MT5765 Medical Stats
Practical2.

Ziteng Dong

Semester 2 2024

I hereby declare that this is my own work and that I have not reproduced, without acknowledgement, the work of another.

Ziteng Dong

Declaration

Please make the following declaration at the top of your work: "I hereby declare that this is my own work and that I have not reproduced, without acknowledgement, the work of another", and ensure that you are identified as the author of your submission.

Administrivia

This is the second assessed practical for MT5765 Medical Statistics in 2023/4. You are required to complete the practical by adding to this document and then submitting the both the R markdown and the compiled report.

This practical is worth 15% of the total mark and will be assessed on the execution, presentation and justification of methods encountered on the course. It may be necessary to extend those methods beyond the use cases you have already seen, but not by much, and only a small proportion of the mark will be dependent on this.

The R markdown language allows one to combine code, analyses and explanatory text in a seamless document. Do not neglect the last of these three.

There are two main sections. The first uses a data set we haven't seen before, and the second uses the prostate cancer data that we have looked at for some time, but adds in gene expression data that we have not previously encountered. You will not be penalized if you do not understand/interpret the gene expression data. For the purposes of this practical, just treat them as continuous covariantes. The two sections are equally weighted.

In order to generate a personalized data set, please set the **seedgen** variable to be either your matriculation number or university username (in either case as a string replacing "abc1" in the following R code).

Note that since the release of this practical was delayed by concerns about access to the prostate data set, I have pushed back the submission deadline to April 1st.

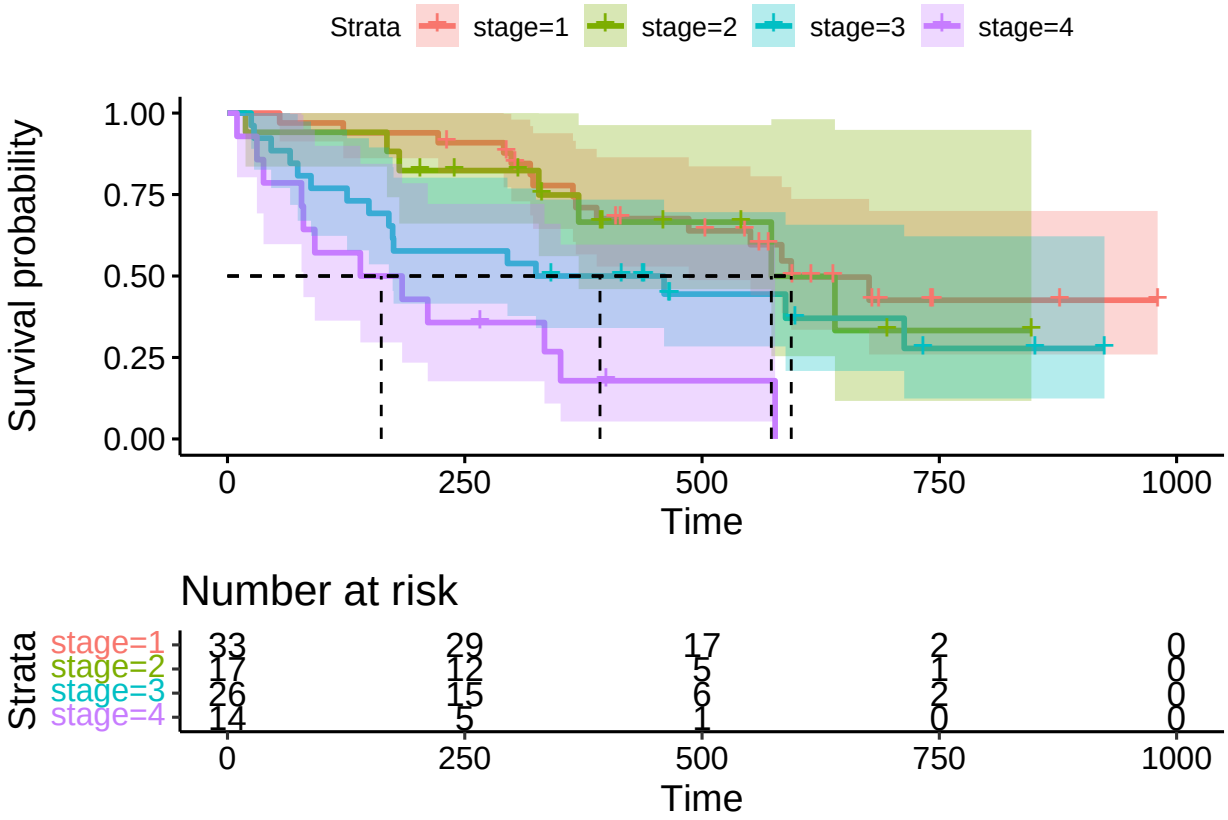
Cancer data

The dataset CANCER contains data for 90 men (event coded as 1 for death, 0 for censored). There are two covariates: tumour stage, and age. In the following analyses we are interested in estimating the differences in survival between different tumour stages. Age should be accounted for as you see fit; You may use it as it is, transform it, convert it to a factor or ignore it.

1. Present the survival data in the most informative figure you can. (2)

Since we are interested in the difference in survival between tumour stages, it is a natural choice to fit a survival model with data divided into different groups based on tumour stage. And we specify `conf.type=log` for a stable and well-behaved confidence interval.

```
# Fit survival model with data divided by stgae
fit_cancer <- survfit(Surv(time, event) ~ stage,
                      data = CANCER,
                      conf.type="log")

# Draw a K-M survival plot
ggsurvplot(fit_cancer,
           data = CANCER,
           risk.table = TRUE,
           surv.median.line = "hv",
           tables.height = 0.32,
           conf.int = TRUE)
```



2. Provide confidence intervals for the median survival associated with tumour stage 4 using what we referred to in class as 'The Delta method'. (2)

```
# Create survival fit object with data at stage 4
fit_cancer_S4 <- survfit(Surv(time, event) ~ 1,
                         data = subset(CANCER, stage == 4),
                         conf.type="log")
```

```
# Gradient of S(t) = 0.5 at stage 4

sur_45 <- max(fit_cancer_S4$surv[fit_cancer_S4$surv <= 0.45]) # largest S(t) before 0.45
sur_55 <- min(fit_cancer_S4$surv[fit_cancer_S4$surv >= 0.55]) # smallest S(t) after 0.55

time_45 <- min(fit_cancer_S4$time[fit_cancer_S4$surv == sur_45]) # time when S(t) = 0.45
time_55 <- min(fit_cancer_S4$time[fit_cancer_S4$surv == sur_55]) # time when S(t) = 0.55

# Calculate gradient
gradient <- (sur_55 - sur_45) / (time_55 - time_45)

gradient
```

```
## [1] -0.001552795
```

```
# Variance of log(S(t))

# Find the median time; first time where S(t) <= 0.5
time_median <- min(fit_cancer_S4$time[fit_cancer_S4$surv <= 0.5])

# Store index of median time
time_index <- which(fit_cancer_S4$time == time_median)

# Extract the variance of log(S(t))
log_sur_var <- fit_cancer_S4$std.err[time_index] ** 2

# Calculate variance of S(t)
sur_var <- (fit_cancer_S4$surv[time_index] ** 2) * log_sur_var

sur_var
```

```
## [1] 0.01785714
```

```
# Confidence interval for median survival

# Calculate standard error of median survival
time_median_var <- sur_var / (gradient ** 2)
time_median_se <- sqrt(time_median_var)

c(time_median, time_median_se)
```

```
## [1] 140.00000  86.05812
```

```r
# Calculate confidence interval
time_median + c(-1.96, 1.96) * time_median_se
```

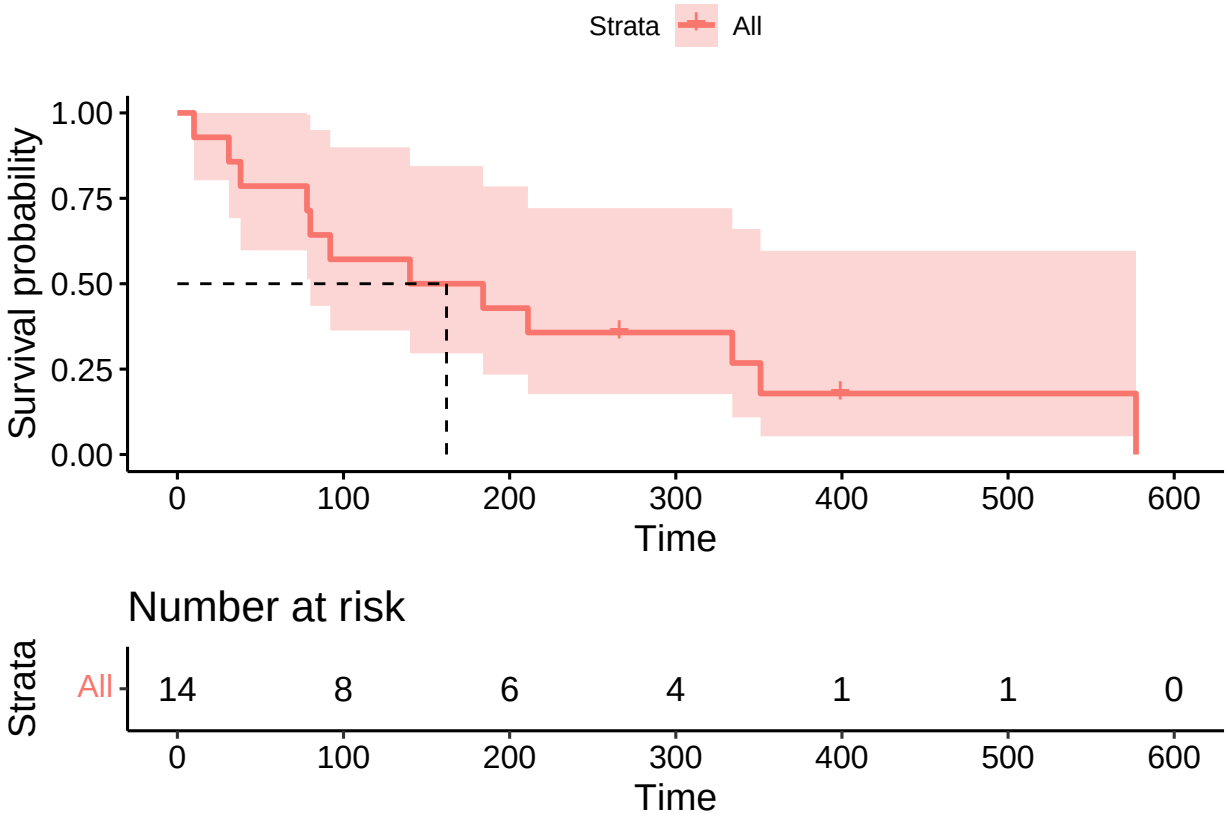
```
## [1] -28.67391 308.67391
```

The confidence interval for median survival of stage 4 is [-28.67, 308.67].

It might seem surprising to have a negative lower value at first glance, but when looking back to the number of patients at risk around median survival, which is lower than 10 , it is reasonable to expect a large variance and a "wide" confidence interval. And as Delta method returns us an asymptotic result, it needs a large sample size to have an adequate performance.

However, a negative value may not be meaningful in our case as patients must be alive when involved in the study at the beginning. Therefore, if we decide to use this result, we need to set the lower value at 0 at least. Or we can consider lower our confidence from 95% to 90% to have a positive interval, but it might not be a good choice.

```r
ggsurvplot(fit_cancer_S4,
           data = CANCER,
           risk.table = TRUE,
           surv.median.line = "hv",
           tables.height = 0.3)
```



3. Perform (a) non-parametric test(s) for differences in survival between tumour stages. (2)

```r
diffobject1 <- survdiff(Surv(time, event) ~ stage, data = CANCER)
diffobject1
```

```
## Call:
## survdiff(formula = Surv(time, event) ~ stage, data = CANCER)
##
##          N Observed Expected (O-E)^2/E (O-E)^2/V
## stage=1 33       15    22.91     2.728     5.109
## stage=2 17        7     9.60     0.703     0.874
## stage=3 26       16    13.36     0.523     0.717
## stage=4 14       12     4.14    14.930    16.743
##
##  Chisq= 19.4  on 3 degrees of freedom, p= 2e-04
```

```r
1-pchisq(14.93, 3)
```

```
## [1] 0.001877456
```

```r
1-pchisq(19.4, 3)
```

```
## [1] 0.0002259703
```

By leveraging the `survdiff` function, we can easily have both standard Chi-square test and Mantel-Haenszel test for identity of survival distribution of different groups. Both methods can be used for the case with more than 3 groups, and their null hypotheses are that survival between stages are the same.

From the Chi-square tests results, with small p-values, we can be sure that there are differences in survival between tumour stages.

Since we have age profile for patients and suspect that patients at different age may have different survival curve within a stage, it is reasonable to look at the distribution of stage at different age.

```r
# Minimum age
min(CANCER$ages)
```

```
## [1] 39
```

```r
# Maximum age
max(CANCER$ages)
```

```
## [1] 85
```

```r
# Age range
max(CANCER$ages) - min(CANCER$ages)
```

```
## [1] 46
```

In order to make it practical, we decide to separate ages into three groups for that there are not many patients and too many groups are not beneficial on detecting the data imbalance. If we divide the age range, which spans from 39 to 85, into three equal parts, each age group covers approximately 15 years.

```
# Change age into categorical variable
CANCER$cat_ages <- cut(CANCER$ages,                          # targeted column
                       breaks = c(39, 55, 70, 85),           # set breaks
                       labels = c("39-55", "56-70", "71-85"), # set labels for intervals
                       include.lowest = TRUE)                # include lower value in interval

# Cross-tabulation of age and stage

# Convert table into data frame
age_stage <- as.matrix.data.frame(table(CANCER$stage, CANCER$cat_ages))

# Set names for rows and columns
rownames(age_stage) <- c("stage 1", "stage 2", "stage 3", "stage 4")
colnames(age_stage) <- c("39-55", "56-70", "71-85")

# Compute total number of patients
age_stage <- cbind(age_stage, total = rowSums(age_stage))
age_stage <- rbind(age_stage, total = colSums(age_stage))

age_stage
```

```
##         39-55 56-70 71-85 total
## stage 1     9    14    10    33
## stage 2     5     7     5    17
## stage 3     4    14     8    26
## stage 4     3     2     9    14
## total      21    37    32    90
```

Based the separation method mentioned above, it doesn't seem to have data imbalance problem in these data. But we will still use `survdiff` function as before to help test the differences in survivals, and stratified by ages.

```
diffobject2 <- survdiff(Surv(time, event) ~ stage + strata(cat_ages), data = CANCER)
diffobject2
```

```
## Call:
## survdiff(formula = Surv(time, event) ~ stage + strata(cat_ages),
##     data = CANCER)
##
##          N Observed Expected (O-E)^2/E (O-E)^2/V
## stage=1 33       15    22.90     2.726     5.243
## stage=2 17        7     8.35     0.219     0.284
## stage=3 26       16    13.64     0.409     0.587
## stage=4 14       12     5.11     9.297    11.735
##
##  Chisq= 14  on 3 degrees of freedom, p= 0.003
```

4. Use a bootstrapping approach to estimate the difference in survival between tumour stage 1 and tumour stage 4. (2)

```r
# Initializing bootstrap
numrep <- 500 # set number of bootstrapped samples
bootmedian1 <- rep(NA, 500) # results for stage 1
bootmedian4 <- rep(NA, 500) # results for stage 4

set.seed(1)
# Loop across all samples (Stage 1)
for (i in seq(numrep)){

  # Re-sample with replacement
  sampling <- sample(99, 99, replace = TRUE)

  # Fit survival model
  fit1 <- survfit(Surv(time[sampling], event[sampling]) ~ 1,
                  data = subset(CANCER, stage == 1))

  # Store results
  bootmedian1[i] <- summary(fit1)$table[7]
}

set.seed(4)
# Loop across all samples (Stage 4)
for (i in seq(numrep)){

  # Re-sample with replacement
  sampling <- sample(99, 99, replace = TRUE)

  # Fit survival model
  fit4 <- survfit(Surv(time[sampling], event[sampling]) ~ 1,
                  data = subset(CANCER, stage == 4))

  # Store results
  bootmedian4[i] <- summary(fit4)$table[7]
}
```

```r
# Check missing values in bootstrap results
table(is.na(bootmedian1))
```

```
##
## FALSE  TRUE
##   381   119
```

```r
table(is.na(bootmedian4))
```

```
##
## FALSE  TRUE
##   499     1
```

```r
# 2.5% and 97.5% quantile of bootstrap results
quantile(bootmedian1, c(0.025, 0.975), na.rm = TRUE)
```

```
##   2.5%  97.5%
## 407.75 676.00
```

```r
quantile(bootmedian4, c(0.025, 0.975), na.rm = TRUE)
```

```
##  2.5% 97.5%
##    78   351
```

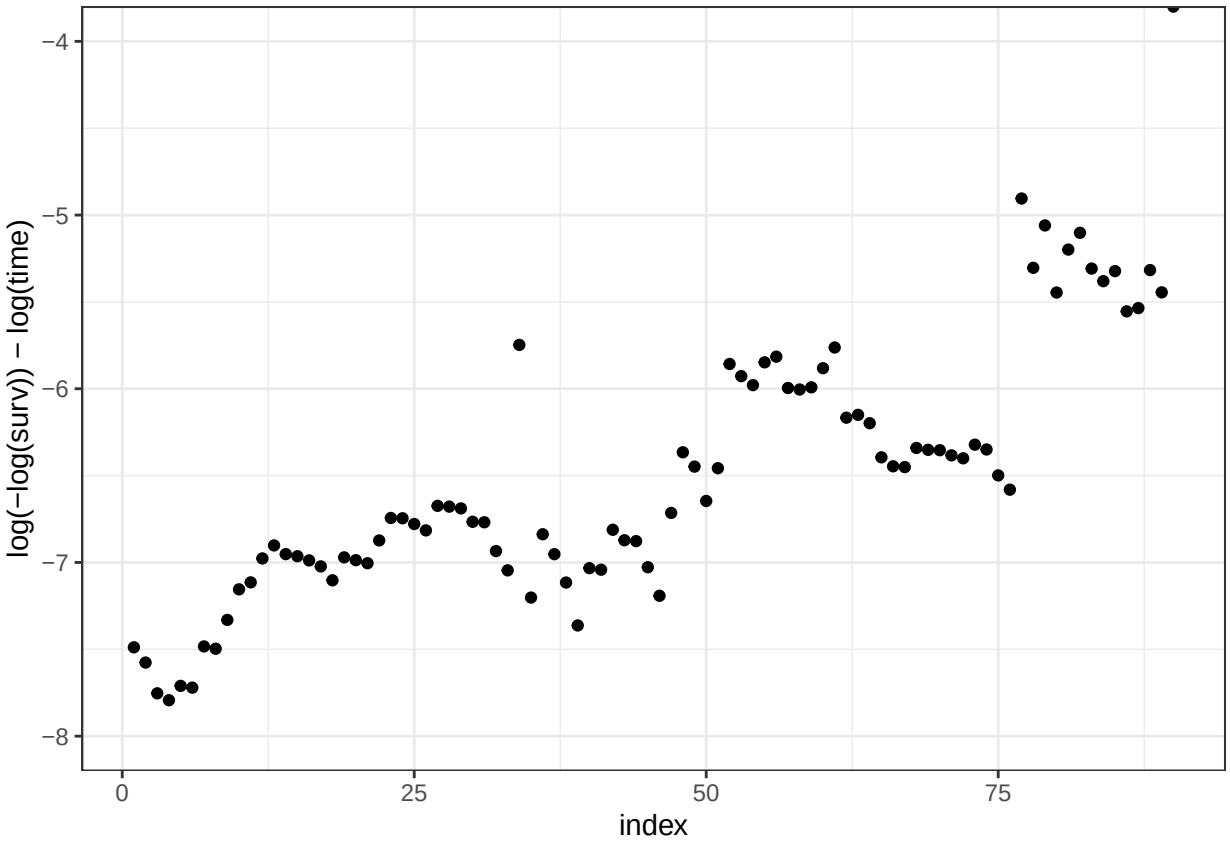
We can see from the 95% confidence intervals that there is no overlap at all, which means there is difference in survival median between stage 1 and stage 4. But we cannot be confident with this result as there are too many missing values in the bootstrap result for stage 1, and we simply dropped all of them. The loss of these values may have significant impact on the quantile of the bootstrap result, which may alter our conclusion.

5. Decide whether you think an exponential model would be appropriate for modelling these data. (2)

Under an exponential model, we assume the hazard rate is constant. To assess the validation of this assessment, we can plot $log(-log(S(t))) - log(t)$ (which is equal to log(λ)) in sequence. And if the hazard rate is constant, then we would expect the data points are nearly in a horizontal line.

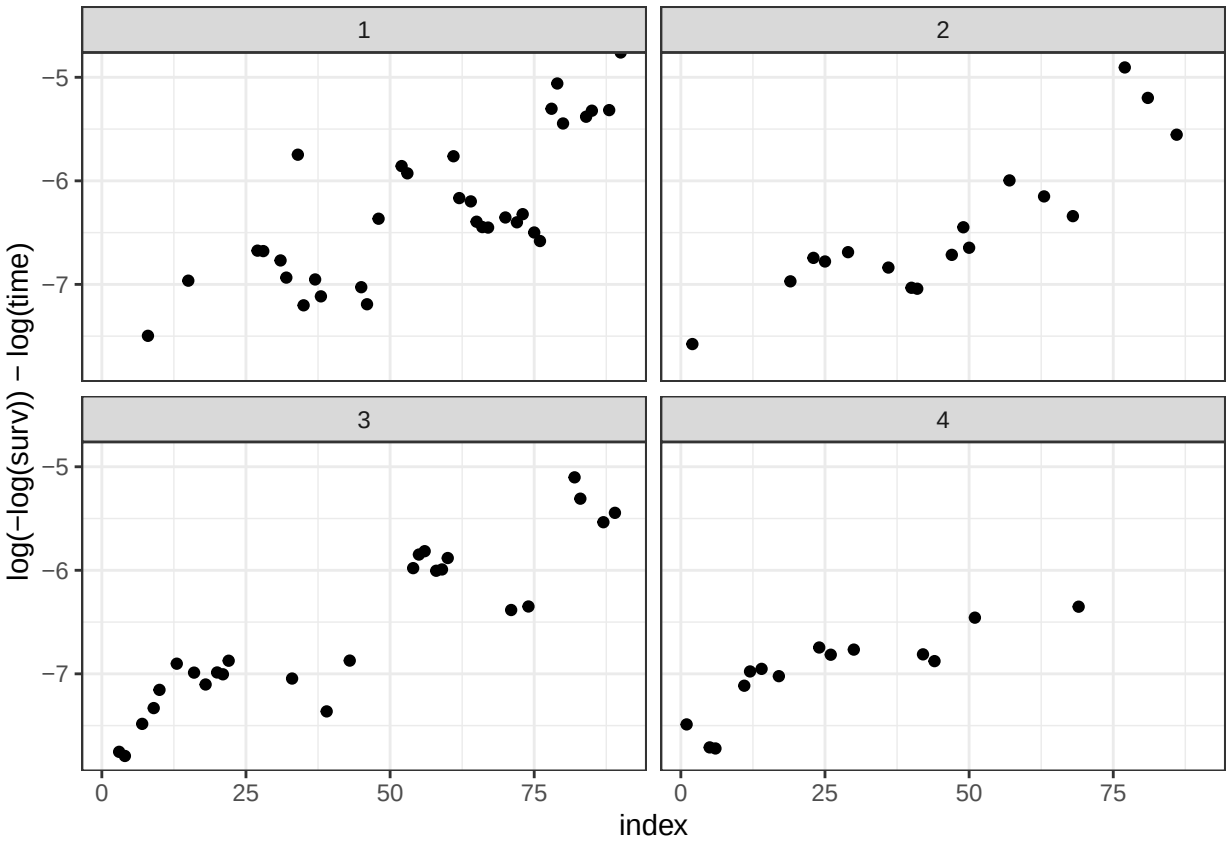
Based on the scatter plot below, we can see the log(λ) has an increasing trend across index, which indicates the assumption of constant hazard ratio is not valid. Therefore, we can conclude that an exponential model might not be appropriate for modelling these data as a whole.

```r
# Create a data frame to store values
df <- data.frame(surv = fit_cancer$surv,
                 time = fit_cancer$time,
                 index = 1: 90)

# Plot log(-log(S(t)))-log(t)
ggplot(df) + ylim(-8, -4) + theme_bw() +
  geom_point(aes(index, log(-log(surv))-log(time)))
```



However, one may think relaxing the assumption on constant hazard by assuming constant ratio for each stage. Therefore, I decide to draw same plot above but one for each stage.

```
# Add stage
df$stage <- CANCER$stage

# Plot log(-log(S(t)))-log(t) for each stage
ggplot(df) + theme_bw() + facet_wrap(~ stage) +
  geom_point(aes(index, log(-log(surv))-log(time)))
```



The data points in the plots still don't present us any sign of horizontal distribution. And we can conclude that even at each stage, the assumption of constant hazard is not valid, and an exponential model is not appropriate for these data.

In each case explain/justify your decisions.

Prostate data

In this section, we look again at the prostate data that have featured in the course. There is a code chunk in the Rmd file that will generate two R objects for you: pd_camcap and e_camcap. Note that the e_camcap object is generated specifically for you. It is important that you replace the "abc1" string in the first code chunk of the Rmd file, and run all of the code chunks in order to generate the correct .

As well as survival time and 'event' in the pd_camcap data object, there are three covariates (ERG, Age and PSA). Additionally there are 20 continuous covariates in the form of gene expression data in the e_camcap data object.

Here you will build the best model for survival in terms of those 23 covariates (or transformations of those covariates) that you can. The final model must be a semi-parametric Cox proportional hazards model, but you can eliminate covariates from consideration by any means you choose.

As ever explain, and justify, any decisions. In particular, it is important that you show how the model evolves from your first attempt.

Tasks:

1. Find the best Cox proportional Hazards model for survival in terms of the 23 covariates that you can. (6)

The first step is combine two data sets by "geo_accession" to facilitate model fitting.

```
# Convert e_camcap into data frame
df_e_camcap <- as.data.frame(t(e_camcap))

# Compute mean of each columns
means <- apply(df_e_camcap, 2, mean, na.rm = TRUE)

# Impute missing value with mean
for (i in colnames(df_e_camcap)){                          # for loop begins
  df_e_camcap[ , i][is.na(df_e_camcap[ , i])] <- means[i]
}                                                          # for loop ends

# Create new column with values same with rownames
df_e_camcap$geo_accession <- rownames(df_e_camcap)

# Merge two data sets
camcap <- inner_join(df_e_camcap, pd_camcap, by = "geo_accession")

camcap <- na.omit(camcap)
```

We have spotted several missing values in e_camcap. Since values in this data set are continuous, we can use the mean of each column to fill these missing values. But we decide to drop rows with missing value in pd_camcap after combined with e_camcap as there are only six missing values and five of them are binary variables, which are not suitable for imputation.

We fit our first model with all available covariates, and there are only 7 of them are significant. In fact we tried to involve interactions in the model, because we are interested in the interactions between gene expressions and other three covariates. However, we got warning for running out of iteration if we do so. We also tried to increase the maximum number of iterations, but it returned us infinite estimated coefficients. We think the reasons why we are in this situation might be that 1)we may have collinearity problem; 2)data may correlated with time; 3)we may not have enough observations. Now we can only fit a model without interactions.

```
# Fit a Cox proportional harzard model
coxfit1 <- coxph(Surv(Time, Event) ~ . -geo_accession, data = camcap, ties = "efron")

summary(coxfit1)
```

```
## Call:
## coxph(formula = Surv(Time, Event) ~ . - geo_accession, data = camcap,
##     ties = "efron")
##
##   n= 105, number of events= 19
##
##                     coef  exp(coef)   se(coef)      z Pr(>|z|)
## ILMN_1683604 -1.700e+01  4.125e-08  6.674e+00 -2.548  0.01084 *
## ILMN_1685079  1.814e+00  6.137e+00  2.443e+00  0.743  0.45761
## ILMN_1700860 -7.107e-02  9.314e-01  2.515e-01 -0.283  0.77749
## ILMN_1706483  1.765e+00  5.841e+00  1.597e+00  1.105  0.26911
## ILMN_1717690 -2.797e+00  6.101e-02  3.811e+00 -0.734  0.46302
## ILMN_1727288  1.598e+00  4.944e+00  1.942e+00  0.823  0.41046
## ILMN_1746515  2.037e+01  7.026e+08  8.380e+00  2.431  0.01507 *
```

```
## ILMN_1753449 -5.063e-01  6.027e-01  2.495e-01 -2.029  0.04244 *
## ILMN_1759448 -1.008e+01  4.191e-05  4.242e+00 -2.376  0.01748 *
## ILMN_1781983  1.163e-02  1.012e+00  1.521e+00  0.008  0.99390
## ILMN_1796074  3.755e+00  4.274e+01  2.126e+00  1.767  0.07729 .
## ILMN_1806862  6.243e+00  5.144e+02  4.640e+00  1.345  0.17851
## ILMN_1907467 -7.904e+00  3.692e-04  4.310e+00 -1.834  0.06668 .
## ILMN_2376313 -3.070e-01  7.356e-01  3.435e-01 -0.894  0.37144
## ILMN_3211132 -1.392e+00  2.486e-01  1.135e+00 -1.226  0.22010
## ILMN_3242603 -4.013e+00  1.808e-02  2.290e+00 -1.752  0.07976 .
## ILMN_3266606 -1.271e+00  2.806e-01  5.247e-01 -2.422  0.01543 *
## ILMN_3290199 -2.982e+00  5.069e-02  2.613e+00 -1.141  0.25374
## ILMN_3304130  4.871e+00  1.304e+02  1.917e+00  2.541  0.01104 *
## ILMN_3308225  1.878e+00  6.541e+00  1.916e+00  0.980  0.32702
## ERG          -5.277e+00  5.105e-03  1.922e+00 -2.745  0.00604 **
## Age           4.366e-02  1.045e+00  5.512e-02  0.792  0.42824
## PSA          -6.600e-02  9.361e-01  1.304e-01 -0.506  0.61269
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##              exp(coef) exp(-coef) lower .95 upper .95
## ILMN_1683604 4.125e-08  2.424e+07 8.608e-14 1.977e-02
## ILMN_1685079 6.137e+00  1.629e-01 5.114e-02 7.365e+02
## ILMN_1700860 9.314e-01  1.074e+00 5.689e-01 1.525e+00
## ILMN_1706483 5.841e+00  1.712e-01 2.553e-01 1.336e+02
## ILMN_1717690 6.101e-02  1.639e+01 3.480e-05 1.070e+02
## ILMN_1727288 4.944e+00  2.023e-01 1.100e-01 2.222e+02
## ILMN_1746515 7.026e+08  1.423e-09 5.169e+01 9.551e+15
## ILMN_1753449 6.027e-01  1.659e+00 3.696e-01 9.829e-01
## ILMN_1759448 4.191e-05  2.386e+04 1.028e-08 1.709e-01
## ILMN_1781983 1.012e+00  9.884e-01 5.129e-02 1.996e+01
## ILMN_1796074 4.274e+01  2.340e-02 6.631e-01 2.756e+03
## ILMN_1806862 5.144e+02  1.944e-03 5.773e-02 4.584e+06
## ILMN_1907467 3.692e-04  2.709e+03 7.913e-08 1.722e+00
## ILMN_2376313 7.356e-01  1.359e+00 3.752e-01 1.442e+00
## ILMN_3211132 2.486e-01  4.022e+00 2.689e-02 2.299e+00
## ILMN_3242603 1.808e-02  5.530e+01 2.032e-04 1.610e+00
## ILMN_3266606 2.806e-01  3.564e+00 1.003e-01 7.847e-01
## ILMN_3290199 5.069e-02  1.973e+01 3.026e-04 8.491e+00
## ILMN_3304130 1.304e+02  7.668e-03 3.048e+00 5.580e+03
## ILMN_3308225 6.541e+00  1.529e-01 1.530e-01 2.796e+02
## ERG          5.105e-03  1.959e+02 1.180e-04 2.210e-01
## Age          1.045e+00  9.573e-01 9.377e-01 1.164e+00
## PSA          9.361e-01  1.068e+00 7.250e-01 1.209e+00
##
## Concordance= 0.911  (se = 0.037 )
## Likelihood ratio test= 57.32  on 23 df,   p=9e-05
## Wald test            = 16.97  on 23 df,   p=0.8
## Score (logrank) test = 51.14  on 23 df,   p=7e-04
```

We run the VIF of covariates, and results are below. Many covariates have VIF larger than 5 and even 10, which indicates that we are facing collinearity problem. The consequence is that the associated p-values of collinear terms may be too large.

```r
# VIF
car::vif(coxfit1)
```

```
## ILMN_1683604 ILMN_1685079 ILMN_1700860 ILMN_1706483 ILMN_1717690 ILMN_1727288
##    10.209916     7.229220     2.728976     6.220389     7.740235     3.106440
## ILMN_1746515 ILMN_1753449 ILMN_1759448 ILMN_1781983 ILMN_1796074 ILMN_1806862
##    13.327826     3.227787     3.017647     4.735184     5.570426     7.987529
## ILMN_1907467 ILMN_2376313 ILMN_3211132 ILMN_3242603 ILMN_3266606 ILMN_3290199
##     7.178159     4.933022    10.050360     5.626926    10.174358    17.047412
## ILMN_3304130 ILMN_3308225          ERG          Age          PSA
##    25.365448     3.593867    12.475712     1.630088     2.958716
```

To address collinearity problem, the simplest way is dropping highly collinear terms. But we will not drop ERG as it is both binary variable and what we interested in.

```r
# Remove collinear terms from model
coxfit2 <- update(coxfit1, .~.-ILMN_1683604
                              -ILMN_1746515
                              -ILMN_3211132
                              -ILMN_3266606
                              -ILMN_3290199
                              -ILMN_3304130)
```

```r
# VIF
car::vif(coxfit2)
```

```
## ILMN_1685079 ILMN_1700860 ILMN_1706483 ILMN_1717690 ILMN_1727288 ILMN_1753449
##     3.056172     1.577450     2.925960     3.212205     1.575956     2.862762
## ILMN_1759448 ILMN_1781983 ILMN_1796074 ILMN_1806862 ILMN_1907467 ILMN_2376313
##     1.932583     1.902539     1.812417     2.361241     1.725179     2.354860
## ILMN_3242603 ILMN_3308225          ERG          Age          PSA
##     3.837079     1.243796     2.366851     1.403629     1.672669
```

After excluding highly collinearity terms, we don't have the collinearity issues any more. We tried to fit a model with interactions. Unfortunately, we were given warning again. (# we decide no to show this model, as we did not develop models from it.)

We think it might be a good choice to select some gene expressions with significant effect first, and reduce the dimension of data. But we cannot remove variables just based on the significance of the summary output as it is measured separately. Since we don't have interaction terms yet, we decide to use Type II ANOVA to select variables.

```r
# Type II ANOVA
car::Anova(coxfit2)
```

```
## Analysis of Deviance Table (Type II tests)
##              LR Chisq Df Pr(>Chisq)
## ILMN_1685079   2.1604  1    0.14161
## ILMN_1700860   0.0441  1    0.83359
## ILMN_1706483   0.0004  1    0.98484
## ILMN_1717690   0.2792  1    0.59725
## ILMN_1727288   0.8903  1    0.34538
```

```
## ILMN_1753449   1.6868  1    0.19402
## ILMN_1759448   3.4175  1    0.06451 .
## ILMN_1781983   4.0384  1    0.04448 *
## ILMN_1796074   3.2147  1    0.07298 .
## ILMN_1806862   0.0063  1    0.93673
## ILMN_1907467   5.2116  1    0.02244 *
## ILMN_2376313   0.1457  1    0.70268
## ILMN_3242603   4.0305  1    0.04469 *
## ILMN_3308225   0.6531  1    0.41901
## ERG            3.3895  1    0.06561 .
## Age            0.1271  1    0.72150
## PSA            0.0599  1    0.80659
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now there are only three gene having significant effect. Because we are interested in the interactions between gene expressions and other three covariates, ERG, AGE and PSA are not removed from model.

```
# Fit model with interactions
coxfit3 <- coxph(Surv(Time, Event) ~ (ILMN_1781983 + ILMN_1907467 +
                                      ILMN_3242603) * (ERG + Age + PSA),
                 data = camcap, ties = "efron")
```

Since we are estimating parameters based on partial likelihood, which is similar to likelihood, we can choose AIC as the criteria for model selection.

```
# Stepwise AIC
MASS::stepAIC(coxfit3)
```

```
## Start:  AIC=129.89
## Surv(Time, Event) ~ (ILMN_1781983 + ILMN_1907467 + ILMN_3242603) *
##     (ERG + Age + PSA)
##
##                     Df    AIC
## - ILMN_3242603:PSA   1 127.89
## - ILMN_1907467:PSA   1 127.90
## - ILMN_1907467:Age   1 127.96
## - ILMN_1781983:PSA   1 128.10
## - ILMN_3242603:Age   1 128.21
## - ILMN_3242603:ERG   1 128.32
## - ILMN_1781983:Age   1 129.20
## <none>                 129.89
## - ILMN_1907467:ERG   1 130.04
## - ILMN_1781983:ERG   1 136.60
##
## Step:  AIC=127.89
## Surv(Time, Event) ~ ILMN_1781983 + ILMN_1907467 + ILMN_3242603 +
##     ERG + Age + PSA + ILMN_1781983:ERG + ILMN_1781983:Age + ILMN_1781983:PSA +
##     ILMN_1907467:ERG + ILMN_1907467:Age + ILMN_1907467:PSA +
##     ILMN_3242603:ERG + ILMN_3242603:Age
##
##                     Df    AIC
```

```
## - ILMN_1907467:PSA  1 125.90
## - ILMN_1907467:Age  1 125.96
## - ILMN_1781983:PSA  1 126.19
## - ILMN_3242603:Age  1 126.21
## - ILMN_3242603:ERG  1 126.34
## - ILMN_1781983:Age  1 127.37
## <none>                127.89
## - ILMN_1907467:ERG  1 128.10
## - ILMN_1781983:ERG  1 135.02
##
## Step:  AIC=125.9
## Surv(Time, Event) ~ ILMN_1781983 + ILMN_1907467 + ILMN_3242603 +
##     ERG + Age + PSA + ILMN_1781983:ERG + ILMN_1781983:Age + ILMN_1781983:PSA +
##     ILMN_1907467:ERG + ILMN_1907467:Age + ILMN_3242603:ERG +
##     ILMN_3242603:Age
##
##                     Df    AIC
## - ILMN_1907467:Age  1 123.96
## - ILMN_1781983:PSA  1 124.23
## - ILMN_3242603:Age  1 124.28
## - ILMN_3242603:ERG  1 124.41
## - ILMN_1781983:Age  1 125.38
## <none>                125.90
## - ILMN_1907467:ERG  1 126.15
## - ILMN_1781983:ERG  1 133.10
##
## Step:  AIC=123.96
## Surv(Time, Event) ~ ILMN_1781983 + ILMN_1907467 + ILMN_3242603 +
##     ERG + Age + PSA + ILMN_1781983:ERG + ILMN_1781983:Age + ILMN_1781983:PSA +
##     ILMN_1907467:ERG + ILMN_3242603:ERG + ILMN_3242603:Age
##
##                     Df    AIC
## - ILMN_1781983:PSA  1 122.25
## - ILMN_3242603:Age  1 122.31
## - ILMN_3242603:ERG  1 122.46
## - ILMN_1781983:Age  1 123.40
## <none>                123.96
## - ILMN_1907467:ERG  1 124.17
## - ILMN_1781983:ERG  1 131.29
##
## Step:  AIC=122.25
## Surv(Time, Event) ~ ILMN_1781983 + ILMN_1907467 + ILMN_3242603 +
##     ERG + Age + PSA + ILMN_1781983:ERG + ILMN_1781983:Age + ILMN_1907467:ERG +
##     ILMN_3242603:ERG + ILMN_3242603:Age
##
##                     Df    AIC
## - PSA                1 120.27
## - ILMN_3242603:Age  1 120.63
## - ILMN_3242603:ERG  1 121.07
## - ILMN_1781983:Age  1 121.42
## <none>                122.25
## - ILMN_1907467:ERG  1 122.42
## - ILMN_1781983:ERG  1 130.53
##
```

```
## Step:  AIC=120.27
## Surv(Time, Event) ~ ILMN_1781983 + ILMN_1907467 + ILMN_3242603 +
##     ERG + Age + ILMN_1781983:ERG + ILMN_1781983:Age + ILMN_1907467:ERG +
##     ILMN_3242603:ERG + ILMN_3242603:Age
##
##                     Df    AIC
## - ILMN_3242603:Age   1 118.65
## - ILMN_3242603:ERG   1 119.09
## - ILMN_1781983:Age   1 119.61
## <none>                 120.27
## - ILMN_1907467:ERG   1 120.46
## - ILMN_1781983:ERG   1 128.54
##
## Step:  AIC=118.65
## Surv(Time, Event) ~ ILMN_1781983 + ILMN_1907467 + ILMN_3242603 +
##     ERG + Age + ILMN_1781983:ERG + ILMN_1781983:Age + ILMN_1907467:ERG +
##     ILMN_3242603:ERG
##
##                     Df    AIC
## - ILMN_3242603:ERG   1 117.12
## - ILMN_1781983:Age   1 117.82
## <none>                 118.65
## - ILMN_1907467:ERG   1 118.66
## - ILMN_1781983:ERG   1 126.55
##
## Step:  AIC=117.12
## Surv(Time, Event) ~ ILMN_1781983 + ILMN_1907467 + ILMN_3242603 +
##     ERG + Age + ILMN_1781983:ERG + ILMN_1781983:Age + ILMN_1907467:ERG
##
##                     Df    AIC
## - ILMN_1781983:Age   1 116.42
## - ILMN_1907467:ERG   1 116.73
## <none>                 117.12
## - ILMN_3242603       1 119.46
## - ILMN_1781983:ERG   1 124.55
##
## Step:  AIC=116.42
## Surv(Time, Event) ~ ILMN_1781983 + ILMN_1907467 + ILMN_3242603 +
##     ERG + Age + ILMN_1781983:ERG + ILMN_1907467:ERG
##
##                     Df    AIC
## - Age                1 114.54
## - ILMN_1907467:ERG   1 116.41
## <none>                 116.42
## - ILMN_3242603       1 119.25
## - ILMN_1781983:ERG   1 123.96
##
## Step:  AIC=114.54
## Surv(Time, Event) ~ ILMN_1781983 + ILMN_1907467 + ILMN_3242603 +
##     ERG + ILMN_1781983:ERG + ILMN_1907467:ERG
##
##                     Df    AIC
## - ILMN_1907467:ERG   1 114.44
## <none>                 114.54
```

```
## - ILMN_3242603      1 117.87
## - ILMN_1781983:ERG  1 121.96
##
## Step:  AIC=114.44
## Surv(Time, Event) ~ ILMN_1781983 + ILMN_1907467 + ILMN_3242603 +
##     ERG + ILMN_1781983:ERG
##
##                     Df    AIC
## <none>                 114.44
## - ILMN_3242603      1 118.10
## - ILMN_1781983:ERG  1 120.50
## - ILMN_1907467      1 126.55
```

```
## Call:
## coxph(formula = Surv(Time, Event) ~ ILMN_1781983 + ILMN_1907467 +
##     ILMN_3242603 + ERG + ILMN_1781983:ERG, data = camcap, ties = "efron")
##
##                          coef  exp(coef)   se(coef)      z        p
## ILMN_1781983      2.989e+00  1.987e+01  8.965e-01  3.334 0.000856
## ILMN_1907467     -6.115e+00  2.210e-03  1.806e+00 -3.387 0.000708
## ILMN_3242603     -2.260e+00  1.043e-01  9.926e-01 -2.277 0.022766
## ERG               4.366e+01  9.179e+18  1.665e+01  2.622 0.008737
## ILMN_1781983:ERG -4.752e+00  8.634e-03  1.794e+00 -2.650 0.008060
##
## Likelihood ratio test=26.47  on 5 df, p=7.247e-05
## n= 105, number of events= 19
```

2. Illustrate your final model. (2)

```
# Final model
final_coxfit <- coxph(Surv(Time, Event) ~ ILMN_1781983 + ILMN_1907467 +
                                          ILMN_3242603 + ERG +
                                          ILMN_1781983:ERG,
                      data = camcap, ties = "efron")

summary(final_coxfit)
```

```
## Call:
## coxph(formula = Surv(Time, Event) ~ ILMN_1781983 + ILMN_1907467 +
##     ILMN_3242603 + ERG + ILMN_1781983:ERG, data = camcap, ties = "efron")
##
##   n= 105, number of events= 19
##
##                          coef  exp(coef)   se(coef)      z Pr(>|z|)
## ILMN_1781983      2.989e+00  1.987e+01  8.965e-01  3.334 0.000856 ***
## ILMN_1907467     -6.115e+00  2.210e-03  1.806e+00 -3.387 0.000708 ***
## ILMN_3242603     -2.260e+00  1.043e-01  9.926e-01 -2.277 0.022766 *
## ERG               4.366e+01  9.179e+18  1.665e+01  2.622 0.008737 **
## ILMN_1781983:ERG -4.752e+00  8.634e-03  1.794e+00 -2.650 0.008060 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                  exp(coef) exp(-coef) lower .95 upper .95
```

```
## ILMN_1781983     1.987e+01  5.033e-02 3.428e+00 1.151e+02
## ILMN_1907467     2.210e-03  4.524e+02 6.421e-05 7.609e-02
## ILMN_3242603     1.043e-01  9.587e+00 1.491e-02 7.298e-01
## ERG              9.179e+18  1.089e-19 6.152e+04 1.370e+33
## ILMN_1781983:ERG 8.634e-03  1.158e+02 2.568e-04 2.903e-01
##
## Concordance= 0.829  (se = 0.046 )
## Likelihood ratio test= 26.47  on 5 df,   p=7e-05
## Wald test            = 21.54  on 5 df,   p=6e-04
## Score (logrank) test = 26.11  on 5 df,   p=8e-05
```

Based on the results from stepwise AIC, we have our final model, which has the lowest AIC, with ILMN_1781983, ILMN_1907467, ILMN_3242603, ERG, and interaction between ILMN_1781983 and ERG. The estimated coefficients are 2.99, -6.12, -2.26, 43.7 and -4.75 respectively, and all of them have significant effect on the hazard ratio.

All gene expressions are continuous variables, so the exp(coef) reported in the output is the hazard ratio associated with one unit increase in the gene expression. For example, ILMN_1781983 has exp(coef)=1.99, which means, all else being equal, one unit increase in expression of this gene, the hazard ratio will increase by 1.99. However, we find that the exp(coef) for ERG is extremely high, which means whether the patients have ERG would have great effect on their hazard ratio.

3. Assess whether your final model is suitable. (2)

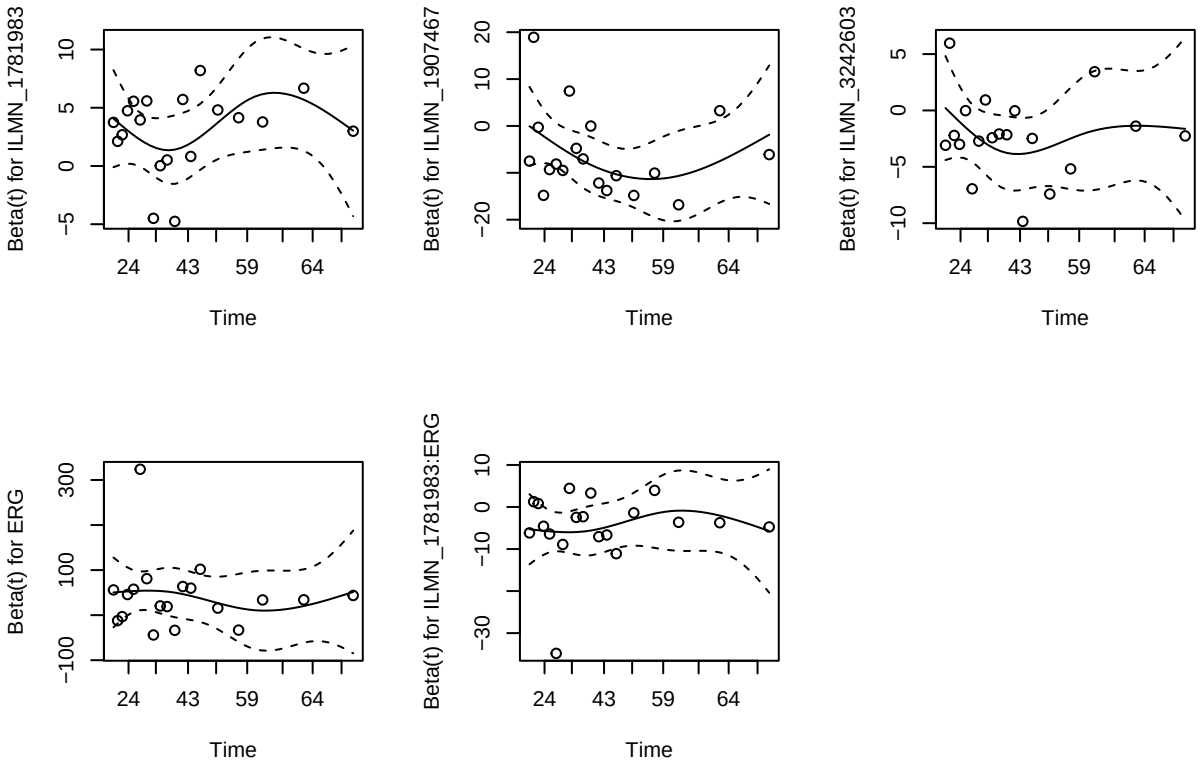
We can obtain tests for proportional hazard assumption form `cox.zph`, which tests the proportional hazards assumption for a Cox regression model fit.

```
# Test for porportional hazard assumption
cox.zph(final_coxfit)
```

```
##                   chisq df    p
## ILMN_1781983     1.8886  1 0.17
## ILMN_1907467     0.5704  1 0.45
## ILMN_3242603     0.2194  1 0.64
## ERG              0.1298  1 0.72
## ILMN_1781983:ERG 0.0864  1 0.77
## GLOBAL           7.1674  5 0.21
```

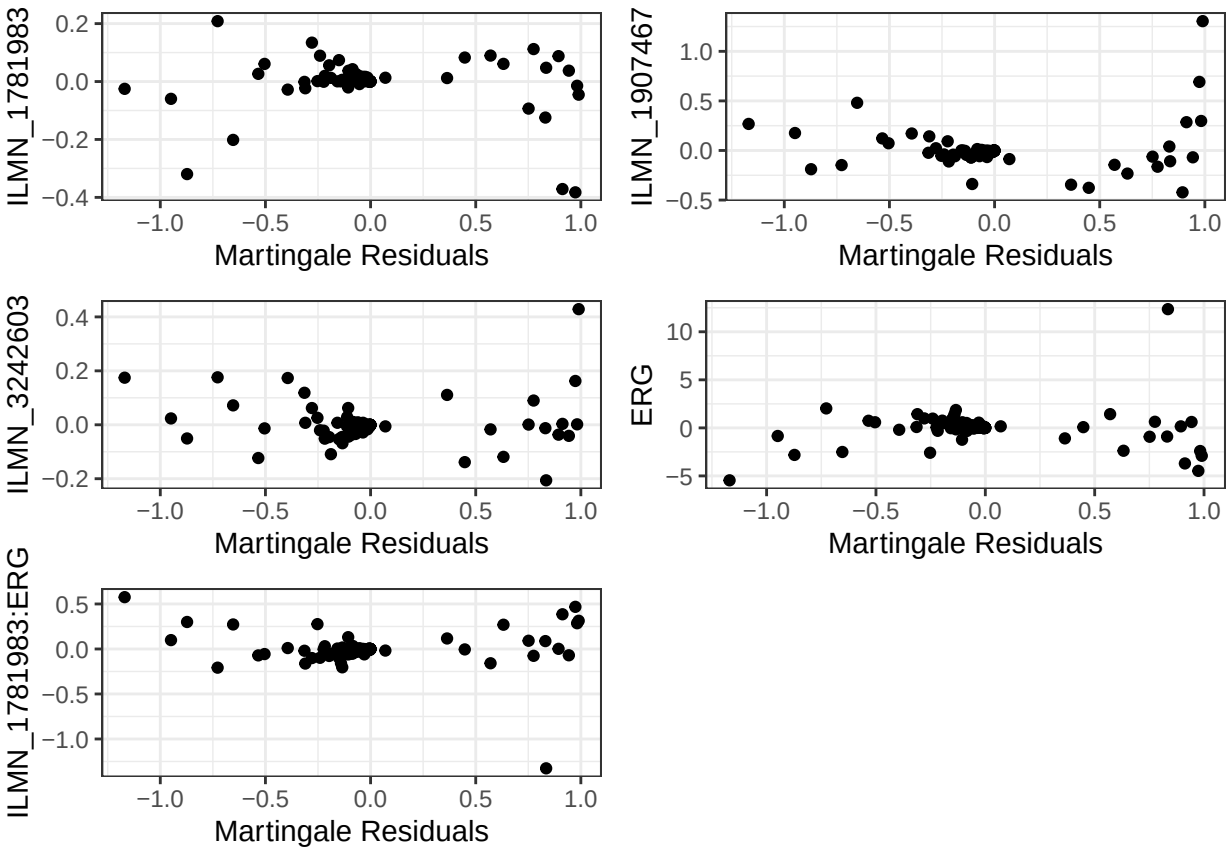
From the tests from "cox.zph", we find that all p-values are larger than 0.05 which means we cannot reject the null hypothesis that the proportional hazards assumption is valid.

```
# Checking proportional hazards
par(mfrow = c(2, 3))

plot(cox.zph(final_coxfit))

par(mfrow = c(1, 1))
```



However, we find Beta(t)s for ILMN_1781983 and ILMN_1907467 are not independent of time when plotted against time. But if the proportional hazard assumption holds, then the true Beta(t) should be a horizontal line. It is very interesting to have a contradicted results. Or maybe we should not rely on our visual.

```
# Influential observations
dfbeta <- residuals(final_coxfit, type = "dfbeta")

Mres <- residuals(final_coxfit, type = "martingale")

p1 <- ggplot() + geom_point(aes(x = Mres, y = dfbeta[, 1])) +
    xlab("Martingale Residuals") + ylab("ILMN_1781983") + theme_bw()
p2 <- ggplot() + geom_point(aes(x = Mres, y = dfbeta[, 2])) +
    xlab("Martingale Residuals") + ylab("ILMN_1907467") + theme_bw()
p3 <- ggplot() + geom_point(aes(x = Mres, y = dfbeta[, 3])) +
    xlab("Martingale Residuals") + ylab("ILMN_3242603") + theme_bw()
p4 <- ggplot() + geom_point(aes(x = Mres, y = dfbeta[, 4])) +
    xlab("Martingale Residuals") + ylab("ERG") + theme_bw()
p5 <- ggplot() + geom_point(aes(x = Mres, y = dfbeta[, 5])) +
    xlab("Martingale Residuals") + ylab("ILMN_1781983:ERG") + theme_bw()
gridExtra::grid.arrange(p1, p2, p3, p4, p5, ncol = 2)
```



Based on the distribution of martingale residuals, we don't see any observations cause great concern about overly-influential problem.

Overall, our model is suitable for these data for that the assumption on proportional hazards is valid and there is no overly influential observation. But we have to point out that the extreme value for estimated coefficient of ERG might be a problem and we did not test if these data are correlated with time.

sessionInfo

Including this as good practice and to aid diagnostics when things go wrong.

```
# Make sure the next line is uncommented prior to submission
sessionInfo()
```

```
## R version 4.2.2 (2022-10-31 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 22621)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=Chinese (Simplified)_China.utf8
## [2] LC_CTYPE=Chinese (Simplified)_China.utf8
## [3] LC_MONETARY=Chinese (Simplified)_China.utf8
```

```
## [4] LC_NUMERIC=C
## [5] LC_TIME=Chinese (Simplified)_China.utf8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] survival_3.4-0              survminer_0.4.9
## [3] ggpubr_0.6.0                knitr_1.45
## [5] prostateCancerCamcap_1.26.0 Biobase_2.58.0
## [7] BiocGenerics_0.44.0         dplyr_1.1.3
## [9] ggplot2_3.4.4
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.11       lattice_0.20-45   tidyr_1.3.0       zoo_1.8-12
##  [5] digest_0.6.33     utf8_1.2.3        R6_2.5.1          backports_1.4.1
##  [9] evaluate_0.21     highr_0.10        pillar_1.9.0      rlang_1.1.1
## [13] rstudioapi_0.15.0 data.table_1.14.8 car_3.1-2         Matrix_1.6-1.1
## [17] rmarkdown_2.25    labeling_0.4.3    splines_4.2.2     stringr_1.5.0
## [21] munsell_0.5.0     gridtext_0.1.5    broom_1.0.5       compiler_4.2.2
## [25] xfun_0.40         pkgconfig_2.0.3   htmltools_0.5.6   ggtext_0.1.2
## [29] tidyselect_1.2.0  tibble_3.2.1      gridExtra_2.3     km.ci_0.5-6
## [33] fansi_1.0.4       withr_2.5.0       MASS_7.3-58.1     commonmark_1.9.0
## [37] grid_4.2.2        xtable_1.8-4      gtable_0.3.4      lifecycle_1.0.3
## [41] magrittr_2.0.3    KMsurv_0.1-5      scales_1.2.1      cli_3.6.1
## [45] stringi_1.7.12    carData_3.0-5     farver_2.1.1      ggsignif_0.6.4
## [49] xml2_1.3.5        survMisc_0.5.6    generics_0.1.3    vctrs_0.6.3
## [53] tools_4.2.2       glue_1.6.2        markdown_1.9      purrr_1.0.2
## [57] abind_1.4-5       fastmap_1.1.1     yaml_2.3.7        colorspace_2.1-0
## [61] rstatix_0.7.2
```