Using 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 to be the treatment patient received. We assume if the patient received intervention (IPD) and if the patient received comparator treatment. The causal effect of treatment vs on the mean of the outcome can be estimated as below

where is the odds that patient received treatment vs (i.e. enrolls in aggregate data study vs IPD study) given baseline characteristics . Thus, the patients receiving are re-weighted to match the distribution of patients receiving . Note that this causal effect would be the case when the outcome is continuous. If the outcome is binary, 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 to follow logistic regression form

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

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

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

Note that this is the first derivative of

which has second derivative

Since is positive-definite for all , is convex and any finite solution from the equation is unique and corresponds to the global minimum of . Thus, we can use optimization methods to calculate .

# 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
* Pseudo 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 internal treatment and comparator treatment.

# Preprocessing

#devtools::install\_github("hta-pharma/maicplus", ref = "main")  
#library(maicplus)  
devtools::load\_all() # for development purposes  
#> ℹ Loading maicplus

Additional suggested packages for this vignette:

library(dplyr) # this is used for data merging/cleaning. Package itself does not depend on dplyr  
#>   
#> Attache Paket: 'dplyr'  
#> Die folgenden Objekte sind maskiert von 'package:maicplus':  
#>   
#> filter, lag, select  
#> Das folgende Objekt ist maskiert 'package:testthat':  
#>   
#> matches  
#> Die folgenden Objekte sind maskiert von 'package:stats':  
#>   
#> filter, lag  
#> Die folgenden Objekte sind maskiert von 'package:base':  
#>   
#> intersect, setdiff, setequal, union  
  
library(clubSandwich) # For robust standard error in logistic regression  
#> Registered S3 method overwritten by 'clubSandwich':  
#> method from   
#> bread.mlm sandwich  
library(sandwich)  
  
library(survival)  
library(survminer) # for ggsurvplot  
#> Lade nötiges Paket: ggplot2  
#> Lade nötiges Paket: ggpubr  
#>   
#> Attache Paket: 'survminer'  
#> Das folgende Objekt ist maskiert 'package:survival':  
#>   
#> myeloma  
library(ggplot2) # for ggplot functions  
library(boot) # for bootstrapping  
#>   
#> Attache Paket: 'boot'  
#> Das folgende Objekt ist maskiert 'package:survival':  
#>   
#> aml

## 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"))  
  
# 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 ECOG0 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).

# Through excel spreadsheet  
# target\_pop <- read.csv(system.file("extdata","aggregate\_data\_example\_1.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,  
 ECOG0\_PROP = 0.40,  
 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 ECOG0 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 ECOG0\_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

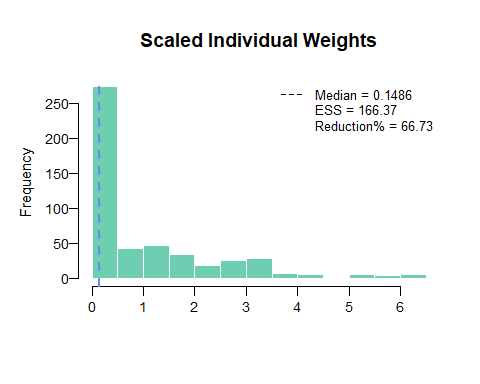
As described by Phillippo et al. 2016, balancing on both mean and standard deviation for continuous variables (where possible) may be considered in some cases. If a standard deviation is provided in the comparator population, preprocessing is done so that in the target population, is calculated using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

### How to handle median aggregate summary

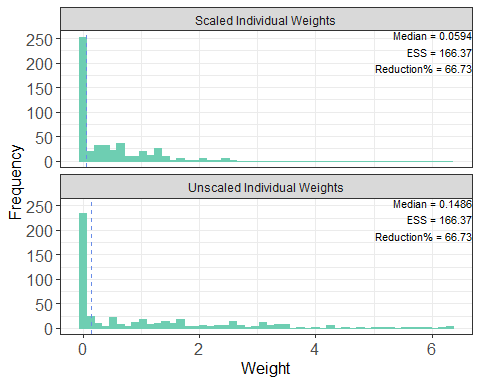
If a median is provided, IPD is preprocessed to categorize the variable into a binary variable. All the values in the IPD that are higher than the comparator population median is assigned a value of 1. Conversely, all values that are lower are assigned a value of 0. Comparator population median is replaced 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.

# Calculating weights

# list variables that are going to be used to match  
centered\_colnames <- c("AGE", "AGE\_SQUARED", "SEX\_MALE", "ECOG0", "SMOKE", "N\_PR\_THER\_MEDIAN")  
centered\_colnames <- paste0(centered\_colnames, "\_CENTERED")  
  
weighted\_data <- 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  
# weighted\_data <- estimate\_weights(ipd\_centered, centered\_colnames = c(14, 16:20))  
  
# Two options to plot weights plot using base R or ggplot  
plot(weighted\_data)



plot(weighted\_data, ggplot = TRUE)



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

outdata <- check\_weights(weighted\_data, 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

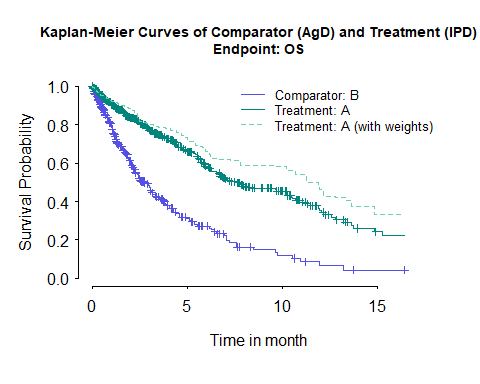
# Time to event analysis

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.

pseudo\_ipd <- read.csv(system.file("extdata", "psuedo\_IPD.csv", package = "maicplus", mustWork = TRUE))  
ipd\_matched <- weighted\_data$data  
  
# Need to specify pseudo\_ipd ARM if not specified  
pseudo\_ipd$ARM <- "B" # Need to specify ARM for pseudo ipd  
  
# Need to specify weights for pseudo\_ipd which is 1  
pseudo\_ipd$weights <- 1  
  
# make sure pseudo\_ipd name has same name compared with ipd\_matched for time, event, arm  
colnames(pseudo\_ipd) <- c("TIME", "EVENT", "ARM", "weights")  
  
combined\_data\_tte <- rbind(  
 ipd\_matched[, colnames(pseudo\_ipd)],  
 pseudo\_ipd  
)  
  
head(combined\_data\_tte)  
#> TIME EVENT ARM weights  
#> 1 281.5195 0 A 6.884028e-01  
#> 2 500.0000 0 A 5.669105e-08  
#> 3 304.6406 0 A 1.146355e-01  
#> 4 102.4731 0 A 1.178307e+00  
#> 5 101.6632 0 A 1.605203e-06  
#> 6 237.0593 1 A 1.178307e+00

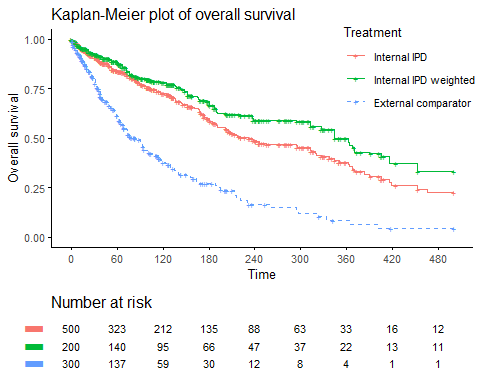
## Report 1: Kaplan-Meier plot

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$weights, 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")



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)  
#> Warning: There was 1 warning in `mutate()`.  
#> ℹ In argument: `survtable = purrr::map2(...)`.  
#> Caused by warning:  
#> ! `select\_()` was deprecated in dplyr 0.7.0.  
#> ℹ Please use `select()` instead.  
#> ℹ The deprecated feature was likely used in the survminer package.  
#> Please report the issue at <https://github.com/kassambara/survminer/issues>.



## 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\_tte$weights, 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=A before matching 500 500.0000 500.0000 190.00000 265.1012  
#> 2 ARM=B before matching 300 300.0000 300.0000 178.00000 130.9893  
#> 3 ARM=A after matching 500 199.8422 199.8422 66.84953 307.7223  
#> 4 ARM=B after matching 300 300.0000 300.0000 178.00000 130.9893  
#> se(rmean) median 0.95LCL 0.95UCL  
#> 1 10.80981 230.94839 191.10767 313.1574  
#> 2 10.24910 83.58535 68.82298 101.0786  
#> 3 16.71338 362.20670 237.05932 452.7209  
#> 4 10.24910 83.58535 68.82298 101.0786  
  
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=B 300.0 178(59.3) 83.6[ 68.8;101.1]  
#> 1 Before/Overall survival ARM=A 500.0 190(38.0) 230.9[191.1;313.2]  
#> 21 After/Overall survival ARM=B 300.0 178(59.3) 83.6[ 68.8;101.1]  
#> 11 After/Overall survival ARM=A 199.8 66.8(33.5) 362.2[237.1;452.7]  
#> HR[95% CI] WaldTest  
#> 2 2.67[2.16;3.29] <0.001  
#> 1   
#> 21 3.46[2.53;4.74] <0.001  
#> 11

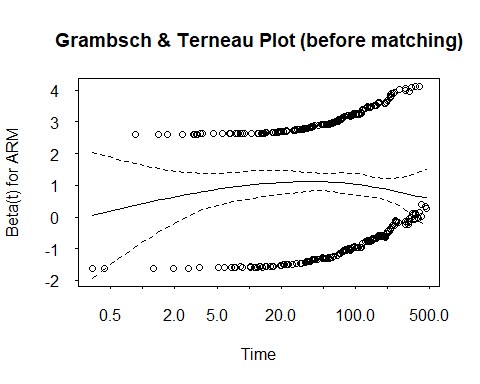
## Report 3: Bootstrap result

set.seed(1)  
HR\_bootstraps <- boot(  
 data = ipd\_centered,  
 statistic = bootstrap\_HR,  
 centered\_colnames = centered\_colnames,  
 pseudo\_ipd = pseudo\_ipd,  
 model = Surv(TIME, EVENT == 1) ~ ARM,  
 ref\_treat = "B",  
 R = 1000  
)

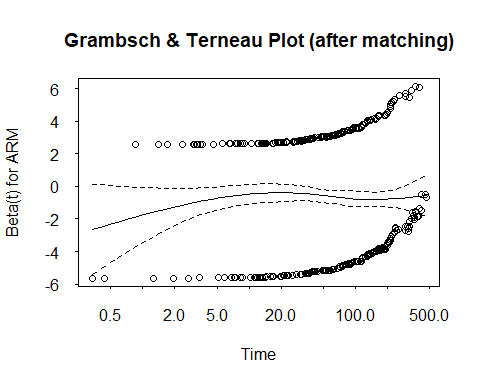
# Median of the bootstrap samples  
HR\_median <- median(HR\_bootstraps$t)  
  
# Bootstrap CI - Percentile CI  
boot\_ci\_HR <- boot.ci(boot.out = HR\_bootstraps, index = 1, type = "perc")  
  
# Bootstrap CI - BCa CI  
boot\_ci\_HR\_BCA <- boot.ci(boot.out = HR\_bootstraps, index = 1, type = "bca")  
  
HR\_median  
#> [1] 0.2858165  
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

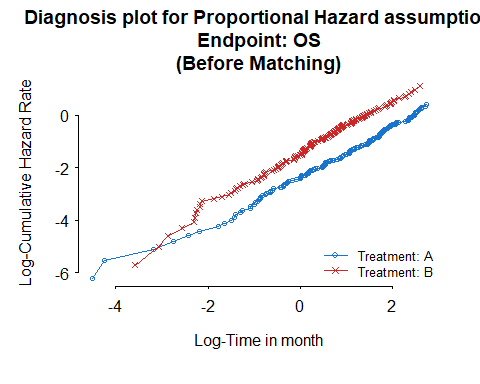
# grambsch & theaneu ph test  
coxdiag <- cox.zph(unweighted\_cox, global = FALSE, transform = "log")  
coxdiag\_adj <- cox.zph(weighted\_cox, global = FALSE, transform = "log")  
  
coxdiag  
#> chisq df p  
#> ARM 0.00996 1 0.92  
par(mfrow = c(1, 1), tcl = -0.15)  
plot(coxdiag, yaxt = "n", main = "Grambsch & Terneau Plot (before matching)")  
axis(2, las = 1)



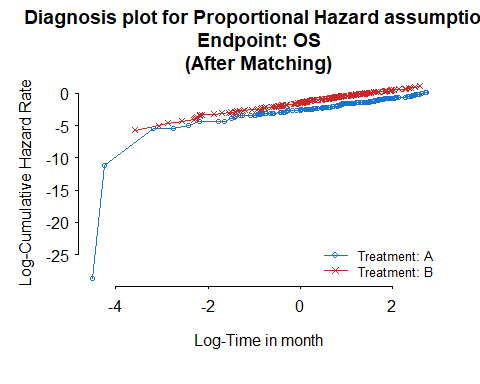
coxdiag\_adj  
#> chisq df p  
#> ARM 0.0438 1 0.83  
par(mfrow = c(1, 1), tcl = -0.15)  
plot(coxdiag\_adj, yaxt = "n", main = "Grambsch & Terneau Plot (after matching)")  
axis(2, las = 1)



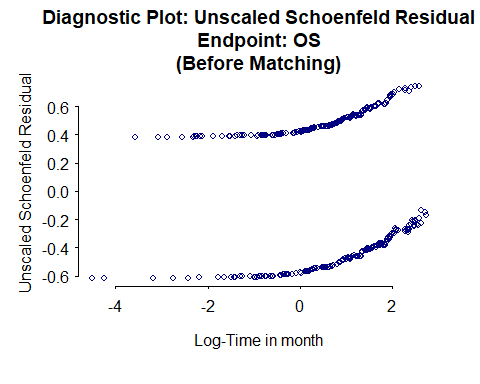
# log-cumulative hazard plot  
log\_cum\_haz\_plot(kmobj,  
 time\_scale = "month", log\_time = TRUE,  
 endpoint\_name = "OS", subtitle = "(Before Matching)"  
)



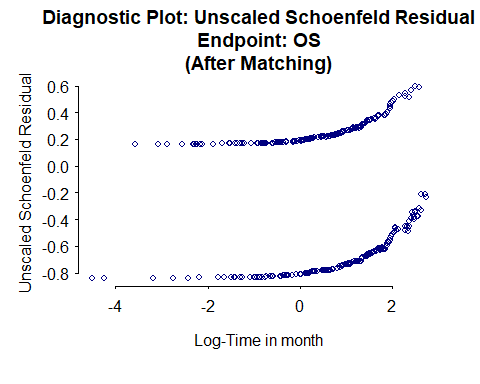
log\_cum\_haz\_plot(kmobj\_adj,  
 time\_scale = "month", log\_time = TRUE,  
 endpoint\_name = "OS", subtitle = "(After Matching)"  
)



# schoenfeld residual plot  
resid\_plot(unweighted\_cox,  
 time\_scale = "month", log\_time = TRUE,  
 endpoint\_name = "OS", subtitle = "(Before Matching)"  
)



resid\_plot(weighted\_cox,  
 time\_scale = "month", log\_time = TRUE,  
 endpoint\_name = "OS", subtitle = "(After Matching)"  
)



## Analysis using a built-in wrapper

One can do all this analysis in a wrapper

# put in wrapper code

# Binary outcome analysis (TODO)

# 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$ARM <- "B" # need to specify ARM for comparator data  
ipd\_matched <- weighted\_data$data  
  
combined\_data\_binary <- merge\_two\_data(comparator\_binary, ipd\_matched, internal\_response\_name = "RESPONSE")  
  
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, ]  
  
# Odds ratio  
OR\_CI <- exp(log\_OR\_CI)  
names(OR\_CI) <- c("OR", "OR\_low\_CI", "OR\_upp\_CI")  
OR\_CI  
  
# 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  
  
# Robust standard error  
vmod <- clubSandwich::vcovCR(weighted\_OR, cluster = seq\_len(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  
  
# 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