MAIC package comparison

Setup

```r
library(dplyr)
library(survival)  # for survfit
library(survminer)  # for ggsurvplot
library(boot)  # for boot

setwd("~/GitHub/maicplus/inst/dev")
source("functions_all.R")

setwd("~/GitHub/maicplus")  # Set your working directory
devtools::load_all()

# Read in relevant ADaM data
adsl <- read.csv(system.file("extdata", "adsl.csv", package = "maicplus",
    mustWork = TRUE))
# Add in a new variable: number of therapies
adsl$n_pr_ther <- sample(1:4, size = dim(adsl)[1], replace = 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))
```

Use dplyr to preprocess

```r
adsl <- adsl %>% # Data containing the matching variables
  mutate(SEX=ifelse(SEX=="Male", 1, 0)) # Coded 1 for males and 0 for females

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

adtte <- adtte %>% # Time to event data (overall survival)
  filter(PARAMCD=="OS") %>%
  mutate(Event=1-CNSR) %>% #Set up coding as Event = 1, Censor = 0
  transmute(USUBJID, ARM, Time=AVAL, Event)

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

```r
# Change to lower case
names(intervention_input) <- tolower(names(intervention_input))
intervention_input <- intervention_input %>% rename(Time = time, Event = event)

# Create a variable for age squared (optional)
intervention_input <- intervention_input %>%
  mutate(age_squared = age^2)
head(intervention_input)
```

```
##   usubjid arm age sex smoke ecog0 n_pr_ther response     Time Event age_squared
## 1       1   A  45   1     0     0         4        0 281.5195     0        2025
## 2       2   A  71   1     0     0         2        1 500.0000     0        5041
## 3       3   A  58   1     1     1         3        1 304.6406     0        3364
## 4       4   A  48   0     0     1         3        1 102.4731     0        2304
## 5       5   A  69   1     0     1         4        0 101.6632     0        4761
## 6       6   A  48   0     0     1         2        0 237.0593     1        2304
```

```r
match_cov <- c("age", "age_squared", "sex", "smoke", "ecog0", "n_pr_ther")
```

Get Aggregate data

```r
# Baseline aggregate data for the comparator population
# Getting data from csv
# target_pop <- read.csv(system.file("extdata", "aggregate_data_updated.csv",
#                                   package = "MAIC", mustWork = TRUE))
# target_pop_standard <- target_pop %>%
#   rename(N = N,
#          Treatment = ARM,
#          age_mean = age.mean,
#          sex_prop = prop.male,
#          smoke_prop = prop.smoke,
#          ecog0_prop = prop.ecog0
#   ) %>%
#   mutate(AGE_SQUARED = AGE^2 + age.sd^2) %>%
#   select(N, Treatment, age_mean, sex_prop, smoke_prop, ecog0_prop)

# Prior step:
# If the specified data is in count form: Requires N, count, and possible missing

# Define target_pop without excel
target_pop <- data.frame(
  N = 300,
  age_mean = 50.06,
  age_sd = 3.23,
  sex_prop = 147/300, #male proportion
  smoke_prop = 58/(300-2), #2 missing patients
  ecog0_prop = 105/300,
  n_pr_ther_median = 3 #number of previous therapies
)
```

Preprocess IPD and aggregate level data

Center IPD using aggregate level means, preprocess standard deviations and medians

```
preprocessed <- preprocess_data(intervention_input, target_pop)
intervention_data <- preprocessed$intervention_data
target_pop <- preprocessed$target_pop
```

Calculate weights

```
weights <- estimate_weights(intervention_data = intervention_data,
    match_cov = match_cov)
```

```
## initial  value 500.000000
## iter  10 value 179.479443
## iter  20 value 166.540778
## final  value 166.540363
## converged
```

```
intervention_data <- intervention_data %>%
    mutate(wt = weights$wt, ARM = "Intervention")
```

Summarize/checking weights

```
weight_summ <- summarize_wts(weights)
weight_summ
```

```
##               type      mean        sd     median         min       max
## 1          Weights 0.3330807 0.5820509 0.02272342 3.22158e-12  3.625332
## 2 Rescaled weights 1.0000000 1.7474769 0.06822196 9.67207e-12 10.884243
```

```
profile_data <- intervention_data %>%
    mutate(wt = weights$wt, wt_rs = weights$wt_rs)
profile_data <- profile_data[!duplicated(profile_data[, match_cov]),
    c(match_cov, "wt", "wt_rs")]
head(profile_data)
```

```
##   age age_squared sex smoke ecog0 n_pr_ther           wt        wt_rs
## 1  45        2025   1     0     0         1 1.853520e+00 5.564776e+00
## 2  71        5041   1     0     0         0 1.712643e-08 5.141825e-08
## 3  58        3364   1     1     1         0 3.972315e-02 1.192598e-01
## 4  48        2304   0     0     1         0 7.022719e-01 2.108413e+00
## 5  69        4761   1     0     1         1 1.102976e-06 3.311439e-06
## 7  47        2209   1     1     0         0 6.227775e-01 1.869749e+00
```

```
check_weights(intervention_data, target_pop, weights, match_cov)
```

```
##                      ARM ESS    age age_squared   sex     smoke ecog0 n_pr_ther
## 1          Intervention 500 59.846    3662.578 0.384 0.3200000 0.406     0.246
## 2 Intervention_weighted 124 50.060    2516.436 0.490 0.1946308 0.350     0.500
## 3            Comparator 300 50.060    2516.436 0.490 0.1946309 0.350     0.500
```

Comparator pseudo data

```
# Read in digitised pseudo survival data, col names must
# match intervention_input
comparator_surv <- read.csv(system.file("extdata", "psuedo_IPD.csv",
    package = "MAIC", mustWork = TRUE))
comparator_input <- comparator_surv %>%
    mutate(wt = 1, ARM = "Comparator")

combined_data <- bind_rows(intervention_data, comparator_input)
combined_data$ARM <- relevel(as.factor(combined_data$ARM), ref = "Comparator")
combined_data$usubjid <- seq(dim(combined_data)[1])
```

Cox model

```
# Fit a Cox model without weights to estimate the
# unweighted HR
unweighted_cox <- coxph(Surv(Time, Event == 1) ~ ARM, data = combined_data)

HR_CI_cox <- summary(unweighted_cox)$conf.int %>%
    as.data.frame() %>%
    transmute(HR = ‘exp(coef)‘, HR_low_CI = ‘lower .95‘, HR_upp_CI = ‘upper .95‘)
HR_CI_cox
```

```
##                        HR HR_low_CI HR_upp_CI
## ARMIntervention 0.3748981  0.303901 0.4624815
```

```
# Fit a Cox model with weights to estimate the weighted HR
weighted_cox <- coxph(Surv(Time, Event == 1) ~ ARM, data = combined_data,
    weights = wt)

HR_CI_cox_wtd <- summary(weighted_cox)$conf.int %>%
    as.data.frame() %>%
    transmute(HR = ‘exp(coef)‘, HR_low_CI = ‘lower .95‘, HR_upp_CI = ‘upper .95‘)
HR_CI_cox_wtd
```

```
##                        HR HR_low_CI HR_upp_CI
## ARMIntervention 0.2966692 0.2081751 0.4227818
```

bootstrap CI

```r
# Bootstrap 1000 HRs
HR_bootstraps <- boot(data = intervention_data, # intervention data
                      statistic = bootstrap_HR, # bootstrap the HR (defined in the MAIC package)
                      match_cov = match_cov, # matching variables
                      R=1000, # number of bootstrap samples
                      comparator_input = comparator_input, # comparator pseudo data
                      model = Surv(Time, Event==1) ~ ARM # model to fit
                      )

# 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")
```

```r
HR_median
```

```
## [1] 0.296533
```

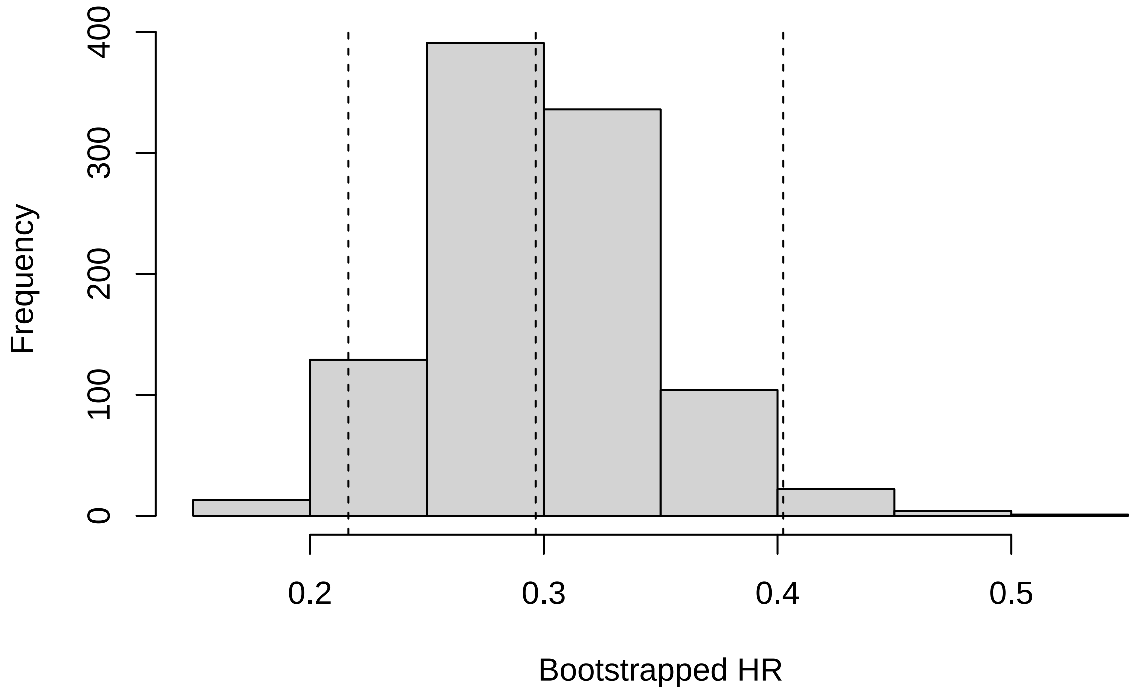
```r
boot_ci_HR
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = HR_bootstraps, type = "perc", index = 1)
##
## Intervals :
## Level     Percentile
## 95%   ( 0.2145,  0.4030 )
## Calculations and Intervals on Original Scale
```

```r
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%   ( 0.2175,  0.4135 )
## Calculations and Intervals on Original Scale
```

```
# Summarize bootstrap estimates in a histogram Vertical
# lines indicate the median and upper and lower CIs
hist(HR_bootstraps$t, main = "", xlab = "Bootstrapped HR")
abline(v = quantile(HR_bootstraps$t, probs = c(0.025, 0.5, 0.975)),
    lty = 2)
```

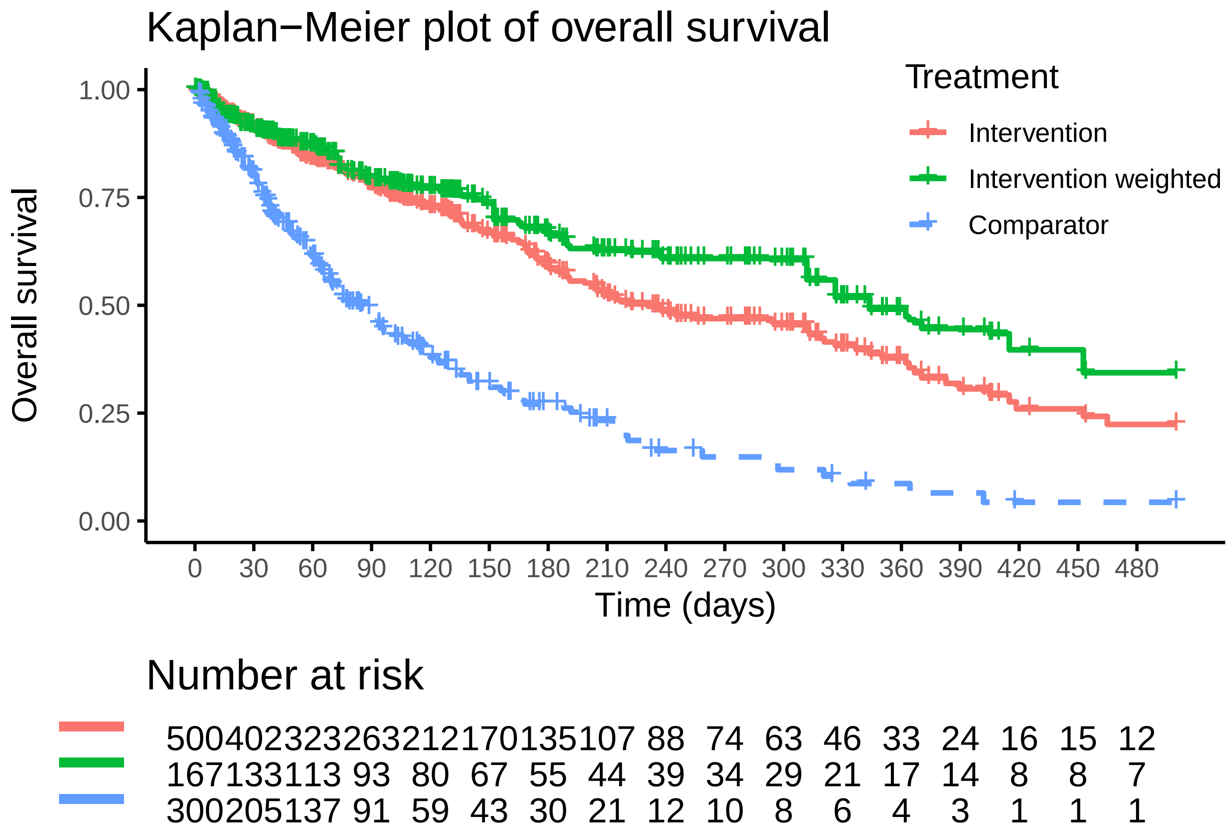


KM plots

```
##### Drawing kaplan meier plots
# Unweighted intervention data
KM_int <- survfit(formula = Surv(Time, Event==1) ~ 1,
                  data = intervention_data,
                  type="kaplan-meier")
# Weighted intervention data
KM_int_wtd <- survfit(formula = Surv(Time, Event==1) ~ 1,
                      data = intervention_data,
                      weights = weights$wt,
                      type="kaplan-meier")
# Comparator data
KM_comp <- survfit(formula = Surv(Time, Event==1) ~ 1,
                   data = comparator_input,
                   type="kaplan-meier")
```

```
# Combine the survfit objects ready for ggsurvplot
KM_list <- list(Intervention = KM_int,
                Intervention_weighted = KM_int_wtd,
                Comparator = KM_comp)

#Produce the Kaplan-Meier plot
KM_plot <- ggsurvplot(KM_list,
                      linetype = c(1,1,2),
                      combine = TRUE,
                      risk.table= TRUE, # numbers at risk displayed on the plot
                      break.x.by= 30, # need to change depending on plotting days/month
                      xlab="Time (days)",
                      ylab="Overall survival", # need?
                      censor=TRUE,
                      legend.title = "Treatment",
                      legend=c(0.85,0.82),
                      title = "Kaplan-Meier plot of overall survival",
                      legend.labs=c("Intervention", "Intervention weighted", "Comparator"),
                      risk.table.y.text.col = T,
                      risk.table.y.text = FALSE,
                      tables.theme = theme_cleantable(),
                      ggtheme = theme_classic(base_size = 13),
                      conf.int = FALSE)
KM_plot
```



Logistic model

```
# Simulate response data based on the known proportion of
# responders
comparator_n <- nrow(comparator_surv)  # total number of patients in the comparator data
comparator_prop_events <- 0.4  # proportion of responders
# 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(comparator_n * comparator_prop_events,
    digits = 0)
comparator_binary <- data.frame(response = c(rep(1, n_with_event),
    rep(0, comparator_n - n_with_event)))

# Join response comparator data (note the rows do not
# represent observations from a particular patient)
comparator_input <- comparator_binary %>%
    mutate(wt = 1, ARM = "Comparator")  # All patients have weight = 1

combined_data <- bind_rows(intervention_data, comparator_input)
combined_data$ARM <- relevel(as.factor(combined_data$ARM), ref = "Comparator")
combined_data$usubjid <- seq(dim(combined_data)[1])
```

```
unweighted_OR <- glm(formula = response ~ ARM, family = binomial(link = "logit"),
    data = combined_data)

# 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
```

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

```
# 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,
    weight = wt))

# 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
##  4.269391  2.815802  6.473360
```

```
# Robust standard error
vmod <- clubSandwich::vcovCR(weighted_OR, cluster = combined_data$usubjid,
    type = "CR2")
coef_res <- clubSandwich::conf_int(weighted_OR, vmod, coef = 2)

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

```
##     Estimate Lower 95% CI Upper 95% CI           SE
##    1.4514711    1.0051866    1.8977556    0.2265359
```

```
# Bootstrap 1000 HRs
OR_bootstraps <- boot(data = intervention_data, # intervention data
                      statistic = bootstrap_OR, # bootstrap the HR (defined in the MAIC package)
                      R=1000, # number of bootstrap samples
                      match_cov = match_cov, # matching variables
                      comparator_input = comparator_input, # comparator pseudo data
                      model = 'response ~ ARM' # model to fit
                      )

# Median of the bootstrap samples
OR_median <- median(OR_bootstraps$t)

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

# Bootstrap CI - BCa CI
boot_ci_OR_BCA <- boot.ci(boot.out = OR_bootstraps, index=1, type="bca")
```

```
OR_median
```

```
## [1] 4.36985
```

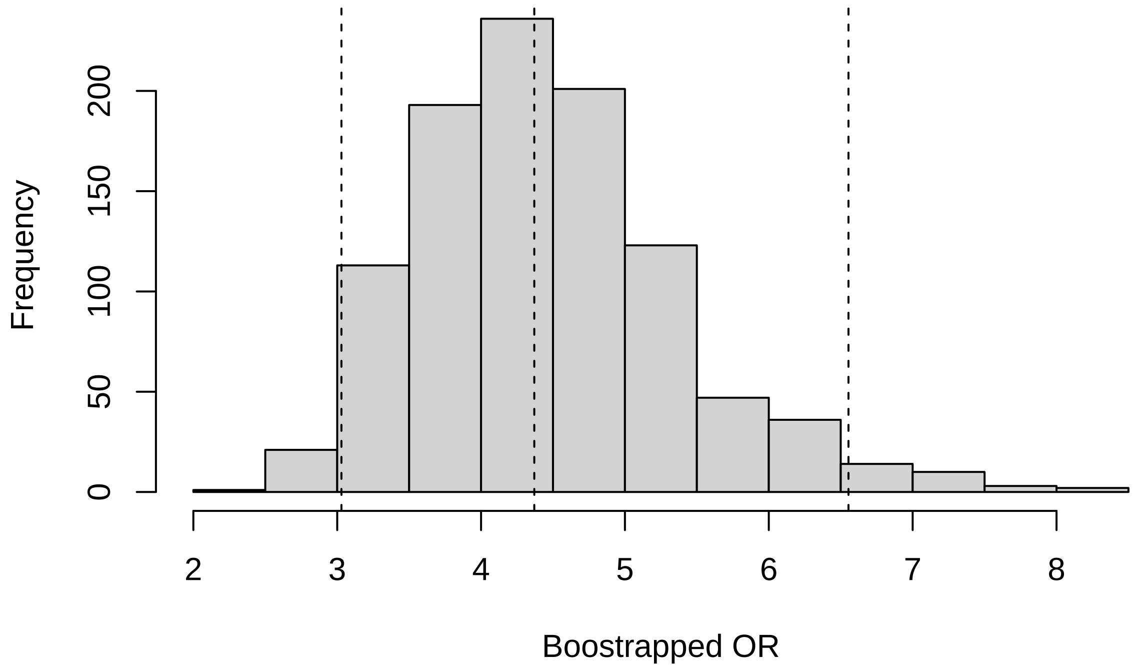
```
boot_ci_OR
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = OR_bootstraps, type = "perc", index = 1)
##
## Intervals :
## Level     Percentile
## 95%   ( 3.018,  6.579 )
## Calculations and Intervals on Original Scale
```

```
boot_ci_OR_BCA
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = OR_bootstraps, type = "bca", index = 1)
##
## Intervals :
## Level       BCa
## 95%   ( 2.808,  6.159 )
## Calculations and Intervals on Original Scale
```

```
# Summarize bootstrap estimates in a histogram Vertical
# lines indicate the median and upper and lower CIs
hist(OR_bootstraps$t, main = "", xlab = "Boostrapped OR")
abline(v = quantile(OR_bootstraps$t, probs = c(0.025, 0.5, 0.975)),
    lty = 2)
```



Things to add

bucher