Example run maicplus

Introduction

This package describes the steps required to perform a matching-adjusted indirect comparison (MAIC) analysis using the `maicplus` package in R where the endpoint of interest is either time-to-event (e.g. overall survival) or binary (e.g. objective tumor response).

The methods described in this document are based on those originally described by Signorovitch et al. 2010 and described in the National Institute for Health and Care Excellence (NICE) Decision Support Unit (DSU) Technical Support Document (TSD) 18. [signorovitch2010; phillippo2016a]

MAIC methods are often required when:

- There is no common comparator treatment to link a clinical trial of a new intervention to clinical trials of other treatments in a given disease area. For example if the only study of a new intervention is a single arm trial with no control group. This is commonly referred to as an unanchored MAIC.
- A common comparator is available to link a clinical trial of a new intervention to a clinical trial of one other treatment in a given disease area but there are substantial differences in patient demographic or disease characteristics that are believed to be treatment effect modifiers. This is commonly referred to as an anchored MAIC.

The premise of MAIC methods is to adjust for between-trial differences in patient demographic or disease characteristics at baseline. When a common treatment comparator or 'linked network' are unavailable, a MAIC assumes that differences between absolute outcomes that would be observed in each trial are entirely explained by imbalances in prognostic variables and treatment effect modifiers. Prognostic variables are those that are predictive of disease outcomes, independent of the treatment received. For example, older patients may have increased risk of death compared to younger patients. Treatment effect modifiers are those variables that influence the relative effect of one treatment compared to another. For example patients with a better performance status may experience a larger treatment benefit than those with a worse performance status. Under this assumption, every prognostic variable and every treatment effect modifier that is imbalanced between the two studies must be available. This assumption is generally considered very difficult to meet. [phillippo2016a] There are several ways of identifying prognostic variables/treatment effect modifiers to be used in the MAIC analyses, some of which include:

- Clinical expertise (when available to a project)
- Published papers/previous submissions (what has been identified in the disease area previously)
- Univariable/multivariable regression analyses to identify which covariates have a significant effect on the outcome
- Subgroup analyses of clinical trials may identify interactions between patient characteristics and the relative treatment effect

Theory behind MAIC

We will briefly go over the theory behind MAIC. For detailed information, see Signorovitch et al. 2010.

Let us define t_i to be the treatment patient i received. We assume $t_i = 0$ if the patient received intervention (IPD) and $t_i = 1$ if the patient received comparator treatment. The causal effect of treatment $T = 0$ vs $T = 1$ on the mean of the outcome Y can be estimated as below

$$\frac{\sum_{i=1}^{n} y_i(1 - t_i)w_i}{\sum_{i=1}^{n}(1 - t_i)w_i} - \bar{y}_1$$

where $w_i = \frac{Pr(T_i=1|x_i)}{Pr(T_i=0|x_i)}$ is the odds that patient i received treatment $T = 1$ vs $T = 0$ (i.e. enrolls in aggregate data study vs IPD study) given baseline characteristics x_i. Thus, the patients receiving $T = 0$ are re-weighted to match the distribution of patients receiving $T = 1$. Note that this causal effect would be the case when the outcome Y is continuous. If the outcome is binary, Y would be a proportion and we would use a link function such as logit to give us the causal effect in an odds ratio scale. As in propensity score methods, we may assume w_i to follow logistic regression form

$$w_i = exp(x_i^T \beta)$$

However, in order to estimate β, we cannot use maximum likelihood approach because we do not have IPD for both trials. Instead, we use method of moments. We estimate β such that the weighted averages of the covariates in the IPD exactly matches the aggregate data averages. Mathematically speaking, we want to estimate β such that:

$$0 = \frac{\sum_{i=1}^{n} x_i exp(x_i^T \hat{\beta})}{\sum_{i=1}^{n} exp(x_i^T \hat{\beta})} - \bar{x}_{agg}$$

If the x_i contains all confounders and the logistic regression for w_i is correctly specified, we obtain a consistent estimate of the causal effect of intervention vs comparator treatment. Above equation is equivalent to

$$0 = \sum_{i=1}^{n}(x_i - \bar{x}_{agg})exp(x_i^T \hat{\beta})$$

We could transform transform IPD by subtracting the aggregate data means (this is why centering is needed when preprocessing).

$$0 = \sum_{i=1}^{n} x_i exp(x_i^T \hat{\beta})$$

Note that this is the first derivative of

$$Q(\beta) = \sum_{i=1}^{n} exp(x_i^T \hat{\beta})$$

which has second derivative

$$Q''(\beta) = \sum_{i=1}^{n} x_i x_i^T exp(x_i^T \hat{\beta})$$

Since $Q''(\beta)$ is positive-definite for all β, $Q(\beta)$ is convex and any finite solution from the equation is unique and corresponds to the global minimum of $Q(\beta)$. Thus, we can use optimization methods to calculate β.

Example scenario

We present an unanchored MAIC of two treatments in lung cancer. The two endpoints being compared are overall survival (a time to event outcome) and objective response (a binary outcome). The data available are:

- Individual patient data from a single arm study
- Aggregate summary data for the comparator study
- Psuedo patient data from the comparator study. This is not required for the matching process but is needed to derive the relative treatment effects between the intervention and comparator.

Preprocessing

Package load

```
setwd("~/GitHub/maicplus")
devtools::load_all()

# devtools::install_github('hta-pharma/maicplus') library(maicplus)

library(dplyr)  # this is used for data merging/cleaning. Package itself does not depend on dplyr

library(clubSandwich)  # For robust standard error in logistic regression
library(sandwich)

library(survminer)  # for ggsurvplot
library(ggplot2)  # for ggplot functions
library(boot)  # for bootstrapping
```

Preprocessing IPD

In this example scenario, age, sex, the Eastern Cooperative Oncology Group (ECOG) performance status, smoking status, and number of previous treatments have been identified as imbalanced prognostic variables/treatment effect modifiers.

This example reads in and combines data from three standard simulated data sets (adsl, adrs and adtte) which are saved as '.csv' files.

```
adsl <- read.csv(system.file("extdata", "adsl.csv", package = "maicplus", mustWork = TRUE))
adrs <- read.csv(system.file("extdata", "adrs.csv", package = "maicplus", mustWork = TRUE))
adtte <- read.csv(system.file("extdata", "adtte.csv", package = "maicplus", mustWork = TRUE))

# Data containing the matching variables
adsl <- adsl %>%
    mutate(SEX_MALE = ifelse(SEX == "Male", 1, 0)) %>%
    mutate(AGE_SQUARED = AGE^2)

# Could use built-in function for dummizing variables adsl <- dummize_ipd(adsl,
# dummize_cols=c('SEX'), dummize_ref_level=c('Female'))
```

```r
# Response data
adrs <- adrs %>%
    filter(PARAM == "Response") %>%
    transmute(USUBJID, ARM, RESPONSE = AVAL)

# Time to event data (overall survival)
adtte <- adtte %>%
    filter(PARAMCD == "OS") %>%
    mutate(EVENT = 1 - CNSR) %>%
    transmute(USUBJID, ARM, TIME = AVAL, EVENT)

# Combine all ipd data
ipd <- adsl %>%
    full_join(adrs, by = c("USUBJID", "ARM")) %>%
    full_join(adtte, by = c("USUBJID", "ARM"))
head(ipd)
```

```
##   X USUBJID ARM AGE    SEX SMOKE ECOGO N_PR_THER SEX_MALE AGE_SQUARED RESPONSE
## 1 1       1   A  45   Male     0     0         4        1        2025        0
## 2 2       2   A  71   Male     0     0         3        1        5041        1
## 3 3       3   A  58   Male     1     1         2        1        3364        1
## 4 4       4   A  48 Female     0     1         4        0        2304        1
## 5 5       5   A  69   Male     0     1         4        1        4761        0
## 6 6       6   A  48 Female     0     1         4        0        2304        0
##        TIME EVENT
## 1 281.5195     0
## 2 500.0000     0
## 3 304.6406     0
## 4 102.4731     0
## 5 101.6632     0
## 6 237.0593     1
```

Preprocessing aggregate data

There are two ways of specifying aggregate data. One approach is to read in aggregate data using an excel spreadsheet. In the spreadsheet, possible variable types include mean, median, or standard deviation for continuous variables and count or proportion for binary variables. The naming should be followed by these suffixes accordingly: _COUNT, _MEAN, _MEDIAN, _SD, _PROP. Then, `process_agd` will convert the count into proportions.

Other way is to define data frame of aggregate data in R. If you do it this way, _COUNT prefix should not be specified and only proportion is allowed for binary variables. Other suffix names would be the same as the first approach.

Possible missingness in the binary variables should be accounted for by subtracting the denominator by the missing count i.e. proportion = count / (N - missing).

```r
# Through excel spreadsheet target_pop <-
# read.csv(system.file('extdata','aggregate_data_example_1.csv', package =
# 'maicplus', mustWork = TRUE)) target_pop2 <- read.csv(system.file('extdata',
# 'aggregate_data_example_2.csv', package = 'maicplus', mustWork = TRUE))
# target_pop3 <- read.csv(system.file('extdata',
# 'aggregate_data_example_3.csv', package = 'maicplus', mustWork = TRUE)) agd
```

```
# <- process_agd(target_pop)

# Second approach by defining a data frame in R
agd <- data.frame(STUDY = "Lung study", ARM = "Total", N = 300, AGE_MEAN = 51, AGE_MEDIAN = 49,
    AGE_SD = 3.25, SEX_MALE_PROP = 147/300, ECOGO_PROP = 0.4, ECOGO_COUNT = 40, SMOKE_PROP = 58/(300 -
        5), N_PR_THER_MEDIAN = 2)
```

Preprocessing aggregate data

```
#### prepare data
ipd_centered <- center_ipd(ipd = ipd, agd = agd)
head(ipd_centered)
```

```
##   X USUBJID ARM AGE    SEX SMOKE ECOGO N_PR_THER SEX_MALE AGE_SQUARED RESPONSE
## 1 1       1   A  45   Male     0     0         4        1        2025        0
## 2 2       2   A  71   Male     0     0         3        1        5041        1
## 3 3       3   A  58   Male     1     1         2        1        3364        1
## 4 4       4   A  48 Female     0     1         4        0        2304        1
## 5 5       5   A  69   Male     0     1         4        1        4761        0
## 6 6       6   A  48 Female     0     1         4        0        2304        0
##       TIME EVENT AGE_CENTERED AGE_MEDIAN_CENTERED AGE_SQUARED_CENTERED
## 1 281.5195     0           -6                -0.5            -586.5625
## 2 500.0000     0           20                 0.5            2429.4375
## 3 304.6406     0            7                 0.5             752.4375
## 4 102.4731     0           -3                -0.5            -307.5625
## 5 101.6632     0           18                 0.5            2149.4375
## 6 237.0593     1           -3                -0.5            -307.5625
##   SEX_MALE_CENTERED ECOGO_CENTERED SMOKE_CENTERED N_PR_THER_MEDIAN_CENTERED
## 1              0.51           -0.4     -0.1966102                       0.5
## 2              0.51           -0.4     -0.1966102                       0.5
## 3              0.51            0.6      0.8033898                      -0.5
## 4             -0.49            0.6     -0.1966102                       0.5
## 5              0.51            0.6     -0.1966102                       0.5
## 6             -0.49            0.6     -0.1966102                       0.5
```

How to handle standard deviation aggregate summary

As described by Phillippo et al. 2016, balancing on both mean and standard deviation for continuous variables (where possible) may be considered in some cases. If a standard deviation is provided in the comparator population, preprocessing is done so that in the target population, $E(X^2)$ is calculated using the variance formula $Var(X) = E(X^2) - E(X)^2$. This $E(X^2)$ in the target population is matched with the IPD level data, which is why X^2 was calculated during the preprocessing stage of IPD.

How to handle median aggregate summary

If a median is provided, IPD is preprocessed to categorize the variable into a binary variable. All the values in the IPD that are higher than the comparator population median is assigned a value of 1. Conversely, all values that are lower are assigned a value of 0. Comparator population median is replaced by 0.5 to adjust to the categorization in the IPD data. The newly created IPD binary variable is matched so that the proportion is 0.5.

How to handle missing counts in aggregate data

If there are missing counts in a categorical variable, we subtract the missing counts from the total sample size in the denominator and estimate the proportion with only the observed values. What are we assuming about the missing patients?

If A is the number of event, B is the number of non-event, and N is the number of missing counts, we ignore N and estimate the proportion of event by

$$\frac{A}{A + B}$$

If X is the number of missing event count, alternate way to estimate the proportion would be

$$\frac{A + X}{A + B + N}$$

We would like the two proportion to be the same.

$$\frac{A + X}{A + B + N} = \frac{A}{A + B}$$

If we assume X follows a binomial distribution with total size N and probability of event p and take the expectation on both sides we get

$$\frac{A + Np}{A + B + N} = \frac{A}{A + B}$$

Solving for the equation

$$(A + Np)(A + B) = A(A + B + N)$$
$$A^2 + AB + ANp + BNp = A^2 + AB + AN$$
$$(A + B)Np = AN$$
$$p = \frac{A}{A + B}$$

Thus, by subtracting missing counts N from the total sample size in the denominator, we are assuming that the missing patients have probability of an event that is equal to the observed proportion of an event. This seems like a fair assumption to make if the patients are missing completely at random.

Calculating weights

```
# list variables that are going to be used to match
centered_colnames <- c("AGE", "AGE_SQUARED", "SEX_MALE", "ECOGO", "SMOKE", "N_PR_THER_MEDIAN")
centered_colnames <- paste0(centered_colnames, "_CENTERED")

match_res <- estimate_weights(data = ipd_centered, centered_colnames = centered_colnames)
```

```
## initial  value 500.000000
## iter   10 value 215.753747
## iter   20 value 199.844445
## final  value 199.842237
## converged

# Alternatively, you can specify the numeric column locations for
# centered_colnames match_res <- estimate_weights(ipd_centered,
# centered_colnames = c(14, 16:20))

# Two options of plotting weights: unscaled or scaled
wt <- match_res$data$weights
wt_scaled <- match_res$data$scaled_weights

par(mfrow = c(1, 2))
plot_weights(wt, bin_col = "orange", vline_col = "darkblue")
plot_weights(wt_scaled, main_title = "Scaled Individual Weights")
```
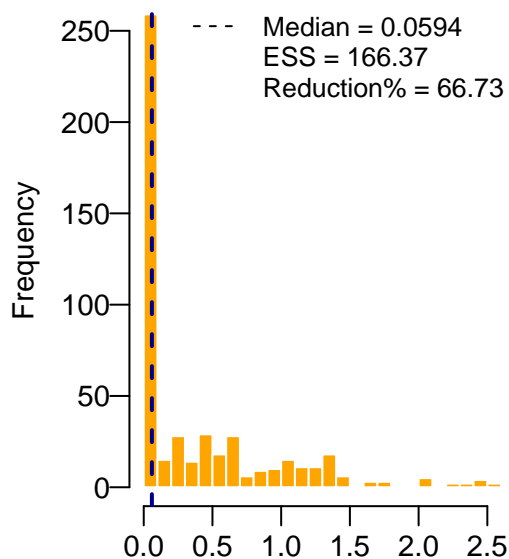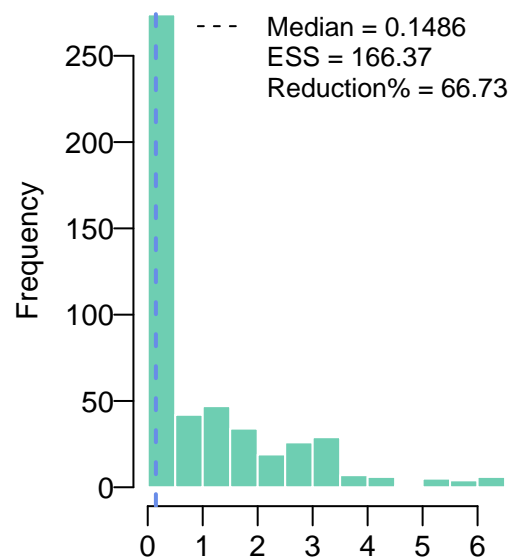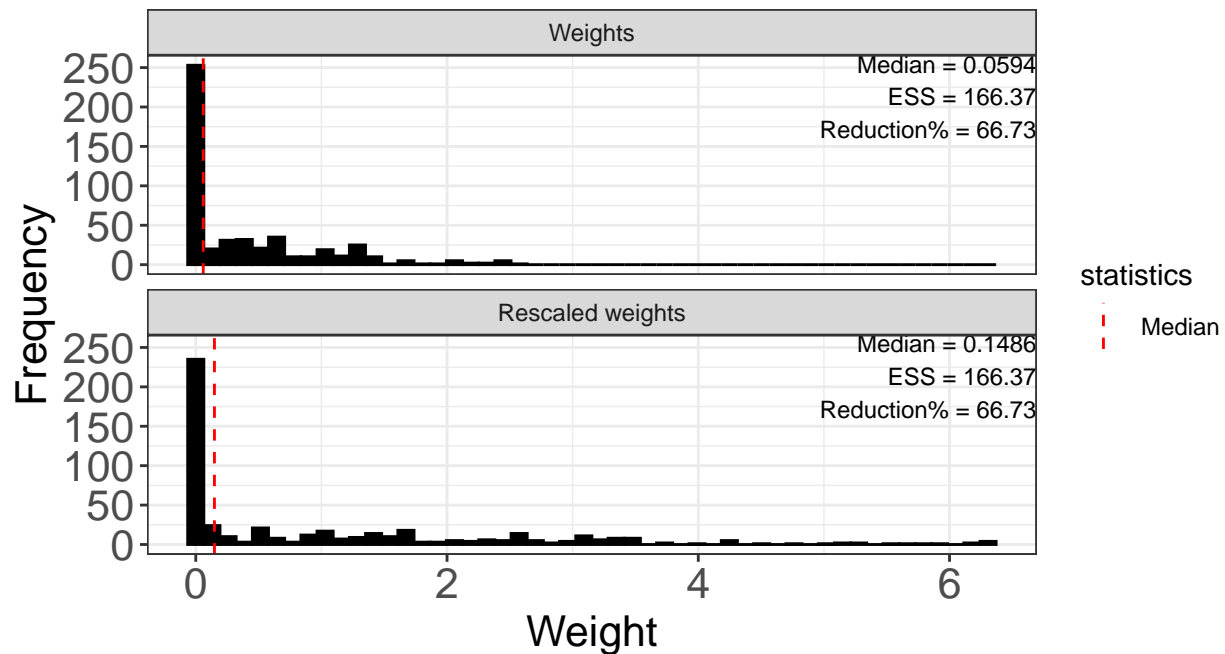
Unscaled Individual Weights

Scaled Individual Weights

There is also an option of printing weights using ggplot.

```r
plot_weights2(match_res, print_caption = TRUE, caption_width = 80)
```


In most applications, weighting considerably reduces the effective sample size from the original AC sample size. The median percentage reduction is 58% (range: 7.9%–94.1%; interquartile range: 42.2%–74.2%). The final effective sample sizes are also representative of those in the technology appraisals, which are also small (median: 80; range: 4.8–639; interquartile range: 37–174). Therefore, an ESS reduction up to ~60% is not unexpected based on the 2021 survey, whereas a reduction of >75% is less common and it may be considered suboptimal.

Another check after the weights are calculated is to look at how the weighted covariates match with the aggregate data summary.

```r
outdata <- check_weights(match_res, agd)
outdata
```

```
##   covariate match_stat internal_trial internal_trial_after_weighted
## 1       AGE       Mean         59.850                         51.00
## 2       AGE         SD          9.011                          3.25
## 3   SEX_MALE       Prop          0.380                          0.49
## 4       ECOG0       Prop          0.410                          0.40
## 5       SMOKE       Prop          0.320                          0.20
## 6 N_PR_THER     Median          3.000                          2.00
##   external_trial sum_centered_IPD_with_weights
## 1          51.00                        0.0001
## 2           3.25                        0.0125
## 3           0.49                        0.0000
## 4           0.40                        0.0000
## 5           0.20                        0.0000
## 6           2.00                        0.0000
```

Time to event analysis

We first need to combine internal IPD data with pseudo comparator IPD. To obtain pseudo comparator
IPD, we would digitize Kaplan Meier curves from the comparator study.

```r
pseudo_ipd <- read.csv(system.file("extdata", "psuedo_IPD.csv", package = "maicplus",
    mustWork = TRUE))
pseudo_ipd$ARM <- "B"  #Need to specify ARM for pseudo ipd
ipd_matched <- match_res$data

combined_data_tte <- merge_two_data(pseudo_ipd, ipd_matched, internal_time_name = "TIME",
    internal_event_name = "EVENT")
str(combined_data_tte)
```

```
## 'data.frame':    800 obs. of  4 variables:
##  $ TIME   : num  282 500 305 102 102 ...
##  $ EVENT  : num  0 0 0 0 0 1 0 1 0 0 ...
##  $ ARM    : Factor w/ 2 levels "B","A": 2 2 2 2 2 2 2 2 2 2 ...
##  $ weights: num  6.88e-01 5.67e-08 1.15e-01 1.18 1.61e-06 ...
```
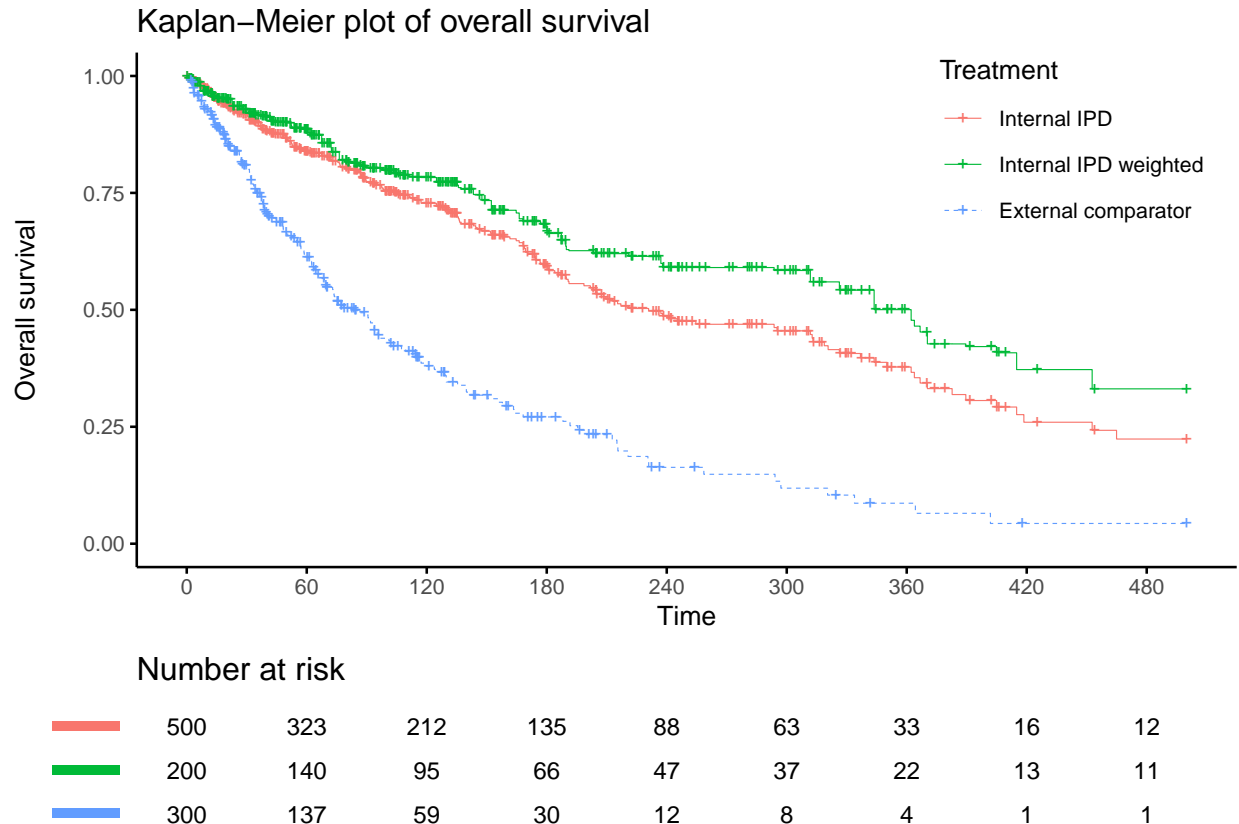
Report 1: Kaplan-Meier plot

```r
kmobj <- survfit(Surv(TIME, EVENT) ~ ARM, combined_data_tte, conf.type = "log-log")
kmobj_adj <- survfit(Surv(TIME, EVENT) ~ ARM, combined_data_tte, weights = combined_data_tte$weight,
    conf.type = "log-log")

par(cex.main = 0.85)
km_plot(kmobj, kmobj_adj, time_scale = "month", trt = "A", trt_ext = "B", endpoint_name = "OS")
```

Kaplan–Meier Curves of Comparator (AgD) and Treatment (IPD)
Endpoint: OS
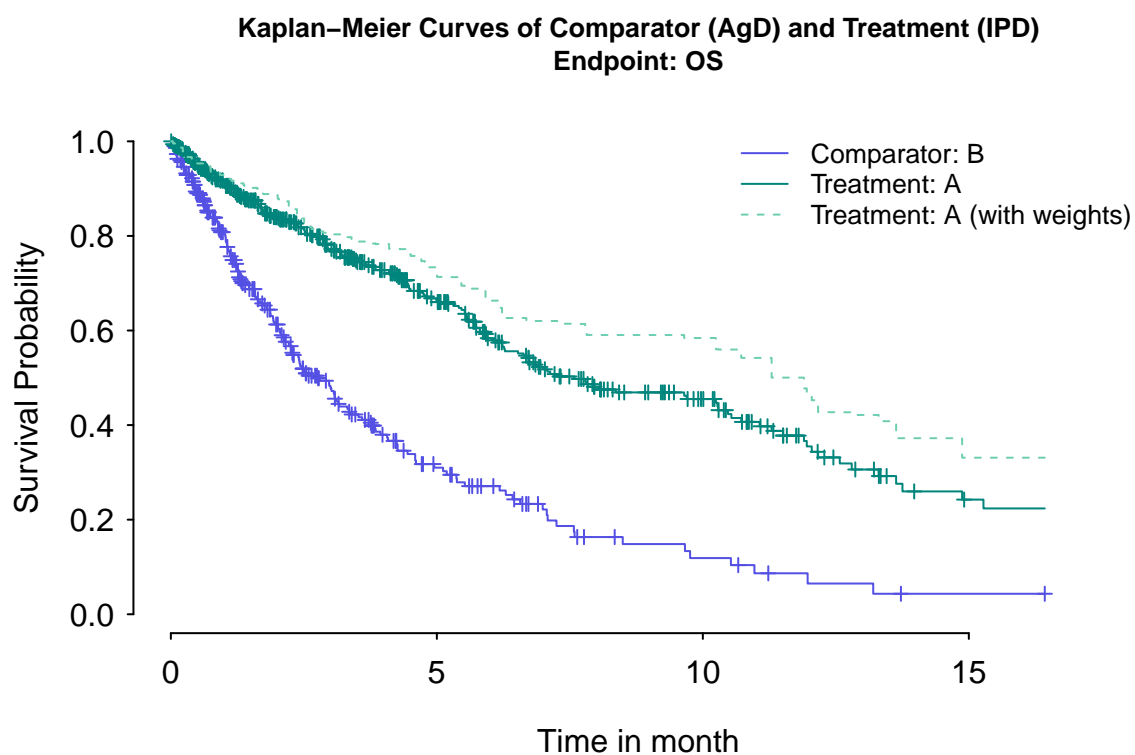
There is also a ggplot option for Kaplan-Meier curves using `survminer` R package.

```
km_plot2(combined_data_tte, trt = "A", trt_ext = "B", censor = TRUE, risk.table = TRUE)
```

Kaplan–Meier plot of overall survival

Number at risk								
500	323	212	135	88	63	33	16	12
200	140	95	66	47	37	22	13	11
300	137	59	30	12	8	4	1	1

Report 2: Analysis table (Cox model) before and after matching, incl Median Survival Time

We can then fit a cox regression model using the combined dataset. For the weight adjusted cox regression, we fit the model with robust standard errors. Along with the hazard ratios, we can also find median survival time using `medSurv_makeup` function. Then, `report_table` function nicely combines the information together and create a result table.

```
# Fit a Cox model with/without weights to estimate the HR
unweighted_cox <- coxph(Surv(TIME, EVENT == 1) ~ ARM, data = combined_data_tte)
weighted_cox <- coxph(Surv(TIME, EVENT == 1) ~ ARM, data = combined_data_tte, weights = combined_data_tt
    robust = TRUE)

# Derive median survival time
medSurv <- medSurv_makeup(kmobj, legend = "before matching", time_scale = "day")
medSurv_adj <- medSurv_makeup(kmobj_adj, legend = "after matching", time_scale = "day")
medSurv_out <- rbind(medSurv, medSurv_adj)
medSurv_out
```

```
##   treatment           type records   n.max   n.start    events    rmean
## 1   ARM=B before matching     300 300.0000  300.0000 178.00000 130.9893
## 2   ARM=A before matching     500 500.0000  500.0000 190.00000 265.1012
## 3   ARM=B  after matching     300 300.0000  300.0000 178.00000 130.9893
## 4   ARM=A  after matching     500 199.8422  199.8422  66.84953 307.7223
```

```
##   se(rmean)    median   0.95LCL  0.95UCL
## 1  10.24910  83.58535  68.82298 101.0786
## 2  10.80981 230.94839 191.10767 313.1574
## 3  10.24910  83.58535  68.82298 101.0786
## 4  16.71338 362.20670 237.05932 452.7209
```

```r
rbind(report_table(unweighted_cox, medSurv, tag = paste0("Before/", "Overall survival")),
      report_table(weighted_cox, medSurv_adj, tag = paste0("After/", "Overall survival")))
```

```
##                    Matching treatment     N n.events(%)      median[95% CI]
## 2  Before/Overall survival    ARM=A 500.0   190(38.0) 230.9[191.1;313.2]
## 1  Before/Overall survival    ARM=B 300.0   178(59.3)  83.6[ 68.8;101.1]
## 21  After/Overall survival    ARM=A 199.8  66.8(33.5) 362.2[237.1;452.7]
## 11  After/Overall survival    ARM=B 300.0   178(59.3)  83.6[ 68.8;101.1]
##           HR[95% CI] WaldTest
## 2  0.37[0.30;0.46]   <0.001
## 1
## 21 0.29[0.21;0.40]   <0.001
## 11
```

Report 3: Bootstrap result

```r
set.seed(1)
HR_bootstraps <- boot(data = ipd_centered, statistic = bootstrap_HR, centered_colnames = centered_colnam
    internal_time_name = "TIME", internal_event_name = "EVENT", pseudo_ipd = pseudo_ipd,
    R = 1000)

# Median of the bootstrap samples
HR_median <- median(HR_bootstraps$t)

# Bootstrap CI - Percentile CI
boot_ci_HR <- boot.ci(boot.out = HR_bootstraps, index = 1, type = "perc")

# Bootstrap CI - BCa CI
boot_ci_HR_BCA <- boot.ci(boot.out = HR_bootstraps, index = 1, type = "bca")

HR_median
```

```
## [1] 0.2858165
```

```r
boot_ci_HR
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = HR_bootstraps, type = "perc", index = 1)
##
## Intervals :
## Level     Percentile
## 95%   ( 0.2236,  0.3689 )
## Calculations and Intervals on Original Scale
```

```
boot_ci_HR_BCA
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = HR_bootstraps, type = "bca", index = 1)
##
## Intervals :
## Level        BCa
## 95%    ( 0.2296,  0.3789 )
## Calculations and Intervals on Original Scale
```
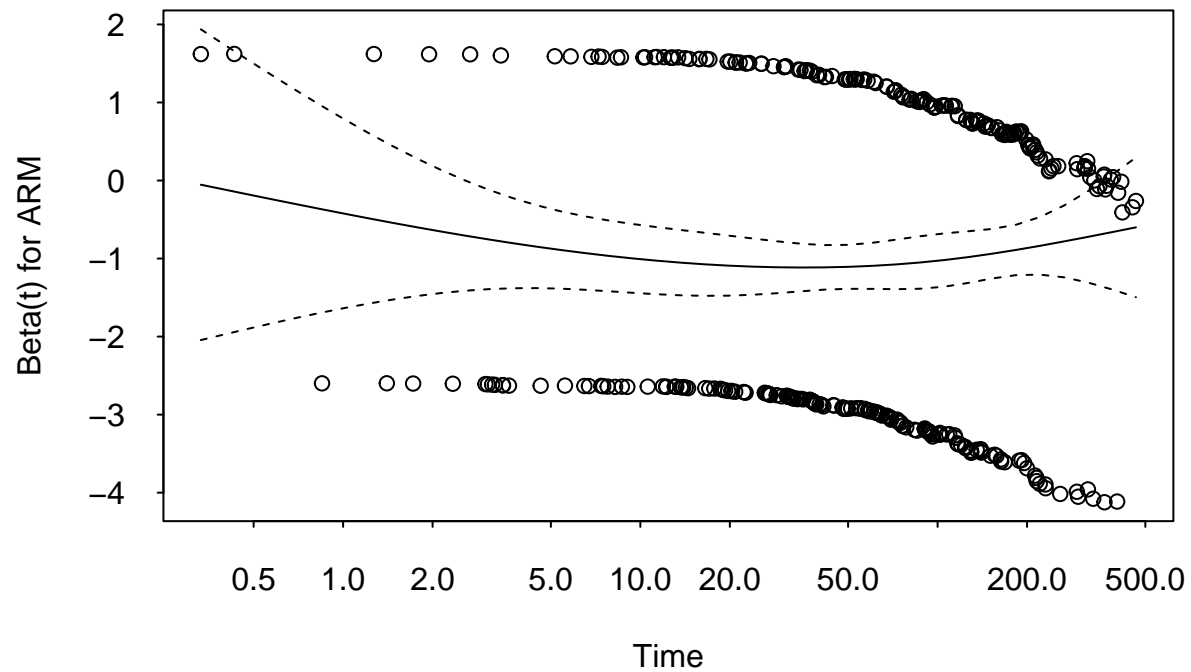
Report 4: Diagnosis Plot

```r
# grambsch & theaneu ph test
coxdiag <- cox.zph(unweighted_cox, global = F, transform = "log")
coxdiag_adj <- cox.zph(weighted_cox, global = F, transform = "log")

coxdiag
```

```
##        chisq df    p
## ARM 0.00996  1 0.92
```

```r
par(mfrow = c(1, 1), tcl = -0.15)
plot(coxdiag, yaxt = "n", main = "Grambsch & Terneau Plot (before matching)")
axis(2, las = 1)
```

Grambsch & Terneau Plot (before matching)
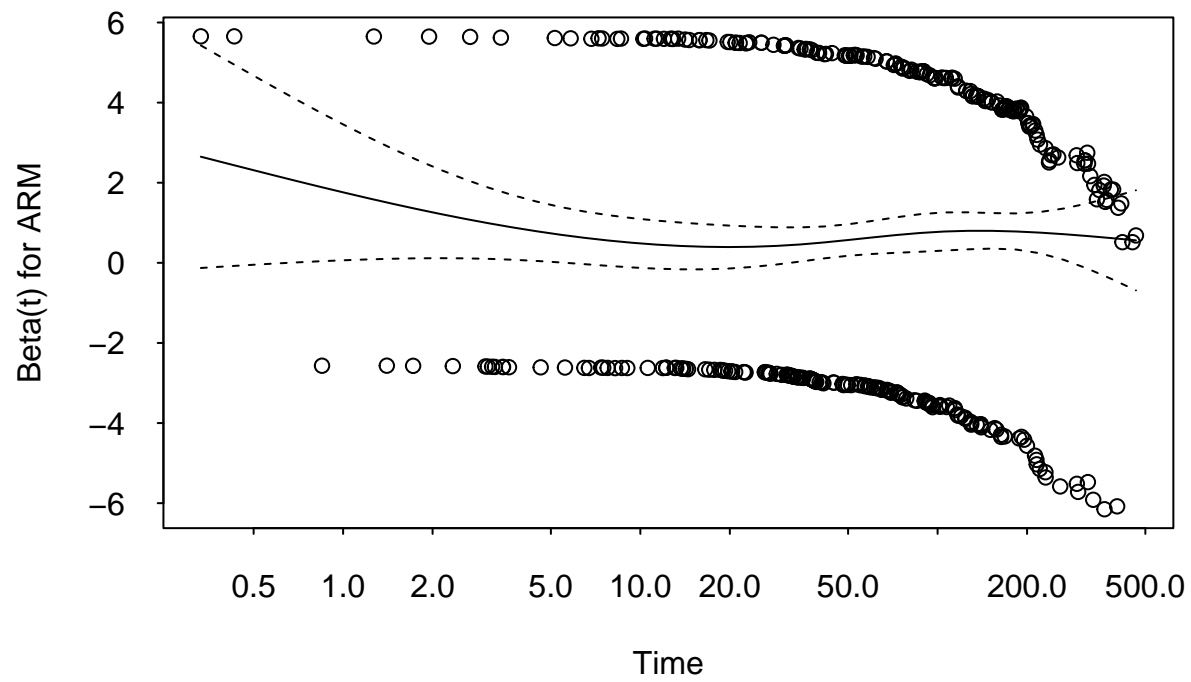


```
coxdiag_adj
```

```
##        chisq df    p
## ARM 0.0438  1 0.83
```
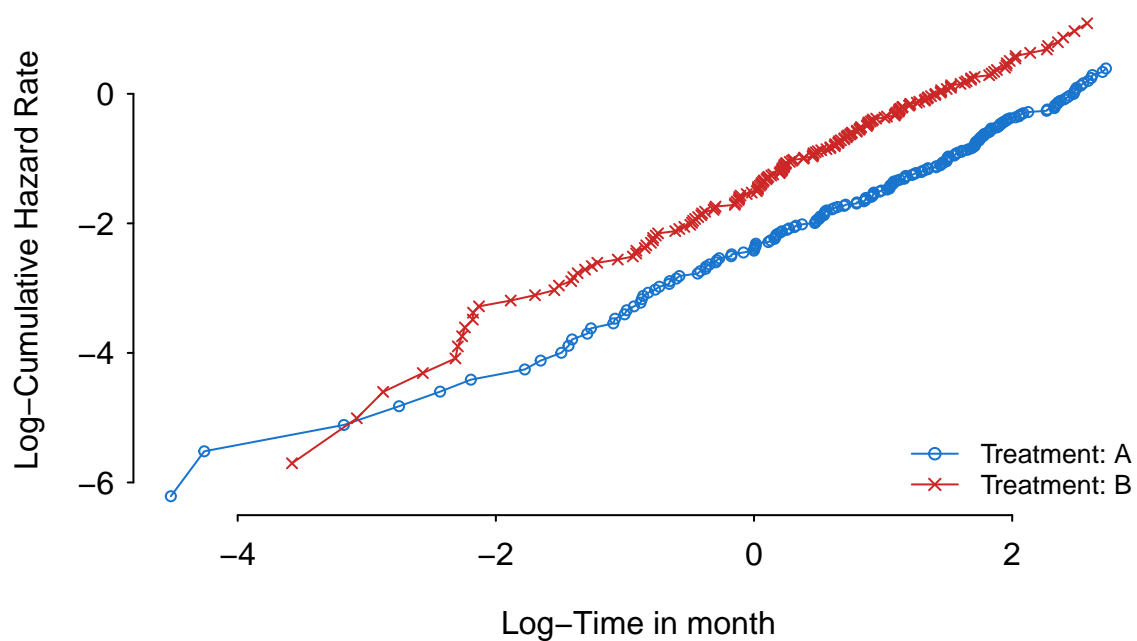
```r
par(mfrow = c(1, 1), tcl = -0.15)
plot(coxdiag_adj, yaxt = "n", main = "Grambsch & Terneau Plot (after matching)")
axis(2, las = 1)
```

**Grambsch & Terneau Plot (after matching)**


```
# log-cumulative hazard plot
log_cum_haz_plot(kmobj, time_scale = "month", log_time = TRUE, endpoint_name = "OS",
    subtitle = "(Before Matching)")
```

Diagnostic plot for Proportional Hazard assumption
Endpoint: OS
(Before Matching)

```r
log_cum_haz_plot(kmobj_adj, time_scale = "month", log_time = TRUE, endpoint_name = "OS",
    subtitle = "(After Matching)")
```

Diagnostic plot for Proportional Hazard assumption
Endpoint: OS
## (After Matching)
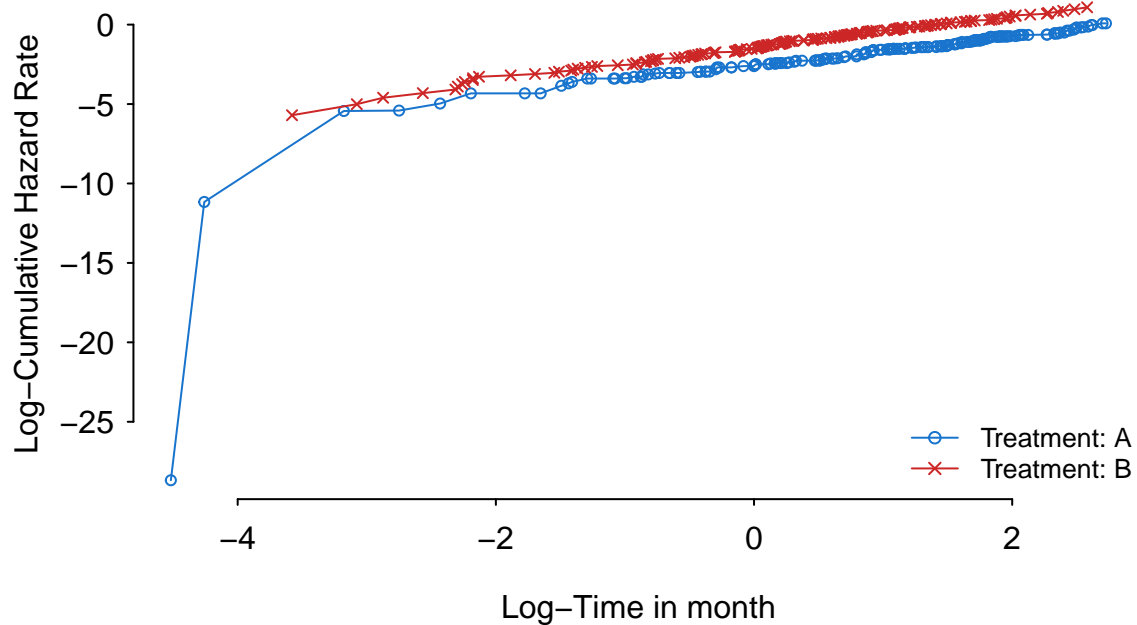

```
# schoenfeld residual plot
resid_plot(unweighted_cox, time_scale = "month", log_time = TRUE, endpoint_name = "OS",
    subtitle = "(Before Matching)")
```

**Diagnosis Plot: Unscaled Schoenfeld Residual
Endpoint: OS
(Before Matching)**
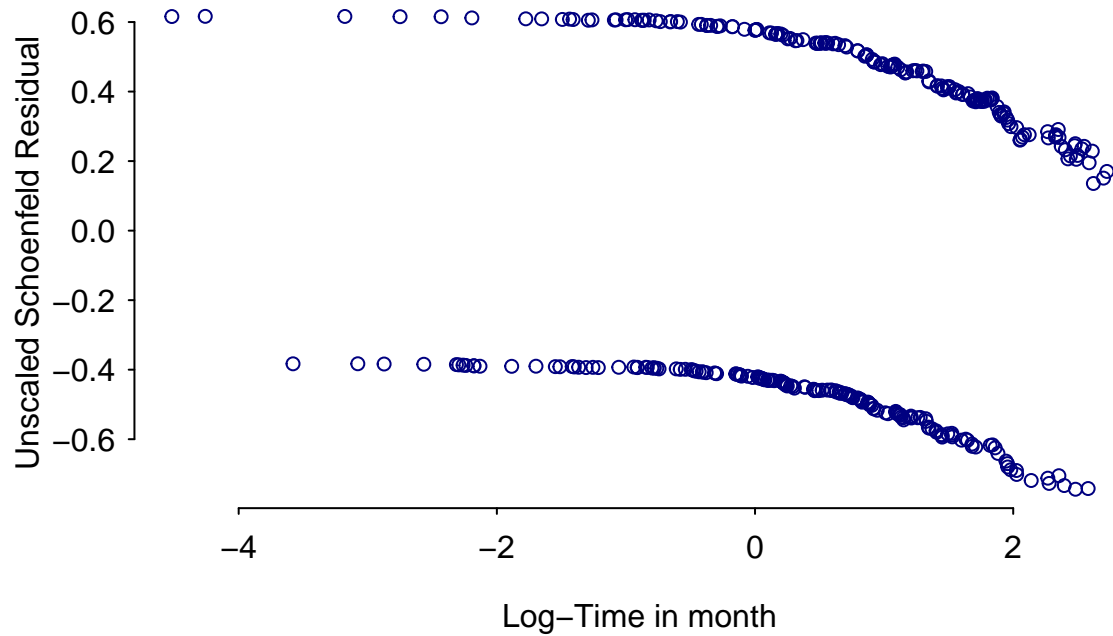

```r
resid_plot(weighted_cox, time_scale = "month", log_time = TRUE, endpoint_name = "OS",
    subtitle = "(After Matching)")
```

Diagnosis Plot: Unscaled Schoenfeld Residual
Endpoint: OS
(After Matching)
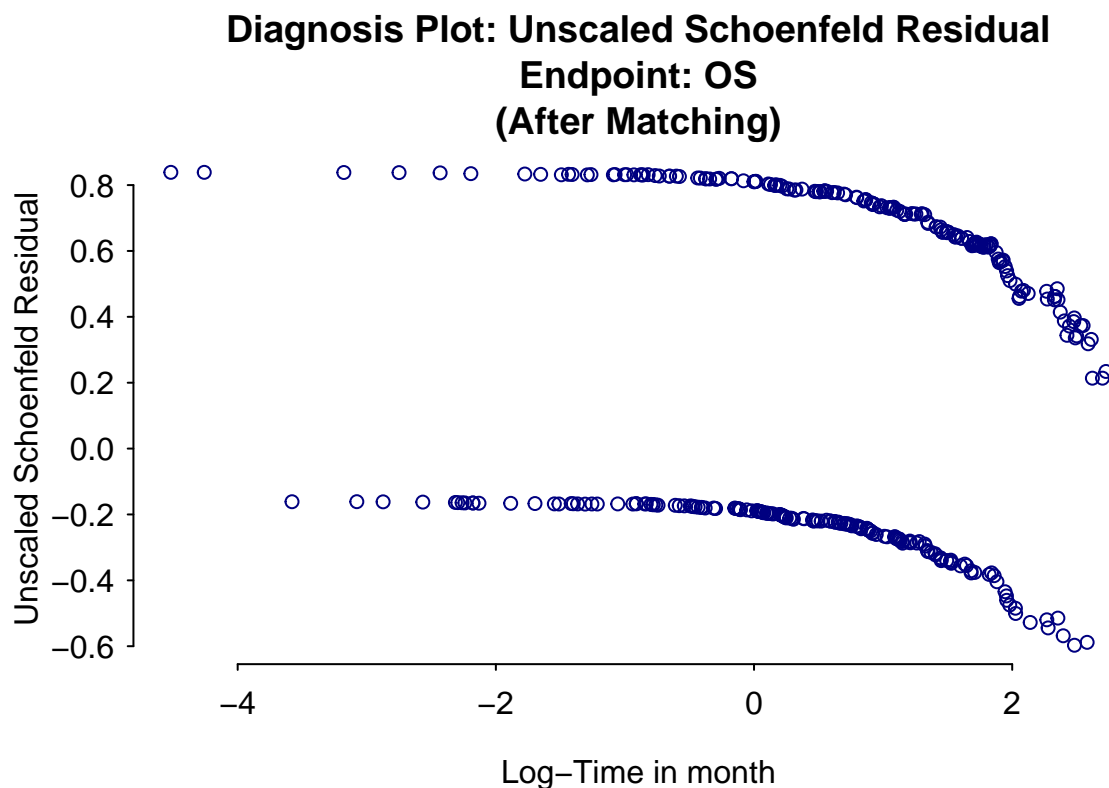
Analysis using a built-in wrapper

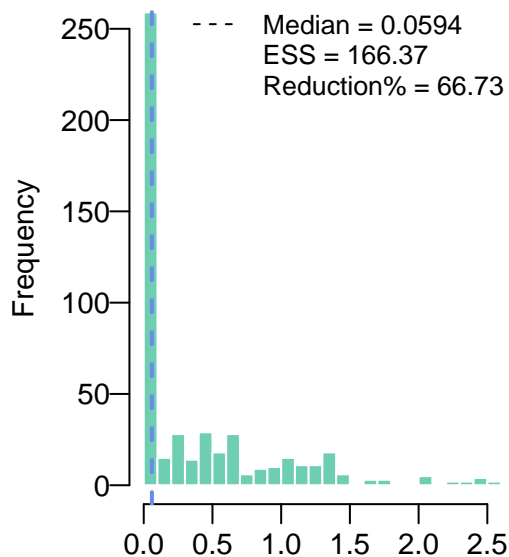One can do all this analysis in a wrapper

```r
fit_unanchored <- maic_tte_unanchored(pseudo_ipd = pseudo_ipd, ipd_matched = ipd_matched,
    internal_time_name = "TIME", internal_event_name = "EVENT", time_scale = "month",
    endpoint_name = "OS", transform = "log")
```

```r
names(fit_unanchored)
```
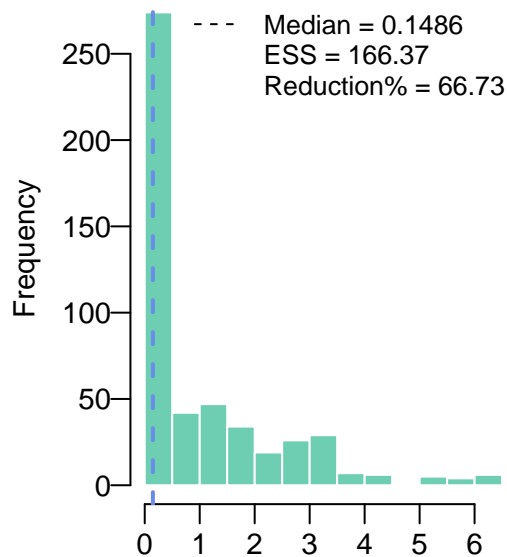
```
##  [1] "plot_weights"       "plot_km"            "fit_km_data_before"
##  [4] "fit_km_data_after"  "report_median_surv" "fit_cox_model_before"
##  [7] "fit_cox_model_after" "report_overall"    "fit_GT_test_before"
## [10] "plot_GT_before"     "fit_GT_test_after"  "plot_GT_after"
## [13] "plot_logCH_before"  "plot_logCH_after"   "plot_resid_before"
## [16] "plot_resid_after"
```

```r
fit_unanchored[["plot_weights"]]
```

Unscaled Individual Weights

- - - Median = 0.0594
ESS = 166.37
Reduction% = 66.73

Scaled Individual Weights

- - - Median = 0.1486
ESS = 166.37
Reduction% = 66.73

Binary outcome analysis (TODO)

```r
# Simulate response data based on the known proportion of responders
comparator_prop_events <- 0.4

# Calculate number with event. Use round() to ensure we end up with a whole
# number of people. number without an event = Total N - number with event to
# ensure we keep the same number of patients
n_with_event <- round(agd$N * comparator_prop_events, digits = 0)
comparator_binary <- data.frame(RESPONSE = c(rep(1, n_with_event), rep(0, agd$N -
    n_with_event)))
comparator_binary$ARM <- "B"  # need to specify ARM for comparator data
ipd_matched <- match_res$data

combined_data_binary <- merge_two_data(comparator_binary, ipd_matched, internal_response_name = "RESPONS

unweighted_OR <- glm(formula = RESPONSE ~ ARM, family = binomial(link = "logit"),
    data = combined_data_binary)

# Log odds ratio
log_OR_CI <- cbind(coef(unweighted_OR), confint.default(unweighted_OR, level = 0.95))[2,
    ]
```

```r
# Odds ratio
OR_CI <- exp(log_OR_CI)
names(OR_CI) <- c("OR", "OR_low_CI", "OR_upp_CI")
OR_CI
```

```
##        OR OR_low_CI OR_upp_CI
##  5.318182  3.887946  7.274549
```

```r
# Fit a logistic regression model with weights to estimate the weighted OR
weighted_OR <- suppressWarnings(glm(formula = RESPONSE ~ ARM, family = binomial(link = "logit"),
    data = combined_data_binary, weight = weights))

# Weighted log odds ratio
log_OR_CI_wtd <- cbind(coef(weighted_OR), confint.default(weighted_OR, level = 0.95))[2,
    ]

# Weighted odds ratio
OR_CI_wtd <- exp(log_OR_CI_wtd)
names(OR_CI_wtd) <- c("OR", "OR_low_CI", "OR_upp_CI")
OR_CI_wtd
```

```
##        OR OR_low_CI OR_upp_CI
##  3.770521  2.567189  5.537897
```

```r
# Robust standard error
vmod <- clubSandwich::vcovCR(weighted_OR, cluster = 1:dim(combined_data_binary)[1],
    type = "CR2")
coef_res <- clubSandwich::conf_int(weighted_OR, vmod, coef = 2)

OR_CI_robust <- exp(with(coef_res, c(beta, CI_L, CI_U)))
names(OR_CI_robust) <- c("Estimate", "Lower 95% CI", "Upper 95% CI")
OR_CI_robust
```

```
##    Estimate Lower 95% CI Upper 95% CI
##    3.770521     2.500835     5.684833
```

```r
# Using sandwich package
V.sw <- sandwich::vcovHC(weighted_OR)  #white's estimator
SD <- sqrt(V.sw[2, 2])
Estimate <- coef(weighted_OR)[2]
OR_CI_robust2 <- exp(c(Estimate, Estimate - 1.96 * SD, Estimate + 1.96 * SD))
names(OR_CI_robust2) <- c("Estimate", "Lower 95% CI", "Upper 95% CI")
OR_CI_robust2
```

```
##    Estimate Lower 95% CI Upper 95% CI
##    3.770521     2.501193     5.684020
```