Companion Diagnostics

Rasmus Brøndum & Martin Bøgsted

3 May 2018

Load libraries

```
library("MASS")
library("splines")
library("survival")
library("Hmisc")
```

```
## Loading required package: lattice

## Loading required package: Formula

## Loading required package: ggplot2

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
library("knitr")
library("survsim")
```

```
## Loading required package: eha

## Loading required package: statmod
```

```
library("ggplot2")
library("foreach")
library("doParallel")
```

```
## Loading required package: iterators

## Loading required package: parallel
```

```
library("survPresmooth")
```

Set up parralel computations

```
registerDoParallel(cores = 20)
```

User defined functions

Load function to extract Individual patient data (IPD) from Kaplan Meier (KM) plots (Guyot et al. 2012).

```
source("Guyot/extractFromKMcurveOrig_temp.R")
```

Define functions to fix problems with digitized data (y data should be descending), x should be sorted.

```
fixMonotony <- function(x){
  for(i in 2:length(x)){
    if(x[i] > x[i-1]) x[i] <- x[i - 1]
  }
```

```r
  return(x)
}

fixMonotonx <- function(x){
  for(i in 1:(length(x)-1)){
    if(x[i] > x[i+1]) x[i] <- x[i + 1]
  }
  return(x)
}
```

Define function to build a risk table in the right format for retrieving IPD data from KM plots.

```r
riskTable <- function(risk_time, nrisk, points){
  k <- seq_along(risk_time)
  interval <- findInterval(points[,1], risk_time)
  lower <- rep(NA, length(risk_time))
  upper <- rep(NA, length(risk_time))
  for(i in seq_along(risk_time)){
    tempPoints <- (1:nrow(points))[interval == i]
    if(length(tempPoints) > 0){
      lower[i] <- min((1:nrow(points))[interval == i], na.rm = TRUE)
      upper[i] <- max((1:nrow(points))[interval == i], na.rm = TRUE)
    }
  }
  result <- data.frame(k, risk_time, lower, upper, nrisk)
  return(result)
}
```

Function to calculate using Simons method (Simon 2015).

```r
simon <- function(HR, nEvents = NA, concordance = "PPV"){
  ## Function to calculate the positive or negative predictive value
  ## using the semi-parametric method from Simon (2015).
  ## Input are the Hazard ratio (HR) of treatment vs control,
  ## the number of events and the desired concordance measure PPV / NPV.
  ## Note that hazard ratios must be calculated in either
  ## Marker posive or negative groups for respectively PPV and NPV

  ## approximate Confidence intervals for HR
  if(!is.na(nEvents)){
    meanConf <- log(HR)
    sdConf   <- sqrt( 4 / nEvents)
    HRlow    <- exp(qnorm(0.025, mean = meanConf, sd = sdConf))
    HRhigh   <- exp(qnorm(0.975, mean = meanConf, sd = sdConf))
  } else{
    HRlow  <- NA
    HRhigh <- NA
  }

  if(concordance == "PPV"){
    est      <- 1 / (1 + HR)
    est_low  <- 1 / (1 + HRlow)
    est_high <- 1 / (1 + HRhigh)
  } else{
    est      <- HR / (1 + HR)
    est_low  <- HRlow / (1 + HRlow)
    est_high <- HRhigh / (1 + HRhigh)
  }
  conf <- sort(c(est_low, est_high))
  pub  <- paste0(round(est,2), " (", round(conf[1],2), "; ", round(conf[2],2), ")")

  results <- list("estimate" = est,
                  "95% conf" = conf,
                  "pub" =pub)
  return(results)
}
```

Function to calculate Non-parametric PPV

```
kNonP <- function(sdata, tau = Inf, concordance = "PPV"){
  library("survPresmooth")
  ## Function to calculate the positive or negative predictive value
  ## using the non-parametric methods suggested in this chapter.
  ## Input is a dataframe with columns t,d,x,os giving respectively:
  ## t: Time to event;
  ## d: event (1) / censoring (0);
  ## x: Treatment (2) vs Control (1)
  ## os: survival object, Surv(t,d),
  ## an integer, tau, for truncating the evaluation interval,
  ## and the desired concordance measure PPV / NPV.

  if(all(sdata$d == 1)){
    sdata$km.cens <- rep(1, nrow(sdata))
  } else{
    # smoothed Kaplan meier for censoring distribution
    km.cens.smooth <- presmooth(times = sdata$t, status = !sdata$d, estimand = "S",
                                bw.selec = "plug-in")
    rank.t <- rank(sdata$t)

    sdata$km.cens <- km.cens.smooth$estimate
    sdata$km.cens[length(sdata$km.cens)] <- sdata$km.cens[length(sdata$km.cens)-1]
    sdata$km.cens <- sdata$km.cens[rank.t]
  }

  sdata$tau <- as.integer(sdata$t < tau)

  if(concordance == "PPV"){
    sdataT <- sdata[sdata$x == 2,]
    sdataC <- sdata[sdata$x == 1,]
    nT <- nrow(sdataT)
    nC <- nrow(sdataC)
    test <- function(x,y) as.integer(x > y)
  } else if(concordance =="NPV"){
    ## Reverse group labeling and test indicator function
    ## since definition for NPV is S_C >= S_T
    sdataC <- sdata[sdata$x == 2,]
    sdataT <- sdata[sdata$x == 1,]
    nT <- nrow(sdataT)
    nC <- nrow(sdataC)
    test <- function(x,y) as.integer(x >= y)
  }



  # Estimate
  noD <- 0
  for(j in 1:nT){
    noD <- noD + sum(test(sdataT$t[j], sdataC$t)* sdataC$km.cens^-2 * sdataC$d * sdataC$tau)
  }
  est <- noD / (nT * nC)

  # Variance
  xi11 <- 0
  for(i in 1:nT){
    xi11 <- xi11 + sum(test(sdataT$t[i], sdataC$t)* sdataC$km.cens^-4 * sdataC$d  * sdataC$tau)
  }
  xi11 <- xi11 /(nT*nC)

  xi01 <- 0
  for(i in 1:nT){
    y1 <- test(sdataT$t[i], sdataC$t)* sdataC$km.cens^-2 * sdataC$d * sdataC$tau
    res <- y1 %*% t(y1)
    xi01 <- xi01 + (sum(res) - sum(diag(res)))/(nC*(nC-1))
  }
  xi01 <- xi01 / nT

  xi10 <- 0
  for(i in 1:nC){
    y1 <- test(sdataT$t,sdataC$t[i]) * sdataC$km.cens[i]^-2 * sdataC$d[i] * sdataC$tau[i]
    res <- y1 %*% t(y1)
    xi10 <- xi10 + (sum(res) - sum(diag(res)))/(nT*(nT-1))
  }
  xi10 <- xi10 / nC

  std <- sqrt( ((nT*nC)        * xi11  +
                (nT*nC*(nC-1)) * xi01  +
                (nC*nT*(nT-1)) * xi10  -
                (nT*nC + nT*nC*(nC-1) + nC*nT*(nT-1))*est^2) / (nT^2*nC^2))


  est_low  <- max(0, qnorm(0.025, mean = est, sd = std))
  est_high <- min(qnorm(0.975, mean = est, sd = std), 1)
  conf <- sort(c(est_low, est_high))
  pub  <- paste0(round(est,2), " (", round(conf[1],2), "; ", round(conf[2],2), ")")

  results <- list("estimate" = est,
                  "95% conf" = conf,
                  "pub" =pub)
  return(results)
}
```

```
kNonP <- function(sdata, tau = Inf){
  if(all(sdata$d == 1)){
```

```
  sdata$km.cens <- rep(1, nrow(sdata))
} else{
  # Kaplan meier for censoring distribution
  #km.cens <- survfit(Surv(sdata$t, as.integer(!sdata$d)) ~ 1)
  km.cens.smooth <- presmooth(times = sdata$t, status = !sdata$d, estimand = "S",
                        bw.selec = "plug-in")
  rank.t <- rank(sdata$t)

  #sdata$km.cens <- summary(km.cens, times = sdata$t)$surv[rank.t] + 0.001
  #sdata$km.cens <- summary(km.cens, times = sdata$t)$surv

  sdata$km.cens <- km.cens.smooth$estimate
  sdata$km.cens[length(sdata$km.cens)] <- sdata$km.cens[length(sdata$km.cens)-1]
  sdata$km.cens <- sdata$km.cens[rank.t]


  #If missing replace with minimum value (assume constant after last event)
  #sdata$km.cens[is.na(sdata$km.cens)] <- min(sdata$km.cens, na.rm = T)
  #sdata$km.cens <- 1-pexp(sdata$t,rate=1/20) # Give it a try
}

sdata$tau <- as.integer(sdata$t < tau)

sdataT <- sdata[sdata$x == 2,]
sdataC <- sdata[sdata$x == 1,]
nT <- nrow(sdataT)
nC <- nrow(sdataC)

# Estimate
noD <- 0
for(j in 1:nT){
  noD <- noD + sum(as.integer(sdataT$t[j] > sdataC$t) * sdataC$km.cens^-2 * sdataC$d * sdataC$tau)
}
est <- noD / (nT * nC)

# Variance

xi11 <- 0
for(i in 1:nT){
  xi11 <- xi11 + sum(as.integer(sdataT$t[i] > sdataC$t) * sdataC$km.cens^-4 * sdataC$d  * sdataC$tau)
}
xi11 <- xi11 /(nT*nC)

xi01 <- 0
for(i in 1:nT){
  y1 <- as.integer(sdataT$t[i] > sdataC$t) * sdataC$km.cens^-2 * sdataC$d * sdataC$tau
  res <- y1 %*% t(y1)
  xi01 <- xi01 + (sum(res) - sum(diag(res)))/(nC*(nC-1))
}
xi01 <- xi01 / nT

xi10 <- 0
for(i in 1:nC){
```

```r
    y1 <- as.integer(sdataT$t > sdataC$t[i]) * sdataC$km.cens[i]^-2 * sdataC$d[i] * sdataC$tau[i]
    res <- y1 %*% t(y1)
    xi10 <- xi10 + (sum(res) - sum(diag(res)))/(nT*(nT-1))
  }
  xi10 <- xi10 / nC

  std <- sqrt( ((nT*nC)        * xi11  +
               (nT*nC*(nC-1)) * xi01  +
               (nC*nT*(nT-1)) * xi10  -
               (nT*nC + nT*nC*(nC-1) + nC*nT*(nT-1))*est^2) / (nT^2*nC^2))


  est_low  <- max(0, qnorm(0.025, mean = est, sd = std))
  est_high <- min(qnorm(0.975, mean = est, sd = std), 1)
  conf <- sort(c(est_low, est_high))
  pub  <- paste0(round(est,2), " (", round(conf[1],2), "; ", round(conf[2],2), ")")

  results <- list("estimate" = est,
                  "95% conf" = conf,
                  "pub" =pub)
  return(results)
}
```

Monte carlo integration of log-normal PPV

```r
fxlnorm = function(x, sigma = 1, beta0 = 1, beta1  = 1){
  1/(x*sigma*sqrt(2*pi))*exp(-(1/(2*sigma^2))*(log(x)-beta0)^2) * # Control density
    (1 - pnorm((log(x) - (beta0+beta1))/sigma))                 # Treatment Survival function
}

LogNormMCI = function(sigma = 1, beta0 = 1, beta1  = 1, interval = c(0,30), nSamples = 1000000){
  y <- runif(n = nSamples, min = interval[1], max = interval[2])
  M <- interval[2] - interval[1]
  M * (1/nSamples) * sum(fxlnorm(y, beta1 = beta1, beta0 = beta0, sigma = sigma))
}
```

Monte Carlo integration of log-logistic PPV

```r
fxloglog = function(x, gamma = 2, beta0 = 1, beta1  = 1){
  lambdaC <- exp(-beta0)
  lambdaT <- exp(-(beta0 + beta1))
  beta = 1/gamma
  alphaC = 1/lambdaC
  alphaT = 1/lambdaT

  dllogis(x, shape = beta, scale = alphaC) * (1 - pllogis(x, shape = beta, scale = alphaT))
}

LogLogMCI = function(gamma = 2, beta0 = 1, beta1  = 1, interval = c(0,30), nSamples = 1000000){
  y <- runif(n = nSamples, min = interval[1], max = interval[2])
  M <- interval[2] - interval[1]
  M * (1/nSamples) * sum(fxloglog(y, beta1 = beta1, beta0 = beta0, gamma = gamma))
}
```

Function to perform simulation and calculate estimates

```r
simFunc <- function(simDist = "weibull", betaZero = 1, betaCens = 3, betaTrue = 1, anc.ev = 1){
  a <- simple.surv.sim(n = 1000,
                       foltime = 1000,
                       dist.ev = simDist,
                       dist.cens = "weibull",
                       anc.ev = anc,
                       anc.cens = 1,
                       beta0.ev = betaZero,
                       beta0.cens = betaCens,
                       x = list(c("bern", 0.5)),
                       beta = list(c(betaTrue)))

  a$S <- with(a, Surv(stop, status))
  cox.res <- with(a, coxph(S~x))
  test.ph <- cox.zph(cox.res)

  results <- c()

  # Store PPV estimate from Simon
  simon.est <- simon(exp(coef(cox.res)), cox.res$nevent)
  results["Simon"] <- simon.est$estimate

  # Check if confidence interval contains true PPV
  results["SimonCoverage"]  <- findInterval(PPV, simon.est$`95% conf`) == 1

  sdata <- data.frame("t"=a$S[,1], "d"=a$S[,2], "x"=a$x, "OS"=a$S)
  sdata$x <- sdata$x + 1


  noPar.est <- kNonP(sdata)
  results["nonPar"] <- noPar.est$estimate
  # Check if confidence interval contains true PPV
  results["nonParCoverage"] <- findInterval(PPV, noPar.est$`95% conf`) == 1
  return(results)
}
```

Simulated data

In this section data are simulated to investigate differences between the method by (Simon 2015) and the non-parametric estimator. We estimate from an exponential distribution (Weibull with shape parameter = 1) which follows the proportional hazards assumptions required by Simons method and from a log-normal follow which does not.

```r
set.seed(314)
betaZero <- 1
betaTrue <- 1
anc      <- 1
nSim     <- 10

allResults <- list()

### Get true PPVs
```

```
PPVvec <- c()
PPVvec["weibull"] <- 1/(1+exp(- anc * betaTrue))
PPVvec["lnorm"] <- LogNormMCI(beta1 = betaTrue, beta0 = betaZero, sigma = anc)
PPVvec["llogistic"] <- LogLogMCI(beta1 = betaTrue, beta0 = betaZero, gamma = anc)

simDist  <- c("weibull", "lnorm", "llogistic")
betaCens <- c(3, 100)

for(sim in simDist){

  ## Set TRUE PPV for the scenario
  PPV <- PPVvec[sim]

  for(betac in betaCens){
    results <- foreach(i=1:nSim, .combine = "rbind", .packages = "survsim") %dopar% {
      simFunc(simDist = sim,
              betaZero = betaZero,
              betaTrue = betaTrue,
              betaCens = betac,
              anc.ev = anc)
    }
    allResults[[sim]][[paste(betac)]] <- results
  }
}


## Build table with results
resultsTable <- cbind("betaCens" = rep(betaCens, length(simDist)),
                      "TruePPV" = rep(PPVvec[simDist], each = 2),
                      t(do.call(cbind, lapply(allResults, function(x) sapply(x, colMeans)))))

kable(resultsTable, digits = 3, caption = "Estimated PPV from simulated data using different approaches
```

Table 1: Estimated PPV from simulated data using different approaches

	betaCens	TruePPV	Simon	SimonCoverage	nonPar	nonParCoverage
weibull	3	0.731	0.731	0.9	0.733	1
weibull	100	0.731	0.727	1.0	0.726	1
lnorm	3	0.762	0.736	0.5	0.756	1
lnorm	100	0.762	0.712	0.0	0.759	1
llogistic	3	0.651	0.648	1.0	0.656	1
llogistic	100	0.651	0.618	0.5	0.656	1

Amado 2008 - KRAS study

Data from (Amado et al. 2008) Fig2A and Fig2B was digitezed using the software DigitizeIT. These data are loaded below for wt KRAS (blue) and mutated KRAS (yellow).

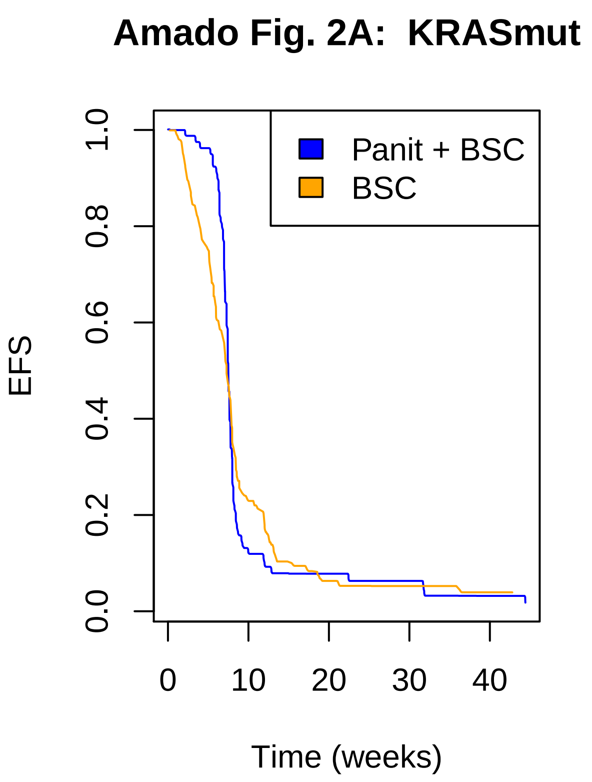
```
digitDir <- "../ExternalData/amado2008/"
Amado2aBlue   <- read.table(file.path(digitDir, "armado_2a_blue.csv"),
                            skip = 1, header = F, sep = ";", dec = ",")
Amado2aYellow <- read.table(file.path(digitDir, "armado_2a_yellow.csv"),
                            skip = 1, header = F, sep = ";", dec = ",")
Amado2bBlue   <- read.table(file.path(digitDir, "armado_2b_blue.csv"),
                            skip = 1, header = F, sep = ";", dec = ",")
Amado2bYellow <- read.table(file.path(digitDir, "armado_2b_yellow.csv"),
                            skip = 1, header = F, sep = ";", dec = ",")
```

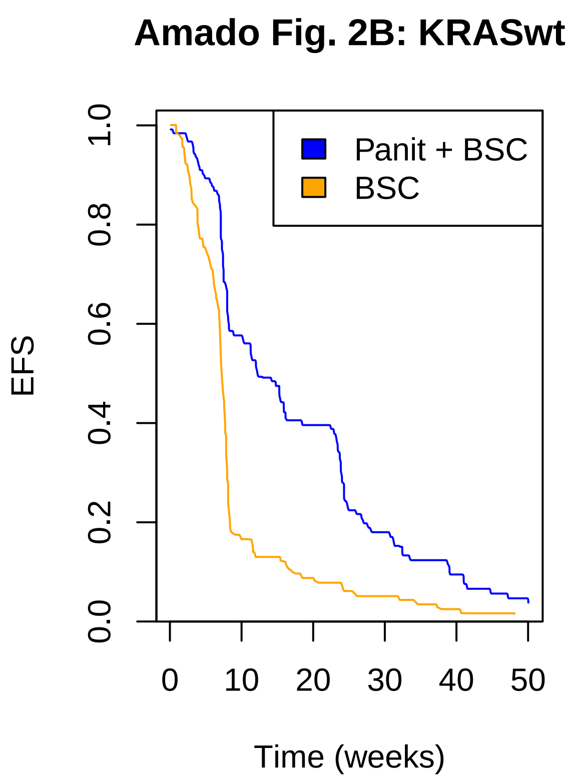
Data are sorted by x coordinate, and possible problems with the digitization procedure are fixed

```
## Sort by x-coordinate
Amado2aBlue   <- Amado2aBlue[order(Amado2aBlue$V1), ]
Amado2aYellow <- Amado2aYellow[order(Amado2aYellow$V1), ]
Amado2bBlue   <- Amado2bBlue[order(Amado2bBlue$V1), ]
Amado2bYellow <- Amado2bYellow[order(Amado2bYellow$V1), ]

## Fix problems with monotony of data
Amado2aBlue[,1]   <- fixMonotonx(Amado2aBlue[,1])
Amado2aYellow[,1] <- fixMonotonx(Amado2aYellow[,1])
Amado2aBlue[,2]   <- fixMonotony(Amado2aBlue[,2])
Amado2aYellow[,2] <- fixMonotony(Amado2aYellow[,2])
Amado2bBlue[,1]   <- fixMonotonx(Amado2bBlue[,1])
Amado2bYellow[,1] <- fixMonotonx(Amado2bYellow[,1])
Amado2bBlue[,2]   <- fixMonotony(Amado2bBlue[,2])
Amado2bYellow[,2] <- fixMonotony(Amado2bYellow[,2])
```

Plot the digitized data to verify it

```
par(mfrow=c(1,2))
plot(Amado2aBlue[,1], Amado2aBlue[,2], col = "blue",
     type = "l", main = "Amado Fig. 2A:  KRASmut", xlab = "Time (weeks)", ylab = "EFS")
lines(Amado2aYellow[,1], Amado2aYellow[,2], col = "orange")
legend("topright", fill = c("Blue", "Orange"), legend = c("Panit + BSC", "BSC"))

plot(Amado2bBlue[,1], Amado2bBlue[,2], col = "blue",
     type  ="l", main = "Amado Fig. 2B: KRASwt", xlab = "Time (weeks)", ylab = "EFS")
lines(Amado2bYellow[,1], Amado2bYellow[,2], col = "orange")
legend("topright", fill = c("Blue", "Orange"), legend = c("Panit + BSC", "BSC"))
```





Read in data from risk tables in (Amado et al. 2008)

```
risk_time <- seq(0, 50, by = 2)
nrisk_aBlue   <- riskTable(risk_time, c(84, 78, 76, 72, 26, 10, 8, 6, 5, 5, 5, 5, 4,
              4, 4, 4, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1), Amado2aBlue)
nrisk_aYellow <- riskTable(risk_time, c(100, 91, 77, 61, 37, 22, 19, 10, 9, 8, 6, 5, 5,
              4, 4, 4, 4, 4, 4, 3, 3, 3, 2, 2, 2, 2), Amado2aYellow)
nrisk_bBlue   <- riskTable(risk_time, c(124, 119, 112, 106, 80, 69, 63, 58, 50, 45, 44,
              44, 33, 25, 21, 20, 17, 13, 13, 13, 10, 7, 7, 6, 5, 5), Amado2bBlue)
nrisk_bYellow <- riskTable(risk_time, c(119, 109, 91, 81, 38, 20, 15, 15, 14, 11, 10, 9,
              9, 6, 6, 6, 6, 5, 4, 3, 3, 2, 2, 2, 2, 1), Amado2bYellow)
```

Specify number of events

```
tot_a1 <- 76
tot_a2 <- 95
tot_b1 <- 115
tot_b2 <- 114
```

Extract IPD data from digitized KM curves

```
path <- "../GeneratedData/amado2008/"
a1 <- dataFromKMCurve(digizeit = Amado2aBlue,
                nrisk = nrisk_aBlue[!is.na(nrisk_aBlue$lower), ],
                KMdatafile = "/armado_2a1_km.txt",
                KMdataIPDfile = "/armado_2a1_km_IPD.txt",
                tot.events = tot_a1,
                arm.id = 2)

a2 <- dataFromKMCurve(digizeit = Amado2aYellow,
                      nrisk = nrisk_aYellow[!is.na(nrisk_aYellow$lower), ],
```

```r
                      KMdatafile = "/armado_2a2_km.txt",
                      KMdataIPDfile = "/armado_2a2_km_IPD.txt",
                      tot.events = tot_a2,
                      arm.id = 1)

b1 <- dataFromKMCurve(digizeit = Amado2bBlue,
                      nrisk = nrisk_bBlue[!is.na(nrisk_bBlue$lower), ],
                      KMdatafile = "/armado_2b1_km.txt",
                      KMdataIPDfile = "/armado_2b1_km_IPD.txt",
                      tot.events = tot_b1,
                      arm.id = 2)

b2 <- dataFromKMCurve(digizeit = Amado2bYellow,
                      nrisk = nrisk_bYellow[!is.na(nrisk_bYellow$lower), ],
                      KMdatafile = "/armado_2b2_km.txt",
                      KMdataIPDfile = "/armado_2b2_km_IPD.txt",
                      tot.events = tot_b2,
                      arm.id = 1)
```

IPD data is written to output files, so these are loaded

```r
Amado2a_BlueIPD   <- read.table(file.path(path, "armado_2a1_km_IPD.txt"))
Amado2a_YellowIPD <- read.table(file.path(path, "armado_2a2_km_IPD.txt"))
Amado2b_BlueIPD   <- read.table(file.path(path, "armado_2b1_km_IPD.txt"))
Amado2b_YellowIPD <- read.table(file.path(path, "armado_2b2_km_IPD.txt"))
```

Combine IPD data into dataframes for Fig2A (true HR = 0.99) and Fig2B (true HR = 0.45), redo cox regression to see if we get the same results from the re-created IPD.

```r
Amado2a_IPD <- rbind(Amado2a_BlueIPD, Amado2a_YellowIPD)
Amado2b_IPD <- rbind(Amado2b_BlueIPD, Amado2b_YellowIPD)
names(Amado2a_IPD) <- c("t", "d", "x")
names(Amado2b_IPD) <- c("t", "d", "x")

## Cox for fig 2a
amadoCoxA <- coxph(with(Amado2a_IPD, Surv(t,d)~x))
amadoCoxA
```

```
## Call:
## coxph(formula = with(Amado2a_IPD, Surv(t, d) ~ x))
##
##     coef exp(coef) se(coef)   z    p
## x 0.0315    1.0320   0.1582 0.2 0.84
##
## Likelihood ratio test=0.04  on 1 df, p=0.842
## n= 184, number of events= 166
```

```r
## Cox for fig 2b
amadoCoxB <- coxph(with(Amado2b_IPD, Surv(t,d)~x))
amadoCoxB
```

```
## Call:
## coxph(formula = with(Amado2b_IPD, Surv(t, d) ~ x))
##
##     coef exp(coef) se(coef)     z     p
## x -0.771     0.463    0.139 -5.54 3e-08
```

```
##
## Likelihood ratio test=30.1  on 1 df, p=4.09e-08
## n= 243, number of events= 221
```

Calculate PPV and NPV using method from (Simon 2015) from published HR in (Amado et al. 2008) and recalculated from IPD and the non-parametric estimate of PPV and NPV from IPD data.

```
results <- array(0, c(2,3))
rownames(results) <- c("PPV: Longer survival for Panitumumab in KRASwt",
                       "NPV: not Longer survival for Panitumumab in KRASmut")
colnames(results) <- c("Simon", "Simon_IPD","NonParametric K")


## Simon PPV from published or recalculated IPD data
results[1,1] <- simon(0.45, (tot_b1+tot_b2))$pub
results[1,2] <- simon(exp(coef(amadoCoxB)), amadoCoxB$nevent)$pub

## Simon PPV from published or recalculated IPD data
results[2,1] <- simon(0.99, (tot_a1+tot_a2), "NPV")$pub
results[2,2] <- simon(exp(coef(amadoCoxA)), amadoCoxA$nevent, "NPV")$pub

## PPV: Longer survival in panitumumab group for KRASwt (Fig2b)
Amado2b_IPD$os <- with(Amado2b_IPD, Surv(t, d))
results[1,3] <- round(kNonP(Amado2b_IPD)$est, 2)

## NPV: Longer survival in panitumumab group for KRASmut (Fig2a)
Amado2a_IPD$os <- with(Amado2a_IPD, Surv(t, d))
results[2,3] <- round(1 - kNonP(Amado2a_IPD)$est, 2)

kable(results, caption="Concordance measures for Amado (2008)")
```

Table 2: Concordance measures for Amado (2008)

	Simon	Simon_IPD	NonParametric K
PPV: Longer survival for Panitumumab in KRASwt	0.69 (0.63; 0.74)	0.68 (0.62; 0.74)	0.71
NPV: not Longer survival for Panitumumab in KRASmut	0.5 (0.42; 0.57)	0.51 (0.43; 0.58)	0.48

Mok 2009 - EGFR study

Data from (Mok et al. 2009) Fig2B and Fig2C was digitezed using DigitizeIT. These data are loaded below

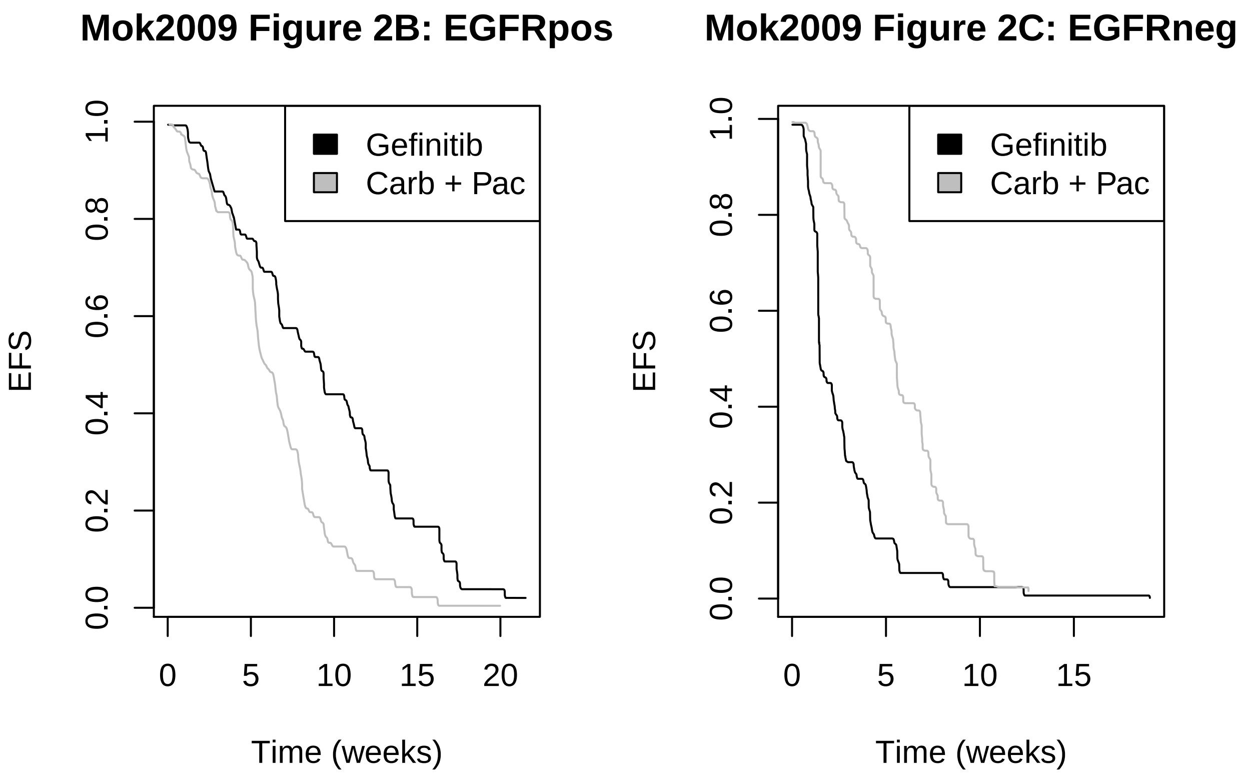
```
digitDir <- "../ExternalData/mok2009/"
mok_2bBlack <- read.table(file.path(digitDir, "2b_black.csv"),
                          skip = 1, header = F, sep = ";", dec = ",")
mok_2bGrey  <- read.table(file.path(digitDir, "2b_gray.csv"),
                          skip = 1, header = F, sep = ";", dec = ",")
mok_2cBlack <- read.table(file.path(digitDir, "2c_black.csv"),
                          skip = 1, header = F, sep = ";", dec = ",")
mok_2cGrey  <- read.table(file.path(digitDir, "2c_gray.csv"),
                          skip = 1, header = F, sep = ";", dec = ",")
```

Fix possible problems with data

```
## Sort by x-coordinate
mok_2bBlack <- mok_2bBlack[order(mok_2bBlack$V1), ]
mok_2bGrey  <- mok_2bGrey[order(mok_2bGrey$V1), ]
mok_2cBlack <- mok_2cBlack[order(mok_2cBlack$V1), ]
mok_2cGrey  <- mok_2cGrey[order(mok_2cGrey$V1), ]

## Fix problems with monotony of data
mok_2bBlack[,1] <- fixMonotonx(mok_2bBlack[,1])
mok_2bGrey[,1]  <- fixMonotonx(mok_2bGrey[,1])
mok_2bBlack[,2] <- fixMonotony(mok_2bBlack[,2])
mok_2bGrey[,2]  <- fixMonotony(mok_2bGrey[,2])
mok_2cBlack[,1] <- fixMonotonx(mok_2cBlack[,1])
mok_2cGrey[,1]  <- fixMonotonx(mok_2cGrey[,1])
mok_2cBlack[,2] <- fixMonotony(mok_2cBlack[,2])
mok_2cGrey[,2]  <- fixMonotony(mok_2cGrey[,2])
```

Plot digitezed data to verify it

```
par(mfrow=c(1,2))
plot(mok_2bBlack[,1], mok_2bBlack[,2], col = "black", type = "l",
     main = "Mok2009 Figure 2B: EGFRpos", xlab = "Time (weeks)", ylab = "EFS")
lines(mok_2bGrey[,1], mok_2bGrey[,2], col = "grey")
legend("topright", fill = c("black", "grey"), legend = c("Gefinitib", "Carb + Pac"))

plot(mok_2cBlack[,1], mok_2cBlack[,2], col = "black", type  ="l",
     main = "Mok2009 Figure 2C: EGFRneg", xlab = "Time (weeks)", ylab = "EFS")
lines(mok_2cGrey[,1], mok_2cGrey[,2], col = "grey")
legend("topright", fill = c("black", "grey"), legend = c("Gefinitib", "Carb + Pac"))
```



Specify data from risk tables

```r
risk_time <- seq(0, 24, by = 4)
mok_nrisk_bBlack <- riskTable(risk_time, c(132, 108, 71, 31, 11, 3, 0), mok_2bBlack)
mok_nrisk_bGrey  <- riskTable(risk_time, c(129, 103, 37, 7, 2, 1, 0), mok_2bGrey)
mok_nrisk_cBlack <- riskTable(risk_time, c(91,21,4,2,1,0,0), mok_2cBlack)
mok_nrisk_cGrey  <- riskTable(risk_time, c(85,58,14,1,0,0,0), mok_2cGrey)
```

Specify number of events

```r
mok_tot_b1 <-  97
mok_tot_b2 <- 111
mok_tot_c1 <-  88
mok_tot_c2 <-  70
```

Extract IPD data from digitized KM curves

```r
path <- "../GeneratedData/mok2009/"
b1 <- dataFromKMCurve(digizeit = mok_2bBlack,
                      nrisk = mok_nrisk_bBlack[!is.na(mok_nrisk_bBlack$lower), ],
                      KMdatafile = "/mok2009_b1_km.txt",
                      KMdataIPDfile = "/mok2009_b1_km_IPD.txt",
                      tot.events = mok_tot_b1,
                      arm.id = 2)

b2 <- dataFromKMCurve(digizeit = mok_2bGrey,
                      nrisk = mok_nrisk_bGrey[!is.na(mok_nrisk_bGrey$lower), ],
                      KMdatafile = "/mok2009_b2_km.txt",
                      KMdataIPDfile = "/mok2009_b2_km_IPD.txt",
                      tot.events = mok_tot_b2,
                      arm.id = 1)

c1 <- dataFromKMCurve(digizeit = mok_2cBlack,
                      nrisk = mok_nrisk_cBlack[!is.na(mok_nrisk_cBlack$lower), ],
                      KMdatafile = "/mok2009_c1_km.txt",
                      KMdataIPDfile = "/mok2009_c1_km_IPD.txt",
                      tot.events = mok_tot_c1,
                      arm.id = 2)

c2 <- dataFromKMCurve(digizeit = mok_2cGrey,
                      nrisk = mok_nrisk_cGrey[!is.na(mok_nrisk_cGrey$lower), ],
                      KMdatafile = "/mok2009_c2_km.txt",
                      KMdataIPDfile = "/mok2009_c2_km_IPD.txt",
                      tot.events = mok_tot_c2,
                      arm.id = 1)
```

Read IPD

```r
mok_2bBlack_IPD <- read.table(file.path(path, "mok2009_b1_km_IPD.txt"))
mok_2bGrey_IPD  <- read.table(file.path(path, "mok2009_b2_km_IPD.txt"))
mok_2cBlack_IPD <- read.table(file.path(path, "mok2009_c1_km_IPD.txt"))
mok_2cGrey_IPD  <- read.table(file.path(path, "mok2009_c2_km_IPD.txt"))
```

Create dataframes for IPD in Fig2B (true HR = 0.48) and Fig2C (true HR = 2.85), redo cox regressions to verify that they are similar to the original publication. HR in publication was adjusted WHO performance status (0 or 1, or 2), smoking history (nonsmoker, or former light smoker), and sex as covariates.

```
mok_2b_IPD <- rbind(mok_2bBlack_IPD, mok_2bGrey_IPD)
mok_2c_IPD <- rbind(mok_2cBlack_IPD, mok_2cGrey_IPD)

names(mok_2b_IPD) <- c("t", "d", "x")
names(mok_2c_IPD) <- c("t", "d", "x")

## Cox for fig 2b
MokCoxB <- coxph(with(mok_2b_IPD, Surv(t,d)~x))
MokCoxB
```

```
## Call:
## coxph(formula = with(mok_2b_IPD, Surv(t, d) ~ x))
##
##      coef exp(coef) se(coef)  z       p
## x -0.680     0.507    0.136 -5 5.7e-07
##
## Likelihood ratio test=25  on 1 df, p=5.75e-07
## n= 261, number of events= 231
```

```
## Cox for fig 2c
MokCoxC <- coxph(with(mok_2c_IPD, Surv(t,d)~x))
MokCoxC
```

```
## Call:
## coxph(formula = with(mok_2c_IPD, Surv(t, d) ~ x))
##
##     coef exp(coef) se(coef)    z       p
## x 1.041     2.831    0.163 6.37 1.9e-10
##
## Likelihood ratio test=40.3  on 1 df, p=2.2e-10
## n= 176, number of events= 164
```

Calculate PPV and NPV using method from (Simon 2015) from published HR in (Mok et al. 2009) and recalculated from IPD and the non-parametric estimate of PPV and NPV from IPD data.

```
results <- array(0, c(2,3))
rownames(results) <- c("PPV: Longer survival for Gefitinib in EGFRpos",
                       "NPV: not Longer survival for Gefitinib in EGFRneg")
colnames(results) <- c("Simon", "Simon_IPD","NonParametric K")


## Simon PPV from published or recalculated IPD data
results[1,1] <- simon(0.48, (mok_tot_b1+mok_tot_b2))$pub
results[1,2] <- simon(exp(coef(MokCoxB)), MokCoxB$nevent)$pub

## Simon NPV from published or recalculated IPD data
results[2,1] <- simon(2.85, (mok_tot_c1+mok_tot_c2), "NPV")$pub
results[2,2] <- simon(exp(coef(MokCoxC)), MokCoxC$nevent, "NPV")$pub

## PPV: Longer survival with Gefitinib for EGFRpos (Fig2b)
mok_2b_IPD$os <- with(mok_2b_IPD, Surv(t, d))
results[1,3] <- round(kNonP(mok_2b_IPD)$est, 2)

## NPV: not Longer survival with Gefitinib for EGFRpos (Fig2c)
mok_2c_IPD$os <- with(mok_2c_IPD, Surv(t, d))
results[2,3] <- round(1 - kNonP(mok_2c_IPD)$est, 2)
```

```r
kable(results, digits = 3, caption="Concordance measures for Mok (2009)")
```

Table 3: Concordance measures for Mok (2009)

	Simon	Simon_IPD	NonParametric K
PPV: Longer survival for Gefitinib in EGFRpos	0.68 (0.61; 0.73)	0.66 (0.6; 0.72)	0.67
NPV: not Longer survival for Gefitinib in EGFRneg	0.74 (0.68; 0.8)	0.74 (0.68; 0.79)	0.82

References

Amado, Rafael G., Michael Wolf, Marc Peeters, Eric Van Cutsem, Salvatore Siena, Daniel J. Freeman, Todd Juan, et al. 2008. "Wild-type KRAS is required for panitumumab efficacy in patients with metastatic colorectal cancer." *Journal of Clinical Oncology* 26 (10): 1626–34. doi:10.1200/JCO.2007.14.7116.

Guyot, Patricia, A. E. Ades, Mario J.N.M. Ouwens, and Nicky J. Welton. 2012. "Enhanced secondary analysis of survival data: Reconstructing the data from published Kaplan-Meier survival curves." *BMC Medical Research Methodology* 12 (1). BioMed Central Ltd: 9. doi:10.1186/1471-2288-12-9.

Mok, Tony S, Yi-long Wu, Sumitra Thongprasert, Chih-hsin Yang, Nagahiro Saijo, Patrapim Sunpaweravong, Baohui Han, et al. 2009. "Gefitinib or Carboplatin–Paclitaxel in Pulmonary Adenocarcinoma." *New England Journal of Medicine* 361 (10): 947–57. doi:10.1056/NEJMoa0810699.

Simon, Richard. 2015. "Sensitivity, Specificity, PPV, and NPV for Predictive Biomarkers." *Journal of the National Cancer Institute* 107 (8): djv153. doi:10.1093/jnci/djv153.