HW6

Robert Chandler

2024-07-21

Setup

Begin by reading the dataset for this set of problems:

```
le <- read.csv("../../datasets/life_expectancy.csv")
y <- le$X2015Life.expectancy
x1 <- le$Medical.doctors
x2 <- le$Nurses
x3 <- le$Pharmacists
```

Problem 1

For the life expectancy data, Consider the MLR model given by $Y \sim X_1 + X_2 X_3$

```
model <- lm(y ~ x1 + x2 + x3)
```

1.a

Obtain a partial regression plot for X_1, X_2, and X_3 and discuss whether the regression relationships in the fitted regression function are inappropriate for any of the predictor variables.

```
library(car)
avPlots(model)
```

The partial regression plots show that adding a linear term for X_1 would be a helpful addition to the regression model already containing X_2, X_3. Adding in X_3 to a model already containing the other variables would be a helpful addition, but the linear increase in Y as a result is not as strong as that of X_1. The same goes for adding X_2 to a model with the others already included. The partial plots for X_2, X_3 are somewhat close to zero, so depending on what kind of confidence is desired, it is possible that these might not be considered appropriate to add to the model, but we conclude for this case that they are. As such, the relationships in the fitted regression function are appropriate for all predictor variables.

1.b

Check for influential points by calculating DFFITS, DFBETAS and Cook's distance values.

DFFITS

Checking the influence on single predictions using DFFITS using the criterion:

$$|DFFITS_i| > 2\sqrt{\frac{p}{n}}$$

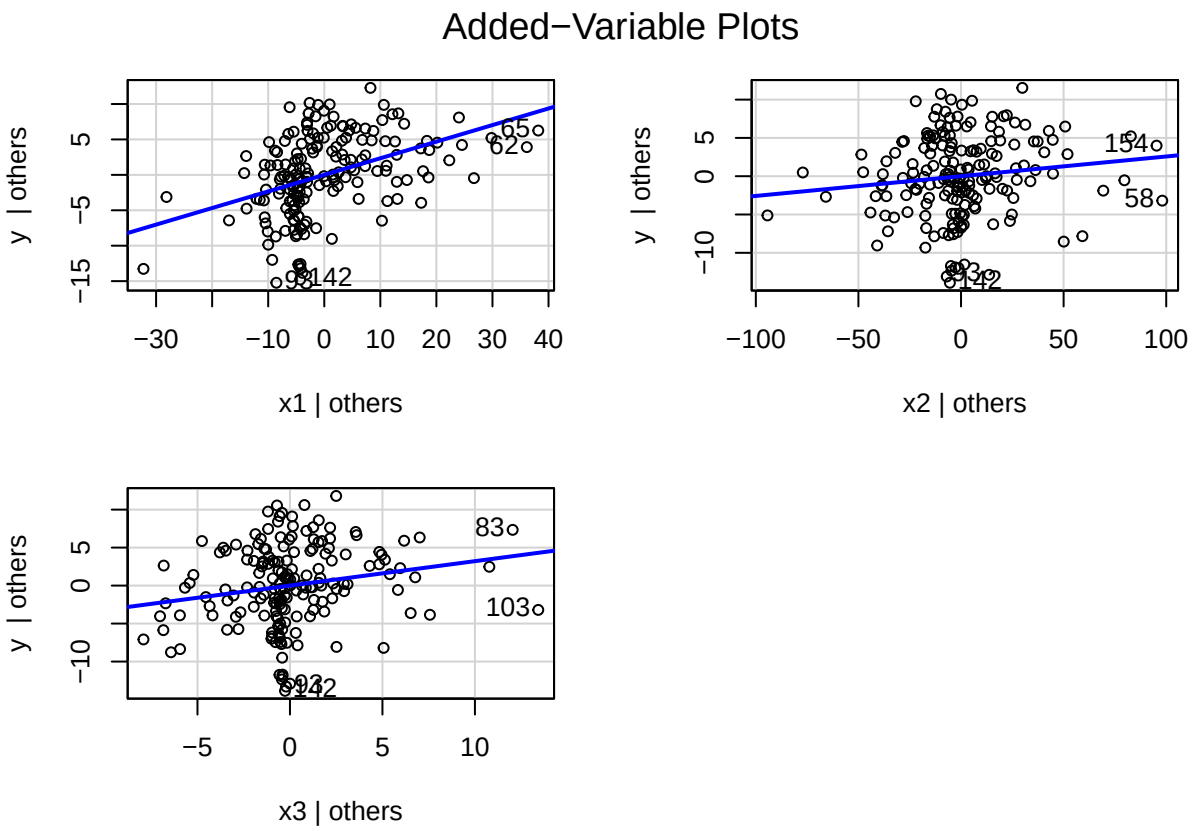

Figure 1: Partial Regression Plot

```r
n <- length(y)
p <- length(model$coefficients)
influential_dffits <- abs(dffits(model)) > 2 * sqrt(p / n)
```

There are 6 influential points according to the DFFITS criterion. The following rows in the dataset (starting at an index of 1) are influential: $i = \{16, 58, 103, 142, 153, 172\}$.

DFBETAS

Checking the influence on the regression coefficients using DFBETAS using the criterion:

$$|DFBETAS_k(i)| > \frac{2}{\sqrt{n}}$$

```r
influential_dfbetas <- abs(dfbetas(model)) > 2 / sqrt(n)
```

There are 40 total influential points according to the DFBETAS criterion. The breakdown of how many correspond to each coefficient is:

```r
colSums(influential_dfbetas)
```

```
## (Intercept)          x1          x2          x3
##          17           7           6          10
```

The following shows which cases are influential on which regression parameters:

```r
influential_dfbetas[rowSums(influential_dfbetas) != 0, ]
```

```
##     (Intercept)    x1    x2    x3
## 4          TRUE FALSE FALSE FALSE
## 12         TRUE FALSE FALSE FALSE
## 15        FALSE FALSE FALSE  TRUE
## 16         TRUE FALSE  TRUE  TRUE
## 30         TRUE FALSE FALSE FALSE
## 32         TRUE FALSE FALSE FALSE
## 33         TRUE FALSE FALSE FALSE
## 40         TRUE FALSE FALSE FALSE
## 46        FALSE FALSE FALSE  TRUE
## 50         TRUE FALSE FALSE FALSE
## 58        FALSE  TRUE  TRUE  TRUE
## 62        FALSE  TRUE FALSE  TRUE
## 65        FALSE  TRUE  TRUE FALSE
## 72         TRUE FALSE FALSE FALSE
## 82         TRUE FALSE FALSE FALSE
## 83        FALSE  TRUE FALSE  TRUE
## 93         TRUE  TRUE FALSE FALSE
## 103        TRUE FALSE FALSE  TRUE
## 107       FALSE  TRUE  TRUE FALSE
## 109        TRUE FALSE FALSE FALSE
## 113       FALSE FALSE FALSE  TRUE
## 117        TRUE FALSE FALSE FALSE
## 142        TRUE FALSE FALSE FALSE
## 145       FALSE FALSE  TRUE FALSE
## 148        TRUE FALSE FALSE FALSE
## 153        TRUE FALSE  TRUE FALSE
## 155       FALSE FALSE FALSE  TRUE
```

```
## 172       FALSE  TRUE FALSE  TRUE
```

The row/index of the case is shown on the left, and the four columns correspond to whether or not that case is influential on the parameter labeled in the respective column header.

Cook's Distance

Checking which cases are influential on all fitted values using Cook's distance with the criterion:

```
f_crit <- qf(0.2, p, n - p)
```

$$D_i > F(0.2, p, n-p) = F(0.2, 4, 174) = 0.4117841$$

```
influential_cooks <- cooks.distance(model) > f_crit
```

There are 0 influential cases under the criterion for Cook's distance, so we conclude that there are no cases in the dataset that are significantly influential on *all* fitted values.

1.c

Determine whether there is multicollinearity present based on VIF.

```
car::vif(model)
```

```
##       x1       x2       x3
## 2.952095 2.817523 2.192316
```

All of the VIF terms are between 2 and 3 with a mean of 2.653978 which is much less than 10, which is the point at which we would consider there to be a serious multicollinearity issue. These values are still somewhat greater than 1, and the mean value indicates that the expected sum of the squared errors in the LS standardized regression coefficients is 2.653978 times what it would be if the X variables were uncorrelated. Therefore, we conclude that while there may be some multicollinearity present, it definitely should not be classified as a serious issue.

Problem 2

Use R to summarize the OLS model. Then compute the confidence interval for the linear impact of X_1, X_2, and X_3.

Showing the R summary of the previously calculated OLS model:

```
summary(model)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.7701  -3.0922  -0.1913   3.8118  10.9957
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 64.47982    0.60561 106.471  < 2e-16 ***
## x1           0.23418    0.03936   5.950 1.44e-08 ***
## x2           0.02574    0.01467   1.755   0.0810 .
## x3           0.32116    0.12417   2.586   0.0105 *
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.27 on 174 degrees of freedom
## Multiple R-squared:  0.5851, Adjusted R-squared:  0.578
## F-statistic:  81.8 on 3 and 174 DF,  p-value: < 2.2e-16
```

Computing the (95%) confidence interval for the linear impact of each input variable:

```
confint(model)[2:4, ]
```

```
##           2.5 %     97.5 %
## x1  0.156496305 0.31186592
## x2 -0.003205879 0.05469436
## x3  0.076085264 0.56623243
```

Problem 3

Use R to complete the WLS analysis on the model (note: just do one iteration). Discuss the difference. Then compute the confidence interval for the linear impact of X_1, X_2, X_3.

```
w <- 1 / fitted(lm(abs(residuals(model)) ~ x1 + x2 + x3))^2
model_wls <- lm(y ~ x1 + x2 + x3, weight = w)
summary(model_wls)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3, weights = w)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1873 -0.7750 -0.0138  0.9943  2.7705
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 64.98956    0.62589 103.835  < 2e-16 ***
## x1           0.21096    0.03453   6.110 6.36e-09 ***
## x2           0.02538    0.01328   1.911  0.05768 .
## x3           0.31598    0.11362   2.781  0.00601 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.265 on 174 degrees of freedom
## Multiple R-squared:  0.5829, Adjusted R-squared:  0.5757
## F-statistic: 81.06 on 3 and 174 DF,  p-value: < 2.2e-16
```

- The estimated values of the parameters are very similar in WLS to the OLS model.
- The standard errors have decreased slightly for the parameters, the t-values are comparable to OLS, and the p-values have all decreased, meaning the estimates on our parameters have become more significant.
- The residual standard error should not be compared as the residuals have been modified.
- The R-squared values have remained nearly exactly the same with the values for WLS dropping just slightly.
- The global F-statistic is nearly the same as well, dropping slightly in the WLS model.

The confidence interval for the linear impact of the input variables is:

```r
confint(model_wls)[2:4, ]
```

```
##            2.5 %     97.5 %
## x1  0.1428067772 0.27910479
## x2 -0.0008354001 0.05159353
## x3  0.0917393982 0.54022606
```

This interval is very similar to the previous one, but has tightened up just a bit more than the OLS interval.

Problem 4

Use R to complete the Robust analysis on the model. Then compute the confidence interval for the linear impact of X_1, X_2, and X_3.

```r
library(MASS)
model_rlm <- rlm(y ~ x1 + x2 + x3, psi = psi.bisquare)
summary_rlm <- summary(model_rlm)
summary_rlm
```

```
##
## Call: rlm(formula = y ~ x1 + x2 + x3, psi = psi.bisquare)
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.0397  -3.2629  -0.2522   3.6421  10.7342
##
## Coefficients:
##             Value    Std. Error t value
## (Intercept)  64.7524   0.6268   103.2982
## x1            0.2226   0.0407     5.4635
## x2            0.0269   0.0152     1.7742
## x3            0.3325   0.1285     2.5872
##
## Residual standard error: 5.162 on 174 degrees of freedom
```

- The Robust model is once again very similar to the previous models.
- It shares similar parameter estimates to the other two models.
- This time, the standard errors of the parameters *increased* slightly relative to the values in the OLS model.
- The residuals have once more been modified, so the residual standard error cannot be compared to OLS, but the value is similar to that of the WLS this time.
- The t-values are once again very similar to the previous values.

```r
ci_rlm <- function(model, alpha = 0.05) {
  stopifnot(any(class(model) == "rlm"))

  n <- nrow(model$model)
  p <- ncol(model$model)
  t_crit <- qt(1 - alpha / 2, n - p)
  summary_model <- summary(model)
  b <- drop(summary_model$coefficients[, "Value"])
  stderr <- drop(summary(model)$coefficients[, "Std. Error"])
  interval_lower <- b - t_crit * stderr
  interval_upper <- b + t_crit * stderr
  intervals <- t(rbind(interval_lower, interval_upper))
  colnames(intervals) <- c(
```

```
    paste(100 * alpha / 2, "%"), paste(100 * (1 - alpha / 2), "%")
  )
  intervals
}

intervals_rlm <- ci_rlm(model_rlm)
```

The confidence intervals for the linear impact of the input variables are shown below:

$$\beta_1 : (0.1421795, 0.3029988)$$
$$\beta_2 : (-0.0030293, 0.0569018)$$
$$\beta_3 : (0.0788468, 0.5861864)$$

Problem 5

Apply the bootstrapping method to compute the confidence interval on the parameters for WLS, OLS, and Robust.

OLS

```
boot_ols <- function(data, indices) {
  data <- data[indices, ]
  model_ols <- lm(y ~ x1 + x2 + x3, data)
  coef(model_ols)
}

boot_ols <- boot(data = df_le, statistic = boot_ols, R = 100)
boot_ols
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = df_le, statistic = boot_ols, R = 100)
##
##
## Bootstrap Statistics :
##        original       bias    std. error
## t1* 64.47981912 -0.049262816  0.68052751
## t2*  0.23418111  0.000290287  0.03497864
## t3*  0.02574424 -0.001386736  0.01386636
## t4*  0.32115885  0.033561810  0.12592158
```

```
ci_x0 <- boot.ci(boot_ols, index = 1, type = "perc")$percent[4:5]
ci_x1 <- boot.ci(boot_ols, index = 2, type = "perc")$percent[4:5]
ci_x2 <- boot.ci(boot_ols, index = 3, type = "perc")$percent[4:5]
ci_x3 <- boot.ci(boot_ols, index = 4, type = "perc")$percent[4:5]
```

The confidence intervals on the parameters are:

$$\beta_0 : (62.918788, 65.7968086)$$
$$\beta_1 : (0.1673779, 0.3158283)$$
$$\beta_2 : (0.0011887, 0.0566325)$$
$$\beta_3 : (0.1063926, 0.6100799)$$

WLS

```
df_le <- data.frame(y, x1, x2, x3)
library(boot)

boot_wls <- function(data, indices) {
  data <- data[indices, ]
  model <- lm(y ~ x1 + x2 + x3, data)
  w <- 1 / fitted(lm(abs(residuals(model)) ~ x1 + x2 + x3, data))^2
  model_wls <- lm(y ~ x1 + x2 + x3, weight = w, data = data)
  coef(model_wls)
}

boot_wls <- boot(data = df_le, statistic = boot_wls, R = 100)
boot_wls
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = df_le, statistic = boot_wls, R = 100)
##
##
## Bootstrap Statistics :
##        original        bias    std. error
## t1* 64.98955905 -5.589554e-03  0.93637487
## t2*  0.21095579  1.270746e-03  0.04039445
## t3*  0.02537906 -6.420023e-05  0.01492929
## t4*  0.31598273  1.004626e-02  0.12376398
```

```
ci_x0 <- boot.ci(boot_wls, index = 1, type = "perc")$percent[4:5]
ci_x1 <- boot.ci(boot_wls, index = 2, type = "perc")$percent[4:5]
ci_x2 <- boot.ci(boot_wls, index = 3, type = "perc")$percent[4:5]
ci_x3 <- boot.ci(boot_wls, index = 4, type = "perc")$percent[4:5]
```

The confidence intervals on the parameters are:

$$\beta_0 : (63.3687445, 66.7579941)$$
$$\beta_1 : (0.1344525, 0.309106)$$
$$\beta_2 : (-0.0030499, 0.0567738)$$
$$\beta_3 : (0.0594685, 0.5805309)$$

Robust

```
library(boot)
boot_huber <- function(data, indices, maxit = 100) {
```

```
  data <- data[indices, ]
  model_rlm <- rlm(y ~ x1 + x2 + x3, data = data, maxit = maxit)
  coef(model_rlm)
}

boot_rlm <- boot(
  data = df_le,
  statistic = boot_huber,
  R = 100,
  maxit = 100,
)
boot_rlm
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = df_le, statistic = boot_huber, R = 100, maxit = 100)
##
##
## Bootstrap Statistics :
##        original       bias    std. error
## t1* 64.74190978 -0.000293399  0.79142203
## t2*  0.22362915 -0.002520783  0.03637239
## t3*  0.02708278  0.001678024  0.01403086
## t4*  0.32803737  0.010746651  0.11007538
```

```
ci_x0 <- boot.ci(boot_rlm, index = 1, type = "perc")$percent[4:5]
ci_x1 <- boot.ci(boot_rlm, index = 2, type = "perc")$percent[4:5]
ci_x2 <- boot.ci(boot_rlm, index = 3, type = "perc")$percent[4:5]
ci_x3 <- boot.ci(boot_rlm, index = 4, type = "perc")$percent[4:5]
```

The confidence intervals on the parameters are:

$$\beta_0 : (63.4229007, 66.5147048)$$
$$\beta_1 : (0.1373512, 0.2909788)$$
$$\beta_2 : (-8.634462 \times 10^{-5}, 0.0580282)$$
$$\beta_3 : (0.1177623, 0.5802277)$$

Problem 6

Note that the lm.ridge() algorithm turns out to be indeterminate when VIF is close to 1, resulting in a flat ridge trace line. On the other hand, the `lmridge()` function only reports $\lambda = 0$ (i.e., OLS). For practicing the `lm.ridge()` and `lmridge()` functions, this question will be to use the body fat example and complete the confidence interval for X_1, X_2, and X_3 in the ridge model.

We can first read in the dataset and model it using `MASS::lm.ridge()`, then show the ridge trace plot.

```
fat <- read.csv("../../datasets/bodyfat.csv")
colnames(fat) <- c("x1", "x2", "x3", "y")
model_ridge_mass <- lm.ridge(
  y ~ x1 + x2 + x3,
  data = fat, lambda = seq(0, 1, 0.01)
```

```
)
plot(model_ridge_mass)
```

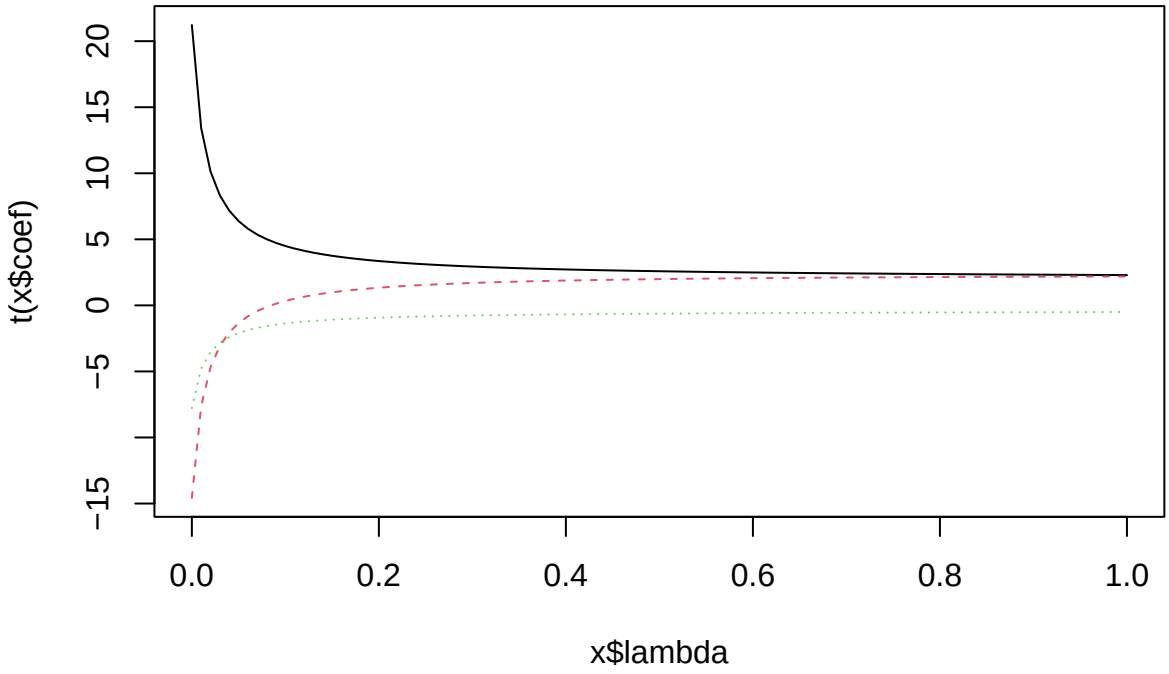

Figure 2: Ridge Trace Plot from MASS Package

Next, we can select the λ value that has the smallest GCV value.

```
select(model_ridge_mass)
```

```
## modified HKB estimator is 0.008505093
## modified L-W estimator is 0.3098511
## smallest value of GCV  at 0.02
```

```
lambda_opt <- 0.02
```

The optimal λ value which minimizes the Generalized Cross Validation equation is $\lambda = 0.02$.

Next, we can model the data using `lmridge::lmridge()` and show the ridge trace plot.

```
model_lmridge <- lmridge::lmridge(
  y ~ x1 + x2 + x3,
  data = fat, K = seq(0, 1, 0.01)
)
plot(model_lmridge)
```

We observe the same plot as before, just with different axis limits. Next, we observe the VIF values for the first six values of λ.

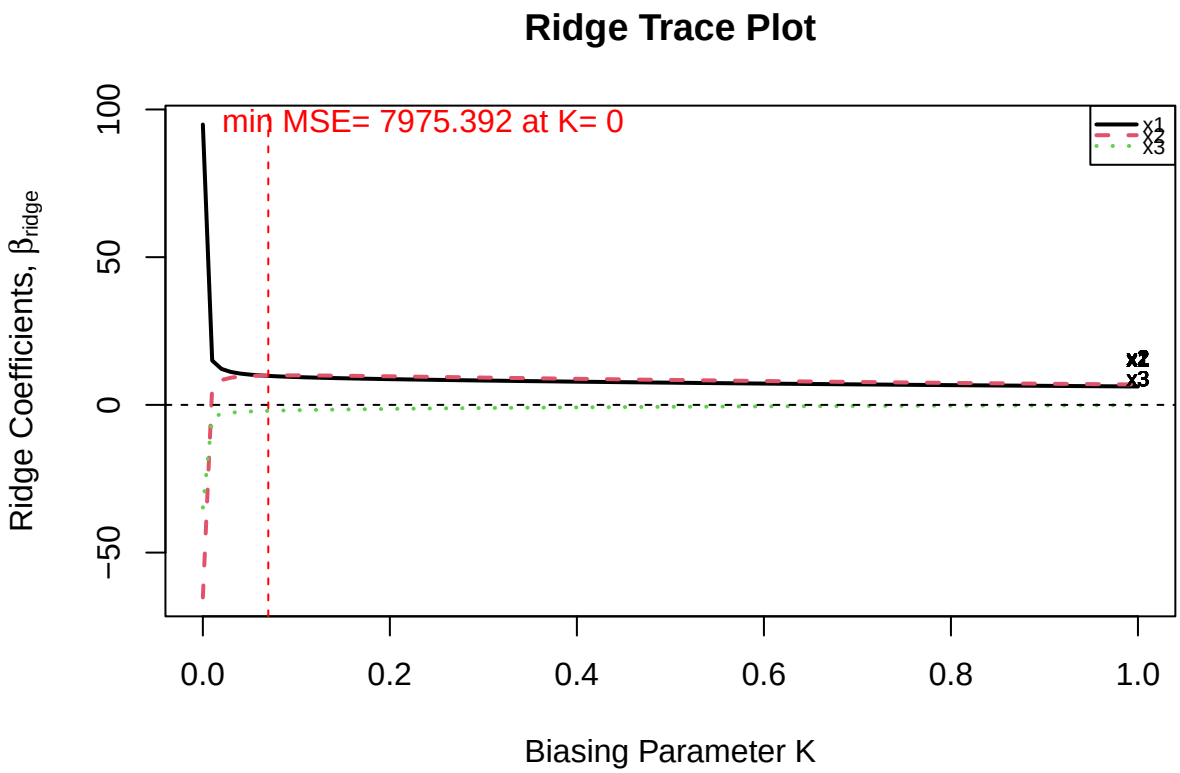

Figure 3: Ridge Trace Plot from lmridge Package

```r
head(lmridge::vif(model_lmridge))
```

```
##               x1        x2        x3
## k=0    708.84291 564.34339 104.60601
## k=0.01   3.48550   2.98127   1.37703
## k=0.02   1.10255   1.08054   1.01051
## k=0.03   0.62570   0.69691   0.92346
## k=0.04   0.45279   0.55529   0.88140
## k=0.05   0.37045   0.48588   0.85311
```

The VIF are closest to 1 for all parameters when $\lambda = k = 0.02$.

Lastly, we observe the model summary for the optimal λ value:

```r
ridge_summary <- summary(lmridge::lmridge(
  y ~ x1 + x2 + x3,
  data = fat, K = lambda_opt
))
ridge_summary
```

```
##
## Call:
## lmridge.default(formula = y ~ x1 + x2 + x3, data = fat, K = lambda_opt)
##
##
## Coefficients: for Ridge parameter K= 0.02
##             Estimate Estimate (Sc) StdErr (Sc) t-value (Sc) Pr(>|t|)
## Intercept   -7.4034     -633.1991    161.1205      -3.9300   0.0011 **
## x1           0.5554       12.1599      2.5781       4.7167   0.0002 ***
## x2           0.3681        8.4000      2.5522       3.2913   0.0043 **
## x3          -0.1916       -3.0464      2.4681      -1.2343   0.2339
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Ridge Summary
##       R2   adj-R2 DF ridge        F      AIC      BIC
##  0.76340  0.73560  2.00448 21.95136 37.75478 99.66535
## Ridge minimum MSE= 13285.04 at K= 0.02
## P-value for F-test ( 2.00448 , 17.93165 ) = 1.500203e-05
## ------------------------------------------------------------------------
```

We can use the values in this summary to calculate the confidence interval for the parameters:

```r
estimates <- ridge_summary$summaries$`summary  1`$coefficients[
  c("x1", "x2", "x3"), "Estimate (Sc)"
]

stderrs <- ridge_summary$summaries$`summary  1`$coefficients[
  c("x1", "x2", "x3"), "StdErr (Sc)"
]

df_ridge <- ridge_summary$summaries$`summary  1`$df1[, "df ridge"]
alpha <- 0.05
n <- nrow(fat)

interval_lower <- estimates - qt(1 - alpha / 2, n - df_ridge) * stderrs
```

```
interval_upper <- estimates + qt(1 - alpha / 2, n - df_ridge) * stderrs
```

The intervals are as follows:

For X_1:

$$b_{1,SC} \pm t(1-\alpha/2, n - df_{\text{ridge}}) s_{1,SC}$$
$$12.1599431 \pm t(0.975, 20 - 2.00448) 2.5780704$$
$$(6.7435216, 17.5763646)$$

Where b_{SC}, s_{SC} refer to the scaled estimate for the parameter β_1 and the scaled standard error of the parameter.

For X_2:

$$b_{2,SC} \pm t(1-\alpha/2, n - df_{\text{ridge}}) s_{2,SC}$$
$$8.4000042 \pm t(0.975, 20 - 2.00448) 2.5522077$$
$$(3.0379191, 13.7620892)$$

For X_3:

$$b_{3,SC} \pm t(1-\alpha/3, n - df_{\text{ridge}}) s_{3,SC}$$
$$-3.0463974 \pm t(0.975, 20 - 2.00448) 2.4681213$$
$$(-8.2318204, 2.1390256)$$