

# Package ‘MouseFM’

January 23, 2025

**Type** Package

**Title** In-silico methods for genetic finemapping in inbred mice

**Version** 1.17.0

**Description** This package provides methods for genetic finemapping in inbred mice by taking advantage of their very high homozygosity rate (>95%).

**Encoding** UTF-8

**LazyData** false

**BugReports** <https://github.com/matmu/MouseFM/issues>

**Depends** R (>= 4.0.0)

**License** GPL-3

**VignetteBuilder** knitr

**biocViews** Genetics, SNP, GeneTarget, VariantAnnotation, GenomicVariation, MultipleComparison, SystemsBiology, MathematicalBiology, PatternLogic, GenePrediction, BiomedicalInformatics, FunctionalGenomics

**Suggests** BiocStyle, testthat, knitr, rmarkdown

**Imports** httr, curl, GenomicRanges, dplyr, ggplot2, reshape2, scales, gtools, tidyr, data.table, jsonlite, rlist, GenomeInfoDb, methods, biomaRt, stats, IRanges

**RoxygenNote** 7.3.2

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

**git\_branch** devel

**git\_last\_commit** 16b409c

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.21

**Date/Publication** 2025-01-22

**Author** Matthias Munz [aut, cre] (ORCID:

<https://orcid.org/0000-0002-4728-3357>),

Inken Wohlers [aut] (ORCID: <https://orcid.org/0000-0003-4004-0464>),

Hauke Busch [aut] (ORCID: <https://orcid.org/0000-0003-4763-4521>)

**Maintainer** Matthias Munz <matthias.munz@gmx.de>

## Contents

annotate_consequences . . . . .	2
annotate_mouse_genes . . . . .	3
avail_chromosomes . . . . .	4
avail_consequences . . . . .	4
avail_strains . . . . .	5
backend_request . . . . .	5
comb . . . . .	6
df2GRanges . . . . .	6
df_split . . . . .	7
ensembl_rest_vep . . . . .	8
fetch . . . . .	8
finemap . . . . .	9
finemap_query . . . . .	10
getURL . . . . .	11
get_top . . . . .	12
GRanges2df . . . . .	12
prio . . . . .	13
reduction . . . . .	14
ref_genome . . . . .	15
setURL . . . . .	15
vis_reduction_factors . . . . .	16
<b>Index</b>	<b>17</b>

---

annotate\_consequences *Annotate with consequences*

---

### Description

Request variant consequences from Variant Effect Predictor (VEP) via Ensembl Rest Service. Not recommended for large queries.

### Usage

```
annotate_consequences(geno, species)
```

### Arguments

geno	Data frame or GenomicRanges::GRanges object including columns rsid, ref, alt.
species	Species name, e.g. mouse (GRCm38) or human (GRCh38).

### Value

Data frame.

### Examples

```
geno = finemap("chr1",
  start = 5000000, end = 6000000,
  strain1 = c("C57BL_6J"), strain2 = c("AKR_J", "A_J", "BALB_cJ")
)

df = annotate_consequences(geno[seq_len(10), ], "mouse")

geno.granges = finemap("chr1",
  start = 5000000, end = 6000000,
  strain1 = c("C57BL_6J"), strain2 = c("AKR_J", "A_J", "BALB_cJ"),
  return_obj = "granges"
)

df2 = annotate_consequences(geno.granges[seq_len(10), ], "mouse")
```

---

annotate\_mouse\_genes *Annotate with genes*

---

### Description

Request mouse genes from Ensembl Biomart.

### Usage

```
annotate_mouse_genes(geno, flanking = NULL)
```

### Arguments

geno	Data frame or GenomicRanges::GRanges object including columns chr, pos.
flanking	Size of flanking sequence to be included.

### Value

Data frame.

### Examples

```
geno = finemap("chr1",
  start = 5000000, end = 6000000,
  strain1 = c("C57BL_6J"), strain2 = c("AKR_J", "A_J", "BALB_cJ")
)

genes = annotate_mouse_genes(geno, 50000)
```

avail\_chromosomes      *Available chromosomes*

---

**Description**

Available mouse chromosomes.

**Usage**

```
avail_chromosomes()
```

**Value**

Data frame

**Examples**

```
avail_chromosomes()
```

---

avail\_consequences      *Available consequences*

---

**Description**

Available consequence and impact types.

**Usage**

```
avail_consequences()
```

**Value**

Data frame.

**Examples**

```
avail_consequences()$consequence  
unique(avail_consequences()$impact)
```

---

avail_strains	<i>Available strains</i>
---------------	--------------------------

---

**Description**

There are 37 strains available.

**Usage**

```
avail_strains()
```

**Value**

Data frame.

**Examples**

```
avail_strains()
```

---

backend_request	<i>Send HTTP request to backend server</i>
-----------------	--

---

**Description**

Send HTTP request to backend server

**Usage**

```
backend_request(q, n.tries = 2, method = "GET")
```

**Arguments**

q	Query string
n.tries	Number of tries
method	HTTP method to use

**Value**

Data frame.

---

comb	<i>Strain combination builder</i>
------	-----------------------------------

---

### Description

Generate strain sets and calculate reduction factors

### Usage

```
comb(geno, min_strain_benef = 0.1, max_set_size = 3)
```

### Arguments

geno	Data frame of genotypes for additional strains.
min_strain_benef	Minimum reduction factor (min) of a single strain. Default is 0.1.
max_set_size	Maximum set of strains. Default is 3.

### Value

Data frame

---

df2GRanges	<i>Data frame to GenomicRanges::GRanges object</i>
------------	--

---

### Description

Wrapper for GenomicRanges::makeGRangesFromDataFrame().

### Usage

```
df2GRanges(
  geno,
  chr_name = "chr",
  start_name = "pos",
  end_name = "pos",
  strand_name = NULL,
  ref_version = ref_genome(),
  seq_lengths = NULL,
  is_circular = FALSE
)
```

**Arguments**

geno	Data frame.
chr_name	Name of chromosome column. Default is 'chr'.
start_name	Name of start position column. Default is 'pos.'
end_name	Name of end position column. Default is 'pos'
strand_name	Name of end position column. Default is NULL.
ref_version	Reference genome version. Default is 'ref_genome()'.
seq_lengths	List of sequence lengths with sequence name as key. Default is NULL.
is_circular	Whether genome is circular. Default is FALSE.

**Value**

GenomicRanges::GRanges object.

**Examples**

```

geno = finemap("chr1",
  start = 5000000, end = 6000000,
  strain1 = c("C57BL_6J"), strain2 = c("AKR_J", "A_J", "BALB_cJ")
)

geno$strand = "+"
seq_lengths = stats::setNames(
  as.list(avail_chromosomes())$length),
  avail_chromosomes())$chr
)
geno.granges = df2GRanges(geno,
  strand_name = "strand",
  seq_lengths = seq_lengths
)

```

---

df\_split

*Splits data frame df into subsets with maximum n rows*


---

**Description**

Splits data frame df into subsets with maximum n rows

**Usage**

```
df_split(df, n)
```

**Arguments**

df	Data frame.
n	Max number of rows per subset.

**Value**

List of data frames.

---

ensembl_rest_vep	<i>Request variant consequences from Variant Effect Predictor (VEP) via Ensembl Rest Service</i>
------------------	--

---

**Description**

Request variant consequences from Variant Effect Predictor (VEP) via Ensembl Rest Service

**Usage**

```
ensembl_rest_vep(geno, species)
```

**Arguments**

geno	Data frame including columns rsid, ref, alt.
species	Species name, e.g. mouse or human.

**Value**

Data frame.

---

fetch	<i>Fetch</i>
-------	--------------

---

**Description**

Fetch homozygous genotypes for a specified chromosomal region in 37 inbred mouse strains.

**Usage**

```
fetch(
  chr,
  start = NULL,
  end = NULL,
  consequence = NULL,
  impact = NULL,
  return_obj = "dataframe"
)
```



**Arguments**

chr	Vector of chromosome names.
start	Optional vector of chromosomal start positions of target regions (GRCm38).
end	Optional vector of chromosomal end positions of target regions (GRCm38).
consequence	Optional vector of consequence types.
impact	Optional vector of impact types.
return_obj	The user can choose to get the result to be returned as data frame ("dataframe") or as a GenomicRanges::GRanges ("granges") object. Default value is "dataframe".

**Value**

Data frame or GenomicRanges::GRanges object containing result data.

**Examples**

```
geno = fetch("chr7", start = 5000000, end = 6000000)
comment(geno)
```

---

finemap

*Finemapping of genetic regions*

---

**Description**

Finemapping of genetic regions in 37 inbred mice by taking advantage of their very high homozygosity rate (>95 chromosomal regions (GRCm38), this method extracts homozygous SNVs for which the allele differs between two sets of strains (e.g. case vs controls) and outputs respective causal SNV/gene candidates.

**Usage**

```
finemap(
  chr,
  start = NULL,
  end = NULL,
  strain1,
  strain2,
  consequence = NULL,
  impact = NULL,
  thr1 = 0,
  thr2 = 0,
  return_obj = "dataframe"
)
```

**Arguments**

chr	Vector of chromosome names.
start	Optional vector of chromosomal start positions of target regions (GRCm38).
end	Optional vector of chromosomal end positions of target regions (GRCm38).
strain1	First strain set with strains from avail_strains().
strain2	Second strain set with strains from avail_strains().
consequence	Optional vector of consequence types.
impact	Optional vector of impact types.
thr1	Number discordant strains in strain1. Between 0 and length(strain1)-1. 0 by default.
thr2	Number discordant strains in strain2. Between 0 and length(strain2)-1. 0 by default.
return_obj	The user can choose to get the result to be returned as data frame ("dataframe") or as a GenomicRanges::GRanges ("granges") object. Default value is "dataframe".

**Value**

Data frame or GenomicRanges::GRanges object containing result data.

**Examples**

```

geno = finemap("chr1",
  start = 5000000, end = 6000000,
  strain1 = c("C57BL_6J"), strain2 = c(
    "129S1_SvImJ", "129S5SvEvBrd",
    "AKR_J"
  )
)
comment(geno)

```

---

finemap\_query

*Finemap query builder*


---

**Description**

Finemap query builder

**Usage**

```

finemap_query(
  chr,
  start = NULL,
  end = NULL,
  strain1 = NULL,

```

```

    strain2 = NULL,
    consequence = NULL,
    impact = NULL,
    thr1 = 0,
    thr2 = 0
)

```

### Arguments

chr	Vector of chromosome names.
start	Optional vector of chromosomal start positions of target regions (GRCm38).
end	Optional vector of chromosomal end positions of target regions (GRCm38).
strain1	First strain set with strains from avail_strains().
strain2	Second strain set with strains from avail_strains().
consequence	Optional vector of consequence types.
impact	Optional vector of impact types.
thr1	Number discordant strains in strain1. Between 0 and length(strain1)-1. 0 by default.
thr2	Number discordant strains in strain2. Between 0 and length(strain2)-1. 0 by default.

### Value

Query string.

---

getURL	<i>Get backend service url</i>
--------	--------------------------------

---

### Description

Get backend service URL. Default: <http://45.85.146.64:9000/rest/finemap/>

### Usage

```
getURL()
```

### Value

URL string.

### Examples

```
getURL()
```

---

get_top	<i>Best strain combinations</i>
---------	---------------------------------

---

**Description**

Get best strain combinations

**Usage**

```
get_top(red, n_top)
```

**Arguments**

red	Reduction factors data frame.
n_top	Number of combinations to be returned.

**Value**

Data frame

**Examples**

```
l = prio("chr1",
        start = 5000000, end = 6000000,
        strain1 = "C57BL_6J", strain2 = "AKR_J"
      )

get_top(l$reduction, 3)
```

---

GRanges2df	<i>GenomicRanges::GRanges object to data frame</i>
------------	--

---

**Description**

Wrapper for as.data.frame().

**Usage**

```
GRanges2df(granges)
```

**Arguments**

granges	GenomicRanges::GRanges object
---------	-------------------------------

**Value**

Data frame.

**Examples**

```

geno.granges = finemap("chr1",
  start = 50000000, end = 60000000,
  strain1 = c("C57BL_6J"), strain2 = c("AKR_J", "A_J", "BALB_cJ"),
  return_obj = "granges"
)

geno = GRanges2df(geno.granges)

```

prio

*Prioritization of inbred mouse strains for refining genetic regions***Description**

This method allows to select strain combinations which best refine a specified genetic region (GRCm38). E.g. if a crossing experiment with two inbred mouse strains 'strain1' and 'strain2' resulted in a QTL, the outputted strain combinations can be used to refine the respective region in further crossing experiments.

**Usage**

```

prio(
  chr,
  start = NULL,
  end = NULL,
  strain1 = NULL,
  strain2 = NULL,
  consequence = NULL,
  impact = NULL,
  min_strain_benef = 0.1,
  max_set_size = 3,
  return_obj = "dataframe"
)

```

**Arguments**

chr	Vector of chromosome names.
start	Optional vector of chromosomal start positions of target regions (GRCm38).
end	Optional vector of chromosomal end positions of target regions (GRCm38).
strain1	First strain set with strains from avail_strains().
strain2	Second strain set with strains from avail_strains().
consequence	Optional vector of consequence types.
impact	Optional vector of impact types.
min_strain_benef	Minimum reduction factor (min) of a single strain.

`max_set_size` Maximum set of strains.

`return_obj` The user can choose to get the result to be returned as data frame ("dataframe") or as a `GenomicRanges::GRanges` ("granges") object. Default value is "data frame".

### Value

Data frame

### Examples

```
res = prio("chr1",
  start = 5000000, end = 6000000, strain1 = "C57BL_6J",
  strain2 = "AKR_J"
)

comment(res$genotypes)
```

---

reduction

*Reduction factor calculation*

---

### Description

Generate strain sets and calculate reduction factors

### Usage

```
reduction(combs, geno)
```

### Arguments

`combs` Data frame of strain sets.

`geno` Data frame of genotypes for additional strains.

### Value

Data frame

---

ref_genome	<i>Reference genome version</i>
------------	---------------------------------

---

**Description**

Returns version of reference genome used in package MouseFM.

**Usage**

```
ref_genome()
```

**Value**

Vector.

**Examples**

```
ref_genome()
```

---

setURL	<i>Set backend service url</i>
--------	--------------------------------

---

**Description**

Set backend service URL. Default: `http://45.85.146.64:9000/rest/finemap/`

**Usage**

```
setURL(url)
```

**Arguments**

`url` URL of backend service. With backslash at the end.

**Value**

No return value.

**Examples**

```
setURL("http://45.85.146.64:9000/rest/finemap/")
```

---

`vis_reduction_factors` *Visualize*

---

**Description**

Visualize reduction factors

**Usage**

```
vis_reduction_factors(geno, red, n_top)
```

**Arguments**

<code>geno</code>	Genotype data frame or GenomicRanges::GRanges object.
<code>red</code>	Reduction factor data frame.
<code>n_top</code>	Number of combinations to be returned.

**Value**

Data frame

**Examples**

```
l = prio(c("chr1", "chr2"),
        start = c(5000000, 5000000),
        end = c(6000000, 6000000), strain1 = c("C3H_HeH"), strain2 = "AKR_J"
        )

plots = vis_reduction_factors(l$genotypes, l$reduction, 2)

plots[[1]]
plots[[2]]
```



# Index

## \* **internal**

- backend\_request, 5
- comb, 6
- df\_split, 7
- ensembl\_rest\_vep, 8
- finemap\_query, 10
- reduction, 14

- annotate\_consequences, 2
- annotate\_mouse\_genes, 3
- avail\_chromosomes, 4
- avail\_consequences, 4
- avail\_strains, 5

- backend\_request, 5

- comb, 6

- df2GRanges, 6
- df\_split, 7

- ensembl\_rest\_vep, 8

- fetch, 8
- finemap, 9
- finemap\_query, 10

- get\_top, 12
- getURL, 11
- GRanges2df, 12

- prio, 13

- reduction, 14
- ref\_genome, 15

- setURL, 15

- vis\_reduction\_factors, 16