r_intro PS 2 answers

Wywin

2022-11-16

Practice Summary Statistics with atomic vector

First thing first, you definitely need `tidyverse` pkg for this exercise. Don't forgot to load it at the top of your answer `rmd` file.

Let's go back to the previous exercise on implementation of the sampling distribution of the means.

2. Construct the list of 100 means values from the `Sepal.Length` and each mean value should construct from 10 sample sizes. (sample size = 10, replication 100 times)

load iris dataset

```
df <- iris
```

```
set.seed(2345)
replicate(n = 100, expr = mean(sample(df$Sepal.Length, size = 10,
                                      replace=TRUE)))
```

```
##   [1] 5.86 5.62 5.66 5.85 5.86 6.13 6.09 5.88 6.07 5.64 5.94 5.68 5.42 5.82 6.26
##  [16] 5.91 6.25 5.91 6.03 5.62 5.45 6.02 5.95 5.88 5.77 5.80 5.54 5.49 6.03 5.76
##  [31] 5.75 5.71 6.23 5.34 5.77 5.75 6.13 6.12 5.83 5.41 5.73 5.89 5.81 5.95 5.95
##  [46] 5.76 5.76 5.61 5.95 5.90 6.23 6.17 5.74 6.33 5.90 5.54 5.89 5.70 5.86 5.89
##  [61] 6.23 5.78 5.70 5.79 5.87 6.04 6.12 6.25 5.78 5.95 5.60 6.03 5.56 5.79 6.10
##  [76] 5.66 5.44 6.46 6.00 5.56 5.77 5.68 5.79 6.28 6.03 5.75 5.71 5.89 5.96 5.77
##  [91] 6.25 5.65 5.53 5.40 5.79 5.72 5.32 6.10 6.03 5.96
```

2. Assigned the result (vector with 100 numbers) as vector name called `means_list` and .

```
means_list <- replicate(n = 100, expr = mean(sample(df$Sepal.Length, size = 10,
                                      replace=TRUE)))
```

3. Construct the another vector called `test` and assigned `1` hundred time in that vector. Use `rep()` function for generation of same value for 100 times.

```
test <- rep(1, 100)
```

4. Then, using `cbind` function to combined those two vectors" `test` and `means_list`, and treat the result as `data.frame` and assigned to the object name `means_list_1`. You can try with below demo code.

```
means_list_1 <- data.frame(cbind(test, means_list))
```

Repeat the same process from numbers 1 to 4, but using different sample size and replication number this time.

- for `means_list_2`, using a sample size 30 and replication time 200.

```
means_list <- replicate(n = 200, expr = mean(sample(df$Sepal.Length, size = 30,
                                    replace=TRUE)))
test <- rep (2, 200)

means_list_2 <- data.frame(cbind(test, means_list))
```

- for `means_list_3`, using a sample size 50 and replication time 1,000.

```
means_list <- replicate(n = 1000, expr = mean(sample(df$Sepal.Length, size = 50,
                                    replace=TRUE)))
test <- rep (3, 1000)

means_list_3 <- data.frame(cbind(test, means_list))
```

- for `means_list_4`, using a sample size 50 and replication time 3,000.

```
means_list <- replicate(n = 3000, expr = mean(sample(df$Sepal.Length, size = 50,
                                    replace=TRUE)))
test <- rep(4,3000)

means_list_4 <- data.frame(cbind(test, means_list))
```

- for `means_list_5`, using a sample size 50 and replication time 10,000.

```
mean_list <- replicate(n = 10000, expr = mean(sample(df$Sepal.Length, size = 50,
                                    replace=TRUE)))
test <- rep(5,10000)

means_list_5 <- data.frame(cbind(test, means_list))
```

```
## Warning in cbind(test, means_list): number of rows of result is not a multiple
## of vector length (arg 2)
```

This time, we are going to create the list to store all those `means_list_x` (where `x 1:5`) and assigned that list as `means_seris`. And, perform the following function from that list.

```
means_seris <- list(means_list_1, means_list_2, means_list_3, means_list_4, means_list_5)
```

1. calculate the mean of `means_list_x` (where `x 1:5`) from that list (using the command related to filtering list from lecture 2.

For example, if we use the following command, we can get the first dataframe. You work is to calculate the mean value of column `mean_list`.

```r
mean1 <- mean(means_seris[[1]][["means_list"]])
mean1
```

```
## [1] 5.7984
```

```r
mean2 <- mean(means_seris[[2]][["means_list"]])
mean2
```

```
## [1] 5.85175
```

```r
mean3 <- mean(means_seris[[3]][["means_list"]])
mean3
```

```
## [1] 5.837384
```

```r
mean4 <- mean(means_seris[[4]][["means_list"]])
mean4
```

```
## [1] 5.845169
```

```r
mean5 <- mean(means_seris[[5]][["means_list"]])
mean5
```

```
## [1] 5.844854
```

2. Using `bind_rows()` function to combined all dataset from `means_seris` and assigned into object called `df_means_combined`. Please make sure that your result dataset should have `16100` observations and 2 variables; `test` and `mean_list`.

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.2.2
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
```

```
## Warning: package 'ggplot2' was built under R version 4.2.2
```

```
## Warning: package 'tidyr' was built under R version 4.2.2
```

```
## Warning: package 'readr' was built under R version 4.2.2
```

```
## Warning: package 'purrr' was built under R version 4.2.2
```

```
## Warning: package 'dplyr' was built under R version 4.2.2
```

```
## Warning: package 'forcats' was built under R version 4.2.2
```

```
## -- Conflicts --------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
df_means_combined <- bind_rows(means_seris)
```

3. Finally, calculate the mean value for each group of `test` using following example. In this exercise, we are going to use `group_by` function from the `tidyverse` packge and `%>%` operator (pipe operator).

```
library(tidyverse)

df_means_combined %>%
  group_by(test) %>%
  summarise(mean = mean(means_list))
```

```
## # A tibble: 5 x 2
##    test  mean
##   <dbl> <dbl>
## 1     1  5.80
## 2     2  5.85
## 3     3  5.84
## 4     4  5.85
## 5     5  5.84
```