Consistency of maxSPRT critical value calculations across packages in R

Signal detection of spontaneous medical device reports over time accounting for multiple comparisons

Ty Stanford et al.

Table of contents

1 Set up

1.1 Packages

```
library("dplyr")
```

```
Warning: package 'dplyr' was built under R version 4.2.3
```

```
Attaching package: 'dplyr'
```

```
The following objects are masked from 'package:stats':

    filter, lag
```

```
The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library("tidyr")
library("tictoc")
library("foreach")
library("EmpiricalCalibration")
```

```
Warning: package 'EmpiricalCalibration' was built under R version 4.2.3
```

```
library("Sequential")
library("arrow")
```

```
Attaching package: 'arrow'
```

```
The following object is masked from 'package:utils':

    timestamp
```

```r
library("ggplot2")
```

1.2 Set constants

```r
min_event <- 1
alpha <- 0.05

cv_df_0 <-
  expand_grid(
    cntl_to_case_ratio = seq(0.5, 5, by = 0.5),
    max_n = 200,
    look_interval = c(1:5, 8, 10)
  )

### testing
# cv_df_0 <-
#   expand_grid(
#     cntl_to_case_ratio = c(0.5, 2, 5),
#     max_n = 200,
#     look_interval = c(5, 10)
#   )
```

2 Example critical value calcultions

2.1 Example calcs in two packages

```
look_i <- 10
max_n_i <- 200
z_i <- 2


# has to be per period (not cumulative) for Sequential::CV.Binomial()
# (and EmpiricalCalibration::computeCvBinomial())
# i.e. test performed at 3 events then when 4 more events come in requires
# GroupSizes = c(3, 4)
### NOT: GroupSizes = c(3, 7)
gs_seq <- rep(look_i, floor(max_n_i / look_i))
if (sum(gs_seq) != max_n_i) { # if doesn't go to max_n, add at end for last look
  gs_seq <- c(gs_seq, max_n_i - sum(gs_seq))
}
gs_seq
```

```
 [1] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
```

```
# both of the below take ~ 6 sec
tic()
Sequential::CV.Binomial(
  N = max_n_i,
  alpha = alpha,
  M = min_event,
  z = z_i,
  GroupSizes = gs_seq
)
```

```
$cv
[1] 2.87369

$Type_I_Error
[1] 0.04895415
```

```
toc()
```

```
6.22 sec elapsed
```

```
tic()
EmpiricalCalibration::computeCvBinomial(
  groupSizes = gs_seq,
  z = z_i,
  minimumEvents = 1,
  alpha = 0.05, # does two-tailed by default? (no)
  sampleSize = 1e+06
)
```

```
Selected alpha: 0.049 (least conservative value below 0.05)
```

```
[1] 2.873683
attr(,"alpha")
[1] 0.049107
```

```
toc()
```

```
5.28 sec elapsed
```

2.2 Critical value calcs for large n

```
look_i <- 10
max_n_i <- 10000
z_i <- 2


# has to be per period (not cumulative) for Sequential::CV.Binomial()
# (and EmpiricalCalibration::computeCvBinomial())
# i.e. test performed at 3 events then when 4 more events come in requires
# GroupSizes = c(3, 4)
### NOT: GroupSizes = c(3, 7)
gs_seq <- rep(look_i, floor(max_n_i / look_i))
```

```r
if (sum(gs_seq) != max_n_i) { # if doesn't go to max_n, add at end for last look
  gs_seq <- c(gs_seq, max_n_i - sum(gs_seq))
}
gs_seq

# takes ~ 5 min but Sequential::CV.Binomial()
# warns against using for sum(groupSizes) > 500
tic()
EmpiricalCalibration::computeCvBinomial(
  groupSizes = gs_seq,
  z = z_i,
  minimumEvents = 1,
  alpha = 0.05, # does two-tailed by default?
  sampleSize = 1e+06
)
toc()
```

3 Compare package critical value calcultions

3.1 Range of calcs in `Sequential` package

```
### testing Sequential::CV.Binomial()

cv_df_seq <- cv_df_0
cv_df_seq$cv <- 0
cv_df_seq$pack <- "Sequential"
cv_df_seq



tic() # takes 5 min
for (i in 1:nrow(cv_df_seq)) {

  look_i <- cv_df_seq$look_interval[i]
  max_n_i <- cv_df_seq$max_n[i]
  z_i <- cv_df_seq$cntl_to_case_ratio[i]

  # has to be per period (not cumulative) for Sequential::CV.Binomial()
  # i.e. test performed at 3 events then when 3 more events come in requires
  # GroupSizes = c(3, 3)
  gs_seq <- rep(look_i, floor(max_n_i / look_i))
  if (sum(gs_seq) != max_n_i) { # if doesn't go to max_n, add at end for last look
    gs_seq <- c(gs_seq, max_n_i - sum(gs_seq))
  }

  cv_df_seq$cv[i] <-
    Sequential::CV.Binomial(
      N = max_n_i,
      alpha = alpha,
      M = min_event,
      z = z_i,
      GroupSizes = gs_seq
    )$cv

}
toc()

cv_df_seq
```

```r
cv_df_seq %>%
  write_parquet(., sink = "out/cv_df_seq.parquet")
```

3.2 Range of calcs in `EmpiricalCalibration` package

```r
### testing EmpiricalCalibration::computeCvBinomial()

cv_df_ec <- cv_df_0
cv_df_ec$cv <- 0
cv_df_ec$pack <- "EmpiricalCalibration"
cv_df_ec


tic() # takes 4 hours using sampleSize = 1e+07
for (i in 1:nrow(cv_df_ec)) {

  look_i <- cv_df_ec$look_interval[i]
  max_n_i <- cv_df_ec$max_n[i]
  z_i <- cv_df_ec$cntl_to_case_ratio[i]

  # has to be per period (not cumulative) for EmpiricalCalibration::computeCvBinomial()
  # i.e. test performed at 3 events then when 3 more events come in requires
  # GroupSizes = c(3, 3)
  gs_seq <- rep(look_i, floor(max_n_i / look_i))
  if (sum(gs_seq) != max_n_i) { # if doesn't go to max_n, add at end for last look
    gs_seq <- c(gs_seq, max_n_i - sum(gs_seq))
  }

  cv_df_ec$cv[i] <-
    EmpiricalCalibration::computeCvBinomial(
      groupSizes = gs_seq,
      z = z_i,
      minimumEvents = min_event,
      alpha = alpha,
      sampleSize = 1e+06
    )

}
toc()

cv_df_ec
```

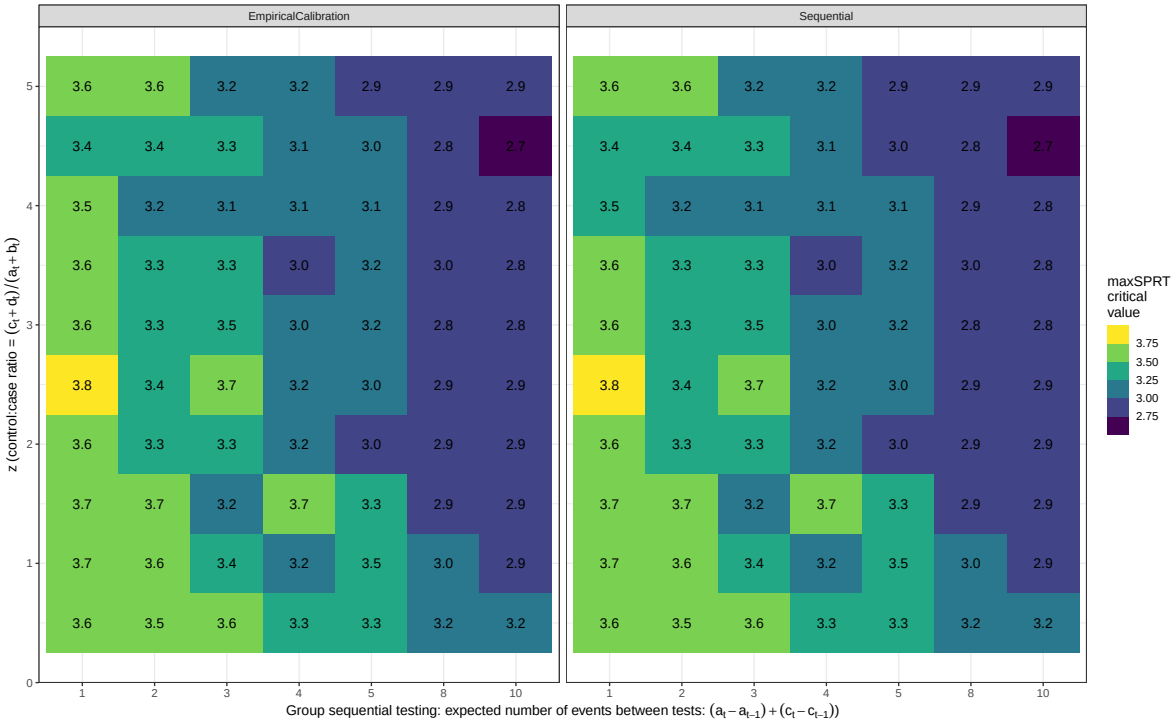
```r
cv_df_ec %>%
  write_parquet(., sink = "out/cv_df_ec.parquet")
```

3.3 Checking equivalence of package results

```
cv_df <-
  bind_rows(
    read_parquet("out/cv_df_ec.parquet"),
    read_parquet("out/cv_df_seq.parquet")
  )



cv_df %>%
  mutate(
    cv_txt = sprintf("%1.1f", cv)
  ) %>%
  ggplot(., aes(x = factor(look_interval), y = cntl_to_case_ratio)) +
  geom_tile(aes(fill = cv)) +
  geom_text(aes(label = cv_txt)) +
  facet_wrap(~ pack) +
  scale_fill_viridis_b() +
  labs(
    x = expression(paste(
      "Group sequential testing: expected number of events between tests: ",
      (a[t] - a[t-1]) + (c[t] - c[t-1]),
      ")"
    )),
    y = expression(paste(
      "z (control:case ratio = ",
      (c[t] + d[t]) / (a[t] + b[t])
    )),
    fill = "maxSPRT\ncritical\nvalue",
    label = "maxSPRT\ncritical\nvalue",
  ) +
  theme_bw()
```




```r
cv_df_w <-
  cv_df %>%
  pivot_wider(
    .,
    id_cols = all_of(c("cntl_to_case_ratio", "max_n", "look_interval")),
    names_from = "pack",
    values_from = "cv"
  ) %>%
  mutate(
    cv_diff = EmpiricalCalibration - Sequential,
    rel_cv_diff = cv_diff / ((EmpiricalCalibration + Sequential) / 2)
  )


cv_df_w %>%
  select(
    `max n` = max_n,
    `z=E[cntl:case]` = cntl_to_case_ratio,
    `events per look` = look_interval,
    empcalib = EmpiricalCalibration,
```

```
    seq = Sequential,
    diff = cv_diff,
    `std diff` = rel_cv_diff
  ) %>%
  knitr::kable(., digits = 2)
```

max n	z=E[cntl:case]	events per look	empcalib	seq	diff	std diff
200	0.5	1	3.65	3.65	0.00	0.00
200	0.5	2	3.53	3.53	0.00	0.00
200	0.5	3	3.65	3.65	0.00	0.00
200	0.5	4	3.31	3.31	0.00	0.00
200	0.5	5	3.28	3.28	0.00	0.00
200	0.5	8	3.24	3.24	0.00	0.00
200	0.5	10	3.16	3.16	0.00	0.00
200	1.0	1	3.68	3.68	0.00	0.00
200	1.0	2	3.62	3.62	-0.01	0.00
200	1.0	3	3.40	3.40	0.00	0.00
200	1.0	4	3.16	3.16	0.00	0.00
200	1.0	5	3.47	3.47	0.00	0.00
200	1.0	8	3.05	3.05	0.00	0.00
200	1.0	10	2.90	2.90	0.01	0.00
200	1.5	1	3.67	3.67	0.00	0.00
200	1.5	2	3.67	3.67	0.00	0.00
200	1.5	3	3.21	3.21	0.00	0.00
200	1.5	4	3.67	3.67	0.00	0.00
200	1.5	5	3.31	3.31	0.00	0.00
200	1.5	8	2.94	2.94	0.00	0.00
200	1.5	10	2.91	2.90	0.00	0.00
200	2.0	1	3.59	3.59	0.00	0.00
200	2.0	2	3.34	3.33	0.01	0.00
200	2.0	3	3.30	3.30	0.00	0.00
200	2.0	4	3.24	3.24	0.00	0.00
200	2.0	5	2.98	2.97	0.01	0.00
200	2.0	8	2.90	2.90	0.00	0.00
200	2.0	10	2.87	2.87	0.00	0.00
200	2.5	1	3.76	3.76	0.00	0.00
200	2.5	2	3.44	3.44	0.00	0.00
200	2.5	3	3.73	3.73	0.00	0.00
200	2.5	4	3.23	3.23	0.00	0.00
200	2.5	5	3.05	3.05	0.00	0.00
200	2.5	8	2.93	2.93	0.00	0.00

max n	z=E[cntl:case]	events per look	empcalib	seq	diff	std diff
200	2.5	10	2.94	2.94	0.00	0.00
200	3.0	1	3.64	3.64	0.00	0.00
200	3.0	2	3.26	3.26	0.00	0.00
200	3.0	3	3.45	3.45	0.00	0.00
200	3.0	4	3.00	3.00	0.00	0.00
200	3.0	5	3.21	3.21	0.00	0.00
200	3.0	8	2.79	2.79	0.00	0.00
200	3.0	10	2.83	2.83	0.00	0.00
200	3.5	1	3.64	3.63	0.01	0.00
200	3.5	2	3.26	3.27	-0.01	0.00
200	3.5	3	3.32	3.32	0.00	0.00
200	3.5	4	2.98	2.98	0.00	0.00
200	3.5	5	3.21	3.21	0.00	0.00
200	3.5	8	2.98	2.98	0.00	0.00
200	3.5	10	2.78	2.78	0.00	0.00
200	4.0	1	3.51	3.48	0.03	0.01
200	4.0	2	3.22	3.22	0.00	0.00
200	4.0	3	3.13	3.13	0.00	0.00
200	4.0	4	3.08	3.08	0.00	0.00
200	4.0	5	3.10	3.10	0.00	0.00
200	4.0	8	2.92	2.92	0.00	0.00
200	4.0	10	2.82	2.82	0.00	0.00
200	4.5	1	3.41	3.41	0.00	0.00
200	4.5	2	3.41	3.41	0.00	0.00
200	4.5	3	3.28	3.28	0.00	0.00
200	4.5	4	3.11	3.11	0.00	0.00
200	4.5	5	3.00	2.97	0.03	0.01
200	4.5	8	2.77	2.77	0.00	0.00
200	4.5	10	2.74	2.74	0.00	0.00
200	5.0	1	3.58	3.58	0.00	0.00
200	5.0	2	3.58	3.58	0.00	0.00
200	5.0	3	3.25	3.25	0.00	0.00
200	5.0	4	3.22	3.22	0.00	0.00
200	5.0	5	2.94	2.94	0.00	0.00
200	5.0	8	2.94	2.95	-0.01	0.00
200	5.0	10	2.94	2.94	0.00	0.00

```
# cv_df_w %>%
#   mutate(
```

```
#     rel_cv_diff_txt = sprintf("%0.3f", rel_cv_diff)
#   ) %>%
#   ggplot(., aes(x = factor(look_interval), y = cntl_to_case_ratio)) +
#   geom_tile(aes(fill = rel_cv_diff)) +
#   geom_text(aes(label = rel_cv_diff_txt)) +
#   scale_fill_viridis_b(option = "B") +
#   theme_bw()
```

4 Session information

```
format(Sys.time(), '%d %b %Y')
```

```
[1] "17 Jul 2023"
```

```
Sys.info() %>% as.data.frame(.)
```

```
                              .
sysname                 Windows
release                  10 x64
version              build 19044
nodename         DESKTOP-R5P5N23
machine                  x86-64
login                        ty
user                         ty
effective_user               ty
```

```
sessionInfo()
```

```
R version 4.2.2 (2022-10-31 ucrt)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 19044)

Matrix products: default

locale:
[1] LC_COLLATE=English_Australia.utf8  LC_CTYPE=English_Australia.utf8
[3] LC_MONETARY=English_Australia.utf8 LC_NUMERIC=C
[5] LC_TIME=English_Australia.utf8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] ggplot2_3.4.1              arrow_11.0.0.2
[3] Sequential_4.3             EmpiricalCalibration_3.1.1
```

```
[5] foreach_1.5.2              tictoc_1.1
[7] tidyr_1.3.0                dplyr_1.1.2

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.10        pillar_1.9.0       compiler_4.2.2     iterators_1.0.14
 [5] tools_4.2.2        boot_1.3-28        digest_0.6.31      bit_4.0.5
 [9] viridisLite_0.4.1  gtable_0.3.1       jsonlite_1.8.4     evaluate_0.20
[13] lifecycle_1.0.3    tibble_3.2.1       pkgconfig_2.0.3    rlang_1.1.1
[17] cli_3.6.0          rstudioapi_0.14    yaml_2.3.7         xfun_0.37
[21] fastmap_1.1.0      withr_2.5.0        knitr_1.42         generics_0.1.3
[25] vctrs_0.6.3        grid_4.2.2         bit64_4.0.5        tidyselect_1.2.0
[29] glue_1.6.2         R6_2.5.1           fansi_1.0.4        rmarkdown_2.20
[33] farver_2.1.1       purrr_1.0.1        tzdb_0.3.0         magrittr_2.0.3
[37] scales_1.2.1       codetools_0.2-18   htmltools_0.5.4    assertthat_0.2.1
[41] colorspace_2.1-0   labeling_0.4.2     utf8_1.2.3         munsell_0.5.0
```