432 Class 22

https://thomaselove.github.io/432-2025

2025-04-03

## Today’s Topic: Time-to-Event Data

* Before Spring Break, we discussed
  + Kaplan-Meier Estimation of the Survival Function
  + Creating Survival Objects, Drawing Survival Curves
  + Testing the difference between Survival Curves
* Today: start Cox Proportional Hazards Regression
  + The Hazard Function and its Estimation

See Chapters 29-31 of our [Course Notes](https://thomaselove.github.io/432-notes/)

## Today’s R Setup

knitr::opts\_chunk$set(comment=NA)  
  
library(janitor)  
library(naniar)  
library(haven)  
library(here)  
library(conflicted)  
library(broom)  
library(gt)  
library(rms)  
library(survival)  
library(survminer)  
library(easystats)  
library(tidyverse)  
  
theme\_set(theme\_bw())

## The Stanford Heart Transplant Study

The Stanford University Heart Transplant Study[[1]](#footnote-23) examined whether an experimental heart transplant program increased lifespan. The heart\_tr.sav data (saved as an SPSS file) includes 103 observations on 7 variables, as we’ll see.

* The data include survtime, which is the number of days the subject was alive after the date they were determined to be a candidate for heart transplant until the termination of the study.

Our first step is to ingest the data, and build a tibble.

## What’s in the heart\_tr data?

| Variable | Description |
| --- | --- |
| id | Random last name, assigned to the subject by Dr. Love |
| age | subject’s age in years at the start of the study |
| survived | survival status (alive or dead) at end of follow-up |
| survtime | days subject was alive from the start of the study to its end |
| prior | did the subject have a prior surgery (yes or no) |
| transplant | treatment (received a transplant) or control (did not) |
| wait | waiting time for transplant (in days) |

## Ingesting data from an SPSS file

We’ll use the read\_spss() function[[2]](#footnote-27), from the **haven** package.

heart\_tr <- read\_spss(here("c22/data/heart\_tr.sav"))  
  
dim(heart\_tr)

[1] 103 7

names(heart\_tr)

[1] "id" "age" "survived" "survtime" "prior"   
[6] "transplant" "wait"

## Looking at the heart\_tr data

heart\_tr

# A tibble: 103 × 7  
 id age survived survtime prior transplant wait  
 <chr> <dbl> <dbl+lbl> <dbl> <dbl+lbl> <dbl+lbl> <dbl>  
 1 Akter 51 2 [dead] 6 2 [no] 2 [control] NA  
 2 Alcorn 30 2 [dead] 50 2 [no] 2 [control] NA  
 3 Ali 33 1 [alive] 1799 2 [no] 1 [treatment] 25  
 4 Alway 40 2 [dead] 39 2 [no] 1 [treatment] 36  
 5 Amadeo 20 2 [dead] 18 2 [no] 2 [control] NA  
 6 Aybar 54 2 [dead] 3 2 [no] 2 [control] NA  
 7 Bargiel 50 2 [dead] 675 2 [no] 1 [treatment] 51  
 8 Bayse 45 2 [dead] 40 2 [no] 2 [control] NA  
 9 Beason 47 2 [dead] 85 2 [no] 2 [control] NA  
10 Boddicker 42 2 [dead] 58 2 [no] 1 [treatment] 12  
# ℹ 93 more rows

* Note the <dbl+lbl> types for the columns containing categorical information.

## What are the labels for survived?

* Here are two options for figuring this out:

heart\_tr |> count(survived) # numeric codes, with labels

# A tibble: 2 × 2  
 survived n  
 <dbl+lbl> <int>  
1 1 [alive] 28  
2 2 [dead] 75

print\_labels(heart\_tr$survived)

Labels:  
 value label  
 1 alive  
 2 dead

* Suppose we want to use a factor, without these specialized value labels, to represent the information in survived.

## Converting survived into a factor, without labels

heart\_tr <- heart\_tr |>  
 mutate(survived = fct\_recode(factor(survived),   
 "alive" = "1", "dead" = "2"))  
  
heart\_tr |> count(survived) # now a factor, without labels

# A tibble: 2 × 2  
 survived n  
 <fct> <int>  
1 alive 28  
2 dead 75

Now, we have a factor representation of the survived information, with values that make sense.

## Converting prior into a factor

heart\_tr |> count(prior) # numeric codes, with labels

# A tibble: 2 × 2  
 prior n  
 <dbl+lbl> <int>  
1 1 [yes] 12  
2 2 [no] 91

heart\_tr <- heart\_tr |>  
 mutate(prior = fct\_recode(factor(prior), "yes" = "1", "no" = "2"))

* and now, we have a meaningful factor for prior, too.

heart\_tr |> count(prior) # now a factor, without labels

# A tibble: 2 × 2  
 prior n  
 <fct> <int>  
1 yes 12  
2 no 91

## tranplant into a 0/1 indicator

Instead of making transplant into a factor, we’ll create a numeric description of the transplant information, where transplant = 1 will mean that the subject did have a heart transplant, and transplant = 0 will mean that the subject did not have a heart transplant.

heart\_tr |> count(transplant) # numeric codes, with labels

# A tibble: 2 × 2  
 transplant n  
 <dbl+lbl> <int>  
1 1 [treatment] 69  
2 2 [control] 34

## Converting transplant to 0/1

heart\_tr <- heart\_tr |>  
 mutate(transplant = as.numeric(transplant == 1))  
  
heart\_tr |> count(transplant) # 1/0, no labels

# A tibble: 2 × 2  
 transplant n  
 <dbl> <int>  
1 0 34  
2 1 69

## heart\_tr after our changes

glimpse(heart\_tr)

Rows: 103  
Columns: 7  
$ id <chr> "Akter", "Alcorn", "Ali", "Alway", "Amadeo", "Aybar", "Barg…  
$ age <dbl> 51, 30, 33, 40, 20, 54, 50, 45, 47, 42, 47, 53, 54, 53, 53,…  
$ survived <fct> dead, dead, alive, dead, dead, dead, dead, dead, dead, dead…  
$ survtime <dbl> 6, 50, 1799, 39, 18, 3, 675, 40, 85, 58, 153, 8, 81, 1386, …  
$ prior <fct> no, no, no, no, no, no, no, no, no, no, no, no, no, no, no,…  
$ transplant <dbl> 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1,…  
$ wait <dbl> NA, NA, 25, 36, NA, NA, 51, NA, NA, 12, 26, NA, 17, 37, NA,…

* So we no longer have any labels, and our categorical variables are presented as factors (survived and prior) or as a 1/0 numeric variable (transplant.)
* Because our subject identifier id was a set of (fake) last names, rather than numbers, this is already a character variable, which is what we want. Are they unique?

identical(nrow(heart\_tr), n\_distinct(heart\_tr$id))

[1] TRUE

## Creating a Survival Object

* survtime shows the in-study time (in days) until death or censoring
* survived is a factor showing “dead” or “alive”.

heart\_tr$S <- Surv(time = heart\_tr$survtime,   
 event = heart\_tr$survived == "dead")  
  
head(heart\_tr$S)

[1] 6 50 1799+ 39 18 3

* The first subject died after 6 days, while the second died after 50 days. The third subject was censored after 1799 days.

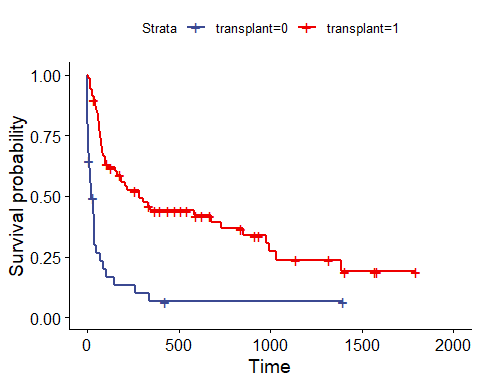
## Compare the two transplant groups

km\_tr <- survfit(S ~ transplant, data = heart\_tr)  
  
km\_tr

Call: survfit(formula = S ~ transplant, data = heart\_tr)  
  
 n events median 0.95LCL 0.95UCL  
transplant=0 34 30 21 12 50  
transplant=1 69 45 285 153 852

## Plotting the K-M Curves

ggsurvplot(km\_tr, data = heart\_tr, palette = "aaas")



## Kaplan-Meier Estimates (table)

summary(km\_tr)

Call: survfit(formula = S ~ transplant, data = heart\_tr)  
  
 transplant=0   
 time n.risk n.event survival std.err lower 95% CI upper 95% CI  
 1 34 1 0.9706 0.0290 0.9154 1.000  
 2 33 3 0.8824 0.0553 0.7804 0.998  
 3 30 3 0.7941 0.0693 0.6692 0.942  
 5 27 1 0.7647 0.0727 0.6346 0.921  
 6 26 2 0.7059 0.0781 0.5682 0.877  
 8 24 1 0.6765 0.0802 0.5362 0.853  
 9 23 1 0.6471 0.0820 0.5048 0.829  
 12 21 1 0.6162 0.0836 0.4723 0.804  
 16 20 1 0.5854 0.0849 0.4405 0.778  
 18 19 1 0.5546 0.0859 0.4094 0.751  
 21 18 2 0.4930 0.0867 0.3493 0.696  
 32 15 1 0.4601 0.0869 0.3177 0.666  
 35 14 1 0.4273 0.0867 0.2871 0.636  
 36 13 1 0.3944 0.0860 0.2572 0.605  
 37 12 1 0.3615 0.0849 0.2281 0.573  
 40 11 2 0.2958 0.0812 0.1727 0.507  
 50 9 1 0.2629 0.0786 0.1464 0.472  
 69 8 1 0.2301 0.0753 0.1211 0.437  
 85 7 1 0.1972 0.0714 0.0970 0.401  
 102 6 1 0.1643 0.0666 0.0743 0.364  
 149 5 1 0.1315 0.0609 0.0531 0.326  
 263 4 1 0.0986 0.0538 0.0338 0.287  
 340 3 1 0.0657 0.0448 0.0173 0.250  
  
 transplant=1   
 time n.risk n.event survival std.err lower 95% CI upper 95% CI  
 5 69 1 0.986 0.0144 0.9577 1.000  
 16 68 2 0.957 0.0246 0.9096 1.000  
 17 66 1 0.942 0.0281 0.8885 0.999  
 28 65 1 0.928 0.0312 0.8683 0.991  
 30 64 1 0.913 0.0339 0.8489 0.982  
 39 63 1 0.899 0.0363 0.8301 0.973  
 43 61 1 0.884 0.0386 0.8113 0.963  
 45 60 1 0.869 0.0407 0.7929 0.953  
 51 59 1 0.854 0.0426 0.7748 0.942  
 53 58 1 0.840 0.0443 0.7571 0.931  
 58 57 1 0.825 0.0459 0.7396 0.920  
 61 56 1 0.810 0.0474 0.7224 0.909  
 66 55 1 0.795 0.0488 0.7053 0.897  
 68 54 2 0.766 0.0512 0.6719 0.873  
 72 52 2 0.737 0.0533 0.6391 0.849  
 77 50 1 0.722 0.0543 0.6229 0.836  
 78 49 1 0.707 0.0551 0.6069 0.824  
 80 48 1 0.692 0.0559 0.5910 0.811  
 81 47 1 0.678 0.0566 0.5752 0.798  
 90 46 1 0.663 0.0573 0.5596 0.785  
 96 45 1 0.648 0.0579 0.5441 0.772  
 100 44 1 0.633 0.0584 0.5287 0.759  
 110 42 1 0.618 0.0589 0.5130 0.745  
 153 40 1 0.603 0.0594 0.4969 0.731  
 165 39 1 0.587 0.0599 0.4810 0.717  
 186 37 1 0.572 0.0603 0.4647 0.703  
 188 36 1 0.556 0.0607 0.4485 0.688  
 207 35 1 0.540 0.0610 0.4325 0.674  
 219 34 1 0.524 0.0613 0.4166 0.659  
 285 32 2 0.491 0.0616 0.3840 0.628  
 308 30 1 0.475 0.0617 0.3680 0.613  
 334 29 1 0.458 0.0617 0.3521 0.597  
 342 27 1 0.441 0.0617 0.3356 0.581  
 583 20 1 0.419 0.0625 0.3132 0.562  
 675 16 1 0.393 0.0638 0.2860 0.540  
 733 15 1 0.367 0.0647 0.2597 0.519  
 852 13 1 0.339 0.0656 0.2317 0.495  
 979 10 1 0.305 0.0672 0.1979 0.470  
 995 9 1 0.271 0.0678 0.1660 0.442  
 1032 8 1 0.237 0.0672 0.1360 0.413  
 1386 5 1 0.190 0.0685 0.0935 0.385

## How about a log rank test?

survdiff(S ~ transplant, data = heart\_tr)

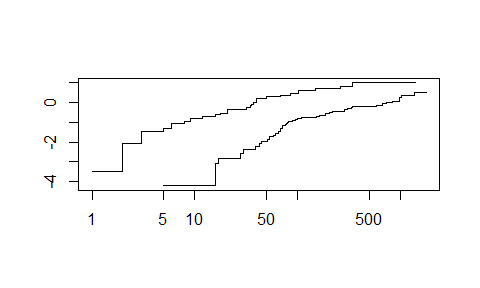
Call:  
survdiff(formula = S ~ transplant, data = heart\_tr)  
  
 N Observed Expected (O-E)^2/E (O-E)^2/V  
transplant=0 34 30 12.1 26.5 33.2  
transplant=1 69 45 62.9 5.1 33.2  
  
 Chisq= 33.2 on 1 degrees of freedom, p= 8e-09

* What can we conclude from this result?

## Log-Log Plot for K-M estimation

* The two curves do not meet during the observation period, indicating the satisfaction of the proportional hazard assumption made by the log rank test.

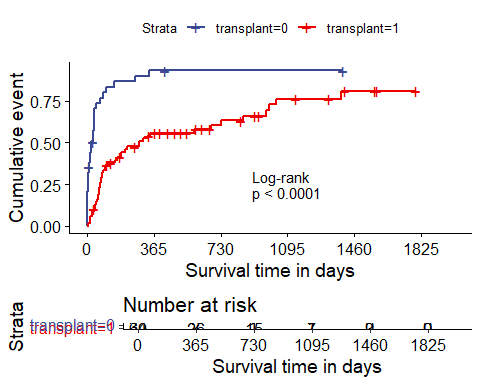
plot(survfit(S ~ transplant, data = heart\_tr), col = c(1), fun = "cloglog")



## Cumulative Event Rate for km\_tr

* Add fun = “event” to our ggsurvplot(), and
* A table of subjects at risk over time, and
* The p value from the log rank test.

ggsurvplot(km\_tr, data = heart\_tr, palette = "aaas",  
 fun = "event",  
 xlab = "Survival time in days",   
 break.time.by = 365,  
 risk.table = TRUE, risk.table.height = 0.25,  
 pval = TRUE, pval.method = TRUE,   
 pval.size = 4, pval.method.size = 4,  
 pval.coord = c(900, 0.20), pval.method.coord = c(900, 0.30))



## The Hazard Function H(t)

If S(t) is the survival function, and time t is taken to be continuous, then the **hazard function** H(t) is defined as:

* H(t) is used to describe the concept of the risk of “failure” in an interval after time t, conditioned on the subject having survived to time t.
* H(t) is the **cumulative** hazard function, to emphasize that its value is the “sum” of the hazard up to time t.

## Understaning the Hazard Function

Consider a subject in the heart transplant study who has a survival time of 1000 days. Let’s ignore the transplant group information for a moment.

* For this subject to die at 1000 days, they had to survive for the first 999 days.
* The subject’s hazard at 1000 days is the failure rate “per day” conditional on the subject being alive through the first 999 days.

## Estimating the Hazard Function

Suppose we want to estimate H(t) across all subjects.

* There are several different methods, but we’ll focus on the inverse Kaplan-Meier estimator.

I’ll build something called H.est1, the inverse K-M estimate…

km\_1 <- survfit(S ~ 1, data = heart\_tr)  
  
Haz1.almost <- -log(km\_1$surv)  
  
H\_est1 <- c(Haz1.almost, tail(Haz1.almost, 1))

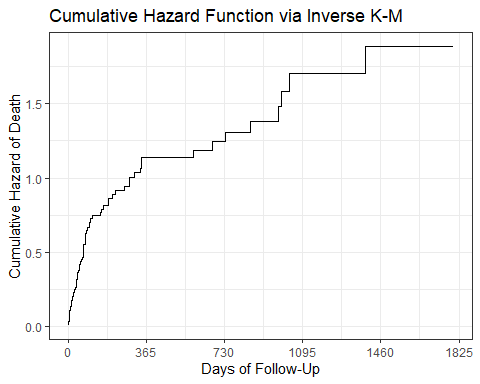
## Tibble of times and hazard estimates

haz\_tib <- tibble(  
 time = c(km\_1$time, tail(km\_1$time, 1)),  
 inverse\_KM = H\_est1)  
  
haz\_tib

# A tibble: 89 × 2  
 time inverse\_KM  
 <dbl> <dbl>  
 1 1 0.00976  
 2 2 0.0396   
 3 3 0.0704   
 4 5 0.0914   
 5 6 0.113   
 6 8 0.124   
 7 9 0.135   
 8 11 0.135   
 9 12 0.146   
10 16 0.181   
# ℹ 79 more rows

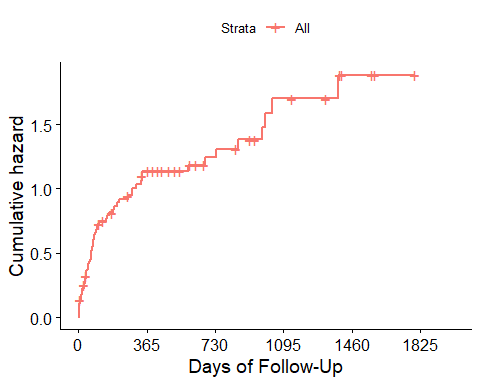
## Cumulative Hazard Function from Inverse Kaplan-Meier Estimates

ggplot(haz\_tib, aes(x = time, y = inverse\_KM)) +   
 geom\_step() +   
 scale\_x\_continuous(breaks = c(0, 365, 730, 1095, 1460, 1825)) +  
 labs(x = "Days of Follow-Up",   
 y = "Cumulative Hazard of Death",  
 title = "Cumulative Hazard Function via Inverse K-M")



## Cumulative Hazard Function from Inverse Kaplan-Meier Estimates

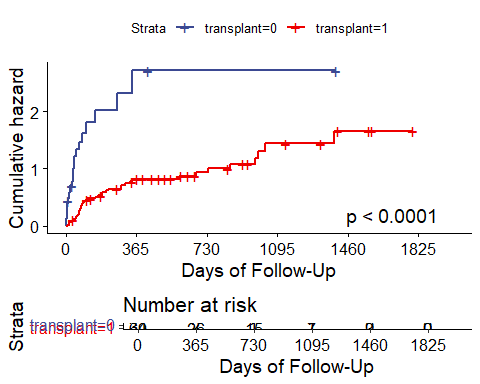
ggsurvplot(km\_1, data = heart\_tr, fun = "cumhaz", conf.int = FALSE,  
 xlab = "Days of Follow-Up", break.time.by = 365)



## Plotting Cumulative Hazard by Transplant Group

For our km\_tr fit, we’d use

ggsurvplot(km\_tr, data = heart\_tr, fun = "cumhaz",  
 xlab = "Days of Follow-Up", palette = "aaas",  
 pval = TRUE, pval.coord = c(1450, 0.2),  
 break.time.by = 365,  
 risk.table = TRUE, risk.table.height = 0.25)



## Cox Proportional Hazards Regression

fit1\_tr <- coxph(S ~ transplant, data = heart\_tr)

The Cox proportional hazards model fits survival data with a constant (not varying over time) covariate (here, transplant group) to a hazard function of the form:

where we estimate the unknown value of and where is the baseline hazard which depends on time but not on the transplant group.

## Cox Model fit1\_tr

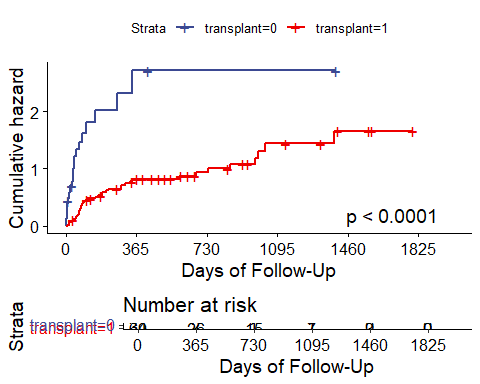
fit1\_tr

Call:  
coxph(formula = S ~ transplant, data = heart\_tr)  
  
 coef exp(coef) se(coef) z p  
transplant -1.3234 0.2662 0.2438 -5.428 5.69e-08  
  
Likelihood ratio test=25.95 on 1 df, p=3.511e-07  
n= 103, number of events= 75

Our hazard ratio estimate is 0.2662 for transplant group 1 (vs. transplant group 0)

* Hazard ratio < 1 indicates a decrease in hazard for subjects who received a transplant as compared to those who did not. Does this match our plot (repeated on next slide)?

## Plotting Cumulative Hazard by Transplant Group



## Cox Model Parameters (fit1\_tr)

model\_parameters(fit1\_tr, pretty\_names = FALSE, ci = 0.90, digits = 3)

Parameter | Coefficient | SE | 90% CI | z | p  
---------------------------------------------------------------------  
transplant | -1.323 | 0.244 | [-1.724, -0.922] | -5.428 | < .001

Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed  
 using a Wald z-distribution approximation.

model\_parameters(fit1\_tr, pretty\_names = FALSE, ci = 0.90, digits = 3,  
 exponentiate = TRUE)

Parameter | Coefficient | SE | 90% CI | z | p  
-------------------------------------------------------------------  
transplant | 0.266 | 0.065 | [0.178, 0.398] | -5.428 | < .001

Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed  
 using a Wald z-distribution approximation.

* Compare to tidy() results from **broom**?

tidy(fit1\_tr, exponentiate = TRUE, conf.int = TRUE, conf.level = 0.90) |>  
 gt() |> tab\_options(table.font.size = 20) |>   
 fmt\_number(decimals = 3) |> opt\_stylize(style = 4, color = "blue")

| term | estimate | std.error | statistic | p.value | conf.low | conf.high |
| --- | --- | --- | --- | --- | --- | --- |
| transplant | 0.266 | 0.244 | -5.428 | 0.000 | 0.178 | 0.398 |

## Cox Model fit1\_tr Performance

model\_performance(fit1\_tr)

Response residuals not available to calculate mean square error. (R)MSE  
 is probably not reliable.

Warning: Can't calculate weighted residuals from model.

# Indices of model performance  
  
AIC | AICc | BIC | Nagelkerke's R2 | RMSE | Sigma  
-------------------------------------------------------------  
572.297 | 572.336 | 574.931 | 0.223 | 0.947 | 0.000

* How about glance() from **broom**?

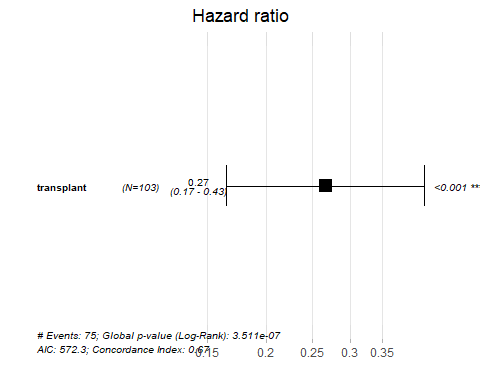
glance(fit1\_tr) |> gt() |> fmt\_number(decimals = 2)

| n | nevent | statistic.log | p.value.log | statistic.sc | p.value.sc | statistic.wald | p.value.wald | statistic.robust | p.value.robust | r.squared | r.squared.max | concordance | std.error.concordance | logLik | AIC | BIC | nobs |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 103.00 | 75.00 | 25.95 | 0.00 | 33.38 | 0.00 | 29.47 | 0.00 | NA | NA | 0.22 | 1.00 | 0.67 | 0.02 | -285.15 | 572.30 | 574.61 | 75.00 |

## Forest Plot for fit1\_tr model

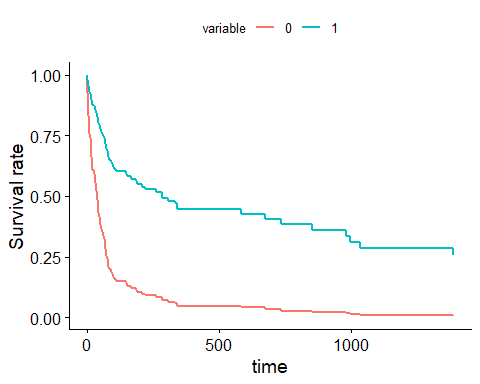
ggforest(fit1\_tr)

Warning in .get\_data(model, data = data): The `data` argument is not provided.  
Data will be extracted from model fit.



## Plot Adjusted Survival Curves

ggadjustedcurves(fit1\_tr, data = data.frame(heart\_tr),   
 method = "average", variable = "transplant")



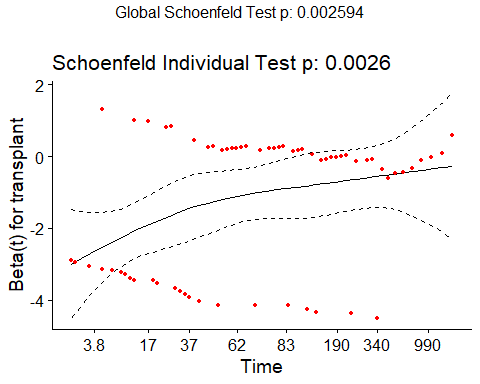
## Checking the Proportional Hazards Assumption

If the proportional hazards assumption is appropriate, we should see a slope of essentially zero in the residuals that are plotted against time on the next slide.

* If we see a slope that seriously different from zero, that will suggest a violation of the proportional hazards assumption.
* A hypothesis test is also performed, where a small p value indicates a potential problem with the assumption.

## Plot to Check Proportional Hazards

ggcoxzph(cox.zph(fit1\_tr))



## Cox Model Diagnostics (fit1\_tr)

ggcoxdiagnostics(fit1\_tr)

`geom\_smooth()` using formula = 'y ~ x'

Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
: pseudoinverse used at -1.3301

Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
: neighborhood radius 1.3301

Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
: reciprocal condition number 0

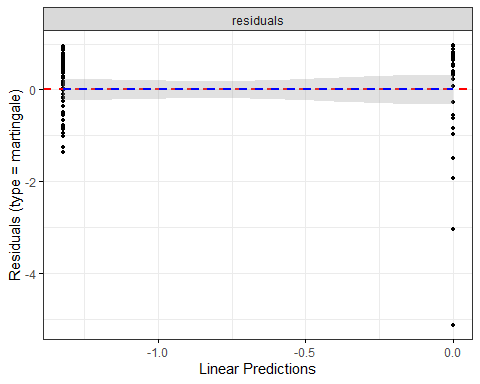
Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
: There are other near singularities as well. 1.7691

Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
else if (is.data.frame(newdata))  
as.matrix(model.frame(delete.response(terms(object)), : pseudoinverse used at  
-1.3301

Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
else if (is.data.frame(newdata))  
as.matrix(model.frame(delete.response(terms(object)), : neighborhood radius  
1.3301

Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
else if (is.data.frame(newdata))  
as.matrix(model.frame(delete.response(terms(object)), : reciprocal condition  
number 0

Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
else if (is.data.frame(newdata))  
as.matrix(model.frame(delete.response(terms(object)), : There are other near  
singularities as well. 1.7691



## What if we also include age?

fit2\_tr <- coxph(S ~ transplant + age, data = heart\_tr)  
  
fit2\_tr

Call:  
coxph(formula = S ~ transplant + age, data = heart\_tr)  
  
 coef exp(coef) se(coef) z p  
transplant -1.78994 0.16697 0.27107 -6.603 4.02e-11  
age 0.06020 1.06205 0.01527 3.943 8.03e-05  
  
Likelihood ratio test=44.55 on 2 df, p=2.121e-10  
n= 103, number of events= 75

## fit2\_tr model parameters

model\_parameters(fit2\_tr, exponentiate = TRUE, pretty\_names = FALSE,   
 ci = 0.90, digits = 3)

Parameter | Coefficient | SE | 90% CI | z | p  
-------------------------------------------------------------------  
transplant | 0.167 | 0.045 | [0.107, 0.261] | -6.603 | < .001  
age | 1.062 | 0.016 | [1.036, 1.089] | 3.943 | < .001

Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed  
 using a Wald z-distribution approximation.

* If Harry is a year older than Steve and each are in the same transplant group, then Harry’s hazard of death is 1.062 (90% CI 1.036, 1.089) times that of Steve.
* If Harry and Sally are the same age, but Sally received a transplant but Harry did not, then Sally’s hazard of death is 0.167 (90% CI 0.107, 0.261) times that of Harry.

## fit2\_tr R-square measures

model\_performance(fit2\_tr)

Response residuals not available to calculate mean square error. (R)MSE  
 is probably not reliable.

Warning: Can't calculate weighted residuals from model.

# Indices of model performance  
  
AIC | AICc | BIC | Nagelkerke's R2 | RMSE | Sigma  
-------------------------------------------------------------  
555.695 | 555.815 | 560.965 | 0.352 | 0.880 | 0.000

* model\_parameters() gives the Nagelkerke .

glance(fit2\_tr) |> select(n, nevent, nobs, r.squared, r.squared.max) |>  
 gt() |> fmt\_number(columns = r.squared:r.squared.max, decimals = 3) |>  
 tab\_options(table.font.size = 20) |> opt\_stylize(style = 5, color = "blue")

| n | nevent | nobs | r.squared | r.squared.max |
| --- | --- | --- | --- | --- |
| 103 | 75 | 75 | 0.351 | 0.997 |

* glance() gives the Cox-Snell along with its maximum value (< 1.)

## fit2\_tr concordance measure

glance(fit2\_tr) |> select(n, nevent, nobs, concordance,   
 se\_conc = std.error.concordance) |>  
 gt() |> fmt\_number(columns = concordance:se\_conc, decimals = 3) |>  
 tab\_options(table.font.size = 20) |> opt\_stylize(style = 5, color = "blue")

| n | nevent | nobs | concordance | se\_conc |
| --- | --- | --- | --- | --- |
| 103 | 75 | 75 | 0.721 | 0.034 |

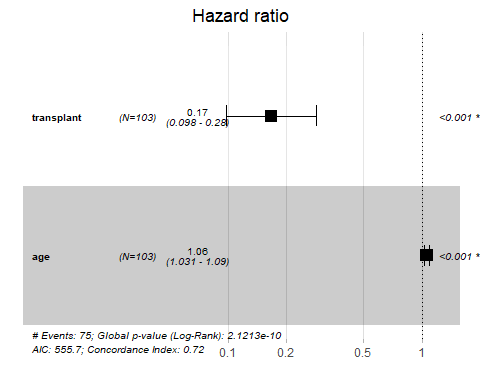
Compare the model’s prediction for a pair of observations in the data. A pair is concordant if the prediction and data agree in direction. Concordance is the fraction of pairs that are concordant.

* Higher Concordance = better Cox model predictions.

## Forest Plot for fit2\_tr model

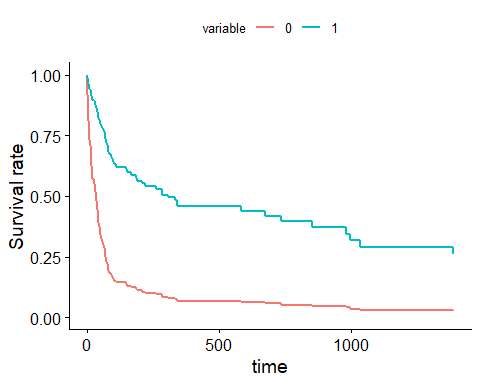
ggforest(fit2\_tr)

Warning in .get\_data(model, data = data): The `data` argument is not provided.  
Data will be extracted from model fit.



## Plot Adjusted Survival Curves

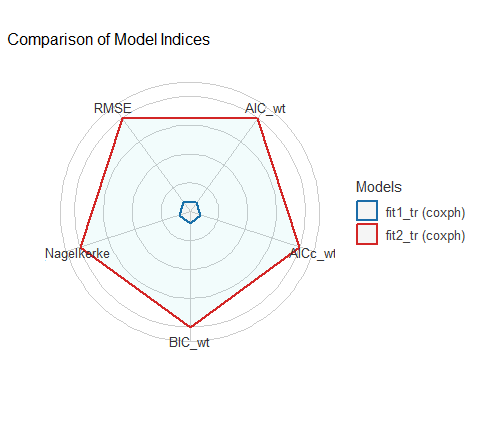
ggadjustedcurves(fit2\_tr, data = data.frame(heart\_tr),  
 method = "average", variable = "transplant")



## Effect of Adding age?

* fit1\_tr includes transplant group, fit2\_tr adds age.

plot(compare\_performance(fit1\_tr, fit2\_tr))



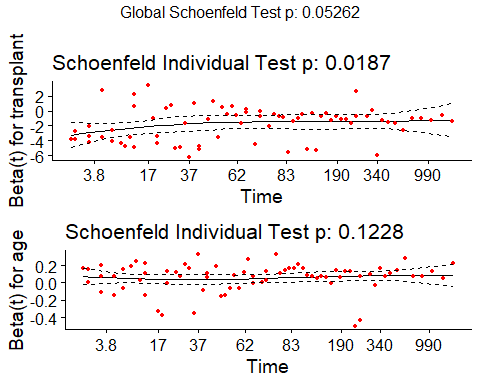
## ANOVA comparing fit1\_tr to fit2\_tr

anova(fit1\_tr, fit2\_tr)

Analysis of Deviance Table  
 Cox model: response is S  
 Model 1: ~ transplant  
 Model 2: ~ transplant + age  
 loglik Chisq Df Pr(>|Chi|)   
1 -285.15   
2 -275.85 18.602 1 1.611e-05 \*\*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Plot to Check Proportional Hazards

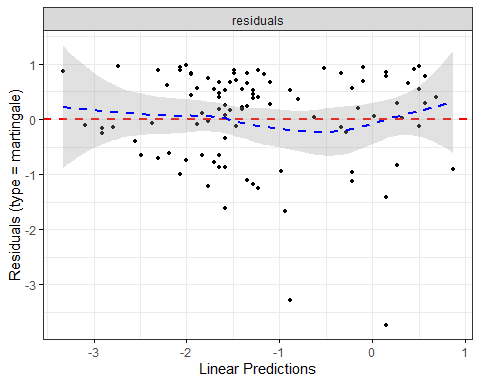
ggcoxzph(cox.zph(fit2\_tr))



## Cox Model Diagnostics (fit2\_tr)

ggcoxdiagnostics(fit2\_tr)

`geom\_smooth()` using formula = 'y ~ x'



## Model fit3\_tr adding prior surgery

fit3\_tr <- coxph(S ~ transplant + age + prior, data = heart\_tr)  
  
fit3\_tr

Call:  
coxph(formula = S ~ transplant + age + prior, data = heart\_tr)  
  
 coef exp(coef) se(coef) z p  
transplant -1.66121 0.18991 0.27588 -6.021 1.73e-09  
age 0.05919 1.06098 0.01494 3.962 7.44e-05  
priorno 0.74266 2.10152 0.44225 1.679 0.0931  
  
Likelihood ratio test=47.89 on 3 df, p=2.247e-10  
n= 103, number of events= 75

## fit3\_tr model parameters

model\_parameters(fit3\_tr, exponentiate = TRUE, pretty\_names = FALSE,   
 ci = 0.90, digits = 3)

Parameter | Coefficient | SE | 90% CI | z | p  
-------------------------------------------------------------------  
transplant | 0.190 | 0.052 | [0.121, 0.299] | -6.021 | < .001  
age | 1.061 | 0.016 | [1.035, 1.087] | 3.962 | < .001  
priorno | 2.102 | 0.929 | [1.015, 4.350] | 1.679 | 0.093

Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed  
 using a Wald z-distribution approximation.

## fit3\_tr R-square measures

model\_performance(fit3\_tr)

Response residuals not available to calculate mean square error. (R)MSE  
 is probably not reliable.

Warning: Can't calculate weighted residuals from model.

# Indices of model performance  
  
AIC | AICc | BIC | Nagelkerke's R2 | RMSE | Sigma  
-------------------------------------------------------------  
554.352 | 554.595 | 562.256 | 0.373 | 0.893 | 0.000

glance(fit3\_tr) |> select(n, nevent, nobs, r.squared, r.squared.max) |>  
 gt() |> fmt\_number(columns = r.squared:r.squared.max, decimals = 3) |>  
 tab\_options(table.font.size = 20) |> opt\_stylize(style = 5, color = "blue")

| n | nevent | nobs | r.squared | r.squared.max |
| --- | --- | --- | --- | --- |
| 103 | 75 | 75 | 0.372 | 0.997 |

## fit3\_tr concordance

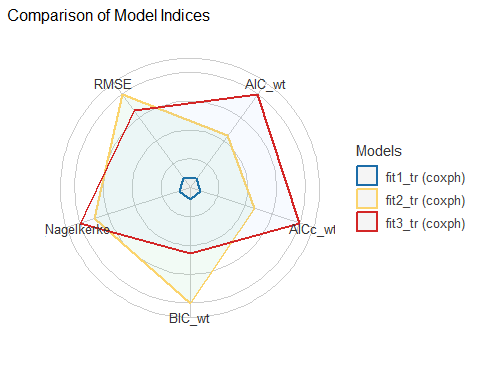
glance(fit3\_tr) |> select(n, nevent, nobs, concordance,   
 se\_conc = std.error.concordance) |>  
 gt() |> fmt\_number(columns = concordance:se\_conc, decimals = 3) |>  
 tab\_options(table.font.size = 20) |> opt\_stylize(style = 5, color = "blue")

| n | nevent | nobs | concordance | se\_conc |
| --- | --- | --- | --- | --- |
| 103 | 75 | 75 | 0.739 | 0.031 |

* Assesses probability of agreement between survival time and the risk score generated by the predictors
* 1 indicates perfect agreement, 0.5 indicates no better than chance

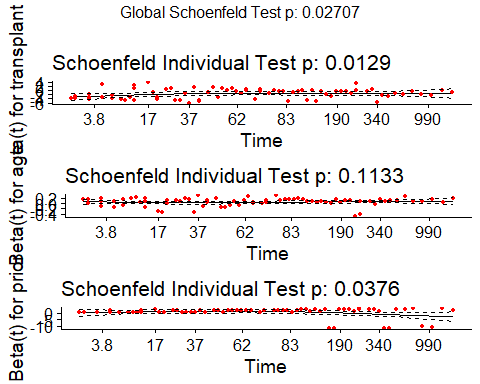
## Compare our 3 models

plot(compare\_performance(fit1\_tr, fit2\_tr, fit3\_tr))



## Checking PH Assumption for fit3\_tr

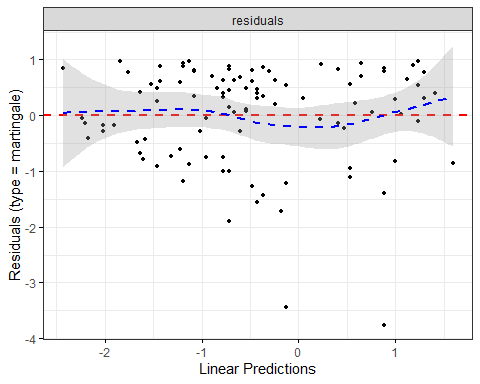
ggcoxzph(cox.zph(fit3\_tr))



## Cox Model Diagnostics for fit3\_tr

ggcoxdiagnostics(fit3\_tr)

`geom\_smooth()` using formula = 'y ~ x'



## What happens if we see a violation?

* We could add a non-linear predictor term or use a different kind of survival model.
* If the PH assumption fails on a categorical predictor, fit a Cox model stratified by that predictor (use strata(var) rather than var in the specification of the coxph model.)
* If the PH assumption is violated, this means the hazard isn’t constant over time, so we could fit separate Cox models for a series of time intervals.

## If we see a violation…

Another option would be to use an extension of the Cox model that permits covariates to vary over time.

Visit <https://cran.r-project.org/web/packages/survival/vignettes/timedep.pdf> for details on building the relevant data sets and models, with examples.

1. See <https://www.openintro.org/data/index.php?data=heart_transplant> for more details [↑](#footnote-ref-23)
2. We could also use read\_sav(), also from **haven**. [↑](#footnote-ref-27)