

# Package ‘fishpond’

October 23, 2020

**Title** Fishpond: differential transcript and gene expression with inferential replicates

**Version** 1.5.39

**Maintainer** Michael Love <michaelisaiahlove@gmail.com>

**Description** Fishpond contains methods for differential transcript and gene expression analysis of RNA-seq data using inferential replicates for uncertainty of abundance quantification, as generated by Gibbs sampling or bootstrap sampling. Also the package contains utilities for working with Salmon and Alevin quantification files.

**Imports** graphics, stats, utils, methods, abind, gtools, qvalue, S4Vectors, SummarizedExperiment, matrixStats, svMisc, Rcpp, Matrix

**Suggests** testthat, knitr, rmarkdown, macrophage, tximeta, org.Hs.eg.db, samr, DESeq2, apeglm, tximportData, SingleCellExperiment

**LinkingTo** Rcpp

**SystemRequirements** C++11

**License** GPL-2

**Encoding** UTF-8

**URL** <https://github.com/mikelove/fishpond>

**biocViews** Sequencing, RNASeq, GeneExpression, Transcription, Normalization, Regression, MultipleComparison, BatchEffect, Visualization, DifferentialExpression, DifferentialSplicing, AlternativeSplicing, SingleCell

**VignetteBuilder** knitr

**LazyData** true

**RoxygenNote** 7.1.0

**git\_url** <https://git.bioconductor.org/packages/fishpond>

**git\_branch** master

**git\_last\_commit** 6180987

**git\_last\_commit\_date** 2020-10-13

**Date/Publication** 2020-10-22

**Author** Anzi Zhu [aut, ctb],  
 Michael Love [aut, cre],  
 Avi Srivastava [aut, ctb],  
 Rob Patro [aut, ctb],  
 Joseph Ibrahim [aut, ctb],  
 Hirak Sarkar [ctb],  
 Scott Van Buren [ctb]

## R topics documented:

fishpond-package	2
addStatsFromCSV	3
computeInfRV	4
deswish	4
isoformProportions	5
labelKeep	6
makeInfReps	6
makeSimSwishData	7
miniSwish	8
plotInfReps	9
plotMASwish	10
readEDS	11
scaleInfReps	12
splitSwish	13
swish	14
<b>Index</b>	<b>17</b>

---

fishpond-package

*Downstream methods for Salmon and Alevin expression data*

---

## Description

This package provides statistical methods and other tools for working with Salmon and Alevin quantification of RNA-seq data. In particular, it contains the Swish non-parametric method for detecting differential transcript expression (DTE). Swish can also be used to detect differential gene expression (DGE).

## Details

The main functions are:

- [scaleInfReps](#) - scaling transcript or gene expression data
- [labelKeep](#) - labelling which features have sufficient counts
- [swish](#) - perform non-parametric differential analysis
- Plots, e.g., [plotMASwish](#), [plotInfReps](#)
- [isoformProportions](#) - convert counts to isoform proportions
- [makeInfReps](#) - create pseudo-inferential replicates
- [splitSwish](#) - split Swish analysis across jobs with Snakemake

All software-related questions should be posted to the Bioconductor Support Site:

<https://support.bioconductor.org>

The code can be viewed at the GitHub repository, which also lists the contributor code of conduct:

<https://github.com/mikelove/fishpond>

### Author(s)

Anqi Zhu, Avi Srivastava, Joseph G. Ibrahim, Rob Patro, Michael I. Love

### References

Swish method:

Zhu, A., Srivastava, A., Ibrahim, J.G., Patro, R., Love, M.I. (2019) Nonparametric expression analysis using inferential replicate counts. *Nucleic Acids Research*. <https://doi.org/10.1093/nar/gkz622>

Compression, makeInfReps and splitSwish:

Van Buren, S., Sarkar, H., Srivastava, A., Rashid, N.U., Patro, R., Love, M.I. (2020) Compression of quantification uncertainty for scRNA-seq counts. *bioRxiv*. <https://doi.org/10.1101/2020.07.06.189639>

---

addStatsFromCSV

*Read statistics and nulls from CSV file*

---

### Description

After running `splitSwish` and the associated Snakefile, this function can be used to gather and add the results to the original object. See the `alevin` section of the vignette for an example.

### Usage

```
addStatsFromCSV(y = NULL, infile, estPi0 = FALSE)
```

### Arguments

<code>y</code>	a <code>SummarizedExperiment</code> (if <code>NULL</code> , function will output a <code>data.frame</code> )
<code>infile</code>	character, path to the <code>summary.csv</code> file
<code>estPi0</code>	logical, see <code>swish</code>

### Value

the `SummarizedExperiment` with metadata columns added, or if `y` is `NULL`, a `data.frame` of compiled results

---

computeInfRV	<i>Compute inferential relative variance (InfRV)</i>
--------------	--

---

### Description

InfRV is used the Swish publication for visualization. This function provides computation of the mean InfRV, a simple statistic that measures inferential uncertainty. Note that InfRV is not used in the swish statistical method at all, it is just for visualization. See function code for details.

### Usage

```
computeInfRV(y, pc = 5, shift = 0.01, meanVariance)
```

### Arguments

y	a SummarizedExperiment
pc	a pseudocount parameter for the denominator
shift	a final shift parameter
meanVariance	logical, use pre-computed inferential mean and variance assays instead of counts and computed variance from infReps. If missing, will use pre-computed mean and variance when present.

### Value

a SummarizedExperiment with meanInfRV in the metadata columns

---

deswish	<i>deswish: DESeq2-apeglm With Inferential Samples Helps</i>
---------	--

---

### Description

The DESeq2-apeglm With Inferential Samples implementation supposes a hierarchical distribution of log2 fold changes. The final posterior standard deviation is calculated by adding the posterior variance from modeling biological replicates computed by apeglm, and the observed variance on the posterior mode over inferential replicates. This function requires the DESeq2 and apeglm packages to be installed and will print an error if they are not found.

### Usage

```
deswish(y, x, coef)
```

### Arguments

y	a SummarizedExperiment containing the inferential replicate matrices, as output by tximeta, and then with labelKeep applied. One does not need to run scaleInfReps as scaling is done internally via DESeq2.
x	the design matrix
coef	the coefficient to test (see lfcShrink)

**Value**

a SummarizedExperiment with metadata columns added: the log2 fold change and posterior SD using inferential replicates, and the original log2 fold change (apeglm) and its posterior SD

**References**

The DESeq and lfcShrink function in the DESeq2 package:

Zhu, Ibrahim, Love "Heavy-tailed prior distributions for sequence count data: removing the noise and preserving large differences" Bioinformatics (2018).

Love, Huber, Anders "Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2" Genome Biology (2014).

**Examples**

```
# a small example... 500 genes, 10 inf reps
y <- makeSimSwishData(m=500, numReps=10)
y <- labelKeep(y)
#y <- deswish(y, ~condition, "condition_2_vs_1")
```

---

isoformProportions      *Create isoform proportions from scaled data*

---

**Description**

Takes output of scaled (and optionally filtered) counts and returns isoform proportions by dividing out the total scaled count for the gene for each sample. The operation is performed on the counts assay, then creating a new assay called isoProp, and on all of the inferential replicates, turning them from counts into isoform proportions. Any transcripts (rows) from single isoform genes are removed, and the transcripts will be re-ordered by gene ID.

**Usage**

```
isoformProportions(y, geneCol = "gene_id", quiet = FALSE)
```

**Arguments**

y	a SummarizedExperiment
geneCol	the name of the gene ID column in the metadata columns for the rows of y
quiet	display no messages

**Value**

a SummarizedExperiment, with single-isoform transcripts removed, and transcripts now ordered by gene

---

labelKeep	<i>Label rows to keep based on minimal count</i>
-----------	--

---

### Description

Adds a column keep to mcols(y) that specifies which rows of the SummarizedExperiment will be included in statistical testing. Rows are not removed, just marked with the logical keep.

### Usage

```
labelKeep(y, minCount = 10, minN = 3, x)
```

### Arguments

y	a SummarizedExperiment
minCount	the minimum count
minN	the minimum sample size at minCount
x	the name of the condition variable, will use the smaller of the two groups to set minN. Similar to edgeR's filterByExpr, as the smaller group grows past 10, minN grows only by 0.7 increments of sample size

### Value

a SummarizedExperiment with a new column keep in mcols(y)

### Examples

```
y <- makeSimSwishData()
y <- scaleInfReps(y)
y <- labelKeep(y)
```

---

makeInfReps	<i>Make pseudo-inferential replicates from mean and variance</i>
-------------	--

---

### Description

Makes pseudo-inferential replicate counts from mean and variance assays. The simulated counts are drawn from a negative binomial distribution, with mu=mean and size set using a method of moments estimator for dispersion.

### Usage

```
makeInfReps(y, numReps, minDisp = 0.001)
```

## Arguments

y	a SummarizedExperiment
numReps	how many inferential replicates
minDisp	the minimal dispersion value, set after method of moments estimation from inferential mean and variance

## Details

Note that these simulated counts only reflect marginal variance (one transcript or gene at a time), and do not capture the covariance of counts across transcripts or genes, unlike imported inferential replicate data. Therefore, `makeInfReps` should not be used with `summarizeToGene` to create gene-level inferential replicates if inferential replicates were originally created on the transcript level. Instead, import the original inferential replicates.

## Value

a SummarizedExperiment

## References

Van Buren, S., Sarkar, H., Srivastava, A., Rashid, N.U., Patro, R., Love, M.I. (2020) Compression of quantification uncertainty for scRNA-seq counts. bioRxiv. <https://doi.org/10.1101/2020.07.06.189639>

## Examples

```
library(SummarizedExperiment)
mean <- matrix(1:4, ncol=2)
variance <- mean
se <- SummarizedExperiment(list(mean=mean, variance=variance))
se <- makeInfReps(se, numReps=50)
```

---

makeSimSwishData

*Make simulated data for swish for examples/testing*

---

## Description

Makes a small swish dataset for examples and testing. The first six genes have some differential expression evidence in the counts, with varying degree of inferential variance across inferential replicates (1-2: minor, 3-4: some, 5-6: substantial). The 7th and 8th genes have all zeros to demonstrate `labelKeep`.

## Usage

```
makeSimSwishData(
  m = 1000,
  n = 10,
  numReps = 20,
  null = FALSE,
  meanVariance = FALSE
)
```

**Arguments**

m	number of genes
n	number of samples
numReps	how many inferential replicates to generate
null	logical, whether to make an all null dataset
meanVariance	logical, whether to output only mean and variance of inferential replicates

**Value**

a SummarizedExperiment

**Examples**

```
library(SummarizedExperiment)
y <- makeSimSwishData()
assayNames(y)
```

---

 miniSwish

*Helper function for distributing Swish on a subset of data*


---

**Description**

This function is called by the Snakefile that is generated by [splitSwish](#). See alevin example in the vignette. As such, it doesn't need to be run by users in an interactive R session.

**Usage**

```
miniSwish(
  infile,
  outfile,
  numReps = 20,
  lengthCorrect = FALSE,
  overwrite = FALSE,
  ...
)
```

**Arguments**

infile	path to an RDS file of a SummarizedExperiment
outfile	a CSV file to write out
numReps	how many inferential replicates to generate
lengthCorrect	logical, see <a href="#">scaleInfReps</a> , and Swish vignette. As this function is primarily for alevin, the default is FALSE
overwrite	logical, whether outfile should overwrite an existing file
...	arguments passed to <a href="#">swish</a>



**Details**

Note that the default for length correction is FALSE, as opposed to the default in [scaleInfReps](#) which is TRUE. The default for numReps here is 20.

**Value**

nothing, files are written out

---

plotInfReps	<i>Plot inferential replicates for a gene or transcript</i>
-------------	---

---

**Description**

For datasets with inferential replicates, boxplots are drawn for the two groups and potentially grouped by covariates. For datasets with only mean and variance, points and intervals (95 approximation) are drawn.

**Usage**

```
plotInfReps(
  y,
  idx,
  x,
  cov = NULL,
  colsDrk = c("dodgerblue", "goldenrod4", "royalblue4", "red3", "purple4", "darkgreen"),
  colsLgt = c("lightblue1", "goldenrod1", "royalblue1", "salmon1", "orchid1",
    "limegreen"),
  xaxis,
  xlab,
  ylim,
  main,
  mainCol,
  legend = FALSE,
  legendPos = "topleft",
  legendTitle = FALSE,
  legendCex = 1,
  useMean = TRUE,
  applySF = FALSE,
  reorder,
  thin
)
```

**Arguments**

y	a SummarizedExperiment (see <a href="#">swish</a> )
idx	the name or row number of the gene or transcript
x	the name of the condition variable for splitting and coloring the samples or cells. Also can be a numeric, e.g. pseudotime, in which case, cov can be used to designate groups for coloring
cov	the name of the covariate for adjustment

colsDrk	dark colors for the lines of the boxes
colsLgt	light colors for the inside of the boxes
xaxis	logical, whether to label the sample numbers. default is TRUE if there are less than 30 samples
xlab	the x-axis label
ylim	y limits
main	title
mainCol	name of metadata column to use for title (instead of rowname)
legend	logical, show simple legend (default FALSE)
legendPos	character, position of the legend (default "topleft")
legendTitle	logical, whether to add the name of the grouping variable as a title on the legend (default FALSE)
legendCex	numeric, size of the legend (default 1)
useMean	logical, when inferential replicates are not present, use the mean assay or the counts assay for plotting
applySF	logical, when inferential replicates are not present, should y\$sizeFactor be divided out from the mean and interval plots (default FALSE)
reorder	logical, should points within a group defined by condition and covariate be re-ordered by their count value (default is FALSE, except for alevin data)
thin	integer, should the mean and interval lines be drawn thin (the default switches from 0 [not thin] to 1 [thinner] at n=150 cells, and from 1 [thinner] to 2 [thinnest] at n=400 cells)

**Value**

nothing, a plot is displayed

**Examples**

```
y <- makeSimSwishData()
plotInfReps(y, 3, "condition")

y <- makeSimSwishData(n=40)
y$batch <- factor(rep(c(1,2,3,1,2,3),c(5,10,5,5,10,5)))
plotInfReps(y, 3, "condition", "batch")
```

---

plotMASwish

*MA plot*

---

**Description**

MA plot

**Usage**

```
plotMASwish(y, alpha = 0.05, sigcolor = "blue", ...)
```

**Arguments**

`y` a SummarizedExperiment (see `swish`)  
`alpha` the FDR threshold for coloring points  
`sigcolor` the color for the significant points  
`...` passed to `plot`

**Value**

nothing, a plot is displayed

**Examples**

```

y <- makeSimSwishData()
y <- scaleInfReps(y)
y <- labelKeep(y)
y <- swish(y, x="condition")
plotMASwish(y)

```

---

readEDS

*readEDS - a utility function for quickly reading in Alevin's EDS format*


---

**Description**

readEDS - a utility function for quickly reading in Alevin's EDS format

**Usage**

```
readEDS(numOfGenes, numOfOriginalCells, countMatFilename, tierImport = FALSE)
```

**Arguments**

`numOfGenes` number of genes  
`numOfOriginalCells`  
number of cells  
`countMatFilename`  
pointer to the EDS file, `quants_mat.gz`  
`tierImport` whether the `countMatFilename` refers to a quants tier file

**Value**

a genes x cells sparse matrix, of the class `dgCMatrix`

---

scaleInfReps                      *Scale inferential replicate counts*

---

### Description

A helper function to scale the inferential replicates to the mean sequencing depth. The scaling takes into account a robust estimator of size factor (median ratio method is used). First, counts are corrected per row using the effective lengths (for gene counts, the average transcript lengths), then scaled per column to the geometric mean sequence depth, and finally are adjusted per-column up or down by the median ratio size factor to minimize systematic differences across samples.

### Usage

```
scaleInfReps(
  y,
  lengthCorrect = TRUE,
  meanDepth = NULL,
  sfFun = NULL,
  minCount = 10,
  minN = 3,
  quiet = FALSE
)
```

### Arguments

<code>y</code>	a SummarizedExperiment with: <code>infReps</code> a list of inferential replicate count matrices, counts the estimated counts matrix, and <code>length</code> the effective lengths matrix
<code>lengthCorrect</code>	whether to use effective length correction (default is TRUE)
<code>meanDepth</code>	(optional) user can specify a different mean sequencing depth. By default the geometric mean sequencing depth is computed
<code>sfFun</code>	(optional) size factors function. An alternative to the median ratio can be provided here to adjust the scaledTPM so as to remove remaining library size differences. Alternatively, one can provide a numeric vector of size factors
<code>minCount</code>	for internal filtering, the minimum count
<code>minN</code>	for internal filtering, the minimum sample size at <code>minCount</code>
<code>quiet</code>	display no messages

### Value

a SummarizedExperiment with the inferential replicates as scaledTPM with library size already corrected (no need for further normalization). A column `log10mean` is also added which is the  $\log_{10}$  of the mean of scaled counts across all samples and all inferential replicates.

### Examples

```
y <- makeSimSwishData()
y <- scaleInfReps(y)
```

---

splitSwish	<i>Function for splitting SummarizedExperiment into separate RDS files</i>
------------	--

---

## Description

The `splitSwish` function splits up the `y` object along genes and writes a Snakefile that can be used with Snakemake to distribute running `swish` across genes. This workflow is primarily designed for large single cell datasets, and so the default is to not perform length correction within the distributed jobs. See the `alevin` section of the vignette for an example. See the Snakemake documentation for details on how to run and customize a Snakefile: <https://snakemake.readthedocs.io>

## Usage

```
splitSwish(y, nsplits, prefix = "swish", snakefile = NULL, overwrite = FALSE)
```

## Arguments

<code>y</code>	a <code>SummarizedExperiment</code>
<code>nsplits</code>	integer, how many pieces to break <code>y</code> into
<code>prefix</code>	character, the path of the RDS files to write out, e.g. <code>prefix="/path/to/swish"</code> will generate <code>swish.rds</code> files at this path
<code>snakefile</code>	character, the path of a Snakemake file, e.g. <code>Snakefile</code> , that should be written out. If <code>NULL</code> , then no Snakefile is written out
<code>overwrite</code>	logical, whether the snakefile and RDS files ( <code>swish1.rds</code> , ...) should overwrite existing files

## Value

nothing, files are written out

## References

Compression and splitting across jobs:

Van Buren, S., Sarkar, H., Srivastava, A., Rashid, N.U., Patro, R., Love, M.I. (2020) Compression of quantification uncertainty for scRNA-seq counts. *bioRxiv*. <https://doi.org/10.1101/2020.07.06.189639>

Snakemake:

Koster, J., Rahmann, S. (2012) Snakemake - a scalable bioinformatics workflow engine. *Bioinformatics*. <https://doi.org/10.1093/bioinformatics/bts480>

swish

*swish: SAMseq With Inferential Samples Helps***Description**

Performs non-parametric inference on rows of  $y$  for various experimental designs. See References for details.

**Usage**

```
swish(
  y,
  x,
  cov = NULL,
  pair = NULL,
  interaction = FALSE,
  nperms = 100,
  estPi0 = FALSE,
  qvaluePkg = "qvalue",
  pc = 5,
  nRandomPairs = 30,
  fast = 1,
  returnNulls = FALSE,
  quiet = FALSE
)
```

**Arguments**

<code>y</code>	a SummarizedExperiment containing the inferential replicate matrices of median-ratio-scaled TPM as assays 'infRep1', 'infRep2', etc.
<code>x</code>	the name of the condition variable. A factor with two levels for a two group analysis (possible to adjust for covariate or matched samples, see next two arguments)
<code>cov</code>	the name of the covariate for adjustment. If provided a stratified Wilcoxon is performed. Cannot be used with <code>pair</code>
<code>pair</code>	the name of the pair variable, which should be the number of the pair. Can be an integer or factor. If specified, a signed rank test is used to build the statistic. All samples across <code>x</code> must be pairs if this is specified. Cannot be used with <code>cov</code> .
<code>interaction</code>	logical, whether to perform a test of an interaction between <code>x</code> and <code>cov</code> . See Details.
<code>nperms</code>	the number of permutations. if set above the possible number of permutations, the function will print a message that the value is set to the maximum number of permutations possible
<code>estPi0</code>	logical, whether to estimate $\pi_0$
<code>qvaluePkg</code>	character, which package to use for q-value estimation, <code>samr</code> or <code>qvalue</code>
<code>pc</code>	pseudocount for finite estimation of $\log_2FC$ , not used in calculation of test statistics, <code>locfdr</code> or <code>qvalue</code>
<code>nRandomPairs</code>	the number of random pseudo-pairs (only used with <code>interaction=TRUE</code> and un-matched samples) to use to calculate the test statistic

<code>fast</code>	an integer, toggles different methods based on speed ( <code>fast=1</code> is default, <code>0</code> is slower). See Details.
<code>returnNulls</code>	logical, only return the stat vector, the <code>log2FC</code> vector, and the nulls matrix (default <code>FALSE</code> )
<code>quiet</code>	display no messages

## Details

**interaction:** The interaction tests are different than the other tests produced by `swish`, in that they focus on a difference in the `log2` fold change across levels of `x` when comparing the two levels in `cov`. If `pair` is specified, this will perform a Wilcoxon rank sum test on the two groups of matched sample LFCs. If `pair` is not included, multiple random pairs of samples within the two groups are chosen, and again a Wilcoxon rank sum test compared the LFCs across groups.

**fast:** '0' involves recomputing ranks of the inferential replicates for each permutation, '1' (default) is roughly 10x faster by avoiding re-computing ranks for each permutation. The `fast` argument is only relevant for the following three experimental designs: (1) two group Wilcoxon, (2) stratified Wilcoxon, e.g. `cov` is specified, and (3) the paired interaction test, e.g. `pair` and `cov` are specified. For paired design and general interaction test, there are not fast/slow alternatives.

## Value

a `SummarizedExperiment` with metadata columns added: the statistic (either a centered Wilcoxon Mann-Whitney or a signed rank statistic, aggregated over inferential replicates), a `log2` fold change (the median over inferential replicates, and averaged over pairs or groups (if groups, weighted by sample size), the local FDR and q-value, as estimated by the `samr` package.

## References

The citation for `swish` method is:

Anqi Zhu, Avi Srivastava, Joseph G Ibrahim, Rob Patro, Michael I Love "Nonparametric expression analysis using inferential replicate counts" *Nucleic Acids Research* (2019). <https://doi.org/10.1093/nar/gkz622>

The `swish` method builds upon the SAMseq method, and extends it by incorporating inferential uncertainty, as well as providing methods for additional experimental designs (see vignette).

For reference, the publication describing the SAMseq method is:

Jun Li and Robert Tibshirani "Finding consistent patterns: A nonparametric approach for identifying differential expression in RNA-Seq data" *Stat Methods Med Res* (2013). <https://doi.org/10.1177/0962280211428386>

## Examples

```
library(SummarizedExperiment)
set.seed(1)
y <- makeSimSwishData()
y <- scaleInfReps(y)
y <- labelKeep(y)
y <- swish(y, x="condition")

# histogram of the swish statistics
hist(mcols(y)$stat, breaks=40, col="grey")
cols = rep(c("blue", "purple", "red"), each=2)
```

```
for (i in 1:6) {  
  arrows(mcols(y)$stat[i], 20,  
         mcols(y)$stat[i], 10,  
         col=cols[i], length=.1, lwd=2)  
}  
  
# plot inferential replicates  
plotInfReps(y, 1, "condition")  
plotInfReps(y, 3, "condition")  
plotInfReps(y, 5, "condition")
```



# Index

- \* **package**
  - fishpond-package, 2
- addStatsFromCSV, 3
- computeInfRV, 4
- deswish, 4
- fishpond-package, 2
- isoformProportions, 2, 5
- labelKeep, 2, 6
- makeInfReps, 2, 6
- makeSimSwishData, 7
- miniSwish, 8
- plotInfReps, 2, 9
- plotMASwish, 2, 10
- readEDS, 11
- scaleInfReps, 2, 8, 9, 12
- splitSwish, 2, 3, 8, 13
- swish, 2, 3, 8, 14