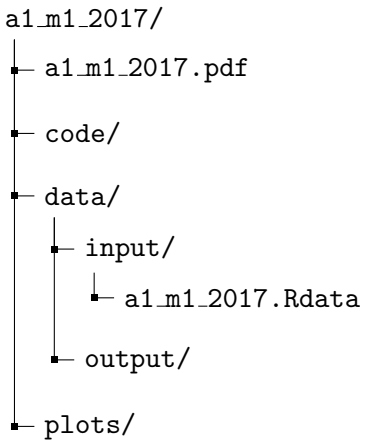
2017 PSS SUMMER SCHOOL
Assignment 3

Due: 12:30pm on June 28

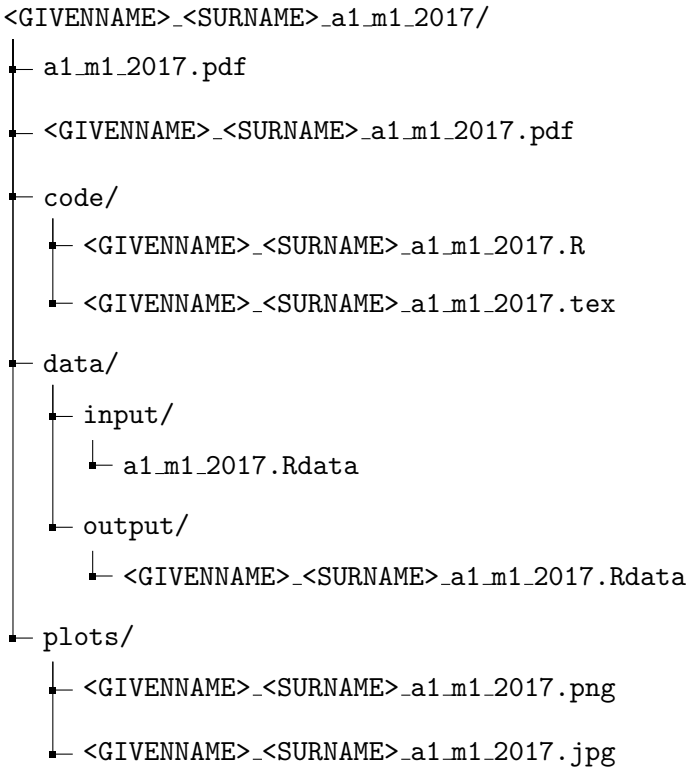
1 Submission instructions

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 Introduction

The goal of this assignment is to familiarize yourself with data manipulation functions to subset and transform data. You will also gain experience with the very common task of combining different datasets.

Our task for this assignment is to determine the effect of weather on demand for bikeshare services. We have two datasets at our disposal, one detailing the number of riders for the bikeshare system over time and another showing the weather over time. Using the data manipulation techniques that we have learned up to this point we will answer this question with the data provided.

3 Data Description

Bikeshare Usage Data

The bikeshare data for this assignment is derived from the publicly available bikeshare trips data located on the Capital BikeShare website. This dataset contains historical information on individual rides since the inception of the system. We have aggregated the raw data to save you some time in cleaning it.

Data Fields

date - Date in native R format

hour - Hours from midnight

type - Either Casual or Subscriber

mduration - The mean duration of trips taken in seconds

ntrips - number of trips taken in the hour interval

DC Weather Data

This dataset was compiled from Weather Underground, a site which maintains historical weather data for locations around the country. The dataset shows weather variables for the DC area (zip code 20001 in specific). The weather is observed in similar intervals throughout the day.

Data Fields

date - Date in native R format

hour - Hour from midnight

temp - Temperature in Fahrenheit

windspeed - Wind speed in MPH

precip - Inches of precipitation

4 Summary Statistics

When exploring a new dataset it is very common to first calculate summary statistics to get a first look. This step usually involves calculating measures such as mean, standard deviation, minimum, and maximum of a given variable. We will do this for both the weather and bikeshare datasets described above.

Bikeshare Data

Problem 4.1.

(a) Remove observations from years which do not have a whole years worth of data (2010 Q4, 2014 Q4). Save this as `sub_whole_years_df` in `answers`.

(b) Create a data frame which aggregates the count of the number of trips taken for each full year split out by the type of rider. This dataframe should look something like the following and should be saved as `agg_trips_by_ridryr` in `answers`.

type	year	ntrips
...		

(c) Reshape the dataframe from the previous problem to wide form such that the years are displayed as columns and the rows are the different types. The dataframe should look something like the following and should be saved as `cas_reg_ridr` in `answers`.

type	2011	2012	2013
Casual			
Registered			

(d) Repeat problem 4.1.b and 4.1.c but using the variable mduration instead of ntrips. Additionally when we aggregate we want to find the mean of the observations as opposed to the sum in this case. Save the resulting dataframes in a list named similarly to the above, save this list into `answers` under `$q41d`.

(e) To determine the seasonality of rider-ship, create a dataframe which counts the number and mean duration of trips which occur in each season. The dataframe should look something like the following and be saved as `season_df` in `answers`.

measure	type	Spring	Summer	Fall	Winter
ntrips	Casual				
ntrips	Registered				
mduration	Casual				
mduration	Registered				

Weather Data

Problem 4.2.

(a) We would like a binary variable to tell us whether it is raining or not. Create variable `is_raining` to have value 1 if percep has value greater than 0 and 0 otherwise.

(b) Ensure that each (date, hour) combination is unique for the weather dataset since we only want one weather observation per hour. To aggregate multiple observations you can take the mean of temp, windspeed, and precip and the max of `is_raining`. (Hint: Remove NA values for calculations by using the na.rm argument). Save this dataframe with unique rows as `uniq_df`.

(c) Create a dataframe which summarizes all of the weather measurement variables (temp, windspeed, precip, is_raining). For each variable calculate the median, mean, standard deviation, minimum, and maximum values. The resulting dataframe should look something like the following and be saved as `summarize_df` in `answers`.

variable	median	mean	stdev	min	max
temp					
windspeed					
precip					
is_raining					

5 Effect of Weather on Ridership

Linear Regression

Problem 5.3. Now that we understand the data better from creating summary statistics in the

previous section, we are now ready to tackle our main question. To do so we will first merge the two datasets together. Then we will summarize the data to show trends on the effect of weather on bikeshare usage.

(a) Merge the two datasets (bikeshare and weather) on the date and hour variables which both have in common. Save this as `merged_df` in `answers`.

(b) Now that we have the data in the form that we want we can run a model to look at the relationship between our two variables of interest. Run the following linear regression model. Save the object returned from the lm function in your answers list, name it `lm_res`.

$$ntrips_t = \alpha + \beta_1 temp_t + \beta_2 windspeed_t + \beta_3 precip_t + \beta_4 israining_t + \epsilon_t$$

6 Bonus

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

Problem 6.4.*

Fuzzy Search: Sometimes, you will be tasked with matching bilateral transaction data from a database (such as Fedwire or CHIPS). Often, the origin and purpose of these transactions will not be known to you as the researcher.

For this exercise, you will simulate a fuzzy search matching algorithm to find pairs of corresponding sending leg payments and return payments .

To establish a match, you must find a return payment that has a one day maturity length, and which falls within 150 basis points of a simulated interest rate and on a business day. Please be sure to take into account business days when finding matches. For example, a sending leg send on Friday and a match found Monday is still considered overnight. To achieve this goal, construct three synthetic data frames:

1. Simulated mean interest rates for each maturity length for each business day in the range of January 1, 2007 to January 1, 2016. The dataset will have two columns: date, overnight interest rate. To simplify the exercise, do NOT annualize the rate.

2. Simulated sending legs (with lender id, borrower id, amount, and date). All amounts should be round to the nearest ten thousand and be larger than 1 million.

3. Simulated return legs (with lender id, borrower id, amount and date). all Amounts should *not* be round to the nearest ten thousand and be larger than 1 million.

Transaction dates in the latter two data frames should fall from January 1, 2007 to January 1, 2016 and should be assigned randomly. The goal is to find "matches" from the return leg dataset to the sending leg dataset. To do this, you will need to compute all possible interests rates for all possible combination of matches that fall within the overnight range. Compute all interests for those remaining combinations. Then, keep only matched loans that are within 150 basis points of the mean interest rate for the day that the sending leg sent out the loan.

In case of multiple return legs matching to a single sending leg or vice versa, you may select one observation from this group at random as a tiebreaker.

The exercise asks that you return a data frame of matched pairs with send amount, return amount, send date, return date, implied interest rate by the matches, and maturity length (in days).

Hint: Refer to the apppendix of Furfine et al for more information on this type of exercise. That example is more tailored to a specific dataset, but it lays out the general algorithm succinctly. You may find it a useful a useful resource. In addition, This questions requires the *timeDate* and *bizdays* packages.

Problem 6.5.*

Using the algorithm from *Problem 6.4*, simulate 2 billion total transactions (sending and return legs combined). Answers must be returned in under 20 minutes to receive full credit. You may use multiple loops in this exercise so long as the answer is returned in under 20 minutes.