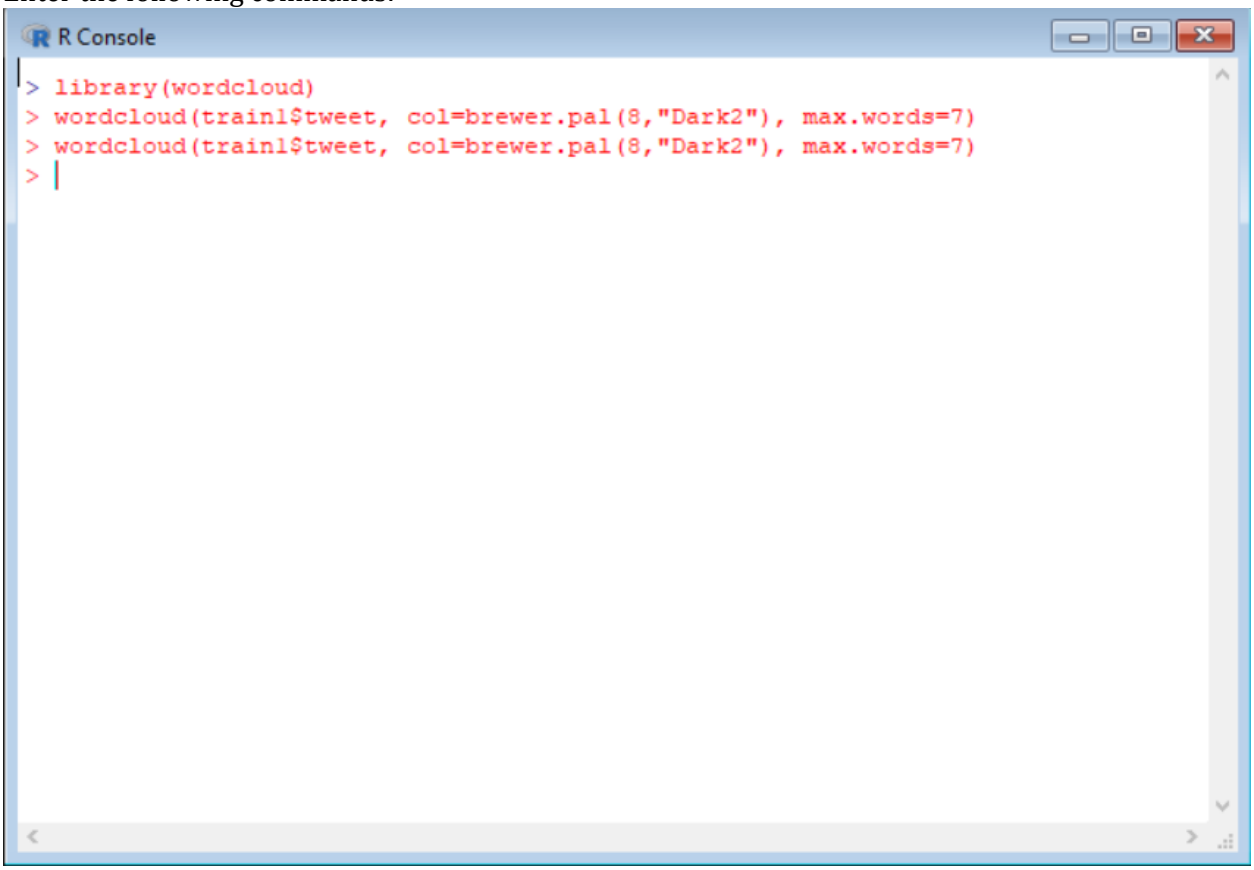
Lecture Notes on Neural Networks: Part 2C-Advanced — WordCloud the Hard Column, Determine What to Keep, Create the Combined Training Set

Nick V. Flor, nickflor@unm.edu

In the previous tutorial, we imported *train.csv* into R, then encoded the "easy" columns. Next, we encode the hard column, which means to convert the tweets into numbers.

Step 1. WordCloud to Find the Top N Words

Enter the following commands:



Explanation:

I did not go through the steps in detail, because you've already loaded the wordcloud package and done wordclouds in a previous homework.

The parameter *max.words* returns the top N words. In this tutorial, I want to use the top 5 words, but I entered 7 for more flexibility in choosing the words with which to train the network. I entered *wordcloud* twice, just to get a nice image.

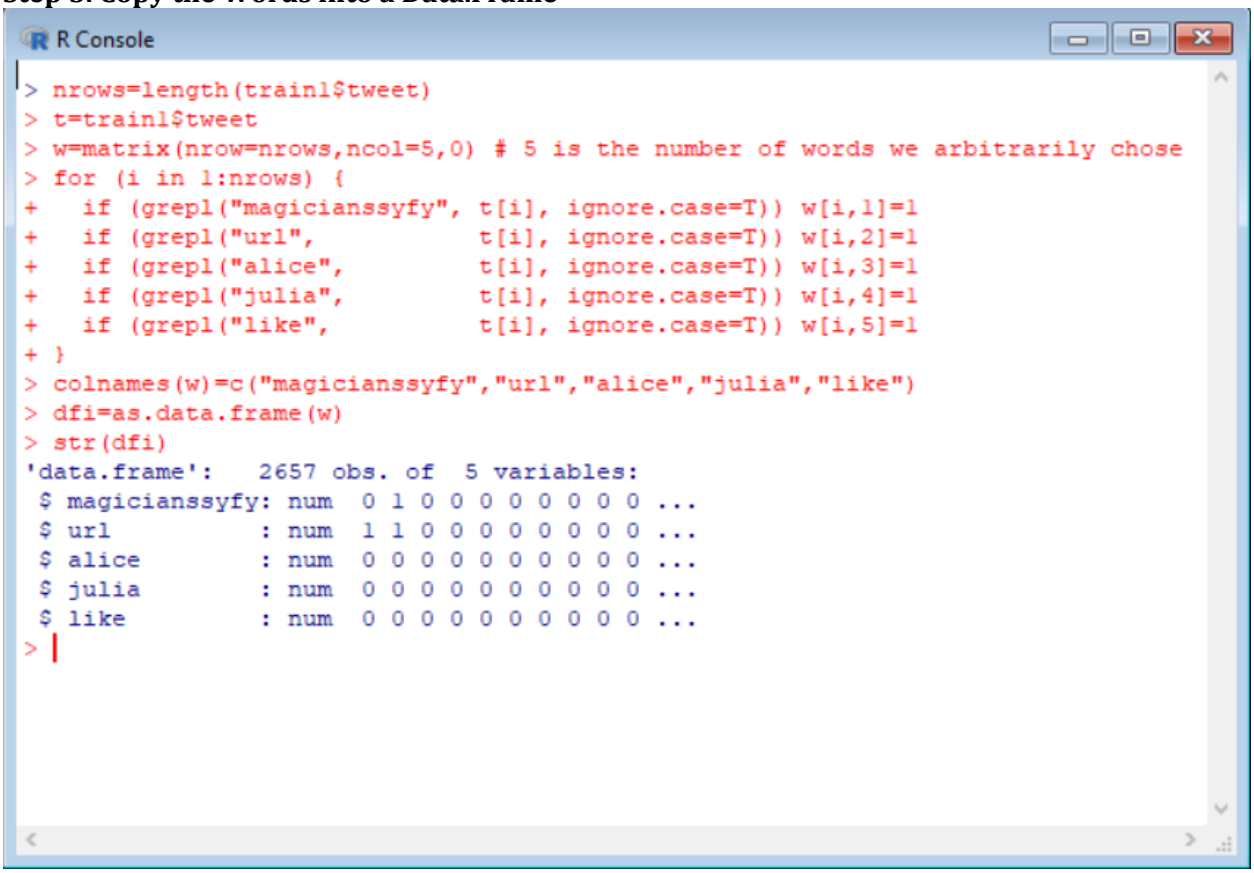
Step 2. Determine What to Keep — Choose Words from the Word Cloud
The previous command returned the word cloud below. From this word cloud, let us choose to train the network on five words: *url, magicianssyfy, julia, alice,* and *like.*

Explanation:

The five words is an arbitrarily low number I choose for teaching purpose. In practice, choose as many words as you feel are needed to train your network. I did not choose "themagicians", because that is the hashtag and it is on every tweet. I did a judgement call on the word "just" — it didn't seem as important as "like". The rest seemed significant in making a tweet go viral — mentioning the social media account, mentioning actresses, having a url, and emphasizing "like".

Key point: We are going to train the network on whether or not every tweet has the five words. And remember five is an arbitrary number.

Step 3. Copy the Words into a Data.Frame



Explanation:
Yes, this was a complex piece of code, so please bear with me. I'll go through each line

```
nrows=length(train1$tweet)
```

Determine how many rows of tweets we have in the *train1* dataset.

```
t=train1$tweet
```

To save on typing, I copy *train1$tweet* into a variable *t.*

```
w=matrix(nrow=nrows,ncol=5,0) # 5 is the number of words we arbitrarily chose
```

is a comment. I create a matrix with *nrows* rows by *5* columns, initializing the matrix to zero.

```
for (i in 1:nrows) {
  if (grepl("magicianssyfy", t[i], ignore.case=T)) w[i,1]=1
  if (grepl("url",           t[i], ignore.case=T)) w[i,2]=1
  if (grepl("alice",         t[i], ignore.case=T)) w[i,3]=1
  if (grepl("julia",         t[i], ignore.case=T)) w[i,4]=1
  if (grepl("like",          t[i], ignore.case=T)) w[i,5]=1
}
```

Go through every row, *i*, in the dataset, *t*, and put a 1 in the appropriate column (1 through 5) if the word appears in the tweet.

```
colnames(w)=c("magicianssyfy","url","alice","julia","like")
```
This is like adding headers to columns in Excel.

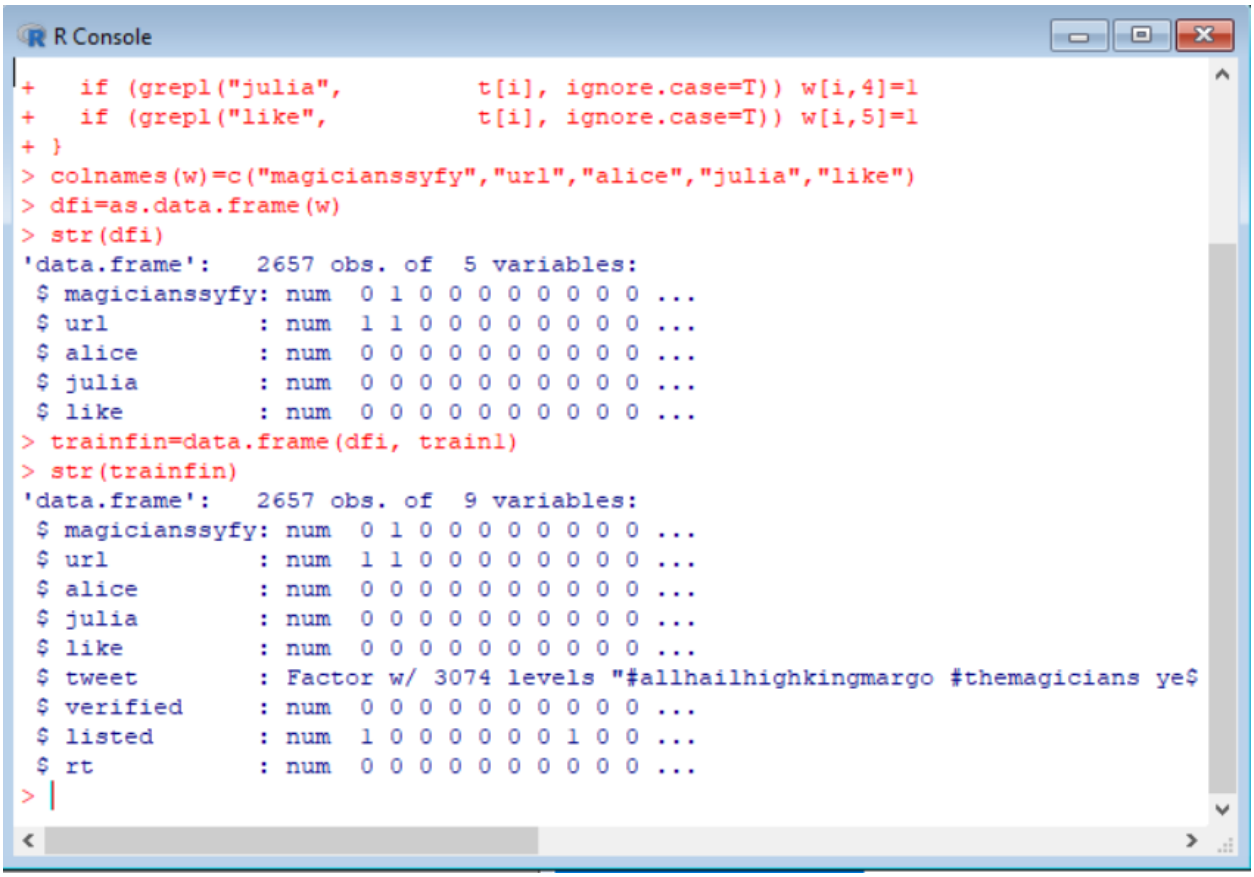
```
dfi=as.data.frame(w)
```

w is a matrix, and we need to convert it to a *data.frame*, which I called *dfi*.

```
str(dfi)
```
this just lists the structure of dfi

Bottom line: We started with *train* as our data frame, removed any tweet that was a retweet AND converted True/False & columns with levels to ones and zeros, THEN we saved this trimmed data frame as *train1*. Finally, we created yet another data frame, based on the tweets, that contained the words that we wanted to train our network with (in this case we only have five words). That data frame is *dfi.*

Step 4. Combine Data.Frames (train1, dfi) into the Final Training Set (trainfin)

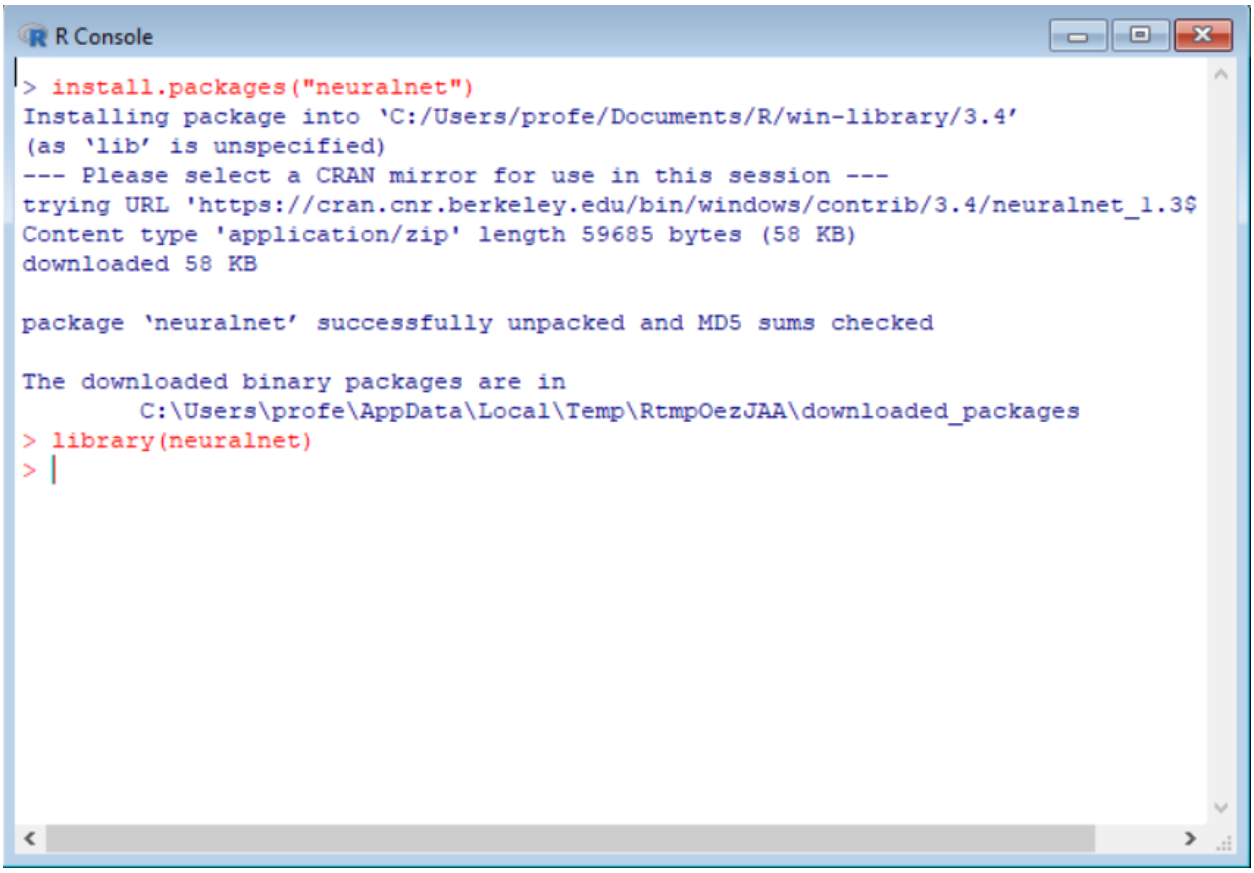


Explanation:

We use a *data.frame* to combine the trimmed Excel data frame (*train1*) with the word data frame (*dfi*). The final data frame is named *trainfin*.

(continued)

Step 5. Load the NeuralNet Package

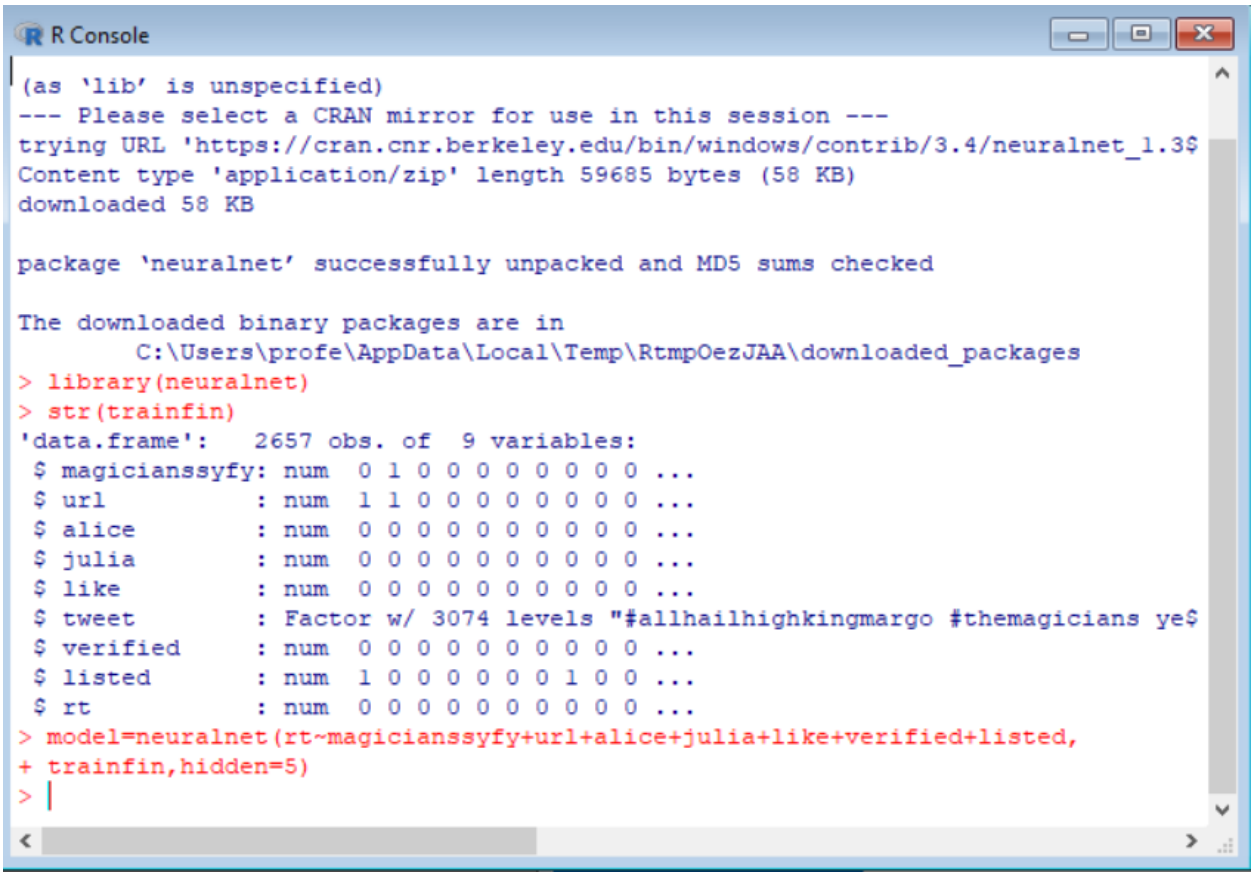


Explanation:

The package is *neuralnet*. Don't forget to use *install.packages* the first time, in order to load the package.

(continued)

Step 6. Train the NeuralNet



Explanation:
The key command is —

```
model=neuralnet(rt~magicianssyfy+url+alice+julia+like+verified+listed,
trainfin,hidden=5)
```

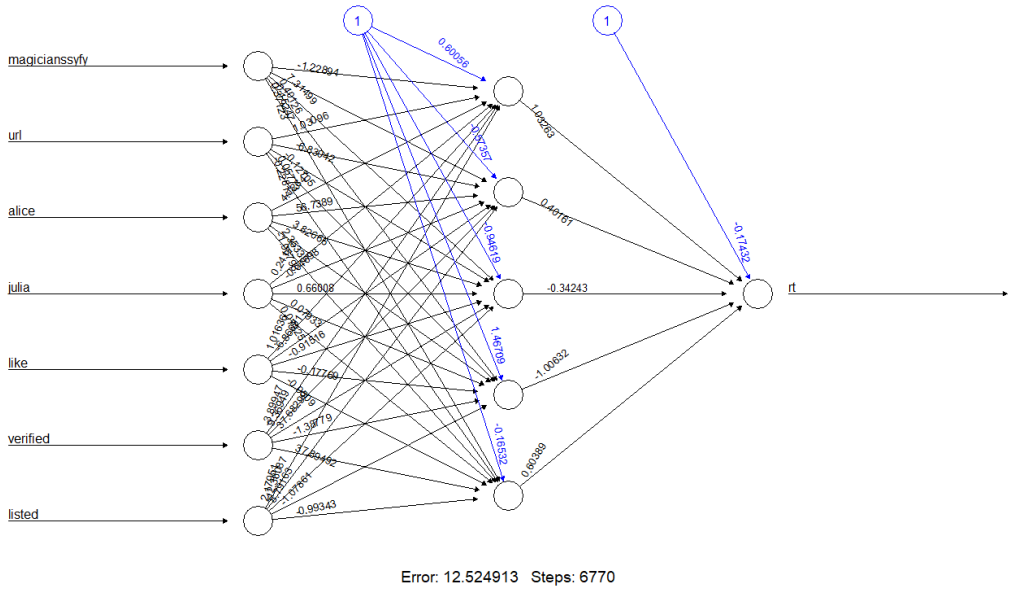
—

— the Syntax is similar to a regression. The only difference is that you have to specify the number of hidden units (*hidden*).

(continued)

Step 7. Plot The Model

- Type *plot(model)*

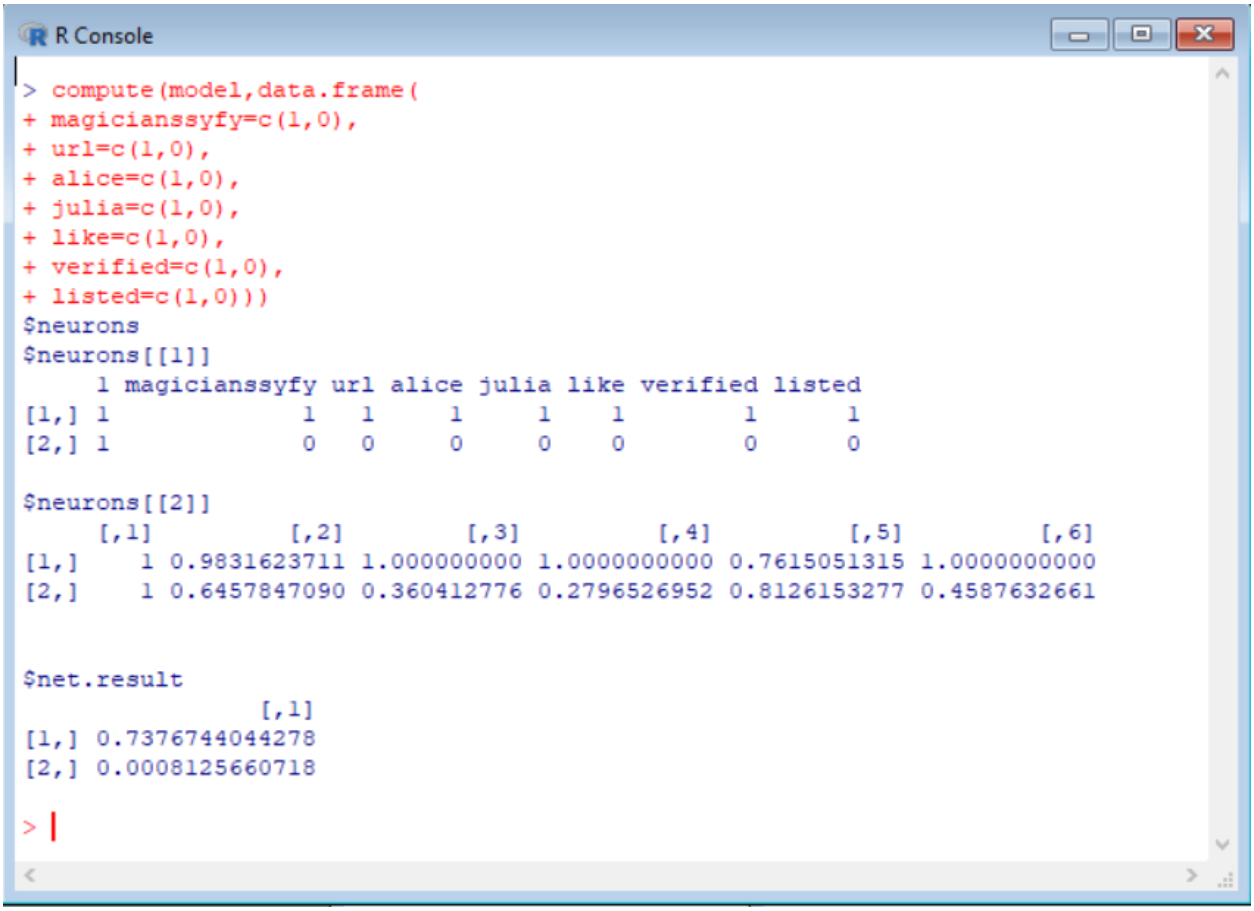


Explanation:

Use the *plot* command to draw the *model*. Note the error is not less than 1, which means we need to do more training or add more hidden units. I'll leave that to you.

(continued)

Step 8. Test the Model With New Input



Explanation:

Regressions in R use the predict function, neural networks in R use *compute.* The syntax is the same. $net.result is the output. I tried two test cases, one denoting all the words in a tweet (all 1's) and the other denoting none of the words in a tweet (all 0's). The network predicted that all the words in a tweet would probably go viral (.7376, which is close to 1), and that none of the words in a tweet would probably not go viral (.0008, which is close to 0).