Hausarbeit2_StatSoftwR

Renate Schmeidler

20 6 2021

Aufgabe 2

Bei dem angegebenen Code liegt eine Endlos-Schleife vor:

- Der Wert von i wird zu Beginn auf 1 gesetzt und bei jedem Schleifendurchlauf um 1 erhöht.

- Damit ist die Bedingung “i > 0” stets erfüllt, und die Schleife bricht nicht ab, sondern wird theoretisch (d. h. solange kein Eingriff von außen o.ä. erfolgt) unendlich oft ausgeführt.

Aufgabe 3

a)

```r
set.seed(20210614)
outcomes <- c("KOPF", "ZAHL")

# This function simulates tossing a coin four times and counting the number of Heads ("KOPF") that appe

count_head_tossing_coin_four_times <- function() {

  number_head <- 0
  for (i in 1:4) {
    coin <- sample(outcomes, 1, prob = c(.5, .5))
    if (coin == "KOPF") {
      number_head <- number_head + 1
    }
  }
  return(number_head)
}
```

b)

```r
# This function
n_repeat_counting_head <- function(n) {

  if (!is.numeric(n))
    stop("Anzahl der Wiederholungen muss natürliche Zahl sein.")
  if (n < 0)
    stop("Negative Anzahl an Wiederholungen nicht sinnvoll.")

  outcome <- vector(mode = "double", length = n)
  for (i in 1:n) {
   outcome[i] <- count_head_tossing_coin_four_times()
  }

  return(outcome)
}


n_times <- 10000
outcome <- n_repeat_counting_head(n_times)

four_head <- which(outcome == 4)

number_four_head <- length(four_head)
number_four_head
```

```
## [1] 644
```

```r
probability_four_head <- number_four_head / n_times
probability_four_head
```

```
## [1] 0.0644
```

c)

```r
gain <- probability_four_head * (-50) + (1 - probability_four_head) * 1
gain
```

```
## [1] -2.2844
```

d)

```r
n_until_four_head <- function() {

  k <- 0
  number_head <- -1

  while (number_head < 4) {
    k <- k + 1
    # print(k)
    number_head <- count_head_tossing_coin_four_times()
  }
  return(k)
}
```

e)

```r
x <- vector(mode = "numeric", length = 100)
for (i in seq_along(x)) {
  x[i] <- n_until_four_head()
}
x
```

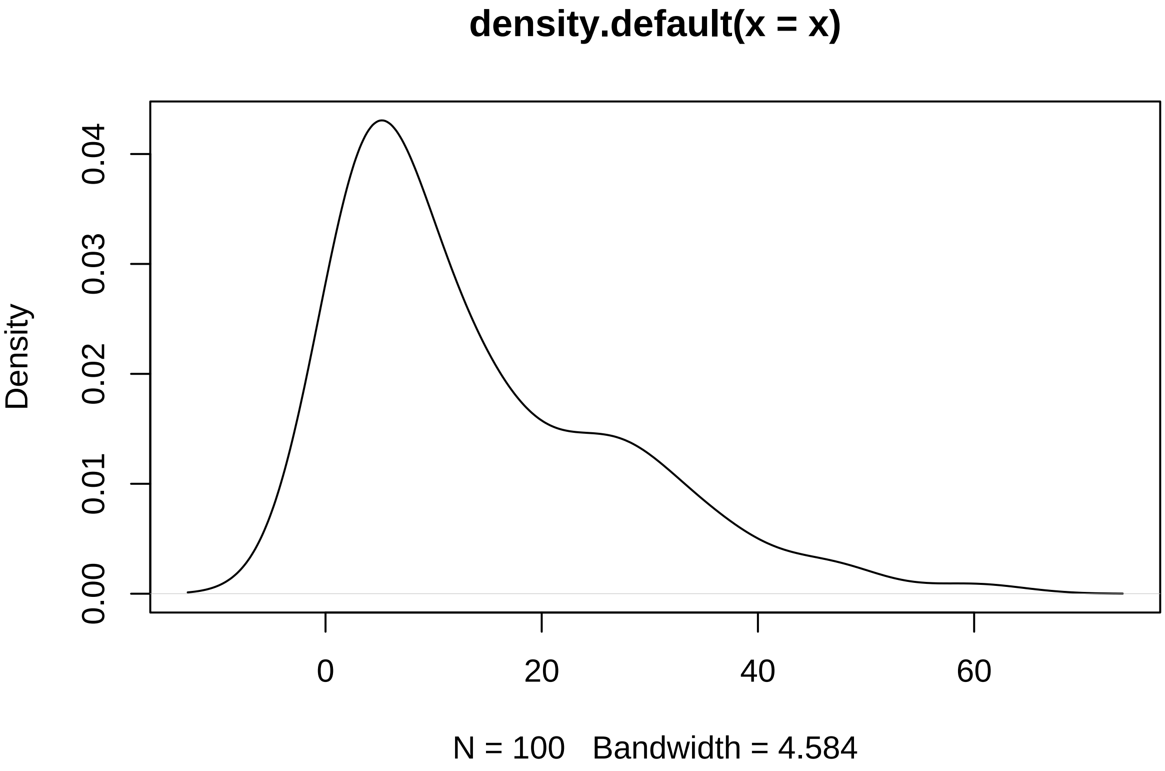
```
##   [1]  3 38 10 22  1 60 12  2 24 30 12  1  4  3  5 18  2  4  7  1 13  6  9  4 27
##  [26]  2 22 35  5 24 31 15 20  5  4 27  6 18  6 27 26  2  6  4  4  5 27 32  4 48
##  [51]  3 15  5 19 27 11  5  4 47 21 10  4 14  6 45  9 27  4 31  2  4 39 14  1 12
##  [76]  8  3 16  7  7 31 36 26 18  1 12 15  1 14 10  9  5  6 11  5  7 36  2  2 14
```

```
mean(x)
```

```
## [1] 14.04
```

f)

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

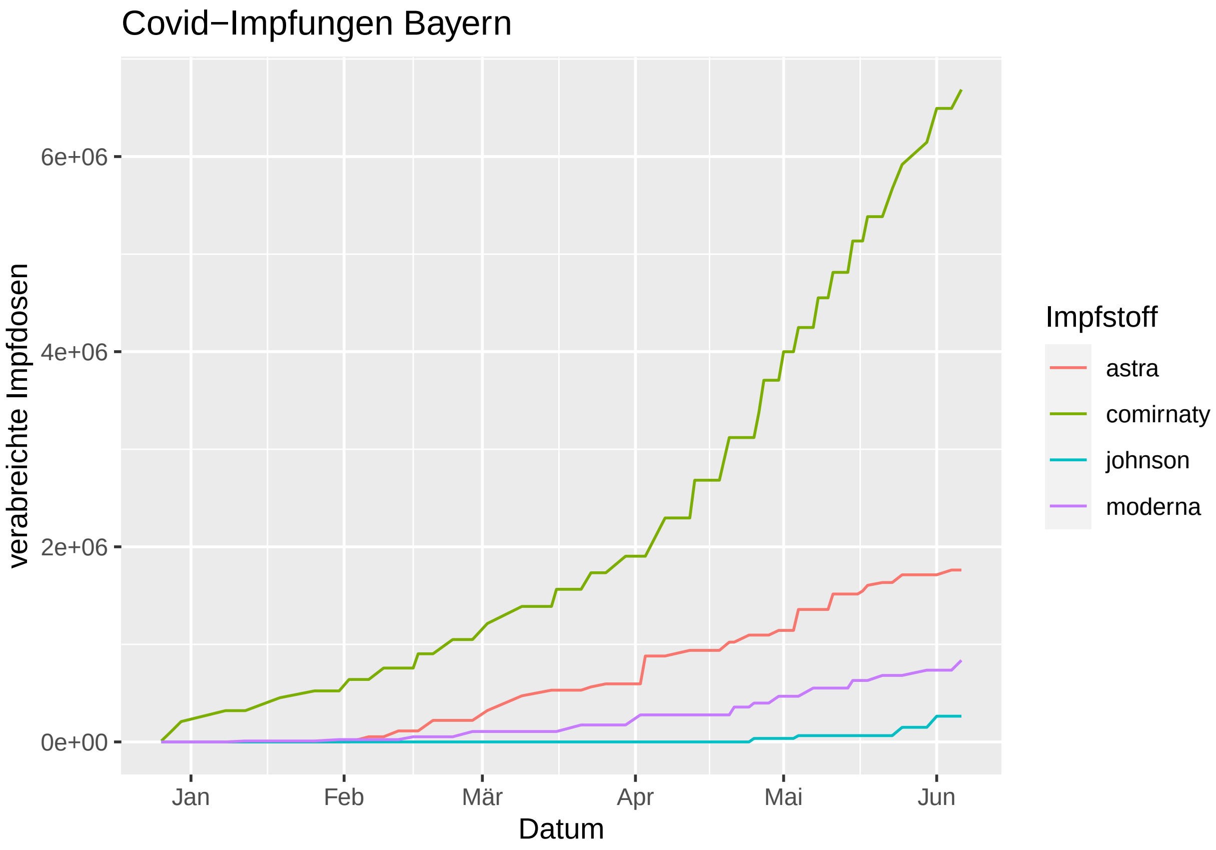


Aufgabe 4

```
library(ggplot2)
vaccs <- readRDS("vaccs.Rds")
```

a)

```
vaccs_bayern <- subset(vaccs, region == "DE-BY")
ggplot(data = vaccs) +
  geom_line(vaccs_bayern, mapping = aes(x = date, y = dosen, color = impfstoff)) +
  labs(title = "Covid-Impfungen Bayern", x = "Datum", y = "verabreichte Impfdosen", color = "Impfstoff"
```



b)

c)

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

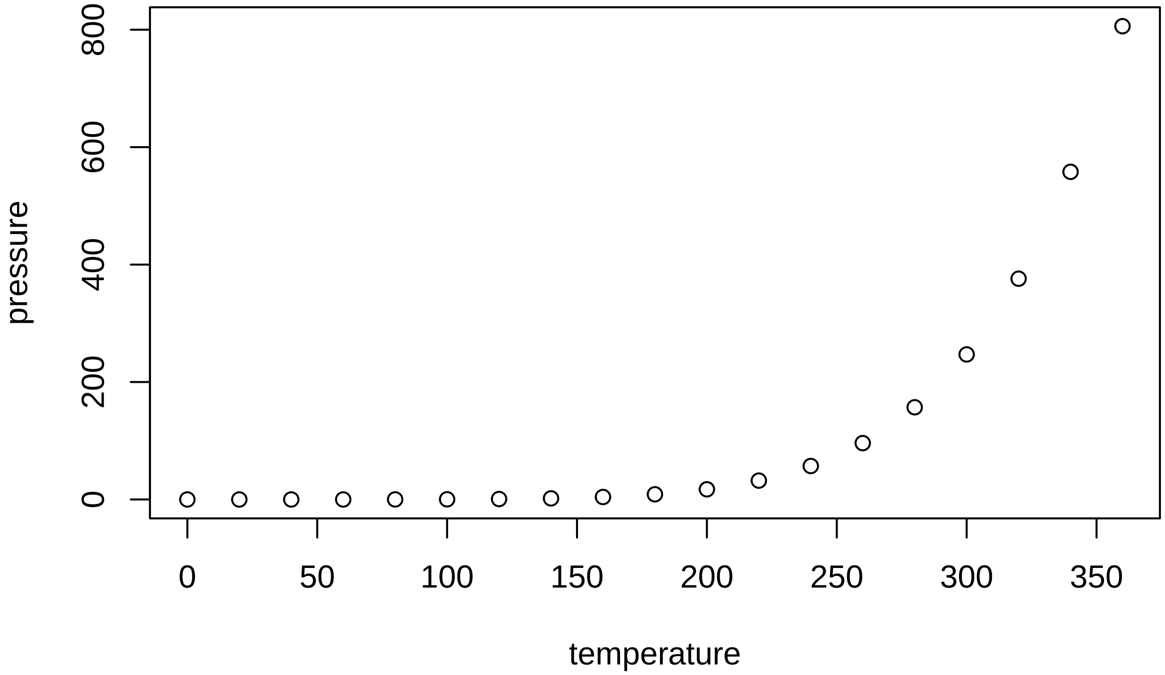
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.