

# Package ‘PathoStat’

April 15, 2017

**Type** Package

**Title** PathoStat Statistical Microbiome Analysis Package

**Version** 1.0.0

**Date** 2016-10-11

**Author** Solaiappan Manimaran <manimaran\_1975@hotmail.com>, Matthew Bendall <bendall@gwmail.gwu.edu>, Sandro Valenzuela Diaz <sandrolvalenzuelad@gmail.com>, Eduardo Castro <castronallar@gmail.com>, Tyler Faits <tfaits@gmail.com>, W. Evan Johnson <wej@bu.edu>

**Maintainer** Solaiappan Manimaran <manimaran\_1975@hotmail.com>

**Description** The purpose of this package is to perform Statistical Microbiome Analysis on metagenomics results from sequencing data samples. In particular, it supports analyses on the PathoScope generated report files. PathoStat provides various functionalities including Relative Abundance charts, Diversity estimates and plots, tests of Differential Abundance, Time Series visualization, and Core OTU analysis.

**URL** <https://github.com/mani2012/PathoStat>

**BugReports** <https://github.com/mani2012/PathoStat/issues>

**License** GPL (>= 2)

**Depends** R (>= 3.3.1)

**Imports** MCMCpack, limma, corpcor, rmarkdown, knitr, pander, matrixStats, reshape2, scales, ggplot2, rentrez, BatchQC, DT, gtools, tidyr, plyr, dplyr, ape, phyloseq, shiny, grDevices, stats, methods, XML, graphics, utils, alluvial, BiocStyle

**Collate** 'pathoStat.R' 'utils.R' 'taxonomy.R' 'confRegion.R' 'allClasses.R' 'coreOTUModule.R'

**Suggests** testthat

**biocViews** Microbiome, Metagenomics, GraphAndNetwork, Microarray, PatternLogic, PrincipalComponent, Sequencing, Software, Visualization, RNASeq

**SystemRequirements** pandoc (<http://pandoc.org/installing.html>) for generating reports from markdown files.

**VignetteBuilder** knitr

**RoxygenNote** 5.0.1

**NeedsCompilation** no

## R topics documented:

coreOTUModule . . . . .	2
coreOTUModuleUI . . . . .	3
createPathoStat . . . . .	4
findRAfromCount . . . . .	5
findTaxonLevel . . . . .	5
findTaxonMat . . . . .	6
findTaxonomy . . . . .	7
formatTaxTable . . . . .	7
getShinyInput . . . . .	8
getShinyInputCombat . . . . .	8
getShinyInputOrig . . . . .	9
get_core . . . . .	9
get_coremat . . . . .	10
get_coremat_lineplot . . . . .	10
grepTid . . . . .	11
loadPathoscopeReports . . . . .	11
loadPstat . . . . .	12
log2CPM . . . . .	13
pathostat . . . . .	13
PathoStat-class . . . . .	14
plotConfRegion . . . . .	14
pstat_data . . . . .	15
readPathoscopeData . . . . .	15
runPathoStat . . . . .	16
savePstat . . . . .	17
setShinyInput . . . . .	17
setShinyInputCombat . . . . .	18
setShinyInputOrig . . . . .	18
<b>Index</b> . . . . .	<b>19</b>

---

coreOTUModule	<i>Server function for Core OTU Module</i>
---------------	--

---

### Description

This function provides the server logic for the Core OTU tab. This function is not called directly; instead, it should be invoked within the Shiny app's server function using the `callModule` function. See <http://shiny.rstudio.com/articles/modules.html> for information about this design pattern.

### Usage

```
coreOTUModule(input, output, session, pstat)
```

## Arguments

input	Shiny server input object created by callModule
output	Shiny server output object created by callModule
session	Session created by callModule
pstat	PathoStat object (third argument to callModule).

## Details

The `callModule` function should be invoked with this function as the first argument. `callModule` is responsible for creating the namespaced input, output, and session arguments. The second argument to `callModule` is the ID to be used for the namespace and *must* match the `id` argument provided to `coreOTUModuleUI`. The third argument to `callModule` should be a `PathoStat` object from the app's server function, and is passed to this function as the `pstat` argument.

## Value

None

## See Also

`coreOTUModuleUI` for the UI function, `callModule` to see how to invoke this function, or <http://shiny.rstudio.com/articles/modules.html> for more information about Shiny modules.

## Examples

```
# This function is not called directly; instead, it should be invoked within
# the app's server function using the shiny::callModule function.
## Not run:
shinyServer(function(input, output, session) {
  shinyInput <- getShinyInput()
  pstat <- shinyInput$pstat
  callModule( coreOTUModule, "coreOTUModule", pstat )
})

## End(Not run)
```

---

coreOTUModuleUI

*UI function for Core OTU Module*

---

## Description

This function creates the UI for the Core OTU tab. The tab panel can be included within a `tabsetPanel`, thus providing a simple way to add or remove this module from the Shiny app. The first argument, `id`, is the ID to be used for the namespace *and* must match the `id` argument provided to `coreOTUModule`.

## Usage

```
coreOTUModuleUI(id, label = "Core OTUs")
```

**Arguments**

id	Namespace for module
label	Tab label

**Value**

A `tabPanel` that can be included within a `tabsetPanel`.

**See Also**

`coreOTUModule` for the server function, `tabPanel` for the UI component returned by this function, or <http://shiny.rstudio.com/articles/modules.html> for more information about Shiny modules.

**Examples**

```
shiny::mainPanel(
  shiny::tabsetPanel(
    coreOTUModuleUI("coreOTUModule")
  )
)
```

---

createPathoStat	<i>Generates a PathoStat object from the PathoScope reports for further analysis using the interactive shiny app</i>
-----------------	--

---

**Description**

Generates a PathoStat object from the PathoScope reports for further analysis using the interactive shiny app

**Usage**

```
createPathoStat(input_dir = ".", sample_data_file = "sample_data.tsv",
  pathoreport_file_suffix = "-sam-report.tsv")
```

**Arguments**

input_dir	Directory where the tsv files from PathoScope are located
sample_data_file	Sample Data file with information about samples
pathoreport_file_suffix	PathoScope report files suffix

**Value**

pstat The pathostat object generated from the given tsv files

**Examples**

```
example_data_dir <- system.file("example/data", package = "PathoStat")
pstat <- createPathoStat(input_dir=example_data_dir,
  sample_data_file="sample_data.tsv")
```

---

findRAfromCount	<i>Return the Relative Abundance (RA) data for the given count OTU table</i>
-----------------	--

---

**Description**

Return the Relative Abundance (RA) data for the given count OTU table

**Usage**

```
findRAfromCount(count_otu)
```

**Arguments**

count\_otu      Count OTU table

**Value**

ra\_otu Relative Abundance (RA) OTU table

**Examples**

```
data_dir <- system.file("data", package = "PathoStat")
infileName <- "pstat_data.rda"
pstat <- loadPstat(data_dir, infileName)
ra_otu <- findRAfromCount(phyloseq::otu_table(pstat))
```

---

findTaxonLevel	<i>Find the taxonomy for the given taxon id</i>
----------------	---

---

**Description**

Find the taxonomy for the given taxon id

**Usage**

```
findTaxonLevel(tid)
```

**Arguments**

tid              Given taxon id

**Value**

taxonomy LineageEx

## Examples

```
example_data_dir <- system.file("example/data", package = "PathoStat")
pathoreport_file_suffix <- "-sam-report.tsv"
datlist <- readPathoscopeData(example_data_dir, pathoreport_file_suffix)
dat <- datlist$data
ids <- rownames(dat)
tids <- unlist(lapply(ids, FUN = grepTid))
taxonLevel <- findTaxonomy(tids[1])
```

---

findTaxonMat

*Find the Taxonomy Information Matrix*

---

## Description

Find the Taxonomy Information Matrix

## Usage

```
findTaxonMat(names, taxonLevels)
```

## Arguments

names	Row names of the taxonomy matrix
taxonLevels	Taxon Levels of all tids

## Value

taxmat Taxonomy Information Matrix

## Examples

```
example_data_dir <- system.file("example/data", package = "PathoStat")
pathoreport_file_suffix <- "-sam-report.tsv"
datlist <- readPathoscopeData(example_data_dir, pathoreport_file_suffix)
dat <- datlist$data
ids <- rownames(dat)
tids <- unlist(lapply(ids, FUN = grepTid))
taxonLevels <- findTaxonomy(tids[1:5])
taxmat <- findTaxonMat(ids[1:5], taxonLevels)
```

---

findTaxonomy	<i>Find the taxonomy for each taxon ids</i>
--------------	---

---

**Description**

Find the taxonomy for each taxon ids

**Usage**

```
findTaxonomy(tids)
```

**Arguments**

tids                    Given taxonomy ids

**Value**

taxondata Data with the taxonomy information

**Examples**

```
example_data_dir <- system.file("example/data", package = "PathoStat")
pathoreport_file_suffix <- "-sam-report.tsv"
datlist <- readPathoscopeData(example_data_dir, pathoreport_file_suffix)
dat <- datlist$data
ids <- rownames(dat)
tids <- unlist(lapply(ids, FUN = grepTid))
taxonLevels <- findTaxonomy(tids[1:5])
```

---

formatTaxTable	<i>Format taxonomy table for rendering</i>
----------------	--

---

**Description**

Format taxonomy table for rendering

**Usage**

```
formatTaxTable(ttable)
```

**Arguments**

ttable                    Taxonomy table

**Value**

Formatted table suitable for rendering with. DT::renderDataTable

---

getShinyInput	<i>Getter function to get the shinyInput option</i>
---------------	---

---

**Description**

Getter function to get the shinyInput option

**Usage**

```
getShinyInput()
```

**Value**

shinyInput option

**Examples**

```
getShinyInput()
```

---

getShinyInputCombat	<i>Getter function to get the shinyInputCombat option</i>
---------------------	---

---

**Description**

Getter function to get the shinyInputCombat option

**Usage**

```
getShinyInputCombat()
```

**Value**

shinyInputCombat option

**Examples**

```
getShinyInputCombat()
```

---

getShinyInputOrig	<i>Getter function to get the shinyInputOrig option</i>
-------------------	---

---

**Description**

Getter function to get the shinyInputOrig option

**Usage**

```
getShinyInputOrig()
```

**Value**

shinyInputOrig option

**Examples**

```
getShinyInputOrig()
```

---

get_core	<i>Select rows of OTU matrix that meet given detection and prevalence thresholds</i>
----------	--

---

**Description**

Select rows of OTU matrix that meet given detection and prevalence thresholds

**Usage**

```
get_core(pstat, detection, prevalence)
```

**Arguments**

pstat	PathoStat object
detection	An integer specifying the minimum value considered to be "detected"
prevalence	An integer specifying the minimum number of samples that must be detected

**Value**

Subsetted PathoStat object

---

get_coremat	<i>Create core OTU matrix containing number of OTUs detected at varying detection and prevalence thresholds.</i>
-------------	--

---

**Description**

Create core OTU matrix containing number of OTUs detected at varying detection and prevalence thresholds.

**Usage**

```
get_coremat(pstat)
```

**Arguments**

pstat	PathoStat object
-------	------------------

**Value**

Data frame containing number of OTUs at varying detection and prevalence thresholds, with rows corresponding to number of samples and columns corresponding to detection thresholds. An additional column called "prev" contains the sample threshold for each row.

---

get_coremat_lineplot	<i>Create line plot from core OTU matrix</i>
----------------------	--

---

**Description**

Create line plot from core OTU matrix

**Usage**

```
get_coremat_lineplot(coremat)
```

**Arguments**

coremat	Core OTU matrix (data.frame)
---------	------------------------------

**Value**

Line plot with number of OTUs on the x-axis and detection threshold on the y-axis. Lines connect data points with the same number of samples.

---

grepTid	<i>Greps the tid from the given identifier string</i>
---------	---

---

**Description**

Greps the tid from the given identifier string

**Usage**

```
grepTid(id)
```

**Arguments**

id	Given identifier string
----	-------------------------

**Value**

tid string

**Examples**

```
tid <- grepTid("ti|367928|org|Bifidobacterium_adolescentis_ATCC_15703")
```

---

loadPathoscopeReports	<i>Loads all data from a set of PathoID reports. For each column in the PathoID report, construct a matrix where the rows are genomes and the columns are samples. Returns a list where each element is named according to the PathoID column. For example, ret[["Final.Best.Hit.Read.Numbers"]] on the result of this function will get you the final count matrix. Also includes elements "total_reads" and "total_genomes" from the first line of the PathoID report.</i>
-----------------------	--

---

**Description**

Loads all data from a set of PathoID reports. For each column in the PathoID report, construct a matrix where the rows are genomes and the columns are samples. Returns a list where each element is named according to the PathoID column. For example, ret[["Final.Best.Hit.Read.Numbers"]] on the result of this function will get you the final count matrix. Also includes elements "total\_reads" and "total\_genomes" from the first line of the PathoID report.

**Usage**

```
loadPathoscopeReports(reportfiles, nrows = NULL)
```

**Arguments**

reportfiles	Paths to report files
nrows	Option to read first N rows of PathoScope reports

**Value**

Returns a list where each element is named according to the PathoID column. For example, `ret[["Final.Best.Hit.Read.Numbers"]]` on the result of this function will get you the final count matrix. Also includes elements "total\_reads" and "total\_genomes" from the first line of the PathoID report.

**Examples**

```
input_dir <- system.file("example/data", package = "PathoStat")
reportfiles <- list.files(input_dir, pattern = "*-sam-report.tsv",
  full.names = TRUE)
loadPathoscopeReports(reportfiles)
```

---

`loadPstat`*Load the R data(.rda) file with pathostat object*

---

**Description**

Load the R data(.rda) file with pathostat object

**Usage**

```
loadPstat(indir = ".", inFileName = "pstat_data.rda")
```

**Arguments**

<code>indir</code>	Input Directory of the .rda file
<code>infileName</code>	File name of the .rda file

**Value**

pstat pathostat object (NULL if it does not exist)

**Examples**

```
data_dir <- system.file("data", package = "PathoStat")
infileName <- "pstat_data.rda"
pstat <- loadPstat(data_dir, infileName)
```

---

log2CPM	<i>Compute log2(counts per mil reads) and library size for each sample</i>
---------	--

---

**Description**

Compute log2(counts per mil reads) and library size for each sample

**Usage**

```
log2CPM(qcounts, lib.size = NULL)
```

**Arguments**

qcounts	quantile normalized counts
lib.size	default is colsums(qcounts)

**Value**

list containing log2(quantile counts per mil reads) and library sizes

**Examples**

```
example_data_dir <- system.file("example/data", package = "PathoStat")
pathoreport_file_suffix <- "-sam-report.tsv"
datlist <- readPathoscopeData(example_data_dir, pathoreport_file_suffix)
countdat <- datlist$countdata
lcpm <- log2CPM(countdat)
```

---

pathostat	<i>Build PathoStat-class object from its phyloseq component.</i>
-----------	--

---

**Description**

Build PathoStat-class object from its phyloseq component.

**Usage**

```
pathostat(physeq1)
```

**Arguments**

physeq1	phyloseq object
---------	-----------------

**Value**

pstat The pathostat object generated from the given phyloseq object

**Examples**

```
rich_dense_biom = system.file("extdata", "rich_dense_otu_table.biom",
  package="phyloseq")
phyob <- phyloseq::import_biom(rich_dense_biom)
pstat_biom <- pathostat(phyob)
```

---

PathoStat-class	<i>PathoStat class to store PathoStat input data including phyloseq object</i>
-----------------	--

---

**Description**

Contains all currently-supported BatchQC output data classes:

**Details**

slots:

**average\_count** a single object of class otu\_tableOrNULL

**besthit\_count** a single object of class otu\_tableOrNULL

**highconf\_count** a single object of class otu\_tableOrNULL

**lowconf\_count** a single object of class otu\_tableOrNULL

---

plotConfRegion	<i>Compute the confidence region for the given proportions</i>
----------------	--

---

**Description**

Compute the confidence region for the given proportions

**Usage**

```
plotConfRegion(p1, p2, size = 100, uselogit = TRUE, n = 10000,
  seed = 1000)
```

**Arguments**

p1	Read counts for first taxon
p2	Read counts for second taxon
size	Total read counts in the sample
uselogit	Use logit transformation to compute confidence region
n	Total number of simulation points to generate
seed	Seed to use in random simulation

**Value**

Confidence region plot

**Examples**

```
p1 <- 20
p2 <- 25
size <- 200
plotConfRegion(p1, p2, size, uselogit=FALSE)
```

---

pstat_data	<i>pathostat object generated from example pathoscope report files</i>
------------	--

---

**Description**

This example data consists of 33 samples from a diet study with 11 subjects taking 3 different diets in random order

**Usage**

```
pstat
```

**Format**

pathostat object extension of phyloseq-class experiment-level object:

**otu\_table** OTU table with 41 taxa and 33 samples

**sample\_data** Sample Data with 33 samples by 18 sample variables

**tax\_table** Taxonomy Table with 41 taxa by 9 taxonomic ranks

**sample\_data** Phylogenetic Tree with 41 tips and 40 internal nodes

**Value**

pathostat object

---

readPathoscopeData	<i>Reads the data from PathoScope reports and returns a list of final guess relative abundance and count data</i>
--------------------	---

---

**Description**

Reads the data from PathoScope reports and returns a list of final guess relative abundance and count data

**Usage**

```
readPathoscopeData(input_dir = ".",
  pathoreport_file_suffix = "-sam-report.tsv")
```

**Arguments**

input\_dir           Directory where the tsv files from PathoScope are located  
 pathoreport\_file\_suffix   PathoScope report files suffix

**Value**

List of final guess relative abundance and count data

**Examples**

```
example_data_dir <- system.file("example/data", package = "PathoStat")
readPathoscopeData(input_dir=example_data_dir)
```

---

runPathoStat	<i>Statistical Microbiome Analysis on the pathostat input and generates a html report and produces interactive shiny app plots</i>
--------------	--

---

**Description**

Statistical Microbiome Analysis on the pathostat input and generates a html report and produces interactive shiny app plots

**Usage**

```
runPathoStat(pstat = NULL, report_file = "PathoStat_report.html",
             report_dir = ".", report_option_binary = "111111111",
             view_report = FALSE, interactive = TRUE)
```

**Arguments**

pstat	phyloseq extension pathostat object
report_file	Output report file name
report_dir	Output report directory path
report_option_binary	9 bits Binary String representing the plots to display and hide in the report
view_report	when TRUE, opens the report in a browser
interactive	when TRUE, opens the interactive shinyApp

**Value**

outputfile The output file with all the statistical plots

**Examples**

```
runPathoStat(interactive = FALSE)
```

---

savePstat	<i>Save the pathostat object to R data(.rda) file</i>
-----------	---

---

**Description**

Save the pathostat object to R data(.rda) file

**Usage**

```
savePstat(pstat, outdir = ".", outfileName = "pstat_data.rda")
```

**Arguments**

pstat	pathostat object
outdir	Output Directory of the .rda file
outfileName	File name of the .rda file

**Value**

outfile .rda file

**Examples**

```
data(pstat_data)  
outfile <- savePstat(pstat)
```

---

setShinyInput	<i>Setter function to set the shinyInput option</i>
---------------	---

---

**Description**

Setter function to set the shinyInput option

**Usage**

```
setShinyInput(x)
```

**Arguments**

x	shinyInput option
---	-------------------

**Value**

shinyInput option

**Examples**

```
setShinyInput(NULL)
```

---

setShinyInputCombat     *Setter function to set the shinyInputCombat option*

---

**Description**

Setter function to set the shinyInputCombat option

**Usage**

```
setShinyInputCombat(x)
```

**Arguments**

x                    shinyInputCombat option

**Value**

shinyInputCombat option

**Examples**

```
setShinyInputCombat(NULL)
```

---

setShinyInputOrig     *Setter function to set the shinyInputOrig option*

---

**Description**

Setter function to set the shinyInputOrig option

**Usage**

```
setShinyInputOrig(x)
```

**Arguments**

x                    shinyInputOrig option

**Value**

shinyInputOrig option

**Examples**

```
setShinyInputOrig(NULL)
```

# Index

## \*Topic **datasets**

pstat\_data, 15

callModule, 2, 3

coreOTUModule, 2, 3, 4

coreOTUModuleUI, 3, 3

createPathoStat, 4

findRAfromCount, 5

findTaxonLevel, 5

findTaxonMat, 6

findTaxonomy, 7

formatTaxTable, 7

get\_core, 9

get\_coremat, 10

get\_coremat\_lineplot, 10

getShinyInput, 8

getShinyInputCombat, 8

getShinyInputOrig, 9

grepTid, 11

loadPathoscopeReports, 11

loadPstat, 12

log2CPM, 13

PathoStat, 3

pathostat, 13

PathoStat-class, 14

pathostat1 (PathoStat-class), 14

plotConfRegion, 14

pstat (pstat\_data), 15

pstat\_data, 15

readPathoscopeData, 15

runPathoStat, 16

savePstat, 17

setShinyInput, 17

setShinyInputCombat, 18

setShinyInputOrig, 18

tabPanel, 4

tabsetPanel, 4