

# Foundations of Data Analysis

*The University of Texas at Austin*

*R Tutorials: Week 1*

## Indexing Data Frames

In this R tutorial, we're going to learn how to index within a data frame and how to use logical indexing to pull out specific cases or variables from a data set. So we've already got the bike data set imported into the environment here. And you can go back to the previous R tutorial about how to import a CSV file into RStudio. But remember our bike data set contains survey data for 121 cyclists who bike to either work or school. And we've got some data on these cyclists— their age, gender, whether or not they're a student or employed, how often they cycle, their distance and time and speed of the bicycle trip that the survey was taken on.

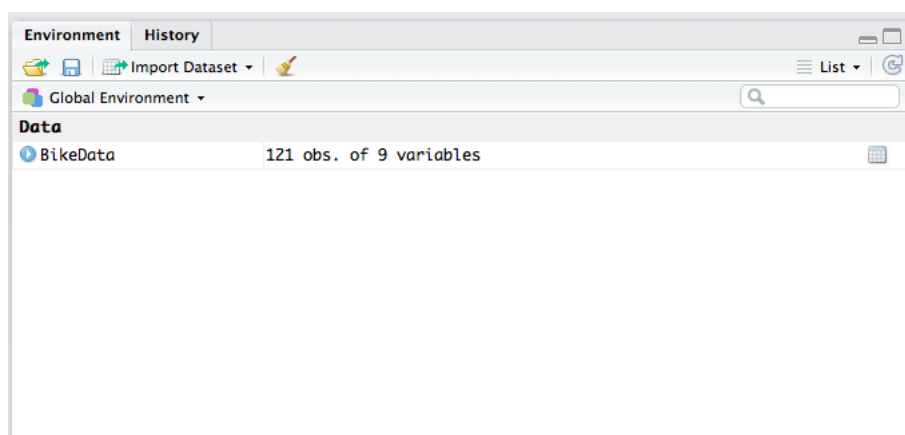


Figure 1: Imported BikeData.

So in a previous tutorial we talked about how to index a vector. And remember that the notation for indexing, pulling out a specific case within a vector, you would use the square brackets. So if I had a vector, “a”, I can pull out the, say, second item in “a” by putting the number 2 in brackets.

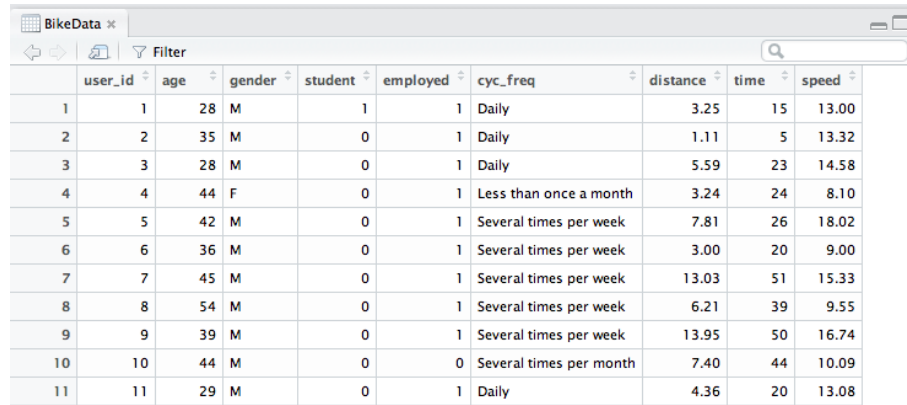
```
a[2]
```

Now within a data frame, if I want to pull out a specific case, it's the same idea. We just have to add one more element. So if I wanted to go into “BikeData” and pull out a certain case, remember that a data frame contains both rows and columns, so we're going to need to give it two indices within these square brackets. So if I want to go to say the second row and the fifth column and see what that gives me, I could just do 2 comma 5 inside my square brackets, run this line, and I would see a 1 returned.

```
BikeData[2,5]
```

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

Now if I go look at my bike data set and I go to the second row and the fifth column, I'm over here in this employed variable, and the case for the second row is that this person was employed, so it returned a 1.



	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 2: View Window in RStudio.

So pretty straightforward. Just remember it's row comma column when you're indexing within a data frame. And there's a couple tricks you can do within this.

Say you wanted to pull out the employed information for every single case in your data set. You could do this by, again, indexing "BikeData". Now I want to pull out every single row, and I'm actually going to leave it blank in this case. Just use a comma to separate into my column index and then give it the column that I want to pull out. So this is basically telling our studio to give me every single case in column 5 from bike data.

```
BikeData[,5]
```

```
## [1] 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0
## [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
## [71] 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
## [106] 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1
```

Now if I run that, I'm going to get a bunch of ones and zeros basically lining up with all the employed values for every single case. Usually what you're going to want to do is pull out certain cases that meet some criteria.

So let's say I wanted to go in and look at the number of males and females in my data set. I can use the table function to get the counts of a categorical variable. So remember that's just table and then open parentheses, my data frame name, dollar sign, and then the variable that I want a frequency table of, which in this case is gender.

```
table(BikeData$gender)
```

```
##
## F M
## 31 90
```

So if I run that, I can see that I have 39 females and 90 males in my data set. Let's say I wanted to look at a subset of my data and I only wanted to consider cases that were female.

Well, I can combine the idea of indexing and then logical statements like we learned in a previous tutorial to do just that. So first let's think about how we can create a logical statement to pull out only females. My logical statement will contain the variable I want to condition on, which is gender. And then remember the equals sign, if I want to say pull out only cases where gender is female, I'm going to go ahead and use the logical equals, which is 2 equal signs back to back. And then I want to give it the condition where I want to

be true. In this case, we have a categorical variable with “F”’s and “M”’s (remember, R is case sensitive). And in order to tell R I want to pull out a character, I need to put that character in quotes. So I just had a single quote. Notice R gives me that double quote to end it so I don’t have to remember to do that. And then I just need to give it my “F” for female. This is going to say, go into bike data set and look where gender equals “F”. So if I run this line, I’m going to get a true/false statement for every single case in my data set, basically showing that this statement is not true for the first row or the second or the third, and the fourth row contains my first female.

```
BikeData$gender == "F"
```

```
## [1] FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [12] FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE
## [23] FALSE TRUE FALSE FALSE TRUE FALSE TRUE FALSE FALSE FALSE FALSE
## [34] FALSE FALSE TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [45] FALSE TRUE TRUE FALSE FALSE FALSE FALSE TRUE TRUE FALSE FALSE
## [56] FALSE TRUE TRUE FALSE FALSE TRUE TRUE FALSE FALSE FALSE TRUE
## [67] FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [78] FALSE TRUE FALSE TRUE TRUE FALSE FALSE FALSE FALSE FALSE
## [89] FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE
## [100] FALSE FALSE FALSE TRUE FALSE TRUE FALSE FALSE FALSE TRUE FALSE
## [111] FALSE FALSE FALSE FALSE TRUE TRUE FALSE TRUE TRUE TRUE FALSE
```

And I can go ahead and confirm that by just looking into my data set and seeing female on my fourth row. So we’re not going to want to output a bunch of trues and false statements like this, but just to start building our logical statement, it helps to kind of go at it piece by piece.

So now that we know how to build a vector of all of our females coming out as true, I can build an object that is simply the data set, bike data, but only where gender equals female. I want to call this new data frame maybe something like females. I’m going to use the assignment key because I’m basically giving a value into this new object name. And I want to take my original bike data and then give it a certain index. And here’s where I’m going to put that logical statement. I want to take the data frame bike data and I only want to take the rows where this condition is true. So I’m just going to go ahead and copy this statement here, paste it in my row spot of my index. Remember to use the comma. And now here comes that trick again. If I want every column in my data set to go into this new females object, I’m just going to leave it blank. So indexing the bike data data frame and I’m pulling out these rows and all columns. And if you’re giving it a logical statement like this, it’s only going to go into the rows and find the true spots and pull those out. So let’s see what this does.

```
female <- BikeData[BikeData$gender == "F", ]
```

If I run this line a couple things happen. The code is submitted to my console as always, but I also see a new data frame pop up. And this one is called females.

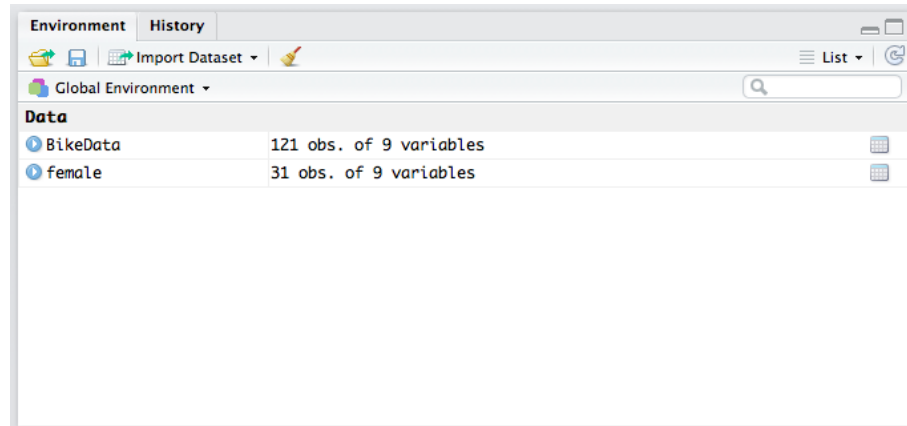


Figure 3: RStudio Workspace.

It has the same 9 variables, because, remember, I pulled out every single column. But it only has 31 observations. And that match is what I would expect because I know that there are 31 females in my original data set. Now this new data frame is exactly like the old one. I can open it up in a spreadsheet view. I can run functions on certain variables from it. But it only contains the females from my original data.

Now sometimes you might not want to make a copy of an entire data frame, you might want to just pull out one variable for a certain criteria. So instead of making an entire data frame of all the females, what if we only wanted to look at say the speed of the female bicyclists? I can do that with just a slight change in notation, but very similar to what we did here. So say I wanted to make a vector now called “femalespeeds” and I want to assign it every speed case from my data set where the gender column is a female. So I’m going to do a very similar notation. I’m going to start with “BikeData”. Now I want to give it the “dollar sign” “speed” variable saying, go in and find this variable within this data frame. But I can also index this with the same criteria I have here right where gender equals F. So if I copy and paste this into my brackets now, this line will give me a vector called female speeds that contains the speed value for every female case in my data set.

```
femalespeeds <- BikeData$speed[BikeData$gender == "F"]
```

Now I don’t need to include a comma here. And the reason is up in the prior line we were indexing a data frame, and a data frame has both rows and columns. Above, I’m pulling out a specific variable and indexing it, so it already knows to go in and find that speed column. If you insert a comma here, it’s OK. You’ll just get an error and it’ll be a reminder that you’ve got some incorrect number of dimensions, meaning it’s asking for a row and a column but you only have a vector, so there’s only one index that’s needed. Notice “femalespeeds” not a data frame but it’s a vector. And it contains the speeds of all my female bicyclists. Now I can ask for different functions of that vector just like I’ve done with other objects. I can get the mean of my female speeds, which is the mean function. And I could see that on average female bicyclists in my study traveled about 13.7 miles per hour.

```
mean(femalespeeds)
```

```
## [1] 13.65516
```