Package 'seqwrap'

September 3, 2026

Type Package

Title Item-by-Item Iterative Model Fitting

Version 0.8.1

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, graphics, grDevices, broom.mixed

Suggests testthat (>= 3.0.0), DHARMa, mgcv, 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-chunk.R' 'seqwrap-priors.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 2026-09-03 11:20:02 UTC

Contents

`dispersion_evaluation` *Dispersion and mean count from a glmmTMB model*

Description

An evaluation function for `eval_fun` in `seqwrap()` and `seqwrap_compose()` that records the estimated dispersion of a negative binomial `glmmTMB` model together with the mean of the observed counts. It provides the per-target quantities that `seqwrap_priors()` uses to build a dispersion prior that depends on the expression level of each target.

Usage

```
dispersion_evaluation(x, ...)
```

Arguments

`x`	A model fitted with `glmmTMB::glmmTMB`, or NULL when fitting failed.
`...`	Currently unused.

Details

The dispersion is read from the `betadisp` element of the model's `sdreport`. When a dispersion formula contains covariates only its first element, the intercept, is reported. The convergence diagnostics returned by `generic_evaluation()` are not included; to keep both, wrap the two in a function that binds their results.

Value

A one row tibble, or NULL when `x` is NULL, with columns:

dispersion The dispersion parameter on the log scale, as estimated by `glmmTMB`. For `nbinom2` this is `log(theta)`, which equals `log(sigma(x))`.

dispersion.se The standard error of `dispersion`, NA when `glmmTMB` did not compute one.

log_mu The natural logarithm of the mean of the observed response values used to fit the model.

See Also

`seqwrap_priors()`, which consumes the columns returned here.

Examples

```
library(seqwrap)

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

  dat <- simcounts(n_genes = 10, n_samples = 30)

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

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

  dispersion_evaluation(m)

  # Equivalent to the dispersion reported by sigma() on the log scale
  log(sigma(m))
}
```

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 surgery

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 collects the convergence diagnostics reported by the fitting algorithm. It is an S3 generic, so the diagnostics are read from whichever model object the engine produced. Methods are supplied for `glmmTMB`, `lme4` (`merMod`), `nlme` (`lme` and `gls`), `mgcv` (`gam` and `bam`), `MASS::glm.nb` (`negbin`), and the base `glm` and `lm` classes. Unrecognised model classes fall through to a default method.

Usage

```
generic_evaluation(x, ...)

## Default S3 method:
generic_evaluation(x, ...)

## S3 method for class '`NULL`'
generic_evaluation(x, ...)

## S3 method for class 'glmmTMB'
generic_evaluation(x, ...)

## S3 method for class 'merMod'
generic_evaluation(x, ...)

## S3 method for class 'negbin'
generic_evaluation(x, ...)

## S3 method for class 'glm'
generic_evaluation(x, ...)

## S3 method for class 'lm'
generic_evaluation(x, ...)

## S3 method for class 'lme'
generic_evaluation(x, ...)

## S3 method for class 'gls'
generic_evaluation(x, ...)

## S3 method for class 'gam'
generic_evaluation(x, ...)
```

Arguments

`x`	A model fitted in seqwrap, or NULL when fitting failed.

... Currently unused, present so that methods can be extended.

Details

converged and singular are deliberately separate. A fit can satisfy the optimiser's convergence criterion and still be untrustworthy: glmmTMB regularly returns convergence code 0 alongside a non-positive-definite Hessian, and lme4 returns code 0 for singular random effect structures. Screening a large set of fits therefore usually means requiring converged & !singular rather than converged alone.

Earlier versions of this function ran DHARMa residual simulations. That is several orders of magnitude more expensive per target, which is prohibitive at the target counts this package is built for, and DHARMa is now an optional dependency. Use residual_diagnostics() to get the previous behaviour.

Value

A one row tibble that can be combined using seqwrap_summarise(), or NULL when x is NULL. The columns are:

engine The model class the diagnostics were read from.

converged Whether the fitting algorithm reported successful convergence. NA when the engine reports no convergence signal.

code The numeric convergence code, where the engine supplies one. Zero conventionally indicates success.

message Any convergence message, NA when the fit was clean.

singular Whether the fit is degenerate: on the boundary of the parameter space, rank deficient, or with a covariance matrix that is not positive definite.

iterations Iterations or function evaluations used, where reported.

Examples

```
# The generic evaluation 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_evaluation(m)
}

# Base engines are supported as well
fit <- stats::glm(mpg ~ wt, data = mtcars, family = stats::gaussian())
generic_evaluation(fit)
```

`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)
}
```

`plot.seqwrap_priors` *Plot priors over the estimates they were built from*

Description

Draws each prior held by a `seqwrap_priors` object over the per-target estimates it summarises, as a visual check that the priors are reasonable before they are used in a second run. Fixed effects and random-effect standard deviations are shown as a histogram of the estimates across targets with the prior density drawn on top. The dispersion prior is shown as the log dispersion of each target against its log mean count, with the fitted trend and a band of one prior standard deviation around it.

Usage

```
## S3 method for class 'seqwrap_priors'
plot(
  x,
  which = c("fixef", "ranef", "dispersion"),
  trim = 0.99,
  layout = TRUE,
  ...
)
```

Arguments

`x`	A `seqwrap_priors` object from `seqwrap_priors()`.
`which`	Character vector selecting the panels to draw, any of `"fixef"`, `"ranef"` and `"dispersion"`. Defaults to all three. Panels for prior classes the object does not hold are skipped.
`trim`	The central fraction of the estimates shown in each histogram. Defaults to 0.99, so that a few targets whose fits went astray do not compress the histogram; estimates outside the shown range are counted in the panel subtitle rather than drawn. Set to 1 to show every estimate.
`layout`	Logical, should the panels be arranged in a grid on the current device? Defaults to TRUE, in which case the graphical parameters are restored on exit. Set to FALSE to draw the panels one after another under a layout of your own, set with `graphics::par()` or `graphics::layout()`.
`...`	Accepted for compatibility with the `plot()` generic, not used.

Details

The plots are drawn with base graphics. The estimates behind them are kept in the `data` attribute of the object (see `seqwrap_priors()`), so that a different display can be built from the same values, for example with `ggplot2`. The method returns that attribute invisibly.

Random-effect priors are gamma distributions parameterised by their mean and shape, as in `glmmTMB`, so the density drawn has shape `shape` and rate `shape / mean`.

Value

Invisibly, the `data` attribute of `x`: a list with the data frames `estimates` and `dispersion` that the panels were drawn from.

See Also

`seqwrap_priors()`, whose `plot` argument calls this method.

Examples

```
library(seqwrap)

if (requireNamespace("glmmTMB", quietly = TRUE)) {
  dat <- simcounts(n_genes = 30, n_samples = 30, clusters = 15)
  counts <- dat$data
  metadata <- dat$metadata
  metadata$ln_libsize <- log(colSums(counts[, -1]))

  first <- seqwrap_compose(
    modelfun = glmmTMB::glmmTMB,
    arguments = list(
      formula = y ~ x + (1 | cluster) + offset(ln_libsize),
      family = glmmTMB::nbinom2
    ),
    data = counts,
    metadata = metadata,
    samplename = "sample",
    eval_fun = dispersion_evaluation
  )
  first_results <- seqwrap(first, cores = 1, verbose = FALSE)

  priors <- seqwrap_priors(first_results, data = counts)
  plot(priors)

  # Only the dispersion trend
  plot(priors, which = "dispersion")

  # The estimates behind the panels, for a display of your own
  str(attr(priors, "data"))
}
```

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

Description

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

Details

The method is registered for the S7 class `seqwrapResults` and is called as `print(x, ...)`, where `x` is the object returned by `seqwrap()`. Further arguments (`...`) are accepted for compatibility with the `print()` generic but are not used.

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)
}
```

`print.seqwrap_priors` *Print method for seqwrap_priors objects*

Description

Printing a `seqwrap_priors` object lists the priors it holds and how they were built.

Details

The method is called as `print(x, ...)`, where `x` is the object returned by `seqwrap_priors()`. Further arguments (`...`) are accepted for compatibility with the `print()` generic but are not used.

Value

Invisibly returns the object.

`residual_diagnostics` *Residual diagnostics using DHARMa*

Description

Simulates scaled residuals with DHARMa and tests them for uniformity, dispersion and outliers. This was the behaviour of `generic_evaluation()` in seqwrap 0.7.0 and earlier, and is kept here as an opt-in evaluation function.

Usage

```
residual_diagnostics(x, n = 250)
```

Arguments

`x`	A model fitted in seqwrap, or NULL when fitting failed.
`n`	Number of simulations passed to `DHARMa::simulateResiduals()`.

Details

Simulating residuals is far more expensive than reading a convergence code: each target is refit-equivalent `n` times. On data sets with many thousands of targets this dominates the total run time, which is why it is no longer the default. Consider applying it to a subset of targets, or lowering `n`.

DHARMa is an optional dependency, so it has to be installed before this function can be used.

Value

A one row tibble of p-values, or NULL when x is NULL.

Examples

```
library(seqwrap)

if (requireNamespace("DHARMa", quietly = TRUE)) {
  fit <- stats::glm(mpg ~ wt, data = mtcars, family = stats::gaussian())
  residual_diagnostics(fit, n = 50)
}
```

---

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 = FALSE,
  save_models = FALSE,
  model_path = NULL,
  subset = NULL,
  chunk_size = NULL,
  cache = c("memory", "none", "disk"),
  cache_path = NULL,
  cores = 1,
  verbose = TRUE
)
```

Arguments

y	An swcontainer object created with `seqwrap_compose()`, or NULL to build one from the arguments given here.
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, `generic_summary()` is used, which tidies model parameters with `broom.mixed::tidy()`.
eval_fun	A custom (user-created) function for model diagnostics/evaluation. If NULL, `generic_evaluation()` is used, which reports the convergence diagnostics supplied by the fitting algorithm. Pass `residual_diagnostics` for DHARMa based residual checks instead, bearing in mind that these are far more expensive per target.
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? Defaults to FALSE, as retaining every fitted model is the largest memory cost a run can incur. Set it to TRUE while developing a model on a subset of targets, where inspecting the fitted objects is useful. A warning is raised when more than ten models would be retained. To keep models from a full run without holding them in memory, use `save_models` and `model_path`.
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.

chunk_size	An integer giving the number of targets handed to a worker as a single unit of work. Larger chunks reduce the per-task overhead of sending data to and from workers, which matters when the number of targets is large. If NULL (default) a chunk size is chosen automatically so that each worker receives several chunks. Chunking does not change the results.
cache	One of "none", "memory" or "disk", controlling how summaries and evaluations are accumulated. See Details.
cache_path	A character path to the directory used when cache = "disk". If NULL (default) a directory inside the session temporary directory is created and removed when the session ends.
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.

Caching summaries and evaluations:

Summary and evaluation functions return one small data frame per target. Holding these as separate objects is convenient but costs roughly ten times more memory than the same rows bound into a single data frame, because the per-data-frame overhead is paid once per target. At high target counts (hundreds of thousands, as in array-scale data) that overhead dominates. The cache argument controls the trade-off:

- "memory" (default) binds summaries and evaluations into a single data frame per slot while still on the worker, adding a target column. This requires that summary_fun and eval_fun return data frames.
- "none" keeps one data frame per target, so @summaries and @evaluations are named lists indexed by target. Use this when a summary or evaluation function returns something that cannot be row-bound, or when per-target access is more convenient than filtering a combined table.
- "disk" additionally writes each chunk's bound data frames to cache_path and leaves @summaries and @evaluations empty. Parent memory then stays flat regardless of the number of targets. Use seqwrap_summarise() to read and combine the cached chunks, and seqwrap_cache_clear() to remove them.

seqwrap_summarise() returns the same combined data frames whichever mode was used, so only code reading @summaries or @evaluations directly is affected by the choice.

Caching is independent of save_models and model_path, which continue to write one file per fitted model.

Note that cache = "disk" caches summaries and evaluations only. With return_models = TRUE the fitted models are still collected in memory, so the two options are normally combined as return_models = FALSE.

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.

Usage

```
seqwrapResults(
  .data = list(),
  models = NULL,
  summaries = NULL,
  evaluations = NULL,
  errors = data.frame(),
  targets = character(0),
  cache = NULL,
  n = integer(0),
  k = integer(0),
  call_arguments = character(0),
  call_engine = character(0),
  elapsed_time = numeric(0)
)
```

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 with one row per condition raised, holding the columns `target`, `stage`, `type`, `class` and `message`. Targets that raised nothing do not appear. Defaults to an empty data frame.
`targets`	A character vector of every target identifier, in the order the targets were fitted
`cache`	A list describing an on-disk cache of summaries and evaluations, or NULL when these are held in memory. See `seqwrap()`.
`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_cache_clear *Remove a seqwrap disk cache*

Description

Deletes the chunk files written when `seqwrap()` was called with `cache = "disk"`. A cache placed in the session temporary directory (the default) is removed when the R session ends, so calling this function is only necessary to reclaim space earlier, or when a user-supplied `cache_path` was used.

Usage

```
seqwrap_cache_clear(x, recursive = TRUE)
```

Arguments

`x`	A seqwrapResults object created with `cache = "disk"`.
`recursive`	Logical. Should the cache directory itself be removed in addition to the chunk files? Defaults to TRUE.

Value

The number of files removed, invisibly.

Examples

```
library(seqwrap)

dat <- simcounts(n_genes = 6)

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

results <- seqwrap(
  container,
  eval_fun = function(x) NULL,
  cache = "disk",
  cores = 1,
  verbose = FALSE
)

# Combine the cached chunks before clearing them
summaries <- seqwrap_summarise(results, verbose = FALSE)

seqwrap_cache_clear(results)
```

`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 swcontainer object to modify. When supplied, the properties named in update are changed and the container is returned; all other arguments are ignored.
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 and each column is available by name. When a list is provided, each element is a data frame whose columns are available by name. A list whose elements are named by target identifier is matched to targets by name and may hold more targets than data. A data frame, or a list without names, is matched by position, row or element i belonging to row i of data; seqwrap() warns when a list is matched this way. See seqwrap_priors() for building target-specific priors for glmmTMB.
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, generic_summary() is used, which tidies model parameters with broom.mixed::tidy().
eval_fun	A custom (user-created) function for model diagnostics/evaluation. If NULL, generic_evaluation() is used, which reports the convergence diagnostics

	supplied by the fitting algorithm. Pass `residual_diagnostics` for DHARMa based residual checks instead, bearing in mind that these are far more expensive per target.
`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. Models are returned here only so that
# they can be compared with the prior-based fits further down; a full run
# would normally leave return_models at its default of FALSE.
results <- seqwrap(container,
                   return_models = TRUE,
                   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 a target-specific list, named by target
prior_list <- list()
for(i in 1:2) {

  prior_list[[counts[i, 1]]] <- 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_errors` *Errors and warnings from a seqwrap run*

Description

Returns the errors and warnings raised during a `seqwrap()` run, with one row per condition and optional filtering. Targets that completed cleanly do not appear, so the result is proportional to the number of problems rather than to the number of targets.

Usage

```
seqwrap_errors(x, stage = NULL, type = NULL, target = NULL)
```

Arguments

`x`	A seqwrapResults object.
`stage`	Optional. One or more of `"fit"`, `"summary"` and `"evaluation"`, to restrict the result to conditions raised at those stages.
`type`	Optional. One or both of `"error"` and `"warning"`.
`target`	Optional. Target identifiers to restrict the result to.

Details

To identify targets that completed without any condition, compare against the `targets` property, which lists every target that was fitted: `setdiff(x@targets, seqwrap_errors(x)$target)`.

Value

A tibble with columns:

target The target identifier.

stage Where the condition arose: `"fit"`, `"summary"` or `"evaluation"`.

type Either `"error"` or `"warning"`.

class The class of the condition object, useful for separating specific warning types from generic ones.

message The condition message.

Examples

```
library(seqwrap)

dat <- simcounts(n_genes = 6)

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

results <- seqwrap(container, cores = 1, verbose = FALSE)

# Every condition raised during the run
seqwrap_errors(results)

# Only warnings from the fitting algorithm
seqwrap_errors(results, stage = "fit", type = "warning")

# Targets that completed cleanly
setdiff(results@targets, seqwrap_errors(results)$target)
```

---

seqwrap_priors *Build empirical Bayes priors for glmmTMB from a seqwrap run*

Description

Summarises the parameter estimates from a completed `seqwrap()` run and turns them into target-specific prior specifications for `glmmTMB::glmmTMB`. The result is a list with one prior data frame per target, ready to be passed as `targetdata` to `seqwrap_compose()` for a second, regularised run. This is a simple empirical Bayes strategy: every target is first fitted on its own, the spread of each parameter across targets then becomes the prior for that parameter.

Usage

```
seqwrap_priors(
  x,
  data,
  rownames = FALSE,
  terms = NULL,
  center = c("zero", "mean"),
  robust = FALSE,
  ranef = TRUE,
  shape = 2,
  dispersion = "dispersion",
  trend = c("loess", "constant"),
  span = 0.75,
  drop_warnings = FALSE,
  digits = 3,
  plot = FALSE
)
```

Arguments

x	A seqwrapResults object from a run without priors, or the list returned by seqwrap_summarise() for that run.
data	The target data frame given to seqwrap_compose(), with target identifiers in the first column (or in the row names when rownames = TRUE) and one column per sample. It determines the targets for which priors are built, their order, and the mean count of each target. When seqwrap_compose() was given a list of data frames, pass the element holding the counts.
rownames	Logical, are target identifiers held in the row names of data? Defaults to FALSE, as in seqwrap_compose().
terms	Character vector naming the fixed-effect terms to place priors on. NULL (the default) uses every fixed-effect term in the summaries except the intercept. The intercept can be included by naming it, "(Intercept)", which is sensible only with center = "mean".
center	Where to center the fixed-effect priors. "zero" (the default) shrinks estimates towards no effect, using normal(0, s) with s the spread of the estimates across targets. "mean" centers each prior on the average estimate across targets, normal(m, s).
robust	Logical. When TRUE, the median and median absolute deviation replace the mean and standard deviation when summarising estimates, and the dispersion trend is fitted with a robust loss. This limits the influence of targets whose fits went astray. Defaults to FALSE.
ranef	Logical, should priors be placed on random-effect standard deviations? Defaults to TRUE. Ignored when the summaries contain no random effect terms. A grouping variable whose standard deviation estimates are essentially zero across targets is skipped with a warning, since the gamma prior needs a positive mean.

shape	The shape of the gamma prior placed on random-effect standard deviations. glmmTMB parameterises this prior by its mean and shape, and the mean is taken from the estimates across targets. Defaults to 2.
dispersion	The name of the column in the evaluations that holds the log dispersion of each target, as produced by dispersion_evaluation(). Defaults to "dispersion". Set to NULL to build no dispersion prior.
trend	How the dispersion prior depends on expression level. "loess" (the default) fits a loess curve of log dispersion against log mean count and gives every target a prior centered on the curve at its own mean count, with the residual standard error of the curve as prior standard deviation. "constant" gives every target the same prior, centered on the average log dispersion across targets.
span	The span of the loess fit when trend = "loess". Defaults to 0.75.
drop_warnings	Passed to seqwrap_summarise() when x is a seqwrapResults object, to exclude targets that raised warnings from the prior estimation. Defaults to FALSE. Ignored when x is already a list of combined results.
digits	The number of decimals kept in the prior specifications. Defaults to 3.
plot	Logical, should the priors be plotted over the estimates they were built from before the object is returned? Defaults to FALSE. See plot.seqwrap_priors(), which can also be called on the result later.

Details

Three kinds of prior are built.

- Fixed effects (class "fixef"). For each term the estimates across targets are summarised by their location and spread, giving a normal prior normal(location, spread). Terms are identified from the effect and component columns of a generic_summary() result. With a custom summary function that lacks these columns, every term other than the intercept and terms starting with sd__ or cor__ is treated as a fixed effect.
- Random-effect standard deviations (class "ranef_sd"). For each grouping variable the standard deviation estimates across targets are averaged and used as the mean of a gamma prior, gamma(mean, shape), on the standard deviation scale. A grouping variable with several standard deviation terms receives one prior for all of them.
- Dispersion (class "fixef_disp"). The log dispersion of each target depends strongly on its expression level, so a trend of log dispersion against log mean count is estimated and each target receives a normal prior centered on the trend at its own mean count. Targets whose mean count lies outside the range seen during estimation, including targets with zero counts, receive the prior at the nearest end of that range.

The spread of estimates across targets includes the estimation error of each estimate, so the resulting priors are somewhat wider than the true between-target variation, which makes them conservative.

The container for the second run must ask for the priors by name, using alist() so that the data frame is built on the worker for each target:

```
arguments = alist(
  formula = ...,
  family = glmmTMB::nbinom2,
```

```
  priors = data.frame(prior = prior, class = class, coef = coef)
)
```

Priors are experimental in glmmTMB; see its priors help page and vignette for the parameterisation of each distribution.

Value

A named list of class seqwrap_priors with one element per row of data, in the same order and named by target identifier. Each element is a data frame with the columns prior, class and coef that glmmTMB::glmmTMB expects in its priors argument. The list carries attributes that describe how it was built: common, a data frame of the priors shared by all targets with their location and scale; dispersion, a data frame with the log mean count and prior location of each target, or NULL; trend, the fitted loess model, or NULL; n, the number of targets that contributed to the estimation; and data, the per-target estimates the priors were built from, see below.

Estimates used to build the priors

The data attribute holds the estimates that went into each prior, so that the priors can be inspected and plotted with other tools than plot.seqwrap_priors(). It is a list with two elements.

- estimates, a data frame with one row per target and prior, with the columns target, class ("fixef" or "ranef_sd"), coef and estimate. For fixed effects estimate is the coefficient of the term in the target's model. For random effects it is the estimated standard deviation, one row per standard deviation term within the grouping variable.
- dispersion, a data frame with the columns target, log_mu and dispersion holding the log mean count and the estimated log dispersion of every target used to estimate the dispersion trend, or NULL when no dispersion prior was built.

The location and scale of each prior is found in the common and dispersion attributes, and the loess fit in trend.

See Also

dispersion_evaluation() to record the dispersion during the first run, and seqwrap_compose() for the targetdata argument.

Examples

```
library(seqwrap)

if (requireNamespace("glmmTMB", quietly = TRUE)) {
  dat <- simcounts(n_genes = 20, n_samples = 30, clusters = 15)
  counts <- dat$data
  metadata <- dat$metadata
  metadata$ln_libsize <- log(colSums(counts[, -1]))

  # First run, without priors, recording the dispersion of each target
  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",
  eval_fun = dispersion_evaluation
)

results <- seqwrap(container, cores = 1, verbose = FALSE)

# Priors pooled across targets, one data frame per target
priors <- seqwrap_priors(results, data = counts)
priors
priors[[1]]


# Second run, with the priors passed as target-specific data
container_prior <- seqwrap_compose(
  modelfun = glmmTMB::glmmTMB,
  arguments = alist(
    formula = y ~ x + (1 | cluster) + offset(ln_libsize),
    family = glmmTMB::nbinom2,
    priors = data.frame(prior = prior, class = class, coef = coef)
  ),
  data = counts,
  metadata = metadata,
  samplename = "sample",
  targetdata = priors
)

results_prior <- seqwrap(container_prior, subset = 1:5,
                         return_models = TRUE, cores = 1,
                         verbose = FALSE)

# The fitted model carries the priors it was given
results_prior@models[[1]]$modelInfo$priors

}
```

`seqwrap_summarise` *Summarise seqwrapResults objects*

Description

Summarise seqwrapResults objects

Usage

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

Arguments

x	A seqwrapResults object
summaries	Logical, should summaries be combined?
evaluations	Logical, should evaluations be combined?
errors	Logical, should errors and warnings be combined? When TRUE the returned list gains an errors element with one row per condition raised.
drop_warnings	Should targets whose fitting, summary or evaluation raised a warning be removed from the combined results? FALSE (the default) keeps every target. TRUE removes targets that warned at any stage. A character vector selects stages, one or more of "fit", "summary" and "evaluation". See Details.
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.

Targets that raised warnings:

drop_warnings defaults to FALSE, so a target that produced a warning is still reported. This is deliberate. Warnings from fitting algorithms differ widely in severity: a singular fit from lme4 means a variance component has reached the boundary of the parameter space, which often leaves inference on the fixed effects intact, whereas a non-positive-definite Hessian from glmmTMB means the standard errors cannot be trusted. Treating both as grounds for removal discards usable results. Removing them silently is also a form of selection. Targets that warn are not a random subset: low count and low variance targets produce singular fits far more often than others, so dropping them biases the remaining set and changes the number of tests carried into any multiplicity correction.

For a graded alternative, generic_evaluation() reports converged and singular per target, which can be joined onto the summaries and filtered explicitly. When drop_warnings is used, the identifiers of the removed targets are returned in the dropped element so that the choice stays visible and reversible.

Value

A list (invisibly) with up to four elements: summaries, combined parameter summaries from each model; evaluations, combined diagnostics from each model; errors, one row per error or warning with columns target, stage, type and message; and dropped, the identifiers of any targets

removed by drop_warnings. Entries are omitted when the corresponding slot of x is empty or the user disables them via the summaries, evaluations and errors 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.

Usage

```
swcontainer(
  .data = list(),
  modelfun = NULL,
  arguments = NULL,
  data = NULL,
  rownames = logical(0),
  metadata = data.frame(),
  targetdata = NULL,
  samplename = character(0),
  additional_vars = character(0),
```

```
  summary_fun = NULL,
  eval_fun = NULL,
  exported = list(),
  model_print = character(0),
  arguments_print = character(0)
)
```

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