results in $\alpha/\beta = \bar{X}$. The moment estimator for α is then given by $\tilde{\alpha} = \tilde{\beta}\bar{X} = (\bar{X})^2/S^2$.

2. `> high_school <- read.csv("high-school.csv")`

```
> new_data <- high_school[high_school$SPORTS>0, ]
> n <- nrow(new_data)
> xbar <- mean(new_data$SPORTS)
> s2 <- (n-1)*var(new_data$SPORTS)/n
> alphatilde <- xbar^2/s2
> betatilde <- xbar/s2
> alphatilde
[1] 1.147473
> betatilde
[1] 0.2308539
```

3. Maximum likelihood estimates can be computed numerically in [R] using either the function `mle()` from the package `stats4` or the function `fitdistr()` from the package `MASS`:

```
> # Using the function mle() from the package stats4:
> library(stats4)
> minuslogl <- function(alpha, beta) {
+  densities <- dgamma(new_data$SPORTS, shape=alpha, rate=
    beta)
+  -sum(log(densities))
+ }
> mle(minuslogl, start=list(alpha=1, beta=0.5), method="L-
    BFGS-B", lower=c(0, 0))

Call:
mle(minuslogl = minuslogl, start = list(alpha = 1, beta =
    0.5),
   method = "L-BFGS-B", lower = c(0, 0))

Coefficients:
   alpha    beta
1.9469556 0.3916986
>
> # Using the function fitdistr() from the package MASS:
> library(MASS)
> fitdistr(new_data$SPORTS, densfun="gamma", start=list(shape
    =1, rate=0.5), lower=c(0, 0))
     shape      rate
  1.946955581 0.391698586
 (0.012083274) (0.002770431)
```

Note that in both function calls we use the argument `lower=c(0, 0)` to indicate that both α and β have a lower bound of 0 (i.e., the two parameters must be positive).

4.
```
> nsim <- 100 # Number of simulations.
> m <- 1000 # Number of children.
> mle_est <- matrix(NA, nrow=nsim, ncol=2)
> mme_alpha <- rep(NA, times=nsim)
> mme_beta <- rep(NA, times=nsim)
>
> set.seed(3746347)
> for (i in 1:nsim) {
+  new_data$id <- 1:nrow(new_data) # Create id variable.
```

```
+  sample_id <- sample(new_data$id, size=m, replace=FALSE)
+  x <- new_data[sample_id, "SPORTS"]
+  mle_sport <- fitdistr(x, densfun="gamma", start=list(shape
    =1, rate=0.5), lower=c(0, 0))
+  mle_est[i, ] <- coef(mle_sport) # Store the MLE for each
    sample.
+  xbar <- mean(x)
+  s2 <- (m-1)*var(x)/m
+  mme_alpha[i] <- xbar^2/s2
+  mme_beta[i] <- xbar/s2
+ }
>
> c(mean(mle_est[, 1]), var(mle_est[, 1]))
[1] 1.94613684 0.01094735
> c(mean(mle_est[, 2]), var(mle_est[, 2]))
[1] 0.3919579846 0.0008714883
> c(mean(mme_alpha), var(mme_alpha))
[1] 1.15167538 0.03255287
> c(mean(mme_beta), var(mme_beta))
[1] 0.232303305 0.001635654
>
> mean(mle_est[, 1])/mean(mle_est[, 2])
[1] 4.965167
> mean(mme_alpha)/mean(mme_beta)
[1] 4.957637
```

For the simulations we obtained the following average estimates (with the variance of the estimates in between brackets):

$$\begin{aligned}
&\text{MLE: } \hat{\alpha} = 1.946 \ (0.0110)\\
&\text{MME: } \tilde{\alpha} = 1.152 \ (0.0326)\\
&\text{MLE: } \hat{\beta} = 0.392 \ (0.0009)\\
&\text{MME: } \tilde{\beta} = 0.232 \ (0.0016)
\end{aligned}$$

The simulation shows that the average estimates are quite different between the two methods of estimation, though it should be noted that the ratio $\hat{\alpha}/\hat{\beta} = 4.965$ is quite close to $\tilde{\alpha}/\tilde{\beta} = 4.958$. This means that the first moment of the gamma distribution is similar for both estimation methods. The simulation also shows that the variability in the estimates from simulation to simulation is larger for the moment estimates than for the maximum likelihood estimates. Based on this variability, it is probably better to use the maximum likelihood estimators.

Chapter 6
Solutions for Chapter 6

6.1

1. ```
   > voting <- read.csv("voting-demo.csv")
   > tab <- table(voting$Vote, voting$Choice)
   > margtab <- addmargins(tab)
   > margtab

              CDU/CSU FDP SPD Sum
     Did not vote 102 45 99 246
     Did vote     230 86 186 502
     Sum          332 131 285 748
   ```

2. ```
   > # Calculate chi squared:
   > chi2 <- unname(chisq.test(tab)$statistic)
   > chi2
   [1] 1.273555
   > # Calculate Phi:
   > n <- sum(tab) # Calculate sample size.
   > Phi <- sqrt(chi2/n)
   > Phi
   [1] 0.04126273
   > # Calculate Cramer's V:
   > t <- min(nrow(tab)-1, ncol(tab)-1)
   > V <- sqrt(chi2/(n*t))
   > V
   [1] 0.04126273
   ```

   We can also calculate χ^2 by hand. Here is some [R] code that mimics how we can compute it by hand:

   ```
   > # cell_ij is the cell in row i and column j:
   > cell_11 <- (102 - 332 * 246 / 748)^2 / (332 * 246 / 748)
   > cell_12 <- (45 - 131 * 246 / 748)^2 / (131 * 246 / 748)
   > cell_13 <- (99 - 285 * 246 / 748)^2 / (285 * 246 / 748)
   > cell_21 <- (230 - 332 * 502 / 748)^2 / (332 * 502 / 748)
   > cell_22 <- (86 - 131 * 502 / 748)^2 / (131 * 502 / 748)
   > cell_23 <- (186 - 285 * 502 / 748)^2 / (285 * 502 / 748)
   ```

```
> chi2 <- cell_11 + cell_12 + cell_13 + cell_21 + cell_22 +
    cell_23
> chi2
[1] 1.273555
```

3. A simple way to visualize the relationship between `Vote` and `Choice` is a grouped bar plot such as this:

   ```
   > barplot(tab, legend.text=rownames(tab), beside=TRUE, xlab="
       Choice", ylab="Number␣of␣persons")
   ```

   The plot is shown in Figure 6.1a.
4. The variable `Age2` is a linear transformation of the variable `Age`. This can be visualized with a plot:

   ```
   > Age <- voting$Age
   > Age2 <- 12*Age
   > plot(Age, Age2)
   ```

   The plot is shown in Figure 6.1b. In this case the sample correlation between `Age` and `Age2` equals 1:

   ```
   > cor(Age, Age2, use="complete.obs")
   [1] 1
   ```

   Note that we use the argument `use="complete.obs"` to omit the observation that has a missing value on `Age` and `Age2`.
5. We can write a little function called `noise()` that takes the standard deviation of the noise (`x`) as input and produces the plot and the correlation as output:

   ```
   > noise <- function(x) {
   +  Age2 <- Age2 + rnorm(length(Age2), mean=0, sd=x)
   +  plot(Age, Age2)
   +  cor(Age, Age2, use="complete.obs")
   + }
   >
   > set.seed(5710)
   > noise(1)
   [1] 0.9999849
   > noise(10)
   [1] 0.9985462
   > noise(100)
   [1] 0.8773142
   ```

   The plots are shown in Figures 6.1c, 6.1d, and 6.1e.

6.2

1. The joint probability mass function of X and Y can be shown in a table like this:

	$Y=0$	$Y=1$	$Y=2$
$X=0$	$\frac{2}{12}$	$\frac{3}{12}$	$\frac{1}{12}$
$X=1$	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{4}{12}$

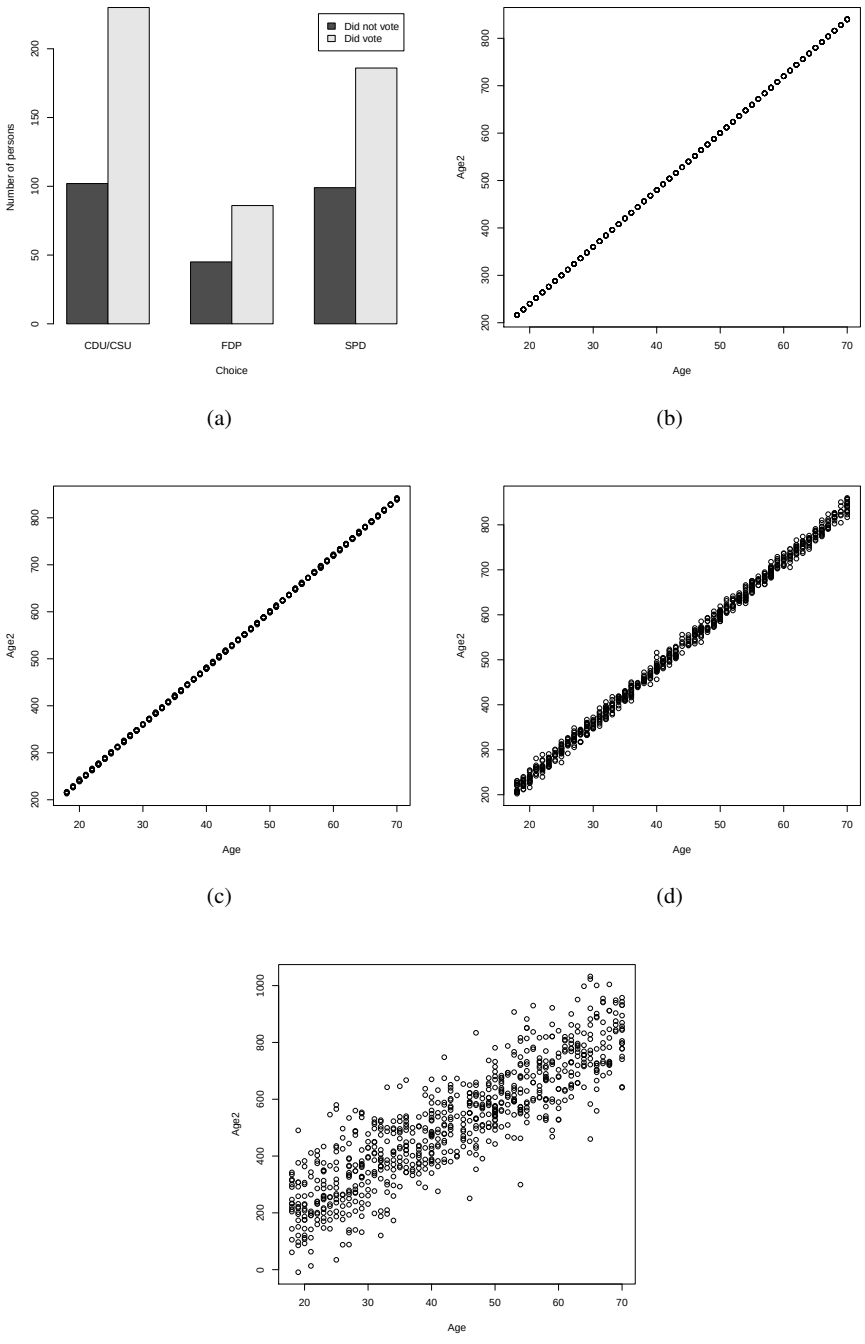


Fig. 6.1: Plots for Problem 6.1.

Note that the ranges of X and Y are given by $R_X = \{0,1\}$ and $R_Y = \{0,1,2\}$, respectively. In order to calculate the marginal expectation of X we first determine the marginal distribution of X. Let $P_X(x) = P(X = x)$ and $P_{XY}(x,y) = P(X = x, Y = y)$. For the joint PMF above we obtain $P_X(0) = \sum_{y_j \in R_Y} P_{XY}(0,y_j) = P_{XY}(0,0) + P_{XY}(0,1) + P_{XY}(0,2) = \frac{2}{12} + \frac{3}{12} + \frac{1}{12} = \frac{6}{12} = \frac{1}{2}$, where y_j is the jth element of R_Y. Similarly, we have that $P_X(1) = P_{XY}(1,0) + P_{XY}(1,1) + P_{XY}(1,2) = \frac{1}{12} + \frac{1}{12} + \frac{4}{12} = \frac{1}{2}$. Note that we just sum the probabilities in the above table row-wise. We can now calculate the marginal expectation of X as $\mathbb{E}(X) = \sum_{x_i \in R_X} x_i P_X(x_i) = 0 \cdot P_X(0) + 1 \cdot P_X(1) = 0 \cdot \frac{1}{2} + 1 \cdot \frac{1}{2} = \frac{1}{2}$, where x_i is the ith element of R_X.

2. We first determine the conditional distribution of Y given $X = 1$. Let $P_{Y|X}(y|x) = P(Y = y|X = x)$. We have that $P_{Y|X}(0|1) = P_{XY}(1,0)/P_X(1) = \frac{1}{12}/\frac{1}{2} = \frac{2}{12} = \frac{1}{6}$. Similarly, we obtain $P_{Y|X}(1|1) = P_{XY}(1,1)/P_X(1) = \frac{1}{12}/\frac{1}{2} = \frac{1}{6}$ and $P_{Y|X}(2|1) = P_{XY}(1,2)/P_X(1) = \frac{4}{12}/\frac{1}{2} = \frac{2}{3}$. The conditional expectation of Y given $X = 1$ is then given by $\mathbb{E}(Y|X = 1) = \sum_{y_j \in R_Y} y_j P_{Y|X}(y_j|1) = 0 \cdot P_{Y|X}(0|1) + 1 \cdot P_{Y|X}(1|1) + 2 \cdot P_{Y|X}(2|1) = 0 \cdot \frac{1}{6} + 1 \cdot \frac{1}{6} + 2 \cdot \frac{2}{3} = \frac{9}{6} = \frac{3}{2}$.

3. First, note that the joint range of X and Y is given by $R_{XY} = R_X \times R_Y = \{0,1\} \times \{0,1,2\}$. Then

$$\begin{aligned}\mathbb{E}(XY) &= \sum_{(x_i,y_j) \in R_{XY}} x_i y_j P_{XY}(x_i,y_j) \\ &= 0 \cdot 0 \cdot P_{XY}(0,0) + 0 \cdot 1 \cdot P_{XY}(0,1) + 0 \cdot 2 \cdot P_{XY}(0,2) + \\ &\qquad 1 \cdot 0 \cdot P_{XY}(1,0) + 1 \cdot 1 \cdot P_{XY}(1,1) + 1 \cdot 2 \cdot P_{XY}(1,2) \\ &= P_{XY}(1,1) + 2 \cdot P_{XY}(1,2) = \tfrac{1}{12} + 2 \cdot \tfrac{4}{12} = \tfrac{9}{12} = \tfrac{3}{4}.\end{aligned}$$

4. Two discrete random variables X and Y are independent if $P_{XY}(x,y) = P_X(x)P_Y(y)$. Now, $P_Y(0) = \sum_{x_i \in R_X} P_{XY}(x_i,0) = P_{XY}(0,0) + P_{XY}(1,0) = \frac{2}{12} + \frac{1}{12} = \frac{1}{4}$ and from part 1 we know that $P_X(0) = \frac{1}{2}$. Then $P_{XY}(0,0) = \frac{2}{12} = \frac{1}{6} \neq P_X(0)P_Y(0) = \frac{1}{2} \cdot \frac{1}{4} = \frac{1}{8}$, which implies that X and Y are not independent.

6.3

1. Let us first approach the question mathematically. We have

$$\begin{aligned}COV(X,Y) &= \mathbb{E}((X - \mathbb{E}(X))(Y - \mathbb{E}(Y))) = \mathbb{E}((X - \mathbb{E}(X))(X^2 - \mathbb{E}(X^2)) \\ &= \mathbb{E}(X(X^2 - \mathbb{E}(X^2)),\end{aligned}$$

where the last equality follows since $\mathbb{E}(X) = 0$. Next, from $\mathbb{E}(X) = 0$, $\mathrm{Var}(X) = 1$, and $\mathrm{Var}(X) = \mathbb{E}(X^2) - (\mathbb{E}(X))^2$ it follows that

$$\mathbb{E}(X^2) = \mathrm{Var}(X) + (\mathbb{E}(X))^2 = 1 + 0 = 1.$$

Therefore

$$COV(X,Y) = \mathbb{E}(X(X^2 - 1)) = \mathbb{E}(X^3 - X) = \mathbb{E}(X^3) - \mathbb{E}(X) = \mathbb{E}(X^3).$$

Given that $X \sim \mathcal{N}(0,1)$, we find

$$\begin{aligned} E(X^3) &= \int_{-\infty}^{+\infty} x^3 \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx \\ &= \int_{0}^{+\infty} x^3 \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx + \int_{-\infty}^{0} x^3 \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx \\ &= \int_{0}^{+\infty} x^3 \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx + \left(-\int_{0}^{+\infty} x^3 \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx\right) \\ &= 0. \end{aligned}$$

Combining these results gives $COV(X,Y) = E(X^3) = 0$. As a consequence, the correlation $\rho_{XY} = COV(X,Y)/\sqrt{\text{Var}(X)\text{Var}(Y)} = 0$ as well.
We can also approach the problem numerically:

```
> set.seed(85276)
> x <- rnorm(10^5, mean=0, sd=1)
> y <- x^2
> cov(x, y)
[1] 0.0006595257
> cor(x, y)
[1] 0.0004633695
```

2. We just saw an example of this in part 1. Clearly, the random variable $Y = X^2$ depends on X. In fact, if we know X we also know Y, and if we know Y we know the absolute value of X. Thus, X and Y are not independent, but nevertheless the correlation between X and Y is 0.
3. It is important to note that the solutions we offer here are not the only methods we can use to examine the associations between ordinal and interval variables as well as nominal and interval variables.
   First, we will find a way to examine the association between nominal and interval variables. (Note here that the association is not the correlation. It is meaningless to examine the correlation when one of the variables is a nominal variable.) Here, for illustration, we will examine the association between `Church` and `Age` in our data. Our general strategy is regressing `Age` on the binary variable `Church` and investigating the R^2 (the variance explained by `Church`):

```
> # Fit the linear regression model:
> M1 <- lm(Age ~ Church, data=voting)
>
> # Obtain the R-squared and the association indicator:
> rsquared <- summary(M1)$r.squared
> assocind <- sqrt(rsquared)
> assocind
[1] 0.2869857
```

   To find the association between ordinal and interval variables we will use `Educ` and `Age` in our data set as an example. Before we analyze the association between `Educ` and `Age`, note that the levels of the factor `Educ` are not in the correct order:

```
> levels(voting$Educ)
[1] "2"  "3"   "High" "Low"
```

The correct order is `"Low" "2" "3" "High"`. We can place the education factor levels in the correct order using the following code:

```
> voting$Educ <- factor(voting$Educ, levels(voting$Educ)[c(4,
    1, 2, 3)], ordered=TRUE)
> levels(voting$Educ)
[1] "Low" "2" "3"  "High"
```

Next, we will present two ways to analyze the association between ordinal and interval variables using `Educ` and `Age` as an example. First, we can treat the ordinal variable(s) as nominal variable(s) and then we can use the same method we used above for `Church` and `Age` to find the association:

```
> # Fit the linear regression model:
> M2 <- lm(Age ~ Educ, data=voting)
>
> # Obtain the R-squared and the association indicator:
> rsquared <- summary(M2)$r.squared
> assocind <- sqrt(rsquared)
> assocind
[1] 0.2377672
```

The second approach is to use Spearman's ρ, which is a measure of rank correlation. This method can be seen as treating the interval variable(s) as ordinal variable(s). Here we use the function `rcorr()` from the package `Hmisc` to compute Spearman's ρ:

```
> library(Hmisc)
> rho <- rcorr(voting$Educ, voting$Age, type="spearman")
> rho$r
          x         y
x 1.0000000 -0.1674913
y -0.1674913 1.0000000
```

Please note that since the square root of R^2 is always positive we cannot really tell the direction of the association using the first approach. The second approach is preferable since it gives more information.

6.4 Note that the data generating function is

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \varepsilon.$$

1. Since we are only interested in the correlation between X_2 and Y, the random variable X_1 may be annoying if we want to achieve our goal (because we have no idea how the variation in X_1 will change the relationship between X_2 and Y). Therefore, we will set the coefficients β_0 (the constant will not change the correlation), β_1, and β_3 to zero. Then the data generating function becomes

$$Y = \beta_2 X_2 + \varepsilon.$$

Next, we derive the correlation between X_2 and Y to see which values of β_2 and the variance of the error term $\text{Var}(\varepsilon)$ will help to form a correlation of 0.8:

$$\begin{aligned}\rho(X_2,Y) = \rho(X_2,\beta_2X_2+\varepsilon) &= \frac{COV(X_2,\beta_2X_2+\varepsilon)}{\sqrt{\text{Var}(X_2)}\sqrt{\text{Var}(\beta_2X_2+\varepsilon)}} \\ &= \frac{COV(X_2,\beta_2X_2)+COV(X_2,\varepsilon)}{\sqrt{\text{Var}(X_2)}\sqrt{\text{Var}(\beta_2X_2)+\text{Var}(\varepsilon)}} \\ &= \frac{\beta_2 COV(X_2,X_2)}{\sqrt{\text{Var}(X_2)}\sqrt{\beta_2^2\text{Var}(X_2)+\text{Var}(\varepsilon)}} \\ &= \frac{\beta_2\text{Var}(X_2)}{\sqrt{\text{Var}(X_2)}\sqrt{\beta_2^2\text{Var}(X_2)+\text{Var}(\varepsilon)}} \\ &= \frac{\beta_2\cdot 100}{\sqrt{100}\cdot\sqrt{\beta_2^2\cdot 100+\text{Var}(\varepsilon)}} = \frac{\beta_2\cdot 10}{\sqrt{\beta_2^2\cdot 100+\text{Var}(\varepsilon)}},\end{aligned}$$

where $COV(X_2,\varepsilon)=0$ and $\text{Var}(X_2)=100$ is given in the [R] code in Problem 6.4 (we use `sd=10` for `x2` there). Now, we want to obtain a correlation of $\rho(X_2,Y)=0.8$. Arbitrarily setting $\beta_2=0.8$ and solving the equation

$$0.8 = \frac{0.8\cdot 10}{\sqrt{0.8^2\cdot 100+\text{Var}(\varepsilon)}}$$

for $\text{Var}(\varepsilon)$ results in $\text{Var}(\varepsilon)=36$. We will try to verify this in the following simulation:

```
> # Generate two predictors x1 and x2:
> set.seed(2095)
> n <- 1000 # Number of observations.
> x1 <- runif(n, min=-10, max=10)
> x2 <- rnorm(n, mean=10, sd=10)
>
> # Specify the regression coefficients and the standard
    deviation of the error term:
> beta0 <- 0
> beta1 <- 0
> beta2 <- 0.8
> beta3 <- 0
> err <- 6
>
> # Generate outcome data y:
> y <- beta0 + beta1*x1 + beta2*x2 + beta3*x1*x2 + rnorm(n,
    mean=0, sd=err)
>
> # Compute the correlation between y and x2:
> cor(y, x2)
[1] 0.795739
```

2. To make $\hat{\beta}_2$ close to 5, we only need to set β_2 to 5 and set a small value for the error variance $\text{Var}(\varepsilon)$ (to create some variation). Again, we verify this using simulation:

```
> # Generate two predictors x1 and x2:
> set.seed(2095)
> n <- 1000 # Number of observations.
> x1 <- runif(n, min=-10, max=10)
> x2 <- rnorm(n, mean=10, sd=10)
>
> # Specify the regression coefficients and the standard
    deviation of the error term:
> beta0 <- 0
> beta1 <- 0
> beta2 <- 5
> beta3 <- 0
> err <- 0.1
>
> # Generate outcome data y:
> y <- beta0 + beta1*x1 + beta2*x2 + beta3*x1*x2 + rnorm(n,
    mean=0, sd=err)
>
> # Estimate the regression coefficients:
> M2 <- lm(y ~ x1 + x2 + x1*x2)
> round(coef(M2), 3)
(Intercept)     x1       x2     x1:x2
    -0.001   0.000    5.000    0.000
```

It can be seen that the estimate $\hat{\beta}_2 = 5.000$.

3. Contrary to what we have just done, to make the mean squared error larger than 10000, our general strategy is to set all coefficients β_0, β_1, β_2, and β_3 to 0 and to set the variance of the error term $\text{Var}(\varepsilon)$ to a large value (e.g., 10^6). We verify this strategy in the following simulation:

```
> # Generate two predictors x1 and x2:
> set.seed(2095)
> n <- 1000 # Number of observations.
> x1 <- runif(n, min=-10, max=10)
> x2 <- rnorm(n, mean=10, sd=10)
>
> # Specify the regression coefficients and the standard
    deviation of the error term:
> beta0 <- 0
> beta1 <- 0
> beta2 <- 0
> beta3 <- 0
> err <- sqrt(10^6)
>
> # Generate outcome data y:
> y <- beta0 + beta1*x1 + beta2*x2 + beta3*x1*x2 + rnorm(n,
    mean=0, sd=err)
>
> # Compute the mean squared error:
> M3 <- lm(y ~ x1 + x2 + x1*x2)
```

```
> mse <- mean(M3$residuals^2)
> mse
[1] 1092665
```

Chapter 7
Solutions for Chapter 7

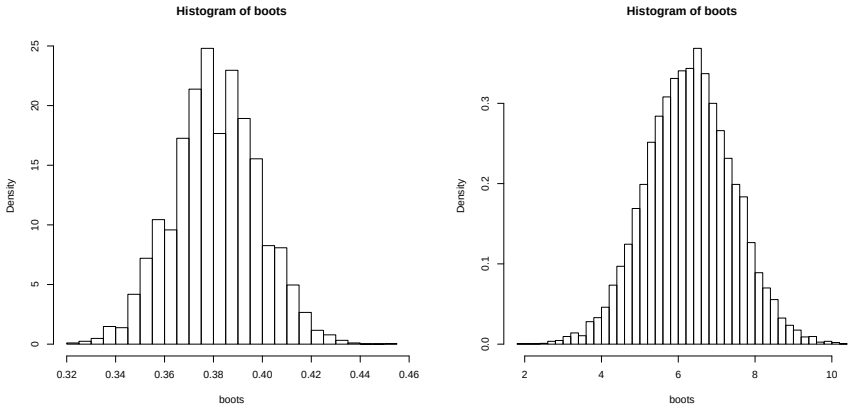
7.1

1. The following [R] code can be used to inspect the frequency distribution of the variable `Choice` and compute a bootstrap distribution of the choice frequency:

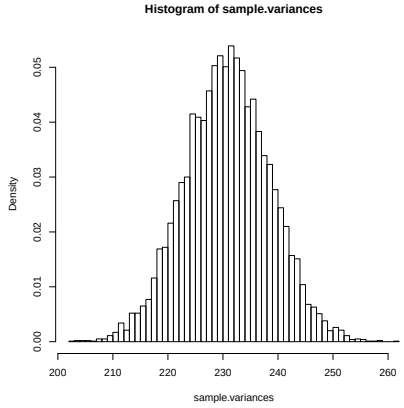
```
> # Import the data:
> voting <- read.csv("voting-demo.csv")
>
> # Inspect the frequency distribution of the variable Choice
    :
> t1 <- table(voting$Choice)
> t2 <- transform(t1, cumulative=cumsum(Freq), relative=prop.
    table(Freq))
> t2
    Var1 Freq cumulative relative
1 CDU/CSU 333    333 0.4440000
2    FDP 131     464 0.1746667
3    SPD 286     750 0.3813333
>
> # Non-parametric bootstrap for the proportion of people who
     vote for the SPD:
> set.seed(42781)
> n <- nrow(voting) # Sample size.
> K <- 10000 # Number of bootstrap replicates.
> boots <- rep(NA, times=K)
> for (k in 1:K) {
+  resamp <- sample(voting$Choice, size=n, replace=TRUE)
+  # Compute the proportion of people who vote for the SPD:
+  boots[k] <- sum(resamp=="SPD")/n
+ }
> hist(boots, breaks=30, freq=FALSE)
```

   From the histogram in Figure 7.1a it can be seen that the distribution of the proportion is approximately bell-shaped.
2. The following [R] code produces a bootstrap distribution of the difference between voters and non-voters:

```
> # Extract the ages of voters and non-voters:
```



(a) (b)



(c)

Fig. 7.1: Plots for Problem 7.1.

```
> Age.vote <- voting$Age[voting$Vote=="Did␣vote"]
> Age.novote <- voting$Age[voting$Vote=="Did␣not␣vote"]
> mean(Age.vote, na.rm=TRUE)
[1] 45.46307
> mean(Age.novote, na.rm=TRUE)
[1] 39.18293
>
> # Calculate the size of each sample:
> n.vote <- length(Age.vote)
> n.novote <- length(Age.novote)
>
> # The bootstrapping procedure:
> set.seed(42781)
```

```
> K <- 10000
> boots <- rep(NA, times=K)
> for (k in 1:K) {
+  resamp.vote <- sample(Age.vote, size=n.vote, replace=TRUE)
+  resamp.novote <- sample(Age.novote, size=n.novote, replace
    =TRUE)
+  mean.vote <- mean(resamp.vote, na.rm=TRUE)
+  mean.novote <- mean(resamp.novote, na.rm=TRUE)
+  boots[k] <- mean.vote - mean.novote
+ }
> hist(boots, breaks=30, freq=FALSE)
```

The histogram is shown in Figure 7.1b. Note that if the voters are on average older than the non-voters, then the difference `mean.vote - mean.novote` is greater than 0. Hence, we can compute the proportion of cases that the voters are on average older than the non-voters as follows:

```
> sum(boots > 0)/K
[1] 1
```

Thus, we can be practically certain that the voters are on average older than the non-voters.

3. In order to solve this problem we need to know the updating formula for the sample variance. This can be found in **?**. An online bootstrap distribution of the variance of the variable `Age` can then be obtained using the following [R] code:

```
> # Extract the variable Age and remove missing values:
> voting.Age <- voting$Age
> voting.Age <- voting.Age[-which(is.na(voting.Age))]
>
> # The online bootstrapping procedure:
> set.seed(42781)
> K <- 10000
>
> # We use starting values of 0 for the means, counts, and
    sums of squares:
> online.means <- rep(0, times=K)
> online.counts <- rep(0, times=K)
> online.sumsofsquares <- rep(0, times=K)
>
> # Function for online update of mean:
> update.mean <- function(current, n, x) {
+  return(current+(x-current)/n)
+ }
>
> # Function for online update of the sum of squares:
> update.sumofsquares <- function(current, xbar, n, d, x) {
+  return(current+d*(x-xbar))
+ }
>
> for (i in 1:length(voting.Age)) {
+  # Sample:
+  update <- sample(c(TRUE, FALSE), size=K, replace=TRUE)
+  # Update:
```

```
+  d <- voting.Age[i] - online.means
+  online.counts[update] <- online.counts[update] + 1
+  online.means[update] <- update.mean(current=online.means[
    update], n=online.counts[update], x=voting.Age[i])
+  online.sumsofsquares[update] <- update.sumofsquares(
    current=online.sumsofsquares[update], xbar=online.means[
    update], n=online.counts[update], d=d[update], x=voting.
    Age[i])
+ }
>
> # Use the online sums of squares and counts to compute the
    sample variances:
> sample.variances <- online.sumsofsquares/(online.counts-1)
> summary(sample.variances)
   Min. 1st Qu. Median Mean 3rd Qu. Max.
  202.8 225.5 230.9 230.8 236.0 261.6
> hist(sample.variances, breaks=50, freq=FALSE)
```

The histogram is shown in Figure 7.1c.

7.2

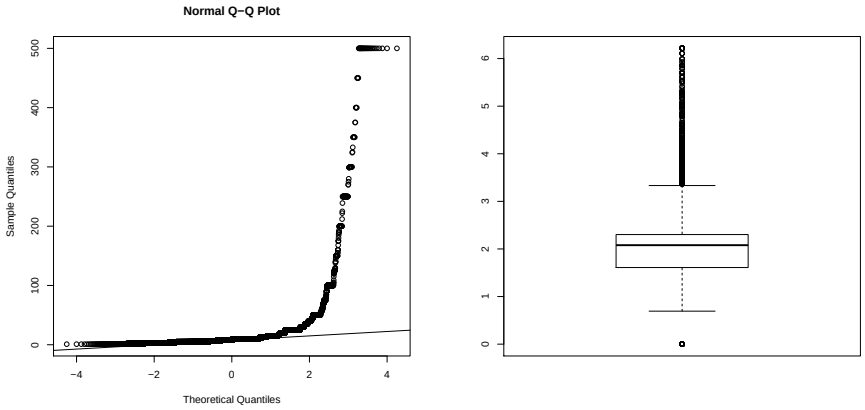
1. Let μ_A denote the mean allowance of all high school children in the first two grades. We are interested in testing the null hypothesis $H_0 : \mu_A = 10$ against the alternative hypothesis $H_1 : \mu_A \neq 10$. We can conduct the test in [R] using the following code:

```
> # Import the data:
> high_school <- read.csv("high-school.csv")
>
> # Remove cases that do not receive an allowance:
> high_school <- high_school[high_school$ALLOWANCE>0, ]
>
> # Conduct the hypothesis test:
> t.test(high_school$ALLOWANCE, mu=10)

      One Sample t-test

data: high_school$ALLOWANCE
t = 17.475, df = 47376, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 10
95 percent confidence interval:
 11.43793 11.80124
sample estimates:
mean of x
 11.61958
```

   The small p-value and the fact that the 95% confidence interval does not contain the value 10 suggest that the null hypothesis is to be rejected.
2. In this case the null and alternative hypothesis are given by $H_0 : |\mu_A - 10| > 2$ and $H_1 : |\mu_A - 10| \leq 2$, respectively, with μ_A the mean allowance of all high-school children in the first two grades. To test the hypotheses we compute a 90% confidence interval for the difference $\mu_A - 10$ and check whether this confidence interval is in between our equivalence bounds of -2 and 2:



(a) (b)

Fig. 7.2: Plots for Problem 7.2.

```
> diff <- mean(high_school$ALLOWANCE) - 10
> n <- length(high_school$ALLOWANCE)
> s2 <- var(high_school$ALLOWANCE)
> SE <- sqrt(s2/n) # Estimated standard error of the mean.
> CI.lower <- diff - SE*qnorm(1-0.05)
> CI.upper <- diff + SE*qnorm(1-0.05)
> -2 < CI.lower & CI.upper < 2
[1] TRUE
```

Since the 90% confidence interval is in between the equivalence bounds, we reject the null hypothesis and conclude that the mean allowance is equivalent to 10 euros.

3. To check if the variable ALLOWANCE is normally distributed we can inspect a q-q plot:

```
> qqnorm(high_school$ALLOWANCE)
> qqline(high_school$ALLOWANCE)
```

The plot is shown in Figure 7.2a. It can be seen that the distribution of ALLOWANCE deviates strongly from normality.

4. After log-transforming the variable ALLOWANCE, we can use the `var.test()` function to check whether the variances of boys and girls are equal. This function tests the null hypothesis of equality of variances $H_0 : \sigma_B^2 = \sigma_G^2$ against the alternative hypothesis $H_1 : \sigma_B^2 \neq \sigma_G^2$ stating that the two variances are not equal, where σ_B^2 and σ_G^2 are the variances of boys and girls, respectively. We can perform the test in [R] as follows:

```
> high_school$logALLOWANCE <- log(high_school$ALLOWANCE)
> logALLOWANCE_boys <- high_school$logALLOWANCE[high_school$
    GENDER=="Boy"]
```

```
> logALLOWANCE_girls <- high_school$logALLOWANCE[high_school$
    GENDER=="Girl"]
> var.test(logALLOWANCE_boys, logALLOWANCE_girls)

      F test to compare two variances

data: logALLOWANCE_boys and logALLOWANCE_girls
F = 1.1145, num df = 23208, denom df = 24167, p-value < 2.2e
    -16
alternative hypothesis: true ratio of variances is not equal
    to 1
95 percent confidence interval:
 1.086476 1.143267
sample estimates:
ratio of variances
       1.114506
```

Based on the small p-value and the fact that the 95% confidence interval does not contain the value 1, we reject the null hypothesis and conclude that the variances are not equal. We can then perform a t-test with unequal variances to test if the means of the log allowances of boys and girls are equal. The null and alternative hypothesis in this t-test are $H_0 : \mu_B = \mu_G$ and $H_1 : \mu_B \neq \mu_G$, where μ_B and μ_G are the means of boys and girls, respectively:

```
> t.test(logALLOWANCE_boys, logALLOWANCE_girls, var.equal=
    FALSE)

      Welch Two Sample t-test

data: logALLOWANCE_boys and logALLOWANCE_girls
t = 13.263, df = 46955, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal
     to 0
95 percent confidence interval:
 0.06687264 0.09006550
sample estimates:
mean of x mean of y
 2.199558 2.121089
```

From the small p-value and the fact that the 95% confidence interval does not contain the value 0, we conclude that the mean log allowance of boys is not equal to the mean log allowance of girls.

5. The following [R] code produces a box plot of the variable `logALLOWANCE` and calculates how many observations are considered an outlier according to Tukey's criterion:

```
> boxplot(high_school$logALLOWANCE)
> q1 <- quantile(high_school$logALLOWANCE, 0.25)
> q3 <- quantile(high_school$logALLOWANCE, 0.75)
> iqr <- q3 - q1
> lower <- q1 - 1.5*iqr
> upper <- q3 + 1.5*iqr
> outlier <- high_school$logALLOWANCE < lower | high_school$
    logALLOWANCE > upper
```

```
> sum(outlier)
[1] 1991
```

The box plot is shown in Figure 7.2b. According to Tukey's criterion there are 1991 outliers.

6. We use Grubbs test to test the null hypothesis H_0 : "there is no outlier in the log allowances" against the alternative hypothesis H_1 : "there is one outlier in the log allowances":

```
> library(outliers)
> grubbs.test(high_school$logALLOWANCE)

        Grubbs test for one outlier

data: high_school$logALLOWANCE
G = 6.29420, U = 0.99916, p-value = 7.257e-06
alternative hypothesis: highest value 6.21460809842219 is an
    outlier
```

The small p-value suggests that the null hypothesis is rejected and we conclude that the highest value 6.215 is an outlier.

7.3

1. We first import the data and remove children that do not work. That is, we only retain children who have a value greater than 0 on the variable `WORK`. Next, we create a new data set that only includes the boys and log transform the variable `WORK` in this new data set. We call this log transformed variable `logWORK`. Following this, we can produce a q-q-plot to examine whether `logWORK` is normally distributed. The following [R] code performs these computations:

```
> # Import the data:
> high_school <- read.csv("high-school.csv")
>
> # Remove children that do not earn money:
> high_school <- high_school[high_school$WORK>0, ]
>
> # Examine normality of log transformed WORK for boys:
> boys <- high_school[high_school$GENDER=="Boy", ]
> boys$logWORK <- log(boys$WORK)
> qqnorm(boys$logWORK)
> qqline(boys$logWORK)
```

The q-q plot in Figure 7.3 does not show strong deviations from normality.

2. Here we use Grubbs test to test the null hypothesis H_0 : "there is no outlier in the log work" against the alternative hypothesis H_1 : "there is one outlier in the log work":

```
> library(outliers)
> girls <- high_school[high_school$GENDER=="Girl", ]
> girls$logWORK <- log(girls$WORK)
> grubbs.test(girls$logWORK)

        Grubbs test for one outlier
```

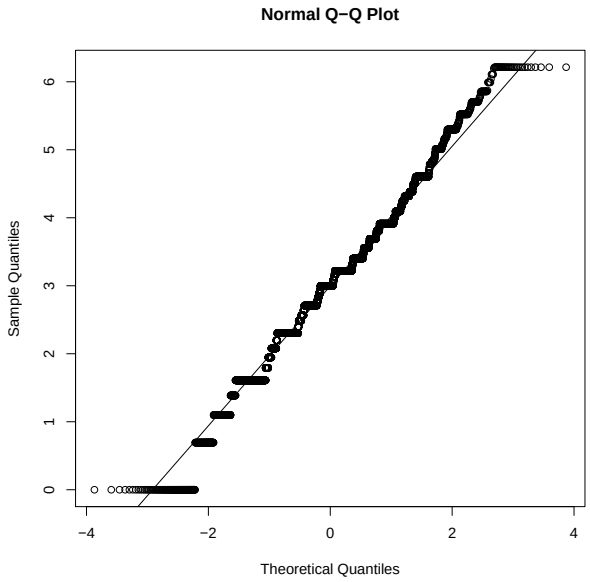


Fig. 7.3: Plot for part 1 of Problem 7.3.

```
data: girls$logWORK
G = 3.62560, U = 0.99833, p-value = 1
alternative hypothesis: highest value 6.21460809842219 is an
    outlier
```

The p-value of 1 indicates that there is not sufficient evidence to reject the null hypothesis.

3. We use a t-test to test the null hypothesis $H_0 : \mu_B = \mu_G$ against the alternative hypothesis $H_1 : \mu_B \neq \mu_G$, with μ_B and μ_G the mean log work of all high school boys and girls in the first two grades, respectively:

```
> t.test(boys$logWORK, girls$logWORK)

      Welch Two Sample t-test

data: boys$logWORK and girls$logWORK
t = 19.89, df = 17014, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal
     to 0
95 percent confidence interval:
 0.2810184 0.3424615
sample estimates:
mean of x mean of y
 2.997431 2.685691
```

Based on the small p-value and the fact that the 95% confidence interval does not contain the value 0, we reject the null hypothesis and conclude that there is a difference in the average log transformed amount of money between boys and girls.

4. In this equivalence test we test the null hypothesis $H_0 : |\mu_B - \mu_G| > 0.1$ against the alternative hypothesis $H_1 : |\mu_B - \mu_G| \leq 0.1$, with μ_B and μ_G the mean log work of all high school boys and girls in the first two grades, respectively:

```
> diff <- mean(boys$logWORK) - mean(girls$logWORK)
> n_boys <- length(boys$logWORK)
> n_girls <- length(girls$logWORK)
> s2_boys <- var(boys$logWORK)
> s2_girls <- var(girls$logWORK)
> SE <- sqrt(s2_boys/n_boys + s2_girls/n_girls) # Estimated
    standard error of the difference between the two means.
> CI.lower <- diff - SE*qnorm(1-0.05)
> CI.upper <- diff + SE*qnorm(1-0.05)
> -0.1 < CI.lower & CI.upper < 0.1
[1] FALSE
```

Since the 90% confidence interval is not in between the equivalence bounds, we cannot reject the null hypothesis.

5. Here we are interested in testing the one-sided null hypothesis $H_0 : \mu_E \leq \mu_N$ against the one-sided alternative hypothesis $H_1 : \mu_E > \mu_N$, with μ_E and μ_N the mean amount of money from work for breakfast eaters and non-breakfast easters, respectively. The sample size is very large, which is why an asymptotic approach can be applied. However, this will almost be the same as the following t-test on the data:

```
> breakfast <- high_school[high_school$BREAKFAST=="Yes", ]
> no_breakfast <- high_school[high_school$BREAKFAST=="No", ]
> t.test(breakfast$WORK, no_breakfast$WORK , alternative="
    greater")

      Welch Two Sample t-test

data: breakfast$WORK and no_breakfast$WORK
t = -9.5009, df = 1125.7, p-value = 1
alternative hypothesis: true difference in means is greater
    than 0
95 percent confidence interval:
 -25.09163   Inf
sample estimates:
mean of x mean of y
 28.72794 50.11402
```

The p-value of 1 indicates that the null hypothesis cannot be rejected.

Chapter 8
Solutions for Chapter 8

8.1

1.
```
> set.seed(78)
> n <- 50
> mu <- 10
> sigma2 <- 2
> y <- rnorm(n, mean=mu, sd=sqrt(sigma2))
```

2.
```
> post.mean.var <- function(y, sigma2, mu0, tau02) {
+  n <- length(y)
+  ybar <- mean(y)
+  tau12 <- (1/tau02 + n/sigma2)^-1
+  mu1 <- (mu0/tau02 + n*ybar/sigma2) * tau12
+  results <- c(mu1, tau12)
+  names(results) <- c("mu1", "tau12")
+  return(results)
+ }
>
> mu0 <- 12
> tau02 <- 0.1
> post <- post.mean.var(y, sigma2, mu0, tau02)
> post
      mu1     tau12
10.42228918 0.02857143
```

3. The following [R] code creates a plot of the prior distribution:

```
> ybar <- mean(y)
> curve(dnorm(x, mean=mu0, sd=sqrt(tau02)),
+     from=min(ybar, mu0)-5*sqrt(post[2]), to=max(ybar, mu0)
    +5*sqrt(post[2]),
+     ylim=c(0, dnorm(post[1], mean=post[1], sd=sqrt(post[2])
    )),
+     xlab=expression(mu), ylab="Density␣/␣Likelihood", cex.
    lab=1.2)
```

The plot is shown in Figure 8.1. The prior distribution is depicted in black (the posterior distribution and the relative likelihood will be added later on). Note

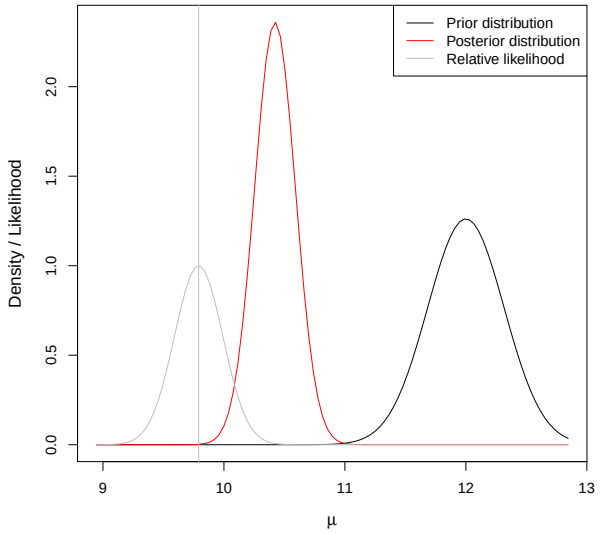


Fig. 8.1: Plot for Problem 8.1.

that the upper limit of the y-axis is the density of the posterior distribution at the posterior mean μ_1 (which is the first element in `post`).

4. The following [R] code adds the posterior distribution in red to the plot in Figure 8.1:

```
> curve(dnorm(x, mean=post[1], sd=sqrt(post[2])), add=TRUE,
    col="red")
> legend("topright", legend=c("Prior␣distribution", "
   Posterior␣distribution"),
+      lty=1, col=c("black", "red"), bg="white")
> abline(v=ybar, col="grey")
```

5. Different choices of μ_0, τ_0^2, and n can be implemented in the [R] code above by assigning different values to the objects `mu0`, `tau02`, and `n`.
6. The `relative.likelihood(y, mu, sigma2)` function below computes the relative likelihood for a given data set $\mathbf{y}$ (`y`) and variance σ^2 (`sigma2`) over a grid of values for the unknown parameter μ (stored in the vector `mu`). To avoid underflow, we first compute the relative log likelihood $\tilde{l}(\mu) = \log \tilde{L}(\mu) = l(\mu) - l(\hat{\mu})$, where $l(\mu)$ is the regular log likelihood and $l(\hat{\mu})$ is the log likelihood evaluated at the maximum likelihood estimate $\hat{\mu} = \bar{y}$. Following this, we compute the relative likelihood as $\tilde{L}(\mu) = \exp\left(\tilde{l}(\mu)\right)$:

```
> relative.likelihood <- function(y, mu, sigma2) { # mu can
    be a vector.
+  ybar <- mean(y)
```

```
+  max.logL <- sum(dnorm(y, mean=ybar, sd=sqrt(sigma2), log=
    TRUE)) # Compute log likelihood at ybar.
+  logL <- rep(NA, length(mu))
+  for (i in 1:length(mu)) {
+    logL[i] <- sum(dnorm(y, mean=mu[i], sd=sqrt(sigma2), log=
    TRUE)) # Compute log likelihood for values in mu.
+  }
+  relative.logL <- logL - max.logL # Compute relative log
    likelihood.
+  relative.L <- exp(relative.logL) # Compute relative
    likelihood.
+  return(relative.L)
+ }
```

We can then use the `curve()` function to add the relative likelihood in grey to the existing plot of the prior and posterior distribution in Figure 8.1:

```
> curve(relative.likelihood(y, x, sigma2), add=TRUE, col="
    grey")
> legend("topright", legend=c("Prior␣distribution", "
    Posterior␣distribution",
+      "Relative␣likelihood"), lty=1, col=c("black", "red", "
    grey"),
+      bg="white")
```

8.2

1. ```
   > simData <- function(n, theta) {
   +  return(sum(rbinom(n, size=1, prob=theta)))
   + }
   ```

2. ```
   > updateBelief <- function(params=c(1, 1), s, n) {
   +  params[1] <- params[1] + s # Update alpha.
   +  params[2] <- params[2] + (n-s) # Update beta.
   +  return(params)
   + }
   ```

3. ```
   > computeLoss <- function(d, theta) {
   +  return((d - theta)^2)
   + }
   ```

4. ```
   > expectedValueBeta <- function(params) {
   +  return(params[1] / sum(params))
   + }
   ```

5. ```
   > params <- c(1, 1) # Specify uniform prior with alpha = beta
       = 1.
   > theta <- 0.2 # True theta is 0.2.
   > n <- 100 # Number of observations.
   > m <- 1000 # Number of simulation runs.
   > EvVector <- rep(NA, times=m) # Results vector for the
     expected value of the posterior.
   >
   ```
   ```

```
> for (i in 1:m) {
+  # Simulate data:
+  s <- simData(n, theta)
+  # Compute expected value and store:
+  posterior.params <- updateBelief(params, s, n)
+  EvVector[i] <- expectedValueBeta(posterior.params)
+ }
```

6.
```
> meanEV <- mean(EvVector)
> curve(computeLoss(x, theta), from=0, to=1, xlab="d", ylab="
    Loss")
> abline(v=meanEV, col="red")
> meanEV
[1] 0.2049118
```

From the plot it can be seen that the posterior mean almost minimizes the squared error loss. "Almost", as the prior currently forces the posterior mean upwards: by choosing a larger n, or by setting the prior to `c(.0001,.0001)` you will see that the expected value over multiple samples of the expected value of the posterior gets closer and closer to .2.

8.3

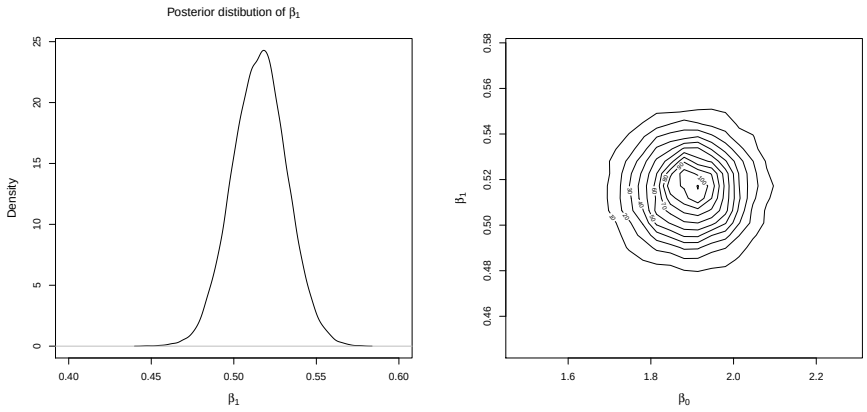
1. We can produce a posterior density plot of β_1 using the following [R] code:

```
> # Generate data and fit Bayesian regression model:
> library(MCMCpack)
> set.seed(13412348)
> n <- 100
> x <- runif(n, min=-10, max=10)
> y <- rnorm(n, mean=2+0.5*x, sd=1)
> bayes <- MCMCregress(y ~ x)
>
> # Create posterior density plot of beta_1:
> plot(density(bayes[, 2]), xlim=c(0.4, 0.6), xlab=expression
    (beta[1]),
+    main=expression(paste("Posterior␣distibution␣of␣", beta
    [1])), cex.lab=1.2)
```

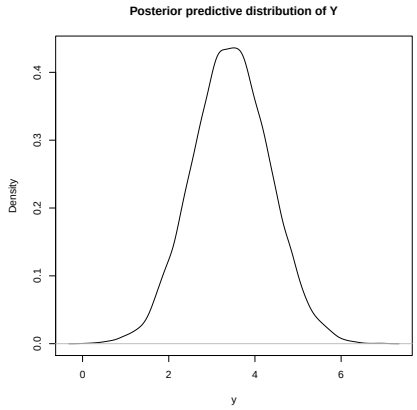
The plot is shown in Figure 8.2a. It can be seen that the posterior density at $\beta_1 = 0.6$ is practically 0. Hence, it is unlikely that $\beta_1 = 0.6$ in the population.

2. In order to create a contour plot of the joint posterior density of β_0 and β_1 we first need to estimate this density. This can be achieved with the function `kde2d()` from the package `MASS`, which performs two-dimensional kernel density estimation. Once we have obtained an estimate of the posterior density, we can produce the contour plot using the `contour()` function:

```
> library(MASS)
> z <- kde2d(bayes[, 1], bayes[, 2])
> contour(z, xlab=expression(beta[0]), ylab=expression(beta
    [1]), cex.lab=1.2)
> cor(bayes[, 1], bayes[, 2])
[1] -0.007613695
```



(a) (b)



(c)

Fig. 8.2: Plots for Problem 8.3.

The contour plot is given in Figure 8.2b. It shows no noteworthy correlation between β_0 and β_1, which is confirmed by a sample correlation close to 0.

3. Here we need to compute the posterior predictive distribution of Y for $x = 3$. We can approximate the posterior predictive distribution using the draws from the joint posterior distribution of $(\beta_0, \beta_1, \sigma^2)$. Let $\left(\beta_0^{(s)}, \beta_1^{(s)}, \sigma^{2(s)}\right)$ denote the sth draw from the posterior, for $s = 1, \ldots, S$, where S is the total number of draws. We can approximate the posterior predictive distribution of Y by drawing $y^{(s)} \sim \mathcal{N}\left(\beta_0^{(s)} + \beta_1^{(s)} \cdot x, \sigma^{2(s)}\right)$, for $s = 1, \ldots, S$. Then $y^{(1)}, \ldots, y^{(S)}$ are an i.i.d. sample from the posterior predictive distribution of Y. We can implement this procedure in [R] as follows:

```
> x <- 3
```

```
> beta0 <- bayes[, 1]
> beta1 <- bayes[, 2]
> sigma2 <- bayes[, 3]
> S <- nrow(bayes)
> y <- rnorm(S, mean=beta0+beta1*x, sd=sqrt(sigma2))
```

With the draws in `y` at hand, we can approximate the posterior predictive distribution using a density plot:

```
> plot(density(y), main="Posterior␣predictive␣distribution␣of
    ␣Y", xlab="y")
```

The plot is shown in Figure 8.2c. It can be seen that the new observation Y will most likely lie around 3.5. To obtain a point prediction for the new observation, we can compute, for example, the mean of the posterior predictive distribution:

```
> mean(y)
[1] 3.443485
```

Finally, we can compute the standard deviation of the draws in `y` and a 95% interval to quantify our uncertainty about the new observation Y:

```
> sd(y)
[1] 0.8956364
> quantile(y, probs=c(0.025, 0.975))
   2.5%  97.5%
1.705621 5.190702
```

Thus, for $x = 3$ the new observation Y will lie between 1.706 and 5.191 with probability 0.95.

References