# Package 'shuffle'

May 2, 2016

**Type** Package

**Title** The Shuffle Estimator for Explainable Variance

**Version** 1.0.1

**Date** 2016-4-24

**Author** Yuval Benjamini

**Maintainer** Yuval Benjamini <yuvalbenj@gmail.com>

**Description** Implementation of the shuffle estimator, a non-
parametric estimator for signal and noise variance under mild noise correlations.

**License** GPL (>= 2.0)

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2016-05-02 06:41:26

## R topics documented:

shuffle-package . . . . . . . . . . 2
design_vec . . . . . . . . . . 3
estimateShuffle . . . . . . . . . . 4
fMRI_responses . . . . . . . . . . 5
getAveraging . . . . . . . . . . 5
getNormalizer . . . . . . . . . . 6
MSbetAvg . . . . . . . . . . 8
prediction_res . . . . . . . . . . 9
prepareShuffle . . . . . . . . . . 9

**Index** **11**

shuffle-package *The Shuffle Estimator for Explainable Variance*

## Description

This package implements the algorithms underlying the shuffle estimators, variance estimators for one-way analysis of variance designs. The estimators can overcome correlated noise by recomputing the mean-square-between statistics on a permuted version of the data. The permutations should preserve the noise covariance matrix, but a parametric model for the noise covariance is not necessary. For more details see Benjamini and Yu, and here http://statweb.stanford.edu/~yuvalben.

Two functions implement the important stages of estimation:
prepareShuffle(design_vec, premutation), which preprocess the design and computes the normalization constant for a given permutation.
estimateShuffle(response_vec, prepare), which estimates variances and effect sizes for a specific data vector.

## Details

| | |
|---|---|
| Package: | shuffle |
| Type: | Package |
| Version: | 1.0.1 |
| Date: | 2013-4-24 |
| License: | GPL (>= 2) |

## Author(s)

Yuval Benjamini <yuvalbenj@gmail.com>

## References

Benjamini and Yu (2013), "The shuffle estimator for explainable variance in fMRI experiments", Annals of Applied Statistics 7 (4) http://projecteuclid.org/euclid.aoas/1387823308

## Examples

```
data(design_vec,fMRI_responses,prediction_res)

# Make example shorter - for paper example use T = ncol(fMRI_responses)
T = 156*4
fMRI_responses_sm = fMRI_responses[,1:T]
design_sm = design_vec[1:T]
permutation = rev(1:T)
```

```
prep_shuffle = prepareShuffle(design_sm,permutation)

var_explained = numeric(nrow(fMRI_responses_sm))
for (i in 1:nrow(fMRI_responses_sm)) {
    var_explained[i] = estimateShuffle(fMRI_responses_sm[i,],prep_shuffle)$effect
}

plot(var_explained, pmax(prediction_res,0)^2,
    xlim = c(0,0.7), ylim = c(0,0.7),
    xlab = "Explainable variance", ylab = "Corr^2")
abline(0,1,col=4)
```

---

## design_vec *The design for an fMRI experiment*

---

### Description

The design vector for the validation data in an fMRI experiment. At time t the image design_vec[t] was shown.

### Usage

```
data(design_vec)
```

### Format

The format is: num [1:1560] 2 3 6 1 11 2 2 7 2 10 ...

### Details

The design vector for the validation data in an fMRI experiment. The experiment consisted of 1560 timeframes, 120 images each repeated 13 times. The imshrd were organized into 10 separate blocks, each repeating 12 images.

### Source

Kay, Naselaris, Prenger and Gallant (2008), "Identifying natural images from human brain activity"

### References

Benjamini and Yu (2013), "The shuffle estimator for explainable variance"

### Examples

```
data(design_vec)
plot(design_vec,xlab = "event", ylab = "treatment", main="Design of the full experiment" )

plot(design_vec[1:120],xlab = "event",ylab= "treatment",main="Design of a single block")
```

## `estimateShuffle` *Calculate the shuffle estimators*

### Description

estimateShuffle estimates the following quantities for a response vector: the signal variance (signalVar), the noise variance (noiseVar), the total variance (YVar), and the explainable variance (effect). Inputs to the function are the response vector, and a preprocessing structure (the output of prepareShuffle) which holds the design, the shuffle permutation, and the calculated normalizer.

### Usage

```
estimateShuffle(dat, prep, neg = FALSE)
```

### Arguments

| | |
|---|---|
| `dat` | A vector of reponses - should be of the same size as the design vector and the shuffle permutation. |
| `prep` | The output of prepareShuffle; includes the design, the shuffling permuation, and a normalizer. |
| `neg` | If neg=FALSE does not allow the signal variance to get arbitrary negative values, but instead sets signal variance to -1e-05. |

### Details

estimateShuffle compares the mean-square-between of the data to the mean-square-between of the permuted data, the difference being the scaled noise variance. Effect size is the ratio between the estimated signal data and the estimated total variance.

### Value

| | |
|---|---|
| `signalVar` | The estimated variance of the signal |
| `noiseVar` | The estimated variance of the noise |
| `YVar` | The estimated total variance |
| `effect` | The proportion of explainable variance (signalVar/Yvar) |

### Author(s)

Yuval Benjamini

### References

Benjamini and Yu (2013), "The shuffle estimator for explainable variance in fMRI experiments".

## `fMRI_responses` *Responses for 30 voxels (of V1) to 1560 stimuli.*

### Description

A 30x1560 data matrix consisting of a sample of 30 fMRI responses from the visual cortex V1 of a human subject, to a sequence of natural images. The treatment index is found in design_vec.

### Usage

```
data(fMRI_responses)
```

### Format

A 30x1560 matrix of real values. Each row corresponds to a different voxel.

### Details

Measurement were recorded as a person was watching a sequence of natural images. Each column corresponds to a displayed image; each row corresponds to the response of a single voxel in the fMRI scan. This data consists of a small subset of the voxels in V1 from the original scans. The data was recorded in the Gallant lab at UC Berkeley.

### Source

Kay, Naselaris, Prenger and Gallant (2008), "Identifying natural images from human brain activity"

### References

Benjamini and Yu (2013), "The shuffle estimator for explainable variance in fMRI experiments"

## `getAveraging` *Convert design into averaging matrices.*

### Description

getAveraging(des) converts a design (either a vector or a matrix) into averaging matrix notation (from the paper). For a response vector Y, (B Y)[t] is the mean of all responses corresponding to the treatment at time t, and (B-G)Y [t] is the averaged-removed treatment mean.

### Usage

```
getAveraging(des)
```

## Arguments

| | |
|---|---|
| des | Either a vector or a matrix representation of design (see designVec2Mat and designMat2Vec). |

## Value

A list with components

| | |
|---|---|
| m | The number of treatments |
| ns | An m-length vector with the number of repeats for each treatment. For balanced designs with n repeats, ns=rep(n,m) |
| B | The averaging matrix according to the design |
| G | 1/T for T the number of measurements |

## Author(s)

Yuval Benjamini

## References

Benjamini and Yu (2013).

## Examples

```
data(design_vec)

design_avg = getAveraging(design_vec)
rand_resp = rnorm(length(design_vec))

global_mean = mean(rand_resp[design_vec != 0 ])
first_treatment_mean=  mean(rand_resp[design_vec == design_vec[1]])
cat((design_avg$B %*% rand_resp)[1], first_treatment_mean )

cat(((design_avg$B-design_avg$G) %*% rand_resp)[1], first_treatment_mean- global_mean)
```

---

`getNormalizer` *getNormalizer*

---

## Description

Computes the normalizer 1/(1-alpha) for a given design and permutation. The shuffle estimator is [MSbet(Y) - MSbet(PY)]*normalizer. \ We prefer the normalizer to be close to 1.

## Usage

```
getNormalizer(avgmat, perm)
```

## Arguments

| | |
|---|---|
| `avgmat` | The output of getAveraging. |
| `perm` | The shuffling permutation. |

## Details

Under balanced designs, the normalizer = 1/[1-alpha]. More generally, we call facA = 1 and facB = alpha(design, permutation).

## Value

| | |
|---|---|
| `norm` | The value by which to correct the difference of variances [1/(facA-facB)] |
| `facA` | The signal coefficient of the original design, should be 1 |
| `facB` | The signal variance coefficient of the permuted design |

## Author(s)

Yuval Benjamini

## References

Benjamini and Yu (2013).

## Examples

```
data(design_vec)

# Make example shorter - for paper example use T = ncol(fMRI_responses) = 156*10
T = 156*4
design_sm = design_vec[1:T]

identity_perm = 1:T
reverse_perm = rev(identity_perm)
shift_perm = c(2:T, 1)

design_avg = getAveraging(design_sm)
identity_norm = getNormalizer(design_avg, identity_perm)
print('For the identity, we cannot get an estimator')
print(sprintf('facA(1) %.3f, facB(alpha) %.3f, normalizer %.3f ',
    identity_norm$facA, identity_norm$facB, identity_norm$norm))

reverse_norm = getNormalizer(design_avg, reverse_perm)
print('For the reverse, we get a normalizer close to 1 ')
print(sprintf('facA(1) %.3f, facB(alpha) %.3f, normalizer %.3f ',
    reverse_norm$facA, reverse_norm$facB, reverse_norm$norm))

shift_norm = getNormalizer(design_avg, shift_perm)
print('The shift mixes across blocks. The normalizer is smaller, but assumptions may not hold')
print(sprintf('facA(1) %.3f, facB(alpha) %.3f, normalizer %.3f ',
    shift_norm$facA, shift_norm$facB, shift_norm$norm))
```

---

MSbetAvg *Calculate Mean-square-between*

---

## Description

MSbetAvg calculates the mean-square-between contrast according to the design vector. Responses for each condition are averaged, and the sample variance is calculated for these averages.

## Usage

```
MSbetAvg(dat, avgmat)
```

## Arguments

| | |
|---|---|
| dat | The vector of measurements on which the constrast is computed. |
| avgmat | The design parameters, as extracted by getAveraging(). |

## Value

The value of the quadratic contrast computed on the data vector.

## Author(s)

Yuval Benjamini

## Examples

```
data(fMRI_responses,design_vec)

msbet = MSbetAvg(fMRI_responses[1,], getAveraging(design_vec))

# Compute same value using "aov" when design is balanced ...
ns =tapply(design_vec,design_vec, length)
# (check that design is balanced)
stopifnot(length(unique(ns))==1)

m = length(unique(design_vec))

aov_sum = summary(aov(fMRI_responses[1,] ~ factor(design_vec)))
ss_bet = aov_sum[[1]][1,2]
# In unbalanced designs, each example should require more care...
msbet_aov = (ss_bet / ns[1] )/(m-1)

cat(msbet, msbet_aov)
```

## prediction_res *Prediction results for V1 voxels as generated by the Gallant lab in UC Berkeley and published in Kay et al. (2008).*

### Description

The correlation between measured response and predicted response on validation data for 1250 V1 voxels. The prediction algorithms are described in detail in Kay et al, 2008. We compare these prediction results to the explainable variance estimated with the shuffle estimator.

### Usage

```
data(prediction_res)
```

### Format

A numerical vector of correlation coefficients.

### Source

Kay, Naselaris, Prenger and Gallant (2008), "Identifying natural images from human brain activity"

### References

Benjamini and Yu (2013)

## prepareShuffle *Prerprocess for the shuffle estimator*

### Description

prepareShuffle computes the averaging matrices and normalizing constants for the shuffle estimator. It can be run once for all data vectors sharing the design.

### Usage

```
prepareShuffle(des, perm)
```

### Arguments

| | |
|---|---|
| `des` | A design vector or matrix |
| `perm` | The shuffling permutation |

## Value

| | |
|---|---|
| `m` | The number of treatments |
| `ns` | An m-length vector with the number of repeats for each treatment. For balanced designs with n repeats, ns=rep(n,m) |
| `B` | The averaging matrix according to the design |
| `G` | 1/T for T the number of measurements |
| `norm` | The value by which to correct the difference of variances [1/(facA-facB)] |
| `facA` | The signal coefficient of the original design |
| `facB` | The signal variance coefficient of the permuted design |

## Author(s)

Yuval Benjamini

## References

Benjamini and Yu (2013)

## See Also

getAverage getNormalizer

# Index

∗Topic **datasets**
design_vec, 3
fMRI_responses, 5
prediction_res, 9
∗Topic **design**
shuffle-package, 2
∗Topic **nonparametric**
shuffle-package, 2
∗Topic **package**
shuffle-package, 2

design_vec, 3

estimateShuffle, 4

fMRI_responses, 5

getAveraging, 5
getNormalizer, 6

MSbetAvg, 8

prediction_res, 9
prepareShuffle, 9

shuffle *(*shuffle-package*)*, 2
shuffle-package, 2