GSEAlm

April 19, 2010

`dfbetasPerGene` *Linear-Model Deletion Diagnostics for Gene Expression (or similar) Data Structures*

Description

This is an extension of standard linear-model diagnostics for use with gene-expression datasets, in which the same model was run simultaneously on each row of a response matrix.

Usage

```
dfbetasPerGene(lmobj)

CooksDPerGene(lmobj)

dffitsPerGene(lmobj)

Leverage(lmobj)
```

Arguments

`lmobj`	An object produced by `lmPerGene`.

Details

Deletion diagnostics gauge the influence of each observation upon model fit, by calculating values after removal of the observation and comparing to the complete-data version.

DFFITS$_i$ measures the distance on the response scale, between fitted values with and without observation y_i, at point i. The distance is normalized by the regression standard error and the point's leverage (see below).

Cook's D$_i$ is the square of the distance, in parameter space, between parameter estimates witn and without observation y_i, normalized and rescaled by standard errors and by a factor depending upon leverage.

DFBETAS$_{i,j}$ breaks the square root of Cook's D into its Euclidean components for each parameter j - but uses a somewhat different scaling function from Cook's D.

The leverage is the diagonal of the "hat matrix" $X'(X'X)^{-1}X'$. This measure provides the relative weight of observation y_i in the fitted value y-hat_i. Typically observations with extreme X values (or belonging to smaller groups if model variables are categorical) will have high leverage.

All these functions exist for standard regression, see `influence.measures`.

The functions described here are extensions for the case in which the response is a matrix, and the same linear model is run on each row separately.

For more details, see the references below.

All functions are implemented in matrix form, which means they run quite fast.

Value

`dfbetasPerGene` A G x n x p array, where G, n are the number of rows and columns in the input's expression matrix, respectively, and p the number of parameters in the linear model (including intercept)

`CooksDPerGene` A G x n matrix.

`dffitsPerGene` A G x n matrix.

`Leverage` A vector of length n, corresponding to the diagonal of the "hat matrix".

Note

The commonly-cited reference alert thresholds for diagnostic measures such as Cook's D and DFBETAS, found in older references, appear to be out of date. See LaMotte (1999) and Jensen (2001) for a more recent discussion. Our suggested practice is to inspect any samples or values that are visibly separate from the pack.

Author(s)

Robert Gentleman, Assaf Oron

References

Belsley, D. A., Kuh, E. and Welsch, R. E. (1980) Regression Diagnostics. New York: Wiley.

Cook, R. D. and Weisberg, S. (1982) Residuals and Influence in Regression. London: Chapman and Hall.

Williams, D. A. (1987) Generalized linear model diagnostics using the deviance and single case deletions. Applied Statistics *36*, 181-191.

Fox, J. (1997) Applied Regression, Linear Models, and Related Methods. Sage.

LaMotte, L. R. (1999) Collapsibility hypotheses and diagnostic bounds in regression analysis. Metrika 50, 109-119.

Jensen, D.R. (2001) Properties of selected subset diagnostics in regression. Statistics and Probability Letters 51, 377-388.

See Also

`influence.measures` for the analogous simple regression diagnostic functions

Examples

```
data(sample.ExpressionSet)
layout(1)
lm1 = lmPerGene( sample.ExpressionSet,~score+type)
CD = CooksDPerGene(lm1)
### How does the distribution of mean Cook's distances across samples look?
```

```
boxplot(log2(CD) ~ col(CD),names=colnames(CD),ylab="Log Cook's
Distance",xlab="Sample")
### There are a few gross individual-observation outliers (which is why we plot on the lo
### scale), but otherwise no single sample pops out as problematic. Here's
### one commonly-used alert level for problems:
lines(c(-5,30),rep(log2(2/sqrt(26)),2),col=2)


DFB = dfbetasPerGene(lm1)

### Looking for simultaneous two-effect outliers - 500 genes times 26
### samples makes 13000 data points on this plot

plot(DFB[,,2],DFB[,,3],main="DFBETAS for Score and Type (all genes)",xlab="Score Effect
Offset (normalized units)",ylab="Type Effect Offset (normalized units)",pch='+',cex=.5)
lines(c(-100,100),rep(0,2),col=2)
lines(rep(0,2),c(-100,100),col=2)

DFF = dffitsPerGene(lm1)
summary(apply(DFF,2,mean))

Lev = Leverage(lm1)
table(Lev)
### should have only two unique values because this is a dichotomous one-factor model
```

`getResidPerGene` *Row-by-Row Linear-Model Residuals for Gene Expression (or similar) Data Structures*

Description

This produces residuals of an identical linear model applied to each row of a gene expression matrix (or similar dataset). Computation speed is achieved via straightforward matrix algebra. Most commonly-used residual types are available.

Usage

```
getResidPerGene(lmobj, type = "extStudent")
```

Arguments

`lmobj`	An object produced by function `lmPerGene`.
`type`	A string indicating the type of residual requeseted (defaults to externally-Studentized).

Details

Types of residuals now available:

"response" Response residuals, observed minus fitted

"normalized" Response residuals divided by the estimated residual S.E.

"intStudent" Internally Studentized residuals, often referred to as "Standardized"

default Externally Studentized residuals, which can be used directly for outlier identification

Value

Returns a instance of ExpressionSet where the expression matrix contains the residuals. The phenoData are inherited from lmobj$eS.

Author(s)

Robert Gentleman, Assaf Oron

See Also

lmPerGene, resplot,dfbetasPerGene,influence.measures

Examples

```
data(sample.ExpressionSet)
lm1 = lmPerGene(sample.ExpressionSet,~sex)
r1 = getResidPerGene(lm1)
### now a boxplot of all residuals by sample
resplot(resmat=exprs(r1),fac=sample.ExpressionSet$sex)
### This plot is not very informative because of some gross outliers;
### try this instead
resplot(resmat=exprs(r1),fac=sample.ExpressionSet$sex,lims=c(-5,5))
```

---

gsealmPerm — *Nonparametric inference for linear models in Gene-Set-Enrichment Analysis (GSEA)*

Description

Provides permutation-based p-values for a main effect at the gene-set level, potentially adjusting for the effect of other variables via a linear model. This is a generalization and upgrade of gseattperm.

Usage

```
gsealmPerm(eSet, formula = "", mat, nperm, na.rm = TRUE,pooled=FALSE,...)
```

Arguments

eSet	An ExpressionSet object.
formula	An object of class formula (or one that can be coerced to that class), specifying only the right-hand side starting with the '~' symbol. The LHS is automatically set as the expression levels provided in eSet. The names of all predictors must exist in the phenotypic data of eSet. See more below in "Details".
mat	A 0/1 incidence matrix with each row representing a gene set and each column representing a gene. A 1 indicates membership of a gene in a gene set.
nperm	Number of permutations used to simulate the reference null distribution.
na.rm	Should missing observations be ignored? (passed on to lmPerGene)
pooled	Should variance be pooled across all genes? (passed on to lmPerGene)
...	Additional parameters passed on to GSNormalize.

Details

If a formula is provided, the permutation test permutes sample (i.e. column) labels, so essentially the effect is compared with the null distribution of effects for *each particular gene-set separately*. This neutralizes the impact of intra-sample correlations. If the formula contains two or more covariates, the effect of interest must be the first one in the formula. This effect's covariate values are permuted within each subgroup defined by identical values on all other covariates. This means, that the other covariates *must* be discrete, otherwise the analysis is meaningless. The effect of interest is the only one that can be continuous.

If a formula is *not* provided, a row-permutation test is performed on average expression levels. This test examines whether each gene-set is differentially expressed (on the average), compared with a permutation baseline of random gene-sets of the same size.

Value

A matrix with the same number of rows as 'mat' and two columns, "Lower" and "Upper". The "Lower" ("Upper") column gives the probability of seeing a t-statistic smaller (larger) than the observed. If 'mat' had row names, so will the output.

Warnings

1. Inference is *only* for the first term in the model. If you want inference for more terms, re-run the function on the same model, changing order of terms each time.

2. To repeat: the adjusting covariates (all terms except the first) have to be discrete. Adding a continuous covariate with unique values for most samples, may result in an infinite loop. However, you *can* put a continuous covariate as your first term.

Note

This function is a generic template for GSEA permutation tests. The particular type of GSEA statistic used is determined by `GSNormalize`, which is called by this function. Permutations are generated via repeated calls to `lmPerGene`.

Author(s)

Assaf Oron

See Also

`gseattperm`,`GSNormalize`, `lmPerGene`. The `GlobalAncova` package provides a generic F-test for model selection, while `gsealmPerm` can be used as a Wald test for the addition of a single covariate to the model.

Examples

```
data(sample.ExpressionSet)

### Generating random pseudo-gene-sets
fauxGS=matrix(sample(c(0,1),size=50000,replace=TRUE,prob=c(.9,.1)),nrow=100)

### inference for sex: sex is first term
sexPvals=gsealmPerm(sample.ExpressionSet,~sex+type,mat=fauxGS,nperm=40)
```

```
### inference for type: type is first term
typePvals=gsealmPerm(sample.ExpressionSet,~type+sex,mat=fauxGS,nperm=40,removeShift=TRUE)

### plotting the p-values; note that the effect direction depends upon
### factor level order (defaults to alphabetical)
layout(t(1:2))
### Sex p-values are center-heavy, typical when the effect is dominated
### by another effect
hist(sexPvals[,2],10,main="Sex Effect p-values",xlab="p-values for Male minus Female",xli
### The dominating effect is type, where there is a baseline shift in
### favor of controls
hist(typePvals[,1],10,main="Type Effect p-values",xlab="p-values for Case minus Control",

############
### Modeling type again - and now we add a baseline-shift removal (the 'removeShift' argu
typePvals1=gsealmPerm(sample.ExpressionSet,~type+sex,mat=fauxGS,nperm=40,removeShift=TRUE
### Modeling type again - and now the shift removal is by mean instead
### of the default median
typePvals2=gsealmPerm(sample.ExpressionSet,~type+sex,mat=fauxGS,nperm=40,removeShift=TRUE

### Now notice the differences between the 3 versions! This is a weird
### dataset indeed; it's also important to undrestand which research
### question you are trying to answer :)
hist(typePvals1[,1],10,main="Type Effect p-values",xlab="p-values for Case minus Control"
hist(typePvals2[,1],10,main="Type Effect p-values",xlab="p-values for Case minus Control"
```

`GSNormalize` *Aggregating and calculating expression statistics by Gene Set*

Description

Provides an interface for producing aggregate gene-set statistics, for gene-set-enrichment analysis (GSEA). The function is best suited for mean or rescaled-mean GSEA approaches, but is hopefully generic enough to enable other approaches as well.

Usage

```
GSNormalize(dataset, incidence, gseaFun = crossprod, fun1 = "/",
            fun2 = sqrt, removeShift=FALSE, removeStat=mean, ...)
identity(x)
one(x)
```

Arguments

`dataset`	a numeric matrix, typically of some gene-level statistics
`incidence`	0/1 incidence matrix indicating genes' membership in gene-sets
`gseaFun`	function name for the type of aggregation to take place, defaults to 'crossprod'. See 'Details'
`fun1`	function name for normalization, defaults to "/". See 'Details'
`fun2`	function name for scaling, defaults to 'sqrt'. See 'Details'

removeShift	logical: should normalization begin with a column-wise removal of the mean shift?
removeStat	(if above is TRUE) the column-wise statistic to be swept out of 'dataset'.
...	Additional arguments optionally passed on to 'gseaFun'.
x	any numerical value

Details

In gene-set-enrichment analysis (GSEA), the core step is aggregating (or calculating) gene-set-level statistics from gene-set statistics. This utility achieves the feat. It is tailored specifically for rescaled-sums of the type suggested by Jiang and Gentleman (2007), but is designed as a generic template that should other GSEA approaches. In such cases, at this moment users should provide their own version of 'gseaFun'.

The default will generate sums of gene-level values divided by the square-root of the gene-set size (in other words, gene-set means multiplied by the square-root of gene-set size). The arithmetic works like this:

gene-set stat = gseaFun(t(incidence),dataset),...) 'fun1' fun2(gene-set size).

In case there is a known (or suspected) overall baseline shift (i.e., the mass of gene-level stats is not centered around zero) it may be scientifically more meaningful to look for gene-set deviating from this baseline rather than from zero. In this case, you can set 'removeShift=TRUE'.

Also provided are the 'identity' function (identity = function(x) x), so that leaving 'gseaFun' and 'fun1' at their default and setting 'fun2 = identity' will generate gene-set means – and the 'one' function to neutralize the effect of both 'fun1' and 'fun2' (see note below).

Value

'GSNormalize' returns a matrix with the same number of rows as 'incidence' and the same number of columns as 'dataset' (if 'dataset' is a vector, the output will be a vector as well). The respective row and column names will carry through from 'dataset' and 'incidence' to the output.

'identity' simply returns x. 'one' returns the number 1.

Note

If you want to create your own GSEA function for 'gseaFun', note that it should receive the transposed incidence matrix as its first argument, and the gene-level stats as its second argument. In other words, both should have genes as rows. also, you can easily neutralize the effect of 'fun1', 'fun2' by setting "fun2 = one".

Author(s)

Assaf Oron

References

Z. Jiang and R. Gentleman, "Extensions to Gene Set Enrichment Analysis",Bioinformatics (23),306-313, 2007.

See Also

gsealmPerm, which relies heavily on this function. The function applyByCategory from the Category package has similar functionality and is preferable when the applied function is complicated. GSNormalize is better optimized for matrix operations.

Examples

```
data(sample.ExpressionSet)
lm1 = lmPerGene(sample.ExpressionSet,~sex+type)

### Generating random pseudo-gene-sets
fauxGS=matrix(sample(c(0,1),size=50000,replace=TRUE,prob=c(.9,.1)),nrow=100)

### "tau-stats" for gene-SET-level type effect, adjusting for sex
fauxEffects=GSNormalize(lm1$coefficients[3,]/sqrt(lm1$coef.var[3,]),incidence=fauxGS)

qqnorm(fauxEffects)
### diagonal line represents zero-shift null; note that it doesn't fit
abline(0,1,col=2)
### a better option may be to run a diagonal through the middle of the
### data (nonzero-shift null, i.e. type may have an effect but it is the
### same for all gene-sets); note that if any outlier shows, it is a purely random one!

abline(median(fauxEffects),1,col=4)

#### Now try with baseline-shift removal

fauxEffects=GSNormalize(lm1$coefficients[3,]/sqrt(lm1$coef.var[3,]),incidence=fauxGS,remo

qqnorm(fauxEffects)
abline(0,1,col=2)
```

`lmPerGene` *Fit linear model for each gene*

Description

For each gene, `lmPerGene` fits the same, user-specified linear model. It returns the estimates of the model parameters and their variances for each fitted model. The function uses matrix algebra so it is much faster than repeated calls to `lm`.

Usage

```
lmPerGene(eSet, formula, na.rm=TRUE,pooled=FALSE)
```

Arguments

`eSet`	An `ExpressionSet` object.
`formula`	an object of class `formula` (or one that can be coerced to that class), specifying only the right-hand side starting with the '~' symbol. The LHS is automatically set as the expression levels provided in `eSet`. The names of all predictors must exist in the phenotypic data of `eSet`.
`na.rm`	Whether to remove missing observations.
`pooled`	Whether to pool the variance calculation across all genes.

Details

This function efficiently computes the least squares fit of a linear regression to a set of gene expression values. We assume that there are `G` genes, on `n` samples, and that there are `p` variables in the regression equation. So the result is that `G` different regressions are computed, and various summary statistics are returned.

Since the independent variables are the same in each model fitting, instead of repeatedly fitting linear model for each gene, `lmPerGene` accelarates the fitting process by calculating the hat matrix $X(X'X)^(-1)X'$ first. Then matrix multiplication, and `solve` are to compute estimates of the model parameters.

Leaving the formula blank (the default) will calculate an intercept-only model, useful for generic pattern and outlier identification.

Value

A list with components:

`eS`	The `ExpressionSet` used in the model fitting.
`x`	The design matrix of the coded predictor variables.
`Hmat`	The Hat matrix.
`coefficients`	A matrix of dimension `p` by `G` containing the estimated model parameters.
`pooled`	Whether the variance was pooled (this affects "coef.var" and "tstat", but not "sigmaSqr").
`sigmaSqr`	A vector of length G containing the mean square error for that model, the sum of the residuals squared divided by `n - p`.
`coef.var`	A matrix of dimension `p` by `G` containing the estimated variances for the model parameters, for each regression.
`tstat`	A matrix of the same dimension as `coefficients`, containing the t-statistics for each model estimate. This is simply `coefficients` divided by the square root of `coef.var`, and is provided for convenience.

Author(s)

Robert Gentleman, Assaf Oron

See Also

`getResidPerGene` to extract row-by-row residuals; `gsealmPerm` for code that utilizes 'lmPerGene' for gene-set-enrichment analysis (GSEA); and `CooksDPerGene` for diagnostic functions on an object produced by 'lmPerGene'. Applying a by-gene regression in the manner performed here is a special case of a more generic linear-model framework available in the `GlobalAncova` package; our assumption here is equivalent to a diagonal covariance structure between genes, with unequal variances.

Examples

```
data(sample.ExpressionSet)
layout(1)
lm1 = lmPerGene(sample.ExpressionSet,~sex)
qqnorm(lm1$coefficients[2,]/sqrt(lm1$coef.var[2,]),main="Sample Dataset: Sex Effect by Ge
abline(0,1,col=2)
lm2 = lmPerGene(sample.ExpressionSet,~type+sex)
```

```
qqnorm(lm2$coefficients[2,]/sqrt(lm2$coef.var[2,]),main="Sample Dataset: Case vs. Control
abline(0,1,col=2)
```

---

resplot *Simple Graphical Summaries for Gene Set Enrichment Analysis (GSEA)*

Description

Diagnostic plots for GSEA. 'resplot' and 'restrip' group residuals (or expression levels) from a specific gene-set by sample. 'mnDiffPlot' shows mean expression differences for a dichotomous phenotype, by gene, for a specific gene set.

Usage

```
resplot(GSname = "All", resmat, incidence = dumminc(resmat), fac,
     atomic = "Gene", core.text = "Residuals by Sample",
     yname = "Standardized Residual", xname = "Sample ID",
     ID = colnames(resmat), lims = 0, gnames = levels(factor(fac)),
     prefix = "", horiz = FALSE, colour=5,pch='+',...)

restrip(GSname = "All", resmat, incidence = dumminc(resmat), fac,
   atomic = "Gene", core.text = "Residuals by Sample",
  yname = "Standardized Residual", xname = "Sample ID", ID = colnames(resmat),
  gnames = levels(factor(fac)), prefix = "", colour=c(2:4,6), resort=TRUE,
  horiz = FALSE, resort.fun=num.positive, pch='+', ...)

mnDiffPlot(GSname = "All", exprmat, incidence = dumminc(exprmat), fac,
  atomic = "Gene", core.text = paste("Mean Expression Difference by",atomic),
  yname="Log Expression Ratio", xname="Log Expression",
  gnames = levels(factor(fac)), prefix = "", fitline=FALSE, varsize=FALSE,
  reverse=FALSE, ...)
```

Arguments

GSname	Gene-set Name. See "Details".
resmat,exprmat	Numerical matrix with the values to be plotted. See "Details".
incidence	Gene-set 0/1 membership matrix
fac	The phenotypical variable to plot by. Must be discrete. For 'mnDiffPlot', must be dichotomous.
atomic	string identifying the meaning of rows in the data matrix. Defaults to "Gene".
core.text,gnames,prefix,xname,yname	strings controlling the text of main and axis captions
ID	Group names associated with the data matrix columns
lims	plotting limits for the response axis
horiz	logical: whether the boxplots or strips should be horizontal (defaults to FALSE)
colour	color of boxplot filling ('resplot') or symbols ('restrip')
pch	the plotting symbol

resort	('restrip' only) whether to sort groups for better visibility
resort.fun	('restrip' only) what function to sort groups by. Ignored unless 'resort==TRUE'. See stripchart documentation for more details
fitline	('mnDiffPlot' only) logical: whether a loess fit should be plotted
varsize	('mnDiffPlot' only) logical: whether symbol sizes should be proportional to (t-test style) standard errors
reverse	('mnDiffPlot' only)logical: whether the factor's order should be reversed so that the second level is on the x-axis rather than the first one
...	Additional graphical parameters passed on to the generic plotting functions.

Details

These functions provide simple graphical summaries for processed gene-expression data, or other similar datasets for which matrix form is useful. They are tailored predominantly for GSEA, but are useful in general as well.

'resplot' calls boxplot and 'restrip' calls stripchart; both summarize *all* data points from those rows in 'resmat' which are members in the gene-set specified by 'GSname'. The summary is by column. For each level of 'fac' there will be a separate pane.

'mnDiffPlot' calls plot; it plots the mean differences, by row, between columns belonging to the two groups specified by 'fac', as a function of the mean values for the first group alone. Each row translates to a single point on the graph. Again, the summary is only for rows indicated by 'GSname'.

For gene-set selective plots to properly work, the incidence matrix needs to have non-empty row names, and 'GSname' must match one of them.

If both 'GSname' and 'incidence' are left blank, automatic utilities are called which help generate a summary of the entire matrix, by column.

All functions plot a reference line signalling zero. 'mnDiffPlot' also optionally plots a loess fit for expression differences (if 'fitline=TRUE').

Note

One can use 'resplot'/'restrip' to plot raw expression values rather than residuals; it all depends on what's in the data matrix.

Author(s)

Assaf Oron

See Also

boxplot,plot,stripchart,par,GOmnplot

Examples

```
data(sample.ExpressionSet)
lm1 = lmPerGene(sample.ExpressionSet,~sex)
r1 = getResidPerGene(lm1)
### now a boxplot of all residuals by sample
resplot(resmat=exprs(r1),fac=sample.ExpressionSet$sex)
### This plot is not very informative because of some gross outliers;
```

```
### try this instead
resplot(resmat=exprs(r1),fac=sample.ExpressionSet$sex,lims=c(-5,5))

### stripchart for first 10 genes
restrip(resmat=exprs(r1)[1:10,],fac=sample.ExpressionSet$type,prefix="Not")

### note the wild trajectory of the loess fit:
mnDiffPlot(exprmat=exprs(sample.ExpressionSet),fac=sample.ExpressionSet$type,xname="Raw E
```

Index