Exercise: **Vectors**

Day 1, Part B

1. Create the following vectors:

   a. The names of everyone sitting at your table (`friends`).

```
> friends <- c("Laura", "Jon", "Nafis", "Kirsten")
> friends
[1] "Laura"   "Jon"     "Nafis"   "Kirsten"
```

   b. Every calendar year since 1995 (`years`).

```
> years <- 1995:2017
> years
[1] 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010
[17] 2011 2012 2013 2014 2015 2016 2017
```

   c. A sequence from 0 to 1 by 0.1 (`tenths`).

```
> tenths <- seq(0, 1, 0.1)
> tenths
[1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

   d. For each day this week, whether or not you have/will attend a boot camp class (`classes`).

```
> classes <- c(F, F, T, T, T, F, F)
> classes
[1] FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE
```

2. Consider the vectors from question 1.

   a. What class is each vector?

```
> class(friends)
[1] "character"
```

```
> class(years)
[1] "integer"
```

```
> class(tenths)
[1] "numeric"
```

```
> class(classes)
[1] "logical"
```

```
> # friends = character; years = integer; tenths = numeric; classes = logical.
```

   b. What happens to each vector when it is coerced to an integer?

```
> as.integer(friends)
Warning: NAs introduced by coercion
[1] NA NA NA NA
```

```
> as.integer(years)
[1] 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010
[17] 2011 2012 2013 2014 2015 2016 2017
```

```
> as.integer(tenths)
[1] 0 0 0 0 0 0 0 0 0 0 1
```

```
> as.integer(classes)
[1] 0 0 1 1 1 0 0
```

```
> # Coercing friends to an integer creates NAs.  Years is already an integer, so nothing
> # changes.  Coercing tenths causes all of the numbers to be rounded down to an
> # integer.  Coercing classes causes FALSEs to become 0s and TRUEs to become 1s.
```

c. What happens to each vector when it is coerced to a numeric?

```
> as.numeric(friends)
Warning: NAs introduced by coercion
[1] NA NA NA NA
```

```
> as.numeric(years)
[1] 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010
[17] 2011 2012 2013 2014 2015 2016 2017
```

```
> as.numeric(tenths)
[1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

```
> as.numeric(classes)
[1] 0 0 1 1 1 0 0
```

```
> # Coercing friends to a numeric creates NAs.  Nothing changes about years when coerced
> # to a numeric.  Tenths is already a numeric, so nothing changes.  Coercing classes
> # causes FALSEs to become 0s and TRUEs to become 1s.
```

d. What happens to each vector when it is coerced to a character?

```
> as.character(friends)
[1] "Laura"   "Jon"     "Nafis"   "Kirsten"
```

```
> as.character(years)
[1] "1995" "1996" "1997" "1998" "1999" "2000" "2001" "2002" "2003" "2004" "2005"
[12] "2006" "2007" "2008" "2009" "2010" "2011" "2012" "2013" "2014" "2015" "2016"
[23] "2017"
```

```
> as.character(tenths)
[1] "0"   "0.1" "0.2" "0.3" "0.4" "0.5" "0.6" "0.7" "0.8" "0.9" "1"
```

```
> as.character(classes)
[1] "FALSE" "FALSE" "TRUE"  "TRUE"  "TRUE"  "FALSE" "FALSE"
```

```
> # Friends is already a character, so nothing changes.  The other three remain the same
> # except that the values are now enclosed in quotes.
```

e. What happens to each vector when it is coerced to a logical?

```
> as.logical(friends)
[1] NA NA NA NA
```

```
> as.logical(years)
[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[17] TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
> as.logical(tenths)
[1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
> as.logical(classes)
[1] FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE
```

```
> # Coercing friends to a logical creates NAs.  Coercing years causes all values to
> # become 'TRUE'.  Coercing tenths causes all values greater than 0 to become 'TRUE'.
> # Classes is already a logical, so nothing changes.
```

f. [Bonus] In general, what happens when you convert numerics/integers to a logical? (hint: try running different numbers through `as.logical()` until the pattern becomes clear)

```
> as.logical(0)
[1] FALSE
```

```
> as.logical(0.1)
[1] TRUE
```

```
> as.logical(100)
[1] TRUE
```

```
> as.logical(-0.1)
[1] TRUE
```

```
> as.logical(-100)
[1] TRUE
```

```
> # 0s become FALSE and everything else becomes TRUE.
```

g. [Bonus] Is it ever possible to convert a character vector to a numeric or logical vector without introducing NAs?

```
> # It is possible to convert a character to a numeric if the character vector contains
> # values that can be interpreted as numbers.
> str_nums <- c("0", "1", "2", "3", "4")
> class(str_nums)
[1] "character"
```

```
> as.numeric(str_nums)
[1] 0 1 2 3 4
```

```
> # It is possible to convert a character to a logical if the character vector contains
> # 'T' and 'F' or 'TRUE' and 'FALSE'.
> str_log <- c("T", "F", "TRUE", "FALSE")
> class(str_log)
[1] "character"
```

```
> as.logical(str_log)
[1]  TRUE FALSE  TRUE FALSE
```

```
> # It's also possible to convert a character to a logical if the character vector
> # contains values that can be interpreted as numbers, BUT it must be converted to a
> # numeric first.
> as.logical(str_nums)
[1] NA NA NA NA NA
```

```
> as.logical(as.numeric(str_nums))
[1] FALSE  TRUE  TRUE  TRUE  TRUE
```

3. Consider the following vectors which contain data about counties in the Puget sound region:

```
# Name
cnty_name <- c("Jefferson", "Kitsap", "Pierce", "King", "Snohomish", "Skagit", "Whatcom",
    "San Juan", "Island")
# Population
cnty_pop <- c(30183, 255104, 835555, 2089564, 763963, 120718, 208935, 16029, 79291)
# Area (sq. miles)
cnty_area <- c(1855, 450.6, 1781, 2238.8, 2116.4, 1771.7, 2175, 254.4, 219.1)
# Life expectancy (yrs)
cnty_e0 <- c(81.3, 79.7, 78.7, 81.4, 80.2, 79.8, 81, 83.7, 81.9)
```

a. Calculate the population density of these counties.

```
> cnty_pop/cnty_area
[1]  16.27116 566.14292 469.14935 933.34108 360.97288  68.13682  96.06207  63.00708
[9] 361.89411
```

b. What is the minimum and maximum life expectancy?

```
> # using range:
> range(cnty_e0)
[1] 78.7 83.7
```

```
> # using quantile:
> quantile(cnty_e0, c(0, 1))
0% 100%
78.7 83.7
```

```
> # using min and max:
> min(cnty_e0)
[1] 78.7
```

```
> max(cnty_e0)
[1] 83.7
```

c. Which county has the lowest life expectancy? The highest?

```
> cnty_name[cnty_e0 == min(cnty_e0)]
[1] "Pierce"
```

```
> cnty_name[cnty_e0 == max(cnty_e0)]
[1] "San Juan"
```

```
> # or, using which.min() and which.max():
> cnty_name[which.min(cnty_e0)]
[1] "Pierce"
```

```
> cnty_name[which.max(cnty_e0)]
[1] "San Juan"
```

d. What is the median population size?

```
> median(cnty_pop)
[1] 208935
```

e. Which counties have populations greater than 100,000?

```
> cnty_name[cnty_pop > 1e+05]
[1] "Kitsap"    "Pierce"    "King"      "Snohomish" "Skagit"    "Whatcom"
```

f. What is the mean area of counties with populations greater than 100,000?

```
> mean(cnty_area[cnty_pop > 1e+05])
[1] 1755.583
```

4. Create a vector called `draws` that is 100 random draws from a Normal(0,1) distribution (hint: see `rnorm()`).

```
> draws <- rnorm(100)
```

a. Find the mean, variance, and standard deviation of `draws`.

```
> mean(draws)
[1] 0.0474962
```

```
> var(draws)
[1] 0.8942142
```

```
> sd(draws)
[1] 0.945629
```

b. Create a second vector (`log_draws`) that is the natural log of the `draws` vector.

```
> log_draws <- log(draws)
Warning in log(draws): NaNs produced
```

c. Show just the non-missing values of `log_draws`.

```
> log_draws[!is.na(log_draws)]
[1] -2.12657310 -1.41186890  0.20902506  0.51622916 -0.48128977  0.10422319
 [7] -0.34589278 -2.81758714 -0.12256263  0.67054055 -1.53926455 -0.86992635
[13] -1.12607287  0.40283543  0.64309618 -0.72265843 -2.40244248  0.46907118
[19]  0.07770104 -0.46083917 -3.05432726  0.26251708  0.82989544  0.43669310
[25] -2.41679327 -0.16840326 -0.03819216 -0.37934508 -0.16293896 -1.74409737
[31] -2.59626945 -0.84824252 -3.70196539 -0.30585152 -0.95184908 -2.13584673
[37] -2.00962712 -1.50950449  0.49521206 -1.78340219  0.15562149  0.05276418
[43]  0.13563440  0.69438778 -2.70753727  0.62425351 -3.86401475  0.22307521
[49] -1.10509574 -1.55126130  0.21242636
```

d. How many values of `log_draws` are missing? (hint: this requires two functions)

```
> sum(is.na(log_draws))
[1] 49
```

```
> length(log_draws[is.na(log_draws)])
[1] 49
```