# Shrinkage

## 1 Introduction

- Given: $Y = \mathbf{X}\beta + \epsilon, \epsilon \sim N_n(0, \sigma^2 I)$ as usual. Again we assume the number of predictors (denoted by $d$) is "large", so that OLS using all of the predictors will tend to give poor predictions.
- We could use one of our algorithms for model selection. But forward/backward selection are not guaranteed to find the best model. If $d$ is larger than say 50, best subsets regression takes too long to run. In many modern datasets $d$ is now in the hundreds of thousands if not larger.
- These notes cover two more computationally feasible approaches for regression with many variables: ridge and lasso. Both methods add a penalty to the least squares criterion which **shrinks** coefficients towards zero. The lasso penalty does a form of model selection by encouraging many of the coefficients to be exactly equal to 0, a property that is called **sparsity**.
- As with model selection, the choice of method should be motivated by the particular goal, whether that be interpretability, prediction, valid inferences, or something else.

## 2 Multicollinearity and variance inflation factor

- One motivation behind model selection is to improve the accuracy of the resulting model by decreasing the number of predictors in the model. Last time I argued that increasing the number of predictors in the model can degrade the accuracy by increasing the variance of OLS. Let's make this a bit more precise.
- Concretely, suppose that in reality only the $j$th predictor was relevant:

  $$\mathbb{E}[Y_i|X_i = x] = \beta_0 + 0 \cdot x_1 + 0 \cdot x_{j-1} + \beta_j x_j + 0 \cdot x_{j+1} + \dots$$

  If an oracle told us this in advance, we would have chosen to compute the simple linear regression of $Y$ on only predictor $j$. The resulting estimated coefficient, call it $\hat{\gamma}_j$, has variance

  $$\mathrm{Var}[\hat{\gamma}_j|\mathbf{X}] = \frac{\sigma^2}{n s_{X_j}^2},$$

  where $s_{X_j} = \frac{1}{n}\sum_{i=1}^n (X_{ij} - \bar{X}_j)^2$ is the empirical variance of predictor $j$.
- Of course there have been no oracles for 2500 years. Not knowing which of the predictors is actually relevant, suppose we instead computed the full regression of $Y$ on all the predictors. The resulting estimated coefficient for predictor $j$ has variance

  $$\mathrm{Var}[\hat{\beta}_j|X_1, \dots, X_n] = \sigma^2 (\mathbf{X}^\top \mathbf{X})^{-1}_{j+1,j+1}.$$

  The ratio of these two quantities is called the **variance inflation factor**:

  $$\mathrm{VIF}_j = \frac{\mathrm{Var}[\hat{\beta}_j|\mathbf{X}]}{\mathrm{Var}[\hat{\gamma}_j|\mathbf{X}]}.$$

- The variance inflation factor measures how much the variance of a given estimated coefficient increases when other predictors are added into the model. It has a nice interpretation. Remember that in

Homework 2 you showed that

$$\text{Var}[\hat{\beta}_j|\mathbf{X}] = \sigma^2(\mathbf{X}^\top\mathbf{X})^{-1}_{j+1,j+1} = \frac{\sigma^2}{ns^2_{r_j}},$$

where $r_j$ are the residuals in a regression of predictor $x_j$ onto all the other predictors $x_k$. This means $\text{VIF}_j = s^2_{X_j}/s^2_{r_j}$. Now, let $R^2_j$ be the R-squared of this regression. We know that $1 - R^2$ is the ratio of the variance of the residuals to the variance of the response, so in this case $1 - R^2_j = s^2_{r_j}/s^2_{X_j}$. Therefore,

$$\text{VIF}_j = \frac{1}{1 - R^2_j}.$$

We see that $\text{VIF}_j$ is always greater than 1, so that the variance of our estimate for $\beta_j$ is always increased by adding more predictors to the model. The magnitude of this increase is determined by the proportion of variance in predictor $j$ that is explained by the other predictors. This means that if the predictors are highly correlated (aka close to **multicollinear**) then the variance of $\hat{\beta}_j$ will be large.

- If we have many predictors then it is very likely that some will be highly correlated just by random chance. In fact if $d > n$ then it must be the case that the predictors are perfectly multicollinear. The upshot is that if we have many predictors – or we have few predictors that happen to be highly correlated – the variance of $\hat{\beta}_j$ will be large.
- One way to reduce the variance of the estimated coefficients is by pulling them towards a specific number (typically, 0). This approach is known as **shrinkage**. We will cover two shrinkers: ridge and the lasso.

# 3 Ridge

- The ridge estimator minimizes the sum of the least-squares criterion plus a **penalty term**:

$$\hat{\beta}_\lambda = \min_b \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - b_0 - \sum_{j=1}^{d} b_j X_{ij}\right)^2 + \lambda\sum_{j=1}^{d} b_j^2.$$

- A nice property of ridge regression is that unlike OLS it has a unique solution even if $d > n$. In fact, some vector calculus analogous to what we did for OLS confirms that this solution has the closed form

$$\hat{\beta}_\lambda = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_d)^{-1}\mathbf{X}^\top Y$$

(Exercise: explain why $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_d$ is invertible whenever $\lambda > 0$.) This means unlike with stepwise or best subsets regression, there is no need to search in a possibly heuristic fashion over a giant collection of models.

- The **penalty term** $\sum_{j=1}^{d}\beta_j^2$ shrinks the OLS coefficients towards 0. The weight of the penalty relative to the least-squares criterion is dictated by $\lambda$. When $\lambda = 0$ the penalty has no weight and the ridge solution is just OLS. The larger $\lambda$ is, as a rule, the more the coefficients get shrunk. (Although strictly speaking it is not true that $\hat{\beta}_\lambda$ is monotonically decreasing in $\lambda$.) In Section 5 we calculate exactly how this shrinkage works when $\mathbf{X}$ is an orthogonal matrix.
- Ridge regression is a linear predictor and so it is easy to compute its expectation and variance:

$$\mathbb{E}[\hat{\beta}_\lambda] = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_d)^{-1}\mathbf{X}^\top\mathbf{X}\beta$$

and

$$\text{Var}[\hat{\beta}_\lambda] = \sigma^2(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_d)^{-1}\mathbf{X}^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_d)^{-1}.$$

As $\lambda$ approaches 0 we see that these formulas converge to our usual expressions for expectation and variance of OLS. As $\lambda$ increases the ridge coefficients have larger bias but smaller variance. When $d$ is large we may want to introduce a little bias in order to decrease the variance.

- With Normal errors it is possible (in principle) to additionally calculate the sampling distribution of $\hat{\beta}_\lambda$ and use this to form confidence intervals and hypothesis tests. This is typically not done for several reasons: (1) when ridge regression is used typically we are more interested in prediction than inference (2) the linear model is itself not identifiable when $d > n$ (i.e. different values of $\beta$ lead to the same $\mathbf{X}\beta$); (3) the inferences are invalid when we use the data to pick $\lambda$.
- The parameter $\lambda$ is known as a **tuning parameter**. One way of choosing $\lambda$ is by minimizing the cross-validation (CV) estimate of prediction error. Just as with OLS, there is a shortcut formula for LOOCV with ridge. As with OLS the shortcut involves the hat matrix, which for ridge regression is $\mathbf{H}_\lambda = \mathbf{X}((\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_d)^{-1}\mathbf{X}^\top$:

$$\widehat{Err}_{CV}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_{(i)i})^2 = \frac{1}{n}\sum_{i=1}^{n}\frac{(Y_i - \hat{Y}_i)^2}{(1 - [\mathbf{H}_\lambda]_{ii})^2}$$

  The ridge hat matrix $\mathbf{H}_\lambda$ is not a projection matrix: even if $v$ belongs to the column space of $\mathbf{X}$, $\mathbf{H}_\lambda v \neq v$.
- Replacing $[\mathbf{H}_\lambda]_{ii}$ by the average diagonal element $\mathrm{tr}(\mathbf{H}_\lambda)/n$ gives the generalized CV approximation to the LOOCV estimate:

$$\widehat{Err}_{GCV}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\frac{(Y_i - \hat{Y}_i)^2}{(1 - \mathrm{tr}(\mathbf{H}_\lambda)/n)^2}$$

  It is often computationally easier to calculate the trace of $\mathbf{H}_\lambda$ rather than the diagonal elements $[\mathbf{H}_\lambda]_{ii}$.
- Recall that for $\mathbf{H} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top$ the hat matrix for OLS, the trace $\mathrm{tr}(H) = p + 1$ is equal to the degrees of freedom. For ridge we call $\mathrm{tr}(\mathbf{H}_\lambda)$ the **effective degrees of freedom**. This is a useful concept when comparing ridge to, say, OLS run on a subset of the predictors. It can be shown that $\mathrm{tr}(\mathbf{H}_\lambda) < p + 1$, so that ridge regression always has fewer effective degrees of freedom than OLS.

# 4 Lasso

- We have written ridge regression as minimizing the sum of a least-squares criterion and a penalty term. We can write best subsets selection in a similar way: using Mallow's $C_p$ as a criterion, best subsets selection solves

$$\min_{b\in\mathbb{R}^d}\frac{1}{n}\sum_{i=1}^{n}\Big(Y_i - b_0 - \sum_{j=1}^{d}b_jX_{ij}\Big)^2 + \frac{2\sigma^2}{n}\sum_{j=1}^{d}\mathbf{1}\{|b_j| \neq 0\}.$$

  Unlike the ridge penalty, the penalty term $\frac{2\sigma^2}{n}\sum_{j=1}^{d}\mathbf{1}\{|b_j| \neq 0\}$ does not "care about" the magnitude of each estimated coefficient; it only cares about whether each estimated coefficient is equal to 0 or not. This means best subsets is not a shrinker. For this reason, best subsets can sometimes have high variance. Also, it is hard to compute.
- On the other hand, ridge regression is not a method for model selection. To find a computationally tractable method for selecting a few variables and performing shrinkage, we turn to the lasso.
- The lasso is defined as the solution to

$$\hat{\beta}_\lambda = \min_b\frac{1}{2n}\sum_{i=1}^{n})(Y_i - b_0 - \sum_{j=1}^{d}X_{ij}b_j) + \lambda\sum_{j=1}^{d}|\beta_j|$$

- This looks like ridge except the penalty uses the $\ell^1$ norm of $\beta$ rather than the squared-$\ell^2$ norm. The change in penalty encourages the solution $\hat{\beta}_\lambda$ to have many values exactly equal to 0. This property is called **sparsity**. To see the geometrical picture explaining why the lasso produces sparse solutions, see the figure on the start of the next page, from the textbook *Elements of Statistical Learning*.

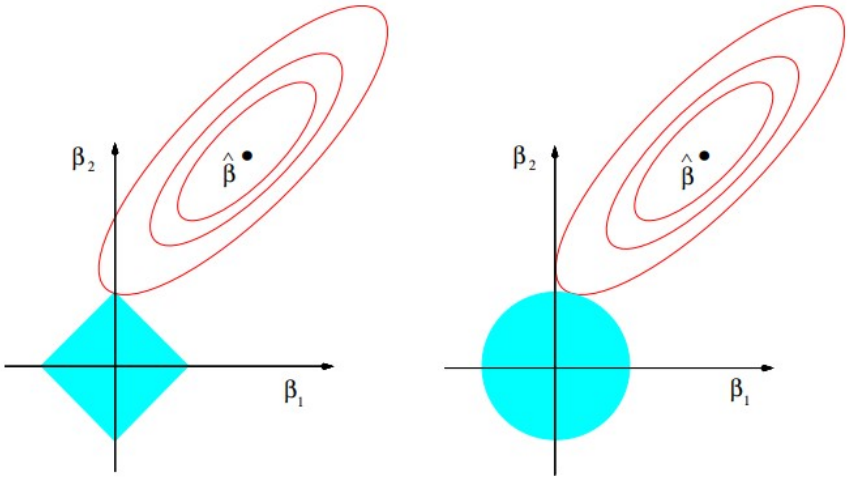



**FIGURE 3.11.** *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions* $|\beta_1| + |\beta_2| \leq t$ *and* $\beta_1^2 + \beta_2^2 \leq t^2$*, respectively, while the red ellipses are the contours of the least squares error function.*

- There is rarely closed-form solution for the lasso which makes it a bit harder to understand what is "going on". To get some insight it helpful to consider a generalization of ridge and lasso where we use a penalty on the $\ell_q$ norm of $\beta$ for some $q \in [0, 2]$:

$$\|\beta\|_q = \Big(\sum_{k=1} |\beta_k|^q\Big)^{1/q}.$$

  Ridge and lasso are special cases where $q = 2$ and $q = 1$, respectively.

  - When $q = 2$ the estimator is as easy to compute as OLS. The penalty shrinks the coefficients towards 0 but does not set them equal to 0.
  - When $1 < q < 2$ the story is broadly similar: the estimators are easy to compute, but do not set coefficients equal to 0.
  - When $q < 1$ the penalty encourages many of the estimated coefficients to be equal to 0. In the extreme case, as $q \to 0$, this "norm"[1] is

$$\|\beta\|_0 = \lim_{q \to 0} \|\beta\|_q = \sum_{k=1}^{p} \mathbf{1}\{\beta_k \neq 0\},$$

    which just counts up the number of non-zero terms in $\beta$! In theory using a sparsity-promoting norm such as $\|\beta\|_q, q < 1$ might be a good idea. But the estimator is very difficult to compute (non-convex objective). In fact solving the problem when $q = 0$ is best subsets regression.
  - The borderline case $q = 1$ – i.e. the lasso – turns out to be the best of both worlds. It encourages sparsity while still being easy(ish) to compute.

# 5 Closed-form expressions with orthogonal X

- To shed more light on the differences between lasso, ridge and best subsets regression, consider the special case where $\mathbf{X}$ has orthonormal columns. In fact, let's consider the even more special case where

[1] Really a quasinorm, as it does not satisfy the triangle inequality.

$n = d$ and $\mathbf{X} = \mathbf{I}_d$. In this case the OLS coefficients $\hat{\beta} = Y$, and all three of lasso, ridge, and best subsets have a very simple expression in terms of the OLS coefficients $\hat{\beta} = Y$.

- For ridge the solution is
$$\frac{Y_j}{1+\lambda}.$$
So we see that ridge is taking each component of the vector $Y$ and smoothly shrinking it towards 0.

- For lasso the solution is
$$\begin{cases} Y_j - \lambda, & \text{if } Y_j > \lambda \\ Y_j + \lambda, & \text{if } Y_j < -\lambda \\ 0, & \text{otherwise.} \end{cases}$$
This operation is known as soft-thresholding. We see that it sets some coefficients equal to 0, and shrinks the rest towards 0.

- For best subsets with Mallow's $C_p$, the solution is
$$\begin{cases} Y_j, & \text{if } |Y_j| > \sqrt{\frac{2\sigma^2}{n}} \\ 0, & \text{otherwise.} \end{cases}$$
This operation is known as hard-thresholding. It performs selection but not shrinkage – some coefficients will be set equal to 0, but those that are not are left unchanged.

## Ridge and lasso on prostate cancer data

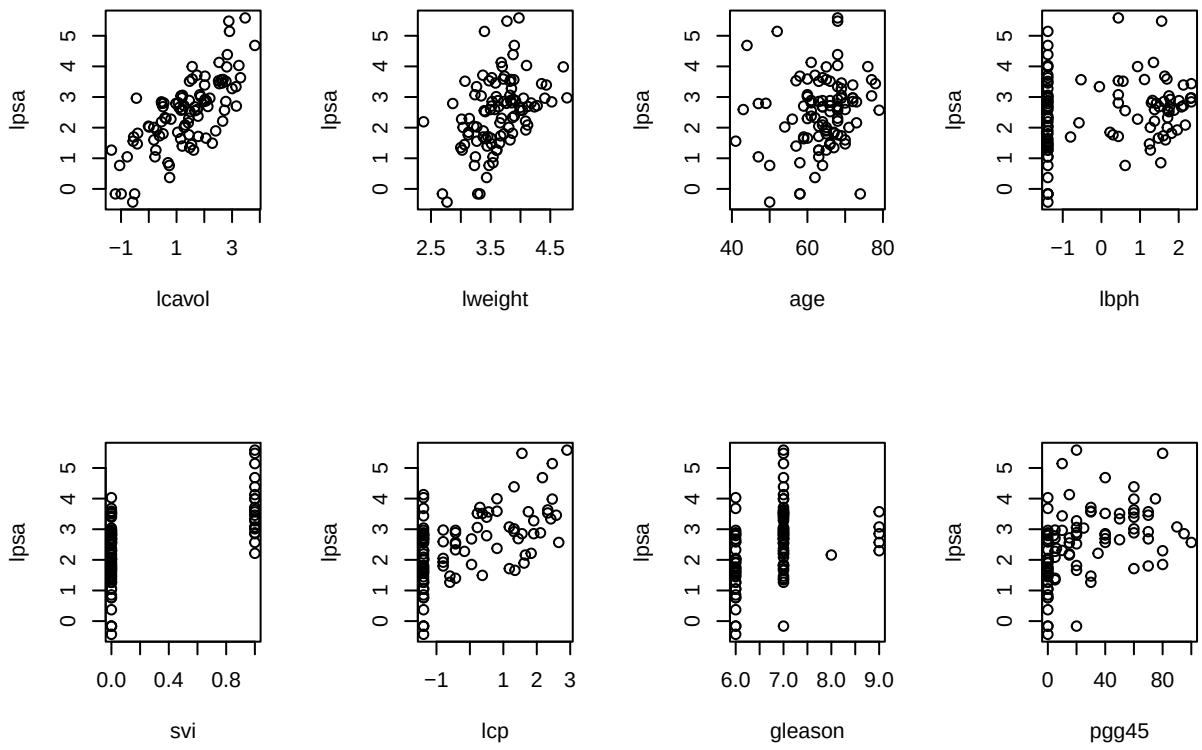
Taken from the *Elements of Statistical Learning* textbook, which is an excellent resource for ridge, lasso, and much else.

> The data for this example come from a study by Stamey et al. (1989). They examined the correlation between the level of prostate-specific antigen and a number of clinical measures in men who were about to receive a radical prostatectomy. The variables are log cancer volume (`lcavol`), log prostate weight (`lweight`), `age`, log of the amount of benign prostatic hyperplasia (`lbph`), seminal vesicle invasion (`svi`), log of capsular penetration (`lcp`), Gleason score (`gleason`), and percent of Gleason scores 4 or 5 (`pgg45`).

## EDA

```
prostate <- read.table("prostate.data", header = T)
attach(prostate) # Useful for the lazy, ?attach if interested
X <- cbind(lcavol, lweight, age, lbph, svi, lcp, gleason, pgg45)

par(mfrow = c(2,4))
plot(lcavol,lpsa)
plot(lweight,lpsa)
plot(age,lpsa)
plot(lbph,lpsa)
plot(svi,lpsa)
plot(lcp,lpsa)
plot(gleason,lpsa)
plot(pgg45,lpsa)
```




A few predictors are discrete. And we see one potential non-linearity. But nothing too dire, particularly since right now we are not focused on inference.

### OLS on prostate data

```
prostate.lm = lm(lpsa ~ X, prostate)
summary(prostate.lm)
```

```
##
## Call:
## lm(formula = lpsa ~ X, data = prostate)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.76644 -0.35510 -0.00328  0.38087  1.55770
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.181561   1.320568   0.137  0.89096
## Xlcavol      0.564341   0.087833   6.425 6.55e-09 ***
## Xlweight     0.622020   0.200897   3.096  0.00263 **
## Xage        -0.021248   0.011084  -1.917  0.05848 .
## Xlbph        0.096713   0.057913   1.670  0.09848 .
```

```
## Xsvi          0.761673   0.241176   3.158  0.00218 **
## Xlcp         -0.106051   0.089868  -1.180  0.24115
## Xgleason      0.049228   0.155341   0.317  0.75207
## Xpgg45        0.004458   0.004365   1.021  0.31000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6995 on 88 degrees of freedom
## Multiple R-squared:  0.6634, Adjusted R-squared:  0.6328
## F-statistic: 21.68 on 8 and 88 DF,  p-value: < 2.2e-16
```

This is pretty good by the standards of applied regression! Not an accident: dataset was chosen intentionally to make the effects of lasso + ridge as clear as possible.

### Implementing ridge

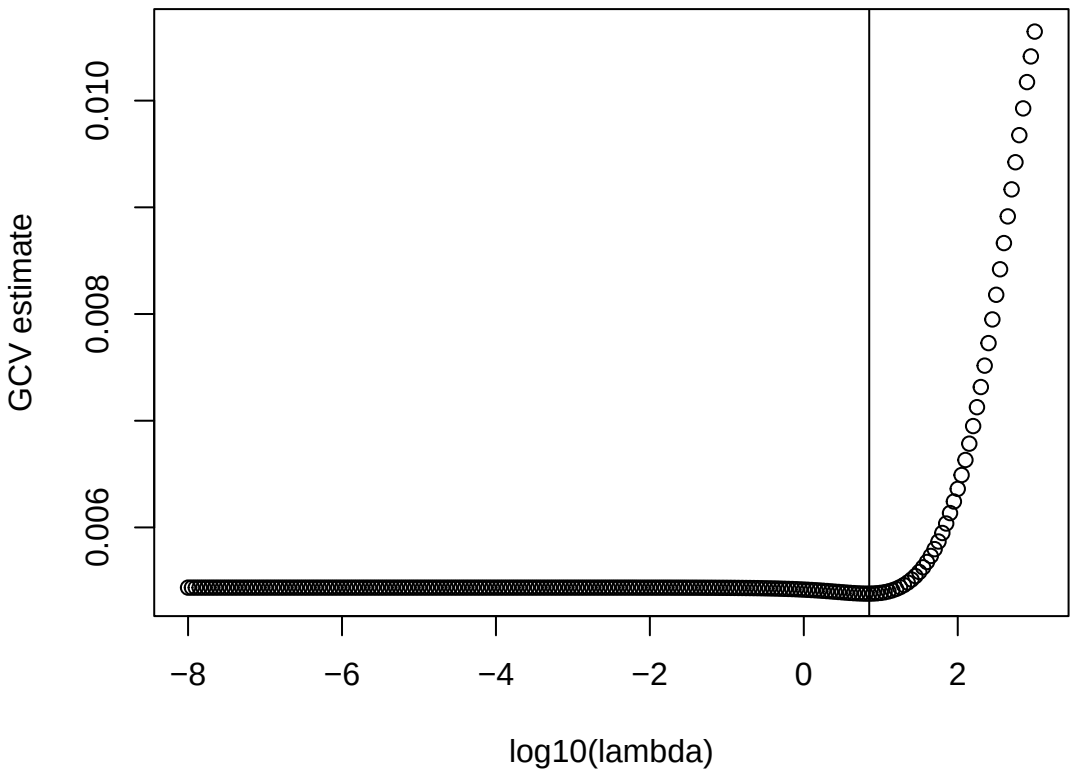
- For ridge it is usually a good idea to standardize the columns of $X$. For each column, subtract off the mean and normalize to have variance equal to 1. Otherwise the penalty term $\sum_{j=1}^{p} \beta_j^2$ favors predictors with larger variance.
- By default the intercept term in ridge is left unpenalized.
- **Solution path**: ridge regression has a separate solution $\hat{\beta}_\lambda$ for each $\lambda \in (0, \infty)$. Often it is useful to plot this.
- Typically we choose a single best value of $\lambda$ using cross-validation (CV).

### Ridge on cancer data

```
library(MASS)

# ridge at many different values of lambda
Xp = scale(X,TRUE,TRUE) # center and standardize columns of X
lambda = 10^(-seq(-3,8,.05)) # values of lambda at which to try ridge
ridge.lms = lm.ridge(lpsa ~ Xp, lambda = lambda)

# GCV
plot(log10(lambda),ridge.lms$GCV, ylab = "GCV estimate")
lambda_min = lambda[which.min(ridge.lms$GCV)]
abline(v = log10(lambda_min))
```

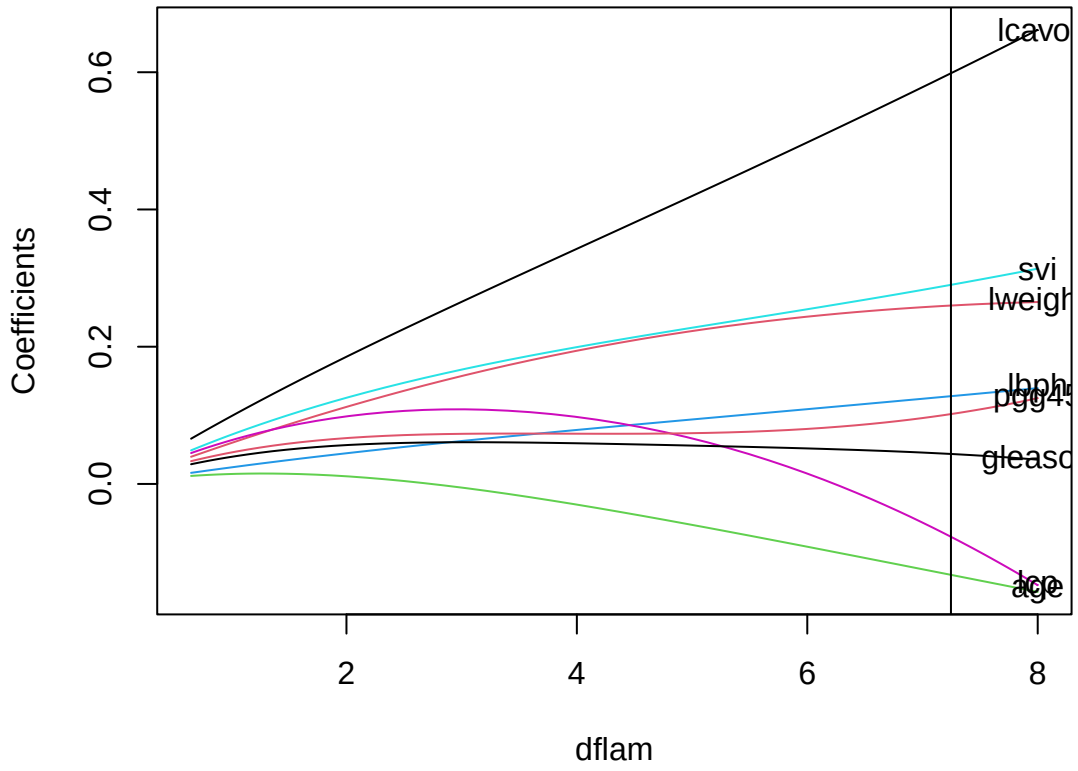



```
# Compare to OLS
prostate.lm = lm(lpsa ~ Xp)
coef.lm = coef(prostate.lm)
coef.ridge = coef(ridge.lms)[which.min(ridge.lms$GCV),]
coefs.df = data.frame(OLS = coef.lm,
                      ridge = coef.ridge)
coefs.df
```

```
##                     OLS       ridge
## (Intercept)  2.47838688  2.47838688
## Xplcavol     0.66514667  0.58054878
## Xplweight    0.26648026  0.25883389
## Xpage       -0.15819522 -0.12471352
## Xplbph       0.14031117  0.12463725
## Xpsvi        0.31532888  0.28404249
## Xplcp       -0.14828568 -0.05596450
## Xpgleason    0.03554917  0.04602347
## Xppgg45      0.12571982  0.09632095
```

### Shrinkage in the cancer data

```
# plot solution path of ridge against **effective df**
coefs <- ridge.lms$coef
G = t(Xp) %*% Xp
svs2 <- eigen(G)$values
smx <- svs2 %o% rep(1, length(lambda))
lmx <- rep(1,length(svs2)) %o% lambda
dflam <- colSums(smx/(smx + lmx))
matplot(dflam, t(coefs), type="l", lty=1, ylab = "Coefficients")
text( 8, coefs[,221], labels= colnames(Xp))
dfopt <- sum(svs2/(svs2 + 5.012))
abline(v= dfopt)
```




### Implementing lasso

- As with ridge important to standardize predictors.
- For lasso there is rarely a closed-form solution.
- For lasso there is no longer a LOOCV shortcut formula so we run $K$-fold CV. Typically with $K$ much smaller than $n$ – say $K = 10$ – to make things computationally reasonable.
- Luckily the package `glmnet` handles computing the lasso and does cv as well.

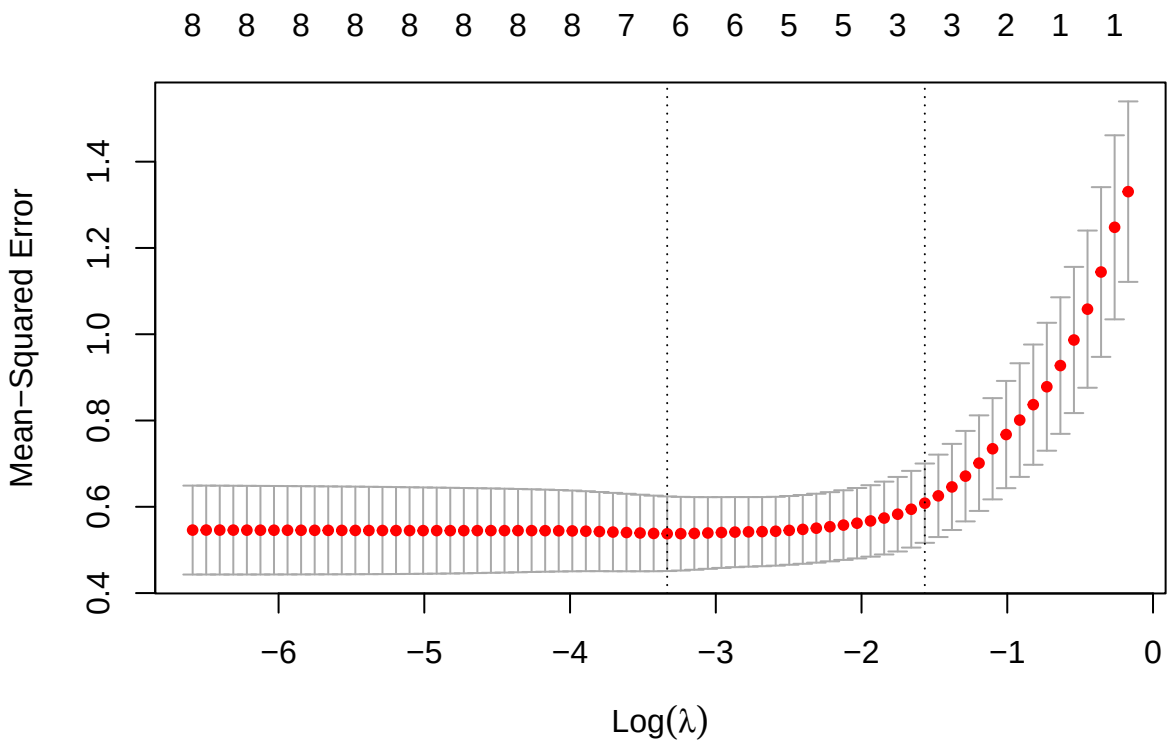
### Lasso on the cancer data

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.3.2
```

```
## Warning: package 'Matrix' was built under R version 4.3.2
```

```
# estimate of lasso risk using 10-fold CV
prostate.lassos <- cv.glmnet(Xp,lpsa)
plot(prostate.lassos)
```




```
# best lasso model, and smallest lasso model within 1 se of the minimum
best.lasso = coef(prostate.lassos, s = prostate.lassos$lambda.min)
onese.lasso = coef(prostate.lassos, s = prostate.lassos$lambda.1se)
coefs.df$lasso.min = as.numeric(best.lasso)
coefs.df$lasso.1se = as.numeric(onese.lasso)
coefs.df
```

```
##                      OLS       ridge     lasso.min lasso.1se
## (Intercept)  2.47838688  2.47838688  2.4783868784 2.4783869
## Xplcavol     0.66514667  0.58054878  0.5977369912 0.5286355
## Xplweight    0.26648026  0.25883389  0.2337383616 0.1201279
## Xpage       -0.15819522 -0.12471352 -0.0625876367 0.0000000
## Xplbph       0.14031117  0.12463725  0.0896842425 0.0000000
```

```
## Xpsvi         0.31532888  0.28404249  0.2442546100 0.1400748
## Xplcp        -0.14828568 -0.05596450  0.0000000000 0.0000000
## Xpgleason     0.03554917  0.04602347  0.0007028457 0.0000000
## Xppgg45       0.12571982  0.09632095  0.0652664723 0.0000000
```

### Algorithm for 10-fold CV

(a) split the data into $K = 10$ roughly equal parts

(b) for the $k$-th part, fit the model to the other $K - 1$ parts, then calculate the prediction error of the fitted model when predicting the $k$-th part of the data

(c) Do this for $k = 1, \ldots, K$ and average the estimates of prediction error. This yields a vector of prediction error, with one entry for each value of $\lambda$ tried.

(d) Choose the value of $\lambda$ that minimizes the prediction error, or alternatively (as here) the largest value of $\lambda$ that is within 1 standard error of the minimum.

### Sparsity and shrinkage with lasso on the cancer data

```
# Plot the solution path of lasso
l1opt <- sum(abs(coef(prostate.lassos, s = prostate.lassos$lambda.1se)[2:9]))
prostate.lasso <- glmnet(Xp[train,],lpsa[train])
plot(prostate.lasso, label=T) # x axis uses l1 norm as a proxy for df
abline(v=l1opt)
```