

Package ‘rumenGP’

October 7, 2026

Title Rumen Gas Production Modeling, Comparison, and Visualization

Version 0.2.1

Description Provides tools for importing, processing, visualizing, fitting, comparing, and interpreting in vitro rumen gas production data. Supports ANKOM RF workflows, generic gas production datasets, and pressure-based measurements. Includes multiple kinetic models, custom nonlinear models, model comparison workflows, treatment-level ranking, diagnostic tools, and visualization functions for rumen fermentation studies.

License MIT + file LICENSE

URL <https://araujorodrig-lab.github.io/rumenGP/>,
<https://github.com/araujorodrig-lab/rumenGP>

BugReports <https://github.com/araujorodrig-lab/rumenGP/issues>

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports dplyr, ggplot2, lubridate, minpack.lm, purrr, readxl, tidyr

Suggests knitr, rmarkdown, pkgdown, spelling

VignetteBuilder knitr

Language en-US

NeedsCompilation no

Author Arlan Araujo Rodrigues [aut, cre],
Hilario Cuquetto Mantovani [aut]

Maintainer Arlan Araujo Rodrigues <araujorodrig@wisc.edu>

Repository CRAN

Date/Publication 2026-10-07 13:50:02 UTC

Contents

as_rumen_gp	3
best_model_by_treatment	6
compare_models	7
compare_models_by_treatment	9
example_data	10
exclude_heads	12
fit_brody	13
fit_burr_xii	15
fit_custom	17
fit_dual_logistic	21
fit_exp0	23
fit_expl	25
fit_gompertz	27
fit_groot	29
fit_inverse_paralogistic	32
fit_le0	34
fit_le1	36
fit_logistic	38
fit_loglogistic	40
fit_mitscherlich	42
fit_mm	44
fit_orskov	47
flag_model	49
model_win_frequency	49
parse_ankom_time	51
plot_all_fits	52
plot_all_treatment_means	54
plot_diagnostics	55
plot_dual_pools	56
plot_fit	58
plot_gp	59
plot_model_comparison	60
plot_model_comparison_all	62
plot_model_comparison_treatment	63
plot_model_performance	65
plot_model_rankings	66
plot_residuals	68
plot_residual_comparison	69
plot_treatment_mean	71
pressure_to_volume	72
process_ankom	74
rank_models	76
rank_models_by_treatment	77
read_ankom	79
read_metadata	80
summary.brody_fit	81

summary.burr_xii_fit	83
summary.custom_fit	84
summary.dual_logistic_fit	86
summary.exp0_fit	88
summary.expl_fit	89
summary.gompertz_fit	91
summary.groot_fit	92
summary.inverse_paralogistic_fit	94
summary.le0_fit	95
summary.lel_fit	97
summary.logistic_fit	98
summary.loglogistic_fit	100
summary.mitscherlich_fit	102
summary.mm_fit	103
summary.orskov_fit	105
validate_ankom	107
validate_metadata	108
Index	110

as_rumen_gp	<i>Convert data to a rumen_gp object</i>
-------------	--

Description

Converts gas production data from any source into the internal rumenGP format.

Usage

```
as_rumen_gp(
  data,
  head_col,
  time_col,
  gas_col = NULL,
  pressure_col = NULL,
  treatment_col = NULL,
  bottle_col = NULL,
  rep_col = NULL,
  pressure_unit = c("psi", "kpa"),
  headspace_volume = NULL,
  headspace_unit = c("mL", "L"),
  temperature = 39,
  zero_negative_pressure = FALSE
)
```

Arguments

<code>data</code>	A data frame.
<code>head_col</code>	Column identifying bottles.
<code>time_col</code>	Column containing incubation time.
<code>gas_col</code>	Optional column containing cumulative gas production (mL).
<code>pressure_col</code>	Optional column containing pressure measurements.
<code>treatment_col</code>	Optional treatment column.
<code>bottle_col</code>	Optional bottle column.
<code>rep_col</code>	Optional replicate column.
<code>pressure_unit</code>	Pressure unit. Either "psi" or "kpa".
<code>headspace_volume</code>	Headspace volume. Required when <code>pressure_col</code> is supplied.
<code>headspace_unit</code>	Headspace unit. Either "mL" or "L".
<code>temperature</code>	Incubation temperature in degC.
<code>zero_negative_pressure</code>	Logical. If TRUE, negative pressure values are converted to zero before gas-volume calculations.

Details

The function accepts either cumulative gas volume or gas pressure measurements.

When pressure is supplied, gas volume is estimated using the same conversion used by `process_ankom()`.

Value

A `rumen_gp` object.

Examples

```
# -----
# Example 1: Gas volume data
# -----

manual_volume <- data.frame(
  Bottle = c(
    1, 1, 1,
    2, 2, 2
  ),
  Treatment = c(
    "Control", "Control", "Control",
    "Corn", "Corn", "Corn"
  ),
  Time = c(
    0, 4, 8,
    0, 4, 8
  )
)
```

```

    ),
    Gas = c(
      0, 20, 40,
      0, 35, 60
    )
  )
)

gp <- as_rumen_gp(
  data = manual_volume,
  head_col = "Bottle",
  treatment_col = "Treatment",
  time_col = "Time",
  gas_col = "Gas"
)

head(gp)

# -----
# Example 2: Pressure data
# -----

manual_pressure <- data.frame(
  Bottle = rep(
    1,
    10
  ),
  Time = c(
    0, 2, 4, 6, 8,
    12, 16, 24, 36, 48
  ),
  PSI = c(
    0,
    0.2,
    0.5,
    0.8,
    1.2,
    1.8,
    2.5,
    3.2,
    4.0,
    4.5
  )
)

gp <- as_rumen_gp(
  data = manual_pressure,
  head_col = "Bottle",
  time_col = "Time",
  pressure_col = "PSI",
  pressure_unit = "psi",
  headspace_volume = 60
)

```

```
head(gp)

# Example model fit
fit <- fit_groot(gp)

summary(fit)
```

best_model_by_treatment

Identify the Best Model for Each Treatment

Description

Returns the top-ranked model within each treatment.

Usage

```
best_model_by_treatment(ranked_comparison)
```

Arguments

ranked_comparison
Output from rank_models_by_treatment().

Details

Rankings are obtained from rank_models_by_treatment() and are based on model performance metrics such as:

- R-squared (R^2)
- Root Mean Squared Error (RMSE)
- Residual Sum of Squares (RSS)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

This function provides a concise summary of the best-performing model for each treatment and is useful for identifying whether different treatments are best described by different kinetic models.

Value

A data frame containing the highest-ranked model for each treatment.

See Also

[compare_models_by_treatment](#), [rank_models_by_treatment](#), [model_win_frequency](#), [compare_models](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

groot_fit <- fit_groot(
  gp
)

gompertz_fit <- fit_gompertz(
  gp
)

comparison <- compare_models_by_treatment(
  Groot = groot_fit,
  Gompertz = gompertz_fit
)

ranking <- rank_models_by_treatment(
  comparison
)

best_model_by_treatment(
  ranking
)
```

`compare_models`*Compare Fitted Kinetic Models*

Description

Compares performance metrics across multiple fitted kinetic models.

Usage

```
compare_models(...)
```

Arguments

... Named fitted model objects.

Details

Model comparison metrics typically include:

- R-squared (R^2)
- Root Mean Squared Error (RMSE)
- Residual Sum of Squares (RSS)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

This function helps researchers identify models that provide the best balance between goodness of fit and model complexity.

The resulting comparison table can be used with:

- `rank_models()`
- `plot_model_performance()`
- `plot_model_rankings()`

Value

A data frame summarizing model performance metrics for each fitted model.

See Also

[rank_models](#), [compare_models_by_treatment](#), [plot_model_performance](#), [plot_model_rankings](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

groot_fit <- fit_groot(
```

```
    gp
  )

  gompertz_fit <- fit_gompertz(
    gp
  )

  comparison <- compare_models(
    Groot = groot_fit,
    Gompertz = gompertz_fit
  )

  comparison
```

`compare_models_by_treatment`*Compare Models by Treatment*

Description

Calculates model performance separately for each treatment.

Usage

```
compare_models_by_treatment(...)
```

Arguments

... Fitted model objects.

Details

Performance metrics are computed using treatment-level predictions and observations, allowing direct comparison of competing models within each treatment.

Typical metrics include:

- R-squared (R^2)
- Root Mean Squared Error (RMSE)
- Residual Sum of Squares (RSS)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

This function is useful for determining whether different treatments are best described by different kinetic models.

Value

A data frame containing treatment-level performance metrics for each fitted model.

See Also

[compare_models](#), [rank_models_by_treatment](#), [best_model_by_treatment](#), [model_win_frequency](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

groot_fit <- fit_groot(
  gp
)

gompertz_fit <- fit_gompertz(
  gp
)

compare_models_by_treatment(
  Groot = groot_fit,
  Gompertz = gompertz_fit
)
```

example_data

Example ANKOM Data

Description

Returns paths to example files included with the package.

Usage

```
example_data()
```

Details

The example dataset can be used to explore package functionality, reproduce examples, and learn rumenGP workflows without requiring external files.

The returned object includes:

- Example ANKOM RF data
- Example metadata

These files are used throughout the package documentation, examples, and vignettes.

Value

A named list containing file paths to package example data.

See Also

[read_ankom](#), [read_metadata](#), [process_ankom](#), [fit_groot](#)

Examples

```
files <- example_data()

files

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

head(
  gp
)

fit <- fit_groot(
  gp
)

summary(
  fit
)
```

exclude_heads	<i>Exclude Problematic ANKOM Heads</i>
---------------	--

Description

Removes one or more bottles from a rumen_gp object while recording exclusion information.

Usage

```
exclude_heads(data, heads, reason = NULL)
```

Arguments

data	A rumen_gp object.
heads	Character vector of bottle identifiers to remove.
reason	Character vector containing the reason for each exclusion.

Details

This function is useful for excluding:

- Leaking bottles
- Sensor failures
- Damaged bottles
- Biologically implausible observations
- Other quality-control issues

Exclusion information is retained to support transparent reporting and reproducible analyses.

Value

A filtered rumen_gp object.

See Also

[validate_ankom](#), [flag_model](#), [process_ankom](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
```

```

)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

gp_filtered <- exclude_heads(
  data = gp,
  heads = c(
    "12",
    "18"
  ),
  reason = c(
    "Bottle leak",
    "Sensor malfunction"
  )
)

gp_filtered

```

fit_brody

*Fit Brody model***Description**

Fits the Brody gas production model to each bottle in a rumen_gp dataset.

Usage

```
fit_brody(data, start = NULL)
```

Arguments

data	A rumen_gp object.
start	Optional list of starting values. May contain any of: • A • b • k

Details**Equation:**

$$V(t) = A (1 - be^{-kt})$$

where:

- $V(t)$ is cumulative gas production at time t
- A is asymptotic gas production
- b is an integration constant
- k is the fractional rate constant

Interpretation:

The Brody model describes gas production as a monotonic increase toward an asymptotic value. The parameter k controls the speed of fermentation, while b controls the initial position of the curve.

Advantages:

- Simple and robust
- Stable convergence
- Easy biological interpretation
- Useful as a baseline model

Limitations:

- No explicit lag parameter
- Less flexible than sigmoidal models
- May not adequately represent strongly sigmoidal fermentation profiles

Value

A brody_fit object containing:

- Parameter estimates
- Model diagnostics
- Predicted values
- Residuals

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
```

```

)

# Fit using package default starting values
fit_default <- fit_brody(
  gp
)

summary(fit_default)

# Fit using custom starting values
fit_custom_start <- fit_brody(
  gp,
  start = list(
    A = 120,
    b = 0.9,
    k = 0.05
  )
)

summary(fit_custom_start)

```

fit_burr_xii

Fit Burr XII model

Description

Fits the Burr XII gas production model to each bottle in a rumen_gp dataset.

Usage

```
fit_burr_xii(data, start = NULL)
```

Arguments

data	A rumen_gp object.
start	Optional list of starting values. May contain any of: • VF • r • a • p

Details**Equation:**

$$V(t) = VF [1 - (1 + (rt)^a)^{-p}]$$

where:

- $V(t)$ is cumulative gas production at time t
- VF is asymptotic gas production
- r is the rate parameter
- a is the first shape parameter
- p is the second shape parameter

Interpretation:

The Burr XII model is a highly flexible sigmoidal model capable of describing a wide range of gas production profiles.

The parameter r controls the speed of gas production, while a and p jointly control curve shape, asymmetry, and inflection behavior.

Advantages:

- Excellent flexibility
- Accommodates diverse curve shapes
- Often produces excellent fits
- Identified as one of the top-performing models in a large comparative study of in vitro gas production profiles

Limitations:

- Requires positive incubation times
- Additional parameters increase model complexity
- Greater risk of overfitting than simpler models

Value

A `burr_xii_fit` object containing:

- Parameter estimates
- Model diagnostics
- Predicted values
- Residuals

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)
```

```

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

# Fit using package default starting values
fit_default <- fit_burr_xii(
  gp
)

summary(fit_default)

# Fit using custom starting values
fit_custom_start <- fit_burr_xii(
  gp,
  start = list(
    VF = 120,
    r = 0.10,
    a = 2,
    p = 1
  )
)

summary(fit_custom_start)

```

fit_custom

Fit Custom Nonlinear Model

Description

Fits a user-defined nonlinear model to each bottle in a rumen_gp dataset.

Usage

```

fit_custom(
  data,
  formula,
  start,
  lower = NULL,
  upper = NULL,
  model_name = "Custom"
)

```

Arguments

<code>data</code>	A <code>rumen_gp</code> object.
<code>formula</code>	A nonlinear model formula.
<code>start</code>	Named list of starting values.
<code>lower</code>	Optional named numeric vector of lower parameter bounds.
<code>upper</code>	Optional named numeric vector of upper parameter bounds.
<code>model_name</code>	Character string used to label the fitted model.

Details**Overview:**

This function allows researchers to fit custom nonlinear kinetic equations using `minpack.lm::nlsLM()`. Custom models integrate directly with:

- `summary()`
- `plot_fit()`
- `plot_residuals()`
- `compare_models()`

making them fully compatible with the `rumenGP` modeling framework.

Formula Requirements:

The model formula must:

- Use `Gas_mL` as the response variable
- Use `Time_h` as the time variable
- Include all parameters listed in `start`

Example:

```
Gas_mL ~
  A *
  (
    Time_h /
    (
      Time_h + K
    )
  )
```

Starting Values:

Starting values are supplied through `start`.

Example:

```
start = list(
  A = 150,
  K = 10
)
```

Good starting values often improve convergence and reduce fitting failures.

Parameter Bounds:

Optional lower and upper bounds may be supplied.

Example:

```
lower = c(  
  A = 0,  
  K = 0  
)
```

```
upper = c(  
  A = 500,  
  K = 100  
)
```

Bounds can improve stability and prevent biologically unrealistic parameter estimates.

Best Practices:

- Start with biologically meaningful equations
- Use reasonable starting values
- Apply parameter bounds when appropriate
- Compare custom models against built-in models
- Evaluate both fit quality and parameter interpretability

Value

A custom_fit object containing:

- Model name
- Formula
- Starting values
- Parameter bounds
- Parameter estimates
- Model diagnostics
- Predicted values
- Residuals

See Also

[fit_groot](#), [fit_mm](#), [compare_models](#), [plot_fit](#), [plot_residuals](#)

Examples

```
files <- example_data()  
  
raw_data <- read_ankom(  
  files$ankom
```

```
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

# Hyperbolic model
custom_fit <- fit_custom(

  data = gp,

  formula =
    Gas_mL ~
    A *
    (
      Time_h /
      (
        Time_h + K
      )
    ),

  start = list(
    A = 150,
    K = 10
  ),

  lower = c(
    A = 0,
    K = 0
  ),

  model_name = "Hyperbolic"

)

summary(custom_fit)

plot_fit(
  custom_fit,
  head = 1
)

plot_residuals(
  custom_fit,
  head = 1
)
```

```
# Compare with built-in models
compare_models(
  Groot = fit_groot(gp),
  Hyperbolic = custom_fit
)
```

fit_dual_logistic	<i>Fit dual-pool logistic model</i>
-------------------	-------------------------------------

Description

Fits a dual-pool logistic model representing rapidly and slowly degradable fractions.

Usage

```
fit_dual_logistic(data, start = NULL)
```

Arguments

data	A rumen_gp object.
start	Optional list of starting values. May contain any of: • V1F • V2F • k1 • k2 • lambda

Details

Equation:

$$V(t) = \frac{V1F}{1 + \exp[2 - 4k1(t - \lambda)]} + \frac{V2F}{1 + \exp[2 - 4k2(t - \lambda)]}$$

where:

- $V(t)$ is cumulative gas production at time t
- $V1F$ is the final gas volume from the rapidly fermentable fraction
- $V2F$ is the final gas volume from the slowly fermentable fraction
- $k1$ is the fractional rate constant of the rapid fraction
- $k2$ is the fractional rate constant of the slow fraction
- λ is lag time

Interpretation:

The Dual Logistic model assumes that gas production originates from two independent fermentation pools:

- A rapidly degradable fraction ($V1F$)
- A slowly degradable fraction ($V2F$)

Each fraction follows a logistic fermentation pattern with its own rate constant.

Advantages:

- Represents complex fermentation dynamics
- Separates rapid and slow fermentation pools
- Biologically meaningful parameterization
- Useful for heterogeneous substrates

Limitations:

- Requires estimation of five parameters
- More computationally demanding
- Greater risk of parameter correlation
- May require careful starting values

Value

A `dual_logistic_fit` object containing:

- Parameter estimates
- Model diagnostics
- Predicted values
- Residuals
- Rapid and slow pool estimates

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
```

```

)

# Fit using package default starting values
fit_default <- fit_dual_logistic(
  gp
)

summary(fit_default)

# Fit using custom starting values
fit_custom_start <- fit_dual_logistic(
  gp,
  start = list(
    V1F = 30,
    V2F = 70,
    k1 = 0.20,
    k2 = 0.05,
    lambda = 0.50
  )
)

summary(fit_custom_start)

```

fit_exp0

*Fit exponential model without lag (EXP0)***Description**

Fits the exponential gas production model without an explicit lag phase.

Usage

```
fit_exp0(data, start = NULL)
```

Arguments

data	A rumen_gp object.
start	Optional list of starting values. May contain any of: • Vf • k

Details**Equation:**

$$V(t) = Vf (1 - e^{-kt})$$

where:

- $V(t)$ is cumulative gas production at time t
- Vf is asymptotic gas production
- k is the fractional rate constant

Interpretation:

The EXP0 model assumes that gas production increases exponentially toward an asymptotic value without an explicit lag phase.

Advantages:

- Simple and computationally efficient
- Stable convergence
- Easy biological interpretation

Limitations:

- Does not model lag time
- Limited flexibility for sigmoidal fermentation profiles

Value

An `exp0_fit` object containing:

- Parameter estimates
- Model diagnostics
- Predicted values
- Residuals

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
```

```

    headspace_ml = 210,
    temperature_c = 39
  )

  # Fit using package defaults
  fit_default <- fit_exp0(
    gp
  )

  summary(fit_default)

  # Fit using user-defined starting values
  fit_custom_start <- fit_exp0(
    gp,
    start = list(
      Vf = 120,
      k = 0.05
    )
  )

  summary(fit_custom_start)

```

fit_expl

*Fit exponential model with lag (EXPL)***Description**

Fits the exponential gas production model with an explicit lag phase.

Usage

```
fit_expl(data, start = NULL)
```

Arguments

data	A rumen_gp object.
start	Optional list of starting values. May contain any of: • Vf • k • lambda

Details

Equation:

$$V(t) = Vf \left(1 - e^{-k(t-\lambda)} \right)$$

where:

- $V(t)$ is cumulative gas production at time t
- Vf is asymptotic gas production
- k is the fractional rate constant
- λ is lag time

Interpretation:

The EXPL model assumes that gas production follows an exponential pattern after a lag phase. The lag parameter represents the delay before substantial fermentation begins.

Advantages:

- Explicit lag parameter
- Simple biological interpretation
- Stable convergence

Limitations:

- Less flexible than sigmoidal models
- May not adequately represent multiple fermentation phases

Value

An `expl_fit` object containing:

- Parameter estimates
- Model diagnostics
- Predicted values
- Residuals

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
```

```

    raw_data,
    metadata,
    headspace_ml = 210,
    temperature_c = 39
  )

  # Fit using package default starting values
  fit_default <- fit_expl(
    gp
  )

  summary(fit_default)

  # Fit using custom starting values
  fit_custom_start <- fit_expl(
    gp,
    start = list(
      Vf = 120,
      k = 0.05,
      lambda = 1
    )
  )

  summary(fit_custom_start)

```

fit_gompertz

Fit Gompertz model

Description

Fits the Zwietering-modified Gompertz model to each bottle in a rumen_gp dataset.

Usage

```
fit_gompertz(data, start = NULL)
```

Arguments

data	A rumen_gp object.
start	Optional list of starting values. May contain any of: • A • mu • lambda

Details**Equation:**

$$V(t) = A \exp \left[- \exp \left(\frac{\mu e}{A} (\lambda - t) + 1 \right) \right]$$

where:

- $V(t)$ is cumulative gas production at time t
- A is asymptotic gas production
- μ is the maximum gas production rate
- λ is lag time
- e is Euler's number

Interpretation:

The modified Gompertz model is one of the most commonly used models for gas production kinetics.

It explicitly estimates:

- Final gas production potential (A)
- Maximum gas production rate (μ)
- Lag time (λ)

making it biologically informative and easy to interpret.

Advantages:

- Explicit lag parameter
- Explicit maximum gas production rate
- Excellent flexibility
- Widely used in gas production studies
- Strong biological interpretation

Limitations:

- More computationally demanding than simple exponential models
- Parameters may exhibit correlation in some datasets

Value

A `gompertz_fit` object containing:

- Parameter estimates
- Model diagnostics
- Predicted values
- Residuals

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

# Fit using package default starting values
fit_default <- fit_gompertz(
  gp
)

summary(fit_default)

# Fit using custom starting values
fit_custom_start <- fit_gompertz(
  gp,
  start = list(
    A = 120,
    mu = 5,
    lambda = 1
  )
)

summary(fit_custom_start)
```

fit_groot

Fit Groot model

Description

Fits the Groot gas production model to each bottle in a rumen_gp dataset.

Usage

```
fit_groot(data, start = NULL)
```

Arguments

data	A rumen_gp object.
start	Optional list of starting values. May contain any of: • VF • b • k

Details**Equation:**

$$V(t) = \frac{VF}{1 + \left(\frac{b}{t}\right)^k}$$

where:

- $V(t)$ is cumulative gas production at time t
- VF is asymptotic gas production
- b is the half-time parameter
- k is the shape parameter

Interpretation:

The Groot model is a flexible sigmoidal model widely used in rumen gas production studies.

The parameter b represents the time required to reach approximately half of the asymptotic gas production, while k controls curve shape and steepness.

Advantages:

- Excellent flexibility
- Biologically interpretable parameters
- Often produces excellent fits
- Widely used in rumen fermentation studies

Limitations:

- Requires positive incubation times
- Shape parameter may be less intuitive than simple exponential models

Notes:

The Groot model is mathematically equivalent to the generalized Michaelis-Menten model implemented in `fit_mm()`.

Parameter correspondence:

- $VF = A$
- $b = K$
- $k = c$

Both formulations produce identical fitted values, residuals, diagnostics, AIC, BIC, RMSE, and R-squared when convergence is achieved.

Researchers may choose either formulation according to the terminology commonly used in their field.

Value

A groot_fit object containing:

- Parameter estimates
- Model diagnostics
- Predicted values
- Residuals

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

# Fit using package default starting values
fit_default <- fit_groot(
  gp
)

summary(fit_default)

# Fit using custom starting values
fit_custom_start <- fit_groot(
  gp,
  start = list(
    VF = 120,
    b = 10,
    k = 2
  )
)

summary(fit_custom_start)
```

fit_inverse_paralogistic

Fit Inverse Paralogistic model

Description

Fits the Inverse Paralogistic gas production model to each bottle in a rumen_gp dataset.

Usage

```
fit_inverse_paralogistic(data, start = NULL)
```

Arguments

data	A rumen_gp object.
start	Optional list of starting values. May contain any of: • VF • r • a

Details

Equation:

$$V(t) = VF \left(1 + (rt)^{-a}\right)^{-a}$$

where:

- $V(t)$ is cumulative gas production at time t
- VF is asymptotic gas production
- r is the rate parameter
- a is the shape parameter

Interpretation:

The Inverse Paralogistic model is a flexible sigmoidal model capable of describing a wide range of gas production profiles.

The parameter r controls the speed of gas production, while a controls curve shape, steepness, and inflection behavior.

Advantages:

- Excellent flexibility
- Biologically interpretable parameters
- Often produces excellent fits
- Identified as one of the top-performing models in a large comparative study of in vitro gas production profiles

Limitations:

- Requires positive incubation times
- Shape parameter may be less intuitive than simple exponential models
- Meaningful inflection-point interpretation generally requires $a > 1$

Value

A `inverse_paralogistic_fit` object containing:

- Parameter estimates
- Model diagnostics
- Predicted values
- Residuals

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

# Fit using package default starting values
fit_default <- fit_inverse_paralogistic(
  gp
)

summary(fit_default)

# Fit using custom starting values
fit_custom_start <- fit_inverse_paralogistic(
  gp,
  start = list(
    VF = 120,
    r = 0.10,
    a = 2
  )
)

summary(fit_custom_start)
```

fit_le0

*Fit Logistic-Exponential model (LE0)***Description**

Fits the Logistic-Exponential model without an explicit lag phase.

Usage

```
fit_le0(data, start = NULL)
```

Arguments

data	A rumen_gp object.
start	Optional list of starting values. May contain any of: • A • k • d

Details**Equation:**

$$V(t) = \frac{A(1 - e^{-kt})}{1 + \exp\left[\ln\left(\frac{1}{d}\right) - kt\right]}$$

where:

- $V(t)$ is cumulative gas production at time t
- A is asymptotic gas production
- k is the fractional rate constant
- d is a shape parameter

Interpretation:

The LE0 model combines an exponential fermentation component with a logistic component. Compared with simple exponential models, LE0 provides additional flexibility in curve shape without requiring an explicit lag parameter.

Advantages:

- Flexible sigmoidal behavior
- More adaptable than simple exponential models
- No lag parameter required
- Can accommodate gradual changes in fermentation rate

Limitations:

- More complex than EXP0
- Shape parameter may be less intuitive biologically
- Additional parameter may increase parameter correlation

Value

A `le0_fit` object containing:

- Parameter estimates
- Model diagnostics
- Predicted values
- Residuals

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

# Fit using package default starting values
fit_default <- fit_le0(
  gp
)

summary(fit_default)

# Fit using custom starting values
fit_custom_start <- fit_le0(
  gp,
  start = list(
    A = 120,
    k = 0.05,
    d = 0.50
  )
)

summary(fit_custom_start)
```

fit_lel

*Fit Logistic-Exponential model (LEL)***Description**

Fits the Logistic-Exponential model with an explicit lag phase.

Usage

```
fit_lel(data, start = NULL)
```

Arguments

data	A rumen_gp object.
start	Optional list of starting values. May contain any of: • A • k • d • lambda

Details**Equation:**

$$V(t) = \frac{A (1 - e^{-k(t-\lambda)})}{1 + \exp \left[\ln \left(\frac{1}{d} \right) - k(t - \lambda) \right]}$$

where:

- $V(t)$ is cumulative gas production at time t
- A is asymptotic gas production
- k is the fractional rate constant
- d is a shape parameter
- λ is lag time

Interpretation:

The LEL model combines an exponential fermentation component, a logistic component, and an explicit lag phase.

This model is more flexible than traditional exponential models and can describe complex fermentation dynamics with delayed onset of gas production.

Advantages:

- Explicit lag parameter
- Flexible sigmoidal behavior
- Can represent delayed fermentation

- Often fits complex gas production profiles well

Limitations:

- More parameters than EXP0 or EXPL
- Greater risk of parameter correlation
- May require careful starting values
- Increased computational complexity

Value

A `lsl_fit` object containing:

- Parameter estimates
- Model diagnostics
- Predicted values
- Residuals

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

# Fit using package default starting values
fit_default <- fit_lsl(
  gp
)

summary(fit_default)

# Fit using custom starting values
fit_custom_start <- fit_lsl(
  gp,
  start = list(
    A = 120,
    k = 0.05,
```

```

      d = 0.50,
      lambda = 1
    )
  )

summary(fit_custom_start)

```

fit_logistic

Fit Logistic model

Description

Fits a Logistic gas-production model to each bottle in a rumen_gp dataset.

Usage

```
fit_logistic(data, start = NULL)
```

Arguments

data	A rumen_gp object.
start	Optional list of starting values. May contain any of: • A • k • lambda

Details

Equation:

$$V(t) = \frac{A}{1 + \exp[2 + 4k(\lambda - t)]}$$

where:

- $V(t)$ is cumulative gas production at time t
- A is asymptotic gas production
- k is the fractional rate constant
- λ is lag time

Interpretation:

The Logistic model describes gas production using a sigmoidal curve with an initial lag phase, a period of rapid fermentation, and a plateau approaching the asymptotic gas production.

The parameter k controls the steepness of the curve, while λ determines the position of the sigmoid along the time axis.

Advantages:

- Explicit lag parameter
- Smooth sigmoidal behavior
- Stable convergence
- Widely used in biological growth and fermentation studies

Limitations:

- Assumes a symmetric sigmoidal curve
- May not adequately fit highly asymmetric fermentation profiles
- Less flexible than Gompertz or Dual Logistic models

Value

A `logistic_fit` object containing:

- Parameter estimates
- Model diagnostics
- Predicted values
- Residuals

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

# Fit using package default starting values
fit_default <- fit_logistic(
  gp
)

summary(fit_default)

# Fit using custom starting values
fit_custom_start <- fit_logistic(
```

```
gp,
start = list(
  A = 120,
  k = 0.05,
  lambda = 1
)
summary(fit_custom_start)
```

fit_loglogistic	<i>Fit Log-logistic model</i>
-----------------	-------------------------------

Description

Fits the Log-logistic gas production model to each bottle in a rumen_gp dataset.

Usage

```
fit_loglogistic(data, start = NULL)
```

Arguments

- | | |
|-------|--|
| data | A rumen_gp object. |
| start | Optional list of starting values. May contain any of: <ul style="list-style-type: none">• VF• r• a |

Details

Equation:

$$V(t) = \frac{VF(rt)^a}{1 + (rt)^a}$$

where:

- $V(t)$ is cumulative gas production at time t
- VF is asymptotic gas production
- r is the rate parameter
- a is the shape parameter

Interpretation:

The Log-logistic model is a flexible sigmoidal model capable of describing a wide range of gas production profiles.

The parameter r controls the speed of gas production, while a controls curve shape, steepness, and inflection behavior.

Advantages:

- Excellent flexibility
- Biologically interpretable parameters
- Often produces excellent fits
- Common in nonlinear growth and fermentation modeling

Limitations:

- Requires positive incubation times
- Shape parameter may be less intuitive than simple exponential models

Notes:

The Log-logistic model is mathematically equivalent to both the Groot model implemented in `fit_groot()` and the generalized Michaelis-Menten model implemented in `fit_mm()`.

Parameter correspondence:

- $VF = A$
- $a = c = k$
- $1/r = K = b$

Both formulations produce identical fitted values, residuals, diagnostics, AIC, BIC, RMSE, and R-squared when convergence is achieved.

Researchers may choose either formulation according to the terminology commonly used in their field.

Value

A `loglogistic_fit` object containing:

- Parameter estimates
- Model diagnostics
- Predicted values
- Residuals

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
```

```

    files$metadata
  )

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

# Fit using package default starting values
fit_default <- fit_loglogistic(
  gp
)

summary(fit_default)

# Fit using custom starting values
fit_custom_start <- fit_loglogistic(
  gp,
  start = list(
    VF = 120,
    r = 0.10,
    a = 2
  )
)

summary(fit_custom_start)

```

fit_mitscherlich

Fit Mitscherlich model

Description

Fits the Mitscherlich gas-production model to each bottle in a rumen_gp dataset.

Usage

```
fit_mitscherlich(data, start = NULL)
```

Arguments

data	A rumen_gp object.
start	Optional list of starting values. May contain any of: • A • k • d • lambda

Details**Equation:**

$$V(t) = A \left[1 - \exp \left(-k(t - \lambda) - d \left(\sqrt{t + 0.001} - \sqrt{\lambda + 0.001} \right) \right) \right]$$

where:

- $V(t)$ is cumulative gas production at time t
- A is asymptotic gas production
- k is the fractional rate constant
- d is a diffusion or shape parameter
- λ is lag time

Interpretation:

The Mitscherlich model combines an exponential fermentation component with a diffusion-like term.

The parameter k describes the primary fermentation rate, while d provides additional flexibility for representing changes in fermentation dynamics over time.

Advantages:

- Explicit lag parameter
- Flexible curve shape
- Can describe complex fermentation dynamics
- Often performs well when simple exponential models are inadequate

Limitations:

- More complex than EXP0 or EXPL
- Increased parameter correlation
- Diffusion parameter may be less intuitive biologically
- May require careful starting values

Value

A `mitscherlich_fit` object containing:

- Parameter estimates
- Model diagnostics
- Predicted values
- Residuals

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

# Fit using package default starting values
fit_default <- fit_mitscherlich(
  gp
)

summary(fit_default)

# Fit using custom starting values
fit_custom_start <- fit_mitscherlich(
  gp,
  start = list(
    A = 120,
    k = 0.05,
    d = 0.05,
    lambda = 0.50
  )
)

summary(fit_custom_start)
```

fit_mm

Fit Michaelis-Menten model

Description

Fits the generalized Michaelis-Menten model to each bottle in a rumen_gp dataset.

Usage

```
fit_mm(data, start = NULL)
```

Arguments

data	A rumen_gp object.
start	Optional list of starting values. May contain any of: • A • K • c

Details**Equation:**

$$V(t) = A \frac{t^c}{t^c + K^c}$$

where:

- $V(t)$ is cumulative gas production at time t
- A is asymptotic gas production
- K is the half-time parameter
- c is the shape parameter

Interpretation:

The generalized Michaelis-Menten model describes cumulative gas production using a flexible sigmoidal function.

The parameter K represents the time required to reach approximately half of the asymptotic gas production, while c controls curve shape and steepness.

Advantages:

- Flexible sigmoidal behavior
- Biologically meaningful half-time parameter
- Usually converges reliably
- Well suited for rumen gas production data

Limitations:

- Shape parameter may be difficult to interpret biologically
- More complex than simple exponential models

Notes:

The generalized Michaelis-Menten model is mathematically equivalent to the Groot model implemented in `fit_groot()`.

Parameter correspondence:

- $A = VF$
- $K = b$

- $c = k$

Both formulations produce identical fitted values and model diagnostics when convergence is achieved.

Researchers may choose either formulation according to the terminology commonly used in their field.

Value

A `mm_fit` object containing:

- Parameter estimates
- Model diagnostics
- Predicted values
- Residuals

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

# Fit using package default starting values
fit_default <- fit_mm(
  gp
)

summary(fit_default)

# Fit using custom starting values
fit_custom_start <- fit_mm(
  gp,
  start = list(
    A = 120,
    K = 10,
    c = 2
  )
)
```

```
summary(fit_custom_start)
```

fit_orсков

Fit Orskov and McDonald model

Description

Fits the Orskov and McDonald gas-production model to each bottle in a rumen_gp dataset.

Usage

```
fit_orсков(data, start = NULL)
```

Arguments

data	A rumen_gp object.
start	Optional list of starting values. May contain any of: • VF • b • k

Details

Equation:

$$V(t) = VF + b(1 - e^{-kt})$$

where:

- $V(t)$ is cumulative gas production at time t
- VF is the intercept (initial gas volume)
- b is the fermentable fraction
- k is the fractional rate constant

Interpretation:

The Orskov and McDonald model partitions gas production into:

- An intercept term (VF)
- A fermentable fraction (b)

The asymptotic gas production is:

$$VF + b$$

The parameter k controls the rate at which the asymptote is approached.

Advantages:

- Widely used in ruminant nutrition research
- Parameters have straightforward biological interpretation
- Stable convergence
- Useful benchmark model for comparison

Limitations:

- No explicit lag parameter
- Limited flexibility for highly sigmoidal fermentation profiles
- Less adaptable than Gompertz, Groot, or Dual Logistic models

Value

An orskov_fit object containing:

- Parameter estimates
- Model diagnostics
- Predicted values
- Residuals

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

# Fit using package default starting values
fit_default <- fit_orсков(
  gp
)

summary(fit_default)

# Fit using custom starting values
fit_custom_start <- fit_orсков(
```

```

gp,
start = list(
  VF = 5,
  b = 120,
  k = 0.05
)
)

summary(fit_custom_start)

```

flag_model

Flag potentially problematic model fits

Description

Flags bottles with poor convergence, low R-squared values and parameter-boundary issues.

Usage

```
flag_model(fit, r2_threshold = 0.9)
```

Arguments

fit A fitted model object.
r2_threshold Minimum acceptable R-squared.

Value

Diagnostic table with QC flags.

model_win_frequency

Model Win Frequency

Description

Summarizes how often each model is the best-performing model across treatments.

Usage

```
model_win_frequency(best_models)
```

Arguments

best_models Output from `best_model_by_treatment()`.

Details

Win frequency is calculated from the output of `best_model_by_treatment()` and reports the number of treatments for which each model achieved the highest overall ranking.

This summary is useful for identifying models that consistently perform well across multiple treatments.

Models with higher win frequencies generally demonstrate greater robustness across a dataset, although treatment-specific performance should also be considered.

Value

A data frame summarizing the number and proportion of treatment-level wins for each model.

See Also

[compare_models_by_treatment](#), [rank_models_by_treatment](#), [best_model_by_treatment](#), [compare_models](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

groot_fit <- fit_groot(
  gp
)

gompertz_fit <- fit_gompertz(
  gp
)

comparison <- compare_models_by_treatment(
  Groot = groot_fit,
  Gompertz = gompertz_fit
)

ranking <- rank_models_by_treatment(
  comparison
)
```

```
best_models <- best_model_by_treatment(  
  ranking  
)  
  
model_win_frequency(  
  best_models  
)
```

parse_ankom_time	<i>Parse ANKOM Timestamps</i>
------------------	-------------------------------

Description

Converts ANKOM RF timestamps into elapsed incubation time expressed in hours.

Usage

```
parse_ankom_time(time_raw)
```

Arguments

time_raw Character vector containing ANKOM timestamps.

Details

ANKOM RF systems record measurements using timestamps. This function converts those timestamps into elapsed incubation time, measured relative to the first observation.

The resulting values are used throughout rumenGP for:

- Data processing
- Model fitting
- Visualization
- Model comparison

In most workflows, this function is called automatically by `process_ankom()` and does not need to be used directly.

Value

A numeric vector containing elapsed incubation time in hours.

See Also

[read_ankom](#), [process_ankom](#), [example_data](#)

Examples

```
timestamps <- c(
  "2024-01-01 08:00:00",
  "2024-01-01 12:00:00",
  "2024-01-01 20:00:00"
)

parse_ankom_time(
  timestamps
)

# Typical workflow
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

head(
  gp$Time_h
)
```

plot_all_fits	<i>Plot All Fitted Curves</i>
---------------	-------------------------------

Description

Displays observed and predicted gas production values for all bottles in a fitted model.

Usage

```
plot_all_fits(fit)
```

Arguments

fit	A fitted model object produced by one of the rumenGP model-fitting functions.
-----	---

Details

Each panel corresponds to a single bottle and shows:

- Observed gas production values
- Model predictions

This plot is useful for quickly evaluating model performance across all bottles in a dataset.

Value

A ggplot2 object.

See Also

[plot_fit](#), [plot_residuals](#), [fit_groot](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

fit <- fit_groot(
  gp
)

plot_all_fits(
  fit
)
```

`plot_all_treatment_means`*Plot Treatment Means for All Treatments*

Description

Compares observed and predicted treatment means across multiple fitted models.

Usage

```
plot_all_treatment_means(..., show_se = TRUE)
```

Arguments

<code>...</code>	Fitted model objects.
<code>show_se</code>	Logical. If TRUE, displays a standard-error ribbon around the observed treatment mean.

Details

The observed treatment mean is shown as a black line with optional standard-error bands. Predicted treatment means from each fitted model are overlaid for comparison.

This visualization is useful for evaluating model performance at the treatment level rather than at the individual bottle level.

Value

A ggplot2 object.

See Also

[plot_treatment_mean](#), [compare_models](#), [fit_groot](#), [fit_gompertz](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
```

```
    headspace_ml = 210,  
    temperature_c = 39  
  )  
  
  groot_fit <- fit_groot(  
    gp  
  )  
  
  gompertz_fit <- fit_gompertz(  
    gp  
  )  
  
  plot_all_treatment_means(  
    Groot = groot_fit,  
    Gompertz = gompertz_fit  
  )
```

plot_diagnostics	<i>Plot Diagnostic Summaries</i>
------------------	----------------------------------

Description

Creates diagnostic histograms for a fitted model.

Usage

```
plot_diagnostics(fit)
```

Arguments

fit	A fitted model object.
-----	------------------------

Details

Diagnostic plots can be used to assess:

- Residual distributions
- Parameter estimates
- Model fit quality
- Potential outliers

These plots are useful for evaluating whether model assumptions appear reasonable and for identifying problematic fits.

Value

A named list of ggplot2 objects.

See Also

[plot_fit](#), [plot_residuals](#), [flag_model](#), [fit_groot](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

fit <- fit_groot(
  gp
)

plot_diagnostics(
  fit
)
```

plot_dual_pools	<i>Plot Dual-Pool Logistic Decomposition</i>
-----------------	--

Description

Visualizes the rapid pool, slow pool, total predicted gas production, and observed gas production for a fitted dual-pool logistic model.

Usage

```
plot_dual_pools(fit, head = NULL, treatment = NULL)
```

Arguments

- | | |
|-----------|--|
| fit | A dual_logistic_fit object. |
| head | Optional bottle identifier. If supplied, only that bottle will be plotted. |
| treatment | Optional treatment name. If supplied, a representative bottle from that treatment will be plotted. |

Details

The plot helps interpret the relative contributions of rapidly and slowly fermentable fractions through time.

Components displayed include:

- Observed gas production
- Predicted total gas production
- Rapid fermentation pool
- Slow fermentation pool

This visualization is useful for understanding substrate heterogeneity and fermentation dynamics.

Value

A ggplot2 object.

See Also

[fit_dual_logistic](#), [plot_fit](#), [plot_residuals](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

fit <- fit_dual_logistic(
  gp
)

plot_dual_pools(
  fit,
  head = 1
)
```

`plot_fit`*Plot Fitted Model*

Description

Plots observed and predicted gas production values for an individual bottle.

Usage

```
plot_fit(fit, head = NULL)
```

Arguments

<code>fit</code>	A fitted model object.
<code>head</code>	Optional Head identifier. If omitted and only one bottle is present, that bottle is plotted automatically.

Details

Observed measurements are displayed alongside the fitted model curve, allowing visual assessment of model performance.

This visualization is useful for:

- Evaluating model fit
- Identifying systematic deviations
- Inspecting individual fermentation profiles
- Comparing observed and predicted values

Value

A ggplot2 object.

See Also

[plot_all_fits](#), [plot_residuals](#), [fit_groot](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)
```

```
gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

fit <- fit_groot(
  gp
)

# Plot a specific bottle
plot_fit(
  fit,
  head = 1
)
```

plot_gp

Plot Raw Gas Production Curve

Description

Plots observed gas production measurements for an individual bottle.

Usage

```
plot_gp(data, head)
```

Arguments

data	A rumen_gp object.
head	Bottle identifier to plot.

Details

This visualization displays the raw gas production profile prior to model fitting and is useful for:

- Inspecting fermentation dynamics
- Identifying unusual observations
- Evaluating data quality
- Comparing individual bottle profiles

Value

A ggplot2 object.

See Also

[plot_fit](#), [plot_residuals](#), [process_ankom](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

plot_gp(
  gp,
  head = 1
)
```

plot_model_comparison *Compare Model Fits*

Description

Displays observed gas production values together with predictions from multiple fitted models for a single bottle.

Usage

```
plot_model_comparison(..., head)
```

Arguments

...	Fitted model objects.
head	Head identifier.

Details

This visualization is useful for:

- Comparing competing kinetic models
- Evaluating model performance
- Identifying differences among fitted curves
- Assessing model agreement with observations

Observed measurements are displayed alongside predictions from each supplied model, allowing direct visual comparison.

Value

A ggplot2 object.

See Also

[compare_models](#), [plot_model_comparison_all](#), [plot_model_comparison_treatment](#), [fit_groot](#), [fit_gompertz](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

groot_fit <- fit_groot(
  gp
)

gompertz_fit <- fit_gompertz(
  gp
)

plot_model_comparison(
  Groot = groot_fit,
  Gompertz = gompertz_fit,
```

```
    head = 1  
)
```

`plot_model_comparison_all`*Compare Models for All Bottles*

Description

Displays observed and predicted gas production values for multiple fitted models across all bottles.

Usage

```
plot_model_comparison_all(...)
```

Arguments

... Fitted model objects.

Details

Observed values are shown alongside model predictions, allowing visual comparison of competing kinetic models across the entire dataset.

This visualization is useful for:

- Comparing model performance
- Evaluating agreement between models
- Identifying systematic deviations
- Exploring treatment responses

Value

A ggplot2 object.

See Also

[compare_models](#), [plot_model_comparison](#), [plot_model_comparison_treatment](#), [fit_groot](#),
[fit_gompertz](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

groot_fit <- fit_groot(
  gp
)

gompertz_fit <- fit_gompertz(
  gp
)

plot_model_comparison_all(
  Groot = groot_fit,
  Gompertz = gompertz_fit
)
```

`plot_model_comparison_treatment`*Compare Models for a Treatment*

Description

Displays observed and predicted gas production values for multiple fitted models across all replicates of a selected treatment.

Usage

```
plot_model_comparison_treatment(..., treatment)
```

Arguments

<code>...</code>	Fitted model objects.
<code>treatment</code>	Treatment name.

Details

Observed measurements are displayed alongside model predictions, allowing visual comparison of competing kinetic models within a treatment.

This visualization is useful for:

- Comparing competing models
- Evaluating model performance by treatment
- Assessing agreement among biological replicates
- Identifying systematic prediction errors

Value

A ggplot2 object.

See Also

[compare_models_by_treatment](#), [plot_model_comparison](#), [plot_model_comparison_all](#), [fit_groot](#), [fit_gompertz](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

groot_fit <- fit_groot(
  gp
)

gompertz_fit <- fit_gompertz(
  gp
)

plot_model_comparison_treatment(
  Groot = groot_fit,
  Gompertz = gompertz_fit,
  treatment = unique(
```

```
      gp$Treatment  
    )[[1]  
  )
```

plot_model_performance

Plot Model Performance

Description

Visualizes model performance metrics produced by `compare_models()`.

Usage

```
plot_model_performance(comparison)
```

Arguments

`comparison` Output from `compare_models()`.

Details

This plot provides a graphical comparison of competing models using goodness-of-fit statistics.

Typical metrics include:

- R-squared (R^2)
- Root Mean Squared Error (RMSE)
- Residual Sum of Squares (RSS)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

The visualization helps identify models that balance goodness of fit and model complexity.

Value

A `ggplot2` object.

See Also

[compare_models](#), [rank_models](#), [plot_model_rankings](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

groot_fit <- fit_groot(
  gp
)

gompertz_fit <- fit_gompertz(
  gp
)

comparison <- compare_models(
  Groot = groot_fit,
  Gompertz = gompertz_fit
)

plot_model_performance(
  comparison
)
```

plot_model_rankings	<i>Plot Model Rankings</i>
---------------------	----------------------------

Description

Visualizes model rankings across multiple performance metrics.

Usage

```
plot_model_rankings(ranking)
```

Arguments

ranking	Output from rank_models().
---------	----------------------------

Details

Rankings are typically based on metrics such as:

- R-squared (R^2)
- Root Mean Squared Error (RMSE)
- Residual Sum of Squares (RSS)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

This visualization helps identify models that consistently perform well across several evaluation criteria.

Value

A ggplot2 object.

See Also

[rank_models](#), [compare_models](#), [plot_model_performance](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

groot_fit <- fit_groot(
  gp
)

gompertz_fit <- fit_gompertz(
  gp
)

comparison <- compare_models(
  Groot = groot_fit,
  Gompertz = gompertz_fit
```

```

)

ranking <- rank_models(
  comparison
)

plot_model_rankings(
  ranking
)

```

plot_residuals	<i>Plot Model Residuals</i>
----------------	-----------------------------

Description

Plots residuals for an individual bottle.

Usage

```
plot_residuals(fit, head = NULL)
```

Arguments

fit	A fitted model object containing a predictions element.
head	Optional Head identifier. If omitted and only one bottle is present, that bottle is plotted automatically.

Details

Residuals are calculated as:

$$Residual = Observed - Predicted$$

Residual plots are useful for:

- Identifying systematic model bias
- Detecting outliers
- Evaluating model assumptions
- Assessing goodness of fit

Ideally, residuals should be randomly distributed around zero with no obvious trend through time.

Value

A ggplot2 object.

See Also

[plot_fit](#), [plot_residual_comparison](#), [plot_diagnostics](#), [fit_groot](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

fit <- fit_groot(
  gp
)

plot_residuals(
  fit,
  head = 1
)
```

plot_residual_comparison

Compare Residuals Across Models

Description

Displays residuals from multiple fitted models for a selected bottle.

Usage

```
plot_residual_comparison(..., head)
```

Arguments

...	Fitted model objects.
head	Head identifier.

Details

Residuals are calculated as:

$$\text{Observed} - \text{Predicted}$$

and can be used to evaluate:

- Model bias
- Systematic prediction errors
- Heteroscedasticity
- Relative model performance

Models with residuals that are randomly distributed around zero are generally preferred over models showing systematic patterns.

Value

A ggplot2 object.

See Also

[plot_residuals](#), [plot_model_comparison](#), [compare_models](#), [fit_groot](#), [fit_gompertz](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

groot_fit <- fit_groot(
  gp
)

gompertz_fit <- fit_gompertz(
  gp
)
```

```
plot_residual_comparison(  
  Groot = groot_fit,  
  Gompertz = gompertz_fit,  
  head = 1  
)
```

plot_treatment_mean	<i>Plot Treatment Means Across Models</i>
---------------------	---

Description

Compares observed and predicted treatment means across multiple fitted models for a selected treatment.

Usage

```
plot_treatment_mean(..., treatment, show_se = TRUE)
```

Arguments

...	Fitted model objects.
treatment	Treatment name.
show_se	Logical. If TRUE, displays a standard-error ribbon around the observed treatment mean.

Details

The observed treatment mean is displayed as a black line with optional standard-error bands. Predicted treatment means from each fitted model are overlaid for visual comparison.

This visualization is useful for:

- Comparing competing kinetic models
- Evaluating treatment-level model performance
- Assessing agreement between observations and predictions
- Comparing fermentation dynamics among models

Value

A ggplot2 object.

See Also

[plot_all_treatment_means](#), [compare_models_by_treatment](#), [fit_groot](#), [fit_gompertz](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

groot_fit <- fit_groot(
  gp
)

gompertz_fit <- fit_gompertz(
  gp
)

plot_treatment_mean(
  Groot = groot_fit,
  Gompertz = gompertz_fit,
  treatment = unique(
    gp$Treatment
  )[1]
)
```

pressure_to_volume	<i>Convert Pressure to Gas Volume</i>
--------------------	---------------------------------------

Description

Converts pressure measurements to estimated gas volumes using the ideal gas law.

Usage

```
pressure_to_volume(
  pressure,
  pressure_unit = c("psi", "kpa"),
  headspace_volume,
  headspace_unit = c("mL", "L"),
```

```

    temperature = 39
  )

```

Arguments

pressure	Numeric pressure values.
pressure_unit	Pressure unit. One of: • "psi" • "kpa"
headspace_volume	Headspace volume.
headspace_unit	Headspace volume unit. One of: • "mL" • "L"
temperature	Incubation temperature in degrees Celsius.

Details

The function supports pressure measurements in PSI or kPa and headspace volumes in mL or L.

Equation:

Gas volume is estimated using the ideal gas law:

$$PV = nRT$$

where:

- P is pressure
- V is headspace volume
- n is gas moles
- R is the gas constant
- T is absolute temperature

Estimated gas moles are converted to an equivalent gas volume.

Value

A numeric vector containing estimated gas volumes in mL.

Examples

```

# Convert PSI measurements
pressure_to_volume(
  pressure = c(
    0.5,
    1.0,
    1.5
  ),
  pressure_unit = "psi",

```

```

    headspace_volume = 60,
    headspace_unit = "mL",
    temperature = 39
)

# Convert kPa measurements
pressure_to_volume(
  pressure = c(
    5,
    10,
    15
  ),
  pressure_unit = "kpa",
  headspace_volume = 0.06,
  headspace_unit = "L",
  temperature = 39
)

```

process_ankom

Process ANKOM RF Data

Description

Converts raw ANKOM RF output into a standardized dataset suitable for rumenGP analyses.

Usage

```

process_ankom(
  raw_data,
  metadata = NULL,
  headspace_ml = 210,
  temperature_c = 39,
  zero_negative_pressure = FALSE
)

```

Arguments

raw_data	Raw ANKOM data table.
metadata	Metadata table.
headspace_ml	Bottle headspace volume (mL).
temperature_c	Incubation temperature (°C).
zero_negative_pressure	Logical. If TRUE, negative pressure values are converted to zero before gas-volume calculations.

Details

The function:

- Merges ANKOM measurements with metadata
- Removes Head 0 (the ANKOM receiver/base station)
- Converts pressure measurements to gas volumes
- Applies headspace and temperature corrections
- Produces a standardized rumen_gp object

Head 0 is reserved by the ANKOM RF system as the receiver/base station and is automatically removed during processing.

Value

A rumen_gp object containing:

- Time points
- Gas production values
- Bottle identifiers
- Treatment assignments
- Additional metadata

See Also

[read_ankom](#), [read_metadata](#), [as_rumen_gp](#), [plot_gp](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

head(gp)

# Alternative behavior:
# convert negative pressures to zero
```

```
gp_zeroed <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39,
  zero_negative_pressure = TRUE
)

head(gp_zeroed)
```

rank_models

*Rank Models***Description**

Ranks fitted models using multiple model performance criteria.

Usage

```
rank_models(comparison)
```

Arguments

comparison Output of `compare_models()`.

Details

Rankings are based on metrics produced by `compare_models()` and may include:

- R-squared (R^2)
- Root Mean Squared Error (RMSE)
- Residual Sum of Squares (RSS)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

Models that perform consistently well across multiple metrics typically receive better overall rankings.

This function is useful when comparing several competing kinetic models and identifying those that provide the best balance between fit quality and model complexity.

Value

A data frame containing model rankings across performance metrics.

See Also

[compare_models](#), [rank_models_by_treatment](#), [plot_model_performance](#), [plot_model_rankings](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

groot_fit <- fit_groot(
  gp
)

gompertz_fit <- fit_gompertz(
  gp
)

comparison <- compare_models(
  Groot = groot_fit,
  Gompertz = gompertz_fit
)

rank_models(
  comparison
)
```

rank_models_by_treatment

Rank Models Within Each Treatment

Description

Ranks fitted models within each treatment using performance metrics produced by `compare_models_by_treatment()`.

Usage

```
rank_models_by_treatment(comparison)
```

Arguments

comparison Output from `compare_models_by_treatment()`.

Details

Rankings can be based on metrics such as:

- R-squared (R^2)
- Root Mean Squared Error (RMSE)
- Residual Sum of Squares (RSS)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

This function is useful for identifying the best-performing model within each treatment and for evaluating whether model performance varies among treatments.

Value

A data frame containing model rankings for each treatment and performance metric.

See Also

[compare_models_by_treatment](#), [best_model_by_treatment](#), [model_win_frequency](#), [rank_models](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

groot_fit <- fit_groot(
  gp
)

gompertz_fit <- fit_gompertz(
  gp
)
```

```
comparison <- compare_models_by_treatment(  
  Groot = groot_fit,  
  Gompertz = gompertz_fit  
)  
  
rank_models_by_treatment(  
  comparison  
)
```

read_ankom	<i>Import ANKOM RF Output</i>
------------	-------------------------------

Description

Reads a raw ANKOM RF export file and returns the contents as a data frame.

Usage

```
read_ankom(file)
```

Arguments

file	Path to an ANKOM Excel file.
------	------------------------------

Details

This function is typically the first step in the ANKOM workflow:

- Import ANKOM data
- Import metadata
- Process data
- Fit kinetic models

The imported data can subsequently be processed using `process_ankom()`.

Value

A data frame containing raw ANKOM RF data.

See Also

[read_metadata](#), [process_ankom](#), [example_data](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

head(
  raw_data
)

# Typical workflow
metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

head(
  gp
)
```

read_metadata	<i>Import Metadata</i>
---------------	------------------------

Description

Reads metadata associated with an ANKOM RF experiment.

Usage

```
read_metadata(file, sheet = "metadata")
```

Arguments

- file Path to a metadata file.
- sheet Sheet name containing metadata.

Details

Metadata are used to identify bottles, treatments, replicates, and other experimental information required for downstream analyses.

This function is typically used together with:

- read_ankom()
- process_ankom()

as part of the standard ANKOM workflow.

Value

A data frame containing experimental metadata.

See Also

[read_ankom](#), [process_ankom](#), [validate_metadata](#), [example_data](#)

Examples

```
files <- example_data()

metadata <- read_metadata(
  files$metadata
)

head(
  metadata
)

# Typical workflow
raw_data <- read_ankom(
  files$ankom
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

head(
  gp
)
```

summary.brody_fit

Summary of Brody Fits

Description

Summarizes parameter estimates and goodness-of-fit statistics for a fitted Brody model.

Usage

```
## S3 method for class 'brody_fit'  
summary(object, ...)
```

Arguments

object	A brody_fit object.
...	Additional arguments passed to methods.

Details

The summary typically includes:

- Parameter estimates
- Standard errors (if available)
- Residual Sum of Squares (RSS)
- Root Mean Squared Error (RMSE)
- R-squared (R^2)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

This method provides a concise overview of model performance and parameter values for each fitted bottle.

Value

A data frame containing parameter estimates and model diagnostics for each fitted bottle.

See Also

[fit_brody](#), [plot_fit](#), [plot_residuals](#), [compare_models](#)

Examples

```
files <- example_data()  
  
raw_data <- read_ankom(  
  files$ankom  
)  
  
metadata <- read_metadata(  
  files$metadata  
)  
  
gp <- process_ankom(  
  raw_data,  
  metadata,  
  headspace_ml = 210,
```

```

    temperature_c = 39
  )

  fit <- fit_brody(
    gp
  )

  summary(
    fit
  )

```

summary.burr_xii_fit *Summary of Burr XII Fits*

Description

Summarizes parameter estimates and goodness-of-fit statistics for a fitted Burr XII model.

Usage

```

## S3 method for class 'burr_xii_fit'
summary(object, ...)

```

Arguments

object	A burr_xii_fit object.
...	Additional arguments passed to methods.

Details

The summary typically includes:

- Asymptotic gas production (VF)
- Rate parameter (r)
- Shape parameter (a)
- Shape parameter (p)
- Residual Sum of Squares (RSS)
- Root Mean Squared Error (RMSE)
- R-squared (R^2)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

The Burr XII model is a highly flexible sigmoidal model capable of describing diverse gas production profiles.

The parameter r controls the speed of gas production, while a and p jointly control curve shape, asymmetry, and inflection behavior.

Value

A data frame containing parameter estimates and model diagnostics for each fitted bottle.

See Also

[fit_burr_xii](#), [plot_fit](#), [plot_residuals](#), [compare_models](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

fit <- fit_burr_xii(
  gp
)

summary(
  fit
)
```

summary.custom_fit	<i>Summary of Custom Model Fits</i>
--------------------	-------------------------------------

Description

Summarizes a fitted custom nonlinear model.

Usage

```
## S3 method for class 'custom_fit'
summary(object, ...)
```

Arguments

object	A custom_fit object.
...	Not used.

Details

The summary typically reports:

- Model name
- Model formula
- Parameter estimates
- Residual Sum of Squares (RSS)
- Root Mean Squared Error (RMSE)
- R-squared (R^2)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

This method provides a concise overview of parameter estimates and model performance for user-defined nonlinear equations fitted with `fit_custom()`.

Value

Invisibly returns the input custom_fit object.

See Also

[fit_custom](#), [plot_fit](#), [plot_residuals](#), [compare_models](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

custom_fit <- fit_custom(
```

```

data = gp,
formula =
  Gas_mL ~
    A *
    (
      Time_h /
      (
        Time_h + K
      )
    ),
start = list(
  A = 150,
  K = 10
),
lower = c(
  A = 0,
  K = 0
),
model_name = "Hyperbolic"
)

summary(
  custom_fit
)

```

summary.dual_logistic_fit

Summary of Dual Logistic Fits

Description

Summarizes parameter estimates and goodness-of-fit statistics for a fitted dual-pool logistic model.

Usage

```
## S3 method for class 'dual_logistic_fit'
summary(object, ...)
```

Arguments

object	A dual_logistic_fit object.
...	Additional arguments passed to methods.

Details

The summary typically includes:

- Rapid pool gas volume (V1F)

- Slow pool gas volume (V2F)
- Rapid pool rate constant (k1)
- Slow pool rate constant (k2)
- Lag time (lambda)
- Residual Sum of Squares (RSS)
- Root Mean Squared Error (RMSE)
- R-squared (R²)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

The Dual Logistic model partitions fermentation into rapidly and slowly degradable fractions, providing a biologically informative description of fermentation dynamics.

Value

A data frame containing parameter estimates and model diagnostics for each fitted bottle.

See Also

[fit_dual_logistic](#), [plot_dual_pools](#), [plot_fit](#), [compare_models](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

fit <- fit_dual_logistic(
  gp
)

summary(
  fit
)
```

summary.exp0_fit	<i>Summary of EXP0 Fits</i>
------------------	-----------------------------

Description

Summarizes parameter estimates and goodness-of-fit statistics for a fitted EXP0 model.

Usage

```
## S3 method for class 'exp0_fit'  
summary(object, ...)
```

Arguments

object	An exp0_fit object.
...	Additional arguments passed to methods.

Details

The summary typically includes:

- Asymptotic gas production (V_f)
- Fractional rate constant (k)
- Residual Sum of Squares (RSS)
- Root Mean Squared Error (RMSE)
- R-squared (R^2)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

The EXP0 model describes gas production as an exponential approach to an asymptotic gas volume without an explicit lag phase.

Value

A data frame containing parameter estimates and model diagnostics for each fitted bottle.

See Also

[fit_exp0](#), [plot_fit](#), [plot_residuals](#), [compare_models](#)

Examples

```

files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

fit <- fit_exp0(
  gp
)

summary(
  fit
)

```

summary.expl_fit

*Summary of EXPL Fits***Description**

Summarizes parameter estimates and goodness-of-fit statistics for a fitted EXPL model.

Usage

```

## S3 method for class 'expl_fit'
summary(object, ...)

```

Arguments

object	An expl_fit object.
...	Additional arguments passed to methods.

Details

The summary typically includes:

- Asymptotic gas production (Vf)

- Fractional rate constant (k)
- Lag time (lambda)
- Residual Sum of Squares (RSS)
- Root Mean Squared Error (RMSE)
- R-squared (R^2)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

The EXPL model describes gas production as an exponential approach to an asymptotic gas volume following a lag phase.

The lag parameter represents the delay before substantial fermentation begins.

Value

A data frame containing parameter estimates and model diagnostics for each fitted bottle.

See Also

[fit_expl](#), [plot_fit](#), [plot_residuals](#), [compare_models](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

fit <- fit_expl(
  gp
)

summary(
  fit
)
```

`summary.gompertz_fit` *Summary of Gompertz Fits*

Description

Summarizes parameter estimates and goodness-of-fit statistics for a fitted Gompertz model.

Usage

```
## S3 method for class 'gompertz_fit'  
summary(object, ...)
```

Arguments

<code>object</code>	A <code>gompertz_fit</code> object.
<code>...</code>	Additional arguments passed to methods.

Details

The summary typically includes:

- Asymptotic gas production (A)
- Maximum gas production rate (μ)
- Lag time (λ)
- Residual Sum of Squares (RSS)
- Root Mean Squared Error (RMSE)
- R-squared (R^2)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

The modified Gompertz model provides a biologically meaningful description of fermentation kinetics by explicitly estimating:

- Final gas production potential
- Maximum fermentation rate
- Lag phase duration

Value

A data frame containing parameter estimates and model diagnostics for each fitted bottle.

See Also

[fit_gompertz](#), [plot_fit](#), [plot_residuals](#), [compare_models](#)

Examples

```

files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

fit <- fit_gompertz(
  gp
)

summary(
  fit
)

```

summary.groot_fit

*Summary of Groot Fits***Description**

Summarizes parameter estimates and goodness-of-fit statistics for a fitted Groot model.

Usage

```

## S3 method for class 'groot_fit'
summary(object, ...)

```

Arguments

object	A groot_fit object.
...	Additional arguments passed to methods.

Details

The summary typically includes:

- Asymptotic gas production (VF)

- Half-time parameter (b)
- Shape parameter (k)
- Residual Sum of Squares (RSS)
- Root Mean Squared Error (RMSE)
- R-squared (R^2)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

The Groot model is a flexible sigmoidal model commonly used in rumen gas production studies.

The parameter b represents the time required to reach approximately half of the asymptotic gas production, while k controls curve shape and steepness.

Notes:

The Groot model is mathematically equivalent to the generalized Michaelis-Menten model implemented in `fit_mm()`.

Parameter correspondence:

- $VF = A$
- $b = K$
- $k = c$

Value

A data frame containing parameter estimates and model diagnostics for each fitted bottle.

See Also

[fit_groot](#), [fit_mm](#), [plot_fit](#), [plot_residuals](#), [compare_models](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

fit <- fit_groot(
```

```

    gp
  )

summary(
  fit
)

```

```
summary.inverse_paralogistic_fit
```

Summary of Inverse Paralogistic Fits

Description

Summarizes parameter estimates and goodness-of-fit statistics for a fitted Inverse Paralogistic model.

Usage

```
## S3 method for class 'inverse_paralogistic_fit'
summary(object, ...)
```

Arguments

object	An inverse_paralogistic_fit object.
...	Additional arguments passed to methods.

Details

The summary typically includes:

- Asymptotic gas production (VF)
- Rate parameter (r)
- Shape parameter (a)
- Residual Sum of Squares (RSS)
- Root Mean Squared Error (RMSE)
- R-squared (R^2)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

The Inverse Paralogistic model is a flexible sigmoidal model capable of describing a broad range of cumulative gas production profiles.

The parameter r controls the speed of gas production, while a controls curve shape, steepness, and inflection behavior.

Value

A data frame containing parameter estimates and model diagnostics for each fitted bottle.

See Also

[fit_inverse_paralogistic](#), [plot_fit](#), [plot_residuals](#), [compare_models](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

fit <- fit_inverse_paralogistic(
  gp
)

summary(
  fit
)
```

summary.le0_fit

Summary of LE0 Fits

Description

Summarizes parameter estimates and goodness-of-fit statistics for a fitted LE0 model.

Usage

```
## S3 method for class 'le0_fit'
summary(object, ...)
```

Arguments

object A `le0_fit` object.
 ... Additional arguments passed to methods.

Details

The summary typically includes:

- Asymptotic gas production (A)
- Fractional rate constant (k)
- Shape parameter (d)
- Residual Sum of Squares (RSS)
- Root Mean Squared Error (RMSE)
- R-squared (R^2)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

The LE0 (Logistic-Exponential) model combines exponential fermentation kinetics with a logistic component to provide additional flexibility in curve shape without requiring an explicit lag phase.

The shape parameter *d* controls the curvature of the fermentation profile and can improve fit performance for sigmoidal gas production data.

Value

A data frame containing parameter estimates and model diagnostics for each fitted bottle.

See Also

[fit_le0](#), [fit_le1](#), [plot_fit](#), [plot_residuals](#), [compare_models](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)
```

```

fit <- fit_le0(
  gp
)

summary(
  fit
)

```

summary.lel_fit	<i>Summary of LEL Fits</i>
-----------------	----------------------------

Description

Summarizes parameter estimates and goodness-of-fit statistics for a fitted LEL model.

Usage

```

## S3 method for class 'lel_fit'
summary(object, ...)

```

Arguments

object	A lel_fit object.
...	Additional arguments passed to methods.

Details

The summary typically includes:

- Asymptotic gas production (A)
- Fractional rate constant (k)
- Shape parameter (d)
- Lag time (lambda)
- Residual Sum of Squares (RSS)
- Root Mean Squared Error (RMSE)
- R-squared (R^2)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

The LEL (Logistic-Exponential with Lag) model combines exponential fermentation kinetics with a logistic component and an explicit lag phase.

The lag parameter (lambda) represents the delay before substantial fermentation begins, while the shape parameter (d) controls curve flexibility.

This combination makes the LEL model suitable for describing complex sigmoidal fermentation profiles with delayed onset.

Value

A data frame containing parameter estimates and model diagnostics for each fitted bottle.

See Also

[fit_le1](#), [fit_le0](#), [plot_fit](#), [plot_residuals](#), [compare_models](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

fit <- fit_le1(
  gp
)

summary(
  fit
)
```

summary.logistic_fit *Summary of Logistic Fits*

Description

Summarizes parameter estimates and goodness-of-fit statistics for a fitted Logistic model.

Usage

```
## S3 method for class 'logistic_fit'
summary(object, ...)
```

Arguments

object	A logistic_fit object.
...	Additional arguments passed to methods.

Details

The summary typically includes:

- Asymptotic gas production (A)
- Fractional rate constant (k)
- Lag time (lambda)
- Residual Sum of Squares (RSS)
- Root Mean Squared Error (RMSE)
- R-squared (R^2)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

The Logistic model describes gas production using a sigmoidal curve characterized by:

- An initial lag phase
- A rapid fermentation phase
- A plateau approaching asymptotic gas production

The lag parameter (lambda) determines the position of the sigmoid along the time axis, while k controls curve steepness.

Value

A data frame containing parameter estimates and model diagnostics for each fitted bottle.

See Also

[fit_logistic](#), [fit_gompertz](#), [plot_fit](#), [plot_residuals](#), [compare_models](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
```

```
    metadata,  
    headspace_ml = 210,  
    temperature_c = 39  
  )  
  
  fit <- fit_logistic(  
    gp  
  )  
  
  summary(  
    fit  
  )
```

summary.loglogistic_fit

Summary of Log-logistic Fits

Description

Summarizes parameter estimates and goodness-of-fit statistics for a fitted Log-logistic model.

Usage

```
## S3 method for class 'loglogistic_fit'  
summary(object, ...)
```

Arguments

object	A loglogistic_fit object.
...	Additional arguments passed to methods.

Details

The summary typically includes:

- Asymptotic gas production (VF)
- Rate parameter (r)
- Shape parameter (a)
- Residual Sum of Squares (RSS)
- Root Mean Squared Error (RMSE)
- R-squared (R²)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

The Log-logistic model is a flexible sigmoidal model capable of describing a broad range of gas production profiles.

The parameter r controls the speed of gas production, while a controls curve shape, steepness, and inflection behavior.

Notes:

The Log-logistic model is mathematically equivalent to both the Groot model implemented in `fit_groot()` and the generalized Michaelis-Menten model implemented in `fit_mm()`.

Parameter correspondence:

- $VF = A$
- $a = c = k$
- $1/r = K = b$

All three formulations produce identical fitted values, residuals, diagnostics, AIC, BIC, RMSE, and R-squared when corresponding parameter values are used.

Value

A data frame containing parameter estimates and model diagnostics for each fitted bottle.

See Also

[fit_loglogistic](#), [fit_groot](#), [fit_mm](#), [plot_fit](#), [plot_residuals](#), [compare_models](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

fit <- fit_loglogistic(
  gp
)

summary(
  fit
)
```

`summary.mitscherlich_fit`*Summary of Mitscherlich Fits*

Description

Summarizes parameter estimates and goodness-of-fit statistics for a fitted Mitscherlich model.

Usage

```
## S3 method for class 'mitscherlich_fit'  
summary(object, ...)
```

Arguments

<code>object</code>	A <code>mitscherlich_fit</code> object.
<code>...</code>	Additional arguments passed to methods.

Details

The summary typically includes:

- Asymptotic gas production (A)
- Fractional rate constant (k)
- Diffusion or shape parameter (d)
- Lag time (λ)
- Residual Sum of Squares (RSS)
- Root Mean Squared Error (RMSE)
- R-squared (R^2)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

The Mitscherlich model combines an exponential fermentation component with a diffusion-like term, allowing greater flexibility in describing complex fermentation dynamics.

The parameter k represents the primary fermentation rate, while d adjusts the shape of the fermentation profile.

Value

A data frame containing parameter estimates and model diagnostics for each fitted bottle.

See Also

[fit_mitscherlich](#), [plot_fit](#), [plot_residuals](#), [compare_models](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

fit <- fit_mitscherlich(
  gp
)

summary(
  fit
)
```

summary.mm_fit

Summary of Michaelis-Menten Fits

Description

Summarizes parameter estimates and goodness-of-fit statistics for a fitted Michaelis-Menten model.

Usage

```
## S3 method for class 'mm_fit'
summary(object, ...)
```

Arguments

object	A mm_fit object.
...	Additional arguments passed to methods.

Details

The summary typically includes:

- Asymptotic gas production (A)

- Half-time parameter (K)
- Shape parameter (c)
- Residual Sum of Squares (RSS)
- Root Mean Squared Error (RMSE)
- R-squared (R^2)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

The generalized Michaelis-Menten model is a flexible sigmoidal model commonly used to describe cumulative gas production.

The parameter K represents the time required to reach approximately half of the asymptotic gas production, while c controls curve shape and steepness.

Notes:

The generalized Michaelis-Menten model is mathematically equivalent to the Groot model implemented in `fit_groot()`.

Parameter correspondence:

- A = VF
- K = b
- c = k

Both formulations produce identical fitted values and model diagnostics when convergence is achieved.

Value

A data frame containing parameter estimates and model diagnostics for each fitted bottle.

See Also

[fit_mm](#), [fit_groot](#), [plot_fit](#), [plot_residuals](#), [compare_models](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
```

```

    temperature_c = 39
  )

  fit <- fit_mm(
    gp
  )

  summary(
    fit
  )

```

summary.orskov_fit	<i>Summary of Orskov and McDonald Fits</i>
--------------------	--

Description

Summarizes parameter estimates and goodness-of-fit statistics for a fitted Orskov and McDonald model.

Usage

```

## S3 method for class 'orskov_fit'
summary(object, ...)

```

Arguments

object	An orskov_fit object.
...	Additional arguments passed to methods.

Details

The summary typically includes:

- Initial gas volume (VF)
- Fermentable fraction (b)
- Fractional rate constant (k)
- Residual Sum of Squares (RSS)
- Root Mean Squared Error (RMSE)
- R-squared (R^2)
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

The Orskov and McDonald model partitions gas production into:

- An intercept term (VF)

- A fermentable fraction (b)

The asymptotic gas production is:

$$VF + b$$

The parameter k controls the rate at which the asymptote is approached.

This model is widely used in ruminant nutrition research because the parameters have straightforward biological interpretation.

Value

A data frame containing parameter estimates and model diagnostics for each fitted bottle.

See Also

[fit_orskov](#), [plot_fit](#), [plot_residuals](#), [compare_models](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

fit <- fit_orskov(
  gp
)

summary(
  fit
)
```

`validate_ankom`*Validate Processed Rumen Gas Production Data*

Description

Performs quality-control checks on a `rumen_gp` object.

Usage

```
validate_ankom(data)
```

Arguments

`data` A `rumen_gp` object.

Details

Supported data sources include datasets created by:

- `process_ankom()`
- `as_rumen_gp()`

Validation checks may include:

- Required columns
- Missing values
- Duplicate observations
- Time ordering
- Gas production values
- ANKOM-specific pressure checks

ANKOM-specific checks are performed only when `Gas_PSI` is available.

This function is useful for confirming that a dataset is suitable for downstream modeling, visualization, and model comparison workflows.

Value

The validated `rumen_gp` object.

See Also

[process_ankom](#), [as_rumen_gp](#), [validate_metadata](#)

Examples

```
files <- example_data()

raw_data <- read_ankom(
  files$ankom
)

metadata <- read_metadata(
  files$metadata
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

validate_ankom(
  gp
)

# Validation also supports datasets
# created using as_rumen_gp()
```

validate_metadata	<i>Validate Metadata</i>
-------------------	--------------------------

Description

Validates experimental metadata prior to analysis.

Usage

```
validate_metadata(metadata)
```

Arguments

metadata Metadata table.

Details

Metadata are required for linking bottles to treatments and biological replicates during data processing and model fitting.

Required columns:

- Head
- Treatment

- Rep

Validation checks may include:

- Presence of required columns
- Missing values
- Duplicate bottle identifiers
- Invalid treatment assignments

This function is typically used before `process_ankom()` to ensure metadata are suitable for downstream analyses.

Value

The validated metadata table.

See Also

[read_metadata](#), [validate_ankom](#), [process_ankom](#), [example_data](#)

Examples

```
files <- example_data()

metadata <- read_metadata(
  files$metadata
)

validate_metadata(
  metadata
)

# Typical workflow
raw_data <- read_ankom(
  files$ankom
)

gp <- process_ankom(
  raw_data,
  metadata,
  headspace_ml = 210,
  temperature_c = 39
)

head(
  gp
)
```

Index

as_rumen_gp, [3](#), [75](#), [107](#)

best_model_by_treatment, [6](#), [10](#), [50](#), [78](#)

compare_models, [6](#), [7](#), [10](#), [19](#), [50](#), [54](#), [61](#), [62](#),
[65](#), [67](#), [70](#), [76](#), [82](#), [84](#), [85](#), [87](#), [88](#), [90](#),
[91](#), [93](#), [95](#), [96](#), [98](#), [99](#), [101](#), [102](#), [104](#),
[106](#)

compare_models_by_treatment, [6](#), [8](#), [9](#), [50](#),
[64](#), [71](#), [78](#)

example_data, [10](#), [51](#), [79](#), [81](#), [109](#)

exclude_heads, [12](#)

fit_brody, [13](#), [82](#)

fit_burr_xii, [15](#), [84](#)

fit_custom, [17](#), [85](#)

fit_dual_logistic, [21](#), [57](#), [87](#)

fit_exp0, [23](#), [88](#)

fit_expl, [25](#), [90](#)

fit_gompertz, [27](#), [54](#), [61](#), [62](#), [64](#), [70](#), [71](#), [91](#),
[99](#)

fit_groot, [11](#), [19](#), [29](#), [53](#), [54](#), [56](#), [58](#), [61](#), [62](#),
[64](#), [69–71](#), [93](#), [101](#), [104](#)

fit_inverse_paralogistic, [32](#), [95](#)

fit_le0, [34](#), [96](#), [98](#)

fit_lel, [36](#), [96](#), [98](#)

fit_logistic, [38](#), [99](#)

fit_loglogistic, [40](#), [101](#)

fit_mitscherlich, [42](#), [102](#)

fit_mm, [19](#), [44](#), [93](#), [101](#), [104](#)

fit_orсков, [47](#), [106](#)

flag_model, [12](#), [49](#), [56](#)

model_win_frequency, [6](#), [10](#), [49](#), [78](#)

parse_ankom_time, [51](#)

plot_all_fits, [52](#), [58](#)

plot_all_treatment_means, [54](#), [71](#)

plot_diagnostics, [55](#), [69](#)

plot_dual_pools, [56](#), [87](#)

plot_fit, [19](#), [53](#), [56](#), [57](#), [58](#), [60](#), [69](#), [82](#), [84](#),
[85](#), [87](#), [88](#), [90](#), [91](#), [93](#), [95](#), [96](#), [98](#), [99](#),
[101](#), [102](#), [104](#), [106](#)

plot_gp, [59](#), [75](#)

plot_model_comparison, [60](#), [62](#), [64](#), [70](#)

plot_model_comparison_all, [61](#), [62](#), [64](#)

plot_model_comparison_treatment, [61](#), [62](#),
[63](#)

plot_model_performance, [8](#), [65](#), [67](#), [76](#)

plot_model_rankings, [8](#), [65](#), [66](#), [76](#)

plot_residual_comparison, [69](#), [69](#)

plot_residuals, [19](#), [53](#), [56–58](#), [60](#), [68](#), [70](#),
[82](#), [84](#), [85](#), [88](#), [90](#), [91](#), [93](#), [95](#), [96](#), [98](#),
[99](#), [101](#), [102](#), [104](#), [106](#)

plot_treatment_mean, [54](#), [71](#)

pressure_to_volume, [72](#)

process_ankom, [11](#), [12](#), [51](#), [60](#), [74](#), [79](#), [81](#),
[107](#), [109](#)

rank_models, [8](#), [65](#), [67](#), [76](#), [78](#)

rank_models_by_treatment, [6](#), [10](#), [50](#), [76](#),
[77](#)

read_ankom, [11](#), [51](#), [75](#), [79](#), [81](#)

read_metadata, [11](#), [75](#), [79](#), [80](#), [109](#)

summary.brody_fit, [81](#)

summary.burr_xii_fit, [83](#)

summary.custom_fit, [84](#)

summary.dual_logistic_fit, [86](#)

summary.exp0_fit, [88](#)

summary.expl_fit, [89](#)

summary.gompertz_fit, [91](#)

summary.groot_fit, [92](#)

summary.inverse_paralogistic_fit, [94](#)

summary.le0_fit, [95](#)

summary.lel_fit, [97](#)

summary.logistic_fit, [98](#)

summary.loglogistic_fit, [100](#)

summary.mitscherlich_fit, [102](#)

summary.mm_fit, [103](#)

`summary.orskov_fit`, [105](#)

`validate_ankom`, [12](#), [107](#), [109](#)

`validate_metadata`, [81](#), [107](#), [108](#)