Biostatistics Task 2

Danyu Zhang & Daniel Alonso

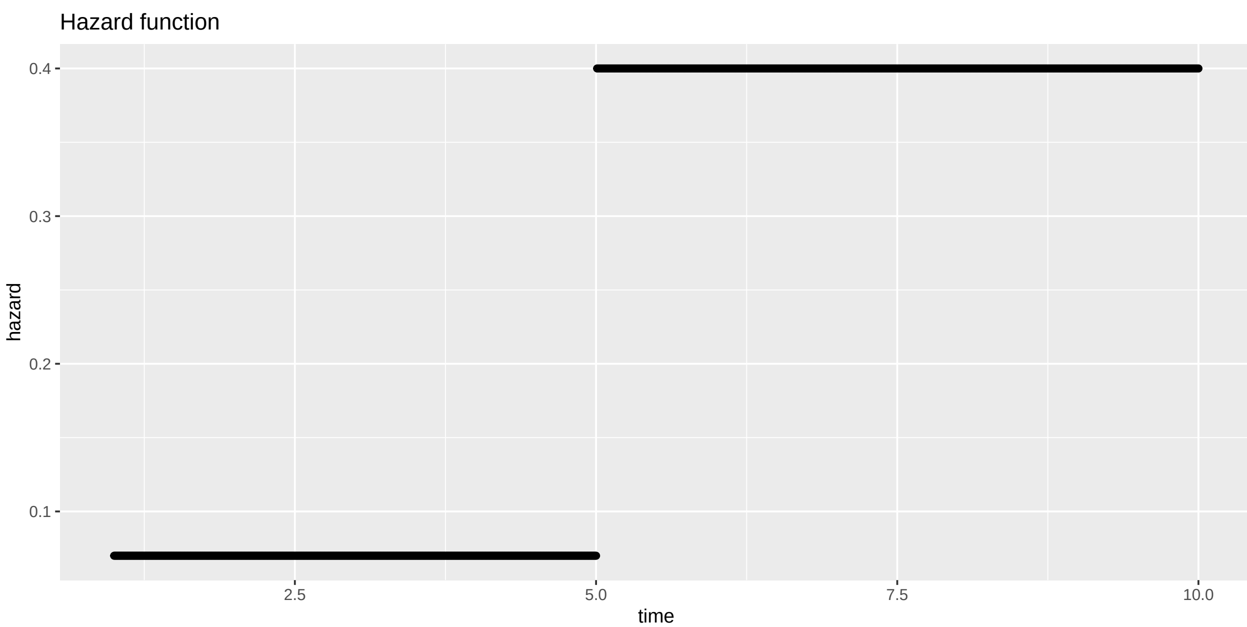
May 28th, 2021

```
library(ggplot2)
library(survival)
library(ggfortify)
library(coin)
```

Exercise 1

Hazard function plot

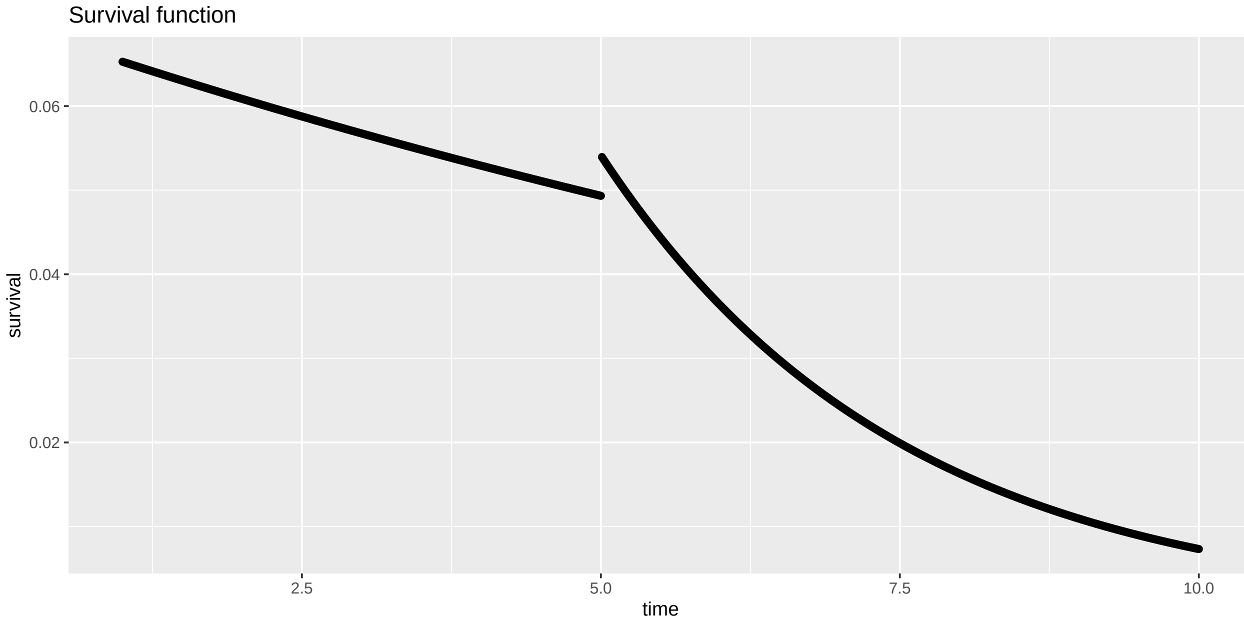
We have a piecewise hazard function as follows:



Survival plot

Given that the survival function must be a smooth function, we obtain a survival function

We obtain a piecewise survival function whose first chunk corresponds

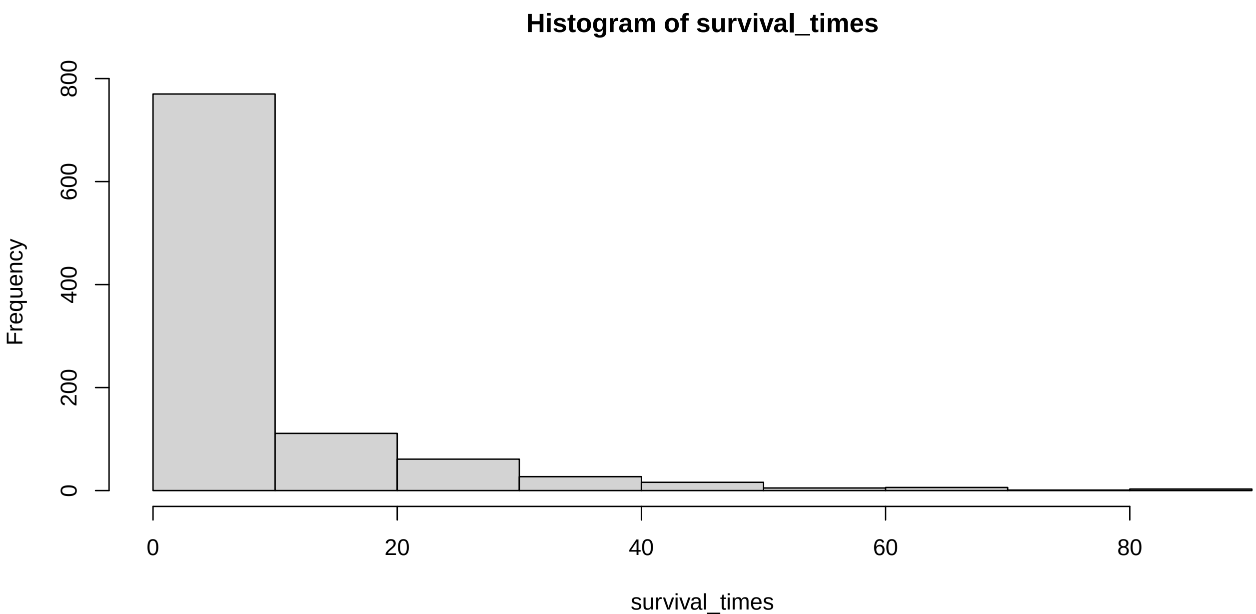


Survival times simulation

```
# number of trials
survival_times <- sapply(theta, function(lambda) {rexp(1,rate=1/lambda)})
```

Histogram for survival times

We plot a histogram for the survival times



Median survival time

As we can see, sampling from the distribution results in the following median survival time:

```
#> [1] 3.271372
```

Exercise 2

Given the following density function:

$$f(y) = (\lambda_0 + \lambda_1 y)e^{-\lambda_0 y - \frac{1}{2}\lambda_1 y^2}$$

We obtain the survival function as follows:

$$\begin{aligned}
S(t) = P(T > t) = &\int_t^\infty (\lambda_0 + \lambda_1 y)e^{-\lambda_0 y - \frac{1}{2}\lambda_1 y^2} dy \\
&= \lim_{b\to\infty} [-e^{\frac{\lambda_1 b^2}{2} - \lambda_0 b}] + e^{\frac{-\lambda_1 t^2}{2} - \lambda_0 t} \\
&= 0 + e^{\frac{-\lambda_1 t^2}{2} - \lambda_0 t} \\
S(t) = &\, e^{\frac{-\lambda_1 t^2}{2} - \lambda_0 t},\ \lambda_1 \in \mathbb{R}, \lambda_0 > 0
\end{aligned}$$

We obtain the hazard function as follows:

$$h(t) = \frac{f(t)}{S(t)} = \frac{(\lambda_0 + \lambda_1 t)e^{-\lambda_0 t - \frac{1}{2}\lambda_1 t^2}}{e^{-\lambda_0 t - \frac{1}{2}\lambda_1 t^2}} = \lambda_0 + \lambda_1 t$$

$$h(t) = \lambda_0 + \lambda_1 t$$

And the cumulative hazard function:

$$H(t) = -log(S(t)) = \frac{\lambda_1 t^2}{2} + \lambda_0 t$$

Exercise 3

KM estimator implementation of the survival function

Our implementation is as follows:

Parameters:

- **dataset**: Dataset to obtain the KM estimation from
- **events**: specific column of the dataset corresponding to the events (deemed *status* for the *aml* dataset)

Algorithm:

- Obtain length of the dataset by selecting the first column and utilizing *length* to obtain it
- Create the survival vector with (by definition) survival probability of 1 in the first time instance
- Initilize a counter j to keep track of events occurring in the column of the dataset passed as the **events** parameter
- Iterate over the length of the dataset (*1:length(dataset[,1])*) using i as iteration variable
  - During the loop we check if the i-th element of **events** does corespond to an event (1) or not (0)
  - If so, we add one to the counter and calculate the survival probability as a product of the previous survival probability obtained in the previous positive (1) event

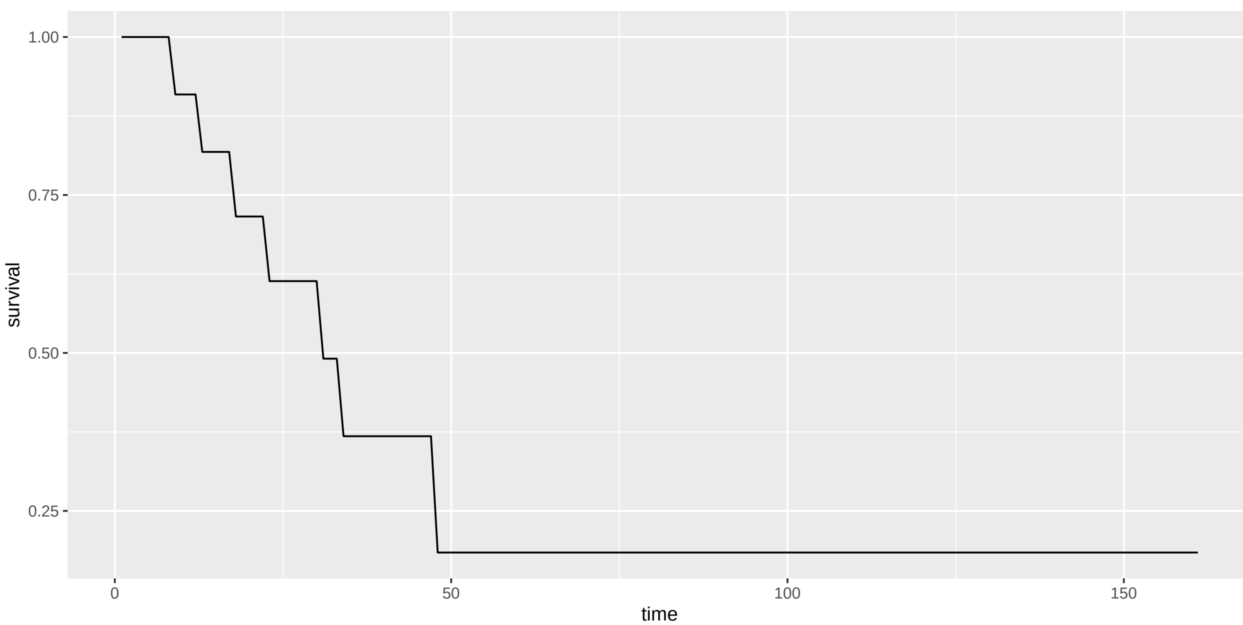
        * We append the survival probability to the survival vector
    - 1 is subtracted from the total length of the dataset as one element in the dataset has been traversed. We use this n_c variable as a component of our survival probability calculation.

- Return the survival probability vector, the times where the survival probability changes correspond to the original time of events in the original passed on **dataset**.

```
km_est <- function(dataset, events) {
    # calculate survival probabilities
    n_c <- length(dataset[,1])
    survival <- c(1)
    j = 0
    for (i in 1:length(dataset[,1])) {
        if (events[i] == 1) {
            j = j+1
            prob <- (1-(1/n_c))*survival[j]
            survival <- c(survival, prob)
        }
        n_c <- n_c - 1
    }
    return(survival)
}
```

Utilizing the function to obtain the survival function for the leukemia dataset

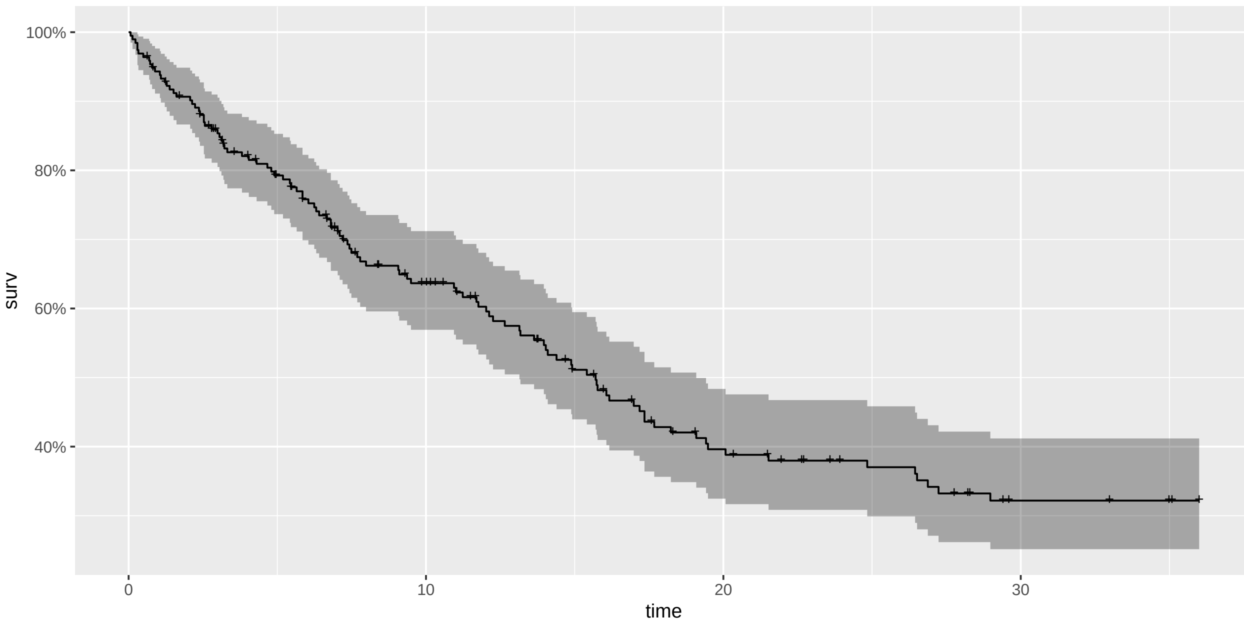
The survival probabilities are as follows, and these change over time at the times displayed on the *time* column of this table:



Exercise 4

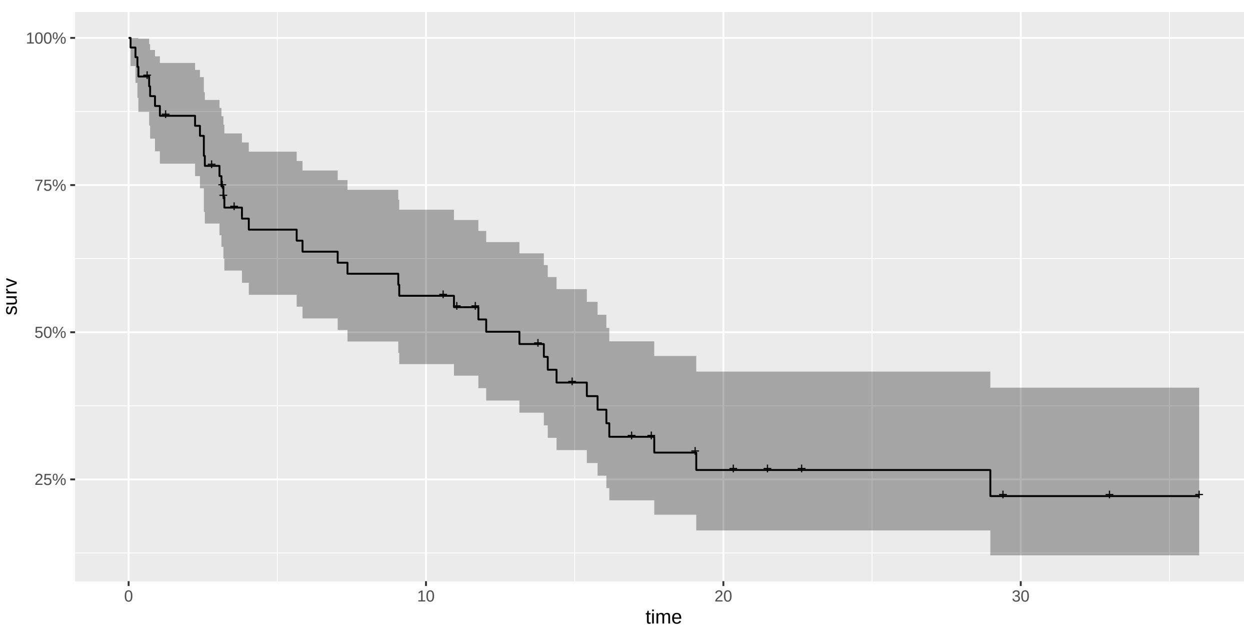
KM estimate of the survival function

We can see the estimate as follows:

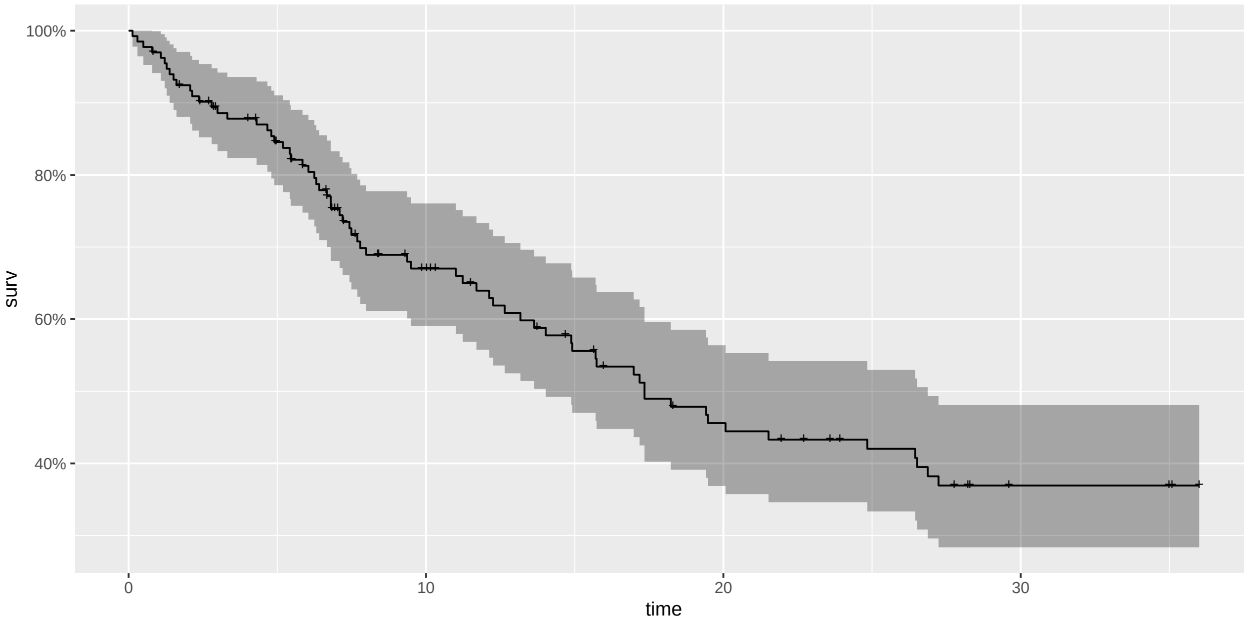


In order to achieve this fit, we have modified the dataset as to convert the columns *personal* and *property* to factors, and then switch the 0s for 1s and vice versa in the *censor* column, given that these are reversed.

Survival function: with crimes against persons

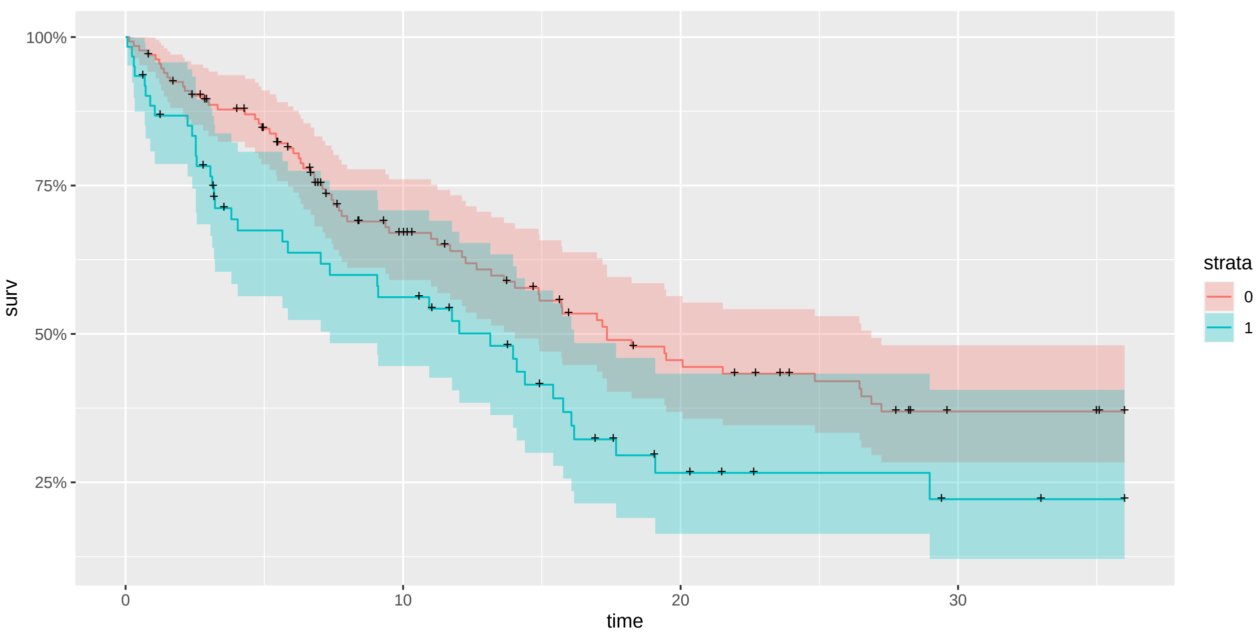


Survival function: without crimes against persons



Comparing both curves

We can see that the curve for nonpersonal crimes decays faster overall, as opposed to the personal crimes.



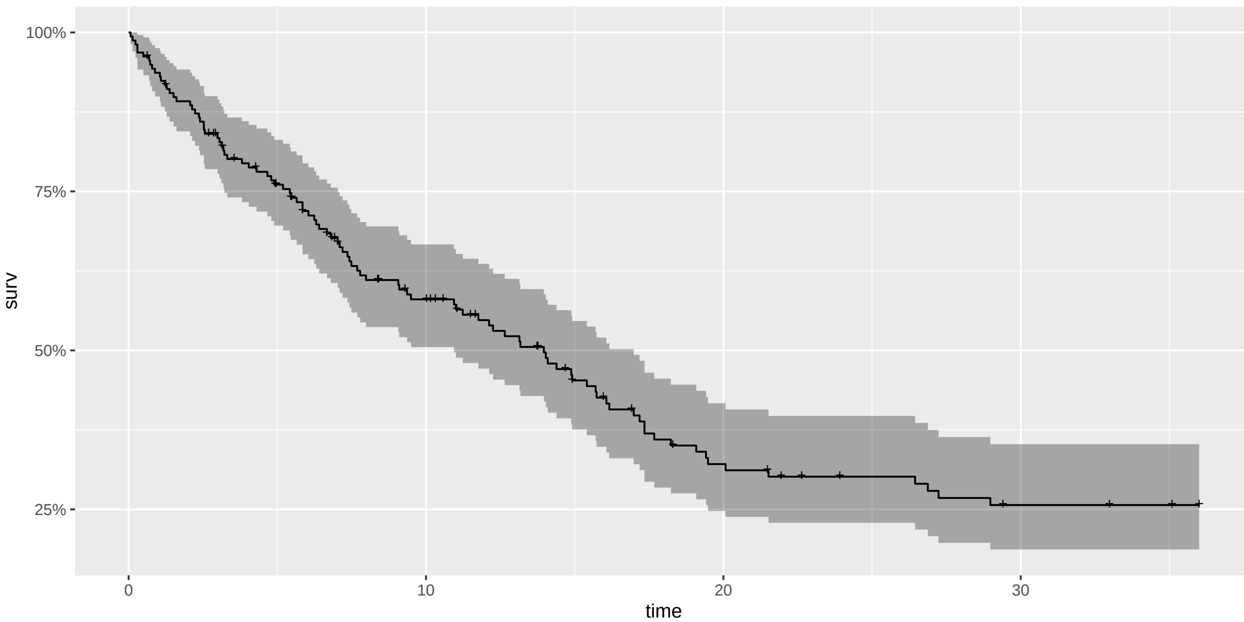
Low-rank test

Acording

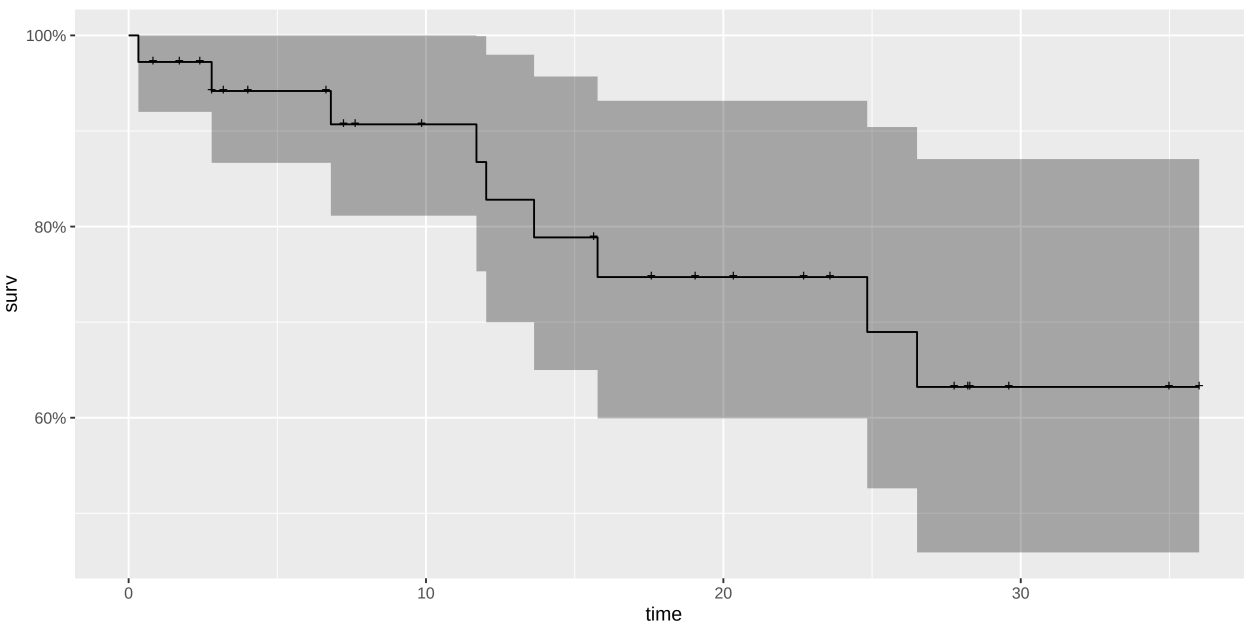
```
#> Call:
#> survdiff(formula = Surv(months, censor) ~ personal, data = henning)
#>
#>              N Observed Expected (O-E)^2/E (O-E)^2/V
```

```
#> personal=0 133       67     77.8      1.50       5.7
#> personal=1  61       39     28.2      4.14       5.7
#>
#>  Chisq= 5.7  on 1 degrees of freedom, p= 0.02
```

Survival function: with crimes against property

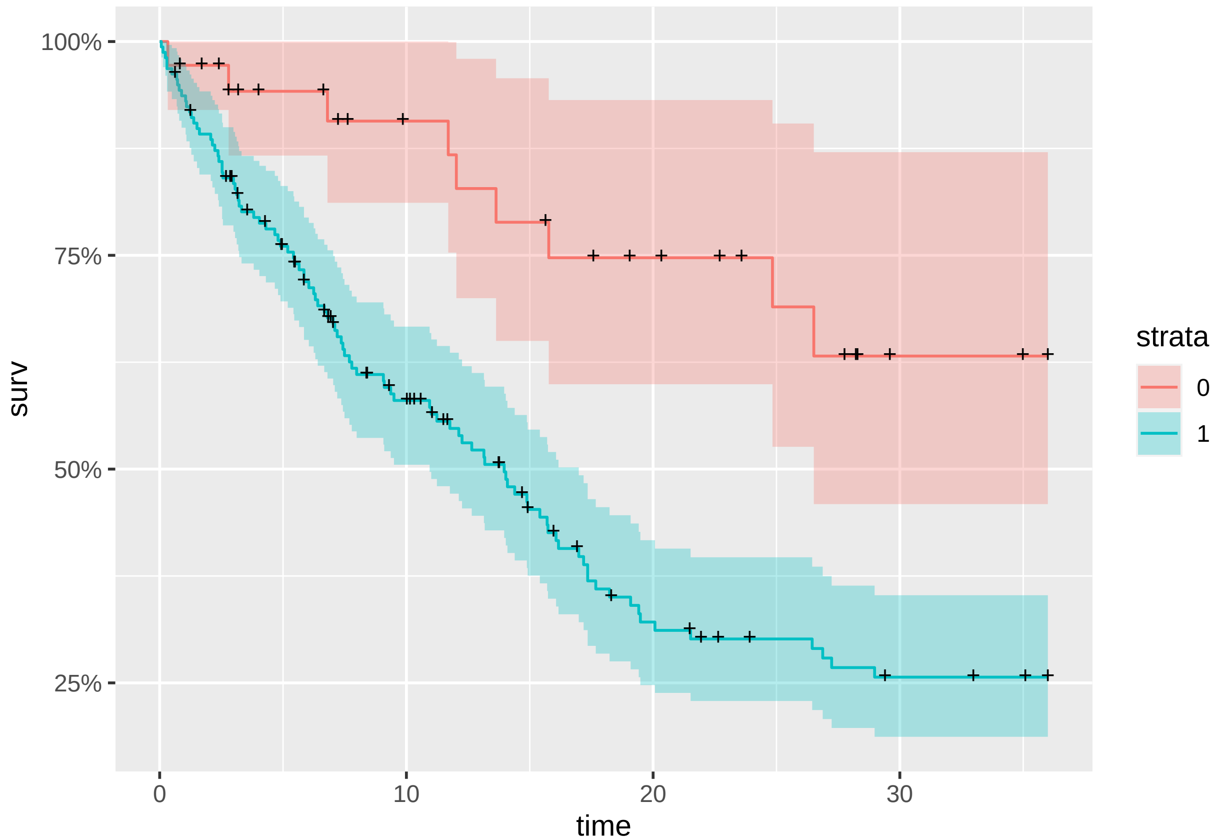


Survival function: with crimes against property



Comparing both curves

We can see that the curve for non-property related crimes overall decays significantly faster than the opposite one.



Low-rank test

```
#> Call:
#> survdiff(formula = Surv(months, censor) ~ property, data = henning)
#>
#>              N Observed Expected (O-E)^2/E (O-E)^2/V
#> property=0  36        9     24.7      9.97      13.1
#> property=1 158       97     81.3      3.02      13.1
#>
#>  Chisq= 13.1  on 1 degrees of freedom, p= 3e-04
```

Fitting a Cox regression

Converting personal and property to leveled factors with labels yes/no.

```
#>   id     months censor personal property       cage
#> 1  1 0.06570842      1      yes      yes  -1.675198
#> 2  2 0.13141684      1       no      yes -10.482864
#> 3  3 0.22997947      1      yes      yes  -4.426738
#> 4  4 0.29568789      1       no      yes -11.328860
#> 5  5 0.29568789      1      yes      yes  -7.164589
#> 6  6 0.32854209      1      yes       no  -2.868901
```

Running the cox regression fit.

```
#> Call:
#> coxph(formula = Surv(months, censor) ~ cage + personal + property,
#>     data = henning)
#>
#>   n= 194, number of events= 106
#>
#>                 coef exp(coef) se(coef)      z Pr(>|z|)
```

```
#> cage        -0.06671   0.93546  0.01678 -3.976 7.01e-05 ***
#> personalyes  0.56914   1.76674  0.20521  2.773  0.00555 **
#> propertyyes  0.93579   2.54922  0.35088  2.667  0.00765 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>             exp(coef) exp(-coef) lower .95 upper .95
#> cage           0.9355     1.0690    0.9052    0.9667
#> personalyes    1.7667     0.5660    1.1817    2.6415
#> propertyyes    2.5492     0.3923    1.2816    5.0708
#>
#> Concordance= 0.694  (se = 0.027 )
#> Likelihood ratio test= 38.96  on 3 df,   p=2e-08
#> Wald test            = 29.02  on 3 df,   p=2e-06
#> Score (logrank) test = 30.3  on 3 df,   p=1e-06
```

According to the p-value on our Wald test, we can see that

The dummy variables personal and property are not significant, given that they have a large p-val. The variable cage, which is centered age (in years) at time of release is significant, and positive, which means that it increases the probability of survival as it increases.

Exercise 5

Given a hazard function $h(t) = c$, where $c > 0$:

We obtain the cumulative hazard function $H(t)$:

$$\begin{aligned} H(t) &= \int_0^t h(u)du \\ &= c\int_0^t du \\ &= ct \end{aligned}$$

With this, we derive the survival function $S(t)$:

$$\begin{aligned} H(t) &= ct \\ H(t) &= -log(S(t)) \\ ct &= -log(S(t)) \\ S(t) &= e^{-ct} \end{aligned}$$

And then we obtain the density function $f(t)$:

$$\begin{aligned} h(t) &= \frac{f(t)}{S(t)} \\ c &= \frac{f(t)}{e^{-ct}} \\ f(t) &= ce^{-ct} \end{aligned}$$

Calculating mean failure time with $c = 5$

We note the functions with $c = 5$ are:

$$
\begin{aligned}
h(t) &= 5 \\
H(t) &= 5t \\
S(t) &= e^{-5t} \\
f(t) &= 5e^{-5t}
\end{aligned}
$$

The mean failure time, according to 100,000 simulations with a length of time of 1000, is as follows:

```
#> [1] 1.007
```

Exercise 6

First we read the data:

```
#>   inst time status age sex ECOG Karnofsky.physician Karnofsky.patient calories
#> 1    3  306      2  74   1    1                  90               100     1175
#> 2    3  455      2  68   1    0                  90                90     1225
#> 3    3 1010      1  56   1    0                  90                90        .
#> 4    5  210      2  57   1    1                  90                60     1150
#> 5    1  883      2  60   1    0                 100                90        .
#> 6   12 1022      1  74   1    1                  50                80      513
#>   weight.loss
#> 1           .
#> 2          15
#> 3          15
#> 4          11
#> 5           0
#> 6           0
```