BIOST 2039

Lab 1

Arvon Clemons

due: September 4, 2019

Contents

Instructions

Please **change the author** of this file to your name instead of mine in the header above.

Type your answer below each question. For questions that require the use of R, you can type both plain text (like this) and R commands by including an R chunk (see example below).

```
# This is an R chunk. Everything in this grey box is R code, not regular text.
print("Hello, world!")
```

```
## [1] "Hello, world!"
```

You can run R chunks individually by clicking the green arrow in the top right corner of an R chunk.

When typing plain text in an R Markdown, leave two spaces at the end of the line when you want a paragraph break. Otherwise, paragraphs are typset together, like this! For additional help with R Markdown, check out this reference sheet https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf.

When you are finished, click `Knit` or `Knit to PDF` at the top of the Rmd editor. This will create an PDF file of that same name. Please submit the **PDF file** through Canvas.

```
set.seed(09042019) # Do not change this!
```

Problem 1 (4 points)

When asked to convert a vector, do not store it as a new vector unless asked to do so.

Consider the vector:

```
num <- c("1.5","2","8")
```

A. Convert `num` into a numeric vector using `as.numeric()`.

```
num <- c("1.5","2","8")
num <- as.numeric(num)
print(num)
```

```
## [1] 1.5 2.0 8.0
```

B. Convert `num` into a factor using `factor`. Store this in a new vector `num_fac`.

```
num <- c("1.5","2","8")
num_fac <- factor(num)
print(num_fac)
```

```
## [1] 1.5 2   8
## Levels: 1.5 2 8
```

C. Convert `num_fac` into a numeric using `as.numeric()`.

```
num_fac <- as.numeric(num_fac)
print(num_fac)
```

```
## [1] 1 2 3
```

D. Convert `num_fac` into a numeric using `as.numeric(as.character())`.

```
num_fac <- as.numeric(as.character(num_fac))
print(num_fac)
```

```
## [1] 1 2 3
```

Problem 2 (4 points)

When asked to convert a vector, do not store it as a new vector unless asked to do so.

Consider the vector:

```
num <- c(0, 1, 1, 0)
```

A. Append `TRUE` to the `num` vector using `c()`. Store the new vector as `num`.

```
num <- c(num, 'TRUE')
print(num)
```

```
## [1] "0"    "1"    "1"    "0"    "TRUE"
```

B. Check the class of `num` using `class()`.

```
num <- c(0, 1, 1, 0)
class(num)
```

```
## [1] "numeric"
```

C. Convert `num` into a logical vector using `as.logical()`.

```
num <- c(0, 1, 1, 0)
num <- as.logical(num)
print(num)
```

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

D. Convert `num` into a logical vector where the result is `TRUE` if `num` is 0, using the `==` operator.

```
num <- c(0, 1, 1, 0)
num <- num == 0
print(num)
```

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

Problem 3 (4 points)

Consider the data set `data_bmi` which is contained in the `bmi_age.txt` file. Download the file from Canvas and store it in the same folder that you have this *.Rmd file stored.

```
file_name = "bmi_age.txt"
data_bmi = read.table(file = file_name, header = TRUE, stringsAsFactors = FALSE)
```

Perform the following operations:

A. Extract the column nams of `data_bmi` using the `colnames()` function.

```
cat('\014')
```

```
colnames(data_bmi)
```

```
## [1] "PID" "BMI" "SEX" "AGE"
```

B. Extract the `AGE` column using the `[,]` operator.

```
cat('\014')
```

```
data_bmi[,'AGE']
```

```
##  [1] 45 57 66 49 33 40 65 59 65 42
```

C. Extract the `AGE` column using the `$` operator.

```
cat('\014')
```

```
data_bmi$AGE
```

```
##  [1] 45 57 66 49 33 40 65 59 65 42
```

D. Extract the `AGE` and `BMI` columns using brackets and the `c()` function.

```
cat('\014')
```

```
data_bmi[,c('AGE','BMI')]
```

```
##    AGE BMI
## 1   45  22
## 2   57  27
## 3   66  31
## 4   49  24
## 5   33  23
## 6   40  18
## 7   65  21
## 8   59  26
## 9   65  34
## 10  42  20
```

Problem 4 (3 points)

Here we will work with sequences and lengths:

A. Create a sequence from 1 to 4.5 by 0.24. Call this `run_num`.

```
cat('\014')
```

```
run_num <- seq(from = 1, to = 4.5, by = 0.24 )
print(run_num)
```

```
##  [1] 1.00 1.24 1.48 1.72 1.96 2.20 2.44 2.68 2.92 3.16 3.40 3.64 3.88 4.12
## [15] 4.36
```

B. Use the `length()` function to find the length of `run_num`?

```
cat('\014')
```

```
length(run_num)
```

```
## [1] 15
```

C. Extract the fifth element of `run_num` using brackets.

```
cat('\014')
```

```
run_num[5]
```

```
## [1] 1.96
```

Problem 5 (6 points)

Lets create a tibble called df:

```
df = dplyr::tibble(x = rnorm(10), y = rnorm(10), z = rnorm(10))
```

A. Extract the column `x` using the `$`.

```
cat('\014')
```

```
df$x
```

```
##  [1]  0.5786590 -0.3981434 -0.4739019  0.2420337  0.4469141  0.1046046
##  [7]  0.3013089 -1.3459991 -0.2647004  1.0165752
```

B. Extract the column `x` using the `[,]` notation.

```
cat('\014')
```

```
df['x']
```

```
## # A tibble: 10 x 1
##         x
##     <dbl>
##  1  0.579
##  2 -0.398
##  3 -0.474
##  4  0.242
##  5  0.447
##  6  0.105
##  7  0.301
##  8 -1.35
```

```
##  9 -0.265
## 10  1.02
```

C. Extract columns `x` and `z`.

```
cat('\014')
```

```
df[c('x','z')]
```

```
## # A tibble: 10 x 2
##         x       z
##     <dbl>   <dbl>
##  1  0.579  0.371
##  2 -0.398  0.206
##  3 -0.474  0.0109
##  4  0.242  0.728
##  5  0.447 -0.280
##  6  0.105 -0.0368
##  7  0.301 -0.283
##  8 -1.35   0.900
##  9 -0.265  1.31
## 10  1.02   0.0730
```

D. Extract the third and fifth rows of `df` and columns `z` and `y`.

```
cat('\014')
```

```
df[c(3,5),c('z','y')]
```

```
## # A tibble: 2 x 2
##         z      y
##     <dbl>  <dbl>
## 1  0.0109 -0.219
## 2 -0.280  -0.107
```

E. Get the mean of the column `x` using the `$` operator and `mean()`.

```
cat('\014')
```

```
mean(df$x)
```

```
## [1] 0.02073507
```

F. Pipe (`%>%`) the column `x` into the `mean()` function.

```
library(magrittr)
cat('\014')
```

```
df$x %>% mean()
```

```
## [1] 0.02073507
```

Problem 6 (10 points)

Consider the data set data_bmi using the BMI data, but read in using `readr`:

```
file_name = "bmi_age.txt"
data_bmi = readr::read_table2(file = file_name)
```

```
## Warning: Missing column names filled in: 'X5' [5]
```

```
## Parsed with column specification:
## cols(
##   PID = col_double(),
##   BMI = col_double(),
##   SEX = col_double(),
##   AGE = col_double(),
##   X5 = col_logical()
## )
```

```
## Warning: 2 parsing failures.
## row col  expected    actual          file
##   2  -- 5 columns 4 columns 'bmi_age.txt'
##   3  -- 5 columns 4 columns 'bmi_age.txt'
```

```
data_bmi
```

```
## # A tibble: 10 x 5
##      PID   BMI   SEX   AGE X5
##    <dbl> <dbl> <dbl> <dbl> <lgl>
##  1     1    22     1    45 NA
##  2     2    27     0    57 NA
##  3     3    31     1    66 NA
##  4     4    24     1    49 NA
##  5     5    23     0    33 NA
##  6     6    18     0    40 NA
##  7     7    21     0    65 NA
##  8     8    26     1    59 NA
##  9     9    34     1    65 NA
## 10    10    20     0    42 NA
```

A. What is the class of `data_bmi`?

```
cat('\014')
```

```
class(data_bmi)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

B. What is the class of `data_bmi[, "AGE"]`?

```
cat('\014')
```

```
class(data_bmi[,'AGE'])
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

C. What is the class of `data_bmi$AGE`?

```
cat('\014')
```

```
class(data_bmi$AGE)
```

```
## [1] "numeric"
```

D. What is the mean of the `AGE` column?

```
cat('\014')
```

```
data_bmi$AGE %>% mean()
```

```
## [1] 52.1
```

E. Set the 3rd element of `data_bmi$AGE` to be 42.

```
cat('\014')
```

```
data_bmi$AGE[3]=42
```

F. What is the mean of the `AGE` column now?

```
cat('\014')
```

```
data_bmi$AGE %>% mean()
```

```
## [1] 49.7
```

G. Remove the `X5` column using `data_bmi$X5 = NULL`.

```
cat('\014')
```

```
data_bmi$X5 = NULL
data_bmi
```

```
## # A tibble: 10 x 4
##      PID   BMI   SEX   AGE
##    <dbl> <dbl> <dbl> <dbl>
##  1     1    22     1    45
##  2     2    27     0    57
##  3     3    31     1    42
##  4     4    24     1    49
##  5     5    23     0    33
##  6     6    18     0    40
##  7     7    21     0    65
##  8     8    26     1    59
##  9     9    34     1    65
## 10    10    20     0    42
```

H. Create `mat`, which is `data_bmi` as a `matrix`, using `as.matrix`.

```
cat('\014')
```

```
mat <- data_bmi %>% as.matrix()
mat
```

```
##       PID BMI SEX AGE
##  [1,]   1  22   1  45
##  [2,]   2  27   0  57
##  [3,]   3  31   1  42
##  [4,]   4  24   1  49
##  [5,]   5  23   0  33
##  [6,]   6  18   0  40
##  [7,]   7  21   0  65
##  [8,]   8  26   1  59
##  [9,]   9  34   1  65
## [10,]  10  20   0  42
```

I. Try to extract `AGE` from mat using the `$`. What happened?

```
cat('\014')
```

```
mat$AGE
```

```
## Error in mat$AGE: $ operator is invalid for atomic vectors
```

J. Extract `AGE` from mat using the `[,]` notation.

```
cat('\014')
```

```
mat[,'AGE']
```

```
##  [1] 45 57 42 49 33 40 65 59 65 42
```

Problem 7 (5 points)

Use the `roll2()` function that you created during the lab for this problem. You will need to re-define the function here by copying/pasting the code into the R chunk below.

```
# put your roll2() function here
roll2 <- function(){
  die <- seq(1:6)
  dice <- sample(die, 2, replace = TRUE)
  return(sum(dice))
}
```

A. Create a numeric vector called `sim_num` which has the numbers 1 through 10.

```
cat('\014')
```

```
sim_num <- c(1,2,3,4,5,6,7,8,9,10)
print(sim_num)
```

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

B. Create a character vector called `roll` which has the results of 10 runs of the `roll2()` function.

```
cat('\014')
```

```
roll <- as.character(sample(roll2(), size = 10, replace = TRUE))
print(roll)
```

```
##  [1] "6" "6" "5" "3" "1" "5" "3" "7" "6" "6"
```

C. Create a data frame consisting of two columns using the two vectors you created above. Call this data frame `myrolls`.

```
cat('\014')
```

```
myrolls <- data.frame(sim_num, roll)
myrolls
```

```
##    sim_num roll
## 1        1    6
## 2        2    6
## 3        3    5
## 4        4    3
## 5        5    1
## 6        6    5
## 7        7    3
## 8        8    7
## 9        9    6
## 10      10    6
```

D. Convert `myrolls` into a tibble.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
cat('\014')
```

```
myrolls <- as.tbl(myrolls)
myrolls
```

```
## # A tibble: 10 x 2
##    sim_num roll
##      <dbl> <fct>
##  1       1 6
##  2       2 6
##  3       3 5
##  4       4 3
##  5       5 1
##  6       6 5
##  7       7 3
##  8       8 7
##  9       9 6
## 10      10 6
```

E. Using the data frame `myrolls` and the `table` function, create a table of the 10 roll results.

```
cat('\014')
```

```
table(myrolls)
```

```
##        roll
## sim_num 1 3 5 6 7
##      1  0 0 0 1 0
##      2  0 0 0 1 0
##      3  0 0 1 0 0
##      4  0 1 0 0 0
##      5  1 0 0 0 0
##      6  0 0 1 0 0
##      7  0 1 0 0 0
##      8  0 0 0 0 1
##      9  0 0 0 1 0
##      10 0 0 0 1 0
```

Session Info

```
sessionInfo()
```

```
## R version 3.5.3 (2019-03-11)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
```

```
## Running under: Windows 7 x64 (build 7601) Service Pack 1
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] dplyr_0.8.3  magrittr_1.5
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.1       knitr_1.22       hms_0.5.1        tidyselect_0.2.5
##  [5] R6_2.4.0         rlang_0.4.0      fansi_0.4.0      stringr_1.4.0
##  [9] tools_3.5.3      xfun_0.6         utf8_1.1.4       cli_1.1.0
## [13] htmltools_0.3.6  yaml_2.2.0       assertthat_0.2.1 digest_0.6.18
## [17] tibble_2.1.3     crayon_1.3.4     readr_1.3.1      purrr_0.3.2
## [21] vctrs_0.2.0      zeallot_0.1.0    glue_1.3.1       evaluate_0.13
## [25] rmarkdown_1.12   stringi_1.4.3    compiler_3.5.3   pillar_1.4.2
## [29] backports_1.1.3  pkgconfig_2.0.2
```