Statistics with R - Exercise 3

Philipp Satlawa - h0640348

07/02/2021

This document contains the answered questions of exercise 3 for the course “Statistics with R”.

Task 1 – Statistical Tests

Load packages

```
library(nortest)
library(exactRankTests)
```

```
##  Package 'exactRankTests' is no longer under development.
##  Please consider using package 'coin' instead.
```

```
library(car)
```

```
## Loading required package: carData
```

```
library(ggplot2)
```

1. One Sample Test to test H0 - if the mean length of the sample is 950mm

```
# create vector with sample of size n = 18 from machine nr 1
(x <- c(870, 930, 932, 935, 938, 1045, 1050, 1052, 1055,
       970, 980, 1001, 1009, 1027, 1030, 1032, 1040, 1046))
```

```
##  [1]  870  930  932  935  938 1045 1050 1052 1055  970  980 1001 1009 1027 1030
## [16] 1032 1040 1046
```

```
# check if the sample has normal distribution
lillie.test(x = x)
```

```
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  x
## D = 0.20706, p-value = 0.03999
```

The Lilliefors normality tests p-value is (0.04) below 0.05, hence have to reject the H0 stating that the sample has a normal distribution. Thus we cannot use the one-sample t-test, since it requires a normal distribution of the data. We will use two non parametric tests instead.

Wilcoxon Signed Rank Test

```
# applying the Wilcoxon Signed Rank Test
wilcox.exact(x = x, alternative = "two.sided", mu = 950)
```

```
##
##  Exact Wilcoxon signed rank test
##
## data:  x
## V = 150, p-value = 0.003212
## alternative hypothesis: true mu is not equal to 950
```

The p-value of 0.00321 is lower than 0.05, hence we reject H0 stating that the mean length of the produced components is 950mm.

Sign-Test

```
# reference mean value
M0 <- 950
# calculate differences to reference
(diff_lenghts <- x - M0)
```

```
##  [1] -80 -20 -18 -15 -12  95 100 102 105  20  30  51  59  77  80  82  90  96
```

```
# create empty vector z
z <- numeric(length = length(diff_lenghts))
# set values of vector z to 0, if differences to reference are smaller than 0
z[diff_lenghts < 0] <- 0
# set 1 otherwise
z[diff_lenghts > 0] <- 1
# two sided binomial test with x = number of ones in vector z
binom.test(x = sum(z), n = length(diff_lenghts), p = 0.5, alternative = "two.sided")
```

```
##
##  Exact binomial test
##
## data:  sum(z) and length(diff_lenghts)
## number of successes = 13, number of trials = 18, p-value = 0.09625
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.4651980 0.9030508
## sample estimates:
## probability of success
##              0.7222222
```

As the p-value is larger than 0.05, there is no evidence in the data to reject H0, hence we can assume that the mean length is 950mm. However one requirement for the Sign-test is data symmetricity and cannot assume this for our data (please see 4. Informative plots - density plot).

2. Two Sample Test to test H0 - if the variance of both machines product lengths is the same

```
# create vector with sample of size n = 18 from machine nr 2
y <- c(1025, 1006, 1005, 1000, 990, 1013, 912, 1011, 909, 947)
# normality test for new machine
lillie.test(x = y)
```

```
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  y
## D = 0.27572, p-value = 0.03002
```

Since the p-value (0.03) is less than 0.05 the sample is not passing the test for normal distribution, similarly to the data of the first machine. Because the data is not normally distributed we cannot use the F-Test and we will instead use the Levene Test.

Levene Test

```
# combine the two vectors with the sample data into a dataframe
data_frame <- data.frame(len = c(x, y),
                         group = c(rep("old machine", length(x)), rep("new machine", length(y))),
                         rank = rank(c(x, y)))

# perform Levene Test
leveneTest(len ~ group, data = data_frame, center = "mean")
```

```
## Levene's Test for Homogeneity of Variance (center = "mean")
##       Df F value Pr(>F)
## group  1   1.317 0.2616
##       26
```

The p-value of the Levene Test is 0.262 (> 0.05), thus we cannot reject H0 stating that the varinaces of the two samples are equal.

3. Two Sample Test to test H0 - if the mean length is the same with the new machine

Since we know that the data is not normally distributed, we need to perform a non-parametric test in our case the Wilcoxon Rank Sum Test to test for the H0 on the two samples.

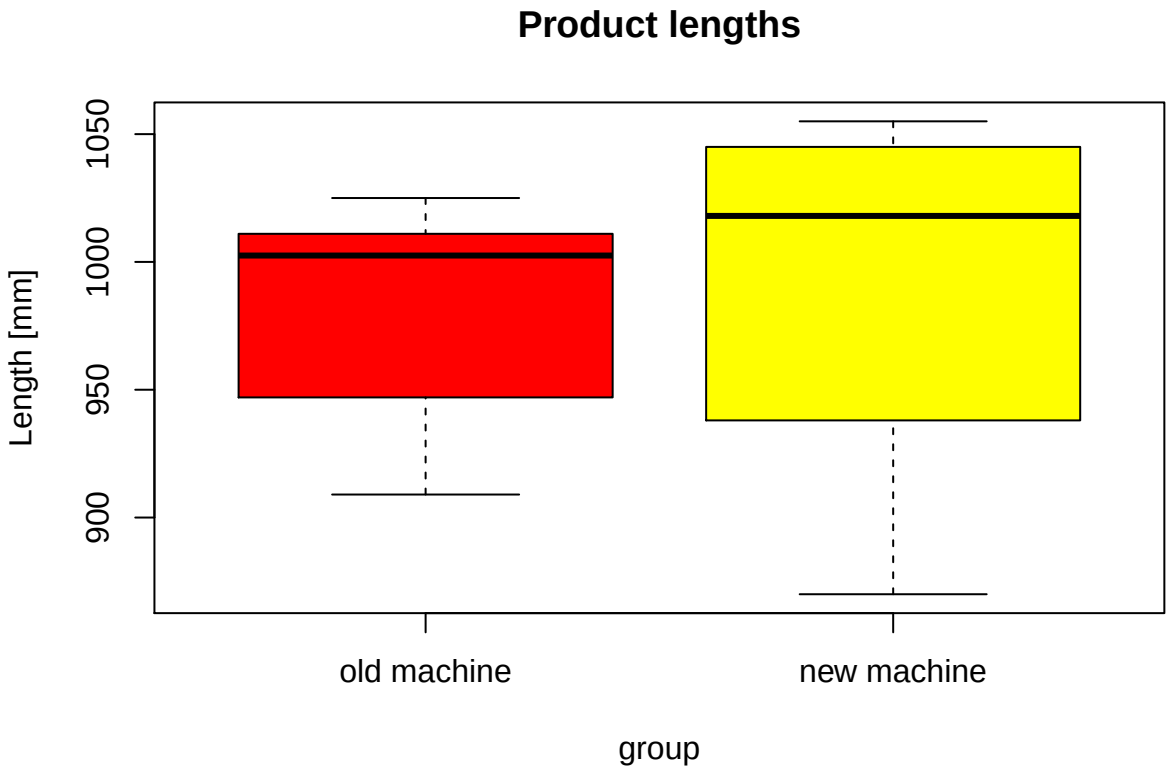
```
# Wilcoxon Rank Sum Test, 2-sided and equal variances
wilcox.test(len ~ group, data = data_frame, alternative = "two.sided", exact = TRUE)
```

```
##
##  Wilcoxon rank sum test
##
## data:  len by group
## W = 64, p-value = 0.2259
## alternative hypothesis: true location shift is not equal to 0
```

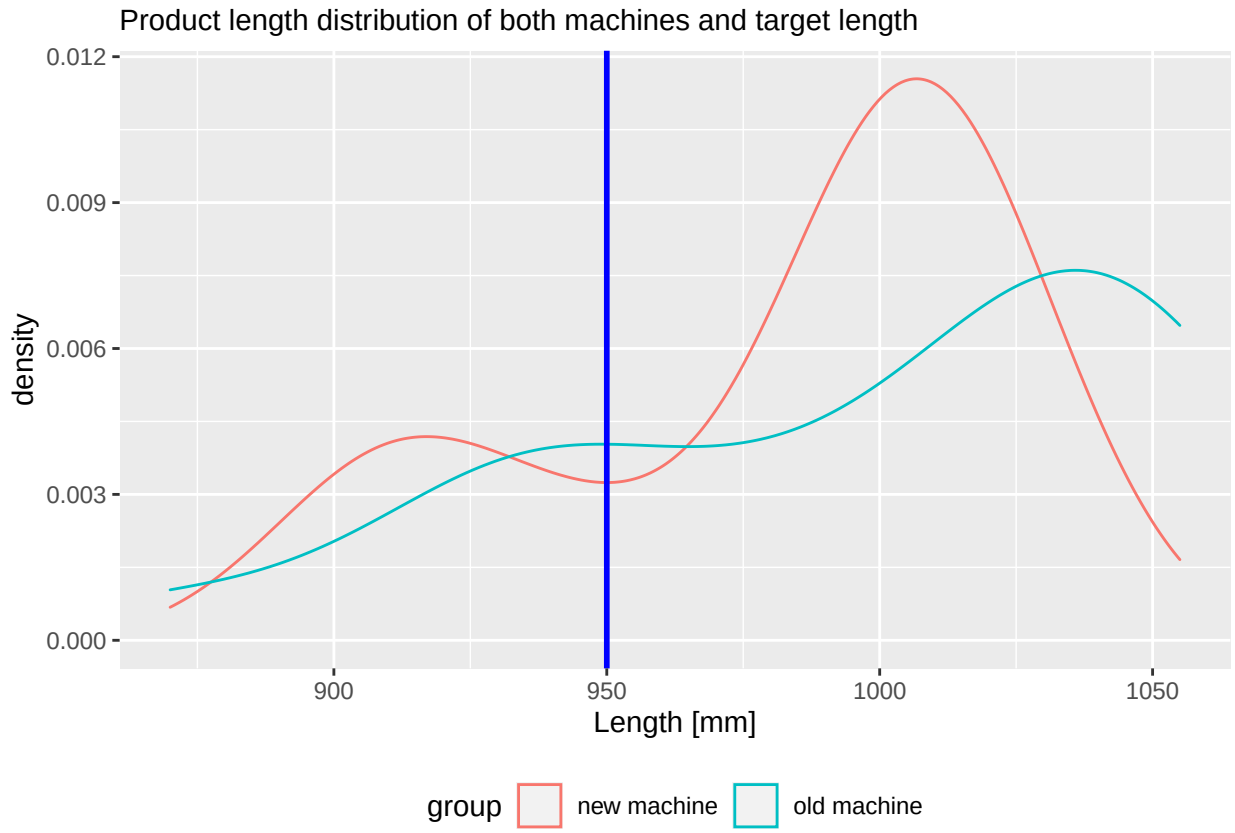
We obtain a p-value of 0.226, thus we do not reject H0 stating that the equal mean length of both machines does not differ.

4. Informative plots

```
# boxplot of product lengths produced by the two machines
boxplot(len ~ group, data = data_frame,
        col = heat.colors(2),
        main = "Product lengths",
        names = c("old machine", "new machine"), ylab = "Length [mm]")
```




```
# density plot of product lengths produced by the two machines with the target length = 950mm
ggplot(data_frame, aes(x = len)) + geom_density(aes(color = group)) + geom_vline(aes(xintercept = M0),
```



Task 2 - Functions

1. Create a function with the name quad_equ()

```
quad_equ <- function(coef){
  x1 <- -(coef[1]/2) - sqrt((coef[1]/2)^2-coef[2])
  x2 <- -(coef[1]/2) + sqrt((coef[1]/2)^2-coef[2])

  return(c(x1,x2))
}
coef <- c(3,-4)
quad_equ(coef)
```

```
## [1] -4  1
```

2. Extend function - check if the passed argument coef gives real solutions

```
quad_equ <- function(coef){
  if((coef[1]/2)^2-coef[2] >= 0){
```

```
    x1 <- -(coef[1]/2) - sqrt((coef[1]/2)^2-coef[2])
    x2 <- -(coef[1]/2) + sqrt((coef[1]/2)^2-coef[2])

    return(c(x1,x2))
  } else{
    print("The supplied vector x has no real solution.")
    #stop("The supplied vector x has no real solution.")
  }
}
coef <- c(3,-4)
quad_equ(coef)
```

```
## [1] -4  1
```

```
coef <- c(-2, 2)
quad_equ(coef)
```

```
## [1] "The supplied vector x has no real solution."
```

3. Extend function - check if the passed argument is valid

```
quad_equ <- function(coef){
  if(length(coef)==2){
    if((coef[1]/2)^2-coef[2] >= 0){
      x1 <- -(coef[1]/2) - sqrt((coef[1]/2)^2-coef[2])
      x2 <- -(coef[1]/2) + sqrt((coef[1]/2)^2-coef[2])

      return(c(x1,x2))
    } else{
      print("The supplied vector x has no real solution.")
    }
  } else{
    print("The supplied vector x has the wrong length.
          The input vector has to have length 2.
          Please enter a valid vector.")
  }
}
# test if the function is working correctly with a correct input
coef <- c(3,-4)
quad_equ(coef)
```

```
## [1] -4  1
```

```
# test if the function invokes an error with an incorrect input
coef <- c(3,-4, 9)
quad_equ(coef)
```

```
## [1] "The supplied vector x has the wrong length. \n         The input vector has to have length 2. \n
```

Task 3 - Graphics

Creation of a statistical graph

Load libraries

```
library(data.table)
```

1. Create a data for the construction of a plot that mimics the template "Plot_9.png"

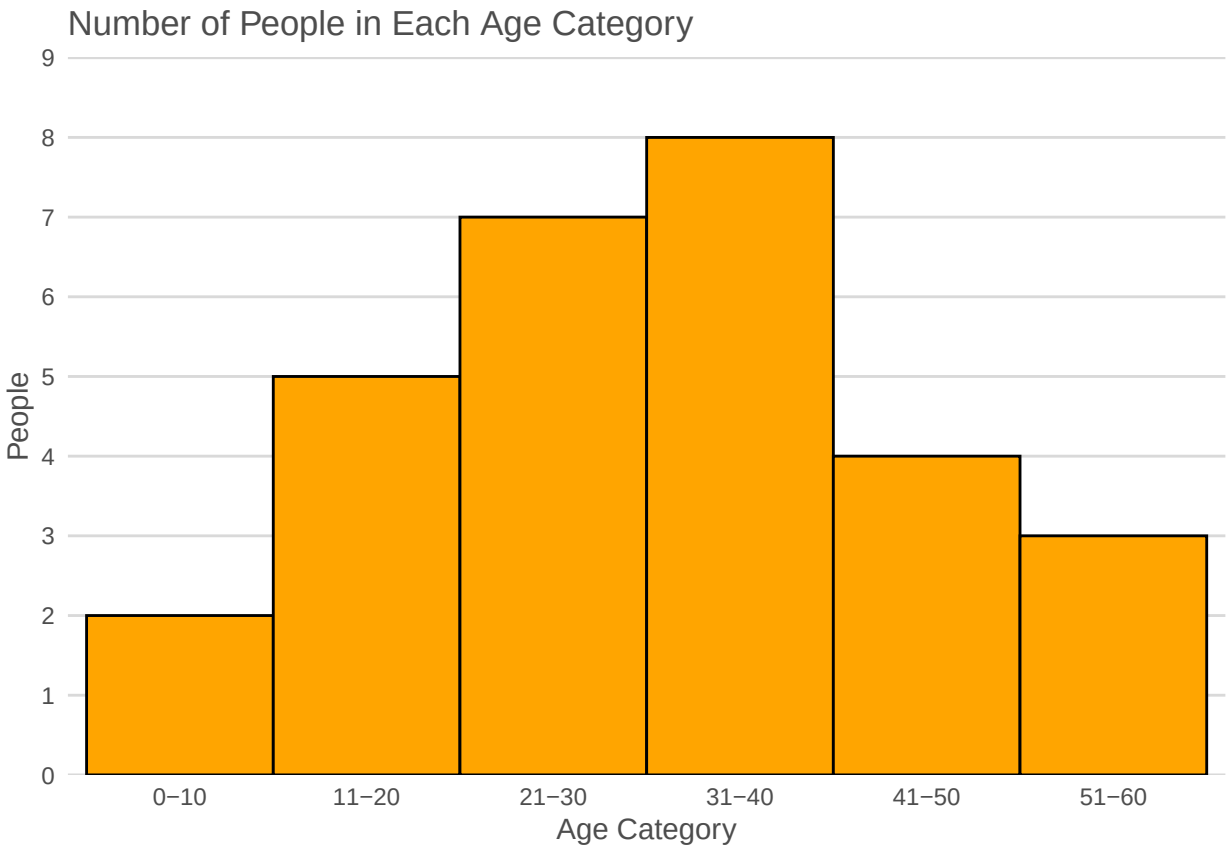
```
# create data
dt <- data.table(id = c(1:29), age = c(25, 21, 5, 15, 47, 33, 39, 56, 3, 45, 31, 28, 44, 15, 13, 22, 40
```

2. combine the feature "age" into age intervals

```
# create vector containing the cutting points
cut_points <- c(0,11,21,31,41,51,61)
# create vector containing the labels
labels <- c("0-10","11-20", "21-30", "31-40", "41-50", "51-60")
# categorize the feature "age" into the created labels
age_cat <- cut(dt$age, breaks=cut_points, include.lowest=TRUE, right=FALSE, labels=labels)
# add age_cat to data.table
dt[, ("age_category") := age_cat]
```

3. Create plot

```
# Create plot
p <- ggplot(data = dt, mapping = aes(x=age_category)) +
  geom_bar(stat="count", fill="orange", color="black", width = 1) +
  labs(title = "Number of People in Each Age Category", x = "Age Category", y = "People") +
  theme(panel.background = element_rect(fill = "white"),
        panel.grid.major = element_line(colour = "grey85"),
        panel.grid.major.x = element_blank(),
        axis.ticks = element_blank(),
        axis.title = element_text(colour = "grey30"),
        plot.title = element_text(colour = "grey30")) +
  scale_y_continuous(breaks = seq(0, 9, 1), expand=c(0,0), limits=c(0,9))
p
```


Number of People in Each Age Category
People
0
1
2
3
4
5
6
7
8
9
0−10
11−20
21−30
31−40
41−50
51−60
Age Category