# NSWPCN Predictor Training

January 19, 2015

## 1 Preparation

```
library(survival)

## Loading required package:  splines

library(glmulti)

## Loading required package:  rJava

library(flexsurv)
library(randomForestSRC)

## Loading required package:  parallel
##
##  randomForestSRC 1.5.5
##
##  Type rfsrc.news() to see new features, changes, and bug fixes.
##

library(reshape2)
library(plyr)
library(ggplot2)

library(MASS)
library(boot)

##
## Attaching package:  'boot'
##
## The following object is masked from 'package:survival':
##
##     aml

library(timeROC)

## Loading required package:  pec
## Loading required package:  mvtnorm
## Loading required package:  timereg

source("stdca.R")

load("03_NSWPCN_subset.rda")
```

# 2 Cohort selection and transformation

```
x = data[,c("Patient.Sex", "History.Diagnosis.AgeAt.Cent", "Path.LocationBody", "Path.Size.Cent", "Path.
colnames(x) = c("SexM", "AgeCent", "LocBody", "SizeCent", "Ca199", "A2", "A4")
x$SexM = x$Sex == "M"
x$Ca199 = x$Ca199 > 100

y = Surv(as.numeric(data$History.Death.Date - data$History.Diagnosis.Date), data$History.DSDeath.Event)
# Note no surgery dates, though for almost all pts there were only a few days difference.

temp = NA
temp = ls()
rm(list = temp[!(temp %in% c("x", "y"))])

sel = !is.na(y[,1]) & !is.na(y[,2]) & !is.na(x$A2) & !is.na(x$A4) & !is.na(x$LocBody)
x = x[sel,]
y = y[sel,]
rm(sel)

# Remove CA-19-9 measurements as they're mostly missing
x = x[,colnames(x) != "Ca199"]

data = as.data.frame(cbind(Time = y[,1], DSD = y[,2], x))
rm(x, y)
data$DSD = data$DSD == 1
```

# 3 Data splitting

There's going to be an awful lot of model manipulation and black magic going on. Create a holdout validation set for final model comparison and selection.

```
set.seed(20150110)
sel.val = sample.int(nrow(data), floor(nrow(data)/4))
sel.val = 1:nrow(data) %in% sel.val
mean(sel.val)

## [1] 0.25

data.val = data[sel.val,,drop = FALSE]
data = data[!sel.val,,drop = FALSE]
```

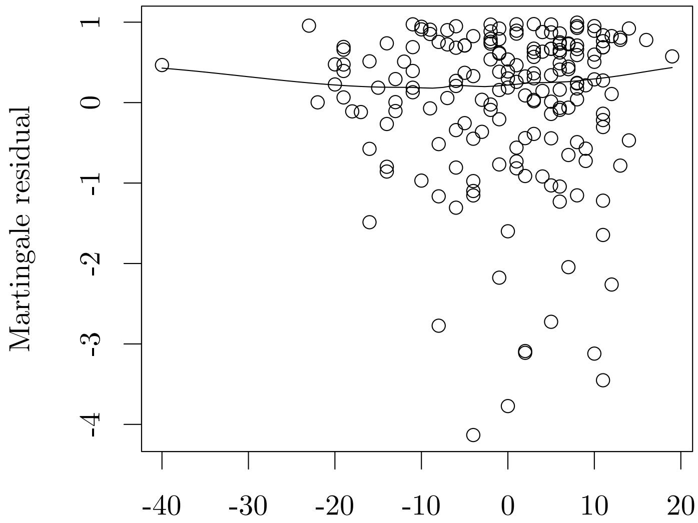
# 4 EDA

Use the CPH model as a convenient framework for EDA.

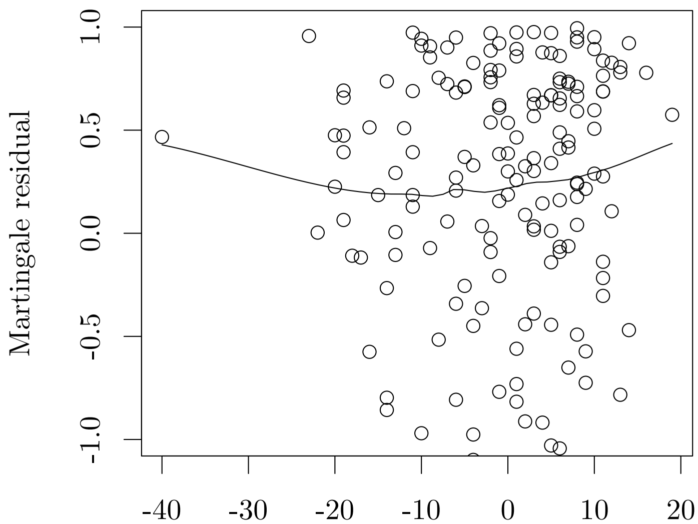
## 4.1 Functional form

Investigate functional form with martingale residuals.

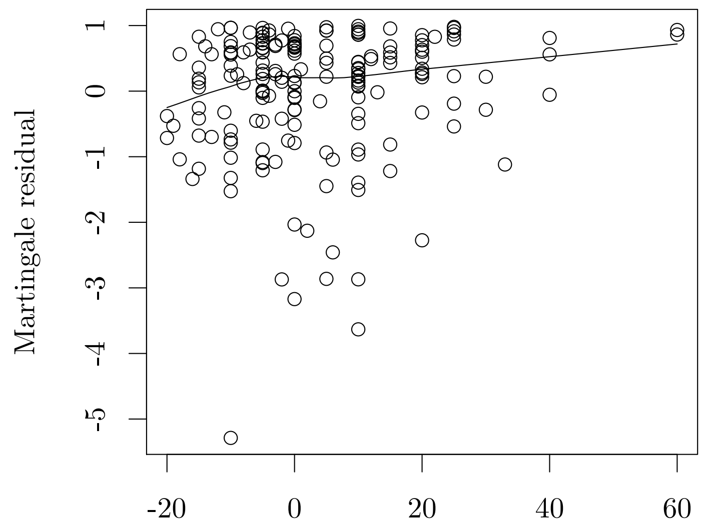
```
fit.cph.NoAge = coxph(Surv(Time, DSD) ~ SexM + LocBody + SizeCent + A2 + A4, data = data)
scatter.smooth(data$AgeCent, resid(fit.cph.NoAge, type = "martingale"), xlab = "", ylab = "Martingale re
```



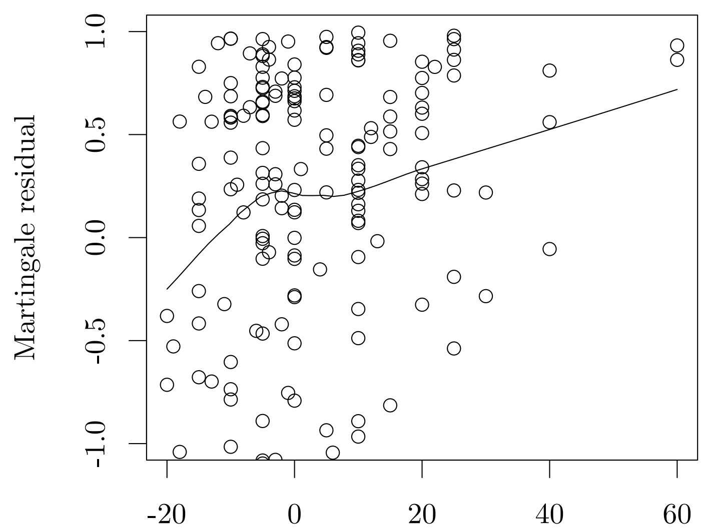

```
scatter.smooth(data$AgeCent, resid(fit.cph.NoAge, type = "martingale"), xlab = "", ylab = "Martingale re
```




```
fit.cph.NoSize = coxph(Surv(Time, DSD) ~ SexM + AgeCent + LocBody + A2 + A4, data = data)
scatter.smooth(data$SizeCent, resid(fit.cph.NoSize, type = "martingale"), xlab = "", ylab = "Martingale
```




```
scatter.smooth(data$SizeCent, resid(fit.cph.NoSize, type = "martingale"), xlab = "", ylab = "Martingale
```




It looks like age has a minor nonlinear component, leading to a quadratic-like U shape. The size relationship appears to have a knee, close to size == 0, around which the relationship is approximately linear.

Model age as: $AgeCent + AgeCent^2$ Model size as: $SizeCent + SizeCentI(SizeCent > 0) \equiv SizeCent + SizeCent_+$

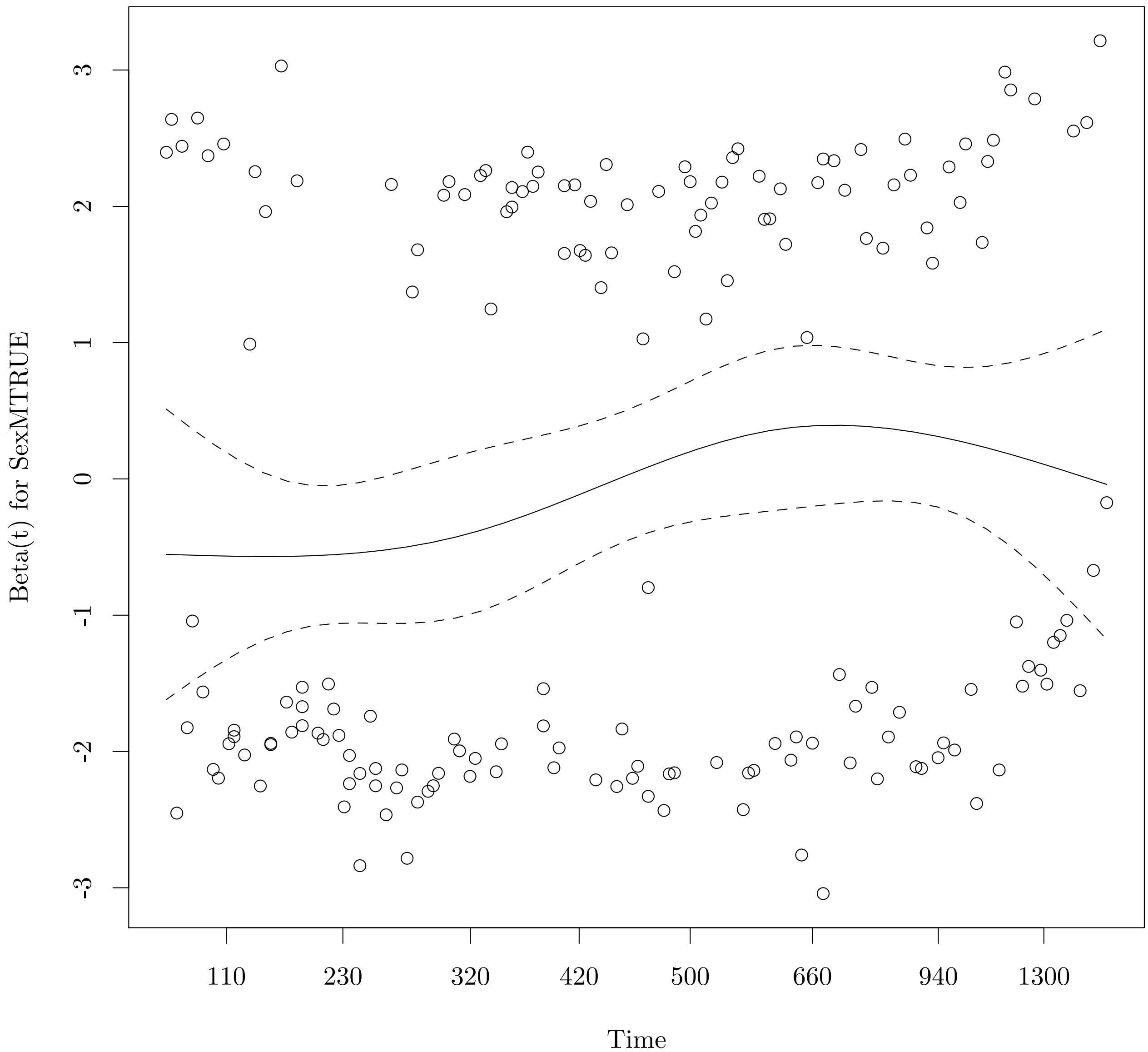
```
data$SizeSmall = data$SizeCent * (data$SizeCent < 0)
data$AgeCent2 = data$AgeCent^2
```

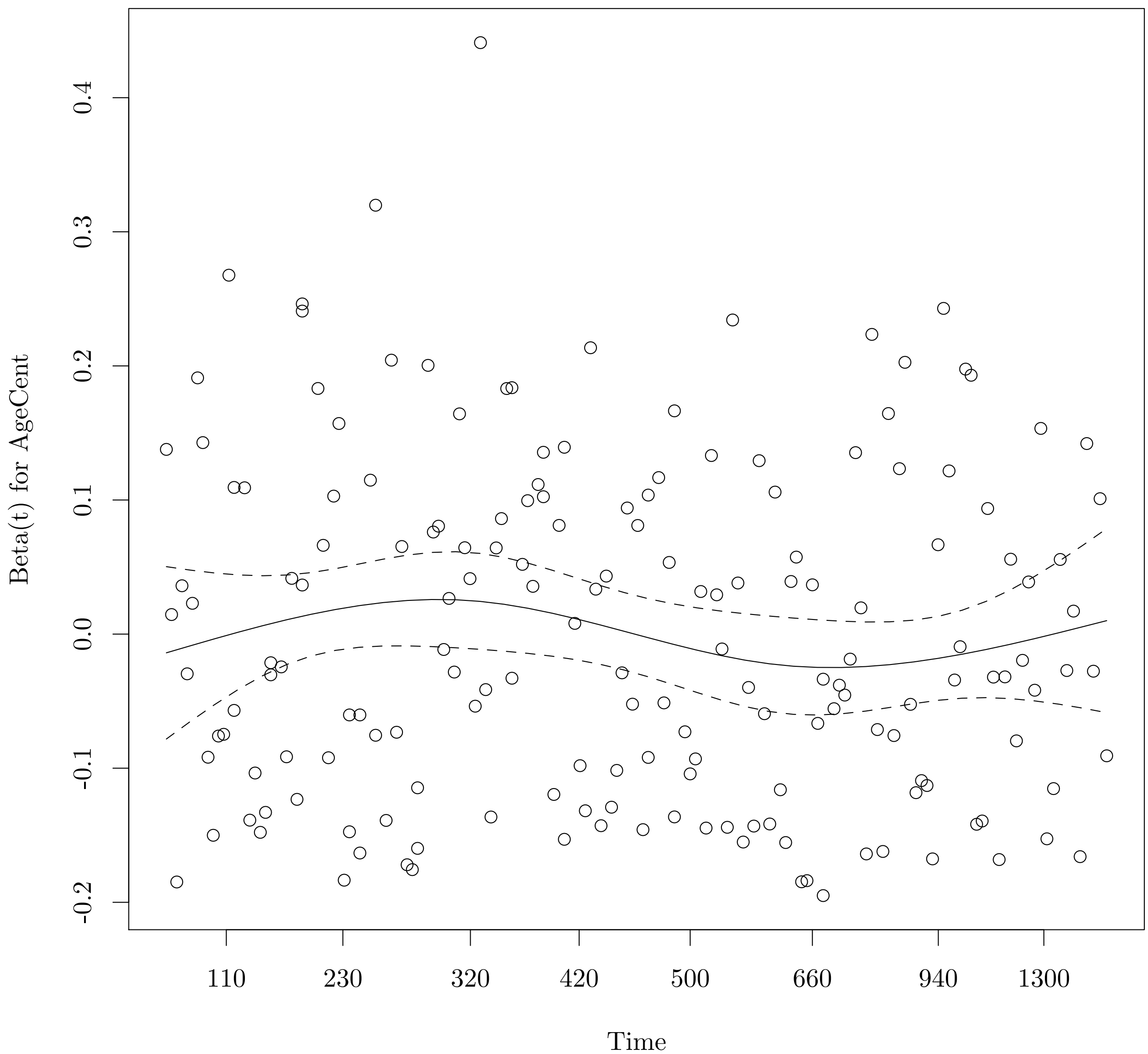
## 4.2 PH assumption: full model

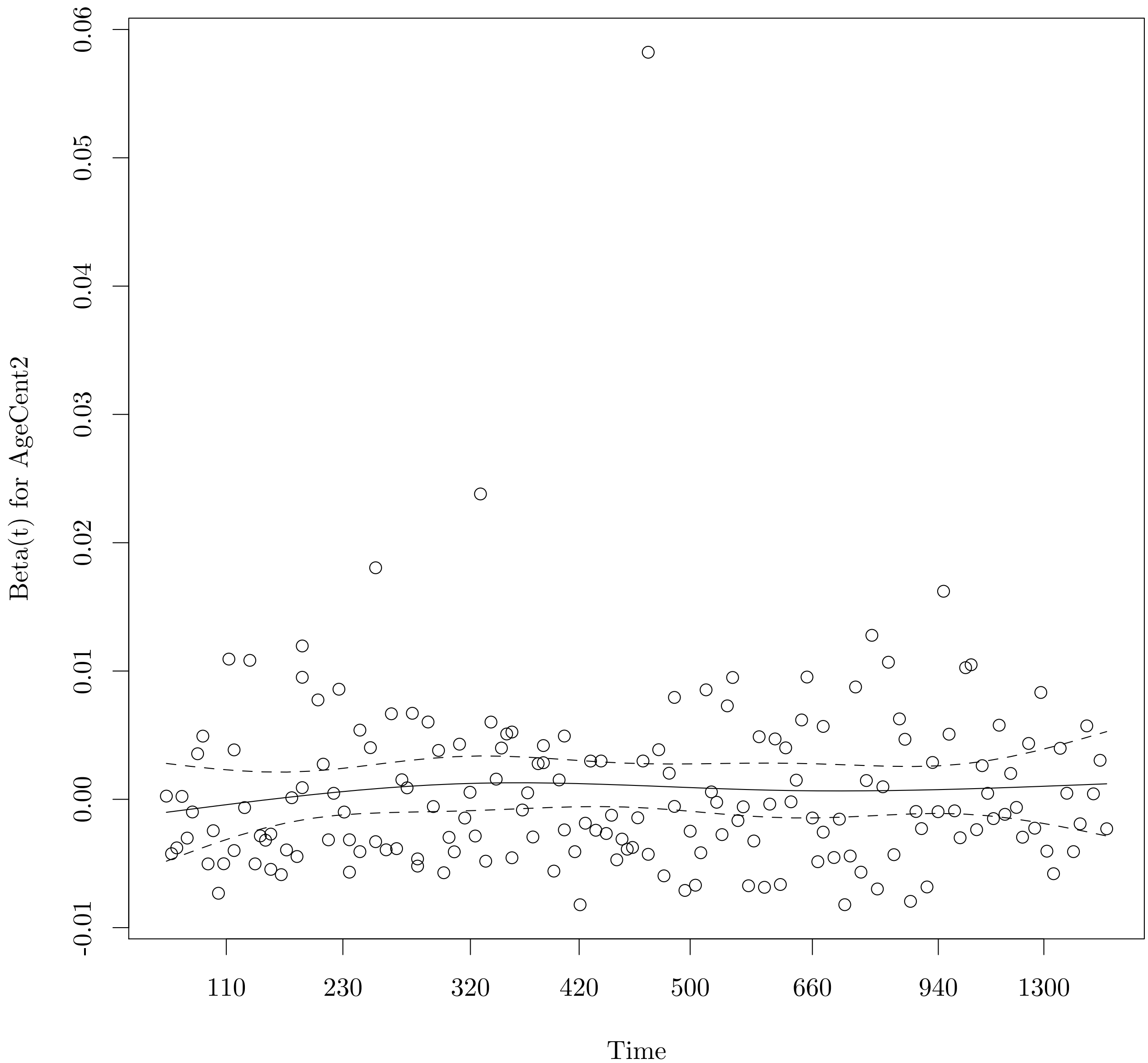
```
fit.cph = coxph(Surv(Time, DSD) ~ SexM + AgeCent + AgeCent2 + LocBody + SizeCent + SizeSmall + A2 + A4,
cox.zph(fit.cph)

##                rho  chisq      p
## SexMTRUE    0.1571 4.2500 0.0393
## AgeCent    -0.0746 0.9839 0.3212
## AgeCent2    0.0391 0.2379 0.6258
```

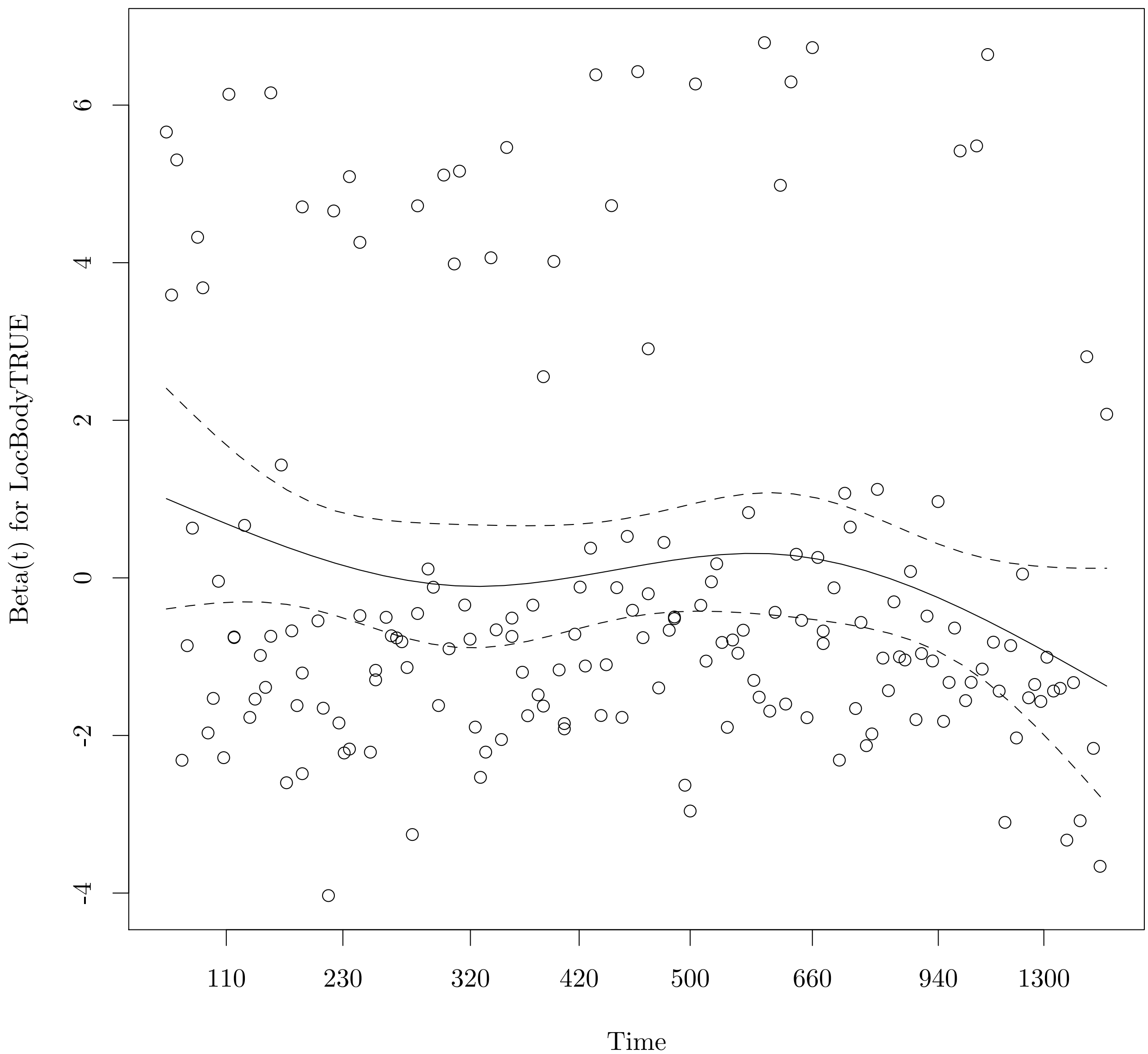
```
## LocBodyTRUE -0.1295  2.6406 0.1042
## SizeCent     0.0074  0.0112 0.9157
## SizeSmall   -0.0575  0.6037 0.4372
## A2TRUE       0.0447  0.3555 0.5510
## A4TRUE      -0.0493  0.4172 0.5183
## GLOBAL           NA 12.8286 0.1179
```

```r
plot(cox.zph(fit.cph))
```

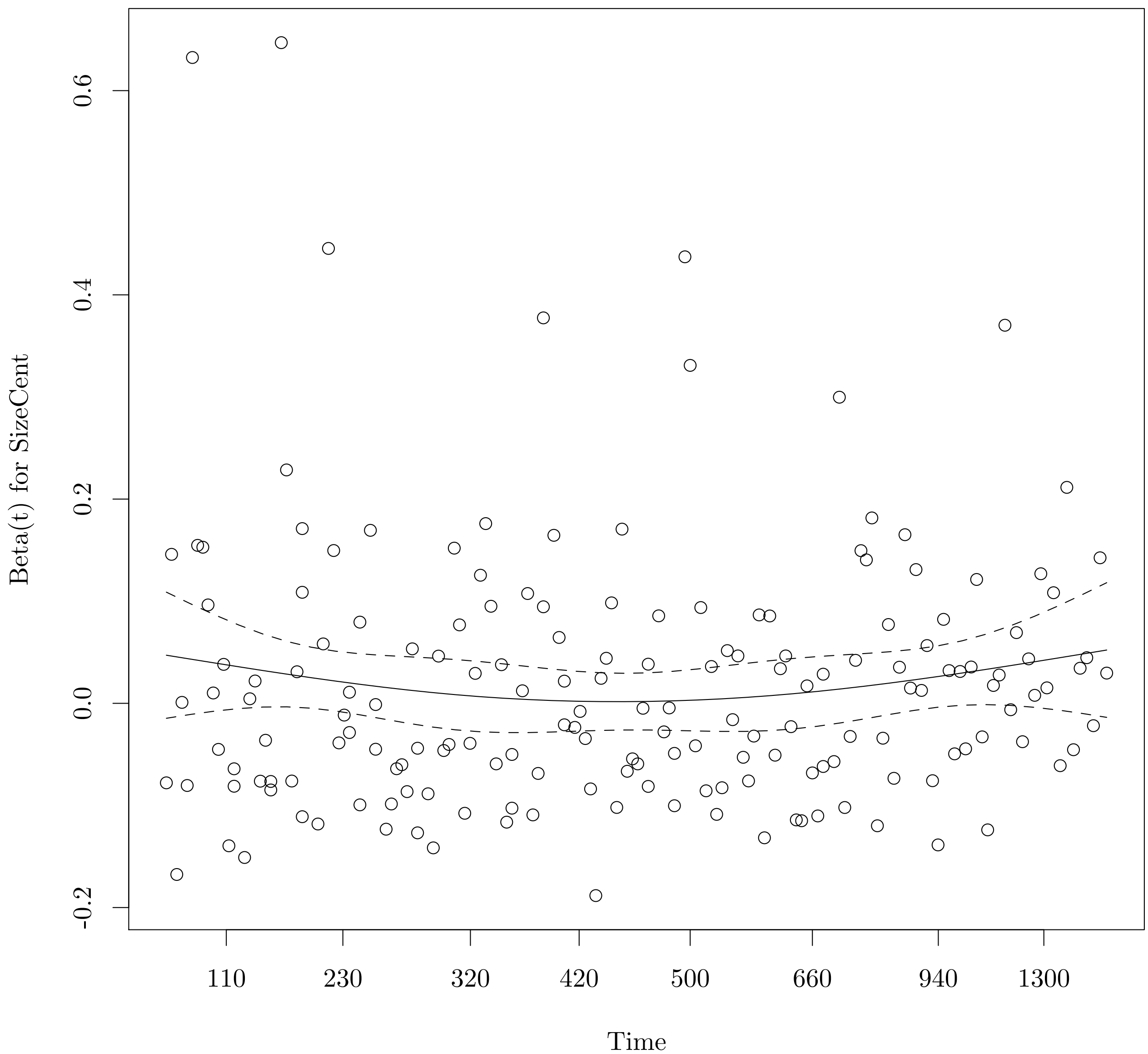



Beta(t) for AgeCent
0.4
0.3
0.2
0.1
0.0
-0.1
-0.2
110
230
320
420
500
660
940
1300
Time

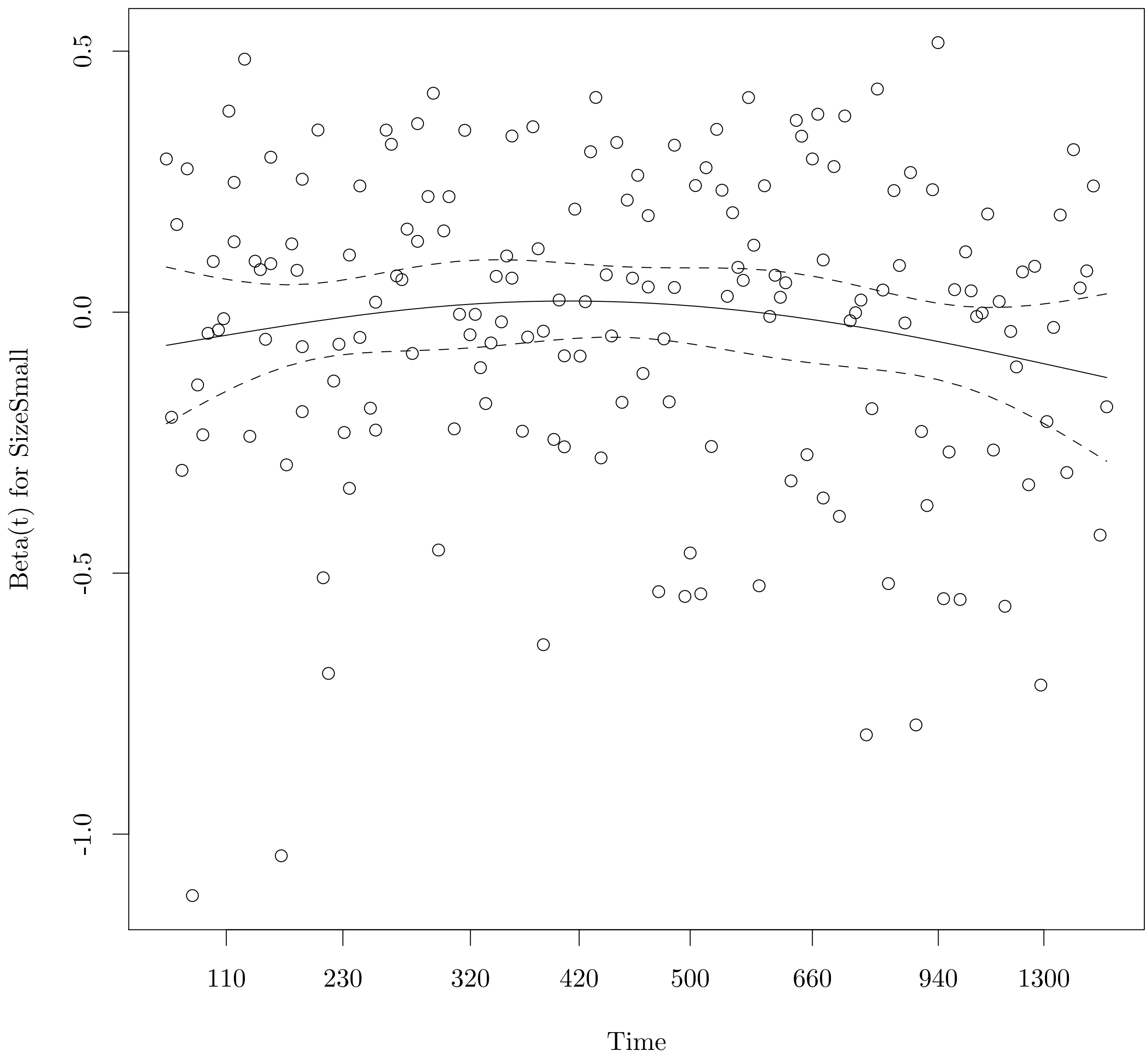

Beta(t) for AgeCent2
0.06
0.05
0.04
0.03
0.02
0.01
0.00
-0.01
110
230
320
420
500
660
940
1300
Time



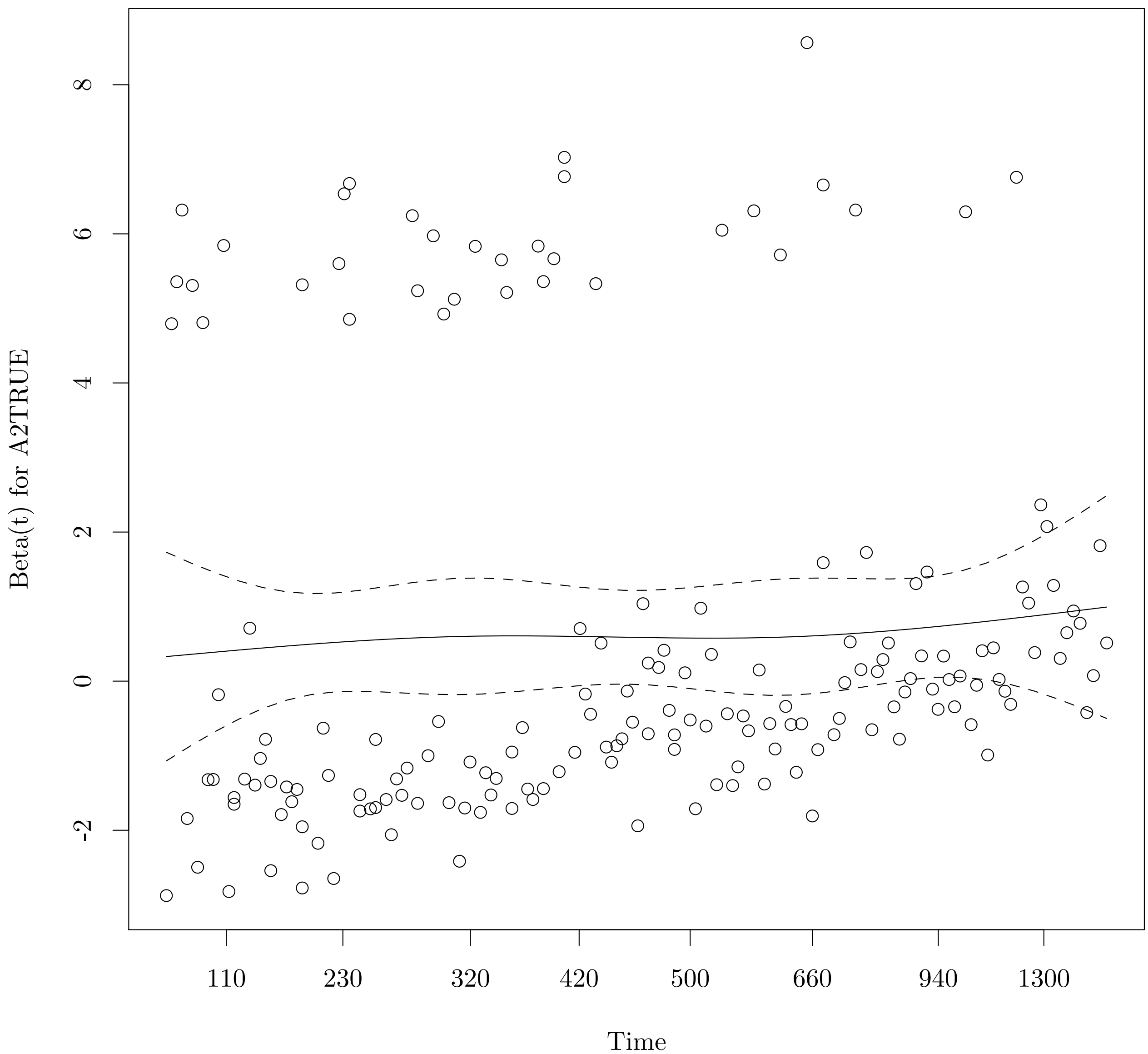
Beta(t) for LocBodyTRUE
6
4
2
0
-2
-4
110
230
320
420
500
660
940
1300
Time



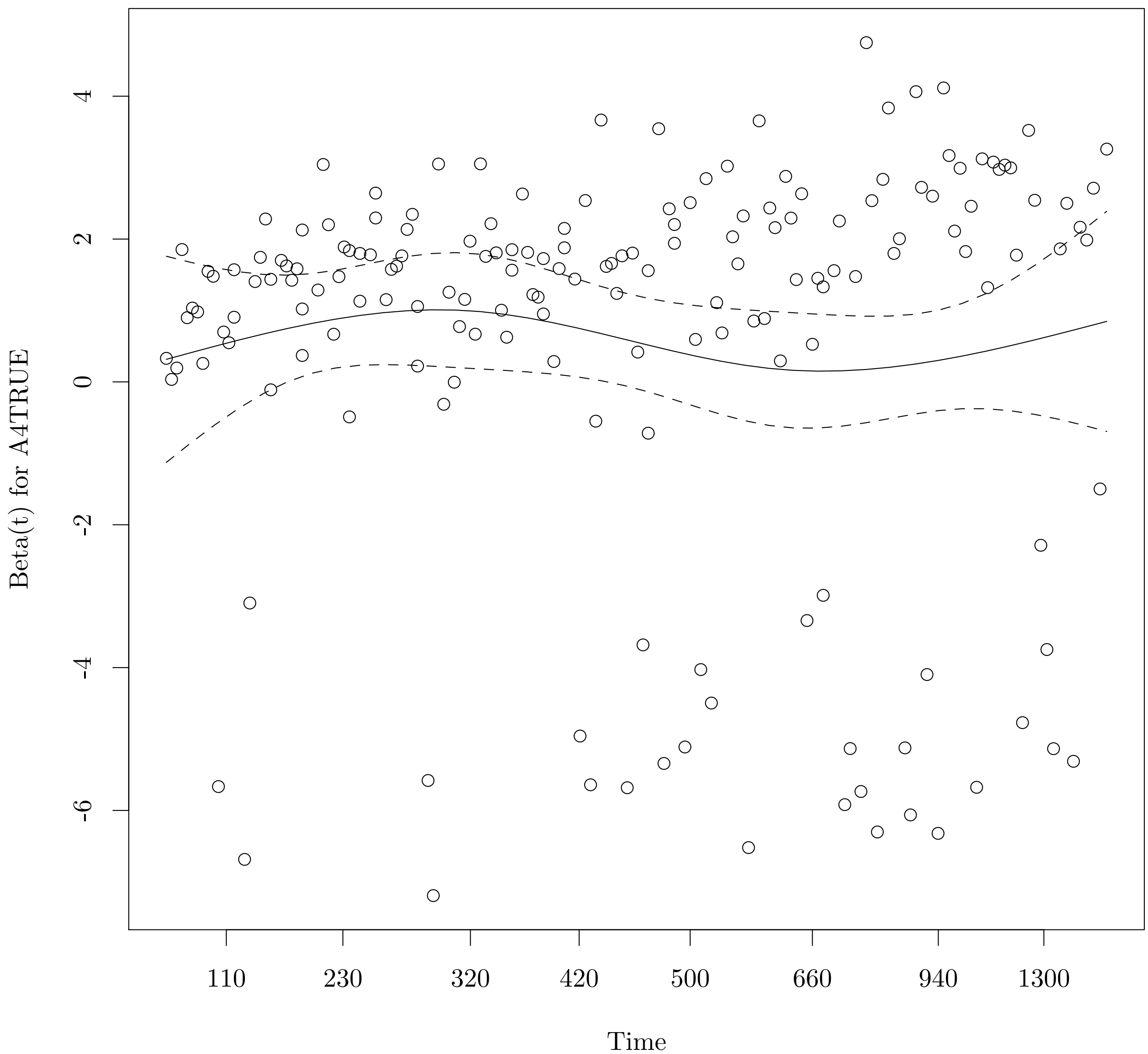
Beta(t) for SizeCent
0.6
0.4
0.2
0.0
-0.2
110
230
320
420
500
660
940
1300
Time



0.5
0.0
-0.5
-1.0
Beta(t) for SizeSmall
110
230
320
420
500
660
940
1300
Time



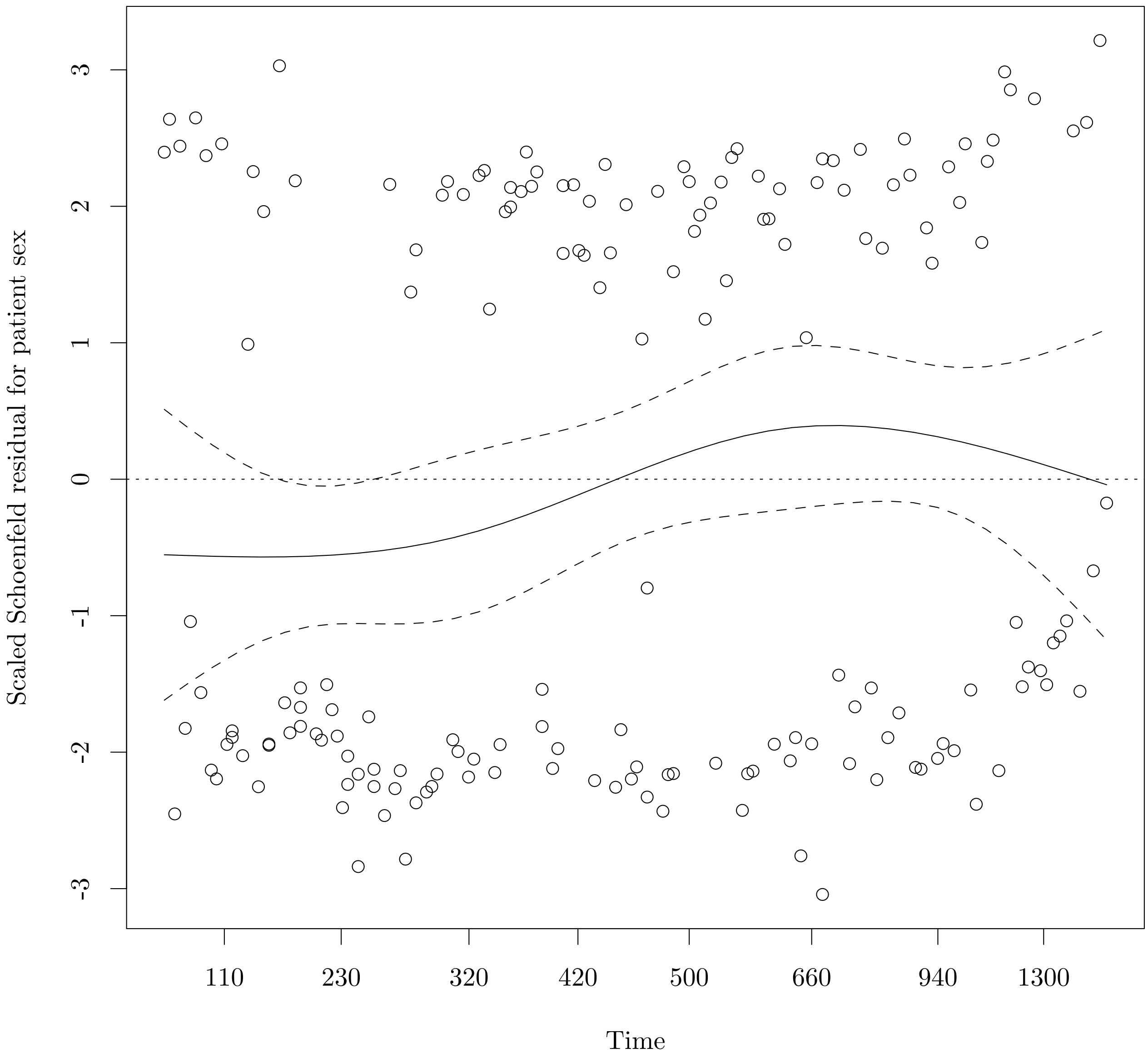
Beta(t) for A2TRUE
8
6
4
2
0
-2
110
230
320
420
500
660
940
1300
Time




```
temp = function (x, resid = TRUE, se = TRUE, df = 4, nsmo = 40, var, ...) {
    xx <- x$x
    yy <- x$y
    d <- nrow(yy)
    df <- max(df)
    nvar <- ncol(yy)
    pred.x <- seq(from = min(xx), to = max(xx), length = nsmo)
    temp <- c(pred.x, xx)
    lmat <- ns(temp, df = df, intercept = TRUE)
    pmat <- lmat[1:nsmo, ]
    xmat <- lmat[-(1:nsmo), ]
    qmat <- qr(xmat)
    if (qmat$rank < df)
        stop("Spline fit is singular, try a smaller degrees of freedom")
    if (se) {
        bk <- backsolve(qmat$qr[1:df, 1:df], diag(df))
        xtx <- bk %*% t(bk)
```

```
        seval <- d * ((pmat %*% xtx) * pmat) %*% rep(1, df)
    }
    ylab <- paste("Beta(t) for", dimnames(yy)[[2]])
    if (missing(var))
        var <- 1:nvar
    else {
        if (is.character(var))
            var <- match(var, dimnames(yy)[[2]])
        if (any(is.na(var)) || max(var) > nvar || min(var) <
            1)
            stop("Invalid variable requested")
    }
    if (x$transform == "log") {
        xx <- exp(xx)
        pred.x <- exp(pred.x)
    }
    else if (x$transform != "identity") {
        xtime <- as.numeric(dimnames(yy)[[1]])
        indx <- !duplicated(xx)
        apr1 <- approx(xx[indx], xtime[indx], seq(min(xx), max(xx),
            length = 17)[2 * (1:8)])
        temp <- signif(apr1$y, 2)
        apr2 <- approx(xtime[indx], xx[indx], temp)
        xaxisval <- apr2$y
        xaxislab <- rep("", 8)
        for (i in 1:8) xaxislab[i] <- format(temp[i])
    }
    for (i in var) {
        y <- yy[, i]
        yhat <- pmat %*% qr.coef(qmat, y)
        if (resid)
            yr <- range(yhat, y)
        else yr <- range(yhat)
        if (se) {
            temp <- 2 * sqrt(x$var[i, i] * seval)
            yup <- yhat + temp
            ylow <- yhat - temp
            yr <- range(yr, yup, ylow)
        }
        if (x$transform == "identity")
            plot(range(xx), yr, type = "n", ...)
        else if (x$transform == "log")
            plot(range(xx), yr, type = "n", log = "x", ...)
        else {
            plot(range(xx), yr, type = "n", axes = FALSE, ...)
            axis(1, xaxisval, xaxislab)
            axis(2)
            box()
        }
        if (resid)
            points(xx, y)
        lines(pred.x, yhat)
        if (se) {
```

```
            lines(pred.x, yup, lty = 2)
            lines(pred.x, ylow, lty = 2)
        }
    }
}

temp(cox.zph(fit.cph), var = 1, ylab = "Scaled Schoenfeld residual for patient sex", xlab = "Time")
abline(h = 0, lty = "dotted")
```



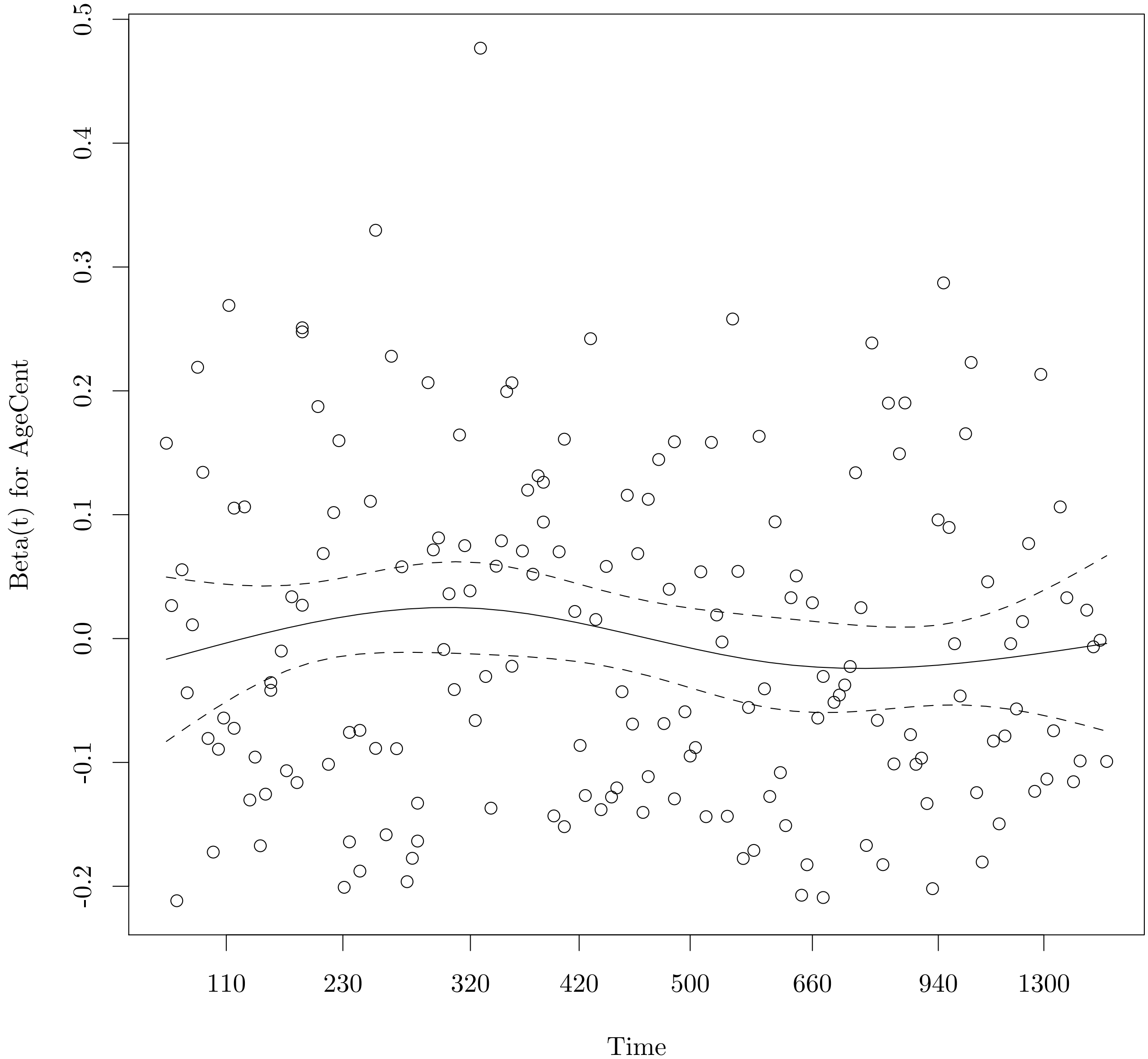

Looks like there's a violation of CPH with gender. Not unexpected. First check whether there is any evidence of gender interaction.

```
anova(coxph(Surv(Time, DSD) ~ SexM*(AgeCent + AgeCent2 + LocBody + SizeCent + SizeSmall + A2 + A4), data

## Analysis of Deviance Table
##  Cox model: response is Surv(Time, DSD)
## Terms added sequentially (first to last)
##
```

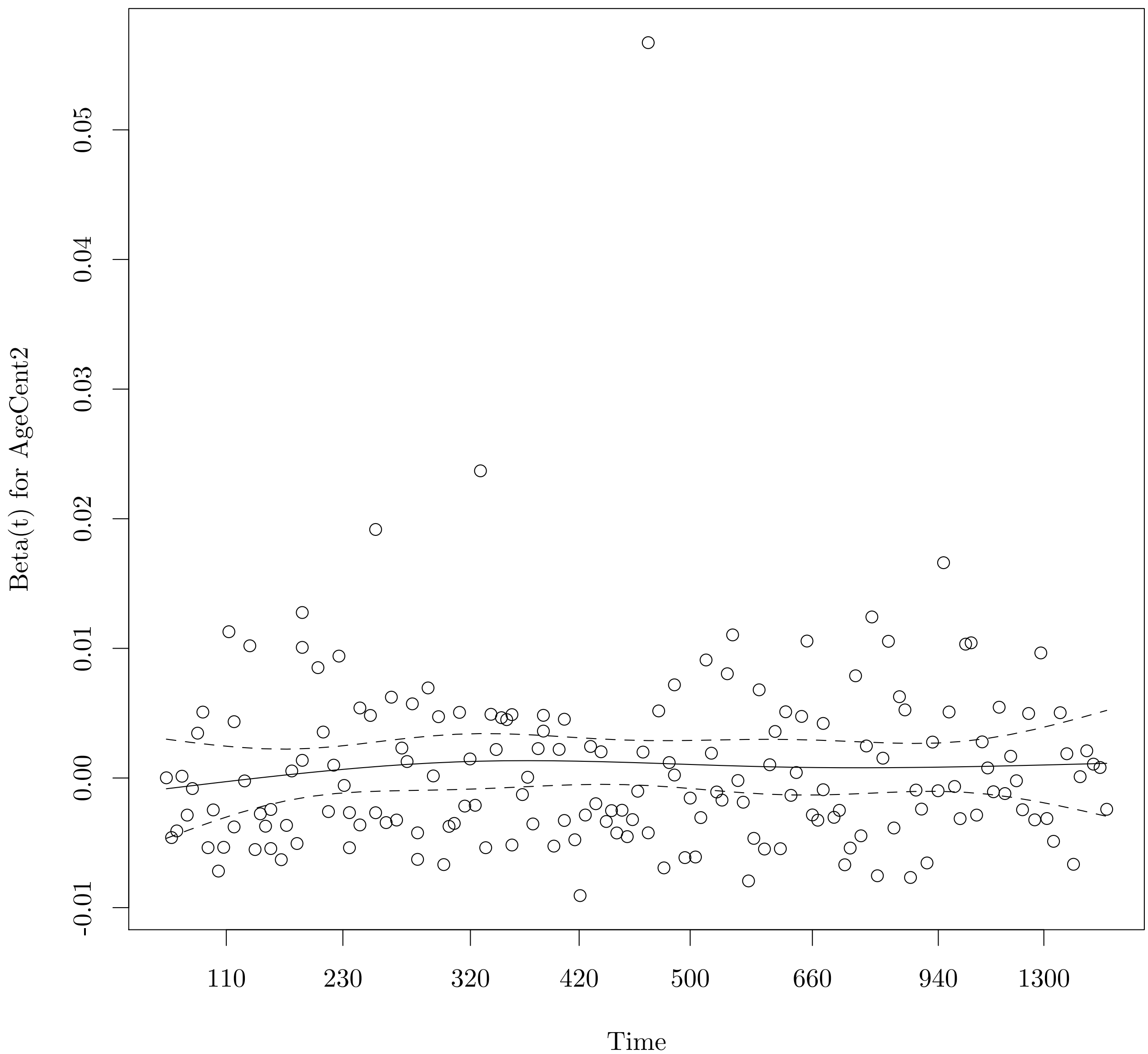
```
##                 loglik Chisq Df Pr(>|Chi|)
## NULL              -748
## SexM              -748  0.51  1     0.4762
## AgeCent           -747  0.19  1     0.6625
## AgeCent2          -747  0.81  1     0.3694
## LocBody           -746  2.40  1     0.1215
## SizeCent          -742  6.82  1     0.0090
## SizeSmall         -742  0.00  1     0.9563
## A2                -738  9.50  1     0.0021
## A4                -734  8.18  1     0.0042
## SexM:AgeCent      -733  0.37  1     0.5408
## SexM:AgeCent2     -733  0.17  1     0.6822
## SexM:LocBody      -733  0.09  1     0.7654
## SexM:SizeCent     -733  0.35  1     0.5568
## SexM:SizeSmall    -733  0.06  1     0.8068
## SexM:A2           -733  0.00  1     0.9588
## SexM:A4           -733  0.06  1     0.8000
```

Nope, good. We're not interested in gender effects so just stratify.

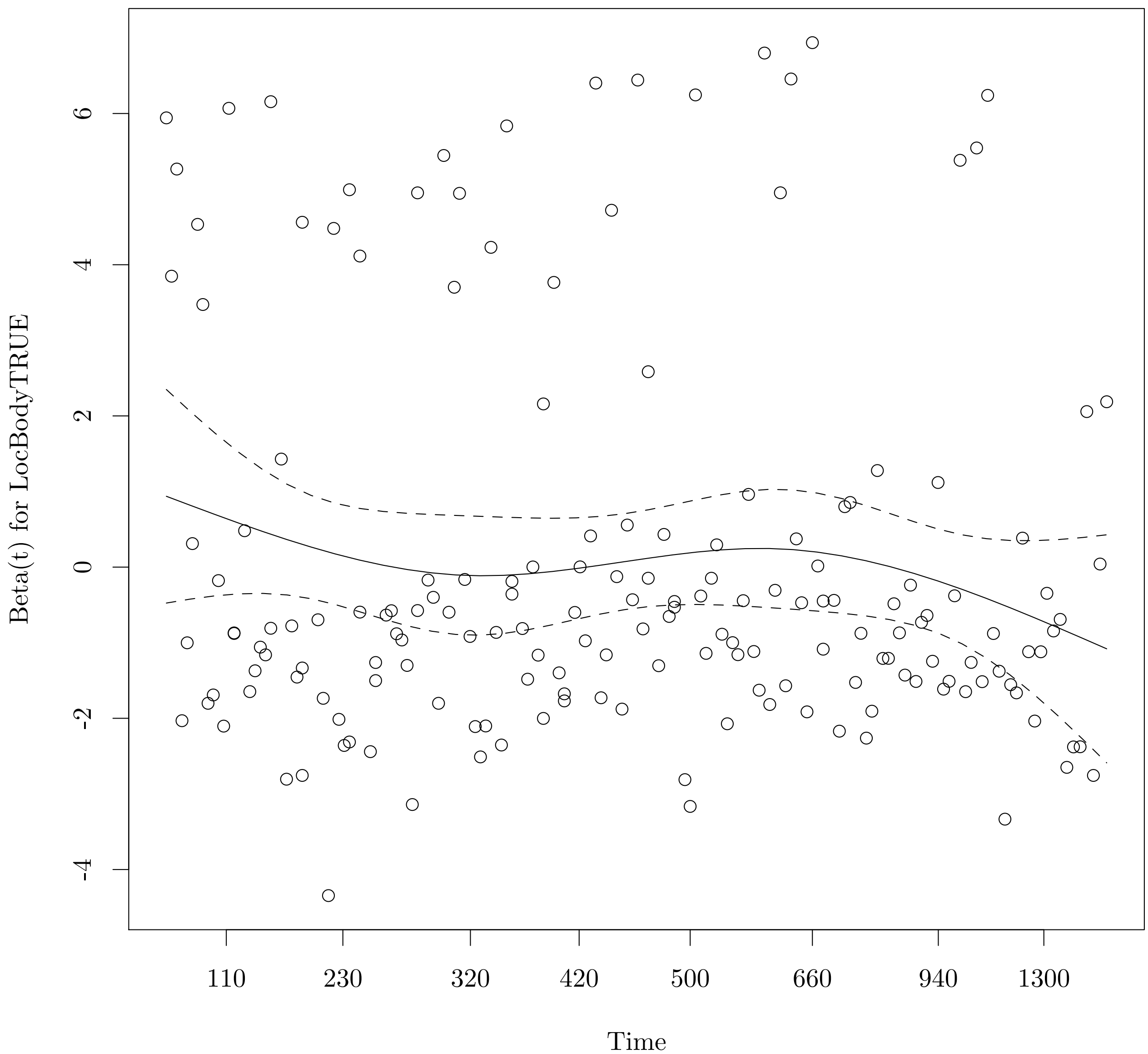
```
fit.cph = coxph(Surv(Time, DSD) ~ strata(SexM) + AgeCent + AgeCent2 + LocBody + SizeCent + SizeSmall + A
cox.zph(fit.cph)

##                  rho   chisq     p
## AgeCent     -0.07987 1.18163 0.277
## AgeCent2     0.03673 0.20762 0.649
## LocBodyTRUE -0.10954 1.84364 0.175
## SizeCent    -0.00689 0.00961 0.922
## SizeSmall   -0.04493 0.36247 0.547
## A2TRUE       0.04775 0.40111 0.527
## A4TRUE      -0.05491 0.51655 0.472
## GLOBAL            NA 6.85340 0.444

plot(cox.zph(fit.cph))
```



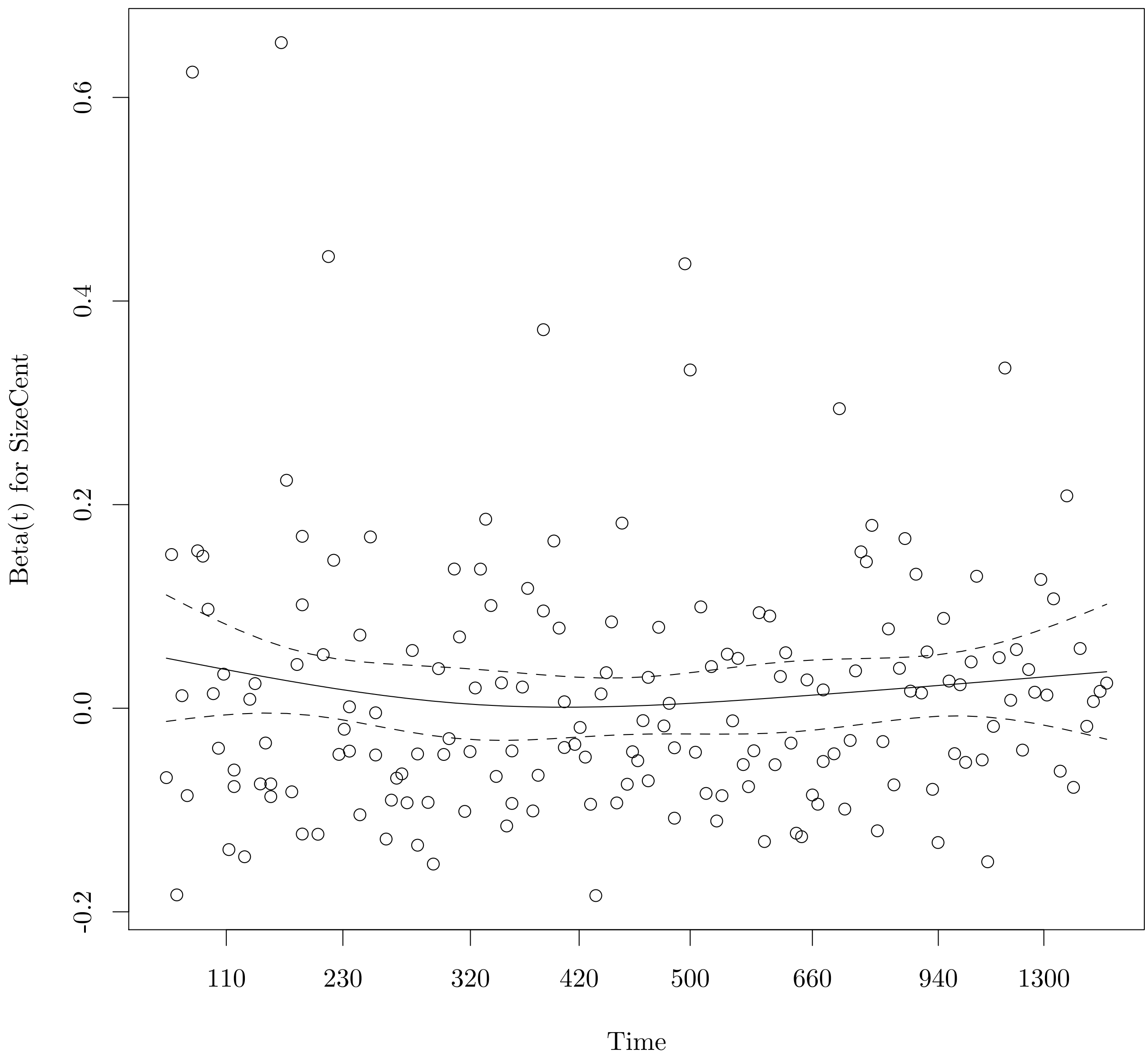
Beta(t) for AgeCent
0.5
0.4
0.3
0.2
0.1
0.0
-0.1
-0.2
110
230
320
420
500
660
940
1300
Time



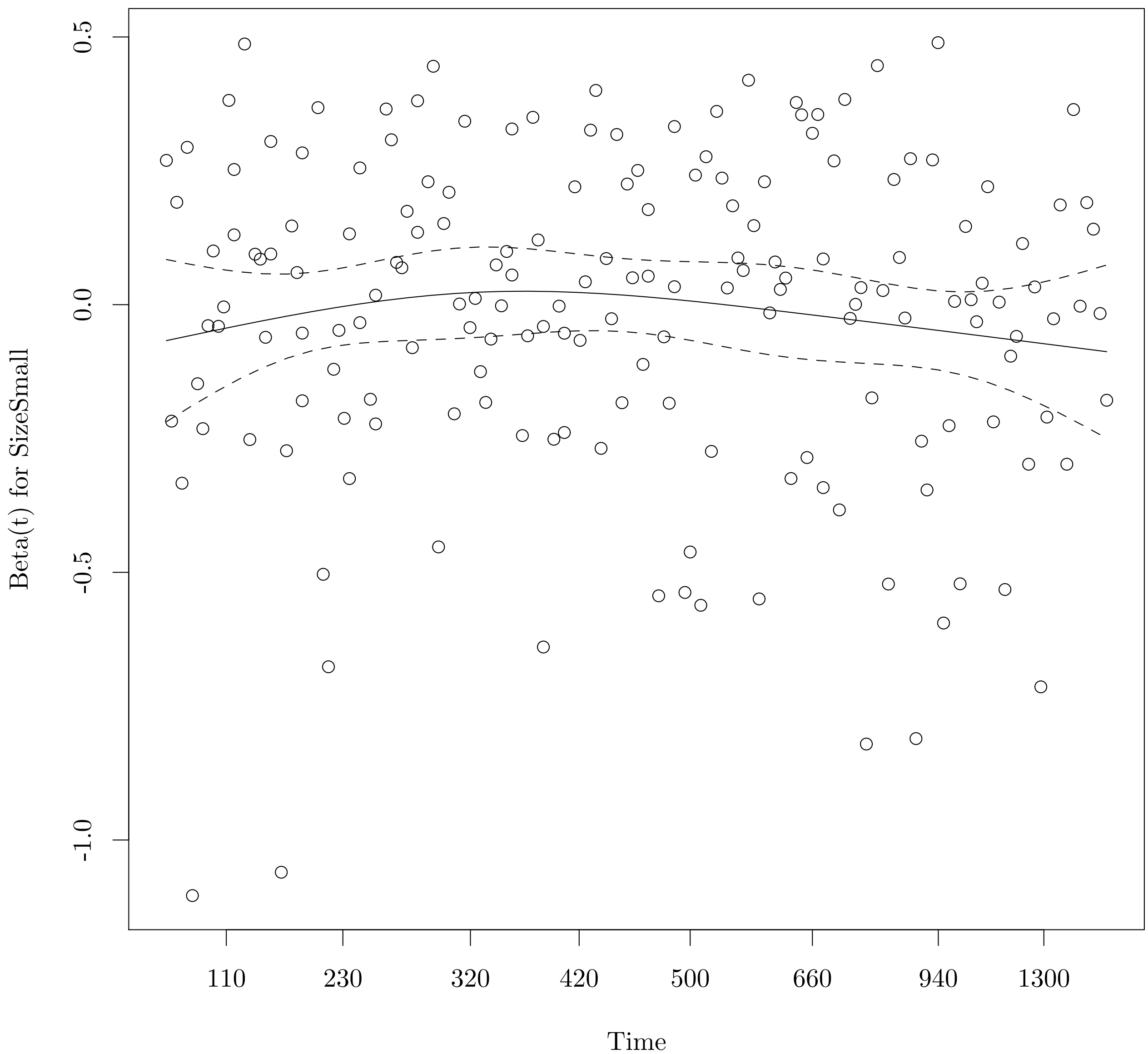
Beta(t) for AgeCent2
0.05
0.04
0.03
0.02
0.01
0.00
-0.01
110
230
320
420
500
660
940
1300
Time



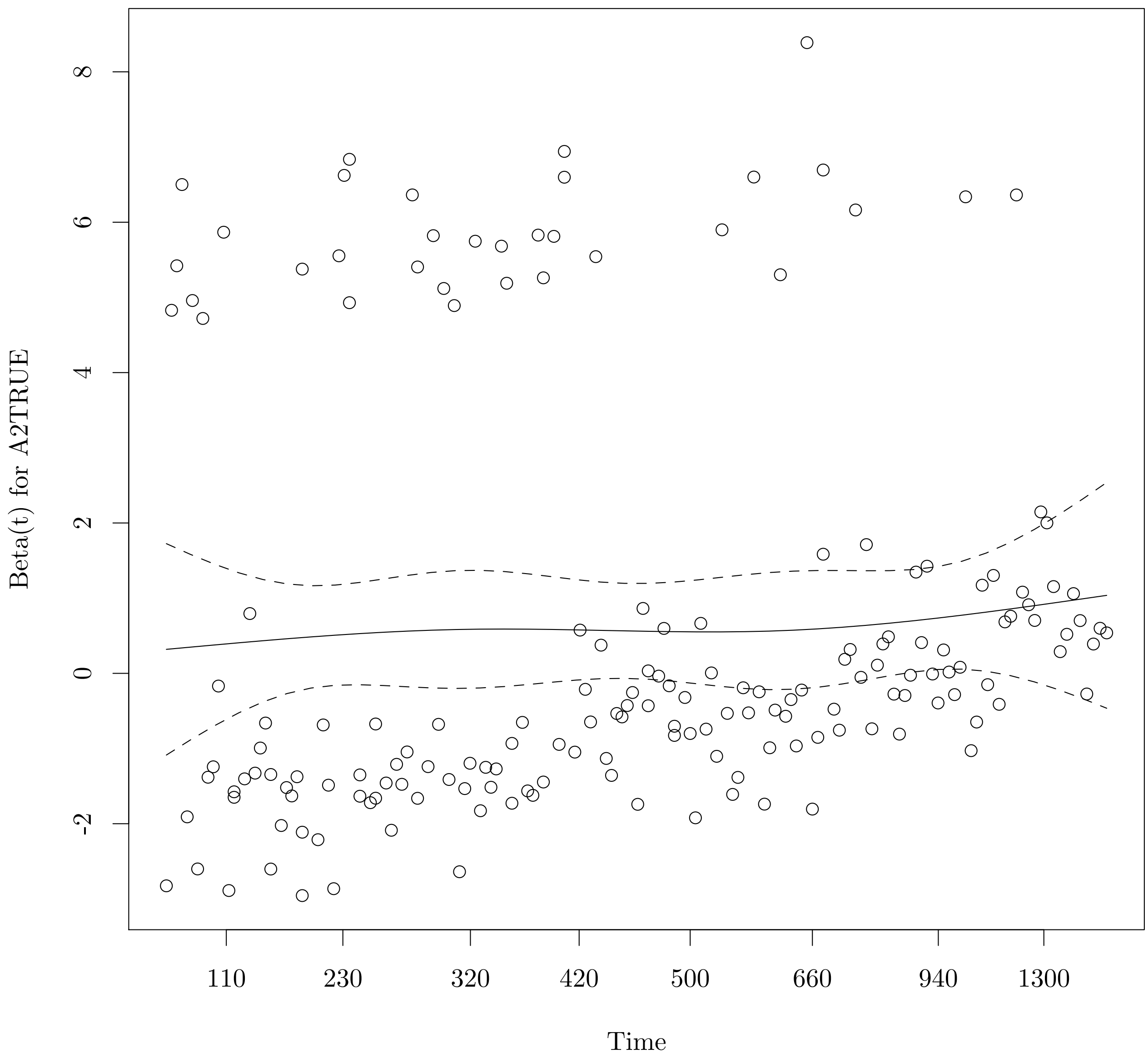
Beta(t) for LocBodyTRUE
6
4
2
0
-2
-4
110
230
320
420
500
660
940
1300
Time

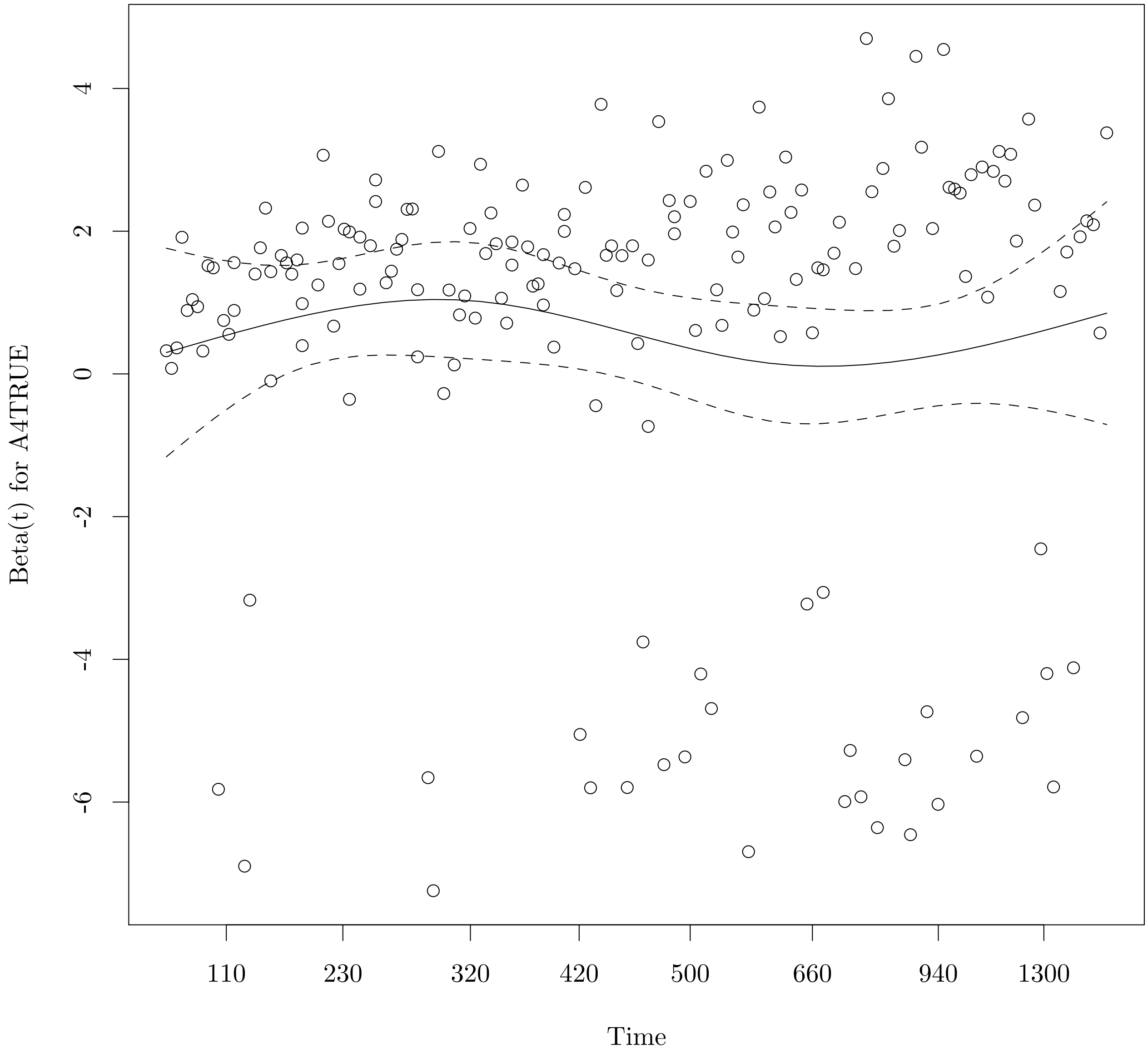


Beta(t) for SizeCent
0.6
0.4
0.2
0.0
-0.2
110
230
320
420
500
660
940
1300
Time



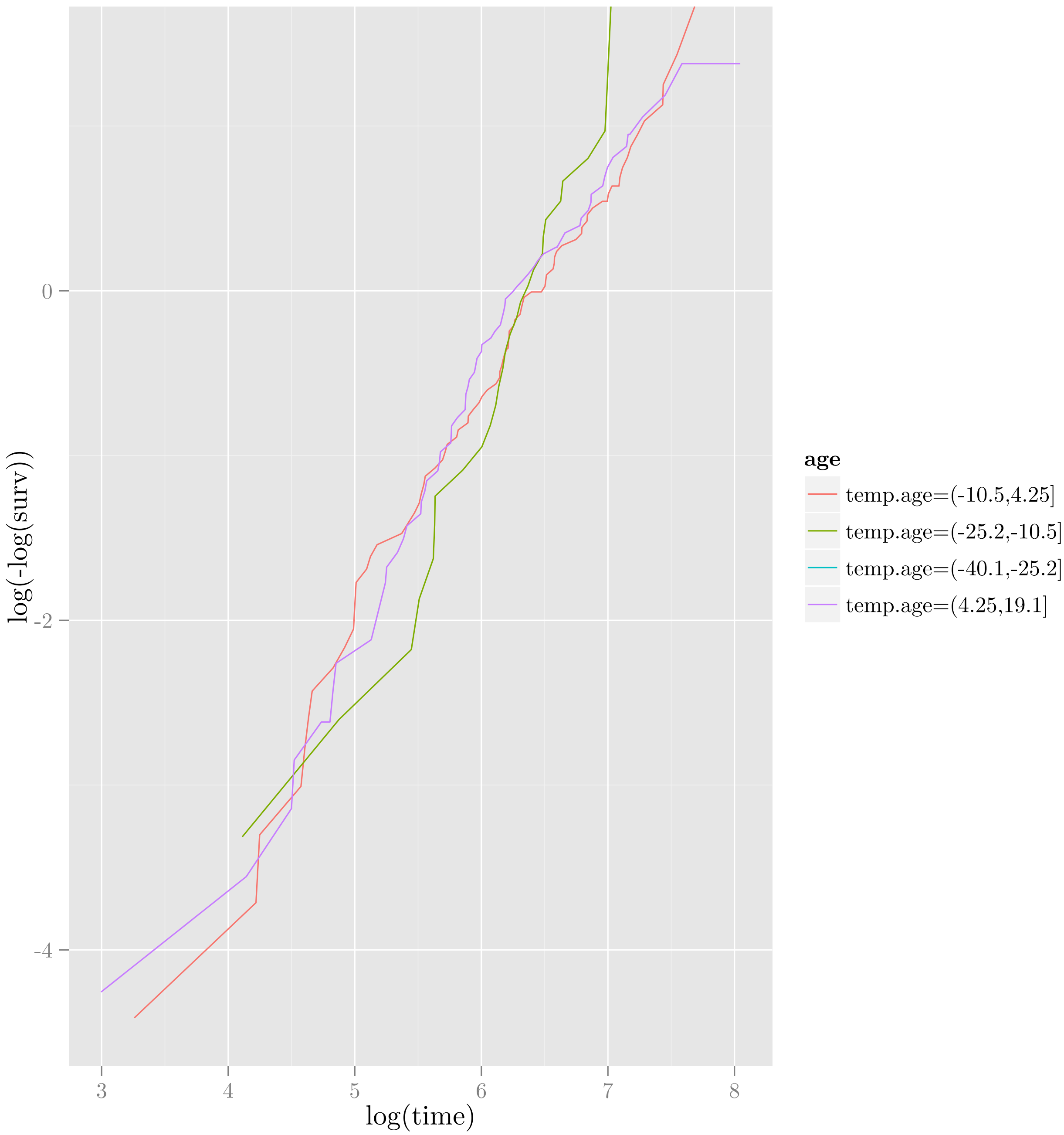
Beta(t) for SizeSmall
0.5
0.0
-0.5
-1.0
110
230
320
420
500
660
940
1300
Time


Beta(t) for A2TRUE
-2
0
2
4
6
8
110
230
320
420
500
660
940
1300
Time




Looks good. Slight snifter with age but I'm not particularly concerned. Split into age groups and do KM plots to verify.

```
temp.age = cut(data$AgeCent, 4)
temp = survfit(Surv(Time, DSD) ~ temp.age, data)
ggplot(data.frame(surv = temp$surv, time = temp$time, age = rep(names(temp$strata), temp$strata)), aes(y
```

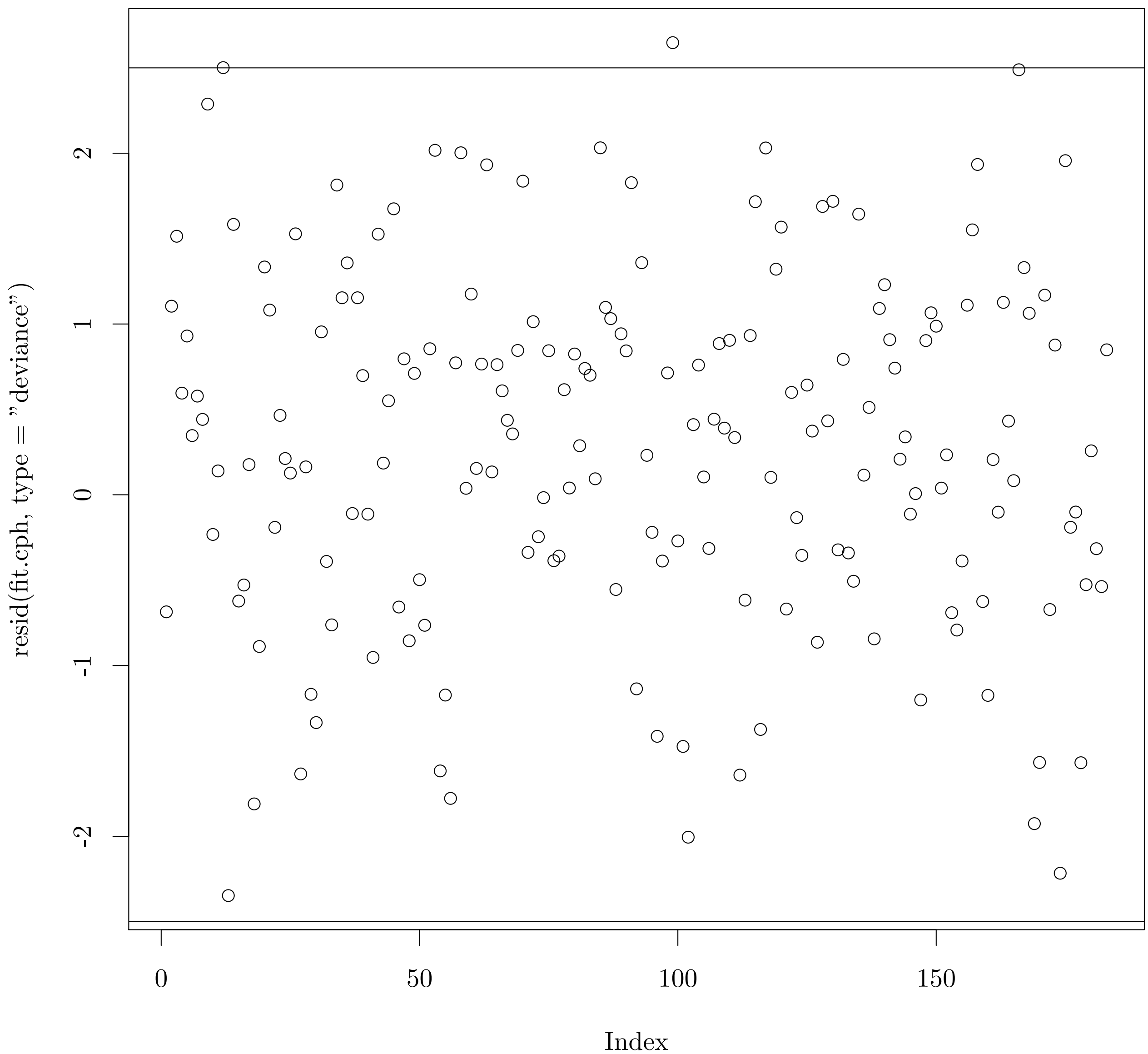



Not perfect but it'll do.

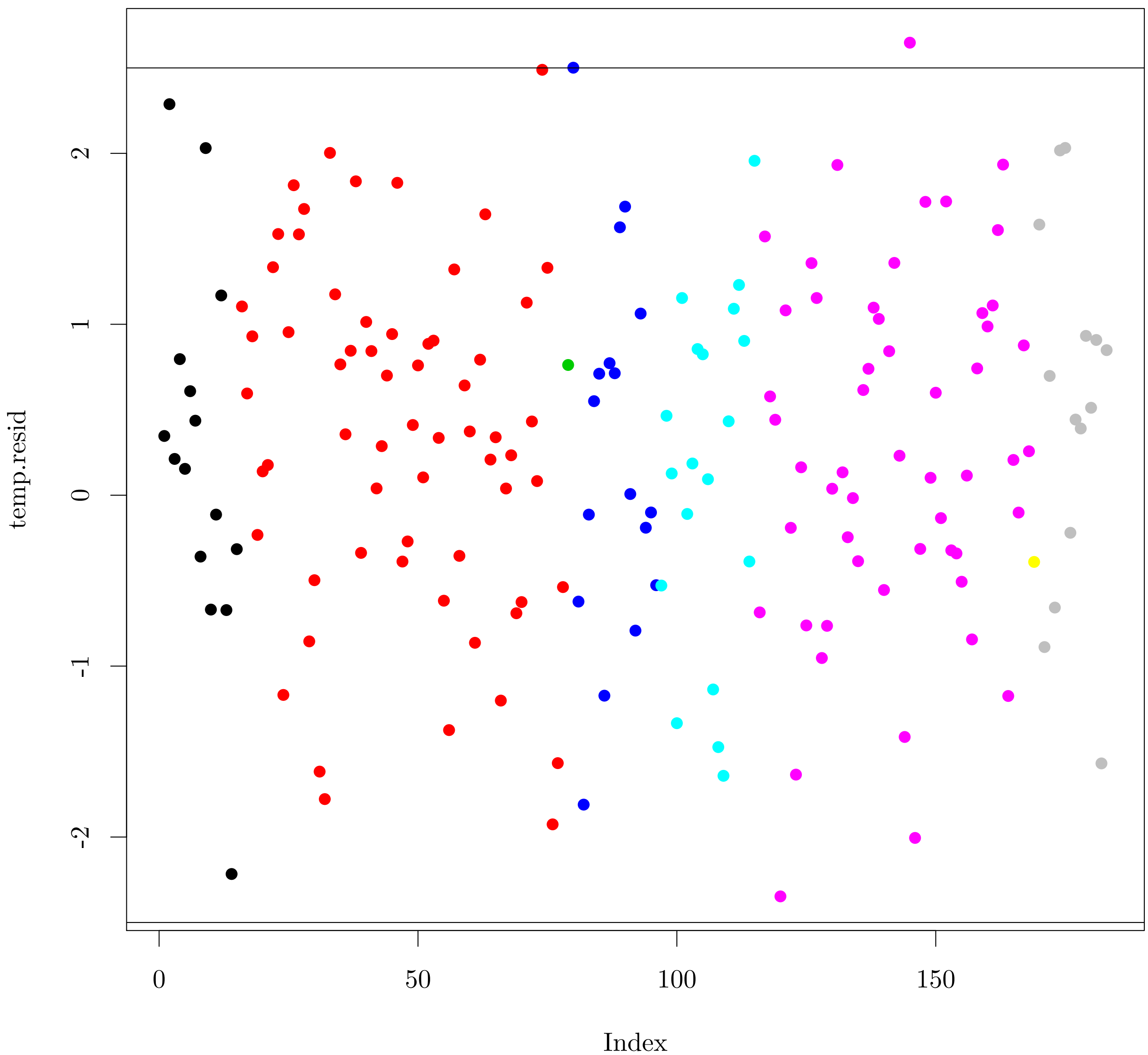
### 4.3 Outliers: full model

Look at deviance residuals, both marginally and stratified by major subgroups.

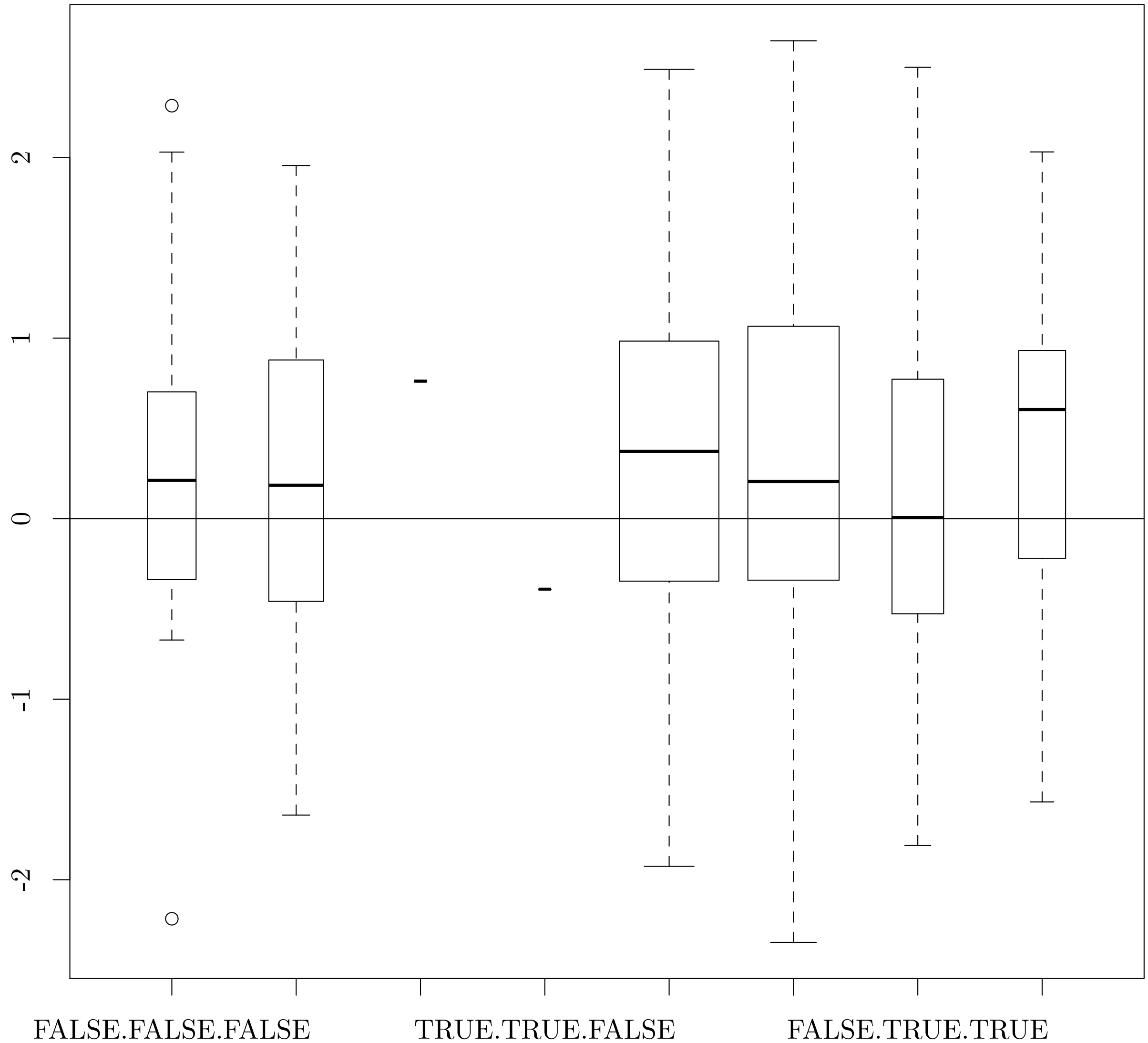
```
plot(resid(fit.cph, type = "deviance"))
abline(h = c(-2.5, 2.5))
```



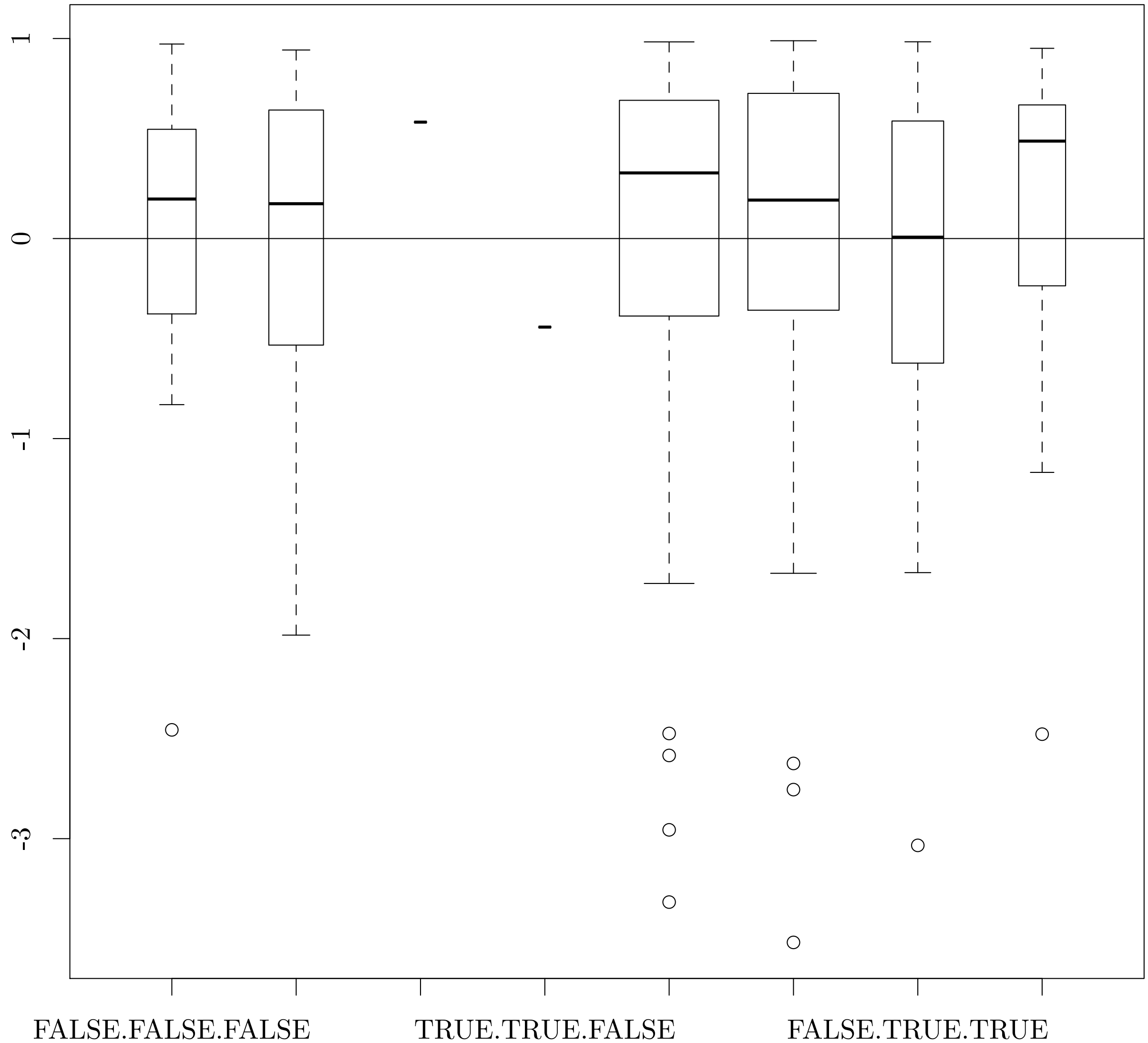

```r
temp.ord = order(data$SexM, data$A2, data$A4)
temp.resid = resid(fit.cph, type = "deviance")[temp.ord]
temp.col = (4*data$SexM + 2*data$A2 + data$A4 + 1)[temp.ord]
plot(temp.resid, col = temp.col, pch = 16)
abline(h = c(-2.5, 2.5))
```




```r
boxplot(resid(fit.cph, type = "deviance") ~ data$SexM + data$A2 + data$A4, varwidth = TRUE)
abline(h = 0)
```

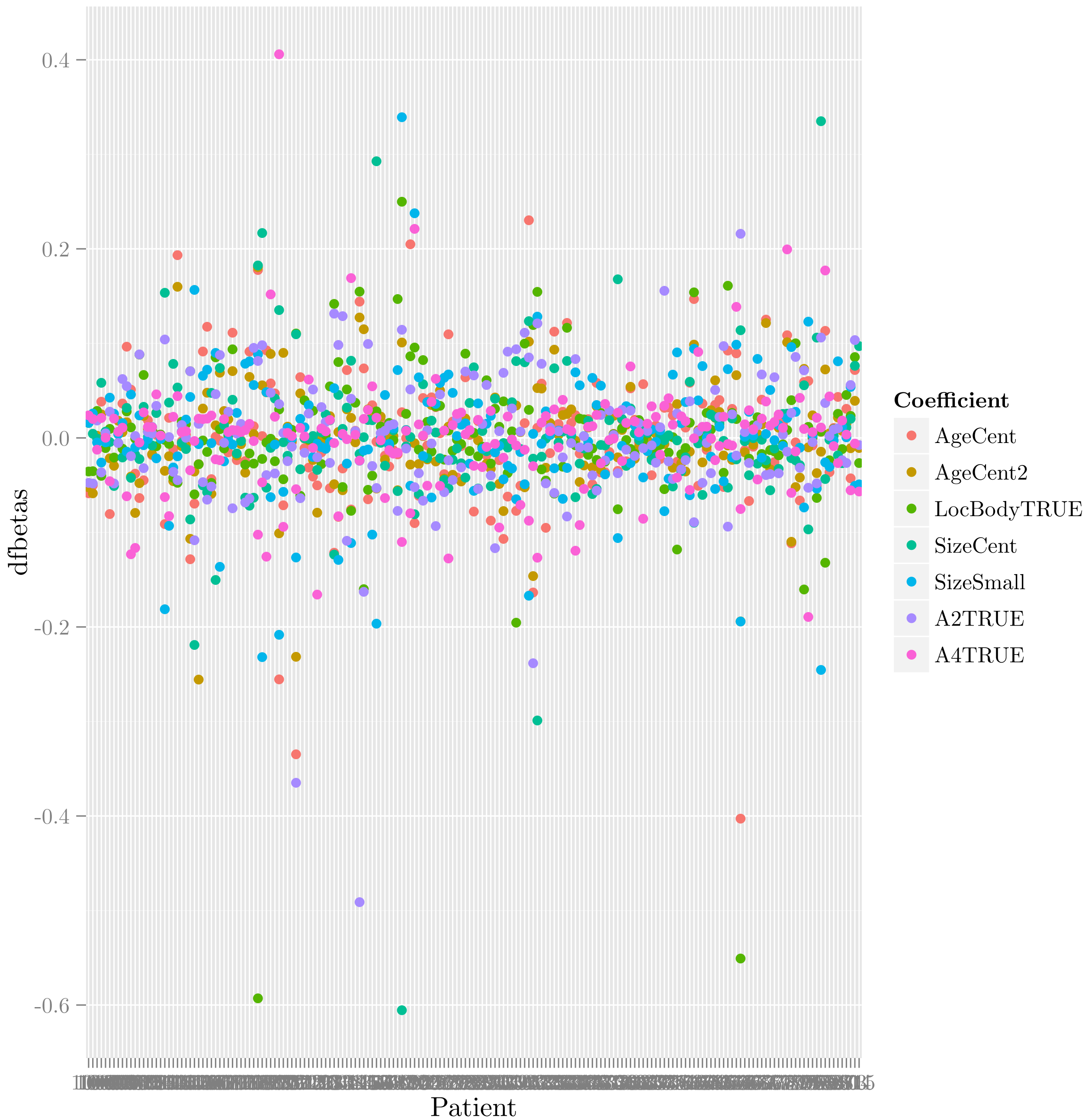



```r
boxplot(resid(fit.cph, type = "martingale") ~ data$SexM + data$A2 + data$A4, varwidth = TRUE)
abline(h = 0)
```




Use DFBETAS to examine influence.

```
temp = resid(fit.cph, type = "dfbetas")
colnames(temp) = names(fit.cph$coefficients)
temp = melt(temp)
colnames(temp) = c("Patient", "Coefficient", "dfbetas")
temp$Patient = gsub("NSWPCN_", "", temp$Patient)
ggplot(temp, aes(y = dfbetas, x = Patient, col = Coefficient)) + geom_point()
```




There is quite a number of rather influential observations. These could do with some checking, but first collapse down the model – there's little point doing dfbeta fucking about based on coefficients that will never get fit in the end anyway.

### 4.4 EDA: Variable selection

```
nobs.coxph <<- function(obj, ...) sum(obj$y[,2])
# Note: Exhaustive search at level 2 is only feasible for at most 5 variables
#fit.cph.as = glmulti(Surv(Time, DSD) ~ strata(SexM) + AgeCent + AgeCent2 + LocBody + SizeCent + SizeSma
set.seed(20150110)
fit.cph.as = glmulti(Surv(Time, DSD) ~ strata(SexM) + AgeCent + AgeCent2 + LocBody + SizeCent + SizeSmal

## TASK: Genetic algorithm in the candidate set.
```

```
## Initialization...
## Algorithm started...

## Warning in fitter(X, Y, strats, offset, init, control, weights = weights, :  Loglik converged
before variable  9 ; beta may be infinite.
## Warning in fitter(X, Y, strats, offset, init, control, weights = weights, :  Loglik converged
before variable  12 ; beta may be infinite.
## Warning in fitter(X, Y, strats, offset, init, control, weights = weights, :  Loglik converged
before variable  12 ; beta may be infinite.
## Warning in fitter(X, Y, strats, offset, init, control, weights = weights, :  Loglik converged
before variable  21 ; beta may be infinite.

## Improvements in best and average IC have bebingo en below the specified goals.
## Algorithm is declared to have converged.
## Completed.

# fit.cph.as
# After 830 generations:
# Best model: Surv(Time,DSD)~1+strata(SexM)+SizeCent+A2+A4
# Crit= 1367.16344569113
# Mean crit= 1401.37248769175
# Improvements in best and average IC have bebingo en below the specified goals.
# Algorithm is declared to have converged.
# Completed.
rm(nobs.coxph)
```

Also run BIC stepwise, because we can.

```
stepAIC(fit.cph, k = log(nrow(data)))

## Start:  AIC=1269
## Surv(Time, DSD) ~ strata(SexM) + AgeCent + AgeCent2 + LocBody +
##     SizeCent + SizeSmall + A2 + A4
##
##             Df  AIC
## - AgeCent    1 1264
## - LocBody    1 1264
## - SizeSmall  1 1264
## - AgeCent2   1 1266
## - SizeCent   1 1267
## <none>         1269
## - A2         1 1272
## - A4         1 1272
##
## Step:  AIC=1264
## Surv(Time, DSD) ~ strata(SexM) + AgeCent2 + LocBody + SizeCent +
##     SizeSmall + A2 + A4
##
##             Df  AIC
## - LocBody    1 1259
## - SizeSmall  1 1259
## - AgeCent2   1 1261
## - SizeCent   1 1262
## <none>         1264
## - A2         1 1266
```

```
## - A4         1 1267
##
## Step:  AIC=1259
## Surv(Time, DSD) ~ strata(SexM) + AgeCent2 + SizeCent + SizeSmall +
##     A2 + A4
##
##             Df  AIC
## - SizeSmall  1 1254
## - AgeCent2   1 1256
## - SizeCent   1 1257
## <none>         1259
## - A2         1 1261
## - A4         1 1262
##
## Step:  AIC=1254
## Surv(Time, DSD) ~ strata(SexM) + AgeCent2 + SizeCent + A2 + A4
##
##            Df  AIC
## - AgeCent2  1 1252
## - SizeCent  1 1253
## <none>        1254
## - A2        1 1257
## - A4        1 1257
##
## Step:  AIC=1252
## Surv(Time, DSD) ~ strata(SexM) + SizeCent + A2 + A4
##
##            Df  AIC
## - SizeCent  1 1250
## <none>        1252
## - A4        1 1253
## - A2        1 1254
##
## Step:  AIC=1250
## Surv(Time, DSD) ~ strata(SexM) + A2 + A4
##
##        Df  AIC
## <none>    1250
## - A4    1 1254
## - A2    1 1254
## Call:
## coxph(formula = Surv(Time, DSD) ~ strata(SexM) + A2 + A4, data = data)
##
##
##         coef exp(coef) se(coef)    z      p
## A2TRUE 0.630      1.88    0.201 3.14 0.0017
## A4TRUE 0.556      1.74    0.203 2.74 0.0061
##
## Likelihood ratio test=19.8  on 2 df, p=4.97e-05  n= 183, number of events= 175
```

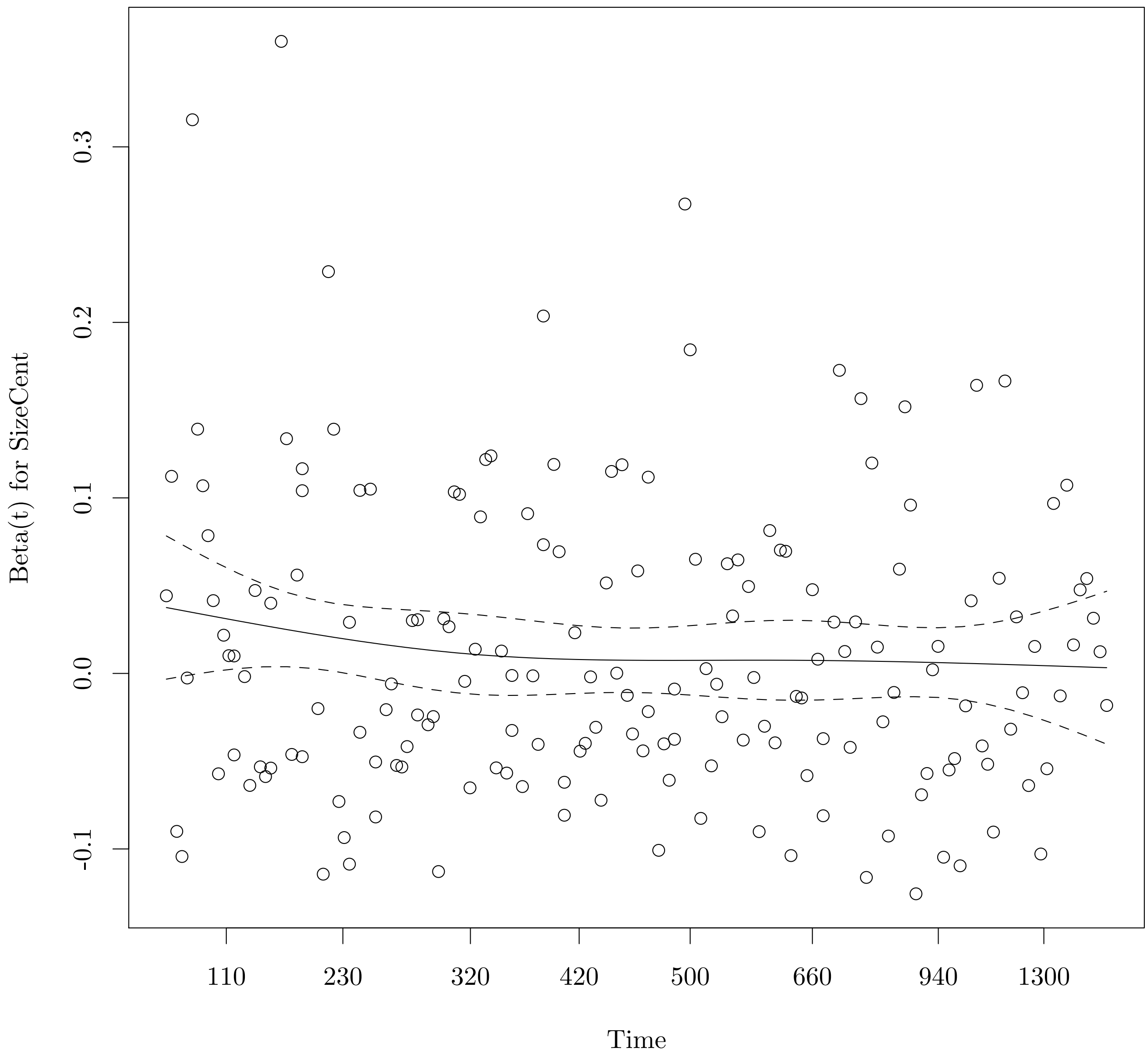
Consensus, excellent.

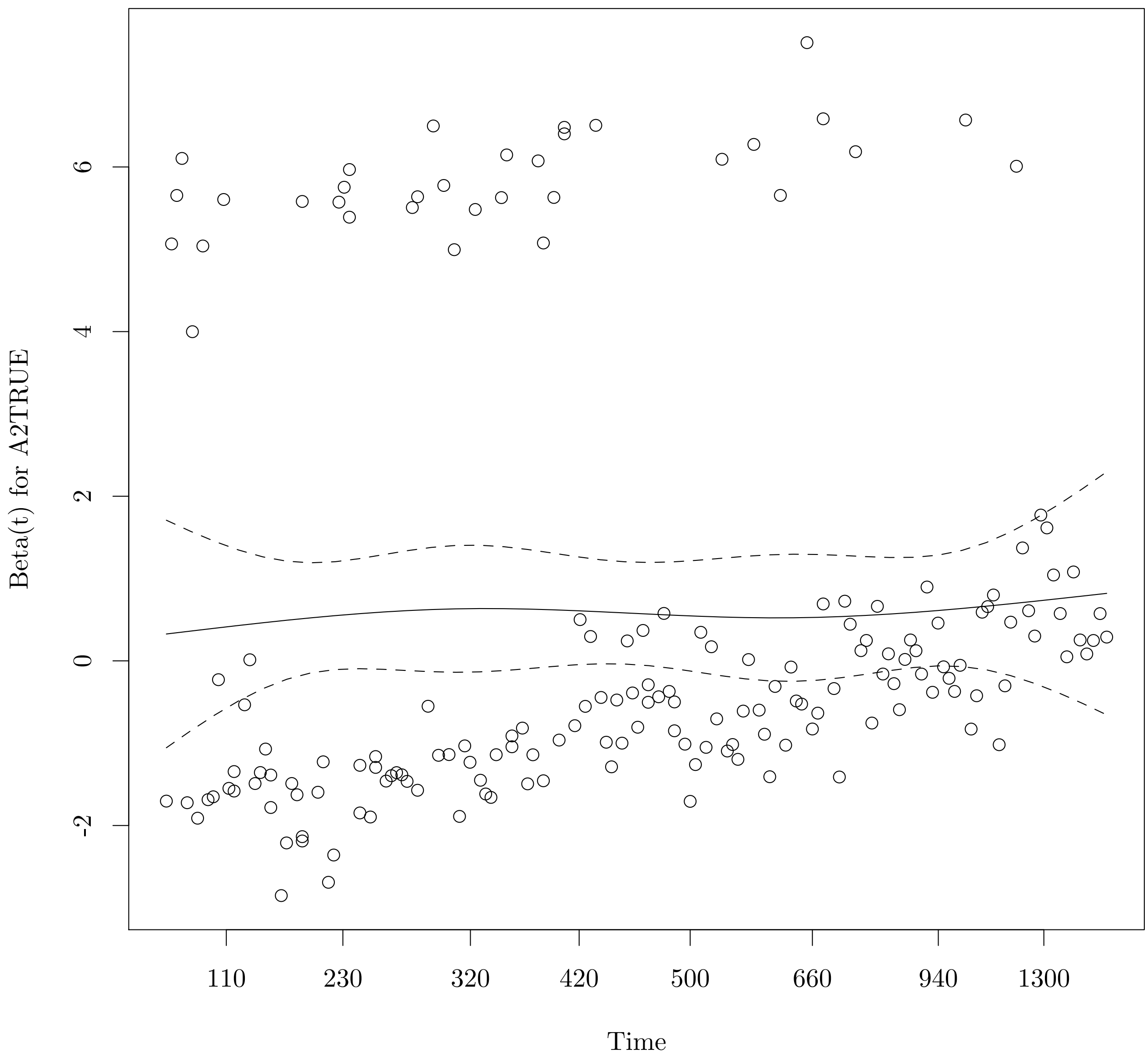
### 4.5 PH assumption: reduced model

```
fit.cph = coxph(Surv(Time, DSD) ~ strata(SexM) + SizeCent + A2 + A4, data = data)
cox.zph(fit.cph)
```

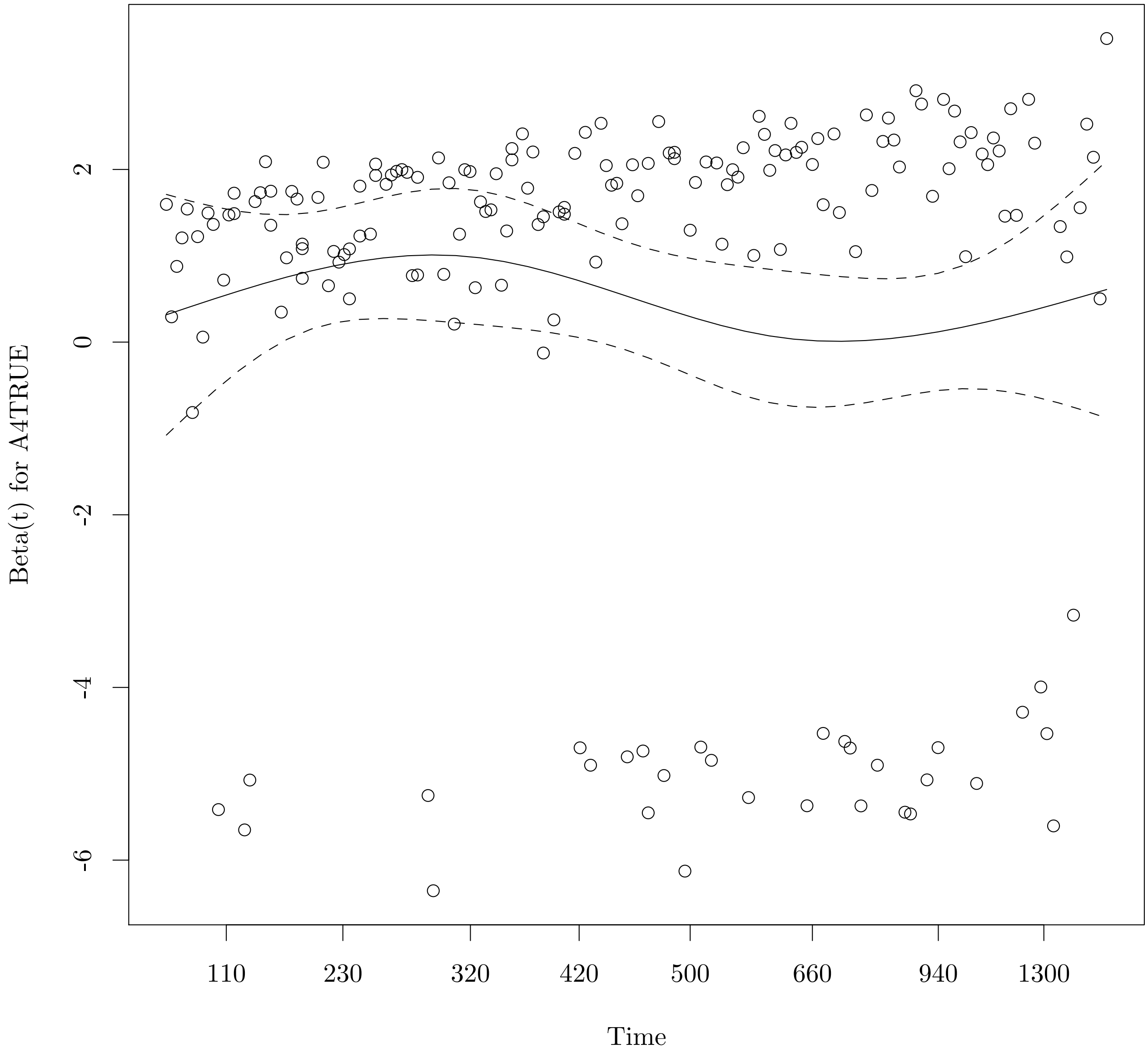
```
##              rho  chisq     p
## SizeCent -0.0905 1.6290 0.202
## A2TRUE    0.0230 0.0922 0.761
## A4TRUE   -0.0815 1.1103 0.292
## GLOBAL        NA 3.1175 0.374
```

```
plot(cox.zph(fit.cph))
```



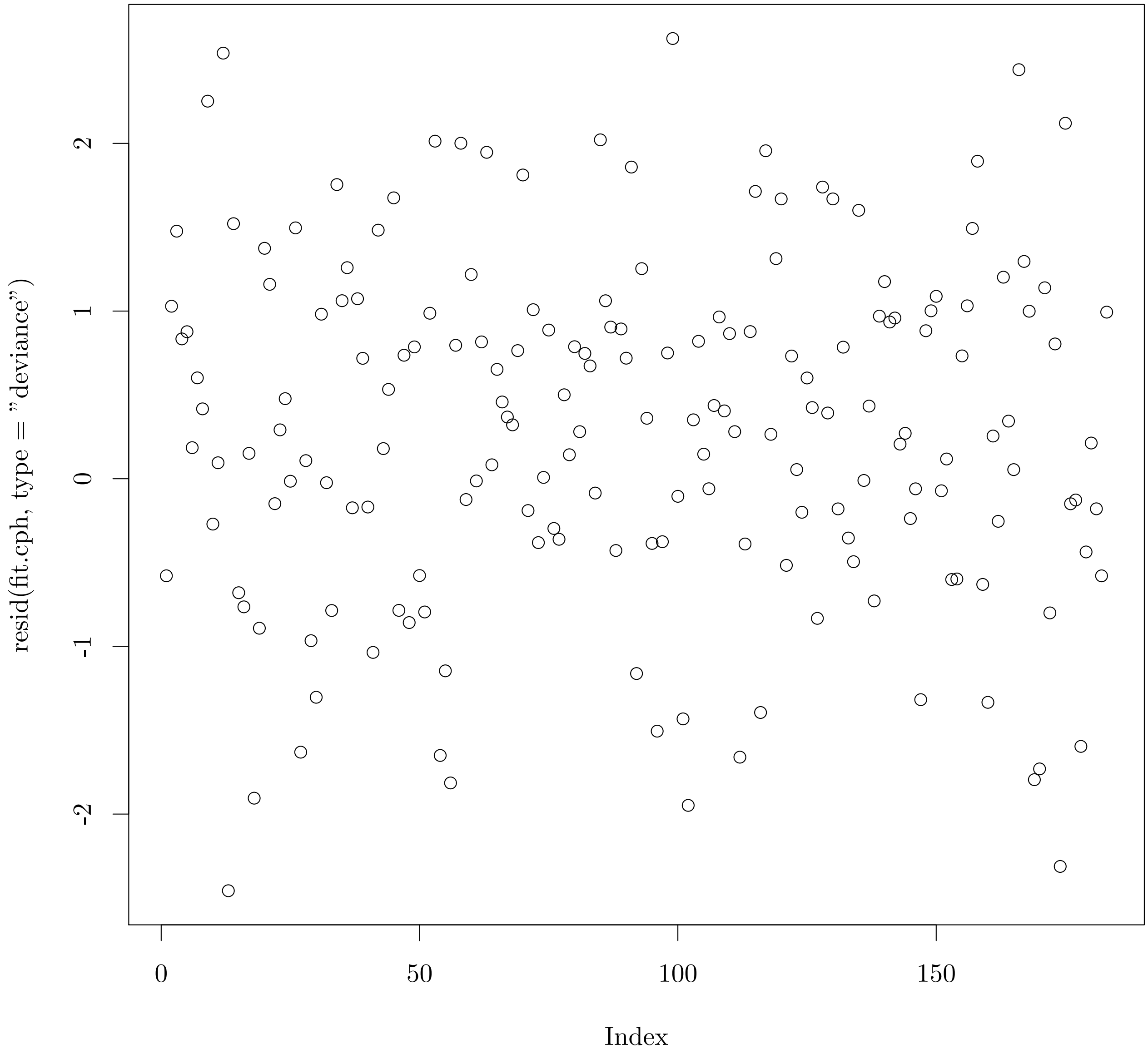


Beta(t) for A2TRUE
-2
0
2
4
6
110
230
320
420
500
660
940
1300
Time



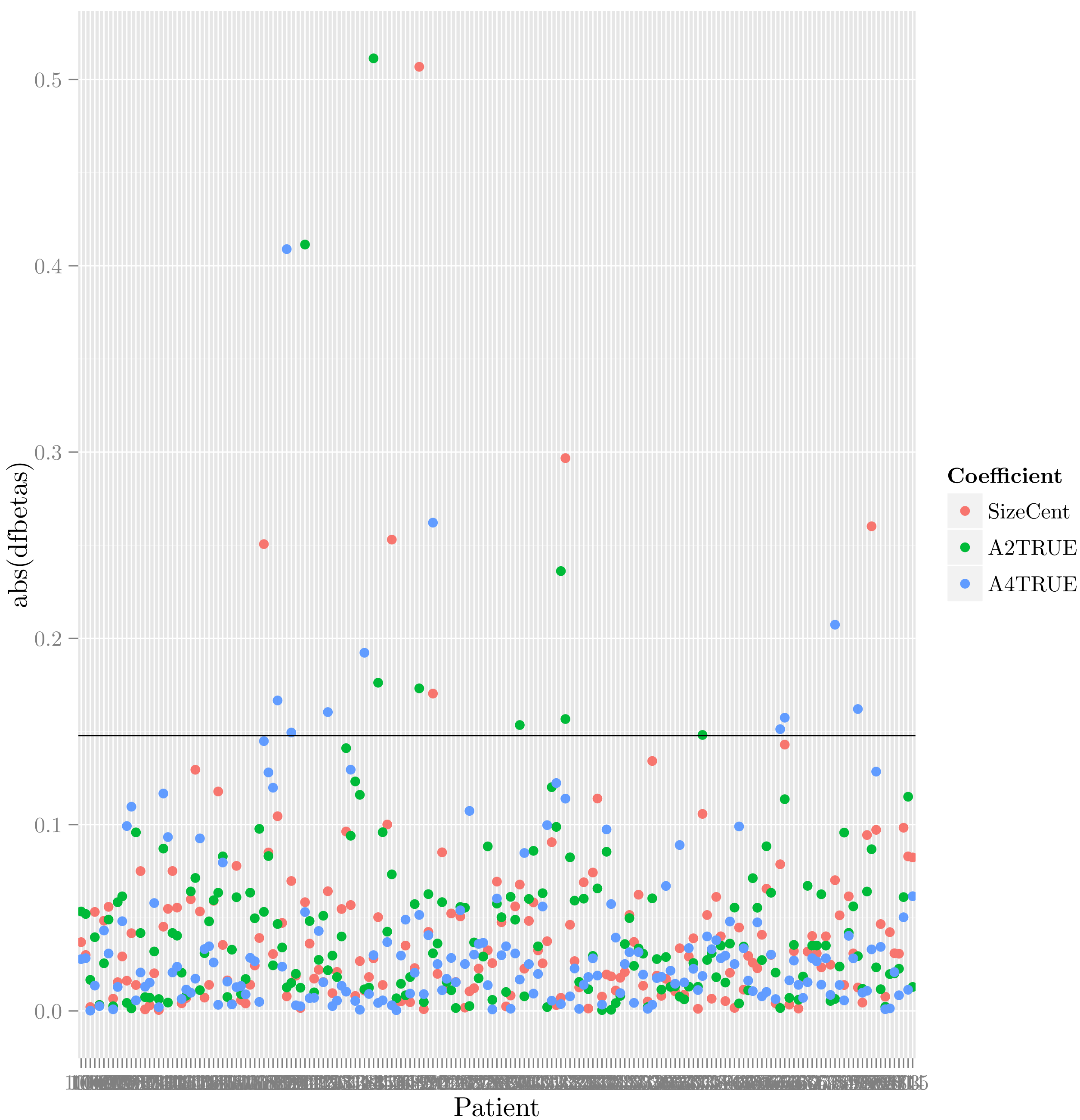

### 4.6 Outliers: reduced model

```
plot(resid(fit.cph, type = "deviance"))
```




Now generate the restricted fit and examine the DFBETAS on the reduced model.

```
temp = resid(fit.cph, type = "dfbetas")
colnames(temp) = names(fit.cph$coefficients)
temp = melt(temp)
colnames(temp) = c("Patient", "Coefficient", "dfbetas")
temp$Patient = gsub("NSWPCN_", "", temp$Patient)
2/sqrt(nrow(data))              # The classic threshold for concern is 2/sqrt(n).

## [1] 0.1478

ggplot(temp, aes(y = abs(dfbetas), x = Patient, col = Coefficient)) + geom_point() + geom_hline(yinterce
```




```
sort(apply(abs(resid(fit.cph, type = "dfbetas")), 1, max), decreasing = TRUE)

##  NSWPCN_144  NSWPCN_183 NSWPCN_1212 NSWPCN_1195  NSWPCN_318  NSWPCN_195
##    0.511340    0.506815    0.411412    0.408937    0.296728    0.262110
##  NSWPCN_799  NSWPCN_154 NSWPCN_1182  NSWPCN_317  NSWPCN_777  NSWPCN_142
##    0.260153    0.253014    0.250597    0.236114    0.207351    0.192280
##  NSWPCN_145 NSWPCN_1188  NSWPCN_795  NSWPCN_125  NSWPCN_655  NSWPCN_296
##    0.176188    0.166665    0.162051    0.160416    0.157420    0.153478
##  NSWPCN_654 NSWPCN_1196  NSWPCN_374  NSWPCN_131  NSWPCN_354  NSWPCN_133
##    0.151246    0.149449    0.148152    0.141058    0.134170    0.129507
## NSWPCN_1155  NSWPCN_802 NSWPCN_1186  NSWPCN_135  NSWPCN_316  NSWPCN_315
##    0.129471    0.128471    0.128036    0.123203    0.122328    0.120121
## NSWPCN_1187 NSWPCN_1167 NSWPCN_1143  NSWPCN_138  NSWPCN_814  NSWPCN_333
```

```
##    0.119838    0.117793    0.116717    0.116009    0.114995    0.114018
## NSWPCN_1072  NSWPCN_269  NSWPCN_152  NSWPCN_312 NSWPCN_1071  NSWPCN_636
##    0.109648    0.107374    0.100089    0.099732    0.099250    0.099010
##  NSWPCN_813 NSWPCN_1179  NSWPCN_335 NSWPCN_1453 NSWPCN_1082  NSWPCN_789
##    0.098347    0.097750    0.097415    0.095938    0.095841    0.095735
##  NSWPCN_798 NSWPCN_1145 NSWPCN_1157  NSWPCN_364  NSWPCN_647  NSWPCN_276
##    0.094433    0.093340    0.092592    0.089047    0.088390    0.088343
##  NSWPCN_305  NSWPCN_200  NSWPCN_303 NSWPCN_1168  NSWPCN_322  NSWPCN_815
##    0.085945    0.085147    0.084780    0.082972    0.082445    0.082391
## NSWPCN_1172 NSWPCN_1146 NSWPCN_1088  NSWPCN_331  NSWPCN_640  NSWPCN_281
##    0.077888    0.075122    0.075085    0.074289    0.071269    0.069418
##  NSWPCN_326  NSWPCN_664  NSWPCN_360 NSWPCN_1153 NSWPCN_1177  NSWPCN_651
##    0.069060    0.067170    0.067089    0.064155    0.063501    0.063455
##  NSWPCN_310  NSWPCN_194  NSWPCN_769  NSWPCN_351 NSWPCN_1029  NSWPCN_790
##    0.063192    0.062687    0.062634    0.062402    0.061529    0.061523
##  NSWPCN_284  NSWPCN_377  NSWPCN_304 NSWPCN_1165  NSWPCN_324 NSWPCN_1028
##    0.061268    0.061213    0.060116    0.059436    0.059216    0.058363
## NSWPCN_1139  NSWPCN_336  NSWPCN_182  NSWPCN_794  NSWPCN_294 NSWPCN_1023
##    0.057912    0.057377    0.057341    0.056127    0.056116    0.055877
##  NSWPCN_257  NSWPCN_445 NSWPCN_1147  NSWPCN_268  NSWPCN_643   NSWPCN_13
##    0.055813    0.055469    0.055464    0.055426    0.055403    0.054730
##   NSWPCN_10 NSWPCN_1019   NSWPCN_24 NSWPCN_1016  NSWPCN_347  NSWPCN_375
##    0.053392    0.053156    0.052313    0.052050    0.051517    0.051489
##  NSWPCN_781 NSWPCN_1227  NSWPCN_282 NSWPCN_1178  NSWPCN_164 NSWPCN_1022
##    0.051333    0.051115    0.050311    0.049749    0.048995    0.048464
## NSWPCN_1213 NSWPCN_1160    NSWPCN_4 NSWPCN_1190  NSWPCN_804 NSWPCN_1219
##    0.048263    0.048105    0.048021    0.047256    0.046638    0.042923
##  NSWPCN_807  NSWPCN_646  NSWPCN_666  NSWPCN_381  NSWPCN_770  NSWPCN_341
##    0.042281    0.040889    0.040225    0.040068    0.040056    0.039362
##  NSWPCN_370  NSWPCN_350  NSWPCN_270  NSWPCN_273   NSWPCN_20  NSWPCN_272
##    0.038992    0.037026    0.036778    0.036591    0.036201    0.035952
##  NSWPCN_346  NSWPCN_657    NSWPCN_7  NSWPCN_283  NSWPCN_309  NSWPCN_637
##    0.035888    0.035464    0.035127    0.034719    0.034677    0.034559
##  NSWPCN_369 NSWPCN_1158  NSWPCN_376 NSWPCN_1171  NSWPCN_810  NSWPCN_352
##    0.033701    0.033208    0.033132    0.032882    0.030954    0.030734
##  NSWPCN_811  NSWPCN_126  NSWPCN_161  NSWPCN_384  NSWPCN_638  NSWPCN_358
##    0.030726    0.029736    0.029730    0.029667    0.029548    0.027843
##  NSWPCN_280  NSWPCN_775  NSWPCN_362  NSWPCN_128  NSWPCN_653 NSWPCN_1150
##    0.025663    0.024846    0.021612    0.020969    0.020569    0.020533
## NSWPCN_1207  NSWPCN_662   NSWPCN_36  NSWPCN_330  NSWPCN_143  NSWPCN_166
##    0.019948    0.018477    0.018340    0.018191    0.018187    0.018185
##  NSWPCN_345 NSWPCN_1215   NSWPCN_21 NSWPCN_1176 NSWPCN_1018  NSWPCN_656
##    0.017812    0.017340    0.017265    0.017131    0.016678    0.016425
## NSWPCN_1170  NSWPCN_325  NSWPCN_256  NSWPCN_366 NSWPCN_1136  NSWPCN_363
##    0.016398    0.015592    0.015547    0.015304    0.015192    0.014593
##  NSWPCN_658 NSWPCN_1175 NSWPCN_1091  NSWPCN_373 NSWPCN_1211  NSWPCN_797
##    0.014004    0.013425    0.013036    0.012972    0.012450    0.011917
## NSWPCN_1152  NSWPCN_190  NSWPCN_334  NSWPCN_806  NSWPCN_157 NSWPCN_1027
##    0.011502    0.009024    0.007702    0.007660    0.006790    0.006582
## NSWPCN_1140  NSWPCN_353 NSWPCN_1020
##    0.006373    0.005084    0.003220
```

```r
sum(apply(abs(resid(fit.cph, type = "dfbetas")), 1, max) > 2/sqrt(nrow(data)))
```

```
## [1] 21
```

### 4.7 Summary of EDA

1. On the basis of pre-operative assessability and data availability, variables were filtered down to Sex, AgeCent, LocBody, SizeCent, A2, A4.

2. Functional forms for the continuous variates AgeCent and SizeCent indicated a possible slight quadratic effect on AgeCent, and a knee on SizeCent. These were modelled by incorporating additional terms.

3. Analysis of a full model fit (with additional nonlinear terms included) indicated violation of PH for gender. This was dealt with by stratification. A slight PH violation by age was deemed unimportant.

4. Variable selection by BIC (both stepwise and genetic all-subset) settled on a final model of Surv(Time,DSD) $\sim 1 +$ strata(SexM) + SizeCent + A2 + A4. This model was refit by coxph.

5. PH was verified on the final model. Deviance residuals showed no egregious outliers. dfBetaS indicated a number of influential observations, which require checking.

# 5 Final fits

```
fit.cph = coxph(Surv(Time, DSD) ~ strata(SexM) + SizeCent + A2 + A4, data = data)
```

```
set.seed(20150111)
fit.rsf = rfsrc(Surv(Time, DSD) ~ SexM + AgeCent + LocBody + SizeCent + A2 + A4, data = data, mtry = 1,
```

```
fit.gg = flexsurvreg(Surv(Time, DSD) ~ SexM + SizeCent + A2 + A4,
        anc = list(
                sigma = ~ SexM,
                Q = ~ SexM),
        data = data, dist = "gengamma")

fit.gf = flexsurvreg(Surv(Time, DSD) ~ SexM + SizeCent + A2 + A4,
        anc = list(
                sigma = ~ SexM,
                Q = ~ SexM,
                P = ~ SexM),
        data = data, dist = "genf")

fit.gg$loglik

## [1] -1263

fit.gf$loglik

## [1] -1262

pchisq(2*(fit.gf$loglik - fit.gg$loglik), 2, lower.tail = FALSE)

## [1] 0.3625

AIC(fit.gg)

## [1] 2545

AIC(fit.gf)
```

```
## [1] 2547
```

```
BIC(fit.gg)
```

```
## [1] 2574
```

```
BIC(fit.gf)
```

```
## [1] 2582
```

```
fit.gg
```

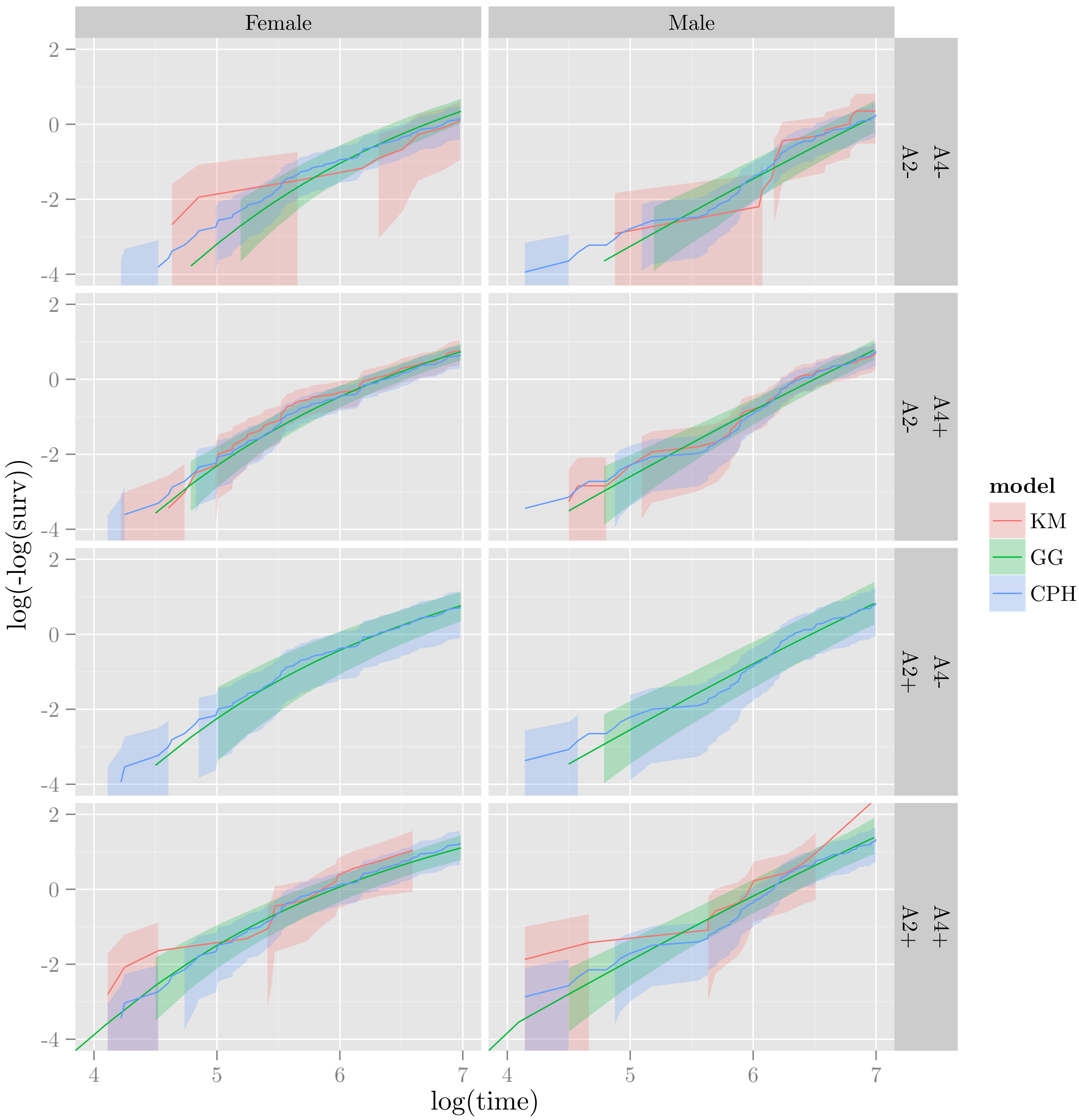
```
##
## Call:
## flexsurvreg(formula = Surv(Time, DSD) ~ SexM + SizeCent + A2 +     A4, anc = list(sigma = ~SexM, Q =
##
## Estimates:
##                   data mean  est       L95%      U95%      se
## mu                     NA     6.44681   6.07286   6.82076   0.19079
## sigma                  NA     0.80245   0.69416   0.92763   0.05935
## Q                      NA     0.06179  -0.51053   0.63411   0.29201
## SexMTRUE          0.47541     0.38255   0.03482   0.73028   0.17742
## SizeCent          3.18579    -0.00953  -0.01742  -0.00164   0.00403
## A2TRUE            0.18033    -0.38859  -0.66061  -0.11657   0.13879
## A4TRUE            0.80328    -0.36208  -0.63874  -0.08542   0.14116
## sigma(SexMTRUE)   0.47541    -0.25308  -0.49389  -0.01227   0.12287
## Q(SexMTRUE)       0.47541     0.78916   0.03792   1.54039   0.38329
##                   exp(est)  L95%      U95%
## mu                     NA        NA        NA
## sigma                  NA        NA        NA
## Q                      NA        NA        NA
## SexMTRUE          1.46602   1.03543   2.07567
## SizeCent          0.99052   0.98274   0.99837
## A2TRUE            0.67801   0.51653   0.88997
## A4TRUE            0.69623   0.52796   0.91812
## sigma(SexMTRUE)   0.77640   0.61025   0.98781
## Q(SexMTRUE)       2.20154   1.03865   4.66643
##
## N = 183,  Events: 175,  Censored: 8
## Total time at risk: 106023
## Log-likelihood = -1263, df = 9
## AIC = 2545
```

# 6 Fit assessment

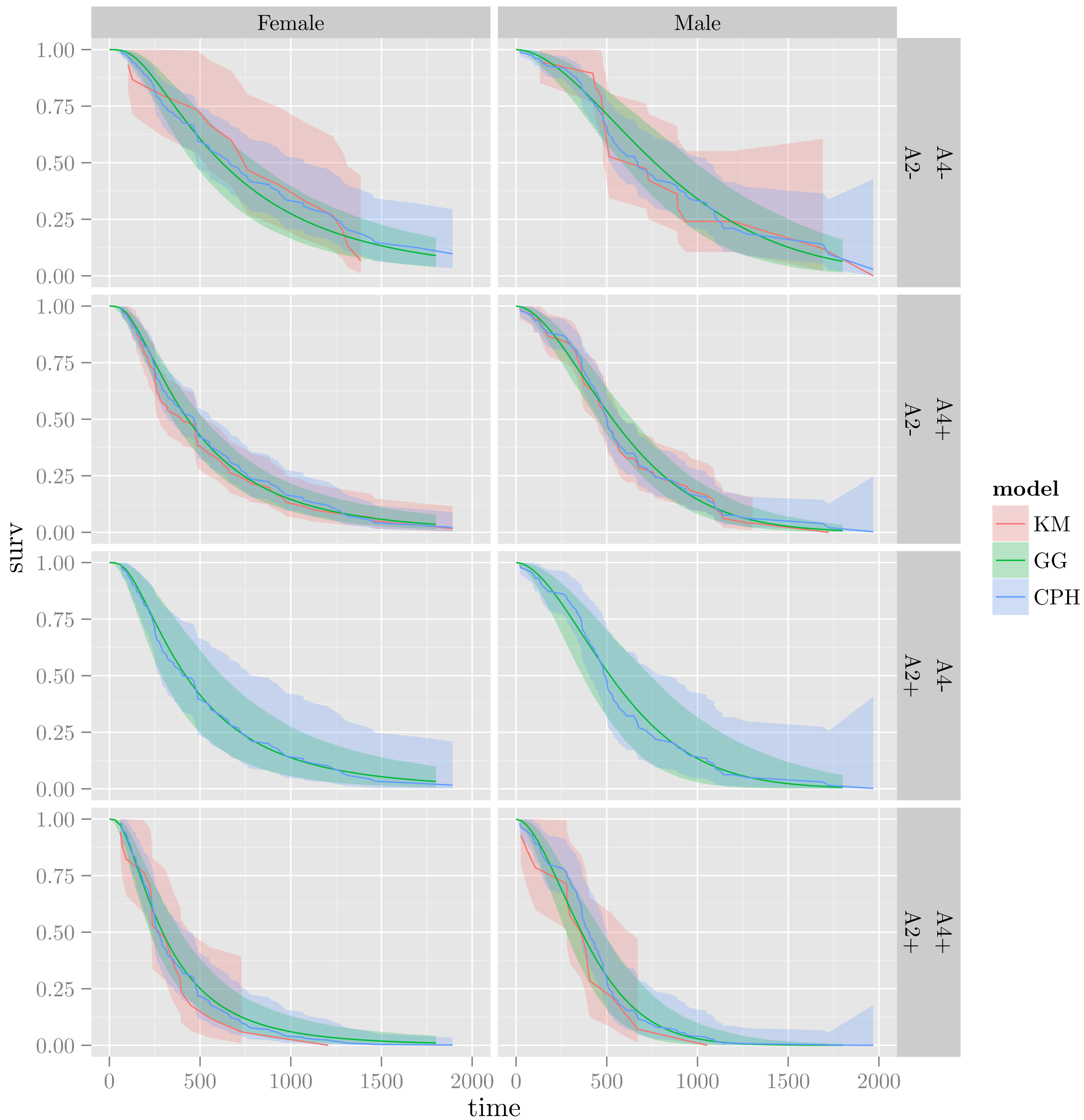
Plot fit stratified by sex, separate curves for A2, A4 status, at median (approx.) Size.

```
temp.grid = expand.grid(A4 = c(FALSE, TRUE), A2 = c(FALSE, TRUE), SexM = c(FALSE, TRUE), SizeCent = 0)
temp.grid$ID = sprintf("SexM=%s, A2=% -5s, A4=% -5s", temp.grid$SexM, temp.grid$A2, temp.grid$A4)
temp.preds = summary(fit.gg, newdata = temp.grid, type = "survival", t = seq(0, 365*5, 30))
temp.preds2 = do.call(rbind, temp.preds)
temp.preds2$group = rep(gsub(".*ID=", "", names(temp.preds)), each = nrow(temp.preds[[1]]))
temp.preds.cox = survfit(fit.cph, newdata = temp.grid)
```

```
temp.survfit = survfit(Surv(Time, DSD) ~ SexM + A2 + A4, data)
temp.data = data.frame(time = temp.survfit$time, surv = temp.survfit$surv, upper = temp.survfit$lower, l
temp.data = rbind(temp.data, data.frame(time = temp.preds2$time, surv = temp.preds2$est, upper = temp.pr
temp.data = rbind(temp.data, data.frame(time = temp.preds.cox$time, surv = temp.preds.cox$surv, upper =

temp.data$Sex = c("Male", "Female")[grepl("SexM=FALSE", temp.data$group)+1]
temp.data$A2 = c("A2-", "A2+")[grepl("A2=TRUE", temp.data$group)+1]
temp.data$A4 = c("A4-", "A4+")[grepl("A4=TRUE", temp.data$group)+1]

ggplot(temp.data, aes(x = log(time), y = log(-log(surv)), ymin = log(-log(lower)), ymax = log(-log(upper
        geom_ribbon(alpha = 0.25, colour = NA) +
        geom_line() +
        xlim(4, 7) + ylim(-4, 2) +
        facet_grid(A2 ~ A4 ~ Sex)

## Warning:  Removed 46 rows containing missing values (geom_path).
## Warning:  Removed 39 rows containing missing values (geom_path).
## Warning:  Removed 48 rows containing missing values (geom_path).
## Warning:  Removed 43 rows containing missing values (geom_path).
## Warning:  Removed 39 rows containing missing values (geom_path).
## Warning:  Removed 36 rows containing missing values (geom_path).
## Warning:  Removed 40 rows containing missing values (geom_path).
## Warning:  Removed 37 rows containing missing values (geom_path).
```




```
ggplot(temp.data, aes(x = time, y = surv, ymin = lower, ymax = upper, colour = model, fill = model)) +
        geom_ribbon(alpha = 0.25, colour = NA) +
        geom_line() + xlim(0, 2000) + ylim(0, 1) +
        facet_grid(A2 ~ A4 ~ Sex)

## Warning:  Removed 3 rows containing missing values (geom_path).
## Warning:  Removed 3 rows containing missing values (geom_path).
## Warning:  Removed 2 rows containing missing values (geom_path).
## Warning:  Removed 2 rows containing missing values (geom_path).
```




Some deviation though not significant. Most concerning is the A2- A4- female group, survival of which is underestimated by the flexsurv model. To approach this in a modelling sense would require interaction terms between Sex and A2, A4. Overfitting seems likely considering the very few data available for the A2+/A4- group. Perhaps just add a single "DoubleNegFemale" term.

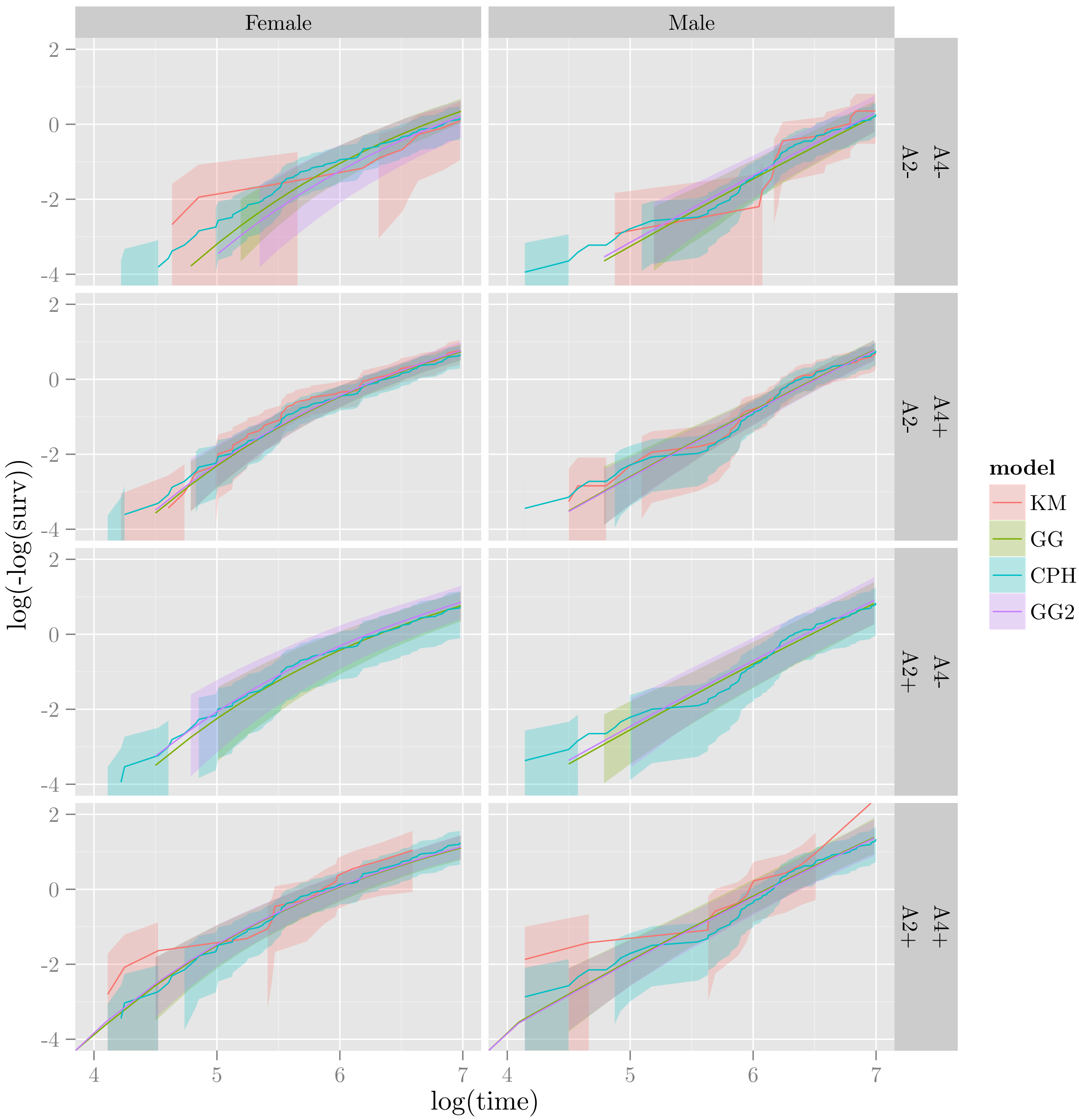
```
fit.gg2 = flexsurvreg(Surv(Time, DSD) ~ SexM + SizeCent + A2 + A4 + I(SexM == FALSE & A2 == FALSE & A4 =
        anc = list(
                sigma = ~ SexM,
                Q = ~ SexM),
        data = data, dist = "gengamma")

fit.gg2

##
```

```
## Call:
## flexsurvreg(formula = Surv(Time, DSD) ~ SexM + SizeCent + A2 +     A4 + I(SexM == FALSE & A2 == FALSE
##
## Estimates:
##                                                 data mean  est
## mu                                                    NA    6.37090
## sigma                                                 NA    0.79990
## Q                                                     NA    0.09541
## SexMTRUE                                         0.47541    0.40816
## SizeCent                                         3.18579   -0.00941
## A2TRUE                                           0.18033   -0.38417
## A4TRUE                                           0.80328   -0.29393
## I(SexM == FALSE & A2 == FALSE & A4 == FALSE)TRUE   0.08197    0.19993
## sigma(SexMTRUE)                                  0.47541   -0.25570
## Q(SexMTRUE)                                      0.47541    0.76926
##                                                 L95%      U95%
## mu                                               5.95233   6.78948
## sigma                                            0.69156   0.92521
## Q                                               -0.46926   0.66007
## SexMTRUE                                         0.05768   0.75863
## SizeCent                                        -0.01726  -0.00156
## A2TRUE                                          -0.65634  -0.11200
## A4TRUE                                          -0.62047   0.03260
## I(SexM == FALSE & A2 == FALSE & A4 == FALSE)TRUE  -0.35382   0.75369
## sigma(SexMTRUE)                                 -0.49656  -0.01485
## Q(SexMTRUE)                                      0.02665   1.51187
##                                                 se        exp(est)
## mu                                               0.21356        NA
## sigma                                            0.05939        NA
## Q                                                0.28810        NA
## SexMTRUE                                         0.17882   1.50404
## SizeCent                                         0.00401   0.99064
## A2TRUE                                           0.13886   0.68102
## A4TRUE                                           0.16660   0.74533
## I(SexM == FALSE & A2 == FALSE & A4 == FALSE)TRUE   0.28253   1.22132
## sigma(SexMTRUE)                                  0.12289   0.77437
## Q(SexMTRUE)                                      0.37889   2.15816
##                                                 L95%      U95%
## mu                                                    NA        NA
## sigma                                                 NA        NA
## Q                                                     NA        NA
## SexMTRUE                                         1.05938   2.13535
## SizeCent                                         0.98289   0.99844
## A2TRUE                                           0.51875   0.89404
## A4TRUE                                           0.53769   1.03313
## I(SexM == FALSE & A2 == FALSE & A4 == FALSE)TRUE   0.70200   2.12482
## sigma(SexMTRUE)                                  0.60862   0.98526
## Q(SexMTRUE)                                      1.02701   4.53518
##
## N = 183,  Events: 175,  Censored: 8
## Total time at risk: 106023
## Log-likelihood = -1263, df = 10
## AIC = 2546
```

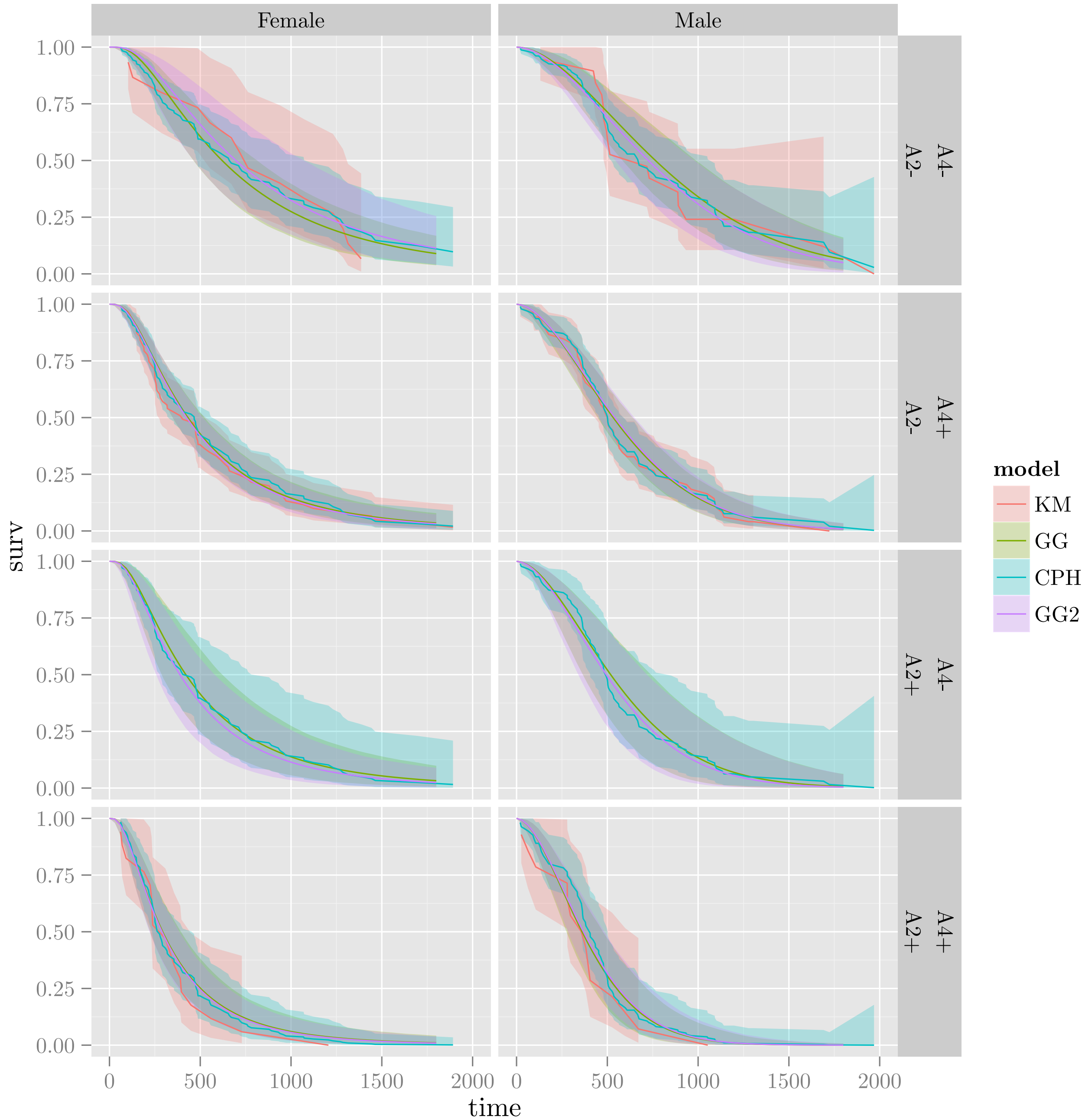
```r
AIC(fit.gg)

## [1] 2545

AIC(fit.gg2)

## [1] 2546

AIC(fit.gg) - AIC(fit.gg2)

## [1] -1.505

# Equivocal on AIC.  BIC would favour gg then.

pchisq(-2*(fit.gg$loglik - fit.gg2$loglik), 1, lower.tail = FALSE)

## [1] 0.4815

# Not good evidence on LRT
```

See how it plots relative to the others.

```r
temp.preds = summary(fit.gg2, newdata = temp.grid, type = "survival", t = seq(0, 365*5, 30))
temp.preds2 = do.call(rbind, temp.preds)
temp.preds2$group = rep(gsub(".*ID=", "", names(temp.preds)), each = nrow(temp.preds[[1]]))
temp.data = rbind(temp.data, data.frame(time = temp.preds2$time, surv = temp.preds2$est, upper = temp.pr
temp.data$Sex = c("Male", "Female")[grepl("SexM=FALSE", temp.data$group)+1]
temp.data$A2 = c("A2-", "A2+")[grepl("A2=TRUE", temp.data$group)+1]
temp.data$A4 = c("A4-", "A4+")[grepl("A4=TRUE", temp.data$group)+1]

ggplot(temp.data, aes(x = log(time), y = log(-log(surv)), ymin = log(-log(lower)), ymax = log(-log(upper
        geom_ribbon(alpha = 0.25, colour = NA) +
        geom_line() +
        xlim(4, 7) + ylim(-4, 2) +
        facet_grid(A2 ~ A4 ~ Sex)

## Warning:  Removed 71 rows containing missing values (geom_path).
## Warning:  Removed 64 rows containing missing values (geom_path).
## Warning:  Removed 73 rows containing missing values (geom_path).
## Warning:  Removed 68 rows containing missing values (geom_path).
## Warning:  Removed 64 rows containing missing values (geom_path).
## Warning:  Removed 61 rows containing missing values (geom_path).
## Warning:  Removed 65 rows containing missing values (geom_path).
## Warning:  Removed 62 rows containing missing values (geom_path).
```



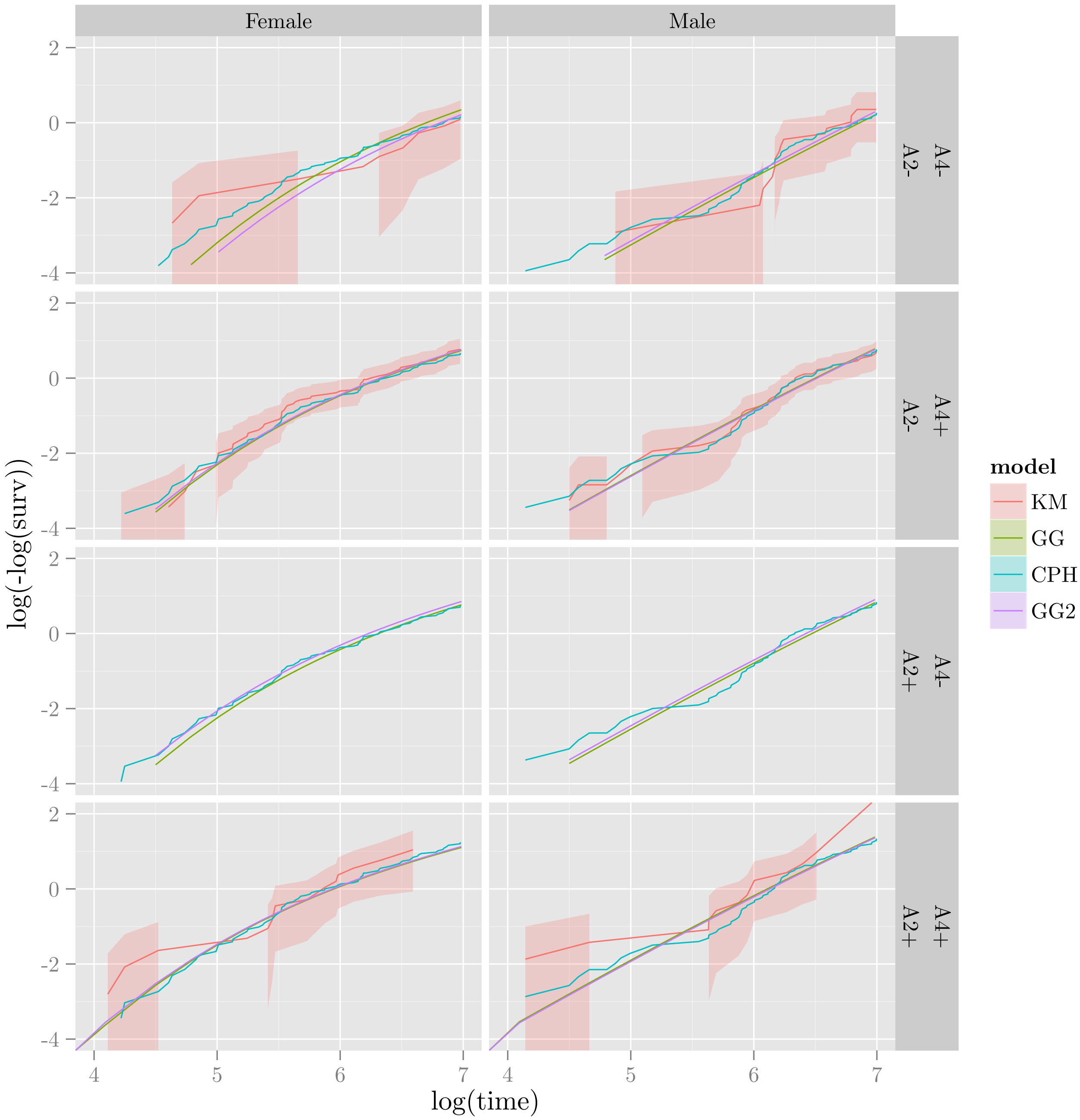

```
ggplot(temp.data, aes(x = time, y = surv, ymin = lower, ymax = upper, colour = model, fill = model)) +
        geom_ribbon(alpha = 0.25, colour = NA) +
        geom_line() + xlim(0, 2000) + ylim(0, 1) +
        facet_grid(A2 ~ A4 ~ Sex)

## Warning:  Removed 3 rows containing missing values (geom_path).
## Warning:  Removed 3 rows containing missing values (geom_path).
## Warning:  Removed 2 rows containing missing values (geom_path).
## Warning:  Removed 2 rows containing missing values (geom_path).
```




An alternative take, showing errors with the KMs only.

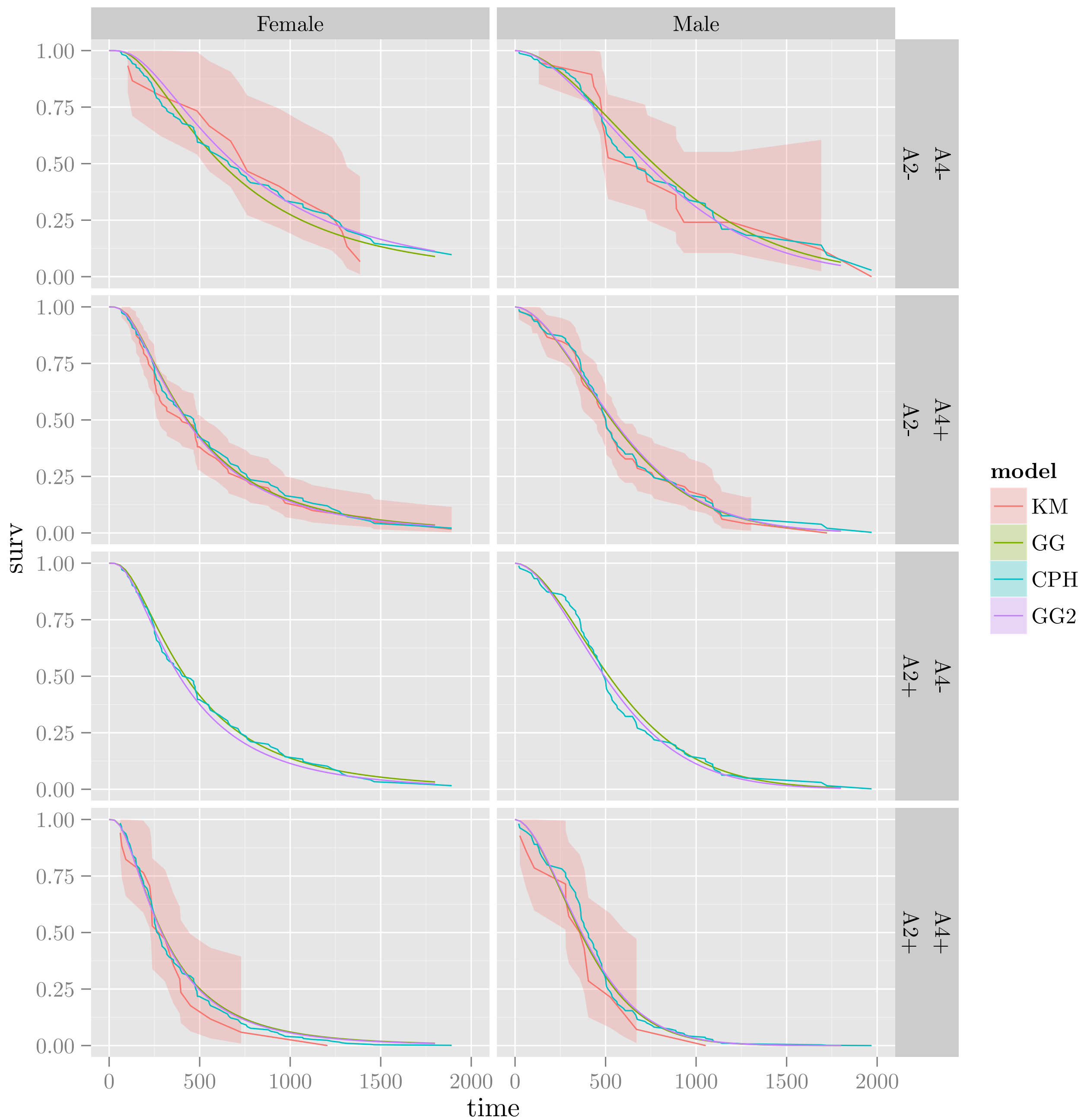
```
temp.data$lower[temp.data$model != "KM"] = NA
temp.data$upper[temp.data$model != "KM"] = NA
ggplot(temp.data, aes(x = log(time), y = log(-log(surv)), ymin = log(-log(lower)), ymax = log(-log(upper
        geom_ribbon(alpha = 0.25, colour = NA) +
        geom_line() +
        xlim(4, 7) + ylim(-4, 2) +
        facet_grid(A2 ~ A4 ~ Sex)

## Warning:  Removed 71 rows containing missing values (geom_path).
## Warning:  Removed 64 rows containing missing values (geom_path).
## Warning:  Removed 73 rows containing missing values (geom_path).
## Warning:  Removed 68 rows containing missing values (geom_path).
```

```
## Warning:  Removed 64 rows containing missing values (geom_path).
## Warning:  Removed 61 rows containing missing values (geom_path).
## Warning:  Removed 65 rows containing missing values (geom_path).
## Warning:  Removed 62 rows containing missing values (geom_path).
```



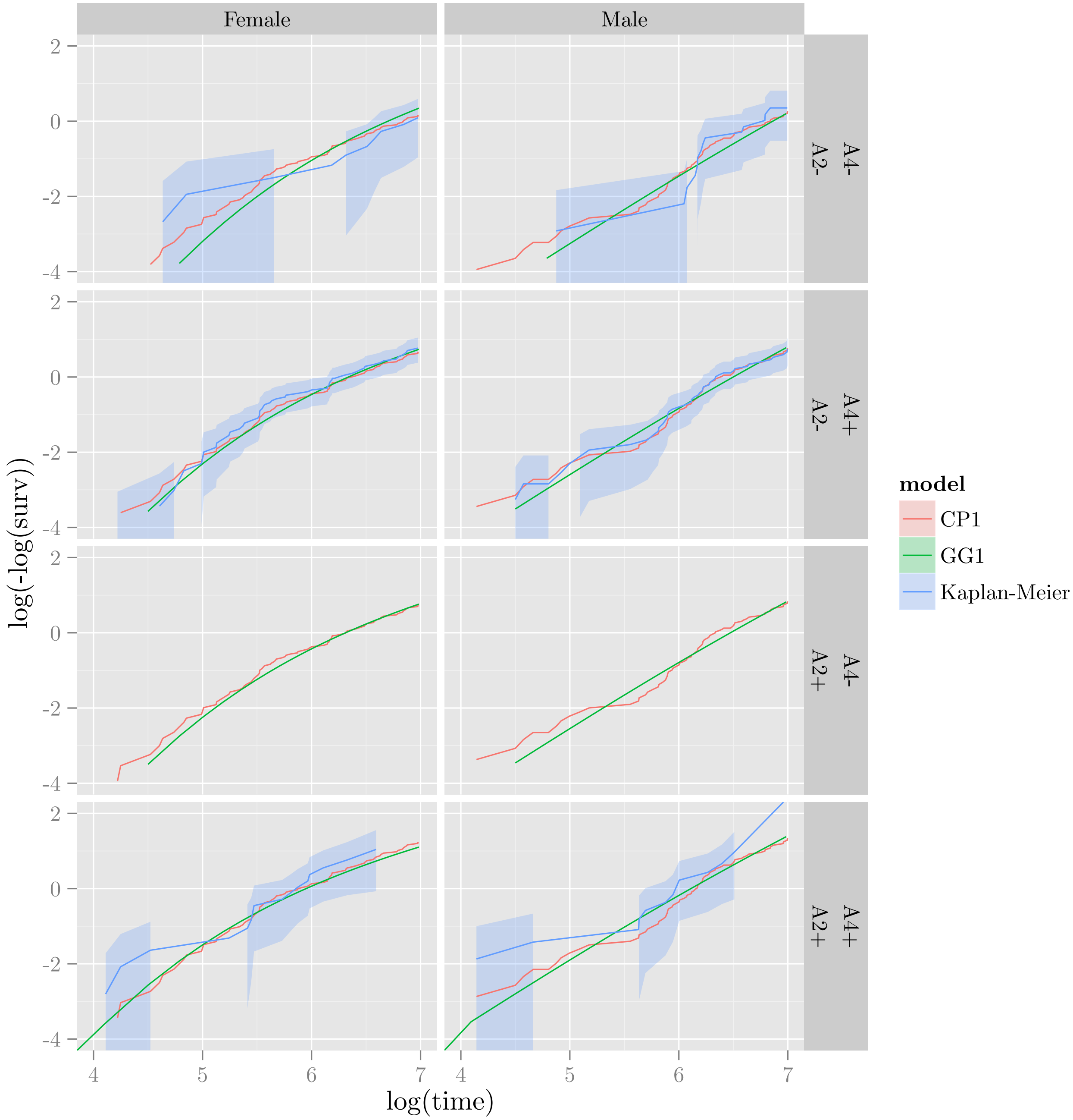

```
ggplot(temp.data, aes(x = time, y = surv, ymin = lower, ymax = upper, colour = model, fill = model)) +
        geom_ribbon(alpha = 0.25, colour = NA) +
        geom_line() + xlim(0, 2000) + ylim(0, 1) +
        facet_grid(A2 ~ A4 ~ Sex)

## Warning:  Removed 3 rows containing missing values (geom_path).
## Warning:  Removed 3 rows containing missing values (geom_path).
```

```
## Warning:  Removed 2 rows containing missing values (geom_path).
## Warning:  Removed 2 rows containing missing values (geom_path).
```



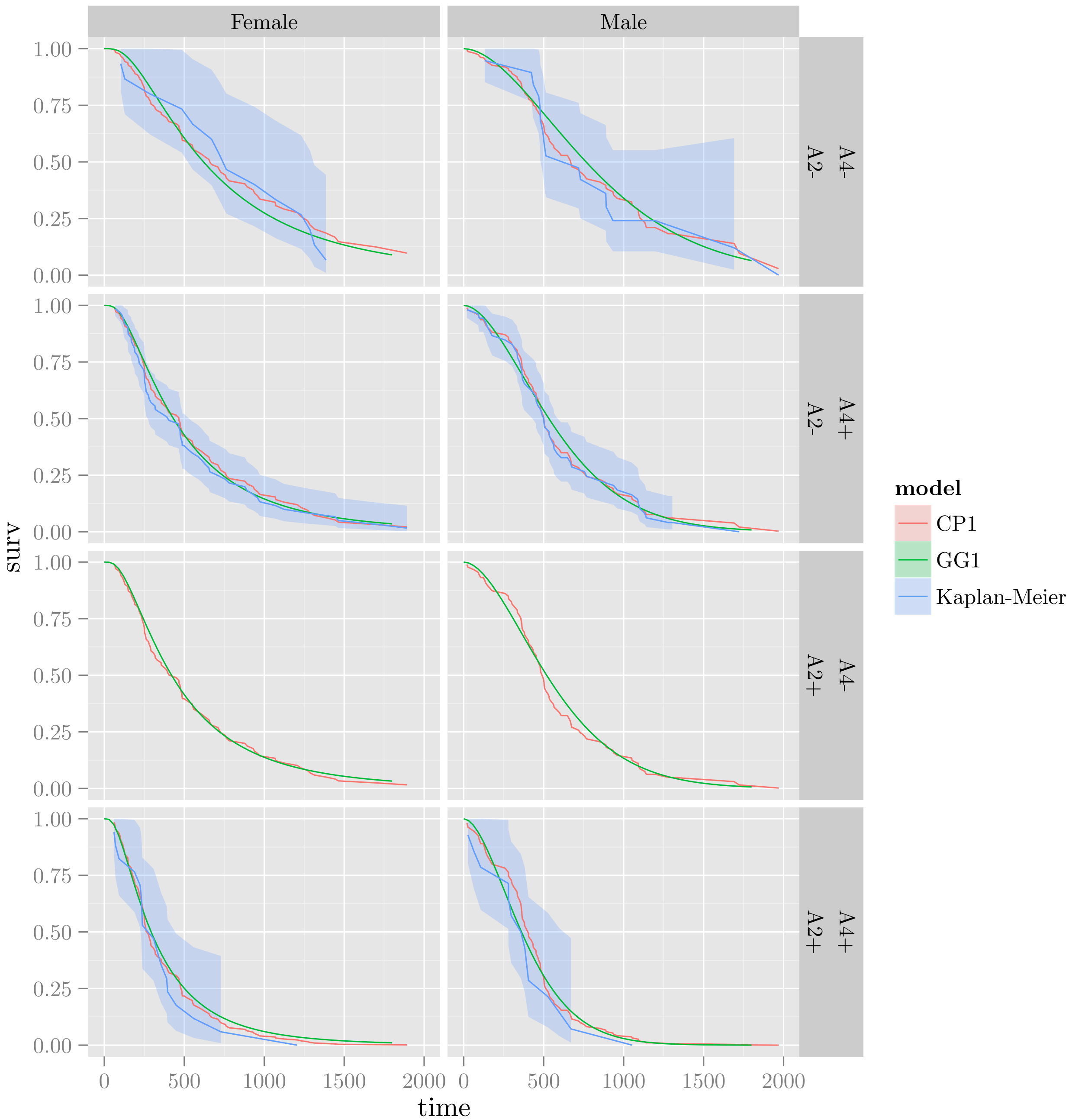

```
temp.data$lower[temp.data$model != "KM"] = NA
temp.data$upper[temp.data$model != "KM"] = NA
temp.data = temp.data[temp.data$model != "GG2",]
temp.data$model = c("KM" = "Kaplan-Meier", "GG" = "GG1", "CPH" = "CP1")[temp.data$model]
ggplot(temp.data, aes(x = log(time), y = log(-log(surv)), ymin = log(-log(lower)), ymax = log(-log(upper
        geom_ribbon(alpha = 0.25, colour = NA) +
        geom_line() +
        xlim(4, 7) + ylim(-4, 2) +
        facet_grid(A2 ~ A4 ~ Sex)
```

```
## Warning:  Removed 46 rows containing missing values (geom_path).
## Warning:  Removed 39 rows containing missing values (geom_path).
## Warning:  Removed 48 rows containing missing values (geom_path).
## Warning:  Removed 43 rows containing missing values (geom_path).
## Warning:  Removed 39 rows containing missing values (geom_path).
## Warning:  Removed 36 rows containing missing values (geom_path).
## Warning:  Removed 40 rows containing missing values (geom_path).
## Warning:  Removed 37 rows containing missing values (geom_path).
```




```
ggplot(temp.data, aes(x = time, y = surv, ymin = lower, ymax = upper, colour = model, fill = model)) +
        geom_ribbon(alpha = 0.25, colour = NA) +
        geom_line() + xlim(0, 2000) + ylim(0, 1) +
```

```
       facet_grid(A2 ~ A4 ~ Sex)

## Warning:  Removed 3 rows containing missing values (geom_path).
## Warning:  Removed 3 rows containing missing values (geom_path).
## Warning:  Removed 2 rows containing missing values (geom_path).
## Warning:  Removed 2 rows containing missing values (geom_path).
```




# 7 Model selection

It looks like that's as far as we can go with tweaking the fits. Time to put the different models against each other on the holdout data, and choose a winner.

DIY IBS, wooo.

```
calcIBS = function(surv, pred, pred_times, max_time)
{
	stopifnot(nrow(surv) == nrow(pred) && length(pred_times) == ncol(pred))

	n = nrow(surv)
	marg_survfit = survfit(surv ~ 1)
	marg_censfit = survfit(Surv(surv[,1], !surv[,2]) ~ 1)
	marg_surv_func = approxfun(marg_survfit$time, marg_survfit$surv, method = "constant", yleft = 1,
	marg_cens_func = approxfun(marg_censfit$time, marg_censfit$surv, method = "constant", yleft = 1,

	pred_funcs = apply(pred, 1, function(pat_preds) approxfun(pred_times, pat_preds, yleft = 1, yrig

	indiv_patient_bsc = function(pat_i, tstars)
	{
		observed_time = surv[pat_i, 1]
		observed_event = surv[pat_i, 2]
		pred_func = pred_funcs[[pat_i]]
		category = 1*(observed_time <= tstars & observed_event) + 2*(observed_time > tstars) + 3
		bsc = rep(NA, length(tstars))
		bsc[category == 1] = pred_func(tstars[category == 1])^2 / marg_cens_func(observed_time)
		bsc[category == 2] = (1 - pred_func(tstars[category == 2]))^2 / marg_cens_func(tstars[ca
		bsc[category == 3] = 0
		bsc
	}

	bsc_func = function(tstars) { rowMeans(sapply(1:n, function(pat_i) indiv_patient_bsc(pat_i, tsta

	weight_func = function(tstars) { (1 - marg_surv_func(tstars)) / (1 - marg_surv_func(max_time)) }

	# Be slack and do trapezoidal int. with a fine grid.  It should be possible
	# to calulate the int. exactly but I cbfed.
	int_grid = seq(0, max_time, length.out = 1e3)
	bsc_vals = bsc_func(int_grid)
	weight_vals = weight_func(int_grid)
	int_vals = bsc_vals * weight_vals
	ibsc = (2*sum(int_vals) - int_vals[1] - int_vals[length(int_vals)]) * (diff(range(int_grid))) /

	return(list(bsc = bsc_vals, weights = weight_vals, eval_times = int_grid, ibsc = ibsc))
}
```

Calculate survival probability predictions for each of the models, on the validation data.

```
ibs_times = sort(unique(data.val$Time))
ibs_preds_gg = as.matrix(t(sapply(summary(fit.gg, newdata = data.val, type = "survival", t = ibs_times),
ibs_preds_gg2 = as.matrix(t(sapply(summary(fit.gg2, newdata = data.val, type = "survival", t = ibs_times
temp_cox_preds = survfit(fit.cph, newdata = data.val)
ibs_preds_cph = simplify2array(tapply(1:length(temp_cox_preds$time), rep(names(temp_cox_preds$strata), t
	approx(x = temp_cox_preds$time[strat_i], y = temp_cox_preds$surv[strat_i], xout = ibs_times, met
ibs_preds_cph = t(ibs_preds_cph[,rownames(data.val)])
temp_rsf_preds = predict(fit.rsf, newdata = data.val)
ibs_preds_rsf = t(apply(temp_rsf_preds$survival, 1, function(survs) approx(temp_rsf_preds$time.interest,
# Patients (from data.val) are in rows, times (from ibs_times) in columns.

# Add a no-information KM predictor
```

```
temp_km0 = survfit(Surv(Time, DSD) ~ 1, data)
ibs_preds_km0 = t(matrix(rep(approx(temp_km0$time, temp_km0$surv, xout = ibs_times, method = "constant",
ibs_preds_all = list(gg = ibs_preds_gg, gg2 = ibs_preds_gg2, cph = ibs_preds_cph, rsf = ibs_preds_rsf, k
```

```
val.prob.times = seq(0, max(data.val$Time), 1)

temp.coefs = coef(fit.gg)
val.linpred.gg = sapply(1:length(temp.coefs), function(coef_i) {
    if (names(temp.coefs)[coef_i] %in% colnames(data.val)) {
        temp.coefs[coef_i] * data.val[,names(temp.coefs)[coef_i]]
    } else if (gsub("TRUE$", "", names(temp.coefs)[coef_i]) %in% colnames(data.val)) {
        temp.coefs[coef_i] * data.val[,gsub("TRUE$", "", names(temp.coefs)[coef_i])]
    } else {
        rep(0, nrow(data.val))
    } })
val.linpred.gg = -rowSums(val.linpred.gg)   # Negate to bring into concordance with the direction of Cox
temp = summary(fit.gg, newdata = data.val, ci = FALSE)
val.prob.gg = sapply(temp, function(x) approx(x[,1], x[,2], xout = val.prob.times, yleft = 1, yright = 0
colnames(val.prob.gg) = rownames(data.val)

temp.coefs = coef(fit.gg2)
val.linpred.gg2 = sapply(1:length(temp.coefs), function(coef_i) {
    if (names(temp.coefs)[coef_i] %in% colnames(data.val)) {
        temp.coefs[coef_i] * data.val[,names(temp.coefs)[coef_i]]
    } else if (gsub("TRUE$", "", names(temp.coefs)[coef_i]) %in% colnames(data.val)) {
        temp.coefs[coef_i] * data.val[,gsub("TRUE$", "", names(temp.coefs)[coef_i])]
    } else {
        rep(0, nrow(data.val))
    } })
val.linpred.gg2 = -rowSums(val.linpred.gg2)   # Negate to bring into concordance with the direction of C
temp = summary(fit.gg2, newdata = data.val, ci = FALSE)
val.prob.gg2 = sapply(temp, function(x) approx(x[,1], x[,2], xout = val.prob.times, yleft = 1, yright =
colnames(val.prob.gg2) = rownames(data.val)

val.linpred.cph = predict(fit.cph, newdata = data.val)
temp = survfit(fit.cph, newdata = data.val)
val.prob.cph = simplify2array(tapply(1:length(temp$surv), rep(names(temp$strata), temp$strata), function

temp = predict(fit.rsf, newdata = data.val)
# val.linpred.rsf = temp£predicted
# Median survival time:
val.linpred.rsf = apply(temp$survival, 1, function(s1) {
    sfunc = approxfun(temp$time.interest, s1, yleft = 1, yright = 0, rule = 2)
    med = uniroot(function(x) sfunc(x) - 0.5, lower = min(temp$time.interest), upper = max(temp$time.int
    med
})
val.linpred.rsf = -val.linpred.rsf
val.prob.rsf = apply(temp$survival, 1, function(s1) approx(temp$time.interest, s1, xout = val.prob.times
colnames(val.prob.rsf) = rownames(data.val)

summary(coxph(Surv(Time, DSD) ~ val.linpred.gg, data.val))

## Call:
```

```
## coxph(formula = Surv(Time, DSD) ~ val.linpred.gg, data = data.val)
##
##   n= 61, number of events= 60
##
##                 coef exp(coef) se(coef)    z Pr(>|z|)
## val.linpred.gg 1.320     3.744    0.431 3.06   0.0022
##
##                exp(coef) exp(-coef) lower .95 upper .95
## val.linpred.gg      3.74      0.267      1.61      8.71
##
## Concordance= 0.659  (se = 0.044 )
## Rsquare= 0.144   (max possible= 0.998 )
## Likelihood ratio test= 9.48  on 1 df,   p=0.00208
## Wald test            = 9.39  on 1 df,   p=0.00219
## Score (logrank) test = 9.54  on 1 df,   p=0.00201
```

```r
summary(coxph(Surv(Time, DSD) ~ val.linpred.gg2, data.val))
```

```
## Call:
## coxph(formula = Surv(Time, DSD) ~ val.linpred.gg2, data = data.val)
##
##   n= 61, number of events= 60
##
##                 coef exp(coef) se(coef)    z Pr(>|z|)
## val.linpred.gg2 1.32      3.75     0.45 2.94   0.0033
##
##                 exp(coef) exp(-coef) lower .95 upper .95
## val.linpred.gg2      3.75      0.267      1.55      9.04
##
## Concordance= 0.642  (se = 0.044 )
## Rsquare= 0.133   (max possible= 0.998 )
## Likelihood ratio test= 8.7  on 1 df,   p=0.00319
## Wald test            = 8.63  on 1 df,   p=0.00331
## Score (logrank) test = 8.76  on 1 df,   p=0.00307
```

```r
summary(coxph(Surv(Time, DSD) ~ val.linpred.cph, data.val))
```

```
## Call:
## coxph(formula = Surv(Time, DSD) ~ val.linpred.cph, data = data.val)
##
##   n= 61, number of events= 60
##
##                  coef exp(coef) se(coef)    z Pr(>|z|)
## val.linpred.cph 1.192     3.295    0.338 3.53  0.00042
##
##                 exp(coef) exp(-coef) lower .95 upper .95
## val.linpred.cph      3.29      0.304       1.7      6.39
##
## Concordance= 0.649  (se = 0.044 )
## Rsquare= 0.177   (max possible= 0.998 )
## Likelihood ratio test= 11.8  on 1 df,   p=0.000578
## Wald test            = 12.4  on 1 df,   p=0.000421
## Score (logrank) test = 12.7  on 1 df,   p=0.000367
```

```r
summary(coxph(Surv(Time, DSD) ~ val.linpred.rsf, data.val))
```

```
## Call:
## coxph(formula = Surv(Time, DSD) ~ val.linpred.rsf, data = data.val)
##
##   n= 61, number of events= 60
##
##                    coef exp(coef) se(coef)    z Pr(>|z|)
## val.linpred.rsf 0.00672   1.00675  0.00195 3.45  0.00055
##
##                 exp(coef) exp(-coef) lower .95 upper .95
## val.linpred.rsf      1.01      0.993         1      1.01
##
## Concordance= 0.679  (se = 0.044 )
## Rsquare= 0.178   (max possible= 0.998 )
## Likelihood ratio test= 12  on 1 df,   p=0.000538
## Wald test            = 11.9  on 1 df,   p=0.000551
## Score (logrank) test = 12.1  on 1 df,   p=0.000494
```

```
anova(coxph(Surv(Time, DSD) ~ offset(val.linpred.gg) + val.linpred.gg, data.val))
```

```
## Analysis of Deviance Table
##  Cox model: response is Surv(Time, DSD)
## Terms added sequentially (first to last)
##
##                 loglik Chisq Df Pr(>|Chi|)
## NULL              -184
## val.linpred.gg    -184  0.55  1       0.46
```

```
anova(coxph(Surv(Time, DSD) ~ offset(val.linpred.gg2) + val.linpred.gg2, data.val))
```

```
## Analysis of Deviance Table
##  Cox model: response is Surv(Time, DSD)
## Terms added sequentially (first to last)
##
##                  loglik Chisq Df Pr(>|Chi|)
## NULL               -185
## val.linpred.gg2    -184  0.51  1       0.48
```

```
anova(coxph(Surv(Time, DSD) ~ offset(val.linpred.cph) + val.linpred.cph, data.val))
```

```
## Analysis of Deviance Table
##  Cox model: response is Surv(Time, DSD)
## Terms added sequentially (first to last)
##
##                  loglik Chisq Df Pr(>|Chi|)
## NULL               -183
## val.linpred.cph    -183  0.32  1       0.57
```

```
anova(coxph(Surv(Time, DSD) ~ offset(val.linpred.rsf) + val.linpred.rsf, data.val))
```

```
## Warning in fitter(X, Y, strats, offset, init, control, weights = weights, :  Ran out of
iterations and did not converge
## Error in fitter(X, Y, strats, offset, init, control, weights = weights, :  NA/NaN/Inf in
foreign function call (arg 6)
```

```
summary(coxph(Surv(Time, DSD) ~ offset(val.linpred.gg) + SexM + AgeCent + LocBody + SizeCent + A2 + A4,
```

```
## Call:
## coxph(formula = Surv(Time, DSD) ~ offset(val.linpred.gg) + SexM +
##     AgeCent + LocBody + SizeCent + A2 + A4, data = data.val)
##
##   n= 61, number of events= 60
##
##                 coef exp(coef) se(coef)     z Pr(>|z|)
## SexMTRUE     0.40127   1.49372  0.27817  1.44    0.149
## AgeCent     -0.02260   0.97766  0.01371 -1.65    0.099
## LocBodyTRUE  0.81060   2.24925  0.40567  2.00    0.046
## SizeCent    -0.00261   0.99740  0.00895 -0.29    0.771
## A2TRUE       0.69591   2.00553  0.50613  1.37    0.169
## A4TRUE       0.26205   1.29960  0.29377  0.89    0.372
##
##             exp(coef) exp(-coef) lower .95 upper .95
## SexMTRUE        1.494      0.669     0.866      2.58
## AgeCent         0.978      1.023     0.952      1.00
## LocBodyTRUE     2.249      0.445     1.016      4.98
## SizeCent        0.997      1.003     0.980      1.02
## A2TRUE          2.006      0.499     0.744      5.41
## A4TRUE          1.300      0.769     0.731      2.31
##
## Concordance= 0.687  (se = 0.044 )
## Rsquare= 0.152   (max possible= 0.998 )
## Likelihood ratio test= 10.1  on 6 df,   p=0.122
## Wald test            = 10.7  on 6 df,   p=0.0972
## Score (logrank) test = 11.2  on 6 df,   p=0.0815
```

```r
summary(coxph(Surv(Time, DSD) ~ offset(val.linpred.gg2) + SexM + AgeCent + LocBody + SizeCent + A2 + A4,
```

```
## Call:
## coxph(formula = Surv(Time, DSD) ~ offset(val.linpred.gg2) + SexM +
##     AgeCent + LocBody + SizeCent + A2 + A4, data = data.val)
##
##   n= 61, number of events= 60
##
##                 coef exp(coef) se(coef)     z Pr(>|z|)
## SexMTRUE     0.42688   1.53246  0.27817  1.53    0.125
## AgeCent     -0.02260   0.97766  0.01371 -1.65    0.099
## LocBodyTRUE  0.81060   2.24925  0.40567  2.00    0.046
## SizeCent    -0.00249   0.99751  0.00895 -0.28    0.781
## A2TRUE       0.70033   2.01442  0.50613  1.38    0.166
## A4TRUE       0.33020   1.39125  0.29377  1.12    0.261
##
##             exp(coef) exp(-coef) lower .95 upper .95
## SexMTRUE        1.532      0.653     0.888      2.64
## AgeCent         0.978      1.023     0.952      1.00
## LocBodyTRUE     2.249      0.445     1.016      4.98
## SizeCent        0.998      1.002     0.980      1.02
## A2TRUE          2.014      0.496     0.747      5.43
## A4TRUE          1.391      0.719     0.782      2.47
##
## Concordance= 0.687  (se = 0.044 )
## Rsquare= 0.162   (max possible= 0.998 )
```

```
## Likelihood ratio test= 10.8  on 6 df,   p=0.0943
## Wald test            = 11.4  on 6 df,   p=0.0767
## Score (logrank) test = 11.9  on 6 df,   p=0.0638
```

```r
summary(coxph(Surv(Time, DSD) ~ offset(val.linpred.cph) + SexM + AgeCent + LocBody + SizeCent + A2 + A4,
```

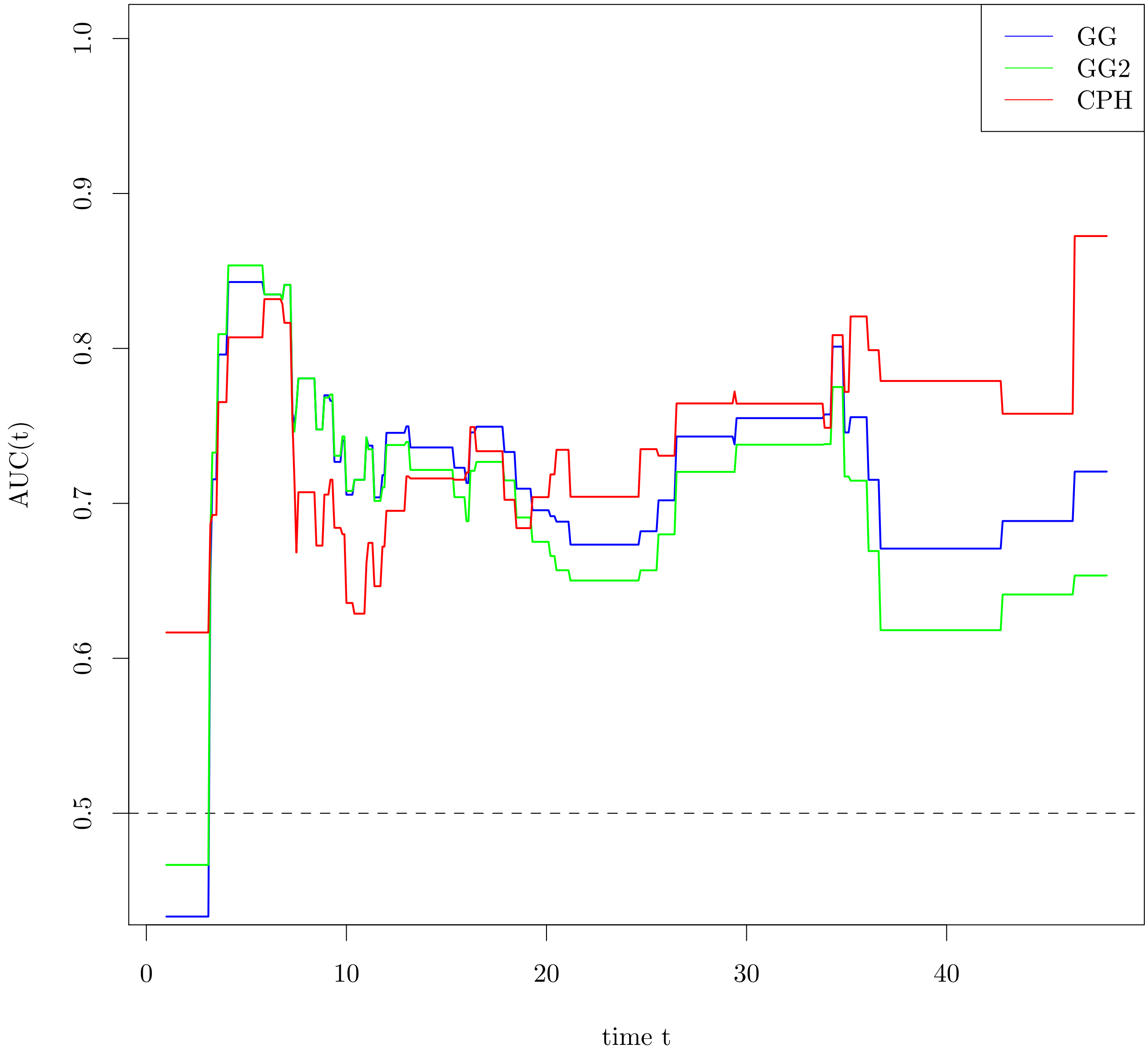
```
## Call:
## coxph(formula = Surv(Time, DSD) ~ offset(val.linpred.cph) + SexM +
##     AgeCent + LocBody + SizeCent + A2 + A4, data = data.val)
##
##   n= 61, number of events= 60
##
##                 coef exp(coef) se(coef)     z Pr(>|z|)
## SexMTRUE    -0.03303   0.96751  0.27817 -0.12    0.905
## AgeCent     -0.02260   0.97766  0.01371 -1.65    0.099
## LocBodyTRUE  0.81060   2.24925  0.40567  2.00    0.046
## SizeCent    -0.00544   0.99457  0.00895 -0.61    0.543
## A2TRUE       0.51021   1.66563  0.50613  1.01    0.313
## A4TRUE       0.12325   1.13117  0.29377  0.42    0.675
##
##             exp(coef) exp(-coef) lower .95 upper .95
## SexMTRUE        0.968      1.034     0.561      1.67
## AgeCent         0.978      1.023     0.952      1.00
## LocBodyTRUE     2.249      0.445     1.016      4.98
## SizeCent        0.995      1.005     0.977      1.01
## A2TRUE          1.666      0.600     0.618      4.49
## A4TRUE          1.131      0.884     0.636      2.01
##
## Concordance= 0.687  (se = 0.044 )
## Rsquare= 0.115   (max possible= 0.998 )
## Likelihood ratio test= 7.48  on 6 df,   p=0.279
## Wald test            = 8.05  on 6 df,   p=0.234
## Score (logrank) test = 8.41  on 6 df,   p=0.209
```

```r
summary(coxph(Surv(Time, DSD) ~ offset(val.linpred.rsf) + SexM + AgeCent + LocBody + SizeCent + A2 + A4,
```

```
## Warning in fitter(X, Y, strats, offset, init, control, weights = weights, :  Ran out of
iterations and did not converge
## Error in fitter(X, Y, strats, offset, init, control, weights = weights, :  NA/NaN/Inf in
foreign function call (arg 6)
```

TD-ROC AUC

```r
temp.times = seq(0.1, 48, 0.1)
temp.gg = timeROC(data.val$Time/365.25*12, data.val$DSD, val.linpred.gg, cause = 1, times = temp.times,
temp.gg2 = timeROC(data.val$Time/365.25*12, data.val$DSD, val.linpred.gg2, cause = 1, times = temp.times
temp.cph = timeROC(data.val$Time/365.25*12, data.val$DSD, val.linpred.cph, cause = 1, times = temp.times
plotAUCcurve(temp.gg, conf.int = FALSE, add = FALSE, col = "blue")
plotAUCcurve(temp.gg2, conf.int = FALSE, add = TRUE, col = "green")
plotAUCcurve(temp.cph, conf.int = FALSE, add = TRUE, col = "red")
legend("topright", legend = c("GG", "GG2", "CPH"), col = c("blue", "green", "red"), lty = "solid")
```



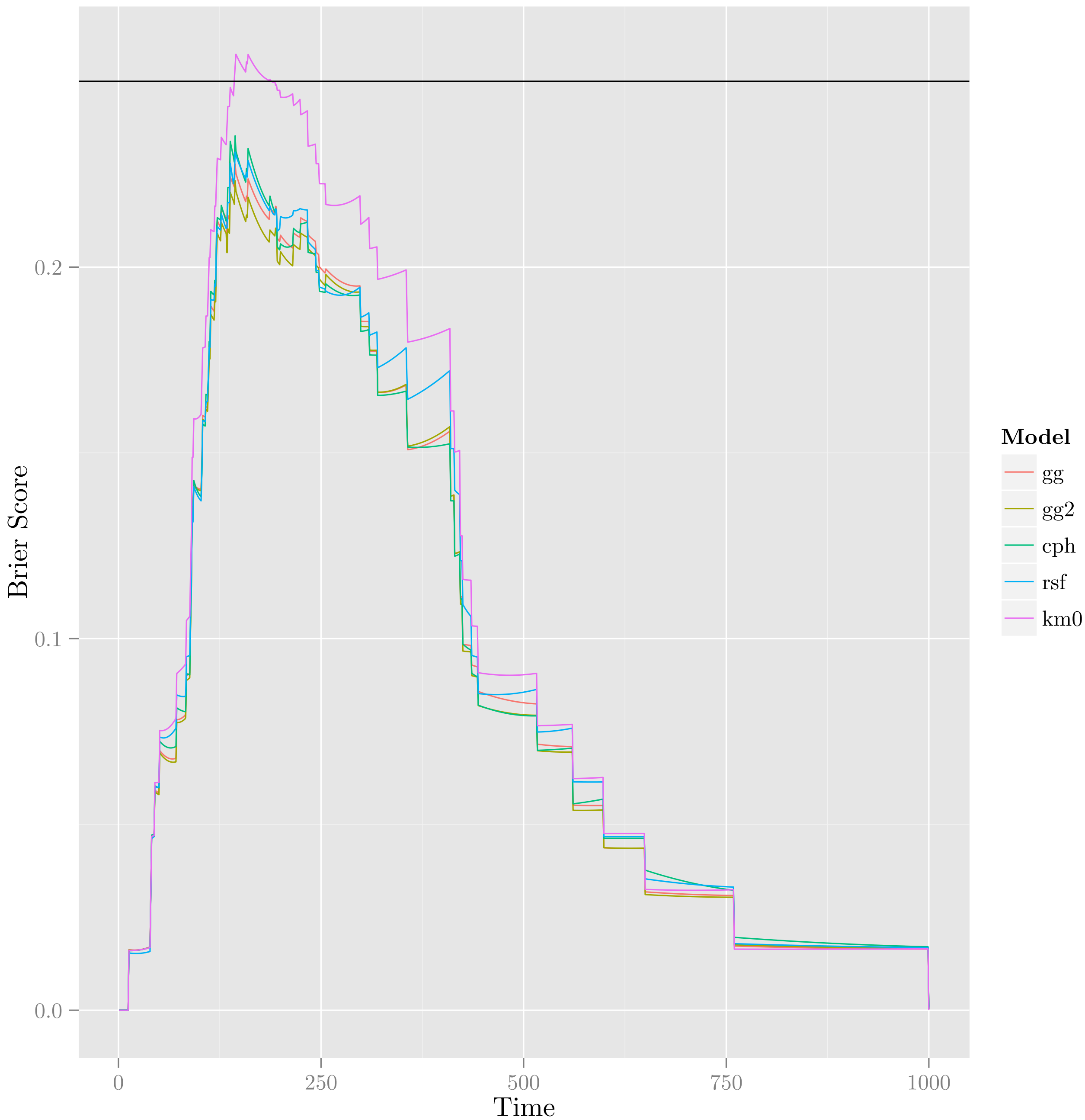

Decision curve analysis.

```
temp.data = data.frame(Time = data.val$Time, DSD = data.val$DSD*1,
    gg.1 = 1-val.prob.gg[val.prob.times == 365,], gg.2 = 1-val.prob.gg[val.prob.times == 365*2,], gg.3 =
    gg2.1 = 1-val.prob.gg2[val.prob.times == 365,], gg2.2 = 1-val.prob.gg2[val.prob.times == 365*2,], gg
    cph.1 = 1-val.prob.cph[val.prob.times == 365,], cph.2 = 1-val.prob.cph[val.prob.times == 365*2,], cp
    rsf.1 = 1-val.prob.rsf[val.prob.times == 365,], rsf.2 = 1-val.prob.rsf[val.prob.times == 365*2,], rs
stdca(data = temp.data, outcome = "DSD", ttoutcome = "Time", predictors = c("gg.1", "cph.1", "rsf.1"), t

## Error in eval(expr, envir, enclos):  could not find function "stdca"

stdca(data = temp.data, outcome = "DSD", ttoutcome = "Time", predictors = c("gg.2", "cph.2", "rsf.2"), t

## Error in eval(expr, envir, enclos):  could not find function "stdca"

stdca(data = temp.data, outcome = "DSD", ttoutcome = "Time", predictors = c("gg.3", "cph.3", "rsf.3"), t

## Error in eval(expr, envir, enclos):  could not find function "stdca"
```

Evaluate IBS point estimates. BS paths over time on bootstrap samples of the holdout set.

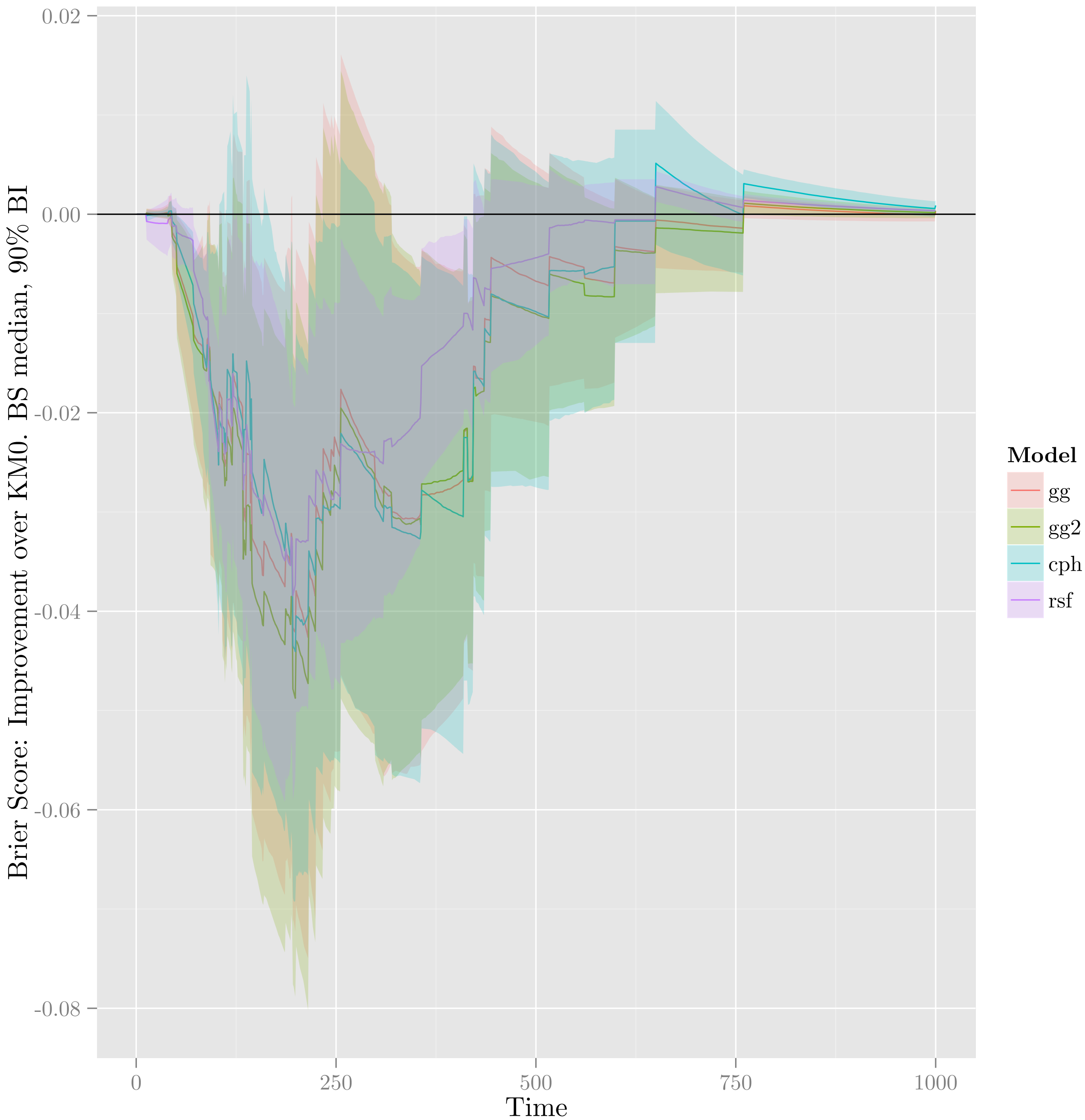
```
set.seed(20150111)
ibs_eval_times = calcIBS(Surv(data.val$Time, data.val$DSD), ibs_preds_gg, ibs_times, max(data.val$Time))
# bsc_boot2 = lapply(ibs_preds_all, function(preds) boot(data.val, statistic = function(d, i) calcIBS(Su
# bsc_boot2ci = lapply(bsc_boot2, function(single_boot) t(sapply(1:length(ibs_eval_times), function(time
#  temp = try(boot.ci(single_boot, index = time_index, type = "bca")£bca, silent = TRUE)
#  if (class(temp) == "try-error" || is.null(temp)) { temp = rep(NA, 5) }
#  temp })))
bsc_boots = laply(1:500, function(i) {
        if (i %% 50 == 0)       { message(i) }
        boot_samp = sample.int(nrow(data.val), replace = TRUE)
        gg = calcIBS(Surv(data.val$Time, data.val$DSD)[boot_samp,], ibs_preds_gg[boot_samp,], ibs_times,
        gg2 = calcIBS(Surv(data.val$Time, data.val$DSD)[boot_samp,], ibs_preds_gg2[boot_samp,], ibs_time
        cph = calcIBS(Surv(data.val$Time, data.val$DSD)[boot_samp,], ibs_preds_cph[boot_samp,], ibs_time
        rsf = calcIBS(Surv(data.val$Time, data.val$DSD)[boot_samp,], ibs_preds_rsf[boot_samp,], ibs_time
        km0 = calcIBS(Surv(data.val$Time, data.val$DSD)[boot_samp,], ibs_preds_km0[boot_samp,], ibs_time
        rbind(gg, gg2, cph, rsf, km0)
})

## 50
## 100
## 150
## 200
## 250
## 300
## 350
## 400
## 450
## 500
```

```
temp = sapply(list(gg = ibs_preds_gg, gg2 = ibs_preds_gg2, cph = ibs_preds_cph, rsf = ibs_preds_rsf, km0
temp = melt(temp)
colnames(temp) = c("Time", "Model", "BS")
ggplot(temp, aes(x = Time, y = BS, colour = Model)) + geom_line() + ylab("Brier Score") + geom_hline(yin
```

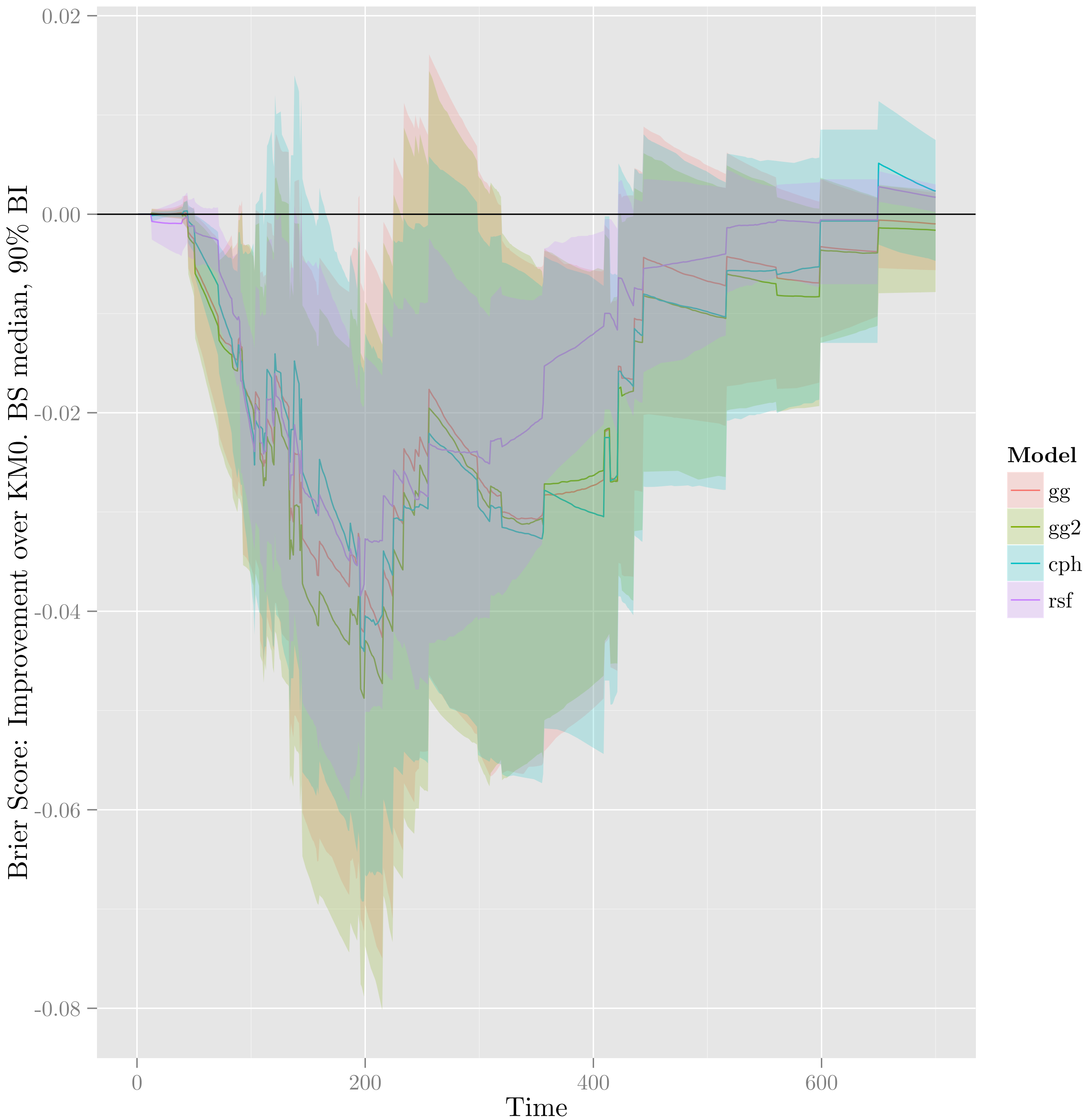



```
temp = melt(aaply(bsc_boots, 2:3, quantile, probs = c(0.05, 0.5, 0.95)))
colnames(temp) = c("Model", "Time", "Quantile", "Value")
temp$Quantile = paste("Q", gsub("%", "", temp$Quantile), sep = "")
temp = dcast(temp, Model + Time ~ Quantile, value.var = "Value")
ggplot(temp, aes(x = Time, y = Q50, ymin = Q5, ymax = Q95, colour = Model, fill = Model)) + geom_line()
```

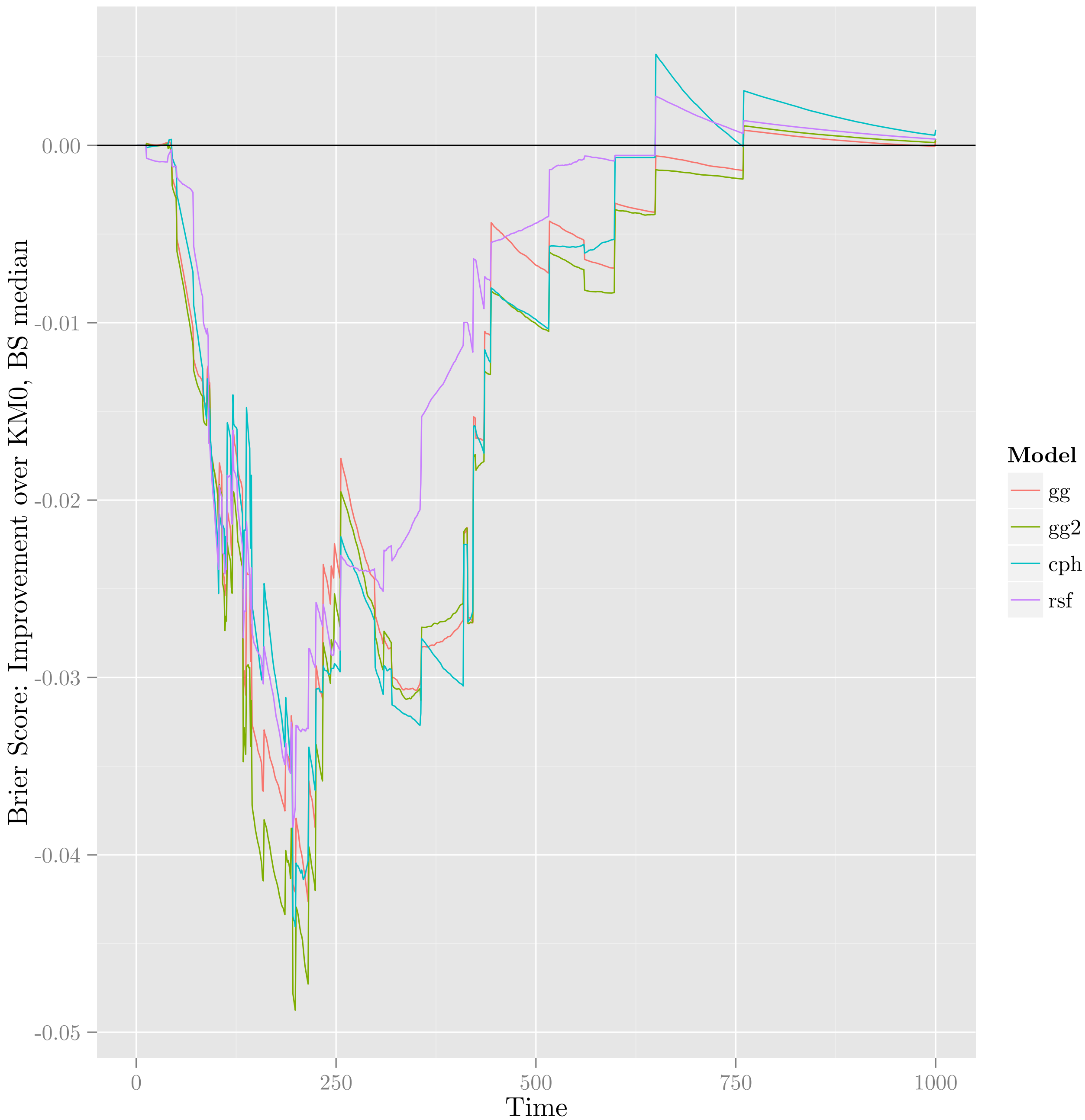

```
bsc_boots_diff = aaply(bsc_boots, 2, function(x) x - bsc_boots[,5,])[1:4,,]
temp = melt(aaply(bsc_boots_diff, c(1,3), quantile, probs = c(0.05, 0.5, 0.95)))
colnames(temp) = c("Model", "Time", "Quantile", "Value")
temp$Quantile = paste("Q", gsub("%", "", temp$Quantile), sep = "")
temp = dcast(temp, Model + Time ~ Quantile, value.var = "Value")
ggplot(temp, aes(x = Time, y = Q50, ymin = Q5, ymax = Q95, colour = Model, fill = Model)) + geom_line()
```



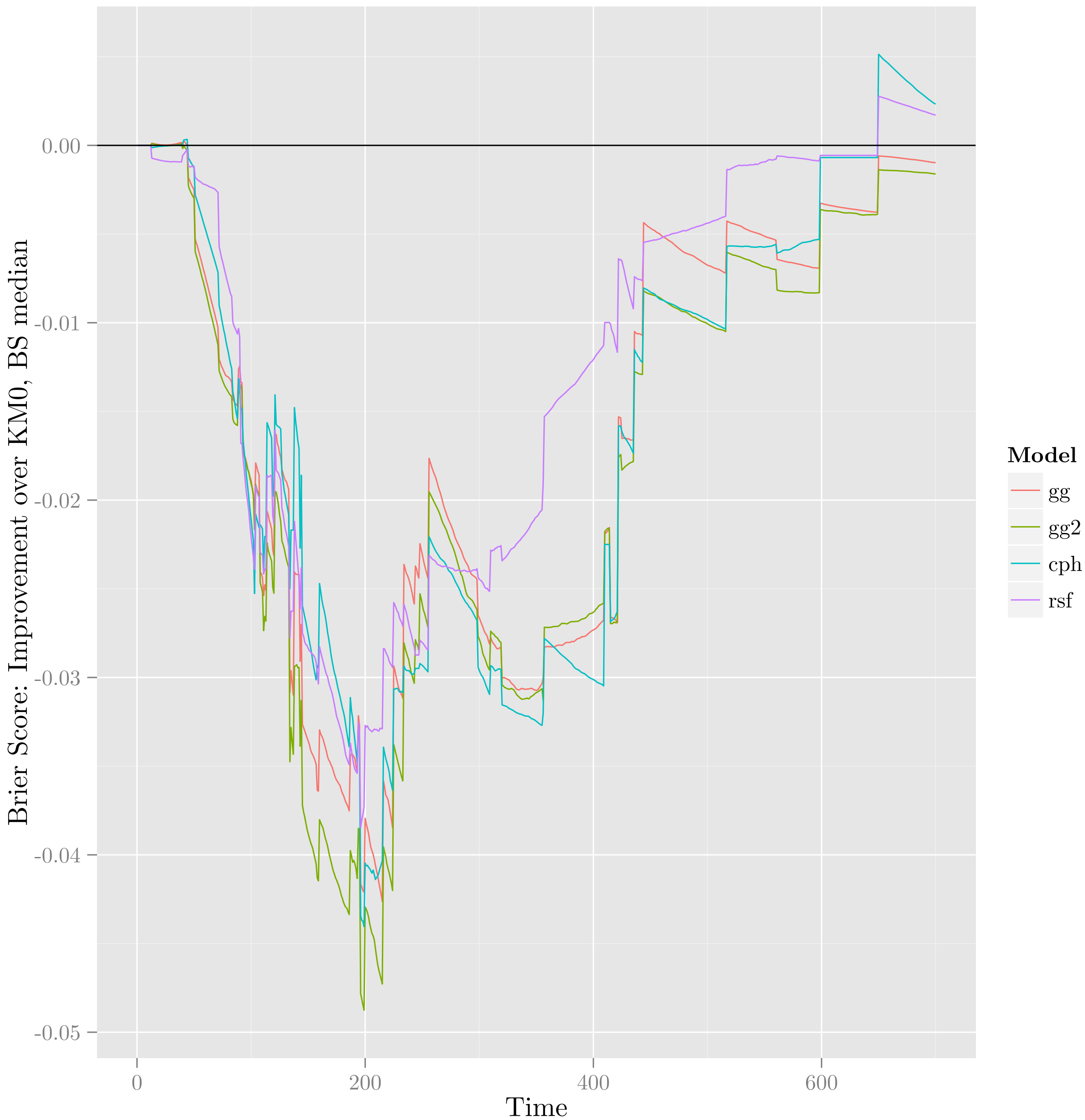

```
ggplot(temp, aes(x = Time, y = Q50, ymin = Q5, ymax = Q95, colour = Model, fill = Model)) + geom_line()

## Warning:  Removed 1200 rows containing missing values (geom_path).
```




```
ggplot(temp, aes(x = Time, y = Q50, colour = Model)) + geom_line() + ylab("Brier Score: Improvement over
```




```
ggplot(temp, aes(x = Time, y = Q50, colour = Model)) + geom_line() + ylab("Brier Score: Improvement over

## Warning:  Removed 1200 rows containing missing values (geom_path).
```

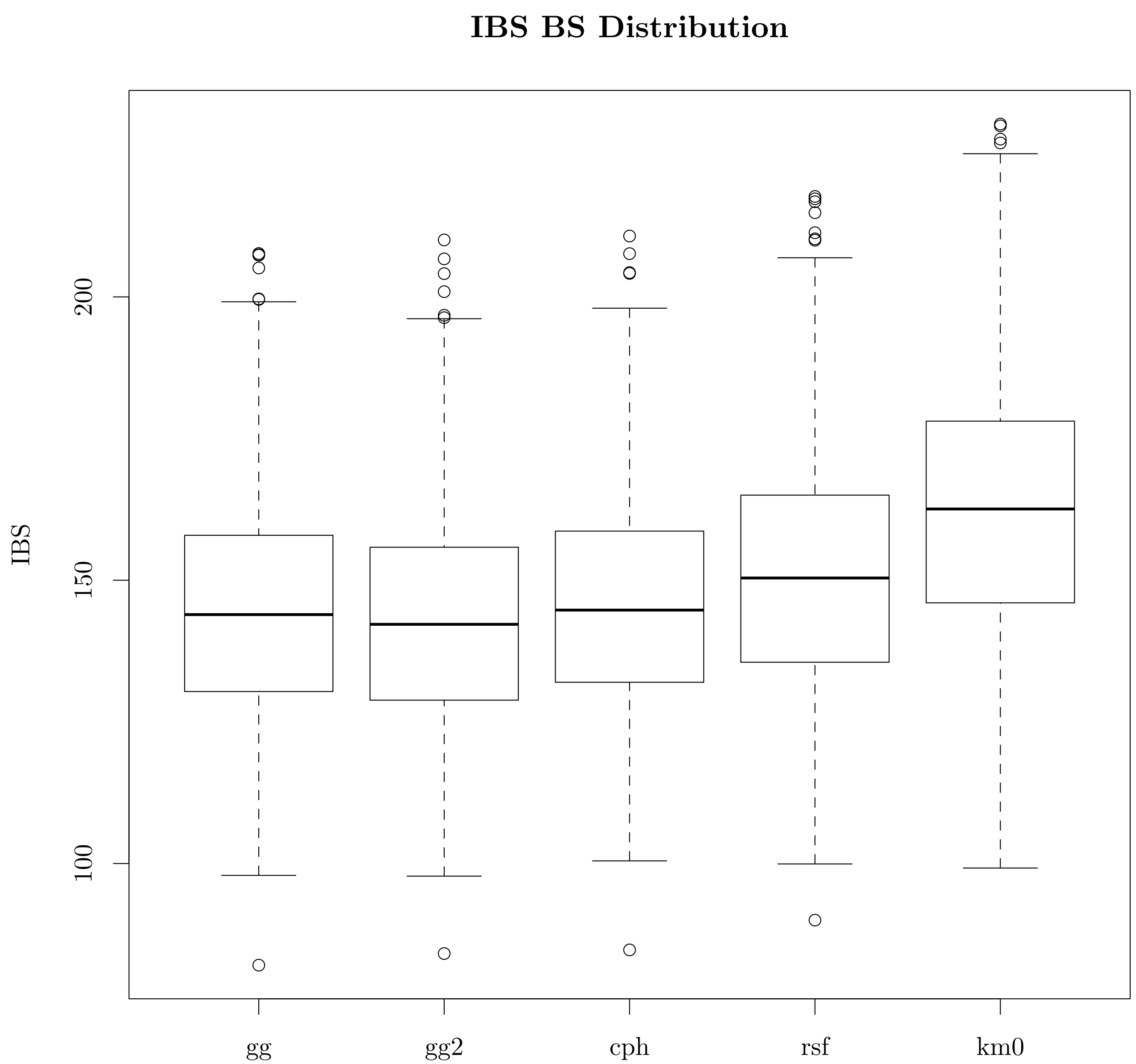



IBS comparisons.

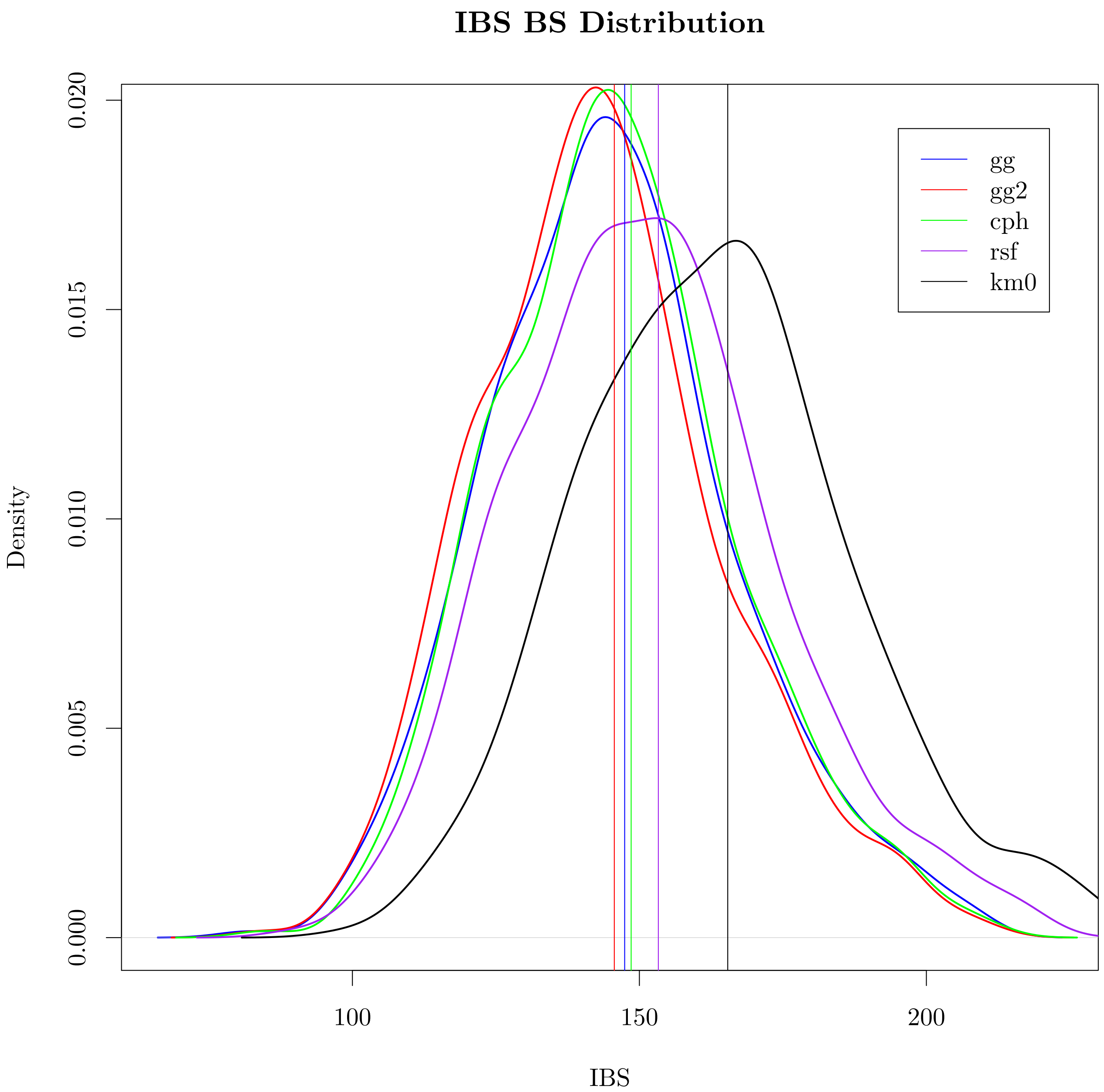
```
set.seed(20150111)
ibsc_boots = t(sapply(1:5e2, function(i) {
        if (i %% 5e1 == 0)      { message(i) }
        boot_samp = sample.int(nrow(data.val), replace = TRUE)
        gg = calcIBS(Surv(data.val$Time, data.val$DSD)[boot_samp,], ibs_preds_gg[boot_samp,], ibs_times,
        gg2 = calcIBS(Surv(data.val$Time, data.val$DSD)[boot_samp,], ibs_preds_gg2[boot_samp,], ibs_time
        cph = calcIBS(Surv(data.val$Time, data.val$DSD)[boot_samp,], ibs_preds_cph[boot_samp,], ibs_time
        rsf = calcIBS(Surv(data.val$Time, data.val$DSD)[boot_samp,], ibs_preds_rsf[boot_samp,], ibs_time
        km0 = calcIBS(Surv(data.val$Time, data.val$DSD)[boot_samp,], ibs_preds_km0[boot_samp,], ibs_time
        c(gg, gg2, cph, rsf, km0)
}))
```

```
## 50
## 100
## 150
## 200
## 250
## 300
## 350
## 400
## 450
## 500
colnames(ibsc_boots) = c("gg", "gg2", "cph", "rsf", "km0")
```

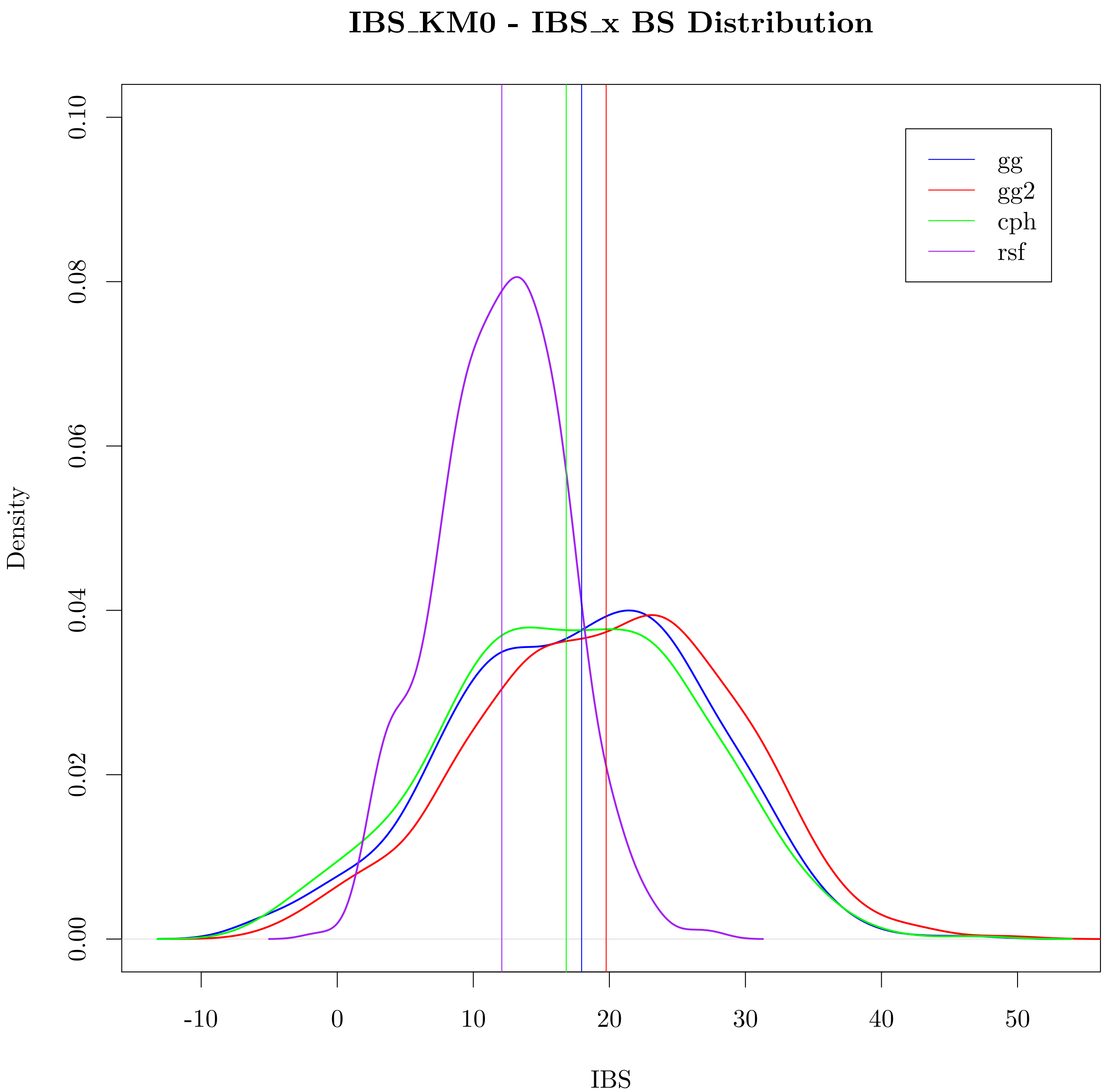
```
calcIBS(Surv(data.val$Time, data.val$DSD), ibs_preds_gg, ibs_times, max(data.val$Time))$ibs

## [1] 147.4

calcIBS(Surv(data.val$Time, data.val$DSD), ibs_preds_gg2, ibs_times, max(data.val$Time))$ibs

## [1] 145.6

calcIBS(Surv(data.val$Time, data.val$DSD), ibs_preds_cph, ibs_times, max(data.val$Time))$ibs

## [1] 148.6

calcIBS(Surv(data.val$Time, data.val$DSD), ibs_preds_rsf, ibs_times, max(data.val$Time))$ibs

## [1] 153.3

calcIBS(Surv(data.val$Time, data.val$DSD), ibs_preds_km0, ibs_times, max(data.val$Time))$ibs

## [1] 165.4

boxplot(ibsc_boots, main = "IBS BS Distribution", ylab = "IBS")
```




```
plot(density(ibsc_boots[,1]), col = "blue", lwd = 2, main = "IBS BS Distribution", xlab = "IBS")
lines(density(ibsc_boots[,2]), col = "red", lwd = 2)
lines(density(ibsc_boots[,3]), col = "green", lwd = 2)
lines(density(ibsc_boots[,4]), col = "purple", lwd = 2)
lines(density(ibsc_boots[,5]), col = "black", lwd = 2)
abline(v = calcIBS(Surv(data.val$Time, data.val$DSD), ibs_preds_gg, ibs_times, max(data.val$Time))$ibs,
abline(v = calcIBS(Surv(data.val$Time, data.val$DSD), ibs_preds_gg2, ibs_times, max(data.val$Time))$ibs
abline(v = calcIBS(Surv(data.val$Time, data.val$DSD), ibs_preds_cph, ibs_times, max(data.val$Time))$ibs
abline(v = calcIBS(Surv(data.val$Time, data.val$DSD), ibs_preds_rsf, ibs_times, max(data.val$Time))$ibs
abline(v = calcIBS(Surv(data.val$Time, data.val$DSD), ibs_preds_km0, ibs_times, max(data.val$Time))$ibs
legend("topright", legend = c("gg", "gg2", "cph", "rsf", "km0"), col = c("blue", "red", "green", "purple
```




```
plot(density(ibsc_boots[,5] - ibsc_boots[,1]), col = "blue", lwd = 2, main = "IBS\\_KM0 - IBS\\_x BS Dis
lines(density(ibsc_boots[,5] - ibsc_boots[,2]), col = "red", lwd = 2)
lines(density(ibsc_boots[,5] - ibsc_boots[,3]), col = "green", lwd = 2)
lines(density(ibsc_boots[,5] - ibsc_boots[,4]), col = "purple", lwd = 2)
abline(v = (calcIBS(Surv(data.val$Time, data.val$DSD), ibs_preds_km0, ibs_times, max(data.val$Time))$ibs
abline(v = (calcIBS(Surv(data.val$Time, data.val$DSD), ibs_preds_km0, ibs_times, max(data.val$Time))$ibs
abline(v = (calcIBS(Surv(data.val$Time, data.val$DSD), ibs_preds_km0, ibs_times, max(data.val$Time))$ibs
abline(v = (calcIBS(Surv(data.val$Time, data.val$DSD), ibs_preds_km0, ibs_times, max(data.val$Time))$ibs
legend("topright", legend = c("gg", "gg2", "cph", "rsf"), col = c("blue", "red", "green", "purple"), lty
```




Do some proper BCA bootstrapping on the differences, just as a double-check test.

```
set.seed(20150111)
ibsc_boots2 = boot(data.val, statistic = function(d, i) {
        gg = calcIBS(Surv(d$Time, d$DSD)[i,], ibs_preds_gg[i,], ibs_times, max(d$Time[i]))$ibs
        gg2 = calcIBS(Surv(d$Time, d$DSD)[i,], ibs_preds_gg2[i,], ibs_times, max(d$Time[i]))$ibs
        cph = calcIBS(Surv(d$Time, d$DSD)[i,], ibs_preds_cph[i,], ibs_times, max(d$Time[i]))$ibs
        rsf = calcIBS(Surv(d$Time, d$DSD)[i,], ibs_preds_rsf[i,], ibs_times, max(d$Time[i]))$ibs
        km0 = calcIBS(Surv(d$Time, d$DSD)[i,], ibs_preds_km0[i,], ibs_times, max(d$Time[i]))$ibs
        c(gg - km0, gg2 - km0, cph - km0, rsf - km0, gg - rsf, gg2 - rsf, cph - rsf, gg - cph, gg2 - cph
}, R = 500)
ibsc_boots2_ci = t(sapply(1:length(ibsc_boots2$t0), function(i) boot.ci(ibsc_boots2, index = i, type = '
rownames(ibsc_boots2_ci) = c("gg-km0", "gg2-km0", "cph-km0", "rsf-km0", "gg-rsf", "gg2-rsf", "cph-rsf",
colnames(ibsc_boots2_ci) = c("level", "orderi1", "orderi2", "lci", "uci")
ibsc_boots2
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = data.val, statistic = function(d, i) {
##     gg = calcIBS(Surv(d$Time, d$DSD)[i, ], ibs_preds_gg[i, ],
##         ibs_times, max(d$Time[i]))$ibs
##     gg2 = calcIBS(Surv(d$Time, d$DSD)[i, ], ibs_preds_gg2[i,
##         ], ibs_times, max(d$Time[i]))$ibs
##     cph = calcIBS(Surv(d$Time, d$DSD)[i, ], ibs_preds_cph[i,
##         ], ibs_times, max(d$Time[i]))$ibs
##     rsf = calcIBS(Surv(d$Time, d$DSD)[i, ], ibs_preds_rsf[i,
##         ], ibs_times, max(d$Time[i]))$ibs
##     km0 = calcIBS(Surv(d$Time, d$DSD)[i, ], ibs_preds_km0[i,
##         ], ibs_times, max(d$Time[i]))$ibs
##     c(gg - km0, gg2 - km0, cph - km0, rsf - km0, gg - rsf, gg2 -
##         rsf, cph - rsf, gg - cph, gg2 - cph, gg - gg2)
## }, R = 500)
##
##
## Bootstrap Statistics :
##      original   bias    std. error
## t1*   -17.960  0.16202       9.092
## t2*   -19.764  0.08589       9.303
## t3*   -16.830 -0.34992       9.293
## t4*   -12.092  0.06313       4.847
## t5*    -5.868  0.09888       4.847
## t6*    -7.672  0.02275       5.138
## t7*    -4.738 -0.41305       5.075
## t8*    -1.129  0.51193       2.188
## t9*    -2.934  0.43580       1.239
## t10*    1.805  0.07613       2.012
```

```
ibsc_boots2_ci
```

```
##         level orderi1 orderi2     lci      uci
## gg-km0   0.95   12.62   488.6 -36.586  0.04987
## gg2-km0  0.95   13.26   489.1 -38.095 -0.93649
## cph-km0  0.95   15.77   490.9 -35.135  1.22396
## rsf-km0  0.95   15.92   491.0 -21.344 -1.94556
## gg-rsf   0.95   16.84   491.8 -13.848  4.78985
## gg2-rsf  0.95   13.27   489.1 -17.593  2.48774
## cph-rsf  0.95   16.67   491.4 -14.727  4.96039
## gg-cph   0.95    6.57   477.5  -5.556  3.09545
## gg2-cph  0.95    1.63   442.8  -5.680 -1.02330
## gg-gg2   0.95   13.43   489.4  -2.239  5.99125
```

All models perform equivalently on the validation set. Select the simplest: gg.

Final model fitting:

```
data.all = rbind(data[colnames(data.val)], data.val)
head(data.all)
```

```
##           Time  DSD  SexM AgeCent LocBody SizeCent    A2    A4
```

```
## NSWPCN_4   937 TRUE  TRUE     -16   FALSE       -1 FALSE  TRUE
## NSWPCN_7   247 TRUE FALSE      -1   FALSE       -2 FALSE  TRUE
## NSWPCN_10  177 TRUE  TRUE      -9   FALSE       10 FALSE  TRUE
## NSWPCN_13  247 TRUE FALSE     -19    TRUE       20 FALSE  TRUE
## NSWPCN_20  256 TRUE FALSE      -8   FALSE        0 FALSE  TRUE
## NSWPCN_21  763 TRUE FALSE      -1   FALSE       -2 FALSE FALSE
```

```
fit.final.gg = flexsurvreg(Surv(Time, DSD) ~ SexM + SizeCent + A2 + A4,
        anc = list(
                sigma = ~ SexM,
                Q = ~ SexM),
        data = data.all, dist = "gengamma")
fit.final.gg2 = flexsurvreg(Surv(Time, DSD) ~ SexM + SizeCent + A2 + A4 + I(SexM == FALSE & A2 == FALSE
    anc = list(
        sigma = ~ SexM,
        Q = ~ SexM),
    data = data.all, dist = "gengamma")
fit.final.cph = coxph(Surv(Time, DSD) ~ strata(SexM) + SizeCent + A2 + A4, data = data.all, x = TRUE, y
set.seed(20150111)
fit.final.rsf = rfsrc(Surv(Time, DSD) ~ SexM + AgeCent + LocBody + SizeCent + A2 + A4, data = data.all,
fit.final.km0 = survfit(Surv(Time, DSD) ~ 1, data.all)
saveRDS(list(gg = fit.final.gg, km0 = fit.final.km0, gg2 = fit.final.gg2, cph = fit.final.cph, rsf = fit
```

# 8 Session information

```
sessionInfo()

## R version 3.1.1 (2014-07-10)
## Platform: x86_64-unknown-linux-gnu (64-bit)
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8          LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8           LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8       LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8          LC_NAME=en_US.UTF-8
##  [9] LC_ADDRESS=en_US.UTF-8        LC_TELEPHONE=en_US.UTF-8
## [11] LC_MEASUREMENT=en_US.UTF-8    LC_IDENTIFICATION=en_US.UTF-8
##
## attached base packages:
## [1] parallel  splines   methods   stats     graphics  grDevices utils
## [8] datasets  base
##
## other attached packages:
##  [1] timeROC_0.2          timereg_1.8.6         mvtnorm_1.0-1
##  [4] pec_2.4.4            boot_1.3-13           MASS_7.3-35
##  [7] ggplot2_1.0.0        plyr_1.8.1            reshape2_1.4
## [10] randomForestSRC_1.5.5 flexsurv_0.5         glmulti_1.0.7
## [13] rJava_0.9-6          survival_2.37-7       tikzDevice_0.7.0
## [16] filehash_2.2-2       knitr_1.8
##
## loaded via a namespace (and not attached):
##  [1] codetools_0.2-9  colorspace_1.2-4 deSolve_1.11     digest_0.6.4
```

```
##  [5] evaluate_0.5.5   foreach_1.4.2    formatR_1.0      grid_3.1.1
##  [9] gtable_0.1.2     highr_0.4        iterators_1.0.7  labeling_0.3
## [13] lava_1.3         muhaz_1.2.6      munsell_0.4.2    prodlim_1.5.1
## [17] proto_0.3-10     Rcpp_0.11.3      scales_0.2.4     stringr_0.6.2
## [21] tools_3.1.1
```