Package 'RegSDC'

October 12, 2022

Type Package

Title Information Preserving Regression-Based Tools for Statistical Disclosure Control

Version 0.7.0

Date 2022-08-19

Author Øyvind Langsrud [aut, cre]

Maintainer Øyvind Langsrud <oyl@ssb.no>

Depends R (>= 3.0.0), Matrix

Imports SSBtools (>= 1.3.4), MASS

Description Implementation of the methods described in the paper with the above title: Langsrud, Ø. (2019) <doi:10.1007/s11222-018-9848-9>. The package can be used to generate synthetic or hybrid continuous microdata, and the relationship to the original data can be controlled in several ways. A function for replacing suppressed tabular cell frequencies with decimal numbers is included.

License Apache License 2.0 | file LICENSE

Encoding UTF-8

URL https://github.com/olangsrud/RegSDC

BugReports https://github.com/olangsrud/RegSDC/issues

RoxygenNote 7.2.0

NeedsCompilation no

Repository CRAN

Date/Publication 2022-08-19 08:30:02 UTC

R topics documented:

`CalculateCdirect` *Calculation of C by solving equation 10 in the paper*

Description

The limit calculated by `FindAlpha` is used when alpha =1 cannot be chosen (warning produced). In output, alpha is attribute.

Usage

```
CalculateCdirect(a, b, epsAlpha = 1e-07, AlphaHandler = warning, alpha = NULL)

CalculateC(a, b, ..., viaQR = NULL, returnAlpha = FALSE)
```

Arguments

`a`	matrix E in paper
`b`	matrix Eg in paper
`epsAlpha`	Precision constant for alpha calculation
`AlphaHandler`	Function (warning or stop) to be used when alpha<1
`alpha`	Possible with alpha as input instead of computing
`...`	Arguments to CalculateCdirect
`viaQR`	When TRUE QR is involved. This may be needed to handle colinear data. When NULL viaQR is set to TRUE if ordinary computations fail.
`returnAlpha`	When TRUE alpha (1 or value below 1) is returned instead of C. Attribute viaQR is included.

Details

When epsAlpha=NULL calculations are performed directly (alpha=1) and alpha is not attribute.

Value

Calculated C with attributes alpha and viaQR (when CalculateC)

Author(s)

Øyvind Langsrud

Examples

```
x <- 1:10
y <- matrix(rnorm(30) + 1:30, 10, 3)
a <- residuals(lm(y ~ x))
b <- residuals(lm(2 * y + matrix(rnorm(30), 10, 3) ~ x))

a1 <- a
b1 <- b
a1[, 3] <- a[, 1] + a[, 2]
b1[, 3] <- b[, 1] + b[, 2]

alpha <- FindAlpha(a, b)
FindAlphaSimple(a, b)  # Same result as above
CalculateC(a, b)
CalculateCdirect(a, b)  # Same result as above without viaQR attribute
CalculateCdirect(a, b, alpha = alpha/(1 + 1e-07))  # Same result as above since epsAlpha = 1e-07
CalculateCdirect(a, b, alpha = alpha/2)  # OK
# CalculateCdirect(a,b, alpha = 2*alpha) # Not OK

FindAlpha(a, b1)
# FindAlphaSimple(a,b1) # Not working since b1 is collinear
CalculateC(a, b1, returnAlpha = TRUE)  # Almost same alpha as above (epsAlpha cause difference)

FindAlpha(b, a)
CalculateC(b, a, returnAlpha = TRUE)  # 1 returned (not same as above)
CalculateC(b, a)

FindAlpha(b1, a)   # alpha smaller than epsAlpha is set to 0 in CalculateC
CalculateC(b1, a)  # When alpha = 0 C is calculated by GenQR insetad of chol
```

`FindAlpha` *Calculation of alpha*

Description

Function to find the largest alpha that makes equation 10 in the paper solvable.

Usage

```
FindAlpha(a, b, tryViaQR = TRUE)

FindAlphaSimple(a, b)
```

Arguments

a	matrix E in paper
b	matrix Eg in paper
tryViaQR	When TRUE QR transformation used (to handle collinearity) when ordinary calculations fail.

Value

alpha

Note

FindAlphaSimple performs the calculations by a simple/direct method. FindAlpha is made to handle problematic special cases.

Author(s)

Øyvind Langsrud

See Also

See examples in the documentation of CalculateC

GenQR *Generalized QR decomposition*

Description

Matrix X decomposed as Q and R (X=QR) where columns of Q are orthonormal. Ordinary QR or SVD may be used.

Usage

```
GenQR(x, doSVD = FALSE, findR = TRUE, makeunique = findR, tol = 1e-07)
```

Arguments

x	Matrix to be decomposed
doSVD	When TRUE SVD instead of QR
findR	When FALSE only Q returned
makeunique	When TRUE force uniqueness by positive diagonal elements (QR) or by column sums (SVD)
tol	As input to qr or, in the case of svd(), similar as input to MASS::ginv().

Details

To handle dependency a usual decomposition of X is PX=QR where P is a permutation matrix. This function returns RP^T as R. When SVD, Q=U and R=SV^T.

Value

List with Q and R or just Q

Author(s)

Øyvind Langsrud

Examples

```
GenQR(matrix(rnorm(15),5,3))
GenQR(matrix(rnorm(15),5,3)[,c(1,2,1,3)])
GenQR(matrix(rnorm(15),5,3)[,c(1,2,1,3)],TRUE)
```

RegSDCadd *Regression-based SDC Tools - Synthetic addition with residual correlation control*

Description

Implementation of equation 6 (arbitrary residual data) and equation 7 (residual correlations) in the paper. The alpha limit is calculated (equation 9). The limit is used when alpha =1 cannot be chosen (warning produced). In output, alpha is attribute.

Usage

```
RegSDCadd(y, resCorr = NULL, x = NULL, yStart = NULL, ensureIntercept = TRUE)
```

Arguments

`y`	Matrix of confidential variables
`resCorr`	Required residual correlations (possibly recycled)
`x`	Matrix of non-confidential variables
`yStart`	Arbitrary data whose residuals will be used. Will be calculated from resCorr when NULL.
`ensureIntercept`	Whether to ensure/include a constant term. Non-NULL x is subjected to `EnsureIntercept`

Details

Use epsAlpha=NULL to avoid calculation of alpha. Use of alpha (<1) will produce a warning. Input matrices are subjected to `EnsureMatrix`.

Value

Generated version of y with alpha as attribute

Author(s)

Øyvind Langsrud

Examples

```
x <- matrix(1:10, 10, 1)
y <- matrix(rnorm(30) + 1:30, 10, 3)
yOut <- RegSDCadd(y, c(0.1, 0.2, 0.3), x)

# Correlations between residuals as required
diag(cor(residuals(lm(y ~ x)), residuals(lm(yOut ~ x))))

# Identical covariance matrices
cov(y) - cov(yOut)
cov(residuals(lm(y ~ x))) - cov(residuals(lm(yOut ~ x)))

# Identical regression results
summary(lm(y[, 1] ~ x))
summary(lm(yOut[, 1] ~ x))

# alpha as attribute
attr(yOut, "alpha")

# With yStart as input and alpha limit in use (warning produced)
yOut <- RegSDCadd(y, NULL, x, 2 * y + matrix(rnorm(30), 10, 3))
attr(yOut, "alpha")

# Same correlation for all variables
RegSDCadd(y, 0.2, x)
# But in this case RegSDCcomp is equivalent and faster
RegSDCcomp(y, 0.2, x)


# Make nearly collinear data
y[, 3] <- y[, 1] + y[, 2] + 0.001 * y[, 3]
# Not possible to achieve correlations. Small alpha with warning.
RegSDCadd(y, c(0.1, 0.2, 0.3), x)
# Exact collinear data
y[, 3] <- y[, 1] + y[, 2]
# Zero alpha with warning
RegSDCadd(y, c(0.1, 0.2, 0.3), x)
```

`RegSDCcomp` *Regression-based SDC Tools - Component score correlation control*

Description

Implementation of equation 8 in the paper.

Usage

```
RegSDCcomp(
  y,
  compCorr = NA,
  x = NULL,
  doSVD = FALSE,
  makeunique = TRUE,
  ensureIntercept = TRUE
)
```

Arguments

`y`	Matrix of confidential variables
`compCorr`	Required component score correlations (possibly recycled)
`x`	Matrix of non-confidential variables
`doSVD`	SVD when TRUE and QR when FALSE
`makeunique`	Parameter to be used in GenQR
`ensureIntercept`	Whether to ensure/include a constant term. Non-NULL x is subjected to `EnsureIntercept`

Details

NA component score correlation means independent random. Input matrices are subjected to `EnsureMatrix`.

Value

Generated version of y

Author(s)

Øyvind Langsrud

Examples

```
x <- matrix(1:10, 10, 1)
y <- matrix(rnorm(30) + 1:30, 10, 3)

# Same as IPSO (RegSDCipso)
RegSDCcomp(y, NA, x)

# Using QR and SVD
yQR <- RegSDCcomp(y, c(0.1, 0.2, NA), x)
ySVD <- RegSDCcomp(y, c(0.1, 0.2, NA), x, doSVD = TRUE)
```

```
# Calculation of residuals
r <- residuals(lm(y ~ x))
rQR <- residuals(lm(yQR ~ x))
rSVD <- residuals(lm(ySVD ~ x))

# Correlations for two first components as required
diag(cor(GenQR(r)$Q, GenQR(rQR)$Q))
diag(cor(GenQR(r, doSVD = TRUE)$Q, GenQR(rSVD, doSVD = TRUE)$Q))

# Identical covariance matrices
cov(yQR) - cov(ySVD)
cov(rQR) - cov(rSVD)

# Identical regression results
summary(lm(y[, 1] ~ x))
summary(lm(yQR[, 1] ~ x))
summary(lm(ySVD[, 1] ~ x))
```

RegSDCdata *Function that returns a dataset*

Description

Function that returns a dataset

Usage

```
RegSDCdata(dataset)
```

Arguments

dataset	Name of data set within the RegSDC package

Details

sec7data: Data in section 7 of the paper as a data frame

sec7y: Y in section 7 of the paper as a matrix

sec7x: X in section 7 of the paper as a matrix

sec7z: Z in section 7 of the paper as a matrix

sec7xAll: Xall in section 7 of the paper as a matrix

sec7zAll: Zall in section 7 of the paper as a matrix

sec7zAllSupp: As Zall with suppressed values set to NA

Value

data frame

Author(s)

Øyvind Langsrud

Examples

```
RegSDCdata("sec7data")
RegSDCdata("sec7y")
RegSDCdata("sec7x")
RegSDCdata("sec7z")
RegSDCdata("sec7xAll")
RegSDCdata("sec7zAll")
RegSDCdata("sec7zAllSupp")
```

RegSDChybrid *Regression-based SDC Tools - Generalized microaggregation*

Description

Implementation of the methodology in section 6 in the paper

Usage

```
RegSDChybrid(
  y,
  clusters = NULL,
  xLocal = NULL,
  xGlobal = NULL,
  clusterPieces = NULL,
  xClusterPieces = NULL,
  groupedClusters = NULL,
  xGroupedClusters = NULL,
  alternative = NULL,
  alpha = NULL,
  ySim = NULL,
  returnParts = FALSE,
  epsAlpha = 1e-07,
  makeunique = TRUE,
  tolerance = sqrt(.Machine$double.eps)
)
```

Arguments

y	Matrix of confidential variables
clusters	Vector of cluster coding
xLocal	Matrix of x-variables to be crossed with clusters
xGlobal	Matrix of x-variables NOT to be crossed with clusters

clusterPieces Vector of coding of cluster pieces

xClusterPieces Matrix of x-variables to be crossed with cluster pieces

groupedClusters Vector of coding of grouped clusters

xGroupedClusters Matrix of x-variables to be crossed with grouped clusters

alternative One of "" (default), "a", "b" or "c"

alpha Possible to specify parameter used internally by alternative "c"

ySim Possible to specify the internally simulated data manually

returnParts Alternative output six matrices: y1 and y2 (fitted), e3s and e4s (new residuals), e3 and e4 (original residuals)

epsAlpha Precision constant for alpha calculation

makeunique Parameter to be used in GenQR

tolerance Parameter to Cdiff used within the algorithm

Details

Input matrices are subjected to EnsureMatrix. Necessary constant terms (intercept) are automatically included. That is, a column of ones is not needed in the input matrices.

Value

Generated version of y

Author(s)

Øyvind Langsrud

Examples

```
##################################################
# Generate example data for introductory examples
##################################################
y <- matrix(rnorm(30) + 1:30, 10, 3)
x <- matrix(1:10, 10, 1)  # x <- 1:10 is equivalent

# Same as RegSDCipso(y)
yOut <- RegSDChybrid(y)

# With a single cluster both are same as RegSDCipso(y, x)
yOut <- RegSDChybrid(y, xLocal = x)
yOut <- RegSDChybrid(y, xGlobal = x)

# Define two clusters
clust <- rep(1:2, each = 5)

# MHa and MHb in paper
yMHa <- RegSDChybrid(y, clusters = clust, xLocal = x)
```

```
yMHb <- RegSDChybrid(y, clusterPieces = clust, xLocal = x)

# An extended variant of MHb as mentioned in paper paragraph below definition of MHa/MHb
yMHbExt <- RegSDChybrid(y, clusterPieces = clust, xClusterPieces = x)

# Identical means within clusters
aggregate(y, list(clust = clust), mean)
aggregate(yMHa, list(clust = clust), mean)
aggregate(yMHb, list(clust = clust), mean)
aggregate(yMHbExt, list(clust = clust), mean)

# Identical global regression results
summary(lm(y[, 1] ~ x))
summary(lm(yMHa[, 1] ~ x))
summary(lm(yMHb[, 1] ~ x))
summary(lm(yMHbExt[, 1] ~ x))

# MHa: Identical local regression results
summary(lm(y[, 1] ~ x, subset = clust == 1))
summary(lm(yMHa[, 1] ~ x, subset = clust == 1))

# MHb: Different results
summary(lm(yMHb[, 1] ~ x, subset = clust == 1))

# MHbExt: Same estimates and different std. errors
summary(lm(yMHbExt[, 1] ~ x, subset = clust == 1))

####################################################
#  Generate example data for more advanced examples
####################################################
x <- matrix((1:90) * (1 + runif(90)), 30, 3)
x1 <- x[, 1]
x2 <- x[, 2]
x3 <- x[, 3]
y <- matrix(rnorm(90), 30, 3) + x
clust <- paste("c", rep(1:3, each = 10), sep = "")

######## Run main algorithm
z0 <- RegSDChybrid(y, clusters = clust, xLocal = x3, xGlobal = cbind(x1, x2))

# Corresponding models by lm
lmy <- lm(y ~ clust + x1 + x2 + x3:clust)
lm0 <- lm(z0 ~ clust + x1 + x2 + x3:clust)

# Preserved regression coef (x3 within clusters)
coef(lmy) - coef(lm0)

# Preservation of x3 coef locally can also be seen by local regression
coef(lm(y ~ x3, subset = clust == "c2")) - coef(lm(z0 ~ x3, subset = clust == "c2"))

# Covariance matrix preserved
cov(resid(lmy)) - cov(resid(lm0))
```

```
# But not preserved within clusters
cov(resid(lmy)[clust == "c2", ]) - cov(resid(lm0)[clust == "c2", ])

######## Modification (a)
za <- RegSDChybrid(y, clusters = clust, xLocal = x3, xGlobal = cbind(x1, x2), alternative = "a")
lma <- lm(za ~ clust + x1 + x2 + x3:clust)

# Now covariance matrices preserved within clusters
cov(resid(lmy)[clust == "c2", ]) - cov(resid(lma)[clust == "c2", ])

# If we estimate coef for x1 and x2 within clusters,
# they become identical and identical to global estimates
coef(lma)
coef(lm(za ~ clust + x1:clust + x2:clust + x3:clust))

######## Modification (c) with automatic calculation of alpha
# The result depends on the randomly generated data
# When the result is that alpha=1, modification (b) is equivalent
zc <- RegSDChybrid(y, clusters = clust, xLocal = x3, xGlobal = cbind(x1, x2), alternative = "c")
lmc <- lm(zc ~ clust + x1 + x2 + x3:clust)

# Preserved regression coef as above
coef(lmy) - coef(lmc)

# Again covariance matrices preserved within clusters
cov(resid(lmy)[clust == "c2", ]) - cov(resid(lmc)[clust == "c2", ])

# If we estimate coef for x1 and x2 within clusters,
# results are different from modification (a) above
coef(lmc)
coef(lm(zc ~ clust + x1:clust + x2:clust + x3:clust))


####################################################
# Make groups of clusters (d) and cluster pieces (e)
####################################################
clustGr <- paste("gr", ceiling(rep(1:3, each = 10)/2 + 0.1), sep = "")
clustP <- c("a", "a", rep("b", 28))

######## Modifications (c), (d) and (e)
zGrP <- RegSDChybrid(y, clusters = clust, clusterPieces = clustP, groupedClusters = clustGr,
                     xLocal = x3, xGroupedClusters = x2, xGlobal = x1, alternative = "c")

# Corresponding models by lm
lmGrP <- lm(zGrP ~ clust:clustP + x1 + x2:clustGr + x3:clust - 1)
lmY <- lm(y ~ clust:clustP + x1 + x2:clustGr + x3:clust - 1)

# Preserved regression coef
coef(lmY) - coef(lmGrP)

# Identical means within cluster pieces
aggregate(y, list(clust = clust, clustP = clustP), mean)
aggregate(zGrP, list(clust = clust, clustP = clustP), mean)
```

```
# Covariance matrix preserved
cov(resid(lmY)) - cov(resid(lmGrP))

# Covariance matrices preserved within clusters
cov(resid(lmY)[clust == "c2", ]) - cov(resid(lmGrP)[clust == "c2", ])

# Covariance matrices not preserved within cluster pieces
cov(resid(lmY)[clustP == "a", ]) - cov(resid(lmGrP)[clustP == "a", ])
```

RegSDCipso *Regression-based SDC Tools - Ordinary synthetic data (IPSO)*

Description

Implementation of equation 4 in the paper.

Usage

```
RegSDCipso(y, x = NULL, ensureIntercept = TRUE)
```

Arguments

`y`	Matrix of confidential variables
`x`	Matrix of non-confidential variables
`ensureIntercept`	Whether to ensure/include a constant term. Non-NULL x is subjected to `EnsureIntercept`

Details

Input matrices are subjected to `EnsureMatrix`.

Value

Generated version of y

Author(s)

Øyvind Langsrud

Examples

```
x <- matrix(1:5, 5, 1)
y <- matrix(rnorm(15) + 1:15, 5, 3)
ySynth <- RegSDCipso(y, x)

# Identical regression results
summary(lm(y[, 1] ~ x))
summary(lm(ySynth[, 1] ~ x))
```

```
# Identical covariance matrices
cov(y) - cov(ySynth)
cov(residuals(lm(y ~ x))) - cov(residuals(lm(ySynth ~ x)))
```

RegSDCnew *Regression-based SDC Tools - Scores from new data*

Description

Implementation of equation 12 in the paper.

Usage

```
RegSDCnew(y, yNew, x = NULL, doSVD = FALSE, ensureIntercept = TRUE)
```

Arguments

y	Matrix of confidential variables
yNew	Matrix of y-data for new scores
x	Matrix of non-confidential variables
doSVD	SVD when TRUE and QR when FALSE
ensureIntercept	Whether to ensure/include a constant term. Non-NULL x is subjected to EnsureIntercept

Details

doSVD has effect on decomposition of y and yNew. Input matrices are subjected to EnsureMatrix.

Value

Generated version of y

Author(s)

Øyvind Langsrud

Examples

```
x <- matrix(1:5, 5, 1)
y <- matrix(rnorm(15) + 1:15, 5, 3)

# Same as IPSO (RegSDCipso)
RegSDCnew(y, matrix(rnorm(15), 5, 3), x)

# Close to y
RegSDCnew(y, y + 0.001 * matrix(rnorm(15), 5, 3), x)
```

RegSDCromm   *Regression-based SDC Tools - Random orthogonal matrix masking (ROMM)*

Description

Implementation based on equations 11, 12 and 17 in the paper.

Usage

```
RegSDCromm(y, lambda = Inf, x = NULL, doSVD = FALSE, ensureIntercept = TRUE)
```

Arguments

`y`	Matrix of confidential variables
`lambda`	ROMM parameter
`x`	Matrix of non-confidential variables
`doSVD`	SVD when TRUE and QR when FALSE
`ensureIntercept`	Whether to ensure/include a constant term. Non-NULL x is subjected to `EnsureIntercept`

Details

doSVD has effect on decomposition of y. The exact behaviour of the method depends on the choice of the decomposition method because of the sequentially phenomenon mentioned in the paper. The similarity to the original data will tend to be highest for the first component. Input matrices are subjected to `EnsureMatrix`.

Value

Generated version of y

Author(s)

Øyvind Langsrud

Examples

```
x <- matrix(1:5, 5, 1)
y <- matrix(rnorm(15) + 1:15, 5, 3)

# Same as IPSO (RegSDCipso)
RegSDCromm(y, Inf, x)

# Close to IPSO
RegSDCromm(y, 100, x)

# Close to y
RegSDCromm(y, 0.001, x)
```

SuppressDec *Suppressed tabular data: Inner cell frequencies as decimal numbers*

Description

Assume that frequencies to be published, z, can be computed from inner frequencies, y, via z = t(x) %*% y, where x is a dummy matrix. Assuming correct suppression, this function will generate safe inner cell frequencies as decimal numbers.

Usage

```
SuppressDec(
  x,
  z = NULL,
  y = NULL,
  suppressed = NULL,
  digits = 9,
  nRep = 1,
  yDeduct = NULL,
  resScale = NULL,
  rmse = NULL,
  sparseLimit = 500
)
```

Arguments

x	Dummy matrix where the dimensions matches z and/or y input. Sparse matrix (Matrix package) is possible.
z	Frequencies to be published. All, only the safe ones or with suppressed as NA.
y	Inner cell frequencies (see details).
suppressed	Logical vector defining the suppressed elements of z.
digits	Output close to whole numbers will be rounded using digits as input to RoundWhole.
nRep	Integer, when >1, several y's will be generated. Extra columns in output.
yDeduct	Values to be subtracted from y and added back after the calculations. Can be used to perform the modulo method described in the paper (see examples).
resScale	Residuals will be scaled by resScale
rmse	Desired root mean square error (residual standard error). Will be used when resScale is NULL or cannot be used.
sparseLimit	Limit for the number of rows of a reduced x-matrix within the algorithm. When exceeded, a sparse algorithm is used (see IpsoExtra).

Details

This function makes use of ReduceX and RegSDCipso. It is not required that y consists of cell frequencies. A multivariate y or z is also possible. Then several values are possible as digits, resScale and rmse input.

Value

The inner cell frequencies as decimal numbers

Note

Capital letters, X, Y and Z, are used in the paper.

Author(s)

Øyvind Langsrud

Examples

```
# Same data as in the paper
z <- RegSDCdata("sec7z")
x <- RegSDCdata("sec7x")
y <- RegSDCdata("sec7y")  # Now z is t(x) %*% y
zAll <- RegSDCdata("sec7zAll")
zAllSupp <- RegSDCdata("sec7zAllSupp")
xAll <- RegSDCdata("sec7xAll")

# When no suppression, output is identical to y
SuppressDec(xAll, zAll, y)
SuppressDec(xAll, zAll)  # y can be seen in z

# Similar to Y* in paper (but other random values)
SuppressDec(x, z, y)

# Residual standard error forced to be 1
SuppressDec(x, z, y, rmse = 1)

# Seven ways of obtaining the same output
SuppressDec(x, z, rmse = 1)  # slower, y must be estimated
SuppressDec(x, y = y, rmse = 1)
SuppressDec(xAll, zAllSupp, y, rmse = 1)
SuppressDec(xAll, zAllSupp, rmse = 1)  # slower, y must be estimated
SuppressDec(xAll, zAll, y, is.na(zAllSupp), rmse = 1)
SuppressDec(xAll, zAll, suppressed = is.na(zAllSupp), rmse = 1)  # y seen in z
SuppressDec(xAll, y = y, suppressed = is.na(zAllSupp), rmse = 1)

# YhatMod4 and YhatMod10 in Table 2 in paper
SuppressDec(xAll, zAllSupp, y, yDeduct = 4 * (y%/%4), resScale = 0)
SuppressDec(xAll, zAllSupp, y, yDeduct = 10 * (y%/%10), rmse = 0)

# As data in Table 3 in paper (but other random values)
SuppressDec(xAll, zAllSupp, y, yDeduct = 10 * (y%/%10), resScale = 0.1)

# rmse instead of resScale and 5 draws
SuppressDec(xAll, zAllSupp, y, yDeduct = 10 * (y%/%10), rmse = 1, nRep = 5)
```

`Z2Yhat` *Suppressed tabular data: Yhat from X and Z*

Description

Implementation of equation 21 in the paper.

Usage

```
Z2Yhat(z, x, digits = 9)
```

Arguments

`z`	Z as a matrix
`x`	X as a matrix
`digits`	When non-NULL, output values close to whole numbers will be rounded using `digits` as input to `RoundWhole`.

Details

Generalized inverse is computed by `ginv`. In practise, the computations can be speeded up using reduced versions of X and Z. See `ReduceX`.

Value

Yhat as a matrix

Author(s)

Øyvind Langsrud

See Also

`IpsoExtra`

Examples

```
# Same data as in the paper
z <- RegSDCdata("sec7z")
x <- RegSDCdata("sec7x")
Z2Yhat(z, x)

# With y known, yHat can be computed in other ways
y <- RegSDCdata("sec7y")  # Now z is t(x) %*% y
fitted(lm(y ~ x - 1))
IpsoExtra(y, x, FALSE, resScale = 0)
```

Index