# Package 'sva'

September 24, 2012

**Title** Surrogate Variable Analysis

**Version** 3.2.1

**Author** Jeffrey T. Leek <jleek@jhsph.edu>, W. Evan Johnson <wej@bu.edu>, Hilary S. Parker <hiparker@jhsph.edu>, Andrew E. Jaffe <ajaffe@jhsph.edu>, John D. Storey <jstorey@princeton.edu>,

**Description** The sva package contains functions for removing batch effects and other unwanted variation in high-throughput experiment. Specifically, the sva package contains functions for the identifying and building surrogate variables for high-dimensional data sets. Surrogate variables are covariates constructed directly from high-dimensional data (like gene expression/RNA sequencing/methylation/brain imaging data) that can be used in subsequent analyses to adjust for unknown, unmodeled, or latent sources of noise. The sva package can be used to remove artifacts in two ways: (1) identifying and estimating surrogate variables for unknown sources of variation in high-throughput experiments (Leek and Storey 2007 PLoS Genetics,2008 PNAS) and (2) directly removing known batch effects using ComBat (Johnson et al. 2007 Biostatistics). Removing batch effects and using surrogate variables in differential expression analysis have been shown to reduce dependence, stabilize error rate estimates, and improve reproducibility, see (Leek and Storey 2007 PLoS Genetics, 2008 PNAS or Leek et al. 2011 Nat. Reviews Genetics). Surrogate variable analysis and ComBat were developed in the context of microarray experiments, but may be used as a general tool for high throughput data sets where dependence may be involved.

**Maintainer** Jeffrey T. Leek <jleek@jhsph.edu>

**Depends** R (>= 2.8), corpcor, mgcv

**Suggests** limma,pamr,bladderbatch

**License** Artistic-2.0

**biocViews** Microarray,Statistics,Preprocessing,MultipleComparisons

## R topics documented:

---

`ComBat` *Adjust for batch effects using empirical Bayes framework.*

---

## Description

ComBat allows users to adjust for batch effects in datasets where the batch covariate is known, using methodology described in Johnson et al. 2007. It uses either parametric or non-parametric empirical Bayes frameworks for adjusting data for batch effects. Users are returned an expression matrix that has been corrected for batch effects.

## Usage

```
ComBat(dat, batch, mod,numCovs=NULL, par.prior=TRUE,prior.plots=FALSE)
```

## Arguments

| | |
|---|---|
| `dat` | Genomic measure matrix (dimensions probe x sample) - for example, expression matrix |
| `batch` | Batch covariate (multiple batches allowed) |
| `mod` | Model matrix for outcome of interest and other covariates besides batch |
| `numCovs` | The column numbers of the variables in mod to be treated as continuous variables (otherwise all covariates are treated as factors) |
| `par.prior` | (Optional) TRUE indicates parametric adjustments will be used, FALSE indicates non-parametric adjustments will be used |
| `prior.plots` | (Optional) TRUE give prior plots with black as a kernel estimate of the empirical batch effect density and red as the parametric estimate |

## Details

ComBat can be applied to genomic measures when batch is known to remove the effect of batch on the data using an empirical Bayesian framework. It was described in Johnson et al. 2007.

## Value

A probe x sample genomic measure matrix, adjusted for batch effects.

## Author(s)

W. Evan Johnson <evan@stat.byu.edu>

## References

Johnson WE, Li C, and Rabinovic A (2007) Adjusting batch effects in microarray expression data using empirical Bayes methods. Biostatistics 8:118-27.http://www.ncbi.nlm.nih.gov/pubmed/16632515

## See Also

`sva`,`irwsva.build`,`twostepsva.build`, `num.sv`

## Examples

```
## Not run:

## Load data
library(bladderbatch)
data(bladderdata)

## Obtain phenotypic data
pheno = pData(bladderEset)
edata = exprs(bladderEset)
batch = pheno$batch
mod = model.matrix(~as.factor(cancer), data=pheno)

## Correct for batch using ComBat
combat_edata = ComBat(dat=edata, batch=batch, mod=mod, par.prior=TRUE, prior.plots=FALSE)


## End(Not run)
```

---

`f.pvalue` *Quickly calculate F-test p-values*

---

## Description

Calculate p-values from a parametric F-test comparing the models mod and mod0 for each row of the data set dat.

## Usage

```
f.pvalue(dat,mod,mod0)
```

## Arguments

| | |
|---|---|
| `dat` | A m genes by n arrays matrix of expression data |
| `mod` | A n by k model matrix corresponding to the primary model fit (see model.matrix) |
| `mod0` | A n by k0 model matrix corresponding to the null model to be compared to mod. |

## Details

The data for test i should be in the ith row of dat, if mod0 is null, the first column of mod is used as the null model.

## Value

| | |
|---|---|
| `p` | A vector of p-values one for each row of dat. |

## Author(s)

Jeffrey T. Leek <jleek@jhsph.edu>, John Storey <jstorey@princeton.edu>

## Examples

```
## Not run:
## Load data
library(bladderbatch)
data(bladderdata)

## Obtain phenotypic data
pheno = pData(bladderEset)
edata = exprs(bladderEset)
batch = pheno$batch
mod = model.matrix(~as.factor(cancer), data=pheno)
mod0 = model.matrix(~1, data=pheno)

#Calculate the p-values
p <- f.pvalue(edat,mod,mod0)
hist(p)


## End(Not run)
```

---

`fsva` *Single sample surrogate variable correction for prediciton problems.*

---

## Description

fsva corrects training databases and performs "single-sample" correction on new samples for prediction problems. The effect of surrogate variables is removed from the training database which can then be used to build a predictor. The new samples are corrected individually to account for batch effects when the group status is unknown.

## Usage

```
fsva(dbdat,mod,sv,newdat=NULL,method=c("fast","exact"))
```

## Arguments

| | |
|---|---|
| `dbdat` | A m genes by n arrays matrix of expression data from the database/training data |
| `mod` | The model matrix for the terms included in the analysis for the training data |
| `sv` | The surrogate variable object created by running sva on dbdat using mod. |
| `newdat` | (Optional) A set of test samples to be adjusted using the training database |
| `method` | If method = "fast" then the online SVD is used, this may introduce slight bias. If method="exact" the exact SVD is calculated, but will be slower. |

## Details

Frozen surrogate variable analysis (fsva) can be applied to remove batch effects for prediction problems.

## Value

A list containing:

| | |
|---|---|
| db | An adjusted version of the training database where the effect of batch/expression heterogeneity has been removed) |
| new | An adjusted version of the new samples, adjusted one at a time using the fsva methodology. |
| newsv | The surrogate variables on the new samples |

## Author(s)

Jeffrey T. Leek <jleek@jhsph.edu>, Hilary S. Parker <hiparker@jhsph.edu>

## References

sva Vignette http://www.biostat.jhsph.edu/~jleek/sva

## See Also

sva,irwsva.build,twostepsva.build, num.sv

## Examples

```
## Not run:

## Load data
library(bladderbatch)
library(pamr)
data(bladderdata)

## Obtain phenotypic data
pheno = pData(bladderEset)
edata = exprs(bladderEset)
batch = pheno$batch
mod = model.matrix(~as.factor(cancer), data=pheno)

## Build the training and test sets
set.seed(12354)
trainIndicator = sample(1:57,size=30,replace=F)
testIndicator = (1:57)[-trainIndicator]

trainData = edata[,trainIndicator]
testData = edata[,testIndicator]

trainPheno = pheno[trainIndicator,]
testPheno = pheno[testIndicator,]

# Fit the sva model to the training set
trainMod = model.matrix(~cancer,data=trainPheno)
trainMod0 = model.matrix(~1,data=trainPheno)
trainSv = sva(trainData,trainMod,trainMod0)

#fsva correct and train
fsvaobj = fsva(dbdat=trainData,mod=trainMod,sv=trainSv,newdat=testData)
mydataSv = list(x=fsvaobj$db,y=trainPheno$cancer)
```

```
mytrainSv = pamr.train(mydataSv)
table(pamr.predict(mytrainSv,fsvaobj$new,threshold=1),testPheno$cancer)


## End(Not run)
```

---

`irwsva.build` *Build surrogate variables with an iterative algorithm from gene expression and model data*

---

## Description

Construct a specified number of surrogate variables from a gene expression data set and a fixed model.

## Usage

```
irwsva.build(dat, mod, mod0=NULL,n.sv,B=5)
```

## Arguments

| | |
|---|---|
| `dat` | A m genes by n arrays matrix of expression data |
| `mod` | A n by k model matrix corresponding to the primary model fit (see model.matrix) |
| `mod0` | A n by k0 model matrix corresponding to the null model to be compared to mod. |
| `n.sv` | The number of surrogate variables to construct. |
| `B` | The number of iterations of the algorithm to perform. |

## Details

The IRW-SVA estimation algorithm is described in Leek and Storey (2008).The basic idea is to estimate surrogate variables based on the subset of rows affected by unmodeled dependence, but not affected by the main variable parameterized in mod.

## Value

A list containing:

| | |
|---|---|
| `sv` | A n by n.sv matrix where each column is a distinct surrogate variable (the main quantity of interest) |
| `pprob.gam` | A vector with the posterior probability estimates that each row is affected by dependence. |
| `pprob.b` | A vector with the posterior probabiliity estimates that each row is affected by the variables in mod, but not in mod0. |
| `n.sv` | The number of suggorate variables estimated. |

## Author(s)

Jeffrey T. Leek <jleek@jhsph.edu>, John Storey <jstorey@princeton.edu>

## References

Leek JT and Storey JD. (2008) A general framework for multiple testing dependence. Proceedings of the National Academy of Sciences, 105: 18718-18723. http://www.biostat.jhsph.edu/~jleek/publications.html

Leek JT and Storey JD. (2007) Capturing heterogeneity in gene expression studies by surrogate variable analysis. PLoS Genetics, 3: e161. http://www.biostat.jhsph.edu/~jleek/publications.html

sva Vignette http://www.biostat.jhsph.edu/~jleek/sva/

## See Also

sva, num.sv, twostepsva.build,ComBat

## Examples

```
## Not run:
## Load data
library(bladderbatch)
data(bladderdata)

## Obtain phenotypic data
pheno = pData(bladderEset)
edata = exprs(bladderEset)
batch = pheno$batch
mod = model.matrix(~as.factor(cancer), data=pheno)

## Construct the surrogate variables
svaobj <- irwsva.build(edata,mod,mod0,n.sv=1)

## Include them in a downstream analysis

mod.sv <- cbind(mod,svaobj$sv)
mod0.sv <- cbind(mod0,svaobj$sv)
adjusted.pvals <- f.pvalue(edata,mod.sv,mod0.sv)



## End(Not run)
```

---

num.sv *Estimate number of surrogate variables to include in analysis*

---

## Description

Estimate number of important surrogate variables from a gene expression data set.

## Usage

```
num.sv(dat, mod,method=c("be","leek"),vfilter=NULL, B=20, sv.sig=0.10,seed=NULL)
```

## Arguments

| | |
|---|---|
| dat | A m genes by n arrays matrix of expression data |
| mod | A n by k model matrix corresponding to the primary model fit (see model.matrix) |
| method | The method to use for estimating surrogate variables, for now there is only one option (based ib Buja and Eyuboglu 1992). |
| vfilter | The number of most variable genes to use when building SVs, must be between 100 and m |
| B | The number of null iterations to perform. Only used when method="be" |
| sv.sig | The significance cutoff for eigengenes. Only used when method="be" |
| seed | A numeric seed for reproducible results. Optional, only used when method="be" |

## Details

The model matrix should include a column for an intercept. num.sv estimates the number of surrogate variables to include in the analysis as described in Leek and Storey (2007), based on the permutation test of Buja and Eyuboglu (1992).

## Value

A list containing:

| | |
|---|---|
| n.sv | The number of significant surrogate variables |

## Author(s)

Jeffrey T. Leek <jleek@jhsph.edu>, John Storey <jstorey@princeton.edu>

## References

Buja A and Eyuboglu N. (1992) Remarks on parrallel analysis. Multivariate Behavioral Research, 27(4), 509-540

Leek JT and Storey JD. (2008) A general framework for multiple testing dependence. Proceedings of the National Academy of Sciences, 105: 18718-18723. http://www.biostat.jhsph.edu/~jleek/publications.html

Leek JT and Storey JD. (2007) Capturing heterogeneity in gene expression studies by surrogate variable analysis. PLoS Genetics, 3: e161. http://www.biostat.jhsph.edu/~jleek/publications.html

sva Vignette http://www.biostat.jhsph.edu/~jleek/sva/

## See Also

sva, irwsva.build,twostepsva.build

## Examples

```
## Not run:

## Load data
library(bladderbatch)
data(bladderdata)
```

```
## Obtain phenotypic data
pheno = pData(bladderEset)
edata = exprs(bladderEset)
batch = pheno$batch
mod = model.matrix(~as.factor(cancer), data=pheno)
mod0 = model.matrix(~1, data=pheno)

## Calculate the number of surrogate variables
xx <- num.sv(edata,mod)
xx


## End(Not run)
```

---

sva *Estimate surrogate variables with an iterative algorithm from gene expression and model data*

---

## Description

Estimate surrogate variables are estimated using either the iteratively re-weighted surrogate variable analysis algorithm of Leek and Storey (2008) or the two-step algorithm of Leek and Storey (2007).

## Usage

```
sva(dat, mod, mod0 = NULL,n.sv=NULL,method=c("irw","two-step"),vfilter=NULL, B=5, numSVmethod
```

## Arguments

| | |
|---|---|
| dat | A m genes by n arrays matrix of expression data |
| mod | A n by k model matrix corresponding to the primary model fit (see model.matrix) |
| mod0 | A n by k0 model matrix corresponding to the null model to be compared to mod. |
| n.sv | Optional. The number of surrogate variables to estimate, can be estimated using the num.sv function |
| method | Choose between the iteratively re-weighted or two-step surrogate variable estimation algorithms. |
| vfilter | The number of most variable genes to use when building SVs, must be between 100 and m |
| B | The number of iterations of the algorithm to perform. |
| numSVmethod | The method for determining the number of surrogate variables to use |

## Details

Surrogate variable estimates are formed based on the algorithms in Leek and Storey (2007,2008). Surrogate variables can be included in a significance analysis to reduce dependence and confounding.

## Value

A list containing:

| | |
|---|---|
| sv | A n by n.sv matrix where each column is a distinct surrogate variable (the main quantity of interest) |
| pprob.gam | A vector with the posterior probability estimates that each row is affected by dependence. |
| pprob.b | A vector with the posterior probabiliity estimates that each row is affected by the variables in mod, but not in mod0. |
| n.sv | The number of suggorate variables estimated. |

## Author(s)

Jeffrey T. Leek <jleek@jhsph.edu>, John Storey <jstorey@princeton.edu>

## References

Leek JT and Storey JD. (2008) A general framework for multiple testing dependence. Proceedings of the National Academy of Sciences, 105: 18718-18723. http://www.biostat.jhsph.edu/~jleek/publications.html

Leek JT and Storey JD. (2007) Capturing heterogeneity in gene expression studies by surrogate variable analysis. PLoS Genetics, 3: e161. http://www.biostat.jhsph.edu/~jleek/publications.html

sva Vignette http://www.biostat.jhsph.edu/~jleek/sva/

## See Also

irwsva.build,twostepsva.build, num.sv,ComBat,fsva

## Examples

```
## Not run:

## Load data
library(bladderbatch)
data(bladderdata)

## Obtain phenotypic data
pheno = pData(bladderEset)
edata = exprs(bladderEset)
batch = pheno$batch
mod = model.matrix(~as.factor(cancer), data=pheno)
mod0 = model.matrix(~1, data=pheno)


## Construct the surrogate variables
svaobj <- sva(edata,mod,mod0,method="irw")


## Include them in a downstream analysis

mod.sv <- cbind(mod,svaobj$sv)
mod0.sv <- cbind(mod0,svaobj$sv)
adjusted.pvals <- f.pvalue(dat,mod.sv,mod0.sv)
```

```
## End(Not run)
```

---

`twostepsva.build` *Build surrogate variables from gene expression and model data*

---

## Description

Construct a specified number of surrogate variables from a gene expression data set based on the two-step algorithm of Leek and Storey (2007).

## Usage

```
twostepsva.build(dat, mod, n.sv)
```

## Arguments

| | |
|---|---|
| `dat` | A m genes by n arrays matrix of expression data |
| `mod` | A n by k model matrix corresponding to the primary model fit (see model.matrix) |
| `n.sv` | The number of surrogate variables to construct. |

## Details

The SVA estimation algorithm is described in Leek and Storey (2007).The basic idea is to estimate surrogate variables based on the subset of rows affected by unmodeled dependence.

## Value

A list containing:

| | |
|---|---|
| `sv` | A n by n.sv matrix where each column is a distinct surrogate variable (the main quantity of interest) |
| `pprob.gam` | A vector indicating whether each row was used in the building of the surrogate variables. 1= row used, 0=not used. |
| `pprob.b` | Null for two-step SVA, see irwsva.build for more info. |
| `n.sv` | The number of suggorate variables estimated. |

## Author(s)

Jeffrey T. Leek <jleek@jhsph.edu>, John Storey <jstorey@princeton.edu>

## References

Leek JT and Storey JD. (2007) Capturing heterogeneity in gene expression studies by surrogate variable analysis. PLoS Genetics, 3: e161. http://www.biostat.jhsph.edu/~jleek/publications.html

sva Vignette http://www.biostat.jhsph.edu/~jleek/sva/

## See Also

sva, num.sv,irwsva.build,ComBat

## Examples

```
## Not run:
## Load data
library(bladderbatch)
data(bladderdata)

## Obtain phenotypic data
pheno = pData(bladderEset)
edata = exprs(bladderEset)
batch = pheno$batch
mod = model.matrix(~as.factor(cancer), data=pheno)
mod0 = model.matrix(~1, data=pheno)

## Construct the surrogate variables
svaobj <- twostepsva.build(edata,mod,n.sv=1)

## Include them in a downstream analysis
mod.sv <- cbind(mod,svaobj$sv)
mod0.sv <- cbind(mod0,svaobj$sv)
adjusted.pvals <- f.pvalue(dat,mod.sv,mod0.sv)



## End(Not run)
```

# Index