Package 'growkar'

September 11, 2026

Title SummarizedExperiment Infrastructure for Microbial Growth Phenotyping

Version 0.99.4

Description growkar provides Bioconductor-native infrastructure for high-throughput microbial growth phenotyping from plate-based assays such as optical density (OD) measurements. The package is built around SummarizedExperiment as its canonical data model, with OD measurements in assays, timepoint annotations in rowData, sample annotations in colData, and derived phenotypes in metadata. growkar supports data import, quality control, normalization, empirical growth-rate estimation, exponential-phase detection, and parametric modeling of growth dynamics. These workflows enable extraction of biologically meaningful phenotypes including lag time, growth rate, doubling time, and carrying capacity, and allow growth-derived phenotypes to be integrated with other Bioconductor analyses in microbial systems biology and multi-omics studies. The data-structure and analysis layers depend only on core Bioconductor infrastructure; graphing is optional and provided by 'plot_*' functions that are enabled when the suggested graphics package is installed.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.6.0)

Imports methods, BiocBaseUtils, grDevices, S4Vectors, SummarizedExperiment, tidySummarizedExperiment, dplyr, purrr, rlang, stats, tibble, tidyr, utils

Suggests ggplot2, knitr, remotes, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/sethiyap/growkar

BugReports https://github.com/sethiyap/growkar/issues

biocViews Software, SystemsBiology, ExperimentalDesign, Visualization, Infrastructure, DataImport, QualityControl, Normalization

Config/roxygen2/version 8.0.0

git_url https://git.bioconductor.org/packages/growkar

git_branch devel

git_last_commit 6c1fa59

git_last_commit_date 2026-09-06

Repository Bioconductor 3.24

Date/Publication 2026-09-10

Author Pooja Sethiya [aut, cre] (ORCID:
<https://orcid.org/0000-0001-6584-3912>)

Maintainer Pooja Sethiya <poojasethiya24@gmail.com>

Contents

growkar-package *growkar: High-Throughput Microbial Phenotyping from Growth Assays*

Description

growkar uses a single Bioconductor data container, the S4 class GrowthExperiment (an extension of SummarizedExperiment::SummarizedExperiment), and a single model-result class, GrowthFit. Tidy manipulation and display of these objects are delegated to tidySummarizedExperiment rather than reimplemented, and graphing is provided by optional plot_*() helpers.

Details

Scalar argument checks and optional-package checks are taken from BiocBaseUtils rather than reimplemented.

The import from tidySummarizedExperiment is deliberate: that package registers the as_tibble() method used to convert a GrowthExperiment to long form, and importing one of its exports loads its namespace (and hence registers those methods) whenever growkar is loaded.

Author(s)

Maintainer: Pooja Sethiya <poojasethiya24@gmail.com> (ORCID)

Authors:

- Pooja Sethiya <poojasethiya24@gmail.com> (ORCID)

See Also

Useful links:

- https://github.com/sethiyap/growkar
- Report bugs at https://github.com/sethiyap/growkar/issues

---

as_tidy_growth_data *Import Growth Data into Canonical Tidy Format*

Description

Import adapter that converts vendor plate-reader exports and user tables into the canonical sample/time/od columns expected by GrowthExperiment().

Usage

```
as_tidy_growth_data(
  data,
  sample_col = "sample",
  time_col = "time",
  od_col = "od",
  sample_sep = "_"
)
```

Arguments

data	A data frame, tibble, SummarizedExperiment, or object coercible to a tibble.
sample_col	Name of the sample column for long-form input.
time_col	Name of the time column.
od_col	Name of the optical density column.
sample_sep	Separator used to infer metadata columns from sample names in wide data or in long data when sample is present.

Details

This function is an *import* layer, not an analysis container. Within growkar the canonical container is SummarizedExperiment::SummarizedExperiment. Once data are in a SummarizedExperiment, tidy manipulation and display are better handled by the tidyomics stack: install tidySummarizedExperiment and use dplyr/tidyr/ggplot2 verbs directly on the object rather than round tripping through this function. See vignette("growkar-introduction", package = "growkar") for worked interoperability examples.

Wide input is expected to contain time in the first column and sample names in the remaining column names. Tidy input is expected to contain at least sample, time, and od. When replicate identifiers are encoded in sample names, use a consistent suffix such as _R1 or _1 so replicate metadata can be inferred reliably. Common machine-exported column labels such as Time [s], Time [h], Sample, Well, OD600, OD 600, and Absorbance 600 are detected automatically where possible, which helps when importing exports from Agilent microplate readers, BioTek Cytation instruments, LogPhase 600, and similar OD600-based plate-reader workflows.

Value

A tibble containing at least sample, time, and od. Additional metadata columns are preserved where available.

Examples

```
data(yeast_growth_data)
tidy_growth <- as_tidy_growth_data(yeast_growth_data)
head(tidy_growth)

se <- GrowthExperiment(yeast_growth_data)
head(as_tidy_growth_data(se))
```

---

augment_growth_fit *Augment Observed Data with Fitted Values*

Description

Return the observed data used for fitting together with the fitted values from a GrowthFit object.

Usage

```
augment_growth_fit(fit, ...)

## S4 method for signature 'GrowthFit'
augment_growth_fit(fit, ...)
```

Arguments

fit	A GrowthFit object returned by fit_growth_curve().
...	Arguments passed to methods.

Value

A tidy tibble containing the observed columns plus .fitted.

Examples

```
data(yeast_growth_data)
tidy_growth <- as_tidy_growth_data(yeast_growth_data)
sample_id <- unique(tidy_growth$sample)[1]
fit <- fit_growth_curve(dplyr::filter(tidy_growth, sample == sample_id))
head(augment_growth_fit(fit))
```

---

compute_doubling_time *Compute Doubling Time*

Description

Compute doubling time from a growth rate estimate.

Usage

```
compute_doubling_time(mu)
```

Arguments

mu	Numeric vector of specific growth-rate estimates. In growkar, mu is the slope of log(od) versus time.

Details

This is a low-level helper for converting one or more growth-rate estimates into doubling times. For a convenience summary across all samples, use summarize_growth_metrics().

Value

A numeric vector equal to log(2) / mu, with NA returned when mu <= 0.

Examples

```
compute_doubling_time(c(0.4, 0.7, NA, -0.1))
```

`compute_growth_rate` *Compute Growth Rate*

Description

Estimate per-sample growth rate from tidy growth data.

Usage

```
compute_growth_rate(
  data,
  method = c("rolling_window", "defined_interval", "rule_based"),
  interval = NULL,
  select_replicates = NULL,
  average_replicates = FALSE,
  window_size = 5,
  min_od = 0.02,
  first_timepoint = NULL
)
```

Arguments

`data`	Growth curve data in tidy, wide, or `SummarizedExperiment` format. All inputs are standardized internally to the canonical `SummarizedExperiment` representation before analysis.
`method`	Estimation method. One of `"rolling_window"`, `"defined_interval"`, or `"rule_based"`.
`interval`	Interval definition for `defined_interval`. Supply either a numeric vector of length two to use the same interval for all samples, or a data frame containing sample-specific start and end times. Interval tables may use columns named `sample`, `start_time`, and `end_time`, or the first three columns may be sample, start time, and end time.
`select_replicates`	Optional character vector of replicate IDs to retain before estimation. When `NULL`, all replicates are retained.
`average_replicates`	Logical; if `TRUE`, average replicates before estimation when replicate metadata are available.
`window_size`	Rolling window size for `rolling_window`.
`min_od`	Minimum OD retained when fitting log-linear models.
`first_timepoint`	Reference time used by the `"rule_based"` method.

Details

The returned `mu` column is the specific growth rate, estimated as the slope of `log(od)` versus time over the selected interval or window.

Available methods:

- `rolling_window`: scans rolling windows across the time series and selects the window with the strongest positive log-linear slope.

- `defined_interval`: fits the growth rate over a user-supplied start and end time interval. Supply either one interval for all samples or a per-sample interval table.
- `rule_based`: starts from a reference OD, finds the nearest successive OD doublings, and estimates growth from the interval between those doubling anchors.

For the `rule_based` method, `growkar` first picks a reference OD at the earliest time point, or at `first_timepoint` when supplied. It then finds the observations nearest to approximately 2x and 4x that reference OD. The growth rate is estimated from the log-linear slope between those two doubling anchors. This provides a simple empirical approximation when users want a heuristic interval rather than a rolling search or manually defined bounds.

Value

A tibble with `sample`, `mu`, `start_time`, `end_time`, `r_squared`, and `method`. Here `mu` is the estimated specific growth rate. Additional diagnostic columns describe the number of points used and whether the estimate required degraded fallback behavior. Here `degraded = TRUE` indicates that the preferred estimation path could not be used cleanly, so the result comes from a reduced-window or fallback path and should be interpreted with extra caution.

Examples

```
data(yeast_growth_data)
tidy_growth <- as_tidy_growth_data(yeast_growth_data)
sample_id <- unique(tidy_growth$sample)[1]
compute_growth_rate(
  dplyr::filter(tidy_growth, sample == sample_id),
  method = "rolling_window"
)
compute_growth_rate(
  tidy_growth,
  method = "defined_interval",
  interval = c(2, 6)
)
```

---

detect_exponential_phase

Detect Candidate Exponential-Phase Windows

Description

Identify rolling windows with the strongest evidence for exponential growth. For multi-sample input, detection is performed separately for each sample. When `average_replicates = TRUE`, replicate trajectories are averaged first and exponential-phase detection is then run on the averaged curves.

Usage

```
detect_exponential_phase(
  data,
  select_replicates = NULL,
  average_replicates = FALSE,
  window_size = 5,
  min_od = 0.02
)
```

Arguments

data	Growth data for one sample or multiple samples in tidy, wide, or SummarizedExperiment format.
select_replicates	Optional character vector of replicate IDs to retain before detection. When NULL, all replicates are retained.
average_replicates	Logical; if TRUE, average replicate trajectories before detection when replicate metadata are available.
window_size	Number of observations in each rolling window.
min_od	Minimum OD retained for log-linear fitting.

Value

A tibble with sample, start_time, end_time, slope, and r_squared, ranked by highest positive slope and then r_squared. The returned tibble also includes rank, n_points, selection_reason, and degraded metadata describing how the candidate windows were selected. For multi-sample input, windows are returned for each sample. Here degraded = TRUE indicates that detection required fallback behavior, such as reducing the requested window size or returning a placeholder result because too few usable observations were available.

Examples

```
data(yeast_growth_data)
tidy_growth <- as_tidy_growth_data(yeast_growth_data)
sample_id <- unique(tidy_growth$sample)[1]
phase_tbl <- detect_exponential_phase(dplyr::filter(tidy_growth, sample == sample_id))
head(phase_tbl)
averaged_phase_tbl <- detect_exponential_phase(
  tidy_growth,
  average_replicates = TRUE
)
head(averaged_phase_tbl)
```

---

extract_params *Extract Fitted Growth Parameters*

Description

Extract fitted coefficients and model-derived quantities, such as the asymptote and the model-based doubling time, from a GrowthFit object.

Usage

```
extract_params(fit, ...)

## S4 method for signature 'GrowthFit'
extract_params(fit, ...)
```

Arguments

fit	A GrowthFit object returned by fit_growth_curve().
...	Arguments passed to methods.

Value

A tibble with one row containing sample, model, asymptote, r, t0, and doubling_time_model.

Examples

```
data(yeast_growth_data)
tidy_growth <- as_tidy_growth_data(yeast_growth_data)
sample_id <- unique(tidy_growth$sample)[1]
fit <- fit_growth_curve(dplyr::filter(tidy_growth, sample == sample_id))
extract_params(fit)
```

---

fit_growth_curve *Fit a Parametric Growth Model*

Description

Fit a logistic or Gompertz growth model to a single sample.

Usage

```
fit_growth_curve(data, model = c("logistic", "gompertz"))
```

Arguments

data	Growth curve data for one sample in tidy, wide, or SummarizedExperiment format.
model	Model type: "logistic" or "gompertz".

Value

A GrowthFit object. Failed fits return the same class with converged = FALSE and a status slot describing the failure, so that plate-wide fitting is not aborted by a single problematic sample.

Examples

```
data(yeast_growth_data)
tidy_growth <- as_tidy_growth_data(yeast_growth_data)
sample_id <- unique(tidy_growth$sample)[1]
fit <- fit_growth_curve(
  dplyr::filter(tidy_growth, sample == sample_id),
  model = "logistic"
)
fit
```

fit_growth_models *Fit and Store Growth Models*

Description

Fit logistic or Gompertz growth models for tidy, wide, or SummarizedExperiment input. When data is a SummarizedExperiment, fitted model objects and extracted parameter summaries are stored in metadata(), and the updated object is returned.

Usage

```
fit_growth_models(
  data,
  model = c("logistic", "gompertz"),
  store_metadata = TRUE
)
```

Arguments

data	Growth curve data in tidy, wide, or SummarizedExperiment format.
model	Model type: "logistic" or "gompertz".
store_metadata	Logical; when TRUE and data is a SummarizedExperiment, store fitted models and parameter summaries in metadata().

Value

A tibble with fitted models for non-SummarizedExperiment input, or an updated SummarizedExperiment when data is a SummarizedExperiment.

Examples

```
data(yeast_growth_data)
se <- GrowthExperiment(yeast_growth_data)
se <- fit_growth_models(se, model = "logistic")
S4Vectors::metadata(se)$growth_model_parameters
```

fit_growth_plate *Fit Growth Models Across a Plate*

Description

Split tidy growth data by sample and fit a parametric model to each sample.

Usage

```
fit_growth_plate(data, model = c("logistic", "gompertz"))
```

Arguments

data	Growth curve data in tidy, wide, or SummarizedExperiment format.
model	Model type: "logistic" or "gompertz".

Value

A tibble with `sample` and a list-column of GrowthFit objects.

Examples

```
data(yeast_growth_data)
fits <- fit_growth_plate(yeast_growth_data, model = "logistic")
fits
```

`GrowthExperiment` *Construct a* `GrowthExperiment`

Description

Build a GrowthExperiment object, the canonical `growkar` container, from tidy or wide growth curve data. `GrowthExperiment` extends SummarizedExperiment::SummarizedExperiment, so the result can be used directly by any Bioconductor package.

Usage

```
GrowthExperiment(
  data,
  metrics = NULL,
  sample_col = "sample",
  time_col = "time",
  od_col = "od",
  sample_sep = "_"
)
```

Arguments

`data`	Growth curve data in tidy or wide format, or a `SummarizedExperiment`.
`metrics`	Optional data frame of precomputed sample-level metrics to store in `metadata(x)$growth_metrics`. Metrics computed by `growkar` itself are normally attached with `growth_metrics()` instead.
`sample_col`	Name of the sample column for long-form input.
`time_col`	Name of the time column.
`od_col`	Name of the optical density column.
`sample_sep`	Separator used to infer metadata columns from sample names.

Details

The resulting object stores time points in `rowData(x)$time`, sample-level metadata in `colData(x)`, optical density measurements in `assay(x, "od")`, and derived results in `metadata(x)`.

Coercion methods are also provided, so `as(x, "GrowthExperiment")` works for a `data.frame`, `tibble`, `matrix`, or existing `SummarizedExperiment`, and `as(x, "SummarizedExperiment")` is available in the other direction. Use the constructor when you need to control column-name resolution; use `as()` for the default behaviour.

Value

A GrowthExperiment object with one assay named od.

See Also

GrowthExperiment

Examples

```
data(yeast_growth_data)

ge <- GrowthExperiment(yeast_growth_data)
ge

# Equivalent, using standard coercion.
as(yeast_growth_data, "GrowthExperiment")
```

GrowthExperiment-class

Growth Assay Experiment

Description

GrowthExperiment is the data container used throughout growkar. It is an S4 class that directly extends SummarizedExperiment::SummarizedExperiment, adding a validity method that enforces the canonical growth-assay layout:

Details

- an assay named od holding optical density measurements, with timepoints in rows and samples in columns;
- a numeric time column in rowData();
- sample annotations in colData();
- derived results (growth metrics, exponential-phase windows, model fits) in metadata().

Because the class extends SummarizedExperiment, every method written for SummarizedExperiment — including subsetting, assay(), rowData(), colData(), metadata(), and the tidyomics verbs provided by tidySummarizedExperiment — applies unchanged, and as(x, "SummarizedExperiment") is always available for handing objects to other Bioconductor packages.

Construct objects with GrowthExperiment(), or coerce with as(x, "GrowthExperiment") from a data.frame, tibble, matrix, or existing SummarizedExperiment.

Value

A GrowthExperiment object.

See Also

GrowthExperiment(), validate_growth_experiment()

Examples

```
data(yeast_growth_data)

ge <- GrowthExperiment(yeast_growth_data)
ge

# Standard coercion in both directions.
se <- as(ge, "SummarizedExperiment")
identical(ge, as(se, "GrowthExperiment"))

# Inherited SummarizedExperiment methods apply unchanged.
SummarizedExperiment::assayNames(ge)
dim(ge[, 1:3])
```

GrowthExperiment-coerce *Coerce to a* GrowthExperiment

Description

Standard S4 coercion methods for GrowthExperiment. Coercion in the opposite direction, `as(x, "SummarizedExperiment")`, is inherited from the class hierarchy and always available.

Arguments

`from`	A `data.frame`, `matrix`, or `SummarizedExperiment` to coerce.

Value

A GrowthExperiment object.

Examples

```
data(yeast_growth_data)

ge <- as(yeast_growth_data, "GrowthExperiment")
ge

se <- as(ge, "SummarizedExperiment")
as(se, "GrowthExperiment")
```

GrowthFit-accessors *Accessors for* GrowthFit *Objects*

Description

Extract individual components of a GrowthFit object. These accessors are the supported interface; slots should not be accessed directly with `@`.

Usage

```
fit_sample(x, ...)

fit_model(x, ...)

fit_status(x, ...)

fit_converged(x, ...)

fit_data(x, ...)

## S4 method for signature 'GrowthFit'
fit_sample(x, ...)

## S4 method for signature 'GrowthFit'
fit_model(x, ...)

## S4 method for signature 'GrowthFit'
fit_status(x, ...)

## S4 method for signature 'GrowthFit'
fit_converged(x, ...)

## S4 method for signature 'GrowthFit'
fit_data(x, ...)
```

Arguments

x	A GrowthFit object.
...	Arguments passed to methods.

Value

- `fit_sample()`: a character scalar naming the fitted sample.
- `fit_model()`: a character scalar naming the fitted model.
- `fit_status()`: a character scalar describing the fit outcome.
- `fit_converged()`: a logical scalar.
- `fit_data()`: a tibble of the observed data used for fitting.

Examples

```
data(yeast_growth_data)
tidy_growth <- as_tidy_growth_data(yeast_growth_data)
sample_id <- unique(tidy_growth$sample)[1]
fit <- fit_growth_curve(dplyr::filter(tidy_growth, sample == sample_id))

fit_sample(fit)
fit_model(fit)
fit_status(fit)
fit_converged(fit)
head(fit_data(fit))
```

GrowthFit-model-methods

Model Components of a GrowthFit *Object*

Description

Methods for the standard modelling generics stats::coef(), stats::fitted(), stats::residuals(), and stats::nobs().

Usage

```
## S4 method for signature 'GrowthFit'
coef(object, ...)

## S4 method for signature 'GrowthFit'
fitted(object, ...)

## S4 method for signature 'GrowthFit'
residuals(object, ...)

## S4 method for signature 'GrowthFit'
nobs(object, ...)
```

Arguments

object	A GrowthFit object.
...	Unused, present for generic compatibility.

Value

- coef(): a named numeric vector of fitted coefficients.
- fitted(): a numeric vector of fitted optical densities.
- residuals(): a numeric vector of observed minus fitted values.
- nobs(): an integer count of observations used for fitting.

Examples

```
data(yeast_growth_data)
tidy_growth <- as_tidy_growth_data(yeast_growth_data)
sample_id <- unique(tidy_growth$sample)[1]
fit <- fit_growth_curve(dplyr::filter(tidy_growth, sample == sample_id))

coef(fit)
head(fitted(fit))
head(residuals(fit))
nobs(fit)
```

growth_assay *Growth Assay Matrix*

Description

Access the canonical OD assay from a SummarizedExperiment.

Usage

```
growth_assay(x)
```

Arguments

x	A SummarizedExperiment compatible with growkar.

Value

A numeric matrix with time points in rows and samples in columns.

Examples

```
data(yeast_growth_data)
se <- GrowthExperiment(yeast_growth_data)
growth_assay(se)
```

growth_metrics *Compute and Store Growth Metrics*

Description

Compute growth metrics for tidy, wide, or SummarizedExperiment input. When data is a SummarizedExperiment, the resulting metrics table is stored in metadata(data)$growth_metrics, and the updated object is returned.

Usage

```
growth_metrics(
  data,
  method = c("rolling_window", "defined_interval", "rule_based"),
  average_replicates = FALSE,
  select_replicates = NULL,
  comparison_col = NULL,
  compare_to = NULL,
  error = c("se", "sd"),
  pvalue_method = c("t_test", "wilcox"),
  store_metadata = TRUE,
  ...
)
```

Arguments

data	Growth curve data in tidy, wide, or SummarizedExperiment format.
method	Estimation method passed to compute_growth_rate().
average_replicates	Logical; if TRUE, average replicate trajectories before computing metrics. When select_replicates is NULL, all available replicates are averaged.
select_replicates	Optional character vector of replicate IDs to retain before summarizing. When NULL, all replicates are retained.
comparison_col	Optional column used to summarize replicate-level doubling times and compute group-wise statistics. Typical values are "condition" or "sample".
compare_to	Optional reference group in comparison_col used for doubling-time p-value comparisons.
error	Statistic used for doubling-time error bars and summary output. One of "se" or "sd".
pvalue_method	Statistical test used when compare_to is supplied. One of "t_test" or "wilcox".
store_metadata	Logical; when TRUE and data is a SummarizedExperiment, store the derived metrics and analysis parameters in metadata().
...	Additional arguments passed to compute_growth_rate().

Value

A tidy tibble for non-SummarizedExperiment input, or an updated SummarizedExperiment when data is a SummarizedExperiment.

Examples

```
data(yeast_growth_data)
se <- GrowthExperiment(yeast_growth_data)
se <- growth_metrics(se, method = "rolling_window", average_replicates = TRUE)
S4Vectors::metadata(se)$growth_metrics
```

---

growth_model_fits *Stored Growth Model Fits*

Description

Retrieve fitted growth model objects stored in metadata().

Usage

```
growth_model_fits(x)
```

Arguments

x	A SummarizedExperiment compatible with growkar.

Value

A tibble of stored model fits, or NULL if no fits have been stored.

Examples

```
data(yeast_growth_data)
se <- GrowthExperiment(yeast_growth_data)
se <- fit_growth_models(se, model = "logistic")
growth_model_fits(se)
```

`nlsOrNULL-class` *Parametric Growth Model Fit*

Description

A formal S4 representation of a parametric growth model fitted to a single sample by `fit_growth_curve()`. `growkar` uses SummarizedExperiment::SummarizedExperiment as its only data container; the `GrowthFit` class represents a *model result*, not assay data, and is stored in `metadata()` alongside the experiment it was derived from.

Details

Fits that fail are returned as valid `GrowthFit` objects with `converged = FALSE` and a `status` slot describing the failure, so that plate-wide fitting never aborts on a single problematic sample.

Value

An object of class `GrowthFit`.

Slots

`sample` Character scalar identifying the fitted sample.

`model` Character scalar naming the model, `"logistic"` or `"gompertz"`.

`fit` The underlying `stats::nls()` object, or `NULL` when the fit failed.

`coefficients` Named numeric vector of fitted coefficients `K`, `r`, and `t0`.

`fitted` A `data.frame` with columns `time` and `.fitted`.

`residuals` Numeric vector of observed minus fitted values.

`rss` Residual sum of squares.

`aic` Akaike information criterion.

`bic` Bayesian information criterion.

`data` A `data.frame` of the observed data used for fitting.

`converged` Logical scalar indicating whether the fit converged.

`status` Character scalar describing the fit outcome, one of `"converged"`, `"insufficient_points"`, `"flat_curve"`, or `"fit_failed"`.

`message` Character scalar with a diagnostic message, or `NA` when the fit converged.

`n_points` Integer number of observations used for fitting.

`starting_values` Named numeric vector of starting values.

`bounds` Named numeric vector of lower bounds.

See Also

`fit_growth_curve()`, `extract_params()`, `augment_growth_fit()`

Examples

```
data(yeast_growth_data)
tidy_growth <- as_tidy_growth_data(yeast_growth_data)
sample_id <- unique(tidy_growth$sample)[1]
fit <- fit_growth_curve(dplyr::filter(tidy_growth, sample == sample_id))
fit
isVirtualClass("GrowthFit")
```

---

phase_windows *Compute and Store Exponential-Phase Windows*

Description

Detect exponential-phase windows for tidy, wide, or SummarizedExperiment input. When data is a SummarizedExperiment, the detected windows are stored in metadata(data)$exponential_phase_windows, and the updated object is returned.

Usage

```
phase_windows(
  data,
  window_size = 4L,
  min_od = 0.02,
  average_replicates = FALSE,
  select_replicates = NULL,
  store_metadata = TRUE
)
```

Arguments

data	Growth data for one sample or multiple samples in tidy, wide, or SummarizedExperiment format.
window_size	Number of observations in each rolling window.
min_od	Minimum OD retained for log-linear fitting.
average_replicates	Logical; if TRUE, average replicate trajectories before detection when replicate metadata are available.
select_replicates	Optional character vector of replicate IDs to retain before detection. When NULL, all replicates are retained.
store_metadata	Logical; when TRUE and data is a SummarizedExperiment, store the detected windows and analysis parameters in metadata().

Value

A tibble for non-SummarizedExperiment input, or an updated SummarizedExperiment when data is a SummarizedExperiment.

Examples

```
data(yeast_growth_data)
se <- GrowthExperiment(yeast_growth_data)
se <- phase_windows(se, average_replicates = TRUE)
S4Vectors::metadata(se)$exponential_phase_windows
```

`plot_doubling_time` *Plot Doubling Time Summary*

Description

Plot replicate-based doubling-time summaries as a bar chart with error bars and optional bracketed p-value annotations.

Usage

```
plot_doubling_time(
  data,
  comparison_col = NULL,
  compare_to = NULL,
  exclude_groups = NULL,
  select_replicates = NULL,
  average_replicates = FALSE,
  method = c("rolling_window", "defined_interval", "rule_based"),
  error = c("se", "sd"),
  pvalue_method = c("t_test", "wilcox"),
  palette_name = "all_colors",
  custom_colors = NULL,
  ...
)
```

Arguments

`data`	Growth curve data in tidy, wide, or `SummarizedExperiment` format. Inputs are standardized internally to the canonical `SummarizedExperiment` representation before plotting.
`comparison_col`	Column used to group replicate-level doubling times. Typical values are `"condition"` or `"sample"`.
`compare_to`	Optional reference group used for p-value comparisons.
`exclude_groups`	Optional character vector of groups in `comparison_col` to remove from the plotted summary.
`select_replicates`	Optional character vector of replicate IDs to retain before computing doubling times.
`average_replicates`	Logical; if `TRUE`, compute doubling time from averaged replicate trajectories and plot one bar per averaged sample.
`method`	Estimation method passed to `summarize_growth_metrics()`.
`error`	Statistic used for error bars. One of `"se"` or `"sd"`.

pvalue_method	Statistical test used when compare_to is supplied. One of "t_test" or "wilcox".
palette_name	Name of the qualitative palette used by select_palette().
custom_colors	Optional character vector of colours. When supplied, these user-defined colours are used instead of the selected palette.
...	Additional arguments passed to summarize_growth_metrics().

Value

A ggplot2 object. Requires the suggested package ggplot2.

Examples

```
data(yeast_growth_data)
plot_doubling_time(
  yeast_growth_data,
  comparison_col = "condition",
  compare_to = "Cg",
  palette_name = "Dark2"
)
```

---

plot_fitted_curve *Plot a Fitted Growth Curve*

Description

Overlay observed points and fitted values from a parametric growth model.

Usage

```
plot_fitted_curve(
  fit,
  model = c("logistic", "gompertz"),
  select_replicates = NULL,
  average_replicates = FALSE,
  colour_col = "sample",
  facet_col = NULL,
  palette_name = "all_colors",
  custom_colors = NULL
)
```

Arguments

fit	A GrowthFit object, or growth curve data in tidy or wide format.
model	Model type used when fit is raw data rather than a GrowthFit object.
select_replicates	Optional character vector of replicate IDs to retain before fitting. When NULL, all replicates are retained.
average_replicates	Logical; if TRUE, average replicate trajectories before fitting and plotting.

colour_col	Name of the column mapped to colour when plotting from raw data.
facet_col	Optional name of a column used for faceting when plotting from raw data.
palette_name	Name of the qualitative palette used by select_palette(). Supported values include "all_colors" (combined palette), "Accent" (8), "Dark2" (8), "Paired" (12), "Pastel1" (9), "Pastel2" (8), "Set1" (9), "Set2" (8), and "Set3" (12).
custom_colors	Optional character vector of colours. When supplied, these user-defined colours are used instead of the selected palette.

Value

A ggplot2 object. Requires the suggested package ggplot2.

Examples

```
data(yeast_growth_data)
tidy_growth <- as_tidy_growth_data(yeast_growth_data)
sample_id <- unique(tidy_growth$sample)[1]
fit <- fit_growth_curve(dplyr::filter(tidy_growth, sample == sample_id))
plot_fitted_curve(fit)
```

---

plot_growth_curve *Plot Growth Curves*

Description

Plot observed growth curves from growth data standardized internally to a canonical SummarizedExperiment.

Usage

```
plot_growth_curve(
  data,
  average_replicates = FALSE,
  colour_col = "sample",
  facet_col = NULL,
  palette_name = "all_colors",
  custom_colors = NULL
)
```

Arguments

data	Growth curve data in tidy, wide, or SummarizedExperiment format.
average_replicates	Logical; if TRUE, average replicate trajectories before plotting when a replicate column is available.
colour_col	Name of the column mapped to colour.
facet_col	Optional name of a column used for faceting.
palette_name	Name of the qualitative palette used by select_palette(). Supported values include "all_colors" (combined palette), "Accent" (8), "Dark2" (8), "Paired" (12), "Pastel1" (9), "Pastel2" (8), "Set1" (9), "Set2" (8), and "Set3" (12).
custom_colors	Optional character vector of colours. When supplied, these user-defined colours are used instead of the selected palette.

Value

A ggplot2 object. Requires the suggested package ggplot2.

Examples

```
data(yeast_growth_data)
plot_growth_curve(
  yeast_growth_data,
  average_replicates = TRUE,
  colour_col = "condition",
  palette_name = "Dark2"
)
```

---

plot_growth_curve_facets

Plot Growth Curves as Facets

Description

Plot observed growth curves with replicates averaged and grouped into separate sample-family facets. For condition labels such as KN99(100) and CM2448(50), this produces one facet for KN99, one for CM2448, and so on, with the different conditions shown within each facet. By default, when labels include parenthesized dose values such as (0) or (100), those shared values are used for colour mapping across facets.

Usage

```
plot_growth_curve_facets(
  data,
  colour_col = NULL,
  palette_name = "all_colors",
  custom_colors = NULL
)
```

Arguments

data	Growth curve data in tidy, wide, or SummarizedExperiment format.
colour_col	Name of the column mapped to colour. When NULL, the plot uses a derived dose label from condition/sample when available and otherwise falls back to condition or sample.
palette_name	Name of the qualitative palette used by select_palette(). Supported values include "all_colors" (combined palette), "Accent" (8), "Dark2" (8), "Paired" (12), "Pastel1" (9), "Pastel2" (8), "Set1" (9), "Set2" (8), and "Set3" (12).
custom_colors	Optional character vector of colours. When supplied, these user-defined colours are used instead of the selected palette.

Value

A ggplot2 object. Requires the suggested package ggplot2.

Examples

```
data(yeast_growth_data)
plot_growth_curve_facets(
  yeast_growth_data,
  palette_name = "Dark2"
)
```

`sample_data` *Sample Metadata*

Description

Access sample-level metadata from a `SummarizedExperiment`.

Usage

```
sample_data(x)
```

Arguments

`x`	A `SummarizedExperiment` compatible with `growkar`.

Value

A tibble containing `colData(x)`.

Examples

```
data(yeast_growth_data)
se <- GrowthExperiment(yeast_growth_data)
sample_data(se)
```

`select_palette` *select_palette*

Description

allows to select contrasting colors from 8 different palettes contributing a total of 74 colors

Usage

```
select_palette(num_of_colors, palette_name = "all_colors")
```

Arguments

num_of_colors	numeric, number of colors to be obtained from palette
palette_name	character,name palettes Default: 'all_colors' Supported individual palettes are:

- Accent (8)
- Dark2 (8)
- Paired (12)
- Pastel1 (9)
- Pastel2 (8)
- Set1 (9)
- Set2 (8)
- Set3 (12)

Details

select_palette is particularly useful as individual palette from many color palettes contain less than 10 colors. The palettes themselves come from grDevices::palette.colors(), which ships with R, so no colour package is needed.

Value

a vector of colors from selected palette

Examples

```
## Not run:
if(interactive()){
 # colors combined from all palettes
 select_palette(num_of_colors = 10)

 # colors from Accent palette
 select_palette(num_of_colors = 4, palette_name = "Accent")
 }

## End(Not run)
```

---

show,GrowthFit-method *Display a* GrowthFit *Object*

Description

Compact one-line summary of a fitted growth model.

Usage

```
## S4 method for signature 'GrowthFit'
show(object)
```

Arguments

object	A GrowthFit object.

Value

object, invisibly. Called for its side effect of printing.

Examples

```
data(yeast_growth_data)
tidy_growth <- as_tidy_growth_data(yeast_growth_data)
sample_id <- unique(tidy_growth$sample)[1]
fit <- fit_growth_curve(dplyr::filter(tidy_growth, sample == sample_id))
fit
```

---

summarize_growth_metrics

Summarize Growth Metrics

Description

Compute growth rate and doubling time per sample.

Usage

```
summarize_growth_metrics(
  data,
  method = c("rolling_window", "defined_interval", "rule_based"),
  average_replicates = FALSE,
  select_replicates = NULL,
  comparison_col = NULL,
  compare_to = NULL,
  error = c("se", "sd"),
  pvalue_method = c("t_test", "wilcox"),
  ...
)
```

Arguments

data	Growth curve data in tidy, wide, or SummarizedExperiment format.
method	Estimation method passed to compute_growth_rate().
average_replicates	Logical; if TRUE, average replicate trajectories before computing metrics. When select_replicates is NULL, all available replicates are averaged.
select_replicates	Optional character vector of replicate IDs to retain before summarizing. When NULL, all replicates are retained.
comparison_col	Optional column used to summarize replicate-level doubling times and compute group-wise statistics. Typical values are "condition" or "sample".
compare_to	Optional reference group in comparison_col used for doubling-time p-value comparisons.
error	Statistic used for doubling-time error bars and summary output. One of "se" or "sd".
pvalue_method	Statistical test used when compare_to is supplied. One of "t_test" or "wilcox".
...	Additional arguments passed to compute_growth_rate().

Details

This is the convenience wrapper for obtaining both metrics across multiple samples in one tidy table. Internally, doubling time is derived from the estimated growth rate using `compute_doubling_time()`. Inputs are standardized internally to the canonical `SummarizedExperiment` representation before summary tables are derived.

Value

A tidy tibble with one row per sample by default. When `comparison_col` is supplied, returns one row per comparison group with replicate-level doubling-time summary statistics and optional p-values.

Examples

```
data(yeast_growth_data)
metrics <- summarize_growth_metrics(
  yeast_growth_data,
  method = "rolling_window",
  average_replicates = TRUE
)
head(metrics)
```

`summary,GrowthFit-method` *Summarize a* `GrowthFit` *Object*

Description

Tabular summary of fit status and goodness-of-fit statistics.

Usage

```
## S4 method for signature 'GrowthFit'
summary(object, ...)
```

Arguments

`object`	A GrowthFit object.
`...`	Unused, present for generic compatibility.

Value

A one-row tibble with `sample`, `model`, `converged`, `status`, `message`, `n_points`, `rss`, `aic`, and `bic`.

Examples

```
data(yeast_growth_data)
tidy_growth <- as_tidy_growth_data(yeast_growth_data)
sample_id <- unique(tidy_growth$sample)[1]
fit <- fit_growth_curve(dplyr::filter(tidy_growth, sample == sample_id))
summary(fit)
```

timepoints *Timepoint Metadata*

Description

Access timepoint metadata from a SummarizedExperiment.

Usage

```
timepoints(x)
```

Arguments

x	A SummarizedExperiment compatible with growkar.

Value

A tibble containing rowData(x).

Examples

```
data(yeast_growth_data)
se <- GrowthExperiment(yeast_growth_data)
timepoints(se)
```

validate_growth_data *Validate Canonical Growth Data*

Description

Validate tidy growth data before downstream analysis.

Usage

```
validate_growth_data(
  data,
  allow_negative_od = FALSE,
  require_increasing_time = TRUE,
  min_points_per_sample = 2L,
  warn_zero_od = FALSE,
  require_finite = TRUE
)
```

Arguments

data	A data frame containing at least sample, time, and od.
allow_negative_od	Logical; if FALSE, negative optical density values are rejected.
require_increasing_time	Logical; if TRUE, time must be strictly increasing within each sample.
min_points_per_sample	Minimum number of rows required within each sample.
warn_zero_od	Logical; if TRUE, emit a warning when zero OD values are present.
require_finite	Logical; if TRUE, reject non-finite numeric values in time or od.

Value

The validated tibble after coercion to tibble.

Examples

```
data(yeast_growth_data)
tidy_data <- as_tidy_growth_data(yeast_growth_data)
validate_growth_data(tidy_data)
```

---

validate_growth_experiment *Validate a Growth* SummarizedExperiment

---

Description

Check that a SummarizedExperiment follows the canonical growkar layout.

Usage

```
validate_growth_experiment(x, require_finite = TRUE)
```

Arguments

x	A SummarizedExperiment.
require_finite	Logical; if TRUE, reject non-finite assay values and time values.

Details

The layout itself (an od assay, a numeric rowData(x)$time of matching length) is enforced by the GrowthExperiment validity method, which this function calls rather than re-checking. Only the finiteness requirement, which the class deliberately does not impose, is checked here.

Value

The validated SummarizedExperiment.

Examples

```
data(yeast_growth_data)
se <- GrowthExperiment(yeast_growth_data)
validate_growth_experiment(se)
```

`yeast_growth_data` *Optical Density (OD600) of growing yeast cells*

Description

A dataset containing the time-interval and OD600 of growing yeast cells recorded upto 24 hours.

Format

A data frame with 49 rows and 6 variables:

Time Time, time-interval at which OD600 was recorded

Cg_R1, Cg_R2, Cg_R3 Optical Density (OD600) of yeast cells, at each time-interval

CgFlu_R1, CgFlu_R2, CgFlu_R3 Optical Density (OD600) of yeast cells treated with fluconazole, at each time-interval

YPD_R1, YPD_R2, YPD_R3 Optical Density (OD600) of medium without cells as control, at each time-interval ...

Index