Package 'strollur'

June 24, 2026

Type Package

Title Store and Transfer Amplicon Sequence Data

Version 0.1.2

Date 2026-06-09

Maintainer Pat Schloss <pschloss@umich.edu>

Description Stores the data associated with your amplicon sequence analysis. This includes nucleotide sequences, abundance, sample and treatment assignments, taxonomic classifications, asv, otu and phylotype clusters, metadata, trees and various reports. It is designed to facilitate data analysis across multiple R packages with utility functions to read / write from 'mothur', 'qiime2', 'dada2', and 'phyloseq'.

URL https://github.com/mothur/strollur, https://mothur.org/strollur/

BugReports https://github.com/mothur/strollur/issues

License GPL (>= 3)

Imports Rcpp, cli, methods, microseq, R.utils, R6, waldo, readr, ape, dplyr, tidyr, yaml, rbiom (>= 3.1.0), stats, utils

LinkingTo Rcpp, cli, Rcereal

Depends R (>= 4.5.0)

Suggests knitr, rmarkdown, testthat (>= 3.0.0), xml2, phyloseq, ggplot2, phylotypr, rhdf5, h5lite, pak

Config/testthat/edition 3

Encoding UTF-8

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

RoxygenNote 8.0.0

NeedsCompilation yes

Author Sarah Westcott [aut] (ORCID: <https://orcid.org/0009-0001-1529-8247>), Gregory Johnson [aut] (ORCID: <https://orcid.org/0009-0008-3890-0297>), Pat Schloss [cph, cre] (ORCID: <https://orcid.org/0000-0002-6935-4275>)

Repository CRAN

Date/Publication 2026-06-24 09:00:02 UTC

Contents

---

abundance *Get the abundance data for sequences, bins, samples, and treatments in a* R*hrefhttps://mothur.org/strollur/reference/strollur.htmlstrollur object*

Description

Get the abundance data for sequences, bins, samples, and treatments in a strollur object

Usage

```
abundance(data, type = "sequence", bin_type = "otu", by_sample = FALSE)
```

Arguments

data	a strollur object
type	string containing the type of data you want the number of. Options include: "sequence", "bin", "sample" and "treatment". Default = "sequence".
bin_type	string containing the bin type you would like the abundance data for. Default = "otu".
by_sample	Boolean. When by_sample is TRUE, the abundance data will be parsed by sample. Default = FALSE.

Value

data.frame

Examples

```
miseq <- miseq_sop_example()

# To the total abundance for each sequence
abundance(data = miseq, type = "sequence")

# To the total abundance for each sequence parsed by sample
abundance(data = miseq, type = "sequence", by_sample = TRUE)

# To the total abundance for each "otu" bin
abundance(data = miseq, type = "bin", bin_type = "otu")

# To the total abundance for each "otu" bin parsed by sample
abundance(data = miseq, type = "bin", bin_type = "otu", by_sample = TRUE)

# To the total abundance for each "asv" bin
abundance(data = miseq, type = "bin", bin_type = "asv")

# To the total abundance for each "asv" bin parsed by sample
abundance(data = miseq, type = "bin", bin_type = "asv", by_sample = TRUE)

# To the total abundance for each sample
abundance(data = miseq, type = "sample")

# To the total abundance for each treatment
abundance(data = miseq, type = "treatment")
```

---

add *Add sequences, reports, metadata or resource references to a strollur object*

Description

Add sequences, reports, metadata or resource references to a strollur object

Usage

```
add(
  data,
  table,
  type = "sequence",
  report_type = NULL,
 table_names = list(sequence_name = "sequence_name", sequence = "sequence", comment =
  "comment", reference_vendor = "vendor", reference_name = "name", reference_version =
  "version", reference_usage = "usage", reference_note = "note", reference_method_url =
  "method_url", reference_documentation_url = "documentation_url", reference_parameter
    = "parameter", reference_citation = "citation"),
  reference = NULL,
  verbose = TRUE
)
```

Arguments

data	a strollur object
table	a data.frame containing the data you wish to add.
type	a string containing the type of data. Options include: 'sequence', 'resource_reference' 'metadata' and 'report'.
report_type	a string containing the type of report you are adding. Options include: 'metadata' and custom reports.
table_names	named list used to indicate the names of the columns in the table. By default: table_names <- list(sequence_name = "sequence_name", comment = "comment", sequence = "sequence", reference_name = "name", reference_vendor = "vendor", reference_version = "version", reference_usage = "usage", reference_note = "note", reference_documentation_url = "documentation_url", reference_method_url = "method_url", reference_parameter = "parameter", reference_citation = "citation") In table_names, 'sequence_name' is a string containing the name of the column in 'table' that contains the sequence names. It is used when you are adding FASTA data. Default column name is 'sequence_name'. In table_names, 'sequence' is a string containing the name of the column in 'table' that contains the sequence nucleotide strings. It is used when you are adding FASTA data. Default column name is 'sequence'. In table_names, 'comment' is a string containing the name of the column in 'table' that contains the sequence comments. It is used when you are adding FASTA data. Default column name is 'comment'. In table_names, 'reference_vendor' is a string containing the name of the column in 'table' that contains the reference vendor names. It is used when you are adding reference data. Default column name is 'vendor'. In table_names, 'reference_name' is a string containing the name of the column in 'table' that contains the reference names. It is used when you are adding reference data. Default column name is 'name'.

In table_names, 'reference_version' is a string containing the name of the column in 'table' that contains the reference versions. Default column name is 'version'.

In table_names, 'reference_usage' is a string containing the name of the column in 'table' that contains the reference usages. Default column name is 'usage'.

In table_names, 'reference_note' is a string containing the name of the column in 'table' that contains the reference notes. Default column name is 'note'.

In table_names, 'reference_method_url' is a string containing the name of the column in 'table' that contains the reference method urls. Default column name is 'method_url'.

In table_names, 'reference_documentation_url' is a string containing the name of the column in 'table' that contains the reference urls. Default column name is 'documentation_url'.

In table_names, 'reference_parameter' is a string containing the name of the column in 'table' that contains the reference parameters. Default column name is 'parameter'.

In table_names, 'reference_citation' is a string containing the name of the column in 'table' that contains the reference citations. Default column name is 'citation'.

`reference` a list created by the function [new_reference]. Optional.

`verbose` boolean indicating whether or not you want progress messages. Default = TRUE.

Value

an updated strollur object

Examples

```
# Create a new empty strollur object named 'example_dataset'
data <- new_dataset(dataset_name = "example_dataset")

# Read FASTA data into data.frame
fasta_data <- read_fasta(fasta = strollur_example("final.fasta.gz"))

# Add FASTA sequence data
add(data = data, table = fasta_data, type = "sequence")

# To add FASTA data with a resource reference

# Create a new empty strollur object named 'example_dataset'
data <- new_dataset(dataset_name = "example_dataset")

# Create a resource reference for the FASTA data silva_resource <-
silva_resource <- new_reference(
  vendor = "SILVA", name =
    "silva.bacteria.fasta", version = "1.38.1",
  usage = "alignment of sequences",
  note = "reference trimmed to V4 region", method_url =
    "https://mothur.org/blog/2024/SILVA-v138_2-reference-files/",
```

```
  documentation_url = "https://mothur.org/wiki/silva_reference_files/"
)

# Add FASTA data with a resource reference

add(
  data,
  table = fasta_data,
  type = "sequence",
  reference = silva_resource
)

# Add contigs assembly report with a 'sequence_name' column named 'Name'

contigs_report <- readRDS(strollur_example("miseq_contigs_report.rds"))

add(
  data,
  table = contigs_report, type = "report",
  report_type = "contigs_report", list(sequence_name = "Name")
)

# To add metadata related to your study

metadata <- readRDS(strollur_example("miseq_metadata.rds"))

add(data, table = metadata, type = "metadata")
```

`assign` *Assign sequence abundances, sequence classifications, bins, bin representative sequences, bin classifications or treatments to a* R*hrefhttps://mothur.org/strollur/reference/strollur.htmlstrollur object*

Description

Assign sequence abundances, sequence classifications, bins, bin representative sequences, bin classifications or treatments to a strollur object

Usage

```
assign(
  data,
  table,
  type = "bin",
  bin_type = "otu",
 table_names = list(sequence_name = "sequence_name", abundance = "abundance", sample =
  "sample", treatment = "treatment", taxonomy = "taxonomy", bin_name = "bin_name"),
  reference = NULL,
```

```
  verbose = TRUE
)
```

Arguments

data	a strollur object
table	a data.frame containing the data you wish to assign
type	a string containing the type of data. Options include: 'sequence_abundance', 'sequence_taxonomy', 'bin', 'bin_representative', 'bin_taxonomy' and 'treatment'. Default = "bin".
bin_type	string containing the bin type you would like the number of bins for. Default = "otu".
table_names	named list used to indicate the names of the columns in the table. By default: table_names <- list(sequence_name = "sequence_name", abundance = "abundance", sample = "sample", treatment = "treatment", taxonomy = "taxonomy", bin_name = "bin_name") In table_names, 'sequence_name' is a string containing the name of the column in 'table' that contains the sequence names. Default column name is 'sequence_name'. In table_names, 'abundance' is a string containing the name of the column in 'table' that contains the abundances. Default column name is 'abundance'. In table_names, 'sample' is a string containing the name of the column in 'table' that contains the samples. Default column name is 'sample'. In table_names, 'treatment' is a string containing the name of the column in 'table' that contains the treatment names. Default column name is 'treatment'. In table_names, 'taxonomy' is a string containing the name of the column in 'table' that contains the classifications. Default column name is 'taxonomy'. In table_names, 'bin_name' is a string containing the name of the column in 'table' that contains the bin names. Default column name is 'bin_name'.
reference	a list created by the function [new_reference]. Optional.
verbose	boolean indicating whether or not you want progress messages. Default = TRUE.

Value

an updated strollur object

Examples

```
# Assign sequence classifications

# create a new empty strollur object named 'example_dataset'
data <- new_dataset(dataset_name = "example_dataset")

sequence_classifications <- read_mothur_taxonomy(strollur_example(
  "final.taxonomy.gz"
))
```

```
assign(
  data,
  table = sequence_classifications, type = "sequence_taxonomy"
)

# Assigning bins

# read mothur's otu list file into data.frame
otu_data <- read_mothur_list(list = strollur_example(
  "final.opti_mcc.list.gz"
))

# read mothur's asv list file into data.frame
asv_data <- read_mothur_list(list = strollur_example(
  "final.asv.list.gz"
))

# read mothur's phylotype list file into data.frame
phylo_data <- read_mothur_list(list = strollur_example(
  "final.tx.list.gz"
))

# read otu bin representative sequences into a data.frame
bin_reps <- readRDS(strollur_example("miseq_representative_sequences.rds"))

# assign 'otu' bins using sequence names
assign(data, table = otu_data, bin_type = "otu")

# assign 'asv' bins using sequence names
assign(data, table = asv_data, bin_type = "asv")

# assign 'phylotype' bins using sequence names
assign(data, table = phylo_data, bin_type = "phylotype")

# assign 'otu' bin representative sequences
assign(data, table = bin_reps, type = "bin_representative")

# To assign abundance only bins

# create a new empty strollur object named 'example_dataset'
data <- new_dataset(dataset_name = "example_dataset")

# read mothur's shared file
otu_data <- read_mothur_shared(strollur_example("final.opti_mcc.shared"))

# assign abundance only otus parsed by sample
assign(data, table = otu_data, bin_type = "otu")

# Assigning bin classifications

# read bin taxonomies
otu_data <- read_mothur_cons_taxonomy(strollur_example(
  "final.cons.taxonomy"
```

```
))

# assign otu consensus taxonomies
assign(
  data,
  table = otu_data,
  type = "bin_taxonomy", bin_type = "otu"
)

# Assign treatments

sample_assignments <- readRDS(strollur_example("miseq_sample_design.rds"))

assign(data, table = sample_assignments, type = "treatment")
```

---

clear *clear*

Description

Clear data from a strollur object

Usage

```
clear(data)
```

Arguments

data	a strollur object

Value

an updated strollur object

Examples

```
data <- miseq_sop_example()
clear(data)
```

copy_dataset *copy_dataset*

Description

Create a new strollur object from an existing dataset.

Usage

```
copy_dataset(data)
```

Arguments

data	a strollur object

Value

a strollur object

See Also

The 'new' method in the strollur class

Examples

```
miseq <- miseq_sop_example()

# to create a new dataset that is a copy of miseq

data <- copy_dataset(miseq)
```

count *Find the number of sequences, samples, treatments or bins of a given type in a* R*hrefhttps://mothur.org/strollur/reference/strollur.htmlstrollur object*

Description

Find the number of sequences, samples, treatments or bins of a given type in a strollur object

Usage

```
count(
  data,
  type = "sequence",
  bin_type = "otu",
  samples = NULL,
  distinct = FALSE
)
```

Arguments

data	a strollur object
type	string containing the type of data you want the number of. Options include: "sequence", "sample", "treatment", "bin", and "resource_reference". Default = "sequence".
bin_type	string containing the bin type you would like the number of bins for. Default = "otu".
samples	vector of strings. samples is only used when 'type' = "sequence" or 'type' = "bin" . samples should contain the names of the samples you want the count for. Default = NULL.
distinct	Boolean. distinct is used when 'type' = "sequence" or 'type' = "bin". When 'type' = "sequence" and distinct is TRUE the number of unique sequences is returned. When 'type' = "sequence" and distinct is FALSE the total number of sequences is returned. This can also be combined with samples to find the number of unique sequences found ONLY in a given set of samples, or to find the number of unique sequences in given set of samples that may also be present in other samples. When 'type' = "bin", you can set distinct = TRUE to return the number of bins that ONLY contain sequences from the given samples. When distinct is FALSE the count returned contains bins with sequences from a given samples, but those bins may also contain other samples. Default = FALSE.

Value

double

Examples

```
miseq <- miseq_sop_example()

# To get the total number of sequences
count(data = miseq, type = "sequence")

# To get number of unique sequences
count(data = miseq, type = "sequence", distinct = TRUE)

# To get number of unique sequences from samples 'F3D0' and 'F3D1'
# Note these sequences will be present in both samples but may be
# be present in other samples as well
```

```
count(data = miseq, type = "sequence", samples = c("F3D0", "F3D1"))

# To get number of unique sequences exclusive to samples 'F3D0' and 'F3D1'
# Note sequences are present in both samples and NOT present in any other
# samples.
count(
  data = miseq, type = "sequence", samples = c("F3D0", "F3D1"),
  distinct = TRUE
)

# To get the number of samples in the dataset
count(data = miseq, type = "sample")

# To get the number of treatments in the dataset
count(data = miseq, type = "treatment")

# To get the number of "otu" bins in the dataset
count(data = miseq, type = "bin", bin_type = "otu")

# To get the number of "asv" bins in the dataset
count(data = miseq, type = "bin", bin_type = "asv")

# To get the number of "phylotype" bins in the dataset
count(data = miseq, type = "bin", bin_type = "phylotype")

# To get number of "otu" bins from samples 'F3D0' and 'F3D1'
# Note these bins will have sequences from both samples but there may be
# other samples present as well
count(
  data = miseq,
  type = "bin", bin_type = "otu", samples = c("F3D0", "F3D1")
)

# To get number of "otu" bins unique to samples 'F3D0' and 'F3D1'
# Note these bins will have sequences from both samples and NO other samples
# will be present in the bins.
count(
  data = miseq, type = "bin", bin_type = "otu",
  samples = c("F3D0", "F3D1"), distinct = TRUE
)
```

`export_dataset` *export_dataset*

Description

Export all data from a strollur object.

Usage

```
export_dataset(data)
```

Arguments

data	a strollur object

Value

Rcpp::List, containing the data in the 'Dataset

Examples

```
dataset <- new_dataset("my_dataset")
export_dataset(dataset)
```

---

get_bin_types *get_bin_types*

Description

Get bin table types of a strollur object

Usage

```
get_bin_types(data)
```

Arguments

data	a strollur object

Value

vector of strings

Examples

```
data <- miseq_sop_example()
get_bin_types(data)
```

has_sample *has_sample*

Description

Determine if a given sample is in a strollur object

Usage

```
has_sample(data, sample)
```

Arguments

data	a strollur object.
sample	a string containing the name of a sample.

Value

boolean indicating whether the dataset has a given sample

Examples

```
data <- miseq_sop_example()
has_sample(data, "F3D0")
has_sample(data, "not a valid sample")
```

has_sequence_strings *has_sequence_strings*

Description

Determine if a strollur object contains sequence nucleotide strings.

Usage

```
has_sequence_strings(data)
```

Arguments

data	a strollur object.

Value

boolean indicating whether the dataset has sequence nucleotide strings.

Examples

```
data <- miseq_sop_example()
has_sequence_strings(data)
```

`import_dataset` *Import strollur object from exported data.frame.*

Description

The import_dataset function will create a strollur object from the exported table of a strollur object.

Usage

```
import_dataset(table)
```

Arguments

`table`	a table containing the data from a strollur object. You can create the table using 'export(data)'.

Value

a strollur object

See Also

[export_dataset()]

Examples

```
miseq <- miseq_sop_example()
data <- import_dataset(export_dataset(miseq))
data
```

is_aligned *is_aligned*

Description

Determine if a strollur object contains aligned sequences.

Usage

```
is_aligned(data)
```

Arguments

data	a strollur object

Value

Boolean

Examples

```
dataset <- miseq_sop_example()
is_aligned(dataset)
```

is_equal *is_equal*

Description

Determine if two strollur objects are equal.

Usage

```
is_equal(data, data2)
```

Arguments

data	a strollur object
data2	a strollur object

Value

a logical

Examples

```
miseq <- miseq_sop_example()

data <- copy_dataset(miseq)

is_equal(miseq, data)
```

`load_dataset` *Load strollur object from .rds file*

Description

The load_dataset function will create a strollur object from an RDS file.

Usage

```
load_dataset(file)
```

Arguments

`file`	a string containing the .rds file name.

Value

a strollur object

See Also

[save_dataset()]

Examples

```
data <- load_dataset(strollur_example("miseq_sop.rds"))
data
```

`miseq_sop_example` *Example strollur object*

Description

The miseq_sop_example function will create 'strollur' object using the analysis files from the MiSeq_SOP example.

Usage

```
miseq_sop_example()
```

Value

A 'strollur' object

Examples

```
miseq <- miseq_sop_example()
```

`names` *Get the names of various data in a* R*hrefhttps://mothur.org/strollur/reference/strollur.htmlstrollur object*

Description

Get the names of names sequences, bins, samples, treatments, and reports data in a strollur object

Usage

```
names(
  data,
  type = "sequence",
  bin_type = "otu",
  samples = NULL,
  distinct = FALSE
)
```

Arguments

data	a strollur object
type	string containing the type of data you would like. Options include: "dataset", "sequence", "bin", "sample", "treatment", "report". Default = "sequence".
bin_type	string containing the bin type you would like the names for. Default = "otu".
samples	vector of strings. samples is only used when 'type' = "sequence" or 'type' = "bin" . samples should contain the names of the samples you want names for. Default = NULL.
distinct	Boolean. distinct is used when 'type' = "sequence" or 'type' = "bin" and the samples parameter is used. The distinct parameter allows you to get the names that present given set of samples. When distinct is TRUE, the names function will return the names that ONLY contain data from the given samples. When distinct is FALSE the data returned contains data from a given samples, but may ALSO contain data from other samples. Default = FALSE.

Value

vector of strings, containing the names requested

Examples

```
miseq <- miseq_sop_example()

# To get the name of the dataset
names(data = miseq, type = "dataset")

# To get the names of the sequences
names(data = miseq, type = "sequence")

# To get the names of the sequences present sample 'F3D0'
names(data = miseq, type = "sequence", samples = c("F3D0"))

#' # To get the names of the sequences unique to sample 'F3D0'
names(data = miseq, type = "sequence", samples = c("F3D0"), distinct = TRUE)

# To get the names of the samples
names(data = miseq, type = "sample")

# To get the names of the treatments
names(data = miseq, type = "treatment")

# To get the names of the bins
names(data = miseq, type = "bin")

# To get the names of the bins that are unique to 'F3D0'
names(data = miseq, type = "bin", samples = c("F3D0"), distinct = TRUE)

# To get the names of the bins that include sequences from 'F3D0'
names(data = miseq, type = "bin", samples = c("F3D0"), distinct = FALSE)
```

```
# To get the names of the reports
names(data = miseq, type = "report")
```

`new_dataset` *new_dataset*

Description

Create a new strollur object

Usage

```
new_dataset(dataset_name = "")
```

Arguments

`dataset_name`	string, a string containing the dataset name. Default = ""

Value

a strollur object

See Also

The 'new' method in the strollur class

Examples

```
data <- new_dataset()

# to create a new dataset named "soil", run the following:

data <- new_dataset(dataset_name = "soil")
```

new_reference *new_reference*

Description

Create a resource reference for your strollur object to aid in reproducibility.

Usage

```
new_reference(
  name,
  vendor = "",
  version = "",
  usage = "",
  note = "",
  documentation_url = "",
  method_url = "",
  parameter = "",
  citation = ""
)
```

Arguments

name	a string containing the name of the resource used. For example: 'silva.bacteria.fasta' or 'R package phylotypr'.
vendor	a string containing name of entity that created original resource. example: "Silva" or "Schloss Lab - University of Michigan"
version	a string containing the version of the reference resource. For example: '1.38.1' or '0.1.1'. Default = "".
usage	a string containing the usage of the resource reference in your analysis. For example: 'alignment of sequences' or 'classification of sequences'. Default = "".
note	a string containing additional notes about the resource reference in your analysis. For example: 'alignment reference trimmed to V4 region' or 'classification of sequences using Bayesian method'. Default = "".
documentation_url	a string containing a web address where the reference may be downloaded or documentation may be found. Default = "".
method_url	a string containing any publications describing the methods used by the resource reference. For example: 'doi:10.1128/mra.01144-24'. Default = "".
parameter	a string containing the any specific parameters used by the resource. For example: 'kmer_size = 8, num_bootstraps = 100, min_confidence = 80' Default = "".

citation — a string containing the citation information for the resource reference. For example: "citation_key = "doi:10.1128/AEM.00062-07", author = "Qiong Wang and George M. Garrity and James M. Tiedje and James R. Cole", title = "Naïve Bayesian Classifier for Rapid Assignment of rRNA Sequences into the New Bacterial Taxonomy", journal = "Applied and Environmental Microbiology", volume = "73", number = "16", pages = "5261-5267", year = "2007", doi = "10.1128/AEM.00062-07"". Default = "".

Value

a list

Examples

```
silva_resource <- new_reference(
  vendor = "SILVA", name =
    "silva.bacteria.fasta", version = "1.38.1",
  usage = "alignment of sequences",
  note = "alignment reference trimmed to V4 region", documentation_url =
    "https://mothur.org/wiki/silva_reference_files/", method_url =
    "https://mothur.org/blog/2024/SILVA-v138_2-reference-files/"
)

phylotypr_resource <- new_reference(
  vendor = "Schloss Lab - University of
Michigan", name = "R phylotypr package", version = "0.1.1", usage =
    "classification of sequences",
  note = "classification using Bayesian method",
  parameter = "kmer_size = 8, num_bootstraps = 100, min_confidence = 80",
  documentation_url = "https://mothur.org/phylotypr/", method_url =
    "doi:10.1128/mra.01144-24",
  citation = "@article{doi:10.1128/AEM.00062-07,
author = {Qiong Wang and George M. Garrity and James M. Tiedje and James R.
Cole}, title = {Naïve Bayesian Classifier for Rapid Assignment of rRNA
Sequences into the New Bacterial Taxonomy}, journal = {Applied and
Environmental Microbiology}, volume = {73}, number = {16}, pages =
{5261-5267}, year = {2007}, doi = {10.1128/AEM.00062-07}, URL =
{https://journals.asm.org/doi/abs/10.1128/aem.00062-07}, eprint =
{https://journals.asm.org/doi/pdf/10.1128/aem.00062-07}}"
)
```

---

read_dada2 — *Create a* R*hrefhttps://mothur.org/strollur/reference/strollur.htmlstrollur object from dada2 outputs*

Description

This function reads a dada2 sequence table and creates a ‘strollur‘ object. The dada2 sequence table is a 2D matrix containing the abundance counts by sample for each ASV. The sample names are stored as row names and the sequence nucleotide strings are stored as column names.

To generate the dada2 sequence table from your own files you can follow this dada2 tutorial.

Usage

```
read_dada2(sequence_table, dataset_name = "")
```

Arguments

sequence_table	A dada2 sequence table
dataset_name	A string containing a name for your dataset.

Value

A ‘strollur‘ object

References

Callahan,B.J., McMurdie,P.J., Rosen,M.J., Han,A.W., Johnson,A.J.A. and Holmes,S.P. (2016), DADA2: High-resolution sample inference from Illumina amplicon data. Nature Methods 13:581-583. <doi:10.1038/nmeth.3869>

Examples

```
seqtab <- readRDS(strollur_example("dada2.rds"))
dim(seqtab)

data <- read_dada2(sequence_table = seqtab, dataset_name = "dada2 example")
```

---

read_fasta *read_fasta*

Description

Read a FASTA formatted sequence file

Usage

```
read_fasta(fasta)
```

Arguments

fasta	FASTA file name (required)

Value

A data.frame containing the FASTA sequence data

Examples

```
fasta_data <- read_fasta(strollur_example("final.fasta.gz"))

# fasta_data is a data.frame.
# To access the names of the sequences in the file, run the following:

fasta_data$sequence_name

# To access the sequences in the file, run the following:

fasta_data$sequence
```

---

read_mothur *Create a* R*hrefhttps://mothur.org/strollur/reference/strollur.htmlstrollur object from mothur outputs*

Description

The read_mothur function reads various file types created by mothur, and creates a ‘strollur‘ object.

To generate the various input files you can follow Pat’s Miseq example analysis.

Usage

```
read_mothur(
  fasta = NULL,
  count = NULL,
  taxonomy = NULL,
  otu_list = NULL,
  asv_list = NULL,
  phylo_list = NULL,
  design = NULL,
  cons_taxonomy = NULL,
  otu_shared = NULL,
  asv_shared = NULL,
  phylo_shared = NULL,
  sample_tree = NULL,
  sequence_tree = NULL,
  dataset_name = ""
)
```

Arguments

`fasta`	filename, a FASTA formatted file containing sequence strings. fasta file
`count`	filename, a mothur count file
`taxonomy`	filename, a mothur taxonomy file, created by classify.seqs
`otu_list`	filename, a mothur list file containing otu bin assignments. The otu_list file is created by cluster, cluster.split, and cluster.fit
`asv_list`	filename, a mothur list file containing asv bin assignments. The asv_list file is created by cluster using the 'unique' method.
`phylo_list`	filename, a mothur list file containing phylotype bin assignments. The phylo_list file is created by phylotype.
`design`	filename, a mothur design file
`cons_taxonomy`	filename, a mothur consensus taxonomy file constaxonomy file. The cons_taxonomy file is created by classify.otu.
`otu_shared`	filename, a mothur shared file containing otu bin sample abundance assignments.
`asv_shared`	filename, a mothur shared file containing asv bin sample abundance assignments.
`phylo_shared`	filename, a mothur shared file containing phylotype bin sample abundance assignments.
`sample_tree`	filename, a tree that relates samples. The sample tree is created by tree.shared. We recommend running tree.shared with subsample = true, and using the 'ave.tre' output for best results.
`sequence_tree`	filename, a tree that relates sequences. The sequence tree is created by clearcut. We DO NOT recommend using sequence trees. With the ever growing size of modern datasets, sequence tree can be difficult / impossible to build without hitting a memory limitation.
`dataset_name`	A string containing a name for your dataset.

Value

A strollur object

Note

- *consensus taxonomy*, The 'strollur' object will generate consensus taxonomies for you based on the sequence taxonomy assignment. You only need to provide the ".cons.taxonomy" file if you are not providing sequence taxonomy assignments.
- *shared / rabund file*, The 'strollur' object will generate shared and rabund data for you based on the otu assignment in the list file and the count data. You only need to provide the ".shared" file if you are not providing the list and count files.

References

Schloss,P.D., Westcott,S.L., Ryabin,T., Hall,J.R., Hartmann,M., Hollister,E.B., Lesniewski,R.A., Oakley,B.B., Parks,D.H., Robinson,C.J., Sahl,J.W., Stres,B., Thallinger,G.G., Van Horn,D.J. and Weber,C.F. (2009), Introducing mothur: Open-source, platform-independent, community-supported software for describing and comparing microbial communities. Applied and Environmental Microbiology 75:7537-7541. <doi:10.1128/AEM.01541-09>

Examples

```
# For dataset's including sequence data:

data <- read_mothur(
  fasta = strollur_example("final.fasta.gz"),
  count = strollur_example("final.count_table.gz"),
  taxonomy = strollur_example("final.taxonomy.gz"),
  design = strollur_example("mouse.time.design"),
  otu_list = strollur_example("final.opti_mcc.list.gz"),
  asv_list = strollur_example("final.asv.list.gz"),
  phylo_list = strollur_example("final.tx.list.gz"),
  sample_tree = strollur_example("final.opti_mcc.jclass.ave.tre"),
  dataset_name = "miseq_sop"
)

# For dataset's with only otu data:

data <- read_mothur(
  otu_shared = strollur_example("final.opti_mcc.shared"),
  cons_taxonomy = strollur_example(
    "final.cons.taxonomy"
  ),
  design = strollur_example("mouse.time.design"),
  sample_tree = strollur_example("final.opti_mcc.jclass.ave.tre"),
  dataset_name = "miseq_sop"
)
```

`read_mothur_cons_taxonomy` *read_mothur_cons_taxonomy*

Description

Read a mothur formatted cons_taxonomy file

Usage

```
read_mothur_cons_taxonomy(taxonomy)
```

Arguments

taxonomy	file name, a mothur consensus taxonomy file. The cons_taxonomy file is created by classify.otu.

Value

A data.frame containing the bin names, bin abundances and bin taxonomies.

Examples

```
# You can add the otu assignments and bin taxonomies to the your data set
# using the following:

# read mothur's consensus taxonomy file into a data.frame
otu_data <- read_mothur_cons_taxonomy(strollur_example(
  "final.cons.taxonomy"
))

data <- new_dataset()

# assign abundance only 'otu' bins
assign(data = data, table = otu_data, type = "bin", bin_type = "otu")

# assign consensus taxonomies to 'otu' bins
assign(
  data = data, table = otu_data,
  type = "bin_taxonomy", bin_type = "otu"
)
```

---

read_mothur_count *read_mothur_count*

Description

Read a mothur formatted count file

Usage

```
read_mothur_count(filename)
```

Arguments

filename	count file name (required)

Value

data.frame

Examples

```
# mothur count file
# Representative_Sequence     total   sample2 sample3 sample4
# seq1 1150 250 400 500
# seq2 115 25 40 50
# seq3 50 25 25 0
# seq4 4 0 0 4

# returns
# sequence_name   sample abundance
# <char>  <char>     <int>
#  1:   seq1 sample2       250
#  2:   seq1 sample3       400
#  3:   seq1 sample4       500
#  4:   seq2 sample2        25
#  5:   seq2 sample3        40
#  6:   seq2 sample4        50
#  7:   seq3 sample2        25
#  8:   seq3 sample3        25
#  9:   seq4 sample4         4

# read a count file with samples
sample_table <- read_mothur_count(strollur_example("final.count_table.gz"))

# You can add your sequence abundance data to your `strollur` object as
# follows:

# create a new empty `strollur` object
data <- new_dataset()

# assign sequence abundances parsed by sample
assign(data, table = sample_table, type = "sequence_abundance")

# print summary of data
data
```

`read_mothur_list` *read_mothur_list*

Description

Read a mothur formatted list file

Usage

```
read_mothur_list(list)
```

Arguments

list	file name. The list file can be created using several of mothur's commands. cluster, cluster.split, cluster.fit and phylotype.

Value

A data.frame containing the sequence otu assignments

Examples

```
# You can add your otu assignments to the your data set using the following:

# read mothur's list file into data.frame
otu_data <- read_mothur_list(strollur_example("final.opti_mcc.list.gz"))

# create a new empty `strollur` object
data <- new_dataset()

# assign sequences to 'otu' bins
assign(data = data, table = otu_data, type = "bin", bin_type = "otu")
```

---

read_mothur_rabund *read_mothur_rabund*

Description

Read a mothur formatted rabund file

Usage

```
read_mothur_rabund(rabund)
```

Arguments

rabund	file name (required)

Value

A data.frame containing the sequence otu assignments

Examples

```
# You can add your otu assignments to the your data set using the following:

# read rabund file into data.frame
otu_data <- read_mothur_rabund(
  rabund =
    strollur_example("final.opti_mcc.rabund")
```

```
)

data <- new_dataset()

# assign abundance only 'otu' bins
assign(data = data, table = otu_data, type = "bin", bin_type = "otu")
```

`read_mothur_shared` *read_mothur_shared*

Description

Read a mothur formatted shared file

Usage

```
read_mothur_shared(shared)
```

Arguments

`shared`	file name (required)

Value

A data.frame containing the sequence otu assignments

Examples

```
# You can add your otu assignments to the your data set using the following:

# read mothur shared file into data.frame
otu_data <- read_mothur_shared(strollur_example("final.opti_mcc.shared"))

# create a new empty `strollur` object
data <- new_dataset()

# assign abundance only 'otu' bins parsed by sample
assign(data = data, table = otu_data, type = "bin", bin_type = "otu")
```

read_mothur_taxonomy *read_mothur_taxonomy*

Description

Read a mothur formatted taxonomy file

Usage

```
read_mothur_taxonomy(taxonomy)
```

Arguments

taxonomy	file name. a mothur taxonomy file, created by classify.seqs

Value

A data.frame containing the sequences names and sequences taxonomies.

Examples

```
# You can add the sequences and their taxonomies to the your data set
# using the following:

# read mothur's taxonomy file into a data.frame
classification_data <- read_mothur_taxonomy(strollur_example(
  "final.taxonomy.gz"
))

# create a new empty `strollur` object
data <- new_dataset()

# assign sequence classifications
assign(data = data, table = classification_data, type = "sequence_taxonomy")
```

read_phyloseq *Create a* R*hrefhttps://mothur.org/strollur/reference/strollur.htmlstrollur object from a phyloseq object*

Description

The ‘read_phyloseq()‘ function reads phyloseq objects created from the phyloseq package (https://www.bioconductor.org/pac and converts it into a strollur object.

Usage

```
read_phyloseq(phyloseq_object, treatment_column_name = NULL, dataset_name = "")
```

Arguments

phyloseq_object	the phyloseq object that is returned when using any read function in the phyloseq package. It has to be of type "phyloseq"
treatment_column_name	the column name inside your phyloseq object within your sample data that is used to describe treatments. It must be a character. Defaults to NULL.
dataset_name	A string containing a name for your dataset.

Value

a strollur object.

References

McMurdie,P.J. and Holmes,S. (2013), phyloseq: An R Package for Reproducible Interactive Analysis and Graphics of Microbiome Census Data. PLoS ONE 8:e61217. <doi:10.1371/journal.pone.0061217>

Examples

```
miseq <- miseq_sop_example()

if (requireNamespace("phyloseq", quietly = TRUE)) {
  phylo_obj <- write_phyloseq(miseq)
  miseq_re_read <- read_phyloseq(phylo_obj)
} else {
  message(paste(
    "To use this functionality you have to install the",
    "phyloseq package."
  ))
}
```

---

read_qiime2 *Create a* R*hrefhttps://mothur.org/strollur/reference/strollur.htmlstrollur object from a qiime2 outputs*

Description

The read_qiime2 function reads various types of .qza files created by qiime2, and creates a ‘strollur‘ object.

Usage

```
read_qiime2(
  qza,
  metadata = NULL,
  dataset_name = "",
```

```
  dir_path = NULL,
  remove_unpacked_artifacts = TRUE
)
```

Arguments

qza	vector of filenames, .qza files containing your data from qiime2.
metadata	filename, a .tsv file containing metadata
dataset_name	A string containing a name for your dataset.
dir_path	a string containing the name of directory where the artifacts files should be unpacked. Default = current working directory.
remove_unpacked_artifacts	boolean, When TRUE, the unpacked artifacts and temporary directories will be removed. Default = TRUE.

Value

A 'strollur' object

References

Bolyen,E., Rideout,J.R., Dillon,M.R. et al. (2019), Reproducible, interactive, scalable and extensible microbiome data science using QIIME 2. Bioinformatics 37:852-857. <doi:10.1038/s41587-019-0209-9>

Examples

```
# Using the example files from moving-pictures, we add FASTA data, assign
# taxonomy and abundance for features, and add a newick tree and
# metadata.

qza_files <- c(
  strollur_example("rep_seqs.qza"),
  strollur_example("table.qza"),
  strollur_example("taxonomy.qza"),
  strollur_example("rooted-tree.qza")
)

if (requireNamespace("h5lite", quietly = TRUE)) {
  data <- read_qiime2(
    qza = qza_files,
    metadata = strollur_example("sample_metadata.tsv"),
    dataset_name = "qiime2_moving_pictures"
  )
  data
} else {
  message(paste(
    "To use this functionality you have to install the",
    "h5lite package."
  ))
```

```
}
```

`read_qiime2_feature_table` *read_qiime2_feature_table*

Description

Read a qiime2 qza containing bin data

Usage

```
read_qiime2_feature_table(
  qza,
  dir_path = NULL,
  remove_unpacked_artifacts = TRUE
)
```

Arguments

`qza`	file name, a qiime2 .qza file containing bin data.
`dir_path`	a string containing the name of directory where the artifacts files should be unpacked. Default = current working directory.
`remove_unpacked_artifacts`	boolean, When TRUE, the artifact's temporary directories will be removed after processing. Default = TRUE.

Value

A list containing artifact

Examples

```
if (requireNamespace("h5lite", quietly = TRUE)) {
  artifact <- read_qiime2_feature_table(strollur_example("table.qza"))

  # access the bin assignment table

  artifact$data

  # to create a `strollur` object with your data

  data <- new_dataset("my_data")

  assign(data = data, table = artifact$data, type = "bin")
  data
} else {
```

```
  message(paste(
    "To use this functionality you have to install the",
    "h5lite package."
  ))
}
```

`read_qiime2_metadata` *read_qiime2_metadata*

Description

Read a qiime2 .tsv table containing metadata.

Usage

```
read_qiime2_metadata(metadata)
```

Arguments

`metadata`	file name, a qiime2 .tsv file containing metadata about your analysis.

Value

A data.frame containing metadata

Examples

```
metadata <- read_qiime2_metadata(strollur_example(
  "sample_metadata.tsv"
))
```

`read_qiime2_taxonomy` *read_qiime2_taxonomy*

Description

Read a qiime2 qza containing taxonomy data

Usage

```
read_qiime2_taxonomy(qza, dir_path = NULL, remove_unpacked_artifacts = TRUE)
```

Arguments

qza	file name, a qiime2 .qza file containing taxonomy data.
dir_path	a string containing the name of directory where the artifacts files should be unpacked. Default = current working directory.
remove_unpacked_artifacts	boolean, When TRUE, the artifact's temporary directories will be removed after processing. Default = TRUE.

Value

A list containing artifact

Examples

```
artifact <- read_qiime2_taxonomy(strollur_example(
  "taxonomy.qza"
))

# access the taxonomy table

artifact$data
```

---

remove_file *remove_file*

Description

Remove file, if it exists

Usage

```
remove_file(filename)
```

Arguments

filename	String containing name of file to remove

report *Get a data.frame containing the given report in a* R*hrefhttps://mothur.org/strollur/reference/strollur.htmlstrollur object*

Description

Get a data.frame containing the report. Reports include FASTA format, sequences reports, sequence_bin_assignments, sequence_taxonomy, bin_taxonomy, bin_representatives, sample_assignments, metadata, references, sequence_scrap, and bin_scrap in a strollur object.

Usage

```
report(data, type = "sequence", bin_type = "otu")
```

Arguments

data	a strollur object
type	string containing the type of report you would like. Options include: "fasta", "sequence", "sequence_bin_assignment", "sequence_taxonomy", "bin_taxonomy", "bin_representative", "sample_assignment", "metadata", "resource_reference", "sequence_scrap", "bin_scrap". If you have added custom reports for alignment, contigs_assembly or chimeras, you can get those as well. Default = "sequence".
bin_type	string containing the bin type you would like a bin_taxonomy report for. Default = "otu".

Value

data.frame

Examples

```
miseq <- miseq_sop_example()

# To get the FASTA data

report(data = miseq, type = "fasta") |> head(n = 5)

# To get a report about the FASTA data

report(data = miseq, type = "sequence") |> head(n = 5)

# To get the sequence bin assignments

report(data = miseq, type = "sequence_bin_assignment", bin_type = "otu") |>
  head(n = 5)

# To get the sample treatment assignments
```

```
report(data = miseq, type = "sample_assignment")

# To get a report about sequence classifications

report(data = miseq, type = "sequence_taxonomy") |> head(n = 10)

# To get a report about bin classifications for 'otu' data

report(data = miseq, type = "bin_taxonomy", bin_type = "otu") |> head(n = 10)

# To get the 'otu' bin representative sequences

report(
  data = miseq, type = "bin_representative",
  bin_type = "otu"
) |> head(n = 5)

# To get a report about the sequences removed during your analysis:

report(data = miseq, type = "sequence_scrap")

# To get a report about the "otu" bins removed during your analysis:

report(data = miseq, type = "bin_scrap", bin_type = "otu")

# To get the metadata associated with your data:

metadata <- report(data = miseq, type = "metadata")

# To get the resource references associated with your data:

references <- report(data = miseq, type = "resource_reference")

# To get our custom report containing the contigs assembly data:

report(data = miseq, type = "contigs_report") |> head(n = 10)
```

`save_dataset` *save_dataset*

Description

The save_dataset function will save the strollur object to file.

Usage

```
save_dataset(data, file)
```

Arguments

data	a strollur object
file	a string containing the file name.

Value

A file containing the ‘strollur‘ object

See Also

[load_dataset()]

Examples

```
data <- read_mothur(
  fasta = strollur_example("final.fasta.gz"),
  count = strollur_example("final.count_table.gz"),
  taxonomy = strollur_example("final.taxonomy.gz"),
  design = strollur_example("mouse.time.design"),
  otu_list = strollur_example("final.opti_mcc.list.gz"),
  dataset_name = "miseq_sop"
)

file_name <- file.path(tempdir(), "miseq_sop.rds")
save_dataset(data, file = file_name)
```

---

sort_dataframe *sort_dataframe*

Description

Sort dataframe

Usage

```
sort_dataframe(data, order, named_col)
```

Arguments

data	the data.frame to be sorted
order	vector containing the order desired

named_col	name of column in data.frame to match order
	# sort results alphabetically
	miseq <- miseq_sop_example()
	sequence_names <- names(miseq)
	fasta <- report(miseq, type = fasta)
	sorted_fasta <- sort_dataframe(fasta, order = sort(sequence_names), named_col = "sequence_names")

Value

sorted data.frame

`strollur` *The 'strollur' object stores the data associated with your amplicon sequence analysis.*

Description

'strollur' is an R6 class that stores nucleotide sequences, abundance, sample and treatment assignments, taxonomic classifications, asv / otu clusters and various reports. It is designed to facilitate data analysis across multiple R packages.

Public fields

`data` Rcpp::XPtr<Dataset> pointer to 'Dataset' c++ class. This allows package developers an easy access point to the underlying C++ code with additional functionality.

`raw` Rcpp::RawVector containing the serialized data of the 'Dataset' c++ class. This allows the load and save functions to work with the class.

`sequence_tree` a tree that relates sequences to each other

`sample_tree` a tree that relates samples to each other

Methods

Public methods:

- `strollur$new()`
- `strollur$print()`
- `strollur$abundance()`
- `strollur$add()`
- `strollur$add_sample_tree()`
- `strollur$add_sequence_tree()`
- `strollur$assign()`
- `strollur$clear()`
- `strollur$count()`
- `strollur$get_bin_types()`
- `strollur$get_sample_tree()`
- `strollur$get_sequence_tree()`
- `strollur$get_version()`
- `strollur$is_equal()`
- `strollur$names()`
- `strollur$report()`
- `strollur$summary()`
- `strollur$clone()`

`strollur$new()`: Create a new strollur dataset

Usage:

```
strollur$new(name = "", dataset = NULL)
```

Arguments:

`name` String, name of dataset (optional)

`dataset` a ‘strollur‘ object.

Returns: A new ‘strollur‘ object.

Examples:

```
# to create an empty strollur object, run the following:

data <- new_dataset("soil")
```

`strollur$print()`: Print summary of ‘strollur‘ object

Usage:

```
strollur$print()
```

Returns: No return value, called for side effects.

Examples:

```
miseq <- load_dataset(strollur_example("miseq_sop.rds"))
miseq
```

`strollur$abundance()`: Get the abundance data for sequences, bins, samples, and treatments.

Usage:

```
strollur$abundance(type = "sequence", bin_type = "otu", by_sample = FALSE)
```

Arguments:

`type` string containing the type of data you want the number of. Options include: "sequence", "bin", "sample" and "treatment". Default = "sequence".

`bin_type` string containing the bin type you would like the abundance data for. Default = "otu".

`by_sample` Boolean. When by_sample is TRUE, the abundance data will be parsed by sample. Default = FALSE.

Returns: data.frame

Examples:

```
miseq <- load_dataset(strollur_example("miseq_sop.rds"))

# To the total abundance for each sequence
miseq$abundance(type = "sequence") |> head(n = 5)

# To the total abundance for each sequence parsed by sample
miseq$abundance(type = "sequence", by_sample = TRUE) |> head(n = 5)

# To the total abundance for each "otu" bin
miseq$abundance(type = "bin", bin_type = "otu") |> head(n = 5)

# To the total abundance for each "otu" bin parsed by sample
```

```
miseq$abundance(type = "bin", bin_type = "otu", by_sample = TRUE) |>
head(n = 5)

# To the total abundance for each "asv" bin
miseq$abundance(type = "bin", bin_type = "asv") |> head(n = 5)

# To the total abundance for each "asv" bin parsed by sample
miseq$abundance(type = "bin", bin_type = "asv", by_sample = TRUE) |>
head(n = 5)

# To the total abundance for each sample
miseq$abundance(type = "sample") |> head(n = 5)

# To the total abundance for each treatment
miseq$abundance(type = "treatment")
```

`strollur$add()`**:** Add sequences, reports, metadata or resource references

Usage:

```
strollur$add(
  table,
  type = "sequence",
  report_type = NULL,
  table_names = list(sequence_name = "sequence_name", sequence = "sequence", comment =
   "comment", reference_vendor = "vendor", reference_name = "name", reference_version =
   "version", reference_usage = "usage", reference_note = "note", reference_method_url =
   "method_url", reference_documentation_url = "documentation_url", reference_parameter
    = "parameter", reference_citation = "citation"),
  reference = NULL,
  verbose = TRUE
)
```

Arguments:

`table` a data.frame containing the data you wish to add.

`type` a string containing the type of data. Options include: 'sequence', 'resource_reference' 'metadata' and 'report'.

`report_type` a string containing the type of report you are adding. Options include: 'metadata' and custom reports.

`table_names` named list used to indicate the names of the columns in the table. By default: table_names <- list(sequence_name = "sequence_name", comment = "comment", sequence = "sequence", reference_name = "name", reference_vendor = "vendor", reference_version = "version", reference_usage = "usage", reference_note = "note", reference_documentation_url = "documentation_url", reference_method_url = "method_url", reference_parameter = "parameter", reference_citation = "citation")

In table_names, 'sequence_name' is a string containing the name of the column in 'table' that contains the sequence names. It is used when you are adding FASTA data. Default column name is 'sequence_name'.

In table_names, 'sequence' is a string containing the name of the column in 'table' that contains the sequence nucleotide strings. It is used when you are adding FASTA data. Default column name is 'sequence'.

In table_names, 'comment' is a string containing the name of the column in 'table' that contains the sequence comments. It is used when you are adding FASTA data. Default column name is 'comment'.
In table_names, 'reference_vendor' is a string containing the name of the column in 'table' that contains the reference vendor names. It is used when ' you are adding reference data. Default column name is 'vendor'.
In table_names, 'reference_name' is a string containing the name of the ' column in 'table' that contains the reference names. It is used when you are ' adding reference data. Default column name is 'name'.
In table_names, 'reference_version' is a string containing the name of the ' column in 'table' that contains the reference versions. Default column name is 'version'.
In table_names, 'reference_usage' is a string containing the name of the column in 'table' that contains the reference usages. Default column name is 'usage'.
In table_names, 'reference_note' is a string containing the name of the column in 'table' that contains the reference notes. Default column name is 'note'.
In table_names, 'reference_method_url' is a string containing the name of the column in 'table' that contains the reference method urls. Default column name is 'method_url'.
In table_names, 'reference_documentation_url' is a string containing the name of the column in 'table' that contains the reference urls. Default column name is 'documentation_url'.
In table_names, 'reference_parameter' is a string containing the name of the column in 'table' that contains the reference parameters. Default column name is 'parameter'.
In table_names, 'reference_citation' is a string containing the name of the column in 'table' that contains the reference citations. Default column name is 'citation'.

`reference` a list created by the function [new_reference]. Optional.

`verbose` boolean indicating whether or not you want progress messages. Default = TRUE.

Returns: Updated 'strollur' object - invisible(self)

Examples:

```
fasta_data <- read_fasta(fasta = strollur_example("final.fasta.gz"))
contigs_report <- readRDS(strollur_example("miseq_contigs_report.rds"))

# Create a new empty `strollur` object named 'example_dataset'
data <- new_dataset(dataset_name = "example_dataset")

data$add(table = fasta_data, type = "sequence")
data$add(
  table = contigs_report, type = "report",
  report_type = "contigs_report", list(sequence_name = "Name")
)

# To add metadata related to your study

metadata <- readRDS(strollur_example("miseq_metadata.rds"))

data$add(table = metadata, type = "metadata")
```

`strollur$add_sample_tree()`: Add phylo tree relating the samples in your dataset

Usage:

```
strollur$add_sample_tree(tree)
```

Arguments:

`tree` a phylo tree object created by ape::read.tree.

Returns: Updated 'strollur' object

Examples:

```
data <- new_dataset("my_dataset")

df <- read_mothur_shared(strollur_example("final.opti_mcc.shared"))
assign(data = data, table = df, type = "bin", bin_type = "otu")

tree <- ape::read.tree(strollur_example(
"final.opti_mcc.jclass.ave.tre"))

data$add_sample_tree(tree)
```

`strollur$add_sequence_tree()`: Add phylo tree relating the sequences in your dataset

Usage:

```
strollur$add_sequence_tree(tree)
```

Arguments:

`tree` a phylo tree object created by ape::read.tree.

Returns: Updated 'strollur' object

Examples:

```
data <- new_dataset("my_dataset")
tree <- ape::read.tree(strollur_example("final.phylip.tre.gz"))
data$add_sequence_tree(tree)
```

`strollur$assign()`: Assign sequence abundances, sequence classifications, bins, bin representative sequences, bin classifications or treatments.

Usage:

```
strollur$assign(
  table,
  type = "bin",
  bin_type = "otu",
  table_names = list(sequence_name = "sequence_name", abundance = "abundance", sample =
   "sample", treatment = "treatment", taxonomy = "taxonomy", bin_name = "bin_name"),
  reference = NULL,
  verbose = TRUE
)
```

Arguments:

`table` a data.frame containing the data you wish to assign

`type` a string containing the type of data. Options include: 'sequence_abundance', 'sequence_taxonomy', 'bin', 'bin_representative', 'bin_taxonomy' and 'treatment'. Default = "bin".

`bin_type` string containing the bin type you would like the number of bins for. Default = "otu".

`table_names` named list used to indicate the names of the columns in the table. By default: table_names <- list(sequence_name = "sequence_name", abundance = "abundance", sample = "sample", treatment = "treatment", taxonomy = "taxonomy", bin_name = "bin_name")
In table_names, 'sequence_name' is a string containing the name of the column in 'table' that contains the sequence names. Default column name is 'sequence_name'.
In table_names, 'abundance' is a string containing the name of the column in 'table' that contains the abundances. Default column name is 'abundance'.
In table_names, 'sample' is a string containing the name of the column in 'table' that contains the samples. Default column name is 'sample'.
In table_names, 'treatment' is a string containing the name of the column in 'table' that contains the treatment names. Default column name is 'treatment'.
In table_names, 'taxonomy' is a string containing the name of the column in 'table' that contains the classifications. Default column name is 'taxonomy'.
In table_names, 'bin_name' is a string containing the name of the column in 'table' that contains the bin names. Default column name is 'bin_name'.

`reference` a list created by the function [new_reference]. Optional.

`verbose` boolean indicating whether or not you want progress messages. Default = TRUE.

Returns: Updated 'strollur' object

Examples:

```
# create a new empty strollur object named 'example_dataset'

data <- new_dataset(dataset_name = "example_dataset")

# Assign sequence abundances

abundance_by_sample <- read_mothur_count(strollur_example(
  "final.count_table.gz"
))

data$assign(table = abundance_by_sample, type = "sequence_abundance")

# Assign sequence classifications

sequence_classifications <- read_mothur_taxonomy(strollur_example(
  "final.taxonomy.gz"
))

data$assign(table = sequence_classifications, type = "sequence_taxonomy")

# Assigning bins

# read mothur's otu list file into data.frame
otu_data <- read_mothur_list(list = strollur_example(
  "final.opti_mcc.list.gz"
))

# read mothur's asv list file into data.frame
asv_data <- read_mothur_list(list = strollur_example(
```

```
  "final.asv.list.gz"
))

# read mothur's phylotype list file into data.frame
phylo_data <- read_mothur_list(list = strollur_example(
  "final.tx.list.gz"
))

# read otu bin representative sequences into a data.frame
bin_reps <- readRDS(strollur_example(
                       "miseq_representative_sequences.rds"))

# assign 'otu' bins using sequence names
data$assign(table = otu_data, bin_type = "otu")

# assign 'asv' bins using sequence names
data$assign(table = asv_data, bin_type = "asv")

# assign 'phylotype' bins using sequence names
data$assign(table = phylo_data, bin_type = "phylotype")

# assign 'otu' bin representative sequences
data$assign(table = bin_reps, type = "bin_representative")

# To assign abundance only bins

# create a new empty strollur object named 'example_dataset'
data <- new_dataset(dataset_name = "example_dataset")

# read mothur's shared file
otu_data <- read_mothur_shared(strollur_example("final.opti_mcc.shared"))

# assign abundance only otus parsed by sample
data$assign(table = otu_data, bin_type = "otu")

# Assigning bin classifications

# read bin taxonomies
otu_data <- read_mothur_cons_taxonomy(strollur_example(
  "final.cons.taxonomy"
))

# assign otu consensus taxonomies
data$assign(
  table = otu_data,
  type = "bin_taxonomy", bin_type = "otu"
)
```

```
# Assign treatments

sample_assignments <- readRDS(
   strollur_example("miseq_sample_design.rds"))

data$assign(table = sample_assignments, type = "treatment")
```

`strollur$clear()`: Clear data from datasest

Usage:

```
strollur$clear()
```

Returns: Updated ‘strollur‘ object

Examples:

```
miseq <- load_dataset(strollur_example("miseq_sop.rds"))
miseq
miseq$clear()
miseq
```

`strollur$count()`: Find the number of sequences, samples, treatments or bins of a given type

Usage:

```
strollur$count(
  type = "sequence",
  bin_type = "otu",
  samples = NULL,
  distinct = FALSE
)
```

Arguments:

`type` string containing the type of data you want the number of. Options include: "sequence", "sample", "treatment", "bin", and "resource_reference". Default = "sequence".

`bin_type` string containing the bin type you would like the number of bins for. Default = "otu".

`samples` vector of strings. samples is only used when ’type’ = "sequence" or ’type’ = "bin" . samples should contain the names of the samples you want the count for. Default = NULL.

`distinct` Boolean. distinct is used when ’type’ = "sequence" or ’type’ = "bin". When ’type’ = "sequence" and distinct is TRUE the number of unique sequences is returned. When ’type’ = "sequence" and distinct is FALSE the total number of sequences is returned. This can also be combined with samples to find the number of unique sequences found ONLY in a given set of samples, or to find the number of unique sequences in given set of samples that may also be present in other samples. When ’type’ = "bin", you can set distinct = TRUE to return the number of bins that ONLY contain sequences from the given samples. When distinct is FALSE the count returned contains bins with sequences from a given samples, but those bins may also contain other samples. Default = FALSE.

Returns: double

Examples:

```
miseq <- load_dataset(strollur_example("miseq_sop.rds"))

# To get the total number of sequences
```

```
miseq$count(type = "sequence")

# To get number of unique sequences
miseq$count(type = "sequence", distinct = TRUE)

# To get number of unique sequences from samples 'F3D0' and 'F3D1'
# Note these sequences will be present in both samples but may be
# be present in other samples as well
miseq$count(type = "sequence", samples = c("F3D0", "F3D1"))

# To get number of unique sequences exclusive to samples 'F3D0' and
# 'F3D1'. Note sequences are present in both samples and NOT present in
# any other samples.

miseq$count(type = "sequence",
            samples = c("F3D0", "F3D1"), distinct = TRUE )

# To get the number of samples in the dataset
miseq$count(type = "sample")

# To get the number of treatments in the dataset
miseq$count(type = "treatment")

# To get the number of "otu" bins in the dataset
miseq$count(type = "bin", bin_type = "otu")

# To get the number of "asv" bins in the dataset
miseq$count(type = "bin", bin_type = "asv")

# To get the number of "phylotype" bins in the dataset
miseq$count(type = "bin", bin_type = "phylotype")

# To get number of "otu" bins from samples 'F3D0' and 'F3D1'
# Note these bins will have sequences from both samples but there may be
# other samples present as well
miseq$count(
  type = "bin", bin_type = "otu", samples = c("F3D0", "F3D1")
)

# To get number of "otu" bins unique to samples 'F3D0' and 'F3D1'
# Note these bins will have sequences from both samples and NO other
# samples will be present in the bins.

miseq$count(
  type = "bin", bin_type = "otu",
  samples = c("F3D0", "F3D1"), distinct = TRUE
)
```

`strollur$get_bin_types()`**:** Get bin table types

Usage:

```
strollur$get_bin_types()
```

Returns: vector of strings

Examples:

```
data <- miseq_sop_example()
data$get_bin_types()
```

`strollur$get_sample_tree()`: Get phylo tree relating the samples in your dataset.

Usage:

```
strollur$get_sample_tree()
```

Returns: ape::tree

Examples:

```
 tree <- ape::read.tree(strollur_example(
  "final.opti_mcc.jclass.ave.tre"))

 df <- read_mothur_shared(strollur_example("final.opti_mcc.shared"))

 data <- new_dataset("my_dataset")

 # assign abundance 'otu' bins
 data$assign(table = df, type = "bin", bin_type = "otu")

 data$add_sample_tree(tree)
 data$get_sample_tree()
```

`strollur$get_sequence_tree()`: Get phylo tree relating the sequences in your strollur object.

Usage:

```
strollur$get_sequence_tree()
```

Returns: ape::tree

Examples:

```
 data <- new_dataset("my_dataset")
 tree <- ape::read.tree(strollur_example("final.phylip.tre.gz"))
 data$add_sequence_tree(tree)
 data$get_sequence_tree()
```

`strollur$get_version()`: Get the version of the strollur object.

Usage:

```
strollur$get_version()
```

Returns: a logical

Examples:

```
data <- new_dataset("test")

data$get_version()
```

`strollur$is_equal()`: Determine if two strollur objects are equal.

Usage:

```
strollur$is_equal(data)
```

Arguments:

`data`, a strollur object

Returns: a logical

Examples:

```
miseq <- load_dataset(strollur_example("miseq_sop.rds"))

data <- copy_dataset(miseq)

miseq$is_equal(data)
```

`strollur$names()`: Get the names of a given type of data

Usage:

```
strollur$names(
  type = "sequence",
  bin_type = "otu",
  samples = NULL,
  distinct = FALSE
)
```

Arguments:

`type` string containing the type of data you would like. Options include: "dataset", "sequence", "bin", "sample", "treatment", "report". Default = "sequence".

`bin_type` string containing the bin type you would like the names for. Default = "otu".

`samples` vector of strings. samples is only used when 'type' = "sequence" or 'type' = "bin" . samples should contain the names of the samples you want names for. Default = NULL.

`distinct` Boolean. distinct is used when 'type' = "sequence" or 'type' = "bin" and the samples parameter is used. The distinct parameter allows you to get the names that present given set of samples. When distinct is TRUE, the names function will return the names that ONLY contain data from the given samples. When distinct is FALSE the data returned contains data from a given samples, but may ALSO contain data from other samples. Default = FALSE.

Returns: vector of strings, containing the names requested

Examples:

```
miseq <- load_dataset(strollur_example("miseq_sop.rds"))

# To get the name of the dataset
miseq$names(type = "dataset")

# To get the names of the sequences
miseq$names(type = "sequence")

# To get the names of the sequences present sample 'F3D0'
```

```
miseq$names(type = "sequence", samples = c("F3D0"))

#' # To get the names of the sequences unique to sample 'F3D0'
miseq$names(type = "sequence", samples = c("F3D0"), distinct = TRUE)

# To get the names of the samples
miseq$names(type = "sample")

# To get the names of the treatments
miseq$names(type = "treatment")

# To get the names of the bins
miseq$names(type = "bin")

# To get the names of the bins that are unique to 'F3D0'
miseq$names(type = "bin", samples = c("F3D0"), distinct = TRUE)

# To get the names of the bins that include sequences from 'F3D0'
miseq$names(type = "bin", samples = c("F3D0"), distinct = FALSE)

# To get the names of the reports
miseq$names(type = "report")
```

`strollur$report()`**:** Get a data.frame containing the given report

Usage:

```
strollur$report(type = "sequence", bin_type = "otu")
```

Arguments:

`type` string containing the type of report you would like. Options include: "fasta", "sequence", "sequence_bin_assignment", "sequence_taxonomy", "bin_taxonomy", "bin_representative", "sample_assignment", "metadata", "resource_reference", "sequence_scrap", "bin_scrap". If you have added custom reports for alignment, contigs_assembly or chimeras, you can get those as well. Default = "sequence".

`bin_type` string containing the bin type you would like a bin_taxonomy report for. Default = "otu".

Returns: data.frame

Examples:

```
miseq <- load_dataset(strollur_example("miseq_sop.rds"))

# To get the FASTA data

miseq$report(type = "fasta") |> head(n = 5)

# To get a report about the FASTA data

miseq$report(type = "sequence") |> head(n = 5)

# To get the sequence bin assignments
```

```
miseq$report(type = "sequence_bin_assignment", bin_type = "otu") |>
head(n = 5)

# To get the sample treatment assignments

miseq$report(type = "sample_assignment") |> head(n = 5)

# To get a report about sequence classifications

miseq$report(type = "sequence_taxonomy") |> head(n = 5)

# To get a report about bin classifications for 'otu' data

miseq$report(type = "bin_taxonomy", bin_type = "otu") |> head(n = 5)

# To get the 'otu' bin representative sequences

miseq$report(type = "bin_representative", bin_type = "otu") |>
head(n = 5)

# To get a report about the sequences removed during your analysis:

miseq$report(type = "sequence_scrap")

# To get a report about the "otu" bins removed during your analysis:

miseq$report(type = "bin_scrap", bin_type = "otu")

# To get the metadata associated with your data:

metadata <- miseq$report(type = "metadata") |> head(n = 5)

# To get the resource references associated with your data:

references <- miseq$report(type = "resource_reference")

# To get our custom report containing the contigs assembly data:

miseq$report(type = "contigs_report") |> head(n = 5)
```

`strollur$summary()`**:** Summarize the sequences data, custom reports, and scrapped data

Usage:

```
strollur$summary(type = "sequence", report_type = NULL, verbose = TRUE)
```

Arguments:

`type` string containing the type of data you want the number of. Options include: "sequence", "report" and "scrap". Default = "sequence".

`report_type` string containing the report type you would summarized. For example, the miseq_sop_example includes contigs assembly data and can be accessed with report_type = "contigs_report". Default = NULL.

`verbose` boolean indicating whether or not you want progress messages. Default = TRUE.

Returns: data.frame Get summary of the sequence reports

Examples:

```
miseq <- load_dataset(strollur_example("miseq_sop.rds"))

# To get the summary of your FASTA data
miseq$summary(type = "sequence")

# summarize contigs_report
miseq$summary(type = "report", report_type = "contigs_report")

# remove sample 'F3D0' to produce a scrap report
xdev_remove_samples(data = miseq, samples = c("F3D0"))

# summarize scrapped data -
# sequences and bins scrapped by removing the sample "F3D0"
miseq$summary(type = "scrap")
```

`strollur$clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
strollur$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sarah Westcott, <swestcot@umich.edu>

Examples

```
## ------------------------------------------------
## Method `strollur$new()`
## ------------------------------------------------


# to create an empty strollur object, run the following:

data <- new_dataset("soil")


## ------------------------------------------------
## Method `strollur$print()`
## ------------------------------------------------

miseq <- load_dataset(strollur_example("miseq_sop.rds"))
```

```
miseq


## ------------------------------------------------
## Method `strollur$abundance()`
## ------------------------------------------------


miseq <- load_dataset(strollur_example("miseq_sop.rds"))

# To the total abundance for each sequence
miseq$abundance(type = "sequence") |> head(n = 5)

# To the total abundance for each sequence parsed by sample
miseq$abundance(type = "sequence", by_sample = TRUE) |> head(n = 5)

# To the total abundance for each "otu" bin
miseq$abundance(type = "bin", bin_type = "otu") |> head(n = 5)

# To the total abundance for each "otu" bin parsed by sample
miseq$abundance(type = "bin", bin_type = "otu", by_sample = TRUE) |>
head(n = 5)

# To the total abundance for each "asv" bin
miseq$abundance(type = "bin", bin_type = "asv") |> head(n = 5)

# To the total abundance for each "asv" bin parsed by sample
miseq$abundance(type = "bin", bin_type = "asv", by_sample = TRUE) |>
head(n = 5)

# To the total abundance for each sample
miseq$abundance(type = "sample") |> head(n = 5)

# To the total abundance for each treatment
miseq$abundance(type = "treatment")


## ------------------------------------------------
## Method `strollur$add()`
## ------------------------------------------------


fasta_data <- read_fasta(fasta = strollur_example("final.fasta.gz"))
contigs_report <- readRDS(strollur_example("miseq_contigs_report.rds"))

# Create a new empty `strollur` object named 'example_dataset'
data <- new_dataset(dataset_name = "example_dataset")

data$add(table = fasta_data, type = "sequence")
data$add(
  table = contigs_report, type = "report",
  report_type = "contigs_report", list(sequence_name = "Name")
)
```

```
# To add metadata related to your study

metadata <- readRDS(strollur_example("miseq_metadata.rds"))

data$add(table = metadata, type = "metadata")


## ------------------------------------------------
## Method `strollur$add_sample_tree()`
## ------------------------------------------------


 data <- new_dataset("my_dataset")

 df <- read_mothur_shared(strollur_example("final.opti_mcc.shared"))
 assign(data = data, table = df, type = "bin", bin_type = "otu")

 tree <- ape::read.tree(strollur_example(
 "final.opti_mcc.jclass.ave.tre"))

 data$add_sample_tree(tree)

## ------------------------------------------------
## Method `strollur$add_sequence_tree()`
## ------------------------------------------------


 data <- new_dataset("my_dataset")
 tree <- ape::read.tree(strollur_example("final.phylip.tre.gz"))
 data$add_sequence_tree(tree)


## ------------------------------------------------
## Method `strollur$assign()`
## ------------------------------------------------


# create a new empty strollur object named 'example_dataset'

data <- new_dataset(dataset_name = "example_dataset")

# Assign sequence abundances

abundance_by_sample <- read_mothur_count(strollur_example(
  "final.count_table.gz"
))

data$assign(table = abundance_by_sample, type = "sequence_abundance")

# Assign sequence classifications

sequence_classifications <- read_mothur_taxonomy(strollur_example(
```

```
  "final.taxonomy.gz"
))

data$assign(table = sequence_classifications, type = "sequence_taxonomy")

# Assigning bins

# read mothur's otu list file into data.frame
otu_data <- read_mothur_list(list = strollur_example(
  "final.opti_mcc.list.gz"
))

# read mothur's asv list file into data.frame
asv_data <- read_mothur_list(list = strollur_example(
  "final.asv.list.gz"
))

# read mothur's phylotype list file into data.frame
phylo_data <- read_mothur_list(list = strollur_example(
  "final.tx.list.gz"
))

# read otu bin representative sequences into a data.frame
bin_reps <- readRDS(strollur_example(
                        "miseq_representative_sequences.rds"))

# assign 'otu' bins using sequence names
data$assign(table = otu_data, bin_type = "otu")

# assign 'asv' bins using sequence names
data$assign(table = asv_data, bin_type = "asv")

# assign 'phylotype' bins using sequence names
data$assign(table = phylo_data, bin_type = "phylotype")

# assign 'otu' bin representative sequences
data$assign(table = bin_reps, type = "bin_representative")

# To assign abundance only bins

# create a new empty strollur object named 'example_dataset'
data <- new_dataset(dataset_name = "example_dataset")

# read mothur's shared file
otu_data <- read_mothur_shared(strollur_example("final.opti_mcc.shared"))

# assign abundance only otus parsed by sample
data$assign(table = otu_data, bin_type = "otu")

# Assigning bin classifications

# read bin taxonomies
otu_data <- read_mothur_cons_taxonomy(strollur_example(
```

```
  "final.cons.taxonomy"
))

# assign otu consensus taxonomies
data$assign(
  table = otu_data,
  type = "bin_taxonomy", bin_type = "otu"
)

# Assign treatments

sample_assignments <- readRDS(
   strollur_example("miseq_sample_design.rds"))

data$assign(table = sample_assignments, type = "treatment")


## ------------------------------------------------
## Method `strollur$clear()`
## ------------------------------------------------

miseq <- load_dataset(strollur_example("miseq_sop.rds"))
miseq
miseq$clear()
miseq

## ------------------------------------------------
## Method `strollur$count()`
## ------------------------------------------------


miseq <- load_dataset(strollur_example("miseq_sop.rds"))

# To get the total number of sequences
miseq$count(type = "sequence")

# To get number of unique sequences
miseq$count(type = "sequence", distinct = TRUE)

# To get number of unique sequences from samples 'F3D0' and 'F3D1'
# Note these sequences will be present in both samples but may be
# be present in other samples as well
miseq$count(type = "sequence", samples = c("F3D0", "F3D1"))

# To get number of unique sequences exclusive to samples 'F3D0' and
# 'F3D1'. Note sequences are present in both samples and NOT present in
# any other samples.

miseq$count(type = "sequence",
            samples = c("F3D0", "F3D1"), distinct = TRUE )

# To get the number of samples in the dataset
miseq$count(type = "sample")
```

```
# To get the number of treatments in the dataset
miseq$count(type = "treatment")

# To get the number of "otu" bins in the dataset
miseq$count(type = "bin", bin_type = "otu")

# To get the number of "asv" bins in the dataset
miseq$count(type = "bin", bin_type = "asv")

# To get the number of "phylotype" bins in the dataset
miseq$count(type = "bin", bin_type = "phylotype")

# To get number of "otu" bins from samples 'F3D0' and 'F3D1'
# Note these bins will have sequences from both samples but there may be
# other samples present as well
miseq$count(
  type = "bin", bin_type = "otu", samples = c("F3D0", "F3D1")
)

# To get number of "otu" bins unique to samples 'F3D0' and 'F3D1'
# Note these bins will have sequences from both samples and NO other
# samples will be present in the bins.

miseq$count(
  type = "bin", bin_type = "otu",
  samples = c("F3D0", "F3D1"), distinct = TRUE
)


## ------------------------------------------------
## Method `strollur$get_bin_types()`
## ------------------------------------------------


data <- miseq_sop_example()
data$get_bin_types()


## ------------------------------------------------
## Method `strollur$get_sample_tree()`
## ------------------------------------------------


 tree <- ape::read.tree(strollur_example(
  "final.opti_mcc.jclass.ave.tre"))

 df <- read_mothur_shared(strollur_example("final.opti_mcc.shared"))

 data <- new_dataset("my_dataset")

 # assign abundance 'otu' bins
 data$assign(table = df, type = "bin", bin_type = "otu")
```

```
 data$add_sample_tree(tree)
 data$get_sample_tree()


## ------------------------------------------------
## Method `strollur$get_sequence_tree()`
## ------------------------------------------------


 data <- new_dataset("my_dataset")
 tree <- ape::read.tree(strollur_example("final.phylip.tre.gz"))
 data$add_sequence_tree(tree)
 data$get_sequence_tree()


## ------------------------------------------------
## Method `strollur$get_version()`
## ------------------------------------------------


data <- new_dataset("test")

data$get_version()


## ------------------------------------------------
## Method `strollur$is_equal()`
## ------------------------------------------------


miseq <- load_dataset(strollur_example("miseq_sop.rds"))

data <- copy_dataset(miseq)

miseq$is_equal(data)


## ------------------------------------------------
## Method `strollur$names()`
## ------------------------------------------------


miseq <- load_dataset(strollur_example("miseq_sop.rds"))

# To get the name of the dataset
miseq$names(type = "dataset")

# To get the names of the sequences
miseq$names(type = "sequence")

# To get the names of the sequences present sample 'F3D0'
miseq$names(type = "sequence", samples = c("F3D0"))
```

```
#' # To get the names of the sequences unique to sample 'F3D0'
miseq$names(type = "sequence", samples = c("F3D0"), distinct = TRUE)

# To get the names of the samples
miseq$names(type = "sample")

# To get the names of the treatments
miseq$names(type = "treatment")

# To get the names of the bins
miseq$names(type = "bin")

# To get the names of the bins that are unique to 'F3D0'
miseq$names(type = "bin", samples = c("F3D0"), distinct = TRUE)

# To get the names of the bins that include sequences from 'F3D0'
miseq$names(type = "bin", samples = c("F3D0"), distinct = FALSE)

# To get the names of the reports
miseq$names(type = "report")


## ------------------------------------------------
## Method `strollur$report()`
## ------------------------------------------------


miseq <- load_dataset(strollur_example("miseq_sop.rds"))

# To get the FASTA data

miseq$report(type = "fasta") |> head(n = 5)

# To get a report about the FASTA data

miseq$report(type = "sequence") |> head(n = 5)

# To get the sequence bin assignments

miseq$report(type = "sequence_bin_assignment", bin_type = "otu") |>
head(n = 5)

# To get the sample treatment assignments

miseq$report(type = "sample_assignment") |> head(n = 5)

# To get a report about sequence classifications

miseq$report(type = "sequence_taxonomy") |> head(n = 5)

# To get a report about bin classifications for 'otu' data
```

```
miseq$report(type = "bin_taxonomy", bin_type = "otu") |> head(n = 5)

# To get the 'otu' bin representative sequences

miseq$report(type = "bin_representative", bin_type = "otu") |>
head(n = 5)

# To get a report about the sequences removed during your analysis:

miseq$report(type = "sequence_scrap")

# To get a report about the "otu" bins removed during your analysis:

miseq$report(type = "bin_scrap", bin_type = "otu")

# To get the metadata associated with your data:

metadata <- miseq$report(type = "metadata") |> head(n = 5)

# To get the resource references associated with your data:

references <- miseq$report(type = "resource_reference")

# To get our custom report containing the contigs assembly data:

miseq$report(type = "contigs_report") |> head(n = 5)


## ------------------------------------------------
## Method `strollur$summary()`
## ------------------------------------------------


miseq <- load_dataset(strollur_example("miseq_sop.rds"))

# To get the summary of your FASTA data
miseq$summary(type = "sequence")

# summarize contigs_report
miseq$summary(type = "report", report_type = "contigs_report")

# remove sample 'F3D0' to produce a scrap report
xdev_remove_samples(data = miseq, samples = c("F3D0"))

# summarize scrapped data -
# sequences and bins scrapped by removing the sample "F3D0"
miseq$summary(type = "scrap")
```

---

strollur_example *strollur_example*

Description

strollur comes bundled with some example files in its ‘inst/extdata‘ directory. This function make them easy to access them.

Usage

```
strollur_example(file = NULL)
```

Arguments

file	Name of file.

Details

Get path to strollur example

Value

string, Full path to example files

Examples

```
strollur_example()
strollur_example("final.fasta.gz")
```

---

summary — *Summarize the sequences data, custom reports, and scrapped data in a* R*hrefhttps://mothur.org/strollur/reference/strollur.htmlstrollur object*

Description

Summarize the sequences data, custom reports, and scrapped data in a strollur object

Usage

```
summary(data, type = "sequence", report_type = NULL, verbose = TRUE)
```

Arguments

data	a strollur object
type	string containing the type of data you want the number of. Options include: "sequence", "report" and "scrap". Default = "sequence".
report_type	string containing the report type you would summarized. For example, the miseq_sop_example includes contigs assembly data and can be accessed with report_type = "contigs_report". Default = NULL.
verbose	boolean indicating whether or not you want progress messages. Default = TRUE.

Value

data.frame

Examples

```
miseq <- miseq_sop_example()

# To get the summary of your FASTA data
summary(data = miseq, type = "sequence")

# summarize contigs_report
summary(data = miseq, type = "report", report_type = "contigs_report")

# remove sample 'F3D0' to produce a scrap report
xdev_remove_samples(data = miseq, samples = c("F3D0"))

# summarize FASTA data after removal of sample F3D0
summary(data = miseq, type = "sequence")

# summarize scrapped data -
# sequences and bins scrapped by removing the sample "F3D0"
summary(data = miseq, type = "scrap")
```

---

unpack_qiime2_artifact

unpack_qiime2_artifact

Description

The unpack_qiime2_artifact function reads .qza files created by qiime2, and returns the artifact.

Usage

```
unpack_qiime2_artifact(qza, dir_path = NULL)
```

Arguments

qza	filename, a .qza file containing artifact
dir_path	a string containing the name of directory where the artifacts files should be written. Default = current working directory.

Value

A unpacked qza artifact

Examples

```
# Using the example files from moving-pictures

artifact <- unpack_qiime2_artifact(
  qza = strollur_example("table.qza"),
  dir_path = tempdir()
)
```

`write_fasta` *write_fasta*

Description

Write a FASTA formatted sequence file

Usage

```
write_fasta(data, filename = NULL, degap = FALSE)
```

Arguments

`data`	A ‘strollur‘ object
`filename`	a string containing the name of the output file. Default = ’dataset_name’.fasta
`degap`	a logical. Default = FALSE. When degap = ‘TRUE‘, all gap characters will be removed from the sequences.

Value

name of FASTA file

Examples

```
miseq <- miseq_sop_example()
write_fasta(miseq, tempfile())
```

write_mothur *write_mothur*

Description

The write_mothur function will write various file types for use with mothur.

Usage

```
write_mothur(data, dir_path = NULL, compress = TRUE, tags = NULL)
```

Arguments

data	A 'strollur' object
dir_path	a string containing the name of directory where the files should be written. Default = current working directory.
compress	boolean, Default = TRUE.
tags	a vector of strings containing the items you wish to write Options are 'sequence_data', 'bin_data', 'metadata', 'resource_reference', 'sequence_tree', 'sample_tree' and 'report'. By default, everything is written to files.

Value

a vector of file names

References

Schloss,P.D., Westcott,S.L., Ryabin,T., Hall,J.R., Hartmann,M., Hollister,E.B., Lesniewski,R.A., Oakley,B.B., Parks,D.H., Robinson,C.J., Sahl,J.W., Stres,B., Thallinger,G.G., Van Horn,D.J. and Weber,C.F. (2009), Introducing mothur: Open-source, platform-independent, community-supported software for describing and comparing microbial communities. Applied and Environmental Microbiology 75:7537-7541. <doi:10.1128/AEM.01541-09>

Examples

```
miseq <- miseq_sop_example()
files <- write_mothur(miseq, tempdir(), compress = FALSE)
```

write_mothur_cons_taxonomy *write_mothur_cons_taxonomy*

Description

Write a mothur formatted cons_taxonomy file

Usage

```
write_mothur_cons_taxonomy(data, file_root = NULL)
```

Arguments

data	A 'strollur' object
file_root	a string containing the root name of the output file. Default = 'dataset_name'. Resulting in output files 'dataset_name'.bin_type'.cons.taxonomy.

Value

vector containing the names of the files created

Examples

```
miseq <- miseq_sop_example()
write_mothur_cons_taxonomy(miseq, tempfile())
```

write_mothur_count *write_mothur_count*

Description

Write a mothur formatted count file

Usage

```
write_mothur_count(data, filename = NULL)
```

Arguments

data	A 'strollur' object
filename	a string containing the name of the output file. Default = 'dataset_name'.count_table

Value

name of count file

Examples

```
miseq <- miseq_sop_example()
write_mothur_count(miseq, tempfile())
```

`write_mothur_design` *write_mothur_design*

Description

Write a mothur formatted design file

Usage

```
write_mothur_design(data, filename = NULL)
```

Arguments

`data`	A 'strollur' object
`filename`	a string containing the name of the output file. Default = 'dataset_name'.design

Value

name of design file

Examples

```
miseq <- miseq_sop_example()
write_mothur_design(miseq, tempfile())
```

write_mothur_list *write_mothur_list*

Description

Write mothur formatted list files

Usage

```
write_mothur_list(data, file_root = NULL)
```

Arguments

data	A ‘strollur‘ object
file_root	a string containing the root name of the output file. Default = ’dataset_name’. Resulting in output files ’dataset_name’.bin_type’.list.

Value

vector containing the names of the files created

Examples

```
miseq <- miseq_sop_example()
write_mothur_list(miseq, tempfile())
```

write_mothur_rabund *write_mothur_rabund*

Description

Write mothur formatted rabund files

Usage

```
write_mothur_rabund(data, file_root = NULL)
```

Arguments

data	A ‘strollur‘ object
file_root	a string containing the root name of the output file. Default = ’dataset_name’. Resulting in output files ’dataset_name’.bin_type’.rabund.

Value

vector containing the names of the files created

Examples

```
miseq <- miseq_sop_example()
write_mothur_rabund(miseq, tempfile())
```

`write_mothur_shared` *write_mothur_shared*

Description

Write mothur formatted shared files

Usage

```
write_mothur_shared(data, file_root = NULL)
```

Arguments

`data`	A ‘strollur‘ object
`file_root`	a string containing the root name of the output file. Default = ’dataset_name’. Resulting in output files ’dataset_name’.bin_type’.shared.

Value

vector containing the names of the files created

Examples

```
miseq <- miseq_sop_example()
write_mothur_shared(miseq, tempfile())
```

`write_phyloseq` *write_phyloseq*

Description

The ‘write_phyloseq()‘ function will take any strollur object and return it as a "phyloseq" object.

Usage

```
write_phyloseq(data)
```

Arguments

`data`	the strollur object you created using one of the many read functions in this package.

Value

returns a "phyloseq" object.

References

McMurdie,P.J. and Holmes,S. (2013), phyloseq: An R Package for Reproducible Interactive Analysis and Graphics of Microbiome Census Data. PLoS ONE 8:e61217. <doi:10.1371/journal.pone.0061217>

Examples

```
miseq <- miseq_sop_example()
if (requireNamespace("phyloseq", quietly = TRUE)) {
  phylo_obj <- write_phyloseq(miseq)
} else {
  message(paste(
    "To use this functionality you have to install the",
    "phyloseq package."
  ))
}
```

`write_taxonomy` *write_taxonomy*

Description

Write a 2 column taxonomy file

Usage

```
write_taxonomy(data, filename = NULL)
```

Arguments

`data`	A ‘strollur‘ object
`filename`	a string containing the name of the output file. Default = ’dataset_name’.taxonomy

Value

name of taxonomy file

Examples

```
miseq <- miseq_sop_example()
write_taxonomy(miseq, tempfile())
```

xdev_abundance *Get a data.frame containing the requested abundance data*

Description

Get a table containing the requested abundance data in a strollur object

Usage

```
xdev_abundance(data, type = "sequence", bin_type = "otu", by_sample = FALSE)
```

Arguments

data	a strollur object
type	string containing the type of data you want the number of. Options include: "sequence", "bin". Default = "sequence".
bin_type	string containing the bin type you would like the number of bins for. Default = "otu".
by_sample	Boolean. When by_sample is TRUE, the abundance data will be parsed by sample. Default = FALSE.

Value

data.frame

Examples

```
miseq <- miseq_sop_example()

# To the total abundance for each sequence
xdev_abundance(data = miseq, type = "sequence")

# To the total abundance for each sequence parsed by sample
xdev_abundance(data = miseq, type = "sequence", by_sample = TRUE)

# To the total abundance for each "otu" bin
xdev_abundance(data = miseq, type = "bin", bin_type = "otu")

# To the total abundance for each "otu" bin parsed by sample
xdev_abundance(data = miseq, type = "bin", bin_type = "otu", by_sample = TRUE)

# To the total abundance for each "asv" bin
xdev_abundance(data = miseq, type = "bin", bin_type = "asv")

# To the total abundance for each "asv" bin parsed by sample
xdev_abundance(data = miseq, type = "bin", bin_type = "asv", by_sample = TRUE)

# To the total abundance of each sample
```

```
xdev_abundance(data = miseq, type = "sample")

# To the total abundance of each treatment
xdev_abundance(data = miseq, type = "treatment")
```

`xdev_add_references` *Add resource references*

Description

Add resource references to a strollur object

Usage

```
xdev_add_references(
  data,
  table,
  name = "name",
  vendor = "vendor",
  version = "version",
  usage = "usage",
  note = "note",
  method_url = "method_url",
  documentation_url = "documentation_url",
  parameter = "parameter",
  citation = "citation",
  verbose = TRUE
)
```

Arguments

`data`	a strollur object
`table`	a data.frame containing reference_names, reference_versions (optional), reference_usages (optional), reference_parameters (optional), reference_methods (optional), and reference_urls (optional).
`name`	a string containing the name of the column in 'table' that contains the reference names. Default column name is 'name'.
`vendor`	a string containing the name of the column in 'table' that contains the reference vendors. Default column name is 'vendor'.
`version`	a string containing the name of the column in 'table' that contains the reference versions. Default column name is 'version'.
`usage`	a string containing the name of the column in 'table' that contains the reference usages. Default column name is 'usage'.
`note`	a string containing the name of the column in 'table' that contains the reference notes. Default column name is 'note'.

method_url	a string containing the name of the column in 'table' that contains the reference methods. Default column name is 'method_url'.
documentation_url	a string containing the name of the column in 'table' that contains the reference documentation urls. Default column name is 'documentation_url'.
parameter	a string containing the name of the column in 'table' that contains the reference parameters. Default column name is 'parameter'.
citation	a string containing the name of the column in 'table' that contains the reference citations. Default column name is 'citation'.
verbose	a boolean whether or not you want progress messages. Default = TRUE.

Value

an updated strollur object

Examples

```
data <- new_dataset("just for fun")
reference_table <- readr::read_csv(strollur_example("references.csv"),
                            col_names = TRUE, show_col_types = FALSE)
xdev_add_references(data, reference_table)
```

---

xdev_add_report *Add a report to a* R*hrefhttps://mothur.org/strollur/reference/strollur.htmlstrollur object*

Description

Add a report to a strollur object

Usage

```
xdev_add_report(
  data,
  table,
  type = "metadata",
  sequence_name = "sequence_name",
  verbose = TRUE
)
```

Arguments

data	a strollur object
table	a data.frame containing your report.
type	a string containing the type of report. Options include: "metadata" and custom report tags. Default = "metadata".

sequence_name	a string containing the name of the column in 'table' that contains the sequence names. This is used for custom reports, metadata does not require a sequence_name column. Default column name is 'sequence_names'.
verbose	a boolean whether or not you want progress messages. Default = TRUE.

Value

an updated strollur object

Examples

```
# To add a custom report including your contigs assembly data

data <- new_dataset("just for fun")
contigs_report <- readRDS(strollur_example("miseq_contigs_report.rds"))

xdev_add_report(data, contigs_report, "contigs_report", "Name")

# To add metadata related to your study

metadata <- readRDS(strollur_example("miseq_metadata.rds"))

xdev_add_report(data, metadata, "metadata")
```

---

xdev_add_sequences *xdev_add_sequences*

Description

Add sequence data to a strollur object

Usage

```
xdev_add_sequences(
  data,
  table,
  reference = NULL,
  sequence_name = "sequence_name",
  sequence = "sequence",
  comment = "comment",
  verbose = TRUE
)
```

Arguments

data	a strollur object
table	a data.frame containing names, sequences(optional) and comments(optional).
reference	a list created by the function [new_reference]. Optional.
sequence_name	a string containing the name of the column in 'table' that contains the sequence names. Default column name is 'sequence_name'.
sequence	a string containing the name of the column in 'table' that contains the sequence nucleotide strings. Default column name is 'sequence'.
comment	a string containing the name of the column in 'table' that contains the sequence comments. Default column name is 'comment'.
verbose	a boolean whether or not you want progress messages. Default = TRUE.

Value

an updated strollur object

Examples

```
data <- new_dataset("miseq_sop")
fasta_data <- read_fasta(strollur_example("final.fasta.gz"))
xdev_add_sequences(data, fasta_data)

# With the additional parameters to add information about the reference

data <- new_dataset("miseq_sop")
fasta_data <- read_fasta(strollur_example("final.fasta.gz"))

xdev_add_sequences(data, fasta_data,
              new_reference("silva.bacteria.fasta",
              "1.38.1",
              "alignment by mothur2 v1.0 using default options",
              "https://mothur.org/wiki/silva_reference_files/"))

# You can also add references using the 'add_references' function.
```

---

xdev_assign_bins *xdev_assign_bins*

Description

Add bin assignments to a strollur object

Usage

```
xdev_assign_bins(
  data,
  table,
  bin_type = "otu",
  reference = NULL,
  bin_name = "bin_name",
  abundance = "abundance",
  sample = "sample",
  sequence_name = "sequence_name",
  verbose = TRUE
)
```

Arguments

data	a strollur object
table	a data.frame containing bin_data assignments
bin_type	a string indicating the type of bin assignments. Default "otu".
reference	a list created by the function [new_reference]. Optional.
bin_name	a string containing the name of the column in 'table' that contains the bin names. Default column name is 'bin_name'.
abundance	a string containing the name of the column in 'table' that contains the bin abundances. Default column name is 'abundance'. Note: You must provide either abundance or sequence_name in the table.
sample	a string containing the name of the column in 'table' that contains the sample names for datasets where the abundances are broken down by sample. Default column name is 'sample'.
sequence_name	a string containing the name of the column in 'table' that contains the sequence names. Default column name is 'sequence_name'. Note: You must provide either abundance or sequence_name in the table.
verbose	a boolean whether or not you want progress messages. Default = TRUE.

Value

an updated strollur object

Examples

```
# To assign sequences to bins:

data <- new_dataset(dataset_name = "miseq_sop")
otu_data <- read_mothur_list(list = strollur_example("final.opti_mcc.list.gz"))

xdev_assign_bins(data = data, table = otu_data, bin_type = "otu")

# To add abundance only bin assignments:
```

```
data <- new_dataset(dataset_name = "miseq_sop")
otu_data <- read_mothur_rabund(rabund = strollur_example("final.opti_mcc.rabund"))

xdev_assign_bins(data = data, table = otu_data, bin_type = "otu")

# To add abundance bin assignments parsed by sample:

data <- new_dataset(dataset_name = "miseq_sop")
otu_data <- readRDS(strollur_example("miseq_shared_otu.rds"))

xdev_assign_bins(data = data, table = otu_data, bin_type = "otu")
```

`xdev_assign_bin_representative_sequences` *xdev_assign_bin_representative_sequences*

Description

Assign representative sequences to bins.

Usage

```
xdev_assign_bin_representative_sequences(
  data,
  table,
  bin_type = "otu",
  reference = NULL,
  bin_name = "bin_name",
  sequence_name = "sequence_name",
  verbose = TRUE
)
```

Arguments

`data`	a strollur object
`table`	a data.frame containing bin representative assignments
`bin_type`	a string indicating the type of bin assignments. Default "otu".
`reference`	a list created by the function [new_reference]. Optional.
`bin_name`	a string containing the name of the column in 'table' that contains the bin names. Default column name is 'bin_name'.
`sequence_name`	a string containing the name of the column in 'table' that contains the bin names. Default column name is 'sequence_name'.
`verbose`	a boolean whether or not you want progress messages. Default = TRUE.

Value

an updated strollur object

Examples

```
miseq <- miseq_sop_example()

bin_reps <- readRDS(strollur_example(
       "miseq_representative_sequences.rds"))

xdev_assign_bin_representative_sequences(data = miseq,
                                         table = bin_reps,
                                         bin_type = "otu")
```

`xdev_assign_bin_taxonomy` *xdev_assign_bin_taxonomy*

Description

Assign bin classifications to a strollur object

Note, if you assign sequence taxonomies and assign bins, 'Dataset' will find the concensus taxonomy for each bin for you.

Usage

```
xdev_assign_bin_taxonomy(
  data,
  table,
  bin_type = "otu",
  reference = NULL,
  bin_name = "bin_name",
  taxonomy = "taxonomy",
  verbose = TRUE
)
```

Arguments

`data`	a strollur object
`table`	a data.frame containing bin taxonomy assignments
`bin_type`	a string indicating the type of bin assignments. Default "otu".
`reference`	a list created by the function [new_reference]. Optional.
`bin_name`	a string containing the name of the column in 'table' that contains the bin names. Default column name is 'bin_name'.
`taxonomy`	a string containing the name of the column in 'table' that contains the bin taxonomies. Default column name is 'taxonomy'.
`verbose`	a boolean whether or not you want progress messages. Default = TRUE.

Value

an updated strollur object

Examples

```
otu_data <- read_mothur_cons_taxonomy(strollur_example(
                        "final.cons.taxonomy"))

data <- new_dataset(dataset_name = "my_dataset")

# assign otu abundances
xdev_assign_bins(data = data, table = otu_data, bin_type = "otu")

# assign otu classifications
xdev_assign_bin_taxonomy(data = data, table = otu_data,
                         bin_type = "otu")
```

---

xdev_assign_sequence_abundance

xdev_assign_sequence_abundance

Description

Assign sequence abundance and optionally assign sample and treatment data to a strollur object

Usage

```
xdev_assign_sequence_abundance(
  data,
  table,
  sequence_name = "sequence_name",
  abundance = "abundance",
  sample = "sample",
  treatment = "treatment",
  verbose = TRUE
)
```

Arguments

data	a strollur object
table	a data.frame containing sequence abundance assignments
sequence_name	a string containing the name of the column in 'table' that contains the sequence names. Default column name is 'sequence_name'.
abundance	a string containing the name of the column in 'table' that contains the sequence abundances. Default column name is 'abundance'.

sample	a string containing the name of the column in 'table' that contains the sequence samples. Default column name is 'sample'. (Optional)
treatment	a string containing the name of the column in 'table' that contains the sequence treatments. Default column name is 'treatment'.
verbose	a boolean whether or not you want progress messages. Default = TRUE.

Value

an updated strollur object

Examples

```
data <- new_dataset("my_dataset")
sequence_abundance <- readRDS(strollur_example("miseq_abundance_by_sample.rds"))

xdev_assign_sequence_abundance(data = data, table = sequence_abundance)
```

---

xdev_assign_sequence_taxonomy

xdev_assign_sequence_taxonomy

Description

Assign sequence classifications to a strollur object

Note, if you assign sequence taxonomies and assign bins, strollur will find the consensus taxonomy for each bin for you.

Usage

```
xdev_assign_sequence_taxonomy(
  data,
  table,
  reference = NULL,
  sequence_name = "sequence_name",
  taxonomy = "taxonomy",
  verbose = TRUE
)
```

Arguments

data	a strollur object
table	a data.frame containing sequence taxonomy assignments
reference	a list created by the function [new_reference]. Optional.
sequence_name	a string containing the name of the column in 'table' that contains the sequence names. Default column name is 'sequence_name'.

taxonomy	a string containing the name of the column in 'table' that contains the sequence taxonomies. Default column name is 'taxonomy'.
verbose	a boolean whether or not you want progress messages. Default = TRUE.

Value

an updated strollur object

Examples

```
sequence_classifications <- read_mothur_taxonomy(strollur_example(
                        "final.taxonomy.gz"))

data <- new_dataset("my_dataset")

xdev_assign_sequence_taxonomy(data, sequence_classifications)

# With the reference parameter you can add information about the reference
# you used to classify your sequences. You can also add references using the
# 'add_references' function.

reference <- new_reference("trainset9_032012.pds.zip", "9_032012",
              "classification by mothur2 v1.0 using default options", "",
"https://mothur.s3.us-east-2.amazonaws.com/wiki/trainset9_032012.pds.zip")

xdev_assign_sequence_taxonomy(data, sequence_classifications, reference)
```

---

xdev_assign_sequence_taxonomy_tidy

xdev_assign_sequence_taxonomy_tidy

Description

Assign sequence classifications to a strollur object

Note, if you assign sequence taxonomies and assign bins, strollur will find the consensus taxonomy for each bin for you.

Usage

```
xdev_assign_sequence_taxonomy_tidy(
  data,
  table,
  reference = NULL,
  sequence_name = "sequence_name",
  level = "level",
  taxonomy = "taxonomy",
  confidence = "confidence",
  verbose = TRUE
)
```

Arguments

data	a strollur object
table	a data.frame containing sequence taxonomy assignments
reference	a list created by the function [new_reference]. Optional.
sequence_name	a string containing the name of the column in 'table' that contains the sequence names. Default column name is 'sequence_name'.
level	a string containing the name of the column in 'table' that contains the taxonomy levels. Default column name is 'level'.
taxonomy	a string containing the name of the column in 'table' that contains the sequence taxonomies. Default column name is 'taxonomy'.
confidence	a string containing the name of the column in 'table' that contains the taxonomies confidence. Default column name is 'confidence'.
verbose	a boolean whether or not you want progress messages. Default = TRUE.

Value

an updated strollur object

Examples

```
sequence_classifications <- readRDS(strollur_example("miseq_tidy_taxonomy.rds"))
str(sequence_classifications)

data <- new_dataset("my_dataset")

xdev_assign_sequence_taxonomy_tidy(data, sequence_classifications)

# With the reference parameter you can add information about the reference
# you used to classify your sequences. You can also add references using the
# 'add_references' function.

reference <- new_reference("trainset9_032012.pds.zip", "9_032012",
              "classification by mothur2 v1.0 using default options", "",
"https://mothur.s3.us-east-2.amazonaws.com/wiki/trainset9_032012.pds.zip")

xdev_assign_sequence_taxonomy_tidy(data, sequence_classifications, reference)
```

---

xdev_assign_treatments

xdev_assign_treatments

Description

Assign samples to treatments in a strollur object

Usage

```
xdev_assign_treatments(
  data,
  table,
  sample = "sample",
  treatment = "treatment",
  verbose = TRUE
)
```

Arguments

data	a strollur object
table	a data.frame containing sample treatment assignments
sample	a string containing the name of the column in 'table' that contains the samples. Default column name is 'sample'.
treatment	a string containing the name of the column in 'table' that contains the treatments. Default column name is 'treatment'.
verbose	a boolean indicating whether or not you want progress messages. Default = TRUE.

Value

an updated strollur object

Examples

```
data <- new_dataset("my_dataset")
sequence_abundance <- readRDS(strollur_example("miseq_abundance_by_sample.rds"))

xdev_assign_sequence_abundance(data, sequence_abundance)

sample_assignments <- readRDS(strollur_example("miseq_sample_design.rds"))

xdev_assign_treatments(data, sample_assignments)
```

---

xdev_count *xdev_count*

Description

Find the number of sequences, samples, treatments or bins of a given type in a strollur object

Usage

```
xdev_count(
  data,
  type = "sequence",
  bin_type = "otu",
  samples = NULL,
  distinct = FALSE
)
```

Arguments

data	a strollur object
type	string containing the type of data you want the number of. Options include: "sequence", "sample", "treatment", "bin" and "resource_reference". Default = "sequence".
bin_type	string containing the bin type you would like the number of bins for. Default = "otu".
samples	vector of strings. samples is only used when 'type' = "sequence" or 'type' = "bin" . samples should contain the names of the samples you want the count for. Default = NULL.
distinct	Boolean. distinct is used when 'type' = "sequence" or 'type' = "bin". When 'type' = "sequence" and distinct is TRUE the number of unique sequences is returned. When 'type' = "sequence" and distinct is FALSE total number of sequences is returned. This can also be combined with samples to find the number of unique sequences found only in a given set of samples, or to find the total number of sequences in a given set of samples. When 'type' = "bin", you can set distinct = TRUE to return the number of bins that ONLY contain sequences from the given samples. When distinct is FALSE the count returned contains bins with sequences from a given samples, but those bins may also contain other samples. Default = FALSE.

Value

double

Examples

```
miseq <- miseq_sop_example()

# To get the total number of sequences
xdev_count(data = miseq, type = "sequence")

# To get number of unique sequences
xdev_count(data = miseq, type = "sequence", distinct = TRUE)

# To get number of unique sequences from samples 'F3D0' and 'F3D1'
# Note these sequences will be present in both samples but may be
# be present in other samples as well
```

```
xdev_count(data = miseq, type = "sequence", samples = c("F3D0", "F3D1"))

# To get number of unique sequences exclusive to samples 'F3D0' and 'F3D1'
# Note these sequences are present in both samples and NOT present in
# other samples
xdev_count(data = miseq, type = "sequence", samples = c("F3D0", "F3D1"),
distinct = TRUE)

# To get the number of samples in the dataset
xdev_count(data = miseq, type = "sample")

# To get the number of treatments in the dataset
xdev_count(data = miseq, type = "treatment")

# To get the number of "otu" bins in the dataset
xdev_count(data = miseq, type = "bin", bin_type = "otu")

# To get the number of "asv" bins in the dataset
xdev_count(data = miseq, type = "bin", bin_type = "asv")

# To get the number of "phylotype" bins in the dataset
xdev_count(data = miseq, type = "bin", bin_type = "phylotype")

# To get number of bins from samples 'F3D0' and 'F3D1'
# Note these bins will have sequences from both samples but there may be
# other samples present as well
xdev_count(data = miseq, type = "bin", samples = c("F3D0", "F3D1"))

# To get number of bins unique to samples 'F3D0' and 'F3D1'
# Note these bins will have sequences from both samples and NO other samples
# will be present in the bins.
xdev_count(data = miseq, type = "bin", samples = c("F3D0", "F3D1"),
distinct = TRUE)
```

`xdev_get_abundances_by_sample` *xdev_get_abundances_by_sample*

Description

Get the sequence abundance data in a strollur object parsed by sample

Usage

```
xdev_get_abundances_by_sample(data, samples = as.character(c()))
```

Arguments

data	a strollur object
samples	a vector of strings containing the names of the samples you would like sequence names for. By default all samples are included.

Value

2D vector of float containing data requested parsed by sample.

Examples

```
data <- miseq_sop_example()

# To get the sequence names parsed by sample
abunds <- xdev_get_abundances_by_sample(data)
```

---

xdev_get_by_sample *xdev_get_by_sample*

Description

Get the requested data in a strollur object parsed by sample

Usage

```
xdev_get_by_sample(
  data,
  type = "sequence_name",
  samples = as.character(c()),
  degap = FALSE
)
```

Arguments

data	a strollur object
type	string containing the type of data you want the totals of. Options include: "sequence_name", "sequence". Default = "sequence_name".
samples	a vector of strings containing the names of the samples you would like sequence names for. By default all samples are included.
degap	a logical. Default = FALSE. When degap = 'TRUE', all gap characters will be removed from the sequences.

Value

2D vector of strings ([num_seqs][num_samples]) containing data requested parsed by sample.

Examples

```
data <- miseq_sop_example()

# To get the sequence names parsed by sample
xdev_get_by_sample(data, "sequence_name")

# To get the sequence nucleotide strings parsed by sample
parsed_sequences <- xdev_get_by_sample(data, "sequence")
```

---

xdev_get_list_vector *xdev_get_list_vector*

Description

Get vector of strings containing the sequences bin data

Usage

```
xdev_get_list_vector(data, type = "otu")
```

Arguments

data	a strollur object
type	a string indicating the type of bin assignments. Default "otu".

Value

vector of strings containing the names of the sequences in each bin separated by commas

Examples

```
data <- miseq_sop_example()
xdev_get_list_vector(data)
```

xdev_get_sequences *xdev_get_sequences*

Description

Get the nucleotide strings for each sequence in a strollur object

Usage

```
xdev_get_sequences(data, sample = "", degap = FALSE)
```

Arguments

data	a strollur object
sample	a string containing the name of the sample you would like sequence names for. For all samples in dataset, sample = "".
degap	a logical. Default = FALSE. When degap = 'TRUE', all gap characters ('-', '.') will be removed from the sequences.

Value

vector of string containing nucleotide strings of the sequences in a strollur object

Examples

```
data <- miseq_sop_example()
xdev_get_sequences(data)
```

xdev_has_sequence_taxonomy *xdev_has_sequence_taxonomy*

Description

Determine if a strollur object has sequence taxonomy assignments

Usage

```
xdev_has_sequence_taxonomy(data)
```

Arguments

data	a strollur object

Value

boolean

Examples

```
data <- miseq_sop_example()
xdev_has_sequence_taxonomy(data)
```

---

xdev_merge_bins *xdev_merge_bins*

Description

Designed with package integration in mind, the merge bins function allows you to merge bins in a strollur object

Usage

```
xdev_merge_bins(data, bin_names, reason = "merged", bin_type = "otu")
```

Arguments

data	a strollur object.
bin_names	a vector of strings containing the names of the bins you would like merge. The resulting merged bin will be stored in the first bin_id in the vector.
reason	a string indicating why you are merging bins. Default = "merged".
bin_type	a string indicating the type of bin clusters. Default = "otu"

Value

an updated strollur object

Examples

```
data <- miseq_sop_example()

# to merge otu5 and otu6

bins_to_merge <- c("Otu005", "Otu006")

xdev_merge_bins(data = data, bin_names = bins_to_merge)

# If you look at the scrap report, you will see Otu006 with the trash code
# set to "merged".

report(data = data, type = "bin_scrap")
```

xdev_merge_sequences *xdev_merge_sequences*

Description

Designed with package integration in mind, the merge sequences function allows you to merge sequences in a strollur object.

Usage

```
xdev_merge_sequences(data, sequence_names, reason = "merged")
```

Arguments

data	a strollur object.
sequence_names	a vector of strings containing the names of the sequences you would like merge. The resulting merged sequence will be stored in the first sequence name in the vector.
reason	a string indicating why you are merging sequences. Default = "merged"

Value

an updated strollur object

Examples

```
sequence_names <- c("seq1", "seq2", "seq3", "seq3",
                "seq4", "seq4", "seq5", "seq6",
                "seq7", "seq8", "seq9", "seq9",
                "seq10", "seq10", "seq10", "seq10")

samples <- c("sample1", "sample2", "sample4", "sample5",
             "sample1", "sample2", "sample1", "sample1",
             "sample2", "sample4", "sample4", "sample5",
             "sample1", "sample3", "sample5", "sample6")

abundances <- c(10, 10, 5, 5, 5, 5,
                10, 10, 10, 10, 5, 5,
                1, 2, 3, 4)

data <- new_dataset("my_data")


assign(data = data,
       table = data.frame(sequence_name = sequence_names,
                          abundance = abundances,
                          sample = samples),
       type = "sequence_abundance")
```

```
# For the sake of example let's merge the first 3 sequences.

seqs_to_merge <- c("seq1", "seq2", "seq3")

xdev_merge_sequences(data = data, sequence_names = seqs_to_merge)

# If you look at the scrap report, you will see the second two sequence
# names, listed with the trash code set to "merged".

report(data = data, type = "sequence_scrap")

# You can see from the get_num_sequences function that the merged sequence's
# abundances are added to the first sequence.

count(data = data, type = "sequence")
```

`xdev_names` *xdev_names*

Description

Get the names of a given type of data in a strollur object

Usage

```
xdev_names(
  data,
  type = "sequence",
  bin_type = "otu",
  samples = NULL,
  distinct = FALSE
)
```

Arguments

`data`	a strollur object
`type`	string containing the type of data you would like. Options include: "dataset", "sequence", "bin", "sample", "treatment", "report". Default = "sequence".
`bin_type`	string containing the bin type you would like the names for. Default = "otu".
`samples`	vector of strings. samples is only used when 'type' = "sequence" or 'type' = "bin" . samples should contain the names of the samples you want names for. Default = NULL.
`distinct`	Boolean. distinct is used when 'type' = "sequence" or 'type' = "bin" and the samples parameter is used. The distinct parameter allows you to get the names that are unique to a given set of samples. When distinct is TRUE, the names function will return the names that ONLY contain data from the given samples. When distinct is FALSE the data returned contains data from a given samples, but may ALSO contain data from other samples. Default = FALSE.

Value

vector of strings, containing the names requested

Examples

```
miseq <- miseq_sop_example()

# To get the name of the dataset
xdev_names(data = miseq, type = "dataset")

# To get the names of the sequences in the dataset
xdev_names(data = miseq, type = "sequence")

# To get the names of the sequences that are unique to sample 'F3D0'
xdev_names(data = miseq, type = "sequence", samples = c("F3D0"), distinct = TRUE)

# To get the names of the sequences that include sample 'F3D0'
xdev_names(data = miseq, type = "sequence", samples = c("F3D0"))

# To get the names of the samples in the dataset
xdev_names(data = miseq, type = "sample")

# To get the names of the treatments in the dataset
xdev_names(data = miseq, type = "treatment")

# To get the names of the bins in the dataset
xdev_names(data = miseq, type = "bin")

# To get the names of the bins in the dataset that are unique to 'F3D0'
xdev_names(data = miseq, type = "bin", samples = c("F3D0"), distinct = TRUE)

# To get the names of the bins in the dataset that include sequences
# from 'F3D0'
xdev_names(data = miseq, type = "bin", samples = c("F3D0"), distinct = FALSE)

# To get the names of the reports in the dataset
xdev_names(data = miseq, type = "report")
```

---

xdev_remove_bins *xdev_remove_bins*

Description

Designed with package integration in mind, the remove bins function allows you to remove bins from a strollur object

Usage

```
xdev_remove_bins(data, bin_names, trash_tags, bin_type = "otu")
```

Arguments

data	a strollur object.
bin_names	a vector of strings containing the names of the bins you would like removed.
trash_tags	a vector of strings containing the reasons you are removing each bin
bin_type	a string indicating the type of clusters.

Value

an updated strollur object

Examples

```
data <- new_dataset(dataset_name = "my_dataset")

bin_names <- c("bin1", "bin2", "bin3")
abundances <- c(110, 525, 80)

xdev_assign_bins(data = data,
                 table = data.frame(bin_name = bin_names,
                                    abundance = abundances),
                 bin_type = "otu")

count(data = data, type = "bin", bin_type = "otu")

bins_to_remove <- c("bin1")
trash_tag <- c("bad_bin")

xdev_remove_bins(data = data,
                 bin_names = bins_to_remove,
                 trash_tags = trash_tag)

count(data = data, type = "bin", bin_type = "otu")
```

---

xdev_remove_lineages *xdev_remove_lineages*

Description

Designed with package integration in mind, the remove lineages function allows you to remove contaminents from a strollur

Usage

```
xdev_remove_lineages(data, contaminants, reason = "contaminant")
```

Arguments

data	a strollur object.
contaminants	vector of strings containing the taxonomies you would like to remove
reason	a string containing reason you are removing the lineages. Default = "contaminant".

Value

an updated strollur object

Examples

```
data <- read_mothur(fasta = strollur_example("final.fasta.gz"),
                        count = strollur_example("final.count_table.gz"),
                        taxonomy = strollur_example("final.taxonomy.gz"),
                        design = strollur_example("mouse.time.design"),
                        otu_list = strollur_example("final.opti_mcc.list.gz"),
                        dataset_name = "miseq_sop")

contaminants <- c("Chloroplast", "Mitochondria", "unknown", "Archaea",
 "Eukaryota")

xdev_remove_lineages(data = data, contaminants = contaminants)
```

---

xdev_remove_samples *xdev_remove_samples*

Description

Designed with package integration in mind, the remove samples function allows you to remove samples from a strollur object

Usage

```
xdev_remove_samples(data, samples, reason = "remove_samples")
```

Arguments

data	a strollur object.
samples	vector of strings containing the names of the samples to remove.
reason	string containing the reason for removal. Default = "remove_samples".

Value

an updated strollur object

Examples

```
data <- miseq_sop_example()

count(data = data, type = "sample")

# To remove samples 'F3D0' and 'F3D1'

xdev_remove_samples(data, c("F3D0", "F3D1"))

count(data = data, type = "sample")
```

`xdev_remove_sequences` *xdev_remove_sequences*

Description

Designed with package integration in mind, the remove sequences function allows you to remove sequences from a strollur object

Usage

```
xdev_remove_sequences(data, sequence_names, trash_tags)
```

Arguments

`data`	a strollur object.
`sequence_names`	vector of strings containing the names of the sequences to remove
`trash_tags`	vector of strings containing the reasons for the sequences removals

Value

an updated strollur object

Examples

```
data <- miseq_sop_example()

count(data = data, type = "sequence")

# For the sake of example let's remove the first 3 sequences from
# miseq_sop_example:

seqs_to_remove <- c("M00967_43_000000000-A3JHG_1_2101_16474_12783",
                    "M00967_43_000000000-A3JHG_1_1113_12711_3318",
                    "M00967_43_000000000-A3JHG_1_2108_14707_9807")
trash_codes <- c("example", "removing", "sequences")
```

```
xdev_remove_sequences(data = data, sequence_names = seqs_to_remove,
                      trash_tags = trash_codes)

# If you look at the scrap report, you the sequences names, listed with the
# trash codes set to "example", "removing", "sequences".

report(data = data, type = "sequence_scrap")

# You can see from the get_num_sequences function that the removed
# sequence's abundances are removed from the dataset.

count(data = data, type = "sequence")
```

---

xdev_report *xdev_report*

Description

Get a data.frame containing the given report in a strollur object

Usage

```
xdev_report(data, type = "sequence", bin_type = "otu")
```

Arguments

data	a strollur object
type	string containing the type of report you would like. Options include: "fasta", "sequence", "sequence_bin_assignment", "sequence_taxonomy", "bin_taxonomy", "bin_representative", "sample_assignment", "metadata", "resource_reference", "sequence_scrap", "bin_scrap". If you have added custom reports for alignment, contigs_assembly or chimeras, you can get those as well. Default = "sequence".
bin_type	string containing the bin type you would like a bin_taxonomy report for. Default = "otu".

Value

data.frame

Examples

```
# First let's create a dataset from the \href{https://mothur.org/wiki/miseq_sop/}{MiSeq_SOP}

miseq <- miseq_sop_example()

# To get the FASTA data

fasta <- xdev_report(data = miseq, type = "fasta")
```

```
head(fasta, n = 10)

# To get a report about the FASTA data

sequence_report <- xdev_report(data = miseq, type = "sequence")
head(sequence_report, n = 10)

# To get the sequence bin assignments

bin_assignments <- xdev_report(data = miseq,
                               type = "sequence_bin_assignment",
                               bin_type = "otu")
head(bin_assignments, n = 10)

# To get the sample treatment assignments

xdev_report(data = miseq, type = "sample_assignment")

# To get a report about sequence classifications

sequence_taxonomy_report <- xdev_report(data = miseq,
                                        type = "sequence_taxonomy")
head(sequence_taxonomy_report, n = 10)

# To get a report about bin classifications for 'otu' data

otu_taxonomy_report <- xdev_report(data = miseq,
                                   type = "bin_taxonomy",
                                   bin_type = "otu")
head(otu_taxonomy_report, n = 10)

# To get a report about bin classifications for 'asv' data

asv_taxonomy_report <- xdev_report(data = miseq,
                                   type = "bin_taxonomy",
                                   bin_type = "asv")
head(asv_taxonomy_report, n = 10)

# To get a report about bin classifications for 'phylotype' data

phylotype_taxonomy_report <- xdev_report(data = miseq,
                                         type = "bin_taxonomy",
                                         bin_type = "phylotype")
head(phylotype_taxonomy_report, n = 10)

# To get the 'otu' bin representative sequences

otu_bin_reps <- xdev_report(data = miseq,
                            type = "bin_representative",
                            bin_type = "otu")
head(otu_bin_reps, n = 10)

# To get a report about the sequences removed during your analysis:
```

```
scrapped_sequence_report <- xdev_report(data = miseq,
                                        type = "sequence_scrap")

# To get a report about the "otu" bins removed during your analysis:

scrapped_otu_report <- xdev_report(data = miseq,
                                   type = "bin_scrap",
                                   bin_type = "otu")

# To get a report about the "phylotype" bins removed during your analysis:

scrapped_phylotype_report <- xdev_report(data = miseq,
                                         type = "bin_scrap",
                                         bin_type = "phylotype")

# To get the metadata associated with your data:

metadata <- xdev_report(data = miseq, type = "metadata")

# To get the resource references associated with your data:

references <- xdev_report(data = miseq, type = "resource_reference")

# To get our custom report containing the contigs assembly data:

contigs_report <- xdev_report(data = miseq, type = "contigs_report")
head(contigs_report, n = 10)
```

---

xdev_set_abundance *xdev_set_abundance*

Description

Designed with package integration in mind, the set abundance function allows you to change the abundances of sequences in a strollur object without samples.

Usage

```
xdev_set_abundance(
  data,
  sequence_names,
  sequence_abundances,
  reason = "update"
)
```

Arguments

data	a strollur object
sequence_names	a vector of strings containing sequence names
sequence_abundances	vector containing the abundances of each sequence.
reason	a string containing the trash tag to be applied to any sequences set to 0 abundance. Default = "update".

Value

an updated strollur object

Examples

```
names <- c("seq1", "seq2", "seq3",  "seq4")
abunds <- c(1250, 65, 50, 4)

data <- new_dataset(dataset_name = "my_dataset")

xdev_assign_sequence_abundance(data = data, table = data.frame(sequence_name = names,
                                                               abundance = abunds))
abundance(data = data, type = "sequence")

seqs_to_update <- c("seq1", "seq3")
new_abunds <- c(1000, 100)

xdev_set_abundance(data = data,
                   sequence_names = seqs_to_update,
                   sequence_abundances = new_abunds)

abundance(data = data, type = "sequence")
```

---

xdev_set_abundances *xdev_set_abundances*

Description

Designed with package integration in mind, the set abundances function allows you to change the abundances of sequences in a strollur object with samples.

Usage

```
xdev_set_abundances(data, sequence_names, abundances, reason = "update")
```

Arguments

data	a strollur object
sequence_names	a vector of strings containing sequence names
abundances	2D vector ([num_seqs][num_samples]) containing the abundances of each sequence parsed by sample.
reason	a string containing the trash tag to be applied to any sequences set to 0 abundance. Default = "update".

Value

an updated strollur object

Examples

```
data <- new_dataset(dataset_name = "my_dataset")

sequence_names <- c("seq1", "seq1", "seq1", "seq2", "seq2", "seq2", "seq3",
                    "seq3", "seq4")
samples <- c("sample2", "sample3", "sample4", "sample2", "sample3",
             "sample4", "sample2", "sample3", "sample4")
abundances <- c(250, 400, 500, 25, 40, 50, 25, 25, 4)

xdev_assign_sequence_abundance(data = data,
                           table = data.frame(sequence_name = sequence_names,
                                              abundance = abundances,
                                              sample = samples))

seqs_to_update <- c("seq4")
new_abunds <- list(c(20, 10, 4))

xdev_set_abundances(data = data,
                    sequence_names = seqs_to_update,
                    abundances = new_abunds)
```

---

xdev_set_dataset_name *xdev_set_dataset_name*

Description

Designed with package integration in mind, set the name of a strollur object.

Usage

```
xdev_set_dataset_name(data, dataset_name)
```

Arguments

data	a strollur object
dataset_name	a string containing the desired name

Value

No return value, called for side effects.

Examples

```
data <- new_dataset(dataset_name = "my_dataset")
xdev_set_dataset_name(data = data, dataset_name = "new_dataset_name")
```

---

xdev_set_sequences *xdev_set_sequences*

Description

Designed with package integration in mind, the set sequences function allows you to change the nucleotide strings of sequences in a strollur object. For example, set_sequences may be used after alignment to overwrite the unaligned sequences with aligned sequences.

Usage

```
xdev_set_sequences(
  data,
  sequence_names,
  sequences,
  comments = as.character(c())
)
```

Arguments

data	a strollur object
sequence_names	a vector of strings containing sequence names
sequences	a vector of strings containing sequence nucleotide strings
comments	a vector of strings containing sequence comments. (Optional)

Value

an updated strollur object

Examples

```
data <- new_dataset(dataset_name = "my_dataset")

xdev_add_sequences(data = data,
              table = data.frame(sequence_name = c("seq1", "seq2",
                                                   "seq3", "seq4")))

xdev_set_sequences(data = data,
                   sequence_names = c("seq1", "seq2","seq3", "seq4"),
                   sequences = c("ATTGC", "ACTGC", "AGTGC", "TTTGC"))
```

Index