Using the *SIMLR* package

Bo Wang[*] Daniele Ramazzotti[†] Luca De Sano[‡] Junjie Zhu[§] Emma Pierson[¶]
Serafim Batzoglou[‖]

September 29, 2017

Overview. Single-cell RNA-seq technologies enable high throughput gene expression measurement of individual cells, and allow the discovery of heterogeneity within cell populations. Measurement of cell-to-cell gene expression similarity is critical to identification, visualization and analysis of cell populations. However, single-cell data introduce challenges to conventional measures of gene expression similarity because of the high level of noise, outliers and dropouts. We develop a novel similarity-learning framework, *SIMLR* (Single-cell Interpretation via Multi-kernel LeaRning), which learns an appropriate distance metric from the data for dimension reduction, clustering and visualization. *SIMLR* is capable of separating known subpopulations more accurately in single-cell data sets than do existing dimension reduction methods. Additionally, *SIMLR* demonstrates high sensitivity and accuracy on high-throughput peripheral blood mononuclear cells (PBMC) data sets generated by the GemCode single-cell technology from 10x Genomics.

In this vignette, we give an overview of the package by presenting some of its main functions.

[*]Department of Computer Science, Stanford University, Stanford, CA , USA.
[†]Department of Pathology, Stanford University, Stanford, CA , USA.
[‡]Dipartimento di Informatica Sistemistica e Comunicazione, Università degli Studi Milano Bicocca Milano, Italy.
[§]Department of Electrical Engineering, Stanford University, Stanford, CA , USA.
[¶]Department of Computer Science, Stanford University, Stanford, CA , USA.
[‖]Department of Computer Science, Stanford University, Stanford, CA , USA.

Contents

1 Changelog 2

2 Algorithms and useful links 2

3 Using SIMLR 2

4 sessionInfo() 10

1 Changelog

1.0.0 implements SIMLR and SIMLR feature ranking algorithms.
1.0.2 implements SIMLR large scale algorithms.

2 Algorithms and useful links

Acronym	Extended name	Reference
SIMLR	Single-cell Interpretation via Multi-kernel LeaRning	Paper

3 Using SIMLR

We first load the data provided as an example in the package. The dataset BuettnerFlorian is used for an example of the standard SIMLR, while the dataset ZeiselAmit is used for an example of SIMLR large scale.

```
library(SIMLR)
data(BuettnerFlorian)
data(ZeiselAmit)
```

The external R package igraph is required for the computation of the normalized mutual information to assess the results of the clustering.

```
library(igraph)
```

We now run SIMLR as an example on an input dataset from Buettner, Florian, et al. "Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells." Nature biotechnology 33.2 (2015): 155-160. For this dataset we have a ground true of 3 cell populations, i.e., clusters.

```
set.seed(11111)
example = SIMLR(X = BuettnerFlorian$in_X, c = BuettnerFlorian$n_clust, cores.ratio = 0)
```

```
## Computing the multiple Kernels.
## Performing network diffiusion.
## Iteration:  1
## Iteration:  2
## Iteration:  3
## Iteration:  4
## Iteration:  5
## Iteration:  6
## Iteration:  7
## Iteration:  8
## Iteration:  9
## Iteration:  10
```

```
## Iteration:  11
## Performing t-SNE.
## Epoch: Iteration # 100  error is:  0.1326426
## Epoch: Iteration # 200  error is:  0.08721104
## Epoch: Iteration # 300  error is:  0.05808032
## Epoch: Iteration # 400  error is:  0.05713627
## Epoch: Iteration # 500  error is:  0.0570977
## Epoch: Iteration # 600  error is:  0.0570602
## Epoch: Iteration # 700  error is:  0.05702496
## Epoch: Iteration # 800  error is:  0.05699161
## Epoch: Iteration # 900  error is:  0.05696026
## Epoch: Iteration # 1000  error is:  0.05693017
## Performing Kmeans.
## Performing t-SNE.
## Epoch: Iteration # 100  error is:  11.21592
## Epoch: Iteration # 200  error is:  0.6659411
## Epoch: Iteration # 300  error is:  0.5755609
## Epoch: Iteration # 400  error is:  0.3986313
## Epoch: Iteration # 500  error is:  0.4913843
## Epoch: Iteration # 600  error is:  0.3970208
## Epoch: Iteration # 700  error is:  0.3979712
## Epoch: Iteration # 800  error is:  0.3530334
## Epoch: Iteration # 900  error is:  0.3026329
## Epoch: Iteration # 1000  error is:  0.2887195
```

We now compute the normalized mutual information between the inferred clusters by SIMLR and the true ones. This measure with values in [0,1], allows us to assess the performance of the clustering with higher values reflecting better performance.

```
nmi_1 = compare(BuettnerFlorian$true_labs[,1], example$y$cluster, method="nmi")
print(nmi_1)
```

```
## [1] 0.888298
```

As a further understanding of the results, we now visualize the cell populations in a plot.

```
plot(example$ydata,
    col = c(topo.colors(BuettnerFlorian$n_clust))[BuettnerFlorian$true_labs[,1]],
    xlab = "SIMLR component 1",
    ylab = "SIMLR component 2",
    pch = 20,
    main="SIMILR 2D visualization for BuettnerFlorian")
```

We also run SIMLR feature ranking on the same inputs to get a rank of the key genes with the related pvalues.

```
set.seed(11111)
ranks = SIMLR_Feature_Ranking(A=BuettnerFlorian$results$S,X=BuettnerFlorian$in_X)
```

```
head(ranks$pval)
```

```
## [1] 2.201015e-125  2.531379e-90  5.632172e-77  6.719501e-76  4.444251e-72  8.822900e-69
```

```
head(ranks$aggR)
```

```
## [1] 5701 1689 7549   57 2653 8081
```

We finally show an example for SIMLR large scale on an input dataset being a reduced version of the dataset provided in Buettner, Zeisel, Amit, et al. "Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq." Science 347.6226 (2015): 1138-1142. For this dataset we have a ground true of 9 cell populations, i.e., clusters.

```
set.seed(11111)
example_large_scale = SIMLR_Large_Scale(X = ZeiselAmit$in_X, c = ZeiselAmit$n_clust, kk = 10)
```

```
## Performing fast PCA.
```

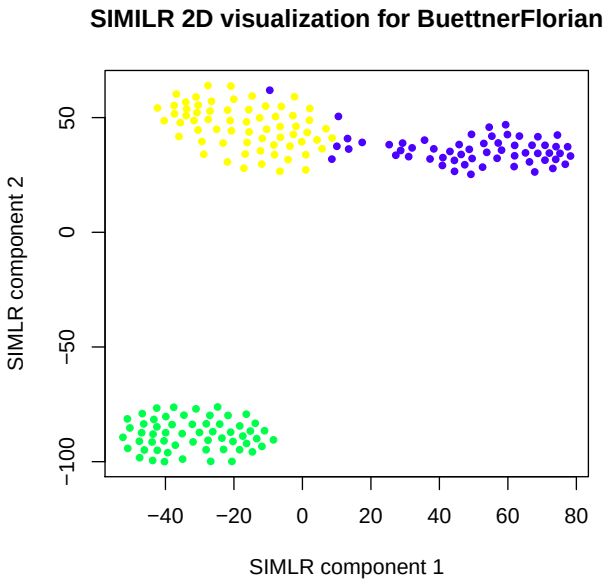

Figure 1: Visualization of the 3 cell populations retrieved by SIMLR on the dataset by Florian, et al.

```
## Performing k-nearest neighbour search.
## Computing the multiple Kernels.
## Performing the iterative procedure  5  times.
## Iteration:  1
## Iteration:  2
## Iteration:  3
## Iteration:  4
## Iteration:  5
## Performing Kmeans.
## Performing t-SNE.
## The main loop will be now performed with a maximum of 300 iterations.
## Performing iteration 1.
## Performing iteration 2.
## Performing iteration 3.
## Performing iteration 4.
## Performing iteration 5.
## Performing iteration 6.
## Performing iteration 7.
## Performing iteration 8.
## Performing iteration 9.
## Performing iteration 10.
## Performing iteration 11.
## Performing iteration 12.
## Performing iteration 13.
## Performing iteration 14.
## Performing iteration 15.
## Performing iteration 16.
## Performing iteration 17.
## Performing iteration 18.
## Performing iteration 19.
## Performing iteration 20.
## Performing iteration 21.
## Performing iteration 22.
## Performing iteration 23.
```

```
## Performing iteration 24.
## Performing iteration 25.
## Performing iteration 26.
## Performing iteration 27.
## Performing iteration 28.
## Performing iteration 29.
## Performing iteration 30.
## Performing iteration 31.
## Performing iteration 32.
## Performing iteration 33.
## Performing iteration 34.
## Performing iteration 35.
## Performing iteration 36.
## Performing iteration 37.
## Performing iteration 38.
## Performing iteration 39.
## Performing iteration 40.
## Performing iteration 41.
## Performing iteration 42.
## Performing iteration 43.
## Performing iteration 44.
## Performing iteration 45.
## Performing iteration 46.
## Performing iteration 47.
## Performing iteration 48.
## Performing iteration 49.
## Performing iteration 50.
## Performing iteration 51.
## Performing iteration 52.
## Performing iteration 53.
## Performing iteration 54.
## Performing iteration 55.
## Performing iteration 56.
## Performing iteration 57.
## Performing iteration 58.
## Performing iteration 59.
## Performing iteration 60.
## Performing iteration 61.
## Performing iteration 62.
## Performing iteration 63.
## Performing iteration 64.
## Performing iteration 65.
## Performing iteration 66.
## Performing iteration 67.
## Performing iteration 68.
## Performing iteration 69.
## Performing iteration 70.
## Performing iteration 71.
## Performing iteration 72.
## Performing iteration 73.
## Performing iteration 74.
## Performing iteration 75.
## Performing iteration 76.
## Performing iteration 77.
## Performing iteration 78.
## Performing iteration 79.
## Performing iteration 80.
## Performing iteration 81.
```

```
## Performing iteration 82.
## Performing iteration 83.
## Performing iteration 84.
## Performing iteration 85.
## Performing iteration 86.
## Performing iteration 87.
## Performing iteration 88.
## Performing iteration 89.
## Performing iteration 90.
## Performing iteration 91.
## Performing iteration 92.
## Performing iteration 93.
## Performing iteration 94.
## Performing iteration 95.
## Performing iteration 96.
## Performing iteration 97.
## Performing iteration 98.
## Performing iteration 99.
## Performing iteration 100.
## Performing iteration 101.
## Performing iteration 102.
## Performing iteration 103.
## Performing iteration 104.
## Performing iteration 105.
## Performing iteration 106.
## Performing iteration 107.
## Performing iteration 108.
## Performing iteration 109.
## Performing iteration 110.
## Performing iteration 111.
## Performing iteration 112.
## Performing iteration 113.
## Performing iteration 114.
## Performing iteration 115.
## Performing iteration 116.
## Performing iteration 117.
## Performing iteration 118.
## Performing iteration 119.
## Performing iteration 120.
## Performing iteration 121.
## Performing iteration 122.
## Performing iteration 123.
## Performing iteration 124.
## Performing iteration 125.
## Performing iteration 126.
## Performing iteration 127.
## Performing iteration 128.
## Performing iteration 129.
## Performing iteration 130.
## Performing iteration 131.
## Performing iteration 132.
## Performing iteration 133.
## Performing iteration 134.
## Performing iteration 135.
## Performing iteration 136.
## Performing iteration 137.
## Performing iteration 138.
## Performing iteration 139.
```

```
## Performing iteration 140.
## Performing iteration 141.
## Performing iteration 142.
## Performing iteration 143.
## Performing iteration 144.
## Performing iteration 145.
## Performing iteration 146.
## Performing iteration 147.
## Performing iteration 148.
## Performing iteration 149.
## Performing iteration 150.
## Performing iteration 151.
## Performing iteration 152.
## Performing iteration 153.
## Performing iteration 154.
## Performing iteration 155.
## Performing iteration 156.
## Performing iteration 157.
## Performing iteration 158.
## Performing iteration 159.
## Performing iteration 160.
## Performing iteration 161.
## Performing iteration 162.
## Performing iteration 163.
## Performing iteration 164.
## Performing iteration 165.
## Performing iteration 166.
## Performing iteration 167.
## Performing iteration 168.
## Performing iteration 169.
## Performing iteration 170.
## Performing iteration 171.
## Performing iteration 172.
## Performing iteration 173.
## Performing iteration 174.
## Performing iteration 175.
## Performing iteration 176.
## Performing iteration 177.
## Performing iteration 178.
## Performing iteration 179.
## Performing iteration 180.
## Performing iteration 181.
## Performing iteration 182.
## Performing iteration 183.
## Performing iteration 184.
## Performing iteration 185.
## Performing iteration 186.
## Performing iteration 187.
## Performing iteration 188.
## Performing iteration 189.
## Performing iteration 190.
## Performing iteration 191.
## Performing iteration 192.
## Performing iteration 193.
## Performing iteration 194.
## Performing iteration 195.
## Performing iteration 196.
## Performing iteration 197.
```

```
## Performing iteration 198.
## Performing iteration 199.
## Performing iteration 200.
## Performing iteration 201.
## Performing iteration 202.
## Performing iteration 203.
## Performing iteration 204.
## Performing iteration 205.
## Performing iteration 206.
## Performing iteration 207.
## Performing iteration 208.
## Performing iteration 209.
## Performing iteration 210.
## Performing iteration 211.
## Performing iteration 212.
## Performing iteration 213.
## Performing iteration 214.
## Performing iteration 215.
## Performing iteration 216.
## Performing iteration 217.
## Performing iteration 218.
## Performing iteration 219.
## Performing iteration 220.
## Performing iteration 221.
## Performing iteration 222.
## Performing iteration 223.
## Performing iteration 224.
## Performing iteration 225.
## Performing iteration 226.
## Performing iteration 227.
## Performing iteration 228.
## Performing iteration 229.
## Performing iteration 230.
## Performing iteration 231.
## Performing iteration 232.
## Performing iteration 233.
## Performing iteration 234.
## Performing iteration 235.
## Performing iteration 236.
## Performing iteration 237.
## Performing iteration 238.
## Performing iteration 239.
## Performing iteration 240.
## Performing iteration 241.
## Performing iteration 242.
## Performing iteration 243.
## Performing iteration 244.
## Performing iteration 245.
## Performing iteration 246.
## Performing iteration 247.
## Performing iteration 248.
## Performing iteration 249.
## Performing iteration 250.
## Performing iteration 251.
## Performing iteration 252.
## Performing iteration 253.
## Performing iteration 254.
## Performing iteration 255.
```

```
## Performing iteration 256.
## Performing iteration 257.
## Performing iteration 258.
## Performing iteration 259.
## Performing iteration 260.
## Performing iteration 261.
## Performing iteration 262.
## Performing iteration 263.
## Performing iteration 264.
## Performing iteration 265.
## Performing iteration 266.
## Performing iteration 267.
## Performing iteration 268.
## Performing iteration 269.
## Performing iteration 270.
## Performing iteration 271.
## Performing iteration 272.
## Performing iteration 273.
## Performing iteration 274.
## Performing iteration 275.
## Performing iteration 276.
## Performing iteration 277.
## Performing iteration 278.
## Performing iteration 279.
## Performing iteration 280.
## Performing iteration 281.
## Performing iteration 282.
## Performing iteration 283.
## Performing iteration 284.
## Performing iteration 285.
## Performing iteration 286.
## Performing iteration 287.
## Performing iteration 288.
## Performing iteration 289.
## Performing iteration 290.
## Performing iteration 291.
## Performing iteration 292.
## Performing iteration 293.
## Performing iteration 294.
## Performing iteration 295.
## Performing iteration 296.
## Performing iteration 297.
## Performing iteration 298.
## Performing iteration 299.
## Performing iteration 300.
```

We compute the normalized mutual information between the inferred clusters by SIMLR large scale and the true ones.

```
nmi_2 = compare(ZeiselAmit$true_labs[,1], example_large_scale$y$cluster, method="nmi")
print(nmi_2)
```

```
## [1] 0.04158302
```

As a further understanding of the results, also in this case we visualize the cell populations in a plot.

```
plot(example_large_scale$ydata,
     col = c(topo.colors(ZeiselAmit$n_clust))[ZeiselAmit$true_labs[,1]],
     xlab = "SIMLR component 1",
     ylab = "SIMLR component 2",
```

```
    pch = 20,
    main="SIMILR 2D visualization for ZeiselAmit")
```

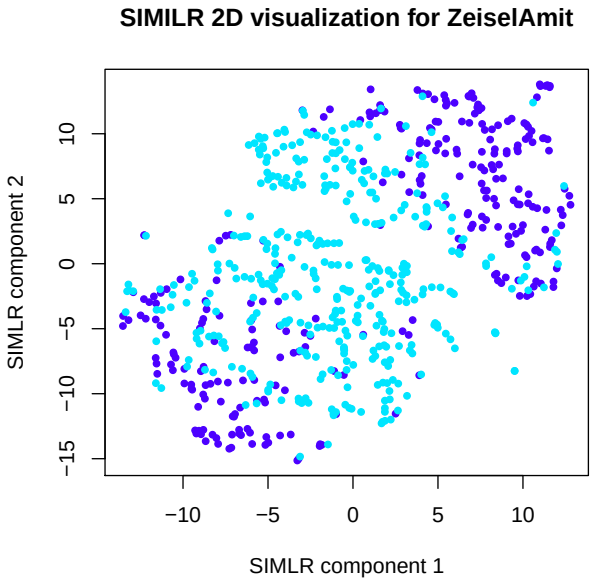

Figure 2: Visualization of the 9 cell populations retrieved by SIMLR large scale on the dataset by Zeisel, Amit, et al.

4 sessionInfo()

- R version 3.4.1 (2017-06-30), x86_64-pc-linux-gnu
- Locale: LC_CTYPE=en_US.UTF-8, LC_NUMERIC=C, LC_TIME=en_US.UTF-8, LC_COLLATE=C, LC_MONETARY=en_US.UTF-8, LC_MESSAGES=en_US.UTF-8, LC_PAPER=en_US.UTF-8, LC_NAME=C, LC_ADDRESS=C, LC_TELEPHONE=C, LC_MEASUREMENT=en_US.UTF-8, LC_IDENTIFICATION=C
- Running under: Ubuntu 16.04.3 LTS
- Matrix products: default
- BLAS: /home/biocbuild/bbs-3.5-bioc/R/lib/libRblas.so
- LAPACK: /home/biocbuild/bbs-3.5-bioc/R/lib/libRlapack.so
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: SIMLR 1.2.3, igraph 1.1.2, knitr 1.17
- Loaded via a namespace (and not attached): BiocStyle 2.4.1, Matrix 1.2-11, RSpectra 0.12-0, Rcpp 0.12.13, RcppAnnoy 0.0.10, backports 1.1.1, codetools 0.2-15, compiler 3.4.1, digest 0.6.12, evaluate 0.10.1, grid 3.4.1, highr 0.6, htmltools 0.3.6, lattice 0.20-35, magrittr 1.5, parallel 3.4.1, pkgconfig 2.0.1, pracma 2.0.7, quadprog 1.5-5, rmarkdown 1.6, rprojroot 1.2, stringi 1.1.5, stringr 1.2.0, tools 3.4.1, yaml 2.1.14