Introduction to Survival Analysis

Modeling Medical Time-to-Event Data Using R Software Packages

Kristen Varin

2023-09-06

Table of contents

1 Survival Analysis

1.1 Survival Analysis Background

Survival analysis is a type of statistical analysis used for analyzing the relationship between time and an event of interest occurring for individuals. In medical applications, the most common event analyzed is death, but many other events can be analyzed such as time it takes to begin recovering from treatment or time until a disease is contracted. These analyses are specifically helpful when comparing groups of people such as treatment groups in a clinical trial. Survival analysis can reveal whether certain treatments are more effective than others in helping individuals to live longer or avoid certain outcomes of interest, such as heart attacks.

Survival analysis can be used for any time-to-event data, not just medical data. Some of these disciplines include: Epidemiology, Finance, Engineering, Marketing, Insurance, and more. For example, in marketing, survival analysis can be used to predict how long a customer will remain a customer. In epidemiology, it can be used for predicting time until disease recurrence. In engineering, survival analysis can be used to predict how long a machine will last. We will focus on medical applications in this paper. Let's begin by creating a simple data set.

1.2 Example data set in R

To demonstrate survival analysis, an example data set was created including ten observations with the following columns:

- *id*: number of each observation ranging from 1 to 10
- *time*: time variable ranging from 1 to 10
- *status*: 1 if the event occurs at that time for that individual, 0 if no event occurs

For those assigned a 1, a survival time was assigned in the range of 1 to 9, representing the time that individual survived until the event occurred for them. For those assigned a 0, a time of $t = 10$ was assigned, suggesting that they lasted until the end of the hypothetical study without the event occurring. Table 1.1 shows these data.

```
# Load Packages
library(tidyverse) |> suppressPackageStartupMessages()
```

```r
library(knitr)
library(survival)
library(ggsurvfit)
library(gt)
```

```r
# Create a simple data set
id <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
time <- c(1, 2, 4, 5, 6, 8, 9, 10, 10, 10)
status <- c(1, 1, 1, 1, 1, 1, 1, 0, 0, 0)
surv <- data.frame(id, time, status)
```

```r
surv %>% gt(caption = "Example Data Set with ID, Status, and Time") %>%
  cols_label(id = "ID", time = "Time", status = "Status")
```

Table 1.1: Example Data Set with ID, Status, and Time

ID	Time	Status
1	1	1
2	2	1
3	4	1
4	5	1
5	6	1
6	8	1
7	9	1
8	10	0
9	10	0
10	10	0

To perform survival analysis on these data, the probability of survival at each time from $t = 0$ to $t = 10$ can be calculated by dividing the amount of people surviving until that time by the total amount of people in the study. For the first observation in the example data, probability of survival is equal to 1, or 100%, because every participant would presumably begin the study alive. To calculate the probability of survival for time $t = 1$, the number of participants that did not experience the outcome of interest by $t = 1$ would be divided by 10, the total number of observations in the study. In this case, 9 participants survived until time $t = 1$, because only one observation was assigned a 1 between times $t = 0$ and $t = 1$. Dividing this total by the 10 total observations in the study shows that the probability of surviving to time 1 is 0.9, or 90%. As time increases, the total probability of survival for the group will decrease in the range of 1 to 0 because more people will be experiencing the outcome. This is why survival curves typically have a rightly skewed distribution. Table 1.2 shows the survival probabilities

for each observation.

```r
# Calculate survival probabilities
probabilities <- data.frame(
  Number_alive = c(10, 9, 8, 8, 7, 6, 5, 4, 3, 3, 3),
  Time = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10),
  Probability =  c(1, 0.9, 0.8, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.3, 0.3))
```

```r
probabilities %>% gt(caption = "Probability of Survival Based on Time") %>%
  cols_label(Number_alive = "Number alive",
             Time = "Time", Probability = "Probability")
```

Table 1.2: Probability of Survival Based on Time

Number alive	Time	Probability
10	0	1.0
9	1	0.9
8	2	0.8
8	3	0.8
7	4	0.7
6	5	0.6
5	6	0.5
4	7	0.4
3	8	0.3
3	9	0.3
3	10	0.3

1.3 Censoring Background

One issue with survival analysis data is the problem of an individual being lost to follow-up, meaning data could not be collected for them at some point during the study. This causes the outcome of interest to not be recorded for that individual, so the survival time for that individual can not be analyzed as it would not be accurate. When this happens, the survival times for those individuals are recorded as *censored*, causing standard analysis techniques to be inappropriate for these data. There are three types of censoring that can occur. The first type of censoring is called *right censoring*, which happens if an individual begins the study but then is lost to follow-up at some point during the study before it ends. The censored survival time for these individuals is thus equal to the total time they were known to be alive in the study's observation period before they were lost to follow-up. Some common examples of when right censoring is needed include: an individual moving away from the study and not being

able to participate or when an individual dies due to a non-related event after the study begins. Another type of censoring is called *left censoring*. In this type, the individual experiences lost to follow-up before the observation period begins. This would happen, for example, in a study that tracks patient recovery from a surgery, but with an observation period beginning one a month after the surgery took place. If a patient died less than a month after surgery, their survival time would need to be left censored. The final type of censoring is called *interval censoring*, which happens when a patient comes in and out of the study, making it possible for them to experience the outcome of interest during a period of time when they aren't being observed. This often happens when recurrence is being tracked in a study. One example could be if recurrence of cancer is being tracked and the study checks in with patients every month. If an individual does not have cancer after the first month but then does after the second month, the recurrence time is somewhere between one and two months, and it therefore needs to be interval censored (Collet (2003)). Out of the three types, right censoring is the most common and will be demonstrated in the next example data-set used in section 2.3.

1.4 Paper Outline

The rest of the paper will be divided into the following sections:

1. *Kaplan Meier Survival Curves*: This section will introduce how to model survival probabilities using the Kaplan Meier survival estimate. It will walk through how to plot survival curves in R, how to run a Log-Rank test for difference in survival curves, and then two sets of case study data.
2. *Cox Proportional Hazards Regression*: This section will introduce modeling survival hazard rates using Cox Proportional Hazards Regression. It will walk through how to run a Cox Proportional Hazards Regression in R, how to interpret the output, and then applying the ideas to the same two case studies.
3. *Discussion*: This section will summarize the paper findings.

1.5 References

2 Kaplan Meier

2.1 Kaplan Meier Survival Estimate

The Kaplan Meier survival estimate is how survival probabilities can be calculated while factoring in censored times. The Kaplan Meier survival estimate is used to calculate probability of survival at a given time where $S(t_j)$ is the probability of being alive at time t_j, $S(t_{j-1})$ is the probability of being alive at t_{j-1}, n_j is the number of patients alive just before t_j, d_j is the number of events at t_j, and j is the time interval of interest. The equation is $S(t_j) = S(t_{j-1})(1 - \frac{d_j}{n_j})$ (Clark (2003)). The equation essentially divides the surviving individuals by the individuals at risk, similar to the previous calculations shown. However, Kaplan Meier curves adjust for right-censored times by dropping observations from the total number of individuals at risk, n_j, after their censored time has been reached. This adjustment prevents overestimating the survival probability because it no longer assumes censored individuals are still alive or and at risk (Goel (2010)).

2.2 Descriptive Statistics and the Kaplan Meier Curve

Summary statistics can be used to understand Kaplan Meier survival estimates. For example, a 95% confidence interval for the survival probability can be found for each time in the data set using the formula $S(t_j) \pm 1.96 * SE(S(t_j))$ (Sullivan (2016b)). This interval will output a range that, with 95% confidence, contains the true survival probability for an individual in the data set at that time. Another commonly reported descriptive statistic is the median of the survival function, $S(t_j)$. The median value is the time t at which $S(t_j) = 0.5$, or when half of the individuals have not yet experienced the outcome (Rao (2023)).

The Kaplan Meier curve is a graphical representation of the survival function. Similar to the survival probabilities discussed in Section 1.2, the Kaplan Meier curve shows the relationships between time, which is typically plotted on the x-axis, and probability of survival, which is typically on the y-axis. The curve always ranges from 0 to 1 and is typically right skewed.

2.3 Kaplan Meier in R

Luckily, modern software makes these calculations easy and fast, as well as plotting them with confidence intervals and risk tables. A new data set with censored times will be created to demonstrate this process. To do this, a 0 will be recorded for some individuals at times before t = 10. Table 2.1 shows this data set's structure.

```
# Load Packages
library(tidyverse) |> suppressPackageStartupMessages()
library(knitr)
library(survival)
library(ggsurvfit)
library(survminer) |> suppressPackageStartupMessages()
library(gt)
```

```
# Create data set
id <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
time <- c(1, 2, 10, 4, 5, 6, 10, 8, 9, 10)
status <- c(1, 1, 0, 0, 1, 0, 0, 1, 1, 0)
censor <- data.frame(id, time, status)
```

```
censor %>% gt(caption = "Example Data Set with Censored Times") %>%
  cols_label(id = "ID", time = "Time", status = "Status")
```

Table 2.1: Example Data Set with Censored Times

ID	Time	Status
1	1	1
2	2	1
3	10	0
4	4	0
5	5	1
6	6	0
7	10	0
8	8	1
9	9	1
10	10	0

The `survival` package in `R` has a function called `Surv()` that takes input data and creates a response object recording survival time for each observation. The function takes in the

time variable and the status variable. The function takes into account right censoring as well, marking censored times with a + symbol in the object created (Zabor (2023)).

```
# Show the time of event or censored time for each observation
times <- Surv(censor$time, censor$status)
times
```

```
 [1]  1   2  10+  4+  5   6+ 10+  8   9  10+
```

The `survfit()` function can be used to calculate the Kaplan Meier survival estimate for each time that a new event occurs. The function takes in the response object created by the `Surv()` function in order to drop censored observations from the regression. We use 'times ~ 1' because we are not including predictor variables in the model. The function returns a survival object that can be used to plot the Kaplan Meier curve and calculate summary statistics. The `time` variable in the object created by the `survfit()` function shows the time of each event, and the `surv` variable shows the survival probability for the remaining individuals after each event occurs.

```
# Calculate survival object
s1 <- survfit(times ~ 1, data = censor)
# View the survival time for each time an event occurs
s1$time
```

```
[1]  1  2  4  5  6  8  9 10
```

```
# View survival probability for remaining individuals after each event occurs
round(s1$surv, 2)
```

```
[1] 0.90 0.80 0.80 0.69 0.69 0.55 0.41 0.41
```

The `ggsurvfit` package can be used to plot the Kaplan Meier curve for this data using the previously created `s1` object. Figure 2.1 shows the Kaplan Meier curve for predicting the event. The risk table shows the number of individuals at risk at each time point and the number of events that occurred at each time point.

```
s1 %>%
  ggsurvfit() +
  labs(
    x = "Time",
```

```
  y = "Overall survival probability",
  title = "Kaplan Meier Survival Curve"
) +
scale_x_continuous(breaks=seq(0,10,by=2)) +
scale_y_continuous(breaks=seq(0,1,by=.2)) +
add_risktable()
```

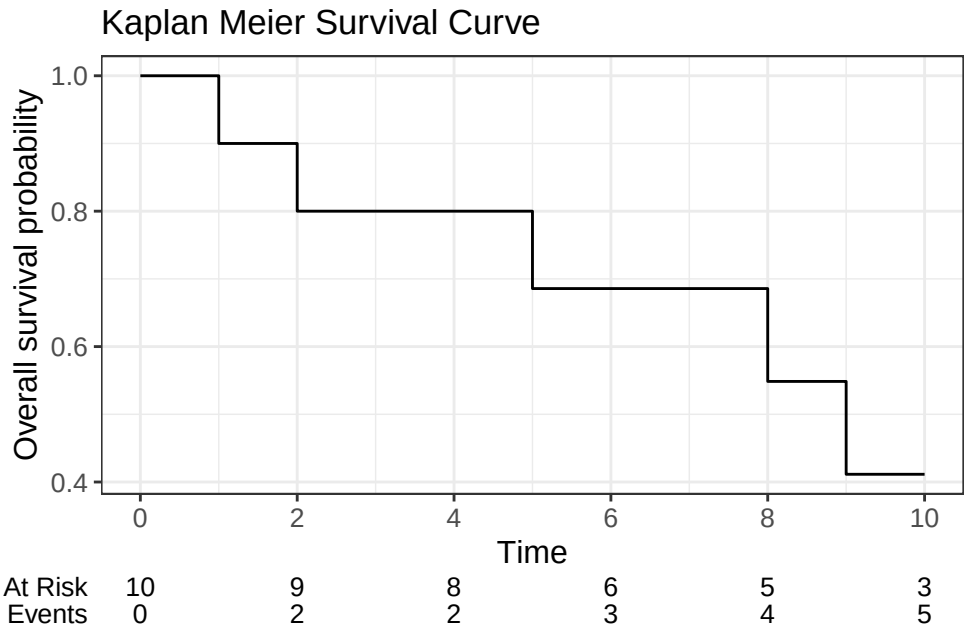

Figure 2.1: Kaplan Meier Survival Curve

2.4 Log-Rank Test

One of the most common applications of survival analysis and Kaplan Meier curves is for comparing survival times between two groups. One example of when this might be useful is if a study is comparing survival after two different treatment plans. A data set with two groups will be created to demonstrate this process.

```
# Create new data set
time <- c(8, 7, 10, 8, 5, 3, 4, 10, 6, 1)
status <- c(0, 1, 0, 1, 0, 1, 0, 0, 1, 1)
group <- c(1, 1, 1, 1, 1, 2, 2 ,2, 2, 2)
```

```r
surv2 <- data.frame(time, status, group)
```

To run an analysis, the statistical test called the Log-Rank Test can be used to test the null hypothesis that there is no difference between the survival estimates of two groups at any point in time (Rich (2010)). The test is conducted by comparing the observed number of events with an estimated number of events for each group at each time. To calculate expected number of events, the assumption that the two curves are identical is used. The expected number of events at a time t is for group i calculated by the formula $E_{it} = \frac{N_{it} \times O_t}{N_t}$, where N_{it} is the observed number of events in group i at time t, O_t is the total number of events in all groups at time t, and N_t is the total number observations at risk in all groups at time t (Sullivan (2016a)). Table 2.2 shows the creation of all of these values. The following variables are in the table:

- $Time$: time when the event occurs
- N_{1t}: the number of people alive and still in the study (at risk), at time t in group 1
- N_{2t}: the number at risk at time t in group 2
- N_t: the total number of people at risk in the study at time t
- O_{1t}: the number of observed events occurring at time t in group 1
- O_{2t}: the number of observed events occurring at time t in group 2
- O_t: the total observed events across both groups at time t
- E_{1t}: the expected number of events at time t in group 1
- E_{2t}: the expected number of events at time t in group 2
- Et: the total expected number of events across both groups at time t

```r
Time <- c(1, 3, 4, 5, 6, 7, 8, 10)
N1t <- c(5, 5, 5, 5, 4, 4, 3, 1)
N2t <- c(5, 4, 3, 2, 2, 1, 1, 1)
O1t <- c(0, 0, 0, 0, 0, 1, 1, 0)
O2t <- c(1, 1, 0, 0, 1, 0, 0, 0)

log_rank <- data.frame(Time, N1t, N2t, O1t, O2t)

# Calculate Totals
log_rank$Nt <- log_rank$N1t + log_rank$N2t
log_rank$Ot <- log_rank$O1t + log_rank$O2t
log_rank$E1t <- (log_rank$N1t * log_rank$Ot) / log_rank$Nt
log_rank$E2t <- (log_rank$N2t * log_rank$Ot) / log_rank$Nt

round(log_rank, 2) %>%
  gt() %>%
  tab_header("Values Needed to Calculate Chi-Square Statistic")
```

Table 2.2: Calculation of Log-Rank Test Statistic

Values Needed to Calculate Chi-Square Statistic

Time	N1t	N2t	O1t	O2t	Nt	Ot	E1t	E2t
1	5	5	0	1	10	1	0.50	0.50
3	5	4	0	1	9	1	0.56	0.44
4	5	3	0	0	8	0	0.00	0.00
5	5	2	0	0	7	0	0.00	0.00
6	4	2	0	1	6	1	0.67	0.33
7	4	1	1	0	5	1	0.80	0.20
8	3	1	1	0	4	1	0.75	0.25
10	1	1	0	0	2	0	0.00	0.00

To test the null hypothesis that there is no difference between the survival estimates of two groups at any point in time, a test statistics is needed. For the Log-Rank test, a Chi-Square test statistic is used because the data follows a Chi-Square curve rather than a normal distribution. The Chi-Square test statistic is calculated by the formula $X^2 = \sum_{i=1}^{k} \frac{(\sum O_{it} - \sum E_{it})^2}{\sum Vi}$, where $\sum_0^T O_{it}$ is the sum of the observed number of events in group i over the entire time interval and $\sum_0^T E_{it}$ is the sum of the expected number of events in group i over the entire time interval. The difference of the sums of these two values are divided by the sum of the variance. The variance for group 1 at time t is calculated by the formula $V_{1t} = \frac{N_{1t} \times N_{2t} \times O_t \times (N_t - O_t)}{N_t^2 \times (N_t - 1)}$ (Collet (2003)). The variance for group 2 is calculated similarly, and the total variance is calculated by the formula $V_t = V_{1t} + V_{2t}$. Table 2.3 shows the calculation of the Chi-Square test statistic for the data set. Two new variables are calculated from the previous table:

- V_{1t}: the variance of the number of events at time t in group 1
- V_{2t}: the variance of the number of events at time t in group 2

```
log_rank$V1t <- (log_rank$N1t * log_rank$N2t * log_rank$Ot *
                   (log_rank$Nt - log_rank$Ot)) /
  (log_rank$Nt^2 * (log_rank$Nt - 1))

log_rank$V2t <- (log_rank$N1t * log_rank$N2t * log_rank$Ot *
                   (log_rank$Nt - log_rank$Ot)) /
  (log_rank$Nt^2 * (log_rank$Nt - 1))

round(log_rank, 2) %>%
  gt() %>%
  tab_header("Calculation of Variance")
```

Table 2.3: Calculation of Variance

Calculation of Variance

Time	N1t	N2t	O1t	O2t	Nt	Ot	E1t	E2t	V1t	V2t
1	5	5	0	1	10	1	0.50	0.50	0.25	0.25
3	5	4	0	1	9	1	0.56	0.44	0.25	0.25
4	5	3	0	0	8	0	0.00	0.00	0.00	0.00
5	5	2	0	0	7	0	0.00	0.00	0.00	0.00
6	4	2	0	1	6	1	0.67	0.33	0.22	0.22
7	4	1	1	0	5	1	0.80	0.20	0.16	0.16
8	3	1	1	0	4	1	0.75	0.25	0.19	0.19
10	1	1	0	0	2	0	0.00	0.00	0.00	0.00

For group 1 at time 1, we would calculate $E_{1t} = \frac{5\times1}{10} = 0.50$ and for group 2 at time 1, $E_{2t} = \frac{5\times1}{10} = 0.50$. We would repeat for each group at each time and then sum the values to get $\sum_{t=1}^{t} E_i$ for each group. For group 1, $\sum_{t=1}^{t} E_1 = 0.50+0.56+0+0+0.67+0.80+0.75+0 = 3.27$ and for group 2, $\sum_{t=0}^{t} E_2 = 0.50 + 0.44 + 0 + 0 + 0.33 + 0.20 + 0.25 + 0 = 1.73$. The sum of the observed events for group 1 is 2 and for group 2 is 3. For group 1 at time 1, we would calculate $V_1t = \frac{5\times5\times1\times(10-1)}{10^2\times(10-1)} = 0.25$ Table 2.4 shows the rest of the variances calculated for both groups for the `surv2` data set. The table includes the following new variables:

- $SumO_{1t}$: the sum of the observed number of events in group 1 over the entire time interval
- $SumO_{2t}$: the sum of the observed number of events in group 2 over the entire time interval
- $SumE_{1t}$: the sum of the expected number of events in group 1 over the entire time interval
- $SumE_{2t}$: the sum of the expected number of events in group 2 over the entire time interval
- $SumV_{1t}$: the sum of the variance of the number of events at time t in group 1
- $SumV_{2t}$: the sum of the variance of the number of events at time t in group 2
- X_1: the Chi-Square test statistic for group 1
- X_2: the Chi-Square test statistic for group 2

```
SumO1t <- cumsum(log_rank$O1t)
SumO2t <- cumsum(log_rank$O2t)
SumE1t <- cumsum(log_rank$E1t)
SumE2t <- cumsum(log_rank$E2t)
SumV1t <- cumsum(log_rank$V1t)
SumV2t <- cumsum(log_rank$V2t)
```

```r
# Create table with each of the last values
sum_stats <- data.frame(SumO1t, SumO2t,
                        SumE1t, SumE2t, SumV1t, SumV2t)
# Get the last row of summary statistics
sum_stats <- sum_stats[nrow(sum_stats), ]
sum_stats$X1 <- ((sum_stats$SumO1t - sum_stats$SumE1t)^2 /
                           sum_stats$SumV1t)
sum_stats$X2 <- ((sum_stats$SumO2t - sum_stats$SumE2t)^2 /
                           sum_stats$SumV2t)

round(sum_stats, 2) %>%
  gt() %>%
  tab_header("Calculation of Chi Square Statistic")
```

Table 2.4: Calculation of Chi Square Statistic

Calculation of Chi Square Statistic

SumO1t	SumO2t	SumE1t	SumE2t	SumV1t	SumV2t	X1	X2
2	3	3.27	1.73	1.07	1.07	1.52	1.52

With all of these values, we can calculate the Chi Square test statistic, $X_i^2 = \frac{(\sum O_{it} - \sum E_{it})^2}{V_{it}}$. We get $X_1^2 = \frac{(2-3.27)^2}{1.07} = 1.51$ and $X_2^2 = \frac{(3-1.73)^2}{1.07} = 1.51$. The test statistic can then be compared to a Chi-Square distribution with one $k-1$ degrees of freedom, with k being the number of groups. So, for this example, k is 2 and there is 1 degree of freedom (Sullivan (2016a)).

Clearly, calculating the test statistic is very tedious, even for a data set with 10 observations. The `survdiff` function can be used to run a Log-Rank test in R, making it much easier to run the test. The function takes in the response object created by the `Surv()` function and the grouping variable (Zabor (2023)). It returns a Chi-Square statistic and a p-value. We can see that the p-value is much higher than 0.05, indicating that we do not have evidence to reject the null hypothesis, and thus that there is no difference between the survival estimates of the two groups.

```r
# Calculate times for events in this data set
times2 <- Surv(surv2$time, surv2$status)
# Run a log rank test based on group variable
test_stat <- survdiff(times2 ~ group, data = surv2)
test_stat
```

```
Call:
```

```
survdiff(formula = times2 ~ group, data = surv2)

        N Observed Expected (O-E)^2/E (O-E)^2/V
group=1 5        2     3.27     0.495      1.52
group=2 5        3     1.73     0.937      1.52

 Chisq= 1.5  on 1 degrees of freedom, p= 0.2
```

The only adjustment needed for plotting the Kaplan Meier curves for two groups is to include the grouping variable while making the survival object using the `survfit` function (Zabor (2023)). The two survival curves are depicted in Figure 2.2.

```
# Create survival object using group as a predictor
s2 <- survfit(Surv(time, status) ~ group, data = surv2)

# Plot the two curves
s2 %>%
  ggsurvfit() +
  labs(
    x = "Days",
    y = "Overall survival probability",
    title = "Kaplan Meier Survival Curve by Group"
  ) +
  scale_x_continuous(breaks=seq(0,10,by=2)) +
  scale_y_continuous(breaks=seq(0,1,by=.2)) +
  add_risktable()
```

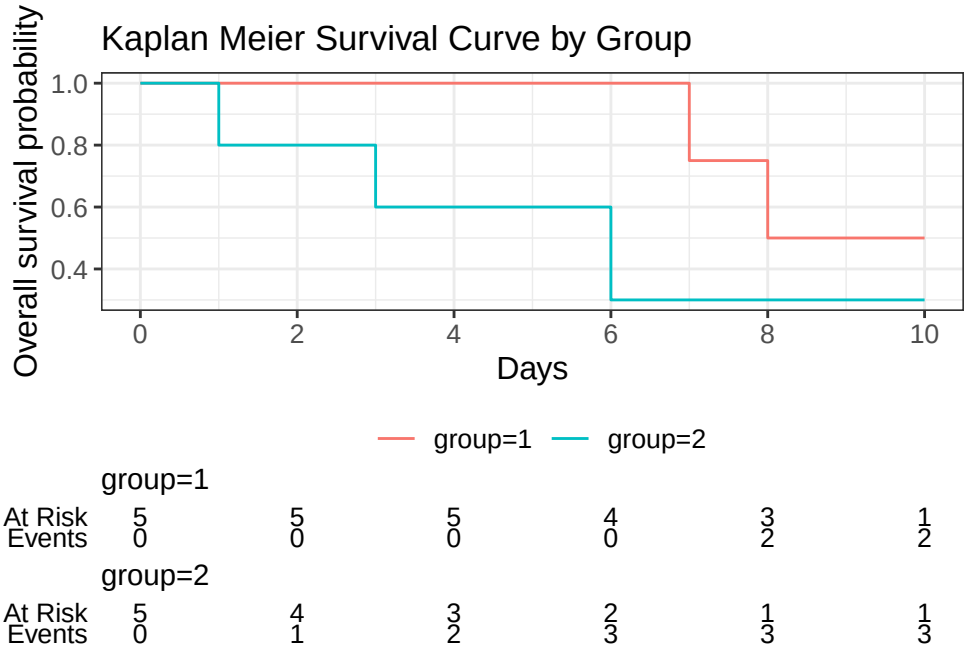

Figure 2.2: Kaplan Meier Survival Curve by Group

2.5 Case Study: Cirrhosis Data

The next example uses a data set with variables for predicting death in patients with cirrhosis, which is permanent scarring of the liver. The data set is from a clinical trial conducted by the Mayo Clinic from 1974 to 1984. The data set contains 424 primary biliary cirrhosis patients with 20 variables (Fedesoriano (2021)). To conduct survival analysis, the `Status` variable needs to be transformed into an indicator variable, labelled `event`, coded as a 1 representing death and 0 representing censored. The `N_Days` variable will be used for the time variable, indicating number of days since the beginning of the trial.

```
cirrhosis <- read_csv("data/cirrhosis.csv", show_col_types = FALSE)
cirrhosis <- cirrhosis %>% mutate(event = if_else(Status == 'D', 1, 0))
```

The main interest of this data is to explore the difference in survival time between patients taking the drug of interest, D-penicillamine, and those given a placebo. The variable `Drug` indicates which group the patient was in and will be used as the grouping variable for a Log-Rank Test. Below, we calculate an object with times of event for this data and then use the `survdiff` function to run a Log-Rank Test.

```r
# Calculate times and store in an object
times3 <- Surv(cirrhosis$N_Days, cirrhosis$event)

# Run a log rank test based on drug group
survdiff(times3 ~ Drug, data = cirrhosis)
```

```
Call:
survdiff(formula = times3 ~ Drug, data = cirrhosis)

n=312, 106 observations deleted due to missingness.

                      N Observed Expected (O-E)^2/E (O-E)^2/V
Drug=D-penicillamine 158       65     63.2    0.0502     0.102
Drug=Placebo         154       60     61.8    0.0513     0.102

 Chisq= 0.1  on 1 degrees of freedom, p= 0.7
```

From the Log-Rank test, we see that the p-value is 0.7, which is much larger than 0.05. This indicates that the test was not statistically significant at the five percent level. Thus, there was no statistically significant difference between patient outcome between the two treatment groups. So, for patients with biliary cirrhosis, we did not find evidence that D-penicillamine is an effective drug for preventing death.

We can visualize this in Figure 2.3, which shows the survival curves for both groups of patients. We can also use the `ggsurvplot` function from the `survminer()` package to plot the survival curves. The curves are very similar and cross multiple times, which makes sense since we know the drug of interest did not have a significant effect on patient survival.

```r
# Create Survival Object
s3 <- survfit(times3 ~ Drug, data = cirrhosis)

# Plot the survival curves
s3 %>% ggsurvplot(
    palette = c("darkgreen", "maroon"),
    legend.labs = c("D-penicillamine", "Placebo"),
    xlab = "Time",
    ylab = "Survival Probability",
    title = "Kaplan Meier Survival Curve by Drug Group"
  )
```

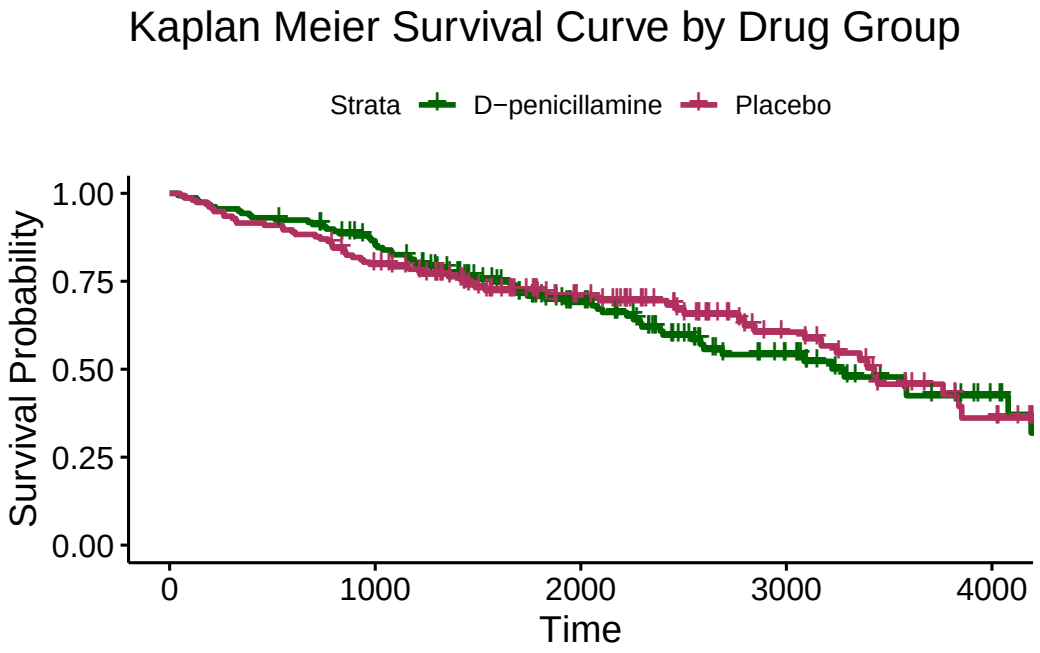

Figure 2.3: Kaplan Meier Survival Curve Predicting Death from Cirrhosis

As we can see, the two survival curves are very similar and cross multiple times, which makes sense since we know the drug of interest did not have a significant effect on patient survival.

2.6 Case Study: Heart Failure Data

Let's look at another data set which contains information on patients with heart failure. The data set contains data on heart failure patients over 40 years old who were admitted to the Institute of Cardiology at the Allied hospital Faisalabad-Pakistan between April and December of 2015. All of the patients in the data set had left ventricular systolic dysfunction and belonged to NYHA class III and IV stages of heart failure (Ahmad (2017)).

```
heart <- read_csv("data/S1Data.csv")
```

```
Rows: 299 Columns: 13
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (13): TIME, Event, Gender, Smoking, Diabetes, BP, Anaemia, Age, Ejection...

i Use `spec()` to retrieve the full column specification for this data.
```

```
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Let's view the Kaplan Meier survival curves to see if there is a visual difference in survival probabilities for individuals with normal versus high blood pressure. In this data set, blood pressure is a binary variable where 0 indicates normal blood pressure and 1 indicates high blood pressure. It is called BP in the data set. We can view the curve in Figure 2.4.

```
# Create Survival Object
times_heart <- Surv(heart$TIME, heart$Event)

# Plot the curves
survfit(times_heart ~ BP, data = heart) %>%
  ggsurvplot(
    palette = c("darkgreen", "maroon"),
    legend.labs = c("Normal BP", "High BP"),
    xlab = "Time (Days)",
    ylab = "Survival Probability",
    title = "Kaplan Meier Survival Curve by Blood Pressure"
  )
```

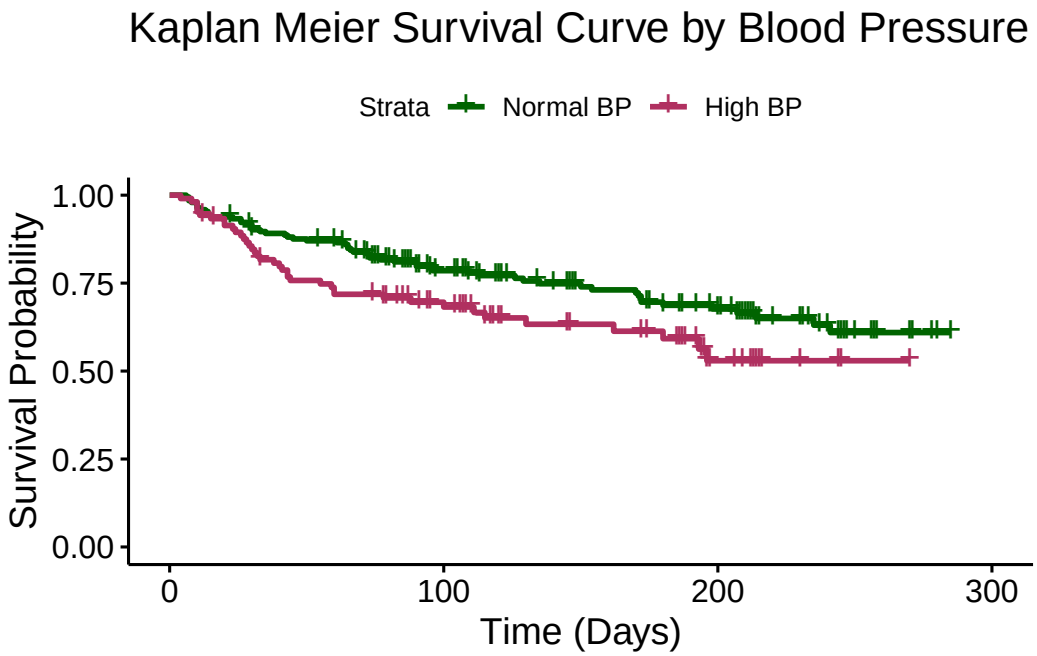

Figure 2.4: Kaplan Meier Survival Curve Predicting Death from Blood Pressure

From the curves, it seems like there will be a difference in survival probabilities between

individuals with normal and high blood pressure based on the two curves. Let's run a Log-Rank test to see if the difference is statistically significant.

```
# Run a log rank test based on blood pressure group
survdiff(times_heart ~ BP, data = heart)
```

```
Call:
survdiff(formula = times_heart ~ BP, data = heart)

       N Observed Expected (O-E)^2/E (O-E)^2/V
BP=0 194       57     66.4      1.34      4.41
BP=1 105       39     29.6      3.00      4.41

 Chisq= 4.4  on 1 degrees of freedom, p= 0.04
```

The p-value from the Log-Rank test is 0.04, so we can conclude that blood pressure is a significant predictor of survival probability in heart failure patients.

2.7 Limitations

It is important to note the limitations of this analysis. First, the Kaplan Meier function does not allow for the use of continuous variables or multiple predictors in the model. Instead, it is limited to one categorical predictor, such as blood pressure in our cirrhosis model. Additionally, the Log-Rank test only tells us whether there is a difference in survival probabilities, but it does not tell us how big this difference is. So, to gain more understanding on how big the effect of blood pressure is on patient survival, we would need to use other methods of analysis. The next section will discuss other methods that can be used for creating more complex models and learning more about how much of an effect the predictors have on survival.

2.8 References

3 Cox Proportional Hazards Regression

3.1 Introduction

So far, we have only dealt with calculating survival probabilities. While the Kaplan Meier survival function is useful for predicting survival probability for simple cases, the number of possible predictors and output interpretation are limited as discussed in section 2.7. When looking to include multiple predictors in the model or to learn about the size of effect predictors have on survival, other methods need to be used. One of these methods, hazard analysis, allows for these things.

3.2 Hazard Analysis

The goal of hazard analysis is to create a hazard function for modelling time to event data, where the outcome $h(t)$ or $\lambda(t)$ is the probability of the event occurring for a subject who has lasted until time t (Clark (2003)). The hazard function is modeled using hazard ratios, which express the ratio between the hazard of an event between two groups at a time. The hazard ratio between two groups can be expressed as $v = \frac{h_1(t)}{h_2(t)}$ (Collet (2003)).

Hazard analysis can be used for more realistic scenarios that deal with additional explanatory variables for the event of interest (Clark (2003)). For example, we can use hazard analysis to calculate the chance of a person recovering from a disease based on type of treatment, while also including variables such as age, sex or history of drugs. Hazard analysis can therefore be more useful during real-world analysis than the survival functions previously discussed.

3.3 Proportional Hazards

The proportional hazards model, also known as the Cox regression model, is a semi-parametric model that can be used to predict the hazard, or risk, of the event occurring over time (Collet (2003)). One key assumption about the proportional hazard model is that the hazard ratio between two groups is constant over time. This means that the hazard ratio between two groups is the same at any time t, thus making the model proportional. In other words, the instantaneous hazard of an event occurring between two individuals of different groups will

remain constant at all times, or the effect of the predictors is the same at all times ("Cox Proportional-Hazards Model" (n.d.)).

3.4 Cox Proportional Hazards Model

The Cox proportional hazards model is expressed in terms of the hazard function on an individual i at time t. This function is expressed as $h_i(t) = vh_0(t)$, where $h_0(t)$ is the baseline hazard function and v is the hazard ratio (Collet (2003)). The baseline hazard function is the hazard function for a subject with all explanatory variables equal to zero. For the models we will discuss, the hazard ratio v is set to equal $exp(\beta)$ since the hazard ratio cannot be a negative value. The parameter *beta* is thus the log of the hazard ratio, expressed $\beta = log(v)$. Any value of β will output a positive value v. We can include many explanatory variables in the hazard function such as factors, which can take on different levels, or covariates, which can be any value on a continuous scale. With many predictors, the Cox proportional hazards model can be expressed as $h_i(t) = h_0(t)exp(\beta_1 x_{1i} + \beta_2 x_{2i} + ... + \beta_p x_{pi})$, where $h_0(t)$ is the baseline hazard function, x_1 to x_p are the values of the explanatory variables for individual i, and β_1 to β_p are the regression coefficients. The model can also be expressed as a linear model in terms of the log of the ratios between the two hazard functions, looking like $log(\frac{h_i(t)}{h_0(t)}) = \beta_1 x_{1i} + \beta_2 x_{2i} + ... + \beta_p x_{pi}$ (Collet (2003)).

3.5 Method of Maximum Likelihood

In the Cox proportional hazards model, there are two unknowns: the baseline hazard function, $h_0(t)$, and the regression coefficients, β_1 to β_p. The method of maximum likelihood is a statistical method used to estimate values for unknown parameters by fitting a model that maximizes the likelihood of outputting the data we have (Allison (1984)). In general, the maximum likelihood estimate for an unknown value, θ, is the value that maximizes the likelihood function, $L(\theta) = \prod_{i=1}^{n} f(x_i|\theta)$, where $f(x_i|\theta)$ is the probability density function, or pdf, of the data ("Maximum Likelihood Estimation" (n.d.)). The pdf and CDF, or the Cumulative Distribution Function, of the data are functions that express probability of certain values based on the distribution of the data. The baseline hazard function, $h_0(t)$, can be expressed as a function of the pdf and CDF of the data, looking like $h_0(t) = \frac{f(t)}{1-F(t)}$, where $f(t)$ is the pdf and $F(t)$ is the CDF (Allison (1984)). However, as mentioned before, a specific distribution of the data is not assumed in Hazard Analysis. This means that the pdf, CDF, and therefore the baseline hazard function, can not be calculated using the method of maximum likelihood. Instead, the method of partial likelihood can be used to estimate the regression coefficients (Collet (2003)).

3.6 Method of Partial Likelihood

The method of partial likelihood, first discovered by Cox himself, allows us to estimate the regression coefficients without knowing the baseline hazard function (Collet (2003)). As a reminder, the baseline hazard function describes the distribution of the data when no events have happened, but we only have data points at times when either an event occurs or an individual is censored. Thus, we need a way to fit a function that maximizes the likelihood of getting these data points without knowing what happens at every time. The method of partial likelihood does this by ranking the times of events in the data, and then using these ordered event times to predict the hazard ratio. It is called a partial likelihood method because it does not actually use the exact times of events but instead just their rankings (Collet (2003)). One key assumption of this method is that there are no ties in the data, or events at the same time. Additionally, it is important to note that this method assumes that the time intervals between each event are independent of the model parameters, or that the time intervals give no information about the model parameters (Waagepetersen (2022)).

3.7 Partial Likelihood Derivation

The method of partial likelihood is based on the assumption that the time intervals between events gives no information about the model parameters. Instead, the method uses the order of the events to estimate the hazard ratio. The method uses the idea of conditional probability that a certain individual has an event at time t_j given that they have survived until time t_i and that an event has occurred at time t_j.

The definition of conditional probability is given by: $P(A|B) = \frac{P(A \cap B)}{P(B)}$. This is saying that the probability of some event A given that an event B has occurred is equal to the probability that both events A and B occur divided by the probability that event B occurs. In the context of the Cox proportional hazards model, the conditional probability would look like:

$P(\text{individual with variables } x_i \text{ experiences event at time } t_j | \text{one event at time } t_j) =$

$$\frac{P(\text{individual with variables } x_i \text{ experiences event at time } t_j)}{P(\text{an event happens at time } t_j)}.$$

The probability of an event happening at t_j can be represented numerically as the sum of the probabilities of event for all of the individuals at risk at time t_j, or $\sum_{l \in R(t_j)} P(\text{individual l dies at time j})$. Now, we can replace these terms with the hazard functions, $h_i(t_j)$ and $h_l t_j)$, for the numerator and denominator, making the new probability $\frac{h_i(t_j)}{\sum_{l \in R(t_j)} h_l(t_j)}$ (Collet (2003)).

Recall that the hazard ratio, $h_i(t)$, is defined as the hazard function of individual i at time t, or $h_i(t) = h_0(t) exp(\beta_1 x_{1i} + \beta_2 x_{2i} + ... + \beta_p x_{pi})$. The baseline hazard function, $h_0(t)$, will thus

cancel out in the conditional probability equation, simplifying it to $\frac{exp(\beta' x_j)}{\sum_{l \in R(t_j)} exp(\beta' x_l)}$, where β' is the vector of regression coefficients and x_j is the vector of predictor variables for individual j (Collet (2003)).

The partial likelihood method states that the likelihood of the data is the product of the conditional probabilities of the events happening at the ordered event times, or $L(\beta) = \prod_{j=1}^{r} \frac{exp(\beta' x_j)}{\sum_{l \in R(t_j)} exp(\beta' x_l)}$, where there are a total of r ordered event times and j is the index of the ordered event times. $R(t_j)$ is the set of individuals at risk at time t_j, and l is the index of the individual within the risk set at time j. This is the likelihood of the data given the regression coefficients, β'.

The partial likelihood function can then be written as the natural logarithm of the likelihood function, or $log(L(\beta)) = log(\prod_{j=1}^{r} \frac{exp(\beta' x_j)}{\sum_{l \in R(t_j)} exp(\beta' x_l)})$. The goal of the method of partial likelihood is to find the values of β' that maximize the partial likelihood function (Collet (2003)). Once we fit this equation, we can estimate these parameters by taking the derivative with respect to β' and setting it the equation equal to zero, or $\frac{\partial l(\beta)}{\partial \beta} = 0$. This will give us the partial likelihood estimate for the regression coefficients.

3.8 Example Data

To demonstrate hazard analysis, we will need to create another simple data set. Let's recall the `surv2` data set we created in section 2.4 now with an additional variable, age, as shown in Table 3.1.

```
# Load Packages
library(tidyverse) |> suppressPackageStartupMessages()
library(knitr)
library(survival)
library(ggsurvfit)
library(survminer) |> suppressPackageStartupMessages()
library(gt)
```

```
# Recreate surv2 and add age variable
id <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
time <- c(2, 7, 9, 8, 5, 3, 4, 10, 6, 1)
status <- c(0, 1, 0, 1, 0, 1, 0, 0, 1, 1)
group <- c(1, 1, 1, 1, 1, 2, 2 ,2, 2, 2)
age <- c(40, 62, 37, 67, 44, 70, 50, 45, 61, 62)
surv3 <- data.frame(id, time, status, group, age)
```

```
surv3 %>%
  arrange(group, time) %>%
  gt(caption = "Example Data Set with Status, Time, Group, and Age") %>%
  cols_label(id = "ID", time = "Time", status = "Status",
             group = "Group", age = "Age")
```

Table 3.1: Example Data Set with Status, Time, Group, and Age

ID	Time	Status	Group	Age
1	2	0	1	40
5	5	0	1	44
2	7	1	1	62
4	8	1	1	67
3	9	0	1	37
10	1	1	2	62
6	3	1	2	70
7	4	0	2	50
9	6	1	2	61
8	10	0	2	45

Let's start by calculating a hazard function between groups 1 and 2. To do this, we will assume that the baseline hazard function is constant, or $h_0(t) = h_0$.

Recall that the Cox proportional hazard analysis equation is $h_i(t) = h_0(t)exp(\beta_1 x_{1i} + \beta_2 x_{2i} + ... + \beta_p x_{pi})$, or $h_i(t) = h_0(t)exp(\beta' x_i)$, where x_i is a binary variable representing what group individual i is in. For group 1 in our example, $h_1(t) = h_0(t)exp(\beta' x_1)$, and for group 2, $h_2(t) = h_0(t)exp(\beta' x_2)$.

We can then calculate the hazard ratio between the two groups as $v = \frac{h_1(t)}{h_2(t)} = \frac{h_0(t)exp(\beta' x_1)}{h_0(t)exp(\beta' x_2)} = \frac{exp(\beta' x_1)}{exp(\beta' x_2)}$ (Collet (2003)).

3.9 Censoring

Since censoring is still a concern for our data, an additional term needs to be added to the partial likelihood function, making it now: $L(\beta) = \prod_{i=1}^{r} [\frac{exp(\beta' x_i)}{\sum_{l \in R(t_i)} exp(\beta' x_l)}]^{\delta_i}$, where { _i} is an indicator variable that is 0 if the rth event time is right-censored and 1 otherwise. This term is added to the likelihood function to account for the fact that the event time is not observed for right-censored individuals (Collet (2003)). When taking the log, the function becomes $log(L(\beta)) = \sum_{i=1}^{r} \delta_i [\beta' x_i - log \sum_{l \in R(t_i)} exp(\beta' x_l)]$ (Collet (2003)).

3.10 Ties in the Data

One of the key assumptions we are making is that there are no ties in event or censor times. However, this is not always the case. There are additional models proposed to account for ties in the data, such as the Efron method and the Breslow method, which add weights to the likelihood function to account for ties in the data. Later, we will use the Breslow method in analysis because it is the simplest. Breslow suggested the equation: $L(B) = \prod_{i=1}^{r} \frac{exp(\beta' s_i)}{(\sum_{l \in R(t_i)} exp(\beta' x_l))^{d_i}}$, where d_i is the number of individuals at risk at time t_i (Collet (2003)). For now, we will assume there are no ties in the data and use the partial likelihood function without weights.

3.11 Fitting the Partial Log Likelihood Equation

To fit our likelihood function, we must order all of the event times. For group 1, there are only two times of event, time $t = 7$ for individual 2 and time $t = 8$ for individual 4. So, we will order the event times as $t_{(1)} < t_{(2)}$, where $t_{(1)}$ represents the event time for individual 2 and $t_{(2)}$ represents the event time for individual 4. Similarly for group 2, there are events at times 1, 3, and 6, so we will order the event times as $t_{(1)} < t_{(2)} < t_{(3)}$, representing the event times for individuals 10, 6, and 9. We will also need to calculate the risk set at each time, which is shown below in Table 3.2.

```
risk <- c(10, 9, 8, 7, 6, 5, 4, 3, 2, 1)

surv3 <- surv3 %>% arrange(time) %>%
  mutate(risk_set = risk)

surv3 %>%
  gt(caption = "Example Data Set with Status, Group, Time,
     Age, and Risk Set") %>%
  cols_label(id = "ID", group = "Group", time = "Time", status = "Status",
             age = "Age", risk_set = "Risk Set")
```

Table 3.2: Example Data Set with Status, Group, Time, Age, and Risk Set

ID	Time	Status	Group	Age	Risk Set
10	1	1	2	62	10
1	2	0	1	40	9
6	3	1	2	70	8
7	4	0	2	50	7
5	5	0	1	44	6

9	6	1	2	61	5
2	7	1	1	62	4
4	8	1	1	67	3
3	9	0	1	37	2
8	10	0	2	45	1

To estimate the parameters, we will need to use the natural log of our partial likelihood function, which we derived as: $log(L(\beta)) = \sum_{i=1}^{r} \delta_i[\beta' x_i - log \sum_{l \in R(t_i)} exp(\beta' x_l)]$.

At time $t = 1$, the risk set will include all 10 individuals because at the tie just before $t = 1$, there are no individuals who have experienced the event or been censored. At time $t = 2$, the risk set will include all individuals except for individuals with ID 10 because they experienced the event at time 1. We can continue this process for each time, and the risk set will be the same for both groups.

Now that we know the risk sets and the event times, we can calculate the partial likelihood function. Recall that the partial likelihood function is $L(\beta) = \prod_{i=1}^{n} \frac{exp(\beta x_i)}{\sum_{j \in R(t_i)} exp(\beta x_j)}$. So, for our data set, we will need to calculate the partial likelihood function for each event time. For each relevant time, we can calculate the risk score for individual i, $v(i) = B' x_i$ (Collet (2003)). This risk score represents the numerator of the partial likelihood equation. For time $t = 1$, in which individual 10 experiences the event of interest, the risk score is denoted $v(10)$. The denominator of the partial likelihood equation is the sum of the risk scores for all individuals in the risk set at time t_i, so all individuals 1 through 10. The denominator is then $v(1) + v(2) + v(3) + v(4) + v(5) + v(6) + v(7) + v(8) + v(9) + v(10)$. The partial likelihood function for time $t = 1$ is then $\frac{v(10)}{v(1)+v(2)+v(3)+v(4)+v(5)+v(6)+v(7)+v(8)+v(9)+v(10)}$.

After doing this for each event time, we get the partial likelihood equation to be:

$\frac{v(10)}{v(1)+v(2)+v(3)+v(4)+v(5)+v(6)+v(7)+v(8)+v(9)+v(10)} * \frac{v(6)}{v(2)+v(3)+v(4)+v(5)+v(6)+v(7)+v(8)+v(9)} *$

$\frac{v(9)}{v(2)+v(3)+v(4)+v(8)+v(9)} * \frac{v(2)}{v(2)+v(3)+v(4)+v(8)} * \frac{v(4)}{v(3)+v(4)+v(8)}$.

We can then take the log of the partial likelihood function to get the log partial likelihood function: $log(L(B)) = 3\beta x_2 + 2\beta x_1 - log(5exp(\beta x_1) + 5exp(\beta x_2))$ - $log(4exp(\beta x_1) + 4exp(\beta x_2)) - log(3exp(\beta x_1) + 2exp(\beta x_2)) - log(3exp(\beta x_1) + exp(\beta x_2)) - log(2exp(\beta x_1) + exp(\beta x_2))$.

3.12 Estimating B'

We can then take the derivative of the log partial likelihood function with respect to β and set it equal to 0 to find the value of β that maximizes the partial likelihood function. The derivative will be:

$\frac{\partial log(L(\beta))}{\partial \beta} = 3x_2 + 2x_1 - \frac{5x_1 exp(\beta x_1)}{5exp(\beta x_1)+5exp(\beta x_2)} - \frac{5x_2 exp(\beta x_2)}{5exp(\beta x_1)+5exp(\beta x_2)} - \frac{4x_1 exp(\beta x_1)}{4exp(\beta x_1)+4exp(\beta x_2)} - \frac{4x_2 exp(\beta x_2)}{4exp(\beta x_1)+4exp(\beta x_2)} - \frac{3x_1 exp(\beta x_1)}{3exp(\beta x_1)+2exp(\beta x_2)} - \frac{2x_2 exp(\beta x_2)}{3exp(\beta x_1)+2exp(\beta x_2)} - \frac{3x_1 exp(\beta x_1)}{3exp(\beta x_1)+exp(\beta x_2)} - \frac{x_2 exp(\beta x_2)}{3exp(\beta x_1)+exp(\beta x_2)} - \frac{2x_1 exp(\beta x_1)}{2exp(\beta x_1)+exp(\beta x_2)} - \frac{x_2 exp(\beta x_2)}{2exp(\beta x_1)+exp(\beta x_2)}$.

Setting this equation equal to zero and solving for β would theoretically give us the value of β that maximizes the partial likelihood function. However, because this equation is not solvable, we can use numerical methods to approximate the value of β. One of the most common methods is called the Newton Raphson method.

3.13 Newton Raphson Method

The Newton Raphson method is an iterative method that is used to find the root of a function, f(x). In our case, we case use this method to find the value of β that maximizes the partial likelihood function. We do this by starting with an initial estimate, x_0, for the root, and repeating a series of steps to improve this estimation. We will first chose a value for x, x_0, and then find the equation of the tangent line to the function at the point $(x_0, f(x_0))$. We can then find the x-intercept of the tangent line at x_0 by setting our function equal to 0. This x-intercept is our next estimate, x_1, for the root (Anstee (n.d.)).

Let f(x) be the function we want to find the root of and r be the root of the equation when $f(x) = 0$. Suppose $r = x_0 + h$. The value of h is the distance from the true root, r, and our initial estimate, x_0. A key assumption of this method is that x_0 is a good estimate and therefore h is close to the true value. Because h is small, we can use the linear approximation of the tangent line. To do this, we know that the slope of the tangent line is the derivative of the function at the point $(x_0, f(x_0))$. Thus we can estimate $0 = f(r) = f(x_0+h) = f(x_0)+f'(x_0)h$. Solving for h we get $h = -\frac{f(x_0)}{f'(x_0)}$. Thus, $r = x_0 + h = x_0 - \frac{f(x_0)}{f'(x_0)}$. The new value of r will be a better estimate for the root, x_1 (Anstee (n.d.)).

The equation used for this iteration looks like: $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$. We can repeat this step until the difference between $x_n + 1$ and x_n is less than a certain tolerance level, meaning that our estimate is not changing much with new iterations (Anstee (n.d.)).

3.14 Newton Raphson Example

Let's consider a simple example of a Newton Raphson iteration. Suppose we want to find the root of the function, $f(x) = x^3+3x+1$. First, we will take the derivative of the function, which will be $f'(x) = 3x^2 + 3$. We can then use the Newton Raphson method to find the root of the function. We will start with an initial estimate, $x_0 = 3$, and use the equation $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ to find our second estimate for the root. In this case, $x_{n+1} = 3 - \frac{f(3)}{f'(3)} = 3 - \frac{3^3+3*3+1}{3*3^2+3} = 3 - \frac{37}{30} = 3 - 1.2333 = 1.7667$. We can then use this value as our new estimate and repeat the

process until the difference between $x_n + 1$ and x_n is less than a certain tolerance level. Let's visualize this process in table Table 3.3.

```
# define function and its derivative
f = function(x) x^3 + 3*x + 1
fprime = function(x) 3*x^2 + 3

# Choose initial estimate to be 3
x = 3

# Use tolerance level of 0.0001
tol = 0.0001

# Initialize a vector to store the results
iterations <- c(x)

# While loop to repeat the process until the difference between
# ${x_n+1}$ and ${x_n}$ is less than the tolerance level
while (abs(f(x)) > tol){
  x = x - f(x)/fprime(x)
  x = round(x, 4)
  iterations <- c(iterations, x)  # Store the current value of x
}

# Create a data frame with results
results <- data.frame(iteration = 1:length(iterations), x = iterations)
results %>%
  gt(caption = "Newton Raphson Iteration") %>%
  cols_label(iteration = "Iteration", x = "x")
```

Table 3.3: Newton Raphson Iteration

Iteration	x
1	3.0000
2	1.7667
3	0.8111
4	0.0135
5	-0.3333
6	-0.3222

The table shows the results of the Newton Raphson method. We can see that the value of x is converging to the root of the function, which is approximately -0.32.

3.15 Newton Raphson for Parametized Functions: Gamma Distribution

The above example shows the general process for the Newton Raphson iteration, but what about when we have a function with multiple parameters? Suppose we want to find the root of the gamma distribution, $f(x) = \frac{\lambda^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\lambda x}$. First, we will need to define a likelihood function for the gamma distribution. This function takes the product of the probability density function of the gamma distribution for each observation. The likelihood function for the gamma distribution is: $L(\alpha, \lambda) = \prod_{i=1}^{n} \frac{\lambda^\alpha}{\Gamma(\alpha)} x_i^{\alpha-1} e^{-\lambda x_i}$.

We will take the natural log of the likelihood function which comes to be $ln(L(\alpha, \lambda)) = n\alpha ln(\lambda) - nln(\Gamma(\alpha)) + (\alpha - 1)\sum_{i=1}^{n} ln(x_i) - \lambda \sum_{i=1}^{n} x_i$.

We will then need the partial derivatives and the second partial derivatives of the natural log of the likelihood function with respect to α and λ.

The first partial derivative with respect to α is: $\frac{\partial ln(L(\alpha,\lambda))}{\partial \alpha} = nln(\lambda) - n\psi(\alpha) + \sum_{i=1}^{n} ln(x_i)$, where ψ is the digamma function, or the derivative of the natural log of the gamma function.

The first partial derivative with respect to λ is: $\frac{\partial ln(L(\alpha,\lambda))}{\partial \lambda} = \frac{n\alpha}{\lambda} - \sum_{i=1}^{n} x_i$.

The second partial derivative with respect to α is: $\frac{\partial^2 ln(L(\alpha,\lambda))}{\partial \alpha^2} = -n\psi'(\alpha)$, where ψ' is the trigamma function, or the second derivative of the natural log of the gamma function.

The second partial derivative with respect to λ is: $\frac{\partial^2 ln(L(\alpha,\lambda))}{\partial \lambda^2} = -\frac{n\alpha}{\lambda^2}$.

Now, we will use the Newton Raphson method to find the values of α and λ that maximize the function. Recall that the Newton Raphson method used the equation $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$. In this case, we will have two equations, one for α and one for λ. We will first need to choose initial estimates for α and λ. These initial estimates can be chosen using other methods such as Method of Moments Estimation. The Method of Moments Estimator of the parameters of the Gamma Distribution are: $\hat{\alpha} = \frac{\bar{x}^2}{s^2}$ and $\hat{\lambda} = \frac{\bar{x}}{s^2}$, where $\bar{x}$ is the sample mean and s is the sample standard deviation. We can then use these estimates as our initial estimates for the Newton Raphson method using the equation $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ to find our second estimate for the root and then repeating the process until our tolerance requirement is met. Let's use R to generate a random sample from a gamma distribution and find the sample mean and sample standard deviation. These statistics are in Table 3.4.

```
# Generate random gamma distribution and summary statistics
set.seed(1)
gamma <- round(rgamma(10, shape = 3, rate = .5), 1)

# Put these values into a data frame
gamma_data <- data.frame("n" = 10,
```

```
                          "sum_x" = sum(gamma),
                          "sum_ln_x" = sum(log(gamma)),
                          "mean" = mean(gamma),
                          "variance" = var(gamma))
```

```
# View data in a table
gamma_data %>%
  gt(caption = "Summary Statistics for Gamma Distribution") %>%
  cols_label(n = "Number of Observations",
             sum_x = "Sum of x",
             sum_ln_x = "Sum of ln(x)",
             mean = "Mean",
             variance = "Variance")
```

Table 3.4: Summary Statistics for Gamma Distribution

Number of Observations	Sum of x	Sum of ln(x)	Mean	Variance
10	67.7	17.79487	6.77	12.75789

We can now determine our initial estimates for α and λ using the Method of Moments Estimation. The sample mean is 2.3 and the sample standard deviation is 1.1. We can use these values to find our initial estimates for α and λ.

```
# Method of Moments Estimation
mean = 6.77
variance = 12.76

# Initial estimates for alpha and lambda
a = mean^2 / variance
L = mean / variance

# Display initial estimates
alpha_lambda <- data.frame("alpha" = a, "lambda" = L)
```

```
# View data in a table
alpha_lambda %>%
  gt(caption = "Initial Estimates for Gamma Distribution") %>%
  cols_label(alpha = "Alpha",
             lambda = "Lambda")
```

Table 3.5: Initial Estimates for Gamma Distribution

Alpha	Lambda
3.59192	0.5305643

Now that we have all the information needed, we can start the Newton Raphson method. We will walk through the first iteration and then use R to do the rest.

To find the first updated estimates for α and λ, we will use the equations $\alpha_{n+1} = \alpha_n - \frac{f(\alpha_n)}{f'(\alpha_n)}$ and $\lambda_{n+1} = \lambda_n - \frac{f(\lambda_n)}{f'(\lambda_n)}$.

We will start with our Method of Moments estimates of $\hat{\alpha} = 3.59192$ and $\hat{\lambda} = 0.5305643$. For alpha, the first iteration would look like: $\alpha_1 = 3.59192 - \frac{nln(\lambda) - n\psi(\alpha) + \sum_{i=1}^{n} ln(x_i)}{-n\psi'(\alpha)}$. Plugging in the values, we get $\alpha_1 = 3.59192 - \frac{10*ln(0.5305643) - 10*digamma(3.59192) + 17.79487}{-10*trigamma(3.59192)} = 3.631203$.

For lambda, the first iteration would look like: $\lambda_1 = 0.5305643 - \frac{\frac{n\alpha}{\lambda} - \sum_{i=1}^{n} x_i}{-\frac{n\alpha}{\lambda^2}}$. Plugging in the values, we get $\lambda_1 = 0.5305643 - \frac{\frac{10*3.59192}{0.5305643} - 67.7}{-\frac{10*3.59192}{0.5305643^2}} = 0.5305643$.

```
# Calculate the first iteration for alpha
3.59192 - ((10 * log(0.5305643) - 10 * digamma(3.59192) + 17.79487) /
             (-10 * trigamma(3.59192)))
```

```
[1] 3.631203
```

```
# Calculate the first iteration for lambda
0.5305643 - (((10 * 3.59192 / 0.5305643) - 67.7) /
               (-10 * 3.59192 / 0.5305643^2))
```

```
[1] 0.5305643
```

Now, we can continue this process using the new estimates of α and λ until the difference between the current and previous estimates is less than a certain tolerance level. We will use a tolerance level of 0.005. Let's visualize this process in table Table 3.6.

```
# Define functions using a for alpha and L for lambda
afunc = function(a, L, n, sum_ln_x) a -
  ((n*log(L) - n * digamma(a) + sum_ln_x) / (-n * trigamma(a)))

Lfunc = function(a, L, n, sum_x) L - (((n*a/L) - sum_x) / (-n * a / L^2))
```

```r
# Choose initial estimates
a = 3.59192
L = 0.5305643
n = 10
sum_ln_x = sum(log(gamma))
sum_x = sum(gamma)

# Use tolerance level of 0.005
tol = 0.005

# Initialize a vector to store the results
iterations <- c(a, L)

# While loop to repeat the process until the difference between a and a1 and
# L and L1 is less than the tolerance level
while (TRUE) {
  a1 <- afunc(a, L, n, sum_ln_x)
  L1 <- Lfunc(a, L, n, sum_x)
  if (abs(a - a1) < tol & abs(L - L1) < tol) {
    break
  }
  a <- a1
  L <- L1
  iterations <- c(iterations, a, L)  # Store the current values of a and L
}

# put the results into a table with every 10th value
results <- data.frame(iteration = 1:length(iterations),
                      a = iterations[seq(1, length(iterations), 2)],
                      L = iterations[seq(2, length(iterations), 2)])

# print the table
results %>%
  gt(caption = "Newton Raphson for Gamma Distribution") %>%
  cols_label(iteration = "Iteration", a = "Alpha", L = "Lambda")
```

Table 3.6: Newton Raphson for Gamma Distribution

Iteration	Alpha	Lambda
1	3.591920	0.5305643
2	3.631203	0.5305643

3	3.631450	0.5363040
4	3.665424	0.5364032
5	3.591920	0.5305643
6	3.631203	0.5305643
7	3.631450	0.5363040
8	3.665424	0.5364032

As we can see, we didn't quite reach the original values of $a = 3$ and $L = 0.5$. This is because we started with a small sample. Let's use a larger sample and a lower tolerance level to see if we can get closer.

```
# Generate random gamma distribution with sample of 10000
set.seed(1)
gamma2 <- rgamma(10000, shape = 3, rate = .5)
```

```
# Define functions using a for alpha and L for lambda
afunc = function(a, L, n, sum_ln_x) a -
  ((n*log(L) - n * digamma(a) + sum_ln_x) / (-n * trigamma(a)))

Lfunc = function(a, L, n, sum_x) L - (((n*a/L) - sum_x) / (-n * a / L^2))

# Choose initial estimates
a = 3.7
L = 0.7
n = 10000
sum_ln_x = sum(log(gamma2))
sum_x = sum(gamma2)

# Use tolerance level of 0.0001
tol = 0.0001

# Initialize a vector to store the results
iterations <- c(a, L)

# While loop to repeat the process until the difference between a and a1 and
# L and L1 is less than the tolerance level
while (TRUE) {
  a1 <- afunc(a, L, n, sum_ln_x)
  L1 <- Lfunc(a, L, n, sum_x)
  if (abs(a - a1) < tol & abs(L - L1) < tol) {
    break
```

```
  }
  a <- a1
  L <- L1
  iterations <- c(iterations, a, L)  # Store the current values of a and L
}

# Find out how many iterations were needed
iterations <- length(iterations)

# put the last 10 results into a table
results2 <- data.frame(iterations, a, L)

# print the table
results2 %>%
  gt(caption = "Alpha and Lambda Estimates with Larger Sample") %>%
  cols_label(iterations = "Numer of Iterations", a = "Alpha", L = "Lambda")
```

Table 3.7: Newton Raphson for Gamma Distribution with Larger Sample

Numer of Iterations	Alpha	Lambda
212	3.010533	0.4976109

As we can see, a larger sample size and lower tolerance level allowed us to get extremely close to the true values of $a = 3$ and $L = 0.5$, although it did take many more iterations. This demonstrates the importance of sample size and tolerance level in the Newton-Raphson method. Now that we understand the Newton-Raphson method for estimating the model coefficients, let's use R to model the hazard function.

3.16 Hazard Analysis in R

Similarly to before with the Kaplan Meier curve, we can use R to model the hazard function. The `coxph()` function in the `survival` package returns the coefficients of the cox proportional hazards model as well as the p value for the coefficients, allowing us to determine whether each coefficient is significant ("Cox Proportional-Hazards Model" (n.d.)).

```
coxph(Surv(time, status) ~ group, data = surv3, ties = "breslow")
```

```
Call:
coxph(formula = Surv(time, status) ~ group, data = surv3, ties = "breslow")
```

```
        coef exp(coef) se(coef)     z     p
group 0.8606    2.3646   0.9325 0.923 0.356

Likelihood ratio test=0.87  on 1 df, p=0.3496
n= 10, number of events= 5
```

As we can see, the coefficient for group is not significant, meaning that the hazard of the event occurring is not significantly different between the two groups. Let's add another predictor and see what happens.

```
coxph(Surv(time, status) ~ group + age, data = surv3, ties = "breslow")
```

```
Call:
coxph(formula = Surv(time, status) ~ group + age, data = surv3,
    ties = "breslow")

          coef exp(coef) se(coef)     z      p
group  2.60097  13.47683  1.79663 1.448 0.1477
age    0.18655   1.20508  0.09491 1.965 0.0494

Likelihood ratio test=8.79  on 2 df, p=0.01232
n= 10, number of events= 5
```

After adding age as a predictor, we see that our model is significant, meaning that the hazard of the event occurring is significantly different for individuals of different groups and ages. We can further interpret the results by looking at the hazard ratios for each predictor. The 'coef' column tells us the natural log of the haard ratio on the predictor, so we are interested in the 'exp(coef)' column, which exponentiates the log hazard ratio to give us the hazard ratio. For group, the hazard ratio is 13.48, meaning that the hazard of the event occurring is 13.48 times higher for the second group compared to the first group. For age, the hazard ratio is 1.21, meaning that the hazard of the event occurring is 1.21 times higher for each additional year of age. The coefficient on the age predictor is significant, but the coefficient for group is not. This means that, when controlling for the other variables, the hazard of the event occurring is not significantly different between the two groups, but the hazard of the event occurring is significantly different for individuals of different ages.

3.17 Case Study: Heart Failure Data

Now that we can interpret the results, let's take a look at some real data. Recall our heart failure data from before in which we found a significant difference in survival curves between

patients with normal and high blood pressure. Although the cox proportional hazards model tests the difference in hazard of death rather than survival curves, we would expect to again get a significant difference between the two groups.

```
# Reload the data
heart <- read_csv("data/S1Data.csv")
```

```
Rows: 299 Columns: 13
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (13): TIME, Event, Gender, Smoking, Diabetes, BP, Anaemia, Age, Ejection...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Create Survival Object
times_heart <- Surv(heart$TIME, heart$Event)
```

Let's first test our theory by fitting a cox proportional hazards model to the data. This will tell us if blood pressure is a significant predictor of the hazard of death for heart failure patients.

```
# Cox Proportional Hazards Model
h1 <- coxph(times_heart ~ BP, data = heart, ties = "breslow")
h1
```

```
Call:
coxph(formula = times_heart ~ BP, data = heart, ties = "breslow")

     coef exp(coef) se(coef)     z      p
BP 0.4354    1.5456   0.2094 2.079 0.0376

Likelihood ratio test=4.18  on 1 df, p=0.04079
n= 299, number of events= 96
```

As we can see, the p-value for this model is significant. Thus, we can conclude that high blood pressure is significantly associated with a higher hazard of death for heart failure patients. The coefficient of interest, exp(coef), on blood pressure in the model is 1.55, which means that the hazard of death for heart failure patients with high blood pressure is estimated to be 1.55 times higher than the hazard of death for heart failure patients with normal blood pressure. In other words, at any given time, an individual with high blood pressure is 55% more likely

to die than an individual with normal blood pressure, given they are a heart failure patient over 40 and all other variables are fixed.

We can also get a better understand of the impact of high blood pressure on the hazard of death for heart failure patients by looking at a confidence interval for the exponentiated coefficient on blood pressure. We can use the confint() function to do this.

```
conf_95 <- exp(confint(h1))
conf_95
```

```
      2.5 %   97.5 %
BP 1.025291 2.329885
```

These values tell us that, with 95% confidence, the true value of the coefficient is between 1.03 and 2.33. That is, that the hazard of death for heart failure patients is between 1.03 and 2.33 times higher for patients with high blood pressure compared to patients with normal blood pressure.

Although we found that the hazard of death for heart failure patients with high blood pressure is significantly higher than the hazard of death for heart failure patients with normal blood pressure, we should check the proportional hazards assumption to ensure that our model is valid. From the Kaplan Meier curve, we saw that the gap between the two survival probability curves seemed to be consistent over time, which indicates that the assumption is likely met. We can use the `cox.zph()` function to check it out. This function tests the null hypothesis that the coefficients are constant over time, which is the assumption of proportional hazards. If the p-value is less than 0.05, then the assumption is violated.

```
# Check Proportional Hazards Assumption
cox.zph(h1)
```

```
       chisq df    p
BP     0.473  1 0.49
GLOBAL 0.473  1 0.49
```

As we can see from the output, the p-value for the test of the proportional hazards assumption is not less than 0.05. This means that we do not have enough evidence to reject the null hypothesis, so our Cox Proportional Hazards model is valid. We can also look at a plot of the cumulative hazard over time for each of the groups for another visual representation of this. The cumulative hazard plot is found by integrating The cumulative hazard plots are shown in Figure 3.1 below.

```r
# Fit the Model
heart_fit = survfit(times_heart ~ BP, data = heart)

# Plot Cumulative Hazard Function
ggsurvplot(heart_fit, data = heart,
           fun = "cumhaz", palette = c("darkgreen", "maroon"),
    legend.labs = c("Normal BP", "High BP"),
    xlab = "Time (Days)",
    ylab = "Cummulative Hazard",
    title = "Cummulative Hazard of Death by Blood Pressure")
```

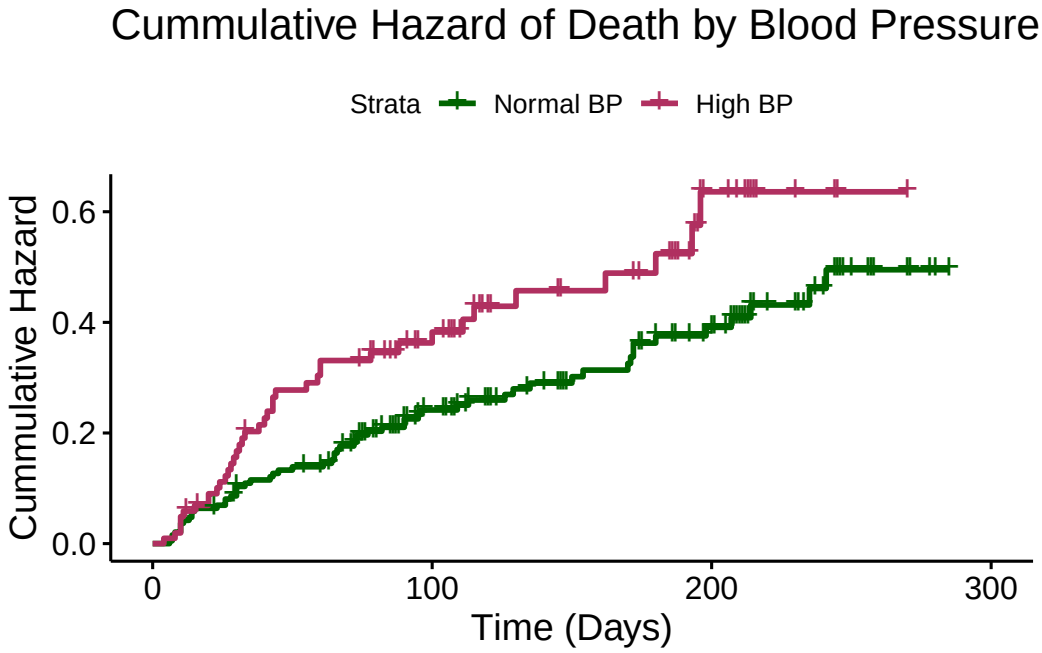

Figure 3.1: Cummulative Hazard of Death by Blood Pressure

As we can see from the cumulative hazard plot, the hazard ratio does seem to be consistent over time. In other words, the gap between the two curves stays fairly consistent. This is consistent with the proportional hazards assumption, which states that the hazard ratio between the two groups is constant over time. Since the assumptions of the Cox Model are met, we can be confident in our results that Blood Pressure is a significant predictor of the hazard of death for heart failure patients.

3.18 References

4 Discussion

4.1 Conclusion

Survival analysis is a great tool for medical research, as it can show insights into which treatments are most effective or how long patients are likely to live. Survival analysis works by calculating survival probability for an individual at each time in the study. The Kaplan Meier survival curve is a great way to model survival probabilities while factoring in censored individuals who were lost-to-follow-up. The `survfit()` and `ggsurvfit()` functions in the `survival` package in R are tools we can use to create Kaplan Meier curves quickly given data.

The Log-Rank test is a statistical test we used to test whether there was a difference between two survival curves. The Log-Rank test is one of the most common ways to test for statistical significance between groups when analyzing medical data. It works by testing the null hypothesis that there is no difference between survival estimate between two groups at any given time. It compares expected to observed time estimates for each group at each time and then calculates a Chi-Square test statistic. These steps are very tedious, so the `survdiff()` R function can be used to make the process much quicker. This process was demonstrated using data from the cirrhosis data set, which predicted survival times patients with liver disease who were either taking a drug or a placebo. We found that the difference between the survival times in the two groups was not statistically significant, thus concluding that the drug was not effective at lengthening a patient with liver disease's life.

Hazard Analysis was the next type of modeling survival data that we looked at. Hazard Analysis is even more useful for analysis of medical data because it allows for the addition of more predictors other than a grouping variable. The Cox Proportional Hazards model uses hazard ratios, or ratio between the hazard of an event between two groups at a time. It calculates the hazard ratio for each individual at each time and uses the log of that ratio to model the hazard function. To use the Cox Proportional Hazards model, we first had to make adjustments to the hazard function to factor censored data and ties in the data. We then had to use the method of partial likelihood and the Newton Rapshon method to find the β coefficients. This step was so complicated that it was computationally infeasible to demonstrate. Luckily, these complications can be managed by using the R funtion `coxph()` in the `survival` package. It fits a Cox proportional Hazard Analysis model in one simple step. We demonstrated this using data on patients who experienced heart failure.

4.2 Additional Methods

While the Kaplan Meier Curve and Cox Proportional Hazard Analysis are both useful for medical research, there are even further methods within Survival Analysis that can allow for different types of analysis. For example, neither of these methods assumed a distribution of the data, but there are other methods called Parametric models that can be used when there are preconceptions about the survival probabilities. This type of analysis assumes a specific distribution and therefore allows interpretation of the analysis to be more precise (Collet (2003)). One example of a Parametric model is the Accelerated Failure Time model, which assumes that the probability of the event occurring gets increasingly large as time passes (Fedesoriano (2021)). This can be particularly useful in cases where the assumption of consistent hazards between groups is violated.

Other pieces of the survival analysis puzzle to take into account include: assessing fit of the model, modelling left-censored and interval-censored survival data sample size requirements for a survival study, and more (Collet (2003)). There are so many aspects to modelling survival data that need to be taken into account when building a model. This is why it is so helpful to have software like R that can do the analysis piece for us.

References

Ahmad, Munir, T. 2017. "Survival Analysis of Heart Failure Patients: A Case Study." *PLOS ONE* 12 (7). https://doi.org/10.1371/journal.pone.0181001.

Allison, P. 1984. *Event History Analysis*. SAGE Publications, Inc. https://methods.sagepub.com/book/event-history-analysis/.

Anstee, R. n.d. *The Newton-Raphson Method*. https://personal.math.ubc.ca/~anstee/math104/newtonmethod.pdf.

Clark, Bradburn, T. G. 2003. "Survival Analysis Part i: Basic Concepts and First Analyses." *British Journal of Cancer*, May. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2394262/.

Collet, David. 2003. *Modelling Survival Data in Medical Research*. Chapman & Hall/CRC.

"Cox Proportional-Hazards Model." n.d. www.sthda.com/english/wiki/cox-proportional-hazards-model.

Fedesoriano. 2021. "Cirrhosis Prediction Dataset." www.kaggle.com/datasets/fedesoriano/cirrhosis-prediction-dataset/data.

Goel, Khanna, M. K. 2010. "Understanding Survival Analysis: Kaplan-Meier Estimate." *International Journal of Ayurveda Research*. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3059453.

"Maximum Likelihood Estimation." n.d. https://online.stat.psu.edu/stat415/lesson/1/1.2.

Rao, & Schoenfeld, S. R. 2023. "Statistical Primer for Cardiovascular Research." *AHA Journals*. https://www.ahajournals.org/doi/pdf/10.1161/circulationaha.106.614859.

Rich, Neely, J. T. 2010. "A Practical Guide to Understanding Kaplan-Meier Curves." *Otolaryngology–Head and Neck Surgery: Official Journal of American Academy of Otolaryngology-Head and Neck Surgery.* https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3932959/.

Sullivan, LaMorte, L. 2016a. "Comparing Survival Curves." sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_survival/BS704_Survival5.html.

———. 2016b. "Estimating the Survival Function." https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_survival/bs704_survival4.html.

Waagepetersen, Rasmus. 2022. "Cox's Proportional Hazards Model and Cox's Partial Likelihood." https://people.math.aau.dk/~rw/Undervisning/DurationAnalysis/Slides/lektion3.pdf.

Zabor, E. C. 2023. "Survival Analysis in r." https://www.emilyzabor.com/tutorials/survival_analysis_in_r_tutorial.html.