LEAP: Constructing gene-coexpression networks for single-cell sequencing data using pseudo-time ordering

Alicia T. Specht and Jun Li

May 14, 2016

Abstract

Summary: To construct gene co-expression networks based on single-cell RNA-Sequencing data, we present an algorithm called LEAP, which utilizes the estimated pseudo-time information of the cells to find stronger associations between pairs of genes.
Contact: aspecht2@nd.edu

Contents

1 Introduction

Advances in sequencing technology now allow researchers to capture the expression profiles of individual cells. Several algorithms have been developed to attempt to account for these effects by determining a cell's so-called 'pseudo-time', or relative biological state of transition.

By applying these algorithms to single-cell sequencing data, we can sort cells into their pseudotemporal ordering based on gene expression. LEAP (Lag-based Expression Association for Pseudotime-series) then applies a time-series inspired lag-based correlation analysis to reveal linearly dependent genetic associations.

2 Data format

LEAP takes a data matrix for which the rows are genes and the columns are experiments, sorted by their pseudo-time. For example, consider this dataset consisting 20 genes from a dataset of high throughput single-cell RNA sequencing counts of *Mus musculus* dendritic cells(Shalek *et al*, 2014):

```
> library("LEAP")
> example_data[,1:5]
```

```
   X..0.000000 X..1.056273 X..1.402318 X..1.607550 X..1.779713
1     5.445686    5.102184   0.0000000   2.9839979   0.0000000
2     4.526036    6.532284   6.0904955   5.6275842   6.1838837
3     6.528766    6.214955   0.0000000   3.2003647   6.4477135
4     7.252316    6.627344   0.0000000   1.6035663   0.0000000
5     0.000000    5.209410   5.5418589   5.5792501   2.8199493
6     0.000000    0.000000   0.0000000   5.8262664   3.4519732
7     3.253274    2.959225   3.9272823   6.3940181   6.2686141
8     0.000000    0.000000   0.9958159   1.3969442   5.0141277
9     6.685728    6.414861   6.6039581   6.0960811   5.8442669
10    0.000000    3.201347   5.9733666   6.4752538   5.1608972
11    0.000000    0.000000   2.8945397   5.3855163   0.0000000
12    0.000000    0.000000   0.0000000   5.9502785   0.0000000
13    0.000000    5.413914   5.6621031   0.0000000   5.2452502
14    0.000000    3.174056   0.0000000   3.2505840   6.6724467
15    6.664693    7.092854   0.0000000   0.0000000   6.7583925
16    1.035870    7.706152   0.0000000   0.9082936   6.6226587
17    5.773570    4.750586   0.0000000   0.0000000   0.9529196
18    3.884139    4.667165   0.0000000   5.6001221   4.5408934
19    6.288354    6.437330   6.4375612   6.2336922   6.5298076
20    0.000000    0.000000   0.0000000   0.0000000   5.3029461
```

We've shown only the first 5 cells here. The column names are the pseudo-times that were generated for each sample using Monocle (Trapnell *et al.*, 2014). As you can see, the samples have been ordered from lowest to greatest pseudo-time. We've also applied a log(x+1) transformation to the count data.

3 Maximum Absolute Correlation (MAC) Counter Function

Once your data is in the above format, you can use the `MAC_counter()` function to calculate the Max Absolute Correlation (MAC) matrix for you data. The output of this function is a matrix where `Row gene index` and `Column gene index` correspond to the indeces for the gene pair (i,j), `Correlation` is the maximum absolute correlation (MAC) achieved for the pair, and `Lag` is the lag at which the MAC occurred. Note that the pair (i,j) and (j,i) will both appear in the results, as they will potentially have different MACs. As can be seen below, setting `MAC_cutoff=0.2` restricts the output to only those pairs with an MAC of 0.2 or greater.

```
> MAC_results = MAC_counter(data=example_data, max_lag_prop=1/3, MAC_cutoff=0.2, file_name="example", lag_m
> MAC_results[41:71,]
```

```
      Correlation Lag Row gene index Column gene index
 [1,]   0.5735428   0             10                 6
 [2,]   0.5735428   0              6                10
 [3,]   0.5734105   0             20                16
 [4,]   0.5734105   0             16                20
 [5,]   0.5656991   0             11                 6
 [6,]   0.5656991   0              6                11
 [7,]   0.5485706   0             10                 7
 [8,]   0.5485706   0               7                10
 [9,]   0.5148319   0              13                 8
[10,]   0.5148319   0               8                13
[11,]   0.5124447   0               8                 1
[12,]   0.5124447   0               1                 8
[13,]   0.5097848   0              12                10
[14,]   0.5097848   0              10                12
[15,]   0.5060178   0               6                 3
[16,]   0.5060178   0               3                 6
[17,]   0.4722967   0              17                 1
[18,]   0.4722967   0               1                17
[19,]   0.4542344   0              11                10
[20,]   0.4542344   0              10                11
[21,]   0.4182790   0               3                 1
[22,]   0.4182790   0               1                 3
[23,]   0.4144345   0              17                13
[24,]   0.4144345   0              13                17
[25,]   0.4000968   0               8                 3
[26,]   0.4000968   0               3                 8
[27,]   0.3968778   0              16                 7
[28,]   0.3968778   0               7                16
[29,]   0.3959174   0              16                12
[30,]   0.3959174   0              12                16
[31,]   0.3891460   0              16                 4
```

Here, `max_lag_ prop` is the largest proportion of your experiments that you want your lag to be. For this example, we have 564 experiments, so the largest lag we will try is 188. We recommend using at most a `max_lag_prop=1/3`. The variable `file_name` is the name you'd to associate with your files. Our example creates the file `MAC_example.csv`.

```
> MAC_example[1:5,1:5]
```

```
          [,1]       [,2]       [,3]      [,4]       [,5]
[1,]        NA -0.1728618  0.4182790 0.2526990 -0.1765579
[2,] 0.1414134         NA -0.1427626 0.1463172  0.2420226
[3,] 0.4182790  0.1583516         NA 0.2224057  0.1622316
[4,] 0.2526990 -0.1593688  0.2224057        NA -0.1583200
[5,] 0.1635862  0.1673278  0.1622316 0.1699593         NA
```

The variable `lag_matrix` decides whether you would like the associated matrix of lag values to be saved as well. For our example, setting `lag_matrix=T` creates the file `lag_example.csv`.

```
> lag_example[1:5,1:5]
```

```
   V1  V2  V3  V4  V5
1  NA 172   0   0  90
2 121  NA 141 165  93
3   0  67  NA   0   0
4   0  47   0  NA 178
5   0  10   0 184  NA
```

Again, note that the diagonal is set to `NA`. It is important to note that each of the values in the lag matrix correspond to the size of the lag used on the gene listed in the column. In our example, 172 corresponds to starting gene 1's expression at its first pseudo-time point and staggering the expression of gene 2 by 172 pseudo-time points (hence starting at 173).

4 Permutation Analysis Function

To determine a cutoff for significant MAC values, you can use the `MAC_perm()` function.

```
> MAC_perm(data=example_data, MACs_observ=MAC_example, num_perms=10, max_lag_prop=1/3,
+          FDR_cutoffs=101, perm_file_name="example")
```

The variable `num_perms` determines the number of permutations to use. Note we've only used 10 here to simplify our example. For larger datasets, using 100 is most likely appropriate. `FDR_cutoffs` determines the number of cutoffs you'd like to use to split the domain [0,1] for the correlation. `data`, `max_lag_prop` and `perm_file_name` follow the same use as described for `MAC_counter()`.

This returns the dataset below, where `cors` are the correlation cutoffs, `MACs_observed` are the number of observed correlations at that cutoff, `MACs_ave_perm` are the average number observed in the permuted datasets at that cutoff, and `fdr` is the false discovery rate (FDR) observed at that cutoff. We can see that for our example dataset, if we would like to control the FDR around 0.1, then a correlation cutoff of 0.18 would be appropriate. Below are shown the results with nonzero FDR:

```
> perm_example[74:101,]

    cors MACs_observed MACs_ave_perm          fdr
74  0.27            98           0.0 0.0000000000
75  0.26            99           0.0 0.0000000000
76  0.25           101           0.1 0.0009900990
77  0.24           104           0.1 0.0009615385
78  0.23           106           0.3 0.0028301887
79  0.22           110           0.7 0.0063636364
80  0.21           113           2.4 0.0212389381
81  0.20           120           4.7 0.0391666667
82  0.19           137           8.6 0.0627737226
83  0.18           152          17.7 0.1164473684
84  0.17           168          35.8 0.2130952381
85  0.16           188          59.8 0.3180851064
86  0.15           217          95.4 0.4396313364
87  0.14           235         135.8 0.5778723404
88  0.13           244         170.9 0.7004098361
89  0.12           248         186.7 0.7528225806
90  0.11           248         191.1 0.7705645161
91  0.10           248         191.7 0.7729838710
92  0.09           248         191.7 0.7729838710
93  0.08           248         191.7 0.7729838710
94  0.07           248         191.7 0.7729838710
95  0.06           248         191.7 0.7729838710
96  0.05           248         191.7 0.7729838710
97  0.04           248         191.7 0.7729838710
98  0.03           248         191.7 0.7729838710
99  0.02           248         191.7 0.7729838710
100 0.01           248         191.7 0.7729838710
101 0.00           248         191.7 0.7729838710
```

5 Plotting lags versus correlation

It may be of interest view the distribution of lags at various correlation cutoffs. This is easy to do with the output from the `MAC_counter()`. First we must convert the numerical values lag matrix `lag_example` into categorical variables based on whatever cutoffs we'd like to designate. First, lets pull out all of the non-`NA` values from both of our matrices:

```
> cors=c()
> lag = c()
> for (i in(1:20)){
+
+   cors = c(cors, na.omit(MAC_example[,i]))
+   lag = c(lag, na.omit(lag_example[,i]))
+ }
```

We then convert the lag values into categorical cutoffs. Keeping with our paper:

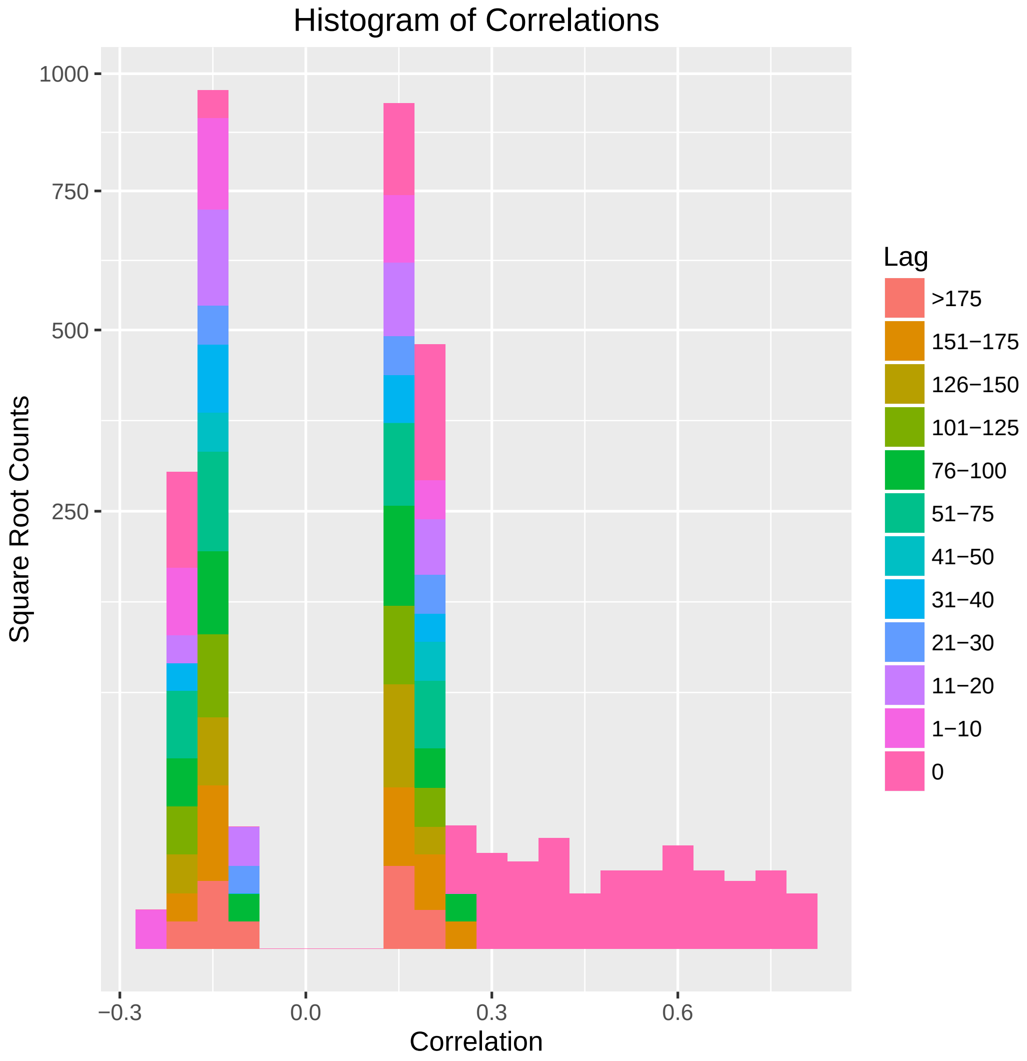
```
> lag_bin=c()
> for(i in (1:380)){
+   if(lag[i]==0){lag_bin[i]="0"
+   }else if(lag[i]>0 & lag[i]<=10){lag_bin[i]="1-10"
+   }else if(lag[i]>10 & lag[i]<=20){lag_bin[i]="11-20"
+   }else if(lag[i]>20 & lag[i]<=30){lag_bin[i]="21-30"
+   }else if(lag[i]>30 & lag[i]<=40){lag_bin[i]="31-40"
+   }else if(lag[i]>40 & lag[i]<=50){lag_bin[i]="41-50"
+   }else if(lag[i]>50 & lag[i]<=75){lag_bin[i]="51-75"
+   }else if(lag[i]>75 & lag[i]<=100){lag_bin[i]="76-100"
+   }else if(lag[i]>100 & lag[i]<=125){lag_bin[i]="101-125"
+   }else if(lag[i]>125 & lag[i]<=150){lag_bin[i]="126-150"
+   }else if(lag[i]>151 & lag[i]<=175){lag_bin[i]="151-175"
+   }else{lag_bin[i]=">175"}
+ }
```

We then need to combine our correlation values and lag bins to create a data frame that `ggplot2` can use:

```
> data = as.data.frame(cbind(cors,lag,lag_bin))
> data$cors = as.numeric(as.character(data$cors))
> data$lag_bin = factor(lag_bin, levels =c(">175", "151-175", "126-150", "101-125", "76-100", "51-75", "41-
+                                         "21-30", "11-20", "1-10", "0"))
```

Finally, we can plot a histrogram of our results:

```
> library(ggplot2)
> ggplot(data = data, aes(data$cors)) + geom_histogram(binwidth = 0.05, aes(fill = lag_bin)) + scale_y_sqrt
```



6 Generating matrices for use in WGCNA

If you intend to use the results from LEAP for further analysis with WGCNA (Langfelder and Horvath, 2008), then you will require a symmetric matrix of correlations. LEAP will compute this matrix by setting `symmetric=T`. This creates two files, `lag_symmetric_example.csv` and `MAC_symmetric_example.csv`. LEAP finds this matrix by comparing the correlation of each (i,j) and (j,i) pair, and keeping the value with the maximum absolute correlation. In our example, the pair gene 1 and gene 2 have correlation -0.17 when gene 2 is the lagged gene, (1,2), and correlation 0.14 when gene 1 is the lagged gene, (2,1), then in the symmetric matrix (1,2)=(2,1) = -0.17. Below are the results when we find the symmetric matrix for our example dataset:

```
> output=MAC_counter(data=example_data, max_lag_prop=1/3, file_name="example", lag_matrix=T, symmetric=T)
```

```
> MAC_symmetric[1:5,1:5]
```

```
          V1         V2        V3         V4         V5
1         NA -0.1728618 0.4182790  0.2526990 -0.1765579
2 -0.1728618         NA 0.1583516 -0.1593688  0.2420226
3  0.4182790  0.1583516        NA  0.2224057  0.1622316
4  0.2526990 -0.1593688 0.2224057         NA  0.1699593
5 -0.1765579  0.2420226 0.1622316  0.1699593         NA
```

7 Session information

```
> sessionInfo()

R version 3.2.2 (2015-08-14)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 7 x64 (build 7601) Service Pack 1

locale:
[1] LC_COLLATE=English_United States.1252
[2] LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] ggplot2_2.1.0 LEAP_0.1

loaded via a namespace (and not attached):
 [1] labeling_0.3     colorspace_1.2-6 scales_0.4.0     plyr_1.8.3
 [5] tools_3.2.2      gtable_0.2.0     Rcpp_0.12.3      grid_3.2.2
 [9] digest_0.6.9     munsell_0.4.3
```

8 Citation information

References

[Langfelder and Horvath, 2008] Langfelder, Peter and Horvath, Steve (2008) WGCNA: an R package for weighted correlation network analysis. *BMC bioinformatics*, **9(1)**, 1.

[Shalek *et al.*, 2014] Shalek AK, Satija R., Shuga J., Trombetta J.J. et al. (2014) Single-cell RNA-seq reveals dynamic paracrine control of cellular variation. *Nature*, **510(7505)**, 363-369.

[Trapnell *et al.*, 2014] Trapnell, Cole and Cacchiarelli, Davide and Grimsby, Jonna and Pokharel, Prapti and Li, Shuqiang and Morse, Michael and Lennon, Niall J. and Livak, Kenneth J. and Mikkelsen, Tarjei S. and Rinn, John L. (2014) The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells, *Nature Biotechnology*, **32(4)**, 381-386.