SAM-R: Spreading of Activation Machine in R

CSQ Siew

2017-10-26

Load required libraries and data:

```
library(igraph)
library(dplyr)
library(samr)
library(rio)
load('gc.net.RData') # giant component of the phonological network
load('hml.RData')    # contains list of words in the HML with the label
```

Brief description:

So I finally decided to take another stab at this. My initial attempts resulted in very inefficient code that took a long time to run. The entire approach to the SAM is now revamped so that it can be scaled up substantially, runs more efficiently, and hopefully will form the basis of a future network model of spoken word recognition. In a nutshell, SAM-R is a home-made R package to simulate the spreading of activation among nodes in a network and to output the data for subsequent analysis.

A key improvement of this version is that the user can specify the initial conditions of the simulation by providing a dataframe with `words` and `activation` columns.

Consider Example 1 below. The initial conditions are:

- Give the "aback" node an initial activation of 100 units.
- Let activation spread for 5 time steps.
- Each node is allowed to retain 50% of its initial activation after each time step.
- The spreading of activation will occur within the giant component of the phonological network, as specified by the `gc.net` igraph object.

```
# Example 1
words <- c('xb@k;aback')
activation <- c(100)
start_1_word <- data.frame(words, activation, stringsAsFactors=FALSE)

butter(start_1_word, 0.5, gc.net, 5)

##          words activation time
## 1     b@k;back  16.666667    1
## 2   xb@k;aback  50.000000    1
```

```
## 3   xb@S;abash  16.666667    1
## 4  xt@k;attack  16.666667    1
## 5     b@k;back  16.666667    2
## 6     b@S;bash   4.427083    2
## 7   st@k;stack   2.083333    2
## 8     t@k;tack   2.343750    2
## 9   xb@k;aback  31.510417    2
## 10  xb@S;abash  16.666667    2
## 11 xt@C;attach   2.083333    2
## 12 xt@k;attack  16.666667    2
## 13    b@k;back  13.708973    3
## 14    b@S;bash   6.640625    3
## 15  st@k;stack   3.156672    3
## 16    t@k;tack   3.580729    3
## 17  xb@k;aback  22.265625    3
## 18  xb@S;abash  13.677300    3
## 19 xt@C;attach   3.125000    3
## 20 xt@k;attack  14.723513    3
## 21    b@k;back  10.752158    4
## 22    b@S;bash   6.953840    4
## 23  st@k;stack   3.467163    4
## 24    t@k;tack   3.943652    4
## 25  xb@k;aback  16.606779    4
## 26  xb@S;abash  10.687934    4
## 27 xt@C;attach   3.402939    4
## 28 xt@k;attack  12.782228    4
## 29    b@k;back   8.342040    5
## 30    b@S;bash   6.316906    5
## 31  st@k;stack   3.384653    5
## 32    t@k;tack   3.845956    5
## 33  xb@k;aback  12.741154    5
## 34  xb@S;abash   8.256635    5
## 35 xt@C;attach   3.299248    5
## 36 xt@k;attack  11.022022    5
```

One thing to take note of is that when specifying `words` I used the labels specified in the `gc.net`. I like that it has both phonological and orthographic representations so that I can check the code while knowing the heck these words are. Hence it might be useful to have the `hml.RData` on hand when selecting words.

```
head(hml)
```

```
##     Phono    Ortho            label
## 1       x        a              x;a
## 2 ardvark aardvark ardvark;aardvark
## 3    xb@k    aback       xb@k;aback
## 4  @bxkxs   abacus    @bxkxs;abacus
## 5   xb@ft    abaft      xb@ft;abaft
## 6 @bxloni  abalone  @bxloni;abalone
```

Another example:

Sometimes, one might want to specify more nodes to have initial activations--especially in the context of spoken word recognition where a node's neighbors are partially activated (due to overlap in phonemes that are activated by the incoming acoustic information). The way that the input for `butter()` function is set up allows the user to do this easily.

Consider Example 2 below. The initial conditions are:

- Give the "aback" node an initial activation of 100 units.
- The neighbors of "aback" get an initial activation of 50 units.
- Let activation spread for 5 time steps.
- Each node is allowed to retain 80% of its initial activation after each time step.
- The spreading of activation will occur within the giant component of the phonological network, as specified by the `gc.net` igraph object.

```
# Example 2
words <- c('xb@k;aback', 'xb@S;abash', 'xt@k;attack', 'b@k;back') # include
neighbors
activation <- c(100, 50, 50, 50)
start_1_word_w_neighbors <- data.frame(words, activation,
stringsAsFactors=FALSE)

butter(start_1_word_w_neighbors, 0.8, gc.net, 5)

##          words activation time
## 1     b@k;back  46.666667    1
## 2     b@S;bash   5.312500    1
## 3   st@k;stack   2.500000    1
## 4     t@k;tack   2.812500    1
## 5   xb@k;aback  87.812500    1
## 6   xb@S;abash  46.666667    1
## 7  xt@C;attach   2.500000    1
## 8  xt@k;attack  46.666667    1
## 9     b@k;back  43.246974    2
## 10    b@S;bash   9.208333    2
## 11  st@k;stack   4.348536    2
## 12    t@k;tack   4.906250    2
## 13  xb@k;aback  77.541667    2
## 14  xb@S;abash  43.231771    2
## 15 xt@C;attach   4.333333    2
## 16 xt@k;attack  43.733953    2
## 17    b@k;back  39.870280    3
## 18    b@S;bash  11.960137    3
## 19  st@k;stack   5.692047    3
## 20    t@k;tack   6.436348    3
## 21  xb@k;aback  68.813502    3
## 22  xb@S;abash  39.831597    3
```

```
## 23 xt@C;attach   5.653364    3
## 24 xt@k;attack  41.104150    3
## 25    b@k;back  36.618249    4
## 26    b@S;bash  13.800459    4
## 27  st@k;stack   6.643636    4
## 28    t@k;tack   7.524626    4
## 29  xb@k;aback  61.338358    4
## 30  xb@S;abash  36.552512    4
## 31 xt@C;attach   6.577899    4
## 32 xt@k;attack  38.707501    4
## 33    b@k;back  33.539501    5
## 34    b@S;bash  14.924482    5
## 35  st@k;stack   7.290958    5
## 36    t@k;tack   8.266985    5
## 37  xb@k;aback  54.890177    5
## 38  xb@S;abash  33.446238    5
## 39 xt@C;attach   7.197694    5
## 40 xt@k;attack  36.494524    5
```

Work flow example:

If you want to simulate a bunch of words, it is easy to set up a workflow to run the simulations and save the results to an external file for subsequent analyses.

Consider Example 3 below. The initial conditions are:

- Give the "aback" and "abash" nodes an initial activation of 100 units.

- Run the simulation twice for each node (independent of each other).
- Let activation spread for 5 time steps.
- Each node is allowed to retain 80% of its initial activation after each time step.
- The spreading of activation will occur within the giant component of the phonological network, as specified by the `gc.net` igraph object.

```
# make a dataset to simulate the spreading of activation independently for
two words
# each with an initial activation value of 100
words <- c('xb@k;aback', 'xb@S;abash')
activation <- c(100,100)
test <- data.frame(words, activation, stringsAsFactors=FALSE)

# use a for loop to go through each word in the test set and save the output
for (x in 1:nrow(test)) { # for each row in the test set

  bread <- butter(test[x, ], 0.8, gc.net, 5)

  # save the output as a .csv file
```

```
  # the target word is used as the file name
  write.csv(bread, file = paste0(test$words[x],'.csv'))

  # view the output in real time
  print(bread)
}
```

```
##          words activation time
## 1     b@k;back   6.666667    1
## 2   xb@k;aback  80.000000    1
## 3   xb@S;abash   6.666667    1
## 4  xt@k;attack   6.666667    1
## 5     b@k;back  10.666667    2
## 6   xb@k;aback  65.041667    2
## 7   xb@S;abash  10.666667    2
## 8  xt@k;attack  10.666667    2
## 9     b@k;back  12.869444    3
## 10    b@S;bash   1.133333    3
## 11  xb@k;aback  53.700000    3
## 12  xb@S;abash  12.869444    3
## 13 xt@k;attack  12.869444    3
## 14    b@k;back  13.885000    4
## 15    b@S;bash   2.274045    4
## 16  xb@k;aback  44.970851    4
## 17  xb@S;abash  13.885000    4
## 18 xt@k;attack  13.875556    4
## 19    b@k;back  14.125007    5
## 20    b@S;bash   3.294517    5
## 21  xb@k;aback  38.145740    5
## 22  xb@S;abash  14.125007    5
## 23 xt@k;attack  14.098501    5
##          words activation time
## 1     b@S;bash  10.000000    1
## 2   xb@k;aback  10.000000    1
## 3   xb@S;abash  80.000000    1
## 4     b@S;bash  16.000000    2
## 5   xb@k;aback  16.000000    2
## 6   xb@S;abash  64.750000    2
## 7     b@k;back   1.200000    3
## 8     b@S;bash  19.275000    3
## 9   xb@k;aback  19.275000    3
## 10  xb@S;abash  53.000000    3
## 11 xt@k;attack   1.066667    3
## 12    b@k;back   2.405625    4
## 13    b@S;bash  20.727500    4
## 14  xb@k;aback  20.780833    4
## 15  xb@S;abash  43.845625    4
## 16 xt@k;attack   2.138333    4
## 17    b@k;back   3.482618    5
## 18    b@S;bash  20.981598    5
```

```
## 19  xb@k;aback  21.131181    5
## 20  xb@S;abash  36.634618    5
## 21 xt@k;attack   3.096056    5
```

Note that this workflow only works if the initial input consists of one node (i.e., only 1 word was activated with 100 units initially).

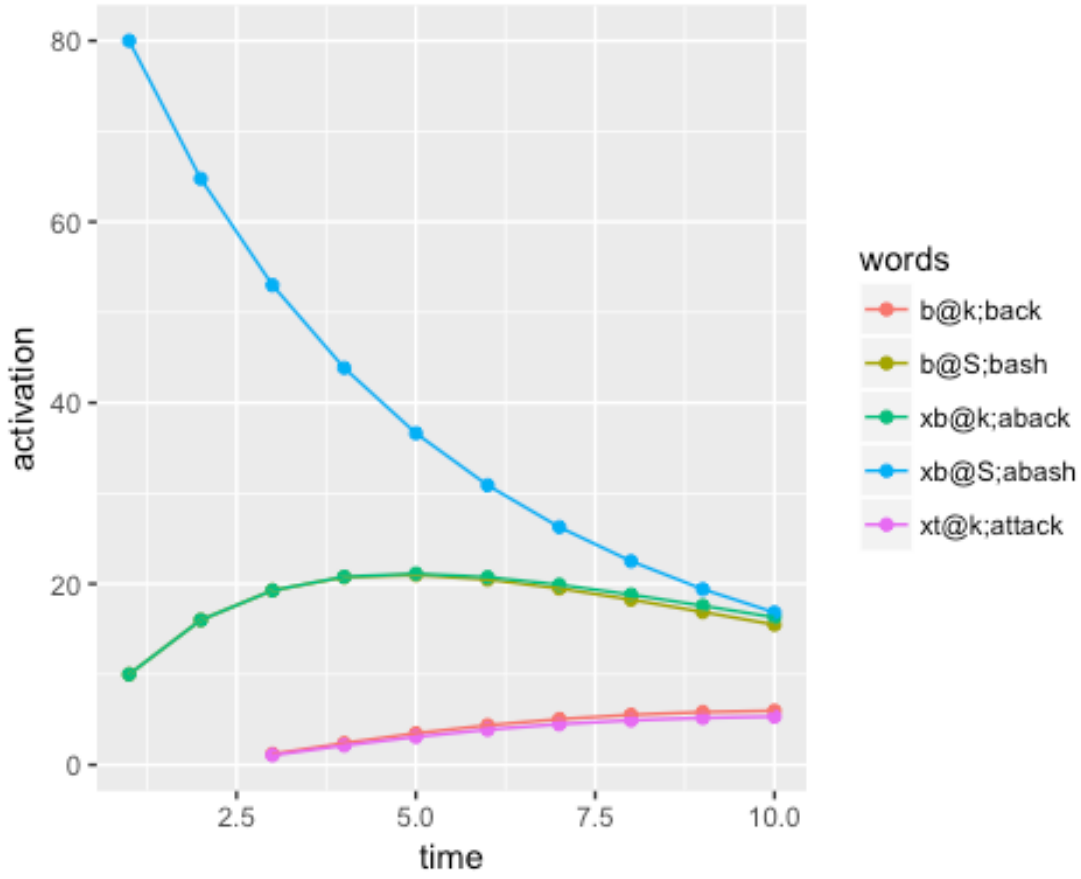
In the near future, the model could expand to allow for simulations with an initial activation 'space' (i.e., see Example 2 where a target word might receive 100 units, its neighbors receive 50 units each (depending on overlap of phonemes)).

Visualizing the results

```
library(ggplot2)

bread <- butter(test[2, ], 0.8, gc.net, 10) # target word = abash

ggplot(data = bread,  aes(x = time, y = activation, color = words, group =
words)) +
  geom_point() +
  geom_line()
```



A quick example to demonstrate how the activation results might be compiled.

One thing to look at (among others, like number of active nodes, relative difference between activation values among active nodes, etc.) is the final activation value of the target node.

```
words <- c('xb@k;aback', 'xb@S;abash')

final_act <- data.frame(words = vector(), activation = vector(), time =
vector(),
                        stringsAsFactors=FALSE)

for (i in 1:length(words)) {
  result <- import(paste0(words[i], '.csv')) # read in .csv file for a given
word
  result <- result[ , -1]                    # remove first column
  result <- result[which(result$words %in% words[i]), ] # extract rows with
the target
  result <- result[order(result$time, decreasing = T), ]    # order by the
latest run first
  final_act <- rbind(final_act, result[1, ])                # save the final
activation value
}

final_act

##         words activation time
## 21 xb@k;aback   38.14574    5
## 20 xb@S;abash   36.63462    5
```

Cynthia's to do list:

- Make a function that creates an activation space depending on phoneme overlap.
- Make helper functions to compile and analyze the simulation output.