Appendix E

Answers

Content *Page*

Chapter 1: Descriptive statistics

1.1 a) Mean: 6.6
Mode: 8
Median: 7
Range: 8
Lower quartile: 4
Upper quartile: 9
Interquartile range: 5

1.1 b) Mean: 5.91
Mode: 7
Median: 6.5
Range: 9
Lower quartile: 3.5
Upper quartile: 8.5
Interquartile range: 6

1.1 c) Assignment 1.1b was probably harder to do as you had to take the middle of two numbers to find the median, and the lower and upper quartiles.

1.1 d) These data sets are negatively skewed.

Explanation: The mean is lower than the median and mode, and so more values are concentrated on the right side (tail) of the distribution graph while the left tail of the distribution graph is longer.

1.2 a) The `View()` command opens a window in which you can inspect the data.

```
dataset1 <- c(2, 7, 4, 5, 8, 10, 10, 7, 9, 2, 8, 8, 9, 4, 6)
View(dataset1)
```

1.2 b) Mean: 6.6
Mode: 8
Median: 7
Range: 8

```
mean(dataset1) # Mean: 6.6
table(dataset1) # Mode: 8 is the most occurring number (3 times)
median(dataset1) # Median: 7
range(dataset1) # Range: 2 to 10 = 8
```

1.2 c) The `quantile()` command returns the minimum (0%), lower quartile (25%), median (50%), upper quartile (75%), and maximum (100%).

```
quantile(dataset1, type = 6)
```

1.2 d) Mean: 5.91
Mode: 7
Median: 6.5
Range: 9
Lower quartile: 3.5
Upper quartile: 8.5

```
dataset2 <- c(7, 7, 6, 5, 2, 1, 3, 7, 5, 9, 9, 10)

mean(dataset2) # Mean: 5.91
table(dataset2) # Mode: 7 is the most occurring number (3 times)
median(dataset2) # Median: 6.5
range(dataset2) # Range: 1 to 10 = 9
```

1.3 a) The code opens a new window in which you can select the `.csv` file that you want to read into your R session. Using the `colnames()` function, you can see that the `bloodPressure.csv` file contains 6 columns named `Number` , `Age` , `BloodPressure` , `Cholestrol` , `Gender` , `Description` .

1.3 b) You can use a relative path to the file on your computer by providing it directly to the `read.csv()` function (in quotes `'bloodPressure.csv'`). Remember to set your working directory correctly, since R will look inside the working directory folder when it receives such a path. You can also specify a full path (like `'C://path/to/file/bloodPressure.csv'`). With full file paths, R will know exactly where to look, and the location of your working directory does not matter.

```
dataset3 <- read.csv('bloodPressure.csv')
```

1.3 c) The mean age of the respondents ($n = 60$) in the data set is 45.15 years. The minimum age of the respondents is 17, and the maximum age is 69. The age of the respondents spans 52 years. The most occurring age is 39. Twenty-five percent of the respondents is aged below 34.5, fifty percent is aged below 46, and 75 percent is aged below 58.5.

```
mean(dataset3$Age) # Mean: 45.15
table(dataset3$Age) # Mode: 39 is the most occurring number (4 times)
median(dataset3$Age) # Median: 46
range(dataset3$Age) # Range: 17 to 69 = 52

quantile(dataset3$Age, type = 6)
# Minimum: 17
# Lower quartile: 34.5
# Median: 46
# Upper quartile: 58.5
# Maximum: 69
```

1.3 d) Mode: 39

```
getMode <- function(x){
   uniqx <- unique(x)
   uniqx[which.max(tabulate(match(x, uniqx)))]
}

getMode(dataset3$Age) # Mode: 39
```

1.3 e) Mean: 130.62

Mode: 129

Median: 131.5

Range: 121

Lower quartile: 118.75

Upper quartile: 145.75

```
mean(dataset3$BloodPressure) # Mean: 130.62
getMode(dataset3$BloodPressure) # Mode: 129
median(dataset3$BloodPressure) # Median: 131.5
range(dataset3$BloodPressure) # Range: 45 to 166

quantile(dataset3$BloodPressure, type = 6)
# Minimum: 45
# Lower quartile: 118.75
# Median: 131.5
# Upper quartile: 145.75
# Maximum: 166
```

1.3 f) These data sets are not skewed.

Explanation: The mean is lower than the median but not than the mode, and so you cannot conclude that the distribution is skewed in any direction.

1.3 g) Variance: 1.059
Standard deviation: 1.029

```
var(dataset3$Cholestrol) # Variance: 1.059
sd(dataset3$Cholestrol) # Standard deviation: 1.029
```

1.3 h)

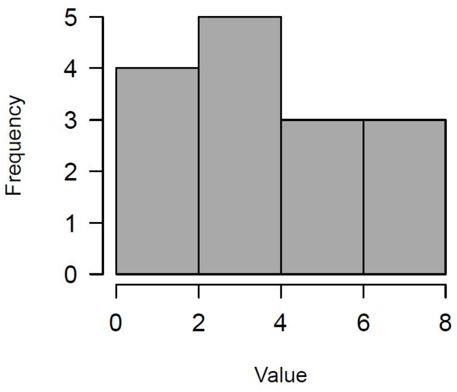
```
# The standard deviation is the square root of the variance
sd(dataset3$Cholestrol) == sqrt(var(dataset3$Cholestrol))
```

Chapter 2: Creating graphs from data

2.1 a)

0 to 2	2 to 4	4 to 6	6 to 8
Frequency: 4	Frequency: 5	Frequency: 3	Frequency: 3

2.1 b)



2.2 a) In the plot window you can see that a histogram for the values in the variable `dataset4` has been drawn by R.

```
dataset4 <- c(1.5, 5.5, 1.7, 7.2, 1.2, 7.9, 1.4, 3.6,
3.1, 3.8, 5.9, 3.6, 5.1, 3.2, 7.1)
hist(dataset4)
```

2.2 b) The histogram drawn by R has four bars for the ranges 1-2, 3-4, 5-6, and 7-8. The histogram from assignment 2.1b has four bars for the ranges 0-2, 2-4, 4-6, and 6-8. The difference in these two histograms is that they have a different number of bars for different ranges.

2.2 c)

```
# The breaks argument determines the number of bars in the histogram
# see ?hist for more help on the hist() function
hist(dataset4, breaks = 4)
```

2.2 d)

```
hist(dataset4, breaks = 4, col = 'blue', # col sets the bar color
xlab = 'x-axis', # xlab sets the x-axis name
ylab = 'Frequency', # ylab sets the y-axis name
main = 'My histogram') # main sets the title
```

2.2 e) The distribution is positively skewed.

Explanation: The mean is higher than the median, and so more values are concentrated on the left side (tail) of the distribution graph while the right tail of the distribution graph is longer.

```
mean(dataset4) # Mean: 4.12
median(dataset4) # Median: 3.6
```

2.3 a)

```
data(swiss) # Import the swiss data

education <- swiss$Education
agriculture <- swiss$Agriculture
```

2.3 b)

```
plot(x = education, y = agriculture,
     xlab = 'Percentage of education beyond primary school',
     ylab = 'Percentage of males involved in agriculture')
```

2.3 c) Looking at the scatter plot, there seems to be a tendency for provinces that have a high percentage of education beyond primary school to also have a low percentage of males involved in agriculture.

2.3 d)

```
plot(x = education, y = agriculture,
     xlab = 'Percentage of education beyond primary school',
     ylab = 'Percentage of males involved in agriculture',
     col = 'blue',
     main = 'Provinces in Switzerland',
     las = 1, # las sets the rotation of the axis labels
     bty = 'n') # bty = 'n'removes the outer border lines
```

2.4 a)

```
data(EuStockMarkets)
stockData <- data.frame(EuStockMarkets)

plot(stockData$DAX,
     type = 'l', # type = 'l'creates lines instead of dots
     xlab = 'Time',
     ylab = 'Price')
```

2.4 b)

```
# The lines() function adds a line to an existing plot
lines(stockData$SMI, col = 'red') # Add a line for the SMI stock
lines(stockData$CAC, col = 'blue') # Add a line for the CAC stock
lines(stockData$FTSE, col = 'green') # Add a line for the FTSE stock
```

2.5 a)

```
Minimum:  1.2                       Lower quartile:  35.3
Median:  54.2                       Upper quartile:  67.8
Maximum:  89.7
```

```
quantile(agriculture, type = 6)
# Minimum: 1.2
# Lower quartile: 35.3
# Median: 54.1
# Upper quartile: 67.8
# Maximum: 89.7
```

2.5 b)

```
boxplot(agriculture)
```

2.5 c) The first line creates a vector called `educationLevel` that contains 47 times `'2.Medium'` . The second line changes the `'2.Medium'` to `'1.Low'` for the provinces that have a percentage of education beyond primary school lower than 6. The third line changes the `'2.Medium'` to `'3.High'` for the provinces that have a percentage higher than 12. The resulting table shows how many provinces had a percentage lower than 6, between 6 and 12, and higher than 12.

```
educationLevel <- rep('2.Medium', 47)
educationLevel[education <= 6] = '1.Low'
educationLevel[education >= 12] = '3.High'
table(educationLevel)
```

2.5 d) The code creates a box plot of the agriculture variable for each education level `'1.Low'` , `'2.Medium'` , and `'3.High'` .

```
boxplot(agriculture ~ educationLevel)
```

Chapter 3: Confidence intervals and hypothesis testing

3.1 a)	$N = 4513$	$s = 25$
	$n = 100$	$\sigma =$ unknown
	$\bar{x} = 145$	$\mu =$ unknown

3.1 b) $\mu =$ 145 seconds

Explanation: The best estimate for the population mean μ is the sample mean $\bar{x}$.

3.1 c) $SE_\mu = \frac{s}{\sqrt{n}} = \frac{25}{\sqrt{100}} = 2.5$

3.1 d) z-value: 2.576

Explanation: In table 2 of the formula sheet, the cumulative probability in that lies the closest to 0.995 (split the risk over two sides) is 0.9949. That value can be found at a z-value of 2.576.

Lower bound: $\bar{x} - z_{0.995} \times SE_\mu = 145 - 2.567 \times 2.5 = 138.56$
Upper bound: $\bar{x} + z_{0.995} \times SE_\mu = 145 - 2.567 \times 2.5 = 151.44$

3.1 e) $\mu_0 = 150$ (the to be tested limit of 150 seconds for the actual population mean call duration).

$H_0 : \mu \geq \mu_0$ $H_0 : \mu < \mu_0$

3.1 f) Upper bound: $\bar{x} + z_{0.99} \times SE_\mu = 145 - 1.645 \times 2.5 = 149.11$

3.1 g) The upper bound of the confidence interval for μ is lower than μ_0. H_0 is rejected with 99% confidence. μ is shown to be significantly lower than 150 seconds. There is a risk of 5% for a type-I error.

3.2 a)

i	x_i	$(x_i - \bar{x})$	$(x_i - \bar{x})^2$
1	3.03	-0.0863	0.0074
2	3.45	0.3338	0.1114
3	3.94	0.8238	0.6786
4	2.34	-0.7763	0.6026
5	3.34	0.2238	0.0501
6	2.53	-0.5863	0.3437
7	2.88	-0.2363	0.0558
8	3.42	0.3038	0.0923

$\sum x_i = 24.93$ $\sum(x_i - \bar{x})^2 = 1.9418$

$\bar{x} = 3.116$ $s^2 = 0.277$

3.2 b) Hartley's F: $\frac{s_m^2 in}{s^{max}} = \frac{1.113}{0.227} = 4.018$

3.2 c) $H_0 : \sigma_1^2 = \sigma_2^2 = \sigma_3^2$ $H_0 : \sigma_1^2 \neq \sigma_2^2 \neq \sigma_3^2$

3.2 d) Hartley's F_{max}: 6.94

3.2 e) The value $F = 4.018$ is lower than critical value $F_{max} = 6.94$. H_0 is not rejected. There is no indication the variance for these months is not homogeneous. There is a risk of a type-II error.

3.3 a)

```
# Be sure to set your working directory when providing a relative path
dataset5 <- read.csv('populations.csv')
```

3.3 b) The code creates four random samples of size 90 from the columns `P1` - `P4` in the data frame called `dataset5` . The seed makes sure that you can recreate the same random samples again, so that if you close R and continue tomorrow we get the same samples.

```
set.seed(54321) # You can replace 54321 with your own seed number

sample1 <- sample(dataset5$P1, size = 90)
sample2 <- sample(dataset5$P2, size = 90)
sample3 <- sample(dataset5$P3, size = 90)
sample4 <- sample(dataset5$P4, size = 90)
```

3.3 c)
```
Standard error sample 1:  29.24
Standard error sample 2:  7.64
Standard error sample 3:  30.61
Standard error sample 4:  18.09
```

```r
# Means
x1 <- mean(sample1) # Mean: 456.78
x2 <- mean(sample2) # Mean: 511.02
x3 <- mean(sample3) # Mean: 790.32
x4 <- mean(sample4) # Mean: 533.37

# Standard deviations
sd1 <- sd(sample1) # Standard deviation: 277.38
sd2 <- sd(sample2) # Standard deviation: 72.43
sd3 <- sd(sample3) # Standard deviation: 290.46
sd4 <- sd(sample4) # Standard deviation: 171.58

# Standard errors se = sd / sqrt(n)
se1 <- sd1 / sqrt(length(sample1)) # Standard error: 29.24
se2 <- sd2 / sqrt(length(sample2)) # Standard error: 7.64
se3 <- sd3 / sqrt(length(sample3)) # Standard error: 30.61
se4 <- sd4 / sqrt(length(sample4)) # Standard error: 18.09
```

3.3 d) The value 1.644854 comes from the standard normal distribution with $\mu = 0$ and $\sigma = 1$. This is the z-value for a 95% one-sided confidence interval (or a 90% two-sided confidence interval).

3.3 e)
```r
# Gives the left-tailed probability (z-value) for 95% confidence
qnorm(p = 0.95) # 1.645
```

3.3 f) In a 95% confidence interval there is 2.5% of the risk at the lower bound and 2.5% of the risk at the upper bound. You can therefore use the 97.5% quantile of the standard normal distribution to get the z-value for a two-sided 95% confidence interval and use it for both upper and lower bound.

```r
z <- qnorm(p = 0.975)
```

3.3 g)

```
# Lower bounds
lb1 <- x1 - z * se1 # Lower bound: 399.48
lb2 <- x2 - z * se2 # Lower bound: 496.06
lb3 <- x3 - z * se3 # Lower bound: 730.31
lb4 <- x4 - z * se4 # Lower bound: 497.92

# Upper bounds
ub1 <- x1 + z * se1 # Upper bound: 514.10
ub2 <- x2 + z * se2 # Upper bound: 525.99
ub3 <- x3 + z * se3 # Upper bound: 850.33
ub4 <- x4 + z * se4 # Upper bound: 568.81
```

3.3 h)

	sample1	sample2	sample3	sample4
ub	514.10	525.99	850.33	568.81
x	456.78	511.01	790.32	533.37
lb	399.48	496.06	730.31	497.92

	P1	P2	P3	P4
mu	495.54	500.08	748.96	556.43
mu in interval?	YES	YES	YES	YES

```
# Population means
mu1 <- mean(dataset5$P1) # Mean: 495.54
mu2 <- mean(dataset5$P2) # Mean: 500.08
mu3 <- mean(dataset5$P3) # Mean: 748.96
mu4 <- mean(dataset5$P4) # Mean: 556.43
```

3.3 i) Yes, in this case all population means are inside the intervals.

3.3 j) The population mean will fall inside the interval 95 out of a 100 times (95% confidence). This means that about 1 in 20 confidence intervals will not have the true population mean between their lower and upper bound.

3.4 a)

```
# install.packages('car')
library(car)

# This create a 2x2 layout
layout(matrix(c(1, 2, 3, 4), byrow = TRUE, nrow = 2))

hist(sample1, col = 'gray')
hist(sample2, col = 'gray')
hist(sample3, col = 'gray')
hist(sample4, col = 'gray')

# This resets the layout to the default (1 plot)
layout(1)
```

3.4 b) `Sample 2 looks like it might come from a normal distribution.`

3.4 c)

```
# This create a 2x2 layout
layout(matrix(c(1, 2, 3, 4), byrow = TRUE, nrow = 2))

qqPlot(sample1, distribution = 'norm')
qqPlot(sample2, distribution = 'norm')
qqPlot(sample3, distribution = 'norm')
qqPlot(sample4, distribution = 'norm')

# This resets the layout to the default (1 plot)
layout(1)
```

3.4 d) The **sample1** histogram looks normal in the middle, but has too many low and too many high values. This can be seen in the qqplot by the dots on the left of the diagonal at the bottom and the dots on the right at the top.

The **sample2** histogram looks normal so the dots in the qqplot are almost everywhere on the diagonal. Only in the right part of the middle the histogram frequency is a bit too low, this is reflected in the dots below the diagonal around zero in the qqplot.

The **sample3** histogram is too high on the sides (or too low in the middle) to be normal. This can be seen in the qqplot by the dots on the left of the diagonal at the bottom and the dots on the right at the top.

sample4 is negatively skewed. This can be seen in the qqplot from the arch shape; the left of the histogram is too low causing the dots on the right of the diagonal and the right of the histogram is too high also causing dots to the right of the diagonal.

3.4 e) H_0: The sample is normally distributed
H_1: The sample is not normally distributed

3.4 f) Samples 1, 3, and 4 are not normally distributed, since the p-value is lower than 1% (for 99% confidence). Sample 2 is normally distributed, since the p-value is higher than 1%.

```
shapiro.test(sample1) # p-value: 0.0022
shapiro.test(sample2) # p-value: 0.949
shapiro.test(sample3) # p-value: 0.0068
shapiro.test(sample4) # p-value: 0.0001
```

3.4 g) When the sample is not normally distributed, the sample can be used to estimate the population mean.

Explanation: This method to estimate the population mean assumes the distribution of sample means to be normally distributed. It does not assume a normally distributed sample. The Central Limit Theorem states that any sample large enough ($n \geq 30$) will have a normal distribution of sample means, so you can use this method here ($n = 90$) without problems.

3.5 a)

```
library(car)
data(iris)

plot(x = iris$Species, y = iris$Sepal.Length,
   col = 'grey', main = 'Sepal Length')

plot(x = iris$Species, y = iris$Sepal.Width,
   col = 'grey', main = 'Sepal Width')
```

3.5 b) Looking at the width of the range and quartile ranges: For sepal length the variance for setosa looks much smaller than for the other two species, for sepal width all variances look similar.

3.5 c) $H_0 : \sigma_1^2 = \sigma_2^2 = \sigma_3^2$ $H_0 : \sigma_1^2 \neq \sigma_2^2 \neq \sigma_3^2$

3.5 d) The p-value for the sepal length is lower than 10%. This implies that H_0 is rejected. This means that the variance of the sepal length over the species is not homogeneous. There is a 5% change of a type-I error.

The p-value for the sepal with is not lower than 10%. This implies that H_0 is not rejected. This means that there is no indication that the variance of the sepal width over the species is not homogeneous. There is a risk of a type-II error.

```
# Levene’s Test for Homogeneity of Variance
leveneTest(y = iris$Sepal.Length,
group = iris$Species) # p-value: 0.002259

leveneTest(y = iris$Sepal.Width,
group = iris$Species) # p-value: 0.5555
```

Chapter 4: Correlation and regression

4.1 a) Two variables can either be positively related, not related, or negatively related. The most logical relationship is that the distance a customer lives from the store is negatively related to how many times they visit the store.

4.1 b)

i	x_i	y_i	$(x_i - \bar{x})$	$(y_i - \bar{y})$	$(x_i - \bar{x})(y_i - \bar{y})$
1	4.87	2.90	1.868	-1.09	-2.053
2	3.04	4.50	0.038	0.501	0.0194
3	1.65	4.94	-1.351	0.941	-1.272
4	2.88	3.28	-0.121	-0.718	0.087
5	2.31	4.73	0.691	0.731	-0.505
6	3.96	2.64	0.958	-1.358	-1.303
7	2.70	3.70	-0.301	-0.298	0.089
8	2.60	5.30	-0.401	1.301	-0.522

$\bar{x} = 3.001$ $\sum(x_i - \bar{x})(y_i - \bar{y}) = -5.458$

$\bar{y} = 3.998$ $n - 1 = 7$

$s_{xy} = -0.779$

4.1 c) The covariance is negative. A negative covariance indicates that that as one variable deviates from the mean, the other variable deviates in the other direction. This means that, when a customer's distance from the store in kilometers is higher than the mean, their average visits per week will likely be lower than the mean.

4.1 d) The disadvantage of using the covariance as a measure for the strength of this relationship is that it depends on the measurement unit (kilometers vs. meters) that the co-worker asks the questions in. If the co-worker would have asked the question in meters the covariance would have increased by a 1000 times, namely -779.84.

4.1 e) $s_x = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} = \sqrt{\frac{6.977}{7}} = 0.998$ $s_y = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n-1}} = \sqrt{\frac{7}{7}} = 1$

4.1 f) $r_{xy} = \frac{s_{xy}}{s_x \times s_y} = \frac{-0.779}{0.998 \times 1} = -0.779$

4.1 g) The coefficient is -0.779, which represents is a relatively strong negative relationship.

4.2 a) $H_0: \quad \rho_{xy} \geq 0$ $\qquad H_1: \quad \rho_{xy} < 0$

4.2 b) $N =$ unknown $\qquad r_{xy} = -0.779$
$n = 8$ $\qquad \rho_{xy} =$ unknown

4.2 c) $z_{0.05} = -1.645$

4.2 d) $z_r = \frac{1}{2} \times log_e(\frac{1+r}{1-r}) = \frac{1}{2} \times log_e(\frac{1-0.779}{1+0.779}) = -1.043$

$SE_r = \frac{1}{\sqrt{n-3}} = \frac{1}{\sqrt{8-3}} = 0.447$

$z_{xy} = \frac{z_r}{SE_r} = \frac{-1.043}{0.447} = -2.33$

4.2 e) The observed z-value is more extreme (lower) than the critical z-value. H_0 is rejected with 95% confidence. You can be 95% confident that ρ_{xy} is negative in the population. There is a 5% change of a type-I error.

4.3 a)

```
# Be sure to set your working directory when providing a relative path
dataset6 <- read.csv('localSupermarket.csv')
```

4.3 b) Covariance: -0.30
Correlation: -0.30

```
cov(dataset6$Distance, dataset6$AvgVisits) # Covariance -0.3000603
cor(dataset6$Distance, dataset6$AvgVisits) # Correlation: -0.3000382
```

4.3 c)

```
cor.test(dataset6$Distance, dataset6$AvgVisits, alternative = 'less')
# Correlation: -0.30
# t-value: -9.936
# p-value: < 2.2e-16
```

4.3 d) The black line represents the normal distribution. The red line represents the t-distribution. The difference between the two distributions, in terms of their shape, is that the t-distribution has slightly thicker tails. When you increase the degrees of freedom of the t-distribution, it will start to look more like the normal distribution.

```
curve(dnorm(x, mean = 0, sd = 1), from = -3, to = 3, ylab = 'Density')
curve(dt(x, df = 3), from = -3, to = 3, add = TRUE, col = 'red')
```

4.3 e) $df = 998$ $t_{xy} = -9.936$

```
n <- nrow(dataset6)
dft <- n - 2 # 998

r <- cor(dataset6$Distance, dataset6$AvgVisits)
tscore <- r * sqrt(n - 2) / sqrt(1 - r^2)
# t-score: -9.936 so you can confirm the value in 4.3c
```

4.3 f) You can find the t-value in the bottom line of the output in the console.

4.3 g) The p-value is < 2.2e-16, which is lower than the significance level of 5%. This means that H_0 can be rejected with 95% confidence.

4.4 a)

```
# Be sure to set your working directory when providing a relative path
dataset7 <- read.csv('nationalSupermarket.csv')
```

4.4 b)

```
plot(x = dataset7$Price,
     y = dataset7$AvgWasted,
     main = 'Scatter plot of Price vs. AvgWasted',
     ylab = 'Average number of cartons wasted',
     xlab = 'Price of a carton of milk',
     las = 1,
     col = 'orange',
     pch = 19,
     bty = 'n')
```

4.4 c) AvgWasted $= \beta_0 + \beta_1 \times$ Price

4.4 d)

```
lmfit <- lm(formula = AvgWasted ~ Price, data = dataset7)
```

4.4 e) AvgWasted $= 0.236 + 2.995 \times$ Price

```
summary(lmfit)
# b0 = 0.236
# b1 = 2.995
# R-squared: 0.64
```

4.4 f)

```r
abline(lmfit)
```

4.4 g) $R^2 = 0.64$

Interpretation: The multiple R^2 is 0.64, meaning that 64% of the variation in the number of milk cartons that are thrown away each day can be explained by the price of the milk cartons.

4.4 h) H_0: $\beta_1 \leq 0$ H_1: $\beta_1 > 0$

4.4 i) The p-value for the regression coefficient is < 2e-16, which is lower than the significance level of 5%. H_0 can be rejected with 95% confidence. You can be 95% sure that β_1 is positive in the population. The price contributes significantly to the average number of milk cartons thrown away. There is a 5% risk of a type-I error.

4.5 a)

```r
newdata <- data.frame(Price = 0.70)
```

4.5 b) Predicted value: 2.33

```r
predict(object = lmfit,
       newdata = newdata) # Prediction: 2.33
```

4.5 c) Predicted value: $0.236 + 2.995 \times 0.70 = 2.33$

4.5 d)

```r
predict(object = lmfit, newdata = newdata,
       interval = 'prediction', level = 0.90)
# Lower bound: 0.734
# Upper bound: 3.931
```

4.5 e) The supermarket will throw away fewer cartons of milk.

Explanation: The current number of milk cartons thrown away (4) lies outside the bounds of the 90% confidence interval for the prediction.

Chapter 5: Comparing one or two means

5.1 a) $n = 49$
$\bar{x} = 0.918$
$s = 0.071$
$\mu_0 = 0.9$

5.1 b) $H_0\colon \quad \mu_0 \leq 0.9$ $\qquad H_1\colon \quad \mu_0 > 0.9$

5.1 c) Lower bound: $\bar{x} - z_\alpha \times SE_\mu = 0.918 - 1.645 \times \frac{0.071}{\sqrt{49}} = 0.901$

5.1 d) The lower bound of the confidence interval for μ is higher than μ_0. H_0 is rejected with 95% confidence. The PFAS level in town is significantly higher than 0.9 microgram/kg dry soil. There is a risk of 5% for a type-I error.

5.1 e) z-score: $\frac{\bar{x}-\mu}{s/\sqrt{n}} = \frac{0.918-0.9}{0.071/\sqrt{49}} = 1.775$

5.1 f) The critical z-value for 95% confidence is 1.645. The p-value (the probability that H_0 is true) is 0.05. For any z-value higher than 1.645 the p-value is lower than 0.05. Since for this test the (absolute of the) calculated z-score = 1.775 and this is higher than 1.645, this means that the p-value is lower than 0.05.

Note: For right sided tests the z-score is negative, but since the standard normal distribution is symmetric we can simply use the positive value for comparing it with the critical z-value.

5.1 g) The calculated z-score is more extreme (higher) than the critical z-value of 1.645. H_0 is rejected with 95% confidence. The PFAS level in town is significantly higher than 0.9 microgram/kg dry soil. There is a risk of 5% for a type-I error. The two methods produce the same answer. It cannot be different; the methods are equivalent.

5.1 h) t-score: $\frac{\bar{x}-\mu}{s/\sqrt{n}} = \frac{0.930-0.9}{0.070/\sqrt{16}} = 1.714$

5.1 i) The calculated t-score of 1.714 is lower than the critical t-value of 1.753. H_0 is not rejected. The PFAS level in town is significantly below the norm. There is a risk type-II error.

5.1 j) Lower bound: $\bar{x} - t_{\alpha(df=15)} \times SE_\mu = 0.930 - 1.753 \times \frac{0.070}{\sqrt{16}} = 0.899$

5.1 k) Yes. The lower bound for the population mean μ is 0.899, it can therefore not be ruled out that the mean PFAS level is below 0.9.

5.1 l) The sample is so much smaller (16 vs. 49) that there is too much uncertainty in the result.

5.2 a) Result: 1.645

This is the z-value for a 95% one-sided confidence interval.

```r
qnorm(p = 0.95, mean = 0, sd = 1, lower.tail = TRUE)
# Or because the standard normal distribution is the default simply use:
qnorm(0.95) # 1.645
```

5.2 b) Because for a two-sided interval you spread the risk over the two tails. There is 2.5% of risk on the left tail and 2.5% of risk on the right tail.

```r
qnorm(p = 0.975) # 1.960
```

5.2 c) The `pnorm()` is the inverse of the `qnorm()` function: It returns the cumulative probability for a given value `q` in a specified normal distribution.

```r
pnorm(q = 1.645, mean = 0, sd = 1) # 0.95
pnorm(1.960) # 0.975
```

5.2 d) The `qt()` function returns the t-value for a given cumulative probability `p` and given number of degrees of freedom `df` .

```r
qt(p = 0.95, df = 15) # 1.753
```

5.2 e) The `pt()` is the inverse of the `qt()` function: It returns the cumulative probability for a given value `q` in a specified t-distribution.

```r
pt(q = 1.753, df = 15) # 0.95
```

5.2 f) t_1: -1.06 t_2: -1.328
t_3: 1.328 t_4: 1.328

```r
qt(p = 0.15, df = 19) # -1.06
qt(p = 0.10, df = 19) # -1.328
qt(p = 0.90, df = 19) # 1.328
qt(p = 0.10, df = 19, lower.tail = FALSE) # 1.328
```

5.2 g) p_1: 0.081 p_2: 0.929 p_2: 0.015

```r
pt(q = -1.5, df = 11) # 0.081
diff(pt(q = c(-2,2), df = 11)) # 0.929
pt(q = 2.5, df = 11, lower.tail = FALSE) # 0.015
```

5.2 h) Two tailed inequality test: 2.467
One-tailed right sided test: 2.153
One-tailed left sided test: -2.153

```r
# Two-tailed inequality test
qt(p = 0.99, df = 28) # 2.467

# One-tailed right sided test
qt(p = 0.98, df = 28) # 2.153

# One-tailed left sided test
qt(p = 0.98, df = 28, lower.tail = FALSE) # -2.153
```

5.2 i) Two-tailed inequality test: H_0 rejected (2.6 > 2.476)
One-tailed right sided test: H_0 rejected (-2.3 < -2.153)
One-tailed left sided test: H_0 not rejected (1.6 < 2.153)

5.3 a) Because different men get the caffeine and the placebo. There are 18 unique test subjects. Everyone gets tested once and every measurement is therefore independent.

5.3 b) H_0: $\mu_1 \leq \mu_2$ H_1: $\mu_1 > \mu_2$

5.3 c) $s_p^2 = \frac{(n_1-1)s_1^2+(n_2-1)s_2^2}{n_1+n_2-2)} = \frac{(9-1)\times 6.49^2+(9-1)\times 8.14^2}{9+9-2} = 7.36$

5.3 d) $t = \frac{(x_1-x_2)-D_0}{\sqrt{s_p^2(\frac{1}{n_1}+\frac{1}{n_2})}} = \frac{(94.11-101.22)}{\sqrt{7.36\times(\frac{1}{9}+\frac{1}{9})}} = 2.050$

5.3 e) The calculated t-score of 2.050 is more extreme (higher) than the critical value of 1.746. H_0 is rejected with 95% confidence. The mean placebo RER level is significantly higher than the mean caffeine RER level. There is a risk of 5% for type-I error.

5.3 f) **placebo**
Mean: 101.22
Standard deviation: 8.14

caffeine
Mean: 94.11
Standard deviation: 6.49

```
# These are the values for the RER test data set
placebo <- c(106, 120, 104, 96, 97, 100, 93, 96, 99)
caffeine <- c(95, 100, 95, 88, 97, 92, 85, 106, 89)


mean(placebo) # Mean: 101.22
sd(placebo) # Standard deviation: 8.14

mean(caffeine) # Mean: 94.11
sd(caffeine) # Standard deviation: 6.49
```

5.3 g) The code runs an independent (not paired) samples t-test with a confidence of 95% for **placebo** and **caffeine** , with the alternative hypothesis H_1 that the difference in means is greater than 0. The variances are assumed equal. The output t-value and p-value confirm the previous result to reject H_0.

```
t.test(x = placebo, y = caffeine, alternative = 'greater', mu = 0,
       paired = FALSE, var.equal = TRUE, conf.level = 0.95)
```

5.3 h) The Welch Two Sample t-test leads to a slightly different p-value of 0.02899, but the same conclusion: rejection of H_0.

```
t.test(x = placebo, y = caffeine, alternative = 'greater', mu = 0,
       paired = FALSE, var.equal = FALSE, conf.level = 0.95)
```

5.3 i) You can use Hartley’s F or Levene’s test.

5.4 a) The observations are not independent because the same twelve people are tested twice. The two blood pressure measurements for one person are connected/dependent: a person with high blood pressure will have higher values in both experiments. That is why in a dependent t-test you look at the difference between the two measurements.

5.4 b) Mean standing: 137.07
Mean lying: 145.43
Mean difference: 8.36

```
# These are the values for the blood pressure data set
standing <- c(136, 144, 152, 133, 140, 129, 131, 133, 145, 134, 142, 140, 125,
135)
sitting <- c(148, 159, 121, 151, 145, 139, 135, 144, 141, 148, 143, 161, 150, 151)

mean(standing) # Mean standing: 137.07
mean(sitting) # Mean sitting: 145.43
differences <- sitting - standing
mean(differences) # Mean difference: 8.36
```

5.4 c) $H_0:\quad \mu_D \leq 0$ $H_1:\quad \mu_D > 0$

5.4 d) The test is done with `alternative = 'greater'`, which means that now R will test for standing greater than sitting, which is quite improbable given the sample results.

```
t.test(x = standing, y = sitting, alternative = 'greater', mu = 0,
       paired = TRUE, conf.level = 0.925)
```

5.4 e) The p-value for this sample outcome is 0.02055, which is below the limit of 0.075 (92.5% confidence). H_0 is rejected with 92.5% confidence. The blood pressure is significantly higher lying down than standing up. There is a risk of 7.5% for type-I error.

```
# Correct: alternative = 'less'
t.test(x = standing, y = sitting, alternative = 'less', mu = 0,
       paired = TRUE, conf.level = 0.925)
```

Chapter 6: Comparing more than two means

6.1 a)

```r
# Be sure to set your working directory when providing a relative path
dataset8 <- read.csv('eyeColor.csv')

ttestData <- subset(dataset8,
                    dataset8$Group == 'Blue'|
                    dataset8$Group == 'Brown')
```

6.1 b) $H_0: \quad \mu_1 = \mu_2$ $\qquad H_1: \quad \mu_1 \neq \mu_2$

6.1 c)

```r
blue <- subset(ttestData$Score, ttestData$Group == 'Blue')
brown <- subset(ttestData$Score, ttestData$Group == 'Brown')
t.test(blue, brown, var.equal = TRUE) # p-value: 0.1401
```

6.1 d) The p-value is 0.1401, which is higher than the 0.05 required to reject H_0. H_0 is not rejected with 95% confidence. You can be 95% confident that the mean of the blue group is the same as the mean of the brown group. There is a risk of a type-II error.

6.1 e) The content of `dummyBrown` is a 0 for blue eyes, and a 1 for brown eyes. This kind of variable is called a dummy variable.

```r
dummyBrown <- as.numeric(ttestData$Group == 'Brown')
ttestData <- cbind(ttestData, dummyBrown)
```

6.1 f)

```r
ttestreg <- lm(formula = Score ~ dummyBrown, data = ttestData)
```

6.1 g)

```r
summary(ttestreg) # p-value dummyBrown: 0.14
```

6.2 a) $H_0: \quad \mu_1 = \mu_2 = \mu_3 = \mu_4$ $\qquad H_1: \quad \mu_1 \neq \mu_2 \neq \mu_3 \neq \mu_4$

6.2 b) $df_M = k - 1 = 4 - 1 = 3$ $\qquad df_R = n - k = 222 - 4 = 218$

6.2 c) Critical F-value: 2.646

```
df1 <- 4 - 1 # 3
df2 <- nrow(dataset8) - 4 # 218
qf(p = 0.95, df1 = df1, df2 = df2) # 2.646
```

6.2 d)

```
anovaResult <- aov(formula = Score ~ Group, data = dataset8)
```

6.2 e) F-value: 2.894 p-value: 0.0362

```
summary(anovaResult)
# F-value: 2.894
# p-value: 0.0362
```

6.2 f) The p-value is 0.0362, which is lower than the 0.05 required to reject H_0. The sample F-value is 2.894, which is higher than the critical F-value of 2.646. H_0 is rejected with 95% confidence. The means of the four groups are not equal to each other. There is a risk of 5% for a type-I error.

6.3 a)

```
dummyBrown <- as.numeric(dataset8$Group == 'Brown') # Brown eyes
dataset8 <- cbind(dataset8, dummyBrown)

dummyBlue <- as.numeric(dataset8$Group == 'Blue') # Blue eyes
dataset8 <- cbind(dataset8, dummyBlue)

dummyGreen <- as.numeric(dataset8$Group == 'Green') # Green eyes
dataset8 <- cbind(dataset8, dummyGreen)
```

6.3 b) anovaReg <- lm(formula = Score ~ dummyBrown + dummyBlue + dummyGreen, data = dataset8)

6.3 c) F-value: 2.894 p-value: 0.0362

```
summary(anovaReg)
# F-value: 2.894
# p-value: 0.0362
# R-squared: 0.0383
```

6.3 d) Yes, the results are the same.

6.4 a)

```
ancovaReg <- lm(formula = Score ~ dummyBrown + dummyBlue +
                                  dummyGreen + initialScore,
                data = dataset8)
```

6.4 b) F-value: 2.252 p-value: 0.064

```
summary(ancovaReg)
# F-value: 2.252
# p-value: 0.064
# R-squared: 0.0398
```

6.4 c) The p-value is 0.0645, which is higher than the 0.05 required to reject H_0. H_0 is not rejected with 95% confidence. The means are equal to each other if you consider the initial score as a covariate. There is a 5% change of a type-I error.

6.4 d) The p-value is of the coefficient of initial score is 0.5539, which means that it is not significantly different from zero. This means that the initial score is not a good predictor of the actual score.

6.4 e) R^2 `anovaReg` : 0.0383 R^2 `ancovaReg` : 0.0398

The `ancovaReg` regression model explains more variation in the outcome variable score.

6.4 f) The groups, together with the initial score, explain 3.98% of the variance in the dependent variable score.

6.4 g) AIC `anovaReg` : 865.54 AIC `ancovaReg` : 867.18

```
AIC(anovaReg) # AIC: 865.54
AIC(ancovaReg) # AIC: 867.18
```

6.4 h) The AIC value of the `anovaReg` regression model is the lowest, which means that the `anovaReg` model fits the data better than the `ancovaReg` regression model. This means that the model without the covariate is a better model. You may have already seen this, since the covariate in the `ancovaReg` model was not a good predictor of the score.

6.5 a)

```
#  Be sure to set your working directory when providing a relative path
load('iowa.RData')
```

6.5 b) The iowa data consists of payments made by the state of Iowa. Payments are assigned to fiscal years that run from July 1 through June 30, and are numbered for the calendar year in which they end. The fiscal year is divided into fiscal periods with 1 being July and 12 being June. The fiscal year also includes a hold-over period for payments made after year end for good and services received on or before June 30.

6.5 c) Rows: 12279009 Columns: 22

```
nrow(iowa) # 12279009 rows
ncol(iowa) # 22 columns
```

6.5 d) Unique services: 8

```
unique(iowa$Service) # 8 unique services
table(iowa$Service)
```

6.5 e) Service: Human Services Rows: 6682159

6.5 f) Number of rows that show a difference: 2624607

```
iowa$Payment.Issue.Date <- as.Date(iowa$Payment.Issue.Date,format= '%m/%d/%Y')
iowa$Invoice.Date <- as.Date(iowa$Invoice.Date,format = '%m/%d/%Y')

length(which(iowa$Payment.Issue.Date != iowa$Invoice.Date)) # 2624607
```

6.5 g)

```
dataDif <- data[which(iowa$Payment.Issue.Date != data$Invoice.Date),]
```

6.5 h)

```
dataDif$dif.days <- dataDif$Payment.Issue.Date - dataDif$Invoice.Date
dataDif$dif.days <- as.numeric(dataDif$dif.days)
```

6.5 i) Minimum: -3651
Mean: 21.815
Maximum: 36529
Upper quartile: 33
Lower quartile: 4
Standard deviation: 89.24

```
min(dataDif$dif.days)
max(dataDif$dif.days)
mean(dataDif$dif.days)
quantile(dataDif$dif.days)
sd(dataDif$dif.days)
```

6.5 j) The default histogram does not provide much information due to the fact that R specifies a very wide *x-axis*.

```
hist(dataDif$dif.days)
```

6.5 k)

```
hist(dataDif$dif.days[dataDif$dif.days > quantile(dataDif$dif.days, 0.05) &
      dataDif$dif.days < quantile(dataDif$dif.days, 0.95)], breaks = 100)
```

6.5 l)

```
dataDif2 <- dataDif[(dataDif$dif.days > (-1)) & (dataDif$dif.days <= 365),]
```

6.5 m)

```
plot(dataDif2$Amount,dataDif2$dif.days)
```

6.5 n) Correlation: -0.0025

```
cor.test(dataDif2$Amount,dataDif2$dif.days)
```

6.5 o) The p-value of the correlation test against the value zero is 3.148e-05, which is sufficient enough to reject H_0 with 95% confidence. This implies that there is, with 95% certainty, a correlation between the the time between invoice and payment, and the amount that is paid.

6.5 p) Administration and regulation: 26.452
Agriculture and natural resources: 28.275
Capital: 35.490
Economic development: 25.815
Education: 23.868
Human services: 18.508
Justice system: 25.478

```
aggregate(dataDif2$dif.days, by = list(dataDif2$Service), FUN = mean)
```

6.5 q) p-value: < 2e-16

Conclusion: The p-value is lower than 0.05, so you can reject H_0 with 95% confidence. This means that the means of all expense categories are not equal. There is a 5% type change of a type-I error.

```
aovRes <- aov(dif.days ~ Service, data = dataDif2)
summary(aovRes)
```

6.5 r) All means, except for the means of the justice system expenses and the economic development expenses, show a p-value below 5% and can thus be regarded to differ from each other.

```
tukeyRes <- TukeyHSD(aovRes)
```

6.5 s) An ANOVA assumes the dependent variable to be continuous. Some examples of appropriate analyses could be:

```
# Poisson regression and then interpret the predictors
poissonReg <- glm(dif.days ~ Expense.Category, family = poisson, data = dataDif2)
summary(poissonReg)

# Kruskal Wallis test and Dunn test to compare individual groups
kruskRes <- kruskal.test(dif.days ~ Service, data = dataDif2)
# install.packages('FSA'); library(FSA)
dunn.res <-dunnTest(dataDif2$dif.days, dataDif2$Service)
```

6.5 t) It is a bad idea, the p-value is affected by the number of samples. The higher the sample, the lower the p-value gets. In other words, p-values lose their meaning quite quickly (unless they are non-significant).

Chapter 7: Comparing proportions and distributions

7.1 a) H_0: The 2019 distribution is equal to the historical distribution
H_1: The 2019 distribution is not equal to the historical distribution

7.1 b) Expected number = Historical % × Observed number

	Historical	Observed (O)	Expected (E)	$O - E$	$\frac{(O-E)^2}{E}$
Spring	4.87 (30%)	2.90	25.2	11.8	5.525
Summer	3.04 (40%)	4.50	33.6	-6.6	1.296
Fall	1.65 (15%)	4.94	12.6	-0.6	0.029
Winter	2.88 (15%)	3.28	12.6	-4.6	1.679
Total	2.31 (100%)	4.73	84		8.530

7.1 c) Since every season has an expected value above 5 you can do a chi-square test.

7.1 d) $X^2 = 8.530$

7.1 e) The calculated chi-square value of 8.530 is higher than the critical chi-square value of 7.8. H_0 is rejected. The 2019 distribution is significantly different from the historical distribution. There is a 5% risk of a type-I error.

7.1 f) You rejected the null hypothesis H_0 with 95% confidence, and so the p-value must be lower than 0.05.

7.1 g) $X^2 = 8.530$

```r
Observed <- c(37, 27, 12, 8)
Historical <- c(0.3, 0.4, 0.15, 0.15)
Expected <- c(25.2, 33.6, 12.6, 12.6)

chi <- sum((Observed - Expected)^2 / Expected) # 8.530
```

7.1 h) Yes.

```r
qchisq(p = 0.95, df = 3) # 7.185
```

7.1 i) R returns the same chi-squared as calculated, so the answer was correct. It also shows a p-value of below 0.05, as expected.

```
# chi-square test: x = observations p = model distribution
# rescale makes sure the model distribution adds up to 100%
chisq.test(x = Observed, p = Historical, rescale.p = TRUE)

# Chi-squared value: 8.5298
# p-value: 0.0362
```

7.1 j) The expected values are 25.2, 33.6, 12.6, and 12.6. R shows the same expected values.

```
chisq <- chisq.test(x = Observed, p = Historical)
chisq$expected # Extract expected values with $expected
# 25.2 33.6 12.6 12.6
```

7.2 a) The `sales` data frame contains 3 columns: `month` , `historical` and `newstore` . It contains the 'Historical' and 'New Store' distribution of sales over the months.

```
# These are the values for the sales data set
sales <- data.frame(month = seq(from = 1, to = 12, by = 1),
                    historical = c(5.1, 5.1, 6.7, 10, 11.4, 10,
                                   6.7, 5.1, 6.7, 10, 11.7, 11.7),
                    newstore = c(5.6, 6.2, 9.4, 8.6, 6.8, 4.8,
                                 5.6, 4.8, 8.8, 12.6, 13.1, 13.7))

summary(sales)
```

7.2 b) The chi-square test requires every cell in the expected distribution to have a value of at least 5 and it requires the total of both groups to be equal. Since the historical distribution contains more than 5 in every cell and both observed and expected values add up to the same number (100) we can use this for a chi-squared test.

7.2 c) H_0: The new distribution is equal to the historical distribution
H_1: The new distribution is not equal to the historical distribution

7.2 d) The p-value of 0.6963 is higher than the critical p-value of 0.10. H_0 is not rejected. The new store distribution is not significantly different from the historical distribution. There is a risk a type-II error.

```
chisq.test(x = sales$newstore, p = sales$historical, rescale.p = TRUE)
# p-value: 0.6963
```

7.3 a) The best estimate of the population proportion π is the sample proportion p.

$$\pi_1 = \frac{k}{n} = \frac{8}{71} = 0.113$$

$$\pi_2 = \frac{k}{n} = \frac{16}{111} = 0.144$$

7.3 b) Confidence interval sample 1:

$$p \pm z_\alpha \times \sqrt{\frac{p(1-p)}{n}} = 0.113 \pm 1.960 \times \sqrt{\frac{0.113 \times (1-0.113)}{71}} = [0.039;\, 0.187]$$

Confidence interval sample 2:

$$p \pm z_\alpha \times \sqrt{\frac{p(1-p)}{n}} = 0.144 \pm 1.960 \times \sqrt{\frac{0.144 \times (1-0.144)}{111}} = [0.078;\, 0.210]$$

7.3 c) H_0: $\pi_2 \leq \pi_1$ H_1: $\pi_2 > \pi_1$

Where π_2 and π_1 are the success proportions for the evening and afternoon calls respectively.

7.3 d) Combined success probability:

$$p^* = \frac{k_1+k_2}{n_1+n_2} = \frac{8+16}{71+111} = 0.132$$

7.3 e) Combined standard error:

$$s_p = \sqrt{p^*(1-p^*)(\frac{1}{n_1} + \frac{1}{n_2})} = \sqrt{0.132(1-0.132)(\frac{1}{71} + \frac{1}{111})} = 0.0514$$

7.3 f) z-score: $\frac{p_1-p_2}{s_p} = \frac{0.144-0.113}{0.514} = 0.612$

Note that p_1 and p_2 were switched because in the hypotheses π_1 and π_2 were also switched.

7.3 g) The calculated z-score of 0.612 is lower than than the critical z-value of 1.645. H_0 is not rejected. The evening success rate is not shown to be significantly higher than the afternoon success rate. There is a risk of a type-II error.

7.3 h) The code creates a vector of successes **k** and a vector of sample sizes **n** . The proportion test then tests the equality. It shows the proportions you calculated earlier, and a p-value of 0.6984 which supports your conclusion if you do not reject H_0.

```
n <- c(71, 111)
k <- c(8, 16)
prop.test(x = k, n = n)
# p-value: 0.6984
```

Note that R actually returns a chi-squared value. Because it actually does a chi-square test it can in fact be used to test more than two proportions at the same time.

Chapter 8: Bayesian statistics

8.1 a) The prior distribution on the left of the figure assigns equal mass to every value of θ, and therefore does not incorporate relevant information about which values of θ are more likely to occur. The prior distribution in the middle reflects the belief that the probability of heads is either very close to zero, or very close to one. The prior distribution on the right reflects the information that it is most likely that the coin is a fair coin (highest probability at 0.5), but that there is some uncertainty about θ around this value.

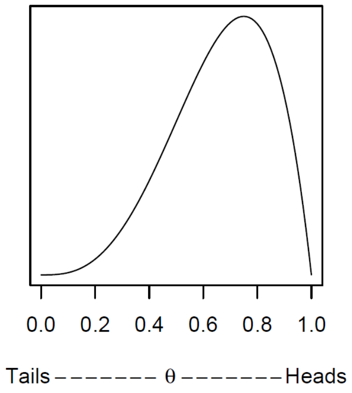
8.1 b)

Middle:		Right	
α:	0.5	α:	2
β:	0.5	β:	2

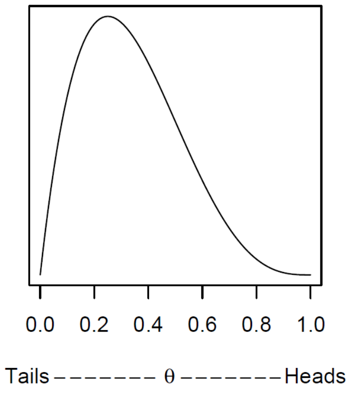
```
alpha <- 0.5
beta <- 0.5
curve(dbeta(x, alpha, beta), xlab = expression(theta), ylab = '', yaxt = 'n')

alpha <- 2
beta <- 2
curve(dbeta(x, alpha, beta), xlab = expression(theta), ylab = '', yaxt = 'n')
```

8.1 c)

Biased towards heads:



Biased towards tails:



8.2 a) The Greek letter θ represents the proportion of customers that leaves the store feeling satisfied in this case.

8.2 b) An example of prior parameters can be $\alpha = 7$ and $\beta = 5$. This combination yields a prior distribution that assigns most mass at the point $\theta = 0.6$, but expresses relatively much uncertainty about θ. However, other values of the parameters can also be correct, since the prior reflects your own beliefs. Higher values of α and β give prior distributions that are less wide, and this reflect more certain prior beliefs.

8.2 c) The answer depends on your own choices of α and β in assignment 8.2b. In the following answers, $alpha = 7$ and $beta = 5$ are used.

```
curve(dbeta(x, shape1 = 7, shape2 = 5),
      xlab = expression(theta), ylab = '', yaxt = 'n')
```

8.2 d) α: $7 + 33 = 40$ β: $5 + 40 - 33 = 12$

8.2 e) Probability: 0.995

```
pbeta(0.6, shape1 = 40, shape2 = 12, lower.tail = FALSE) # 0.995
```

8.2 f) Probability: 0.875

```
diff(pbeta(c(0.7, 0.9), shape1 = 40, shape2 = 12)) # 0.875
```

8.2 g) The posterior distribution is fairly robust to changes in the prior distribution. In Bayesian inference, the data quickly overwhelm the prior distribution when it comes to parameter estimation. Therefore, the more data you have the more robust the posterior distribution generally is.

8.3 a)

```
samples <- extract(stanFit)
```

8.3 b) The histogram approximates the analytical posterior pretty well. The more samples you draw, the better the histogram will resemble the actual posterior distribution.

```
hist(samples, breaks = 100, probability = TRUE)
curve(dbeta(x, shape1 = 40, shape2 = 12), add = TRUE)
```

8.3 c)

```
modelCode <- '
data {
     int n;
     int k;
}
parameters {
     real<lower=0,upper=1> theta;
}
model {
     theta ~ normal(0.6, 0.1);
     k ~ binomial(n, theta);
}
'

# Note: The following line can take a while to execute
compiledModel <- stan_model(model_code = modelCode, model_name = 'model')

stanFit <- sampling(compiledModel, data = list(n = 40, k = 33),
                    iter = 5000, warmup = 500, chains = 4)
```

8.3 d) Probability: 0.990

```
length(which(samples$theta > 0.6)) / length(samples$theta) # 0.990
```

8.3 e) Probability: 0.796

```
length(which(samples$theta >= 0.7 & samples$theta <= 0.9)) /
length(samples$theta) # 0.796
```

8.3 f) The answers of assignments 8.3d and 8.3e do not differ substantially from those of assignments 8.2e and 8.2f. Overall you can conclude that your posterior distribution is fairly robust to changes in the prior distribution. You can therefore be highly sure that the percentage of satisfied customers is higher than 60 percent, and reasonably sure that the percentage lies between 70 and 90 percent.

8.4 a) H_0: $\theta = 0.5$ H_0: $\theta \neq 0.5$

8.4 b) An example of a possible answer can be $\alpha = 3$ and $\beta = 3$. This prior distribution is fairly wide, and thus expresses not much confidence in your programming skills.

$\theta \sim Beta(3, 3)$

8.4 c) Assuming the values $\alpha = 2$ and $\beta = 2$:

α: 3 + 22 = 25 β: 3 + 30 - 22 = 11

8.4 d)

```
alpha <- 3
beta <- 3
n <- 30
k <- 22

# Plot the posterior distribution first so no adjustment of the axes is
needed curve(dbeta(x, alpha + k, beta + n - k),
      bty = 'n', las = 1, lty = 1, yaxt = 'n', ylab = '', xlab = expression(theta))
curve(dbeta(x, alpha, beta), add = TRUE, lty = 2)
```

8.4 e)

```
heightPrior <- dbeta(0.5, alpha, beta)
```

8.4 f)

```
heightPosterior <- dbeta(0.5, alpha + k, beta + n - k)
```

8.4 g) The value of the Bayes factor `BF10` is 7.02. This implies that the data are 7.18 times more likely to occur under the alternative hypothesis $H_1 : \theta \neq 0.5$ than under the null hypothesis $H_0 : \theta = 0.5$.

8.4 h) The strength of the evidence is moderate, and so you cannot be very confident that your algorithm performs better than change. You will need to collect more data if your want a more decisive Bayes factor.

8.5 a) M_1: $\theta \leq$ 0.75 M_2: $\theta \geq$ 0.75

8.5 b)

```
model2code <- '
data {
    int n;
    int k;
}
parameters {
    real<lower=0.75,upper=1> theta;
}
model {
    theta ~ beta(1, 1)T[0.75, 1];
    k ~ binomial(n, theta);
}
'

# Note: The following line can take a while to execute
model2 <- stan_model(model_code = model2code, model_name = 'model2')
```

8.5 c) The prior distribution for θ in M_1 is a $Beta(1,1)$ prior distribution truncated to the [0, 0.75] interval (`beta(1, 1)T[0, 0.75]`). The prior distribution for θ in M_2 is a $Beta(1,1)$ prior distribution truncated to the [0.75, 1] interval (`beta(1, 1)T[0.75, 1]`).

8.5 d)

```
stanFitM1 <- sampling(model1, data = list(n = 156, k = 123),
                      iter = 5000, warmup = 500, chains = 4)

stanFitM2 <- sampling(model2, data = list(n = 156, k = 123),
                      iter = 5000, warmup = 500, chains = 4)
```

8.5 e)

```
mLike1 <- bridge_sampler(stanFitM1)
mLike2 <- bridge_sampler(stanFitM2)
```

8.5 f)

```
BF12 <- bf(mLike2, mLike1)
```

8.5 g) The value of `BF12` is around 17.77. This indicates that the data is 17.77 times more likely to occur under model $M_2 : \theta \geq 0.75$ than under model $M_2 : \theta \leq 0.75$.

8.5 h) The evidence in favor of M_2 is moderate, and so additional testing might be required to come to a reasonable conclusion.

8.5 a)

```
# Be sure to set your working directory when providing a relative path
insurance <- read.csv('insurance.csv')
```

8.5 b) charges = $\beta_0 + \beta_1 \times$ age $+\beta_2 \times$ bmi $+\beta_3 \times$ neighbors

8.5 c) β_0: -6670.25 β_1: 333.92 β_2: 241.56 β_3: 45.84

```
regressionFit <- sampling(object = regressionModel,
                          data = list(N = nrow(insurance),
                                      x1 = insurance$bmi,
                                      x2 = insurance$age,
                                      x3 = insurance$neighbors,
                                      y = insurance$charges),
                          iter = 5000, warmup = 500, chains = 4)

regressionSamples <- extract(regressionFit)
summary(regressionSamples)
```

8.5 d) YES

```
summary(lm(charges ~ 1 + age + bmi + neighbors, data = insurance))
```

8.5 e)

```
layout(matrix(1:4, nrow = 2))

plot(density(regressionSamples$beta1), xlab = '', ylab = '',
     yaxt = 'n', bty = 'n', main = expression(beta[1]))
plot(density(regressionSamples$beta0), xlab = '', ylab = '',
     yaxt = 'n', bty = 'n', main = expression(beta[0]))
plot(density(regressionSamples$beta2), xlab = '', ylab = '',
     yaxt = 'n', bty = 'n', main = expression(beta[2]))
plot(density(regressionSamples$beta3), xlab = '', ylab = '',
     yaxt = 'n', bty = 'n', main = expression(beta[3]))

layout(1)
```

8.5 f) Probability: 0.966

```
length(which(regressionSamples$beta2 > 200)) / length(regressionSamples$beta2)
```

8.5 g) `Probability: 0.052`

```
length(which(regressionSamples$beta1 < 250)) / length(regressionSamples$beta1)
```

8.5 h) `Lower bound: -79.98` `Upper bound: 169.74`

```
quantile(regressionSamples$beta3, probs = c(0.10, 0.90))
```

8.5 i)

```
regressionmodelcode2 <- '
data {
     int<lower=0> N;
     vector[N] x1;
     vector[N] x2;
     vector[N] y;
}
parameters {
     real beta0;
     real beta1;
     real beta2;
     real<lower=0> sigma;
}
model {
     y ~ normal(beta0 + beta1 * x1 + beta2 * x2, sigma);
}
'

# Note: The following line can take a while to execute
regressionModel2 <- stan_model(model_code = regressionmodelcode2,
                               model_name = 'Regression2')
```

8.5 j)

```
regressionFit2 <- sampling(object = regressionModel2,
                           data = list(N = nrow(insurance),
                                       x1 = insurance$bmi,
                                       x2 = insurance$age,
                                       y = insurance$charges),
                           iter = 5000, warmup = 500, chains = 4)

mLike1 <- bridge_sampler(regressionFit)
mLike2 <- bridge_sampler(regressionFit2)

BF12 <- bf(mLike1, mLike2)
```

8.5 k) The value of **BF12** is 258.65. This implies that the data are 258.65 times as likely to be observed under the model where β_3 is not included in the regression equation under the model where β_3 is included. The evidence against the number of **neighbors** adding to the prediction accuracy is extreme.

Practical assignment

Question	**Points**	**Criteria**
1	1 point	The data set follows the set criteria of size and parameters.
	1 point	Relevant research question that connects to business scenario.
	1 point	Research question can be answered properly using the collected data.
2	1 point	Correct calculation of appropriate descriptive statistics.
	1 point	Relevant and appropriate figures included.
3a	1 point	Explained choice of statistical test that is consistent with the research question.
	1 point	The hypotheses are correctly formulated.
3b	1 point	The right formulas are selected, and the calculation is explained clearly.
	1 point	Correct calculation of appropriate corresponding test statistic.
3c	1 point	Correct definition of critical region for test statistic.
	1 point	Correct conclusion is drawn and explained using the test statistic and critical region.
4a	1 point	Correct parameters are selected regarding the research question.
	1 point	The hypotheses are correctly formulated.
4b	1 point	Correctly performed linear regression using R.
	1 point	Relevant output and statistics are shown.
	1 point	Interpretation of R results is provided, the test statistic is identified.
	1 point	Correct conclusion is drawn and explained using the test statistic.
	1 point	The conclusions drawn from the sample are related to the population of interest.
	1 point	Correct discussion of generalization of results with respect to representativity.
	1 point	Correct interpretation on the type I/II risks that apply to the conclusions.
Total	20 points	

Beginner R exercises

B 1) The code created two variables (`a` and `b`) in your environment. The value of variable `a` is 2 (although at first it is specified as 1) and the value of variable `b` is 1. It does not matter whether variables are assigned with the `=` or the `<-` operator.

B 2)

```
rm(a); rm(b)
```

B 3) Adding up `apples` and `pears` equals 11.
The variable `Pears` does not exist. Instead, you should add `apples` and `pears` together (notice the lack of a capital P).

B 4)

```
t1 <- sqrt(81)
```

B 5)

```
t2 <- 81^0.5
```

B 6)

```
t1 == t2
```

B 7) The working directory is a file path on your computer that sets the default location of any files you read into R, or save out of R. In other words, a working directory is like a little flag somewhere on your computer which is tied to a specific analysis project. If you ask R to import a dataset from a text file, or save a data frame as a text file, it will assume that the file is inside of your working directory.

B 8) You can change the working directory using the `setwd()` function. You can check whether your change worked by calling the `getwd()` function again and checking whether it is now at the folder you specified in `setwd()` . Note that RStudio also has the option to change the working directory from the "Session" dropdown menu, then select "Set Working Directory".

B 9) The code shows all the available demos that are built into R. The `persp` demo can be viewed with:

```
demo(persp)
```

B 10) It gives `NA` because the `mean()` function does not handle `NA` 's by default. You can compute the mean without the missing value by setting `na.rm = TRUE` .

B 11) Just typing `mvrnorm()` gives an error because the package `MASS` (from which the function comes) is not loaded yet. You can load the `MASS` package by typing:

```
library(MASS)
```

B 12) The `%%` operator gives the modulo (the remaining number after division) of the first number divided by the last number. You cannot find help by typing `?%%` , but you can search Google for help on how to use this operator. Note that in RStudio you can also type `%%` into the search bar in the Help section (in the Files and Plots part of RStudio, usually displayed at the bottom right of your screen).

B 13) You can test the correlation and find the confidence interval for the correlation by using the `cor.test()` function. You can find all function that have 'cor' in their name by typing `apropos('cor')` .

```
cor.test(c(1, 2, 3, 4), c(1, 4, 7, 15))
```

B 14) The modes of `a1` , `a2` , and `a3` are character, numeric, and logical respectively. The modes of `b1` , `b2` , `b3` , and `b4` are character, character, numeric, and character respectively. When vectors of different modes are combined R converts the vectors to one and the same mode, because elements in a vector can only be of one mode.

B 15) As a standard, R converts `TRUE` to a 1 and `FALSE` to a 0.
$^{1}/_{1} = 1$ $^{0}/_{1} = 1$ $^{0}/_{0} =$ undefined $^{1}/_{0} = \infty$

B 16) In the first case the numeric `1` is converted to character mode, while in the last case the logical `TRUE` is converted to character mode.

```
as.numeric(TRUE)
as.character(TRUE)
```

B 17) You can check whether a vector is numeric by using the `is.numeric()` function. `c(1, 0)` is numeric; `c(TRUE, FALSE)` is not; `c(TRUE, FALSE, 1, 0)` is numeric because R automatically converts `TRUE` and `FALSE` to a `1` and a `0` because vectors have to be of one mode.

B 18) The length of this vector is 11.

```
v1 <- c(-2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8)
# or
v1 <- -2:8
length(v1)
```

B 19)

```
v2 <- seq(from = -5, to = 5, by = 0.5)
v2 <- (-10:10)/2
```

B 20)

```
v3 <- seq(from = 13, to = 33, by = 2)
```

B 21)

```
c(a, b)
```

B 22) The result of `rep(2, 10)` is a repeated vector containing 10 times the number 2. The function argument `times = 10` is implicitly set.

B 23)

```
rep(3:7, each = 3)
```

B 24) You have to explicitly specify that the 2 refers to the `each` argument. By default, the second argument of the `rep()` function is times, see also `?rep` .

B 25) The sum of these numbers is 100.

```
sum((1:10)*2-1)
```

B 26)

```
v3 <- (1:10) * 2
v4 <- v3 / 5
v5 <- seq(5, 32, 3)
```

B 27)

```
vector('logical', 5)
```

B 28)

```
paste(rep(c('x', 'y'), each = 4), rep(1:2, each = 2), c('m', 'f'), sep = '')
```

B 29) The options `decreasing = TRUE` can be set to sort the numbers from highest to lowest. The default is `FALSE` , so sorting from lowest to highest does not require specification of this option.

```
sort(s, decreasing = TRUE) # Highest to lowest
sort(s) # Lowest to highest
```

B 30) R is actually pretty smart and will recycle values for you. 10 is not divisible by 3, hence full recycling of all numbers is not possible.

B 31)

```
matrix(25:1, nrow = 5, byrow = TRUE)
```

B 32)

```
matrix(rep(0:1, 8), nrow = 4)
```

B 33) You do not have to specify the number of rows (only the number of columns) because, given the number of values and the number of columns, the number of rows for the matrix is known.

B 34) You can transpose a matrix using the `t()` function.

B 35)

```
colMeans(m1)
```

B 36) You can select the diagonal of a matrix using the `diag()` function.

```
m2 <- matrix(rep(0, 16), nrow = 4)
diag(m2) <- diag(m1)

# or:
m2 <- diag(diag(m1))
```

B 37)

```r
m1 <- rbind(m1, rowSums(m1))
```

B 38) The means of the columns are all zero (calculate with `rowSums(m1)`), and the standard deviations are all one (calculate with `apply(m2,2,sd)` . The `scale()` function therefore transforms the values in the columns so that their mean is zero and their standard deviation is one.

B 39)

```r
data.frame(subject = rep(1:5, each = 2),
           time = c('t1', 't2'),
           score = c(7, 8, 8, 8, 9, 8, 9, 7, 7, 6))
```

B 40)

```r
as.data.frame(a)
```

B 41)

```r
list(subject = 1:5,
     time = c('t1', 't2'),
     score = matrix(1:25, 5, 5))
```

B 42) No, `g` is not a true vector but a factor. That is because `g` consist of factor levels. You can check this using `is.vector(g)` or `is.factor(g)` . Note that you can also find this out by using the `str()` function: `str(g)` .

B 43)

```r
v <- as.factor(v)
```

B 44) The factor `x` has three levels, which you can find out using the `levels()` function.

```r
x <- as.factor(x)
levels(x)
```

B 45) The value of `x[2]` is `NA` .

B 46)

```
a[1] <- 8
a[a == 2] <- 0
```

B 47)

```
l[2, 3]
```

B 48)

```
l[c(3, 5), ]
```

B 49)

```
m[m[, 1] < 5, ]
```

B 50)

```
m[ , -4]
```

B 51)

```
m <- as.data.frame(m)
colnames(m) <- paste('trial', 1:10, sep = '.')
```

B 52)

```
m$'trial.1'[2:5]
m$'trial.4'[1:2]
```

B 53)

```
b <- ifelse(b == 1, yes = 2, no = 1)
```

B 54)

```
gsub('a', '.', n)
```

B 55)

```
passed <- grades[rowMeans(grades[, c('exer','exam')]) > 5.5 &
                 grades[, 'exer'] >= 5 &
                 grades[, 'exam'] >= 5 , 1]
failed <- grades$student[-passed]
```

B 56) The condition `(a < 0 | b < 0)` `(a * b < 0)` means that `a` is smaller than `0` or `b` is smaller than `0` , and `a * b` is smaller than `0` . If this condition is `TRUE` and `a` is negative, then `b` must be positive.

B 57) The number of `NA` ’s in the vector is 1, hence it is smaller than or equal to 2. So, `sum(is.na(c(1, 2, 3, 4, NA, 7))) <= 2` returns `TRUE` , and its logical negation (`!`) is therefore `FALSE` .

B 58) These are the numbers by which the number 24 can be divided (24 is divisible by 1, 2, 3, 4, 6, 8, 12, and 24).

B 59) This code tests whether two vectors are identical. The `identical()` function does this automatically.

B 60) You can use the `table()` function to create a frequency table of the throws.

B 61)

```
sample(c('jack', 'queen', 'king', 'ace'), 4, replace = TRUE)
```

B 62) You can use the `runif()` function to sample uniform random numbers. `sample(1:100, 20)` is not correct in this case because the sampling vector starts at 1 and is generating integers instead of floating point numbers.

```
runif(n = 20, min = 0, max = 100)
```

B 63)

```
r1 <- runif(n = 21, min = 0, max = 100)
median(r1)
r2 <- sort(r1, decreasing = TRUE)
r2[11]
```

B 64) Simulating data using the same code twice does not result in the same samples because of the inherent randomness of a simulation. You can use the `set.seed()` function to make your code reproducible.

```
set.seed(123)
r3 <- rnorm(n = 100, mean = 100, sd = 15)
var(r3)
```

B 65)

```
cov(x)
cor(x)
```

B 66) The `read.csv()` function reads in `.csv` files. You can read in Excel files (`.xlsx`) using the `read.xslx()` function (from the `xlsx` package). The function `read.spss()` is featured in the package `foreign` (see `?read.spss`) and reads `.spss` files. For people that were born before 1980, we acknowledge that you can also read in `.dbf` files with `read.dbf()` .

B 67)

```
# Be sure to set your working directory when providing a relative path
dataset <- read.csv('example.csv')
```

B 68)

```
# Be sure to set your working directory when providing a relative path
write.csv(d, file = 'example.csv')
```

This page is intentionally left blank.