Statistics with R - Exercise 1

Philipp Satlawa - h0640348

06/11/2020

This document contains the answered questions of exercise 1 for the course "Statistics with R".

Task 1 - Sequences

Creation and comparison of vectors.

1. Create a vector x that contains the sequence of even numbers from 0 to 40 ($x \in \mathbb{R}$)

```
(x <- seq(from = 0, to = 40, by = 2))
```

```
##  [1]  0  2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
```

2. Create a vector y, which contains the elements of vector x but in *random* order ($y \in x$)

```
# set seed to produce reproducible results
set.seed(10)
# take random samples from the elements of vector x
(y <- sample(x))
```

```
##  [1] 20 16 18 30 22 12 14 10 34  4 28 36 24  2 40  8 32 38  6  0 26
```

3. The values of x and y agree on the following positions:

```
# number of positions the two vectors agree on
length(which(x == y))
```

```
## [1] 2
```

```
# indices (positions) on which the values of the two vectors agree on
which(x == y)
```

```
## [1] 13 17
```

Task 2 - Sequences

Verification of three approximation formulas for π. Since the calculation of π by the following formulas includes infinite sums or products, we use a large $n = 10000$.

Creating vector `i` as a sequence from 1 to 100000 ($\mathtt{x} \in \mathbb{R}$).

```
# create vector i
i <- 1:10000
```

1. John Wallis (1616-1703):

```
# calculation of pi with the formula of John Wallis
prod((2*i/(2*i-1)) * (2*i/(2*i+1)))*2
```

```
## [1] 3.141514
```

2. Gottfried Leibnitz (1646-1716):

```
# calculation of pi with the formula of Gottfried Leibnitz
sum(((-1)**(i+1))/(2*i-1))*4
```

```
## [1] 3.141493
```

3. Leonhard Euler (1707-1783):

```
# calculation of pi with the formula of Leonhard Euler
sqrt(sum(1/i**2)*6)
```

```
## [1] 3.141497
```

To compute the smallest relative deviation from π given the vector `i`, we need π's current best approximation. Acording to ikipedia (2020) pi's current best approximation is 3.14159265358979323846264 33.

```
# store pi's current best approximation for comparison
pi <- 3.14159265358979323846264 33

# calculate the difference between John Wallis (1616-1703) and current pi
(diff_JW <- abs(prod((2*i/(2*i-1)) * (2*i/(2*i+1)))*2 - pi))
```

```
## [1] 7.853491e-05
```

```
# calculate the difference between Gottfried Leibnitz (1646-1716) and current pi
(diff_GL <- abs(sum(((-1)**(i+1))/(2*i-1))*4 - pi))
```

```
## [1] 1e-04
```

```r
# calculate the difference between Leonhard Euler (1707-1783) and current pi
(diff_LE <- abs(sqrt(sum(1/i**2)*6) - pi))
```

```
## [1] 9.548964e-05
```

```r
# get the absolute difference of the formula with the smallest absolute difference
min(diff_JW, diff_GL, diff_LE)
```

```
## [1] 7.853491e-05
```

Hance, the approximation formula for π with the smallest relative deviation for $n = 10000$ is John Wallis (1616-1703) formula.

Wikipedia (2020), https://en.wikipedia.org/wiki/Approximations_of_%CF%80

Task 3 - Vectors

Creation and manipulation of vectors.

1. Create vector `x` containing the sequence 1 to 100 ($x \in \mathbb{R}$) and vector 'y' containing a sample of size n = 70 of the sequence 1 to 150 ('y' $\in \mathbb{R}$).

```r
# create vector x containing the sequence 1 to 100 (natural numbers)
(x <- 1:100)
```

```
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  [91]  91  92  93  94  95  96  97  98  99 100
```

```r
# set seed to produce reproducible results
set.seed(10)
# create vector y containing a sample of size n = 70 of the sequence 1 to 150 (natural numbers)
(y <- sample(1:150, 70, replace = TRUE))
```

```
##  [1] 137  74 112  72  88  15 143  74  24  13  95 136 110   7  86  82  29  29 121
## [20]  92  50 109 101 122  33 135  68  93 114  88  51  32  11  79  92  91  42  78
## [39]  13 105 144 117  26  89  48  15 110  24  61 132  14  35  10  74  58 144  15
## [58]  31 138 101 101 109  39 118  89  18 131  42 138  79
```

2. Determine the amount of elements that are contained in `x` but not in `y`.

```r
# elements of x not contained in y
(setdiff(x,y))
```

```
##  [1]   1   2   3   4   5   6   8   9  12  16  17  19  20  21  22  23  25  27  28
## [20]  30  34  36  37  38  40  41  43  44  45  46  47  49  52  53  54  55  56  57
## [39]  59  60  62  63  64  65  66  67  69  70  71  73  75  76  77  80  81  83  84
## [58]  85  87  90  94  96  97  98  99 100
```

```r
# number of elements of x not contained in y
(length(setdiff(x,y)))
```

```
## [1] 66
```

3. Check for duplicate elements in `y` and depending if there are duplicate elements or not create a different `z`.

```r
# check if there are duplicate elements in y
if(length(y[duplicated(y)]) > 0) {
  # create a new vector z containing the duplicate elements of y
  z <- y[duplicated(y)]
} else {
  # create a new vector z ac a copy of y
  z <- y
}
```

4. Determine the number of elements of `z` that are multiples of 3.

```r
# calculate the number of elements of z that are multiples of 3
length(z[z %% 3 == 0])
```

```
## [1] 6
```

5. Revert the `y` without using the function `rev()`.

```r
# revert vector y
y[length(y):1]
```

```
##  [1]  79 138  42 131  18  89 118  39 109 101 101 138  31  15 144  58  74  10  35
## [20]  14 132  61  24 110  15  48  89  26 117 144 105  13  78  42  91  92  79  11
## [39]  32  51  88 114  93  68 135  33 122 101 109  50  92 121  29  29  82  86   7
## [58] 110 136  95  13  24  74 143  15  88  72 112  74 137
```

Task 4 - Point Estimation

Assuming a normally distributed population, we create random sample and estimate μ and σ^2 for this sample.

1. Draw a reproducible sample of size $n = 30$ from a normal distribution with $\mu = 5$ and $\sigma^2 = 4$.

```r
# set seed to produce reproducible results
set.seed(10)
# draw random sample with n = 30, mu = 5, sigma^2 = 4
(x <- rnorm(30, mean = 5, sd = sqrt(4)))
```

```
##  [1] 5.0374923 4.6314949 2.2573389 3.8016646 5.5890903 5.7795886 2.5838476
##  [8] 4.2726480 1.7466546 4.4870432 7.2035590 6.5115630 4.5235329 6.9748894
## [15] 6.4827803 5.1786945 3.0901123 4.6096992 6.8510425 5.9659570 3.8073787
## [22] 0.6294263 3.6502681 0.7618776 2.4696040 4.2526769 3.6248891 3.2556823
## [29] 4.7964780 4.4924389
```

2.Estimate μ and σ^2 on the basis of your sample using the formulas to estimate the population mean μ with the sample mean $\overline{x}$ and the population variance σ^2 with the empirical variance s^2, without using the functions `mean()`, `var()` and `sd()`.

```r
# calculation of mean
(sum(x)/length(x))
```

```
## [1] 4.310647
```

```r
# calculation of variance
(1/(length(x)-1) * sum((x-mean(x))**2))
```

```
## [1] 3.00578
```

```r
# calculation of standard deviation
(sqrt(1/(length(x)-1) * sum((x-mean(x))**2)))
```

```
## [1] 1.733718
```

3. Compare your results with the output of the functions `mean()` and `var()`.

```r
# compute the mean with inbuilt function mean()
(mean(x))
```

```
## [1] 4.310647
```

```r
# check if the results of the inbuilt function mean() and the formula for calculating the mean are the
if(round(sum(x)/length(x),5) == round(mean(x),5)) {
  ("same")
} else {
  ("diffrent")
}
```

```
## [1] "same"
```

```r
# compute the variance with inbuilt function var()
(var(x))
```

```
## [1] 3.00578
```

```r
# check if the results of the inbuilt function var() and the formula for calculating the variance are t
if(round((1/(length(x)-1) * sum((x-mean(x))**2)),5) == round(var(x),5)) {
  ("same")
} else {
  ("diffrent")
}
```

```
## [1] "same"
```

```r
# compute the standard variation with inbuilt function sd()
(sd(x))
```

```
## [1] 1.733718
```

```r
# check if the results of the inbuilt function sd() and the formula for calculating the variance are th
if(round(sqrt(1/(length(x)-1) * sum((x-mean(x))**2)),5) == round(sd(x),5)) {
  ("same")
} else {
  ("diffrent")
}
```

```
## [1] "same"
```

4. Are your estimates close to the population values? Repeat the steps 1 and 3 from above with a sample of size n = 3000. What do we learn?

```r
# set seed to produce reproducible results
set.seed(10)
# draw random sample with n = 3000, mu = 5, sigma^2 = 4
x <- rnorm(3000, mean = 5, sd = sqrt(4))

# calculation of mean
(sum(x)/length(x))
```

```
## [1] 5.002634
```

```r
# calculation of variance
(1/(length(x)-1) * sum((x-mean(x))**2))
```

```
## [1] 4.143197
```

```r
# calculation of standard deviation
(sqrt(1/(length(x)-1) * sum((x-mean(x))**2)))
```

```
## [1] 2.035484
```

```r
# compute the mean with inbuilt function mean()
(mean(x))
```

```
## [1] 5.002634
```

```r
# check if the results of the inbuilt function mean() and the formula for calculating the mean are the s
if(round(sum(x)/length(x),5) == round(mean(x),5)) {
  ("same")
} else {
  ("diffrent")
}
```

```
## [1] "same"
```

```r
# compute the variance with inbuilt function var()
(var(x))
```

```
## [1] 4.143197
```

```r
# check if the results of the inbuilt function var() and the formula for calculating the variance are th
if(round((1/(length(x)-1) * sum((x-mean(x))**2)),5) == round(var(x),5)) {
  ("same")
} else {
  ("diffrent")
}
```

```
## [1] "same"
```

```r
# compute the standard variation with inbuilt function sd()
(sd(x))
```

```
## [1] 2.035484
```

```r
# check if the results of the inbuilt function sd() and the formula for calculating the variance are th
if(round(sqrt(1/(length(x)-1) * sum((x-mean(x))**2)),5) == round(sd(x),5)) {
  ("same")
} else {
  ("diffrent")
}
```

```
## [1] "same"
```

The conclusion of the above calculations shows, the higher the sample size n the smaller is the deviation between the sample mean $\overline{x}$ and the population mean μ. The same applies for the variance and standard deviation.

Task 5 - Interval Estimation

Calculate the confidence intervals for the mean and the variance.

1. Draw a reproducible sample of size $n = 30$ from a normal distribution with $\mu = 5$ and $\sigma^2 = 4$.

```r
# set seed to produce reproducible results
set.seed(10)
# draw random sample
(x <- rnorm(30, mean = 5, sd = sqrt(4)))
```

```
##  [1] 5.0374923 4.6314949 2.2573389 3.8016646 5.5890903 5.7795886 2.5838476
##  [8] 4.2726480 1.7466546 4.4870432 7.2035590 6.5115630 4.5235329 6.9748894
## [15] 6.4827803 5.1786945 3.0901123 4.6096992 6.8510425 5.9659570 3.8073787
## [22] 0.6294263 3.6502681 0.7618776 2.4696040 4.2526769 3.6248891 3.2556823
## [29] 4.7964780 4.4924389
```

2. Calculate a confidence interval for μ and σ^2 for $\alpha = 0.05$ (hence the confidence level is $1 - \alpha 1 = 0.95$). Inbuilt functions such as mean(), sd() and var() are allowed. looked up in sktiptum t(29;0.975) = 2.045

```r
# looked up the t-value in the skript t(29;0.975) = 2.045
# get t-value for n-1 = 29; 1-alpha = 0.975 from an inbuilt R function
(t <- qt(0.975,df=29))
```

```
## [1] 2.04523
```

```r
# lower bound mean
(low_bound_mean <- mean(x) - t * sd(x)/sqrt(length(x)))
```

```
## [1] 3.663266
```

```r
# upper bound mean
(up_bound_mean <- mean(x) + t * sd(x)/sqrt(length(x)))
```

```
## [1] 4.958028
```

```r
# get chi-value for n-1 = 29; 1-alpha = 0.975
(qchisq(0.975, df=29))
```

```
## [1] 45.72229
```

```r
# lower bound standard dev
(low_bound_var <- ((length(x)-1) * var(x)) / qchisq(0.975, df=29))
```

```
## [1] 1.906458
```

```r
# upper bound standard dev
(up_bound_var <- ((length(x)-1) * var(x)) / qchisq(0.025, df=29))
```

```
## [1] 5.431995
```

3. Determine if true parameters lie in the confidence interval.

```r
# check if the mean lies within the confidence interval of the mean
if((mean(x) > low_bound_mean) & (mean(x) < up_bound_mean)){
  ("the mean lies within the confidence interval")
} else{
  ("the mean lies outside the confidence interval")
}
```

```
## [1] "the mean lies within the confidence interval"
```

```r
# check if the variance lies within the confidence interval of the variance
if((var(x) > low_bound_var) & (var(x) < up_bound_var)){
  ("the variation lies within the confidence interval")
} else{
  ("the variation lies outside the confidence interval")
}
```

```
## [1] "the variation lies within the confidence interval"
```

Yes, in our case the true parameters lie within our confidence intervals. This is true for both the mean and the variance. The mean and the variance are in within the confidence intervals with a 5%probability of error.