Predicting Survival of Breast Cancer Patients Who Received Hormone and/or Radiation Therapy Incoporating High-dimensional Gene Expression Information

Bei Wang

06/02/2022

1. Goal of the predictor

Accurate estimation of prognosis is essential for clinical decision making and care for breast cancer patients. Patients with same clinical characteristics (e.g., tumor stage) can respond to treatments differently and experience different survival outcomes, which might be attributable to complex variation in individual gene expression. Hormone and radiation therapy are common treatments for breast cancer and there are many patients who receive both. Recent studies raised the concern that hormonal therapy may reduce the efficacy of radiation (Cecchini et al., 2015), and it is not clear whether accurate estimation of survival can be achieved depending on patient treatment assignments. The **goal** of the current analysis is to create predictors for survival among patients who received only hormone, only radiation, or both hormone and radiation therapy.

2. Study design

The data is from the Molecular Taxonomy of Breast Cancer International Consortium (METABRIC) database and accessed from Kaggle (clink for link). The prospective data consists of 31 clinical features and 506 genetic attributes collected from almost 2000 breast cancer patients. These patients were followed up and the time (months) to either censoring or death were collected.

A part of the genetic attributes are numeric z-scores representing the number of standard deviations in mRNA expression from the mean expression in a reference population, which measures whether a gene is increased or decreased relative to tumor-free people or patients of other tumors. The rest of the genetic features are binary indicators of mutation, which is very sparse. The clinical attributes include 31 features such as cancer stage, treatment, tumor size, etc.

The structure of the data lead me to conduct a survival analysis. I also decided to drop the mutation features in the data for they are too sparse. One of the clinical variable summarized the number of mutations of each patient, so I felt this is a good enough representation of the mutation data.

3. Algorithms

Because I have time-to-event data, the algorithms that I tried are the **Cox proportional hazard regression**, **Cox proportional hazard regression with regularization (lasso, ridge, alpha=0.5)**, and **random survival forest**. I choose the C-index as my loss function for its good interpretability like AUROC.

For the regularized cox models, I plan to use cross-validation to tune the lambda value and use the lambda value that gives minimum mean cross-validated error. For the random survival forest model, I use the parameters that were suggested by Pickett and colleagues (2021).

```r
#original dataset dimension
df<- read.csv("./METABRIC_RNA_Mutation 2.csv")
dim(df)
```

```
## [1] 1904  693
```

4.Feature engenieering and exploratory data analysis

4.1 Data dimension & key variable distributions

The orginal dataset contains 1904 observations and 693 features. Among the features, there are 31 clinical variables. There are 331 mRNA level z-scores (numeric) for 331 genes and mutation (binary) for 175 genes. Data on mutation of genes are very sparse, so that I decided to just focus on clinical features and mRNA data.

My exploration indicates that 801 deaths occurred during the follow up period. The average age of the cohort if 61. Patients in this cohort receive three types of therapies and combinations of them. The data also shows that very few patients have mutation on more than 15 genes. I also found that **missing values** were only partially coded as NA. Some of them are empty cells.

```r
#binary overall survival status
table(df$overall_survival)
```

```
##
##    0    1
## 1103  801
```

```r
#age
summary(df$age_at_diagnosis)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   21.93   51.38   61.77   61.09   70.59   96.29
```

```r
#cancer type
table(df$cancer_type_detailed)
```

```
##
##
##                                       15
##                                   Breast
##                                       17
##          Breast Invasive Ductal Carcinoma
##                                     1500
##         Breast Invasive Lobular Carcinoma
##                                      142
##  Breast Invasive Mixed Mucinous Carcinoma
##                                       22
## Breast Mixed Ductal and Lobular Carcinoma
##                                      207
##                 Metaplastic Breast Cancer
##                                        1
```

```r
#treatment
table(df$chemotherapy)
```

```
##
##    0    1
## 1508  396
```

```r
table(df$hormone_therapy)
```

```
##
##    0    1
##  730 1174
```

```r
table(df$radio_therapy)
```

```
##
##    0    1
##  767 1137
```

```r
table(df$chemotherapy, df$hormone_therapy, df$radio_therapy)
```

```
## , ,  = 0
##
##
##       0   1
##   0 289 405
##   1  45  28
##
## , ,  = 1
##
##
##       0   1
##   0 228 586
##   1 168 155
```

```r
#number of mutation
table(df$mutation_count)
```

```
##
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
## 107 193 229 248 268 232 168 121  90  61  38  25  17  16  11   8   5   2   2   4
##  21  22  23  24  26  28  30  40  46  80
##   1   4   2   1   1   1   1   1   1   1
```

```r
#missing values ranked; not all are correctly represented
missing<- data.frame(sapply(df, function(x) sum(is.na(x)))) %>%
  arrange(desc(sapply.df..function.x..sum.is.na.x...))
head(missing, 10)
```

```
##                          sapply.df..function.x..sum.is.na.x...
## tumor_stage                                                501
## neoplasm_histologic_grade                                   72
## mutation_count                                              45
## tumor_size                                                  20
## patient_id                                                   0
## age_at_diagnosis                                             0
## type_of_breast_surgery                                       0
## cancer_type                                                  0
## cancer_type_detailed                                         0
## cellularity                                                  0
```

5. Data pre-processing

In my data pre-processing, I follow the steps below:

- Explore missing values and correctly code missing as NAs
- Manually drop features that will not be useful for this task
- Multiple imputation by chained equation
- Create categorical variables to stratify patients based on treatment

5.1 Explore missing values and replace missing with NAs

```
#there isn't missing values in the outcome
table(is.na(df$overall_survival))
```

```
##
## FALSE
##  1904
```

```
table(is.na(df$overall_survival_months))
```

```
##
## FALSE
##  1904
```

```
#replace empty cells with NA
df<- df %>%
  replace_with_na_all(condition= ~.x=="")
#a subset of clinical variables for viewing purpose
clinical<- clinical %>%
  replace_with_na_all(condition= ~.x=="")
```

5.2 Drop clinical features that will not be useful

```r
#clinical feature list
str(clinical)
```

```
## tibble [1,904 x 31] (S3: tbl_df/tbl/data.frame)
##  $ patient_id                    : int [1:1904] 0 2 5 6 8 10 14 22 28 35 ...
##  $ age_at_diagnosis              : num [1:1904] 75.7 43.2 48.9 47.7 77 ...
##  $ type_of_breast_surgery        : chr [1:1904] "MASTECTOMY" "BREAST CONSERVING" "MASTECTOMY" "MASTE
##  $ cancer_type                   : chr [1:1904] "Breast Cancer" "Breast Cancer" "Breast Cancer" "Brea
##  $ cancer_type_detailed          : chr [1:1904] "Breast Invasive Ductal Carcinoma" "Breast Invasive I
##  $ cellularity                   : chr [1:1904] NA "High" "High" "Moderate" ...
##  $ chemotherapy                  : int [1:1904] 0 0 1 1 1 0 1 0 0 0 ...
##  $ pam50_._claudin.low_subtype   : chr [1:1904] "claudin-low" "LumA" "LumB" "LumB" ...
##  $ cohort                        : num [1:1904] 1 1 1 1 1 1 1 1 1 1 ...
##  $ er_status_measured_by_ihc     : chr [1:1904] "Positve" "Positve" "Positve" "Positve" ...
##  $ er_status                     : chr [1:1904] "Positive" "Positive" "Positive" "Positive" ...
##  $ neoplasm_histologic_grade     : num [1:1904] 3 3 2 2 3 3 2 2 3 2 ...
##  $ her2_status_measured_by_snp6  : chr [1:1904] "NEUTRAL" "NEUTRAL" "NEUTRAL" "NEUTRAL" ...
##  $ her2_status                   : chr [1:1904] "Negative" "Negative" "Negative" "Negative" ...
##  $ tumor_other_histologic_subtype: chr [1:1904] "Ductal/NST" "Ductal/NST" "Ductal/NST" "Mixed" ...
##  $ hormone_therapy               : int [1:1904] 1 1 1 1 1 1 1 1 0 ...
##  $ inferred_menopausal_state     : chr [1:1904] "Post" "Pre" "Pre" "Pre" ...
##  $ integrative_cluster           : chr [1:1904] "4ER+" "4ER+" "3" "9" ...
##  $ primary_tumor_laterality      : chr [1:1904] "Right" "Right" "Right" "Right" ...
##  $ lymph_nodes_examined_positive : num [1:1904] 10 0 1 3 8 0 1 1 1 0 ...
##  $ mutation_count                : num [1:1904] NA 2 2 1 2 4 4 1 4 5 ...
##  $ nottingham_prognostic_index   : num [1:1904] 6.04 4.02 4.03 4.05 6.08 ...
##  $ oncotree_code                 : chr [1:1904] "IDC" "IDC" "IDC" "MDLC" ...
##  $ overall_survival_months       : num [1:1904] 140.5 84.6 163.7 164.9 41.4 ...
##  $ overall_survival              : int [1:1904] 1 1 0 1 0 0 1 0 0 0 ...
##  $ pr_status                     : chr [1:1904] "Negative" "Positive" "Positive" "Positive" ...
##  $ radio_therapy                 : int [1:1904] 1 1 0 1 1 1 1 1 1 0 ...
##  $ X3.gene_classifier_subtype    : chr [1:1904] "ER-/HER2-" "ER+/HER2- High Prolif" NA NA ...
##  $ tumor_size                    : num [1:1904] 22 10 15 25 40 31 10 29 16 28 ...
##  $ tumor_stage                   : num [1:1904] 2 1 2 2 2 4 2 2 2 2 ...
##  $ death_from_cancer             : chr [1:1904] "Living" "Living" "Died of Disease" "Living" ...
```

```r
#drop redundant, not useful, outcome features..
drop<-c("cancer_type", "cohort","patient_id", "er_status_measured_by_ihc", "her2_status_measured_by_snp
outcome<- df[, colnames(df) %in% c("overall_survival", "overall_survival_months")]
df_select<-df[, !colnames(df) %in% drop]

#drop mutations in genes because too sparse
df_select<- df_select %>%
  select(-ends_with("_mut"))

#dims after dropping
dim(df_select)
```

```
## [1] 1904  511
```

```
dim(outcome)
```

```
## [1] 1904    2
```

5.3 Multiple imupatation

Now that all missing are correctly coded, I first rank the features by numbers of missingness. I can see that the missing values are all on the clinical features. I use multiple imputation with chained equations depending on the type of the features. The data table below looks like there is no missingness.

The imputation process is muted here because I saved the imputed dataset to save time. But the code is shown here.

```
#rank by missing #
missing<- data.frame(sapply(df_select, function(x) sum(is.na(x)))) %>%
  arrange(desc(sapply.df_select..function.x..sum.is.na.x...))
head(missing, 15)
```

```
##                                sapply.df_select..function.x..sum.is.na.x...
## tumor_stage                                                             501
## X3.gene_classifier_subtype                                              204
## primary_tumor_laterality                                                106
## neoplasm_histologic_grade                                                72
## cellularity                                                              54
## mutation_count                                                           45
## type_of_breast_surgery                                                   22
## tumor_size                                                               20
## cancer_type_detailed                                                     15
## tumor_other_histologic_subtype                                           15
## oncotree_code                                                            15
## age_at_diagnosis                                                          0
## chemotherapy                                                              0
## pam50_._claudin.low_subtype                                               0
## er_status                                                                 0
```

```
#convert all character features to factor for imputation
df_select[sapply(df_select, is.character)] <- lapply(df_select[sapply(df_select, is.character)],
                                                     as.factor)
#imputation (excluding the outcome features)
set.seed(777)
imp<- mice(df_select[, !colnames(df_select) %in% c("overall_survival_months","overall_survival")], m=3,
df_select_imp<- complete(imp)

#add outcome back
df_select_imp<- data.frame(df_select_imp, outcome)
```

```
#save this imputed data so that I don't have to run again
save(df_select_imp, file="df_select_imp.rda")
```

```
load("./df_select_imp.rda")
#check missing after imputation (ranked)
```

```r
missing<- data.frame(sapply(df_select_imp, function(x) sum(is.na(x)))) %>%
  arrange(desc(sapply.df_select_imp..function.x..sum.is.na.x...))
head(missing, 10)
```

```
##                                 sapply.df_select_imp..function.x..sum.is.na.x...
## age_at_diagnosis                                                               0
## type_of_breast_surgery                                                         0
## cancer_type_detailed                                                           0
## cellularity                                                                    0
## chemotherapy                                                                   0
## pam50_._claudin.low_subtype                                                    0
## er_status                                                                      0
## neoplasm_histologic_grade                                                      0
## her2_status                                                                    0
## tumor_other_histologic_subtype                                                 0
```

5.4 Feature selection based on correlation

I dropped one of a pair of features if the pair correlation is over 0.7. I ended up with 497 features now.

```r
#correlation matrix of numeric features
num_cols<-select_if(df_select_imp, is.numeric)
cor<- melt(round(x= cor(num_cols), digits = 2)) %>%
  arrange(desc(abs(value))) %>%
  filter(Var1!=Var2)

#remove |r|>0.7
big_cor<- cor %>%
  filter(abs(value)>0.7)
drop_r<- as.vector(big_cor[,1]) %>%
  unique()
#drop these features
df_select_imp<- df_select_imp[, !colnames(df_select_imp)%in% drop_r]
dim(df_select_imp)
```

```
## [1] 1904  497
```

5.5 Stratify based on treatment

There three main treatment types in this data, which are chemo, radiation, and hormone. Many patients have more than 1 treatment. Corresponding to my stated purpose, I decide to limit my data to patients who receive hormone and/or radiation therapy. I ended up with **1219 patients and 495** features. These patients are further stratified into hormone only, radiation only, and both for the purpose of creating different predictors.

```r
# chemotherapy, hormone_therapy, radio_therapy
df_select_imp<- df_select_imp %>%
  mutate(treatment= case_when(chemotherapy==1&hormone_therapy==1&radio_therapy==1 ~"all",
                              chemotherapy==1&hormone_therapy==1&radio_therapy==0 ~"che_hor",
                              chemotherapy==1&hormone_therapy==0&radio_therapy==1 ~"che_rad",
                              chemotherapy==1&hormone_therapy==0&radio_therapy==0 ~"che",
```

```
                                  chemotherapy==0&hormone_therapy==1&radio_therapy==1 ~"hor_rad",
                                  chemotherapy==0&hormone_therapy==0&radio_therapy==1 ~"rad",
                                  chemotherapy==0&hormone_therapy==1&radio_therapy==0 ~ "hor",
                                  chemotherapy==0&hormone_therapy==0&radio_therapy==0 ~"none"
                                  )) %>%
  filter(treatment %in% c("hor", "hor_rad", "rad")) %>%
  mutate(treatment= factor(treatment, levels= c("hor_rad", "hor", "rad"))) %>%
  select(-chemotherapy, -hormone_therapy, -radio_therapy) %>%
  rename(time= overall_survival_months, status= overall_survival)

dim(df_select_imp)
```

```
## [1] 1219  495
```

```
table(df_select_imp$treatment)
```

```
##
## hor_rad     hor     rad
##     586     405     228
```

```
#drop "chemotherapy", "hormone_therapy", "radio_therapy"))
```

6. Model training and evaluation

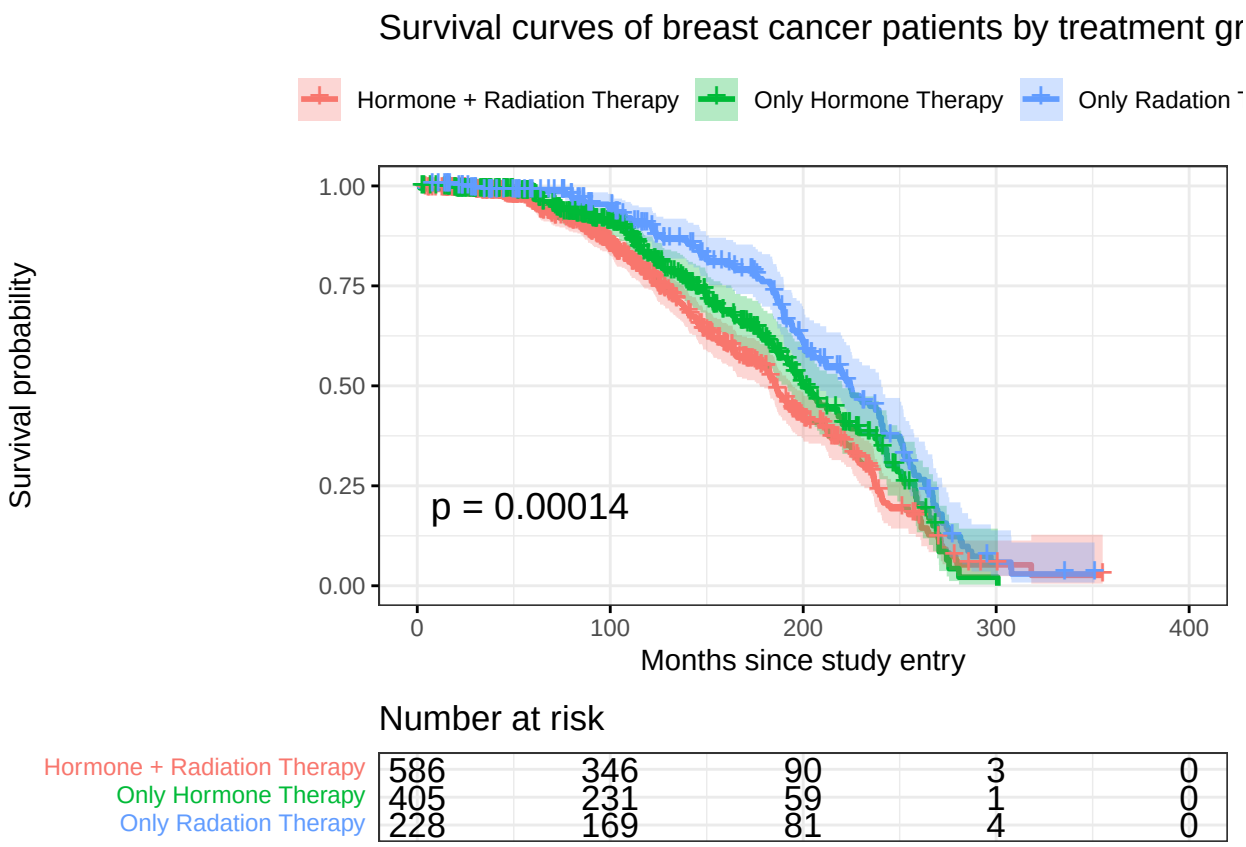
I generally followed the following steps:

- Explore survival curves of patients who had different treatment
- Fit different models using the afromentioned algorithms and parameters tuning for patients of each of the 3 types of treatment
- Evaluate the 10-fold cross-validated C-index
- Repeat the last two steps using the full data set of all patients who have had hormone and/or radiation therapy
- Compare performance

6.1 Survival curves of patients who received different treatment

We can see that the unadjusted survival curves are significantly different among patients who receive different treatments. Echoing the concern about reduced efficacy of using both hormone and radiation therapy, we do see the worst survival for patients who receive both treatment. But this is likely confounded by other factors such as tumor stage and type.

```
cox1<- survfit(Surv(time, status)~ treatment, data=df_select_imp)
ggsurvplot(cox1, data= df_select_imp,
           title= "Survival curves of breast cancer patients by treatment group",
           legend.title="",
           xlab= "Months since study entry",
           legend.labs=c("Hormone + Radiation Therapy", "Only Hormone Therapy", "Only Radation Therapy"),
           pval=T, conf.int = T, risk.table = T, tables.height = 0.2, tables.theme = theme_cleantable()
          )
```




```
#subset data
hor_rad<- df_select_imp %>%
  filter(treatment=="hor_rad") %>%
  select(-treatment)
hor<- df_select_imp %>%
  filter(treatment=="hor") %>%
  select(-treatment)
rad<- df_select_imp %>%
  filter(treatment=="rad") %>%
  select(-treatment)
```

6.2 Predictors for patients who received hormone and radiation therapy

```
set.seed(777)
N<- nrow(hor_rad)
V<- 10
folds<- split(1:N, rep(1:V, length=N))
eval1<- matrix(NA, nrow=V, ncol = 5)
colnames(eval1)<-c("coxph", "lasso", "ridge", "el", "rf")

#coxph
for (v in 1:V){
  train<- hor_rad[-folds[[v]], ]
  test<- hor_rad[folds[[v]], ]
  fit_cv_cph<- coxph(Surv(time, status)~., data= train)
```

```
  pred_cph<- predict(fit_cv_cph,
                     type= "lp",
                     newdata= test)
  eval1[v,1]<- concordance(Surv(time, status)~ pred_cph, data= test, reverse = TRUE)$concordance
}
```

```
#lasso
Y<- data.matrix(hor_rad[c("time", "status")])
X<- data.matrix(hor_rad[, !colnames(hor_rad)%in% c("time", "status")])

for (v in 1:V){
  train_x<- X[-folds[[v]], ]
  test_x<- X[folds[[v]], ]
  train_y<- Y[-folds[[v]], ]
  test_y<- Y[folds[[v]], ]
  fit_cv_lasso<- cv.glmnet(y= train_y,
                           x= train_x,
                           family= "cox",
                           alpha=1)
  pred_lasso<- predict(fit_cv_lasso$glmnet.fit,
                       newx= test_x,
                       s= fit_cv_lasso$lambda.min,
                       type="link")
  df_test<- data.frame(test_y, test_x)
  eval1[v,2]<- concordance(Surv(time, status)~ pred_lasso,
                           data= df_test, reverse = TRUE)$concordance
}

#ridge
for (v in 1:V){
  train_x<- X[-folds[[v]], ]
  test_x<- X[folds[[v]], ]
  train_y<- Y[-folds[[v]], ]
  test_y<- Y[folds[[v]], ]
  fit_cv_lasso<- cv.glmnet(y= train_y,
                           x= train_x,
                           family= "cox",
                           alpha=0)
  pred_lasso<- predict(fit_cv_lasso$glmnet.fit,
                       newx= test_x,
                       s= fit_cv_lasso$lambda.min,
                       type="link")
  df_test<- data.frame(test_y, test_x)
  eval1[v,3]<- concordance(Surv(time, status)~ pred_lasso,
                           data= df_test, reverse = TRUE)$concordance
}

#el
for (v in 1:V){
  train_x<- X[-folds[[v]], ]
  test_x<- X[folds[[v]], ]
  train_y<- Y[-folds[[v]], ]
  test_y<- Y[folds[[v]], ]
```

```
  fit_cv_lasso<- cv.glmnet(y= train_y,
                           x= train_x,
                           family= "cox",
                           alpha=0.5)
  pred_lasso<- predict(fit_cv_lasso$glmnet.fit,
                       newx= test_x,
                       s= fit_cv_lasso$lambda.min,
                       type="link")
  df_test<- data.frame(test_y, test_x)
  eval1[v,4]<- concordance(Surv(time, status)~ pred_lasso,
                           data= df_test, reverse = TRUE)$concordance
}
```

```
#rf
for (v in 1:V){
  train<- hor_rad[-folds[[v]], ]
  test<- hor_rad[folds[[v]], ]
  fit_cv_rf<- rfsrc(Surv(time, status)~., data= train,
                    ntree=500,
                    samptype = "swr")#default for other pars, see reference
  eval1[v,5]<- 1- tail(predict.rfsrc(fit_cv_rf, newdata = test)$err.rate,1)
}
```

6.3 Predictors for patients who received only hormone therapy

```
set.seed(555)

N<- nrow(hor)
V<- 10
folds<- split(1:N, rep(1:V, length=N))
eval2<- matrix(NA, nrow=V, ncol = 5)
colnames(eval2)<-c("coxph", "lasso", "ridge", "el", "rf")

#coxph
for (v in 1:V){
  train<- hor[-folds[[v]], ]
  test<- hor[folds[[v]], ]
  fit_cv_cph<- coxph(Surv(time, status)~., data= train)
  pred_cph<- predict(fit_cv_cph,
                     type= "lp",
                     newdata= test)
  eval2[v,1]<- concordance(Surv(time, status)~ pred_cph, data= test, reverse = TRUE)$concordance
}
```

```
#lasso
Y<- data.matrix(hor[c("time", "status")])
X<- data.matrix(hor[, !colnames(hor)%in% c("time", "status")])

for (v in 1:V){
  train_x<- X[-folds[[v]], ]
  test_x<- X[folds[[v]], ]
```

```
  train_y<- Y[-folds[[v]], ]
  test_y<- Y[folds[[v]], ]
  fit_cv_lasso<- cv.glmnet(y= train_y,
                           x= train_x,
                           family= "cox",
                           alpha=1)
  pred_lasso<- predict(fit_cv_lasso$glmnet.fit,
                       newx= test_x,
                       s= fit_cv_lasso$lambda.min,
                       type="link")
  df_test<- data.frame(test_y, test_x)
  eval2[v,2]<- concordance(Surv(time, status)~ pred_lasso,
                           data= df_test, reverse = TRUE)$concordance
}

#ridge
for (v in 1:V){
  train_x<- X[-folds[[v]], ]
  test_x<- X[folds[[v]], ]
  train_y<- Y[-folds[[v]], ]
  test_y<- Y[folds[[v]], ]
  fit_cv_lasso<- cv.glmnet(y= train_y,
                           x= train_x,
                           family= "cox",
                           alpha=0)
  pred_lasso<- predict(fit_cv_lasso$glmnet.fit,
                       newx= test_x,
                       s= fit_cv_lasso$lambda.min,
                       type="link")
  df_test<- data.frame(test_y, test_x)
  eval2[v,3]<- concordance(Surv(time, status)~ pred_lasso,
                           data= df_test, reverse = TRUE)$concordance
}

#el
for (v in 1:V){
  train_x<- X[-folds[[v]], ]
  test_x<- X[folds[[v]], ]
  train_y<- Y[-folds[[v]], ]
  test_y<- Y[folds[[v]], ]
  fit_cv_lasso<- cv.glmnet(y= train_y,
                           x= train_x,
                           family= "cox",
                           alpha=0.5)
  pred_lasso<- predict(fit_cv_lasso$glmnet.fit,
                       newx= test_x,
                       s= fit_cv_lasso$lambda.min,
                       type="link")
  df_test<- data.frame(test_y, test_x)
  eval2[v,4]<- concordance(Surv(time, status)~ pred_lasso,
                           data= df_test, reverse = TRUE)$concordance
}
```

```
#rf
for (v in 1:V){
  train<- hor[-folds[[v]], ]
  test<- hor[folds[[v]], ]
  fit_cv_rf<- rfsrc(Surv(time, status)~., data= train,
                    ntree=500,
                    samptype = "swr")#default for other pars, see reference
  eval2[v,5]<- 1- tail(predict.rfsrc(fit_cv_rf, newdata = test)$err.rate,1)
}
```

6.4 Predictors for patients who received only radiation therapy

```
set.seed(333)
N<- nrow(rad)
V<- 10
folds<- split(1:N, rep(1:V, length=N))
eval3<- matrix(NA, nrow=V, ncol = 5)
colnames(eval3)<-c("coxph", "lasso", "ridge", "el", "rf")

#coxph
for (v in 1:V){
  train<- rad[-folds[[v]], ]
  test<- rad[folds[[v]], ]
  fit_cv_cph<- coxph(Surv(time, status)~., data= train)
  pred_cph<- predict(fit_cv_cph,
                     type= "lp",
                     newdata= test)
  eval3[v,1]<- concordance(Surv(time, status)~ pred_cph, data= test, reverse = TRUE)$concordance
}
```

```
#lasso
Y<- data.matrix(rad[c("time", "status")])
X<- data.matrix(rad[, !colnames(rad)%in% c("time", "status")])

for (v in 1:V){
  train_x<- X[-folds[[v]], ]
  test_x<- X[folds[[v]], ]
  train_y<- Y[-folds[[v]], ]
  test_y<- Y[folds[[v]], ]
  fit_cv_lasso<- cv.glmnet(y= train_y,
                           x= train_x,
                           family= "cox",
                           alpha=1)
  pred_lasso<- predict(fit_cv_lasso$glmnet.fit,
                       newx= test_x,
                       s= fit_cv_lasso$lambda.min,
                       type="link")
  df_test<- data.frame(test_y, test_x)
  eval3[v,2]<- concordance(Surv(time, status)~ pred_lasso,
                           data= df_test, reverse = TRUE)$concordance
}
```

```r
#ridge

for (v in 1:V){
  train_x<- X[-folds[[v]], ]
  test_x<- X[folds[[v]], ]
  train_y<- Y[-folds[[v]], ]
  test_y<- Y[folds[[v]], ]
  fit_cv_lasso<- cv.glmnet(y= train_y,
                           x= train_x,
                           family= "cox",
                           alpha=0)
  pred_lasso<- predict(fit_cv_lasso$glmnet.fit,
                       newx= test_x,
                       s= fit_cv_lasso$lambda.min,
                       type="link")
  df_test<- data.frame(test_y, test_x)
  eval3[v,3]<- concordance(Surv(time, status)~ pred_lasso,
                           data= df_test, reverse = TRUE)$concordance
}

#el

for (v in 1:V){
  train_x<- X[-folds[[v]], ]
  test_x<- X[folds[[v]], ]
  train_y<- Y[-folds[[v]], ]
  test_y<- Y[folds[[v]], ]
  fit_cv_lasso<- cv.glmnet(y= train_y,
                           x= train_x,
                           family= "cox",
                           alpha=0.5)
  pred_lasso<- predict(fit_cv_lasso$glmnet.fit,
                       newx= test_x,
                       s= fit_cv_lasso$lambda.min,
                       type="link")
  df_test<- data.frame(test_y, test_x)
  eval3[v,4]<- concordance(Surv(time, status)~ pred_lasso,
                           data= df_test, reverse = TRUE)$concordance
}

#rf
for (v in 1:V){
  train<- rad[-folds[[v]], ]
  test<- rad[folds[[v]], ]
  fit_cv_rf<- rfsrc(Surv(time, status)~., data= train,
                    ntree=500,
                    samptype = "swr")#default for other pars, see reference
  eval3[v,5]<- 1- tail(predict.rfsrc(fit_cv_rf, newdata = test)$err.rate,1)
}
```

6.5 Predictors for all patients

```
set.seed(111)
N<- nrow(df_select_imp)
V<- 10
folds<- split(1:N, rep(1:V, length=N))
eval_all<- matrix(NA, nrow=V, ncol = 5)
colnames(eval_all)<-c("coxph", "lasso", "ridge", "el", "rf")

#coxph
for (v in 1:V){
  train<- df_select_imp[-folds[[v]], ]
  test<- df_select_imp[folds[[v]], ]
  fit_cv_cph<- coxph(Surv(time, status)~., data= train)
  pred_cph<- predict(fit_cv_cph,
                     type= "lp",
                     newdata= test)
  eval_all[v,1]<- concordance(Surv(time, status)~ pred_cph, data= test, reverse = TRUE)$concordance
}
```

```
Y<- data.matrix(df_select_imp[c("time", "status")])
X<- data.matrix(df_select_imp[, !colnames(df_select_imp)%in% c("time", "status")])
#lasso
for (v in 1:V){
  train_x<- X[-folds[[v]], ]
  test_x<- X[folds[[v]], ]
  train_y<- Y[-folds[[v]], ]
  test_y<- Y[folds[[v]], ]
  fit_cv_lasso<- cv.glmnet(y= train_y,
                           x= train_x,
                           family= "cox",
                           alpha=1)
  pred_lasso<- predict(fit_cv_lasso$glmnet.fit,
                       newx= test_x,
                       s= fit_cv_lasso$lambda.min,
                       type="link")
  df_test<- data.frame(test_y, test_x)
  eval_all[v,2]<- concordance(Surv(time, status)~ pred_lasso,
                          data= df_test, reverse = TRUE)$concordance
}

#ridge
for (v in 1:V){
  train_x<- X[-folds[[v]], ]
  test_x<- X[folds[[v]], ]
  train_y<- Y[-folds[[v]], ]
  test_y<- Y[folds[[v]], ]
  fit_cv_lasso<- cv.glmnet(y= train_y,
                           x= train_x,
                           family= "cox",
                           alpha=0)
  pred_lasso<- predict(fit_cv_lasso$glmnet.fit,
                       newx= test_x,
```

```
                        s= fit_cv_lasso$lambda.min,
                        type="link")
  df_test<- data.frame(test_y, test_x)
  eval_all[v,3]<- concordance(Surv(time, status)~ pred_lasso,
                           data= df_test, reverse = TRUE)$concordance
}

#el
for (v in 1:V){
  train_x<- X[-folds[[v]], ]
  test_x<- X[folds[[v]], ]
  train_y<- Y[-folds[[v]], ]
  test_y<- Y[folds[[v]], ]
  fit_cv_lasso<- cv.glmnet(y= train_y,
                           x= train_x,
                           family= "cox",
                           alpha=0.5)
  pred_lasso<- predict(fit_cv_lasso$glmnet.fit,
                        newx= test_x,
                        s= fit_cv_lasso$lambda.min,
                        type="link")
  df_test<- data.frame(test_y, test_x)
  eval_all[v,4]<- concordance(Surv(time, status)~ pred_lasso,
                           data= df_test, reverse = TRUE)$concordance
}
```

```
#rf
for (v in 1:V){
  train<- df_select_imp[-folds[[v]], ]
  test<- df_select_imp[folds[[v]], ]
  fit_cv_rf<- rfsrc(Surv(time, status)~., data= train,
                    ntree=500,
                    samptype = "swr")#default for other pars, see reference
  eval_all[v,5]<- 1- tail(predict.rfsrc(fit_cv_rf, newdata = test)$err.rate,1)
}
```

6.6 Summary of cross-validated performance (C-index)

Based on the results, ridge predictor had the best performance in the predictor for patients who had hormone and radiation therapy. Lasso predictor had the best performance in the predictors for patients who only had hormone therapy. Random survival forest had the best performance for the predictor for patients who only had radiation therapy and overall patients.

The overall performance is not that great, which might related to the fact that there are too many features vs. patients number. The performance using regularized regression and random survival forest are also close to each other. They all outperform the traditional cox proportional regression.

```
#patients had both hormone and radiation therapy
summary(eval1)
```

```
##      coxph            lasso            ridge              el
##  Min.   :0.3944   Min.   :0.5775   Min.   :0.6257   Min.   :0.5835
```

```
##  1st Qu.:0.4543   1st Qu.:0.6051   1st Qu.:0.6528   1st Qu.:0.6079
##  Median :0.5094   Median :0.6706   Median :0.6686   Median :0.6663
##  Mean   :0.4984   Mean   :0.6602   Mean   :0.6901   Mean   :0.6613
##  3rd Qu.:0.5411   3rd Qu.:0.6993   3rd Qu.:0.7317   3rd Qu.:0.6956
##  Max.   :0.5924   Max.   :0.7796   Max.   :0.7734   Max.   :0.7692
##        rf
##  Min.   :0.6086
##  1st Qu.:0.6524
##  Median :0.6785
##  Mean   :0.6706
##  3rd Qu.:0.6949
##  Max.   :0.7254
```

```
#patients only had hormone therapy
summary(eval2)
```

```
##      coxph            lasso            ridge              el
##  Min.   :0.3218   Min.   :0.5123   Min.   :0.4632   Min.   :0.5123
##  1st Qu.:0.4532   1st Qu.:0.6189   1st Qu.:0.6004   1st Qu.:0.6161
##  Median :0.5064   Median :0.6493   Median :0.6243   Median :0.6402
##  Mean   :0.5190   Mean   :0.6681   Mean   :0.6388   Mean   :0.6673
##  3rd Qu.:0.5841   3rd Qu.:0.7176   3rd Qu.:0.6990   3rd Qu.:0.7246
##  Max.   :0.7912   Max.   :0.8411   Max.   :0.7815   Max.   :0.8344
##        rf
##  Min.   :0.4840
##  1st Qu.:0.5509
##  Median :0.6628
##  Mean   :0.6462
##  3rd Qu.:0.7431
##  Max.   :0.7815
```

```
#patients only had radiation therapy
summary(eval3)
```

```
##      coxph            lasso            ridge              el
##  Min.   :0.3919   Min.   :0.4757   Min.   :0.4757   Min.   :0.4583
##  1st Qu.:0.4420   1st Qu.:0.5479   1st Qu.:0.5442   1st Qu.:0.5331
##  Median :0.5137   Median :0.5855   Median :0.5916   Median :0.5969
##  Mean   :0.5189   Mean   :0.5832   Mean   :0.6033   Mean   :0.5799
##  3rd Qu.:0.5756   3rd Qu.:0.6252   3rd Qu.:0.6514   3rd Qu.:0.6321
##  Max.   :0.7000   Max.   :0.6807   Max.   :0.7582   Max.   :0.6723
##        rf
##  Min.   :0.4567
##  1st Qu.:0.5336
##  Median :0.6236
##  Mean   :0.6145
##  3rd Qu.:0.6748
##  Max.   :0.7692
```

```
#among all patients
summary(eval_all)
```

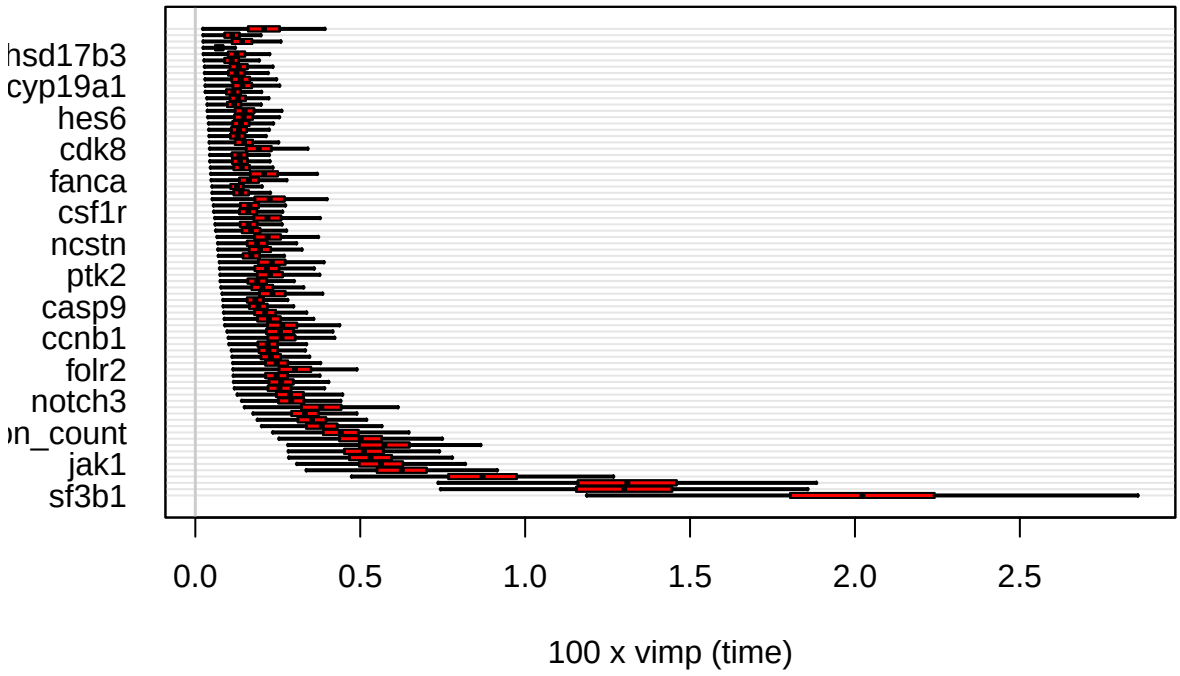
```
##      coxph            lasso            ridge              el
##  Min.   :0.4739   Min.   :0.5844   Min.   :0.5910   Min.   :0.5838
##  1st Qu.:0.5546   1st Qu.:0.6497   1st Qu.:0.6543   1st Qu.:0.6500
##  Median :0.5831   Median :0.6827   Median :0.6817   Median :0.6793
##  Mean   :0.5788   Mean   :0.6812   Mean   :0.6850   Mean   :0.6812
##  3rd Qu.:0.6144   3rd Qu.:0.7127   3rd Qu.:0.7086   3rd Qu.:0.7150
##  Max.   :0.6788   Max.   :0.7608   Max.   :0.7602   Max.   :0.7527
##        rf
##  Min.   :0.5932
##  1st Qu.:0.6628
##  Median :0.6817
##  Mean   :0.6868
##  3rd Qu.:0.7233
##  Max.   :0.7683
```

6.7 Feature importance plot of the random survival forest predictor among all patients

```
set.seed(500)
rf_1<- rfsrc(Surv(time, status)~., data= df_select_imp, ntree=500,
                  samptype = "swr", importance=TRUE)
jk.obj<- subsample(rf_1)#default B=100 #of bootstrap
```

```
##   |                                                                      |
```

```
plot(jk.obj, xlab = "Variable Importance*100")
```



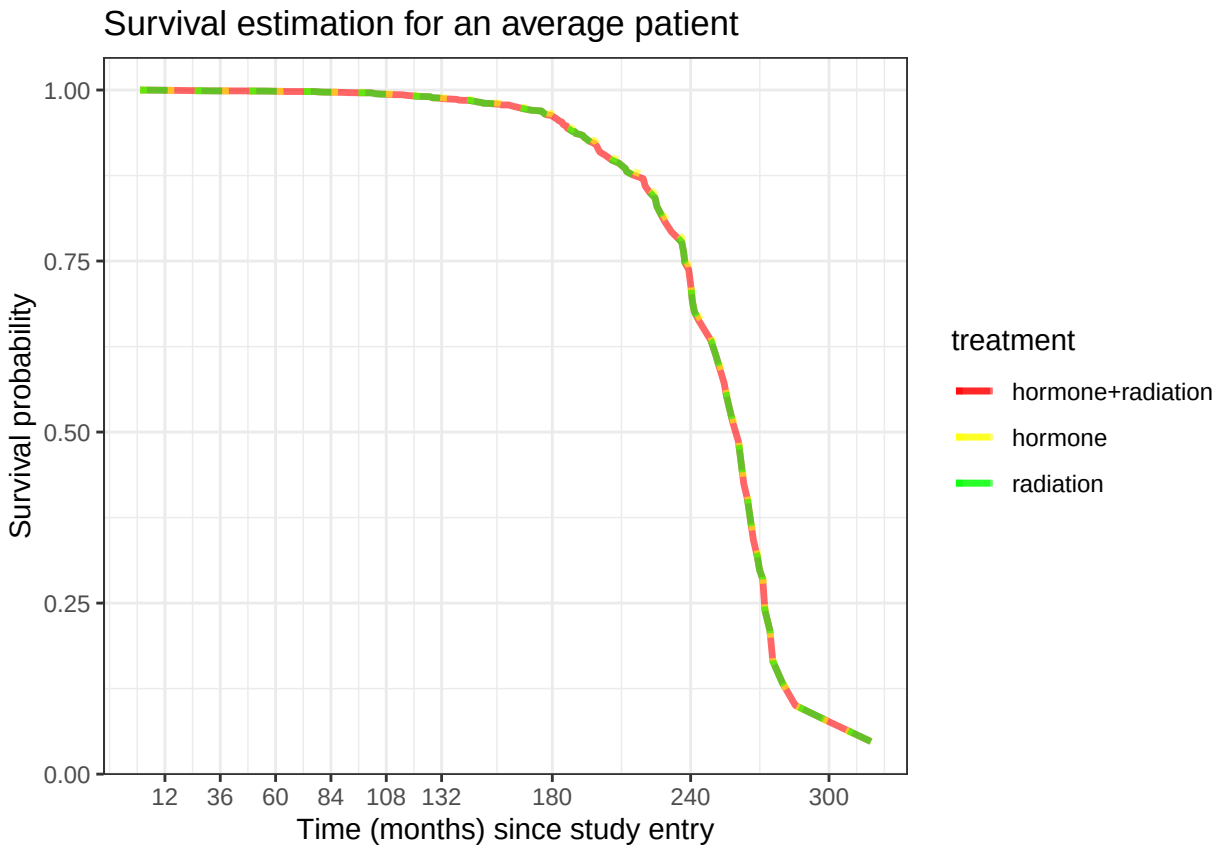
6.8 Plot the estimated survival function for a simulated average patient

I use the random forest predictor created among all patients to estimate survival of 3 new simulated patients. Their numeric features were extracted from the medians in the population and categorical features were extracted using the most frequent value in the population. The 3 patients are identical except for their treatment assignments. Their estimated survival probability were plotted over time below. The plot shows that the survival estimate for the 3 patients are almost identical.

```
#create 3 patients data
num_cols<-select_if(rf_1$xvar, is.numeric)
fac_cols<- select_if(rf_1$xvar, is.factor)
new_num<- data.frame(lapply(1:ncol(num_cols), function(i){
  median(num_cols[,i])
  }))
colnames(new_num)<- colnames(num_cols)
new_fac<- data.frame(lapply(1:ncol(fac_cols), function(i){
  which.max(table(fac_cols[,i]))
    }))
colnames(new_fac)<- colnames(fac_cols)
new<- cbind(new_num, new_fac)
new1<- new2<- new3<- new

#the three individuals only vary in terms of treatment
new1[,which(rf_1$xvar.names=="treatment")]<- "hor_rad"
new2[,which(rf_1$xvar.names=="treatment")]<- "hor"
new3[,which(rf_1$xvar.names=="treatment")]<- "rad"
new_df<- rbind(new1, new2, new3)
y.pred<- predict(rf_1, newdata = new_df)
```

```
#plot survival estimates for the 3 patients
plot_df<- data.frame(time= as.vector(y.pred$time.interest),
           new1=as.vector(y.pred$survival[1,]),
           new2=as.vector(y.pred$survival[2,]),
           new3=as.vector(y.pred$survival[1,]))
plot_df %>%
  ggplot(aes(x=time))+
  geom_line(aes(y=new1,colour= "hormone+radiation"), alpha=0.6,size=1.2)+
  geom_line(aes(y=new2, colour="hormone"), alpha=0.6, linetype="dotted", size=1.2)+
  geom_line(aes(y=new3, colour="radiation"),alpha=0.6, linetype="dashed", size=1.2)+
  xlab("Time (months) since study entry")+
  ylab("Survival probability")+
  ggtitle("Survival estimation for an average patient")+
  scale_x_continuous(breaks = c(12, 36, 60, 84, 108, 132, 180, 240, 300))+
  scale_color_manual(name="treatment",
                     breaks=c("hormone+radiation", "hormone", "radiation"),
                     values = c("red", "yellow", "green"))+
  theme_bw()
```



7. Conclusion

To summarize, I created predictors for survival among breast cancer patients who received different combos of hormone and radiation therapy. Because of the high dimension of the data that involves hundreds of gene expression features, I prioritize the predictive ability over interpretability (e.g., risk factors). The performance of these predictors are averagely in the range of 0.65-0.7, which is not excellent. However, the regularized cox hazard proportional models and the random survival forest all outperformed the traditional cox proportional model by a lot, indicating of the potential of more accurate predictors with more observations.

A major limitation of the study is the small number of observation relative to the number of features. Another limitation is that the time-to-event measurement seems do not start from a coherent time point (e.g.,time of diagnosis of breast cancer), which makes it harder to interpret the results in terms of how to apply it to practice.

Because of limited data, this initial development needs further **implementation** steps to improve accuracy and reliability. One of the steps is to make sure that new clinical, mRNA and mutation data are measured in a standard way compared to the current data collection plan. As new patient data increase, it is ideal to expand this to patients who receive other types of treatment and combos (e.g., chemotherapy). We also need to constantly monitor the patterns of key features, missing values, and predictive performance over time. Specifically, tumor stage is a feature that has a lot of missing values. Further steps should takend to examine why this information is missing and how to improve.

For the purpose of the predictors are to assist clinical decision making, I expect the audience to be practitioners in clinical settings or patients themselves. So, I think visualizing patients estimation of survival probability over time like I did above would be useful. I also think the eventual predictor should be able to automatically receive lab data (e.g., mRNA and mutation) for patient. It should also be an interactive tool

where people can input and change fields if needed. However, it needs to be experimented whether imperfect predictions of one's lifespan would be beneficial or harmful overall.