Unanchored binary analysis

2024-09-30

Loading R packages

```
# install.packages("maicplus")
library(maicplus)
```

Additional R packages for this vignette:

```
library(dplyr)
```

Illustration using example data

This example reads in `centered_ipd_sat` data that was created in `calculating_weights` vignette and uses `adrs_sat` dataset to run binary outcome analysis using the `maic_unanchored` function by specifying `endpoint_type = "binary"`.

Note that parameters `ipd` and `pseudo_ipd` in the `maic_unanchored` function for binary data analysis needs to have the following columns: USUBJID, ARM, RESPONSE. USUBJID in `ipd` needs to match USUBJID in `weights_object`. `pseudo_ipd` for binary data can be conveniently generated using `get_pseudo_ipd_binary` function.

Robust standard error for the adjusted result are calculated by sandwich variance estimator in `sandwich` package with the function `vcovHC`. Default type of variance estimator (specified by parameter `binary_robust_cov_type`) is `HC3`, but other types can be specified. For more information on different types, see `vcovHC`.

```
data(centered_ipd_sat)
data(adrs_sat)

centered_colnames <- c("AGE", "AGE_SQUARED", "SEX_MALE", "ECOGO", "SMOKE", "N_PR_THER_MEDIAN")
centered_colnames <- paste0(centered_colnames, "_CENTERED")

weighted_data <- estimate_weights(
  data = centered_ipd_sat,
  centered_colnames = centered_colnames
)

# get dummy binary pseudo IPD
pseudo_adrs <- get_pseudo_ipd_binary(
  binary_agd = data.frame(
    ARM = "B",
    RESPONSE = c("YES", "NO"),
```

```
    COUNT = c(280, 120)
  ),
  format = "stacked"
)

result <- maic_unanchored(
  weights_object = weighted_data,
  ipd = adrs_sat,
  pseudo_ipd = pseudo_adrs,
  trt_ipd = "A",
  trt_agd = "B",
  normalize_weight = FALSE,
  endpoint_type = "binary",
  endpoint_name = "Binary Endpoint",
  eff_measure = "OR",
  # binary specific args
  binary_robust_cov_type = "HC3"
)
```

There are two summaries available in the result: descriptive and inferential. In the descriptive section, we have summaries of events.

```
result$descriptive
```

```
## $summary
##   trt_ind treatment            type   n   events events_pct
## 1       B         B Before matching 400 280.0000   70.00000
## 2       A         A Before matching 500 390.0000   78.00000
## 3       B         B  After matching 400 280.0000   70.00000
## 4       A         A  After matching 500 142.8968   28.57935
```

In the inferential section, we have the fitted models stored (i.e. logistic regression) and the results from the `glm` models (i.e. odds ratios and CI). If other effect measures are needed besides odds ratios, we have an option to fit risk ratios or risk differences via `eff_measure` parameter. Here is the overall summary.

```
result$inferential$summary
```

```
##          case       OR       LCL      UCL        pval
## 1          AB 1.519481 1.1247154 2.052805 0.006417064
## 2 adjusted_AB 1.083350 0.7268601 1.614683 0.694183560
```

Here are model and results before adjustment.

```
result$inferential$fit$model_before
```

```
##
## Call:  glm(formula = RESPONSE ~ ARM, family = glm_link, data = dat)
##
## Coefficients:
## (Intercept)         ARMA
##      0.8473       0.4184
```

```
##
## Degrees of Freedom: 899 Total (i.e. Null);  898 Residual
## Null Deviance:       1023
## Residual Deviance: 1016  AIC: 1020
```

```
result$inferential$fit$res_AB_unadj
```

```
## $est
## [1] 1.519481
##
## $se
## [1] 0.2373883
##
## $ci_l
## [1] 1.124715
##
## $ci_u
## [1] 2.052805
##
## $pval
## [1] 0.006417064
```

Here are model and results after adjustment.

```
result$inferential$fit$model_after
```

```
##
## Call:  glm(formula = RESPONSE ~ ARM, family = glm_link, data = dat,
##     weights = weights)
##
## Coefficients:
## (Intercept)         ARMA
##     0.84730      0.08006
##
## Degrees of Freedom: 899 Total (i.e. Null);  898 Residual
## Null Deviance:       726.7
## Residual Deviance: 726.5     AIC: 712.5
```

```
result$inferential$fit$res_AB
```

```
## $est
## [1] 1.08335
##
## $se
## [1] 0.2275624
##
## $ci_l
## [1] 0.7268601
##
## $ci_u
## [1] 1.614683
##
## $pval
## [1] 0.6941836
```

Using bootstrap to calculate standard errors

If bootstrap standard errors are preferred, we need to specify the number of bootstrap iteration (`n_boot_iteration`) in `estimate_weights` function and proceed fitting `maic_unanchored` function. Then, the outputs include bootstrapped CI. Different types of bootstrap CI can be found by using parameter `boot_ci_type`. See `boot.ci` in `boot` package for more details.

```r
weighted_data2 <- estimate_weights(
  data = centered_ipd_sat,
  centered_colnames = centered_colnames,
  n_boot_iteration = 100,
  set_seed_boot = 1234
)

result_boot <- maic_unanchored(
  weights_object = weighted_data2,
  ipd = adrs_sat,
  pseudo_ipd = pseudo_adrs,
  trt_ipd = "A",
  trt_agd = "B",
  normalize_weight = FALSE,
  endpoint_type = "binary",
  endpoint_name = "Binary Endpoint",
  eff_measure = "OR",
  boot_ci_type = "perc",
  # binary specific args
  binary_robust_cov_type = "HC3"
)

result_boot$inferential$fit$boot_res_AB
```

```
## $est
## [1] 1.08335
##
## $se
## [1] NA
##
## $ci_l
## [1] 0.846157
##
## $ci_u
## [1] 1.796863
##
## $pval
## [1] NA
```