`maeve` long-form overview

Bill Forrest (Genentech OMNI Bioinformatics)

2020-10-19

Abstract

This document contains lots of working commands with real data sets, but with minimal explanation. There is a companion manuscript in preparation to discuss the problems we address in translational oncology and theoretical aspects of the models and summary measures employed.

Contents

1 Introduction

The R package maeve coordinates fitting of linear and generalized additive mixed models for normally distributed longitudinal data from multi-group studies with multiple subjects per group, then computes one statistical summary per group and compares the resulting summary statistics across groups. The motivating application is to tumor volume growth data from mouse studies in translational oncology. Below are a number of detailed maeve code examples with real data. This vignette is light on conceptual motivation or explanation, for which readers should see the companion manuscript.

```
## This is a vignette for `maeve`. Here are `R` packages used in the analysis:
##
suppressPackageStartupMessages( library( magrittr ) )
suppressPackageStartupMessages( library(    maeve ) )  # in GRAN
suppressPackageStartupMessages( library(  ggplot2 ) )
##
options( width = 110 )
##
ggplot2::theme_set( switch( 2, theme_grey(), theme_bw() ) )
##
```

2 Option management with maeve_options(), maeve_reset(), and maeve_defaults()

The package settings is used to establish a local version of an options list that can be incorporated into package functions and locally over-ruled as needed. These options are managed by the functions maeve_options(), maeve_reset(), and maeve_defaults().

```
###
### script with commands for maeve demonstration

### maeve_options(), maeve_reset()
###
### Options will persist and provide a shared set of
### arguments to maeve functions.  They can be inpected
### or reset individually with the function `maeve_options()`.
###
### The function 'maeve_reset()` will reset all options to
### their default values.
###
### The function `maeve_defaults()` will list the package
### default values for the argument given (e.g.,
### 'maeve_defaults("endpoint_name")'), or all the default
### values if called with no arguments: 'maeve_defaults()'.


maeve_options('group_name')
```

```
## [1] "group_name"
```

```
maeve_options('subject_ID')
```

```
## [1] "animalID"
```

```
maeve_options('x_name')
```

```
## [1] "DAY_OF_STUDY"
```

```
maeve_options('endpoint_name')
```

```
## [1] "y"
```

```
maeve_options('reference_Dunnett')
```

```
## NULL
```

```
### reset names of variables in the maeve_options() list so that package functions can access them:
maeve_options( group_name = 'assay_rep',
               subject_ID = 'cell_line',
                 'x_name' = 'log_concentration', # option names (left-hand side) can be quoted or unquoted.
           endpoint_name = 'OD',
       reference_Dunnett = 'my favorite reference'
             )

 ## See that they're changed:
 maeve_options( 'group_name' , 'subject_ID', 'x_name',  'endpoint_name', 'reference_Dunnett' )
```

```
## $group_name
## [1] "assay_rep"
##
## $subject_ID
## [1] "cell_line"
##
## $x_name
## [1] "log_concentration"
##
## $endpoint_name
## [1] "OD"
##
## $reference_Dunnett
## [1] "my favorite reference"
```

```
 ## But information on original values is always available from maeve_defaults(), which
 ## will not change anything:
 maeve_defaults( 'group_name' , 'subject_ID', 'x_name',  'endpoint_name', 'reference_Dunnett' )
```

```
## $group_name
## [1] "group_name"
##
## $subject_ID
## [1] "animalID"
##
## $x_name
## [1] "DAY_OF_STUDY"
##
## $endpoint_name
## [1] "TUMOR_VOLUME"
##
## $reference_Dunnett
## NULL
```

```
 ## Return all the options to package defaults with the maeve_reset() function:
 maeve_reset()

 ## See that they've returned to the original values:
```

```
maeve_options( 'group_name' , 'subject_ID', 'x_name',  'endpoint_name', 'reference_Dunnett' )
```

```
## $group_name
## [1] "group_name"
##
## $subject_ID
## [1] "animalID"
##
## $x_name
## [1] "DAY_OF_STUDY"
##
## $endpoint_name
## [1] "TUMOR_VOLUME"
##
## $reference_Dunnett
## NULL
```

```
## Here are all the  option names:
names( maeve_options( ) )
```

```
##  [1] "group_name"              "subject_ID"              "trace_ID"                "x_name"
##  [5] "endpoint_name"           "EOS_CR_minval"           "PR_threshold"            "summary_first_day"
##  [9] "full_study_data_frame"   "modeling_data_frame"     "autoset_full_study_data" "autoset_modeling_data"
## [13] "truncate_fit"            "truncation_return_type"  "truncated_group_levels"  "min_n_in_group"
## [17] "min_frac_in_group"       "min_frac_in_study"       "overall_x_min"           "overall_x_max"
## [21] "abbreviate_n"            "reference_Dunnett"       "add_to_endpoint"         "restrict_x"
## [25] "number_basis_vecs"       "min_basis_vecs"          "max_basis_vecs"          "trans_func_char"
## [29] "inv_func_char"           "test_func_x"             "weight_lmer_option"      "metric"
## [33] "metrics_supported"       "N_integration_grid"      "contrast"                "xrange_norm_method"
## [37] "progress"                "number_break_points"     "break_points"            "adjust_break_points"
## [41] "poly_degree"             "poly_object"             "mismatch_action"         "x_pred_type"
## [45] "x_pred_vec"              "x_pred_spacing"          "x_pred_interior_grid"    "include_newdata_ID"
## [49] "title_label"             "y_label"                 "x_label"                 "legend_position_char"
## [53] "axis_text_x_angle"       "axis_text_x_size"        "axis_text_x_hjust"       "axis_text_x_vjust"
## [57] "axis_text_y_size"        "legend_text_size"        "strip_text_size"         "linear_predictor"
## [61] "linear_color"            "linear_lwd"              "spline_predictor"        "spline_color"
## [65] "spline_lwd"              "piecewise_predictor"     "piecewise_color"         "piecewise_lwd"
## [69] "poly_predictor"          "poly_color"              "poly_lwd"                "geom_na_rm"
## [73] "geom_point_size"         "nrow_value"              "ncol_value"              "alpha_value"
```

3 Data sets in maeve, real and simulated.

Three real data sets are included in the package `/data` directory: `vismodegib`, `palb_temo_01`, and `gdc0084_temo_01`.

```
data_set_names <- data( package = 'maeve' )
data_set_names <- data_set_names$results[,'Item']
print( data_set_names )
```

```
## [1] "gdc0084_temo_01" "palb_temo_01"    "vismodegib"
```

Several simulated data sets are in the `/inst/extdata` directory.

4 Optional QC with `check_study_data_frame()`

The function `check_study_data_frame()` takes a data frame intended for analysis in `maeve` and does a number of checks on columns designated by `group_name`, `subject_ID`, `x_name`, and `endpoint_name`. This step is optional (e.g., check that levels of `subject_ID` are nested within levels of `group_name`). You do not *need* to run the data frame through `check_study_data_frame()`, but it encapsulates many QC checks for sometimes subtle but common formatting issues that would then crop up down the road in harder-to-trace guises.

```
### check_study_data_frame() basics

### `check_study_data_frame()` takes a data frame and
### returns another data.frame after doing some QC checks.
### In brief:
### It looks at maeve_options() and finds the four columns
### matching 'group_name', 'subject_ID', 'x_name', and 'endpoint_name'
###
### (1) 'group_name' and 'subject_ID' must be factors.
### (2) 'subject_ID' must be nested within 'group_name'.
### (3) 'xname' must be numeric.
### (4) 'endpoint_name' must be numeric.
###
### By default it returns just these four columns.

 maeve_reset()

 data( vismodegib, package = 'maeve' ) # cleanly formatted version of "hedgehog" data

### Make a mis-formatted version of the vismodegib data by converting 'animalID' to character
### and 'DAY_OF_STUDY' to integer:

 vismo_bad_format <-
     vismodegib %>%
     dplyr::mutate( animalID     = as.character( animalID ),  # started as a factor
                    DAY_OF_STUDY = as.integer( DAY_OF_STUDY ) # started as a numeric
                  )


### maeve::check_study_data_frame() will return warnings about these
### mis-formatted fields and by default convert the 'animalID' field
### to a factor and the 'DAY_OF_STUDY' field to a numeric:

 vismo_fixed_format <- maeve::check_study_data_frame( vismo_bad_format )
```

```
## Warning in maeve::check_study_data_frame(vismo_bad_format): Converting animalID to factor in
## maeve::check_study_data_frame().
```

```
## Warning in maeve::check_study_data_frame(vismo_bad_format): Converting DAY_OF_STUDY from integer to numeric in
## maeve::check_study_data_frame().
```

```
### If we run the original data frame through 'check_study_data_frame()',
### it returns the four needed columns with no warnings:

 vismo_already_OK <- maeve::check_study_data_frame( vismodegib )


### Check that 'check_study_data_frame()' has fixed the "problems":

 identical( vismo_fixed_format, vismo_already_OK )
```

```
## [1] TRUE
```

```
### See the output:
 vismodegib %>% check_study_data_frame() %>% head( 5 )
```

```
##   group_name animalID DAY_OF_STUDY TUMOR_VOLUME
## 1   dose_0.0  3810580            0     212.7522
## 2   dose_0.0  3810580            3     283.0926
## 3   dose_0.0  3810580            6     337.0389
## 4   dose_0.0  3810580           10     479.5100
## 5   dose_0.0  3810580           14     429.2777
```

```
### See the output with in-line changes in the
### requested columns (here, e.g., switch from
### "TUMOR_VOLUME" to "BODY_WEIGHT":
###
 vismodegib %>% check_study_data_frame( endpoint_name = 'BODY_WEIGHT' ) %>% head( 5 )
```

```
##   group_name animalID DAY_OF_STUDY BODY_WEIGHT
## 1   dose_0.0  3810580            0        28.5
## 2   dose_0.0  3810580            3        29.2
## 3   dose_0.0  3810580            6        28.8
## 4   dose_0.0  3810580           10        30.1
## 5   dose_0.0  3810580           14        29.6
```

```
### You can re-set variable names in maeve_options() and then
### use off-the-shelf tidyverse tools to rename covariates and / or
### define new variables on the fly:

 maeve_options( x_name = 'day', endpoint_name = 'log_TUMOR_VOLUME' )

 vismodegib %>%
     dplyr::rename( day = DAY_OF_STUDY ) %>%
     dplyr::mutate( log_TUMOR_VOLUME = log( 1 + TUMOR_VOLUME ) ) %>%
     check_study_data_frame() %>%
     head( 10 )
```

```
##    group_name animalID day log_TUMOR_VOLUME
## 1    dose_0.0  3810580   0         5.364818
## 2    dose_0.0  3810580   3         5.649300
## 3    dose_0.0  3810580   6         5.823161
## 4    dose_0.0  3810580  10         6.174848
## 5    dose_0.0  3810580  14         6.064431
## 6    dose_0.0  3810580  17         6.386843
## 7    dose_0.0  3810580  21         6.288881
## 8    dose_0.0  3810587   0         5.415381
## 9    dose_0.0  3810587   3         5.450238
## 10   dose_0.0  3810587   7         5.833105
```

```
 maeve_reset()
```

5 Pre-modeling group-level summaries with `tally_study()`

The function `tally_study()` takes a study data frame and returns one row per group, with a number of summary values that can be computed without longitudinal statistical modeling. By default, it tells the number of subjects per group, the first and last time point with observations from each group, and whether all subjects' observations started or ended on the first or last time point, respectively. Optionally, it can provide simple response rate counts (e.g., partial response, end-of-study complete response) by group, as shown in the examples. These counts also include log odds ratios and asymptotic standard errors relative to

a reference group (by default, the first group in the study). More detail on how these are calculated is in the section `End of Study Complete Response and Partial Response` later in this document.

```
### tally_study(), basics

 maeve_reset()

 head( vismodegib, 15 )
```

```
##    study_id group_name cohort_id animalID DAY_OF_STUDY BODY_WEIGHT TUMOR_VOLUME
## 1     22384   dose_0.0        02  3810580            0        28.5     212.7522
## 2     22384   dose_0.0        02  3810580            3        29.2     283.0926
## 3     22384   dose_0.0        02  3810580            6        28.8     337.0389
## 4     22384   dose_0.0        02  3810580           10        30.1     479.5100
## 5     22384   dose_0.0        02  3810580           14        29.6     429.2777
## 6     22384   dose_0.0        02  3810580           17        30.2     592.9786
## 7     22384   dose_0.0        02  3810580           21        29.2     537.5505
## 8     22384   dose_0.0        05  3810587            0        24.5     223.8383
## 9     22384   dose_0.0        05  3810587            3        23.5     231.8135
## 10    22384   dose_0.0        05  3810587            7        25.1     340.4171
## 11    22384   dose_0.0        05  3810587           11        24.9     401.2144
## 12    22384   dose_0.0        05  3810587           14        25.2     389.1488
## 13    22384   dose_0.0        05  3810587           18        25.2     390.5655
## 14    22384   dose_0.0        05  3810587           21        25.5     483.5132
## 15    22384   dose_0.0        06  3810601            0        26.6     213.5640
```

```
### Which maeve options will get used in tallying the study?

 maeve_options() %>%
 with( print( c( group_name, subject_ID, x_name, endpoint_name ) ) )
```

```
## [1] "group_name"   "animalID"     "DAY_OF_STUDY" "TUMOR_VOLUME"
```

```
### Here is the data frame tally_study() will use:
 maeve_options() %>%
 with( vismodegib %>%
       dplyr::select( dplyr::one_of( group_name, subject_ID, x_name, endpoint_name ) ) %>%
       head(15)
      )
```

```
##    group_name animalID DAY_OF_STUDY TUMOR_VOLUME
## 1    dose_0.0  3810580            0     212.7522
## 2    dose_0.0  3810580            3     283.0926
## 3    dose_0.0  3810580            6     337.0389
## 4    dose_0.0  3810580           10     479.5100
## 5    dose_0.0  3810580           14     429.2777
## 6    dose_0.0  3810580           17     592.9786
## 7    dose_0.0  3810580           21     537.5505
## 8    dose_0.0  3810587            0     223.8383
## 9    dose_0.0  3810587            3     231.8135
## 10   dose_0.0  3810587            7     340.4171
## 11   dose_0.0  3810587           11     401.2144
## 12   dose_0.0  3810587           14     389.1488
## 13   dose_0.0  3810587           18     390.5655
## 14   dose_0.0  3810587           21     483.5132
## 15   dose_0.0  3810601            0     213.5640
```

```
 vismodegib %>% tally_study()
```

```
##    group_name N_in_group first_day common_start last_day common_end
## 1    dose_0.0          6         0         TRUE       21       TRUE
## 2    dose_0.3          5         0         TRUE       21      FALSE
## 3    dose_1.0          5         0         TRUE       21       TRUE
## 4    dose_3.0          5         0         TRUE       21       TRUE
## 5    dose_6.0          4         0         TRUE       21       TRUE
## 6     dose_10          4         0         TRUE       21       TRUE
## 7     dose_25          4         0         TRUE       21       TRUE
## 8     dose_50          4         0         TRUE       21       TRUE
## 9     dose_75          4         0         TRUE       21       TRUE
## 10   dose_100          5         0         TRUE       74      FALSE
```

```
### Partial responses:

 vismodegib %>% tally_study( response = 'PR' )
```

```
##    group_name N_in_group first_day common_start last_day common_end PR PR_log_OR PR_SE_log_OR
## 1    dose_0.0          6         0         TRUE       21       TRUE  0 0.0000000     0.000000
## 2    dose_0.3          5         0         TRUE       21      FALSE  0 0.1670541     2.082226
## 3    dose_1.0          5         0         TRUE       21       TRUE  0 0.1670541     2.082226
## 4    dose_3.0          5         0         TRUE       21       TRUE  0 0.1670541     2.082226
## 5    dose_6.0          4         0         TRUE       21       TRUE  0 0.3677248     2.091905
## 6     dose_10          4         0         TRUE       21       TRUE  2 2.5649494     1.718676
## 7     dose_25          4         0         TRUE       21       TRUE  4 4.7621739     2.091905
## 8     dose_50          4         0         TRUE       21       TRUE  4 4.7621739     2.091905
## 9     dose_75          4         0         TRUE       21       TRUE  4 4.7621739     2.091905
## 10   dose_100          5         0         TRUE       74      FALSE  5 4.9628446     2.082226
```

```
 vismodegib %>% tally_study( response = 'PR', PR_threshold = 0.90 )
```

```
##    group_name N_in_group first_day common_start last_day common_end PR PR_log_OR PR_SE_log_OR
## 1    dose_0.0          6         0         TRUE       21       TRUE  0 0.0000000     0.000000
## 2    dose_0.3          5         0         TRUE       21      FALSE  0 0.1670541     2.082226
## 3    dose_1.0          5         0         TRUE       21       TRUE  1 1.4663371     1.744344
## 4    dose_3.0          5         0         TRUE       21       TRUE  2 2.2284771     1.685100
## 5    dose_6.0          4         0         TRUE       21       TRUE  4 4.7621739     2.091905
## 6     dose_10          4         0         TRUE       21       TRUE  4 4.7621739     2.091905
## 7     dose_25          4         0         TRUE       21       TRUE  4 4.7621739     2.091905
## 8     dose_50          4         0         TRUE       21       TRUE  4 4.7621739     2.091905
## 9     dose_75          4         0         TRUE       21       TRUE  4 4.7621739     2.091905
## 10   dose_100          5         0         TRUE       74      FALSE  5 4.9628446     2.082226
```

```
 vismodegib %>%
 tally_study( response = 'PR', PR_threshold = 0.90, reference = 'dose_25' ) %>%
 dplyr::select( group_name, N_in_group, PR, PR_log_OR, PR_SE_log_OR )
```

```
##    group_name N_in_group PR  PR_log_OR PR_SE_log_OR
## 1    dose_0.0          6  0 -4.7621739     2.091905
## 2    dose_0.3          5  0 -4.5951199     2.098581
## 3    dose_1.0          5  1 -3.2958369     1.763834
## 4    dose_3.0          5  2 -2.5336968     1.705267
## 5    dose_6.0          4  4  0.0000000     2.108185
## 6     dose_10          4  4  0.0000000     2.108185
## 7     dose_25          4  4  0.0000000     0.000000
## 8     dose_50          4  4  0.0000000     2.108185
## 9     dose_75          4  4  0.0000000     2.108185
## 10   dose_100          5  5  0.2006707     2.098581
```

```
### Both end-of-study complete responses:

 vismodegib %>% tally_study( response = 'EOS_CR' )
```

```
##    group_name N_in_group first_day common_start last_day common_end EOS_CR EOS_CR_log_OR EOS_CR_SE_log_OR
## 1    dose_0.0          6         0         TRUE       21       TRUE      0     0.0000000         0.000000
## 2    dose_0.3          5         0         TRUE       21      FALSE      0     0.1670541         2.082226
## 3    dose_1.0          5         0         TRUE       21       TRUE      0     0.1670541         2.082226
## 4    dose_3.0          5         0         TRUE       21       TRUE      0     0.1670541         2.082226
## 5    dose_6.0          4         0         TRUE       21       TRUE      0     0.3677248         2.091905
## 6     dose_10          4         0         TRUE       21       TRUE      0     0.3677248         2.091905
## 7     dose_25          4         0         TRUE       21       TRUE      0     0.3677248         2.091905
## 8     dose_50          4         0         TRUE       21       TRUE      0     0.3677248         2.091905
## 9     dose_75          4         0         TRUE       21       TRUE      0     0.3677248         2.091905
## 10   dose_100          5         0         TRUE       74      FALSE      0     0.1670541         2.082226
```

```
 vismodegib %>%
 tally_study( response = 'EOS_CR', EOS_CR_minval = 50 ) %>%
 dplyr::select( group_name, N_in_group, EOS_CR, EOS_CR_log_OR, EOS_CR_SE_log_OR )
```

```
##    group_name N_in_group EOS_CR EOS_CR_log_OR EOS_CR_SE_log_OR
## 1    dose_0.0          6      0     0.0000000         0.000000
## 2    dose_0.3          5      0     0.1670541         2.082226
## 3    dose_1.0          5      0     0.1670541         2.082226
```

```
## 4    dose_3.0          5      0     0.1670541         2.082226
## 5    dose_6.0          4      0     0.3677248         2.091905
## 6     dose_10          4      0     0.3677248         2.091905
## 7     dose_25          4      0     0.3677248         2.091905
## 8     dose_50          4      4     4.7621739         2.091905
## 9     dose_75          4      4     4.7621739         2.091905
## 10   dose_100          5      2     2.2284771         1.685100
```

```
### Both partial and complete responses combined:

 vismodegib %>% tally_study( response = 'PR_or_EOS_CR' )
```

```
##    group_name N_in_group first_day common_start last_day common_end PR_or_EOS_CR PR_or_EOS_CR_log_OR
## 1    dose_0.0          6         0         TRUE       21       TRUE            0           0.0000000
## 2    dose_0.3          5         0         TRUE       21      FALSE            0           0.1670541
## 3    dose_1.0          5         0         TRUE       21       TRUE            0           0.1670541
## 4    dose_3.0          5         0         TRUE       21       TRUE            0           0.1670541
## 5    dose_6.0          4         0         TRUE       21       TRUE            0           0.3677248
## 6     dose_10          4         0         TRUE       21       TRUE            2           2.5649494
## 7     dose_25          4         0         TRUE       21       TRUE            4           4.7621739
## 8     dose_50          4         0         TRUE       21       TRUE            4           4.7621739
## 9     dose_75          4         0         TRUE       21       TRUE            4           4.7621739
## 10   dose_100          5         0         TRUE       74      FALSE            5           4.9628446
##    PR_or_EOS_CR_SE_log_OR
## 1                0.000000
## 2                2.082226
## 3                2.082226
## 4                2.082226
## 5                2.091905
## 6                1.718676
## 7                2.091905
## 8                2.091905
## 9                2.091905
## 10               2.082226
```

```
 vismodegib %>%
 tally_study( response = 'PR_or_EOS_CR', PR_threshold = 0.80, EOS_CR_minval = 50, reference = 'dose_6.0' ) %>%
 dplyr::select( group_name, N_in_group, PR_or_EOS_CR, PR_or_EOS_CR_log_OR, PR_or_EOS_CR_SE_log_OR )
```

```
##    group_name N_in_group PR_or_EOS_CR PR_or_EOS_CR_log_OR PR_or_EOS_CR_SE_log_OR
## 1    dose_0.0          6            0           -3.412247               1.762449
## 2    dose_0.3          5            0           -3.245193               1.770367
## 3    dose_1.0          5            0           -3.245193               1.770367
## 4    dose_3.0          5            2           -1.183770               1.279881
## 5    dose_6.0          4            3            0.000000               0.000000
## 6     dose_10          4            4            1.349927               1.781742
## 7     dose_25          4            4            1.349927               1.781742
## 8     dose_50          4            4            1.349927               1.781742
## 9     dose_75          4            4            1.349927               1.781742
## 10   dose_100          5            5            1.550597               1.770367
```

```
### Multiple response options can be requested at once:

 vismodegib %>% tally_study( response = c('PR', 'EOS_CR') )
```

```
##    group_name N_in_group first_day common_start last_day common_end PR PR_log_OR PR_SE_log_OR EOS_CR
## 1    dose_0.0          6         0         TRUE       21       TRUE  0 0.0000000     0.000000      0
## 2    dose_0.3          5         0         TRUE       21      FALSE  0 0.1670541     2.082226      0
## 3    dose_1.0          5         0         TRUE       21       TRUE  0 0.1670541     2.082226      0
## 4    dose_3.0          5         0         TRUE       21       TRUE  0 0.1670541     2.082226      0
## 5    dose_6.0          4         0         TRUE       21       TRUE  0 0.3677248     2.091905      0
## 6     dose_10          4         0         TRUE       21       TRUE  2 2.5649494     1.718676      0
## 7     dose_25          4         0         TRUE       21       TRUE  4 4.7621739     2.091905      0
## 8     dose_50          4         0         TRUE       21       TRUE  4 4.7621739     2.091905      0
## 9     dose_75          4         0         TRUE       21       TRUE  4 4.7621739     2.091905      0
## 10   dose_100          5         0         TRUE       74      FALSE  5 4.9628446     2.082226      0
##    EOS_CR_log_OR EOS_CR_SE_log_OR
## 1      0.0000000         0.000000
## 2      0.1670541         2.082226
## 3      0.1670541         2.082226
```

```
## 4      0.1670541         2.082226
## 5      0.3677248         2.091905
## 6      0.3677248         2.091905
## 7      0.3677248         2.091905
## 8      0.3677248         2.091905
## 9      0.3677248         2.091905
## 10     0.1670541         2.082226
```

```
vismodegib %>%
 tally_study(
                  response = c( 'PR', 'EOS_CR', 'PR_or_EOS_CR'  ),
              PR_threshold = 0.80,
             EOS_CR_minval = 50,
                 reference = 'dose_6.0'
             ) %>%
 dplyr::select( group_name,
                N_in_group,
                PR,               PR_log_OR,           PR_SE_log_OR,
                EOS_CR,       EOS_CR_log_OR,       EOS_CR_SE_log_OR,
          PR_or_EOS_CR, PR_or_EOS_CR_log_OR, PR_or_EOS_CR_SE_log_OR
              )
```

```
##    group_name N_in_group PR  PR_log_OR PR_SE_log_OR EOS_CR EOS_CR_log_OR EOS_CR_SE_log_OR PR_or_EOS_CR
## 1    dose_0.0          6  0 -3.4122472     1.762449      0    -0.3677248         2.091905            0
## 2    dose_0.3          5  0 -3.2451931     1.770367      0    -0.2006707         2.098581            0
## 3    dose_1.0          5  0 -3.2451931     1.770367      0    -0.2006707         2.098581            0
## 4    dose_3.0          5  2 -1.1837701     1.279881      0    -0.2006707         2.098581            2
## 5    dose_6.0          4  3  0.0000000     0.000000      0     0.0000000         0.000000            3
## 6     dose_10          4  4  1.3499267     1.781742      0     0.0000000         2.108185            4
## 7     dose_25          4  4  1.3499267     1.781742      0     0.0000000         2.108185            4
## 8     dose_50          4  0 -3.0445224     1.781742      4     4.3944492         2.108185            4
## 9     dose_75          4  0 -3.0445224     1.781742      4     4.3944492         2.108185            4
## 10   dose_100          5  3 -0.5108256     1.279881      2     1.8607523         1.705267            5
##    PR_or_EOS_CR_log_OR PR_or_EOS_CR_SE_log_OR
## 1            -3.412247               1.762449
## 2            -3.245193               1.770367
## 3            -3.245193               1.770367
## 4            -1.183770               1.279881
## 5             0.000000               0.000000
## 6             1.349927               1.781742
## 7             1.349927               1.781742
## 8             1.349927               1.781742
## 9             1.349927               1.781742
## 10            1.550597               1.770367
```

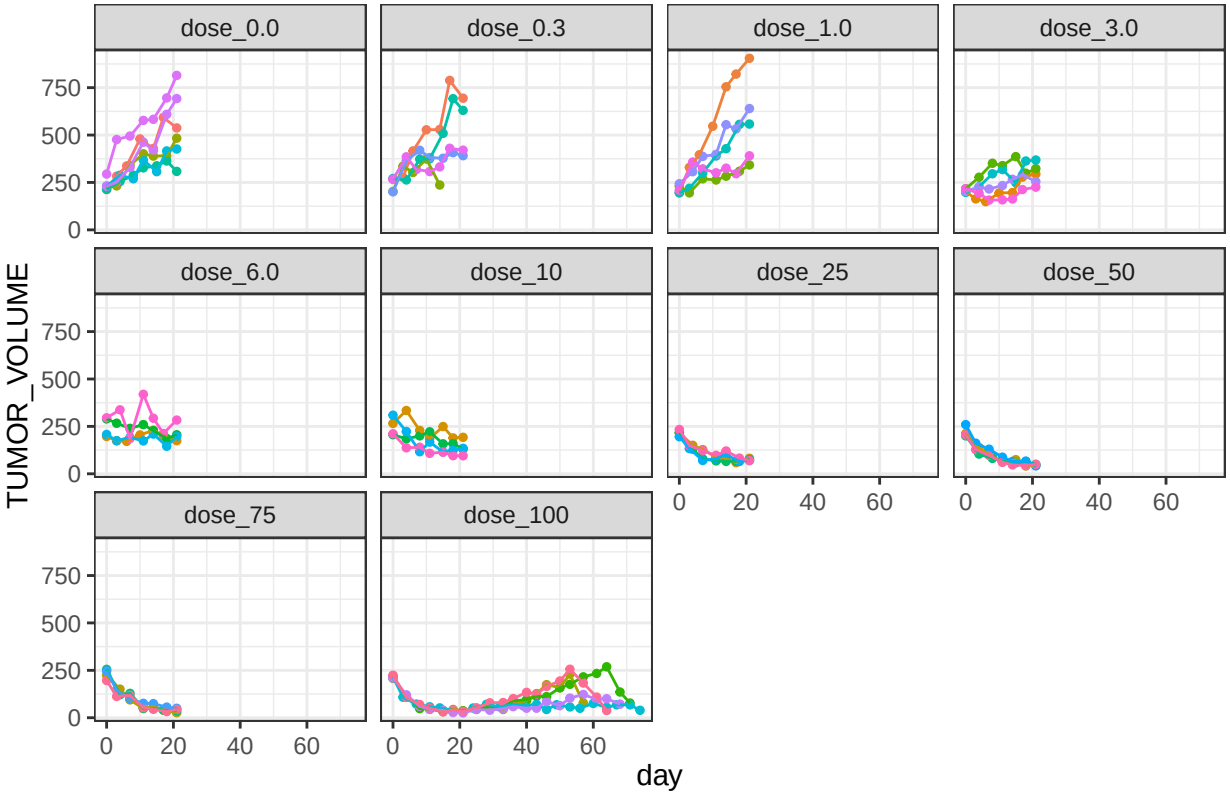
```
### Alternate endpoints:
###
### A different endpoint can be provided, but the onus of interpretation & good
### sense is on the user:

vismodegib %>%
tally_study( endpoint_name = 'BODY_WEIGHT', PR_threshold = 0.90, response = 'PR' )
```

```
##    group_name N_in_group first_day common_start last_day common_end PR PR_log_OR PR_SE_log_OR
## 1    dose_0.0          6         0         TRUE       21       TRUE  0 0.0000000     0.000000
## 2    dose_0.3          5         0         TRUE       21      FALSE  1 1.4663371     1.744344
## 3    dose_1.0          5         0         TRUE       21       TRUE  1 1.4663371     1.744344
## 4    dose_3.0          5         0         TRUE       21       TRUE  0 0.1670541     2.082226
## 5    dose_6.0          4         0         TRUE       21       TRUE  0 0.3677248     2.091905
## 6     dose_10          4         0         TRUE       21       TRUE  0 0.3677248     2.091905
## 7     dose_25          4         0         TRUE       21       TRUE  0 0.3677248     2.091905
## 8     dose_50          4         0         TRUE       21       TRUE  0 0.3677248     2.091905
## 9     dose_75          4         0         TRUE       21       TRUE  2 2.5649494     1.718676
## 10   dose_100          5         0         TRUE       74      FALSE  5 4.9628446     2.082226
```

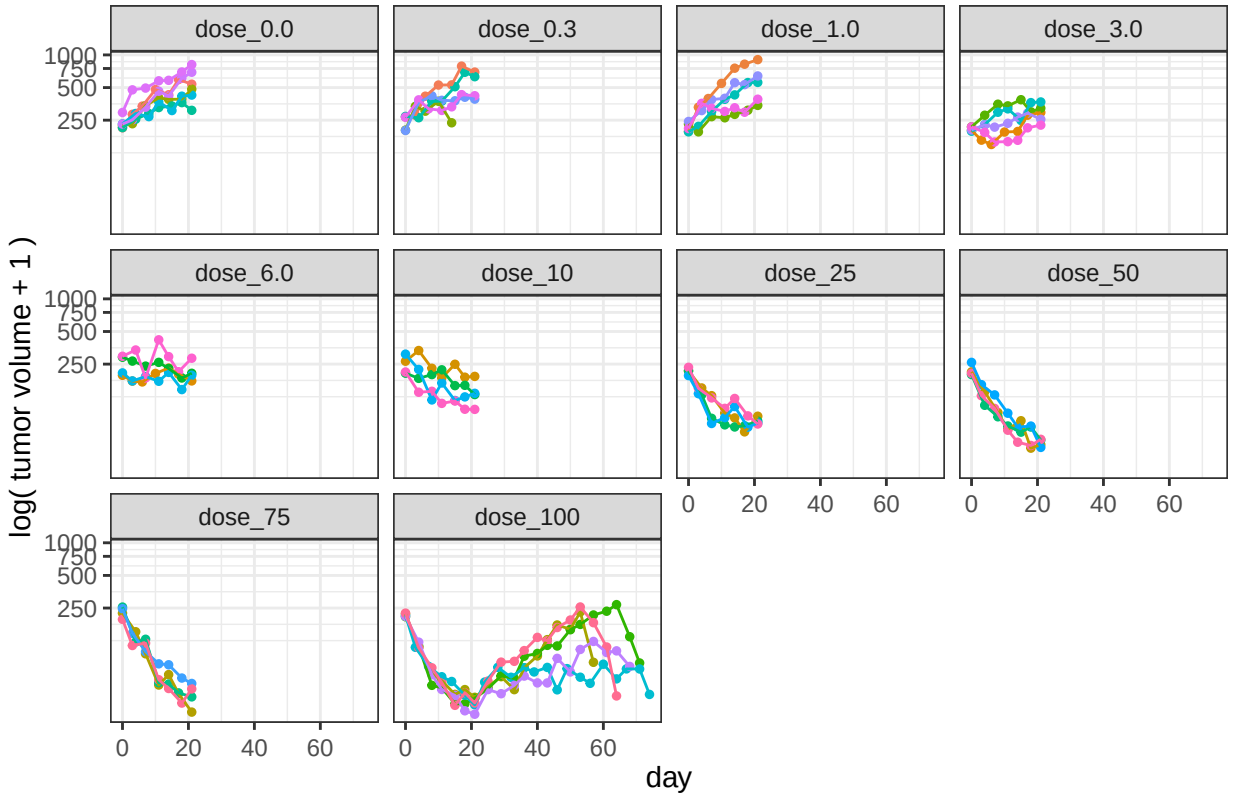
6 Pre-modeling pictures with draw_study() and draw_waterfall()

The functions `maeve::draw_study()` and `maeve::draw_waterfall()` generate pictures that do not require any statistical modeling (though a waterfall plot does display some *ad hoc* summary statistics). The `draw_study()` function will be revisited later to include modeling fits with the raw data. Both produce `ggplot` objects, and so can be augmented with additional layers in the usual ways.

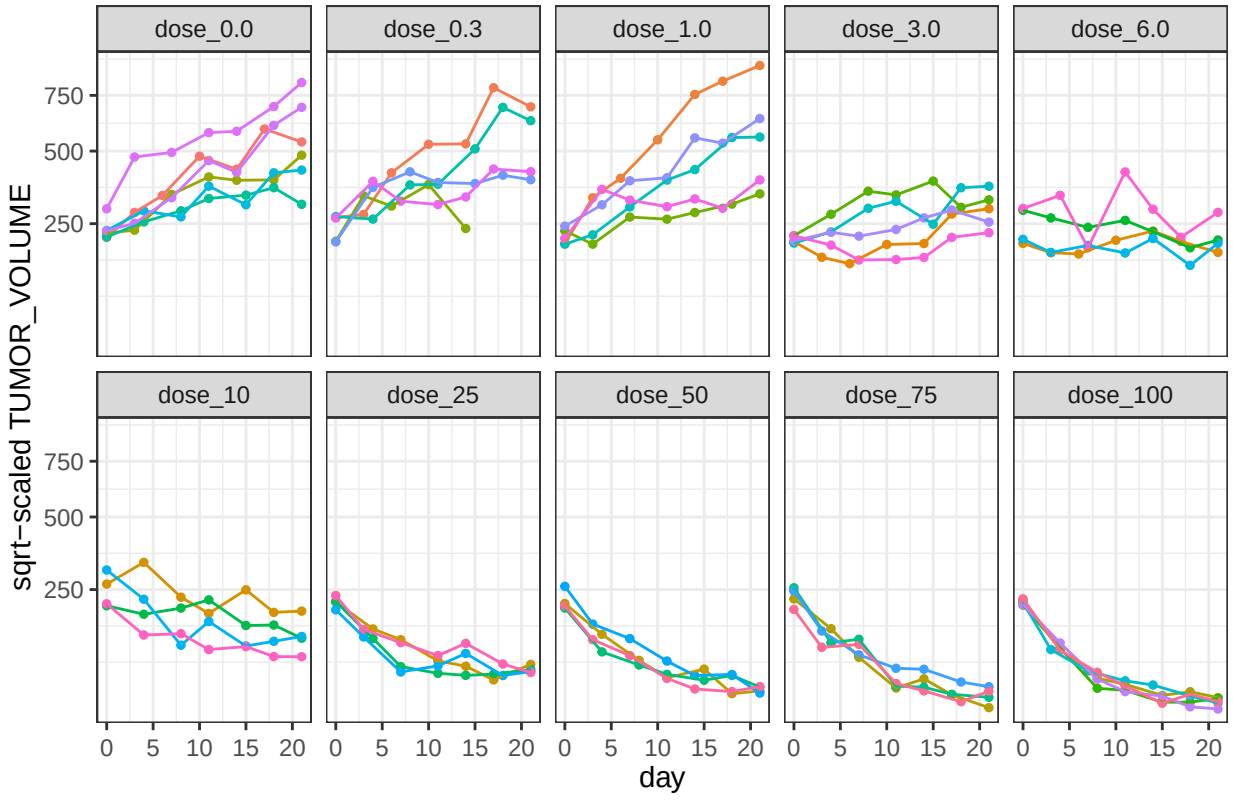
```
### draw_study(), basics

vismodegib %>% draw_study( y_label = 'TUMOR_VOLUME' )
```



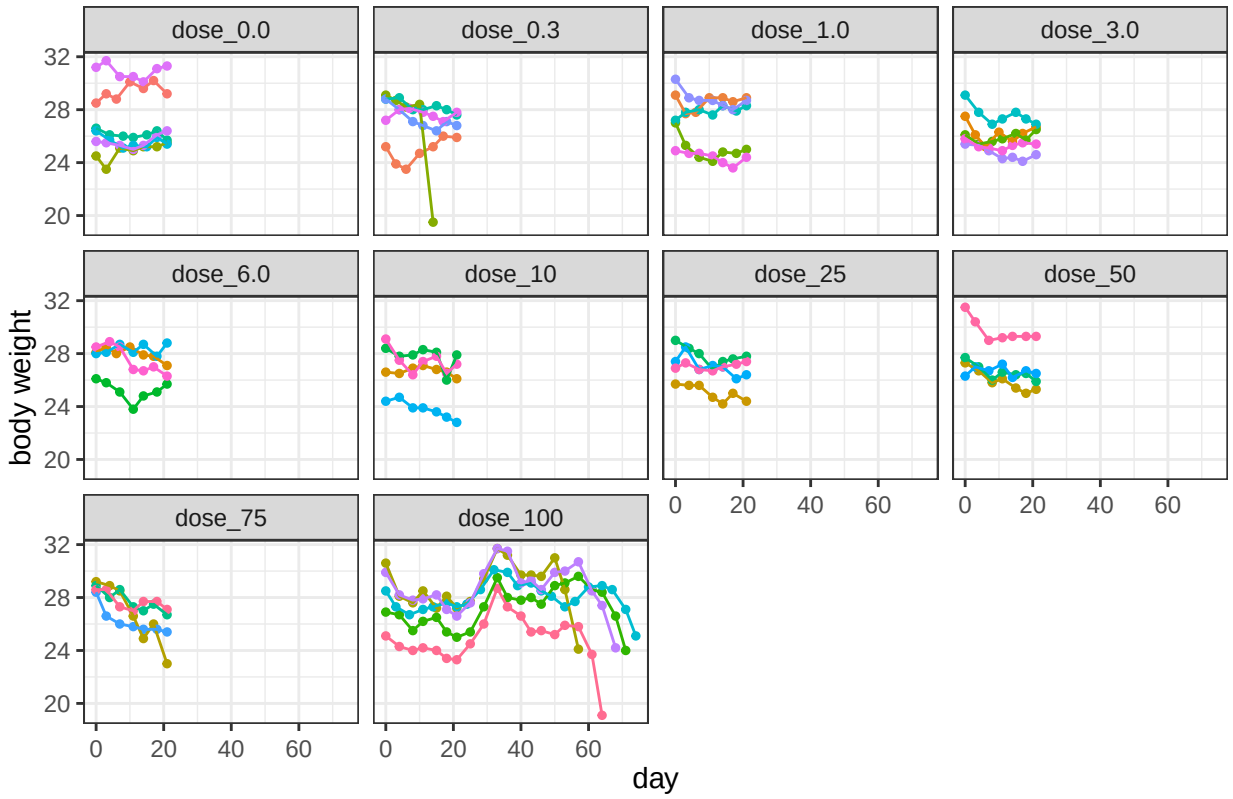

```
vismodegib %>% draw_study( ) + scale_y_continuous( trans = "log1p" )
```



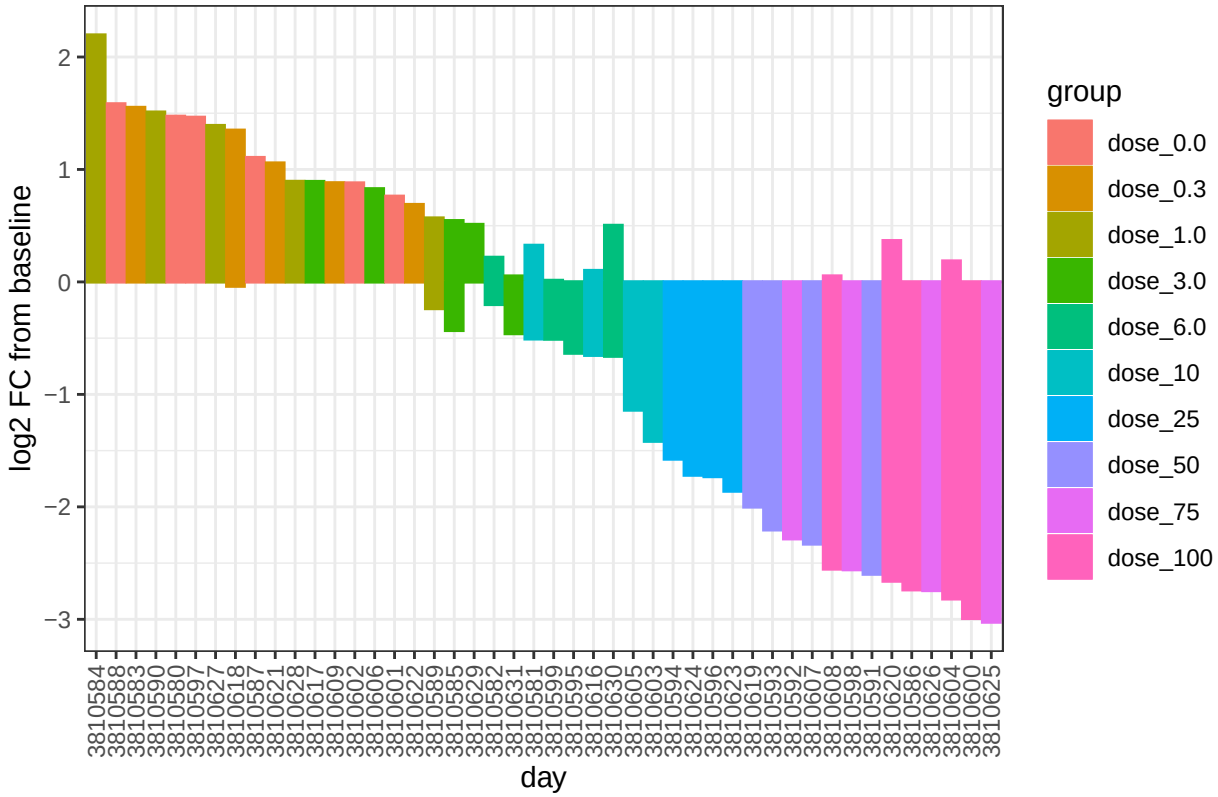

```
vismodegib %>%
    dplyr::filter( DAY_OF_STUDY <= 21 ) %>%
    draw_study( ncol_value = 5, y_label = 'sqrt-scaled TUMOR_VOLUME' ) + scale_y_sqrt()
```



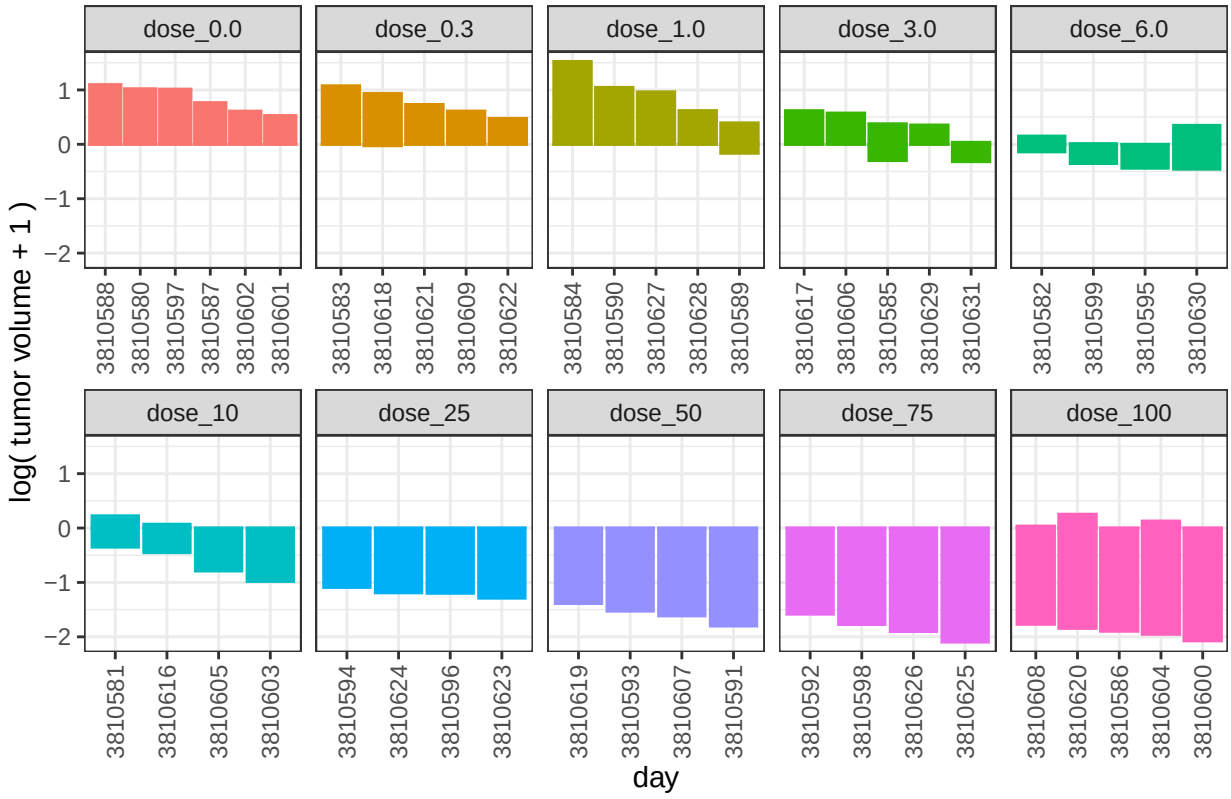
```
vismodegib %>% draw_study( endpoint_name = 'BODY_WEIGHT', y_label = 'body weight')
```




```
### draw_waterfall(), example

vismodegib %>%
 dplyr::mutate( log2_TV = log2( 1 + TUMOR_VOLUME ) ) %>%
 draw_waterfall( legend_position_char = 'right',
                         endpoint_name = 'log2_TV',
                               y_label = 'log2 FC from baseline'
                 )
```




```
vismodegib %>%
 dplyr::mutate( y = log( 1 + TUMOR_VOLUME ) ) %>%
 draw_waterfall( endpoint_name = 'y', facet_char = 'group_name', ncol_value = 5 )
```



7 auc_by_id() and summarize_by_id(): pre-modeling by-ID summaries

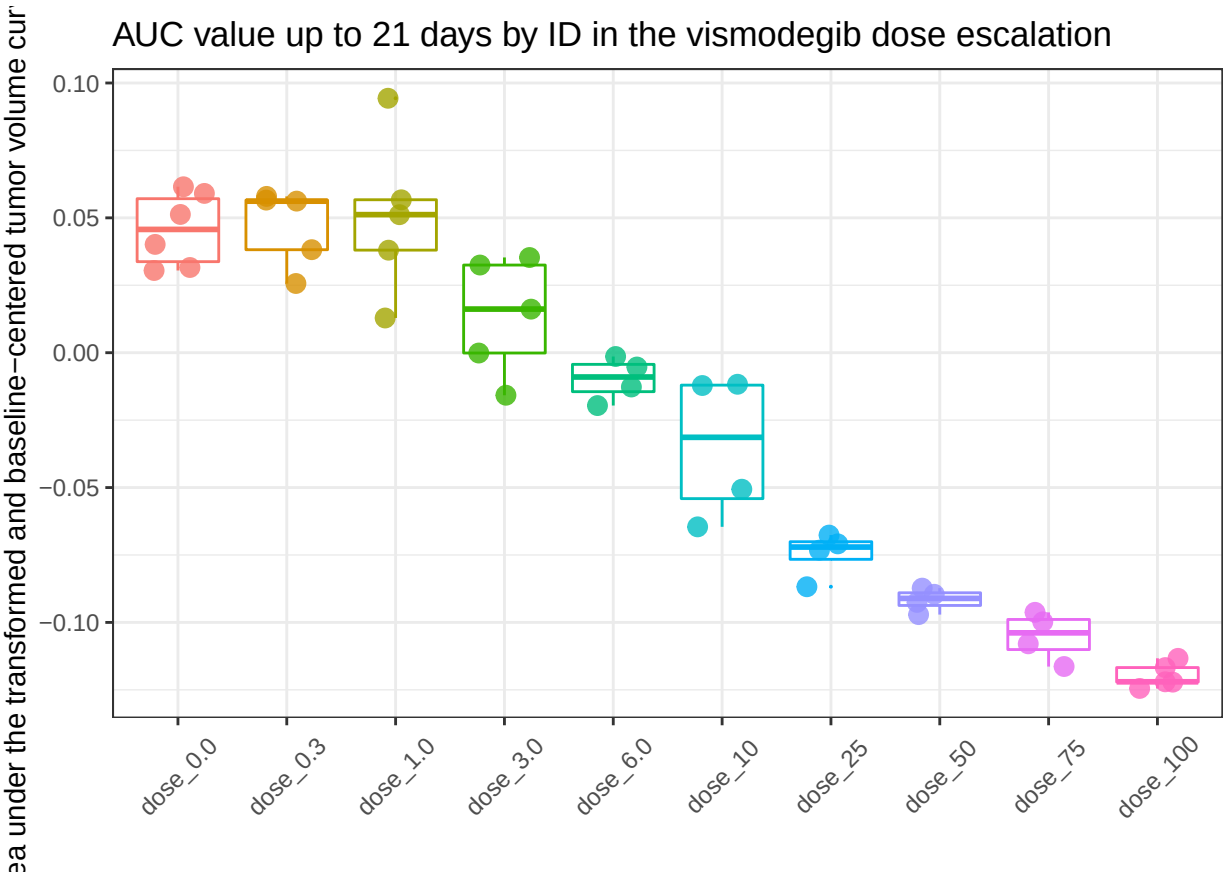
The function `maeve::auc_by_id()` will compute an area-under-curve (AUC) statistic to distill the `(x,y)` data on each separate subject identifier into a single value without any explicit modeling. The function `maeve::summarize_by_id()` generalizes this to compute within-ID versions of slope or empirical estimates of the spline summaries from the metrics `ITGR` or `AUC` (i.e., `eDOT` or `eGaIT`, respectively, with the default parameter settings).

By default, values are variance-stabilized and centered for each subject prior to summarization.

```
maeve_reset()

## restrict vismodegib dose escalation to to first 21 days:
vismo21 <- vismodegib %>% dplyr::filter( DAY_OF_STUDY <= 21 ) %>% droplevels()

## auc_by_id(), basics
vismo21_AUC_TV <- vismo21 %>% auc_by_id() %>% dplyr::rename( AUC_TV = AUC )

head( vismo21_AUC_TV, 7 )
```

```
##   group_name animalID first_day last_day     AUC_TV
## 1   dose_0.0  3810580         0       21 0.06154936
## 2   dose_0.0  3810587         0       21 0.04013998
## 3   dose_0.0  3810601         0       21 0.03163079
## 4   dose_0.0  3810602         0       21 0.03048146
## 5   dose_0.0  3810588         0       21 0.05127813
## 6   dose_0.0  3810597         0       21 0.05903663
## 7   dose_0.3  3810583         0       21 0.05618604
```

```
fig_AUC_by_group <-
vismo21_AUC_TV %>%
ggplot( aes( x = group_name, y = AUC_TV, colour = group_name ) ) +
geom_boxplot( outlier.size = 0 ) +
geom_point( position = position_jitter( width = 0.25 ), size = 3, alpha = 0.80 ) +
labs( x = '',
      y = 'Area under the transformed and baseline-centered tumor volume curves, by ID',
      title = 'AUC value up to 21 days by ID in the vismodegib dose escalation'
     ) +
theme( legend.position = 'none', axis.text.x = element_text( angle = 45, vjust = 0.5 ) )

print( fig_AUC_by_group )
```




```
### similar calculations with summarize_by_id():
vismo21_TV_3_metrics <-
    vismodegib %>%
    summarize_by_id( metric = c('linear', 'ITGR', 'AUC'), xmin = 0, xmax = 21 )

head( format( vismo21_TV_3_metrics, digits = 3 ), 8 )
```

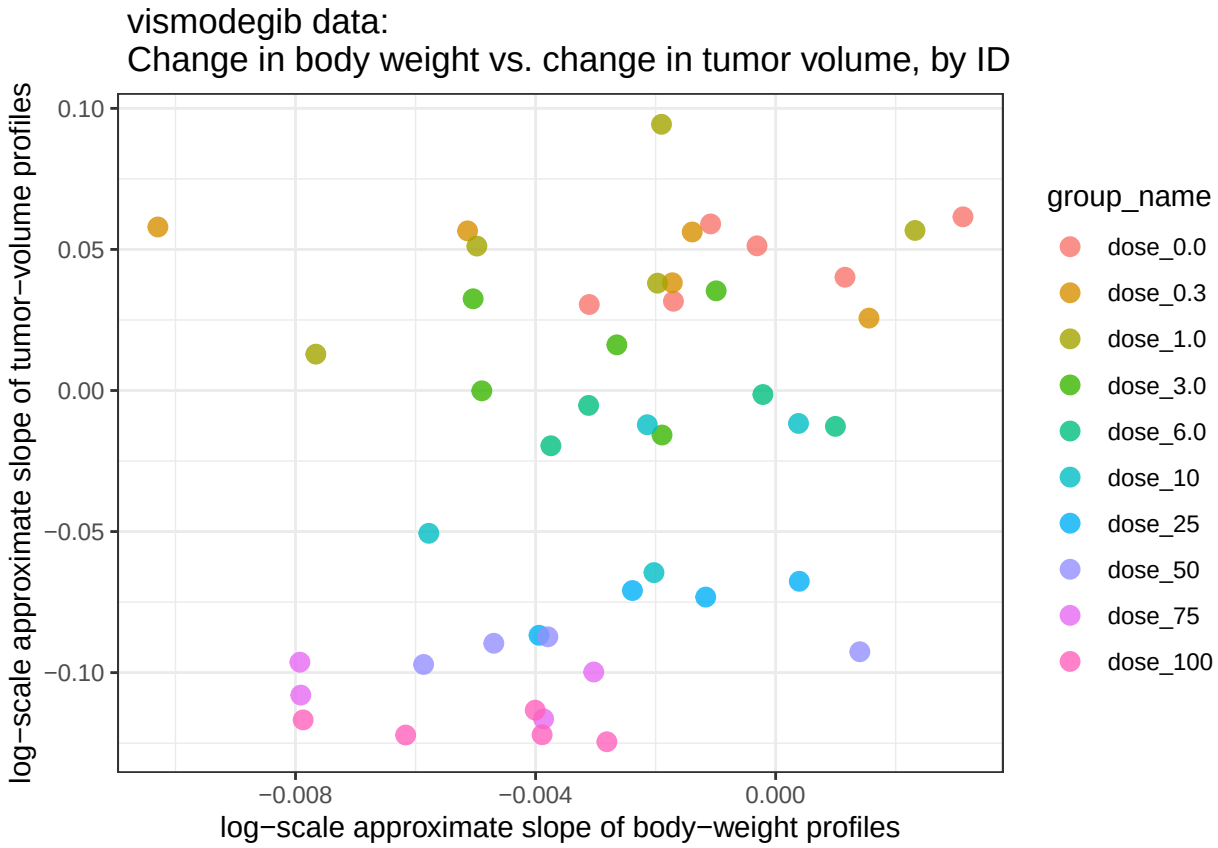
```
##   group_name animalID first_day last_day   linear     ITGR      AUC
## 1   dose_0.0  3810580         0       21  0.04473  0.04400  0.06155
## 2   dose_0.0  3810587         0       21  0.03497  0.03656  0.04014
## 3   dose_0.0  3810601         0       21  0.02051  0.01741  0.03163
## 4   dose_0.0  3810602         0       21  0.02717  0.02910  0.03048
## 5   dose_0.0  3810588         0       21  0.05331  0.05232  0.05128
## 6   dose_0.0  3810597         0       21  0.03945  0.04837  0.05904
## 7   dose_0.3  3810583         0       21  0.05175  0.04524  0.05619
## 8   dose_0.3  3810609         0       14  0.00953  0.01111  0.05795
```

```
### auc_by_id(): computing by-ID averages over time:
###
### By changing parameters, the AUC will instead become the time-interval-weighted
### average over the observed range:
    vismo21 %>%
    maeve::auc_by_id(
                endpoint_name = 'BODY_WEIGHT',
                          VST = FALSE,
      subtract_starting_value = FALSE,
           xrange_norm_method = 'xrange'
                ) %>%
      dplyr::rename( AUC_BW_avg = AUC ) %>%
```

```
      head( 10 )
```

```
##    group_name animalID first_day last_day AUC_BW_avg
## 1    dose_0.0  3810580         0       21   29.48810
## 2    dose_0.0  3810587         0       21   24.81905
## 3    dose_0.0  3810601         0       21   26.11190
## 4    dose_0.0  3810602         0       21   25.52381
## 5    dose_0.0  3810588         0       21   25.51667
## 6    dose_0.0  3810597         0       21   30.84048
## 7    dose_0.3  3810583         0       21   24.83571
## 8    dose_0.3  3810609         0       14   27.21786
## 9    dose_0.3  3810618         0       21   28.26905
## 10   dose_0.3  3810621         0       21   27.24524
```

```
### auc_by_id(): Bivariate changes to tumor volume and body weight, by ID:

  ## Get the approximate slope (in grams per day) of body weight change
  vismo21_AUC_BW_slope <-
    vismo21 %>%
    maeve::auc_by_id(endpoint_name = 'BODY_WEIGHT') %>%
    dplyr::rename( AUC_BW_slope = AUC )

  merge_cols <- c("group_name", "animalID", "first_day", "last_day" )

  vismo21_TV_BW <- dplyr::full_join( vismo21_AUC_TV, vismo21_AUC_BW_slope, by = merge_cols )

  maeve_reset()

  fig_TV_vs_BW_by_ID <-
  vismo21_TV_BW %>%
  ggplot( aes( x = AUC_BW_slope, y = AUC_TV, colour = group_name ) ) +
  geom_point( size = 3, alpha = 0.80 ) +
  labs( x = 'log-scale approximate slope of body-weight profiles',
        y = 'log-scale approximate slope of tumor-volume profiles',
        title = ' vismodegib data:\n Change in body weight vs. change in tumor volume, by ID'
       )

  print( fig_TV_vs_BW_by_ID )
```



8 `window_by_id()`: Interpolate-by-ID on an interval.

The function `maeve::window_by_id()` accepts a data.frame with columns for `group_name`, `subject_ID`, `x_name`, and `endpoint_name`. By default, it uses the names for these four columns from the current values in `maeve_options()`. It also accepts parameters `xmin` and `xmax`, with defaults of `-Inf` and `Inf`. The function will:

1. filter out `x`-values that do *not* fall within this range.
2. add `x`-values and interpolated `y`-values for longitudinal traces that span either endpoint. The method for interpolation is one of

a. 'trapezoid' / 'linear' (these are synonyms in this function) uses a linear approximation from `stats::approxfun( ..., method = 'linear', ...)`.
b. 'step' uses a left-continuous step-function from `stats::approxfun( ..., method = 'constant', f = 0, ...)`.
c. 'spline' uses a spline from `stats::splinefun( ..., method = "fmm", ...)`.

3. return only the 4 columns named by `group_name`, `subject_ID`, `x_name`, and `endpoint_name`. If additional columns are returned (i.e., `return_all_columns = TRUE`), then the function sets numeric fields that are **not** the `x_name` or `endpoint_name` to `NA` for any rows with interpolated values.

The default behavior of window_by_id() is to transform the endpoint (i.e., using `maeve_options("trans_func_char")` and `maeve_options("add_to_endpoint")`) then do any interpolation at the endpoints for (`xmin`,`xmax`) on the transformed scale, then back-transform the window-ed and any interpolated values. See this example:

```
  maeve_reset()

  print( maeve_options('trans_func_char') )
```

```
## [1] "log"
```

```
  print( maeve_options('add_to_endpoint') )
```

```
## [1] 1
```

```
### restrict vismodegib dose escalation to days in the interval [5, 15]
### in two different ways.


### Method #1:
  vismo_5_to_15_V01 <-
  vismodegib %>%
  dplyr::mutate( y = log( 1 + TUMOR_VOLUME ) ) %>%
  ## window [5, 15]:
  ## 'VST = FALSE' below, so no variance-stabilizing transform is applied.
  ## Instead, we take the already-variance-stabilized 'y' just created,
  ## interpolate it at 'x = 5' and 'x = 15', then back-transform to get
  ## the 'TUMOR_VOLUME' values at the interval endpoints.
  maeve::window_by_id( xmin = 5, xmax = 15, endpoint_name = 'y', VST = FALSE ) %>%
  dplyr::mutate( TUMOR_VOLUME = exp( y ) - 1 ) %>%
  dplyr::select( group_name, animalID, DAY_OF_STUDY, TUMOR_VOLUME, y )

  head( vismo_5_to_15_V01, 7 )
```

```
##   group_name animalID DAY_OF_STUDY TUMOR_VOLUME        y
## 1   dose_0.0  3810580            5     318.0052 5.765207
## 2   dose_0.0  3810580            6     337.0389 5.823161
## 3   dose_0.0  3810580           10     479.5100 6.174848
## 4   dose_0.0  3810580           14     429.2777 6.064431
## 5   dose_0.0  3810580           15     478.0963 6.171902
## 6   dose_0.0  3810587            5     280.9335 5.641671
## 7   dose_0.0  3810587            7     340.4171 5.833105
```

```
### Method #2:
  ## A shorter way is to just send 'TUMOR_VOLUME' to 'window_by_id()' and
  ## then pipe the output through dplyr::mutate() and create 'y' there.
  vismo_5_to_15_V02 <-
  vismodegib %>%
  maeve::window_by_id( xmin = 5, xmax = 15 ) %>%
  dplyr::mutate( y = log( 1 + TUMOR_VOLUME ) )

  head( vismo_5_to_15_V02, 7 )
```

```
##   group_name animalID DAY_OF_STUDY TUMOR_VOLUME        y
## 1   dose_0.0  3810580            5     318.0052 5.765207
## 2   dose_0.0  3810580            6     337.0389 5.823161
## 3   dose_0.0  3810580           10     479.5100 6.174848
## 4   dose_0.0  3810580           14     429.2777 6.064431
## 5   dose_0.0  3810580           15     478.0963 6.171902
## 6   dose_0.0  3810587            5     280.9335 5.641671
## 7   dose_0.0  3810587            7     340.4171 5.833105
```

Observe that the interpolated values at the interval end points match. This is *not* guaranteed to be true if the `"spline"` method is set for interpolation (which is arguably a better approach), but it should be close.

The advantage of the `window_by_id()` function over simply filtering / subsetting the data is that it will

automatically interpolate the endpoint values onto a common x-range. It will never *extrapolate* longitudinal traces out to interval endpoints, however.

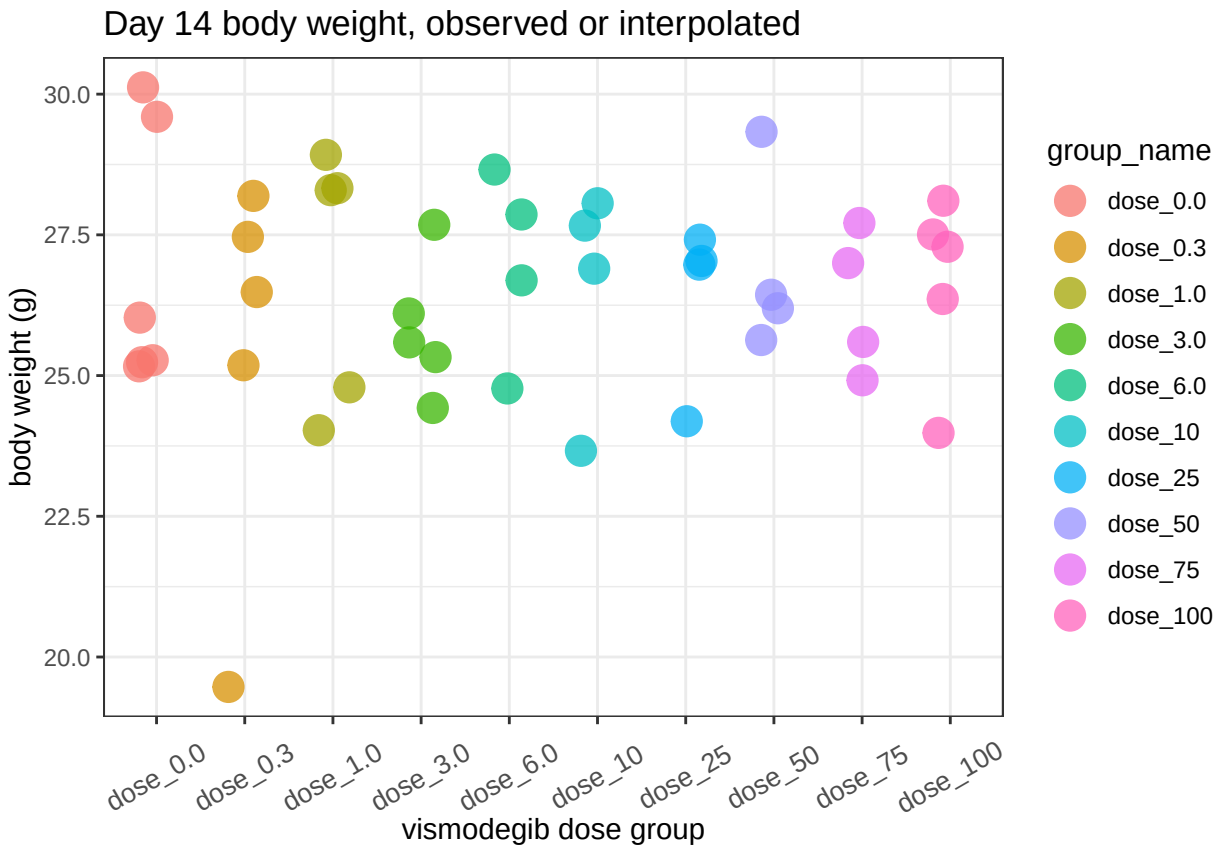
If we want an endpoint evaluated at a given time for all subjects, we can get a value by restricting the data to a very narrow window (perhaps with interpolation at the window endpoints), then calculating the AUC over that interval and dividing by the x-range. This is true even if the subject does not have a value within that interval, so long as there are values bracketing the interval on either side and the interpolation method is adjudged acceptable.

As an example, suppose we want to assess body weight for each animal in the vismodegib study at day 14 (this is the last day at which all animals are still on study). Note that only 30 of the 46 animals have body weight recorded on day 14, but those *without* readings on that day all have bracketing measurements. If we are willing to interpolate these bracketing weights for an animal in order to estimate its body weight on day 14, then we can get an observation for each animal at a common time as follows:

```
maeve_reset()
maeve_options( endpoint_name = 'BODY_WEIGHT',
               xrange_norm_method = 'xrange'
             )

x_delta <- 5e-4 # interval radius about common day.

BW_common_day <-
vismodegib %>%
maeve::window_by_id( xmin = common_day - x_delta, xmax = common_day + x_delta, print_warnings = FALSE ) %>%
maeve::auc_by_id( VST = FALSE, subtract_starting_value = FALSE ) %>%
dplyr::select( -first_day, -last_day ) %>%
dplyr::rename( BW_common_day = AUC ) %>%
maeve::round_numerics( 1 ) # keep measurements at same recorded precision

head( BW_common_day, 10 )
```

```
##    group_name animalID BW_common_day
## 1    dose_0.0  3810580          29.6
## 2    dose_0.0  3810587          25.2
## 3    dose_0.0  3810601          26.0
## 4    dose_0.0  3810602          25.2
## 5    dose_0.0  3810588          25.3
## 6    dose_0.0  3810597          30.1
## 7    dose_0.3  3810583          25.2
## 8    dose_0.3  3810609          19.5
## 9    dose_0.3  3810618          28.2
## 10   dose_0.3  3810621          26.5
```

```
fig_title <- paste( 'Day', common_day, 'body weight, observed or interpolated' )

fig_common_day <-
BW_common_day %>%
ggplot() +
aes( x = group_name, y = BW_common_day, colour = group_name ) +
geom_point( size = 5, alpha = 0.75, position = position_jitter( width = 0.2 ) ) +
labs( title = eval( fig_title ), y = 'body weight (g)', x = 'vismodegib dose group') +
theme( axis.text.x = element_text( angle = 30, size = 10, vjust = .4 ) )

print( fig_common_day )
```



In short, the `window_by_id()` function paired with `auc_by_id()` can be used for a number of common requests, with minimal statistical modeling assumptions.

9 `grid_by_trace()`: Interpolate-by-trace on a grid.

The `grid_by_trace()` function allows for interpolation of one or more longitudinal series per subject onto a common set of grid points. This function uses the `maeve_options('trace_ID')` parameter to track longitudinal series nested within subject.

For some applications (e.g., biomedical imaging, a.k.a. BMI), multiple longitudinal data sets can be collected for each of several subjects. We term each such data set nested wwithin a subject identifer a *trace*. Two quick examples:

1. A mouse is treated with vehicle, then image intensity recorded for an organ at finely-spaced time points over an hour. After a washout period, the mouse is treated with an active molecule and the same kind of image intensity is recorded for the same organ for another hour. The two sets of time points from the two scans are not necessarily identical. The two scans are two *traces* on the same mouse, but with different treatments.

2. A mouse is simultaneously scanned on five different body tissues (e.g., blood, liver, brain, spleen, lung) on a fine grid of time points (not necessarily the same points, but they usually are). The five collections of longitudinal data are five *traces* on one mouse.

Often, then, scientists want to align and compare the BMI traces to derive some efficacy statistic, e.g., the "spleen-to-blood signal ratio". The function `maeve::grid_by_trace()` accepts a data.frame with columns for `group_name`, `trace_ID`, `x_name`, and `endpoint_name`. By default, it uses the names for these four columns

from the current values in `maeve_options()`. It also accepts parameters `xmin` and `xmax`, with defaults of `-Inf` and `Inf`. The function will:

1. Identify a range of `x`-values that includes *all* the traces in the data set.
2. Define a grid on the full `x`-range that is the union of the observed `x`-values and a finely spaced grid.
3. Interpolate `y`-values for each longitudinal trace separately at the `x`-values for which that trace does not already have a value. The method for interpolation is one of
    a. 'trapezoid' / 'linear' (these are synonyms in this function) uses a linear approximation from `stats::approxfun( ..., method = 'linear', ...)`.
    b. 'step' uses a left-continuous step-function from `stats::approxfun( ..., method = 'constant', f = 0, ...)`.
    c. 'spline' uses a spline from `stats::splinefun( ..., method = "fmm", ...)`.
4. Return only the 4 columns named by `group_name`, `trace_ID`, `x_name`, and `endpoint_name`. If additional columns are returned (i.e., `return_all_columns = TRUE`), then the function sets numeric fields that are **not** the `x_name` or `endpoint_name` to `NA` for any rows with interpolated values.

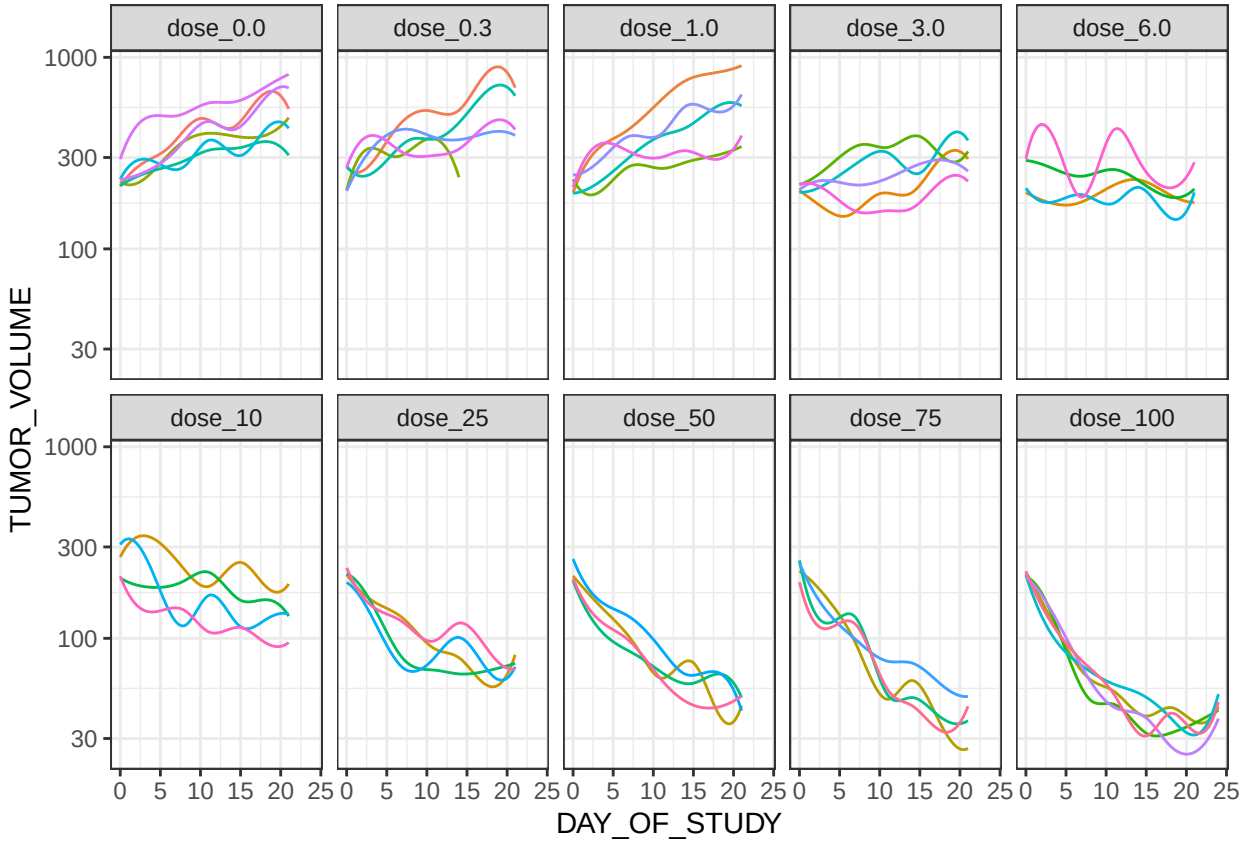
The default behavior of window_by_id() is to transform the endpoint (i.e., using `maeve_options("trans_func_char")` and `maeve_options("add_to_endpoint")`) then do any interpolation at the endpoints for (`xmin`,`xmax`) on the transformed scale, then back-transform the window-ed and any interpolated values. See this example:

```
maeve_reset()
maeve_options( trace_ID = 'animalID', endpoint_name = 'TUMOR_VOLUME' )

vismo_trace <-
    vismodegib %>%
    maeve::window_by_id( xmin = 0, xmax = 24 ) %>%
    maeve::grid_by_trace( method = 'spline' )

head( vismo_trace, 15 ) # a few lines
```

```
##    group_name animalID DAY_OF_STUDY         type TUMOR_VOLUME
## 1    dose_0.0  3810580         0.00     observed     212.7522
## 2    dose_0.0  3810580         0.24 interpolated     219.9040
## 3    dose_0.0  3810580         0.48 interpolated     226.8233
## 4    dose_0.0  3810580         0.72 interpolated     233.4942
## 5    dose_0.0  3810580         0.96 interpolated     239.9035
## 6    dose_0.0  3810580         1.20 interpolated     246.0406
## 7    dose_0.0  3810580         1.44 interpolated     251.8982
## 8    dose_0.0  3810580         1.68 interpolated     257.4714
## 9    dose_0.0  3810580         1.92 interpolated     262.7586
## 10   dose_0.0  3810580         2.16 interpolated     267.7604
## 11   dose_0.0  3810580         2.40 interpolated     272.4804
## 12   dose_0.0  3810580         2.64 interpolated     276.9246
## 13   dose_0.0  3810580         2.88 interpolated     281.1013
## 14   dose_0.0  3810580         3.00     observed     283.0926
## 15   dose_0.0  3810580         3.12 interpolated     285.0222
```

```
vismo_trace_fig <-
    vismo_trace %>%
    dplyr::filter( !is.na( TUMOR_VOLUME ) ) %>%
    dplyr::mutate( color_order = maeve::splice_color_order( group_name, animalID ) ) %>%
    ##
    ggplot() +
    aes( x = DAY_OF_STUDY, y = TUMOR_VOLUME, group = animalID, colour = color_order ) +
    geom_line() +
    facet_wrap( ~ group_name, nrow = 2 ) +
    scale_y_log10() +
```

```
    guides( colour = FALSE ) # do not show a legend coloured by animalID

print( vismo_trace_fig )
```



This first example has only one trace for each animal, though two or more are possible by redefining `trace_ID`. Note that `grid_by_trace()` will *create* the traces on a common grid, but will not do anything to sort or compare them. This is intentional. Downstream steps are potentially wide-ranging right now, so the current function's scope is limited.

10 Truncating study range with `truncate_study()`

(This section is optional and can be skipped on a first reading)

Truncation in `maeve` refers to excluding data from certain `x`-values *before* modeling is performed. For example, if a group of ten subjects has repeated measures over time, with nine subjects stopping on day 20, then the tenth subject's data continuing out to, say, day 50, we might choose to model the group's average response only out to day 20. The truncated data is not by default removed from the data set. Rather, it remains in the data, but is excluded from the data passed internally to model.

Truncation is performed through the function `truncate_study()`, which is called internally in `model_study()` (below), but can also be called separately if desired. Behavior of the truncation is performed through a number of package-level parameters available in `maeve_options()`:

1. `min_n_in_group`: Data for a group are truncated from model fitting at `x`-values beyond which the number of subjects drops below `min_n_in_group`. E.g., when `min_n_in_group = 3` for a longitudinal study data are used only until last time point with three subjects.

2. `min_frac_in_group` and `min_frac_in_study`: These two parameters work in tandem.
    a. Data for a group are truncated when the fraction of subjects left in the group is below `min_frac_in_group` and the fraction of subjects left in the entire study is below `min_frac_in_study`.
    b. Example: If `min_frac_in_group = 0.50` and `min_frac_in_study = 0.30`, then data from the group will be used until over 50% of the subjects from the group have dropped out and over 70% of the subjects from the entire study have dropped out.
3. `overall_x_min`: This sets a numeric lower bound below which `x`-values are not used in modelling. By default it is `-Inf`, so that no truncation occurs
4. `overall_x_max`: This sets a numeric upper bound above which `x`-values are not used in modelling. By default it is `Inf`, so that no truncation occurs.

The values of `overall_x_min` & `overall_x_max` are applied last, and so further extend any truncation done by earlier parameters. They will not "undo" truncation implemented by `min_n_in_group`, `min_frac_in_group`, and `min_frac_in_study`.

The function `maeve::truncate_study()` takes as input a `data.frame` with the study data, and has three distinct options for what is returned, depending on what the user wants to do with the output. These options are determined by `maeve_options("truncation_return_type")`, which can take one of the values `"data.frame"`, `"logical"`, or `"list"`. The `"data.frame"` option returns the subset of the input data frame that is not truncated. The `"logical"` option returns the logical vector with length equal to the number of rows in the input `data.frame` and `TRUE` for rows that are not truncated. The `"list"` option returns these two in addition to much more information.

```
### truncate_study() examples

maeve_reset()

## maeve package defaults:
vismo_all <- vismodegib %>% truncate_study()
identical( vismodegib, vismo_all ) %>% print # By default, there is no truncation.
```

```
## [1] TRUE
```

```
## gRED TO defaults:
vismo_trunc_v01 <-
    vismodegib %>% truncate_study( min_n_in_group = 3, min_frac_in_group = 0.50, min_frac_in_study = 0.30 )
identical( vismodegib, vismo_trunc_v01 ) %>% print # gRED default settings will truncate some data rows
```

```
## [1] FALSE
```

```
vismodegib      %>% nrow %>% print
```

```
## [1] 385
```

```
vismo_trunc_v01 %>% nrow %>% print
```

```
## [1] 381
```

```
vismo_trunc_v01 %>% tally_study()
```

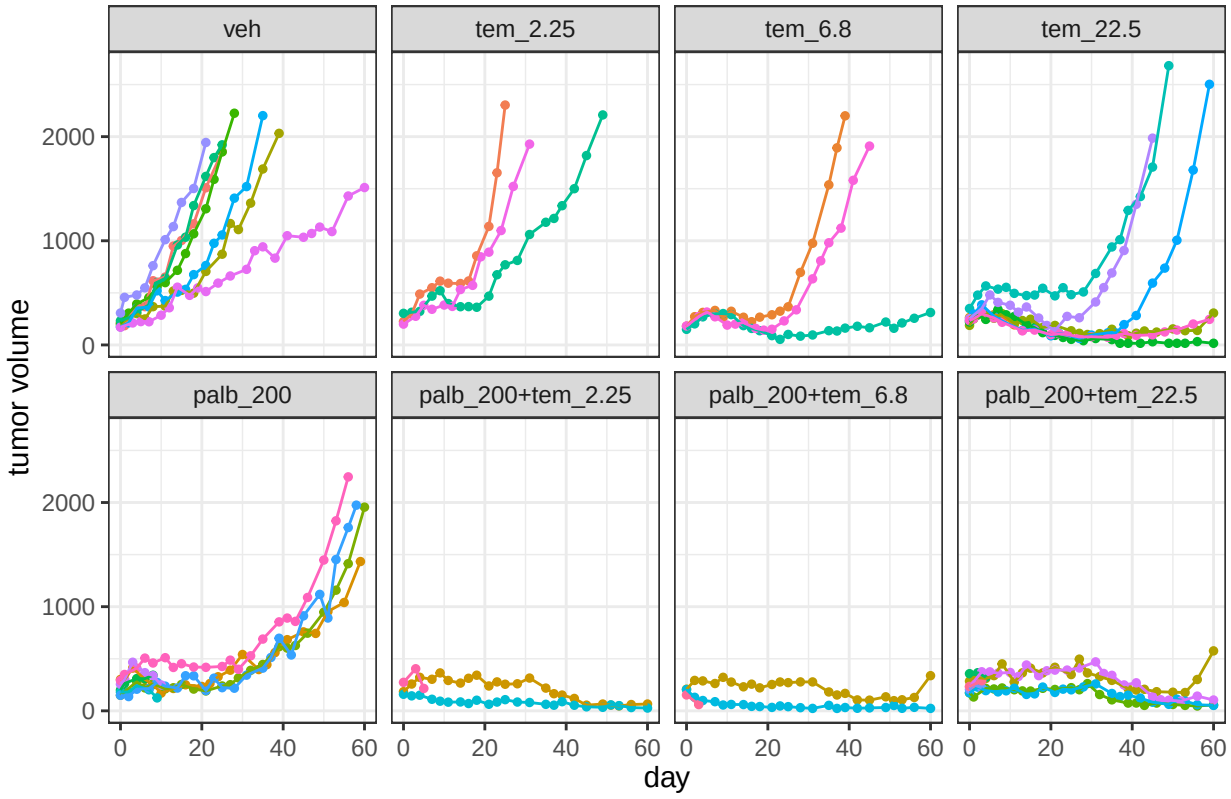
```
##    group_name N_in_group first_day common_start last_day common_end
## 1    dose_0.0          6         0         TRUE       21       TRUE
## 2    dose_0.3          5         0         TRUE       21      FALSE
## 3    dose_1.0          5         0         TRUE       21       TRUE
## 4    dose_3.0          5         0         TRUE       21       TRUE
## 5    dose_6.0          4         0         TRUE       21       TRUE
## 6     dose_10          4         0         TRUE       21       TRUE
## 7     dose_25          4         0         TRUE       21       TRUE
```

```
## 8     dose_50          4         0         TRUE       21       TRUE
## 9     dose_75          4         0         TRUE       21       TRUE
## 10   dose_100          5         0         TRUE       68      FALSE
```

```
## gRED TO defaults:
maeve_reset()
maeve_options( min_n_in_group    = 3,
               min_frac_in_group = 0.50,
               min_frac_in_study = 0.30,
               truncation_return_type = 'data.frame' # one of 'data.frame', 'logical', or 'list'.
              )

data( palb_temo_01, package = 'maeve')

maeve_options( ncol_value = 4, y_label = 'tumor volume' ) # just for these palb+temo 01 plots.

# palbociclib + temozolomide study 01 with no truncation at all
palb_temo_01 %>% tally_study()
```

```
##          group_name N_in_group first_day common_start last_day common_end
## 1               veh          7         0         TRUE       60      FALSE
## 2          tem_2.25          3         0         TRUE       49      FALSE
## 3           tem_6.8          3         0         TRUE       60      FALSE
## 4          tem_22.5          7         0         TRUE       60      FALSE
## 5          palb_200          7         0         TRUE       60      FALSE
## 6 palb_200+tem_2.25          3         0         TRUE       60      FALSE
## 7  palb_200+tem_6.8          3         0         TRUE       60      FALSE
## 8 palb_200+tem_22.5          7         0         TRUE       60      FALSE
```

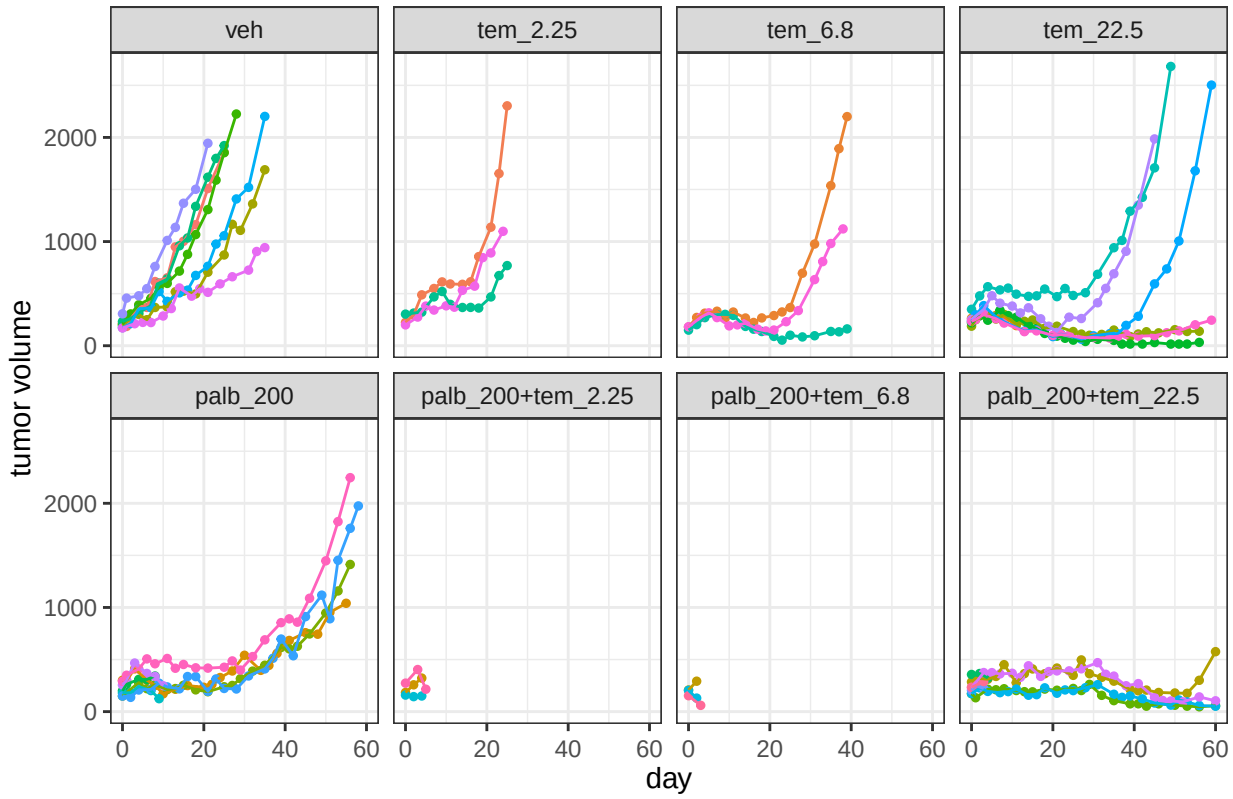
```
palb_temo_01 %>% draw_study()
```




```
# palbociclib + temozolomide study 01 with gRED default truncation
palb_temo_01 %>% truncate_study() %>% tally_study()
```

```
##          group_name N_in_group first_day common_start last_day common_end
## 1               veh          7         0         TRUE       35      FALSE
## 2          tem_2.25          3         0         TRUE       25      FALSE
## 3           tem_6.8          3         0         TRUE       39      FALSE
## 4          tem_22.5          7         0         TRUE       59      FALSE
## 5          palb_200          7         0         TRUE       58      FALSE
## 6 palb_200+tem_2.25          3         0         TRUE        5      FALSE
## 7  palb_200+tem_6.8          3         0         TRUE        3      FALSE
## 8 palb_200+tem_22.5          7         0         TRUE       60      FALSE
```

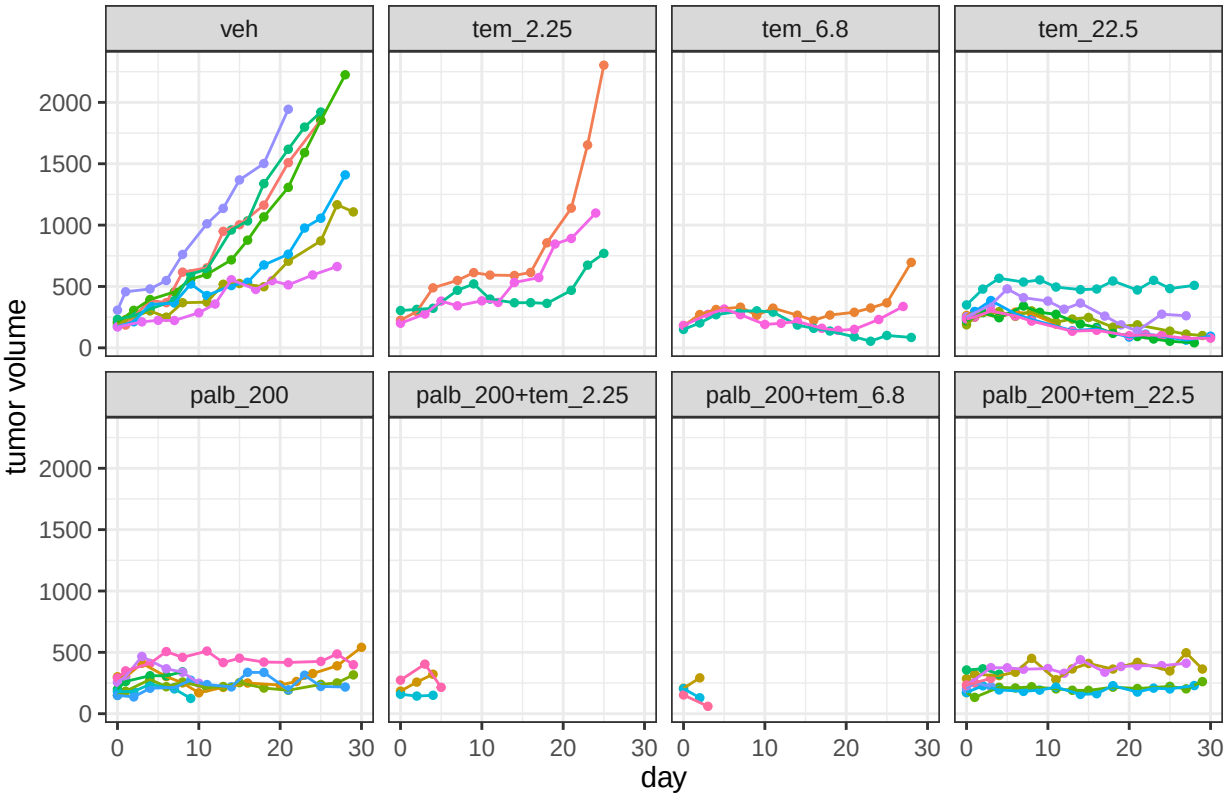
```
palb_temo_01 %>% truncate_study() %>% draw_study()
```




```
# palbociclib + temozolomide study 01 with gRED default truncation + hard truncation at 30 days:
palb_temo_01 %>% truncate_study(overall_x_max = 30) %>% tally_study()
```

```
##          group_name N_in_group first_day common_start last_day common_end
## 1               veh          7         0         TRUE       29      FALSE
## 2          tem_2.25          3         0         TRUE       25      FALSE
## 3           tem_6.8          3         0         TRUE       28      FALSE
## 4          tem_22.5          7         0         TRUE       30      FALSE
## 5          palb_200          7         0         TRUE       30      FALSE
## 6 palb_200+tem_2.25          3         0         TRUE        5      FALSE
## 7  palb_200+tem_6.8          3         0         TRUE        3      FALSE
## 8 palb_200+tem_22.5          7         0         TRUE       29      FALSE
```

```
palb_temo_01 %>% truncate_study(overall_x_max = 30) %>% draw_study()
```



11 Fitting models with `model_study()`

The `model_study()` function takes as input a study data frame, then fits from one to four different mixed models. Which models are fit is automatically determined by the `maeve_options('metric')` values chose to summarize the groups. Supported `metric` values are

```
maeve_options('metrics_supported')
```

```
## [1] "linear"    "ITGR"      "AUC"       "ITGR_pwl"  "AUC_pwl"   "ITGR_poly" "AUC_poly"
```

For each mixed model, each subject has a random intercept and slope term, while the group-level curve is parameterized by fixed effects. The model(s) fit will be:

1. A linear mixed model (LMM) if `metric` contains `"linear"`.
2. A generalized additive mixed model (GAMM) if `metric` contains `"AUC"` and / or `"ITGR"`. With default settings, which are to to subtract out the starting value and then scale by half the squared study range (for `"AUC"`) or the study range (for `"ITGR"`), the metrics returned for `"AUC"` and `"ITGR"` are termed `eGaIT` and `eDOT`, respectively, and are described in a companion manuscript to this package.
3. A piecewise linear mixed model if `metric` contains `"AUC_pwl"` and / or `"ITGR_pwl"`. The piecewise linear fit depends on setting parameters such as `maeve_options('number_break_points')` and `maeve_options('break_points')`. This is mostly intended for cases with very few (e.g., three) time points. See examples.
4. A polynomial linear mixed model if `metric` contains `"AUC_poly"` and / or `"ITGR_poly"`. The polynomial linear fit depends on setting parameters such as `maeve_options('poly_degree')`, and makes a naive polynomial basis with the `stats::poly()`. This is mostly intended for cases with very few (e.g., three) time points. When more time points are available, a GAMM will generally outperform this approach. See examples.

By default, the LMM and GAMM are fit. The output from `model_study()` is usually sent on to another function (e.g., `predict_study()`, `compare_groups()`) for visualizing and summarizing results.

```
### model_study() examples

 vismo21 <- vismodegib %>% dplyr::filter( DAY_OF_STUDY <= 21 )

### Fit a GAMM by specifying "metric = 'AUC'" or "metric = 'ITGR'",
### the two metrics that derive from a GAMM:
 maeve_reset()

### We can return a larger list of results if needed with 'return_all = TRUE':
###
 modeling_list = maeve::model_study( vismo21, metric = 'AUC', return_all = TRUE )

### Here are the contents of the 'return_all = TRUE' option.
 modeling_list %>% names
```

```
## [1] "models" "data"   "extra"
```

```
 modeling_list %>% with( models ) %>% names # these are what you get with 'return_all = FALSE'
```

```
## [1] "md4_lmer"      "md4_gamm4"     "md4_lmer_pwl"  "md4_lmer_poly"
```

```
 modeling_list %>% with(   data ) %>% names
```

```
## [1] "clean_DF_full_xrange" "clean_DF_restricted"  "study_data_frame"
```

```
 modeling_list %>% with(  extra ) %>% names
```

```
##  [1] "metric"            "xrange"            "reference_Dunnett" "group_name"        "subject_ID"
##  [6] "x_name"            "endpoint_name"     "func_list"         "trans_func_char"   "inv_func_char"
```

```
### Usually, we just need the list of models:
 models_vismo21 = maeve::model_study( vismo21, metric = 'AUC' ) # return_all = FALSE by default.

 names( models_vismo21 )
```

```
## [1] "md4_lmer"      "md4_gamm4"     "md4_lmer_pwl"  "md4_lmer_poly"
```

```
 lapply( models_vismo21, class ) # metric = 'AUC', so 3 of the 4 models (all but 'md4_gamm4') will be NULL.
```

```
## $md4_lmer
## [1] "NULL"
##
## $md4_gamm4
## [1] "list"
##
## $md4_lmer_pwl
## [1] "NULL"
##
## $md4_lmer_poly
## [1] "NULL"
```

```
### The gamm4 results are in two distinct parts, returned in a list:
 names( models_vismo21[['md4_gamm4']] )
```

```
## [1] "mer" "gam"
```

```
### names of 'gam' object in gamm4 results:
 models_vismo21 %>% with( md4_gamm4 ) %>% with( gam ) %>%  names
```

```
##  [1] "model"             "formula"           "smooth"            "nsdf"              "family"
##  [6] "df.null"           "y"                 "terms"             "pterms"            "xlevels"
## [11] "contrasts"         "assign"            "na.action"         "cmX"               "var.summary"
```

```
## [16] "pred.formula"      "coefficients"      "scale.estimated"   "prior.weights"     "weights"
## [21] "R"                 "sp"                "edf"               "df.residual"       "sig2"
## [26] "method"            "Vp"                "Ve"                "linear.predictors" "fitted.values"
## [31] "residuals"         "gcv.ubre"
```

```
### names of attributes of 'mer' object in gamm4 results:
 models_vismo21 %>% with( md4_gamm4 ) %>% with( mer ) %>% attributes %>% names
```

```
##  [1] "resp"    "Gp"      "call"    "frame"   "flist"   "cnms"    "lower"   "theta"   "beta"    "u"
## [11] "devcomp" "pp"      "optinfo" "class"
```

```
### Fit a simple linear mixed model by specifying "metric = 'linear'":
 models_vismo21 = maeve::model_study( vismo21, metric = 'linear' )
 names( models_vismo21 )
```

```
## [1] "md4_lmer"      "md4_gamm4"     "md4_lmer_pwl"  "md4_lmer_poly"
```

```
 lapply( models_vismo21, class )
```

```
## $md4_lmer
## [1] "lmerMod"
## attr(,"package")
## [1] "lme4"
##
## $md4_gamm4
## [1] "NULL"
##
## $md4_lmer_pwl
## [1] "NULL"
##
## $md4_lmer_poly
## [1] "NULL"
```

```
 summary( models_vismo21[['md4_lmer']] )
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: y ~ group + group:x + (1 + x | ID)
##    Data: data.frame(clean_DF, weight_lmer = weight_lmer)
## Weights: weight_lmer
##
## REML criterion at convergence: 21.4
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.68265 -0.66596 -0.03618  0.67780  2.42072
##
## Random effects:
##  Groups   Name        Variance  Std.Dev. Corr
##  ID       (Intercept) 7.710e-03 0.087805
##           x           6.592e-05 0.008119 0.88
##  Residual             3.439e-02 0.185433
## Number of obs: 320, groups:  ID, 46
##
## Fixed effects:
##                  Estimate Std. Error t value
## (Intercept)      5.534860   0.062818  88.110
## groupdose_0.3    0.055251   0.093837   0.589
## groupdose_1.0   -0.081847   0.093063  -0.879
## groupdose_3.0   -0.235127   0.093636  -2.511
## groupdose_6.0   -0.072489   0.098985  -0.732
## groupdose_10    -0.093209   0.100624  -0.926
## groupdose_25    -0.401343   0.099484  -4.034
```

```
## groupdose_50    -0.344168   0.099750  -3.450
## groupdose_75    -0.280963   0.099484  -2.824
## groupdose_100   -0.466673   0.093979  -4.966
## groupdose_0.0:x  0.036672   0.005221   7.024
## groupdose_0.3:x  0.032185   0.006089   5.285
## groupdose_1.0:x  0.041710   0.005754   7.249
## groupdose_3.0:x  0.017354   0.005781   3.002
## groupdose_6.0:x -0.006018   0.006412  -0.939
## groupdose_10:x  -0.028033   0.006440  -4.353
## groupdose_25:x  -0.048045   0.006442  -7.458
## groupdose_50:x  -0.069783   0.006401 -10.902
## groupdose_75:x  -0.084307   0.006442 -13.088
## groupdose_100:x -0.085720   0.005746 -14.919

##
## Correlation matrix not shown by default, as p = 20 > 12.
## Use print(x, correlation=TRUE)  or
##     vcov(x)        if you need it
```

```
### Fit a piecewise linear model by specifying "metric = 'AUC_pwl'"
### (or 'ITGR_pwl') and also the break_points spanning the time range:
 maeve_reset()
 maeve_options( break_points = c( 0, 7, 14, 21 ) )
 models_vismo21 = maeve::model_study( vismo21, metric = 'AUC_pwl' )

### Fit a simple polynomial model by specifying "metric = 'AUC_poly'"
### (or 'ITGR_poly') and also the degree.
 maeve_reset()
 maeve_options( poly_degree = 3 )
 models_vismo21 = maeve::model_study( vismo21, metric = 'AUC_poly' )

### Fit all the endpoints at once:
 maeve_reset()
 maeve_options(
                metric = maeve_options('metrics_supported'), # all available summary metrics
                break_points = c(0, 7, 14, 21),
                poly_degree = 3
               )
 models_vismo21 = maeve::model_study( vismo21 )

 lapply( models_vismo21, class ) # see that all four models are now non-NULL
```

```
## $md4_lmer
## [1] "lmerMod"
## attr(,"package")
## [1] "lme4"
##
## $md4_gamm4
## [1] "list"
##
## $md4_lmer_pwl
## [1] "lmerMod"
## attr(,"package")
## [1] "lme4"
##
## $md4_lmer_poly
## [1] "lmerMod"
## attr(,"package")
## [1] "lme4"
```

```
### Save a couple of current options settings to make it easy to revisit
### this example after some other studies:
###
### On to-do list is add an "except = c(...)" argument to "maeve_reset()".
        poly_object_vismo21 <- maeve_options('poly_object')
 modeling_data_frame_vismo21 <- maeve_options('modeling_data_frame')
```

12 Predicting from models with `predict_study()`

The `predict_study()` function can be used to predict conditional and unconditional values (i.e., with & without random effects, respectively) for up to four models fit in `model_study()`. By default, this is at observed x-values, but use various options (e.g., `maeve_options('x_pred_type')`, etc.) for predicting at non-observed times. See examples.

```
### predict_study()

### Get a data.frame with predicted responses at a desired set of x-values.
###
### NB: Predicted values are retained in the data.frame even if you do NOT
### want them.  Get rid of them if desired by filtering on 'in_pred_x_set'

### Predictions over an evenly-spaced grid spanning observed x-values, with grid points 0.5 apart:
 pred_data_frame <-
   predict_study( models_vismo21, x_pred_type = 'grid', x_pred_spacing = 0.5 )

### Provide a user-specified x-grid over which to make predictions.
 pred_data_frame <-
   predict_study( models_vismo21, x_pred_type = 'custom', x_pred_vec = seq( 0, 20, by = 2) )

### By default, "predict_study()" makes predictions only at observed x-values:
 pred_data_frame <- predict_study( models_vismo21 )
 head( pred_data_frame, 4 )
```

```
##   group_name animalID DAY_OF_STUDY TUMOR_VOLUME    group      ID  x   y_orig        y model_x_value
## 1   dose_0.0  3810580            0     212.7522 dose_0.0 3810580  0 212.7522 5.364818          TRUE
## 2   dose_0.0  3810580            3     283.0926 dose_0.0 3810580  3 283.0926 5.649300          TRUE
## 3   dose_0.0  3810580            6     337.0389 dose_0.0 3810580  6 337.0389 5.823161          TRUE
## 4   dose_0.0  3810580           10     479.5100 dose_0.0 3810580 10 479.5100 6.174848          TRUE
##   in_pred_x_set in_model_x_range intercept_re_lmer slope_re_lmer intercept_re_gamm4 slope_re_gamm4
## 1          TRUE             TRUE        0.02831629   0.003034573        0.008323168    0.004904741
## 2          TRUE             TRUE        0.02831629   0.003034573        0.008323168    0.004904741
## 3          TRUE             TRUE        0.02831629   0.003034573        0.008323168    0.004904741
## 4          TRUE             TRUE        0.02831629   0.003034573        0.008323168    0.004904741
##   intercept_re_lmer_pwl slope_re_lmer_pwl intercept_re_lmer_poly slope_re_lmer_poly pred_lin pred_lin_ID
## 1            0.01034241       0.004785803            0.008717379        0.004802489 5.534860    5.563176
## 2            0.01034241       0.004785803            0.008717379        0.004802489 5.644875    5.682295
## 3            0.01034241       0.004785803            0.008717379        0.004802489 5.754891    5.801414
## 4            0.01034241       0.004785803            0.008717379        0.004802489 5.901578    5.960240
##   pred_gam pred_gam_ID pred_pwl pred_pwl_ID pred_poly pred_poly_ID    y_norm pred_lin_norm pred_gam_norm
## 1 5.485855    5.494178 5.458929    5.469271  5.450738     5.459455  0.000000      0.000000      0.000000
## 2 5.639951    5.662988 5.639877    5.664577  5.655722     5.678847  5.302743      1.987683      2.808977
## 3 5.783585    5.821337 5.820824    5.859882  5.811013     5.848545  8.543504      3.975367      5.427244
## 4 5.947002    6.004372 5.956143    6.014344  5.961288     6.018030 15.098936      6.625612      8.406109
##   pred_pwl_norm pred_poly_norm pred_gam_deriv1 base_pred_lin base_pred_gam base_pred_gam_deriv1
## 1      0.000000       0.000000      0.05197849      5.534860      5.485855           0.05197849
## 2      3.314711       3.760674      0.05013869      5.644875      5.639951           0.05013869
```

```
## 3      6.629422       6.609651      0.04520969      5.754891      5.783585           0.04520969
## 4      9.108274       9.366626      0.03598394      5.901578      5.947002           0.03598394
```

```
 pred_data_frame %>%
     dplyr::select( group_name, animalID, DAY_OF_STUDY, in_pred_x_set, TUMOR_VOLUME, pred_gam ) %>% head( 15 )
```

```
##    group_name animalID DAY_OF_STUDY in_pred_x_set TUMOR_VOLUME pred_gam
## 1    dose_0.0  3810580            0          TRUE     212.7522 5.485855
## 2    dose_0.0  3810580            3          TRUE     283.0926 5.639951
## 3    dose_0.0  3810580            6          TRUE     337.0389 5.783585
## 4    dose_0.0  3810580           10          TRUE     479.5100 5.947002
## 5    dose_0.0  3810580           14          TRUE     429.2777 6.072502
## 6    dose_0.0  3810580           17          TRUE     592.9786 6.157073
## 7    dose_0.0  3810580           21          TRUE     537.5505 6.264261
## 8    dose_0.0  3810587            0          TRUE     223.8383 5.485855
## 9    dose_0.0  3810587            3          TRUE     231.8135 5.639951
## 10   dose_0.0  3810587            7          TRUE     340.4171 5.827764
## 11   dose_0.0  3810587           11          TRUE     401.2144 5.981587
## 12   dose_0.0  3810587           14          TRUE     389.1488 6.072502
## 13   dose_0.0  3810587           18          TRUE     390.5655 6.184431
## 14   dose_0.0  3810587           21          TRUE     483.5132 6.264261
## 15   dose_0.0  3810601            0          TRUE     213.5640 5.485855
```

```
### More examples:
 pred_data_frame <- predict_study( models_vismo21, x_pred_type = 'union_observed_and_grid', x_pred_spacing = 2 )

 pred_data_frame %>%
     dplyr::select( group_name, animalID, DAY_OF_STUDY, in_pred_x_set, TUMOR_VOLUME, pred_gam ) %>% head( 20 )
```

```
##    group_name animalID DAY_OF_STUDY in_pred_x_set TUMOR_VOLUME pred_gam
## 1    dose_0.0  3810580            0          TRUE     212.7522 5.485855
## 2    dose_0.0  3810580            2          TRUE           NA 5.589267
## 3    dose_0.0  3810580            3          TRUE     283.0926 5.639951
## 4    dose_0.0  3810580            4          TRUE           NA 5.689428
## 5    dose_0.0  3810580            6          TRUE     337.0389 5.783585
## 6    dose_0.0  3810580            7          TRUE           NA 5.827764
## 7    dose_0.0  3810580            8          TRUE           NA 5.869828
## 8    dose_0.0  3810580           10          TRUE     479.5100 5.947002
## 9    dose_0.0  3810580           11          TRUE           NA 5.981587
## 10   dose_0.0  3810580           12          TRUE           NA 6.013561
## 11   dose_0.0  3810580           14          TRUE     429.2777 6.072502
## 12   dose_0.0  3810580           15          TRUE           NA 6.100959
## 13   dose_0.0  3810580           16          TRUE           NA 6.129196
## 14   dose_0.0  3810580           17          TRUE     592.9786 6.157073
## 15   dose_0.0  3810580           18          TRUE           NA 6.184431
## 16   dose_0.0  3810580           20          TRUE           NA 6.237869
## 17   dose_0.0  3810580           21          TRUE     537.5505 6.264261
## 18   dose_0.0  3810587            0          TRUE     223.8383 5.485855
## 19   dose_0.0  3810587            2          TRUE           NA 5.589267
## 20   dose_0.0  3810587            3          TRUE     231.8135 5.639951
```

```
 pred_data_frame %>%
  dplyr::select( group_name, animalID, DAY_OF_STUDY, TUMOR_VOLUME,
                 y, pred_lin, pred_gam, pred_pwl, pred_poly
               ) %>%
  head( 10 )
```

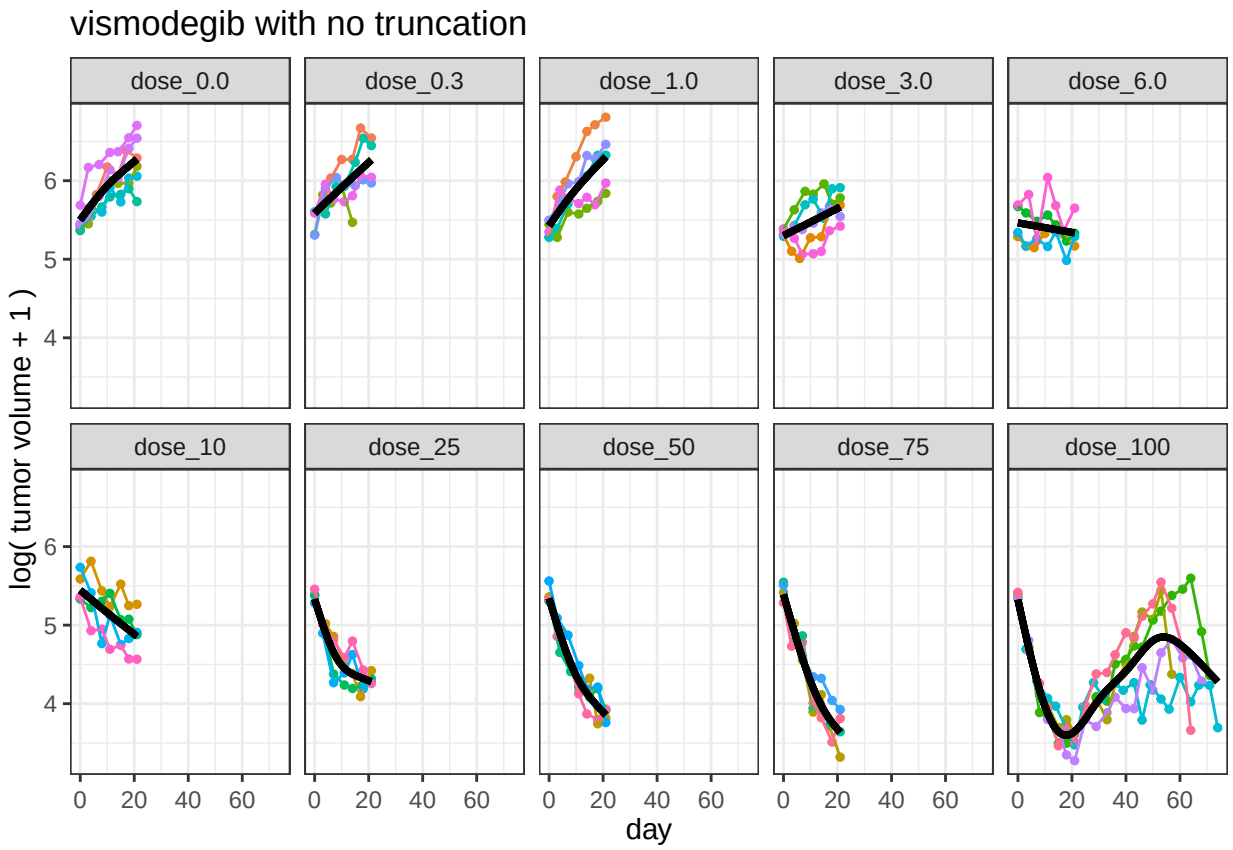
```
##    group_name animalID DAY_OF_STUDY TUMOR_VOLUME        y pred_lin pred_gam pred_pwl pred_poly
## 1    dose_0.0  3810580            0     212.7522 5.364818 5.534860 5.485855 5.458929  5.450738
## 2    dose_0.0  3810580            2           NA       NA 5.608203 5.589267 5.579561  5.593502
## 3    dose_0.0  3810580            3     283.0926 5.649300 5.644875 5.639951 5.639877  5.655722
```

```
## 4    dose_0.0  3810580           4          NA       NA 5.681547 5.689428 5.700193  5.712421
## 5    dose_0.0  3810580           6    337.0389 5.823161 5.754891 5.783585 5.820824  5.811013
## 6    dose_0.0  3810580           7          NA       NA 5.791562 5.827764 5.881140  5.853785
## 7    dose_0.0  3810580           8          NA       NA 5.828234 5.869828 5.906141  5.892796
## 8    dose_0.0  3810580          10    479.5100 6.174848 5.901578 5.947002 5.956143  5.961288
## 9    dose_0.0  3810580          11          NA       NA 5.938250 5.981587 5.981144  5.991650
## 10   dose_0.0  3810580          12          NA       NA 5.974921 6.013561 6.006145  6.020009
```

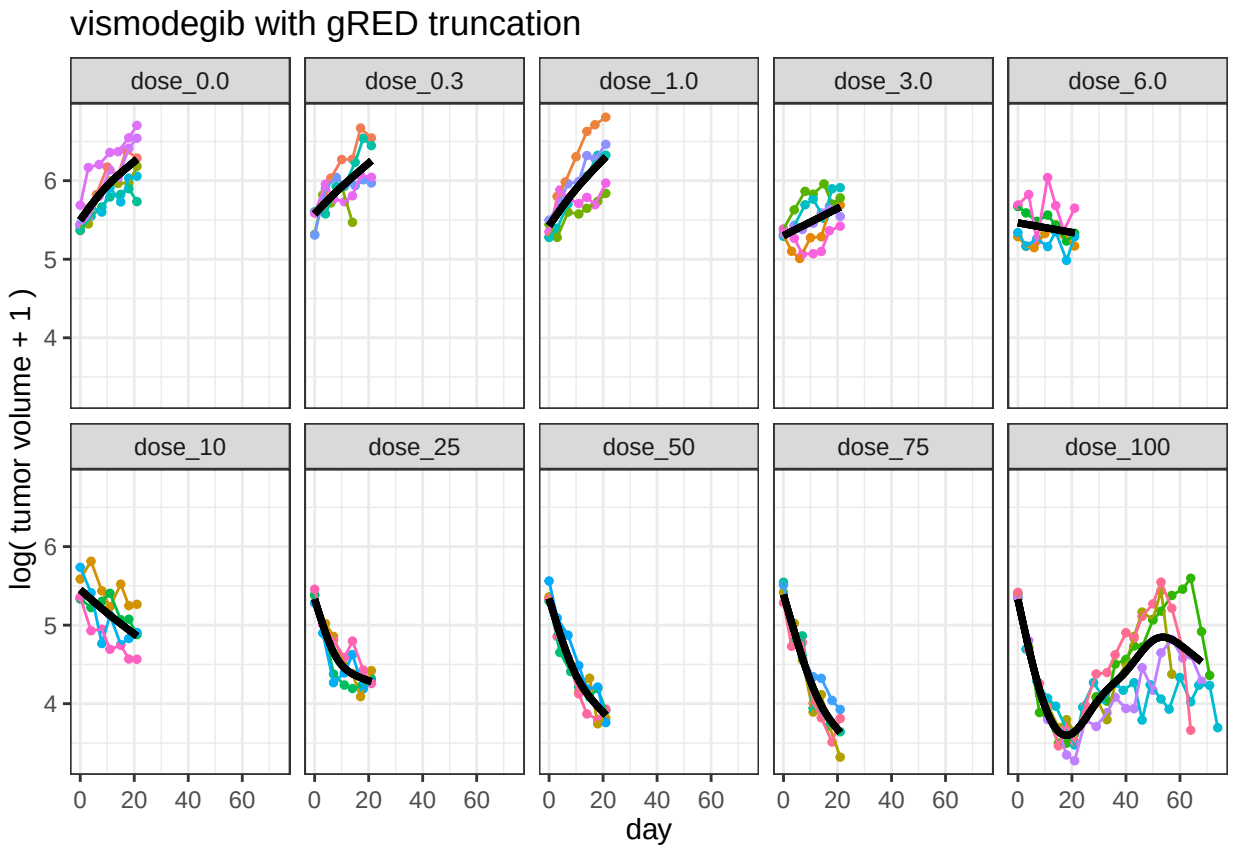
13 Truncation with `model_study()` and `predict_study()`

While study truncation *can* be called explicitly via `maeve::truncate_study()`, a better approach is simply to set the truncation parameters in `maeve_options()`, then run `model_study()`. This will yield models that are fit on only the non-truncated data while (by default) keeping track of all the data (both truncated and untruncated) in `maeve_options("full_study_data_frame")`. The results of `model_study()` can be passed directly to `predict_study()` in order to return a data frame that tracks both the full original data and the predicted values, at whatever grid of predicted `x`-locations was requested.

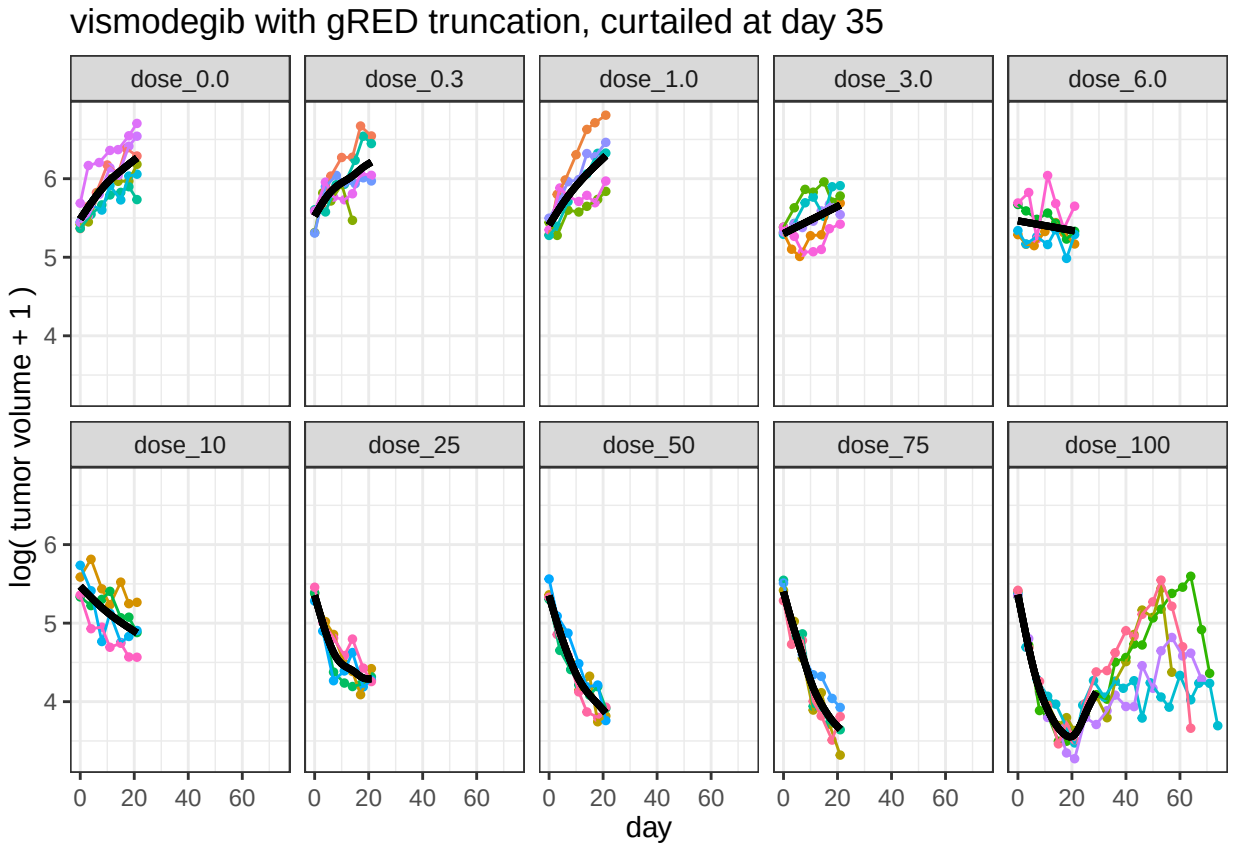
```
maeve_reset()
maeve_options( "autoset_full_study_data" = TRUE,  "autoset_modeling_data" = TRUE, truncate_fit = TRUE )
maeve_options( x_pred_type = 'grid', x_pred_spacing = 1, metric = 'AUC' )

## In general, truncate_study() will get called *within* model_study().

data( vismodegib, package = 'maeve' )

### Set the truncation parameters at their package defaults, just to explicitly show them:
maeve_options( min_n_in_group = 0, min_frac_in_group = 0.00, min_frac_in_study = 0.00 ) # no truncation
maeve_options( overall_x_min = -Inf, overall_x_max = Inf ) # no truncation from here, either
vismodegib %>%
  model_study %>%
  predict_study %>%
  draw_study( endpoint_name = 'y', fit = 'spline', ncol_value = 5, title_label = 'vismodegib with no truncation' )
```



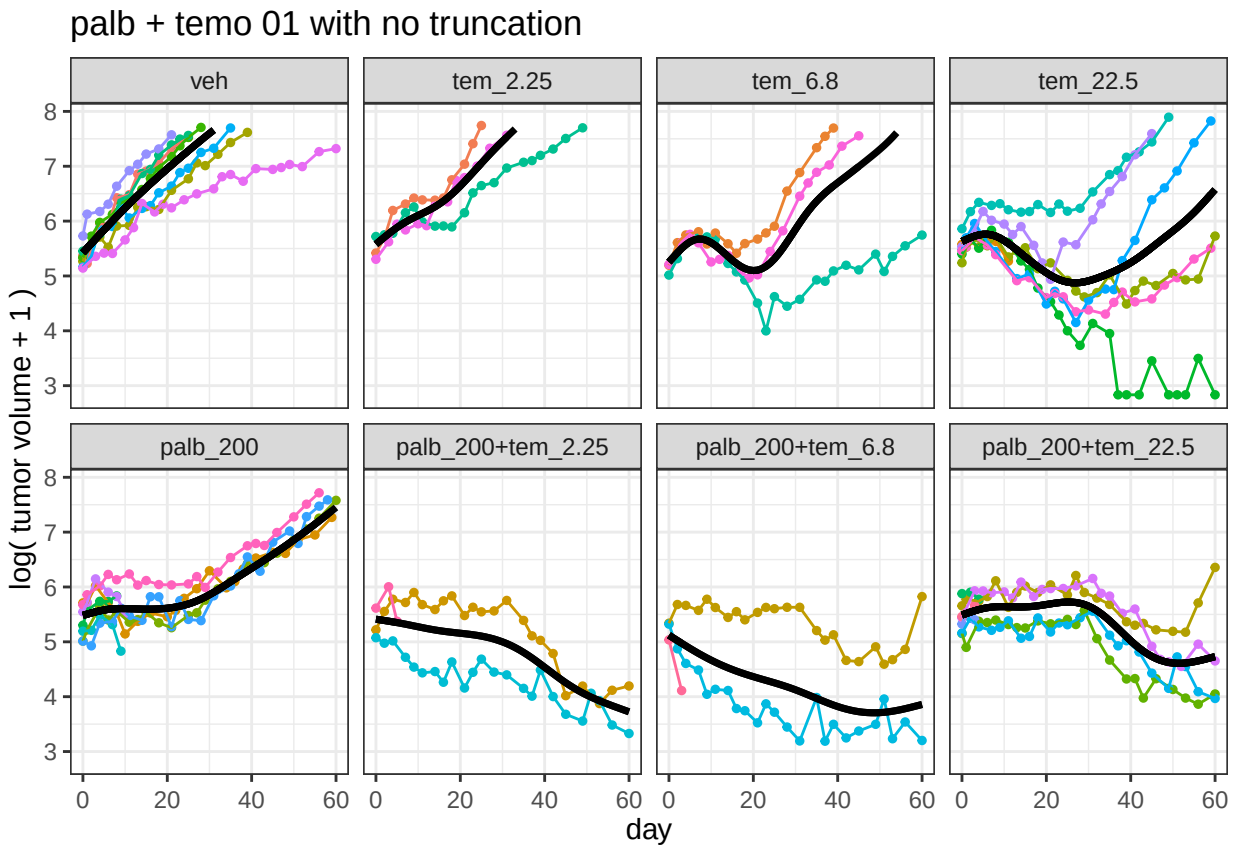

```
### Here is the the vismodegib data with gRED default truncation
maeve_options( min_n_in_group = 3, min_frac_in_group = 0.50, min_frac_in_study = 0.30 ) # gRED truncation
vismodegib %>%
  model_study %>%
  predict_study %>%
  draw_study( endpoint_name = 'y', fit = 'spline', ncol_value = 5,
              title_label = 'vismodegib with gRED truncation'
            )
```



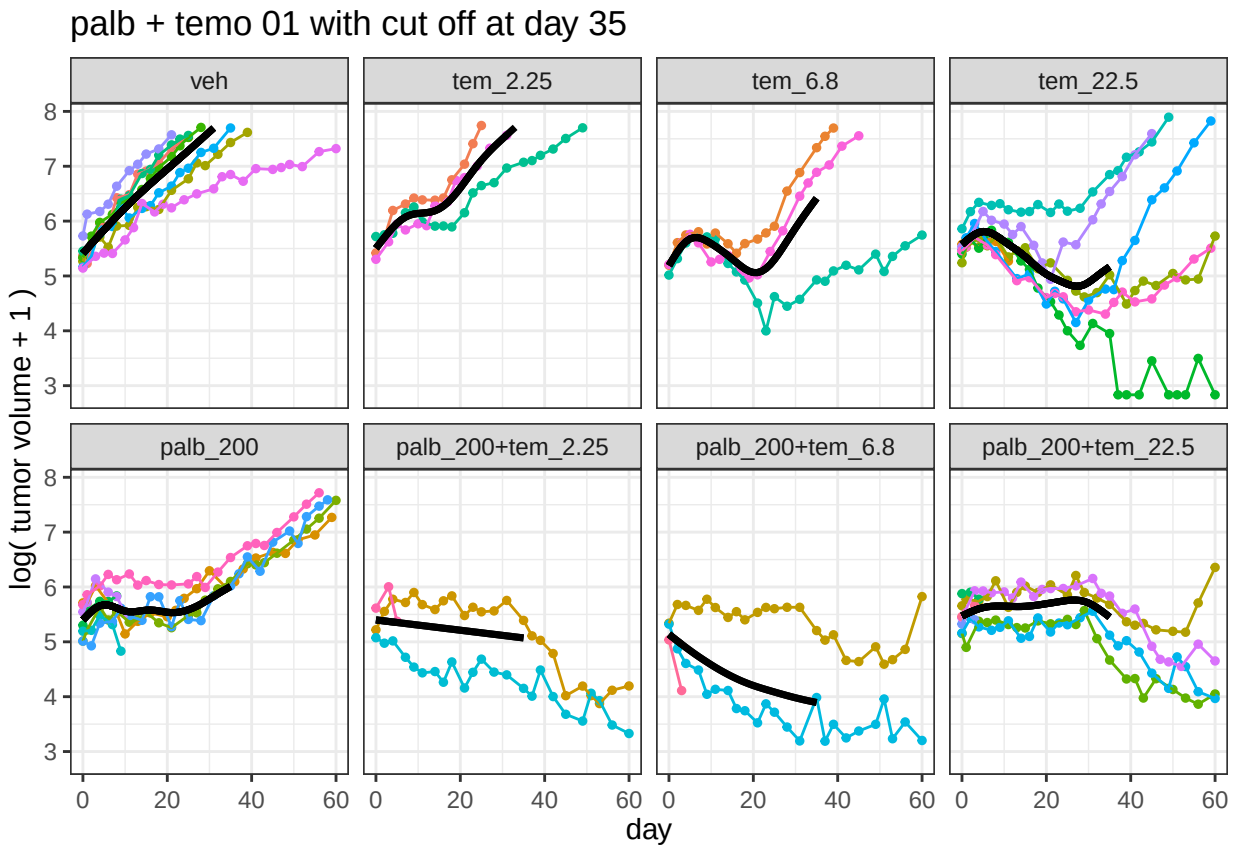

```
### Here is the the vismodegib data with gRED default truncation, but a hard truncation at day 30:
maeve_options( min_n_in_group = 3, min_frac_in_group = 0.50, min_frac_in_study = 0.30, overall_x_max = 30 )
vismodegib %>%
  model_study %>%
  predict_study %>%
  draw_study( endpoint_name = 'y', fit = 'spline', ncol_value = 5,
              title_label = 'vismodegib with gRED truncation, curtailed at day 35'
            )
```



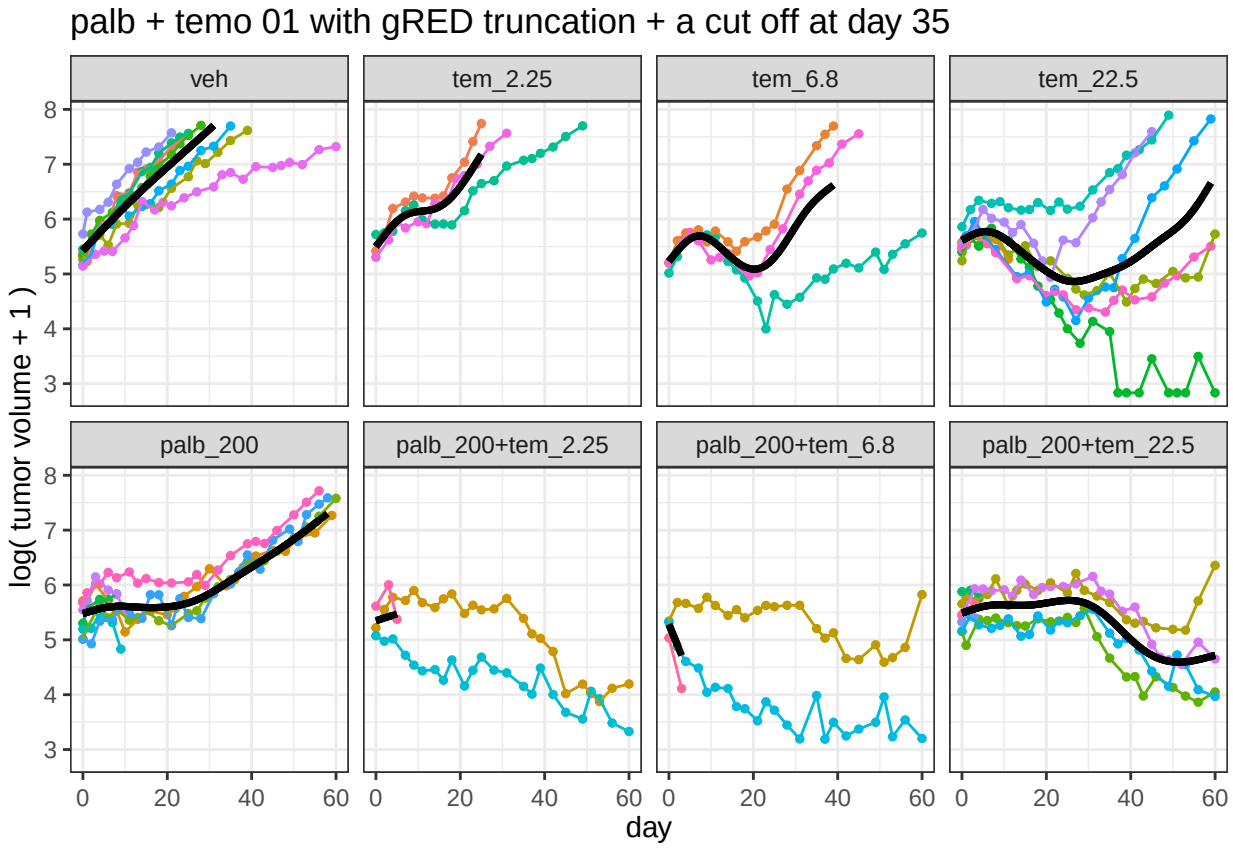

```
### Here is the the first palbociclib + temozolomide data set with no truncation:
data( palb_temo_01 )
maeve_options( min_n_in_group = 0, min_frac_in_group = 0.00, min_frac_in_study = 0.00, overall_x_max = Inf )
palb_temo_01 %>%
  model_study %>%
  predict_study %>%
  draw_study( endpoint_name = 'y', fit = 'spline', ncol_value = 4,
              title_label = 'palb + temo 01 with no truncation'
            )
```



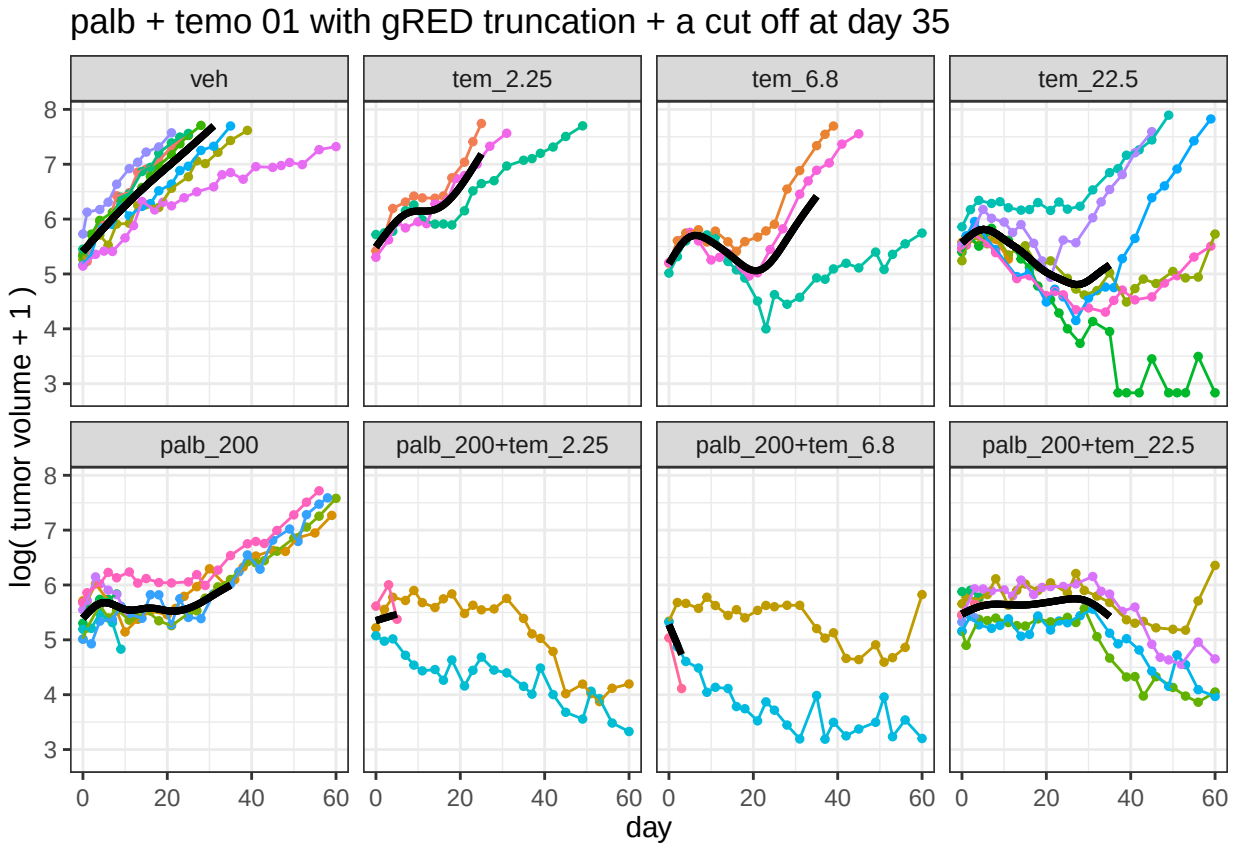

```
### Here is the the first palbociclib + temozolomide data set with no truncation but a hard cutoff at day 35
maeve_options( min_n_in_group = 0, min_frac_in_group = 0, min_frac_in_study = 0, overall_x_max = 35 )
palb_temo_01 %>%
  model_study %>%
  predict_study %>%
  draw_study( endpoint_name = 'y', fit = 'spline', ncol_value = 4,
              title_label = 'palb + temo 01 with cut off at day 35'
            )
```




```
### Here is the the first palbociclib + temozolomide data set with gRED truncation
maeve_options( min_n_in_group = 3, min_frac_in_group = 0.50, min_frac_in_study = 0.30, overall_x_max = Inf )
palb_temo_01 %>%
  model_study %>%
  predict_study %>%
  draw_study( endpoint_name = 'y', fit = 'spline', ncol_value = 4,
              title_label = 'palb + temo 01 with gRED truncation + a cut off at day 35'
            )
```



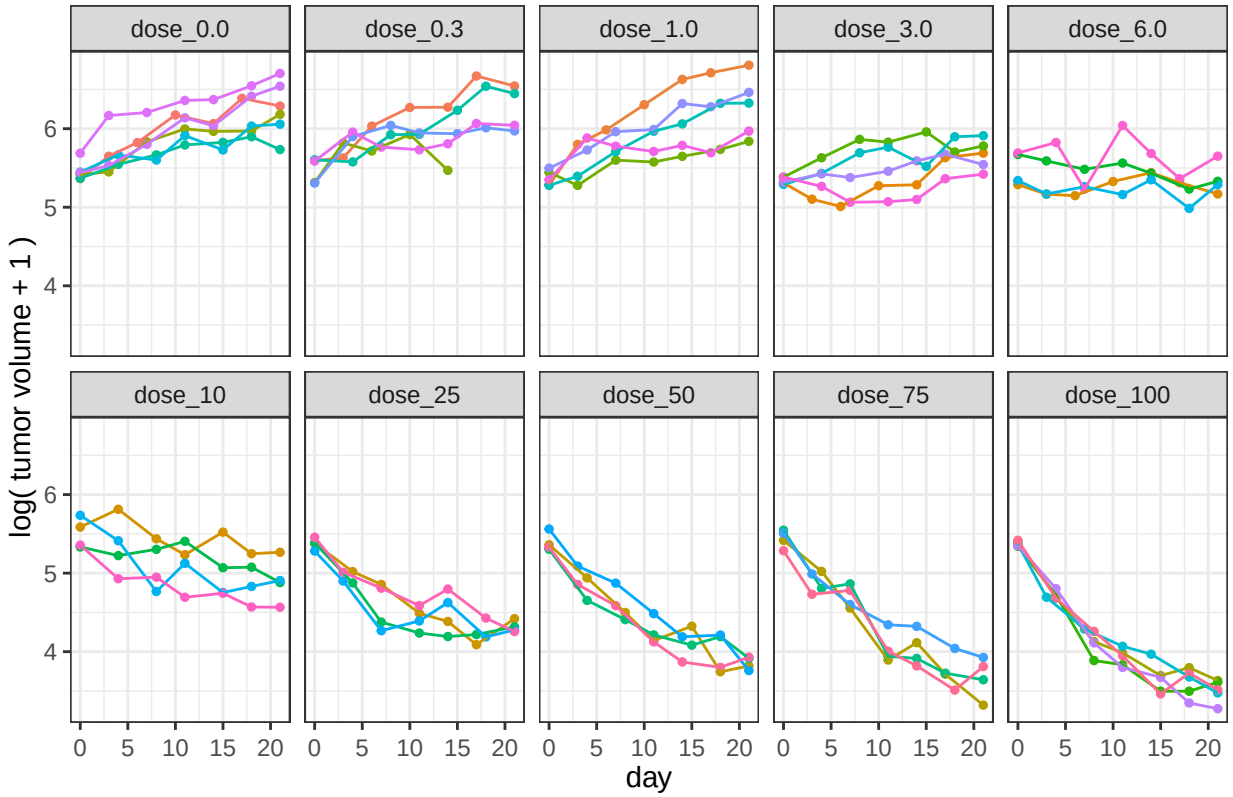

```
### Here is the the first palbociclib + temozolomide data set with gRED truncation and a hard cutoff at day 35
maeve_options( min_n_in_group = 3, min_frac_in_group = 0.50, min_frac_in_study = 0.30, overall_x_max = 35 )
palb_temo_01 %>%
  model_study %>%
  predict_study %>%
  draw_study( endpoint_name = 'y', fit = 'spline', ncol_value = 4,
              title_label = 'palb + temo 01 with gRED truncation + a cut off at day 35'
            )
```



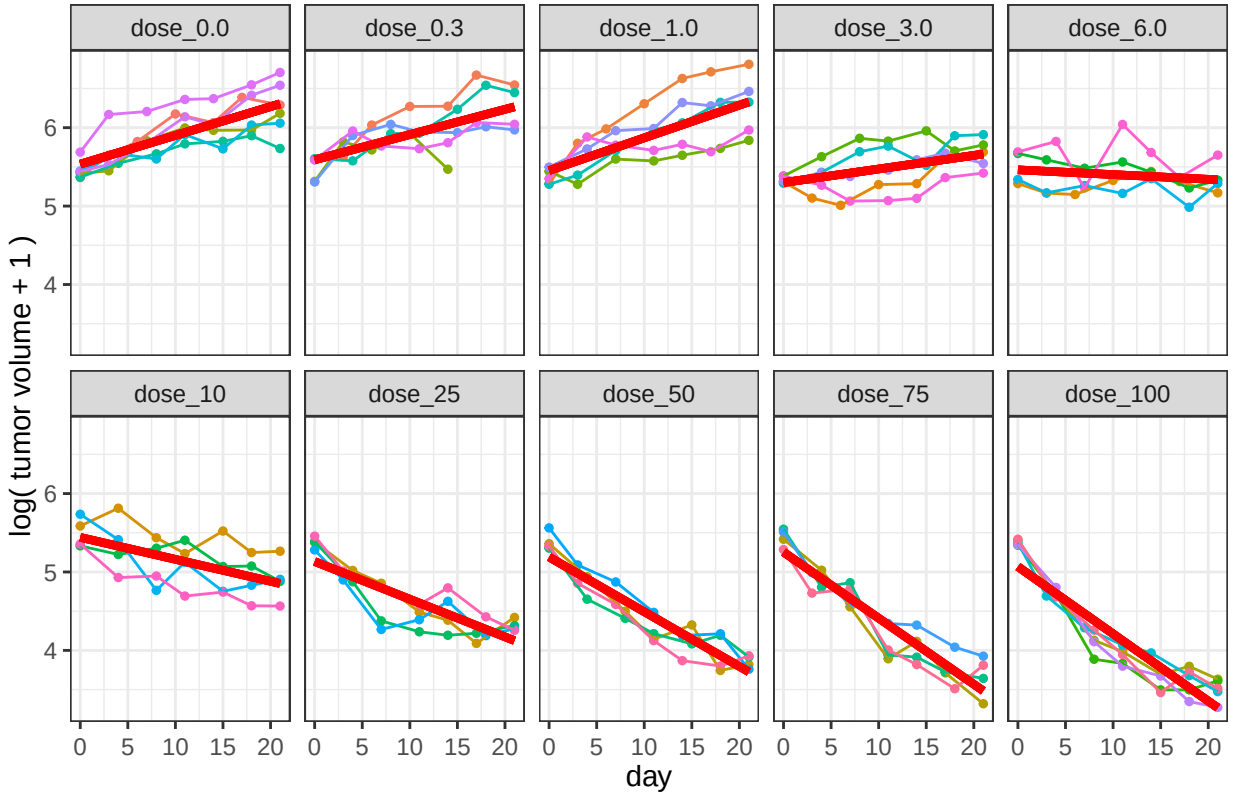
14 Using the predicted data frame from `predict_study()` in figures: `draw_study()`, `draw_waterfall()`, `draw_overlay()`, `draw_noise()`.

The functions `draw_study()`, `draw_waterfall()`, `draw_overlay()`, and `draw_noise()` allow for simple visualizations of commonly queried features in a study. Examples below show how to use them to see results captured in the output from the `predict_study()` function.

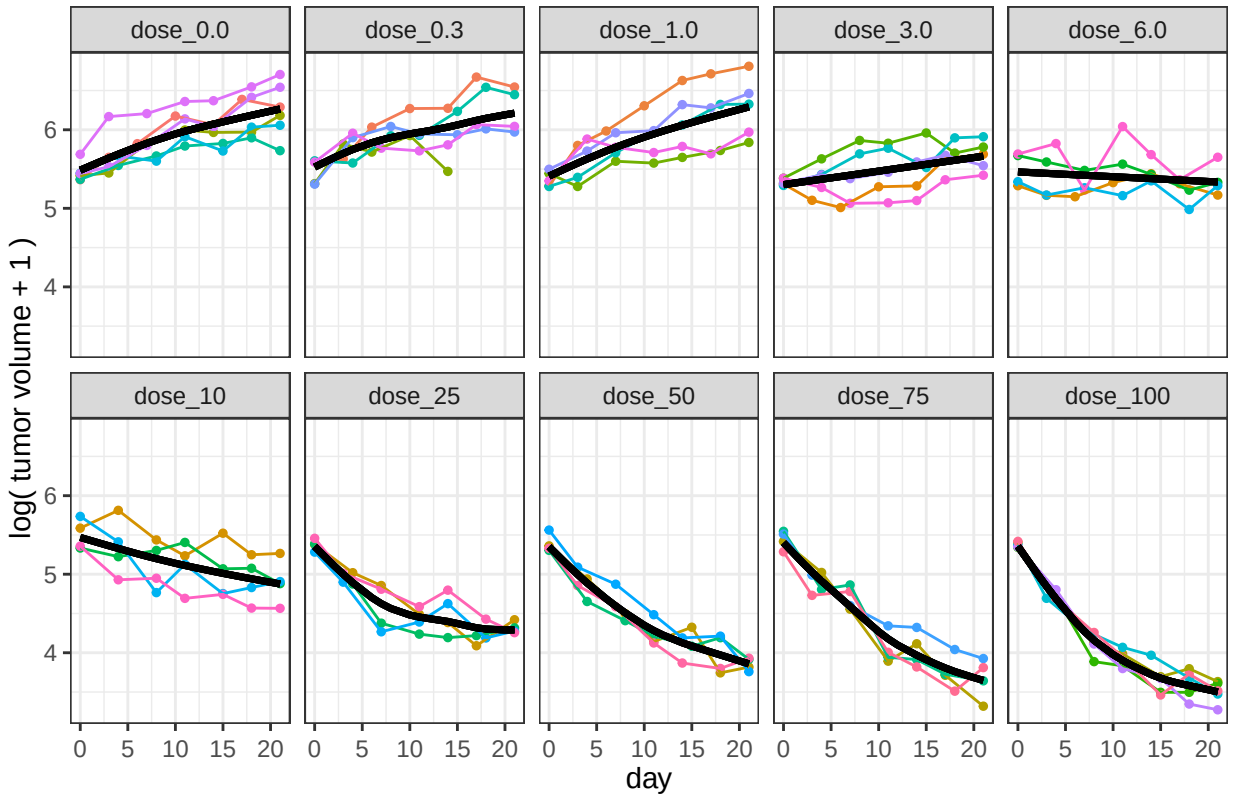
```
### 6. draw_study(), draw_overlay(), draw_noise(), draw_waterfa() using results of predict_study().

### A bunch of plots with fixed effects curves from different models.
 maeve_reset()
 maeve_options( endpoint_name = 'y', ncol_value = 5 )
 pred_data_frame %>% draw_study( )
```



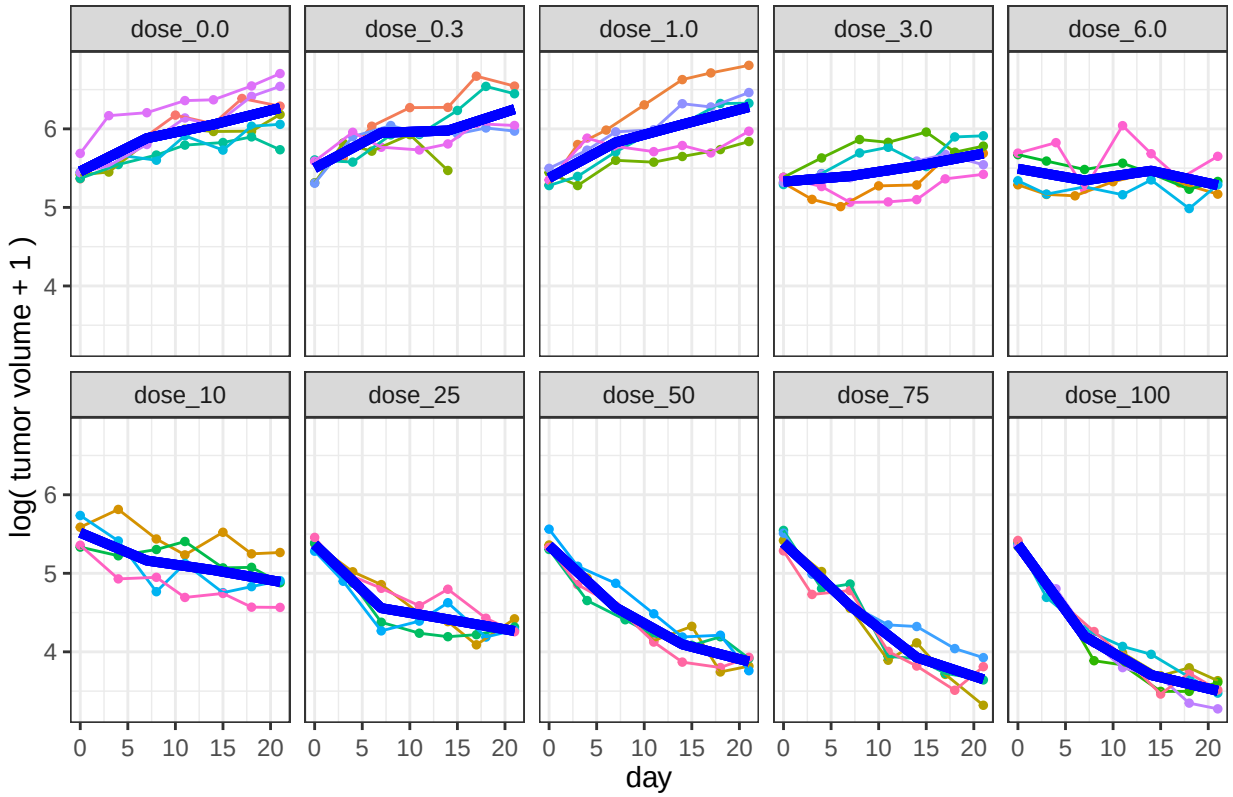

```
pred_data_frame %>% draw_study( fit = 'linear' )
```



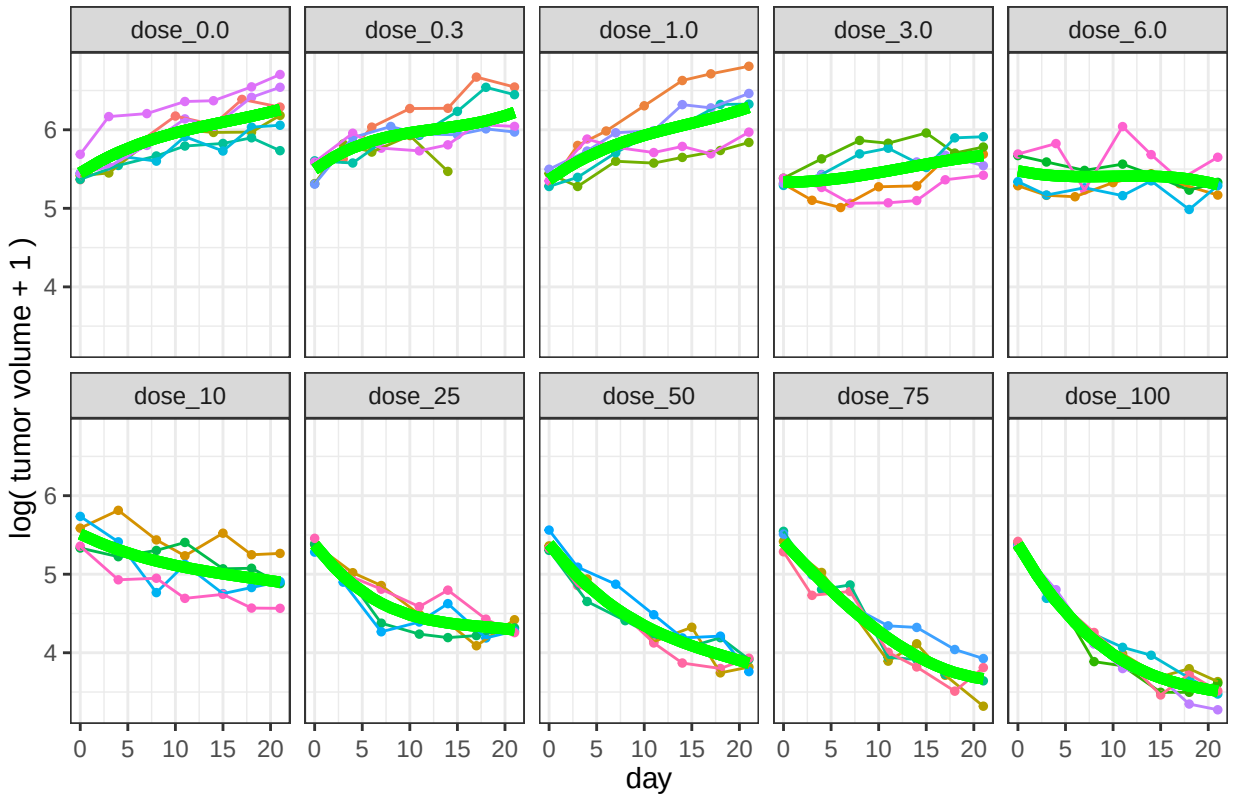
```
pred_data_frame %>% draw_study( fit = 'spline' )
```



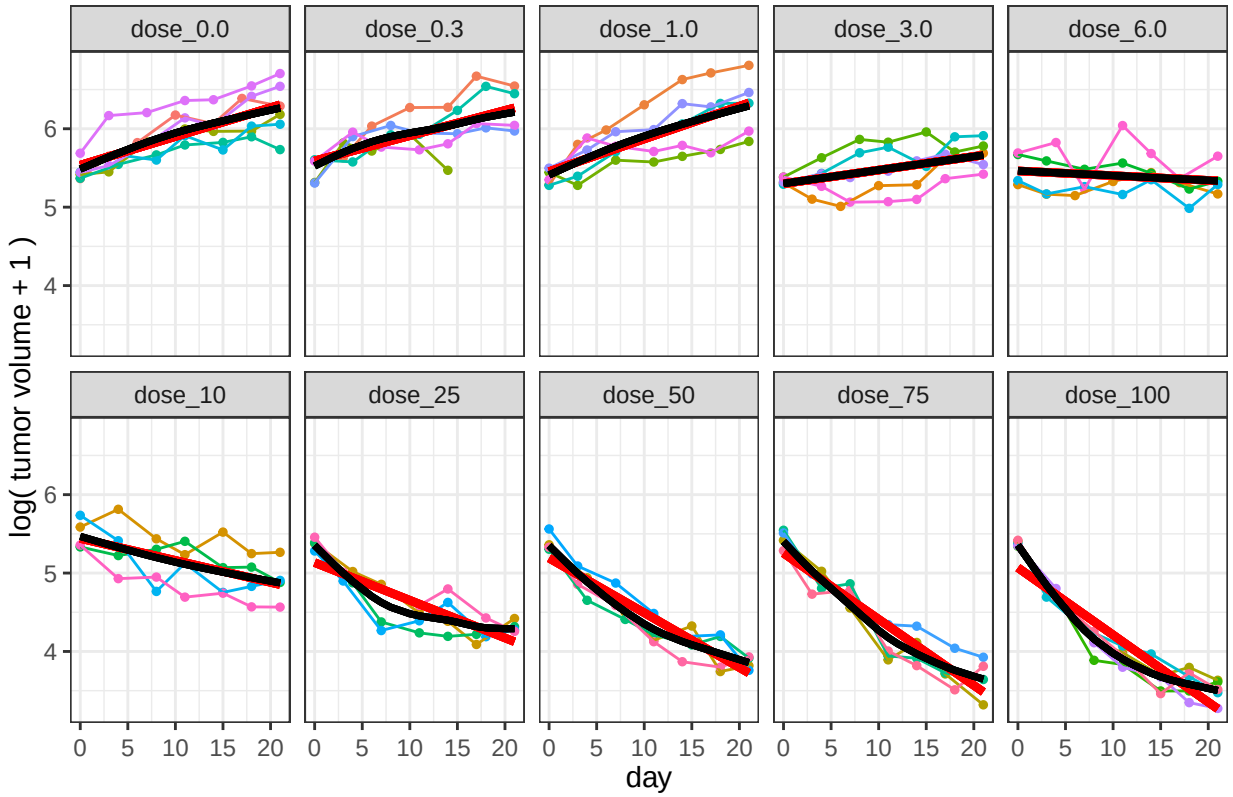

```
pred_data_frame %>% draw_study( fit = 'piecewise' )
```



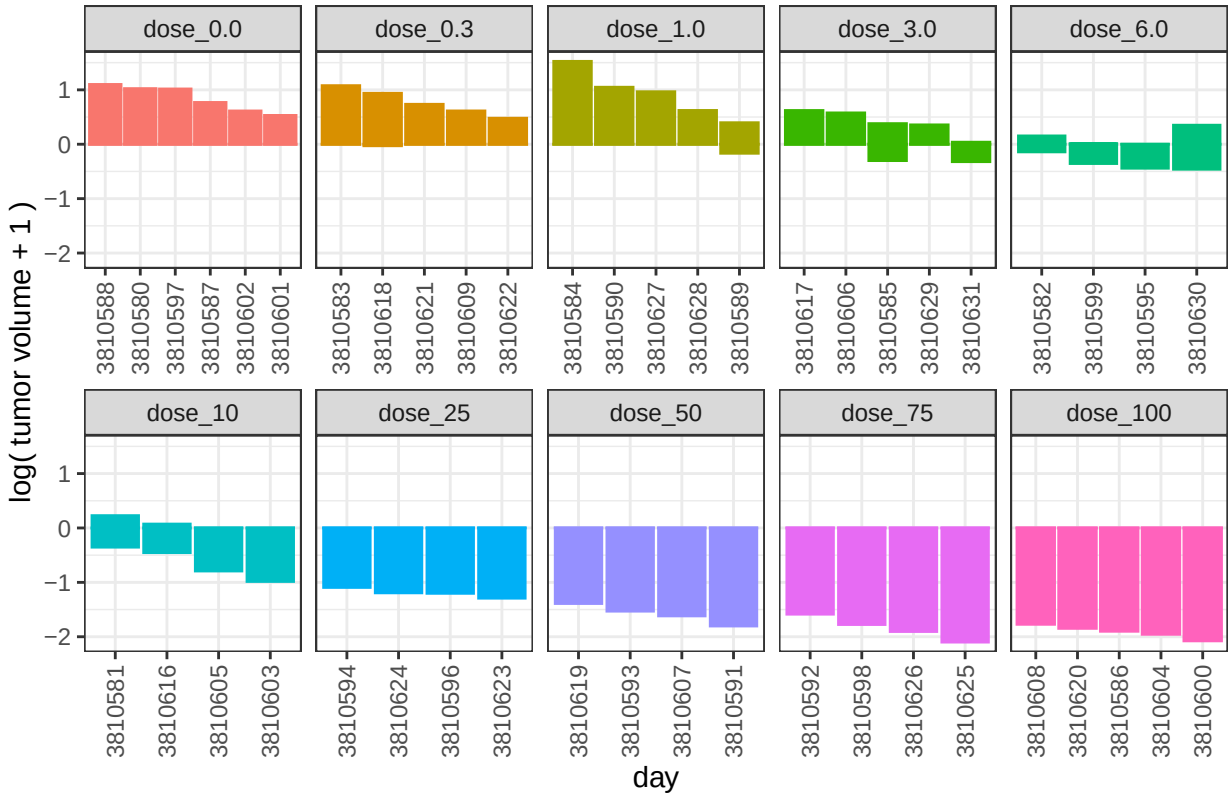
```
pred_data_frame %>% draw_study( fit = 'poly' )
```



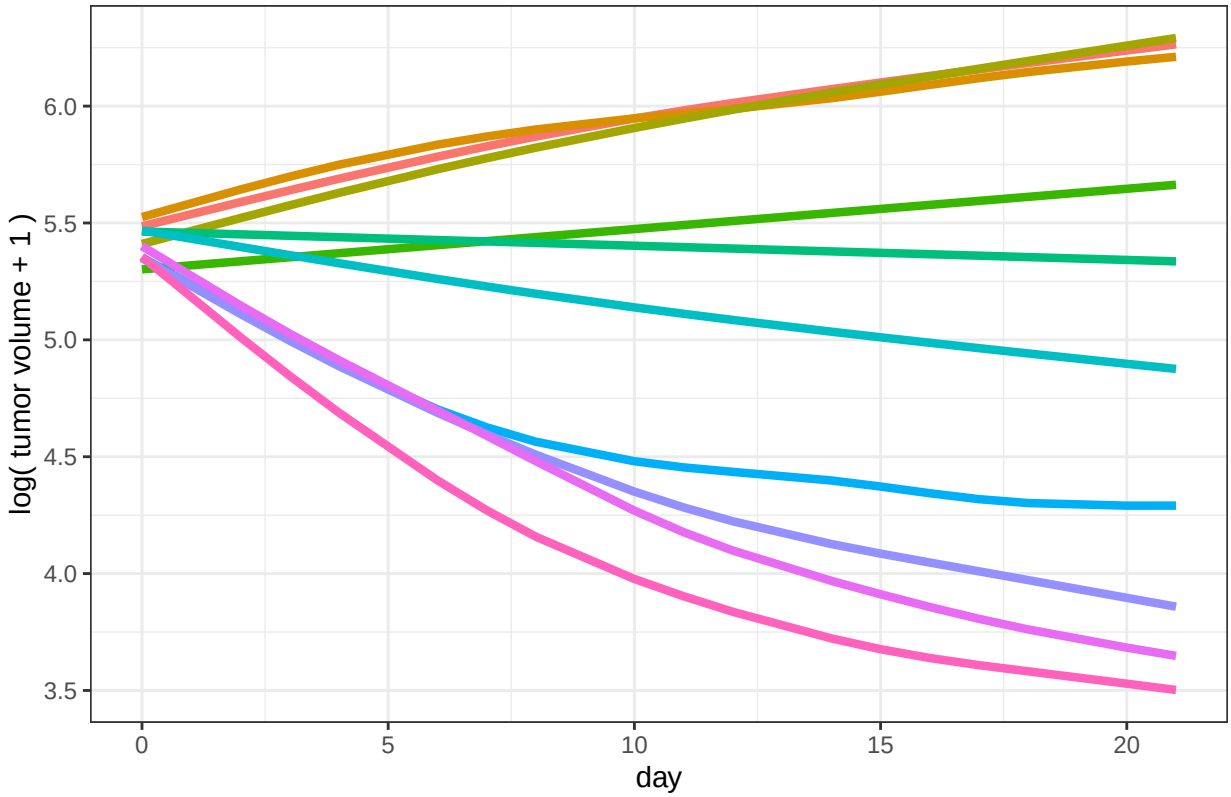

```
pred_data_frame %>% draw_study( fit = c('linear', 'spline') )
```



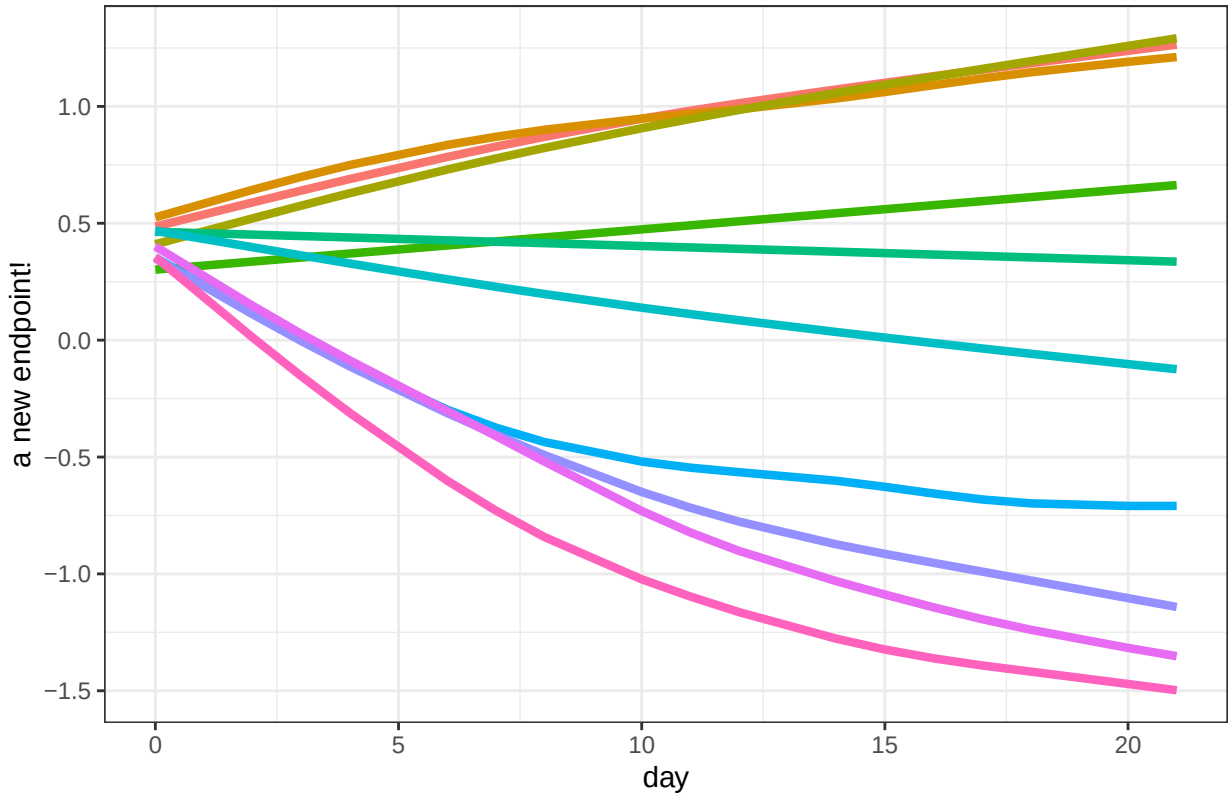
```
###
 pred_data_frame %>%
  draw_waterfall( endpoint_name = 'y', facet_char = 'group_name', ncol_value = 5 )
```



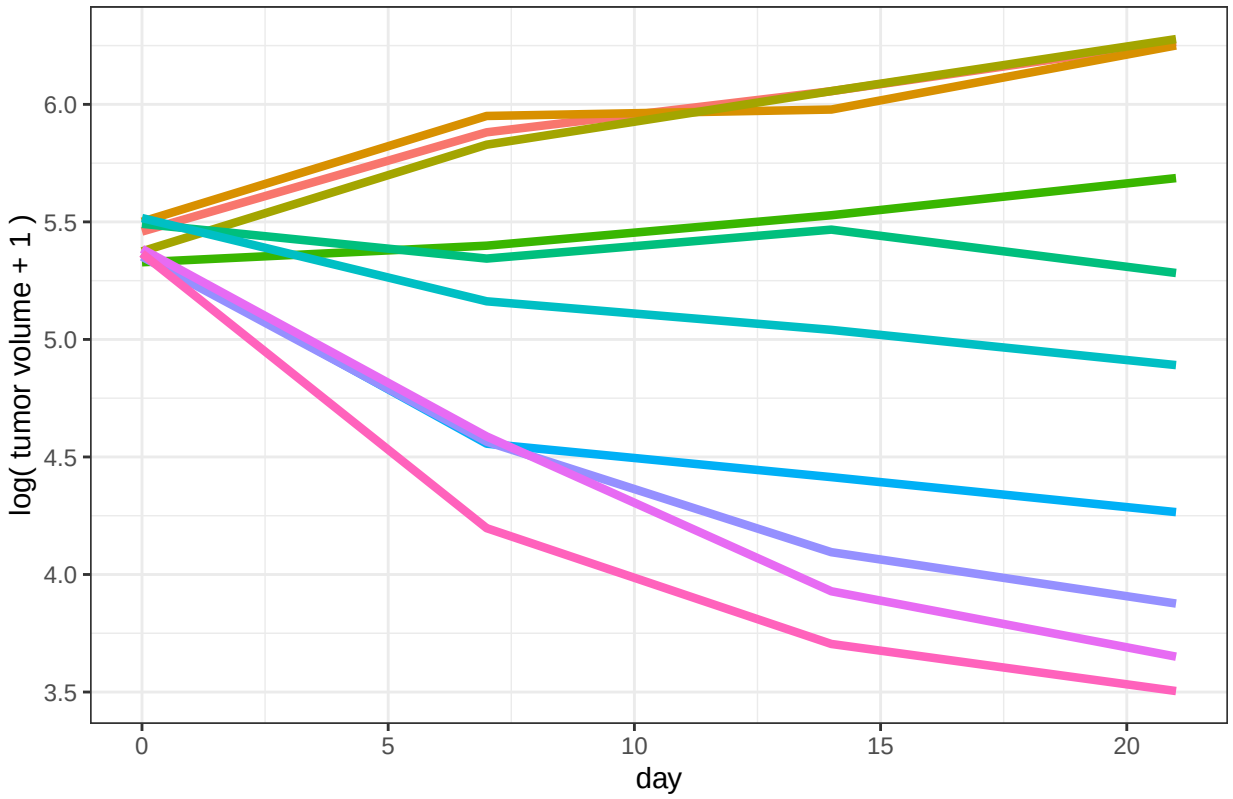

```
### draw_overlay()

 pred_data_frame %>% draw_overlay( fit = 'spline' )
```




```
pred_data_frame %>% # make up another predictor from an exisiting predictor as we go:
   dplyr::mutate( pred_gam_minus_5 = pred_gam - 5 ) %>%
   draw_overlay( fit = 'spline', spline_predictor = 'pred_gam_minus_5', y_label = 'a new endpoint!')
```



```
pred_data_frame %>% draw_overlay( fit = 'piecewise' )
```



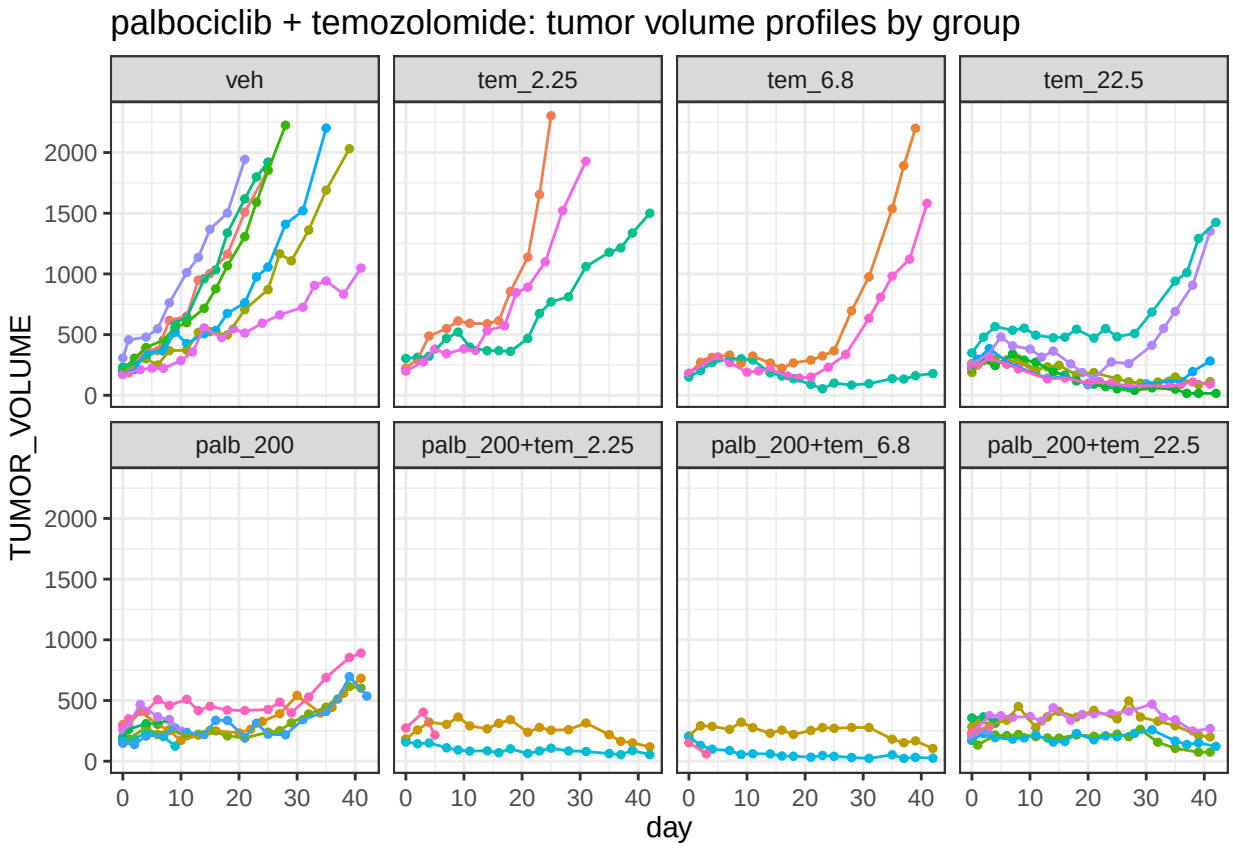

```
### Look at a second data set:

data( palb_temo_01, package = 'maeve' )
maeve_reset()
maeve_options( break_points = c( 0, 7, 21, 42 ),
               metric = c( 'linear', 'ITGR', 'AUC', 'ITGR_pwl', 'AUC_pwl', 'ITGR_poly', 'AUC_poly' ),
               x_pred_type = 'grid', x_pred_spacing = 1,
               ncol_value = 4 # 8 groups total
              )

pred_palb_temo_list <-
    palb_temo_01 %>%
    dplyr::filter( DAY_OF_STUDY <= 42 ) %>%
    model_study %>%
    predict_study( return_list = TRUE ) # return extra components (used below) in a list

pred_palb_temo <- pred_palb_temo_list[["clean_DF_pred"]]


palb_temo_01 %>%
 dplyr::group_by( group_name ) %>%
 dplyr::summarise( first_day = min(DAY_OF_STUDY), last_day = max(DAY_OF_STUDY) )
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

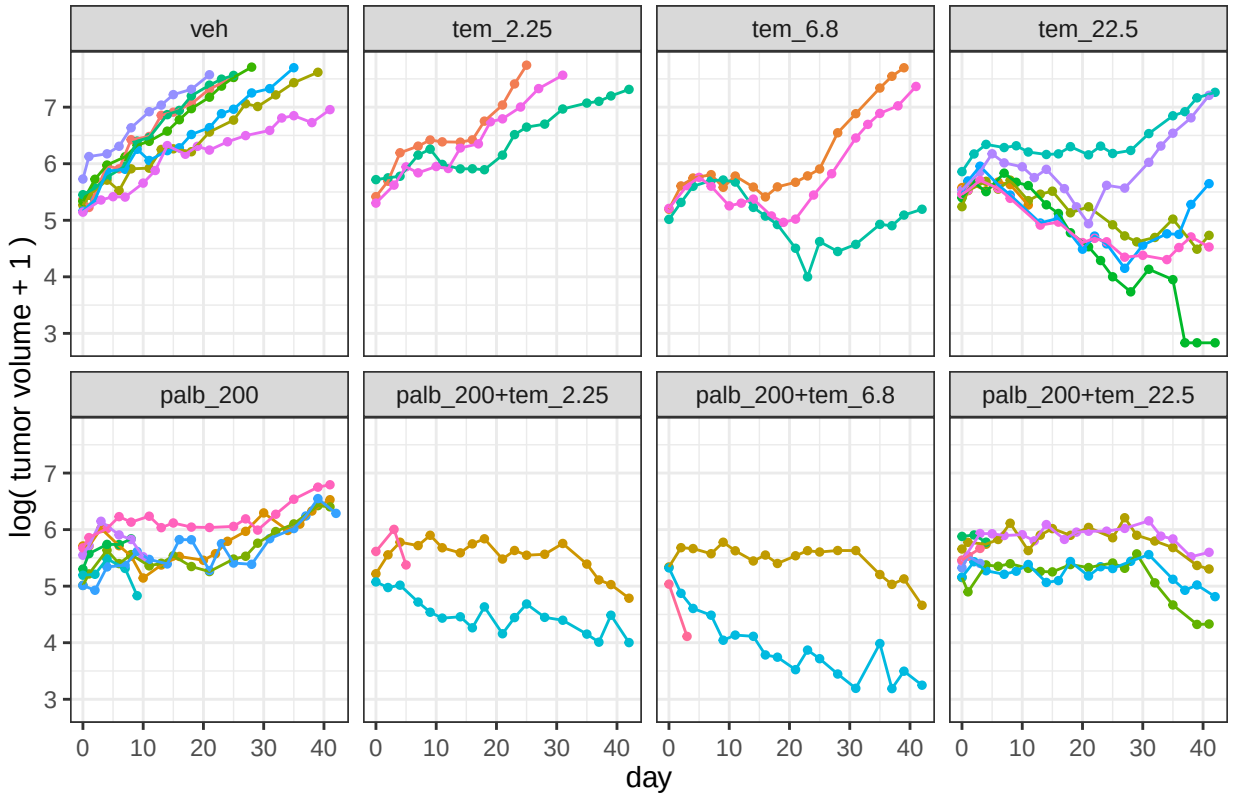
```
## # A tibble: 8 x 3
##   group_name        first_day last_day
##   <fct>                 <dbl>    <dbl>
## 1 veh                       0       60
## 2 tem_2.25                  0       49
```

```
## 3 tem_6.8                   0       60
## 4 tem_22.5                  0       60
## 5 palb_200                  0       60
## 6 palb_200+tem_2.25         0       60
## 7 palb_200+tem_6.8          0       60
## 8 palb_200+tem_22.5         0       60
```

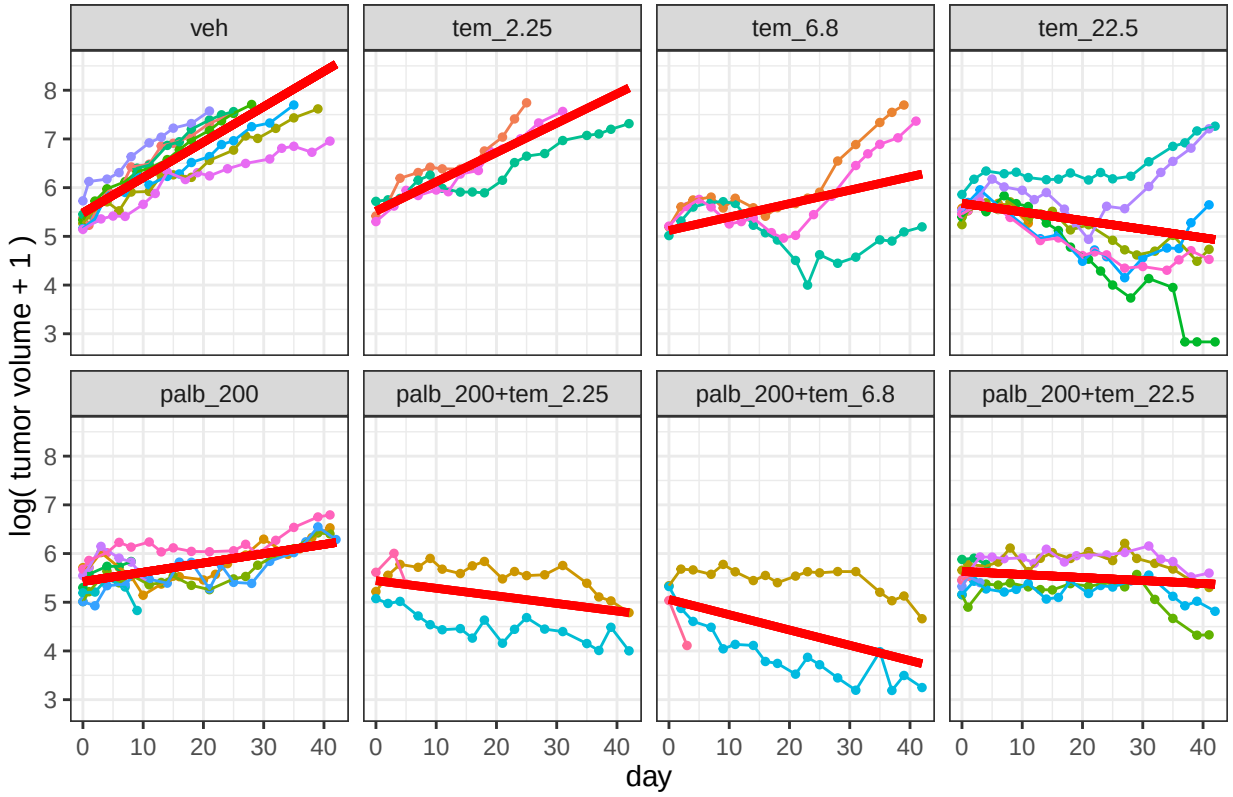
```
pred_palb_temo %>%
 draw_study(     y_label = 'TUMOR_VOLUME',
             title_label = 'palbociclib + temozolomide: tumor volume profiles by group'
           )
```



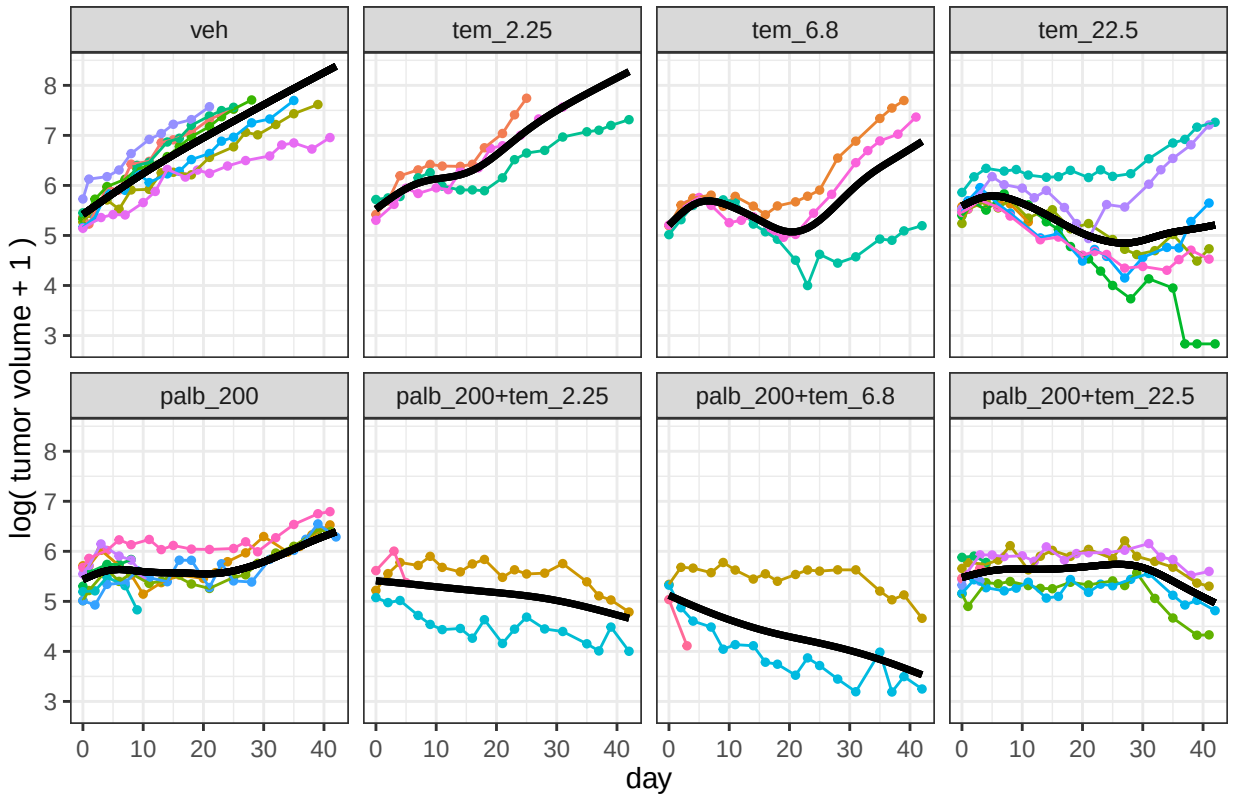

```
maeve_options( endpoint_name = 'y' ) # figures below will be on transformed endpoint.

pred_palb_temo %>% draw_study( )
```



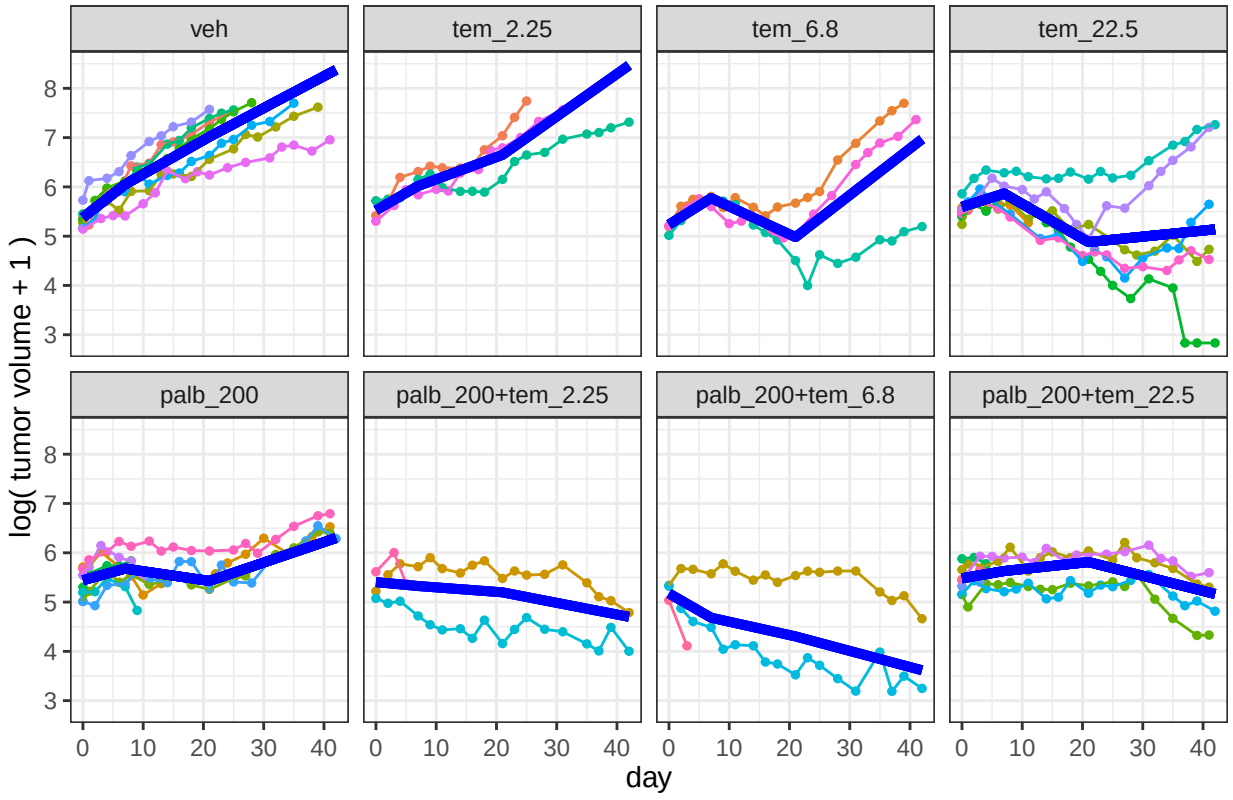

```
pred_palb_temo %>% draw_study( fit = 'linear' )
```



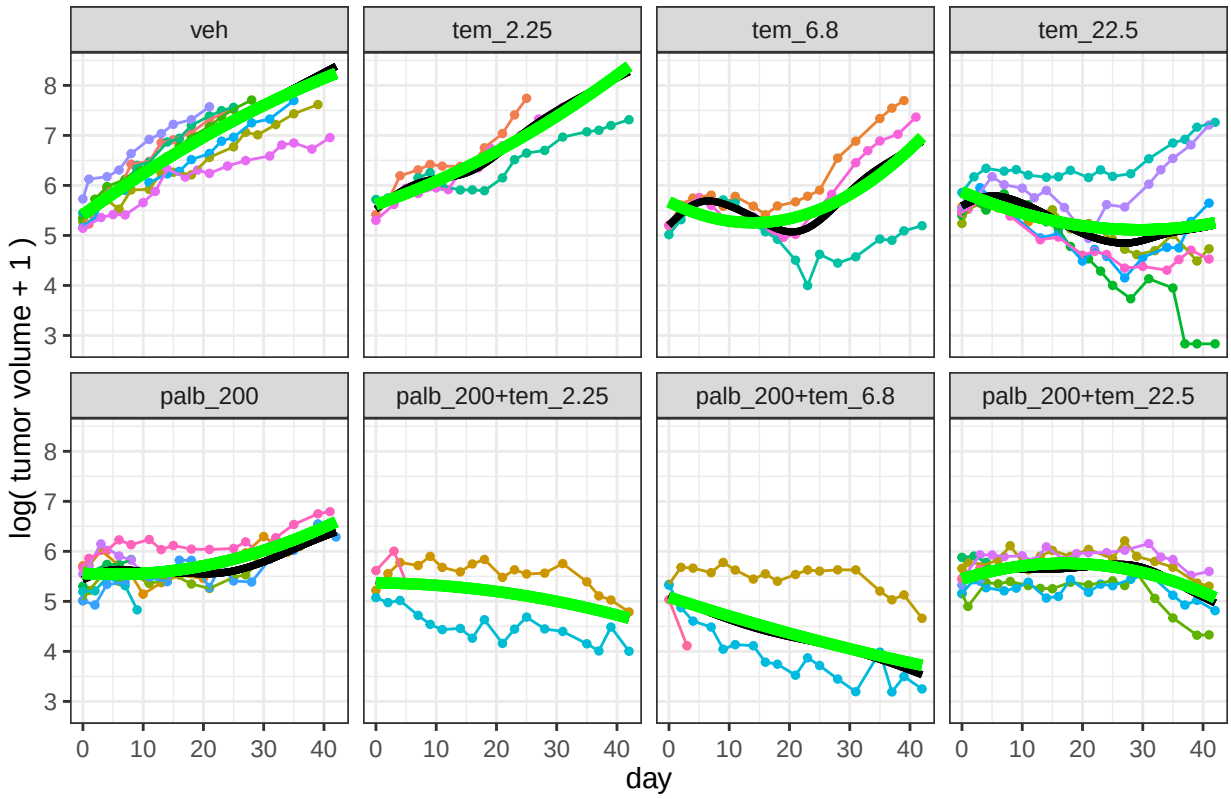
```
pred_palb_temo %>% draw_study( fit = 'spline' )
```



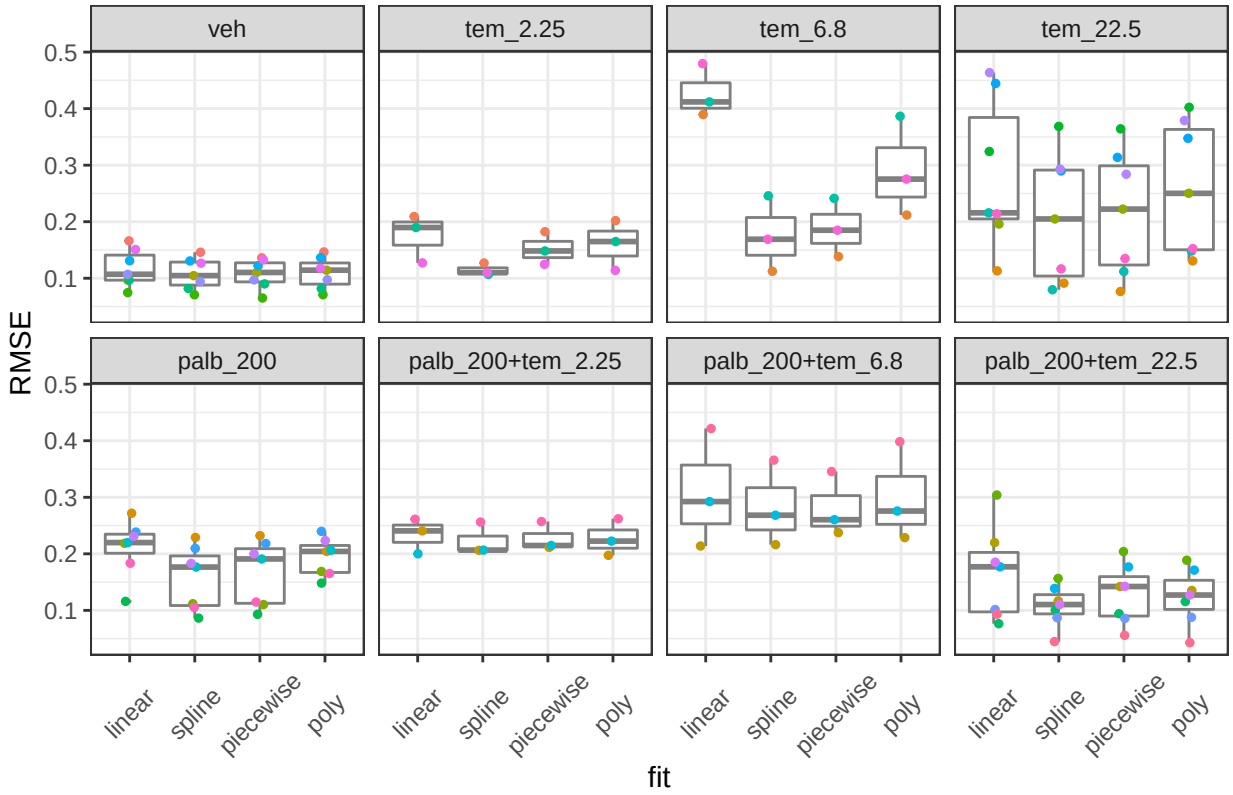

```
pred_palb_temo %>% draw_study( fit = 'piecewise' )
```



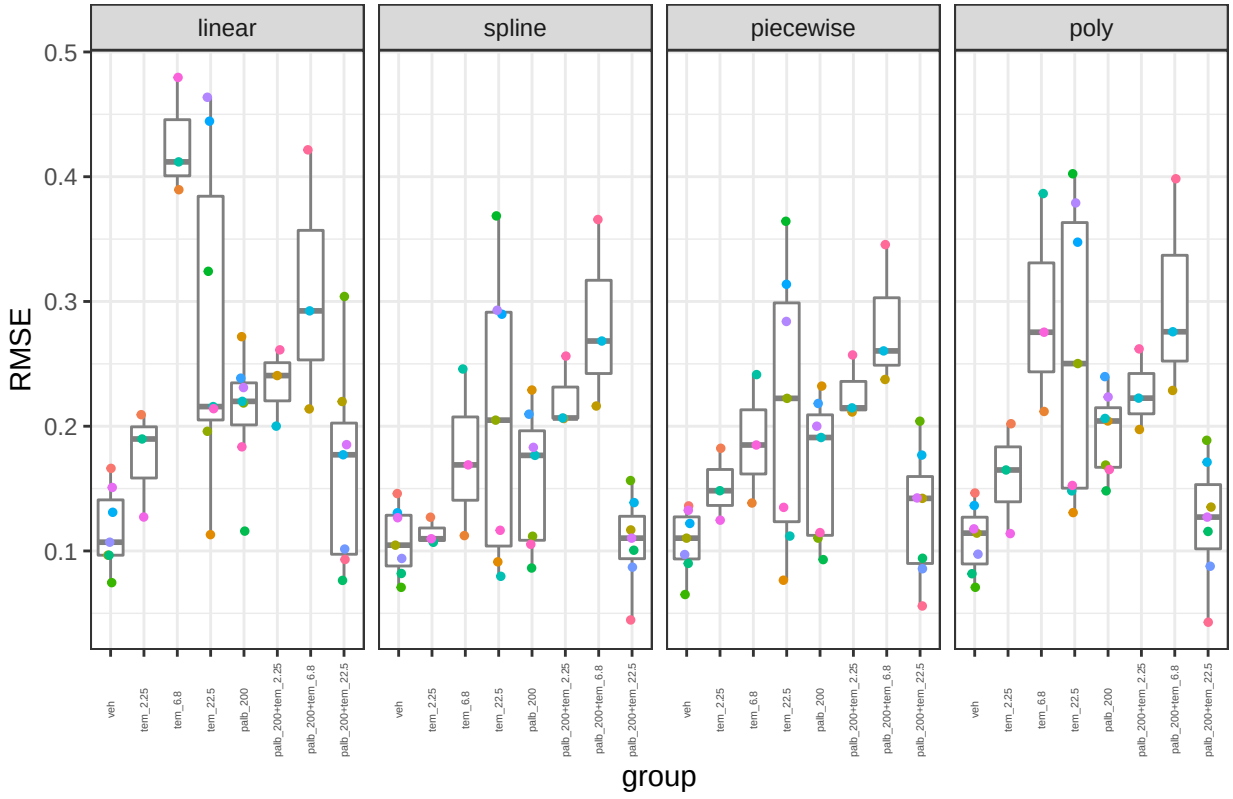
```
### overlay polynomial fits with spline fits on raw data:
 pred_palb_temo %>% draw_study( fit = c( 'poly', 'spline') )
```



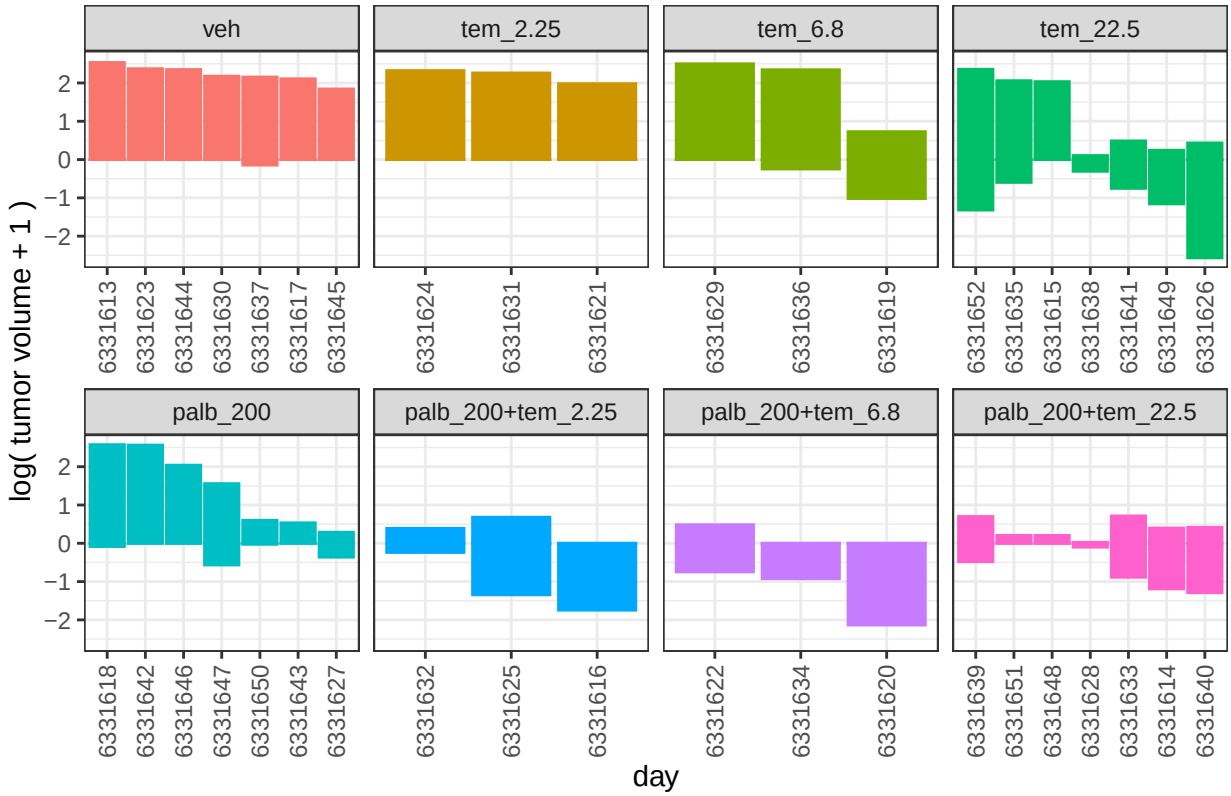

```
### draw_noise()

### For each animal and model, Show the RMS prediction error of the conditional fits to the data:
### ("conditional" = "predictions including BLUPs for each animal")

 pred_palb_temo_list[['RMSE_by_ID']] %>%
    draw_noise() + theme( axis.text.x = element_text( angle = 45, vjust = 0.5 ) )
```




```
### same values, but faceted instead by fit method:
 pred_palb_temo_list[['RMSE_by_ID']] %>%
    draw_noise(facet_by = 'fit') + theme( axis.text.x = element_text( angle = 90, size = 4 ) )
```



```
### draw_waterfall()
 palb_temo_01 %>%
  dplyr::mutate( y = log( 1 + TUMOR_VOLUME ) ) %>%
  draw_waterfall( endpoint_name = 'y', facet_char = 'group_name', ncol_value = 4 )
```




```
maeve_reset()
```

15 Use `compare_groups()` to summarize longitudinal model fits with error estimates and estimate requested contrasts.

The `compare_groups()` function takes fitted models and summarizes each metric (e.g., `linear`, `AUC`, `AUC_pwl`, etc.) into one number per requested comparison. Which comparisons are produced depends on the parameter `maeve_options('contrast')`, which can either be set globally or just passed explicitly to `compare_groups()`.

By default, the contrast is `"Identity"`, causing each group to be summarized to a single number, but with no inter-group comparison:

```
  maeve_options('contrast')
```

```
## [1] "Identity"
```

The supported contrast types are:

1. `"Identity"`: Summarize each metric into one number per group. No inter-group comparisons are performed, so the function name `compare_groups()` is arguably a misnomer in this case.
2. `"Dunnett"`: Summarize each metric into one number per group, then compare each "active" group to a common control, which is determined by `maeve_options("reference_Dunnett")`. By default, the common control is just the group named as the first factor level of `group_name`.

3. `"Tukey"`: Summarize each metric into one number per group, then make all pairwise comparisons.
4. `"Sequen"`: Summarize each metric into one number per group, then compare each group to the factor level just before it in the factor levels of `group_name`. This is mostly useful in dose escalation studies.
5. `"custom"`: Summarize each metric into one number per group, then compare each groups as specified by a user-provided custom contrast matrix to perform a Wald test for each row of the custom contrast matrix. See examples.

The LME slope can be obtained by setting `maeve_options( metric = "linear" )`.

The group summaries `eGaIT` and `eDOT` described in the manuscript are derived by setting `maeve_options( "metric" )` to `maeve_options( "metric" = "AUC")` or `maeve_options( "metric" = "ITGR")`, respectively, while leaving `maeve_options( "xrange_norm_method" = "slope_equivalent" )` (its default) and the parameter `..., subtract_starting_value = TRUE, ...` in `compare_groups()` (also its default). Hence, `eGaIT` and `eDOT` are special – albeit default – cases of the metrics available from `compare_groups()` by toggling between `metric = "AUC"` and `metric = "ITGR"` in `maeve_options()`.

The metrics `"AUC_pwl"` and `"ITGR_pwl"` are the analogs of those for `"AUC"` and `"ITGR"`, but derived from a piecewise linear mixed model fit instead of from a spline. Similarly, The metrics `"AUC_poly"` and `"ITGR_poly"` are the analogs of those for `"AUC"` and `"ITGR"`, but derived from a simple polynomial basis linear mixed model fit instead of from a spline. These are intended to be used mainly when there are very few (e.g., 3-5) distinct time points in a study, so that fitting a spline may be difficult or undefined.

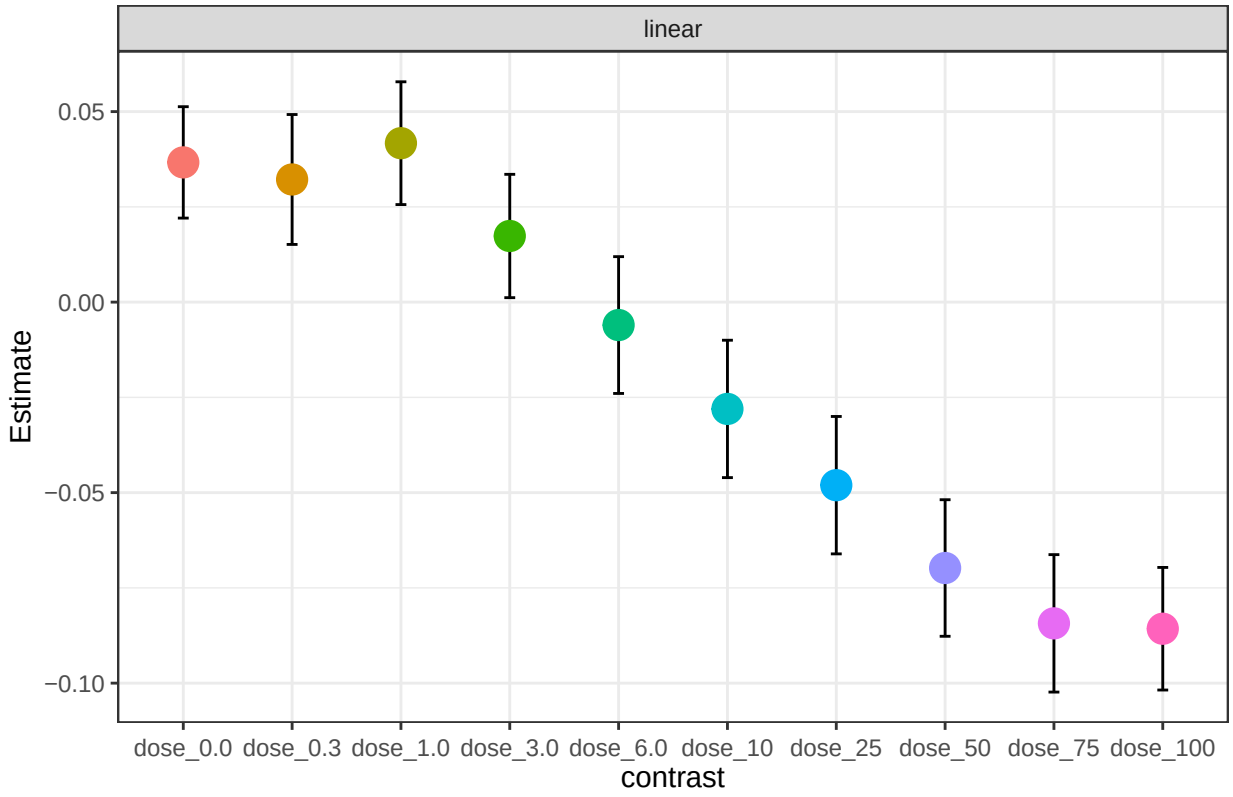
By default, each group is assigned a summary statistic with a 95% confidence interval. Setting `..., extended_output = TRUE, ...` in `compare_groups()` will further include in the comparison table output columns with standard errors, t-statistics, and p-values of the null hypothesis that the numeric contrast considered is zero, with p-values adjusted by the method in `adjustment_method` (for which `none` results in no adjustment).

Below are a number of worked examples summarizing estimated response curves. Note that when `maeve_options("contrast")` is `"Identity"`, there are not any actual comparisons *per se*, just reporting of the summary metric for each group (in which case `compare_groups()` is something of a misnomer). Various comparisons are performed when `"contrast"` is set to, e.g., `"Dunnett"` or `"Tukey"`. A `"custom"` option is also available for customized linear contrasts – see the example below.

```
### 7. compare_groups().

### (Recall that "models_vismo21" is from the vismo21 analysis above.)

### linear summary:

 cg <- models_vismo21 %>% compare_groups( metric = 'linear' )

 cg %>% with( data ) %>% with( effectDF ) %>%  format( digits = 3 )
```

```
##    metric contrast Estimate      lwr     upr
## 1  linear dose_0.0  0.03667  0.02205  0.0513
## 2  linear dose_0.3  0.03219  0.01514  0.0492
## 3  linear dose_1.0  0.04171  0.02560  0.0578
## 4  linear dose_3.0  0.01735  0.00117  0.0335
## 5  linear dose_6.0 -0.00602 -0.02397  0.0119
## 6  linear  dose_10 -0.02803 -0.04606 -0.0100
## 7  linear  dose_25 -0.04804 -0.06608 -0.0300
## 8  linear  dose_50 -0.06978 -0.08770 -0.0519
## 9  linear  dose_75 -0.08431 -0.10234 -0.0663
## 10 linear dose_100 -0.08572 -0.10181 -0.0696
```

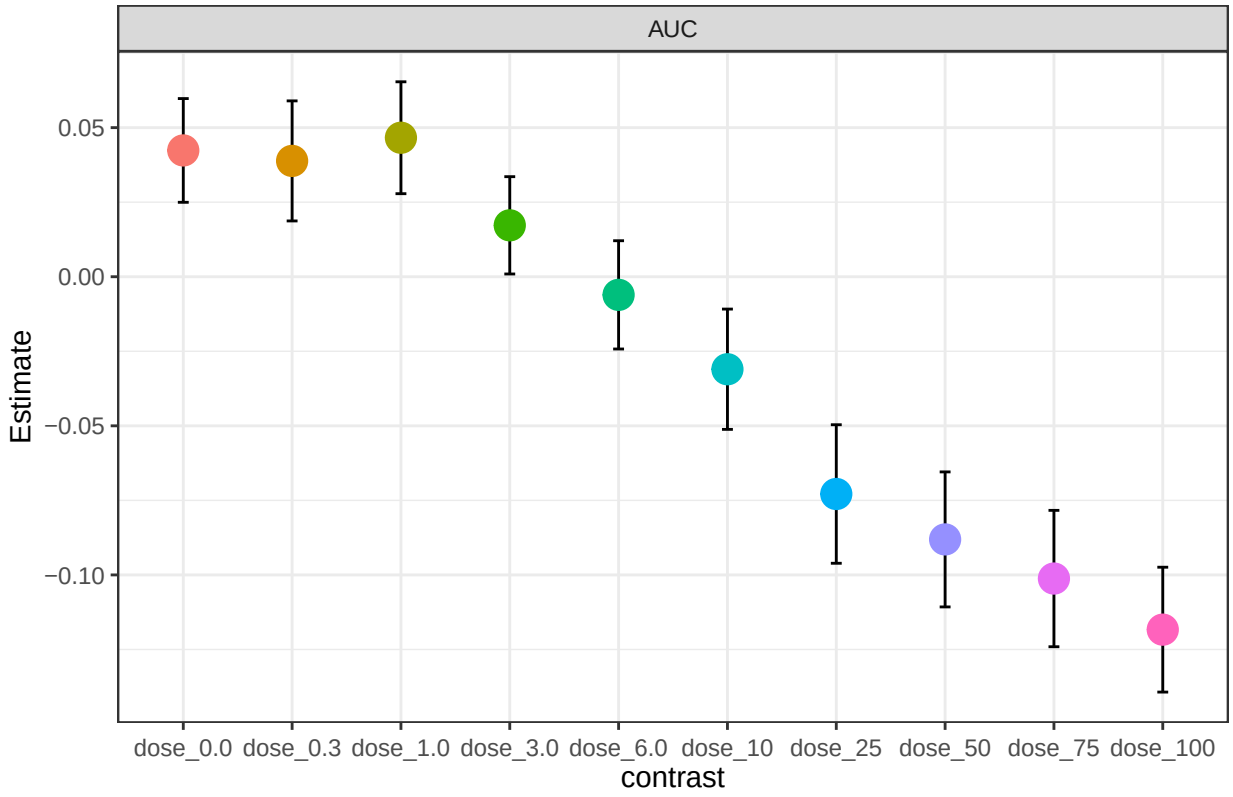
```
 cg %>% with( figures ) %>% with( figCI ) %>% print
```




```
### spline summary:
 cg <- models_vismo21 %>% compare_groups( metric = 'AUC' )

 cg %>% with( data ) %>% with( effectDF ) %>%  format( digits = 3 )
```

```
##    metric contrast Estimate       lwr     upr
## 1     AUC dose_0.0  0.04235  0.024958  0.0597
## 2     AUC dose_0.3  0.03884  0.018709  0.0590
## 3     AUC dose_1.0  0.04661  0.027850  0.0654
## 4     AUC dose_3.0  0.01724  0.000918  0.0336
## 5     AUC dose_6.0 -0.00608 -0.024239  0.0121
## 6     AUC  dose_10 -0.03102 -0.051192 -0.0109
## 7     AUC  dose_25 -0.07286 -0.096089 -0.0496
## 8     AUC  dose_50 -0.08809 -0.110728 -0.0655
## 9     AUC  dose_75 -0.10121 -0.124081 -0.0783
## 10    AUC dose_100 -0.11835 -0.139282 -0.0974
```

```
 cg %>% with( figures ) %>% with( figCI ) %>% print
```



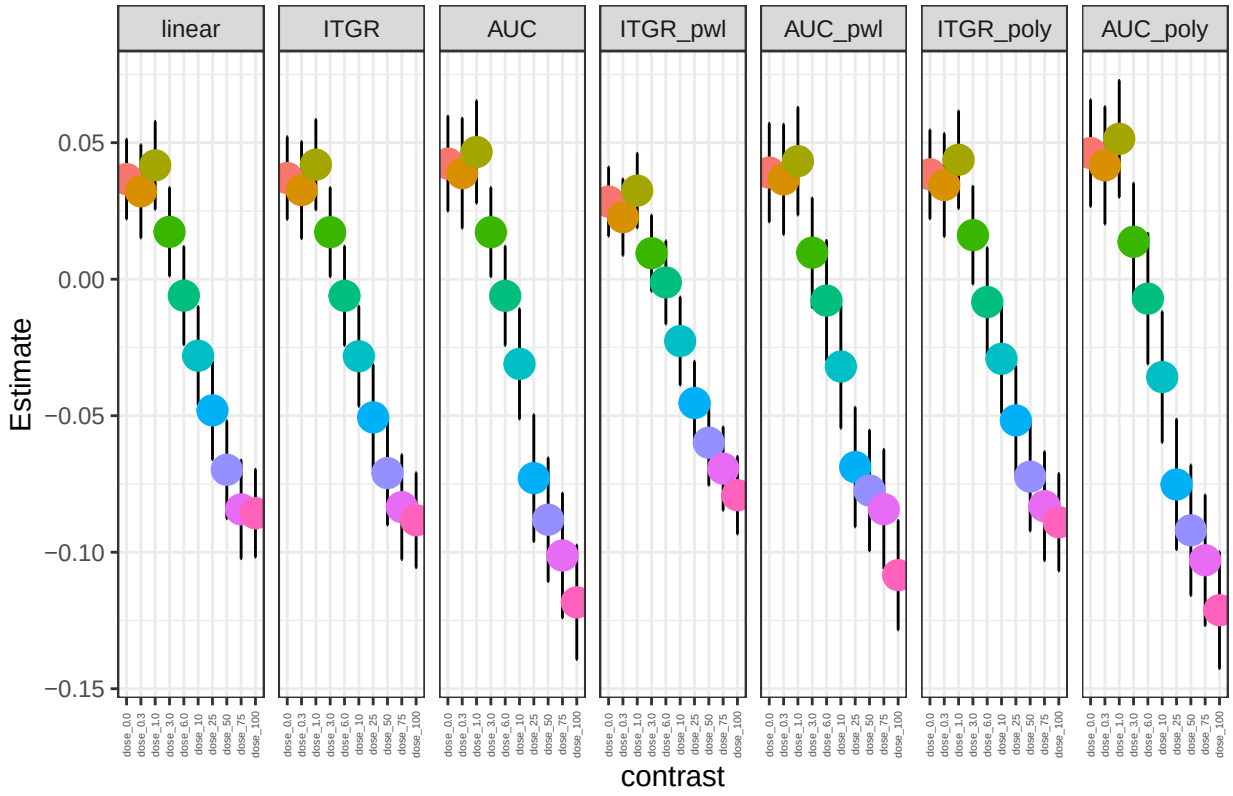

```
### All of them...

 maeve_options( break_points = c( 0, 7, 21, 42 ),
                ncol_value = 7,
                modeling_data_frame = pred_data_frame, # saved from earlier
                poly_object = poly_object_vismo21,     # saved from earlier
                poly_degree = 3
               )

### To get *all* metrics, you can access maeve_options( 'metrics_supported' ),
### assuming it has not been changed.
 cg <- models_vismo21 %>% compare_groups( metric = maeve_options( 'metrics_supported' ) )

### One line for each treatment group, one block of such lines for each metric.  A lot:
 cg %>% with( data ) %>% with( effectDF ) %>%  format( digits = 3 )
```

```
##       metric contrast Estimate       lwr      upr
## 1     linear dose_0.0  0.03667  0.022055  0.05129
## 2     linear dose_0.3  0.03219  0.015137  0.04923
## 3     linear dose_1.0  0.04171  0.025601  0.05782
## 4     linear dose_3.0  0.01735  0.001168  0.03354
## 5     linear dose_6.0 -0.00602 -0.023969  0.01193
## 6     linear  dose_10 -0.02803 -0.046062 -0.01000
## 7     linear  dose_25 -0.04804 -0.066079 -0.03001
## 8     linear  dose_50 -0.06978 -0.087703 -0.05186
## 9     linear  dose_75 -0.08431 -0.102341 -0.06627
## 10    linear dose_100 -0.08572 -0.101805 -0.06963
## 11      ITGR dose_0.0  0.03707  0.021969  0.05216
## 12      ITGR dose_0.3  0.03266  0.014899  0.05042
## 13      ITGR dose_1.0  0.04195  0.025420  0.05849
## 14      ITGR dose_3.0  0.01724  0.000918  0.03356
## 15      ITGR dose_6.0 -0.00608 -0.024239  0.01207
```

```
## 16      ITGR  dose_10 -0.02822 -0.046556 -0.00987
## 17      ITGR  dose_25 -0.05062 -0.069975 -0.03126
## 18      ITGR  dose_50 -0.07102 -0.090018 -0.05202
## 19      ITGR  dose_75 -0.08348 -0.102661 -0.06430
## 20      ITGR dose_100 -0.08830 -0.105662 -0.07093
## 21       AUC dose_0.0  0.04235  0.024958  0.05974
## 22       AUC dose_0.3  0.03884  0.018709  0.05897
## 23       AUC dose_1.0  0.04661  0.027850  0.06537
## 24       AUC dose_3.0  0.01724  0.000918  0.03356
## 25       AUC dose_6.0 -0.00608 -0.024239  0.01207
## 26       AUC  dose_10 -0.03102 -0.051192 -0.01085
## 27       AUC  dose_25 -0.07286 -0.096089 -0.04963
## 28       AUC  dose_50 -0.08809 -0.110728 -0.06546
## 29       AUC  dose_75 -0.10121 -0.124081 -0.07834
## 30       AUC dose_100 -0.11835 -0.139282 -0.09741
## 31  ITGR_pwl dose_0.0  0.02844  0.015815  0.04106
## 32  ITGR_pwl dose_0.3  0.02273  0.008744  0.03671
## 33  ITGR_pwl dose_1.0  0.03246  0.018821  0.04610
## 34  ITGR_pwl dose_3.0  0.00952 -0.004382  0.02343
## 35  ITGR_pwl dose_6.0 -0.00119 -0.016422  0.01405
## 36  ITGR_pwl  dose_10 -0.02266 -0.038754 -0.00656
## 37  ITGR_pwl  dose_25 -0.04535 -0.060617 -0.03009
## 38  ITGR_pwl  dose_50 -0.05987 -0.075479 -0.04427
## 39  ITGR_pwl  dose_75 -0.06937 -0.084628 -0.05410
## 40  ITGR_pwl dose_100 -0.07912 -0.093322 -0.06491
## 41   AUC_pwl dose_0.0  0.03906  0.020994  0.05713
## 42   AUC_pwl dose_0.3  0.03657  0.016405  0.05674
## 43   AUC_pwl dose_1.0  0.04325  0.023539  0.06295
## 44   AUC_pwl dose_3.0  0.00971 -0.010331  0.02975
## 45   AUC_pwl dose_6.0 -0.00784 -0.029862  0.01418
## 46   AUC_pwl  dose_10 -0.03197 -0.054567 -0.00938
## 47   AUC_pwl  dose_25 -0.06881 -0.090703 -0.04692
## 48   AUC_pwl  dose_50 -0.07741 -0.099526 -0.05530
## 49   AUC_pwl  dose_75 -0.08424 -0.106128 -0.06235
## 50   AUC_pwl dose_100 -0.10840 -0.128428 -0.08837
## 51 ITGR_poly dose_0.0  0.03842  0.022155  0.05468
## 52 ITGR_poly dose_0.3  0.03449  0.015651  0.05333
## 53 ITGR_poly dose_1.0  0.04375  0.025911  0.06159
## 54 ITGR_poly dose_3.0  0.01614 -0.001737  0.03401
## 55 ITGR_poly dose_6.0 -0.00836 -0.028290  0.01157
## 56 ITGR_poly  dose_10 -0.02915 -0.049113 -0.00919
## 57 ITGR_poly  dose_25 -0.05171 -0.071660 -0.03177
## 58 ITGR_poly  dose_50 -0.07231 -0.092229 -0.05239
## 59 ITGR_poly  dose_75 -0.08313 -0.103079 -0.06319
## 60 ITGR_poly dose_100 -0.08902 -0.106856 -0.07118
## 61  AUC_poly dose_0.0  0.04621  0.026700  0.06571
## 62  AUC_poly dose_0.3  0.04170  0.020203  0.06320
## 63  AUC_poly dose_1.0  0.05142  0.030034  0.07280
## 64  AUC_poly dose_3.0  0.01371 -0.007705  0.03512
## 65  AUC_poly dose_6.0 -0.00712 -0.031023  0.01678
## 66  AUC_poly  dose_10 -0.03583 -0.059754 -0.01191
## 67  AUC_poly  dose_25 -0.07519 -0.099088 -0.05129
## 68  AUC_poly  dose_50 -0.09197 -0.115855 -0.06809
## 69  AUC_poly  dose_75 -0.10295 -0.126847 -0.07905
## 70  AUC_poly dose_100 -0.12125 -0.142625 -0.09987
```

```
cg %>%
    with( figures ) %>%
    with( figCI +
```

```
          facet_wrap( ~ metric, ncol = 7 ) +
          theme( axis.text.x = element_text( angle = 90, size = 4 ) ) # rotate & shrink axis text labels
        ) %>%
    print
```




```
### Change the contrast and / or testing parameters:
 cg <- models_vismo21 %>% compare_groups( metric = 'AUC', contrast = 'Dunnett' )
 cg %>% with( data ) %>% with( effectDF ) %>%  format( digits = 3 )
```

```
##   metric            contrast Estimate     lwr      upr
## 1    AUC dose_0.3 - dose_0.0 -0.00351 -0.0293  0.02225
## 2    AUC dose_1.0 - dose_0.0  0.00426 -0.0205  0.02903
## 3    AUC dose_3.0 - dose_0.0 -0.02511 -0.0482 -0.00202
## 4    AUC dose_6.0 - dose_0.0 -0.04843 -0.0728 -0.02409
## 5    AUC  dose_10 - dose_0.0 -0.07337 -0.0992 -0.04758
## 6    AUC  dose_25 - dose_0.0 -0.11521 -0.1433 -0.08711
## 7    AUC  dose_50 - dose_0.0 -0.13044 -0.1581 -0.10280
## 8    AUC  dose_75 - dose_0.0 -0.14356 -0.1714 -0.11574
## 9    AUC dose_100 - dose_0.0 -0.16070 -0.1871 -0.13434
```

```
###
 cg <- models_vismo21 %>% compare_groups( metric = 'AUC', contrast = 'Dunnett', adjustment_method = 'none' )

 cg %>% with( data ) %>% with( effectDF ) %>%  format( digits = 3 )
```

```
##   metric            contrast Estimate     lwr      upr
## 1    AUC dose_0.3 - dose_0.0 -0.00351 -0.0221  0.01511
## 2    AUC dose_1.0 - dose_0.0  0.00426 -0.0136  0.02217
## 3    AUC dose_3.0 - dose_0.0 -0.02511 -0.0418 -0.00842
## 4    AUC dose_6.0 - dose_0.0 -0.04843 -0.0660 -0.03083
## 5    AUC  dose_10 - dose_0.0 -0.07337 -0.0920 -0.05473
## 6    AUC  dose_25 - dose_0.0 -0.11521 -0.1355 -0.09490
```

```
## 7    AUC  dose_50 - dose_0.0 -0.13044 -0.1504 -0.11046
## 8    AUC  dose_75 - dose_0.0 -0.14356 -0.1637 -0.12345
## 9    AUC dose_100 - dose_0.0 -0.16070 -0.1798 -0.14164
```

```
###
 cg <-
    models_vismo21 %>%
    compare_groups( metric = 'AUC', contrast = 'Dunnett', adjustment_method = 'holm', extended_output = TRUE)

 cg %>% with( data ) %>% with( effectDF ) %>%  format( digits = 3 )
```

```
##   metric            contrast Estimate   sigma     lwr      upr   tstat adj_method  pvalues
## 1    AUC dose_0.3 - dose_0.0 -0.00351 0.00950 -0.0293  0.02224  -0.369       holm 1.00e+00
## 2    AUC dose_1.0 - dose_0.0  0.00426 0.00914 -0.0205  0.02902   0.466       holm 1.00e+00
## 3    AUC dose_3.0 - dose_0.0 -0.02511 0.00852 -0.0482 -0.00203  -2.948       holm 9.60e-03
## 4    AUC dose_6.0 - dose_0.0 -0.04843 0.00898 -0.0728 -0.02410  -5.394       holm 2.76e-07
## 5    AUC  dose_10 - dose_0.0 -0.07337 0.00951 -0.0991 -0.04759  -7.713       holm 0.00e+00
## 6    AUC  dose_25 - dose_0.0 -0.11521 0.01036 -0.1433 -0.08713 -11.116       holm 0.00e+00
## 7    AUC  dose_50 - dose_0.0 -0.13044 0.01020 -0.1581 -0.10281 -12.793       holm 0.00e+00
## 8    AUC  dose_75 - dose_0.0 -0.14356 0.01026 -0.1714 -0.11575 -13.990       holm 0.00e+00
## 9    AUC dose_100 - dose_0.0 -0.16070 0.00972 -0.1870 -0.13436 -16.531       holm 0.00e+00
```

```
### Create a "custom" contrast matrix to summarize arbitrary linear combinations:

 vismo21_custom_contrast <-
  rbind(
         matrix( c(   -1,    0,    0,    0,    0,    1,    0,   0,   0,    0 ), ncol = 10 ),
         matrix( c( -1/3, -1/3, -1/3,    0,    0,    0,    0, 1/3, 1/3,  1/3 ), ncol = 10 ),
         matrix( c(    0,    0,    0, -1/4, -1/4, -1/4, -1/4, 1/3, 1/3,  1/3 ), ncol = 10 )
        )

  colnames( vismo21_custom_contrast ) <- levels( vismo21$group_name )

  rownames( vismo21_custom_contrast ) <- c('ten_minus_zero', 'high_minus_low', 'high_minus_middle')

  print( t( vismo21_custom_contrast  ) )
```

```
##          ten_minus_zero high_minus_low high_minus_middle
## dose_0.0             -1     -0.3333333         0.0000000
## dose_0.3              0     -0.3333333         0.0000000
## dose_1.0              0     -0.3333333         0.0000000
## dose_3.0              0      0.0000000        -0.2500000
## dose_6.0              0      0.0000000        -0.2500000
## dose_10               1      0.0000000        -0.2500000
## dose_25               0      0.0000000        -0.2500000
## dose_50               0      0.3333333         0.3333333
## dose_75               0      0.3333333         0.3333333
## dose_100              0      0.3333333         0.3333333
```

```
### Pass the customized contrast matrix to the compare_groups() function:
  cg <-
   models_vismo21 %>%
   compare_groups( metric = 'AUC',
                   contrast = 'custom',
                   custom_contrast = vismo21_custom_contrast,
                   extended_output = TRUE
                  )

  cg %>% with( data ) %>% with( effectDF ) %>%  format( digits = 3 )
```

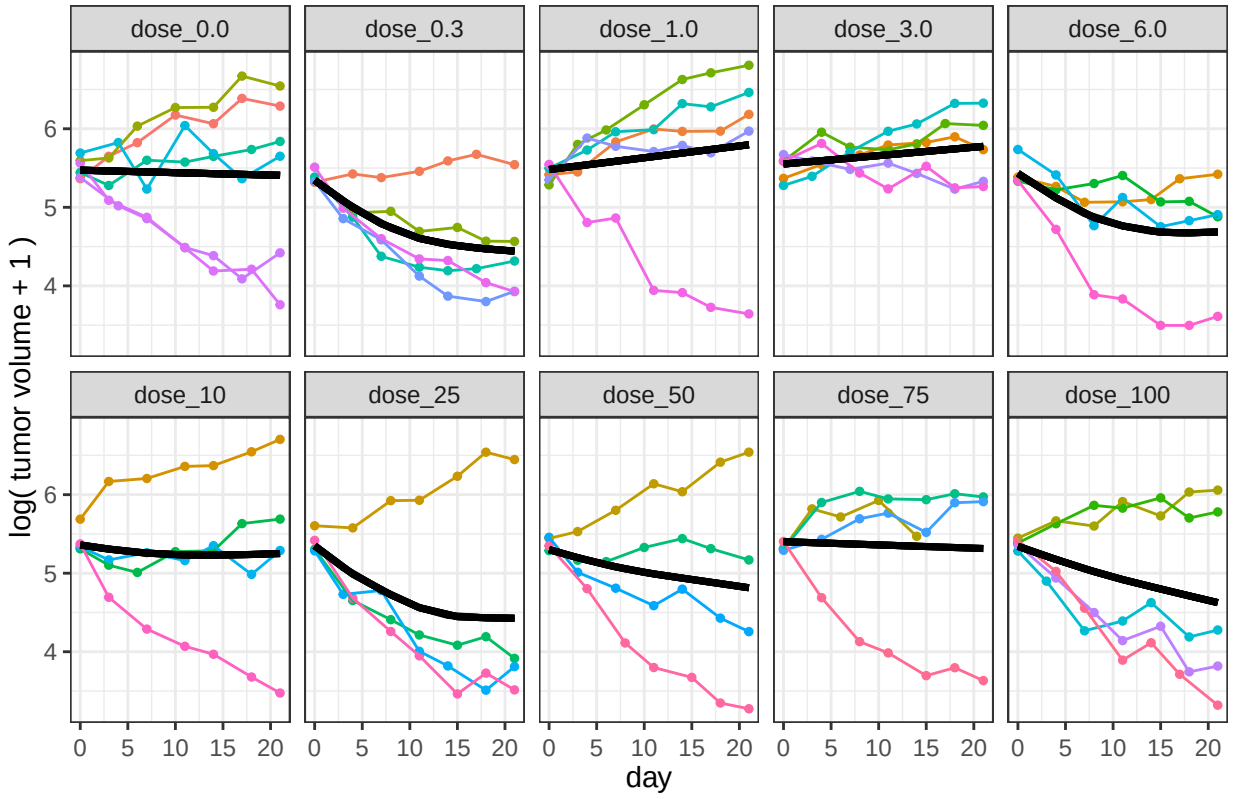
```
##   metric          contrast Estimate   sigma     lwr     upr  tstat  adj_method pvalues
```

```
## 1    AUC    ten_minus_zero  -0.0734 0.00951 -0.0958 -0.0509  -7.71 single-step       0
## 2    AUC    high_minus_low  -0.1451 0.00599 -0.1593 -0.1310 -24.22 single-step       0
## 3    AUC high_minus_middle  -0.0794 0.00576 -0.0930 -0.0658 -13.78 single-step       0
```

16 In-line permutation tests with permute_study()

The function `permute_study()` takes a study data.frame with `group_name` and `subject_ID` identifiers and randomly permutes the group membership of each subject, then returns the data frame. A `random_seed` parameter is included to facilitate reproducibility.

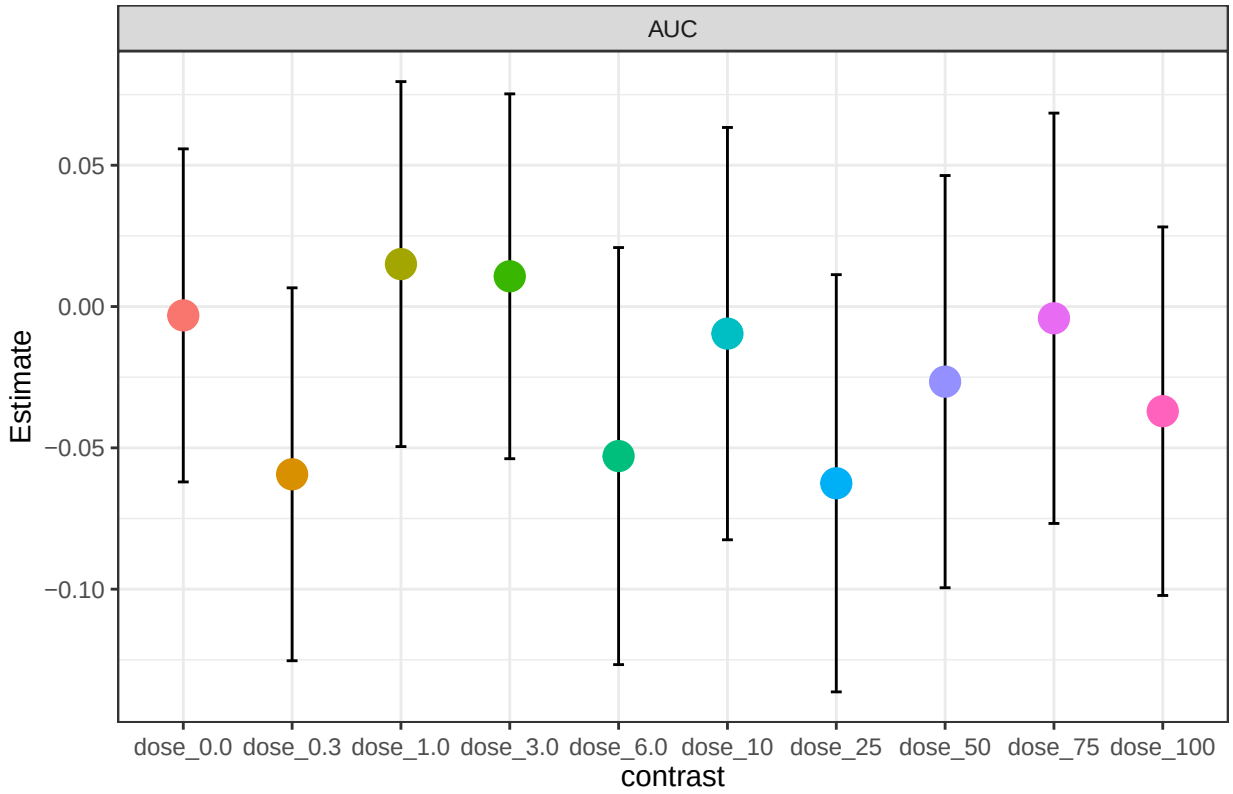
```
### 8. permute_study()
### Run simple permutation tests.

maeve_reset( )

RS <- 20180830 # random.seed value (optional)

vismo21 %>%
 permute_study( random_seed = RS ) %>%
 model_study                       %>%
 predict_study                     %>%
 draw_study( fit = 'spline', endpoint_name = 'y', ncol_value = 5 )
```




```
vismo21 %>%
 permute_study( random_seed = RS ) %>%
 model_study                       %>%
 compare_groups( metric = 'AUC' )  %>%
 with( data )                      %>%
 with( effectDF )                  %>%
 format( digits = 3 )
```

```
##    metric contrast Estimate     lwr     upr
## 1     AUC dose_0.0 -0.00313 -0.0621 0.05579
## 2     AUC dose_0.3 -0.05935 -0.1253 0.00662
## 3     AUC dose_1.0  0.01502 -0.0495 0.07958
## 4     AUC dose_3.0  0.01070 -0.0538 0.07525
## 5     AUC dose_6.0 -0.05291 -0.1267 0.02087
## 6     AUC  dose_10 -0.00958 -0.0825 0.06336
## 7     AUC  dose_25 -0.06253 -0.1363 0.01129
## 8     AUC  dose_50 -0.02657 -0.0995 0.04636
## 9     AUC  dose_75 -0.00415 -0.0767 0.06845
## 10    AUC dose_100 -0.03703 -0.1022 0.02817
```

```
vismo21 %>%
 permute_study( random_seed = RS ) %>%
 model_study                       %>%
 compare_groups( metric = 'AUC' )  %>%
 with( figures )                   %>%
 with( figCI )                     %>%
 print
```



17 Treatment-to-reference summaries with `generate_summary_table()`.

The "summary table" is a `Dunnett`-themed study summary output by default as a simple study summary. It assumes a designated reference group and focuses mainly on statistics comparing each of the other group regimens to this reference group. A little more explanation of the statistics is included in this section since these measures are not discussed in a manuscript. This functionality is **not necesarily recommended**. It provides support for a number of unpublished "legacy" statistics in use in the setting in which this package was developed, and so is included for back-comparison.

```r
### 9. generate_summary_table()

### Reset options for vismodegib dose-response analysis:

 maeve_reset() # reset existing options to defaults.

 maeve_options( ## maeve_options('metrics_supported') =
                ## c('linear', 'ITGR', 'AUC', 'ITGR_pwl', 'AUC_pwl', 'ITGR_poly', 'AUC_poly' )
                metric = maeve_options('metrics_supported'),
                modeling_data_frame = modeling_data_frame_vismo21,
                break_points = c( 0, 7, 14, 21 ),
                poly_degree = 3,
                poly_object = poly_object_vismo21
               )

### Generating a summary table is done for one chosen metric.
### Obviously it could be parallelized, but it's a slow operation (due in part  to parametric bootstrapping),
### so we haven't currently done that as it's not used that way.
   summary_linear     <- models_vismo21 %>% generate_summary_table( pred_data_frame, metric = 'linear' )
   ##
   summary_ITGR       <- models_vismo21 %>% generate_summary_table( pred_data_frame, metric = 'ITGR' )
   summary_AUC        <- models_vismo21 %>% generate_summary_table( pred_data_frame, metric =  'AUC' )
   ##
   summary_ITGR_pwl   <- models_vismo21 %>% generate_summary_table( pred_data_frame, metric = 'ITGR_pwl' )
   summary_AUC_pwl    <- models_vismo21 %>% generate_summary_table( pred_data_frame, metric =  'AUC_pwl' )
   ##
   summary_ITGR_poly  <- models_vismo21 %>% generate_summary_table( pred_data_frame, metric = 'ITGR_poly' )
   summary_AUC_poly   <- models_vismo21 %>% generate_summary_table( pred_data_frame, metric =  'AUC_poly' )
   ##

print( summary_AUC %>% head(4) ) # typical gRED summary table results accessed via IDA.
```

```
##      group N_first_day first_day integration_first_day integration_last_day fit_last_day last_day PR EOS_CR
## 1 dose_0.0           6         0                     0                   21           21       21  0      0
## 2 dose_0.3           5         0                     0                   21           21       21  0      0
## 3 dose_1.0           5         0                     0                   21           21       21  0      0
## 4 dose_3.0           5         0                     0                   21           21       21  0      0
##   effect_start effect_end effect_duration TTP_2X TTP_5X metric   Estimate     Diff_Ref       sigma
## 1      21.0000         21          0.0000   17.7     NA    AUC 0.04234917  0.000000000 0.000000000
## 2       3.9309         21         17.0691     NA     NA    AUC 0.03883934 -0.003509829 0.009502330
## 3      21.0000         21          0.0000   15.3     NA    AUC 0.04660828  0.004259111 0.009136903
## 4       0.0000         21         21.0000     NA     NA    AUC 0.01723652 -0.025112647 0.008518543
##           lwr          upr      tstat    pvalues xrange_norm_width firstDay_orig
## 1  0.00000000  0.000000000         NA         NA                21      240.2550
## 2 -0.02927195  0.022252287 -0.3693651 0.99994595                21      249.9890
## 3 -0.02051228  0.029030507  0.4661439 0.99963462                21      222.7022
## 4 -0.04820758 -0.002017713 -2.9479981 0.02506906                21      199.6556
##   firstDay_mean_originalScale_reference AUC_orig AUC_originalScale_reference TV_change TV_change_reference
## 1                               240.255 384.4930                     384.493 144.23798            144.238
## 2                               240.255 382.5589                     384.493 132.56985            144.238
## 3                               240.255 375.3776                     384.493 152.67537            144.238
## 4                               240.255 240.7940                     384.493  41.13835            144.238
##          TGI TGI_baseline
## 1  0.0000000     0.000000
## 2  0.5030336     8.089503
## 3  2.3707679    -5.849632
## 4 37.3736415    71.478839
```

```
print( summary_linear   %>% head(4) )
```

```
##      group N_first_day first_day integration_first_day integration_last_day fit_last_day last_day PR EOS_CR
## 1 dose_0.0           6         0                     0                  21           21       21  0      0
## 2 dose_0.3           5         0                     0                  21           21       21  0      0
## 3 dose_1.0           5         0                     0                  21           21       21  0      0
## 4 dose_3.0           5         0                     0                  21           21       21  0      0
##   effect_start effect_end effect_duration TTP_2X TTP_5X metric   Estimate     Diff_Ref       sigma
## 1           NA         NA              NA   18.9     NA linear 0.03667183  0.000000000 0.000000000
## 2           NA         NA              NA     NA     NA linear 0.03218513 -0.004486700 0.008021288
## 3           NA         NA              NA   16.6     NA linear 0.04171018  0.005038348 0.007769679
## 4           NA         NA              NA     NA     NA linear 0.01735387 -0.019317964 0.007789886
##           lwr         upr      tstat   pvalues xrange_norm_width firstDay_orig
## 1  0.00000000 0.000000000         NA        NA                21      252.3722
## 2 -0.02625350 0.017280101 -0.5593491 0.9985380                21      266.7653
## 3 -0.01604568 0.026122374  0.6484628 0.9955997                21      232.4604
## 4 -0.04045682 0.001820897 -2.4798777 0.0928704                21      199.2833
##   firstDay_mean_originalScale_reference AUC_orig AUC_originalScale_reference TV_change TV_change_reference
## 1                             252.3722 380.7515                   380.7515 128.37935            128.3794
## 2                             252.3722 381.6893                   380.7515 114.92392            128.3794
## 3                             252.3722 372.5599                   380.7515 140.09950            128.3794
## 4                             252.3722 240.6611                   380.7515  41.37774            128.3794
##          TGI TGI_baseline
## 1  0.0000000     0.000000
## 2 -0.2462783    10.480995
## 3  2.1514476    -9.129308
## 4 36.7931492    67.769166
```

```
print( summary_ITGR_pwl %>% tail(4) )
```

```
##       group N_first_day first_day integration_first_day integration_last_day fit_last_day last_day PR EOS_CR
## 7   dose_25           4         0                     0                  21           21       21  4      0
## 8   dose_50           4         0                     0                  21           21       21  4      0
## 9   dose_75           4         0                     0                  21           21       21  4      0
## 10 dose_100           5         0                     0                  21           21       21  5      0
##    effect_start effect_end effect_duration TTP_2X TTP_5X   metric    Estimate    Diff_Ref       sigma
## 7            NA         NA              NA     NA     NA ITGR_pwl -0.05243975 -0.09086999 0.008792934
## 8            NA         NA              NA     NA     NA ITGR_pwl -0.07028037 -0.10871061 0.008778359
## 9            NA         NA              NA     NA     NA ITGR_pwl -0.08261068 -0.12104092 0.008792934
## 10           NA         NA              NA     NA     NA ITGR_pwl -0.08865354 -0.12708377 0.008266694
##           lwr         upr     tstat pvalues xrange_norm_width firstDay_orig
## 7  -0.1147180 -0.06702200 -10.33443       0                21      213.1401
## 8  -0.1325191 -0.08490216 -12.38393       0                21      210.0773
## 9  -0.1448889 -0.09719294 -13.76570       0                21      217.1709
## 10 -0.1495045 -0.10466305 -15.37299       0                21      212.9676
##    firstDay_mean_originalScale_reference  AUC_orig AUC_originalScale_reference TV_change TV_change_reference
## 7                              233.8458 103.25907                   385.2784 -109.8810            151.4326
## 8                              233.8458  91.38845                   385.2784 -118.6889            151.4326
## 9                              233.8458  88.08513                   385.2784 -129.0858            151.4326
## 10                             233.8458  71.13444                   385.2784 -141.8331            151.4326
##         TGI TGI_baseline
## 7  73.19884     172.5610
## 8  76.27989     178.3773
## 9  77.13728     185.2430
## 10 81.53687     193.6609
```

```
print( summary_AUC_poly %>% tail(3) )
```

```
##       group N_first_day first_day integration_first_day integration_last_day fit_last_day last_day PR EOS_CR
## 8   dose_50           4         0                     0                  21           21       21  4      0
```

```
## 9   dose_75           4         0                    0                   21           21       21  4      0
## 10 dose_100           5         0                    0                   21           21       21  5      0
##    effect_start effect_end effect_duration TTP_2X TTP_5X   metric   Estimate   Diff_Ref      sigma        lwr
## 8            NA         NA              NA     NA     NA AUC_poly -0.0919713 -0.1381770 0.01101442 -0.1680480
## 9            NA         NA              NA     NA     NA AUC_poly -0.1029460 -0.1491517 0.01101917 -0.1790356
## 10           NA         NA              NA     NA     NA AUC_poly -0.1212450 -0.1674507 0.01033735 -0.1954855
##           upr     tstat pvalues xrange_norm_width firstDay_orig firstDay_mean_originalScale_reference
## 8  -0.1083060 -12.54510       0                21      217.0043                                231.93
## 9  -0.1192678 -13.53565       0                21      223.7284                                231.93
## 10 -0.1394160 -16.19861       0                21      216.1371                                231.93
##    AUC_orig AUC_originalScale_reference TV_change TV_change_reference      TGI TGI_baseline
## 8  90.63241                     386.005 -126.3719             154.075 76.52040     182.0197
## 9  87.38633                     386.005 -136.3421             154.075 77.36135     188.4907
## 10 70.62451                     386.005 -145.5126             154.075 81.70373     194.4427
```

```
### You can also pass multiple metrics if this is desirable.  It might get slow, but is handy to compare
### summary table statistics across different metrics.  Here, e.g., we compare TGI_baseline computed
### using the metrics 'linear', and 'AUC':
 summary_linear_and_AUC = models_vismo21 %>% generate_summary_table( pred_data_frame, metric = c('linear','AUC') )
 summary_linear_and_AUC %>% dplyr::select( group, metric, TGI_baseline ) %>% tidyr::spread( key = metric, -group )
```

```
##       group     linear        AUC
## 1  dose_0.0   0.000000   0.000000
## 2  dose_0.3  10.480995   8.089503
## 3  dose_1.0  -9.129308  -5.849632
## 4  dose_3.0  67.769166  71.478839
## 5  dose_6.0 111.124520 110.018841
## 6   dose_10 143.880197 143.887731
## 7   dose_25 148.874564 174.713311
## 8   dose_50 166.404573 182.052168
## 9   dose_75 179.072966 192.266437
## 10 dose_100 166.282867 196.889486
```

17.1 End of Study Complete Response and Partial Response

Two kinds of responses are assessed and recorded. Note that in both cases, the "math" of response-determination is done on the original scale of the endpoint rather than the transformed scale (e.g., on "tumor volume" rather than "log-transformed tumor volume"). These are computed from the observed values rather than model-fitted values.

17.1.1 End of Study Complete Response (`EOS_CR`)

A complete response is one for which a tumor is not detectable at the last observation. Specifically, the tumor must have a value at or below the value `EOS_CR_minval` in `maeve_options( "EOS_CR_minval" )`. By default, `EOS_CR_minval = 0` but can be changed to account for the endpoint and measurement system (e.g., if "less than detectable" for an endpoint were coded as a common, low number). If `TUMOR_VOLUME` were kept as the `endpoint_name` variable, then a value of `maeve_options( EOS_CR_minval = 50)` would cause any tumor with a final reading at or below 50 mm^3 to be classified as a complete response.

17.1.2 Partial Response (`PR`)

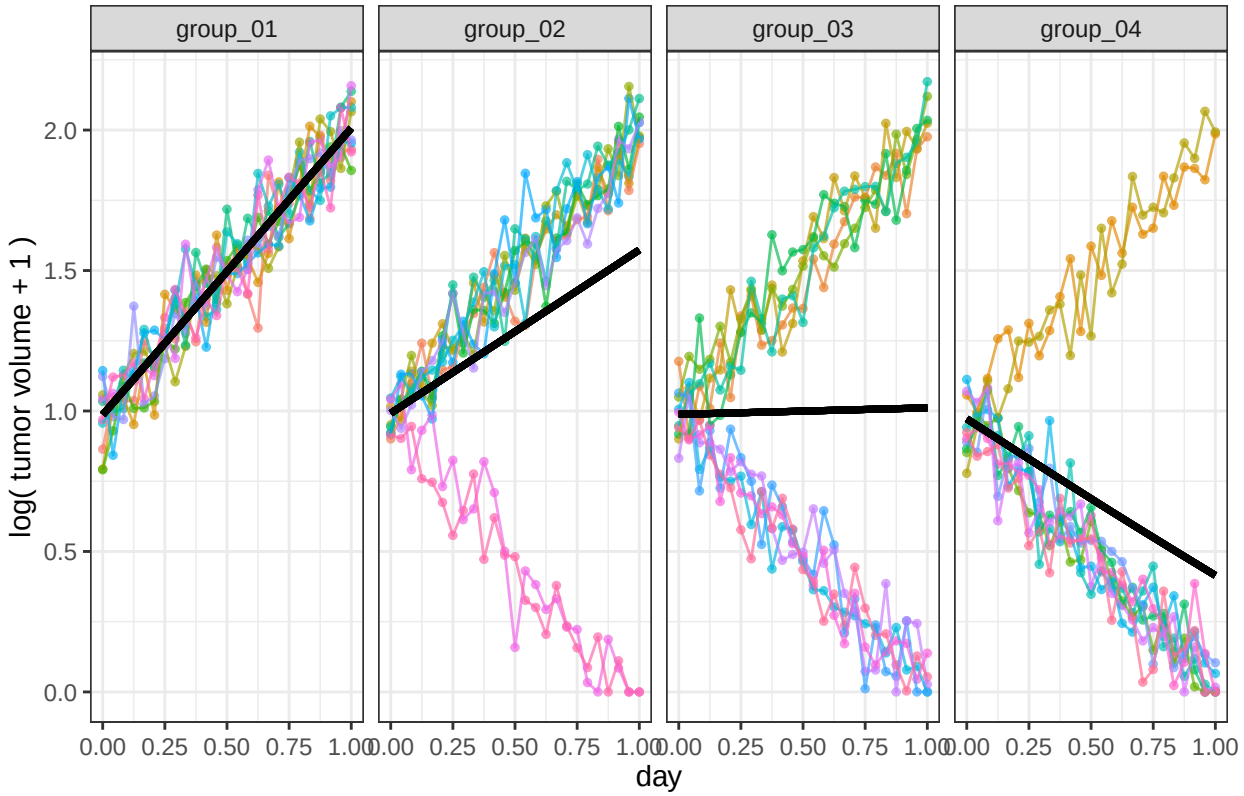
A partial response is a tumor that is not an End of Study Complete Response, but for which at least one observation fell at or below a specified fraction of that tumor's baseline size. The "specified fraction" is controlled by the parameter `PR_threshold` in `maeve_options( "PR_threshold" )` and passed to the function

`generate_summary_table()`. By default, `PR_threshold = 0.5` in `maeve`. Note that by construction, `PR` and `EOS_CR` are mutually exclusive categories.

17.1.3 A demonstration of counted responses with simulated data

These two counts are next demonstrated with the simulated data set `bifurcatedGrowthSim`. This data set has values on the unit interval with no censoring.

```
maeve_reset() # set all options to package defaults.

maeve_options( metric = 'AUC', ncol_value = 4, truncate_fit = TRUE )

## An R data file (".rda" format) is stored in /inst/extdata/data/
load( system.file( 'extdata', 'data', 'bifurcatedGrowthSim.rda',  package = 'maeve' ) )

### Fit models
model_list <- bifurcatedGrowthSim %>% model_study( number_basis_vecs = 5 )

### Set endpoint name for figures:
maeve_options( endpoint_name = 'y' ) # 'y' will be the transformed response -- usually what we want for graphics.

### Predict responses from models
model_list %>%
 predict_study %>%
 draw_study( fit = 'spline', endpoint_name = 'y', alpha_value = 0.67 ) %>%
 print
```




```
### Set prediction grid options and store predictions in a data frame
maeve_options( x_pred_type = 'grid', x_pred_spacing = 0.025 )
pred_data_frame = predict_study( model_list )
```

Next, run the `maeve::generate_summary_table()` and select out the reported counts.

```
Response_tab <-
 generate_summary_table( model_list,
                         pred_data_frame,
                         xmin = 0, xmax = 1,
                         x_time_spacing = 0.01,
                         metric = 'AUC',
                         PR_threshold = 0.50,
                         ##
                         conf_int = "none"
                        )

Response_tab %>% dplyr::select( group, N_first_day, PR, EOS_CR )
```

```
##      group N_first_day PR EOS_CR
## 1 group_01          10  0      0
## 2 group_02          10  0      2
## 3 group_03          10  3      2
## 4 group_04          10  4      4
```

Since this simulated data set has no censoring and linear (albeit bifurcated) trends, a simple if informal check on the results is to filter out the rows at the final time point (`DAY_OF_STUDY == 1`) with small tumor volumes. Based on the picture of log tumor volume, we picked `TUMOR_VOLUME < 4`. The filtered values are ordered first by group, then in descending order of `TUMOR_VOLUME` within group. The observed counts should recapitulate the count table just above.

```
bifurcatedGrowthSim %>%
 dplyr::filter( DAY_OF_STUDY >= 1, TUMOR_VOLUME < 4 ) %>%
 dplyr::select( group_name, animalID, DAY_OF_STUDY, TUMOR_VOLUME ) %>%
 dplyr::arrange( group_name, desc( TUMOR_VOLUME ) ) %>%
 format( digits = 3 )
```

```
##    group_name animalID DAY_OF_STUDY TUMOR_VOLUME
## 1    group_02   id_019            1       0.0000
## 2    group_02   id_020            1       0.0000
## 3    group_03   id_029            1       0.1479
## 4    group_03   id_030            1       0.0553
## 5    group_03   id_028            1       0.0270
## 6    group_03   id_026            1       0.0000
## 7    group_03   id_027            1       0.0000
## 8    group_04   id_037            1       0.1093
## 9    group_04   id_036            1       0.0679
## 10   group_04   id_038            1       0.0167
## 11   group_04   id_034            1       0.0035
## 12   group_04   id_033            1       0.0000
## 13   group_04   id_035            1       0.0000
## 14   group_04   id_039            1       0.0000
## 15   group_04   id_040            1       0.0000
```

Finally, we connect this back to the `tally_study()` function discussed earlier. If we want *only* the partial and complete responses, we could skip using `generate_summary_table()` and instead compute them in `tally_study()`:

```
    maeve_reset()
    tally_study(bifurcatedGrowthSim, response = c('PR', 'EOS_CR') ) %>%
    ## select out a subset of columns for presentation and round off the digits:
    dplyr::select( group_name, N_in_group, PR:EOS_CR_SE_log_OR ) %>%
    format( digits = 3 )
```

```
##   group_name N_in_group PR PR_log_OR PR_SE_log_OR EOS_CR EOS_CR_log_OR EOS_CR_SE_log_OR
```

```
## 1   group_01        10  0      0.00         0.00      0         0.00           0.00
## 2   group_02        10  0      0.00         2.05      2         1.82           1.62
## 3   group_03        10  3      2.28         1.59      2         1.82           1.62
## 4   group_04        10  4      2.68         1.57      4         2.68           1.57
```

17.2 Effect Duration statistics

Effect duration is a relative metric, pre-supposing a chosen control / reference group with growth curve $g_0(t)$ against which a treatment group with growth curve $g_1(t)$ is compared over a time interval $[a, b]$. On this interval, the time of effect start T_s & effect end T_e are defined. Effect duration is the time elapsed between the effect start and end.

1. Time of the start of efficacy T_s is defined as the first time in $[a, b]$ such that the growth rate for treatment is less than that for control, i.e.,
$$T_s = \min_{t \in [a,b]} \{t : g_1'(t) < g_0'(t)\}$$
This time can be (and often is) at an interval endpoint.
2. Time of the end of efficacy T_e is then the first time after efficacy starts such that the growth rate for treatment matches or exceeds the growth rate for control at time $t = T_s$, i.e.,
$$T_e = \min_{t \in [T_s,b]} \{t : g_1'(t) \geq g_0'(T_s)\}$$
This time can also be at an interval endpoint.
3. The effect duration is defined as the time elapsed time $T_e - T_s$.

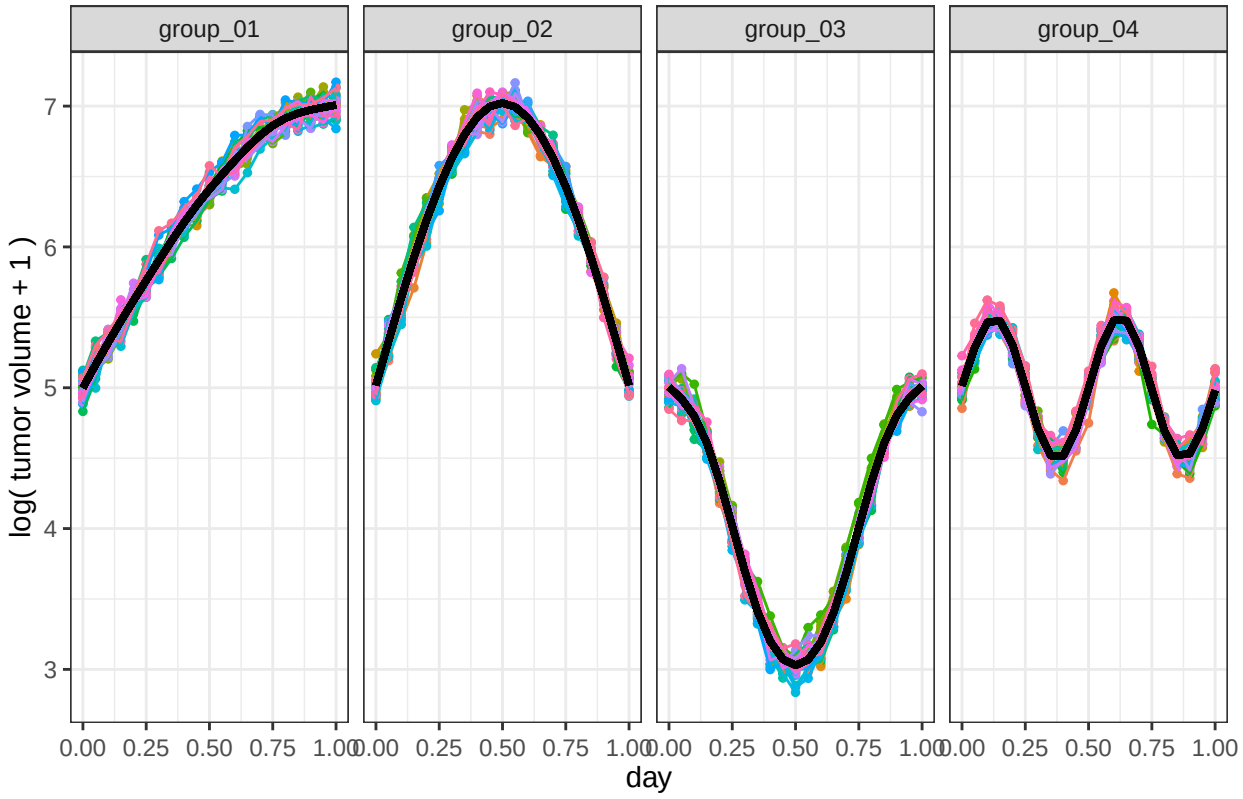
The effect duration statistics are computed only from the spline model (not linear, piecewise linear, or simple polynomial), so they will be `NA` unless `metric = "ITGR"` or `metric = "AUC"`.

Here is the model fitting with the default spline summary measures:

```
 maeve_reset() # set all options to package defaults.

 maeve_options( ncol_value = 4, truncate_fit = TRUE )


 ## An R data file (".rda" format) is stored in /inst/extdata/data/
 load( system.file( 'extdata', 'data', 'effect_duration_trig_sim.rda',  package = 'maeve' ) )

### Set working data
### data( effect_duration_trig_sim, package = 'maeve' )



## ### Presence pattern ("0" or "1") for filtered data:
##  effect_duration_trig_sim %>%
##   dplyr::select( group_name, DAY_OF_STUDY ) %>%
##   unique %>%
##   table  %>%
##   t()    %>%
##   print

### Fit models
  model_list <- effect_duration_trig_sim %>% model_study( number_basis_vecs = 8 )
```

```
## boundary (singular) fit: see ?isSingular
```

```
### Set endpoint name for figures:
 maeve_options( endpoint_name = 'y' ) # 'y' is the transformed response & usually what we want for graphics.

### Predict responses from models
 model_list %>%
  predict_study %>%
  draw_study( fit = 'spline', endpoint_name = 'y' ) %>%
  print
```




```
### run some compare_groups() analyses just to check against the later results in
### maeve::generate_summary_table():
 cg_Identity_AUC  <-
  compare_groups( model_list, metric = 'AUC',  xmin = 0, xmax = 1, contrast = 'Identity', draw_figure = FALSE )
 print( cg_Identity_AUC$data$effectDF )
```

```
##   metric contrast    Estimate         lwr         upr
## 1    AUC group_01  2.54986444  2.49489499  2.60483389
## 2    AUC group_02  2.54375340  2.48732110  2.60018569
## 3    AUC group_03 -1.99768797 -2.05364326 -1.94173269
## 4    AUC group_04 -0.03243146 -0.08847461  0.02361169
```

```
### Use the 'ITGR' metric over [0, 1], which gives results quite different from 'AUC':
 cg_Identity_ITGR <-
  compare_groups( model_list, metric = 'ITGR', xmin = 0, xmax = 1, contrast = 'Identity', draw_figure = FALSE )
 print( cg_Identity_ITGR$data$effectDF )
```

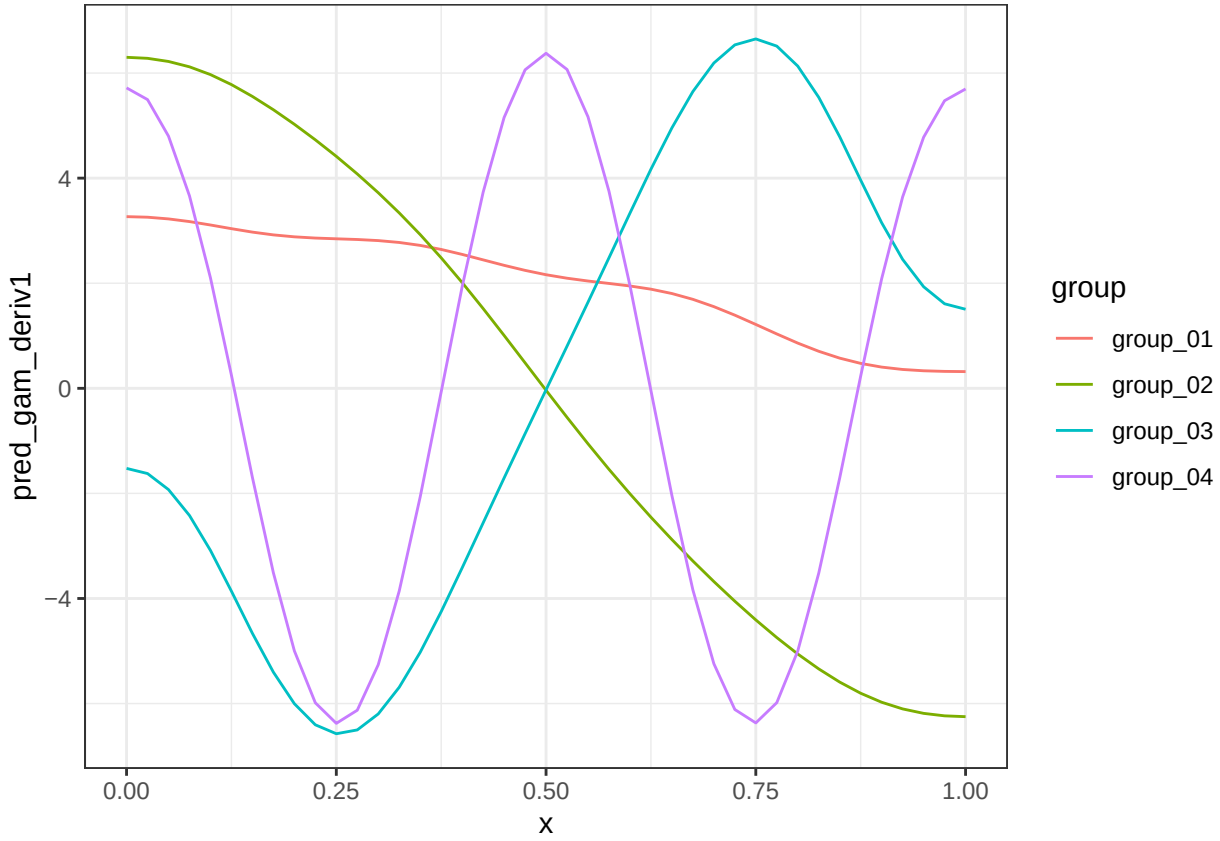
```
##   metric contrast      Estimate         lwr         upr
## 1   ITGR group_01  2.0072999882  1.96605813 2.048541844
## 2   ITGR group_02  0.0001976487 -0.04300768 0.043402980
## 3   ITGR group_03  0.0069519380 -0.03563956 0.049543441
## 4   ITGR group_04 -0.0348318604 -0.07695092 0.007287199
```

```
### Dunnett comparisons with 'AUC':
 cg_Dunnett  <-
  compare_groups( model_list, metric = 'AUC', xmin = 0, xmax = 1, contrast = 'Dunnett', draw_figure = FALSE )
 print( cg_Dunnett$data$effectDF )
```

```
##   metric            contrast     Estimate         lwr         upr
## 1    AUC group_02 - group_01 -0.006111045 -0.08039672  0.06817463
## 2    AUC group_03 - group_01 -4.547552415 -4.62151655 -4.47358828
## 3    AUC group_04 - group_01 -2.582295899 -2.65631916 -2.50827264
```

Here are the estimated first derivatives of the trigonometric growth curves:

```
### Set prediction grid options and store predictions in a data frame
 pred_data_frame = predict_study( model_list, x_pred_type = 'grid', x_pred_spacing = 0.025 )

### Draw numerically estimated first derivatives for each estimated group-level response:
deriv_norm_fig <-
 pred_data_frame %>%
  dplyr::select( group, x, pred_gam_deriv1 ) %>%
  unique() %>%
  ggplot( aes( x, pred_gam_deriv1, colour = group ) ) +
  geom_line( ) # + geom_vline( xintercept = c( .4174, .5796 ), lty = 3 )

print( deriv_norm_fig )
```



Here are the estimates of effect duration statistics, with confidence intervals based on parametric bootstrapping.

```
EffectDuration_tab <-
 generate_summary_table( model_list,
                         pred_data_frame,
                         xmin = 0, xmax = 1,
```

```
                          x_time_spacing = 0.01,
                          metric = 'AUC',
                          ##
                          conf_int = "none"
                         )

EffectDuration_tab %>%
 dplyr::select( group, effect_start, effect_end, effect_duration ) %>%
 print
```

```
##      group effect_start effect_end effect_duration
## 1 group_01       1.0000     1.0000          0.0000
## 2 group_02       0.3644     1.0000          0.6356
## 3 group_03       0.0000     0.5986          0.5986
## 4 group_04       0.0841     0.4174          0.3333
```

```
if(FALSE){ ## ERROR note (20200625):
## There is a known bug to be resolved in
## getting confidence intervals for the
## EffectDuration via the parametric bootstrap.

EffectDuration_tab_with_CI <-
 generate_summary_table( model_list,
                          pred_data_frame,
                          xmin = 0, xmax = 1,
                          x_time_spacing = 0.01,
                          metric = 'AUC',
                          ##
                          conf_int = "Effect_Duration"
                         )

EffectDuration_tab_with_CI %>%
 dplyr::select( group, effect_start,    CI_lower_Eff_Start, CI_upper_Eff_Start ) %>%
 format( digits = 4 ) %>%
 print

EffectDuration_tab_with_CI %>%
 dplyr::select( group, effect_end,      CI_lower_Eff_End,   CI_upper_Eff_End   ) %>%
 format( digits = 4 ) %>%
 print

EffectDuration_tab_with_CI %>%
 dplyr::select( group, effect_duration, CI_lower_Eff_Dur,   CI_upper_Eff_Dur   ) %>%
 format( digits = 4 ) %>%
 print
} ## end of ' if(FALSE){ ## ERROR note (20200625): ...'
```

17.3 Legacy metrics `TGI` and `TGIb`

In the special case that attention is restricted to so-called Dunnett comparisons (i.e., compare the effect of each treatment to a common reference group), the gRED legacy reporting tool reports two summaries of treatment effect, which we include here. These are referred to as tumor growth inhibition (TGI) and tumor growth inhibition baseline-normalized (TGIb).

17.3.1 Tumor Growth Inhibition (TGI)

Tumor growth inhibition is the ratio between treatment and control of areas under the respective growth curves on the *original* scale (i.e., any transformation of the raw measurements is inverted point-wise before numerical integration). We introduce the notation $V_{\beta_i}(t) \equiv h^{-1}\left(g_{\beta_i}(t)\right)$ for this inverted function with the mnemonic that $V_{\beta_i}(t)$ describes the tumor *volume* over the study times $t \in [a, b]$. Without loss of generality, denote by $i = 1$ the reference group. Then for $i \in 2, .., I$, define the TGI of group i relative to reference by

$$\hat{\text{TGI}}_i \equiv \left(1 - \frac{\int_{t=a}^{b} V_{\hat{\beta}_i}(t)dt}{\int_{t=a}^{b} V_{\hat{\beta}_1}(t)dt} \right) \times 100\%$$

Note that even when the model holds, typically neither the dividend nor the divisor (nor their ratio) will follow a Gaussian distribution. Consequently, inference uses a parametric bootstrap.

17.3.2 Tumor Growth Inhibition normalized, baseline-normalized (TGIb)

Tumor growth inhibition baseline-normalized (TGIb) extends the TGI measure by inverting the fitted curves to the original scale and also subtracting out the estimated baseline tumor burden from each group prior to numerical integration. Again letting $i = 1$ denote the reference group, the baseline-normalized tumor growth inhibition of group i ($i \in 2, .., I$) relative to reference is

$$\hat{\text{TGIb}}_i \equiv \left(1 - \frac{\int_{t=a}^{b} V_{\hat{\beta}_i}(t)dt - \left[(b-a) \times V_{\hat{\beta}_i}(a)\right]}{\int_{t=a}^{b} V_{\hat{\beta}_1}(t)dt - \left[(b-a) \times V_{\hat{\beta}_1}(a)\right]} \right) \times 100\%$$

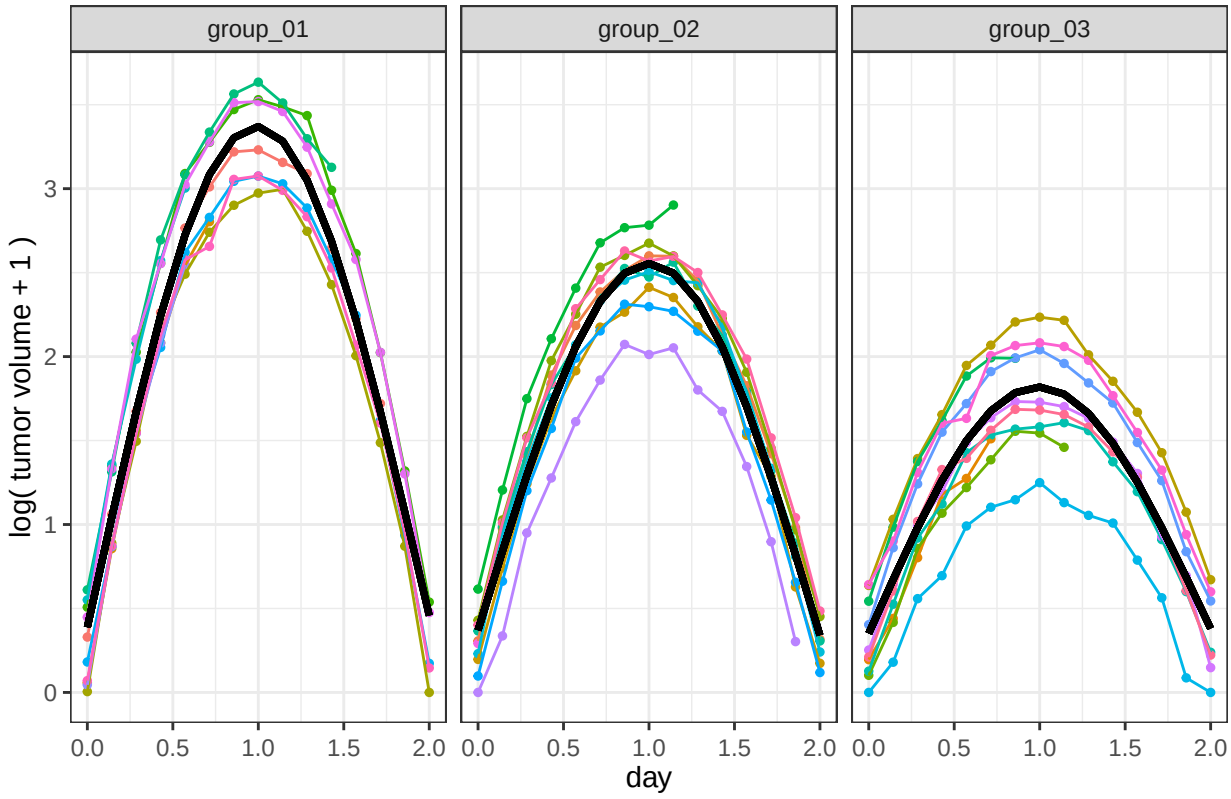
The terms subtracted within the numerator and denominator can be interpreted as areas of rectangles with base lengths equal to the evaluated study period $(b - a)$ and heights the estimated initial tumor burdens $V_{\hat{\beta}_i}(a)$ and $V_{\hat{\beta}_1}(a)$ for the respective groups. Like TGI, the TGIb statistic typically will not fillow a Gaussian sampling distribution but is amenable to parametric bootstrapping. Note also that by the centering and scaling, the statistic is negative for treatments inducing tumor regression, zero for the case of treatment stasis, and one for the case that treatment is exactly like the reference. It is undefined in the case of stasis for the reference group, but this was typically regarded as a failed experiment anyway, and so not of interest to analyze.

These will also be demonstrated with a simulated data set, but not yet.

```
maeve_reset() # set all options to package defaults.

maeve_options( ncol_value = 3, truncate_fit = TRUE )

## An R data file (".rda" format) is stored in /inst/extdata/data/
load( system.file( 'extdata', 'data', 'logQuadraticGrowthSim_01.rda', package = 'maeve' ) )

maeve_options( add_to_endpoint = 0 ) # This has been the case for log-quadratic data.

### Fit models
model_list <- logQuadraticGrowthSim_01 %>% model_study( number_basis_vecs = 4 )
```

```
## boundary (singular) fit: see ?isSingular
```

```
### Set endpoint name for figures:
 maeve_options( endpoint_name = 'y' ) # 'y' is the transformed response & usually what we want for graphics.

### Plot transformed raw data along with predicted responses from models
 model_list %>%
  predict_study %>%
  draw_study( fit = 'spline', endpoint_name = 'y' ) %>%
  print
```

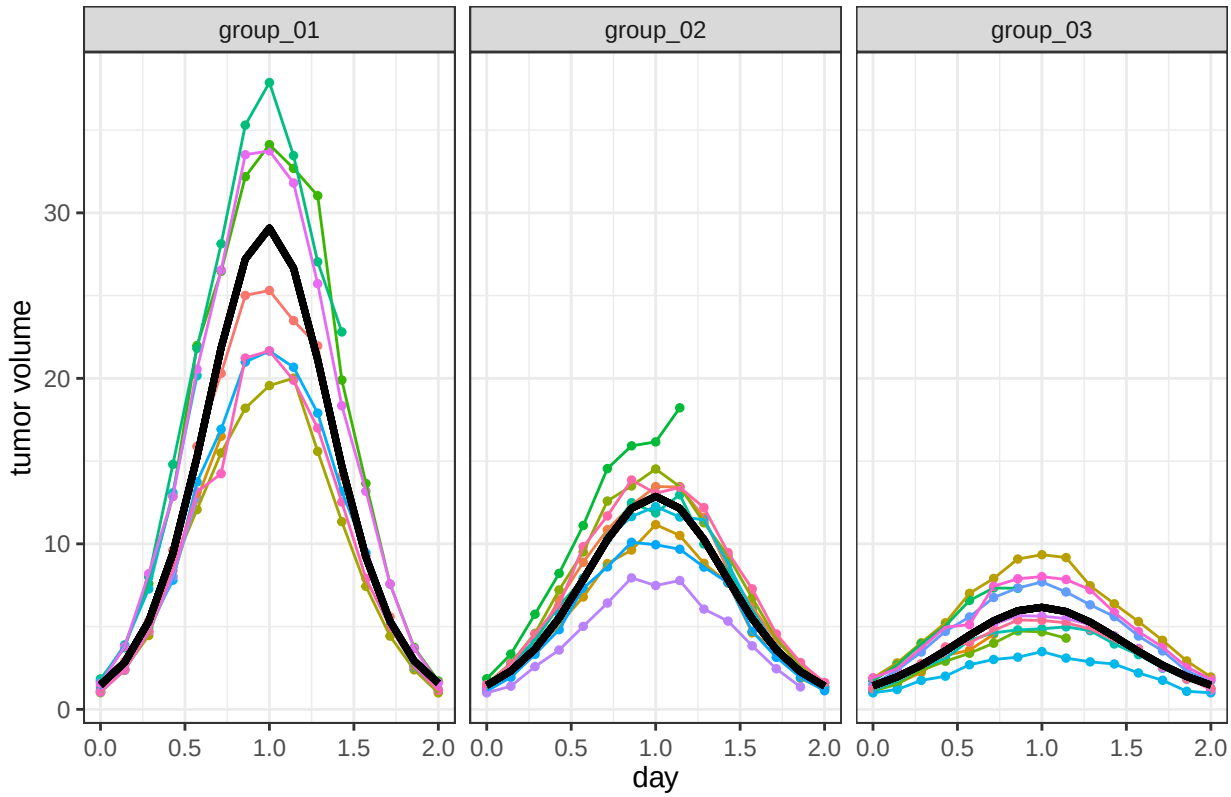



```
### run some compare_groups() analyses just to check against the later results in
### maeve::generate_summary_table():
 cg_Identity_AUC  <-
  compare_groups( model_list, metric = 'AUC',  xmin = 0, xmax = 2, contrast = 'Identity', draw_figure = FALSE )
 print( cg_Identity_AUC$data$effectDF )
```

```
##   metric contrast  Estimate       lwr       upr
## 1    AUC group_01 1.8941538 1.8464886 1.9418190
## 2    AUC group_02 1.3723123 1.3257152 1.4189095
## 3    AUC group_03 0.9312421 0.8859001 0.9765841
```

```
 pred_data_frame = predict_study( model_list, x_pred_type = 'grid', x_pred_spacing = 0.025 )

### Plot original-scale raw data along with back-transformed predicted responses from model
 model_list %>%
  predict_study %>%
  dplyr::mutate( # make a "just-in-time" version of back-transformed spline fit:
            pred_gam_orig = get(maeve_options('inv_func_char'))( pred_gam ) - maeve_options('add_to_endpoint')
              ) %>%
  draw_study(
                          fit = 'spline',
                endpoint_name = 'y_orig',        # raw data in original units.
             spline_predictor = 'pred_gam_orig', # spline back-transformed to original units
```

```
                    y_label = 'tumor volume'    # This plot is on original scale, not transformed.
            ) %>%
 print
```




```
TGI_with_CI <-
 generate_summary_table( model_list,
                         pred_data_frame,
                         xmin = 0, xmax = 2,
                         x_time_spacing = 0.01,
                         metric = 'AUC',
                         ##
                         conf_int = "TGI"
                        )

TGI_with_CI %>%
 dplyr::select( group, firstDay_orig:CI_upper_TGI, TGI_baseline:CI_upper_TGI_baseline ) %>%
 print
```

```
##      group firstDay_orig  AUC_orig      TGI CI_lower_TGI CI_upper_TGI TGI_baseline CI_lower_TGI_baseline
## 1 group_01      1.474535 13.756706  0.00000      0.00000      0.00000      0.00000               0.00000
## 2 group_02      1.448701  6.980962 49.25412     37.23994     59.45747     54.95698              44.20848
## 3 group_03      1.421071  3.955257 71.24852     64.48368     77.09190     79.36695              74.27090
##   CI_upper_TGI_baseline
## 1               0.00000
## 2              64.14806
## 3              83.63099
```

17.4 Appendix: R Session Information

```
sessionInfo() %>% print
```

R version 4.0.0 (2020-04-24) Platform: x86_64-pc-linux-gnu (64-bit) Running under: CentOS Linux 7 (Core)

Matrix products: default BLAS: /gstore/apps/R/R_4.0.0_Bioc_3.11/R-4.0.0-Bioc-3.11-tst-20200504/lib64/R/lib/libRblas.so
LAPACK: /gstore/apps/R/R_4.0.0_Bioc_3.11/R-4.0.0-Bioc-3.11-tst-20200504/lib64/R/lib/libRlapack.so

locale: [1] LC_CTYPE=en_US.UTF-8 LC_NUMERIC=C LC_TIME=en_US.UTF-8
[4] LC_COLLATE=en_US.UTF-8 LC_MONETARY=en_US.UTF-8 LC_MESSAGES=en_US.UTF-8
[7] LC_PAPER=en_US.UTF-8 LC_NAME=C LC_ADDRESS=C
[10] LC_TELEPHONE=C LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages: [1] stats graphics grDevices utils datasets methods base

other attached packages: [1] ggplot2_3.3.2 maeve_0.9.9.45 magrittr_1.5 bookdown_0.21 rmarkdown_2.4

loaded via a namespace (and not attached): [1] Rcpp_1.0.5 mvtnorm_1.1-1 lattice_0.20-41 tidyr_1.1.2 class_7.3-17
[6] zoo_1.8-8 assertthat_0.2.1 digest_0.6.26 utf8_1.1.4 R6_2.4.1
[11] plyr_1.8.6 evaluate_0.14 rootSolve_1.8.2.1 e1071_1.7-4 pillar_1.4.6
[16] rlang_0.4.8 settings_0.2.4 Exact_2.1 multcomp_1.4-14 rstudioapi_0.11
[21] minqa_1.2.4 nloptr_1.2.2.2 Matrix_1.2-18 labeling_0.3 splines_4.0.0
[26] lme4_1.1-23 statmod_1.4.34 stringr_1.4.0 munsell_0.5.0 gamm4_0.2-6
[31] tinytex_0.26 compiler_4.0.0 xfun_0.18 pkgconfig_2.0.3 mgcv_1.8-33
[36] DescTools_0.99.38 htmltools_0.5.0 tidyselect_1.1.0 tibble_3.0.4 lmom_2.8
[41] expm_0.999-5 codetools_0.2-16 fansi_0.4.1 crayon_1.3.4 dplyr_1.0.2
[46] withr_2.3.0 MASS_7.3-53 grid_4.0.0 nlme_3.1-149 gtable_0.3.0
[51] lifecycle_0.2.0 scales_1.1.1 gld_2.6.2 cli_2.1.0 stringi_1.5.3
[56] farver_2.0.3 ellipsis_0.3.1 generics_0.0.2 vctrs_0.3.4 boot_1.3-25
[61] sandwich_3.0-0 TH.data_1.0-10 tools_4.0.0 forcats_0.5.0 glue_1.4.2
[66] purrr_0.3.4 survival_3.2-7 yaml_2.2.1 colorspace_1.4-1 knitr_1.30