Unanchored survival 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 `adtte_sat` and `pseudo_ipd_sat` outcome datasets to run survival analysis using the `maic_unanchored` function by specifying `endpoint_type = "tte"`.

Note that parameters `ipd` and `pseudo_ipd` in the `maic_unanchored` function for survival data analysis needs to have the following columns: USUBJID, ARM, EVENT, and TIME. USUBJID in `ipd` needs to match USUBJID in `weights_object`. USUBJID in `pseudo_ipd` is not strictly required, as if left unspecified, subject numbers are assigned using the row indexes.

Furthermore, TIME in both these datasets have to be in days. For instance, when we digitize a KM plot where time is recorded in months, we need to multiply the months by the appropriate factor (i.e. 30.4375) to get the time in days.

If you would like to run an analysis using scaled weights, set `normalize_weight` to TRUE. One clear benefit of using scaled weights is that the risk table at time = 0 in the Kaplan-Meier curve will match the IPD sample size.

```
data(centered_ipd_sat)
data(adtte_sat)
data(pseudo_ipd_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
)

result <- maic_unanchored(
```

```r
  weights_object = weighted_data,
  ipd = adtte_sat,
  pseudo_ipd = pseudo_ipd_sat,
  trt_ipd = "A",
  trt_agd = "B",
  normalize_weight = FALSE,
  endpoint_name = "Overall Survival",
  endpoint_type = "tte",
  eff_measure = "HR",
  time_scale = "month",
  km_conf_type = "log-log"
)
```

There are two summaries available in the result: descriptive and inferential. In the descriptive section, we have summaries from fitting `survfit` function. Note that restricted mean (rmean), median, and 95% CI are summarized in the `time_scale` specified.

```r
result$descriptive$summary
```

```
##   trt_ind treatment            type records    n.max  n.start    events
## 1       B         B Before matching     300 300.0000 300.0000 178.00000
## 2       A         A Before matching     500 500.0000 500.0000 190.00000
## 3       B         B  After matching     300 300.0000 300.0000 178.00000
## 4       A         A  After matching     500 199.4265 199.4265  65.68878
##       rmean se(rmean)    median  0.95LCL   0.95UCL
## 1  4.303551 0.3367260  2.746131 2.261125  3.320857
## 2  8.709690 0.3551477  7.587627 6.278691 10.288538
## 3  4.303551 0.3367260  2.746131 2.261125  3.320857
## 4 10.166029 0.5499915 11.900015 7.815275 14.873786
```

```r
# Not shown due to long output
# result$descriptive$survfit_before
# result$descriptive$survfit_after
```

In the inferential section, we have the fitted models stored (i.e. `survfit` and `coxph`) and the results from the `coxph` models (i.e. hazard ratios and CI). Note that the p-values summarized are from `coxph` model Wald test and not from a log-rank test. Here is the overall summary.

```r
result$inferential$summary
```

```
##          case        HR       LCL       UCL         pval
## 1          AB 0.3748981 0.3039010 0.4624815 5.245204e-20
## 2 adjusted_AB 0.2834780 0.2074664 0.3873387 2.473442e-15
```

Here are models and results before adjustment.

```r
result$inferential$fit$km_before
```

```
## Call: survfit(formula = Surv(TIME, EVENT) ~ ARM, data = dat, conf.type = km_conf_type)
##
##          n events median 0.95LCL 0.95UCL
## ARM=B 300    178   83.6    68.8     101
## ARM=A 500    190  230.9   191.1     313
```

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

```
## Call:
## coxph(formula = Surv(TIME, EVENT) ~ ARM, data = dat)
##
##         coef exp(coef) se(coef)      z      p
## ARMA -0.9811    0.3749   0.1071 -9.159 <2e-16
##
## Likelihood ratio test=80.62  on 1 df, p=< 2.2e-16
## n= 800, number of events= 368
```

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

```
## $est
## [1] 0.3748981
##
## $se
## [1] 0.0405065
##
## $ci_l
## [1] 0.303901
##
## $ci_u
## [1] 0.4624815
##
## $pval
## [1] 5.245204e-20
```

Here are models and results after adjustment.

```r
result$inferential$fit$km_after
```

```
## Call: survfit(formula = Surv(TIME, EVENT) ~ ARM, data = dat, weights = dat$weights,
##     conf.type = km_conf_type)
##
##       records   n events median 0.95LCL 0.95UCL
## ARM=B     300 300  178.0   83.6    68.8     101
## ARM=A     500 199   65.7  362.2   237.9     453
```

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

```
## Call:
## coxph(formula = Surv(TIME, EVENT) ~ ARM, data = dat, weights = weights,
##     robust = TRUE)
##
##         coef exp(coef) se(coef) robust se      z        p
## ARMA -1.2606    0.2835   0.1504    0.1593 -7.915 2.47e-15
##
## Likelihood ratio test=80.4  on 1 df, p=< 2.2e-16
## n= 800, number of events= 368
```

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

```
## $est
## [1] 0.283478
##
## $se
## [1] 0.04601759
##
## $ci_l
## [1] 0.2074664
##
## $ci_u
## [1] 0.3873387
##
## $pval
## [1] 2.473442e-15
```

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 specifying the 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 = adtte_sat,
  pseudo_ipd = pseudo_ipd_sat,
  trt_ipd = "A",
  trt_agd = "B",
  normalize_weight = FALSE,
  endpoint_name = "Overall Survival",
  endpoint_type = "tte",
  eff_measure = "HR",
  boot_ci_type = "perc",
  time_scale = "month",
  km_conf_type = "log-log"
)

result_boot$inferential$fit$boot_res_AB
```

```
## $est
## [1] 0.283478
##
```

```
## $se
## [1] NA
##
## $ci_l
## [1] 0.2144978
##
## $ci_u
## [1] 0.3740624
##
## $pval
## [1] NA
```