Package: pkpd.Release (via r-universe)

October 10, 2026

Type Package

Title Model Fitting and Simulation for Drug Release Kinetics and PK/PD

Version 0.1.0

Author Paul Angelo C. Manlapaz [aut, cre] (ORCID: <https://orcid.org/0000-0002-1203-2064>)

Maintainer Paul Angelo C. Manlapaz <pacmanlapaz@gmail.com>

Description Provides a comprehensive framework for model fitting and simulation of drug release kinetics, pharmacokinetics (PK), and pharmacodynamics (PD). The package implements widely used mechanistic and empirical models for in vitro drug release, including zero-order, first-order, Higuchi, Korsmeyer-Peppas, Hixson-Crowell, and Weibull models. Pharmacokinetic functionality includes linear and nonlinear functions for one- and two-compartment models for intravenous bolus and oral administration, Michaelis-Menten kinetics, and non-compartmental analysis (NCA). Pharmacodynamic and dose-response modeling is supported through Emax-based models, including stimulatory (sigmoid Emax) and inhibitory (sigmoid Imax) Hill models, four- and five-parameter logistic models, as well as median toxic dose (TD50) and lethal dose (LD50) models. The package is intended to support parameter estimation, simulation, and model comparison in pharmaceutical research, drug development, and pharmacometrics education. For more details, see Gabrielsson & Weiner (2000) <ISBN:9186274929>, Holford & Sheiner (1981) <doi:10.2165/00003088-198106060-00002>, and Manlapaz (2025) <doi:10.32614/CRAN.package.adsoRptionCMF>.

License GPL-3

Encoding UTF-8

Depends R (>= 4.1.0)

Imports stats, ggplot2, minpack.lm, gridExtra, scales

Suggests testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

RoxygenNote 7.3.2

NeedsCompilation no

Repository https://piey27.r-universe.dev

Date/Publication 2026-01-27 08:40:02 UTC

RemoteUrl https://github.com/cran/pkpd.Release

RemoteRef HEAD

RemoteSha 6bf05c115d64299ffede8aa8a47639afe6f8d50e

Contents

first_order_release 2
higuchi_release 5
hixson_crowell_model 7
korsmeyer_peppas_model 10
ld50_model 12
logistic_4pl 14
logistic_5pl 16
michaelis_menten 18
michaelis_menten_nl 22
non_compartmental 24
non_compartmental_nl 26
one_compartment_iv_bolus 28
one_compartment_iv_bolus_nl 31
one_compartment_oral 33
one_compartment_oral_nl 35
sigmoid_emax 38
sigmoid_imax 39
td50_model 41
two_compartment_iv_bolus 44
two_compartment_iv_bolus_nl 46
weibull_model 48
zero_order_release 50

Index **53**

`first_order_release` *First-Order Drug Release Kinetic Model*

Description

Fits experimental cumulative drug release data to a first-order kinetic model using linear regression on the log-transformed unreleased fraction. The function supports optional grouping (formulation/batch) and pH-dependent analysis. It can generate plots with straight lines and annotations for first-order rate constant (k1), intercept, coefficient of determination (R^2), and time required for 50-percent drug release (t50).

Arguments

data	A data frame containing experimental drug release data.
time_col	Character string specifying the column name for time.
log_remain_col	Column name for log cumulative percent drug remaining.
group_col	Optional character string specifying a column for grouping (e.g., formulation/batch).
pH_col	Optional character string specifying a column containing pH values.
plot	Logical; if TRUE, generates a plot of experimental data with first-order fitted curves.
annotate	Logical; if TRUE, annotates the plot with k1, intercept, R^2, and t50 (only if <= 2 groups).

Value

A list containing:

fitted_parameters Data frame with k1, intercept, R^2, and t50 values for each group or pH condition.

data The processed data used for model fitting and plotting.

Author(s)

Paul Angelo C. Manlapaz

References

Ostwald, W. (1884) <doi:10.1002/prac.18840290139> Studien zur chemischen Dynamik. Journal für Praktische Chemie, 29(1), 385–408.

Noyes, A. A., & Whitney, W. R. (1897) <doi:10.1021/ja02086a003> The rate of solution of solid substances in their own solutions. Journal of the American Chemical Society, 19(12), 930–934.

Examples

```
# Example I: Single formulation
df_1 <- data.frame(
  time = c(0, 15, 30, 45, 60, 90, 120, 150, 180),
  log_remain = c(2, 1.947, 1.899, 1.840, 1.780, 1.625, 1.447, 1.182, 0.813)
)
first_order_release(
  data = df_1,
  time_col = "time",
  log_remain_col = "log_remain"
)

# Example II: Two formulations (grouped, not pH-dependent)
df_2 <- data.frame(
  time = rep(c(0, 30, 60, 90, 120, 150), 2),
  log_remain = c(
    2.00, 1.84, 1.73, 1.53, 1.37, 1.25,  # Formulation A
```

```
    2.00, 1.88, 1.76, 1.67, 1.53, 1.39   # Formulation B
  ),
  formulation = rep(c("Formulation A", "Formulation B"), each = 6)
)
first_order_release(
  data = df_2,
  time_col = "time",
  log_remain_col = "log_remain",
  group_col = "formulation"
)

# Example III: pH-dependent first-order release
df_pH <- data.frame(
  time = rep(c(0, 60, 120, 180), 2),
  log_remain = c(
    2.00, 1.74, 1.38, 1.11,  # pH 7.4
    2.00, 1.84, 1.66, 1.48   # pH 4.5
  ),
  pH = rep(c(7.4, 4.5), each = 4)
)
first_order_release(
  data = df_pH,
  time_col = "time",
  log_remain_col = "log_remain",
  pH_col = "pH"
)

# Example IV: Two formulations under two pH conditions
df1 <- data.frame(
  time = rep(c(0, 30, 60, 90, 120, 150, 180), 2),
  log_remain = c(
    2.000, 1.918, 1.842, 1.755, 1.685, 1.598, 1.520,
    2.000, 1.865, 1.748, 1.612, 1.488, 1.352, 1.225
  ),
  pH = rep(c(4.5, 7.6), each = 7)
)
df2 <- data.frame(
  time = rep(c(0, 20, 40, 60, 80, 100, 120), 2),
  log_remain = c(
    2.000, 1.936, 1.872, 1.806, 1.742, 1.675, 1.610,
    2.000, 1.882, 1.760, 1.645, 1.522, 1.408, 1.295
  ),
  pH = rep(c(4.5, 7.6), each = 7)
)
df_all <- rbind(
  cbind(formulation = "Dataset 1", df1),
  cbind(formulation = "Dataset 2", df2)
)
first_order_release(
  data = df_all,
  time_col = "time",
  log_remain_col = "log_remain",
  group_col = "formulation",
```

```
  pH_col = "pH"
)
```

`higuchi_release` *Higuchi Drug Release Kinetic Model*

Description

Fits experimental cumulative drug release data to the Higuchi square-root kinetic model using linear regression of cumulative percent drug released versus square root of time. The function supports optional grouping variables (e.g., formulation, batch) and optional pH-dependent analysis. It generates publication-quality plots with experimental curves, fitted Higuchi straight lines, and annotations for Higuchi release constant (kH), intercept, coefficient of determination (R^2), and the time required for 50-percent drug release (t50).

Arguments

`data`	A data frame containing experimental drug release data.
`sqrt_time_col`	Character string specifying the column name for square root of time.
`release_col`	Character string specifying the column name for cumulative percent drug released.
`group_col`	Optional character string specifying a grouping variable (e.g., formulation, batch). Default is NULL.
`pH_col`	Optional character string specifying a column containing pH values. If provided, Higuchi models are fitted separately for each pH.
`plot`	Logical; if TRUE, generates a plot (default = TRUE).
`annotate`	Logical; if TRUE, annotates the plot with kH, intercept, R^2, and t50 values (only if <= 2 groups).

Value

A list containing:

`fitted_parameters` A data frame with kH, intercept, R^2, and t50 values for each group or pH condition.

`data` The processed data used for fitting and plotting.

Author(s)

Paul Angelo C. Manlapaz

References

Higuchi, T. (1963) <doi:10.1002/jps.2600521210> Mechanism of sustained-action medication. Journal of Pharmaceutical Sciences, 52(12), 1145–1149.

Examples

```
# Example I: Single Formulation
df <- data.frame(
  sqrt_time = c(0, 3.873, 5.477, 6.708, 7.746, 9.487, 10.954, 12.247, 13.416),
  release = c(0, 11.4, 20.8, 30.8, 39.8, 57.8, 72, 84.8, 93.5)
)
higuchi_release(
  data = df,
  sqrt_time_col = "sqrt_time",
  release_col = "release"
)

# Example II: Two formulations (grouped, not pH-dependent)
df_2 <- data.frame(
  sqrt_time = c(0, 3.873, 5.477, 6.708, 7.746, 9.487, 10.954, 12.247, 13.416),
  release = c(
    0, 11.4, 20.8, 30.8, 39.8, 57.8, 72.0, 84.8, 93.5,   # Formulation A
    0, 10.2, 18.6, 29.7, 37.8, 56.5, 71.9, 83.7, 92.9   # Formulation B
  ),
  formulation = rep(c("Formulation A", "Formulation B"), each = 9)
)
higuchi_release(
  data = df_2,
  sqrt_time_col = "sqrt_time",
  release_col = "release",
  group_col = "formulation"
)

# Example III: pH-dependent Higuchi release
df_pH <- data.frame(
  sqrt_time = rep(
    c(0, 3.873, 5.477, 6.708, 7.746, 9.487, 10.954, 12.247, 13.416),
    2
  ),
  release = c(
    0, 11.4, 20.8, 30.8, 39.8, 57.8, 72.0, 84.8, 93.5,  # pH 7.4
    0, 17.2, 23.8, 35.5, 41.5, 58.3, 73.6, 86.2, 93.1   # pH 4.5
  ),
  pH = rep(c(7.4, 4.5), each = 9)
)
higuchi_release(
  data = df_pH,
  sqrt_time_col = "sqrt_time",
  release_col = "release",
  pH_col = "pH"
)

# Example IV: Two formulations under two pH conditions
df1 <- data.frame(
  sqrt_time = rep(c(0.0, 2.5, 4.0, 5.2, 6.3, 7.4, 8.6, 9.8, 11.0, 12.2), 2),
  release = c(
    0.0, 12.5, 21.8, 31.2, 39.6, 50.8, 63.5, 74.2, 84.9, 92.8,  # pH 7.4
```

```
    0.0, 9.8, 18.6, 26.9, 34.7, 45.3, 56.8, 67.9, 77.4, 85.2   # pH 4.5
  ),
  pH = rep(c(7.4, 4.5), each = 10)
)
df2 <- data.frame(
  sqrt_time = rep(c(0.0, 2.5, 4.0, 5.2, 6.3, 7.4, 8.6, 9.8, 11.0, 12.2), 2),
  release = c(
    0.0, 11.3, 20.4, 29.1, 37.8, 48.6, 60.1, 71.0, 81.5, 89.6,  # pH 7.4
    0.0, 8.9, 16.7, 24.6, 32.1, 42.5, 53.4, 64.0, 73.1, 80.8   # pH 4.5
  ),
  pH = rep(c(7.4, 4.5), each = 10)
)
df_all <- rbind(
  cbind(formulation = "Dataset 1", df1),
  cbind(formulation = "Dataset 2", df2)
)
higuchi_release(
  data = df_all,
  sqrt_time_col = "sqrt_time",
  release_col = "release",
  group_col = "formulation",
  pH_col = "pH"
)
```

`hixson_crowell_model` *Hixson-Crowell Drug Release Kinetic Model*

Description

Fits experimental cumulative drug release data to the Hixson-Crowell cube-root model, which describes drug release from systems where dissolution occurs with a change in surface area and particle diameter over time. The model is based on a linear regression of $(W_0^{1/3} - W_t^{1/3})$ versus time.

The function assumes that the initial drug amount (W_0) is known and that the remaining drug amount (W_t) can be calculated from cumulative percent drug remaining data. It supports optional grouping variables (e.g., formulation, batch) and optional pH-dependent analysis. The function generates publication-quality plots with experimental data points, fitted Hixson-Crowell straight lines, and annotations for the Hixson-Crowell rate constant (kHC), intercept, coefficient of determination (R^2), and the time required for 50-percent drug release (t50).

Model:

$$W_0^{1/3} - W_t^{1/3} = k_{HC} * t$$

where: - W_0 is the initial amount of drug - W_t is the remaining amount of drug at time t - k_{HC} is the Hixson-Crowell dissolution rate constant

The Hixson-Crowell model is applicable to dosage forms where drug release is governed by erosion or dissolution with a decreasing surface area (e.g., tablets, pellets).

Arguments

data	A data frame containing experimental drug release data.
time_col	Character string specifying the column name for time (minutes).
remain_col	Character string specifying the column name for cumulative percent drug remaining.
group_col	Optional character string specifying a grouping variable (e.g., formulation, batch).
pH_col	Optional character string specifying a column containing pH values.
plot	Logical; if TRUE, generates a plot with fitted Hixson-Crowell lines.
annotate	Logical; if TRUE, annotates the plot with kHC, intercept, R^2, and t50 (only if <= 2 groups).

Value

A list containing:

fitted_parameters Data frame with Hixson-Crowell parameters (kHC, intercept), coefficient of determination (R^2), and t50 for each group.

data Processed data used for model fitting and plotting.

Author(s)

Paul Angelo C. Manlapaz

References

Hixson, A. W., & Crowell, J. H. (1931) <doi:10.1021/ie50260a018> Dependence of reaction velocity upon surface and agitation. Industrial & Engineering Chemistry, 23(8), 923–931.

Examples

```
# Example I: Single formulation
df <- data.frame(
  time = c(0, 15, 30, 45, 60, 90, 120, 150, 180),
  remain = c(100, 88.6, 79.2, 69.2, 60.2, 42.2, 28.0, 15.2, 6.5)
)
hixson_crowell_model(
  data = df,
  time_col = "time",
  remain_col = "remain"
)

# Example II: Two formulations (grouped, not pH-dependent)
df2 <- data.frame(
  time = rep(c(0, 20, 40, 60, 80, 100, 120, 140, 160, 180), 2),
  remain = c(
    100, 90, 81, 72, 63, 54, 45, 36, 27, 18,  # Formulation A
    100, 92, 84, 76, 68, 60, 52, 44, 36, 28   # Formulation B
  ),
  formulation = rep(c("Formulation A", "Formulation B"), each = 10)
```

```
)
hixson_crowell_model(
  data = df2,
  time_col = "time",
  remain_col = "remain",
  group_col = "formulation"
)

# Example III: pH-dependent release
df_pH <- data.frame(
  time = rep(c(0, 20, 40, 60, 80, 100, 120, 140, 160, 180), 2),
  remain = c(
    100, 90, 80, 70, 60, 50, 40, 30, 20, 10,  # pH 7.4
    100, 92, 84, 76, 68, 60, 52, 44, 36, 28   # pH 4.5
  ),
  pH = rep(c(7.4, 4.5), each = 10)
)
hixson_crowell_model(
  data = df_pH,
  time_col = "time",
  remain_col = "remain",
  pH_col = "pH"
)

# Example IV: Two formulations under two pH conditions
df1 <- data.frame(
  time = rep(c(0, 20, 40, 60, 80, 100, 120, 140, 160, 180), 2),
  remain = c(
    100, 88, 75, 62, 50, 38, 28, 18, 10, 5,   # pH 4.5
    100, 90, 78, 65, 52, 40, 30, 20, 12, 6    # pH 7.6
  ),
  pH = rep(c(4.5, 7.6), each = 10)
)
df2 <- data.frame(
  time = rep(c(0, 15, 30, 45, 60, 75, 90, 105, 120, 135), 2),
  remain = c(
    100, 90, 78, 66, 54, 44, 34, 25, 16, 8,   # pH 4.5
    100, 92, 80, 68, 56, 44, 34, 24, 15, 7    # pH 7.6
  ),
  pH = rep(c(4.5, 7.6), each = 10)
)
df_all <- rbind(
  cbind(formulation = "Dataset 1", df1),
  cbind(formulation = "Dataset 2", df2)
)
hixson_crowell_model(
  data = df_all,
  time_col = "time",
  remain_col = "remain",
  group_col = "formulation",
  pH_col = "pH"
)
```

`korsmeyer_peppas_model` *Korsmeyer-Peppas Drug Release Kinetic Model*

Description

Fits experimental cumulative drug release data to the Korsmeyer-Peppas model using log10-transformed fraction released vs. log10-transformed time. The function automatically normalizes cumulative percent drug release to fraction (0-1) by default and removes t = 0. In addition, the function supports optional grouping variables (e.g., formulation, batch) and optional pH-dependent analysis. It generates publication-quality plots with experimental curves, fitted Korsmeyer-Peppas straight lines, mechanism table, and annotations for Korsmeyer-Peppas release constant (kKP), release exponent (n), intercept, coefficient of determination (R^2), and the time required for 50-percent drug release (t50).

Users can toggle 'normalize = TRUE/FALSE' to use fraction (0-1) or raw percent drug release.

Model: log10(Mt/MInf) = log10(k) + n * log10(t)

The release exponent, n, indicates the drug release mechanism: - n = 0.5 : Fickian diffusion - 0.5 < n < 1 : Non-Fickian (anomalous) diffusion - n = 1 : Case II transport (zero-order release) - n > 1 : Super Case II transport

Arguments

`data`	A data frame containing experimental cumulative percent drug release.
`time_col`	Character string specifying the column name for time (minutes).
`release_col`	Character string specifying the column name for cumulative percent drug release.
`group_col`	Optional character string specifying a grouping variable (e.g., formulation, batch).
`pH_col`	Optional character string specifying a column containing pH values.
`plot`	Logical; if TRUE, generates a plot with fitted Korsmeyer-Peppas lines.
`annotate`	Logical; if TRUE, annotates the plot with kKP, n, intercept, R^2, and t50 (only if <= 2 groups).
`normalize`	Logical; if TRUE (default), normalizes cumulative percent release to fraction (0-1). If FALSE, retains original percent values.

Value

A list containing:

`fitted_parameters` Data frame with kKP, n, intercept, R^2, and t50 for each group.

`data` Processed data used for fitting and plotting.

Author(s)

Paul Angelo C. Manlapaz

References

Korsmeyer, R. W., Gurny, R., Doelker, E., Buri, P., & Peppas, N. A. (1983) <doi:10.1016/0378-5173(83)90064-9> Mechanisms of solute release from porous hydrophilic polymers. International Journal of Pharmaceutics, 15(1), 25–35.

Examples

```
# Example I: Single formulation
df1 <- data.frame(
  time = c(0, 15, 30, 45, 60, 90, 120, 150, 180),
  release = c(0, 11.4, 20.8, 30.8, 39.8, 57.8, 72, 84.8, 93.5)
)
korsmeyer_peppas_model(
  data = df1,
  time_col = "time",
  release_col = "release",
  normalize = TRUE  # default
)

# Example II: Two formulations (grouped, not pH-dependent)
df2 <- data.frame(
  time = rep(c(0, 30, 60, 90, 120, 150), 2),
  release = c(
    0, 18, 35, 55, 72, 88,  # Formulation A
    0, 12, 26, 40, 58, 70   # Formulation B
  ),
  formulation = rep(c("Formulation A", "Formulation B"), each = 6)
)
korsmeyer_peppas_model(
  data = df2,
  time_col = "time",
  release_col = "release",
  group_col = "formulation"
)

# Example III: pH-dependent release
df_pH <- data.frame(
  time = rep(c(0, 60, 120, 180), 2),
  release = c(0, 40, 75, 95, 0, 30, 60, 80),
  pH = rep(c(7.4, 4.5), each = 4)
)
korsmeyer_peppas_model(
  data = df_pH,
  time_col = "time",
  release_col = "release",
  pH_col = "pH"
)

# Example IV: Two formulations under two pH conditions
df1 <- data.frame(
  time = rep(c(0, 20, 40, 60, 80, 100, 120, 140, 160, 180), 2),
  release = c(
```

```
    0, 12, 25, 38, 50, 62, 73, 82, 89, 95,  # pH 4.5
    0, 15, 30, 45, 59, 70, 79, 86, 91, 97   # pH 7.6
  ),
  pH = rep(c(4.5, 7.6), each = 10)
)
df2 <- data.frame(
  time = rep(c(0, 15, 30, 45, 60, 75, 90, 105, 120, 135), 2),
  release = c(
    0, 10, 22, 34, 46, 57, 67, 76, 84, 91,  # pH 4.5
    0, 13, 27, 41, 55, 67, 77, 85, 92, 98   # pH 7.6
  ),
  pH = rep(c(4.5, 7.6), each = 10)
)
df_all <- rbind(
  cbind(formulation = "Dataset 1", df1),
  cbind(formulation = "Dataset 2", df2)
)
korsmeyer_peppas_model(
  data = df_all,
  time_col = "time",
  release_col = "release",
  group_col = "formulation",
  pH_col = "pH"
)
```

`ld50_model` *Lethal Dose 50 (LD50) Pharmacodynamic Model*

Description

Fits quantal mortality data to a logistic dose-response model to estimate the Lethal Dose 50 (LD50), defined as the dose expected to cause death in 50

The model uses binomial logistic regression and supports optional grouping (e.g., species, sex, strain, formulation) and stratification by experimental conditions (e.g., exposure route, duration).

In addition to LD50 estimation, the model provides the following interpretable parameters:

- **Slope**: Represents the steepness of the dose-mortality relationship. A larger slope indicates a rapid transition from survival to lethality with increasing dose (high sensitivity and narrow safety margin), whereas a smaller slope reflects a more gradual increase in mortality, suggesting greater inter-individual variability in response.
- **Intercept**: Represents the baseline log-odds of mortality at zero dose. A strongly negative intercept indicates negligible background mortality, while a positive intercept suggests mortality occurring in the absence of administered dose, which may reflect experimental bias, underlying disease, or study design limitations.
- **LD50 95% Confidence Interval**: An approximate 95 interval for the LD50, computed using the delta method. This provides an uncertainty range around the estimated dose causing 50

- **McFadden Pseudo-R^2**: A likelihood-based measure of model goodness-of-fit that quantifies the improvement of the fitted model over a null (intercept-only) model. Values between 0.1 and 0.2 indicate acceptable biological fit, while values above 0.3 suggest a strong and reliable dose-mortality relationship.

The function can generate dose-response plots with fitted curves and annotate LD50, slope, intercept, LD50 confidence intervals, and McFadden pseudo-R^2 values.

Arguments

`data`	A data frame containing mortality response data.
`dose_col`	Character string specifying the administered dose column.
`response_col`	Character string specifying the binary mortality outcome (0 = alive, 1 = dead).
`group_col`	Optional character string specifying a grouping variable.
`condition_col`	Optional character string specifying an experimental condition.
`plot`	Logical; if TRUE, generates dose-mortality plots.
`annotate`	Logical; if TRUE, annotates the plot with LD50, confidence intervals, and model parameters (only if <=2 groups).

Value

A list containing:

`fitted_parameters` Data frame with LD50, 95% confidence intervals, slope, intercept, and pseudo-R2 values for each group.

`data` The processed data used for model fitting and plotting.

Author(s)

Paul Angelo C. Manlapaz

References

Bliss, C. I. (1935) <doi:10.1111/j.1744-7348.1935.tb07713.x> The calculation of the dosage-mortality curve. Annals of Applied Biology, 22(1), 134–167.

Finney, D. J. (1971) <isbn:9780521080415> Probit Analysis, 3rd Edition. Cambridge University Press, Cambridge.

Examples

```
# Example I: Single-species acute toxicity study
df1 <- data.frame(
  dose = c(10, 25, 50, 100, 200),
  dead = c(0, 0, 1, 1, 1)
)
ld50_model(
  data = df1,
  dose_col = "dose",
  response_col = "dead"
```

```
)

# Example II: Sex-dependent LD50 analysis
df2 <- data.frame(
  dose = rep(c(10, 25, 50, 100), 2),
  dead = c(0, 0, 1, 1, 0, 0, 0, 1),
  sex = rep(c("Male", "Female"), each = 4)
)
ld50_model(
  data = df2,
  dose_col = "dose",
  response_col = "dead",
  group_col = "sex"
)

# Example III: Species and exposure route comparison
df3 <- data.frame(
  dose = rep(c(20, 40, 80, 160), 4),
  dead = c(0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1),
  species = rep(c("Rat", "Mouse"), each = 8),
  route = rep(c("Oral", "IP"), each = 4, times = 2)
)
ld50_model(
  data = df3,
  dose_col = "dose",
  response_col = "dead",
  group_col = "species",
  condition_col = "route"
)
```

`logistic_4pl` *4PL Logistic Dose-Response Model (Emax/Imax)*

Description

Fits pharmacodynamic dose-response data to a 4-parameter logistic (4PL) model using nonlinear least squares regression.

The model can handle **both increasing (Stimulatory / Emax)** and **decreasing (Inhibitory / Imax)** dose-response curves automatically.

The 4PL model generalizes the Hill/Sigmoid model:

$$E = E_{min} + \frac{E_{max} - E_{min}}{1 + (EC_{50}/D)^n} \quad \text{for increasing curves}$$

$$E = E_{min} + \frac{E_{max} - E_{min}}{1 + (D/IC_{50})^n} \quad \text{for decreasing curves}$$

Key features: * Automatically detects increasing vs decreasing responses * E_{min} = minimum response (floor) * E_{max} = maximum response (ceiling) * EC_{50}/IC_{50} = dose producing half-maximal effect * n = Hill coefficient (steepness) * Symmetric sigmoid curve about midpoint

Arguments

data	A data frame containing dose-response experimental data.
dose_col	Character string specifying the column name for dose.
response_col	Character string specifying the column name for measured response.
group_col	Optional character string specifying a column for grouping.
log_dose	Logical; if TRUE, dose values are log10-transformed for plotting.
plot	Logical; if TRUE, generates a dose-response plot with fitted 4PL curves.
annotate	Logical; if TRUE, annotates the plot with model parameters and fit metrics.

Value

A list containing:

fitted_parameters Data frame with E_min, E_max, EC50, n, RMSE, AIC, and BIC for each group.

data Processed data used for fitting and plotting.

Author(s)

Paul Angelo C. Manlapaz

References

Holford, N. H. G. & Sheiner, L. B. (1981) <doi:10.2165/00003088-198106060-00002> Understanding the dose-effect relationship. Clinical Pharmacokinetics, 6(6), 429–453.

Finney, D. J. (1971) <isbn:9780521080415> Probit Analysis, 3rd Edition. Cambridge University Press, Cambridge.

Examples

```
# Example I: Single increasing curve (Emax)
df_emax <- data.frame(
  dose = c(0.1, 0.3, 1, 3, 10, 30, 100),
  response = c(5, 12, 28, 55, 75, 90, 98)
)
logistic_4pl(
  data = df_emax,
  dose_col = "dose",
  response_col = "response"
)

# Example II: Single decreasing curve (Imax)
df_imax <- data.frame(
  dose = c(0.1, 0.3, 1, 3, 10, 30, 100),
  response = c(95, 88, 70, 50, 30, 15, 5)
)
logistic_4pl(
  data = df_imax,
```

```
  dose_col = "dose",
  response_col = "response"
)

# Example III: Two treatment groups, mixed Emax/Imax
df_groups <- data.frame(
  dose = rep(c(0.1, 0.3, 1, 3, 10, 30), 2),
  response = c(
    4, 10, 25, 55, 78, 92,   # Group A: increasing (Emax)
    90, 75, 55, 35, 20, 10   # Group B: decreasing (Imax)
  ),
  treatment = rep(c("Group A", "Group B"), each = 6)
)
logistic_4pl(
  data = df_groups,
  dose_col = "dose",
  response_col = "response",
  group_col = "treatment",
  log_dose = TRUE
)
```

`logistic_5pl` *5PL Logistic Dose-Response Model (Emax/Imax, Asymmetric)*

Description

Fits pharmacodynamic dose-response data to a 5-parameter logistic (5PL) model using nonlinear least squares regression.

The model can handle **both increasing (Stimulatory / Emax)** and **decreasing (Inhibitory / Imax)** dose-response curves automatically.

The 5PL model extends 4PL by adding an asymmetry factor s:

$$E = E_{min} + \frac{E_{max} - E_{min}}{(1 + (EC_{50}/D)^n)^s} \quad \text{for increasing curves}$$

$$E = E_{min} + \frac{E_{max} - E_{min}}{(1 + (D/IC_{50})^n)^s} \quad \text{for decreasing curves}$$

Key features: * Automatically detects increasing vs decreasing responses * E_{min} = minimum response (floor) * E_{max} = maximum response (ceiling) * EC_{50}/IC_{50} = dose producing half-maximal effect * n = Hill coefficient (steepness) * s = asymmetry factor (skew) * Skewed sigmoidal curve; reduces bias when the rising and falling slopes are not symmetric around EC50

Arguments

`data`	A data frame containing dose-response experimental data.
`dose_col`	Character string specifying the column name for dose.

response_col	Character string specifying the column name for measured response.
group_col	Optional character string specifying a column for grouping.
log_dose	Logical; if TRUE, dose values are log10-transformed for plotting.
plot	Logical; if TRUE, generates a dose-response plot with fitted 5PL curves.
annotate	Logical; if TRUE, annotates the plot with model parameters and fit metrics.

Value

A list containing:

fitted_parameters Data frame with E_min, E_max, EC50, n, s, RMSE, AIC, and BIC for each group.

data Processed data used for fitting and plotting.

Author(s)

Paul Angelo C. Manlapaz

References

Richards, F. J. (1959) <doi:10.1093/jxb/10.2.290> A flexible growth function for empirical use. Journal of Experimental Botany, 10(2), 290–301.

Examples

```
# Example I: Single increasing curve (Emax)
df_emax_5pl <- data.frame(
  dose = c(0.1, 0.3, 1, 3, 10, 30, 100),
  response = c(5, 12, 28, 55, 75, 90, 98)
)
logistic_5pl(
  data = df_emax_5pl,
  dose_col = "dose",
  response_col = "response"
)

# Example II: Single decreasing curve (Imax)
df_imax_5pl <- data.frame(
  dose = c(0.1, 0.3, 1, 3, 10, 30, 100),
  response = c(95, 88, 70, 50, 30, 15, 5)
)
logistic_5pl(
  data = df_imax_5pl,
  dose_col = "dose",
  response_col = "response"
)

# Example III: Two treatment groups, mixed Emax/Imax
df_groups_5pl <- data.frame(
  dose = rep(c(0.1, 0.3, 1, 3, 10, 30), 2),
```

```
  response = c(
    4, 10, 25, 55, 78, 92,   # Group A: increasing (Emax)
    90, 75, 55, 35, 20, 10   # Group B: decreasing (Imax)
  ),
  treatment = rep(c("Group A", "Group B"), each = 6)
)
logistic_5pl(
  data = df_groups_5pl,
  dose_col = "dose",
  response_col = "response",
  group_col = "treatment",
  log_dose = TRUE
)
```

`michaelis_menten` *Michaelis-Menten Kinetics (Linear Form) with Inhibition Comparison*

Description

Performs Michaelis-Menten kinetic analysis using linearized transformations (Lineweaver-Burk by default) to compare multiple datasets and infer inhibition type.

The function fits linear regressions to transformed data and compares slopes and intercepts across conditions.

Interpretation of inhibition patterns (Lineweaver-Burk):

- Competitive inhibition: Same y-intercept (1/Vmax), increased slope -> apparent Km increases
- Noncompetitive inhibition: Same x-intercept (-1/Km), increased y-intercept -> Vmax decreases
- Uncompetitive inhibition: Parallel lines -> both Km and Vmax decrease proportionally

Arguments

`data`	A data.frame containing concentration and rate data.
`conc_col`	Column name for substrate or drug concentration.
`rate_col`	Column name for reaction rate or elimination rate.
`group_col`	Column indicating different conditions (e.g., inhibitor levels).
`transform`	Linearization method: "Lineweaver-Burk" (default), "Eadie-Hofstee", or "Hanes-Woolf".
`inhibition_type`	Expected inhibition type: "competitive", "noncompetitive", "uncompetitive", "multi-inhibition" or "none".
`plot`	Logical; if TRUE, generates linearized comparison plot.

Value

A list containing:

`fitted_parameters` A data frame summarizing linear regression results for each group or condition, including the estimated slope, intercept, coefficient of determination (R^2), and derived Michaelis-Menten parameters (`Km` and `Vmax`) computed according to the selected linear transformation.

`transformed_data` A data frame containing the processed and linearized concentration and rate data (`x` and `y`) used for model fitting and visualization, along with the original group labels.

`interpretation` A character string describing the expected inhibition pattern based on the specified `inhibition_type` and the comparison of slopes and intercepts across groups.

Author(s)

Paul Angelo C. Manlapaz

References

Michaelis, L. and Menten, M. (1913) Die kinetik der invertinwirkung. Biochemistry Zeitung, 79, 333-369.

Examples

```
# Example I: Single Michaelis-Menten dataset (no inhibition)
df1 <- data.frame(
  concentration = c(0.5, 1, 2, 4, 6, 8, 10),
  rate = c(0.48, 0.85, 1.45, 2.20, 2.70, 3.05, 3.25),
  group = "No Inhibitor"
)
# Lineweaver-Burk
michaelis_menten(
  data = df1,
  conc_col = "concentration",
  rate_col = "rate",
  group_col = "group",
  transform = "Lineweaver-Burk",
  inhibition_type = "none",
  plot = TRUE
)
# Eadie-Hofstee
michaelis_menten(
  data = df1,
  conc_col = "concentration",
  rate_col = "rate",
  group_col = "group",
  transform = "Eadie-Hofstee",
  inhibition_type = "none",
  plot = TRUE
)
# Hanes-Woolf
michaelis_menten(
```

```
  data = df1,
  conc_col = "concentration",
  rate_col = "rate",
  group_col = "group",
  transform = "Hanes-Woolf",
  inhibition_type = "none",
  plot = TRUE
)

# Example II: Two datasets compared (inhibition analysis)
df2 <- data.frame(
  concentration = rep(c(0.5, 1, 2, 4, 6, 8, 10), 2),
  rate = c(
    # Reference (no inhibitor)
    0.50, 0.90, 1.50, 2.30, 2.80, 3.10, 3.30,
    # Condition B (possible inhibitor)
    0.35, 0.65, 1.10, 1.70, 2.10, 2.40, 2.55
  ),
  group = rep(c("No Inhibitor", "Inhibitor"), each = 7)
)
# Lineweaver-Burk
michaelis_menten(
  data = df2,
  conc_col = "concentration",
  rate_col = "rate",
  group_col = "group",
  transform = "Lineweaver-Burk",
  inhibition_type = "uncompetitive",
  plot = TRUE
)
# Eadie-Hofstee
michaelis_menten(
  data = df2,
  conc_col = "concentration",
  rate_col = "rate",
  group_col = "group",
  transform = "Eadie-Hofstee",
  inhibition_type = "competitive",
  plot = TRUE
)
# Hanes-Woolf
michaelis_menten(
  data = df2,
  conc_col = "concentration",
  rate_col = "rate",
  group_col = "group",
  transform = "Hanes-Woolf",
  inhibition_type = "competitive",
  plot = TRUE
)

# Example III: Six datasets compared (one reference, five test conditions)
df3 <- data.frame(
```

```
  concentration = rep(c(0.5, 1, 2, 4, 6, 8, 10), 6),
  rate = c(
    # Reference
    0.50, 0.90, 1.50, 2.30, 2.80, 3.10, 3.30,
    # Mixed Noncompetitive inhibitor A
    0.35, 0.65, 1.10, 1.70, 2.10, 2.40, 2.55,
    # Uncompetitive inhibitor
    0.40, 0.70, 1.15, 1.75, 2.15, 2.45, 2.60,
    # Mixed Noncompetitive inhibitor B
    0.30, 0.55, 0.95, 1.45, 1.85, 2.10, 2.20,
    # Mixed Noncompetitive + higher dose
    0.28, 0.50, 0.85, 1.30, 1.65, 1.85, 1.95,
    # Uncompetitive + higher dose
    0.38, 0.65, 1.10, 1.65, 2.05, 2.30, 2.40
  ),
  group = rep(c(
    "Reference",
    "Noncompetitive (Mixed) A",
    "Uncompetitive",
    "Noncompetitive (Mixed) B",
    "Noncompetitive (Mixed) High",
    "Uncompetitive High"
  ), each = 7)
)
# Lineweaver-Burk
michaelis_menten(
  data = df3,
  conc_col = "concentration",
  rate_col = "rate",
  group_col = "group",
  transform = "Lineweaver-Burk",
  inhibition_type = "multi-inhibition",
  plot = TRUE
)
# Eadie-Hofstee
michaelis_menten(
  data = df3,
  conc_col = "concentration",
  rate_col = "rate",
  group_col = "group",
  transform = "Eadie-Hofstee",
  inhibition_type = "multi-inhibition",
  plot = TRUE
)
# Hanes-Woolf
michaelis_menten(
  data = df3,
  conc_col = "concentration",
  rate_col = "rate",
  group_col = "group",
  transform = "Hanes-Woolf",
  inhibition_type = "multi-inhibition",
  plot = TRUE
```

```
)
```

`michaelis_menten_nl` *Michaelis-Menten Kinetics (Nonlinear Form) with Inhibition Comparison*

Description

Performs Michaelis-Menten kinetic analysis using nonlinear least squares fitting to estimate Km and Vmax for one or more datasets.

This function fits the standard Michaelis-Menten equation:

$$rate = (Vmax * concentration)/(Km + concentration)$$

to each group separately and allows comparison of kinetic parameters across conditions (e.g., inhibitors).

Interpretation of inhibition patterns:

- Competitive inhibition: Km increases, Vmax unchanged
- Noncompetitive inhibition: Vmax decreases, Km unchanged
- Uncompetitive inhibition: both Km and Vmax decrease

Arguments

`data`	A data.frame containing concentration and rate data.
`conc_col`	Column name for substrate or drug concentration.
`rate_col`	Column name for reaction rate or elimination rate.
`group_col`	Column indicating different conditions (e.g., inhibitor levels).
`inhibition_type`	Expected inhibition type: "competitive", "noncompetitive", "uncompetitive", "multi-inhibition" or "none".
`plot`	Logical; if TRUE, generates nonlinear fit comparison plot.

Value

A list containing:

`fitted_parameters` A data frame with estimated nonlinear Michaelis-Menten parameters (`Km` and `Vmax`) for each group or condition, obtained via nonlinear least squares fitting.

`raw_data` A data frame containing the processed concentration and rate data used for model fitting and plotting, including group labels.

`interpretation` A character string summarizing the expected inhibition pattern based on the specified `inhibition_type` and the comparison of `Km` and `Vmax` values across groups.

Author(s)

Paul Angelo C. Manlapaz

References

Michaelis, L. and Menten, M. (1913) Die kinetik der invertinwirkung. Biochemistry Zeitung, 79, 333-369.

Examples

```
# Example I: Single Michaelis-Menten dataset (no inhibition)
df1 <- data.frame(
  concentration = c(0.5, 1, 2, 4, 6, 8, 10),
  rate = c(0.48, 0.85, 1.45, 2.20, 2.70, 3.05, 3.25),
  group = "No Inhibitor"
)
michaelis_menten_nl(
  data = df1,
  conc_col = "concentration",
  rate_col = "rate",
  group_col = "group",
  inhibition_type = "none",
  plot = TRUE
)

# Example II: Two datasets compared (inhibition analysis)
df2 <- data.frame(
  concentration = rep(c(0.5, 1, 2, 4, 6, 8, 10), 2),
  rate = c(
    # Reference (no inhibitor)
    0.50, 0.90, 1.50, 2.30, 2.80, 3.10, 3.30,
    # Condition B (possible inhibitor)
    0.35, 0.65, 1.10, 1.70, 2.10, 2.40, 2.55
  ),
  group = rep(c("No Inhibitor", "Inhibitor"), each = 7)
)
michaelis_menten_nl(
  data = df2,
  conc_col = "concentration",
  rate_col = "rate",
  group_col = "group",
  inhibition_type = "uncompetitive",
  plot = TRUE
)

# Example III: Six datasets compared (one reference, five test conditions)
df3 <- data.frame(
  concentration = rep(c(0.5, 1, 2, 4, 6, 8, 10), 6),
  rate = c(
    # Reference
    0.50, 0.95, 1.80, 2.90, 3.60, 4.00, 4.30,
    # Competitive inhibitor
```

```
    0.35, 0.70, 1.35, 2.40, 3.05, 3.40, 3.65,
    # Mixed Noncompetitive inhibitor A
    0.30, 0.55, 1.00, 1.65, 2.05, 2.35, 2.50,
    # Uncompetitive inhibitor
    0.25, 0.50, 0.90, 1.40, 1.75, 2.00, 2.10,
    # Mixed Noncompetitive inhibitor high dose
    0.20, 0.40, 0.80, 1.50, 1.95, 2.25, 2.40,
    # Mixed Noncompetitive inhibitor B
    0.25, 0.45, 0.85, 1.35, 1.70, 1.95, 2.05
  ),
  group = rep(c(
    "Reference",
    "Competitive",
    "Noncompetitive (Mixed) A",
    "Uncompetitive",
    "Noncompetitive (Mixed) High",
    "Noncompetitive (Mixed) B"
  ), each = 7)
)
michaelis_menten_nl(
  data = df3,
  conc_col = "concentration",
  rate_col = "rate",
  group_col = "group",
  inhibition_type = "multi-inhibition",
  plot = TRUE
)
```

`non_compartmental` *Non-Compartmental Analysis (NCA) of Plasma Concentration-Time Data (Linear Form)*

Description

Performs non-compartmental analysis (NCA) on plasma concentration-time data assuming linear pharmacokinetics. Computes area under the curve (AUC) using trapezoidal rule, estimates terminal elimination rate constant (kel) by linear regression on the log-linear terminal phase, calculates half-life (t1/2), clearance (CL), and volume of distribution (Vd).

Arguments

`data`	A data.frame containing plasma concentration-time data.
`time_col`	Character string indicating the column name for time.
`conc_col`	Character string indicating the column name for concentration.
`dose`	Numeric value for the administered dose.
`group_col`	Optional character string specifying a grouping variable for multiple groups.
`terminal_points`	Number of last points to use for terminal slope estimation (default = 3).

plot	Logical; if TRUE, plots concentration-time profile and terminal phase regression.
annotate	Logical; if TRUE, annotates plot with PK parameters (only for <= 2 groups).

Value

A list containing:

fitted_parameters Data frame with kel, t1/2, AUC, CL, Vd, and R^2 for each group.

data Processed data used for fitting and plotting.

Author(s)

Paul Angelo C. Manlapaz

References

Gibaldi, M. & Perrier, D. (1982) <isbn:9780824710422> Pharmacokinetics, 2nd Edition. Marcel Dekker, New York.

Gabrielsson, J. & Weiner, D. (2000) <isbn:9186274929> Pharmacokinetic/Pharmacodynamic Data Analysis: Concepts and Applications, 3rd Edition, Revised and Expanded. Swedish Pharmaceutical Press, Stockholm.

Examples

```
# Example I: Single-subject non-compartmental analysis
df <- data.frame(
  time = c(0.25, 0.5, 1, 2, 4, 6, 8, 12),
  concentration = c(18.6, 16.9, 14.2, 10.8, 6.9, 4.6, 3.1, 1.9)
)
non_compartmental(
  data = df,
  time_col = "time",
  conc_col = "concentration",
  dose = 100,          # mg
  terminal_points = 3, # last 3 points (6, 8, 12 h)
  plot = TRUE,
  annotate = TRUE
)

# Example II: Two-group comparison (reference vs test formulation)
df_groups <- data.frame(
  time = rep(c(0.25, 0.5, 1, 2, 4, 6, 8), 2),
  concentration = c(
    17.9, 16.2, 13.7, 10.1, 6.3, 4.1, 2.8,  # Reference
    20.4, 18.9, 16.1, 12.4, 8.1, 5.6, 3.9   # Test
  ),
  formulation = rep(c("Reference", "Test"), each = 7)
)
non_compartmental(
  data = df_groups,
```

```
  time_col = "time",
  conc_col = "concentration",
  dose = 100,            # same dose in both groups
  group_col = "formulation",
  terminal_points = 3,
  plot = TRUE,
  annotate = TRUE
)

# Example III: Multiple subjects with extended terminal phase
df_subjects <- data.frame(
  time = rep(c(0.5, 1, 2, 4, 8, 12, 24), 3),
  concentration = c(
    15.2, 13.9, 11.4, 7.6, 4.1, 2.6, 1.3,  # Subject 1
    14.8, 13.2, 10.9, 7.2, 3.9, 2.4, 1.2,  # Subject 2
    16.0, 14.6, 12.0, 8.1, 4.5, 2.9, 1.5   # Subject 3
  ),
  subject = rep(paste0("S", 1:3), each = 7)
)
non_compartmental(
  data = df_subjects,
  time_col = "time",
  conc_col = "concentration",
  dose = 150,
  group_col = "subject",
  terminal_points = 4,
  plot = TRUE,
  annotate = FALSE
)
```

`non_compartmental_nl` *Non-Compartmental Analysis (NCA) of Plasma Concentration-Time Data (Nonlinear Form)*

Description

Performs non-compartmental analysis (NCA) on plasma concentration-time data, fitting the terminal elimination phase using nonlinear regression of an exponential decay. Computes area under the curve (AUC) using trapezoidal rule, estimates terminal elimination rate constant (kel), half-life (t1/2), clearance (CL), and volume of distribution (Vd).

Arguments

`data`	A data.frame containing plasma concentration-time data.
`time_col`	Character string indicating the column name for time.
`conc_col`	Character string indicating the column name for concentration.
`dose`	Numeric value for the administered dose.
`group_col`	Optional character string specifying a grouping variable for multiple groups.

terminal_points
: Number of last points to use for terminal phase estimation (default = 3).

plot
: Logical; if TRUE, plots concentration-time profile and terminal phase fit.

annotate
: Logical; if TRUE, annotates plot with PK parameters (only for <= 2 groups).

Value

A list containing:

fitted_parameters Data frame with kel, t1/2, AUC, CL, Vd, and R^2 for each group.

data Processed data used for fitting and plotting.

Author(s)

Paul Angelo C. Manlapaz

References

Gibaldi, M. & Perrier, D. (1982) <isbn:9780824710422> Pharmacokinetics, 2nd Edition. Marcel Dekker, New York.

Gabrielsson, J. & Weiner, D. (2000) <isbn:9186274929> Pharmacokinetic/Pharmacodynamic Data Analysis: Concepts and Applications, 3rd Edition, Revised and Expanded. Swedish Pharmaceutical Press, Stockholm.

Examples

```
# Example I: Single-subject nonlinear non-compartmental analysis
df <- data.frame(
  time = c(0.25, 0.5, 1, 2, 4, 6, 8, 12),
  concentration = c(18.4, 16.9, 14.3, 10.9, 7.1, 4.7, 3.2, 2.0)
)
non_compartmental_nl(
  data = df,
  time_col = "time",
  conc_col = "concentration",
  dose = 100,
  terminal_points = 3,
  plot = TRUE,
  annotate = TRUE
)

# Example II: Two-group nonlinear NCA (e.g., formulation comparison)
df_groups <- data.frame(
  time = rep(c(0.25, 0.5, 1, 2, 4, 6, 8), 2),
  concentration = c(
    17.8, 16.3, 13.9, 10.5, 6.6, 4.3, 3.0,  # Group A
    20.1, 18.7, 16.2, 12.7, 8.4, 5.9, 4.1   # Group B
  ),
  formulation = rep(c("Reference", "Test"), each = 7)
)
non_compartmental_nl(
```

```
  data = df_groups,
  time_col = "time",
  conc_col = "concentration",
  dose = 100,
  group_col = "formulation",
  terminal_points = 3,
  plot = TRUE,
  annotate = TRUE
)

# Example III: Six-subject nonlinear NCA
df_subjects <- data.frame(
  time = rep(c(0.5, 1, 2, 4, 8, 12, 24), 6),
  concentration = c(
    15.6, 14.1, 11.7, 7.9, 4.3, 2.8, 1.4,  # S1
    14.9, 13.5, 11.1, 7.4, 4.0, 2.6, 1.3,  # S2
    16.3, 14.9, 12.4, 8.4, 4.7, 3.1, 1.6,  # S3
    15.1, 13.7, 11.3, 7.6, 4.2, 2.7, 1.3,  # S4
    14.6, 13.2, 10.8, 7.2, 3.9, 2.5, 1.2,  # S5
    16.0, 14.6, 12.0, 8.1, 4.5, 3.0, 1.5   # S6
  ),
  subject = rep(paste0("S", 1:6), each = 7)
)
non_compartmental_nl(
  data = df_subjects,
  time_col = "time",
  conc_col = "concentration",
  dose = 150,
  group_col = "subject",
  terminal_points = 4,
  plot = TRUE,
  annotate = FALSE
)
```

`one_compartment_iv_bolus` *One-Compartment IV Bolus Pharmacokinetic Model (Linear)*

Description

Fits plasma concentration-time data to the one-compartment intravenous (IV) bolus pharmacokinetic model. The model assumes instantaneous drug distribution throughout a single, well-mixed compartment and first-order elimination kinetics.

The function performs linear regression on log-transformed plasma concentration versus time to estimate the elimination rate constant (k_el), elimination half-life (t1/2), initial concentration (C0), apparent volume of distribution (Vd), and clearance (CL). Optional grouping (e.g., formulation, subject) and pH-dependent analysis are supported. Publication-quality plots with fitted regression lines and parameter annotations are generated.

Model: C(t) = C0 * exp(-k_el * t)

Linearized form: log(C) = log(C0) - k_el * t

where:

- C(t) is plasma concentration at time t
- C0 is the initial plasma concentration
- k_el is the elimination rate constant

Pharmacokinetic parameters:

- Elimination half-life: t1/2 = ln(2) / k_el
- Clearance: CL = Dose / AUC
- Volume of distribution: Vd = CL / k_el

Arguments

`data`	A data frame containing plasma concentration-time data.
`time_col`	Character string specifying the column name for time.
`conc_col`	Character string specifying the column name for plasma concentration.
`dose`	Numeric value specifying the administered IV bolus dose.
`group_col`	Optional character string specifying a grouping variable (e.g., formulation, subject).
`plot`	Logical; if TRUE, generates a concentration-time plot with fitted lines.
`annotate`	Logical; if TRUE, annotates the plot with PK parameters (only if <= 2 groups).

Value

A list containing:

`fitted_parameters` Data frame with C0, k_el, t1/2, Vd, CL, and R2.

`data` Processed data used for fitting and plotting.

Author(s)

Paul Angelo C. Manlapaz

References

Widmark, E. M. P. (1919) Studies in the concentration of indifferent narcotics in blood and tissues. Acta Medica Scandinavica, 52(1), 87–164.

Gibaldi, M. & Perrier, D. (1982) <isbn:9780824710422> Pharmacokinetics, 2nd Edition. Marcel Dekker, New York.

Gabrielsson, J. & Weiner, D. (2000) <isbn:9186274929> Pharmacokinetic/Pharmacodynamic Data Analysis: Concepts and Applications, 3rd Edition, Revised and Expanded. Swedish Pharmaceutical Press, Stockholm.

Examples

```
# Example I: Single subject IV bolus data
df <- data.frame(
  time = c(0.25, 0.5, 1, 2, 4, 6, 8, 12),
  concentration = c(18.2, 16.1, 13.5, 10.2, 6.8, 4.9, 3.6, 2.1)
)
one_compartment_iv_bolus(
  data = df,
  time_col = "time",
  conc_col = "concentration",
  dose = 100
)

# Example II: Condition-dependent pharmacokinetics (e.g., pH or physiological state)
df_cond <- data.frame(
  time = rep(c(0.25, 0.5, 1, 2, 4, 6), 2),
  concentration = c(
    17.8, 15.6, 13.1, 9.8, 6.4, 4.8,   # Condition A
    14.9, 13.0, 10.9, 8.0, 5.2, 3.9    # Condition B
  ),
  condition = rep(c("Condition A", "Condition B"), each = 6)
)
one_compartment_iv_bolus(
  data = df_cond,
  time_col = "time",
  conc_col = "concentration",
  dose = 100,
  group_col = "condition"
)

# Example III: Multiple subjects (population-style IV bolus pharmacokinetics)
df_subjects <- data.frame(
  time = rep(c(0.25, 0.5, 1, 2, 4, 6, 8), 6),
  concentration = c(
    18.6, 16.3, 13.9, 10.5, 7.0, 5.1, 3.8,   # Subject 1
    17.9, 15.7, 13.2, 9.9, 6.6, 4.9, 3.6,    # Subject 2
    17.1, 15.0, 12.6, 9.4, 6.3, 4.7, 3.4,    # Subject 3
    16.4, 14.4, 12.0, 9.0, 6.0, 4.4, 3.2,    # Subject 4
    15.8, 13.9, 11.6, 8.7, 5.8, 4.2, 3.1,    # Subject 5
    15.2, 13.3, 11.0, 8.3, 5.5, 4.0, 2.9     # Subject 6
  ),
  subject = rep(paste0("S", 1:6), each = 7)
)
one_compartment_iv_bolus(
  data = df_subjects,
  time_col = "time",
  conc_col = "concentration",
  dose = 100,
  group_col = "subject"
)
```

`one_compartment_iv_bolus_nl` *One-Compartment IV Bolus Pharmacokinetic Model (Nonlinear)*

Description

Fits plasma concentration-time data to a one-compartment intravenous (IV) bolus pharmacokinetic model using nonlinear regression. The model assumes instantaneous drug distribution throughout a single, well-mixed compartment and first-order elimination kinetics.

Model parameters are estimated by nonlinear least squares: - Elimination rate constant (k_el) - Initial plasma concentration (C0) - Apparent volume of distribution (Vd = Dose / C0)

Secondary pharmacokinetic parameters are derived: - Elimination half-life (t1/2 = ln(2)/k_el) - Clearance (CL = k_el * Vd)

The function supports optional grouping (e.g., subjects, conditions). Publication-quality plots with fitted curves are generated, and annotations summarizing key PK parameters appear in the upper-right corner when <= 2 groups.

Model: C(t) = C0 * exp(-k_el * t)

Arguments

`data`	A data frame containing plasma concentration-time data.
`time_col`	Character string specifying the column name for time.
`conc_col`	Character string specifying the column name for plasma concentration.
`dose`	Numeric value specifying the administered IV bolus dose.
`group_col`	Optional character string specifying a grouping variable (e.g., subject, condition).
`plot`	Logical; if TRUE, generates a concentration-time plot with fitted curves.
`annotate`	Logical; if TRUE, annotates the plot with PK parameters (only if <= 2 groups).

Value

A list containing:

`fitted_parameters` Data frame with C0, k_el, t1/2, Vd, CL, RMSE, AIC, and BIC.

`data` Processed data used for fitting and plotting.

Author(s)

Paul Angelo C. Manlapaz

References

Widmark, E. M. P. (1919) Studies in the concentration of indifferent narcotics in blood and tissues. Acta Medica Scandinavica, 52(1), 87–164.

Gibaldi, M. & Perrier, D. (1982) <isbn:9780824710422> Pharmacokinetics, 2nd Edition. Marcel Dekker, New York.

Gabrielsson, J. & Weiner, D. (2000) <isbn:9186274929> Pharmacokinetic/Pharmacodynamic Data Analysis: Concepts and Applications, 3rd Edition, Revised and Expanded. Swedish Pharmaceutical Press, Stockholm.

Examples

```
# Example I: Single subject one-compartment IV bolus data
df <- data.frame(
  time = c(0.08, 0.25, 0.5, 1, 2, 4, 6, 8, 12),
  concentration = c(18.2, 16.1, 13.5, 10.2, 6.8, 4.9, 3.6, 2.1, 1.2)
)
one_compartment_iv_bolus_nl(
  data = df,
  time_col = "time",
  conc_col = "concentration",
  dose = 100
)

# Example II: Condition-dependent pharmacokinetics (e.g., pH or physiological state)
df_cond <- data.frame(
  time = rep(c(0.25, 0.5, 1, 2, 4, 6), 2),
  concentration = c(
    17.8, 15.6, 13.1, 9.8, 6.4, 4.8,   # Condition A
    14.9, 13.0, 10.9, 8.0, 5.2, 3.9    # Condition B
  ),
  condition = rep(c("Condition A", "Condition B"), each = 6)
)
one_compartment_iv_bolus_nl(
  data = df_cond,
  time_col = "time",
  conc_col = "concentration",
  dose = 100,
  group_col = "condition"
)

# Example III: Multiple subjects (population-style one-compartment IV bolus pharmacokinetics)
df_subjects <- data.frame(
  time = rep(c(0.25, 0.5, 1, 2, 4, 6, 8), 6),
  concentration = c(
    18.6, 16.3, 13.9, 10.5, 7.0, 5.1, 3.8,   # Subject 1
    17.9, 15.7, 13.2, 9.9, 6.6, 4.9, 3.6,    # Subject 2
    17.1, 15.0, 12.6, 9.4, 6.3, 4.7, 3.4,    # Subject 3
    16.4, 14.4, 12.0, 9.0, 6.0, 4.4, 3.2,    # Subject 4
    15.8, 13.9, 11.6, 8.7, 5.8, 4.2, 3.1,    # Subject 5
    15.2, 13.3, 11.0, 8.3, 5.5, 4.0, 2.9     # Subject 6
  ),
```

```
  subject = rep(paste0("S", 1:6), each = 7)
)
one_compartment_iv_bolus_nl(
  data = df_subjects,
  time_col = "time",
  conc_col = "concentration",
  dose = 100,
  group_col = "subject"
)
```

`one_compartment_oral` *One-Compartment Oral Pharmacokinetic Model (Linear, First-Order Absorption)*

Description

Fits plasma concentration-time data to the one-compartment oral pharmacokinetic model using a linearized approach. The model assumes first-order absorption and first-order elimination.

Model: C(t) = (F * Dose * ka / (Vd * (ka - kel))) * (exp(-kel * t) - exp(-ka * t))

Linearized approximation: Using log-transformed data in the elimination phase (t » t_max), log(C) ~= log(C0) - kel * t

Parameters:

- ka: absorption rate constant
- kel: elimination rate constant
- C0: apparent initial concentration for elimination phase
- t_half: elimination half-life
- Vd: apparent volume of distribution
- CL: clearance

Arguments

`data`	A data frame containing plasma concentration-time data.
`time_col`	Character string specifying the column name for time.
`conc_col`	Character string specifying the column name for plasma concentration.
`dose`	Numeric value specifying the administered oral dose.
`group_col`	Optional character string specifying a grouping variable (e.g., formulation, subject).
`plot`	Logical; if TRUE, generates a concentration-time plot with fitted lines.
`annotate`	Logical; if TRUE, annotates the plot with PK parameters (only if <= 2 groups).

Value

A list containing:

`fitted_parameters` Data frame with C0, kel, t_half, Vd, CL, and R^2.

`data` Processed data used for fitting and plotting.

Author(s)

Paul Angelo C. Manlapaz

References

Gibaldi, M. & Perrier, D. (1982) <isbn:9780824710422> Pharmacokinetics, 2nd Edition. Marcel Dekker, New York.

Gabrielsson, J. & Weiner, D. (2000) <isbn:9186274929> Pharmacokinetic/Pharmacodynamic Data Analysis: Concepts and Applications, 3rd Edition, Revised and Expanded. Swedish Pharmaceutical Press, Stockholm.

Examples

```
# Example I: Single subject oral data
df <- data.frame(
  time = c(0.25, 0.5, 1, 2, 4, 6, 8, 12),
  concentration = c(5.1, 9.8, 14.2, 13.5, 10.2, 6.8, 4.5, 2.1)
)
one_compartment_oral(
  data = df,
  time_col = "time",
  conc_col = "concentration",
  dose = 100
)

# Example II: Condition-dependent kinetics
df_cond <- data.frame(
  time = rep(c(0.25, 0.5, 1, 2, 4, 6, 8), 2),
  concentration = c(
    4.8, 9.5, 13.7, 12.8, 9.2, 6.4, 3.9,   # Condition A
    5.2, 10.1, 14.0, 12.5, 8.7, 5.8, 3.5   # Condition B
  ),
  condition = rep(c("Condition A", "Condition B"), each = 7)
)
one_compartment_oral(
  data = df_cond,
  time_col = "time",
  conc_col = "concentration",
  dose = 100,
  group_col = "condition"
)

# Example III: Multiple subjects
df_subjects <- data.frame(
```

```
  time = rep(c(0.25, 0.5, 1, 2, 4, 6, 8), 10),
  concentration = c(
    5.0, 9.7, 14.0, 13.2, 10.0, 6.6, 4.2,  # Subject 1
    4.9, 9.5, 13.8, 12.9, 9.5, 6.3, 4.0,   # Subject 2
    5.1, 9.9, 14.1, 13.5, 10.3, 6.9, 4.3,  # Subject 3
    4.8, 9.6, 13.9, 13.1, 9.8, 6.5, 4.1,   # Subject 4
    5.2, 10.0, 14.3, 13.6, 10.5, 7.1, 4.4, # Subject 5
    5.1, 9.8, 14.0, 13.3, 10.1, 6.7, 4.3,  # Subject 6
    4.9, 9.6, 13.7, 12.8, 9.4, 6.2, 3.9,   # Subject 7
    5.0, 9.9, 14.2, 13.4, 10.2, 6.8, 4.1,  # Subject 8
    5.2, 10.1, 14.5, 13.7, 10.7, 7.2, 4.5, # Subject 9
    4.8, 9.5, 13.6, 12.7, 9.1, 6.0, 3.8    # Subject 10
  ),
  subject = rep(paste0("S", 1:10), each = 7)
)
one_compartment_oral(
  data = df_subjects,
  time_col = "time",
  conc_col = "concentration",
  dose = 100,
  group_col = "subject"
)
```

`one_compartment_oral_nl` *One-Compartment Oral Pharmacokinetic Model (Nonlinear, First-Order Absorption)*

Description

Fits plasma concentration-time data to a one-compartment pharmacokinetic model following oral administration with first-order absorption and first-order elimination. The model assumes that drug absorption from the gastrointestinal tract and systemic elimination are both proportional to the amount of drug present (linear kinetics), and that the body can be represented as a single, well-mixed compartment.

Model parameters are estimated by nonlinear least squares regression for each group (if specified). The primary parameters include the absorption rate constant (k_a), elimination rate constant (k_el), and apparent volume of distribution (Vd/F). Model-consistent secondary pharmacokinetic parameters are derived, including time to maximum concentration (Tmax), maximum concentration (Cmax), area under the concentration-time curve (AUC), and apparent clearance (CL/F), where AUC = Dose / (k_el * Vd/F) and CL/F = k_el * Vd/F.

The function includes safeguards against numerical instability when k_a and k_el are similar, enforces positive parameter bounds during fitting, and performs validity checks for the computation of Tmax and Cmax. Model performance is summarized using root mean squared error (RMSE) and information criteria (AIC and BIC), which are more appropriate than R^2 for nonlinear pharmacokinetic models.

If grouping is specified (e.g., by subject or formulation), the model is fit independently to each group. Groups with insufficient observations trigger a warning, as parameter estimates may be unreliable.

When plotting is enabled, the function generates publication-quality concentration-time plots showing observed data and the corresponding model-predicted curves based on the fitted parameters. Annotations summarizing key pharmacokinetic parameters and model diagnostics are optionally added for one group.

An optional LOESS (locally estimated scatterplot smoothing) curve may be overlaid on the observed data for exploratory visualization only. This smoother is purely descriptive, does not represent any pharmacokinetic mechanism, and should not be interpreted as part of the fitted model. For this reason, LOESS smoothing is disabled by default.

Model: C(t) = ((F * Dose * k_a) / (Vd * (k_a - k_el))) * [exp(-k_el * t) - exp(-k_a * t)]

Arguments

`data`	A data frame containing plasma concentration-time data.
`time_col`	Character string specifying the column name for time.
`conc_col`	Character string specifying the column name for plasma concentration.
`dose`	Numeric value specifying the administered oral dose.
`group_col`	Optional character string specifying a grouping variable (e.g., formulation, subject).
`plot`	Logical; if TRUE, generates a concentration-time plot with fitted curves.
`annotate`	Logical; if TRUE, annotates the plot with PK parameters (only if <= 2 groups).

Value

A list containing:

`fitted_parameters` Data frame with k_a, k_el, Tmax, Cmax, Vd/F, CL/F, and R^2.

`data` Processed data used for fitting and plotting.

Author(s)

Paul Angelo C. Manlapaz

References

Gibaldi, M. & Perrier, D. (1982) <isbn:9780824710422> Pharmacokinetics, 2nd Edition. Marcel Dekker, New York.

Gabrielsson, J. & Weiner, D. (2000) <isbn:9186274929> Pharmacokinetic/Pharmacodynamic Data Analysis: Concepts and Applications, 3rd Edition, Revised and Expanded. Swedish Pharmaceutical Press, Stockholm.

Examples

```
# Example I: Single subject oral dosing
df <- data.frame(
  time = c(0.25, 0.5, 1, 2, 4, 6, 8, 12),
  concentration = c(1.2, 2.8, 5.1, 6.4, 5.2, 4.1, 3.0, 1.8)
)
one_compartment_oral_nl(
  data = df,
  time_col = "time",
  conc_col = "concentration",
  dose = 100
)

# Example II: Condition-dependent oral pharmacokinetics (e.g., formulation or pH effect)
df_cond <- data.frame(
  time = rep(c(0.25, 0.5, 1, 2, 4, 6, 8), 2),
  concentration = c(
    1.4, 3.1, 5.6, 6.8, 5.9, 4.7, 3.6,   # Condition A
    0.9, 2.2, 4.1, 5.3, 4.8, 3.9, 3.0    # Condition B
  ),
  condition = rep(c("Condition A", "Condition B"), each = 7)
)
one_compartment_oral_nl(
  data = df_cond,
  time_col = "time",
  conc_col = "concentration",
  dose = 100,
  group_col = "condition"
)

# Example III: Multiple subjects (population-style oral pharmacokinetics)
df_subjects <- data.frame(
  time = rep(c(0.25, 0.5, 1, 2, 4, 6, 8, 12), 6),
  concentration = c(
    1.3, 3.0, 5.4, 6.7, 5.8, 4.6, 3.5, 2.3,   # Subject 1
    1.1, 2.7, 5.0, 6.3, 5.5, 4.4, 3.3, 2.2,   # Subject 2
    1.0, 2.5, 4.7, 6.0, 5.3, 4.2, 3.2, 2.1,   # Subject 3
    0.9, 2.3, 4.4, 5.7, 5.0, 4.0, 3.0, 2.0,   # Subject 4
    0.8, 2.1, 4.1, 5.4, 4.8, 3.8, 2.9, 1.9,   # Subject 5
    0.7, 2.0, 3.9, 5.2, 4.6, 3.7, 2.8, 1.8    # Subject 6
  ),
  subject = rep(paste0("S", 1:6), each = 8)
)
one_compartment_oral_nl(
  data = df_subjects,
  time_col = "time",
  conc_col = "concentration",
  dose = 100,
  group_col = "subject"
)
```

`sigmoid_emax` *Sigmoid Emax (Stimulatory Hill Model) for Dose-Response Analysis*

Description

Fits pharmacodynamic dose-response data to the stimulatory Hill (Sigmoid Emax) model using nonlinear least squares regression.

The stimulatory Hill model describes increasing pharmacological effects with increasing dose (or concentration) according to:

$$E = E_0 + \frac{E_{max} \cdot D^n}{EC_{50}^n + D^n}$$

where E_0 is the baseline effect, E_{max} is the maximum stimulatory effect, EC_{50} is the dose producing 50 E_{max}, and n is the Hill coefficient.

This model is appropriate when the observed response increases monotonically with dose.

Arguments

`data`	A data frame containing dose-response experimental data.
`dose_col`	Character string specifying the column name for dose or concentration.
`response_col`	Character string specifying the column name for measured response.
`group_col`	Optional character string specifying a column for grouping.
`log_dose`	Logical; if TRUE, dose values are log10-transformed for plotting (model fitting uses original dose values).
`plot`	Logical; if TRUE, generates a dose-response plot with fitted Hill curves.
`annotate`	Logical; if TRUE, annotates the plot with model parameters and fit metrics (only if <=1 group).

Value

A list containing:

`fitted_parameters` Data frame with E0, Emax, EC50, Hill coefficient (`n`), RMSE, AIC, and BIC values for each group.

`data` The processed dataset used for model fitting and plotting.

Author(s)

Paul Angelo C. Manlapaz

References

Hill, A. V. (1910) The possible effects of the aggregation of the molecules of hæmoglobin on its dissociation curves. The Journal of Physiology, 40(4), iv–vii.

Holford, N. H. G. & Sheiner, L. B. (1981) <doi:10.2165/00003088-198106060-00002> Understanding the dose-effect relationship. Clinical Pharmacokinetics, 6(6), 429–453.

Examples

```
# Example I: Single dose-response dataset
df <- data.frame(
  dose = c(0.1, 0.3, 1, 3, 10, 30, 100),
  response = c(5, 10, 25, 55, 80, 92, 98)
)
sigmoid_emax(
  data = df,
  dose_col = "dose",
  response_col = "response"
)

# Example II: Two treatment groups
df2 <- data.frame(
  dose = rep(c(0.1, 0.3, 1, 3, 10, 30), 2),
  response = c(
    3, 8, 20, 45, 70, 85,     # Group A
    2, 6, 15, 35, 60, 78      # Group B
  ),
  treatment = rep(c("Group A", "Group B"), each = 6)
)
sigmoid_emax(
  data = df2,
  dose_col = "dose",
  response_col = "response",
  group_col = "treatment",
  log_dose = TRUE
)

# Example III: Multiple subjects (population-style dose-response pharmacodynamics)
df_subjects <- data.frame(
  dose = rep(c(0.1, 0.3, 1, 3, 10, 30), 5),
  response = c(
    5, 13, 30, 56, 80, 92,   # Subject 1
    4, 12, 28, 54, 78, 90,   # Subject 2
    6, 15, 33, 59, 83, 95,   # Subject 3
    5, 14, 31, 57, 81, 93,   # Subject 4
    3, 11, 26, 52, 76, 88    # Subject 5
  ),
  subject = rep(paste0("S", 1:5), each = 6)
)
sigmoid_emax(
  data = df_subjects,
  dose_col = "dose",
  response_col = "response",
  group_col = "subject",
  log_dose = TRUE
)
```

`sigmoid_imax` *Sigmoid Imax (Inhibitory Hill Model) for Dose-Response Analysis*

Description

Fits pharmacodynamic dose-response data to the inhibitory Hill (Sigmoid Imax) model using non-linear least squares regression.

The inhibitory Hill model describes decreasing pharmacological effects with increasing dose (or concentration) according to:

$$E = E_0 - \frac{I_{max} \cdot D^n}{IC_{50}^n + D^n}$$

An equivalent parameterization is:

$$E = E_{min} + \frac{(E_0 - E_{min}) \cdot IC_{50}^n}{IC_{50}^n + D^n}$$

where E_0 is the baseline response, I_{max} is the maximum inhibitory effect, IC_{50} is the dose producing 50 inhibition, and n is the Hill coefficient.

This model is appropriate when the observed response decreases monotonically with increasing dose.

Arguments

`data`	A data frame containing dose-response experimental data.
`dose_col`	Character string specifying the column name for dose or concentration.
`response_col`	Character string specifying the column name for measured response.
`group_col`	Optional character string specifying a column for grouping.
`log_dose`	Logical; if TRUE, dose values are log10-transformed for plotting (model fitting uses original dose values).
`plot`	Logical; if TRUE, generates a dose-response plot with fitted Hill curves.
`annotate`	Logical; if TRUE, annotates the plot with model parameters and fit metrics (only if <=1 group).

Value

A list containing:

`fitted_parameters` Data frame with E0, Imax, IC50, Hill coefficient (`n`), RMSE, AIC, and BIC values for each group.

`data` The processed dataset used for model fitting and plotting.

Author(s)

Paul Angelo C. Manlapaz

References

Holford, N. H. G. & Sheiner, L. B. (1981) <doi:10.2165/00003088-198106060-00002> Understanding the dose-effect relationship. Clinical Pharmacokinetics, 6(6), 429–453.

Examples

```
# Example I: Single inhibitory dose-response dataset
df <- data.frame(
  dose = c(0.1, 0.3, 1, 3, 10, 30, 100),
  response = c(95, 90, 75, 45, 20, 8, 3)
)
sigmoid_imax(
  data = df,
  dose_col = "dose",
  response_col = "response"
)

# Example II: Two treatment groups (e.g., two inhibitors or conditions)
df2 <- data.frame(
  dose = rep(c(0.1, 0.3, 1, 3, 10, 30), 2),
  response = c(
    92, 85, 65, 40, 18, 7,    # Group A (stronger inhibitor)
    95, 88, 72, 50, 30, 15    # Group B (weaker inhibitor)
  ),
  treatment = rep(c("Group A", "Group B"), each = 6)
)
sigmoid_imax(
  data = df2,
  dose_col = "dose",
  response_col = "response",
  group_col = "treatment",
  log_dose = TRUE
)

# Example III: Multiple subjects (population-style inhibitory pharmacodynamics)
df_subjects <- data.frame(
  dose = rep(c(0.1, 0.3, 1, 3, 10, 30), 5),
  response = c(
    94, 86, 68, 42, 20, 9,    # Subject 1
    96, 88, 70, 45, 22, 10,   # Subject 2
    93, 84, 65, 40, 18, 8,    # Subject 3
    95, 87, 69, 44, 21, 9,    # Subject 4
    97, 89, 72, 48, 25, 12    # Subject 5
  ),
  subject = rep(paste0("S", 1:5), each = 6)
)
sigmoid_imax(
  data = df_subjects,
  dose_col = "dose",
  response_col = "response",
  group_col = "subject",
  log_dose = TRUE
)
```

`td50_model` *Toxic Dose 50 (TD50) Pharmacodynamic Model*

Description

Fits quantal toxicity response data to a logistic dose-response model to estimate the Toxic Dose 50 (TD50), defined as the dose producing toxicity in 50

The model uses binomial logistic regression and supports optional grouping (e.g., sex, species, formulation) and stratification by experimental conditions (e.g., exposure route).

In addition to TD50 estimation, the model provides the following interpretable parameters:

- **Slope**: Represents the steepness of the dose-response curve. A larger slope indicates a rapid increase in toxicity with small increases in dose (narrow tolerance or high population sensitivity), whereas a smaller slope reflects a more gradual response, suggesting greater inter-individual variability in susceptibility.
- **Intercept**: Represents the baseline log-odds of observing toxicity at zero dose. A strongly negative intercept indicates minimal background toxicity, while a positive intercept suggests appreciable toxicity in the absence of administered dose, which may indicate experimental bias or background risk.
- **TD50 95% Confidence Interval**: An approximate 95 interval for the TD50, computed using the delta method. This provides an uncertainty range around the estimated dose causing 50
- **McFadden Pseudo-R^2**: A likelihood-based measure of model goodness-of-fit that quantifies how much better the fitted model explains the data compared to a null (intercept-only) model. Values between 0.1 and 0.2 indicate acceptable biological fit, while values above 0.3 suggest a strong and reliable dose-response relationship.

The function can generate dose-response plots with fitted curves and annotate TD50, slope, intercept, TD50 confidence intervals, and McFadden pseudo-R^2.

Arguments

`data`	A data frame containing toxicity response data.
`dose_col`	Character string specifying the dose column.
`response_col`	Character string specifying the binary toxicity response (0 = no toxicity, 1 = toxic response).
`group_col`	Optional character string specifying a grouping variable.
`condition_col`	Optional character string specifying an experimental condition.
`plot`	Logical; if TRUE, generates dose-response plots.
`annotate`	Logical; if TRUE, annotates the plot with TD50, confidence intervals, and model parameters (only if <=2 groups).

Value

A list containing:

`fitted_parameters` Data frame with TD50, 95 slope, intercept, and pseudo-R2 values for each group.

`data` The processed data used for model fitting and plotting.

Author(s)

Paul Angelo C. Manlapaz

References

Bliss, C. I. (1935) <doi:10.1111/j.1744-7348.1935.tb07713.x> The calculation of the dosage-mortality curve. Annals of Applied Biology, 22(1), 134–167.

Finney, D. J. (1971) <isbn:9780521080415> Probit Analysis, 3rd Edition. Cambridge University Press, Cambridge.

Examples

```
# Example I: Single population toxicity study
df1 <- data.frame(
  dose = c(5, 10, 20, 40, 80, 160),
  toxic = c(0, 0, 0, 1, 1, 1)
)
td50_model(
  data = df1,
  dose_col = "dose",
  response_col = "toxic"
)

# Example II: Grouped analysis (Male vs Female)
df2 <- data.frame(
  dose = rep(c(5, 10, 20, 40, 80), 2),
  toxic = c(0,0,1,1,1, 0,0,0,1,1),
  sex = rep(c("Male","Female"), each = 5)
)
td50_model(
  data = df2,
  dose_col = "dose",
  response_col = "toxic",
  group_col = "sex"
)

# Example III: Grouped by formulation and exposure route
df3 <- data.frame(
  dose = rep(c(10, 25, 50, 100), 4),
  toxic = c(0,0,1,1, 0,1,1,1, 0,0,0,1, 0,0,1,1),
  formulation = rep(c("A","B"), each = 8),
  route = rep(c("Oral","IV"), each = 4, times = 2)
)
td50_model(
  data = df3,
  dose_col = "dose",
  response_col = "toxic",
  group_col = "formulation",
  condition_col = "route"
)
```

`two_compartment_iv_bolus` *Two-Compartment IV Bolus Pharmacokinetic Model (Linear)*

Description

Fits plasma concentration-time data following an intravenous (IV) bolus dose using a **log-linear approximation of a two-compartment pharmacokinetic model**.

This function applies linear regression to the logarithm of plasma concentrations versus time to estimate the **terminal elimination phase** parameters. The approach provides an empirical approximation to a biexponential decline but does **not explicitly decompose** the curve into distribution and elimination exponentials.

Under the two-compartment IV bolus model, concentration-time profiles are classically described by:

$$C(t) = Ae^{-\alpha t} + Be^{-\beta t}$$

In this implementation, the terminal log-linear phase is approximated as:

$$\log(C) = \log(C_0) - k_{el}\, t$$

where C_0 is the extrapolated initial concentration and k_{el} is the apparent elimination rate constant.

From the fitted log-linear model, secondary pharmacokinetic parameters are derived, including terminal half-life, apparent volume of distribution, and clearance.

Arguments

`data`	A data frame with plasma concentration-time data.
`time_col`	Character string for the time column.
`conc_col`	Character string for plasma concentration column.
`dose`	Numeric value specifying IV bolus dose.
`group_col`	Optional character string specifying grouping variable.
`plot`	Logical; if TRUE, generates concentration-time plots.
`annotate`	Logical; if TRUE, annotates plot (only if <=2 groups).

Value

A list containing:

`fitted_parameters` Data frame with C0, kel, t_half, Vd, CL, and R^2 for each group.

`data` Processed data used for fitting and plotting.

Author(s)

Paul Angelo C. Manlapaz

References

Gibaldi, M. & Perrier, D. (1982) <isbn:9780824710422> Pharmacokinetics, 2nd Edition. Marcel Dekker, New York.

Gabrielsson, J. & Weiner, D. (2000) <isbn:9186274929> Pharmacokinetic/Pharmacodynamic Data Analysis: Concepts and Applications, 3rd Edition, Revised and Expanded. Swedish Pharmaceutical Press, Stockholm.

Examples

```
# Example I: Single subject two-compartment IV bolus data
df <- data.frame(
  time = c(0.08, 0.25, 0.5, 1, 2, 4, 6, 8, 12),
  concentration = c(40.0, 30.5, 25.0, 17.5, 10.2, 6.4, 4.1, 2.8, 1.5)
)
two_compartment_iv_bolus(
  data = df,
  time_col = "time",
  conc_col = "concentration",
  dose = 100
)

# Example II: Condition-dependent pharmacokinetics (e.g., physiological state)
df_cond <- data.frame(
  time = rep(c(0.25, 0.5, 1, 2, 4, 6, 8), 2),
  concentration = c(
    25.3, 22.1, 18.5, 13.2, 8.5, 5.6, 3.8,   # Condition A
    20.7, 18.0, 14.9, 11.3, 7.1, 4.7, 3.2    # Condition B
  ),
  condition = rep(c("Condition A", "Condition B"), each = 7)
)
two_compartment_iv_bolus(
  data = df_cond,
  time_col = "time",
  conc_col = "concentration",
  dose = 100,
  group_col = "condition"
)

# Example III: Multiple subjects (population-style two-compartment IV bolus pharmacokinetics)
df_subjects <- data.frame(
  time = rep(c(0.25, 0.5, 1, 2, 4, 6, 8), 5),
  concentration = c(
    26.1, 23.2, 19.6, 14.0, 9.0, 6.0, 4.0,   # Subject 1
    24.8, 21.8, 18.4, 13.3, 8.8, 5.8, 3.9,   # Subject 2
    25.5, 22.5, 19.0, 13.8, 8.7, 5.7, 3.7,   # Subject 3
    23.9, 20.9, 17.7, 12.8, 8.4, 5.5, 3.5,   # Subject 4
    24.4, 21.5, 18.0, 13.0, 8.5, 5.6, 3.6    # Subject 5
  ),
  subject = rep(paste0("S", 1:5), each = 7)
)
two_compartment_iv_bolus(
```

```
  data = df_subjects,
  time_col = "time",
  conc_col = "concentration",
  dose = 100,
  group_col = "subject"
)
```

`two_compartment_iv_bolus_nl` *Two-Compartment IV Bolus Pharmacokinetic Model (Nonlinear)*

Description

Fits plasma concentration-time data to a two-compartment intravenous (IV) bolus pharmacokinetic model. The model assumes instantaneous drug administration and distribution between central and peripheral compartments, with first-order elimination.

Model: C(t) = A * exp(-alpha * t) + B * exp(-beta * t)

where:

- A, B: intercept coefficients
- alpha: distribution rate constant (alpha > beta)
- beta: elimination rate constant

Arguments

`data`	A data frame containing plasma concentration-time data.
`time_col`	Character string specifying the column name for time.
`conc_col`	Character string specifying the column name for plasma concentration.
`dose`	Numeric value specifying the administered IV bolus dose.
`group_col`	Optional character string specifying a grouping variable (e.g., formulation, subject).
`plot`	Logical; if TRUE, generates a concentration-time plot with fitted curves.
`annotate`	Logical; if TRUE, annotates the plot with PK parameters (only if <=2 groups).

Value

A list containing:

`fitted_parameters` Data frame with biexponential coefficients (A and B), alpha, beta, t_half_alpha, t_half_beta, and R^2 for each group.

`data` Processed data used for fitting and plotting.

Author(s)

Paul Angelo C. Manlapaz

References

Gibaldi, M. & Perrier, D. (1982) <isbn:9780824710422> Pharmacokinetics, 2nd Edition. Marcel Dekker, New York.

Gabrielsson, J. & Weiner, D. (2000) <isbn:9186274929> Pharmacokinetic/Pharmacodynamic Data Analysis: Concepts and Applications, 3rd Edition, Revised and Expanded. Swedish Pharmaceutical Press, Stockholm.

Examples

```
# Example I: Single subject two-compartment IV bolus data
df <- data.frame(
  time = c(0.08, 0.25, 0.5, 1, 2, 4, 6, 8, 12),
  concentration = c(40.0, 30.5, 25.0, 17.5, 10.2, 6.4, 4.1, 2.8, 1.5)
)
two_compartment_iv_bolus_nl(
  data = df,
  time_col = "time",
  conc_col = "concentration",
  dose = 100
)

# Example II: Condition-dependent pharmacokinetics (e.g., physiological state)
df_cond <- data.frame(
  time = rep(c(0.25, 0.5, 1, 2, 4, 6, 8), 2),
  concentration = c(
    25.3, 22.1, 18.5, 13.2, 8.5, 5.6, 3.8,   # Condition A
    20.7, 18.0, 14.9, 11.3, 7.1, 4.7, 3.2    # Condition B
  ),
  condition = rep(c("Condition A", "Condition B"), each = 7)
)
two_compartment_iv_bolus_nl(
  data = df_cond,
  time_col = "time",
  conc_col = "concentration",
  dose = 100,
  group_col = "condition"
)

# Example III: Multiple subjects (population-style two-compartment IV bolus pharmacokinetics)
df_subjects <- data.frame(
  time = rep(c(0.25, 0.5, 1, 2, 4, 6, 8), 5),
  concentration = c(
    26.1, 23.2, 19.6, 14.0, 9.0, 6.0, 4.0,   # Subject 1
    24.8, 21.8, 18.4, 13.3, 8.8, 5.8, 3.9,   # Subject 2
    25.5, 22.5, 19.0, 13.8, 8.7, 5.7, 3.7,   # Subject 3
    23.9, 20.9, 17.7, 12.8, 8.4, 5.5, 3.5,   # Subject 4
    24.4, 21.5, 18.0, 13.0, 8.5, 5.6, 3.6    # Subject 5
  ),
  subject = rep(paste0("S", 1:5), each = 7)
)
two_compartment_iv_bolus_nl(
```

```
  data = df_subjects,
  time_col = "time",
  conc_col = "concentration",
  dose = 100,
  group_col = "subject"
)
```

`weibull_model` *Weibull Drug Release Kinetic Model*

Description

Fits experimental cumulative drug release data to the Weibull model using a linearized regression of log(-log(1 - Mt/MInf)) versus log(time).The function automatically normalizes cumulative percent drug release to fraction (0-1) by default and removes t = 0. In addition, the function supports optional grouping variables (e.g., formulation, batch) and optional pH-dependent analysis. It generates publication-quality plots with experimental curves, fitted Weibull straight lines, a shape-parameter interpretation table, and annotations for the scale parameter (alpha), shape parameter (beta), coefficient of determination (R^2), and the time required for 50-percent drug release (t50).

Users can toggle 'normalize = TRUE/FALSE' to use fraction (0-1) or raw percent drug release. Normalization is recommended (fraction release) because the model assumes 0 <= Mt/MInf <= 1.

Model: log(-log(1 - Mt/MInf)) = beta * log(t) - beta * log(alpha)

The shape parameter beta indicates release kinetics: - beta = 1 : Exponential (first-order release) - beta < 1 : Parabolic (decelerating release) - beta > 1 : Sigmoidal (accelerating then decelerating)

Arguments

`data`	A data frame containing experimental cumulative percent drug release data.
`time_col`	Character string specifying the column name for time (minutes).
`release_col`	Character string specifying the column name for cumulative percent drug release or fraction released.
`group_col`	Optional character string specifying a grouping variable (e.g., formulation, batch).
`pH_col`	Optional character string specifying a column containing pH values.
`plot`	Logical; if TRUE, generates a plot with fitted Weibull release curves.
`annotate`	Logical; if TRUE, annotates the plot with alpha, beta, R^2, and t50 (only if <= 2 groups).
`normalize`	Logical; if TRUE (default), normalizes cumulative percent drug release to fraction released (0-1). If FALSE, assumes the input release data are already expressed as fraction released.

Value

A list containing:

`fitted_parameters` Data frame with Weibull parameters (`alpha`, `beta`), R^2, and t50 for each group.

`data` Processed data used for model fitting and plotting.

Author(s)

Paul Angelo C. Manlapaz

References

Weibull, W. (1951) <doi:10.1115/1.4010337> A statistical distribution function of wide applicability. Journal of Applied Mechanics, 18(3), 293–297.

Examples

```
# Example I: Single formulation
df1 <- data.frame(
  time = c(0, 15, 30, 45, 60, 90, 120, 150, 180),
  release = c(0, 11.4, 20.8, 30.8, 39.8, 57.8, 72, 84.8, 93.5)
)
weibull_model(
  data = df1,
  time_col = "time",
  release_col = "release",
  normalize = TRUE
)

# Example II: Two formulations (grouped, not pH-dependent)
df2 <- data.frame(
  time = rep(c(0, 15, 30, 45, 60, 75, 90, 105, 120, 150), 2),
  release = c(
    0, 10, 22, 35, 48, 60, 72, 80, 86, 92,   # Formulation A
    0, 8, 18, 28, 38, 48, 58, 64, 68, 72     # Formulation B
  ),
  formulation = rep(c("Formulation A", "Formulation B"), each = 10)
)
weibull_model(
  data = df2,
  time_col = "time",
  release_col = "release",
  group_col = "formulation"
)

# Example III: pH-dependent release
df_pH <- data.frame(
  time = rep(c(0, 27, 60, 88, 95, 120, 138, 155, 175, 180), 2),
  release = c(
    0, 12, 25, 38, 52, 63, 72, 80, 88, 95,   # pH 7.4
    0, 10, 20, 30, 42, 53, 63, 70, 77, 85    # pH 4.5
  ),
  pH = rep(c(7.4, 4.5), each = 10)
)
weibull_model(
  data = df_pH,
  time_col = "time",
  release_col = "release",
  pH_col = "pH"
```

```
)

# Example IV: Two formulations under two pH conditions
df1 <- data.frame(
  time = rep(c(0, 20, 40, 60, 80, 100, 120, 140, 160, 180), 2),
  release = c(
    0, 12, 25, 38, 50, 62, 73, 82, 89, 95,   # pH 4.5
    0, 15, 30, 45, 59, 70, 79, 86, 91, 97    # pH 7.6
  ),
  pH = rep(c(4.5, 7.6), each = 10)
)
df2 <- data.frame(
  time = rep(c(0, 15, 30, 45, 60, 75, 90, 105, 120, 135), 2),
  release = c(
    0, 10, 22, 34, 46, 57, 67, 76, 84, 91,   # pH 4.5
    0, 13, 27, 41, 55, 67, 77, 85, 92, 98    # pH 7.6
  ),
  pH = rep(c(4.5, 7.6), each = 10)
)
df_all <- rbind(
  cbind(formulation = "Dataset 1", df1),
  cbind(formulation = "Dataset 2", df2)
)
weibull_model(
  data = df_all,
  time_col = "time",
  release_col = "release",
  group_col = "formulation",
  pH_col = "pH"
)
```

`zero_order_release` *Zero-Order Drug Release Kinetic Model*

Description

Fits experimental cumulative drug release data to a zero-order kinetic model using linear regression. The function supports optional grouping variables (e.g., formulation, batch) and optional pH-dependent analysis. It can generate publication-quality plots with fitted straight lines and annotations for zero-order rate constant (k0), intercept, coefficient of determination (R^2), and time required for 50-percent drug release (t50).

Arguments

`data`	A data frame containing experimental drug release data.
`time_col`	Character string specifying the column name for time (e.g., minutes or hours).
`release_col`	Character string specifying the column name for cumulative drug release (typically percentage).

group_col	Optional character string specifying a column name used for grouping (e.g., formulation, batch). Default is NULL.
pH_col	Optional character string specifying a column name containing pH values. If provided, zero-order models are fitted separately for each pH.
plot	Logical; if TRUE, generates a plot of experimental data with zero-order fitted straight lines (default is TRUE).
annotate	Logical; if TRUE, annotates the plot with k0, intercept, R^2, and t50 values for each group (default is TRUE).

Value

A list containing:

fitted_parameters A data frame with k0, intercept, R^2, and t50 values for each group or pH condition.

data The processed data used for model fitting and plotting.

Author(s)

Paul Angelo C. Manlapaz

References

Higuchi, T. (1961) <doi:10.1002/jps.2600501018> Rate of release of medicaments from ointment bases containing drugs in suspension. Journal of Pharmaceutical Sciences, 50(10), 874–875.

Examples

```
# Example I: Single formulation
df_1 <- data.frame(
  time = c(0, 15, 30, 45, 60, 90, 120, 150, 180),
  release = c(0, 11.4, 20.8, 30.8, 39.8, 57.8, 72, 84.8, 93.5)
)
zero_order_release(
  data = df_1,
  time_col = "time",
  release_col = "release"
)

# Example II: Two formulations (grouped, not pH-dependent)
df_2 <- data.frame(
  time = rep(c(0, 30, 60, 90, 120, 150), 2),
  release = c(
    0, 18, 35, 55, 72, 88,   # Formulation A
    0, 12, 26, 40, 58, 70    # Formulation B
  ),
  formulation = rep(c("Formulation A", "Formulation B"), each = 6)
)
zero_order_release(
  data = df_2,
```

```
  time_col = "time",
  release_col = "release",
  group_col = "formulation"
)

# Example III: pH-dependent release
df_pH <- data.frame(
  time = rep(c(0, 60, 120, 180), 2),
  release = c(0, 40, 75, 95, 0, 30, 60, 80),
  pH = rep(c(7.4, 4.5), each = 4)
)
zero_order_release(
  data = df_pH,
  time_col = "time",
  release_col = "release",
  pH_col = "pH"
)

# Example IV: Two formulations under two pH conditions
df1 <- data.frame(
  time = rep(c(0, 30, 60, 90, 120, 150, 180), 2),
  release = c(
    0, 12, 25, 38, 52, 65, 78,   # pH 4.5
    0, 15, 30, 47, 63, 78, 90    # pH 7.6
  ),
  pH = rep(c(4.5, 7.6), each = 7)
)
df2 <- data.frame(
  time = rep(c(0, 20, 40, 60, 80, 100, 120), 2),
  release = c(
    0, 10, 22, 35, 50, 64, 77,   # pH 4.5
    0, 14, 28, 45, 61, 76, 88    # pH 7.6
  ),
  pH = rep(c(4.5, 7.6), each = 7)
)
df_all <- rbind(
  cbind(dataset = "Dataset 1", df1),
  cbind(dataset = "Dataset 2", df2)
)
zero_order_release(
  data = df_all,
  time_col = "time",
  release_col = "release",
  group_col = "dataset",
  pH_col = "pH"
)
```

Index

first_order_release, 2

higuchi_release, 5
hixson_crowell_model, 7

korsmeyer_peppas_model, 10

ld50_model, 12
logistic_4pl, 14
logistic_5pl, 16

michaelis_menten, 18
michaelis_menten_nl, 22

non_compartmental, 24
non_compartmental_nl, 26

one_compartment_iv_bolus, 28
one_compartment_iv_bolus_nl, 31
one_compartment_oral, 33
one_compartment_oral_nl, 35

sigmoid_emax, 38
sigmoid_imax, 39

td50_model, 41
two_compartment_iv_bolus, 44
two_compartment_iv_bolus_nl, 46

weibull_model, 48

zero_order_release, 50