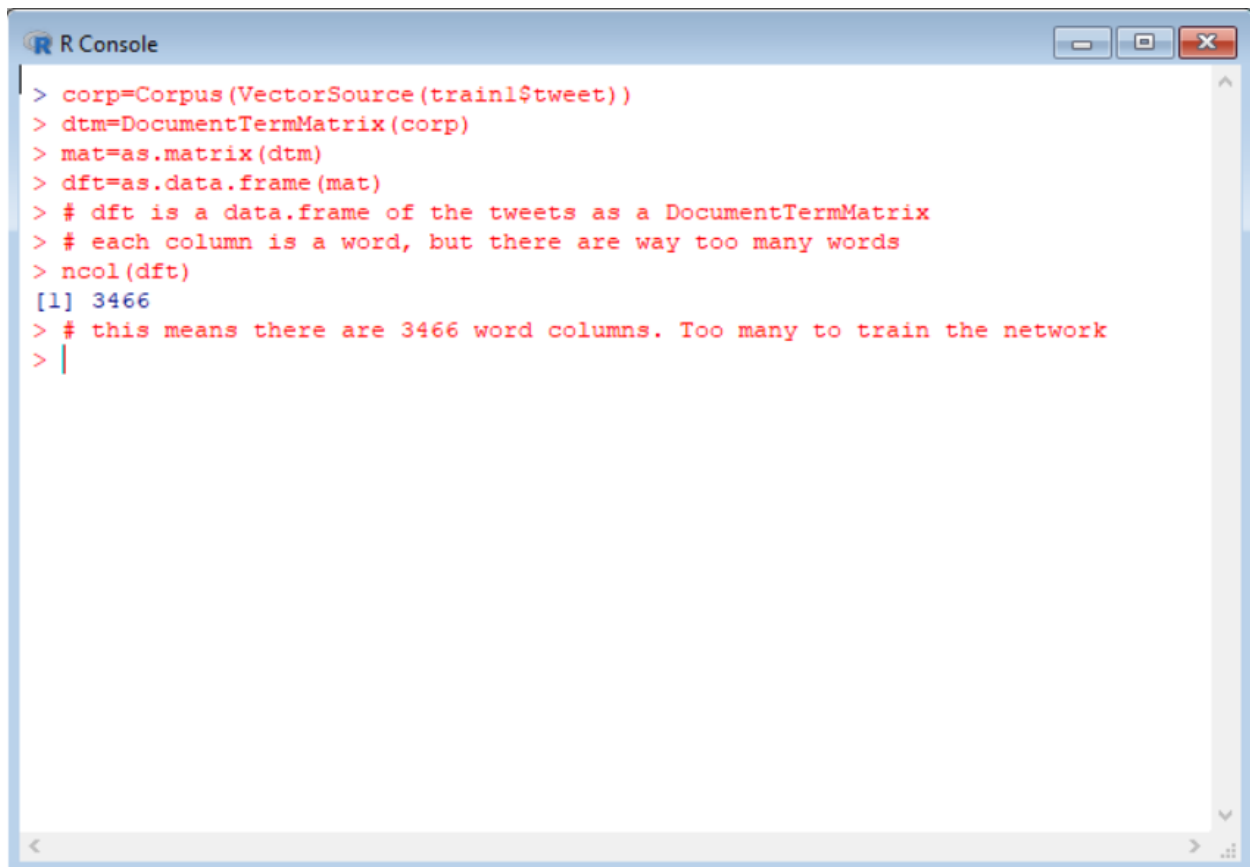


Lecture Notes on Neural Networks: Part 2C, Encoding the Hard Column, Determine What to Keep, Create the Combined Training Set

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. Encode the Tweet Column into a Document Term Matrix Data Frame



```
> corp=Corpus(VectorSource(train1$tweet))
> dtm=DocumentTermMatrix(corp)
> mat=as.matrix(dtm)
> dft=as.data.frame(mat)
> # dft is a data.frame of the tweets as a DocumentTermMatrix
> # each column is a word, but there are way too many words
> ncol(dft)
[1] 3466
> # this means there are 3466 word columns. Too many to train the network
> |
```

Explanation:

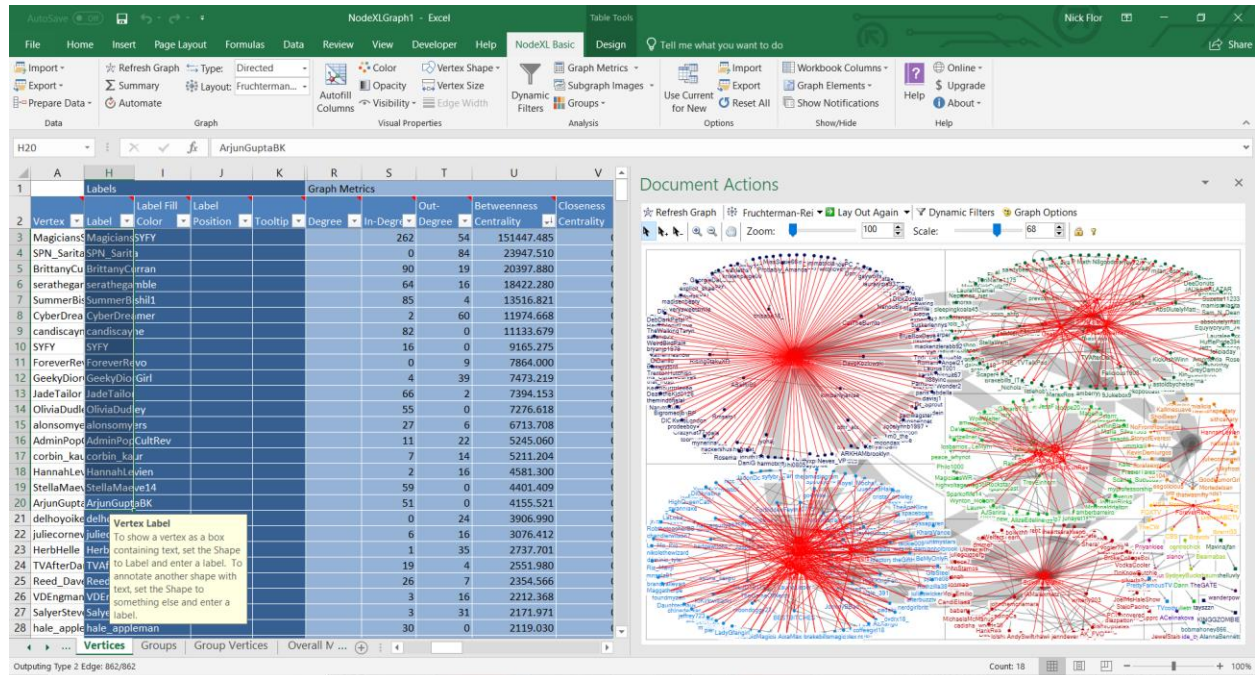
I did not go through the steps in detail, because you’ve already created a Document Term Matrix in a previous homework on clustering. The document term matrix denotes, for each tweet, whether or not that tweet contains a word in the entire spike corpus.

The only new statement is *dft=as.data.frame(mat)*. This converts a matrix (*mat*) into a data frame (*dft*).

Our ultimate goal is to combine part of the *dft* data.frame with the *train1* data.frame. I say “part of” because there are 3466 columns in *dft*. So we need to trim down the columns.

Bottom Line: We have a data frame containing a document term matrix, which denotes which tweets contain the various words in the spike corpus.

Step 2. Determine What to Keep — Do a Social Network Analysis to Help Trim Down the Document Term Matrix Columns



Explanation:

I won't walk you through this step because it's just doing a social network analysis (SNA) on the spike (similar to an earlier homework) — calculating metrics, grouping by cluster, then sorting by Betweenness Centrality.

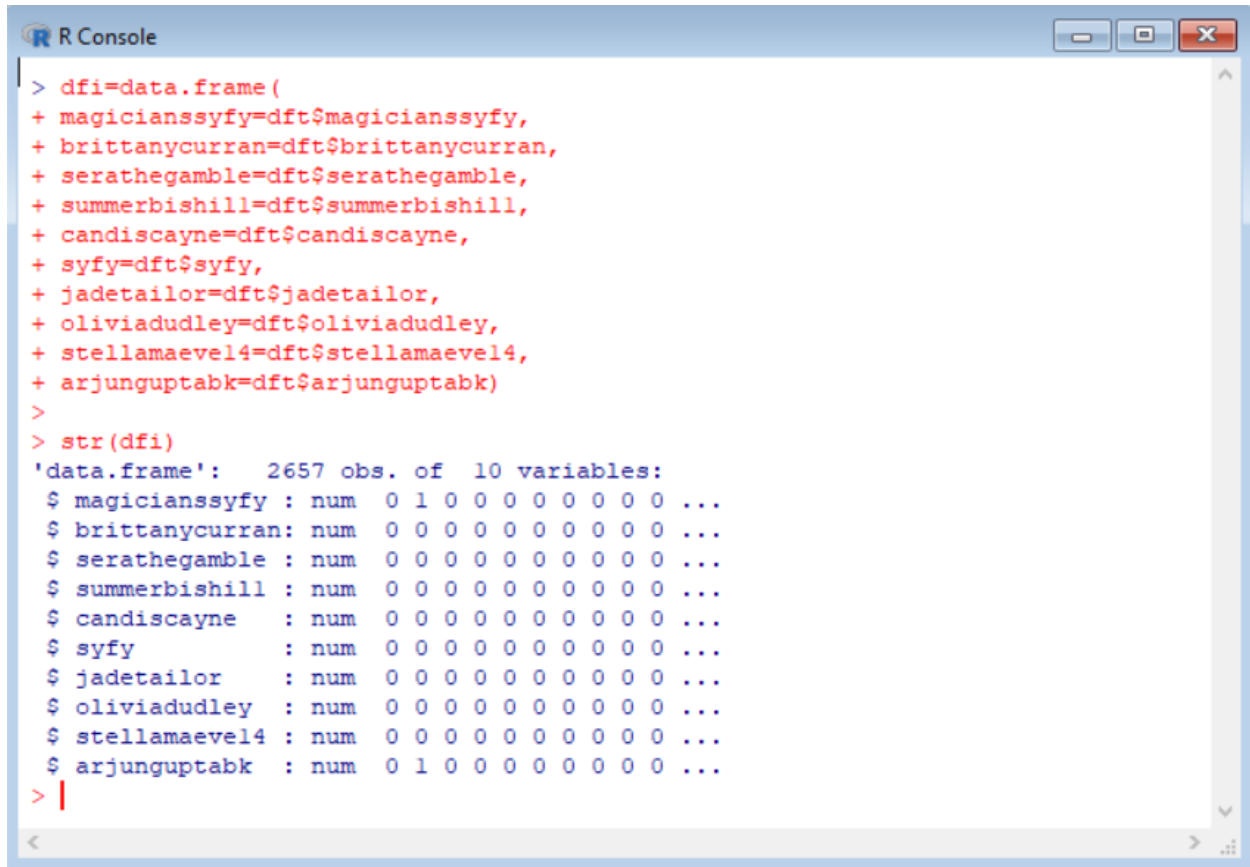
Aside. An easier alternative to help trim down the columns, is to calculate word frequencies and only use the top words. The hypothesis is that high frequency words are associated with viral tweets. However, I decided to do a social networking analysis, with the hypothesis that viral tweets are associated with mentioning popular people. A better alternative would be to combine both high-frequency words AND popular people. But we only have time for one.

Note: the top 20 influencers are: MagiciansSYFY, SPN_Sarita, BrittanyCurran, serathegamble, SummerBishil1, CyberDreamer, candiscayne, SYFY, ForeverRevo, GeekyDiorGirl, JadeTailor, OliviaDudley, alonsomyers, AdminPopCultRev, corbin_kaur, HannahLevien, StellaMaeve14, ArjunGuptaBK.

I checked the document term matrix to see if these users were mentioned (by typing `dft$username`; username in lower case). Of these 20, only 10 appeared in the document term matrix: `magicianssyfy`, `brittanycurran`, `serathegamble`, `summerbishil1`, `candiscayne`, `syfy`, `jadetailor`, `oliviadudley`, `stellamaeve14`, `arjunguptabk`.

Bottom line: We can reduce the 3466 word columns down to 10 or fewer.

Step 3. Copy the Names from the Document Term Matrix into a DataFrame



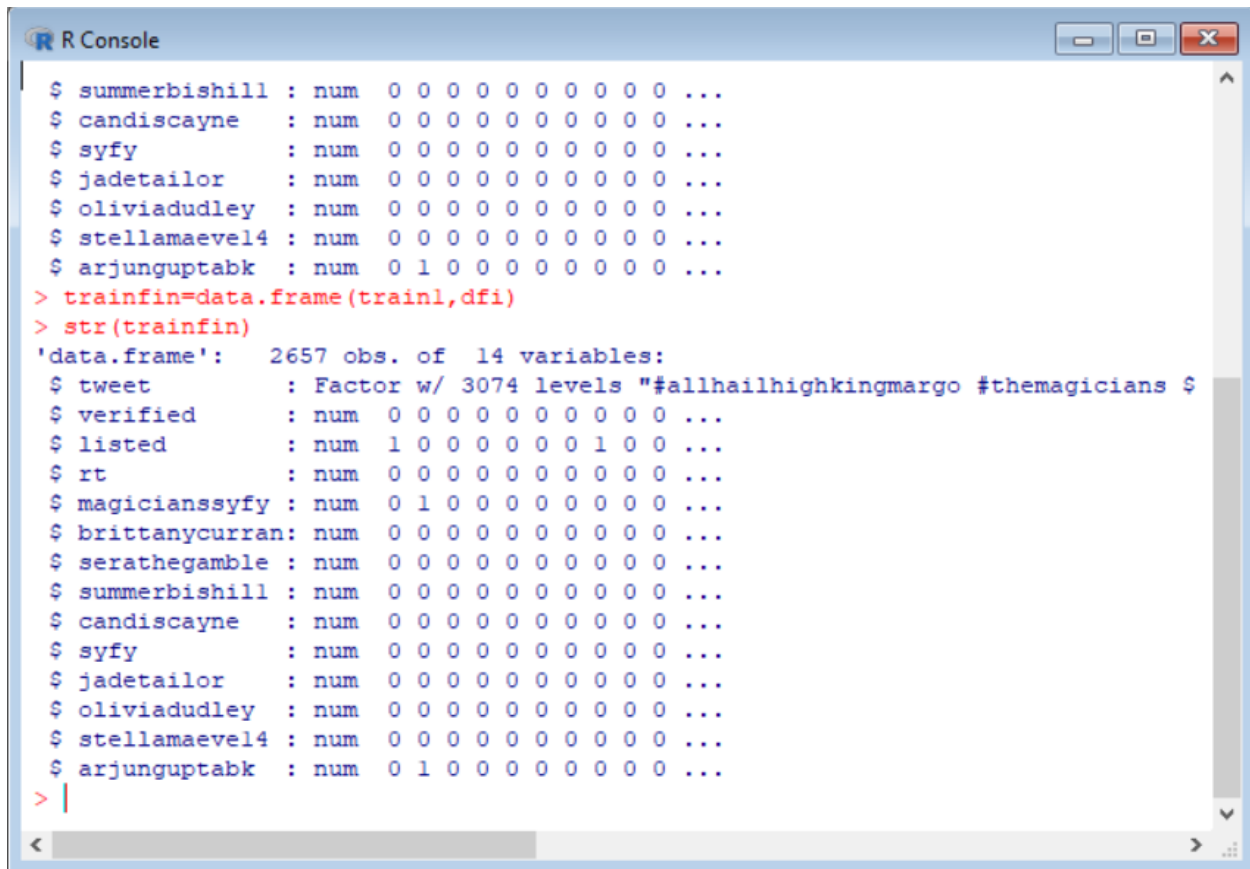
```
> dfi=data.frame (
+ magicianssyfy=dft$magicianssyfy,
+ brittanycurran=dft$brittanycurran,
+ serathegamble=dft$serathegamble,
+ summerbishill=dft$summerbishill,
+ candiscayne=dft$candiscayne,
+ syfy=dft$syfy,
+ jadetailor=dft$jadetailor,
+ oliviadudley=dft$oliviadudley,
+ stellamaevel4=dft$stellamaevel4,
+ arjunguptabk=dft$arjunguptabk)
>
> str(dfi)
'data.frame':  2657 obs. of  10 variables:
 $ magicianssyfy : num  0 1 0 0 0 0 0 0 0 0 ...
 $ brittanycurran: num  0 0 0 0 0 0 0 0 0 0 ...
 $ serathegamble : num  0 0 0 0 0 0 0 0 0 0 ...
 $ summerbishill : num  0 0 0 0 0 0 0 0 0 0 ...
 $ candiscayne   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ syfy          : num  0 0 0 0 0 0 0 0 0 0 ...
 $ jadetailor    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ oliviadudley  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ stellamaevel4 : num  0 0 0 0 0 0 0 0 0 0 ...
 $ arjunguptabk  : num  0 1 0 0 0 0 0 0 0 0 ...
> |
```

Explanation:

The original document term matrix (dtm) had 3466 columns, we have reduced this to 10 columns.

(continued)

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



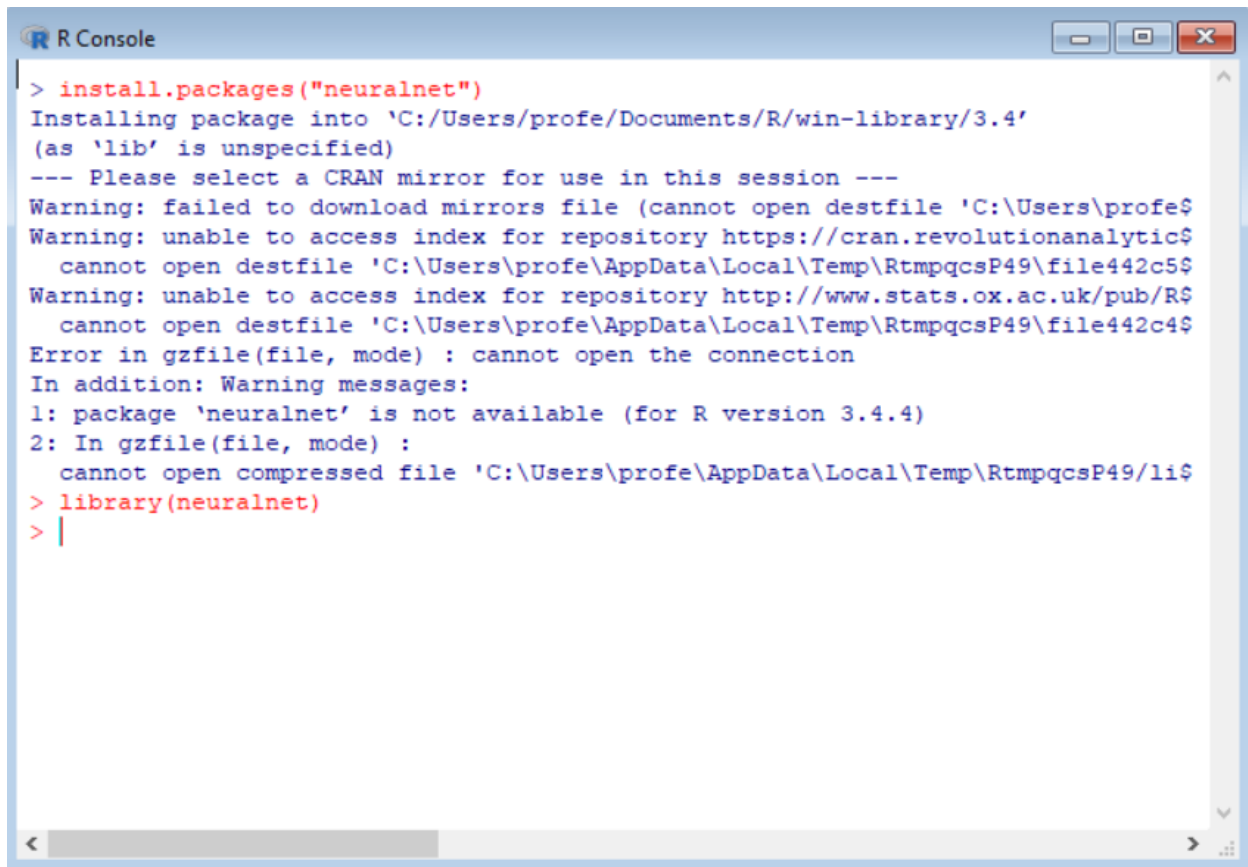
```
R Console
$ summerbishill : num 0 0 0 0 0 0 0 0 0 0 ...
$ candiscayne   : num 0 0 0 0 0 0 0 0 0 0 ...
$ syfy          : num 0 0 0 0 0 0 0 0 0 0 ...
$ jadetailor    : num 0 0 0 0 0 0 0 0 0 0 ...
$ oliviadudley  : num 0 0 0 0 0 0 0 0 0 0 ...
$ stellamaevel4 : num 0 0 0 0 0 0 0 0 0 0 ...
$ arjunguptabk  : num 0 1 0 0 0 0 0 0 0 0 ...
> trainfin=data.frame(train1,dfi)
> str(trainfin)
'data.frame': 2657 obs. of 14 variables:
 $ tweet      : Factor w/ 3074 levels "#allhailhighkingmargo #themagicians $
 $ verified   : num 0 0 0 0 0 0 0 0 0 0 ...
 $ listed     : num 1 0 0 0 0 0 0 0 1 0 ...
 $ rt         : num 0 0 0 0 0 0 0 0 0 0 ...
 $ magicianssyfy : num 0 1 0 0 0 0 0 0 0 0 ...
 $ brittanycurran: num 0 0 0 0 0 0 0 0 0 0 ...
 $ serathegamble : num 0 0 0 0 0 0 0 0 0 0 ...
 $ summerbishill : num 0 0 0 0 0 0 0 0 0 0 ...
 $ candiscayne   : num 0 0 0 0 0 0 0 0 0 0 ...
 $ syfy         : num 0 0 0 0 0 0 0 0 0 0 ...
 $ jadetailor    : num 0 0 0 0 0 0 0 0 0 0 ...
 $ oliviadudley  : num 0 0 0 0 0 0 0 0 0 0 ...
 $ stellamaevel4 : num 0 0 0 0 0 0 0 0 0 0 ...
 $ arjunguptabk  : num 0 1 0 0 0 0 0 0 0 0 ...
> |
```

Explanation:

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

(continued)

Step 5. Load the NeuralNet Package



```
> install.packages("neuralnet")
Installing package into 'C:/Users/profe/Documents/R/win-library/3.4'
(as 'lib' is unspecified)
--- Please select a CRAN mirror for use in this session ---
Warning: failed to download mirrors file (cannot open destfile 'C:\Users\profe$
Warning: unable to access index for repository https://cran.revolutionanalytic$
cannot open destfile 'C:\Users\profe\AppData\Local\Temp\RtmpqcsP49\file442c5$
Warning: unable to access index for repository http://www.stats.ox.ac.uk/pub/R$
cannot open destfile 'C:\Users\profe\AppData\Local\Temp\RtmpqcsP49\file442c4$
Error in gzfile(file, mode) : cannot open the connection
In addition: Warning messages:
1: package 'neuralnet' is not available (for R version 3.4.4)
2: In gzfile(file, mode) :
cannot open compressed file 'C:\Users\profe\AppData\Local\Temp\RtmpqcsP49/li$
> library(neuralnet)
> |
```

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

```
R Console
> str(trainfin)
'data.frame': 2657 obs. of 14 variables:
 $ tweet      : Factor w/ 3074 levels "#allhailhighkingmargo #themagicians $
 $ verified   : num 0 0 0 0 0 0 0 0 0 0 ...
 $ listed     : num 1 0 0 0 0 0 0 1 0 0 ...
 $ rt         : num 0 0 0 0 0 0 0 0 0 0 ...
 $ magicianssyfy : num 0 1 0 0 0 0 0 0 0 0 ...
 $ brittanycurran: num 0 0 0 0 0 0 0 0 0 0 ...
 $ serathegamble : num 0 0 0 0 0 0 0 0 0 0 ...
 $ summerbishill : num 0 0 0 0 0 0 0 0 0 0 ...
 $ candiscayne  : num 0 0 0 0 0 0 0 0 0 0 ...
 $ syfy        : num 0 0 0 0 0 0 0 0 0 0 ...
 $ jadetailor   : num 0 0 0 0 0 0 0 0 0 0 ...
 $ oliviadudley : num 0 0 0 0 0 0 0 0 0 0 ...
 $ stellamaevel4 : num 0 0 0 0 0 0 0 0 0 0 ...
 $ arjunguptabk : num 0 1 0 0 0 0 0 0 0 0 ...

> model=neuralnet(rt~verified+listed+magicianssyfy+brittanycurran+
+ serathegamble+summerbishill+candiscayne+syfy+jadetailor+oliviadudley+
+ stellamaevel4+arjunguptabk,trainfin, hidden=5)
> |
```

Explanation:

The key command is —

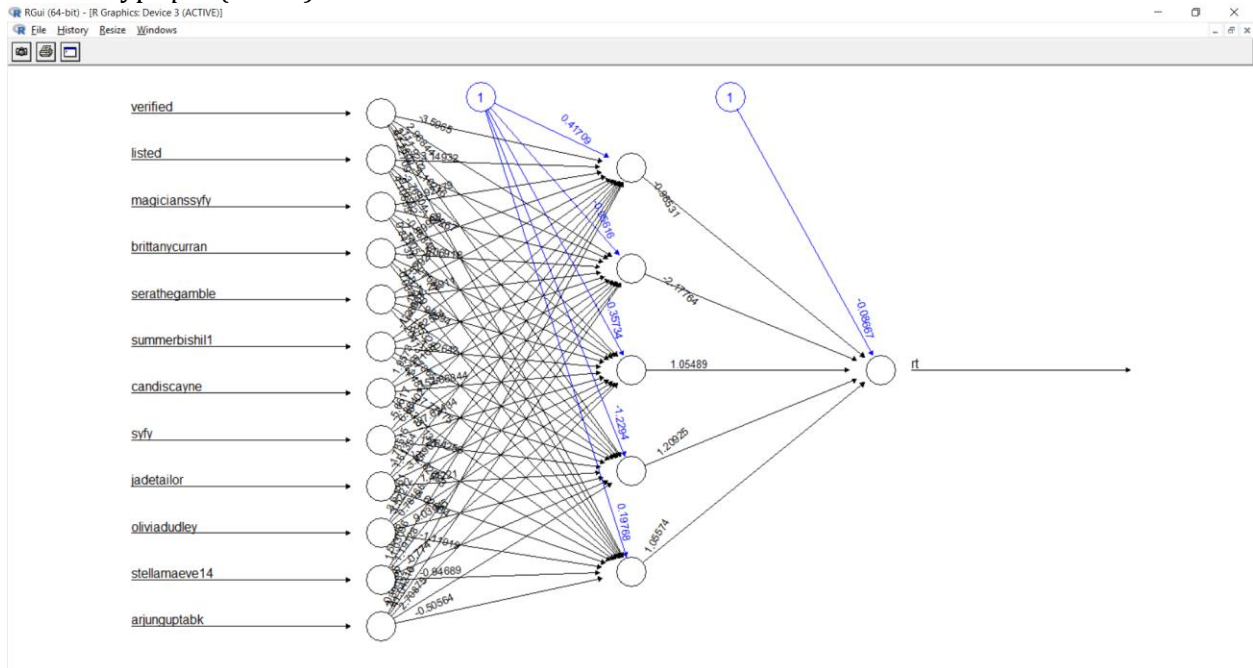
```
model=neuralnet(rt~verified+listed+magicianssyfy+brittanycurran+
serathegamble+summerbishill+candiscayne+syfy+jadetailor+oliviadudley+
stellamaevel4+arjunguptabk,trainfin, hidden=5)
```

— 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*.

(continued)

Step 8. Test the Model With New Input

```
R Console
> compute(model, data.frame(
+ verified=c(1),
+ listed=c(1),
+ magicianssyfy=c(1),
+ brittanycurran=c(1),
+ serathegamble=c(1),
+ summerbishill=c(1),
+ candiscayne=c(1),
+ syfy=c(1),
+ jadetailor=c(1),
+ oliviadudley=c(1),
+ stellamaevel4=c(1),
+ arjunguptabk=c(1)))
$neurons
$neurons[[1]]
  1 verified listed magicianssyfy brittanycurran serathegamble summerbishill candiscay$
[1,] 1          1          1              1              1              1          1          $
  arjunguptabk
[1,]          1

$neurons[[2]]
  [,1]          [,2]          [,3] [,4]          [,5]          [,6]
[1,]  1 0.0002365435065 0.9999612941  1 0.999983209 0.000000000000000005983061575

$net.result
          [,1]
[1,] -0.0003323009734
> |
```

Explanation:

Regressions in R use the predict function, neural networks in R use *compute*. The syntax is the same. `$net.result` is the output.