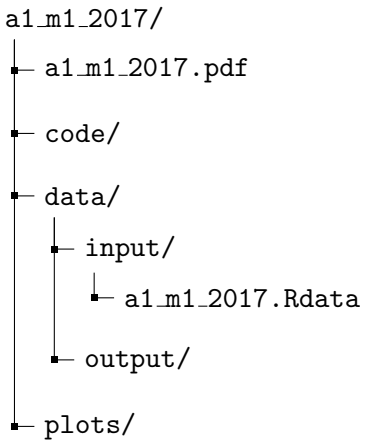
2017 PSS SUMMER SCHOOL
Assignment 1

Due: 12:30pm on June 22

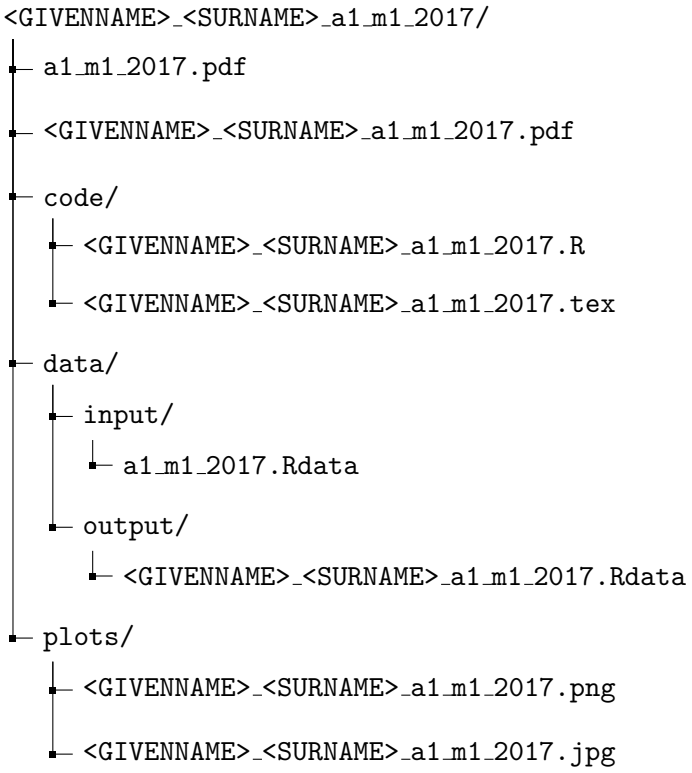
1 Submission instructions

To submit your assignment, email all of your files in a single zip folder to `pss.ss.hw.submit@sp.frb.gov`. Please using the naming convention `<GIVENNAME>_<SURNAME>_a<NUMBER>_m1_2017.<FILEXTENSION>` for any **file that you edit or create**, for files that you simply use without modifying, such as the assignment directions or a data file, leave the name as is, usually something like `a1_m1_2017.pdf`. The below directory comparison shows you an example of the zip directory you will receive and the one you will email submit after completion.

(zip to students)



(zip from students)



Where you make the proper substitutions, e.g. `JUSTIN_SKILLMAN_a3_m1_2017.zip`. Remember that you don't need to change the names of the files you do not modify, such as `a1_m1_2017.Rdata`, only the files that you create or modify. Your code for each assignment example would be saved under `code/` in a `.R` file, named as shown above.

Input/Output from R

Below are some basic commands you should put in your script

```
load(file=[filename])
```

```
[fullpathfilename] <- paste0([path_to_file],[filename])
save([variables]*, file=[fullpathfilename])
```

For future reference:

```
df <- read.csv([filename])
write.csv(df, file=[filename])

library(foreign)
df <- read.dta([filename])
write.dta(df, file=[filename])
```

List Submission

Please submit all answers in a list data structure. Below is a basic example, instead of *result* use the name given in each problem:

```
> answers <- list()
> result <- 10 + 15
> answers$q1a <- result

> answers$q1a
[1] 25
```

We start out by generating an empty list to eventually store all answers in. Now let us say that we calculated the answer to question q1a and stored it in the variable result. We then add the result value to the answers list as shown above.

2 LaTeX

Problem 2.1. Create a `.tex` document named following the conventions described above. For each command in the couple of blocks below, write a sentence describing its functionality. Be explicit with the type of its inputs and outputs.

(a) `str()`, `is.vector()`, `is.atomic()`, `is.list()`

(b) `typeof()`, `is.numeric()`, `is.integer()`, `is.double()`, `is.character()`, `is.logical()`.

In your LaTeXwrite up use different formatting, e.g. see the commands `\texttt{}`, `\textit{}`. Compile this and save the resulting `.pdf` in the parent directory.

3 Data Structure

Load the data file `a1_m1_2017.Rdata` for the following questions. For each question, save the answer to a named list data structure called *answers* where the name corresponds to the question. For example the result of question 1a can be retrieved by calling answers$q1a. As a reminder, save your *answers* list in a .Rdata file in the `data/output/` directory.

Vectors

Problem 3.2. Use v1-v4 from the loaded data to:

(a) Determine types of vectors for v1-v4, save as a vector of length four called *vec_types* into answers$q3.2a

(b) Determine the number of elements in each vector and find the last element in each vector. Save these results along with *vec_types* in a vector of length 12 of the form {`length of vector 1,last element in vector 1, type of vector 1,...`} named *vec_info*

Problem 3.3. Create vectors with the following properties

(a) A character vector named *fed_vec* of length 450 where each element is a random draw from the set of elements of a character vector containing "federalreserve", where each element of the vector should be one character

(b) A numeric vector of length 500 that is bimodal normal with means 5 and -5 and standard deviation 2 named *bimodal_vec*

(c) The previous vector coerced into an integer *bimodal_int_vec*

Lists

Problem 3.4. The following questions use l1 from the loaded data

(a) Return the 17^{th} element in l1 and save this as *seventeenth*

(b) Return the element corresponding to the name "boat" and save this as *boat_name*

Problem 3.5. Create lists with the following properties

(a) A list where the elements are the characters of the alphabet in order called *alpha*

(b) A two element named list called *two_list* where "five" corresponds to a vector of five zeros, and "multi" corresponds to a character vector spelling "yellen"

(c) A list of length 2 named *equal_methods* where the elements use *two_list* to return the letter "y" using named indexing and numeric indexing

Matrices

Problem 3.6. The following questions use m1-m2 from the loaded data

(a) Determine the dimensions of m1, save as *dims*

(b) Find the bottom right corner element of m1, save as *br_corn*

(c) Find the subset of m1 with only the first five rows, save as *m1_subset*

(d) Calculate the result of the following element-wise operations, save as *elem_mult*

$$\frac{m1 + m2}{m2} - m1 * m2 \tag{1}$$

(e) Perform matrix multiplication of m1 by m2, save as *matrix_mult*

Dataframes

Problem 3.7. The following questions will use df1 from the loaded data

(a) Determine the number of rows in the dataframe, save as *rows*

(b) Return the names of the columns in the dataframe, save as *cols*

(c) Return the column *a*, save as *col_a*

Problem 3.8. The following questions will be based off the given trading log records

Table 1: Trading log records.

date	sender	receiver	parval	bonds
2014-01-05	Matt	John	100.00	8
2014-05-05	Jake	Mike	350.00	6
2014-08-01	Lauren	Joe	140.50	15
2014-10-17	Jill	Alice	1050.25	2

(a) Create a dataframe named *trade_log* with the data above (be sure to include the same variable names), making sure to format the date column as shown in Table 1 into an R-date type column

(b) Create a new (larger) dataframe named *trade_log_improved*, that adds a column *total* as the product of *parval* and *bonds* to *trade_log*

Problem 3.9. Please note that answers$q3.9a, answers$q3.9b, answers$q3.9c, answers$q3.9d will contain dataframes of the same name where for example the first is a dataframe full of NAs and the second has one of the columns populated with dates.

(a) Create a dataframe named *sample_df* of NAs that is 5,000 $\times$ 5 in size

(b) Create a column, *date*, that randomly samples from the period 01/01/1900 to 01/01/2016 (make sure you only use "real dates", for example 2/29/2015 did not happen)

(c) Create a vector of character strings that are three randomly chosen letters each as the second column, called *name*

(d) Map each letter A:Z to 1:26 and create a third column that sums the three letters in the second column, called *name_sum*

4 Control Flow and Functions

Functions

For all solutions in this answer, save the function into the list directly, e.g. `answers$q4.10b <- dist1`

Problem 4.10.

(a) Create a function *date_diff(x)* which calculates the difference in number of *business days* (non-weekend days, ignoring holidays) between a given date, x, and the beginning of the month in which that date is in. For example Friday May 20th 2016 is **15** business days after Sunday May 1st 2016. Please program conditional tests that will return a message to the user if the input to your function are not of the R date type (*Hint: use* `if else` *statements and the* `print` *command*).

(b) Create a function *dist1(a,b)* that takes in two coordinates (represented as two element vectors) and returns the euclidean distance between them. As a reminder the distance for a pair of dimensional coordinates (x_1, x_2) and (y_1, y_2) is:

$$D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

for and while loops

Problem 4.11. We will now define function *dist2(a,b,dims,method)* to extend our function *dist1(a,b)* to n dimensional coordinates. The euclidean distance formula can be extended to n dimension as follows:

$$D = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

The function should now be able to take two n-dimensional numeric vectors and find the distance between them. The *method* input to this function should be either method=loop or method=vectorize, loop tells your function to conduct a for loop and vectorize tells your function to use vector addition/subtraction and the linear algebra dot product to compute D.

if else statements

Problem 4.12. Define a function *sqr_mat(z)* where z is an square matrix **or** vector of length n, where $\sqrt{n}$ is an integer. You need to test for these conditions, if neither are met return a message to the user reporting it and exit the process. If given a square matrix your function should return a vector of the elements stored in column-major order (see below). If given a vector your function should return an square matrix similarly assuming column-major order. Below is a schematic of how the function translates objects.

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \quad \rightarrow \quad \{a, d, g, b, e, h, c, f, i\}$$

or

$$\{a, d, g, b, e, h, c, f, i\} \quad \rightarrow \quad \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

5 Bonus

The following question will push you to think hard about complex programming topics that are common in the current research environment. For those that successfully complete this problem, bonus points will be awarded on the assignment. Submit the answers to the bonus question in a separate .R file appending `_bonus` at the end of the filename.

Problem 5.13.*

(a) Following the algorithm below, create a function *my_sort(x)* that sorts a vector x of *distinct* integers of length n.

```
1: procedure MY_SORT
2:     i ← 1
3:     if x_i > x_{i+1} then
4:         move x_{i+1} to the first position in the vector
5:         return to ln. 2
6:     else
7:         if i + 1 ≠ n then
8:             i ← i + 1
9:             return to ln. 3
10:        else
11:            exit the procedure, the list is sorted
```

As an example, the vector $\{8, 2, 1, 4\}$ would be sorted in the following way.

<u>8 2</u> 1 4	8 and 2 are out of order, move 2 to the first position and start at the beginning
<u>2 8</u> 1 4	2 and 8 are in order, move to next pair
2 <u>8 1</u> 4	8 and 1 are out of order, move 1 to the first position and start at the beginning
<u>1 2</u> 8 4	1 and 2 are in order, move to next pair
1 <u>2 8</u> 4	2 and 8 are in order, move to next pair
1 2 <u>8 4</u>	8 and 4 are out of order, move 4 to the first position and start at the beginning
<u>4 1</u> 2 8	4 and 1 are out of order, move 1 to the first position and start at the beginning
<u>1 4</u> 2 8	1 and 4 are in order, move to next pair
1 <u>4 2</u> 8	4 and 2 are out of order, move 2 to the first position and start at the beginning
<u>2 1</u> 4 8	2 and 1 are out of order, move 1 to the first position and start at the beginning
1 2 4 8	list is in order, complete

(b) Let $C(x)$ be the count of how many times *ln. 4* occurs, e.g. when the smaller element is moved to the first position. In our example, $C(\{8, 2, 1, 4\}) = 6$. Also let $\mathbb{E}[C(P_x)]$ be the expectation of $C(P_x)$ where P_x is the set of all permutations of the vector x. Intuitively we want to find out how many swaps would be necessary, on average, regardless of the starting order of the vector. Please note that $C(P_x)$ is then a set of counts, over which you are taking the expectation. Use your function from (a) to create a function *exp_my_sort(x)* that outputs this expectation when given any vector x, as described above.

(c) Implement another sorting algorithm as shown below. Similarly as above your function *our_sort* should take in some vector x and output a sorted vector.

```
1: procedure OUR_SORT
2:     swap ← 0
3:     i ← 1
4:     if x_i > x_{i+1} then
5:         switch the positions of x_i and x_{i+1}
6:         i ← i + 1
7:         swap ← 1
8:         return to ln. 4
9:     else
10:        if i + 1 ≠ n then
11:            i ← i + 1
12:            return to ln. 4
13:        else
14:            if swap = 0 then
15:                exit the procedure, the list is sorted
16:            else
17:                return to ln. 2
```

(d) Compare how well *my_sort* and *our_sort* do over a number of dimensions: size of the input vector, complexity of the input vector (how "out of order" it is), computer resource usage (RAM & CPU), and any other dimension you think is helpful. Please include some quantitative data (perhaps some timed trials) presented in a concise manner. Also write some of your thoughts on how the two functions compare, why do you think one might be better than the other?