Lesson 3: Data Frames

Today:

1. Data frames
    - Useful functions: `dim()`, `names()`, `head()`
    - Getting datasets: (1) built-in datasets; (2) datasets in your computer or from the internet using `read.csv()`
    - Accessing columns of a data frame
    - Accessing entries of a data frame
2. Working with data frames
    - Arithmetic with lists and columns
    - More useful functions: `sum()`, `length()`, `max()`, `min()`
    - Finding average of a column using `sum()` and `length()`; using `mean()`
    - Finding proportions/percentages

1. Data Frames

Data Frames are essentially what R calls **tables of data** (similar to Excel spreadsheets).

- Each **column** of a data frame corresponds to a **variable**
- Each **row** of a data frame corresponds to one **observation**/one **individual**.

In our first class, we saw some examples of data. One of our examples was the UC Berkeley 1973 graduate admissions data below.

Questions:

- How many variables are there? How many observations?
- What does each observation correspond to?

In [ ]:
```
berkeleydata <- read.csv("berkeley73.csv")
berkeleydata
```

In this example, each row corresponds to a department and each column is a variable about that department. For example, the number of women applicants, the number men applicants, etc. are all variables.

How do we get our data to be "imported" to R as a data frame? Where do data frames that we work on in R come from?

There are three basic ways we can get data frames in R.

1. Built-in datasets that comes with R
2. Importing a file from a directory in you computer or from the internet
3. Entering data manually into a data frame

1. Built-in datasets

   R comes with a few "built-in" data sets. This makes it easier for beginners who want to start working with data sets that many people have found interesting and worth studying. We will look at a couple of these today.

1. Importing a file from a directory in your computer or from the internet

   However, there are a lot of data out there and often the data we want to work with will not be the ones that automatically come with R. Therefore, we would need to **import** the data as a data frame in R that we can work with. We will learn how to do this today.

1. Entering data manually into a data frame

   Ocasionally, we might want to type in our data manually into a data frame (for example, maybe you are collecting the data yourself. We will learn how to do this next class (Lesson 4).

1.1. Built-In Datasets

R comes with some datasets that are ready for us to explore. They come in an R package called `datasets`.

At this point, this `datasets` package is a bit of a mystery. What are in it?

To find out further information about any R packages, you can type

```
? packagename
```

and run the cell. Try it below.

*(A note about **R packages**: People around the world develop new R packages all the time. You can think of them as useful toolboxes that you can use when you find them convenient. To tell your jupyter notebook that you want to use an R package, you need to type `library('packagename')` and run the cell; your package will then be loaded and ready to use. We do not need to do this with the `datasets` package because it's built-in and already loaded automatically.)*

In [ ]:
```
? datasets
```

You should see a message that you can find the list of all data sets by typing and running

```
library(help = 'datasets')
```

Do it in the following code cell.

In [ ]:
```
library( help = 'datasets' )
```

You will see that one such built-in datasets is the `faithful` dataset.

In [ ]:
```
faithful
```

In [ ]:
```
# To see a description of what the variables (columns) are, type `?faithful`
#    (recall that in general, typing ? followed by the R command/dataset will
 tell you further information about that command/dataset.)
?faithful
```

1.2. Importing a file from a directory in your computer or from the internet

We have seen the UC Berkeley 1973 graduate admissions dataset a few times now.

This dataset comes from a **"comma separated file" (CSV)** called `berkeley73.csv` that is stored in our lesson03 folder in our JupyterHub (the same folder this Jupyter notebook is in).

We use the function

```
read.csv( 'FILENAME' )
```

when we want R to "read" a csv file and store it as a data frame:

- input: the name of the csv file
- output: an R data frame

In [ ]:
```
berkeleydata <- read.csv( 'berkeley73.csv')
```

Working with files in a different folder

You might notice that we have a copy of `berkeley73.csv` in each of the folders: lesson01, lesson02, and lesson03 folder. This seems inefficient!

From now on, datasets that we will work with will be stored in the following folder:

```
/class_share/datasets/FILENAME
```

So, we will have just that one copy of the csv file that we can refer to from anywhere else in our JupyterHub.

Exercise

In the `/class_share/datasets` folder, there is a file called `NYC_Dog_Licensing_small.csv` .

In the code cell below, please import that dataset into an R data frame named `nyc_pups` .

```
In [ ]: read.csv( '../../../class_share/datasets/NYC_Dog_Licensing_small.csv' )
```

This dataset comes from NYC Open Data, a program that makes NYC government data available for the public.

The following is the description for the above dataset (https://data.cityofnewyork.us/Health/NYC-Dog-Licensing-Dataset/nu7n-tubp): "All dog owners residing in NYC are required by law to license their dogs. The data is sourced from the DOHMH Dog Licensing System (https://a816-healthpsi.nyc.gov/DogLicense (https://a816-healthpsi.nyc.gov/DogLicense)), where owners can apply for and renew dog licenses. Each record represents a unique dog license that was active during the year, but not necessarily a unique record per dog, since a license that is renewed during the year results in a separate record of an active license period. Each record stands as a unique license period for the dog over the course of the yearlong time frame."

The original NYC Dog Licensing dataset (https://data.cityofnewyork.us/Health/NYC-Dog-Licensing-Dataset/nu7n-tubp) datasets is very large (more than 300,000 rows of data). The above smaller dataset contains 1000 randomly chosen rows from the original dataset.

We could also import a csv file found directly from the internet using

```
read.csv(  url( 'LINK TO THE CSV FILE' ) )
```

Example
We will import a dataset from the NY State governement open data. This page contains description of a dataset of a directory of criminal justice agencies: https://data.ny.gov/Public-Safety/Directory-of-Criminal-Justice-Agencies/gugp-n5ip (https://data.ny.gov/Public-Safety/Directory-of-Criminal-Justice-Agencies/gugp-n5ip)

- Click on the above link
- Click on the `Export` button on that page
- Right click on the `CSV` button
- Choose "copy link address"

Read the csv file linked from that address and store it as an R data frame called `criminal_justice_agencies` .

In [ ]:
```
criminal_justice_agencies <- read.csv( url( 'https://data.ny.gov/api/views/gugp-n5ip/rows.csv?accessType=DOWNLOAD' )  )
```

In [ ]:
```
criminal_justice_agencies
```

1.3. Useful functions for working with data frames

As with lists, there are some basic functions that are useful for working with data frames. These are:

- `dim( DATAFRAMENAME )` : to find the number of rows and columns in a data frame
- `names( DATAFRAMENAME )` : to find the column names of a data frame
- `head( DATAFRAMENAME )` : to preview the first few rows (ten) of a data frame.

  If you want to see the first n rows (where n is any number of your choice), you can run `head(DATAFRAMENAME, n)` .

Try these functions in the code cells below.

In [ ]:
```
dim(berkeleydata)
names(berkeleydata)
head(berkeleydata, 3)
```

1.4. Accessing a column of a data frame

Each column of a data frame is simply a list!

Given a data frame, to obtain a list containing just one of its columns is easy, and there are two ways to do this.

1. `DATAFRAMENAME$COLUMNNAME` : Gives you a list containing all entries in the column `COLUMNNAME` of the data frame `DATAFRAMENAME`
2. `DATAFRAMENAME[, COLNUM ]` : Gives you a list containing all entries in column number `COLNUM` of the data frame `DATAFRAMENAME`

In [ ]:
```
berkeleydata$Men_Applicants
```

In [ ]:
```
berkeleydata[,2]
```

1.5. Accessing an entry in a data frame

There are also two ways to access an entry in a data frame

1. `DATAFRAMENAME$COLUMNNAME[ ROWNUM ]` : To access an entry in row number `ROWNUM` and column called `COLUMNNAME` :
2. `DATAFRAMENAME[ ROWNUM, COLNUM ]` : To access an entry in row number `ROWNUM` and column number `COLNUM` :

Example
To access the number of women applicants in department F (row 6, column 4):

In [ ]:
```
berkeleydata$Women_Applicants[6]
```

In [ ]:
```
berkeleydata[6, 4]
```

2. Working with data frames

2.1. Arithmetic with lists and columns

Recall the data frame `berkeleydata` from before. Suppose that we would like to compute the admission rates of men and women in each of the departments.

In [ ]:
```
berkeleydata
```

For example, to compute the admission rate of women in each department, we want to divide each number in the `Women_Admitted` column by the corresponding number in the `Women_Applicants` column.

In [ ]:
```
berkeleydata$Women_Admitted / berkeleydata$Women_Applicants
```

2.2. Adding a new column to a data frame

```
DATAFRAMENAME$NEWCOLUMNNAME <- list containing entries for the new column
```

Example Suppose that we would like to create a new column called `Men_AdmissionRate` in the `berkeleydata` dataframe.

In [ ]:
```
berkeleydata$Men_AdmissionRate <- berkeleydata$Men_Admitted / berkeleydata$Men_Applicants

berkeleydata
```

Exercise

- Create a new column called `Total_Admitted` , which consists of the total number of men and women addmitted to each department.
- Create a new column called `Overall_Admission_Rate` , which consists of the overal admission rate (men and women) admitted to each department.

In [ ]:
```
berkeleydata

# add a new column called Total_Admitted
# ...

# add a new column called Overall_AdmissionRate
# ...
```

2.3. Finding sums and averages of a column

Recall that the function `sum( LISTNAME )` takes the sum of the numbers in the list `LISTNAME` .

Since a column in a data frame is simply a list, we can use `sum()` for a column of a data frame.

Example
Compute the total number of admitted women across the six departments.

In [ ]:
```
sum(berkeleydata$Women_Admitted)
```

We can do arithmetic operations to columns of a data frame.

Example
Compute the average number of admitted women across the six departments.

In [ ]:
```
sum(berkeleydata$Women_Admitted)/ length(berkeleydata$Women_Admitted)
```

In [ ]:
```
mean(berkeleydata$Women_Admitted)
```