# Package ‘adelie’

June 20, 2024

**Title** A Fast and Flexible Group Elastic Net Solver

**Version** 1.0.1

**Description** R bindings for the Python package 'adelie'.
These bindings offer a general purpose group elastic net solver,
a wide range of matrix classes that can exploit special structure
to allow large-scale inputs, and an assortment of
generalized linear model classes for fitting various types of data.
The package is an implementa-
tion of Yang, J. and Hastie, T. (2024) <doi:10.48550/arXiv.2405.08631>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.1

**LinkingTo** Rcpp, RcppEigen

**SystemRequirements** C++17

**Imports** Matrix, r2r, Rcpp, methods

**Suggests** ggplot2, reshape2, latex2exp, cowplot, gridExtra, testthat
(>= 3.0.0), knitr, rmarkdown

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**URL** `https://github.com/JamesYang007/adelie-r`

**BugReports** `https://github.com/JamesYang007/adelie-r/issues`

**NeedsCompilation** yes

**Author** James Yang [aut, cre, cph],
Trevor Hastie [aut, cph, fnd],
Balasubramanian Narasimhan [aut]

**Maintainer** James Yang <`jamesyang916@gmail.com`>

**Repository** CRAN

**Date/Publication** 2024-06-20 05:40:02 UTC

## Contents

---

`gaussian_cov` *Solves group elastic net via covariance method.*

---

### Description

Solves group elastic net via covariance method.

### Usage

```
gaussian_cov(
  A,
  v,
  constraints = NULL,
  groups = NULL,
  alpha = 1,
  penalty = NULL,
  lmda_path = NULL,
  max_iters = as.integer(1e+05),
  tol = 1e-07,
  rdev_tol = 0.001,
  newton_tol = 1e-12,
  newton_max_iters = 1000,
  n_threads = 1,
  early_exit = TRUE,
  screen_rule = "pivot",
  min_ratio = 0.01,
  lmda_path_size = 100,
  max_screen_size = NULL,
  max_active_size = NULL,
  pivot_subset_ratio = 0.1,
  pivot_subset_min = 1,
  pivot_slack_ratio = 1.25,
  check_state = FALSE,
  progress_bar = TRUE,
  warm_start = NULL
)
```

## Arguments

| | |
|---|---|
| `A` | Positive semi-definite matrix. |
| `v` | Linear term. |
| `constraints` | Constraints. |
| `groups` | Groups. |
| `alpha` | Elastic net parameter. |
| `penalty` | Penalty factor. |
| `lmda_path` | The regularization path. |
| `max_iters` | Maximum number of coordinate descents. |
| `tol` | Coordinate descent convergence tolerance. |
| `rdev_tol` | Relative percent deviance explained tolerance. |
| `newton_tol` | Convergence tolerance for the BCD update. |
| `newton_max_iters` | Maximum number of iterations for the BCD update. |
| `n_threads` | Number of threads. |
| `early_exit` | `TRUE` if the function should early exit. |
| `screen_rule` | Screen rule. |
| `min_ratio` | Ratio between largest and smallest regularization. |
| `lmda_path_size` | Number of regularizations. |
| `max_screen_size` | Maximum number of screen groups. |
| `max_active_size` | Maximum number of active groups. |
| `pivot_subset_ratio` | Subset ratio of pivot rule. |

| | |
|---|---|
| `pivot_subset_min` | Minimum subset of pivot rule. |
| `pivot_slack_ratio` | Slack ratio of pivot rule. |
| `check_state` | Check state. |
| `progress_bar` | Progress bar. |
| `warm_start` | Warm start. |

### Value

State of the solver.

### Examples

```
set.seed(0)
n <- 100
p <- 200
X <- matrix(rnorm(n * p), n, p)
y <- X[,1] * rnorm(1) + rnorm(n)
A <- t(X) %*% X / n
v <- t(X) %*% y / n
state <- gaussian_cov(A, v)
```

---

## `glm.binomial` *Creates a Binomial GLM family object.*

---

### Description

Creates a Binomial GLM family object.

### Usage

```
glm.binomial(y, weights = NULL, link = "logit")
```

### Arguments

| | |
|---|---|
| `y` | Response vector. |
| `weights` | Observation weights. |
| `link` | The link function type. |

### Value

Binomial GLM object.

### Examples

```
n <- 100
y <- rbinom(n, 1, 0.5)
obj <- glm.binomial(y)
```

---

## glm.cox *Creates a Cox GLM family object.*

---

### Description

Creates a Cox GLM family object.

### Usage

```
glm.cox(start, stop, status, weights = NULL, tie_method = "efron")
```

### Arguments

| | |
|---|---|
| `start` | Start time vector. |
| `stop` | Stop time vector. |
| `status` | Status vector. |
| `weights` | Observation weights. |
| `tie_method` | The tie-breaking method. |

### Value

Cox GLM object.

### Examples

```
n <- 100
start <- sample.int(20, size=n, replace=TRUE)
stop <- start + 1 + sample.int(5, size=n, replace=TRUE)
status <- rbinom(n, 1, 0.5)
obj <- glm.cox(start, stop, status)
```

## glm.gaussian *Creates a Gaussian GLM family object.*

### Description

Creates a Gaussian GLM family object.

### Usage

```
glm.gaussian(y, weights = NULL, opt = TRUE)
```

### Arguments

| | |
|---|---|
| `y` | Response vector. |
| `weights` | Observation weights. |
| `opt` | If `TRUE`, an optimized routine is run. |

### Value

Gaussian GLM

### Examples

```
n <- 100
y <- rnorm(n)
obj <- glm.gaussian(y)
```

## glm.multigaussian *Creates a MultiGaussian GLM family object.*

### Description

Creates a MultiGaussian GLM family object.

### Usage

```
glm.multigaussian(y, weights = NULL, opt = TRUE)
```

### Arguments

| | |
|---|---|
| `y` | Response vector. |
| `weights` | Observation weights. |
| `opt` | If `TRUE`, an optimized routine is run. |

### Value

MultiGaussian GLM object.

### Examples

```
n <- 100
K <- 5
y <- matrix(rnorm(n*K), n, K)
obj <- glm.multigaussian(y)
```

---

## `glm.multinomial` *Creates a Multinomial GLM family object.*

---

### Description

Creates a Multinomial GLM family object.

### Usage

```
glm.multinomial(y, weights = NULL)
```

### Arguments

| | |
|---|---|
| `y` | Response vector. |
| `weights` | Observation weights. |

### Value

Multinomial GLM object.

### Examples

```
n <- 100
K <- 5
y <- t(rmultinom(n, 1, rep(1/K, K)))
obj <- glm.multinomial(y)
```

## glm.poisson *Creates a Poisson GLM family object.*

### Description

Creates a Poisson GLM family object.

### Usage

```
glm.poisson(y, weights = NULL)
```

### Arguments

| | |
|---|---|
| y | Response vector. |
| weights | Observation weights. |

### Value

Poisson GLM object.

### Examples

```
n <- 100
y <- rpois(n, 1)
obj <- glm.poisson(y)
```

## grpnet *Solves group elastic net via naive method.*

### Description

Solves group elastic net via naive method.

### Usage

```
grpnet(
  X,
  glm,
  constraints = NULL,
  groups = NULL,
  alpha = 1,
  penalty = NULL,
  offsets = NULL,
  lmda_path = NULL,
  irls_max_iters = as.integer(10000),
  irls_tol = 1e-07,
  max_iters = as.integer(1e+05),
  tol = 1e-07,
  adev_tol = 0.9,
  ddev_tol = 0,
  newton_tol = 1e-12,
  newton_max_iters = 1000,
  n_threads = 1,
  early_exit = TRUE,
  intercept = TRUE,
  screen_rule = "pivot",
  min_ratio = 0.01,
  lmda_path_size = 100,
  max_screen_size = NULL,
  max_active_size = NULL,
  pivot_subset_ratio = 0.1,
  pivot_subset_min = 1,
  pivot_slack_ratio = 1.25,
  check_state = FALSE,
  progress_bar = TRUE,
  warm_start = NULL
)
```

## Arguments

| | |
|---|---|
| `X` | Feature matrix. |
| `glm` | GLM object. |
| `constraints` | Constraints. |
| `groups` | Groups. |
| `alpha` | Elastic net parameter. |
| `penalty` | Penalty factor. |
| `offsets` | Offsets. |
| `lmda_path` | The regularization path. |
| `irls_max_iters` | Maximum number of IRLS iterations. |
| `irls_tol` | IRLS convergence tolerance. |
| `max_iters` | Maximum number of coordinate descents. |
| `tol` | Coordinate descent convergence tolerance. |
| `adev_tol` | Percent deviance explained tolerance. |
| `ddev_tol` | Difference in percent deviance explained tolerance. |
| `newton_tol` | Convergence tolerance for the BCD update. |
| `newton_max_iters` | Maximum number of iterations for the BCD update. |
| `n_threads` | Number of threads. |
| `early_exit` | `TRUE` if the function should early exit. |

| | |
|---|---|
| `intercept` | `TRUE` to fit with intercept. |
| `screen_rule` | Screen rule. |
| `min_ratio` | Ratio between largest and smallest regularization. |
| `lmda_path_size` | Number of regularizations. |
| `max_screen_size` | Maximum number of screen groups. |
| `max_active_size` | Maximum number of active groups. |
| `pivot_subset_ratio` | Subset ratio of pivot rule. |
| `pivot_subset_min` | Minimum subset of pivot rule. |
| `pivot_slack_ratio` | Slack ratio of pivot rule. |
| `check_state` | Check state. |
| `progress_bar` | Progress bar. |
| `warm_start` | Warm start. |

## Value

State of the solver.

## Examples

```
set.seed(0)
n <- 100
p <- 200
X <- matrix(rnorm(n * p), n, p)
y <- X[,1] * rnorm(1) + rnorm(n)
state <- grpnet(X, glm.gaussian(y))
```

---

`io.snp_phased_ancestry` *IO handler for SNP phased, ancestry matrix.*

---

## Description

IO handler for SNP phased, ancestry matrix.

## Usage

```
io.snp_phased_ancestry(filename, read_mode = "file")
```

### Arguments

| | |
|---|---|
| `filename` | File name. |
| `read_mode` | Reading mode. |

### Value

IO handler for SNP phased, ancestry data.

### Examples

```
n <- 123
s <- 423
A <- 8
filename <- paste(tempdir(), "snp_phased_ancestry_dummy.snpdat", sep="/")
handle <- io.snp_phased_ancestry(filename)
calldata <- matrix(
    as.integer(sample.int(
        2, n * s * 2,
        replace=TRUE,
        prob=c(0.7, 0.3)
    ) - 1),
    n, s * 2
)
ancestries <- matrix(
    as.integer(sample.int(
        A, n * s * 2,
        replace=TRUE,
        prob=rep_len(1/A, A)
    ) - 1),
    n, s * 2
)
handle$write(calldata, ancestries, A, 1)
handle$read()
file.remove(filename)
```

---

## `io.snp_unphased` *IO handler for SNP unphased matrix.*

---

### Description

IO handler for SNP unphased matrix.

### Usage

```
io.snp_unphased(filename, read_mode = "file")
```

### Arguments

| | |
|---|---|
| `filename` | File name. |
| `read_mode` | Reading mode. |

### Value

IO handler for SNP unphased data.

### Examples

```
n <- 123
s <- 423
filename <- paste(tempdir(), "snp_unphased_dummy.snpdat", sep="/")
handle <- io.snp_unphased(filename)
mat <- matrix(
    as.integer(sample.int(
        3, n * s,
        replace=TRUE,
        prob=c(0.7, 0.2, 0.1)
    ) - 1),
    n, s
)
impute <- double(s)
handle$write(mat, "mean", impute, 1)
handle$read()
file.remove(filename)
```

---

## `matrix.block_diag` *Creates a block-diagonal matrix.*

---

### Description

Creates a block-diagonal matrix.

### Usage

```
matrix.block_diag(mats, n_threads = 1)
```

### Arguments

| | |
|---|---|
| `mats` | List of matrices. |
| `n_threads` | Number of threads. |

### Value

Block-diagonal matrix.

## Examples

```
n <- 100
ps <- c(10, 20, 30)
mats <- lapply(ps, function(p) {
    X <- matrix(rnorm(n * p), n, p)
    matrix.dense(t(X) %*% X, method="cov")
})
out <- matrix.block_diag(mats)
```

---

`matrix.concatenate` *Creates a concatenation of the matrices.*

---

## Description

Creates a concatenation of the matrices.

## Usage

```
matrix.concatenate(mats, axis = 0, n_threads = 1)
```

## Arguments

| | |
|---|---|
| `mats` | List of matrices. |
| `axis` | The axis along which the matrices will be joined. |
| `n_threads` | Number of threads. |

## Value

Concatenation of matrices.

## Examples

```
n <- 100
ps <- c(10, 20, 30)
mats <- lapply(ps, function(p) {
    matrix.dense(matrix(rnorm(n * p), n, p))
})
out <- matrix.concatenate(mats, axis=1)
ns <- c(10, 20, 30)
p <- 100
mats <- lapply(ns, function(n) {
    matrix.dense(matrix(rnorm(n * p), n, p))
})
out <- matrix.concatenate(mats, axis=0)
```

## matrix.dense — *Creates a viewer of a dense matrix.*

### Description

Creates a viewer of a dense matrix.

### Usage

```
matrix.dense(mat, method = "naive", n_threads = 1)
```

### Arguments

| | |
|---|---|
| `mat` | The dense matrix. |
| `method` | Method type. |
| `n_threads` | Number of threads. |

### Value

Dense matrix.

### Examples

```
n <- 100
p <- 20
X_dense <- matrix(rnorm(n * p), n, p)
out <- matrix.dense(X_dense, method="naive")
A_dense <- t(X_dense) %*% X_dense
out <- matrix.dense(A_dense, method="cov")
```

## matrix.interaction — *Creates a matrix with pairwise interactions.*

### Description

Creates a matrix with pairwise interactions.

### Usage

```
matrix.interaction(mat, intr_keys, intr_values, levels = NULL, n_threads = 1)
```

### Arguments

| | |
|---|---|
| `mat` | The dense matrix. |
| `intr_keys` | List of feature indices. |
| `intr_values` | List of list of feature indices. |
| `levels` | Levels. |
| `n_threads` | Number of threads. |

### Value

Pairwise interaction matrix.

### Examples

```
n <- 10
p <- 20
X_dense <- matrix(rnorm(n * p), n, p)
X_dense[,1] <- rbinom(n, 4, 0.5)
intr_keys <- c(0, 1)
intr_values <- list(NULL, c(0, 2))
levels <- c(c(5), rep(0, p-1))
out <- matrix.interaction(X_dense, intr_keys, intr_values, levels)
```

---

## `matrix.kronecker_eye` *Creates a Kronecker product with identity matrix.*

---

### Description

Creates a Kronecker product with identity matrix.

### Usage

```
matrix.kronecker_eye(mat, K, n_threads = 1)
```

### Arguments

| | |
|---|---|
| `mat` | The matrix to view as a Kronecker product. |
| `K` | Dimension of the identity matrix. |
| `n_threads` | Number of threads. |

### Value

Kronecker product with identity matrix.

### Examples

```
n <- 100
p <- 20
K <- 2
mat <- matrix(rnorm(n * p), n, p)
out <- matrix.kronecker_eye(mat, K)
mat <- matrix.dense(mat)
out <- matrix.kronecker_eye(mat, K)
```

---

## `matrix.lazy_cov` *Creates a lazy covariance matrix.*

---

### Description

Creates a lazy covariance matrix.

### Usage

```
matrix.lazy_cov(mat, n_threads = 1)
```

### Arguments

| | |
|---|---|
| `mat` | The data matrix. |
| `n_threads` | Number of threads. |

### Value

Lazy covariance matrix.

### Examples

```
n <- 100
p <- 20
mat <- matrix(rnorm(n * p), n, p)
out <- matrix.lazy_cov(mat)
```

## `matrix.one_hot` *Creates a one-hot encoded matrix.*

### Description

Creates a one-hot encoded matrix.

### Usage

```
matrix.one_hot(mat, levels = NULL, n_threads = 1)
```

### Arguments

| | |
|---|---|
| `mat` | The dense matrix. |
| `levels` | Levels. |
| `n_threads` | Number of threads. |

### Value

One-hot encoded matrix.

### Examples

```
n <- 100
p <- 20
mat <- matrix(rnorm(n * p), n, p)
out <- matrix.one_hot(mat)
```

## `matrix.snp_phased_ancestry` *Creates a SNP phased, ancestry matrix.*

### Description

Creates a SNP phased, ancestry matrix.

### Usage

```
matrix.snp_phased_ancestry(io, n_threads = 1)
```

### Arguments

| | |
|---|---|
| `io` | IO handler. |
| `n_threads` | Number of threads. |

## Value

SNP phased, ancestry matrix.

## Examples

```
n <- 123
s <- 423
A <- 8
filename <- paste(tempdir(), "snp_phased_ancestry_dummy.snpdat", sep="/")
handle <- io.snp_phased_ancestry(filename)
calldata <- matrix(
    as.integer(sample.int(
        2, n * s * 2,
        replace=TRUE,
        prob=c(0.7, 0.3)
    ) - 1),
    n, s * 2
)
ancestries <- matrix(
    as.integer(sample.int(
        A, n * s * 2,
        replace=TRUE,
        prob=rep_len(1/A, A)
    ) - 1),
    n, s * 2
)
handle$write(calldata, ancestries, A, 1)
out <- matrix.snp_phased_ancestry(handle)
file.remove(filename)
```

---

## `matrix.snp_unphased` *Creates a SNP unphased matrix.*

---

### Description

Creates a SNP unphased matrix.

### Usage

```
matrix.snp_unphased(io, n_threads = 1)
```

### Arguments

| | |
|---|---|
| `io` | IO handler. |
| `n_threads` | Number of threads. |

### Value

SNP unphased matrix.

### Examples

```
n <- 123
s <- 423
filename <- paste(tempdir(), "snp_unphased_dummy.snpdat", sep="/")
handle <- io.snp_unphased(filename)
mat <- matrix(
    as.integer(sample.int(
        3, n * s,
        replace=TRUE,
        prob=c(0.7, 0.2, 0.1)
    ) - 1),
    n, s
)
impute <- double(s)
handle$write(mat, "mean", impute, 1)
out <- matrix.snp_unphased(handle)
file.remove(filename)
```

---

## `matrix.sparse` *Creates a viewer of a sparse matrix.*

---

### Description

Creates a viewer of a sparse matrix.

### Usage

```
matrix.sparse(mat, method = "naive", n_threads = 1)
```

### Arguments

| | |
|---|---|
| `mat` | The sparse matrix to view. |
| `method` | Method type. |
| `n_threads` | Number of threads. |

### Value

Sparse matrix.

### Examples

```
n <- 100
p <- 20
X_dense <- matrix(rnorm(n * p), n, p)
X_sp <- as(X_dense, "dgCMatrix")
out <- matrix.sparse(X_sp, method="naive")
A_dense <- t(X_dense) %*% X_dense
A_sp <- as(A_dense, "dgCMatrix")
out <- matrix.sparse(A_sp, method="cov")
```

## `matrix.standardize` *Creates a standardized matrix.*

### Description

Creates a standardized matrix.

### Usage

```
matrix.standardize(mat, centers = NULL, scales = NULL, ddof = 0, n_threads = 1)
```

### Arguments

| | |
|---|---|
| `mat` | The underlying matrix. |
| `centers` | The center values. |
| `scales` | The scale values. |
| `ddof` | Degrees of freedom. |
| `n_threads` | Number of threads. |

### Value

Standardized matrix.

### Examples

```
n <- 100
p <- 20
X <- matrix(rnorm(n * p), n, p)
out <- matrix.standardize(matrix.dense(X))
```

## `matrix.subset` *Creates a subset of the matrix along an axis.*

### Description

Creates a subset of the matrix along an axis.

### Usage

```
matrix.subset(mat, indices, axis = 0, n_threads = 1)
```

### Arguments

| | |
|---|---|
| `mat` | The matrix to subset. |
| `indices` | Array of indices to subset the matrix. |
| `axis` | The axis along which to subset. |
| `n_threads` | Number of threads. |

### Value

Subset of the matrix along an axis.

### Examples

```
n <- 100
p <- 20
X <- matrix.dense(matrix(rnorm(n * p), n, p))
indices <- c(1, 3, 10)
out <- matrix.subset(X, indices, axis=0)
out <- matrix.subset(X, indices, axis=1)
```

---

## `set_configs` *Set configuration settings.*

---

### Description

Set configuration settings.

### Usage

```
set_configs(name, value = NULL)
```

### Arguments

| | |
|---|---|
| `name` | Configuration variable name. |
| `value` | Value to assign to the configuration variable. |

### Value

Assigned value.

### Examples

```
set_configs("hessian_min", 1e-6)
set_configs("hessian_min")
```

# Index