Week 2 Different Regularization Methods

```
library(hdm)
library(xtable)
library(glmnet)
library(ggplot2)
library(dplyr)
library(ggpubr)
```

Model Setup with Different Data Generating Process

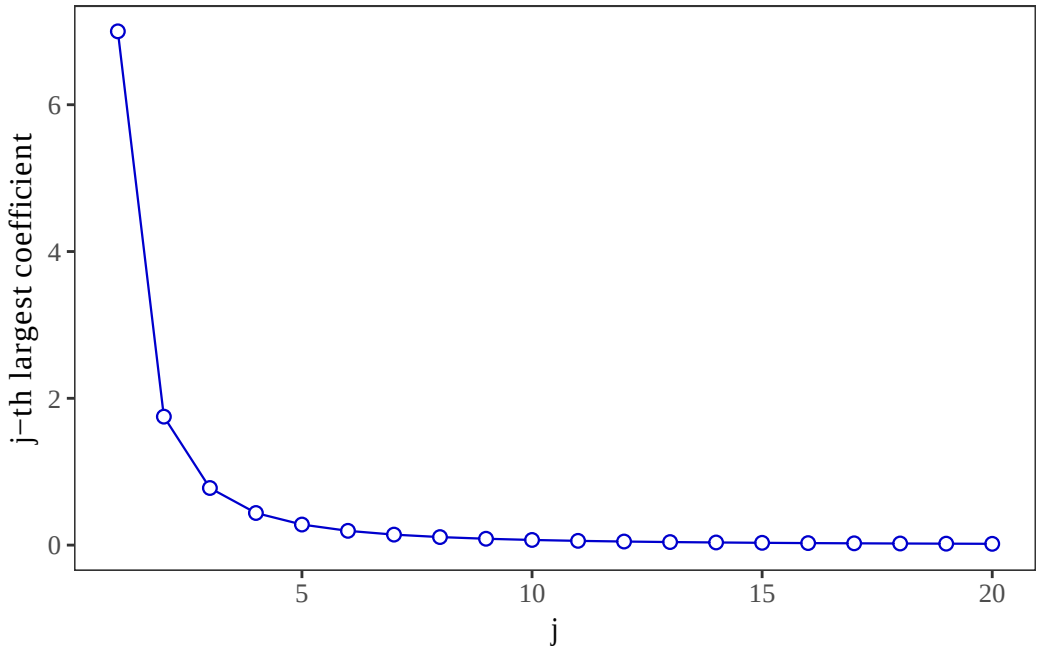
```
## here three types of DGP, sparse, dense, and sparse and dense, are included
gen_data <- function(n, p, regime = "sparse") {

  ## constants chosen to get R^2 of approximately .80
  if (regime == "sparse") {
    beta <- (1 / seq(1:p)^2) * 7
  } else if (regime == "dense") {
    beta <- rnorm(p) * 0.2
  } else if (regime == "sparsedense") {
    beta_1 <- (1 / seq(1:p)^2) * 6.5
    beta_2 <- rnorm(p, 0, 0.5) * 0.7
    beta <- beta_1 + beta_2
  }

  ## true CEF
  true_fn <- function(x) {
    x[, seq_len(dim(x)[2])] %*% beta
  }

  ## generate variables
```

```
  X <- matrix(runif(n * p, min = -0.5, max = 0.5), n, p)
  gX <- true_fn(X)
  y <- gX + rnorm(n)

  Xtest <- matrix(runif(n * p, min = -0.5, max = 0.5), n, p)
  gXtest <- true_fn(Xtest)
  ytest <- gXtest + rnorm(n)

  Xpop <- matrix(runif(100000 * p, min = -0.5, max = 0.5), 100000, p)
  gXpop <- true_fn(Xpop)
  ypop <- gXpop + rnorm(100000)

  return(list(
    X = X, y = y, gX = gX, Xtest = Xtest, ytest = ytest, gXtest = gXtest,
    Xpop = Xpop, ypop = ypop, gXpop = gXpop, beta = beta
  ))
}
```

LASSO Regression in Approximately Sparse Setting

```
## set up the basic parameters
set.seed(1)
n <- 100
p <- 400
res <- gen_data(n, p, regime = "sparse")
X <- res$X
y <- res$y
betas <- res$beta

## plot betas
ggplot(data.frame(
  beta = seq_along(betas),
  magnitude = abs(betas)),
       aes(x = beta, y = magnitude)) +
  geom_line(linetype = "solid", color = "blue3",lwd=0.4) +
  geom_point(shape = 21, size = 2, stroke = 0.6,
             color = "blue3", fill = "white") +
  # scale_y_log10() +   # equivalent to log = "y"
  xlim(1,20) +
  labs(
```

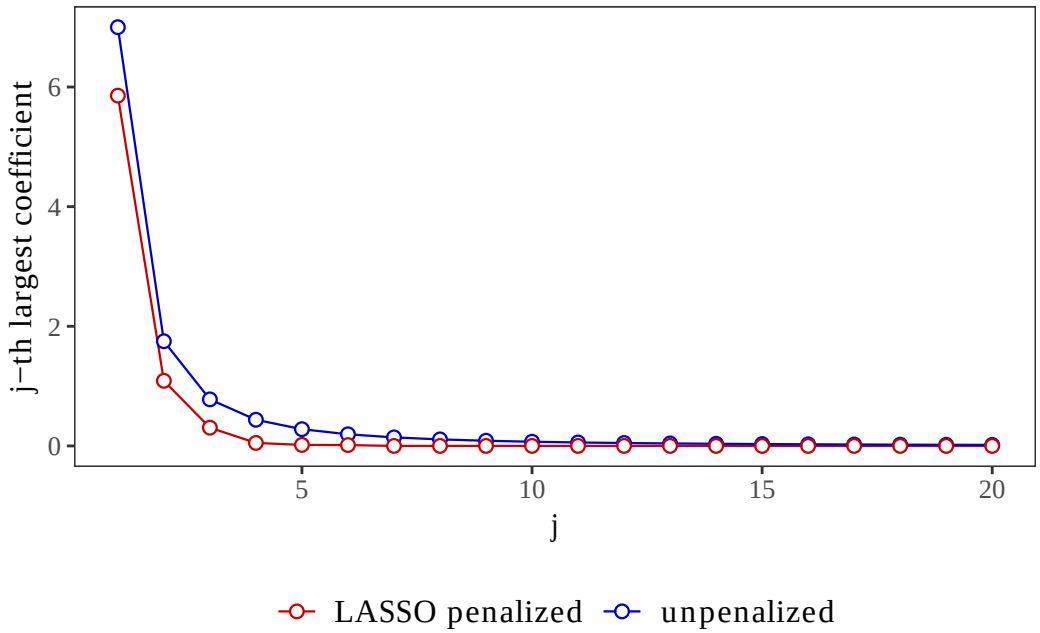
```
    x = "j",
    y = "j-th largest coefficient"
  ) +
  theme_bw() +
  theme(text = element_text(family="Palatino"),
        panel.grid.minor.x = element_blank(),
        panel.grid.major.x = element_blank(),
        panel.grid.minor.y = element_blank(),
        panel.grid.major.y = element_blank(),
        axis.line = element_blank(),
        axis.text.y = element_text(size=10),
        axis.text.x = element_text(size=10),
        axis.title = element_text(size=12))
```




```
## save the figure
## ggsave("Figure 3.1.jpg", dpi=400,width=11,height=7,units = "cm")
```

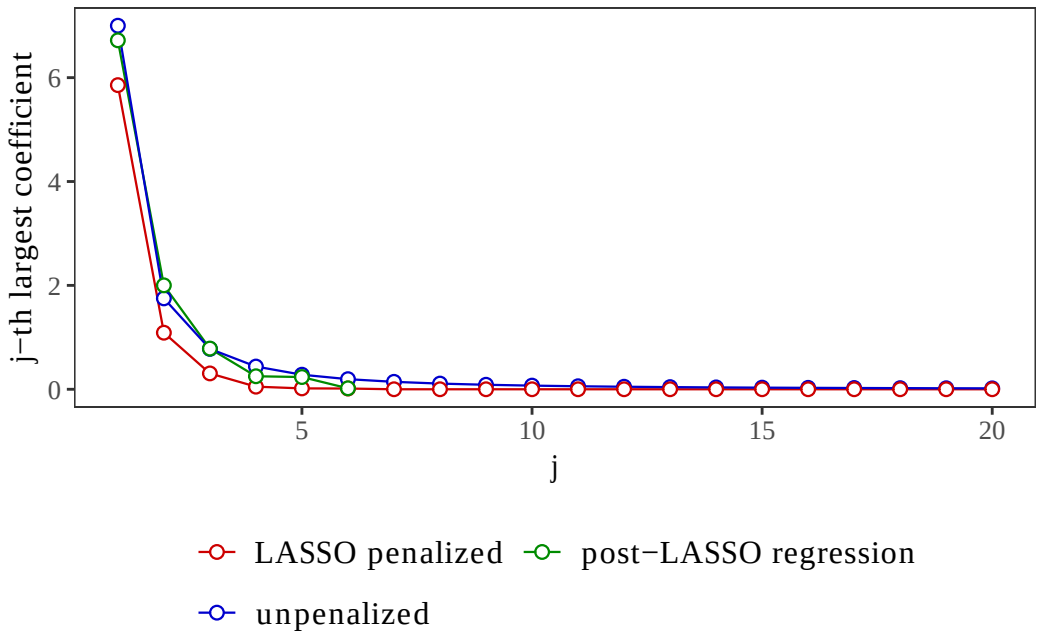
```
## fit Lasso with penalty level determined by CV
fit_lasso_cv <- cv.glmnet(X, y, family = "gaussian",
                          alpha = 1, nfolds = 5)
lasso_betas <- as.numeric(coef(fit_lasso_cv, s = "lambda.min")[,1]) ## cv.glmnet() identifie

## plot
```

```r
ggplot(rbind(
  data.frame(
  beta = seq_along(betas),
  magnitude = abs(betas),
  type = "unpenalized"),
  data.frame(
  beta = seq_along(lasso_betas),
  magnitude = sort(abs(lasso_betas),decreasing = TRUE),
  type = "LASSO penalized")
  ),
       aes(x = beta, y = magnitude, color=type)) +
  geom_line(linetype = "solid", lwd=0.4) +
  geom_point(shape = 21, size = 2, stroke = 0.6, fill = "white") +
  scale_color_manual(values = c("unpenalized" = "blue3",
                                "LASSO penalized" = "red3")) +
  xlim(1,20) +
  labs(
    x = "j",
    y = "j-th largest coefficient"
  ) +
  theme_bw() +
  theme(text = element_text(family="Palatino"),
        legend.title=element_blank(),
        legend.position = "bottom",
        panel.grid.minor.x = element_blank(),
        panel.grid.major.x = element_blank(),
        panel.grid.minor.y = element_blank(),
        panel.grid.major.y = element_blank(),
        axis.line = element_blank(),
        axis.text.y = element_text(size=10),
        axis.text.x = element_text(size=10),
        axis.title = element_text(size=12),
        legend.text=element_text(size=12))
```




```
ggsave("Figure 3.2.jpg", dpi=400,width=11,height=8.5,units = "cm")

## post-Lasso OLS (incorrect but handy)
postlasso_coef <- as.numeric(summary(lm(y~X[,which(lasso_betas!=0)]))$coef[2:7,1])

## plot
ggplot(rbind(
  data.frame(
  beta = seq_along(betas),
  magnitude = abs(betas),
  type = "unpenalized"),
  data.frame(
  beta = seq_along(lasso_betas),
  magnitude = sort(abs(lasso_betas),decreasing = TRUE),
  type = "LASSO penalized"),
  data.frame(
  beta = seq_along(as.numeric(summary(lm(y~X[,which(lasso_betas!=0)]))$coef[2:7,1])),
  magnitude = sort(abs(as.numeric(summary(lm(y~X[,which(lasso_betas!=0)]))$coef[2:7,1])),dec
  type = "post-LASSO regression")
  ),
       aes(x = beta, y = magnitude, color=type)) +
  geom_line(linetype = "solid", lwd=0.4) +
  geom_point(shape = 21, size = 2, stroke = 0.6, fill = "white") +
  scale_color_manual(values = c("unpenalized" = "blue3",
```

```
                                "LASSO penalized" = "red3",
                                "post-LASSO regression" = "green4")) +
  xlim(1,20) +
  labs(
    x = "j",
    y = "j-th largest coefficient"
  ) +
  theme_bw() +
  theme(text = element_text(family="Palatino"),
        legend.title=element_blank(),
        legend.position = "bottom",
        panel.grid.minor.x = element_blank(),
        panel.grid.major.x = element_blank(),
        panel.grid.minor.y = element_blank(),
        panel.grid.major.y = element_blank(),
        axis.line = element_blank(),
        axis.text.y = element_text(size=10),
        axis.text.x = element_text(size=10),
        axis.title = element_text(size=12),
        legend.text=element_text(size=12)) +
  guides(color = guide_legend(nrow = 2, byrow = TRUE))
```



```r
## ggsave("Figure 3.3.jpg", dpi=400,width=11,height=9.3,units = "cm")
```

```r
## I also fit Ridge, Elastic Net, LASSO based on plug-in penalty, and default (correct) post
## family gaussian means that we'll be using square loss
fit_ridge <- cv.glmnet(X, y, family = "gaussian", alpha = 0, nfolds = 5)
fit_elnet <- cv.glmnet(X, y, family = "gaussian", alpha = .5, nfolds = 5)
fit_rlasso <- hdm::rlasso(y ~ X, post = FALSE) ## LASSO with plug-in penalty level
fit_rlasso_post <- hdm::rlasso(y ~ X, post = TRUE) ## post-LASSO with plug-in penalty level
```

```r
r2_score <- function(preds, actual, ytrain = y) {
  rss <- sum((preds - actual)^2) ## residual sum of squares
  ## total sum of squares, we take mean(ytrain) as mean(actual) is an out-of-sample object
  tss <- sum((actual - mean(ytrain))^2)
  rsq <- 1 - rss / tss
  return(rsq)
}

## I use the "population" as the test sample
Xpop <- res$Xpop
ypop <- res$ypop

## calculate R square for each method
r2_lasso_cv <- r2_score(predict(fit_lasso_cv, newx = Xpop, s = "lambda.min"), ypop)
r2_ridge <- r2_score(predict(fit_ridge, newx = Xpop, s = "lambda.min"), ypop)
r2_elnet <- r2_score(predict(fit_elnet, newx = Xpop, s = "lambda.min"), ypop)
r2_lasso <- r2_score(predict(fit_rlasso, newdata = (Xpop)), (ypop))
r2_lasso_post <- r2_score(predict(fit_rlasso_post, newdata = (Xpop)), (ypop))
r2_lava_pop <- lava_yhat_r2(X, Xpop, y, ypop)
```

```
Min Lava Lasso CV Penalty:  0.2358025
Min Lava Ridge CV Penalty:  23.03814
In sample R2 (CV-min):  0.8283694
Out of Sample R2 (CV-min):  0.770166
In sample R2 (Average Across Folds):  0.8269549
Out of Sample R2 (Average Across Folds):  0.7668658
```

```r
table <- matrix(0, 6, 1)
table[1, 1] <- r2_lasso_cv
table[2, 1] <- r2_ridge
table[3, 1] <- r2_elnet
table[4, 1] <- r2_lasso
```

```r
table[5, 1] <- r2_lasso_post
table[6, 1] <- r2_lava_pop[[6]]

colnames(table) <- c("R square (sparse setting)")
rownames(table) <- c(
  "Cross-Validated LASSO", "Cross-Validated Ridge", "Cross-Validated Elastic Net",
  "LASSO (plug-in)", "Post-LASSO", "Lava (plug-in)"
)
tab <- xtable(table, digits = 3)
## print(tab, type = "latex") ## set type="latex" for printing table in LaTeX
print(table)
```

```
                           R square (sparse setting)
Cross-Validated LASSO                      0.77307062
Cross-Validated Ridge                      0.09727575
Cross-Validated Elastic Net                0.74079575
LASSO (plug-in)                            0.77513986
Post-LASSO                                 0.80039759
Lava (plug-in)                             0.77016596
```

‘‘‘

Dense and Sparse and Dense Setting Setup (Omitted)