Midterm solutions

Oscar

2022-10-28

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.2.1
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Midterm solutions. Questions 2a and 2b

```
df <- tibble(Y0 = c(1,2,0,3,1,5,2,1,7,0),
Y1 = c(0, -3, 1,2,1,-8,2,-1,4,0),
D = c(1,0,1,1,1,1,0,1,1,0))
df
```

```
## # A tibble: 10 x 3
##       Y0    Y1     D
##    <dbl> <dbl> <dbl>
##  1     1     0     1
##  2     2    -3     0
##  3     0     1     1
##  4     3     2     1
##  5     1     1     1
##  6     5    -8     1
##  7     2     2     0
##  8     1    -1     1
##  9     7     4     1
## 10     0     0     0
```

```
# using case_when
df_1 <- df |>
  mutate(Y = case_when(D == 1 ~ Y1,
                       TRUE ~ Y0))
df_1
```

```
## # A tibble: 10 x 4
##       Y0    Y1     D     Y
##    <dbl> <dbl> <dbl> <dbl>
##  1     1     0     1     0
##  2     2    -3     0     2
##  3     0     1     1     1
##  4     3     2     1     2
##  5     1     1     1     1
##  6     5    -8     1    -8
##  7     2     2     0     2
##  8     1    -1     1    -1
##  9     7     4     1     4
## 10     0     0     0     0
```

```
# alternatively
df_2 <- df |>
  mutate(Y = Y1*D + Y0*(1 - D))

df_2
```

```
## # A tibble: 10 x 4
##       Y0    Y1     D     Y
##    <dbl> <dbl> <dbl> <dbl>
##  1     1     0     1     0
##  2     2    -3     0     2
##  3     0     1     1     1
##  4     3     2     1     2
##  5     1     1     1     1
##  6     5    -8     1    -8
##  7     2     2     0     2
##  8     1    -1     1    -1
##  9     7     4     1     4
## 10     0     0     0     0
```

For loops

Warm up

Recall, for-loops are an iterator that help us repeat tasks while changing inputs. The most common structure for your code will look like the following code. Complete and run the code.

```
# what are you iterating over? The vector from -10:10
items_to_iterate_over <- c(-10:10)
items_to_iterate_over
```

```
##  [1] -10  -9  -8  -7  -6  -5  -4  -3  -2  -1   0   1   2   3   4   5   6   7   8
## [20]   9  10
```

```
# pre-allocate the results
out <- rep(0, length(items_to_iterate_over))

# write the iteration statement --
```

```
# we'll use indices so we can store the output easily
# i is a temporary container / object
for (i in seq_along(items_to_iterate_over)) {

# do something
# we capture the median of three random numbers from normal distributions various means
out[[i]] <- median(rnorm(n = 3, mean = items_to_iterate_over[[i]]))
}

out
```

```
##  [1] -9.8770044 -9.9186576 -6.4719213 -7.2343257 -7.7578847 -5.5333774
##  [7] -4.1608683 -3.7206009 -3.9046424 -1.2741851  0.5829218  0.6437747
## [13]  1.9483228  2.7466135  4.4461938  5.5088997  6.0119127  7.2494193
## [19]  8.9737850  9.3904345  9.0877838
```

I. Writing for-loops

```
x <- c(5, 10, 15, 20, 25000)

for (number in x){
  print(number)
}
```

```
## [1] 5
## [1] 10
## [1] 15
## [1] 20
## [1] 25000
```

```
x <- c(5, 10, 15, 20, 250000)

for (i in seq_along(x) ){
  print(x[[i]])
}
```

```
## [1] 5
## [1] 10
## [1] 15
## [1] 20
## [1] 250000
```

```
set.seed(60615)

random <- rnorm(5)
random
```

```
## [1]  1.1778038  0.6995375  1.4527076 -1.0487119 -0.2286733
```

```r
set.seed(60615)

sd(rnorm(5))
```

```
## [1] 1.03638
```

```r
sd(rnorm(10))
```

```
## [1] 1.232284
```

```r
sd(rnorm(15))
```

```
## [1] 1.133005
```

```r
sd(rnorm(20))
```

```
## [1] 0.9436211
```

```r
sd(rnorm(25000))
```

```
## [1] 1.007314
```

```r
set.seed(60615)

sd1 <- sd(rnorm(5))
sd2 <- sd(rnorm(10))
sd3 <- sd(rnorm(15))
sd4 <- sd(rnorm(20))
sd5 <- sd(rnorm(25000))

sds <- c(sd1, sd2, sd3, sd4, sd5)
sds
```

```
## [1] 1.0363803 1.2322838 1.1330053 0.9436211 1.0073138
```

```r
set.seed(60615)

x <- c(5, 10, 15, 20, 25000)

# replace the ... with the relevant code
for (i in seq_along(x) ){
  print(sd(rnorm(x[[i]])))
}
```

```
## [1] 1.03638
## [1] 1.232284
## [1] 1.133005
## [1] 0.9436211
## [1] 1.007314
```

```r
set.seed(60615)

for (i in seq_along(x) ){
  n <- x[[i]]
  print(sd(rnorm(n, mean = 4)))
}
```

```
## [1] 1.03638
## [1] 1.232284
## [1] 1.133005
## [1] 0.9436211
## [1] 1.007314
```

```r
set.seed(60615)
for (i in seq_along(x) ){
  n <- x[[i]]
  print(sd(rnorm(n, sd = 4)))
}
```

```
## [1] 4.145521
## [1] 4.929135
## [1] 4.532021
## [1] 3.774484
## [1] 4.029255
```