There are many ways to select and subset data in R, depending on what data structure is being used. When it comes to vectors, the easiest way to select data is using the bracket ("[ ]") notation. For example, if we have a numeric vector x with 10 values and want to select the third value, we would use the following statements:

> x <- c(3, 3, 2, 2, 5, 5, 8, 8, 9)

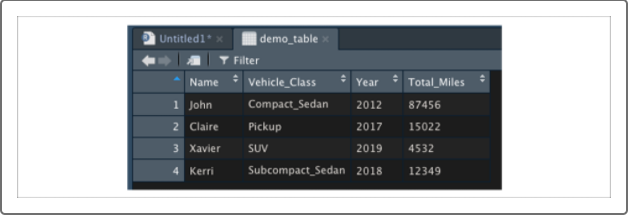
> x[3]



**IMPORTANT**

Unlike Python, R's index starts at 1. So, the third element would be index = 3.

You can also use bracket notation to select data from two-dimensional data structures, such as matrices, data frames, and tibbles. For example, let's look at our demo\_table again:

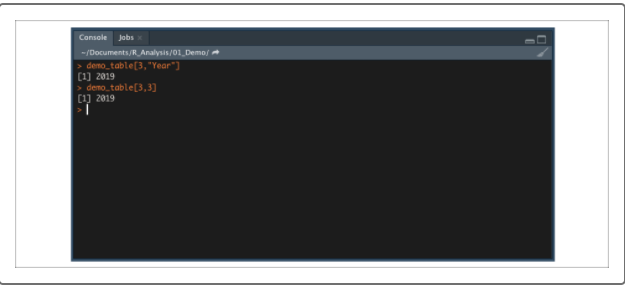


If we want to select the third row of the Year column using bracket notation, our statement would appear as follows:

> demo\_table[3,"Year"]

Because R keeps track of both the row indices as well as the column indices as integers under the hood, we can also select the same data using just number indices:

> demo\_table[3,3]

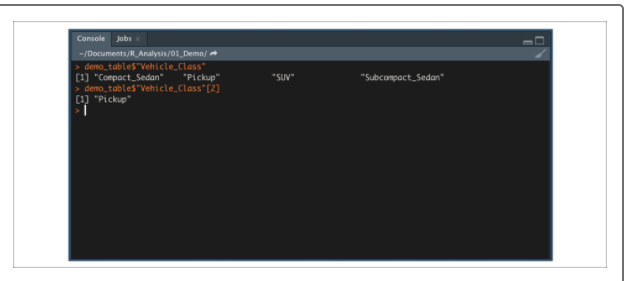


There is a third way to select data from an R data frame that behaves very similarly to Pandas. By using the $ operator, we can select columns from any two-dimensional R data structure as a single vector, similar to selecting a series from a Pandas DataFrame. For example, if we want to select the vector of vehicle classes from demo\_table, we would use the following statement:

> demo\_table$"Vehicle\_Class"

Once we have selected the single vector, we can use bracket notation to select a single value.

> demo\_table$"Vehicle\_Class"[2]



**Select Data with Logic**

Just as it is for selecting single values, there are multiple ways to subset and filter data from our larger data frames. As with most programming languages, we use a combination of operators and logical statements to tell R what data to filter. Thankfully, most operators are the same between R and Python, as shown below:

One of the most common ways to filter and subset a dataset in R is to use bracket notation. To use bracket notation to filter a data frame, we can supply a logical statement to assert our row and columns.

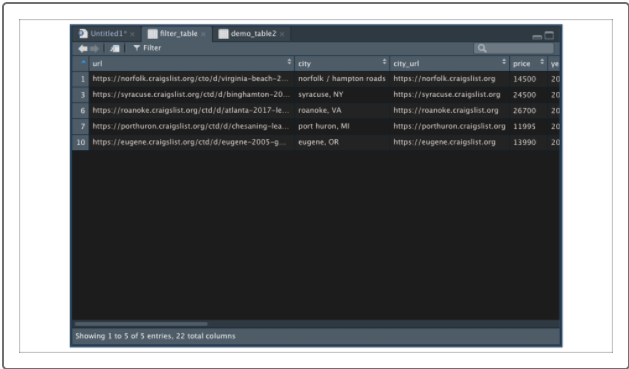
For example, if we want to filter our used car data demo\_table2 so that we only have rows where the vehicle price is greater than $10,000, we would use the following statement:

> filter\_table <- demo\_table2[demo\_table2$price > 10000,]

You might be wondering why a comma trails 10000. The comma is necessary to subset by rows. Adding column(s) after the comma specifies the columns to select.



This filter statement generates a view-only data frame tab listing vehicles priced greater than $10,000, as shown in the following image



**NOTE**

If you do not supply a logical statement to either rows or columns, R will default to returning all elements in that dimension.

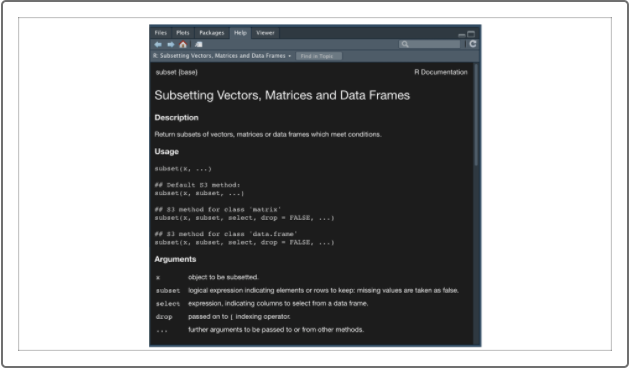
In this example, the demo\_table2$price > 10000 logical statement creates a vector of TRUE/FALSE values that R uses to consider all rows that satisfy our logical statement.

When our logical statements are simple (using only one or two operators), bracket notation is easy to read and write. However, if we need to filter and subset our data using more complicated logic, bracket notation can become cumbersome. In these cases, we'll use an R function such as subset() to filter our data.

**Subset Data in R**

Another method to filter and subset data frames in R is to use the function subset(). Type the following code into the R console to look at the subset() documentation in the Help pane:

> ?subset()

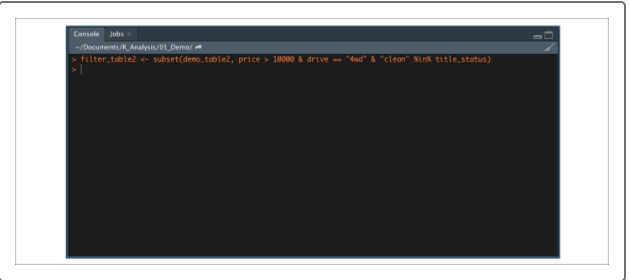


The subset() function uses a few arguments to subset and filter a two-dimensional R data structure:

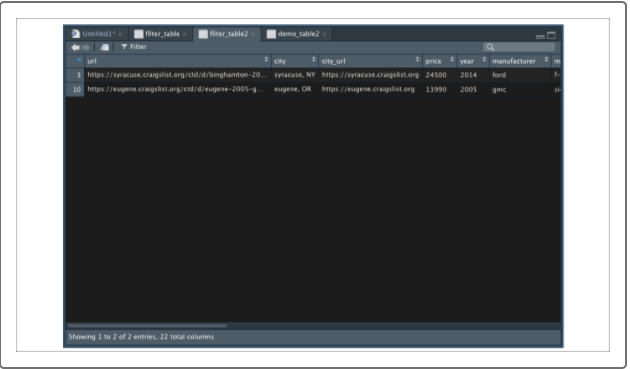
* **x**
* **subset**
* **select**

For example, if we want to create a more elaborate filtered dataset from our used car data demo\_table2 where price > 10000, drive == 4wd, and "clean" %in% title\_status, we would use the following statement:

> filter\_table2 <- subset(demo\_table2, price > 10000 & drive == "4wd" & "clean" %in% title\_status) #filter by price and drivetrain



Here's what the filtered data frame will look like:



**IMPORTANT**

When combining logical statements in R, use element-wise AND operator or element-wise OR operator.

In this case, using subset() is cleaner than using brackets, which would look like this:

> filter\_table3 <- demo\_table2[("clean" %in% demo\_table2$title\_status) & (demo\_table2$price > 10000) & (demo\_table2$drive == "4wd"),]

The subset() function makes filtering and subsetting easier to read by assuming column names in the **subset** argument, which cuts down on statement length. In almost all cases, the bracket notation and subset() function are functionally equivalent (especially when using logical statements) and interchangeable.

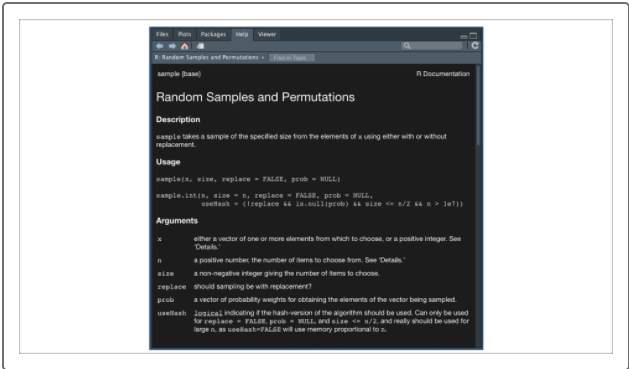
## Sample Data in R

Often in data science, we need to generate a random sample of data points from a larger dataset. For example, some models might take too long to run on a massive dataset and require a smaller sample of the data.

Using filtering and subsetting methods may be appropriate for certain cases (such as looking at data within a specific timeframe), but usually we'll want to randomly sample our larger data to reduce bias. In these cases, we can use the built-in function sample(). Let's try it now.

Type the following code into the R console to look at the sample() documentation in the Help pane:

> ?sample()



The sample() function uses a few arguments to create a sampled vector from a larger vector:

* **x**
* **size**
* **replace**
* If we want to sample a large vector and create a smaller vector, we can set the vector to "x":
* > sample(c("cow", "deer", "pig", "chicken", "duck", "sheep", "dog"), 4)



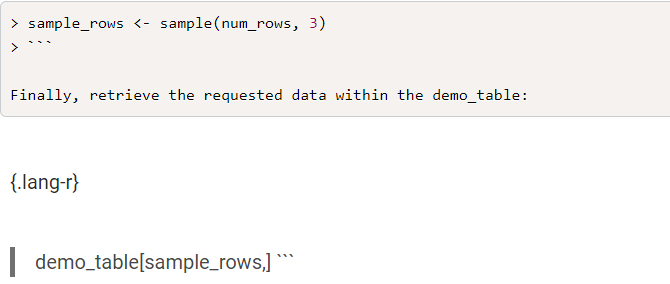
When it comes to sampling a two-dimensional data structure, we need to supply the index of each row we want to sample. This process can be completed in three steps:

1. Create a numerical vector that is the same length as the number of rows in the data frame using the colon (:) operator.
2. Use the sample() function to sample a list of indices from our first vector.
3. Use bracket notation to retrieve data frame rows from sample list.

So, first capture the number of rows in demo\_table in a variable. The nrow() function counts the number of rows in a dataframe.

> num\_rows <- 1:nrow(demo\_table)

Next, sample 3 of those rows, as shown in this code:

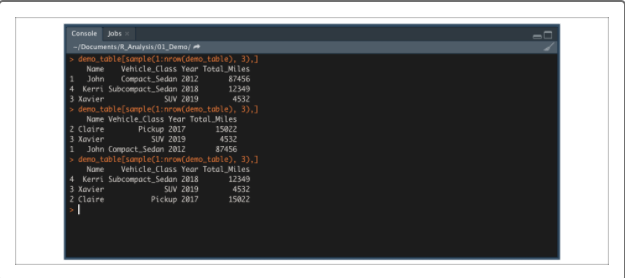


If we want to combine all three steps in a single R statement, our code would be as follows:

> demo\_table[sample(1:nrow(demo\_table), 3),]

If we want to combine all three steps in a single R statement, our code would be as follows:

> demo\_table[sample(1:nrow(demo\_table), 3),]



After we have successfully loaded in and selected our data, our next steps in analysis are to group, transform, and reshape our data as to prepare for visualizations and modeling.