Quick start for `maeve`

Bill Forrest (Genentech OMNI Bioinformatics)

2020-06-28

Abstract

The `R` package `maeve` helps to visualize and model longitudinal data from multi-group studies with multiple subjects per group into one statistical summary per group, and then to make comparisons of summary statistics across groups. The motivating application is to tumor volume growth data from mouse studies. In this short-form vignette we show a few working commands on one real data set. It is not intended to be a comprehensive overview. More extensive descriptions and examples are in the `maeve_LongVignette.pdf` document within the package.

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. In this short vignette we show a few working commands on one real data set. This is not intended to be a comprehensive set of examples. More extensive descriptions and examples are in the `maeve_LongVignette.pdf` document within the package.

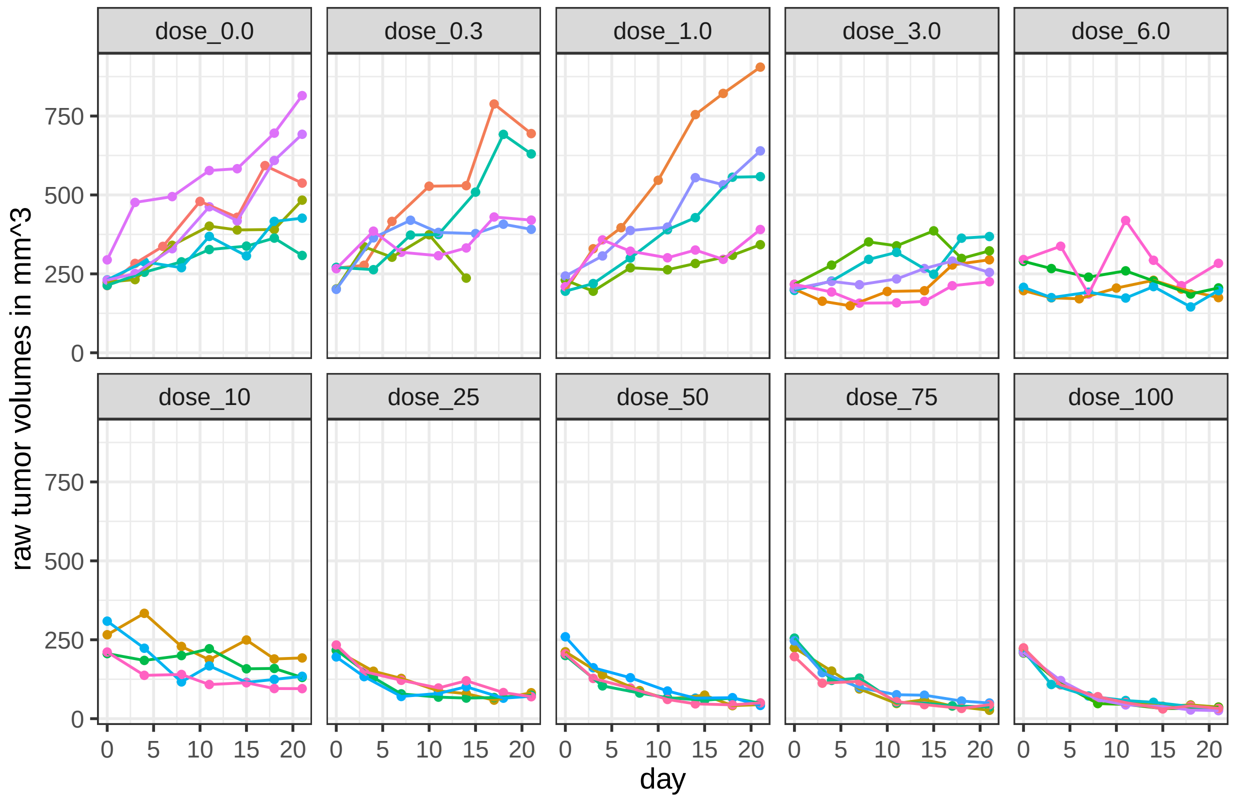
2 Set up a data set and draw some pictures.

We demonstrate some core `maeve` functions in a brief case study of trichoblastoma allograft growth curves in mice randomized into ten groups and treated with escalating doses of vismodegib in a data set that comes with the package. To read in a *new* data set, it may help to read the long-form vignette, especially the sections around `check_study_data_frame()` and `maeve_options()`.

Package parameters set with `maeve_options()` provide a shared list of arguments accessed as defaults by several functions in `maeve`. An option value will persist until explicitly changed with `maeve_options()` or until all the options are reset to their respective default values by `maeve_reset()`. The defaults of options are always available in `maeve_defaults()`. These functions rely on the `R` package `settings`[1]. If the user forgets earlier options that have been set, unexpected behavior can result. Starting a new series of analyses with a `maeve_reset()` call will avoid this issue.

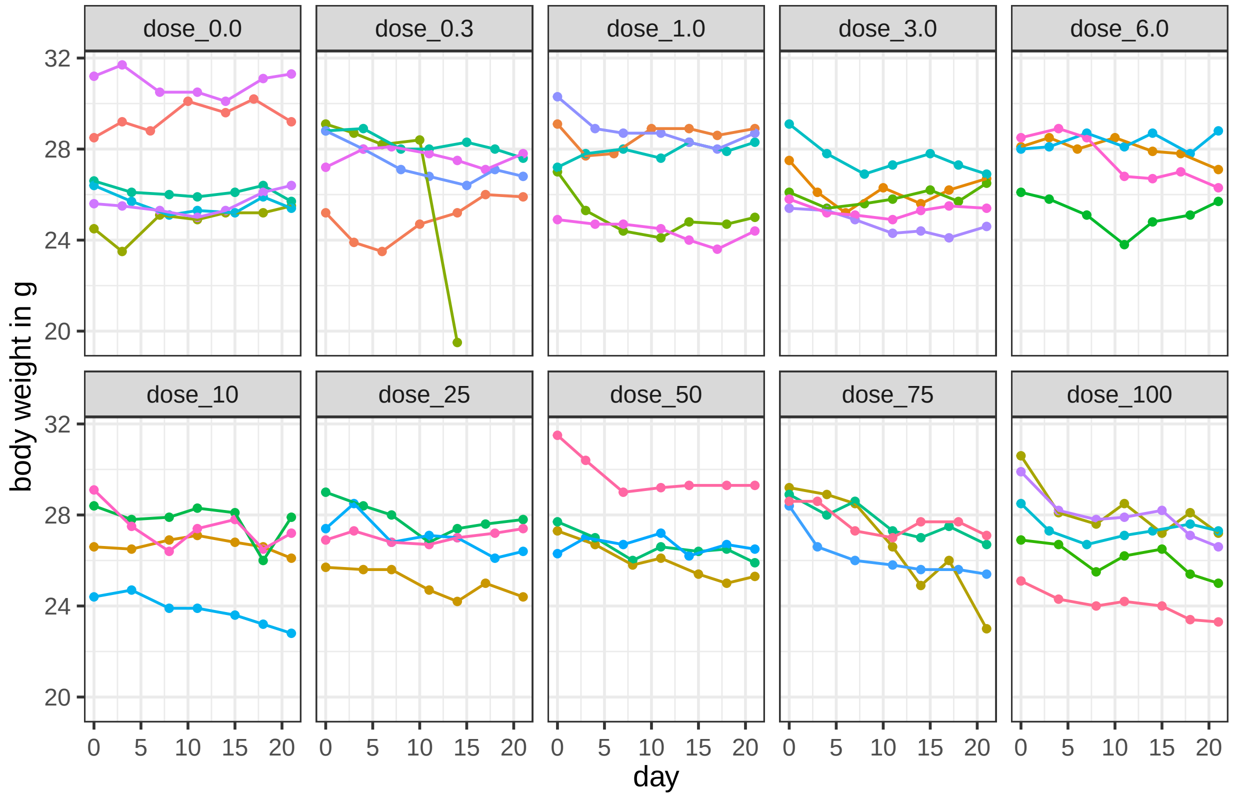
In the following code chunk, we set a few package options, plot raw tumor volumes with one dose group per panel and observations from different mice in distinct colors, then show how to switch to another longitudinal endpoint (body weight, here) in the same sort of plot.

```
maeve_reset() # make sure package options start at their defaults.
maeve_options( ncol_value = 5 ) # with 10 groups, this facets to 2 rows & 5 columns.

## The high-dose group was observed post-treatment for several weeks.
## For this example, restrict to the three weeks of treatment common
## to all ten dose groups.
##
vismo21 <- dplyr::filter( vismodegib, DAY_OF_STUDY <= 21 ) # restrict to days [0, 21].

##
figure_with_raw_TUMOR_VOLUME <- maeve::draw_study( vismo21, y_label = 'raw tumor volumes in mm^3' )
print( figure_with_raw_TUMOR_VOLUME )
```



```
##
## The 'endpoint_name' default value of 'TUMOR_VOLUME' can be overridden in-place by giving a
## different endpoint_name to draw_study():
figure_with_raw_BODY_WEIGHT <-
    maeve::draw_study( vismo21, endpoint_name = 'BODY_WEIGHT', y_label = 'body weight in g' )
```

```
print( figure_with_raw_BODY_WEIGHT )
```



The `tally_study()` function creates a basic tabular summary of the study with the number per group, first and last day of observations for each group, and whether a group's observations *all* started or ended on those days. For the `vismodegib` study, a tally might look like this:

```
tally_study( vismodegib )
```

```
##    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
```

```
## Tally the study, but include the count of partial responses (PR),
## defined as a mouse with a tumor that falls to 75% of its baseline
## at least once in the study.  Include log odds ratio estimate and
## standard error relative to the reference group (which is the first
## group by default).
tally_study( vismodegib, response = 'PR', PR_threshold = .75 ) %>% format( digits = 3 )
```

```
##    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     0.000         0.00
## 2    dose_0.3          5         0         TRUE       21      FALSE  0     0.167         2.08
## 3    dose_1.0          5         0         TRUE       21       TRUE  0     0.167         2.08
## 4    dose_3.0        5       0       TRUE     21     TRUE  2     2.228       1.69
## 5    dose_6.0        4       0       TRUE     21     TRUE  3     3.412       1.76
## 6     dose_10        4       0       TRUE     21     TRUE  4     4.762       2.09
## 7     dose_25        4       0       TRUE     21     TRUE  4     4.762       2.09
## 8     dose_50        4       0       TRUE     21     TRUE  4     4.762       2.09
## 9     dose_75        4       0       TRUE     21     TRUE  4     4.762       2.09
## 10   dose_100        5       0       TRUE     74    FALSE  5     4.963       2.08
```

`PR_threshold` is an example of a parameter that can be set in the options manager, e.g. `maeve_options( PR_threshold = 2/3 )`, so that its value persists throughout an analysis session. The `tally_study()` function has further options (e.g., end-of-study complete responses) explored in the long-form vignette.

3 Statistical model fitting

Four kinds of statistical mixed effects model are supported in `maeve`. Which ones are fit depends on the requested group summary metric(s) specified in the `maeve_options("metric")` parameter. All four rely at least in part on the `lme4` package[2]. The most commonly used model is based on thin-plate regression splines[3],[4] and relies on the `R` packages `gamm4`[5] and `mgcv`[6], and this is what we do here. See the long-form vignette for further details.

We next fit a generalized additive mixed model (GAMM) with fixed-effects regression splines by group and random intercept and slope terms by animal. This is the workhorse model used within `maeve`. Setting `maeve_options( metric = 'AUC')` indicates that we want group summary metrics to be some flavor of *area under the curve* (AUC) statistic, so `maeve::model_study()` below will fit a GAMM, then the spline for each group will be summarized over the study interval as an AUC statistic via numerical integration in `maeve::compare_groups()`, demonstrated below. By default, the height of each group's curve at baseline will be subtracted out and these baseline-corrected AUCs then scaled by half the square of the study interval. With these defaults, the resulting AUC statistic is the *endpoint Gain Integrated in Time* ("eGaIT"), and will behave like the slope when log tumor volume growth is linear. In summary, eGaIT is a special (albeit default) case of AUC summary statistic for a spline.

If the `"metric"` option is changed to something else (e.g., `"linear"`, `"ITGR"`), a different model and / or summary is provided. E.g., an `lmerMod` object from `lme4` is fit if `metric = "linear"`, while a GAMM is fit with `metric = "ITGR"`, but the spline is then summarized by `compare_groups()` (again, by default) into the *endpoint Difference Over Time* (eDOT) statistic, essentially the "rise over the run" of the spline curve over the study interval. Further details are in the long-form vignette. Fitted models are stored in a list for further analysis steps.

Decisions about the independent and dependent variables in the model, treatment groups, and subject-level identifiers are all determined either inline to the functions or (more often) set in `maeve_options()`, which both allows the code statements to be shorter. See, e.g., the two `model_XXXX` lists fit in the next code chunk, which provide identical fits.

```
maeve_reset() ## Reset package options to their defaults.
maeve_options( metric = 'AUC', # area-under-curve group-summary metric, 'eGaIT' by default.
               ncol_value = 5  # with 10 groups, this facets to 2 rows & 5 columns.
              )

model_list    <- maeve::model_study( vismo21 )

## This next code fits the *same* model as the one line above.
## The options passed are normally accessed by 'model_study()'
## from 'maeve_options()'.  They are also needed by lots of
## subsequent functions, so managing them by keeping a common
## list in 'maeve_options()' simplifies the commands.
model_verbose <- maeve::model_study( vismo21,
                                     ##
                                     group_name    = 'group_name',
                                     subject_ID    = 'animalID',
                                     x_name        = 'DAY_OF_STUDY',
                                     endpoint_name = 'TUMOR_VOLUME'
                                    )

## check that the predicted values match:
identical( predict(    model_list[['md4_gamm4']]$gam ),
           predict( model_verbose[['md4_gamm4']]$gam )
          )
```

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

4 Predicting values from a model

Starting from the model object in `model_list`, the next step will create a data frame with group-level fitted values, among many other columns. In the predicted data frame is a column labeled simply `y` with the endpoint after a variance-stabilizing transform (VST). By default in `maeve`, this VST is the natural logarithm of “1” plus the endpoint from the `maeve_options("endpoint_value")` field. This transformed `y` is then the variable on which modeling and fits are done.

```
predicted_data_frame <- maeve::predict_study( model_list )

## Among several other columns are the original four used in modeling and
## the 'y' column, where here y = log( 1 + TUMOR_VOLUME ):
##
predicted_data_frame %>%
    dplyr::select( group_name, animalID, DAY_OF_STUDY, TUMOR_VOLUME, y ) %>%
    format( digits = 3 ) %>%
    head( 10 )
```

```
##    group_name animalID DAY_OF_STUDY TUMOR_VOLUME    y
## 1    dose_0.0  3810580            0        212.8 5.36
## 2    dose_0.0  3810580            3        283.1 5.65
## 3    dose_0.0  3810580            6        337.0 5.82
## 4    dose_0.0  3810580           10        479.5 6.17
## 5    dose_0.0  3810580           14        429.3 6.06
## 6    dose_0.0  3810580           17        593.0 6.39
## 7    dose_0.0  3810580           21        537.6 6.29
## 8    dose_0.0  3810587            0        223.8 5.42
## 9    dose_0.0  3810587            3        231.8 5.45
## 10   dose_0.0  3810587            7        340.4 5.83
```

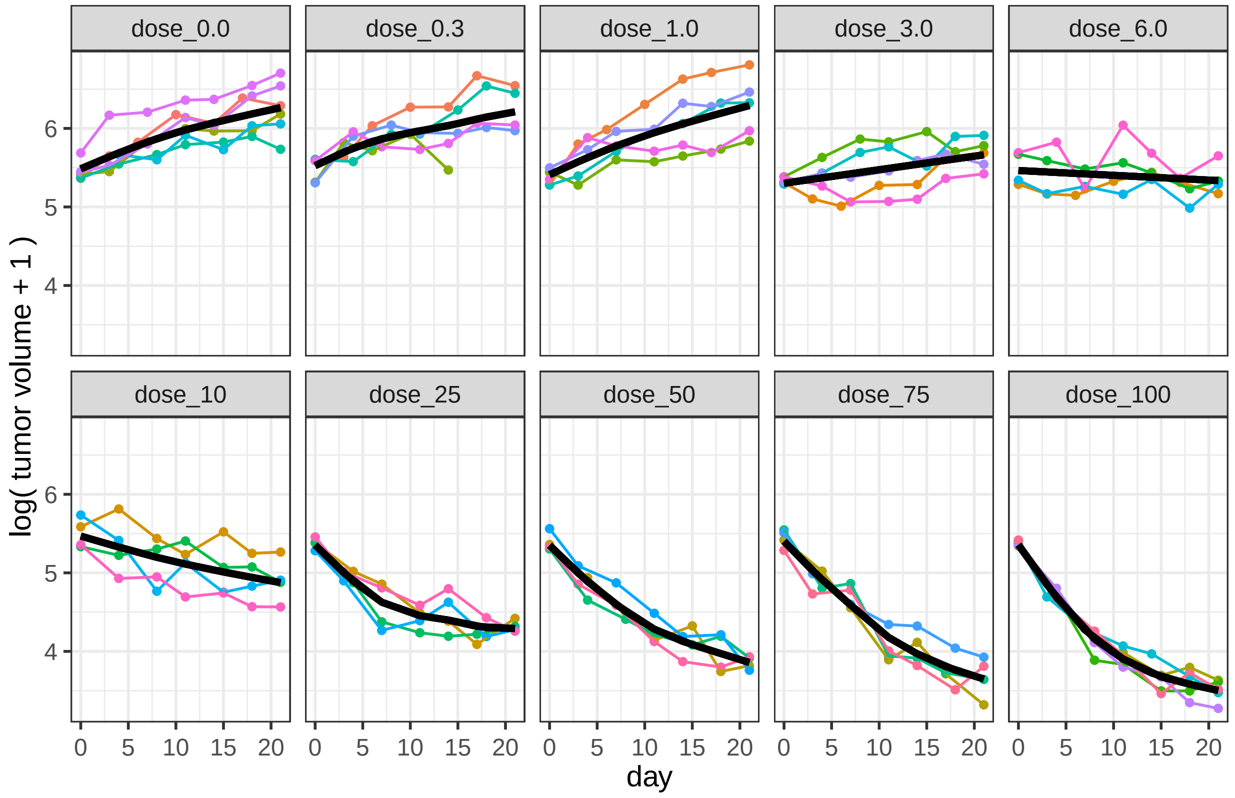
The `TUMOR_VOLUME` values shown are the original ones from the data, though they can also be recovered by inverting the VST as $\texttt{TUMOR\_VOLUME} = e^y - 1$. The `predict_study()` function has functionality to interpolate the fits both on the transformed and original scale at user-defined grids. These are described in the long-form vignette and documentation.

5 Drawing figures with predicted curves included

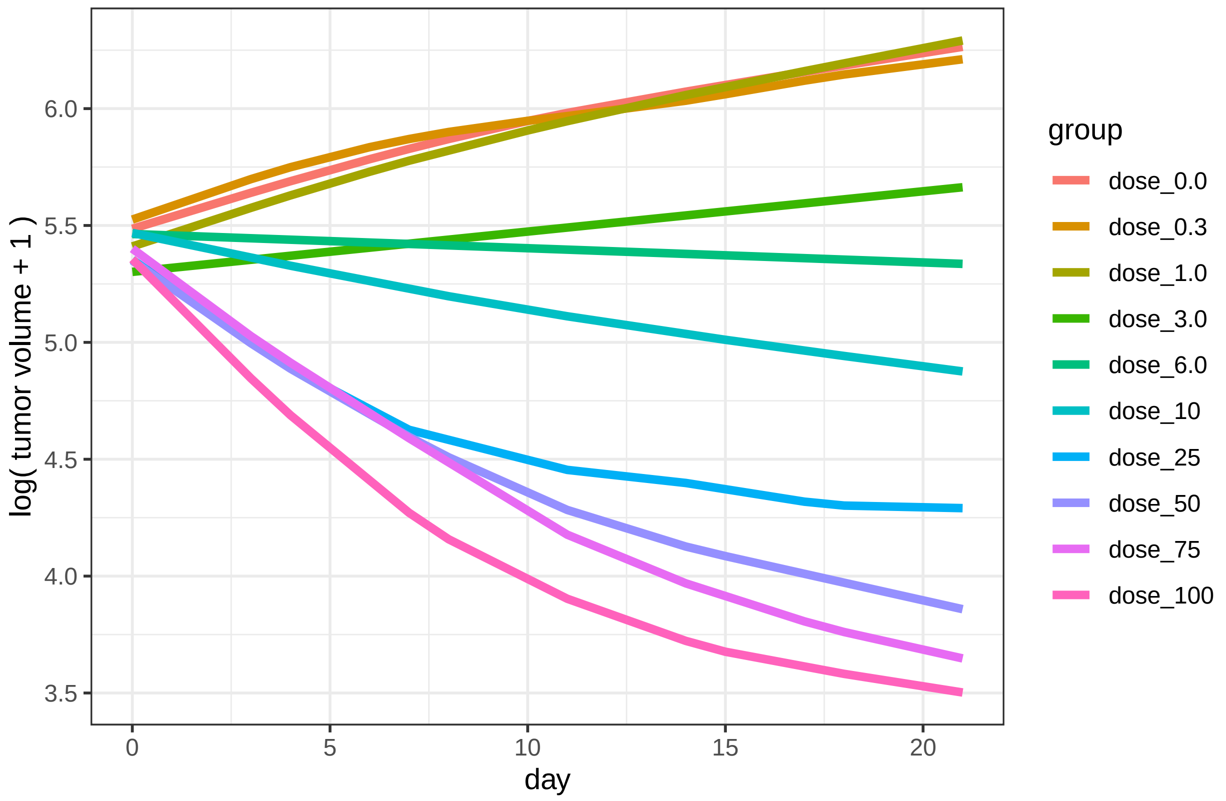
Make another `ggplot` figure, but now with the fitted curves included for the log tumor volume:

```
## revise the raw data plot to draw the transformed variable with fitted values.
##
figure_with_fits_by_dose <- maeve::draw_study( predicted_data_frame, endpoint_name = 'y', fit = 'spline' )
```

```
print( figure_with_fits_by_dose )
```



```
## Add an overlay plot with all the group-level fits in one plot:
figure_overlay <- draw_overlay( predicted_data_frame, legend_position_char = 'right' )

print( figure_overlay )
```



6 Statistical summaries of group-level effects and comparisons across groups

Finally, we want some statistical analysis summarizing how the groups compare. The `compare_groups()` function takes a list of fitted models from `model_study()` and summarizes each metric (e.g., `linear`, `AUC`, `AUC_pwl`, etc.) into one number per requested comparison. Which comparisons are produced depends on the `contrast` parameter held in `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, then returned *without* any inter-group comparison. Re-setting `maeve_options("contrast")` allows other options, e.g., inter-group comparisons, with an example below. First, here are the by-group summaries:

```r
## Generate a list with multcomp::glht() models, summary data frames, and simple figures:
cg_Identity <- compare_groups( model_list, extended_output = TRUE ) # include, SEs, p-values, etc.

## The output is arranged into three sub-lists containing, respectively:
## (1) multcomp::glht() models,
## (2) summary data, and
## (3) basic figures of some summary results
names( cg_Identity )
```

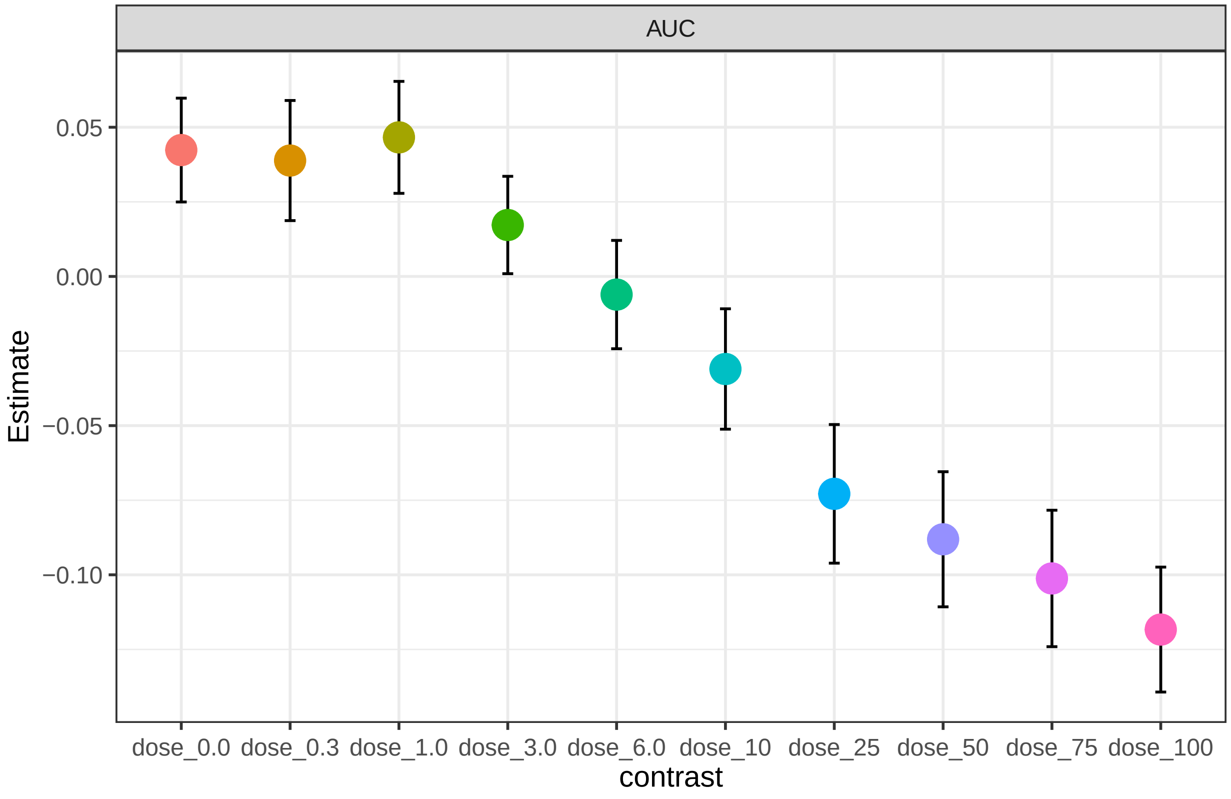
[1] “models” “data” “figures”

```
## Here is a summary table of effect estimates for each group.  The number
## shown is the eGaIT summary statistic for each treatment, which will be
## similar to the linear mixed effects slope of log tumor volume on day in
## this setting, since the log tumor volume profiles are fairly linear.
##
```

```r
cg_Identity$data$effectDF %>%
    base::format( digits = 3 ) %>%
    ## NB: If running this as an R script, just comment out or remove
    ## the function "print_table_by_output_format()".  It is included
    ## to facilitate table printing in different document formats.
    print_table_by_output_format() # defined in header, based on compilation format ( PDF, HTML, DOCX ).
```

metric	contrast	Estimate	sigma	lwr	upr	tstat	adj_method	pvalues
AUC	dose_0.0	0.04235	0.00621	0.024958	0.0597	6.817	single-step	9.30e-11
AUC	dose_0.3	0.03884	0.00719	0.018709	0.0590	5.401	single-step	6.61e-07
AUC	dose_1.0	0.04661	0.00670	0.027850	0.0654	6.956	single-step	3.50e-11
AUC	dose_3.0	0.01724	0.00583	0.000918	0.0336	2.957	single-step	3.06e-02
AUC	dose_6.0	-0.00608	0.00648	-0.024239	0.0121	-0.938	single-step	9.86e-01
AUC	dose_10	-0.03102	0.00720	-0.051192	-0.0109	-4.306	single-step	1.66e-04
AUC	dose_25	-0.07286	0.00830	-0.096089	-0.0496	-8.782	single-step	0.00e+00
AUC	dose_50	-0.08809	0.00809	-0.110728	-0.0655	-10.895	single-step	0.00e+00
AUC	dose_75	-0.10121	0.00817	-0.124081	-0.0783	-12.391	single-step	0.00e+00
AUC	dose_100	-0.11835	0.00748	-0.139282	-0.0974	-15.828	single-step	0.00e+00

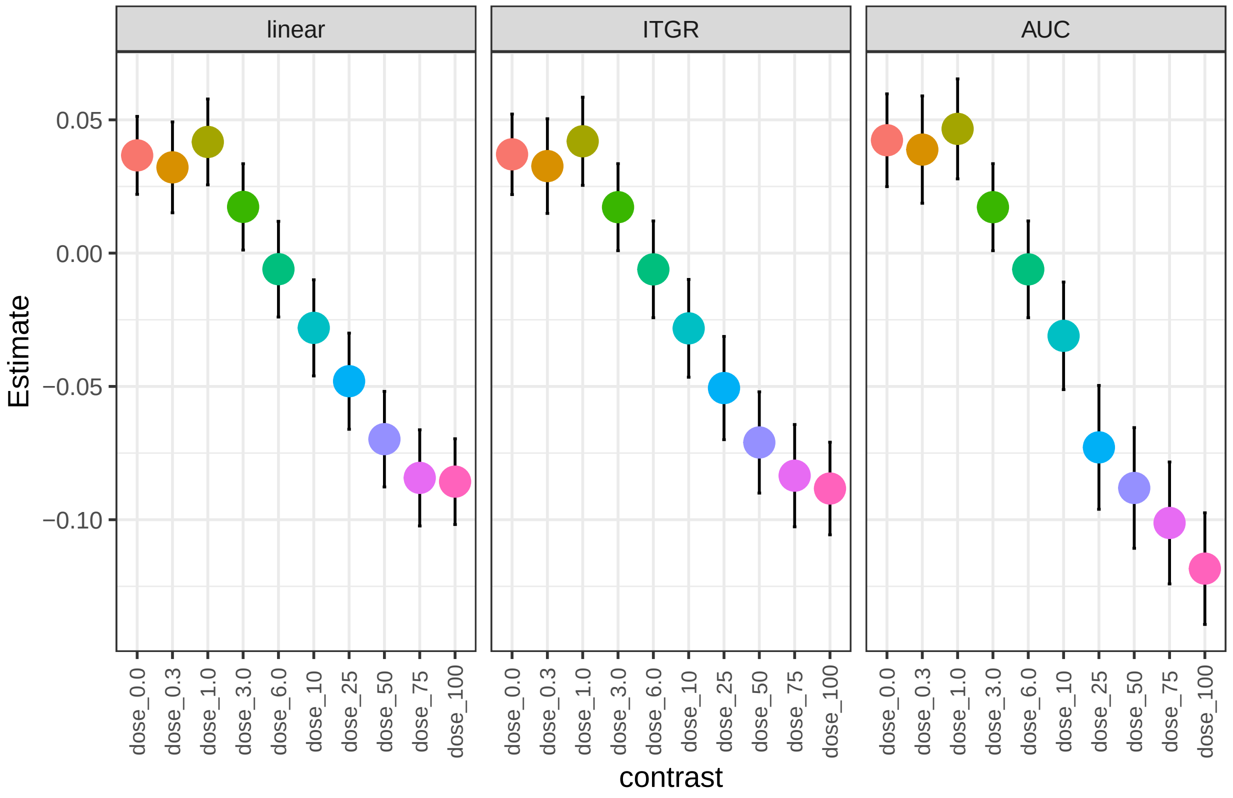
```r
## A vanilla one-way layout figure of the contrasted effects with confidence intervals is auto-generated.
##
print( cg_Identity$figures$figCI )
```



In the above summary table, each `contrast` denotes one of the ten dose groups. The `Estimate` column gives the summary statistic eGaIT obtained by setting `metric = "AUC"`. Setting, e.g., `metric = "linear"` and regenerating the `model_list` from `model_study` would give the LMM slope estimates for each group, while setting `metric = c("linear","AUC")` would fit both the LMM and GAMM and cause `compare_groups()` to compute both the LMM slope and eGaIT, doubling the output shown in the summary above. In each line, the t-statistic and p-value are for the null hypothesis that the underlying value for the contrast is zero.

If *two* or more metrics are specified in the original model, then `compare_groups()` will include them all in its output. We briefly repeat the example above with three metrics: `linear` for the LME slope, `ITGR` for the eDOT score (assuming other parameters at their defaults), and `AUC` for the eGaIT score (again, for other parameters at defaults).

```
maeve_reset() ## Reset package options to their defaults.
maeve_options( metric = c( 'linear', 'ITGR', 'AUC' ) )
model_list_3metrics <- maeve::model_study( vismo21 )

## Generate a list with multcomp::glht() models, summary data frames, and simple figures:
cg_Identity_3metrics <- compare_groups( model_list_3metrics )

## We repeat just the one-way layout summary figure:
print(
    cg_Identity_3metrics$figures$figCI +
    ## set 'axis.text.x' to avoid overwriting the now more crowded contrast labels:
    theme( axis.text.x = element_text( angle = 90, size = 8 ) )
)
```



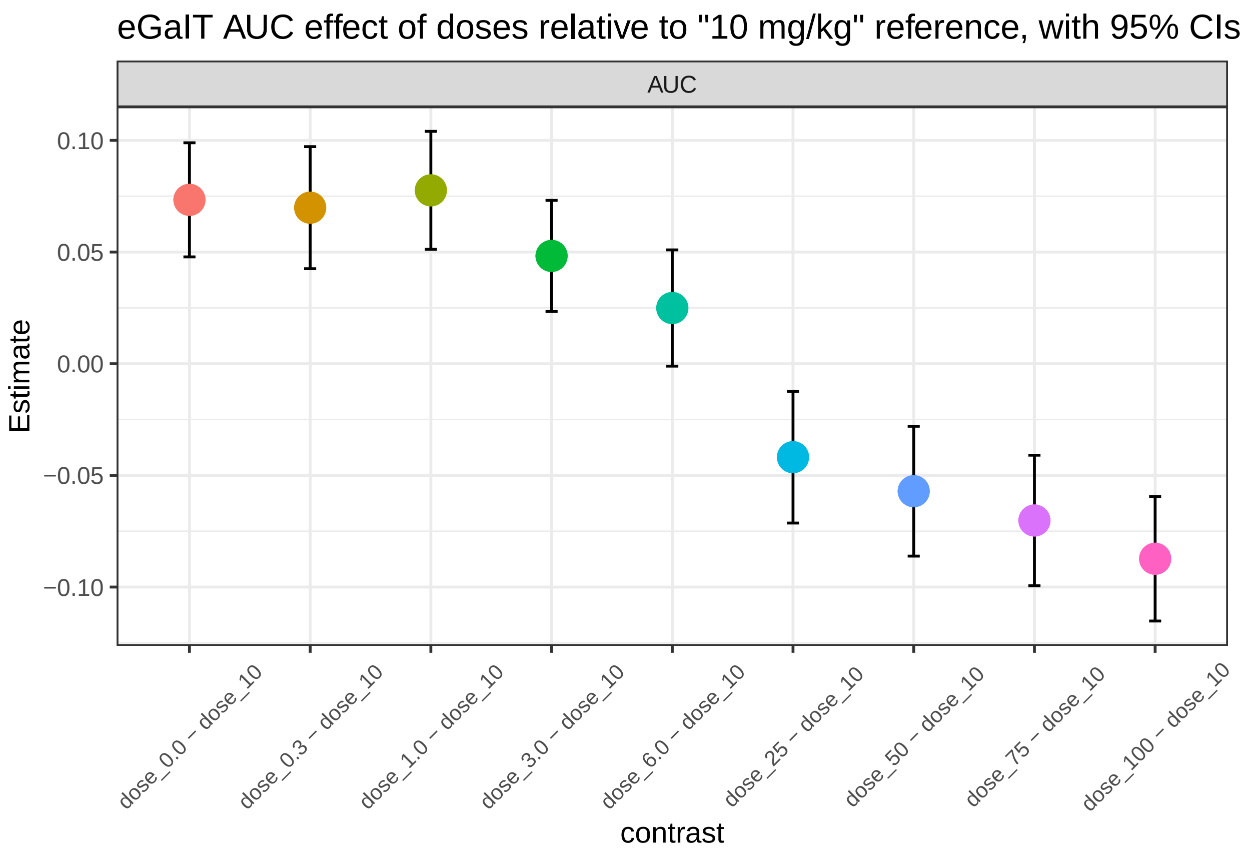
Frequently, a study has a designated "reference" group, and researchers want to evaluate each active treatment relative to that reference. This can be accomplished by setting the contrast option to `"Dunnett"`. By default, the reference group used will be the first level of the `group_name` factor (e.g., `"dose_0.0"` in the vismodegib data set we are analyzing in this vignette). However, a different common reference group can be set via `"reference_Dunnett"` within `maeve_options()`. An example is shown in the next chunk. We also demonstrate one of several alternative methods for multiple comparison adjustment (Holm's method, here, but other adjustment methods from `stats::p.adjust()` can be substituted).

```
## Request Dunnett test comparisons, using the 10 mg/kg dose group as the common reference.
## By default, "reference_Dunnett" will be the first level of the 'group_name' factor, i.e.,
## 'dose_0.0' in this case.
maeve_reset()
maeve_options( contrast = 'Dunnett', reference_Dunnett = 'dose_10', metric = 'AUC' )
```

```r
cg_Dunnett <- compare_groups( model_list, adjustment_method = 'holm', extended_output = TRUE )

## Here is a summary table of each active treatment's effect estimate minus the control group estimate.
## The 'Estimate' column is the difference in eGaIT summary statistics between active treatment
## and control.
##
cg_Dunnett$data$effectDF %>%
    base::format( digits = 3 ) %>%
    ## NB: If running this as an R script, just comment out or remove
    ## the function "print_table_by_output_format()".  It is included
    ## to facilitate table printing in different document formats.
    print_table_by_output_format() # defined in header, based on compilation format ( PDF, HTML, DOCX ).
```

metric	contrast	Estimate	sigma	lwr	upr	tstat	adj_method	pvalues
AUC	dose_0.0 - dose_10	0.0734	0.00951	0.04783	0.0989	7.71	holm	0.00e+00
AUC	dose_0.3 - dose_10	0.0699	0.01018	0.04254	0.0972	6.86	holm	4.00e-11
AUC	dose_1.0 - dose_10	0.0776	0.00984	0.05122	0.1040	7.89	holm	0.00e+00
AUC	dose_3.0 - dose_10	0.0483	0.00927	0.02338	0.0731	5.21	holm	5.74e-07
AUC	dose_6.0 - dose_10	0.0249	0.00969	-0.00108	0.0510	2.57	holm	1.01e-02
AUC	dose_25 - dose_10	-0.0418	0.01099	-0.07134	-0.0123	-3.81	holm	2.81e-04
AUC	dose_50 - dose_10	-0.0571	0.01083	-0.08614	-0.0280	-5.27	holm	5.46e-07
AUC	dose_75 - dose_10	-0.0702	0.01089	-0.09943	-0.0410	-6.44	holm	5.80e-10
AUC	dose_100 - dose_10	-0.0873	0.01038	-0.11520	-0.0595	-8.41	holm	0.00e+00

```r
## A one-way layout figure of the contrasted effects with confidence intervals:
##
print(
    cg_Dunnett$figures$figCI +
    ## set 'axis.text.x' to avoid overwriting the now-longer contrast labels:
    theme( axis.text.x = element_text( angle = 45, size = 8 ) ) +
    labs( title = 'eGaIT AUC effect of doses relative to "10 mg/kg" reference, with 95% CIs' )
)
```



Several further options for comparing group summary metrics are available, including customized linear contrasts. The `compare_groups()` function relies on functionality of the `R` package `multcomp`[7]. See the long-form vignette for details.

7 Discussion

This has been an intentionally brief look at some of the capabilities of `maeve` for visualizing, tabulating, modeling, and summarizing by-group longitudinal data. More details are available in the long-form vignette, and a more detailed description of the eGaIT and eDOT parameters (obtained by setting `metric = "AUC"` and `metric = "ITGR"`, respectively) is in the companion manuscript.

References

1. van der Loo, M. *Settings: Software option settings manager for r.* (2015).

2. Bates, D., Mächler, M., Bolker, B. & Walker, S. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software* **67,** 1–48 (2015).

3. Wood, S. N. Thin plate regression splines. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **65,** 95–114 (2003).

4. Simpson, G. L. Modelling palaeoecological time series using generalised additive models. *Frontiers in Ecology and Evolution* **6,** 149 (2018).

5. Wood, S. & Scheipl, F. *Gamm4: Generalized additive mixed models using 'mgcv' and 'lme4'.* (2016).

6. Wood, S. N. Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **73,** 3–36 (2011).

7. Hothorn, T., Bretz, F. & Westfall, P. Simultaneous inference in general parametric models. *Biometrical Journal* **50,** 346–363 (2008).