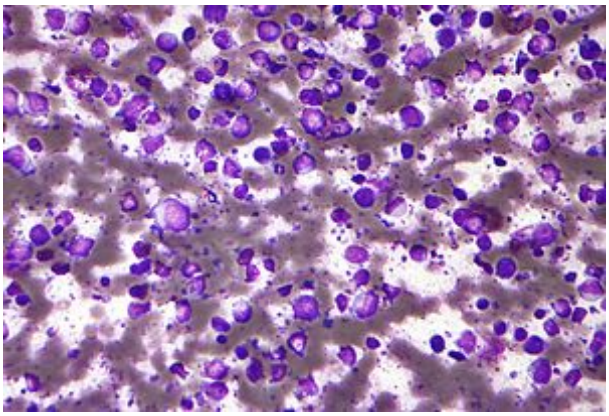
High Dimensional Analysis

Octavio Mesner

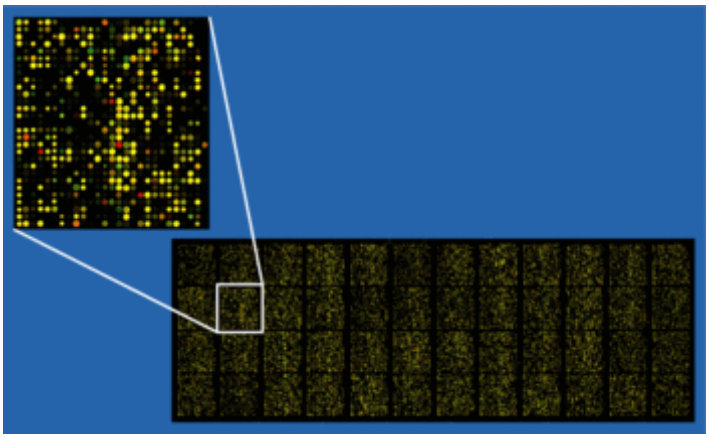
Understanding Gene Expression and Lymphoma



Micrograph (Field stain) of a Diffuse Large B-Cell Lymphoma

From Wikipedia:

- Diffuse large B-cell lymphoma (DLBCL) is the most common lymphoid malignancy in adults
- Cancer of B cells, a type of lymphocyte
- Incidence: 7-8 cases per 100,000 people per year in the US
- occurs primarily in older individuals, with a median age of diagnosis at ~70 years
- it can occur in young adults and, in rare cases, children
- DLBCL can arise in virtually any part of the body and, depending on various factors, is often a very aggressive malignancy
- The causes of diffuse large B-cell lymphoma are not well understood
- Curable in < 50% of patients (in 2002)
- Here, we want to use gene expression or *microarray data* to predict DLBCL
- Microarrays (https://en.wikipedia.org/wiki/Microarray) simultaneously detect the expression of thousands of genes from a sample



Microarray Data

- This paper (https://idp.nature.com/authorize/casa?redirect_uri=https://www.nature.com/articles/nm0102-68&casa_token=vw6a-r_uXz8AAAAA:PBI84ZmryGuDsD_exYAi6aTypeES0fxnpFfASvUwbJ7o_Ihb5FsyW_IvxXGlrCj4UIlyydKIpRIbH4rx8A) use a supervised learning model on 6817 gene expressions (microarray) from 71 patients to distinguish between DLBCL and follicular lymphoma (FL)
- This data is on Github (https://github.com/ramhiser/datamicroarray/wiki/Shipp-%282002%29)
- In this data, each row corresponds to an individual
- Each column corresponds to gene expression using a microarray, one column gives the type of lymphoma
- **Goal**: Determine which genes are associated with with lymphoma type
- Start by looking at the data

```
load("./shipp.RData")
ls() # .RData files can include any R data structure.  The "ls()" command shows what it con
tains.
```

```
## [1] "shipp"
```

```
names(shipp)
```

```
## [1] "x" "y"
```

```
dim(shipp$x)
```

```
## [1]   77 7129
```

```
names(shipp$x)[1:10]  #first 10 var names
```

```
##  [1] "V1"  "V2"  "V3"  "V4"  "V5"  "V6"  "V7"  "V8"  "V9"  "V10"
```

```
shipp$x[1:10,1:6]
```

```
##       V1   V2   V3  V4   V5   V6
## 1  -104 -187  -26  59 -238 -258
## 2  -152 -328  -52 267 -300 -314
## 3  -158 -129   11  88 -239 -429
## 4  -124 -121   -3 -37 -210 -309
## 5   -93 -258  -36 109 -109 -272
## 6   -34 -257 -104  71 -196 -250
## 7  -251 -264  -99  31 -244 -110
## 8  -204 -293  -32 148 -327 -215
## 9  -144 -356 -194  84 -269 -235
## 10  -94 -204  -28  53 -166 -284
```

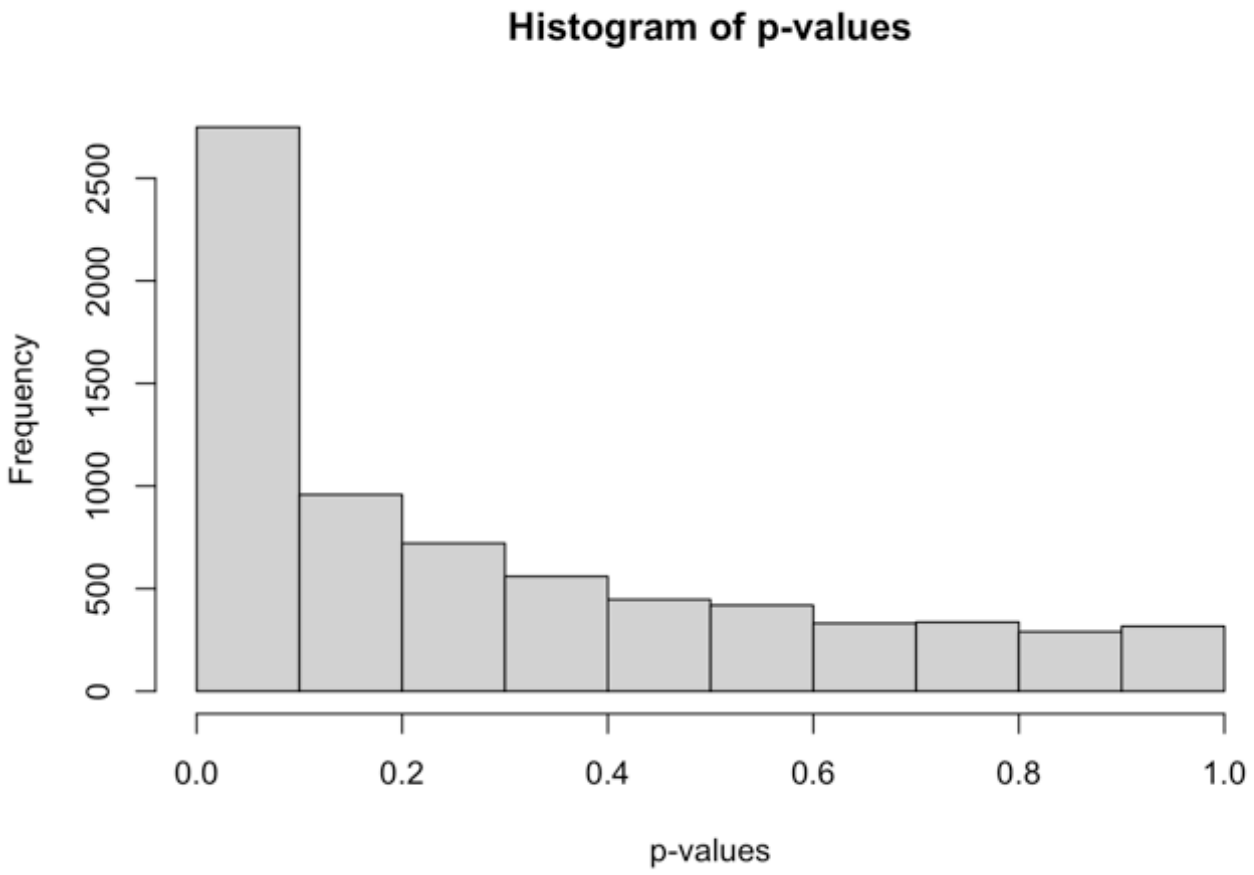
```
table(shipp$y)
```

```
##
## DLBCL    FL
##    58    19
```

- Note that there are far more columns than rows here
- For this, I would visually inspect data data
- It doesn't make sense to try to look at >7k plots
- Question: Will standard regression methods will not work?
    - A: Yes
    - B: No, there would be more parameters in the model than rows of data
    - C: No, X^TX is a singular matrix
    - D: Not Sure

- Question: Should we use a t-test to determine which genes are associated with DLBCL?
    - A: Yes
    - B: No
    - C: Not Sure

```
pvals <-c()
for(var in names(shipp$x)){
  pvals <- c(pvals, t.test(shipp$x[[var]] ~ shipp$y)$p.value)
}
hist(pvals, xlab = 'p-values', main='Histogram of p-values')
```




```
sum(pvals < 0.05)
```

```
## [1] 2065
```

```
sum(pvals < 0.05)/length(pvals) # pct significant at alpha = 0.05
```

```
## [1] 0.2896619
```

- Question: Do you believe that all tests with p < 0.05 are actually statistically significant?
  - A: Yes
  - B: No
  - C: Not Sure

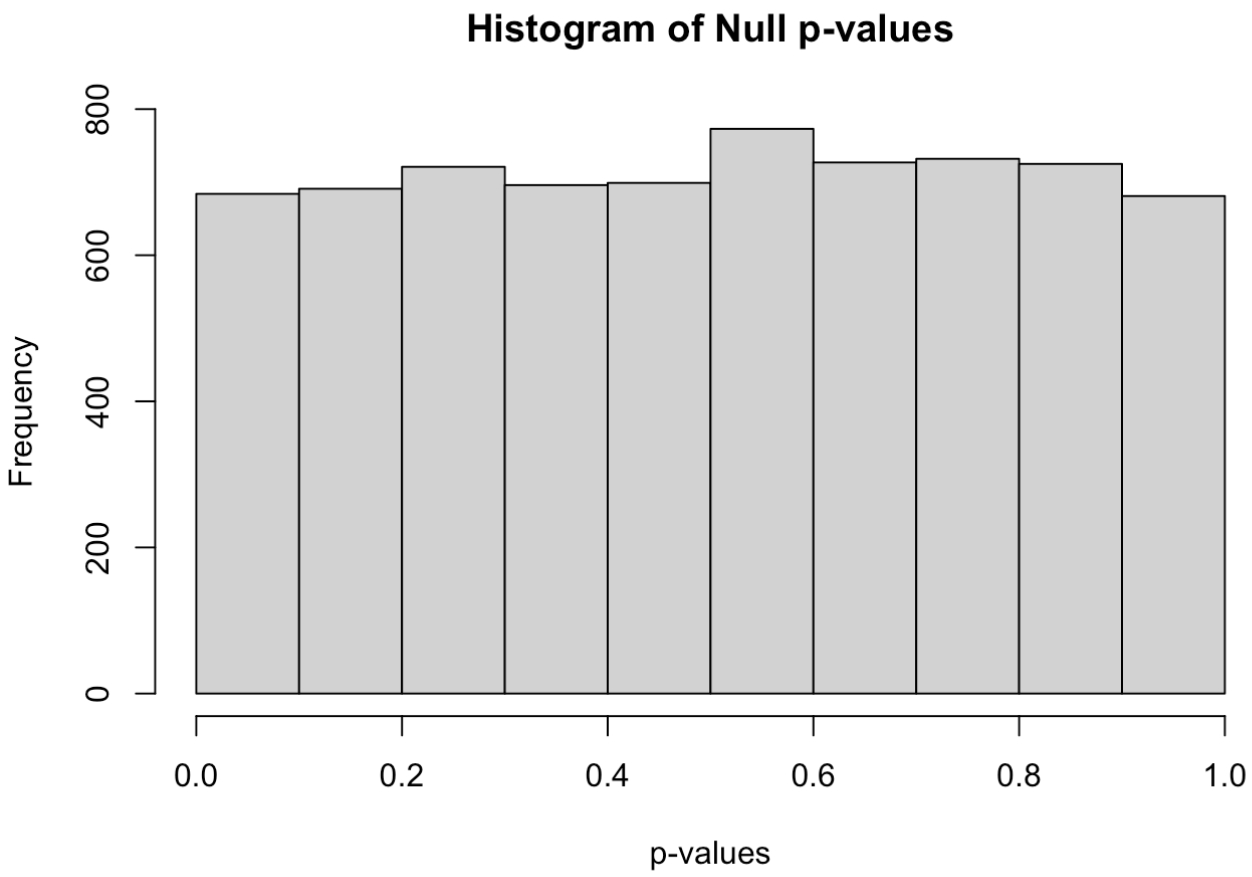
Multiple testing error

- What would happen if we ran many t-tests for columns where none are associated with the outcome?
- The data below are generated randomly and independent from the outcome
- Randomly generate each column independent of the outcome
- For each column, V in the generated data set, we know that $V \perp Y$

```
set.seed(1234)
null_dat <- data.frame(matrix(rnorm(dim(shipp$x)[1] * dim(shipp$x)[2]), ncol=dim(shipp$x)[2
]))
dim(null_dat)
```

```
## [1]   77 7129
```

```
null_pvals <-c()
for(var in names(null_dat)){
  null_pvals <- c(null_pvals, t.test(null_dat[[var]] ~ shipp$y)$p.value)
}
hist(null_pvals, xlab = 'p-values', main='Histogram of Null p-values')
```



- Question: What density does this look like?
  - A: Exponential
  - B: Beta
  - C: Negative Binomial
  - D: Uniform

- Question: What proportion is less than 0.05?
  - Enter response as decimal

```
sum(null_pvals < 0.05)/length(null_pvals) # pct significant at alpha = 0.05
```

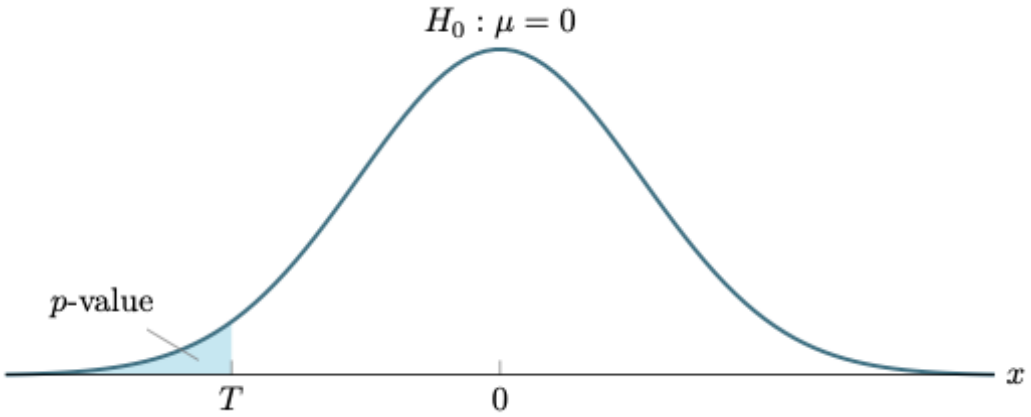
```
## [1] 0.04586899
```

- Intuitively, what is the problem here? Why are so many p-values < 0.05 when none are associated with the outcome?
- Question: Did you expect the p-values to have a uniform distribution?
  - A: Yes
  - B: No
  - C: Not Sure
- Why do they look uniform?

What is a hypothesis test/p-value?

- Example:
  - Let $X_1, X_2, \ldots, X_n$ be an independent, identically distributed sample
  - Let $E[X_i] = \mu$ (unknown) and $E[(X_i - \mu)^2] = \sigma^2 = 1$ (known) for each $i = 1, 2, \ldots, n$

- Is μ positive or negative?
- Let $H_0 : \mu \geq 0$ and $H_1 : \mu < 0$
- Let $\bar{X}_n = \frac{1}{n} \sum_{i=1}^{n} X_i$
- Intuitively, if $\frac{\bar{X}_n - \mu}{\sigma/\sqrt{n}} = \sqrt{n}\bar{X}$ is small (too negative), then we can reject H_0
- From the central limit theorem (https://en.wikipedia.org/wiki/Central_limit_theorem), we know that $\frac{\bar{X}_n - \mu}{\sigma/\sqrt{n}} \rightsquigarrow N(0,1)$ (converges in distribution) as $n \to \infty$
- That is, our test statistic, $T = \sqrt{n}\bar{X}$ is approximately $N(0,1)$ under H_0 (we're setting $\mu = 0$)
- p-value: $p = P[T < N(0,1)] = \Phi(T) = \int_{-\infty}^{T} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$



p-value illustration

other illustration (http://blog.analytics-toolkit.com/2017/statistical-significance-ab-testing-complete-guide/)

Theorem: p-values are uniformly distributed under the null hypothesis.

Proof:

1. Let T be a test statistic of a hypothesis test and let F be the CDF of T under H_0, the null hypothesis
2. p-value: $p = F(T)$
3. Claim: A random variable, $Y \sim G$, is uniformly distributed on $[0,1] \Leftrightarrow P[Y < y] = G(y) = y$ on $[0,1]$
4. Goal: show that $P[p < y] = y$ under the null hypothesis
5. Using F^{-1}, the inverse function of F
6. Assume that F is montonically increasing, so that F^{-1} is also monontically increasing

$$
\begin{aligned}
P[p < y] &= P[F(T) < y] && p = F(T) \\
&= P[F^{-1}(F(T)) < F^{-1}(y)] && F^{-1} \text{ monotonically increasing} \\
&= P[T < F^{-1}(y)] && F^{-1}(F(y)) = y \\
&= F(F^{-1}(y)) && F(t) = P(T < t) \\
&= y
\end{aligned}
$$

7. So, under the null hypothesis, p-value $\sim$Unif(0,1) ■

- Question: Did this proof make sense?
  - 1 (not at all) to 5 (perfect sense)

Multiple Testing

- Want to run m tests
- Let $H_{0,i}$ be the null hypothesis for the ith test where $1 \leq i \leq m$
- Let p_i be the p-value of the ith test for $H_{0,i}$
- Let $R_i = 1$ if we reject $H_{0,i}$ and $R_i = 0$ if we fail to reject $H_{0,i}$ (sometimes called a discovery or positive result)
- $R = \sum_{i=1}^{m} R_i$ be the number of rejected tests
- Let $V_i = 1$ if we wrongly reject $H_{0,i}$ (false positive or type 1 error or false discovery)
- Let $V = \sum_{i=1}^{m} V_i$ be the false positive (discovery) count
- Question: If we run $m = 3$ *independent* hypotheses tests at an $\alpha = 0.1$ significance level, and all null hypotheses, $H_{0,1}, H_{0,2}, H_{0,3}$, are true, what is $P(V = 0)$, the probability that there are zero false discoveries?

- Question: Using the same scenario, what is $P(V > 0)$, the probability that there is at least one false discovery?

- **Family-wise error rate (FWER)**

  $$P[V > 0] \leq \alpha$$

  Same as $P[V = 0] \geq 1 - \alpha$
- Question: If $m = 100$, and all tests are *independent*, and the null hypothesis is true for all, what is $E[V]?$, the expected number of false discoveries?

- **Per family error rate (PFER)**

$$E[V] \leq \alpha$$

- **False discovery rate (FDR)** controls

$$E\left[\frac{V}{R}\right]$$

  - If $R = 0$, use $R = 1$ instead
- **Global null**
  - Can we reject at least one of the m null hypotheses?
  - The global null hypothesis is

$$H_0 = \bigcap_{i=1}^{m} H_{0,i}$$

  - Does not indicate which $H_{0,i}$ to reject, only that we can reject at least one $H_{0,i}$
- Note: We only know the distribution of a p-value under the null hypothesis; we typically won't know the distribution of the p-value under the alternative hypothesis.
- Note: We can only multiply p-values when their corresponding hypothesis tests are independent.

Tests

Bonferroni

- Reject $H_{0,i}$ if $p_i \leq p_{\text{Bon}} := \frac{\alpha}{m}$
- uses FWER, tells us which $H_{0,i}$ we can reject
- Question: We have a standard, 52 playing cards (4 suits, 13 ranks). $P[\text{Ace OR club}] \leq P[\text{Ace}] + P[\text{club}]$
  - A. True
  - B. False

- Union bound: For a countable number of events, $A_1, A_2, \ldots,$

$$P\left(\bigcup_{i=1}^{\infty} A_i\right) \leq \sum_{i=1}^{\infty} P(A_i)$$

More on union bound (https://en.wikipedia.org/wiki/Boole%27s_inequality)

Proof:

Let $I := \{i : H_{0,i} = 1\}$ be the set of true null hypotheses.

$$
\begin{aligned}
P[V > 0] &= P\left[\bigcup_{i \in I} \left\{p_i \leq \frac{\alpha}{m}\right\}\right] && V > 0 \Leftrightarrow \text{ at least one } p_i \text{ significant in nulls} \\
&\leq \sum_{i \in I} P\left[p_i \leq \frac{\alpha}{m}\right] && \text{Union bound} \\
&= \sum_{i \in I} \frac{\alpha}{m} && p_i \sim \text{Unif}(0,1) \text{ for } i \in I \\
&= \frac{\alpha |I|}{m} && |I| = \#\text{ true null hypotheses} \\
&\leq \alpha && \frac{|I|}{m} \leq 1
\end{aligned}
$$

- Because of union bound, even holds when p-values are statistically dependent
- We can also get all Bonferroni-corrected p-values: $\frac{p_i}{m}$ for all i
- Using Bonferroni, which gene's should I recommend are associated with lymphoma?

```
bcorrection <- 0.05/length(pvals)
bcorrection
```

```
## [1] 7.013606e-06
```

```
sum(pvals <= bcorrection)
```

```
## [1] 245
```

```
names(shipp$x)[pvals <= bcorrection]
```

```
##   [1] "V163"  "V171"  "V173"  "V191"  "V200"  "V203"  "V228"  "V233"  "V259"
##  [10] "V275"  "V282"  "V302"  "V316"  "V322"  "V323"  "V355"  "V373"  "V375"
##  [19] "V379"  "V385"  "V386"  "V405"  "V407"  "V438"  "V440"  "V441"  "V463"
##  [28] "V506"  "V538"  "V544"  "V582"  "V592"  "V605"  "V640"  "V649"  "V658"
##  [37] "V778"  "V792"  "V834"  "V922"  "V972"  "V1016" "V1092" "V1110" "V1119"
##  [46] "V1127" "V1132" "V1157" "V1166" "V1173" "V1176" "V1188" "V1217" "V1245"
##  [55] "V1248" "V1346" "V1352" "V1369" "V1372" "V1373" "V1394" "V1430" "V1445"
##  [64] "V1479" "V1480" "V1490" "V1551" "V1621" "V1691" "V1704" "V1733" "V1778"
##  [73] "V1780" "V1790" "V1804" "V1818" "V1823" "V1825" "V1829" "V1832" "V1860"
##  [82] "V1903" "V1930" "V1934" "V1942" "V2030" "V2043" "V2094" "V2111" "V2121"
##  [91] "V2137" "V2145" "V2177" "V2185" "V2220" "V2229" "V2237" "V2282" "V2306"
## [100] "V2308" "V2309" "V2319" "V2380" "V2383" "V2386" "V2392" "V2412" "V2419"
## [109] "V2421" "V2527" "V2645" "V2668" "V2669" "V2729" "V2761" "V2789" "V2883"
## [118] "V2919" "V2929" "V2937" "V2941" "V2947" "V2988" "V3005" "V3137" "V3163"
## [127] "V3181" "V3248" "V3257" "V3258" "V3283" "V3310" "V3325" "V3351" "V3387"
## [136] "V3397" "V3461" "V3494" "V3499" "V3545" "V3639" "V3693" "V3736" "V3757"
## [145] "V3818" "V3887" "V3926" "V3928" "V3943" "V3987" "V4010" "V4024" "V4028"
## [154] "V4078" "V4087" "V4110" "V4116" "V4130" "V4133" "V4153" "V4158" "V4183"
## [163] "V4202" "V4242" "V4243" "V4254" "V4262" "V4273" "V4292" "V4328" "V4330"
## [172] "V4340" "V4372" "V4373" "V4387" "V4406" "V4418" "V4424" "V4453" "V4463"
## [181] "V4485" "V4502" "V4503" "V4518" "V4535" "V4546" "V4567" "V4580" "V4582"
## [190] "V4594" "V4667" "V4754" "V4970" "V5077" "V5092" "V5169" "V5187" "V5254"
## [199] "V5280" "V5302" "V5324" "V5336" "V5409" "V5579" "V5594" "V5600" "V5621"
## [208] "V5655" "V5666" "V5671" "V5704" "V5743" "V5882" "V5935" "V5956" "V5959"
## [217] "V5994" "V5998" "V6023" "V6050" "V6058" "V6110" "V6179" "V6191" "V6295"
## [226] "V6310" "V6322" "V6334" "V6352" "V6362" "V6377" "V6378" "V6474" "V6524"
## [235] "V6549" "V6612" "V6659" "V6676" "V6815" "V6986" "V7005" "V7091" "V7102"
## [244] "V7119" "V7123"
```

- Question: Which of the following interpretations are acceptable? (can choose more than one)
  - A. The genes listed above are the most likely to be associated with lymphoma at an $\alpha = 0.05$ significance level
  - B. At an $\alpha = 0.05$ significance level, all of the genes listed above are associated with lymphoma
  - C. All of the genes above are significantly associated with lymphoma using a Bonferroni-corrected $\alpha = 0.05$ significance level
  - D. Controlling the aggregate probability of having at least one false discovery at an $\alpha = 0.05$ significance level, the genes listed above are associated with lymphoma

- What about the independent simulated data?

```
which(null_pvals<=bcorrection)
```

```
## integer(0)
```

- Notice that test may be associated (knowing one p-value may give information about others)
- Bonferroni still works here
- How could we construct a global null hypothesis from Bonferroni?
    - If at least one $p_i \leq \frac{\alpha}{m}$, then we can reject global null

Bonferroni is conservative

- Types of error: false positives (type I error) and false negatives (type II error)
- Bonferroni is very good at limiting false positives but not limiting false negatives
- Power = 1-P(type II error), is the probability of rejecting the null hypothesis when the alternative hypothesis is true
- Bonferroni may not always be a powerful test

Fisher Combination

- Uses global null framework
- Assumes that each p_i is independent (why might this not be reasonable in most settings?)
- If $H_{0,i}$ is true for each i (and each test is independent), then

$$T = \sum_{i=1}^{m} -2\log(p_i) \sim \chi^2(2m)$$

- uses that fact that transforming independent, uniform random variables this way will have a χ^2 distribution
- This test does not indicate which $H_{0,i}$ to reject
- Question: Can we use Fisher combination test here?
    - A. Yes
    - B. No
    - C. Not sure

- Question: We have 10 clinical trial studies on the same treatment on different study participants. Can we use Fisher combination here?
    - A. Yes
    - B. No
    - C. Not sure

Simes Test

- Uses global null framework, i.e. at least one null hypothesis is false

- Order all p-values, $p_{(1)} \leq p_{(2)} \leq \cdots \leq p_{(m)}$ for the m hypotheses
- Simes test statistic

$$p_{\text{Simes}} = \min_{i \in [m]} \left\{ \frac{m p_{(i)}}{i} \right\}$$

- Reject global null if $p_{\text{simes}} \leq \alpha$
- Equivalent: Reject global null if any $p_{(i)} \leq \frac{i\alpha}{m}$
- Proof is a little more complicated
    - Need to show that $p_{\text{Simes}} \sim \text{Unif}(0, 1)$
    - Uses order statistics properties to show this
- Does not require all p_i are independent
- More powerful (type II error is smaller) than Bonferroni global null test
    - Bonferroni will reject global null when $mp_{(1)} \leq \alpha$

```
simes <- function(vec) length(vec)*min(sort(vec)/(1:length(vec)))

simes(pvals)
```

```
## [1] 1.47472e-12
```

```
simes(null_pvals)
```

```
## [1] 0.6610137
```

- Coding tip: DRY (don't repeat yourself)
    - if you're tempted to copy your own code to reuse it, write a function for it instead

Kolmogorov-Smirnov Test

- Assume we observe 10,000 p-values, all between 0.2 and 0.4.
- Question: Are there any significant Bonferroni-corrected p-values?
    - A. Yes
    - B. No
    - C. Not sure

- Question: Would Simes test give a significant p-values?

- A. Yes
- B. No
- C. Not sure

- Question: Can we (logically) reject global null here?
  - A. Yes
  - B. No
  - C. Not sure

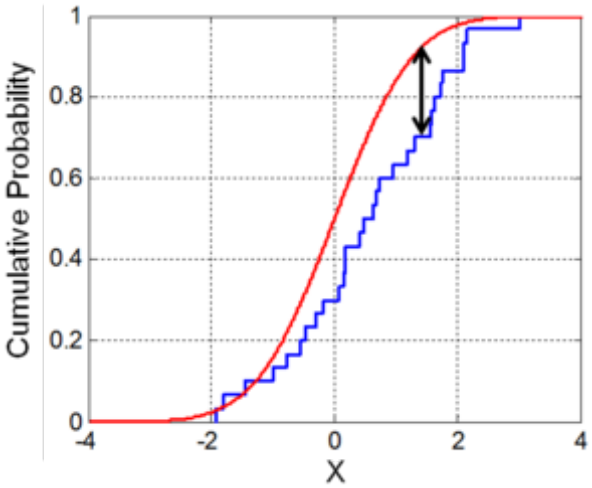


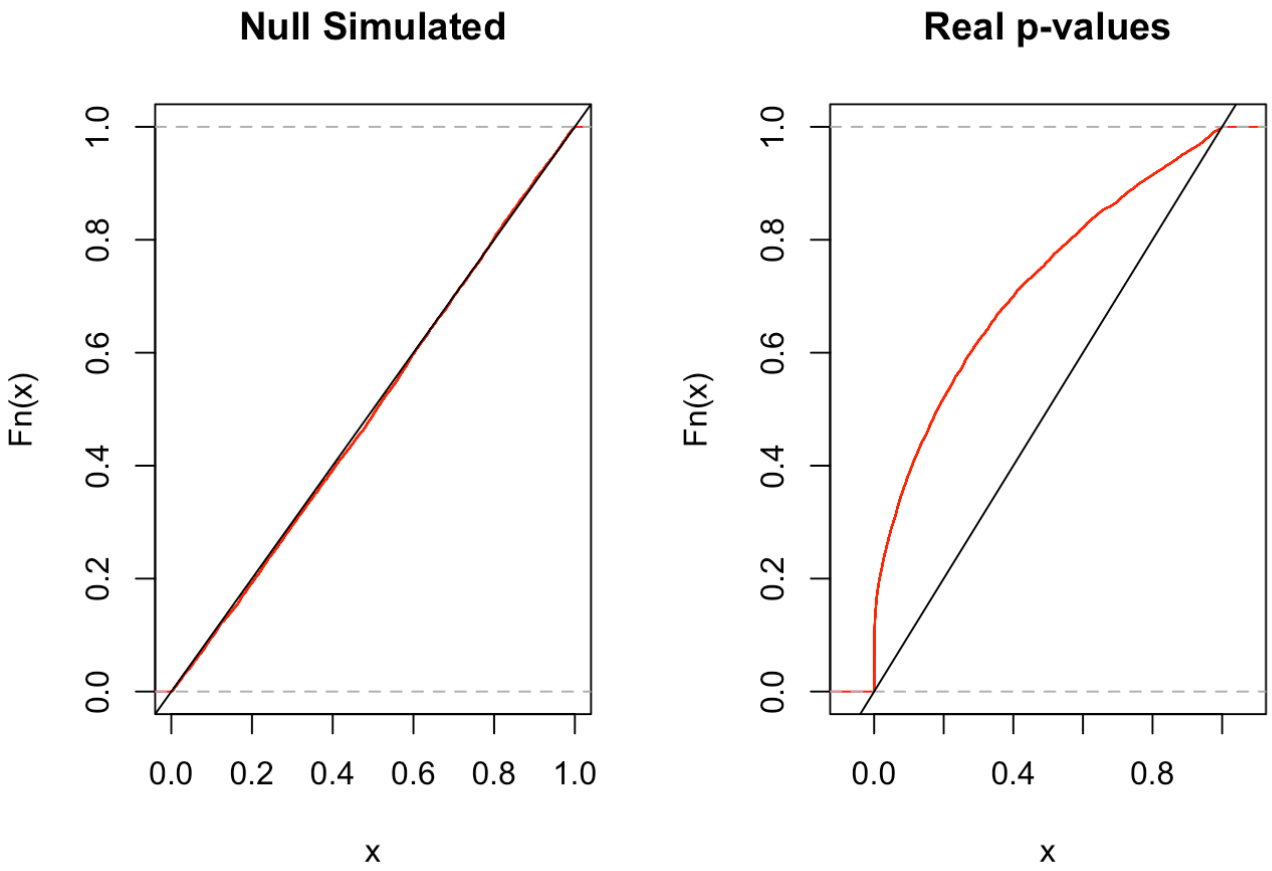
image from wikipedia

- Kolmogorov-Smirnov Test compares an empirical cdf to a theoretical cdf (not only uniform)
- Here: Assess fit of empirical p-value cumulative distribution compared to uniform
- Uses global null framework
- Empirical CDF of p-values is

$$\hat{F}_m(t) = \frac{1}{m}\sum_{i=1}^{m} I(p_i \leq t)$$

- Uniform CDF $F(t) = t$ for $t \in [0, 1]$

```
par(mfrow=c(1,2))
plot(ecdf(null_pvals), col='red', xlim=c(0,1), ylim=c(0,1),
     main="Null Simulated")
abline(c(0,0), c(1,1))
plot(ecdf(pvals), col='red',
     main="Real p-values")
abline(c(0,0), c(1,1))
```



- Test statistic

$$T_m = \sup_{t \in [0,1]} \left| \hat{F}_m(t) - t \right|$$

- Reject global null when T_m is large
- We don't know the CDF of the test statistic, but we *consentration inequality* above allows us to bound the p-value
- Using Hoeffding's inequality, under the null hypothesis

$$P\left[T_m > \epsilon\right] \leq 2\exp\left(-2\epsilon^2\right)$$

- Want to find ϵ so that $2\exp\left(-2\epsilon^2\right) \leq \alpha$
- Reject global null if

$$T_m > \sqrt{\frac{1}{2}\log\left(\frac{2}{\alpha}\right)}$$

```
ks.test(pvals, runif(100000))
```

```
## Warning in ks.test(pvals, runif(1e+05)): p-value will be approximate in the
## presence of ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  pvals and runif(1e+05)
## D = 0.32639, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
ks.test(null_pvals, runif(100000))
```

```
## Warning in ks.test(null_pvals, runif(1e+05)): p-value will be approximate in the
## presence of ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  null_pvals and runif(1e+05)
## D = 0.011325, p-value = 0.3607
## alternative hypothesis: two-sided
```

See ks.test documentation (https://stat.ethz.ch/R-manual/R-devel/library/stats/html/ks.test.html) for more info

Benjamini-Hochberg

- Method for controlling the false discovery rate (FDR), $E\left[\frac{V}{R}\right]$, the expected ratio of false discoveries to all discoveries
- Interpretation: At an FDR of α, we would expect, at most, α of our significant tests to be false positives
- Method:
    1. Order all p-values, $p_{(1)} \leq p_{(2)} \leq \cdots \leq p_{(m)}$ for the m hypotheses
    2. Let $j = \max\left\{i : p_{(i)} < \frac{i\alpha}{m}\right\}, T_{\text{BH}} = p_{(j)}$
    3. Reject $H_{0,i}$ for $p_i \leq T_{\text{BH}}$
- Alternative:
    - adjust p-values using

$$\frac{mp_{(i)}}{i}$$

Proof outline

- Recall

$$\text{FDR} = E\left[\frac{V}{R}\right] = E\left[\frac{\text{\# Type 1 Error}}{\text{\# Rejections}}\right]$$

- Let $W_i = 1$ if $H_{0,i}$ is true and $W_i = 0$ otherwise.
- Let G be the true CDF of the p-values and let $\hat{G}$ be the empirical CDF as before.
- Let t be the significance level.

$$\begin{aligned}
\text{FDR} &= E\left[\frac{V}{R}\right] && \text{definition of FDR} \\
&= E\left[\frac{\sum_{i=1}^m W_i I(p_i < t)}{\sum_{i=1}^m I(p_i < t)}\right] && \frac{\text{\# false discoveries}}{\text{\# all discoveries}} \\
&\approx \frac{E\left[\frac{1}{m}\sum_{i=1}^m W_i I(p_i < t)\right]}{E\left[\frac{1}{m}\sum_{i=1}^m I(p_i < t)\right]} && \text{Approximately} \\
&= \frac{\frac{1}{m}\sum_{i=1}^m W_i E\left[I(p_i < t)\right]}{\frac{1}{m}\sum_{i=1}^m E\left[I(p_i < t)\right]} && \text{linearity of expectation} \\
&= \frac{\frac{t}{m}\sum_{i=1}^m W_i}{G(t)} && \frac{\text{null hypothesis true}}{\text{true } p\text{-value CDF}} \\
&\leq \frac{t}{G(t)} && \frac{1}{m}\sum_{i=1}^m \leq 1 \\
&\approx \frac{t}{\widehat{G}(t)} && \text{empirical CDF close to true CDF}
\end{aligned}$$

- Let $t = p_{(i)}$ for some i.
- Notice that $\widehat{G}(p_{(i)}) = \frac{i}{m}$, so that

$$\text{FDR} \leq \frac{p_{(i)}}{\widehat{G}(p_{(i)})} = \frac{p_{(i)}m}{i}$$

- We can use BH-adjusted p-values also: $\frac{p_{(i)}m}{i}$

```
bh_adj <- p.adjust(pvals, 'BH')
round(bh_adj, 3)[1:10]
```

```
##  [1] 0.150 0.649 0.989 0.501 0.855 0.678 0.436 0.984 0.591 0.936
```

```
round(pvals[1:10], 3)
```

```
##  [1] 0.040 0.492 0.982 0.318 0.773 0.529 0.248 0.975 0.424 0.892
```

```
which(bh_adj < 0.05)[1:10]
```

```
##  [1] 38 44 45 46 49 52 57 64 65 87
```

```
sum(bh_adj<0.05)
```

```
## [1] 1232
```

- Question: Which interpretations are acceptable here?
  - A. Using Benjamini-Hochberg false discovery rate of 0.05, 1232 genes were associated with lymphoma
  - B. 1232 genes were associated with lymphoma allowing for a 5% false discovery rate

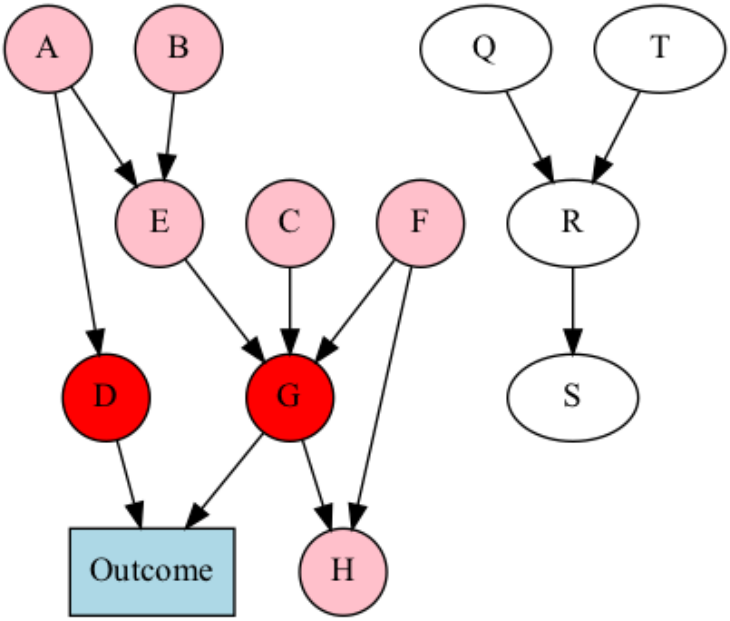
See p.adjust documentation (https://stat.ethz.ch/R-manual/R-devel/library/stats/html/p.adjust.html) for more info

Final thoughts on multiple testing

- Family-wise error and false discovery
- Only Bonferroni and Benjamini-Hochberg indicate which null hypotheses to reject, all others are global null
- Many applied papers do not use multiple testing procedures but should
- Prediction, causation, and dependence
  - Smoking and cancer can both be used to predict death, but should we use both?

- What about something more complex?
- Question: Should D in the model below associated with Outcome?
  - A: Yes
  - B: No
- Question: Should A in the model below associated with Outcome?
  - A: Yes
  - B: No
- Question: Should Q in the model below associated with Outcome?
  - A: Yes
  - B: No
- Question: In a regression model (Outcome ~ A+B+C+D+E+F+G+H+Q+R+S+T), which variables should be associated/significant?
  - A: Red Only
  - B: Pink Only
  - C: Red and Pink Only

    - D: All Variables
- Question: In a regression model (Outcome ~ A+B+C+E+F+G+H+Q+R+S+T, D is removed), which variables should be associated/significant?
    - A: G only
    - B: A and G only
    - C: A, E, and G only
    - D: A, B, E, and G only



- pink and red variables will be associated with Outcome
- D and G are sufficient for prediction
- if D and G are used, other variables provide no additional accuracy
- regression takes this into account
- $\beta = [\beta_A, \beta_B, \beta_C, \beta_D, \beta_E, \beta_F, \beta_G, \beta_H, \beta_Q, \beta_R, \beta_S, \beta_T] = [0, 0, 0, \beta_D, 0, 0, \beta_G, 0, 0, 0, 0, 0]$ where $\beta_D, \beta_G \neq 0$
- $\{\beta_D, \beta_G\}$ is the *active* set
- What should the active set be if G were missing?

Penalized Regression

- Using tests, we looked for *marginal* associations between each gene and lymphoma
- If we need to *control* for other variables, we have to use regression
- Penalized regression was developed for high dimensional data

Ordinary least squares

$$\text{Have } \mathbf{X} = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1p} \\ X_{21} & X_{22} & \cdots & X_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \cdots & X_{np} \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}. \text{ Want to estimate } \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}$$

so that

$$\hat{\beta} = \arg\min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|_2^2$$

Note: The Euclidean norm, $\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_p^2} = \sqrt{x^T x}$ for $x \in \mathbb{R}^p$

Using calculus, it's easy to show that

$$\hat{\beta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

Proof: Taking the gradient (derivative with respect to vector) and setting it equal to zero,

$$\begin{aligned}\frac{\partial}{\partial\beta}\|\mathbf{y} - \mathbf{X}\beta\|_2^2 &= \frac{\partial}{\partial\beta}(\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) \\ &= \frac{\partial}{\partial\beta}\mathbf{y}^T\mathbf{y} - \frac{\partial}{\partial\beta}2\mathbf{y}^T\mathbf{X}\beta + \frac{\partial}{\partial\beta}\beta^T X^T X\beta \\ &= -2\mathbf{y}\mathbf{X} + 2\mathbf{X}^T\mathbf{X}\beta = 0\ .\end{aligned}$$

Solving for zero, we get the solution above.

OLS Pros:

- When true relationship between response and predictors is approximately linear, low bias
- When $n \gg p$, low variance

OLS Cons:

- When $n \not\gg p$, *overfitting* can be a problem
- $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ only exists for $n >> p$.

Ridge Regression

- Ridge regression uses the OLS optimization problem augmented with a penalty for the size of β

  $$\hat{\beta} = \arg\min_{\beta} \left\|y - X^T\beta\right\|_2^2 + \lambda\|\beta\|_2^2$$

  where y is a $n \times 1$ vector and X is a $n \times p$ matrix
- We can think of this optimization as a compromise between fit and size of β
- Similar to OLS, $\hat{\beta}$ also has a closed form

  $$\hat{\beta} = \left(\mathbf{X}^T\mathbf{X} + \lambda I\right)^{-1}\mathbf{X}^T\mathbf{y}$$

  where I is the identity matrix
- Sometimes written differently

$$\hat{\beta} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} \left(y_i - X_i^T \beta\right)_2^2 + \lambda \|\beta\|_2^2$$

- This may change the size of λ but λ is a nuisance parameter
- Question: As λ increases, β
    - A. increases
    - B. decreases
    - C. stays the same
    - D. not sure

Proof: Taking the gradient (derivative with respect to vector) and setting it equal to zero,

$$\begin{aligned}
\frac{\partial}{\partial \beta}\left[\|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda\|\beta\|_2^2\right] &= \frac{\partial}{\partial \beta}\left[(\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda\beta^T\beta\right] \\
&= \frac{\partial}{\partial \beta}\mathbf{y}^T\mathbf{y} - \frac{\partial}{\partial \beta}2\mathbf{y}^T\mathbf{X}\beta + \frac{\partial}{\partial \beta}\beta^T X^T X\beta + \frac{\partial}{\partial \beta}\lambda\beta^T\beta \\
&= -2\mathbf{y}\mathbf{X} + 2\mathbf{X}^T\mathbf{X}\beta + 2\lambda\beta = 0\,.
\end{aligned}$$

- Solving for zero, we get the solution above.

Using `GLMNET`

- `GLMNET` is typically the go-to R package for penalized regression
- Below: create a dataset where 10 variable are associated with the outcome and 20 are not
- We are choosing the true β randomly from a uniform distribution

```
### Generating data
sample_size <- 50 # remember DRY
set.seed(1234)
num_active_vars <- 10 # 10 variables will be associated with the outcome
num_null_vars <- 20 # 20 variables will not be associated with the outcome
true_beta <- 2*runif(num_active_vars) # randomly choosing true beta
true_beta # printing out active beta values
```

```
##  [1] 0.22740682 1.24459881 1.21854947 1.24675888 1.72183077 1.28062121
##  [7] 0.01899151 0.46510101 1.33216752 1.02850228
```

```
# active_x values will be used to construct y
active_x <- matrix(rnorm(sample_size*num_active_vars), nrow=sample_size)

# null_x won't be used to construct y but will be included in the data
null_x <- apply(matrix(3*rnorm(sample_size*num_null_vars), nrow=sample_size), 2,
                function(x) x + 10*runif(1))

# constructing y with noise
y <- active_x %*% true_beta + rnorm(sample_size)

dim(y) #sanity check
```

```
## [1] 50  1
```

```
# making data frame
dat <- data.frame(cbind(active_x, null_x, y))

dim(dat) # sanity check
```

```
## [1] 50 31
```

```
names(dat) #sanity check
```

```
##  [1] "X1"  "X2"  "X3"  "X4"  "X5"  "X6"  "X7"  "X8"  "X9"  "X10" "X11" "X12"
## [13] "X13" "X14" "X15" "X16" "X17" "X18" "X19" "X20" "X21" "X22" "X23" "X24"
## [25] "X25" "X26" "X27" "X28" "X29" "X30" "X31"
```

```
names(dat)[31] <- 'Y' # renaming response variable
head(dat)
```

```
##           X1         X2          X3         X4          X5         X6
## 1  0.5060559  0.5630558  0.16698928  1.3621307  0.76046236 -0.3704975
## 2 -0.5747400  1.6478175 -0.89626463 -0.2346211  1.84246363  1.4769696
## 3 -0.5466319 -0.7733534  0.16818539 -1.0533828  1.11236284 -1.2239038
## 4 -0.5644520  1.6059096  0.35496826 -0.8697836  0.03266396  0.2580684
## 5 -0.8900378 -1.1578085 -0.05210512 -0.3901270 -1.11444896  0.4050028
## 6 -0.4771927  0.6565885 -0.19593462 -0.8473501  0.41805782  0.9758033
##           X7         X8          X9        X10        X11        X12       X13
## 1 -0.6490282  0.5475242  0.15344474 -0.6512008  2.8035838 11.0983215  6.223361
## 2 -0.5043742  1.8912270  1.46328305 -1.4736558 -1.0474032  6.8697732 10.087990
## 3  1.6143915 -0.8780771 -1.12150250 -1.2016658  5.0094210  0.7218307 11.205312
## 4 -0.4469598 -0.1125589 -0.51778808 -0.1487205  4.5974288 12.1763416  3.324929
## 5  0.7631768  1.9487131 -0.07494709  1.7970624  3.4810388 11.1607255  9.802090
## 6  1.4717187  0.9338163 -1.40779008  0.1048087  0.3642669  7.2122591  4.870920
##          X14        X15        X16       X17       X18      X19        X20
## 1  4.0752264 -0.6053643  4.7564041  5.321064  7.799665 5.592186  4.6772059
## 2  6.4132250  2.0202636  0.2970701 -0.318436  9.808652 4.291757 -0.1177785
## 3  2.6473692 -1.7636239 -0.8032459  2.324419  8.307224 2.954393  2.2330444
## 4  3.1922946  3.0787080  3.2034946  4.687638  6.210801 2.632948  3.8991051
## 5  5.0369160 -2.4553716  0.2688438  1.924475 10.546669 1.804470  2.5989062
## 6 -0.2713349  6.3026241  5.3144006  5.345141  7.745985 3.486791  7.8291542
##         X21       X22      X23       X24       X25       X26      X27       X28
## 1 -2.177259  9.976276 5.141210 14.573507 11.989885  7.121203 8.097692  5.660709
## 2  6.357154  7.971671 4.697821 12.306464 10.624856 10.030938 9.797545  5.769629
## 3  2.866913  7.527770 1.023797  7.109257 10.052675 11.150259 1.218631  2.688446
## 4  1.518939  9.099298 6.546246  9.615181  9.290212  5.690831 7.503465 12.380381
## 5  4.877752  6.373329 2.150257 13.436050 10.712848 10.109707 7.914734  4.128210
## 6  7.382241 12.917893 1.876893 12.719626  5.757507  7.645144 7.074355  2.815231
##         X29        X30          Y
## 1 10.888932  0.7104575  1.9144653
## 2  6.300410  0.3718349  9.8368493
## 3  5.952818 -0.5419693 -4.0199055
## 4  6.531286 -2.8194315  1.9355236
## 5  7.981174  3.4402196 -2.8156338
## 6  8.345330 -1.1286243 -0.1450641
```

- Using the `glmnet` function, run ridge regression (`alpha=0`) with various values for λ
- `glmnet` does not use the standard R convention `y ~ x1 + x2 + x3` formula
- must give it a y vector and X design matrix
- **Must standardize variables**:
  - standardizing a variable: $x \rightarrow \frac{x-\hat{\mu}}{\hat{\sigma}}$
  - constraints on the size of the coefficients associated to each variable
  - this value will depend on the magnitude of each variable
  - necessary to center and reduce, or standardize, the variables
  - already done here because all variables generated using standard normal

```
# install.packages("glmnet", repos = "http://cran.us.r-project.org") # only need 1st time
library(glmnet)
```

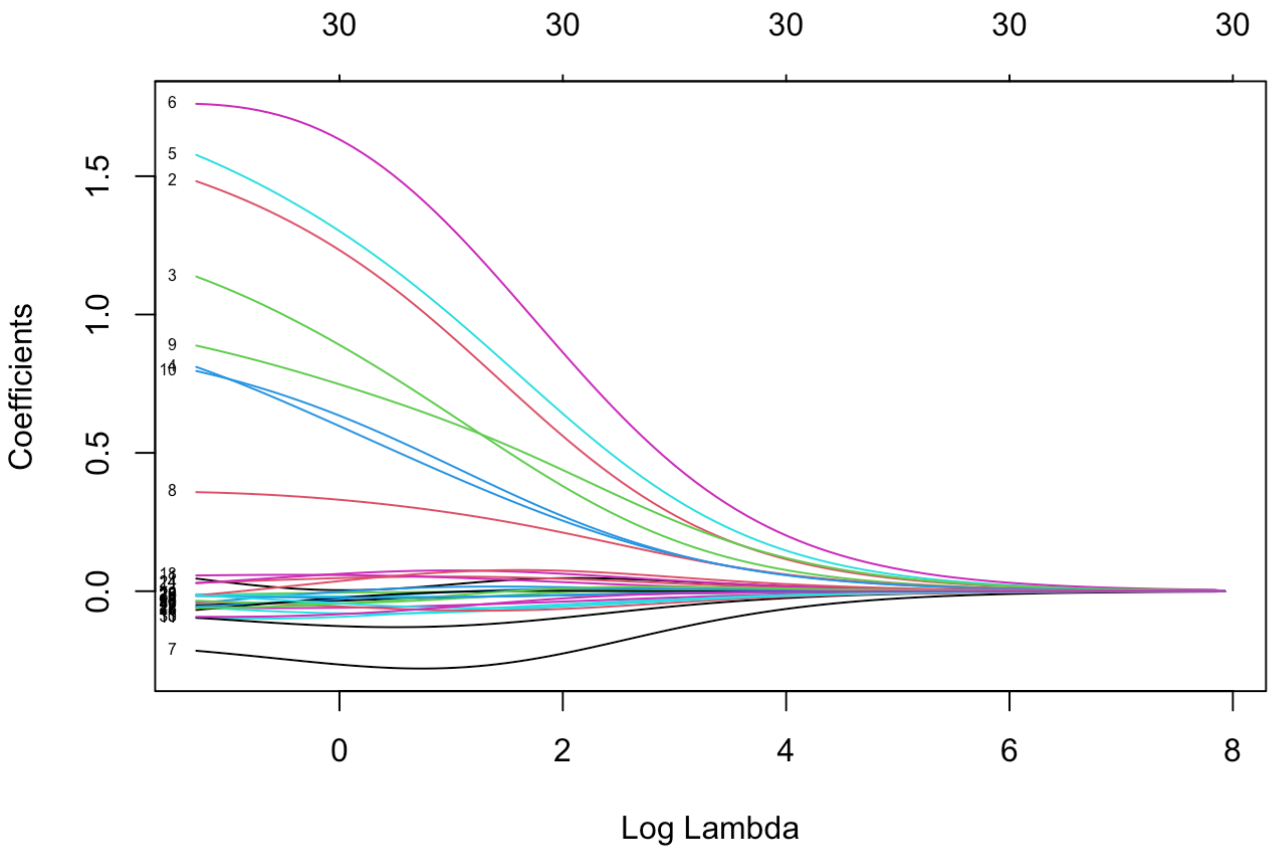
```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.0-2
```

```
design_mat <- cbind(active_x, null_x) # glmnet only takes matrices, not dataframes
l2_fit <- glmnet(design_mat, y, family="gaussian", alpha=0) # alpha = 0 gives Ridge regress
ion
```

- Because of closed form, fast to compute
- Plot change in coefficient estimates with lambda

```
plot(l2_fit, xvar='lambda', label=TRUE)
```



- As λ gets bigger, coefficients shrink
- coefficients get closer to zero but are never exactly zero

- Question: Is this plot helpful for non-statistical experts?
    - A. No
    - B. Yes, with explanation
    - C. Yes, without explanation
    - D. Not sure

- Let see what else `glmnet` gives us using `names`

```
names(l2_fit)
```

```
##  [1] "a0"        "beta"      "df"        "dim"       "lambda"    "dev.ratio"
##  [7] "nulldev"   "npasses"   "jerr"      "offset"    "call"      "nobs"
```

```
coef(l2_fit)[1:10,1:4]
```

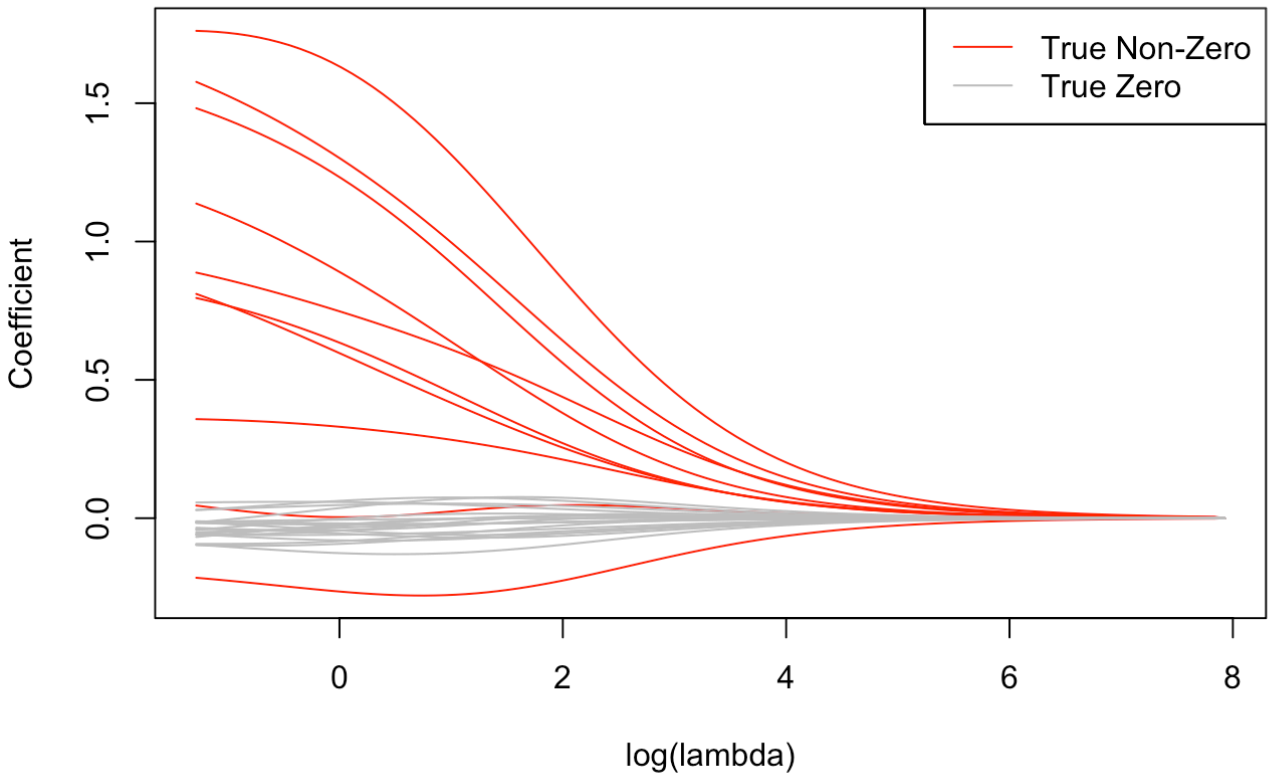
```
## 10 x 4 sparse Matrix of class "dgCMatrix"
##                          s0            s1            s2           s3
## (Intercept)  7.001009e-01  0.6946076979  0.6940756757  0.693492506
## V1           2.683107e-37  0.0004376648  0.0004801991  0.000526852
## V2           1.665212e-36  0.0027187723  0.0029832648  0.003273426
## V3           1.082759e-36  0.0017686261  0.0019407721  0.002129642
## V4           8.842758e-37  0.0014418534  0.0015819206  0.001735540
## V5           2.190053e-36  0.0035726384  0.0039198744  0.004300743
## V6           3.029803e-36  0.0049414403  0.0054215992  0.005948243
## V7          -1.010926e-36 -0.0016470104 -0.0018068629 -0.001982153
## V8           9.657436e-37  0.0015728551  0.0017254526  0.001892775
## V9           1.899821e-36  0.0030947077  0.0033950155  0.003724314
```

```
l2_fit$lambda
```

```
##   [1] 2774.4134622 2527.9422170 2303.3667979 2098.7420401 1912.2955818
##   [6] 1742.4125130 1587.6213878 1446.5814795 1318.0711679 1200.9773582
##  [11] 1094.2858398  997.0725019  908.4953289  827.7871079  754.2487828
##  [16]  687.2433998  626.1905903  570.5615442  519.8744292  473.6902178
##  [21]  431.6088844  393.2659407  358.3292784  326.4962930  297.4912622
##  [26]  271.0629584  246.9824723  225.0412302  205.0491875  186.8331829
##  [31]  170.2354379  155.1121908  141.3324513  128.7768658  117.3366840
##  [36]  106.9128164   97.4149765   88.7608984   80.8756249   73.6908573
##  [41]   67.1443646   61.1794443   55.7444311   50.7922495   46.2800060
##  [46]   42.1686179   38.4224741   35.0091274   31.8990130   29.0651926
##  [51]   26.4831210   24.1304335   21.9867523   20.0335098   18.2537880
##  [56]   16.6321718   15.1546155   13.8083212   12.5816280   11.4639109
##  [61]   10.4454886    9.5175402    8.6720282    7.9016291    7.1996701
##  [66]    6.5600713    5.9772926    5.4462864    4.9624534    4.5216027
##  [71]    4.1199160    3.7539140    3.4204266    3.1165653    2.8396983
##  [76]    2.5874273    2.3575674    2.1481276    1.9572939    1.7834133
##  [81]    1.6249798    1.4806210    1.3490867    1.2292376    1.1200355
##  [86]    1.0205346    0.9298732    0.8472658    0.7719970    0.7034149
##  [91]    0.6409255    0.5839874    0.5321076    0.4848366    0.4417651
##  [96]    0.4025199    0.3667611    0.3341791    0.3044915    0.2774413
```

- Let's see what happened to the active versus non-active coefficient estimates

```
glmnet_plot <- function(fit, num_active){ #assumes active vars are first
  plot(0, type='n', ylim = range(coef(fit)[-1,]), xlim = log(range(fit$lambda)),
     ylab = "Coefficient", xlab="log(lambda)")
  num_vars <- dim(coef(fit))[1]-1 # removing intercept
  for(itr in 1:num_vars){
    active = c(rep('red', num_active), rep('gray', num_vars-num_active))
    lines(log(fit$lambda), coef(fit)[itr+1,], col=active[itr])
    legend('topright', legend = c('True Non-Zero', 'True Zero'), col=c('red', 'gray'), lty
= 1)
  }
}
glmnet_plot(l2_fit, num_active_vars)
```




```
true_beta
```

```
##  [1] 0.22740682 1.24459881 1.21854947 1.24675888 1.72183077 1.28062121
##  [7] 0.01899151 0.46510101 1.33216752 1.02850228
```

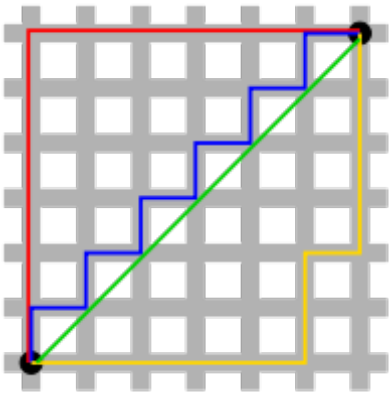
- Question: How should we choose λ?
  - A. AIC
  - B. Deviance
  - C. Cross validation
  - D. Not sure

LASSO

- LASSO (least absolute shrinkage and selection operator) uses an L_1 penalty, not an L_2 (Euclidean) penalty like ridge

$$\hat{\beta} = \arg\min_{\beta} \|y - X^T\beta\|_2^2 + \lambda\|\beta\|_1$$

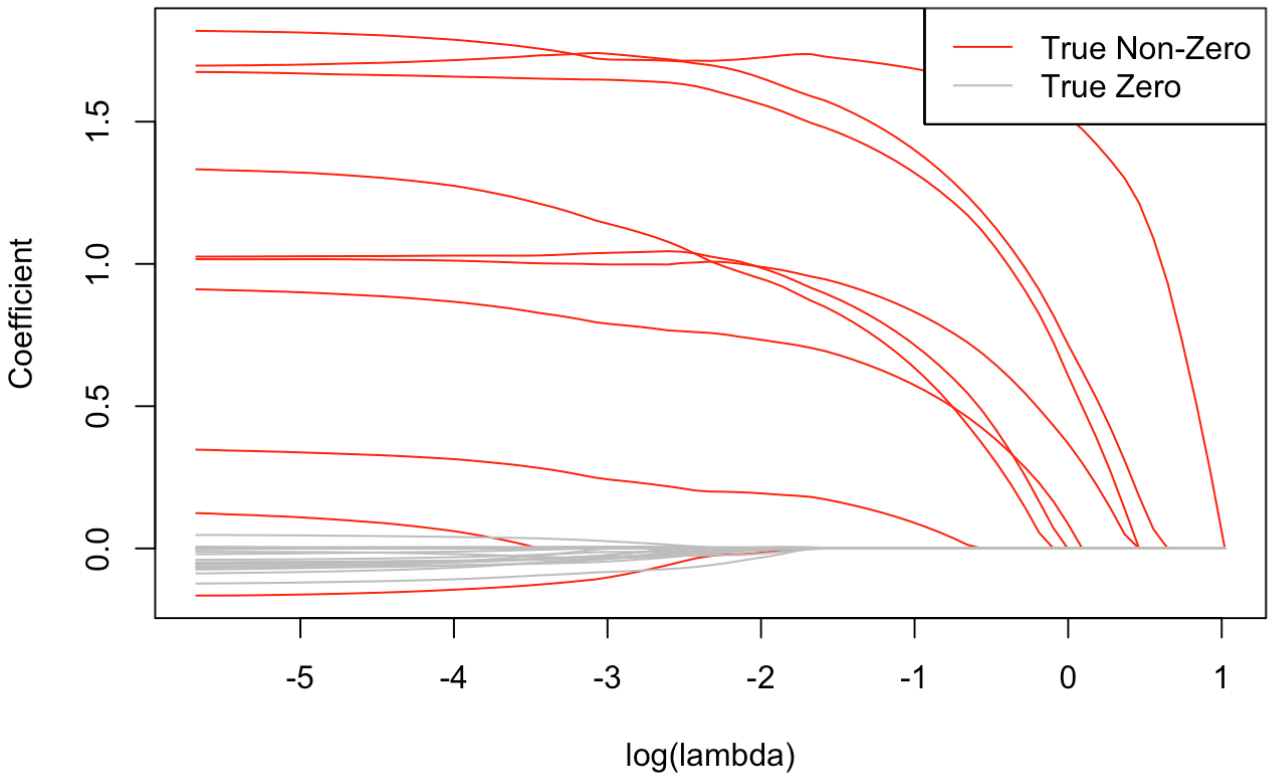
- The L_1 norm or taxicab norm, $\|x\|_1 = |x_1| + |x_2| + \cdots + |x_p|$ for $x \in \mathbb{R}^p$
- In $\mathbb{R}^2$, distance between (x_1, y_1) and (x_2, y_2) is $\|(x_1, y_1) - (x_2, y_2)\|_1 = |x_1 - x_2| + |y_1 - y_2|$



from wikipedia

- Using `alpha=1` in `glmnet` gives lasso parameter estimates

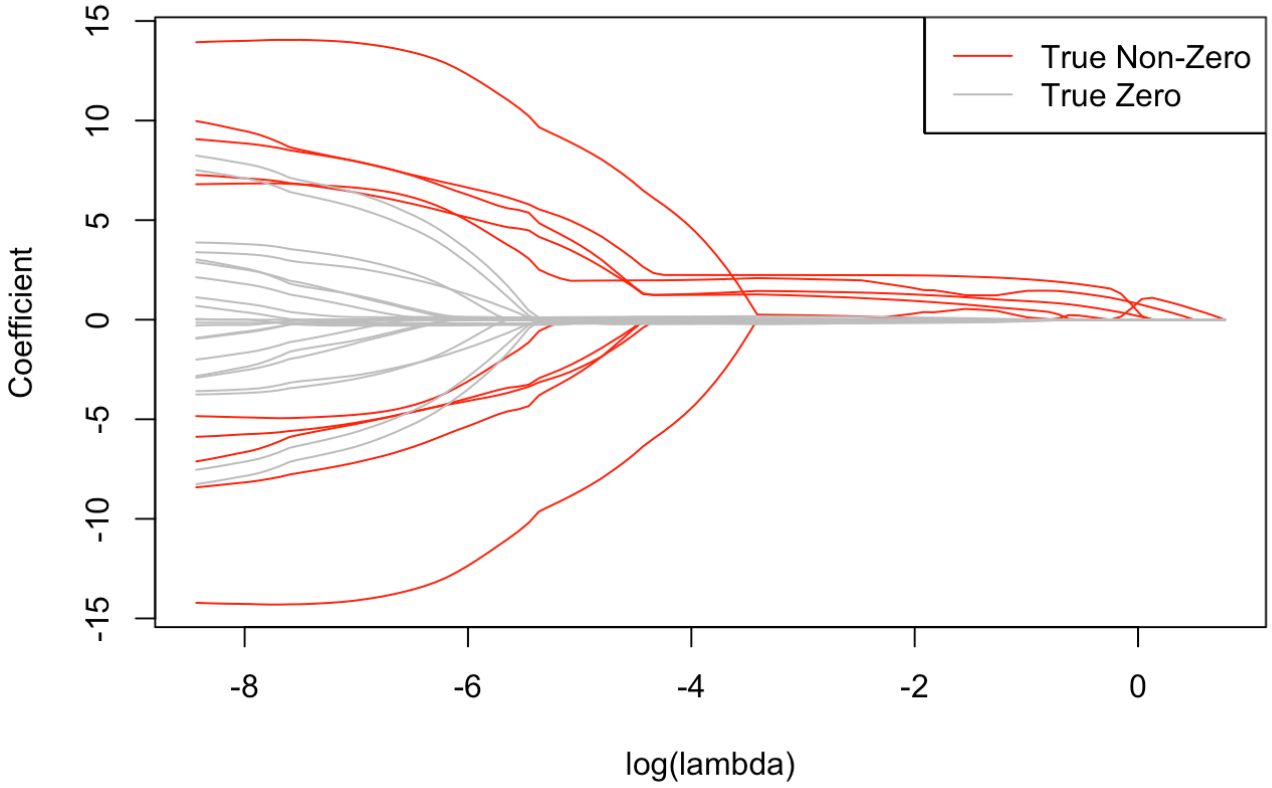
```
l1_fit <- glmnet(design_mat, y, family="gaussian", alpha=1) # alpha=1 gives lasso
glmnet_plot(l1_fit, num_active_vars)
```



- L_1 penalty generates sparse β (many zeros)
- As λ gets bigger, β shrinks and some go to exactly zero
- Because of the absolute value, this function is not differentiable everywhere, standard optimization uses coordinate descent, also fast
- if $p > n$, lasso selects at most n variables
- if using grouped (dummy) variables (like race or other categorical variables with more than 2 levles), lasso will ignore groups
- Question: Why is LASSO helpful? (short answer)

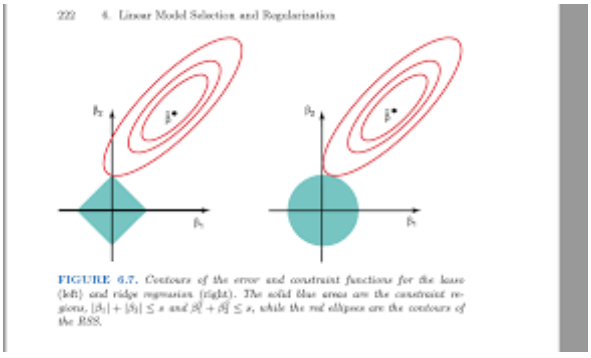
- Can be erratic on collinear data (covariates associated)

```
set.seed(1234)
dm_half <- design_mat[, c(1:5, 11:21)] # taking half of active and null columns
collin_vars <- dm_half+ 0.1*matrix(rnorm(dim(dm_half)[1]*dim(dm_half)[2]), nrow=sample_size
) #collinear vars
collin_dm <- cbind(dm_half[,1:5], collin_vars[,1:5], dm_half[,6:16], collin_vars[,6:16])
l1_fit_collin <- glmnet(collin_dm, y, family="gaussian", alpha=1)
glmnet_plot(l1_fit_collin, num_active_vars)
```



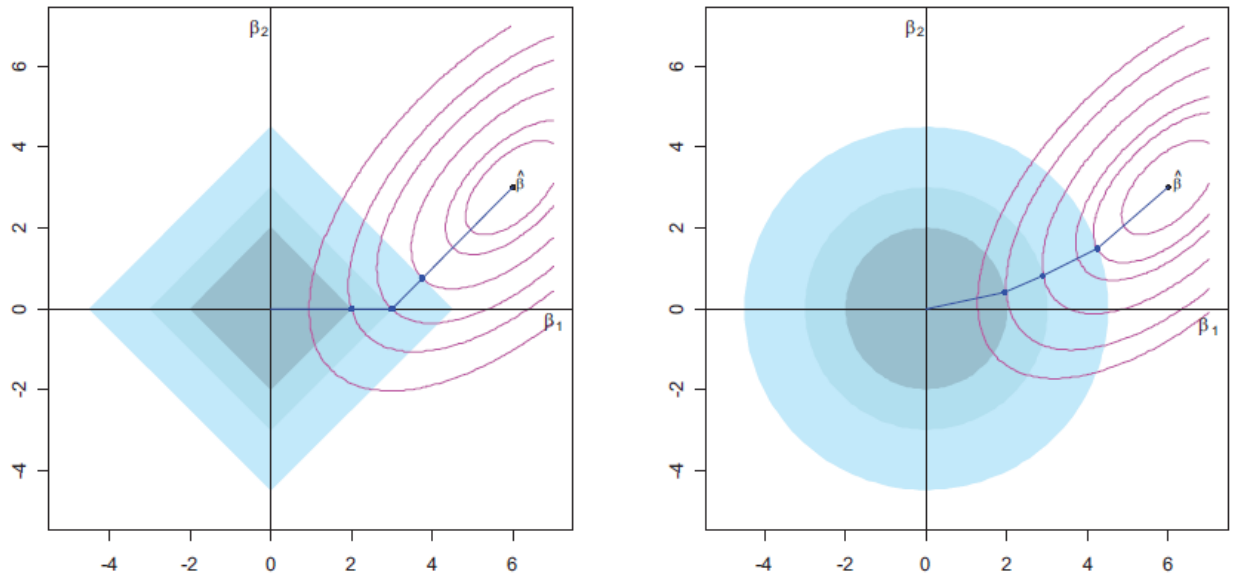
Difference between Ridge and LASSO

- Why does the L_1 -norm penalty in LASSO zero out some β values but not for the L_2 -norm penalty in Ridge?



from Elements of Statistical Learning

- Think of xy-plane as all possible values of $\beta = (\beta_1, \beta_2)$
- z-axis is the value of $\left\|y - X^T\beta\right\|_2^2 + \lambda\|\beta\|_1$
- The optimization problem wants to make $\left\|y - X^T\beta\right\|_2^2 + \lambda\|\beta\|_1$ small
- Least squares is quadratic as a function of β, $h(\beta_1, \beta_2) = \sum_{i=1}^{n}(y_i - X_i^T\beta)^2$, forms a paraboloid (https://en.wikipedia.org/wiki/Paraboloid), or bowl shape
- $g(\beta_1, \beta_2) = \|\beta\|_1$ is an upside down square pyramid with its bottom point at the origin
    - e.g. a point, b, on the line between $(2, 0)$ and $(0, 2)$, $\|b\|_1 = 2$
    - the level contours create empty diamond-shapes ($\{(x_1, x_2) : \|(x_1, x_2)\|_1 = z\}$ is a diamond-shape)
- By adding these two functions, the optimization must balance the contribution from each
    - h wants the paraboid min and g wants the origin
    - Because g has a sharp point, the lowest contour of h will likely hit there first

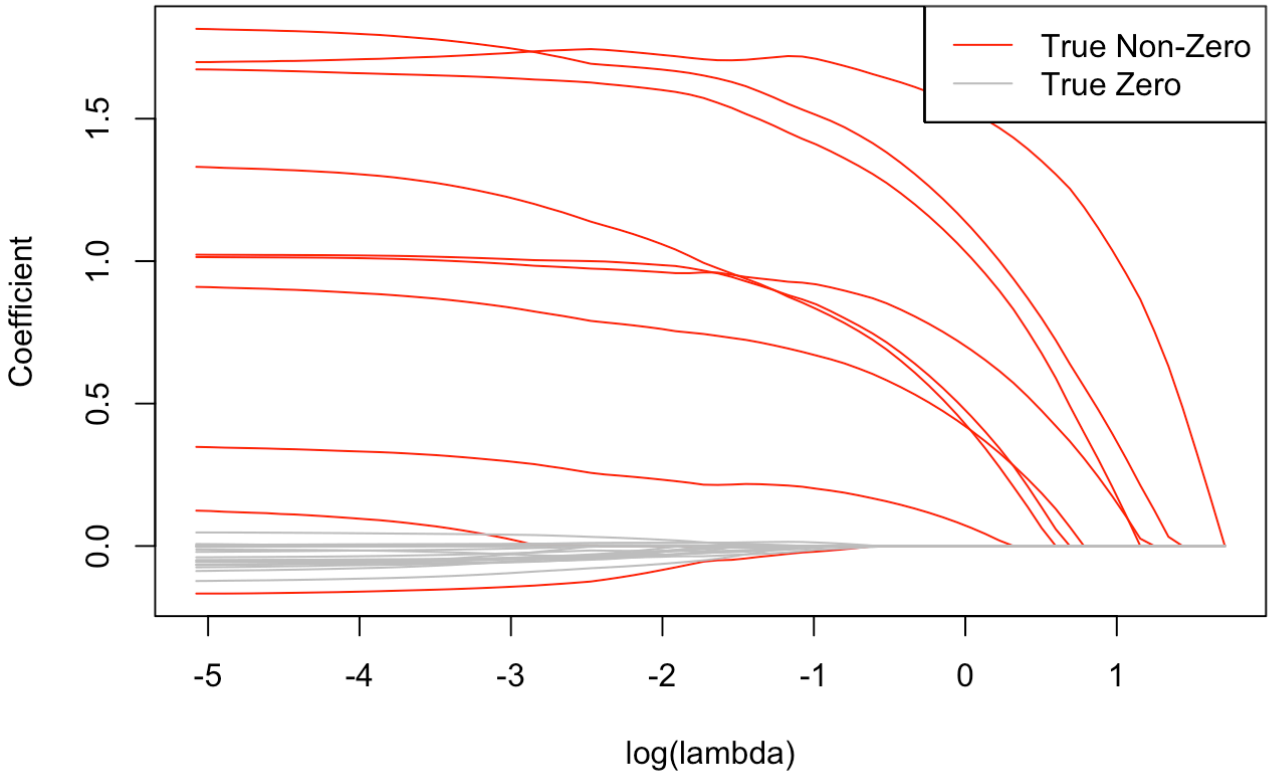


like in this image

Elastic Net

$$\hat{\beta} = \arg\min_{\beta} \left(y - X^T\beta\right)^2 + \lambda\left[(1-\alpha)\|\beta\|_2^2/2 + \alpha|\beta\|_1\right]$$

```
l5_fit <- glmnet(design_mat, y, family="gaussian", alpha=0.5)
glmnet_plot(l5_fit, num_active_vars)
```



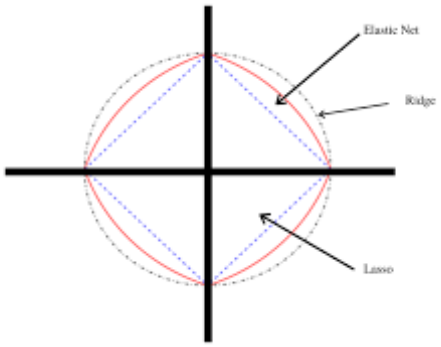


image from here (https://corporatefinanceinstitute.com/resources/knowledge/other/elastic-net/)

- L_1 penalty generates sparse model
- L_2 penalty
  - number of selected variables not bounded by n
  - stabilizes L_1 regularization path

Penalized Regression as Bayesian Method

- Bayes' Theorem

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

where $P(E) \neq 0$

- Note: Bayes' theorem is also frequently used outside of Bayesian methods
- Bayesian: parameter, θ is a random variable
- Frequentist: parameter, θ unknown, but fixed value
- Question: How well you feel like you already understand Bayesian inference?
    - A. Not at all
    - B.
    - C. Somewhat
    - D.
    - E. Very well

- **Bayesian inference goal:** estimate $P(\theta|x_1, \dots, x_n)$, the probability of parameter given data, by estimating the likelihood $P(x_1, \dots, x_n|\theta)$ assuming the parameter has some prior density, $P(\theta)$
- $P(\theta|x_1, \dots, x_n) := \pi(\theta|x_1, \dots, x_n)$, the **posterior** probability, is the probability θ given evidence (data) $x_1, \dots, x_n$
- $P(\theta) := \pi(\theta)$, the **prior** probability, is the probability of θ before evidence (data) is observed
- $P(x_1, \dots, x_n|\theta)$ is the **likelihood**; density is $f(x; \theta)$, likeihood is $f(x_1, \dots, x_n; \theta) = \prod_{i=1}^{n} f(x_i; \theta)$ because of independence
- $P(x_1, \dots, x_n) := m(x)$ is the marginal likelihood but is frequently used to make sure that $\int_\theta \frac{f(x_1,\dots,x_n;\theta)\pi(\theta)}{m(x)} d\theta = 1$
- Set $m(x) = \int_\theta f(x_1, \dots, x_n; \theta)\pi(\theta)d\theta$, a constant in terms of θ
- Calculate posterior:

$$\pi(\theta|x_1, \dots, x_n) = \frac{f(x_1, \dots, x_n|\theta)\pi(\theta)}{m(x)} \propto \prod_{i=1}^{n} f(x_i; \theta)\pi(\theta)$$

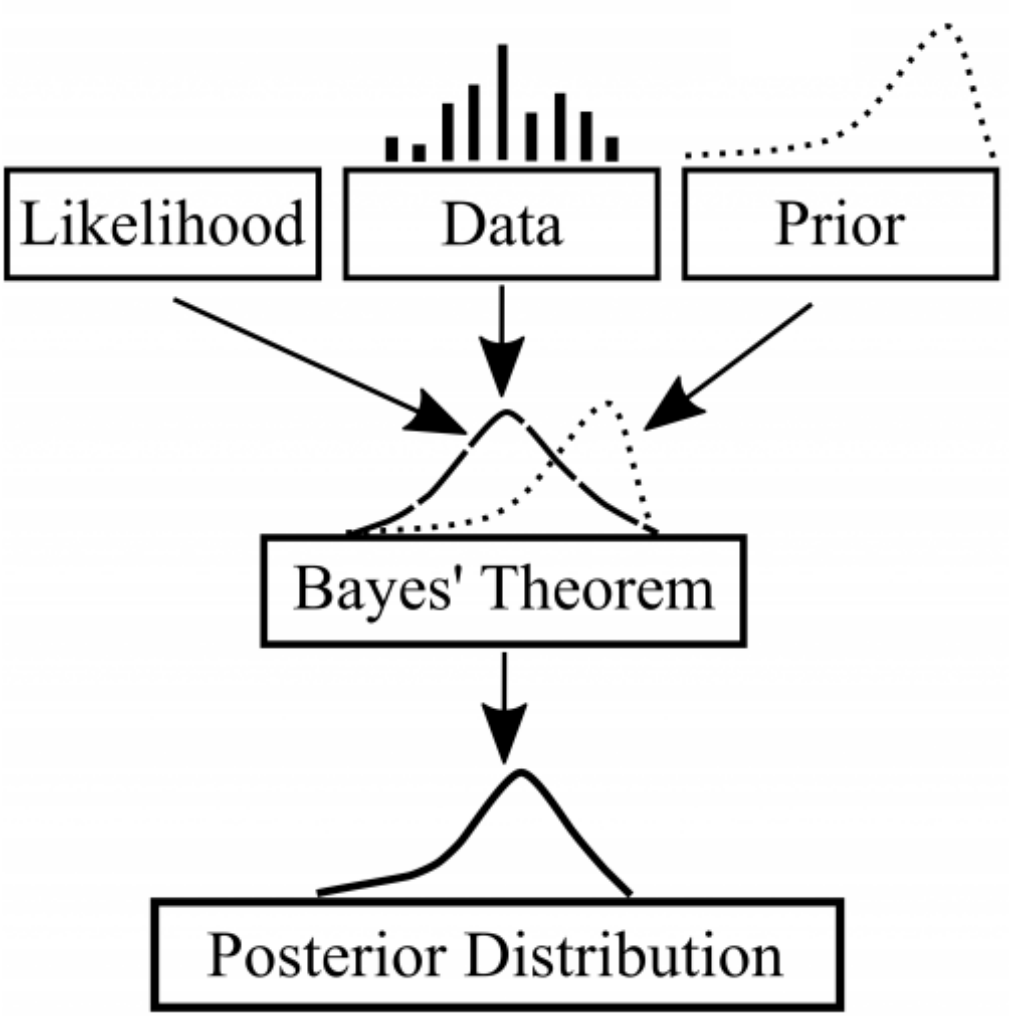


Image from Medium.com

- θ is still a random variable, with updated density:

$$\pi(\theta|x_1, \dots, x_n) = \frac{f(x_1, \dots, x_n|\theta)\pi(\theta)}{m(x)}$$

- Usually, only need mean and variance of posterior for point estimate and confidence interval
- Example: Let $X_1, \dots, X_n \sim \text{Bernoulli}(p)$ and $Y = \sum_i X_i$ (binomial random variable).
- Goal: Estimate p.
- Assume $p \sim \text{Beta}(\alpha, \beta)$ is a prior distribution for p. That is,

$$\pi(p) = \frac{p^{\alpha-1}(1-p)^{\beta-1}}{B(\alpha, \beta)}$$

  where B is the beta function. Then

$$\pi(p|Y) \propto \underbrace{\binom{n}{Y} p^Y (1-p)^{n-Y}}_{\text{likelihood}} \times \underbrace{\frac{p^{\alpha-1}(1-p)^{\beta-1}}{B(\alpha,\beta)}}_{\text{prior}} \propto p^{Y+\alpha-1}(1-p)^{n-Y+\beta-1}$$

- Then

$$p|Y \sim \text{Beta}(Y+\alpha, n-Y+\beta)$$

so that

$$\hat{p} = \frac{Y+\alpha}{\alpha+\beta+n}$$

using the mean of a Beta distributed (https://en.wikipedia.org/wiki/Beta_distribution) random variable

- Penalized regression was introduced as a frequentist method in 1996
- Park and Casella (https://people.eecs.berkeley.edu/~jordan/courses/260-spring09/other-readings/park-casella.pdf) showed that it was equivalent to a Bayesian paradigm where parameters have a mean-zero Gaussian prior (ridge regression) or a meanzero-Laplace prior (lasso)
- Lasso as an optimization:

$$\beta_{\text{Lasso}} = \arg\min_{\beta} \|y - X\beta\|^2 + \lambda\|\beta\|_1$$

- Lasso as Bayesian: $y \sim \text{Gaussian}(X\beta, \sigma^2 I_n)$ with prior, $\beta_j \sim \frac{\lambda}{2\sigma} e^{-\lambda|\beta_j|/\sigma}$ (Laplace distribution)
- Ridge as optimization:

$$\beta_{\text{Ridge}} = \arg\min_{\beta} \|y - X\beta\|^2 + \lambda\|\beta\|_2^2$$

- Ridge as Bayesian: $y \sim \text{Gaussian}(X\beta, \sigma^2 I_n)$ with prior, $\beta_j \sim \text{Gaussian}\left(0, \frac{\sigma^2}{\lambda^2}\right)$ (Gaussian distribution)

$$\begin{aligned}\text{Posterior Distribution} &= \underbrace{\prod_{i=1}^{n} \text{Gaussian}(y_n|X\beta, \sigma^2 I_n)}_{\text{likelihood}} \times \underbrace{\text{Gaussian}\left(\beta \middle| 0, \frac{\sigma^2}{\lambda^2} I_p\right)}_{\text{prior}} \\ &\propto \exp\left(-\sum_{i=1}^{n} \frac{(y_i - X_i\beta)^2}{2\sigma}\right) \times \exp\left(-\frac{\lambda}{2\sigma}\sum_{j=1}^{p}\beta_j^2\right) \\ &= \exp\left(-\frac{1}{2\sigma}\|y - X\beta\|_2^2 - \frac{\lambda}{2\sigma}\|\beta\|_2^2\right)\end{aligned}$$

- Optimize the posterior over β:

$$\arg\max_{\beta} \exp\left(-\frac{1}{2\sigma}\|y - X\beta\|^2 - \frac{\lambda}{2\sigma}\|\beta\|_2^2\right) = \arg\min_{\beta}\left(\|y - X\beta\|^2 + \lambda\|\beta\|^2\right)$$

Overfitting and Model Validation

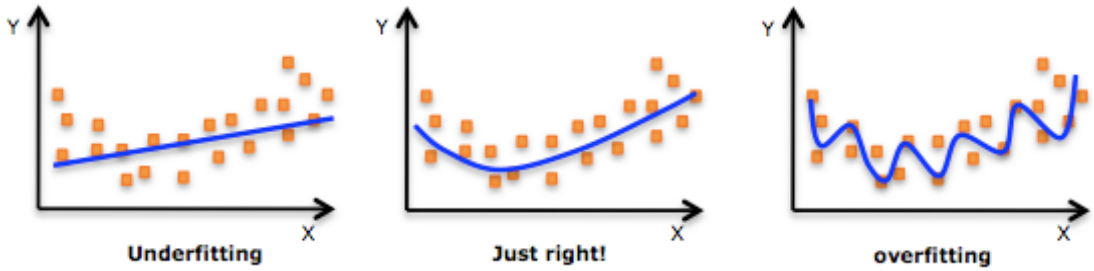


image from here (https://www.quora.com/What-are-the-key-trade-offs-between-overfitting-and-underfitting)

- Question: Which of the following is likely going to overfit? (select multiple)
  - A. Determining a GLM model and variables first, then fitting the model
  - B. Using the data with AIC to choose the best GLM model and subset of variables
  - C. Using the data with penalized regression to choose the best value of λ
  - D. Using the data to fit a regression tree

- Data generation paradigm:

$$Y = \underbrace{f(X)}_{\text{signal}} + \underbrace{\epsilon}_{\text{noise}}$$

- Signal: Typically want $f(X) = E[Y|X]$
- Noise: Variation in Y that cannot be explained using X, generally $X \perp \epsilon$
- Prediction goal: estimate f as closely as possible with a model $\hat{f}$
- Overfitting uses variation in ϵ to explain Y
- Question: Which model is most likely to overfit? (choose one)
  - A. `Y ~ X1 + X2`
  - B. `Y ~ X1 + X2 + X3`
  - C. `Y ~ X1 + X2 + X3 + X4`
  - D. Not sure
- Question: Which is most likely to underfit? (choose one)

- Complex models, with more parameters, are more likely to overfit
- Simple models, with less parameters, are more likely to underfit (miss signal)
- Issue: how much complexity is too much?

- Question: Does a large λ correspond more or less complexity?
    - A. More complexity
    - B. Less complexity
    - C. Not sure
- How do we assess fit?

Validation

- If a model has good fit, it should be able to make accurate predictions on *new* data
- Accuracy is determined by a *loss function* (typically chosen by Y data type)
    - Mean squared error: $L(\hat{Y}, Y) = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)$
    - Zero-one loss: $L(\hat{Y}, Y) = \frac{1}{n}\sum_{i=1}^{n} I(\hat{Y}_i \neq Y_i)$

Held out validation:

1. Randomly split data into training set, $(X_{\text{tr}}, Y_{\text{tr}})$, and test set, $(X_{\text{te}}, Y_{\text{te}})$ (why random?)
2. Fit models, $\hat{f}_1, \ldots, \hat{f}_K$, on the training set, $(X_{\text{tr}}, Y_{\text{tr}})$
3. For each model, $\hat{f}_1, \ldots, \hat{f}_K$, use the X values from the training set to get predicted Y values, $\hat{f}_1(X_{\text{te}}), \ldots, \hat{f}_k(X_{\text{te}})$
4. Evaluate predictions using loss function, $L(\hat{f}_1(X_{\text{te}}), Y_{\text{te}}) \ldots, L(\hat{f}_K(X_{\text{te}}), Y_{\text{te}})$ and choose simplest model with smallest loss

```
# this code is for learning
# don't do it this way in practice
set.seed(1234)
# step 1
test_ind <- sample(1:sample_size, 10, replace=FALSE)
test_ind
```

```
##  [1] 28 16 22 37 44  9  5 38 49  4
```

```
train_ind <- (1:sample_size)[! 1:sample_size %in% test_ind]
train_ind
```

```
##  [1]  1  2  3  6  7  8 10 11 12 13 14 15 17 18 19 20 21 23 24 25 26 27 29 30 31
## [26] 32 33 34 35 36 39 40 41 42 43 45 46 47 48 50
```

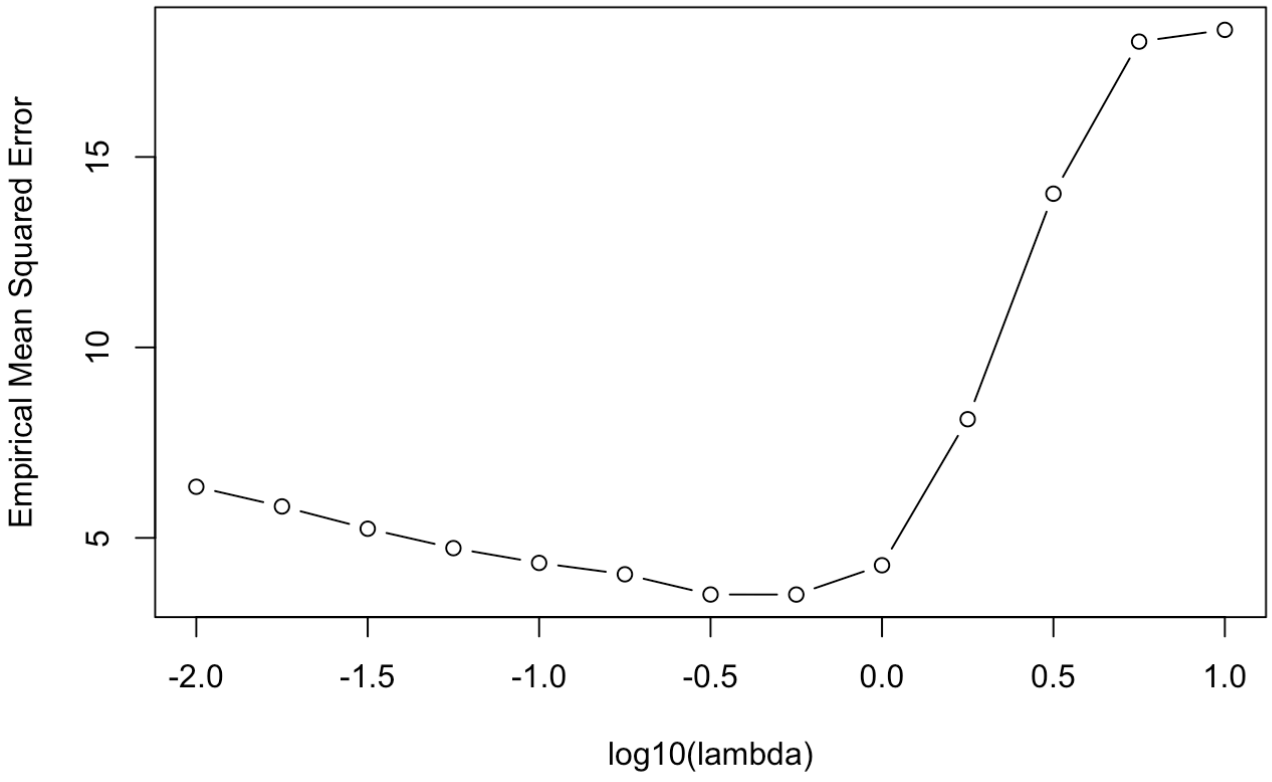
```
# step 2
lambdas <- 10^(seq(-2, 1, by=0.25))
train_fit <- glmnet(design_mat[train_ind,], y[train_ind], family="gaussian", alpha=0.5, lam
bda=lambdas)

# step 3
test_preds <- predict(train_fit, newx=design_mat[test_ind,], type='response')
test_preds
```

```
##            s0        s1          s2          s3         s4          s5
##  [1,] 1.03722 0.9799444  0.04124718 -2.13321088 -4.6132609 -5.89159016
##  [2,] 1.03722 1.0080332  0.70093102  0.09122676  0.2152240  0.07747343
##  [3,] 1.03722 1.0869251  2.13017022  4.12973971  4.6224140  4.67001125
##  [4,] 1.03722 1.0426066  1.14097945  1.28362668  0.2544462 -0.46344845
##  [5,] 1.03722 1.0546809  1.29727505  2.01094871  2.8265896  3.20813284
##  [6,] 1.03722 0.9046631 -1.15138925 -4.52055477 -6.8662968 -8.35131462
##  [7,] 1.03722 1.0583218  1.16285464 -0.20734041 -1.1782192 -1.71675234
##  [8,] 1.03722 0.9044524 -1.08515044 -2.23548545 -2.4349246 -2.67741218
##  [9,] 1.03722 1.1011828  2.03163064  2.10824104  1.8438863  1.62696710
## [10,] 1.03722 1.0479090  1.16670290  1.48476698  1.3522061  1.40061480
##                s6         s7          s8          s9         s10         s11
##  [1,] -6.49116681 -6.9638938  -7.2961855  -7.3165097  -7.2382415  -6.9887669
##  [2,]  0.01922609 -0.1142625  -0.4268567  -0.9615482  -1.4507142  -1.8457500
##  [3,]  4.57018627  4.3095933   3.9242865   3.3545116   2.8950431   2.5890005
##  [4,] -0.63554424 -0.4637328  -0.7591467  -1.0338734  -1.0972837  -1.1121568
##  [5,]  3.48088787  3.7975922   4.0542326   3.8981791   3.7690801   3.6036003
##  [6,] -9.12185418 -9.7381413 -10.1625744 -10.2693985 -10.2798794 -10.2292054
##  [7,] -2.10413303 -2.1243873  -1.9059435  -1.7056224  -1.4587594  -1.3625702
##  [8,] -2.78641409 -3.0023223  -3.1662041  -3.3177372  -3.4355102  -3.7107626
##  [9,]  1.49403911  1.5290357   1.2380084   0.8888110   0.6175895   0.4505577
## [10,]  1.46239504  1.2137308   0.9433404   0.8284501   0.7543063   0.6846772
##               s12
##  [1,]  -6.6470625
##  [2,]  -2.1718833
##  [3,]   2.3138798
##  [4,]  -1.2035047
##  [5,]   3.4399662
##  [6,] -10.1478933
##  [7,]  -1.3544433
##  [8,]  -3.9677990
##  [9,]   0.3609254
## [10,]   0.6304905
```

```
# step 4
# calculating mean squared error for each model
mses <- apply(test_preds, 2, function(x) mean((x-y[test_ind])^2))
lambdas <- train_fit$lambda
cbind(lambdas, mses) # showing MSE with lambda value
```

```
##         lambdas      mses
## s0  10.00000000 18.337272
## s1   5.62341325 18.028366
## s2   3.16227766 14.035926
## s3   1.77827941  8.115448
## s4   1.00000000  4.281505
## s5   0.56234133  3.512273
## s6   0.31622777  3.513439
## s7   0.17782794  4.044139
## s8   0.10000000  4.342859
## s9   0.05623413  4.730743
## s10  0.03162278  5.240411
## s11  0.01778279  5.825288
## s12  0.01000000  6.341917
```

```
plot(log10(train_fit$lambda), mses, type='b', ylab='Empirical Mean Squared Error', xlab='lo
g10(lambda)')
```



- Comparing estimated values with true values
- Which are bigger?

```
true_vals <- c(0,true_beta, rep(0, 20))
cbind(coef(train_fit, s=0.56234133), true_vals)
```

```
## 31 x 2 sparse Matrix of class "dgCMatrix"
##                       1  true_vals
## (Intercept) -0.84449896 .
## V1           .          0.22740682
## V2           1.12811732 1.24459881
## V3           0.67991453 1.21854947
## V4           0.64046300 1.24675888
## V5           1.40340133 1.72183077
## V6           1.98831224 1.28062121
## V7          -0.04047174 0.01899151
## V8           0.09139017 0.46510101
## V9           0.87757184 1.33216752
## V10          0.39330737 1.02850228
## V11          .          .
## V12          .          .
## V13         -0.04207614 .
## V14          0.02194223 .
## V15          .          .
## V16          .          .
## V17          .          .
## V18          .          .
## V19          .          .
## V20          .          .
## V21          .          .
## V22          .          .
## V23          0.03878893 .
## V24          .          .
## V25          .          .
## V26          0.08983453 .
## V27          .          .
## V28          .          .
## V29          .          .
## V30          .          .
```

Cross Validation

- Normally used over held out validation
- Runs held out validation k-fold times

Steps:

1. Partition data into k folds
2. Run held out validation k times:

i. On step j, remove fold j, train models on all other folds

ii. Get predictions with trained models on *j*th fold
iii. Save predictions

3. Evaluate model using loss function

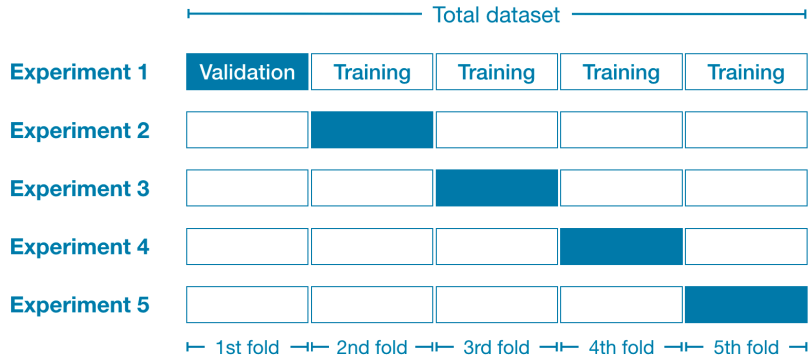
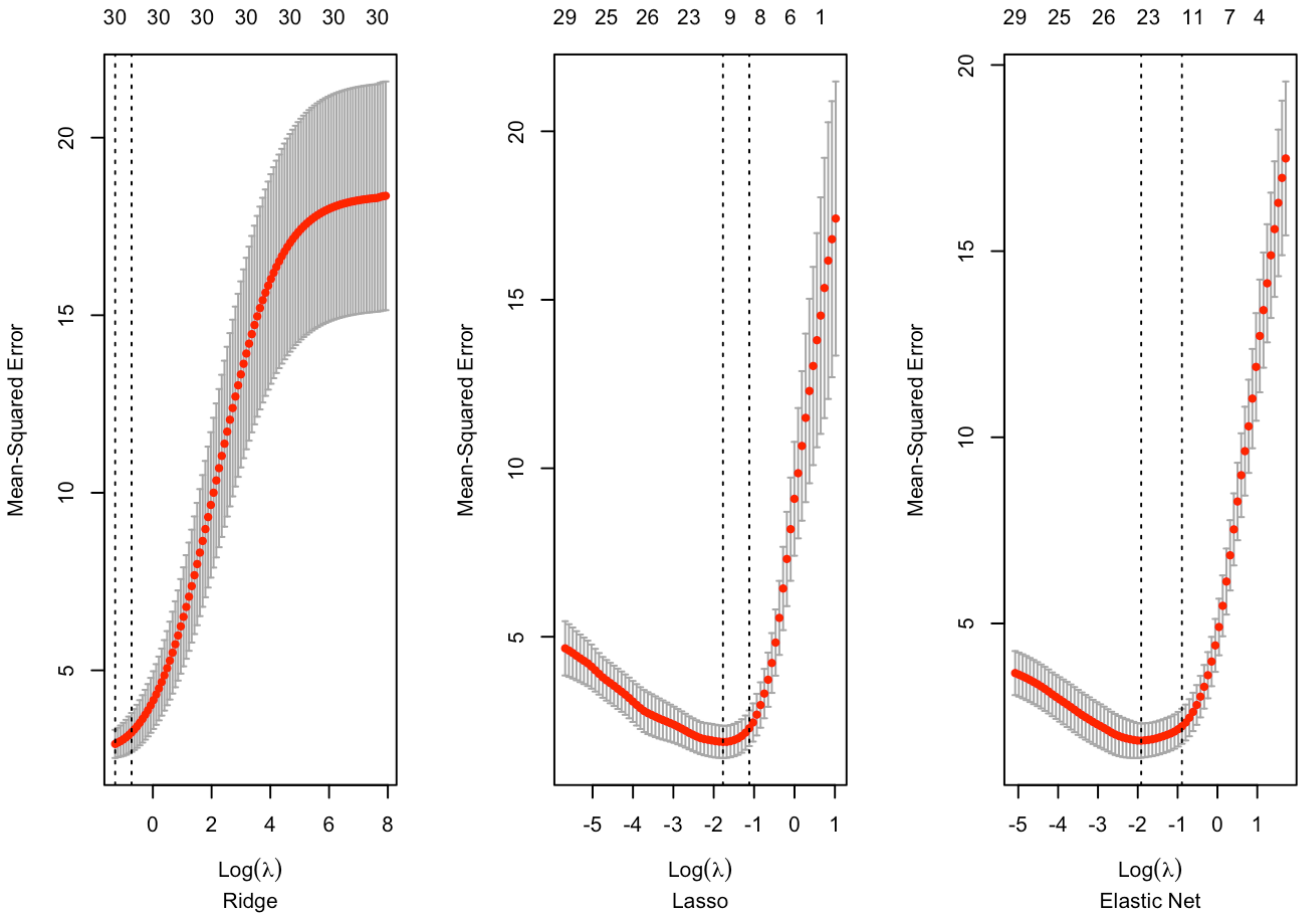


image from here (https://www.kaggle.com/alexisbcook/cross-validation)

- glmnet does all of this for you

```
cv_fit0 <- cv.glmnet(design_mat, y, family="gaussian", alpha=0, nfolds=5)
cv_fit1 <- cv.glmnet(design_mat, y, family="gaussian", alpha=1, nfolds=5)
cv_fit5 <- cv.glmnet(design_mat, y, family="gaussian", alpha=0.5, nfolds=5)

par(mfrow=c(1,3))
plot(cv_fit0, sub='Ridge')
plot(cv_fit1, sub='Lasso')
plot(cv_fit5, sub='Elastic Net')
```



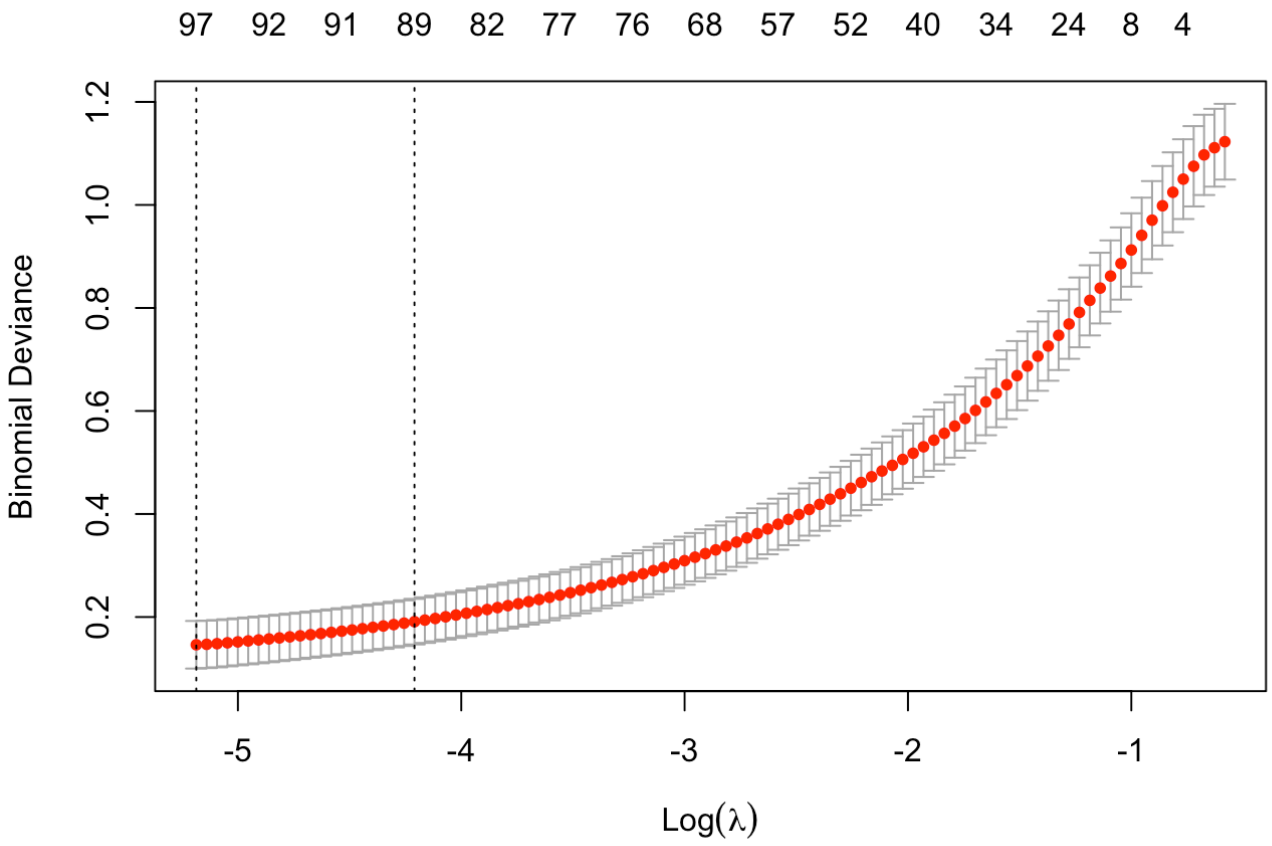
- Number above plot indicates number of nonzero coefficients at current λ

Gene data with penalized regression

Question: Which penalized regression model should we use? (choose one) - A. Ridge - B. Lasso - C. Elastic Net - D. Not sure

- Because there are not many rows of data (71), first using cross validation on all data to choose λ
  - Then using the optimal λ on entire dataset
  - If there were more data, I would set aside more data to do the final fit

```
gene_mat <- as.matrix(shipp$x)
# changing outcome to 0's and 1's
lymph_type <-ifelse(shipp$y == 'FL', 1, 0)

# running cross validation using elastic net
gene_fit_cv <- cv.glmnet(gene_mat, shipp$y, family="binomial", alpha=0.5, nfolds=5)
plot(gene_fit_cv)
```




```
# optimal value of lambda
gene_fit_cv$lambda.min
```

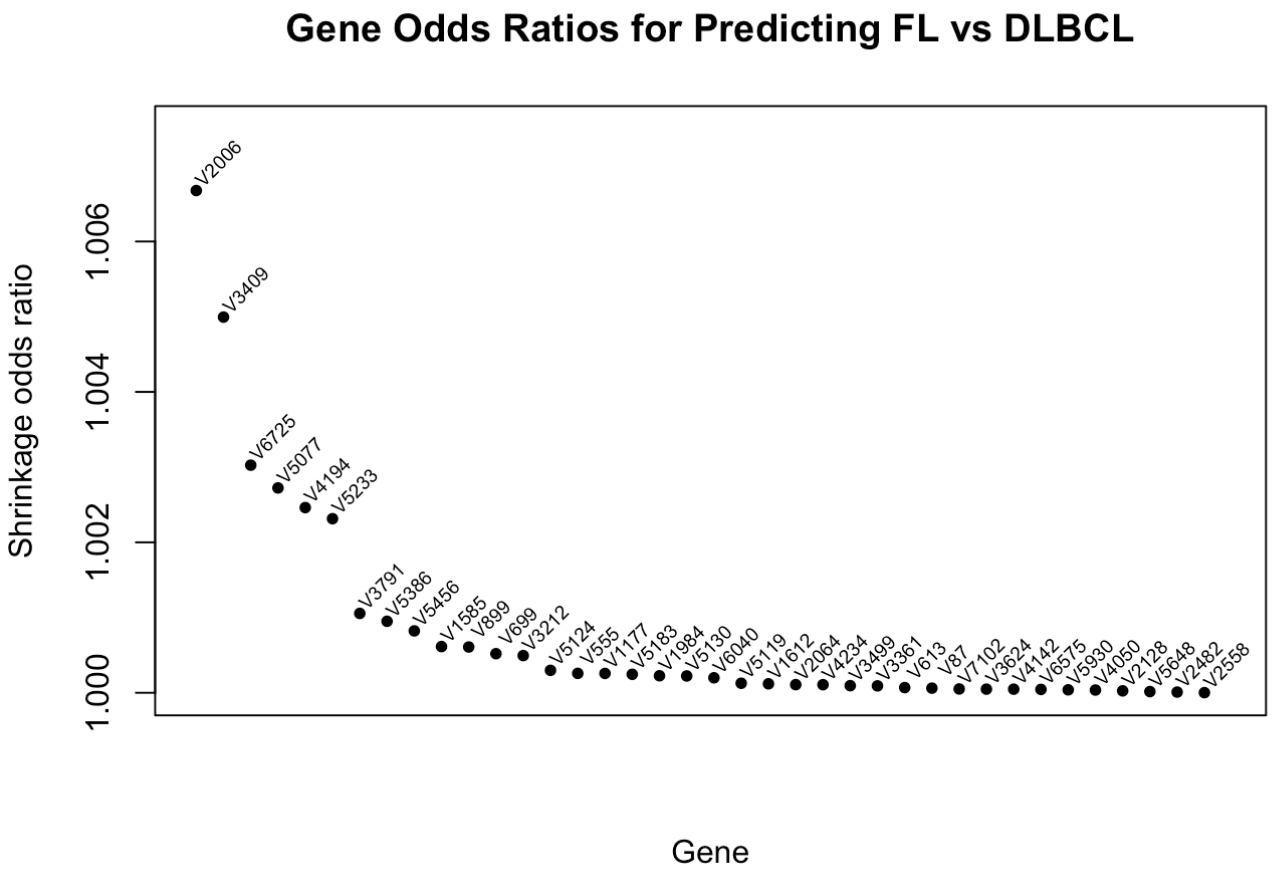
```
## [1] 0.005592006
```

```
# running Lasso on all data with optimal lambda
gene_fit <- glmnet(gene_mat, lymph_type, family="binomial", alpha=0.5, lambda = gene_fit_cv
$lambda.min)
```

```
nonzero <- order(gene_fit$beta, decreasing = T)[1:sum(gene_fit$beta >0)]
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

```
#nonzero <- which(gene_fit$beta >0)
exp_beta <- exp(gene_fit$beta[nonzero])
names(exp_beta) <- names(shipp$x)[nonzero]
plot(exp_beta, ylab='Shrinkage odds ratio', xlab='Gene', xaxt='n', pch=20, xlim=c(1,length(
nonzero)+0.75), ylim=c(1, 1.0075), main='Gene Odds Ratios for Predicting FL vs DLBCL')
text(1:length(nonzero)+1, exp_beta, names(exp_beta), cex=0.6, pos=3, offset=0.6, srt=45)
```




```
data.frame(exp_beta)
```

```
##       exp_beta
## V2006 1.006677
## V3409 1.004995
## V6725 1.003026
## V5077 1.002723
## V4194 1.002462
## V5233 1.002314
## V3791 1.001055
## V5386 1.000951
## V5456 1.000823
## V1585 1.000614
## V899  1.000609
## V699  1.000519
## V3212 1.000495
## V5124 1.000298
## V555  1.000258
## V1177 1.000257
## V5183 1.000246
## V1984 1.000226
## V5130 1.000223
## V6040 1.000199
## V5119 1.000127
## V1612 1.000119
## V2064 1.000108
## V4234 1.000107
## V3499 1.000095
## V3361 1.000092
## V613  1.000069
## V87   1.000062
## V7102 1.000051
## V3624 1.000048
## V4142 1.000048
## V6575 1.000045
## V5930 1.000039
## V4050 1.000036
## V2128 1.000025
## V5648 1.000016
## V2482 1.000009
## V2558 1.000002
```

- Interpretation: After controlling for all other genes, 38 genes were predictive for FL vs DLBCL. Table 1 in the appendix give the shrinkage odds ratios for determining FL. These values can be interpreted as magnitude controlled association with FL vs DLBCL.

Graphical Lasso

- Lasso can also be used to construct *conditional independence graphs*

- Conditional independence graphs are graphical models that connect two variables (nodes) if they are dependent after conditioning on all other variables
- Given a Bayesian network, one can construct the corresponding conditional independence graph.

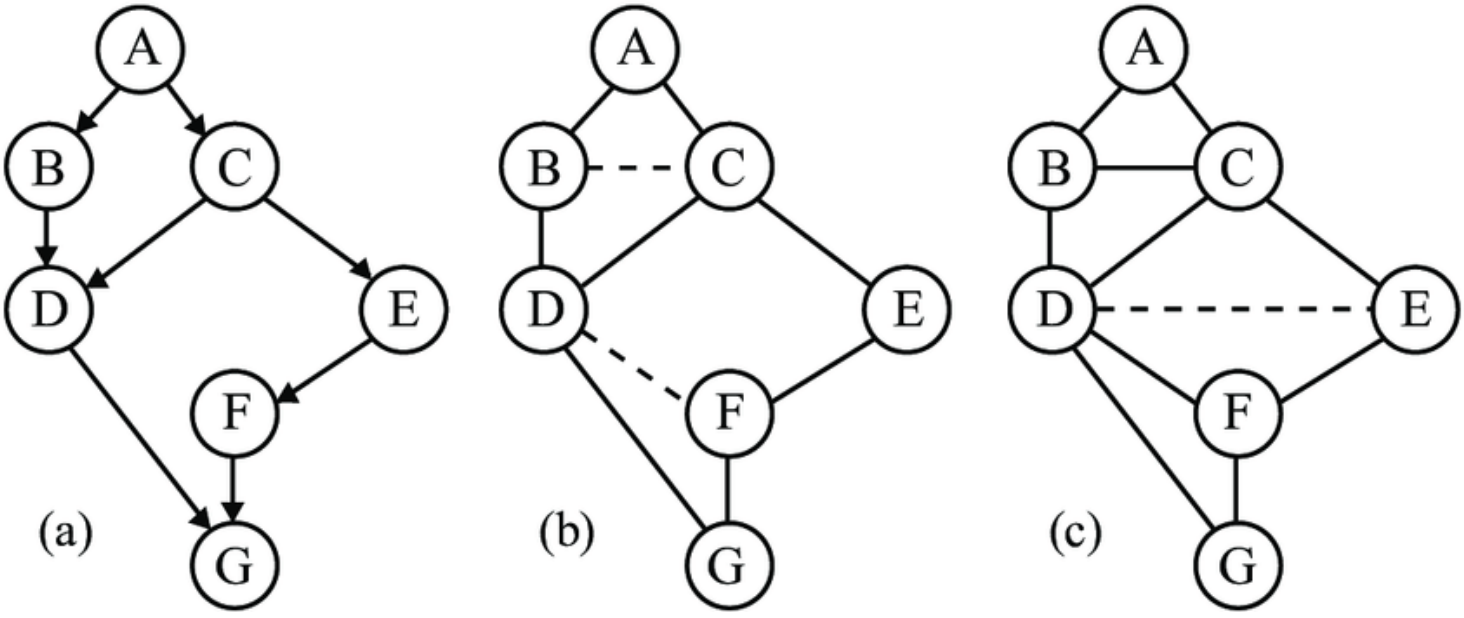


image from Kevin Gimpel

- Assume that $X = (X_1, \dots, X_p)$ is a random, $p \times 1$-vector $X \sim \text{Gaussian}(\mu, \Sigma)$, where μ is $p \times 1$ and Σ is a $p \times p$ covariance matrix
- Let $\Omega = \Sigma^{-1}$
- We can show that

$$X_j \perp X_k | \{X_1, \dots, X_p\} \backslash \{X_j, X_k\} \Leftrightarrow \Omega_{jk} = 0$$

- The log-likelihood of the inverse covariance matrix is

$$p(\Omega) = -\frac{p}{2}\log(2\pi) + \log\det\Omega - \text{tr}\left(\hat{\Sigma}\Omega\right)$$

where $\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})(x_i - \hat{\mu})^T$

- Recall that the multivariate Gaussian distribution is

$$\begin{aligned} p(x) &= \left[(2\pi)^p \det(\Sigma)\right]^{-1/2} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \\ &= (2\pi)^{-p/2}\left[\det(\Sigma^{-1})\right]^{1/2} \exp\left(-\frac{1}{2}\text{tr}\ \left((x-\mu)^T(x-\mu)\Sigma^{-1}\right)\right) \\ \log p(x) &= -\frac{p}{2}\log(2\pi) + \frac{1}{2}\log\left[\det(\Sigma^{-1})\right] - \frac{1}{2}\text{tr}\ \left(\hat{\Sigma}\Sigma^{-1}\right) \\ &= -\frac{p}{2}\log(2\pi) + \frac{1}{2}\log[\det(\Omega)] - \frac{1}{2}\text{tr}\ \left(\hat{\Sigma}\Omega\right) \end{aligned}$$

- Using a Lasso penalty on the likelihood, we the following optimization problem

$$\hat{\Omega} = \arg\max_{\Omega} \left[\log\det\Omega - \text{tr}\left(\hat{\Sigma}\Omega\right) - \lambda\|\Omega\|_1\right]$$

where $\|\Omega\|_1 = \sum_{j,k} |\Omega_{jk}|$

- The corresponding conditional independence graph will be constructed from $\hat{\Omega}$ where variables (nodes) j and k are connected if $\Omega_{jk} \neq 0$
- The code below is for running glasso the data may be too large for it

```
#library(glasso)
#glasso(cov(shipp$x), rho=0.5, nobs=dim(shipp$x)[1])
```