Simulation Example

Daniel Gomon, Hein Putter, Rob Nelissen, Stéphanie van der Pas

2022-08-22

Pre-requisites

```
#Determine Risk-adjustment Cox and GLM model on full data set:
exprfit <- as.formula("Surv(survtime, censorid) ~ age + sex + BMI")
#50 day after surgery followup for Bernoulli CUSUM
exprfitglm <- as.formula("(survtime <= 50) & (censorid == 1)~ age + sex + BMI")
coxmod <- coxph(exprfit, data= surgerydat)
glmmod <- glm(exprfitglm, data = surgerydat, family = binomial(link = "logit"))
```

Type I error restriction

Simulation parameters

```
#Number of in-control samples - increase for more accuracy (Article: 500)
n_sim_ic <- 30
#Number of out-of-control samples - increase for better validation (Article: 500)
n_sim_oc <- 30
#Study duration (in days)
time <- 365
#arrival rate (per day)
psi <- 1
#Expected hazard rate (theta) for Bernoulli and BK-CUSUM
theta <- log(2)
#Follow-up period (only for Bernoulli CUSUM)
followup <- 50
#Required Type I error when determining control limits
alpha <- 0.05
```

Control limits

Find control limits restricting type I error to 0.05 over 1 year for BK-, CGR- and Bernoulli CUSUM.

```
bk_control <- bk_control_limit(time = time, alpha = alpha, psi = psi,
                               n_sim = n_sim_ic, theta = theta, coxphmod = coxmod,
                               baseline_data = surgerydat)

cgr_control <- cgr_control_limit(time = time, alpha = alpha, psi = psi,
                                 n_sim = n_sim_ic, coxphmod = coxmod,
                                 baseline_data = surgerydat)

bernoulli_control <- bernoulli_control_limit(time = time, alpha = alpha,
```

```
                                                  psi = psi, n_sim = n_sim_ic,
                                                  followup = followup, theta = theta,
                                                  glmmod = glmmod,
                                                  baseline_data = surgerydat)

print(c(bk_control$h, cgr_control$h, bernoulli_control$h))
```

```
## [1] 7.63 8.39 4.10
```

Data generation

Generate out-of-control units with $\mu = log(2)$ (twice the failure rate). This means that BK- and Bernoulli CUSUM have perfect parameters!

```
newdat <- generate_units(time = 2*time, psi = psi, n_sim = n_sim_oc,
                         coxphmod = coxmod, baseline_data = surgerydat,
                         mu = theta)
```

Control chart determination

Determine control chart for out-of-control data

```
charts <- vector(mode = "list", length = n_sim_oc)
for (i in 1:n_sim_oc){
  tdat <- subset(newdat, unit == i)
  charts[[i]]$bernoulli <- bernoulli_cusum(data = tdat, followup = followup,
                                           glmmod = glmmod, theta = theta,
                                           h = bernoulli_control$h)
  charts[[i]]$bk <- bk_cusum(data = tdat, theta = theta, coxphmod = coxmod,
                             h = bk_control$h)
  charts[[i]]$cgr <- cgr_cusum(data = tdat, coxphmod = coxmod,
                               h = cgr_control$h)
}
```

Power over time

Determine power over time for each of the charts:

```
powerber <- sapply(charts, FUN = function(x){
  runlength(x$bernoulli, h = bernoulli_control$h)})
powercgr <- sapply(charts, FUN = function(x){
  runlength(x$cgr, h = cgr_control$h)})
powerbk <- sapply(charts, FUN = function(x){
  runlength(x$bk, h = bk_control$h)})
```

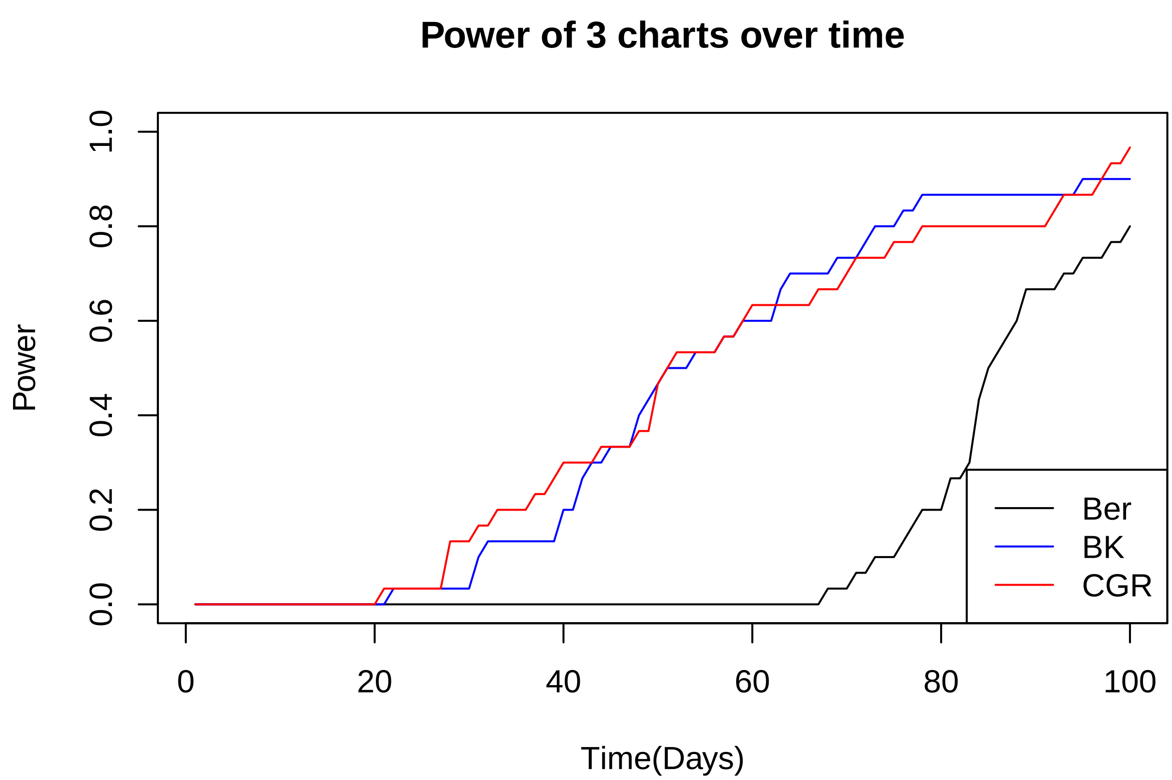
Determine the power of the charts in the first 100 days after the start of the study

```
power <- matrix(data = NA, nrow = 3, ncol = 100)
for(i in 1:ncol(power)){
  power[1,i] <- sum(powerber <= i)/n_sim_oc
  power[2,i] <- sum(powerbk <= i)/n_sim_oc
  power[3,i] <- sum(powercgr <= i)/n_sim_oc
}
rownames(power) <- c("Ber", "BK", "CGR")
```

Visualisation

Make a simple plot displaying power over time for all 3 charts. In this case, the BK-CUSUM does a bit better than the CGR-CUSUM, but with a control limit determined on a sample size of 30 and sample size of 30 out-of-control hospitals it's hard to draw conclusions. Increase `n_sim_ic` and `n_sim_oc` to get more reliable estimates! Warning: increases computation time considerably.

```
plot(1:100, power[1,], col = "black", xlab = "Time(Days)", ylab = "Power",
     main = "Power of 3 charts over time", type = "l", ylim = c(0,1))
lines(1:100, power[2,], col = "blue")
lines(1:100, power[3,], col = "red")
legend("bottomright", legend=c("Ber", "BK", "CGR"),
       col=c("black", "blue", "red"), lty = 1)
```



ARL restriction

Parameters chosen such that the in-control Average Run Length (ARL) is approximately 180 days on a sample of 30 hospitals. Then we determine the out-of-control ARL on a sample of 30 hospitals and compare the ARL of the 3 charts.

Simulation parameters

```
set.seed(01041996)
#Number of in-control samples - increase for more accuracy (Article: 500)
n_sim_ic <- 30
#Number of out-of-control samples - increase for better validation (Article: 500)
```

```r
n_sim_oc <- 30
#arrival rate (per day)
psi <- 2500/1095
#Expected hazard rate (theta) for Bernoulli and BK-CUSUM
theta <- log(2)
#Follow-up period (only for Bernoulli CUSUM)
followup <- 50
#Required Type I error when determining control limits
alpha <- 0.05


#Desired in-control average run length (under the null hypothesis)
ARL_0 <- 180
#t_stoptime is the stopping time for calculating charts to reduce computation time.
#In this case, 500 suits our needs
#The value you should choose depends on the failure rate and desired ARL_0
t_stoptime <- 500
#Study time in which patients arrivals happen (must be \geq t_stoptime)
time <- 1000
```

For the results in the article: change `t_stoptime` to approx $17 \cdot 365$ to restrict ARL to 15 years $(15 \cdot 365)$ We want `t_stoptime` to be as small as possible (reduce computation time), while at the same time we want each in-control chart to hit the control limit. Strategy: determine a few in-control charts until $17 \cdot 365$ and memorize their values. Then you can stop all charts around those values by specifying h = mean(values) + margin (about 0.5). This will greatly reduce computation time. This works, because run length of the charts is a non-decreasing function of the control limit.

Generating in-control data to determine control limits

We generate in-control data, then determine the control limits to use to get the desired ARL under the null.

Generate `n_sim_ic = 30` in-control hospitals with arrivals until time $= 1000$ and approx $\psi = 2500/1095 = 2.28$ arrivals per day with exponential $(\lambda = 0.002)$ failure rate.

```r
ic_hospitals <- generate_units(time = time, psi = psi, n_sim = n_sim_ic,
                               inv_cbaseh = function(t) inv_chaz_exp(t, lambda = 0.002))
```

Determine in-control charts

```r
ic_charts <- vector(mode = "list", length = n_sim_ic)
for (i in 1:n_sim_ic){
  tdat <- subset(ic_hospitals, unit == i)
  ic_charts[[i]]$bk <- bk_cusum(data = tdat, theta = theta, cbaseh = function(t) chaz_exp(t, lambda = 0
                               stoptime = t_stoptime)
  ic_charts[[i]]$cgr <- cgr_cusum(data = tdat, cbaseh = function(t) chaz_exp(t, lambda = 0.002),
                                 stoptime = t_stoptime)
}
```

Determine average run length for different control limits

```r
hosps <- unique(ic_hospitals$unit)
hseq <- seq(2, 20, 0.01)
ARL_bk <- vector(mode = "numeric", length = length(hseq))
ARL_cgr <- vector(mode = "numeric", length = length(hseq))
for(i in seq_along(hseq)){
  ARL_bk[i] <- mean(sapply(ic_charts, FUN = function(x) runlength(x$bk, h = hseq[i])))
```

```
  ARL_cgr[i] <- mean(sapply(ic_charts, FUN = function(x) runlength(x$cgr, h = hseq[i])))
}
```

Determine control limits to use for BK and CGR-CUSUM

```
id_bk <- which.min(abs(ARL_bk - ARL_0))
id_cgr <- which.min(abs(ARL_cgr - ARL_0))
h_bk <- hseq[id_bk]
#3.22
h_cgr <- hseq[id_cgr]
#4.5
print(c(h_bk, h_cgr))
```

```
## [1] 3.08 2.65
```

Generate out-of-control data and determine out-of-control ARL

Generate out-of-control charts with $e^{\theta} = 2$. This procedure can be repeated for different value of mu in correspondence to the article.

```
oc_hospitals_log2 <- generate_units(time = 500, psi = 2500/1095, n_sim = 100,
                              inv_cbaseh = function(t) inv_chaz_exp(t, lambda = 0.002),
                              mu = log(2))
```

Determine charts on out-of-control hospitals.

```
oc_charts_log2 <- vector(mode = "list", length = 100)
for (i in 1:100){
  tdat <- subset(oc_hospitals_log2, unit == i)
  oc_charts_log2[[i]]$bk <- bk_cusum(data = tdat, theta = log(2), cbaseh = function(t) chaz_exp(t, lamb
                              stoptime = t_stoptime, h = h_bk)
  oc_charts_log2[[i]]$cgr <- cgr_cusum(data = tdat, cbaseh = function(t) chaz_exp(t, lambda = 0.002),
                                stoptime = t_stoptime, h = h_cgr)
}

ARL_log2 <- c(mean(sapply(oc_charts_log2, FUN = function(x) runlength(x$bk, h = h_bk))),
              mean(sapply(oc_charts_log2, FUN = function(x) runlength(x$cgr, h = h_cgr))))
names(ARL_log2) <- c("bk", "cgr")
```

```
ARL_log2
```

```
##       bk      cgr
## 50.13199 34.04363
```

So CGR has smaller out of control ARL with same in control ARL, while the parameter for the BK-CUSUM was chosen perfectly. Note that in this case, the success package automatically takes `maxtheta = log(6)` in `cgr_cusum()`. To re-do the simulation as in the article set `maxtheta = Inf` in `cgr_cusum()` runs.

```
sessionInfo()
```

```
## R version 4.2.0 (2022-04-22 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19044)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_Netherlands.utf8  LC_CTYPE=English_Netherlands.utf8
```

```
## [3] LC_MONETARY=English_Netherlands.utf8 LC_NUMERIC=C
## [5] LC_TIME=English_Netherlands.utf8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] survival_3.3-1     success_0.1.2      Rfast_2.0.6        RcppZiggurat_0.1.6
## [5] Rcpp_1.0.8.3       RColorBrewer_1.1-3 plotly_4.10.0      pbapply_1.5-0
## [9] ggplot2_3.3.6
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.1.2  xfun_0.31         purrr_0.3.4       splines_4.2.0
##  [5] lattice_0.20-45   colorspace_2.0-3  vctrs_0.4.1       generics_0.1.2
##  [9] htmltools_0.5.2   viridisLite_0.4.0 yaml_2.3.5        utf8_1.2.2
## [13] rlang_1.0.2       pillar_1.7.0      glue_1.6.2        withr_2.5.0
## [17] lifecycle_1.0.1   stringr_1.4.0     munsell_0.5.0     gtable_0.3.0
## [21] htmlwidgets_1.5.4 evaluate_0.15     knitr_1.39        fastmap_1.1.0
## [25] parallel_4.2.0    fansi_1.0.3       highr_0.9         scales_1.2.0
## [29] jsonlite_1.8.0    digest_0.6.29     stringi_1.7.6     dplyr_1.0.9
## [33] grid_4.2.0        cli_3.3.0         tools_4.2.0       magrittr_2.0.3
## [37] lazyeval_0.2.2    tibble_3.1.7      crayon_1.5.1      tidyr_1.2.0
## [41] pkgconfig_2.0.3   ellipsis_0.3.2    Matrix_1.4-1      data.table_1.14.2
## [45] rmarkdown_2.14    httr_1.4.3        rstudioapi_0.13   R6_2.5.1
## [49] compiler_4.2.0
```