Package ‘jcode64’

February 28, 2017

Type    Package
Title   A set of misc R codes written by Joe Atwood
Version 3.10
Date 2015-02-20
Author Joe Atwood
Maintainer Joe Atwood <jatwood@montana.edu>
Depends doBy,maptools,numDeriv,MASS,chron
Description A set of misc code commonly used by Joe Atwood
License GPL-2
LazyLoad yes
Archs i386, x64

R topics documented:

jcode64-package .............................................. 2
A2SM ......................................................... 3
autosim .................................................... 4
bearcalc ..................................................... 5
bigeight.na ............................................... 6
bigseven.na ............................................... 7
bigsix.na .................................................. 8
class_df ................................................... 9
closeDist ................................................. 10
colorum.gplots .......................................... 11
cor2cov .................................................. 12
CtyData .................................................... 13
distbearcalc ............................................. 14
distcalc ................................................... 15
distmat .................................................... 16
gradum .................................................... 17
ImanConover .............................................. 18
ImanConoverYcZc ......................................... 19
int2real_df ............................................... 20
JBpars ..................................................... 21
jmonth ..................................................... 22
lagum ...................................................... 23
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>length.na</td>
<td>24</td>
</tr>
<tr>
<td>longlatdistbear</td>
<td>25</td>
</tr>
<tr>
<td>longlatkeep</td>
<td>27</td>
</tr>
<tr>
<td>lookup</td>
<td>28</td>
</tr>
<tr>
<td>maxgdhess</td>
<td>28</td>
</tr>
<tr>
<td>meanOA</td>
<td>34</td>
</tr>
<tr>
<td>moveAVG</td>
<td>34</td>
</tr>
<tr>
<td>OavgM</td>
<td>35</td>
</tr>
<tr>
<td>pickum</td>
<td>36</td>
</tr>
<tr>
<td>pointslope</td>
<td>37</td>
</tr>
<tr>
<td>PolyfindplotM</td>
<td>38</td>
</tr>
<tr>
<td>pullMdata</td>
<td>39</td>
</tr>
<tr>
<td>pvalx</td>
<td>40</td>
</tr>
<tr>
<td>Pvalx_V</td>
<td>41</td>
</tr>
<tr>
<td>quicksort</td>
<td>41</td>
</tr>
<tr>
<td>radcalc</td>
<td>42</td>
</tr>
<tr>
<td>rbetaN</td>
<td>43</td>
</tr>
<tr>
<td>rho2cor</td>
<td>44</td>
</tr>
<tr>
<td>rnormJ</td>
<td>45</td>
</tr>
<tr>
<td>RunMA</td>
<td>46</td>
</tr>
<tr>
<td>R_A2SM</td>
<td>47</td>
</tr>
<tr>
<td>R_SM2A</td>
<td>48</td>
</tr>
<tr>
<td>seconds.local</td>
<td>49</td>
</tr>
<tr>
<td>sin_d</td>
<td>49</td>
</tr>
<tr>
<td>SM2A</td>
<td>50</td>
</tr>
<tr>
<td>summaryDF</td>
<td>51</td>
</tr>
<tr>
<td>ulong</td>
<td>52</td>
</tr>
<tr>
<td>USMapsS48</td>
<td>53</td>
</tr>
<tr>
<td>wait</td>
<td>53</td>
</tr>
<tr>
<td>waittalk</td>
<td>54</td>
</tr>
<tr>
<td>ylddata</td>
<td>55</td>
</tr>
</tbody>
</table>

**Index**

<table>
<thead>
<tr>
<th>Package</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>jcode64-package</td>
<td>56</td>
</tr>
</tbody>
</table>

**Description**

A set of misc code developed by Joe Atwood

**Details**

- **Package**: jcode64
- **Type**: Package
- **Version**: 3.10
- **Date**: 2017-02-20
- **License**: GPL-2
- **LazyLoad**: yes
An overview of how to use the package, including the most important functions

**Author(s)**

Joe Atwood

---

**A2SM**

*Uses fortran to convert a real matrix to a sparse matrix form*

**Description**

Converts a real matrix to a sparse matrix form

**Usage**

```r
A2SM(A, SMM='CRI', eps=1e-9, CINDEX=0)
```

**Arguments**

- **A**: real matrix of dimension nr x nc
- **SMM**: Sparse matrix method. 'CRI'=both row and col index, 'CMO' = Column major order, 'RMO' = row major order
- **eps**: if abs(aij) < eps treated as zero
- **CINDEX**: 1 => use C zero indexes, 1 => use R/Fortran 1 indexes

**Value**

- **A**: matrix
- **nr**: number of rows in A
- **nc**: number of columns in A
- **SMM**: Sparse matrix method
- **CINDEX**: "C" numbering index
- **ac**: non-zero elements in A
- **ia**: row indexes
- **ja**: column matrices

**Author(s)**

Joe Atwood

**Examples**

```
#(A=matrix(c(1,0,2,0,0,0,0,0,5,0,6),3,4,byrow=T))
#A2SM(A=A, SMM='CRI')
#A2SM(A=A, SMM='CMO')
#A2SM(A=A, SMM='RMO')
```
**autosim**

_Simulates vector autocorrelated disturbance vectors_

**Description**

Simulates vector autocorrelated disturbance vectors

**Usage**

```r
autosim(U,R,a)
```

**Arguments**

- **U**
  - real array of dimension (nobs, number of equations)
- **R**
  - Array of dimension (eq.number, eq.number, number of vector autocorrelation lags)
- **a**
  - Vector of intercept terms

**Value**

Returns a simulated Ucor or difference equation matrix of dimension (nobs, eq.number)

**Author(s)**

Joe Atwood

**Examples**

```r
# Examples
nobs=10000
neqs=4
U=matrix(rnorm(nobs*neqs),nobs,neqs)
R1=array(0,c(neqs,neqs,1))
(diag(R1[,1])=0.9)
Ucor1=autosim(U,R1)
R2=array(0,c(neqs,neqs,2))
(diag(R2[,1])=1.50)
(diag(R2[,2])=-0.90)
Ucor2=autosim(U,R2)
DE1=autosim(U,R2,a=c(10,20,30,40))
E=matrix(0,nobs,neqs)
DE2=autosim(E,R2,a=c(10,20,30,40))
```
bearcalc

Computes the bearing from long-lat0 to long-lat1

Description
Computes the bearing (from North) from long-lat0 to long-lat1.

Usage
bearcalc(long0, lat0, long1, lat1, rads = "N")

Arguments
long0 Longitude of initial point
lat0 Latitude of initial point
long1 Longitude of distant point (or points)
lat1 Longitude of distant point (or points)
rads Long-lat are in radians ("Y") or degrees ("N")

Value
Bearing from North (in radians if rads='Y') (or degrees if rads='N') from initial point to distant point (points)

Author(s)
Joe Atwood

Examples

# long-lat of Bozeman, Montana = -111.047,45.68
# long-lat of Seattle, Washington = -122.35,47.62
# long-lat of Atlanta, Georgia = -84.39,33.75

bearcalc(-111.047,45.68,-122.35,47.62)
#[1] 71.8899
# Bearing from Bozeman to Seattle is 71.89 degrees counter clockwise from North

# Compute bearing from Bozeman to both Seattle and Atlanta
long1=c(-122.35,-84.39)
lat1=c(47.62,33.75)
bearcalc(-111.047,45.68,long1,lat1)
#[1] 71.8899 248.9611
# Bearing from Bozeman to Atlanta is 248.96 degrees counter clockwise from North
Function to compute summary values while deleting NA’s and possibly zeros

Description

Functions to compute summary values while deleting NA’s and possibly zeros.

Usage

bigeight.na(df1,q1=0.10,q2=0.90, skipzero = "N", Tpose='N')

Arguments

- df1: data.frame or matrix of values
- q1: first quantile to report
- q2: second quantile to report
- skipzero: N=include zeros in calcs Y=exclude zeros in calcs
- Tpose: N=do not transpose summary table Y=transpose summary table

Value

returns corresponding values for numeric columns only

Author(s)

Joe Atwood

Examples

```r
set.seed(2010)
M1=as.data.frame(matrix(rnorm(1000),200,5))
bigeight.na(M1,q1=0.05,q2=0.95)
# V1 V2 V3 V4 V5
#Min -3.3521028 -2.45626548 -2.54567945 -2.25896357 -3.00011051
#Q1 -1.5895583 -1.96254025 -1.63694281 -1.69154387 -1.73053592
#Quartile1 -0.8697535 -0.81104527 -0.54553587 -0.71105677 -0.67842843
#Median -0.0910531 0.01898896 0.05293012 -0.02809267 0.06219167
#Mean -0.1512827 -0.04009036 0.02263271 0.01730692 -0.01060624
#Quartile2 0.5554403 0.75341160 0.69609078 0.75036167 0.67842843
#Q95 1.4007719 1.46689375 1.39250250 1.54234982 1.45674245
#Max 2.4432845 2.16007120 2.33807078 3.10254199 2.40308570
```

M2=M1
M2[,1]=as.character(M2[,1])
bigeight.na(M2)
### bigseven.na

Function to compute summary values while deleting NA's and possibly zeros

---

**Description**

Functions to compute summary values while deleting NA's and possibly zeros

**Usage**

```r
bigseven.na(df1, q1 = 0.25, q2 = 0.75, skipzero = "N", Tpose = "Y")
```

**Arguments**

- `df1`: data.frame or matrix of values
- `q1`: first quantile to report
- `q2`: second quantile to report
- `skipzero`: `N`=include zeros in calcs `Y`=exclude zeros in calcs
- `Tpose`: `N`=do not transpose summary table `Y`=transpose summary table

**Value**

returns corresponding values for numeric columns only

**Author(s)**

Joe Atwood

---

<table>
<thead>
<tr>
<th>#</th>
<th>V2</th>
<th>V3</th>
<th>V4</th>
<th>V5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>-2.45626548</td>
<td>-2.54567945</td>
<td>-2.25896357</td>
<td>-3.00011051</td>
</tr>
<tr>
<td>Q10</td>
<td>-1.38961514</td>
<td>-1.22504631</td>
<td>-1.39941308</td>
<td>-1.27168723</td>
</tr>
<tr>
<td>Quartile1</td>
<td>-0.81104527</td>
<td>-0.54553587</td>
<td>-0.71105677</td>
<td>-0.67842843</td>
</tr>
<tr>
<td>Median</td>
<td>0.01898896</td>
<td>0.05293012</td>
<td>-0.02809267</td>
<td>0.06219167</td>
</tr>
<tr>
<td>Mean</td>
<td>-0.04009036</td>
<td>0.02263271</td>
<td>0.01730692</td>
<td>-0.01060624</td>
</tr>
<tr>
<td>Quartile2</td>
<td>0.75341160</td>
<td>0.69609068</td>
<td>0.75036167</td>
<td>0.67340246</td>
</tr>
<tr>
<td>Q90</td>
<td>1.15618422</td>
<td>1.19710266</td>
<td>1.35163289</td>
<td>1.13729509</td>
</tr>
<tr>
<td>Max</td>
<td>2.16007120</td>
<td>2.33807078</td>
<td>3.10254199</td>
<td>2.40308570</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>#</th>
<th>Min</th>
<th>Quartile1</th>
<th>Median</th>
<th>Mean</th>
<th>Quartile2</th>
<th>Q90</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>V2</td>
<td>-2.456265</td>
<td>-1.389615</td>
<td>0.01898896</td>
<td>-0.04009036</td>
<td>0.75341160</td>
<td>1.15618422</td>
<td>2.16007120</td>
</tr>
<tr>
<td>V3</td>
<td>-2.545679</td>
<td>-1.225046</td>
<td>0.05293012</td>
<td>0.02263271</td>
<td>0.69609068</td>
<td>1.19710266</td>
<td>2.338071</td>
</tr>
<tr>
<td>V4</td>
<td>-2.258964</td>
<td>-1.399413</td>
<td>-0.02809267</td>
<td>0.01730692</td>
<td>0.75036167</td>
<td>1.35163289</td>
<td>3.102542</td>
</tr>
<tr>
<td>V5</td>
<td>-3.000111</td>
<td>-1.271687</td>
<td>0.06219167</td>
<td>-0.01060624</td>
<td>0.67340246</td>
<td>1.13729509</td>
<td>2.403086</td>
</tr>
</tbody>
</table>
Examples

```r
set.seed(2010)
M1=as.data.frame(matrix(rnorm(1000),200,5))
bigseven.na(M1,q1=0.05,q2=0.95)
# V1 V2 V3 V4 V5
#Min -3.3521028 -2.45626548 -2.54567945 -2.25896357 -3.00011051
#Q5 -1.5895583 -1.96254025 -1.63694281 -1.69154387 -1.73053592
#Median -0.0910531 0.01898896 0.05293012 -0.02809267 0.06219167
#Mean -0.1512827 -0.04009036 0.02263271 0.01730692 -0.01060624
#Stdev 0.9793965 1.00990177 0.94822607 1.04224635 0.98500197
#Q95 1.4007719 1.46689375 1.39250250 1.54234982 1.45674245
#Max 2.4432845 2.16007120 2.33807078 3.10254199 2.40308570

M2=M1
M2[,1]=as.character(M2[,1])
bigseven.na(M2)
# V2 V3 V4 V5
#Min -2.45626548 -2.54567945 -2.25896357 -3.00011051
#Q25 -0.81104527 -0.54553587 -0.71105677 -0.67842843
#Median 0.01898896 0.05293012 -0.02809267 0.06219167
#Mean -0.04009036 0.02263271 0.01730692 -0.01060624
#Stdev 1.00990177 0.94822607 1.04224635 0.98500197
#Q75 0.75341160 0.69609068 0.75036167 0.67340246
#Max 2.16007120 2.33807078 3.10254199 2.40308570

bigseven.na(M2,Tpose='Y')
# Min Q25 Median Mean Stdev Q75 Max
#V2 -2.456265 -0.8110453 0.01898896 0.04009036 1.009902 0.7534116 2.160071
#V3 -2.545679 -0.5455359 0.05293012 0.02263271 0.948226 0.6960907 2.338071
#V4 -2.258964 -0.7110568 -0.02809267 0.01730692 1.042246 0.7503617 3.102542
#V5 -3.000111 -0.6784284 0.06219167 -0.01060624 0.985002 0.6734025 2.403086
```

---

**bigsix.na**

*Function to compute summary values while deleting NA's and possibly zeros*

**Description**

Functions to compute summary values while deleting NA's and possibly zeros

**Usage**

```r
bigsix.na(df1,q1=0.25,q2=0.75, skipzero = "N",Tpose='N')
```

**Arguments**

- **df1**: data.frame or matrix of values
- **q1**: first quantile to report
q2 second quantile to report
skipzero N=include zeros in calcs Y=exclude zeros in calcs
Tpose N=do not transpose summary table Y=transpose summary table

Value
returns corresponding values for numeric columns only

Author(s)
Joe Atwood

Examples
set.seed(2010)
M1=as.data.frame(matrix(rnorm(1000),200,5))
bigsix.na(M1,q1=0.05,q2=0.95)
# V1 V2 V3 V4 V5
#Min -3.3521028 -2.45626548 -2.54567945 -2.25896357 -3.00011051
#Q5 -1.5895583 -1.96254025 -1.63694281 -1.69154387 -1.73053592
#Median -0.0910531 0.01898896 0.05293012 -0.02809267 0.06219167
#Mean -0.1512827 -0.04009036 0.02263271 0.01730692 -0.01060624
#Q95 1.4007719 1.46689375 1.39250250 1.54234982 1.45674245
#Max 2.4432845 2.16007120 2.33807078 3.10254199 2.40308570
M2=M1
M2[,1]=as.character(M2[,1])
bigsix.na(M2)
# V2 V3 V4 V5
#Min -2.45626548 -2.54567945 -2.25896357 -3.000111051
#Q25 -0.81104527 -0.54553587 -0.71105677 -0.67842843
#Median 0.01898896 0.05293012 -0.02809267 0.06219167
#Mean -0.04009036 0.02263271 0.01730692 -0.01060624
#Q75 0.75341160 0.69080668 0.75036167 0.67340246
#Max 2.16007120 2.33807078 3.10254199 2.40308570
bigsix.na(M2,Tpose=’Y’)
# V2 V3 V4 V5
#Min -2.456265 -0.8110453 0.01898896 -0.0400908 0.75341160 2.160071
#Q25 -0.545536 -0.7110568 -0.02809267 0.01730692 0.75036167 3.102542
#Median 0.05293012 0.02263271 0.69080668 0.67340246
#Mean -0.04009036 0.01898896 0.05293012 0.02263271 0.01730692 0.6734025
#Q75 -0.04009036 0.02263271 0.01730692 0.01060624 0.01060624 0.6734025
#Max 2.16007120 2.33807078 3.10254199 2.40308570

class_df

Check classes of columns in data.frame

Description
Check classes of columns in data.frame
Usage
class_df(df0)

Arguments
df0 dataframe

Value
returns corresponding dataframe

Author(s)
Joe Atwood

Examples
## not run
#data(CtyData)
#class_df(CtyData)
#tmp=int2real_df(CtyData,c('sfips'))
#class_df(tmp)

closeDist  Find closest locations in two sets of locations

Description
Find closest locations in second data set for each location in first data set

Usage
closeDist(id0,LGLT0,id2,LGLT2,nclose,dout,rads)

Arguments
id0 integer vector of id’s for first or origin data
LGLT0 matrix or data.frame of long-lat positions for origin data
id2 integer vector of id’s for second or destination data
LGLT2 matrix or data.frame of long-lat positions for destination data
nclose number of closest locations to find
dout distance below which distances will be treated as zero and not included in closest list
rads 'T' -> long-lat in radians 'F' -> long-lat in degrees

Value
returns a list containing: (id0,LGLT0,id2,LGLT2,nclose,dout,rads) and CID = id’s of closest nclose locations CLONG = longitudes of closest nclose locations CLAT = latitudes of closest nclose locations CDIST = distance to closest nclose locations
Author(s)

Joe Atwood

Examples

#########################################################################
## Not Run
## long-lat of Seattle Washington = -122.35,47.62
## long-lat of Bozeman Montana = -111.047,45.68
## long-lat of Atlanta, Georgia = -84.39,33.75
## location off Georgia Coast = -81.09,31.35
# require(jcode64)
#long0=c(-122.35,-111.047,-84.39,-81.09)
#lat0=c(47.62,45.68,33.75,31.35)
#id0=1:4
#LGLT0=data.frame(cbind(long0,lat0))
#id2=id0
#LGLT2=LGLT0
#
#(tmp=closeDist(id0,LGLT0,id2,LGLT2,nclose=2,dout=0.001,rads='F'))
#
###


colorum.gplots

Compute color gradients and values for possible legend plotting

Description

Compute color gradients and values for possible legend plotting

OUR THANKS to the authors of the "gplots" package

Usage

colorum.gplots(scores,ncoLs,coL1,coL2,coL3,Nlgd,Lround,scoremin,scoremax)

Arguments

scores Values to be plotted
ncoLs Number of color gradients to use
coL1 Color at lower end of gradient
coL2 Color at middle of gradient
coL3 Color at upper end of gradient
Nlgd Number of color grids in legend
Lround Rounding for legend text
scoremin Minimum score if truncation desired
scoremax Maximum score if truncation desired
cor2cov

Convert correlations and variance terms to covariance matrix ...

Description

Convert correlations and variance terms to covariance matrix

Usage

cor2cov(R, V)

Arguments

R  Correlation matrix
V  Covariance matrix or vector

Value

Covariance matrix

Author(s)

Joe Atwood
Examples

R=matrix(0.5,5,5)
diag(R)=1
R
(V=diag((5:1)^2))
(V2=cor2cov(R,V))
cov2cor(V2)

CtyData

US County long-lat data

Description

US County long-lat data

Usage

data(CtyData)

Format

A data frame with 3139 observations on the following 17 variables.
sfips  state fips code
name   county names
state_name state names
state_fips state fips code
cnty_fips county fips code
fips   county fips code
index  mapping index
cfips  county fips number
xmin  min long
xavg  avg long
xmax  max long
ymin  min lat
yavg  avg lat
ymax  max lat
long  longitude of county centroid
lat   latitude of county centroid
sabbr state abbreviations

Examples

data(CtyData)
distbearcalc

Computes the distance and bearing from long-lat0 to long-lat1

Description

Computes the distance (miles) and bearing (from North) from long-lat0 to long-lat1.

Usage

distbearcalc(long0, lat0, long1, lat1, rads = "N")

Arguments

long0 Longitude of initial point
lat0 Latitude of initial point
long1 Longitude of distant point (or points)
lat1 Longitude of distant point (or points)
rads Long-lat are in radians ("Y") or degrees ("N")

Value

distance distance (in miles) from initial point to distant point (points)
bear bearing from North (in radians if rads='Y') (or degrees if rads='N') from initial point to distant point (points)

Author(s)

Joe Atwood

Examples

#############################################################
# long-lat of Bozeman Montana = -111.047,45.68
# long-lat of Seattle Washington = -122.35,47.62
# long-lat of Atlanta, Georgia = -84.39,33.75
#
# Compute distance and bearing from Bozeman, Montana to Seattle, Washington

distbearcalc(-111.047,45.68,-122.35,47.62)

#$distance
#[1] 552.6337 # great circle distance from Bozeman to Seattle is 552.6 miles

#$bear
# [1] 71.8899 # Bearing from Bozeman to Seattle is 71.89 degrees counter clockwise from North

#############################################################
# Compute distance and bearing from Bozeman to both Seattle and Atlanta
long1=c(-122.35,-84.39)
lat1=c(47.62,33.75)

distbearcalc(-111.047,45.68,long1, lat1)
# The distance and bearing from Bozeman to Atlanta is
# 1629.46 miles and 248.96 degrees counter clockwise from North

## distcalc

**Description**

Computes the distance (miles) and bearing (from North) from long-lat0 to long-lat1.

**Usage**

```
distcalc(long0, lat0, long1, lat1, rads = "N")
```

**Arguments**

- `long0`  
  Longitude of initial point
- `lat0`  
  Latitude of initial point
- `long1`  
  Longitude of distant point (or points)
- `lat1`  
  Longitude of distant point (or points)
- `rads`  
  Long-lats are in radians ("Y") or degrees ("N")

**Value**

distance (in miles) from point 0 to point (points) 1

**Author(s)**

Joe Atwood

**Examples**

```
# long-lat of Bozeman Montana = -111.047,45.68
# long-lat of Seattle Washington = -122.35,47.62
# long-lat of Atlanta, Georgia = -84.39,33.75

# Compute distance from Bozeman, Montana to Seattle, Washington

distcalc(-111.047,45.68,-122.35,47.62)
```
#[1] 552.6337
# Distance from Bozeman to Seattle is about 552.6 miles

# Compute distance and bearing from Bozeman to both Seattle and Atlanta
long1=c(-122.35,-84.39)
lat1=c(47.62,33.75)

distcalc(-111.047,45.68,long1,lat1)

# [1] 552.6337 1629.4640
# Distance from Bozeman to Atlanta is 1629.46 miles

---

**distmat**  
*Computes distance matrix and stacked distances*

**Description**

Computes distance matrix and stacked distances from each point in vector to other points in vector

**Usage**

```r
distmat(x, y, ids = 1:length(x), zerodist = 0.1)
```

**Arguments**

- `x` longitude of points vector
- `y` latitude of points vector
- `ids` vector ids
- `zerodist` distances below zerodist are set to zero

**Value**

- `dstack` Stacked distances
- `dmat` Matrix of distances
- ...

**Author(s)**

Joe Atwood

**Examples**

```r
data(CtyData)
us=CtyData[,c('sfips','fips','long','lat')]
mt=subset(us,sfips==30)
mt=mt[order(mt$fips),]
distmat(mt$long,mt$lat,mt$sfips)
```
Functions to compute gradient, left-hand gradient, and right-hand gradient or hessian

Description
Compute gradient (gradum)
Compute left-hand gradient (gradn)
Compute right-hand gradient (gradp)
Compute numerical hessian (hessum)

Usage
gradum(obj, b, epsmin = 1e-08, ...)
gradn(obj, b, epsmin = 1e-08, ...)
gradp(obj, b, epsmin = 1e-08, ...)
hessum(obj, b, epsmin = 1e-08, ...)

Arguments
obj objective function
b parameters
epsmin epsilon tolerance
... other arguments or data sent to the objective function

Value
returns corresponding gradient vector

Author(s)
Joe Atwood

Examples
#examples of gradient and hessian functions
function(x) {
}
b=c(1,1)
b2=c(12,6)

gradum(obj,b)
hessum(obj,b)
gradum(obj,b2)
hessum(obj,b2)
library(numDeriv)
grad(obj,b)
hessian(obj,b)
grad(obj,b2)
hessian(obj,b2)

library(nlme)
fdHess(b,obj)

---

**ImanConover**  
**Iman-Conover Process**

**Description**

Function implements Iman-Conover process. The process reorders the columns in matrix "yi" to have the same rank order as a multivariate normal matrix of the same dimension and having Correlation matrix "sigma"

**Usage**

ImanConover(yi, sigma, Icor='F')

**Arguments**

- **yi**  
  Matrix of possible independent marginal observations

- **sigma**  
  Correlation matrix of dimension equal number of columns in yi

- **Icor**  
  Convert the possible rescaled sigma matrix to a correlation matrix

**Value**

Returns correlated matrix yc with same column rank order as multivariate normal matrix with correlation matrix sigma

**Author(s)**

Joe Atwood

**References**


**Examples**

```
set.seed(2009)
U=matrix(runif(4000),1000,4)
sigma=matrix(0.75,4,4)
diag(sigma)=1
sigma
  #[1,] 1.00 0.75 0.75 0.75
```
Description

Function implements Iman-Conover process. The process reorders the columns in matrix "yi" to have the same rank order as a multivariate normal matrix of the same dimension and having Correlation matrix "sigma". Returns a list with Yc and normal correlated matrix Zc

Usage

ImanConoverYcZc(yi,sigma,Icor='F')

Arguments

yi Matrix of possible independent marginal observations
sigma Correlation matrix of dimension equal number of columns in yi
Icor Convert the possible rescaled sigma matrix to a correlation matrix

Value

Returns a list with Yc=correlated matrix yc with same column rank order as multivariate normal matrix with correlation matrix sigma and Zc=matrix of multivariate normal variates with exact covariance sigma sigma=the possible rescaled covariance matrix

Author(s)

Joe Atwood

References

int2real_df

Functions to convert columns in data.frame from integer class to real class

Description

Functions to convert columns in data.frame from integer class to real class

Usage

int2real_df(df0, cnames=NULL)

Arguments

df0 : dataframe
cnames : optional list of columns to convert to real

Value

returns corresponding dataframe

Author(s)

Joe Atwood
**JBpars**

*Functions to compute Johnson SB distribution values given the bounds and k1 and k2*

**Description**

Functions to compute Johnson SB distribution values given teh bounds k1 and k2

**Usage**

```r
JBpars(x, k1=0, k2=1)
```

```r
rJB(n, Jmn, Jsd, k1=0, k2=1)
```

```r
pJB(q, Jmn, Jsd, k1=0, k2=1)
```

```r
qJB(p, Jmn, Jsd, k1=0, k2=1)
```

```r
dJB(x, Jmn, Jsd, k1=0, k2=1)
```

**Arguments**

- `x`: Vector of numeric data
- `k1`: lower bound on x
- `k2`: upper bound on x
- `n`: number of observations
- `Jmn`: mean of transformed JSB observations
- `Jsd`: sd of transformed JSB observations

**Value**

returns corresponding values

**Author(s)**

Joe Atwood
Examples

```r
set.seed(2011)
nobs=20
(x=rnorm(nobs))
(k1=min(x)-.1)
(k2=max(x)+.1)
(tmp=JBpars(x,k1=k1,k2=k2))
(Jmn=tmp$Jmn)
(Jsd=tmp$Jsd)

rJB(nobs,Jmn=Jmn,Jsd=Jsd,k1=k1,k2=k2)
pJB(0,Jmn=Jmn,Jsd=Jsd,k1=k1,k2=k2)
qJB(0.5,Jmn=Jmn,Jsd=Jsd,k1=k1,k2=k2)
dJB(x,Jmn=Jmn,Jsd=Jsd,k1=k1,k2=k2)
```

jmonth

Compute julian month from year0

Description

Compute julian months from year0

Usage

```
jmonth(mo,yr)
mo.yr(jmo)
jbimonth(mo,yr,day)
bimo.yr(jbimo)
```

Arguments

- `mo` vector of months
- `yr` vector of years
- `jmo` vector of julian months

Value

julian months or months years – See examples below

Author(s)

Joe Atwood
lagum

Generate a non-time series lagged vector or matrix

Description
lagum generates a lagged vector lagumMat generates a lagged matrix

Usage
lagum(x, lagum = 1)
lagumMat(x, lagum=2)

Arguments
x vector of data to be lagged
lagum specifies the lag length

Value
lagum returns a vector lagumMat returns a lagged matrix – See examples below

Author(s)
Joe Atwood
Examples

#x=1:10
lagum(x,1)
# [1] NA 1 2 3 4 5 6 7 8 9
lagum(x,3)
# [1] NA NA NA 1 2 3 4 5 6 7
uplag(x)
# [1] 2 3 4 5 6 7 8 9 10 NA

lagumMat(x,3)
# [1,] 1 NA NA NA
# [2,] 2 1 NA NA
# [3,] 3 2 1 NA
# [4,] 4 3 2 1
# [5,] 5 4 3 2
# [6,] 6 5 4 3
# [7,] 7 6 5 4
# [8,] 8 7 6 5
# [9,] 9 8 7 6
#[10,] 10 9 8 7

lagum(x,-2)
# [1] 3 4 5 6 7 8 9 10 NA NA
lagumMat(x,-2)
# [,1] [,2] [,3]
# [1,] 1 2 3
# [2,] 2 3 4
# [3,] 3 4 5
# [4,] 4 5 6
# [5,] 5 6 7
# [6,] 6 7 8
# [7,] 7 8 9
# [8,] 8 9 10
# [9,] 9 10 NA
#[10,] 10 NA NA

length.na(x, skipzero = "N")
mean.na(x, skipzero = "N")
min.na(x, skipzero = "N")

---

*length.na*

Functions to compute values while deleting NA's and possibly zeros

Description

Functions to compute values while deleting NA's and possibly zeros

Usage

length.na(x, skipzero = "N")
mean.na(x, skipzero = "N")
min.na(x, skipzero = "N")
Arguments

x Vector of numeric data
skipzero N=include zeros in calcs Y=exclude zeros in calcs

Value

returns corresponding values

Author(s)

Joe Atwood

Examples

x=rnorm(100)
x[1:10]=NA
mean(x, na.rm=TRUE)
mean.na(x)
max.na(x)
min.na(x)
sum.na(x)

Description

Computes long-lat position(s) for given distance(s) and bearing(s) from an initial location long0, lat0

Usage

longlatdistbear(long0, lat0, distance, bear, rads = "N")
Arguments

- `long0`  Longitude of initial point
- `lat0`  Latitude of initial point
- `distance`  Distance from initial point to given point(s)
- `bear`  Bearing (counter clockwise from North) from initial point to given point(s)
- `rads`  rads="N" indicates longitude and latitude are in (and returned in) degrees rads="Y" indicates longitude and latitude are in (and returned in) radians

Value

returns dataframe containing long-lat positions for points distant from (long0,lat0)

Author(s)

Joe Atwood

Examples

```r
# long-lat of Bozeman Montana = -111.047,45.68
# long-lat of Seattle Washington = -122.35,47.62
# long-lat of Atlanta, Georgia = -84.39,33.75

# Compute distance and bearing from Bozeman, Montana to Seattle, Washington
distbearcalc(-111.047,45.68,-122.35,47.62)
#$distance
#[1] 552.6337
#$bear
#[1] 71.8899

# Compute distance and bearing from Bozeman to both Seattle and Atlanta
long1=c(-122.35,-84.39)
lata1=c(47.62,33.75)
distbearcalc(-111.047,45.68,long1,lat1)
#$distance
#[1] 552.6337 1629.4640
#$bear
#[1] 71.8899 248.9611

# test longlatdistbear
dbear=distbearcalc(-111.047,45.68,long1,lat1)
longlatdistbear(-111.047,45.68,dbear$distance,dbear$bear)
#$ long1 lat1
#1 -122.3500 47.61999
#2 -84.3900 33.74996
```
longlatkeep

Pulls all points from set LL0 within given distance of any point in set LL1

Description
Pulls all points from set LL0 within given distance of any point in set LL1

Usage
longlatkeep(LL0, LL1, distin)

Arguments
LL0 n0 by 2 matrix or dataframe of long-lat points
LL1 n1 by 2 matrix or dataframe of long-lat points
distin distance to keep data for

Value
list containing: LL0, LL1, keep0 = vector with first point found in point in set LL1 within distance or zero if none, dist1 = vector with distance from point in LL0 to first point found in LL1 or closest point if none

Author(s)
Joe Atwood

Examples
#Not Run
#df0=read.csv('LL0.txt',header=T)
#df1=read.csv('LL1.txt',header=T)
#LL0=as.matrix(df0[,2:3])
#LL1=as.matrix(df1[,2:3])
#distin=50.0
#df1=longlatkeep(LL0=LL0,LL1=LL1,distin=50.0)
lookup

Description
emulates a simple version of the lookup function in excel

Usage
lookup(x,getvals,xmax=1000000)

Arguments
- x: values to be assigned values
- getvals: dataframe with first column containing the lookup reference
- xmax: upper bound on last interval

Value
a data.frame containing x and the lookup values

Author(s)
Joe Atwood

Examples
########################################################
# Not Run
#cv=seq(0.5,0.85,0.05)
#sub=c(0.67,0.64,0.64,0.59,0.59,0.55,0.48,0.38)
#df0=data.frame(cbind(cv,sub))
#x=seq(0.49,0.86,0.01)
#cbind(x,lookup(x,getvals=df0))
########################################################-

maxgdhess

Description
Numerically maximize a function:
maxgdhess (maximize function with no restrictions)
maxgdhessr (maximize function with restrictions)
maxgdhess2 (maximize function with no restrictions)
maxgdhessr2 (maximize function with restrictions)
Usage

# Notes: DO NOT pass x as an argument to the maxgdhess or maxgdhessr
# functions. x is reserved for the parameters in the grad and
# hessian function within the numDeriv package
# The pair of functions maxgdhess2 and and maxgdhessr2 are
# included as they sometimes run substantially faster than
# the maxgdhess and and maxgdhessr functions. The
# '.2' functions use my gradient and hessian routines which
# may be somewhat less accurate than the numDeriv functions. However
# my functions often run faster than the numDeriv functions when extra
# arguments are passed to the function. (See below)
#
# A possible compromise I have used is to first run the '.2'
# functions and use the resulting parameters as starting values
# for maxgdhessr. (See below)
#
maxgdhess(obj,b,itlim = 100,epstop=1e-07,plotum="N",ptitle = ",ndirs=1,useDIRg='N', ...)
maxgdhessr(obj,b,itlim = 100,epstop=1e-07,plotum="N",ptitle = ",ndirs=1,useDIRg='N', ...)
maxgdhess2(obj,b,itlim=100,epstop=1e-07,plotum="N",ptitle="",ndirs=1,useDIRg='N', ...)
maxgdhessr2(obj,b,itlim=100,epstop=1e-07,plotum="N",ptitle="",ndirs=1,useDIRg='N', ...)

Arguments

obj    objective function
b      parameter starting values
itlim  maximal number of iterations allowed
epstop Stopping tolerance level
plotum Plot (plotum='Y') or do not plot (plotum='N') the objective value progress
ptitle Title to print on the plots if plotum = 'Y'
ndirs  Number of search directions between the gradiant and the Hessian search directions
useDIRg Add extra partial derivative search direction for each parameter
...    Data or setting to pass to the objective function

Value

b      parameter estimates
obj    the objective value
gradb  gradiant vector
gradpos vector of RHS derivatives
gradneg  vector of LHS derivatives
hess     numerical hessian
sigma    estimated parameter covariance matrix
status   status of search routine
iter     number of iterations completed
hesstest second order condition test on Hessian
b.restrict check on restricted parameter estimates

Author(s)
Joe Atwood

Examples

# NOTE: IMPORTANT
# DO NOT use and pass x as a extra argument to the
# maxgdhess or maxgdhessr functions. The variable x is reserved
# by the numDeriv package functions grad and hessian and will
# not pass through grad() or hessian() to the objective function.
# The pair of functions maxgdhess2 and and maxgdhessr2 are
# included as they sometimes run substantially faster than
# the maxgdhess and and maxgdhessr functions. The
# '.2' functions use my gradient and hessian routines which
# may be somewhat less accurate than the numDeriv functions. However
# my functions often run faster than the numDeriv functions when extra
# arguments are passed to the function. (See below)
# A possible compromise I have used is to first run the '.2'
# functions and use the resulting parameters as starting values
# for maxgdhessr. (See below)
# Example of unrestricted quadratic function
# Optimal answer = c(10,10) Maximal obj = 100

obj1=function(b=c(0,0)){
}

maxgdhess(obj=obj1,b=c(0,0))
maxgdhess2(obj=obj1,b=c(0,0))

# EXAMPLES OF RESTRICTED FUNCTIONS

# Standardized Beta Distribution maximum likelihood example
# Note: The returnpars lines must be included if you want
# to use maxgdhessr

```r
obj2=function(b,z=z,blo=c(0.5,0.5),returnpars='N') {
  b=ifelse(b<blo,blo,b)
  fz=dbeta(z,b[1],b[2])
  obj=sum(log(fz))
  if(returnpars!='Y') return(obj)
  if(returnpars=='Y') return(list(obj=obj,b=b))
}
```

```r
set.seed(1)
z=rbeta(100,2,2)
b0=c(2,2)
obj2(b=b0,z=z)
maxgdhessr(obj=obj2,b=b0,z=z)
maxgdhessr(obj=obj2,b=c(4,4),blo=c(3,3),z=z)
maxgdhessr2(obj=obj2,b=c(4,4),blo=c(3,3),z=z)
```

# NON-STANDARDIZED BETA DISTRIBUTION MAXIMUM LIKELIHOOD EXAMPLE

```r
rbetaN=function(n=100,al=5,bt=5,k1=0,k2=1){
  if(k1>=k2) stop("k2 must exceed k1")
  if(al<=0) stop("The alpha parameter must exceed zero")
  if(bt<=0) stop("The beta parameter must exceed zero")
  z=rbeta(n,al,bt)
  k1+z*(k2-k1)
}
```

```r
dbetaN=function(x,al=5,bt=5,k1=0,k2=1){
  if(k1==k2) stop("k2 must exceed k1")
  if(al<=0) stop("The alpha parameter must exceed zero")
  if(bt<=0) stop("The beta parameter must exceed zero")
  if(min(x)<k1|max(x)>k2) stop("The vector values must lie between k1 and k2")
  z=(x-k1)/(k2-k1)
  (1/(k2-k1))*dbeta(z,al,bt)
}
```

# Objective Function

```r
f1=function(b,z=z,returnpars='N') {
  b3lim=min(z)-.001
  b4lim=max(z)+.001
  b[1]=max(0.5,b[1])
  b[2]=max(0.5,b[2])
  b[3]=min(b[3],b3lim)
  b[4]=max(b[4],b4lim)
  fz=dbetaN(z,al=b[1],bt=b[2],k1=b[3],k2=b[4])
  obj=sum(log(fz))
}
```
```r
if (returnpars != 'Y') return(obj)
if (returnpars == 'Y') return(list(obj=obj,b=b))
}

set.seed(1)
zN = rbetaN(1000,2,2,100,200)
b1 = c(1,1,50,300)
f1(z = zN, b = b1)

time0 = date()
(dftmp = maxgdhessr(obj = f1, b = b1, z = zN, plotum = 'Y', useDIRg = 'Y'))
time1 = date()

# much faster to obtain initial values from maxgdhessr
(dftmp2 = maxgdhessr2(obj = f1, b = b1, z = zN, plotum = 'Y', useDIRg = 'Y'))
(dftmp = maxgdhessr(obj = f1, b = dftmp2$b, z = zN, plotum = 'Y', useDIRg = 'Y'))
time2 = date()
time0
time1
time2

f1(z = zN, b = dftmp$b)

b2 = c(5,5,0,400)
(dftmp3 = maxgdhessr2(obj = f1, b = b1, z = zN, plotum = 'Y', ndirs = 3))

# CROP YIELD TREND ESTIMATION EXAMPLE

# Pull Dawson County data for regressions
yld = ylddata$yld[ylddata$fips == 30021]
yr = ylddata$yr[ylddata$fips == 30021]
yr2 = ylddata$yr2[ylddata$fips == 30021]

# Pull Madison County data for regressions
yld = ylddata$yld[ylddata$fips == 30057]
yr = ylddata$yr[ylddata$fips == 30057]
yr2 = ylddata$yr2[ylddata$fips == 30057]

# Define our nonlinear least squares regression objective function
sserr = function(b, yld, yr2) {
  ehat = yld - yhat
  return(sum(ehat^2))
}
```

sse = sum(ehat * ehat)
sse
}

# convert the sserr obj to a max form
f0 = function(b, ...) {-1 * sserr(b, ...)}

# run linear regression (nested in model when b3 = 0 and b4 = 1)
reg1 = lm(yld ~ yr2)

# Use linear results as starting values for the nonlinear model
(b = c(as.numeric(reg1$coefficients), 0, 1))

# store the initial values and construct the predicted values using
# the starting values and the non-linear function
b.lm = b

# run the nonlinear model with 250 iteration limit
(nl1 = maxgdhess(obj = f0, b = b.lm, itlim = 250, ndirs = 1, yld = yld, yr2 = yr2, plotum = 'Y', useDIRg = 'Y'))

# We get better answer than nlm although nlm is much faster
(nl2 = nlm(sserr, b.lm, yld = yld, yr2 = yr2))

# Run maxgdhess with larger iteration limits
(nl3 = maxgdhess(obj = f0, b = nl1$b, itlim = 250, ndirs = 1, yld = yld, yr2 = yr2, plotum = 'Y', useDIRg = 'Y'))
(nl4 = maxgdhess(obj = f0, b = nl3$b, itlim = 500, ndirs = 1, yld = yld, yr2 = yr2, plotum = 'Y', useDIRg = 'Y'))

b.nl1 = nl1$b

b.nl2 = nl2$est

b.nl3 = nl3$b

b.nl4 = nl4$b

plot(yr, yld, type = 'b', pch = 20)
points(yr, yhat.lm, type = 'l', lwd = 2)
points(yr, yhat.nl1, type = 'l', lwd = 2, col = 1)
points(yr, yhat.nl2, type = 'l', lwd = 2, col = 2)
points(yr, yhat.nl3, type = 'l', lwd = 2, col = 3)
points(yr, yhat.nl4, type = 'l', lwd = 2, col = 4)

(sser = sserr(b.lm, yr2 = yr2, yld = yld))
(sseu = sserr(b.nl4, yr2 = yr2, yld = yld))

nobs = length(yr)

(fstat = (sser - sseu) / sseu / (nobs - 4))
dof1 = 2
dof2 = nobs - 4

(pval = 1 - pf(fstat, dof1, dof2))

meanOA  Olympic Average

Description
Olympic Average

Usage
meanOA(x, ncut = 1)

Arguments
x  Vector of numeric data
ncut  Number of observations to cut from each end of sorted data

Value
returns corresponding values

Author(s)
Joe Atwood

Examples
x=1:10
x[2:3]=NA
meanOA(x,1)
meanOA(x,2)
meanOA(x,4)

moveAVG  Compute Moving Averages on a Vector of Numbers

Description
Compute Moving Average on a Vector of numbers

Usage
moveAVG(x,lagum=5,na.rm='T',include='T')
### OavgM

**Compute Olympic Averages on Rows of Matrix**

**Description**

Compute Olympic Averages on Rows of Matrix

**Usage**

OavgM(XM, ntrim = 1)

**Arguments**

- **XM**: Matrix or dataframe of numbers
- **ntrim**: number of elements to trim from each end of sample

**Value**

returns a vector of Olympic Average Values

**Author(s)**

Joe Atwood
Examples

```r
set.seed(2011)
XM=matrix(rnorm(10000),2000,5)
OA5=OavgM(XM,ntrim=1)
OA5[1]
x=XM[1,]
mean(x,trim=0.2)
```

Description

Pick data from matrices

Usage

```r
pickum(Z,Y,n)
```

Arguments

- `Z`: integer array of dimension (ndays,nstations,ndi)
- `Y`: real array of dimension (ndays,nstations,ndr)
- `n`: number of stations to find

Value

returns a list of integer array IP(nday,1:n,ndi) and YP(nday,1:n,ndr)

Author(s)

Joe Atwood

Examples

```r
# Note: The first dimension in Z must have zero-one values
# with 1's indicating a valid value. Missing values in Y are
# assumed to be coded with -99999 values for use in the underlying
# Fortran code

nr=2500
nc=100
N=nr*nc
n=5

Z1=matrix(rbinom(N,1,0.5),nr,nc)
Y1=matrix(round(runif(N,1,2),1),nr,nc)
Y1=Y1*Z1

Z=array(0,dim=c(nr,nc,1))
```
pointslope

pointslope

Plots line segments with given slope through a given point

Description

Plots line segments with given slope through a given point

Usage

pointslope(x, m, lty=1, lwd=1, col=1, font=1, xlim=c(-1000000, 1000000))

Arguments

x       point
m       slope
lty     line type
lwd     line width
col     line color
font    line font
xlim    range for x value to be plotted

Value

none

Author(s)

Joe Atwood

Examples

###########################################################
plot(10, 10, xlim=c(-10, 20), ylim=c(-10, 20), type='n')
abline(h=0, lwd=2, font=2)
adline(v=0, lwd=2, font=2)
point=c(10, 10)
pointslope(point, m=2)
pointslope(point, m=1, lty=3, lwd=3, col=2, xlim=c(6, 14))
###########################################################
**PolyfindplotM**

**Locate polygon containing and closest to longlat location**

**Description**

Locate polygon containing and closest to longlat location as well as distance to boundary of closest polygon. Assumes dfunction is on primary workspace. If not uses distcalc from jcode or jcode64.

**Usage**

```
PolyfindplotM(X0, shapes, polynum=5, npoints=10, plotum='N', pointbuffer=5, nsecs=0.0, dfunction=distcalc, mytext=' ', newmap='N')
```

**Arguments**

- `X0` Matrix or data frame containing only long0,lat0
- `shapes` An R polygon data object
- `polynum` Number of polygons to check - finds closest polynum centroids
- `npoints` Number of points along line segments to check
- `plotum` Plot polygon assigned to
- `pointbuffer` Maps units to plot surrounding the given point
- `nsecs` Seconds to wait between plots
- `dfunction` Distance function for x0,y0,x2,y2
- `mytext` text to add to plots
- `newmap` plot a new map for each point

**Value**

returns dataframe containing x0,y0,polyin,polyclose,x2,y2,dist2

**Author(s)**

Joe Atwood

**Examples**

```r
#require(maptools)
## Not Run
## long-lat of Bozeman Montana = -111.047,45.68
## long-lat of Seattle Washington = -122.35,47.62
## long-lat of Atlanta, Georgia = -84.39,33.75
## location off Georgia Coast = -81.09,31.35
#
#data(USMapsS48)
#x0=c(-111.047,-122.35,-84.39,-81.09)
#y0=c(45.68,47.62,33.75,31.35)
#X=data.frame(cbind(x0,y0))
#(tmp=PolyfindplotM(X,USMapsS48,plotum='Y',nsecs=2,newmap='Y'))
```
# x0  y0  polyin  polyclose  x2  y2  dist2
#1 -111.047 45.68  2  2 -111.053 44.99570 47.332577
#2 -122.350 47.62  1  1 -122.389 47.60989 1.951914
#3 -84.390 33.75  44  44 -85.332 33.63763 54.770122
#4 -81.090 31.35  NA  44 -81.236 31.43287 10.355521

Bozeman, Seattle, Atlanta are in polygons 2, 1, 44 respectively
and are, respectively, 47.3, 1.9, and 54.7 miles from the closest polygon boundary.
The last point is not in a polygon but is closest (10.36 miles) from the Georgia polygon

USMapsS48@data[tmp$polyin[1:3],]

Montana MT 30 2
Washington WA 53 1
Georgia GA 13 44

i.e. Bozeman is in Montana, Seattle in Washington and Atlanta in Georgia

---

**pullMdata**

*Pull data from an indexed matrix*

**Description**

Pull data from an indexed matrix

**Usage**

`pullMdata(rp, cp, M)`

**Arguments**

- `rp`: Rows to pull data from
- `cp`: Columns to pull data from
- `M`: Data matrix

**Value**

A vector of values from rows `rp` and columns `cp` in matrix `M` – See examples below

**Author(s)**

Joe Atwood

**Examples**

```r
set.seed(2011)
# create demo matrix
M=matrix(1:28,7,4)
# create row and column indexes to demonstrate
# the use of the match command below
rindex=paste('r',1:7,sep='')
```
pvalx

Description

Compute empirical p-value from a vector of data

Usage

pvalx(x, x0 = mean(x))

Arguments

x
vector of data

x0
pvalue item

Value

empirical pvalue – See examples below

Author(s)

Joe Atwood

Examples

# create a data frame of labels to be matched
rp = sample(rindex, 7000000, replace = TRUE)
cp = sample(cindex, 7000000, replace = TRUE)
df1 = data.frame(rp, cp)
pick1 = match(df1$rp, rindex)
pick2 = match(df1$cp, cindex)
time0 = seconds.local()
df1$value = pullMdata(pick1, pick2, M)
time1 = seconds.local()
time1 - time0

pvalx

Compute empirical pvalue from a vector of data

set.seed(2012)
x = rnorm(100000, 100, 25)
pvalx(x)
pvalx(x, 90)
pgnorm(90, 100, 25)

# create a data frame of labels to be matched
rp = sample(rindex, 7000000, replace = TRUE)
cp = sample(cindex, 7000000, replace = TRUE)
df1 = data.frame(rp, cp)
pick1 = match(df1$rp, rindex)
pick2 = match(df1$cp, cindex)
time0 = seconds.local()
df1$value = pullMdata(pick1, pick2, M)
time1 = seconds.local()
time1 - time0
Pvalx_V  

Compute empirical pvalue from a vector of data

Description

Compute empirical p-values

Usage

Pvalx_V(x,q)

Arguments

x  vector of data
q  vector of pval item

Value

empirical pvalue – See examples below

Author(s)

Joe Atwood

Examples

###########################################
#set.seed(2012)
#x=rnorm(100000,100,25)
#pvalx(x,q=c(70,80,90))
pnorm(c(70,80,90),100,25)
###########################################

quicksort  

sorts a vector of real variables

Description

sorts a vector of real variables

Usage

quicksort(x)

Arguments

x  a vector of real variables to sort
Value

Returns a list containing the vector xsort of resorted values and a vector order containing the location of xsort values in the original x vector.

Author(s)

Joe Atwood

Examples

```r
x=rep(5:1,each=2)
x[4]=1
quicksort(x)
```

---

**radcalc**

computes radians or degree bearings between origin and one or more points

Description

computes radians or degree bearings between origin and one or more points

Usage

```
radcalc=(x1,y1,x0=0,y0=0,degrees='F')
```

Arguments

- `x1`: vector of data
- `y1`: vector of data
- `x0`: origin
- `y0`: origin
- `degrees`: result in degrees - default in radians

Value

radian or degree bearing from `east` – See examples below

Author(s)

Joe Atwood
**Examples**

```
radcalc(1,1);(1/4)*pi
radcalc(-1,1); (3/4)*pi
radcalc(-1,-1); (5/4)*pi
radcalc(1,-1); (7/4)*pi
radcalc(1,0)
radcalc(0,1)
radcalc(0,-1)
radcalc(0,-1,degrees='T')
x1=c(1,0,-1,-1, 0, 1,1)
x2=c(1, 1, 0,-1,-1,0)
radcalc(x1,x2,degrees='T')
```

---

**rbetaN**  
Data generation and values from the non-standardized Beta distribution

**Description**

Conventional Functions that generate data and values from the non-standardized Beta distribution

**Usage**

```
rbetaN(n=100,al=5,bt=5,k1=0,k2=1)
dbetaN(x,al,bt,k1,k2)
pbetaN(x,al,bt,k1,k2)
qbetaN(p,al,bt,k1,k2)
LLbetaN(x,al,bt,k1,k2)
betaN_est(x,k1,k2)
```

**Arguments**

- `n`: number of observations to generate
- `al`: alpha parameter for non-standard Beta distribution
- `bt`: Beta parameter for non-standard Beta distribution
- `k1`: Lower bound for non-standard Beta distribution
- `k2`: Upper bound for non-standard Beta distribution
- `x`: Scalar or Vector of data observations
- `p`: Scalar or Vector of pvalues

**Value**

- `rbetaN` returns observation generate from BetaN(al,bt,k1,k2)
- `dbetaN` returns pdf density value(s) from BetaN(al,bt,k1,k2) for point(s) x
- `pbetaN` returns cumulative density value(s) from BetaN(al,bt,k1,k2) for point(s) x
- `betaN_est` returns estimated alpha and beta parameters given prespecified k1 and k2 bounds
rho2cor

Constructs and pulls data from correlation matrices

Description

Constructs and pulls data from correlation matrices

Usage

rho2cor(rho)
cor2rho(R)

Arguments

rho a vector of correlations
R a matrix of correlations

Value

rho2cor returns a correlation matrix cor2rho returns a vector of correlations

Author(s)

Joe Atwood
Examples

 rho3=c(0.755,0.655,0.505)
 (R3=rho2cor(rho3))
 (rho4=(6:1)/10)
 (R4=rho2cor(rho4))
 cor2rho(R4)

---

rnormJ

Data generation and values from the univariate and multivariate Normal distribution

Description

Modifies existing R functions so that mean, sd, or covariance matrices exactly matches specified parameters

Usage

rnormJ(n,mu=0,sdev=1)
rmvnormJ(n,mu=c(0,0),sigma=diag(c(1,1))

Arguments

n number of observations to generate
mu mean values
sdev standard deviations for rnormJ
sigma covariance matrix

Value

rnormJ returns a vector of normally distributed random numbers rmvnormJ returns a matrix of multivariate normal random numbers

Author(s)

Joe Atwood

Examples

# Not Run
#set.seed(2012)
#x=rnormJ(100)
#mean(x);sd(x)

#mu=c(10,5)
#Var=c(9,4)
#(sigma=diag(Var))
#
#X=rmvnormJ(100,mu,sigma)
### RunMA

**Computes running moving averages on matrix columns**

**Description**

Computes running moving averages on matrix columns

**Usage**

`RunMA(A, nc1, nc2, n)`

**Arguments**

- `A` matrix
- `nc1` starting column
- `nc2` ending column
- `n` number of columns in the average

**Value**

Returns a matrix of running moving averages

**Author(s)**

Joe Atwood

**Examples**

```r
nr=10000
nc=500
A=matrix(rnorm(nr*nc),nr,nc)
RMA=RunMA(A, nc1=1, nc2=200, n=30)
```
Converts a real matrix to a sparse matrix form

Usage

\[ \text{R\_A2SM}(A, \text{SMM} = \text{'CRI'}, \text{eps} = 1e-9, \text{CINDEX} = 0) \]

Arguments

- **A**: real matrix of dimension \( n_r \times n_c \)
- **SMM**: Sparse matrix method. 'CRI' = both row and col index, 'CMO' = Column major order, 'RMO' = row major order
- **eps**: \( \text{if abs}(a_{ij}) < \text{eps} \text{ treated as zero} \)
- **CINDEX**: \( 1 \Rightarrow \text{use C zero indexes, } 1 \Rightarrow \text{use R/Fortran 1 indexes} \)

Value

- **A**: matrix
- **nr**: number of rows in A
- **nc**: number of columns in A
- **SMM**: Sparse matrix method
- **CINDEX**: "C" numbering index
- **ac**: non-zero elements in A
- **ia**: row indexes
- **ja**: column matrices

Author(s)

Joe Atwood

Examples

```
#(A=matrix(c(1,0,0,2,0,0,0,0,0,5,0,6),3,4,byrow=T))
#R\_A2SM(A=A, SMM=\text{'CRI'})
#R\_A2SM(A=A, SMM=\text{'CMO'})
#R\_A2SM(A=A, SMM=\text{'RMO'})
```
R_SM2A

Converts sparse matrix data to a matrix

Description
Converts sparse matrix data to a matrix

Usage
R_SM2A(nr,nc,ac,ia,ja,SMM='CRI',CINDEX=0)

Arguments
nr number of rows in matrix
nc number of columns in matrix
ac non-zero elements in A
ia row indexes
ja column matrices
SMM Sparse matrix method. 'CRI'=both row and col index, 'CMO' = Column major order. 'RMO' = row major order
CINDEX 1 => use C 'zero' indexes, 1 => use R/Fortran 1 indexes

Value
A matrix

Author(s)
Joe Atwood

Examples

#(A=matrix(c(1,0,2,0,0,0,0,0,5,6),3,4,byrow=T))
#R_A2SM(A=A,SMM='CRI')
#R_A2SM(A=A,SMM='CMO')
#R_A2SM(A=A,SMM='RMO')
#
#tmp=R_A2SM(A=A,SMM='CRI')
#nr=tmp$nr
#nc=tmp$nc
#ac=tmp$ac
#ia=tmp$ia
#ja=tmp$ja
#SMM=tmp$SMM
#CINDEX=tmp$CINDEX
#
#A
#R_SM2A(nr,nc,ac,ia,ja,SMM,CINDEX)
seconds.local

Pull seconds from system timer

Description
Pull seconds from system timer

Usage
seconds.local()

Arguments
none

Value
Pull seconds from system timer

Author(s)
Joe Atwood

Examples
###########################################
time1=seconds.local()
for (i in 1:1000000) {x=sqrt(i)}
time2=seconds.local()
time2-time1
class=seconds.local()

sin_d

Functions to compute cos and sin with for angles measured in degrees

Description
Functions to compute cos and sin with for angles measured in degrees

Usage

sin_d(45)
cos_d(45)

Arguments
x vector of degrees
Value

returns corresponding values

Author(s)

Joe Atwood

Examples

\[ \cos_d(45) \]
\[ \sin_d(45) \]

---

**SM2A**

*Converts sparse matrix data to a matrix*

Description

Converts sparse matrix data to a matrix

Usage

\[ \text{SM2A}(nr, nc, ac, ia, ja, SMM='CRI', CINDEX=0) \]

Arguments

- **nr**: number of rows in matrix
- **nc**: number of columns in matrix
- **ac**: non-zero elements in A
- **ia**: row indexes
- **ja**: column matrices
- **SMM**: Sparse matrix method. ‘CRI’=both row and col index, ‘CMO’ = Column major order, ‘RMO’ = row major order
- **CINDEX**: 1 => use C ‘zero’ indexes, 1 => use R/Fortran 1 indexes

Value

A matrix

Author(s)

Joe Atwood
Examples

#(A=matrix(c(1,0,0,2,0,0,0,0,0,5,0,6),3,4,byrow=T))
#A2SM(A=A,SMM='CRI')
#A2SM(A=A,SMM='OMO')
#A2SM(A=A,SMM='RMO')
#
#tmp=A2SM(A=A,SMM='CRI')
#nr=tmp$nr
#nc=tmp$nc
#ac=tmp$ac
#ia=tmp$ia
#ja=tmp$ja
#SMM=tmp$SMM
#CINDEX=tmp$CINDEX
#
#A
#SM2A(nr,nc,ac,ia,ja,SMM,CINDEX)

summaryDF  

 computes summary by variable

Description

computes summary by variable

Usage

summaryDF(data, splits, operate.on.columns, functions = "mean", ...)

Arguments

data data frame to operate on
splits variables to summarize by
operate.on.columns variables to summarize over
functions functions to apply to variables
... misc arguments to pass through to the functions

Value

summaries passed back to the main workspace

Author(s)

Jim Robison-Cox
Examples

```r
x = rep(1:10, each=10)
y = rep(1:10, 10)
z1 = rnorm(100)
z2 = rnorm(100)
dftmp = data.frame(cbind(x, y, z1, z2))
summaryDF(dftmp, c('x'), c(z1, 'z2'), c('mean', 'sd'))
```

---

**ulong**

Computes number of unique elements in a vector

### Description

Computes number of unique elements in a vector

### Usage

`ulong(x)`

### Arguments

- `x`: vector of values
- `na.rm`: remove NA elements of vector

### Value

Number of unique values

### Author(s)

Joe Atwood

### Examples

```r
# example usage
x1 = rep(1:5, each=3)
x2 = rep(1:5, 3)
x2[5] = NA
ulong(x2)
# ulong(x2, na.rm = T)
tmp = data.frame(x1, x2)
summaryBy(x2 ~ x1, data = tmp, FUN = ulong)
# summaryBy(x2 ~ x1, data = tmp, FUN = ulong, na.rm = T)
```

---

**ulong**

Computes number of unique elements in a vector
USMapsS48

US 48 state maps

Description

US 48 state maps

Usage

data(USMapsS48)

Format

The format is: Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots ..@ data ..'data.frame':: 49 obs. of 6 variables: .. ..$ sfips: int [1:49] 53 30 23 38 46 56 55 16 50 27 ... .. ..$ sname: Factor w/ 51 levels "Alabama","Alaska"...: 48 27 20 35 42 51 50 13 46 24 ... .. ..$ sabbr: Factor w/ 51 levels "AK","AL","AR"...: 48 27 22 29 42 51 49 14 47 24 ... .. ..$ index: int [1:49] 1 2 3 4 5 6 7 8 9 10 ... .. ..$ long : num [1:49] -120.4 -109.7 -69.2 -100.5 -100.2 ... .. ..$ lat : num [1:49] 47.4 47 45.4 47.4 44.4 ... .. ..@ polygons :List of 49 .. etc

Examples

data(USMapsS48)
# not run
# require(maptools)
# plot(USMapsS48,col=1:10)

wait

wait nsecs seconds

Description

Creates nsec second pause in R execution

Usage

wait(nsecs=5)

Arguments

nsecs Seconds to wait before proceeding

Value

None

Author(s)

Joe Atwood
Examples

wait(nsecs=5,talk=1)

waittalk

wait nsecs seconds while printing remaining time

Description

Creates nsec second pause in R execution while printing remaining time

Usage

wait(nsecs=5,talk=1)

Arguments

nsecs Seconds to wait before proceeding

talk Seconds to wait before communicating to console

Value

seconds remaining in wait loop

Author(s)

Joe Atwood

Examples

wait(nsecs=5,talk=1)

wait(nsecs=5,talk=1)
**Description**

NASS Wheat yield data for Dawson(30021) and Madison(30057) Counties, Montana

**Usage**

data(ylddata)

**Format**

A data frame with 109 observations on the following 7 variables.

- **fips**: a state-county fips code giving a unique number for each county in the US
- **nst**: state fips code
- **ncnty**: county fips code
- **yr**: year
- **yr2**: a rescaled year vector for the nonlinear regressions
- **acres**: acres harvested
- **yld**: per acre yield (bushels)

**Details**

~~ ~~

**Source**


**References**

~~ possibly secondary sources and usages ~~

**Examples**

data(ylddata)
Index

<table>
<thead>
<tr>
<th>Topic</th>
<th>datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>USMapsS48, 53</td>
</tr>
<tr>
<td></td>
<td>ylddata, 55</td>
</tr>
<tr>
<td></td>
<td>A2SM, 3</td>
</tr>
<tr>
<td></td>
<td>autosim, 4</td>
</tr>
<tr>
<td></td>
<td>bearcalc, 5</td>
</tr>
<tr>
<td></td>
<td>betaN_est (rbetaN), 43</td>
</tr>
<tr>
<td></td>
<td>bigeight.na, 6</td>
</tr>
<tr>
<td></td>
<td>bigfour_.na (length.na), 24</td>
</tr>
<tr>
<td></td>
<td>bigseven.na, 7</td>
</tr>
<tr>
<td></td>
<td>bigsix.na, 8</td>
</tr>
<tr>
<td></td>
<td>bimo.yr (jmonth), 22</td>
</tr>
<tr>
<td></td>
<td>class_df, 9</td>
</tr>
<tr>
<td></td>
<td>closeDist, 10</td>
</tr>
<tr>
<td></td>
<td>colorum.gplots, 11</td>
</tr>
<tr>
<td></td>
<td>cor2cov, 12</td>
</tr>
<tr>
<td></td>
<td>cor2rho (rho2cor), 44</td>
</tr>
<tr>
<td></td>
<td>cos_d (sin_d), 49</td>
</tr>
<tr>
<td></td>
<td>CtyData, 13</td>
</tr>
<tr>
<td></td>
<td>dbetaN (rbetaN), 43</td>
</tr>
<tr>
<td></td>
<td>distbearcalc, 14</td>
</tr>
<tr>
<td></td>
<td>distcalc, 15</td>
</tr>
<tr>
<td></td>
<td>distmat, 16</td>
</tr>
<tr>
<td></td>
<td>dJB (JBpars), 21</td>
</tr>
<tr>
<td></td>
<td>gradn (gradum), 17</td>
</tr>
<tr>
<td></td>
<td>gradp (gradum), 17</td>
</tr>
<tr>
<td></td>
<td>gradum, 17</td>
</tr>
<tr>
<td></td>
<td>hessum (gradum), 17</td>
</tr>
<tr>
<td></td>
<td>ImanConover, 18</td>
</tr>
<tr>
<td></td>
<td>ImanConoverYcZc, 19</td>
</tr>
<tr>
<td></td>
<td>int2real_df, 20</td>
</tr>
<tr>
<td></td>
<td>jbimonth (jmonth), 22</td>
</tr>
<tr>
<td></td>
<td>JBpars, 21</td>
</tr>
<tr>
<td></td>
<td>jcode64 (jcode64-package), 2</td>
</tr>
<tr>
<td></td>
<td>jcode64-package, 2</td>
</tr>
<tr>
<td></td>
<td>jday_shift (jmonth), 22</td>
</tr>
<tr>
<td></td>
<td>jmonth, 22</td>
</tr>
<tr>
<td></td>
<td>lagum, 23</td>
</tr>
<tr>
<td></td>
<td>lagumMat (lagum), 23</td>
</tr>
<tr>
<td></td>
<td>length.na, 24</td>
</tr>
<tr>
<td></td>
<td>LLbetaN (rbetaN), 43</td>
</tr>
<tr>
<td></td>
<td>longlatdistbear, 25</td>
</tr>
<tr>
<td></td>
<td>longlatkeep, 27</td>
</tr>
<tr>
<td></td>
<td>lookup, 28</td>
</tr>
<tr>
<td></td>
<td>max.na (length.na), 24</td>
</tr>
<tr>
<td></td>
<td>maxgdhess, 28</td>
</tr>
<tr>
<td></td>
<td>maxgdhess2 (maxgdhess), 28</td>
</tr>
<tr>
<td></td>
<td>maxgdhessr (maxgdhess), 28</td>
</tr>
<tr>
<td></td>
<td>maxgdhessr2 (maxgdhess), 28</td>
</tr>
<tr>
<td></td>
<td>mean.na (length.na), 24</td>
</tr>
<tr>
<td></td>
<td>meanOA, 34</td>
</tr>
<tr>
<td></td>
<td>min.na (length.na), 24</td>
</tr>
<tr>
<td></td>
<td>mo.yr (jmonth), 22</td>
</tr>
<tr>
<td></td>
<td>moveAVG, 34</td>
</tr>
<tr>
<td></td>
<td>OavgM, 35</td>
</tr>
<tr>
<td></td>
<td>pbetaN (rbetaN), 43</td>
</tr>
<tr>
<td></td>
<td>pickum, 36</td>
</tr>
<tr>
<td></td>
<td>pJB (JBpars), 21</td>
</tr>
<tr>
<td></td>
<td>pointslope, 37</td>
</tr>
<tr>
<td></td>
<td>PolyfindplotM, 38</td>
</tr>
<tr>
<td></td>
<td>pullMdata, 39</td>
</tr>
<tr>
<td></td>
<td>pvalx, 40</td>
</tr>
<tr>
<td></td>
<td>Pvalx_Y, 41</td>
</tr>
<tr>
<td></td>
<td>qbetaN (rbetaN), 43</td>
</tr>
<tr>
<td></td>
<td>qJB (JBpars), 21</td>
</tr>
<tr>
<td></td>
<td>quicksort, 41</td>
</tr>
<tr>
<td></td>
<td>R_A2SM, 47</td>
</tr>
<tr>
<td></td>
<td>R_SM2A, 48</td>
</tr>
<tr>
<td></td>
<td>radcalc, 42</td>
</tr>
<tr>
<td></td>
<td>rbetaN, 43</td>
</tr>
<tr>
<td></td>
<td>rho2cor, 44</td>
</tr>
<tr>
<td></td>
<td>rJB (JBpars), 21</td>
</tr>
<tr>
<td></td>
<td>rmvnormJ (rnormJ), 45</td>
</tr>
<tr>
<td></td>
<td>rnormJ, 45</td>
</tr>
<tr>
<td></td>
<td>RunMA, 46</td>
</tr>
<tr>
<td></td>
<td>sd.na (length.na), 24</td>
</tr>
</tbody>
</table>
INDEX

seconds.local, 49
sin_d, 49
SM2A, 50
sum.na (length.na), 24
summaryDF, 51
ulong, 52
uplag (lagum), 23
USMapsS48, 53
wait, 53
waittalk, 54
ylddata, 55