Predicting Lung Cancer Survival 2022

Courtney Van Den Elzen

Jan 30th, 2023

Libraries

```
# data cleaning and manipulation tools
library(tidyverse)

# machine learning tools
library(caret)

# Survival analysis
library(tidymodels)
library(censored)
library(glmnet)
```

Data

```
# Clinical data:
# (1) ID - patient identifier [1-190]
# (2) Outcome - alive/dead ["Alive"/"Dead"]
# (3) Survival.months - followup time (length of time at which outcome was assessed) [9.0-71.0]
# (4) Age - patient age at diagnosis in years [56-84]
# (5) Grade - tumor grade [1-4 or missing (9 = missing)]
# (6) Num.Primaries - number of primary tumors
# (7) T - tumor stage (I ASSUMED THIS WAS THE FINAL STAGE)
# (8) N - number of metastasis to lymph nodes
# (9) M - number of distance metastases
# (10) Radiation - whether the patient had radiation (5 = yes, 0 = no)
# (11) Stage - Stage at diagnosis
# (12) Primary.Site - Location of primary tumor
# (13) Histology - type of tumor (microscopic structure)
# (14) Tumor.Size - size at diagnosis
# (15) Num.Mutated.Genes - number of genetic mutations found in tumor
# (16) Num.Mutations - number of genes with mutation
clindat <- read_csv("./clinical.csv")
```

```
## Rows: 190 Columns: 16-- Column specification --------------------------------------------------------------
## Delimiter: ","
## chr (8): Outcome, T, N, M, Stage, Primary.Site, Histology, Tumor.Size
## dbl (8): ID, Survival.Months, Age, Grade, Num.Primaries, Radiation, Num.Muta...
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# (1) ID - Patient ID
# (2) Gene - Name of mutated gene
genodat <- read_csv("./genomics.csv")
```

```
## Rows: 510 Columns: 2-- Column specification -----------------------------------------------------------
## Delimiter: ","
## chr (1): Gene
## dbl (1): ID
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

1. Please walk us through how you cleaned up this dataset. Do you see any interesting trends?

For these datasets, I cleaned and reorganized the data using several steps. Firstly, I loaded both the clinical and genomics data into R and familiarized myself with the columns of both data frames, including their biological interpretation and expected values and types.

For the clinical data, I made several changes. Firstly, there were a couple cases where the biological interpretation was ambiguous, so I made a couple of implicit assumptions: I assumed that the 'T' column was the tumor stage at last observation since the 'Stage' column contained the tumor stage at diagnosis. I also assumed that a value of "5" in 'Radiation' meant that the patient had gotten radiation treatment. Next, I assessed the data type that was automatically assigned to each column by R and changed it if necessary. The most common change was from numeric to factor. I also renamed the T column to 'Tumor.Stage' because T is a special character in R. I then looked at the unique values for each category and checked for typos and unusual values. I found a typo in the 'Histology' column which I corrected, and I changed the value "9" in the 'Grade' column to NA since the tumor grade value range was supposed to be 1-4. I changed All "NULL" values in the dataset to "NA" so R could interpret the missing data correctly. I also recoded the 'Stage' column to match those in in the 'Tumor.Stage' column. Finally, I recoded the 'Outcome' column to 0/1 dummy variable coding for compatibility with survival models.

For the genomics data, I first checked for typos and changed the datatypes to factors. I then re-worked the dimensions of the dataset using pivot_wider() such that each column was a dummy variable encoding for presence of a mutation in a specific gene and each row was one patient. Since not every patient had a mutation, there were a few missing patient IDs. I used full_join() to add rows for these patients to the genomics data. I was then able to fill all NAs in as 0s for proper dummy coding. I then used full_join to join the clinical and genomics datasets so that both the gene mutation data and the other clinical variables could be used as features in the models.

Clean Data: Clinical

```r
clindat_clean <-

     clindat %>%

     # correct data types
     dplyr::mutate(ID = as.factor(ID),
                   Grade = as.integer(Grade),
                   Survival.Months = as.integer(Survival.Months),
                   Radiation = as.factor(Radiation),
                   Num.Primaries = as.factor(Num.Primaries),
                   Num.Mutated.Genes = as.integer(Num.Mutated.Genes),
                   Num.Mutations = as.integer(Num.Mutations)) %>%

     # rename to avoid built-in function incompatibilities
     dplyr::rename(Tumor.Stage = "T") %>%

     # clean data, add NAs where needed, change coding for clarity
     dplyr::mutate(Grade = na_if(Grade, 9),
                   Tumor.Stage = as.factor(na_if(Tumor.Stage, "UNK")),
                   N = as.numeric(na_if(N, "NULL")),
                   M = as.numeric(na_if(M, "NULL")),

                   # assuming that 0 means "no" here
                   Radiation =
                        as.factor(
                             case_when(Radiation == 0 ~ "no",
                                       Radiation == 5 ~ "yes")),

                   # change NULL to NA so R recognizes it.
                   Tumor.Size =
                        as.numeric(
                             na_if(Tumor.Size, "NULL")),

                   # assuming these are the final tumor sizes.
                   # recoded levels to match the coding in "Tumor.Stage"
                   Stage =
                        as.factor(
                             case_when(Stage == "IV" ~ "4",
                                       Stage == "IIIA" ~ "3a",
                                       Stage == "IA" ~ "1a",
                                       Stage == "IVB" ~ "4b",
                                       Stage == "IIA" ~ "2a",
                                       Stage == "IIIB" ~ "3b",
                                       Stage == "IIB" ~ "2b",
                                       Stage == "IB" ~ "1b",
                                       Stage == "1B" ~ "1b")),

                   # typo
                   Primary.Site =
                        as.factor(
                             case_when(
                                  Primary.Site == "Righ Upper Lobe" ~ "Right Upper Lobe",
                                  TRUE ~ Primary.Site))) %>%

     dplyr::mutate(Outcome = case_when(Outcome == "Alive" ~ 0,
                                       Outcome == "Dead" ~ 1))

clindat_clean
```

```
## # A tibble: 190 x 16
##    ID    Outcome Surviva~1   Age Grade Num.P~2 Tumor~3      N      M Radia~4 Stage
##    <fct>   <dbl>     <int> <dbl> <int> <fct>   <fct>   <dbl> <dbl> <fct>   <fct>
##   1 1           0         9    67      4 0       <NA>           2     NA no      4
##  2 2           1         19    73     2 0       <NA>        2     0 yes     4
##  3 3           1         13    72     3 0       2           2     0 no      3a
##  4 4           1         15    69    NA 1       1a          0     1 no      1a
##  5 5           1         10    76    NA 0       <NA>       NA    NA no      3a
##  6 6           1         11    62    NA 0       3           2    NA no      4b
##  7 7           1         13    72     2 0       4          NA     1 yes     4
##  8 8           1         13    72     2 0       3          NA     0 no      2a
##  9 9           1         19    72    NA 0       4          NA     0 no      2a
## 10 10          0          9    83     3 0       <NA>       NA    NA yes     4
## # ... with 180 more rows, 5 more variables: Primary.Site <fct>,
## #   Histology <chr>, Tumor.Size <dbl>, Num.Mutated.Genes <int>,
## #   Num.Mutations <int>, and abbreviated variable names 1: Survival.Months,
## #   2: Num.Primaries, 3: Tumor.Stage, 4: Radiation
## # i Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

Clean Data: Genomic Data

```
# add a data frame of just IDs, this allows us to add rows for those patients
# where no mutation was found
ID_frame <-
     data.frame(ID = 1:190) %>%
     dplyr::mutate(ID = as.factor(ID))

# change both columns to factors.
genodat_clean <-

     genodat %>%

     # add this dummy column for pivot_wider
     dplyr::mutate(values = 1) %>%

     # change orientation of data frame so each gene is a column
     # (allows for joining with clinical data)
     tidyr::pivot_wider(names_from = Gene, values_from = values) %>%

     # change ID column type for compatibility with clindat
     dplyr::mutate(ID = as.factor(ID))

# join with ID_frame to add patient IDs with no mutations
# and replace NAs with 0s.
genodat_clean2 <-
     dplyr::full_join(ID_frame, genodat_clean) %>%
     dplyr::mutate_if(is.numeric, ~replace(., is.na(.), 0))
```

```
## Joining, by = "ID"
```

```
# Join with clinical data (each mutation can now be used as a feature)
cgdat_full <-
     dplyr::full_join(clindat_clean, genodat_clean2)
```

```
## Joining, by = "ID"
```

```
cgdat_full
```

```
## # A tibble: 190 x 66
##    ID    Outcome Surviva~1   Age Grade Num.P~2 Tumor~3     N     M Radia~4 Stage
##    <fct>   <dbl>     <int> <dbl> <int> <fct>   <fct>   <dbl> <dbl> <fct>   <fct>
##  1 1           0         9    67     4 0       <NA>        2    NA no      4
##  2 2           1        19    73     2 0       <NA>        2     0 yes     4
##  3 3           1        13    72     3 0       2           2     0 no      3a
##  4 4           1        15    69    NA 1       1a          0     1 no      1a
##  5 5           1        10    76    NA 0       <NA>       NA    NA no      3a
##  6 6           1        11    62    NA 0       3           2    NA no      4b
##  7 7           1        13    72     2 0       4          NA     1 yes     4
##  8 8           1        13    72     2 0       3          NA     0 no      2a
##  9 9           1        19    72    NA 0       4          NA     0 no      2a
## 10 10          0         9    83     3 0       <NA>       NA    NA yes     4
## # ... with 180 more rows, 55 more variables: Primary.Site <fct>,
## #   Histology <chr>, Tumor.Size <dbl>, Num.Mutated.Genes <int>,
## #   Num.Mutations <int>, AKT1 <dbl>, ALK_Col1 <dbl>, ALK_Col2 <dbl>, APC <dbl>,
## #   ATM_Col1 <dbl>, ATM_Col2 <dbl>, BRAF <dbl>, CCND2 <dbl>, CDKN2A <dbl>,
## #   CTNNB1 <dbl>, DNMT3A <dbl>, EGFR <dbl>, ERBB3 <dbl>, ERBB4 <dbl>,
## #   ESR1 <dbl>, FBXW7 <dbl>, FGFR1 <dbl>, FGFR3 <dbl>, FLT4 <dbl>, FOXL2 <dbl>,
## #   GNAS <dbl>, HNF1A <dbl>, KRAS_Col1 <dbl>, KRAS_Col2 <dbl>, ...
## # i Use ‘print(n = ...)‘ to see more rows, and ‘colnames()‘ to see all variable names
```

2. Tell us how you decided which features to use for your model. Are there any new features it might be productive to engineer from the current features?

Once I had cleaned the data, I looked at the distribution and variances of all of the different potential features and I assessed collinearity of potential features where possible. From this I found that the correlation between ‘Num.Mutated.Genes’ and ‘Num.Mutations’ was too high to include both in the analysis, so I only included ‘Num.Mutated.Genes’. Additionally, I decided to exclude ‘Stage’, ‘N’, and ‘M’ since they had many missing values which reduced the usable sample size below and acceptable level. I considered including specific gene mutation dummy variables in the analysis but since these variables are categorical, they take up many degrees of freedom each and there were simply too many to include given the number of rows in the data.

As for new features, I think it would have been interesting to consider tumor stage or size change as a time-sensitive variable for event prediction. It also would have been helpful to interpolate the ‘N’ and ‘M’ variables to fill in the missing data. Additionally, if I had specific knowledge about the genes with mutations I may have been able to reduce dimensionality by grouping some of the genes together either by function or genomic location, which may have allowed for easier inclusion as features.

3. Which algorithm did you use for the prediction and why?

I used a cox proportional hazards model because I believed it would be the most accurate model for predicting the time to an event. This model is not one I have worked with before and I underestimated the complexity of getting predictions out on the scale of months (as opposed to a relative risk/hazard estimate). Additionally, this was my first time using tidymodels for fitting predictive models.

Split the data into test and train sets

```
trainIndex <- caret::createDataPartition(cgdat_full$Outcome, p = .8,
                                          list = FALSE,
                                          times = 1)
```

```
cgdat_train <- cgdat_full[trainIndex,]
cgdat_test <- cgdat_full[-trainIndex,]
```

Modeling survival: Cox Regression

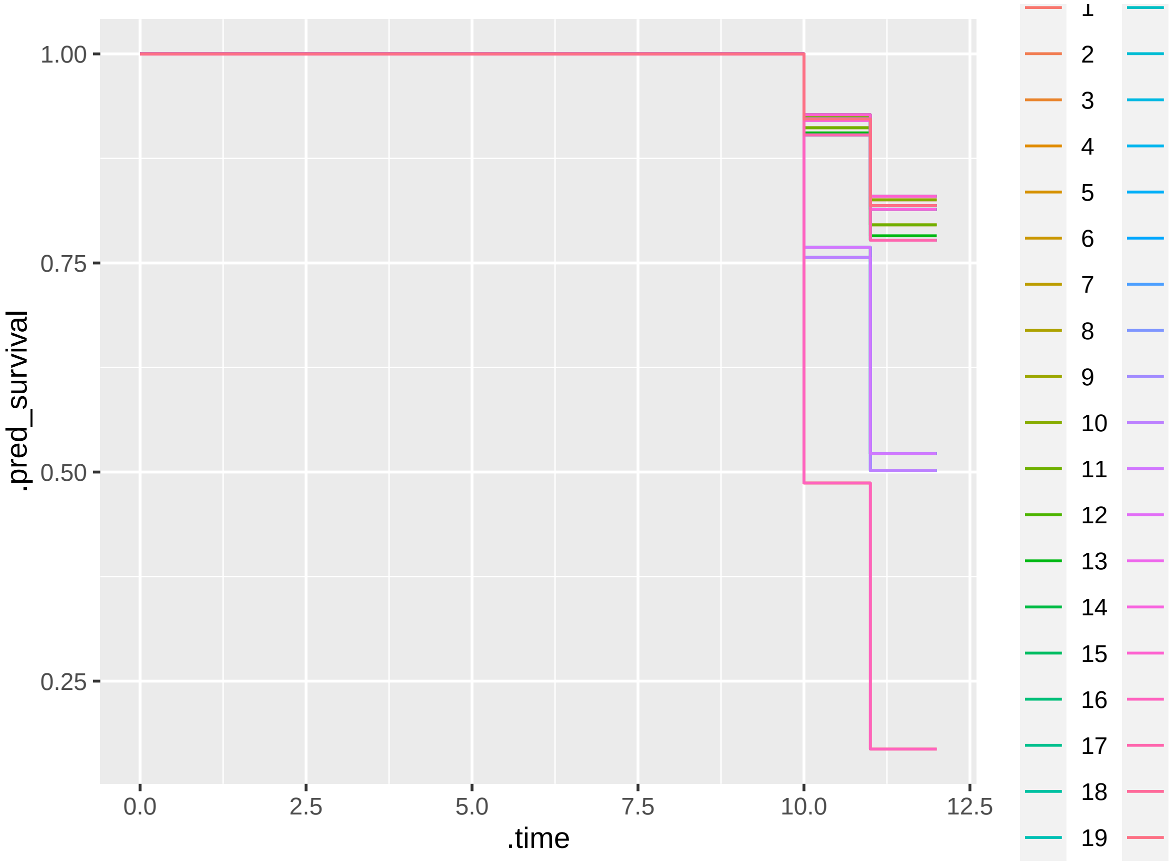
```
# Use parsnip + glmnet packages to set up the cox models
# here we used the default settings set by the glmnet package
cox_spec_glmnet <- parsnip::proportional_hazards(penalty = 0.1) %>%
    set_engine("glmnet")


# glmnet cox model
coxmod_fit_glmnet <-
  fit(cox_spec_glmnet,
      Surv(Survival.Months, Outcome, type = "right") ~
        Stage + Age + Radiation + Histology + Num.Primaries + Num.Mutated.Genes,
      data = cgdat_train)


# generating predictions of survival to 12 months
coxmod_glmnet_pred <-
  predict(coxmod_fit_glmnet,
          new_data = cgdat_test,
          type = "survival", time = seq(0, 12, 1))

# reformatting predictions
coxmod_glmnet_pred <-
  coxmod_glmnet_pred %>%
  mutate(id = factor(1:nrow(cgdat_test))) %>%
  unnest(cols = .pred)

# plot the predicted survival to 12 months
coxmod_glmnet_pred %>%
  ggplot(aes(x = .time, y = .pred_survival, col = id)) +
  geom_step()
```



4. How did you assess the predictive model's quality? Summarize your findings.

If given more time, I would have applied leave-one-out cross validation and assessed the predictive model's quality using root mean squared error (RMSE). I ran into limitations of my knowledge around both the cox proportional hazards model and the 'tidymodels' package which limited the scope of model performance assessment. What I was able to get from the model was a probability of death prediction for each row in the test set, which was very low and did not discriminate cases where the known outcome was death, suggesting that the sensitivity of this model was low.

5. Next steps? What might you do with more time or access to additional data or expertise?

If given more time:

(1) I would further familiarize myself with the tidymodels package. I realized pretty early into this assignment that the caret package, the package I have previously used to fit predictive models, does not have support for cox models. Since I believed cox models were the right tool for the job here, I then attempted to use the survival package alone and realized that I could not get actual survival time estimates using the predict function from that package. I found a vignette from the tidymodels authors on predicting survival time from cox models and I used that to find predictions for survival to 12 months, but I would need more time to make sure that the underlying assumptions and hyperparameter values were tuned correctly for this dataset.

(2) I would apply more complex algorithms for feature selection and assessing predictive performance. In particular I would use LOO cross-validation to assess model performance instead of a single test-train split. I would also consider an automated algorithm for choosing the best set of features, including consideration of individual gene mutation outcome variables.