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 ................................. 1
2 Data format ................................. 2
3 Maximum Absolute Correlation (MAC) Counter Function ................................. 2
4 Permutation Analysis Function ........... 3
5 Plotting lags versus correlation ......... 5
6 Generating matrices for use in WGCNA .................................................................. 6
7 Session information ......................... 7
8 Citation information ..................... 7

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 \textit{Mus musculus} dendritic cells (Shalek et al., 2014):

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

2.546586  5.102184  0.000000  2.983997  0.000000  4.526036  6.532284  6.090495  5.627550  6.183883
3.592766  6.214955  0.000000  3.200364  6.447713  7.252316  6.627344  0.000000  1.603566  0.000000
0.000000  5.209410  5.541859  5.792501  2.819949  0.000000  0.000000  0.000000  5.826266  3.451973
3.253274  2.959225  3.927282  6.394018  6.268614  0.000000  0.000000  0.000000  0.000000  0.000000
1.035870  7.706152  0.000000  0.908293  6.622659  5.773570  4.750586  0.000000  0.000000  0.952919
3.884139  4.667165  0.000000  5.600122  4.540893  6.233692  6.529807  4.540893  0.000000  0.000000
6.288354  6.437330  6.437561  6.233692  6.529807  0.000000  0.000000  0.000000  0.000000  5.302946
```

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 \texttt{MAC_counter()} function to calculate the Max Absolute Correlation (MAC) matrix for your data. The output of this function is a matrix where \texttt{Row gene index} and \texttt{Column gene index} correspond to the indeces for the gene pair (i,j), \texttt{Correlation} is the maximum absolute correlation (MAC) achieved for the pair, and \texttt{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 \texttt{MAC_cutoff=0.2} restricts the output to only those pairs with an MAC of 0.2 or greater.

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

<table>
<thead>
<tr>
<th>Correlation</th>
<th>Lag</th>
<th>Row gene index</th>
<th>Column gene index</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>0.5735428</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>[2,]</td>
<td>0.5735428</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>[3,]</td>
<td>0.5734105</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>[4,]</td>
<td>0.5734105</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>[5,]</td>
<td>0.5656991</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>[6,]</td>
<td>0.5656991</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>[7,]</td>
<td>0.5485706</td>
<td>0</td>
<td>10</td>
</tr>
</tbody>
</table>
```
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,] 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.
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,]

<table>
<thead>
<tr>
<th>cors</th>
<th>MACs_observed</th>
<th>MACs_ave_perm</th>
<th>fdr</th>
</tr>
</thead>
<tbody>
<tr>
<td>74</td>
<td>0.27</td>
<td>98</td>
<td>0.0 0.0000000000</td>
</tr>
<tr>
<td>75</td>
<td>0.26</td>
<td>99</td>
<td>0.0 0.0000000000</td>
</tr>
<tr>
<td>76</td>
<td>0.25</td>
<td>101</td>
<td>0.1 0.0009900990</td>
</tr>
<tr>
<td>77</td>
<td>0.24</td>
<td>104</td>
<td>0.1 0.0009615385</td>
</tr>
<tr>
<td>78</td>
<td>0.23</td>
<td>106</td>
<td>0.3 0.0028301887</td>
</tr>
<tr>
<td>79</td>
<td>0.22</td>
<td>110</td>
<td>0.7 0.0063636364</td>
</tr>
<tr>
<td>80</td>
<td>0.21</td>
<td>113</td>
<td>2.4 0.0212389381</td>
</tr>
<tr>
<td>81</td>
<td>0.20</td>
<td>120</td>
<td>4.7 0.0391666667</td>
</tr>
<tr>
<td>82</td>
<td>0.19</td>
<td>137</td>
<td>8.6 0.0627737226</td>
</tr>
<tr>
<td>83</td>
<td>0.18</td>
<td>152</td>
<td>17.7 0.1164473684</td>
</tr>
<tr>
<td>84</td>
<td>0.17</td>
<td>168</td>
<td>35.8 0.2130952381</td>
</tr>
<tr>
<td>85</td>
<td>0.16</td>
<td>188</td>
<td>59.8 0.3180851064</td>
</tr>
<tr>
<td>86</td>
<td>0.15</td>
<td>217</td>
<td>95.4 0.4396313364</td>
</tr>
<tr>
<td>87</td>
<td>0.14</td>
<td>235</td>
<td>135.8 0.5778723404</td>
</tr>
<tr>
<td>88</td>
<td>0.13</td>
<td>244</td>
<td>170.9 0.7004098361</td>
</tr>
<tr>
<td>89</td>
<td>0.12</td>
<td>248</td>
<td>186.7 0.7528225806</td>
</tr>
<tr>
<td>90</td>
<td>0.11</td>
<td>248</td>
<td>191.1 0.7705645161</td>
</tr>
<tr>
<td>91</td>
<td>0.10</td>
<td>248</td>
<td>191.7 0.7729838710</td>
</tr>
<tr>
<td>92</td>
<td>0.09</td>
<td>248</td>
<td>191.7 0.7729838710</td>
</tr>
<tr>
<td>93</td>
<td>0.08</td>
<td>248</td>
<td>191.7 0.7729838710</td>
</tr>
<tr>
<td>94</td>
<td>0.07</td>
<td>248</td>
<td>191.7 0.7729838710</td>
</tr>
<tr>
<td>95</td>
<td>0.06</td>
<td>248</td>
<td>191.7 0.7729838710</td>
</tr>
<tr>
<td>96</td>
<td>0.05</td>
<td>248</td>
<td>191.7 0.7729838710</td>
</tr>
<tr>
<td>97</td>
<td>0.04</td>
<td>248</td>
<td>191.7 0.7729838710</td>
</tr>
<tr>
<td>98</td>
<td>0.03</td>
<td>248</td>
<td>191.7 0.7729838710</td>
</tr>
<tr>
<td>99</td>
<td>0.02</td>
<td>248</td>
<td>191.7 0.7729838710</td>
</tr>
<tr>
<td>100</td>
<td>0.01</td>
<td>248</td>
<td>191.7 0.7729838710</td>
</tr>
<tr>
<td>101</td>
<td>0.00</td>
<td>248</td>
<td>191.7 0.7729838710</td>
</tr>
</tbody>
</table>
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, let's pull out all of the non-NA values from both of our matrices:

```r
> 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:

```r
> 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] > 150 & 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:

```r
> 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 histogram of our results:

```r
> 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:

```r
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:

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

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

<table>
<thead>
<tr>
<th></th>
<th>V1</th>
<th>V2</th>
<th>V3</th>
<th>V4</th>
<th>V5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NA</td>
<td>-0.1728618</td>
<td>0.4182790</td>
<td>0.2526990</td>
<td>-0.1765579</td>
</tr>
<tr>
<td>2</td>
<td>-0.1728618</td>
<td>NA</td>
<td>0.1583516</td>
<td>-0.1593688</td>
<td>0.2420226</td>
</tr>
<tr>
<td>3</td>
<td>0.4182790</td>
<td>0.1583516</td>
<td>NA</td>
<td>0.2224057</td>
<td>0.1622316</td>
</tr>
<tr>
<td>4</td>
<td>0.2526990</td>
<td>-0.1593688</td>
<td>0.2224057</td>
<td>NA</td>
<td>0.1699593</td>
</tr>
<tr>
<td>5</td>
<td>-0.1765579</td>
<td>0.2420226</td>
<td>0.1622316</td>
<td>0.1699593</td>
<td>NA</td>
</tr>
</tbody>
</table>
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

