Package ‘splinetrials’

December 18, 2025

Type Package

Title Facilitate Clinical Trials Analysis Using Natural Cubic Splines

Version 0.1.0

Description Create mixed models with repeated measures using natural cubic splines applied to an observed continuous time variable, as described by Donohue et al. (2023) <doi:10.1002/pst.2285>. Iterate through multiple covariance structure types until one converges. Categorize observed time according to scheduled visits. Perform subgroup analyses.

License Apache License (>= 2)

URL `https://github.com/NikKrieger/splinetrials`, `https://nikkrieger.github.io/splinetrials/`

BugReports `https://github.com/NikKrieger/splinetrials/issues`

Imports car, cli, dplyr, emmeans, mmrm (>= 0.3.16), rlang, splines

Suggests ggplot2, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Nik Krieger [aut, cre],
Daniel Sabanes Bove [ctb],
Eli Lilly and Company [cph]

Maintainer Nik Krieger <`nikkrieger@gmail.com`>

Repository CRAN

Date/Publication 2025-12-18 14:20:02 UTC

Contents

`bin_timepoints` *Categorize Observed Timepoints According to Scheduled Timepoints*

Description

Create an ordered factor from a vector of observed values, associating each observed value with the level corresponding to a vector of expected/`scheduled` values.

Usage

```
bin_timepoints(
  observed,
  scheduled = unique(observed[!is.na(observed)]),
  breaks = c(-Inf, midpoints(scheduled), Inf),
  labels = make_visit_labels(seq_along(scheduled) - 1),
  ...
)
```

Arguments

`observed`	A numeric vector of values.
`scheduled`	A numeric vector of unique, finite values. Length must be at least 2. The default is to take the unique, finite values of `observed`.
`breaks`	A numeric vector of unique values. `-Inf` and `Inf` are valid. Passed to `cut()`. The default is to take the midpoints of `scheduled` and to put them in between `c(-Inf, [Inf])`.

`labels`	A vector of labels for the resulting `ordered` factor. Passed to `cut()`. Must have `length()` equal to `scheduled`. Defaults to `"Baseline"` as the first level's label and `"Visit#"` for all subsequent levels, where # is the numeric index of the timepoint minus 1.
`...`	Additional arguments passed to `cut()`.

Value

And `[ordered] factor` with the same length as `observed`.

Examples

```
observed_timepoints <- c(0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89)
scheduled_timepoints <- c(0, 1, 2, 3, 4, 5, 10, 15, 20, 30, 50, 75)
bin_timepoints(
  observed_timepoints,
  scheduled = scheduled_timepoints
)

bin_timepoints(
  observed_timepoints,
  scheduled = scheduled_timepoints,
  breaks = c(-Inf, 0.1, 1.5, 2.5, 3.5, 4.4, 7, 11, 15.1, 21, 31, 58, 80)
)

bin_timepoints(
  observed_timepoints,
  scheduled = scheduled_timepoints,
  labels = month.name
)

bin_timepoints(
  observed_timepoints,
  scheduled = scheduled_timepoints,
  labels = make_visit_labels(scheduled_timepoints, visit = "Week")
)

bin_timepoints(observed_timepoints)
```

`change_from_baseline` *Calculate the Change from Baseline or Treatment Effects from Estimated Marginal Means*

Description

Pass `emmeans::emmeans()` objects (probably obtained via `ncs_emmeans()`) to `emmeans::contrast()` using specially constructed contrast matrices so that change from baseline and treatment effects can be calculated.

- `change_from_baseline` calculate the change from baseline for each of the different study arms/subgroups.
- `treatment_effect()` calculate the treatment effect for each study arm when there is no subgroup. When there is a subgroup, calculate the treatment effect *between* subgroups (examining the differences *between* the subgroups within each study arm) or *within* subgroups (examining the differences between the study arms *within* each subgroup).

Usage

```
change_from_baseline(
  emmeans,
  time_observed_continuous = emmeans@roles$predictors[2],
  time_scheduled_baseline = 0,
  arm = emmeans@roles$predictors[1],
 subgroup = if (length(emmeans@roles$predictors) == 3) emmeans@roles$predictors[3],
  contrast_args = list(adjust = "none"),
  ...,
  as_tibble = FALSE,
  confint_args = list(level = 0.95)
)

treatment_effect(
  emmeans,
  time_observed_continuous = emmeans@roles$predictors[2],
  time_scheduled_baseline,
  arm = emmeans@roles$predictors[1],
 subgroup = if (length(emmeans@roles$predictors) == 3) emmeans@roles$predictors[3],
  ref_value,
  subgroup_type = c("between", "within"),
  contrast_args = list(adjust = "none"),
  ...,
  as_tibble = FALSE,
  confint_args = list(level = 0.95)
)
```

Arguments

`emmeans`	(`emmGrid`) an object of class `emmGrid`, ideally obtained via `ncs_emmeans()`, which wraps `emmeans::emmeans()`.
`time_scheduled_baseline`	(`scalar numeric`) the continuous time point when baseline was *scheduled* to occur. Defaults to 0.
`arm`, `time_observed_continuous`, `subgroup`	(`string`) strings identifying the study arm variable, *observed* continuous time variable, and (optionally) subgroup variable supplied to `emmeans::emmeans()`, probably via `ncs_emmeans()`. If `ncs_emmeans()` was indeed used, these strings *should*

be contained in the character vector `emmeans@roles$predictors` (see the default arguments).

`contrast_args, ...`
: (named `list`)
arguments to be passed to `emmeans::contrast()`. Any arguments with the names `object` or `method` will be overwritten. Arguments in `contrast_args` override identically named arguments in `...`.

`as_tibble`
: (`flag`)
`TRUE` or `FALSE` indicating whether or not the results of `emmeans::contrast()` should be processed and returned as a tibble.

`confint_args`
: (named `list`)
arguments to be passed to `stats::confint()` when calculating confidence intervals. Ignored if `as_tibble = FALSE`. If `NULL`, confidence intervals will not be calculated. Defaults to `list(level = 0.95)`.

`ref_value`
: (`string`)
the value in `arm` (if `subgroup = NULL` *or* if `subgroup_type = "within"`) or the value in `subgroup` (if `subgroup` is not `NULL` *and* `subgroup_type = "between"`) denoting the control group.

`subgroup_type`
: (`string`)
either `"between"` or `"within"`, denoting whether to calculate the treatment effect *between* subgroups (examining the differences between the subgroups within each study arm) and once *within* subgroups (examining the differences between the study arms within each subgroup).

Value

When `as_tibble = FALSE`, the value returned by `emmeans::contrast()`. If `as_tibble = TRUE`, a tibble:

1. {*column name will be the value of the* `arm` *argument*}: the study arm.
2. {*column name will be the value of the* `time_observed_continuous` *argument*}: the *observed* continuous time variable.
3. {*column name will be the value of the* `subgroup` *argument*}: the subgroup. **Only present if** `subgroup` **is not** `NULL`**.**
4. `estimate`: estimate for change from baseline or treatment effect.
5. `SE`: standard error of `estimate`.
6. `df`: degrees of freedom for calculating the confidence interval for and estimating the significance of `estimate`.
7. `lower.CL`: lower bound of confidence interval for `estimate`. **Only present if** `confint_args` **is not** `NULL`**.**
8. `upper.CL`: upper bound of confidence interval for `estimate`. **Only present if** `confint_args` **is not** `NULL`**.**
9. `t.ratio`: test statistic measuring the significance of `estimate`.
10. `p.value`: p-value for the significance of `estimate`.

Examples

```
# Create a usable data set out of mmrm::fev_data
fev_mod <- mmrm::fev_data
fev_mod$VISITN <- fev_mod$VISITN * 10
fev_mod$time_cont <- fev_mod$VISITN + rnorm(nrow(fev_mod))
fev_mod$obs_visit_index <- round(fev_mod$time_cont)

fit <-
  ncs_mmrm_fit(
    data = fev_mod,
    type = "subgroup_full",
    response = FEV1,
    subject = USUBJID,
    cov_structs = c("ar1", "us"),
    time_observed_continuous = time_cont,
    df = 2,
    time_observed_index = obs_visit_index,
    time_scheduled_continuous = VISITN,
    arm = ARMCD,
    control_group = "PBO",
    subgroup = SEX,
    subgroup_comparator = "Male",
    covariates = ~ FEV1_BL + RACE
  )

marginal_means <-
  ncs_emmeans(
    fit = fit,
    observed_time = "time_cont",
    scheduled_time = "VISITN",
    arm = "ARMCD",
    subgroup = "SEX"
  )

change_from_baseline(
  emmeans = marginal_means,
  time_observed_continuous = "time_cont",
  time_scheduled_baseline = 10,
  arm = "ARMCD",
  subgroup = "SEX"
)

# Same thing as a tibble:
change_from_baseline(
  emmeans = marginal_means,
  time_observed_continuous = "time_cont",
  time_scheduled_baseline = 10,
  arm = "ARMCD",
  subgroup = "SEX",
  as_tibble = TRUE
)
```

```
treatment_effect(
  emmeans = marginal_means,
  time_observed_continuous = "time_cont",
  time_scheduled_baseline = 10,
  arm = "ARMCD",
  subgroup = "SEX",
  ref_value = "Male",
  as_tibble = TRUE
)
```

`make_visit_labels` *Make Visit Labels Based on a Numeric Vector*

Description

Create a character vector of values to be used as labels for a factor.

Usage

```
make_visit_labels(t, visit = "VIS", baseline = "BASELINE", pad = "0")
```

Arguments

`t`	A non-empty numeric vector of unique, finite elements in ascending order.
`visit`	A single character string specifying the prefix to add to `t`.
`baseline`	A single character string to use for the first timepoint's label. Alternatively, set to `NULL` so that all timepoints will have the prefix specified by `visit`.
`pad`	The character to use to pad between `visit` and `t` so that the places of `t` are aligned. Alternatively, set to `NULL` so that `t` is automatically converted to `character` without special formatting. This can result in numbers in labels not being aligned or not being in "alphabetical" order.

Details

Places `visit` as a prefix before the values of `t`. If pad is not `NULL`, the values of `t` are first `formatted` so that their places are aligned, and they are left-padded with zeros.

If `baseline` is not `NULL` it is used as the first label regardless of the value of `t[1]`.

Uses `make.unique(sep = "_")` in case any elements are identical after formatting.

Value

A character vector of length `length(t)`.

Examples

```
make_visit_labels(c(0, 5, 13, 101))

make_visit_labels(c(0, 5.23453, 13, 101.4))

make_visit_labels(c(0, 5.23453, 13, 101.4), baseline = NULL, pad = " ")

make_visit_labels(c(0, 5.23453, 13, 101.4), visit = "Week", pad = NULL)
```

`midpoints` *Midpoints of a Numeric Vector*

Description

Returns the midpoints between the elements of a vector in the order the elements appear.

Usage

```
midpoints(x)
```

Arguments

`x`	A numeric vector with at least 2 elements.

Details

This function does not sort.

Value

A numeric vector of length `length(x) - 1`.

Examples

```
midpoints(c(0, 1, 10, 4))
```

`ncs_analysis` *Run a Natural Cubic Spline (NCS) Analysis.*

Description

Fit and analyze an `mmrm` model wherein the continuous time variable has splines applied.

- `ncs_analysis()` fits such a model without involving subgroups.
- `ncs_analysis_subgroup()` fits a model that involves subgroups and performs additional analyses.

Usage

```
ncs_analysis(
  data,
  response = "response",
  subject = "subject",
  arm = "arm",
  control_group,
  time_observed_continuous = "time_observed_continuous",
  df = 2,
  spline_basis = NULL,
  time_observed_index = "time_observed_index",
  time_scheduled_continuous = "time_scheduled_continuous",
  time_scheduled_baseline = 0,
  time_scheduled_label = "time_scheduled_label",
  covariates = ~1,
  cov_structs = c("us", "toeph", "ar1h", "csh", "cs"),
  cov_struct_group = NULL,
  mmrm_args = list(method = "Satterthwaite"),
  emmeans_args = list(nesting = NULL),
  average_nuisance = TRUE,
  conf.level = 0.95,
  change_in_bl_contrast_args = list(adjust = "none"),
  treatment_effect_contrast_args = list(adjust = "none"),
  confint_args = list(level = conf.level),
  return_models = FALSE,
  expand_spline_terms = TRUE
)

ncs_analysis_subgroup(
  data,
  response = "response",
  subject = "subject",
  arm = "arm",
  control_group,
  subgroup = "subgroup",
  subgroup_comparator = "subgroup1",
  time_observed_continuous = "time_observed_continuous",
  df = 2,
  spline_basis = NULL,
  time_observed_index = "time_observed_index",
  time_scheduled_continuous = "time_scheduled_continuous",
  time_scheduled_baseline = 0,
  time_scheduled_label = "time_scheduled_label",
  covariates = ~1,
  cov_structs = c("us", "toeph", "ar1h", "csh", "cs"),
  cov_struct_group = NULL,
  mmrm_args = list(method = "Satterthwaite"),
  emmeans_args = list(nesting = NULL),
  average_nuisance = TRUE,
  conf.level = 0.95,
  change_in_bl_contrast_args = list(adjust = "none"),
  treatment_effect_contrast_args = list(adjust = "none"),
  confint_args = list(level = conf.level),
  subgroup_interaction_test = TRUE,
  return_models = FALSE,
  expand_spline_terms = TRUE
)
```

Arguments

`data`	(`data frame`) data set supplied to the `data` argument of `mmrm::mmrm()` when fitting models. The supplied expression is quoted and must evaluate to a data frame. See **Tidy evaluation support**.
`response`	(`numeric` or `string`) the response variable. It can be a `string` identifying the name of an existing variable; otherwise, the supplied expression will be quoted and added to the formula as is (see **Tidy evaluation support**).
`subject`	(`atomic` or `string`) the unique subject identifier forwarded to the `subject` argument of `mmrm::cov_struct()`. Ignored if `cov_structs` is a `list`. Can be a `string` identifying an existing variable; otherwise the supplied expression will be quoted and turned into a `string` with `rlang::expr_deparse()` (see **Tidy evaluation support**).
`arm`	(`factor` or `string`) the study arm. It must be a `string` or a name identifying an existing variable (i.e., it cannot be a call). If a name, it will be quoted before being added to the model formula (see **Tidy evaluation support**). If it does not evaluate to a factor or if `control_group` is not its first level, the `data` argument will be wrapped in a `dplyr::mutate()` call that forces this to be the case.
`control_group`	(`string`) the value in `arm` denoting the control group. If necessary, `arm` will be preprocessed such that it is a factor with `control_group` as its first level.

`time_observed_continuous`	(`numeric` or `string`) the visit's *observed* time point. It must either be a `string` or a name identifying an existing variable (i.e., it cannot be a call). If a name is provided, it is quoted and incorporated into the model formula as is (see **Tidy evaluation support**).
`df`	(scalar `integer`) number of degrees of freedom to use to create the spline basis. Passed to the `df` argument of `time_spline_basis()`. Ignored if the `spline_basis` argument is not `NULL`.
`spline_basis`	(`basis matrix`) a spline basis: probably a value returned by `time_spline_basis()` (which wraps `splines::ns()`). If `NULL` (the default), then the spline basis will be the result of forwarding `time_observed_continuous` and `df` to `time_spline_basis()`. See **Providing a spline basis**.
`time_observed_index`	(`ordered` or `string`) the visit index that the visit shall be associated with, based on the visit's *observed* time point. This will be passed as the `visits` argument of `mmrm::cov_struct()`. It can be a `string` identifying an existing variable; otherwise the supplied expression will be quoted and turned into a `string` with `rlang::expr_deparse()` (see **Tidy evaluation support**). If it does not evaluate to an `ordered` factor, it will be wrapped with `as.ordered()`. Ignored if `cov_structs` is a `list`.
`time_scheduled_continuous`	(`numeric` or `string`) the continuous time point when the visit was *scheduled* to occur. Its unique values will identify the time points at which the marginal means and other results will be calculated. It can be a `string` identifying an existing variable name; otherwise the supplied expression will be quoted before being evaluated (see **Tidy evaluation support**).
`time_scheduled_baseline`	(`scalar numeric`) the continuous time point when baseline was *scheduled* to occur. Defaults to 0.
`time_scheduled_label`	(`character` or `string`) the label associated with the scheduled visit. It can be a `string` identifying an existing variable name; otherwise the supplied expression will be quoted before being evaluated (see **Tidy evaluation support**).
`covariates`	(`formula`) formula containing additional terms that should be added to the `mmrm` model. Defaults to `~ 1`, in which no additional terms will be added. Must not have a left side. Cannot contain `.`. To specify that the model shall not have an intercept, use include `+ 0` or `- 1` in this formula.
`cov_structs`	(`character` or `list`) either a `list` of unique `cov_struct` objects or a `character` vector of one or more of the covariance structure abbreviations as described in `mmrm::cov_types()`. These covariance structures will be attempted in order until one of them achieves a converging model fit. Defaults to `c("us", "toeph", "ar1h", "csh", "cs")`.

`cov_struct_group`
: (`atomic` or `string`)
optional grouping variable to be passed to the `group` argument of `mmrm::cov_struct()`. It can be a `string` identifying an existing variable name; otherwise the supplied expression will be quoted and turned into a `string` with `rlang::expr_deparse()` (see **Tidy evaluation support**). Ignored if `cov_structs` is a `list`. Defaults to `NULL`, in which case no grouping variable will be used.

`mmrm_args`
: (named `list`)
arguments to be passed to `mmrm::mmrm()`. If any elements have the names `formula`, `data`, or `covariance` they will be ignored. An element named `vcov` will also be ignored unless fitting a model with an unstructured covariance. Defaults to `list(method = "Satterthwaite")`.

`emmeans_args`
: (named `list`)
arguments to be passed to `emmeans::emmeans()`. If any elements have the names `object` `specs`, or `at` they will be ignored. If `average_nuisance = TRUE`, any element named `nuisance` will be ignored. Any elements named `params` may be ignored. Defaults to `list(nesting = NULL)`.

`average_nuisance`
: (`flag`)
flag indicating whether the names of the terms in `covariates` should be supplied as the `nuisance` argument to `emmeans::emmeans()`. This results in treating all the covariates as nuisance parameters and averaging over them when calculating the reference grid to estimate marginal means. See `emmeans::ref_grid()` for details and limitations.

`conf.level`
: (scalar `numeric`)
confidence level for the calculation of p-values. Defaults to `0.95`.

`change_in_bl_contrast_args, treatment_effect_contrast_args`
: (named `list`)
arguments to be passed to `emmeans::contrast()` when calculating the change from baseline and treatment effect results. If any elements have the names `object` or `method` they will be ignored. Defaults to `list(adjust = "none")`.

`confint_args`
: (named `list`)
arguments to be passed to `stats::confint()` when calculating confidence intervals for change in baseline and treatment effect. If any element has the name `object` it will be ignored. Defaults to `list(level = conf.level)`.

`return_models`
: (`flag`)
flag indicating whether or not to return the model(s) used to calculate the results. See **Obtaining the models used** below.

`expand_spline_terms`
: (`flag`)
flag indicating whether or not to separate the cubic spline matrix into separate terms (one for each degree of freedom). Defaults to `TRUE`. See **Expanding spline terms**.

`subgroup`
: (`factor` or `string`)
the subgroup. It must be a `string` or a name identifying an existing variable (i.e., it cannot be a call). If a name, it will be quoted before being added to

the model formula (see **Tidy evaluation support**). If it does not evaluate to a factor or if subgroup_comparator is not its first level, the data argument will be wrapped in a dplyr::mutate() call that forces this to be the case.

subgroup_comparator
: (string)
the value in subgroup denoting the "main" subgroup that all other subgroups should be compared to. If necessary, subgroup will be preprocessed such that it is a factor with control_group as its first level.

subgroup_interaction_test
: (flag)
flag indicating whether or not the subgroup interaction test should be performed. If TRUE, the returned value will include an interaction element, a data frame of results. Defaults to TRUE. See **Subgroup interaction test** for details.

Value

For ncs_analysis(), see splinetrials_analysis. For ncs_analysis_subgroup(), see splinetrials_subgroup_anal

Overview

These functions create an mmrm model from the user-specified arguments. They then perform a series of analyses and produce a data frame of results with a unique row for each combination of arm, time_scheduled_continuous, and subgroup (for ncs_analysis_subgroup() only). The results include:

1. Basic diagnostics on the response variable
2. Estimated marginal means
3. Change from baseline
4. Treatment effect
5. Percent slowing

Building a model

See the details of ncs_mmrm_fit() for information on how the model is built.

Subgroup analysis

ncs_analysis_subgroup() contains more analyses and results than ncs_analysis(). Whereas the latter produces a data frame by default, the former produces a list of data frames.

Treatment effects:

The treatment effect is calculated twice: once *between* subgroups (examining the differences between the subgroups within each study arm) and once *within* subgroups (examining the differences between the study arms within each subgroup). The main results table is effectively returned twice as both the between element and the within element. These elements' treatment effect values differ, and only the within element contains the percent slowing analysis results.

Type-III ANOVA:

The subgroup analyses include a type-III analysis of variance (ANOVA) on the main analysis model's terms, using a Chi-squared test statistic. This is accomplished via the `mmrm` method for `car::Anova()`. The results are included in the returned value as the `type3` element. See `vignette("hypothesis_testing", "mmrm")` for details on the type-III ANOVA.

Subgroup interaction test:

When `subgroup_interaction_test = TRUE`, the function runs an ANOVA to compare a maximum-likelihood-estimated (ML) version of the original model to a reduced version. This happens as follows:

1. The original analysis model is refit with `reml = FALSE` if it was originally created with `reml = TRUE`. This may be dubbed the "full" model.
2. A reduced version of the "full" model is created, removing the second-order interaction term (see the `arm` **and** `subgroup` **terms** section above). This may be dubbed the "reduced" model.
3. The "full" and "reduced" models are compared using the `mmrm` method of `stats::anova()`.
4. The results are processed into a table and added to the returned value as the `interaction` element.

Returning the models used

The model(s) used to conduct the analyses can be obtained by setting `return_models = TRUE`.

For `ncs_analysis()`, the analysis model will be included as the `splinetrials_analysis_model` attribute of the returned value.

For `ncs_analysis_subgroup()`, the analysis model is added to the returned value as the `analysis_model` element. Furthermore, if `subgroup_interaction_test = TRUE`, the "full" and "reduced" models will be included in the returned value as the elements `full` and `reduced` (see **Subgroup interaction test** above for details).

Examples

```
# Create a usable data set out of mmrm::fev_data
fev_mod <- mmrm::fev_data
fev_mod$VISITN <- fev_mod$VISITN * 10
fev_mod$time_cont <- fev_mod$VISITN + rnorm(nrow(fev_mod))
fev_mod$obs_visit_index <- round(fev_mod$time_cont)

# Without subgroup:
ncs_analysis(
  data = fev_mod,
  response = FEV1,
  subject = USUBJID,
  arm = ARMCD,
  control_group = "PBO",
  time_observed_continuous = time_cont,
  df = 2,
  time_observed_index = obs_visit_index,
  time_scheduled_continuous = VISITN,
  time_scheduled_baseline = 10,
  time_scheduled_label = AVISIT,
  covariates = ~ FEV1_BL + RACE,
  cov_structs = c("ar1", "us")
)

# With subgroup:
ncs_analysis_subgroup(
  data = fev_mod,
  response = FEV1,
  subject = USUBJID,
  arm = ARMCD,
  control_group = "PBO",
  subgroup = SEX,
  subgroup_comparator = "Male",
  time_observed_continuous = time_cont,
  df = 2,
  time_observed_index = obs_visit_index,
  time_scheduled_continuous = VISITN,
  time_scheduled_baseline = 10,
  time_scheduled_label = AVISIT,
  covariates = ~ FEV1_BL + RACE,
  cov_structs = c("ar1", "us")
)
```

ncs_emmeans *Estimate Marginal Means for a Natural Cubic Splines Analysis*

Description

This is wrapper around `emmeans::emmeans()` for a natural cubic splines analysis in which there is a continuous time variable, a study arm, and (optionally) a subgroup variable.

Usage

```
ncs_emmeans(
  fit,
  data = fit[["data"]],
  observed_time = NULL,
  scheduled_time = NULL,
  arm = NULL,
  subgroup = NULL,
  average_nuisance = TRUE,
  emmeans_args = list(nesting = NULL),
  ...,
  scheduled_time_spec = sort(unique(data[[scheduled_time]])),
  arm_spec = as.character(sort(unique(data[[arm]]))),
  subgroup_spec = as.character(sort(unique(data[[subgroup]]))),
  .__caller_env = rlang::caller_env()
)
```

Arguments

`fit`	(`mmrm`) an `mmrm` object whose terms include the variables supplied to `observed_time`, `scheduled_time`, `arm`, and (optionally) `subgroup`.
`data`	(`data frame`) a data frame on which to estimate marginal means. Defaults to `fit[["data"]]`.
`observed_time`	(`string`) string specifying the *observed* continuous time variable in both `fit` and in `data`.
`scheduled_time`	(`string`) string specifying the *scheduled* continuous time variable in both `fit` and in `data`. Ignored if `scheduled_time_spec` is provided.
`arm`	(`string`) string specifying the study arm variable in both `fit` and in `data`.
`subgroup`	(`string`) string specifying the subgroup variable in both `fit` and in `data`.
`average_nuisance`	(`flag`) flag indicating whether the names of the terms in `covariates` should be supplied as the `nuisance` argument to `emmeans::emmeans()`. This results in treating all the covariates as nuisance parameters and averaging over them when calculating the reference grid to estimate marginal means. See `emmeans::ref_grid()` for details and limitations.
`emmeans_args`, `...`	(named `list`) arguments to be passed to `emmeans::emmeans()`. If any elements have the names `object`, `specs`, or `at` they will be ignored. If `average_nuisance = TRUE`, any element named `nuisance` will be ignored. Any elements named `params` may be ignored. `emmeans_args` defaults to `list(nesting = NULL)`. Arguments named in `emmeans_args` supersede any named arguments in `...`.
`scheduled_time_spec`	(`numeric`) vector of unique, non-missing time points on which to calculate marginal means. Defaults to `sort(unique(data[[scheduled_time]]))`.
`arm_spec`	(`character`) vector of unique study arm values on which to calculate marginal means. Defaults to `as.character(sort(unique(data[[arm]])))`.
`subgroup_spec`	vector of unique subgroup values on which to calculate marginal means. Ignored if `subgroup` is `NULL`. Defaults to `as.character(sort(unique(data[[subgroup]])))`.
`.__caller_env`	(`environment`) the environment from which this function was called. Defaults to `rlang::caller_env()`.

Value

An object of class `emmGrid`: the result of `emmeans::emmeans()`. Note that for a result `result`, the elements `result@model.info$nesting` and `result@misc$display` are removed.

Examples

```
# Create a usable data set out of mmrm::fev_data
fev_mod <- mmrm::fev_data
fev_mod$VISITN <- fev_mod$VISITN * 10
fev_mod$time_cont <- fev_mod$VISITN + rnorm(nrow(fev_mod))
fev_mod$obs_visit_index <- round(fev_mod$time_cont)

fit <-
  ncs_mmrm_fit(
    data = fev_mod,
    type = "subgroup_full",
    response = FEV1,
    subject = USUBJID,
    cov_structs = c("ar1", "us"),
    time_observed_continuous = time_cont,
    df = 2,
    time_observed_index = obs_visit_index,
    time_scheduled_continuous = VISITN,
    arm = ARMCD,
    control_group = "PBO",
    subgroup = SEX,
    subgroup_comparator = "Male",
    covariates = ~ FEV1_BL + RACE
  )

ncs_emmeans(
  fit = fit,
  observed_time = "time_cont",
  scheduled_time = "VISITN",
  arm = "ARMCD",
  subgroup = "SEX"
)
```

`ncs_mmrm_fit` *Create a Mixed Model with Repeated Measures Using Natural Cubic Splines.*

Description

Builds an `mmrm` model that includes a study arm, optionally a subgroup, and natural cubic splines applied to a continuous time variable. A wrapper around `mmrm::mmrm()`.

Constructs a call to `mmrm::mmrm()` for ncs analysis. Implements natural cubic splines for the continuous time variable. Attempts a sequence of covariance structures in order until one of them successfully converges. Title

Usage

```
ncs_mmrm_fit(
  data,
  type = c("basic", "subgroup_full", "subgroup_reduced"),
  response,
  subject,
  cov_structs = c("us", "toeph", "ar1h", "csh", "cs"),
  cov_struct_group = NULL,
  time_observed_continuous,
  df = 2,
  spline_basis = NULL,
  time_observed_index,
  time_scheduled_continuous = NULL,
  arm = NULL,
  control_group = "control",
  subgroup = NULL,
  subgroup_comparator = NULL,
  covariates = ~1,
  expand_spline_terms = TRUE,
  mmrm_args = list(method = "Satterthwaite"),
  ...
)
```

Arguments

`data`	(`data frame`) data set supplied to the `data` argument of `mmrm::mmrm()` when fitting models. The supplied expression is quoted and must evaluate to a data frame. See **Tidy evaluation support**.
`type`	(`string`) one of `"basic"`, `"subgroup_full"`, or `"subgroup_reduced"`.
`response`	(`numeric` or `string`) the response variable. It can be a `string` identifying the name of an existing variable; otherwise, the supplied expression will be quoted and added to the formula as is (see **Tidy evaluation support**).
`subject`	(`atomic` or `string`) the unique subject identifier forwarded to the `subject` argument of `mmrm::cov_struct()`. Ignored if `cov_structs` is a `list`. Can be a `string` identifying an existing variable; otherwise the supplied expression will be quoted and turned into a `string` with `rlang::expr_deparse()` (see **Tidy evaluation support**).
`cov_structs`	(`character` or `list`) either a `list` of unique `cov_struct` objects or a `character` vector of one or more of the covariance structure abbreviations as described in `mmrm::cov_types()`. These covariance structures will be attempted in order until one of them achieves a converging model fit. Defaults to `c("us", "toeph", "ar1h", "csh", "cs")`.
`cov_struct_group`	(`atomic` or `string`) optional grouping variable to be passed to the `group` argument of `mmrm::cov_struct()`. It can be a `string` identifying an existing variable name; otherwise the supplied expression will be quoted and turned into a `string` with `rlang::expr_deparse()`

(see **Tidy evaluation support**). Ignored if `cov_structs` is a `list`. Defaults to `NULL`, in which case no grouping variable will be used.

`time_observed_continuous`
: (`numeric` or `string`)
the visit's *observed* time point. It must either be a `string` or a name identifying an existing variable (i.e., it cannot be a call). If a name is provided, it is quoted and incorporated into the model formula as is (see **Tidy evaluation support**).

`df`
: (scalar `integer`)
number of degrees of freedom to use to create the spline basis. Passed to the `df` argument of `time_spline_basis()`. Ignored if the `spline_basis` argument is not `NULL`.

`spline_basis`
: (`basis matrix`)
a spline basis: probably a value returned by `time_spline_basis()` (which wraps `splines::ns()`). If `NULL` (the default), then the spline basis will be the result of forwarding `time_observed_continuous` and `df` to `time_spline_basis()`. See **Providing a spline basis**.

`time_observed_index`
: (`ordered` or `string`)
the visit index that the visit shall be associated with, based on the visit's *observed* time point. This will be passed as the `visits` argument of `mmrm::cov_struct()`. It can be a `string` identifying an existing variable; otherwise the supplied expression will be quoted and turned into a `string` with `rlang::expr_deparse()` (see **Tidy evaluation support**). If it does not evaluate to an `ordered` factor, it will be wrapped with `as.ordered()`. Ignored if `cov_structs` is a `list`.

`time_scheduled_continuous`
: (`numeric` or `string`)
the continuous time point when the visit was *scheduled* to occur. Its unique values will identify the time points at which the marginal means and other results will be calculated. It can be a `string` identifying an existing variable name; otherwise the supplied expression will be quoted before being evaluated (see **Tidy evaluation support**).

`arm`
: (`factor` or `string`)
the study arm. It must be a `string` or a name identifying an existing variable (i.e., it cannot be a call). If a name, it will be quoted before being added to the model formula (see **Tidy evaluation support**). If it does not evaluate to a factor or if `control_group` is not its first level, the `data` argument will be wrapped in a `dplyr::mutate()` call that forces this to be the case.

`control_group`
: (`string`)
the value in `arm` denoting the control group. If necessary, `arm` will be preprocessed such that it is a factor with `control_group` as its first level.

`subgroup`
: (`factor` or `string`)
the subgroup. It must be a `string` or a name identifying an existing variable (i.e., it cannot be a call). If a name, it will be quoted before being added to the model formula (see **Tidy evaluation support**). If it does not evaluate to a factor or if `subgroup_comparator` is not its first level, the `data` argument will be wrapped in a `dplyr::mutate()` call that forces this to be the case.

`subgroup_comparator`
: (`string`)
the value in `subgroup` denoting the "main" subgroup that all other subgroups should be compared to. If necessary, `subgroup` will be preprocessed such that it is a factor with `control_group` as its first level.

`covariates`
: (`formula`)
formula containing additional terms that should be added to the `mmrm` model. Defaults to `~ 1`, in which no additional terms will be added. Must not have a left side. Cannot contain `.`. To specify that the model shall not have an intercept, use include `+ 0` or `- 1` in this formula.

`expand_spline_terms`
: (`flag`)
flag indicating whether or not to separate the cubic spline matrix into separate terms (one for each degree of freedom). Defaults to `TRUE`. See **Expanding spline terms**.

`mmrm_args`
: (named `list`)
arguments to be passed to `mmrm::mmrm()`. If any elements have the names `formula`, `data`, or `covariance` they will be ignored. An element named `vcov` will also be ignored unless fitting a model with an unstructured covariance. Defaults to `list(method = "Satterthwaite")`.

`...`
: additional arguments to be passed to `mmrm::mmrm()`. If any elements have the names `formula`, `data`, or `covariance` they will be ignored. An element named `vcov` will also be ignored unless fitting a model with an unstructured covariance. Defaults to `list(method = "Satterthwaite")`. Arguments named in `mmrm_args` supersede any named arguments in `...`.

Value

An `mmrm` object created by `mmrm::mmrm()`.

Providing a spline basis

This function's `spline_basis` argument was designed with `splines::ns()` in mind, which creates a matrix object with classes `basis` and `matrix` as well as multiple attributes. In theory, `spline_basis` does not have to be a `matrix`; however, it still must have a `stats::predict()` method wherein `stats::predict(spline_basis, data[[time_observed_continuous]])` produces an object that can serve as a term in the model.

Covariance structures

The user specifies covariance structure *candidates* via the `cov_structs` argument. These structures will be attempted in order until a model converges successfully.

When any covariance structure other than `"us"` (heterogeneous unstructured) is used, `"Empirical-Bias-Reduced"` is passed to `mmrm::mmrm()` as the `vcov` argument (see `mmrm::mmrm_control()`).

When fitting models, these analysis functions specify the covariance structure through the `covariance` argument of `mmrm::mmrm()`.

Building the model formula

These analysis functions automatically build the model formula from its arguments. The user cannot remove any of these auto-generated terms, but terms can be added via the `covariates` argument.

Time spline terms:

Natural cubic splines will be applied to the `time_observed_continuous` variable in `data`. These splines will be constructed according to the user-specified `spline_basis`. A custom `spline_fn()` is constructed under the hood that accepts `time_observed_continuous` and produces a spline matrix based on the `spline_basis`. Thus, the model formula includes a time spline term resembling `spline_fn(time_observed_continuous)`.

`arm` and `subgroup` terms:

All generated models include an interaction term between the time spline term and the study `arm` term, but `arm` is not included as a main effect by default. If this is desired, use the `covariates` argument (e.g., specify `covariates = ~ arm`).

Concerning `ncs_analysis_subgroup()`, the `subgroup` variable is included as a main effect, and its interaction with the time spline is also included. Furthermore, the second-order interaction term between the time spline, `subgroup`, and `arm` is also included for the main analysis model and the "full" model (when `subgroup_interaction_test = TRUE`; see **Subgroup interaction test** below).

Adding terms with `covariates`:

The user can specify additional terms through the `covariates` argument, which must be a formula.

The user cannot specify the covariance structure with this argument. See the **Covariance structures** section above.

The user can remove the intercept from the model by including `0` as a term in `covariates`.

Model formula templates:

The model formulas that the analysis functions construct will take the form of the formula templates below.

`ncs_analysis()` *(i.e., no subgroup):*

```
response ~
  spline_fn(time_observed_continuous) +
  spline_fn(time_observed_continuous):arm {+
  covariates}
```

`ncs_analysis_subgroup()`*:*
Main analysis model and "full" model:

```
response ~
  spline_fn(time_observed_continuous) +
  subgroup +
  spline_fn(time_observed_continuous):subgroup +
  spline_fn(time_observed_continuous):arm +
  spline_fn(time_observed_continuous):subgroup:arm {+
  covariates}
```

"reduced" model:

```
 response ~
   spline_fn(time_observed_continuous) +
   subgroup +
   spline_fn(time_observed_continuous):subgroup +
   spline_fn(time_observed_continuous):arm {+
   covariates}
```

Expanding spline terms:

When expand_spline_terms = TRUE and spline_basis has at least two dimensions (e.g., if it is a matrix, which is typical), the spline term will be split into multiple terms: one for each of its columns.

For instance, if the user specifies a spline_basis with 3 degrees of freedom, the above no-subgroup model formula template would become:

```
response ~
  spline_fn(time_observed_continuous)[, 1] +
  spline_fn(time_observed_continuous)[, 2] +
  spline_fn(time_observed_continuous)[, 3] +
  spline_fn(time_observed_continuous)[, 1]:arm +
  spline_fn(time_observed_continuous)[, 2]:arm +
  spline_fn(time_observed_continuous)[, 3]:arm {+
  covariates}
```

Examples

```
# Create a usable data set out of mmrm::fev_data
fev_mod <- mmrm::fev_data
fev_mod$VISITN <- fev_mod$VISITN * 10
fev_mod$time_cont <- fev_mod$VISITN + rnorm(nrow(fev_mod))
fev_mod$obs_visit_index <- round(fev_mod$time_cont)

# Example without subgroup:
ncs_mmrm_fit(
  data = fev_mod,
  type = "basic",
  response = FEV1,
  subject = USUBJID,
  cov_structs = c("ar1", "us"),
  time_observed_continuous = time_cont,
  df = 2,
  time_observed_index = obs_visit_index,
  time_scheduled_continuous = VISITN,
  arm = ARMCD,
  control_group = "PBO",
  covariates = ~ FEV1_BL + RACE
)

# Example with subgroup:
ncs_mmrm_fit(
  data = fev_mod,
  type = "subgroup_full",
  response = FEV1,
  subject = USUBJID,
  cov_structs = c("ar1", "us"),
  time_observed_continuous = time_cont,
  df = 2,
  time_observed_index = obs_visit_index,
  time_scheduled_continuous = VISITN,
  arm = ARMCD,
  control_group = "PBO",
  subgroup = SEX,
  subgroup_comparator = "Male",
  covariates = ~ FEV1_BL + RACE
)
```

`ncs_plot_means` *Plot Actual and Predicted Response Variable Means by Study Arm.*

Description

This function accepts a data set, probably produced by `ncs_analysis()`, and it uses ggplot2 to produce a panel of plots, one for each study `arm`. The `time` variable is along the x-axis, and the response variable is along the y-axis. The actual means of the response variable are points plotted in one color, and the modeled means are plotted in another color. Each point also has its confidence interval plotted.

Usage

```
ncs_plot_means(
  data,
  arm = "arm",
  time = "time",
  est = "est",
  lower = "lower",
  upper = "upper",
  model_est = "response_est",
  model_lower = "response_lower",
  model_upper = "response_upper"
)
```

Arguments

`data`	(`data frame`) a data frame, probably produced by `ncs_analysis()`, containing the actual and predicted means. Each row should have a unique combination of `arm` and `time`.
`arm`	(`string`) the name of the study arm variable in `data`. There will be a separate plot produced for each study arm.

`time`	(`string`) the name of the time or visit variable in `data`. These values correspond to the x-axis.
`est`, `lower`, `upper`	(`string`) the name of the variables in `data` containing the actual response variable's mean and confidence interval bounds. These values correspond to the y-axis.
`model_est`, `model_lower`, `model_upper`	(`string`) the name of the variables in `data` containing the predicted response variable's mean and confidence interval bounds. These values correspond to the y-axis.

Value

An object returned by `ggplot2::ggplot()`.

Examples

```
# Create a usable data set out of mmrm::fev_data
fev_mod <- mmrm::fev_data
fev_mod$VISITN <- fev_mod$VISITN * 10
fev_mod$time_cont <- fev_mod$VISITN + rnorm(nrow(fev_mod))
fev_mod$obs_visit_index <- round(fev_mod$time_cont)

# Analysis result data set
ncs_data_results <-
  ncs_analysis(
    data = fev_mod,
    response = FEV1,
    subject = USUBJID,
    arm = ARMCD,
    control_group = "PBO",
    time_observed_continuous = time_cont,
    df = 2,
    time_observed_index = obs_visit_index,
    time_scheduled_continuous = VISITN,
    time_scheduled_baseline = 10,
    time_scheduled_label = AVISIT,
    covariates = ~ FEV1_BL + RACE,
    cov_structs = c("ar1", "us")
  )

ncs_plot_means(ncs_data_results)
```

---

ncs_plot_means_subgroup

Plot Actual and Predicted Response Variable Means by Study Arm and Subgroup.

Description

This function accepts a data set, probably produced by `ncs_analysis_subgroup()`, and it uses ggplot2 to produce a grid of plots, one for each combination of study `arm` and `subgroup`. The `time` variable is along the x-axis, and the response variable is along the y-axis. The actual means of the response variable are points plotted in one color, and the modeled means are plotted in another color. Each point also has its confidence interval plotted.

Usage

```
ncs_plot_means_subgroup(
  data,
  arm = "arm",
  time = "time",
  subgroup = "subgroup",
  est = "est",
  lower = "lower",
  upper = "upper",
  model_est = "response_est",
  model_lower = "response_lower",
  model_upper = "response_upper"
)
```

Arguments

`data`	`(data frame)` a data frame, probably produced by `ncs_analysis()`, containing the actual and predicted means. Each row should have a unique combination of `arm`, `time`, and `subgroup`.
`arm`	`(string)` the name of the study arm variable in `data`. There will be a separate column of plots produced for each study arm.
`time`	`(string)` the name of the time or visit variable in `data`. These values correspond to the x-axis.
`subgroup`	`(string)` the name of the subgroup variable in `data`. There will be a separate row of plots produced for each subgroup.
`est`, `lower`, `upper`	`(string)` the name of the variables in `data` containing the actual response variable's mean and confidence interval bounds. These values correspond to the y-axis.
`model_est`, `model_lower`, `model_upper`	`(string)` the name of the variables in `data` containing the predicted response variable's mean and confidence interval bounds. These values correspond to the y-axis.

Value

An object returned by `ggplot2::ggplot()`.

Examples

```
# Create a usable data set out of mmrm::fev_data
fev_mod <- mmrm::fev_data
fev_mod$VISITN <- fev_mod$VISITN * 10
fev_mod$time_cont <- fev_mod$VISITN + rnorm(nrow(fev_mod))
fev_mod$obs_visit_index <- round(fev_mod$time_cont)

# Analysis result data set
ncs_data_results_subgroup <-
  ncs_analysis_subgroup(
    data = fev_mod,
    response = FEV1,
    subject = USUBJID,
    arm = ARMCD,
    control_group = "PBO",
    subgroup = RACE,
    subgroup_comparator = "Asian",
    time_observed_continuous = time_cont,
    df = 2,
    time_observed_index = obs_visit_index,
    time_scheduled_continuous = VISITN,
    time_scheduled_baseline = 10,
    time_scheduled_label = AVISIT,
    covariates = ~ FEV1_BL + RACE,
    cov_structs = c("ar1", "us")
  )

ncs_plot_means_subgroup(ncs_data_results_subgroup$between)
```

`percent_slowing_using_change_from_bl` *Calculates Percent Slowing from a Data Frame of Change-from-Baseline Data*

Description

Accepts a data frame of change-from-baseline data (probably created with `change_from_baseline()`) and returns a table of percent slowing results.

Usage

```
percent_slowing_using_change_from_bl(
  change_from_bl_tbl,
  time_observed_continuous,
  arm,
  control_group,
  subgroup = NULL,
  est = "estimate",
  se = "SE",
  conf.level = 0.95
)
```

Arguments

`change_from_bl_tbl`
: (data frame)
a data frame of change-from-baseline data whose columns include `time_observed_continuous`, `arm`, (optionally) `subgroup`, `est`, and `se`.

`time_observed_continuous`, `arm`, `subgroup`, `est`, `se`
: (`string`)
strings identifying the columns in `change_from_bl_tbl` that contain the continuous time variable, the study arm, (optionally) the subgroup, the change-from-baseline estimate, and the change-from-baseline standard error.

`control_group`
: (`string`)
the value in the `arm` column of `change_from_bl_tbl` denoting the control group.

`conf.level`
: (scalar `numeric`)
confidence level for the calculation of p-values. Defaults to `0.95`.

Details

For each study arm that is not the control group,

$$\text{Let } \theta = \frac{\text{treatment estimate}}{\text{control estimate}}$$

$$\text{Let } \alpha = 1 - \texttt{conf.level}$$

$$\text{Let MOE} = 100 \times z_{1-\alpha/2} \times \frac{\sqrt{\text{treatment SE}^2 + (\theta \times \text{control SE})^2}}{|\text{control estimate}|}$$

Therefore, the percent slowing estimates and their respective confidence intervals are calculated thus:

$$\text{Percent slowing estimate} = (1 - \theta) \times 100$$

$$\text{Percent slowing CI} = \text{Percent slowing estimate} \pm \text{MOE}$$

Value

A data frame with a row for each combination of the unique values of `change_from_bl_tbl[[time_observed_continuous` `change_from_bl_tbl[[arm]]` (except the value denoted in `control_group`), and `change_from_bl_tbl[[subgroup]]` (if `subgroup` is not `NULL`). It will contain the following columns:

1. {column name will be the value of the `arm` argument}: the study arm.

2. {column name will be the value of the `time_observed_continuous` argument}: the *observed* continuous time variable.
3. {column name will be the value of the `subgroup` argument}: the subgroup. Only present if `subgroup` is not `NULL`.
4. `percent_slowing_est`: the percent slowing estimate
5. `percent_slowing_lower`: the lower bound of the confidence interval for `percent_slowing_est`.
6. `percent_slowing_lower`: the upper bound of the confidence interval for `percent_slowing_est`.

Examples

```
# Create a usable data set out of mmrm::fev_data
fev_mod <- mmrm::fev_data
fev_mod$VISITN <- fev_mod$VISITN * 10
fev_mod$time_cont <- fev_mod$VISITN + rnorm(nrow(fev_mod))
fev_mod$obs_visit_index <- round(fev_mod$time_cont)

fit <-
  ncs_mmrm_fit(
    data = fev_mod,
    type = "subgroup_full",
    response = FEV1,
    subject = USUBJID,
    cov_structs = c("ar1", "us"),
    time_observed_continuous = time_cont,
    df = 2,
    time_observed_index = obs_visit_index,
    time_scheduled_continuous = VISITN,
    arm = ARMCD,
    control_group = "PBO",
    subgroup = SEX,
    subgroup_comparator = "Male",
    covariates = ~ FEV1_BL + RACE
  )

marginal_means <-
  ncs_emmeans(
    fit = fit,
    observed_time = "time_cont",
    scheduled_time = "VISITN",
    arm = "ARMCD",
    subgroup = "SEX"
  )

change_from_bl_tbl <-
  change_from_baseline(
    emmeans = marginal_means,
    time_observed_continuous = "time_cont",
    time_scheduled_baseline = 10,
    arm = "ARMCD",
    subgroup = "SEX",
    as_tibble = TRUE
  )

percent_slowing_using_change_from_bl(
  change_from_bl_tbl = change_from_bl_tbl,
  time_observed_continuous = "time_cont",
  arm = "ARMCD",
  control_group = "PBO",
  subgroup = "SEX"
)
```

plot_outcome_by_visit_and_group

Plot Outcome Variable by Timepoint and Study Arm

Description

Plot a continuous outcome for each combination of scheduled visit and study arm.

Usage

```
plot_outcome_by_visit_and_group(
  data,
  outcome_var,
  scheduled_timepoint_var,
  group_var,
  ...,
  geom = ggplot2::geom_boxplot,
  geom_args = list(na.rm = TRUE)
)
```

Arguments

`data`	(`data frame`) The data frame that will be supplied to `ggplot2::ggplot()`.
`outcome_var`	(`numeric`) The continuous outcome variable to supply to the y argument of `ggplot2::aes()`. Whatever is supplied will be quoted and evaluated in the context of `data`.
`scheduled_timepoint_var`	(`ordered`) The variable containing the scheduled timepoints to supply to the `x` argument of `ggplot2::aes()`.. Whatever is supplied will be quoted and evaluated in the context of `data`.
`group_var`	(`numeric`) The grouping variable (probably the study arm) to supply to the `fill` argument of `ggplot2::aes()`.. Whatever is supplied will be quoted and evaluated in the context of `data`.
`...`	Forwarded onto `ggplot2::ggplot(ggplot2::aes)`.

`geom`	(`function`) The ggplot2 "geom" to use. Defaults to `ggplot2::geom_boxplot()`.
`geom_args`	(`list`) A list of arguments to supply to `geom`. Defaults to `list(na.rm = TRUE)`.

Value

A ggplot object.

Examples

```
# Create a usable data set out of mmrm::fev_data
fev_mod <- mmrm::fev_data
fev_mod$VISITN <- fev_mod$VISITN * 10
fev_mod$time_cont <- fev_mod$VISITN + rnorm(nrow(fev_mod))
fev_mod$obs_visit_index <- round(fev_mod$time_cont)

plot_outcome_by_visit_and_group(
    data = fev_mod,
    outcome_var = FEV1,
    scheduled_timepoint_var = as.ordered(VISITN),
    group_var = ARMCD
)
```

`splinetrials_analysis-class`
`splinetrials_analysis` *object*

Description

`ncs_analysis()` returns an object of class `splinetrials_analysis`: a 32-column `tibble` with one row per unique combination of `data[[arm]]` and `data[[time_scheduled_label]]` (see the arguments of `ncs_analysis()`).

Columns

1. `arm`: values of `data[[arm]]`.
2. `time`: values of `data[[time_scheduled_label]]`.
3. `n`: number of times the combination appears in data.
4. `est`: mean of `data[[response]]`.
5. `sd`: standard deviation of `data[[response]]`.
6. `se`: standard error of `data[[response]]` (i.e., `sd / sqrt(n)`).
7. `lower`: lower bound of confidence interval.
8. `upper`: upper bound of confidence interval.
9. `response_est`: estimated marginal mean.

10. `response_se`: standard error of `response_est`.
11. `response_df`: degrees of freedom used for calculating the confidence interval for `response_est`.
12. `response_lower`: lower bound of confidence interval for `response_est`.
13. `response_upper`: upper bound of confidence interval for `response_est`.
14. `change_est`: estimated change from baseline.
15. `change_se`: standard error of `change_est`.
16. `change_df`: degrees of freedom used for calculating the confidence interval for and testing the significance of `change_est`.
17. `change_lower`: lower bound of confidence interval for `change_est`.
18. `change_upper`: upper bound of confidence interval for `change_est`.
19. `change_test_statistic`: test statistic measuring the significance of `change_est`.
20. `change_p_value`: p-value for the significance of `change_est`.
21. `diff_est`: treatment effect.
22. `diff_se`: standard error of `diff_est`.
23. `diff_df`: degrees of freedom used for calculating the confidence interval for and testing the significance of `diff_est`.
24. `diff_lower`: lower bound of confidence interval for `diff_est`.
25. `diff_upper`: upper bound of confidence interval for `diff_est`.
26. `diff_test_statistic`: test statistic measuring the significance of `diff_est`.
27. `diff_p_value`: p-value for the significance of `diff_est`.
28. `percent_slowing_est`: estimated percent slowing.
29. `percent_slowing_lower`: lower bound of confidence interval for `percent_slowing_est`.
30. `percent_slowing_upper`: upper bound of confidence interval for `percent_slowing_est`.
31. `correlation`: the covariance structure of the analysis model. This is the same value repeated for each row.
32. `optimizer`: invariably `mmrm+tmb` to indicate that `mmrm::mmrm()` (which uses the `TMB` package) was used to fit the model.

Optional `analysis_model` attribute

If `ncs_analysis()` had `return_models = TRUE`, then the analysis model, an `mmrm` object, will be included as the `analysis_model` attribute.

See Also

The function `ncs_analysis()`, which produces objects of this class.

`splinetrials_subgroup_analysis-class` `splinetrials_subgroup_analysis` *object*

Description

`ncs_analysis_subgroup()` returns an object of class `splinetrials_subgroup_analysis`: a named list with three to seven elements.

`between` and `within`

These are each `tibbles`, and they share many of the same columns and values but are sorted in a different order. Each contains one row per unique combination of `arm`, `time_scheduled_label`, and `subgroup` found in the `data` (see the arguments of `ncs_analysis_subgroup()`). The values in columns `arm` through `change_p_value` as well as `correlation` and `optimizer` are identical. The two tables' treatment effect analysis results columns differ in name and content, with `between`'s columns bearing the prefix `diff_subgroup_` and `within`'s columns bearing the prefix `diff_arm_` (see the **Treatment effects** section of `ncs_analysis_subgroup()`). Lastly, only `within` contains the percent slowing analysis results.

`between`:

A 30-column `tibble` sorted by `time`, then by `arm`, then by `subgroup`.

Columns:

1. `arm`: values of `data[[arm]]`.
2. `time`: values of `data[[time_scheduled_label]]`.
3. `subgroup`: values of `data[[subgroup]]`.
4. `n`: number of times the combination appears in data.
5. `est`: mean of `data[[response]]`.
6. `sd`: standard deviation of `data[[response]]`.
7. `se`: standard error of `data[[response]]` (i.e., `sd / sqrt(n)`).
8. `lower`: lower bound of confidence interval.
9. `upper`: upper bound of confidence interval.
10. `response_est`: estimated marginal mean.
11. `response_se`: standard error of `response_est`.
12. `response_df`: degrees of freedom used to calculate the confidence interval for `response_est`.
13. `response_lower`: lower bound of confidence interval for `response_est`.
14. `response_upper`: upper bound of confidence interval for `response_est`.
15. `change_est`: estimated change from baseline.
16. `change_se`: standard error of `change_est`.
17. `change_df`: degrees of freedom used for calculating the confidence interval for and testing the significance of `change_est`.
18. `change_lower`: lower bound of confidence interval for `change_est`.
19. `change_upper`: upper bound of confidence interval for `change_est`.

20. `change_test_statistic`: test statistic measuring the significance of `change_est`.
21. `change_p_value`: p-value for the significance of `change_est`.
22. `diff_subgroup_est`: treatment effect of `subgroup` within `arm`.
23. `diff_subgroup_se`: standard error of `diff_subgroup_est`.
24. `diff_subgroup_df`: degrees of freedom used for calculating the confidence interval for and testing the significance of `diff_subgroup_est`.
25. `diff_subgroup_lower`: lower bound of confidence interval for `diff_subgroup_est`.
26. `diff_subgroup_upper`: upper bound of confidence interval for `diff_subgroup_est`.
27. `diff_subgroup_test_statistic`: test statistic measuring the significance of `diff_subgroup_est`.
28. `diff_subgroup_p_value`: p-value for the significance of `diff_subgroup_est`.
29. `correlation`: the covariance structure of the analysis model. This is the same value repeated for each row.
30. `optimizer`: invariably `mmrm+tmb` to indicate that `mmrm::mmrm()` (which uses the `TMB` package) was used to fit the model.

`within`

A 33-column `tibble` sorted by `subgroup`, then by `arm`, then by `time`.

Columns:

1. `arm`: values of `data[[arm]]`.
2. `time`: values of `data[[time_scheduled_label]]`.
3. `subgroup`: values of `data[[subgroup]]`.
4. `n`: number of times the combination appears in data.
5. `est`: mean of `data[[response]]`.
6. `sd`: standard deviation of `data[[response]]`.
7. `se`: standard error of `data[[response]]` (i.e., `sd / sqrt(n)`).
8. `lower`: lower bound of confidence interval.
9. `upper`: upper bound of confidence interval.
10. `response_est`: estimated marginal mean.
11. `response_se`: standard error of `response_est`.
12. `response_df`: degrees of freedom used for calculating the confidence interval for `response_est`.
13. `response_lower`: lower bound of confidence interval for `response_est`.
14. `response_upper`: upper bound of confidence interval for `response_est`.
15. `change_est`: estimated change from baseline.
16. `change_se`: standard error of `change_est`.
17. `change_df`: degrees of freedom for calculating the confidence interval for and estimating the significance of `change_est`.
18. `change_lower`: lower bound of confidence interval for `change_est`.
19. `change_upper`: upper bound of confidence interval for `change_est`.
20. `change_test_statistic`: test statistic measuring the significance of `change_est`.

21. `change_p_value`: p-value for the significance of `change_est`.
22. `diff_arm_est`: treatment effect of `arm` within `subgroup`.
23. `diff_arm_se`: standard error of `diff_arm_est`.
24. `diff_arm_df`: degrees of freedom for calculating the confidence interval for and testing the significance of `diff_arm_est`.
25. `diff_arm_lower`: lower bound of confidence interval for `diff_arm_est`.
26. `diff_arm_upper`: upper bound of confidence interval for `diff_arm_est`.
27. `diff_arm_test_statistic`: test statistic measuring the significance of `diff_arm_est`.
28. `diff_arm_p_value`: p-value for the significance of `diff_arm_est`.
29. `percent_slowing_est`: estimated percent slowing.
30. `percent_slowing_lower`: lower bound of confidence interval for `percent_slowing_est`.
31. `percent_slowing_upper`: upper bound of confidence interval for `percent_slowing_est`.
32. `correlation`: the covariance structure of the analysis model. This is the same value repeated for each row.
33. `optimizer`: invariably `mmrm+tmb` to indicate that `mmrm::mmrm()` (which uses the `TMB` package) was used to fit the model.

`type3`

A `tibble` with a row for each term in the model (not counting any intercepts). Contains the following six columns:

1. `effect`: the name of the model term.
2. `chisquare_test_statistic`: the Chi-squared test statistic measuring the significance of the model term.
3. `df`: the degrees of freedom used for testing the significance of the model term.
4. `p_value`: the p-value for the significance of the model term.
5. `correlation`: the covariance structure of the analysis model. This is the same value repeated for each row.
6. `optimizer`: invariably `mmrm+tmb` to indicate that `mmrm::mmrm()` (which uses the `TMB` package) was used to fit the model.

`interaction`

This element is only present if `subgroup_interaction_test = TRUE`.

A 2 by 10 data frame with class `anova.mmrm`. The first row represents the "reduced" model and the second row represents the "full" model. The columns are as follows:

1. `model`: `c("reduced model", "full model")`, identifying the model associated with each row.
2. `aic`: the AIC of the model.
3. `bic`: the BIC of the model.
4. `loglik`: the log likelihood of the model.

5. `-2*log(l)`: equal to `-2 * loglik`.
6. `test_statistic`: the test statistic used for testing the significance of the second-order interaction term(s) between the spline time, `subgroup`, and `arm`. This value is the second element of the column; the first element is always a missing value.
7. `df`: the degrees of freedom used for testing the significance of the second-order interaction term(s) between the spline term, `subgroup`, and `arm`. This value is the second element of the column; the first element is always a missing value.
8. `p_value`: the p-value for the significance of the second-order interaction term(s) between the spline term, `subgroup`, and `arm`. This value is the second element of the column; the first element is always a missing value.
9. `correlation`: the covariance structure of the analysis model. This is the same value repeated for each row.
10. `optimizer`: invariably `mmrm+tmb` to indicate that `mmrm::mmrm()` (which uses the `TMB` package) was used to fit the model.

`analysis_model`

This element is only present if `return_models = TRUE`.

An `mmrm` object: the fitted model used to perform analyses that produced the `between`, `within`, and `type3` results.

`full` **and** `reduced`

These elements are only present if `subgroup_interaction_test = TRUE` and `return_models = TRUE`.

Both are `mmrm` objects: the two maximum-likelihood-estimated models used to perform the subgroup interaction test whose results are in the `interaction` element. See the **Subgroup interaction test** section of `ncs_analysis_subgroup()`.

See Also

The function `ncs_analysis_subgroup()`, which produces objects of this class.

`time_spline` *Create Natural Cubic Spline Approximations for Continuous Time*

Description

Accepts or constructs a natural cubic spline `basis` for continuous `time` and yields a matrix of approximations for `time` according to that `basis`.

Usage

```
time_spline(
  time,
  df = NULL,
  ...,
  basis = time_spline_basis(time, df = df, ...)
)
```

Arguments

`time`	A numeric vector of values.
`df, ...`	Only used if `basis` is left as the default. Passed to `time_spline_basis()` (which passes all arguments to `splines::ns()`) to calculate the spline `basis`.
`basis`	Spline basis for which to create approximations of `time`. Defaults to `time_spline_basis(time, df = df`

Details

`time_spline()` is primarily useful because it can use one step to create the spline `basis` from `time` and then re-input `time` into the spline `basis` to obtain the spline approximations. Alternatively, it can calculate predictions from a basis supplied to the `basis` argument.

Value

Matrix with the same dimensions as `basis`. Contains `basis` as an attribute.

Examples

```
time_spline(Theoph$Time, df = 3)

# Or, compute the spline basis beforehand, and then pass it to time_spline()
basis <-
  splines::bs(Theoph$Time, df = 3, Boundary.knots = c(0, max(Theoph$Time)))

time_spline(Theoph$Time, basis = basis)
```

`time_spline_basis` *Natural Cubic Spline Basis Matrix for Continuous Time.*

Description

Wrapper around `splines::ns()` with default `Boundary.knots` of `c(0, max(time))`.

Usage

```
time_spline_basis(time, df, Boundary.knots = c(0, max(time)), ...)
```

Arguments

`time`	Continuous time variable, passed directly to `splines::ns()` as the first argument.
`df`	Degrees of freedom, passed directly to the `df` argument of `splines::ns()`.
`Boundary.knots`	Boundary knots, passed directly to the `Boundary.knots` argument of `splines::ns()`. Defaults to `c(0, max(time))`.
`...`	Passed to `splines::ns()`.

Details

`time_spline()` is primarily useful because it can create the spline basis from `time` and then re-input `time` into the spline basis to obtain the predictions in one step. Or, it can calculate predictions from a basis supplied to the `basis` argument.

Value

A matrix of dimension `length(time) * df`. See the *Value* section of `splines::ns()`.

Examples

```
time_spline_basis(Theoph$Time, df = 3)
```

Index