

NSWPCN Predictor Training

January 16, 2015

1 Preparation

```
library(survival)
library(glmulti)

## Loading required package: rJava

library(flexsurv)
library(randomForestSRC)

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

library(MASS)
library(boot)

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)/5))
sel.val = 1:nrow(data) %in% sel.val
mean(sel.val)

## [1] 0.1967

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

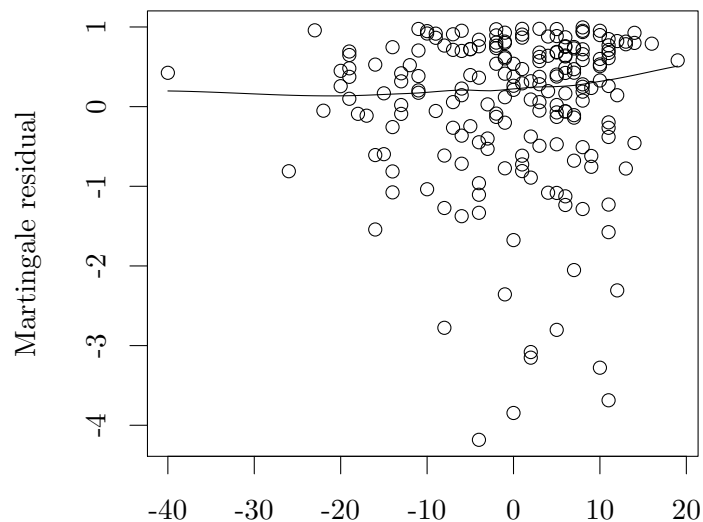
4 EDA

Use the CPH model as a convenient framework for EDA.

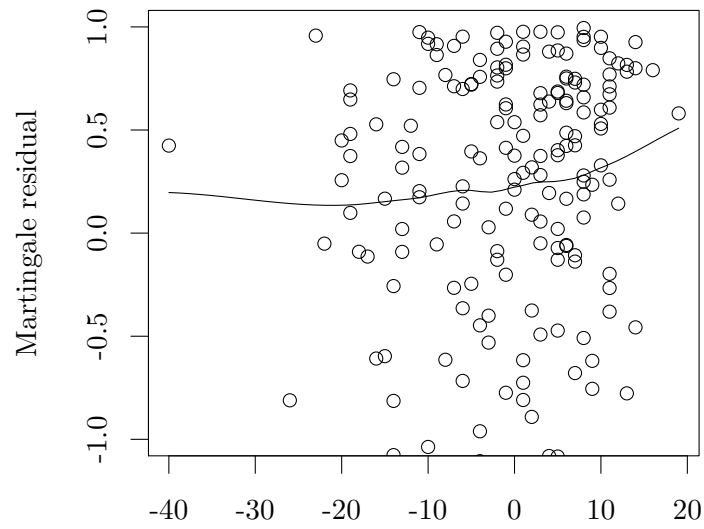
4.1 Functional form

Investigate functional form with martingale residuals.

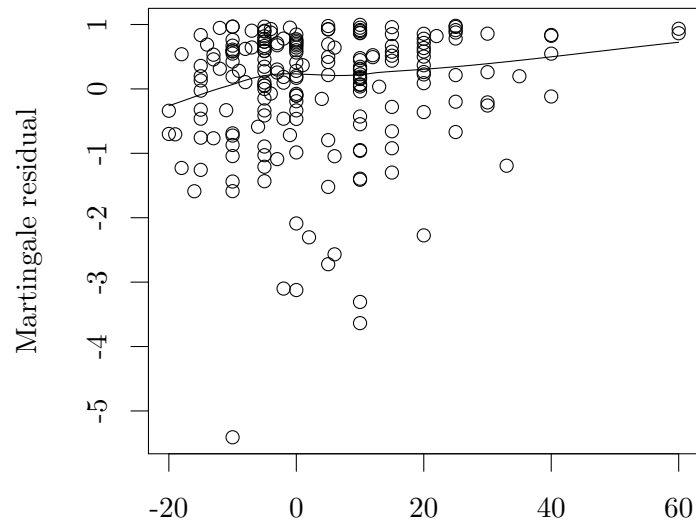
```
fit.cph.NoAge = coxph(Surv(Time, DSD) ~ SexM + LocBody + SizeCent + A2 + A4, data = data)
scatter.smooth(data$AgeCent, resid(fit.cph.NoAge, type = "martingale"), xlab = "", ylab = "Martingale re
```



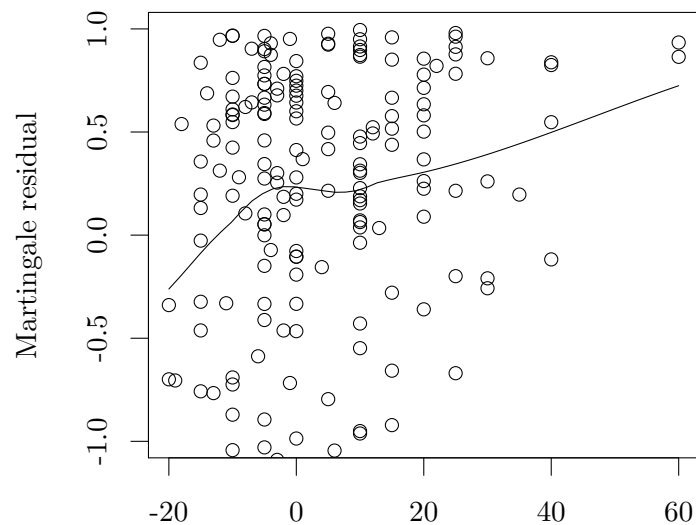
```
scatter.smooth(data$AgeCent, resid(fit.cph.NoAge, type = "martingale"), xlab = "", ylab = "Martingale re
```



```
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: $SizeCent + SizeCentI(SizeCent > 0) \equiv SizeCent + SizeCent_+$

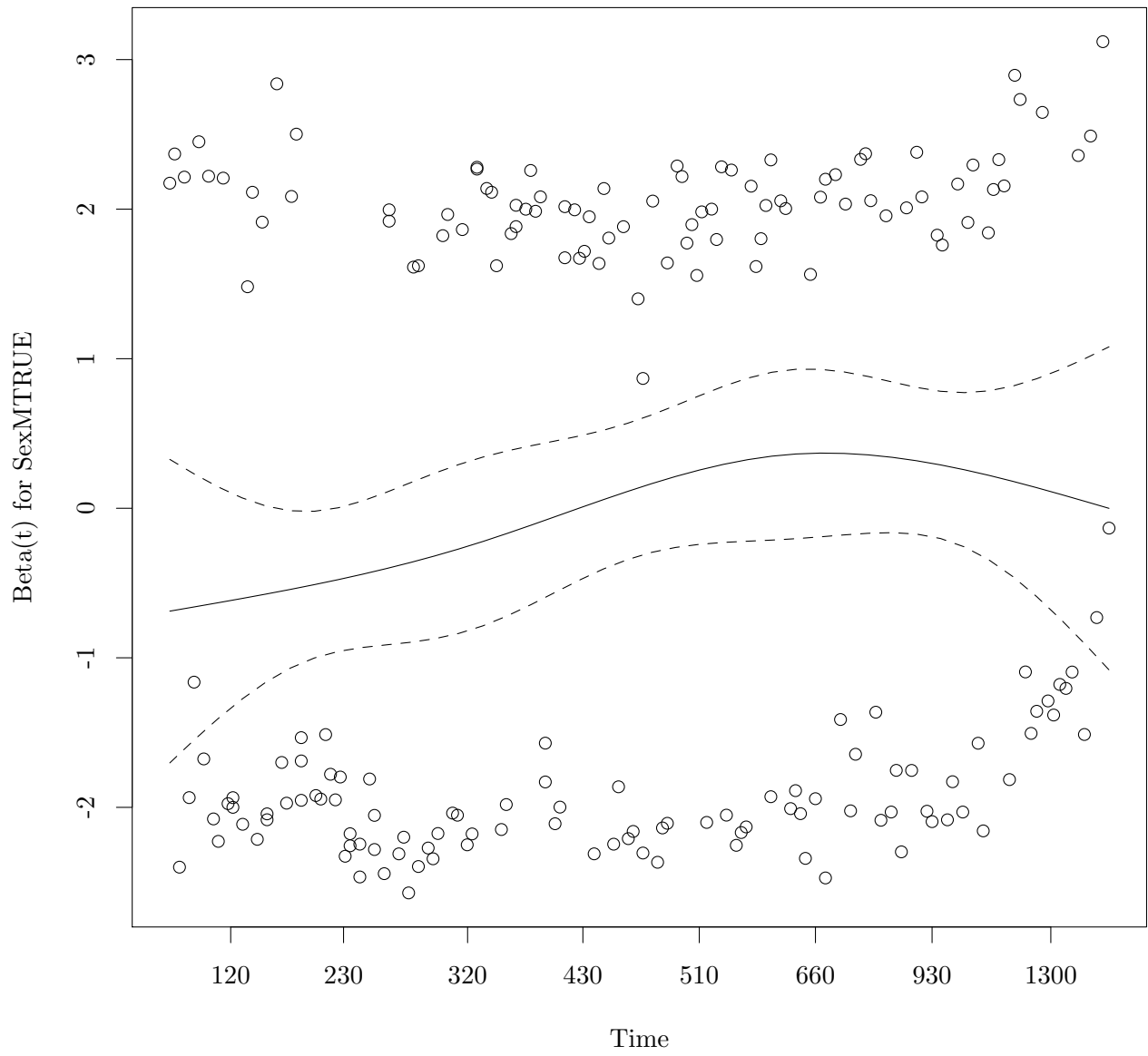
```
data$SizeSmall = data$SizeCent * (data$SizeCent < 0)
data$AgeCent2 = data$AgeCent^2
```

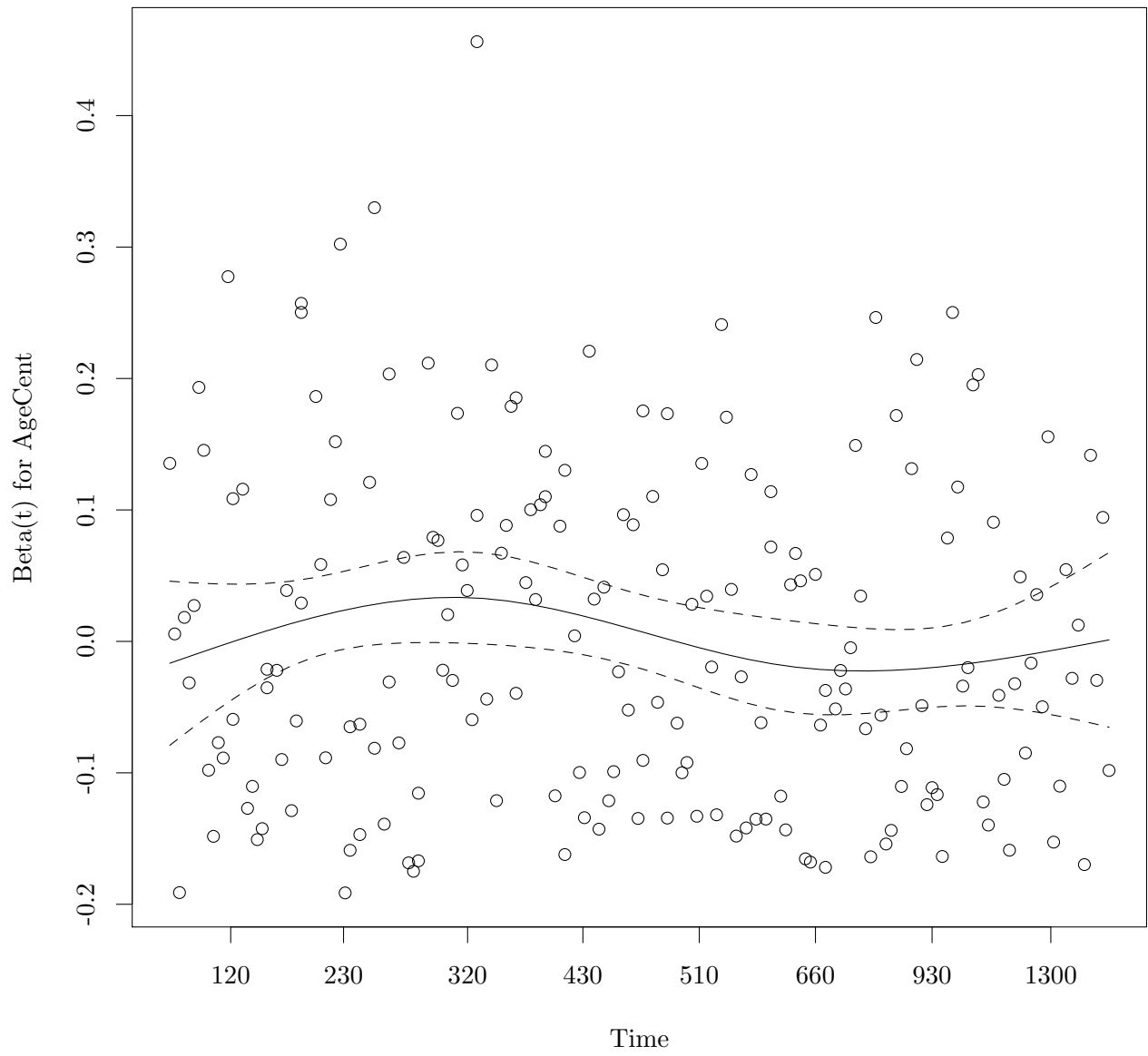
4.2 PH assumption: full model

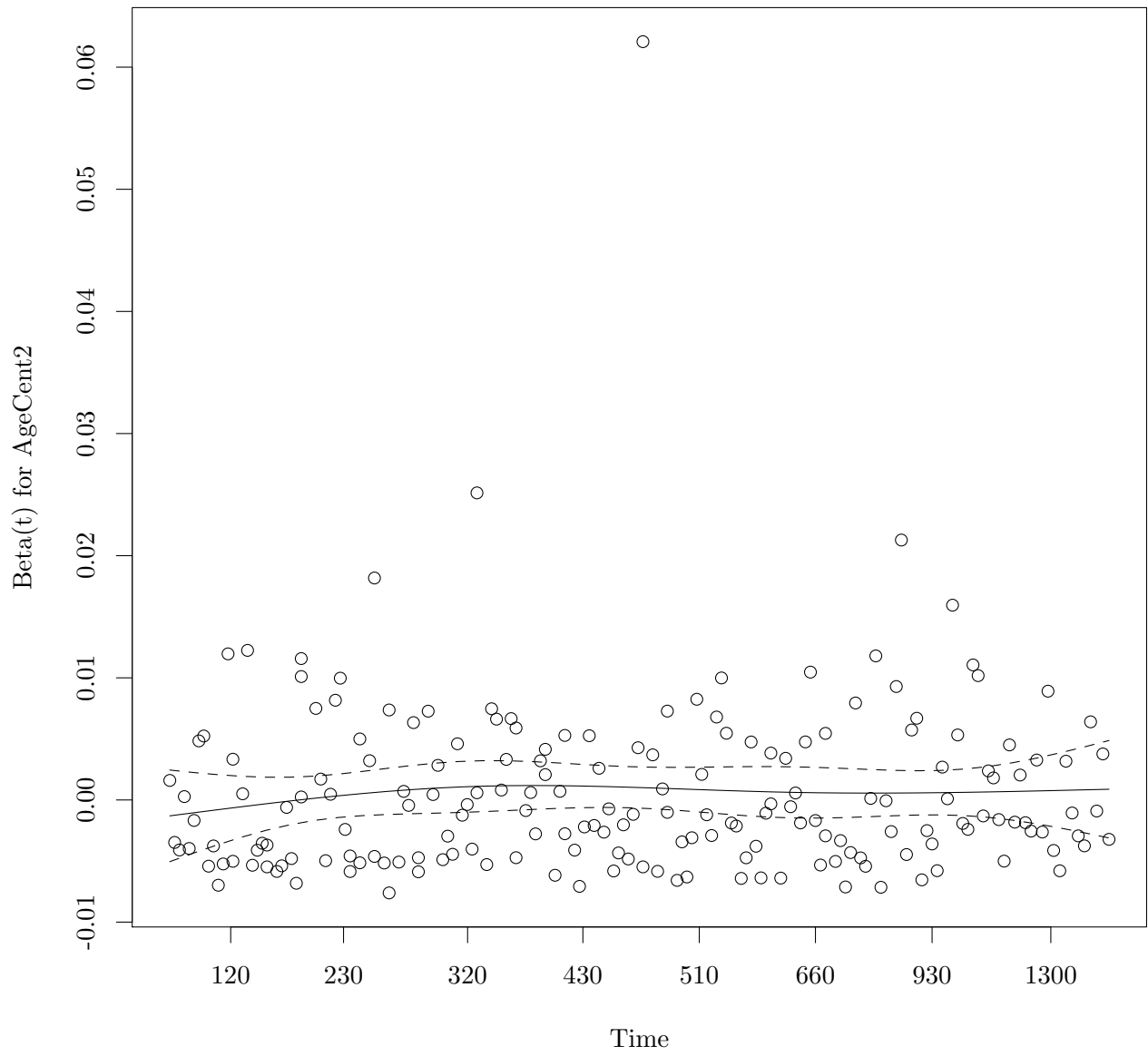
```
fit.cph = coxph(Surv(Time, DSD) ~ SexM + AgeCent + AgeCent2 + LocBody + SizeCent + SizeSmall + A2 + A4,
cox.zph(fit.cph))

##           rho    chisq      p
## SexMTRUE    0.1520  4.2830 0.0385
## AgeCent     -0.0897  1.5736 0.2097
## AgeCent2     0.0392  0.2733 0.6012
## LocBodyTRUE -0.1287  2.7244 0.0988
## SizeCent     0.0088  0.0168 0.8970
## SizeSmall    -0.0592  0.6803 0.4095
## A2TRUE       0.0533  0.5483 0.4590
## A4TRUE      -0.0596  0.6487 0.4206
## GLOBAL              NA 14.0077 0.0816

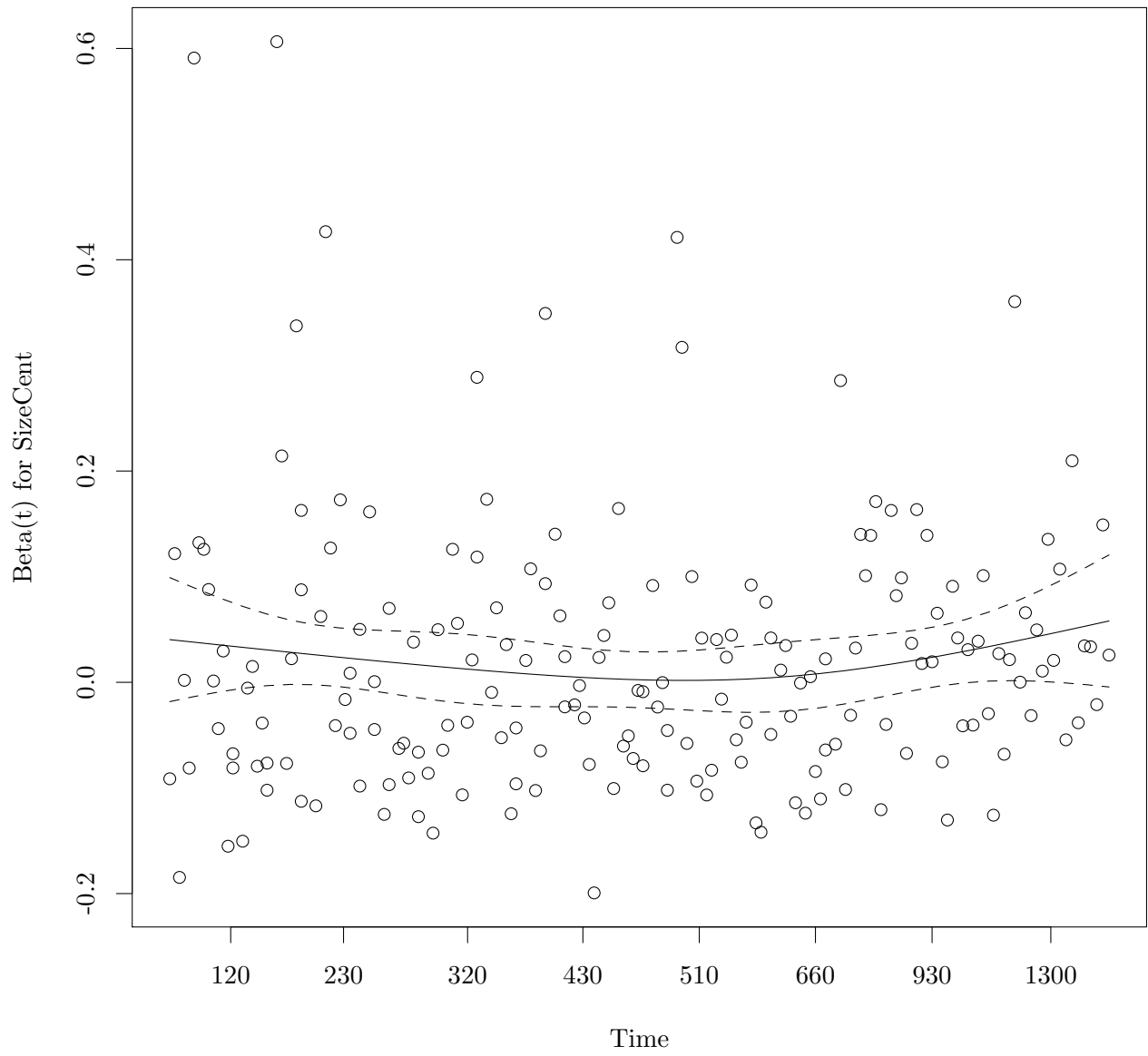
plot(cox.zph(fit.cph))
```

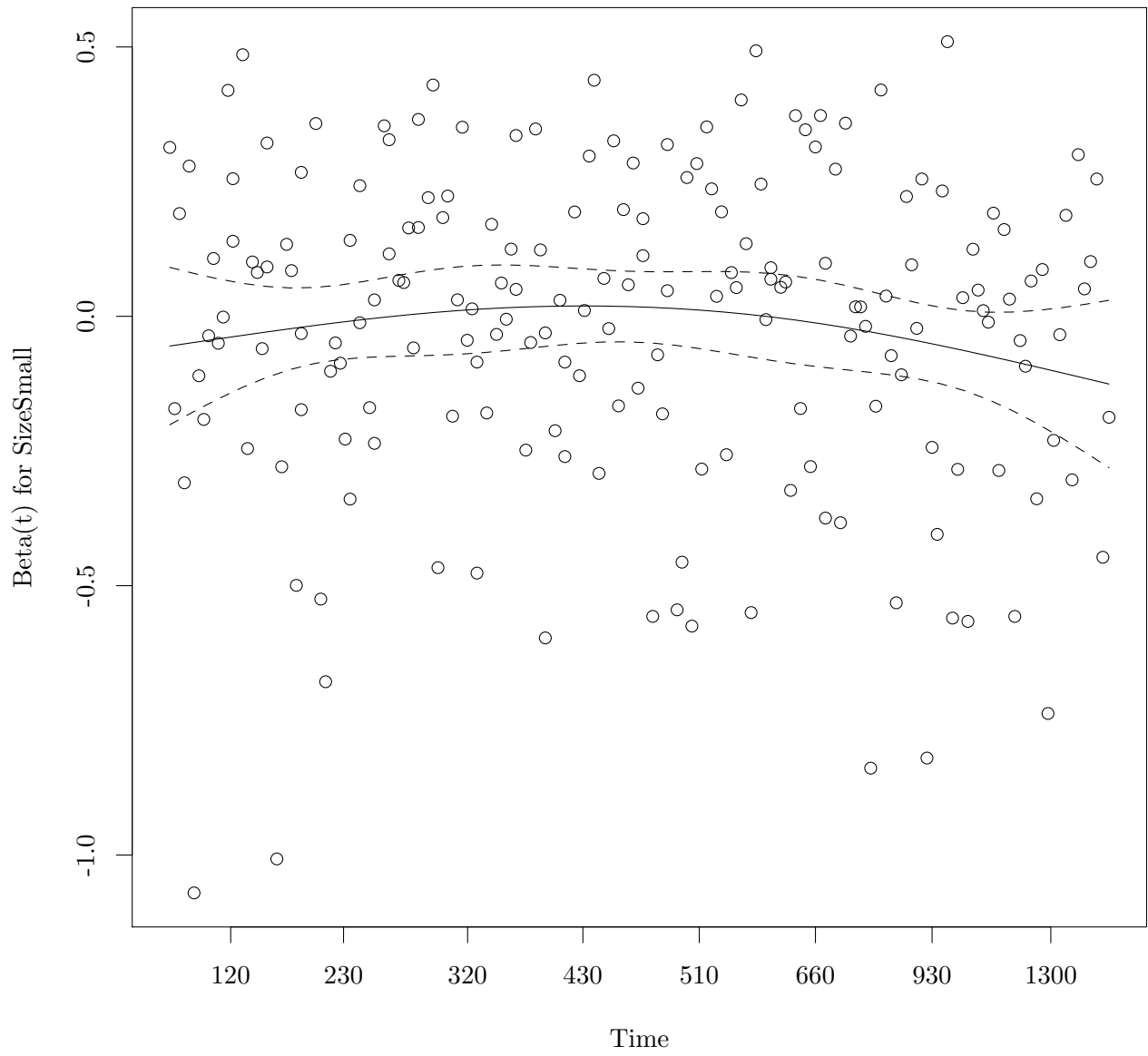


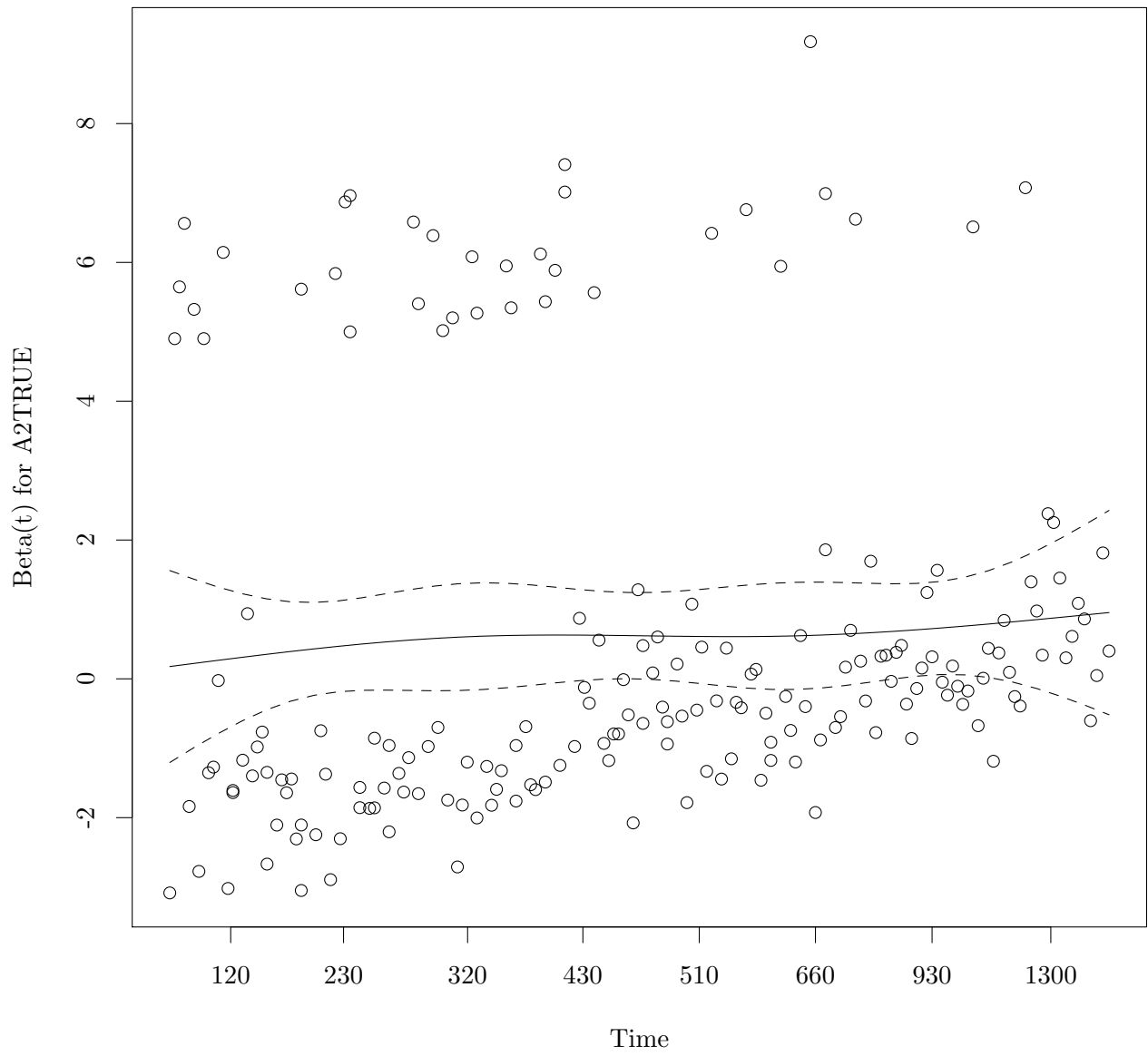


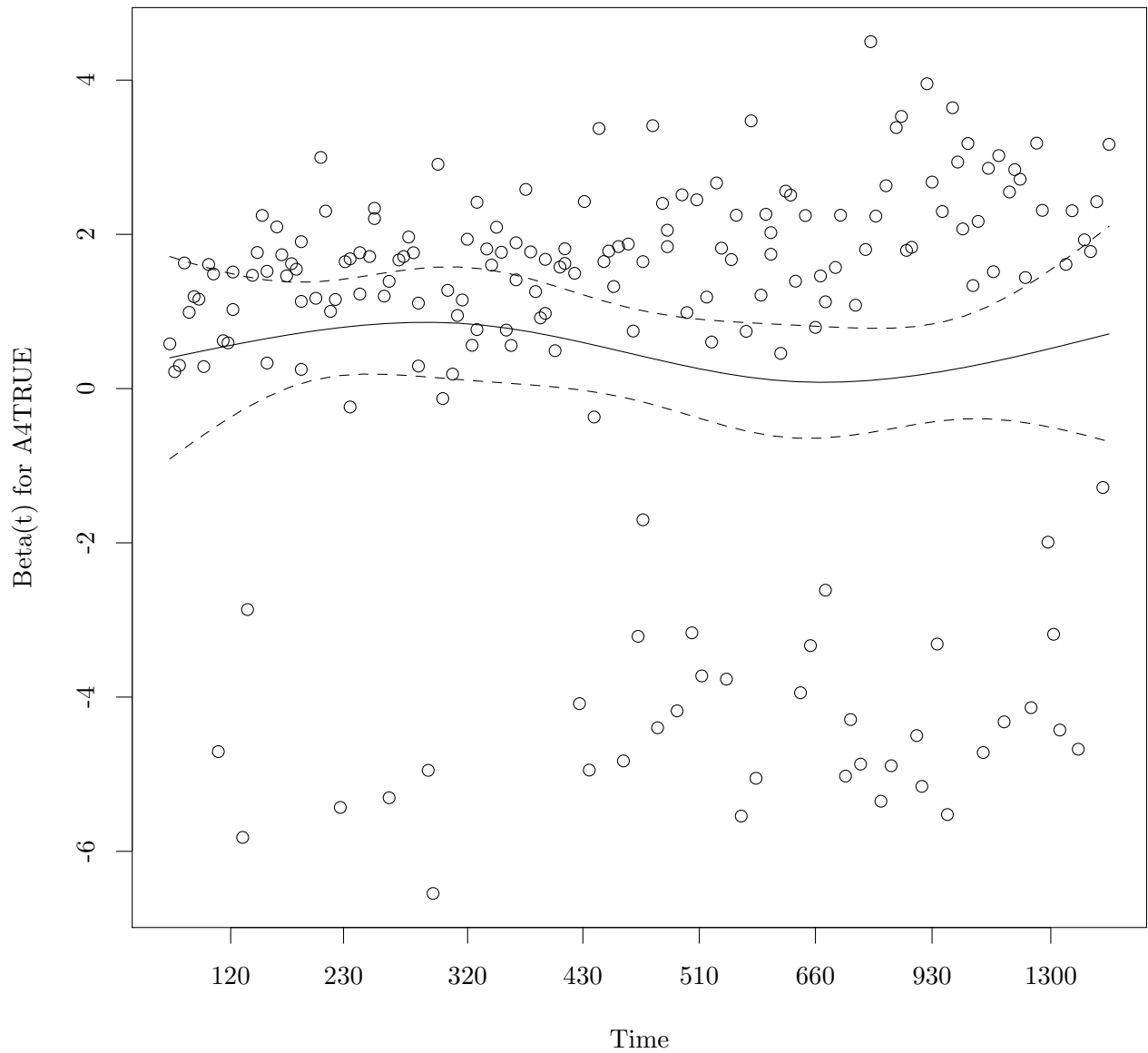












```
temp = function(x, resid = TRUE, se = TRUE, df = 4, nsmo = 40, var, ...) {
  xx <- x$x
  yy <- x$y
  d <- nrow(yy)
  df <- max(df)
  nvar <- ncol(yy)
  pred.x <- seq(from = min(xx), to = max(xx), length = nsmo)
  temp <- c(pred.x, xx)
  lmat <- ns(temp, df = df, intercept = TRUE)
  pmat <- lmat[1:nsmo, ]
  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) {

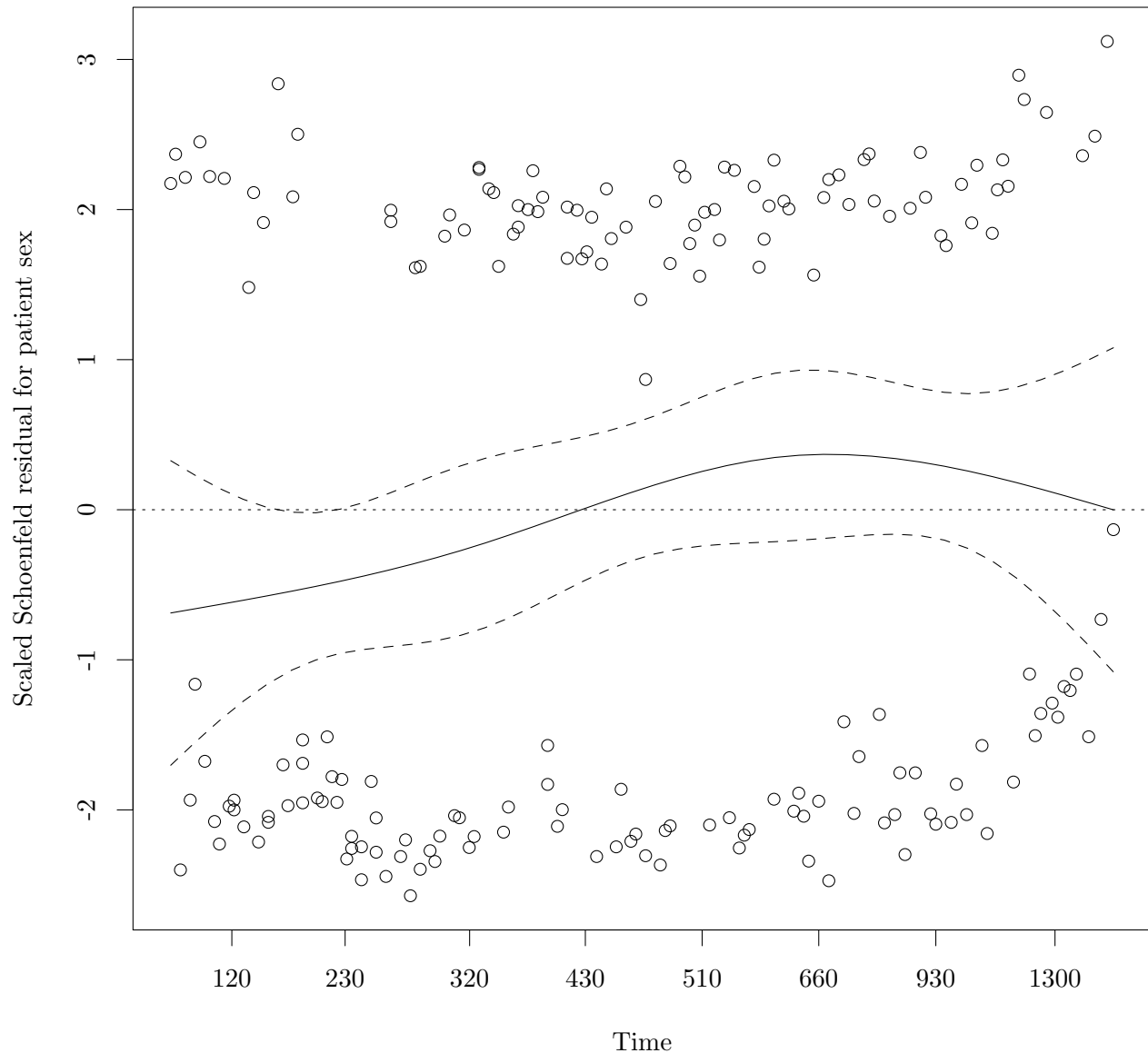
```

```

    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 = data)

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

```

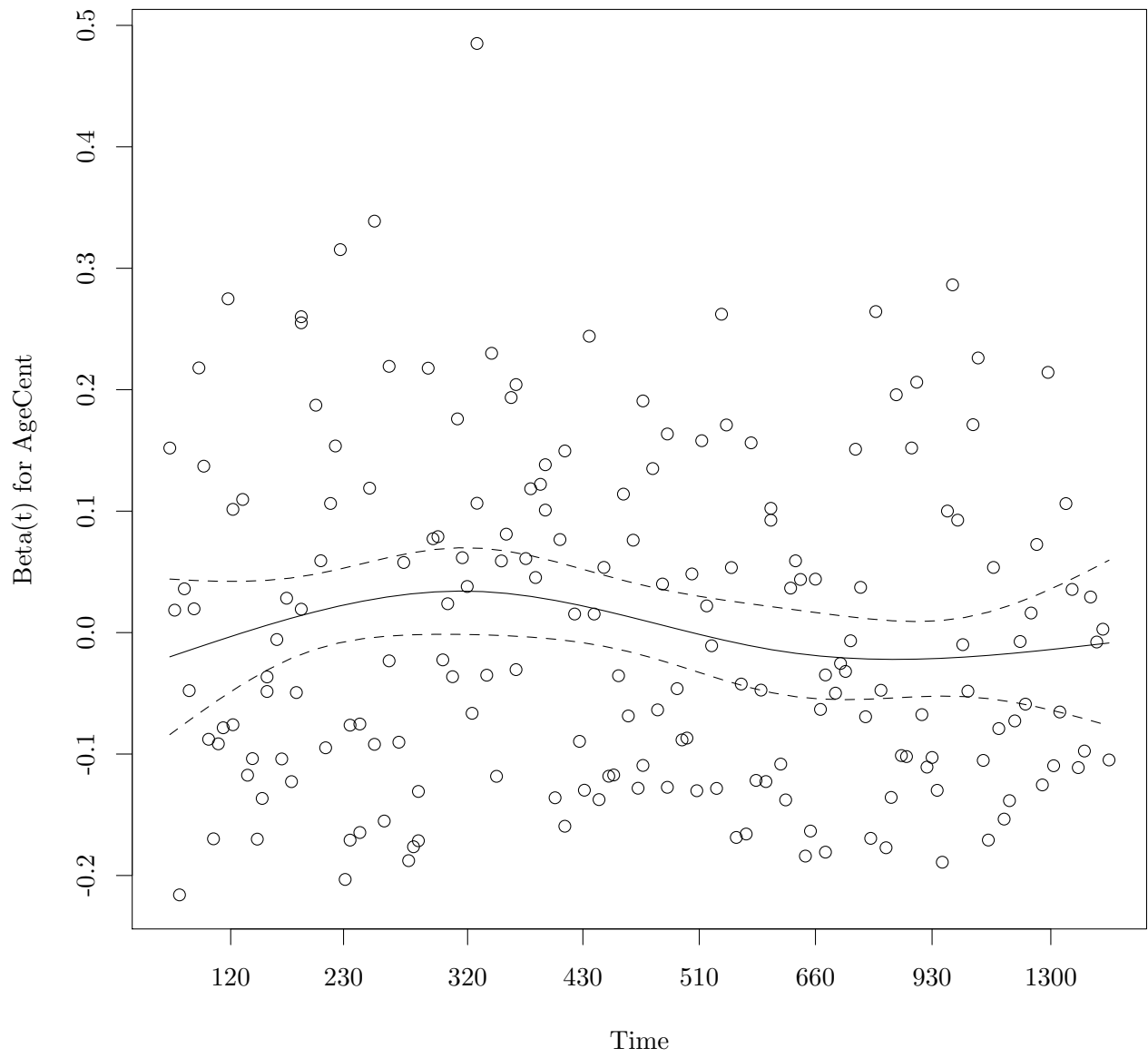
```
##          loglik Chisq Df Pr(>|Chi|)
## NULL          -816
## SexM          -816  0.31  1    0.5770
## AgeCent       -816  0.00  1    0.9623
## AgeCent2      -815  0.78  1    0.3773
## LocBody       -813  3.45  1    0.0634
## SizeCent      -809  7.86  1    0.0050
## SizeSmall     -809  0.00  1    0.9983
## A2            -805  9.64  1    0.0019
## A4            -801  6.85  1    0.0088
## SexM:AgeCent  -800  1.65  1    0.1993
## SexM:AgeCent2 -800  0.00  1    0.9808
## SexM:LocBody  -800  0.10  1    0.7568
## SexM:SizeCent -800  0.65  1    0.4218
## SexM:SizeSmall -800  0.01  1    0.9108
## SexM:A2       -800  0.00  1    0.9960
## SexM:A4       -800  0.03  1    0.8537
```

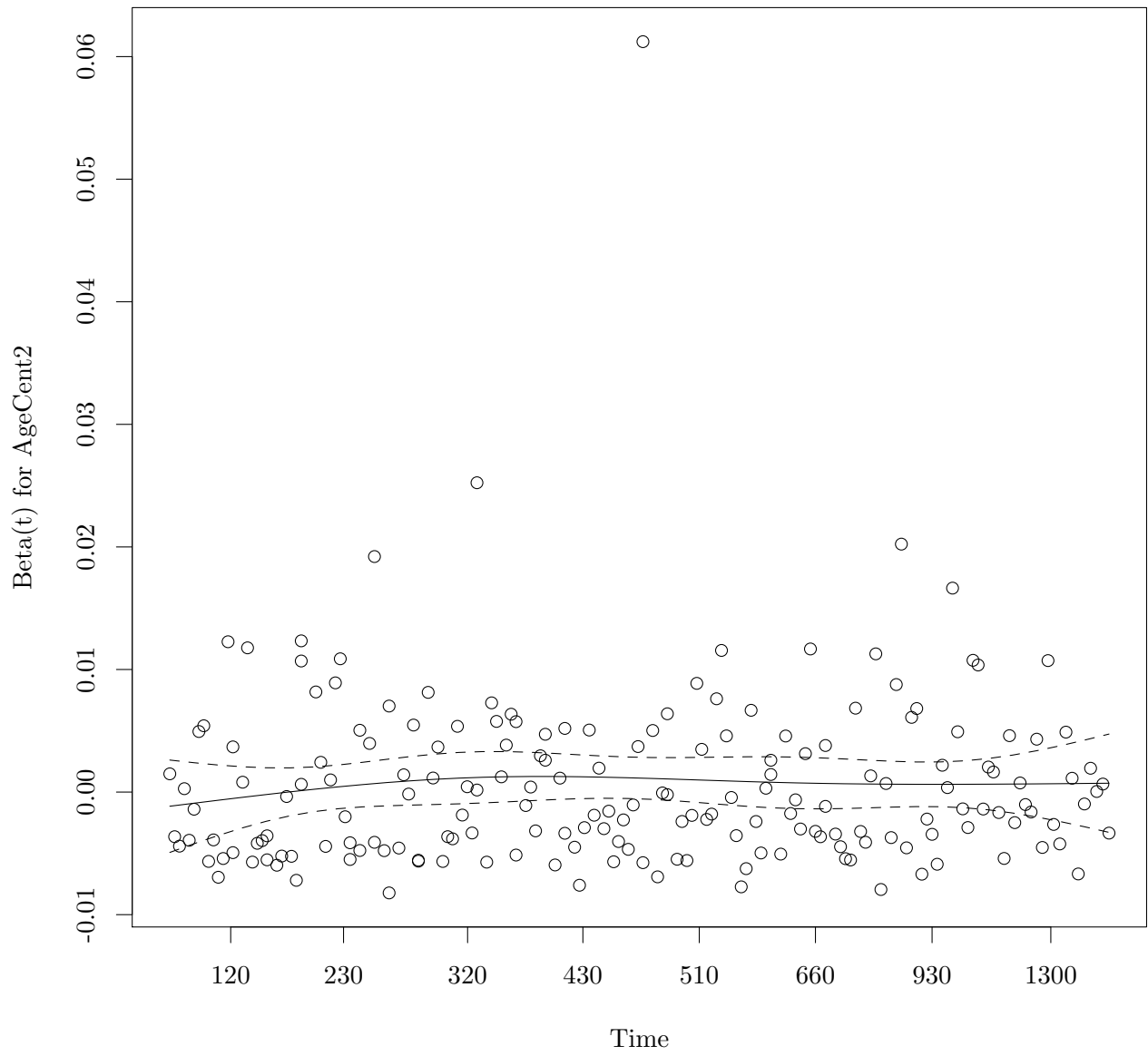
Nope, good. We're not interested in gender effects so just stratify.

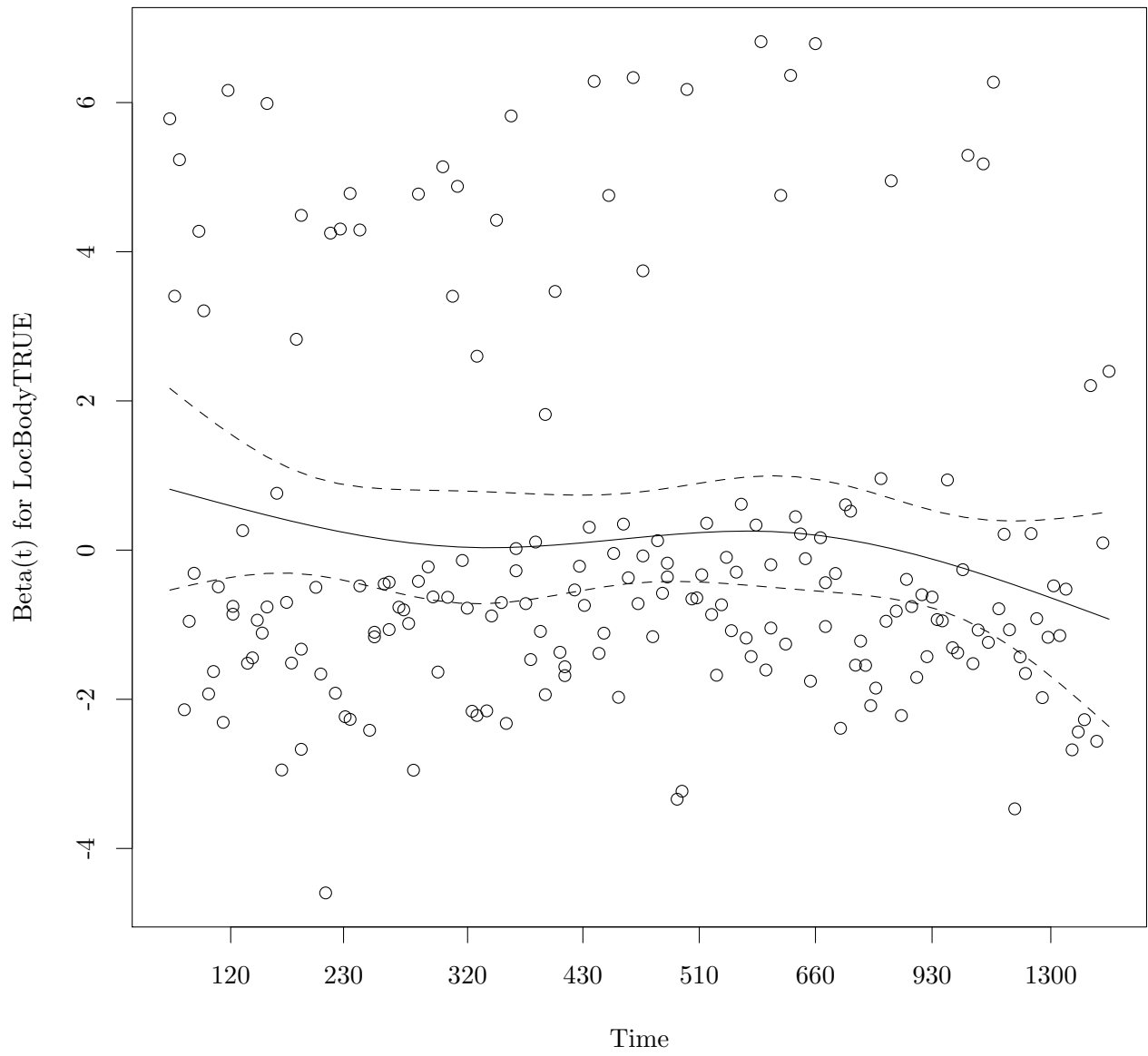
```
fit.cph = coxph(Surv(Time, DSD) ~ strata(SexM) + AgeCent + AgeCent2 + LocBody + SizeCent + SizeSmall + A2 + A4)
cox.zph(fit.cph)

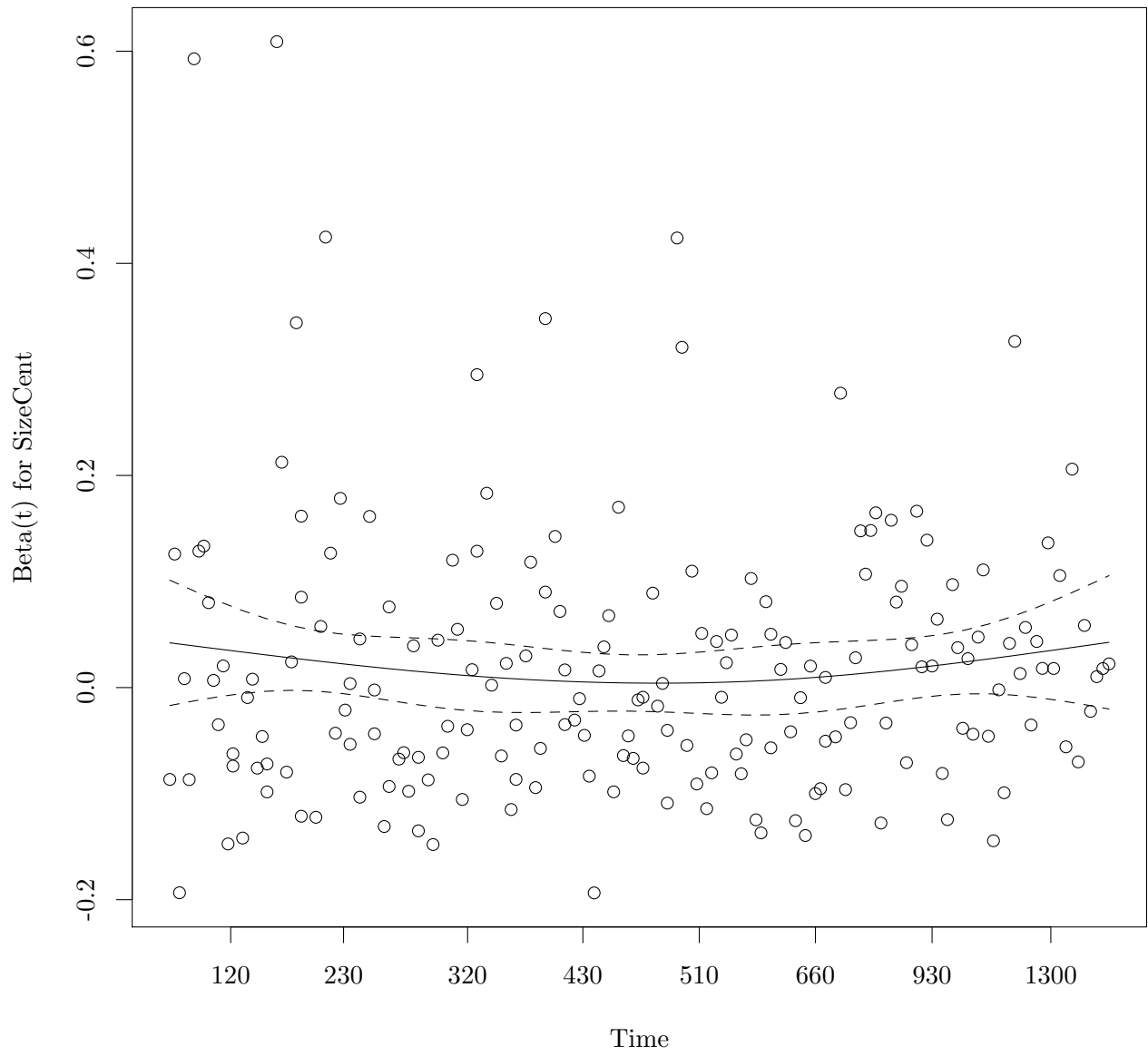
##          rho  chisq    p
## AgeCent    -0.09066 1.6632 0.197
## AgeCent2     0.03371 0.2006 0.654
## LocBodyTRUE -0.10840 1.8729 0.171
## SizeCent    -0.00856 0.0157 0.900
## SizeSmall   -0.04531 0.3927 0.531
## A2TRUE      0.05681 0.6145 0.433
## A4TRUE     -0.06539 0.7755 0.379
## GLOBAL      NA  8.3356 0.304

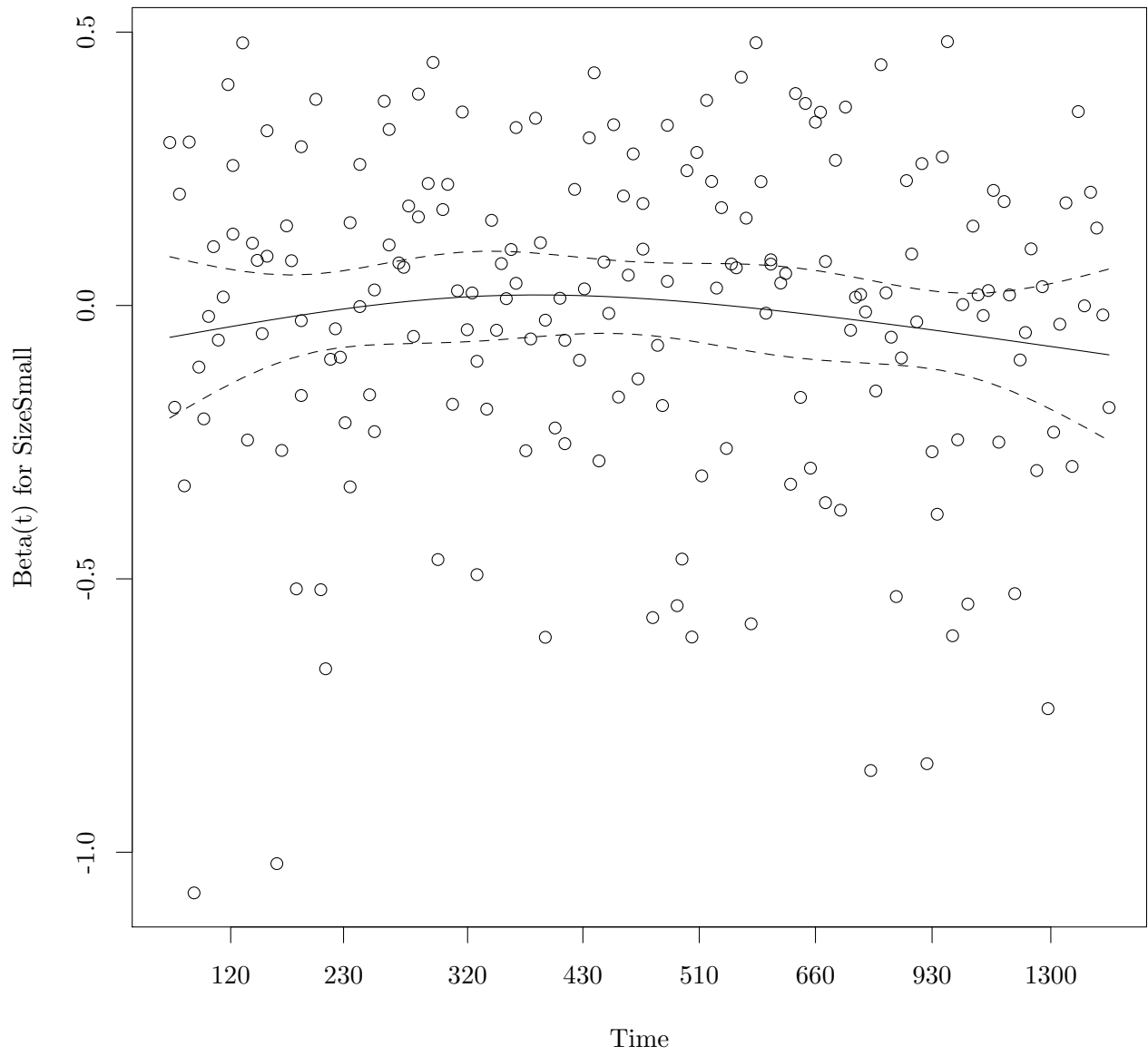
plot(cox.zph(fit.cph))
```

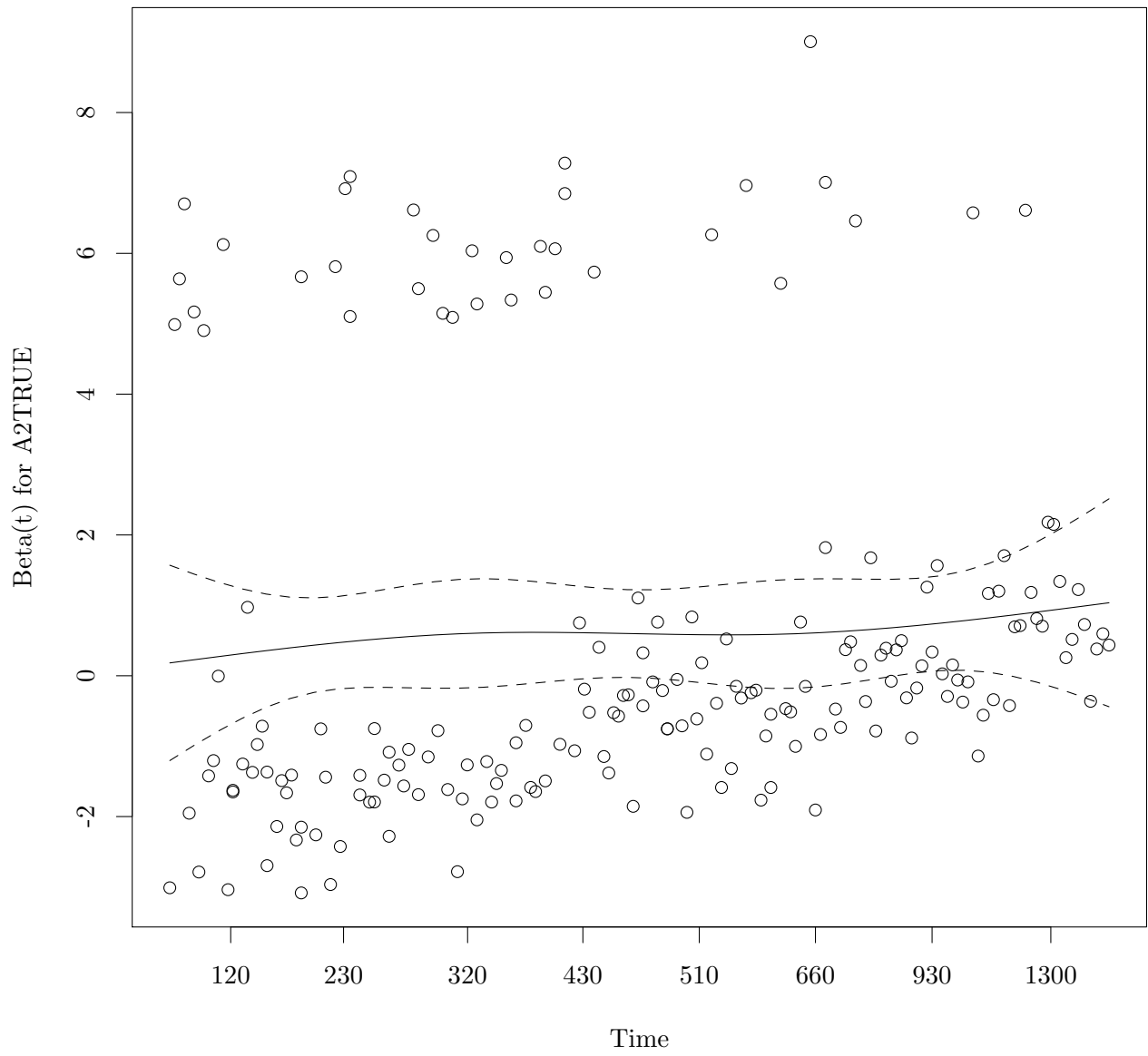


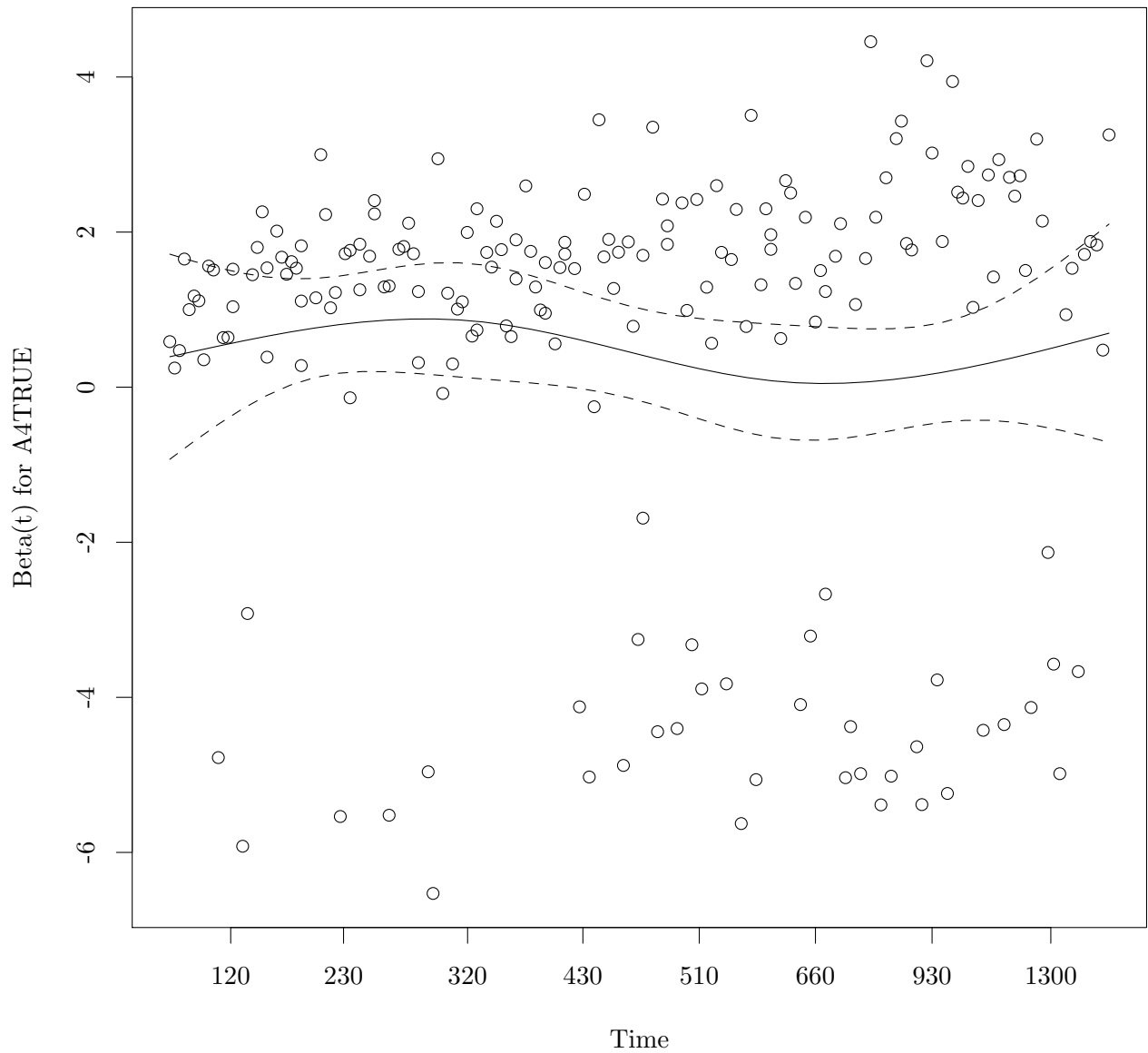






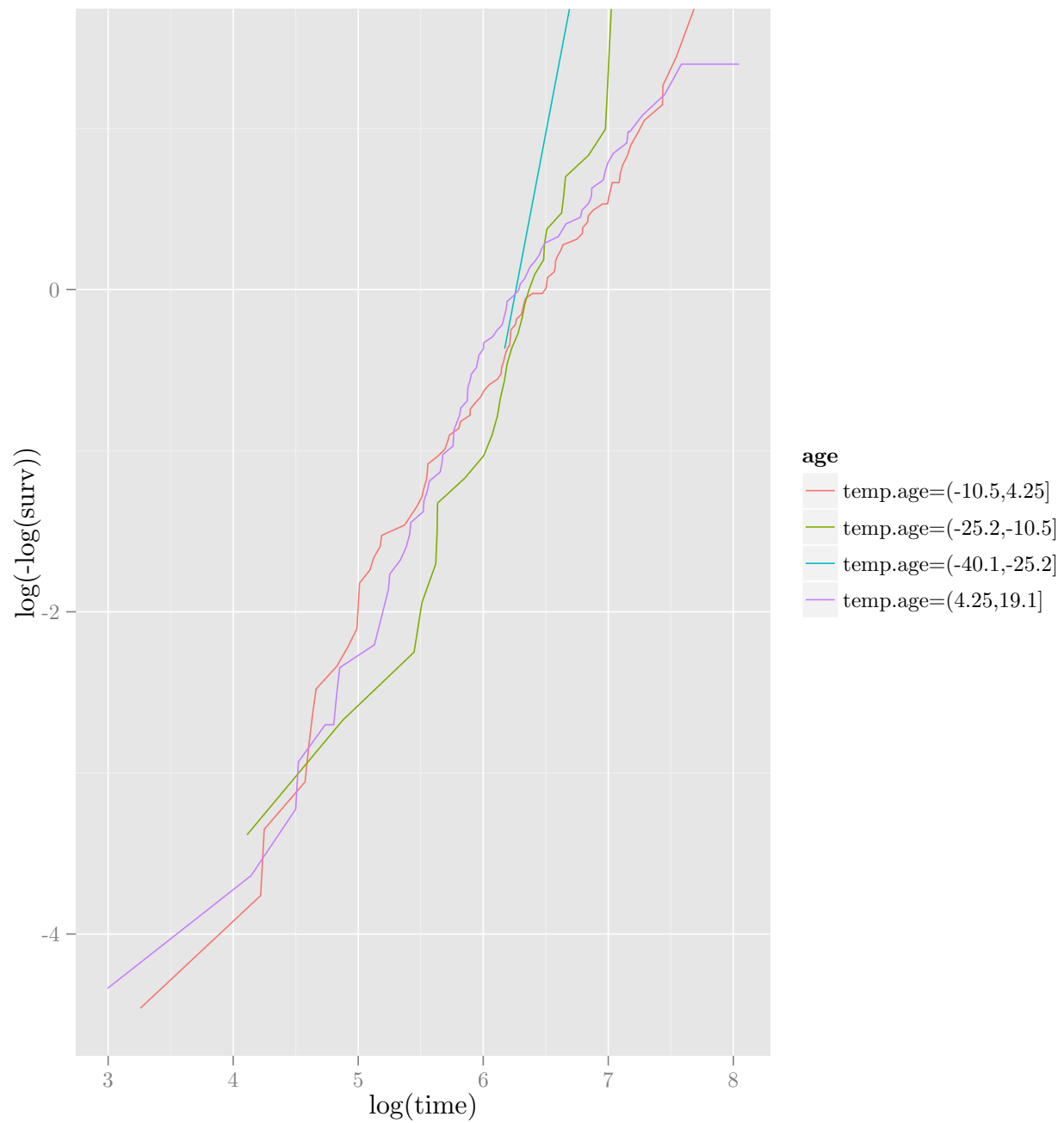






Looks good. Slight snifter with age but I'm not particularly concerned. Split into age groups and do KM plots to verify.

```
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
```

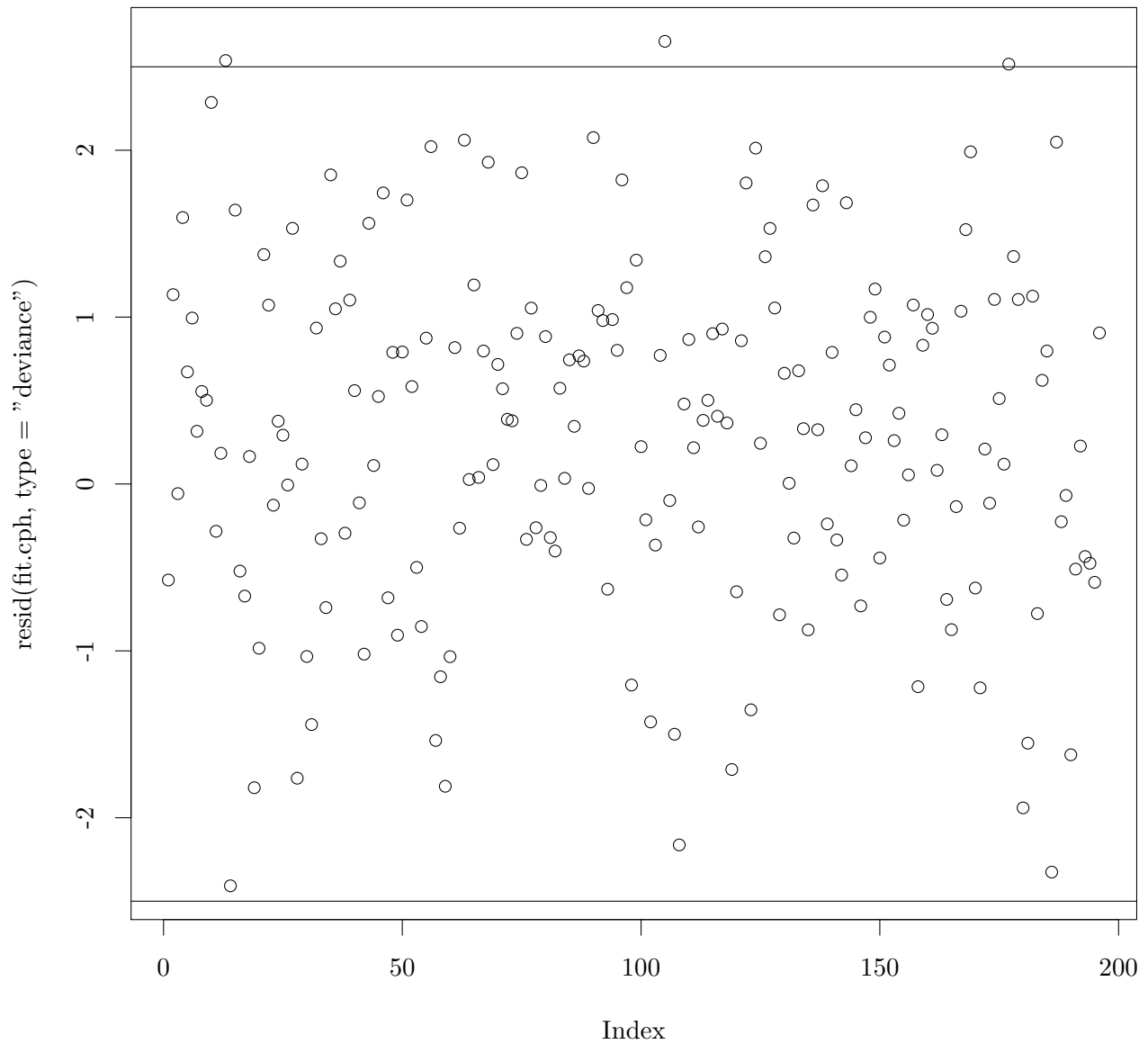


Not perfect but it'll do.

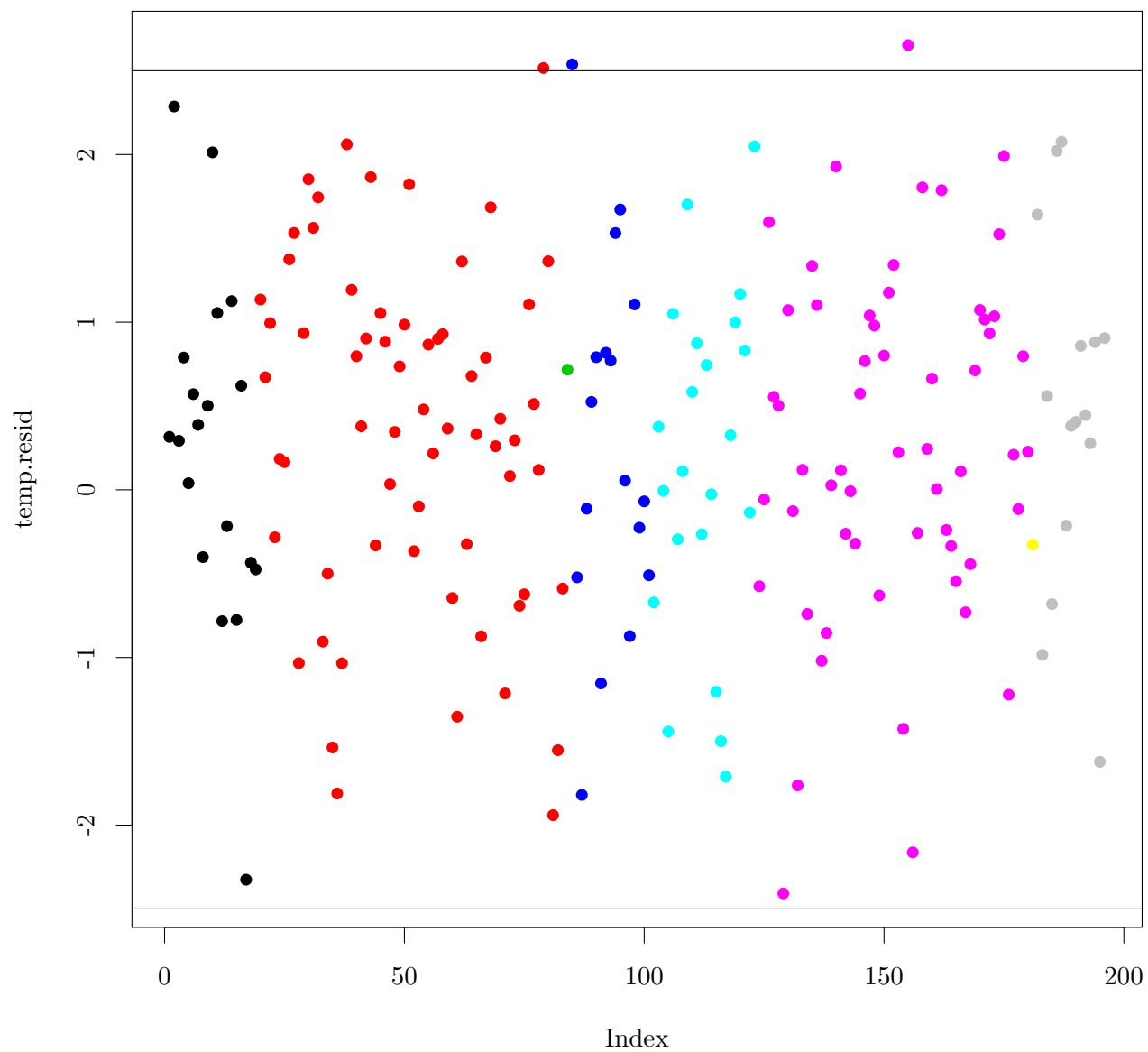
4.3 Outliers: full model

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

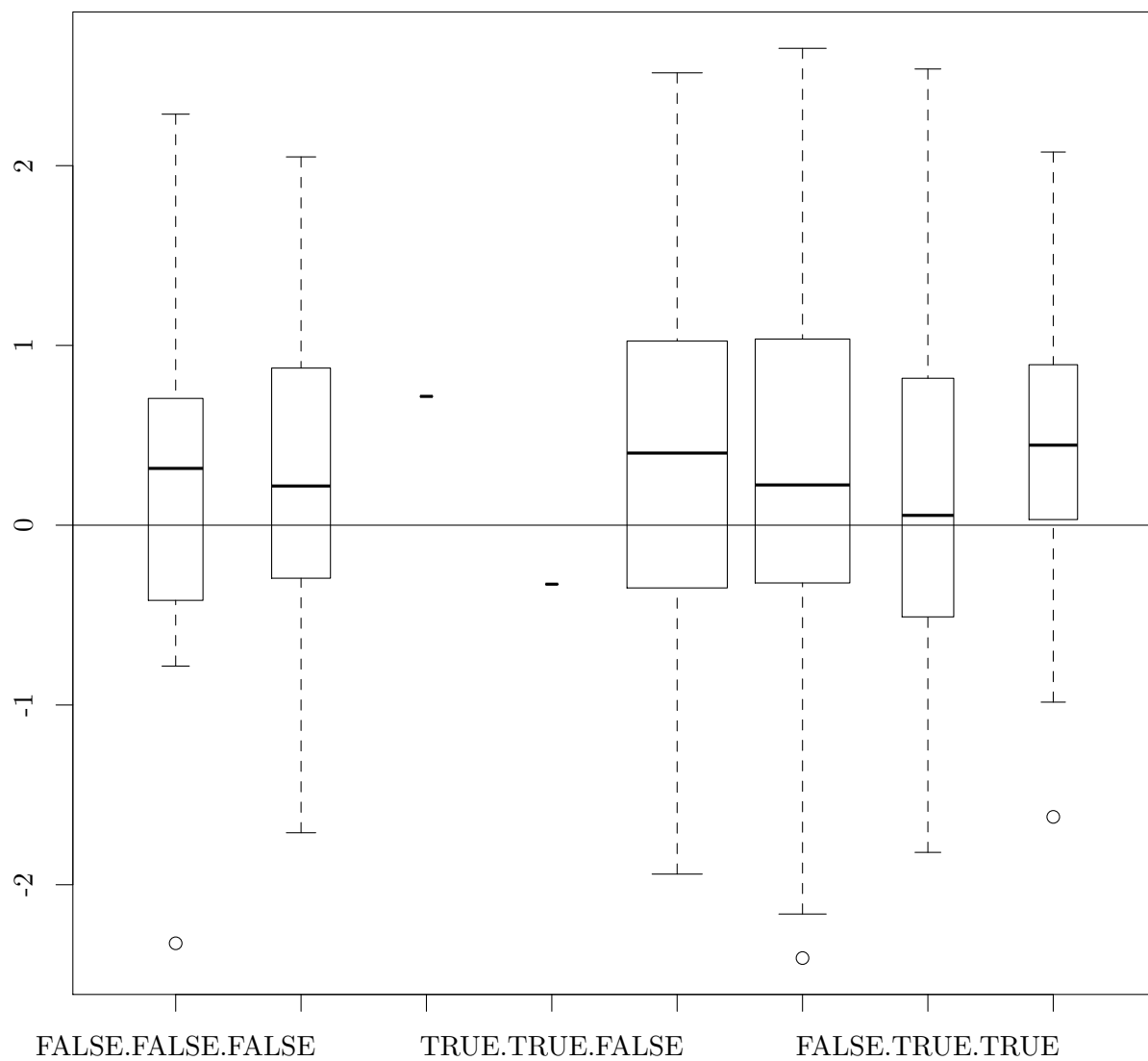
```
plot(resid(fit.cph, type = "deviance"))
abline(h = c(-2.5, 2.5))
```



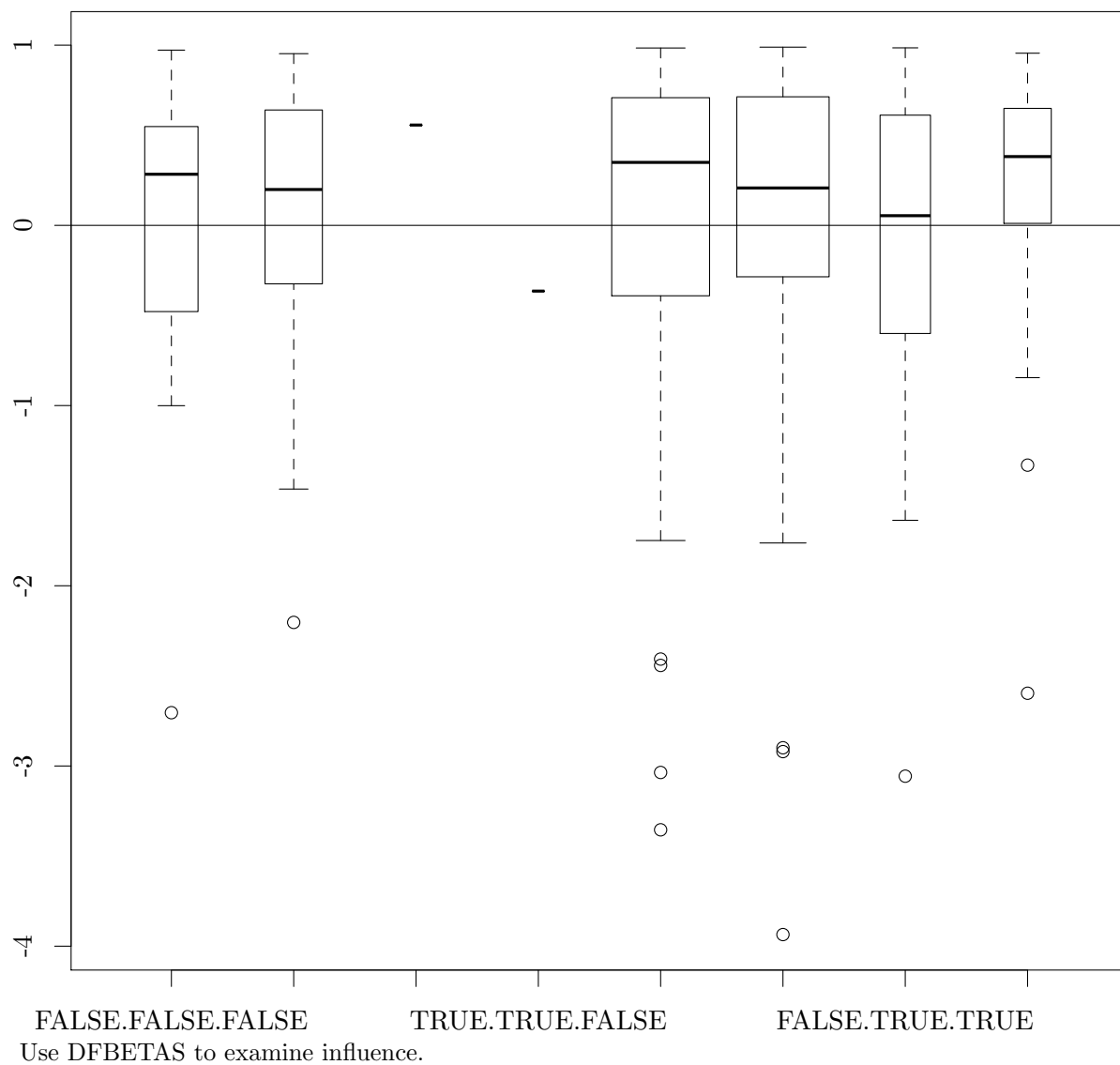
```
temp.ord = order(data$SexM, data$A2, data$A4)
temp.resid = resid(fit.cph, type = "deviance")[temp.ord]
temp.col = (4*data$SexM + 2*data$A2 + data$A4 + 1)[temp.ord]
plot(temp.resid, col = temp.col, pch = 16)
abline(h = c(-2.5, 2.5))
```

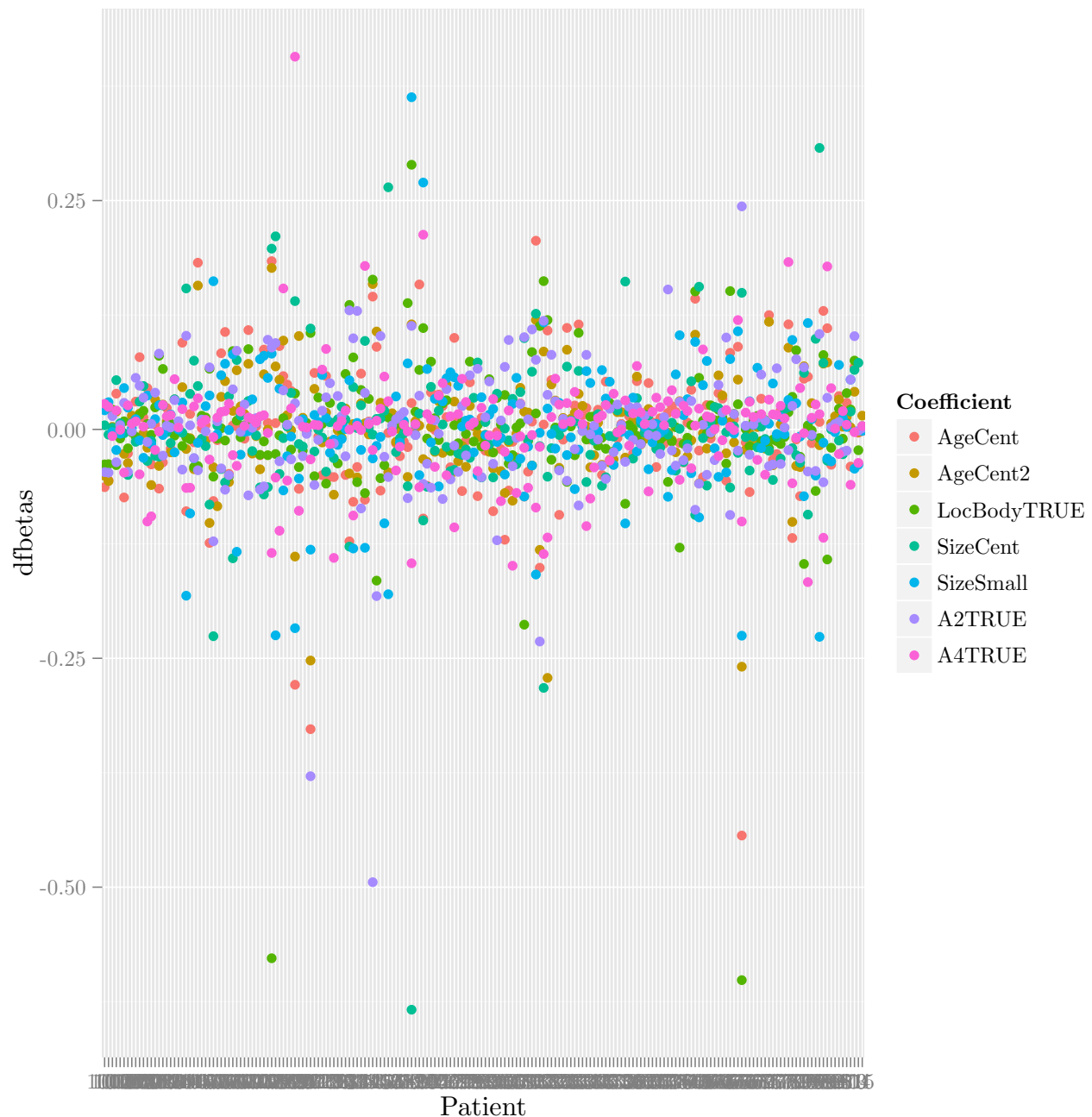
```
boxplot(resid(fit.cph, type = "deviance") ~ data$SexM + data$A2 + data$A4, varwidth = TRUE)
abline(h = 0)
```



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



```
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 + SizeSmall)
set.seed(20150110)
fit.cph.as = glmulti(Surv(Time, DSD) ~ strata(SexM) + AgeCent + AgeCent2 + LocBody + SizeCent + SizeSmall)
## 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 18 ; beta may be infinite.
## Warning in fitter(X, Y, strats, offset, init, control, weights = weights, : Loglik converged
before variable 18 ; beta may be infinite.
## Warning in fitter(X, Y, strats, offset, init, control, weights = weights, : Loglik converged
before variable 22 ; beta may be infinite.
## Warning in fitter(X, Y, strats, offset, init, control, weights = weights, : Loglik converged
before variable 18 ; beta may be infinite.
## Warning in fitter(X, Y, strats, offset, init, control, weights = weights, : Loglik converged
before variable 14 ; beta may be infinite.
## Warning in fitter(X, Y, strats, offset, init, control, weights = weights, : Loglik converged
before variable 13 ; beta may be infinite.
## Warning in fitter(X, Y, strats, offset, init, control, weights = weights, : Loglik converged
before variable 13 ; 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=1386
## Surv(Time, DSD) ~ strata(SexM) + AgeCent + AgeCent2 + LocBody +
## SizeCent + SizeSmall + A2 + A4
##
##           Df  AIC
## - AgeCent    1 1381
## - LocBody    1 1381
## - SizeSmall  1 1381
## - AgeCent2   1 1382
## - SizeCent   1 1385
## <none>       1386
## - A4         1 1387
## - A2         1 1388
##
## Step: AIC=1381
## Surv(Time, DSD) ~ strata(SexM) + AgeCent2 + LocBody + SizeCent +
## SizeSmall + A2 + A4
##
##           Df  AIC
```

```

## - LocBody      1 1376
## - SizeSmall    1 1376
## - AgeCent2     1 1377
## - SizeCent     1 1379
## <none>         1381
## - A4           1 1382
## - A2           1 1383
##
## Step: AIC=1376
## Surv(Time, DSD) ~ strata(SexM) + AgeCent2 + SizeCent + SizeSmall +
##      A2 + A4
##
##           Df  AIC
## - SizeSmall  1 1371
## - AgeCent2   1 1372
## - SizeCent   1 1375
## <none>       1376
## - A4         1 1377
## - A2         1 1379
##
## Step: AIC=1371
## Surv(Time, DSD) ~ strata(SexM) + AgeCent2 + SizeCent + A2 + A4
##
##           Df  AIC
## - AgeCent2  1 1367
## <none>       1371
## - SizeCent  1 1371
## - A4        1 1372
## - A2        1 1374
##
## Step: AIC=1367
## Surv(Time, DSD) ~ strata(SexM) + SizeCent + A2 + A4
##
##           Df  AIC
## <none>       1367
## - A4        1 1368
## - SizeCent  1 1368
## - A2        1 1370
## Call:
## coxph(formula = Surv(Time, DSD) ~ strata(SexM) + SizeCent + A2 +
##      A4, data = data)
##
##           coef exp(coef) se(coef)      z      p
## SizeCent 0.0139      1.01  0.00563  2.47 0.0140
## A2TRUE   0.5845      1.79  0.19894  2.94 0.0033
## A4TRUE   0.4311      1.54  0.18733  2.30 0.0210
##
## Likelihood ratio test=25.7 on 3 df, p=1.08e-05 n= 196, number of events= 188

```

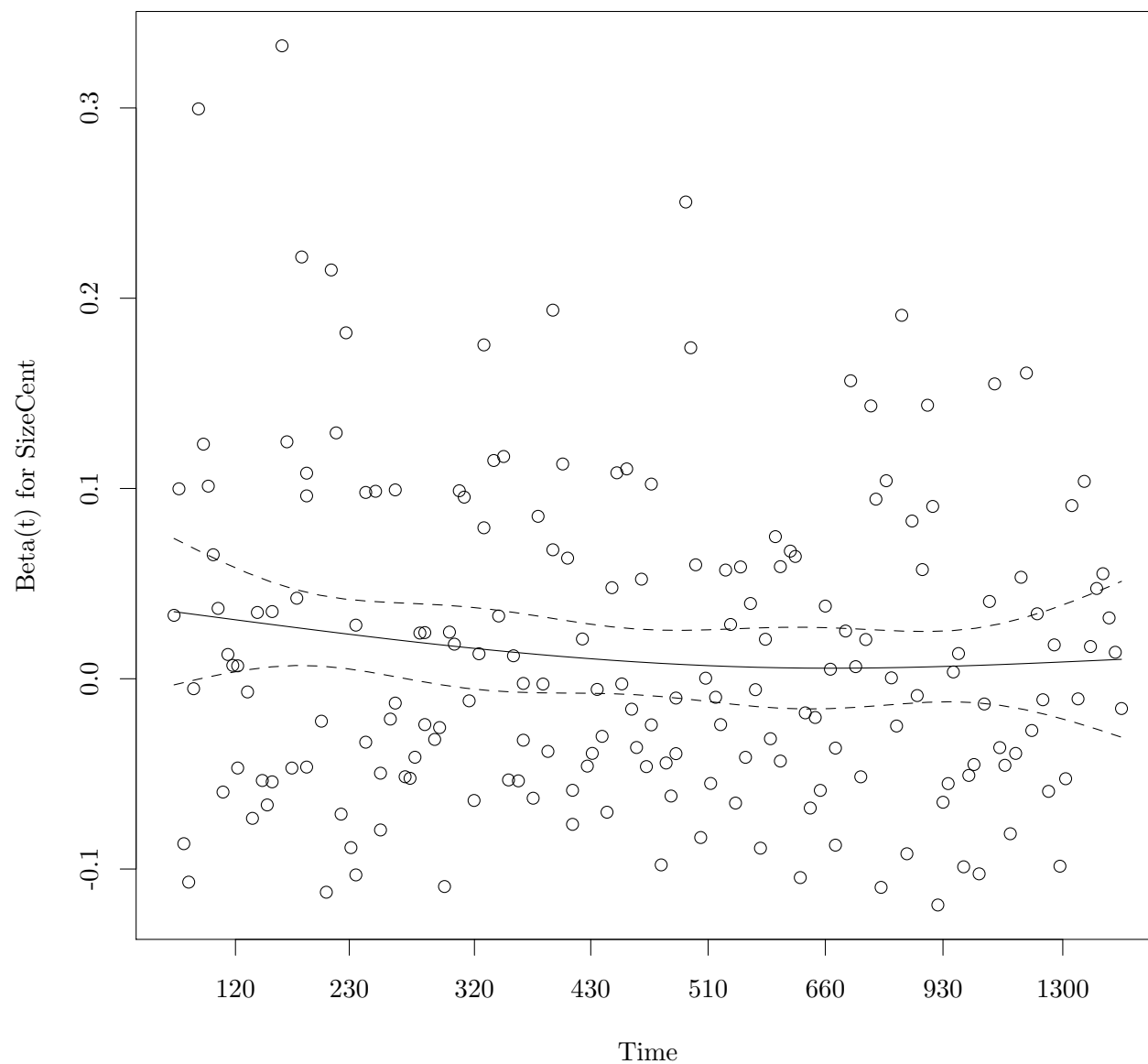
Consensus, excellent.

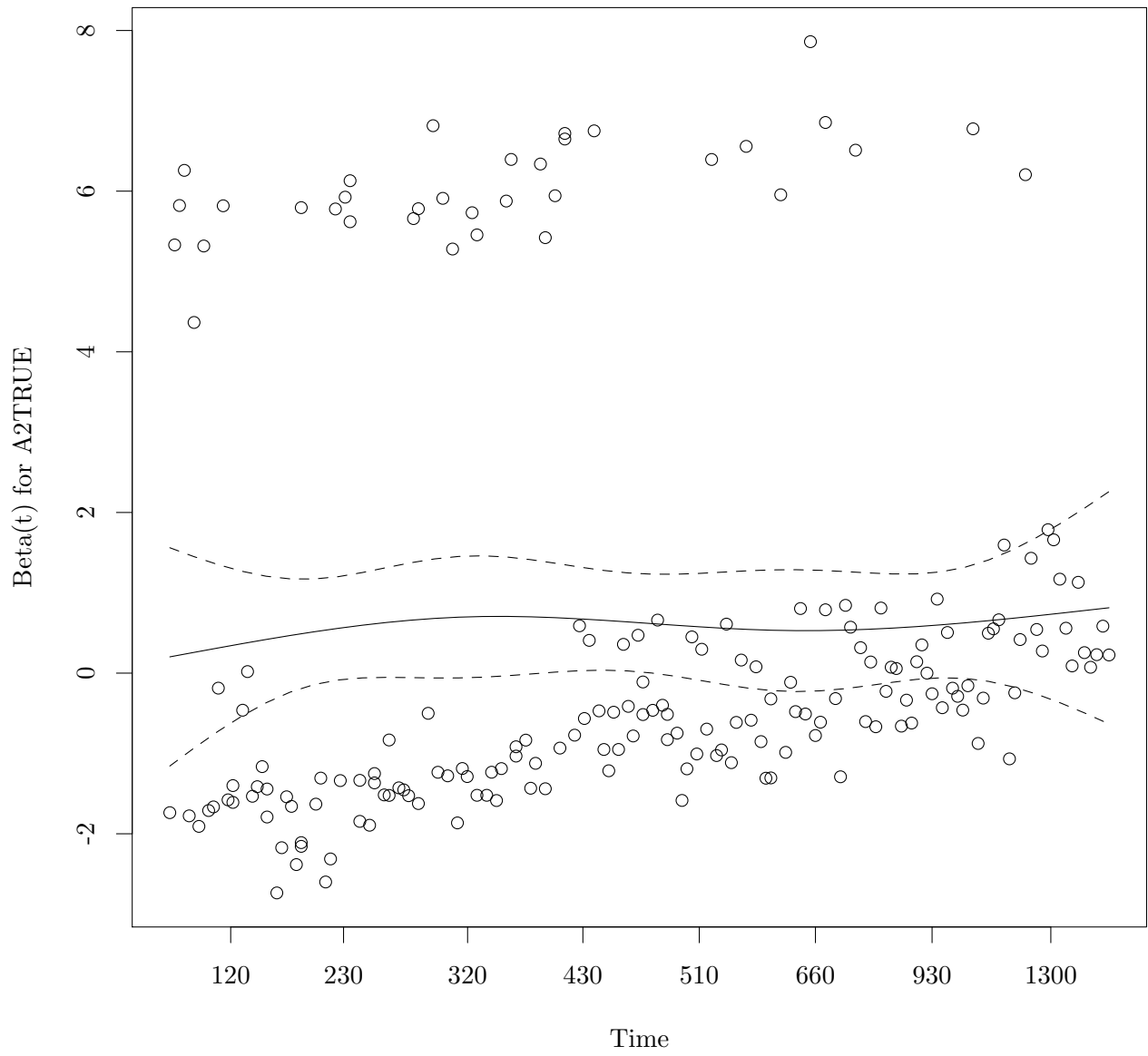
4.5 PH assumption: reduced model

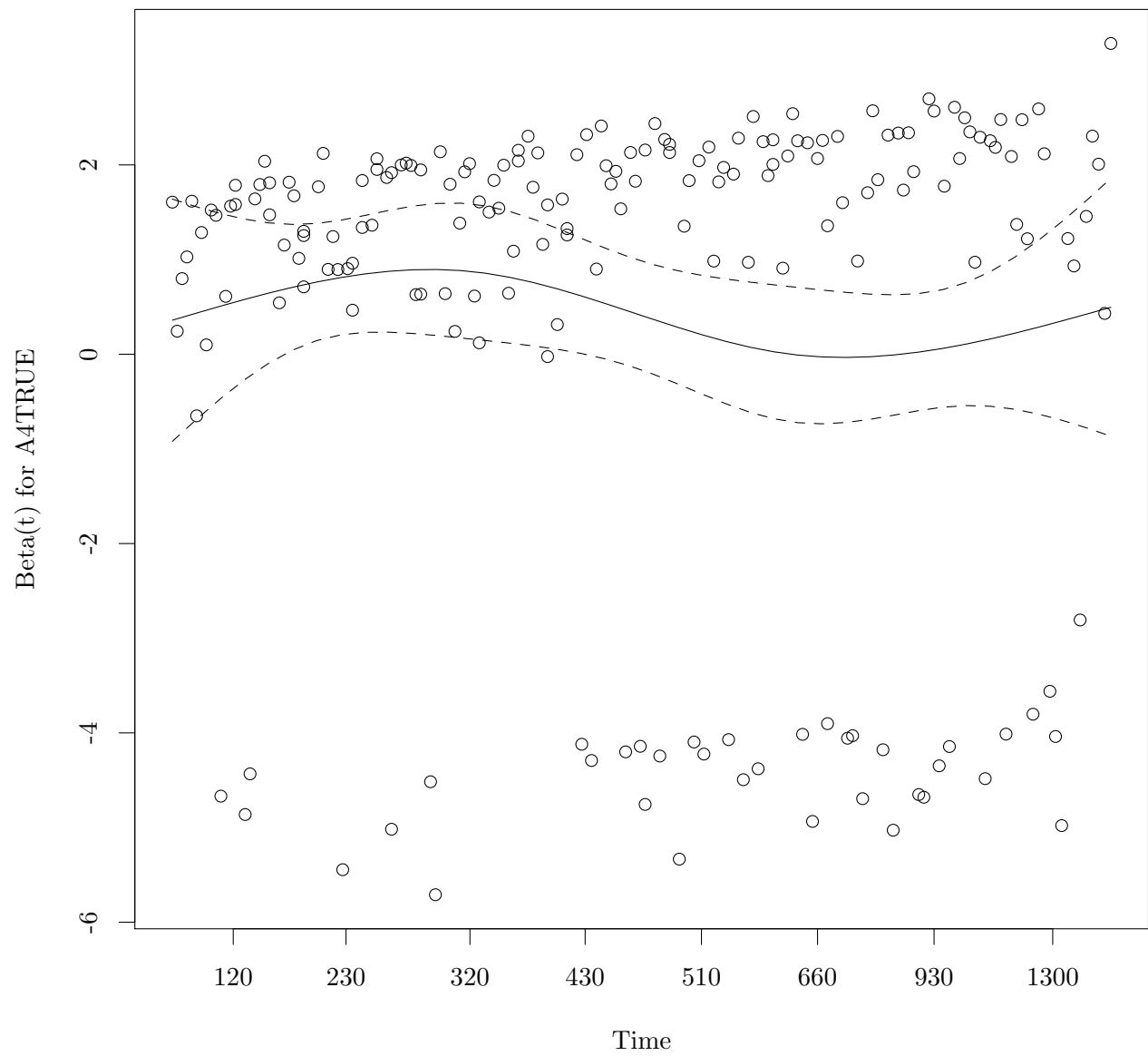
```
fit.cph = coxph(Surv(Time, DSD) ~ strata(SexM) + SizeCent + A2 + A4, data = data)
cox.zph(fit.cph)
```

```
##          rho chisq    p
## SizeCent -0.0937 1.894 0.169
## A2TRUE    0.0248 0.115 0.734
## A4TRUE    -0.0890 1.430 0.232
## GLOBAL      NA 3.747 0.290
```

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

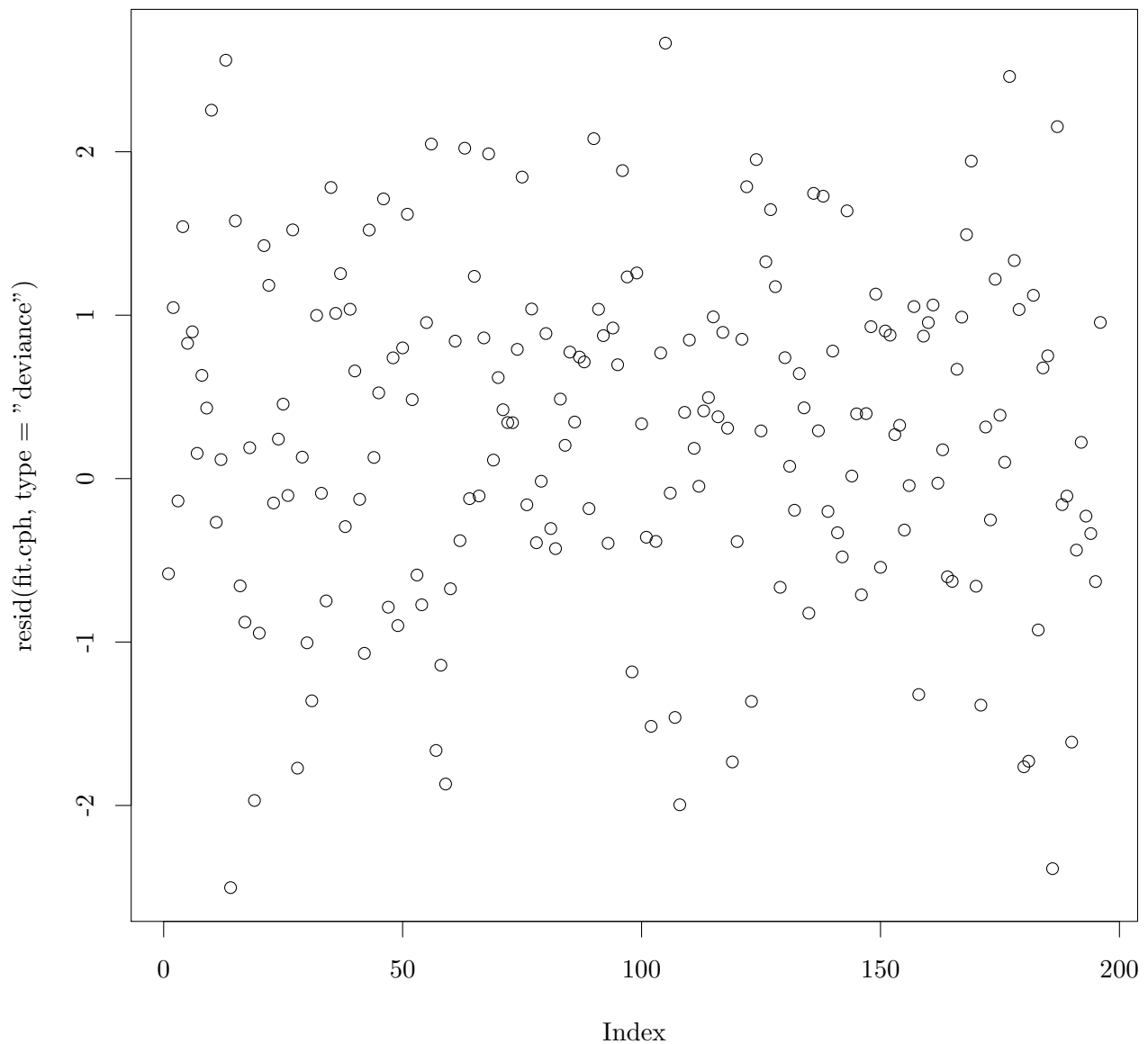






4.6 Outliers: reduced model

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

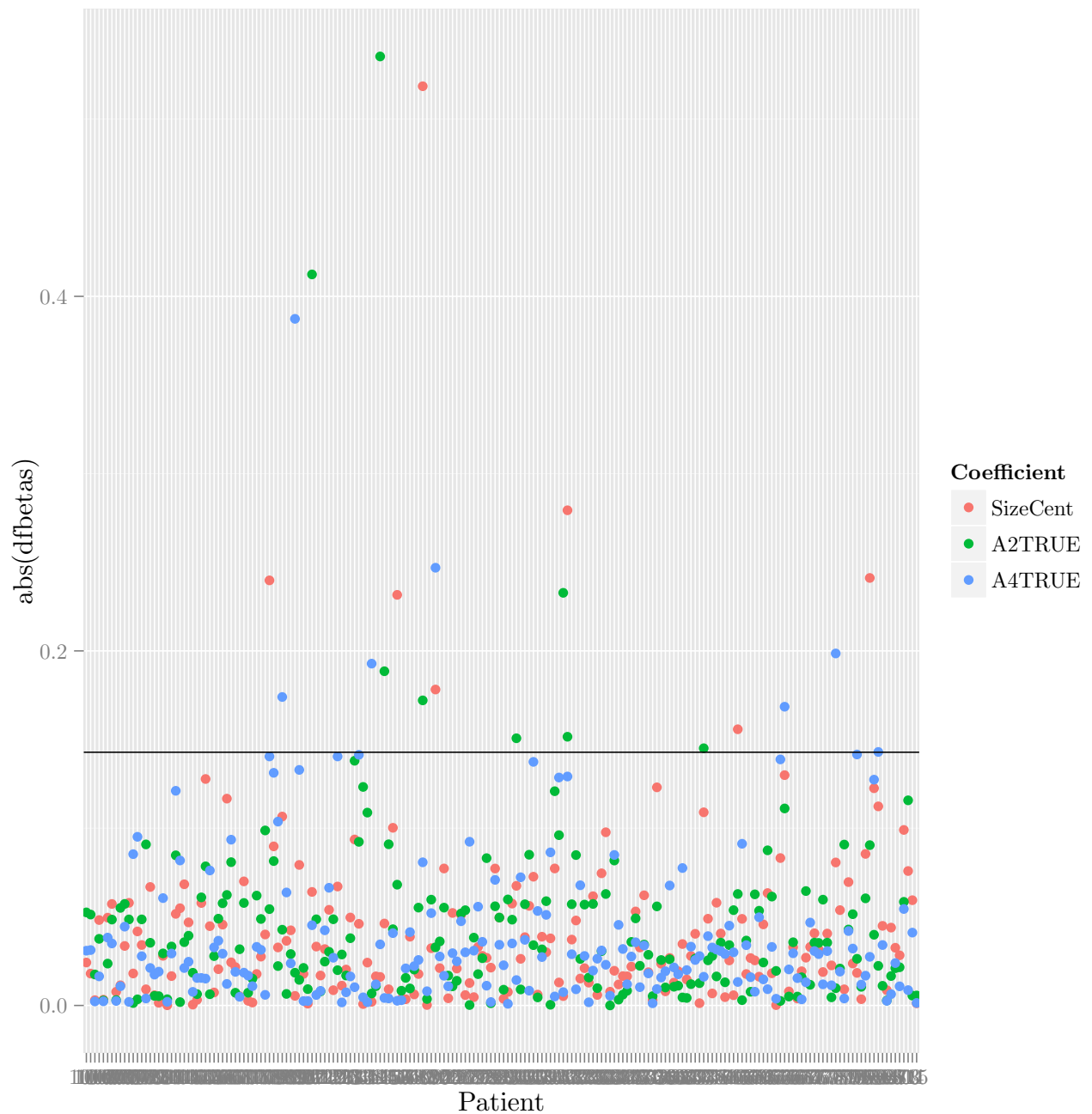


Now generate the restricted fit and examine the DFBETAS on the reduced model.

```
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] 0.1429

ggplot(temp, aes(y = abs(dfbetas), x = Patient, col = Coefficient)) + geom_point() + geom_hline(yintercept = 2/sqrt(nrow(data)))
```



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

##	NSWPCN_144	NSWPCN_183	NSWPCN_1212	NSWPCN_1195	NSWPCN_318	NSWPCN_195
##	0.535319	0.518522	0.412372	0.387309	0.279328	0.246925
##	NSWPCN_799	NSWPCN_1182	NSWPCN_317	NSWPCN_154	NSWPCN_777	NSWPCN_142
##	0.241159	0.239789	0.232758	0.231619	0.198560	0.192828
##	NSWPCN_145	NSWPCN_1188	NSWPCN_655	NSWPCN_606	NSWPCN_296	NSWPCN_374
##	0.188531	0.174002	0.168515	0.155810	0.150737	0.145092
##	NSWPCN_802	NSWPCN_795	NSWPCN_133	NSWPCN_125	NSWPCN_654	NSWPCN_131
##	0.143009	0.141584	0.141390	0.140472	0.138800	0.138044
##	NSWPCN_307	NSWPCN_1196	NSWPCN_1186	NSWPCN_316	NSWPCN_1155	NSWPCN_801
##	0.137409	0.132883	0.131227	0.128578	0.127761	0.127389
##	NSWPCN_135	NSWPCN_354	NSWPCN_1143	NSWPCN_315	NSWPCN_1167	NSWPCN_814

##	0.123283	0.123043	0.121087	0.120854	0.116675	0.115720
##	NSWPCN_138	NSWPCN_1187	NSWPCN_152	NSWPCN_813	NSWPCN_1179	NSWPCN_333
##	0.108794	0.103714	0.100222	0.099014	0.098728	0.097707
##	NSWPCN_1072	NSWPCN_1168	NSWPCN_269	NSWPCN_636	NSWPCN_1453	NSWPCN_1082
##	0.095141	0.093522	0.092355	0.091215	0.090934	0.090854
##	NSWPCN_789	NSWPCN_647	NSWPCN_312	NSWPCN_798	NSWPCN_1071	NSWPCN_305
##	0.090769	0.087516	0.086378	0.085496	0.085364	0.085038
##	NSWPCN_335	NSWPCN_322	NSWPCN_276	NSWPCN_1145	NSWPCN_364	NSWPCN_200
##	0.084907	0.084847	0.083106	0.081794	0.077536	0.077276
##	NSWPCN_281	NSWPCN_1157	NSWPCN_331	NSWPCN_303	NSWPCN_1172	NSWPCN_790
##	0.077244	0.076161	0.074559	0.072324	0.070007	0.069602
##	NSWPCN_1146	NSWPCN_323	NSWPCN_360	NSWPCN_1088	NSWPCN_1222	NSWPCN_664
##	0.068367	0.067802	0.067654	0.066832	0.066379	0.064549
##	NSWPCN_1189	NSWPCN_640	NSWPCN_351	NSWPCN_1177	NSWPCN_326	NSWPCN_651
##	0.063748	0.062695	0.062145	0.061975	0.061638	0.061411
##	NSWPCN_1153	NSWPCN_1139	NSWPCN_284	NSWPCN_194	NSWPCN_769	NSWPCN_815
##	0.060979	0.060563	0.059802	0.059695	0.059694	0.059379
##	NSWPCN_310	NSWPCN_377	NSWPCN_1031	NSWPCN_1165	NSWPCN_294	NSWPCN_1029
##	0.058826	0.057956	0.057888	0.057640	0.057463	0.057289
##	NSWPCN_1023	NSWPCN_320	NSWPCN_304	NSWPCN_324	NSWPCN_272	NSWPCN_182
##	0.057254	0.057106	0.057097	0.056915	0.055722	0.055199
##	NSWPCN_1028	NSWPCN_781	NSWPCN_445	NSWPCN_268	NSWPCN_643	NSWPCN_308
##	0.055045	0.053779	0.053775	0.053737	0.053434	0.053314
##	NSWPCN_347	NSWPCN_10	NSWPCN_24	NSWPCN_257	NSWPCN_794	NSWPCN_1016
##	0.053023	0.052498	0.052189	0.051883	0.051498	0.051284
##	NSWPCN_13	NSWPCN_282	NSWPCN_1022	NSWPCN_1160	NSWPCN_375	NSWPCN_1178
##	0.049613	0.049572	0.049531	0.048958	0.048914	0.048792
##	NSWPCN_1227	NSWPCN_1213	NSWPCN_1075	NSWPCN_1019	NSWPCN_1147	NSWPCN_665
##	0.048631	0.048614	0.048561	0.048203	0.046858	0.046764
##	NSWPCN_646	NSWPCN_336	NSWPCN_4	NSWPCN_804	NSWPCN_807	NSWPCN_1219
##	0.045742	0.045469	0.045022	0.044914	0.043924	0.042407
##	NSWPCN_1190	NSWPCN_164	NSWPCN_666	NSWPCN_381	NSWPCN_770	NSWPCN_370
##	0.042400	0.041379	0.040729	0.040591	0.040581	0.040421
##	NSWPCN_309	NSWPCN_270	NSWPCN_637	NSWPCN_20	NSWPCN_273	NSWPCN_346
##	0.038785	0.038276	0.036619	0.036126	0.035909	0.035845
##	NSWPCN_657	NSWPCN_7	NSWPCN_1141	NSWPCN_369	NSWPCN_1158	NSWPCN_350
##	0.035660	0.035272	0.033258	0.033176	0.032752	0.032716
##	NSWPCN_376	NSWPCN_810	NSWPCN_341	NSWPCN_1171	NSWPCN_384	NSWPCN_126
##	0.032695	0.032560	0.031861	0.031740	0.029033	0.028767
##	NSWPCN_352	NSWPCN_811	NSWPCN_373	NSWPCN_638	NSWPCN_330	NSWPCN_358
##	0.028581	0.028304	0.027879	0.026735	0.026383	0.025635
##	NSWPCN_256	NSWPCN_283	NSWPCN_775	NSWPCN_166	NSWPCN_1170	NSWPCN_362
##	0.024904	0.022669	0.022611	0.022181	0.021654	0.021446
##	NSWPCN_280	NSWPCN_1207	NSWPCN_161	NSWPCN_656	NSWPCN_128	NSWPCN_366
##	0.021341	0.021195	0.020940	0.020342	0.020270	0.019967
##	NSWPCN_653	NSWPCN_363	NSWPCN_36	NSWPCN_662	NSWPCN_1136	NSWPCN_1150
##	0.019505	0.019311	0.019303	0.019302	0.019072	0.018446
##	NSWPCN_1018	NSWPCN_1091	NSWPCN_1215	NSWPCN_1175	NSWPCN_21	NSWPCN_345
##	0.017496	0.017407	0.016899	0.016870	0.016860	0.016454
##	NSWPCN_143	NSWPCN_325	NSWPCN_1152	NSWPCN_658	NSWPCN_1176	NSWPCN_797
##	0.016435	0.015767	0.015516	0.015427	0.015148	0.011796
##	NSWPCN_1211	NSWPCN_806	NSWPCN_157	NSWPCN_190	NSWPCN_334	NSWPCN_1027
##	0.009221	0.008634	0.008317	0.008019	0.007766	0.007733
##	NSWPCN_9	NSWPCN_353	NSWPCN_1140	NSWPCN_1020		

```
##      0.005630      0.005273      0.003534      0.003253

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

## [1] 19
```

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 $\text{Surv}(\text{Time}, \text{DSD}) \sim 1 + \text{strata}(\text{SexM}) + \text{SizeCent} + \text{A2} + \text{A4}$. 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.rsfc = 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] -1355
```

```
fit.gf$loglik
```

```
## [1] -1354
```

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

```
## [1] 0.3371
AIC(fit.gg)
## [1] 2727
AIC(fit.gf)
## [1] 2729
BIC(fit.gg)
## [1] 2757
BIC(fit.gf)
## [1] 2765

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    6.41920    6.08210    6.75630    0.17199
## sigma            NA    0.79193    0.68758    0.91211    0.05709
## Q                NA    0.06489   -0.48729    0.61708    0.28173
## SexMTRUE         0.48469    0.36244    0.03548    0.68940    0.16682
## SizeCent         3.82143   -0.01028   -0.01751   -0.00305    0.00369
## A2TRUE           0.17347   -0.37995   -0.64171   -0.11818    0.13356
## A4TRUE           0.78061   -0.32171   -0.57137   -0.07206    0.12738
## sigma(SexMTRUE)  0.48469   -0.25903   -0.48789   -0.03017    0.11677
## Q(SexMTRUE)      0.48469    0.76270    0.04909    1.47631    0.36409
##
##      exp(est)  L95%      U95%
## mu              NA         NA         NA
## sigma            NA         NA         NA
## Q                NA         NA         NA
## SexMTRUE         1.43683    1.03612    1.99251
## SizeCent         0.98977    0.98264    0.99695
## A2TRUE           0.68390    0.52639    0.88854
## A4TRUE           0.72491    0.56475    0.93048
## sigma(SexMTRUE)  0.77180    0.61392    0.97028
## Q(SexMTRUE)      2.14406    1.05032    4.37675
##
## N = 196,  Events: 188,  Censored: 8
## Total time at risk: 113142
## Log-likelihood = -1355, df = 9
## AIC = 2727
```

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)

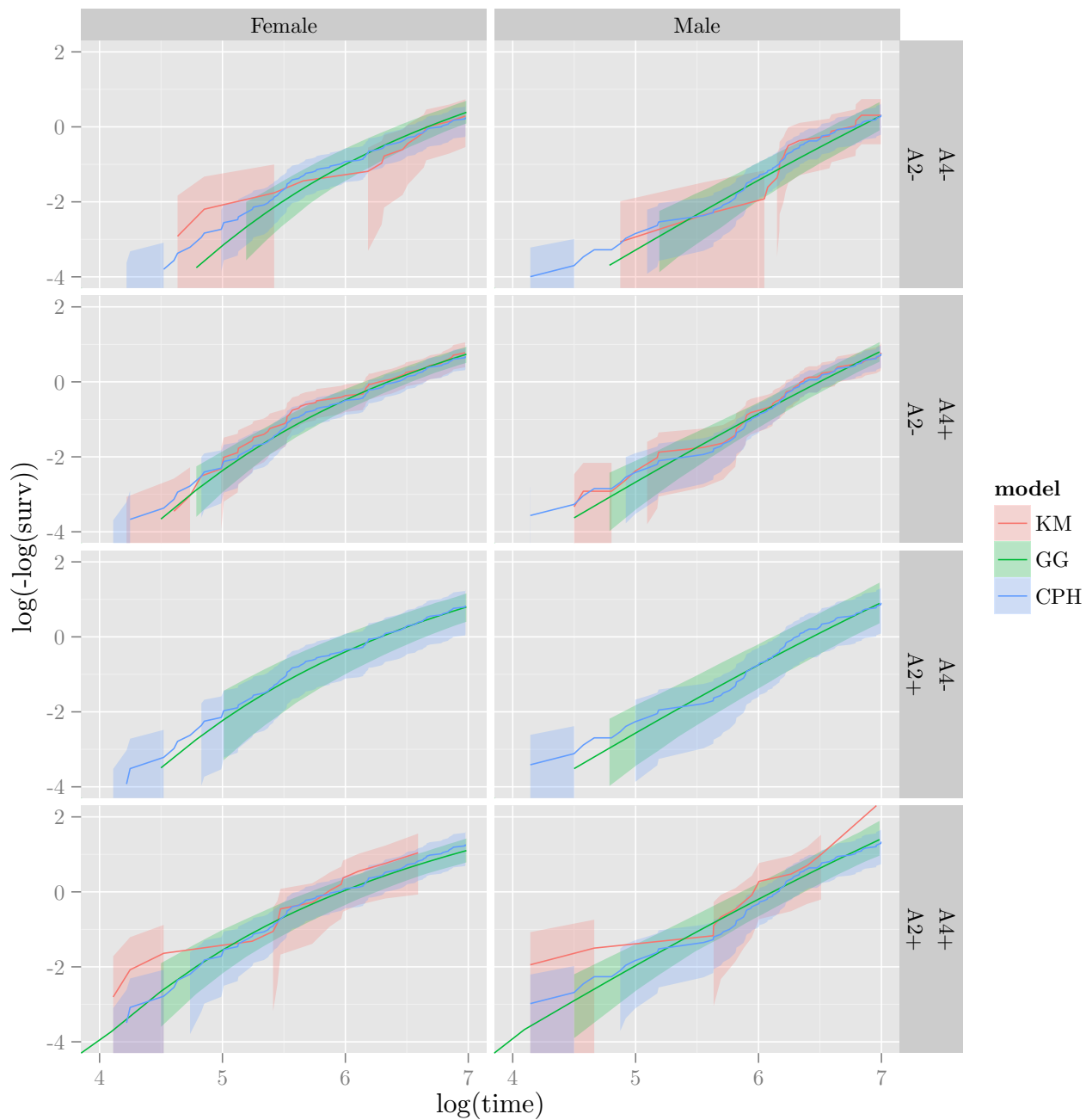
temp.survfit = survfit(Surv(Time, DSD) ~ SexM + A2 + A4, data)
temp.data = data.frame(time = temp.survfit$time, surv = temp.survfit$surv, upper = temp.survfit$lower, 1)
temp.data = rbind(temp.data, data.frame(time = temp.preds2$time, surv = temp.preds2$est, upper = temp.preds2$upper))
temp.data = rbind(temp.data, data.frame(time = temp.preds.cox$time, surv = temp.preds.cox$surv, upper = temp.preds.cox$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)

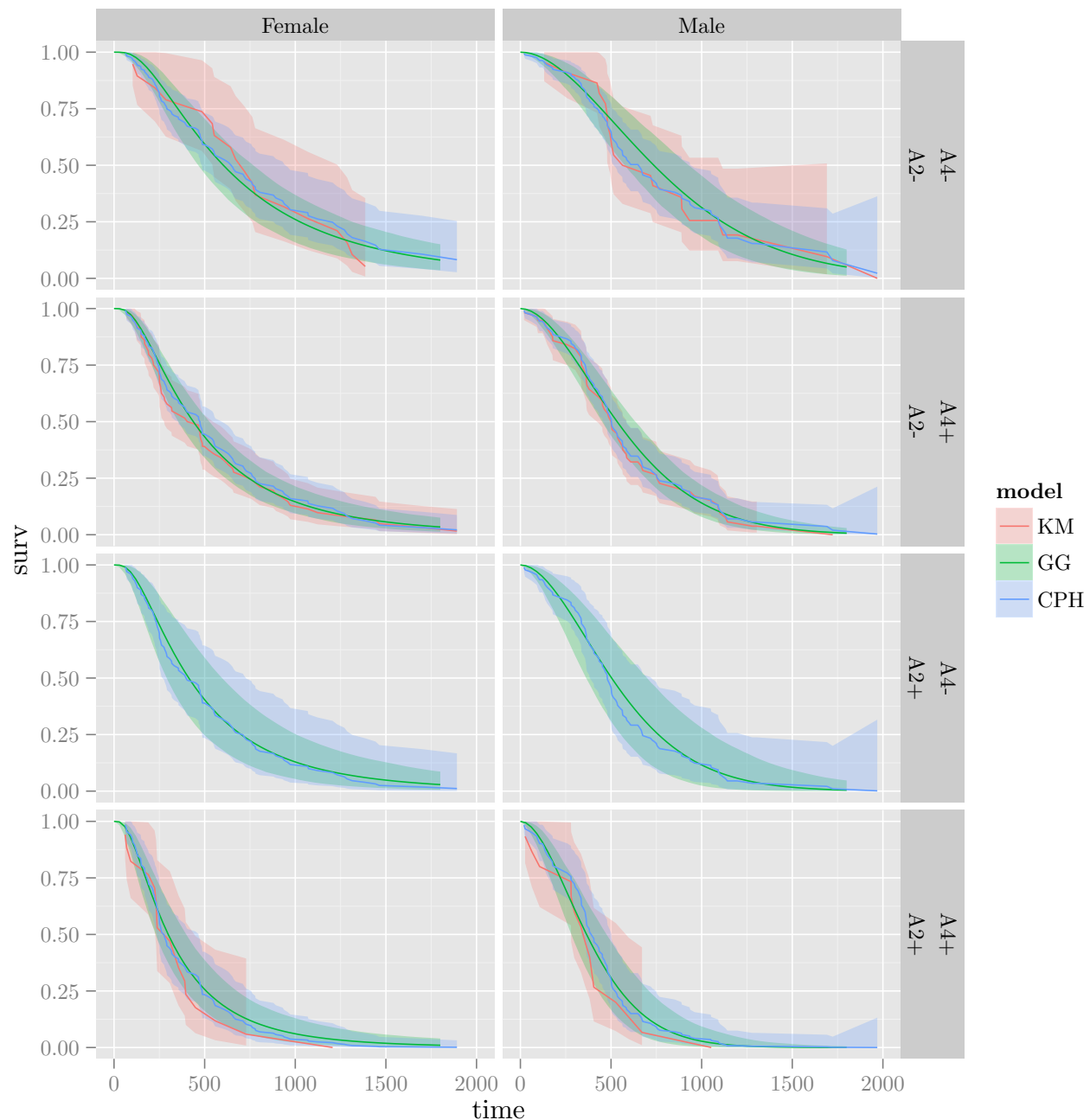
## Warning: Removed 46 rows containing missing values (geom_path).
## Warning: Removed 41 rows containing missing values (geom_path).
## Warning: Removed 48 rows containing missing values (geom_path).
## Warning: Removed 44 rows containing missing values (geom_path).
## Warning: Removed 39 rows containing missing values (geom_path).
## Warning: Removed 37 rows containing missing values (geom_path).
## Warning: Removed 40 rows containing missing values (geom_path).
## Warning: Removed 38 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 == FALSE))
# anc = list(
#   sigma = ~ SexM,
#   Q = ~ SexM),
# data = data, dist = "gengamma")
fit.gg2 = flexsurvreg(Surv(Time, DSD) ~ SexM + SizeCent + A2 + A4,
  anc = list(
    sigma = ~ SexM * A2 * A4,
    Q = ~ SexM * A2 * A4),
```

```

data = data, dist = "gengamma")

## Warning in flexsurvreg(Surv(Time, DSD) ~ SexM + SizeCent + A2 + A4, anc = list(sigma = ~SexM
* : Optimisation has probably not converged to the maximum likelihood - Hessian is not positive
definite.

fit.gg2

##
## Call:
## flexsurvreg(formula = Surv(Time, DSD) ~ SexM + SizeCent + A2 +      A4, anc = list(sigma = ~SexM * A2
##
## Estimates:
##
##              data mean  est      L95%      U95%
## mu              NA      6.0860      NA      NA
## sigma            NA      0.9904      NA      NA
## Q                NA     -0.4199      NA      NA
## SexMTRUE         0.4847      0.6227      NA      NA
## SizeCent         3.8214     -0.0123      NA      NA
## A2TRUE           0.1735     -0.3551      NA      NA
## A4TRUE           0.7806     -0.1720      NA      NA
## sigma(SexMTRUE)  0.4847     -0.6193      NA      NA
## sigma(A2TRUE)    0.1735     -2.3753      NA      NA
## sigma(A4TRUE)    0.7806     -0.2541      NA      NA
## sigma(SexMTRUE:A2TRUE) 0.0816     -2.1227      NA      NA
## sigma(SexMTRUE:A4TRUE) 0.3673      0.4070      NA      NA
## sigma(A2TRUE:A4TRUE)  0.1633      2.3410      NA      NA
## sigma(SexMTRUE:A2TRUE:A4TRUE) 0.0765      2.0758      NA      NA
## Q(SexMTRUE)      0.4847      1.1046      NA      NA
## Q(A2TRUE)        0.1735      0.0461      NA      NA
## Q(A4TRUE)        0.7806      0.0959      NA      NA
## Q(SexMTRUE:A2TRUE) 0.0816     -0.1042      NA      NA
## Q(SexMTRUE:A4TRUE) 0.3673      0.1483      NA      NA
## Q(A2TRUE:A4TRUE)  0.1633     -0.2241      NA      NA
## Q(SexMTRUE:A2TRUE:A4TRUE) 0.0765      0.5290      NA      NA
##
##              se      exp(est)  L95%      U95%
## mu              NA      NA      NA      NA
## sigma            NA      NA      NA      NA
## Q                NA      NA      NA      NA
## SexMTRUE         NA      1.8639      NA      NA
## SizeCent         NA      0.9877      NA      NA
## A2TRUE           NA      0.7011      NA      NA
## A4TRUE           NA      0.8420      NA      NA
## sigma(SexMTRUE)  NA      0.5383      NA      NA
## sigma(A2TRUE)    NA      0.0930      NA      NA
## sigma(A4TRUE)    NA      0.7756      NA      NA
## sigma(SexMTRUE:A2TRUE) NA      0.1197      NA      NA
## sigma(SexMTRUE:A4TRUE) NA      1.5023      NA      NA
## sigma(A2TRUE:A4TRUE) NA     10.3914      NA      NA
## sigma(SexMTRUE:A2TRUE:A4TRUE) NA      7.9708      NA      NA
## Q(SexMTRUE)      NA      3.0180      NA      NA
## Q(A2TRUE)        NA      1.0472      NA      NA
## Q(A4TRUE)        NA      1.1006      NA      NA
## Q(SexMTRUE:A2TRUE) NA      0.9010      NA      NA
## Q(SexMTRUE:A4TRUE) NA      1.1598      NA      NA

```

```
## Q(A2TRUE:A4TRUE)          NA    0.7992      NA      NA
## Q(SexMTRUE:A2TRUE:A4TRUE) NA    1.6973      NA      NA
##
## N = 196, Events: 188, Censored: 8
## Total time at risk: 113142
## Log-likelihood = -1349, df = 21
## AIC = 2741

AIC(fit.gg)

## [1] 2727

AIC(fit.gg2)

## [1] 2741

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

## [1] -13.54

# Equivocal on AIC. BIC would favour gg then.

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

## [1] 0.001217

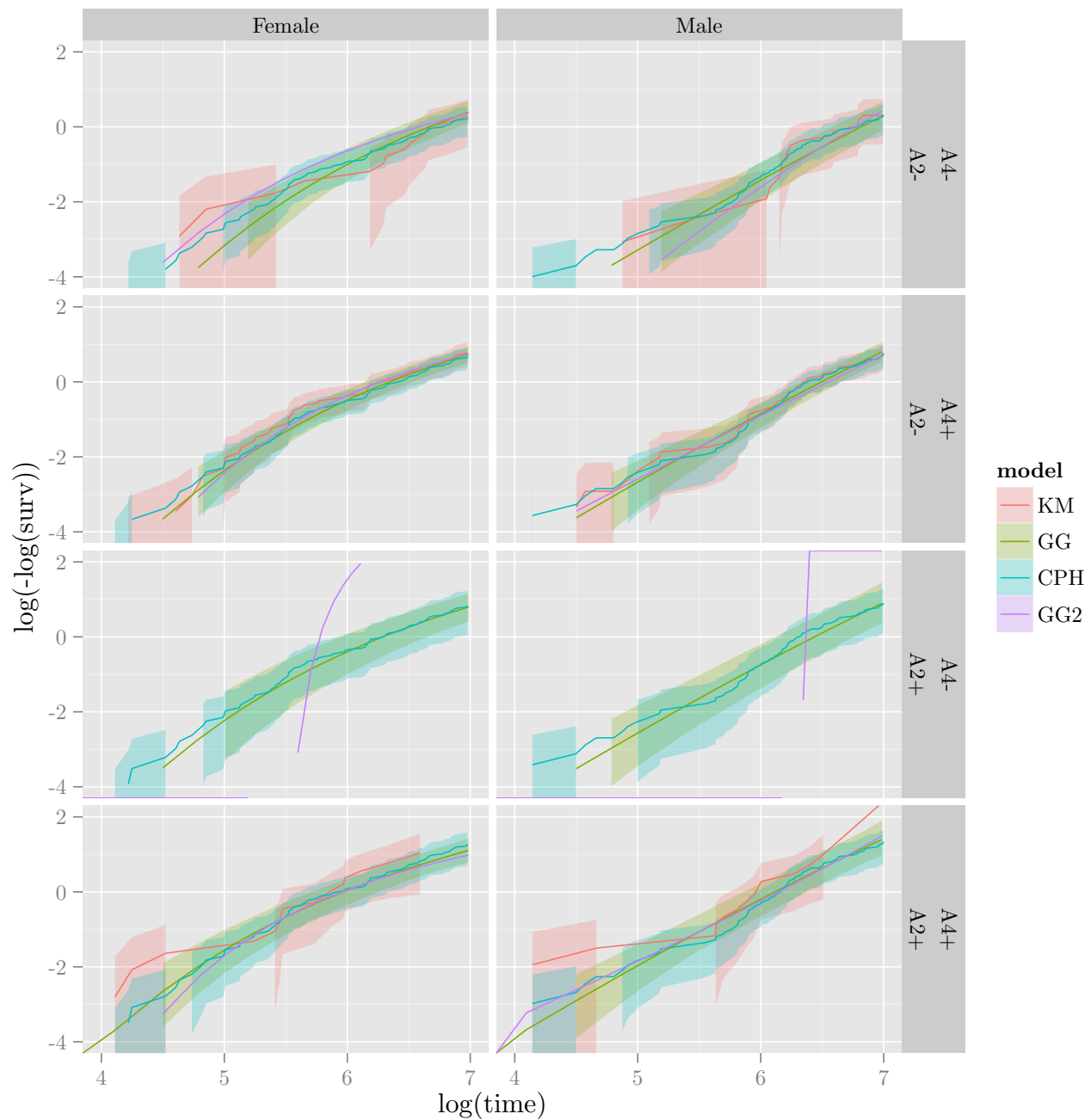
# Not good evidence on LRT
```

See how it plots relative to the others.

```
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$est, upper = temp.preds2$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]

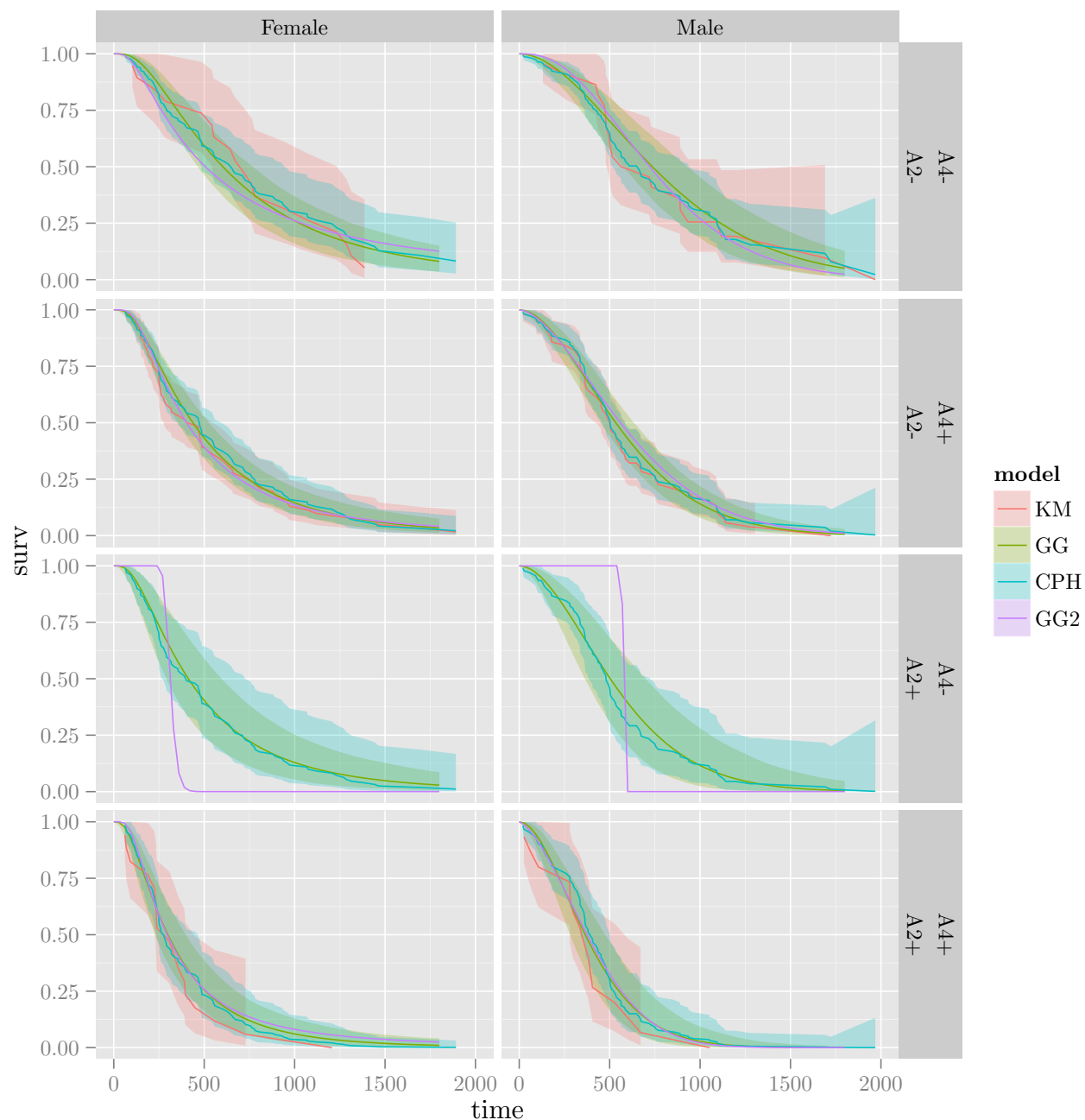
ggplot(temp.data, aes(x = log(time), y = log(-log(surv)), ymin = log(-log(lower)), ymax = log(-log(upper)))) +
  geom_ribbon(alpha = 0.25, colour = NA) +
  geom_line() +
  xlim(4, 7) + ylim(-4, 2) +
  facet_grid(A2 ~ A4 ~ Sex)

## Warning: Removed 71 rows containing missing values (geom_path).
## Warning: Removed 66 rows containing missing values (geom_path).
## Warning: Removed 73 rows containing missing values (geom_path).
## Warning: Removed 69 rows containing missing values (geom_path).
## Warning: Removed 85 rows containing missing values (geom_path).
## Warning: Removed 62 rows containing missing values (geom_path).
## Warning: Removed 65 rows containing missing values (geom_path).
## Warning: Removed 63 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).
```

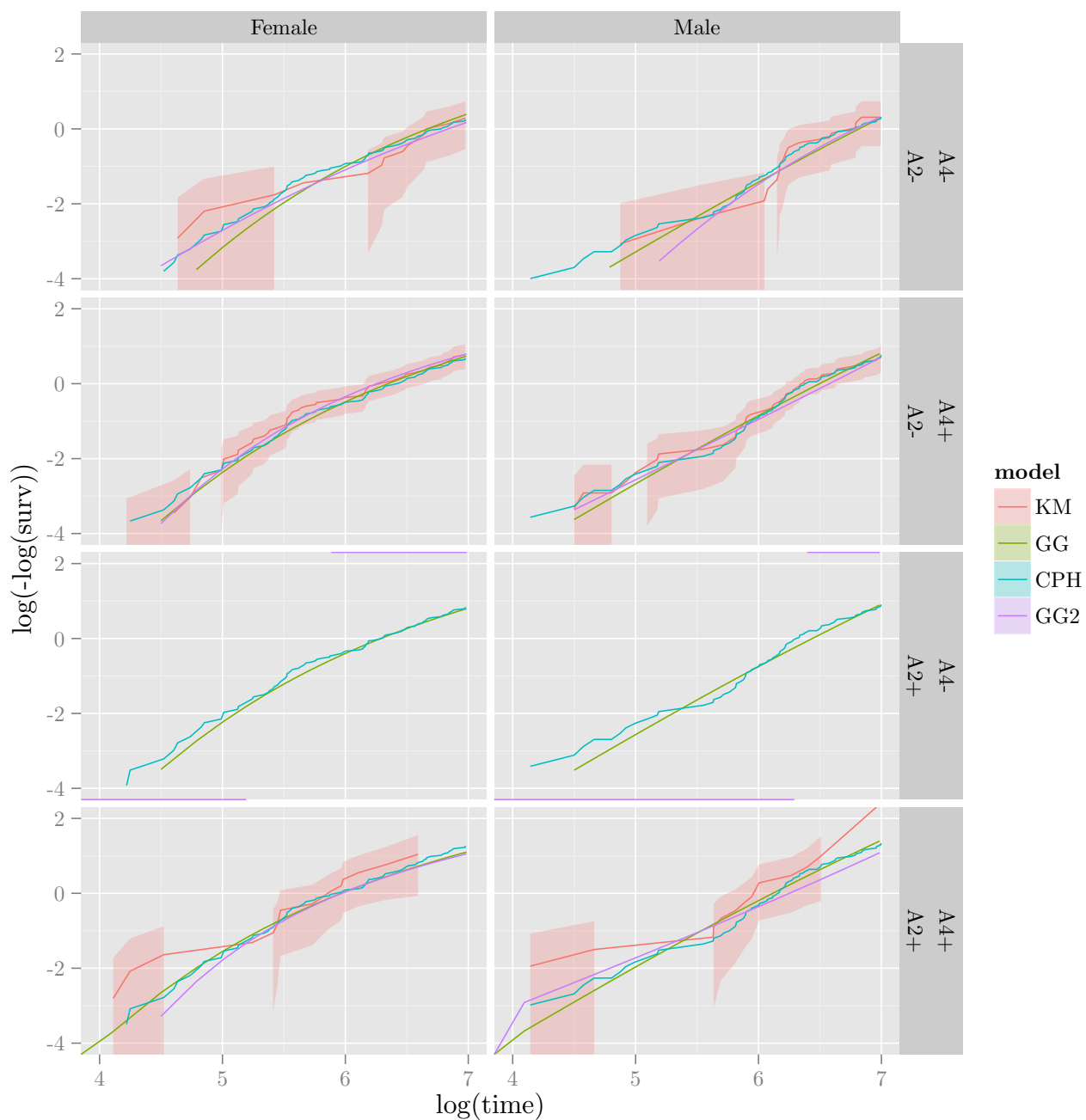


An alternative take, showing errors with the KMs only.

```
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 66 rows containing missing values (geom_path).
 ## Warning: Removed 73 rows containing missing values (geom_path).
 ## Warning: Removed 69 rows containing missing values (geom_path).

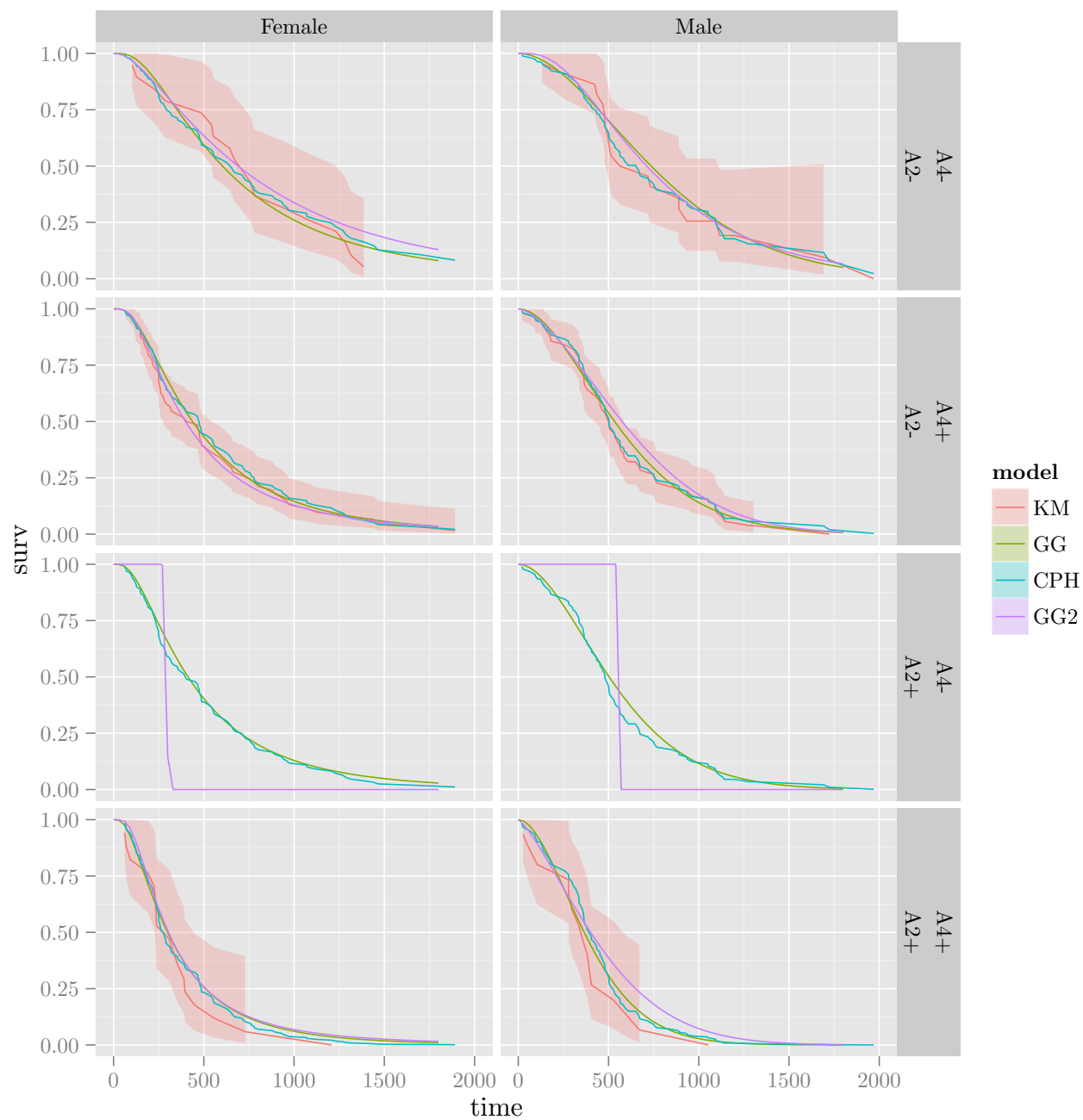
```
## Warning: Removed 64 rows containing missing values (geom_path).
## Warning: Removed 62 rows containing missing values (geom_path).
## Warning: Removed 65 rows containing missing values (geom_path).
## Warning: Removed 63 rows containing missing values (geom_path).
```



```
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)
```

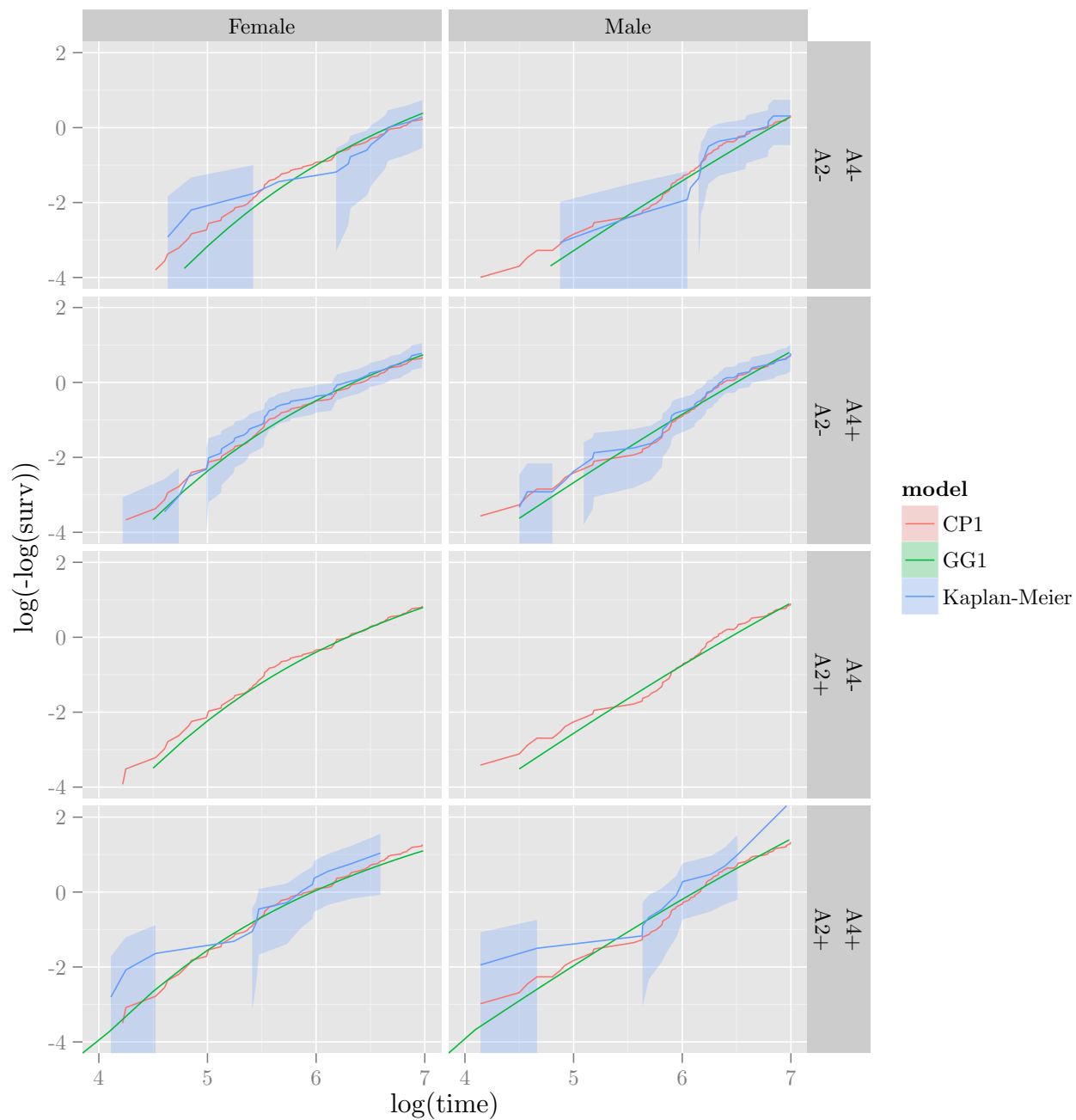
```
## Warning: Removed 3 rows containing missing values (geom_path).
## Warning: Removed 3 rows containing missing values (geom_path).
```

```
## 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 41 rows containing missing values (geom_path).
## Warning: Removed 48 rows containing missing values (geom_path).
## Warning: Removed 44 rows containing missing values (geom_path).
## Warning: Removed 39 rows containing missing values (geom_path).
## Warning: Removed 37 rows containing missing values (geom_path).
## Warning: Removed 40 rows containing missing values (geom_path).
## Warning: Removed 38 rows containing missing values (geom_path).
```

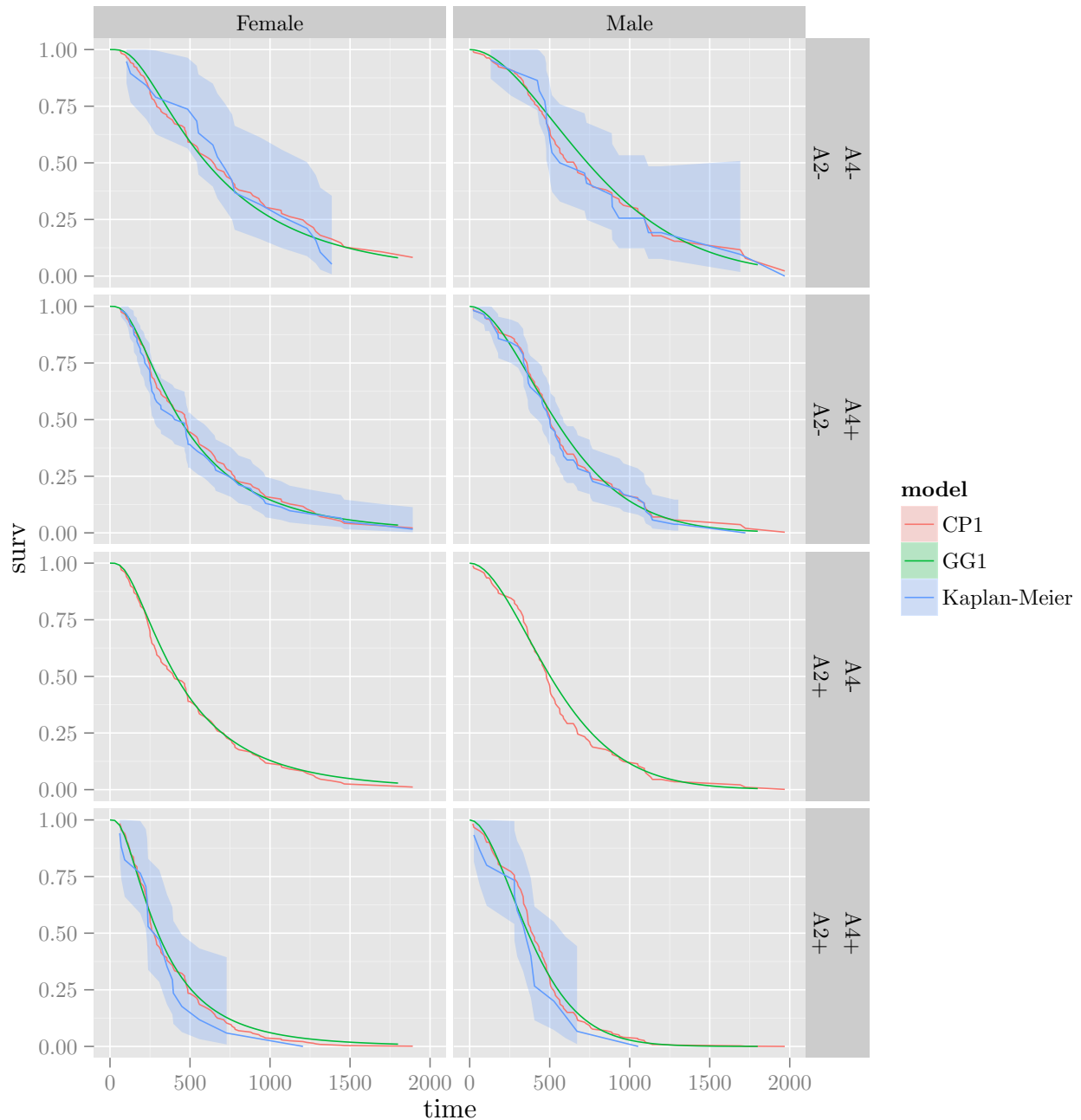


```
ggplot(temp.data, aes(x = time, y = surv, ymin = lower, ymax = upper, colour = model, fill = model)) +
  geom_ribbon(alpha = 0.25, colour = NA) +
  geom_line() + xlim(0, 2000) + ylim(0, 1) +
```



```
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, woo.

```

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, yright = 0)
  marg_cens_func = approxfun(marg_censfit$time, marg_censfit$surv, method = "constant", yleft = 1, yright = 0)

  pred_funcs = apply(pred, 1, function(pat_preds) approxfun(pred_times, pat_preds, yleft = 1, yright = 0))

  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*(observed_time >= max_time)
    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[category == 2])
    bsc[category == 3] = 0
  }

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

  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 calculate 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))) / length(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$Time))
ibs_preds_gg = as.matrix(t(sapply(summary(fit.gg, newdata = data.val, type = "survival", t = ibs_times),
  function(t) survfit(fit.gg, newdata = data.val, type = "survival", t = t))))
ibs_preds_gg2 = as.matrix(t(sapply(summary(fit.gg2, newdata = data.val, type = "survival", t = ibs_times),
  function(t) survfit(fit.gg2, newdata = data.val, type = "survival", t = t))))
temp_cox_preds = survfit(fit.cph, newdata = data.val)
ibs_preds_cph = simplify2array(tapply(1:length(temp_cox_preds$time), rep(names(temp_cox_preds$strata), length(temp_cox_preds$time)),
  function(strat_i) approxfun(temp_cox_preds$time[strat_i], temp_cox_preds$surv[strat_i], method = "constant", yleft = 1, yright = 0)),
  MARGINs = 2, SIMPLIFY = FALSE)
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$survival, 1, function(surv) approxfun(temp_rsf_preds$time.interest, surv, method = "constant", yleft = 1, yright = 0)))
# Patients (from data.val) are in rows, times (from ibs_times) in columns.

# Add a no-information KM predictor

```

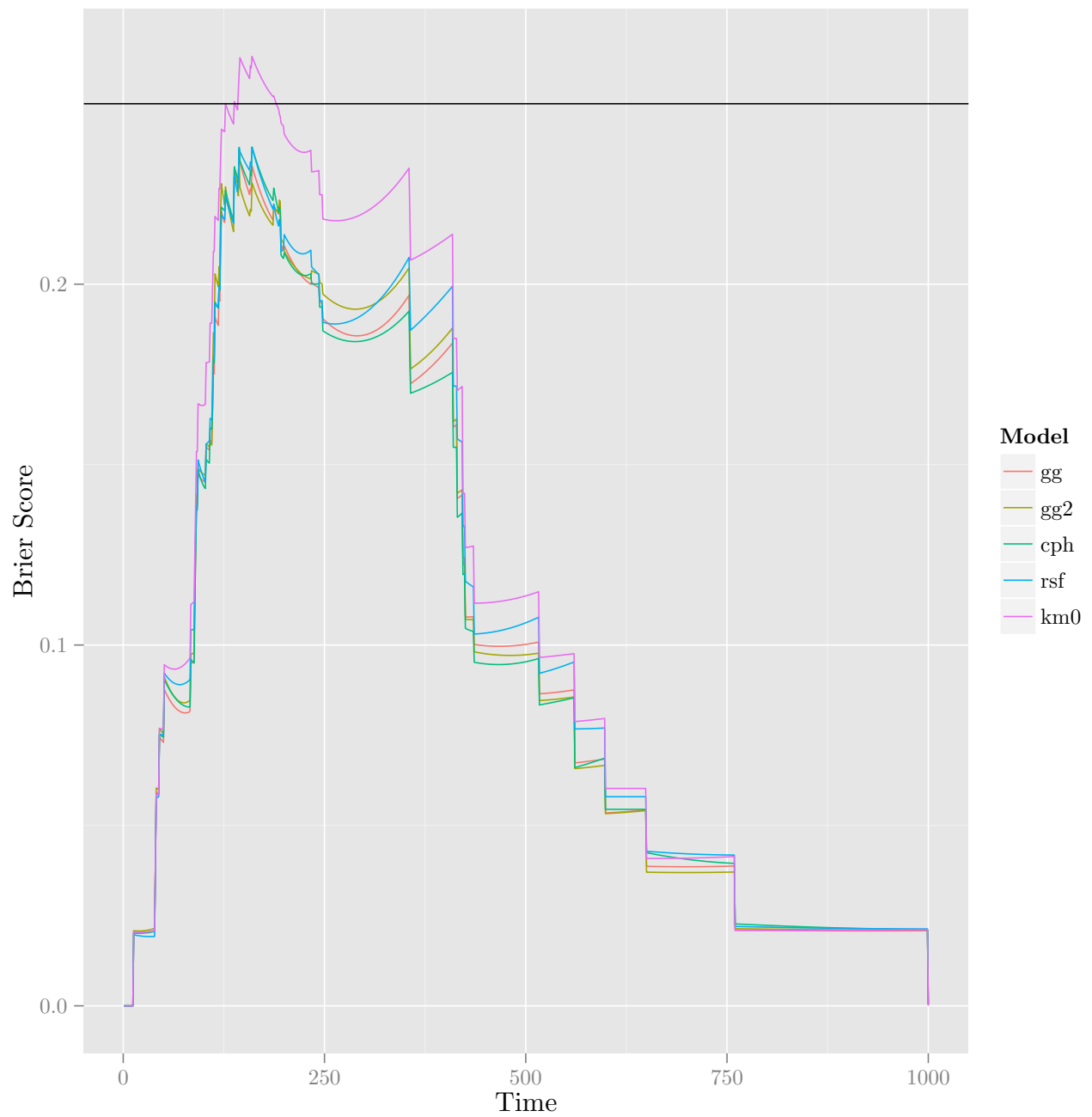
```
temp_km0 = survfit(Surv(Time, DSD) ~ 1, data)
ibs_preds_km0 = t(matrix(rep(approx(temp_km0$time, temp_km0$surv, xout = ibs_times, method = "constant",
ibs_preds_all = list(gg = ibs_preds_gg, gg2 = ibs_preds_gg2, cph = ibs_preds_cph, rsf = ibs_preds_rsf, km0 = ibs_preds_km0))
```

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

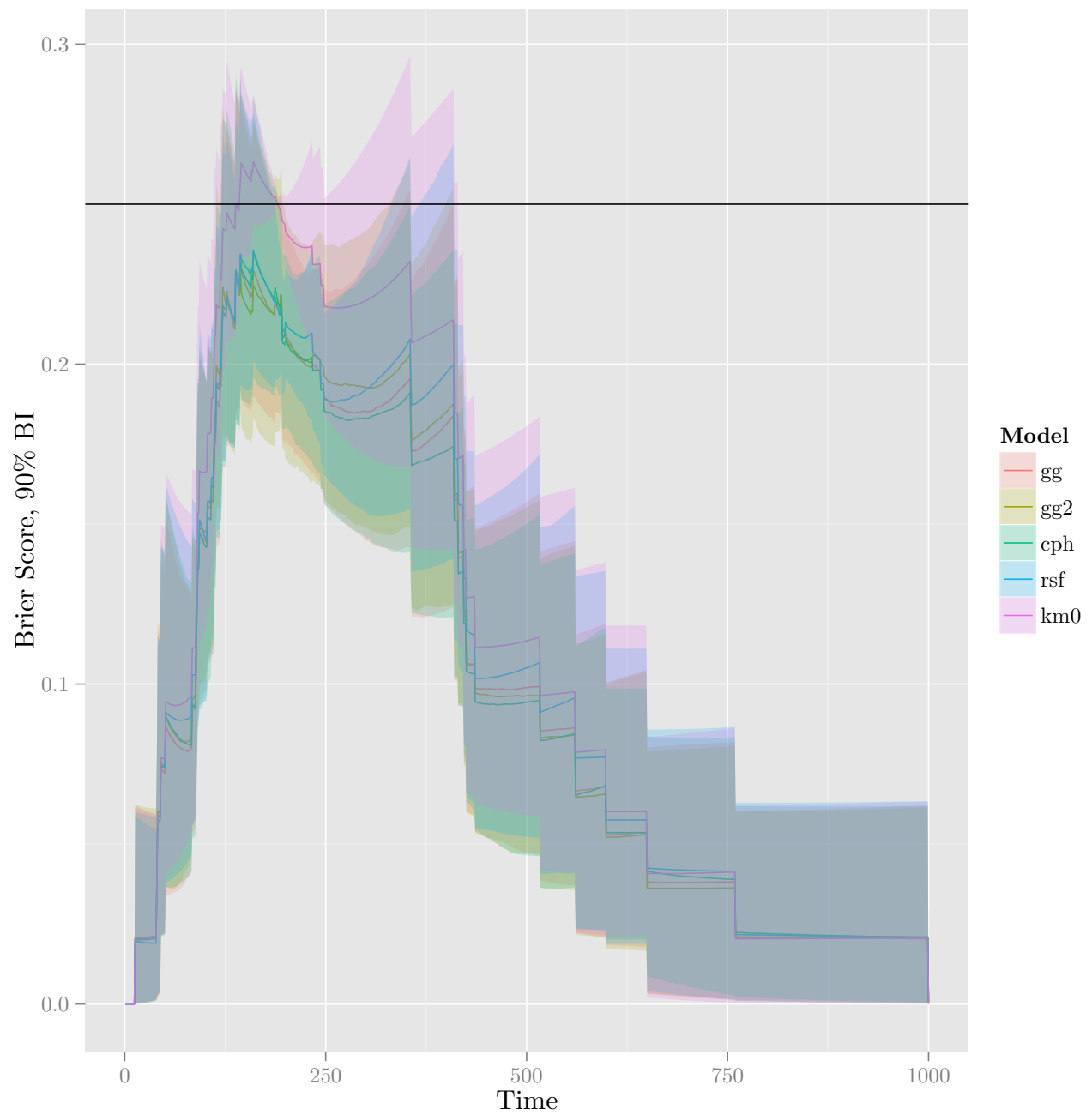
```
set.seed(20150111)
ibs_eval_times = calcIBS(Surv(data.val$Time, data.val$DSD), ibs_preds_gg, ibs_times, max(data.val$Time))
# bsc_boot2 = lapply(ibs_preds_all, function(preds) boot(data.val, statistic = function(d, i) calcIBS(Surv(d$Time, d$DSD), preds, ibs_times, max(d$Time)))
# bsc_boot2ci = lapply(ibs_preds_all,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

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

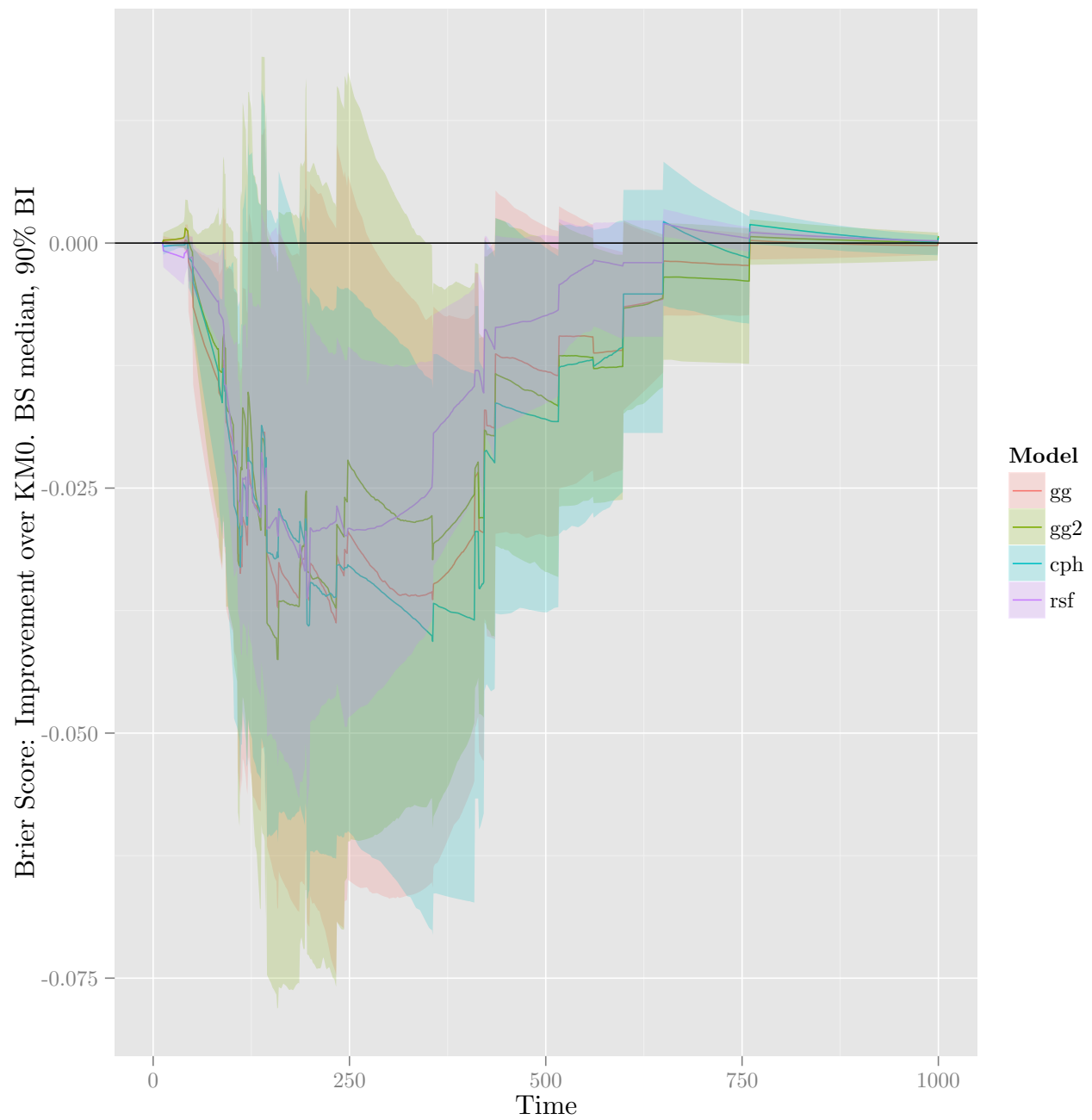
```
temp = sapply(list(gg = ibs_preds_gg, gg2 = ibs_preds_gg2, cph = ibs_preds_cph, rsf = ibs_preds_rsf, km0 = ibs_preds_km0), function(x) boot.ci(boot(data.val, statistic = function(d, i) calcIBS(Surv(d$Time, d$DSD), x, ibs_times, max(d$Time))),
temp = melt(temp)
colnames(temp) = c("Time", "Model", "BS")
ggplot(temp, aes(x = Time, y = BS, colour = Model)) + geom_line() + ylab("Brier Score") + geom_hline(yintercept = 0.25)
```



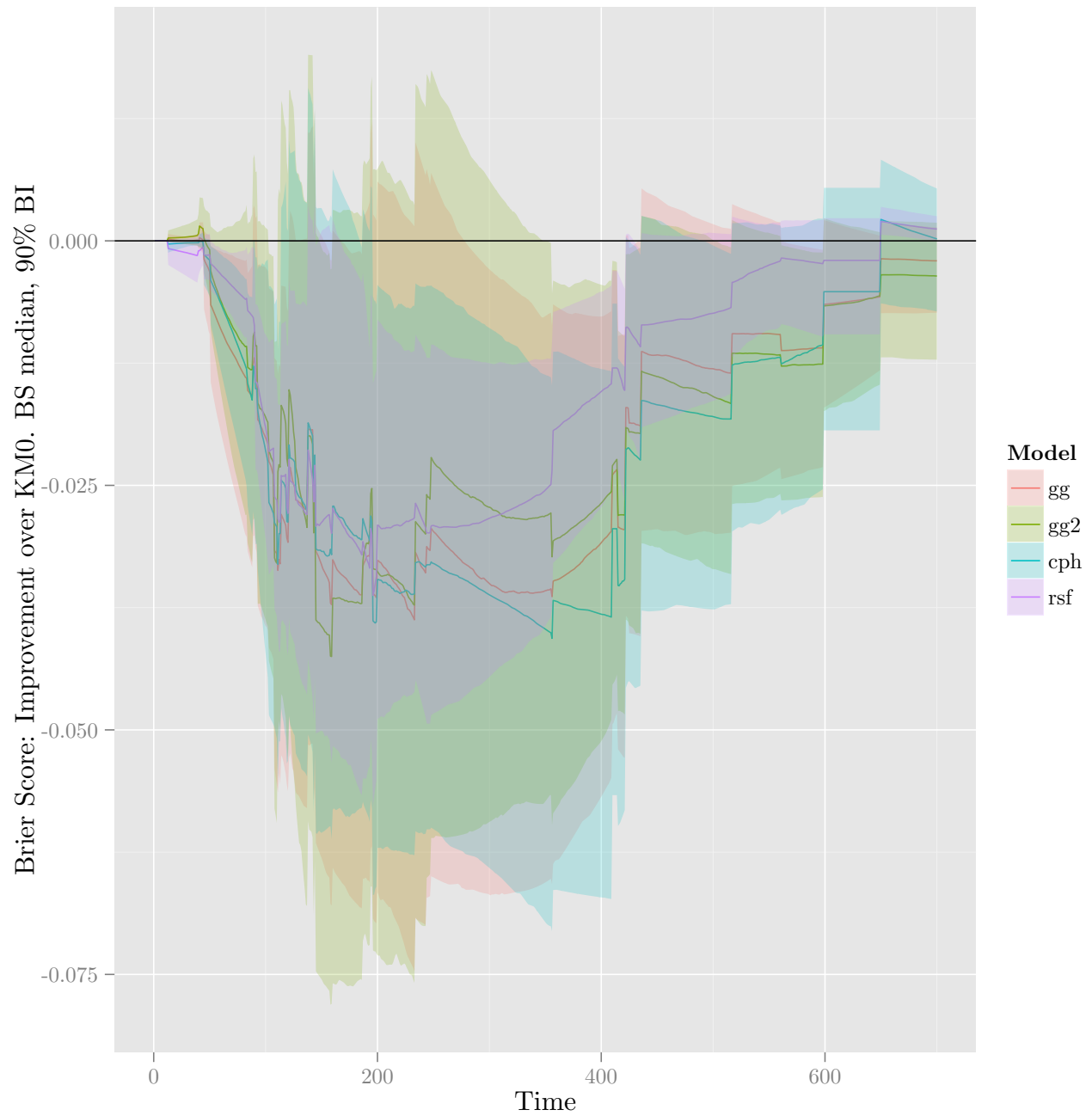
```
temp = melt(aapply(bsc_boots, 2:3, quantile, probs = c(0.05, 0.5, 0.95)))
colnames(temp) = c("Model", "Time", "Quantile", "Value")
temp$Quantile = paste("Q", gsub("%", "", temp$Quantile), sep = "")
temp = dcast(temp, Model + Time ~ Quantile, value.var = "Value")
ggplot(temp, aes(x = Time, y = Q50,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



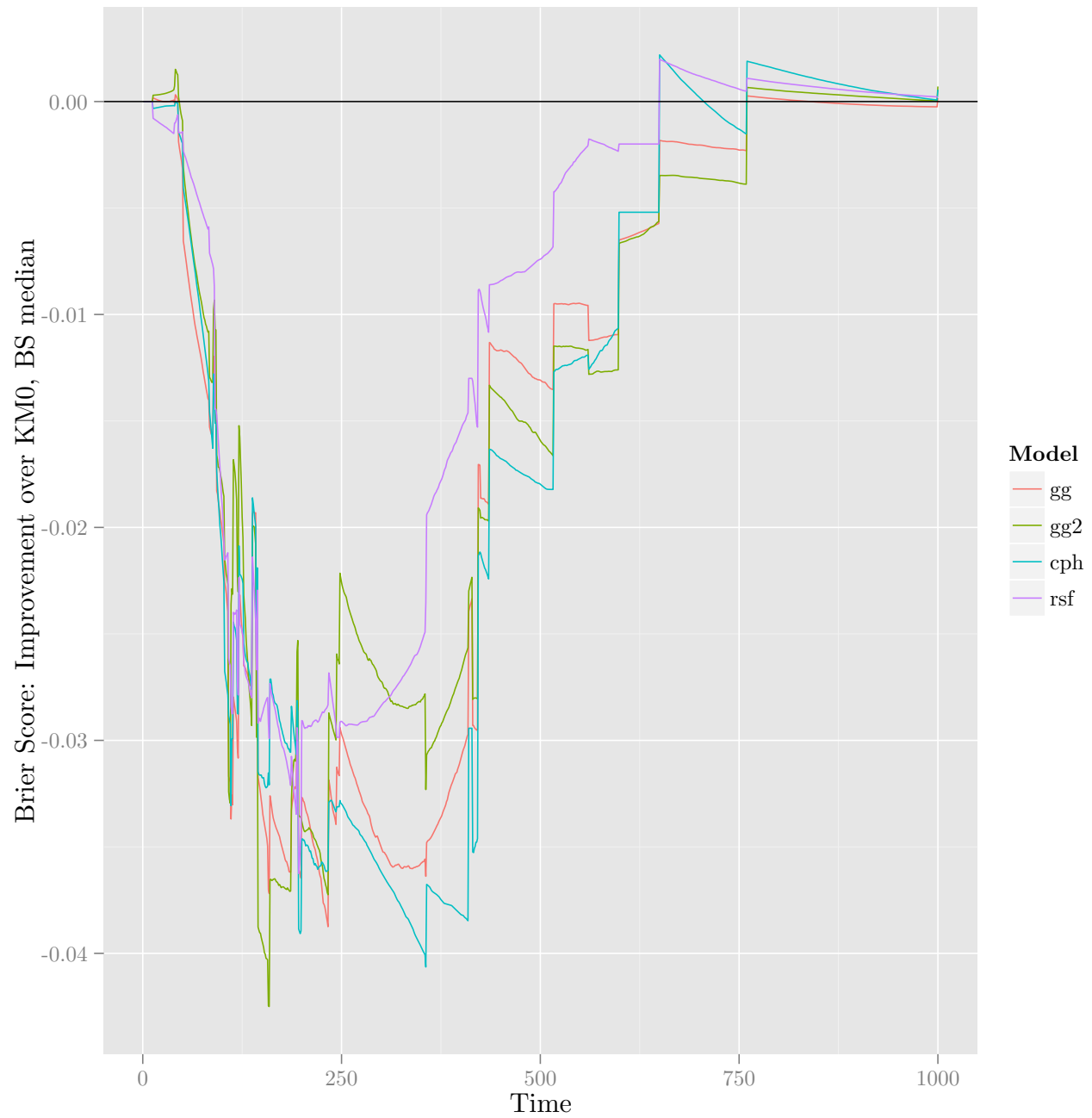
```
bsc_boots_diff = aapply(bsc_boots, 2, function(x) x - bsc_boots[,5,])[1:4,,]
temp = melt(aapply(bsc_boots_diff, c(1,3), quantile, probs = c(0.05, 0.5, 0.95)))
colnames(temp) = c("Model", "Time", "Quantile", "Value")
temp$Quantile = paste("Q", gsub("%", "", temp$Quantile), sep = "")
temp = dcast(temp,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



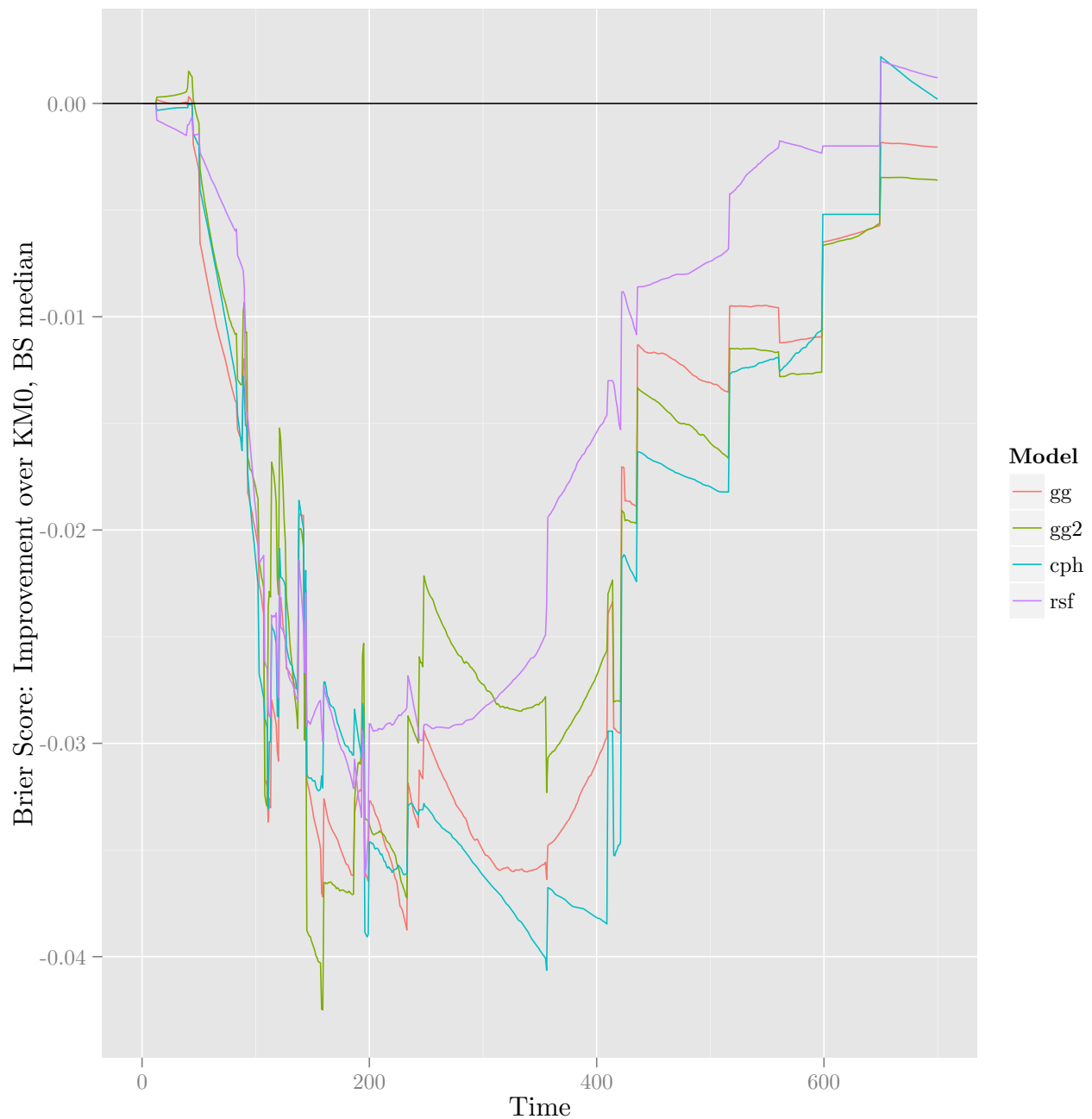
```
ggplot(temp, aes(x = Time, y = Q50, ymin = Q5, ymax = Q95, colour = Model, fill = Model)) + geom_line()
## Warning: Removed 1200 rows containing missing values (geom_path).
```



```
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 KM0, BS median")
## 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_time,
  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_rsfc[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] 162.1
```

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

```
## [1] 162.8
```

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

```
## [1] 161.2
```

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

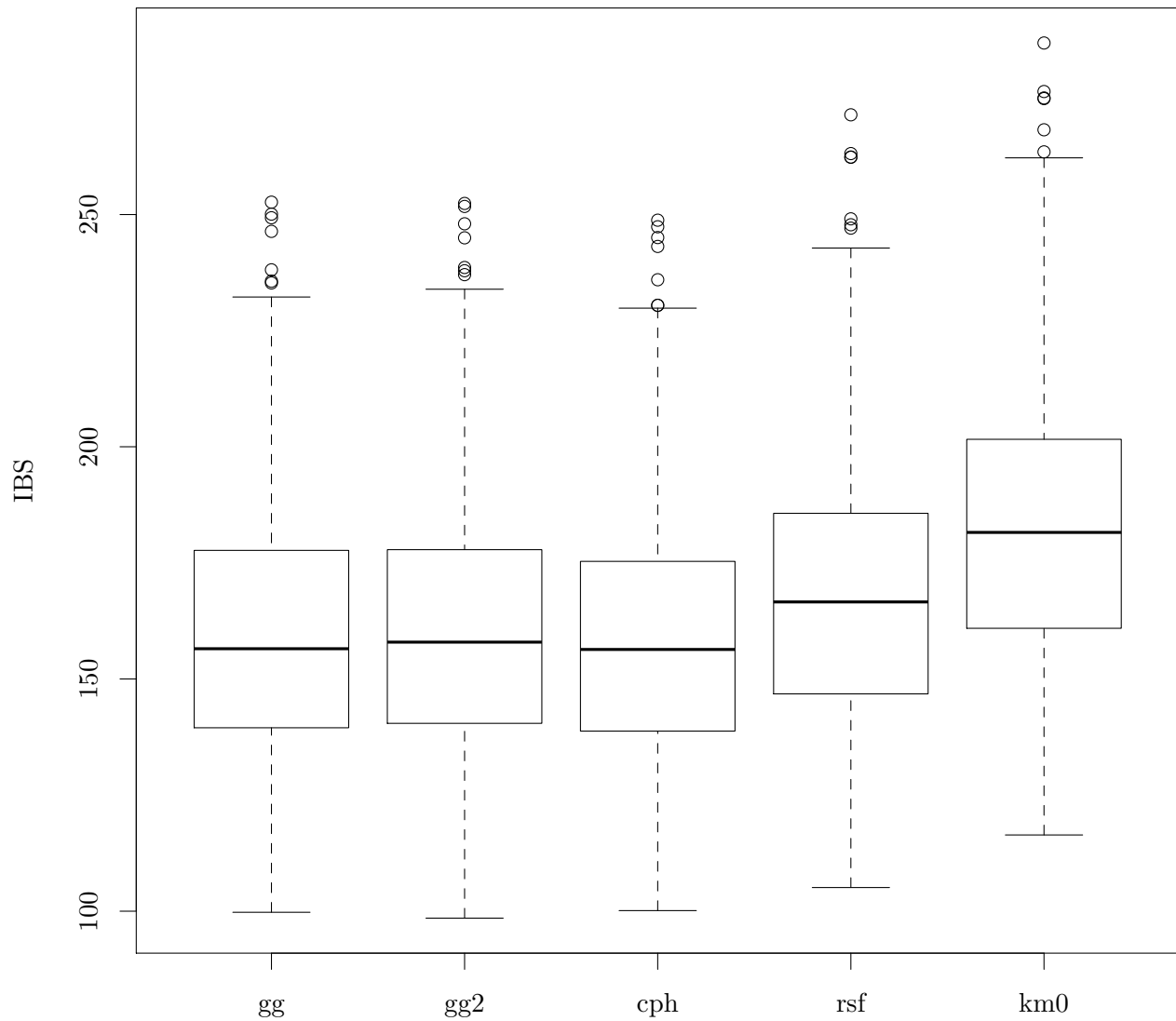
```
## [1] 170.1
```

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

```
## [1] 184.4
```

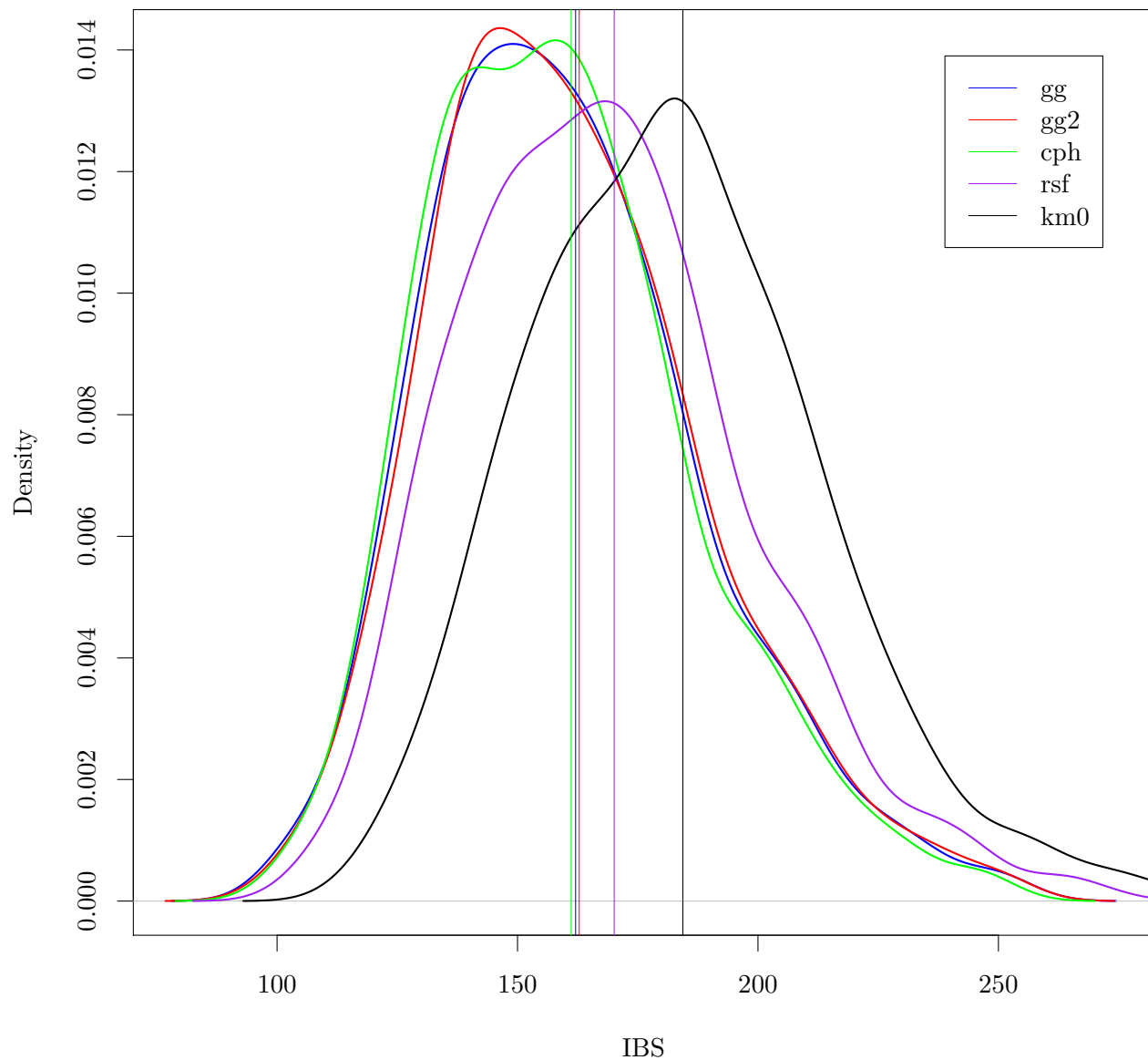
```
boxplot(ibsc_boots, main = "IBS BS Distribution", ylab = "IBS")
```

IBS BS Distribution



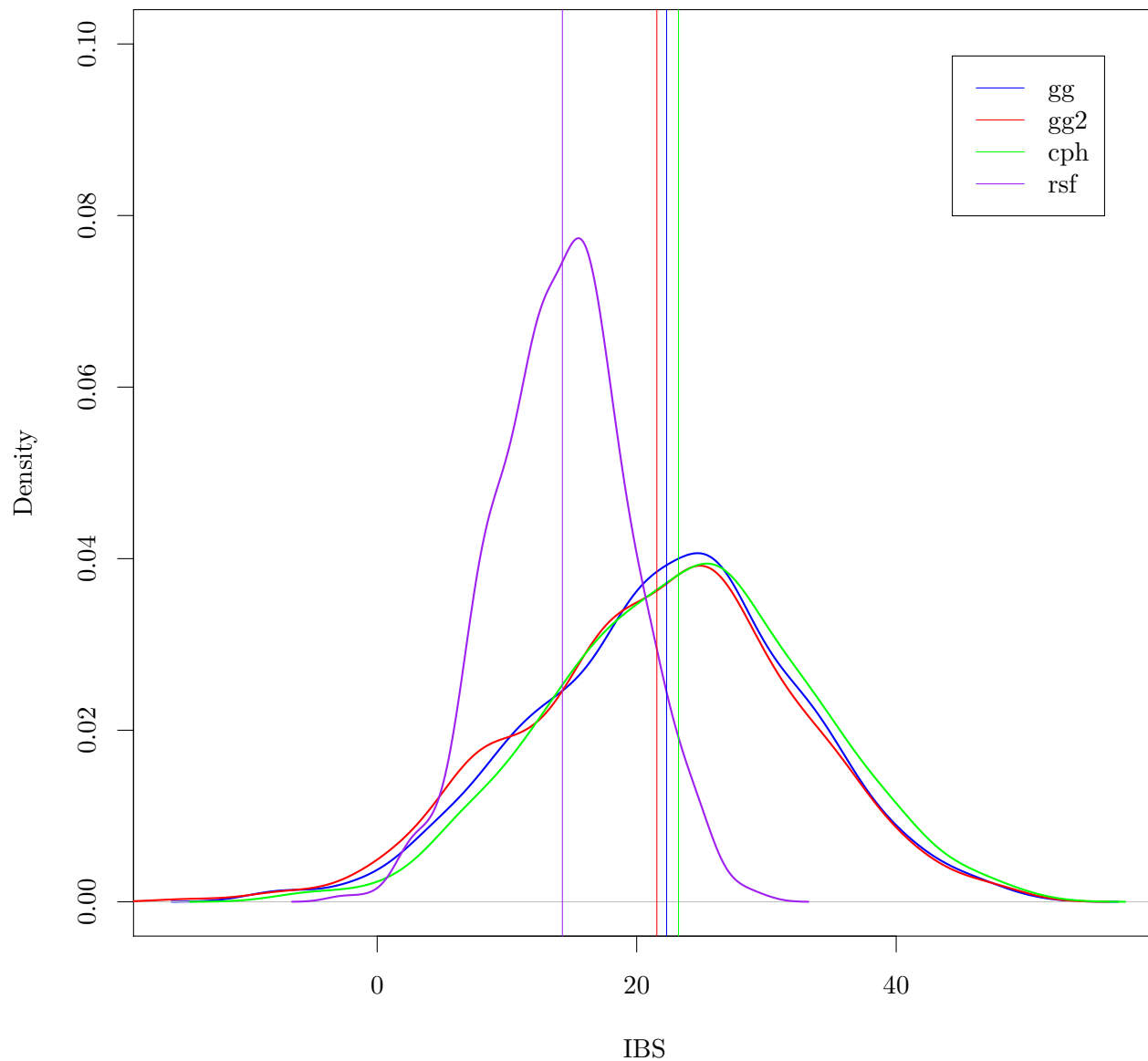
```
plot(density(ibsc_boots[,1]), col = "blue", lwd = 2, main = "IBS BS Distribution", xlab = "IBS")
lines(density(ibsc_boots[,2]), col = "red", lwd = 2)
lines(density(ibsc_boots[,3]), col = "green", lwd = 2)
lines(density(ibsc_boots[,4]), col = "purple", lwd = 2)
lines(density(ibsc_boots[,5]), col = "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_rsfc, 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", "black"))
```

IBS BS Distribution



```
plot(density(ibsc_boots[,5] - ibsc_boots[,1]), col = "blue", lwd = 2, main = "IBS\\_KMO - 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
```

IBS_KM0 - IBS_x BS Distribution



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_rsfc[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_rsfc[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*   -22.3076 -0.17520        9.532
## t2*   -21.5540 -0.31056        9.815
## t3*   -23.2248 -0.37205        9.365
## t4*   -14.2733 -0.06822        4.884
## t5*    -8.0342 -0.10698        5.437
## t6*    -7.2807 -0.24233        6.069
## t7*    -8.9515 -0.30383        5.362
## t8*     0.9172  0.19685        2.317
## t9*     1.6708  0.06150        3.412
## t10*   -0.7535  0.13536        2.108

ibsc_boots2_ci

##      level orderi1 orderi2      lci      uci
## gg-km0   0.95   13.01  488.9 -40.323 -1.9131
## gg2-km0   0.95   15.87  491.1 -40.496 -0.6571
## cph-km0   0.95   14.97  490.4 -41.702 -4.1385
## rsf-km0   0.95   10.17  486.0 -24.568 -5.1008
## gg-rsf    0.95   19.02  493.1 -17.119  5.0004
## gg2-rsf   0.95   21.94  494.3 -16.649  7.7191
## cph-rsf   0.95   15.74  490.9 -19.355  3.1486
## gg-cph    0.95   11.40  487.1  -3.381  5.8085
## gg2-cph   0.95   13.35  489.3  -5.026  8.7521
## gg-gg2    0.95   11.96  487.9  -4.463  3.4726
```

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_9  587 TRUE  TRUE     5  FALSE   10 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

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 = FALSE,
  set.seed(20150111))
fit.final.rsfc = rfsrc(Surv(Time, DSD) ~ SexM + AgeCent + LocBody + SizeCent + A2 + A4, data = data.all, y = FALSE,
  set.seed(20150111))
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.final.rsfc), "fit.RDS")
```

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] ggplot2_1.0.0      plyr_1.8.1          reshape2_1.4
##  [4] glmulti_1.0.7      rJava_0.9-6         risksetROC_1.0.4
##  [7] MASS_7.3-35        timeROC_0.2         timereg_1.8.6
## [10] mvtnorm_1.0-1      pec_2.4.4           randomForestSRC_1.5.5
## [13] boot_1.3-13        flexsurv_0.5        survival_2.37-7
## [16] tikzDevice_0.7.0   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
```