

Lab 21 - Survival and Bootstrapping

Nick Sumpter (Edited by Eddie-Williams Owiredo & Guy Twa)

2023-09-15

- Today's Lab
- Loading Packages and Data
 - Getting to Know Your Data
- Kaplan-Meier Analysis
 - Assumptions
 - Running the Analysis
 - Reporting the Results for a Kaplan-Meier Analysis
- Cox Regression Analysis
 - Assumptions
 - Kaplan-Meier Curves
 - Statistical Test of Proportional Hazards
 - Running the Test
 - Reporting the Results
 - Bootstrapping the Hazard Ratio
- Independent Practice

Today's Lab

This lab covers two basic survival analyses: the Kaplan-Meier survival analysis, and the Cox Proportional Hazards survival analysis. Additionally, we will demonstrate how to perform bootstrapping on a statistic.

Loading Packages and Data

We will be using the `survival`, `survminer`, and `ggfortify` packages for our analyses today. As `survminer` and `ggfortify` are not installed by default, we will need to install these first. `survival` is the base R package that does the heavy lifting here while the other two provide useful support functions for plotting survival curves, calculating medians, etc.

```
install.packages("survminer")
install.packages("ggfortify")
```

```
library(survival)
library(survminer)
library(ggfortify)
library(tidyverse)

setwd("/Users/eddie-williamsowiredo/Desktop/grd770_23/Lab21")

load("Ex_Survival.RData")
```

Getting to Know Your Data

As you will see, there are two datasets loaded in from the `Ex_Survival.RData` file called `surv1` and `surv2`. We will be using these for our Kaplan-Meier analysis and Cox Proportional Hazards analysis respectively. The `surv1` data has 40 observations of 6 variables, let's run the `summary` function to learn more about them:


```
summary(surv1)
```

```
##           id           status    starttime seizuretime      drug
## Min.      : 1.000  No Seizure:23   Min.      :0      Min.      : 0.201   No :20
## 1st Qu.: 1.000  Seizure   :17   1st Qu.:0      1st Qu.: 6.300   Yes:20
## Median :20.50                      Median :0      Median :18.742
## Mean    :20.50                      Mean    :0      Mean    :18.446
## 3rd Qu.:30.25                      3rd Qu.:0      3rd Qu.:33.589
## Max.     :40.00                      Max.     :0      Max.     :35.000
##      status.num
## Min.      :1.000
## 1st Qu.:1.000
## Median :1.000
## Mean     :1.425
## 3rd Qu.:2.000
## Max.     :2.000
```

1. `id` = numeric identifier column
2. `status` = factor with 2 levels saying whether a seizure occurred in that individual
3. `starttime` = numeric column telling us when the time point that measurement began for that individual (all 0 in this dataset)
4. `seizuretime` = time at which the event (seizure) occurred (or time at which sample was censored from the study)
5. `drug` = whether the individual was on the drug or not
6. `status.num` = numeric version of `status` where 1 = No Seizure and 2 = Seizure

Now let's do the same for `surv2`:

```
summary(surv2)
```

```
##           id           status    starttime    stoptime        ab40        tau
## Min.      : 1.00    Alive:11    Min.      :0    Min.      : 0.3831    No :20    No :20
## 1st Qu.:10.75    Dead :29    1st Qu.:0    1st Qu.: 5.4463    Yes:20    Yes:20
## Median :20.50                Median :0    Median :13.5747
## Mean    :20.50                Mean    :0    Mean    :15.3746
## 3rd Qu.:30.25                3rd Qu.:0    3rd Qu.:30.0000
## Max.     :40.00                Max.     :0    Max.     :30.0000
## ab40.tau.intxn    status.num
## neither:10        Min.      :1.000
## ab           :10    1st Qu.:1.000
## tau          :10    Median   :2.000
## both         :10    Mean     :1.725
##               3rd Qu.:2.000
##              Max.      :2.000
```

This dataset has the following columns with 40 rows each:

1. `id` = same as for `surv1` dataset
2. `status` = factor indicating whether the individual was alive or dead at the final time point
3. `starttime` = same as for `surv1`
4. `stoptime` = same as `seizuretime` in `surv1`
5. `ab40` = does the mouse express the gene for creating the a-beta 40 protein
6. `tau` = does the mouse express the gene for creating the tau protein
7. `ab40.tau.intxn` = describes if the mouse expresses both genes, just ab40, just tau, or neither
8. `status.num` = numeric version of status (1 = Alive , 2 = Dead)

Kaplan-Meier Analysis

Assumptions

For the Kaplan-Meier analysis, we have 4 specific assumptions to look at:

1. Any cases that are censored should have the same survival prospects as those that continue to the end of the experiment
2. Survival probabilities are the same for subjects that started early and late in the study
3. The event happens at a very specific time rather than at some point during a range of times
4. Independence - all samples should be independent of each other

For the first assumption, there just needs to be a similar amount of censorship per group. For this, we can view the number of censored points graphically (during our analysis) and compare. Our design passes both other assumptions as well. All of our mice were administered the drug/placebo at the same time which passes assumption 2. We have very specific measures of the time of first seizure which passes assumption 3. And of course, we assume independence as each row represents a unique sample.

Running the Analysis

To run the analysis, we are going to use the `Surv` function from the `survival` package, and the `surv_fit` function from the `survminer` package. For the `Surv` function, we just need to pass in two variables: `time` = the amount of time that each individual was observed, and `event` = whether the mouse had a seizure or not (as a numeric variable). Even though we have a factor saying whether mouse had a seizure or not, the function only works correctly if that is input as a numeric variable with the reference level (No Seizure in this case) as the lowest number. This will create a series of numbers which represent the time to event data for each individual, along with a + for those samples that are still event free at that time point (i.e. censored individuals or those that made it to the end of the study).

We are then going to fit survival curves to this data as a function of whether they took the drug or the placebo. This will be done with a formula in the `surv_fit` function. The `surv_table` variable is our dependent variable and the drug treatment is our independent variable. For this case, the drug treatment should be a factor as opposed to a numeric variable.

```
surv_table <- Surv(time = surv1$seizuretime, event = surv1$status.num)

km.fit <- surv_fit(formula = surv_table ~ drug, data = surv1)
```

Now that we have run our model, we will summarize it using the `summary` function. By default it will automatically set a series of time points for determining percent survival, though we can provide this manually using the `seq` function in the `times` argument. Basically, this just creates a series of numbers between one value (`from`) and another value (`to`), with the interval between these variables set as the `by` argument. In our case we want the number 0, 5, 10, 15 ... 30, 35.

```
summary(km.fit, times = seq(from = 0, to = 35, by = 5))
```

```
## Call: survfit(formula = surv_table ~ drug, data = structure(list(id = c(1,
## 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
## 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
## 36, 37, 38, 39, 40), status = structure(c(2L, 2L, 2L, 2L, 2L,
## 1L, 2L, 2L, 2L, 1L, 1L, 2L, 2L, 1L, 1L, 2L, 1L, 1L, 2L, 2L, 1L,
## 2L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 2L, 2L, 1L, 1L, 1L, 1L, 1L, 1L,
## 2L, 1L, 1L), levels = c("No Seizure", "Seizure"), class = "factor"),
##   starttime = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
##   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
##   0, 0, 0, 0, 0, 0), seizuretime = c(4.68484938519829, 8.00199755196658,
##   0.451036287371672, 0.991761039721032, 26.9492107237754, 35,
##   29.9630503759079, 1.70374770432105, 22.8842266097321, 7.82711922154954,
##   35, 2.994752173, 9.58873433149431, 3.2300856650395, 7.92064965889274,
##   5.16635282678698, 15.4646594621864, 22.3449513033702, 22.4927557261188,
##   6.4190917609278, 35, 15.7506655849697, 35, 9.6274366330952,
##   28.3450122408293, 30.5148402258189, 5.94123283563396, 35,
##   12.6082216962699, 0.20098767061449, 21.8540873045226, 21.7326012145873,
##   35, 33.1190822471468, 35, 1.34050470991417, 35, 8.5845784013265,
##   35, 35), drug = structure(c(1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L,
##   1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 2L, 2L, 2L,
##   2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L,
##   2L, 2L), levels = c("No", "Yes"), class = "factor"), status.num = c(2,
##   2, 2, 2, 1, 2, 2, 2, 1, 1, 2, 2, 1, 1, 2, 1, 1, 2, 2,
##   1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 2, 1,
##   1)), class = "data.frame", row.names = c(NA, -40L)))
```

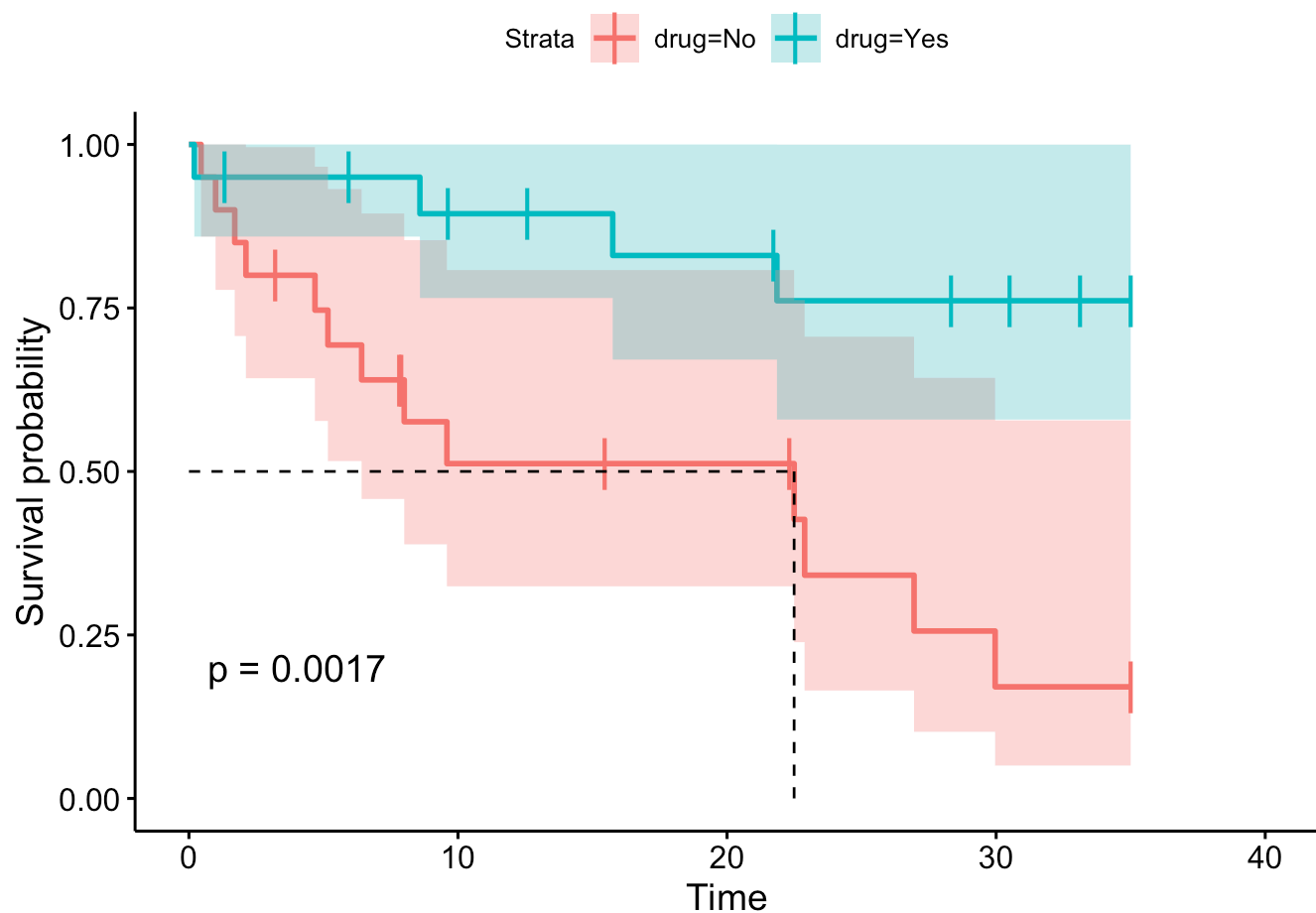
```
##
##               drug=No
## time n.risk n.event survival std.err lower 95% CI upper 95% CI
##    0      20       0   1.000  0.0000      1.0000      1.000
##    5      14       5   0.747  0.0981      0.5772      0.966
##   10       8       4   0.512  0.1192      0.3245      0.808
##   15       8       0   0.512  0.1192      0.3245      0.808
##   20       7       0   0.512  0.1192      0.3245      0.808
##   25       4       2   0.341  0.1266      0.1650      0.706
##   30       2       2   0.171  0.1062      0.0504      0.578
##   35       2       0   0.171  0.1062      0.0504      0.578
##
##               drug=Yes
## time n.risk n.event survival std.err lower 95% CI upper 95% CI
##    0      20       0   1.000  0.0000      1.000      1
##    5      18       1   0.950  0.0487      0.859      1
##   10      15       1   0.894  0.0710      0.765      1
##   15      14       0   0.894  0.0710      0.765      1
##   20      13       1   0.830  0.0902      0.671      1
##   25      11       1   0.761  0.1059      0.579      1
##   30      10       0   0.761  0.1059      0.579      1
##   35       8       0   0.761  0.1059      0.579      1
```

The summary output first gives a bunch of numbers detailing the model which you can just ignore. The things we care about are the two tables at the bottom. For this example, the top table shows the results of the survival analysis in those that were not given the drug treatment, and the bottom table shows these results for those that

were treated. You will see that the `time` column is as we specified with the `times` argument in the `summary` function. The `n.risk` column gives the number of at risk samples (i.e. those who have not had a seizure and have not been censored) at each of the time points. `n.event` gives the number of events between the previous and current timepoint. Then the other columns show the calculated survival percentages and their confidence limits.

We can plot the Kaplan-Meier curves for both drug levels using the `ggsurvplot` function.

```
ggsurvplot(fit = km.fit, data = surv1, # provide the model and the data
            conf.int = TRUE, # plot shaded confidence intervals
            surv.median.line = 'hv', # plot horizontal and vertical lines at the median
            for each curve
            pval = TRUE, # plot the overall p-value
            censor.shape = 124, # change the shape for the censor mark
            censor.size = 7) # change the size for the censor mark
```




You see two separate survival curves, one for each drug level. The shaded region is the confidence interval at that point, and the horizontal marks indicate a censored data point along each of the curves. Just from looking at it, you can see there is some difference in survival between the drug level groups, and the displayed p-value confirms this. We can also confirm that roughly the same number of individuals were censored in the two treatment groups, and thus this assumption was met.

We will perform the mathematical test for whether there is a difference in survival using the `survdif` function. It is going to have the exact same form as the `surv_fit` function from above.

```
survdifff(formula = surv_table ~ drug, data = surv1)
```

```
## Call:
## survdifff(formula = surv_table ~ drug, data = surv1)
##
##           N Observed Expected (O-E)^2/E (O-E)^2/V
## drug=No  20         13      6.77      5.75      9.84
## drug=Yes 20          4     10.23      3.80      9.84
##
##  Chisq= 9.8  on 1 degrees of freedom, p= 0.002
```

The output of this  is a Chi-square test for a difference in observed and expected values for the two drug groups. As you can see, the observed number of events for the two groups are significantly different from their expected values, with a p-value of 0.002 (the same as that displayed on the plot).

Another important measure is the time at which 50% of the group remains unaffected for both drug treatment levels. We can get this using the `surv_median` function.

```
surv_median(fit = km.fit)
```

```
##      strata  median    lower upper
## 1 drug=No 22.49276 6.419092    NA
## 2 drug=Yes      NA      NA     NA
```

According to this, the time point at which 50% of the sample was left was 22.49 minutes for the group who took the placebo, whereas that point does not exist for the drug group since the survival curve never dropped below 50% for them.

Reporting the Results for a Kaplan-Meier Analysis

“A Kaplan-Meier curve was used to compare differences in time until a seizure occurred for two groups of mice: those treated with a drug and vehicle controls. The results indicated that 50% of vehicle-treated mice had a seizure by 22.49 minutes after injection, while 76.1% of drug-treated mice had still not exhibited a seizure by the end of the 35 minute trial. Survival times of the two groups were significantly different (Chi-square(1) = 9.8, p = 0.002”).

Cox Regression Analysis

Now that we have performed the more simple Kaplan-Meier analysis, it is time to move to the more complex Cox Proportional Hazards model. For this analysis, we are going to be looking at the effect of expression of two genes (a-beta and tau) on time-to-death in mice using the `surv2` dataset.

Assumptions

1. Independence - as for the Kaplan-Meier analysis, we need to make sure each sample is independent from each other sample. In this case, each mouse was only used once, so the independence assumption is met.
2. Proportional hazards - we can do this with a statistical test and by inspecting curves graphically.

For graphical tests of proportional hazards, we will want to plot the Kaplan-Meier curves for each unique group in two ways: both normally (survival rate vs time) and on a log minus log plot ($\log(-\log(\text{survival rate}))$ vs $\log(\text{time})$). We are looking to make sure the different curves are similarly shaped, or basically that they don't substantially cross over.

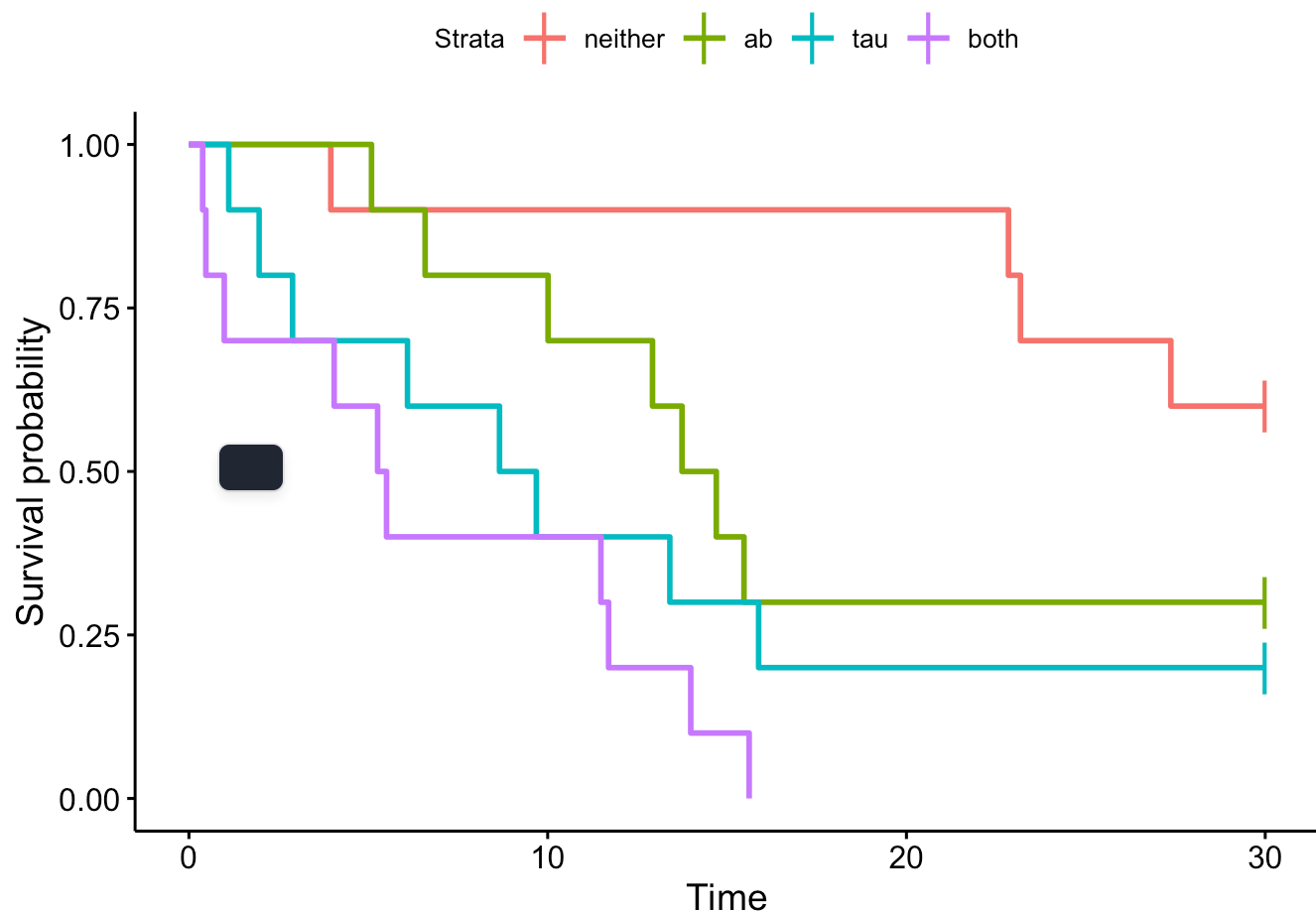
Kaplan-Meier Curves

We will first make our normal curves in the same method as before, using the `surv_fit` and `ggsurvplot` functions. The only difference is that we are going to pass in the `ab40.tau.intxn` variable that describes the interaction of tau and ab40. This has 4 groups that describe mice that express ab40 only, tau only, neither, or both in a single column. `surv_fit` does not like having interaction terms in the formula, so the interaction was created as a separate variable.

```
surv_table <- Surv(time = surv2$stoptime, event = surv2$status.num)

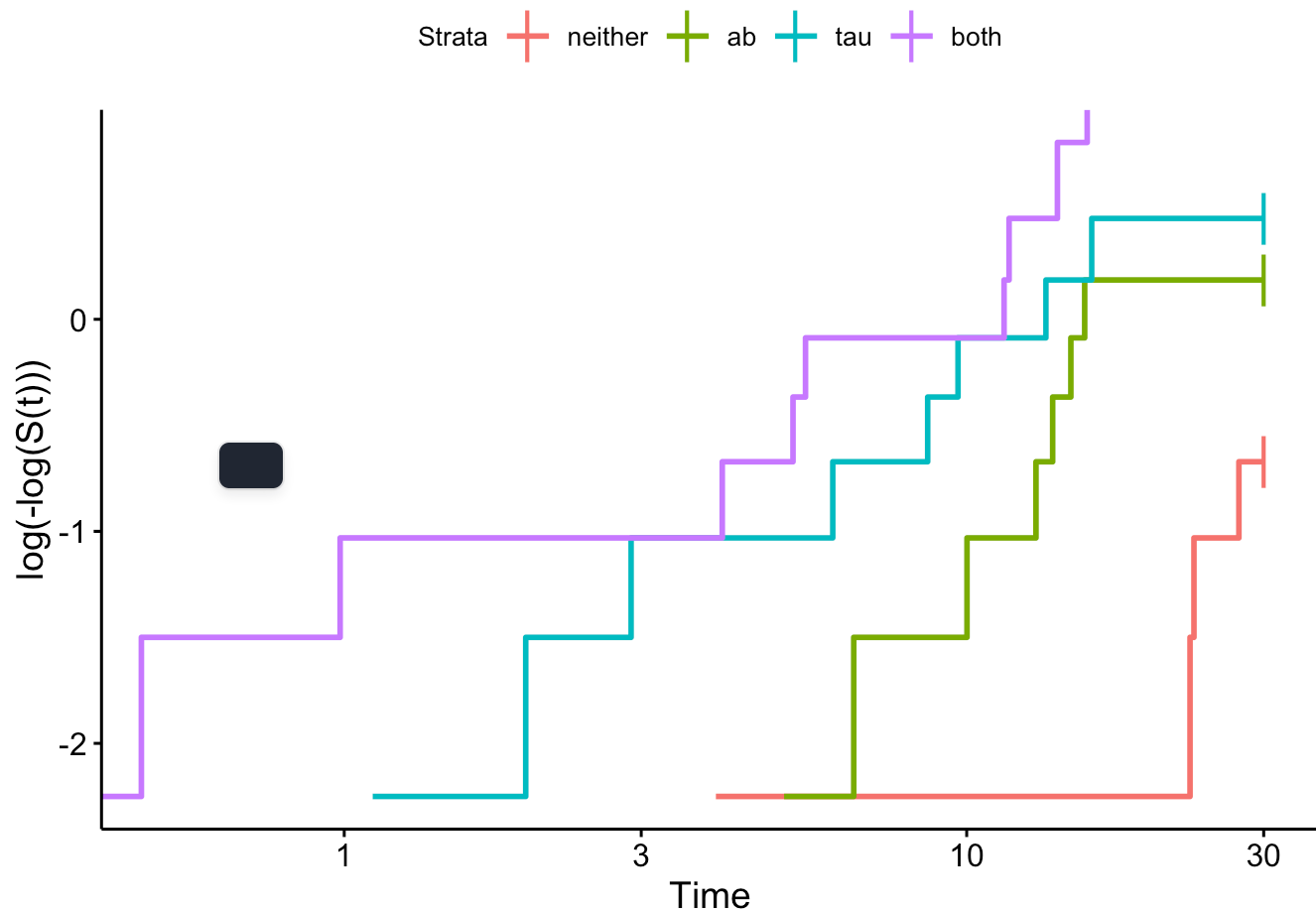
surv.curv <- surv_fit(formula = surv_table ~ ab40.tau.intxn, data = surv2)

ggsurvplot(fit = surv.curv, data = surv2,
            conf.int = FALSE,
            censor.shape = 124,
            censor.size = 7,
            legend.labs = c('neither', 'ab', 'tau', 'both'))
```

We can create the log minus log plot using basically the same `ggsurvplot` function call, however we will want to add `fun = 'cloglog'` and `xlim = c(0.5,30)` as well. The 'cloglog' causes both axes to be transformed, but we also have to exclude 0 from the x-axis since 0 cannot be log transformed. That is why we are limiting the values on the x-axis to be from 0.5 to 30 (our maximum time in our experiment).

```
ggsurvplot(surv.curv, data = surv2,
  conf.int = FALSE,
  censor.shape = 124,
  censor.size = 7,
  legend.labs = c('neither', 'ab', 'tau', 'both'),
  fun = 'cloglog',
  xlim = c(0.5, 30))
```



There is no substantial crossover in our graph among any of our lines, so none of the levels need to be dropped. If there is a large amount of crossover, the analysis would need to be done on an individual factor basis, i.e. as a function of tau and ab40 independently.

Statistical Test of Proportional Hazards

In order to test the proportional hazards assumption statistically, we will need to first create a Cox model using the `coxph` function. This function behaves exactly like the `surv_fit` function from above, though the right hand side will have ab40, tau, and their interaction (denoted with the `*` function).

```
cox.m <- coxph(formula = surv_table ~ ab40*tau, data = surv2)
```

Now, all we need to do for the statistical test is pass in the `cox.m` object into the `cox.zph` function.

```
cox.zph(cox.m)
```

```
##          chisq df    p
## ab40      0.31247  1 0.58
## tau       1.25575  1 0.26
## ab40:tau  0.00897  1 0.92
## GLOBAL    1.80474  3 0.61
```

This gives our test statistics and p-values for each of the individual terms in the model as well as a test for the overall model. Notice that here, none of the terms are significant, meaning the assumption of proportional hazards is met. If any of the terms are significant, they will need to be removed from the model. If there is a discrepancy between the graphical analysis and the statistical outcome, you can remove a term to be conservative, or consult a statistician. If you do remove a term from the model, you will need to recheck the survival curves without that term to make sure there is not any significant crossing again.

Running the Test

We have already created our model above using the `coxph` function, so we do not need to do it again since we are not removing any terms from the model. To view a summary of the model, just use the `summary` command with the cox model we just made.

```
summary(cox.m)
```

```
## Call:
## coxph(formula = surv_table ~ ab40 * tau, data = surv2)
##
##      n= 40, number of events= 29
##
##              coef exp(coef) se(coef)      z Pr(>|z|)
## ab40Yes          1.0042    2.7297  0.6338  1.584   0.1131
## tauYes           1.4641    4.3238  0.6203  2.360   0.0183 *
## ab40Yes:tauYes  -0.2375    0.7886  0.7877 -0.302   0.7630
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##              exp(coef) exp(-coef) lower .95 upper .95
## ab40Yes              2.7297      0.3663      0.7881      9.454
## tauYes               4.3238      0.2313      1.2819     14.584
## ab40Yes:tauYes       0.7886      1.2681      0.1684      3.692
##
## Concordance= 0.718 (se = 0.044 )
## Likelihood ratio test= 15.38 on 3 df,  p=0.002
## Wald test              = 13.83 on 3 df,  p=0.003
## Score (logrank) test = 16.72 on 3 df,  p=8e-04
```

As you can see, two tables are included in the output along with a few tests at the bottom. The top table presents the main effect of `ab40` and `tau` on survival, along with their interaction. For interpretation, just focus on the `exp(coef)` column, that gives the hazard ratio for that term, and the `Pr(>|z|)` column that gives the p-value for that effect. The second table repeats the hazard ratio, and presents the flipped hazard ratio along with a 95% confidence interval (for the non-flipped hazard ratio). At the bottom, we have results for the omnibus tests, which simply tell us if the overall model is significant or not.

So the interpretation of this model is: if a mouse expresses `tau`, they have a significantly higher chance of dying than if a mouse does not express `tau` regardless of the expression of `a-beta`. The odds of death are around 4.32 times higher for `tau` expressing mice than non-expressing mice (this is the hazard ratio).

Reporting the Results

“A Cox Regression proportional hazards analysis indicated a significant effect of tau on mortality (Chi-square(3) = 15.38, $p = 0.002$). When a mouse expressed tau, we observed a significant reduction in survival over the 30-day period regardless of a-beta expression [Hazard Ratio (95% CI): 4.3238 (1.2819, 14.5835)]. Regardless of tau expression, a-beta expression did not significantly affect mortality [1.0042 (0.7882, 9.4541)].”

Bootstrapping the Hazard Ratio

In this case, we are provided with a nice confidence interval that is derived from the analysis. Sometimes, this is not the case, and we may wish to estimate the confidence interval of a parameter using other means. The bootstrapping procedure allows us to calculate the confidence interval with basically no additional assumptions. Let's bootstrap the hazard ratio for tau as calculated in the Cox regression and see how the bootstrap confidence interval compares to the parametric derived one.

For this example, we will use the `sample_n` function from the `tidyverse`, a modified version of `sample` that samples full rows from a dataset. For a non-parametric bootstrap (there is also a parametric version, which we will not cover), you take a random sample of your data WITH replacement, keeping the total sample size the same each time. This implies that for each sample, some subjects will be present one or more times, and some will not be present at all. In this case, there were 40 total subjects, so we will randomly draw 40 subjects in each iteration. The `sample` function requires you to specify what is sampled (in this case, we will use the row numbers) and how many to sample (size). We will set `replace` to `TRUE`. We will then create a new dataset that uses the bootstrapped row numbers. In order to make this reproducible, we will set the seed as a random number (say 902) before running the loop. We will do 500 iterations in a `for` loop, and calculate the hazard ratio with each bootstrapped data set. We will store all of the hazard ratio estimates in a vector.

```
set.seed(902)

# create a variable to store our 500 hazard ratios in (we will start by setting all 500 to NA)
boot.HR.estimates <- rep(NA, 500)

# for each of 500 iterations ....
for(i in 1:500){
  # sample with replacement a number of samples equal to the number of rows in our dataset (i.e. 40)
  boot.surv2 <- sample_n(tbl = surv2, size = nrow(surv2), replace = TRUE)

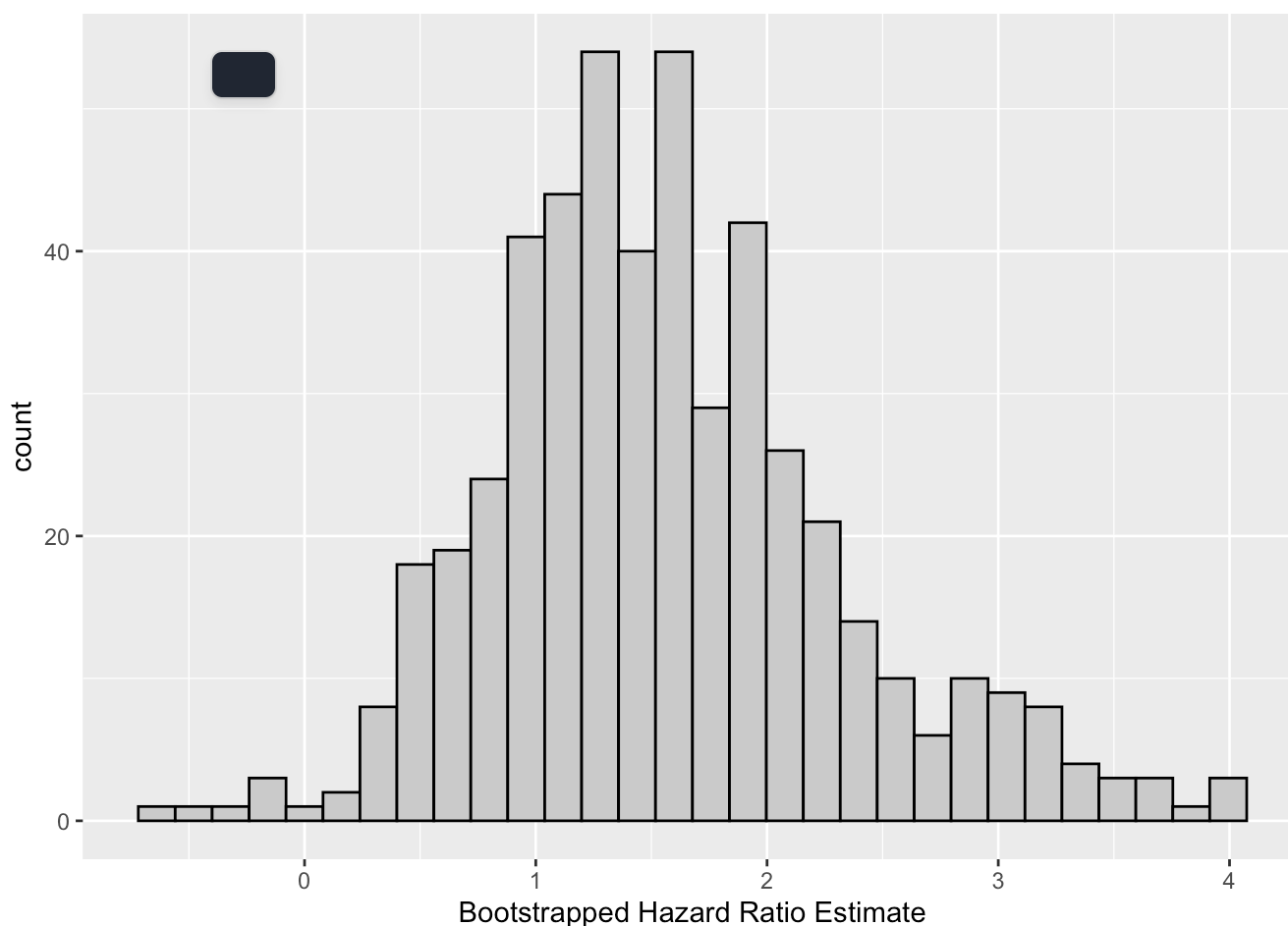
  # run a coxph model on the sampled data and store it in boot.cox.tau
  boot.cox.tau <- coxph(formula = Surv(time = stoptime, event = status.num) ~ ab40*tau,
    data = boot.surv2)

  # store the hazard ratio (extracted from the model as follows) as the "i"th value of the output
  boot.HR.estimates[i] <- boot.cox.tau$coefficients["tauYes"]
}
```

Notice that for each iteration, we create the `boot.surv2` dataset and the model results `boot.cox.tau`. For each iteration, we write over the previous iteration, so these are not stored in memory, but we do extract the hazard ratio each time.

Now that we have the distribution of the bootstrap hazard ratio estimates, let's look at it with a simple histogram. Notice that the data are maybe a bit right skewed, but overall fairly normally distributed.

```
ggplot(data = tibble(boot.HR.estimates), mapping = aes(x = boot.HR.estimates)) +  
  geom_histogram(color = "black", fill = "gray80") +  
  labs(x = "Bootstrapped Hazard Ratio Estimate")
```



To calculate the bootstrapped 95% CI of the hazard ratio, we will just extract the 2.5th and 97.5th percentiles (i.e. those with 95% of the observations lying in between), then we will exponentiate the result to get the hazard ratios themselves.

```
exp(quantile(boot.HR.estimates, c(0.025, 0.975)))
```

```
##      2.5%      97.5%  
## 1.358238 28.415726
```

Notice that the upper end of the confidence interval is quite a bit larger with the bootstrap estimate than compared to the one from the Cox model (which was around 1.28 to 14.6). This likely has to do with the underlying bootstrap distribution being skewed. There are ways to derive more valid estimates, but we will not go into them as they require some more complex coding.

Independent Practice

For this experiment, we are testing whether there is a difference in survival rate between treatments A and B using a Cox proportional hazards model. Use the dataset `ex` in the `Survival_Data_2020.RData` file.

1. Describe your variables (e.g., independent vs dependent, continuous or categorical)
2. Discuss the primary assumptions of this test, and determine whether they are met.
3. Regardless of whether assumptions are met, run a Cox Proportional hazards analysis and determine the hazard ratio estimate for treatment B. Report your findings as you would in a Results section.

Next, do the following:

1. Create a distribution of 100 values with mean 0 and sd 1 using the `rnorm` function. Make it reproducible by setting the seed prior to simulating the data.
2. Calculate the confidence interval of the mean using a bootstrap with 10,000 iterations. You will need to use the `sample` function instead of `sample_n`.
3. Calculate the 95% CI of the mean using a parametric z-score approach ($\text{mean} \pm 1.96 * \text{standard error}$) and compare the CI to the bootstrap CI. How do these compare?
 - **Hint:** You can use the `se()` function from the `sciplot` package to calculate standard error