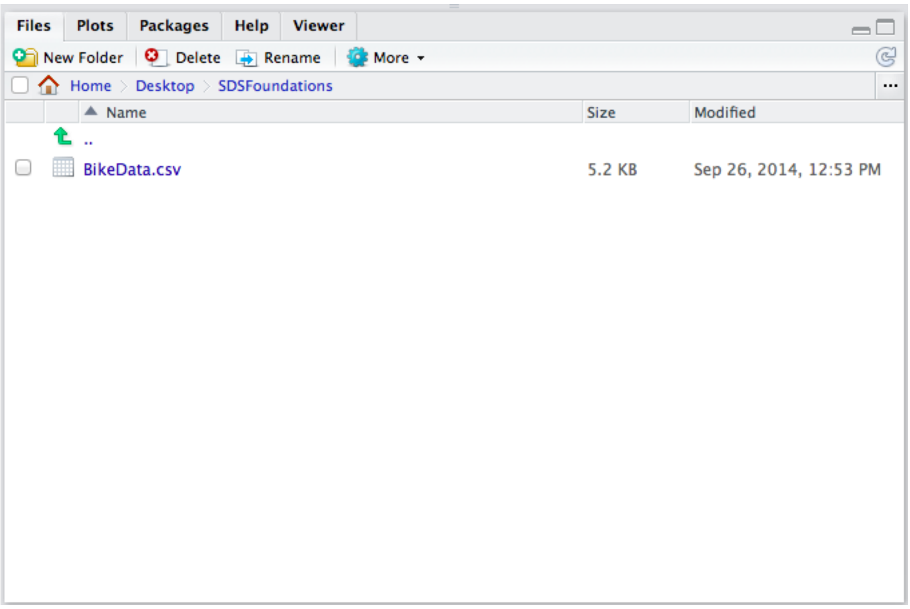
Foundations of Data Analysis

The University of Texas at Austin

R Tutorials: Week 1

Importing a Data Frame

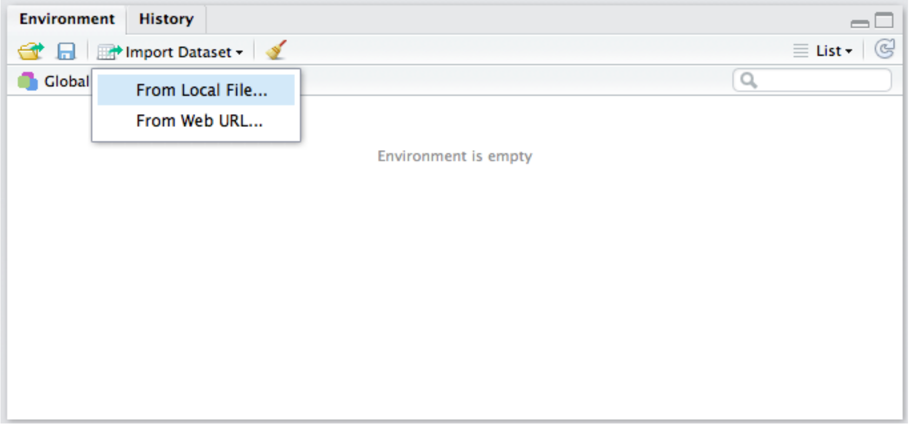
In this R tutorial, we will learn how to import a data set into R Studio. This is really important because most the time, you're not going to be collecting your data directly into R Studio. You'll have it in a spreadsheet, and in order to do the analysis, you need to bring that data into R Studio. So I have an example data set here called "BikeData.csv".



Name	Size	Modified
..		
BikeData.csv	5.2 KB	Sep 26, 2014, 12:53 PM

Figure 1: Working Directory.

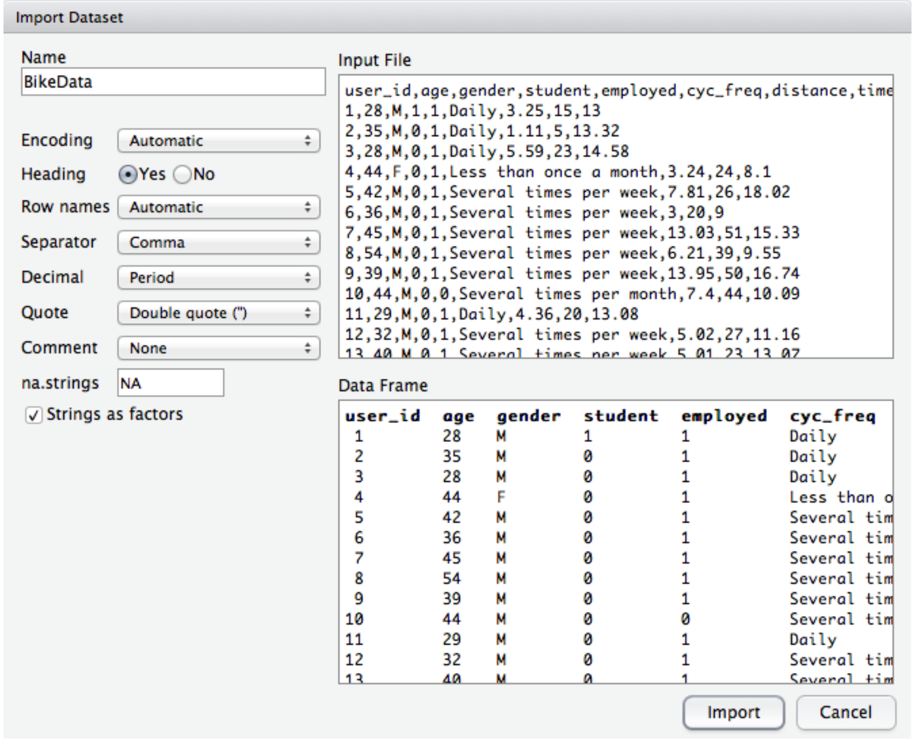
Now, CSV is the file extension that most easily is imported into R. If you have your data in Excel and it's saved as a .XLS extension or anything like that, just go ahead and go into Excel, save as a CSV. It makes it much easier to import the data into R Studio if it's in that CSV format. The next step to take is to come up to the Environment or Workspace window and import the data set into R Studio. That's really easy. You just click this Import Data Set button.



Environment is empty

Figure 2: Importing Data in RStudio.

It's from a [CSV] file. A CSV is just another version of a text file. And [a] window is going to pop up and ask you which file you'd like to import, and it's going to start with the thing that's in your working directory, this "BikeData.csv", so I'm just going to select it and open. Now you're going to see this box pop up, and it gives you some pieces of information, such as what the data set looks like in its current state, what it will look like as it's imported into R Studio, and an object called a data frame, which we'll talk about in a second, and then it gives you some options to name your data.



Name: BikeData

Encoding: Automatic

Heading: ● Yes ○ No

Row names: Automatic

Separator: Comma

Decimal: Period

Quote: Double quote (")

Comment: None

na.strings: NA

☑ Strings as factors

Input File

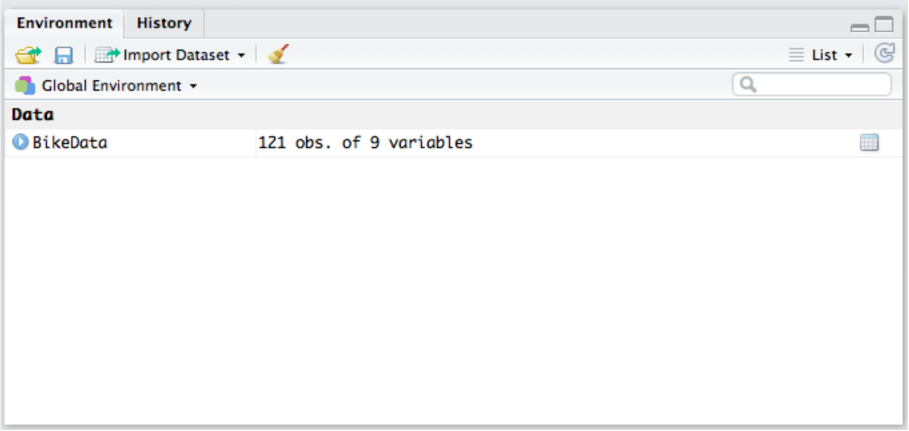
```
user_id,age,gender,student,employed,cyc_freq,distance,time
1,28,M,1,1,Daily,3.25,15,13
2,35,M,0,1,Daily,1.11,5,13.32
3,28,M,0,1,Daily,5.59,23,14.58
4,44,F,0,1,Less than once a month,3.24,24,8.1
5,42,M,0,1,Several times per week,7.81,26,18.02
6,36,M,0,1,Several times per week,3,20,9
7,45,M,0,1,Several times per week,13.03,51,15.33
8,54,M,0,1,Several times per week,6.21,39,9.55
9,39,M,0,1,Several times per week,13.95,50,16.74
10,44,M,0,0,Several times per month,7.4,44,10.09
11,29,M,0,1,Daily,4.36,20,13.08
12,32,M,0,1,Several times per week,5.02,27,11.16
13,40,M,0,1,Several times per week,5.01,23,13.07
```

Data Frame

```
user_id  age  gender  student  employed  cyc_freq
 1       28   M       1        1         Daily
 2       35   M       0        1         Daily
 3       28   M       0        1         Daily
 4       44   F       0        1         Less than o
 5       42   M       0        1         Several tim
 6       36   M       0        1         Several tim
 7       45   M       0        1         Several tim
 8       54   M       0        1         Several tim
 9       39   M       0        1         Several tim
10       44   M       0        0         Several tim
11       29   M       0        1         Daily
12       32   M       0        1         Several tim
13       40   M       0        1         Several tim
```

Figure 3: CSV Import Window.

This is the object name you're going to use for R Studio. It's usually recommended to use something short and sweet. "BikeData" sounds fine to me. This heading option lets you decide if the first row of your data set contains the column names, the variable names that you want to use. Ours does, so I'm going to keep that selected as Yes, and then the rest of these are usually going to not be modified, especially if you're working with a CSV file. I'm going to go ahead and hit Import.

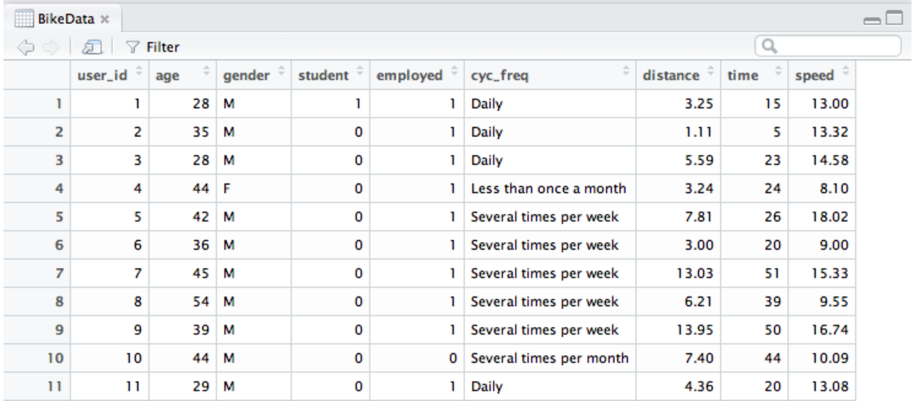


Data

BikeData 121 obs. of 9 variables

Figure 4: RStudio Workspace.

You'll see a couple things happen right away. My workspace or environment is now populated with this data frame called "BikeData." It tells me my number of observations, number of rows or cases. I have 121 of them. And it also tells me the number of variables or columns that I have in my data set. And then it opened up a spreadsheet view of this data set automatically.



	user_id	age	gender	student	employed	cyc_freq	distance	time	speed
1	1	28	M	1	1	Daily	3.25	15	13.00
2	2	35	M	0	1	Daily	1.11	5	13.32
3	3	28	M	0	1	Daily	5.59	23	14.58
4	4	44	F	0	1	Less than once a month	3.24	24	8.10
5	5	42	M	0	1	Several times per week	7.81	26	18.02
6	6	36	M	0	1	Several times per week	3.00	20	9.00
7	7	45	M	0	1	Several times per week	13.03	51	15.33
8	8	54	M	0	1	Several times per week	6.21	39	9.55
9	9	39	M	0	1	Several times per week	13.95	50	16.74
10	10	44	M	0	0	Several times per month	7.40	44	10.09
11	11	29	M	0	1	Daily	4.36	20	13.08

Figure 5: View Window in RStudio.

See, it says "BikeData", and then it actually has the cases listed down, and you can scroll through and see the raw data that way. Notice that they put the spreadsheet view of the data set right up here with the tabs of the other script files, so you can close this out just like you can a script file. If you ever want to open it back up and look at it again, you can just click over in the environment and it'll pop back up for you. Now, like I said, this data set was brought in as a data frame, which is an object in R that is made up of cases in the rows and variables in the columns, so it's basically a matrix.

Alternatively, you can import a data set (as long as it's a .csv file) by using the "**read.csv()**" function, and pointing to the file:

```r
BikeData <- read.csv("~/Desktop/SDSFoundations/BikeData.csv")
```

The newest version of the "SDSFoundations" package also has the data in it (be sure you're using the newest version available from the website). To use the data this way, you'll just need to call the package, then assign a name (the left side of the arrow) to the stored data (the right side of the arrow):

```
library(SDSFoundations)
BikeData <- BikeData
```

Now, in this particular data set, we've got some survey data from people that use bikes to get to work or school. So basically, we have information on their age, their gender, whether they're male or female, and some dummy variables for whether or not this person is a student or has a job. Then we have the frequency that they ride their bicycle, whether it's daily, just a few times a month, et cetera. We have the distance of the trip they took, we have the time, which is in minutes. The distances in miles, and then we have the speed, which is just miles per hour. So we have a nice collection here of both categorical and numeric variables, and we can see how we might want to treat both of those types of variables when we're working with a data frame. So the one thing to keep in mind when you have a data frame in R Studio is the way to call a specific variable. It's really simple notation, but say we wanted to get the mean distance traveled of these bicyclists.

We're going to want to use our "**mean()**" function, like we learned in a previously, open parentheses, and then we just want to give basically a vector of numbers and it'll calculate the mean for us. In order to do that, we're going to use the data frame name, "BikeData." Remember to watch your capitalization here because R is case sensitive. Then we're going to define which variable we want from the data frame Bike Data by placing a dollar sign and then just simply the column name, Distance.

```
mean(BikeData$distance)
```

```
## [1] 5.990661
```

Now, when we run this, R would go in and look into "BikeData" and find that "distance" column and calculate the mean. So we see that there's about a mean of six miles that these bicyclists rode.

Now, in order to get a quick snapshot of a categorical variable, we can use the "**table()**" function. Say we wanted to go in and see the counts of how many people rode daily, how many people rode several times per week, and all the various categories in this "cyc_freq" variable.

Well again, we can call that specific variable with our dollar sign notation, and in order to get a frequency table, we can just use the "**table()**" command. So we want to get a table of "BikeData$cyc_freq".

```
table(BikeData$cyc_freq)
```

```
##
##                   Daily  Less than once a month Several times per month
##                      47                       2                      14
##  Several times per week
##                      58
```

And here we see output in our console window the values that exist in this categorical variable– daily, less than once a month, several times per month, several times per week– and also the count of cases that have those specific values.