Package: RNAseqCovarImpute (via r-universe)

June 30, 2024

Title Impute Covariate Data in RNA Sequencing Studies

Version 1.3.0

URL https://github.com/brennanhilton/RNAseqCovarImpute

BugReports https://github.com/brennanhilton/RNAseqCovarImpute/issues

Description The RNAseqCovarImpute package makes linear model analysis for RNA sequencing read counts compatible with multiple imputation (MI) of missing covariates. A major problem with implementing MI in RNA sequencing studies is that the outcome data must be included in the imputation prediction models to avoid bias. This is difficult in omics studies with high-dimensional data. The first method we developed in the RNAseqCovarImpute package surmounts the problem of high-dimensional outcome data by binning genes into smaller groups to analyze pseudo-independently. This method implements covariate MI in gene expression studies by 1) randomly binning genes into smaller groups, 2) creating M imputed datasets separately within each bin, where the imputation predictor matrix includes all covariates and the log counts per million (CPM) for the genes within each bin, 3) estimating gene expression changes using `limma::voom` followed by `limma::lmFit` functions, separately on each M imputed dataset within each gene bin, 4) un-binning the gene sets and stacking the M sets of model results before applying the `limma::squeezeVar` function to apply a variance shrinking Bayesian procedure to each M set of model results, 5) pooling the results with Rubins' rules to produce combined coefficients, standard errors, and P-values, and 6) adjusting P-values for multiplicity to account for false discovery rate (FDR). A faster method uses principal component analysis (PCA) to avoid binning genes while still retaining outcome information in the MI models. Binning genes into smaller groups requires that the MI and limma-voom analysis is run many times (typically hundreds). The more computationally efficient MI PCA method implements covariate MI in gene expression studies by 1) performing PCA on the log CPM values for all genes using the Bioconductor `PCAtools` package, 2) creating M imputed datasets where the imputation predictor matrix includes all covariates and the optimum number of PCs to retain (e.g., based on Horn's parallel analysis or the number of PCs that account for >80% explained variation), 3) conducting the standard limma-voom pipeline with the `voom` followed by `lmFit` followed by `eBayes` functions on each M imputed dataset, 4) pooling the results with Rubins' rules to produce combined coefficients, standard errors, and P-values, and 5) adjusting P-values for multiplicity to account for false discovery rate (FDR).

License GPL-3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

VignetteBuilder knitr

biocViews RNASeq, GeneExpression, DifferentialExpression, Sequencing

Suggests BiocStyle, knitr, PCAtools, rmarkdown, tidyr, stringr, testthat (>= 3.0.0)

Depends R (>= 4.3.0)

Imports Biobase, BiocGenerics, BiocParallel, stats, limma, dplyr, magrittr, rlang, edgeR, foreach, mice

Config/testthat/edition 3

Collate 'RNAseqCovarImpute-package.R' 'combine_rubins.R' 'combine_rubins_pca.R' 'example_DGE.R' 'example_data.R' 'get_gene_bin_intervals.R' 'impute_by_gene_bin_helper.R' 'impute_by_gene_bin.R' 'voom_sx_sy.R' 'lowess_all_gene_bins.R' 'voom_master_lowess.R' 'limmavoom_imputed_data_list_helper.R' 'limmavoom_imputed_data_list.R' 'limmavoom_imputed_data_pca_helper.R' 'limmavoom_imputed_data_pca.R'

Repository https://bioc.r-universe.dev

RemoteUrl https://github.com/bioc/RNAseqCovarImpute

RemoteRef HEAD

RemoteSha 930fa0b1590173ef27ebb2b5ca8ce430c14d1b7d

Contents

`combine_rubins` *combine_rubins*

Description

Combines results from each imputed dataset using Rubin's rules.

Usage

```
combine_rubins(
  DGE,
  model_results,
  predictor,
  covariate = NULL,
  robust = FALSE,
  winsor.tail.p = c(0.05, 0.1)
)
```

Arguments

`DGE`	A DGEList object.
`model_results`	Output from limmavoom_imputed_datalist.
`predictor`	Independent variable of interest, in the form of a linear model contrast. Must be a variable in voom_formula.
`covariate`	Arguments passed to limma::squeezeVar. If non-NULL, var.prior will depend on this numeric covariate. Otherwise, var.prior is constant.
`robust`	Arguments passed to limma::squeezeVar. Llogical, should the estimation of df.prior and var.prior be robustified against outlier sample variances?
`winsor.tail.p`	Arguments passed to limma::squeezeVar. Numeric vector of length 1 or 2, giving left and right tail proportions of x to Winsorize. Used only when robust=TRUE.

Value

Dataframe with one row per gene containing coefficients standard errors, degrees of freedom, t-statistics, P-Values, and adjusted P-values from the limma-voom pipeline.

`coef_combined`	combined logFCs across the multiple imputed datasets using Rubin's rules
`SE_P`	pooled standard error across the multiple imputed datasets using Rubin's rules

`SE_P_bayes`	pooled standard error across the multiple imputed datasets using Rubin's rules squeezed to global mean variance trend curve with limma-voom Bayesian procedure
`df`	limma-voom residual degrees of freedom adjusted for Rubin's rules
`df_bayes`	limma-voom residual degrees of freedom adjusted for Rubin's rules and Bayesian procedure
`rubins_t`	t-statistic = coef_combined divided by SE_p
`rubins_t_bayes`	t-statistic = coef_combined divided by SE_p_bayes
`combined_p`	p-value from two-sided t-distribution alpha = 0.05 using rubins_t
`combined_p_bayes`	p-value from two-sided t-distribution alpha = 0.05 using rubins_t_bayes
`combined_p_adj`	false discovery rate (FDR) adjusted combined_p
`combined_p_adj_bayes`	false discovery rate (FDR) adjusted combined_p_bayes

Examples

```
data(example_data)
data(example_DGE)
intervals <- get_gene_bin_intervals(example_DGE, example_data, n = 10)
gene_bin_impute <- impute_by_gene_bin(example_data,
    intervals,
    example_DGE,
    m = 2
)
coef_se <- limmavoom_imputed_data_list(
    gene_intervals = intervals,
    DGE = example_DGE,
    imputed_data_list = gene_bin_impute,
    m = 2,
    voom_formula = "~x + y + z + a + b"
)

final_res <- combine_rubins(
    DGE = example_DGE,
    model_results = coef_se,
    predictor = "x"
)
```

`example_data` *Simulated dataset*

Description

The exact code used to generate these data are found in the Example_Data_for_RNAseqCovarImpute vignette. In short, `example_data` contains 500 rows with data for variables x, y, and z, which are continuous normally distributed, and a and b, which are binary variables. Missingness was simulated for all variables other than x such that a complete case analysis would drop 24.2% of participants. `example_DGE` contains random count data from the Poisson distribution for 500 made up genes, ENS1-ENS500

Usage

```
data(example_data)
```

Format

`example_data`**:**

data frame with 500 rows and 5 variables

x continuous normally distributed

y continuous normally distributed

z continuous normally distributed

a binary

b binary ...

Value

Tibble with 500 rows of data for variables x, y, and z

Examples

```
data(example_data)
```

`example_DGE` *Simulated counts in DGE list*

Description

The exact code used to generate these data are found in the Example_Data_for_RNAseqCovarImpute vignette. In short, `example_data` contains 500 rows with data for variables x, y, and z, which are continuous normally distributed, and a and b, which are binary variables. Missingness was simulated for all variables other than x such that a complete case analysis would drop 24.2% of participants. `example_DGE` contains random count data from the Poisson distribution for 500 made up genes, ENS1-ENS500

Usage

```
data(example_DGE)
```

Format

`example_DGE`:

A DGElist with 500 genes and 500 samples

Value

DGElist for 500 made up genes, ENS1-ENS500

Examples

```
data(example_DGE)
```

`get_gene_bin_intervals` *get_gene_bin_intervals*

Description

Creates gene bins. Input DGE list, sample data, and 'n' number of individuals per genes. By default, number of bins and genes per bin are set so that each bin has approximately 1 gene per 10 individuals in the data.

Usage

```
get_gene_bin_intervals(DGE, data, n = 10)
```

Arguments

`DGE`	A DGEList object.
`data`	Sample data with one row per sample. Sample row order should match the col order in the DGEList.
`n`	Genes per bin are set so that each bin has approximately 1 gene per n individuals in the data.

Value

Data frame with one row per gene bin. Columns indicate the start and end positions and the number of genes of each bin.

Examples

```
data(example_data)
data(example_DGE)
intervals <- get_gene_bin_intervals(example_DGE, example_data, n = 10)
gene_bin_impute <- impute_by_gene_bin(example_data,
    intervals,
    example_DGE,
    m = 2
)
coef_se <- limmavoom_imputed_data_list(
    gene_intervals = intervals,
    DGE = example_DGE,
    imputed_data_list = gene_bin_impute,
    m = 2,
    voom_formula = "~x + y + z + a + b"
)

final_res <- combine_rubins(
    DGE = example_DGE,
    model_results = coef_se,
    predictor = "x"
)
```

`impute_by_gene_bin` *impute_by_gene_bin*

Description

Loops through DGE list using the gene bin intervals from the "get_gene_bin_intervals" function and makes imputed datasets. For instance, if n = 100 and intervals contains 200 gene bin intervals, output will be a list of 200 sets of 100 imputed datasets. Each of the 200 sets are imputed using only the genes in one gene bin.

Usage

```
impute_by_gene_bin(data, intervals, DGE, m, maxit = 10, BPPARAM = bpparam())
```

Arguments

`data`	Sample data with one row per sample. Sample row order should match the col order in the DGEList.
`intervals`	Output from get_gene_bin_intervals function. A dataframe where each row contains the start (first col) and end (second col) values for each gene bin interval.
`DGE`	A DGEList object.
`m`	Number of imputed data sets.
`maxit`	Used by mice function.
`BPPARAM`	A BiocParallelParam object

Value

A list of sets of n imputed datasets, one per gene bin.

Examples

```
data(example_data)
data(example_DGE)
intervals <- get_gene_bin_intervals(example_DGE, example_data, n = 10)
gene_bin_impute <- impute_by_gene_bin(example_data,
    intervals,
    example_DGE,
    m = 2
)
coef_se <- limmavoom_imputed_data_list(
    gene_intervals = intervals,
    DGE = example_DGE,
    imputed_data_list = gene_bin_impute,
    m = 2,
    voom_formula = "~x + y + z + a + b"
)

final_res <- combine_rubins(
    DGE = example_DGE,
    model_results = coef_se,
    predictor = "x"
)
```

`limmavoom_imputed_data_list`
limmavoom_imputed_data_list

Description

Loops through the imputed data list (output from "impute_by_gene_bin" function) and runs limma-voom RNA seq analysis.

Usage

```
limmavoom_imputed_data_list(
  gene_intervals,
  DGE,
  imputed_data_list,
  m,
  voom_formula,
  BPPARAM = bpparam()
)
```

Arguments

`gene_intervals`	Output from get_gene_bin_intervals function. A dataframe where each row contains the start (first col) and end (second col) values for each gene bin interval.
`DGE`	A DGEList object.
`imputed_data_list`	Output from impute_by_gene_bin.
`m`	Number of imputed data sets.
`voom_formula`	Formula for design matrix.
`BPPARAM`	A BiocParallelParam object

Value

A dataframe with coefficient, standard error, sigma, and residual degrees of freedom values from limma-voom gene expression analysis. One row per gene and one set of values per imputed dataset.

Examples

```
data(example_data)
data(example_DGE)
intervals <- get_gene_bin_intervals(example_DGE, example_data, n = 10)
gene_bin_impute <- impute_by_gene_bin(example_data,
    intervals,
    example_DGE,
    m = 2
)
coef_se <- limmavoom_imputed_data_list(
    gene_intervals = intervals,
    DGE = example_DGE,
    imputed_data_list = gene_bin_impute,
    m = 2,
    voom_formula = "~x + y + z + a + b"
)

final_res <- combine_rubins(
    DGE = example_DGE,
    model_results = coef_se,
    predictor = "x"
)
```

`limmavoom_imputed_data_pca` *limmavoom_imputed_data_pca*

Description

Combines results from each imputed dataset using Rubin's rules.

Usage

```
limmavoom_imputed_data_pca(imp, DGE, voom_formula, BPPARAM = bpparam())
```

Arguments

`imp`	Imputed data from mice (mids object)
`DGE`	A DGEList object.
`voom_formula`	Formula for design matrix.
`BPPARAM`	A BiocParallelParam object

Value

Dataframe with one row per gene. Columns contain coefficients, standard errors, and p-values from the limma-voom pipeline.

Examples

```
data(example_data)
data(example_DGE)
pca_data = limma::voom(example_DGE)$E
p = PCAtools::pca(pca_data)
pcs = p$rotated[,1:78]
example_data = cbind(example_data, pcs)
imp = mice::mice(example_data, m=3)
mi_pca_res = limmavoom_imputed_data_pca(
   imp = imp,
   DGE = example_DGE,
   voom_formula = "~x + y + z + a + b"
   )
```

Index