BAGEL Mutational Signature Analysis Toolkit

true true

Contents

Introduction

BAGEL Mutational Signature TookKit discovers novel signatures and predicts sample exposure to known signatures across multiple motif classes including single base substitutions (SBS), double base substitutions (DBS), insertions (INS) and deletions (DEL) and SBS with replication strand and SBS with transcription strand. BAGEL also plots signatures and sample exposures along with advanced downstream analysis including UMAP.

Installation

Currently BAGEL can be installed from Github; in the future it will be available on Bioconductor. To install from Github directly please use the code below: library(devtools) install_github("campbio/BAGEL")

Setting up a bagel object

In order to discover or predict, we must first set up our BAGEL object by 1) extracting variants, 2) selecting a genome 3) creating our BAGEL object 4) building a counts table for our BAGEL (SBS, DBS, INS, DEL, etc.)

Reproducibility note

Many functions in *BAGEL* make use of stochastic algorithms or procedures which require the use of random number generator (RNG) for simulation or sampling. To maintain reproducibility, all these functions use a **default seed of 1** to make sure same results are generated each time one of these functions is called. Explicitly setting the `seed` arguments is needed for greater control and randomness. These functions include *discover_signatures*, *predict_exposure*, and *create_umap*.

Extracting variants

Variants can be extracted from VCFs and MAFs, and R objects such as data.frames and VCF and MAF representation via VariantAnnotation and maftools.

```
# Extract variants from a MAF File
lusc_maf <- system.file("testdata", "public_TCGA.LUSC.maf", package = "BAGEL")
lusc.variants <- extract_variants_from_maf_file(maf_file = lusc_maf)

# Extract variants from an individual VCF file
luad_vcf <- system.file("testdata", "public_LUAD_TCGA-97-7938.vcf",
                        package = "BAGEL")
luad.variants <- extract_variants_from_vcf_file(vcf_file = luad_vcf)

# Extract variants from multiple files and/or objects
melanoma_vcfs <- list.files(system.file("testdata", package = "BAGEL"),
                            pattern = glob2rx("*SKCM*vcf"), full.names = TRUE)
variants <- extract_variants(c(lusc_maf, luad_vcf, melanoma_vcfs), verbose = TRUE)
```

```
##   |                                                                      |
## Extracted 1 out of 5 inputs: /Users/atgc/Documents/Aaron_Tools/BAGEL/inst/testdata/public_TCGA.LUSC.
##   |                                                                      |==================
## Extracted 2 out of 5 inputs: /Users/atgc/Documents/Aaron_Tools/BAGEL/inst/testdata/public_LUAD_TCGA-
##   |                                                                      |==================
## Extracted 3 out of 5 inputs: /Users/atgc/Documents/Aaron_Tools/BAGEL/inst/testdata/public_SKCM_TCGA-
##   |                                                                      |==================
## Extracted 4 out of 5 inputs: /Users/atgc/Documents/Aaron_Tools/BAGEL/inst/testdata/public_SKCM_TCGA-
##   |                                                                      |==================
## Extracted 5 out of 5 inputs: /Users/atgc/Documents/Aaron_Tools/BAGEL/inst/testdata/public_SKCM_TCGA-
```

Choosing genome

BAGEL uses BSgenome objects to access genomic information such as reference bases for count tables. The BSgenome represents and stores full genome sequeces for different organisms. To make selection easy we provide an accessor method for human genome build versions 38, and 19 (hg38, hg19), but many other less common builds and organisms are available via BSgenome as well.

```
g = select_genome("hg38")
```

Create bagel object

```
bagel = create_bagel(x = variants, genome = g)
```

```
## Checking that chromosomes in the 'variant' object match chromosomes in the 'genome' object.

## Checking that the reference bases in the 'variant' object match the reference bases in the 'genome' o

## Warning in .check_variant_ref_in_genome(dt = dt, genome = genome): Reference
## bases for 6 out of 911 variantsdid not match the reference base in the 'genome'.
## Make sure the genome reference is correct.
```

Create mutation count tables

Motifs are the building blocks of mutational signatures. Motifs themselves are a mutation combined with other genomic information. For instance, **SBS96** motifs are constructed from an SBS mutation and one upsteam and one downstream base sandwiched together. We build tables by counting these motifs for each sample.

```
build_standard_table(bagel, "SBS96")
```

Discover Signatures/Exposures

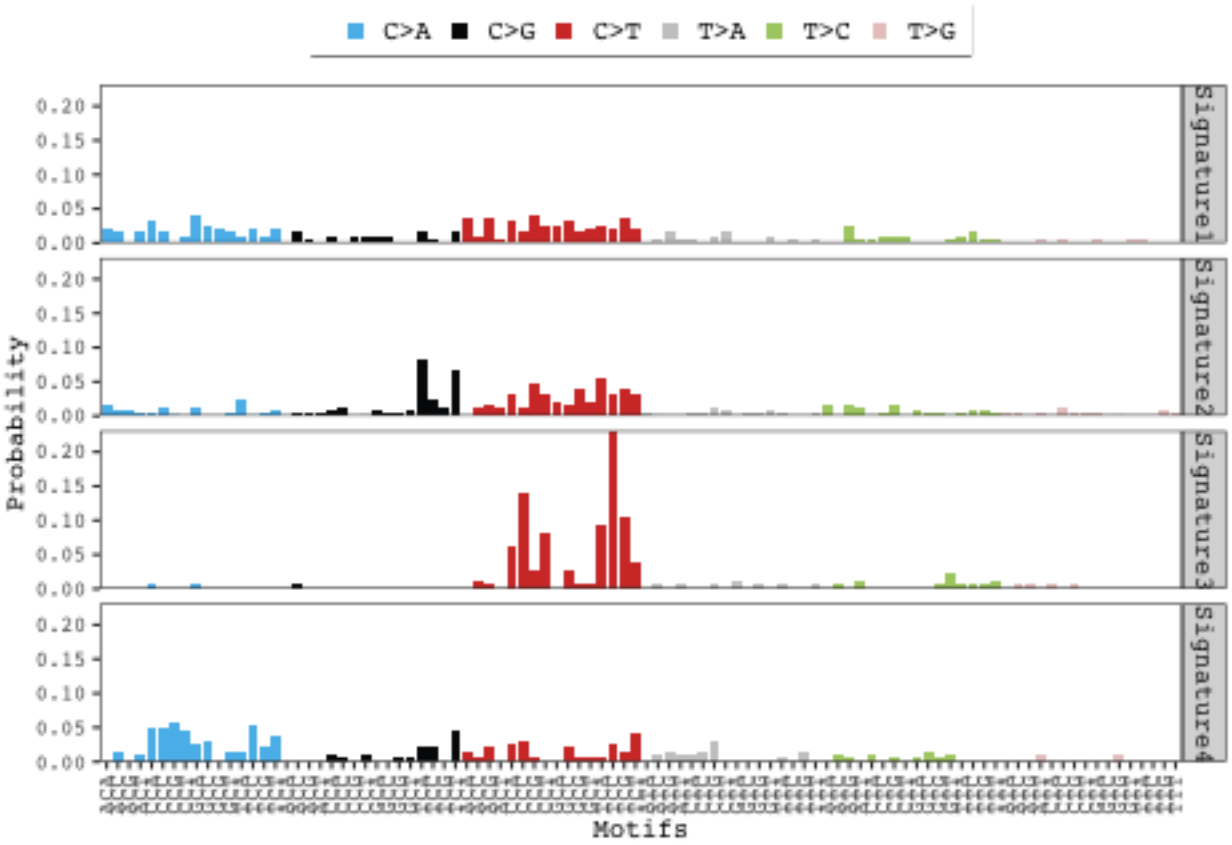
Discovery and prediction result are loaded into a self-contained result object that includes signatures and sample exposures.

```
result = discover_signatures(bagel = bagel, table_name = "SBS96",
                             num_signatures = 4, method = "lda", nstart = 10,
                            seed = 1)
```

Plotting

Signatures

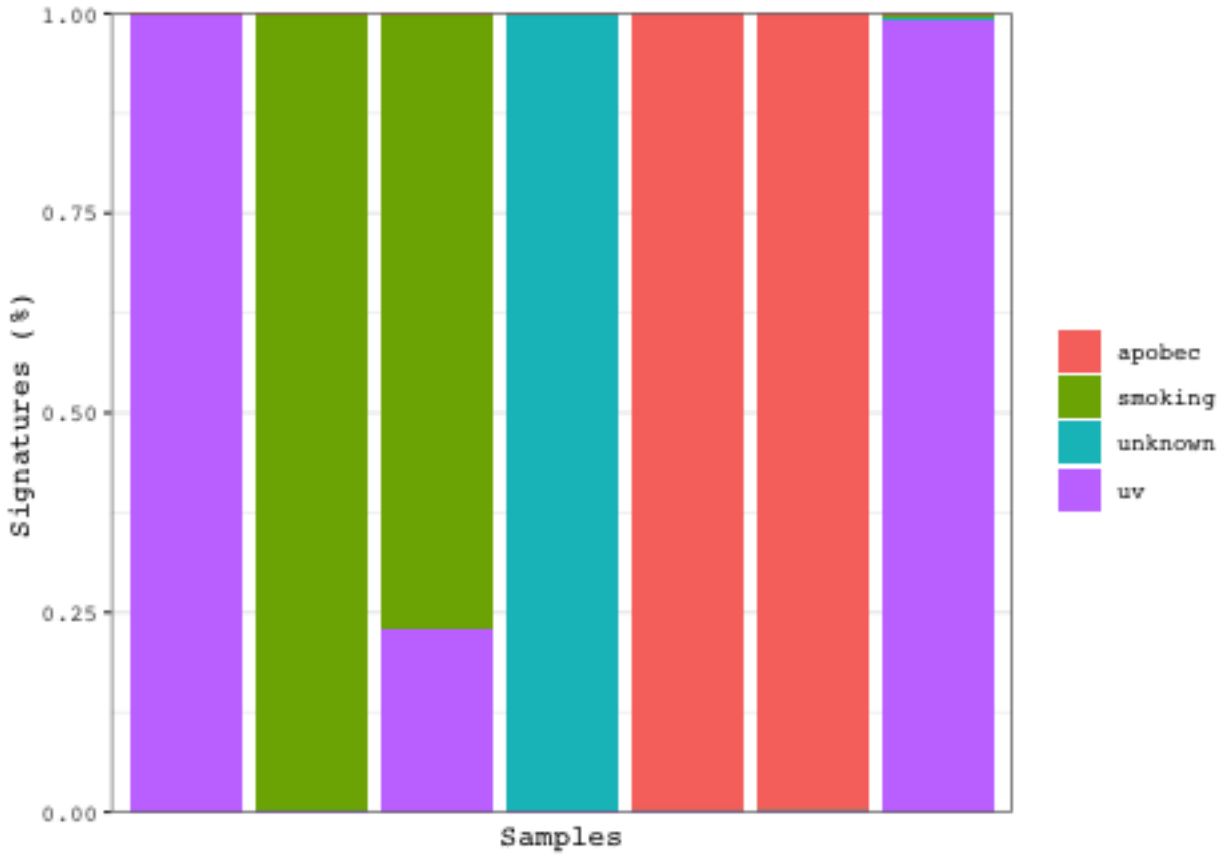
```
##Plot results
plot_signatures(result)
```



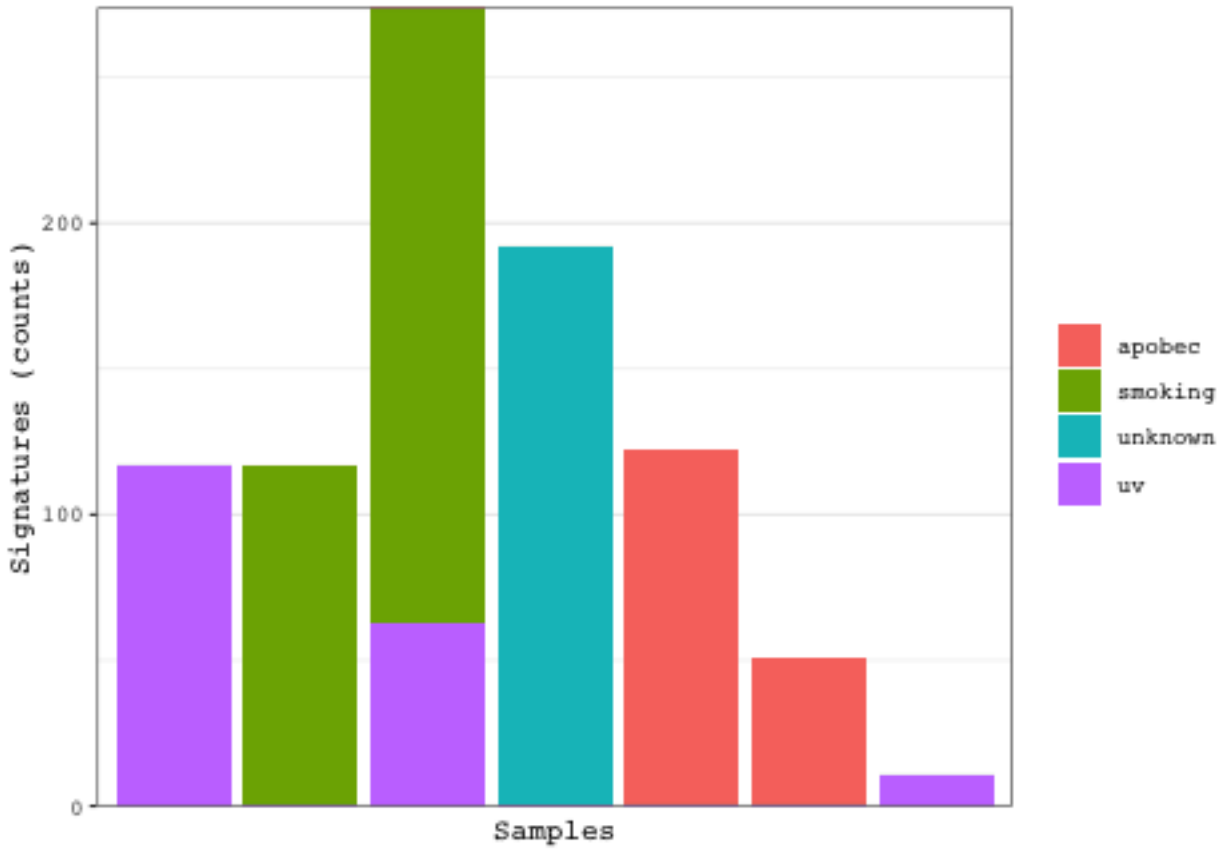

```
# We can name signatures based on prior knowledge
name_signatures(result, c("unknown", "smoking", "apobec", "uv"))
```

Exposures

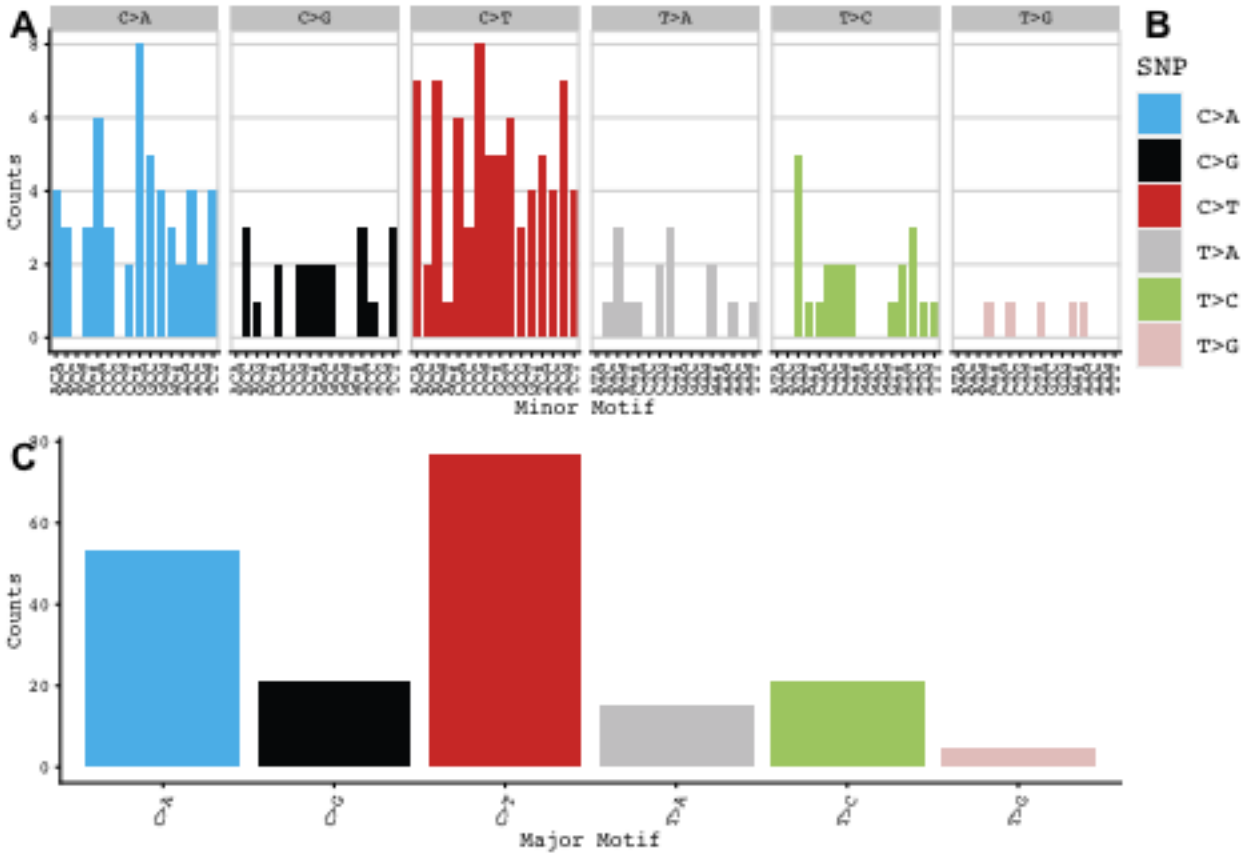
```
plot_exposures(result, proportional = TRUE)
```




```
plot_exposures(result, proportional = FALSE)
```



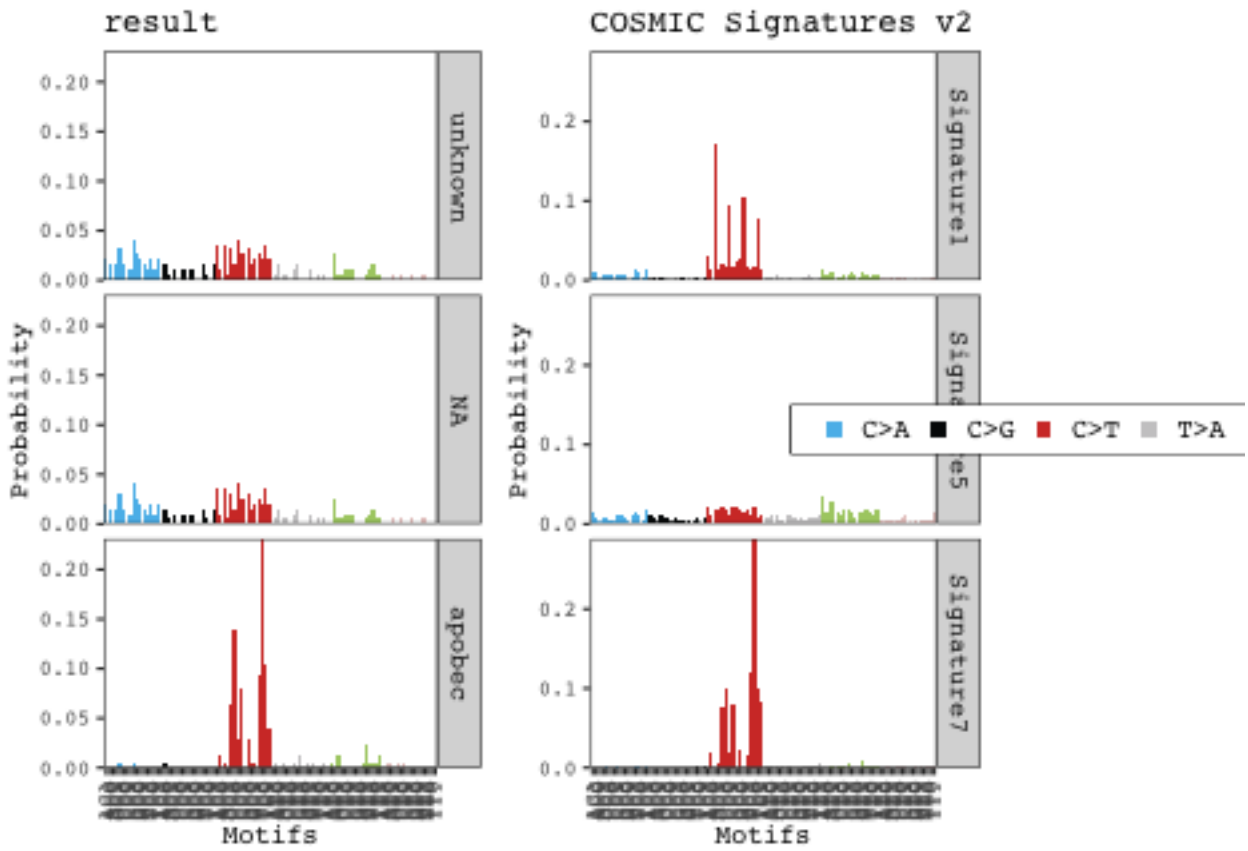

```
plot_sample_counts(bagel, "SBS96", get_sample_names(bagel)[1])
```



Comparison to external signatures (e.g. COSMIC)

```
##Compare to COSMIC signatures by leaving the second result as default
compare_cosmic_v2(result, threshold = 0.78)
```

```
## Warning: The `.data` argument of `add_column()` must have unique names as of tibble 3.0.0.
## Use `.name_repair = "minimal"`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```




```
##         cor xindex yindex    xcol       ycol
## 1 0.7828181      1      1 unknown Signature1
## 2 0.7969306      1      5 unknown Signature5
## 3 0.9160303      3      7  apobec Signature7
```

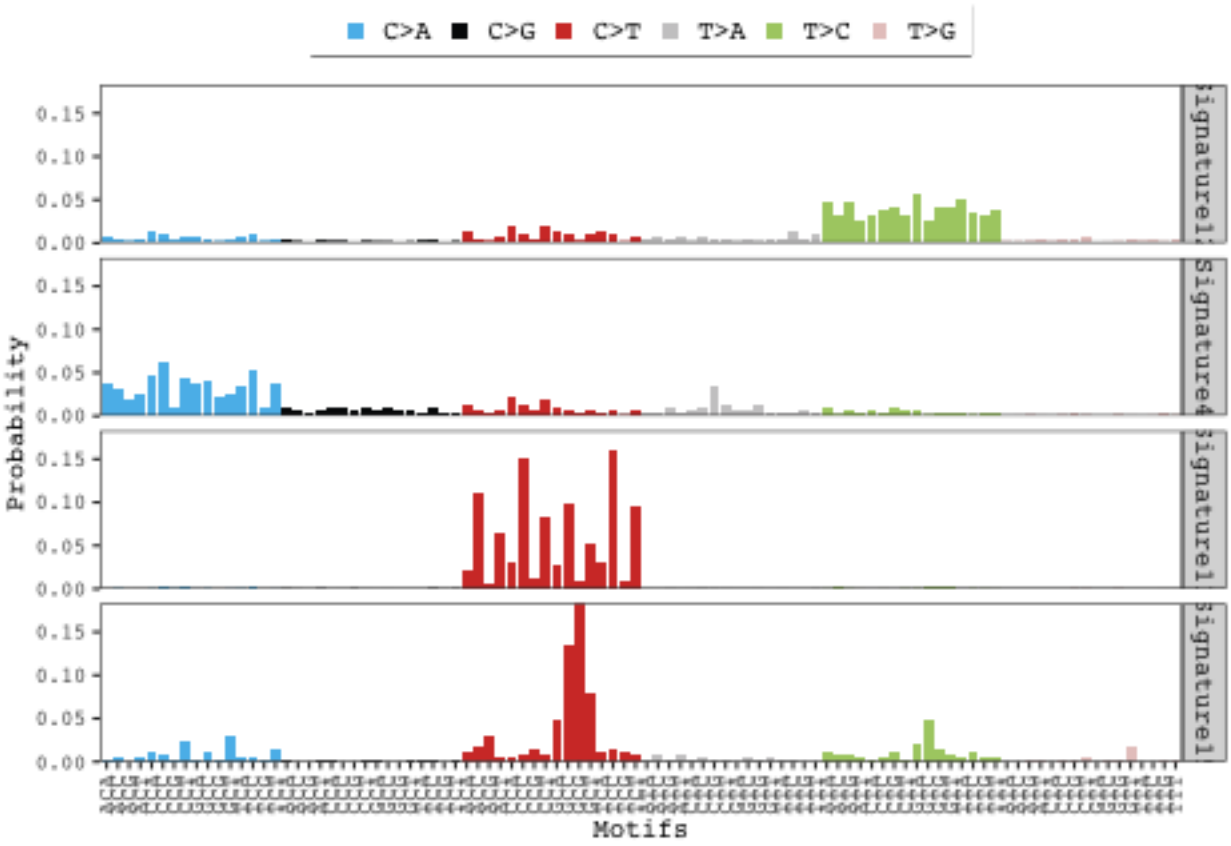
Predicting exposures using existing signatures

```
#List which signatures correspond to subtypes including "lung"
cosmic_v2_subtype_map("lung")
```

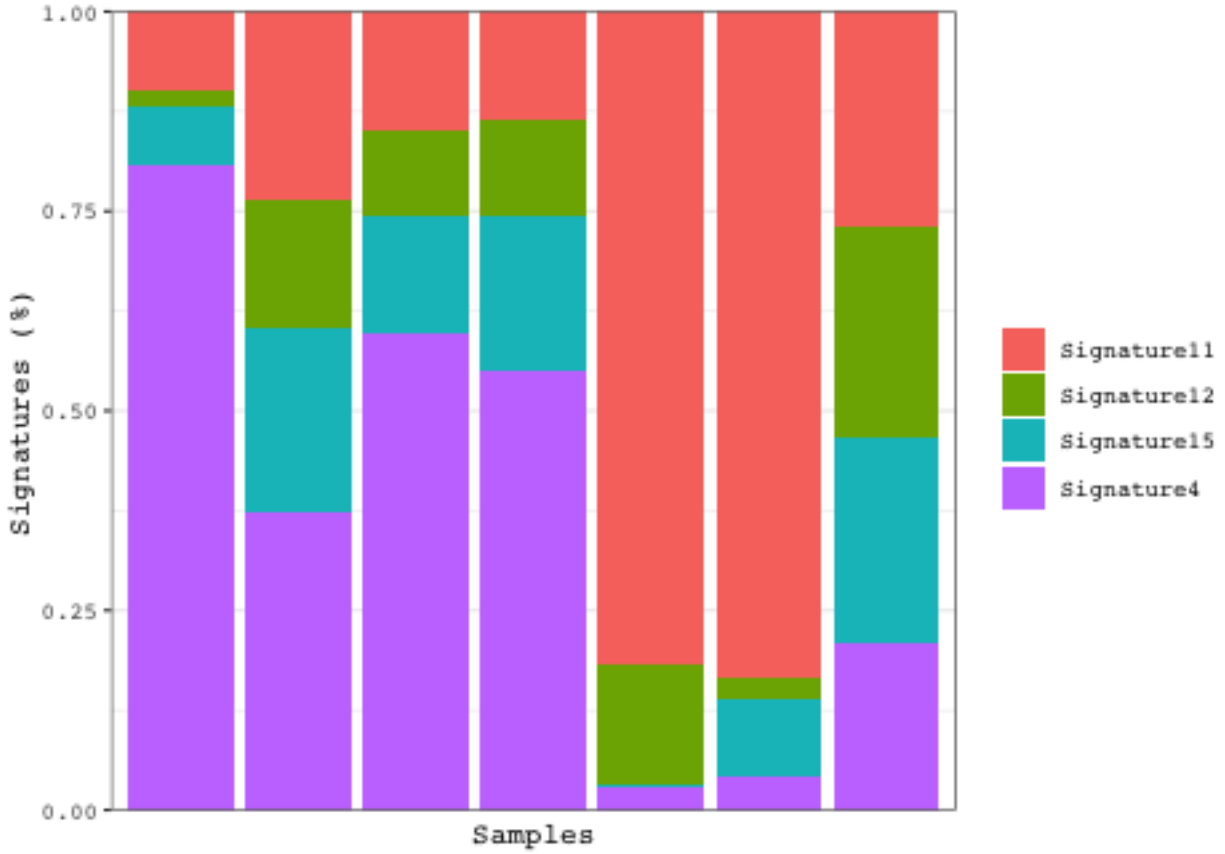
```
## [1] "lung adeno"
## [1]  1  2  4  5  6 13 17
## [1] "lung  small cell"
## [1]  1  4  5 15
## [1] "lung  squamous"
## [1]  1  2  4  5 13
```

```
#Calculate posterior based on COSMIC signatures 4, 11, 12, 15
cosmic_post = predict_exposure(bagel = bagel, "SBS96", signature_res =
                               cosmic_v2_sigs, signatures_to_use =
                               c(12, 4, 11, 15), algorithm = "lda")

#Calculate posterior based on our novel signatures
our_sigs_post = predict_exposure(bagel = bagel, "SBS96", signature_res =
                                 result, algorithm = "lda")
```

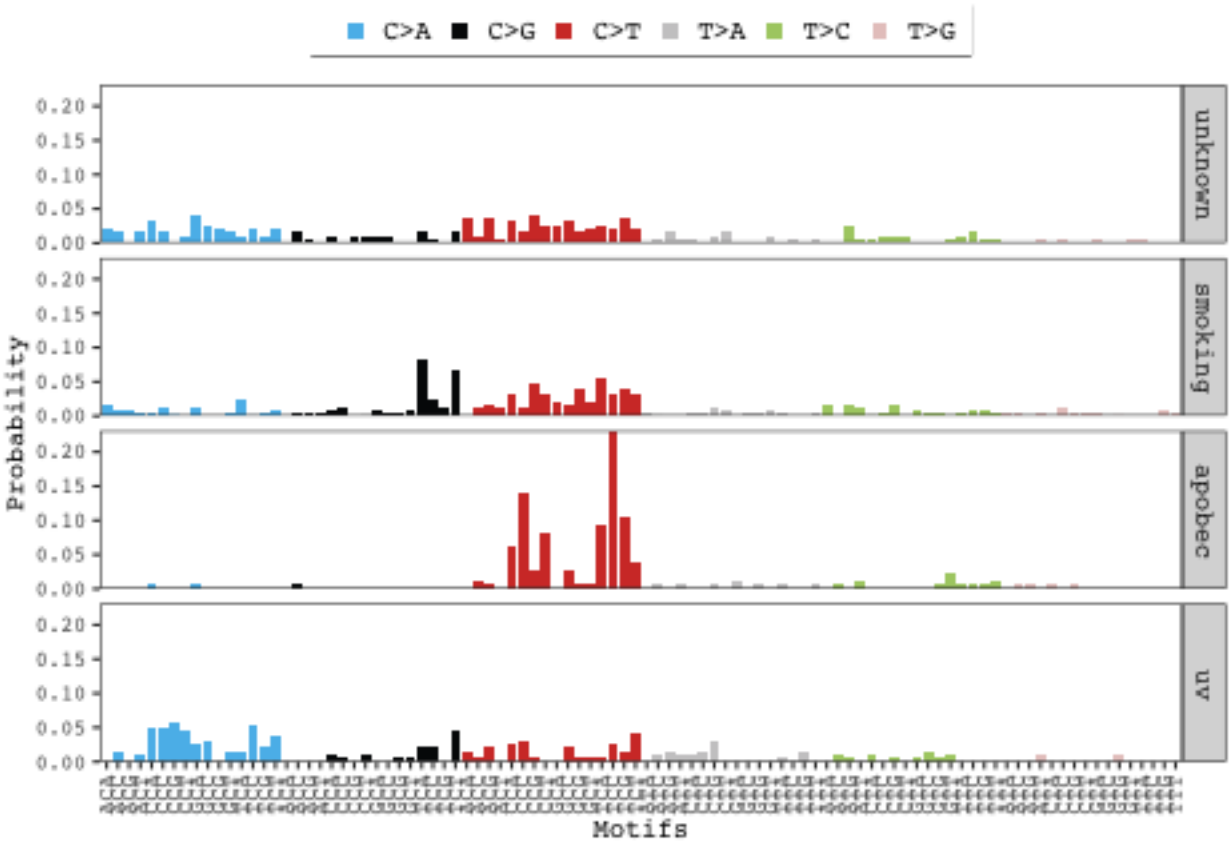
```
#Plot results from posterior calculation
plot_signatures(cosmic_post)
```



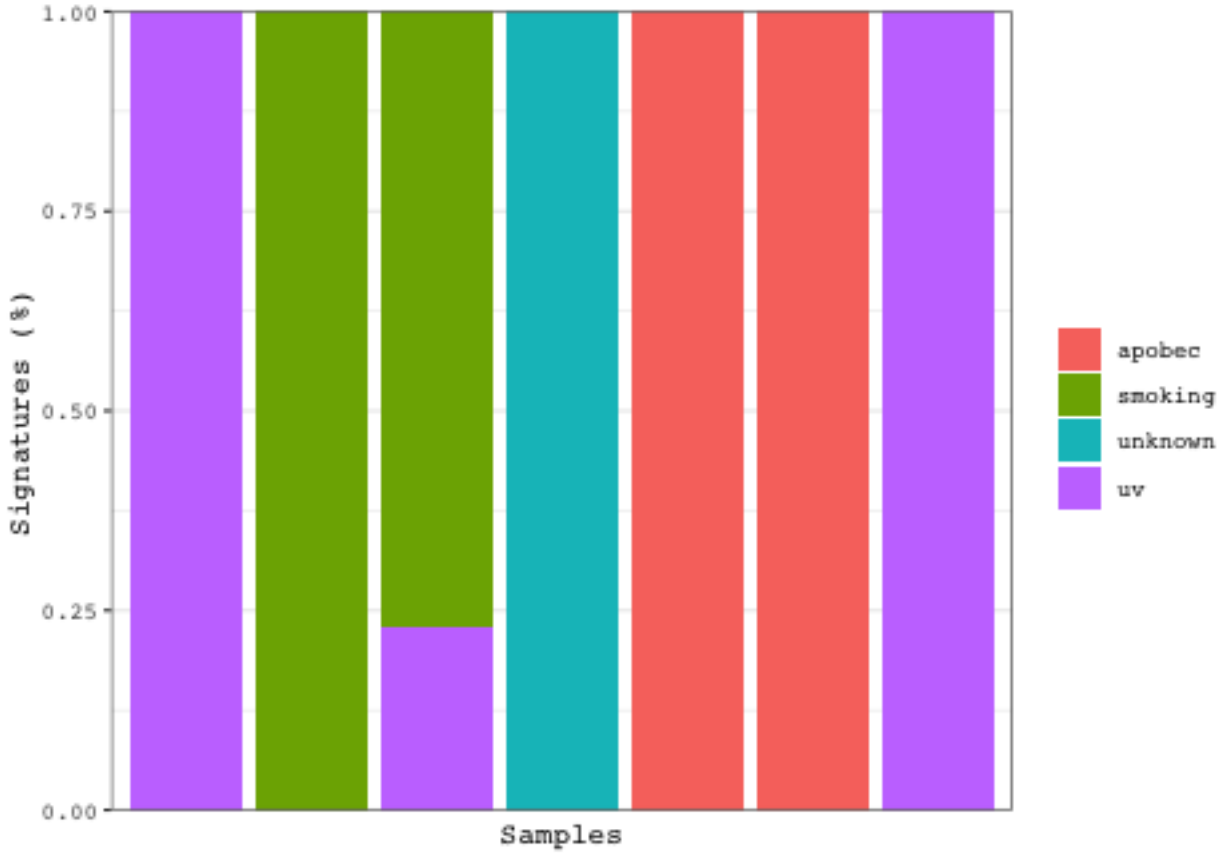

```
plot_exposures(cosmic_post, proportional = TRUE)
```



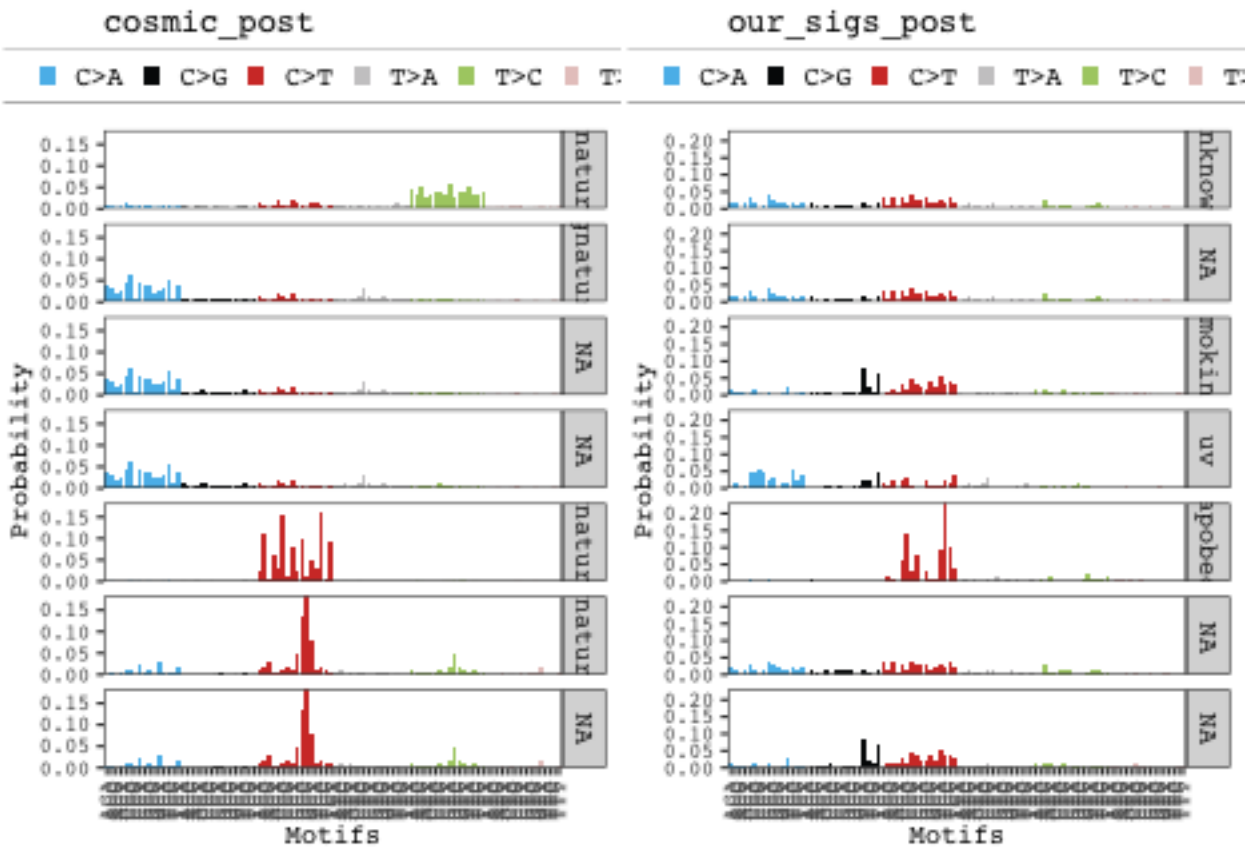

```
plot_signatures(our_sigs_post)
```




```
plot_exposures(our_sigs_post, proportional = TRUE)
```




```
#Compare posterior results to each other
compare_results(result = cosmic_post, other_result = our_sigs_post,
                threshold = 0.60)
```




```
##          cor xindex yindex        xcol    ycol
## 1 0.6346334      1      1 Signature12 unknown
## 2 0.7758602      2      1  Signature4 unknown
## 3 0.6368810      2      2  Signature4 smoking
## 4 0.7738200      2      4  Signature4      uv
## 5 0.7438254      3      3 Signature11  apobec
## 6 0.6748794      4      1 Signature15 unknown
## 7 0.6332559      4      2 Signature15 smoking
```

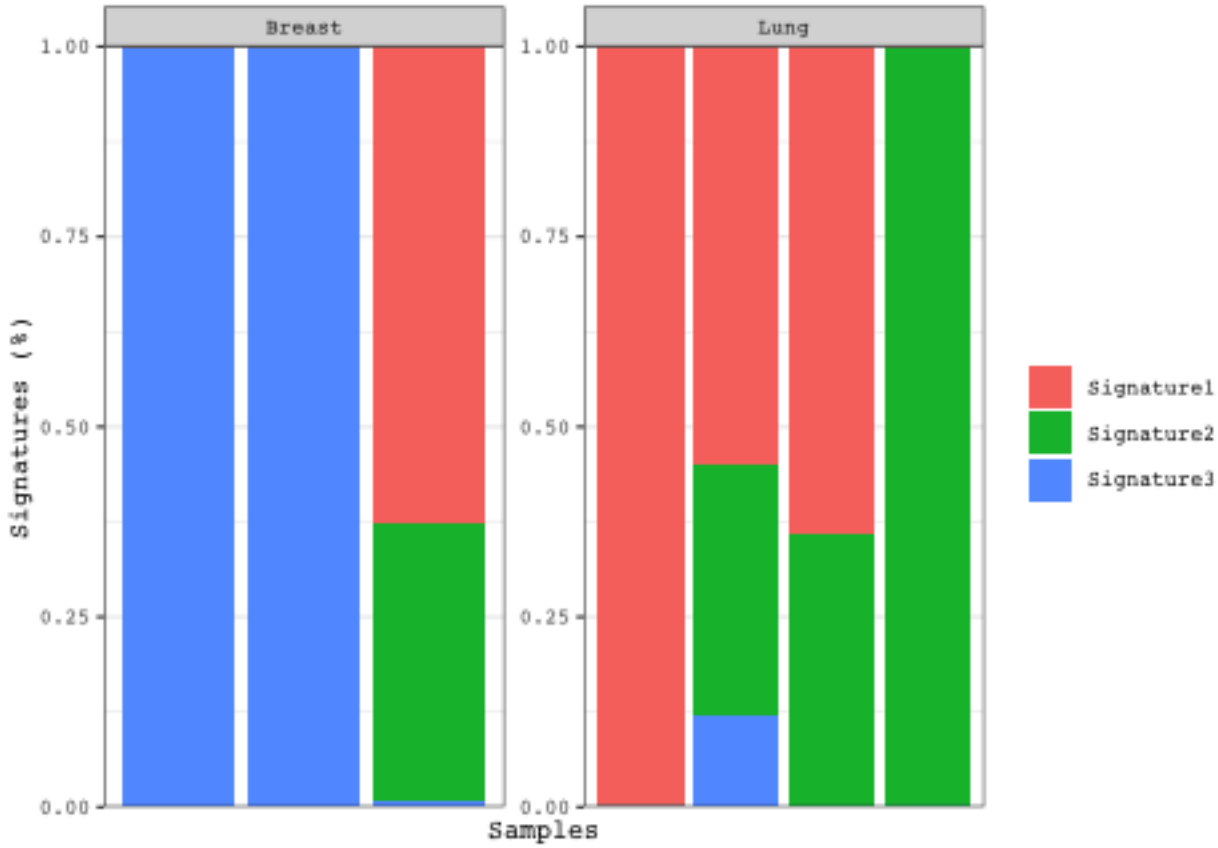
Use of sample annotations for advanced sample comparisons

Adding annotations

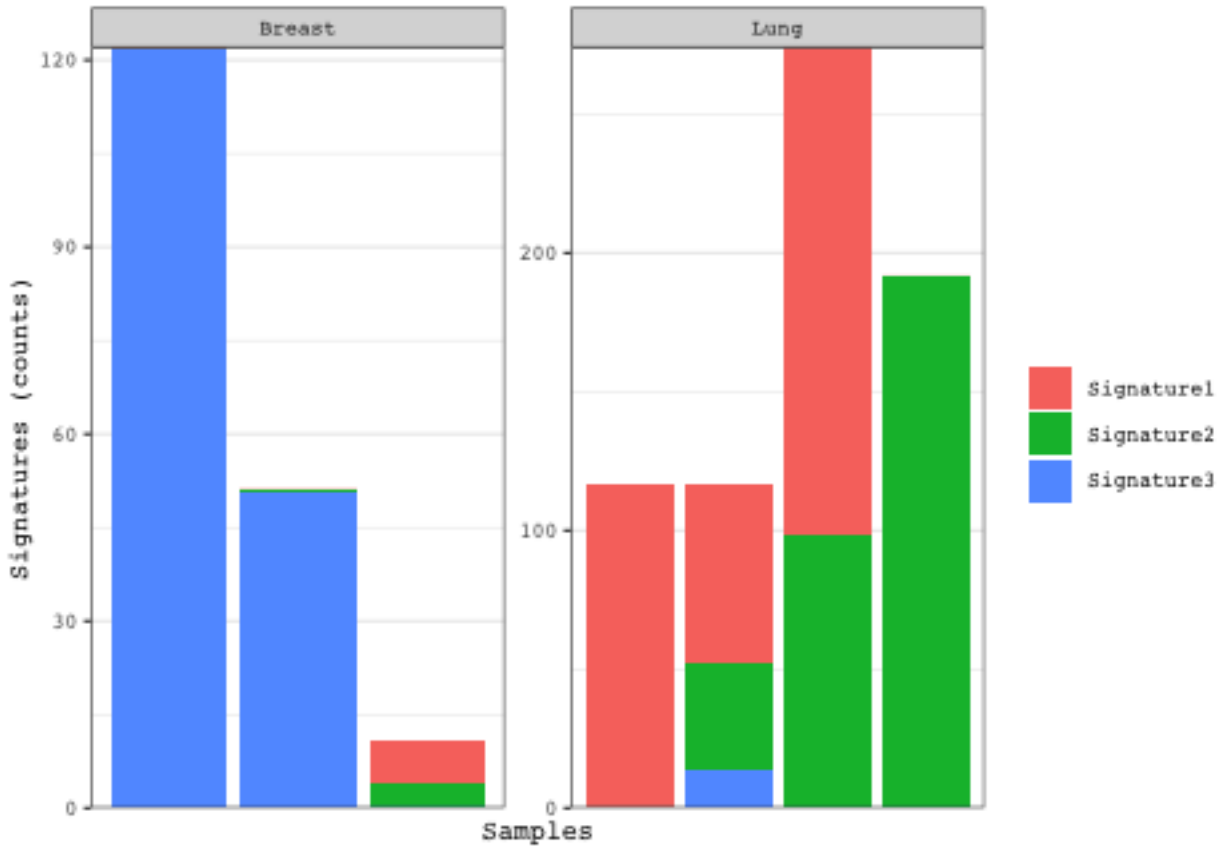
```
sample_annotations <- read.table(system.file("testdata",
                                             "sample_annotations.txt",
                                             package = "BAGEL"), sep = "\t",
                                 header=TRUE)
init_sample_annotations(bagel)
add_sample_annotations(bay = bagel, annotations = sample_annotations,
                       sample_column = "Sample_Names",
                       columns_to_add = "Tumor_Subtypes")
```

Standard discovery using BAGEL with sample annotations

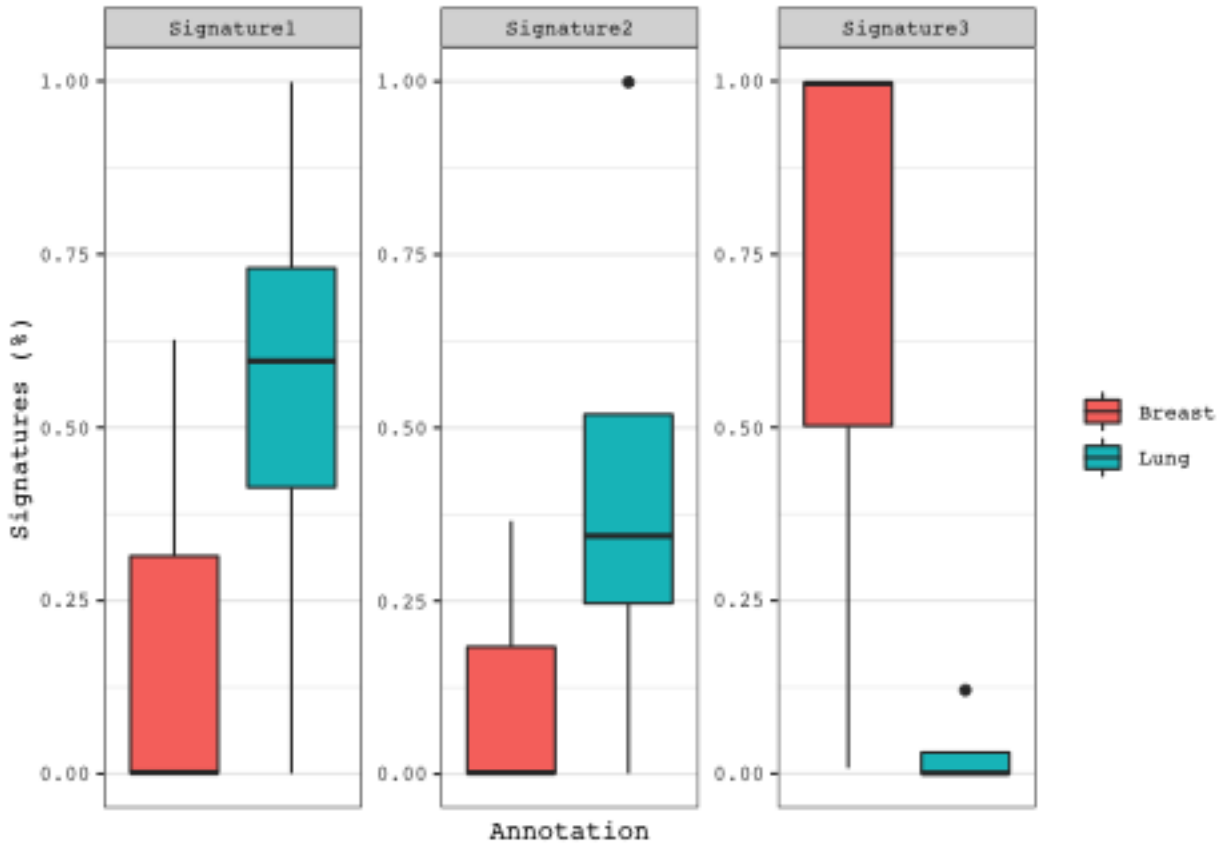
```
#Add tumor type annotations to our samples
res <- discover_signatures(bagel, table_name = "SBS96", num_signatures = 3,
                           seed = 1)

plot_exposures_by_annotation(res, annotation = "Tumor_Subtypes")
```




```
plot_exposures_by_annotation(res, annotation = "Tumor_Subtypes",
                             proportional = FALSE)
```




```
plot_exposures_by_annotation(res, annotation = "Tumor_Subtypes",
                             by_group = FALSE)
```



Other functions

```
#Display what samples were added
get_sample_names(bagel)
```

```
## [1] TCGA-94-7557-01A-11D-2122-08 TCGA-56-7582-01A-11D-2042-08
## [3] TCGA-77-7335-01A-11D-2042-08 TCGA-97-7938-01A-11D-2167-08
## [5] TCGA-EE-A3J5-06A-11D-A20D-08 TCGA-ER-A197-06A-32D-A197-08
## [7] TCGA-ER-A19O-06A-11D-A197-08
## 7 Levels: TCGA-56-7582-01A-11D-2042-08 ... TCGA-ER-A19O-06A-11D-A197-08
```

Session Information

```
sessionInfo()
```

```
## R version 4.0.2 (2020-06-22)
## Platform: x86_64-apple-darwin17.7.0 (64-bit)
## Running under: macOS High Sierra 10.13.6
##
## Matrix products: default
## BLAS/LAPACK: /usr/local/Cellar/openblas/0.3.10_1/lib/libopenblasp-r0.3.10.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] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] BiocStyle_2.16.0    BAGEL_0.99.0        NMF_0.23.0
## [4] cluster_2.1.0       rngtools_1.5        pkgmaker_0.31.1
## [7] registry_0.5-1      Biobase_2.48.0      BiocGenerics_0.34.0
##
## loaded via a namespace (and not attached):
##   [1] backports_1.1.10
##   [2] BiocFileCache_1.12.1
##   [3] plyr_1.8.6
##   [4] splines_4.0.2
##   [5] BiocParallel_1.22.0
##   [6] topicmodels_0.2-11
##   [7] usethis_1.6.3
##   [8] GenomeInfoDb_1.24.2
##   [9] ggplot2_3.3.2
##  [10] gridBase_0.4-7
##  [11] digest_0.6.25
##  [12] foreach_1.5.0
##  [13] htmltools_0.5.0
##  [14] fansi_0.4.1
##  [15] magrittr_1.5
##  [16] memoise_1.1.0
##  [17] BSgenome_1.56.0
##  [18] tm_0.7-7
##  [19] doParallel_1.0.15
##  [20] remotes_2.2.0
##  [21] Biostrings_2.56.0
##  [22] matrixStats_0.56.0
##  [23] BSgenome.Hsapiens.UCSC.hg38_1.4.3
##  [24] askpass_1.1
##  [25] prettyunits_1.1.1
##  [26] colorspace_1.4-1
##  [27] blob_1.2.1
##  [28] rappdirs_0.3.1
##  [29] xfun_0.17
##  [30] dplyr_1.0.2
##  [31] callr_3.4.4
##  [32] crayon_1.3.4
##  [33] RCurl_1.98-1.2
##  [34] TxDb.Hsapiens.UCSC.hg19.knownGene_3.2.2
##  [35] roxygen2_7.1.1
##  [36] survival_3.2-3
##  [37] VariantAnnotation_1.34.0
##  [38] iterators_1.0.12
##  [39] glue_1.4.2
##  [40] gtable_0.3.0
##  [41] zlibbioc_1.34.0
##  [42] XVector_0.28.0
##  [43] DelayedArray_0.14.1
```

```
##  [44] TxDb.Hsapiens.UCSC.hg38.knownGene_3.10.0
##  [45] pkgbuild_1.1.0
##  [46] scales_1.1.1
##  [47] DBI_1.1.0
##  [48] bibtex_0.4.2.3
##  [49] Rcpp_1.0.5
##  [50] xtable_1.8-4
##  [51] progress_1.2.2
##  [52] bit_4.0.4
##  [53] stats4_4.0.2
##  [54] httr_1.4.2
##  [55] RColorBrewer_1.1-2
##  [56] ellipsis_0.3.1
##  [57] modeltools_0.2-23
##  [58] farver_2.0.3
##  [59] pkgconfig_2.0.3
##  [60] XML_3.99-0.5
##  [61] uwot_0.1.8
##  [62] dbplyr_1.4.4
##  [63] labeling_0.3
##  [64] tidyselect_1.1.0
##  [65] rlang_0.4.7
##  [66] reshape2_1.4.4
##  [67] AnnotationDbi_1.50.3
##  [68] munsell_0.5.0
##  [69] tools_4.0.2
##  [70] cli_2.0.2
##  [71] generics_0.0.2
##  [72] RSQLite_2.2.0
##  [73] devtools_2.3.2
##  [74] evaluate_0.14
##  [75] stringr_1.4.0
##  [76] yaml_2.2.1
##  [77] processx_3.4.4
##  [78] knitr_1.30
##  [79] bit64_4.0.5
##  [80] fs_1.5.0
##  [81] purrr_0.3.4
##  [82] slam_0.1-47
##  [83] xml2_1.3.2
##  [84] biomaRt_2.44.1
##  [85] compiler_4.0.2
##  [86] rstudioapi_0.11
##  [87] curl_4.3
##  [88] testthat_2.3.2
##  [89] maftools_2.4.12
##  [90] tibble_3.0.3
##  [91] stringi_1.5.3
##  [92] ps_1.3.4
##  [93] GenomicFeatures_1.40.1
##  [94] desc_1.2.0
##  [95] lattice_0.20-41
##  [96] Matrix_1.2-18
##  [97] vctrs_0.3.4
```

```
##  [98] pillar_1.4.6
##  [99] lifecycle_0.2.0
## [100] BiocManager_1.30.10
## [101] combinat_0.0-8
## [102] data.table_1.13.0
## [103] cowplot_1.1.0
## [104] bitops_1.0-6
## [105] rtracklayer_1.48.0
## [106] GenomicRanges_1.40.0
## [107] R6_2.4.1
## [108] gridExtra_2.3
## [109] IRanges_2.22.2
## [110] sessioninfo_1.1.1
## [111] codetools_0.2-16
## [112] MCMCprecision_0.4.0
## [113] gtools_3.8.2
## [114] assertthat_0.2.1
## [115] pkgload_1.1.0
## [116] SummarizedExperiment_1.18.2
## [117] openssl_1.4.3
## [118] rprojroot_1.3-2
## [119] withr_2.3.0
## [120] GenomicAlignments_1.24.0
## [121] Rsamtools_2.4.0
## [122] S4Vectors_0.26.1
## [123] GenomeInfoDbData_1.2.3
## [124] hms_0.5.3
## [125] grid_4.0.2
## [126] tidyr_1.1.2
## [127] rmarkdown_2.3
## [128] NLP_0.2-0
```