SUITOR: selecting the number of mutational signatures through cross-validation

February 8, 2022

Contents

Introduction

Mutational signatures are patterns of somatic mutations imprinted on the cancer genome by operative mutational processes, and have been proposed to identify cancer predisposition genes and to stratify cancer patients for precision treatment. For the *de novo* mutational signature analysis, estimating the correct number of signatures is the crucial starting point, since it influences all the downstream steps, including extraction of signature profiles, estimation of signature activities and classification of tumors based on the estimated activities. Despite the many algorithms proposed to extract signature profiles and to estimate signature contributions, relatively little emphasis has been placed on selecting the correct number of de novo mutational signatures in cancer genomics studies. The SUITOR package uses unsupervised cross-validation to select the optimal number of signatures.

1 Installation

Installing the SUITOR package from Bioconductor.

1.1 Install from Bioconductor

```
>   #if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")
>   #BiocManager::install("SUITOR")
```

2 Loading the package

Before using the SUITOR package, it must be loaded into an R session.

```
> library(SUITOR)
```

3 Example data

For illustrative purposes, we simulated a 96×300 mutational catalog matrix which contains 300 tumors with respect to 96 single base substitution categories. Each element of the matrix is generated from the Poisson distribution ($rpois()$ function) with the mean corresponding to each element of WH, where W is the true signature matrix of size 96×8 for 8 signatures, and H is the activity matrix of size 8×300. Specifically, we used the profile of eight COSMIC signatures 4, 6, 7a, 9, 17b, 22, 26, 39 (`https://cancer.sanger.ac.uk/cosmic/signatures/SBS`) for W. H is generated from a uniform distribution between 0 and 100 with some randomly chosen elements of H set to 0 in order to mimic real data.

```
>   data(SimData, package="SUITOR")
>   dim(SimData)
```

```
[1]  96 300
```

```
>   SimData[1:6, 1:6]
```

```
        X1 X2 X3 X4 X5 X6
A[C>A]A  0  1  1  6  0  4
A[C>A]C  0  0  0  1  2  4
A[C>A]G  0  0  0  2  1  2
A[C>A]T  1  2  0  3  2  3
C[C>A]A  0  0  0  8  1  7
C[C>A]C  2  0  0 17  0 13
```

4 Selecting the number of mutational signatures

The main function $suitor(data, op)$ is to select the number of mutational signatures based on cross-validation. It has two arguments described below.

4.1 Input data

The first argument of the function $suitor()$ is *data*. It could be an R data frame or matrix containing a mutational catalog whose elements are non-negative counts. Each column of *data* corresponds to a tumor (or sample) while its rows represent a mutation type. Although selection of the number of signatures is independent to the order of mutation type, we specify the order of mutation type according to the COSMIC database (`https://cancer.sanger.ac.uk/signatures/sbs/`) for extracting signature profiles using $suitor\_extract\_WH()$ after estimating the optimal rank.

4.2 Options

Since SUITOR is based on cross-validation and the Expectation Conditional Maximization (ECM) algorithm, it is necessary to set a list of tuning parameters which control the fitting process. These parameters are defined in the table below.

Name	Description	Default Value
min.value	Minimum value of matrix before factorizing	1e-4
min.rank	Minimum rank	1
max.rank	Maximum rank	10
$k.fold$	Number of folds	10
em.eps	EM algorithm stopping tolerance	1e-5
max.iter	Maximum number of iterations in EM algorithm	2000
n.starts	Number of starting points	30
n.cores	Number of cores to use	1
get.summary	0 or 1 to create summary results	1
plot	0 or 1 to produce an error plot	1
print	0 or 1 to print info (0=no printing)	1

The option *min.value* is a small number added to the data matrix for stable computation of non-negative matrix factorization. For a given number of signatures or ranks $r(min.rank \leq r \leq max.rank)$, the data matrix is divided into $k.fold$ parts for the cross-validation. The default value of the maximal rank *max.rank* is 10 but it can be changed depending on the cancer type. The default value of the number of folds K ($k.fold$) is 10 and it can be modified depending on the computer resources. Since the ECM algorithm may converge

to a local saddle point, SUITOR tries multiple initial values for W and H. For this purpose, the number of starts (*n.starts*) is used. Although the default *n.starts* is set to 30, it can be increased depending on the size of the data matrix and/or computational resources. For the ECM algorithm, the default value of the maximal iteration *max.iter* is set to 2000. It is possible for some cases to reach the maximal iteration, for which the function would produce a warning message. Overall, we recommend a two-stage approach where the user would run *suitor*() with the default option first and then narrow down the set of plausible ranks ($min.rank \leq r \leq max.rank$) with more starts (*n.starts*) and a larger number of maximal iteration (*max.iter*) if necessary. To ease the computation burden, SUITOR supports parallel computing by specifying *n.cores* greater than 1. To check whether parallel computing is available for the computer, the *detectCores*() function can be used. If *detectCores*() returns a value greater than 1, then that return value may be used for *n.cores*.

5 Running suitor() with the default options

First we start with the default options. Note that this will take a while to run, so we will load the previously saved results.

```
> #re <- suitor(SimData)
> load(system.file("extdata", "re.rda", package="SUITOR"))
> str(re)

List of 4
 $ rank       : num 8
 $ summary    :'data.frame':        20 obs. of  13 variables:
  ..$ Rank  : num [1:20] 1 1 2 2 3 3 4 4 5 5 ...
  ..$ Type  : chr [1:20] "Train" "Test" "Train" "Test" ...
  ..$ MSErr : num [1:20] 1.066 1.086 0.995 1.052 0.923 ...
  ..$ fold1 : num [1:20] 29511 3335 25651 3218 22041 ...
  ..$ fold2 : num [1:20] 29642 3200 25747 3093 22120 ...
  ..$ fold3 : num [1:20] 29595 3247 25788 3083 22257 ...
  ..$ fold4 : num [1:20] 29395 3452 25638 3241 22075 ...
  ..$ fold5 : num [1:20] 29355 3484 25591 3295 22094 ...
  ..$ fold6 : num [1:20] 29486 3357 25666 3167 22091 ...
  ..$ fold7 : num [1:20] 29416 3427 25674 3081 22100 ...
  ..$ fold8 : num [1:20] 29363 3485 25546 3216 22013 ...
  ..$ fold9 : num [1:20] 29388 3454 25643 3224 22090 ...
  ..$ fold10: num [1:20] 29362 3496 25571 3278 21968 ...
 $ all.results: num [1:3000, 1:6] 1 2 3 4 5 6 7 8 9 10 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:6] "Rank" "k" "Start" "Error.Train" ...
 $ op         :List of 18
  ..$ min.rank   : num 1
  ..$ max.rank   : num 10
  ..$ k.fold     : num 10
  ..$ n.starts   : num 30
  ..$ max.iter   : num 2000
```

```
  ..$ em.eps     : num 1e-05
  ..$ plot       : logi TRUE
  ..$ print      : num 1
  ..$ min.value  : num 1e-04
  ..$ get.summary: num 1
  ..$ n.cores    : num 1
  ..$ kfold.vec  : int [1:10] 1 2 3 4 5 6 7 8 9 10
  ..$ seeds      : int [1:30] 1 2 3 4 5 6 7 8 9 10 ...
  ..$ parMat     : int [1:3000, 1:3] 1 2 3 4 5 6 7 8 9 10 ...
  ..$ parStart   : num 1
  ..$ parEnd     : num 3000
  ..$ algorithm  : num 1
  ..$ type       : chr "FORK"
```

By default, the *suitor*() function returns a list containing the estimated optimal rank ($re\$rank$), a summary matrix ($re\$summary$) where cross validation errors are tabulated, as well as the detailed results ($re\$all.results$) which contain the training and testing errors, the total number of ECM updates, and options ($re\$op$) used by the *suitor*() function. In addition to the estimated optimal rank provided by $re\$rank$, a cross validation error plot is created by default.

6 Running suitor() with the different option values

To run the function *suitor*() with different option values discussed above, we create a list of options as follows. Note that the names in the option list should be matched with names in the table of options above. Again, given the time it takes to run, previously saved results will be loaded.

```
> OP  <- list(min.rank=5, max.rank=13, k.fold=5, n.seeds=50, get.summary=0)
> #re2 <- suitor(SimData, op=OP)
> load(system.file("extdata", "re2.rda", package="SUITOR"))
> str(re2)

List of 2
 $ all.results: num [1:2250, 1:6] 5 6 7 8 9 10 11 12 13 5 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:6] "Rank" "k" "Start" "Error.Train" ...
 $ op         :List of 18
  ..$ min.rank   : num 5
  ..$ max.rank   : num 13
  ..$ k.fold     : num 5
  ..$ n.starts   : num 50
  ..$ get.summary: num 0
  ..$ max.iter   : num 2000
  ..$ em.eps     : num 1e-05
  ..$ plot       : logi TRUE
  ..$ print      : num 1
  ..$ min.value  : num 1e-04
```

```
  ..$ n.cores    : num 1
  ..$ kfold.vec  : int [1:5] 1 2 3 4 5
  ..$ seeds      : int [1:50] 1 2 3 4 5 6 7 8 9 10 ...
  ..$ parMat     : int [1:2250, 1:3] 5 6 7 8 9 10 11 12 13 5 ...
  ..$ parStart   : num 1
  ..$ parEnd     : num 2250
  ..$ algorithm  : num 1
  ..$ type       : chr "FORK"
```

Since *get.summary* is set to 0, *suitor*() only produces a matrix containing all possible results. In that case, as shown below one needs to use the $getSummary(obj, NC, NR)$ function to compute the estimated optimal rank, where *obj* is the matrix containing all results from *suitor*(), NC and NR are the numbers of columns and rows respectively in the input data for *suitor*(). Please note that NR has a default value of 96 for single base substitution signature analysis. The *plotErrors*() function can be used to draw a cross validation error plot.

```
> Summary2 <- getSummary(re2$all.results, ncol(SimData))
> str(Summary2)
```

```
List of 3
 $ rank       : num 8
 $ summary    :'data.frame':        18 obs. of  8 variables:
  ..$ Rank : num [1:18] 5 5 6 6 7 7 8 8 9 9 ...
  ..$ Type : chr [1:18] "Train" "Test" "Train" "Test" ...
  ..$ MSErr: num [1:18] 0.798 0.883 0.738 0.823 0.7 ...
  ..$ fold1: num [1:18] 14636 4457 12550 3945 11292 ...
  ..$ fold2: num [1:18] 14722 4432 12609 3856 11351 ...
  ..$ fold3: num [1:18] 14749 4409 12592 3880 11373 ...
  ..$ fold4: num [1:18] 14626 4621 12562 3891 11264 ...
  ..$ fold5: num [1:18] 14564 4548 12503 3936 11219 ...
 $ all.results: num [1:2250, 1:6] 5 6 7 8 9 10 11 12 13 5 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:6] "Rank" "k" "Start" "Error.Train" ...
```

7 Extracting the signature profiles and activities

Once the optimal number of signatures or called rank is estimated by *suitor*(), we can extract the signature profiles $\hat{W}$ and activities $\hat{H}$ with the function $suitor\_extract\_WH(data, rank, op)$. As in the *suitor*() function, the input data is a data frame or matrix containing a mutational catalog whose elements are non-negative counts. A non-negative integer rank is the number of mutational signatures to be extracted. The possible option values are summarized in the following table and they can be used in the same manner as *suitor*().

Name	Description	Default Value
min.value	Minimum value of matrix before factorizing	1e-4
n.starts	Number of starting points	30
n.cores	Number of cores to use	1
print	0 or 1 to print info (0=no printing)	1

```
> re$rank
```

```
[1] 8
```

```
> #Extract <- suitor_extract_WH(SimData, re$rank)
> load(system.file("extdata", "Extract.rda", package="SUITOR"))
> head(Extract$W)
```

```
            denovo A    denovo B   denovo C     denovo D     denovo E
A[C>A]A 5.518637e-20 0.010288492 0.04983770 1.122787e-03 9.051462e-04
A[C>A]C 9.925830e-04 0.007365908 0.03619431 1.056966e-03 6.523467e-04
A[C>A]G 7.267837e-11 0.001870838 0.01906978 2.335085e-07 4.919793e-20
A[C>A]T 2.590101e-03 0.008468910 0.03532306 5.436354e-20 4.919793e-20
C[C>A]A 5.518637e-20 0.010042262 0.08704119 1.901878e-03 4.919793e-20
C[C>A]C 1.461339e-03 0.003824709 0.10695051 5.436354e-20 4.919793e-20
            denovo F     denovo G     denovo H
A[C>A]A 5.222997e-20 6.787960e-20 5.014520e-03
A[C>A]C 5.222997e-20 2.928684e-03 4.228025e-08
A[C>A]G 5.222997e-20 1.764140e-03 5.655417e-20
A[C>A]T 3.778066e-07 8.427020e-04 2.018224e-03
C[C>A]A 1.038244e-04 7.256455e-03 1.320490e-03
C[C>A]C 1.272696e-06 6.787960e-20 2.425324e-13
```

```
> Extract$H[,1:3]
```

```
                [,1]         [,2]         [,3]
denovo A 8.603342e+01 1.260570e+01 18.015492711
denovo B 2.283030e+01 7.749698e-13  2.708432674
denovo C 2.819917e+00 2.074107e+00  0.046758724
denovo D 1.430752e+01 2.081513e+01 89.761618106
denovo E 8.999499e+01 1.227424e+01  2.234705117
denovo F 4.591597e+01 4.631091e+01  0.000222601
denovo G 1.510148e+01 2.097038e+00 18.027880717
denovo H 8.717979e-13 6.882745e+01 56.209381094
```

$Extract\$W$ and $Extract\$H$ are estimated matrices for the profile $\hat{W}$ and the activity $\hat{H}$, respectively.

8 Summarizing signature profiles with MutationalPatterns

The R package MutationalPatterns (Blokzijl et al., 2021) provides some utility functions to summarize signature profiles and contains matrices of pre-defined signatures like COSMIC. The function $plot\_96\_profile()$ can draw the signature profile plot with respect to the 96 trinucleotide categories. In addition, the function $cos\_sim\_matrix()$ computes the cosine similarity between two profiles.

```
> library(MutationalPatterns)
> COSMIC <- get_known_signatures(source = "COSMIC_v3.2")
> plot_96_profile(Extract$W, condensed=TRUE, ymax=0.3)
> CS <- cos_sim_matrix(Extract$W, COSMIC)
> CS[, 1:3]
```

```
               SBS1        SBS2      SBS3
denovo A 0.013102394 0.002029316 0.4132147
denovo B 0.047350988 0.002278346 0.6455886
denovo C 0.010758046 0.030519166 0.5670385
denovo D 0.047933845 0.711122442 0.2126533
denovo E 0.001271672 0.001889550 0.1062690
denovo F 0.776644297 0.018611915 0.1905089
denovo G 0.014674267 0.018377734 0.4398092
denovo H 0.006683554 0.017863687 0.3020826
```

9 Session Information

```
> sessionInfo()

R version 4.1.0 (2021-05-18)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: CentOS Linux 7 (Core)

Matrix products: default
BLAS/LAPACK: /usr/local/intel/compilers_and_libraries_2020.2.254/linux/mkl/lib/intel64_lin

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats4    parallel  stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
 [1] MutationalPatterns_3.4.0 NMF_0.23.0               synchronicity_1.3.5
 [4] bigmemory_4.5.36         Biobase_2.54.0           cluster_2.1.2
 [7] rngtools_1.5.2           pkgmaker_0.32.2          registry_0.5-1
[10] GenomicRanges_1.46.1     GenomeInfoDb_1.30.0      IRanges_2.28.0
[13] S4Vectors_0.32.3         BiocGenerics_0.40.0      SUITOR_0.99.1
[16] ggplot2_3.3.5            doParallel_1.0.17        iterators_1.0.14
[19] foreach_1.5.2

loaded via a namespace (and not attached):
 [1] tidyselect_1.1.1       purrr_0.3.4            reshape2_1.4.4
 [4] colorspace_2.0-2       vctrs_0.3.8            generics_0.1.2
 [7] pracma_2.3.6           utf8_1.2.2             rlang_1.0.1
[10] pillar_1.7.0           glue_1.6.1             withr_2.4.3
[13] DBI_1.1.2              RColorBrewer_1.1-2     uuid_1.0-3
[16] GenomeInfoDbData_1.2.7 lifecycle_1.0.1        plyr_1.8.6
```

```
[19] stringr_1.4.0          zlibbioc_1.40.0        munsell_0.5.0
[22] gtable_0.3.0           ggalluvial_0.12.3      codetools_0.2-18
[25] fansi_1.0.2            Rcpp_1.0.8             xtable_1.8-4
[28] scales_1.1.1           XVector_0.34.0         farver_2.1.0
[31] digest_0.6.29          stringi_1.7.6          dplyr_1.0.7
[34] grid_4.1.0             cli_3.1.1              tools_4.1.0
[37] bitops_1.0-7           magrittr_2.0.2         RCurl_1.98-1.5
[40] tibble_3.1.6           tidyr_1.2.0            crayon_1.4.2
[43] bigmemory.sri_0.1.3    pkgconfig_2.0.3        ellipsis_0.3.2
[46] gridBase_0.4-7         assertthat_0.2.1       R6_2.5.1
[49] compiler_4.1.0
```