Package ‘gammaFuncModel’

June 13, 2025

Type Package

Title Non-Linear Mixed Effects Model Based on the Gamma Function Form

Version 4.0

Date 2025-06-11

Description Identifies biomarkers that exhibit differential response dynamics by time across groups and estimates kinetic properties of biomarkers.

License GPL

RoxygenNote 7.2.3

Imports Rdpack, nlme, dplyr, cubature, rootSolve, patchwork, scales, ggplot2, future.apply, gridExtra, rlang, grDevices, stats, grid

NeedsCompilation no

Author Hongting Chen [aut, cre],
Liming Liang [aut]

Maintainer Hongting Chen <`hongtingchen@berkeley.edu`>

Repository CRAN

Date/Publication 2025-06-13 19:50:02 UTC

Contents

`calculate_AUC` *Function that produces Area Under the Curve(AUC) property for a single individual in a particular group, for a specific metabolite*

Description

Function that produces Area Under the Curve(AUC) property for a single individual in a particular group, for a specific metabolite

Usage

```
calculate_AUC(f, upperbound)
```

Arguments

`f`	function that returns the prediction of a metabolite concentration, for a single individual in a particular group
`upperbound`	Numeric value that serves as the upperbound of integration

Value

AUC for this metabolite, in a particular group for a single individual

References

Wickham, H. (2022). dplyr: A Grammar of Data Manipulation. R package version 1.0.10. Available at: https://CRAN.R-project.org/package=dplyr

Pinheiro, J. C., & Bates, D. M. (2022). nlme: Linear and Nonlinear Mixed Effects Models. R package version 3.1-153. Available at: https://CRAN.R-project.org/package=nlme

Examples

```
require(gammaFuncModel)
require(cubature)
require(dplyr)
require(nlme)
modify.df <- data.frame(
  ID = rep(sprintf("%02d", 1:10), each = 9 * 3),
  Time = rep(rep(1:9, each = 3), 10),
  Diet = as.factor(rep(1:3, times = 9 * 10)),
  Age = rep(sample(20:70, 10, replace = TRUE), each = 9 * 3),
  BMI = round(rep(runif(10, 18.5, 35), each = 9 * 3), 1),
  Concentration = NA
)
for (i in 1:10) {
 for (d in 1:3) {
   C0 <- runif(1, 10, 15)    # initial concentration
```

```
   k <- runif(1, 0.1, 0.3)   # decay rate constant
   modify.df$Concentration[modify.df$ID == sprintf("%02d", i) & modify.df$Diet == d] <-
    C0 * exp(-k * modify.df$Time[modify.df$ID == sprintf("%02d", i) & modify.df$Diet == d])
 }
}
covariates <- c("ID", "Diet", "Age", "BMI")
model <- gammaFunction(
  modify.df,
  covariates,
  time_grp_inter = FALSE,
return_ml_model = FALSE, include_grp = TRUE
)[[1]]
test_data <- modify.df %>%
  filter(Diet == 1 & ID == "04") %>%
  select(-c("Concentration", "ID", "Diet"))
f_dat = modify.df %>%
  filter(Diet == 1 & ID == "04") %>%
  select(-Concentration)
f <- generate_f_function(
  data = f_dat,
  model = model,
  grp_var = 1,
  grp_name = "Diet",
  ID = "04",
  ref = 1
)
AUC <- calculate_AUC(f, 9)
AUCInf <- calculate_AUC(f, Inf)
```

`calculate_Cmax` *Function that produces Cmax property for a single individual in a particular group, for a specific metabolite*

Description

Function that produces Cmax property for a single individual in a particular group, for a specific metabolite

Usage

```
calculate_Cmax(data, model, grp_var, ID, grp_name = "Diet", Tmax)
```

Arguments

`data`	Data frame containing columns Group(factor); ID(subject ID: character); Time(positive: numeric); other individiual characteristics covariates (exlcluding other forms of 'Time')
`model`	Fitted model for the metabolite in question

`grp_var`	Value of the grouping variable
`ID`	Subject ID
`grp_name`	Name of the grouping variable. Default is 'Diet'
`Tmax`	for this metabolite, in a particular group for a single individual

Value

Cmax for this metabolite, in a particular group for a single individual

References

Wickham, H. (2022). dplyr: A Grammar of Data Manipulation. R package version 1.0.10. Available at: https://CRAN.R-project.org/package=dplyr

Pinheiro, J. C., & Bates, D. M. (2022). nlme: Linear and Nonlinear Mixed Effects Models. R package version 3.1-153. Available at: https://CRAN.R-project.org/package=nlme

Examples

```
require(gammaFuncModel)
require(rootSolve)
require(dplyr)
require(nlme)
df <- data.frame(
  ID = rep(sprintf("%02d", 1:10), each = 9 * 3),
  Time = rep(rep(1:9, each = 3), 10),
  Diet = as.factor(rep(1:3, times = 9 * 10)),
  Age = rep(sample(20:70, 10, replace = TRUE), each = 9 * 3),
  BMI = round(rep(runif(10, 18.5, 35), each = 9 * 3), 1),
  Concentration = round(runif(270, 5, 15), 2)
)
covariates <- c("ID", "Diet", "Age", "BMI")
model <- gammaFunction(
  df,
  covariates,
  time_grp_inter = FALSE,
  return_ml_model = FALSE,
  include_grp = TRUE
  )[[1]]
test_data = df %>% filter(Diet == 1 & ID == "02") %>% select(-c("Concentration", "ID", "Diet"))
Tmax <- calculate_Tmax(data = test_data, model, grp_var = 1, ID = "02", grp_name = 'Diet', ref = 1)
Cmax <- calculate_Cmax(data = test_data, model, grp_var = 1, ID = "02", grp_name = "Diet", Tmax)
```

`calculate_half_life` *Function that produces Half-life property for a single individual in a particular group, for a specific metabolite*

Description

Function that produces Half-life property for a single individual in a particular group, for a specific metabolite

Usage

```
calculate_half_life(f, Tmax, Cmax)
```

Arguments

`f`	function that returns the prediction of a metabolite concentration, for a single individual in a particular group
`Tmax`	Tmax property of a metabolite, for a single individual in a particular group
`Cmax`	Cmax property of a metabolite, for a single individual in a particular group

Value

Half-life for this metabolite, in a particular group for a single individual

References

Wickham, H. (2022). dplyr: A Grammar of Data Manipulation. R package version 1.0.10. Available at: `https://CRAN.R-project.org/package=dplyr`

Pinheiro, J. C., & Bates, D. M. (2022). nlme: Linear and Nonlinear Mixed Effects Models. R package version 3.1-153. Available at: `https://CRAN.R-project.org/package=nlme`

Examples

```
require(gammaFuncModel)
require(rootSolve)
require(dplyr)
require(nlme)

modify.df <- data.frame(
  ID = rep(sprintf("%02d", 1:10), each = 9 * 3),
  Time = rep(rep(1:9, each = 3), 10),
  Diet = as.factor(rep(1:3, times = 9 * 10)),
  Age = rep(sample(20:70, 10, replace = TRUE), each = 9 * 3),
  BMI = round(rep(runif(10, 18.5, 35), each = 9 * 3), 1),
  Concentration = NA
)
for (i in 1:10) {
 for (d in 1:3) {
   C0 <- runif(1, 10, 15)    # initial concentration
   k <- runif(1, 0.1, 0.3)   # decay rate constant
   modify.df$Concentration[modify.df$ID == sprintf("%02d", i) & modify.df$Diet == d] <-
    C0 * exp(-k * modify.df$Time[modify.df$ID == sprintf("%02d", i) & modify.df$Diet == d])
 }
}
covariates <- c("ID", "Diet", "Age", "BMI")
```

```
model <- gammaFunction(
  modify.df,
  covariates,
  time_grp_inter = FALSE,
  return_ml_model = FALSE,
  include_grp = TRUE
  )[[1]]
test_data = modify.df %>%
  filter(Diet == 1 & ID == "03") %>%
  select(-c("Concentration", "ID", "Diet"))
Tmax <- calculate_Tmax(data = test_data, model, grp_var = 1, ID = "03", grp_name = 'Diet', ref = 1)
Cmax <- calculate_Cmax(data = test_data, model, grp_var = 1, ID = "03", grp_name = "Diet", Tmax)
f_dat = modify.df %>% filter(Diet == 1 & ID == "03") %>% select(-Concentration)
f <- generate_f_function(
  data = f_dat,
  model = model,
  grp_var = 1,
  grp_name = "Diet",
  ID = "03",
  ref = 1)
half_life <- calculate_half_life(f, Tmax, Cmax)
```

`calculate_Tmax` *Function that produces Tmax property for a single individual in a particular group, for a specific metabolite*

Description

Function that produces Tmax property for a single individual in a particular group, for a specific metabolite

Usage

```
calculate_Tmax(data, model, grp_var, ID, grp_name = "Diet", ref = 1)
```

Arguments

`data`	Data frame containing columns Group(factor); ID(subject ID: character); Time(positive: numeric); other individiual characteristics covariates(excluding other forms of 'Time')
`model`	Fitted model for the metabolite in question
`grp_var`	Value of the grouping variable
`ID`	Subject ID
`grp_name`	Name of the grouping variable. Default is 'Diet'
`ref`	numeric or character. The reference level for the grouping variable, as a factor

Value

Tmax for this metabolite, in a particular group for a single individual

References

Wickham, H. (2022). dplyr: A Grammar of Data Manipulation. R package version 1.0.10. Available at: https://CRAN.R-project.org/package=dplyr

Pinheiro, J. C., & Bates, D. M. (2022). nlme: Linear and Nonlinear Mixed Effects Models. R package version 3.1-153. Available at: https://CRAN.R-project.org/package=nlme

Examples

```
require(gammaFuncModel)
require(rootSolve)
require(dplyr)
require(nlme)
df <- data.frame(
  ID = rep(sprintf("%02d", 1:10), each = 9 * 3),
  Time = rep(rep(1:9, each = 3), 10),
  Diet = as.factor(rep(1:3, times = 9 * 10)),
  Age = rep(sample(20:70, 10, replace = TRUE), each = 9 * 3),
  BMI = round(rep(runif(10, 18.5, 35), each = 9 * 3), 1),
  Concentration = round(runif(270, 5, 15), 2)
)
covariates <- c("ID", "Diet", "Age", "BMI")
model <- gammaFunction(
  df,
  covariates,
  time_grp_inter = FALSE,
  return_ml_model = FALSE,
  include_grp = TRUE
)[[1]]
test_data = df %>% filter(Diet == 1 & ID == "01") %>% select(-c("Concentration", "ID", "Diet"))
Tmax <- calculate_Tmax(data = test_data, model, grp_var = 1, ID = "01", grp_name = 'Diet', ref = 1)
```

`diffGrpResponse` *Function that produces a summary table for coefficient estimates, their p-values and LRT p-values for every metabolite in the dataframe*

Description

Function that produces a summary table for coefficient estimates, their p-values and LRT p-values for every metabolite in the dataframe

Usage

```
diffGrpResponse(
  df,
  met_vec,
  covariates,
  time_terms = c("Time", "log(Time)"),
  grp = "Diet",
  random_formula = ~1 + Time | ID/Diet,
  correlation_formula = corSymm(form = ~Time | ID/Diet),
  weights = varIdent(form = ~1 | Time)
)
```

Arguments

`df`	Data frame containing columns Group(numeric or character); ID(subject ID: character); Time(positive: numeric); other Time terms (numeric); other individual characteristics covariates; as well columns of metabolite concentrations
`met_vec`	Vector of metabolite names
`covariates`	Vector containing the names of the "ID" covariate, grouping covariate and other covariates excluding any "Time" covariates
`time_terms`	Vector that contains all additional form of the covariate 'Time" (including the 'Time' covariate), and must contain 'log(Time)', other forms also include I(Time^2) and I(Time^3);
`grp`	Grouping variable;
`random_formula`	Random effects formula for the model, nested effects of Diet within ID (could also add random slope for 'Time');
`correlation_formula`	Correlation formula. Default is autorgressive but can accommodate other forms such as unstructured covariance or exponential covariance;
`weights`	specify a variance function that models heteroscedasticity

Value

Data frame that contains the coefficient estimates, their corresponding p-values; LRT p-values for Time-Group interactions (for every 'Time' term);LRT p-values for Group and Time-Group interactions (for every 'Time' term); as well as the fitted models for each metabolite

References

Wickham, H. (2022). dplyr: A Grammar of Data Manipulation. R package version 1.0.10. Available at: https://CRAN.R-project.org/package=dplyr

Pinheiro, J. C., & Bates, D. M. (2022). nlme: Linear and Nonlinear Mixed Effects Models. R package version 3.1-153. Available at: https://CRAN.R-project.org/package=nlme

Examples

```
require(gammaFuncModel)
require(dplyr)
require(nlme)
df <- data.frame(
  ID = rep(sprintf("%02d", 1:10), each = 9 * 3),
  Time = rep(rep(1:9, each = 3), 10),
  Diet = as.factor(rep(1:3, times = 9 * 10)),
  Age = rep(sample(20:70, 10, replace = TRUE), each = 9 * 3),
  BMI = round(rep(runif(10, 18.5, 35), each = 9 * 3), 1)
)
metvar <- paste0("met", 1:10)
concentration_data <- replicate(10, round(runif(270, 5, 15), 2))
colnames(concentration_data) <- metvar[1:10]
df <- cbind(df, as.data.frame(concentration_data))
covariates <- c("ID", "Diet", "Age", "BMI")
result <- diffGrpResponse(df, metvar, covariates)[[1]]
summary(result)
```

`diffGrpResponse_parallel`

Parallelized version of diffGrpResponse()

Description

Parallelized version of diffGrpResponse()

Usage

```
diffGrpResponse_parallel(
  df,
  met_vec,
  covariates,
  time_terms = c("Time", "log(Time)"),
  grp = "Diet",
  random_formula = ~1 + Time | ID/Diet,
  correlation_formula = corSymm(form = ~Time | ID/Diet),
  weights = varIdent(form = ~1 | Time)
)
```

Arguments

`df`	Data frame containing columns Group(numeric or character); ID(subject ID: character); Time(positive: numeric); other Time terms (numeric); other individual characteristics covariates; as well columns of metabolite concentrations
`met_vec`	Vector of metabolite names

`covariates`	Vector containing the names of the "ID" covariate, grouping covariate and other covariates excluding any "Time" covariates
`time_terms`	Vector that contains all additional form of the covariate 'Time" (including the 'Time' covariate), and must contain 'log(Time)', other forms also include I(Time^2) and I(Time^3);
`grp`	Grouping variable;
`random_formula`	Random effects formula for the model, nested effects of Diet within ID (could also add random slope for 'Time');
`correlation_formula`	Correlation formula. Default is autorgressive but can accommodate other forms such as unstructured covariance or exponential covariance;
`weights`	specify a variance function that models heteroscedasticity

Value

Data frame that contains the coefficient estimates, their corresponding p-values; LRT p-values for Time-Group interactions (for every 'Time' term);LRT p-values for Group and Time-Group interactions (for every 'Time' term); as well as the fitted models for each metabolite

Note

This function uses parallel processing via the ‘future.apply‘ package. To enable parallel execution, runs the following before calling this function:

library(future.apply) plan(multisession, workers = parallel::detectCores() - 1)

You only need to set the plan once per session.

References

Wickham, H. (2022). dplyr: A Grammar of Data Manipulation. R package version 1.0.10. Available at: https://CRAN.R-project.org/package=dplyr

Pinheiro, J. C., & Bates, D. M. (2022). nlme: Linear and Nonlinear Mixed Effects Models. R package version 3.1-153. Available at: https://CRAN.R-project.org/package=nlme

Examples

```
require(gammaFuncModel)
require(dplyr)
require(nlme)
df <- data.frame(
  ID = rep(sprintf("%02d", 1:10), each = 9 * 3),
  Time = rep(rep(1:9, each = 3), 10),
  Diet = as.factor(rep(1:3, times = 9 * 10)),
  Age = rep(sample(20:70, 10, replace = TRUE), each = 9 * 3),
  BMI = round(rep(runif(10, 18.5, 35), each = 9 * 3), 1)
)
metvar <- paste0("met", 1:10)
concentration_data <- replicate(10, round(runif(270, 5, 15), 2))
colnames(concentration_data) <- metvar[1:10]
```

```
df <- cbind(df, as.data.frame(concentration_data))
covariates <- c("ID", "Diet", "Age", "BMI")
library(future.apply)
plan(multisession, workers = parallel::detectCores() - 1)
result <- diffGrpResponse(df, metvar, covariates)[[1]]
summary(result)
```

`gammaFunction` *Implementation of the novel non-linear mixed-effects model based on gamma function form with nested covariance structure where random effects are specified for each Diet level within each subject (ID), capturing within-subject correlation across dietary conditions. to identify metabolites that responds to time differentially across dietary groups*

Description

Implementation of the novel non-linear mixed-effects model based on gamma function form with nested covariance structure where random effects are specified for each Diet level within each subject (ID), capturing within-subject correlation across dietary conditions. to identify metabolites that responds to time differentially across dietary groups

Usage

```
gammaFunction(
  data,
  covariates,
  time_terms = c("Time", "log(Time)"),
  grp = "Diet",
  random_formula = ~1 + Time | ID/Diet,
  correlation_formula = corSymm(form = ~Time | ID/Diet),
  weights = varIdent(form = ~1 | Time),
  time_grp_inter = TRUE,
  return_ml_model = FALSE,
  include_grp
)
```

Arguments

`data`	Data frame that contains the ’ID’ column along with all covariates as well as concentration column, named ’Concentration’, for a single metabolite
`covariates`	Vector containing the names of the "ID" covariate, grouping covariate and other covariates excluding any "Time" covariates
`time_terms`	Vector that contains all additional form of the covariate ’Time" (including the ’Time’ covariate), and must contain ’log(Time)’, other forms also include I(Time^2) and I(Time^3);
`grp`	Grouping variable;

`random_formula`	Random effects formula for the model, nested effects of Diet within ID (could also add random slope for 'Time');
`correlation_formula`	Correlation formula. Default is autorrgressive but can accomodate other forms such as unstructured covariance or exponential covariance;
`weights`	specify a variance function that models heteroscedasticity
`time_grp_inter`	Boolean value that indicates if the model should include interactions terms of 'time_terms' with 'Group';
`return_ml_model`	Boolean value that indicates if the model should fit "ML" model as well as "REML" model(default)
`include_grp`	boolean value to indicate whether or not 'grp' should be included in the model construction

Value

mixed effects models for a single metabolites: one with REML, the other with ML

References

Wickham, H. (2022). dplyr: A Grammar of Data Manipulation. R package version 1.0.10. Available at: https://CRAN.R-project.org/package=dplyr

Pinheiro, J. C., & Bates, D. M. (2022). nlme: Linear and Nonlinear Mixed Effects Models. R package version 3.1-153. Available at: https://CRAN.R-project.org/package=nlme

Examples

```
require(gammaFuncModel)
require(dplyr)
require(nlme)
df <- data.frame(
  ID = rep(sprintf("%02d", 1:10), each = 9 * 3),
  Time = rep(rep(1:9, each = 3), 10),
  Diet = as.factor(rep(1:3, times = 9 * 10)),
  Age = rep(sample(20:70, 10, replace = TRUE), each = 9 * 3),
  BMI = round(rep(runif(10, 18.5, 35), each = 9 * 3), 1),
  Concentration = round(runif(270, 5, 15), 2)
)
covariates <- c("ID", "Diet", "Age", "BMI")
model <- gammaFunction(
  df,
  covariates,
  random_formula = ~ 1  | ID/Diet,
  correlation_formula = corAR1(form = ~ Time | ID/Diet),
  weights = NULL,
  include_grp = TRUE)[[1]]
summary(model)
```

`generatePlot` *Function that generate plots for metabolite models*

Description

Function that generate plots for metabolite models

Usage

```
generatePlot(
  graph,
  df,
  met_vec,
  covariates,
  grp = "Diet",
  models,
  save_path = NULL
)
```

Arguments

`graph`	character string, 'None' by default. If not 'None, in addition to returning models, produces pdf file of graphs based on the specific value of 'graph'.
`df`	Data frame containing columns Group(factor); ID(subject ID: character); Time(positive: numeric); other Time terms (numeric); other individidual characteristics covariates; as well columns of metabolite concentrations
`met_vec`	the vector of metabolite names
`covariates`	Vector containing the names of the "ID" covariate, grouping covariate and other covariates excluding any "Time" covariates;
`grp`	is the grouping variable;
`models`	a list of fitted non-linear mixed effects metabolite models
`save_path`	location where the pdf file will be saved; default is NULL, i.e. pdf is saved to a temporary location

Value

A pdf file for fitted concenration curves that is saved to a user provided file location; otherwise saved to a temporary location

References

Wickham, H. (2022). dplyr: A Grammar of Data Manipulation. R package version 1.0.10. Available at: https://CRAN.R-project.org/package=dplyr

Pinheiro, J. C., & Bates, D. M. (2022). nlme: Linear and Nonlinear Mixed Effects Models. R package version 3.1-153. Available at: https://CRAN.R-project.org/package=nlme

Examples

```
require(gammaFuncModel)
require(dplyr)
require(nlme)
require(patchwork)
require(scales)
 df <- data.frame(
  ID = rep(sprintf("%02d", 1:10), each = 9 * 3),
  Time = rep(rep(1:9, each = 3), 10),
  Diet = as.factor(rep(1:3, times = 9 * 10)),
  Age = rep(sample(20:70, 10, replace = TRUE), each = 9 * 3),
  BMI = round(rep(runif(10, 18.5, 35), each = 9 * 3), 1)
)
metvar <- paste0("met", 1:10)
concentration_data <- replicate(10, round(runif(270, 5, 15), 2))
colnames(concentration_data) <- metvar[1:10]
df <- cbind(df, as.data.frame(concentration_data))
covariates <- c("ID", "Diet", "Age", "BMI")
mods <- generate_models(df = df, met_vec = metvar, covariates = covariates, graph = 'None')
generatePlot(
  graph = "individual_separated",
  df = df,
  met_vec = metvar,
  covariates = covariates,
  grp = "Diet",
  models = mods,
  save_path = NULL
 )
```

`generate_f_function` *Function produce predictions from the model*

Description

Function produce predictions from the model

Usage

```
generate_f_function(data, model, grp_var, grp_name = "Diet", ID, ref = 1)
```

Arguments

`data`	Data frame containing columns Group(factor); ID(subject ID: character); Time(positive: numeric); other individiual characteristics covariates (exlcluding other forms of 'Time')
`model`	Fitted model for the metabolite in question
`grp_var`	Value of the grouping variable

`grp_name`	Name of the grouping variable. Default is 'Diet'
`ID`	Subject ID
`ref`	reference group

Value

f function that produces the prediction from this model for a specific individual in a specific group

Examples

```
require(gammaFuncModel)
require(dplyr)
require(nlme)
modify.df <- data.frame(
  ID = rep(sprintf("%02d", 1:10), each = 9 * 3),
  Time = rep(rep(1:9, each = 3), 10),
  Diet = as.factor(rep(1:3, times = 9 * 10)),
  Age = rep(sample(20:70, 10, replace = TRUE), each = 9 * 3),
  BMI = round(rep(runif(10, 18.5, 35), each = 9 * 3), 1),
  Concentration = NA
)
for (i in 1:10) {
 for (d in 1:3) {
   C0 <- runif(1, 10, 15)    # initial concentration
   k <- runif(1, 0.1, 0.3)   # decay rate constant
   modify.df$Concentration[modify.df$ID == sprintf("%02d", i) & modify.df$Diet == d] <-
    C0 * exp(-k * modify.df$Time[modify.df$ID == sprintf("%02d", i) & modify.df$Diet == d])
 }
}
covariates <- c("ID", "Diet", "Age", "BMI")
model <- gammaFunction(
  modify.df,
  covariates,
  time_grp_inter = FALSE,
  return_ml_model = FALSE,
  include_grp = TRUE
 )[[1]]
test_data = modify.df %>%
  filter(Diet == 1 & ID == "04") %>%
  select(-c("Concentration", "ID", "Diet"))
f_dat = modify.df %>% filter(Diet == 1 & ID == "04") %>% select(-Concentration)
f <- generate_f_function(
  data = f_dat,
  model = model,
  grp_var = 1,
  grp_name = "Diet",
  ID = "04",
  ref = 1
 )
```

`generate_models` *Function that produces a fitted gamma model for each metabolite*

Description

Function that produces a fitted gamma model for each metabolite

Usage

```
generate_models(
  df,
  met_vec,
  covariates,
  time_terms = c("Time", "log(Time)"),
  grp_name = "Diet",
  random_formula = ~1 + Time | ID/Diet,
  correlation_formula = corSymm(form = ~Time | ID/Diet),
  weights = varIdent(form = ~1 | Time),
  graph = "None",
  save_path = NULL
)
```

Arguments

`df`	Data frame containing columns Group(factor); ID(subject ID: character); Time(positive: numeric); other Time terms (numeric); other individidual characteristics covariates; as well columns of metabolite concentrations
`met_vec`	the vector of metabolite names
`covariates`	Vector containing the names of the "ID" covariate, grouping covariate and other covariates excluding any "Time" covariates
`time_terms`	is the vector that contains all additional form of the covariate 'Time" (including the 'Time' covariate), and must contain 'log(Time)', other forms also include I(Time^2) and I(Time^3);
`grp_name`	is the grouping variable;
`random_formula`	is the random effects formula for the model, nested effects of Diet within ID (could also add random slope for 'Time');
`correlation_formula`	is the correlation formula. Default is autorgressive but can accommodate other forms such as unstructured covariance or exponential covariance;
`weights`	specify a variance function that models heteroscedasticity;
`graph`	character string, 'None' by default. If not 'None, in addition to returning models, produces pdf file of graphs based on the specific value of 'graph'.
`save_path`	location where the pdf file will be saved; default is NULL, i.e. pdf is saved to a temporary location

Value

List that contains fitted models for each metabolite and a pdf file for fitted concenration curves.

References

Wickham, H. (2022). dplyr: A Grammar of Data Manipulation. R package version 1.0.10. Available at: https://CRAN.R-project.org/package=dplyr

Pinheiro, J. C., & Bates, D. M. (2022). nlme: Linear and Nonlinear Mixed Effects Models. R package version 3.1-153. Available at: https://CRAN.R-project.org/package=nlme

Examples

```
require(gammaFuncModel)
require(dplyr)
require(nlme)
 df <- data.frame(
  ID = rep(sprintf("%02d", 1:10), each = 9 * 3),
  Time = rep(rep(1:9, each = 3), 10),
  Diet = as.factor(rep(1:3, times = 9 * 10)),
  Age = rep(sample(20:70, 10, replace = TRUE), each = 9 * 3),
  BMI = round(rep(runif(10, 18.5, 35), each = 9 * 3), 1)
)
metvar <- paste0("met", 1:10)
concentration_data <- replicate(10, round(runif(270, 5, 15), 2))
colnames(concentration_data) <- metvar[1:10]
df <- cbind(df, as.data.frame(concentration_data))
covariates <- c("ID", "Diet", "Age", "BMI")
mods <- generate_models(
  df = df,
 met_vec = metvar,
  covariates = covariates,
  graph = 'None',
  save_path = NULL)
```

`grpResp2Time` *Function that produces a summary table for coefficient estimates, their p-values and LRT p-values for every metabolite in the dataframe, for a single Group*

Description

Function that produces a summary table for coefficient estimates, their p-values and LRT p-values for every metabolite in the dataframe, for a single Group

Usage

```
grpResp2Time(
  df,
  met_vec,
  covariates,
  time_terms = c("Time", "log(Time)"),
  grp = "Diet",
  random_formula = ~1 | ID,
  correlation_formula = corAR1(form = ~Time | ID),
  weights = NULL
)
```

Arguments

`df`	Data frame containing information for a single group, containing columns grp; ID(subject ID: character); Time(positive: numeric); other Time terms (numeric); other individual characteristics covariates; as well columns of metabolite concentrations
`met_vec`	Vector of metabolite names
`covariates`	Vector containing the names of the "ID" covariate, grouping covariate and other covariates excluding any "Time" covariates
`time_terms`	Vector that contains all additional form of the covariate 'Time" (including the 'Time' covariate), and must contain 'log(Time)', other forms also include I(Time^2) and I(Time^3);
`grp`	Grouping variable (should be a single valued column);
`random_formula`	Random effects formula for the model, within ID (could also add random slope for 'Time');
`correlation_formula`	Correlation formula. Default is autorgressive but can accommodate other forms such as unstructured covariance or exponential covariance;
`weights`	specify a variance function that models heteroscedasticity

Value

Data frame that contains the coefficient estimates, their corresponding p-values as well as LRT p-values for 'Time' terms

References

Wickham, H. (2022). dplyr: A Grammar of Data Manipulation. R package version 1.0.10. Available at: https://CRAN.R-project.org/package=dplyr

Pinheiro, J. C., & Bates, D. M. (2022). nlme: Linear and Nonlinear Mixed Effects Models. R package version 3.1-153. Available at: https://CRAN.R-project.org/package=nlme

Examples

```
require(gammaFuncModel)
require(dplyr)
require(nlme)
df <- data.frame(
  ID = rep(sprintf("%02d", 1:10), each = 9 * 3),
  Time = rep(rep(1:9, each = 3), 10),
  Diet = as.factor(rep(1:3, times = 9 * 10)),
  Age = rep(sample(20:70, 10, replace = TRUE), each = 9 * 3),
  BMI = round(rep(runif(10, 18.5, 35), each = 9 * 3), 1)
)
metvar <- paste0("met", 1:10)
concentration_data <- replicate(10, round(runif(270, 5, 15), 2))
colnames(concentration_data) <- metvar[1:10]
df <- cbind(df, as.data.frame(concentration_data))
df_single_diet <- subset(df, Diet == 1)
covariates <- c("ID","Diet", "Age", "BMI")
result_SD <- grpResp2Time(df_single_diet, metvar, covariates)[[1]]
summary(result_SD)
```

`grpResp2Time_parallel` *Vectorized version of grpRes2Time()*

Description

Vectorized version of grpRes2Time()

Usage

```
grpResp2Time_parallel(
  df,
  met_vec,
  covariates,
  time_terms = c("Time", "log(Time)"),
  grp = "Diet",
  random_formula = ~1 | ID,
  correlation_formula = corAR1(form = ~Time | ID),
  weights = NULL
)
```

Arguments

`df`	Data frame containing information for a single group, containing columns grp; ID(subject ID: character); Time(positive: numeric); other Time terms (numeric); other individual characteristics covariates; as well columns of metabolite concentrations
`met_vec`	Vector of metabolite names

`covariates`	Vector containing the names of the "ID" covariate, grouping covariate and other covariates excluding any "Time" covariates
`time_terms`	Vector that contains all additional form of the covariate 'Time" (including the 'Time' covariate), and must contain 'log(Time)', other forms also include I(Time^2) and I(Time^3);
`grp`	Grouping variable (should be a single valued column);
`random_formula`	Random effects formula for the model, within ID (could also add random slope for 'Time');
`correlation_formula`	Correlation formula. Default is autorgressive but can accommodate other forms such as unstructured covariance or exponential covariance;
`weights`	specify a variance function that models heteroscedasticity

Value

Data frame that contains the coefficient estimates, their corresponding p-values as well as LRT p-values for 'Time' terms

Note

This function uses parallel processing via the 'future.apply' package. To enable parallel execution, runs the following before calling this function:

library(future.apply) plan(multisession, workers = parallel::detectCores() - 1)

You only need to set the plan once per session.

References

Wickham, H. (2022). dplyr: A Grammar of Data Manipulation. R package version 1.0.10. Available at: https://CRAN.R-project.org/package=dplyr

Pinheiro, J. C., & Bates, D. M. (2022). nlme: Linear and Nonlinear Mixed Effects Models. R package version 3.1-153. Available at: https://CRAN.R-project.org/package=nlme

Examples

```
require(gammaFuncModel)
require(dplyr)
require(nlme)
df <- data.frame(
  ID = rep(sprintf("%02d", 1:10), each = 9 * 3),
  Time = rep(rep(1:9, each = 3), 10),
  Diet = as.factor(rep(1:3, times = 9 * 10)),
  Age = rep(sample(20:70, 10, replace = TRUE), each = 9 * 3),
  BMI = round(rep(runif(10, 18.5, 35), each = 9 * 3), 1)
)
metvar <- paste0("met", 1:10)
concentration_data <- replicate(10, round(runif(270, 5, 15), 2))
colnames(concentration_data) <- metvar[1:10]
df <- cbind(df, as.data.frame(concentration_data))
```

```
df_single_diet <- subset(df, Diet == 1)
covariates <- c("ID","Diet", "Age", "BMI")
library(future.apply)
plan(multisession, workers = parallel::detectCores() - 1)
result_SD <- grpResp2Time_parallel(df_single_diet, metvar, covariates)[[1]]
summary(result_SD)
```

`pk_calculation` *Function that returns a data frame for Tmax, Cmax, half-life, AUC and AUCInf for metabolites*

Description

Function that returns a data frame for Tmax, Cmax, half-life, AUC and AUCInf for metabolites

Usage

```
pk_calculation(df, met_vec, models, grp_name = "Diet", covariates, ref = 1)
```

Arguments

`df`	Data frame containing columns Group(factor); ID(subject ID: character); Time(positive: numeric); other individiual characteristics covariates (exlcluding other forms of 'Time')
`met_vec`	Vector of metabolite names
`models`	Fitted models for all metabolites of interest
`grp_name`	Name of the grouping variable
`covariates`	Vector containing the names of the "ID" covariate, grouping covariate and other covariates excluding any "Time" covariates
`ref`	reference level for the grouping variable. could be numeric or character

Value

Data frame with the pharmacokinetic properties of each metabolite

References

Wickham, H. (2022). dplyr: A Grammar of Data Manipulation. R package version 1.0.10. Available at: https://CRAN.R-project.org/package=dplyr

Pinheiro, J. C., & Bates, D. M. (2022). nlme: Linear and Nonlinear Mixed Effects Models. R package version 3.1-153. Available at: https://CRAN.R-project.org/package=nlme

Examples

```
require(gammaFuncModel)
require(dplyr)

df <- data.frame(
 ID = rep(sprintf("%02d", 1:10), each = 9 * 3),
 Time = rep(rep(1:9, each = 3), 10),
 Diet = as.factor(rep(1:3, times = 9 * 10)),
 Age = rep(sample(20:70, 10, replace = TRUE), each = 9 * 3),
 BMI = round(rep(runif(10, 18.5, 35), each = 9 * 3), 1)
)
metvar <- paste0("met", 1:10)
n_rows <- nrow(df)
concentration_data <- sapply(1:10, function(m) {
 shape <- runif(1, 2, 5)
 scale <- runif(1, 1, 3)
 rgamma(n_rows, shape = shape, scale = scale)
})
colnames(concentration_data) <- metvar
df <- cbind(df, as.data.frame(concentration_data))
covariates <- c("ID", "Diet", "Age", "BMI")
mods <- generate_models(df = df, met_vec = metvar, covariates = covariates, graph = 'None')
result <- pk_calculation(
  df = df,
  met_vec = metvar,
  models = mods,
  grp_name = "Diet",
  covariates = covariates
 )
```

Index