18.S096 Problem Set 1 Spring 2018
Due Date: 2/16/2018
Where: On Stellar, prior to 11:59pm

Collaboration on homework is encouraged, but you will benefit from independent effort to solve the problems before discussing them with other people. **You must write your solution in your own words. List all your collaborators.**

1. **QR-Decomposition and Least-Squares Regression**

   Consider the standard setup for a linear regression model:

   $$\vec{y} = \boldsymbol{X}\vec{\beta} + \vec{\epsilon}$$

   where $\vec{y} \in R^n$, is the n-vector of response variable values, $\boldsymbol{X}$ is an $n \times p$ matrix of covariate variable values (with p variables in the columns), $\vec{\beta} \in R^p$ is the regression parameter, and $\vec{\epsilon} \in R^n$ is the regression error.

   The standard (Gauss-Markov) assumptions on the regression error vector is that:

   $$E[\vec{\epsilon}] = \vec{0} \text{ (n-vector of 0s)}$$
   $$Cov[\vec{\epsilon}] = \sigma^2 \mathrm{I}_n$$

   where I_n is the $n \times n$ identity matrix and $\sigma^2 > 0$, is the (constant) variance of the errors.

   With this model, the least squares estimate of $\vec{\beta}$ is

   $$\hat{\beta} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \vec{y}$$

   and the expectation and covariance of $\hat{\beta}$ are:

   $$E[\hat{\beta}] = \vec{\beta} \text{ (unbiased)}$$
   $$Cov[\hat{\beta}] = \sigma^2 (\boldsymbol{X}^\top \boldsymbol{X})^{-1}.$$

   The standard estimate of σ^2 is the unbiased estimate:

   $$\begin{aligned} \hat{\sigma}^2 &= \frac{RSS}{n-p} = |\vec{y} - \boldsymbol{X}\hat{\beta}|^2/(n-p) = \frac{[\vec{y}-\boldsymbol{X}\hat{\beta}]^T[\vec{y}-\boldsymbol{X}\hat{\beta}]}{(n-p)} \\ &= \frac{\vec{y}^T[\mathrm{I}_n - \boldsymbol{H}]\vec{y}}{n-p} \end{aligned}$$

   where $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}$ is the 'hat' matrix projecting R^n onto the column space of $\boldsymbol{X}$.

   The Q-R Decomposition of $\boldsymbol{X}$

   Consider expressing the $(n \times p)$ matrix $\boldsymbol{X}$ of explanatory variables as

   $$\boldsymbol{X} = \boldsymbol{Q} \cdot \boldsymbol{R}$$

   where

   $\boldsymbol{Q}$ is an $(n \times p)$ orthonormal matrix, i.e., $\boldsymbol{Q}^T\boldsymbol{Q} = I_p$.

   $\boldsymbol{R}$ is a $(p \times p)$ upper-triangular matrix.

The columns of $\boldsymbol{Q} = [\boldsymbol{Q}_{[1]}, \boldsymbol{Q}_{[2]}, \ldots, \boldsymbol{Q}_{[p]}]$ can be constructed by performing the *Gram-Schmidt Orthonormalization* procedure on the columns of $\boldsymbol{X} = [\boldsymbol{X}_{[1]}, \boldsymbol{X}_{[2]}, \ldots, \boldsymbol{X}_{[p]}]$

If

$$\boldsymbol{R} = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,p-1} & r_{1,p} \\ 0 & r_{2,2} & \cdots & r_{2,p-1} & r_{2,p} \\ 0 & 0 & \ddots & \vdots & \vdots \\ 0 & 0 & & r_{p-1,p-1} & r_{p-1,p} \\ 0 & 0 & \cdots & 0 & r_{p,p} \end{bmatrix},$$

then

- $\boldsymbol{X}_{[1]} = \boldsymbol{Q}_{[1]} r_{1,1}$
  $\Longrightarrow$
  $$\begin{aligned} r_{1,1}^2 &= \boldsymbol{X}_{[1]}^T \boldsymbol{X}_{[1]} \\ \boldsymbol{Q}_{[1]} &= \boldsymbol{X}_{[1]} / r_{1,1} \end{aligned}$$
- $\boldsymbol{X}_{[2]} = \boldsymbol{Q}_{[1]} r_{1,2} + \boldsymbol{Q}_{[2]} r_{2,2}$
  $\Longrightarrow$
  $$\begin{aligned} \boldsymbol{Q}_{[1]}^T \boldsymbol{X}_{[2]} &= \boldsymbol{Q}_{[1]}^T \boldsymbol{Q}_{[1]} r_{1,2} + \boldsymbol{Q}_{[1]}^T \boldsymbol{Q}_{[2]} r_{2,2} \\ &= 1 \cdot r_{1,2} + 0 \cdot r_{2,2} \\ &= r_{1,2} \quad \text{(known since } \boldsymbol{Q}_{[1]} \text{ specfied)} \end{aligned}$$
- With $r_{1,2}$ and $\boldsymbol{Q}_{[1]}$ specfied we can solve for $r_{2,2}$:
  $\Longrightarrow$
  $$\boldsymbol{Q}_{[2]} r_{2,2} = \boldsymbol{X}_{[2]} - \boldsymbol{Q}_{[1]} r_{1,2}$$

  Take squared norm of both sides:
  $$r_{2,2}^2 = \boldsymbol{X}_{[2]}^T \boldsymbol{X}_{[2]} - 2 r_{1,2} \boldsymbol{Q}_{[1]}^T \boldsymbol{X}_{[2]} + r_{1,2}^2$$

  (all terms on RHS are known)

  With $r_{2,2}$ specified
  $\Longrightarrow$
  $$\boldsymbol{Q}_{[2]} = \tfrac{1}{r_{2,2}} \left[\boldsymbol{X}_{[2]} - r_{1,2} \boldsymbol{Q}_{[1]} \right]$$
- Etc. (solve for elements of $\boldsymbol{R}$, and columns of $\boldsymbol{Q}$)

With the Q-R Decomposition

$$\boldsymbol{X} = \boldsymbol{Q}\boldsymbol{R}$$

($\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}_p$, and $\boldsymbol{R}$ is $p \times p$ upper-triangular)

1(a) Show that:

$$\hat{\boldsymbol{\beta}} = \boldsymbol{R}^{-1}\boldsymbol{Q}^T\boldsymbol{y}$$

and

$$Cov(\hat{\boldsymbol{\beta}}) = \sigma^2\boldsymbol{R}^{-1}(\boldsymbol{R}^{-1})^T$$

1(b) Write an R script to fit the regression parameter of the 10-variable linear regression model for the diabetes data in $Rproject1$ using just matrix algebra and the QR decomposition formula of part (a). Confirm that the estimates obtained are the same as those computed with the R function $lm()$.

Notes: The RSweave file '$RProject1\_IntroToR_Diabetes.Rnw$' has R commands for reading in the the diabetes data and fitting the model using the R function $lm()$.

The R script file '$Rproject1\_script1.r$' in $RProject1$ illustrates the use of the R function $qr()$. (There are companion functions $qr.Q()$ and $qr.R()$ which operate on the output object from $qr()$.)

1(c) Partition the regression model according the the first k independent variables (I), and the last $(p-k)$ independent variables (II):

$$X = [X_I X_{II}], \text{ and } \beta = \begin{bmatrix} \beta_I \\ \beta_{II} \end{bmatrix}$$

Consider the least-squares fit of the linear regression model using only the first k covariates (columns) of $\boldsymbol{X}$.

The estimate:

$$\hat{\beta}_0 = \begin{pmatrix} \hat{\beta}_I \\ 0_{p-k} \end{pmatrix} \text{ where}$$

$$\begin{aligned} \hat{\beta}_I &= (X_I^T X_I)^{-1} X_I^T y \\ X_I &= \left[X_{[1]} X_{[2]} \cdots X_{[k]}\right] \end{aligned}$$

The estimate $\hat{\beta}_0$ is the constrained least-squares estimate of β corresponding to the hypothesis H_0, i.e.,

$$\hat{\beta}_0 \text{ minimizes: } SS(\beta) = (y - X\beta)^T(y - X\beta)$$

subject to

$$\hat{\beta}_j = 0,\ j = k+1, k+2, \ldots, p.$$

Show that the QR-decomposition of X_I is $X_I = Q_I R_I$, where Q_I is the matrix of the first k columns of Q and R_I is the upper-left $k \times k$ block of R. Furthermore, verify that:

$$\hat{\beta}_I = R_I^{-1} Q_I^T y \text{ and}$$

$$\hat{y}_I = H_I y,$$

where $H_I = Q_I Q_I^T$, the $n \times n$ projection/Hat matrix under the null hypothesis.

1(d) Using RStudio/R, fit the linear regression model of the diabetes data for dependent variable *prog* using only the first four covariates (age, sex, bmi, and map). Obtain the sub-model fit of the regression parameter in two ways:

- Use the function $lm()$ with the model expression
$$prog\ age + sex + bmi + map$$
- Use matrix algebra with the QR-decomposition.

Verify that you get the same estimates.

2. **Writing/testing your own R functions and Data Quality Analysis**

The initial analysis of the diabetes data computed summary statistics for each (column) variable using the R function *summary*() in conjunction with the *apply*() function.

```
> # 1.  Read data into R ----
> diabetes= read.csv(file="EfronData/diabetes.csv", sep=",", header=TRUE)
> #   Use str() to display structure
> #str(diabetes)
> # 2. Compute summary statistics ----
> apply(diabetes,2,summary)
```

	age	sex	bmi	map	tc	ldl	hdl	tch	ltg	glu	prog
Min.	19.00	0.0000	18.00	62.00	97.0	41.60	22.00	2.00	1.410	58.00	25.0
1st Qu.	38.25	0.0000	23.20	84.00	164.2	96.05	40.25	3.00	1.860	83.25	87.0
Median	50.00	0.0000	25.70	93.00	186.0	113.00	48.00	4.00	2.005	91.00	140.5
Mean	48.52	0.4683	26.38	94.65	189.1	115.40	49.79	4.07	2.016	91.26	152.1
3rd Qu.	59.00	1.0000	29.28	105.00	209.8	134.50	57.75	5.00	2.170	98.00	211.5
Max.	79.00	1.0000	42.20	133.00	301.0	242.40	99.00	9.09	2.650	124.00	346.0

When analyzing new data sets, it is important to assess data quality. Many quality/features of the data variables can be measured by standard statistics (mean, standard deviation,etc.). Additionally, data sets can be affected by the presence of missing data and the discretization of variables. These latter issues can be measured by measuring the count/proportion of dataset variables that are missing (NA values in R) and the proportion of data values that are unique, This uniqueness statistic is low for well-defined discrete variables, but can be low for continuous variables whose measurements are discretized. These features of a dataset's variables are easily computed and awareness of them contributes to higher-quality analyses.

2(a) Customize a new summary function for data quality analysis:

$fcn.dqa0()$

to compute the following statistics for an input (numeric) vector $x = (x_1, x_2, \ldots, x_n)^\top$.

- sample size: n
- $n.missing$ Number of missing values
- $n.unique$: number of unique values
- sample mean: $\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$
- sample standard deviation: $\hat{\sigma} = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}}$
- sample skewness: $\hat{skew} = (\frac{\sum_{i=1}^n (x_i - \bar{x})^3}{n})/(\hat{\sigma})^3$
- sample kurtosis: $\hat{kurtosis} = (\frac{\sum_{i=1}^n (x_i - \bar{x})^4}{n})/(\hat{\sigma})^4$
- $median = median(x)$
- iqr (interquartile range of x)
- $q05$ (5th percentile)
- $q95$ (95th percentile)

and a function $fcn.dqa0.matrix()$ to apply the column-wise sumamry function $fcn.dqa0()$ to each column of an input numeric matrix/data frame.

Notes/hints:

You may edit the R script file $Pset1\_Problem1.r$ which constructs the following R functions which partially accomplishes this purpose:

- $fcn.dqa1()$: computes the following statistics for a numeric input vector.
  - sample size: n
  - sample mean: $\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$
  - sample standard deviation: $\hat{\sigma} = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}}$
  - sample skewness: $\hat{skew} = (\frac{\sum_{i=1}^n (x_i - \bar{x})^3}{n})/(\hat{\sigma})^3$
  - sample kurtosis: $\hat{kurtosis} = (\frac{\sum_{i=1}^n (x_i - \bar{x})^4}{n})/(\hat{\sigma})^4$
- $fcn.dqa1.matrix()$: computes a statistics table for a numeric input matrix/data frame, applying the function $fcn.dqa1()$ for each column variable.
  The output table has columns corresponding to the variables/columns of the input matrix and rows corresponding to the different statistics.
- To complete this problem, the following R functions/expressions can be used:
  - $quantile(x, probs = c(.5, .25, .75, .05, .95))$
    (computes sample quantiles corresponding to the given probabilities)

- $sum(is.na(x))$
  (computes the count of missing (NA) values in a vector x.
- $length(unique(x))$
  (computes the number of distinct values in a vector; much smaller than length(x) when input vector x is discrete or discretized.)
  $$quantile(x, probs = c(.5, .25, .75, .05, .95))$$
  (computes sample quantiles corresponding to the given probabilities)

2(b) Revise the functions of part (a) to handle missing values and compute the statistics for the non-missing values of each numeric input vector (for $fcn.dqa0()$) or variable/column of a numeric input matrix (for $fcn.dqa0.matrix()$).

Notes/hints:

Data sets often suffer the problem of having some missing data (represented by the values 'NA'. In R, the default result of any numerical computation with missing values is NA as well. To illustrate this, we create a copy of the diabetes data set and set to missing the first and second column variables of cases 1 and 2:

```
> diabetes2=diabetes
> diabetes2[1,1]<-NA
> diabetes2[2,2]<-NA
> head(diabetes2)
```

```
  age sex  bmi map  tc   ldl hdl tch  ltg glu prog
1  NA   1 32.1 101 157  93.2  38   4 2.11  87  151
2  48  NA 21.6  87 183 103.2  70   3 1.69  69   75
3  72   1 30.5  93 156  93.6  41   4 2.03  85  141
4  24   0 25.3  84 198 131.4  40   5 2.12  89  206
5  50   0 23.0 101 192 125.4  52   4 1.86  80  135
6  23   0 22.6  89 139  64.8  61   2 1.82  68   97
```

The summary table created by $fcn.dqa1.matrix()$ is

```
> fcn.dqa1.matrix(diabetes2)
```

```
         age sex         bmi         map          tc         ldl         hdl
n        442 442 442.0000000 442.0000000 442.0000000 442.000000 442.0000000
mean      NA  NA  26.3757919  94.6466063 189.1402715 115.439140  49.7884615
sd        NA  NA   4.4181216  13.8319998  34.6080517  30.413081  12.9342022
skew      NA  NA   0.5940948   0.2886392   0.3755457   0.433633   0.7938385
kurtosis  NA  NA   3.0665551   2.4483946   3.2022380   3.564870   3.9390113
                tch         ltg         glu        prog
n        442.000000 442.0000000 442.0000000 442.0000000
mean       4.070249   2.0157466  91.2601810 152.1334842
sd         1.290450   0.2270465  11.4963347  77.0930045
skew       0.730390   0.2884487   0.2065075   0.4375772
kurtosis   3.410362   2.8405942   3.2061443   2.1038045
```

The statistics for the *age* and *sex* variables are missing because of the single cases being missing.

R functions can accommodate missing values by including the function argument:

$$na.rm = TRUE$$

With most functions, if this argument is added, the computations exclude the missing values giving useful statistics for the data set. For example, the basic summary statistics function *summary*() gives the following output when applied to the first 4 columns of the data set *diabetes*2:

```
> apply(diabetes2[,1:4],2,summary,na.rm=TRUE)

$age
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  19.00   38.00   50.00   48.49   59.00   79.00       1

$sex
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
 0.0000  0.0000  0.0000  0.4694  1.0000  1.0000       1

$bmi
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  18.00   23.20   25.70   26.38   29.28   42.20

$map
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  62.00   84.00   93.00   94.65  105.00  133.00
```

3. **Fast(er) Computations of Exponential Moving Average**[1]

For a series: $x_i, i = 1, \ldots, n$, the exponential moving average of x, is given by the following recursive fomula:

$$y_t = \begin{cases} x_1 & , \quad t = 1 \\ \alpha x_t + (1 - \alpha) y_{t-1} & , \quad t \geq 1 \end{cases}$$

Implement the following versions of the exponential moving average, and compare their performance using the R function *microbenchmark*()

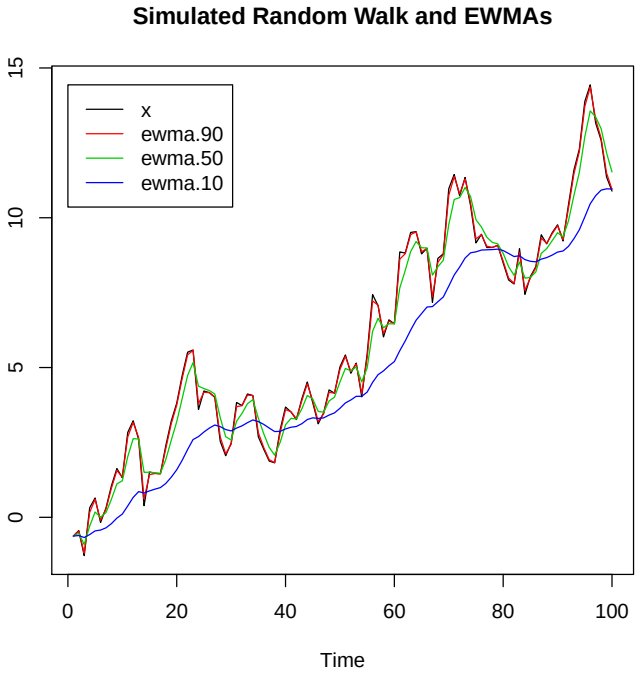
3(a) Simple loop function:

```
> ewma <- function(x, alpha) {
+   n <- length(x)
+   s <- rep(NA,n)
+   s[1] <- x[1]
+   if (n > 1) {
+     for (i in 2:n) {
+       s[i] <- alpha * x[i] + (1 - alpha) * s[i-1]
+     }
+   }
+   return(s)
+ }
```

Consider plotting the ewma of a 100-step normal random walk

```
> set.seed(1)
> # initialize random number generator (for reproducibility)
> x=cumsum(rnorm(100))
> x.ewma.90<-ewma(x,alpha=.90)
> x.ewma.50<-ewma(x,alpha=.50)
> x.ewma.10<-ewma(x,alpha=.10)
> ts.plot(ymat0<-cbind(x=x,
+                ewma.90=x.ewma.90,
+                ewma.50=x.ewma.50,
+                ewma.10=x.ewma.10), col=c(1:4),
+         main="Simulated Random Walk and EWMAs")
> legend(x=0, y=max(as.vector(ymat0)),
+        legend=dimnames(ymat0)[[2]],
+        col=c(1:4),
+        lty=rep(1,times=4))
```

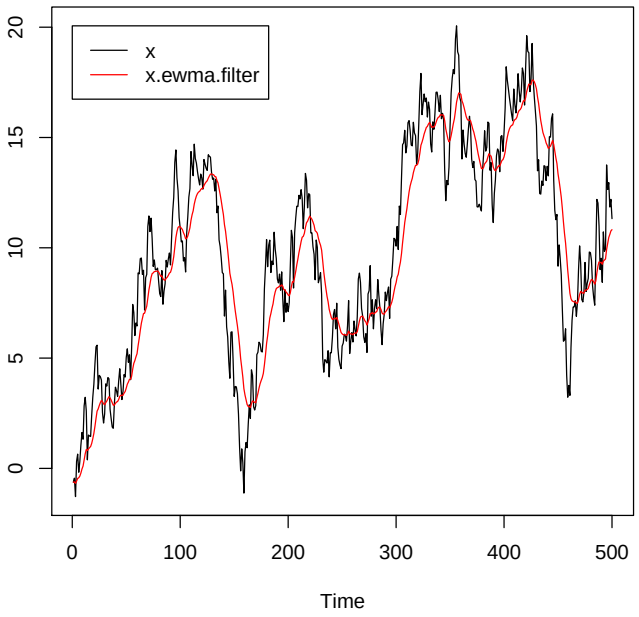
[1] This problem is drawn from a question/answers on *stackoverflow.com* :
https://stackoverflow.com/questions/42774001/fast-r-implementation-of-an-exponentially-weighted-moving-average



Re-do the plot with a random walk length of $n = 500$ steps.

3(b) The computation of the exponential moving average can be slow because of the loop in the function $ewma()$. An alternative is to apply the R function $filter()$

```
> ewma.filter<-function(x,alpha){
+   result<-c(filter(x*alpha, 1-alpha,method="recursive", init=x[1]))
+   return(result)
+ }
> ts.plot(ymat<-cbind(
+     x=x,
+     x.ewma.filter=ewma.filter(x,0.1)),
+     col=c(1,2))
> legend(x=0,
+       y=max(as.vector(ymat)),
+       legend=dimnames(ymat)[[2]],
+       col=c(1:ncol(ymat)),
+       lty=rep(1,times=ncol(ymat)))
>
```



Use the function *all.equal*() to test the equality of the output from these two functions:

```
> #all.equal(ewma.filter(x,alpha=.9), ewma(x,alpha=.9))
```

Use the function *microbenchmark*() to compare the two functions

```
> microbm.1<-microbenchmark::microbenchmark(ewma(x,alpha=.9), ewma.filter(x,alpha=.
> # Read the help function for microbenchmark, and
> # print out/plot the results:
> #        print(microbm.1)
> #         plot(microbm.1)
> #         library(ggplot2)
> #         autoplot(microbm.1)
> # Note: the function autoplot() requires the library ggplot2
> #       you need to install the package before executing autoplot()
> #   install.packages("ggplot2")
```

3(c) R has a byte code compiler which compiles an expression into a byte code object. This operation can significantly increase the speed of computations.

```
> ewma.cmpfun <- compiler::cmpfun(function(x, alpha) {
+   n <- length(x)
+   s <- rep(NA,n)
+   s[1] <- x[1]
```

```
+   if (n > 1) {
+     for (i in 2:n) {
+       s[i] <- alpha * x[i] + (1 - alpha) * s[i-1]
+     }
+   }
+   return(s)
+ })
>
```

Use the function *all.equal*() to test the equality of the output from this function and two previous versions:

```
> #all.equal(ewma(x,alpha=.9), ewma.cmpfun(x,alpha=.9))
> #all.equal(ewma.filter(x,alpha=.9), ewma.cmpfun(x,alpha=.9))
```

Use the function *microbenchmark*() to compare the three functions

```
> microbm.2<-microbenchmark::microbenchmark(
+     ewma(x,alpha=.9),
+     ewma.filter(x,alpha=.9),
+     ewma.cmpfun(x,alpha=.0))
>
>
> # As before,
> # print out/plot the results:
> #         print(microbm.2)
> #         plot(microbm.2)
> #         library(ggplot2)
> #         autoplot(microbm.2)
```

Note: The R package *Rcpp* allows functions to use $C++$ in R. The following code should create the R function ewmaRcpp:

```
# If using for the first time, install the Rcpp package
#   install.packages("Rcpp")
    library(Rcpp)

sourceCpp(  code =
    "
     #include <Rcpp.h>
     // [[Rcpp::export]]
     Rcpp::NumericVector ewmaRcpp(Rcpp::NumericVector x, double alpha){
       int n = x.length();
       Rcpp::NumericVector s(n);
       s[0] = x[0];
       if (n > 1) {
         for (int i = 1; i < n; i++) {
           s[i] = alpha * x[i] + (1 - alpha) * s[i-1];
         }
       }
       return s;
     }
")
```

(The above code executed successfully in the RStudio code edit panel, but would not compile as part of a sweave document. If you want to use Rcpp, you may seek help during office hours or consult *stackoverflow.com*.)

4. **Independent Study of R**

   There are many useful resources on the web for learning R, including

   - "An Introduction to R":
     https://cran.r-project.org/doc/manuals/R-intro.pdf
   - A free introduction to R from *datacamp.com*
     https://www.datacamp.com/courses/free-introduction-to-r
   - Hadley Wickham's online book *Advanced R*
     http://adv-r.had.co.nz/

   Choose one of these (or any other you might find/choose) and create a 1-2 page write-up suitable for sharing with your classmates to help them learn R.