Appendix

censoringCode.Rmd

10 Dec 2020

Initialization

```
rm(list = ls())
library("knitr")
library("survival")
library("OptimalCutpoints")
library("xtable")
```

Survival functions

This function takes 2 quantile times and the corresponding quantile proportions and returns the parameters of a Weibull distribution

@param Q_A A positive number

@param Q_B A positive number

@param p_A A probability $p_A \in (0,1)$ corresponding to the quantile indicated by the time Q_A

@param p_B A probability $p_B \in (0,1)$ corresponding to the quantile indicated by the time Q_B

@return a list The parameters *shape* and *scale* of a Weibull distribution

```
quantiles2weibull <- function(Q_A, Q_B, p_A = 0.5, p_B = 0.99) {
   k = log(log(1 - p_B) / log(1- p_A)) / log(Q_B / Q_A)
   b = Q_A / (-log(1- p_A))^(1/k)
   list(shape = k, scale = b)
}
```

This function takes 2 quantile times and the corresponding quantile proportions and returns a function that transform a vector of probabilities into a vector of times following the corresponding Weibull distribution

@param Q_A A positive number

@param Q_B A positive number

@param p_A A probability $p_A \in (0,1)$ corresponding to the quantile indicated by the time Q_A

@param p_B A probability $p_B \in (0,1)$ corresponding to the quantile indicated by the time Q_B

@param p A vector of probabilities

@return a vector The vector of times

```
inverseWeibull <- function(Q_A, Q_B, p_A = 0.5, p_B = 0.99) {
   function(p = runif(1)) {
      qWeibull <- quantiles2weibull(Q_A, Q_B, p_A, p_B)
      with(qWeibull, sapply(p, function(p) scale * (- log(p))^(1/shape)))
   }
}
```

This function accepts a number of cases and the parameters of a mixed model (median, Q_{99}, and cure-rate), estimates the cured cases and the inverse Weibull event times, and returns a virtual dataset with complete follow-up to be used by the survival package.

@param nCases The number of cases in the virtual dataset

@param Q_A A positive number, the median event time

@param Q_B A positive number, the Q_{99} event time

@param pCure The cure rate $pCure \in (0, 1)$ of the mixed model

@param x The label for the dataset

@return the virtual dataset. A data.frame with columns c(time, status, x)

```
completeFA <- function(nCases = 10, Q_A = 25, Q_B = 100, pCure = 0, x = "Uncensored") {
   nEvents <- as.integer(nCases * (1 - pCure))
   time = sort(inverseWeibull(Q_A * (1 - pCure), Q_B)(runif(nEvents)))
   tmax = max(time)
   time = c(time, rep(tmax, nCases - nEvents))
   status <- c(rep(1, nEvents), rep(0, nCases - nEvents))
   data.frame(time, status, x = x, stringsAsFactors = FALSE)
}
```

Censoring functions

This function takes a complete follow-up dataset and the parameters of a mixed model (events median and Q_{99}, and proportion of censoring), estimates a vector of random probabilities and the inverse Weibull time, updates the dataset time and status when the new time is shorter, and returns a censored dataset

@param uncensored The complete follow-up dataset

@param Q_A The median time of the distribution

@param Q_B the Q_{99} time of the distribution

@param pCensoring A proportion of censored cases

@return a censored dataset. A data.frame with columns c(time, status, x)

```
timeCensoring <- function(uncensored, Q_A = 25, Q_B = 100, pCensoring = 0) {
   censored <- uncensored
   n <- nrow(censored)
   pRandom <- runif(n)^((1-pCensoring)/(pCensoring))
   timeCens <- sapply(pRandom, inverseWeibull(Q_A, Q_B))
   censor <- timeCens < censored$time
   censored$time[censor] <- timeCens[censor]
   censored$status[censor] <- 0
   censored$x <- paste("Time censoring", round(100 * pCensoring, 1), "%")
   censored[order(censored$time), ]
}
```

This function takes a complete follow-up dataset and the parameters of a mixed model (events median and Q_{99}, and proportion of censoring), estimates a vector of random interim times (from a vector of random probabilities and the inverse Weibull time) and of random recruit times, updates the dataset time and status when the difference of recruit and interim times is shorter than the original time, and returns a censored dataset

@param uncensored The complete follow-up dataset

@param Q_A The median time of the distribution

@param Q_B the Q_{99} time of the distribution

@param pCensoring A proportion of censored cases

@return a censored dataset. A data.frame with columns c(time, status, x)

```
interimCensoring <- function(uncensored, Q_A = 25, Q_B = 100, pCensoring = 0) {
   censored <- uncensored
   n <- nrow(censored)
   interimTime <- inverseWeibull(Q_A, Q_B)(pCensoring) * 2
   recruitTime <- runif(n, 0, interimTime)
   interval <- interimTime - recruitTime
   censor <- (interimTime - recruitTime) < censored$time
   censored$status[censor] <- 0
   censored$time[censor] <- interval[censor]
   censored$x <- paste("Interim at t =", round(interimTime, 1))
   censored <- censored[censored$time > 0, ]
   censored[order(censored$time), ]
}
```

This function takes a complete follow-up dataset and a proportion of censoring, selects a random sample of cases with that probability, updating the status and times (shortening them a random amount), and returns a censored dataset

@param uncensored The complete follow-up dataset

@param pCensoring A proportion of censored cases

@return a censored dataset. A data.frame with columns c(time, status, x)

```
caseCensoring <- function(group0, pCensoring = 0) {
   n <- nrow(group0)
   nCens  <- as.integer(n * pCensoring)
   censor <- sample(1:n, nCens)
   censored <- group0
   nCensored <- as.integer(nrow(group0) * pCensoring)
   censored$time[censor] <- group0$time[censor] * runif(nCensored, 0.2, 1)
   censored$status[censor] <- 0
   censored$x <- paste("Case", round(100 * (1 - pCensoring), 1), "% censored")
   censored[order(censored$time), ]
}
```

Bias indexes (BI)

This function estimates the quantile bias index (QBI) of a dataset

@param dataset The survival dataset with columns c(time, status, x)

@param time The name of the column corresponding to time

@param status The name of the column correspondint to status

@param event The label of status corresponding to censoring

@return a number The QBI

```
QBI <- function(dataset, time = "time", status = "status", event = 1, ...) {
   events <- dataset[dataset[, status] == event, time]
   censor <- dataset[dataset[, status] != event, time]
   aCensor <- censor[censor < max(events)]
   if(length(aCensor) == 0) NA else quantile(events, 0.95) / quantile(aCensor, 0.95)
}
```

This function estimates the scaled quantile bias index (SQBI) of a dataset

@param dataset The survival dataset with columns c(time, status, x)

@param time The name of the column corresponding to time

@param status The name of the column correspondint to status

@param event The label of status corresponding to censoring

@return a number The SQBI

```
SQBI <- function(dataset, time = "time", status = "status", event = 1, ...) {
   QBI(dataset, time, status, event) / 1.2
}
```

This function estimates the under the mean bias index (UMBI) of a dataset

@param dataset The survival dataset with columns c(time, status, x)

@param time The name of the column corresponding to time

@param status The name of the column correspondint to status

@param event The label of status corresponding to censoring

@return a number The UMBI

```
UMBI <- function(dataset, time = "time", status = "status", event = 1, ...) {
   events <- dataset[dataset[, status] == event, time]
   censor <- dataset[dataset[, status] != event, time]
   lastEvent <- max(events)
   meanEvent <- mean(events)
   aCensor <- censor[censor < lastEvent]
   sum(aCensor < meanEvent) / length(aCensor)
}
```

This function estimates the adjusted bias index (ABI) of a dataset

@param dataset The survival dataset with columns c(time, status, x)

@param time The name of the column corresponding to time

@param status The name of the column correspondint to status

@param event The label of status corresponding to censoring

@return a number The ABI

```r
ABI <- function(dataset, time = "time", status = "status", event = 1, ...) {
   events <- dataset[dataset[, status] == event, time]
   censor <- dataset[dataset[, status] != event, time]
   lastEvent <- max(events)
   meanEvent <- mean(events)
   aCensor<- censor[censor < lastEvent]
   formula1 <- Surv(time, status == event) ~ 1
   survfit1 <- survfit(formula1, data = dataset)
   pLongTerm <- summary(survfit1, times = lastEvent)$surv
   medianCorrection <- mean(events) / median(events)
   longTermCorrection <- exp(pLongTerm)
   sum(aCensor < meanEvent) / length(aCensor) * medianCorrection * longTermCorrection
}
```

This function estimates the scaled adjusted bias index (ABI) of a dataset

@param dataset The survival dataset with columns c(time, status, x)

@param time The name of the column corresponding to time

@param status The name of the column correspondint to status

@param event The label of status corresponding to censoring

@return

```r
SABI <- function(dataset, time = "time", status = "status", event = 1, ...) {
   ABI(dataset, time, status, event) / 0.932
}
```

Experiment functions

This function extracts the hazard ratio and p-value between the two groups labeled in the x column of a survival dataset

@param dataset The survival dataset with columns c(time, status, x)

@param time The name of the column corresponding to time

@param status The name of the column correspondint to status

@param event The label of status corresponding to censoring

@param x Labels for the two groups in the dataset

@return a list with the hazard ratio and p-value betwen the two groups

```r
getHR <- function(dataset, time = "time", status = "status", event = 1, x = "x") {
   formula1 = Surv(time, status == event) ~ x
   cx <- summary(suppressWarnings(coxph(formula1, data = dataset)))
   list(hr = cx$conf.int[[1]], p = cx$coefficients[[5]])
}
```

This function takes a complete follow-up and a censored datasets, the names of columns corresponding to time and status, and the label of status corresponding to censoring, and a BI function, returning a vector with the parameters of the trial

@param group0

@param group1

@param time The name of the column corresponding to time

@param status The name of the column correspondint to status

@param event The label of status corresponding to censoring

@return a vector The parameters of the trial: number of cases, proportion of long-term survivors, hazard ratio, p-value and BI result

```
trialResults <-
   function(group0, group1, time = "time", status = "status", event = 1) {
      events <- group1[group1[, status] == event, time]
      lastEvent <- max(events)
      aCensor <- group1[group1[, time] < lastEvent, ]
      pCensor <- sum(aCensor[, status] != event) / nrow(aCensor)
      pLongTerm <- 1 - nrow(aCensor) / nrow(group1)
      trialDataset <- rbind(group1, group0)
      trialDataset$x <- factor(trialDataset$x, levels = c(group0$x[[1]], group1$x[[1]]))
      cox <- getHR(trialDataset)
      c(nrow(group1), pCensor, pLongTerm, cox$hr, cox$p,
        SQBI(group1), UMBI(group1), SABI(group1))
   }
```

This function simulates a number of virtual trials with the indicated conditions and returns a data.frame with the result of the trial. Every row is a virtual trial and columns are type of censoring, number of cases, proportion of censoring, proportion of long-term survivors, hazard ratio, p-value and BI result

@param nTrials The number of virtual trials to be simulated

@param nCases The number of cases in every virtual trial

@param mediana The median time of events in the censoring dataset

@param pCure The cure-rate in the complete follow-up dataset

@param pCensoring The proportion of censored cases

@param bias The bias function

@return a data.frame

```
simulTrials <- function(nTrials = 300, nCases = 1000, mediana = 20, pCure = 0,
                        pCensoring = "runif(1, 0.05, 0.95)", ...) {
   df <- data.frame(matrix(nrow = 3 * nTrials, ncol = 9), stringsAsFactors = FALSE)
   colnames(df) <-
      c("type", "nCases", "pCensored", "pCured", "hr", "pValue", "SQBI", "UMBI", "SABI")
   for(i in 1:nTrials) {
      nCases1  <- eval(parse(text = nCases))
      median1  <- eval(parse(text = mediana))
      pCure1   <- eval(parse(text = pCure))
      pCensor1 <- eval(parse(text = pCensoring))
      group0 <- completeFA(nCases = nCases1, median1, 100, pCure = pCure1)
      timeCensored    <- timeCensoring   (group0, Q_A = median1, 100, pCensoring = pCensor1)
      interimCensored <- interimCensoring(group0, Q_A = median1, 100, pCensoring = pCensor1)
      caseCensored    <- caseCensoring   (group0, pCensoring = pCensor1)
```

```
      df[3 * i - 2:0, "type"] <- c("time", "interim", "case")
      df[3 * i - 2:0, -1] <- rbind(trialResults(group0, timeCensored),
                                   trialResults(group0, interimCensored),
                                   trialResults(group0, caseCensored))
   }
   df
}
```

This takes a cancer clinical dataset and returns the row of a data.frame with columns: name of the trial, number of cases, proportion of censoring, SQBI and SABI indexes and reference

@param db The clinical cancer dataset

@param trial The name of the trial

@param reference The bibliographic reference

@param time The name of the column corresponding to time

@param status The name of the column correspondint to status

@param event The label of status corresponding to censoring

@return a 1 row data.frame

```
biTrials <- function(db, trial, reference = "", time = "time", status = "status", event = 1) {
   events    <- db[, status] == event
   censor    <- db[, status] != event
   lastEvent <- max(db[events, time], na.rm = FALSE)
   bi1 <- SQBI(db, time = time, status = status, event = event)
   bi3 <- SABI(db, time = time, status = status, event = event)
   data.frame(trial = trial,
              n = nrow(db),
              pCens = sum(censor) / nrow(db),
              SQBI = bi1,
              SABI = bi3,
              reference = if(reference == "") "" else paste0("\\cite{", reference, "}"),
              row.names = "", stringsAsFactors = FALSE)
}
```

Plot functions

This function plots a Kaplan-Meier curve from a survival dataset. If the logical newplot is TRUE, the curve is plotted from scratch, otherwise, the curve is added to the plot

@param dataset The survival dataset with columns c(time, status, x)

@param time The name of the column corresponding to time

@param status The name of the column correspondint to status

@param event The label of status corresponding to censoring

@param x Labels for the groups in the dataset

@param newplot A logical if the curve is to be drawn from scratch or added

@param col The color for the curve

@return a plot A Kaplan-Meier curve with the median

```
KMplot <- function(dataset, time = "time", status = "status", event = 1, x = "x",
                   newplot = FALSE, col = "black", ...) {
   op <- par(mar = c(3, 4, 4, 0.8), mgp = c(2, 0.6, 0))
      if(newplot) {
         xlim = c(0, max(dataset$time, na.rm = FALSE) * 1.1)
         plot(NA, type="n", xlab = "Time", ylab = "Proportion", las = 1,
              xaxs = "i", yaxs = "i", xlim = xlim, ylim = c(1e-2, 1), ...)
         abline(h = 1:3 * 0.25, lty = "dotted")
      }
      formula1 <- Surv(time, status == event) ~ x
      survfit1 <- survfit(formula1, data = dataset)
      median1  <- summary(survfit1)$table["median"]
      lines(survfit1, mark.time = TRUE, conf.int = FALSE, col = col, ...)
      lines(c(median1, median1), c(-1, 0.5), lty = "dotted", lwd = 2, col = col)
   par(op)
}
```

This function takes the complete follow-up and censored datasets and calls KMplot to plot the Kaplan-Meier curves. The censored dataset is plotted in black and the censored dataset in color (red for *time censoring*, blue for *interim censoring*, and green for *case censoring*). In addition, the censored dataset is split into two datasets: events dataset (plotted in orange), and censored in the events interval (plotted in purple). The hazard ratio and p-value comparing the two datasets are written in the title after a label

@param group0 The complete follow-up dataset

@param group1 The censored follow-up dataset

@param time The name of the column corresponding to time

@param status The name of the column correspondint to status

@param event The label of status corresponding to censoring

@param x Labels for the groups in the dataset

@param newplot A logical if the curve is to be drawn from scratch or added

@param main Is a label to compose the plot title

@param col The color for the censored curve

@return a plot The Kaplan-Meier curves of complete follow-up and censored datasets

```
trialPlot <- function(group0, group1, time = "time", status = "status", event = 1, x = "x",
                      main = "", col = "black", ...) {
   KMplot(group0, newplot = TRUE, lwd = 2)
   KMplot(group1, col = col, ...)
   lastEvent <- max(group0[group0$status == event, time])
   events <- group1[group1[, status] == event, ]
   censor <- group1[group1[, status] != event, ]
   censor$status <- 1
   KMplot(events, col = "darkorange", lwd = 2)
   aCensor<- censor[censor[, time] < lastEvent, ]
   KMplot(aCensor, col = "purple", lwd = 2)
   censUncens <- rbind(group1, group0)
   censUncens$x <- factor(censUncens$x, levels = c(group0$x[[1]], group1$x[[1]]))
```

```
   cox <- lapply(getHR(censUncens), signif, digits = 3)
   title(main = paste0(main, ",    hr: ", cox$hr, "    p: ", cox$p), ...)
   subgroups <- c("uncensored dataset", "censored dataset", "events", "censored cases")
   legend("topright", bty = "n", lwd = 2, xpd=TRUE,  legend = subgroups,
          col = c("black", col, "darkorange", "purple") )
}
```

This function plots a set of 3 virtual trials from a complete follow-up dataset and the three types of censoring with the indicated parameters

@param nCases The number of cases in every dataset

@param pCensoring The proportion of censoring in the censoring dataset

@param Q99 The quantile 99 of the events distribution

@param pCure The proportion of cured/long-term survivors

@param mediana The median of the events distribution

@return 3 plots Calls trialPlot 3 times

```
virtualTrials <- function(nCases = 1000, pCensoring = 0.5, Q99 = 100, pCure = 0.5, mediana = 25) {
      group0 <- completeFA(nCases, mediana, Q99, pCure = pCure)
      group1 <- timeCensoring(group0, mediana, Q99, pCensoring = pCensoring)
      group2 <- interimCensoring(group0, mediana, Q99, pCensoring = pCensoring)
      group3 <- caseCensoring(group0, pCensoring = pCensoring)
      trialPlot(group0, group1, main = "Time censoring", col = "red", line = -1)
      trialPlot(group0, group2, main = "Interim censoring", col = "blue", line = -1)
      trialPlot(group0, group3, main = "Case censoring", col = "green", line = -1)
}
```

This function plots a virtual experiment as a scatter plot of *hazard ratio* vs *proportion of censoring* where every experiment is a dot with the corresponding color to the type of censoring (red for *time censoring*, blue for *interim censoring*, and green for *case censoring*)

@param df The experiment dataframe. Every row, a virtual trial, is represented as a dot

@param cols The colores to be used for the 3 types of censoring

@return a scatter plot

```
censorPlot <- function(df, cols = c("red", "blue", "green"), ...) {
   types <-  c("time", "interim", "case")
   pch <- ifelse(df$pValue < 0.05, 19, 21)
   col <- cols[factor(df$type, levels = types)]
   with(df, plot(pCensored, hr, las = 1, pch = pch, col = col, ylim = c(0, 1.2),  ...))
   caseCensored <- df[df[, "type"] == "case", ]
   lm1 <- lm(formula = hr ~ pCensored, data = caseCensored)
   abline(h = 1, lm1)
   df$sig <- (df$pValue < 0.05) * 1
   leg <- legend("bottomleft", bty = "n", paste(types, "censoring"))
   points(x = leg$text$x + leg$rect$w, y = leg$text$y, col = cols, pch = 19)
   points(x = leg$text$x + leg$rect$w * 1.3, y = leg$text$y, col = cols)
   text  (x = leg$text$x - .05, y = leg$text$y[1] + 0.05, "p-value  <0.05  >0.05",
          pos = 4, offset = 6, ...)
```

```
   legend("bottomright", bty = "n",
          legend = paste("r =", with(caseCensored, signif(cor(hr, pCensored), 3))))
}
```

This function plots a virtual experiment as a scatter plot of *hazard ratio* vs *bias index* where every experiment is a dot with the corresponding color to the type of censoring (red for *time censoring*, blue for *interim censoring*, and green for *case censoring*)

@param df The experiment dataframe. Every row, a virtual trial, is represented as a dot

@param cols The colores to be used for the 3 types of censoring

@return a plot A scatter plot of *hazard ratio* vs *proportion of censoring*. Every virtual trial is a dot with the corresponding color to the type of censoring (red for *time censoring*, blue for *interim censoring*, and green for *case censoring*)

```
biasPlot <- function(df, cols = c("red", "blue", "green"), BI, ...) {
   X <- df[, BI]
   types <-  c("time", "interim", "case")
   pch <- ifelse(df$pValue < 0.05, 19, 1)
   col <- cols[factor(df$type, levels = types)]
   with(df, plot(X, hr, las = 1, pch = pch, col = col, xlab = BI, ylim = c(0, 1.6), ...))
   leg1 <- legend("topleft", "               p-value  <0.05  >0.05", bty = "n")
   leg <- legend(x = leg1$rect$left, y = leg1$text$y, bty = "n", paste(types, "censoring"))
   points(x = leg$text$x + leg$rect$w, y = leg$text$y, col = cols, pch = 19)
   points(x = leg$text$x + leg$rect$w * 1.3, y = leg$text$y, col = cols)
   df$sig <- (df$pValue < 0.05) * 1
   oCutPoints <- optimal.cutpoints(
      X = BI, status = "sig", data = df, tag.healthy = 0, methods = "Youden")
   sCutPoints <- summary(oCutPoints)$p.table$Global
   roc <- legend("topright", bty = "n", xjust = 0.5,
                 legend = c("Area under ROC", sCutPoints$AUC_CI))
   params <- signif(sCutPoints$Youden[[1]][1:5, ], 3)
   cutoff <- params[[1]]
   params <- paste(c("Cutoff", "Sensit", "Specif", "posPV", "negPV"), params, sep = ": ")
   abline(h = 1, v = cutoff)
   threshold <- min(BI[df$pValue < 0.05], na.rm = FALSE)
   legend("bottomleft", legend = params, bty = "n")
}
```

Figures

Figure 1

Simulation of 12 virtual experiments based on 4 complete follow-up datasets with median survival of 25 and 50 time units ($Q_{99} = 100$), and cure-rate of 0 and 0.4. For every complete follow-up dataset with 1000 cases, three censored datasets are obtained with the three different mechanisms of censoring and proportion of censoring of 50%.

```
set.seed(1963)
op <- par(mfrow = c(4, 3), cex = 0.8)
   virtualTrials(pCure = 0, mediana = 25)
   virtualTrials(pCure = 0, mediana = 50)
```

```
   virtualTrials(pCure = 0.4, mediana = 50)
   virtualTrials(pCure = 0.4, mediana = 50)
par(op)
```

Virtual experiments

Simulates different experiments, every one with 3000 virtual trials based on 1000 random complete follow-up datasets with a range of median distribution of events and cure-rates

```
n = 1000
set.seed(1963); experiment1 <-
   simulTrials(nTrials = n, mediana = "runif(1, 5, 50)", pCure = "0")
set.seed(1963); experiment2 <-
   simulTrials(nTrials = n, mediana = "runif(1, 50, 95)", pCure = "0")
set.seed(1963); experiment3 <-
   simulTrials(nTrials = n, mediana = "runif(1, 5, 50)", pCure = "0.5")
set.seed(1963); experiment4 <-
   simulTrials(nTrials = n, mediana = "runif(1, 50, 95)", pCure = "0.5")
set.seed(1963); experiment5 <-
   simulTrials(nTrials = n, mediana = "runif(1, 5, 95)", pCure = "runif(1, 0, 0.8)")
```

Figure 2

Simulation of 3000 virtual trials based on 1000 random complete follow-up datasets with medians of 5 to 95 time units and cure-rate of 0 to 0.8; time censoring (red), interim censoring (blue) and case censoring (green) datasets are simulated with probability of censoring of 5% to 95%.

```
op <- par(cex = 0.8)
   censorPlot(experiment5, xlim = c(0.075, 0.925), xlab = "proportion of censoring",
              main = "median: 5-95   pCure = 0.-0.8")
par(op)
```

Figure 3

Simulation of 2 experiments, each of them with 3000 trials based on 1000 complete follow-up datasets and the 3 mechanisms of censoring described, with random proportion of censoring in the range 0.05 to 0.95.

```
op <- par(mfrow = c(1, 2))
   biasPlot(experiment1, BI = "SQBI",  main = "median: 5-50   pCure = 0", xlim = c(0.2, 1.9))
   biasPlot(experiment5, BI = "SQBI",  main = "median: 5-95   pCure = 0-0.8")
par(op)
```

Figure 4

Simulation of 4 virtual experiments with a 2x2 design: right skewed Weibull distributions or not, and cure-rate of 0 or 50%.

```
op <- par(mfrow = c(2, 2))
   biasPlot(experiment1, BI = "UMBI", main = "median:  5-50   pCure = 0.")
   biasPlot(experiment2, BI = "UMBI", main = "median: 50-95   pCure = 0.")
```

```
   biasPlot(experiment3, BI = "UMBI", main = "median:  5-50   pCure = 0.5")
   biasPlot(experiment4, BI = "UMBI", main = "median: 50-95   pCure = 0.5")
par(op)
```

Figure 5

Simulation of 2 experiments, each of them with 3000 trials based on 1000 complete follow-up datasets and the 3 mechanisms of censoring described, with random proportion of censoring in the range 0.05 to 0.95.

```
op <- par(mfrow = c(1, 2))
   biasPlot(experiment1, BI = "SABI",  main = "median: 5-50   pCure = 0", xlim = c(0.2, 1.9))
   biasPlot(experiment5, BI = "SABI",  main = "median: 5-95   pCure = 0-0.8", xlim = c(0.2, 2.5))
par(op)
```

Clinical available datasets

Available clinical datasets were retrieved from Documentation for package *survival* version 2.44-1.1, based on 6 published articles (https://stat.ethz.ch/R-manual/R-devel/library/survival/html/00Index.html). Bias indexes SQBI and SABI were applied to the datasets and to the research arms where available.

Acute Myelogenous Leukemia survival data

https://stat.ethz.ch/R-manual/R-devel/library/survival/html/aml.html

```
trialsBI <- biTrials(aml, "Acute Myelogenous Leukemia survival data", "Miller1997")
```

Bladder Cancer Recurrences

https://stat.ethz.ch/R-manual/R-devel/library/survival/html/bladder.html

```
data(bladder)
bladder1$time   <- with(bladder1, stop - start)
bladder1$status <- (bladder1$status == 1 | bladder1$status == 2) * 1
names(bladder1)[2] <- "x"
trialsBI <- rbind(trialsBI,
                  biTrials(bladder1, "Bladder Cancer Recurrences - bladder1", "Wei1989"))
placebo    <- bladder1[bladder1$x == "placebo", ]
pyridoxine <- bladder1[bladder1$x == "pyridoxine", ]
thiotepa   <- bladder1[bladder1$x == "thiotepa", ]
trialsBI <- rbind(trialsBI, biTrials(placebo,    " - placebo", ""))
trialsBI <- rbind(trialsBI, biTrials(pyridoxine, " - pyridoxine", ""))
trialsBI <- rbind(trialsBI, biTrials(thiotepa,   " - thiotepa", ""))

bladder2$time <- with(bladder2, stop - start)
names(bladder2)[2] <- "x"
names(bladder2)[7] <- "status"
trialsBI <- rbind(trialsBI,
                  biTrials(bladder2, "Bladder Cancer Recurrences - bladder2", "Wei1989"))
rx1 <- bladder2[bladder2$x == 1, ]
rx2 <- bladder2[bladder2$x == 2, ]
```

```
trialsBI <- rbind(trialsBI, biTrials(rx1, " - rx1", ""))
trialsBI <- rbind(trialsBI, biTrials(rx2, " - rx2", ""))
```

NCCTG Lung Cancer Data

https://stat.ethz.ch/R-manual/R-devel/library/survival/html/lung.html

```
data(lung)
lung$status <- (lung$status == 2) * 1
lung$x = ""
trialsBI <- rbind(trialsBI, biTrials(lung, "NCCTG Lung Cancer Data", "Loprinzi1994"))
```

Chemotherapy for Stage B/C colon cancer

https://stat.ethz.ch/R-manual/R-devel/library/survival/html/colon.html

```
data(colon)
colon$x <- colon$rx
trialsBI <- rbind(trialsBI,
                  biTrials(colon, "Chemotherapy for Stage B/C colon cancer", "Moertel1990"))
Obs <- colon[colon$rx == "Obs", ]
Lev <- colon[colon$rx == "Lev", ]
LFU <- colon[colon$rx == "Lev+5FU", ]
trialsBI <- rbind(trialsBI, biTrials(Obs, " - Observation", ""))
trialsBI <- rbind(trialsBI, biTrials(Lev, " - Levamisol", ""))
trialsBI <- rbind(trialsBI, biTrials(LFU, " - Levamisol + 5FU", ""))
```

Ovarian Cancer Survival Data

https://stat.ethz.ch/R-manual/R-devel/library/survival/html/ovarian.html

```
data(ovarian)
ovarian$time   <- ovarian$futime
ovarian$status <- ovarian$fustat
ovarian$x      <- ovarian$rx
trialsBI <- rbind(trialsBI,
                  biTrials(ovarian, "Ovarian Cancer Survival Data", "Edmonson1979"))
rx1 <- ovarian[ovarian$rx == 1, ]
rx2 <- ovarian[ovarian$rx == 2, ]
trialsBI <- rbind(trialsBI, biTrials(rx1, " - rx1", ""))
trialsBI <- rbind(trialsBI, biTrials(rx2, " - rx2", ""))
```

Veterans' Administration Lung Cancer Study

https://stat.ethz.ch/R-manual/R-devel/library/survival/html/veteran.html

```
data(veteran)
veteran$x <- veteran$trt
trialsBI <- rbind(trialsBI,
            biTrials(veteran, "Veterans' Administration Lung Cancer Study", "Kalbfleisch1980"))
```

```
standard <- veteran[veteran$trt == 1, ]
test     <- veteran[veteran$trt == 2, ]
trialsBI <- rbind(trialsBI, biTrials(standard, " - standard", ""))
trialsBI <- rbind(trialsBI, biTrials(test,     " - test", ""))
```

Table 1

```
table1 <- trialsBI
rownames(table1) <- NULL
table1[, c("pCens", "SQBI", "SABI")] <- signif(trialsBI[, c("pCens", "SQBI", "SABI")], 2)

highlight <- function(values, threshold = 1) {
   selected <- values > threshold
   values[selected] <- paste0("\\textbf{", values[selected], "}")
   values
}
table1[, "SQBI"] <- highlight(table1[, "SQBI"], 1)
table1[, "SABI"] <- highlight(table1[, "SABI"], 1)

options(xtable.comment = FALSE)
xt <- xtable(table1,  format = "latex", digits = 2,
             caption = "Bias in available clinical datasests")
print(xt, caption.placement = "top", type = "latex", include.rownames = FALSE,
      sanitize.text.function = identity)
```

sessionInfo()

```
## R version 4.0.3 (2020-10-10)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Catalina 10.15.7
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] xtable_1.8-4           OptimalCutpoints_1.1-4 survival_3.2-7
## [4] knitr_1.30
##
## loaded via a namespace (and not attached):
##  [1] lattice_0.20-41 digest_0.6.27   grid_4.0.3      magrittr_2.0.1
##  [5] evaluate_0.14   rlang_0.4.9     stringi_1.5.3   Matrix_1.2-18
##  [9] rmarkdown_2.5   splines_4.0.3   tools_4.0.3     stringr_1.4.0
## [13] xfun_0.19       yaml_2.2.1      compiler_4.0.3  htmltools_0.5.0
```