

# Package: MouseFM (via r-universe)

June 30, 2024

**Type** Package

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

**Version** 1.15.0

**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** 7.1.0

**Repository** <https://bioc.r-universe.dev>

**RemoteUrl** <https://github.com/bioc/MouseFM>

**RemoteRef** HEAD

**RemoteSha** 120d04a635a54a8f8a37fee6e8e9eecd5206bff

## Contents

annotate_consequences . . . . .	2
annotate_mouse_genes . . . . .	3

avail_chromosomes . . . . .	3
avail_consequences . . . . .	4
avail_strains . . . . .	4
df2GRanges . . . . .	5
fetch . . . . .	6
finemap . . . . .	7
getURL . . . . .	8
get_top . . . . .	8
GRanges2df . . . . .	9
prio . . . . .	10
ref_genome . . . . .	11
setURL . . . . .	11
vis_reduction_factors . . . . .	12

<b>Index</b>	<b>13</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	<i>Available consequences</i>
--------------------	-------------------------------

---

**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()
```

---

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

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

```

---

fetch

*Fetch*


---

### 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

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

```

---

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

---

**Description**

Get backend service URL. Default: <http://mousefm.genehopper.de/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

*GenomicRanges::GRanges object to data frame*

---

**Description**

Wrapper for `as.data.frame()`.

**Usage**

```
GRanges2df(granges)
```

**Arguments**

`granges`            `GenomicRanges::GRanges` object

**Value**

Data frame.

**Examples**

```
geno.granges = finemap("chr1",
  start = 5000000, end = 6000000,
  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)
```

---

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://mousefm.genehopper.de/rest/finemap/>

**Usage**

```
setURL(url)
```

**Arguments**

url	URL of backend service.
-----	-------------------------

**Value**

No return value.

**Examples**

```
setURL("http://backendserver.com")
```

---

`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

[annotate\\_consequences](#), [2](#)  
[annotate\\_mouse\\_genes](#), [3](#)  
[avail\\_chromosomes](#), [3](#)  
[avail\\_consequences](#), [4](#)  
[avail\\_strains](#), [4](#)

[df2GRanges](#), [5](#)

[fetch](#), [6](#)  
[finemap](#), [7](#)

[get\\_top](#), [8](#)  
[getURL](#), [8](#)  
[GRanges2df](#), [9](#)

[prio](#), [10](#)

[ref\\_genome](#), [11](#)

[setURL](#), [11](#)

[vis\\_reduction\\_factors](#), [12](#)