

MAIC package comparison

Setup

```
library(dplyr)
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.. :)

```
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"))

# Change to lower case
names(intervention_input) <- tolower(names(intervention_input))

# 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	4	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	4	1	102.4731	0	2304
## 5	5	A	69	1	0	1	2	0	101.6632	0	4761
## 6	6	A	48	0	0	1	2	0	237.0593	1	2304

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

Get Aggregate data. TODO: change csv to our standard format

```
# 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

```
#' @param intervention_input A data frame containing individual patient data from
#'   the intervention study.
#' @param target_pop A data frame containing aggregate dataset for the target population.
#'   Variables are followed by one of the following suffixes to denote the type of summary:
#'   varname_mean, varname_sd, varname_median, varname_prop.
#'   After preprocessing these summary suffixes, intervention_input is
```

```

#' centered using the aggregate data averages.

preprocess_data <- function(intervention_input, target_pop) {

  # Check intervention_data and target_pop are data frame
  # and match_cov is a character vector
  if (!is.data.frame(intervention_input)) {
    stop("intervention_input is expected to be a data frame")
  }
  if (!is.data.frame(target_pop)) {
    stop("target_pop is expected to be a data frame")
  }

  # Check if target_pop is 1 row of aggregate data
  if (nrow(target_pop) != 1) {
    stop("target_pop should have exactly 1 row")
  }

  # Strip off naming convention in the aggregate data
  varnames <- gsub("_([~_]+)$", "", names(target_pop))
  vartype <- gsub("^.*_", "", names(target_pop))

  # Preprocess standard deviation
  for (i in 1:dim(target_pop)[2]) {
    if (vartype[i] == "sd") {

      # retrieve which variable sd was specified
      varwithsd <- varnames[i]

      if (!paste0(varwithsd, "_mean") %in% names(target_pop)) {
        stop(paste0("Also need to provide mean for ",
                    varwithsd, " when specifying sd"))
      }

      # derive squared mean term
      target_pop[, paste0(varwithsd, "_squared_mean")] <- target_pop[,
        paste0(varwithsd, "_mean")]^2 + target_pop[,
        paste0(varwithsd, "_sd")]^2

      # remove standard deviation from the data frame
      target_pop <- target_pop[, -which(colnames(target_pop) ==
        paste0(varwithsd, "_sd"))]
    }
  }

  # Preprocess median
  for (i in 1:dim(target_pop)[2]) {
    if (vartype[i] == "median") {

      # retrieve which variable median was specified
      varwithmedian <- varnames[i]

      # make median into binary category

```

```

        intervention_input[, varwithmedian] <- ifelse(intervention_input[,
            varwithmedian] > target_pop[, paste0(varwithmedian,
                "_median")], 1, 0)
        target_pop[, paste0(varwithmedian, "_median")] <- 0.5
    }
}

# Remove everything that is not mean or prop from
# target_pop
if (!is.null(target_pop$N)) {
    N <- target_pop$N
}

vartype <- gsub("^.*_", "", names(target_pop))
target_pop <- target_pop[, which(vartype %in% c("mean", "median",
    "prop"))]

# intervention_input is centered using the aggregate
# data averages.
varnames <- gsub("_([~_]+)$", "", names(target_pop))
if (any(duplicated(varnames))) {
    stop("Cannot have more than 1 summary stat for each variable")
}
names(target_pop) <- varnames

intervention_data <- intervention_input
for (i in varnames) {
    intervention_data[, paste0(i, "_centered")] <- intervention_input[,
        i] - target_pop[, i]
}

# Add back in N
target_pop$N <- N

return(list(intervention_data = intervention_data, target_pop = target_pop))
}

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

```

Combined calculate weights function from Roche and MSD

```

#' Estimate MAIC propensity weights
#'
#' Estimate propensity weights for matching-adjusted indirect comparison (MAIC).
#'
#' @param intervention_data A data frame containing individual patient data from
#'   the intervention study. Intervention_data is assumed to have been preprocessed using
#'   preprocess_data (i.e. centered using aggregate data means)
#' @param match_cov A character vector giving the names of the covariates to

```

```

#' use in matching. These names must match the column names in intervention_data.
#' @param method The method used for optimisation - The default is method =
#' 'BFGS'. Refer to \link[stats]{optim} for options.
#' @param startVal a scalar, the starting value for all coefficients of the propensity score
#' regression
#' @param ... Additional arguments to be passed to optimisation functions such
#' as the method for maximum likelihood optimisation. Refer to \link[stats]{optim}
#' for options.
#' @return a list with 4 elements,
#' \describe{
#' \item wt - a numeric vector of unscaled individual weights.
#' \item wt.rs - a numerical vector of rescaled individual weights, with summation equaling to sample
#' \item ess - effective sample size, square of sum divided by sum of squares
#' \item opt - R object returned by \code{base::optim()}, for assess convergence and other details
#' }
#' @references NICE DSU TECHNICAL SUPPORT DOCUMENT 18: METHODS FOR
#' POPULATION-ADJUSTED INDIRECT COMPARISONS IN SUBMISSIONS TO NICE, REPORT BY
#' THE DECISION SUPPORT UNIT, December 2016
#' @seealso \link{optim}
#' @export

estimate_weights <- function(intervention_data, match_cov, startVal = 0,
  method = "BFGS", ...) {

  # Check intervention_data is a data frame and match_cov
  # is a character vector
  if (!is.data.frame(intervention_data)) {
    stop("intervention_data is expected to be a data frame")
  }
  if (!is.character(match_cov)) {
    stop("match_cov is expected to be a character vector")
  }

  # Check if there is any missingness in
  # intervention_data
  missing <- apply(intervention_data, 1, function(x) any(is.na(x)))
  if (any(missing)) {
    stop(paste0("Following rows have missing values: ", paste(which(missing),
      collapse = ",")))
  }

  for (i in match_cov) {
    # Check that match_vars is in one of the columns of
    # intervention_data
    if (!i %in% colnames(intervention_data)) {
      stop(paste0("Variable ", i, " is not one of intervention_data column names"))
    }

    # Check whether intervention_data has not been
    # centered by the aggregate data means by looking
    # at whether binary variables have only values of 0
    # and 1
    if (all(unique(intervention_data[, i]) == 2 & unique(intervention_data[,

```

```

        i])) %in% c(0, 1))) {
          stop("intervention_data does not seem to be centered by the aggregate data means")
        }
      }

      # Objective function
      objfn <- function(a1, X) {
        sum(exp(X %*% a1))
      }

      # Gradient function
      gradfn <- function(a1, X) {
        colSums(sweep(X, 1, exp(X %*% a1), "*"))
      }

      # Optimise Q(b) using Newton-Raphson techniques
      opt1 <- stats::optim(par = rep(startVal, dim(intervention_data[,
        match_cov])[2]), fn = objfn, gr = gradfn, X = as.matrix(intervention_data[,
        paste0(match_cov, "_centered"))], method = method, control = list(maxit = 300,
        trace = 2), ...)

      alpha <- opt1$par
      wt <- as.vector(exp(as.matrix(intervention_data[, paste0(match_cov,
        "_centered"))] %*% alpha))
      wt_rs <- (wt/sum(wt)) * nrow(intervention_data)

      output <- list(wt = wt, wt_rs = wt_rs, ess = sum(wt)^2/sum(wt^2),
        opt = opt1)
      return(output)
    }

    # Estimate weights
    weights <- estimate_weights(intervention_data = intervention_data,
      match_cov = match_cov)

```

```

## initial value 500.000000
## iter 10 value 175.469253
## iter 20 value 160.522650
## final value 160.513987
## converged

```

Summarize weights

```

summarize_wts <- function(weights) {

  with(weights, {
    summary <- data.frame(type = c("Weights", "Rescaled weights"),
      mean = c(mean(wt), mean(wt_rs)), sd = c(stats::sd(wt),
        stats::sd(wt_rs)), median = c(stats::median(wt),
        stats::median(wt_rs)), min = c(min(wt), min(wt_rs)),
      max = c(max(wt), max(wt_rs)))
  })
}

```

```

    return(summary)
  })
}

weight_summ <- summarize_wts(weights)
weight_summ

##           type      mean      sd      median      min      max
## 1      Weights 0.321028 0.5810205 0.01474677 5.045515e-13 4.117337
## 2 Rescaled weights 1.000000 1.8098751 0.04593609 1.571675e-12 12.825477

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 2.347332e+00 7.311925e+00
## 2  71        5041   1     0     0         1 2.003876e-08 6.242061e-08
## 3  58        3364   1     1     1         0 2.462215e-02 7.669784e-02
## 4  48        2304   0     0     1         1 2.018773e+00 6.288465e+00
## 5  69        4761   1     0     1         0 1.538100e-07 4.791171e-07
## 6  48        2304   0     0     1         0 6.225942e-01 1.939377e+00

```

Has optimization worked

```

check_weights <- function(intervention_data, weights, match_cov,
  target_pop) {

  ARM <- c("Intervention", "Intervention_weighted", "Comparator")
  ESS <- round(c(nrow(intervention_data), weights$ess, target_pop$N))

  weighted_cov <- intervention_data %>%
    summarise_at(match_cov, list(~weighted.mean(., weights$wt)))
  unweighted_cov <- intervention_data %>%
    summarise_at(match_cov, list(~mean(.)))
  comparator_cov <- select(target_pop, all_of(match_cov))

  cov <- rbind(unweighted_cov, weighted_cov, comparator_cov)
  baseline_summary <- cbind(ARM, ESS, cov)

  return(baseline_summary)
}

check_weights(intervention_data, weights, match_cov, target_pop)

##           ARM ESS      age age_squared      sex      smoke ecog0 n_pr_ther
## 1      Intervention 500 59.846    3662.578 0.384 0.3200000 0.406      0.26
## 2 Intervention_weighted 117 50.060    2516.437 0.490 0.1946309 0.350      0.50
## 3      Comparator 300 50.060    2516.436 0.490 0.1946309 0.350      0.50

```