

# Package ‘supersigs’

January 24, 2026

**Title** Supervised mutational signatures

**Version** 1.19.0

**Date** 2021-12-02

**Depends** R (>= 4.1)

**Imports** assertthat, caret, dplyr, tidyr, rsample, methods, rlang,  
utils, Biostrings, stats, SummarizedExperiment

**Suggests** BSgenome.Hsapiens.UCSC.hg19, BSgenome.Hsapiens.UCSC.hg38,  
knitr, rmarkdown, ggplot2, testthat, VariantAnnotation

**Description** Generate SuperSigs (supervised mutational signatures) from single nucleotide variants in the cancer genome. Functions included in the package allow the user to learn supervised mutational signatures from their data and apply them to new data. The methodology is based on the one described in Afsari (2021, ELife).

**biocViews** FeatureExtraction, Classification, Regression, Sequencing,  
WholeGenome, SomaticMutation

**BugReports** <https://github.com/TomasettiLab/supersigs/issues>

**URL** <https://tomasettelab.github.io/supersigs/>

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**LazyDataCompression** gzip

**RoxygenNote** 7.1.1

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**git\_url** <https://git.bioconductor.org/packages/supersigs>

**git\_branch** devel

**git\_last\_commit** 281d3c5

**git\_last\_commit\_date** 2025-10-29

**Repository** Bioconductor 3.23

**Date/Publication** 2026-01-23

**Author** Albert Kuo [aut, cre] (ORCID: <<https://orcid.org/0000-0001-5155-0748>>),  
Yifan Zhang [aut],  
Bahman Afsari [aut],  
Cristian Tomasetti [aut]

**Maintainer** Albert Kuo <[albertkuo@jhu.edu](mailto:albertkuo@jhu.edu)>

Contents

|                              |          |
|------------------------------|----------|
| example_dt . . . . .         | 2        |
| get_signature . . . . .      | 3        |
| make_matrix . . . . .        | 3        |
| partial_signature . . . . .  | 4        |
| predict_signature . . . . .  | 5        |
| process_vcf . . . . .        | 5        |
| simplify_signature . . . . . | 6        |
| SuperSig-class . . . . .     | 7        |
| supersig_ls . . . . .        | 7        |
| <b>Index</b>                 | <b>8</b> |

---

|            |                                     |
|------------|-------------------------------------|
| example_dt | <i>Example dataset of mutations</i> |
|------------|-------------------------------------|

---

Description

A dataset containing a list of mutations and other necessary attributes

Usage

example\_dt

Format

A data frame with 10 rows and 5 columns:

- sample\_id** ID of the patient
- age** age of the patient
- chromosome** chromosomal position of the mutation
- position** position of the mutation
- ref** original nucleotide
- alt** mutated nucleotide

---

|               |                                      |
|---------------|--------------------------------------|
| get_signature | <i>Function to obtain a SuperSig</i> |
|---------------|--------------------------------------|

---

**Description**

Generate a tissue-specific SuperSig for a given dataset of mutations and exposure factor. Returns the SuperSig and a classification model trained with the SuperSig.

**Usage**

```
get_signature(data, factor, wgs = FALSE)
```

**Arguments**

|        |                                                                                                                                                    |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| data   | a data frame of mutations containing columns for sample_id, age, IndVar, and the 96 trinucleotide mutations (see vignette for details)             |
| factor | the factor/exposure (e.g. "age", "smoking"). If the factor = "age", the SuperSig is computed using counts. Otherwise, rates (counts/age) are used. |
| wgs    | logical value indicating whether sequencing data is whole-genome (wgs = TRUE) or whole-exome (wgs = FALSE)                                         |

**Value**

get\_signature returns an object of class SuperSig

**Examples**

```
head(example_dt) # use example data from package
input_dt <- make_matrix(example_dt) # convert to correct format
input_dt$IndVar <- c(1, 1, 1, 0, 0) # add IndVar column
get_signature(data = input_dt, factor = "Age") # get SuperSig
```

---

|             |                                                             |
|-------------|-------------------------------------------------------------|
| make_matrix | <i>Function to transform mutations into "matrix" format</i> |
|-------------|-------------------------------------------------------------|

---

**Description**

Transform a data frame of mutations in long format into a data frame of trinucleotide mutations with flanking bases in a wide matrix format.

**Usage**

```
make_matrix(data, genome = "hg19")
```

**Arguments**

`data` a data frame of mutations in VCF format (see vignette for details)  
`genome` the reference genome used ("hg19" or "hg38")

**Value**

`make_matrix` returns a data frame of mutations, one row per sample

**Examples**

```
head(example_dt) # use example data from package
input_dt <- make_matrix(example_dt) # convert to correct format
head(input_dt)
```

---

|                   |                                                          |
|-------------------|----------------------------------------------------------|
| partial_signature | <i>Function to remove the contribution of a SuperSig</i> |
|-------------------|----------------------------------------------------------|

---

**Description**

Remove the contribution of a SuperSig from the data and return the data.

**Usage**

```
partial_signature(data, object)
```

**Arguments**

`data` a data frame of mutations containing columns for `sample_id`, `age`, `IndVar`, and the 96 trinucleotide mutations (see vignette for details)  
`object` an object of class `SuperSig`

**Value**

`predict_signature` returns the original data frame with the contribution of a supervised signature removed

**Examples**

```
head(example_dt) # use example data from package
input_dt <- make_matrix(example_dt) # convert to correct format
input_dt$IndVar <- c(1, 1, 1, 0, 0) # add IndVar column
supersig <- get_signature(data = input_dt, factor = "Age") # get SuperSig
partial_signature(data = input_dt, object = supersig)
```

---

|                   |                                                  |
|-------------------|--------------------------------------------------|
| predict_signature | <i>Function to predict using SuperSig object</i> |
|-------------------|--------------------------------------------------|

---

**Description**

Using a generated SuperSig, predict on a new dataset and return predicted probabilities for each observation.

**Usage**

```
predict_signature(object, newdata, factor)
```

**Arguments**

|         |                                                                                                                                        |
|---------|----------------------------------------------------------------------------------------------------------------------------------------|
| object  | an object of class SuperSig                                                                                                            |
| newdata | a data frame of mutations containing columns for sample_id, age, IndVar, and the 96 trinucleotide mutations (see vignette for details) |
| factor  | the factor/exposure (e.g. "age", "smoking")                                                                                            |

**Value**

predict\_signature returns the original data frame with additional columns for the feature counts and classification score

**Examples**

```
head(example_dt) # use example data from package
input_dt <- make_matrix(example_dt) # convert to correct format
input_dt$IndVar <- c(1, 1, 1, 0, 0) # add IndVar column
out <- get_signature(data = input_dt, factor = "Age") # get SuperSig
newdata <- predict_signature(out, newdata = input_dt, factor = "age")
suppressPackageStartupMessages({library(dplyr)})
head(newdata %>% select(score))
```

---

|             |                                                              |
|-------------|--------------------------------------------------------------|
| process_vcf | <i>Function to transform VCF object into "matrix" format</i> |
|-------------|--------------------------------------------------------------|

---

**Description**

Transform a VCF object into a data frame of trinucleotide mutations with flanking bases in a wide matrix format. The function assumes that the VCF object contains only one sample and that each row in rowRanges represents an observed mutation in the sample.

**Usage**

```
process_vcf(vcf)
```

**Arguments**

vcf                      a VCF object (from VariantAnnotation package)

**Value**

process\_vcf returns a data frame of mutations, one row per mutation

**Examples**

```
# Use example vcf from VariantAnnotation
suppressPackageStartupMessages({library(VariantAnnotation)})
fl <- system.file("extdata", "chr22.vcf.gz", package="VariantAnnotation")
vcf <- VariantAnnotation::readVcf(fl, "hg19")

# Subset to first sample
vcf <- vcf[, 1]
# Subset to row positions with homozygous or heterozygous alt
positions <- geno(vcf)$GT != "0|0"
vcf <- vcf[positions[, 1],]
colData(vcf)$age <- 50                      # Add patient age to colData (optional)

# Run function
dt <- process_vcf(vcf)
head(dt)
```

---

|                    |                                                                                                           |
|--------------------|-----------------------------------------------------------------------------------------------------------|
| simplify_signature | <i>Function to simplify signature representation into interpretable labels for visualization purposes</i> |
|--------------------|-----------------------------------------------------------------------------------------------------------|

---

**Description**

Take a signature representation from SuperSig and group trinucleotides within each feature into interpretable labels, with optional IUPAC labeling from IUPAC\_CODE\_MAP in the Biostrings package

**Usage**

```
simplify_signature(object, iupac)
```

**Arguments**

object                      an object of class SuperSig

iupac                      logical value indicating whether to use IUPAC labels (iupac = TRUE) or not (iupac = FALSE)

Value

simplify\_signature returns a vector of simplified features and their difference in mean mean rates between exposed and unexposed (or average rate if the factor is "age")

Examples

```
head(example_dt) # use example data from package
input_dt <- make_matrix(example_dt) # convert to correct format
input_dt$IndVar <- c(1, 1, 1, 0, 0) # add IndVar column
supersig <- get_signature(data = input_dt, factor = "Smoking")
simplify_signature(object = supersig, iupac = FALSE)
simplify_signature(object = supersig, iupac = TRUE)
```

---

|                |                                 |
|----------------|---------------------------------|
| SuperSig-class | <i>An S4 class for SuperSig</i> |
|----------------|---------------------------------|

---

Description

An S4 class for SuperSig

Slots

Signature data frame of features and their difference in mean rates between exposed and unexposed (or the average rate if the factor is "age")

Features list of features that comprise the signature and their representation in terms of the fundamental (trinucleotide) mutations

AUC length-one numeric vector of the apparent AUC (i.e. not cross-validated)

Model list of a glm class for trained logistic regression model

---

|             |                                    |
|-------------|------------------------------------|
| supersig_ls | <i>Trained SuperSigs from TCGA</i> |
|-------------|------------------------------------|

---

Description

A list containing 67 SuperSigs

Usage

```
supersig_ls
```

Format

A named list with 67 elements, each of which is a ‘SuperSig’

# Index

## \* **datasets**

- example\_dt, [2](#)
- supersig\_ls, [7](#)

example\_dt, [2](#)

get\_signature, [3](#)

make\_matrix, [3](#)

partial\_signature, [4](#)

predict\_signature, [5](#)

process\_vcf, [5](#)

simplify\_signature, [6](#)

SuperSig (SuperSig-class), [7](#)

SuperSig-class, [7](#)

supersig\_ls, [7](#)