Emory Rollins School of Public Health | Department of Biostatistics and Bioinformatics

BIOS522: Survival Analysis Methods

Week 7: R Tutorial

Cox proportional hazards basics

Data example

For this tutorial, we will use data from the AIDS Clinical Trial Group (ACTG) randomized trial ACTG320. The trial assessed the safety and efficacy of the protease inhibitor indinavir in individuals with HIV and low CD4 cell count.

The study enrolled 1156 patients aged 16 years or older who had not previously been treated with protease inhibitors and who had CD4 cell count ≤200 cells/mL. The data for 1151 of these patients are contained within `actg320.csv`.

The primary outcome was time until the development of a new AIDS-defining event or death (`time, censor`). A secondary outcome was time until death (`time_d, censor_d`).

Patients were randomly assigned to receive either indinavir or matching placebo; the variable `tx` is equal to 1 for patients randomized to indinavir or 0 otherwise.

All patients received zidovudine/stavudine and lamivudine. Baseline covariates included sex, age, race or ethnic group, injection drug use, hemophilia, Karnofsky score (measure of functional impairment), baseline CD4 cell count (≤50 or 51-200 cells/mm3), and prior treatment with zidovudine.

We start by importing the data set and examining the columns.

```r
library(survival)
actg320 <- read.csv("actg320.csv", header = TRUE, sep = ",")
actg320[1:10,]
```

```
##    id time censor time_d censor_d tx txgrp cd4_grp sex raceth ivdrug hemophil
## 1   1  189      0    189        0  0     1       1   1      1      1        0
## 2   2  287      0    287        0  0     1       1   2      2      1        0
## 3   3  242      0    242        0  1     2       0   1      1      1        1
## 4   4  199      0    199        0  0     1       1   1      1      1        0
## 5   5  286      0    286        0  1     2       0   1      1      3        0
## 6   6  285      0    285        0  1     2       0   1      1      1        0
## 7   7  270      0    270        0  0     1       1   1      2      1        0
## 8   8  285      0    285        0  1     2       1   1      2      3        0
## 9   9  276      0    276        0  0     1       1   1      1      1        0
## 10 10  306      0    306        0  0     1       1   1      1      1        0
##    karnof   cd4 priorzdv age
## 1     100 169.0       39  34
## 2      90 149.5       15  34
## 3     100  23.5        9  20
## 4      90  46.0       53  48
## 5      90  10.0       12  46
## 6      70   0.0       24  51
## 7     100  54.5        6  51
## 8      80 117.5       24  40
## 9     100  95.0        7  34
## 10     90  71.0        7  38
```

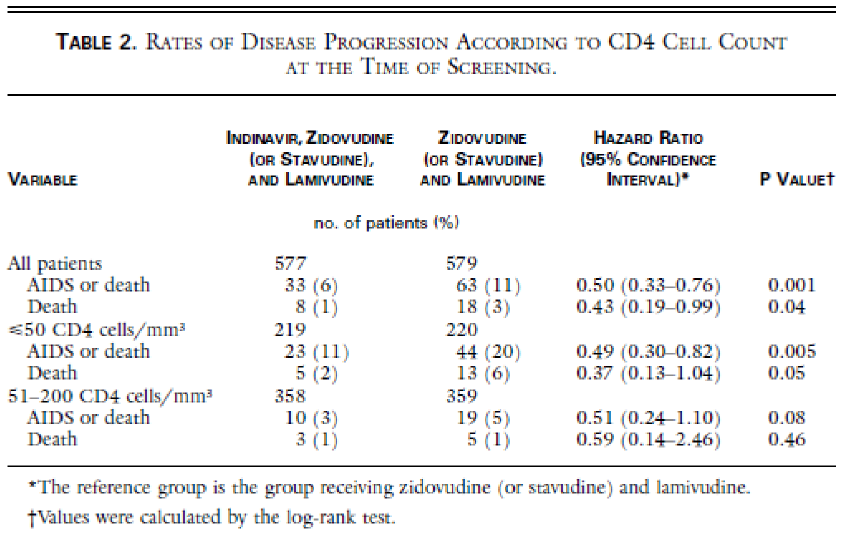


TABLE 2. Rates of Disease Progression According to CD4 Cell Count at the Time of Screening.

Variable	Indinavir, Zidovudine (or Stavudine), and Lamivudine	Zidovudine (or Stavudine) and Lamivudine	Hazard Ratio (95% Confidence Interval)*	P Value†
	no. of patients (%)			
All patients	577	579		
AIDS or death	33 (6)	63 (11)	0.50 (0.33–0.76)	0.001
Death	8 (1)	18 (3)	0.43 (0.19–0.99)	0.04
≤50 CD4 cells/mm³	219	220		
AIDS or death	23 (11)	44 (20)	0.49 (0.30–0.82)	0.005
Death	5 (2)	13 (6)	0.37 (0.13–1.04)	0.05
51–200 CD4 cells/mm³	358	359		
AIDS or death	10 (3)	19 (5)	0.51 (0.24–1.10)	0.08
Death	3 (1)	5 (1)	0.59 (0.14–2.46)	0.46

*The reference group is the group receiving zidovudine (or stavudine) and lamivudine.

†Values were calculated by the log-rank test.

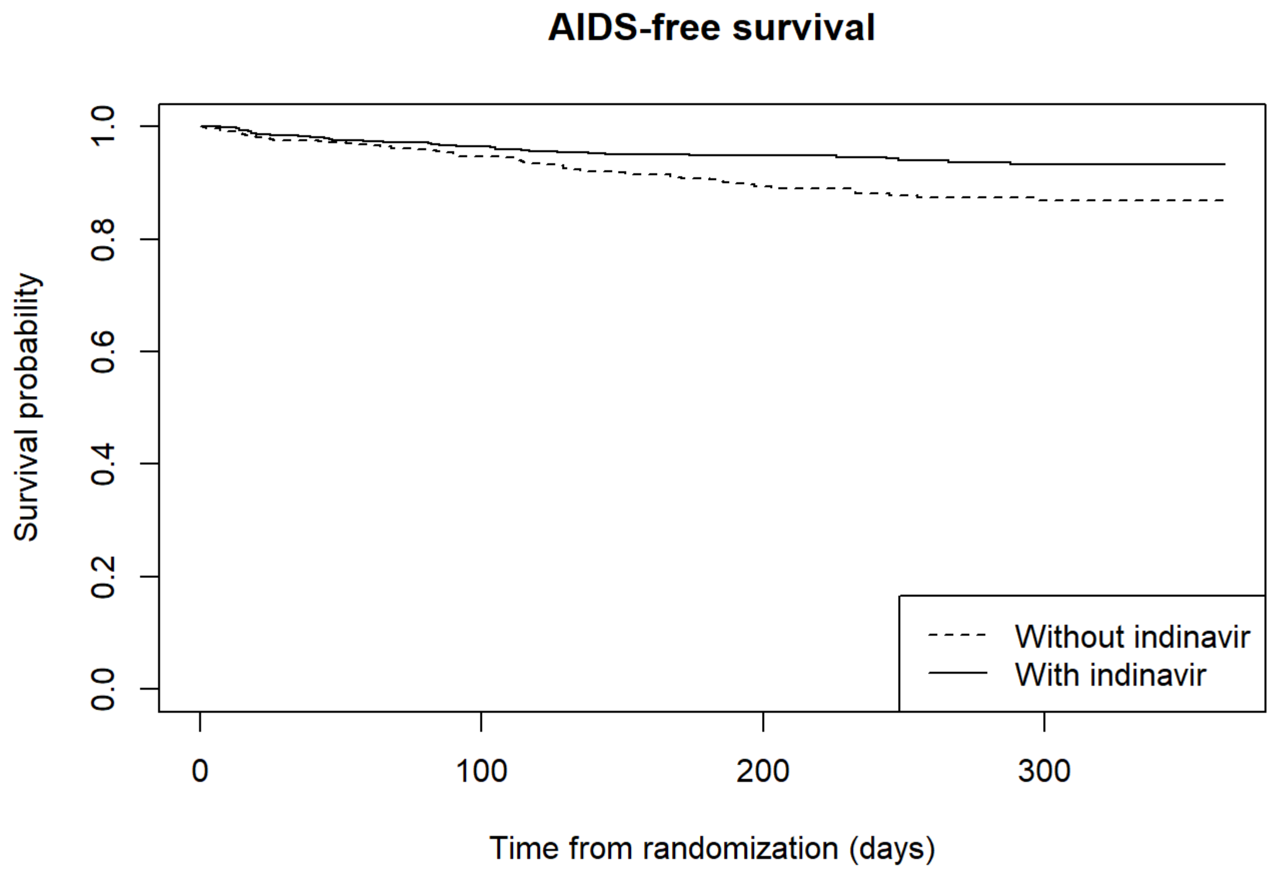
As part of this tutorial, we will replicate some of the primary results reported in Table 2.

Fitting a Cox model

The basic function for performing Cox proportional hazards regression is `coxph()`. Just like `survfit()` and `survdiff()`, we specify the censored failure time variable on the left of the model statement with the term `Surv(time,status)`. On the right of the model statement, we specify the independent covariate(s) in our model.

We can use the data to replicate the results in the top line of the paper's Table 2, including the hazard ratio (0.50), 95% confidence interval (0.33-0.76), and log-rank test p-value (0.001). First, we plot the data, and then we fit a Cox model with a single binary covariate for treatment, using the primary outcome. We can print the object to view abbreviated output, or use the `summary` function for more detail.

```r
plot(survfit(Surv(time,censor)~tx,data=actg320), lty=c(2,1),
     xlab = "Time from randomization (days)", ylab = "Survival probability",
     main = "AIDS-free survival")
legend("bottomright", legend=c("Without indinavir","With indinavir"),
      lty=c(2,1))
```



```r
ACTGcox.tx <- coxph(Surv(time,censor)~tx,data=actg320)
ACTGcox.tx
```

```
## Call:
## coxph(formula = Surv(time, censor) ~ tx, data = actg320)
##
##       coef exp(coef) se(coef)      z       p
## tx -0.6844    0.5044   0.2149 -3.185 0.00145
##
## Likelihood ratio test=10.7  on 1 df, p=0.001072
## n= 1151, number of events= 96
```

```r
summary(ACTGcox.tx)
```

```
## Call:
## coxph(formula = Surv(time, censor) ~ tx, data = actg320)
##
##   n= 1151, number of events= 96
##
##       coef exp(coef) se(coef)      z Pr(>|z|)
## tx -0.6844    0.5044   0.2149 -3.185  0.00145 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##    exp(coef) exp(-coef) lower .95 upper .95
## tx    0.5044      1.983     0.331    0.7686
##
## Concordance= 0.58  (se = 0.025 )
## Likelihood ratio test= 10.7  on 1 df,   p=0.001
## Wald test            = 10.14  on 1 df,   p=0.001
## Score (logrank) test = 10.54  on 1 df,   p=0.001
```

Review the summary statistics reported. `coef` refers to the log hazard ratio, and `exp(coef)` refers to the hazard ratio. A 95% confidence interval for the hazard ratio is included in the bottom half. Three global tests (likelihood ratio test, Wald test, score/log-rank test) are reported at the end, comparing the fitted model with a binary covariate for treatment to the null model with no covariates.

Hazard ratio and confidence interval

We can extract the coefficient(s) from the fitted model object using the `$` operator. Similarly, we can extract the variance/covariance matrix for the coefficients. Although the output provides confidence intervals for the hazard ratio constructed on the log scale, we could construct these confidence intervals ourselves as seen below.

```
coxph(Surv(time,censor)~tx,data=actg320)
```

```
## Call:
## coxph(formula = Surv(time, censor) ~ tx, data = actg320)
##
##       coef exp(coef) se(coef)      z       p
## tx -0.6844    0.5044   0.2149 -3.185 0.00145
##
## Likelihood ratio test=10.7  on 1 df, p=0.001072
## n= 1151, number of events= 96
```

```
ACTGcox.tx$coefficients
```

```
##         tx
## -0.6844423
```

```
ACTGcox.tx$var
```

```
##            [,1]
## [1,] 0.04619008
```

```
exp(ACTGcox.tx$coefficients - qnorm(0.975)*sqrt(ACTGcox.tx$var))
```

```
##           [,1]
## [1,] 0.3309873
```

```
exp(ACTGcox.tx$coefficients + qnorm(0.975)*sqrt(ACTGcox.tx$var))
```

```
##           [,1]
## [1,] 0.7685811
```

Methods for handling ties

For handling ties in failure times, `coxph()` uses Efron's partial likelihood as the default, and also allows the estimation based on the exact partial likelihood and Breslow's partial likelihood. This is managed with the `ties` option.

```
coxph(Surv(time,censor)~tx,data=actg320,ties="efron")
```

```
## Call:
## coxph(formula = Surv(time, censor) ~ tx, data = actg320, ties = "efron")
##
##       coef exp(coef) se(coef)      z       p
## tx -0.6844    0.5044   0.2149 -3.185 0.00145
##
## Likelihood ratio test=10.7  on 1 df, p=0.001072
## n= 1151, number of events= 96
```

```
coxph(Surv(time,censor)~tx,data=actg320,ties="breslow")
```

```
## Call:
## coxph(formula = Surv(time, censor) ~ tx, data = actg320, ties = "breslow")
##
##       coef exp(coef) se(coef)      z       p
## tx -0.6843    0.5044   0.2149 -3.184 0.00145
##
## Likelihood ratio test=10.7  on 1 df, p=0.001074
## n= 1151, number of events= 96
```

```
coxph(Surv(time,censor)~tx,data=actg320,ties="exact")
```

```
## Call:
## coxph(formula = Surv(time, censor) ~ tx, data = actg320, ties = "exact")
##
##       coef exp(coef) se(coef)      z       p
## tx -0.6846    0.5043   0.2150 -3.185 0.00145
##
## Likelihood ratio test=10.7  on 1 df, p=0.001071
## n= 1151, number of events= 96
```

In this example, observe only small differences in the point estimates and p-values.

Categorical covariates

We can fit multivariable models by adding covariates to the right-hand side of the model statement, separated by `+`. For categorical covariates, we include a `factor()` statement. Otherwise, `R` will default to treat the variable as a continuous covariate. An example here is Karnofsky score, which is a measure of functional impairment. In this data set, it takes values of 70, 80, 90, and 100, with 100 indicating no impairment.

```
table(actg320$karnof)
```

```
##
##  70  80  90 100
##  32 182 541 396
```

```
coxph(Surv(time,censor)~tx + factor(karnof),data=actg320)
```

```
## Call:
## coxph(formula = Surv(time, censor) ~ tx + factor(karnof), data = actg320)
##
##                        coef exp(coef) se(coef)      z        p
## tx                  -0.7035    0.4949   0.2150 -3.272  0.00107
## factor(karnof)80  -0.6660    0.5138   0.3638 -1.831  0.06714
## factor(karnof)90  -1.6229    0.1973   0.3555 -4.565 4.99e-06
## factor(karnof)100 -2.1212    0.1199   0.3987 -5.320 1.04e-07
##
## Likelihood ratio test=50.17  on 4 df, p=3.33e-10
## n= 1151, number of events= 96
```

Note how `R` automatically selects the lowest value (70) as the reference group. Three coefficients are reported for Karnofsky score, reporting the hazard ratio for that score (80, 90, or 100) versus 70, holding age constant. Note the likelihood ratio test, Wald test, and score test at the bottom are global tests (4 df) that compare the fitted model to a model with no covariates. Local Wald statistics are reported for each covariate at the top.

For a categorical variable like Karnofsky score, we may desire a likelihood ratio test comparing the models with and without Karnofsky score. This captures the overall significance of the Karnofsky variable, adjusting for other variables in the model. We can fit two models and use the `anova()` function to calculate a likelihood ratio statistic.

```
m1 <- coxph(Surv(time,censor)~tx,data=actg320)
m2 <- coxph(Surv(time,censor)~tx + factor(karnof),data=actg320)
anova(m1,m2)
```

```
## Analysis of Deviance Table
##  Cox model: response is  Surv(time, censor)
##  Model 1: ~ tx
##  Model 2: ~ tx + factor(karnof)
##    loglik  Chisq Df P(>|Chi|)
## 1 -653.09
## 2 -633.36 39.469  3 1.381e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The resulting chi-square test statistic on 3 df is highly significant.

Continuous covariates

We can also add in continuous variables like age.

```
coxph(Surv(time,censor)~tx + factor(karnof) + age,data=actg320)
```

```
## Call:
## coxph(formula = Surv(time, censor) ~ tx + factor(karnof) + age,
##     data = actg320)
##
##                        coef exp(coef) se(coef)      z        p
## tx                 -0.70541   0.49391  0.21501 -3.281  0.00104
## factor(karnof)80   -0.64245   0.52600  0.36443 -1.763  0.07792
## factor(karnof)90   -1.57910   0.20616  0.35781 -4.413 1.02e-05
## factor(karnof)100 -2.06710   0.12655  0.40198 -5.142 2.71e-07
## age                 0.01181   1.01188  0.01104  1.070  0.28450
##
## Likelihood ratio test=51.29  on 5 df, p=7.558e-10
## n= 1151, number of events= 96
```

If we need to transform a continuous variable (e.g. log transformation), we can include certain types of transformations directly in the model statement, such as `log` or `sqrt` .

```
coxph(Surv(time,censor)~tx + factor(karnof) + log(age),data=actg320)
```

```
## Call:
## coxph(formula = Surv(time, censor) ~ tx + factor(karnof) + log(age),
##     data = actg320)
##
##                       coef exp(coef) se(coef)      z        p
## tx                 -0.7054    0.4939   0.2150 -3.280  0.00104
## factor(karnof)80  -0.6450    0.5247   0.3644 -1.770  0.07678
## factor(karnof)90  -1.5825    0.2055   0.3580 -4.420 9.88e-06
## factor(karnof)100 -2.0730    0.1258   0.4020 -5.156 2.52e-07
## log(age)           0.4198    1.5216   0.4577  0.917  0.35904
##
## Likelihood ratio test=51.01  on 5 df, p=8.605e-10
## n= 1151, number of events= 96
```

```
coxph(Surv(time,censor)~tx + factor(karnof) + sqrt(age),data=actg320)
```

```
## Call:
## coxph(formula = Surv(time, censor) ~ tx + factor(karnof) + sqrt(age),
##     data = actg320)
##
##                       coef exp(coef) se(coef)      z        p
## tx                 -0.7055    0.4939   0.2150 -3.281  0.00103
## factor(karnof)80  -0.6434    0.5255   0.3644 -1.765  0.07749
## factor(karnof)90  -1.5801    0.2059   0.3579 -4.415 1.01e-05
## factor(karnof)100 -2.0693    0.1263   0.4020 -5.147 2.65e-07
## sqrt(age)          0.1444    1.1553   0.1437  1.004  0.31527
##
## Likelihood ratio test=51.17  on 5 df, p=8e-10
## n= 1151, number of events= 96
```

For more complex transformations, we can perform these directly in the model statement if we use `I` to specify the transformation. For example, we could create a binary variable for age greater than or equal to 60 years.

```
coxph(Surv(time,censor)~tx + factor(karnof) + I(age>=60),data=actg320)
```

```
## Call:
## coxph(formula = Surv(time, censor) ~ tx + factor(karnof) + I(age >=
##     60), data = actg320)
##
##                       coef exp(coef) se(coef)      z        p
## tx                 -0.7057    0.4938   0.2152 -3.279  0.00104
## factor(karnof)80  -0.6687    0.5124   0.3640 -1.837  0.06618
## factor(karnof)90  -1.6274    0.1964   0.3561 -4.570 4.87e-06
## factor(karnof)100 -2.1284    0.1190   0.4000 -5.321 1.03e-07
## I(age >= 60)TRUE  -0.1233    0.8840   0.5897 -0.209  0.83431
##
## Likelihood ratio test=50.21  on 5 df, p=1.253e-09
## n= 1151, number of events= 96
```

Interactions

We can also include interactions in our model statement. Considering the data from the ACTG 320 trial, investigators were interested in whether the effect of treatment with indinavir differed between individuals with higher CD4 (healthier) and individuals with low CD4 (sicker). We create an interaction between `tx` and the binary variable `cd4_grp` by adding `tx*cd4_grp` .

```
coxph(Surv(time,censor)~tx*cd4_grp + factor(karnof) + age,data=actg320)
```

```
## Call:
## coxph(formula = Surv(time, censor) ~ tx * cd4_grp + factor(karnof) +
##     age, data = actg320)
##
##                         coef exp(coef) se(coef)      z        p
## tx                  -0.65784   0.51797  0.25942 -2.536 0.011220
## cd4_grp             -1.15891   0.31383  0.28365 -4.086  4.4e-05
## factor(karnof)80    -0.50551   0.60320  0.36548 -1.383 0.166623
## factor(karnof)90    -1.21855   0.29566  0.36395 -3.348 0.000813
## factor(karnof)100   -1.65442   0.19120  0.40651 -4.070  4.7e-05
## age                  0.02045   1.02066  0.01133  1.805 0.071027
## tx:cd4_grp          -0.07426   0.92843  0.47083 -0.158 0.874669
##
## Likelihood ratio test=80.62  on 7 df, p=1.028e-14
## n= 1151, number of events= 96
```

We note that the Wald test for the interaction (`tx:cd4_grp`) is not statistically significant.

Predicted survival

Predicted survival from Cox models can be calculated using the functions `survfit()` or `predict()` . In our `survfit()` statement, we first specify a stored Cox model that we will use to predict survival probabilities. For this example, we use our model with `tx` only, saved previously as `ACTGcox.tx` .

We use the option `newdata` to construct a sample data set with the covariate patterns we want to examine. In this model, there are only two possible covariate patterns: `tx=1` (indinavir), `tx=0` (placebo). We start by generating predictions for placebo patients only. We specify `newdata = data.frame(tx=0)` . We store our output under the name `ACTGcox.txfit` .

```
ACTGcox.txfit <- survfit(ACTGcox.tx,newdata=data.frame(tx=0))
```

Then, using the summary() function, we can specify times for which we want to predict survival. Below, times 50, 100, 150 days are specified. The output includes the predicted survival, standard error, and a 95% confidence interval at each requested time.

```
summary(ACTGcox.txfit, times=c(50,100,150))
```

```
## Call: survfit(formula = ACTGcox.tx, newdata = data.frame(tx = 0))
##
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##    50   1090      30    0.965 0.00672        0.952        0.978
##   100   1015      19    0.942 0.00899        0.925        0.960
##   150    916      22    0.914 0.01147        0.892        0.937
```

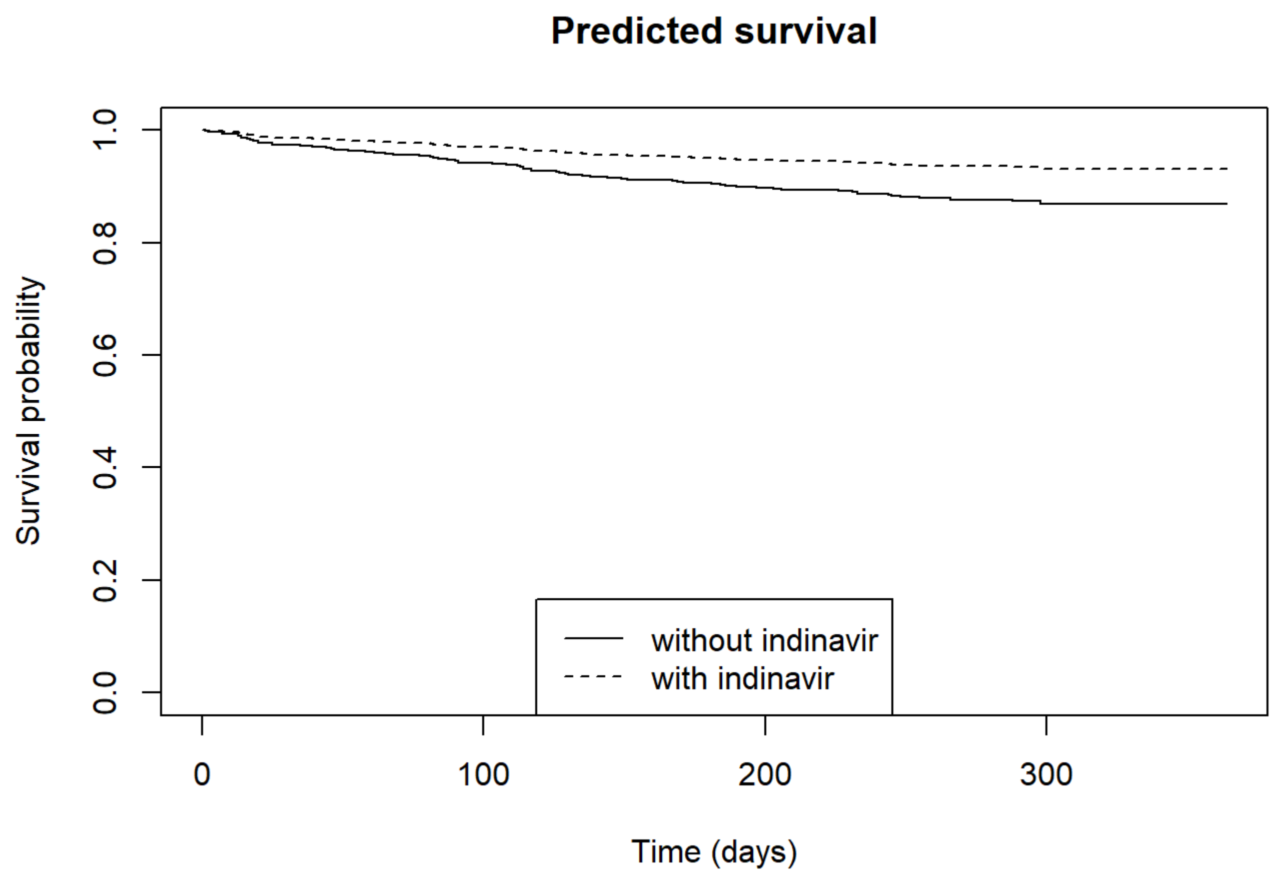
If we want to examine more than one covariate pattern at a time, we can specify `newdata = data.frame(tx=c(0,1))` . We store our output under the name `ACTGcox.txfit2` . Then, using the `summary()` function, we can examine survival at requested times. When `newdata` includes more than one covariate pattern, a new column is displayed for each pattern. The column `survival1` corresponds to the first group in the data frame (`tx=0`). The column `survival2` corresponds to the second group in the data set (`tx=1`). Confidence intervals are not displayed.

```
ACTGcox.txfit2 <- survfit(ACTGcox.tx,newdata=data.frame(tx=c(0,1)))
summary(ACTGcox.txfit2, times = c(50,100,150))
```

```
## Call: survfit(formula = ACTGcox.tx, newdata = data.frame(tx = c(0,
##     1)))
##
##  time n.risk n.event survival1 survival2
##    50   1090      30     0.965     0.982
##   100   1015      19     0.942     0.970
##   150    916      22     0.914     0.956
```

We can plot the results using the `plot()` function. We distinguish the two lines by specifying the line type and add an informative legend.

```
plot(ACTGcox.txfit2, lty=c(1,2), main="Predicted survival", xlab="Time (days)",
     ylab="Survival probability")
legend("bottom", lty=c(1,2), legend=c("without indinavir","with indinavir"))
```



Model diagnostics

Residuals

The survival package includes a built-in function residuals() for calculating residuals for a fitted Cox proportional hazards model. The types include: `type=”martingale”` , `type=”deviance”` , `type=”schoenfeld”` , and `type=”scaledsch”` (scaled Schoenfeld)

There is no built-in function for Cox-Snell (generalized) residuals, though they can be calculated from the Martingale residuals and the status variable. We will instead focus on the residual methods that are readily available in `R` .

For this exercise, we will start with a model that includes treatment, Karnofsky score, and baseline CD4 value (continuous)

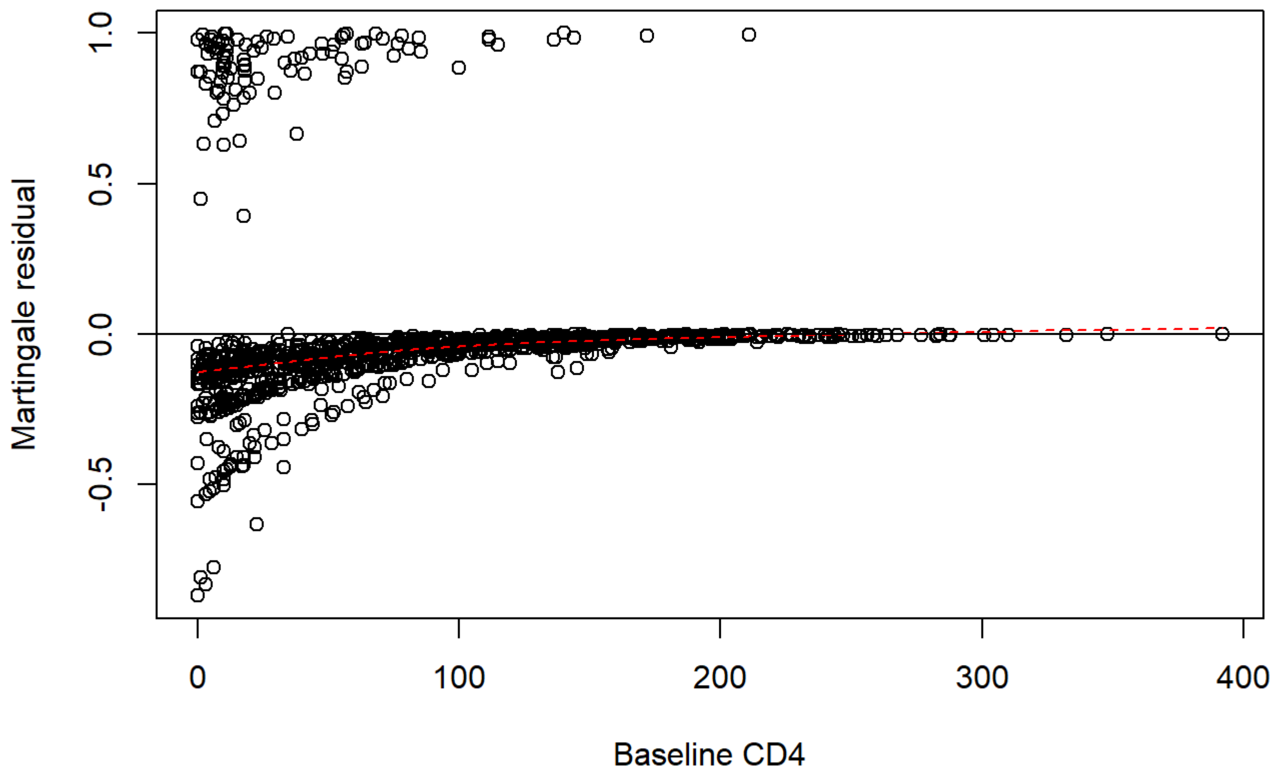
```
ACTGcox <- coxph(Surv(time,censor)~tx + factor(karnof) + cd4,data=actg320)
summary(ACTGcox)
```

```
## Call:
## coxph(formula = Surv(time, censor) ~ tx + factor(karnof) + cd4,
##     data = actg320)
##
##   n= 1151, number of events= 96
##
##                         coef exp(coef)  se(coef)      z Pr(>|z|)
## tx                 -0.646524  0.523863  0.215381 -3.002  0.00268 **
## factor(karnof)80  -0.446976  0.639559  0.364834 -1.225  0.22052
## factor(karnof)90  -1.142121  0.319141  0.360471 -3.168  0.00153 **
## factor(karnof)100 -1.589237  0.204081  0.404259 -3.931 8.45e-05 ***
## cd4                -0.014031  0.986066  0.002492 -5.631 1.80e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                    exp(coef) exp(-coef) lower .95 upper .95
## tx                    0.5239      1.909   0.34347    0.7990
## factor(karnof)80     0.6396      1.564   0.31285    1.3075
## factor(karnof)90     0.3191      3.133   0.15745    0.6469
## factor(karnof)100    0.2041      4.900   0.09241    0.4507
## cd4                   0.9861      1.014   0.98126    0.9909
##
## Concordance= 0.777  (se = 0.023 )
## Likelihood ratio test= 95.84  on 5 df,   p=<2e-16
## Wald test            = 79.82  on 5 df,   p=9e-16
## Score (logrank) test = 97.56  on 5 df,   p=<2e-16
```

Martingale residuals

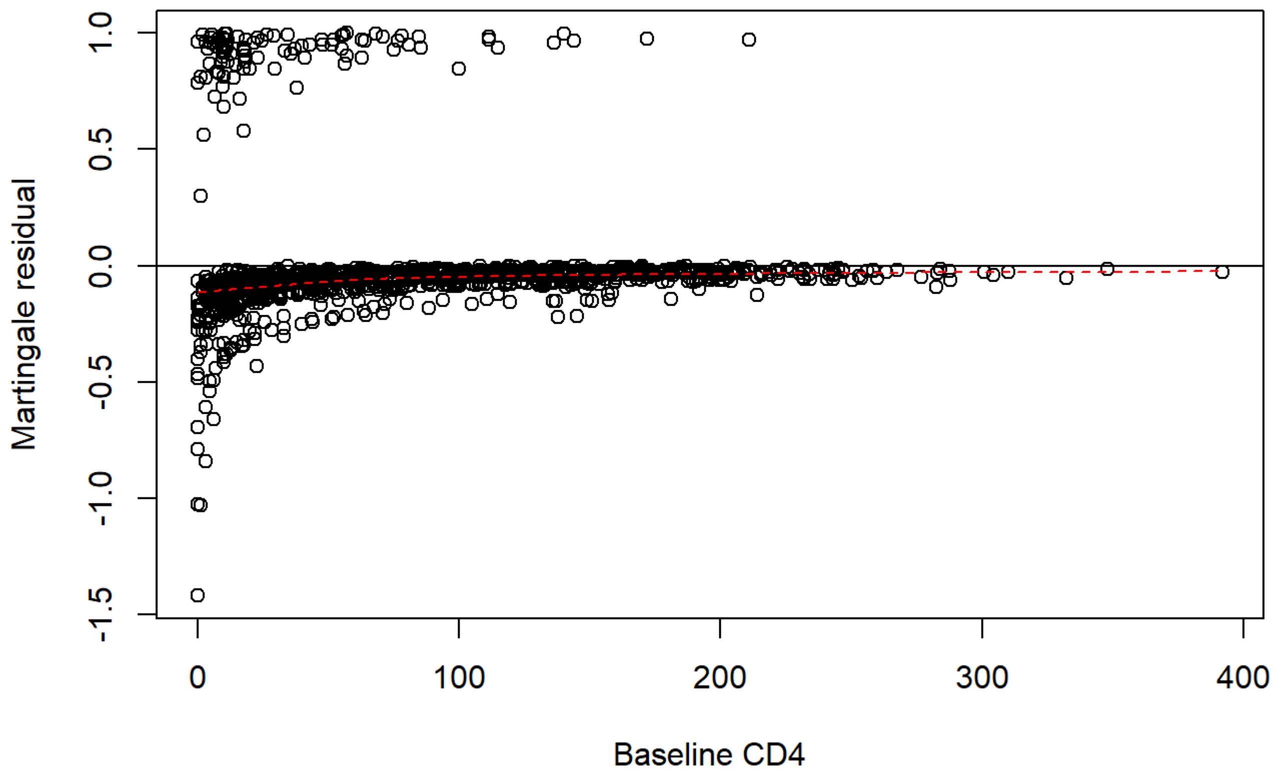
We can fit Martingale residuals to the data using the `residuals()` function by first specifying the stored Cox model (`ACTGcox`) and then residual type. We store the residuals in the vector `martinres` . We plot the residuals on the y-axis against the x-axis of CD4 (actg320$cd4). We add a helpful smoothing line and a horizontal line at 0.

```
martinres <- residuals(ACTGcox,type="martingale")
plot(x=actg320$cd4,y=martinres,
     xlab="Baseline CD4",ylab="Martingale residual")
lines(lowess(x=actg320$cd4,y=martinres),lty=2,col="red")
abline(h=0)
```



We use this plot to examine the assumption of linearity of the effect of baseline CD4. We see some evidence of lack of fit for low levels of baseline CD4. We may investigate transformations (e.g. log transformation) to improve fit. Because some individuals have CD4 of 0, we add one before taking the log.

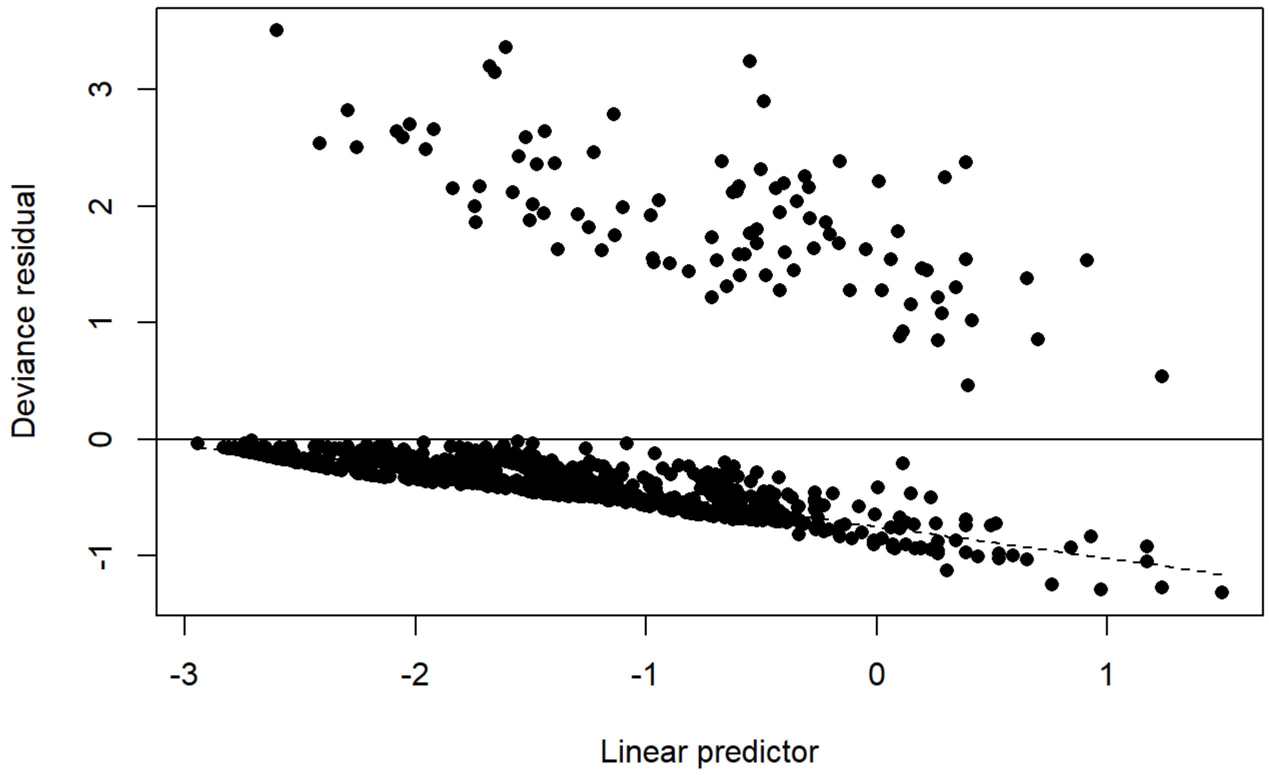
```
ACTGcox2 <- coxph(Surv(time,censor)~tx + factor(karnof) + I(log(cd4+1)),data=actg320)
martinres2 <- residuals(ACTGcox2,type="martingale")
plot(x=actg320$cd4,y=martinres2,
     xlab="Baseline CD4",ylab="Martingale residual")
lines(lowess(x=actg320$cd4,y=martinres2),lty=2,col="red")
abline(h=0)
```



Deviance residuals

We can fit deviance residuals in the same manner, changing only the residual type (type="deviance"). The deviance residuals are easier to examine because they are more symmetric around 0. For both the Martingale and deviance residuals, we can plot them against the linear predictor on the x-axis. We can use the `predict()` function in `R` to calculate the linear predictor for each individual using the fitted model.

```
devres <- residuals(ACTGcox,type="deviance")
ACTGcox2_lp <- predict(ACTGcox2,type="lp")
plot(x=ACTGcox2_lp,y=devres,pch=16,
     xlab="Linear predictor",ylab="Deviance residual")
lines(lowess(x=ACTGcox2_lp,y=devres),lty=2)
abline(h=0)
```



This result suggest that the model would benefit from more variables to explain survival, particularly for patients with high values of the linear predictor (those identified by the covariates included as being at highest risk).

Schoenfeld residuals

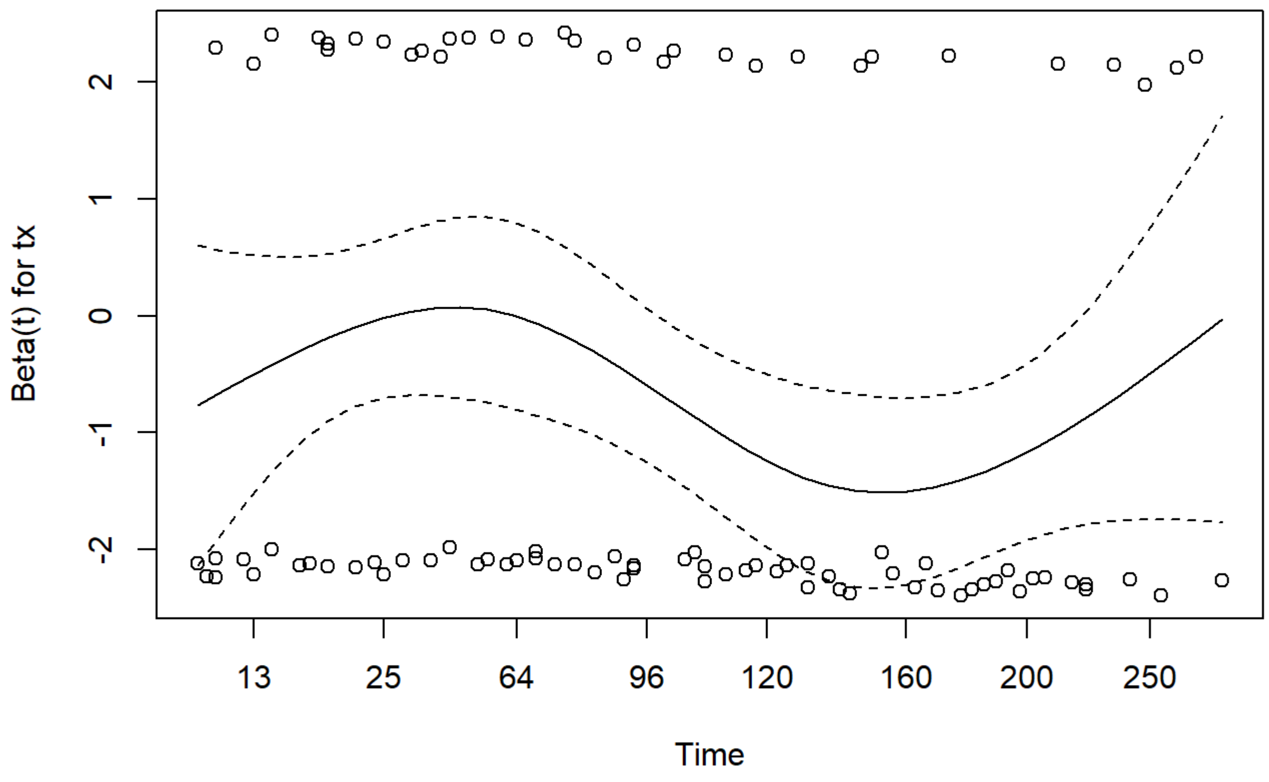
We can fit Schoenfeld and scaled Schoenfeld residuals in the same manner, changing only the residual type (`type=”schoenfeld”` or `type=”scaledsch”`). Unlike the other residual methods, though, this will produce one residual per covariate, and it will only calculate a residual for the participants with an observed failure time.

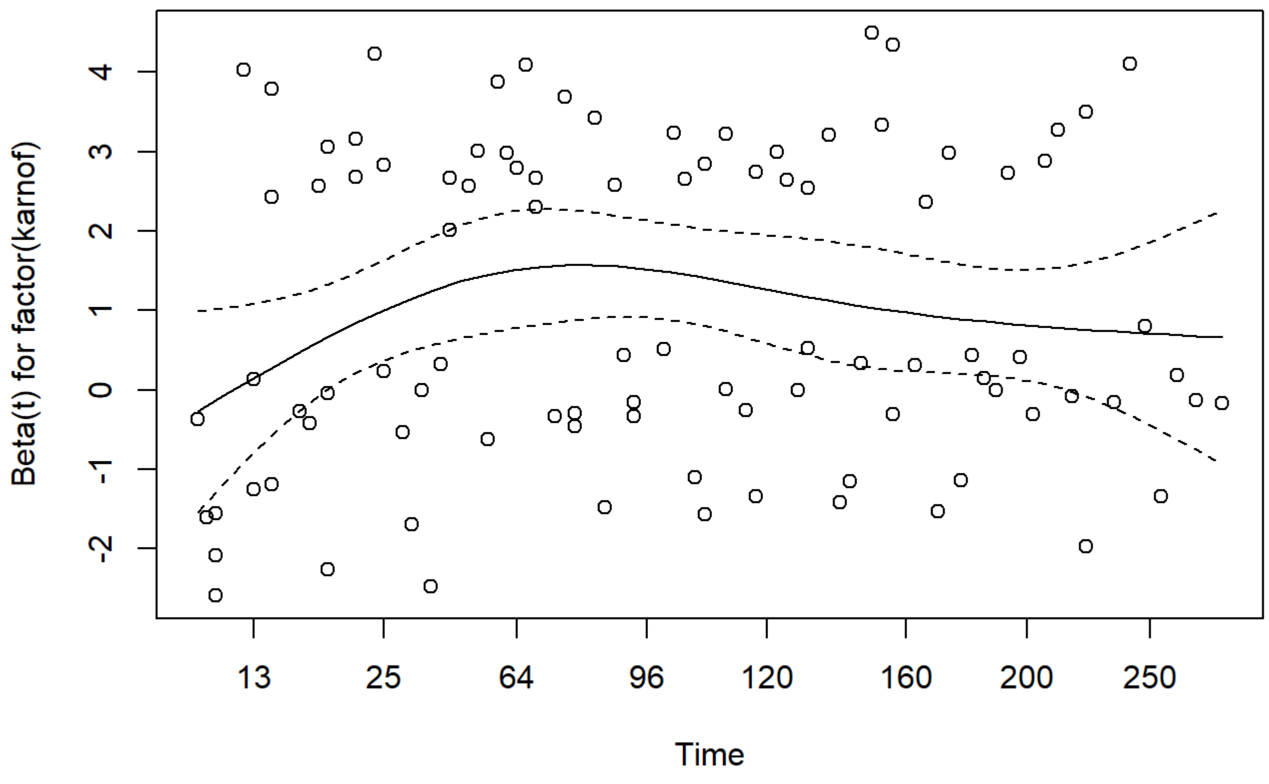
To generate plots of the scaled Schoenfeld residuals for each covariate, R has a built-in function `cox.zph()` . This function will also test whether there is evidence of a linear association between the covariate and the scaled Schoenfeld residual.

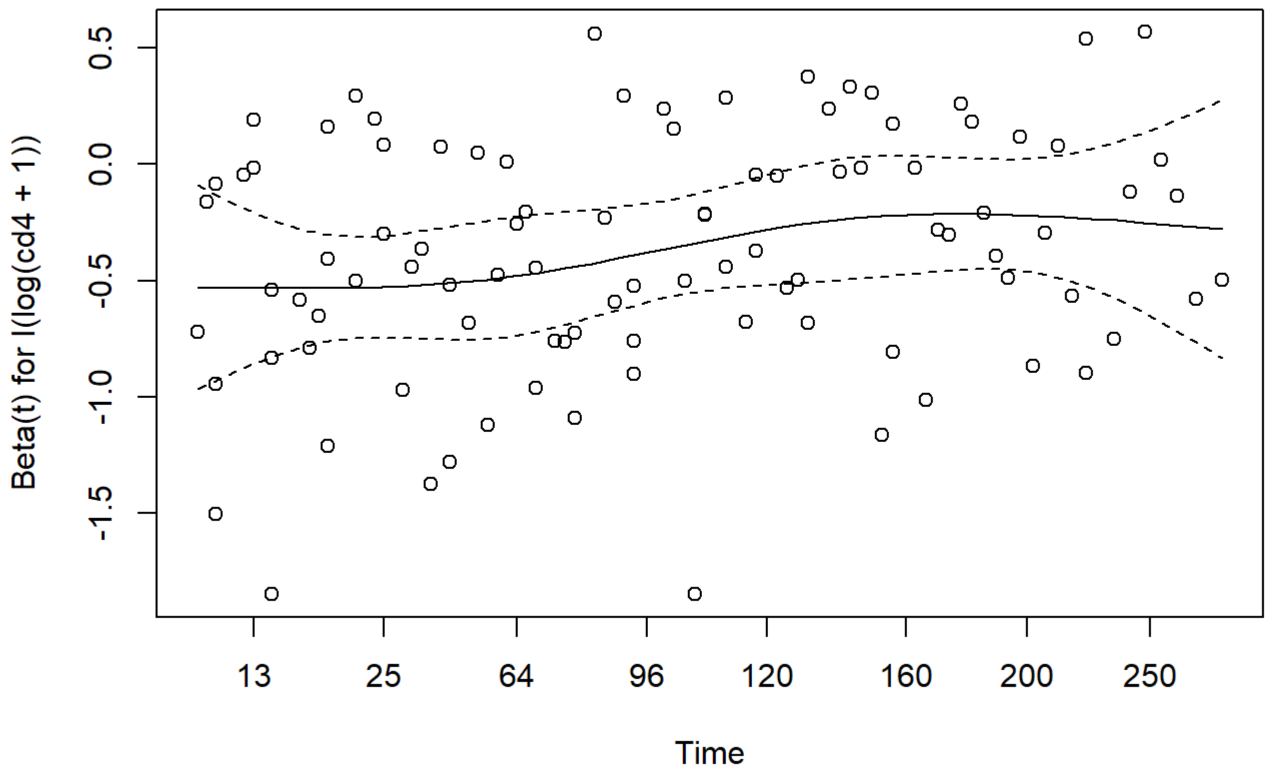
```
scaledres <- residuals(ACTGcox2,type="scaledsch")
cox.zph(ACTGcox2)
```

```
##                   chisq df    p
## tx                 1.88  1 0.17
## factor(karnof)     2.07  3 0.56
## I(log(cd4 + 1))    2.64  1 0.10
## GLOBAL             7.00  5 0.22
```

```
plot(cox.zph(ACTGcox2))
```


Beta(t) for tx
2
1
0
-1
-2
13
25
64
96
120
160
200
250
Time


Beta(t) for factor(karnof)
4
3
2
1
0
-1
-2
13
25
64
96
120
160
200
250
Time



None of the covariates have scaled Schoenfeld residuals with a statistically significant non-zero slope, although the pattern for log(CD4) shows some evidence of a deviation.

AIC

To calculate AIC, we can use the `extractAIC()` function on any fitted `coxph` object.

```
extractAIC(ACTGcox2)
```

```
## [1]    5.000 1249.819
```

The first term is the number of parameters. The second is the AIC itself.

End of tutorial