R Programming Interview Questions and Answers

1. What is R?

This should be an easy one for data science job applicants. R is an open-source language and environment for statistical computing and analysis, or for our purposes, data science.

2. Can you write and explain some of the most common syntax in R?

Again, this is an easy—but crucial—one to nail. For the most part, this can be demonstrated through any other code you might write for other R interview questions, but sometimes this is asked as a standalone. Some of the basic syntax for R that's used most often might include:

`#` — as in many other languages, `#` can be used to introduce a line of comments. This tells the compiler not to process the line, so it can be used to make code more readable by reminding future inspectors what blocks of code are intended to do.

`""` — quotes operate as one might expect; they denote a string data type in R.

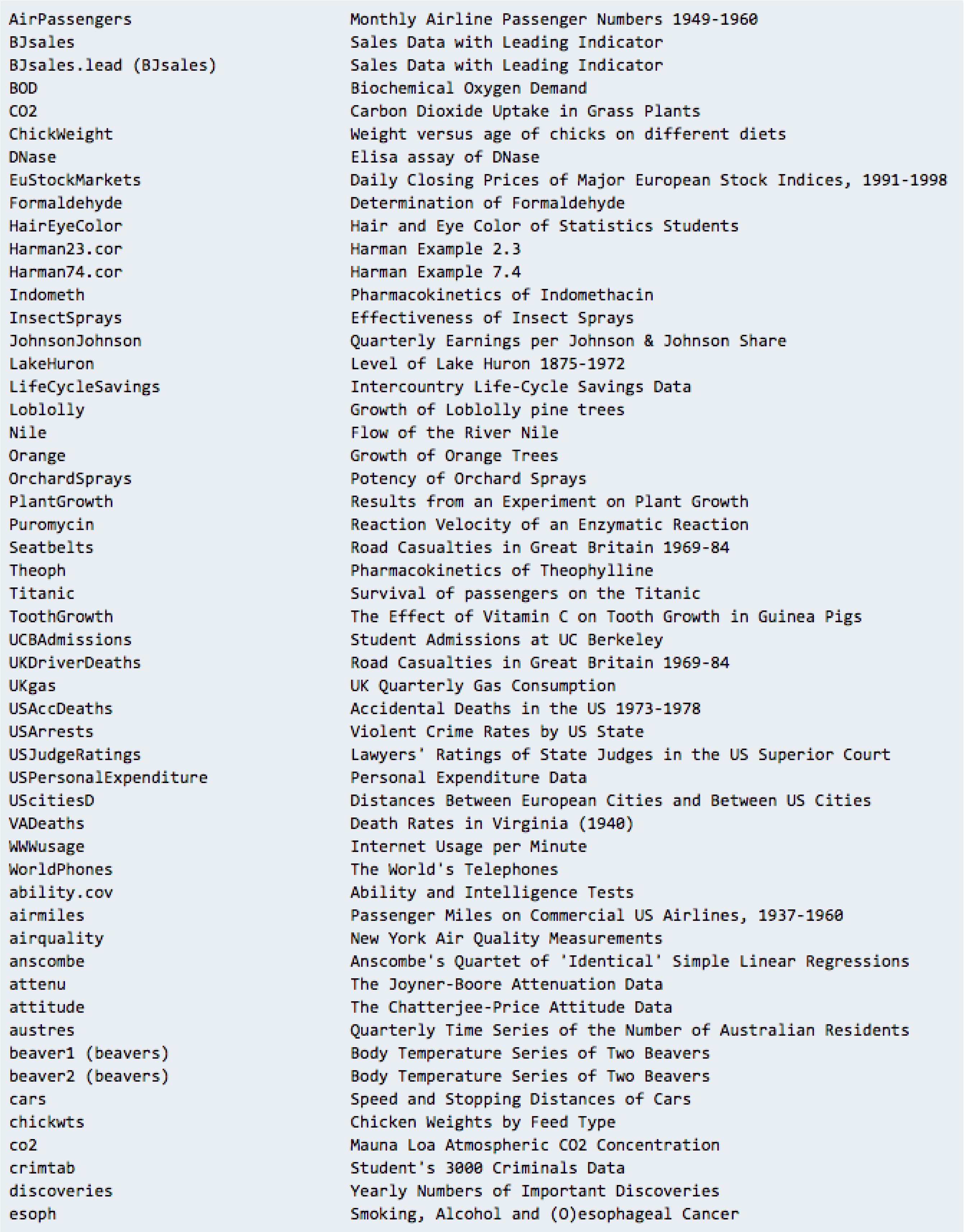
`<-` — one of the quirks of R, the assignment operator is `<-` rather than the relatively more familiar use of `=`. This is an essential thing for those using R to know, so it would be good to display your knowledge of it if the question comes up.

`\` — the backslash, or reverse virgule, is the escape character in R. An escape character is used to "escape" (or ignore) the special meaning of certain characters in R and, instead, treat them literally.

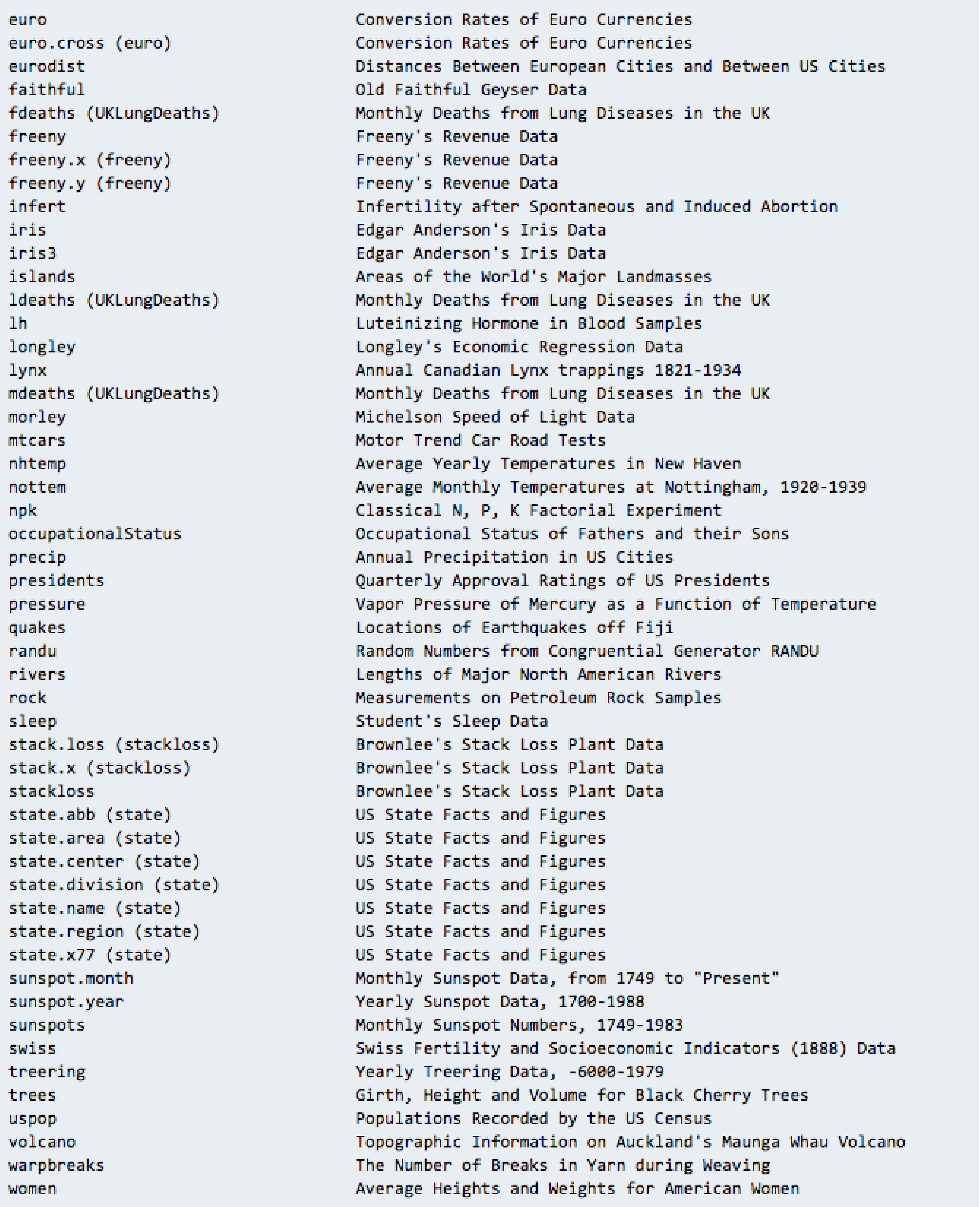
3. How do you list the preloaded datasets in R?

To view a list of preloaded datasets in R, simply type data() into the console and hit enter.

I'm running R version 3.5.1 for Windows on my machine with a standard, default install. The output in RStudio for me looks something like this:



```
AirPassengers                       Monthly Airline Passenger Numbers 1949-1960
BJsales                             Sales Data with Leading Indicator
BJsales.lead (BJsales)              Sales Data with Leading Indicator
BOD                                 Biochemical Oxygen Demand
CO2                                 Carbon Dioxide Uptake in Grass Plants
ChickWeight                         Weight versus age of chicks on different diets
DNase                               Elisa assay of DNase
EuStockMarkets                      Daily Closing Prices of Major European Stock Indices, 1991-1998
Formaldehyde                        Determination of Formaldehyde
HairEyeColor                        Hair and Eye Color of Statistics Students
Harman23.cor                        Harman Example 2.3
Harman74.cor                        Harman Example 7.4
Indometh                            Pharmacokinetics of Indomethacin
InsectSprays                        Effectiveness of Insect Sprays
JohnsonJohnson                      Quarterly Earnings per Johnson & Johnson Share
LakeHuron                           Level of Lake Huron 1875-1972
LifeCycleSavings                    Intercountry Life-Cycle Savings Data
Loblolly                            Growth of Loblolly pine trees
Nile                                Flow of the River Nile
Orange                              Growth of Orange Trees
OrchardSprays                       Potency of Orchard Sprays
PlantGrowth                         Results from an Experiment on Plant Growth
Puromycin                           Reaction Velocity of an Enzymatic Reaction
Seatbelts                           Road Casualties in Great Britain 1969-84
Theoph                              Pharmacokinetics of Theophylline
Titanic                             Survival of passengers on the Titanic
ToothGrowth                         The Effect of Vitamin C on Tooth Growth in Guinea Pigs
UCBAdmissions                       Student Admissions at UC Berkeley
UKDriverDeaths                      Road Casualties in Great Britain 1969-84
UKgas                               UK Quarterly Gas Consumption
USAccDeaths                         Accidental Deaths in the US 1973-1978
USArrests                           Violent Crime Rates by US State
USJudgeRatings                      Lawyers' Ratings of State Judges in the US Superior Court
USPersonalExpenditure               Personal Expenditure Data
UScitiesD                           Distances Between European Cities and Between US Cities
VADeaths                            Death Rates in Virginia (1940)
WWWusage                            Internet Usage per Minute
WorldPhones                         The World's Telephones
ability.cov                         Ability and Intelligence Tests
airmiles                            Passenger Miles on Commercial US Airlines, 1937-1960
airquality                          New York Air Quality Measurements
anscombe                            Anscombe's Quartet of 'Identical' Simple Linear Regressions
attenu                              The Joyner-Boore Attenuation Data
attitude                            The Chatterjee-Price Attitude Data
austres                             Quarterly Time Series of the Number of Australian Residents
beaver1 (beavers)                   Body Temperature Series of Two Beavers
beaver2 (beavers)                   Body Temperature Series of Two Beavers
cars                                Speed and Stopping Distances of Cars
chickwts                            Chicken Weights by Feed Type
co2                                 Mauna Loa Atmospheric CO2 Concentration
crimtab                             Student's 3000 Criminals Data
discoveries                         Yearly Numbers of Important Discoveries
esoph                               Smoking, Alcohol and (O)esophageal Cancer
```



```
euro                        Conversion Rates of Euro Currencies
euro.cross (euro)           Conversion Rates of Euro Currencies
eurodist                    Distances Between European Cities and Between US Cities
faithful                    Old Faithful Geyser Data
fdeaths (UKLungDeaths)      Monthly Deaths from Lung Diseases in the UK
freeny                      Freeny's Revenue Data
freeny.x (freeny)           Freeny's Revenue Data
freeny.y (freeny)           Freeny's Revenue Data
infert                      Infertility after Spontaneous and Induced Abortion
iris                        Edgar Anderson's Iris Data
iris3                       Edgar Anderson's Iris Data
islands                     Areas of the World's Major Landmasses
ldeaths (UKLungDeaths)      Monthly Deaths from Lung Diseases in the UK
lh                          Luteinizing Hormone in Blood Samples
longley                     Longley's Economic Regression Data
lynx                        Annual Canadian Lynx trappings 1821-1934
mdeaths (UKLungDeaths)      Monthly Deaths from Lung Diseases in the UK
morley                      Michelson Speed of Light Data
mtcars                      Motor Trend Car Road Tests
nhtemp                      Average Yearly Temperatures in New Haven
nottem                      Average Monthly Temperatures at Nottingham, 1920-1939
npk                         Classical N, P, K Factorial Experiment
occupationalStatus          Occupational Status of Fathers and their Sons
precip                      Annual Precipitation in US Cities
presidents                  Quarterly Approval Ratings of US Presidents
pressure                    Vapor Pressure of Mercury as a Function of Temperature
quakes                      Locations of Earthquakes off Fiji
randu                       Random Numbers from Congruential Generator RANDU
rivers                      Lengths of Major North American Rivers
rock                        Measurements on Petroleum Rock Samples
sleep                       Student's Sleep Data
stack.loss (stackloss)      Brownlee's Stack Loss Plant Data
stack.x (stackloss)         Brownlee's Stack Loss Plant Data
stackloss                   Brownlee's Stack Loss Plant Data
state.abb (state)           US State Facts and Figures
state.area (state)          US State Facts and Figures
state.center (state)        US State Facts and Figures
state.division (state)      US State Facts and Figures
state.name (state)          US State Facts and Figures
state.region (state)        US State Facts and Figures
state.x77 (state)           US State Facts and Figures
sunspot.month               Monthly Sunspot Data, from 1749 to "Present"
sunspot.year                Yearly Sunspot Data, 1700-1988
sunspots                    Monthly Sunspot Numbers, 1749-1983
swiss                       Swiss Fertility and Socioeconomic Indicators (1888) Data
treering                    Yearly Treering Data, -6000-1979
trees                       Girth, Height and Volume for Black Cherry Trees
uspop                       Populations Recorded by the US Census
volcano                     Topographic Information on Auckland's Maunga Whau Volcano
warpbreaks                  The Number of Breaks in Yarn during Weaving
women                       Average Heights and Weights for American Women
```

4. What are some advantages of R?

It's important to be familiar with the advantages and disadvantages of certain languages and ecosystems. R is no exception. So what are the advantages of R?

Its open-source nature. This qualifies as both an advantage and disadvantage for various reasons, but being open source means it's widely accessible, free to use, and extensible.

Its package ecosystem. The built-in functionality available via R packages means you don't have to spend a ton of time reinventing the wheel as a data scientist.

Its graphical and statistical aptitude. By many people's accounts, R's graphing capabilities are unmatched.

5. What are the disadvantages of R?

Just as you should know what R does well, you should understand its failings.

Memory and performance. In comparison to Python, R is often said to be the lesser language in terms of memory and performance. This is disputable, and many think it's no longer relevant as 64-bit systems dominate the marketplace.

***Related**: Our list of Python Interview Questions and Answers*

Open source. Being open source has its disadvantages as well as its advantages. For one, there's no governing body managing R, so there's no single source for support or quality control. This also means that sometimes the packages developed for R are not the highest quality.

Security. R was not built with security in mind, so it must rely on external resources to mind these gaps.

6. What are the similarities and differences between R and Python?

Attribute	*Python*	*R*
Cost	★★★★★	★★★★★
Security	★★★★☆	★☆☆☆☆
Model building	★★★★★	★★★★★
Learning curve	★★★☆☆	★★☆☆☆
Visualization tools and libraries	★★★☆☆	★★★★★

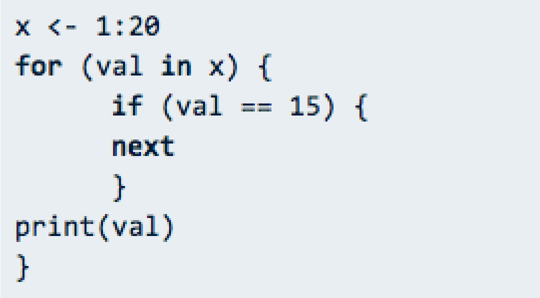
There are many comparisons to draw between Python and R. They are both free. They both have strong modeling capabilities. Python is generally considered more secure and easier to learn, but R is typically thought to have better visualization tools and libraries. In many jobs, you'll be expected to use both R and Python, so it's good to know about both, even if you aren't fluent in both languages.

7. Write code to accomplish a task

In just about any interview for a position that involves coding, companies will ask you to accomplish a specific task by actually writing code. Facebook and Google both do as much. Because it's difficult to predict what task an interviewer will set you to, just be prepared to write "whiteboard code" on the fly.

8. When is it appropriate to use the “next” statement in R?

A data scientist will use next to skip an iteration in a loop. As an example:



```
x <- 1:20
for (val in x) {
      if (val == 15) {
      next
      }
print(val)
}
```

This code iterates through a range of numbers from 1 to 20 and prints the values. I don’t want to print 15, though, so I’ve used the next statement to skip that iteration and move on to other values. The output would print 1-14 and 16-20.

9. How do you assign a variable in R?

Variable assignment in R is a bit different from other languages. Rather than using an = sign, we typically use a less-than sign, < ,followed by a minus, –. An equals sign, =, still works, but there are arguments about its readability in addition to instances where it can actually muck up your code. I suggest you stick with <- for assignment in R, if for no other reason than it’s expected (Google’s R style manual, which is widely used, prohibits = for assignment).

```
myVar <- 15
print(myVar
```

Notice the following is also valid:

```
myVar = 15
print(myVar)
```

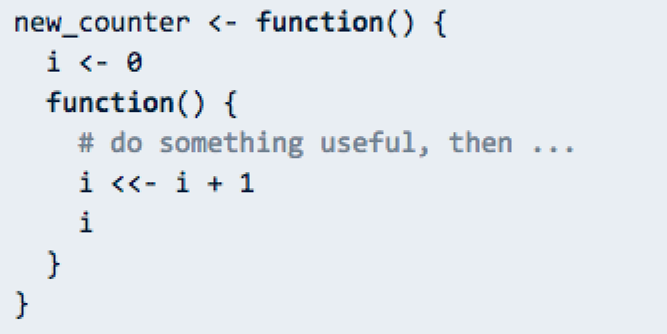
You can also assign variables the other way around in R.

```
"helloWorld" -> myVar
```

The string “helloWorld” is now stored in the myVar variable. Note that this would produce an error if we got mixed up and tried to plug an undefined variable object into a string, as in:

```
myVar -> "helloWorld"
Error: object 'myVar' not found
```

Scoping of variables is something to consider as well. <<- acts as the “superassignment” operator and is useful for closures. A good example on how to use this can be found on stackoverflow.



```
new_counter <- function() {
  i <- 0
  function() {
    # do something useful, then ...
    i <<- i + 1
    i
  }
}
```

10. What are the different data types/objects in R?

This is another good opportunity to show that you *know* R, and you're not winging it. Unlike other object-oriented languages such as C, R doesn't ask users to declare a data type when assigning a variable. Instead, everything in R correlates to an R data object. When you assign a variable in R, you assign it a data object and that object's data type determines the data type of the variable. The most commonly used data objects include:

- Vectors
- Matrices
- Lists
- Arrays
- Factors
- Data frames

11. What are the objects you use most frequently?

This question is meant to gather a sense of your experiences in R. Simply think about some recent work you've done in R and explain the data objects you use most often. If you use arrays frequently, explain why and how you've used them.

12. Why use R?

This is a variant of the "advantages of R" question. Reasons to use R include its open-source nature and the fact that it's a versatile tool for statistical plotting, analysis, and portrayal. Don't be afraid to give some personal reasons as well. Maybe you simply love the assignment operator in R or feel that it's more elegant than other languages—but always remember to explicate. You should be answering follow-up questions before they're even asked.

13. What are some of your favorite functions in R?

As a user of R, you should be able to come up with some functions on the spot and describe them. Functions that save time and, as a result, money will always be something an interviewer likes to hear about.

14. Write a custom function in R

Sometimes you'll be asked to create a custom function on the fly. An example of a custom function from quick-r's guide:

```
myFunction <- function(arg1, arg2, ... ){
statements
return(object)
}
```

Functions can be simple or complex, but they should make your code more extensible, readable, and efficient. This is a chance to show your ingenuity and experience.

15. How do you import data in R?

Let's use CSV as an example, as it's a very common data format. Simply make sure the file is saved in a CSV format, then use the read function to import the data.

```
yourRDateHere <- read.csv("Data.csv", header = TRUE)
```

Though not required, strictly speaking, the argument `header = TRUE` is used to ensure that labels are not parsed as data. Making this argument false means you either don't have labels in your CSV or you want them to be part of the data output in R.

To do the same with data in a .txt file, simply change `read.csv` to `read.table`.

16. How do you install a package in R?

There are many ways to install a package in R. Some even include using the GUI. We're coders, so we're not going to give those attention.

Type the following into your console and hit enter:

```
install.packages("package_name")
```

Followed by:

```
library(package_name)
```

It's that simple. The first command installs the package and the second loads the package into the session.

17. What is the use of with() in R?

We use the with() function to write simpler code by applying an expression to a data set. Its syntax looks like this:

```
with(randomDataSet, expression.test(sample))
```

18. What is the use of by() in R?

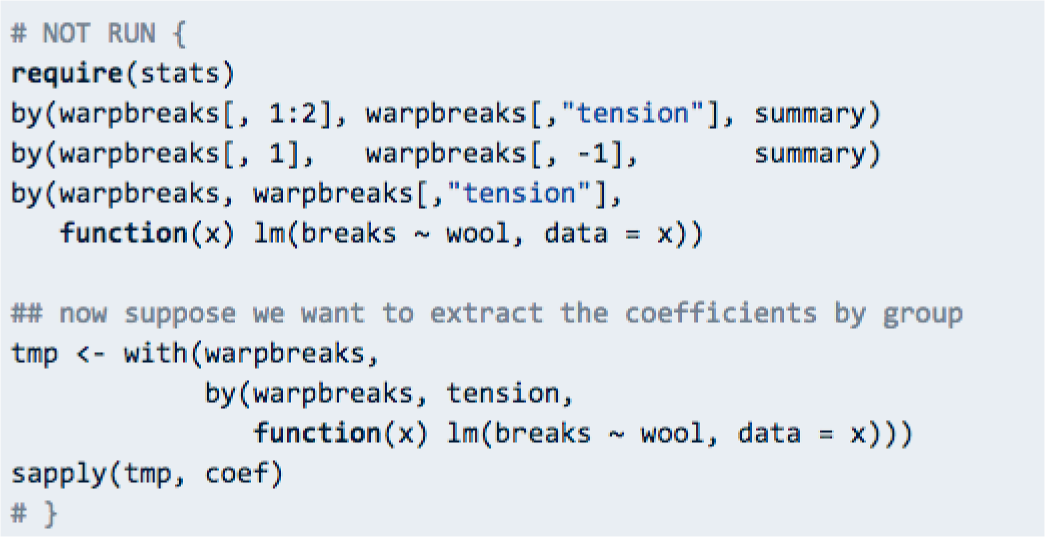
Like `with()`, `by()` can help write DRY (don't repeat yourself) code.

You can use `by()` to apply a function to a data frame split by factors. Its usage is something like this:

```
by(data, factor, function, ...)
```

The data frame plugged into this function is split into data frames (by row) subsetted by the values of factor(s), and a function is then applied to each subset.

A good example of this function in action can be found here:



```
# NOT RUN {
require(stats)
by(warpbreaks[, 1:2], warpbreaks[,"tension"], summary)
by(warpbreaks[, 1],   warpbreaks[, -1],       summary)
by(warpbreaks, warpbreaks[,"tension"],
   function(x) lm(breaks ~ wool, data = x))

## now suppose we want to extract the coefficients by group
tmp <- with(warpbreaks,
            by(warpbreaks, tension,
               function(x) lm(breaks ~ wool, data = x)))
sapply(tmp, coef)
# }
```

19. When is it appropriate to use mode()?

By default, `mode()` gets or sets the storage mode of an object. It's default usage is equivalent to storage.mode(). A sample usage:

```
x <- 1:25
mode(x)
[1] "numeric"
y <- "helloWorld"
mode(y)
[1] "character"
mode(state.name)
[1] "character"
```

In the first line, we assign x to values 1 through 25, so when we run it through mode, we get "numeric" because the variable stores numeric values. If we, instead, assign the variable to a string such as in the y variable above, we get "character" as the mode. You can try this out with predefined data sets as well, such as with `state.name`.

You can view documentation here.

20. What is a factor variable, and why would you use one?

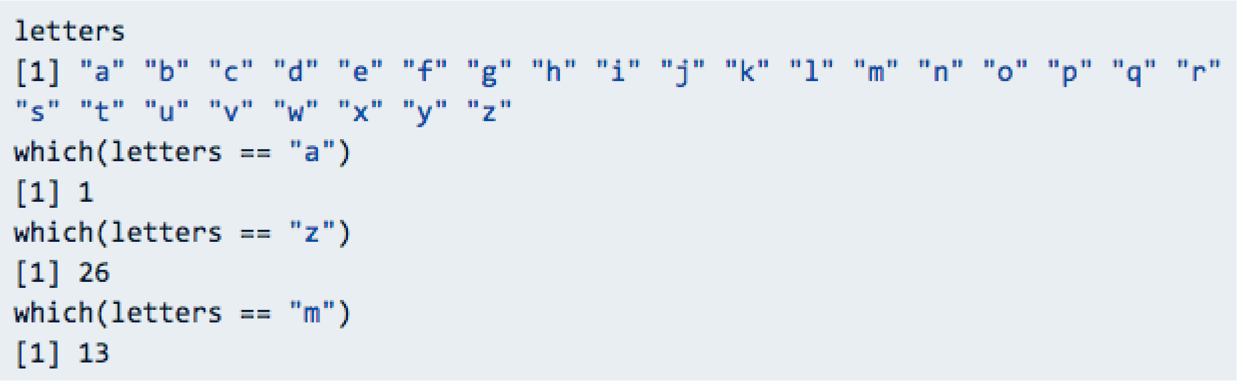
A factor variable is a form of categorical variable that accepts either numeric or character string values. The most salient reason to use a factor variable is that it can be used in statistical modeling with great accuracy. Another reason is that they are more memory efficient.

Simply use the `factor()` function to create a factor variable. There's more information here.

21. When is it appropriate to use the which() function?

The `which()` function loops through a logical object until the condition returns TRUE and returns the index `(position)` of the element.

To get a sense of how this works, plug in the letters array and search for the index of a specific letter using `which()`.



```
letters
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r"
"s" "t" "u" "v" "w" "x" "y" "z"
which(letters == "a")
[1] 1
which(letters == "z")
[1] 26
which(letters == "m")
[1] 13
```

In my console, I’ve checked the letters array, which contains the English alphabet in lowercase. I’ve used `which()` to find the positions of `a`, `z`, and `m`, which returned the indexes `1`, `26`, and `13`, respectively, because these are the positions in the array, as they are typically the positions in the alphabet.

22. How do you concatenate strings in R?

Concatenating strings in R is less than intuitive. You don’t use a `.` operator, nor a `+` operator, and forget about the `&` operator. In fact, you don’t use an operator at all. Concatenating strings in R requires the use of the `paste()` function. Here’s an example:

```
hello <- "Hello, "
world <- "World."
paste(hello, world)
[1] "Hello, World."
```

I’ve stored `Hello, ` and `World.` in variables aptly named `hello` and `world`. With paste(), I’ve simply plugged in the two variables, and it concatenates them such that it creates the single phrase “Hello, World.”

An oddity with this function is that it will automatically insert spaces between the terms. Try it out with some numbers.

```
paste(1,2,3,4)
[1] "1 2 3 4"
```

This can be a sort of “gotcha” with this question. If we don’t want spaces, we can adjust the sep parameter in the function, which defaults to a space `" "`.

```
paste(1,2,3,4,sep=“”)
[1] "1234"
```

23. How do you read a CSV file in R?

We’ve covered this already with the import process. Simply use the `read.csv()` function.

```
yourRDateHere <- read.csv("Data.csv", header = TRUE)
```

24. What are 3 sorting algorithms available in R?

R uses the `sort()` function to order a vector or factor, listed and described below.

Radix: Usually the most performant algorithm, this is a non-comparative sorting algorithm that avoids overhead. It's stable, and it's the default algorithm for integer vectors and factors.

Quick Sort: This method "uses Singleton (1969)'s implementation of Hoare's Quicksort method and is only available when x is numeric (double or integer) and partial is NULL," according to R Documentation. It's not considered a stable sort.

Shell: This method "uses Shellsort (an O(n4/3) variant from Sedgewick (1986))," according to R Documentation.

25. Can you create an R decision tree?

A decision tree is a familiar graph for data scientists. It represents choices and results through the graphical form of a tree. To keep things simple, let's just go over the basics.

Install the party package to get started with making the tree.

```
install.packages("party")
```

This gives you access to a fancy new function: ctree(), and, at its most basic, this is all we need to create a tree. First, let's grab some data from our package; make sure the package is loaded.

```
library(party)
```

Now we have access to some new data sets. Part of the strucchange package that bundles with party includes data on youth homicides in Boston called BostonHomicide. Let's use that one. You can print the data to the screen if you like.

```
print(BostonHomicide)
```

Now we'll create the tree. The usage of ctree() goes something like this:

```
ctree(formula,dataset)
```

We've got our data set. I'll assign it to a variable for simplicity.

```
inputData <- BostonHomicides
```

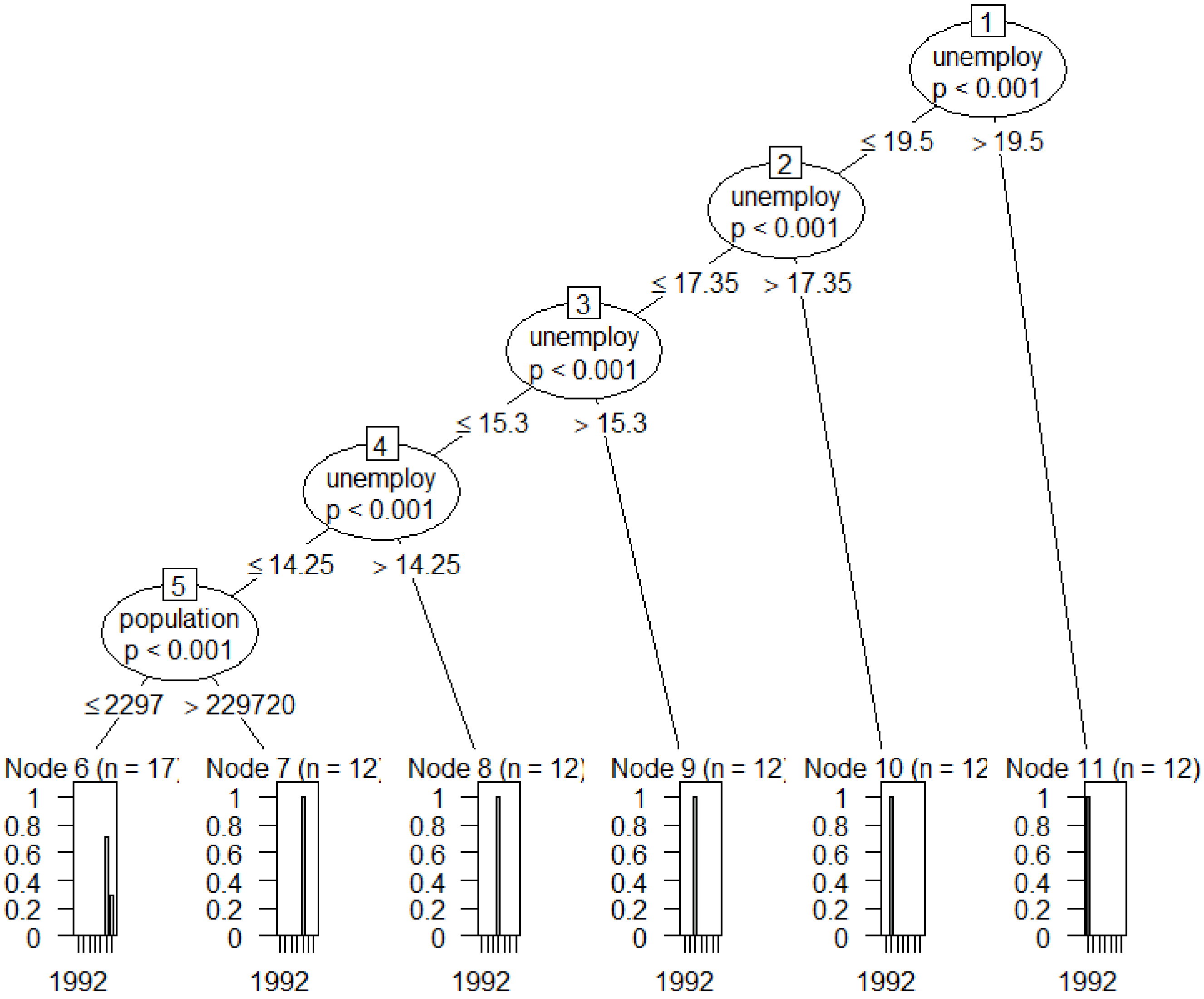
Now we can determine our formula and create the tree.

```
treeAnalysis <- ctree(year~population+homicides+unemploy, data = inputData)
```

Let's plot it!

```
plot(treeAnalysis)
```

Here's the result I got.



26. Why is R useful for data science?

R turns otherwise hours of graphically intensive jobs into minutes and keystrokes. In reality, you probably wouldn’t encounter the language of R outside the realm of data science or an adjacent field. It’s great for linear modeling, nonlinear modeling, time-series analysis, plotting, clustering, and so much more.

Simply put, R is designed for data manipulation and visualization, so it’s natural that it would be used for data science.

27. Describe how R can be used for predictive analysis

As a data manipulation and visualization tool, R can most definitely be used for predictive analytics. Using the same sort of decision tree we developed earlier, one could predict how many shootings might occur in 2019 in Boston. R as a whole provides numerous tools and packages for predictive modeling, so it’s the right tool for a data scientist.