# STAT 33B Workbook 5

Ming Fong (3035619833)

Oct 1, 2020

This workbook is due **Oct 1, 2020** by 11:59pm PT.

The workbook is organized into sections that correspond to the lecture videos for the week. Watch a video, then do the corresponding exercises *before* moving on to the next video.

Workbooks are graded for completeness, so as long as you make a clear effort to solve each problem, you'll get full credit. That said, make sure you understand the concepts here, because they're likely to reappear in homeworks, quizzes, and later lectures.

As you work, write your answers in this notebook. Answer questions with complete sentences, and put code in code chunks. You can make as many new code chunks as you like.

In the notebook, you can run the line of code where the cursor is by pressing `Ctrl` + `Enter` on Windows or `Cmd` + `Enter` on Mac OS X. You can run an entire code chunk by clicking on the green arrow in the upper right corner of the code chunk.

Please do not delete the exercises already in this notebook, because it may interfere with our grading tools.

You need to submit your work in two places:

- Submit this Rmd file with your edits on bCourses.
- Knit and submit the generated PDF file on Gradescope.

If you have any last-minute trouble knitting, **DON'T PANIC**. Submit your Rmd file on time and follow up in office hours or on Piazza to sort out the PDF.

## Apply Function Basics

Watch the "Apply Function Basics" lecture video.

For exercises that mention the dogs data, you can use either `dogs.rds` or `dogs_tibble.rds`. Both are on the bCourse.

### Exercise 1

1. Suppose you call `sapply()` with a function that returns vectors. What class of object does `sapply()` return if all of the result vectors have the same length?

   For instance, what if the applied function returns a length-3 vector for each element?

   *Hint: `rnorm()` and `class()` are examples of functions that return vectors.*

2. Again suppose you call `sapply()` with a function that returns vectors. What class of object does `sapply()` return if the result vectors have the different lengths?

3. Suppose you call `sapply()` with a function that returns different types. What happens?

   *Hint: `all.equal()` is one function that returns different types.*

**YOUR ANSWER GOES HERE:**

1. `sapply` will return a matrix as the most simplified form.

```
x = c(3, 3, 3, 3)
sapply(x, rnorm, mean = 0, sd = 1)
```

```
##             [,1]       [,2]       [,3]        [,4]
## [1,] -0.34237151 -1.4588001 -0.7633501  0.71850761
## [2,] -0.92829259 -0.3983990  1.1365780  0.03275026
## [3,] -0.03682201 -0.1115983  1.0760445 -1.21581809
```

```
class(sapply(x, rnorm, mean = 0, sd = 1))
```

```
## [1] "matrix" "array"
```

2. `sapply` will return a list here because a matrix could not be formed with different row lengths.

```
x = c(1, 2, 3, 4)
sapply(x, rnorm, mean = 0, sd = 1)
```

```
## [[1]]
## [1] 0.892161
##
## [[2]]
## [1] 1.1881436 0.5410065
##
## [[3]]
## [1] -0.7440533 -0.4310260  0.9136143
##
## [[4]]
## [1] -1.0356595  0.9785687 -0.2834609  1.7555315
```

```
class(sapply(x, rnorm, mean = 0, sd = 1))
```

```
## [1] "list"
```

3. `sapply` returns a vector of class character. The `TRUE`s from `all.apply` are coerced into characters.

```
x = c(1, 2, 1, 4, 1, 6)
y = 1
sapply(x, all.equal, current = y)
```

```
## [1] "TRUE"                                "Mean relative difference: 0.5"
## [3] "TRUE"                                "Mean relative difference: 0.75"
## [5] "TRUE"                                "Mean relative difference: 0.8333333"
```

```
class(sapply(x, all.equal, current = y))
```

```
## [1] "character"
```

## Exercise 2

1. Use `sapply()` and `is.numeric()` to identify all of the numeric columns in the dogs data frame.
2. Use `sapply()` and your result from part 1 to compute the range of every numeric column in the dogs data frame.

**YOUR ANSWER GOES HERE:**

1.

```
dogs = readRDS("data/dogs.rds")
dogs_numerics = sapply(dogs, is.numeric)
dogs_numerics
```

```
##             breed             group           datadog    popularity_all
##             FALSE             FALSE              TRUE              TRUE
##        popularity     lifetime_cost intelligence_rank         longevity
##              TRUE              TRUE              TRUE              TRUE
##          ailments             price         food_cost          grooming
##              TRUE              TRUE              TRUE             FALSE
##              kids     megarank_kids          megarank              size
##             FALSE              TRUE              TRUE             FALSE
##            weight            height
##              TRUE              TRUE
```

2.

```
sapply(dogs[ , dogs_numerics], range, na.rm = TRUE)
```

```
##      datadog popularity_all popularity lifetime_cost intelligence_rank
## [1,]    0.99              1          1         12653                 1
## [2,]    3.64            173         87         26686                80
##      longevity ailments price food_cost megarank_kids megarank weight height
## [1,]      6.29        0   283       270             1        1      5      5
## [2,]     16.50        9  3460      1349            87       87    175     32
```

# The Split-Apply Strategy

Watch the "The Split-Apply Strategy" lecture video.

## Exercise 3

Use the split-apply strategy to compute the minimum weight (ignoring missing values) for each size of dog.

**YOUR ANSWER GOES HERE:**

```
sapply(split(dogs$weight, dogs$size), min, na.rm = TRUE)
```

```
##  large medium  small
##     55     16      5
```

## Exercise 4

Use `tapply()` to compute a `summary()` of the weight column for each group (hound, herding, etc) of dog.

**YOUR ANSWER GOES HERE:**

```
tapply(dogs$weight, dogs$group, summary)
```

```
## $herding
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   22.00   27.38   32.25   36.67   42.00   62.50      19
##
## $hound
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   23.00   51.12   63.75   63.83   83.12   97.50      14
##
```

```
## $`non-sporting`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   12.00   17.50   24.00   27.93   36.00   52.50      12
##
## $sporting
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   25.00   37.50   59.50   51.97   63.75   70.00      13
##
## $terrier
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   12.00   16.25   20.00   23.41   28.50   60.00       5
##
## $toy
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   5.000   5.500  10.000   9.818  12.750  16.000       8
##
## $working
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   47.50   79.38  107.50  105.00  126.25  175.00      15
```

## Exercise 5

The `aggregate()` function also implements the split-apply strategy, but returns the results as a data frame.

Use `aggregate()` to compute the maximum weight (ignoring missing values) for each group of dog.

*Hint: The `by` parameter in `aggregate()` expects a list or data frame, so use `[` to select columns for `by` rather than `$` or `[[`.*

**YOUR ANSWER GOES HERE:**

```
aggregate(dogs$weight, dogs["group"], max, na.rm = TRUE)
```

```
##          group     x
## 1      herding  62.5
## 2        hound  97.5
## 3 non-sporting  52.5
## 4     sporting  70.0
## 5      terrier  60.0
## 6          toy  16.0
## 7      working 175.0
```

## Exercise 6

Like `table()`, the `tapply()` function can use multiple categorical features to cross-tabulate results.

Use `tapply()` to compute the median price (ignoring missing values) for dogs, grouped by both size and grooming.

*Hint: see the `tapply()` documentation for the `INDEX` parameter.*

**YOUR ANSWER GOES HERE:**

```
tapply(dogs$price, list(dogs$size, dogs$grooming), median, na.rm = TRUE)
```

```
##        daily weekly monthly
## large  842.5   1040      NA
## medium 832.0    810     650
## small  693.0    740      NA
```

## Even More Apply Functions

Watch the "Even More Apply Functions" lecture video.

### Exercise 7

Translate your code from Exercise 1, Part 1 to use `vapply()` rather than `sapply()`.

**YOUR ANSWER GOES HERE:**

1. The return type here is still a matrix and the example output given was a length 3 numeric vector.

```
x = c(3, 3, 3, 3)
vapply(x, rnorm, c(1, 2, 3), mean = 0, sd = 1)
```

```
##             [,1]        [,2]       [,3]       [,4]
## [1,] -0.22329790 -0.09117337 -1.7723024 -0.2870713
## [2,] -0.03488387  0.08232011  0.3955784  0.4670120
## [3,]  0.88693093 -0.61688309 -0.3357295 -0.6706475
```

```
class(vapply(x, rnorm, c(1, 2, 3), mean = 0, sd = 1))
```

```
## [1] "matrix" "array"
```

## Choosing an Apply Function

Watch the "Choosing an Apply Function" lecture video.

No exercises for this section.

## Conditional Expressions

Watch the "Choosing an Apply Function" lecture video.

No exercises for this section.

## The `switch()` Function

Watch the "The switch() Function" lecture video.

No exercises for this section.

## The Congruent Vectors Strategy

Watch the "The Congruent Vectors Strategy" lecture video.

No exercises for this section.