Part III: Survival Models

Load Packages

Again, we must load the packages that will be used in the first part of this workshop.

```r
library(pastecs, quietly = TRUE)
library(lm.beta,  quietly = TRUE)
library(lmtest,  quietly = TRUE)
library(foreign,  quietly = TRUE)
library(lattice,  quietly = TRUE)
library(lme4,  quietly = TRUE)
library(nlme,  quietly = TRUE)
library(survival,  quietly = TRUE)
library(dplyr,  quietly = TRUE)
library(ggfortify,  quietly = TRUE)
library(survminer,  quietly = TRUE)
library(rms,  quietly = TRUE)
library(MASS, quietly = TRUE)
```

Survival Models

Introduction

Survival models concerns the analysis of the time for an event to occur. The response variable is the time for the event to occur. The event is generally called "death."

Definitions

Survival models involve two functions:

- $S(t)$: the survival function. $S(t)$ is probability that death has not occurred until after time, t.
- $\lambda(t)$: the hazard function.

$$\lambda(t) = \frac{\text{probability of dying in at time, } t}{\text{probability of survival until time, } t} \approx \frac{\text{number of people who died at time, } t}{\text{number of people who lived until time, } t}$$

  $\lambda(t)$ measures the likelihood of death in a very small time interval, t and $t + dt$. It is a measure of *risk*.

- $\Lambda(t)$: the cumulative hazard. It is total hazard from 0 to time, t.

Note that these two functions are related:

- $\lambda(t) \leftrightarrow S(t)$

$$\lambda(t) = -\frac{d}{dt}\log S(t)$$

- $\lambda(t) \leftrightarrow \Lambda(t)$

$$\Lambda(t) = \int_0^t \lambda(t)\, dt$$

- $S(t) \leftrightarrow \Lambda(t)$ and $S(t) \leftrightarrow \lambda(t)$

$$S(t) = \exp\left(-\Lambda(t)\right) = \exp\left(-\int_0^t \lambda(t)\,dt\right).$$

Censoring

Like most models, survival models suceptible to imperfect data. Let's say a subject is recorded for a study up until a time, t^*. After time t^*, the subject may decide not to continue with study or it is not possible to locate the subject. Many things could have caused a lack of follow up. This subject is called *censored*. While it maybe reasonable to discard this data point, the censored data actually contains information that we know the event has not occurred prior to t^*. This gives more information to our model about time prior to t^* than if we were to discard the censored data.

Data

Description

We will be working the `colon` data set. This data comes from one of the first successful trials of a drug for colon cancer. The recurrence and death times are recorded for all patients in the study.

The `colon` dataset has the following columns:

- `id`: id
- `study`: 1 for all patients
- `rx`: Treatment - Obs(ervation), Lev(amisole), Lev(amisole)+5-FU. Levamisole is a low-toxicity compound previously used to treat worm infestations in animals; 5-FU is a moderately toxic (as these things go) chemotherapy agent.
- `sex`: 0 = female, 1 = male
- `age`: age of the patient
- `obstruct`: 0 = if tumour did not obstructed colon, 1 = if tumour obstructed colon
- `perfor`: perforation of colon
- `adhere`: adherence to nearby organs
- `nodes`: number of lymph nodes with detectable cancer
- `time`: days until event or censoring
- `status`: censoring status
- `differ`: differentiation of tumour (1=well, 2=moderate, 3=poor)
- `extent`: Extent of local spread (1=submucosa, 2=muscle, 3=serosa, 4=contiguous structures)
- `surg`: time from surgery to registration (0=short, 1=long)
- `node4`: more than 4 positive lymph nodes
- `etype`: event type: 1=recurrence,2=death

```r
attach(colon)
head(colon)
```

```
##   id study      rx sex age obstruct perfor adhere nodes status differ
## 1  1     1 Lev+5FU   1  43        0      0      0     5      1      2
## 2  1     1 Lev+5FU   1  43        0      0      0     5      1      2
## 3  2     1 Lev+5FU   1  63        0      0      0     1      0      2
## 4  2     1 Lev+5FU   1  63        0      0      0     1      0      2
## 5  3     1     Obs   0  71        0      0      1     7      1      2
## 6  3     1     Obs   0  71        0      0      1     7      1      2
##   extent surg node4 time etype
## 1      3    0     1 1521     2
## 2      3    0     1  968     1
## 3       3    0      0 3087      2
## 4       3    0      0 3087      1
## 5       2    0      1  963      2
## 6       2    0      1  542      1
```

Subsetting data and converting data

We will be studying the recurrence event of colon cancer.

```
colon_subset_recurrence = colon[colon$etype==1,]
```

Some survival models can only handle variables encoded in 0 and 1. We need to convert continuous variables, such as age and nodes, to 0 and 1.

```
colon_subset_recurrence$age.ds = sapply(colon_subset_recurrence$age,
                                        function(x) ifelse(x > 60, 1, 0))
```

If the binary variables are stored as `numeric` variables, the survival models will treat the explanatory variables as continuous variables rather than as discrete variables.

```
sapply(colon,class)
```

```
##        id     study        rx       sex       age  obstruct    perfor
## "numeric" "numeric"  "factor" "numeric" "numeric" "numeric" "numeric"
##    adhere     nodes    status    differ    extent      surg     node4
## "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"
##      time     etype
## "numeric" "numeric"
```

Many discrete variables are stored as `numeric` variables. We have to convert these columns to `factor`.

The `factor` takes as arguments:

- the dicrete data in the first argument
- `level` is current coding the discrete data. This is an optional argument.
- `label` is the encoding that you would like to change to discrete data. This is an optional argument. Use this argument if you would to change the labeling of the discrete data.

```
colon_subset_recurrence$age.ds <- factor(colon_subset_recurrence$age.ds,
                                         levels= c("0","1"),
                                         labels=c("<60",">60"))
```

```
colon_subset_recurrence$nodes.ds = sapply(colon_subset_recurrence$nodes,
                                          function(x) ifelse(x > 3, 1, 0))
colon_subset_recurrence$nodes.ds <- factor(colon_subset_recurrence$nodes.ds,
                                           levels= c("0","1"),
                                           labels=c("<3",">3"))
```

```
colon_subset_recurrence$sex <- factor(colon_subset_recurrence$sex,
                                      levels= c("0","1"), labels=c("F","M"))
```

```
colon_subset_recurrence$obstruct <- factor(colon_subset_recurrence$obstruct,
                                           levels= c("0","1"),
                                           labels=c("no obstruct","obstruct"))
colon_subset_recurrence$adhere <- factor(colon_subset_recurrence$adhere,
                                         levels= c("0","1"),
                                         labels=c("no adhere","adhere"))
colon_subset_recurrence$perfor <- factor(colon_subset_recurrence$perfor,
                                         levels= c("0","1"),
                                         labels=c("no perfor","perfor"))
```

```
colon_subset_recurrence$differ <- factor(colon_subset_recurrence$differ,
                                        levels= c("1","2","3"),
                                        labels=c("well","mod","poor"))
colon_subset_recurrence$extent <- factor(colon_subset_recurrence$extent,
                                        levels= c("1","2","3","4"),
                                        labels=c("submucosa", "muscle", "serosa", "contiguous"))
colon_subset_recurrence$surg <- factor(colon_subset_recurrence$surg,
                                      levels= c("0","1"),
                                      labels=c("short","long"))
```

Now, let's take a look at the data.

```
head(colon_subset_recurrence)
```

```
##    id study      rx sex age    obstruct    perfor    adhere nodes status
## 2   1     1 Lev+5FU   M  43 no obstruct no perfor no adhere     5      1
## 4   2     1 Lev+5FU   M  63 no obstruct no perfor no adhere     1      0
## 6   3     1     Obs   F  71 no obstruct no perfor    adhere     7      1
## 8   4     1 Lev+5FU   F  66    obstruct no perfor no adhere     6      1
## 10  5     1     Obs   M  69 no obstruct no perfor no adhere    22      1
## 12  6     1 Lev+5FU   F  57 no obstruct no perfor no adhere     9      1
##    differ extent  surg node4 time etype age.ds nodes.ds
## 2     mod serosa short     1  968     1    <60      >3
## 4     mod serosa short     0 3087     1    >60      <3
## 6     mod muscle short     1  542     1    >60      >3
## 8     mod serosa  long     1  245     1    >60      >3
## 10    mod serosa  long     1  523     1    >60      >3
## 12    mod serosa short     1  904     1    <60      >3
```

Surv Object

The `Surv` function takes as input the time and censoring status (0 or 1) of a data point. It returns a object that packages together time and censoring status.

```
surv <-with(colon_subset_recurrence, Surv(time,status))
head(surv)
```

```
## [1]  968  3087+  542   245   523   904
```

The + at the end of the time indicates that the data point was censored.

Kalpan-Meier Estimator

First, let t_i be the ith recorded time in the data. That is, t_1 is the 1st recorded time, t_2 is the 2nd recorded time, $\ldots$, t_{20} is the 20th recorded, etc.

Kalpan-Meier assumes that the survival function can be estimated as

$$\hat{S}(t) = \prod_{\text{for } i:\, t_i \leq t} \left(1 - \frac{d_i}{n_i}\right)$$

where d_i is the number of persons that "died" after time t_i and n_i is the number of uncensored persons that have lived up to t_i.

Kalpan-Meier Estimator for the entire data

To fit $\hat{S}(t) = \prod_{\text{for } i:\, t_i \leq t} 1 - \frac{d_i}{n_i}$ to the entire data, we use the command below.

```
km_fit <- survfit(surv~1, data=colon_subset_recurrence)
```

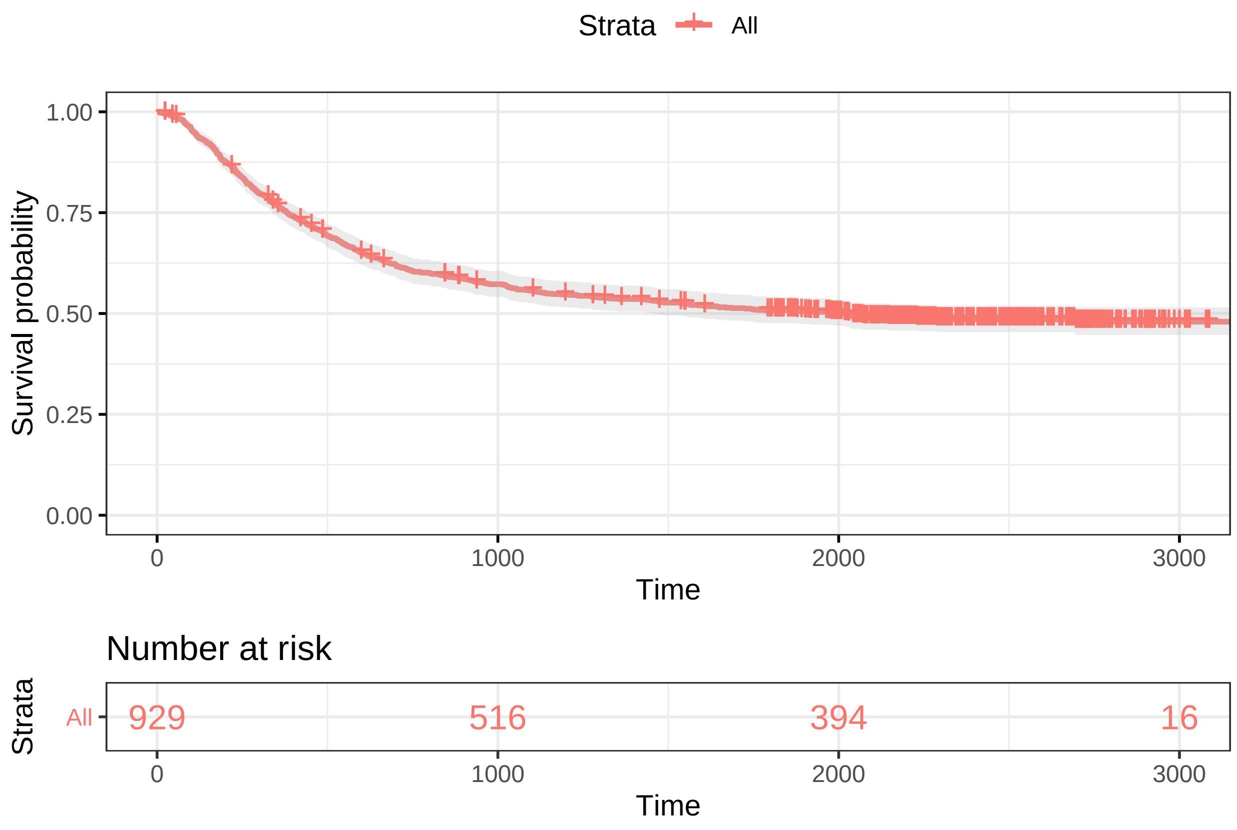
We can return a summary of the $\hat{S}(t)$ at certain time points. `summary(km_fit)` will return a summary `km_fit` for all time points in the data. Since the data set is large, I do not run the command, `summary(km_fit)`. However, you are free to do this on your own.

```
summary(km_fit,times=c(1,10,20,30,40,50))
```

```
## Call: survfit(formula = surv ~ 1, data = colon_subset_recurrence)
##
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1    929       0    1.000 0.00000        1.000        1.000
##    10    927       2    0.998 0.00152        0.995        1.000
##    20    926       2    0.996 0.00215        0.991        1.000
##    30    922       1    0.995 0.00240        0.990        0.999
##    40    919       4    0.990 0.00322        0.984        0.997
##    50    914       3    0.987 0.00371        0.980        0.994
```

There is a convience function `ggsurvplot` that generates a plot for a `survfit` object. `conf.int = TRUE` shows the confidence interval around the estimate. `risk.table = TRUE` shows a tabulation of risk below $\hat{S}(t)$.

```
ggsurvplot(km_fit, data = colon_subset_recurrence,
           conf.int = TRUE,risk.table = TRUE,
           ggtheme = theme_bw(),
           risk.table.col = "strata")
```



Kalpan-Meier Estimator for the data divided into obstruct and no obstruct

`colon_subset_recurrence` can be divided two data sets by the `obstruct` column. Those patients whose colons are obstructed by the tumour and those whose colons aren't. We can fit to each data partition to a Kalpan-Meier Estimator

$$\hat{S}_{\text{obstruct}}(t) = \prod_{\substack{\text{for } i:\, t_i \le t \\ \text{obstruct}_i = \text{obstruct}}} \left(1 - \frac{d_i}{n_i}\right)$$

$$\hat{S}_{\text{no obstruct}}(t) = \prod_{\substack{\text{for } i:\, t_i \le t \\ \text{obstruct}_i = \text{no obstruct}}} \left(1 - \frac{d_i}{n_i}\right)$$

to the entire data. To do this, we use the command below.

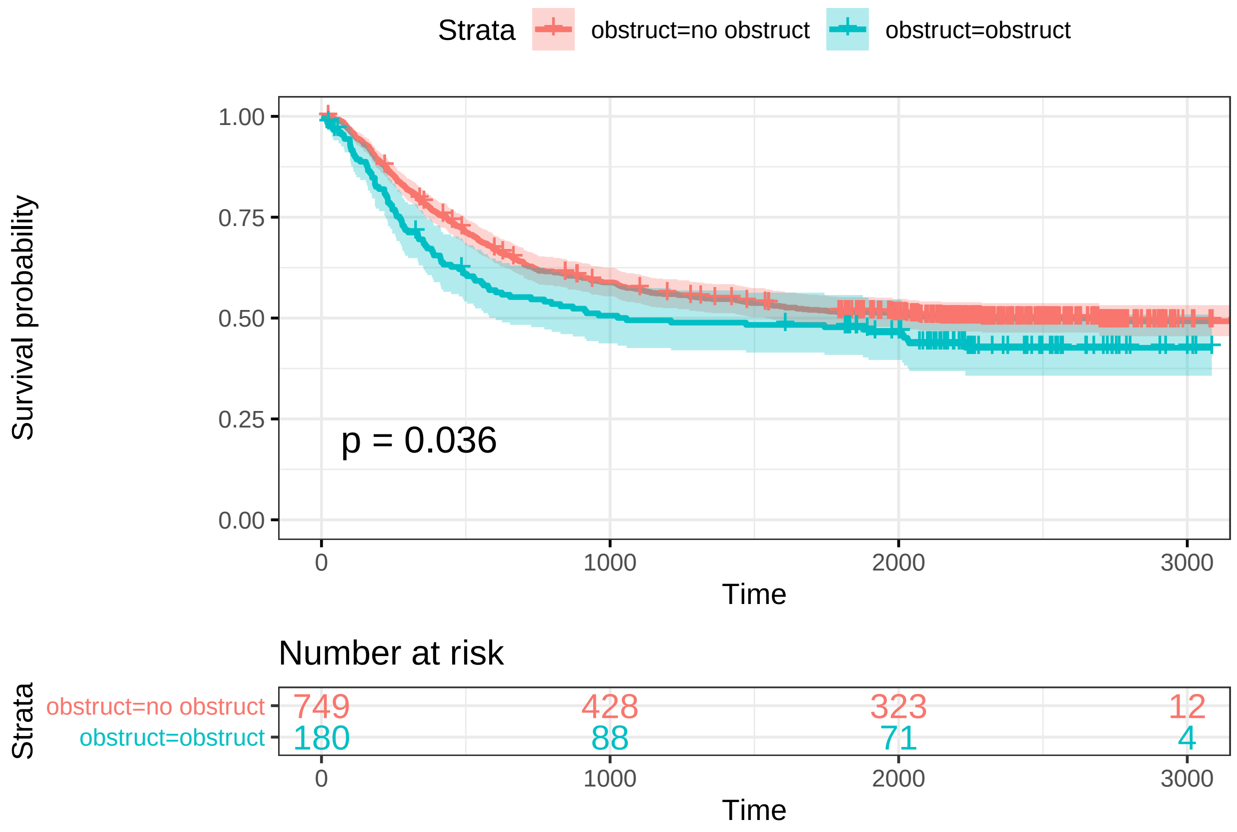
```
km_fit <- survfit(surv~obstruct, data=colon_subset_recurrence)
```

```
summary(km_fit,times=c(1,10,20,30,40,50))
```

```
## Call: survfit(formula = surv ~ obstruct, data = colon_subset_recurrence)
##
##                 obstruct=no obstruct
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1    749       0    1.000 0.00000        1.000        1.000
##    10    748       1    0.999 0.00133        0.996        1.000
##    20    748       0    0.999 0.00133        0.996        1.000
##    30    746       1    0.997 0.00189        0.994        1.000
##    40    745       1    0.996 0.00231        0.991        1.000
##    50    742       3    0.992 0.00326        0.986        0.998
```

```
##
##                 obstruct=obstruct
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1    180       0    1.000 0.00000        1.000        1.000
##    10    179       1    0.994 0.00554        0.984        1.000
##    20    178       2    0.983 0.00954        0.965        1.000
##    30    176       0    0.983 0.00954        0.965        1.000
##    40    174       3    0.967 0.01342        0.941        0.993
##    50    172       0    0.967 0.01342        0.941        0.993
```

```r
ggsurvplot(km_fit, data = colon_subset_recurrence,
           pval = TRUE,conf.int = TRUE,
           risk.table = TRUE, ggtheme = theme_bw(),
           risk.table.col = "strata")
```



Kalpan-Meier Estimator for the data divided into adhere and no adhere

`colon_subset_recurrence` can be divided two data sets by the `adhere` column. Those patients whose colons are obstructed by the tumour and those whose colons aren't. We can fit to each data partition to a Kalpan-Meier Estimator

$$\hat{S}_{\text{adhere}}(t) = \prod_{\substack{\text{for } i:\, t_i \leq t \\ \text{adher}_i = \text{adhere}}} \left(1 - \frac{d_i}{n_i}\right)$$

$$\hat{S}_{\text{no adhere}}(t) = \prod_{\substack{\text{for } i:\, t_i \leq t \\ \text{adher}_i = \text{no adhere}}} \left(1 - \frac{d_i}{n_i}\right)$$

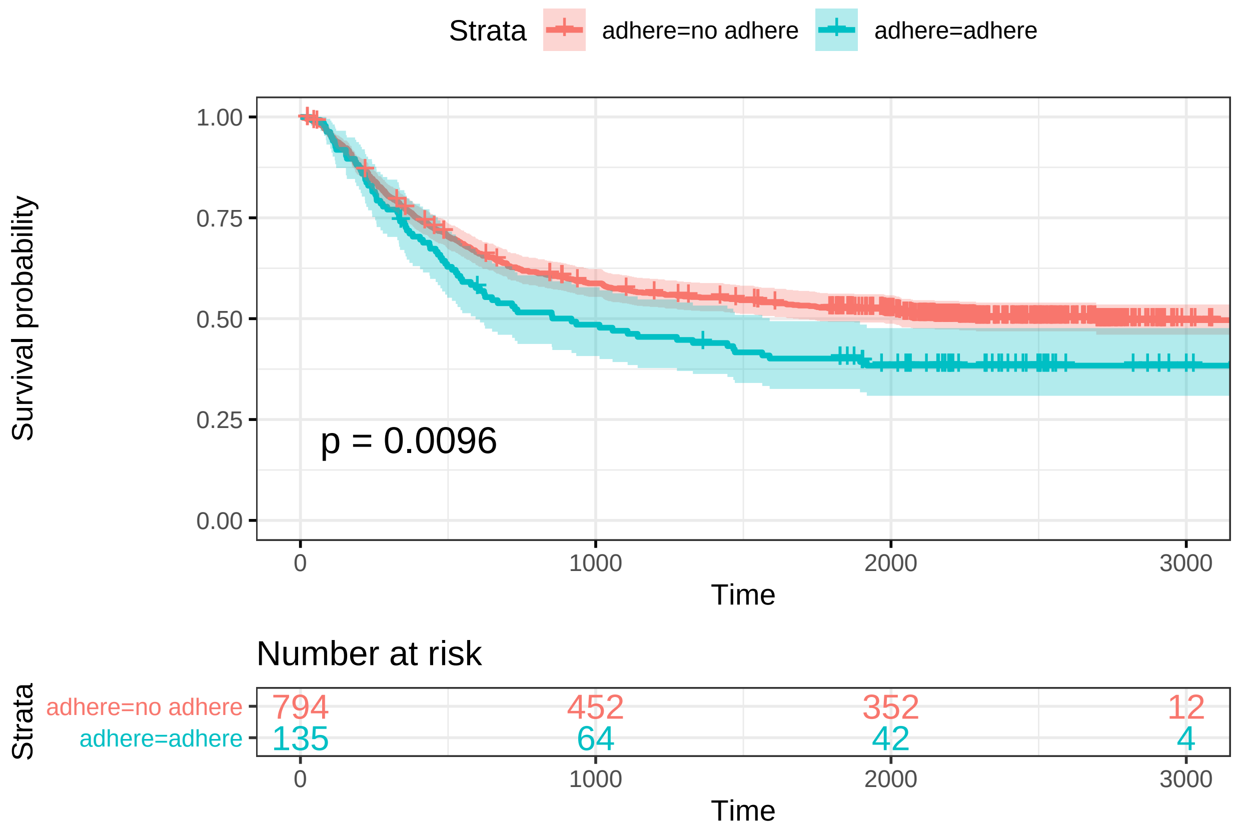
to the entire data. To do this, we use the command below.

```r
km_fit <- survfit(surv~adhere, data=colon_subset_recurrence)
```

```r
summary(km_fit,times=c(1,10,20,30,40,50))
```

```
## Call: survfit(formula = surv ~ adhere, data = colon_subset_recurrence)
##
##                 adhere=no adhere
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1    794       0    1.000 0.00000        1.000        1.000
##    10    792       2    0.997 0.00178        0.994        1.000
##    20    791       2    0.995 0.00251        0.990        1.000
##    30    787       1    0.994 0.00281        0.988        0.999
##    40    785       3    0.990 0.00355        0.983        0.997
##    50    780       3    0.986 0.00416        0.978        0.994
##
##                 adhere=adhere
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1    135       0    1.000 0.00000        1.000            1
##    10    135       0    1.000 0.00000        1.000            1
##    20    135       0    1.000 0.00000        1.000            1
##    30    135       0    1.000 0.00000        1.000            1
##    40    134       1    0.993 0.00738        0.978            1
##    50    134       0    0.993 0.00738        0.978            1
```

```r
ggsurvplot(km_fit, data = colon_subset_recurrence,
           pval = TRUE,conf.int = TRUE,
           risk.table = TRUE, ggtheme = theme_bw(),
           risk.table.col = "strata")
```



Kalpan-Meier Estimator for the data divided into (adhere, obstruct), (adhere, no obstruct), (no adhere, obstruct) and (no adhere, no obstruct)

`colon_subset_recurrence` can be divided in any amount by the explanatory variables Let's consider breaking up the data based on a patient's obstruction and adherence status. We can fit to each data partition to a Kalpan-Meier Estimator

$$\hat{S}_{\text{adhere,obstruct}}(t) = \prod_{\substack{\text{for } i:\, t_i \le t \\ \text{adher}_i=\text{adhere} \\ \text{obstruct}_i=\text{ obstruct}}} \left(1 - \frac{d_i}{n_i}\right)$$

$$\hat{S}_{\text{no adhere,obstruct}}(t) = \prod_{\substack{\text{for } i:\, t_i \le t \\ \text{adher}_i=\text{no adhere} \\ \text{obstruct}_i=\text{ obstruct}}} \left(1 - \frac{d_i}{n_i}\right)$$

$$\hat{S}_{\text{adhere,no obstruct}}(t) = \prod_{\substack{\text{for } i:\, t_i \le t \\ \text{adher}_i=\text{adhere} \\ \text{obstruct}_i=\text{no obstruct}}} \left(1 - \frac{d_i}{n_i}\right)$$

$$\hat{S}_{\text{no adhere,no obstruct}}(t) = \prod_{\substack{\text{for } i:\, t_i \le t \\ \text{adher}_i=\text{no adhere} \\ \text{obstruct}_i=\text{no obstruct}}} \left(1 - \frac{d_i}{n_i}\right)$$

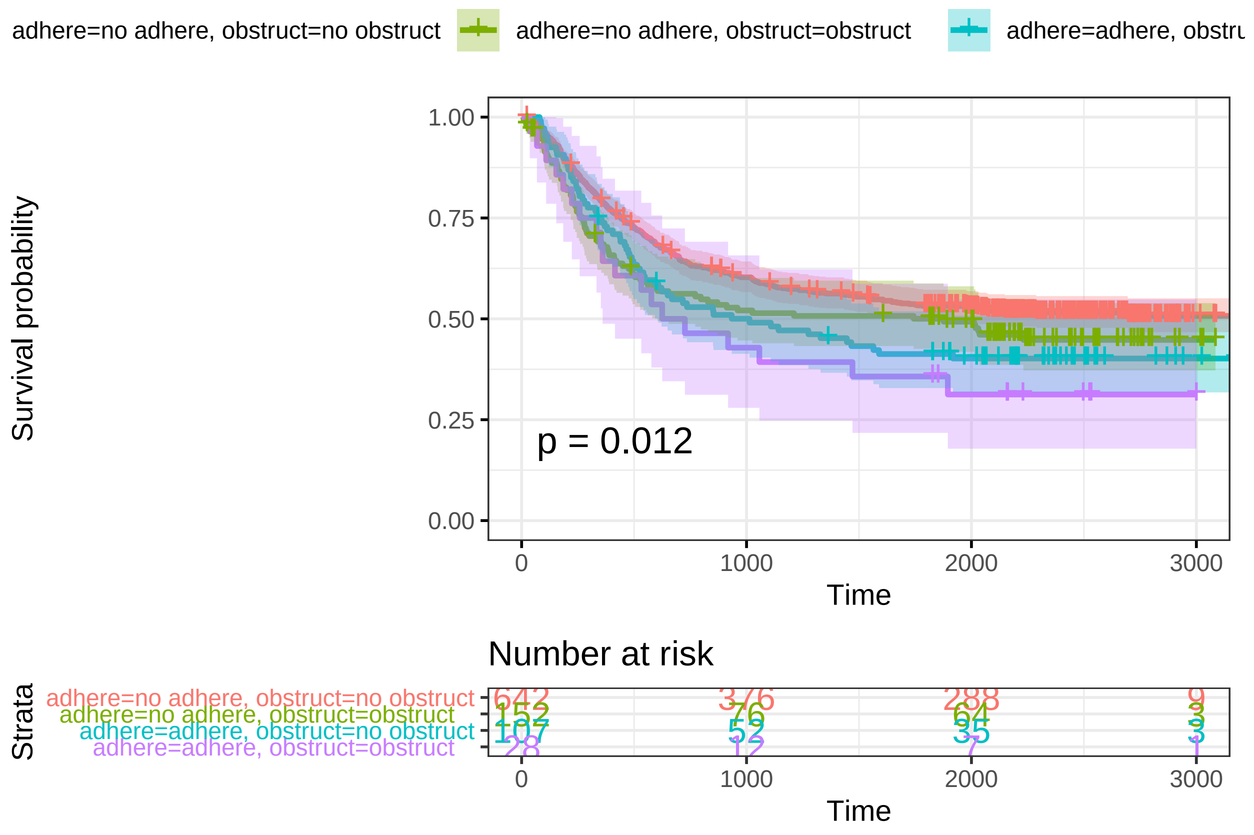
to the entire data. To do this, we use the command below.

```r
km_fit <- survfit(surv~adhere + obstruct, data=colon_subset_recurrence)
```

```r
summary(km_fit,times=c(1,10,20,30,40,50))
```

```
## Call: survfit(formula = surv ~ adhere + obstruct, data = colon_subset_recurrence)
##
##                 adhere=no adhere, obstruct=no obstruct
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1    642       0    1.000 0.00000        1.000        1.000
##    10    641       1    0.998 0.00156        0.995        1.000
##    20    641       0    0.998 0.00156        0.995        1.000
##    30    639       1    0.997 0.00220        0.993        1.000
##    40    638       1    0.995 0.00269        0.990        1.000
##    50    635       3    0.991 0.00380        0.983        0.998
##
##                 adhere=no adhere, obstruct=obstruct
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1    152       0    1.000 0.00000        1.000        1.000
##    10    151       1    0.993 0.00656        0.981        1.000
##    20    150       2    0.980 0.01128        0.958        1.000
##    30    148       0    0.980 0.01128        0.958        1.000
##    40    147       2    0.967 0.01451        0.939        0.996
##    50    145       0    0.967 0.01451        0.939        0.996
##
##                 adhere=adhere, obstruct=no obstruct
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1    107       0        1       0            1            1
##    10    107       0        1       0            1            1
##    20    107       0        1       0            1            1
##    30    107       0        1       0            1            1
##     40     107        0         1        0            1             1
##     50     107        0         1        0            1             1
##
##                    adhere=adhere, obstruct=obstruct
##   time n.risk n.event survival std.err lower 95% CI upper 95% CI
##      1      28       0    1.000  0.0000        1.000             1
##     10      28       0    1.000  0.0000        1.000             1
##     20      28       0    1.000  0.0000        1.000             1
##     30      28       0    1.000  0.0000        1.000             1
##     40      27       1    0.964  0.0351        0.898             1
##     50      27       0    0.964  0.0351        0.898             1
```

```r
ggsurvplot(km_fit, data = colon_subset_recurrence,
           pval = TRUE,conf.int = TRUE,
           risk.table = TRUE, ggtheme = theme_bw(),
           risk.table.col = "strata")
```



Cox Proportional Hazard

In the limit of large data, the Kaplan-Meier estimator converges to true survival function. However, the Kaplan-Meier has two disadvantages:

- it cannot effectively accomodate continuous data
- it is non-parameteric – this means that given a data point, we cannot predict their life trajectory from data. This will be seen more clearly later in this section.

Rather than estimating survival function at each time interval, the *Cox Proportional Hazard* assumes that hazard function is an exponentiated linear function of explanatory variables. That is,

$$\lambda(t) = \lambda_0(t)\exp\left(\beta_1 X_1 + \cdots + \beta_n X_n\right).$$

$\lambda_0(t)$ is called the baseline function. $\lambda(t) = \lambda_0(t)$ when $X_1 = X_2 = \cdots = X_n = 0$.

The Cox Proportional Hazard models the effects of the covariates on the baseline function. The baseline function is generally unknown. However, the effects of the covariates can still be determined regardless of the baseline function. The β_i's is calculated using *partial maximum likelihood.* Avoiding the estimation of $\lambda_0(t)$ prevents accumulation of errors in a unknown function.

Note that the Cox Proportional Hazard does not solve all the problems of the Kaplan-Meier estimator. Cox Proportional Hazard has one (or 1/2) disadvantage:

- it is semi-parametric. Given a data point, we can estimate the effect of a covariate on the baseline function. However, we cannot predict the life trajectory of data point unless we know $\lambda_0(t)$.

Cox Proportional Hazard for $X_1 =$ obstruct

Given only one covariate $X_1 =$ obstruct, our Cox Proportional Hazard function takes the form

$$\lambda(t) = \lambda_0(t)\exp\left(\beta_1 X_1\right).$$

where

$$X_1 = \begin{cases} 0 & \text{if } X_1 = \text{no obstruct} \\ 1 & \text{if } X_1 = \text{obstruct} \end{cases}.$$

We fit the Cox Proportional Hazard model accordingly.

```
cox <- coxph(surv ~  obstruct,
             data=colon_subset_recurrence)
```

```
summary(cox)
```

```
## Call:
## coxph(formula = surv ~ obstruct, data = colon_subset_recurrence)
##
##   n= 929, number of events= 468
##
##                     coef exp(coef) se(coef)     z Pr(>|z|)
## obstructobstruct 0.2370    1.2675   0.1132 2.094   0.0363 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                  exp(coef) exp(-coef) lower .95 upper .95
## obstructobstruct     1.267      0.789     1.015     1.582
##
```

```
## Concordance= 0.523  (se = 0.01 )
## Likelihood ratio test= 4.18  on 1 df,   p=0.04
## Wald test            = 4.38  on 1 df,   p=0.04
## Score (logrank) test = 4.4  on 1 df,   p=0.04
```

```
coef(cox)
```

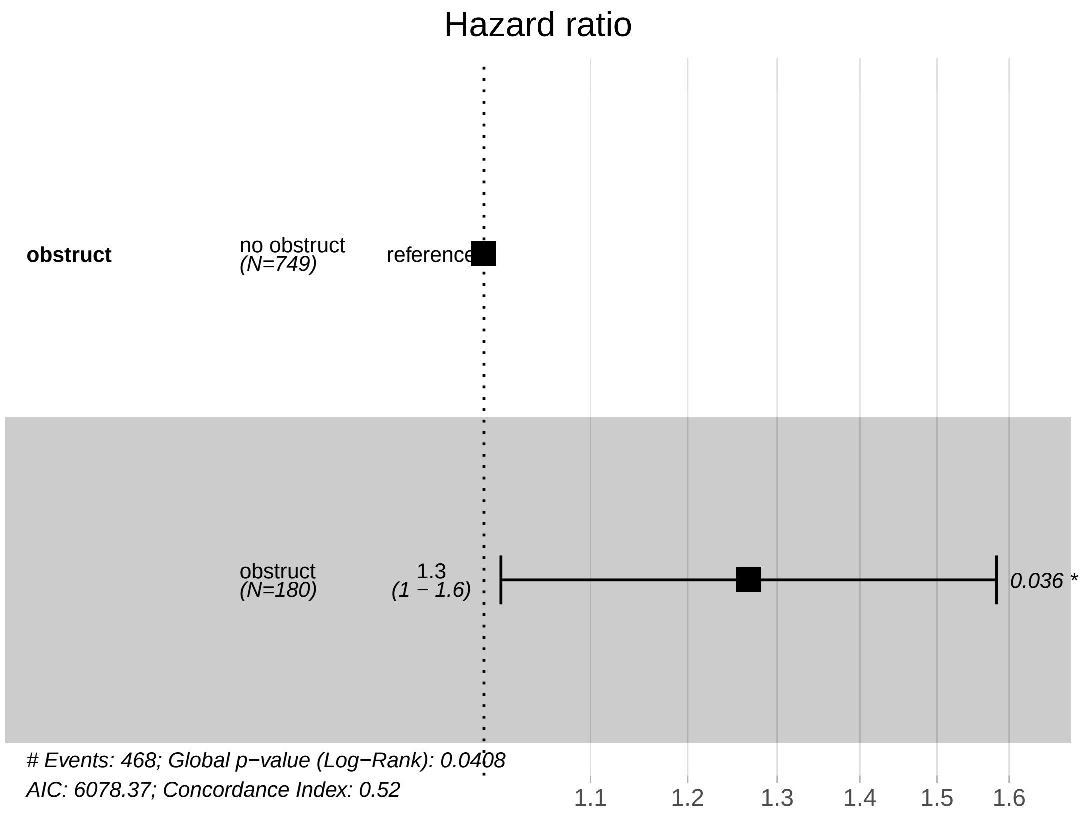
```
## obstructobstruct
##        0.2370211
```

```
test.ph <- cox.zph(cox)
test.ph
```

```
##                     rho chisq      p
## obstructobstruct -0.101  4.76 0.0291
```

```
ggforest(cox, data = colon_subset_recurrence)
```

```
## Warning: Removed 1 rows containing missing values (geom_errorbar).
```



Cox Proportional Hazard for $X_1 =$ obstruct, $X_2 =$ adher

Given only two covariate $X_1 =$ obstruct, our Cox Proportional Hazard function takes the form

$$\lambda(t) = \lambda_0(t)\exp\left(\beta_1 X_1 + \beta_2 X_2\right).$$

where

$$X_1 = \begin{cases} 0 & \text{if } X_1 = \text{no obstruct} \\ 1 & \text{if } X_1 = \text{obstruct} \end{cases}, X_2 = \begin{cases} 0 & \text{if } X_2 = \text{no adhere} \\ 1 & \text{if } X_2 = \text{adhere} \end{cases}.$$

We fit the Cox Proportional Hazard model accordingly.

```r
cox <- coxph(surv ~  obstruct + adhere,
             data=colon_subset_recurrence)
```

```r
summary(cox)
```

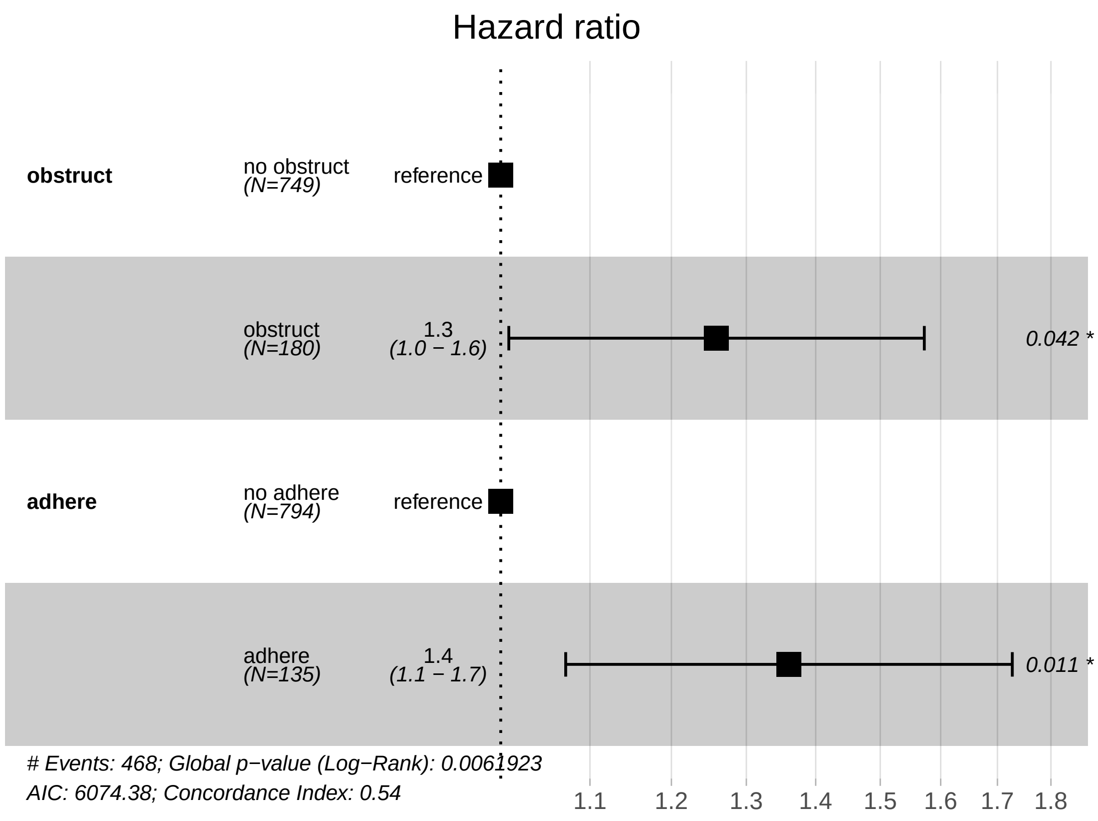
```
## Call:
## coxph(formula = surv ~ obstruct + adhere, data = colon_subset_recurrence)
##
##   n= 929, number of events= 468
##
##                      coef exp(coef) se(coef)     z Pr(>|z|)
## obstructobstruct 0.2306    1.2593   0.1132 2.036   0.0417 *
## adhereadhere     0.3080    1.3606   0.1217 2.530   0.0114 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                  exp(coef) exp(-coef) lower .95 upper .95
## obstructobstruct     1.259     0.7941     1.009     1.572
## adhereadhere         1.361     0.7349     1.072     1.727
##
## Concordance= 0.536  (se = 0.011 )
## Likelihood ratio test= 10.17  on 2 df,   p=0.006
## Wald test            = 10.81  on 2 df,   p=0.004
## Score (logrank) test = 10.88  on 2 df,   p=0.004
```

```r
coef(cox)
```

```
## obstructobstruct     adhereadhere
##        0.2305705        0.3079530
```

```r
ggforest(cox, data = colon_subset_recurrence)
```

```
## Warning: Removed 2 rows containing missing values (geom_errorbar).
```



```r
test.ph <- cox.zph(cox)
test.ph
```

```
##                      rho chisq      p
## obstructobstruct -0.1020 4.853 0.0276
## adhereadhere      0.0449 0.943 0.3316
## GLOBAL                NA 5.693 0.0581
```

Cox Proportional Hazard for $X_1 =$ obstruct, $X_2 =$ adher, $X_3 =$ nodes

Given only three covariate $X_1 =$ obstruct, our Cox Proportional Hazard function takes the form

$$\lambda(t) = \lambda_0(t)\exp\left(\beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3\right).$$

where

$$X_1 = \begin{cases} 0 & \text{if } X_1 = \text{no obstruct} \\ 1 & \text{if } X_1 = \text{obstruct} \end{cases}, X_2 = \begin{cases} 0 & \text{if } X_2 = \text{no adhere} \\ 1 & \text{if } X_2 = \text{adhere} \end{cases}$$

and X_3 is any positive integer.

We fit the Cox Proportional Hazard model accordingly.

```r
cox <- coxph(surv ~ obstruct + adhere + nodes,
             data=colon_subset_recurrence)
```

```r
summary(cox)
```

```
## Call:
## coxph(formula = surv ~ obstruct + adhere + nodes, data = colon_subset_recurrence)
##
```

```
##   n= 911, number of events= 456
##    (18 observations deleted due to missingness)
##
##                       coef exp(coef) se(coef)     z Pr(>|z|)
## obstructobstruct 0.260612  1.297724 0.115168 2.263   0.0236 *
## adhereadhere     0.305644  1.357499 0.123190 2.481   0.0131 *
## nodes            0.085076  1.088800 0.008847 9.617   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                  exp(coef) exp(-coef) lower .95 upper .95
## obstructobstruct     1.298     0.7706     1.036     1.626
## adhereadhere         1.357     0.7366     1.066     1.728
## nodes                1.089     0.9184     1.070     1.108
##
## Concordance= 0.638  (se = 0.013 )
## Likelihood ratio test= 74.27  on 3 df,   p=5e-16
## Wald test            = 101.4  on 3 df,   p=<2e-16
## Score (logrank) test = 103.6  on 3 df,   p=<2e-16
```

```
coef(cox)
```

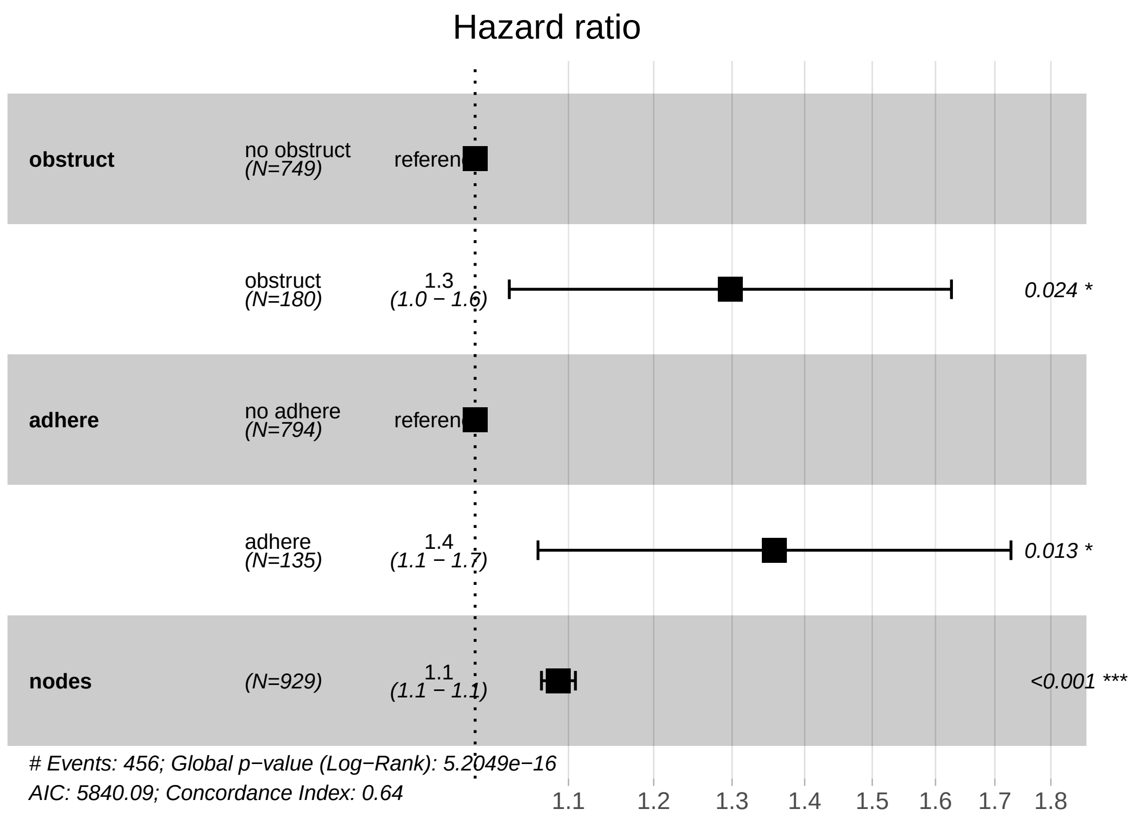
```
## obstructobstruct     adhereadhere            nodes
##       0.26061166       0.30564393       0.08507602
```

```
test.ph <- cox.zph(cox)
test.ph
```

```
##                       rho chisq      p
## obstructobstruct -0.1001 4.536 0.0332
## adhereadhere      0.0333 0.503 0.4781
## nodes            -0.0595 0.994 0.3186
## GLOBAL                 NA 5.722 0.1259
```

```
ggforest(cox, data = colon_subset_recurrence)
```

```
## Warning: Removed 2 rows containing missing values (geom_errorbar).
```



Cox Proportional Hazard for $X_1 =$ obstruct, $X_2 =$ adher, $X_3 =$ nodes.ds

Given only three covariate $X_1 =$ obstruct, our Cox Proportional Hazard function takes the form

$$\lambda(t) = \lambda_0(t)\exp\left(\beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3\right).$$

where

$$X_1 = \begin{cases} 0 & \text{if } X_1 = \text{no obstruct} \\ 1 & \text{if } X_1 = \text{obstruct} \end{cases}, \; X_2 = \begin{cases} 0 & \text{if } X_2 = \text{no adhere} \\ 1 & \text{if } X_2 = \text{adhere} \end{cases}, \; X_3 = \begin{cases} 0 & \text{if } X_3 = >3 \\ 1 & \text{if } X_3 = <3 \end{cases}.$$

We fit the Cox Proportional Hazard model accordingly.

```
cox <- coxph(surv ~ obstruct + adhere + nodes.ds,
             data=colon_subset_recurrence)
```

```
summary(cox)
```

```
## Call:
## coxph(formula = surv ~ obstruct + adhere + nodes.ds, data = colon_subset_recurrence)
##
##   n= 911, number of events= 456
##    (18 observations deleted due to missingness)
##
##                      coef exp(coef) se(coef)     z Pr(>|z|)
## obstructobstruct 0.23954   1.27066  0.11495 2.084   0.0372 *
## adhereadhere     0.30972   1.36304  0.12324 2.513   0.0120 *
## nodes.ds>3       0.82056   2.27178  0.09434 8.698   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                   exp(coef) exp(-coef) lower .95 upper .95
## obstructobstruct      1.271     0.7870     1.014     1.592
## adhereadhere          1.363     0.7337     1.071     1.735
## nodes.ds>3            2.272     0.4402     1.888     2.733
##
## Concordance= 0.625  (se = 0.013 )
## Likelihood ratio test= 81.98  on 3 df,   p=<2e-16
## Wald test            = 86.02  on 3 df,   p=<2e-16
## Score (logrank) test = 90.28  on 3 df,   p=<2e-16
```

```
coef(cox)
```

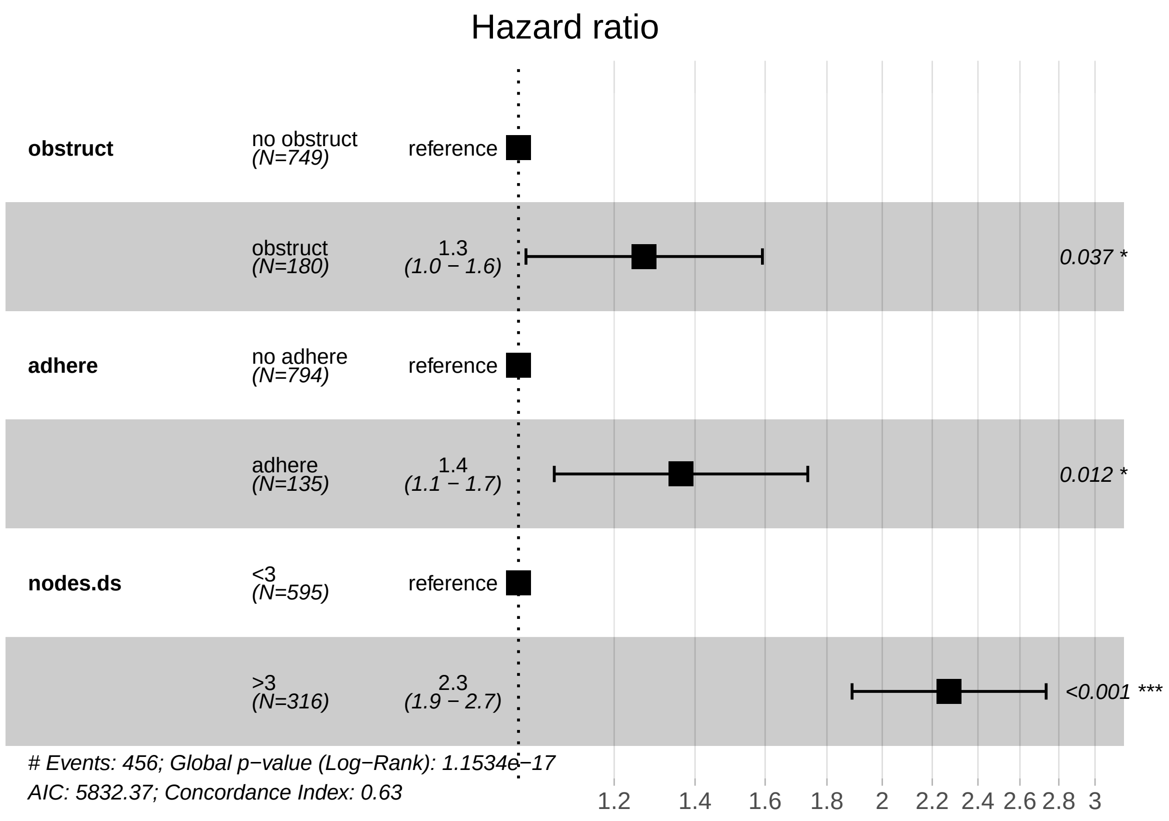
```
## obstructobstruct     adhereadhere       nodes.ds>3
##        0.2395393        0.3097155        0.8205624
```

```
test.ph <- cox.zph(cox)
test.ph
```

```
##                      rho  chisq        p
## obstructobstruct -0.1032  4.831 2.80e-02
## adhereadhere      0.0467  0.994 3.19e-01
## nodes.ds>3       -0.1917 16.141 5.88e-05
## GLOBAL                NA 21.509 8.25e-05
```

```
ggforest(cox, data = colon_subset_recurrence)
```

```
## Warning: Removed 3 rows containing missing values (geom_errorbar).
```



Estimating Survival Curve

It is possible to estimate the survival curve for the Cox Proportional Model as long as we have some estimate for $\lambda_0(t)$. One way to estimate $\lambda_0(t)$ from data is to use formula:

$$\lambda_0(t_i) \approx \frac{d_i}{\sum_{s \in R_i} \exp\left(\beta_1 X_{1s} + \cdots + \beta_n X_{ns}\right)}$$

where d_i is the number of deaths in at time t_i, R_i is set of persons alive after t_i and X_{ij} is the ith explanatory variable of the jth person.

Now let's create some data point. This data point will have the `obstruct` set to `no obstruct`, `adhere` set to `no adhere`, `nodes.ds` set to `<3` and `extent` set to `serosa`.

```r
subject_one <- data.frame(obstruct = factor('no obstruct'),
                          adhere = factor('adhere'), nodes.ds = factor('<3'))
```

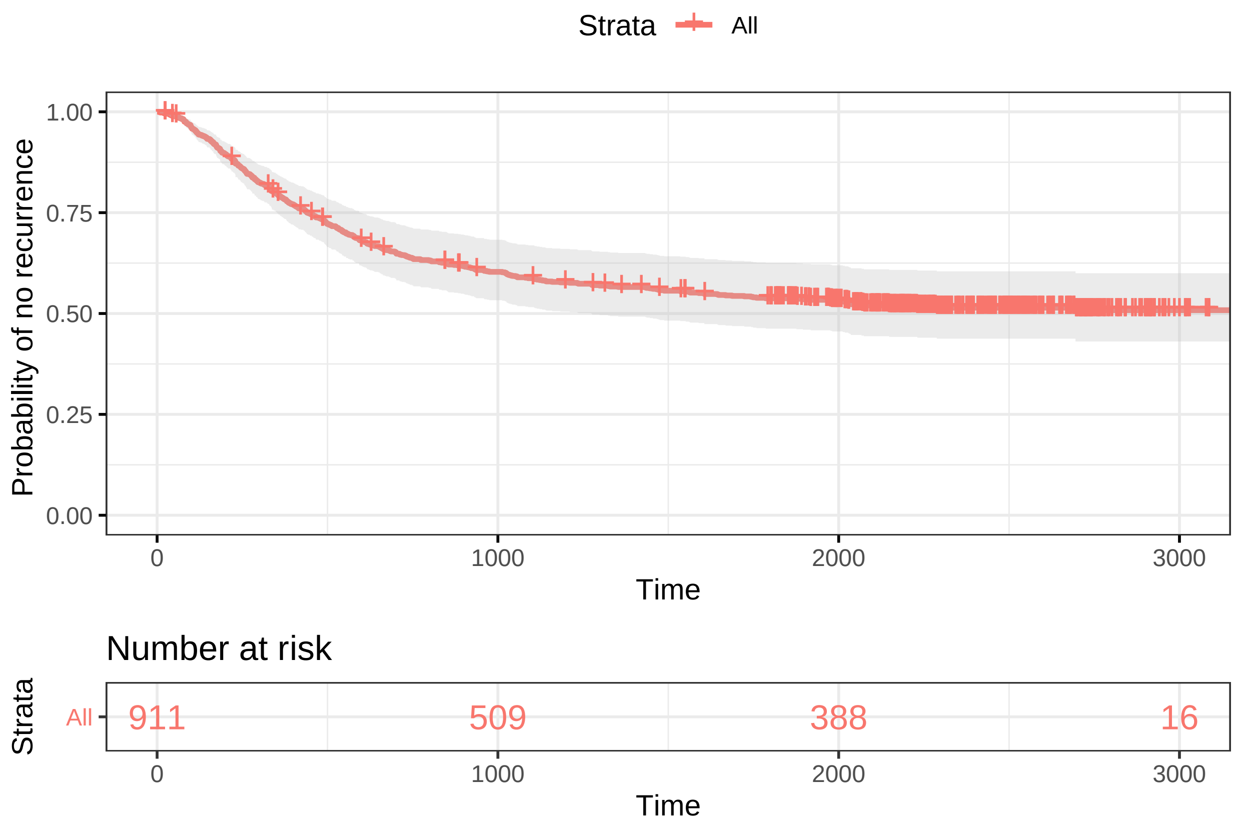
Using the `survfit` function, we can generate an object which will be used for plotting. `survfit` takes as argument:

- first argment: cox proportional hazard model fit with `coxph`
- second argment: the data point in question. It must have the same explanatory variables as the model in the first argument
- `data`: the data set used to fit the `coxph` object.

```r
prediction_one <- survfit(cox, subject_one,
                          data = colon_subset_recurrence)
```

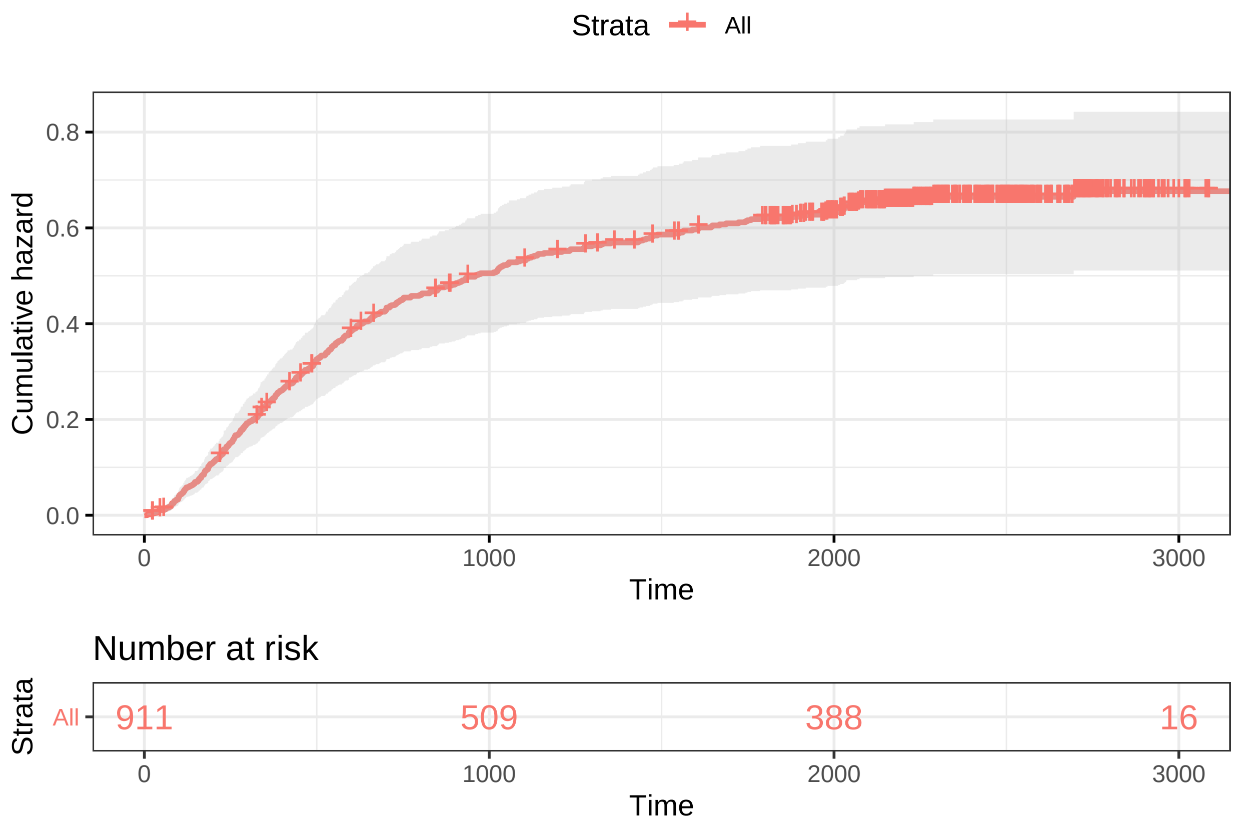
We then use the `ggsurvplot` function to plot the estimate of the survival curve from `survfit` fit object.

```r
ggsurvplot(prediction_one, ylab = "Probability of no recurrence ",
           conf.int = TRUE,risk.table = TRUE,
           ggtheme = theme_bw(),
           risk.table.col = "strata")
```



We can also use the `ggsurvplot` function to plot the estimate of the cumulative hazard curve from `survfit` fit object

```r
ggsurvplot(prediction_one, fun="cumhaz",
           conf.int = TRUE,risk.table = TRUE,
           ggtheme = theme_bw(),
           risk.table.col = "strata")
```


Strata
All
Cumulative hazard
0.0
0.2
0.4
0.6
0.8
0
1000
2000
3000
Time
Number at risk
Strata
All
911
509
388
16
0
1000
2000
3000
Time

Accelerated failure time models

Accelerated failure time model assume the functional form of $\lambda_0(t)$. While this makes the model fully parametric, this introduces errors in our model if $\lambda_0(t)$ is of the wrong form.

We use the function, `survreg`, to fit accelerated failure time models. The argument, `dist`, specifies the distribution which implies the form of $\lambda_0(t)$. We will be considering:

- exponential models, `dist="exponential"`
- weibull models, `dist="weibull"`

Exponential models

This assumes that $\lambda_0(t)$ is a constant, λ.

$$\lambda_0(t) = \lambda$$

Therefore, the hazard function is now

$$\lambda_i(t) = \lambda \exp\left(\beta_1 X_{1i} + \cdots + \beta_n X_{ni}\right).$$

`survreg` learns the parameter value, λ, and the regression coefficients.

As an example, we will be consider the model: `surv ~ 1 + obstruct + adhere + nodes.ds + age` for all the accelerated time models.

```
survregExp <- survreg(surv ~ 1 + obstruct + adhere + nodes.ds,
                           dist="exponential",data=colon_subset_recurrence)
summary(survregExp)
```

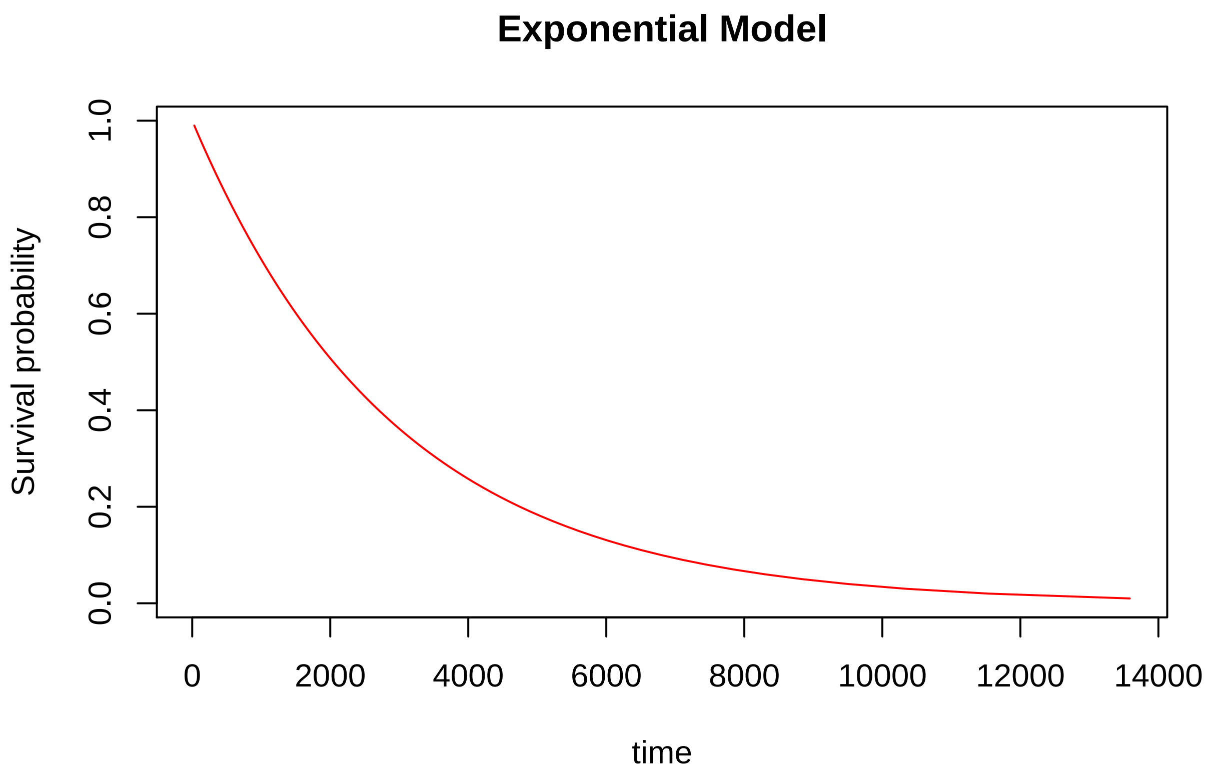
```
##
## Call:
## survreg(formula = surv ~ 1 + obstruct + adhere + nodes.ds, data = colon_subset_recurrence,
##     dist = "exponential")
##                    Value Std. Error      z      p
## (Intercept)       8.3723     0.0715 117.10 <2e-16
## obstructobstruct -0.2416     0.1149  -2.10 0.0355
## adhereadhere     -0.3827     0.1231  -3.11 0.0019
## nodes.ds>3       -0.9216     0.0940  -9.81 <2e-16
##
## Scale fixed at 1
##
## Exponential distribution
## Loglik(model)= -4026.4   Loglik(intercept only)= -4078.5
##  Chisq= 104.16 on 3 degrees of freedom, p= 2e-22
## Number of Newton-Raphson Iterations: 5
## n=911 (18 observations deleted due to missingness)
```

Therefore, $\lambda = \exp(8.08361)$.

Plot of data point

```
subject_two = list(obstruct = factor('no obstruct'), adhere = factor('adhere'), nodes.ds = factor('<3'))
```

```r
plot(predict(survregExp, newdata=subject_two,
             type="quantile",p=seq(.01,.99,by=.01)),
     seq(.99,.01,by=-.01), col="red",type='l',xlab='time',
     ylab='Survival probability',main='Exponential Model')
```



```r
detach(colon)
```

Weibull models

This assumes that $\lambda_0(t) = \lambda\gamma t^{\gamma-1}$.

Therefore, the hazard function is now

$$\lambda_i(t) = \lambda\gamma t^{\gamma-1}\exp\left(\beta_1 X_{1i} + \cdots + \beta_n X_{ni}\right).$$

`survreg` learns the parameter value, λ and γ,and the regression coefficients.

As an example, we will be consider the model: `surv ~ 1 + obstruct + adhere + nodes.ds + age` for all the accelerated time models.

```r
survregWeibull = survreg(surv ~ 1 + obstruct + adhere + nodes.ds,
                 dist="weibull",data=colon_subset_recurrence)
summary(survregWeibull)
```

```
##
## Call:
## survreg(formula = surv ~ 1 + obstruct + adhere + nodes.ds, data = colon_subset_recurrence,
##     dist = "weibull")
##                      Value Std. Error     z      p
## (Intercept)         8.7024     0.1120 77.68 <2e-16
## obstructobstruct -0.3219     0.1632 -1.97 0.0486
## adhereadhere     -0.4852     0.1751 -2.77 0.0056
## nodes.ds>3       -1.1957     0.1379 -8.67 <2e-16
## Log(scale)        0.3487     0.0411  8.48 <2e-16
##
## Scale= 1.42
##
## Weibull distribution
## Loglik(model)= -3984.8   Loglik(intercept only)= -4028.2
##  Chisq= 86.89 on 3 degrees of freedom, p= 1e-18
## Number of Newton-Raphson Iterations: 5
## n=911 (18 observations deleted due to missingness)
```

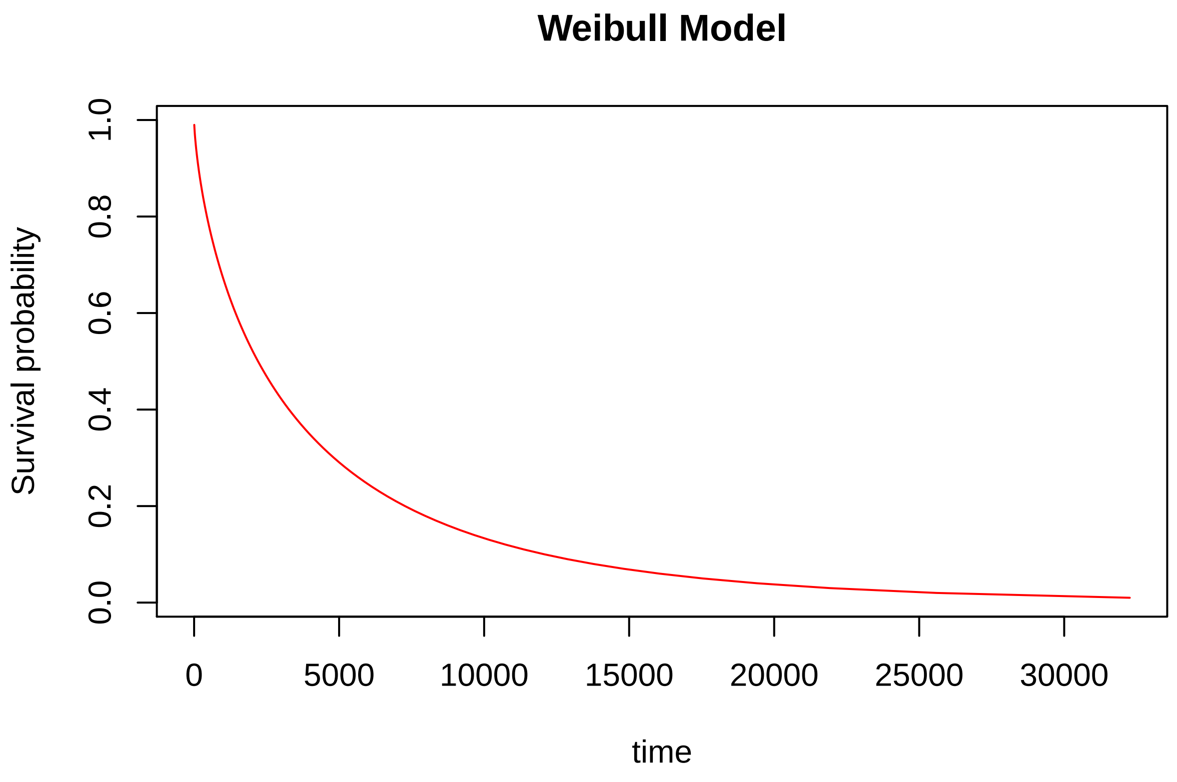
Therefore,

$$\gamma = \exp(-0.3487)$$

$$\lambda = \exp(-8.7024 \times \gamma)$$

Plot of data point

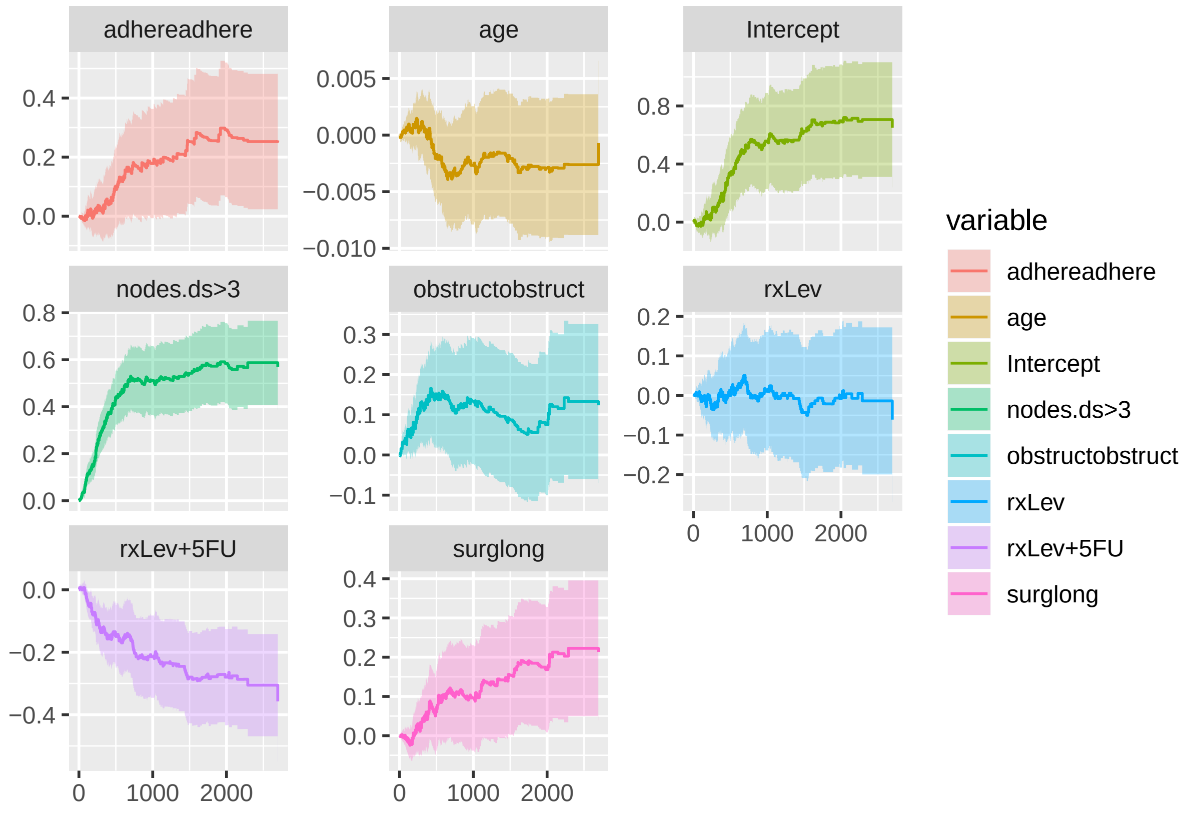
```r
subject_two = list(obstruct = factor('no obstruct'),
                   adhere = factor('adhere'),
                   nodes.ds = factor('<3'))

plot(predict(survregWeibull, newdata=subject_two,
             type="quantile",p=seq(.01,.99,by=.01)),
     seq(.99,.01,by=-.01), col="red",type='l',xlab='time',
     ylab='Survival probability',main='Weibull Model')
```


Weibull Model
Survival probability
1.0
0.8
0.6
0.4
0.2
0.0
0
5000
10000
15000
20000
25000
30000
time

Aalen's additive regression model

```
aa_fit <- aareg(surv ~1 + obstruct + adhere + nodes.ds + surg + rx + age, data = colon_subset_recurrenc
```

```
autoplot(aa_fit)
```



```
summary(aa_fit)
```

```
## $table
##                          slope          coef     se(coef)          z
## Intercept         6.597051e-04  1.377027e-03 4.064307e-04  3.3880990
## obstructobstruct  2.227032e-04  3.443914e-04 1.968629e-04  1.7493973
## adhereadhere      2.365420e-04  4.920522e-04 2.279335e-04  2.1587532
## nodes.ds>3        8.565802e-04  1.379027e-03 1.811496e-04  7.6126428
## surglong          1.566790e-04  3.878978e-04 1.706020e-04  2.2736996
## rxLev            -5.158377e-06 -2.824746e-05 1.872186e-04 -0.1508795
## rxLev+5FU        -3.045652e-04 -6.412875e-04 1.661360e-04 -3.8600152
## age              -2.435937e-06 -4.691427e-06 6.387975e-06 -0.7344153
##                             p
## Intercept        7.037886e-04
## obstructobstruct 8.022236e-02
## adhereadhere     3.086932e-02
## nodes.ds>3       2.685473e-14
## surglong         2.298405e-02
## rxLev            8.800707e-01
## rxLev+5FU        1.133800e-04
## age              4.626957e-01
##
```

```
## $test
## [1] "aalen"
##
## $test.statistic
##        Intercept obstructobstruct     adhereadhere       nodes.ds>3
##        13.185956        15.070926        17.522066        83.902510
##         surglong            rxLev        rxLev+5FU              age
##        22.225231        -1.387123       -32.924325      -189.973929
##
## $test.var
##             b0
## b0   15.146459  -4.2233453    1.6837933  -3.3588005 -5.334958  -8.212835
##      -4.223345  74.2169205   -1.3006687   0.3969698 -4.690905   2.467741
##       1.683793  -1.3006687   65.8816791  -0.6391166 -2.534853  -2.415475
##      -3.358800   0.3969698   -0.6391166 121.4727883  7.007111  -1.419363
##      -5.334958  -4.6909046   -2.5348534   7.0071108 95.549050   6.417505
##      -8.212835   2.4677410   -2.4154748  -1.4193632  6.417505  84.521926
##      -5.315153  -1.1534131    2.0218232  -5.1914204  6.839721  45.191605
##    -942.534084 135.4896632 -234.8021720  54.8798699 80.436713 -19.197271
##
## b0   -5.315153  -942.53408
##      -1.153413   135.48966
##       2.021823  -234.80217
##      -5.191420    54.87987
##       6.839721    80.43671
##      45.191605   -19.19727
##      72.753808  -240.53670
##    -240.536700 66912.08626
##
## $test.var2
## NULL
##
## $chisq
##          [,1]
## [1,] 90.66855
##
## $n
## [1] 911 371 371
##
## attr(,"class")
## [1] "summary.aareg"
```