Rare Mutational Signatures Extraction workflow - user manual

signature.tools.lib version: 2.2.0
latest edit: 30/09/2022

Andrea Degasperi, University of Cambridge, UK
ad923@cam.ac.uk

1. Introduction

This document describes how to use the *signature.tools.lib* R package to set up an R workflow to extract common and rare mutational signatures from a matrix of single base substitution (SBS) mutational catalogues.

2. Installation and documentation

The functions used below are available from v2.2.0 of the signature.tools.lib R package. You can download it and install it from GitHub:

https://github.com/Nik-Zainal-Group/signature.tools.lib

Once signature.tools.lib is installed, you can type *?nameoffunction* in R for an up-to-date documentation of each function.

3. Example of common and rare signature extraction workflow

The R code for this example is available as *Example05.R* in the *examples/* folder of the signature.tools.lib GitHub.

In this example, we provide an example of signature extraction workflow where both common and rare signatures are extracted.

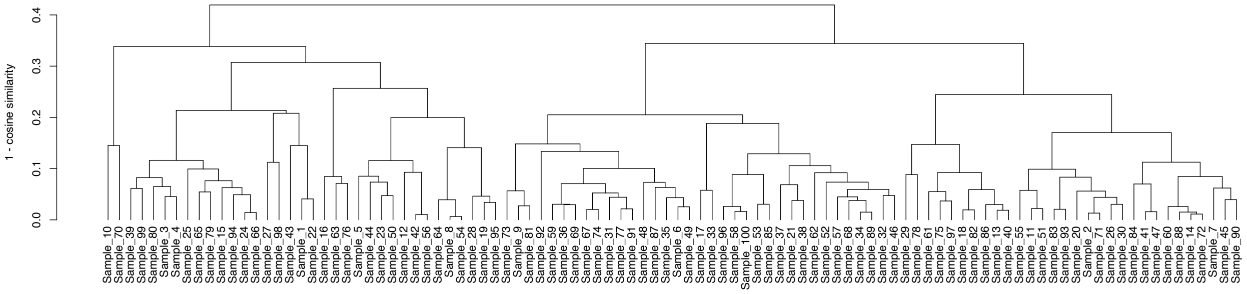
We being by importing the package and setting an output directory where we will store all the results.

```
library(signature.tools.lib)
outdir <- "~/Example05/"
dir.create(outdir,showWarnings = F,recursive = T)
```

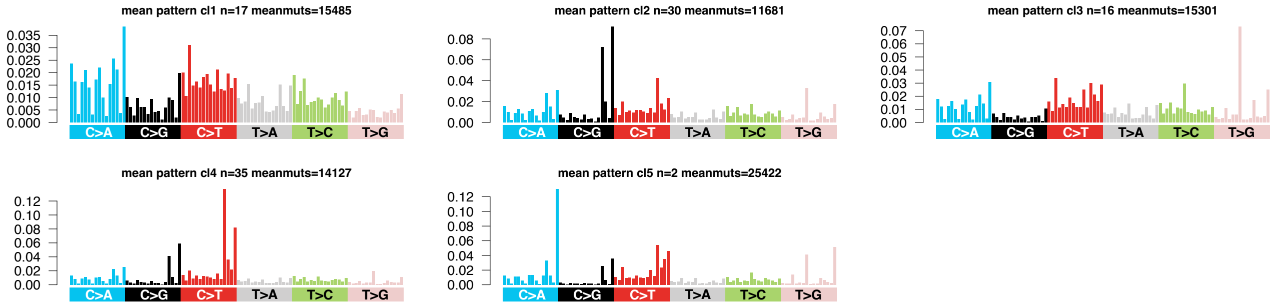
We load the catalogues and perform a clustering to see if samples with rare profiles can be identified and excluded from the extraction of common signatures. The data *SDExample05* are simulated data with 100 catalogues (9 common breast cancer signatures and 2 rare signatures).

```
catalogues <- readTable("tests/testthat/rareExtraction/",
                        "SDExample05/catalogues.tsv")
resCl <- cataloguesClustering(catalogues,nclusters = 1:15,
                              outdir = paste0(outdir,"cataloguesClustering/"))
```

Clustering results should produce a dendrogram as shown below.



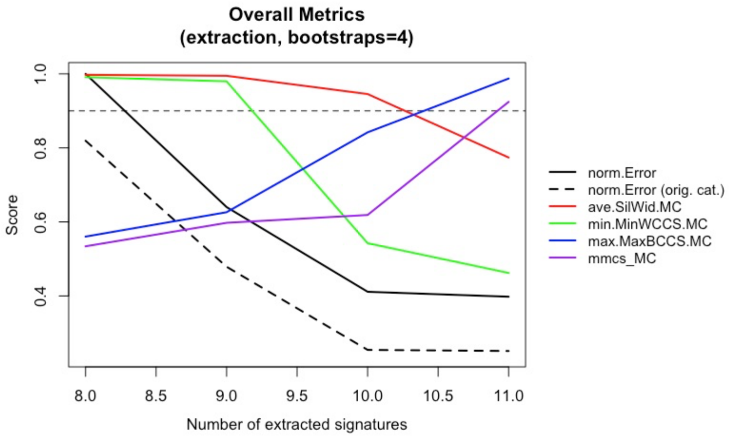
After inspection of the clustering results, we can clearly identify one cluster with just two samples that we can exclude (nclusters=5, remove cluster 5).



Then, we perform a signature extraction using NMF on the remaining samples.

```
nclustersSelection <- "5"
clusters <- resCl$clusters_table[,nclustersSelection]
samples <- rownames(resCl$clusters_table)
clusters_to_keep <- c(1:4)
samplesSelected <- samples[clusters %in% clusters_to_keep]
SignatureExtraction(cat = catalogues[,samplesSelected],
                    outFilePath = paste0(outdir,"Extraction/"),
                    nrepeats = 25,
                    nboots = 4,
                    filterBestOfEachBootstrap = T,
                    nparallel = 4,
                    nsig = 8:11,
                    plotResultsFromAllClusteringMethods = F,
                    parallel = T)
```

Notice that this is just for illustration, and for a realistic example with hundreds or thousands of samples we advise to use at least 20 bootstraps (*nboots=20*) and 2-300 repeats each (*nrepeats=200*). Each bootstrap will sit on a different parallel process, so one can set *nparallel* accordingly. The range for the number of signatures (*nsig*) can also be wider. We now need to decide how many signatures are present, based on the error and the average silhouette width (ASW).



You will notice that there are various metrics tracked here, the most important ones are the ASW (*ave.SilWid.MC*) and the error with respect to the original catalogue (*norm.Error orig. cat.*), as opposed to the error with respect to each bootstrap catalogue (*norm.Error*).

In this case, 10 signatures seem optimal, as it is the point just before the ASW drops and the error w.r.t. the original catalogue reaches a plateau. This is correct, because there are 9 common and 1 rare signatures left after we removed all samples with one of the two rare signatures at the initial clustering.

If we inspect the solution with 9 signatures, we can see that the 9 common signatures are reported correctly, and the rare signature is not reported.

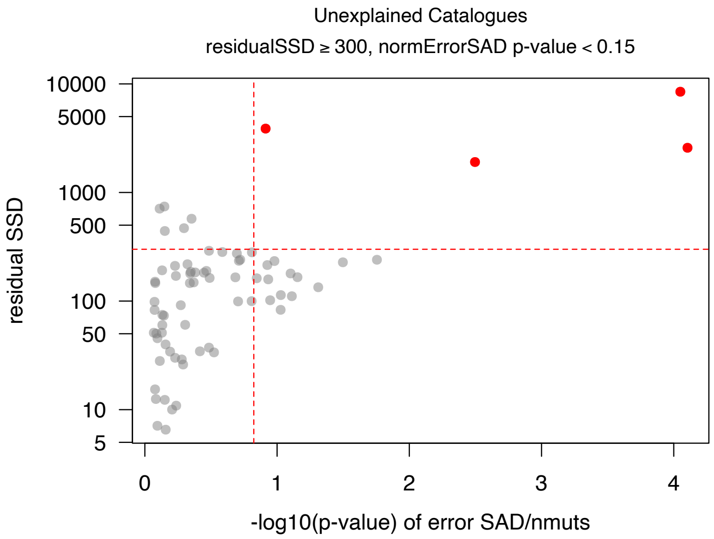
Let us choose 9 here, just as an illustrative example, to show how the following steps will be able to recover both rare signatures and assign them to the correct samples. You can try for yourself to select 10 signatures and see how the results are affected downstream.

```
estimated_signatures <- readTable(paste0(outdir,"Extraction/round_1/sig_9/",
                                 "Sigs_plot_extraction_ns9_nboots4.tsv"))
```

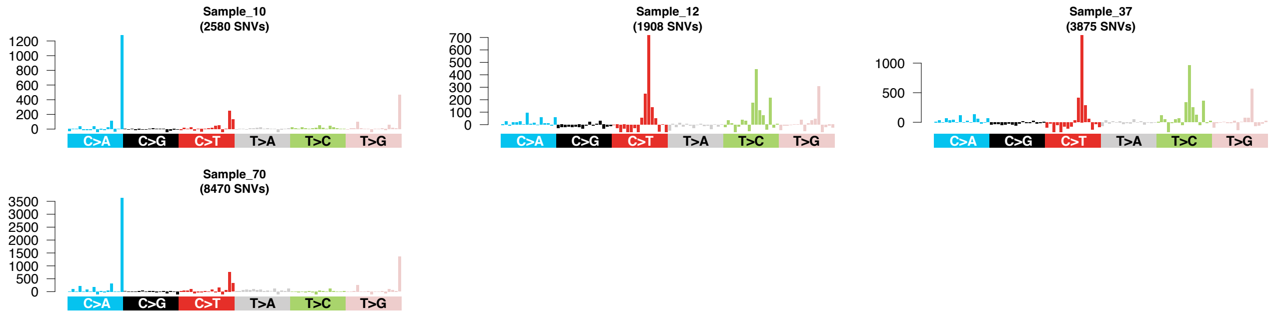
Now we have a set of common signatures and we want to find whether some samples are still not fully explained by these signatures and see if we can extract rare signatures out of them.

```
unexplSamples <- unexplainedSamples(outfileRoot = paste0(outdir,
                                                  "unexplained/Example05"),
                                   catalogues = catalogues,
                                   sigs = estimated_signatures,
                                   nmuts_threshold = 300,
                                   pvalue_threshold = 0.15)
```

Indeed, there are four samples that are not fully explained. As there is some randomness you might have to tune the *nmuts_threshold* and *pvalue_threshold* parameters to select them. Use the *Example05_UnexplainedCataloguesSelectionPlot.pdf* scatter plot in the unexplained folder to guide you.



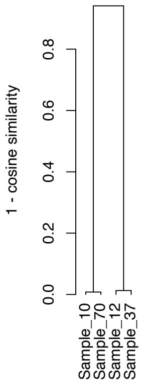
The residuals of the four samples are shown below.

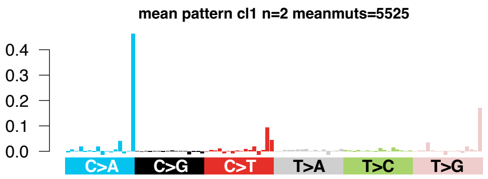


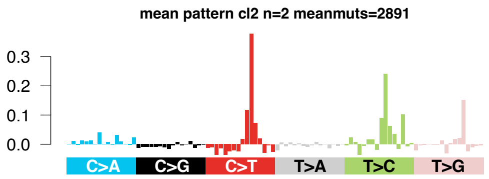
We cluster the residuals of the four samples, and we identify two clusters of two samples, each of them having very similar residuals. We can use the samples in these clusters to then extract two rare mutational signatures.

```
all_residuals <- unexplSamples$all_residuals
significant_residuals <- all_residuals[,unexplSamples$which_significant]
resCl_residuals <- cataloguesClustering(significant_residuals,
                                        nclusters = 1:3,
                                        outdir = paste0(outdir,
                                                    "residualClustering/"))
```

The result of the residuals clustering is shown below.







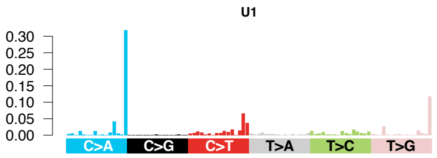
At this point, we know in which samples the rare signatures are, so we can extract them.

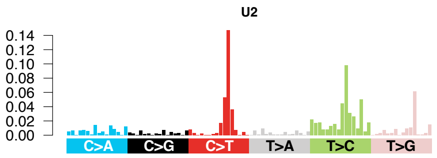
```
resRareSigs <- rareSignatureExtraction(outfileRoot = paste0(outdir,
                                                   "ExtractionRare/Example05"),
                             catalogues = catalogues,
                             residuals = unexplSamples$all_residuals,
                             unexpl_samples = unexplSamples$unexplSamples,
                             clusters = resCl_residuals$clusters_table[,"2"],
                             useclusters = list(c(1),c(2)),
                             commonSignatures = estimated_signatures,
                             commonExposures = unexplSamples$exposures)
```

The most important parameter of the *rareSignatureExtraction* function is *useclusters*. This is a list with as many elements as how many rare signatures we want to extract. Each element is a vector of numbers, indicating the clusters to use for each rare signature. This means that it is possible to merge two or more clusters in one rare signature extraction. Here we simply indicate one cluster for each of the two rare signatures to extract.

The extraction will use a variant of NMF where the common signatures are fitted while extracting one additional signature. This procedure might have infinite solutions, so to help the algorithm to use the common signatures before finding something new, we set the starting point of the exposures to the exposures obtained from fitting the common signatures only. You can also limit the number of iterations of NMF by using the *maxiter* parameter.

The rare signatures extracted are shown below.





Finally, we can use all the common and rare signatures, as well as the information about where we extracted the rare signatures from, to estimate the exposures of each signature in each sample.

```
resFinalExpo <- finaliseCommonRareSignatureExposures(outfileRoot =
                                              paste0(outdir,
                                              "ExtractionRare/Example05"),
                        catalogues = catalogues,
                        commonSigs = estimated_signatures,
                        listofsignatures = resRareSigs$listofsignatures,
                        listofsamples = resRareSigs$listofsamples,
                        nboot = 50,nparallel = 4)
```

Also in this case, please bear in mind this is an illustration and that we recommend to use a higher number of bootstraps (e.g., *nboot = 200*) in a realistic scenario.

Since we analysed a simulated dataset, we can check how well we did as follows. We load the true values for the signatures and exposures from the simulated dataset *SDExample05*.

```
true_signatures <- readTable("tests/testthat/rareExtraction/",
                             "SDExample05/signatures.tsv")
true_exposures <- readTable("tests/testthat/rareExtraction/",
                            "SDExample05/exposures.tsv")
```

We can then try to match our estimated signatures, both common and rare, with the true signatures.

```
sigsPerf <- evaluatePerformanceSignatureSimilarity(true_signatures,
                                        resRareSigs$commonAndRareSignatures)
```

In order to check the performance of the exposures, we need to update the temporary names of the extracted signatures with the names of the corresponding true signatures.

```
  estimated_exposures <- resFinalExpo$fitWithRare$exposures
  rownames(estimated_exposures) <-
updateSigNamesWithMatchTable(rownames(estimated_exposures),
                                        sigsPerf$matchTable)
```

Finally, we can check the performance of our estimated exposures for common and rare signatures separately.

```
sigNames <- colnames(true_signatures)
whichCommon <- grepl(sigNames,pattern = "common")
commonNames <- sigNames[whichCommon]
rareNames <- sigNames[!whichCommon]
expPerf <- evaluatePerformanceExposures(t(true_exposures),
                                        t(estimated_exposures),
                                        commonNames = commonNames,
                                        rareNames = rareNames)
```