Package: eggla (via r-universe)

August 23, 2024

Title Early Growth Genetics Longitudinal Analysis

Version 1.0.1

Description Tools for longitudinal analysis within the EGG (Early Growth Genetics) Consortium (<http://egg-consortium.org/>).

License MIT + file LICENSE

URL https://github.com/mcanouil/eggla, https://m.canouil.dev/eggla/

BugReports https://github.com/mcanouil/eggla/issues

Depends R (>= 4.2)

Imports stats, nlme, utils, grDevices, data.table (>= 1.15.0), future.apply (>= 1.11.1), growthcleanr (>= 2.2.0), ggplot2 (>= 3.5.0), ggtext (>= 0.1.2), ggdist (>= 3.3.1), ggbeeswarm (>= 0.7.2), patchwork (>= 1.2.0), broom.mixed (>= 0.2.9.4), rlang (>= 1.1.3), performance (>= 0.10.9)

Suggests knitr (>= 1.42), rmarkdown (>= 2.20), R.utils (>= 2.12.2), scales (>= 1.2.1), future (>= 1.32.0), gt (>= 0.8.0), lme4 (>= 1.1.31), see (>= 0.7.4), qqplotr (>= 0.0.6), future.callr (>= 0.8.1), forcats (>= 1.0.0), roxygen2 (>= 7.2.3), testthat (>= 3.1.7)

Remotes github::carriedaymont/growthcleanr@v2.2.0

SystemRequirements BCFtools (>= 1.16) (https://github.com/samtools/BCFtools), PLINK2 (>= 2.0) (https://www.cog-genomics.org/plink/2.0)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

VignetteBuilder knitr

Config/testthat/edition 3

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

RemoteUrl https://github.com/mcanouil/eggla

RemoteRef HEAD

RemoteSha eceaaa87522143ebbede65ab183af5be27fb6322

Contents

`bmigrowth` *BMI Measurements For 100 Individuals From 0 To 17 Years.*

Description

A dataset containing the age, sex, weight, height and BMI for 100 individuals. Measurements performed from birth to 17 years old.

Usage

```
bmigrowth
```

Format

A data frame with 1050 rows and 6 variables:

ID (character) ID using three digits.

age (numeric) age in years.

sex (integer) sex with 1: male and 0: female.

weight (numeric) weight in kilograms.

height (integer) height in centimetres.

bmi (numeric) Body Mass Index in kilograms per quare metre.

---

compute_apar *Compute adiposity peak (AP) and adiposity rebound (AR).*

Description

Compute adiposity peak (AP) and adiposity rebound (AR).

Usage

```
compute_apar(
  fit,
  from = c("predicted", "observed"),
  start = 0.25,
  end = 10,
  step = 0.01,
  filter = NULL
)
```

Arguments

fit	A model object from a statistical model such as from a call `nlme::lme()`, `time_model()` or `egg_model()`.
from	A string indicating the type of data to be used for the AP and AR computation, either "predicted" or "observed". Default is "predicted".
start	The start of the time window to compute AP and AR.
end	The end of the time window to compute AP and AR.
step	The step to increment the sequence.
filter	A string following `data.table` syntax for filtering on `"i"` (*i.e.*, row elements), *e.g.*, `filter = "source == 'A'"`. Argument pass through `compute_apar()` (see `predict_bmi()`). Default is `NULL`.

Value

A `data.table` object.

Examples

```
library(eggla)
data("bmigrowth")
res <- egg_model(
  formula = log(bmi) ~ age,
  data = bmigrowth[bmigrowth[["sex"]] == 0, ],
  id_var = "ID",
  random_complexity = 1
)

head(compute_apar(fit = res, from = "predicted")[AP | AR])

# Comparing observed and predicted values
library(data.table)
library(ggplot2)
library(patchwork)
list_gg <- melt(
  data = rbindlist(
    l = lapply(
      X = (function(.x) `names<-`(.x, .x))(c("predicted", "observed")),
      FUN = compute_apar,
      fit = res
    ),
    idcol = "from"
  )[
    AP | AR
  ][
    j = what := fifelse(paste(AP, AR) %in% paste(FALSE, TRUE), "AR", "AP")
  ],
  id.vars = c("from", "egg_id", "what"),
  measure.vars = c("egg_ageyears", "egg_bmi")
)[
  j = list(gg = list({
    dt <- dcast(data = .SD, formula = egg_id + what ~ from)
    range_xy <- range(dt[, c("observed", "predicted")], na.rm = TRUE)
    ggplot(data = dt) +
      aes(x = observed, y = predicted, colour = what) +
      geom_abline(intercept = 0, slope = 1) +
      geom_segment(aes(xend = observed, yend = observed), alpha = 0.5) +
      geom_point() +
      scale_colour_manual(values = c("#E69F00FF", "#56B4E9FF")) +
      labs(
        x = sprintf("Observed: %s", sub(".*_", "", toupper(variable))),
        y = sprintf("Predicted: %s", sub(".*_", "", toupper(variable))),
        colour = NULL,
        title = sub(".*_", "", toupper(variable))
      ) +
      coord_cartesian(xlim = range_xy, ylim = range_xy)
  })),
  by = "variable"
]
wrap_plots(list_gg[["gg"]], guides = "collect")
```

compute_aucs *Compute area under the curves for several intervals using a model fitted by* time_model().

Description

Compute area under the curves for "clubic slope", "linear splines" and "cubic splines" fitted using time_model().

Usage

```
compute_aucs(
  fit,
  method,
  period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17),
 knots = list(cubic_slope = NULL, linear_splines = c(0.75, 5.5, 11), cubic_splines =
    c(1, 8, 12))[[method]]
)
```

Arguments

fit	A model object from a statistical model such as from a call to time_model().
method	The type of model provided in fit, *i.e.*, one of "cubic_slope", "linear_splines" or "cubic_splines".
period	The intervals knots on which AUCs are to be computed.
knots	The knots as defined fit and according to method.

Value

A data.frame with AUC for each individuals/samples.

Examples

```
data("bmigrowth")
ls_mod <- time_model(
  x = "age",
  y = "log(bmi)",
  cov = NULL,
  data = bmigrowth[bmigrowth[["sex"]] == 0, ],
  method = "linear_splines"
)
head(compute_aucs(
  fit = ls_mod,
  method = "linear_splines",
  period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17)#,
  # knots = list(
  #   "cubic_slope" = NULL,
  #   "linear_splines" = c(0.75, 5.5, 11),
```

```
  #   "cubic_splines" = c(1, 8, 12)
  # )[[method]]
))
```

`compute_correlations` *Compute the derived parameters correlations from a cubic splines mixed-effects model by* `time_model()`.

Description

Based on computed area under the curves (*i.e.*, `compute_aucs()`) and slopes (*i.e.*, `compute_slopes()`) for several intervals using a model fitted by `time_model()`, compute the correlations between each intervals derived parameters.

Usage

```
compute_correlations(
  fit,
  method,
  period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17),
 knots = list(cubic_slope = NULL, linear_splines = c(0.75, 5.5, 11), cubic_splines =
    c(1, 8, 12))[[method]]
)
```

Arguments

`fit`	A model object from a statistical model such as from a call to `time_model()`.
`method`	The type of model provided in `fit`, *i.e.*, one of `"cubic_slope"`, `"linear_splines"` or `"cubic_splines"`.
`period`	The intervals knots on which AUCs are to be computed.
`knots`	The knots as defined `fit` and according to `method`.

Value

A `list` object with correlations between each intervals derived parameters.

Examples

```
data("bmigrowth")
ls_mod <- time_model(
  x = "age",
  y = "log(bmi)",
  cov = NULL,
  data = bmigrowth[bmigrowth[["sex"]] == 0, ],
  method = "linear_splines"
)
compute_correlations(
  fit = ls_mod,
```

```
  method = "linear_splines",
  period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17)#,
  # knots = list(
  #   "cubic_slope" = NULL,
  #   "linear_splines" = c(0.75, 5.5, 11),
  #   "cubic_splines" = c(1, 8, 12)
  # )[[method]]
)
```

`compute_outliers` *Compute outliers detection in derived parameters from a cubic splines mixed-effects model by* `time_model()`.

Description

Based on computed area under the curves (*i.e.*, `compute_aucs()`) and slopes (*i.e.*, `compute_slopes()`) for several intervals using a model fitted by `time_model()`, compute an outlier detection. For details, see methods `iqr` and `zscore` of `performance::check_outliers()`.

Usage

```
compute_outliers(
  fit,
  method,
  period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17),
 knots = list(cubic_slope = NULL, linear_splines = c(0.75, 5.5, 11), cubic_splines =
    c(1, 8, 12))[[method]],
  from = c("predicted", "observed"),
  start = 0.25,
  end = 10,
  step = 0.01,
  filter = NULL,
  outlier_method = "iqr",
  outlier_threshold = list(iqr = 2)
)
```

Arguments

`fit`	A model object from a statistical model such as from a call to `time_model()`.
`method`	The type of model provided in `fit`, *i.e.*, one of `"cubic_slope"`, `"linear_splines"` or `"cubic_splines"`.
`period`	The intervals knots on which AUCs are to be computed.
`knots`	The knots as defined `fit` and according to `method`.
`from`	A string indicating the type of data to be used for the AP and AR computation, either "predicted" or "observed". Default is "predicted".
`start`	The start of the time window to compute AP and AR.

end	The end of the time window to compute AP and AR.
step	The step to increment the sequence.
filter	A string following data.table syntax for filtering on "i" (*i.e.*, row elements), *e.g.*, filter = "source == 'A'". Argument pass through compute_apar() (see predict_bmi()). Default is NULL.
outlier_method	The outlier detection method(s). Default is "iqr". Can be "cook", "pareto", "zscore", "zscore_robust", "iqr", "ci", "eti", "hdi", "bci", "mahalanobis", "mahalanobis_robust", "mcd", "ics", "optics" or "lof". See performance::check_outliers() https://easystats.github.io/performance/reference/check_outliers.html for details.
outlier_threshold	A list containing the threshold values for each method (*e.g.*, list('mahalanobis' = 7, 'cook' = 1)), above which an observation is considered as outlier. If NULL, default values will be used (see 'Details'). If a numeric value is given, it will be used as the threshold for any of the method run. See performance::check_outliers() https://easystats.github.io/performance/reference/check_outliers.html for details.

Value

A data.frame listing the individuals which are not outliers based on several criteria.

Examples

```
data("bmigrowth")
ls_mod <- time_model(
  x = "age",
  y = "log(bmi)",
  cov = NULL,
  data = bmigrowth[bmigrowth[["sex"]] == 0, ],
  method = "cubic_splines"
)
head(compute_outliers(
  fit = ls_mod,
  method = "cubic_splines",
  period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17)#,
  # knots = list(
  #   "cubic_slope" = NULL,
  #   "linear_splines" = c(0.75, 5.5, 11),
  #   "cubic_splines" = c(1, 8, 12)
  # )[[method]]
)[Outlier != 0])
```

---

compute_slopes *Predict average slopes for several intervals using a model fitted by* time_model().

Description

Comoute average slopes for "clubic slope", "linear splines" and "cubic splines" fitted using `time_model()`.

Usage

```
compute_slopes(
  fit,
  method,
  period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17),
 knots = list(cubic_slope = NULL, linear_splines = c(0.75, 5.5, 11), cubic_splines =
    c(1, 8, 12))[[method]]
)
```

Arguments

`fit`	A model object from a statistical model such as from a call to `time_model()`.
`method`	The type of model provided in `fit`, *i.e.*, one of `"cubic_slope"`, `"linear_splines"` or `"cubic_splines"`.
`period`	The intervals knots on which slopes are to be computed.
`knots`	The knots as defined `fit` and according to `method`.

Value

A `data.frame` with slopes for each individuals/samples.

Examples

```
data("bmigrowth")
ls_mod <- time_model(
  x = "age",
  y = "log(bmi)",
  cov = NULL,
  data = bmigrowth[bmigrowth[["sex"]] == 0, ],
  method = "linear_splines"
)
head(compute_slopes(
  fit = ls_mod,
  method = "linear_splines",
  period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17)#,
  # knots = list(
  #   "cubic_slope" = NULL,
  #   "linear_splines" = c(0.75, 5.5, 11),
  #   "cubic_splines" = c(1, 8, 12)
  # )[[method]]
))
```

egg_aucs *Derived areas under the curve from a cubic splines mixed-effects model by* egg_model().

Description

Derived areas under the curve (AUCs) for differentintervals based on a fitted cubic splines mixed-effects model from egg_model(). This function is a specific version of compute_aucs designed to work specifically on egg_model().

Usage

```
egg_aucs(
  fit,
  period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17),
  knots = c(1, 8, 12)
)
```

Arguments

fit	A model object from a statistical model such as from a call to egg_model().
period	The intervals knots on which AUCs are to be computed.
knots	The knots as defined fit and according to method.

Value

A data.frame with AUC for each individuals/samples.

Examples

```
data("bmigrowth")
res <- egg_model(
  formula = log(bmi) ~ age,
  data = bmigrowth[bmigrowth[["sex"]] == 0, ],
  id_var = "ID",
  random_complexity = 1
)
head(
  egg_aucs(
    fit = res,
    period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17),
    knots = c(1, 8, 12)
  )
)
```

`egg_correlations` *Compute the derived parameters correlations from a cubic splines mixed-effects model by* `egg_model()`.

Description

Based on computed area under the curves (*i.e.*, `egg_aucs()`) and slopes (*i.e.*, `egg_slopes()`) for several intervals using a model fitted by `egg_model()`, compute the correlations between each intervals derived parameters.

Usage

```
egg_correlations(
  fit,
  period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17),
  knots = c(1, 8, 12),
  start = 0.25,
  end = 10,
  step = 0.01,
  filter = NULL
)
```

Arguments

`fit`	A model object from a statistical model such as from a call to `egg_model()`.
`period`	The intervals knots on which slopes are to be computed.
`knots`	The knots as defined `fit` and according to `method`.
`start`	The start of the time window to compute AP and AR.
`end`	The end of the time window to compute AP and AR.
`step`	The step to increment the sequence.
`filter`	A string following `data.table` syntax for filtering on `"i"` (*i.e.*, row elements), *e.g.*, `filter = "source == 'A'"`. Argument pass through `compute_apar()` (see `predict_bmi()`). Default is `NULL`.

Value

A `data.table` object with correlations between each intervals derived parameters.

Examples

```
data("bmigrowth")
res <- egg_model(
  formula = log(bmi) ~ age,
  data = bmigrowth[bmigrowth[["sex"]] == 0, ],
  id_var = "ID",
  random_complexity = 1
```

```
)
egg_correlations(
  fit = res,
  period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17),
  knots = c(1, 8, 12)
)
```

`egg_model` *Fit a cubic splines mixed model.*

Description

Fit a cubic splines mixed model regression with three splines parametrisation as random effect. This function is a specific version of `time_model()`.

Usage

```
egg_model(
  formula,
  data,
  id_var,
  random_complexity = "auto",
  use_car1 = FALSE,
  knots = c(1, 8, 12),
  quiet = FALSE
)
```

Arguments

`formula`	An object of class "formula": a symbolic description of the model to be fitted with, time component as the first term in the right-hand side.
`data`	A data.frame containing the variables defined in formula.
`id_var`	A string indicating the name of the variable to be used as the individual identifier.
`random_complexity`	A numeric (1-3) indicating the complexity of the random effect term. Default, `"auto"` will try from the more complex to the less complex if no success.
`use_car1`	A logical indicating whether to use continuous auto-correlation, i.e., CAR(1) as correlation structure.
`knots`	The knots defining the splines.
`quiet`	A logical indicating whether to suppress the output.

Value

An object of class "lme" representing the linear mixed-effects model fit.

Examples

```
data("bmigrowth")
res <- egg_model(
  formula = log(bmi) ~ age,
  data = bmigrowth[bmigrowth[["sex"]] == 0, ],
  id_var = "ID",
  random_complexity = 1
)
sres <- as.data.frame(summary(res)[["tTable"]])
rownames(sres) <- sub("gsp\\(.*\\)\\)", "gsp(...)", rownames(sres))
sres
```

`egg_outliers` *Compute outliers detection in AUCs/Slopes derived parameters from a cubic splines mixed-effects model by* `egg_model()`.

Description

Based on computed area under the curves (*i.e.*, `egg_aucs()`) and slopes (*i.e.*, `egg_slopes()`) for several intervals using a model fitted by `egg_model()`, compute an outlier detection. For details, see methods `iqr` and `zscore` of `performance::check_outliers()`.

Usage

```
egg_outliers(
  fit,
  period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17),
  knots = c(1, 8, 12),
  from = c("predicted", "observed"),
  start = 0.25,
  end = 10,
  step = 0.01,
  filter = NULL,
  outlier_method = "iqr",
  outlier_threshold = list(iqr = 2)
)
```

Arguments

`fit`	A model object from a statistical model such as from a call to `egg_model()`.
`period`	The intervals knots on which slopes are to be computed.
`knots`	The knots as defined `fit` and according to `method`.
`from`	A string indicating the type of data to be used for the AP and AR computation, either "predicted" or "observed". Default is "predicted".
`start`	The start of the time window to compute AP and AR.
`end`	The end of the time window to compute AP and AR.

step — The step to increment the sequence.

filter — A string following data.table syntax for filtering on "i" (*i.e.*, row elements), *e.g.*, filter = "source == 'A'". Argument pass through compute_apar() (see predict_bmi()). Default is NULL.

outlier_method — The outlier detection method(s). Default is "iqr". Can be "cook", "pareto", "zscore", "zscore_robust", "iqr", "ci", "eti", "hdi", "bci", "mahalanobis", "mahalanobis_robust", "mcd", "ics", "optics" or "lof". See performance::check_outliers() https://easystats.github.io/performance/reference/check_outliers.html for details.

outlier_threshold — A list containing the threshold values for each method (*e.g.*, list('mahalanobis' = 7, 'cook' = 1)), above which an observation is considered as outlier. If NULL, default values will be used (see 'Details'). If a numeric value is given, it will be used as the threshold for any of the method run. See performance::check_outliers() https://easystats.github.io/performance/reference/check_outliers.html for details.

Value

A data.frame listing the individuals which are not outliers based on several criteria.

Examples

```
data("bmigrowth")
res <- egg_model(
  formula = log(bmi) ~ age,
  data = bmigrowth[bmigrowth[["sex"]] == 0, ],
  id_var = "ID",
  random_complexity = 1
)
head(egg_outliers(
  fit = res,
  period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17),
  knots = c(1, 8, 12)
)[Outlier != 0])
```

---

egg_slopes — *Derived slopes from a cubic splines mixed-effects model by* egg_model().

Description

Derived slopes for different intervals based on a fitted cubic splines mixed-effects model from egg_model(). This function a specific version of compute_slopes designed to work specifically on egg_model().

Usage

```
egg_slopes(
  fit,
  period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17),
  knots = c(1, 8, 12)
)
```

Arguments

fit	A model object from a statistical model such as from a call to egg_model().
period	The intervals knots on which slopes are to be computed.
knots	The knots as defined fit and according to method.

Value

A data.frame with slopes for each individuals/samples.

Examples

```
data("bmigrowth")
res <- egg_model(
  formula = log(bmi) ~ age,
  data = bmigrowth[bmigrowth[["sex"]] == 0, ],
  id_var = "ID",
  random_complexity = 1
)
head(
  egg_slopes(
    fit = res,
    period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17),
    knots = c(1, 8, 12)
  )
)
```

---

gsp *General regression splines with variable degrees and ness, smoothing splines.*

Description

From https://github.com/gmonette/spida2 because namespace and dependencies are not properly listed. Source: https://github.com/gmonette/spida2/blob/master/R/gsp.R, https://github.com/gmonette/spida2/blob/master/R/

Usage

```
gsp(
  x,
  knots,
  degree = 3,
  smoothness = pmax(pmin(degree[-1], degree[-length(degree)]) - 1, -1),
  lin = NULL,
  periodic = FALSE,
  intercept = 0,
  signif = 3
)
```

Arguments

x	value(s) where spline is evaluated.
knots	vector of knots.
degree	vector giving the degree of the spline in each interval. Note the number of intervals is equal to the number of knots + 1. A value of 0 corresponds to a constant in the interval. If the spline should evaluate to 0 in the interval, use the `intercept` argument to specify some value in the interval at which the spline must evaluate to 0.
smoothness	vector with the degree of smoothness at each knot (0 = continuity, 1 = smoothness with continuous first derivative, 2 = continuous second derivative, etc. The value -1 allows a discontinuity at the knot. A scalar is recycled so its length equals the number of knots. Alternatively, a list of length equal to the number of knots. Each element of the list is a vector of the orders of derivatives which are required to be smooth. THis allows non-sequential constraints, *e.g.*, to have the same first and second derivative on either side of a knot but a possible discontinuity and change in higher-order derivatives, the vector would be c(1,2). Note that if a list is used, all elements must provide all desired constraints. That is the list argument corresponding to `smoothness = c(1,2,-1)` is `smoothness=list(0:1, 0:2, -1)`.
lin	provides a matrix specifying additional linear contraints on the 'full' parametrization consisting of blocks of polynomials of degree equal to max(degree) in each of the length(knots)+1 intervals of the spline. See below for examples of a spline that is 0 outside of its boundary knots.
periodic	if TRUE generates a period spline on the base interval (0,max(knots)). A constraint is generated so that the coefficients generate the same values to the right of max(knots) as they do to the right of 0. Note that all knots should be strictly positive.
intercept	value(s) of x at which the spline has value 0, *i.e.*, the value(s) of x for which yhat is estimated by the intercept term in the model. The default is 0. If NULL, the spline is not constrained to evaluate to 0 for any x.
signif	number of significant digits used to label coefficients.

Value

gsp returns a matrix generating a spline.

Author(s)

Monette, G. <georges@yorku.ca>

Examples

```
simd <- data.frame(
  age = rep(1:50, 2),
  y = sin(2 * pi * (1:100) / 5) + rnorm(100),
  G = rep(c("male", "female"), c(50, 50))
)
sp <- function(x) {
  gsp(x, knots = c(10, 25, 40), degree = c(1, 2, 2, 1), smoothness = c(1, 1 ,1))
}

summary(lm(formula = y ~ sp(age) * G, data = simd))
```

---

plot_aucs *Plot derived area under the curves from a model fitted by* time_model().

Description

Plot derived area under the curves from a model fitted by time_model().

Usage

```
plot_aucs(
  fit,
  method,
  period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17),
 knots = list(cubic_slope = NULL, linear_splines = c(0.75, 5.5, 11), cubic_splines =
    c(1, 8, 12))[[method]]
)
```

Arguments

fit	A model object from a statistical model such as from a call to time_model().
method	The type of model provided in fit, *i.e.*, one of "cubic_slope", "linear_splines" or "cubic_splines".
period	The intervals knots on which AUCs are to be computed.
knots	The knots as defined fit and according to method.

Value

A patchwork ggplot2 object.

Examples

```
library(ggplot2)
library(eggla)
data("bmigrowth")
ls_mod <- time_model(
  x = "age",
  y = "log(bmi)",
  cov = NULL,
  data = bmigrowth[bmigrowth[["sex"]] == 0, ],
  method = "linear_splines"
)
plot_aucs(
  fit = ls_mod,
  method = "linear_splines"
)
```

`plot_egg_aucs` *Plot derived area under the curves from a model fitted by* `egg_model()`.

Description

Plot derived area under the curves from a model fitted by `egg_model()`.

Usage

```
plot_egg_aucs(
  fit,
  period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17),
  knots = c(1, 8, 12)
)
```

Arguments

`fit`	A model object from a statistical model such as from a call to `egg_model()`.
`period`	The intervals knots on which AUCs are to be computed.
`knots`	The knots as defined `fit` and according to `method`.

Value

A patchwork ggplot2 object.

Examples

```
library(ggplot2)
library(eggla)
data("bmigrowth")
res <- egg_model(
  formula = log(bmi) ~ age,
  data = bmigrowth[bmigrowth[["sex"]] == 0, ],
  id_var = "ID",
  random_complexity = 1
)
plot_egg_aucs(
  fit = res,
  period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17)
)
```

---

plot_egg_slopes *Plot derived slopes from a model fitted by* egg_model().

Description

Plot derived slopes from a model fitted by egg_model().

Usage

```
plot_egg_slopes(
  fit,
  period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17),
  knots = c(1, 8, 12)
)
```

Arguments

fit	A model object from a statistical model such as from a call to egg_model().
period	The intervals knots on which slopes are to be computed.
knots	The knots as defined fit and according to method.

Value

A patchwork ggplot2 object.

Examples

```
library(ggplot2)
library(eggla)
data("bmigrowth")
res <- egg_model(
  formula = log(bmi) ~ age,
  data = bmigrowth[bmigrowth[["sex"]] == 0, ],
```

```
  id_var = "ID",
  random_complexity = 1
)
plot_egg_slopes(
  fit = res,
  period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17)
)
```

`plot_residuals` *Plot several residuals plots for diagnostics.*

Description

Plot several residuals plots for diagnostics.

Usage

```
plot_residuals(x, y, fit)
```

Arguments

`x`	A length one character vector with the main covariate name (*i.e.*, right-hand side), as defined in `fit`.
`y`	A length one character vector with the variable name to be explained (*i.e.*, left-hand side), as defined in `fit`.
`fit`	A model object from a statistical model such as from a call `time_model()` or `egg_model()`.

Value

A `patchwork` `ggplot2` object.

Examples

```
library(ggplot2)
library(patchwork)
library(eggla)
data("bmigrowth")
res <- egg_model(
  formula = log(bmi) ~ age,
  data = bmigrowth[bmigrowth[["sex"]] == 0, ],
  id_var = "ID",
  random_complexity = 1
)
plot_residuals(
  x = "age",
  y = "log(bmi)",
  fit = res
) +
```

```
  plot_annotation(
    title = "Cubic Splines (Random Linear Splines) - BMI - Female",
    tag_levels = "A"
  )
```

---

plot_slopes *Plot derived slopes from a model fitted by* time_model().

Description

Plot derived slopes from a model fitted by time_model().

Usage

```
plot_slopes(
  fit,
  method,
  period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17),
 knots = list(cubic_slope = NULL, linear_splines = c(0.75, 5.5, 11), cubic_splines =
    c(1, 8, 12))[[method]]
)
```

Arguments

fit	A model object from a statistical model such as from a call to time_model().
method	The type of model provided in fit, *i.e.*, one of "cubic_slope", "linear_splines" or "cubic_splines".
period	The intervals knots on which AUCs are to be computed.
knots	The knots as defined fit and according to method.

Value

A patchwork ggplot2 object.

Examples

```
library(ggplot2)
library(eggla)
data("bmigrowth")
ls_mod <- time_model(
  x = "age",
  y = "log(bmi)",
  cov = NULL,
  data = bmigrowth[bmigrowth[["sex"]] == 0, ],
  method = "linear_splines"
)
plot_slopes(
  fit = ls_mod,
  method = "linear_splines"
)
```

predict_bmi *Predict BMI for a range of ages from a model fit.*

Description

Predict BMI values a cubic splines mixed model regression with three splines parametrisation as random effect. This function also works for any model obtained using `time_model()`.

Usage

```
predict_bmi(fit, start = 0.25, end = 10, step = 0.01, filter = NULL)
```

Arguments

`fit`	A model object from a statistical model such as from a call `nlme::lme()`, `time_model()` or `egg_model()`.
`start`	The start of the time window to compute AP and AR.
`end`	The end of the time window to compute AP and AR.
`step`	The step to increment the sequence.
`filter`	A string following `data.table` syntax for filtering on `"i"` (*i.e.*, row elements), *e.g.*, `filter = "source == 'A'"`. Argument pass through `compute_apar()` (see `predict_bmi()`). Default is `NULL`.

Value

A `data.table` object.

Examples

```
data("bmigrowth")
res <- egg_model(
  formula = log(bmi) ~ age,
  data = bmigrowth[bmigrowth[["sex"]] == 0, ],
  id_var = "ID",
  random_complexity = 1
)

predict_bmi(res)[]

## For multiple sources of measures or multiple measures at one age
set.seed(1234)
dta <- bmigrowth[bmigrowth[["sex"]] == 0, ]
dta[["source"]] <- c("A", "B")[rbinom(n = nrow(dta), size = 1, prob = 0.65) + 1]

res <- egg_model(
  formula = log(bmi) ~ age + source,
  data = dta,
  id_var = "ID",
```

```
  random_complexity = 1
)

predict_bmi(res)[order(egg_id, egg_ageyears)]

predict_bmi(res, filter = "source == 'A'")[order(egg_id, egg_ageyears)]
```

`run_eggla_gwas` *Perform GWAS using PLINK2 (and BCFtools)*

Description

Format VCF file(s) by filtering out all variants not satisfaying "–min-alleles 2 –max-alleles 2 –types snps" and setting IDs (if no annotation file using VEP is provided) with "%CHROM:%POS:%REF:%ALT" (see https://samtools.github.io/bcftools/). GWAS is performed on the formatted VCF file(s) by PLINK2 software (https://www.cog-genomics.org/plink/2.0).

Usage

```
run_eggla_gwas(
  data,
  results,
  id_column,
  traits = c("slope_.*", "auc_.*", "^AP_.*", "^AR_.*"),
  covariates,
  vcfs,
  working_directory,
  vep_file = NULL,
  use_info = TRUE,
  bin_path = list(bcftools = "/usr/bin/bcftools", plink2 = "/usr/bin/plink2"),
  bcftools_view_options = NULL,
  build = "38",
  strand = "+",
  info_type = "IMPUTE2 info score via 'bcftools +impute-info'",
  threads = 1,
  quiet = FALSE,
  clean = TRUE
)
```

Arguments

`data`	Path to the phenotypes stored as a CSV file.
`results`	Paths to the zip archives or directories generated by `run_eggla_lmm()` (vector of length two, one male and one female path).
`id_column`	Name of the column where sample/individual IDs are stored.
`traits`	One or multiple traits, *i.e.*, columns' names from `data`, to be analysed separately.

`covariates` One or several covariates, *i.e.*, columns' names from `data`, to be used. Binary trait should be coded as '1' and '2', where sex must be coded: '1' = male, '2' = female, 'NA'/'0' = missing.

`vcfs` Path to the "raw" VCF file(s) containing the genotypes of the individuals to be analysed.

`working_directory` Directory in which computation will occur and where output files will be saved.

`vep_file` Path to the VEP annotation file to be used to set variants RSIDs and add gene SYMBOL, etc.

`use_info` A logical indicating whether to extract all informations stored in the "INFO" field.

`bin_path` A named list containing the path to the PLINK2 and BCFtools binaries For PLINK2, an URL to the binary can be provided (see https://www.cog-genomics.org/plink/2.0).

`bcftools_view_options` A string or a vector of strings (which will be pass to `paste()`) containing BCFtools view parameters, *e.g.*, `"--min-af 0.05"`, `"--exclude 'INFO/INFO < 0.8'"`, and/or `"--min-alleles 2 --max-alleles 2 --types snps"`.

`build` Build of the genome on which the SNP is orientated. Default is "38".

`strand` Orientation of the site to the human genome strand used. Should be "+" (default).

`info_type` Type of information provided in the INFO column, *e.g.*, "IMPUTE2 info score via 'bcftools +impute-info'",

`threads` Number of threads to be used by some BCFtools and PLINK2 commands.

`quiet` A logical indicating whether to suppress the output.

`clean` A logical indicating whether to clean intermediary files or not.

Value

Path to results file.

Examples

```
if (interactive()) {
  data("bmigrowth")
  bmigrowth_csv <- file.path(tempdir(), "bmigrowth.csv")
  fwrite(
    x = bmigrowth,
    file = bmigrowth_csv
  )
  results_archives <- run_eggla_lmm(
    data = fread(
      file = file.path(tempdir(), "bmigrowth.csv"),
      colClasses = list(character = "ID")
    ),
    id_variable = "ID",
    age_days_variable = NULL,
    age_years_variable = "age",
```

```
    weight_kilograms_variable = "weight",
    height_centimetres_variable = "height",
    sex_variable = "sex",
    covariates = NULL,
    male_coded_zero = FALSE,
    random_complexity = 1,
    parallel = FALSE,
    parallel_n_chunks = 1,
    working_directory = tempdir()
  )
  run_eggla_gwas(
    data = fread(
      file = file.path(tempdir(), "bmigrowth.csv"),
      colClasses = list(character = "ID")
    ),
    results = results_archives,
    id_column = "ID",
    traits = c("slope_.*", "auc_.*", "^AP_.*", "^AR_.*"),
    covariates = c("sex"),
    vcfs = list.files(
      path = system.file("vcf", package = "eggla"),
      pattern = "\\.vcf$|\\.vcf.gz$",
      full.names = TRUE
    ),
    working_directory = tempdir(),
    vep_file = NULL,
    bin_path = list(
      bcftools = "/usr/bin/bcftools",
      plink2 = "/usr/bin/plink2"
    ),
    threads = 1
  )
}
```

`run_eggla_lmm` *Perform EGG longitudinal analysis and derived areas under the curves and slopes.*

Description

Perform Daymont's quality-control for BMI, fit a cubic splines mixed model regression with linear splines as random effect, save model object, generates residuals figures fot model validity, derived area under the curve and slopes for male and femal. This function is a wrapper around `egg_model()`, `egg_slopes()` and `egg_aucs()`.

Usage

```
run_eggla_lmm(
  data,
  id_variable,
```

```
  age_days_variable,
  age_years_variable,
  weight_kilograms_variable,
  height_centimetres_variable,
  sex_variable,
  covariates,
  male_coded_zero = FALSE,
  random_complexity = "auto",
  use_car1 = FALSE,
  knots = c(1, 8, 12),
  period = c(0, 0.5, 1.5, 3.5, 6.5, 10, 12, 17),
  start = 0.25,
  end = 10,
  step = 0.01,
  filter = NULL,
  outlier_method = "iqr",
  outlier_threshold = list(iqr = 2),
  outlier_exclude = TRUE,
  parallel = FALSE,
  parallel_n_chunks = 1,
  working_directory = getwd(),
  quiet = FALSE,
  clean = TRUE
)
```

Arguments

`data`	Phenotypes data that inherits from `data.frame` class.
`id_variable`	Name of the column where sample/individual IDs are stored.
`age_days_variable`	Name of the column where age in days is stored. `NULL` if age in days is not available.
`age_years_variable`	Name of the column where age in years is stored. `NULL` if age in years is not available.
`weight_kilograms_variable`	Name of the column where weight in kilograms is stored.
`height_centimetres_variable`	Name of the column where height in centimetres is stored.
`sex_variable`	Name of the column where sex is stored.
`covariates`	A vector of columns' names to be used as covariates. `NULL` if there are no covariates to add.
`male_coded_zero`	Is male coded "0" (and female coded "1")?
`random_complexity`	A numeric (1-3) indicating the complexity of the random effect term. Default, `"auto"` will try from the more complex to the less complex if no success.

use_car1 A logical indicating whether to use continuous auto-correlation, i.e., CAR(1) as correlation structure.

knots The knots defining the splines.

period The intervals knots on which slopes are to be computed.

start The start of the time window to compute AP and AR.

end The end of the time window to compute AP and AR.

step The step to increment the sequence.

filter A string following `data.table` syntax for filtering on `"i"` (*i.e.*, row elements), *e.g.*, `filter = "source == 'A'"`. Argument pass through `compute_apar()` (see `predict_bmi()`). Default is `NULL`.

outlier_method The outlier detection method(s). Default is `"iqr"`. Can be `"cook"`, `"pareto"`, `"zscore"`, `"zscore_robust"`, `"iqr"`, `"ci"`, `"eti"`, `"hdi"`, `"bci"`, `"mahalanobis"`, `"mahalanobis_robust"`, `"mcd"`, `"ics"`, `"optics"` or `"lof"`. See `performance::check_outliers()` https://easystats.github.io/performance/reference/check_outliers.html for details.

outlier_threshold A list containing the threshold values for each method (*e.g.*, `list('mahalanobis' = 7, 'cook' = 1)`), above which an observation is considered as outlier. If `NULL`, default values will be used (see 'Details'). If a numeric value is given, it will be used as the threshold for any of the method run. See `performance::check_outliers()` https://easystats.github.io/performance/reference/check_outliers.html for details.

outlier_exclude Whether or not the values/individuals flagged as being outliers should be excluded. Default is `TRUE`.

parallel Determines if `growthcleanr::cleangrowth()` function shoud be run in parallel. Defaults to `FALSE`.

parallel_n_chunks Specify the number of batches (in `growthcleanr::cleangrowth()`) to run in parallel. Only applies if parallel is set to TRUE. Defaults to the number of workers returned by the getDoParWorkers function in the foreach package.

working_directory Directory in which computation will occur and where output files will be saved.

quiet A logical indicating whether to suppress the output.

clean A logical indicating whether to clean `working_directory` once the archives are created.

Value

Path to zip archives.

Examples

```
if (interactive()) {
  data("bmigrowth")
  fwrite(
```

```
  x = bmigrowth,
  file = file.path(tempdir(), "bmigrowth.csv")
 )
 res <- run_eggla_lmm(
  data = fread(file.path(tempdir(), "bmigrowth.csv")),
  id_variable = "ID",
  age_days_variable = NULL,
  age_years_variable = "age",
  weight_kilograms_variable = "weight",
  height_centimetres_variable = "height",
  sex_variable = "sex",
  covariates = NULL,
  random_complexity = 1,
  working_directory = tempdir()
 )
}
```

`time_model` *Fit one of three mixed model.*

Description

Fit a mixed model regression with "cubic slope", "linear splines" or "cubic splines" as fixed and random effects.

Usage

```
time_model(
  x,
  y,
  cov = NULL,
  data,
  method = c("cubic_slope", "linear_splines", "cubic_splines"),
 knots = list(cubic_slope = NULL, linear_splines = c(0.75, 5.5, 11), cubic_splines =
    c(1, 8, 12))[[method]],
  use_car1 = FALSE,
  id_var = "ID",
  quiet = FALSE
)
```

Arguments

`x`	A length one character vector with the main covariate name (*i.e.*, right-hand side).
`y`	A length one character vector with the variable name to be explained (*i.e.*, left-hand side).
`cov`	A vector of addtional/optional covariates names to included in the fixed effect part of the linear mixed-effects models.

data	A data.frame containing the variables named in x and y.
method	The type of model, *i.e.*, one of "cubic_slope", "linear_splines" or "cubic_splines".
knots	The knots defining the splines for "linear_splines" and "cubic_splines" methods.
use_car1	A logical indicating whether to use continuous auto-correlation, i.e., CAR(1) as correlation structure.
id_var	A string indicating the name of the variable to be used as the individual identifier.
quiet	A logical indicating whether to suppress the output.

Value

An object of class "lme" representing the linear mixed-effects model fit.

Examples

```
data("bmigrowth")
ls_mod <- time_model(
  x = "age",
  y = "log(bmi)",
  cov = NULL,
  data = bmigrowth[bmigrowth[["sex"]] == 0, ],
  method = "linear_splines"
)
sres <- as.data.frame(summary(ls_mod)[["tTable"]])
rownames(sres) <- sub("gsp\\(.*\\)\\)", "gsp(...)", rownames(sres))
sres
```

Index