STAT 115 Lab 11

TCGA, Tumor Subtypes, Methylation, Survival Analysis

Andy Shi

April 10-12, 2018

```
library(FirebrowseR)
library(bladderbatch)
library(limma)
library(sva)
library(dplyr)
library(survival)
library(glmnet)
library(ggplot2)
```

Overview of Homework 6

This week, we will cover:

- Part I: Accessing Data from TCGA
- Part II: Tumor Subtype Analysis
  - LIMMA to analyze differential gene expression and methylation
  - K-Means clustering
  - PCA for visualization
- Part III: Survival Analysis
  - Kaplan-Meier curves
  - Cox proportional hazards model
  - Gene signatures

Next week:

- Part IV: Mutation Analysis
- Part V: Precision Medicine
- Part VI: CRISPR Screens
  - New topic this year
  - MAGeCK
  - Having some issues with the data right now, hopefully will be resolved in the next few days.

Part I: Accessing Data from TCGA

Q1. TCGA Website

- TCGA's website contains raw data that you can download
- Should be fairly straightforward, involves searching on the provided website.

Q2. Broad Firehose

- Contains processed data that you can download and analyze
- Access using firebrowse
- R API: FirebrowseR
- Code adapted from FirebrowseR vignette: https://github.com/mariodeng/FirebrowseR
- Let's download all breast cancer patients' clinical data.

```
# download all available cohorts
cohorts <- Metadata.Cohorts(format = "csv")
# show what cohorts are available
#cohorts

# have to do this because we can only receive 150 patients at a time
all.Received <- FALSE
page.Counter <- 1
page.size <- 150
brca_pats <- list()
while(all.Received == FALSE) {
    brca_pats[[page.Counter]] <- Samples.Clinical(format = "csv",
            cohort = "BRCA", page_size = page.size, page = page.Counter)
    if(page.Counter > 1) {
        colnames(brca_pats[[page.Counter]]) <-
            colnames(brca_pats[[page.Counter-1]])
    }

    if(nrow(brca_pats[[page.Counter]]) < page.size) {
        all.Received = TRUE
    } else {
        page.Counter = page.Counter + 1
    }
}

brca_pats <- do.call(rbind, brca_pats)
dim(brca_pats)
```

```
## [1] 1097  111
```

Now, can you find out how many are alive? How about the mean and median age at initial diagnosis? Can you plot a histogram of the age at initial diagnosis?

```
# your turn
table(brca_pats$vital_status)
```

```
##
## alive  dead
##   945   152
```

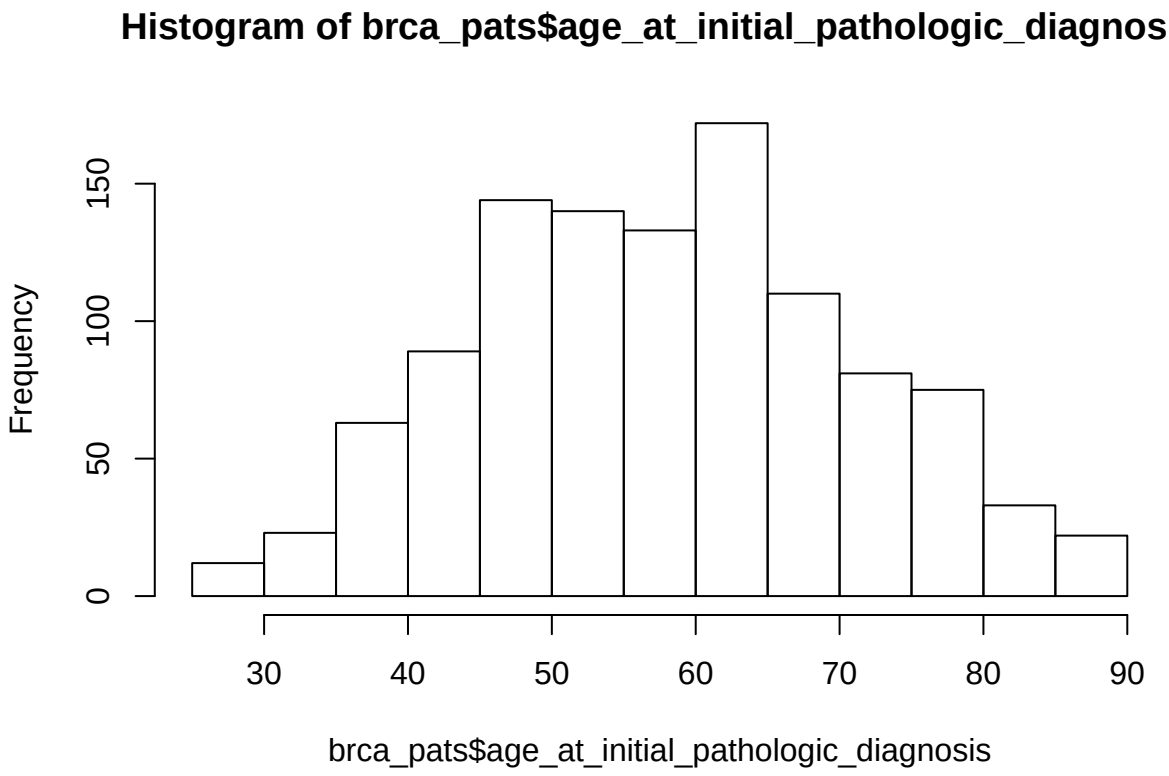
```
mean(brca_pats$age_at_initial_pathologic_diagnosis)
```

```
## [1] 58.46308
```

```r
median(brca_pats$age_at_initial_pathologic_diagnosis)
```

```
## [1] 58
```

```r
hist(brca_pats$age_at_initial_pathologic_diagnosis)
```



Part II: Tumor Subtype Analysis

- Q3 and Q4: Using LIMMA to find differentially expressed genes. Please review Lab 2.
- You can assume that we have already performed normalization (RMA) and batch effect removal (ComBat), so you can jump right in to using LIMMA.

Expression Data: Clustering and PCA

Task: using the bladder batch data, can you perform kmeans clustering (try $k = 2$ for now) on differentially expressed genes (FDR < 0.05 and log2-fold-change > 2), and then plot the result on a PCA plot, with the color of each point denoting its cluster and the shape denoting its cancer status?

```r
set.seed(20180410)
# I am running ComBat because this data has batch effect, but you don't
# need this for your HW
data(bladderdata)
pheno <- pData(bladderEset)
pheno$hasCancer <- as.numeric(pheno$cancer == "Cancer")
edata <- exprs(bladderEset)
```

```
model <- model.matrix(~hasCancer, data = pheno)
combat_edata <- ComBat(dat = edata, batch = pheno$batch, mod = model)
```

```
## Found5batches

## Adjusting for1covariate(s) or covariate level(s)

## Standardizing Data across genes

## Fitting L/S model and finding priors

## Finding parametric adjustments

## Adjusting the Data
```

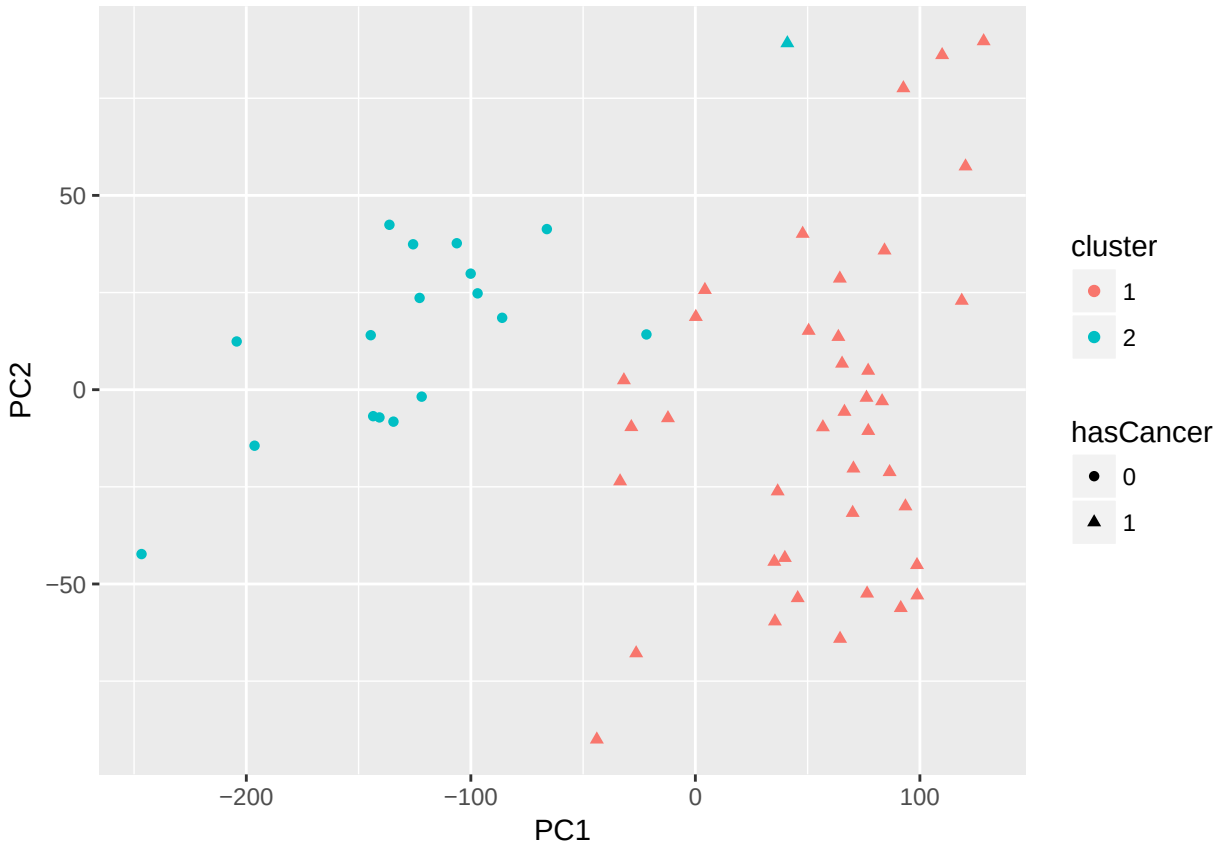
```
# run LIMMA to get the top genes (use data after running ComBat)
# your turn
fit <- lmFit(combat_edata, model)
fit <- eBayes(fit)
topgenes <- topTable(fit, coef = "hasCancer", p.value = 0.05, lfc = 2,
                     number = Inf)

# run kmeans clustering on top genes
# your turn
kmeans_res <- kmeans(t(combat_edata[rownames(topgenes),]),
                     2, nstart = 10, iter.max = 100)

# run PCA
# your turn
pca_raw <- prcomp(t(combat_edata), center = TRUE, scale. = TRUE)

# assemble the data
# your turn
edata_pc_df <- as.data.frame(pca_raw$x)
edata_pc_df$cluster <- as.factor(kmeans_res$cluster)
edata_pc_df$hasCancer <- as.factor(pheno$hasCancer)

# draw the plot
# your turn
ggplot(edata_pc_df, aes(x = PC1, y = PC2, color = cluster, shape =
                        hasCancer)) + geom_point()
```



Methylation Data

- Logit-transform to map from $[0,1] \to (-\infty, \infty)$. Then analysis proceeds in the same way as microarray data with LIMMA.

Part III: Survival Analysis

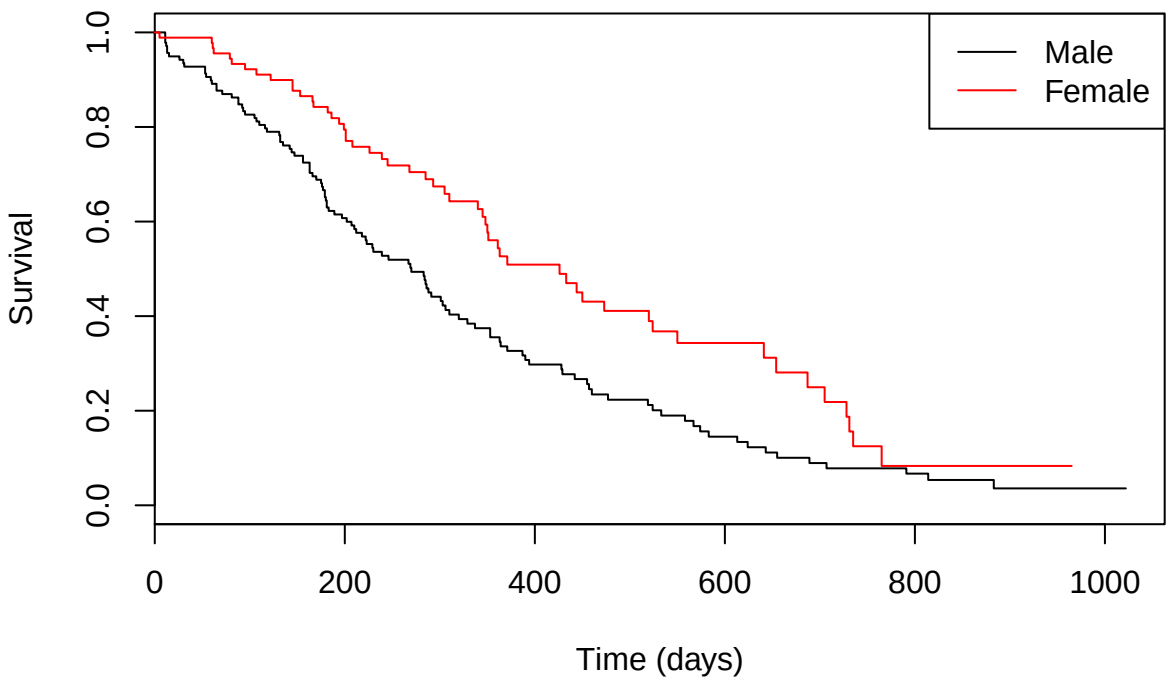
- T_i: the time to event for the ith individual.
- C_i: the corresponding censoring time.
- We observe $Y_i = \min(T_i, C_i)$ and $\delta_i = I(T_i \leq C_i)$ (i.e. $\delta_i = 1$ if $T_i \leq C_i$ and $\delta_i = 0$ if $T_i > C_i$).
- We also have predictors X_i for each individual.

Kaplan-Meier Curve

- A way to estimate the *survival function* $P(T_i > t)$ from our observed data, taking into account the censoring.
- We pass in Y_i and δ_i into the `Surv` function.

```
# data wrangling to make this easier
lung2 <- lung
# 1 = died, 0 = still alive at last observation
lung2$death <- lung$status - 1
```

```
km_fit <- survfit(Surv(time, death) ~ sex, data = lung2)
plot(km_fit, col = c("black", "red"), ylab = "Survival",
     xlab = "Time (days)")
legend("topright", legend = c("Male", "Female"), col =
       c("black","red"), lty = 1)
```



- The log-rank test compares the survival curves across the observed time frame. Significant p-value means the two curves are different.

```
survdiff(Surv(time, death) ~ sex, data = lung2)
```

```
## Call:
## survdiff(formula = Surv(time, death) ~ sex, data = lung2)
##
##         N Observed Expected (O-E)^2/E (O-E)^2/V
## sex=1 138      112     91.6      4.55      10.3
## sex=2  90       53     73.4      5.68      10.3
##
##  Chisq= 10.3  on 1 degrees of freedom, p= 0.00131
```

Cox proportional hazards model

- The hazard function $\lambda(t)$ is defined as $\lambda(t) = \lim_{\delta \to 0} \frac{1}{\delta} P(t \leq T < t + \delta | T \geq t)$.
- Interpretation: instantaneous rate at time t, given that the event has not occurred prior to time t.
- Cox proportional hazards model: $\lambda(t_i) = \lambda_0(t_i) \exp(X_1\beta_1 + \cdots + X_p\beta_p)$.
- We are only interested in the β's
- We can perform estimation and inference without specifying $\lambda_0(t_i)$. $\lambda_0(t_i)$ is the hazard when all $X_i = 0$, and is called the baseline hazard.

```r
lung2$sex <- lung2$sex - 1
cox_mod1 <- coxph(Surv(time, death) ~ sex, data = lung2)
summary(cox_mod1)
```

```
## Call:
## coxph(formula = Surv(time, death) ~ sex, data = lung2)
##
##   n= 228, number of events= 165
##
##        coef exp(coef) se(coef)      z Pr(>|z|)
## sex -0.5310    0.5880   0.1672 -3.176  0.00149 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##     exp(coef) exp(-coef) lower .95 upper .95
## sex     0.588      1.701    0.4237     0.816
##
## Concordance= 0.579  (se = 0.022 )
## Rsquare= 0.046   (max possible= 0.999 )
## Likelihood ratio test= 10.63  on 1 df,   p=0.001111
## Wald test            = 10.09  on 1 df,   p=0.001491
## Score (logrank) test = 10.33  on 1 df,   p=0.001312
```

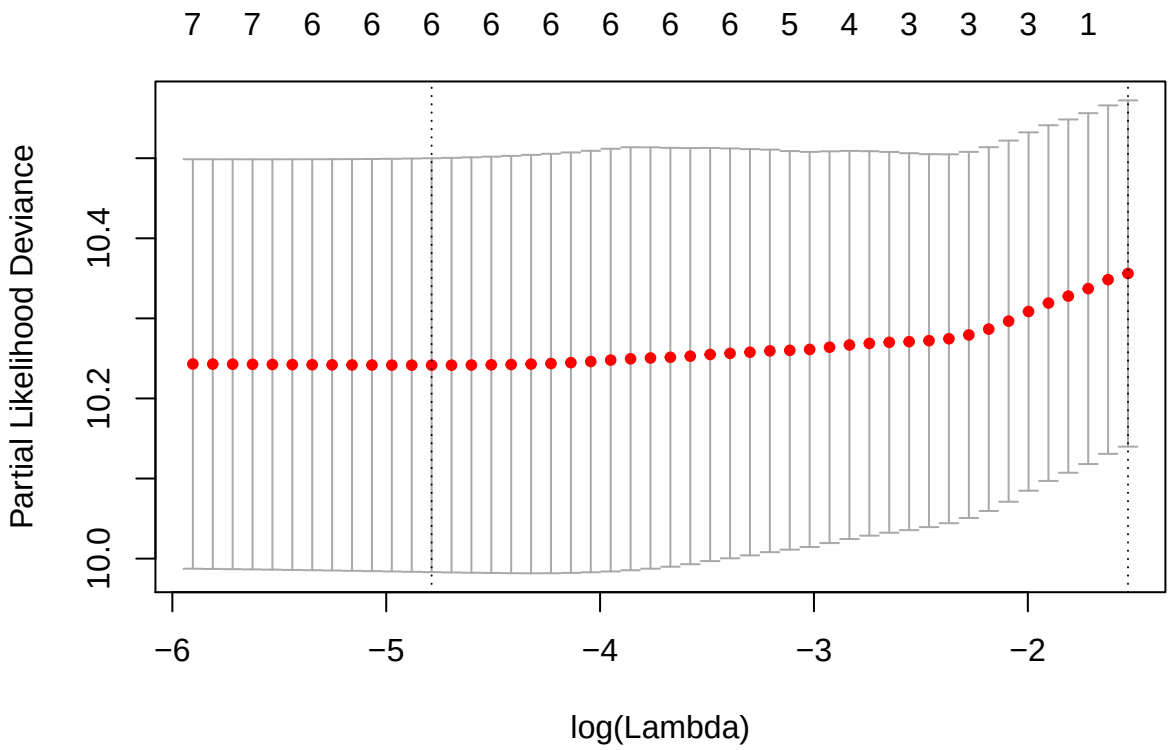
```r
cox_mod2 <- coxph(Surv(time, death) ~ sex + age + ph.ecog, data = lung2)
summary(cox_mod2)
```

```
## Call:
## coxph(formula = Surv(time, death) ~ sex + age + ph.ecog, data = lung2)
##
##   n= 227, number of events= 164
##    (1 observation deleted due to missingness)
##
##              coef exp(coef)  se(coef)      z Pr(>|z|)
## sex     -0.552612  0.575445  0.167739 -3.294 0.000986 ***
## age      0.011067  1.011128  0.009267  1.194 0.232416
## ph.ecog  0.463728  1.589991  0.113577  4.083 4.45e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##         exp(coef) exp(-coef) lower .95 upper .95
## sex        0.5754     1.7378    0.4142    0.7994
## age        1.0111     0.9890    0.9929    1.0297
## ph.ecog    1.5900     0.6289    1.2727    1.9864
##
## Concordance= 0.637  (se = 0.026 )
## Rsquare= 0.126   (max possible= 0.999 )
## Likelihood ratio test= 30.5  on 3 df,   p=1.083e-06
## Wald test            = 29.93  on 3 df,   p=1.428e-06
## Score (logrank) test = 30.5  on 3 df,   p=1.083e-06
```

LASSO for Cox proportional hazards model

- We can also apply LASSO to the Cox proportional hazards model when we have too many predictors and/or we want to do model selection.
- Code is very similar to last time: plug in a **matrix** of predictors and a **vector** of responses. Note the `family = "cox"` argument.

```
lung_nona <- na.omit(lung2)
x <- as.matrix(lung_nona[,4:10])
survobj <- Surv(lung_nona$time, lung_nona$death)
cvfit <- cv.glmnet(x, survobj, family = "cox")
plot(cvfit)
```




```
coef(cvfit, s = "lambda.min")
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## age         0.008713168
## sex        -0.520189680
## ph.ecog     0.659596141
## ph.karno    0.017863574
## pat.karno -0.010730423
## meal.cal    .
## wt.loss    -0.012277564
```

Data Wrangling

- In your HW, you will have to merge data from two different datasets.

- Practice: merge the survival information in `lung_surv` with the predictors in `lung_predictors`. Use the rownames (`id_##`) to distinguish between different subjects.

```
lung_surv <- lung2[, c("time", "death")]
lung_predictors <- select(lung2, -time, -death, -status)
lung_predictors <- lung_predictors[order(lung_predictors$ph.ecog),]
random_predictors <- matrix(rnorm(20 * nrow(lung2)), ncol = 20)
colnames(random_predictors) <- paste0("predictor_", 1:20)
lung_predictors <- cbind(lung_predictors, random_predictors)

# merge the predictors with the survival information so you can
# run a Cox regression using the predictors sex + ph.ecog
# your turn
lung3 <- merge(lung_surv, lung_predictors, by = "row.names")
cox_mod3 <- coxph(Surv(time, death) ~ sex + ph.ecog, data = lung3)
summary(cox_mod3)
```

```
## Call:
## coxph(formula = Surv(time, death) ~ sex + ph.ecog, data = lung3)
##
##   n= 227, number of events= 164
##    (1 observation deleted due to missingness)
##
##             coef exp(coef) se(coef)      z Pr(>|z|)
## sex     -0.5530    0.5752   0.1676 -3.300 0.000967 ***
## ph.ecog  0.4875    1.6282   0.1122  4.344  1.4e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##         exp(coef) exp(-coef) lower .95 upper .95
## sex        0.5752     1.7384    0.4142    0.7989
## ph.ecog    1.6282     0.6142    1.3067    2.0288
##
## Concordance= 0.642  (se = 0.026 )
## Rsquare= 0.12   (max possible= 0.999 )
## Likelihood ratio test= 29.05  on 2 df,   p=4.91e-07
## Wald test            = 28.96  on 2 df,   p=5.145e-07
## Score (logrank) test = 29.41  on 2 df,   p=4.104e-07
```

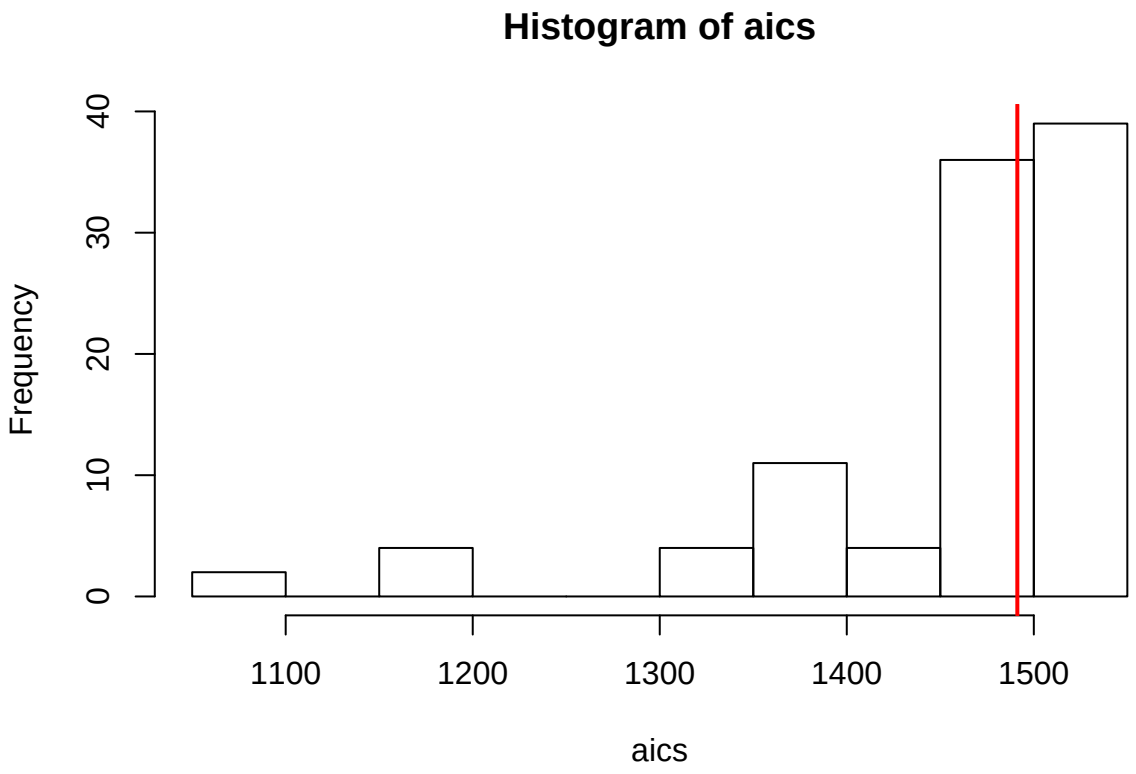
Randomly selecting predictors

- In Q10, you will have to randomly sample predictors and see if the resultant model performs better than a model based on top differentially expressed genes.
- How to compare models?
  - Naively: just look at (an analog of) mean squared error.
  - Not good because as we add more predictors, we will artificially decrease the mean squared error.
  - One alternative is the AIC
- AIC: For a model with k parameters, the AIC is $2k - 2\log\text{-likelihood}$.
  - Smaller is better
  - Penalizes models that have too many useless predictors.

- Let's practice! Above, we just merged a bunch of random predictors with the original lung cancer data. Run 100 simulations to see if randomly selecting 3 predictors does better than `cox_mod2`.
- Let's break it down into steps:

```
set.seed(20180410)
# your turn
aics <- rep(NA, 100)
for (i in 1:100) {
    # sample which predictors you want to use
    pred_index <- sample(1:ncol(lung_predictors), 3, replace = FALSE)
    # subset the predictors
    predictors_touse <- lung_predictors[, pred_index]
    # merge predictors with survival information
    data_touse <- merge(lung2[, c("time", "death")], predictors_touse,
                        by = "row.names")
    data_touse$Row.names <- NULL
    # fit the model
    mod <- coxph(Surv(time, death) ~ ., data = data_touse)
    # extract the AIC
    aics[i] <- AIC(mod)
}

# visualize
hist(aics)
abline(v = AIC(cox_mod1), lwd = 2, col = "red")
```




```
mean(aics < AIC(cox_mod1))
```

```
## [1] 0.51
```