

# Package ‘HeritSeq’

February 11, 2019

**Title** Heritability of Gene Expression for Next-Generation Sequencing

**Version** 1.0.1

**Date** 2019-02-11

**Author** W. Jenny Shi [aut, cre],  
Pamela Russell [aut],  
Pratyaydipta Rudra [aut, cre],  
Brian Vestal [aut, cre],  
Katerina Kechris [aut],  
Laura Saba [aut]

**Maintainer** W. Jenny Shi <wjennyshi@gmail.com>

**Description** Statistical framework to analyze heritability of gene expression based on next-generation sequencing data and simulating sequencing reads. Variance partition coefficients (VPC) are computed using linear mixed effects and generalized linear mixed effects models. Compound Poisson and negative binomial models are included. Reference: Rudra, Pratyaydipta, et al. "Model based heritability scores for high-throughput sequencing data." BMC bioinformatics 18.1 (2017): 143.

**BugReports** <https://github.com/KechrisLab/HeritSeq/issues>

**Depends** R (>= 3.2.3)

**License** GPL-2

**Encoding** UTF-8

**LazyData** true

**Imports** cplm, DESeq2, lme4, pbapply, tweedie, MASS,  
SummarizedExperiment

**Suggests** glmmADMB

**Additional\_repositories** <http://glmmadmb.r-forge.r-project.org/repos>

**RoxygenNote** 5.0.1

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2019-02-11 21:13:29 UTC

## R topics documented:

computeVPC.CP	2
computeVPC.NB	3
fit.CP	4
fit.NB	5
fitComputeVPC.lmer	6
getBootCI	7
getReadMatrix.CP	8
getReadMatrix.NB	9
para_cp	10
para_nb	10
simData	10
simData_voom	11
simData_vst	11
strains	11
weights_voom	12
<b>Index</b>	<b>13</b>

---

computeVPC.CP	<i>Calculate the compound Poisson (CP) variance partition coefficient (VPC) for one or more features.</i>
---------------	---

---

### Description

Calculate the CP VPC for one or more features following the model fitting function fit.CP().

### Usage

```
computeVPC.CP(para)
```

### Arguments

para	A $G \times 4$ matrix of CP fit parameters for $G$ features, $G \geq 1$ . The column order is intercept $\alpha_g$ , random effect $\sigma_g^2$ ( $\sigma_g^2 \geq 0$ ), tweedie parameter $p_g$ ( $1 < p_g < 2$ ), and dispersion $\phi$ ( $\phi > 0$ ).
------	---

### Value

A  $G \times 1$  matrix consisting of VPC for  $G$  features based on compound Poisson mixed models. Column name is "CP-fit"; row names are the feature names.

**Examples**

```
## Compute VPC for each feature under compound Poisson mixed models.
vpc.cp <- computeVPC.CP(para_cp)

## Visualize the distribution of the VPCs.
hist(vpc.cp, breaks = 50, col = "cyan")

## Plot sorted VPCs.
plot(sort(vpc.cp), ylab = "Heritability (h2)", ylim = c(0,1), main = "Sorted CP VPC scores")
abline(h = 0.9, lty = 2, col = "red")
text(50, 0.92, "h2 = 0.9", col = "red")
```

---

computeVPC.NB	<i>Calculate the negative binomial (NB) variance partition coefficient (VPC) for one or more features.</i>
---------------	--

---

**Description**

Calculate the NB VPC for one or more features following the model fitting function fit.NB().

**Usage**

```
computeVPC.NB(para)
```

**Arguments**

para	A $G \times 3$ matrix of negative binomial fit parameters for $G$ features, $G \geq 1$ . The column order is intercept $\alpha_g$ , random effect $\sigma_g^2$ ( $\sigma_g^2 \geq 0$ ), and dispersion $\phi$ ( $\phi > 0$ ).
------	---

**Value**

A  $G \times 1$  matrix consisting of VPC for  $G$  features based on negative binomial mixed model. Column name is "NB-fit"; row names are the feature names.

**Examples**

```
## Compute VPC for each feature under negative binomial mixed model.
vpc.nb <- computeVPC.NB(para_nb)

## Visualize the distribution of the VPCs.
hist(vpc.nb, breaks = 50, col = "cyan")

## Plot sorted VPCs.
plot(sort(vpc.nb), ylab = "Heritability (h2)", ylim = c(0,1),
main = "Sorted NB VPC scores")
abline(h = 0.9, lty = 2, col = "red")
text(50, 0.92, "h2 = 0.9", col = "red")
```

---

fit.CP	<i>Fit compound Poisson mixed effect models (CPMM) for one or more features.</i>
--------	--

---

### Description

Fit a CPMM for one or more features and output the fit parameters. It is used before the function `computeVPC.CP()`. This function also allows to test the presence of heritability via random effect variance of the model.

### Usage

```
fit.CP(CountMatrix, Strains, test = FALSE, optimizer = "nlminb")
```

### Arguments

CountMatrix	Sequencing count matrix for one or more features. Each row is for one feature, and the columns are for samples.
Strains	Strain labels for the samples.
test	TRUE or FALSE (default). Test the presence of heritability through examining the random effect variance $\sigma_g^2 = 0$ .
optimizer	A character string that determines which optimization routine is to be used. Possible choices are "nlminb" (default), "L-BFGS-B", and "bobyqa".

### Value

A list with two objects. The first object is a  $G \times 4$  matrix indicating the fitted parameters for each feature. The columns are ordered by intercept  $\alpha_g$ , tweedie parameter  $p_g$ , random effect variance  $\sigma_g^2$ , and dispersion  $\phi_g$ . Row names are feature names. If the argument `test` is set to be true, the second object of the list consists of p-values for testing the hypothesis that random effects  $\sigma_a^2 = 0$ ; otherwise, the second object is NULL.

### Examples

```
## Fit CPMM for the first two features and test the presence of
## heritability.
result.cp <- fit.CP(simData[1:2, ], strains, test = TRUE)
## Extract parameters
para.cp <- result.cp[[1]]
## Extract p-values
pval.cp <- result.cp[[2]]
```

---

fit.NB	<i>Fit negative binomial mixed models (NBMM) for one or more features.</i>
--------	--

---

## Description

Fit NBMM for one or more features and output the fit parameters. It is used before the function `computeVPC.NB()`. This function also allows to test the presence of heritability via random effect variance of the model. To fit a NBMM, the `glmmADMB` package is needed.

## Usage

```
fit.NB(CountMatrix, Strains, test = FALSE)
```

## Arguments

CountMatrix	Sequencing count matrix for a list of features. Each row is for one feature, and the columns are for samples.
Strains	Strain labels for the samples.
test	TRUE or FALSE (default). Test the presence of heritability through examining the random effect variance $\sigma_g^2 = 0$ .

## Value

A list with two objects. The first object is a  $G \times 3$  matrix indicating the fitted parameters for each feature. The columns are ordered by  $\alpha_g, \sigma_g^2, \phi_g$ . Row names are feature names. If the argument `test` is set to be true, the second object of the list consists of p-values for testing the hypothesis that random effects  $\sigma_a^2 = 0$ ; otherwise, the second object is NULL.

## Examples

```
## Compute vpc for each feature under NBMM. This will take a while on the
## entire dataset. For the purpose of illustration, here we only fit on
## the first 2 features.
library(glmmADMB)
result.nb <- fit.NB(simData[1:2, ], strains)
```

---

fitComputeVPC.lmer	<i>Fit linear mixed models (LMM) and compute the VPC values for one or more features.</i>
--------------------	---

---

### Description

Fit the Gaussian-like data to LMM and compute the VPC values for one or more features.

### Usage

```
fitComputeVPC.lmer(CountMatrix, Strains, PriorWeights = NULL, test = FALSE,
  VPCname = "LMM")
```

### Arguments

CountMatrix	Sequencing count matrix for one or more features. Each row is for one feature, and the columns are for samples.
Strains	Strain labels for the samples.
PriorWeights	Weights used in the lmer function in the package lme4. It is an optional vector used in the fitting process.
test	TRUE or FALSE (default). Test the presence of heritability through examining the random effect variance $\sigma_g^2 = 0$ .
VPCname	Name of the VPC result, default = "LMM".

### Value

A list with two objects. The first object is a  $1 \times G$  vector indicating the variance partition coefficients (VPC). If the argument test is set to be true, the second object of the list consists of p-values for testing the hypothesis that random effects  $\sigma_a^2 = 0$ ; otherwise, the second object is NULL.

### Examples

```
## Compute VPC for the first two features under linear mixed models for Gaussian-like datasets.

## Provide normalized data and include hypothesis testing on presence of
## heritability:
result.vst <- fitComputeVPC.lmer(simData_vst[1:2,], strains, test = TRUE)
## Extract parameters
vpc.vst <- result.vst[[1]]
## Extract p-values
pval.vst <- result.vst[[2]]

## Visualize the distribution of p-values.
hist(pval.vst, breaks = 30, col = "cyan")
```

---

getBootCI	<i>Compute variance partition coefficient (VPC) confidence intervals (CI) for one or more features.</i>
-----------	---

---

### Description

Compute VPC CI based on parametric bootstrap for one or more features.

### Usage

```
getBootCI(CountMatrix, Strains, which.features, num.boot, method = "NB-fit",
  alpha = 0.05, optimizer = "nlminb")
```

### Arguments

CountMatrix	A $G \times N$ count matrix. $G$ is the number of features; $N$ is the total number of samples.
Strains	A $1 \times N$ vector of strain labels corresponding to each sample.
which.features	A $1 \times k$ vector of select feature numbers for which CI is desired. $k \leq G$ .
num.boot	Number of bootstraps.
method	Which method should be used, "CP-fit", "NB-fit" (default), or "VST". "VST" method bootstraps data under negative binomial mixed models.
alpha	A numerical value between 0 and 1, indicating the significance level of the CI. The CI will be $100 * (1 - \alpha)$ percent CI. Default value is 0.05.
optimizer	A character string that determines which optimization routine is to be used. It is only used for method = "CP-fit". Possible choices are "nlminb" (default), "L-BFGS-B", and "bobyqa".

### Value

A list of two objects. The first object is a  $k \times 2$  matrix containing the CI. The second object consists of a  $k \times \text{num.boot}$  matrix of all bootstrapped VPC values.

### Examples

```
## Compute CI based on 100 bootstrap samples for the first feature
## under NBMM. It takes a few minutes.
NBboot <- getBootCI(simData, strains, 1, 100)
## Extract CI
NBboot.ci <- NBboot[[1]]
## Extract vpcs
NBboot.vpc <- NBboot[[2]]

## Compute CI based on 100 bootstrap samples for the first feature
## under vst.
```

```
VSTboot <- getBootCI(simData, strains, 1, 100, method = "VST")
```

---

```
getReadMatrix.CP      Simulate a read matrix from compound Poisson mixed effect models
                      (CPMM).
```

---

### Description

Simulate a (possibly unbalanced) read matrix from CPMM. For a compound Poisson (CP) random variable  $Y_{gstr}$  with mean  $\mu_{gs}$ , its variance can be expressed as  $\phi_g \mu_{gs}^{p_g}$ , for some  $1 < p_g < 2$ . Under the CPMM, with a log-link, the regression on the mean has the form:

$$\log(\mu_{gs}) = \alpha_g + b_{gs}, \quad b_{gs} \sim N(0, \sigma_g^2).$$

### Usage

```
getReadMatrix.CP(vec.num.rep, alphas, sigma2s, ps, phis)
```

### Arguments

vec.num.rep	A vector of replicate numbers for each strain.
alphas	Intercept vector $\alpha_g$ 's, $1 \times \text{num. features}$ .
sigma2s	Random effect variance vector $\sigma_g^2$ 's, $1 \times \text{num. features}$ .
ps	Tweedie parameter in CP models, $p_g$ 's, a $1 \times \text{num. features}$ vector.
phis	Dispersion parameter in CP models, $\phi_g$ 's, a $1 \times \text{num. features}$ vector.

### Value

A  $G \times N$  matrix with CP reads.  $N$  is the total number of samples;  $G$  is the number of features. Column names are sample names of the form "Ss\_r", where S stands for sample, s is the strain number, r is the replicate number within the strain. Row names are the feature names of the form "Gene g", where g is the feature index.

### Examples

```
## Generate a sequencing dataset with 5 features and 6 strains.
## Assign parameter values.
rep.num <- c(3, 5, 2, 3, 4, 2)
a0s <- c(-1, 1, 2, 5, 10)
sig2s <- c(10, 0.2, 0.1, 0.03, 0.01)
ps <- rep(1.5, 5)
phis <- c(1.5, 1, 0.5, 0.1, 0.1)

set.seed(1234)
## Generate reads:
cpData <- getReadMatrix.CP(rep.num, a0s, sig2s, ps, phis)
## Generate strain names:
```



```

str <- sapply(1:length(rep.num), function(x){
  str.x <- paste0("S", x)
  return(rep(str.x, rep.num[x]))
})
str <- do.call(c, str)

```

---

getReadMatrix.NB      *Simulate a count matrix from negative binomial mixed effect models (NBMM).*

---

### Description

Simulate a (possibly unbalanced) count matrix from NBMM. Under NBMM, an observed number of reads aligned to feature/gene  $g$ ,  $Y_{gSR}$ , follows a negative binomial (NB) distribution with mean  $\mu_{gs}$  and variance  $\mu_{gs} + \phi_g \mu_{gs}^2$ , where  $\phi_g$  is the dispersion parameter, shared across strains. The generalized linear model uses a log-link:

$$\log(\mu_{gs}) = \alpha_g + b_{gs}, \quad b_{gs} \sim N(0, \sigma_g^2).$$

### Usage

```
getReadMatrix.NB(vec.num.rep, alphas, sigma2s, phis)
```

### Arguments

vec.num.rep      A vector of replicate numbers for each strain.  
 alphas            Intercept vector  $\alpha_g$ 's,  $1 \times \text{num. features}$ .  
 sigma2s          Random effect variance vector  $\sigma_g^2$ 's,  $1 \times \text{num. features}$ .  
 phis              Dispersion parameter in NB models,  $\phi_g$ 's, a  $1 \times \text{num. features}$  vector.

### Value

A  $G \times N$  matrix with NB reads.  $N$  is the total number of samples;  $G$  is the number of features. Column names are sample names of the form "Ss\_r", where S stands for sample, s is the strain number, r is the replicate number within the strain. Row names are the feature names of the form "Gene g", where g is the feature index.

### Examples

```

## Generate a sequencing dataset with 5 features and 6 strains.
## Assign parameter values.
rep.num <- c(3, 5, 2, 3, 4, 2)
a0s <- c(-1, 1, 2, 5, 10)
sig2s <- c(10, 0.2, 0.1, 0.03, 0.01)
phis <- c(0.5, 1, 0.05, 0.01, 0.1)

set.seed(1234)
## Generate reads:
nbData <- getReadMatrix.NB(rep.num, a0s, sig2s, phis)

```

---

para_cp	<i>Parameter matrix obtained from simData by fitting CPMM.</i>
---------	--

---

**Description**

Parameter matrix obtained from simData by fitting CPMM.

**Usage**

para\_cp

**Format**

An object of class `matrix` with 100 rows and 4 columns.

---

para_nb	<i>Parameter matrix obtained from simData by fitting NBMM.</i>
---------	--

---

**Description**

Parameter matrix obtained from simData by fitting NBMM.

**Usage**

para\_nb

**Format**

An object of class `matrix` with 100 rows and 3 columns.

---

simData	<i>A simulated sequencing dataset.</i>
---------	--

---

**Description**

A matrix containing simulated counts for 100 features (rows) and 175 samples (columns)

**Usage**

simData

**Format**

A matrix with 100 rows and 175 columns

---

simData_voom	<i>Voom transformed version of simData.</i>
--------------	---

---

**Description**

Voom transformed version of simData.

**Usage**

```
simData_voom
```

**Format**

An object of class `matrix` with 881 rows and 175 columns.

---

simData_vst	<i>Variance stabilize transformed version of simData.</i>
-------------	---

---

**Description**

Variance stabilize transformed version of simData.

**Usage**

```
simData_vst
```

**Format**

An object of class `matrix` with 881 rows and 175 columns.

---

strains	<i>List of strain names for the samples.</i>
---------	--

---

**Description**

List of strain names for the samples.

**Usage**

```
strains
```

**Format**

An object of class `character` of length 175.

---

`weights_voom`*Weights used in the voom transformation.*

---

**Description**

Weights used in the voom transformation.

**Usage**

```
weights_voom
```

**Format**

An object of class `matrix` with 881 rows and 175 columns.

# Index

## \*Topic **datasets**

- para\_cp, [10](#)
- para\_nb, [10](#)
- simData, [10](#)
- simData\_voom, [11](#)
- simData\_vst, [11](#)
- strains, [11](#)
- weights\_voom, [12](#)

  

- computeVPC.CP, [2](#)
- computeVPC.NB, [3](#)

  

- fit.CP, [4](#)
- fit.NB, [5](#)
- fitComputeVPC.lmer, [6](#)

  

- getBootCI, [7](#)
- getReadMatrix.CP, [8](#)
- getReadMatrix.NB, [9](#)

  

- para\_cp, [10](#)
- para\_nb, [10](#)

  

- simData, [10](#)
- simData\_voom, [11](#)
- simData\_vst, [11](#)
- strains, [11](#)

  

- weights\_voom, [12](#)