Package 'seqwrap'

May 19, 2026

Type Package

Title Item-by-Item Iterative Model Fitting

Version 0.7.0

Description Models high-dimensional data, such as RNA-seq or proteomic data using an item-by-item strategy. The package contains functions to wrap high-dimensional data and iterate over them using established R packages for regression modelling (e.g., 'glmmTMB' or 'mgcv').

License GPL-3

Encoding UTF-8

LazyData true

LazyDataCompression xz

URL https://github.com/trainome/seqwrap

BugReports https://github.com/trainome/seqwrap/issues

Depends R (>= 4.1)

Imports cli, S7, tibble, pbapply, parallel, stats, DHARMa, broom.mixed

Suggests testthat (>= 3.0.0), dplyr, knitr, purrr, quarto, glmmTMB, lme4, nlme, MASS, rmarkdown, gt, edgeR, tidyselect, ggplot2, cowplot, ggtext, R.rsp

Config/testthat/edition 3

Collate 'seqwrap-global.R' 'aaa-.R' 'data-helper.R' 'data.R' 'generic-summaries.R' 'seqwrap.R' 'seqwrap_mtf.R' 'simcounts.R' 'simcounts2.R'

VignetteBuilder quarto, R.rsp

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Daniel Hammarström [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-8360-2100>), Chidimma Echebiri [ctb] (ORCID: <https://orcid.org/0000-0003-3134-5186>)

Maintainer Daniel Hammarström <daniel.hammarstrom@inn.no>

Repository CRAN

Date/Publication 2026-05-19 10:00:02 UTC

Contents

`dungan_counts` *Dungan et al 2022 counts*

Description

Raw count data from Dungan et al. 2022 was downloaded from NCBI Gene Expression Omnibus (GEO) and prepared for use in the seqwrap package. Preparation included adding gene symbols to the first column of the count data frame (counts), and sorting relevant variables to metadata data frame (metadata).

Usage

```
dungan_counts
```

Format

A list with two data frames:

counts Raw gene expression count matrix (genes x samples).

genesymbol HGNC gene symbol

OS.. OV.. Raw counts, one column for each sample containing integer read counts, named by experimental id.

metadata Sample-level metadata.

seq_sample_id Experimental unit id (mouse identifier)

treatment Senolytic or control (Vehicle) treatment

surgery Surgery inducing overload (synergist ablation) or sham surgury

Source

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE195707

References

Dungan, C. M. et al. Senolytic treatment rescues blunted muscle hypertrophy in old mice. *GeroScience* 44, 1925–1940 (2022).

---

generic_evaluation *Generic evaluation for model fits*

Description

When no evaluation function is provided to seqwrap (eval_fun), this function uses DHARMa to simulate residuals and test for uniformity, dispersion and outliers. These tests could guide troubleshooting and detection of model misspecification.

Usage

```
generic_evaluation(x)
```

Arguments

x	A model fitted in seqwrap.

Value

A tidy data frame possible to combine using seqwrap_summarise

Examples

```
# The generic summary works on model objects
library(seqwrap)

if (requireNamespace("glmmTMB", quietly = TRUE)) {
  library(glmmTMB)

  dat <- simcounts(n_genes = 1000,
                   n_samples = 30,
                   beta_0 = 5,
                   overdispersion_min_max = c(1, 10))

  counts <- dat$data
  metadata <- dat$metadata
  metadata$ln_libsize <- log(colSums(counts[,-1]))

  # Save counts for gene i in the same data frame
  metadata$y <- as.integer(counts[1, -1])
```

```
  m <- glmmTMB(y ~ as.factor(x) + offset(ln_libsize),
               data = metadata,
               family = nbinom2)

  generic_summary(m)
  generic_evaluation(m)
}
```

---

generic_summary *Generic summaries for model parameter estimates.*

Description

When no summary function is provided to seqwrap (summary_fun), this function uses broom(.mixed)::tidy to give a table of model parameter estimates.

Usage

```
generic_summary(x)
```

Arguments

x	A model fitted in seqwrap.

Value

A tidy data frame possible to bind using seqwrap_summarise

Examples

```
# The generic summary works on model objects
library(seqwrap)

if (requireNamespace("glmmTMB", quietly = TRUE)) {
  library(glmmTMB)

  dat <- simcounts(n_genes = 1000,
                   n_samples = 30,
                   beta_0 = 5,
                   overdispersion_min_max = c(1, 10))

  counts <- dat$data
  metadata <- dat$metadata
  metadata$ln_libsize <- log(colSums(counts[,-1]))

  # Save counts for gene i in the same data frame
  metadata$y <- as.integer(counts[1, -1])

  m <- glmmTMB(y ~ as.factor(x) + offset(ln_libsize),
```

```
                data = metadata,
                family = nbinom2)

  generic_summary(m)
  generic_evaluation(m)
}
```

`print.seqwrapResults` *Print method for objects of class seqwrapResults*

Description

Invoking the print method on seqwrapResults gives a summary of the fitted objects.

Arguments

`x`	A seqwrapResults object
`...`	Currently unused; included for compatibility with the `print` generic.

Value

Invisibly returns the object

Examples

```
# Load packages and prepare data for examples --------------------------------

library(seqwrap)

if (requireNamespace("glmmTMB", quietly = TRUE)) {
library(glmmTMB)

dat <- simcounts2()

# Save simulated data as separate objects
counts <- dat$counts
metadata <- dat$metadata

# Prepare library size for use as offset
metadata$ln_libsize <- log(metadata$library_size)


# A mixed effects negative binomial model of RNA-seq counts ------------------

# Populate the seqwrap container
container <- seqwrap_compose(
  modelfun = glmmTMB::glmmTMB,
  arguments = list(
    formula = y ~ time * condition + (1|id) + offset(ln_libsize),
```

```
    family = glmmTMB::nbinom2()
  ),
  data = counts,
  metadata = metadata,
  samplename = "seq_sample_id"
)

# Run seqwrap using the container
results <- seqwrap(container,
                   cores = 1)
print(results)
}
```

`seqwrap` *A flexible upper-level wrapper for iterative modelling using any available fitting algorithm*

Description

A flexible upper-level wrapper for iterative modelling using any available fitting algorithm

Usage

```
seqwrap(
  y = NULL,
  modelfun = NULL,
  arguments = NULL,
  data = NULL,
  metadata = NULL,
  samplename = NULL,
  additional_vars = NULL,
  summary_fun = NULL,
  eval_fun = NULL,
  exported = list(),
  return_models = TRUE,
  save_models = FALSE,
  model_path = NULL,
  subset = NULL,
  cores = 1,
  verbose = TRUE
)
```

Arguments

`y`	An swcontainer object created with seqwrap_compose, a named list or a DGEList object.
`modelfun`	A model fitting function like stats::lm, glmmTMB::glmmTMB or lme4::lmer

`arguments` An alist or list of arguments to be passed to the fitting function, this should not contain data. Note that the formula must have y as the dependent variable.

`data` A data frame or a list of data frames with targets (e.g. genes, transcripts) as rows and sample names as columns. If rownames = FALSE (default), each data frame should have target identifications as the first column in the data frame(s). If rownames = TRUE row names will be converted to target identifications. If data is provided as a list, each element of the list should be named. The corresponding names be available as variables for the fitting function.

`metadata` A data frame with sample names (corresponding to column names in the target matrix) and design variables.

`samplename` A character value indicating the variable by which metadata can merge with the target data. This defaults to "seq_sample_id" as this is used in the trainomeMetaData package.

`additional_vars` A vector of additional variables that is contained in the metadata data set that is needed to fit the model. By default the metadata is reduced to variables contained in the slots formula/model/fixed/random in additional arguments. More variables may be needed for offsets, weights etc.

`summary_fun` A custom (user-created) function for evaluating/summarizing models. If NULL, a list of fitted models are returned

`eval_fun` A custom (user-created) function for model diagnostics/evaluation. If NULL, no evaluation/diagnostics of models are returned

`exported` A list of functions, values etc. to be passed to summary_fun and eval_fun. This list must contain any functions that should be used in model summarise or evaluations.

`return_models` Logical, should models be returned as part of the output? Save models during development on subsets of the data. If used on large data sets, this will result in large memory usage.

`save_models` Logical, should models be saved? Models may be saved on disk to save working memory.

`model_path` A character. The path to saved models.

`subset` A sequence, random samples or integers to indicate which rows to keep in data. This is useful if you want to test the model in a subset of targets. If left to the default (NULL), all rows will be used.

`cores` An integer indicating the number of cores to be used in parallel computations. If NULL, a sequential for loop is used. If "max", all available cores are used.

`verbose` Logical, should the function print diagnostics after checking the data container?

Details

This function provides a flexible wrapper to fit, summarize and evaluate statistical models fitted to high dimensional omics-type data. Models are fitted and passed to user defined functions to summarize and evaluate models.

Value

A nested list with three upper levels slots: models, a list of fitted objects; summaries, a list of summaries created from the summary_fun function; evaluations, a list of diagnostics created from eval_fun.

Examples

```
library(seqwrap)

if (requireNamespace("glmmTMB", quietly = TRUE)) {
library(glmmTMB)

# Simulate n targets
dat <- simcounts(n_genes = 10000)

# Save simulated data as separate objects
counts <- dat$data
metadata <- dat$metadata

# Prepare library size for use as offset
metadata$ln_libsize <- log(colSums(counts[,-1]))


# A mixed effects negative binomial model of RNA-seq counts -----------------

# Populate the seqwrap container
container <- seqwrap_compose(
  modelfun = glmmTMB::glmmTMB,
  arguments = list(
    formula = y ~ x + (1|cluster) + offset(ln_libsize),
    family = glmmTMB::nbinom2()
  ),
  data = counts,
  metadata = metadata,
  samplename = "sample"
)

# Run seqwrap using the container on a subset of targets
results <- seqwrap(container,
                   subset = 1:5,
                   cores = 1)


# Summarise results only contains the subset
summaries <- seqwrap_summarise(results)
}
```

`seqwrapResults` *seqwrapResults class*

Description

seqwrapResults constructor function.

Arguments

.data	A list containing initial data (inherited from S7::class_list)
models	A list of fitted model objects
summaries	A list of model summaries
evaluations	A list of model evaluations
errors	A data frame of errors and warnings. defaults to an empty dataframe
n	Number of samples
k	Number of targets
call_arguments	Character string of function arguments used
call_engine	Character string of modeling engine used
elapsed_time	An proc.time() object giving the time needed to complete iterative model fitting.

Value

An S7 object of class `seqwrap_results` storing fitted models, summaries, evaluations, and diagnostic information from a `seqwrap()` run.

Examples

```
# seqwrapResults is the S7 class returned by seqwrap(). End users do
# not normally call the constructor directly -- it is invoked
# internally once iterative model fitting has finished.

library(seqwrap)

dat <- simcounts(n_genes = 5)

container <- seqwrap_compose(
  modelfun = stats::lm,
  arguments = list(formula = y ~ x),
  data = dat$data,
  metadata = dat$metadata,
  samplename = "sample"
)

results <- seqwrap(container, subset = 1:2, cores = 1)

# The object returned by seqwrap() is a seqwrapResults instance
S7::S7_inherits(results, seqwrapResults)
```

seqwrap_compose *Compose a swcontainer object for use in the seqwrap function.*

Description

This function makes it possible to compose and run checks on combined data sets (meta data and target data) and fitting functions to avoid issues in iterative modelling. See examples and vignettes for details.

Usage

```
seqwrap_compose(
  x = NULL,
  modelfun,
  arguments,
  data,
  rownames = FALSE,
  metadata,
  targetdata = NULL,
  samplename = "seq_sample_id",
  additional_vars = NULL,
  summary_fun = NULL,
  eval_fun = NULL,
  exported = list(),
  update = list()
)
```

Arguments

x	An optional named list or DGEList object (DESeqDataSet not yet implemented).
modelfun	A model fitting function like stats::lm, glmmTMB::glmmTMB or lme4::lmer
arguments	An alist or list of arguments to be passed to the fitting function, this should not contain data. Note that the formula must have y as the dependent variable.
data	A data frame or a list of data frames with targets (e.g. genes, transcripts) as rows and sample names as columns. If rownames = FALSE (default), each data frame should have target identifications as the first column in the data frame(s). If rownames = TRUE row names will be converted to target identifications. If data is provided as a list, each element of the list should be named. The corresponding names be available as variables for the fitting function.
rownames	should row names in data be used as target identifications? Defaults to FALSE.
metadata	A data frame with sample names (corresponding to column names in the target matrix) and design variables.
targetdata	A data frame or a list with target-wise values (e.g. dispersion or start values) for each target. This data is made available for the model fitting function and can be used to specify target specific data in each iteration of seqwrap. When a

data frame is provided each row corresponds to the target specific value. When a list is provided, each column of the data frame is available and can be called by name.

`samplename` A character value indicating the variable by which metadata can merge with the target data. This defaults to "seq_sample_id" as this is used in the trainomeMetaData package.

`additional_vars` A vector of additional variables that is contained in the metadata data set that is needed to fit the model. By default the metadata is reduced to variables contained in the slots formula/model/fixed/random in additional arguments. More variables may be needed for offsets, weights etc.

`summary_fun` A custom (user-created) function for evaluating/summarizing models. If NULL, a list of fitted models are returned

`eval_fun` A custom (user-created) function for model diagnostics/evaluation. If NULL, no evaluation/diagnostics of models are returned

`exported` A list of functions, values etc. to be passed to summary_fun and eval_fun. This list must contain any functions that should be used in model summarise or evaluations.

`update` A list of named parameters to update a swcontainer object.

Value

A swcontainer object for direct use in seqwrap.

Examples

```
# Load packages and prepare data for examples ------------------------------

library(seqwrap)

if (requireNamespace("glmmTMB", quietly = TRUE)) {
library(glmmTMB)

dat <- simcounts2()

# Save simulated data as separate objects
counts <- dat$counts
metadata <- dat$metadata

# Prepare library size for use as offset
metadata$ln_libsize <- log(metadata$library_size)


# A mixed effects negative binomial model of RNA-seq counts -----------------

# Populate the seqwrap container
container <- seqwrap_compose(
  modelfun = glmmTMB::glmmTMB,
  arguments = list(
```

```
    formula = y ~ time * condition + (1|id) + offset(ln_libsize),
    family = glmmTMB::nbinom2()
  ),
  data = counts,
  metadata = metadata,
  samplename = "seq_sample_id"
)

# Run seqwrap using the container
results <- seqwrap(container,
                   cores = 1)




# Summarise results
summaries <- seqwrap_summarise(results)


# Including target-specific data ------------------------------------------

# Target specific data can be supplied to seqwrap_compose to enable, e.g.,
# the use of priors for empirical Bayes shrinkage. In this example we are
# setting a dummy-prior on the `condition` parameter and target-specific
# prior on the dispersion parameter.

fixef_priors <- data.frame(
  prior = "normal(0, 1)",
  class = "fixef",
  coef = "conditionB"
)

dips_priors <- data.frame(
  prior = c(
    "normal(0.5, 0.25)",
    "normal(1, 0.25)"),
  class = "fixef_disp",
  coef = 1
)


# Combine information in target-specific list
prior_list <- list()
for(i in 1:2) {

  prior_list[[i]] <- rbind(
    fixef_priors,
    dips_priors[i,]
  )


}
```

```
container <- seqwrap_compose(
  modelfun = glmmTMB::glmmTMB,
  # NOTE: The use of `alist` prevents evaluation of list components
  arguments = alist(
    formula = y ~ time * condition + (1|id) + offset(ln_libsize),
    family = glmmTMB::nbinom2(),
    priors = data.frame(
      prior = prior,
      class = class,
      coef = coef
    )
  ),
  data = counts,
  metadata = metadata,
  targetdata = prior_list,
  samplename = "seq_sample_id"
)

# Run seqwrap using the container
results_prior <- seqwrap(container,
                         # Return models to confirm use of prior information
                         # The use of prior information is not recommended
                         return_models = TRUE,
                         cores = 1)

# Confirm prior information
summary(results_prior@models[[1]])
# Compare to naive model
summary(results@models[[1]])
}
```

`seqwrap_summarise` *Summarise seqwrapResults objects*

Description

Summarise seqwrapResults objects

Usage

```
seqwrap_summarise(
  x,
  summaries = TRUE,
  evaluations = TRUE,
  errors = TRUE,
  verbose = TRUE
)
```

Arguments

x	A seqwrapResults object
summaries	Logical, should summaries be combined?
evaluations	Logical, should evaluations be combined?
errors	Logical, should errors be combined?
verbose	Logical should progress be printed? Default TRUE

Details

This functions attempts to summarize results from the summary and evaluation functions applied in each iteration during modelling. The function expects that the summary and evaluation functions return data frames.

Value

A list (invisibly) with up to two data frames: summaries, combined parameter summaries from each model, and evaluations, combined diagnostics from each model. Entries are omitted when the corresponding slot of x is empty or the user disables them via the summaries / evaluations arguments.

Examples

```
# Load packages and prepare data for examples --------------------------------

library(seqwrap)

if (requireNamespace("glmmTMB", quietly = TRUE)) {
library(glmmTMB)

# Simulate n targets
dat <- simcounts(n_genes = 10000)

# Save simulated data as separate objects
counts <- dat$data
metadata <- dat$metadata

# Prepare library size for use as offset
metadata$ln_libsize <- log(colSums(counts[,-1]))


# A mixed effects negative binomial model of RNA-seq counts -----------------

# Populate the seqwrap container
container <- seqwrap_compose(
  modelfun = glmmTMB::glmmTMB,
  arguments = list(
    formula = y ~ x + (1|cluster) + offset(ln_libsize),
    family = glmmTMB::nbinom2()
  ),
  data = counts,
```

```
  metadata = metadata,
  samplename = "sample"
)

# Run seqwrap using the container on a subset of targets
results <- seqwrap(container,
                   subset = 1:5,
                   cores = 1)


# Summarise results only contains the subset
summaries <- seqwrap_summarise(results)

# Get summaries
summaries$summaries

# Get model evaluations
summaries$evaluations
}
```

`simcounts` *Simulate counts from a simple experiment (paired or unpaired) using Poisson or negative binomial distributions.*

Description

This function is used for internal testing and tutorials.

Usage

```
simcounts(
  n_genes = 10,
  n_samples = 16,
  beta_0 = 1,
  sigma_0 = 0.5,
  beta_1 = 1,
  sigma_1 = 0.5,
  b_0 = 0.5,
  clusters = 8,
  sample_sd = 0.5,
  overdispersion_min_max = c(1, 10),
  seed = 123
)
```

Arguments

n_genes	Number of genes to simulate.
n_samples	Number of samples to simulate.
beta_0	Gene intercepts.
sigma_0	SD of gene intercepts
beta_1	Gene slopes.
sigma_1	SD of gene slopes
b_0	Cluster-specific intercept. If NULL, non-paired data is simulated.
clusters	Number of clusters. If NULL, non-paired data is simulated.
sample_sd	SD of sample-specific effects.
overdispersion_min_max	Range of overdispersion parameter. If NULL, the Poisson distribution is used to simulate data.
seed	Random seed for reproducibility.

Value

A list with three elements: data, a data frame of simulated counts with one row per gene and one column per sample (first column identifies the gene); parameters, a data frame of true gene-wise coefficients and overdispersion values; and metadata, a data frame of sample-level covariates (sample, cluster, x).

Examples

```
# Simulate n targets from the negative binomial distribution
dat <- simcounts(n_genes = 10,
                 overdispersion_min_max = c(1, 5))

# Simulate n targets from the Poisson distribution
dat <- simcounts(n_genes = 10,
                 overdispersion_min_max = NULL)


# Data are organized in counts
dat$data
# .. and meta data
dat$metadata
```

simcounts2 *Simulate counts from a parallel groups design with three time-points*

Description

Simulate gene counts from a Negative Binomial distribution conditional on a two-condition, repeated measures experiment with three time points. The function is used for internal testing and tutorials.

Usage

```
simcounts2(
  n1 = 5,
  n2 = 5,
  beta0 = c(2.3, 3),
  conditionB = c(0.2, 0.1),
  timet2 = c(0, 0),
  timet3 = c(0.5, 0.25),
  conditionB_timet2 = c(0.1, 0.2),
  conditionB_timet3 = c(0.5, 0.6),
  b0 = c(1, 1),
  b1 = c(0, 0),
  b2 = c(0, 0),
  phi_model = NULL,
  library_size = NULL,
  lib_size_mean = 1e+06,
  lib_size_cv = 0.3,
  max_prop = 0.02
)
```

Arguments

n1	Number clusters in group 1 (control)
n2	Number of clusters in group 2 (treatment)
beta0	The intercept parameter on the log scale.
conditionB	Baseline differences between conditions
timet2	Changes from baseline to time-point 2 in the reference group
timet3	Changes from baseline to time-point 3 in the reference group
conditionB_timet2	Difference from reference group in changes from baseline to time-point 2
conditionB_timet3	Difference from reference group in changes from baseline to time-point 3
b0	Vector of SD of the between cluster variation in intercept
b1	Vector of SD of the between cluster variation in timet1 effects

b2	Vector of SD of the between cluster variation in timet2 effects
phi_model	A model (lm or loess) of a dispersion ~ log_mu relationship
library_size	A vector of library sizes with the length (n1 + n2) * 3 or NULL if library sizes are to be simulated
lib_size_mean	Mean of the distribution of library sizes.
lib_size_cv	Coefficient of variation for the distribution of library sizes.
max_prop	The maximum count for a single observation as a proportion of the library size.

Details

Fixed effects parameter values are specified as vectors (beta0, conditionB, timet2, etc.) with the common vector length being the number of genes simulated.

The function simulation of counts from a conditional Negative Binomial distribution:

$$y_{i[g]} \sim \text{NB}(\mu_{i[g]}, \phi_{[g]})$$

$$\log(\mu_{i[g]}) = \beta_0 + \beta 1 \text{condition}_B + \beta_2 \text{time}_{t2} + \beta_3 \text{time}_{t3} + \beta_4 \text{condition}_B \times \text{time}_{t2} + \beta_5 \text{condition}_B \times \text{time}_{t3} + \text{offset(library size)} + \text{l}$$

Varying effects are added as:

$$\text{b}_l \sim \text{Normal}(0, \sigma_l)$$

The library size is used as an offset for simulating data. Library sizes are simulated from a log-normal distribution or provided in library_size. In the simulation of counts, the library size is entered as offset after scaling to the median library size. Raw library sizes are included in the meta data. The dispersion parameter can be simulated from a model (lm or loess) provided to the function where the predictor variable should be $\log(\mu)$ and the outcome variable (dispersion) should be $\log(\phi)$. If no model is provided, values are simulated from a log-normal distribution based on hard coded parameter values from an observed mean-dispersion relationship.

Value

A list of (1) simulated counts, (2) the eta, and (3) phi values used for simulating data, and (4) the meta data data frame.

Examples

```
# Simulate n_genes number of genes
n_genes <- 1000

dat <- simcounts2(
  n1 = 5,
  n2 = 5,
  beta0 = rnorm(n_genes, mean = 5),
  conditionB =rnorm(n_genes),
  timet2 = rnorm(n_genes),
  timet3 = rnorm(n_genes),
  conditionB_timet2 = rnorm(n_genes),
  conditionB_timet3 = rnorm(n_genes),
  b0 = abs(rnorm(n_genes)),
  b1 = abs(rnorm(n_genes)),
```

```
  b2 = abs(rnorm(n_genes))
)

# Data are organized in counts
dat$counts
# .. and meta data
dat$metadata
```

`swcontainer` *seqwrap container class*

Description

The seqwrap container is used to store and validate data input to the to seqwrap.

Arguments

`.data`	A list containing initial data (inherited from S7::class_list)
`modelfun`	A function used to fit models
`arguments`	A list of arguments for the fitting function
`data`	A data frame or list of data frames with target data
`rownames`	Logical, should row names be used as target IDs?
`metadata`	A data frame with sample information
`targetdata`	A data frame with target-wise information
`samplename`	Character for sample name identification
`additional_vars`	Character vector of additional variables
`summary_fun`	A function for summarizing models
`eval_fun`	A function for evaluating models
`exported`	A list of objects to export to workers
`model_print`	Character representation of model function
`arguments_print`	Character representation of arguments

Value

An S7 object of class `swcontainer` bundling data, metadata, and the modelling function plus arguments to be consumed by `seqwrap()`.

Examples

```
# swcontainer is the S7 class produced by seqwrap_compose(). End users
# normally build one via seqwrap_compose() rather than calling this
# constructor directly.

library(seqwrap)

dat <- simcounts(n_genes = 5)

container <- seqwrap_compose(
  modelfun = stats::lm,
  arguments = list(formula = y ~ x),
  data = dat$data,
  metadata = dat$metadata,
  samplename = "sample"
)

# The object returned by seqwrap_compose() is a swcontainer instance
S7::S7_inherits(container, swcontainer)
```

Index