Package ‘gettingtothebottom’

February 19, 2015

Title Learning Optimization and Machine Learning for Statistics

Description Getting to the Bottom accompanies the “Getting to the Bottom” optimization methods series at Statisticsviews.com. It contains data and code to reproduce the examples in the articles.

URL http://jocelynchi.com/gettingtothebottom

Version 3.2

Author Jocelyn T. Chi <jocelynchi@alum.berkeley.edu>

Maintainer Jocelyn T. Chi <jocelynchi@alum.berkeley.edu>

Depends R (>= 3.0.2), ggplot2, grid, Matrix

License MIT + file LICENSE

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2014-12-05 01:37:22

R topics documented:

baltimoreyouth .............................................................. 2
diff_norm ................................................................. 4
ingel ................................................................. 5
dexample.alpha ........................................................... 5
dexample.quadratic.approx ........................................... 6
gdescent ................................................................. 6
generate_nnm ............................................................ 9
gettingtothebottom ..................................................... 10
init.lambda .............................................................. 10
makeLambdaseq .......................................................... 11
makeOmega ............................................................... 12
makeY ................................................................. 12
makeZ ................................................................. 13
matrixcomplete .......................................................... 14
moviebudgets ............................................................. 15
Description

Data from the 2011 release of Baltimore’s Community Health Profiles and Healthy Baltimore 2015 Safe Homes and Families indicators. Data provided by the Maryland Department of Health and Mental Hygiene, Maryland Department of the Environment, Baltimore Substance Abuse Systems (BSAS), the Baltimore City Health Department, and the United States Bureau of the Census

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSA2010</td>
<td>Name of Community Statistical Areas in 2010</td>
</tr>
<tr>
<td>eattendXX</td>
<td>Number of Students Ever Enrolled 1st - 5th Grade</td>
</tr>
<tr>
<td>mattendXX</td>
<td>Number of Students Ever Enrolled 6th - 8th Grade</td>
</tr>
<tr>
<td>heatattendXX</td>
<td>Number of Students Ever Enrolled 9th - 12th Grade</td>
</tr>
<tr>
<td>eenrolXX</td>
<td>Number of Students Enrolled in 1st - 5th Grade</td>
</tr>
<tr>
<td>menrolXX</td>
<td>Number of Students Enrolled in 6th - 8th Grade</td>
</tr>
<tr>
<td>henrolXX</td>
<td>Number of Students Enrolled in 9th - 12th Grade</td>
</tr>
<tr>
<td>aastudXX</td>
<td>Percent of Students that are African American</td>
</tr>
<tr>
<td>wstudXX</td>
<td>Percent of Students that are White (non-Hispanic)</td>
</tr>
<tr>
<td>hstudXX</td>
<td>Percent of Students that are Hispanic</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>abseXX</td>
<td>Percent of 1st-5th Grade Students that are Chronically Absent (Missing at least 20 days)</td>
</tr>
<tr>
<td>absmdXX</td>
<td>Percent of 6th-8th Grade Students that are Chronically Absent (Missing at least 20 days)</td>
</tr>
<tr>
<td>abshsXX</td>
<td>Percent of 9th-12th Grade Students that are Chronically Absent (Missing at least 20 days)</td>
</tr>
<tr>
<td>suspXX</td>
<td>Percentage of Students Suspended or Expelled During School Year</td>
</tr>
<tr>
<td>farmsXX</td>
<td>Percentage of Students Receiving Free or Reduced Meals</td>
</tr>
<tr>
<td>spedXX</td>
<td>Percentage of Students Enrolled in Special Education Programs</td>
</tr>
<tr>
<td>readyXX</td>
<td>Kindergarten School Readiness</td>
</tr>
<tr>
<td>3mathXX</td>
<td>Percentage of 3rd Grade Students Passing MSA Math</td>
</tr>
<tr>
<td>3readXX</td>
<td>Percentage of 3rd Grade Students Passing MSA Reading</td>
</tr>
<tr>
<td>5mathXX</td>
<td>Percentage of 5th Grade Students Passing MSA Math</td>
</tr>
<tr>
<td>5readXX</td>
<td>Percentage of 5th Grade Students Passing MSA Reading</td>
</tr>
<tr>
<td>8mathXX</td>
<td>Percentage of 8th Grade Students Passing MSA Math</td>
</tr>
<tr>
<td>8readXX</td>
<td>Percentage of 8th Grade Students Passing MSA Reading</td>
</tr>
<tr>
<td>hsaengXX</td>
<td>Percentage of Students Passing H.S.A. English</td>
</tr>
<tr>
<td>hsabioXX</td>
<td>Percentage of Students Passing H.S.A. Biology</td>
</tr>
<tr>
<td>hsagovXX</td>
<td>Percentage of Students Passing H.S.A. Government</td>
</tr>
<tr>
<td>hsaalgXX</td>
<td>Percentage of Students Passing H.S.A. Algebra</td>
</tr>
<tr>
<td>dropXX</td>
<td>High School Dropout/Withdrawal Rate</td>
</tr>
<tr>
<td>complXX</td>
<td>High School Completion Rate</td>
</tr>
<tr>
<td>sclswXX</td>
<td>Percent of Students Switching Schools within School Year</td>
</tr>
<tr>
<td>sclempXX</td>
<td>Percentage of Population aged 16-19 in School and/or Employed</td>
</tr>
<tr>
<td>teenbirXX</td>
<td>Teen Birth Rate per 1,000 Females (aged 15-19)</td>
</tr>
<tr>
<td>termbirXX</td>
<td>Percent of Births Delivered at Term (37-42 Weeks)</td>
</tr>
<tr>
<td>birthwtxXX</td>
<td>Percent of Babies Born with a Satisfactory Birth Weight</td>
</tr>
<tr>
<td>prenatal1XX</td>
<td>Percent of Births Where the Mother Received Early Prenatal Care (First Trimester)</td>
</tr>
<tr>
<td>leadtestXX</td>
<td>Number of Children (aged 0-6) Tested for Elevated Blood Lead Levels</td>
</tr>
<tr>
<td>eb11XX</td>
<td>Percent of Children (aged 0-6) with Elevated Blood Lead Levels</td>
</tr>
<tr>
<td>leadvXX</td>
<td>Percent of Lead Violations per 1,000 Residential Units</td>
</tr>
<tr>
<td>tanfXX</td>
<td>Percent of Families Receiving TANF</td>
</tr>
<tr>
<td>liquorXX</td>
<td>Liquor Outlet density (per 1,000 Residents)</td>
</tr>
<tr>
<td>fastfdXX</td>
<td>Fast Food Outlet Density (per 1,000 Residents)</td>
</tr>
<tr>
<td>LifeExpXX</td>
<td>Life Expectancy</td>
</tr>
<tr>
<td>mort1_XX</td>
<td>Mortality by Age (Less than 1 year old)</td>
</tr>
<tr>
<td>mort14_XX</td>
<td>Mortality by Age (1-14 years old)</td>
</tr>
<tr>
<td>mort24_XX</td>
<td>Mortality by Age (15-24 years old)</td>
</tr>
<tr>
<td>mort44_XX</td>
<td>Mortality by Age (25-44 years old)</td>
</tr>
<tr>
<td>mort64_XX</td>
<td>Mortality by Age (45-64 years old)</td>
</tr>
<tr>
<td>mort84_XX</td>
<td>Mortality by Age (65-84 years old)</td>
</tr>
<tr>
<td>mort85_XX</td>
<td>Mortality by Age (85 and over)</td>
</tr>
</tbody>
</table>
**Format**

A data frame with 55 rows and 79 variables

**Source**

Data obtained from the Baltimore Neighborhood Indicators Alliance using the Children and Family Health & Well-Being and Education indicators. [http://www.bniajfi.org/data_downloads](http://www.bniajfi.org/data_downloads)

---

**diff_norm**

*MM Algorithm - Normed Difference*

**Description**

`diff_norm` Function for finding the normed difference between two matrices based on vector containing indices of differing elements

**Usage**

`diff_norm(X, Z, omega)`

**Arguments**

- **X**: Original data matrix
- **Z**: Model matrix for comparison
- **omega**: Set of unobserved indices

**Author(s)**

Jocelyn T. Chi

**Examples**

```r
Z <- matrix(rnorm(9,0,1),3,3)
X <- matrix(rnorm(9,0,2),3,3)
omega <- c(2,5,6)
diff_norm(X,Z,omega)
```
**Description**

Data on income and food expenditure for 235 working class households in 1857 Belgium.

**Arguments**

- **income**: Annual household income (Belgian francs)
- **foodexp**: Annual household food expenditure (Belgian francs)

**Format**

A dataset containing 235 observations on 2 variables

**Source**

This dataset was used in Koenker and Bassett (1982) and obtained from the quantreg package for R. Citations: Koenker, R. and Bassett, G (1982) Robust Tests of Heteroscedasticity based on Regression Quantiles; Econometrica 50, 43-61. [http://CRAN.R-project.org/package=quantreg](http://CRAN.R-project.org/package=quantreg)

---

**Example**

**Description**

`example.alpha` Plots the function values for gradient descent in Example 1 (without the intercept) given a particular value for alpha.

**Usage**

`example.alpha(alpha)`

**Arguments**

- **alpha**: Step-size for the gradient descent algorithm.

**Author(s)**

Jocelyn T. Chi

**Examples**

- `example.alpha(0.01)`
- `example.alpha(0.12)`
**example.quadratic.approx**

*Gradient Descent Algorithm - Plots Depicting How Different Choices of Alpha Result in Differing Quadratic Approximations*

---

**Description**

`example.quadratic.approx` Shows how the quadratic approximations for the function in Example 1 change with choice of alpha.

**Usage**

`example.quadratic.approx(alpha1 = 0.01, alpha2 = 0.12)`

**Arguments**

- `alpha1`: A smaller step-size for the gradient descent algorithm.
- `alpha2`: A larger step-size for the gradient descent algorithm. (alpha2 > alpha1)

**Author(s)**

Jocelyn T. Chi

**Examples**

`example.quadratic.approx(alpha1=0.01, alpha2=0.12)`

---

**gdescent**

*Gradient Descent Algorithm*

---

**Description**

`gdescent` Performs gradient descent algorithm given an objective function and a gradient for the objective function

**Usage**

`gdescent(f, grad_f, X, y, alpha = 1e-06, iter = 3000, liveupdates = FALSE, tol = 1e-06, intercept = TRUE, autoscaling = TRUE)`
Arguments

- **f**: objective function as a function of X, y, and b
- **grad_f**: gradient of f as a function of X, y, and b
- **X**: matrix of independent variables
- **y**: vector containing dependent variable
- **alpha**: (optional) step size for the algorithm
- **iter**: (optional) the number of iterations to include in the algorithm
- **liveupdates**: (optional) if TRUE, the function will print live updates showing the norm of the gradient vector in each iteration
- **tol**: (optional) tolerance for determining convergence
- **intercept**: (optional) if TRUE, the model includes an estimate for the intercept
- **autoscaling**: (optional) if TRUE, the function will automatically rescale the columns of X (divides each element in X by the maximal element in that column)

Author(s)

Jocelyn T. Chi

Examples

```r
#-----------------------------------------------
# EXAMPLE 1 - A Simple Example
#-----------------------------------------------

# Generate some data for a simple bivariate example
set.seed(12345)
x <- sample(seq(from = -1, to = 1, by = 0.1), size = 50, replace = TRUE)
y <- 2*x + rnorm(50)
plot(x, y)

# Setting up for gradient descent
X <- as.matrix(x)
y <- as.vector(y)
f <- function(X,y,b) {
  (1/2)*norm(y-X%*%b,"F")^2
}
grad_f <- function(X,y,b) {
  t(X)%*%(X%*%b - y)
}

# Run a simple gradient descent example
simple_ex <- gdescent(f,grad_f,X,y,0.01)

# We can compare our gradient descent results with what we get if we use the lm function
lm(y~X)

# Notice that the algorithm may diverge if the step size (alpha) is not small enough
# THE FOLLOWING NOT RUN
```
# simple_ex2 <- gdescent(f, grad_f, x, y, alpha=0.05, liveupdates=TRUE)
# The live updates show the norm of the gradient in each iteration.
# We notice that the norm of the gradient diverges when alpha is not small enough.

# EX fazer atualizações
# EX K 2 - Linear Regression & Feature Scaling
# EXK---------------------------------------------------------------

f <- function(x, y, b) {
  (1/2) * norm(y - x %*% b, "F")^2
}

grad_f <- function(x, y, b) {
  t(x) %*% (x %*% b - y)
}

data(moviebudgets)
X <- as.matrix(moviebudgets$budget)
y <- as.vector(moviebudgets$rating)
# THE FOLLOWING NOT RUN
# movies1 <- gdescent(f, grad_f, x, y, 1e-4, 5000)

# We can compare our gradient descent results with what we get if we use the lm function
# THE FOLLOWING NOT RUN
# lm(y ~ x)

# Compare the above result with what we get without feature scaling
# Not run:
# movies2 <- gdescent(f, grad_f, x, y, alpha=1e-19, iter=10000, liveupdates=TRUE, autoscaling=FALSE)
## Note that running the gradient descent algorithm on unscaled column vectors
## requires a much smaller step size and many more iterations.

# EX fazer atualizações
# EXK---------------------------------------------------------------

f <- function(x, y, b) {
  (1/2) * norm(y - x %*% b, "F")^2
}

grad_f <- function(x, y, b) {
  t(x) %*% (x %*% b - y)
}

data(baltimoreyouth)
B <- baltimoreyouth
X <- matrix(c(B$farms11, B$susp11, B$sclemp11, B$abshs11), nrow=nrow(B), byrow=FALSE)
y <- as.vector(B$comp11)
# THE FOLLOWING NOT RUN
# meals_graduations <- gdescent(f, grad_f, x, y, 0.01, 12000)

# We can compare our gradient descent results with what we get if we use the lm function
# THE FOLLOWING NOT RUN
# lm(y~X)
generate_nnm
Generate random nonnegative mixture components

Description

generate_nnm Function to random nonnegative mixture components

Usage

generate_nnm(n, p, seed = 12345)

Arguments

n Number of samples
p Number of components
seed Random seed
Examples

```r
n <- 1e3
p <- 10
nnm <- generate_nnm(n, p)
```

Description

Getting to the Bottom is a companion package for the "Getting to the Bottom" optimization methods series at Statisticisviews.com. The package contains data and code to reproduce the examples in the articles.

Author(s)

Jocelyn T. Chi

init.lambda

**MM Algorithm - Initial lambda**

Description

init.lambda Function for finding an initial value for lambda

Usage

```r
init.lambda(X, omega)
```

Arguments

- `X`: Original data matrix
- `omega`: Set of unobserved indices

Author(s)

Jocelyn T. Chi
Examples

```r
# Generate a test matrix
seed <- 12345
m <- 100
n <- 100
r <- 3
T <- testmatrix(m, n, r, seed=seed)

# Add some noise to the test matrix
E <- 0.1*matrix(rnorm(m*n), m, n)
A <- T + E

# Obtain a vector of unobserved entries
temp <- makeOmega(m, n, percent=0.5)
omega <- temp$omega

# Remove unobserved entries from test matrix
X <- A
X[omega] <- NA

init.lambda(X, omega)
```

Description

makeLambdaseq Function for making sequence of lambdas for solution paths given starting lambda value and desired sequence length

Usage

```r
makeLambdaseq(L, lambdaseq_length)
```

Arguments

- `L` : Initial lambda value
- `lambdaseq_length` : Desired length of lambda sequence

Author(s)

Jocelyn T. Chi

Examples

```r
makeLambdaseq(11, 20)
```
makeOmega

**Description**

makeOmega Function for generating omega vector of missing values

**Usage**

makeOmega(m, n, percent, seed = 123)

**Arguments**

- **m**: Number of rows in matrix to be generated
- **n**: Number of columns in matrix to be generated
- **percent**: Percent missing in matrix
- **seed**: Random seed

**Author(s)**

Jocelyn T. Chi

**Examples**

```r
m <- 1000
n <- 1000
percent <- 0.75
omega <- makeOmega(m, n, percent)
```

makeY

**Description**

makeY Function for making the Y matrix

**Usage**

makeY(X, Z, omega)

**Arguments**

- **X**: Matrix containing observed entries
- **Z**: Matrix containing last iterates
- **omega**: Vector containing indices of unobserved entries (by column)
**makeZ**

**Author(s)**
Jocelyn T. Chi

**Examples**

```r
n <- 5
A <- matrix(rnorm(n^2), n, n)
omega <- c(1, 5, 8, 10, 16, 23)
Z <- Matrix(0, n, n, sparse=TRUE)
makeY(A, Z, omega)
```

---

### Description

`makeZ` Function for making the Z matrix

### Usage

```r
makeZ(M, lambda)
```

### Arguments

- **M**: Matrix containing observed entries
- **lambda**: Softthreshold parameter

### Author(s)
Jocelyn T. Chi

### Examples

```r
A <- matrix(rnorm(9), 3, 3)
makeZ(A, lambda=3)
```
**matrixcomplete**

**MM Algorithm - Matrix Completion**

**Description**

`matrixcomplete` Function for performing matrix completion using a majorization-minimization algorithm given data matrix $X$

**Usage**

```r
matrixcomplete(X, Z, omega, lambda, maxiter = 100, tol = 1e-04,
              liveupdates = TRUE)
```

**Arguments**

- **X**: Data matrix to be completed
- **Z**: Matrix containing last iterates
- **omega**: Vector containing indices of unobserved entries
- **lambda**: Softthreshold parameter
- **maxiter**: (Optional) Max number of iterations (Default: 100)
- **tol**: (Optional) Tolerance for convergence (Default: 1e-4)
- **liveupdates**: (Optional) If FALSE, no notification will be given upon completion of each iteration. (Default: TRUE)

**Author(s)**

Jocelyn T. Chi

**Examples**

```r
# (Examples not run)
# Generate an m-by-n test matrix of rank r
# seed <- 12345
# m <- 1000
# n <- 1000
# r <- 5
# T <- testmatrix(m,n,r,seed=seed)

# Add some noise to the test matrix
# E <- .1*matrix(rnorm(m*n),m,n)
# A <- T + E

# Obtain a vector of unobserved entries
# temp <- makeOmega(m,n,percent=0.5)
# omega <- temp$omega

# Remove unobserved entries from test matrix
```
# X <- A
# X[omega] <- NA

# Make initial model matrix Z and find initial lambda
# Z <- matrix(0,m,n)
# lambda <- init.lambda(X,omega)

# Example (Not run)
# Sol <- matrixcomplete(X,Z,omega,lambda)

## moviebudgets

.Movie ratings and budget database derived from data from IMDB.com.

---

**Description**

A dataset containing movie ratings and budget data for 5,183 movies.

**Arguments**

- **title**: Title of the movie
- **year**: Year the movie was released
- **budget**: Total budget (if known) in U.S. dollars
- **length**: Length of movie (in minutes)
- **rating**: Average IMDB user rating
- **votes**: Number of IMDB users who rated the movie
- **r1-10**: Distribution of votes for each rating, to mid point of nearest decile: 0 = no votes, 4.5 = 1-9 percent votes, 14.5 = 11-19 percent of votes, etc. Due to rounding errors these may not sum to 100.
- **mpaa**: MPAA rating
- **genre**: Binary variables indicating whether movie belongs to any of the following genres: action, animation, comedy, drama, documentary, romance, short

**Format**

A data frame with 5183 rows and 24 variables

**Source**

Data obtained from Hadley Wickham. The data in this package contains only those movies not exceeding 400 minutes in length and those with known total budgets. [http://had.co.nz/data/movies/](http://had.co.nz/data/movies/).
movieratings  Movie ratings database derived from data from IMDB.com

Description
A dataset containing movie ratings for 58,771 movies.

Arguments
- `title`: Title of the movie
- `year`: Year the movie was released
- `length`: Length of movie (in minutes)
- `rating`: Average IMDB user rating
- `votes`: Number of IMDB users who rated the movie
- `r1-10`: Distribution of votes for each rating, to mid point of nearest decile: 0 = no votes, 4.5 = 1-9 percent votes, 14.5 = 11-19 percent of votes, etc. Due to rounding errors these may not sum to 100.
- `mpaa`: MPAA rating
- `genre`: Binary variables indicating whether movie belongs to any of the following genres: action, animation, comedy, drama, documentary, romance, short

Format
A data frame with 58771 rows and 23 variables

Source
Data obtained from Hadley Wickham. The data in this package contains only those movies not exceeding 400 minutes in length. http://had.co.nz/data/movies/.

nnls_mm  Nonnegative Least Squares via MM

Description
`nnls_mm` iteratively computes the solution to the nonnegative least squares problem via a majorization-minimization algorithm.

Usage
`nnls_mm(y, X, b, max_iter = 100, tol = 1e-04)`
Arguments

- **y**: Nonnegative response
- **X**: Nonnegative design matrix
- **b**: Nonnegative initial regression vector
- **max_iter**: Maximum number of iterations
- **tol**: Relative tolerance for convergence

Examples

```r
set.seed(12345)
n <- 100
p <- 3
X <- matrix(rexp(n*p, rate=1), n, p)
b <- matrix(runif(p), p, 1)
y <- X %*% b + matrix(abs(rnorm(n)), n, 1)

## Setup mixture example
n <- 1e3
p <- 10
nnm <- generate_nnm(n, p)
set.seed(124)
X <- nnm$X
b <- double(p)
nComponents <- 3
k <- sample(1:p, nComponents, replace=FALSE)
b[k] <- matrix(runif(nComponents), ncol=1)
y <- X%*%b + 0.25*matrix(abs(rnorm(n)), n, 1)

# Obtain solution to mixture problem
nnm_sol <- nnls_mm(y, X, runif(p))
```

Description

A vector describing 1943 dietary requirements for a "moderately active" man of 154 pounds. Obtained from Table 1 of George Stigler’s 1945 paper on "The Cost of Subsistence".

Arguments

- **Calories**: Calories (in kilocalories)
- **Protein**: Protein (in grams)
- **Calcium**: Calcium (in grams)
Iron (in milligrams)
Vitamin A (in 1000 International Units)
Thiamine (in milligrams)
Riboflavin (in milligrams)
Niacin (in milligrams)
Ascorbic Acid (in milligrams)

Format
A vector of length 9

Source

---

`plot_gradient` *Gradient Descent Algorithm - Plotting the Gradient Function*

**Description**

`plot_gradient` plots the norm of the gradient function of an object containing the results of a gradient descent object implementation.

**Usage**

```r
plot_gradient(obj)
```

**Arguments**

- `obj`: Object containing the results of a gradient descent implementation.

**Author(s)**

Jocelyn T. Chi

**Examples**

```r
# Generate some data for a simple bivariate example
set.seed(12345)
x <- sample(seq(from = -1, to = 1, by = 0.1), size = 50, replace = TRUE)
y <- 2*x + rnorm(50)

# Components required for gradient descent
X <- as.matrix(x)
y <- as.vector(y)
f <- function(X, y, b) {
```

```r
```
(1/2)*norm(y-X*x%*%b,"F")^2
}

grad_f <- function(X,y,b) {
  t(X)%*%(X%*%b - y)
}

# Run a simple gradient descent example
simple_ex <- gdescent(f,grad_f,X,y,alpha=0.01)

# Plot the norm of the gradient function
plot_gradient(simple_ex)

---

### Description

plot_iterates Plots the iterates of an object containing the results of a gradient descent object implementation

### Usage

plot_iterates(obj)

### Arguments

- **obj** Object containing the results of a gradient descent implementation

### Author(s)

Jocelyn T. Chi

### Examples

# Generate some data for a simple bivariate example
set.seed(12345)
x <- sample(seq(from = -1, to = 1, by = 0.1), size = 50, replace = TRUE)
y <- 2*x + rnorm(50)

# Components required for gradient descent
X <- as.matrix(x)
y <- as.vector(y)
f <- function(X,y,b) {
  (1/2)*norm(y-X*x%*%b,"F")^2
}
grad_f <- function(X,y,b) {
  t(X)%*%(X%*%b - y)
}

# Run a simple gradient descent example
```
simple_ex <- gdescent(f, grad_f, X, y, alpha=0.01)

# Plot the iterates
plot_iterates(simple_ex)

plot_loss(obj)
```

**Description**

`plot_loss` Plots the loss function of an object containing the results of a gradient descent object implementation.

**Usage**

```r
plot_loss(obj)
```

**Arguments**

- **obj** Object containing the results of a gradient descent implementation.

**Author(s)**

Jocelyn T. Chi

**Examples**

```r
# Generate some data for a simple bivariate example
set.seed(12345)
x <- sample(seq(from = -1, to = 1, by = 0.1), size = 50, replace = TRUE)
y <- 2*x + rnorm(50)

# Components required for gradient descent
X <- as.matrix(x)
y <- as.vector(y)
f <- function(x, y, b) {
  (1/2)*norm(y - x*x*b, "F")^2
}
grad_f <- function(x, y, b) {
  t(X)*%%(X*x*b - y)
}

# Run a simple gradient descent example
simple_ex <- gdescent(f, grad_f, X, y, alpha=0.01)

# Plot the loss function
plot_loss(simple_ex)
```
plot_nnm

Description
plot_nnm Function for plotting nnm

Usage
plot_nnm(nnm)

Arguments
nnm NNM object from generate_nnm function

Author(s)
Jocelyn T. Chi

Examples
# Generate nonnegative matrix
n <- 1e3
p <- 10
nnm <- generate_nnm(n,p)

# Plot nonnegative matrix
plot_nnm(nnm)

plot_nnm_coef

Description
plot_nnm_coef Function for plotting the NNLS regression coefficients

Usage
plot_nnm_coef(nnm_sol)

Arguments
nnm_sol Solution object from nnls_mm function
Examples

```r
# Setup mixture example
n <- 1e3
p <- 10
nnm <- generate_nnm(n,p)

set.seed(12345)
X <- nnm$X
b <- double(p)
nComponents <- 3
k <- sample(1:p,nComponents,replace=FALSE)
b[k] <- matrix(runif(nComponents),ncol=1)
y <- X %*% b + 0.25 * matrix(abs(rnorm(n)),n,1)

# Obtain solution to mixture problem
nnm_sol <- nnls_mm(y,X,runif(p))

# Plot the regression coefficients
plot_nnm_coef(nnm_sol)
```

---

**plot_nnm_obj**

*MM Algorithm - Plot NNM Objective*

**Description**

`plot_nnm_obj` Function for plotting the NNM Objective Function

**Usage**

```r
plot_nnm_obj(y, X, b, max_iter = 100)
```

**Arguments**

- `y` Nonnegative response
- `X` Nonnegative design matrix
- `b` Nonnegative initial regression vector
- `max_iter` (Optional) Maximum number of iterations

**Author(s)**

Jocelyn T. Chi
Examples

```r
set.seed(12345)
n <- 100
p <- 3
X <- matrix(rexp(n*p, rate=1), n, p)
b <- matrix(runif(p), p, 1)
y <- X %*% b + matrix(abs(rnorm(n)), n, 1)
plot_nnm_obj(y, X, b)
```

plot_nnm_reconstruction

**MM Algorithm - Plotting the Reconstruction**

### Description

plot_nnm_reconstruction Function for plotting the nnm_sol reconstruction

### Usage

```r
plot_nnm_reconstruction(nnm, X, nnm_sol)
```

### Arguments

- `nnm`: NNM object from generate_nnm function
- `X`: Nonnegative design matrix
- `nnm_sol`: Solution object from nnls_mm function

### Examples

```r
# Setup mixture example
n <- 1e3
p <- 10
nnm <- generate_nnm(n, p)

set.seed(12345)
X <- nnnm$n
b <- double(p)
nComponents <- 3
k <- sample(1:p, nComponents, replace=FALSE)
b[k] <- matrix(runif(nComponents), ncol=1)
y <- X %*% b + 0.25*matrix(abs(rnorm(n)), n, 1)

# Obtain solution to mixture problem
nnm_sol <- nnls_mm(y, X, runif(p))

# Plot the reconstruction
plot_nnm_reconstruction(nnm, X, nnm_sol)
```
plot_nnm_truth

**MM Algorithm - Plotting the True Signal**

**Description**

plot_nnm_truth Function for plotting the true mixture signal

**Usage**

plot_nnm_truth(X, b, nnm)

**Arguments**

- **X**: Nonnegative design matrix
- **b**: Nonnegative initial regression vector
- **nnm**: NNM object from generate_nnm function

**Examples**

```r
# Setup mixture example
n <- 1e3
p <- 10
nnm <- generate_nnm(n, p)

set.seed(12345)
X <- nnnm$x
b <- double(p)
nComponents <- 3
k <- sample(1:p,nComponents,replace=FALSE)
b[k] <- matrix(runif(nComponents),ncol=1)
y <- X%*%b + 0.25*matrix(abs(rnorm(n)),n,1)

# Plot the truth
plot_nnm_truth(X,b,nnm)
```

plot_softhreshold

**MM Algorithm - Plot the Softthreshold Function**

**Description**

plot_softhreshold Function for plotting the softthreshold function

**Usage**

plot_softhreshold(from = -5, to = 5, lambda = 3)
plot_solpaths_error

Arguments
- **from**: The starting value of the sequence of inputs into the function
- **to**: The ending value of the sequence of inputs into the function
- **lambda**: Lambda value for softhreshold function

Author(s)
- Jocelyn T. Chi

Examples
- `plot_softhreshold(-5,5,3)`

---

plot_solpaths_error

**Description**

plot_solpaths_error Function for plotting the imputed values against the truth for minimum error solution found using solutionpaths function

**Usage**

`plot_solpaths_error(A, omega, ans)`

Arguments
- **A**: Initial test matrix of fully observed entries
- **omega**: Vector of unobserved entries
- **ans**: Results from solpaths function

Author(s)
- Jocelyn T. Chi

Examples
- # Generate a test matrix
  seed <- 12345
  m <- 100
  n <- 100
  r <- 3
  T <- testmatrix(m,n,r,seed=seed)

  # Add some noise to the test matrix
  E <- 0.1*matrix(rnorm(m*n),m,n)
A <- T + E

# Obtain a vector of unobserved entries
temp <- makeOmega(m,n,percent=0.5)
omega <- temp$omega

# Remove unobserved entries from test matrix
X <- A
X[omega] <- NA

# Make initial model matrix Z and find initial lambda
Z <- matrix(0,m,n)
lambda.start <- init.lambda(X,omega)
lambdaseq_length=20
tol <- 1e-2
ans <- solutionpaths(A,X,Z,omega,lambdaseq_length=20,
  liveupdates=FALSE,lambdaseq_length=lambdaseq_length)

plot_solutionpaths_error(A,omega,ans)

---

**plot_solutionpaths**  
*MM Algorithm - Plot results of solutionpaths function*

---

**Description**

plot_solutionpaths Function for plotting results of the solutionpaths function

**Usage**

plot_solutionpaths(results)

**Arguments**

results Results from the solutionpaths function

**Author(s)**

Jocelyn T. Chi

**Examples**

# Generate a test matrix
seed <- 12345
m <- 100
n <- 100
r <- 3
T <- testmatrix(m,n,r,seed=seed)

# Add some noise to the test matrix
plot_spect

```r
E <- 0.1*matrix(rnorm(m*n),m,n)
A <- T + E

# Obtain a vector of unobserved entries
temp <- makeOmega(m,n,percent=0.5)
omega <- temp$omega

# Remove unobserved entries from test matrix
X <- A
X[omega] <- NA

# Make initial model matrix Z and find initial lambda
Z <- matrix(0,m,n)
lambda.start <- init.lambda(X,omega)
lambdaseq_length=20
tol <- 1e-2

ans <- solutionpaths(A,X,Z,omega,lambda.start,tol=tol,
                      liveupdates=FALSE,lambdaseq_length=lambdaseq_length)

# Plot using results from solutionpaths function
plot_solutionpaths(ans)
```

---

**plot_spect**

*MM Algorithm - Plotting the Spectroscopic Signal*

---

**Description**

The `plot_spect` function is used for plotting the spectroscopic signal.

**Usage**

```r
plot_spect(n, y, X, b, nmm)
```

**Arguments**

- `n`: Number of samples
- `y`: Nonnegative response
- `X`: Nonnegative design matrix
- `b`: Nonnegative initial regression vector
- `nmm`: NNM object from `generate_nnm` function

**Author(s)**

Jocelyn T. Chi
Examples

```r
# Setup mixture example
n <- 1e3
p <- 10
nnm <- generate_nnm(n,p)

set.seed(12345)
X <- nnm$x
b <- double(p)
nComponents <- 3
k <- sample(1:p,nComponents,replace=FALSE)
b[k] <- matrix(runif(nComponents),ncol=1)
y <- X%*%b + 0.25*matrix(abs(rnorm(n)),n,1)

plot_spect(n,y,X,b,nnm)
```

---

softhreshold  

**MM Algorithm - Softhreshold Function**

### Description

softhreshold Function for computing the softhreshold

### Usage

```r
softhreshold(x, lambda)
```

### Arguments

- `x`  
  Vector of values to be softhresholded
- `lambda`  
  Softthreshold parameter

### Author(s)

Jocelyn T. Chi

### Examples

```r
x <- seq(-10,10,1)
softhreshold(x,lambda=3)
```
solutionpaths

**MM Algorithm - Find the best fit lambda for a given problem based on an initial guess for lambda**

**Description**

solutionpaths Function for finding the best fit lambda for a given problem based on an initial guess for lambda

**Usage**

```r
solutionpaths(A, X, Z, omega, lambda.start, tol = 1e-04, liveupdates = FALSE, lambdaseq_length = 20)
```

**Arguments**

- `A` Original data matrix (no unobserved entries)
- `X` Data matrix (with unobserved entries)
- `Z` Initial model matrix
- `omega` Vector of unobserved entries in the data matrix X
- `lambda.start` Initial value for lambda
- `tol` (Optional) Tolerance for convergence (Default: 1e-4)
- `liveupdates` (Optional) Set to TRUE to view progress of comparisons. (Default: FALSE)
- `lambdaseq_length` (Optional) Length of lambda sequence for convergence. (Default: 20)

**Author(s)**

Jocelyn T. Chi

**Examples**

```r
# Generate a test matrix
seed <- 12345
m <- 100
n <- 100
r <- 3
T <- testmatrix(m,n,r,seed=seed)

# Add some noise to the test matrix
E <- 0.1*matrix(rnorm(m*n),m,n)
A <- T + E

# Obtain a vector of unobserved entries
temp <- makeOmega(m,n,percent=0.5)
omega <- temp$omega
```
```r
# Remove unobserved entries from test matrix
X <- A
X[omega] <- NA

# Make initial model matrix Z and find initial lambda
Z <- matrix(0, m, n)
lambda.start <- init.lambda(X, omega)
lambdaseq_length = 20
tol <- 1e-2

ans <- solutionpaths(A, X, Z, omega, lambda.start, tol = tol,
                     liveupdates = TRUE, lambdaseq_length = lambdaseq_length)
```

---

**stigler**

*The Diet Problem: “Nutritive Values of Common Foods per Dollar of Expenditure, August 15, 1944”, from George Stigler’s 1945 paper on "The Cost of Subsistence"*

---

**Description**

A dataset of 10 rows and 14 columns describing 1944 prices and nutritional data for 14 food commodities. Obtained from George Stigler’s 1945 paper on "The Cost of Subsistence".

**Arguments**

- **calories**: Calories (in kilocalories) per dollar
- **protein**: Protein (in grams) per dollar
- **calcium**: Calcium (in grams) per dollar
- **iron**: Iron (in milligrams) per dollar
- **vitaminNa**: Vitamin A (in 1000 International Units) per dollar
- **thiamine**: Thiamine (in milligrams) per dollar
- **riboflavin**: Riboflavin (in milligrams) per dollar
- **niacin**: Niacin (in milligrams) per dollar
- **ascorbicAcid**: Ascorbic Acid (in milligrams) per dollar

**Format**

A data frame with 10 rows and 14 columns

**Source**

**Description**

`testmatrix` Function for generating random rank-r matrix

**Usage**

`testmatrix(m, n, r, seed = 123)`

**Arguments**

- `m` Number of rows in matrix to be generated
- `n` Number of columns in matrix to be generated
- `r` Rank of matrix to be generated (r >= 2)
- `seed` Random seed

**Author(s)**

Jocelyn T. Chi

**Examples**

```r
m <- 100
n <- 1000
r <- 5
testmatrix(m,n,r)
```
Index

*Topic datasets
  baltimoreyouth, 2
  engel, 5
  moviebudgets, 15
  movieratings, 16
  nutrition, 17
  stigler, 30

baltimoreyouth, 2

diff_norm, 4

engel, 5
example. alpha, 5
example.quadratic.approx, 6

gdescent, 6
generate_nnm, 9
gettingtothebottom, 10
gettingtothebottom-package
  (gettingtothebottom), 10

init.lambda, 10

makeLambdaseq, 11
makeOmega, 12
makeY, 12
makeZ, 13
matrixcomplete, 14
moviebudgets, 15
movieratings, 16

nnls_mm, 16
nutrition, 17

plot.gradient, 18
plot.iterates, 19
plot.loss, 20
plot_nnm, 21
plot_nnm_coef, 21
plot_nnm_obj, 22

plot_nnm_reconstruction, 23
plot_nnm_truth, 24
plot_softthreshold, 24
plot_solpaths_error, 25
plot_solutionpaths, 26
plot_spect, 27

softthreshold, 28
solutionpaths, 29
stigler, 30

testmatrix, 31