# Package 'diffdriver'

June 26, 2026

**Type** Package

**Title** Identify Differential Selection

**Version** 0.1.7

**Maintainer** Siming Zhao <siming.zhao@dartmouth.edu>

**Description** Tests for context-dependent selection on cancer driver genes using somatic mutation data. The package implements the DiffDriver statistical framework to assess whether the strength of positive selection on mutations in a driver gene is associated with tumor- or individual-level context variables, such as clinical traits, genomic features, or immune microenvironment subtypes. DiffDriver estimates individual- and position-specific background mutation rates, models selection as a deviation from the background rate using functional annotations, and tests context effects through a latent-variable logistic model. It provides utilities for preparing mutation and annotation data, fitting differential-selection models, running gene-level association tests, summarizing candidate genes, and visualizing mutation patterns. The method is described in Zhou et al. (2026) ``Detecting context-dependent selection on cancer driver genes with DiffDriver" <doi:10.64898/2026.04.06.716771>.

**URL** https://szhaolab.github.io/diffdriver/, https://github.com/szhaolab/diffdriver

**License** MIT + file LICENSE

**Encoding** UTF-8

**Imports** Matrix, stats, data.table, brglm, fastTopics, SQUAREM

**Suggests** knitr, rmarkdown, logging, testthat (>= 3.0.0)

**RoxygenNote** 7.2.3

**Config/testthat/edition** 3

**Depends** R (>= 3.5.0)

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Siming Zhao [aut, cre],
Jie Zhou [aut],
Qirui Zhang [aut]

**Repository** CRAN

**Date/Publication** 2026-06-26 09:50:10 UTC

# Contents



---

`ddmcode` *Add intercept, if have functtypecode, then code and move to the front. different from driverMAPS! only allows functtypecode =7 ||8 when functypecode is included in selectvars.*

---

## Description

Add intercept, if have functtypecode, then code and move to the front. different from driverMAPS! only allows functtypecode =7 ||8 when functypecode is included in selectvars.

## Usage

```
ddmcode(matrixlist, selectvars, functypecodelevel = NULL)
```

## Arguments

| | |
|---|---|
| `matrixlist` | A list of mutation or annotation matrices. |
| `selectvars` | Variables selected for the model. |
| `functypecodelevel` | Functional annotation code level. |

ddmodel *diffDriver model*

## Description

This model is applied on data of a single gene. It will infer effect size for both sample-level variable and positional level functional annotations. We used an EM algorithm to infer parameters.

## Usage

```
ddmodel(mut, e, mr, fe, label, ...)
```

## Arguments

| | |
|---|---|
| mut | a matrix of mutation status 0 or 1, rows positions, columns are samples. |
| e | a vector,phenotype of each sample, should match the columns of mut and mr |
| mr | a matrix, mutation rate of each sample at each mutation (log scale) that is not dependent on sample level factor |
| fe | a vector, increased mutation rate at each position, depending on e (log scale), should match the rows of mut and mr |
| label | A vector or factor of length nrow(mut) identifying mutation-position rows to be grouped and aggregated. |
| ... | Additional arguments passed to internal functions. |

## Value

A list with the following elements:

pvalue The likelihood-ratio test p-value comparing the null model without a context effect to the alternative model with a context effect.

res.null A list returned by the null-model fit. It contains loglikelihood, beta0, and alpha.

res.alt A list returned by the alternative-model fit. It contains loglikelihood, beta0, and alpha.

## Examples

```
# Synthetic mutation matrix: rows are positions and columns are samples.
mut <- matrix(0, nrow = 6, ncol = 20)

mut[1, 1] <- 1
mut[2, 5] <- 1

mut[c(1, 2), 11] <- 1
mut[c(2, 3), 12] <- 1
mut[c(3, 4), 13] <- 1
mut[c(4, 5), 14] <- 1
```

```
mut[c(5, 6), 15] <- 1
mut[c(1, 6), 16] <- 1
mut[c(1, 3), 17] <- 1
mut[c(2, 4), 18] <- 1
mut[c(3, 5), 19] <- 1
mut[c(4, 6), 20] <- 1

e <- c(rep(0, 10), rep(1, 10))

# Mutation rates and functional effects are on the log scale.
mr <- matrix(log(0.002), nrow = nrow(mut), ncol = ncol(mut))
fe <- rep(log(3), nrow(mut))

# Each row represents a separate mutation position.
label <- factor(seq_len(nrow(mut)))

result <- ddmodel(mut, e, mr, fe, label = label)
result$pvalue
```

---

`ddmodel_binary` *diffDriver model only for binary phenotype.*

---

## Description

This function uses the model as cmodel.frac, but generalizes to take more than 1 functional categories. This model is applied on data of a single gene

## Usage

```
ddmodel_binary(mut, e, bmr, fe)
```

## Arguments

| | |
|---|---|
| `mut` | a matrix of mutation status 0 or 1 |
| `e` | a vector,phenotype of each sample, should match the columns of `mut` and `bmr` |
| `bmr` | a matrix, background mutation rate of each sample at each mutation (log scale) |
| `fe` | a vector, increased mutation rate at each mutation, due to functional effect (log scale), should match the rows of `mut` and `bmr` |

## Value

A list with the following elements:

`pvalue` The likelihood-ratio test p-value comparing the null and alternative binary-phenotype models.

`null.eta0` The fitted null-model parameter.

`alt.eta` The fitted alternative-model parameters for the two phenotype groups.

`null.ll` The maximized log-likelihood under the null model.

`alt.ll` The maximized log-likelihood under the alternative model.

`all` A named numeric vector containing `pvalue`, `null_para`, `e=0`, `e=1`, `null_likelihood`, and `alt_likelihood`.

## Examples

```
# Synthetic mutation matrix: rows are positions and columns are samples.
mut <- matrix(
  c(
    1, 0, 0, 0, 0, 0, 1, 0,
    0, 0, 0, 1, 0, 0, 0, 0,
    0, 1, 0, 0, 0, 0, 0, 1
  ),
  nrow = 3,
  byrow = TRUE
)
e <- c(0, 0, 0, 0, 1, 1, 1, 1)

# Background mutation rates and functional effects are on the log scale.
bmr <- matrix(log(0.001), nrow = nrow(mut), ncol = ncol(mut))
fe <- log(c(2, 2, 2))

result <- ddmodel_binary(mut, e, bmr, fe)
result$pvalue
```

---

# ddmodel_binary_simple *diffDriver model only for binary phenotype, assuming bmr the same across samples.*

---

## Description

This function uses the model as cmodel.frac, but generalizes to take more than 1 functional categories. This model is applied on data of a single gene. This should give the same results as ddmodel_binary defined above.

## Usage

```
ddmodel_binary_simple(mut, e, bmr, fe)
```

## Arguments

| | |
|---|---|
| `mut` | a matrix of mutation status 0 or 1 |
| `e` | a vector,phenotype of each sample, should match the columns of `mut` and `bmr` |
| `bmr` | a matrix, background mutation rate of each sample at each mutation (log scale), as we assume bmr the same across samples, only the first column will be used. |
| `fe` | a vector, increased mutation rate at each mutation, due to functional effect (log scale), should match the rows of `mut` and `bmr` |

## Value

A list with the following elements:

`pvalue` The likelihood-ratio test p-value comparing the null and alternative binary-phenotype models.

`null.eta0` The fitted null-model parameter.

`alt.eta` The fitted alternative-model parameters for the two phenotype groups.

`null.ll` The maximized log-likelihood under the null model.

`alt.ll` The maximized log-likelihood under the alternative model.

## Examples

```
# Synthetic mutation matrix: rows are positions and columns are samples.
mut <- matrix(
  c(
    1, 0, 0, 0, 0, 0, 1, 0,
    0, 0, 0, 1, 0, 0, 0, 0,
    0, 1, 0, 0, 0, 0, 0, 1
  ),
  nrow = 3,
  byrow = TRUE
)
e <- c(0, 0, 0, 0, 1, 1, 1, 1)

# Background mutation rates and functional effects are on the log scale.
bmr <- matrix(log(0.001), nrow = nrow(mut), ncol = ncol(mut))
fe <- log(c(2, 2, 2))

result <- ddmodel_binary_simple(mut, e, bmr, fe)
result$pvalue
```

---

`diffdriver` *Run diffDriver with Input Files*

---

## Description

This function runs diffDriver.

## Usage

```
diffdriver(
  gene,
  mut,
  pheno,
  anno_dir = ".",
  k = 6,
  totalnttype = 96,
```

```
  BMRmode = c("signature", "regular"),
  output_dir = tempdir(),
  output_prefix = "diffdriver_results"
)
```

## Arguments

| | |
|---|---|
| gene | A vector of genes to be included in the analysis. |
| mut | A data frame containing all somatic mutations from the cohort. The format is:<br>**Chromosome** <int><br>**Position** <int><br>**Ref** <chr><br>**Alt** <chr><br>**SampleID** <chr><br>Example:<br>`Chromosome Position Ref Alt SampleID`<br>`1 19 55653236 C T TCGA-N6-A4VE-01A-11D-A28R-08` |
| pheno | A data frame containing sample phenotypes. The format is: #'<br>**SampleID** <chr><br>**Phenotype1** <dbl><br>**Phenotype2** <dbl><br>**...** ...<br>Example:<br>`SampleID SmokingCessation BMI`<br>`TCGA-N5-A4R8-01A-11D-A28R-08 0.5319630 20.0`<br>`TCGA-N5-A4RD-01A-11D-A28R-08 0.0448991 24.4` |
| anno_dir | The path to the directory with all the annotation files. Please download from Zenodo.The default is current folder |
| k | The number of topics used in modeling background mutation rate. The default is 6. |
| totalnttype | either 9 or 96. Will look for annotation files anno9_ntypexxx_annodata.txt when totalnttype is 9 or anno96_ntypexxx_annodata when totalnttype is 96. |
| BMRmode | There are two modes to run diffdriver. One is "signature", this will model individual level BMR, this is the default. The second one is "regular", this assumes BMR is the same across individuals, only models position-level difference. |
| output_dir | The path to the output directory. Defaults to `tempdir()`. |
| output_prefix | The prefix being added to the output file names. |

## Value

A data frame with one row per tested gene. The returned columns include `dd.p`, `mlr.p`, `mlr.v2.p`, `fisher.p`, `binom.p`, and `lr.p`, which are p-values from DiffDriver and comparator methods; the corresponding `.fdr` columns, which contain Benjamini-Hochberg adjusted p-values; and `mut.E1`, `mut.E0`, `E1`, and `E0`, which summarize mutation and sample counts in the two phenotype groups used by the Fisher test.

## Examples

```
phenof <- system.file(
  "extdata", "example_phenotypes.txt",
  package = "diffdriver"
)
genef <- system.file(
  "extdata", "example_gene.txt",
  package = "diffdriver"
)
mutf <- system.file(
  "extdata", "example_mutations.txt",
  package = "diffdriver"
)

pheno <- read.table(phenof, header = TRUE)
gene <- read.table(genef, header = FALSE)
mut <- read.table(mutf, header = TRUE)

## Not run:
# Download the annotation files and replace this with their directory.
annodir <- "/path/to/annodir96"

result <- diffdriver(
  gene = gene,
  mut = mut,
  pheno = pheno,
  anno_dir = annodir,
  k = 6,
  totalnttype = 96,
  BMRmode = "signature",
  output_dir = tempdir(),
  output_prefix = "diffdriver_example"
)
result

## End(Not run)
```

---

# genebinom *gene level binomial test*

---

## Description

gene level binomial test

## Usage

```
genebinom(mut, e)
```

### Arguments

| | |
|---|---|
| mut | Mutation data. |
| e | Phenotype or context variable. |

### Value

A list with the following element:

pvalue The p-value from a gene-level binomial test comparing mutation counts between the two phenotype groups.

### Examples

```
# Synthetic mutation matrix: rows are positions and columns are samples.
mut <- matrix(
  c(
    1, 0, 0, 1, 0, 0,
    0, 1, 0, 0, 1, 0
  ),
  nrow = 2,
  byrow = TRUE
)
e <- c(0, 0, 0, 1, 1, 1)

result <- genebinom(mut, e)
result$pvalue
```

---

## genefisher *gene level fisher's exact test*

---

### Description

gene level fisher's exact test

### Usage

```
genefisher(mut, e)
```

### Arguments

| | |
|---|---|
| mut | Mutation data. |
| e | Phenotype or context variable. |

## Value

A list with the following elements:

`pvalue` The p-value from a gene-level Fisher's exact test.

`count` A numeric vector `c(gmutc, gmutn, mu.c, mu.n)`, where `gmutc` and `gmutn` are mutation counts in the two phenotype groups, and `mu.c` and `mu.n` are the corresponding numbers of samples.

## Examples

```
# Synthetic mutation matrix: rows are positions and columns are samples.
mut <- matrix(
  c(
    1, 0, 0, 1, 0, 0,
    0, 1, 0, 0, 1, 0
  ),
  nrow = 2,
  byrow = TRUE
)
e <- c(0, 0, 0, 1, 1, 1)

result <- genefisher(mut, e)
result$pvalue
result$count
```

---

`genelr` *gene level logistic regression*

---

## Description

gene level logistic regression

## Usage

```
genelr(mut, e, covariates = rep(1, length(e)))
```

## Arguments

| | |
|---|---|
| `mut` | Mutation data. |
| `e` | Phenotype or context variable. |
| `covariates` | Optional covariates included in the model. |

## Value

A list with the following elements:

`res` The summary object from the fitted logistic regression model `glm(e ~ dstatus + covariates, family = binomial(link = "logit"))`.

`pvalue` The p-value for the mutation-status coefficient `dstatus`.

## Examples

```
# Synthetic mutation matrix: rows are positions and columns are samples.
mut <- matrix(
  c(
    1, 0, 0, 1, 0, 0,
    0, 1, 0, 0, 1, 0
  ),
  nrow = 2,
  byrow = TRUE
)
e <- c(0, 0, 0, 1, 1, 1)

result <- genelr(mut, e)
result$pvalue
```

---

`mlr` *gene level multiple linear regression*

---

## Description

gene level multiple linear regression

## Usage

```
mlr(mut, e, covariates = rep(1, length(e)))
```

## Arguments

| | |
|---|---|
| `mut` | Mutation data. |
| `e` | Phenotype or context variable. |
| `covariates` | Optional covariates included in the model. |

## Value

A list with the following elements:

`res` The summary object from the fitted linear model `lm(e ~ dstatus + covariates)`.

`pvalue` The p-value for the mutation-status coefficient `dstatus`.

## Examples

```
# Synthetic mutation matrix: rows are positions and columns are samples.
mut <- matrix(
  c(
    1, 0, 0, 1, 0, 0,
    0, 1, 0, 0, 1, 0
  ),
  nrow = 2,
```

```
  byrow = TRUE
)
e <- c(0, 0, 0, 1, 1, 1)

result <- mlr(mut, e)
result$pvalue
```

---

`mlr.v2` *gene level multiple linear regression, correcting for total number of mutations*

---

## Description

gene level multiple linear regression, correcting for total number of mutations

## Usage

```
mlr.v2(mut, e, nmut, covariates = 1)
```

## Arguments

| | |
|---|---|
| `mut` | Mutation data. |
| `e` | Phenotype or context variable. |
| `nmut` | Number of mutations. |
| `covariates` | Optional covariates included in the model. |

## Value

A list with the following elements:

`res` The summary object from the fitted linear model `lm(e ~ dstatus + nmut + covariates)`.

`pvalue` The p-value for the mutation-status coefficient `dstatus`, after adjusting for total mutation count and covariates.

## Examples

```
# Synthetic mutation matrix: rows are positions and columns are samples.
mut <- matrix(
  c(
    1, 0, 0, 1, 0, 0,
    0, 1, 0, 0, 1, 0
  ),
  nrow = 2,
  byrow = TRUE
)
e <- c(0, 0, 0, 1, 1, 1)

# Synthetic per-sample mutation counts.
```

```
nmut <- c(3, 4, 5, 6, 7, 8)

result <- mlr.v2(mut, e, nmut)
result$pvalue
```

---

optifix *optifix. Optimise with fixed parameters*

---

## Description

its like optim, but with fixed parameters.

## Usage

```
optifix(
  par,
  fixed,
  fn,
  gr = NULL,
  ...,
  method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN"),
  lower = -Inf,
  upper = Inf,
  control = list(),
  hessian = FALSE
)
```

## Arguments

| | |
|---|---|
| par | Initial values for the parameters to be optimized. |
| fixed | Logical or index vector indicating fixed parameters. |
| fn | Function to be minimized. |
| gr | Optional gradient function. |
| ... | Additional arguments passed to fn and gr. |
| method | Optimization method. |
| lower | Lower bounds for parameters. |
| upper | Upper bounds for parameters. |
| control | Control parameters passed to optimization. |
| hessian | Logical; whether to return the Hessian matrix. |

## Details

specify a second argument 'fixed', a vector of TRUE/FALSE values. If TRUE, the corresponding parameter in fn() is fixed. Otherwise its variable and optimised over.

The return thing is the return thing from optim() but with a couple of extra bits - a vector of all the parameters and a vector copy of the 'fixed' argument.

Written by Barry Rowlingson <b.rowlingson@lancaster.ac.uk> October 2011



---

# plot_mut *plot phenotype, mutation and annotation for a gene across samples*

---

## Description

plot phenotype, mutation and annotation for a gene across samples

## Usage

```
plot_mut(
  gene_name,
  mut,
  pheno,
  totalnttype = 96,
  anno_dir = ".",
  output_prefix = "plot",
  output_dir = tempdir()
)
```

## Arguments

| | |
|---|---|
| gene_name | Gene name to plot. |
| mut | Mutation data. |
| pheno | Phenotype or context data. |
| totalnttype | Total number of nucleotide context types. |
| anno_dir | Directory containing annotation files. |
| output_prefix | Prefix for output files. |
| output_dir | Directory for output files. Defaults to `tempdir()`. |

## Value

A numeric vector returned by the final `barplot()` call, giving the coordinates of the plotted bars. The function is primarily called for its side effect of drawing mutation and phenotype summary plots for the selected gene.

## Examples

```
phenof <- system.file(
  "extdata", "example_phenotypes.txt",
  package = "diffdriver"
)
mutf <- system.file(
  "extdata", "example_mutations.txt",
  package = "diffdriver"
)

pheno <- read.table(phenof, header = TRUE)
mut <- read.table(mutf, header = TRUE)

## Not run:
# Download the annotation files and replace this with their directory.
annodir <- "/path/to/annodir9"

plot_mut(
  gene_name = "PIK3CA",
  mut = mut,
  pheno = pheno,
  totalnttype = 9,
  anno_dir = annodir,
  output_dir = tempdir()
)

## End(Not run)
```

# Index