# SUITOR: selecting the number of mutational signatures

***DongHyuk Lee and Bin Zhu***

**9 November 2022**

## Contents

# 1 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.

# 2 Installing the SUITOR package from Bioconductor

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

# 3 Loading the package

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

```
library(SUITOR)
```

# 4 Example data

For illustrative purposes, we simulated a 96 by 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 with the mean corresponding to each element of `WH`, where `W` is the true signature matrix of size 96 by 8 for 8 signatures, and `H` is the activity matrix of size 8 by 300. Specifically for `W`, we used the profile of eight COSMIC signatures 4, 6, 7a, 9, 17b, 22, 26, 39 from **COSMIC**. `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
```

# 5 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.

## 5.1 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** for extracting signature profiles using `suitorExtractWH()` after estimating the optimal rank.

## 5.2 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 |
| 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 |

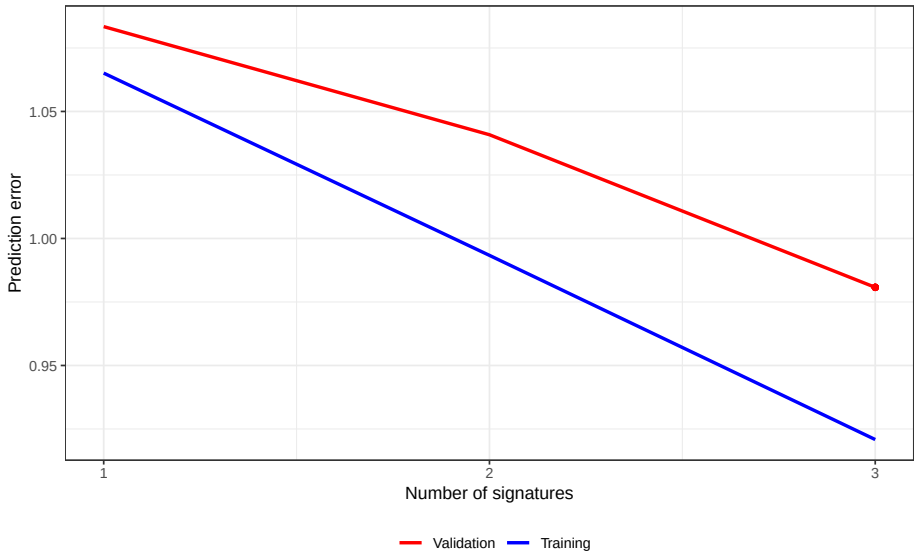
The option `min.value` is a small number added to the data matrix for stable computation of non-negative matrix factorization. For a given number of signatures or ranks `rnk (min.rank <= rnk <= max.rank)`, the data matrix is divided into `k.fold` parts for the cross-validation. The default value of the 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 <= rnk <= max.rank`) with more starts (`n.starts`) and a larger number of maximal iteration (`max.iter`) if necessary.

# 6 Running suitor()

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.

```
OP  <- list(max.rank=3, k.fold=5, n.starts=4)
set.seed(123)
re <- suitor(SimData, op=OP)
## start = 1, rank = 1, k = 1
## start = 1, rank = 2, k = 1
## start = 1, rank = 3, k = 1
## start = 1, rank = 1, k = 2
## start = 1, rank = 2, k = 2
## start = 1, rank = 3, k = 2
## start = 1, rank = 1, k = 3
## start = 1, rank = 2, k = 3
## start = 1, rank = 3, k = 3
## start = 1, rank = 1, k = 4
## start = 1, rank = 2, k = 4
## start = 1, rank = 3, k = 4
## start = 1, rank = 1, k = 5
## start = 1, rank = 2, k = 5
## start = 1, rank = 3, k = 5
## start = 2, rank = 1, k = 1
## start = 2, rank = 2, k = 1
## start = 2, rank = 3, k = 1
## start = 2, rank = 1, k = 2
## start = 2, rank = 2, k = 2
## start = 2, rank = 3, k = 2
## start = 2, rank = 1, k = 3
## start = 2, rank = 2, k = 3
## start = 2, rank = 3, k = 3
## start = 2, rank = 1, k = 4
## start = 2, rank = 2, k = 4
## start = 2, rank = 3, k = 4
## start = 2, rank = 1, k = 5
## start = 2, rank = 2, k = 5
## start = 2, rank = 3, k = 5
## start = 3, rank = 1, k = 1
## start = 3, rank = 2, k = 1
## start = 3, rank = 3, k = 1
## start = 3, rank = 1, k = 2
## start = 3, rank = 2, k = 2
## start = 3, rank = 3, k = 2
## start = 3, rank = 1, k = 3
## start = 3, rank = 2, k = 3
## start = 3, rank = 3, k = 3
```

```
## start = 3, rank = 1, k = 4
## start = 3, rank = 2, k = 4
## start = 3, rank = 3, k = 4
## start = 3, rank = 1, k = 5
## start = 3, rank = 2, k = 5
## start = 3, rank = 3, k = 5
## start = 4, rank = 1, k = 1
## start = 4, rank = 2, k = 1
## start = 4, rank = 3, k = 1
## start = 4, rank = 1, k = 2
## start = 4, rank = 2, k = 2
## start = 4, rank = 3, k = 2
## start = 4, rank = 1, k = 3
## start = 4, rank = 2, k = 3
## start = 4, rank = 3, k = 3
## start = 4, rank = 1, k = 4
## start = 4, rank = 2, k = 4
## start = 4, rank = 3, k = 4
## start = 4, rank = 1, k = 5
## start = 4, rank = 2, k = 5
## start = 4, rank = 3, k = 5
```




```
str(re)
## List of 4
##  $ rank       : num 3
##  $ summary    :'data.frame': 6 obs. of  8 variables:
##   ..$ Rank : num [1:6] 1 1 2 2 3 3
##   ..$ Type : chr [1:6] "Train" "Test" "Train" "Test" ...
##   ..$ MSErr: num [1:6] 1.065 1.083 0.993 1.041 0.921 ...
##   ..$ fold1: num [1:6] 26237 6652 22766 6476 19507 ...
##   ..$ fold2: num [1:6] 26293 6600 22819 6062 19583 ...
##   ..$ fold3: num [1:6] 26196 6699 22799 6078 19644 ...
##   ..$ fold4: num [1:6] 26014 6887 22716 6163 19525 ...
```

```
##   ..$ fold5: num [1:6] 25947 6967 22572 6421 19425 ...
##  $ all.results: num [1:60, 1:6] 1 2 3 1 2 3 1 2 3 1 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:6] "Rank" "k" "Start" "Error.Train" ...
##  $ op         :List of 18
##   ..$ max.rank   : num 3
##   ..$ k.fold     : num 5
##   ..$ n.starts   : num 4
##   ..$ min.rank   : num 1
##   ..$ max.iter   : num 2000
##   ..$ em.eps     : num 1e-05
##   ..$ plot       : logi TRUE
##   ..$ print      : num 1
##   ..$ min.value  : num 1e-04
##   ..$ get.summary: num 1
##   ..$ kfold.vec  : int [1:5] 1 2 3 4 5
##   ..$ seeds      : int [1:4] 1 2 3 4
##   ..$ n.cores    : num 1
##   ..$ BPPARAM    :Reference class 'SerialParam' [package "BiocParallel"] with 19 fields
##   .. ..$ workers        : int 1
##   .. ..$ tasks          : int 0
##   .. ..$ jobname        : chr "BPJOB"
##   .. ..$ progressbar    : logi FALSE
##   .. ..$ log            : logi FALSE
##   .. ..$ logdir         : chr NA
##   .. ..$ threshold      : chr "INFO"
##   .. ..$ resultdir      : chr NA
##   .. ..$ stop.on.error  : logi TRUE
##   .. ..$ timeout        : int NA
##   .. ..$ exportglobals  : logi FALSE
##   .. ..$ exportvariables: logi FALSE
##   .. ..$ RNGseed        : NULL
##   .. ..$ RNGstream      : NULL
##   .. ..$ force.GC       : logi FALSE
##   .. ..$ fallback       : logi FALSE
##   .. ..$ .finalizer_env :<environment: 0x1130e8038>
##   .. ..$ .uid           : chr(0)
##   .. ..$ backend        : NULL
##   .. ..and 15 methods, of which 1 is  possibly relevant:
##   .. ..  show#envRefClass
##   ..$ parMat     : int [1:60, 1:3] 1 2 3 1 2 3 1 2 3 1 ...
##   ..$ parStart   : num 1
##   ..$ parEnd     : num 60
##   ..$ algorithm  : num 1
```

# 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 `W` and activities `H` with the function `suitorExtractWH(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 |
| print | 0 or 1 to print info (0=no printing) | 1 |

```
re$rank
## [1] 3
set.seed(123)
Extract <- suitorExtractWH(SimData, re$rank)
##
 start 1
 start 2
 start 3
 start 4
 start 5
 start 6
 start 7
 start 8
 start 9
 start 10
 start 11
 start 12
 start 13
 start 14
 start 15
 start 16
 start 17
 start 18
 start 19
 start 20
 start 21
 start 22
 start 23
 start 24
 start 25
 start 26
 start 27
 start 28
 start 29
 start 30
head(Extract$W)
##            denovo A     denovo B     denovo C
## A[C>A]A 0.013425452 7.815586e-17 1.747062e-03
```

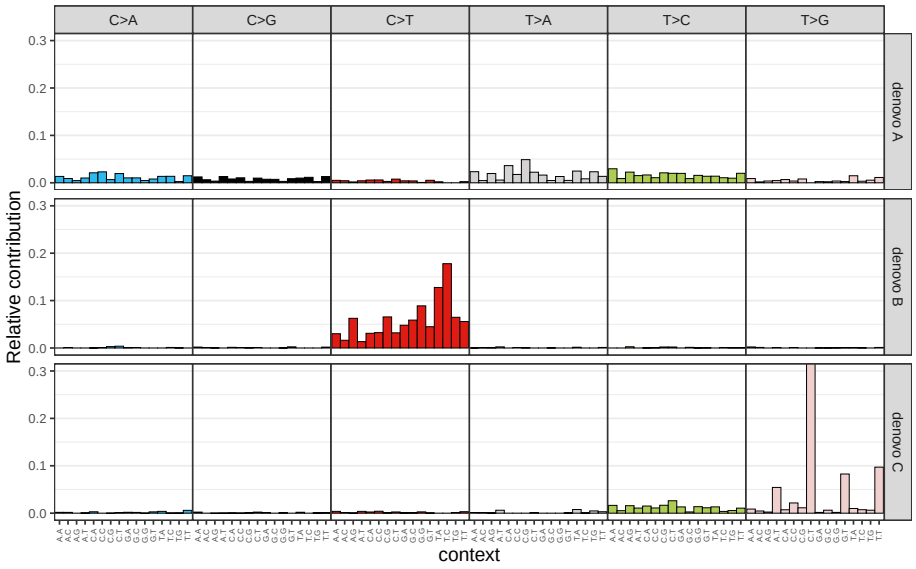
```
## A[C>A]C 0.009022917 1.002781e-03 1.696961e-03
## A[C>A]G 0.004639836 1.887708e-06 7.178991e-20
## A[C>A]T 0.010069058 3.058166e-19 5.781870e-04
## C[C>A]A 0.021047613 1.951143e-05 2.938524e-03
## C[C>A]C 0.023021126 7.621931e-04 7.178991e-20
Extract$H[,1:3]
##               [,1]     [,2]      [,3]
## denovo A  58.42606 87.34461 97.331080
## denovo B  58.62291 63.49136 85.972758
## denovo C 159.95458 14.16863  3.700667
```

`Extract$W` and `Extract$H` are estimated matrices for the profile `W` and the activity `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.

```
suppressPackageStartupMessages(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 0.040068483 0.024860249 0.7939277
```

```
## denovo B 0.457004654 0.586714824 0.2740816
## denovo C 0.008262569 0.004342014 0.1726003
```

# 9 Session Information

```
sessionInfo()
## R version 4.2.1 (2022-06-23)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS Ventura 13.0
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRlapack.dylib
##
## locale:
## [1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] MutationalPatterns_3.8.0 NMF_0.24.0              bigmemory_4.6.1
##  [4] Biobase_2.58.0           cluster_2.1.3           rngtools_1.5.2
##  [7] pkgmaker_0.32.2          registry_0.5-1          GenomicRanges_1.50.1
## [10] GenomeInfoDb_1.34.2      IRanges_2.32.0          S4Vectors_0.36.0
## [13] BiocGenerics_0.44.0      SUITOR_1.0.0            BiocStyle_2.26.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.9             tidyr_1.2.0            assertthat_0.2.1
##  [4] digest_0.6.29          foreach_1.5.2          utf8_1.2.2
##  [7] gridBase_0.4-7         plyr_1.8.7             R6_2.5.1
## [10] ggalluvial_0.12.3      pracma_2.3.8           bigmemory.sri_0.1.3
## [13] evaluate_0.15          ggplot2_3.3.6          pillar_1.7.0
## [16] zlibbioc_1.44.0        rlang_1.0.4            uuid_1.1-0
## [19] rmarkdown_2.14         labeling_0.4.2         BiocParallel_1.32.1
## [22] stringr_1.4.0          RCurl_1.98-1.7         munsell_0.5.0
## [25] compiler_4.2.1         xfun_0.31              pkgconfig_2.0.3
## [28] htmltools_0.5.2        tidyselect_1.1.2       tibble_3.1.7
## [31] GenomeInfoDbData_1.2.8 bookdown_0.27          codetools_0.2-18
## [34] fansi_1.0.3            crayon_1.5.1           dplyr_1.0.9
## [37] withr_2.5.0            bitops_1.0-7           grid_4.2.1
## [40] xtable_1.8-4           gtable_0.3.0           lifecycle_1.0.1
## [43] DBI_1.1.3              magrittr_2.0.3         scales_1.2.0
## [46] cli_3.3.0              stringi_1.7.8          reshape2_1.4.4
## [49] farver_2.1.1           XVector_0.38.0         doParallel_1.0.17
## [52] ellipsis_0.3.2         generics_0.1.3         vctrs_0.4.1
## [55] RColorBrewer_1.1-3     iterators_1.0.14       tools_4.2.1
## [58] glue_1.6.2             purrr_0.3.4            parallel_4.2.1
```

```
## [61] fastmap_1.1.0          yaml_2.3.5             colorspace_2.0-3
## [64] BiocManager_1.30.18    knitr_1.39
```