MATH 284/FMPH 291 Survival Analysis Supplementary Learing Materials

Yuyao Wang

2024-05-13

Tutorial

Here is the link of a tutorial for survival analysis in R by Emily C. Zabor: https://www.emilyzabor.com/tutorials/survival_analysis_in_r_tutorial.html#Part_1:_Introduction_to_Survival_Analysis

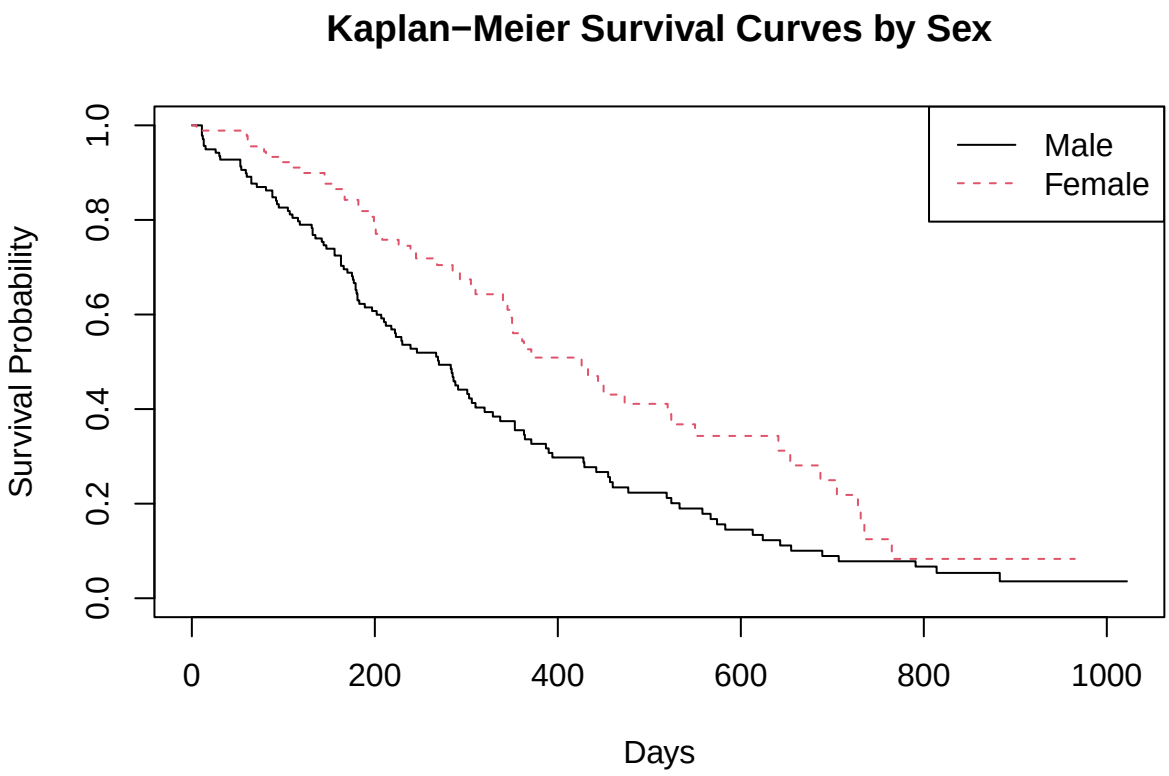
Part 1 in the above tutorial includes creating survival objects and curve, Kaplan-Meier plots, Estimating x-year survival, estimating median survival time, comparing survival times between groups (using log-rank test), and Cox regression model.

Below are some additional material for different topics.

Kaplan-Meier Curve

The tutorial uses the '`ggsurvfit`' R package for doing Kaplan-Meier plots. Alternatively, survival plots can be created using base R or the '`survminer`' package. For example, below is an example of fiting KM curves for the two sex groups using the 'lung' data set from the '`survival`' R package. You may find more details of the usage of the functions (for example how to add confidence intervals) by checking the R document for `survfit()` and `plot.survfit()`.

```
# Load the survival package
library(survival)
# Fit the Kaplan-Meier model stratified by sex
# The lung dataset is automatically available after loading the survival package
km_fit <- survfit(Surv(time, status) ~ sex, data = lung)
# Plotting the Kaplan-Meier curves for two sex groups
plot(km_fit, main = "Kaplan-Meier Survival Curves by Sex",
        xlab = "Days", ylab = "Survival Probability", col = c(1, 2), lty = 1:2)
# Adding a legend to the plot
legend("topright", legend = c("Male", "Female"), col = c(1, 2), lty = 1:2)
```



Log-rank test

The function `survdiff()` used in the above tutorial for comparing survival times between groups is a function that conduct G-rho family tests. The default `rho=0`, which corresponds to log-rank test. More details can be found in the R documentation of this package. An alternative way of doing logrank test is using the score test for coxph score test. Try the code below, and you will find the two approaches gives exactly the same results!

BTW, `survdiff()` does not handle left truncated data, but `coxph()` can properly handle it by specifying the survival object to be `Surv(Q,X,Delta)`, where Q is the left truncation time, X is the censored event time, and Delta is the event indicator.

```
# logrank test using G^\rho test with rho = 0
logrank.1 = survdiff(Surv(time, status) ~ sex, data = lung)
print(logrank.1)
```

```
## Call:
## survdiff(formula = Surv(time, status) ~ sex, data = lung)
##
##         N Observed Expected (O-E)^2/E (O-E)^2/V
## sex=1 138      112     91.6      4.55      10.3
## sex=2  90       53     73.4      5.68      10.3
##
##  Chisq= 10.3  on 1 degrees of freedom, p= 0.001
```

```
# logrank test using coxph score test
coxphfit = coxph(Surv(time, status) ~ sex, data = lung)
summary(coxphfit)
```

```
## Call:
## coxph(formula = Surv(time, status) ~ sex, data = lung)
##
##   n= 228, number of events= 165
##
##        coef exp(coef) se(coef)      z Pr(>|z|)
## sex -0.5310    0.5880   0.1672 -3.176  0.00149 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##     exp(coef) exp(-coef) lower .95 upper .95
## sex     0.588      1.701    0.4237     0.816
##
## Concordance= 0.579  (se = 0.021 )
## Likelihood ratio test= 10.63  on 1 df,   p=0.001
## Wald test            = 10.09  on 1 df,   p=0.001
## Score (logrank) test = 10.33  on 1 df,   p=0.001
```

The dual between estimating T distribution and estimating C distribution

The observed censored event time $X = \min(T, C)$ not only contains information of the event time T, but also information of the censoring time C. The even indicator $\Delta = I(T < C)$. In other words, when $\Delta = 1$, $X = T$; when $\Delta = 0$, $X = C$.

When we focus on estimating T distribution, T is censored by C, we construct survival object using `Surv(X, delta)`. When we focus on estimating C distribution, C can be viewed as censored by T. With this point of view think about how can we construct the survival object.

Notes on the exponenetial model and weibull model fitted using `survreg()`

The location-scale parameterization of a Weibull distribution found in `survreg` is not the same as the parameterization of rweibull. See the Example for `survreg()` and the chunk for `survreg.distributions` in '`survival`' R package documentation for more details on how to interpret the outputs.

When fitting an exponential model using `fit = survreg(..., dist = "exponential")`, `1/exp(fit$icoef)` correspond to the rate parameter in `rexp()`. Try the example below.

```r
n=10000
lambda = 3
TT = rexp(n, rate = lambda)
C = runif(n, 1, 2)
X = pmin(TT,C)
delta = as.numeric(TT<C)

dat = data.frame(X = X, delta = delta)

fit = survreg(Surv(X, delta)~1, data = dat, dist = "exponential")
1/exp(fit$icoef)
```

```
## (Intercept)
##    2.993624
```

Cox proportional hazards regression

We will use the `lung` data set in the `survival` R package as an example.

```
summary(lung)
```

```
##       inst             time            status           age        
##  Min.   : 1.00   Min.   :   5.0   Min.   :1.000   Min.   :39.00  
##  1st Qu.: 3.00   1st Qu.: 166.8   1st Qu.:1.000   1st Qu.:56.00  
##  Median :11.00   Median : 255.5   Median :2.000   Median :63.00  
##  Mean   :11.09   Mean   : 305.2   Mean   :1.724   Mean   :62.45  
##  3rd Qu.:16.00   3rd Qu.: 396.5   3rd Qu.:2.000   3rd Qu.:69.00  
##  Max.   :33.00   Max.   :1022.0   Max.   :2.000   Max.   :82.00  
##  NA's   :1                                                       
##       sex           ph.ecog          ph.karno        pat.karno     
##  Min.   :1.000   Min.   :0.0000   Min.   : 50.00   Min.   : 30.00  
##  1st Qu.:1.000   1st Qu.:0.0000   1st Qu.: 75.00   1st Qu.: 70.00  
##  Median :1.000   Median :1.0000   Median : 80.00   Median : 80.00  
##  Mean   :1.395   Mean   :0.9515   Mean   : 81.94   Mean   : 79.96  
##  3rd Qu.:2.000   3rd Qu.:1.0000   3rd Qu.: 90.00   3rd Qu.: 90.00  
##  Max.   :2.000   Max.   :3.0000   Max.   :100.00   Max.   :100.00  
##                  NA's   :1        NA's   :1        NA's   :3       
##     meal.cal        wt.loss        
##  Min.   :  96.0   Min.   :-24.000  
##  1st Qu.: 635.0   1st Qu.:  0.000  
##  Median : 975.0   Median :  7.000  
##  Mean   : 928.8   Mean   :  9.832  
##  3rd Qu.:1150.0   3rd Qu.: 15.750  
##  Max.   :2600.0   Max.   : 68.000  
##  NA's   :47       NA's   :14       
```

Fitting the model

Try whether you get the same result if you change the data type for `sex` into factor. How to interpret the results when using `sex` as a continuous variable and when using `sex` as a factor?

```
# lung$sex <- as.factor(lung$sex)
coxfit = coxph(Surv(time, status) ~ sex, data = lung)
coxfit
```

```
## Call:
## coxph(formula = Surv(time, status) ~ sex, data = lung)
##
##        coef exp(coef) se(coef)      z       p
## sex -0.5310    0.5880   0.1672 -3.176 0.00149
##
## Likelihood ratio test=10.63  on 1 df, p=0.001111
## n= 228, number of events= 165
```

```r
ss = summary(coxfit)
ss
```

```
## Call:
## coxph(formula = Surv(time, status) ~ sex, data = lung)
##
##   n= 228, number of events= 165
##
##        coef exp(coef) se(coef)      z Pr(>|z|)
## sex -0.5310    0.5880   0.1672 -3.176  0.00149 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##     exp(coef) exp(-coef) lower .95 upper .95
## sex     0.588      1.701    0.4237     0.816
##
## Concordance= 0.579  (se = 0.021 )
## Likelihood ratio test= 10.63  on 1 df,   p=0.001
## Wald test            = 10.09  on 1 df,   p=0.001
## Score (logrank) test = 10.33  on 1 df,   p=0.001
```

```r
# Extract the coefficient
coef(coxfit)
```

```
##        sex
## -0.5310235
```

```r
# or
coxfit$coefficients
```

```
##        sex
## -0.5310235
```

```r
# Extract the coefficients and p-value from Wald test
coef(ss)
```

```
##           coef exp(coef)  se(coef)         z    Pr(>|z|)
## sex -0.5310235 0.5880028 0.1671786 -3.176385 0.001491229
```

```r
# or
ss$coefficients
```

```
##           coef exp(coef)  se(coef)         z    Pr(>|z|)
## sex -0.5310235 0.5880028 0.1671786 -3.176385 0.001491229
```

Prediction

Note that when using `basehaz()` to get the estimator of cumulative baseline hazard, one should use the argument `center = FALSE` in order to get the estimate for the cumulative baseline hazard function $\Lambda_0(t)$ in the slides. See the example below.

When using `center = FALSE` in `basehaz()`, one need to further center the covariate values when using the formula:

$$\hat{S}(t|Z) = e^{-\hat{\Lambda}_0(t)e^{\hat{\beta}^\top Z}}.$$

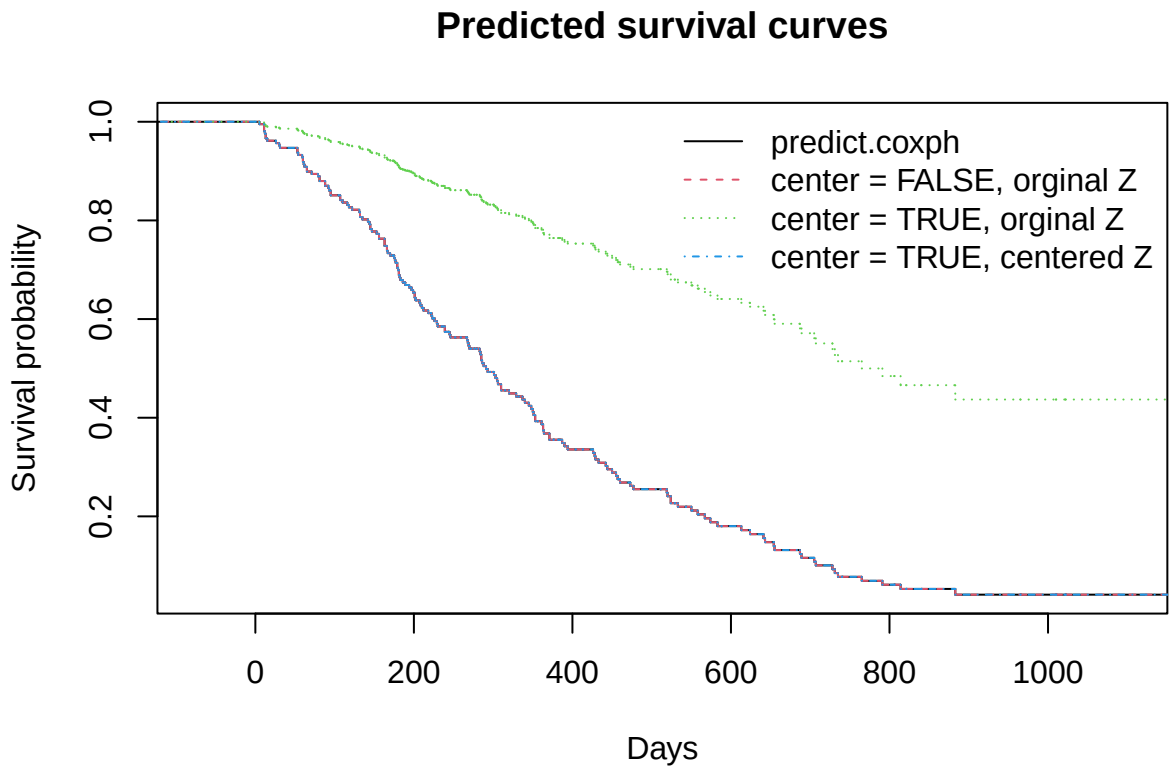
See the example below.

```
coxfit = coxph(Surv(time, status) ~ ph.karno, data = lung)
coxfit
```

```
## Call:
## coxph(formula = Surv(time, status) ~ ph.karno, data = lung)
##
##               coef exp(coef)  se(coef)     z       p
## ph.karno -0.016448  0.983686  0.005854 -2.81 0.00496
##
## Likelihood ratio test=7.56  on 1 df, p=0.005966
## n= 227, number of events= 164
##    (1 observation deleted due to missingness)
```

```
ph.karno.new = 75

## Compute the baseline hazard
baseh = basehaz(coxfit, center = FALSE)
beta = coef(coxfit)

baseh_centerT = basehaz(coxfit, center = TRUE)


newdat = data.frame(time = baseh$time,
                    status = 1,
                    ph.karno = ph.karno.new)

pred = predict(coxfit, newdata = newdat, type = "survival", se.fit = TRUE)
pred_surv  = pred$fit
# To see more usage of the function `prediction()` for `coxph` object, try the following.
# ?predict.coxph

# Another method - Compute the estimates from `beta` and `baseh`
# sex = 1 -> to the baseline survival, since sex = 1 is the reference group
pred2_surv_stepf <- stepfun(baseh$time, c(1,exp(-baseh$hazard*exp(beta*ph.karno.new))))
pred3_surv_stepf <- stepfun(baseh_centerT$time, c(1, exp(-baseh_centerT$hazard*exp(beta*ph.karno.new))))
pred4_surv_stepf <- stepfun(baseh_centerT$time, c(1, exp(-baseh_centerT$hazard*exp( beta*(ph.karno.new -

## Plot out the step function for the estimated survival curve
pred_surv_stepf <- stepfun(newdat$time, c(1,pred_surv))
plot(pred_surv_stepf, do.points = FALSE, col = 1, lty = 1,
     main = "Predicted survival curves",
     xlab = "Days", ylab = "Survival probability")
# ?plot.stepfun  # for more usage of plotting the `stepfun` object
plot(pred2_surv_stepf, do.points = FALSE, col = 2, lty = 2, add = TRUE)
plot(pred3_surv_stepf, do.points = FALSE, col = 3, lty = 3, add = TRUE)
plot(pred4_surv_stepf, do.points = FALSE, col = 4, lty = 4, add = TRUE)
```

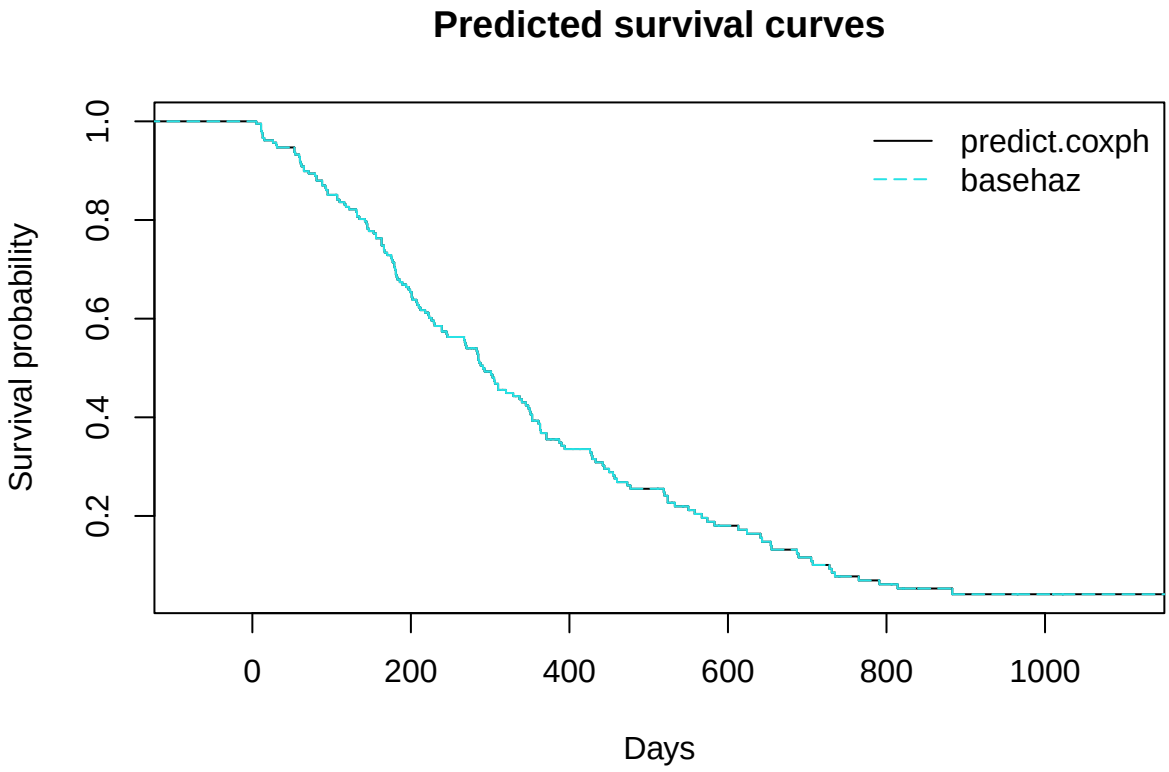
```
legend("topright",
       legend = c("predict.coxph", "center = FALSE, orginal Z", "center = TRUE, orginal Z", "center = TR
       col = 1:4, lty = 1:4,
       bty = "n")
```



See the R documentation for `predict.coxph()` for details about how to do prediction.

Another way to do prediction is using the `basehaz(.., newdata)`.

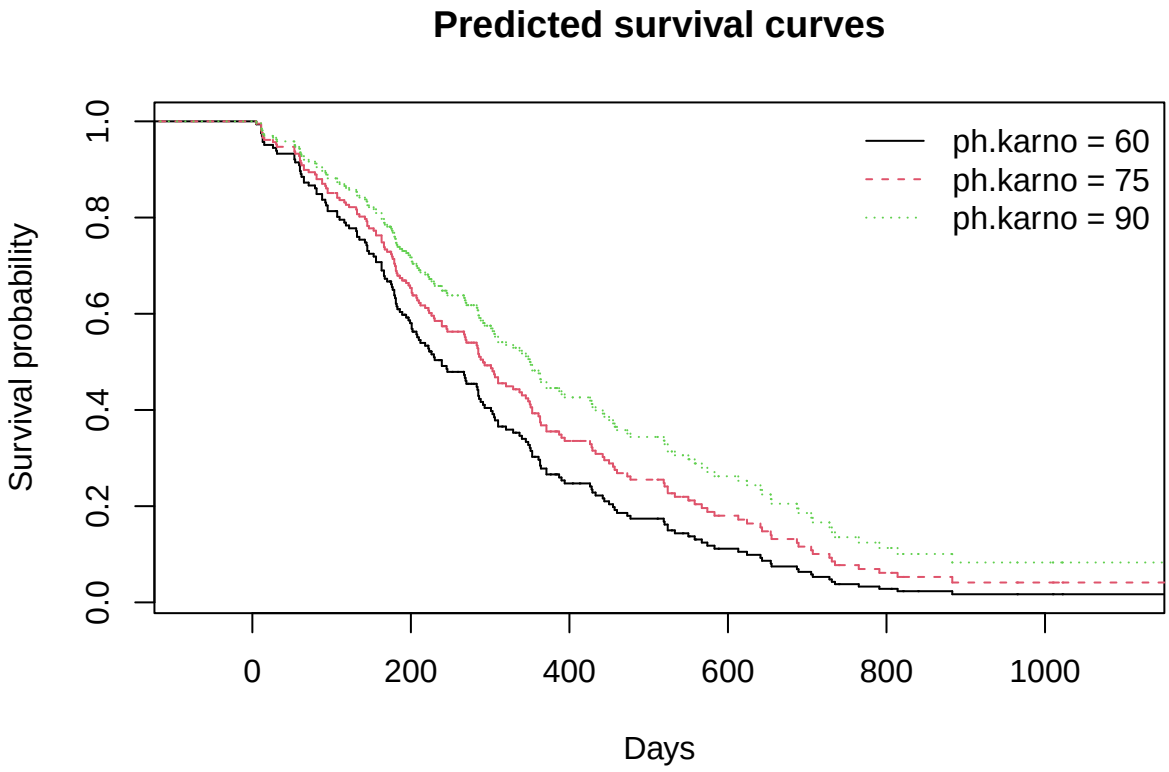
```
newdat = data.frame(ph.karno = ph.karno.new)
cumhazard = basehaz(coxfit, newdata = newdat)

pred5_surv_stepf <- stepfun(baseh$time, c(1, exp(-cumhazard$hazard)))

plot(pred_surv_stepf, do.points = FALSE, col = 1, lty = 1,
     main = "Predicted survival curves",
     xlab = "Days", ylab = "Survival probability")
plot(pred5_surv_stepf, do.points = FALSE, col = 5, lty = 5, add = TRUE)
legend("topright",
       legend = c("predict.coxph", "basehaz"),
       col = c(1,5), lty = c(1,5),
       bty = "n")
```




```
# To predict multiple survival curves at once
ph.karno.new = c(60, 75, 90)
newdat = data.frame(ph.karno = ph.karno.new, sex = c(1,2,1))
cumhazard = basehaz(coxfit, newdata = newdat)
colnames(cumhazard)
```

```
## [1] "hazard.1" "hazard.2" "hazard.3" "time"
```

```
surv_stepf <- stepfun(cumhazard$time, c(1, exp(-cumhazard[,1])))
plot(surv_stepf, do.points = FALSE, col = 1, lty = 1,
     main = "Predicted survival curves",
     xlab = "Days", ylab = "Survival probability")
k = nrow(newdat)
for(i in 2:k){
    surv_stepf <- stepfun(cumhazard$time, c(1, exp(-cumhazard[,i])))
    plot(surv_stepf, do.points = FALSE, col = i, lty = i, add = TRUE)
}
legend("topright",
       legend = paste("ph.karno =",  ph.karno.new),
       col = 1:k, lty = 1:k,
       bty = "n")
```



Cox PH Model diagnostics

We will use `pbc` data from `survival` R package as illustration.
Consider the five explanatory variables that are found to be important by Fleming & Harrington (1991): age, edema, bilirubin, albumin, protime. As a illustration, we also include an additional binary variable "sex".

```
library(survival)
# library(timereg)

data(pbc, package="survival")
summary(pbc)
```

```
##        id              time          status            trt
##  Min.   :  1.0   Min.   :  41   Min.   :0.0000   Min.   :1.000
##  1st Qu.:105.2   1st Qu.:1093   1st Qu.:0.0000   1st Qu.:1.000
##  Median :209.5   Median :1730   Median :0.0000   Median :1.000
##  Mean   :209.5   Mean   :1918   Mean   :0.8301   Mean   :1.494
##  3rd Qu.:313.8   3rd Qu.:2614   3rd Qu.:2.0000   3rd Qu.:2.000
##  Max.   :418.0   Max.   :4795   Max.   :2.0000   Max.   :2.000
##                                                  NA's   :106
##       age        sex        ascites            hepato          spiders
##  Min.   :26.28   m: 44   Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:42.83   f:374   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :51.00           Median :0.00000   Median :1.0000   Median :0.0000
##  Mean   :50.74           Mean   :0.07692   Mean   :0.5128   Mean   :0.2885
##  3rd Qu.:58.24           3rd Qu.:0.00000   3rd Qu.:1.0000   3rd Qu.:1.0000
```

```
##  Max.   :78.44           Max.   :1.00000   Max.   :1.0000   Max.   :1.0000
##                          NA's   :106       NA's   :106      NA's   :106
##      edema             bili             chol           albumin
##  Min.   :0.0000   Min.   : 0.300   Min.   : 120.0   Min.   :1.960
##  1st Qu.:0.0000   1st Qu.: 0.800   1st Qu.: 249.5   1st Qu.:3.243
##  Median :0.0000   Median : 1.400   Median : 309.5   Median :3.530
##  Mean   :0.1005   Mean   : 3.221   Mean   : 369.5   Mean   :3.497
##  3rd Qu.:0.0000   3rd Qu.: 3.400   3rd Qu.: 400.0   3rd Qu.:3.770
##  Max.   :1.0000   Max.   :28.000   Max.   :1775.0   Max.   :4.640
##                                    NA's   :134
##      copper          alk.phos            ast              trig
##  Min.   :  4.00   Min.   :  289.0   Min.   : 26.35   Min.   : 33.00
##  1st Qu.: 41.25   1st Qu.:  871.5   1st Qu.: 80.60   1st Qu.: 84.25
##  Median : 73.00   Median : 1259.0   Median :114.70   Median :108.00
##  Mean   : 97.65   Mean   : 1982.7   Mean   :122.56   Mean   :124.70
##  3rd Qu.:123.00   3rd Qu.: 1980.0   3rd Qu.:151.90   3rd Qu.:151.00
##  Max.   :588.00   Max.   :13862.4   Max.   :457.25   Max.   :598.00
##  NA's   :108      NA's   :106       NA's   :106      NA's   :136
##     platelet        protime          stage
##  Min.   : 62.0   Min.   : 9.00   Min.   :1.000
##  1st Qu.:188.5   1st Qu.:10.00   1st Qu.:2.000
##  Median :251.0   Median :10.60   Median :3.000
##  Mean   :257.0   Mean   :10.73   Mean   :3.024
##  3rd Qu.:318.0   3rd Qu.:11.10   3rd Qu.:4.000
##  Max.   :721.0   Max.   :18.00   Max.   :4.000
##  NA's   :11      NA's   :2       NA's   :6
```

```
covs = c("sex", "age", "edema", "bili", "albumin", "protime")
dat = pbc[, c("id", "time", "status", covs)]

# Remove subjects with missingness
dat = dat[complete.cases(dat), ]
dim(dat)
```

```
## [1] 416   9
```

```
# View death as event and others as censoring
dat$status <- as.numeric(dat$status == 2)
summary(as.factor(dat$status))
```

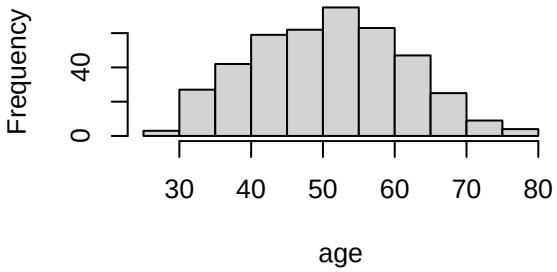
```
##   0   1
## 256 160
```

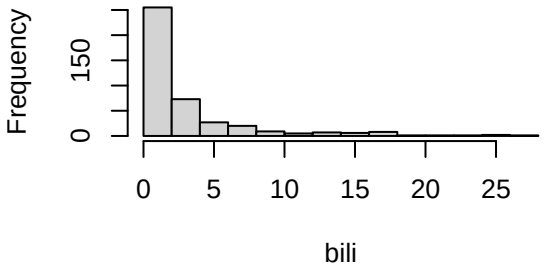
```
# Change time to years
dat$time = dat$time/365.25


## Summary statistics
summary(dat)
```

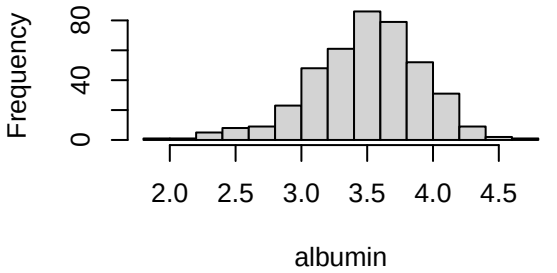
```
##        id              time             status       sex           age
##  Min.   :  1.0   Min.   : 0.1123   Min.   :0.0000   m: 44   Min.   :26.28
```

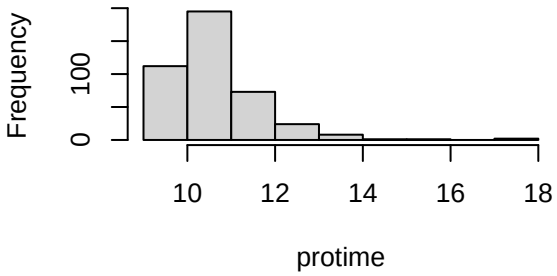
```
##  1st Qu.:104.8   1st Qu.: 2.9959   1st Qu.:0.0000   f:372   1st Qu.:42.78
##  Median :208.5   Median : 4.7365   Median :0.0000           Median :51.00
##  Mean   :208.8   Mean   : 5.2589   Mean   :0.3846           Mean   :50.76
##  3rd Qu.:312.2   3rd Qu.: 7.1656   3rd Qu.:1.0000           3rd Qu.:58.27
##  Max.   :418.0   Max.   :13.1280   Max.   :1.0000           Max.   :78.44
##      edema            bili           albumin         protime
##  Min.   :0.000   Min.   : 0.300   Min.   :1.960   Min.   : 9.00
##  1st Qu.:0.000   1st Qu.: 0.800   1st Qu.:3.257   1st Qu.:10.00
##  Median :0.000   Median : 1.400   Median :3.535   Median :10.60
##  Mean   :0.101   Mean   : 3.201   Mean   :3.501   Mean   :10.73
##  3rd Qu.:0.000   3rd Qu.: 3.400   3rd Qu.:3.772   3rd Qu.:11.10
##  Max.   :1.000   Max.   :28.000   Max.   :4.640   Max.   :18.00
```

```
# Histogram for the covariates
# covs.hist = c("age", "albumin", "protime")
covs.hist = c("age", "bili", "albumin", "protime")
par(mfrow=c(2,2))
for(i in 1:length(covs.hist)){
    hist(dat[,covs.hist[i]], xlab = covs.hist[i], main = "")
}
```







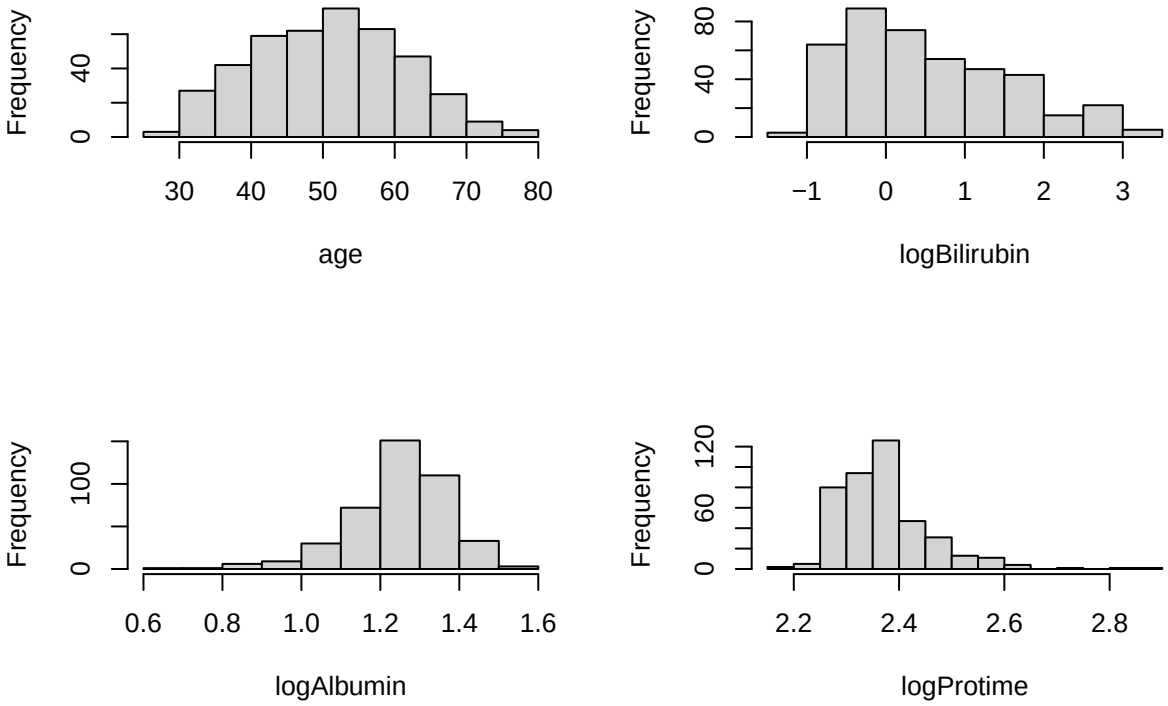



```
dat$logAlbumin = log(dat$albumin)
dat$logProtime = log(dat$protime)
dat$logBilirubin = log(dat$bili)

summary(dat)
```

```
##        id              time             status       sex          age
##  Min.   :  1.0   Min.   : 0.1123   Min.   :0.0000   m: 44   Min.   :26.28
##  1st Qu.:104.8   1st Qu.: 2.9959   1st Qu.:0.0000   f:372   1st Qu.:42.78
```

```
##  Median :208.5   Median : 4.7365   Median :0.0000           Median :51.00
##  Mean   :208.8   Mean   : 5.2589   Mean   :0.3846           Mean   :50.76
##  3rd Qu.:312.2   3rd Qu.: 7.1656   3rd Qu.:1.0000           3rd Qu.:58.27
##  Max.   :418.0   Max.   :13.1280   Max.   :1.0000           Max.   :78.44
##      edema             bili           albumin         protime
##  Min.   :0.000   Min.   : 0.300   Min.   :1.960   Min.   : 9.00
##  1st Qu.:0.000   1st Qu.: 0.800   1st Qu.:3.257   1st Qu.:10.00
##  Median :0.000   Median : 1.400   Median :3.535   Median :10.60
##  Mean   :0.101   Mean   : 3.201   Mean   :3.501   Mean   :10.73
##  3rd Qu.:0.000   3rd Qu.: 3.400   3rd Qu.:3.772   3rd Qu.:11.10
##  Max.   :1.000   Max.   :28.000   Max.   :4.640   Max.   :18.00
##    logAlbumin       logProtime     logBilirubin
##  Min.   :0.6729   Min.   :2.197   Min.   :-1.2040
##  1st Qu.:1.1810   1st Qu.:2.303   1st Qu.:-0.2231
##  Median :1.2627   Median :2.361   Median : 0.3365
##  Mean   :1.2451   Mean   :2.369   Mean   : 0.5685
##  3rd Qu.:1.3277   3rd Qu.:2.407   3rd Qu.: 1.2238
##  Max.   :1.5347   Max.   :2.890   Max.   : 3.3322
```

```r
covs.hist = c("age", "logBilirubin", "logAlbumin", "logProtime")
par(mfrow=c(2,2))
for(i in 1:length(covs.hist)){
    hist(dat[,covs.hist[i]], xlab = covs.hist[i], main = "")
}
```



Fit univariate Cox models

```r
# Fit a univariate model for sex
fit <-  coxph(Surv(time,status)~sex,
```

```
              data = dat)
fit
```

```
## Call:
## coxph(formula = Surv(time, status) ~ sex, data = dat)
##
##         coef exp(coef) se(coef)      z      p
## sexf -0.3850    0.6804   0.2223 -1.732 0.0832
##
## Likelihood ratio test=2.74  on 1 df, p=0.09762
## n= 416, number of events= 160
```

```
summary(fit)
```

```
## Call:
## coxph(formula = Surv(time, status) ~ sex, data = dat)
##
##   n= 416, number of events= 160
##
##         coef exp(coef) se(coef)      z Pr(>|z|)
## sexf -0.3850    0.6804   0.2223 -1.732   0.0832 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      exp(coef) exp(-coef) lower .95 upper .95
## sexf    0.6804       1.47    0.4401     1.052
##
## Concordance= 0.518  (se = 0.014 )
## Likelihood ratio test= 2.74  on 1 df,   p=0.1
## Wald test            = 3  on 1 df,   p=0.08
## Score (logrank) test = 3.04  on 1 df,   p=0.08
```

```
# Fit a univariate model for age
fit<- coxph(Surv(time,status)~age,
            data = dat)
fit
```

```
## Call:
## coxph(formula = Surv(time, status) ~ age, data = dat)
##
##        coef exp(coef) se(coef)     z        p
## age 0.03936   1.04015  0.00787 5.002 5.68e-07
##
## Likelihood ratio test=25.28  on 1 df, p=4.946e-07
## n= 416, number of events= 160
```

```
summary(fit)
```

```
## Call:
## coxph(formula = Surv(time, status) ~ age, data = dat)
##
```

```
##   n= 416, number of events= 160
##
##        coef exp(coef) se(coef)     z Pr(>|z|)
## age 0.03936   1.04015  0.00787 5.002 5.68e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##     exp(coef) exp(-coef) lower .95 upper .95
## age      1.04     0.9614     1.024     1.056
##
## Concordance= 0.617  (se = 0.023 )
## Likelihood ratio test= 25.28  on 1 df,   p=5e-07
## Wald test            = 25.02  on 1 df,   p=6e-07
## Score (logrank) test = 25.38  on 1 df,   p=5e-07
```

Fit a multivariate Cox-PH model

```
fit_multi <- coxph(Surv(time,status)~sex+age+edema+logBilirubin+logAlbumin+logProtime,
             data = dat)
fit_multi
```

```
## Call:
## coxph(formula = Surv(time, status) ~ sex + age + edema + logBilirubin +
##     logAlbumin + logProtime, data = dat)
##
##                   coef exp(coef)  se(coef)      z        p
## sexf         -0.217957  0.804160  0.232884 -0.936 0.349324
## age           0.037844  1.038569  0.007861  4.814 1.48e-06
## edema         0.929118  2.532274  0.273416  3.398 0.000678
## logBilirubin  0.861819  2.367462  0.083154 10.364  < 2e-16
## logAlbumin   -2.534959  0.079265  0.652813 -3.883 0.000103
## logProtime    2.399128 11.013572  0.771682  3.109 0.001877
##
## Likelihood ratio test=231.8  on 6 df, p=< 2.2e-16
## n= 416, number of events= 160
```

```
summary(fit_multi)
```

```
## Call:
## coxph(formula = Surv(time, status) ~ sex + age + edema + logBilirubin +
##     logAlbumin + logProtime, data = dat)
##
##   n= 416, number of events= 160
##
##                   coef exp(coef)  se(coef)      z Pr(>|z|)
## sexf         -0.217957  0.804160  0.232884 -0.936 0.349324
## age           0.037844  1.038569  0.007861  4.814 1.48e-06 ***
## edema         0.929118  2.532274  0.273416  3.398 0.000678 ***
## logBilirubin  0.861819  2.367462  0.083154 10.364  < 2e-16 ***
## logAlbumin   -2.534959  0.079265  0.652813 -3.883 0.000103 ***
## logProtime    2.399128 11.013572  0.771682  3.109 0.001877 **
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##              exp(coef) exp(-coef) lower .95 upper .95
## sexf           0.80416     1.2435   0.50946    1.2693
## age            1.03857     0.9629   1.02269    1.0547
## edema          2.53227     0.3949   1.48176    4.3276
## logBilirubin   2.36746     0.4224   2.01142    2.7865
## logAlbumin     0.07926    12.6159   0.02205    0.2849
## logProtime    11.01357     0.0908   2.42701   49.9788
##
## Concordance= 0.834  (se = 0.017 )
## Likelihood ratio test= 231.8  on 6 df,   p=<2e-16
## Wald test            = 233.9  on 6 df,   p=<2e-16
## Score (logrank) test = 301.9  on 6 df,   p=<2e-16
```

Compute the R-square meansure based on the partial likelihood ratio statistic under the Cox model

```
library(CoxR2)
coxr2fit = coxr2(fit_multi)
coxr2fit$rsq
```

```
##       rsq
## 0.7651574
```

Check the PH assumption using cumulative martingale residuals

Here we refit the Cox model with `timereg:cox.aalen()`, which has cox regression as a special case. This package gives us easy implementation for diagnostic plots and tests of proportional hazards assumption using cumulative martingale residuals.

```
library(timereg)
fit_multi2 <- cox.aalen(Surv(time,status)~prop(sex)+prop(age)+prop(edema)+
                            prop(logBilirubin)+prop(logAlbumin)+prop(logProtime),
                        data = dat)
fit_multi2
```

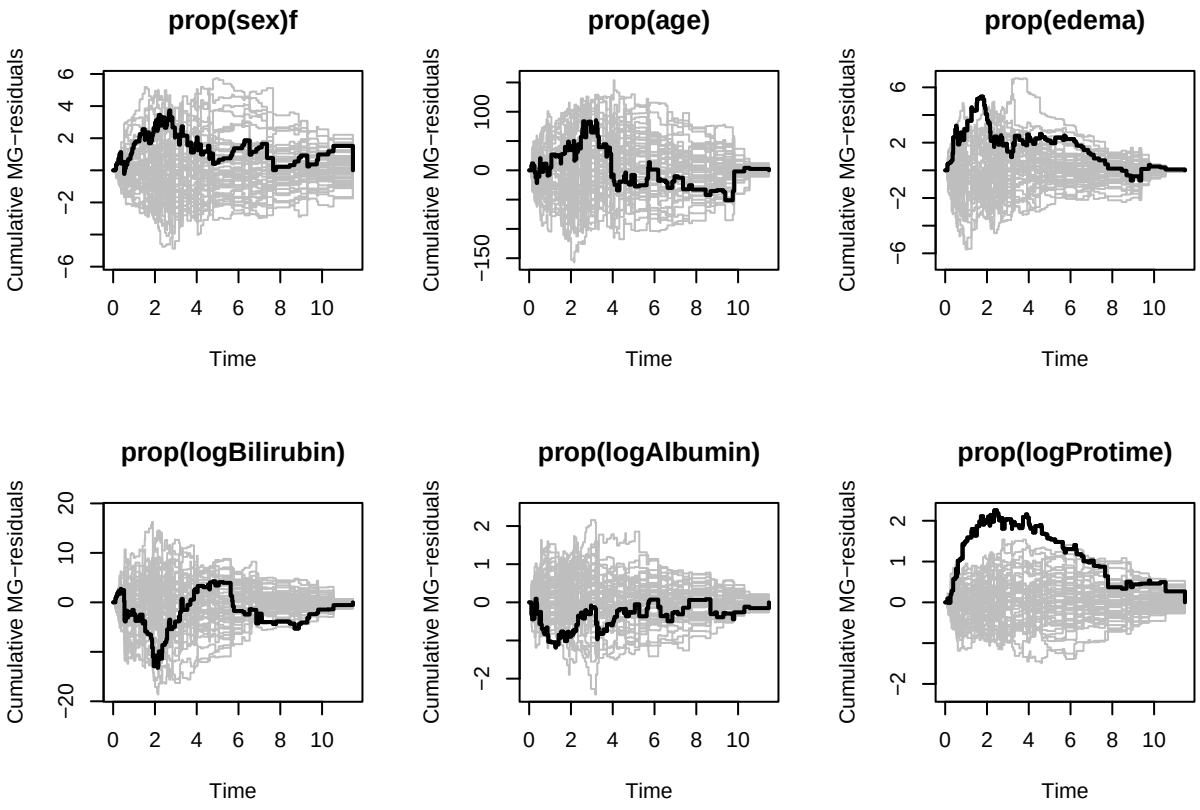
```
## Cox-Aalen Model
##
## Test for Aalen terms
## Test not computed, sim=0
##
## Proportional Cox terms :
##                       Coef.      SE Robust SE D2log(L)^-1      z    P-val
## prop(sex)f         -0.2180 0.23800   0.24700     0.23300 -0.882 3.78e-01
## prop(age)           0.0379 0.00726   0.00886     0.00786  4.270 1.92e-05
## prop(edema)         0.9300 0.28300   0.30700     0.27300  3.030 2.44e-03
## prop(logBilirubin)  0.8620 0.07730   0.08680     0.08320  9.940 0.00e+00
## prop(logAlbumin)   -2.5400 0.66900   0.61400     0.65300 -4.140 3.42e-05
```

```
## prop(logProtime)    2.4000 0.62900   0.93800     0.77100  2.560 1.05e-02
##                     lower2.5% upper97.5%
## prop(sex)f            -0.6840     0.2480
## prop(age)              0.0237     0.0521
## prop(edema)            0.3750     1.4800
## prop(logBilirubin)     0.7100     1.0100
## prop(logAlbumin)      -3.8500    -1.2300
## prop(logProtime)       1.1700     3.6300
## Test of Proportionality
##                     sup|  hat U(t) | p-value H_0
## prop(sex)f                   3.73        0.326
## prop(age)                   85.60        0.572
## prop(edema)                  5.33        0.030
## prop(logBilirubin)          13.20        0.120
## prop(logAlbumin)             1.18        0.552
## prop(logProtime)             2.26        0.002
```

```
# p-values from test of PH assumption
summary(fit_multi2)
```

```
## Cox-Aalen Model
##
## Test for Aalen terms
## Test not computed, sim=0
##
## Proportional Cox terms :
##                       Coef.      SE Robust SE D2log(L)^-1      z    P-val
## prop(sex)f         -0.2180 0.23800   0.24700     0.23300 -0.882 3.78e-01
## prop(age)           0.0379 0.00726   0.00886     0.00786  4.270 1.92e-05
## prop(edema)         0.9300 0.28300   0.30700     0.27300  3.030 2.44e-03
## prop(logBilirubin)  0.8620 0.07730   0.08680     0.08320  9.940 0.00e+00
## prop(logAlbumin)   -2.5400 0.66900   0.61400     0.65300 -4.140 3.42e-05
## prop(logProtime)    2.4000 0.62900   0.93800     0.77100  2.560 1.05e-02
##                     lower2.5% upper97.5%
## prop(sex)f            -0.6840     0.2480
## prop(age)              0.0237     0.0521
## prop(edema)            0.3750     1.4800
## prop(logBilirubin)     0.7100     1.0100
## prop(logAlbumin)      -3.8500    -1.2300
## prop(logProtime)       1.1700     3.6300
## Test of Proportionality
##                     sup|  hat U(t) | p-value H_0
## prop(sex)f                   3.73        0.326
## prop(age)                   85.60        0.572
## prop(edema)                  5.33        0.030
## prop(logBilirubin)          13.20        0.120
## prop(logAlbumin)             1.18        0.552
## prop(logProtime)             2.26        0.002
```

```
par(mfrow=c(2,3))
plot(fit_multi2, score = TRUE)
```

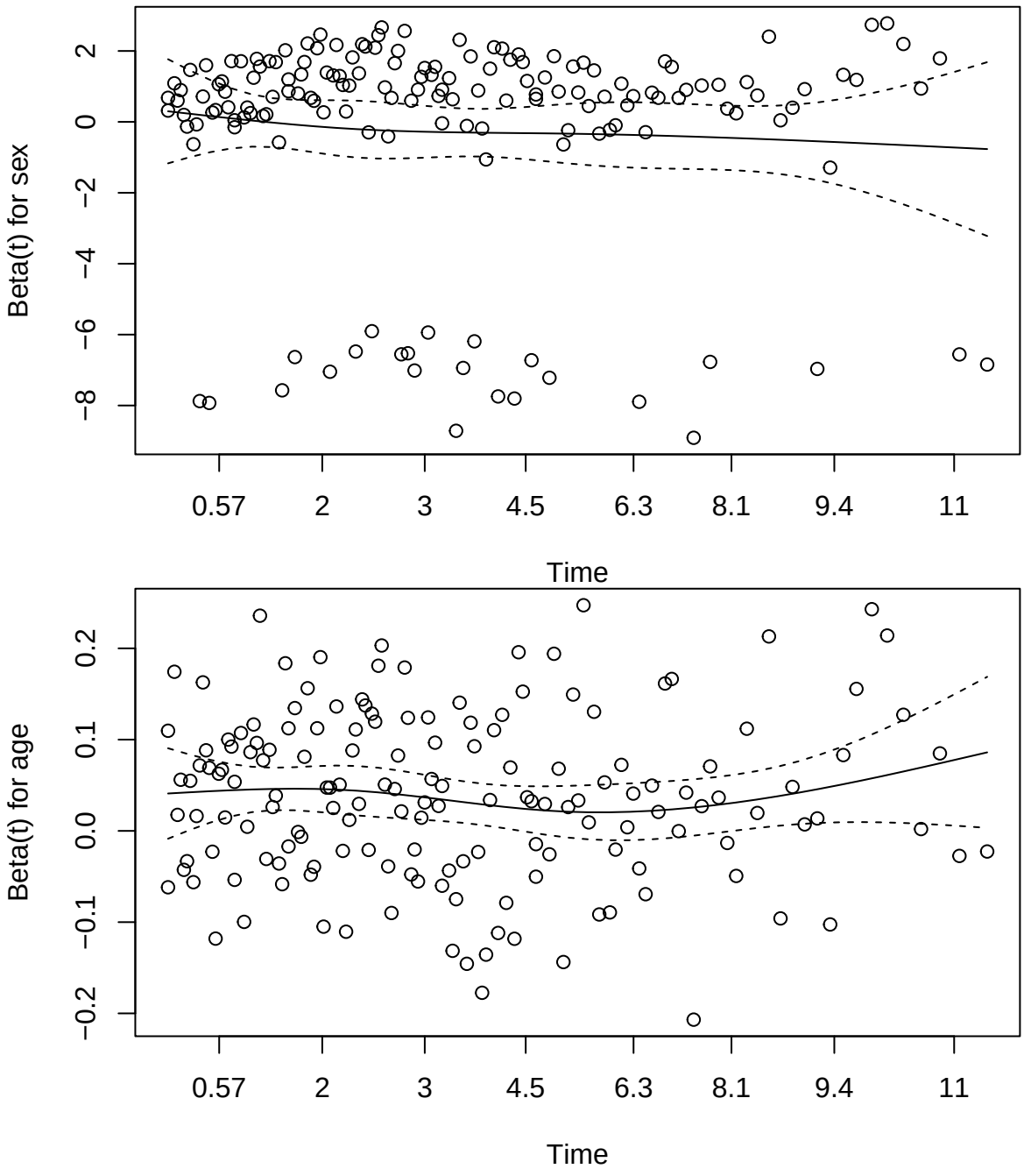


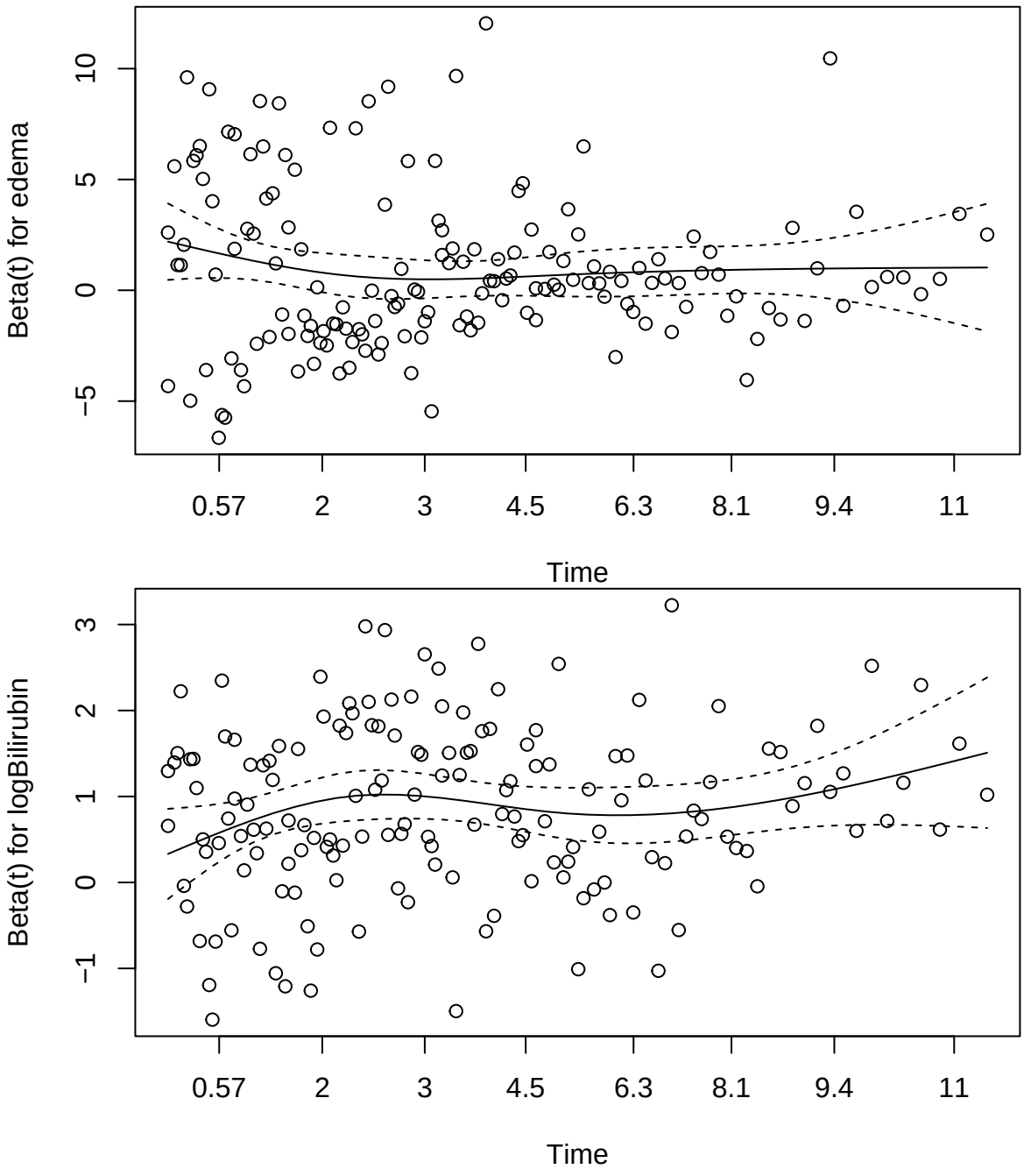
Check the PH assumption using Schoenfeld residuals

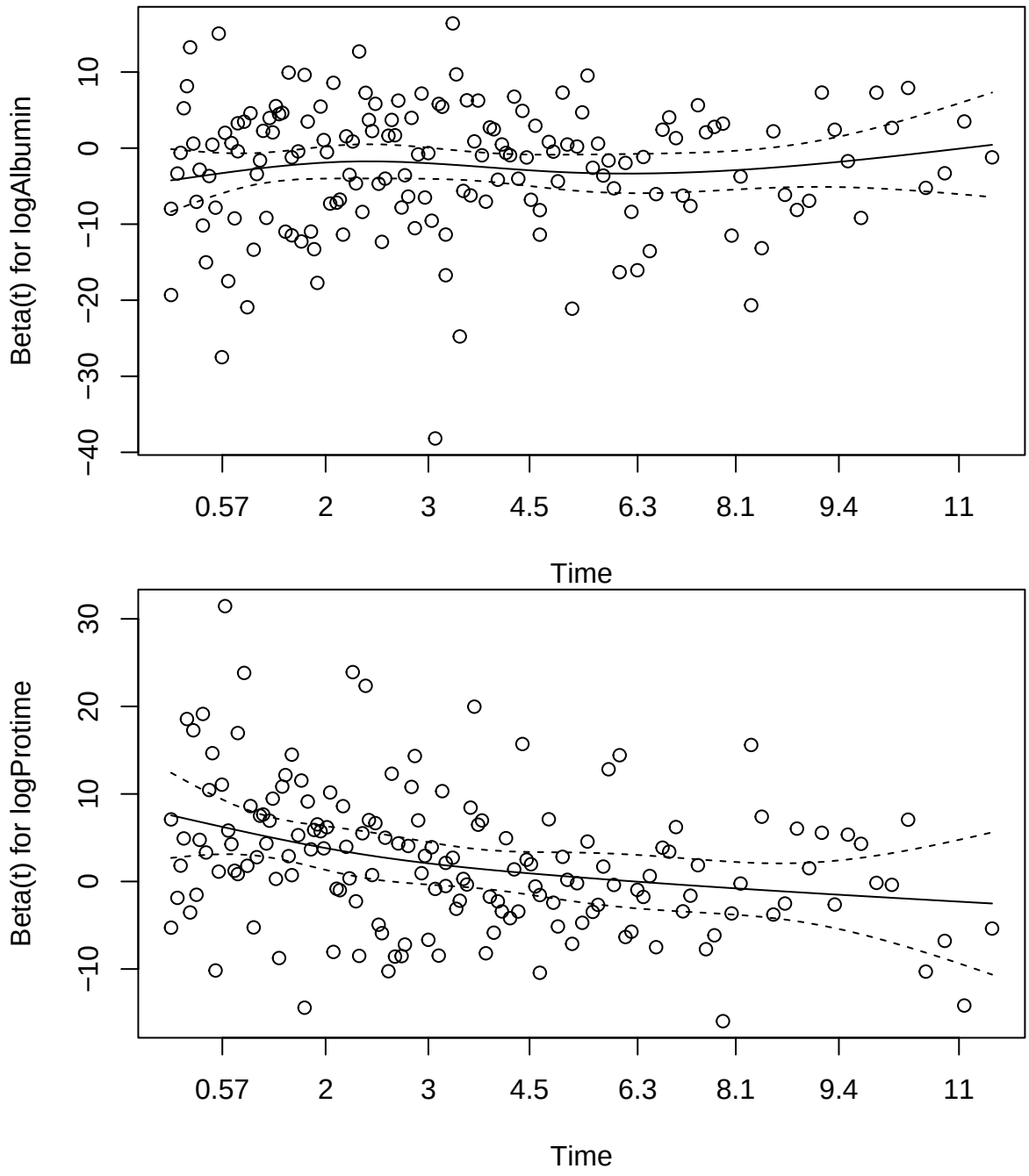
```
fit_cox = coxph(Surv(time,status)~sex+age+edema+logBilirubin+logAlbumin+logProtime,
                data = dat)
test.ph = cox.zph(fit_cox)
print(test.ph)
```

```
##                  chisq df      p
## sex           1.58e+00  1 0.2084
## age           8.12e-04  1 0.9773
## edema         2.99e+00  1 0.0840
## logBilirubin 9.51e-01  1 0.3295
## logAlbumin    1.11e+00  1 0.2927
## logProtime    9.27e+00  1 0.0023
## GLOBAL        1.40e+01  6 0.0299
```

```
plot(test.ph)
```


Beta(t) for sex
2
0
−2
−4
−6
−8
0.57
2
3
4.5
6.3
8.1
9.4
11
Time
Beta(t) for age
0.2
0.1
0.0
−0.1
−0.2
0.57
2
3
4.5
6.3
8.1
9.4
11
Time


Beta(t) for edema
10
5
0
−5
0.57
2
3
4.5
6.3
8.1
9.4
11
Time
Beta(t) for logBilirubin
3
2
1
0
−1
0.57
2
3
4.5
6.3
8.1
9.4
11
Time



We can also do a graphical diagnostic using the function `ggcoxzph()` (in the `survminer` R package), which produces, for each covariate, graphs of the scaled Schoenfeld residuals against the time.

```
library(survminer)
```

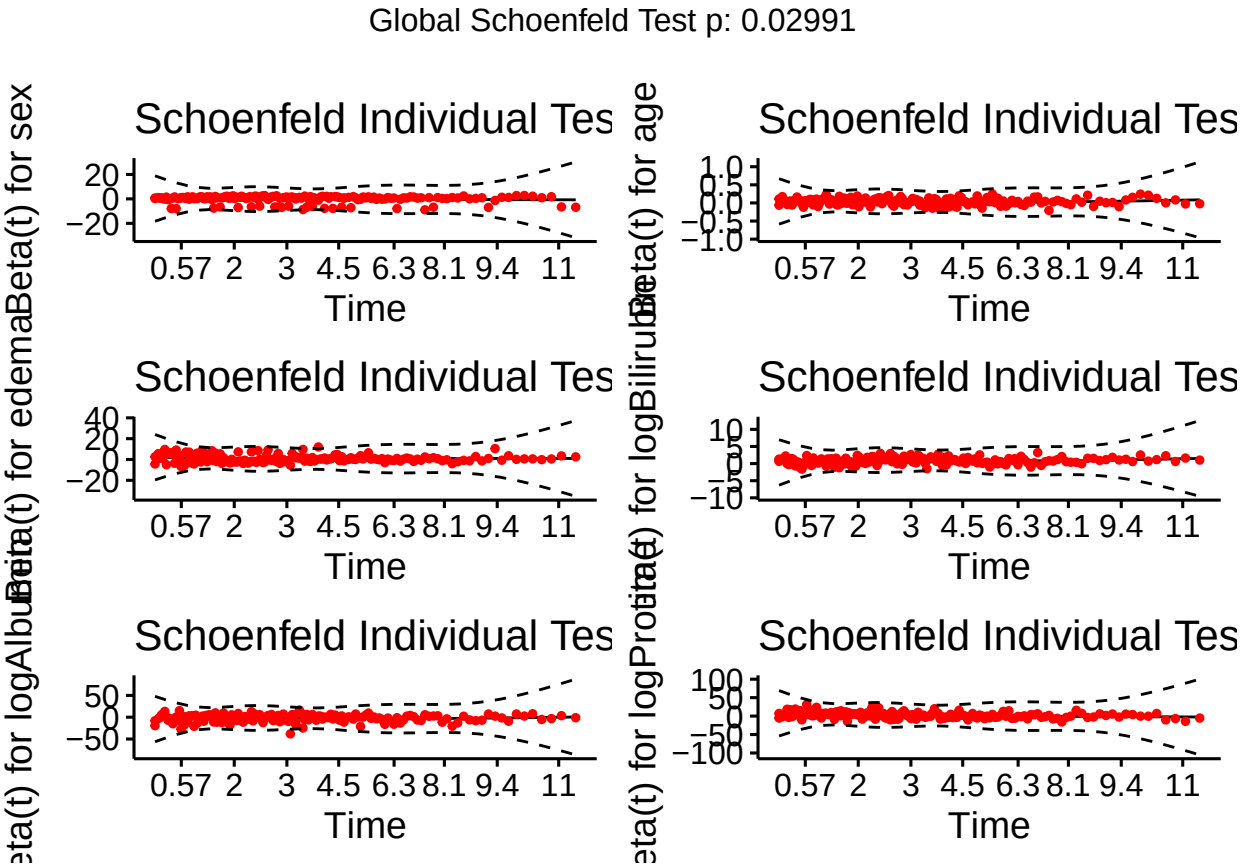
```
## Loading required package: ggplot2
```

```
## Loading required package: ggpubr
```

```
##
## Attaching package: 'survminer'
```

```
## The following object is masked from 'package:survival':
##
##     myeloma
```

```
ggcoxzph(test.ph)
```



Remarks

- Comparing the test using cumulative martingale residuals and the one using Schoenfeld residuals, we see that the test using Schoenfeld residuals seems to be less sensitive to violations of proportional hazards assumption compared with the test using cumulative martingale residuals.

Stratified Cox model

Here we fit a stratified Cox-PH model with strata defined by edema. Note that edema only takes 3 values: 0, 0.5, 1.

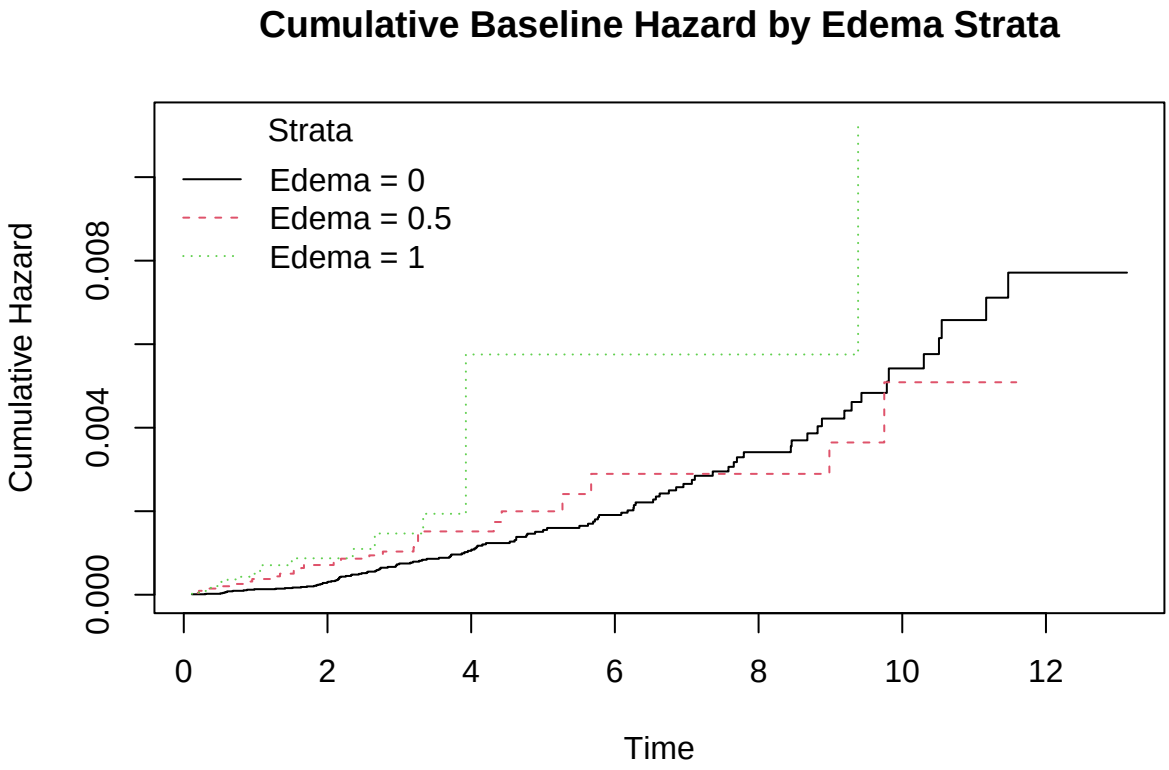
```
summary(as.factor(dat$edema))
```

```
##   0 0.5   1
## 352  44  20
```

```
fit_strata <- coxph(Surv(time,status)~sex+age+strata(edema)+logBilirubin+logAlbumin+logProtime,
                    data = dat)
fit_strata
```

```
## Call:
## coxph(formula = Surv(time, status) ~ sex + age + strata(edema) +
##     logBilirubin + logAlbumin + logProtime, data = dat)
##
##                    coef exp(coef)  se(coef)      z        p
## sexf         -0.174643  0.839757  0.234383 -0.745 0.456202
## age           0.039385  1.040171  0.008105  4.860 1.18e-06
## logBilirubin  0.847501  2.333806  0.083762 10.118  < 2e-16
## logAlbumin   -2.478805  0.083843  0.656328 -3.777 0.000159
## logProtime    2.444015 11.519195  0.769100  3.178 0.001484
##
## Likelihood ratio test=161.4  on 5 df, p=< 2.2e-16
## n= 416, number of events= 160
```

```
## The cumulative baseline hazards for the stratified Cox model
bhaz <- basehaz(fit_strata, centered = FALSE)  # Get baseline hazards

# Plot
par(mfrow=c(1,1))
plot(bhaz$time[bhaz$strata=="edema=0"], bhaz$hazard[bhaz$strata=="edema=0"],
     type = "s", col = 1, lty = 1,
     xlab = "Time", ylab = "Cumulative Hazard",
     ylim = range(bhaz$hazard),
     main = "Cumulative Baseline Hazard by Edema Strata")
lines(bhaz$time[bhaz$strata=="edema=0.5"], bhaz$hazard[bhaz$strata=="edema=0.5"],
      type = "s", col = 2, lty = 2)
lines(bhaz$time[bhaz$strata=="edema=1"], bhaz$hazard[bhaz$strata=="edema=1"],
      type = "s", col = 3, lty = 3)
legend("topleft", legend = c("Edema = 0", "Edema = 0.5", "Edema = 1"),
       col = 1:3, lty = 1:3, bty = "n", title = "Strata")
```



Here, `type = "s"` in `plot()` specifies that the line should be drawn as a step function. This is an easier way of doing plots for step function than the way before (first define a step function and then do the plot).

Cox model with piecewise constant time-varying coefficient

We fit a 3 piece piecewise constant model where the time intervals are chosen such that eahc interval has approximately the same number of events.

We use the approach mentioned in the lecture of fitting the equivalent Cox model with time varying covariates.

The following is one way of creating the pseudo data and fitting a piecewise model utilizing `survival::survSplit()` function.

```
qtl = quantile(dat$time[dat$status==1], c(1/3, 2/3))
qtl = as.numeric(qtl)
qtl
```

```
## [1] 2.130048 4.424367
```

```
dat_cut = survSplit(Surv(time,status)~., cut = qtl,
                    data = dat, episode = "tgroup")
# This gives the same pseudo data as in the approach we saw last time.

# Fit the piecewise Cox model
fit_3p = coxph(Surv(tstart,time,status) ~ strata(tgroup):(sex+age+edema+logBilirubin+logAlbumin+logProt
               data = dat_cut, id = id)
fit_3p
```

```
## Call:
## coxph(formula = Surv(tstart, time, status) ~ strata(tgroup):(sex +
##     age + edema + logBilirubin + logAlbumin + logProtime), data = dat_cut,
##     id = id)
##
##                                        coef exp(coef)  se(coef)      z
## strata(tgroup)tgroup=1:sexm          -0.42064   0.65663   0.49135 -0.856
## strata(tgroup)tgroup=2:sexm           0.42213   1.52520   0.37854  1.115
## strata(tgroup)tgroup=3:sexm           0.33526   1.39830   0.39403  0.851
## strata(tgroup)tgroup=1:sexf                NA        NA   0.00000     NA
## strata(tgroup)tgroup=2:sexf                NA        NA   0.00000     NA
## strata(tgroup)tgroup=3:sexf                NA        NA   0.00000     NA
## strata(tgroup)tgroup=1:age            0.04643   1.04752   0.01475  3.148
## strata(tgroup)tgroup=2:age            0.02994   1.03039   0.01338  2.238
## strata(tgroup)tgroup=3:age            0.03997   1.04078   0.01425  2.805
## strata(tgroup)tgroup=1:edema          1.17325   3.23247   0.37394  3.138
## strata(tgroup)tgroup=2:edema          0.81323   2.25517   0.53488  1.520
## strata(tgroup)tgroup=3:edema         -0.17274   0.84136   0.85562 -0.202
## strata(tgroup)tgroup=1:logBilirubin   0.51354   1.67119   0.13757  3.733
## strata(tgroup)tgroup=2:logBilirubin   1.17056   3.22381   0.14178  8.256
## strata(tgroup)tgroup=3:logBilirubin   0.79377   2.21172   0.16311  4.867
## strata(tgroup)tgroup=1:logAlbumin    -3.30175   0.03682   1.01924 -3.239
## strata(tgroup)tgroup=2:logAlbumin    -1.61244   0.19940   1.15735 -1.393
## strata(tgroup)tgroup=3:logAlbumin    -3.20214   0.04067   1.31988 -2.426
## strata(tgroup)tgroup=1:logProtime     6.09361 443.01814   1.23032  4.953
## strata(tgroup)tgroup=2:logProtime     2.21004   9.11609   1.60046  1.381
## strata(tgroup)tgroup=3:logProtime    -0.28010   0.75571   1.47765 -0.190
##                                          p
## strata(tgroup)tgroup=1:sexm         0.391954
## strata(tgroup)tgroup=2:sexm         0.264782
## strata(tgroup)tgroup=3:sexm         0.394859
## strata(tgroup)tgroup=1:sexf               NA
## strata(tgroup)tgroup=2:sexf               NA
## strata(tgroup)tgroup=3:sexf               NA
## strata(tgroup)tgroup=1:age          0.001641
## strata(tgroup)tgroup=2:age          0.025240
## strata(tgroup)tgroup=3:age          0.005031
## strata(tgroup)tgroup=1:edema        0.001704
## strata(tgroup)tgroup=2:edema        0.128412
## strata(tgroup)tgroup=3:edema        0.840005
## strata(tgroup)tgroup=1:logBilirubin 0.000189
## strata(tgroup)tgroup=2:logBilirubin  < 2e-16
## strata(tgroup)tgroup=3:logBilirubin 1.14e-06
## strata(tgroup)tgroup=1:logAlbumin   0.001198
## strata(tgroup)tgroup=2:logAlbumin   0.163553
## strata(tgroup)tgroup=3:logAlbumin   0.015262
## strata(tgroup)tgroup=1:logProtime   7.31e-07
## strata(tgroup)tgroup=2:logProtime   0.167316
## strata(tgroup)tgroup=3:logProtime   0.849655
##
## Likelihood ratio test=261  on 18 df, p=< 2.2e-16
## n= 996, number of events= 160
```

Below we create the pseudo data ourselves without using the `survival::survSplit()` function. We can see

that the results are the same as before, which is expected.

```
# pick cuts
qtl = quantile(dat$time[dat$status==1], c(1/3, 2/3))
qtl
```

```
## 33.33333% 66.66667%
##  2.130048  4.424367
```

```
qtl = as.numeric(qtl)
qtl
```

```
## [1] 2.130048 4.424367
```

```
covs = c("sex", "age", "edema", "logBilirubin", "logAlbumin", "logProtime")
dat$sex = as.numeric(dat$sex== "f")   # 1: female, 0: male

# Create pseudo data with time-varying covariates Z*I_{(cut1, cut2]}(t)
dat2 = dat[dat$time> qtl[1], ]
dat3 = dat[dat$time> qtl[2], ]
covs0_1 = matrix(0, nrow = nrow(dat), ncol = length(covs))
covs0_2 = matrix(0, nrow = nrow(dat2), ncol = length(covs))
covs0_3 = matrix(0, nrow = nrow(dat3), ncol = length(covs))

covsname1 = paste(covs, "1", sep = "")
covsname2 = paste(covs, "2", sep = "")
covsname3 = paste(covs, "3", sep = "")

dat_pseudo1 = cbind(id = dat[,"id"], start = 0, stop = pmin(dat$time, qtl[1]),
                    delta = as.numeric(dat$time<=qtl[1])*dat$status,
                    dat[, covs], covs0_1, covs0_1)
colnames(dat_pseudo1)[5:22] <- c(covsname1, covsname2, covsname3)

dat_pseudo2 = cbind(id = dat2[,"id"], start = qtl[1], stop = pmin(dat2$time, qtl[2]),
                     delta = as.numeric(dat2$time<=qtl[2])*dat2$status,
                     covs0_2, dat2[, covs], covs0_2)
colnames(dat_pseudo2)[5:22] <- c(covsname1, covsname2, covsname3)

dat_pseudo3 = cbind(id = dat3[,"id"], start = qtl[2], stop = dat3$time,
                     delta = dat3$status,
                     covs0_3, covs0_3, dat3[, covs])
colnames(dat_pseudo3)[5:22] <- c(covsname1, covsname2, covsname3)

dat_pseudo = rbind(dat_pseudo1, dat_pseudo2, dat_pseudo3)


# Change the order of roles - put the rows for same subjects together
order = order(as.numeric(dat_pseudo$id))
dat_psd = dat_pseudo[order, ]

# colnames(dat_psd)[5:22] <- c(covsname1, covsname2, covsname3)

dat_psd[1:10,]
```

```
##      id    start      stop delta sex1     age1 edema1 logBilirubin1 logAlbumin1
## 1     1 0.000000  1.095140     1    1 58.76523    1.0    2.67414865   0.9555114
## 2     2 0.000000  2.130048     0    1 56.44627    0.0    0.09531018   1.4206958
## 2100  2 2.130048  4.424367     0    0  0.00000    0.0    0.00000000   0.0000000
## 2102  2 4.424367 12.320329     0    0  0.00000    0.0    0.00000000   0.0000000
## 3     3 0.000000  2.130048     0    0 70.07255    0.5    0.33647224   1.2470323
## 359   3 2.130048  2.770705     1    0  0.00000    0.0    0.00000000   0.0000000
## 4     4 0.000000  2.130048     0    1 54.74059    0.5    0.58778666   0.9321641
## 419   4 2.130048  4.424367     0    0  0.00000    0.0    0.00000000   0.0000000
## 420   4 4.424367  5.270363     1    0  0.00000    0.0    0.00000000   0.0000000
## 5     5 0.000000  2.130048     0    1 38.10541    0.0    1.22377543   1.2612979
##      logProtime1 sex2     age2 edema2 logBilirubin2 logAlbumin2 logProtime2
## 1       2.501436    0  0.00000    0.0    0.00000000   0.0000000    0.000000
## 2       2.360854    0  0.00000    0.0    0.00000000   0.0000000    0.000000
## 2100    0.000000    1 56.44627    0.0    0.09531018   1.4206958    2.360854
## 2102    0.000000    0  0.00000    0.0    0.00000000   0.0000000    0.000000
## 3       2.484907    0  0.00000    0.0    0.00000000   0.0000000    0.000000
## 359     0.000000    0 70.07255    0.5    0.33647224   1.2470323    2.484907
## 4       2.332144    0  0.00000    0.0    0.00000000   0.0000000    0.000000
## 419     0.000000    1 54.74059    0.5    0.58778666   0.9321641    2.332144
## 420     0.000000    0  0.00000    0.0    0.00000000   0.0000000    0.000000
## 5       2.388763    0  0.00000    0.0    0.00000000   0.0000000    0.000000
##      sex3     age3 edema3 logBilirubin3 logAlbumin3 logProtime3
## 1       0  0.00000    0.0    0.00000000   0.0000000    0.000000
## 2       0  0.00000    0.0    0.00000000   0.0000000    0.000000
## 2100    0  0.00000    0.0    0.00000000   0.0000000    0.000000
## 2102    1 56.44627    0.0    0.09531018   1.4206958    2.360854
## 3       0  0.00000    0.0    0.00000000   0.0000000    0.000000
## 359     0  0.00000    0.0    0.00000000   0.0000000    0.000000
## 4       0  0.00000    0.0    0.00000000   0.0000000    0.000000
## 419     0  0.00000    0.0    0.00000000   0.0000000    0.000000
## 420     1 54.74059    0.5    0.58778666   0.9321641    2.332144
## 5       0  0.00000    0.0    0.00000000   0.0000000    0.000000
```

```
head(dat)
```

```
##   id      time status sex      age edema bili albumin protime logAlbumin
## 1  1  1.095140      1   1 58.76523   1.0 14.5    2.60    12.2  0.9555114
## 2  2 12.320329      0   1 56.44627   0.0  1.1    4.14    10.6  1.4206958
## 3  3  2.770705      1   0 70.07255   0.5  1.4    3.48    12.0  1.2470323
## 4  4  5.270363      1   1 54.74059   0.5  1.8    2.54    10.3  0.9321641
## 5  5  4.117728      0   1 38.10541   0.0  3.4    3.53    10.9  1.2612979
## 6  6  6.852841      1   1 66.25873   0.0  0.8    3.98    11.0  1.3812818
##   logProtime logBilirubin
## 1   2.501436   2.67414865
## 2   2.360854   0.09531018
## 3   2.484907   0.33647224
## 4   2.332144   0.58778666
## 5   2.388763   1.22377543
## 6   2.397895  -0.22314355
```

```
# Fit the Cox model
fit_3piece = coxph(as.formula(paste("Surv(start, stop, delta)~",
```

```
                                    paste(c(covsname1, covsname2, covsname3), collapse = "+"))),
                  data = dat_psd, id = id)

fit_3piece
```

```
## Call:
## coxph(formula = as.formula(paste("Surv(start, stop, delta)~",
##     paste(c(covsname1, covsname2, covsname3), collapse = "+"))),
##     data = dat_psd, id = id)
##
##                     coef exp(coef)  se(coef)      z        p
## sex1             0.42064   1.52293   0.49135  0.856 0.391954
## age1             0.04643   1.04752   0.01475  3.148 0.001641
## edema1           1.17325   3.23247   0.37394  3.138 0.001704
## logBilirubin1    0.51354   1.67119   0.13757  3.733 0.000189
## logAlbumin1     -3.30175   0.03682   1.01924 -3.239 0.001198
## logProtime1      6.09361 443.01814   1.23032  4.953 7.31e-07
## sex2            -0.42213   0.65565   0.37854 -1.115 0.264782
## age2             0.02994   1.03039   0.01338  2.238 0.025240
## edema2           0.81323   2.25517   0.53488  1.520 0.128412
## logBilirubin2    1.17056   3.22381   0.14178  8.256  < 2e-16
## logAlbumin2     -1.61244   0.19940   1.15735 -1.393 0.163553
## logProtime2      2.21004   9.11609   1.60046  1.381 0.167316
## sex3            -0.33526   0.71515   0.39403 -0.851 0.394859
## age3             0.03997   1.04078   0.01425  2.805 0.005031
## edema3          -0.17274   0.84136   0.85562 -0.202 0.840005
## logBilirubin3    0.79377   2.21172   0.16311  4.867 1.14e-06
## logAlbumin3     -3.20214   0.04067   1.31988 -2.426 0.015262
## logProtime3     -0.28010   0.75571   1.47765 -0.190 0.849655
##
## Likelihood ratio test=261  on 18 df, p=< 2.2e-16
## n= 996, number of events= 160
```

Remarks

- The Schoenfeld resiudals plot for `logProtime` shows a downward trend, indicating that $\beta(t)$ is decreasing over time. This coincides with the results from the 3-piece piecewise constant Cox model, which shows the coefficient for `logProtime` on the three time intervals are decreasing.