Regression III

Feature Selection and Regularization

Dave Armstrong

Goals for Today

1. Discuss feature selection and its relationship with more conventional model testing and discrimination.
2. Develop all subsets regressions and consider a comparison of models.
3. Describe cross-validation and its utility for helping choose tuning parameters.
4. Discuss regularization and regularized regression models - Ridge regression, the LASSO, Elastic Net and Adaptive LASSO.
    - Consider how these models adjudicate collinearity problems.
5. Consider the problem of post-selection inference.

Feature Selection

Sometimes, we may not want have a couple of different models; instead, we have a bunch of variables and we want to find out which ones are "important".

- Important, in this case, means predictive power - ability to capture variation or discriminate among values in the dependent variable.
- Feature selection automate the process of choosing features based on what "works" in the data.

Some questions you might have:

- Q: Isn't this atheoretical? A: Yes
- Q: Isn't this data mining? A: Yes
- Q: Isn't this kind of analysis disingenuous? A: It depends.

Notes

Type notes here...

Feature Selection: Manually

How many of you have written a paper where you had a theory, that theory produced a single model specification, the operationalization of the concepts in measures was utterly uncontroversial and the empirical model was so thoroughly beyond reproach and obviously useful that no diagnostics were needed?

- We generally use **ad hoc** methods of feature selection.
- These are only slightly less problematic - more on volume than principle.
- If we're going to use the data to select features, why not go all the way?

Notes

Type notes here...

Subset Methods

- The goal of subset methods is to examine *which subsets give the best fit to the data for a given number of predictors*
- Even when the number of variables is large, it is feasible to examine all subsets
  - If there are p potential predictors, then there are 2^p possible models
- Subset techniques have the advantage over stepwise regression of revealing alternative *nearly equivalent* models and thus avoid the appearance of a uniquely "correct" result
- Several measures can be used to determine the best model subset
  - R^2
  - AIC
  - BIC
  - Mallow's C_p-statistic

Notes

Type notes here...

Mallow's Cp Statistic

Mallow's C_p-statistic is defined as:

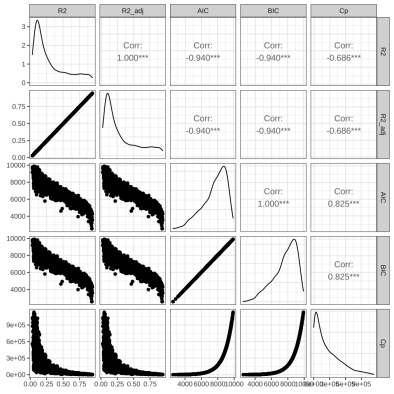
$$\begin{aligned} C_p &= \frac{\sum E_i^2}{S_E^2} + 2p - n \\ &= (K + 1 - p)(F_p - 1) + p \end{aligned}$$

- S_E^2 is for the full model containing k explanatory variables; RSS $(\sum E_i^2)$ is from the subset model with p explanatory variables
- F_p is the incremental F-test for the hypothesis that the regressors omitted from the subset have slope 0. If the hypothesis is true, $E(F_p) \simeq 1$, and thus $C_p \simeq p$
- C_p increases with the residual sum of squares.
- A good model, then, has C_p as close to p as possible
- A plot of C_p against p allows us to choose the model

Notes

Type notes here...

Comparison



Notes

Type notes here...

Model Selection Example: Ericksen Data (1)

```
library(leaps)
library(car)
Ericksen <- DAMisc::scaleDataFrame(Ericksen)
X <- model.matrix(undercount ~ .,
  data=Ericksen)[,-1]
y <- model.response(model.frame(undercount
   ~ ., data=Ericksen))
rmods <- regsubsets(x=X, y=y, method="exhaustive",
    all.best=TRUE, nbest=10)
```


Notes

Type notes here...

Model Selection Example: Ericksen Data (3)

- Subset selection is implement in R using two packages: `leaps` and `car`
- Using the `regsubsets` function, you specify the full model and how many subsets you want
- The `subsets` function in `car` graphs the models with the subset size on the horizontal axis and the statistic used for fit on the vertical axis
- The `subsets` function allows you to specify the following statistics: Mallows C_p `cp`, R^2 `rsq`, adjusted R^2 `adjrs2`, RSS `rss` or BIC `bic`
- You can also specify the number of predictors you want in the model (below specifies 3 to 5 predictors)

Notes

Type notes here...

Subsets plot for the Ericksen Data

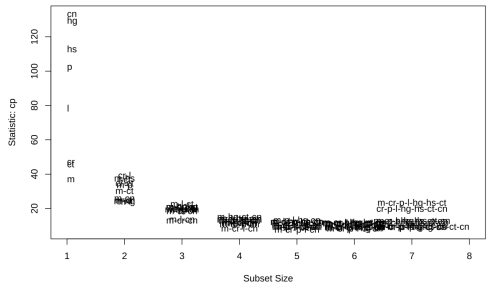
```
library(car)
subsets(rmods, statistic="cp", legend=F)
```


Notes

Type notes here...

Subsets plot for the Ericksen Data (2)

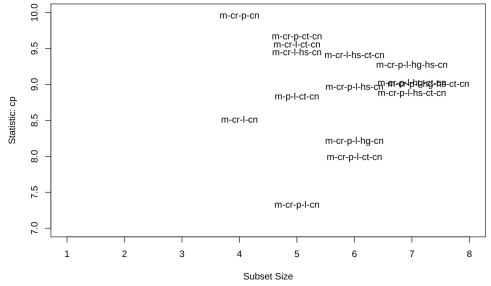
```
library(car)
subsets(rmods, statistic="cp", legend=F)
```



Notes

Type notes here...

Subsets plot zoomed in



Notes

Type notes here...

$C_p - K$

We could also consider the models that have the smallest $C_p - K$.

```
s <- summary(rmods)
K <- rowSums(s$which)
abbrevs <- c("m", "cr", "p", "l", "hg", "hs", "ct", "cn")
mod <- apply(s$which[,-1], 1, function(i)
  paste(abbrevs[which(i)], collapse="-"))
dat <- tibble(
  K=K,
  Cp = s$cp,
  adjr2 = s$adjr2,
  mod = mod,
  diff = Cp-K)
sub <- dat %>%
  filter(K < 9) %>%
  slice_min(diff, n=10)
```

```
## # A tibble: 10 x 5
##        K    Cp adjr2 mod                 diff
##    <dbl> <dbl> <dbl> <chr>              <dbl>
##  1     8  8.87 0.662 m-cr-p-l-hs-ct-cn  0.874
##  2     7  7.98 0.661 m-cr-p-l-ct-cn     0.983
##  3     8  9.01 0.661 m-cr-p-l-hg-ct-cn  1.01
##  4     7  8.21 0.660 m-cr-p-l-hg-cn     1.21
##  5     8  9.27 0.659 m-cr-p-l-hg-hs-cn  1.27
##  6     6  7.32 0.659 m-cr-p-l-cn        1.32
##  7     7  8.96 0.656 m-cr-p-l-hs-cn     1.96
##  8     7  9.41 0.653 m-cr-l-hs-ct-cn    2.41
##  9     6  8.83 0.651 m-p-l-ct-cn        2.83
## 10     8 11.4  0.647 m-cr-l-hg-hs-ct-cn 3.41
```


Notes

Type notes here...

Overfit Much?

How do we know we're not overfitting our data?

- Sometimes it's obvious - it's hard to argue that you're overfitting when your R^2 is 0.03.
- Generally, we don't know.

Cross-validation is a way of trying to protect us against overfitting the model.

Notes

Type notes here...

Cross-Validation (1)

If no two observations have the same Y, a p-variable model fit to $p + 1$ observations will fit the data precisely

- Of course, this will lead to biased estimators that are likely to give quite different predictions on another dataset (generated with the same DGP)
- Model validation allows us to assess whether the model is likely to predict accurately on future observations or observations not used to develop this model
  - External validation involves retesting the model on new data collected at a different point in time or from a different population
  - Internal validation (or cross-validation) involves fitting and evaluating the model carefully using only one sample

Notes

Type notes here...

Cross-Validation (2)

Cross-validation is similar to bootstrapping in that it resamples from the original data

- The basic form involves randomly dividing the sample into two subsets:
  - The first subset of the data (screening sample) is used to select or estimate a statistical model
  - The second subset is then used to test the findings
- Can be helpful in avoiding capitalizing on chance and over-fitting the data - i.e., findings from the first subset may not always be confirmed by the second subsets
- Cross-validation is often extended to use several subsets (either a preset number chosen by the researcher or leave-one-out cross-validation)

Notes

Type notes here...

Cross-Validation (3)

- The data are split into k subsets (usually $3 \leq k \leq 10$)
- Each of the subsets are left out in turn, with the regression run on the remaining data
- Prediction error is then calculated as the sum of the squared errors:

$$RSS = \sum (Y_i - \hat{Y}_i)^2$$

- We choose the model with the smallest average "error"

$$MSE = \frac{\sum (Y_i - \hat{Y}_i)^2}{n}$$

- We could also look to the model with the largest average R^2

Notes

Type notes here...

Cross-Validation (4)

How many observations should I leave out from each fit?

- There is no rule on how many cases to leave out, but Efron (1983) suggests that grouped cross-validation (with approximately 10% of the data left out each time) is better than leave-one-out cross-validation

Number of repetitions

- Harrell (2001:93) suggests that one may need to leave $\frac{1}{10}$ of the sample out 200 times to get accurate estimates

Cross-validation does not validate the complete sample

- External validation, on the other hand, validates the model on a new sample
- Of course, limitations in resources usually prohibits external validation in a single study

Notes

Type notes here...

Cross-Validation in R

```
library(boot)
dat <- read.csv("http://www.quantoid.net/files/reg3/weakliem.txt"
  header=T)
dat <- dat[-c(25,49), ]
mod1 <- glm(secpay ~ poly(gini, 3)*democrat, data=dat)
mod2 <- glm(secpay ~ gini*democrat, data=dat)

deltas <- NULL
for(i in 1:25){
deltas <- rbind(deltas, c(
  cv.glm(dat, mod1, K=5)$delta,
  cv.glm(dat, mod2, K=5)$delta)
)}
out <- matrix(colMeans(deltas), ncol=2)
rownames(out) <- c("delta_1", "delta_2")
colnames(out) <- c("Model 1", "Model 2")
```

```
##         Model 1 Model 2
## delta_1  0.0212  0.0058
## delta_2  0.0180  0.0057
```

The `delta_1` term is the average raw cross-validation error. The `delta_2` term corrects for using k-fold rather than leave-one-out CV.

Notes

Type notes here...

Tidy CV

```
cv_fun <- function(split, ...){
  u1 <- update(mod1, data=analysis(split))
  u2 <- update(mod2, data=analysis(split))
  e1_sq <- (assessment(split)$secpay -
              predict(u1, newdata=assessment(split)))^2
  e2_sq <- (assessment(split)$secpay -
              predict(u2, newdata=assessment(split)))^2
  tibble(
    err = c(sum(e1_sq), sum(e2_sq)),
    n = c(length(e1_sq), length(e2_sq)),
    model = factor(c("poly_int", "linear_int")))
}
library(purrr)
library(rsample)
v <- vfold_cv(dat, v=5, repeats=250) %>%
  mutate(err = map(splits, cv_fun))
```

```
v %>% unnest(err) %>%
  group_by(id, model) %>%
  summarise(mse = sum(err)/sum(n)) %>%
  pivot_wider(names_from="model", values_from = "mse") %>%
  mutate(diff = poly_int - linear_int) %>%
  ungroup %>%
  summarise(across(c(linear_int, poly_int), mean),
            "p(MSE_M1 > MSE_M2)" = mean(diff > 0))
```

```
## # A tibble: 1 x 3
##   linear_int poly_int `p(MSE_M1 > MSE_M2)`
##        <dbl>    <dbl>                <dbl>
## 1    0.00578   0.0103                   1
```


Notes

Type notes here...

Cross-validating Span in Loess

We could use cross-validation to tell us something about the span in our Loess model.

- First, split the sample into K groups (usually 10).
- For each of the $k = 10$ groups, estimate the model on the other 9 and get predictions for the omitted groups observations. Do this for each of the 10 subsets in turn.
- Calculate the CV error: $\frac{1}{n}\sum(y_i - \hat{y}_i)^2$
- Potentially, do this lots of times and average across the CV error.

```
set.seed(1)
n <- 400
x <- 0:(n-1)/(n-1)
f <- 0.2*x^11*(10*(1-x))^6+10*(10*x)^3*(1-x)^10
y <- f + rnorm(n, 0, sd = 2)
tmp <- data.frame(y=y, x=x)
lo.mod <- loess(y ~ x, data=tmp, span=.75)
```


Notes

Type notes here...

Minimizing CV Criterion Directly

```
library(DAMisc)
cvlo <- DAMisc:::cv.lo2
best.span <- optimize(cvlo, c(.05,.95), form=y ~ x, data=tmp,
         numiter=5, K=10)
best.span
```

```
## $minimum
## [1] 0.2611968
##
## $objective
## [1] 3.914841
```

There is also a canned function in `fANCOVA` that optimizes the span via AICc or GCV.

```
library(fANCOVA)
best.span2 <- loess.as(tmp$x, tmp$y, criterion="aicc")
best.span2$pars$span
```

```
## [1] 0.2136455
```

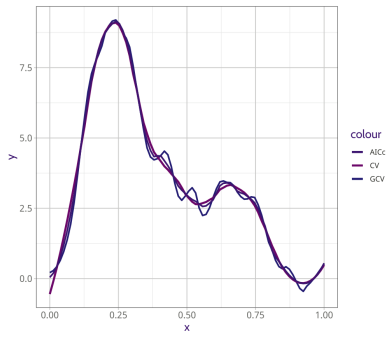
```
best.span3 <- loess.as(tmp$x, tmp$y, criterion="gcv")
best.span3$pars$span
```

```
## [1] 0.1483344
```


Notes

Type notes here...

The Curve



Notes

Type notes here...

Manually Cross-Validating λ in the Y-J Transform

Sometimes, optimizing the cross-validation criterion fails.

- Randomness in the CV procedure can produce a function that has several local minima.
- You could force the same random split at every evaluation by hand-coding the CV, but this might not be the best idea.
- If optimization of the CV criterion fails, you could always do it manually.

Notes

Type notes here...

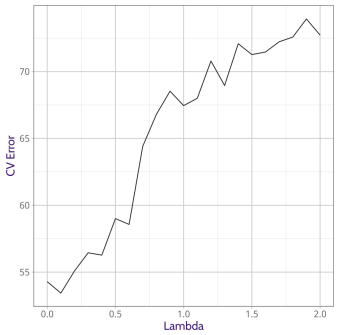
Example

```
cvoptim_yj <- function(pars, form, data, trans.vars, K=5, numiter=10){
    require(boot)
      require(VGAM)
    form <- as.character(form)
    for(i in 1:length(trans.vars)){
        form <- gsub(trans.vars[i], paste("yeo.johnson(", trans.vars[i],
        ",", pars[i], ")", sep=""), form)
    }
    form <- as.formula(paste0(form[2], form[1], form[3]))
    m <- glm(as.formula(form), data, family=gaussian)
    d <- lapply(1:numiter, function(x)cv.glm(data, m, K=K))
    mean(sapply(d, function(x)x$delta[1]))
}

lams <- seq(0,2, by=.1)
s <- sapply(lams, function(x)cvoptim_yj(x, form=prestige ~ income + education + women,
    data=Prestige, trans.vars="income", K=3))
ggplot() +
  geom_line(mapping=aes(y=s, x=lams)) +
  theme_bw() +
  mytheme() +
  labs(x="Lambda", y="CV Error")
```


Notes

Type notes here...

Figure



Notes

Type notes here...

Shrinkage Estimators

Shrinkage estimators can reduce sampling variability and sometimes improve model fit (particularly in the presence of collinearity).

- Shrinkage estimators impose constraints on the fitted model (particularly on the size of the coefficients).
- The result of these constraints is to shrink the estimates toward zero.
- Ridge Regression and the LASSO are the two most prominent shrinkage estimators.

NB: these are *biased* estimators, so they might be good for stabilizing predictions, but they won't be particularly good for more conventional theory testing.

Notes

Type notes here...

Ridge Regression

Ridge Regression minimizes the following function:

$$\sum_{i=1}^{N}\left(y_i - \beta_0 + \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2$$

- λ is a tuning parameter that governs the relative of RSS and the penalty on fitting the regression surface.
- As $\lambda \to 0$, the estimates get increasingly close to the OLS estimates.
- As $\lambda \to \infty$, the estimates get increasingly close to zero.

The choice of λ is important and is often done with cross-validation.

Notes

Type notes here...

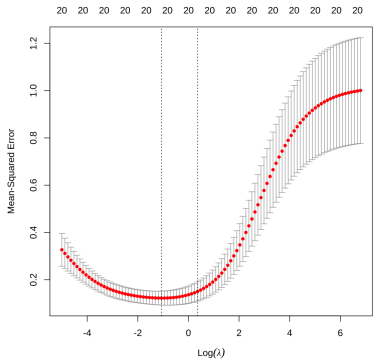
CV MSE

```
library(glmnet)
library(ggplot2)
library(rio)
library(tidyr)
banks99 <- import(
  "http://quantoid.net/files/reg3/banks99.dta")
banks99s <- scaleDataFrame(banks99[,-c(1,2,4)])
X <- scale(model.matrix(gdppc_mp ~. , data=banks99s))[,-1]
y <- model.response(model.frame(gdppc_mp ~. , data=banks99s))

library(glmnet)
loglam <- seq(6.8, -5, length=100)
g1 <- glmnet(X, y, alpha=0)

rcv <- cv.glmnet(X, y, alpha=0, lambda=exp(loglam))
plot(rcv)
```


Notes

Type notes here...

CV MSE (2)



Notes

Type notes here...

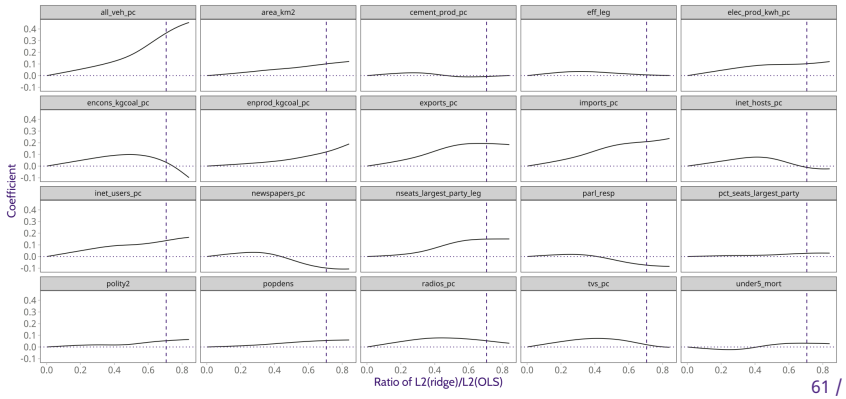
CV with Ridge Regression

```
r <- glmnet(X, y, alpha=0, lambda=exp(loglam))
ridge.mod <- glmnet(X,y, alpha=0, lambda=rcv$lambda.min)
mod <- lm(y ~ X)

l2o <- sqrt(sum(coef(mod)^2))
l2r <- apply(r$beta, 2, function(x)sqrt(sum(x^2)))
br <- r$beta %>% as.matrix %>% t %>% as.data.frame
br$ratio <- l2r/l2o
br <- br %>% pivot_longer(under5_mort:all_veh_pc, names_to="variable", values_to="coef")
ggplot(br, aes(x=ratio, y=coef)) +
  geom_line() +
  geom_vline(xintercept=(l2r/l2o)[87], lty=2) +
  geom_hline(yintercept=0, linetype=3) +
  facet_wrap(~variable) +
  theme_bw() +
  mytheme(panel.grid=element_blank()) +
  labs(x="Ratio of L2(ridge)/L2(OLS)", y="Coefficient")
```


Notes

Type notes here...

Plot



Notes

Type notes here...

Collinearity

```
set.seed(1234)
Sig <- diag(5)
Sig[3:5,3:5] <- .99
diag(Sig) <- 1
X <- MASS::mvrnorm(500,rep(0,5), Sig)
b <- c(1,1,1,0,0)
ystar <- X %*% b
y <- ystar + rnorm(500, 0, 2)
```

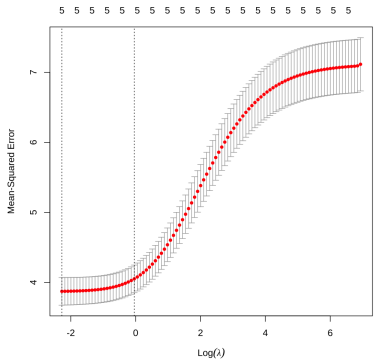
```
summary(m1 <- lm(y~ X))
```

```
##
## Call:
## lm(formula = y ~ X)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4195 -1.3696 -0.0068  1.4012  5.3665
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.01686    0.08762   0.192   0.8475
## X1           1.17283    0.09359  12.532   <2e-16 ***
## X2           1.11657    0.09163  12.185   <2e-16 ***
## X3           1.21441    0.69342   1.751   0.0805 .
## X4           1.04118    0.68252   1.525   0.1278
## X5          -1.09301    0.70047  -1.560   0.1193
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.953 on 494 degrees of freedom
## Multiple R-squared:  0.469,    Adjusted R-squared:  0.4637
## F-statistic: 87.28 on 5 and 494 DF,  p-value: < 2.2e-16
```


Notes

Type notes here...

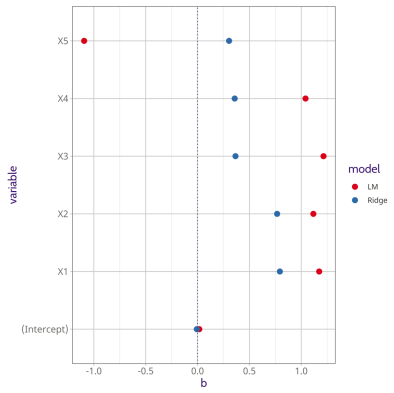
Collinearity (2)



Notes

Type notes here...

Collinearity (3)

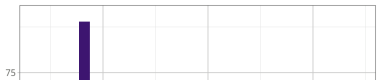


Notes

Type notes here...

Prediction Variances

```
library(boot)
df <- cbind(y, X)
boot.ridge <- function(data, inds, ...){
  tmp <- data[inds,]
  y <- tmp[,1]
  X <- tmp[,-1]
  out <- glmnet(X,y,alpha=0, lambda=.8736)
  as.vector(coef(out))
}
br <- boot(statistic=boot.ridge,
           data=df, R=100)
v <- var(br$t)
pred.vars <- diag(cbind(1, X) %*%
                    v %*% t(cbind(1,X)))
lm.vars <- diag(cbind(1, X) %*%
                  vcov(m1) %*% t(cbind(1,X)))
```



Notes

Type notes here...

Predictions

```
summary(pred.vars/lm.vars)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1473  0.4181  0.5551  0.5619  0.6800  1.2081
```

```
ridge.preds <- cbind(1, X) %*% coef(r2)
lm.preds <- cbind(1,X) %*% coef(m1)
cor(as.vector(lm.preds), as.vector(ridge.preds))
```

```
## [1] 0.9847665
```


Notes

Type notes here...

LASSO (the L1 norm)

The LASSO (Least Absolute Shrinkage and Selection Operator) is another regularization method for estimating regression.

- Uses a different penalty than ridge regression:

$$\sum_{i=1}^{N}\left(y_i - \beta_0 + \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j|$$

- Doesn't necessarily use all of the variables (i.e., some coefficients could be zero)
- Since not all variables are used in each fit, bootstrapping is more problematic here (though not impossible).

Notes

Type notes here...

The LASSO in R

```
banks99 <- import(
  "http://quantoid.net/files/reg3/banks99.dta")
banks99s <- scaleDataFrame(banks99[,-c(1,2,4)])
X <- scale(model.matrix(gdppc_mp ~. , data=banks99s))[,-1]
y <- model.response(model.frame(gdppc_mp ~. , data=banks99s))

loglam <- seq(6.8, -5, length=100)
cvg <- cv.glmnet(X,y, lambda=exp(loglam))
g <- glmnet(X, y, lambda=cvg$lambda.min)
```

```
round(cbind(coef(cvg), coef(mod)), 4)
```

```
## 21 x 2 sparse Matrix of class "dgCMatrix"
##                                s1
## (Intercept)              0.0000  0.0000
## under5_mort              .       0.0214
## area_km2                 .       0.1365
## inet_hosts_pc            .      -0.0032
## inet_users_pc            0.1022  0.1813
## enprod_kgcoal_pc         .       0.2801
## encons_kgcoal_pc         0.0125 -0.2730
## elec_prod_kwh_pc         .       0.1422
## cement_prod_pc           .       0.0073
## nseats_largest_party_leg .       0.1520
## eff_leg                  .      -0.0026
## pct_seats_largest_party  .       0.0250
## radios_pc                .       0.0140
## tvs_pc                   .      -0.0025
## newspapers_pc            .      -0.0930
## polity2                  .       0.0765
## parl_resp                .      -0.0853
## popdens                  .       0.0607
## imports_pc               0.1844  0.2825
## exports_pc               .       0.1673
## all_veh_pc               0.4832  0.5060
```


Notes

Type notes here...

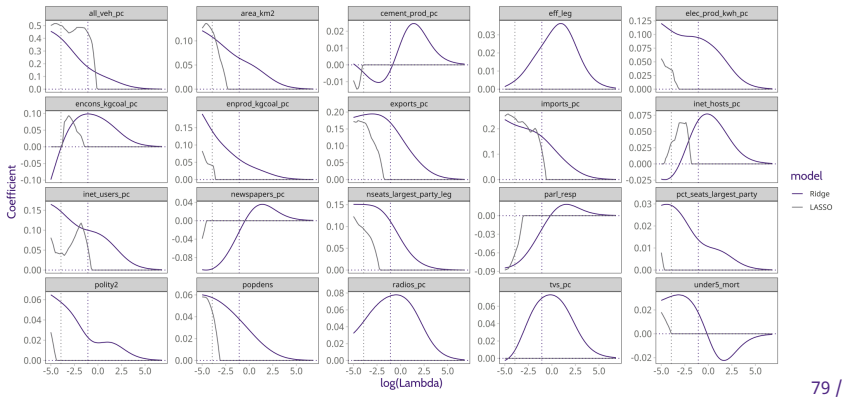
Regularization path

```
r <- glmnet(X, y, alpha=0, lambda=exp(loglam))
g <- glmnet(X, y, alpha=1, lambda=exp(loglam))
mod <- lm(y ~ X)


br1 <- r$beta %>% as.matrix %>% t %>% as.data.frame
br2 <- g$beta %>% as.matrix %>% t %>% as.data.frame
br1$lambda <- br2$lambda <- loglam
br1 <- br1 %>% pivot_longer(under5_mort:all_veh_pc, names_to="variable", values_to="coef")
br2 <- br2 %>% pivot_longer(under5_mort:all_veh_pc, names_to="variable", values_to="coef")
br1$model <- factor(1, levels=c(1,2), labels=c("Ridge", "LASSO"))
br2$model <- factor(2, levels=c(1,2), labels=c("Ridge", "LASSO"))

br <- bind_rows(br1, br2)

ggplot(br, aes(x=lambda, y=coef, colour=model)) +
  geom_line() +
  facet_wrap(~variable, scales="free_y") +
  geom_vline(xintercept=log(rcv$lambda.min), col=pal2[1], lty=3) +
  geom_vline(xintercept=log(cvg$lambda.min), col=pal2[2], lty=3) +
  geom_hline(yintercept=0, linetype=3) +
  scale_colour_manual(values=pal2) +
  theme_bw() +
  mytheme(panel.grid=element_blank()) +
  labs(x="log(Lambda)", y="Coefficient")
```


Notes

Type notes here...

Plot



Notes

Type notes here...

Predictions

```
r1 <- glmnet(X, y, alpha=0, lambda=rcv$lambda.mi
g1 <- glmnet(X, y, alpha=1, lambda=cvg$lambda.mi
yhat <- mod$fitted.values
names(yhat) <- NULL

preds <- tibble(
  ols=yhat,
  ridge = as.vector(predict(r1, newx=X)),
  lasso= as.vector(predict(g1, newx=X))
)
```

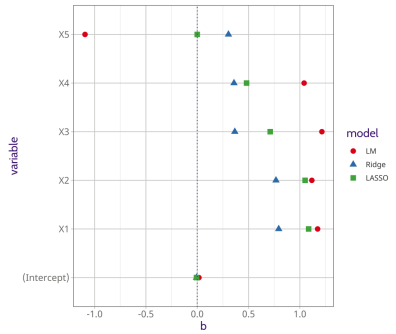
```
ggpairs(preds) + mytheme() + theme_bw()
```

Notes

Type notes here...

LASSO and collinearity

```
cvg2 <- cv.glmnet(scale(coll$X), coll$y, alpha=1
g2 <- glmnet(scale(coll$X), coll$y,
             alpha=1, lambda=cvg2$lambda.min)
r2 <- glmnet(scale(coll$X), coll$y,
             alpha=0, lambda=rcv2$lambda.1se)
coefs <- tibble(
  b = c(as.vector(m1$coef), as.vector(coef(r2)),
        as.vector(coef(g2))),
  model = factor(rep(1:3, each=length(coef(m1))),
           labels=c("LM", "Ridge", "LASSO")),
  variable = rep(names(m1$coef), 3))
p1 <- ggplot(coefs, aes(x=b, y=variable,
                        colour=model, shape=model)) +
  geom_point(size=3) +
  theme_bw() +
  scale_colour_manual(values=pal5[1:3]) +
  geom_vline(xintercept=0, lty=3)+
  mytheme()
```



Notes

Type notes here...

Elastic Net

The *Elastic Net* is a compromise between Ridge and LASSO regression:

$$\min_{\beta_0,\beta} \frac{1}{N} \sum_{i=1}^{N} w_i l(y_i, \beta_0 + \beta^T x_i) + \lambda \left[(1-\alpha)||\beta||_2^2/2 + \alpha||\beta||_1 \right],$$

- LASSO: $\alpha = 1$, Ridge: $\alpha = 0$
- α can be chosen a priori or you can experiment with several different values.
- Often setting α close to, but not exactly, 1 has nice properties.

Notes

Type notes here...

Elastic Net in Action

```
cv.enet <- list()
s <- seq(0.01, .99, length=25)
for(i in 1:length(s)){
    cv.enet[[i]] <- cv.glmnet(X, y, alpha = s[i])
}
cv.err <- sapply(cv.enet, function(x)min(x$cvm))
s[which.min(cv.err)]
```

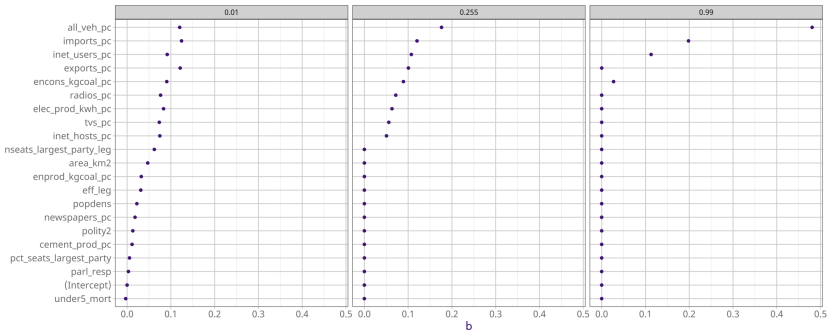
```
## [1] 0.01
```

```
b <- sapply(cv.enet[c(1,7,25)],
    function(x)as.matrix(coef(x)))

plot.dat <- data.frame(
    b = c(b),
    group = as.factor(rep(c(.010,.255,.990),
        each = 21)),
    var = factor(rep(
        rownames(coef(cv.enet[[1]])), 3)))
library(ggplot2)
g <- ggplot(plot.dat, aes(x=b,
    y=reorder(var, b, mean))) +
    geom_point() +
    scale_colour_manual(values=pal3) +
    theme_bw() +
    facet_wrap(~group, nrow=1) +
    mytheme() +
    ylab("")
```


Notes

Type notes here...

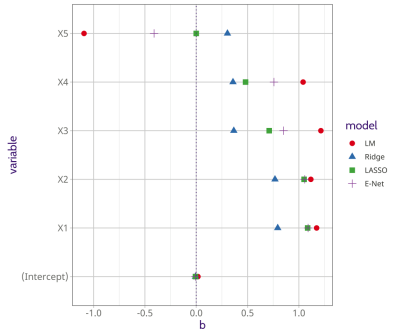
Figure



Notes

Type notes here...

Elastic Net and Collinearity



Notes

Type notes here...

Adaptive Lasso

The lasso gives all variables the same penalty (λ). The adaptive lasso relaxes this assumption by allowing each parameter to have a different weight:

$$\underset{\beta}{\operatorname{argmin}} \left\| y - \sum_{j-1}^{p} \mathbf{x}_j \beta_j \right\|^2 + \lambda \sum_{j=1}^{p} w_j |\beta_j|$$

Where we use results from an auxiliary regression (OLS, Ridge or LASSO) to make the weights:

$$\hat{w}_j = \frac{1}{|\hat{\beta}_j|^\gamma}$$

γ is not usually estimated, but values 0.5, 1, and 2 are tried to evaluate sensitivity. The only technical constraint is that $\gamma > 0$. \

Notes

Type notes here...

Oracle Property

The Adaptive Lasso has been shown to have the Oracle property, that the selection procedure asymptotically chooses the right model:

- True 0 coefficients are estimated as 0 with probability that tends toward 1
- True non-zero coefficients are estimated as if the true sub-model were known.

Notes

Type notes here...

Steps for Adaptive LASSO

- Estimate the initial coefficients via regression model (OLS, Ridge or LASSO).
- Calculate the weights $w_j = \frac{1}{|\beta_j|^\gamma} \quad \gamma = \{0.5, 1, 2\}$.
- Use the weights as input to the LASSO routine.

Notes

Type notes here...

Adaptive LASSO

```
# estimate initial ridge regression and save coefficients
b.ridge <- coef(cv.glmnet(X,y, alpha=0))
# calculate weights
gamma <- 1
w.banks <- 1/(abs(b.ridge)^gamma)
# estimate the LASSO with the weights
cval <- cv.glmnet(X,y, penalty.factor=w.banks[-1])
```

```
coef(cval)
```

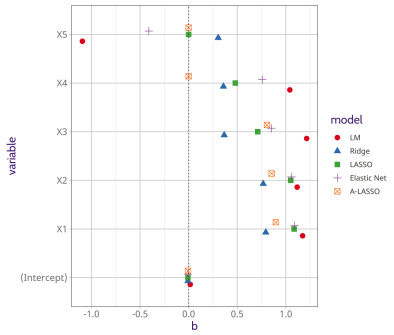
```
## 21 x 1 sparse Matrix of class "dgCMatrix"
##                                      s1
## (Intercept)              -1.059584e-17
## under5_mort               .
## area_km2                  .
## inet_hosts_pc             .
## inet_users_pc             .
## enprod_kgcoal_pc          .
## encons_kgcoal_pc          .
## elec_prod_kwh_pc          .
## cement_prod_pc            .
## nseats_largest_party_leg  .
## eff_leg                   .
## pct_seats_largest_party   .
## radios_pc                 .
## tvs_pc                    .
## newspapers_pc             .
## polity2                   .
## parl_resp                 .
## popdens                   .
## imports_pc                2.229577e-01
## exports_pc                .
## all_veh_pc                5.418146e-01
```


Notes

Type notes here...

Adaptive LASSO and Collinearity

```
b.ridge <- coef(cv.glmnet(scale(coll$X),
                          coll$y, alpha=0))
# calculate weights
gamma <- 1
w <- 1/(abs(b.ridge)^gamma)
# estimate the LASSO with the weights
cval <- cv.glmnet(scale(coll$X),coll$y,
                  penalty.factor=w[-1])
```



Notes

Type notes here...

All predictions

```
r1 <- glmnet(X, y, alpha=0, lambda=rcv$lambda.mi
g1 <- glmnet(X, y, alpha=1, lambda=cvg$lambda.mi
g3 <- glmnet(X, y, alpha=s[which.min(cv.err)],
             lambda=cvg$lambda.min)
g4 <- glmnet(X, y, alpha=1, penalty.factor=w.ban
             lambda=cval$lambda.min)
yhat <- mod$fitted.values
names(yhat) <- NULL

preds <- tibble(
  ols=yhat,
  ridge = as.vector(predict(r1, newx=X)),
  lasso= as.vector(predict(g1, newx=X)),
  enet = as.vector(predict(g3, newx=X)),
  alasso = as.vector(predict(g4, newx=X)))
```

```
ggpairs(preds) + mytheme() + theme_bw()
```

Notes

Type notes here...

PGI Analysis

Table 2. Elastic net coefficients.

	Federal	Provincial	Regional
Intercept	0.000 [0.000, 0.000]	0.000 [−0.000, 0.000]	0.000 [−0.000, −0.000]
Eligible Voters (log)	−0.749 [−1.831, −0.126]	−0.002 [−0.027, −0.000]	−0.015 [−0.047, 0.000]
Average Age	0.000 [0.000, 0.000]	0.002 [0.000, 0.037]	0.000 [0.000, 0.000]
Median Income (log)	0.445 [0.000, 1.058]	−0.001 [−0.013, −0.000]	−0.510 [−0.566, −0.417]
% Unemployed	0.257 [0.000, 0.516]	0.002 [0.000, 0.037]	0.000 [0.000, 0.000]
% No Degree	0.001 [0.000, 0.297]	−0.002 [−0.035, −0.000]	0.000 [0.000, 0.000]
% University Degree	0.082 [−0.113, 0.183]	0.000 [−0.013, 0.000]	0.000 [0.000, 0.000]

Main entries are average coefficients across the cross-validation runs.
Entries in brackets show the range of coefficients across all 1000 cross-validation runs and are *not* confidence intervals.
In the results above, all variables were standardized to have mean 0 and unit variance.

Notes

Type notes here...

Inference After Selection

Inference gets much more complicated after model selection, given that variables are often selected *because* they are significant predictors. There are a few options for post-selection inference.

- Data Splitting - Split the sample into two halves - select on one set, test on the other. Most conservative (loss of power due to lower N).
- Data Carving - A small proportion of the sample is withheld from training and then the entire sample is used for testing (Fithian 2014).
- Exact post-selection inference possible for Forward Selection Regression and LASSO with fixed λ - `{SelectiveInferecen}` package in R (Tibshirani et al 2014).
- Valid post-selection inference for Linear LS Models - implemented in the `{PoSI}` package in R (Berk et al 2013)

Notes

Type notes here...

Variable Selection Methods: Cautions (1)

- If we have a very large number of predictors and we simply want a parsimonious predictive model, subset methods and the lasso could be really useful.
- When tackling collinearity, however, variable selection may results in a re-specified model that does not address the original research question (ridge regression could help).
- If the original model is correctly specified, then coefficient estimates following variable selection are *biased*. However, the bias may not be overwhelming if you started off with a severe collinearity problem

Notes

Type notes here...

Variable Selection Methods: Cautions (2)

- If our goal is to assess the individual predictors (or their relative impacts), variable selection models have serious implications
  - Standard errors calculated following variable selection overstate the precision of results - they do not control for relevant predictors and they do not account for model selection uncertainty.
  - A new sample may give different results, leading to inconsistent interpretation of "effects"
- These models, again, are really about *prediction* not hypothesis testing, though the can still be quite valuable.

Notes

Type notes here...

Using Regularization Techniques

- Smooth out otherwise complex functions.
- Use alternative methods to identify **important** variables.
- Select features that generate accurate predictions with lower variance.
  - Help solve collinearity problems.
- Theory testing? Not so much...

Notes

Type notes here...