Task Three Q&A

**Simple Operations and Data Structures**

1. Name three ways you could use attributes to make data analysis code more reproducible:
   1. You can set a class to it
   2. You can name it
   3. You can add a dimension to it
2. "What happens to a factor when you modify its levels?"

f1 <- factor(letters)

levels(f1) <- rev(levels(f1))

This code reverses the order of the levels of the factor.

1. What does this code do?

f2 <- **rev**(**factor**(letters))

f3 <- **factor**(letters, levels = **rev**(letters))

f2 reverses the factor list from “z” to “a” from f1, and f3 reverses it back so that f1 and f3 have letters running in the same direction from “a” to “z”

1. What does `dim` return when applied to a vector, \*\*and why\*\*?

dim(x) -> NULL

'dim' returns the value "NULL" because dim only applies to learning the multiple dimensions of a matrices or a data.frame and not a one-dimensional vector

1. What attributes does a data frame possess?

A data frame can possess a host of different attributes given to a column. You can assign a unit to a data frame column, or a relative size.

1. What does `as.matrix()` do when applied to a data frame with columns of different types?

It will convert the data frame into a matrix which changes all of the columns of different types to columns of the same data type.

1. Can you have a data frame with 0 rows? What about 0 columns?

Yes, to both:

> mtcars[mtcars$bob]

data frame with **0 columns** and 32 rows

> df[mtcars$cyl == 4, ]

[1] mpg cyl disp hp drat wt qsec vs am gear carb

<0 rows> (or 0-length row.names)

or

> data.frame(a=numeric(0))

[1] a

<**0 rows**> (or 0-length row.names)

**Simple Operations**

Q1: Use read.csv() to read the file “2016\_10\_11\_plate\_reader.csv” in the github data directory, and store it in memory as an object. This is an output from an instrument that I have, that measures fluorescence in each well of a 96-well plate. (Hint: use the optional argument skip = 33. What effect does that have?)

The “skip = 33” will skip 33 lines of the print out.

Q2: What kind of object did you create? What data type is each column of that object? (str())

> str(plate.reader)

'data.frame': 94 obs. of 3 variables:

$ well : Factor w/ 94 levels "A1","A10","A11",..: 1 5 6 7 8 9 10 11 2 3 ...

$ voltage : num -12533 -11667 -3267 -3000 -933 ...

$ r.squared: int 1 1 1 1 1 1 1 1 1 1 ...

The object created was a matrix and the three columns “well, voltage, and r.squared” were factors, numbers, and integers, respectively.

Now install and load the **tidyverse** package. (Remember, this package is a little unusual - it is a wrapper for about a dozen interrelated packages. Here you're using the **readr** package)

Read the same file using the read\_csv function. How is the resulting object different?

> read\_csv(file="2016\_10\_11\_plate\_reader.csv")

# A tibble: 127 × 3

`Software Version`

<chr>

1 <NA>

2 <NA>

3 <NA>

4 Experiment File Path:

5 Protocol File Path:

6 <NA>

7 <NA>

8 <NA>

9 Plate Number

10 Date

# ... with 117 more rows, and 2 more variables: `2.05.5` <chr>, X3 <chr>

This object is now like a data frame, but it is a tibble which is part of the tidyverse package.

**Subsetting**

1. Why does nrow(mtcars) give a different result than length(mtcars)? What does ncol(mtcars) return? What is each telling you, and why?

‘nrow’ gives you the number of rows from the table, ‘length’ gives you the number of columns in the table, and ‘ncol’ also gives you the number of columns in the table

1. Create a vector that is the cyl column of mtcars in two different ways:
   1. using the $ operator:

cyl <- mtcars$cyl

* 1. using [] subsetting:

cheese <- mtcars[[2]]

1. Create a data frame that contains all the columns of mtcars, but only with cars that weigh less than 3.0 **OR** more than 4.0 (weight is in the wt column)

> less\_than <- subset(df,wt<3|wt>4)

1. Create a data frame that contains all the **rows** of mtcars, but only the mpg and wt

> mpg\_wt <- mtcars[c('mpg','wt')]

1. Which cars in the database get gas mileage (mpg) equal to the median gas mileage for the set? (Use median and which).s

> equal.median<-mtcars[c(which(mtcars$mpg == median(mtcars$mpg))), ]

MERC 280 and Pontiac Firebird

1. AR 3.1.7.1: Fix the following common subsetting errors (note that mtcars is a dataset that is built into base R; you don't have to do anything special to load it:

mtcars[mtcars$cyl = 4, ] # Trying to create a data frame of cars with only 4 cylinders

> mtcars[mtcars$cyl == 4, ]

mtcars[-1:4, ]

> mtcars[-0:4, ]

mpg cyl disp hp drat wt qsec vs am gear carb

Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4

Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4

Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1

Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1

> mtcars[-0:4]

mpg cyl disp hp

Mazda RX4 21.0 6 160.0 110

Mazda RX4 Wag 21.0 6 160.0 110

Datsun 710 22.8 4 108.0 93

Hornet 4 Drive 21.4 6 258.0 110

Hornet Sportabout 18.7 8 360.0 175

Valiant 18.1 6 225.0 105

etc

mtcars[mtcars$cyl <= 5]

> mtcars[mtcars$cyl <= 5, ]

mtcars[mtcars$cyl == 4 | 6, ] # The | is an 'or' operator - you want a data frame of cars with 4 OR 6 cylinder engines

mtcars[mtcars$cyl == 4 | mtcars$cyl == 6, ]