Package: MAIC (via r-universe)

August 17, 2024

Type Package

Title Package to Perform Matched-adjusted Indirect Comparisons

Version 0.3.0

Description A group of functions designed to perform matched-adjusted indirect comparisons as per NICE TSD 18 and summarise the results for survival data (using hazard ratios) and binary data (using odds ratios.)

License Apache License (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

VignetteBuilder knitr

Imports assertthat (>= 0.2.1), dplyr (>= 0.8.4), tidyselect (>= 1.0.0), ggplot2 (>= 3.3.0), magrittr (>= 1.5), survival (>= 3.1), stats (>= 2.0)

Depends R (>= 2.10)

Suggests knitr, flextable, officer, survminer, testthat, boot

Repository https://pharmaverse.r-universe.dev

RemoteUrl https://github.com/Roche/MAIC

RemoteRef HEAD

RemoteSha 92b1aa5dc19961aeb59df521b95d9230bd462ccc

Contents

`bootstrap_HR` *Bootstrapping for MAIC weighted hazard ratios*

Description

Bootstrapping for MAIC weighted hazard ratios

Usage

```
bootstrap_HR(
  intervention_data,
  matching,
  i,
  model,
  comparator_data,
  min_weight = 1e-04
)
```

Arguments

`intervention_data`	A data frame containing individual patient data from the intervention study.
`matching`	A character vector giving the names of the covariates to use in matching. These names must match the column names in intervention_data.
`i`	Index used to select a sample within `boot`.
`model`	A model formula in the form 'Surv(Time, Event==1) ~ ARM'. Variable names need to match the corresponding columns in intervention_data.
`comparator_data`	A data frame containing pseudo individual patient data from the comparator study needed to derive the relative treatment effect. The outcome variables names must match intervention_data.
`min_weight`	A numeric value that defines the minimum weight allowed. This value (default 0.0001) will replace weights estimated at 0 in a sample.

Details

This function is intended to be used in conjunction with the `boot` function to return the statistic to be bootstrapped. In this case by performing MAIC weighting using estimate_weights and returning a weighted hazard ratio (HR) from a Cox proportional hazards model. This is used as the 'statistic' argument in the boot function.

Value

The HR as a numeric value.

See Also

estimate_weights, boot

Examples

```
# This example code combines weighted individual patient data for 'intervention'
# and pseudo individual patient data for 'comparator' and performs analyses for
# two endpoints: overall survival (a time to event outcome) and objective
# response (a binary outcome)

library(dplyr)
library(boot)
library(survival)
library(MAIC)
library(ggplot2)
library(survminer)
library(flextable)
library(officer)

# load intervention data with weights saved in est_weights
data(est_weights, package = "MAIC")

# Combine data ------------------------------------------------------------------

# Combine the the comparator pseudo data with the analysis data, outputted from
# the estimate_weights function

# Read in digitised pseudo survival data for the comparator
comparator_surv <- read.csv(system.file("extdata", "psuedo_IPD.csv",
                                        package = "MAIC", mustWork = TRUE)) %>%
                            rename(Time=Time, Event=Event)


# 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 survival and response comparator data
# Note the rows do not represent observations from a particular patient
# i.e. there is no relationship between the survival time and response status
# for a given row since this is simulated data
```

```
# In a real data set this relationship would be present
comparator_input <- cbind(comparator_surv, comparator_binary) %>%
  mutate(wt=1, wt_rs=1, ARM="Comparator") # All patients have weight = 1
head(comparator_input)

# Join comparator data with the intervention data (naming of variables should be
# consistent between both datasets)
# Set factor levels to ensure "Comparator" is the reference treatment
combined_data <-  bind_rows(est_weights$analysis_data, comparator_input)
combined_data$ARM <- relevel(as.factor(combined_data$ARM), ref="Comparator")


#### Estimating the relative effect ------------------------------------------

### Example for survival data ------------------------------------------------

## Kaplan-Meier plot

# Unweighted intervention data
KM_int <- survfit(formula = Surv(Time, Event==1) ~ 1 ,
                  data = est_weights$analysis_data,
                  type="kaplan-meier")

# Weighted intervention data
KM_int_wtd <- survfit(formula = Surv(Time, Event==1) ~ 1 ,
                      data = est_weights$analysis_data,
                      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,
                      combine = TRUE,
                      risk.table=TRUE, # numbers at risk displayed on the plot
                      break.x.by=50,
                      xlab="Time (days)",
                      censor=FALSE,
                      legend.title = "Treatment",
                      title = "Kaplan-Meier plot of overall survival",
                      legend.labs=c("Intervention",
                                    "Intervention weighted",
                                    "Comparator"),
                      font.legend = list(size = 10)) +
  guides(colour=guide_legend(nrow = 2))
```

```
## Estimating the hazard ratio (HR)

# 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`)

# 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`)

## Bootstrap the confidence interval of the weighted HR

HR_bootstraps <- boot(data = est_weights$analysis_data, # intervention data
                 statistic = bootstrap_HR, # bootstrap the HR (defined in the MAIC package)
                      R=1000, # number of bootstrap samples
                      comparator_data = comparator_input, # comparator pseudo data
                      matching = est_weights$matching_vars, # matching variables
                      model = Surv(Time, Event==1) ~ ARM # model to fit
  )

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

# 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")

## Summary

# Produce a summary of HRs and CIs
HR_summ <-  rbind(HR_CI_cox, HR_CI_cox_wtd) %>% # Unweighted and weighted HRs and CIs from Cox models
```

```
  mutate(Method = c("HR (95% CI) from unadjusted Cox model",
                    "HR (95% CI) from weighted Cox model")) %>%

  # Median bootstrapped HR and 95% percentile CI
  rbind(data.frame("HR" = HR_median,
                   "HR_low_CI" = boot_ci_HR$percent[4],
                   "HR_upp_CI" = boot_ci_HR$percent[5],
                   "Method"="Bootstrap median HR (95% percentile CI)")) %>%

  # Median bootstrapped HR and 95% bias-corrected and accelerated bootstrap CI
  rbind(data.frame("HR" = HR_median,
                   "HR_low_CI" = boot_ci_HR_BCA$bca[4],
                   "HR_upp_CI" = boot_ci_HR_BCA$bca[5],
                   "Method"="Bootstrap median HR (95% BCa CI)")) %>%
  #apply rounding for numeric columns
  mutate_if(.predicate = is.numeric, sprintf, fmt = "%.3f") %>%
  #format for output
  transmute(Method, HR_95_CI = paste0(HR, " (", HR_low_CI, " to ", HR_upp_CI, ")"))

# Summarize the results in a table suitable for word/ powerpoint
HR_table <- HR_summ %>%
    regulartable() %>% #make it a flextable object
    set_header_labels(Method = "Method",  HR_95_CI = "Hazard ratio (95% CI)") %>%
    font(font = 'Arial', part = 'all') %>%
    fontsize(size = 14, part = 'all') %>%
    bold(part = 'header') %>%
    align(align = 'center', part = 'all') %>%
    align(j = 1, align = 'left', part = 'all') %>%
    border_outer(border = fp_border()) %>%
    border_inner_h(border = fp_border()) %>%
    border_inner_v(border = fp_border()) %>%
    autofit(add_w = 0.2, add_h = 2)


### Example for response data ------------------------------------------------

## Estimating the odds ratio (OR)

# Fit a logistic regression model without weights to estimate the unweighted OR
unweighted_OR <- glm(formula = response~ARM,
                     family = binomial(link="logit"),
                     data = combined_data)

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

# Exponentiate to get Odds ratio
OR_CI_logit <- exp(log_OR_CI_logit)
#tidy up naming
names(OR_CI_logit) <- c("OR", "OR_low_CI", "OR_upp_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,
                                    weight = wt))

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

# Exponentiate to get weighted odds ratio
OR_CI_logit_wtd <- exp(log_OR_CI_logit_wtd)
#tidy up naming
names(OR_CI_logit_wtd) <- c("OR", "OR_low_CI", "OR_upp_CI")

## Bootstrap the confidence interval of the weighted OR
OR_bootstraps <- boot(data = est_weights$analysis_data, # intervention data
                      statistic = bootstrap_OR, # bootstrap the OR
                      R = 1000, # number of bootstrap samples
                      comparator_data = comparator_input, # comparator pseudo data
                      matching = est_weights$matching_vars, # matching variables
                      model = 'response ~ ARM' # model to fit
                      )

## Bootstrapping diagnostics
# 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)

# 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")

## Summary
# Produce a summary of ORs and CIs
OR_summ <- rbind(OR_CI_logit, OR_CI_logit_wtd) %>% # Unweighted and weighted ORs and CIs
  as.data.frame() %>%
  mutate(Method = c("OR (95% CI) from unadjusted logistic regression model",
                    "OR (95% CI) from weighted logistic regression model")) %>%

  # Median bootstrapped HR and 95% percentile CI
  rbind(data.frame("OR" = OR_median,
                   "OR_low_CI" = boot_ci_OR$percent[4],
                   "OR_upp_CI" = boot_ci_OR$percent[5],
                   "Method"="Bootstrap median HR (95% percentile CI)")) %>%

  # Median bootstrapped HR and 95% bias-corrected and accelerated bootstrap CI
  rbind(data.frame("OR" = OR_median,
                   "OR_low_CI" = boot_ci_OR_BCA$bca[4],
                   "OR_upp_CI" = boot_ci_OR_BCA$bca[5],
```

```
                    "Method"="Bootstrap median HR (95% BCa CI)")) %>%
  #apply rounding for numeric columns
  mutate_if(.predicate = is.numeric, sprintf, fmt = "%.3f") %>%
  #format for output
  transmute(Method, OR_95_CI = paste0(OR, " (", OR_low_CI, " to ", OR_upp_CI, ")"))

# turns the results to a table suitable for word/ powerpoint
OR_table <- OR_summ %>%
            regulartable() %>% #make it a flextable object
            set_header_labels(Method = "Method",  OR_95_CI = "Odds ratio (95% CI)")  %>%
            font(font = 'Arial', part = 'all') %>%
            fontsize(size = 14, part = 'all') %>%
            bold(part = 'header') %>%
            align(align = 'center', part = 'all') %>%
            align(j = 1, align = 'left', part = 'all') %>%
            border_outer(border = fp_border()) %>%
            border_inner_h(border = fp_border()) %>%
            border_inner_v(border = fp_border()) %>%
            autofit(add_w = 0.2)
```

`bootstrap_OR` *Bootstrapping for MAIC weighted odds ratios*

Description

Bootstrapping for MAIC weighted odds ratios

Usage

```
bootstrap_OR(
  intervention_data,
  matching,
  i,
  model,
  comparator_data,
  min_weight = 1e-04
)
```

Arguments

`intervention_data`	A data frame containing individual patient data from the intervention study.
`matching`	A character vector giving the names of the covariates to use in matching. These names must match the column names in intervention_data.
`i`	Index used to select a sample within `boot`.
`model`	A model formula in the form 'endpoint ~ treatment_var'. Variable names need to match the corresponding columns in intervention_data.

comparator_data
: A data frame containing pseudo individual patient data from the comparator study needed to derive the relative treatment effect. The outcome variables names must match intervention_data.

min_weight
: A numeric value that defines the minimum weight allowed. This value (default 0.0001) will replace weights estimated at 0 in a sample.

Details

This function is intended to be used in conjunction with the boot function to return the statistic to be bootstrapped. In this case by performing MAIC weighting using estimate_weights and returning a weighted odds ratio (OR) from a logistic regression model. This is used as the 'statistic' argument in the boot function.

Value

The OR as a numeric value.

See Also

estimate_weights, boot

Examples

```
# This example code combines weighted individual patient data for 'intervention'
# and pseudo individual patient data for 'comparator' and performs analyses for
# two endpoints: overall survival (a time to event outcome) and objective
# response (a binary outcome)

library(dplyr)
library(boot)
library(survival)
library(MAIC)
library(ggplot2)
library(survminer)
library(flextable)
library(officer)

# load intervention data with weights saved in est_weights
data(est_weights, package = "MAIC")

# Combine data ------------------------------------------------------------------

# Combine the the comparator pseudo data with the analysis data, outputted from
# the estimate_weights function

# Read in digitised pseudo survival data for the comparator
comparator_surv <- read.csv(system.file("extdata", "psuedo_IPD.csv",
                                        package = "MAIC", mustWork = TRUE)) %>%
                            rename(Time=Time, Event=Event)
```

```
# 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 survival and response comparator data
# Note the rows do not represent observations from a particular patient
# i.e. there is no relationship between the survival time and response status
# for a given row since this is simulated data
# In a real data set this relationship would be present
comparator_input <- cbind(comparator_surv, comparator_binary) %>%
  mutate(wt=1, wt_rs=1, ARM="Comparator") # All patients have weight = 1
head(comparator_input)

# Join comparator data with the intervention data (naming of variables should be
# consistent between both datasets)
# Set factor levels to ensure "Comparator" is the reference treatment
combined_data <-  bind_rows(est_weights$analysis_data, comparator_input)
combined_data$ARM <- relevel(as.factor(combined_data$ARM), ref="Comparator")


#### Estimating the relative effect ------------------------------------------

### Example for survival data ------------------------------------------------

## Kaplan-Meier plot

# Unweighted intervention data
KM_int <- survfit(formula = Surv(Time, Event==1) ~ 1 ,
                  data = est_weights$analysis_data,
                  type="kaplan-meier")

# Weighted intervention data
KM_int_wtd <- survfit(formula = Surv(Time, Event==1) ~ 1 ,
                      data = est_weights$analysis_data,
                      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,
                      combine = TRUE,
                      risk.table=TRUE, # numbers at risk displayed on the plot
                      break.x.by=50,
                      xlab="Time (days)",
                      censor=FALSE,
                      legend.title = "Treatment",
                      title = "Kaplan-Meier plot of overall survival",
                      legend.labs=c("Intervention",
                                    "Intervention weighted",
                                    "Comparator"),
                      font.legend = list(size = 10)) +
  guides(colour=guide_legend(nrow = 2))


## Estimating the hazard ratio (HR)

# 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`)

# 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`)

## Bootstrap the confidence interval of the weighted HR

HR_bootstraps <- boot(data = est_weights$analysis_data, # intervention data
                  statistic = bootstrap_HR, # bootstrap the HR (defined in the MAIC package)
                      R=1000, # number of bootstrap samples
                      comparator_data = comparator_input, # comparator pseudo data
                      matching = est_weights$matching_vars, # matching variables
                      model = Surv(Time, Event==1) ~ ARM # model to fit
  )

## Bootstrapping diagnostics
# Summarize bootstrap estimates in a histogram
# Vertical lines indicate the median and upper and lower CIs
hist(HR_bootstraps$t, main = "", xlab = "Boostrapped HR")
```

```
abline(v= quantile(HR_bootstraps$t, probs = c(0.025, 0.5, 0.975)), lty=2)

# 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")

## Summary

# Produce a summary of HRs and CIs
HR_summ <- rbind(HR_CI_cox, HR_CI_cox_wtd) %>% # Unweighted and weighted HRs and CIs from Cox models
  mutate(Method = c("HR (95% CI) from unadjusted Cox model",
                    "HR (95% CI) from weighted Cox model")) %>%

  # Median bootstrapped HR and 95% percentile CI
  rbind(data.frame("HR" = HR_median,
                   "HR_low_CI" = boot_ci_HR$percent[4],
                   "HR_upp_CI" = boot_ci_HR$percent[5],
                   "Method"="Bootstrap median HR (95% percentile CI)")) %>%

  # Median bootstrapped HR and 95% bias-corrected and accelerated bootstrap CI
  rbind(data.frame("HR" = HR_median,
                   "HR_low_CI" = boot_ci_HR_BCA$bca[4],
                   "HR_upp_CI" = boot_ci_HR_BCA$bca[5],
                   "Method"="Bootstrap median HR (95% BCa CI)")) %>%
  #apply rounding for numeric columns
  mutate_if(.predicate = is.numeric, sprintf, fmt = "%.3f") %>%
  #format for output
  transmute(Method, HR_95_CI = paste0(HR, " (", HR_low_CI, " to ", HR_upp_CI, ")"))

# Summarize the results in a table suitable for word/ powerpoint
HR_table <- HR_summ %>%
    regulartable() %>% #make it a flextable object
    set_header_labels(Method = "Method",  HR_95_CI = "Hazard ratio (95% CI)") %>%
    font(font = 'Arial', part = 'all') %>%
    fontsize(size = 14, part = 'all') %>%
    bold(part = 'header') %>%
    align(align = 'center', part = 'all') %>%
    align(j = 1, align = 'left', part = 'all') %>%
    border_outer(border = fp_border()) %>%
    border_inner_h(border = fp_border()) %>%
    border_inner_v(border = fp_border()) %>%
    autofit(add_w = 0.2, add_h = 2)


### Example for response data ------------------------------------------------

## Estimating the odds ratio (OR)
```

```
# Fit a logistic regression model without weights to estimate the unweighted OR
unweighted_OR <- glm(formula = response~ARM,
                     family = binomial(link="logit"),
                     data = combined_data)

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

# Exponentiate to get Odds ratio
OR_CI_logit <- exp(log_OR_CI_logit)
#tidy up naming
names(OR_CI_logit) <- c("OR", "OR_low_CI", "OR_upp_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,
                                    weight = wt))

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

# Exponentiate to get weighted odds ratio
OR_CI_logit_wtd <- exp(log_OR_CI_logit_wtd)
#tidy up naming
names(OR_CI_logit_wtd) <- c("OR", "OR_low_CI", "OR_upp_CI")

## Bootstrap the confidence interval of the weighted OR
OR_bootstraps <- boot(data = est_weights$analysis_data, # intervention data
                      statistic = bootstrap_OR, # bootstrap the OR
                      R = 1000, # number of bootstrap samples
                      comparator_data = comparator_input, # comparator pseudo data
                      matching = est_weights$matching_vars, # matching variables
                      model = 'response ~ ARM' # model to fit
                      )

## Bootstrapping diagnostics
# 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)

# 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")

## Summary
# Produce a summary of ORs and CIs
```

```
OR_summ <- rbind(OR_CI_logit, OR_CI_logit_wtd) %>% # Unweighted and weighted ORs and CIs
  as.data.frame() %>%
  mutate(Method = c("OR (95% CI) from unadjusted logistic regression model",
                    "OR (95% CI) from weighted logistic regression model")) %>%

  # Median bootstrapped HR and 95% percentile CI
  rbind(data.frame("OR" = OR_median,
                   "OR_low_CI" = boot_ci_OR$percent[4],
                   "OR_upp_CI" = boot_ci_OR$percent[5],
                   "Method"="Bootstrap median HR (95% percentile CI)")) %>%

  # Median bootstrapped HR and 95% bias-corrected and accelerated bootstrap CI
  rbind(data.frame("OR" = OR_median,
                   "OR_low_CI" = boot_ci_OR_BCA$bca[4],
                   "OR_upp_CI" = boot_ci_OR_BCA$bca[5],
                   "Method"="Bootstrap median HR (95% BCa CI)")) %>%
  #apply rounding for numeric columns
  mutate_if(.predicate = is.numeric, sprintf, fmt = "%.3f") %>%
  #format for output
  transmute(Method, OR_95_CI = paste0(OR, " (", OR_low_CI, " to ", OR_upp_CI, ")"))

# turns the results to a table suitable for word/ powerpoint
OR_table <- OR_summ %>%
            regulartable() %>% #make it a flextable object
            set_header_labels(Method = "Method",  OR_95_CI = "Odds ratio (95% CI)")  %>%
            font(font = 'Arial', part = 'all') %>%
            fontsize(size = 14, part = 'all') %>%
            bold(part = 'header') %>%
            align(align = 'center', part = 'all') %>%
            align(j = 1, align = 'left', part = 'all') %>%
            border_outer(border = fp_border()) %>%
            border_inner_h(border = fp_border()) %>%
            border_inner_v(border = fp_border()) %>%
            autofit(add_w = 0.2)
```

`estimate_ess` *Estimate effective sample size*

Description

Estimate the effective sample size (ESS).

Usage

```
estimate_ess(data, wt_col = "wt")
```

Arguments

`data`	A data frame containing individual patient data from the intervention study, including a column containing the weights (derived using `estimate_weights`).

wt_col The name of the weights column in the data frame containing the intervention individual patient data and the MAIC propensity weights. The default is wt.

Details

For a weighted estimate, the effective sample size (ESS) is the number of independent non-weighted individuals that would be required to give an estimate with the same precision as the weighted sample estimate. A small ESS, relative to the original sample size, is an indication that the weights are highly variable and that the estimate may be unstable. This often occurs if there is very limited overlap in the distribution of the matching variables between the populations being compared. If there is insufficient overlap between populations it may not be possible to obtain reliable estimates of the weights

Value

The effective sample size (ESS) as a numeric value.

References

NICE DSU TECHNICAL SUPPORT DOCUMENT 18: METHODS FOR POPULATION-ADJUSTED INDIRECT COMPARISONS IN SUBMSISSIONS TO NICE, REPORT BY THE DECISION SUPPORT UNIT, December 2016

See Also

estimate_weights

Examples

```
# This example code uses the weighted individual patient data, outputted from
# the estimate_weights function to perform weight diagnostics. The weighted data
# is saved within est_weights. To check the weighted aggregate baseline
# characteristics for 'intervention' match those in the comparator data,
# standardized data "target_pop_standard" is used. Please see the package
# vignette for more information on how to use the estimate_weights function and
# derive the "target_pop_standard" data.

library(dplyr)
library(MAIC)

# load est_weights
data(est_weights, package = "MAIC")

# load target_pop_standard
data(target_pop_standard, package = "MAIC")

# List out the uncentered variables used in the matching
match_cov <- c("AGE",
               "SEX",
               "SMOKE",
               "ECOG0")
```

```
# Are the weights sensible? ------------------------------------------------------

# The wt_diagnostics function requires the output from the estimate_weights
# function and will output:
# - the effective sample size (ESS)
# - a summary of the weights and rescaled weights (mean, standard deviation,
#   median, minimum and maximum)
# - a unique set of weights with the corresponding patient profile based on the
#   matching variables

diagnostics <- wt_diagnostics(est_weights$analysis_data,
                              vars = match_cov)

diagnostics$ESS
diagnostics$Summary_of_weights
diagnostics$Weight_profiles

# Each of the wt_diagnostics outputs can also be estimated individually
ESS <- estimate_ess(est_weights$analysis_data)
weight_summ <- summarize_wts(est_weights$analysis_data)
wts_profile <- profile_wts(est_weights$analysis_data, vars = match_cov)

# Plot histograms of unscaled and rescaled weights
# bin_width needs to be adapted depending on the sample size in the data set
histogram <- hist_wts(est_weights$analysis_data, bin = 50)
histogram


# Has the optimization worked? ---------------------------------------------------

# The following code produces a summary table of the intervention baseline
# characteristics before and after matching compared with the comparator
# baseline characteristics:

# Create an object to hold the output
baseline_summary <- list('Intervention' = NA,
                         'Intervention_weighted' = NA,
                         'Comparator' = NA)

# Summarise matching variables for weighted intervention data
baseline_summary$Intervention_weighted <- est_weights$analysis_data %>%
  transmute(AGE, SEX, SMOKE, ECOG0, wt) %>%
  summarise_at(match_cov, list(~ weighted.mean(., wt)))

# Summarise matching variables for unweighted intervention data
baseline_summary$Intervention <- est_weights$analysis_data %>%
  transmute(AGE, SEX, SMOKE, ECOG0, wt) %>%
  summarise_at(match_cov, list(~ mean(.)))

# baseline data for the comparator study
baseline_summary$Comparator <- transmute(target_pop_standard,
                                         AGE,
```

```
                                        SEX,
                                        SMOKE,
                                        ECOG0)

# Combine the three summaries
# Takes a list of data frames and binds these together
trt <- names(baseline_summary)
baseline_summary <-  bind_rows(baseline_summary) %>%
  transmute_all(sprintf, fmt = "%.2f") %>% #apply rounding for presentation
  transmute(ARM = as.character(trt), AGE, SEX, SMOKE, ECOG0)

# Insert N of intervention  as number of patients
baseline_summary$`N/ESS`[baseline_summary$ARM == "Intervention"] <- nrow(est_weights$analysis_data)

# Insert N for comparator from target_pop_standard
baseline_summary$`N/ESS`[baseline_summary$ARM == "Comparator"] <- target_pop_standard$N

# Insert the ESS as the sample size for the weighted data
# This is calculated above but can also be obtained using the estimate_ess function as shown below
baseline_summary$`N/ESS`[baseline_summary$ARM == "Intervention_weighted"] <- est_weights$analysis_data %>%
  estimate_ess(wt_col = 'wt')

baseline_summary <- baseline_summary %>%
  transmute(ARM, `N/ESS`=round(`N/ESS`,1), AGE, SEX, SMOKE, ECOG0)
```

`estimate_weights` *Estimate MAIC propensity weights*

Description

Estimate propensity weights for matching-adjusted indirect comparison (MAIC).

Usage

```
estimate_weights(intervention_data, matching_vars, method = "BFGS", ...)
```

Arguments

`intervention_data`	A data frame containing individual patient data from the intervention study.
`matching_vars`	A character vector giving the names of the covariates to use in matching. These names must match the column names in intervention_data.
`method`	The method used for optimisation - The default is method = "BFGS". Refer to `optim` for options.
`...`	Additional arguments to be passed to optimisation functions such as the method for maximum likelihood optimisation. Refer to `optim` for options.

Details

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.

The aim of the MAIC method is to estimate a set of propensity weights based on prognostic variables and treatment effect modifiers. These weights can be used in subsequent statistical analysis to adjust for differences in patient characteristics between the population in the intervention trial and the population in a comparator study. For additional details on the statistical methods, refer to the package vignette.

The data required for an unanchored MAIC are:

- Individual patient data from a single arm study of 'intervention'
- Aggregate summary data for 'comparator'. This could be from a single arm study of the comparator or from one arm of a randomized controlled trial.
- 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.

For the matching process:

1. All binary variables to be used in the matching should be coded 1 and 0
2. The variable names need to be listed in a character vector called match_cov
3. Aggregate baseline characteristics (number of patients, mean and SD for continuous variables and proportion for binary variables) from the comparator trial are needed as a data frame. Naming of the covariates in this data frame should be consistent with variable names in the intervention data.
4. Patient baseline characteristics in the intervention study are centered on the value of the aggregate data from the comparator study
5. The estimate_weights function can then be used to estimate propensity weights for each patient in the intervention study

For full details refer to the example below and the package vignette

Value

A list containing 2 objects. First, a data frame named analysis_data containing intervention_data with additional columns named wt (weights) and wt_rs (rescaled weights). Second, a vector called matching_vars of the names of the centered matching variables used.

References

NICE DSU TECHNICAL SUPPORT DOCUMENT 18: METHODS FOR POPULATION-ADJUSTED INDIRECT COMPARISONS IN SUBMSISSIONS TO NICE, REPORT BY THE DECISION SUPPORT UNIT, December 2016

See Also

optim

Examples

```
# This example code estimates weights for individual patient data from a single
# arm study of 'intervention' based on aggregate baseline characteristics from
# the comparator trial

library(dplyr)
library(MAIC)


#### Prepare the data -----------------------------------------------------------

### Intervention data

# Read in relevant ADaM data and rename variables of interest
adsl <- read.csv(system.file("extdata", "adsl.csv", package = "MAIC",
                             mustWork = TRUE))
adrs <- read.csv(system.file("extdata", "adrs.csv", package = "MAIC",
                             mustWork = TRUE))
adtte <- read.csv(system.file("extdata", "adtte.csv", package = "MAIC",
                              mustWork = TRUE))

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"))

# List out the variables in the intervention data that have been identified as
# prognostic factors or treatment effect modifiers and will be used in the
# matching
match_cov <- c("AGE",
               "SEX",
               "SMOKE",
               "ECOG0")

## Baseline data from the comparator trial
# Baseline aggregate data for the comparator population
```

```
target_pop <- read.csv(system.file("extdata", "aggregate_data.csv",
                                   package = "MAIC", mustWork = TRUE))

# Rename target population cols to be consistent with match_cov
target_pop_standard <- target_pop %>%
        rename(
           N=N,
           Treatment=ARM,
           AGE=age.mean,
           SEX=prop.male,
           SMOKE=prop.smoke,
           ECOG0=prop.ecog0
              ) %>%
        transmute(N, Treatment, AGE, SEX, SMOKE, ECOG0)


#### Estimate weights -----------------------------------------------------------

### Center baseline characteristics
# (subtract the aggregate comparator data from the corresponding column of
# intervention PLD)
intervention_data <- intervention_input %>%
    mutate(
     Age_centered = AGE - target_pop$age.mean,
     # matching on both mean and standard deviation for continuous variables (optional)
     Age_squared_centered = (AGE^2) - (target_pop$age.mean^2 + target_pop$age.sd^2),
     Sex_centered = SEX - target_pop$prop.male,
     Smoke_centered = SMOKE - target_pop$prop.smoke,
     ECOG0_centered = ECOG0 - target_pop$prop.ecog0)

## Define the matching covariates
cent_match_cov <- c("Age_centered",
                    "Age_squared_centered",
                    "Sex_centered",
                    "Smoke_centered",
                    "ECOG0_centered")

## Optimization procedure
# Following the centering of the baseline characteristics of the intervention
# study, patient weights can be estimated using estimate_weights
# The function output is a list containing (1) a data set of the individual
# patient data with the assigned weights "analysis_data" and (2) a vector
# containing the matching variables "matching_vars"
est_weights <- estimate_weights(intervention_data = intervention_data,
                                matching_vars = cent_match_cov)
```

---

est_weights *est_weights*

Description

An object containing weighted intervention data (est_weights$analysis_data) and a vector of the centered matching variables (est_weights$matching_vars).

Usage

```
est_weights
```

Format

est_weights$analysis_data is a data frame with 500 rows and 16 variables:

USUBJID unique patient number

ARM Name of the arm of the trial the patient was randomized to

AGE Patient age, years

SEX Patient gender: 1 for males and 0 for females

SMOKE Variable to indicate whether the patients smokes: 1 for yes and 0 for no

ECOG0 ECOOG PS: 1 for ECOG PS 0 and 0 for ECOG PS >=1

response Objective response

Time Time to overall survival event/censor

Event Overall survival event = 1, censor = 0

response Objective response

Age_centered Patient's age minus average age in the comparator population

Age_squared_centered Patient's age squared - (Mean age in the target population squared + Standard deviation of age in the target population squared)

Sex_centered SEX variable minus the proportion of males in the comparator population

Smoke_centered SMOKE variable minus the proportion of smokers in the comparator population

ECOG0_centered ECOG0 variable minus the proportion of patients with ECOG0 in the comparator population

wt Propensity weight assigned to patient

wt_rs Propensity weight rescaled

---

hist_wts *Produce histograms of weights and rescaled weights*

Description

Produce a plot containing two histograms (one of the weights and one of the rescaled weights).

Usage

```
hist_wts(data, wt_col = "wt", rs_wt_col = "wt_rs", bin = 30)
```

Arguments

data	A data frame containing individual patient data from the intervention study, including a column containing the weights (derived using estimate_weights).
wt_col	The name of the weights column in the data frame containing the intervention individual patient data and the MAIC propensity weights. The default is wt.
rs_wt_col	The name of the rescaled weights column in the data frame containing the intervention individual patient data and the MAIC propensity weights. The default is wt_rs.
bin	Number of bins to plot histogram. The default is 30.

Value

A histogram plot of the weights and rescaled weights.

See Also

estimate_weights

Examples

```
# This example code uses the weighted individual patient data, outputted from
# the estimate_weights function to perform weight diagnostics. The weighted data
# is saved within est_weights. To check the weighted aggregate baseline
# characteristics for 'intervention' match those in the comparator data,
# standardized data "target_pop_standard" is used. Please see the package
# vignette for more information on how to use the estimate_weights function and
# derive the "target_pop_standard" data.

library(dplyr)
library(MAIC)

# load est_weights
data(est_weights, package = "MAIC")

# load target_pop_standard
data(target_pop_standard, package = "MAIC")

# List out the uncentered variables used in the matching
match_cov <- c("AGE",
               "SEX",
               "SMOKE",
               "ECOG0")

# Are the weights sensible? --------------------------------------------------

# The wt_diagnostics function requires the output from the estimate_weights
# function and will output:
# - the effective sample size (ESS)
# - a summary of the weights and rescaled weights (mean, standard deviation,
#   median, minimum and maximum)
```

```
# - a unique set of weights with the corresponding patient profile based on the
#   matching variables

diagnostics <- wt_diagnostics(est_weights$analysis_data,
                              vars = match_cov)

diagnostics$ESS
diagnostics$Summary_of_weights
diagnostics$Weight_profiles

# Each of the wt_diagnostics outputs can also be estimated individually
ESS <- estimate_ess(est_weights$analysis_data)
weight_summ <- summarize_wts(est_weights$analysis_data)
wts_profile <- profile_wts(est_weights$analysis_data, vars = match_cov)

# Plot histograms of unscaled and rescaled weights
# bin_width needs to be adapted depending on the sample size in the data set
histogram <- hist_wts(est_weights$analysis_data, bin = 50)
histogram


# Has the optimization worked? ------------------------------------------------

# The following code produces a summary table of the intervention baseline
# characteristics before and after matching compared with the comparator
# baseline characteristics:

# Create an object to hold the output
baseline_summary <- list('Intervention' = NA,
                         'Intervention_weighted' = NA,
                         'Comparator' = NA)

# Summarise matching variables for weighted intervention data
baseline_summary$Intervention_weighted <- est_weights$analysis_data %>%
  transmute(AGE, SEX, SMOKE, ECOG0, wt) %>%
  summarise_at(match_cov, list(~ weighted.mean(., wt)))

# Summarise matching variables for unweighted intervention data
baseline_summary$Intervention <- est_weights$analysis_data %>%
  transmute(AGE, SEX, SMOKE, ECOG0, wt) %>%
  summarise_at(match_cov, list(~ mean(.)))

# baseline data for the comparator study
baseline_summary$Comparator <- transmute(target_pop_standard,
                                         AGE,
                                         SEX,
                                         SMOKE,
                                         ECOG0)

# Combine the three summaries
# Takes a list of data frames and binds these together
trt <- names(baseline_summary)
baseline_summary <-  bind_rows(baseline_summary) %>%
```

```
  transmute_all(sprintf, fmt = "%.2f") %>% #apply rounding for presentation
  transmute(ARM = as.character(trt), AGE, SEX, SMOKE, ECOG0)

# Insert N of intervention  as number of patients
baseline_summary$`N/ESS`[baseline_summary$ARM == "Intervention"] <- nrow(est_weights$analysis_data)

# Insert N for comparator from target_pop_standard
baseline_summary$`N/ESS`[baseline_summary$ARM == "Comparator"] <- target_pop_standard$N

# Insert the ESS as the sample size for the weighted data
# This is calculated above but can also be obtained using the estimate_ess function as shown below
baseline_summary$`N/ESS`[baseline_summary$ARM == "Intervention_weighted"] <- est_weights$analysis_data %>%
  estimate_ess(wt_col = 'wt')

baseline_summary <- baseline_summary %>%
  transmute(ARM, `N/ESS`=round(`N/ESS`,1), AGE, SEX, SMOKE, ECOG0)
```

`MAIC` *MAIC: A package for performing matched-adjusted indirect comparisons*

Description

The MAIC package provides functions to help perform and summarize results from matched-adjusted indirect comparisons

---

`profile_wts` *Produce a data frame of the weights assigned to patient profiles*

Description

Select the patient characteristics used in the matching and the MAIC weights and output a data frame of unique propensity weight values with the associated summary baseline characteristics. This data frame helps to understand how different patient profiles are contributing to the analyses by illustrating the patient characteristics associated with different weight values. For example, min, max and median weights. This function is most useful when only matching on binary variables as there are fewer unique values.

Usage

```
profile_wts(data, wt_col = "wt", wt_rs = "wt_rs", vars)
```

Arguments

data	A data frame containing individual patient data from the intervention study, including a column containing the weights (derived using estimate_weights).
wt_col	The name of the weights column in the data frame containing the intervention individual patient data and the MAIC propensity weights. The default is wt.
wt_rs	The name of the rescaled weights column in the data frame containing the intervention individual patient data and the MAIC propensity weights. The default is wt_rs.
vars	A character vector giving the variable names of the baseline characteristics (not centered). These names must match the column names in the data.

Value

A data frame that includes a summary of patient characteristics associated with each weight value.

See Also

estimate_weights

Examples

```
# This example code uses the weighted individual patient data, outputted from
# the estimate_weights function to perform weight diagnostics. The weighted data
# is saved within est_weights. To check the weighted aggregate baseline
# characteristics for 'intervention' match those in the comparator data,
# standardized data "target_pop_standard" is used. Please see the package
# vignette for more information on how to use the estimate_weights function and
# derive the "target_pop_standard" data.

library(dplyr)
library(MAIC)

# load est_weights
data(est_weights, package = "MAIC")

# load target_pop_standard
data(target_pop_standard, package = "MAIC")

# List out the uncentered variables used in the matching
match_cov <- c("AGE",
               "SEX",
               "SMOKE",
               "ECOG0")

# Are the weights sensible? --------------------------------------------------

# The wt_diagnostics function requires the output from the estimate_weights
# function and will output:
# - the effective sample size (ESS)
# - a summary of the weights and rescaled weights (mean, standard deviation,
```

```
#   median, minimum and maximum)
# - a unique set of weights with the corresponding patient profile based on the
#   matching variables

diagnostics <- wt_diagnostics(est_weights$analysis_data,
                              vars = match_cov)

diagnostics$ESS
diagnostics$Summary_of_weights
diagnostics$Weight_profiles

# Each of the wt_diagnostics outputs can also be estimated individually
ESS <- estimate_ess(est_weights$analysis_data)
weight_summ <- summarize_wts(est_weights$analysis_data)
wts_profile <- profile_wts(est_weights$analysis_data, vars = match_cov)

# Plot histograms of unscaled and rescaled weights
# bin_width needs to be adapted depending on the sample size in the data set
histogram <- hist_wts(est_weights$analysis_data, bin = 50)
histogram


# Has the optimization worked? ------------------------------------------------

# The following code produces a summary table of the intervention baseline
# characteristics before and after matching compared with the comparator
# baseline characteristics:

# Create an object to hold the output
baseline_summary <- list('Intervention' = NA,
                         'Intervention_weighted' = NA,
                         'Comparator' = NA)

# Summarise matching variables for weighted intervention data
baseline_summary$Intervention_weighted <- est_weights$analysis_data %>%
  transmute(AGE, SEX, SMOKE, ECOG0, wt) %>%
  summarise_at(match_cov, list(~ weighted.mean(., wt)))

# Summarise matching variables for unweighted intervention data
baseline_summary$Intervention <- est_weights$analysis_data %>%
  transmute(AGE, SEX, SMOKE, ECOG0, wt) %>%
  summarise_at(match_cov, list(~ mean(.)))

# baseline data for the comparator study
baseline_summary$Comparator <- transmute(target_pop_standard,
                                         AGE,
                                         SEX,
                                         SMOKE,
                                         ECOG0)

# Combine the three summaries
# Takes a list of data frames and binds these together
trt <- names(baseline_summary)
```

```
baseline_summary <-  bind_rows(baseline_summary) %>%
  transmute_all(sprintf, fmt = "%.2f") %>% #apply rounding for presentation
  transmute(ARM = as.character(trt), AGE, SEX, SMOKE, ECOG0)

# Insert N of intervention  as number of patients
baseline_summary$`N/ESS`[baseline_summary$ARM == "Intervention"] <- nrow(est_weights$analysis_data)

# Insert N for comparator from target_pop_standard
baseline_summary$`N/ESS`[baseline_summary$ARM == "Comparator"] <- target_pop_standard$N

# Insert the ESS as the sample size for the weighted data
# This is calculated above but can also be obtained using the estimate_ess function as shown below
baseline_summary$`N/ESS`[baseline_summary$ARM == "Intervention_weighted"] <- est_weights$analysis_data %>%
  estimate_ess(wt_col = 'wt')

baseline_summary <- baseline_summary %>%
  transmute(ARM, `N/ESS`=round(`N/ESS`,1), AGE, SEX, SMOKE, ECOG0)
```

`summarize_wts` *Summarize the weight values*

Description

Produce a summary of the weights (minimum, maximum, median, mean, standard deviation). Mean and standard deviation are provided for completeness. In practice the distribution of weights may be skewed in which case mean and SD should be interpreted with caution.

Usage

```
summarize_wts(data, wt_col = "wt", rs_wt_col = "wt_rs")
```

Arguments

`data`	A data frame containing individual patient data from the intervention study, including a column containing the weights (derived using `estimate_weights`).
`wt_col`	The name of the weights column in the data frame containing the intervention individual patient data and the MAIC propensity weights. The default is wt.
`rs_wt_col`	The name of the rescaled weights column in the data frame containing the intervention individual patient data and the MAIC propensity weights. The default is wt_rs.

Value

A data frame that includes a summary (minimum, maximum, median, mean, standard deviation) of the weights and rescaled weights.

See Also

estimate_weights

Examples

```
# This example code uses the weighted individual patient data, outputted from
# the estimate_weights function to perform weight diagnostics. The weighted data
# is saved within est_weights. To check the weighted aggregate baseline
# characteristics for 'intervention' match those in the comparator data,
# standardized data "target_pop_standard" is used. Please see the package
# vignette for more information on how to use the estimate_weights function and
# derive the "target_pop_standard" data.

library(dplyr)
library(MAIC)

# load est_weights
data(est_weights, package = "MAIC")

# load target_pop_standard
data(target_pop_standard, package = "MAIC")

# List out the uncentered variables used in the matching
match_cov <- c("AGE",
               "SEX",
               "SMOKE",
               "ECOG0")

# Are the weights sensible? ------------------------------------------------

# The wt_diagnostics function requires the output from the estimate_weights
# function and will output:
# - the effective sample size (ESS)
# - a summary of the weights and rescaled weights (mean, standard deviation,
#   median, minimum and maximum)
# - a unique set of weights with the corresponding patient profile based on the
#   matching variables

diagnostics <- wt_diagnostics(est_weights$analysis_data,
                              vars = match_cov)

diagnostics$ESS
diagnostics$Summary_of_weights
diagnostics$Weight_profiles

# Each of the wt_diagnostics outputs can also be estimated individually
ESS <- estimate_ess(est_weights$analysis_data)
weight_summ <- summarize_wts(est_weights$analysis_data)
wts_profile <- profile_wts(est_weights$analysis_data, vars = match_cov)

# Plot histograms of unscaled and rescaled weights
# bin_width needs to be adapted depending on the sample size in the data set
```

```
histogram <- hist_wts(est_weights$analysis_data, bin = 50)
histogram


# Has the optimization worked? ------------------------------------------------

# The following code produces a summary table of the intervention baseline
# characteristics before and after matching compared with the comparator
# baseline characteristics:

# Create an object to hold the output
baseline_summary <- list('Intervention' = NA,
                         'Intervention_weighted' = NA,
                         'Comparator' = NA)

# Summarise matching variables for weighted intervention data
baseline_summary$Intervention_weighted <- est_weights$analysis_data %>%
  transmute(AGE, SEX, SMOKE, ECOG0, wt) %>%
  summarise_at(match_cov, list(~ weighted.mean(., wt)))

# Summarise matching variables for unweighted intervention data
baseline_summary$Intervention <- est_weights$analysis_data %>%
  transmute(AGE, SEX, SMOKE, ECOG0, wt) %>%
  summarise_at(match_cov, list(~ mean(.)))

# baseline data for the comparator study
baseline_summary$Comparator <- transmute(target_pop_standard,
                                         AGE,
                                         SEX,
                                         SMOKE,
                                         ECOG0)

# Combine the three summaries
# Takes a list of data frames and binds these together
trt <- names(baseline_summary)
baseline_summary <-  bind_rows(baseline_summary) %>%
  transmute_all(sprintf, fmt = "%.2f") %>% #apply rounding for presentation
  transmute(ARM = as.character(trt), AGE, SEX, SMOKE, ECOG0)

# Insert N of intervention  as number of patients
baseline_summary$`N/ESS`[baseline_summary$ARM == "Intervention"] <- nrow(est_weights$analysis_data)

# Insert N for comparator from target_pop_standard
baseline_summary$`N/ESS`[baseline_summary$ARM == "Comparator"] <- target_pop_standard$N

# Insert the ESS as the sample size for the weighted data
# This is calculated above but can also be obtained using the estimate_ess function as shown below
baseline_summary$`N/ESS`[baseline_summary$ARM == "Intervention_weighted"] <- est_weights$analysis_data %>%
  estimate_ess(wt_col = 'wt')

baseline_summary <- baseline_summary %>%
  transmute(ARM, `N/ESS`=round(`N/ESS`,1), AGE, SEX, SMOKE, ECOG0)
```

target_pop_standard *target_pop_standard*

Description

A data frame containing the aggregate baseline characteristics in the comparator population.

Usage

```
target_pop_standard
```

Format

A data frame with 1 row and 6 variables:

N Total number of patients in the comparator population

Treatment Treatment received in the comparator population

AGE Mean age

SEX Proportion of males

SMOKE Proportion of patients who smoke

ECOG0 Proportion of patients with ECOG 0

wt_diagnostics *Weight diagnostics*

Description

Produce a set of useful diagnostic metrics to summarize propensity weights

- ESS (`estimate_ess`)
- Summary statistics of the weights: minimum, maximum, median, mean, SD (`summarize_wts`)
- Patient profile associated with weight values (`profile_wts`)

Usage

```
wt_diagnostics(data, wt_col = "wt", wt_rs = "wt_rs", vars)
```

Arguments

data	A data frame containing individual patient data from the intervention study, including a column containing the weights (derived using estimate_weights).
wt_col	The name of the weights column in the data frame containing the intervention individual patient data and the MAIC propensity weights. The default is wt.
wt_rs	The name of the rescaled weights column in the data frame containing the intervention individual patient data and the MAIC propensity weights. The default is wt_rs.
vars	A character vector giving the variable names of the baseline characteristics (not centered). These names must match the column names in the data.

Value

List of the following:

- The effective sample size (ESS) as a numeric value.
- A data frame that includes a summary (minimum, maximum, median, mean, standard deviation) of the weights and rescaled weights.
- A data frame that includes a summary of patient characteristics associated with each weight value.

See Also

estimate_weights, estimate_ess, summarize_wts, profile_wts

Examples

```
# This example code uses the weighted individual patient data, outputted from
# the estimate_weights function to perform weight diagnostics. The weighted data
# is saved within est_weights. To check the weighted aggregate baseline
# characteristics for 'intervention' match those in the comparator data,
# standardized data "target_pop_standard" is used. Please see the package
# vignette for more information on how to use the estimate_weights function and
# derive the "target_pop_standard" data.

library(dplyr)
library(MAIC)

# load est_weights
data(est_weights, package = "MAIC")

# load target_pop_standard
data(target_pop_standard, package = "MAIC")

# List out the uncentered variables used in the matching
match_cov <- c("AGE",
               "SEX",
               "SMOKE",
               "ECOG0")
```

```
# Are the weights sensible? ------------------------------------------------

# The wt_diagnostics function requires the output from the estimate_weights
# function and will output:
# - the effective sample size (ESS)
# - a summary of the weights and rescaled weights (mean, standard deviation,
#   median, minimum and maximum)
# - a unique set of weights with the corresponding patient profile based on the
#   matching variables

diagnostics <- wt_diagnostics(est_weights$analysis_data,
                              vars = match_cov)

diagnostics$ESS
diagnostics$Summary_of_weights
diagnostics$Weight_profiles

# Each of the wt_diagnostics outputs can also be estimated individually
ESS <- estimate_ess(est_weights$analysis_data)
weight_summ <- summarize_wts(est_weights$analysis_data)
wts_profile <- profile_wts(est_weights$analysis_data, vars = match_cov)

# Plot histograms of unscaled and rescaled weights
# bin_width needs to be adapted depending on the sample size in the data set
histogram <- hist_wts(est_weights$analysis_data, bin = 50)
histogram


# Has the optimization worked? ---------------------------------------------

# The following code produces a summary table of the intervention baseline
# characteristics before and after matching compared with the comparator
# baseline characteristics:

# Create an object to hold the output
baseline_summary <- list('Intervention' = NA,
                         'Intervention_weighted' = NA,
                         'Comparator' = NA)

# Summarise matching variables for weighted intervention data
baseline_summary$Intervention_weighted <- est_weights$analysis_data %>%
  transmute(AGE, SEX, SMOKE, ECOG0, wt) %>%
  summarise_at(match_cov, list(~ weighted.mean(., wt)))

# Summarise matching variables for unweighted intervention data
baseline_summary$Intervention <- est_weights$analysis_data %>%
  transmute(AGE, SEX, SMOKE, ECOG0, wt) %>%
  summarise_at(match_cov, list(~ mean(.)))

# baseline data for the comparator study
baseline_summary$Comparator <- transmute(target_pop_standard,
                                         AGE,
```

```
                                        SEX,
                                        SMOKE,
                                        ECOG0)

  # Combine the three summaries
  # Takes a list of data frames and binds these together
  trt <- names(baseline_summary)
  baseline_summary <-  bind_rows(baseline_summary) %>%
    transmute_all(sprintf, fmt = "%.2f") %>% #apply rounding for presentation
    transmute(ARM = as.character(trt), AGE, SEX, SMOKE, ECOG0)

  # Insert N of intervention  as number of patients
  baseline_summary$`N/ESS`[baseline_summary$ARM == "Intervention"] <- nrow(est_weights$analysis_data)

  # Insert N for comparator from target_pop_standard
  baseline_summary$`N/ESS`[baseline_summary$ARM == "Comparator"] <- target_pop_standard$N

  # Insert the ESS as the sample size for the weighted data
  # This is calculated above but can also be obtained using the estimate_ess function as shown below
  baseline_summary$`N/ESS`[baseline_summary$ARM == "Intervention_weighted"] <- est_weights$analysis_data %>%
    estimate_ess(wt_col = 'wt')

  baseline_summary <- baseline_summary %>%
    transmute(ARM, `N/ESS`=round(`N/ESS`,1), AGE, SEX, SMOKE, ECOG0)
```

Index