Package 'gettingtothebottom'

August 2, 2014

Title Getting to the Bottom, A Package for Learning Optimization Methods

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

URL http://jocelynchi.com/gettingtothebottom

Version 3.0

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

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

Depends R (>= 3.0.2), ggplot2, grid, Matrix, lpSolve, reshape2

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2014-08-02 23:50:41

R topics documented:

`baltimoreyouth` *Baltimore Youth Indicators - 2010 and 2011*

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

CSA2010	Name of Community Statistical Areas in 2010
eattendXX	Number of Students Ever Enrolled 1st - 5th Grade
mattendXX	Number of Students Ever Enrolled 6th - 8th Grade
heattendXX	Number of Students Ever Enrolled 9th - 12th Grade
eenrolXX	Number of Students Enrolled in 1st - 5th Grade
menrolXX	Number of Students Enrolled in 6th - 8th Grade
henrolXX	Number of Students Enrolled in 9th - 12th Grade
aastudXX	Percent of Students that are African American
wstudXX	Percent of Students that are White (non-Hispanic)
hstudXX	Percent of Students that are Hispanic
abseXX	Percent of 1st-5th Grade Students that are Chronically Absent (Missing at least 20 days)
absmdXX	Percent of 6th-8th Grade Students that are Chronically Absent (Missing at least 20 days)
abshsXX	Percent of 9th-12th Grade Students that are Chronically Absent (Missing at least 20 days)
suspXX	Percentage of Students Suspended or Expelled During School Year
farmsXX	Percentage of Students Receiving Free or Reduced Meals
spedXX	Percentage of Students Enrolled in Special Education Programs
readyXX	Kindergarten School Readiness
3mathXX	Percentage of 3rd Grade Students Passing MSA Math
3readXX	Percentage of 3rd Grade Students Passing MSA Reading
5mathXX	Percentage of 5th Grade Students Passing MSA Math
5readXX	Percentage of 5th Grade Students Passing MSA Reading
8mathXX	Percentage of 8th Grade Students Passing MSA Math
8readXX	Percentage of 8th Grade Students Passing MSA Reading
hsaengXX	Percentage of Students Passing H.S.A. English
hsabioXX	Percentage of Students Passing H.S.A. Biology
hsagovXX	Percentage of Students Passing H.S.A. Government
hsaalgXX	Percentage of Students Passing H.S.A. Algebra
dropXX	High School Dropout/Withdrawl Rate
complXX	High School Completion Rate
sclswXX	Percent of Students Switching Schools within School Year
sclempXX	Percentage of Population aged 16-19 in School and/or Employed
teenbirXX	Teen Birth Rate per 1,000 Females (aged 15-19)
termbirXX	Percent of Births Delivered at Term (37-42 Weeks)
birthwtXX	Percent of Babies Born with a Satisfactory Birth Weight

prenatalXX	Percent of Births Where the Mother Received Early Prenatal Care (First Trimester)
leadtestXX	Number of Children (aged 0-6) Tested for Elevated Blood Lead Levels
ebllXX	Percent of Children (aged 0-6) with Elevated Blood Lead Levels
leadvXX	Percent of Lead Violations per 1,000 Residential Units
tanfXX	Percent of Families Receiving TANF
liquorXX	Liquor Outlet density (per 1,000 Residents)
fastfdXX	Fast Food Outlet Density (per 1,000 Residents)
LifeExpXX	Life Expectancy
mort1_XX	Mortality by Age (Less than 1 year old)
mort14_XX	Mortality by Age (1-14 years old)
mort24_XX	Mortality by Age (15-24 years old)
mort44_XX	Mortality by Age (25-44 years old)
mort64_XX	Mortality by Age (45-64 years old)
mort84_XX	Mortality by Age (65-84 years old)
mort85_XX	Mortality by Age (85 and over)

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

---

check_func *Linear Programming - Check function*

Description

check_func Check function

Usage

```
check_func(r, tau)
```

Arguments

r	A vector to feed into the check function
tau	Tau values to be used in the check function

Examples

```
r <- seq(-1,1,length.out=1e4)
tau <- 0.5
check_func(r,tau)
```

`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

```
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)
```

`engel` *Engel's Law - Engel Food Expenditures Data from the quantreg package for R*

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

---

`example.alpha` *Gradient Descent Algorithm - Plots Depicting Gradient Descent Results in Example 1 Using Different Choices for the Step Size*

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

```
#-------------------------------------------------------
# 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.

#-------------------------------------------------------
# EXAMPLE 2 - Linear Regression & Feature Scaling
#-------------------------------------------------------

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.

#-------------------------------------------------------
# EXAMPLE 3 - Multivariate Linear Regression
#-------------------------------------------------------

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$compl11)
# 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)

#-------------------------------------------------------
# EXAMPLE 4 - Logistic Regression
#-------------------------------------------------------

set.seed(12345)
n <- 100
p <- 10
X <- matrix(rnorm(n*p),n,p)
b <- matrix(rnorm(p),p,1)
e <- 0.5*matrix(rnorm(n),n,1)
z <- X%*%b + e
y <- as.vector((plogis(z) <= runif(n)) + 0)

l <- function(X,y,b) {
  -t(y)%*%(X%*%b) + sum(log(1+exp(X%*%b)))
}
grad_l <- function(X,y,b) {
  -t(X)%*%(y-plogis(X%*%b))
}
alpha = 1/(0.25*svd(cbind(1,X))$d[1]**2)

# Use gradient descent algorithm to solve logistic regression problem
# THE FOLLOWING NOT RUN
```

```
# logistic_ex <- gdescent(l,grad_l,X,y,alpha=alpha,iter=15000)

# Use glm function to solve logistic regression problem
# THE FOLLOWING NOT RUN
# glm(y~X, family=binomial)
```

---

generate_data *Linear Programming - Generate simulated data for LAD regression*

Description

generate_data Function for generating simulated data for LAD regression

Usage

```
generate_data(n = 100, p = 1, sigma = 1, seed = 12345)
```

Arguments

n	Number of data points
p	Number of total covariates
sigma	Standard deviance
seed	Random seed

Author(s)

Jocelyn T. Chi

Examples

```
generate_data(n=1000,p=100)
```

---

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

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

---

generate_sparse_data *Linear Programming - Generate simulated data for sparse quantile regression*

Description

generate_sparse_data Function for generating simulated data for sparse quantile regression

Usage

```
generate_sparse_data(n = 100, p = 20, k = 3, seed = 12345)
```

Arguments

n	Number of data points
p	Number of total covariates
k	Number of true covariates
seed	Random seed

Author(s)

Jocelyn T. Chi

Examples

```
data <- generate_sparse_data(n=100,p=20,k=3)
```

gettingtothebottom *gettingtothebottom*

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

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

Arguments

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

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

init.lambda(X,omega)
```

`ladlp` *Linear Programming - Least absolute deviations (LAD) regression*

Description

`ladlp` Function for performing least absolute deviations using linear programming

Usage

```
ladlp(formula)
```

Arguments

`formula`	Formula for LAD regression (e.g., y~X)

Author(s)

Jocelyn T. Chi

Examples

```
# Simulate data for LAD regression
m <- 1000 # number of observations
n <- 100 # number of variables
reg <- generate_data(m,n)
X <- reg$X
y <- reg$y

# Obtain LAD solution (not run)
# sol_lad <- ladlp(y~X)
# sol_lad
```

makeLambdaseq — *MM Algorithm - Function for making sequence of lambdas for solution paths*

Description

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

Usage

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

Arguments

L	Initial lambda value
lambdaseq_length	Desired length of lambda sequence

Author(s)

Jocelyn T. Chi

Examples

```
makeLambdaseq(11,20)
```

makeOmega — *MM Algorithm - Generate Omega*

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

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

---

makeY *MM Algorithm - Make Y*

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)

Author(s)

Jocelyn T. Chi

Examples

```
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)
```

makeZ *MM Algorithm - Make Z*

Description

makeZ Function for making the Z matrix

Usage

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

Arguments

M	Matrix containing observed entries
lambda	Softthreshold parameter

Author(s)

Jocelyn T. Chi

Examples

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

make_noise *Linear Programming - Generate matrix of noisy data for sparse quantile regression*

Description

make_noise Function for generating matrix of noisy data for sparse quantile regression

Usage

```
make_noise(y, X, p, seed = 12345)
```

Arguments

y	Response vector
X	True predictor
p	Number of desired columns of noise
seed	(optional) Random seed (Default seed=12345)

Author(s)

Jocelyn T. Chi

Examples

```
y <- engel$foodexp
X <- engel$income
noisydata <- make_noise(y,X,10)
head(noisydata)
```

---

matrixcomplete *MM Algorithm - Matrix Completion*

Description

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

Usage

```
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

```
# (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 <- 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 <- 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/.

---

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

```
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))
```

nutrition — *The Diet Problem: "Daily Allowances of Nutrients for a Moderately Active Man (weighing 154 pounds)" from George Stigler's 1945 paper on "The Cost of Subsistence"*

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`	Iron (in milligrams)
`Vitamin.A`	Vitamin A (in 1000 Interntional Units)
`Thiamine`	Thiamine (in milligrams)
`Riboflavin`	Riboflavin (in milligrams)
`Niacin`	Niacin (in milligrams)
`Ascorbic.Acid`	Ascorbin Acid (in milligrams)

Format

A vector of length 9

Source

George J. Stigler, "The Cost of Subsistence", Journal of Farm Economics, Vol. 27, No. 2 (May, 1945), pp. 303-314. http://www.jstor.org/stable/1231810

plot_check — *Linear Programming - Plot Check Function*

Description

`plot_check` Plot Check Function

Usage

```
plot_check(r, taus = taus)
```

Arguments

r	A vector to feed into the check function
taus	A vector of tau values to be used in the check function

Examples

```
r <- seq(-1,1,length.out=1e4)
taus <- c(0.25,0.5,0.75)
plot_check(r,taus)
```

---

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

```
plot_gradient(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%*%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)
```

---

plot_iterates *Gradient Descent Algorithm - Plotting the Iterates*

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%*%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 *Gradient Descent Algorithm - Plotting the Loss Function*

Description

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

Usage

```
plot_loss(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%*%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 loss function
plot_loss(simple_ex)
```

plot_nnm *MM Algorithm - 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 *MM Algorithm - Plotting the NNMLS regression coefficients*

Description

plot_nnm_coef Function for plotting the NNMLS regression coefficients

Usage

```
plot_nnm_coef(nnm_sol)
```

Arguments

nnm_sol	Solution object from nnls_mm function

Examples

```
# 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

```
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

```
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

```
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

```
# 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 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

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

Examples

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

plot_quantreg *Linear Programming - Function for plotting results of bivariate quantile regression.*

Description

plot_quantreg Function for plotting results of bivariate quantile regression.

Usage

```
plot_quantreg(x, y, xlab = "x", ylab = "y", taus = c(0.05, 0.25, 0.5,
  0.75, 0.95))
```

Arguments

x	Variable on the x-axis.
y	Variable on the y-axis.
taus	(optional) Quantiles for plotting, default is 'c(0.05,0.25,0.5,0.75,0.95)'.
xlab	(optional) Default is x.
ylab	(optional) Default is y.

Author(s)

Jocelyn T. Chi

Examples

```
data(baltimoreyouth)
x <- baltimoreyouth$teenbir10 # Teen births
y <- baltimoreyouth$compl10 # Highschool completion rate
plot_quantreg(x,y,xlab="Teen Births per 1000 Females (aged 15-19)",
     ylab="High School Completion Rates")
```

---

plot_quantreg_noisy *Linear Programming - Function for plotting results of sparse quantile regression*

Description

plot_noisyquantreg Function for plotting results of bivariate quantile regression with noise.

Usage

```
plot_quantreg_noisy(y, X, taus = c(0.05, 0.25, 0.5, 0.75, 0.95),
  lambdas = c(0, 0.5, 1, 1.5))
```

Arguments

y	Response variable
X	Noisy matrix (includes x)
taus	(optional) Quantiles for plotting, default is 'c(0.05,0.25,0.5,0.75,0.95)'
lambdas	(optional) Regularization parameters (Default is lambdas=c(0,0.5,1,1.5).)

Author(s)

Jocelyn T. Chi

Examples

```
data <- generate_sparse_data(n=100,p=20,k=3)
y <- data$y
X <- data$X

lambdas=c(0,1,5,7)
plot_quantreg_noisy(y,X,lambdas=lambdas) # Not run
```

`plot_softhreshold` *MM Algorithm - Plot the Softhreshold Function*

Description

`plot_softhreshold` Function for plotting the softhreshold function

Usage

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

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` *MM Algorithm - Function for plotting the imputed values against the truth for minimum error solution*

Description

`plot_solpaths_error` Function for plotting the imputed values against the truth for minimium 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,lambda.start,tol=tol,
   liveupdates=FALSE,lambdaseq_length=lambdaseq_length)

plot_solpaths_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
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

plot_spect Function for plotting the spectroscopic signal

Usage

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

Arguments

n	Number of samples
nnm	NNM object from generate_nnm function
y	Nonnegative response
X	Nonnegative design matrix
b	Nonnegative initial regression vector

Author(s)

Jocelyn T. Chi

Examples

```
# 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)
```

---

quantreg *Linear Programming - Quantile regression*

Description

quantreg quantreg is used to fit quantile regression models.

Usage

```
quantreg(formula, tau = 0.5, lambda = 0)
```

Arguments

formula	An object of class "formula" (e.g., y~X)
tau	(optional) Quantile for quantile regression. Default is 0.5 but can also be a vector (e.g., tau = c(0.25,0.5,0.75)).
lambda	(optional) Regularization parameter. Default is lambda=0 (i.e., no regularization).

Author(s)

Jocelyn T. Chi

Examples

```
set.seed(12345)
n <- 20
p <- 20
X <- matrix(rnorm(n*p),n,p)
b0 <- double(p)
k <- 4
b0[sample(1:p,k,replace=FALSE)] <- 10*rnorm(k)
y <- X%*%b0 + 0.1*rnorm(n)

lambda <- 0
tau <- c(0.05,0.25,0.5,0.75,0.95)
sol <- quantreg(y~X,tau,lambda)
coef(sol)
```

---

quantreglp — *Linear Programming - Linear programming solver for quantile regression*

Description

quantreglp Function for performing quantile regression using linear programming

Usage

```
quantreglp(y, X, tau = 0.5, lambda = 0)
```

Arguments

y	An mx1 vector containing the response variables in the model
X	An nxp matrix containing the predicting variables in the model
tau	(optional) quantile
lambda	(optional) regularization parameter

Author(s)

Jocelyn T. Chi

Examples

```
set.seed(12345)
n <- 20
p <- 20
X <- matrix(rnorm(n*p),n,p)
b0 <- double(p)
k <- 4
b0[sample(1:p,k,replace=FALSE)] <- 10*rnorm(k)
y <- X%*%b0 + 0.1*rnorm(n)

lambda <- 2
tau <- 0.5
sol <- quantreglp(y,X,tau,lambda)
```

---

softhreshold *MM Algorithm - Softhreshold Function*

Description

`softhreshold` Function for computing the softhreshold

Usage

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

Arguments

`x`	Vector of values to be softhresholded
`lambda`	Softhreshold parameter

Author(s)

Jocelyn T. Chi

Examples

```
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

```
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

```
# 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,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
`Vitamin.A`	Vitamin A (in 1000 Interntional Units) per dollar
`Thiamine`	Thiamine (in milligrams) per dollar
`Riboflavin`	Riboflavin (in milligrams) per dollar
`Niacin`	Niacin (in milligrams) per dollar
`Ascorbic.Acid`	Ascorbin Acid (in milligrams) per dollar

Format

A data frame with 10 rows and 14 columns

Source

George J. Stigler, "The Cost of Subsistence", Journal of Farm Economics, Vol. 27, No. 2 (May, 1945), pp. 303-314. http://www.jstor.org/stable/1231810

`testmatrix` *MM Algorithm - Generate Test Matrix*

Description

`testmatrix` Function for generating random rank-r matrix

Usage

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

Arguments

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

Author(s)

Jocelyn T. Chi

Examples

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

Index