Overview

The **HeritSeq** package provides heritability score analyses under linear mixed models (LMM) or generalized linear mixed models (GLMM) for count data motivated by high-throughput sequencing. It is applicable to counts with biological replicates. This package includes functions to:

- compute heritability score under LMM for a normalized/transformed dataset, and under negative binomial mixed models (NBMM) or compound Poisson mixed models (CPMM) for data without transformations.

- test presence of heritability under LMM, NBMM or CPMM.
- generate confidence intervals of estimated heritability score.
- simulate synthetic high-throughput sequencing datasets using NBMM or CPMM.

See [1] for model details and performance comparisons.

[1]: Rudra, P., Shi, W. J., Vestal, B., Russell, P. H., Odell, A., Dowell, R., Radcliffe, R., Saba, L. M., & Kechris, K. *Model based heritability scores for high-throughput sequencing data.* (submitted).

Example dataset

The **HeritSeq** package includes an example high-throughput sequencing dataset called *simData*. This dataset was generated based on a recombinant inbred mice panel miRNA sequencing counts. It is an 881 by 175 matrix, containing 881 features and 175 samples. The total number of strains is 59, and the strain labels are recorded by the variable *strains*.

Installation

The package requires **R** version >=3.2.3.

The installation of dependencies only needs to be done once, if at all. Instructions for installing the dependencies:

```
install.packages("lme4", repos="http://cran.r-project.org")
install.packages("cplm", repos="http://cran.r-project.org")
install.packages("pbapply", repos="http://cran.r-project.org")
install.packages("statmod", repos="http://cran.r-project.org")
install.packages("MASS", repos="http://cran.r-project.org")
install.packages("stats", repos="http://cran.r-project.org")
install.packages("utils", repos="http://cran.r-project.org")

install.packages("glmmADMB",
                 repos=c("http://glmmadmb.r-forge.r-project.org/repos",
                         getOption("repos")),
                 type="source")

source("https://bioconductor.org/biocLite.R")
biocLite("DESeq2")
biocLite("SummarizedExperiment")
```

Since R for OS X Maverick (and latter versions) was compiled using gfortran-4.8, Mac users might receive a "-lgfortran" error when installing the cplm package. This problem and its solution are stated here.

Next, download the package tarball and install the package:

```
install.packages("HeritSeq_1.0.0.tar.gz",repos=NULL,type="source")
library("HeritSeq")
```

Estimate heritability scores

We will use *simData* to illustrate the procedure of heritability estimation under different models.

Before fitting any model, make sure that the input dataset has been adjusted for library sizes and batch bias. The dataset *simData* is post such adjustment, we therefore omit the process.

NBMM

Under NBMM, an observed number of reads aligned to feature/gene g, Y_{gsr}, follows a negative binomial distribution with mean μ_{gs} and variance $\mu_{gs} + \phi_g \mu_{gs}^2$, where ϕ_g is the dispersion parameter for feature/gene g, shared across strains. The generalized linear model uses a log-link: $\log(\mu_{gs}) = \alpha_g + b_{gs}$, $b_{gs} \sim N(0, \sigma_g^2)$.

The corresponding heritability score, aka Variance Partition Coefficient (VPC), is $\frac{e^{\sigma_g^2}-1}{e^{\sigma_g^2}-1+\phi_g e^{\sigma_g^2}+e^{-\alpha_g-\sigma_g^2/2}}$.

Compute VPC for all features using NBMM:

```
result.nb <- fit.NB(CountMatrix = simData, Strains = strains, test = FALSE)
vpc.nb <- computeVPC.NB(para = result.nb[[1]])
```

The function *fit.NB( )* returns a list with two objects. The first object is a $G \times 3$ matrix indicating the fitted parameters for each feature, where G is the total number of features/genes. The columns are ordered by the fitted parameters $\alpha_g, \sigma_g^2, \phi_g$. The second object provides the p-value for testing the presence of heritability if test = TRUE; it returns NULL otherwise (default).

The function *computeVPC.NB( )* takes in the list of NBMM parameters and outputs the corresponding VPC values.

CPMM

For a CP random variable Y_{gsr} with mean μ_{gs}, its variance can be expressed as $\phi_g \mu_{gs}^{p_g}$, for some tweedie parameter $1 < p_g < 2$ and dispersion parameter ϕ_g for feature/gene g [1]. Under the CPMM, with a log-link, the regression on the mean has the same form as the NBMM: $\log(\mu_{gs}) = \alpha_g + b_{gs}$, $b_{gs} \sim N(0, \sigma_g^2)$.

The corresponding VPC is $\frac{e^{\sigma_g^2}-1}{e^{\sigma_g^2}-1+\phi_g e^{(p_g-2)\alpha_g+(p_g^2/2-1)\sigma_g^2}}$.

Compute VPC for all features using CPMM:

```
result.cp <- fit.CP(CountMatrix = simData, Strains = strains, test = FALSE,
                    optimizer = "nlminb")
vpc.cp <- computeVPC.CP(para = result.cp[[1]])
```

[1] Tweedie, M. C. K. (1981). *An index which distinguishes between some important exponential families.* Statistics: applications and new directions, 579–604.

Similar to *fit.NB( )*, the function *fit.CP( )* returns a list of two objects. The first object consists the fitted parameters $\alpha_g, \sigma_g^2, p_g, \phi_g$. The second object provides the p-value for testing the presence of heritability if test = TRUE; it returns NULL otherwise (default). The argument optimizer determines the optimization routine. Other possible choices are "L-BGFS-B" and "bobyqa".

The function *computeVPC.CP( )* takes in the list of CPMM parameters and outputs the corresponding VPC values.

VST

In order to use a linear mixed model (LMM), the sequence reads first need to be transformed to Gaussian-like data. The variance stablizing transformation (VST) was introduced in the package **DESeq2** [2]. It is based on a negative binomial assumption of the original data.

Transform *simData* using VST and fit features using LMM:

```
cds <- DESeqDataSetFromMatrix(countData = round(simData),
                              colData = data.frame(strain = strains),
                              design = formula(~strain))
cds <- estimateSizeFactors(object = cds)
cds <- estimateDispersions(object = cds, fitType = "local")
vsd <- varianceStabilizingTransformation(cds, fitType = "local")
simData.vst <- assay(x = vsd)

vpc.vst <- fitComputeVPC.lmer(CountMatrix = simData.vst,
                              Strains = strains,
                              PriorWeight = NULL,
                              test = FALSE,
                              VPCname = "VST")[[1]]
```

The function *fitComputeVPC.lmer( )* fits a linear mixed model for each feature and directly computes the VPC scores. The output includes a list of VPC values and significance result for testing presence of heritability. If test = TRUE, the second object of the list are the p-values; otherwise, it is NULL (default). The argument PriorWeight is an optional input used in the lmer function in the package **lme4** [3]. Note that variance stablizating transformation does not output a prior weight, hence the argument PriorWeight is set to be NULL for VST.

voom

limma voom is another popular method to transform counts to Gaussian-like data [4]. It is a less preferrable method for computing VPC comparing to the ones above (see reference [1]).

Transform *simData* using voom and fit features using LMM as above:

```
library(limma)
voomed.librarySize <- voom(counts = simData,
                           normlize.method = "scale",
                           lib.size = sizeFactors(cds))
```

[2] Love, M. I., Huber, W., & Anders, S. (2014). *Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2.* Genome biology, 15(12), 1.

[3] Bates, D., Mächler, M., Bolker, B., & Walker, S. (2014). *Fitting linear mixed-effects models using lme4.* arXiv preprint arXiv:1406.5823.

[4] Ritchie, M. E., Phipson, B., Wu, D., Hu, Y., Law, C. W., Shi, W., & Smyth, G. K. (2015). *limma powers differential expression analyses for RNA-sequencing and microarray studies.* Nucleic acids research, gkv007.

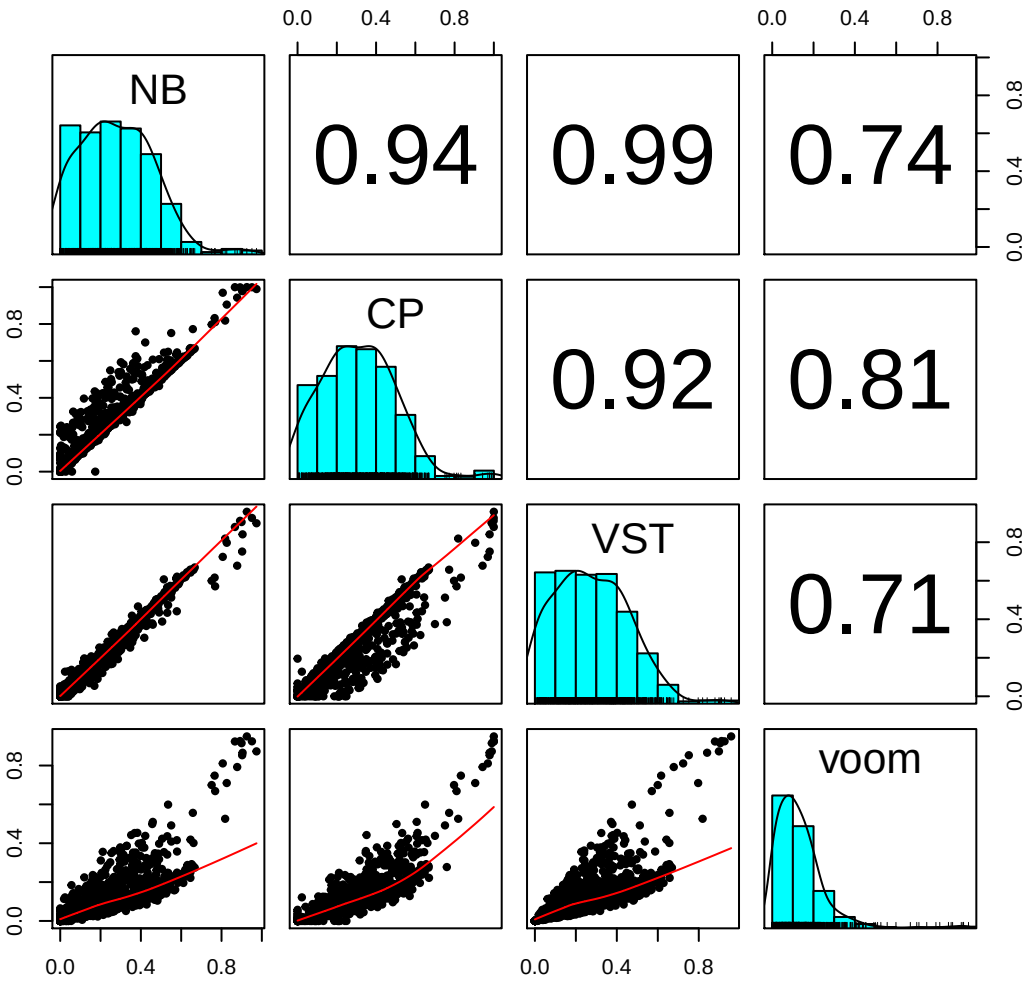
```
simData.voom <- voomed.librarySize$E
weights.voom <- voomed.librarySize$weights

vpc.voom <- fitComputeVPC.lmer(CountMatrix = simData.voom,
                               Strains = strains,
                               PriorWeights = weights.voom,
                               test = FALSE,
                               VPCname = "voom")[[1]]
```

Note that the voom method does provide a prior weight for fitting the regression.

Compare results from various methods

In most cases, it is not clear which method is more appropriate for the given dataset. We suggest to fit multiple models and compare the estimation results.

```
library(psych)
h2 <- cbind(vpc.nb, vpc.cp, vpc.vst, vpc.voom)
colnames(h2) <- c("NB", "CP", "VST", "voom")
pairs.panels(x = h2, ellipses = FALSE, main = "Compare VPC")
```



The function *pairs.panels( )* creates multiple panels for pairwise comparison of the methods. Along the diagonal are the histograms and kernel density plot of estimated VPC values. The panels below show pairwise comparison of the methods with locally weighted scatterplot smoothing (loess regression) in red. Above the diagonal are the correlation coefficient values.

The first three methods: NB, CP, and voom present relatively consistent results. The features with top VPC scores are similar.

Hypothesis testing

As illustrated in the examples, each of the four methods above also allows hypothesis testing for presence of heritability, simply by setting Test = TRUE in the functions *fit.NB( )*, *fit.CP( )*, and *fitComputeVPC.lmer( )* for NBMM, CPMM, and LMM respectively.

Confidence intervals

The VPC confidence intervals (CIs) can be generated using a parametric bootstrap. When choosing the VST method, the bootstraped data are generated from negative binomal models followed by the variance stablized transformation. Due to its less superior performance (detailed discussion can be found in Reference [1]), the voom approach is excluded for the *getBootCI( )* function.

Compute CI based on 100 boostrap samples for the first 3 features:

```
boot.vst <- getBootCI(CountMatrix = simData,
                      Strains = strains,
                      which.features = 1:3,
                      num.boot = 100,
                      method = "VST",
                      alpha=0.05,
                      optimizer = "nlminb")
```

The *getBootCI( )* function computes CIs based on parametric bootstrap for one or more features. It allows three models: NB, CP, and VST. Although the NB method is defined as the default, it is the most time-consuming. When choosing the VST method, the input data *CountMatrix* should still be the un-transformed data. The argument *which.features* specifies the feature indices for which the CIs should be computed for; *alpha* defines the significance level; *optimizer* is only used for the CP method.

Simulate synthetic datasets

Synthetic sequencing data with biological replicates can be simulated using functions *getReadMatrix.NB( )* or *getReadMatrix.CP( )*.

To siumulate counts under NBMM, the parameters are:

- number of biological replicates for each strain (rep.num)
- model intercepts α_g's (a0s)
- strain variances σ_g^2's (sig2s)
- dispersions ϕ_g's (phis1)

```
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)
phis1 <- c(0.5, 1, 0.05, 0.01, 0.1)

set.seed(1234)
nbData <- getReadMatrix.NB(vec.num.rep = rep.num,
                           alphas = a0s,
                           sigma2s = sig2s,
                           phis = phis1)
```

To simulate synthetic data under CPMM, the parameters are:

- number of biological replicates for each strain (rep.num)
- model intercepts α_g's (a0s)
- strain variances σ_g^2's (sig2s)
- Tweedie parameter p_g's (ps)
- dispersions ϕ_g's (phis2)

```
ps <- rep(1.5, 5)
phis2 <- c(1.5, 1, 0.5, 0.1, 0.1)

set.seed(1234)
```

```
cpData <- getReadMatrix.CP(vec.num.rep = rep.num,
                           alphas = a0s,
                           sigma2s = sig2s,
                           ps = ps,
                           phis = phis2)
```

Versioning

Date: 2016-09-06

Version: 1.0.0

Acknowledgements

We would like to thank Yanwei (Wayne) Zhang, the cplm package author, for helpful discussions regarding the compound Poisson regression fit in cplm and for providing a modified function before the new version of his package is released.

Authors

- W. Jenny Shi wjennyshi@gmail.com
- Pamela Russell pamela.russell@ucdenver.edu
- Pratyaydipta Rudra pratyaydipta.rudra@ucdenver.edu
- Brian Vestal brian.vestal@ucdenver.edu
- Katerina Kechris katerina.kechris@ucdenver.edu
- Laura Saba laura.saba@undenver.edu

License

GPL-2