

# Package ‘bugsigdbr’

May 14, 2025

**Version** 1.14.1

**Title** R-side access to published microbial signatures from BugSigDB

**Description** The bugsigdbr package implements convenient access to bugsigdb.org from within R/Bioconductor. The goal of the package is to facilitate import of BugSigDB data into R/Bioconductor, provide utilities for extracting microbe signatures, and enable export of the extracted signatures to plain text files in standard file formats such as GMT.

**URL** <https://github.com/waldronlab/bugsigdbr>

**BugReports** <https://github.com/waldronlab/bugsigdbr/issues>

**Depends** R (>= 4.1)

**Imports** BiocFileCache, methods, vroom, utils

**Suggests** BiocStyle, knitr, ontologyIndex, rmarkdown, testthat (>= 3.0.0)

**License** GPL-3

**VignetteBuilder** knitr

**biocViews** DataImport, GeneSetEnrichment, Metagenomics, Microbiome

**Encoding** UTF-8

**RoxygenNote** 7.3.2

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

**git\_branch** RELEASE\_3\_21

**git\_last\_commit** 526476f

**git\_last\_commit\_date** 2025-04-23

**Repository** Bioconductor 3.21

**Date/Publication** 2025-05-14

**Author** Ludwig Geistlinger [aut, cre],  
Jennifer Wokaty [aut],  
Levi Waldron [aut]

**Maintainer** Ludwig Geistlinger <ludwig.geistlinger@gmail.com>

## Contents

browseSignature . . . . .	2
browseTaxon . . . . .	3
extractTaxLevel . . . . .	3
getMetaSignatures . . . . .	4
getOntology . . . . .	6
getSignatures . . . . .	7
importBugSigDB . . . . .	8
restrictTaxLevel . . . . .	9
subsetByOntology . . . . .	10
writeGMT . . . . .	11
<b>Index</b>	<b>13</b>

---

browseSignature	<i>Displaying BugSigDB signatures pages in a web browser</i>
-----------------	--

---

### Description

Functionality for programmatically displaying microbe signatures on BugSigDB signature pages.

### Usage

```
browseSignature(sname)
```

### Arguments

sname            character. Signature name. Expected to start with a prefix of the form "bsdb:<X>/<Y>/<Z>\_" encoding the corresponding BugSigDB signature ID.

### Value

The URL of the selected BugSigDB signature page. If interactive, opens the URL in the default web browser.

### References

BugSigDB: <https://bugsigdb.org>

### Examples

```
sname <- "bsdb:215/1/1_eczema:infant-with-eczema_vs_healthy-control_UP"
browseSignature(sname)
```

---

`browseTaxon`*Displaying BugSigDB taxon pages in a web browser*

---

**Description**

Functionality for programmatically displaying BugSigDB taxon pages.

**Usage**

```
browseTaxon(tax.id)
```

**Arguments**

`tax.id` character. NCBI taxonomy ID.

**Value**

The URL of the selected BugSigDB taxon page. If interactive, opens the URL in the default web browser.

**References**

BugSigDB: <https://bugsigdb.org>

**Examples**

```
# BugSigDB taxon page for Escherichia coli
browseTaxon("562")
```

---

`extractTaxLevel`*Extract specific taxonomic levels from a microbe signature*

---

**Description**

Functionality for extracting specific taxonomic levels (such as genus and species) from a microbe signature containing taxonomic clades in MetaPhlAn format.

**Usage**

```
extractTaxLevel(
  sig,
  tax.id.type = c("metaphlan", "taxname"),
  tax.level = "mixed",
  exact.tax.level = TRUE
)
```

**Arguments**

sig	character. Microbe signature containing taxonomic clades in MetaPhlAn format.
tax.id.type	Character. Taxonomic ID type of the returned microbe sets. Either "metaphlan" (default) or "taxname".
tax.level	character. Either "mixed" or any subset of c("kingdom", "phylum", "class", "order", "family", "genus", "species", "strain"). This full vector is equivalent to "mixed".
exact.tax.level	logical. Should only the exact taxonomic level specified by tax.level be returned? Defaults to TRUE. If FALSE, a more general tax.level is extracted for microbes given at a more specific taxonomic level.

**Value**

a character vector storing taxonomic clades restricted to chosen taxonomic level(s).

**References**

BugSigDB: <https://bugsigdb.org>

**Examples**

```
ord <- "k__Bacteria|p__Firmicutes|c__Bacilli|o__Lactobacillales"
sig <- c("f__Lactobacillaceae|g__Lactobacillus",
        "f__Aerococcaceae|g__Abiotrophia|s__Abiotrophia defectiva",
        "f__Lactobacillaceae|g__Limosilactobacillus|s__Limosilactobacillus mucosae")
sig <- paste(ord, sig, sep = "|")
sig <- extractTaxLevel(sig, tax.level = "genus")
sig <- extractTaxLevel(sig, tax.level = "genus", exact.tax.level = FALSE)
sig <- extractTaxLevel(sig,
                       tax.id.type = "taxname",
                       tax.level = "genus",
                       exact.tax.level = FALSE)
```

---

getMetaSignatures

*Obtain meta-signatures for a column of interest*

---

**Description**

Functionality for obtaining meta-signatures for a column of interest

## Usage

```
getMetaSignatures(  
  df,  
  column,  
  direction = c("BOTH", "UP", "DOWN"),  
  min.studies = 2,  
  min.taxa = 5,  
  comb.fun = sum,  
  ...  
)
```

## Arguments

df	data.frame storing BugSigDB data. Typically obtained via <a href="#">importBugSigDB</a> .
column	character. Column of interest. Need to be a valid column name of df.
direction	character. Indicates direction of abundance change for signatures to be included in the computation of meta-signatures. Use "UP" to restrict computation to signatures with increased abundance in the exposed group. Use "DOWN" to restrict to signatures with decreased abundance in the exposed group. Defaults to "BOTH" which will not filter signatures by direction of abundance change.
min.studies	integer. Minimum number of studies for a category in column to be included. Defaults to 2, which will then only compute meta-signatures for categories investigated by at least two studies.
min.taxa	integer. Minimum size for meta-signatures. Defaults to 5, which will then only include meta-signatures containing at least 5 taxa.
comb.fun	function. Function for combining sample size of the exposed group and sample size of the unexposed group into an overall study sample size. Defaults to sum which will simply add sample sizes of exposed and unexposed group.
...	additional argument passed on to <a href="#">getSignatures</a> .

## Value

A list of meta-signatures, each meta-signature being a named numeric vector. Names are the taxa of the meta-signature, numeric values correspond to sample size weights associated with each taxon.

## See Also

[getSignatures](#)

## Examples

```
df <- importBugSigDB()  
  
# Body-site specific meta-signatures composed from signatures reported as both  
# increased or decreased across all conditions of study:  
bs.meta.sigs <- getMetaSignatures(df, column = "Body site")
```

```
# Condition-specific meta-signatures from fecal samples, increased
# in conditions of study. Use taxonomic names instead of the default NCBI IDs:
df.feces <- df[df$`Body site` == "Feces", ]
cond.meta.sigs <- getMetaSignatures(df.feces, column = "Condition",
                                   direction = "UP", tax.id.type = "taxname")

# Inspect the results
names(cond.meta.sigs)
cond.meta.sigs["Bipolar disorder"]
```

---

getOntology

*Obtain the EFO and UBERON ontology*

---

## Description

Lightweight wrapper around `ontologyIndex::get_ontology` to parse the Experimental Factor Ontology (EFO) or the Uber-anatomy ontology (UBERON) from OBO format into an R object.

## Usage

```
getOntology(onto = c("efo", "uberon"), cache = TRUE)
```

## Arguments

onto	character. Ontology to obtain. Should be either "efo" (default) to obtain the Experimental Factor Ontology (EFO) or "uberon" to obtain the Uber-anatomy ontology (UBERON).
cache	logical. Should a locally cached version used if available? Defaults to TRUE.

## Value

An object of class `ontology_index` as defined in the `ontologyIndex` package.

## References

EFO: <https://www.ebi.ac.uk/ols/ontologies/efo>

UBERON: <https://www.ebi.ac.uk/ols/ontologies/uberon>

## See Also

`get_ontology` from the `ontologyIndex` package.

## Examples

```
uberon <- getOntology("uberon")
```

---

`getSignatures`*Obtain microbe signatures from BugSigDB*

---

### Description

Functionality for obtaining microbe signatures from BugSigDB

### Usage

```
getSignatures(  
  df,  
  tax.id.type = c("ncbi", "metaphlan", "taxname"),  
  tax.level = "mixed",  
  exact.tax.level = TRUE,  
  min.size = 1  
)
```

### Arguments

<code>df</code>	data.frame storing BugSigDB data. Typically obtained via <code>importBugSigDB</code> .
<code>tax.id.type</code>	Character. Taxonomic ID type of the returned microbe sets. Either "ncbi" (default), "metaphlan", or "taxname".
<code>tax.level</code>	character. Either "mixed" or any subset of c("kingdom", "phylum", "class", "order", "family", "genus", "species", "strain"). This full vector is equivalent to "mixed".
<code>exact.tax.level</code>	logical. Should only the exact taxonomic level specified by <code>tax.level</code> be returned? Defaults to TRUE. If FALSE, a more general <code>tax.level</code> is extracted for microbes given at a more specific taxonomic level.
<code>min.size</code>	integer. Minimum signature size. Defaults to 1, which will filter out empty signature. Use <code>min.size = 0</code> to keep empty signatures.

### Value

a list of microbe signatures. Each signature is a character vector of taxonomic IDs depending on the chosen `tax.id.type`.

### References

BugSigDB: <https://bugsigdb.org>

### See Also

`importBugSigDB`

**Examples**

```
df <- importBugSigDB()
sigs <- getSignatures(df)
```

---

importBugSigDB	<i>Obtain published microbial signatures from bugsigdb.org</i>
----------------	--

---

**Description**

Obtain published microbial signatures from bugsigdb.org

**Usage**

```
importBugSigDB(version = "10.5281/zenodo.13997429", cache = TRUE)
```

**Arguments**

version	character. A Zenodo DOI, git commit hash, or "devel". Defaults to the most recent stable release on Zenodo, which includes complete and reviewed content from BugSigDB. See details.
cache	logical. Should a locally cached version used if available? Defaults to TRUE.

**Details**

There are three different options to obtain data from BugSigDB, as determined by the `version` argument.

- a Zenodo DOI: use this option if you would like to obtain one of the stable release versions of BugSigDB on Zenodo. These stable release versions of BugSigDB have been automatically checked and manually reviewed and provide for the highest data quality. Select this option if you would like to incorporate BugSigDB into analysis and published research. If not specified otherwise, the `importBugSigDB` function will obtain the most recent stable release from Zenodo by default.
- "devel": use this option to obtain the latest version ("bleeding edge") of BugSigDB from the BugSigDBExports GitHub repo (see references). Note that this will also include incomplete and not reviewed content, which should be filtered out prior to an analysis. Select this option if you are a curator that actively contributes to BugSigDB and would like to access data that you and other curators have recently contributed to BugSigDB and that has not been included in a stable release yet.
- a git commit hash: it might be occasionally of interest to obtain a specific snapshot of the BugSigDBExports GitHub repo, e.g. for the sake of debugging and troubleshooting. This can be done by providing the short 7-character git commit hash (SHA) or the full SHA of the export of choice. To provide the full SHA, go to the BugSigDBExports commits page (see references) and use the copy symbol to the left of the 7-character codes to copy the full SHA code of the export version you want to use.



**Value**

a `data.frame`.

**References**

BugSigDB: <https://bugsigdb.org>

Stable release: <https://doi.org/10.5281/zenodo.10627578>

Latest version (incl. not reviewed content): <https://github.com/waldronlab/BugSigDBExports>

Release v1.2.2: <https://zenodo.org/records/13997429>

Release v1.2.1: <https://zenodo.org/records/10627578>

Release v1.2.0: <https://zenodo.org/records/10407666>

Release v1.1.0: <https://zenodo.org/records/6468009>

Release v1.0.2: <https://zenodo.org/records/5904281>

Release v1.0.1: <https://zenodo.org/records/5819260>

Release v1.0.0: <https://zenodo.org/records/5606166>

BugSigDBExports commits page: <https://github.com/waldronlab/BugSigDBExports/commits/devel>

**Examples**

```
df <- importBugSigDB()
```

---

restrictTaxLevel	<i>Restrict microbe signatures to specific taxonomic levels</i>
------------------	---

---

**Description**

Functionality for restricting microbe signatures to specific taxonomic levels such as genus and species.

**Usage**

```
restrictTaxLevel(df, tax.level = "mixed", exact.tax.level = TRUE, min.size = 1)
```

**Arguments**

df	data.frame storing BugSigDB data. Typically obtained via <code>importBugSigDB</code> .
tax.level	character. Either "mixed" or any subset of <code>c("kingdom", "phylum", "class", "order", "family", "genus", "species", "strain")</code> . This full vector is equivalent to "mixed".

<code>exact.tax.level</code>	logical. Should only the exact taxonomic level specified by <code>tax.level</code> be returned? Defaults to TRUE. If FALSE, a more general <code>tax.level</code> is extracted for microbes given at a more specific taxonomic level.
<code>min.size</code>	integer. Minimum signature size. Defaults to 1, which will filter out empty signatures. Use <code>min.size = 0</code> to keep empty signatures.

**Value**

a data.frame with microbe signature columns restricted to chosen taxonomic level(s).

**References**

BugSigDB: <https://bugsigdb.org>

**See Also**

`importBugSigDB`

**Examples**

```
df <- importBugSigDB()
df <- restrictTaxLevel(df, tax.level = "genus")
```

---

subsetByOntology

*Ontology-based subsetting of BugSigDB signatures*

---

**Description**

This function facilitates ontology-based queries for experimental factors and body sites.

**Usage**

```
subsetByOntology(df, column = c("Body site", "Condition"), term, ontology)
```

**Arguments**

<code>df</code>	data.frame storing BugSigDB data. Typically obtained via <code>importBugSigDB</code> .
<code>column</code>	character. Column of <code>df</code> on which subsetting should be performed. Should be either "Body site" (default) or "Condition".
<code>term</code>	character. A valid term of ontology. Subsetting by this term then involves subsetting <code>column</code> to this term and all descendants of that term in the the chosen ontology and that are present in the chosen column of <code>df</code> .
<code>ontology</code>	an object of class <code>ontology_index</code> as defined in the <code>ontologyIndex</code> package. Typically obtained via <code>getOntology</code> .

**Value**

a data.frame with the chosen column restricted to descendants of the chosen term in the chosen ontology.

**References**

EFO: <https://www.ebi.ac.uk/ols/ontologies/efo>

UBERON: <https://www.ebi.ac.uk/ols/ontologies/uberont>

**See Also**

importBugSigDB, getOntology

**Examples**

```
# (1) Obtain BugSigDB data
df <- importBugSigDB()

# (2) Obtain ontology of interest as an R object
uberont <- getOntology("uberont")

# (3) High-level query on body site
sdf <- subsetByOntology(df,
                        column = "Body site",
                        term = "digestive system element",
                        ontology = uberont)
table(sdf[, "Body site"])
```

---

writeGMT

*Write microbe signatures to file in GMT format*

---

**Description**

Functionality for writing microbe signatures to file in GMT format.

**Usage**

```
writeGMT(sigs, gmt.file, ...)
```

**Arguments**

sigs	A list of microbe signatures (character vectors of taxonomic IDs).
gmt.file	character. Path to output file in GMT format.
...	Arguments passed on to cat()

**Value**

none, writes to file.

**References**

GMT file format: [http://www.broadinstitute.org/cancer/software/gsea/wiki/index.php/Data\\_formats](http://www.broadinstitute.org/cancer/software/gsea/wiki/index.php/Data_formats)

**Examples**

```
bsdb <- importBugSigDB()
sigs <- getSignatures(bsdb)
writeGMT(sigs, gmt.file = "signatures.gmt")
file.remove("signatures.gmt")
```

# Index

`browseSignature`, [2](#)  
`browseTaxon`, [3](#)

`data.frame`, [9](#)

`extractTaxLevel`, [3](#)

`getMetaSignatures`, [4](#)  
`getOntology`, [6](#), [10](#)  
`getSignatures`, [5](#), [7](#)

`importBugSigDB`, [5](#), [7](#), [8](#), [9](#), [10](#)

`restrictTaxLevel`, [9](#)

`subsetByOntology`, [10](#)

`writeGMT`, [11](#)