Carson Slater
STA 5381 Homework #4: Due at the beginning of class on Thursday, April 11*th*.

---

1. (a) For the birth weight data from Homework #3, problem #9, there are some missing data coded with values that are impossible for that particular variable. Remove these values. Hint: Use the summary function in R to identify these values. I found 62 unusual values from the original dataset that should be removed. What are the unusual values given for each variable?

```
bw <- read.delim("birth_weight.txt",
                 sep ="", header = TRUE, dec =".")

before <- nrow(bw)

bw <- bw |>
  dplyr::filter(
    gestation < 999,
    age < 99,
    height < 99,
    weight < 999,
    smoke <= 1
  )

after <- nrow(bw)

before - after

## [1] 62
```

For the `gestation` column, there were many values that were 999. There was also a supposed 99 year-old in the dataset (`age`, very impossible unless you are Sarah in the Old Testament, but even she was only like 90 years old). Likewise, there was a 99" tall woman in the dataset (in the `height` column, which is not likely). There were also a few values for `weight` column that were recorded as 999. Finally, there were a few values for the factor variable `smoke` that were not 0 or 1, but rather 9.

(b) Build a model for birth weight using two different approaches to model building. Describe the two approaches that you use and the models that result. Include the two-way interaction between smoking & gestation in the model-building process.

```
bw <- bw |> dplyr::mutate(
  gestsmoke = gestation*smoke
)

mod1 <- lm(bwt ~ ., data = bw)
mod2 <- lm(bwt ~ gestsmoke, data = bw)
```

```
# forward selection
fw_selectBIC <- step(mod2,
                     direction = "forward",
                     scope = formula(mod1),
                     k = log(nrow(bw))
                     )
```

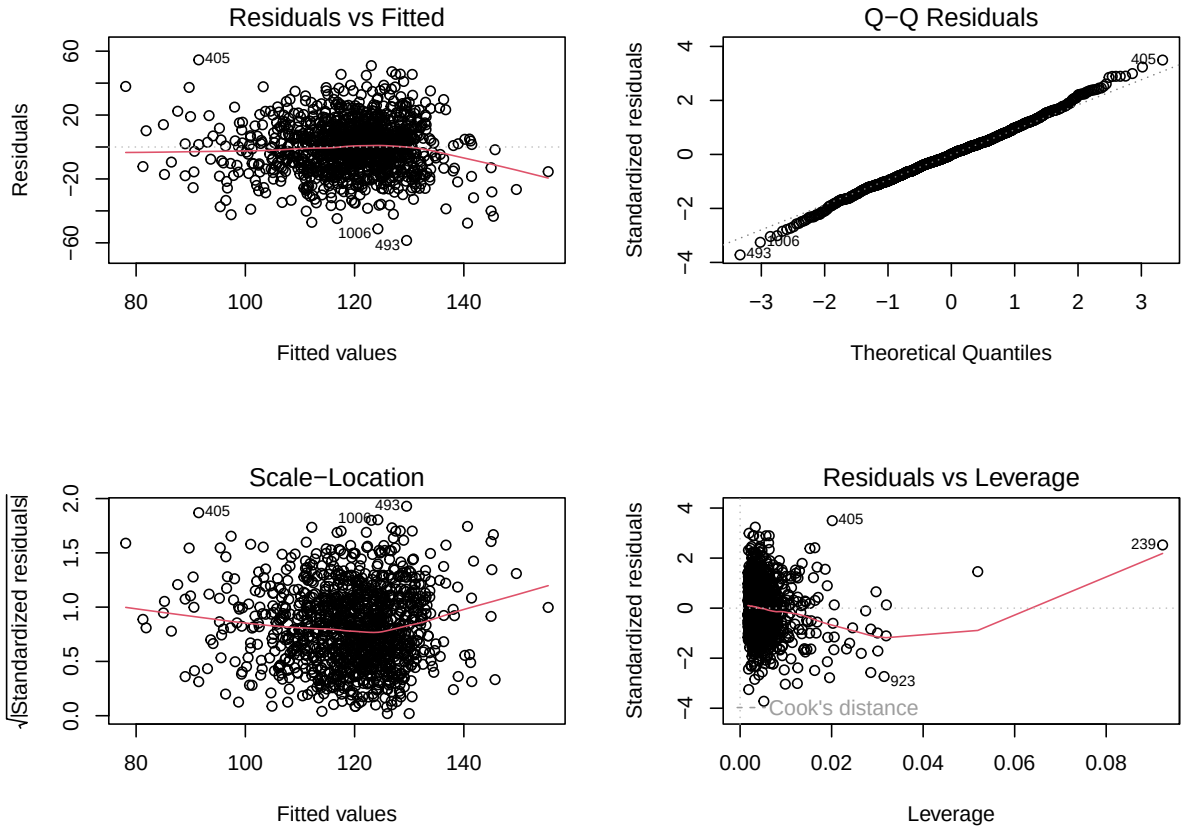
```
fw_selectBIC

##
## Call:
## lm(formula = bwt ~ gestsmoke + gestation + height + smoke + parity,
##     data = bw)
##
## Coefficients:
## (Intercept)    gestsmoke    gestation       height        smoke       parity
##    -63.0426       0.2153       0.3681       1.3130     -68.5154      -3.5442

mod_fw <- lm(bwt ~ parity + height + smoke + gestation + gestsmoke, data = bw)
```

```
# backward selection
bw_selectBIC <- step(mod1,
                     direction = "backward",
                     k = log(nrow(bw))
                     )
```

```
bw_selectBIC

##
## Call:
## lm(formula = bwt ~ gestation + parity + height + smoke + gestsmoke,
##     data = bw)
##
## Coefficients:
## (Intercept)    gestation       parity       height        smoke    gestsmoke
##    -63.0426       0.3681      -3.5442       1.3130     -68.5154       0.2153

mod_bw <- lm(bwt ~ parity + height + smoke + gestation + gestsmoke, data = bw)
```

(c) Do the two models you have developed differ, and if so, how do they differ.
These models do not differ. The same model was selected using forward and backward selection, using BIC as the selection criterion.

```
par(mfrow = c(2,2))
plot(mod_fw)
```



(d) Check the assumptions of the model(s). Most of the model assumptions seem viable with the exception of a few high leverage-high residual points that could be biasing the coefficient estimates. There appears to be normality and constant variance across the residuals for the predicted values.

(e) Check the model(s) for outliers.

In our diagnositc plots, a few observations were flagged due to their residual and leverage values

(f) If you had to choose one of the models (if you have 2 different ones), which would it be, and why?
The same model was selected, so the model is:

$$birthweight = \beta_0 + \beta_1 gestation + \beta_2 parity + \beta_3 height + \beta_4 weight + \beta_5 gestation * smoke + \varepsilon.$$

(g) Use your chosen model to obtain the 95% CI and PI for the birth weight of a baby whose gestational age is 280 days, is the first born, the age of mother is 32, height is 60, weight is 135, and whose mother does not smoke.

```
new_observation <- data.frame(
  "gestation" = 280,
  "parity"    = 0,
  "age"       = 32,
  "height"    = 60,
```

```
  "weight"    = 135,
  "smoke"     = 0,
  "gestsmoke" = 0
)

predict(mod_bw, interval = "confidence", newdata = new_observation)

##        fit      lwr      upr
## 1 118.8032 116.8915 120.7149

predict(mod_bw, interval = "prediction", newdata = new_observation)

##        fit      lwr      upr
## 1 118.8032 87.81612 149.7903
```

2. In this question, you'll form a training and a testing set for the birth weight data.

   (a) Describe a suitable division of the birth weight data into a training and a testing dataset, and create both sets.

   A suitable division of the birthweight data would be some specified ratio (e.g. 80%-20%) split of the data whereby the data is stratified by `smoke` (80% of positive observations for smoke status are in the training data and the other 20% are in the testing set).

```
set.seed(613)
mod_split <- bw |>
  initial_split(
    prop = 0.8,
    strata = c("smoke")
  )

bw_test <- testing(mod_split)
bw_train <- training(mod_split)

bw_train |> nrow(); bw_test |> nrow()
```

```
## [1] 939
## [1] 235
```

   (b) Based on the training dataset, identify 2 potential models using the regsubsets function in the leaps package to choose a model based on either minimizing BIC or maximizing adjusted R^2. The subsets function in the car package may also be helpful.

```
# code for all subsets
mod3 <- lm(bwt ~ ., data = bw)

X <- model.matrix(mod3)[ , c(-1,-8)]

subsets <- leaps::regsubsets(as.matrix(X), y = bw$bwt)
rs <- summary(subsets)
rs
```

```
## Subset selection object
## 6 Variables  (and intercept)
##           Forced in Forced out
## gestation     FALSE      FALSE
## parity        FALSE      FALSE
## age           FALSE      FALSE
## height        FALSE      FALSE
## weight        FALSE      FALSE
## smoke         FALSE      FALSE
## 1 subsets of each size up to 6
## Selection Algorithm: exhaustive
```

```
##          gestation parity age height weight smoke
## 1  ( 1 ) "*"       " "    " " " "    " "    " "
## 2  ( 1 ) "*"       " "    " " " "    " "    "*"
## 3  ( 1 ) "*"       " "    " " "*"    " "    "*"
## 4  ( 1 ) "*"       "*"    " " "*"    " "    "*"
## 5  ( 1 ) "*"       "*"    " " "*"    "*"    "*"
## 6  ( 1 ) "*"       "*"    "*" "*"    "*"    "*"
```

```
library("car")
subsets(mod_subsets, statistic = c("adjr2"), main = expression(paste("Adjusted "
subsets(mod_subsets, statistic = c("bic"), main = "BIC")
```

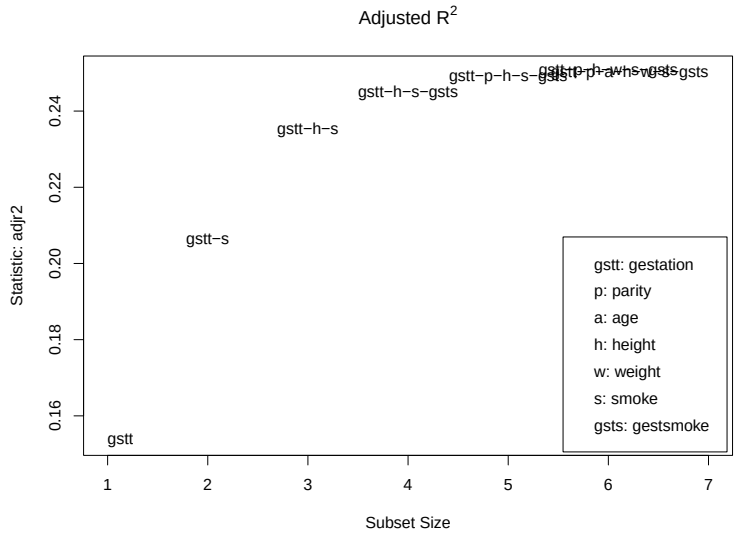
See Figure 1 for the plots from the subsets() function. Here the model that was selected optimizing R^2 kept all features except age, whereas the model that was selected minimizing BIC only contained gestation, smoke, their interaction, and height.

(c) Apply these models to the testing data, and determine which performs the best based on R^2, RMSE, and MAE.

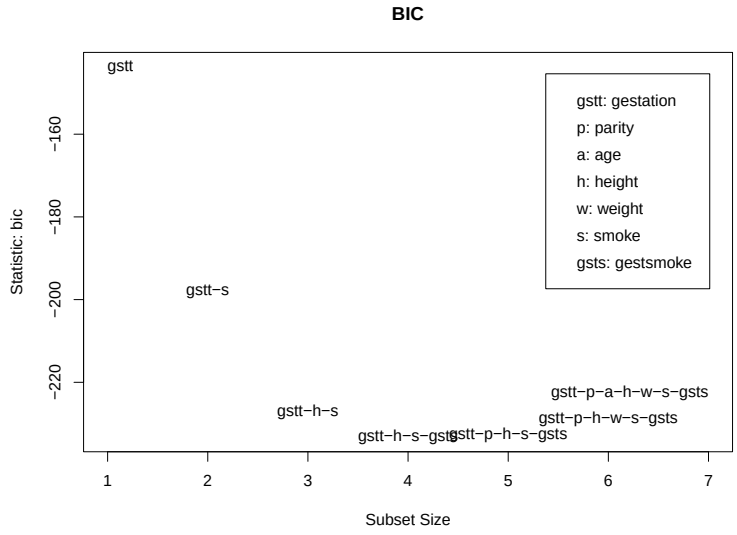
```
mod_bic <- lm(bwt ~ gestation + height + smoke + gestsmoke,
              data = bw_train)
mod_adjr2 <- lm(bwt ~ gestation + parity + age + height + weight + smoke + gests
                data = bw_train)

metrics <- data.frame("Adj R-Squared" = c(summary(mod_adjr2)$adj.r.squared,
                                          sqrt(mean(mod_adjr2$residuals^2)),
                                          mean(abs(mod_adjr2$residuals))),
                      "BIC" = c(summary(mod_bic)$adj.r.squared,
                                          sqrt(mean(mod_bic$residuals^2)),
                                          mean(abs(mod_bic$residuals)))
                      )
rownames(metrics) <- c("R-Squared", "RMSE", "MAE")
```

	Adj. R-Squared	BIC
R-Squared	0.250	0.245
RMSE	15.772	15.855
MAE	12.323	12.369

Between the three model metrics fitted using the training data, the model that was selected using adjusted R^2 has a marginally higher adjusted R^2, marginally lower RMSE and a marginally lower MAE, than the model that was selected by minimizing BIC. So between the three metrics, the model selected by optimizing R^2 was the better for all three metrics; however, the model selected by minimizing BIC was far simpler with marginal performance metrics. Depending on the goals, the BIC could prove to be the 'better' model, as it is simpler.



(a)



(b)

Figure 1: (a) The output of subsets() showing the model that optimizes R^2. (b) The output of subsets() showing the model that optimizes BIC.

3. Use the full birth weight dataset again, removing the 62 unusual values. Apply lasso, ridge, and adaptive lasso to the birth weight data. For the adaptive lasso, construct the weights using pilot estimates of the coefficients from the ridge regression, and use $\gamma = 1$.

   (a) What are the estimated coefficients for each of the variables in each model using the "1se" value of λ? Which variables are removed in each of the models based on the "1se" value of λ?

```
library("glmnet") |> invisible()
```

```
## Loading required package:  Matrix
##
## Attaching package:  'Matrix'
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
## Loaded glmnet 4.1-7
```

```
bw <- read.delim("birth_weight.txt",
                 sep ="", header = TRUE, dec =".")

before <- nrow(bw)

bw <- bw |>
  dplyr::filter(
    gestation < 999,
    age < 99,
    height < 99,
    weight < 999,
    smoke <= 1
  ) |>
  mutate(gestsmoke = gestation*smoke)

mod_full <- lm(bwt ~ ., data = bw)

preds <- bw[,-1] |> as.matrix()
response <- bw[,1]

mod_full <- lm(bwt ~ ., data = bw)

# lasso
mod_lasso <- cv.glmnet(x = preds, y = response)

# ridge
mod_ridge <- cv.glmnet(x = preds, y = response, alpha = 0)

lasso_coefs <- coef(mod_lasso)[,1]
```

```
ridge_coefs <- coef(mod_ridge)[,1]

#Adaptive Lasso with OLS estimates for initial weights
weights <- 1/abs(matrix(coef(mod_full)))
mod_adaptive_lasso <- cv.glmnet(x = preds, y = response,
                                alpha = 1,
                                parallel = TRUE,
                                standardize = TRUE,
                                type.measure = 'auc',
                                penalty.factor = weights[-1,])


a_lasso_coefs <- coef(mod_adaptive_lasso)[,1] |> unname()


mod_coefs <- data.frame(
  "Lasso"          = lasso_coefs,
  "Ridge"          = ridge_coefs,
  "Adaptive Lasso" = a_lasso_coefs
)

rownames(mod_coefs) <- coef(mod_adaptive_lasso)@Dimnames[[1]]
```

	Lasso	Ridge	Adaptive.Lasso
(Intercept)	-22.289	1.643	-29.895
gestation	0.353	0.252	0.298
parity	0.000	-1.394	-2.492
age	0.000	0.015	0.000
height	0.683	0.684	1.096
weight	0.013	0.047	0.000
smoke	-5.450	-3.324	-8.771
gestation*smoke	0.000	-0.010	0.000

Table 1: Coefficients for each respective model that are removed are the coefficients in the rows with a red '0.000'. Here the lasso and the adaptive lasso removed three predictors, each, but they were not all the same predictors. age and gestation*smoke were removed from both, while the lasso also removed parity while the adaptive lasso removed weight.

(b) In the lasso and adaptive lasso models, which variables' coefficient is the last to be forced to zero as λ increases?

For both models with the lambda.1se coefficient estimates, the sixth predictor is the model that gets driven to zero last as λ increases. This is the coefficient that corresponds to the gestation for the lasso model, and smoke for the adaptive lasso model.

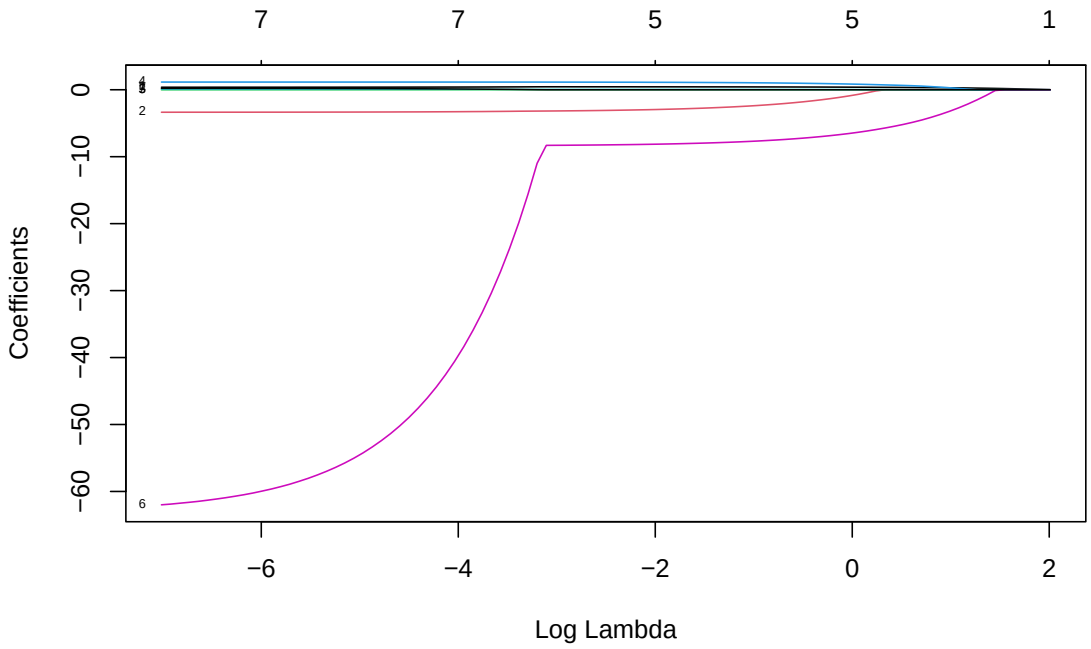


Figure 2: Coefficients for the lasso model as coefficients get driven to zero.

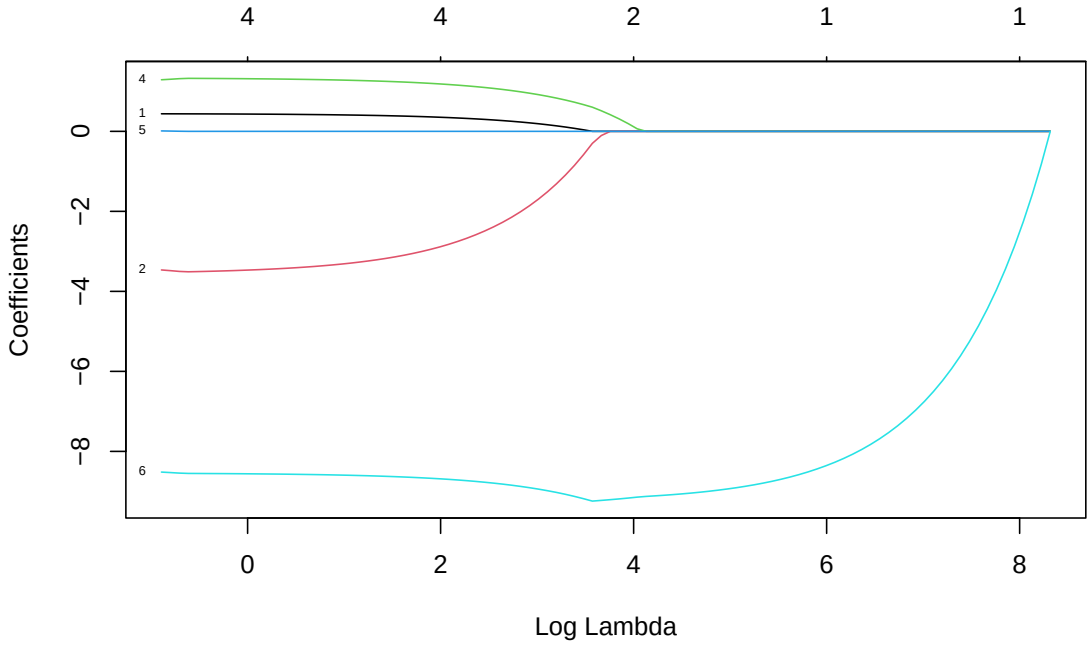


Figure 3: Coefficients for the adaptive lasso model as coefficients get driven to zero.

4. One of the drawbacks of lasso/adaptive lasso is that the uncertainty in the coefficient estimates is not automatically provided in the output because there are no closed forms. In this question, we will use bootstrap samples to estimate the standard errors of the estimated coefficients.

   (a) Create 1,000 bootstrap samples, assuming that the predictors are random. For each bootstrap sample, estimate and save the coefficients from adaptive lasso, using ridge regression estimates for the pilot values and $\gamma = 1$. Show your code.

```
n_samples <- 1000
n_obs <- 1000
my_dfs <- vector("list", n_samples)

set.seed(613)

for (i in seq_len(n_samples)) {
  my_dfs[[i]] <- data.frame(
    "Birthweight" = sample(bw[,1], n_obs, replace = TRUE),
    "gestation"   = sample(bw[,2], n_obs, replace = TRUE),
    "parity"      = sample(bw[,3], n_obs, replace = TRUE),
    "age"         = sample(bw[,4], n_obs, replace = TRUE),
    "height"      = sample(bw[,5], n_obs, replace = TRUE),
    "weight"      = sample(bw[,6], n_obs, replace = TRUE),
    "smoke"       = sample(bw[,7], n_obs, replace = TRUE),
    "gestsmoke"   = sample(bw[,8], n_obs, replace = TRUE)
  )
}

ad_lasso_non_boot <- coef(
  cv.glmnet(x = bw[,-1] |> as.matrix(),
            y = bw[,1],
            alpha = 0,
            parallel = TRUE,
            standardize = TRUE,
            type.measure = 'auc',
            penalty.factor = weights[-1,]),
  s='lambda.1se')[,1] |> unname()

mod_interact <- lm(bwt ~ ., data = bw)

weights <- 1/abs(matrix(coef(mod_interact)))

bootstrap_coefs <- lapply(my_dfs,
                          \(df) {
                            coef(
                              cv.glmnet(x = df[,-1] |> as.matrix(),
                                        y = df[,1],
                                        alpha = 0,
```

```
                                        parallel = TRUE,
                                        standardize = TRUE,
                                        type.measure = 'auc',
                                        penalty.factor = weights[-1,])
                            )[,1]
                          }
                        )


bootstrap_coefs <- bootstrap_coefs |>
  reduce(.f = rbind) |>
  as.data.frame()

numeric_vars <- names(bootstrap_coefs)
numeric_vars[8] <- "Gestation*Smoke"
rownames(bootstrap_coefs) <- NULL
bootstrap_coefs |> head(5)
```

```
##   (Intercept)     gestation        parity           age        height
## 1     119.008  2.407183e-37 -3.485670e-35 -4.746340e-39 -2.623437e-36
## 2     118.873 -1.931028e-37  5.486176e-36  3.655228e-39  3.249195e-37
## 3     119.412 -1.514553e-37 -1.075170e-34 -1.882661e-38 -8.727278e-37
## 4     119.243  5.068905e-38  8.340160e-35 -1.170244e-38  2.280998e-36
## 5     118.931  9.901036e-38 -6.457399e-35  2.369248e-38 -5.621500e-37
##          weight         smoke     gestsmoke
## 1  9.817175e-39 -2.112668e-34 -8.195555e-39
## 2 -3.359668e-39  2.657527e-33 -1.331640e-38
## 3  2.736657e-38 -7.235316e-35  8.560523e-39
## 4  7.169757e-39 -8.740604e-34 -1.833888e-38
## 5 -1.842866e-39 -7.762671e-36  7.777076e-39
```

(b) Create a 2×4 matrix of histograms of the 8 bootstrapped coefficients, including the intercept. On each one, overlay a vertical line for the observed coefficient and two dashed vertical lines for the 2.5% and 97.5% quantiles of the bootstrapped coefficients.

```
cols <- viridisLite::viridis(8)

# Finish making histograms

l_quantile <- sapply(1:8, \(i) quantile(bootstrap_coefs[,i], 0.025)) |> unname()
u_quantile <- sapply(1:8, \(i) quantile(bootstrap_coefs[,i], 0.975)) |> unname()

par(mfrow = c(2, 4))
for (i in 1:8) {
  hist(bootstrap_coefs[,i],
       col = cols[i],
       main = paste("Histogram of", numeric_vars[i]),
```

```
       xlab = paste(numeric_vars[i]),
       breaks = "FD"
      )
  abline(v = l_quantile[i], col = "red", lty = 2)
  abline(v = u_quantile[i], col = "red", lty = 2)
  abline(v = a_lasso_coefs[i], col = "red", lty = 3, lwd = 2)
}
```

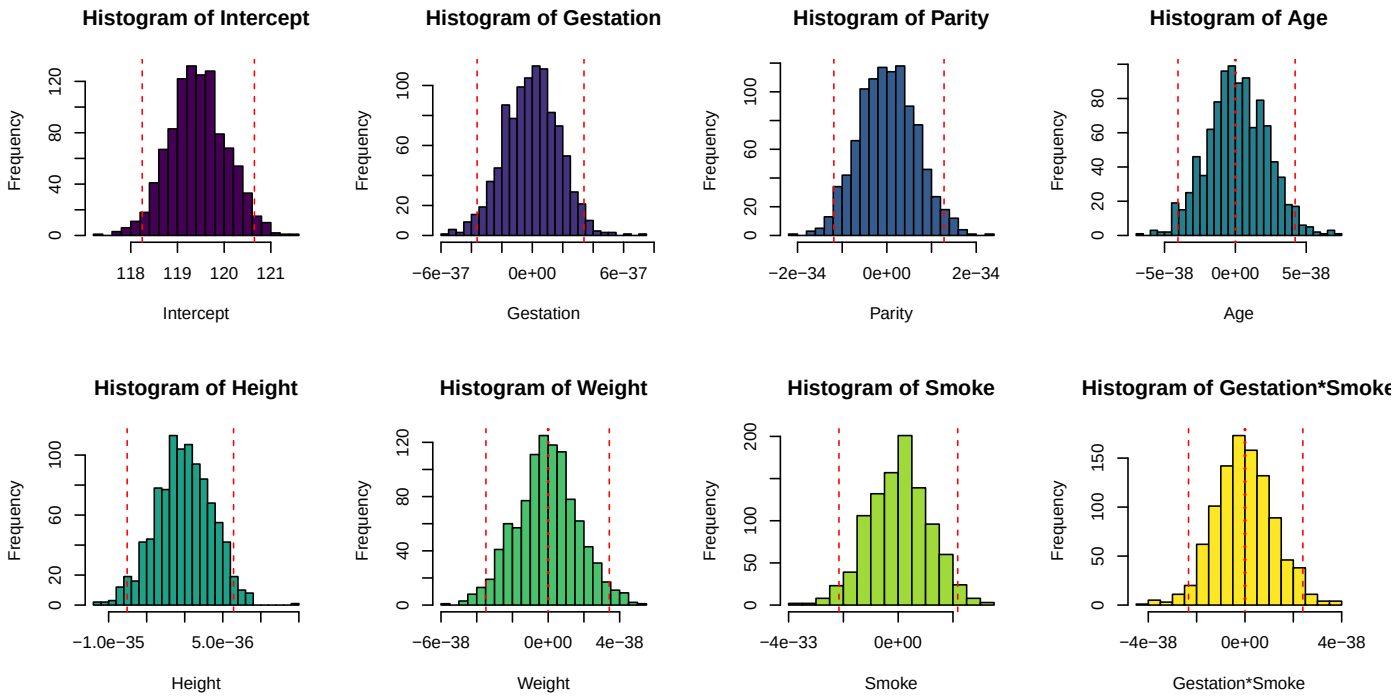


Figure 4: Histograms of coefficients from bootstrap samples for the adaptive lasso model, with ridge regression pilot estimates

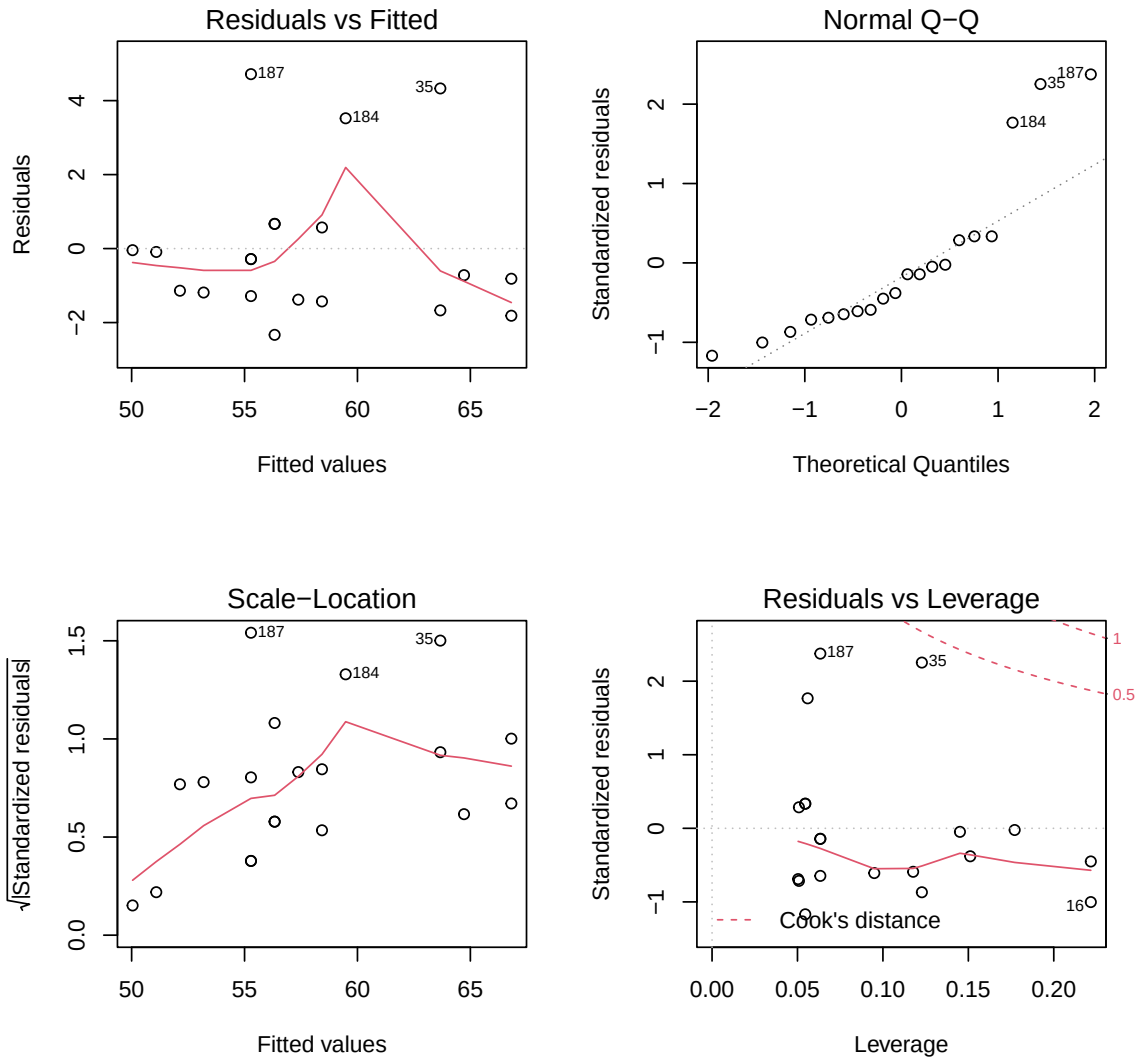
(c) Was the bootstrap approach successful in obtaining standard errors for the variables whose coefficients were constrained to be zero by adaptive lasso in the observed data? Explain your answer.

The bootstrap approach was successful in creating standard errors for the intercepts, because they yielded symmetric distributions where bootstrap standard errors can be computed.

5. From the reported versus measured weights of 98 females after having removed the row with missing data and the one outlier (See Example 8.2 used in the class notes.), take the following 20 rows of observations, and answer the following questions:

```
index <- c(74, 89, 78, 23, 86, 70,  4, 55, 93,  7, 97,
           91, 43, 82, 61, 12, 51, 72, 18, 25)
```

```
index <- c(74, 89, 78, 23, 86, 70,  4, 55, 93,  7,
           97, 91, 43, 82, 61, 12, 51, 72, 18, 25)
# a
library("car") |> invisible()

## Loading required package:  carData
##
## Attaching package:  'car'
## The following object is masked from 'package:purrr':
##
##     some
## The following object is masked from 'package:dplyr':
##
##     recode

#Getting just Measured & Reported weights for 100 Women
fdata <- Davis[Davis$sex== "F", ]

#Find the missing data & remove them
missing <- which(complete.cases(fdata) == FALSE)
fdata <- fdata[-missing, ]
fdata <- fdata[-4, ]

my_data <- fdata[index,]

mod_fdata <- lm(weight ~ repwt, data = my_data)

par(mfrow = c(2, 2))
plot(mod_fdata)
```



(a) Why is a bootstrap approach for estimating the measured weight for a woman with a reported weight of 62.5 kg needed?

Here, we observe deviations from normality in the residuals; a small sample size would also indicate that parametric estimation might not be robust or sufficient.

(b) Assume that the reported weights are random and that a 95% confidence interval for the measured weight of a woman with a reported weight of 62.5 kg is desired. Using 2,000 bootstrap samples, obtain the 3 bootstrapping-based confidence intervals: a normal theory interval with bootstrap standard error, a percentile interval, and a bias-corrected interval.

```
# b
# Assume predictors are fixed

set.seed(613)
x_star <- 62.5
```

```r
B <- 2000
n <- dim(my_data)[1]
pred_star <- array()
# fit <- fitted(mod_fdata)
# res <- resid(mod_fdata)
pred_hat <- mod_fdata$coef[1] + mod_fdata$coef[2]*x_star

mean_repwt <- numeric(B)

for(i in 1:B){
  new_rows <- sample(1:n, n, replace = T)
  new_data <- my_data[new_rows,]
  mod_star <- lm(new_data$weight ~ new_data$repwt)
  mean_repwt[i] <- mean(new_data$repwt)
  pred_star[i] <- mod_star$coef[1] + mod_star$coef[2]*x_star
}

SE_pred_star <- sd(pred_star)
mean_pred_star <- mean(pred_star)
SE_pred_star <- sqrt(sum((pred_star - mean_pred_star)^2) / (B - 1))
round(SE_pred_star, 4)

## [1] 0.7839

# normal theory interval
norm_int <- c(
  "lower" = round(pred_hat - 1.96*SE_pred_star, 4),
  "upper" = round(pred_hat + 1.96*SE_pred_star, 4)
  )

# Percentile Interval
lower <- 0.025*B #25
upper <- 0.975*B #975
percent_int <- c(
  "lower" = round(sort(pred_star)[lower], 4),
  "upper" = round(sort(pred_star)[upper], 4)
  )

# Adjusted Percentile Interval
z <- qnorm((length(which(pred_star < pred_hat)))/B)

jk <- array()
for(i in 1:n){
  bs.mod <- lm(my_data$weight[-i] ~ my_data$repwt[-i])
  jk[i] <- bs.mod$coef[1] + bs.mod$coef[2]*x_star
}
jkbar <- mean(jk)
```

```r
A <- sum((jk - jkbar)^3)/(6*(sum((jk - jkbar)^2))^1.5)

z.a.2 <- qnorm(0.975)
A1 <- pnorm(z + (z - z.a.2) / (1 - A*(z - z.a.2)))
A2 <- pnorm(z + (z + z.a.2) / (1 - A*(z + z.a.2)))

lower <- round(B*A1) #32
upper <- round(B*A2) #1929

bias_adj_int <- c(
  "lower" = round(sort(pred_star)[lower], 4), #61.9082
  "upper" = round(sort(pred_star)[upper], 4) #64.8516
  )
norm_int; percent_int; bias_adj_int

## lower.(Intercept) upper.(Intercept)
##           61.6098           64.6827
##   lower   upper
## 61.9495 65.0380
##   lower   upper
## 61.9082 64.8516
```

Here, the normal theory interval is (61.60, 64.68), the percentile interval is (61.94, 65.04), and the bias-adjusted percentile interval is (61.91, 64.85).

(c) Plot the distribution of estimated reported weights based on the bootstrap samples. How does the bootstrap distribution of estimated reported weights differ from a normal distribution?

This distribution seems right skewed, which indicates it differs from a normal distribution.

(d) What is the width of each of the three intervals, and which would appear to be the most relevant for this sample?

```r
diff(norm_int) |> unname()

## [1] 3.0729

diff(percent_int) |> unname()

## [1] 3.0885

diff(bias_adj_int) |> unname()

## [1] 2.9434
```

The differences are shown in the output above, and the most relevant one for this sample is the bias-adjusted percentile interval to mitigate the bias in the Kernel and the bootstrap samples. Additionally, it is also the shortest which also helps with desirability.

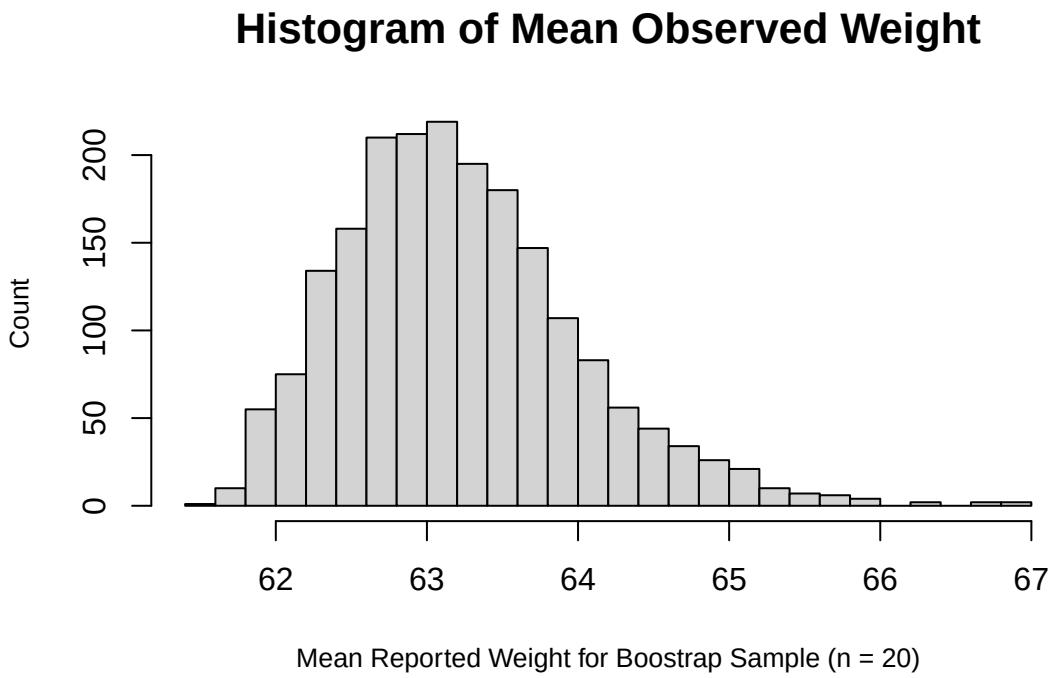


Figure 5: Histogram of bootstrap predicted measured weight for a reported weight of 62.5kg.

6. From Example 8.4 in the class notes, what value of h is selected via cross-validation in which $\hat{m}(\cdot)$ is the Nadaraya-Watson estimator? Report the value of h, and give your code.

```r
set.seed(77); n <- 200
xi <- runif(n, 0, 1)
yi <- 15*(1 + xi*cos(4*pi*xi)) + rnorm(n)
x <- seq(min(xi), max(xi), len = 1000)
f <- function(x) 15*(1 + x*cos(4*pi*x))
data <- data.frame(
  "x"  = xi,
  "y" = yi
)

# Make a Nadaraya-Watson Function
nadaraya_watson <- function(xi, x, y, h) {
  weights <- dnorm((xi - x) / h)
  numerator <- sum(weights * y)
  denominator <- sum(weights)
  if (denominator == 0) return(NA)  # Avoid division by zero
  return(numerator / denominator)
}

# Define the function for cross-validation to find optimal bandwidth
find_best_h <- function(data, h_grid) {
  # Initialize variables to store results
  # saves R from 'growing vectors problem'
  cv_errors <- rep(0, length(h_grid))

  # Perform leave-one-out cross-validation for each bandwidth
  for (i in 1:length(h_grid)) {
    h <- h_grid[i]
    errors <- rep(0, nrow(data))
    for (j in 1:nrow(data)) {
      xi <- data$x[j]
      x_train <- data$x[-j]
      y_train <- data$y[-j]
      # implementing NW function
      y_pred <- nadaraya_watson(xi, x_train, y_train, h)
      errors[j] <- (y_pred - data$y[j])^2 # Using RMSE
    }
    cv_errors[i] <- sqrt(mean(errors)) # dividing by n does not change min
  }

  # Find the optimal bandwidth with the lowest cross-validated error
  h_grid[which.min(cv_errors)]
}
```

```r
# creating a grid to search for h
h <- seq(0, 2, by = 0.0001)

# Find the bandwidth
best_h <- find_best_h(data, h)
print(paste("Optimal bandwidth:", best_h))

## [1] "Optimal bandwidth: 0.0089"
```

The optimal bandwidth here is $h = 0.0089$.

7. In Example 8.5 of the class notes (data are available on the class website in the file Hardness.txt), the value of the span parameter that was selected based on cross-validation of a local polynomial fit of order 2 to the hardness_6Al variable was 0.483.

   (a) Recreate the code to obtain this same result. Give your code.

```
hardness <- read.delim("Hardness.txt",
                       sep ="", header = TRUE, dec =".")

find_best_span <- function(data, span_grid, degree = 1) {
  cv_errors <- rep(0, length(span_grid))

  # Perform leave-one-out cross-validation for each span
  for (i in 1:length(span_grid)) {
    s <- span_grid[i]
    errors <- rep(0, nrow(data))
    for (j in 1:nrow(data)) {
      # using the loess() function
      y_pred <- loess(hardness_6Al ~ d_mm, degree = degree,
                      data = hardness[-j,], span = s)
      errors[j] <- (y_pred$residuals)^2
    }
    cv_errors[i] <- sqrt(mean(errors))
  }

  # Find the optimal bandwidth with the lowest cross-validated error
  span_grid[which.min(cv_errors)]
}

# a

s_grid <- seq(0.3, 1, by = 0.001)

best_span_deg2 <- find_best_span(hardness, s_grid, 2)

print(paste("Optimal span for degree 2 approx:", best_span_deg2))
```

```
cat("Optimal span for degree 2 approx: 0.396")

## Optimal span for degree 2 approx: 0.396
```

   (b) Reselect the span based on a local polynomial fit of order 1. What value minimizes CV?

```
# b
s_grid <- seq(0.3, 2, by = 0.001)

best_span_deg1 <- find_best_span(hardness, s_grid, 1)
```

```
print(paste("Optimal span degree 1 approx:", best_span_deg1))
```

```
cat("Optimal span for degree 1 approx: 1.617")

## Optimal span for degree 1 approx: 1.617
```

(c) Overlay both of the fitted curves from parts (a) and (b) on the data. Which do you prefer?

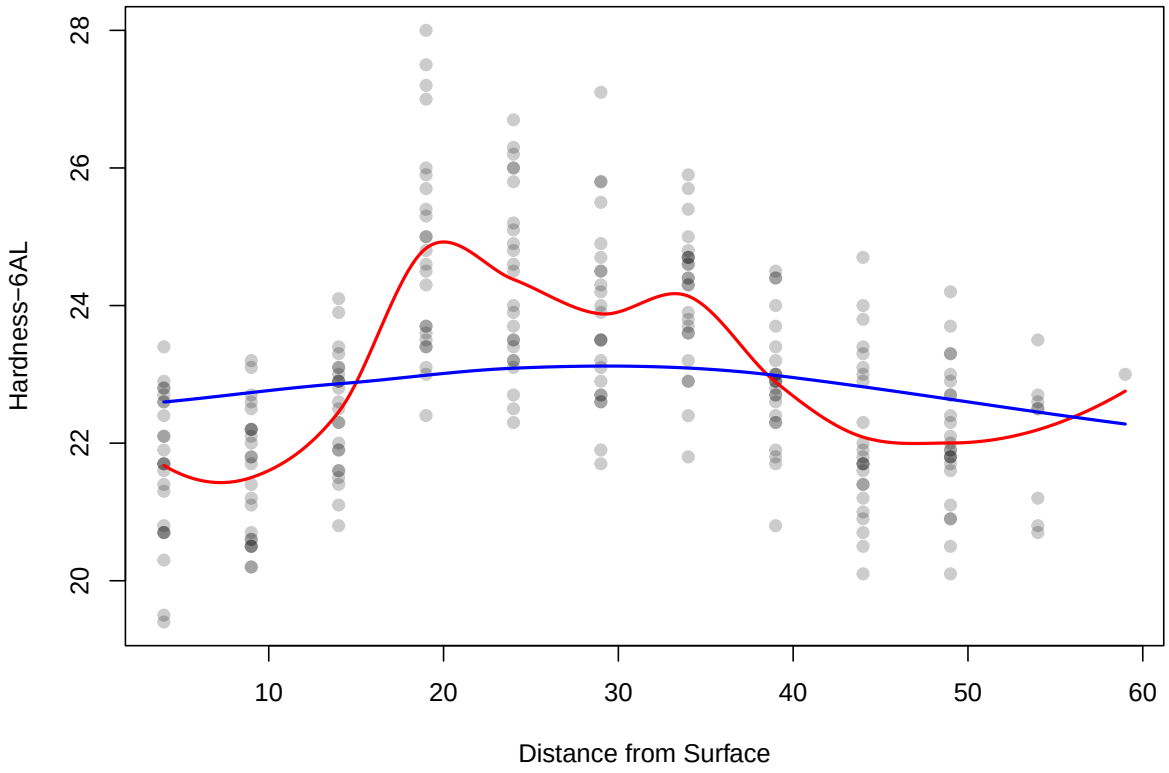


Figure 6: Two loess curves with optimal span, with blue as the degree 1 approximation, and red as the degree 2 approximation.