Mutation and Clinical Analysis of Liver Hepatocellular Carcinoma Code Base

Group 15

2025-11-25

SECTION: 1

```
#loaded Libraries

library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(readr)
library(tibble)
library(ggplot2)
library(tidyr)
library(e1071)
```

```
##
## Attaching package: 'e1071'
```

```
## The following object is masked from 'package:ggplot2':
##
##     element
```

```
library(purrr)
library(survival)
library(survminer)
```

```
## Loading required package: ggpubr
```

```
##
## Attaching package: 'survminer'

## The following object is masked from 'package:survival':
##
##     myeloma
```

SECTION: 2

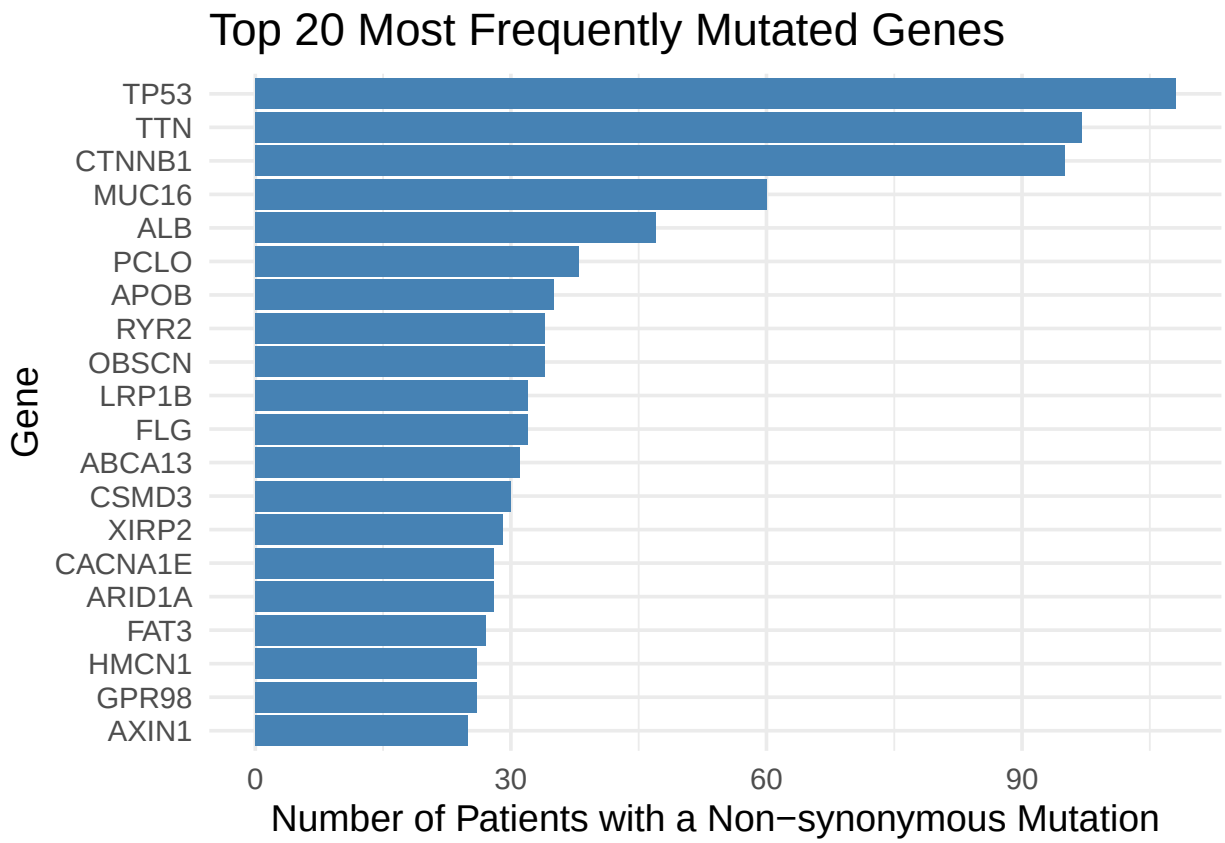
```
knitr::opts_chunk$set(echo = TRUE)
```

```
clinical <- read_tsv(
  "data_clinical_patient.txt",
  comment = "#",
  show_col_types = FALSE
)

mutation <- read.delim(
  "data_mutations.txt",
  sep = "\t",
  header = TRUE,
  stringsAsFactors = FALSE
)

# --- Helper: normalize TCGA IDs to first 12 chars with dashes ---
normalize_tcga <- function(x) {
  x <- as.character(x)
  x <- gsub("\\.", "-", x)
  substr(x, 1, 12)
}

# --- Get patient IDs in clinical and mutation data ---
clin_ids <- unique(normalize_tcga(clinical$PATIENT_ID))
mut_ids  <- unique(normalize_tcga(mutation$Tumor_Sample_Barcode))

# Only keep patients that are in BOTH clinical and mutation datasets
common_ids <- intersect(clin_ids, mut_ids)

# --- Filter clinical to common patients (not actually used below, but kept for completeness) ---
clinical <- clinical %>%
  mutate(PATIENT_ID = normalize_tcga(PATIENT_ID)) %>%
  filter(PATIENT_ID %in% common_ids)

# --- Filter to NON-SYNONYMOUS mutations (single clean list) ---
nonsynonymous <- c(
  "Missense_Mutation", "Nonsense_Mutation",
  "Nonstop_Mutation",
  "Frame_Shift_Del", "Frame_Shift_Ins",
  "In_Frame_Del", "In_Frame_Ins",
  "Splice_Site", "Translation_Start_Site"
)
```

```r
mutation_ns <- mutation %>%
  mutate(PATIENT_ID = normalize_tcga(Tumor_Sample_Barcode)) %>%
  filter(
    Variant_Classification %in% nonsynonymous,
    PATIENT_ID %in% common_ids
  )

# --- Build Gene × Patient mutation matrix (0/1) ---
genes    <- sort(unique(mutation_ns$Hugo_Symbol))
patients <- sort(unique(mutation_ns$PATIENT_ID))

mut_matrix <- matrix(
  0L,
  nrow = length(genes),
  ncol = length(patients),
  dimnames = list(genes, patients)
)

for (i in seq_len(nrow(mutation_ns))) {
  g <- mutation_ns$Hugo_Symbol[i]
  p <- mutation_ns$PATIENT_ID[i]
  mut_matrix[g, p] <- 1L
}

# --- Gene mutation frequencies across the cohort ---
gene_freq <- rowSums(mut_matrix)  # how many patients mutated per gene

top_n <- 20  # change this if you want more/less
top_genes <- sort(gene_freq, decreasing = TRUE)[1:top_n]

top_genes_df <- tibble(
  Gene = names(top_genes),
  Mutated_Patients = as.integer(top_genes)
)
```

SECTION: 3

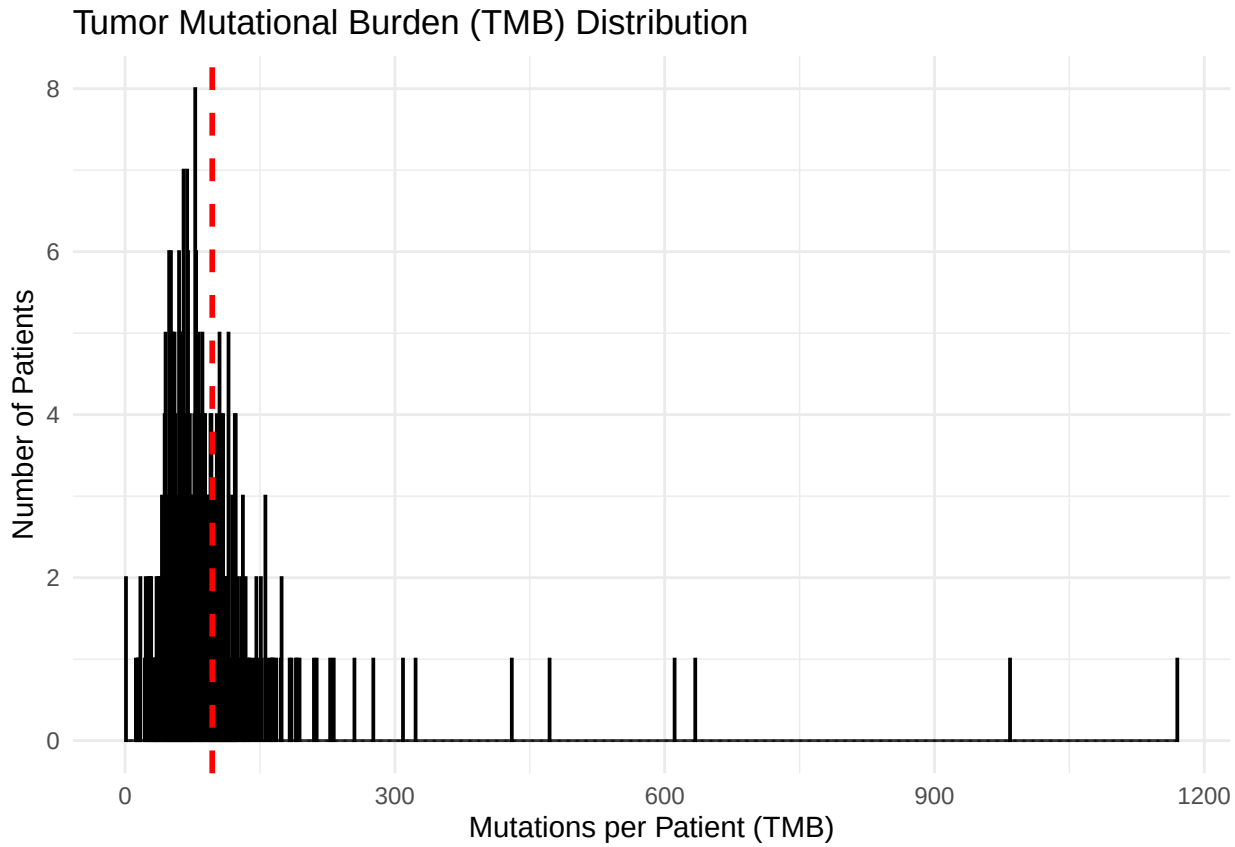
```r
# Bar plot of most frequently mutated genes ---
ggplot(top_genes_df,
       aes(x = reorder(Gene, Mutated_Patients),
           y = Mutated_Patients)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  theme_minimal(base_size = 14) +
  labs(
    title = paste("Top", top_n, "Most Frequently Mutated Genes"),
    x = "Gene",
    y = "Number of Patients with a Non-synonymous Mutation"
  )
```



SECTION: 4

```
gene_mut_count <- rowSums(mut_matrix)
gene_df <- data.frame(
  Gene = names(gene_mut_count),
  Mutated_Patients = gene_mut_count
)


tmb_per_patient <- colSums(mut_matrix)
tmb_df <- data.frame(
  Patient = names(tmb_per_patient),
  TMB = tmb_per_patient
)

# Plot: TMB distribution
ggplot(tmb_df, aes(x = TMB)) +
  geom_histogram(binwidth = 1, fill = "purple", color = "black") +
  geom_vline(aes(xintercept = mean(TMB)), color = "red", linetype = "dashed", size = 1) +
  labs(title = "Tumor Mutational Burden (TMB) Distribution",
       x = "Mutations per Patient (TMB)",
       y = "Number of Patients") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



SECTION: 5

```
patient_mut_counts <- colSums(mut_matrix)

mean_mutated_genes <- mean(patient_mut_counts)

mean_mutated_genes
```

```
## [1] 96.90503
```

SECTION: 6

```
gene_variance <- apply(mut_matrix, 1, var)

# Sort descending
```

```r
gene_variance_sorted <- sort(gene_variance, decreasing = TRUE)

# Look at the top variance genes
head(gene_variance_sorted, 20)
```

```
##       TP53        TTN     CTNNB1      MUC16        ALB       PCLO       APOB
## 0.21125769 0.19808929 0.19549160 0.13989954 0.11436865 0.09514420 0.08845438
##      OBSCN       RYR2      LRP1B        FLG     ABCA13      CSMD3      XIRP2
## 0.08619314 0.08619314 0.08162371 0.08162371 0.07931553 0.07699169 0.07465221
##     ARID1A    CACNA1E       FAT3      HMCN1      GPR98      AXIN1
## 0.07229708 0.07229708 0.06992629 0.06753987 0.06753987 0.06513779
```

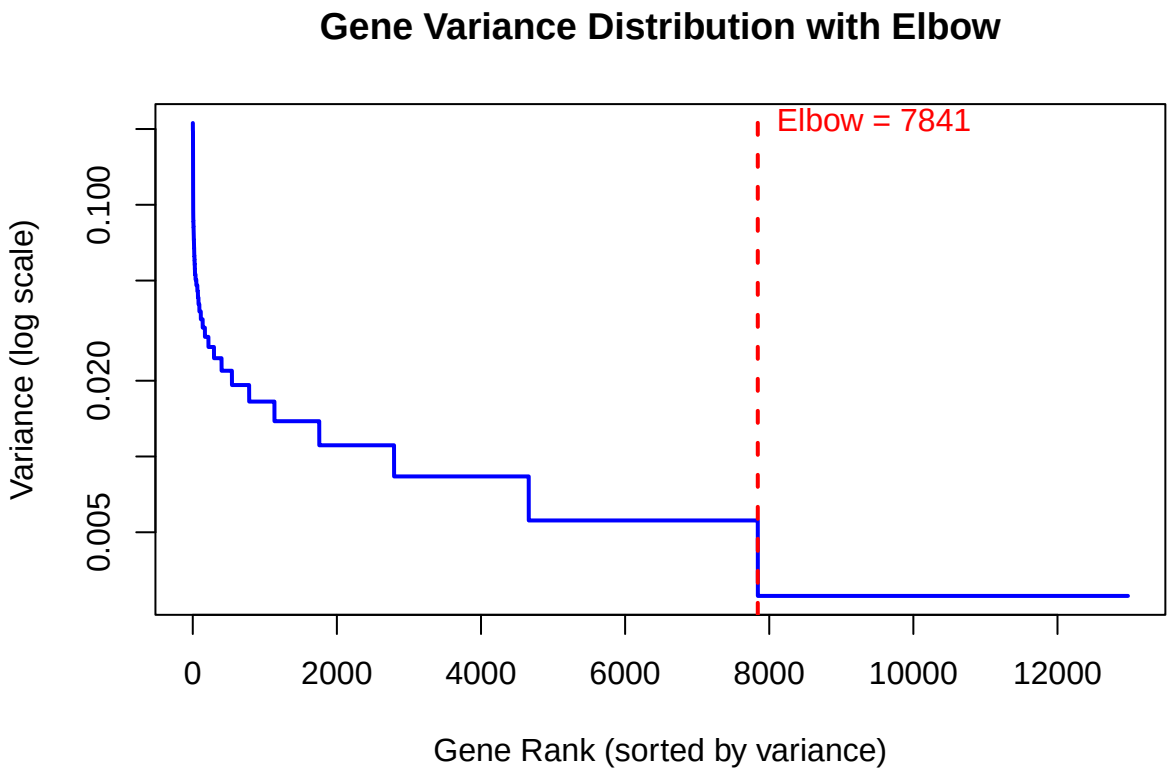
SECTION: 7

```r
# 1. Gene variance across patients
gene_variance <- apply(mut_matrix, 1, var)

# 2. Sort by variance (high → low)
gene_variance_sorted <- sort(gene_variance, decreasing = TRUE)

# 3. Basic variance curve (log scale)
plot(
  gene_variance_sorted,
  type = "l",
  log = "y",
  lwd = 2,
  col = "blue",
  xlab = "Gene Rank (sorted by variance)",
  ylab = "Variance (log scale)",
  main = "Gene Variance Distribution with Elbow"
)

# 4. Approximate elbow via second derivative on log-variance
y_log <- log10(gene_variance_sorted + 1e-10)  # avoid log(0)
d1 <- diff(y_log)
d2 <- diff(d1)

# We look for the point of **maximum curvature**
elbow <- which.max(-d2) + 1   # +1 to align with original index

elbow
```

```
## ABCB7
##  7841
```

```r
# 5. Add vertical line at elbow
abline(v = elbow, col = "red", lwd = 2, lty = 2)

text(
  x = elbow,
  y = max(gene_variance_sorted),
```

```
  labels = paste("Elbow =", elbow),
  pos = 4,
  col = "red"
)
```




```
# 6. Extract "significant" genes (above elbow)
significant_genes <- names(gene_variance_sorted)[1:elbow]
head(significant_genes)
```

```
## [1] "TP53"   "TTN"    "CTNNB1" "MUC16"  "ALB"    "PCLO"
```

```
length(significant_genes)
```

```
## [1] 7841
```

A small group of genes drive almost all variation Most genes in the genome are not mutated in this cancer type The “important” high-variance genes are the ones involved in the disease

Top 6 most frequelntly mutated genes “TP53” “TTN” “CTNNB1” “MUC16” “ALB” “PCLO”

SECTION: 8

```r
# --- PCA to Assess Variance Structure ---

# 1. Transpose mut_matrix → rows = patients, cols = genes
X <- t(mut_matrix)

# 2. PCA (center & scale recommended for binary mutation data)
pca_res <- prcomp(X, center = TRUE, scale. = TRUE)

# 3. Variance explained per PC
var_explained <- pca_res$sdev^2 / sum(pca_res$sdev^2)

# 4. Cumulative variance explained
cum_var <- cumsum(var_explained)

# 5. Find how many PCs needed for 90% variance
num_pcs_90 <- which(cum_var >= 0.90)[1]

num_pcs_90
```

```
## [1] 270
```

I ran PCA here to understand why we don't want to be using PCA! 270 PCAs is alot typically when doing PCA analysis we end up / want to end up with less then 10 PCAs, however here the reason that we have so many is because there is so many genes and some of them only have a couple mutations means that this will have low variance but becasue so many of them have low varianvce PCA needs alot more PCAs to achive 90 percent

SECTION: 9

Question 2: can a model be predicted that can accuratly determine the stage of cancer.

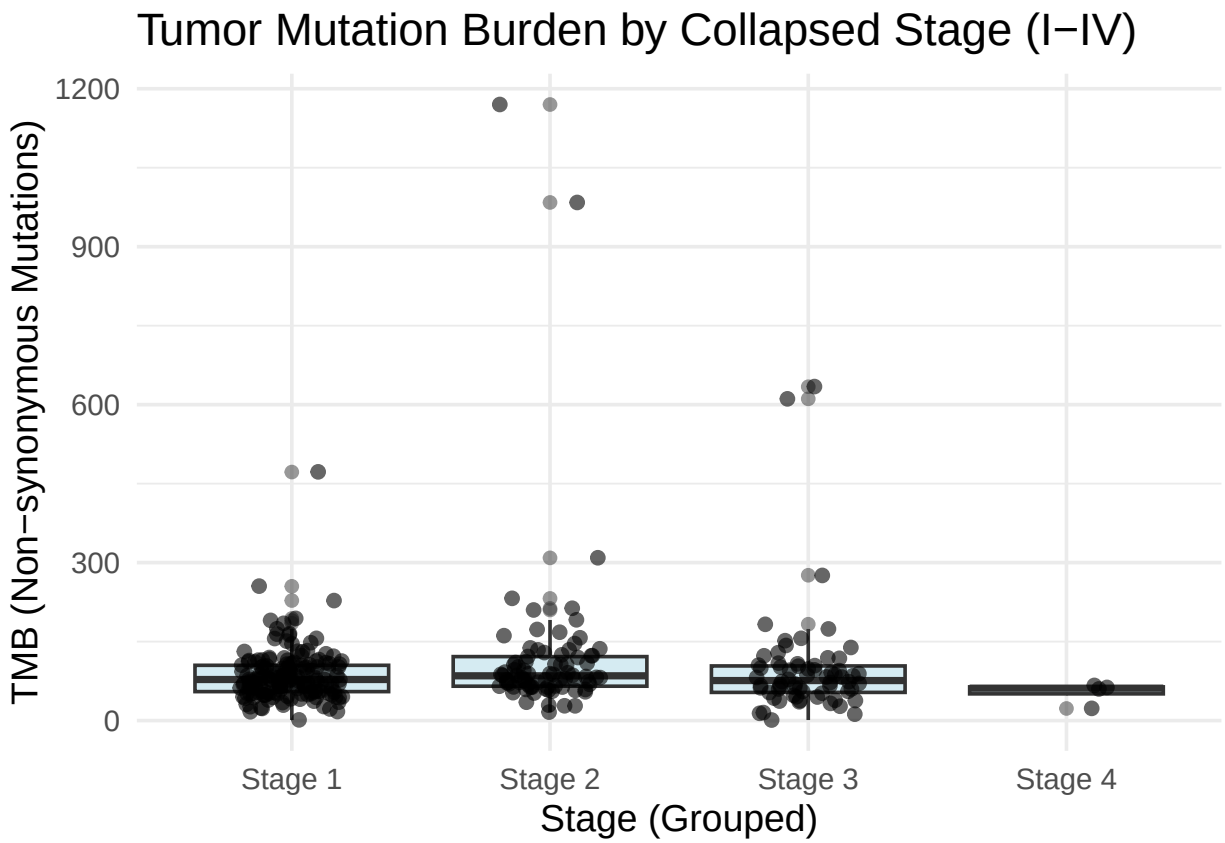
```r
stage_col <- "AJCC_PATHOLOGIC_TUMOR_STAGE"



clean_stage_to_14 <- function(x) {
  x <- toupper(as.character(x))
  x <- gsub("^STAGE\\s*", "", x)
  x <- gsub("[A]+$", "", x)
  x <- trimws(x)

  dplyr::case_when(
    x %in% c("I", "1")   ~ 1L,
    x %in% c("II", "2")  ~ 2L,
    x %in% c("III", "3") ~ 3L,
    x %in% c("IV", "4")  ~ 4L,
    TRUE ~ NA_integer_
  )
}

clinical_stage <- clinical %>%
  mutate(
```

```
    Stage_raw = .data[[stage_col]],
    Stage_num = clean_stage_to_14(Stage_raw),
    Stage_grp = factor(
      Stage_num,
      levels = 1:4,
      labels = paste("Stage", 1:4)
    )
  ) %>%
  filter(!is.na(Stage_num)) %>%
  select(PATIENT_ID, Stage_raw, Stage_num, Stage_grp)

clinical_stage
```

```
## # A tibble: 321 x 4
##    PATIENT_ID   Stage_raw  Stage_num Stage_grp
##    <chr>        <chr>          <int> <fct>
##  1 TCGA-2V-A95S STAGE II           2 Stage 2
##  2 TCGA-2Y-A9GS STAGE IV           4 Stage 4
##  3 TCGA-2Y-A9GT STAGE I            1 Stage 1
##  4 TCGA-2Y-A9GU STAGE I            1 Stage 1
##  5 TCGA-2Y-A9GV STAGE I            1 Stage 1
##  6 TCGA-2Y-A9GW STAGE I            1 Stage 1
##  7 TCGA-2Y-A9GX STAGE I            1 Stage 1
##  8 TCGA-2Y-A9GY STAGE II           2 Stage 2
##  9 TCGA-2Y-A9GZ STAGE II           2 Stage 2
## 10 TCGA-2Y-A9H0 STAGE IIIA         3 Stage 3
## # i 311 more rows
```

SECTION: 10

```
tmb_df <- tmb_df %>%
  rename(PATIENT_ID = Patient)

tmb_stage <- tmb_df %>%
  inner_join(clinical_stage, by = "PATIENT_ID")

ggplot(tmb_stage, aes(x = Stage_grp, y = TMB)) +
  geom_boxplot(fill = "lightblue", alpha = 0.5) +
  geom_jitter(width = 0.2, alpha = 0.6) +
  theme_minimal(base_size = 14) +
  labs(
    title = "Tumor Mutation Burden by Collapsed Stage (I-IV)",
    x = "Stage (Grouped)",
    y = "TMB (Non-synonymous Mutations)"
  )
```



Look into the TMB to see if maybe total number of mutations could be an indicator of stage of cancer. From this box plot it looks like most are in similar ranges meaning purely quantity of mutated genes is not a good indicator. So further analysis on specifc mutated genes would potenitally be a better indicator of cancer stage be a better indicator

TMB alone is not a good staging biomarker These box plots do not have enough variation a model will not be able to distiguish these catigoeies

SECTION: 11

```
table(clinical_stage$Stage_grp)
```

```
##
## Stage 1 Stage 2 Stage 3 Stage 4
##     169      82      66       4
```

```
stage_counts <- clinical_stage %>%
  count(Stage_grp)

stage_counts
```

```
## # A tibble: 4 x 2
##   Stage_grp     n
##   <fct>     <int>
```

```
## 1 Stage 1     169
## 2 Stage 2      82
## 3 Stage 3      66
## 4 Stage 4       4
```

Troubles: cannot do with by simply splitting stages because stage for only has 4 patients so we will combine into early and late stage cancer instead.

SECTION: 12

```
clinical_stage <- clinical_stage %>%
  mutate(
    Stage_binary_2 = case_when(
      Stage_num %in% c(1, 2) ~ "Early Stages",
      Stage_num %in% c(3, 4) ~ "Late Stages",
      TRUE ~ NA_character_
    ),
    Stage_binary_2 = factor(
      Stage_binary_2,
      levels = c("Early Stages", "Late Stages")
    )
  )
```

SECTION: 13

```
gene_freq <- rowSums(mut_matrix)
top_genes <- names(sort(gene_freq, decreasing = TRUE))[1:100] # look at top 100 mutated genes
top_genes
```

```
##   [1] "TP53"    "TTN"     "CTNNB1"  "MUC16"   "ALB"     "PCLO"    "APOB"
##   [8] "OBSCN"   "RYR2"    "FLG"     "LRP1B"   "ABCA13"  "CSMD3"   "XIRP2"
##  [15] "ARID1A"  "CACNA1E" "FAT3"    "GPR98"   "HMCN1"   "AXIN1"   "RYR1"
##  [22] "SPTA1"   "USH2A"   "CSMD1"   "DNAH7"   "CUBN"    "DOCK2"   "PRKDC"
##  [29] "PKHD1L1" "RYR3"    "AHNAK2"  "BAP1"    "DMD"     "FBN2"    "FRAS1"
##  [36] "HERC2"   "SDK1"    "WDR87"   "ARID2"   "BIRC6"   "COL11A1" "DNAH5"
##  [43] "FASN"    "FREM2"   "KMT2D"   "LRP1"    "PCDH15"  "RB1"     "ZNF469"
##  [50] "ABCA12"  "COL6A6"  "DCHS1"   "DNAH6"   "DNAH8"   "EYS"     "HTT"
##  [57] "KEAP1"   "LAMA1"   "MUC5B"   "NBEA"    "PREX2"   "PRUNE2"  "UNC80"
##  [64] "DNAH9"   "FAT2"    "KMT2C"   "LRP2"    "MYT1L"   "NEB"     "PKHD1"
##  [71] "PTPRQ"   "DNAH10"  "DYNC2H1" "FAT4"    "FCGBP"   "FMN2"    "LRRIQ1"
##  [78] "SYNE1"   "SYNE2"   "COL12A1" "FBN1"    "HYDIN"   "KMT2B"   "MUC17"
##  [85] "MYO3A"   "RPS6KA3" "SPEG"    "TG"      "TPO"     "TRPA1"   "VWDE"
##  [92] "AHNAK"   "CACNA1B" "COL6A3"  "DCHS2"   "DNAH1"   "DNAH17"  "DNAH2"
##  [99] "MUC2"    "MUC4"
```

```
gene_df <- t(mut_matrix[top_genes, ]) %>%
  as.data.frame() %>%
  mutate(PATIENT_ID = rownames(.)) %>%
  inner_join(clinical_stage, by = "PATIENT_ID")
```

```r
# Calculate mutation proportion per stage
stage_gene_freq <- gene_df %>%
  group_by(Stage_binary_2) %>%
  summarise(across(top_genes, mean))
```

```
## Warning: There was 1 warning in `summarise()`.
## i In argument: `across(top_genes, mean)`.
## Caused by warning:
## ! Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(top_genes)
##
##   # Now:
##   data %>% select(all_of(top_genes))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
```

```r
stage_gene_freq
```

```
## # A tibble: 2 x 101
##   Stage_binary_2  TP53   TTN CTNNB1 MUC16   ALB   PCLO   APOB  OBSCN   RYR2
##   <fct>          <dbl> <dbl>  <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 Early Stages   0.315 0.271  0.263 0.163 0.139 0.0996 0.0876 0.0876 0.0996
## 2 Late Stages    0.286 0.243  0.3   0.157 0.114 0.129  0.114  0.1    0.0714
## # i 91 more variables: FLG <dbl>, LRP1B <dbl>, ABCA13 <dbl>, CSMD3 <dbl>,
## #   XIRP2 <dbl>, ARID1A <dbl>, CACNA1E <dbl>, FAT3 <dbl>, GPR98 <dbl>,
## #   HMCN1 <dbl>, AXIN1 <dbl>, RYR1 <dbl>, SPTA1 <dbl>, USH2A <dbl>,
## #   CSMD1 <dbl>, DNAH7 <dbl>, CUBN <dbl>, DOCK2 <dbl>, PRKDC <dbl>,
## #   PKHD1L1 <dbl>, RYR3 <dbl>, AHNAK2 <dbl>, BAP1 <dbl>, DMD <dbl>, FBN2 <dbl>,
## #   FRAS1 <dbl>, HERC2 <dbl>, SDK1 <dbl>, WDR87 <dbl>, ARID2 <dbl>,
## #   BIRC6 <dbl>, COL11A1 <dbl>, DNAH5 <dbl>, FASN <dbl>, FREM2 <dbl>, ...
```

SECTION: 14

```r
genes <- setdiff(names(stage_gene_freq), "Stage_binary_2")

# Extract rows for Early and Late
early_row <- stage_gene_freq %>%
  filter(Stage_binary_2 == "Early Stages") %>%
  select(all_of(genes))

late_row <- stage_gene_freq %>%
  filter(Stage_binary_2 == "Late Stages") %>%
  select(all_of(genes))

diff_vec <- as.numeric(late_row[1, ] - early_row[1, ])

gene_diff_tbl <- tibble(
  Gene     = genes,
```

```
  Diff     = diff_vec,
  Abs_Diff = abs(diff_vec)
) %>%
  arrange(desc(Abs_Diff))

gene_diff_tbl <- gene_diff_tbl[1:20, ]

write.table(gene_diff_tbl, row.names = FALSE, sep = ",")
```

```
## "Gene","Diff","Abs_Diff"
## "FAT3",0.0688104723961298,0.0688104723961298
## "FCGBP",0.0681274900398406,0.0681274900398406
## "DNAH6",-0.0637450199203187,0.0637450199203187
## "DNAH8",0.0601593625498008,0.0601593625498008
## "GPR98",-0.0590779738190097,0.0590779738190097
## "RPS6KA3",0.0578258394991463,0.0578258394991463
## "PKHD1L1",-0.0494593056346044,0.0494593056346044
## "USH2A",0.048207171314741,0.048207171314741
## "LRP1B",0.0449060899260102,0.0449060899260102
## "COL11A1",-0.0414911781445646,0.0414911781445646
## "LAMA1",-0.0414911781445646,0.0414911781445646
## "PKHD1",-0.0414911781445646,0.0414911781445646
## "FMN2",-0.0414911781445646,0.0414911781445646
## "CACNA1E",0.0385885031303358,0.0385885031303358
## "DNAH5",0.0379055207740467,0.0379055207740467
## "BAP1",-0.0375071143995447,0.0375071143995447
## "LRP2",-0.0375071143995447,0.0375071143995447
## "CTNNB1",0.0370517928286853,0.0370517928286853
## "HERC2",-0.0351735913488902,0.0351735913488902
## "ZNF469",-0.0351735913488902,0.0351735913488902
```

SECTION: 15

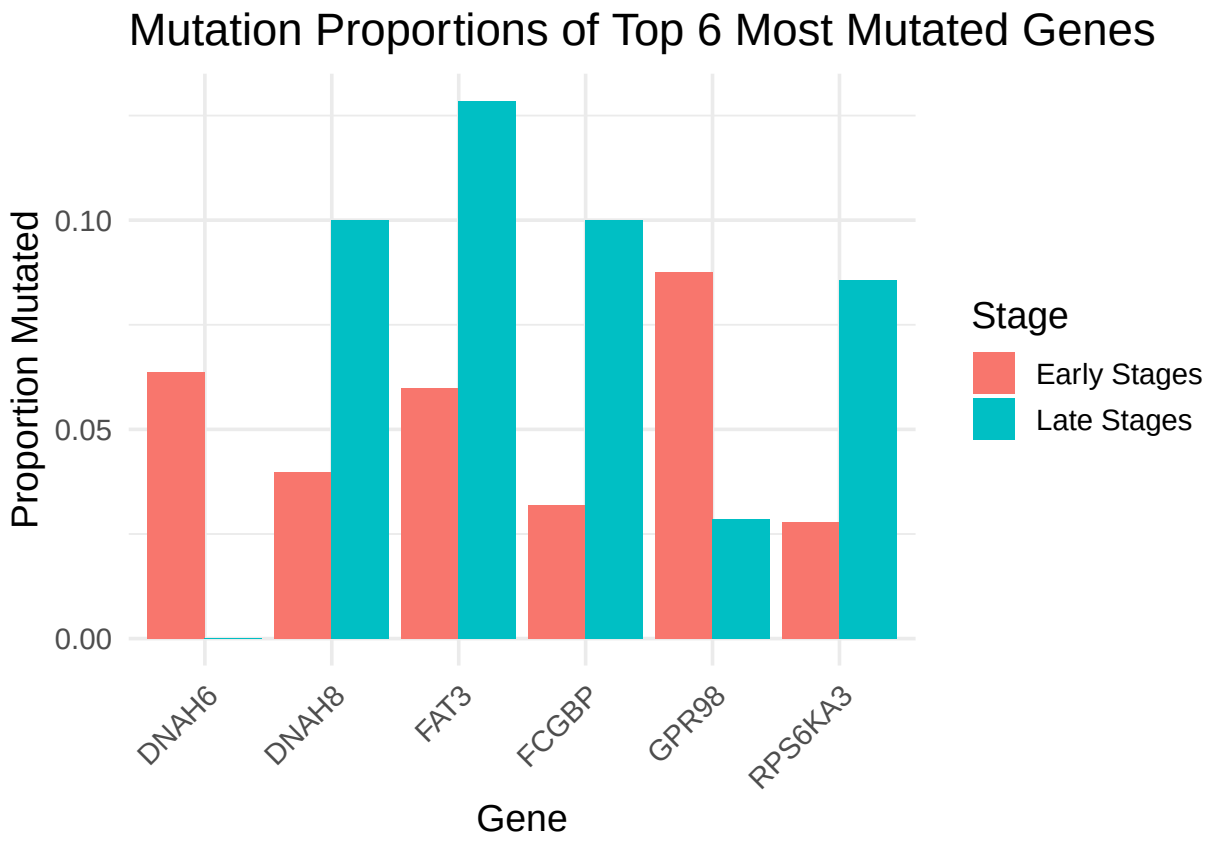
```
gene_freq <- rowSums(mut_matrix)
top_50_genes <- names(sort(gene_freq, decreasing = TRUE))[1:100]     #changed to 100 but didnt want to

top50_rank_lookup <- setNames(seq_along(top_50_genes), top_50_genes)

gene_diff_ranked <- gene_diff_tbl %>%
  mutate(
    DiffRank = row_number(),
    Rank_in_Top50 = top50_rank_lookup[Gene]
  )

gene_diff_ranked
```

```
## # A tibble: 20 x 5
##    Gene       Diff Abs_Diff DiffRank Rank_in_Top50
##    <chr>     <dbl>    <dbl>    <int>         <int>
##  1 FAT3     0.0688   0.0688        1            17
##  2 FCGBP    0.0681   0.0681        2            75
```

```
##  3 DNAH6   -0.0637   0.0637        3            53
##  4 DNAH8    0.0602   0.0602        4            54
##  5 GPR98   -0.0591   0.0591        5            18
##  6 RPS6KA3  0.0578   0.0578        6            86
##  7 PKHD1L1 -0.0495   0.0495        7            29
##  8 USH2A    0.0482   0.0482        8            23
##  9 LRP1B    0.0449   0.0449        9            11
## 10 COL11A1 -0.0415   0.0415       10            41
## 11 LAMA1   -0.0415   0.0415       11            58
## 12 PKHD1   -0.0415   0.0415       12            70
## 13 FMN2    -0.0415   0.0415       13            76
## 14 CACNA1E  0.0386   0.0386       14            16
## 15 DNAH5    0.0379   0.0379       15            42
## 16 BAP1    -0.0375   0.0375       16            32
## 17 LRP2    -0.0375   0.0375       17            67
## 18 CTNNB1   0.0371   0.0371       18             3
## 19 HERC2   -0.0352   0.0352       19            36
## 20 ZNF469  -0.0352   0.0352       20            49
```

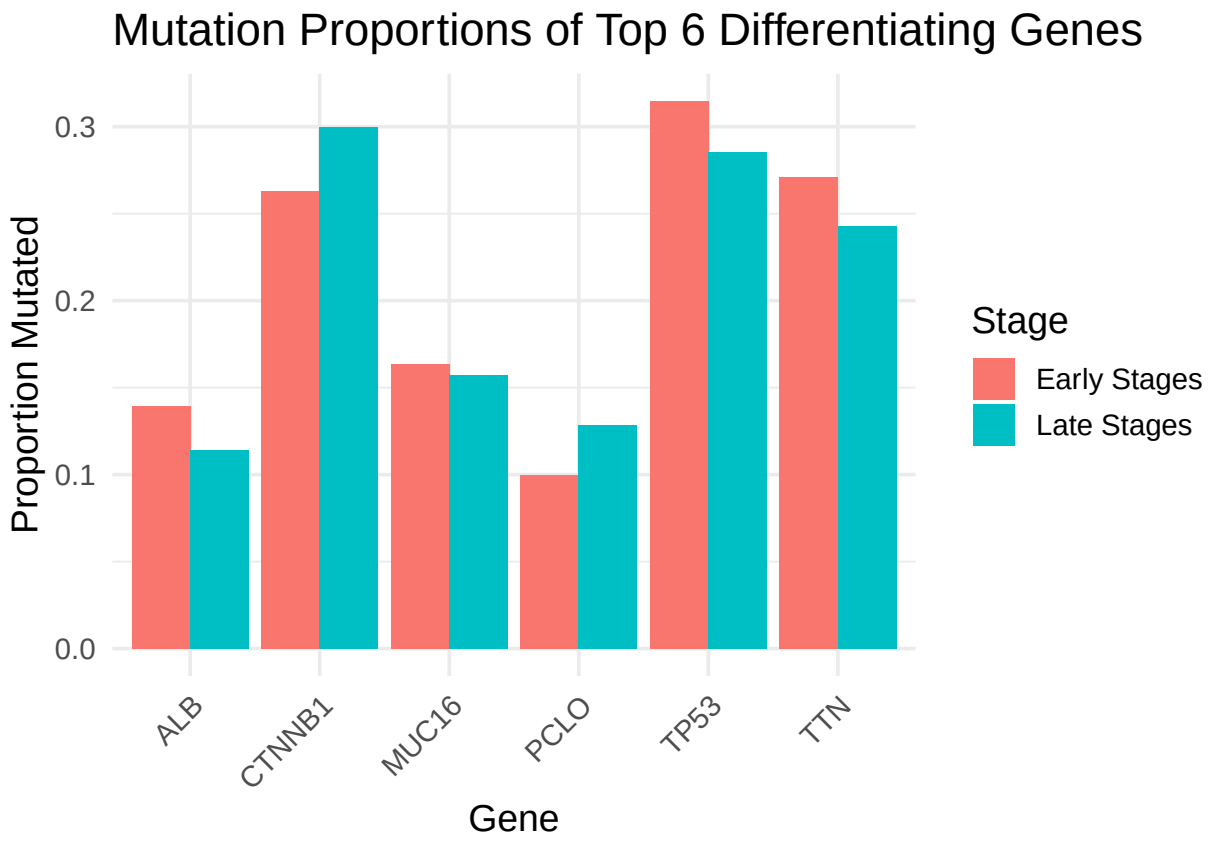
SECTION: 16

```r
top10_genes <- gene_diff_tbl$Gene[1:6]

bar_df <- stage_gene_freq %>%
  select(Stage_binary_2, all_of(top10_genes)) %>%
  pivot_longer(cols = all_of(top10_genes),
               names_to = "Gene",
               values_to = "Proportion")

ggplot(bar_df, aes(x = Gene, y = Proportion, fill = Stage_binary_2)) +
  geom_col(position = "dodge") +
  theme_minimal(base_size = 14) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(
    title = "Mutation Proportions of Top 6 Most Mutated Genes",
    y = "Proportion Mutated",
    fill = "Stage"
  )
```



SECTION: 17

```
gene_freq <- rowSums(mut_matrix)
top10_genes <- names(sort(gene_freq, decreasing = TRUE))[1:6]

bar_df <- stage_gene_freq %>%
  select(Stage_binary_2, all_of(top10_genes)) %>%
  pivot_longer(cols = all_of(top10_genes),
               names_to = "Gene",
               values_to = "Proportion")

ggplot(bar_df, aes(x = Gene, y = Proportion, fill = Stage_binary_2)) +
  geom_col(position = "dodge") +
  theme_minimal(base_size = 14) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(
    title = "Mutation Proportions of Top 6 Differentiating Genes",
    y = "Proportion Mutated",
    fill = "Stage"
  )
```



This shows that after performing some preliminary re-organization of the data, we now have a structured set of mutation features that can more effectively separate cancer stages into early and late groups. This improves the usefulness of the dataset and should increase the accuracy of any future predictive models by ensuring they are trained on meaningful, relevant information — garbage in, garbage out.

Additionally, we can now see that DNAH6 appears only in the early-stage group. This suggests that DNAH6 may be a particularly valuable feature for helping the model distinguish between early and late cancer stages.

SECTION: 18

```
clean_stage_to_14 <- function(x) {
  x <- toupper(as.character(x))
  x <- gsub("^STAGE\\s*", "", x)
  x <- gsub("[A]+$", "", x)
  x <- trimws(x)

  dplyr::case_when(
    x %in% c("I", "1")   ~ 1L,
    x %in% c("II", "2")  ~ 2L,
    x %in% c("III", "3") ~ 3L,
    x %in% c("IV", "4")  ~ 4L,
    TRUE ~ NA_integer_
  )
}
```

```r
clinical_stage <- clinical %>%
  mutate(
    Stage_raw = AJCC_PATHOLOGIC_TUMOR_STAGE,
    Stage_num = clean_stage_to_14(Stage_raw),
    Stage_lowhigh = case_when(
      Stage_num %in% c(1, 2) ~ "Low Stage",   # Stage I-II
      Stage_num %in% c(3, 4) ~ "High Stage",  # Stage III-IV
      TRUE ~ NA_character_
    )
  ) %>%
  filter(!is.na(Stage_lowhigh)) %>%
  select(PATIENT_ID, Stage_num, Stage_lowhigh)
```

SECTION: 19

```r
library(dplyr)
library(e1071)

genes_for_model <- intersect(gene_diff_tbl$Gene[1:30], rownames(mut_matrix))

mut_df <- t(mut_matrix[genes_for_model, , drop = FALSE]) %>%
  as.data.frame() %>%
  mutate(PATIENT_ID = rownames(.))

df <- mut_df %>%
  inner_join(clinical_stage %>% select(PATIENT_ID, Stage_lowhigh),
             by = "PATIENT_ID") %>%
  filter(!is.na(Stage_lowhigh))

df$Stage_lowhigh <- factor(df$Stage_lowhigh)

# Prepare results storage
results <- data.frame(seed = integer(), accuracy = numeric(),
                      sensitivity = numeric(), specificity = numeric(),
                      precision = numeric(), stringsAsFactors = FALSE)

threshold <- 0.99

# Run multiple seeds
for(i in 1:10){   # can increase to 50
  set.seed(i)

  # stratified split
  low <- df %>% filter(Stage_lowhigh == "Low Stage")
  high <- df %>% filter(Stage_lowhigh == "High Stage")

  # skip if too few patients
  if(nrow(low) < 2 | nrow(high) < 2) next

  low_idx <- sample(seq_len(nrow(low)), size = floor(0.75 * nrow(low)))
  high_idx <- sample(seq_len(nrow(high)), size = floor(0.75 * nrow(high)))
```

```
  train <- bind_rows(low[low_idx, ], high[high_idx, ])
  test  <- bind_rows(low[-low_idx, ], high[-high_idx, ])

  train_nb <- train[, c(genes_for_model, "Stage_lowhigh")]

  nb_model <- naiveBayes(Stage_lowhigh ~ ., data = train_nb)

  # Predict probabilities
  nb_prob <- predict(nb_model, newdata = test[, genes_for_model, drop = FALSE], type = "raw")

  # Handle cases where one class is missing
  if(!"High Stage" %in% colnames(nb_prob)){
    nb_prob <- cbind(nb_prob, "High Stage" = 0)
  }

  nb_pred <- ifelse(nb_prob[, "High Stage"] > threshold, "High Stage", "Low Stage")
  nb_pred <- factor(nb_pred, levels = levels(df$Stage_lowhigh))

  cm <- table(Predicted = nb_pred, Actual = test$Stage_lowhigh)

  TP <- cm["High Stage", "High Stage"] %||% 0
  TN <- cm["Low Stage",  "Low Stage"]  %||% 0
  FP <- cm["High Stage", "Low Stage"]  %||% 0
  FN <- cm["Low Stage",  "High Stage"] %||% 0

  results <- rbind(results, data.frame(
    seed = i,
    accuracy = (TP + TN)/(TP+TN+FP+FN),
    sensitivity = if((TP+FN)>0) TP/(TP+FN) else NA,
    specificity = if((TN+FP)>0) TN/(TN+FP) else NA,
    precision = if((TP+FP)>0) TP/(TP+FP) else NA
  ))
}

# Average metrics
colMeans(results[,-1], na.rm = TRUE)
```

```
##    accuracy sensitivity specificity   precision
##   0.4197531   0.9555556   0.2666667   0.2755362
```

Because Naive Bayes tends to produce extremely high or low probabilities on this sparse mutation data, a very high decision threshold (0.999) was used for calling a tumor “High Stage”. This biases the classifier toward catching almost all truly high-stage tumors (sensitivity 94%), but at the cost of low specificity (32%) and modest overall accuracy (46%).

Could be a valid model because it rarely misses the high

SECTION: 20

```
library(dplyr)
library(e1071)
```

```r
# --------------------------------
# 1. Gene list for the model
# --------------------------------

gene_freq <- rowSums(mut_matrix)
genes_for_model_2 <- names(sort(gene_freq, decreasing = TRUE))[1:30]

# ensure genes exist in mut_matrix
genes_for_model_2 <- intersect(genes_for_model_2, rownames(mut_matrix))

# --------------------------------
# 2. Build patient × gene + stage data
# --------------------------------

mut_df <- t(mut_matrix[genes_for_model_2, , drop = FALSE]) %>%
  as.data.frame() %>%
  mutate(PATIENT_ID = rownames(.))

df <- mut_df %>%
  inner_join(
    clinical_stage %>% select(PATIENT_ID, Stage_lowhigh),
    by = "PATIENT_ID"
  ) %>%
  filter(!is.na(Stage_lowhigh))

df$Stage_lowhigh <- factor(df$Stage_lowhigh)

# --------------------------------
# 3. Run multiple seeds with stratified splits
# --------------------------------

threshold <- 0.75
results <- data.frame(seed = integer(),
                      accuracy = numeric(),
                      sensitivity = numeric(),
                      specificity = numeric(),
                      precision = numeric(),
                      stringsAsFactors = FALSE)

`%||%` <- function(x, y) if (is.null(x) || length(x) == 0) y else x

for(i in 1:10){   # change to 50 if desired
  set.seed(i)

  low <- df %>% filter(Stage_lowhigh == "Low Stage")
  high <- df %>% filter(Stage_lowhigh == "High Stage")

  # skip iteration if too few patients in any class
  if(nrow(low) < 2 | nrow(high) < 2) next

  low_idx <- sample(seq_len(nrow(low)), size = floor(0.75 * nrow(low)))
  high_idx <- sample(seq_len(nrow(high)), size = floor(0.75 * nrow(high)))
```

```r
  train <- bind_rows(low[low_idx, ], high[high_idx, ])
  test  <- bind_rows(low[-low_idx, ], high[-high_idx, ])

  train_nb <- train[, c(genes_for_model_2, "Stage_lowhigh")]

  nb_model <- naiveBayes(Stage_lowhigh ~ ., data = train_nb)

  nb_prob <- predict(nb_model, newdata = test[, genes_for_model_2, drop = FALSE], type = "raw")

  # handle missing High Stage column
  if(!"High Stage" %in% colnames(nb_prob)){
    nb_prob <- cbind(nb_prob, "High Stage" = 0)
  }

  nb_pred <- ifelse(nb_prob[, "High Stage"] > threshold, "High Stage", "Low Stage")
  nb_pred <- factor(nb_pred, levels = levels(df$Stage_lowhigh))

  cm <- table(Predicted = nb_pred, Actual = test$Stage_lowhigh)

  TP <- cm["High Stage", "High Stage"] %||% 0
  TN <- cm["Low Stage",  "Low Stage"]  %||% 0
  FP <- cm["High Stage", "Low Stage"]  %||% 0
  FN <- cm["Low Stage",  "High Stage"] %||% 0

  results <- rbind(results, data.frame(
    seed = i,
    accuracy = (TP + TN)/(TP+TN+FP+FN),
    sensitivity = if((TP+FN)>0) TP/(TP+FN) else NA,
    specificity = if((TN+FP)>0) TN/(TN+FP) else NA,
    precision = if((TP+FP)>0) TP/(TP+FP) else NA
  ))
}

# --------------------------------
# 4. Average metrics across runs
# --------------------------------

summary_stats <- colMeans(results[,-1], na.rm = TRUE)
results
```

```
##    seed  accuracy sensitivity specificity precision
## 1     1 0.4938272   0.2222222   0.5714286 0.1290323
## 2     2 0.5925926   0.2777778   0.6825397 0.2000000
## 3     3 0.6296296   0.0000000   0.8095238 0.0000000
## 4     4 0.6419753   0.1111111   0.7936508 0.1333333
## 5     5 0.4320988   0.2222222   0.4920635 0.1111111
## 6     6 0.6172840   0.1111111   0.7619048 0.1176471
## 7     7 0.3209877   0.6666667   0.2222222 0.1967213
## 8     8 0.6172840   0.1666667   0.7460317 0.1578947
## 9     9 0.3580247   0.7222222   0.2539683 0.2166667
## 10   10 0.2962963   0.7777778   0.1587302 0.2089552
```

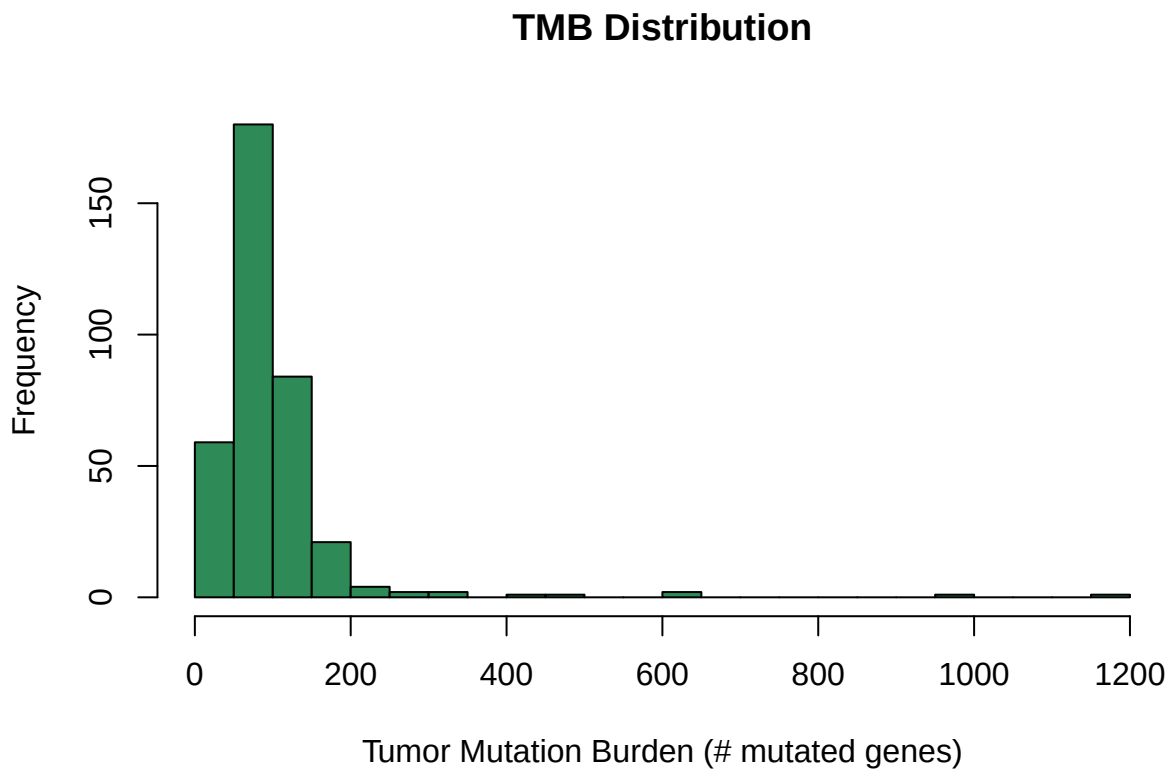
```
summary_stats
```

```
##    accuracy sensitivity specificity   precision
##   0.5000000   0.3277778   0.5492063   0.1471362
```

Section 21

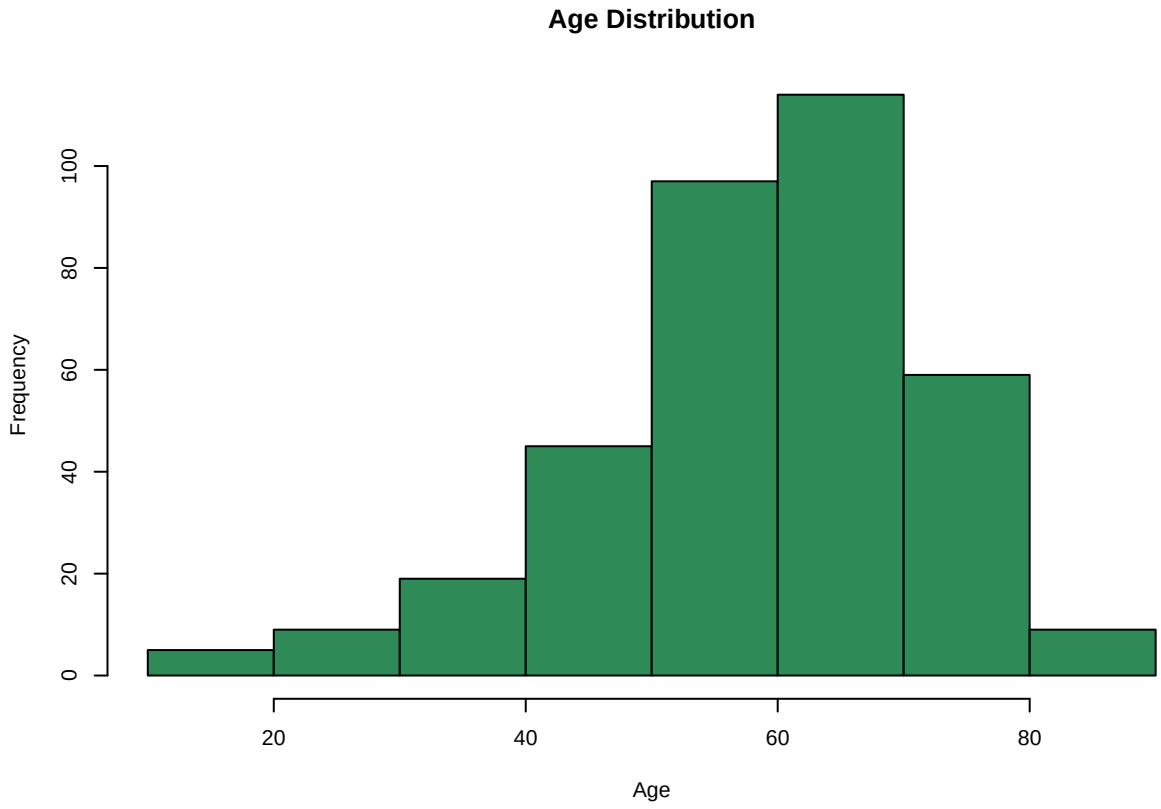
```
clinical <- read.delim("data_clinical_patient.txt", header = TRUE,
                       sep = "\t", stringsAsFactors = FALSE)
mutations <- read.delim("data_mutations.txt", header = TRUE,
                       sep = "\t", stringsAsFactors = FALSE)


clinical$Patient_ID <- clinical$X.Patient.Identifier
mutations$Patient_ID <- substr(mutations$Tumor_Sample_Barcode, 1, 12)

common_ids <- intersect(clinical$Patient_ID, mutations$Patient_ID)
clinical <- clinical[clinical$Patient_ID %in% common_ids, ]
mutations <- mutations[mutations$Patient_ID %in% common_ids, ]

nonsynonymous <- c(
  "Missense_Mutation", "Nonsense_Mutation",
  "Frame_Shift_Del", "Frame_Shift_Ins",
  "In_Frame_Del", "In_Frame_Ins",
  "Splice_Site", "Splice_Region", "Nonstop_Mutation",
  "Translation_Start_Site"
)
mut_filt <- mutations[mutations$Variant_Classification %in% nonsynonymous, ]

genes <- unique(mut_filt$Hugo_Symbol)
patients <- unique(mut_filt$Patient_ID)

mut_matrix <- matrix(0, nrow = length(genes), ncol = length(patients))
rownames(mut_matrix) <- genes
colnames(mut_matrix) <- patients

for (i in seq_len(nrow(mut_filt))) {
    g <- mut_filt$Hugo_Symbol[i]
    p <- mut_filt$Patient_ID[i]
    mut_matrix[g, p] <- 1
}
parse_status <- function(x) {
    as.numeric(sub(":.*", "", x))
}

# Convert time strings ("63.72") to numeric
parse_time <- function(x) {
    as.numeric(trimws(x))
}

# Age
```

```r
clinical$Diagnosis.Age <- parse_time(clinical$Diagnosis.Age)
# Survival Times
clinical$Overall_Time  <- parse_time(clinical$Overall.Survival..Months.)
clinical$DSS_Time      <- parse_time(clinical$Months.of.disease.specific.survival)
clinical$DFS_Time      <- parse_time(clinical$Disease.Free..Months.)
clinical$PFS_Time      <- parse_time(clinical$Progress.Free.Survival..Months.)
# Survival Status
clinical$Overall_Status <- parse_status(clinical$Overall.Survival.Status)
clinical$DSS_Status     <- parse_status(clinical$Disease.specific.Survival.status)
clinical$DFS_Status     <- parse_status(clinical$Disease.Free.Status)
clinical$PFS_Status     <- parse_status(clinical$Progression.Free.Status)

tmb <- colSums(mut_matrix)
hist(tmb, breaks = 30, col = "seagreen", border = "black",
     main = "TMB Distribution",
     xlab = "Tumor Mutation Burden (# mutated genes)")
```




```r
# Create grouped clinical stage
clinical$Stage_Grouped <- clinical$Neoplasm.Disease.Stage.American.Joint.Committee.on.Cancer.Code  # st

clinical$Stage_Grouped <- ifelse(grepl("^STAGE I$", clinical$Stage), "I",
                          ifelse(grepl("^STAGE II$", clinical$Stage), "II",
                          ifelse(grepl("^STAGE III", clinical$Stage), "III",
                          ifelse(grepl("^STAGE IV", clinical$Stage), "IV",
                                 NA))))
```

```r
clinical_NAStage <- clinical[!is.na(clinical$Stage_Grouped), ]


age_data <- clinical[!is.na(clinical$Diagnosis.Age), ]
par(cex = 0.67)
hist(age_data$Diagnosis.Age,
     main = "Age Distribution",
     xlab = "Age",
     col = "seagreen", border = "black")
```



Section 22

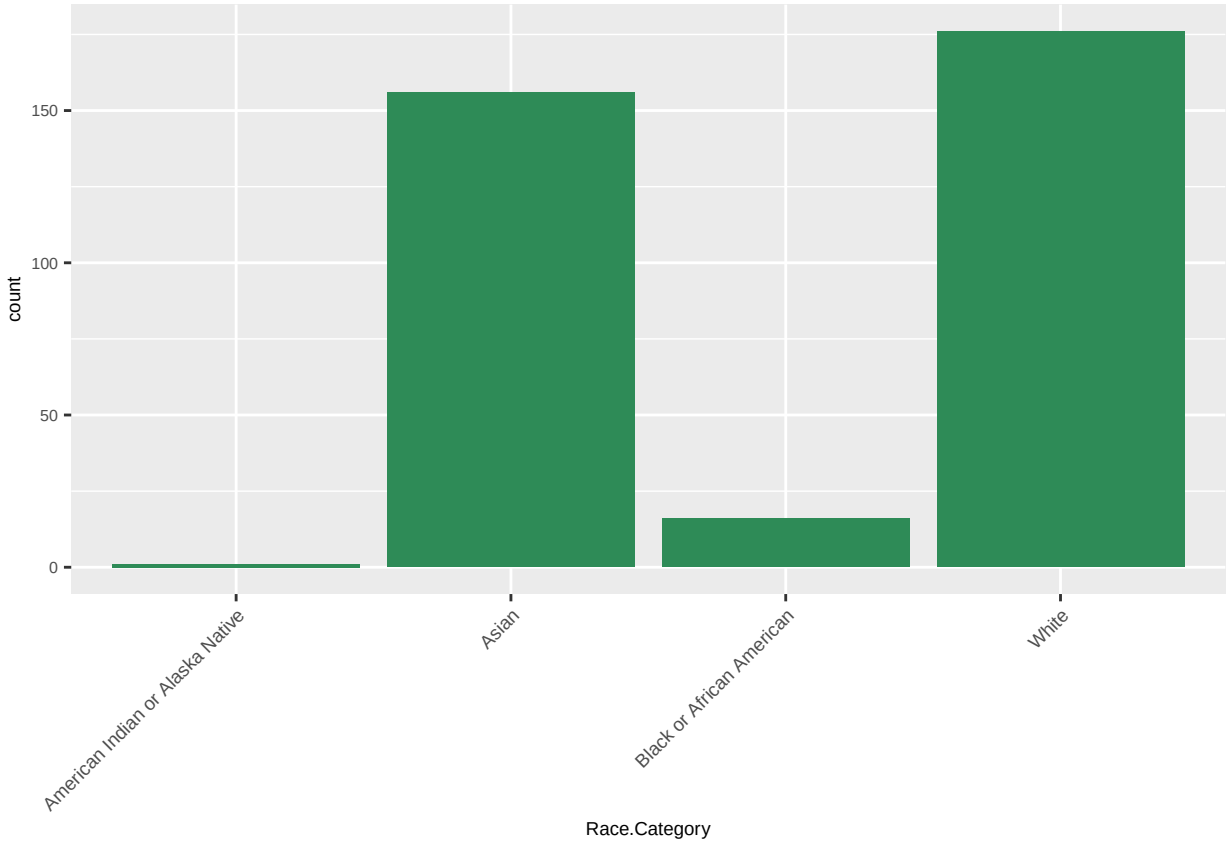
```r
table(clinical$Sex)
```

```
##
## Female   Male
##    118    240
```

```r
par(cex = 0.67)
barplot(table(clinical$Sex),
        main = "Sex Distribution",
        col = c("pink", "lightblue"))
```

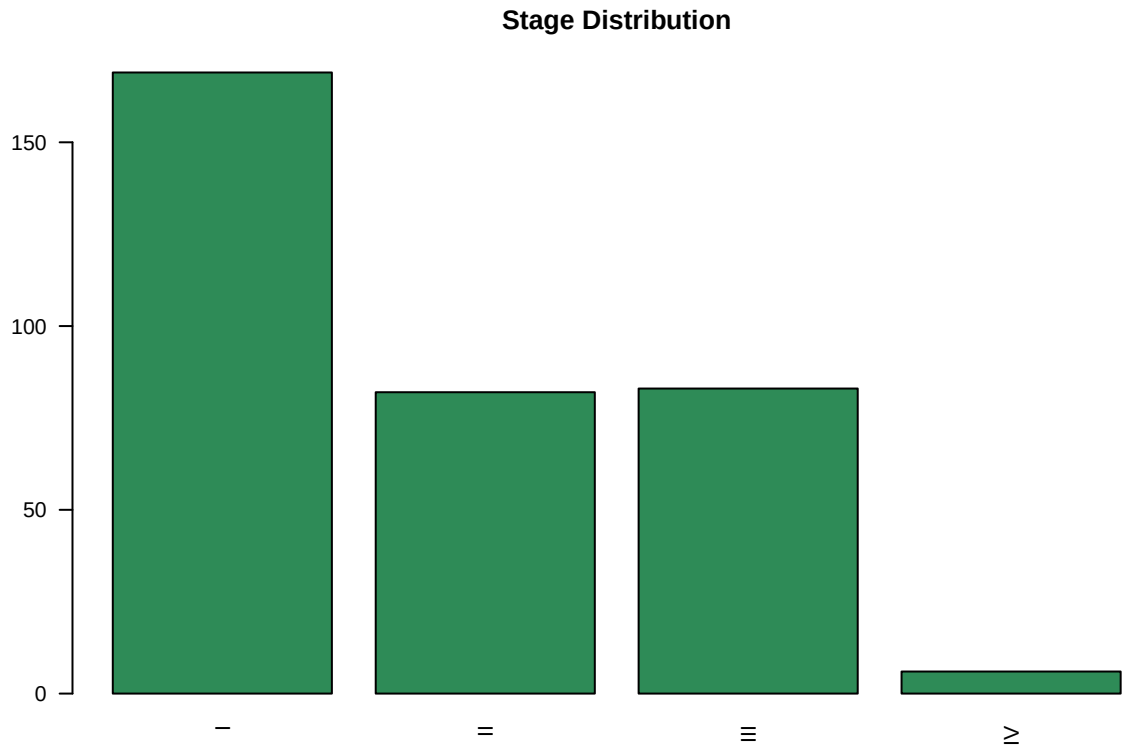
Section 23

```
ggplot(subset(clinical, Race.Category != "" & !is.na(Race.Category)),
       aes(x = Race.Category)) +
  geom_bar(fill = "seagreen") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 7.3),
    text = element_text(size = 7.3))
```



Section 24

```
par(cex = 0.67)
barplot(table(clinical$Stage_Grouped),
        main = "Stage Distribution", las = 2, col = "seagreen")
```



Section 25

```
summary(clinical$Overall_Time)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.00   11.21   19.33   26.50   35.67  120.82       1
```

```
summary(clinical$DFS_Time)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   5.638  13.085  19.571  25.085 120.821      51
```

```
summary(clinical$PFS_Time)
```

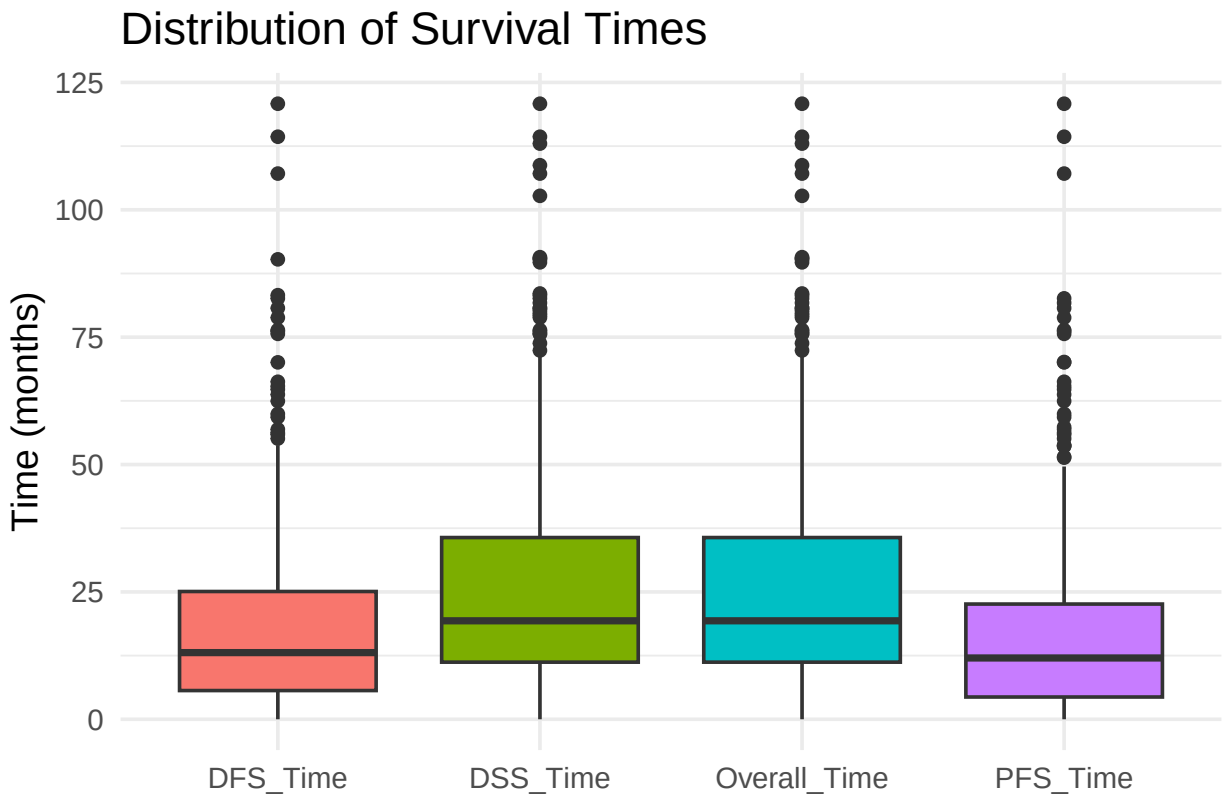
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   4.373  12.033  18.017  22.619 120.821       1
```

```
summary(clinical$DSS_Time)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.00   11.21   19.33   26.50   35.67  120.82       1
```

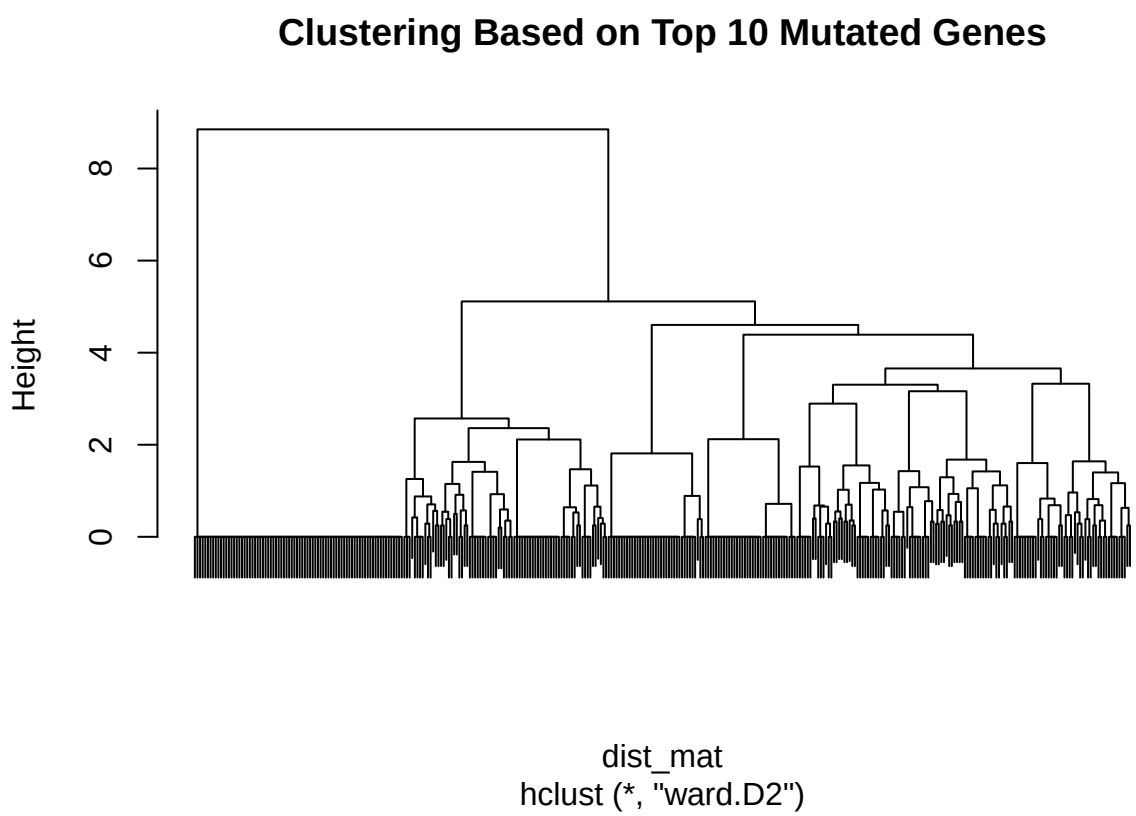
```r
# Select survival time columns
surv_df <- clinical %>%
  select(Overall_Time, DFS_Time, PFS_Time, DSS_Time)

# Convert to long format for ggplot
surv_long <- surv_df %>%
  pivot_longer(cols = everything(),
               names_to = "Endpoint",
               values_to = "Time")

# Boxplot
ggplot(surv_long, aes(x = Endpoint, y = Time, fill = Endpoint)) +
  geom_boxplot() +
  theme_minimal(base_size = 14) +
  labs(
    title = "Distribution of Survival Times",
    y = "Time (months)",
    x = ""
  ) +
  theme(legend.position = "none")
```

```
## Warning: Removed 54 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```



Section 26

```r
# Compute mutation frequency per gene
gene_counts <- rowSums(mut_matrix)

# Select top 10 most mutated genes
top_genes <- names(sort(gene_counts, decreasing = TRUE))[1:10]

# Subset mutation matrix
mut_top <- mut_matrix[top_genes, ]

# Match clinical rows to mutation matrix columns
clinical2 <- clinical[match(colnames(mut_top), clinical$Patient_ID), ]

# Check alignment
stopifnot(all(clinical2$Patient_ID == colnames(mut_top)))

dist_mat <- dist(t(mut_top), method = "binary")
hc <- hclust(dist_mat, method = "ward.D2")

plot(hc, labels = FALSE,
     main = "Clustering Based on Top 10 Mutated Genes")
```



```r
clusters <- cutree(hc, k = 2)
clinical2$Mutation_Cluster <- clusters
```

Section 27

Building contingency tables

```r
clinical2$Sex <- factor(clinical2$Sex)

clinical2$Stage_Grouped <- factor(clinical2$Stage_Grouped,
                                  levels = c("I", "II", "III", "IV"), ordered = TRUE)

clinical2$Mutation_Cluster <- factor(clinical2$Mutation_Cluster)

clinical2$Stage_Binary <- ifelse(
    clinical2$Stage_Grouped %in% c("I", "II"),
    "Early",
    "Late"
)

clinical2$Stage_Binary <- factor(clinical2$Stage_Binary, levels = c("Early", "Late"))

clinical2$Stage_3Group <- dplyr::case_when(
    clinical2$Stage_Grouped %in% c("I")  ~ "I",
    clinical2$Stage_Grouped %in% c("II") ~ "II",
    clinical2$Stage_Grouped %in% c("III", "IV") ~ "III/IV",
    TRUE ~ NA_character_
)

clinical2$Stage_3Group <- factor(
    clinical2$Stage_3Group,
    levels = c("I", "II", "III/IV")
)

clinical3 <- clinical2 %>%
    dplyr::filter(
        !is.na(Sex),
        !is.na(Stage_Grouped),
        !is.na(Stage_Binary),
        !is.na(Stage_3Group),
        !is.na(Mutation_Cluster)
    )

# Sex vs cluster
table_sex <- table(clinical3$Mutation_Cluster, clinical3$Sex)
table_sex
```

```
##
##     Female Male
##   1     74  190
##   2     36   40
```

```r
# Stage vs cluster
table_stage <- table(clinical3$Mutation_Cluster, clinical3$Stage_Grouped)
table_stage
```

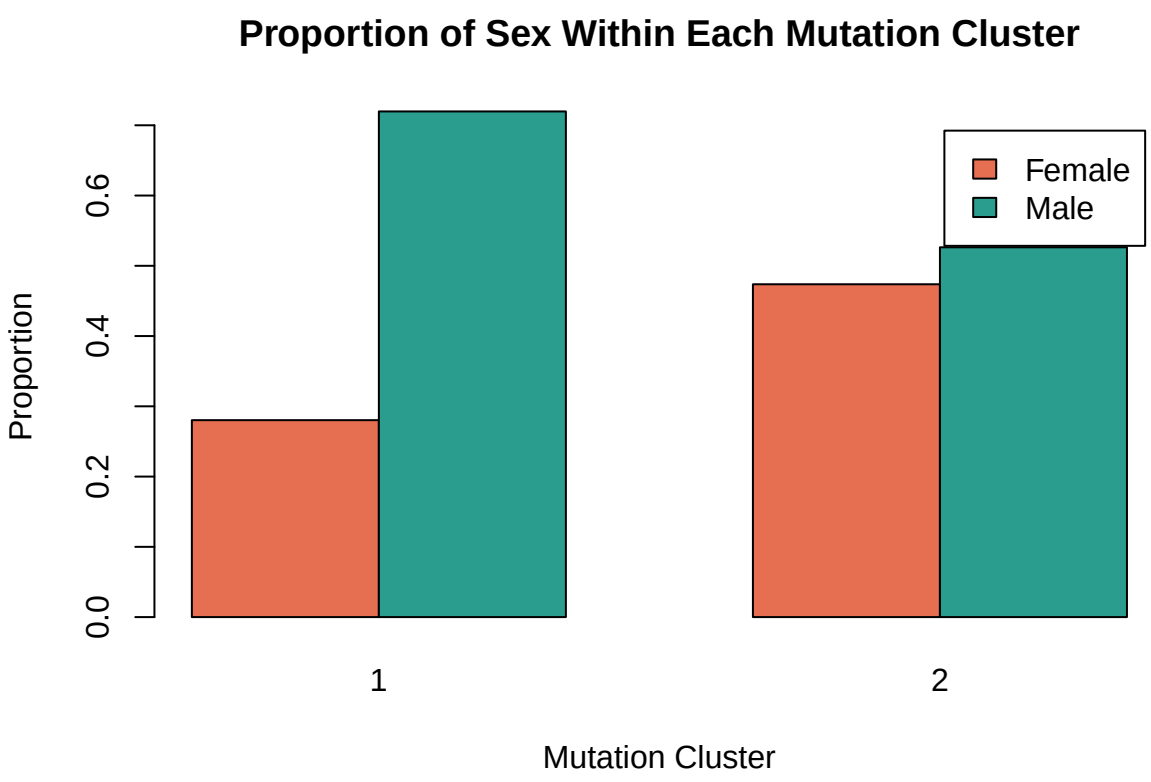
```
##
##       I  II III  IV
##   1 127  72  61   4
##   2  42  10  22   2
```

Section 28

```r
if (any(table_sex < 5)) {
    test_sex <- fisher.test(table_sex)
} else {
    test_sex <- chisq.test(table_sex)
}
test_sex
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table_sex
## X-squared = 9.2191, df = 1, p-value = 0.002395
```

```r
prop_sex <- prop.table(table_sex, margin = 1)

barplot(t(prop_sex),
        beside = TRUE,
        col = c("#E76F51", "#2A9D8F"),
        legend = TRUE,
        xlab = "Mutation Cluster",
        ylab = "Proportion",
        main = "Proportion of Sex Within Each Mutation Cluster")
```



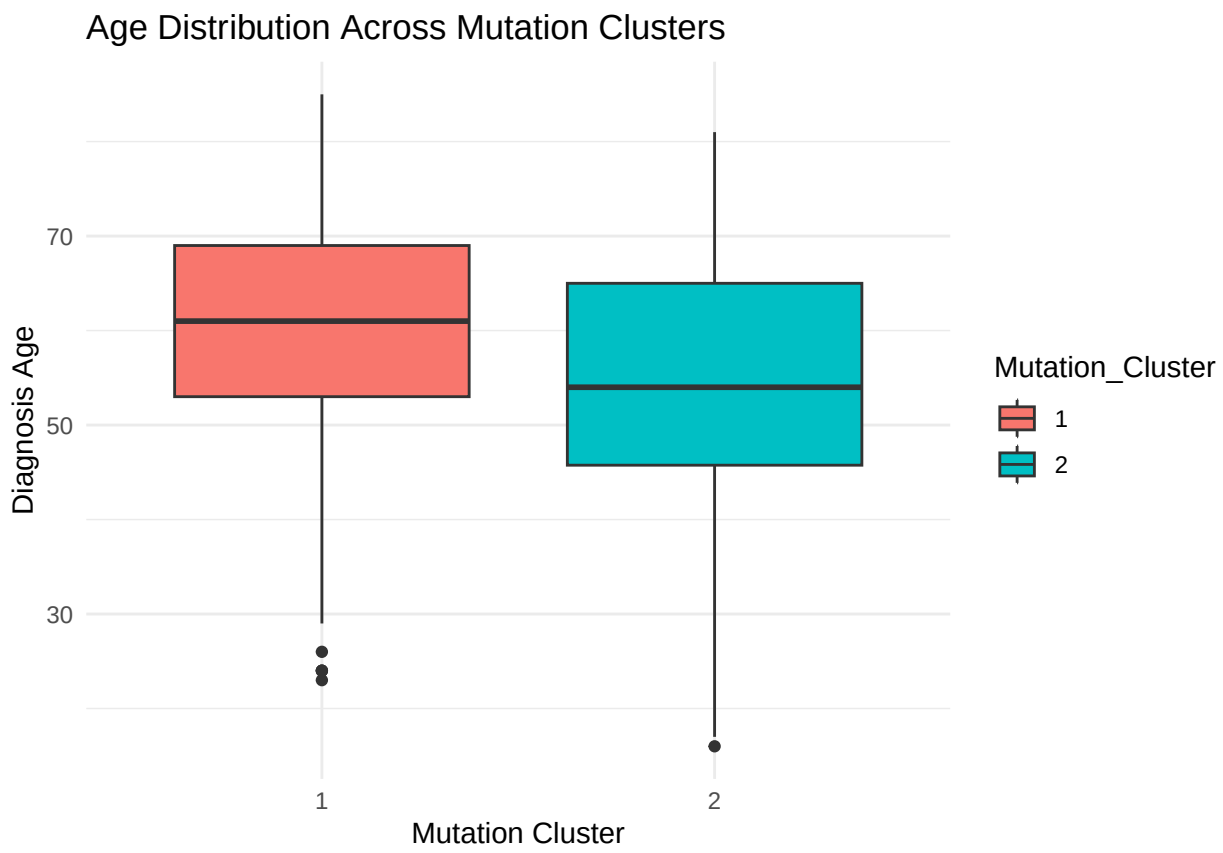
Section 29

```
clinical2$Mutation_Cluster <- factor(clinical2$Mutation_Cluster)
wilcox_age <- wilcox.test(Diagnosis.Age ~ Mutation_Cluster, data = clinical2)
wilcox_age
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  Diagnosis.Age by Mutation_Cluster
## W = 13826, p-value = 0.0007285
## alternative hypothesis: true location shift is not equal to 0
```

```
#for visual
ggplot(clinical3, aes(x = Mutation_Cluster, y = Diagnosis.Age, fill = Mutation_Cluster)) +
    geom_boxplot() +
    labs(title = "Age Distribution Across Mutation Clusters",
         x = "Mutation Cluster", y = "Diagnosis Age") +
    theme_minimal()
```

```
## Warning: Removed 1 row containing non-finite outside the scale range
## (`stat_boxplot()`).
```



Section 30

```
if (any(table_stage < 5)) {
    test_stage <- fisher.test(table_stage)
} else {
    test_stage <- chisq.test(table_stage)
}
test_stage
```

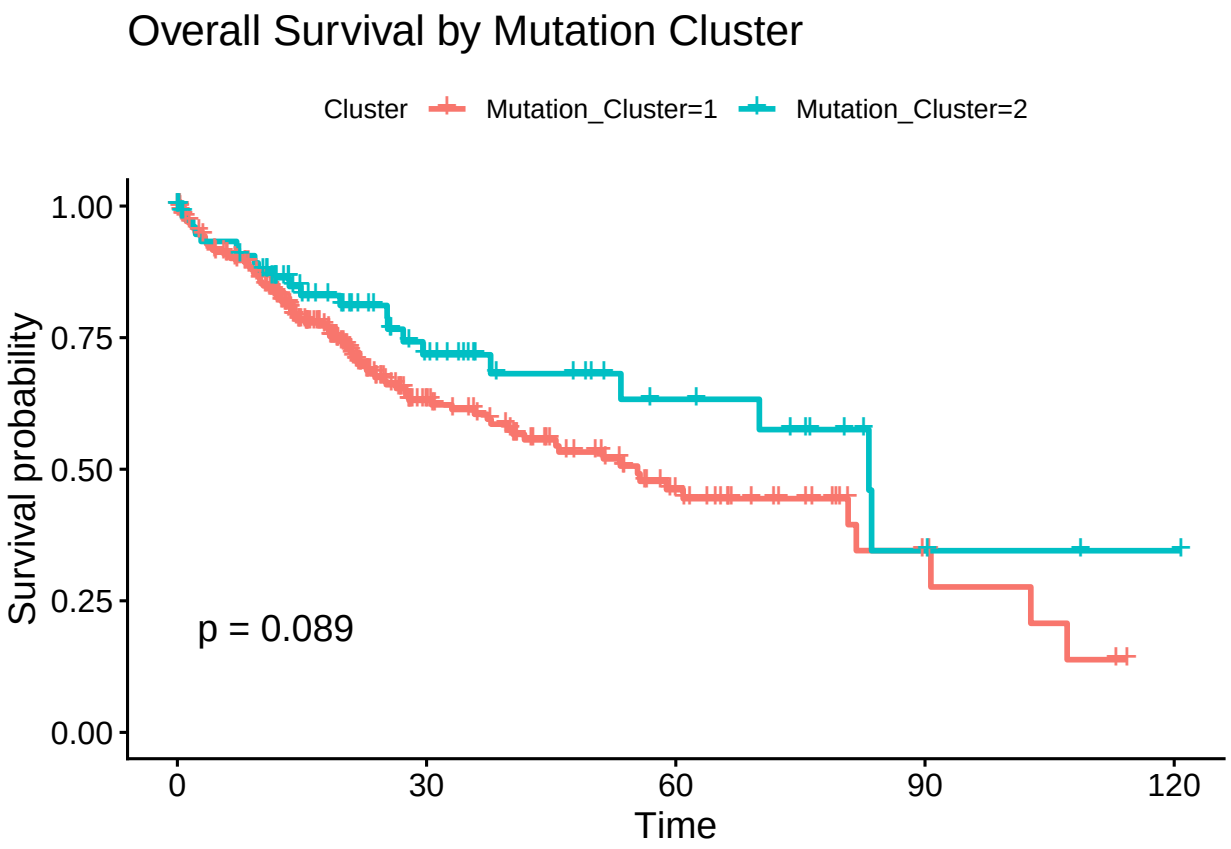
```
##
##  Fisher's Exact Test for Count Data
##
## data:  table_stage
## p-value = 0.04877
## alternative hypothesis: two.sided
```

```
ggplot(clinical3, aes(x = Mutation_Cluster, fill = Stage_Grouped)) +
    geom_bar(position = "fill") +
    labs(title = "Stage Distribution (I, II, III, IV) Across Mutation Clusters",
         x = "Mutation Cluster", y = "Proportion") +
    theme_minimal()
```

Section 31

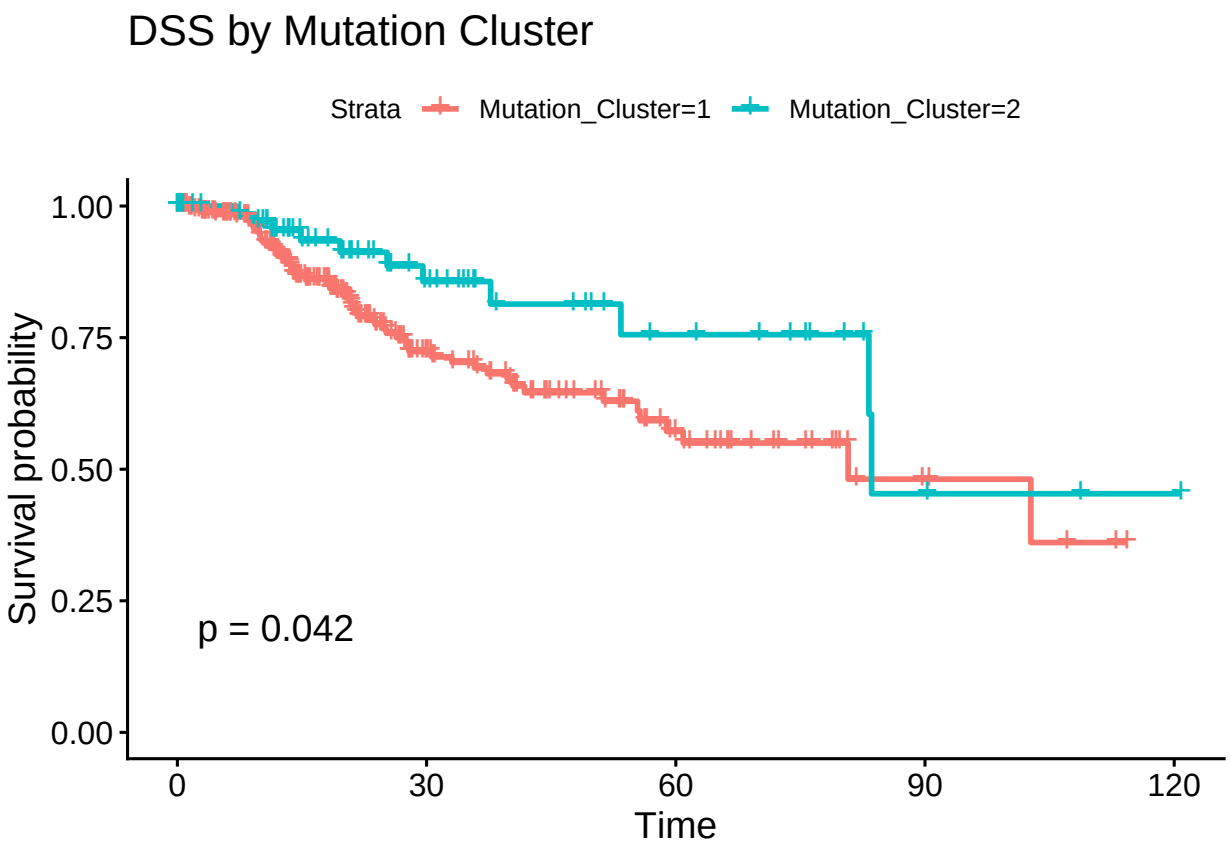
Now we are seeing if the clusters have different survival profile. We will construct Kaplan–Meier plot for each OS, DSS, DFS and PFS

```
#construct only column needed for KM plot
os_data <- clinical2[, c("Overall_Time", "Overall_Status", "Mutation_Cluster")]
os_data <- na.omit(os_data)

#fit plot
fit_os <- survfit(
  Surv(os_data$Overall_Time, os_data$Overall_Status) ~ Mutation_Cluster,
  data = os_data
)

ggsurvplot(fit_os, data = os_data,
           pval = TRUE, risk.table = FALSE,
           title = "Overall Survival by Mutation Cluster",
           legend.title = "Cluster")
```



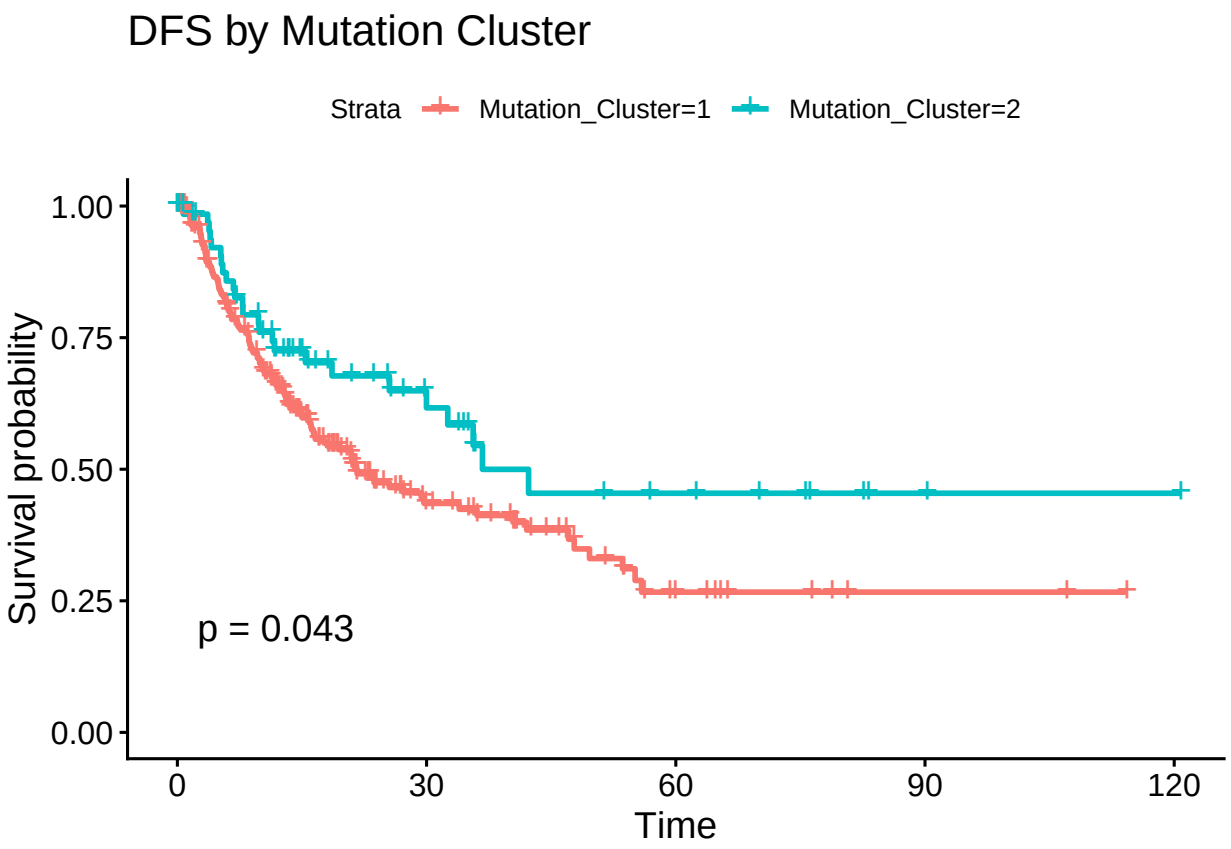
Section 32

```
dss_data <- clinical2[, c("DSS_Time", "DSS_Status", "Mutation_Cluster")]
dss_data <- na.omit(dss_data)

fit_dss <- survfit(
  Surv(DSS_Time, DSS_Status) ~ Mutation_Cluster,
  data = dss_data
)

ggsurvplot(fit_dss, data = dss_data,
           pval = TRUE, risk.table = FALSE,
           title = "DSS by Mutation Cluster")
```



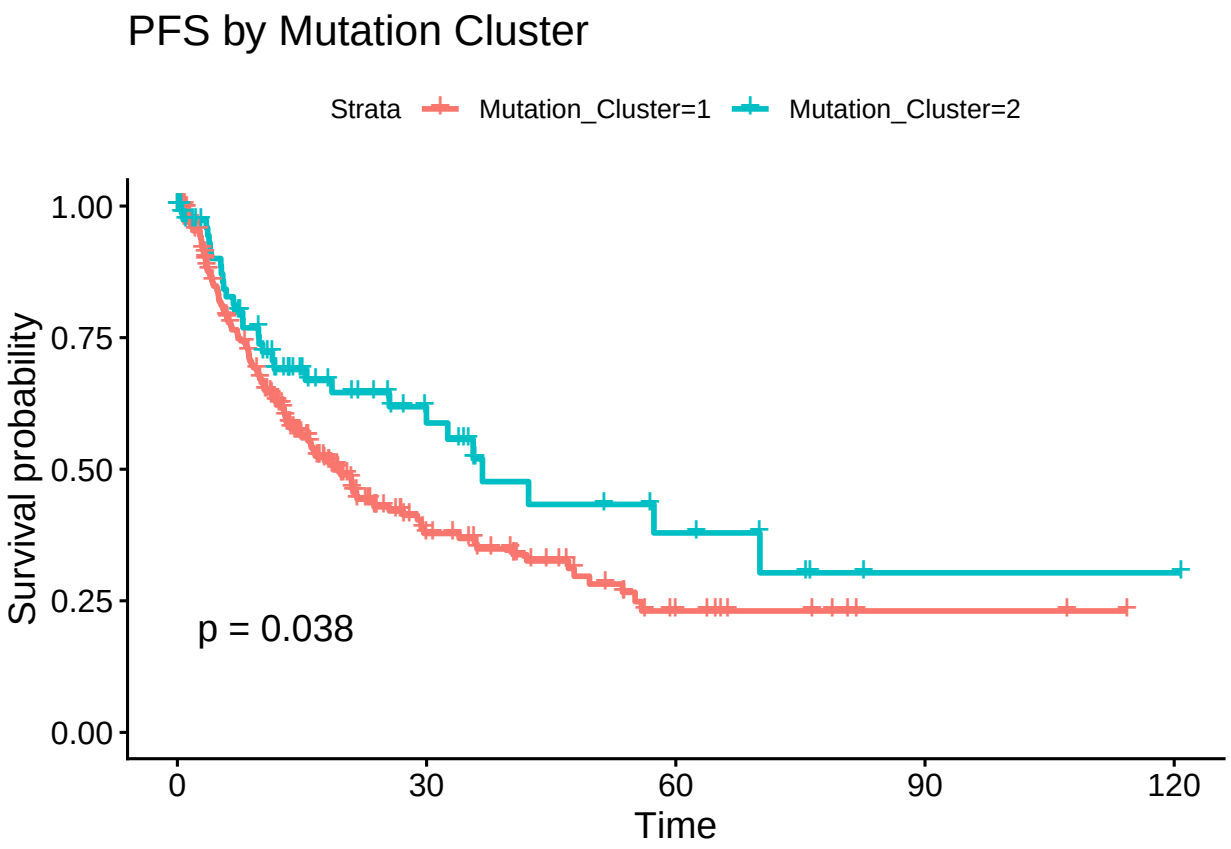
Section 33

```
dfs_data <- clinical2[, c("DFS_Time", "DFS_Status", "Mutation_Cluster")]
dfs_data <- na.omit(dfs_data)

fit_dfs <- survfit(
  Surv(DFS_Time, DFS_Status) ~ Mutation_Cluster,
  data = dfs_data
)

ggsurvplot(fit_dfs, data = dfs_data,
           pval = TRUE, risk.table = FALSE,
           title = "DFS by Mutation Cluster")
```



Section 34

```
pfs_data <- clinical2[, c("PFS_Time", "PFS_Status", "Mutation_Cluster")]
pfs_data <- na.omit(pfs_data)

fit_pfs <- survfit(
  Surv(PFS_Time, PFS_Status) ~ Mutation_Cluster,
  data = pfs_data
)

ggsurvplot(fit_pfs, data = pfs_data,
           pval = TRUE, risk.table = FALSE,
           title = "PFS by Mutation Cluster")
```

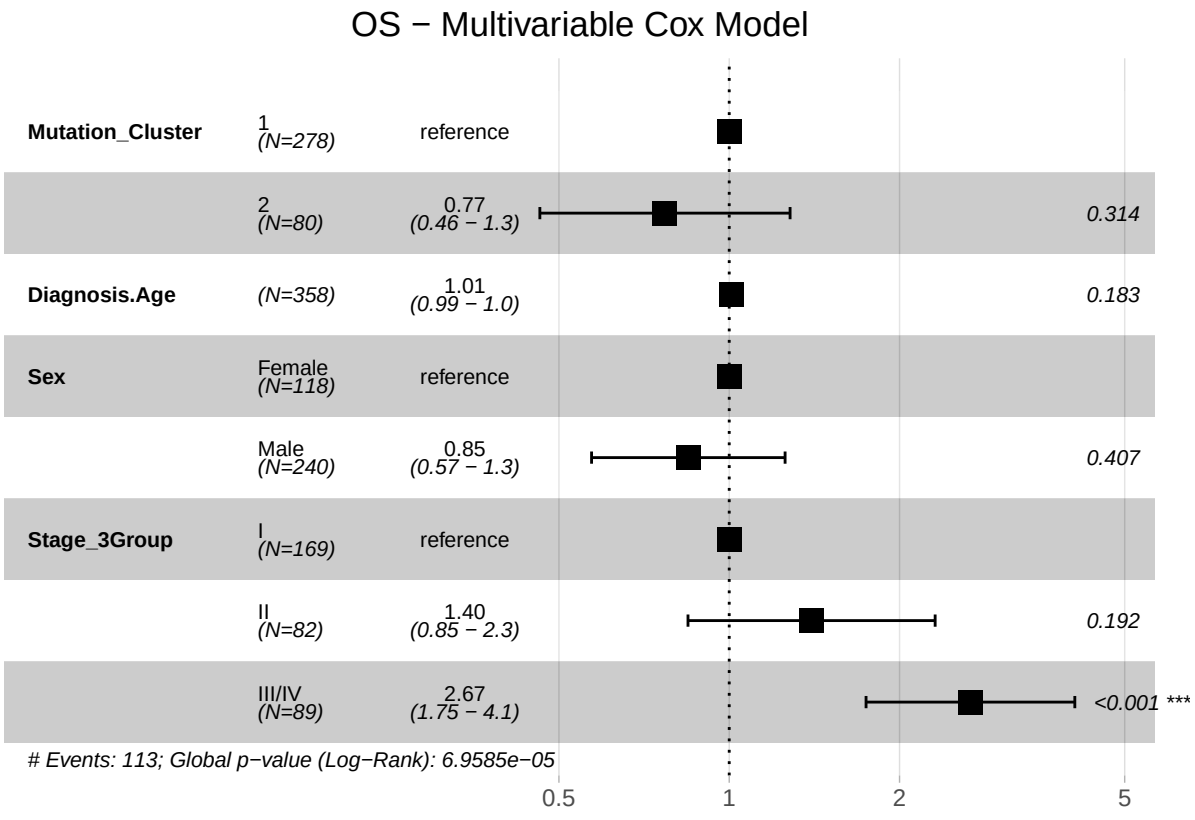


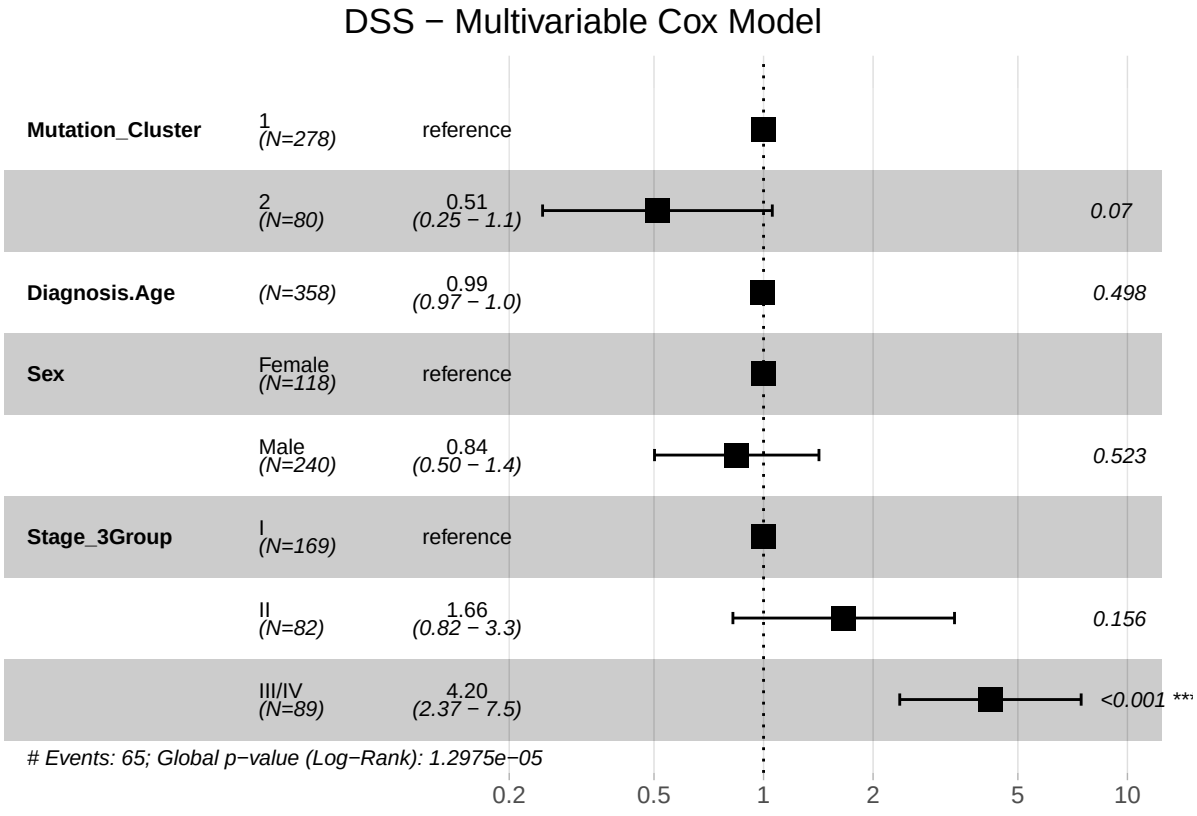
Section 35

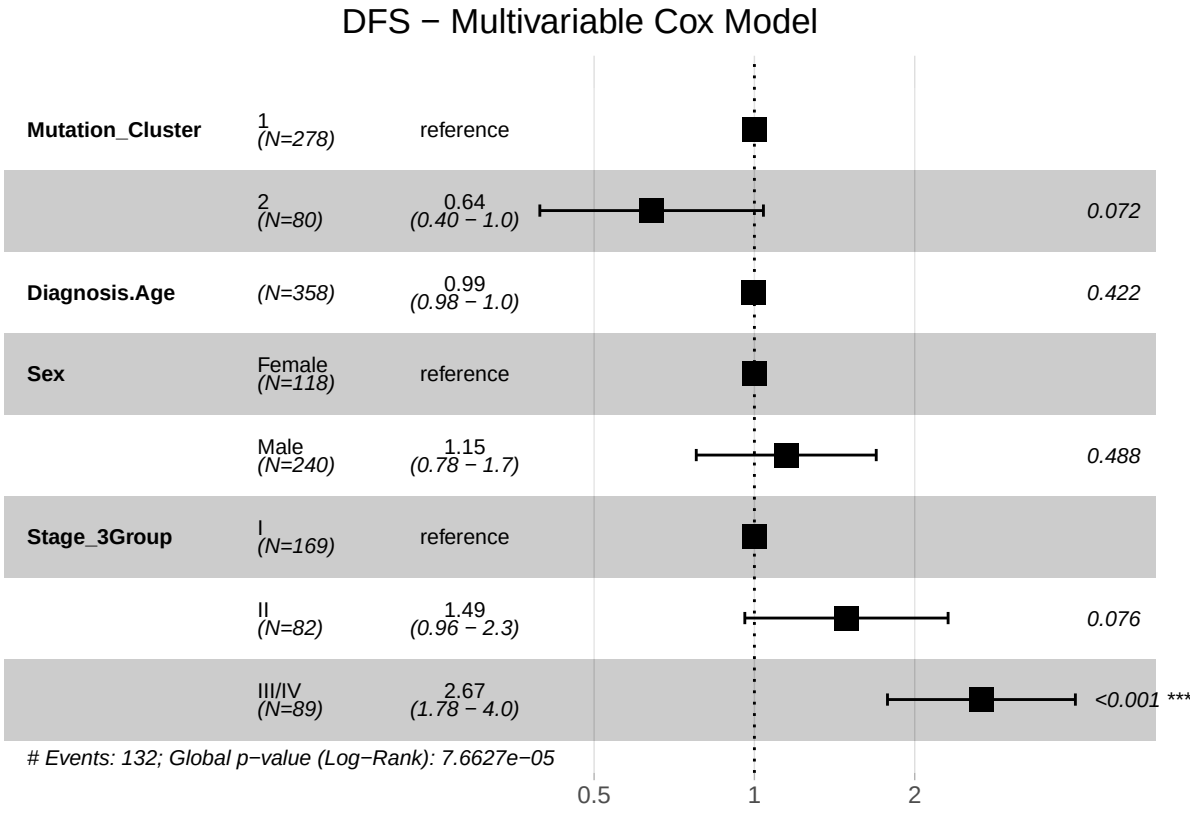
We are running adjusted, multivariable Cox regression across multiple survival endpoints to determine whether the mutation-derived clusters have independent prognostic significance.

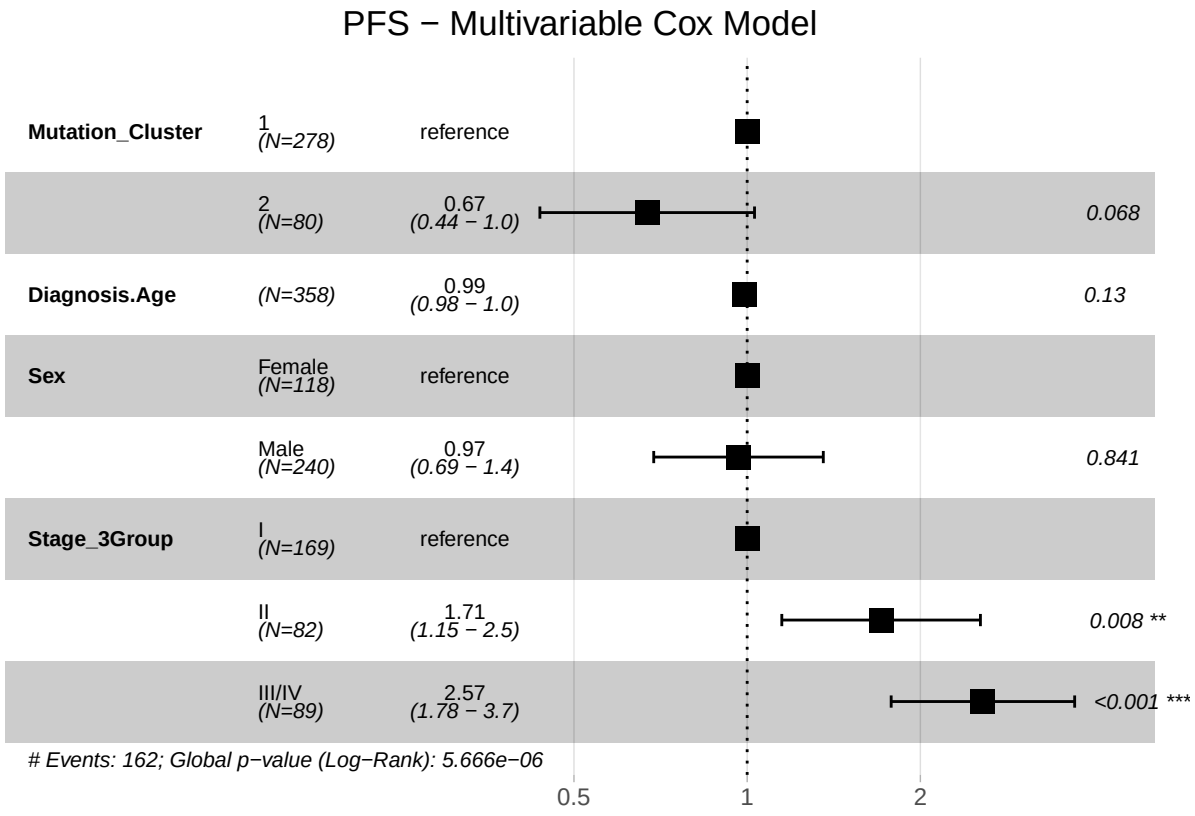
```
#Convert variables to factors
clinical2$Mutation_Cluster <- factor(clinical2$Mutation_Cluster)
clinical2$Sex <- factor(clinical2$Sex)

# choose stage variable
stage_var <- "Stage_3Group"

#Define all survival endpoints
surv_endpoints <- list(
  OS  = c("Overall_Time", "Overall_Status"),
  DSS = c("DSS_Time",     "DSS_Status"),
  DFS = c("DFS_Time",     "DFS_Status"),
  PFS = c("PFS_Time",     "PFS_Status")
)

cox_results <- list()

for (ep in names(surv_endpoints)) {

  time_col  <- surv_endpoints[[ep]][1]
```

```r
  event_col <- surv_endpoints[[ep]][2]

  # skip if any endpoint does not exist
  if (!all(c(time_col, event_col) %in% colnames(clinical2))) {
    message("Skipping ", ep, " - missing columns.")
    next
  }

  formula_string <- paste0(
    "Surv(", time_col, ", ", event_col, ") ~ ",
    "Mutation_Cluster + Diagnosis.Age + Sex + ", stage_var
  )

  fit <- coxph(as.formula(formula_string), data = clinical2)
  cox_results[[ep]] <- fit

  # optional forest plot
  print(
    ggforest(fit, data = clinical2,
             main = paste0(ep, " - Multivariable Cox Model"))
  )
}
```




DSS − Multivariable Cox Model
Mutation_Cluster
1 (N=278)
reference
2 (N=80)
0.51 (0.25 − 1.1)
0.07
Diagnosis.Age
(N=358)
0.99 (0.97 − 1.0)
0.498
Sex
Female (N=118)
reference
Male (N=240)
0.84 (0.50 − 1.4)
0.523
Stage_3Group
I (N=169)
reference
II (N=82)
1.66 (0.82 − 3.3)
0.156
III/IV (N=89)
4.20 (2.37 − 7.5)
<0.001 ***
Events: 65; Global p−value (Log−Rank): 1.2975e−05
0.2
0.5
1
2
5
10


DFS − Multivariable Cox Model
Mutation_Cluster
1 (N=278)
reference
2 (N=80)
0.64 (0.40 − 1.0)
0.072
Diagnosis.Age
(N=358)
0.99 (0.98 − 1.0)
0.422
Sex
Female (N=118)
reference
Male (N=240)
1.15 (0.78 − 1.7)
0.488
Stage_3Group
I (N=169)
reference
II (N=82)
1.49 (0.96 − 2.3)
0.076
III/IV (N=89)
2.67 (1.78 − 4.0)
<0.001 ***
Events: 132; Global p−value (Log−Rank): 7.6627e−05
0.5
1
2



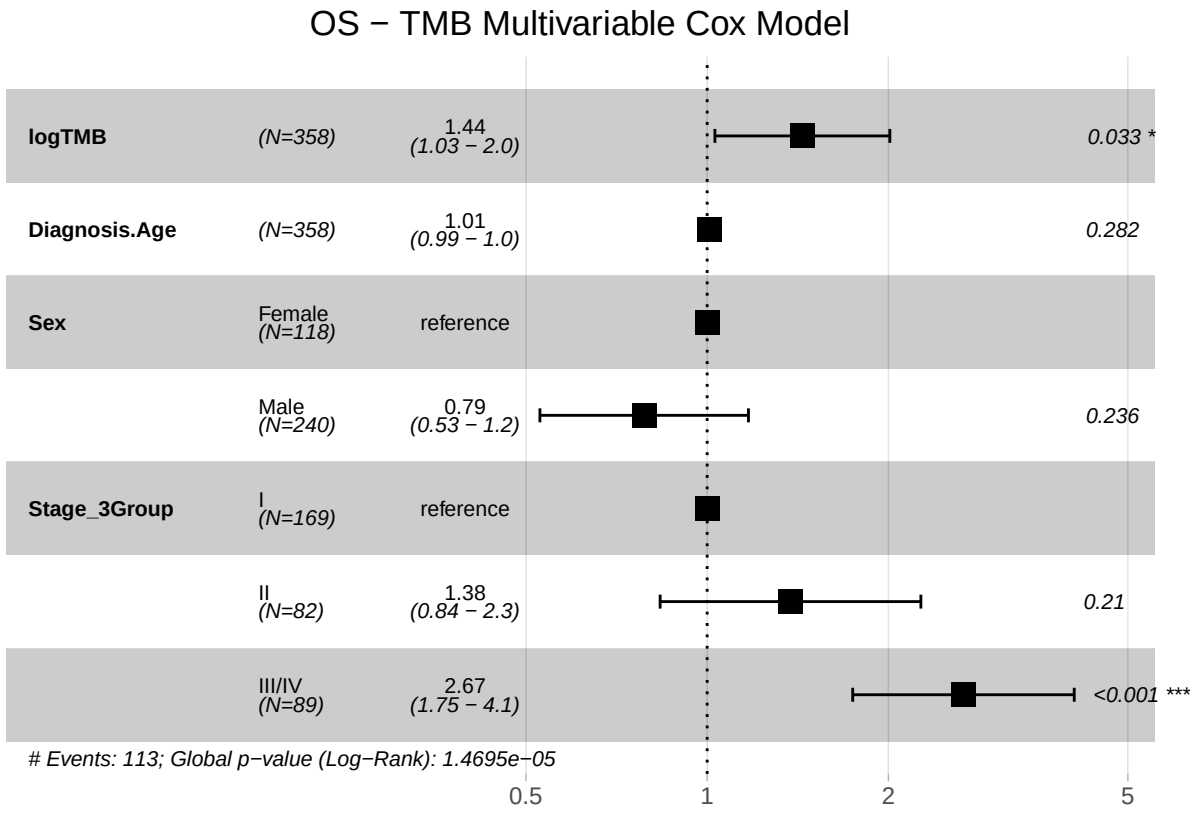
Section 36

```
# mut_matrix is gene x patient with 0/1 mutation presence
# TMB = number of mutated genes per patient
clinical2$TMB <- colSums(mut_matrix)
clinical2$logTMB <- log1p(clinical2$TMB)    # handles TMB=0 safely


clinical2$Mutation_Cluster <- factor(clinical2$Mutation_Cluster)
clinical2$Sex <- factor(clinical2$Sex)

stage_var <- "Stage_3Group"


surv_endpoints <- list(
  OS  = c("Overall_Time", "Overall_Status"),
  DSS = c("DSS_Time",     "DSS_Status"),
  DFS = c("DFS_Time",     "DFS_Status"),
  PFS = c("PFS_Time",     "PFS_Status")
)
cox_TMB_results <- list()
cox_TMB_results_adj <- list()

for (ep in names(surv_endpoints)) {
```

```
  time_col  <- surv_endpoints[[ep]][1]
  event_col <- surv_endpoints[[ep]][2]

  # skip if missing columns
  if (!all(c(time_col, event_col) %in% names(clinical2))) {
    message("Skipping ", ep, ": missing time/event")
    next
  }

  cat("\n==================================================\n")
  cat("               TMB Survival Analysis -", ep, "\n")
  cat("==================================================\n")

  # ----------------------------
  # 1. Univariate Cox: TMB only
  # ----------------------------
  form_uni <- as.formula(
    paste0("Surv(", time_col, ", ", event_col, ") ~ logTMB")
  )

  fit_uni <- coxph(form_uni, data = clinical2)
  cox_TMB_results[[ep]] <- fit_uni

  cat("\n--- Univariate Cox (TMB only) ---\n")
  print(summary(fit_uni))

  # ----------------------------
  # 2. Multivariable Cox: TMB + age + sex + stage
  # ----------------------------
  form_multi <- as.formula(
    paste0("Surv(", time_col, ", ", event_col, ") ~ ",
           "logTMB + Diagnosis.Age + Sex + ", stage_var)
  )

  fit_multi <- coxph(form_multi, data = clinical2)
  cox_TMB_results_adj[[ep]] <- fit_multi

  # Optional: forest plot
  print(
    ggforest(fit_multi,
             data = clinical2,
             main = paste0(ep, " - TMB Multivariable Cox Model"))
  )
}
```

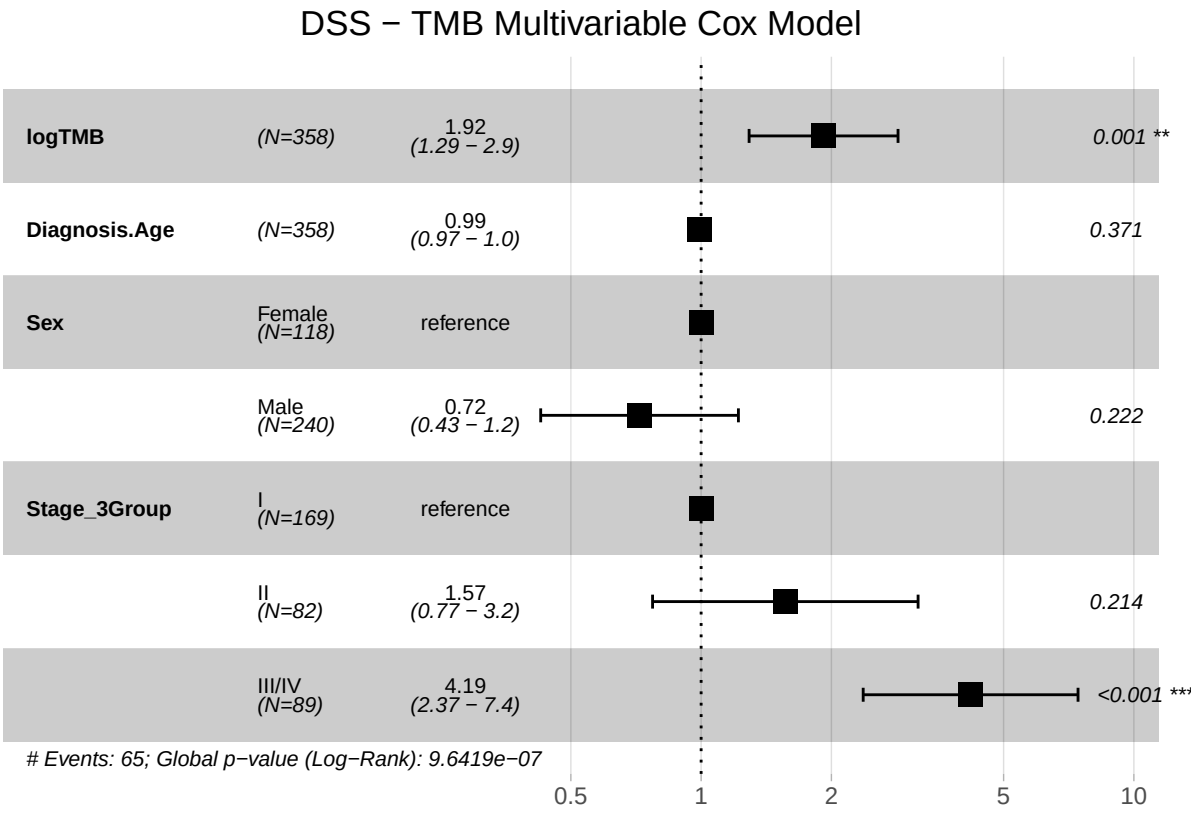
```
##
## ==================================================
##                TMB Survival Analysis - OS
## ==================================================
##
## --- Univariate Cox (TMB only) ---
## Call:
## coxph(formula = form_uni, data = clinical2)
```

```
##
##   n= 357, number of events= 122
##    (1 observation deleted due to missingness)
##
##          coef exp(coef) se(coef)     z Pr(>|z|)
## logTMB 0.3929    1.4813   0.1600 2.455   0.0141 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##        exp(coef) exp(-coef) lower .95 upper .95
## logTMB     1.481     0.6751     1.082     2.027
##
## Concordance= 0.579  (se = 0.028 )
## Likelihood ratio test= 5.94  on 1 df,   p=0.01
## Wald test            = 6.03  on 1 df,   p=0.01
## Score (logrank) test = 5.71  on 1 df,   p=0.02
```



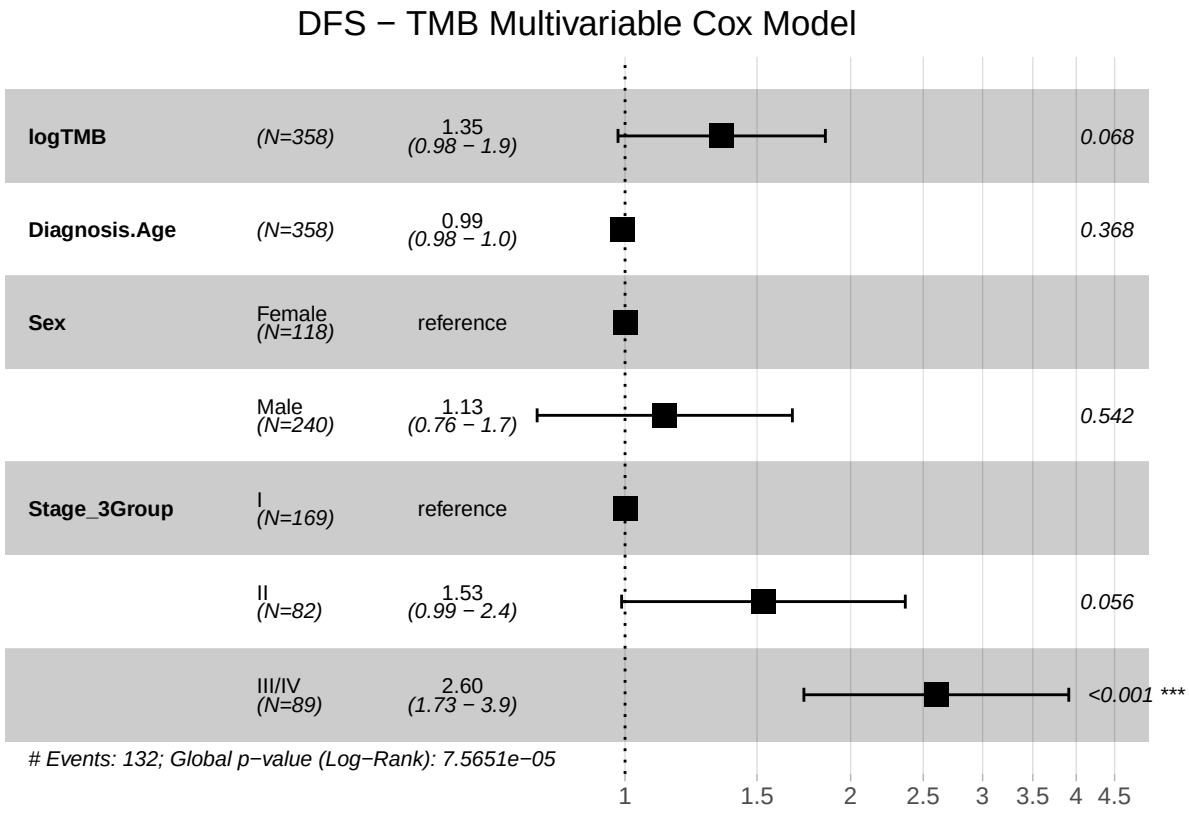

```
##
## ================================================
##               TMB Survival Analysis - DSS
## ================================================
##
## --- Univariate Cox (TMB only) ---
## Call:
## coxph(formula = form_uni, data = clinical2)
```

```
##
##   n= 348, number of events= 74
##    (10 observations deleted due to missingness)
##
##          coef exp(coef) se(coef)     z Pr(>|z|)
## logTMB 0.5949    1.8128   0.2048 2.904  0.00368 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##        exp(coef) exp(-coef) lower .95 upper .95
## logTMB     1.813     0.5516     1.213     2.708
##
## Concordance= 0.602  (se = 0.037 )
## Likelihood ratio test= 7.98  on 1 df,   p=0.005
## Wald test            = 8.43  on 1 df,   p=0.004
## Score (logrank) test = 7.56  on 1 df,   p=0.006
```




```
##
## ================================================
##               TMB Survival Analysis - DFS
## ================================================
##
## --- Univariate Cox (TMB only) ---
## Call:
## coxph(formula = form_uni, data = clinical2)
```

```
##
##   n= 307, number of events= 139
##    (51 observations deleted due to missingness)
##
##          coef exp(coef) se(coef)     z Pr(>|z|)
## logTMB 0.3329    1.3950   0.1540 2.162   0.0306 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##        exp(coef) exp(-coef) lower .95 upper .95
## logTMB     1.395     0.7168     1.032     1.887
##
## Concordance= 0.54  (se = 0.028 )
## Likelihood ratio test= 4.87  on 1 df,   p=0.03
## Wald test            = 4.67  on 1 df,   p=0.03
## Score (logrank) test = 4.49  on 1 df,   p=0.03
```




```
##
## ================================================
##               TMB Survival Analysis - PFS
## ================================================
##
## --- Univariate Cox (TMB only) ---
## Call:
## coxph(formula = form_uni, data = clinical2)
```

```
##
##   n= 357, number of events= 174
##    (1 observation deleted due to missingness)
##
##          coef exp(coef) se(coef)     z Pr(>|z|)
## logTMB 0.2881    1.3339   0.1393 2.068   0.0386 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##        exp(coef) exp(-coef) lower .95 upper .95
## logTMB     1.334     0.7497     1.015     1.753
##
## Concordance= 0.537  (se = 0.025 )
## Likelihood ratio test= 4.44  on 1 df,   p=0.04
## Wald test            = 4.28  on 1 df,   p=0.04
## Score (logrank) test = 4.15  on 1 df,   p=0.04
```