ds1-practice

Marcus Birkenkrahe

November 1, 2024

File: $7_{\textbf{subsettinglab}}$

Link: tinyurl.com/sub-lab

TODO README

- Put your name in the header next to "(pledged)"
- Practice file for the lecture on "Subsetting and extracting indices in R" (GitHub)
- All content from that lecture is also available on YouTube
- Create, execute and debug R code blocks as needed
- Upload the completed file as a class assignment to Canvas
- You find the solutions in the PDF repository later (GitHub)

TODO 10 SUBSETTING QUESTIONS

1. Store a vector of these 10 values in `foo`: `7 5 6 1 2 10 8 3 8 2`, print it and show that it has 10 elements using a function.

```
c(7, 5, 6, 1, 2, 10, 8, 3, 8, 2) -> foo
foo
length(foo)
```

```
 [1]  7  5  6  1  2 10  8  3  8  2
[1] 10
```

(a) Extract the fifth through the seventh element of `foo` and add `5` to

these elements to get the output: `7 15 13`

```
foo[5:7] + 5 # vectorization!
```

```
[1]  7 15 13
```

2. Create a logical flag vector of those elements of `foo` that are greater than `5` and store it in `idx`.

```
foo > 5 -> idx  # vectorization
idx
```

```
[1]  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE
```

3. Extract the values of `foo` that are greater than `5` using `idx`.

```
foo[idx]  # using the logical flag vector to subset
```

```
[1]  7  6 10  8  8
```

4. `Which` vector indices of `foo` correspond to the values greater or equal than `5`? Store the answer in `idx2` - Output: `1 2 3 6 7 9`

```
which(foo >= 5) -> idx2
idx2 # index values of elements greater or equal than 5
```

```
[1] 1 2 3 6 7 9
```

5. Use `idx2` to extract the values of `foo` that are greater or equal than `5`.

```
foo[idx2]
```

```
[1]  7  5  6 10  8  8
```

6. Extract every 2nd element of `foo` using `seq`. Output: `7 6 2 8 8`.

```
foo
seq(from=1,to=length(foo),by=2) -> second # indices of every 2nd element
second
foo[second]  # values for those indices
```

```
 [1]  7  5  6  1  2 10  8  3  8  2
[1] 1 3 5 7 9
[1] 7 6 2 8 8
```

7. Remove the elements with even indices from `foo` and store them in `bar`, and store the elements with odd indices from `foo` in `baz`.

```
foo
seq(from=2,to=length(foo),by=2) -> even
even # even indices
foo[-even] -> bar
bar
foo[even] -> baz
baz
```

```
 [1]  7  5  6  1  2 10  8  3  8  2
[1]  2  4  6  8 10
[1] 7 6 2 8 8
[1]  5  1 10  3  2
```

8. Make all elements of `baz` negative.

```
baz * (-1) -> baz
baz
```

```
[1]  -5  -1 -10  -3  -2
```

9. Append `baz` to `bar` and store the result in `foo2`.

```
bar
baz
c(bar,baz) -> foo2
foo2
```

```
[1] 7 6 2 8 8
[1]  -5  -1 -10  -3  -2
 [1]   7   6   2   8   8  -5  -1 -10  -3  -2
```

TODO 10 QUESTIONS ABOUT THE NILE

1. Print the `length` of the data set `Nile` using an R function.

```
length(Nile)
```

```
[1] 100
```

2. Retrieve the second to fifth value of the `Nile` data set - there are (at least) two ways to do this", with the concatenation function or with the colon operator.

```
Nile[c(2,3,4,5)]  # with concatenation function
Nile[2:5] # with colon operator
```

```
[1] 1160  963 1210 1160
[1] 1160  963 1210 1160
```

3. Which data science (not R) question does the last output answer? Write a full sentence.

> "What was the average flow of the river Nile in the second through fifth years of observation?"

4. Extract the year that corresponds to the last value of `Nile`. Remember that the years are stored in `time(Nile)`.

```
head(time(Nile))
time(Nile)[length(Nile)]
```

```
[1] 1871 1872 1873 1874 1875 1876
[1] 1970
```

5. What was the average Nile flow in 1967? Tip: use a logical operator, and store `time(Nile)` in `t` for the remaining questions.

```
time(Nile) == 1967 -> sixty_seven # observation in 1967
sixty_seven
Nile[sixty_seven] # average flow value in that year
```

```
Time Series:
Start = 1871
End = 1970
Frequency = 1
  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
 [17] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
 [33] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
 [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
 [65] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
 [81] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
 [97]  TRUE FALSE FALSE FALSE
[1] 919
```

6. What is the index of the third-to-last element of `Nile`? Use the `which` function to answer this question.

```
Nile[length(Nile)-3] -> val # third-to-last value of Nile
val
idx <- which(Nile==Nile[length(Nile)-3]) # its index
idx
```

```
[1] 919
[1] 97
```

7. How many values of `Nile` are larger than the third-to-last value?

```
Nile > val # elements larger than val (logical)
sum(Nile > val) # NOT ‘length‘ (counts T and F)
```

```
Time Series:
Start = 1871
End = 1970
Frequency = 1
  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRU
 [17]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALS
 [33]  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALS
 [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALS
 [65]  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALS
 [81] FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE FALS
 [97] FALSE FALSE FALSE FALSE
[1] 43
```

8. How much water flowed down the Nile between 1871 and 1970?

```
sum(Nile)
paste("Nile flow 1871-1970:", sum(Nile), "million cubic metres.")
```

```
[1] 91935
[1] "Nile flow 1871-1970: 91935 million cubic metres."
```

9. In which year was the Nile at its lowest level? Use the `min` function for this task and store the result in ~tmin

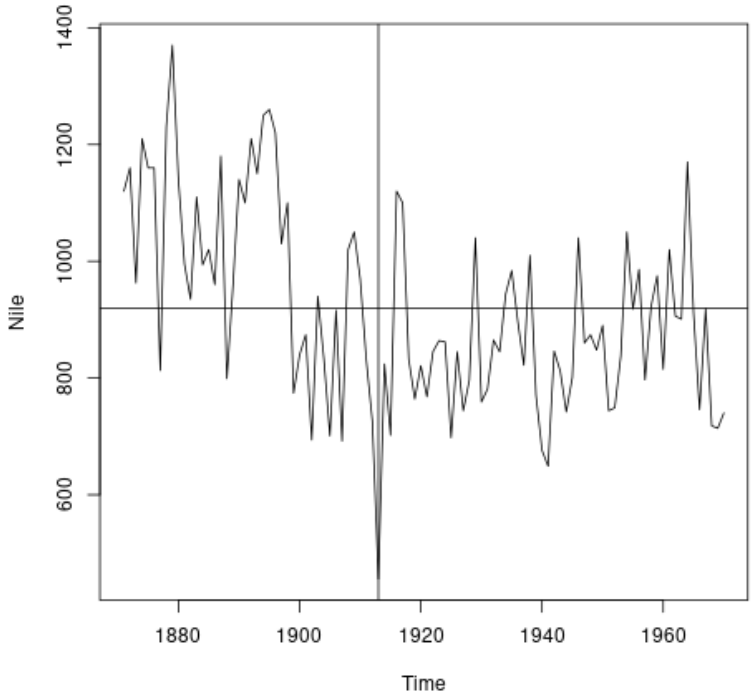
```
min(Nile) # minimum flow value
which(Nile == min(Nile)) -> idx  # get the corresponding index
time(Nile)[idx] -> tmin # extract the corresponding time
tmin
```

```
[1] 456
[1] 1913
```

10. Make a line plot of all observations in the data set `Nile` using `plot`, mark the year of the lowest level of the Nile with a vertical line, and the average flow through the Nile with a horizontal line. The result is stored in `nile.png`.

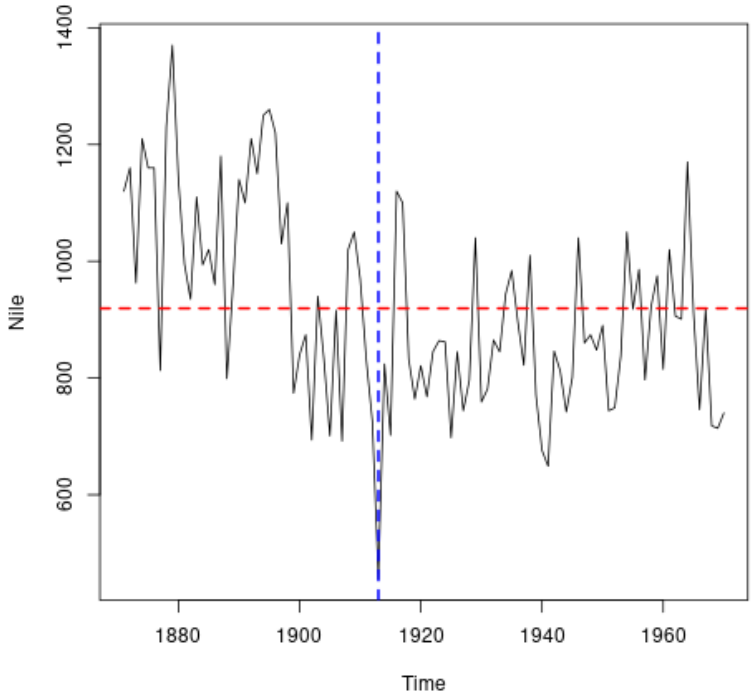
    Tip: You can draw a vertical line at point `x` with `abline(v=x)`, and a horizontal line at point `y` with `abline(h=y)`.

```
plot(Nile)
abline(v=tmin)
abline(h=mean(Nile))
```



Tip: You can change the appearance of lines with the parameters `col`, `lty`, `lwd`. E.g. `col="red, lty=2, lwd=2` for a red, dashed, thick line.

```
plot(Nile)
abline(v=tmin, col="blue",lwd=2,lty=2)
abline(h=mean(Nile), col="red",lwd=2,lty=2)
```


Nile
1400
1200
1000
800
600
1880
1900
1920
1940
1960
Time