Task 3

Emily Nield

January 26, 2017

Data Structures

Question 1: a) Explore vectorization. Explain the differences and the similarities between the following code snippets.

```
a <- 1
b <- 2
c <- a + b
```

and

```
set.seed(0) # This ensures that 'random' results will be the same for everyone
d <- rnorm(20)
e <- rnorm(20)
f <- d + e
```

Both of these codes are vectors and are their elements are the same type.We can determine the type of each code snippet using typeof(). Both of them are double. a, b, and c are all single value vectors as they represent one digit while d, e, and f are length 20 vectors.

```
typeof(c)
```

```
## [1] "double"
```

```
typeof(f)
```

```
## [1] "double"
```

b) What data structure are a, b, and c?

There are three options: atomic vectors, lists, and data frames.

a, b, and c are all atomic vectors. You can quickly test this by using the is.atomic() or is.list() function.

```
is.atomic(a)
```

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

```
is.list(a)
```

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

```
is.data.frame(a)
```

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

Attributes

Question 2: Name three ways you could use attributes to make data analysis code more reproducible (i.e., easier for yourself and others to understand).

Attributes are used to store metadata about the object which makes them ideal to use in order to make your code more reproducible. When writing code you could use a lot of objects and you will want to know what

functions have been performed on your objects, what type they are, and what they represent. Attributes can store this information and you can quickly recall is using the attr() function.

Ex.

```
y <- 1:10
attr(y, "my_attribute") <- "This is a vector"
attr(y, "my_attribute")
```

```
## [1] "This is a vector"
```

Attributes can be used to define factors (used to store categorical data).

Ex.

```
sex_char <- c("m", "m", "m")
sex_factor <- factor(sex_char, levels = c("m", "f"))
table(sex_char)
```

```
## sex_char
## m
## 3
```

```
table(sex_factor)
```

```
## sex_factor
## m f
## 3 0
```

The three most important attributes (names, dimensions, and class) are preserved when one modifies their vector.

Question 3: Create a vector of length 5, and use the attr function to associate two different attributes to the vector.

```
vector.length.5 <- 1:5
attr(vector.length.5, "attr1") <- "This is a vector"
attr(vector.length.5, "attr2") <- "Emily is awesome!"
attr(vector.length.5, "attr1") #view attributes individually
```

```
## [1] "This is a vector"
```

```
attr(vector.length.5, "attr2")
```

```
## [1] "Emily is awesome!"
```

```
str(attributes(vector.length.5)) #view all atributes as a list
```

```
## List of 2
##  $ attr1: chr "This is a vector"
##  $ attr2: chr "Emily is awesome!"
```

Question 2.2.2.2 from book: What happens to a factor when you modify its levels?

```
f1 <- factor(letters)
f1
```

```
##  [1] a b c d e f g h i j k l m n o p q r s t u v w x y z
## Levels: a b c d e f g h i j k l m n o p q r s t u v w x y z
```

f1 is a factor with 26 levels corresponding to the alphabet.

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

```
##  [1] z y x w v u t s r q p o n m l k j i h g f e d c b a
## Levels: z y x w v u t s r q p o n m l k j i h g f e d c b a
```

When you reverse it there are still 26 levels but in reverse order (Z to a) not (a to z).

Matrices and Arrays

Question 2.3.1.1: What does `dim` return when applied to a vector, and why?

Adding a dim attribute to an atomic vector allows it to behave like a multi-dimensional array. Matrices and arrays are created with matrix() and array(), or by using the assignment form of dim().

Apply dim to a vector and...

```
dim(d)
```

```
## NULL
```

it returns a NULL because it is not combined with any other vectors to make an array. It's all on its own. Let's fix this.

```
array <- cbind(d,e)
dim(array)
```

```
## [1] 20  2
```

Data Frames

Question 2.4.5.1: What attributes does a data frame possess?

A data frame is a list of equal-length vectors. This makes it a 2-dimensional structure, so it shares properties of both the matrix and the list. This means that a data frame has names(), colnames(), and rownames(), although names() and colnames() are the same thing. The length() of a data frame is the length of the underlying list and so is the same as ncol(); nrow() gives the number of rows.

Let's create a data frame to play around with.

```
df <- data.frame(x = 1:3, y = c("a", "b", "c"))
str(df)
```

```
## 'data.frame':    3 obs. of  2 variables:
##  $ x: int  1 2 3
##  $ y: Factor w/ 3 levels "a","b","c": 1 2 3
```

Question 2.4.5.2: What does `as.matrix()` do when applied to a data frame with columns of different types? *Ask Drew*

Let's create a data frame with different types of data structures.

```
df2 <- data.frame(x = 1:3)
df2$y <- list(1:2, 1:3, 1:4)
df2
```

```
##   x          y
## 1 1       1, 2
## 2 2    1, 2, 3
## 3 3 1, 2, 3, 4
```

Apply the as.matrix function to the new data frame.

```
as.matrix (df2)
```

```
##      x y
## [1,] 1 Integer,2
## [2,] 2 Integer,3
## [3,] 3 Integer,4
```

The function still works but it summarized the data so that the matrix in the y column was reduced to the last number.

Question 2.4.5.3: Can you have a data frame with 0 rows? What about 0 columns?

Yes! Fill it with empty vectors. Source: http://stackoverflow.com/questions/10689055/create-an-empty-data-frame

```
df3 <- data.frame(Date=as.Date(character()),
                  File=character(),
                  User=character(),
                  stringsAsFactors=FALSE)
```

An alternate way to do this is...

```
df4 <- data.frame(Doubles=double(),
                  Ints=integer(),
                  Factors=factor(),
                  Logicals=logical(),
                  Characters=character(),
                  stringsAsFactors=FALSE)

str(df4)
```

```
## 'data.frame':    0 obs. of  5 variables:
##  $ Doubles   : num
##  $ Ints      : int
##  $ Factors   : Factor w/ 0 levels:
##  $ Logicals  : logi
##  $ Characters: chr
```

Simple Operations

Question: 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?)

```
flur<-read.csv("2016_10_11_plate_reader.csv")
head(flur)
```

```
##   well    voltage r.squared
## 1   A1 -12533.333         1
## 2   A2 -11666.667         1
## 3   A3  -3266.667         1
## 4   A4  -3000.000         1
## 5   A5   -933.333         1
## 6   A6   -866.667         1
```

Question: What kind of object did you create?

A data frame

Question: What data type is each column of that object?

```
str(flur)
```

```
## '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 ...
```

Atomic vectors

Question: Now install and load the tidyverse package. Read the same file using the read_csv function. How is the resulting object different?

```
library(tidyverse)
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr
```

```
## Conflicts with tidy packages ----------------------------------------------
```

```
## filter(): dplyr, stats
## lag():    dplyr, stats
```

```
flur2<-read.csv("2016_10_11_plate_reader.csv")
head(flur2)
```

```
##   well    voltage r.squared
## 1   A1 -12533.333         1
## 2   A2 -11666.667         1
## 3   A3  -3266.667         1
## 4   A4  -3000.000         1
## 5   A5   -933.333         1
## 6   A6   -866.667         1
```

It's not different but I did modify the original csv file by deleting the first 33 rows.

Subsetting

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

Start by previewing the data set.

```
head(mtcars)
```

```
##                   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
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Now apply the nrow(), ncols(), and length() functions.

```r
nrow(mtcars)
```

```
## [1] 32
```

```r
ncol(mtcars)
```

```
## [1] 11
```

```r
length(mtcars)
```

```
## [1] 11
```

nrow() and ncol() return the number of rows or columns present in x.

length(): Get or set the length of vectors (including lists) and factors, and of any other R object for which a method has been defined.

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

o using [] subsetting

```r
cyl<- as.vector(mtcars$cyl)
cyl2 <- as.vector(mtcars['cyl'])
class(cyl2) #Still getting it as data frame not vector. ASK!!!!
```

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

Question: 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)

```r
wt <- subset(mtcars, wt > 4 | wt < 3)
head(wt)
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
```

Question: Create a data frame that contains all the rows of mtcars, but only the mpg and wt. Which cars in the database get gas mileage (mpg) equal to the median gas mileage for the set? (Use median and which)

```r
mpg_wt <- subset(mtcars, select=c(mpg, wt))
head(mpg_wt)
```

```
##                    mpg    wt
## Mazda RX4         21.0 2.620
## Mazda RX4 Wag     21.0 2.875
## Datsun 710        22.8 2.320
## Hornet 4 Drive    21.4 3.215
## Hornet Sportabout 18.7 3.440
## Valiant           18.1 3.460
```

```r
mpg<- as.vector(mtcars$mpg) #isolate mpg column
med<-median(mpg) #calculate the median
which(mpg==med)
```

```
## [1] 10 25
```

Alternate way

```
medians_cars<-mtcars[which(mtcars$mpg==median(mtcars$mpg)),] #By doing this you are returning the names
```

Question 3.1.7.1: Fix the following common subsetting errors.

Wrong:

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

Right:

```
mtcars[mtcars$cyl==4, ]
```

```
##                 mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Merc 240D      24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230       22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona  21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

Wrong:

```
#mtcars[-1:4, ]
```

Right:

```
mtcars[c(1,4), ]
```

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

Wrong:

```
#mtcars[mtcars$cyl <= 5]
```

Right:

```
mtcars[mtcars$cyl <= 5, ]
```

```
##                 mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Merc 240D      24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230       22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona  21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

Wrong:

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

Right:

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

```
##                 mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4      21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Valiant        18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Merc 240D      24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230       22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280       19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C      17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona  21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ferrari Dino   19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```