MATH 284/FMPH 291 Survival Analysis Supplementary Learing Materials

Yuyao Wang

2024-05-02

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

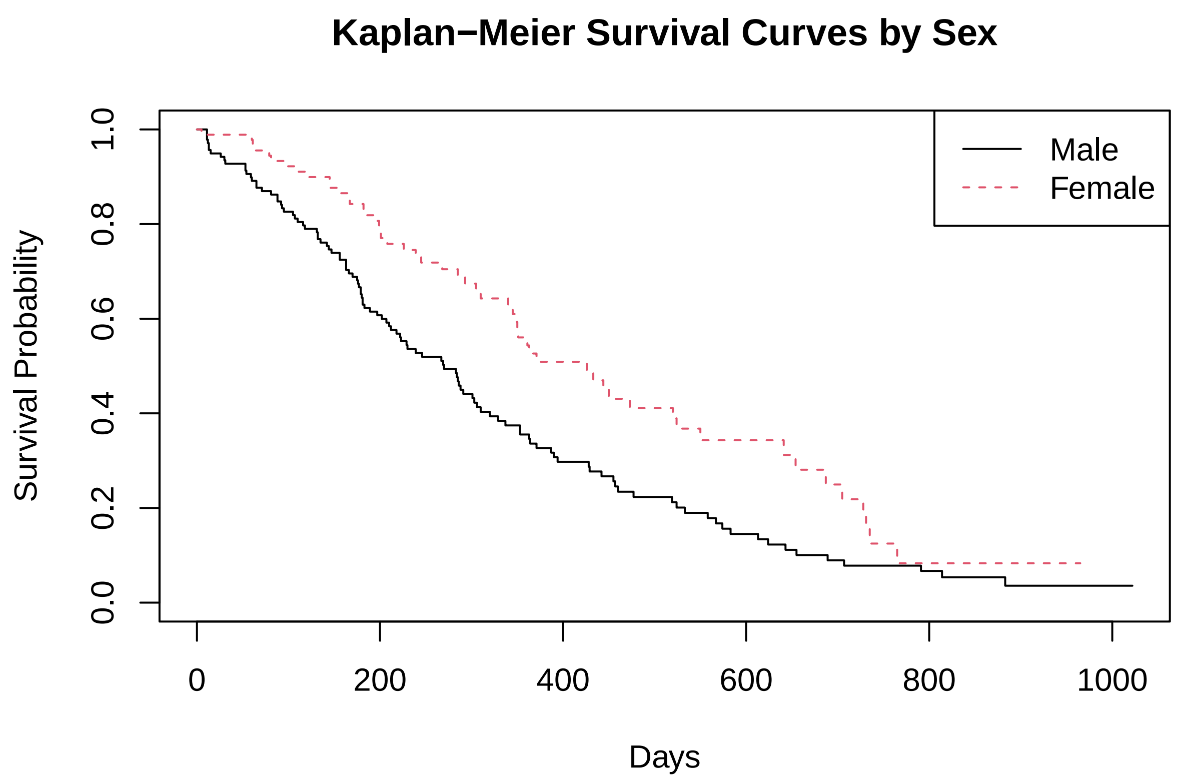
Part 1 in the above tutorial includes creating survival objects and curve, Kaplan-Meier plots, Estimating x-year survival, estimating median survival time, comparing survival times between groups (using log-rank test), and Cox regression model.

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()`.

```r
# 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)
##    2.954834
```

Cox 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:

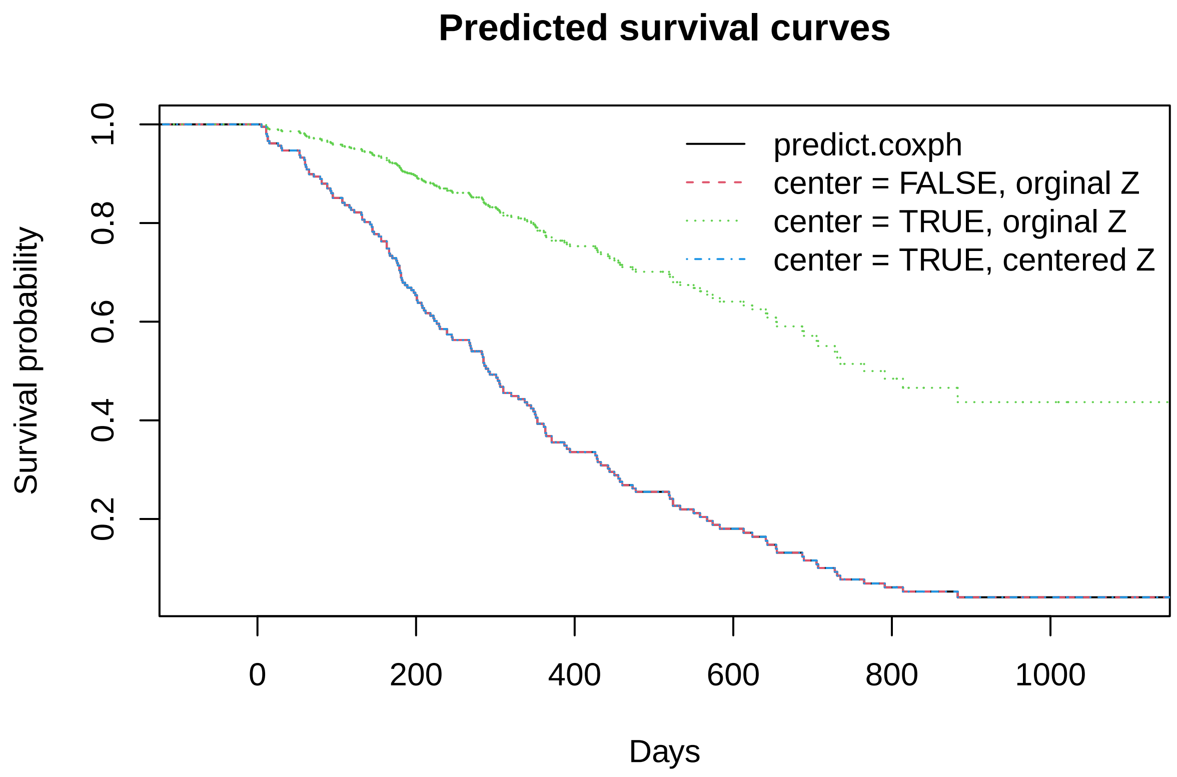
$$\hat{S}(t|Z) = e^{-\hat{\Lambda}_0(t)e^{\hat{\beta}^\top z}}.$$

See the example below.

```r
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)
```

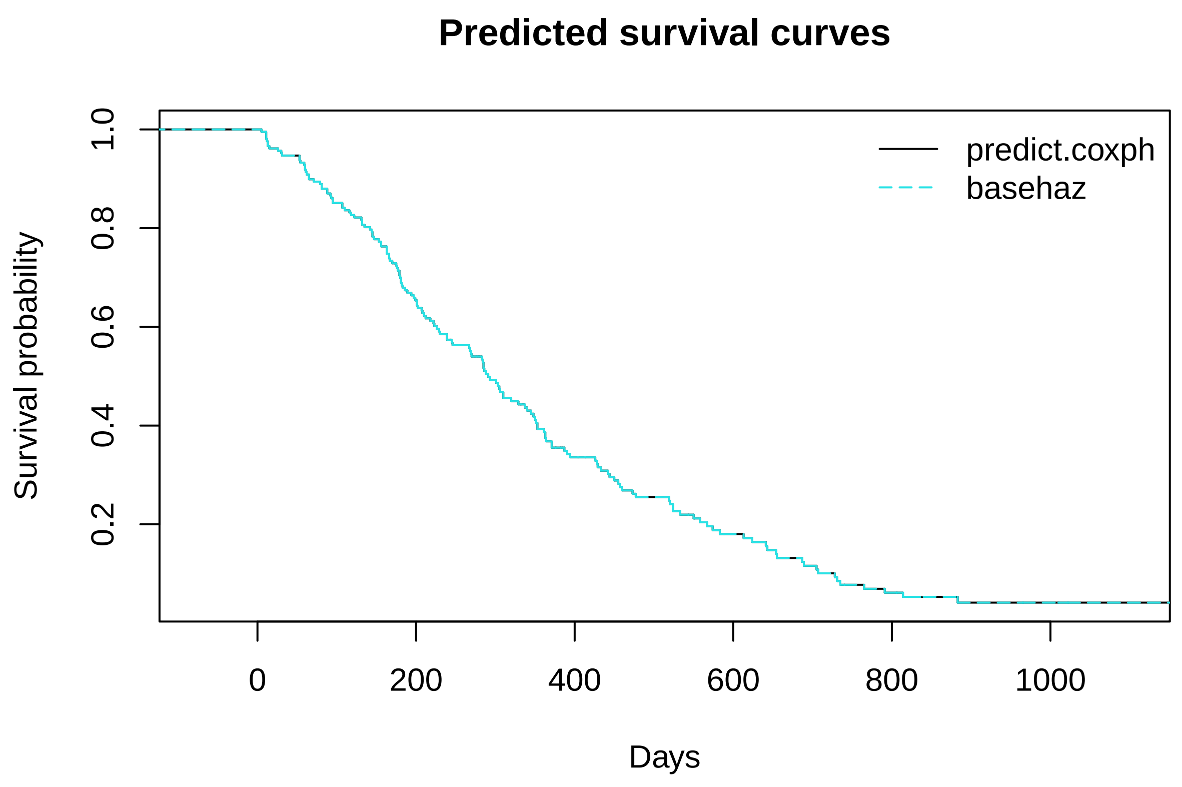
```r
ph.karno.new = 75

## 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 = T
       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 `hasehaz(.., newdata)`.

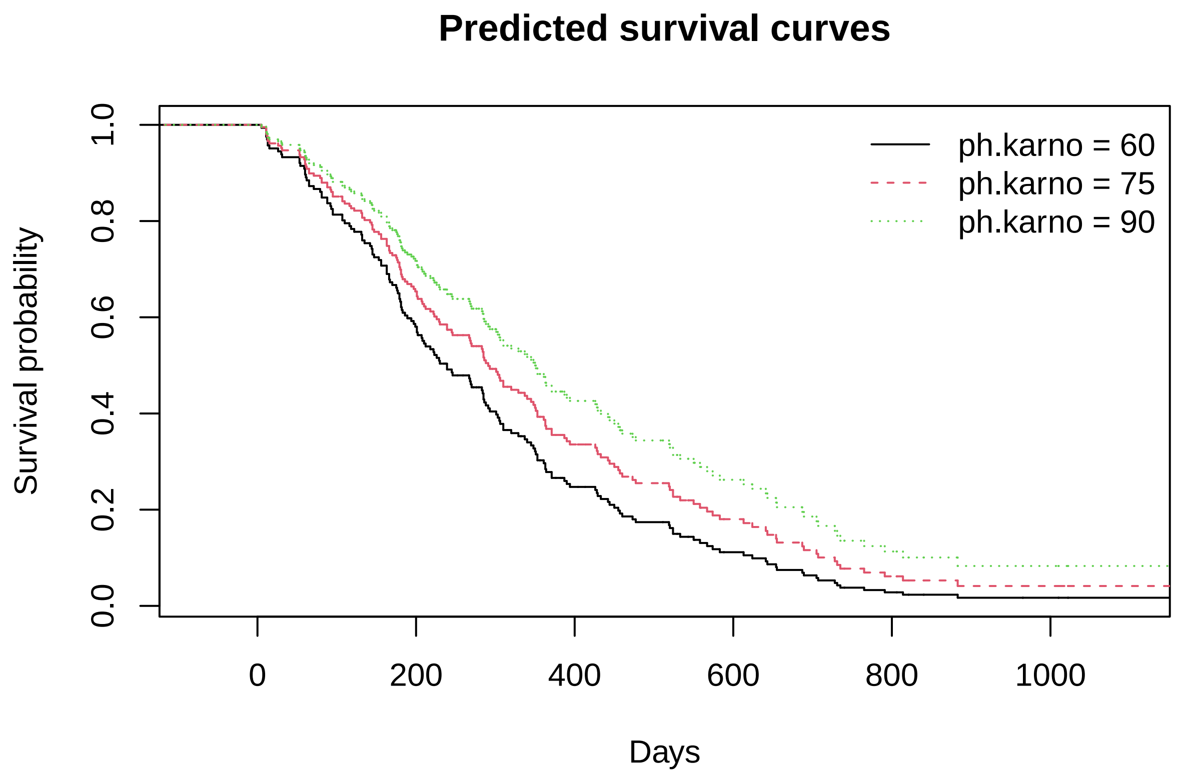
```
newdat = data.frame(ph.karno = ph.karno.new)
cumhazard = basehaz(coxfit, newdata = newdat)

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")
```



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

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

```r
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")
```


Predicted survival curves
Survival probability
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000
Days
ph.karno = 60
ph.karno = 75
ph.karno = 90