Compulsory exercise 1: Group 18

TMA4268 Statistical Learning V2023

Chester Henry Charlton, Kasper Eikeland, Tinus Garshol

2023-02-23

```r
library("knitr")
library("ggplot2")
library("dplyr")
library("tidyr")
library("carData")
library("class")
library("pROC")
library("plotROC")
library("ggmosaic")
library("readr")
library("ggfortify")
library("MASS")
knitr::opts_chunk$set(echo=TRUE, eval=TRUE, message=FALSE, warning = FALSE,
                      strip.white = TRUE, prompt = FALSE, cache = TRUE,
                      size = "scriptsize", fig.width = 4, fig.height = 3,
                      fig.align = "center")
```

Problem 1

Part A

We calculate the expected value of $\tilde{\boldsymbol{\beta}}$ as follows:

$$\begin{aligned}
\mathrm{E}(\tilde{\boldsymbol{\beta}}) &= \mathrm{E}[(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{Y}] \\
&= (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathrm{E}(\mathbf{Y}) \\
\mathbf{Y} &= \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \\
\mathrm{E}(\mathbf{Y}) &= \mathrm{E}(\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}) \\
&= \mathbf{X}\boldsymbol{\beta} \\
\mathrm{E}(\tilde{\boldsymbol{\beta}}) &= (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta}
\end{aligned}$$

Furthermore, we calculate the variance-covariance of $\tilde{\beta}$ as follows: We have that $\mathrm{Var}(\mathbf{Y}) = \mathrm{Var}(\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}) = \mathrm{Var}(\boldsymbol{\varepsilon}) = \sigma^2$. Defining $\mathbf{W} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}$ we compute the variance-covariance matrix $\mathrm{Var}(\tilde{\boldsymbol{\beta}})$ as

$$\begin{aligned}
\mathrm{Var}(\tilde{\boldsymbol{\beta}}) &= \mathrm{Var}(\mathbf{W}\mathbf{X}^\top\mathbf{Y}) \\
&= \mathbf{W}\mathbf{X}^\top\mathrm{Var}(\mathbf{Y})(\mathbf{W}\mathbf{X}^\top)^\top \\
&= \sigma^2\mathbf{W}\mathbf{X}^\top\mathbf{X}\mathbf{W}
\end{aligned}$$

since $\mathbf{W}$ is symmetric.

Part B

Given $\tilde{f}(\mathbf{x_0}) = \mathbf{x_0}^\top \tilde{\boldsymbol{\beta}}$ we have that

$$\begin{aligned}
\mathrm{E}(\tilde{f}(\mathbf{x_0})) &= \mathbf{x_0}^\top \mathrm{E}(\tilde{\boldsymbol{\beta}}) \\
&= \mathbf{x_0}^\top (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{X} \boldsymbol{\beta}
\end{aligned}$$

and

$$\begin{aligned}
\mathrm{Var}(\tilde{f}(\mathbf{x_0})) &= \mathbf{x_0}^\top \mathrm{Var}(\tilde{\boldsymbol{\beta}}) \mathbf{x_0} \\
&= \mathbf{x_0}^\top \sigma^2 \mathbf{W} \mathbf{X}^\top \mathbf{X} \mathbf{W} \mathbf{x_0}
\end{aligned}$$

Part C

Bias is interpreted as follows: Whenever we fit a model to data, it can either be underfit or overfit in relation to to the true form of where the data is originating. In this sense, bias occurs when we underfit, using a simpler model such as linear regression to approximate a complex system in reality.

Variance is closely related to this, occurring most characteristically when we overfit our model. This could occur if the true form of the system was quadratic but we fit a higher order polynomial in an attempt to match the data. This results in a model that fits to too much of the "noise" in the data and thus does not fare well on the test set. Another way of thinking of it is the degree to which our model "varies" depending on small changes in the training data. This is leads to the correct conclusion that more complex models have more variance than simpler models.

Irreducible error is the "noise" in our data that results from the fact that our sample data is being drawn from some distribution. This means that we expect even a "perfect" model to have test error since with noise in the data, the random error cannot and should not be fitted to. Otherwise, the model will be overfitted.

Part D

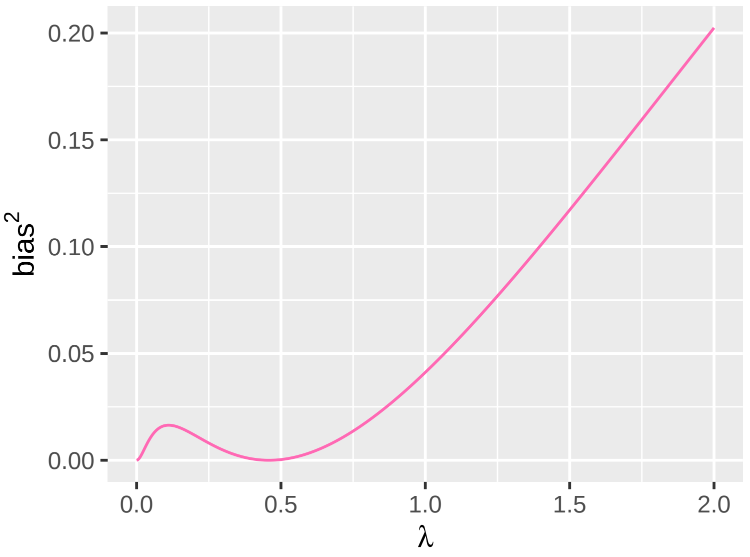
We can calculate the mean squared error of a particular observation by decomposing it as the sum of the variance, squared bias, and standard deviation at a particular point. We already have expressions for the variance from part b and the standard deviation is known. Therefore, we need only derive the expression for the squared bias at a particular point, and then substitute.

$$\begin{aligned}
\mathrm{Bias}^2 &= \mathrm{E}(y_0 - \tilde{f}(\mathbf{x}_0)) \\
&= \mathbf{x}_0^\top \boldsymbol{\beta} - \mathbf{x}_0^\top \mathrm{E}(\boldsymbol{\beta}) \\
\mathrm{MSE}(\mathbf{Y} - \hat{\mathbf{Y}}) &= \mathrm{Var}(\hat{\mathbf{Y}}) + \mathrm{Bias}^2(\hat{\mathbf{Y}}) + \mathbf{Var}(\boldsymbol{\varepsilon}) \\
&= \mathrm{Var}(\mathbf{x}_0^\top \tilde{\boldsymbol{\beta}}) + \mathrm{Bias}^2(\mathbf{x}_0^\top \tilde{\boldsymbol{\beta}}) + \mathrm{Var}(\boldsymbol{\varepsilon}) \\
&= \mathbf{x_0}^\top \sigma^2 \mathbf{W} \mathbf{X}^\top \mathbf{X} \mathbf{W} \mathbf{x_0} + \mathrm{Bias}^2(\mathbf{x}_0^\top \tilde{\boldsymbol{\beta}}) + \sigma^2
\end{aligned}$$

Part E

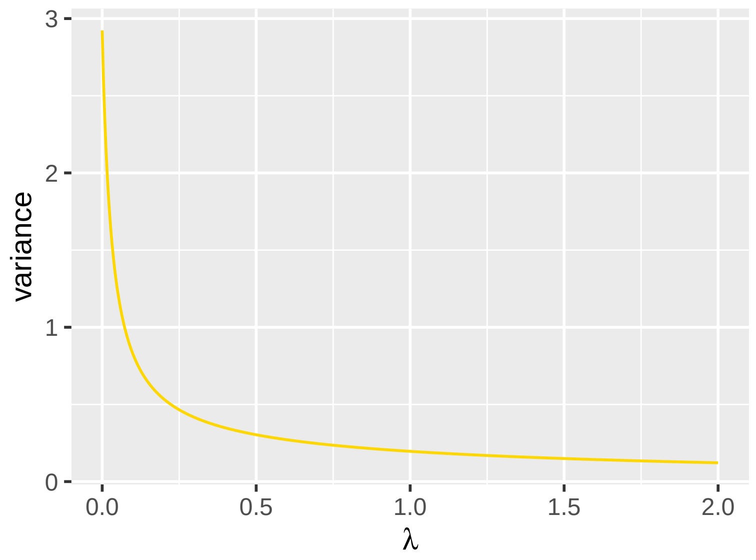
```r
id <- "1X_8OKcoYbng1XvYFDirxjEWr7LtpNr1m" # google file ID
values <- dget(sprintf("https://docs.google.com/uc?id=%s&export=download", id))
X <- values$X
x0 <- values$x0
beta <- values$beta
sigma <- values$sigma
```

```r
bias <- function(lambda, X, x0, beta) {
  p <- ncol(X)
  value <- (t(x0) %*% beta - t(x0) %*%
              (solve(t(X)%*%X + lambda * diag(p)) %*% (t(X) %*% X) %*% beta))**2
  return(value)
}
lambdas <- seq(0, 2, length.out = 500)
BIAS <- rep(NA, length(lambdas))
for (i in seq_along(lambdas)) BIAS[i] <- bias(lambdas[i], X, x0, beta)
dfBias <- data.frame(lambdas = lambdas, bias = BIAS)
ggplot(dfBias, aes(x = lambdas, y = bias)) +
  geom_line(color = "hotpink") +
  xlab(expression(lambda)) +
  ylab(expression(bias^2))
```



Part F

```r
variance <- function(lambda, X, x0, sigma) {
  p <- ncol(X)
  inv <- solve(t(X) %*% X + lambda * diag(p))
  W <- solve((t(X) %*% X) + (lambda * diag(p))) %*% t(X)
  value <- (sigma**2)* t(x0) %*% (W) %*% t(W) %*% (x0)
  return(value)
}
lambdas <- seq(0, 2, length.out = 500)
VAR <- rep(NA, length(lambdas))
for (i in seq_along(lambdas)) VAR[i] <- variance(lambdas[i], X, x0, sigma)
dfVar <- data.frame(lambdas = lambdas, var = VAR)
ggplot(dfVar, aes(x = lambdas, y = var)) +
  geom_line(color = "gold") +
  xlab(expression(lambda)) +
  ylab("variance")
```

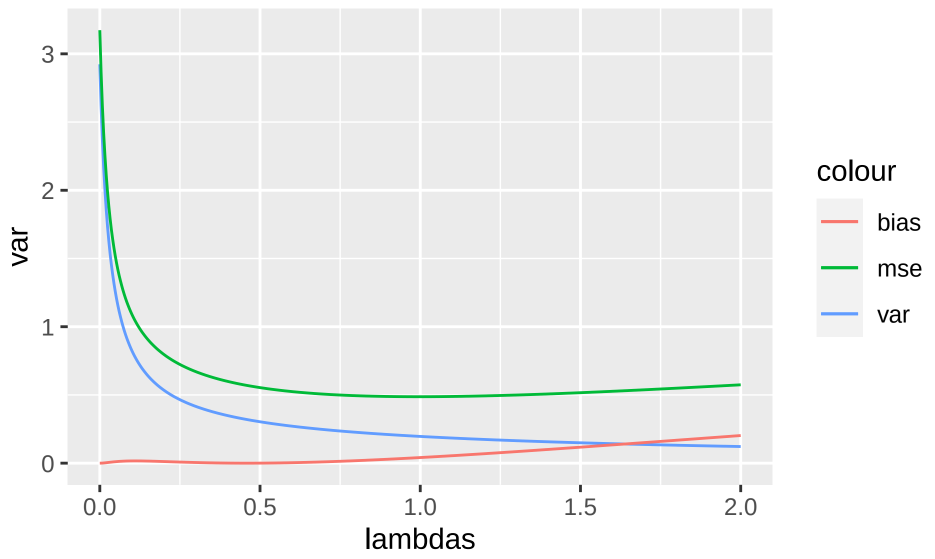


Part G

```
exp_mse <- VAR + BIAS + sigma**2
lambdas[which.min(exp_mse)]
```

```
## [1] 0.993988
```

```
dfMSE <- data.frame(lambdas = lambdas, MSE = exp_mse)
dfVarBiasMSE <- cbind(dfVar,dfBias$bias, dfMSE$MSE)

ggplot(dfVarBiasMSE, aes(x=lambdas)) +
         geom_line(aes(y = var, color = "var")) +
         geom_line(aes(y = dfBias$bias, color = "bias")) +
         geom_line(aes(y = dfMSE$MSE, color = "mse"))
```



Problem 2

```r
library(carData)
```

```r
# Fit full model
model1 <- lm(salary ~ ., data = Salaries)
summary(model1)
```

```
##
## Call:
## lm(formula = salary ~ ., data = Salaries)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -65248 -13211  -1775  10384  99592
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    65955.2     4588.6  14.374  < 2e-16 ***
## rankAssocProf  12907.6     4145.3   3.114  0.00198 **
## rankProf       45066.0     4237.5  10.635  < 2e-16 ***
## disciplineB    14417.6     2342.9   6.154 1.88e-09 ***
## yrs.since.phd    535.1      241.0   2.220  0.02698 *
## yrs.service     -489.5      211.9  -2.310  0.02143 *
## sexMale         4783.5     3858.7   1.240  0.21584
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22540 on 390 degrees of freedom
## Multiple R-squared:  0.4547, Adjusted R-squared:  0.4463
## F-statistic:  54.2 on 6 and 390 DF,  p-value: < 2.2e-16
```

Part A

i. The lm() function encodes for these categorical variables by operating under the assumption that each instance of the covariate "rank" must be one of the following classes: AssocProf, AsstProf, and Prof. Then it assigns a two binary variables to two of those classes, let us say AssocProf and Prof. This designates AsstProf as the "reference" class, to which the predictions about AsstProf and Prof are made relatively to.

ii. Analysis of variance

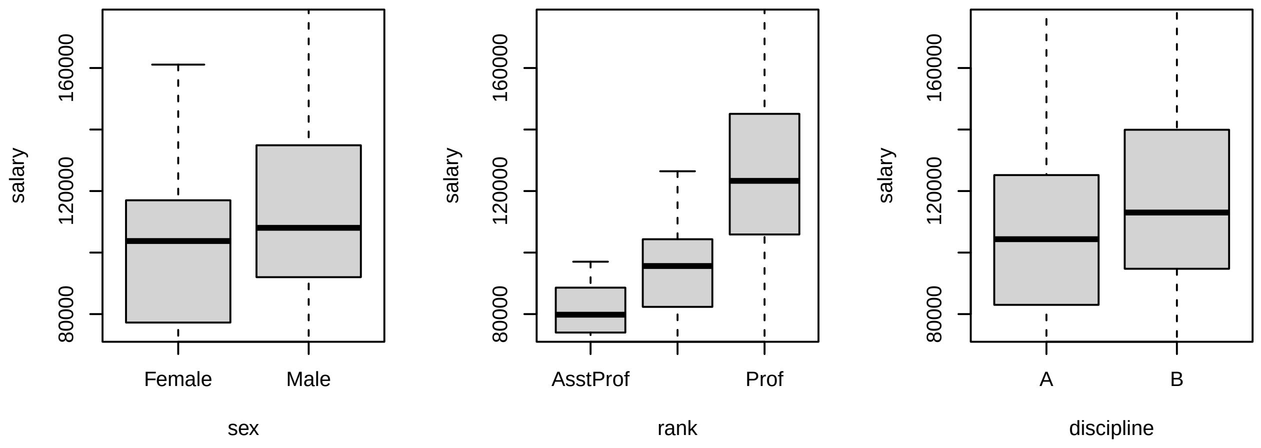
```r
anova(model1)[1,]
```

```
## Analysis of Variance Table
##
## Response: salary
##       Df     Sum Sq    Mean Sq F value    Pr(>F)
## rank  2 1.4323e+11 7.1616e+10  140.98 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Through an analysis of variance it is clear that the p-value for the F-statistic shows a significant variance between salary and rank, showing there is a strong correlation between the two.

Part B

```r
par(mfrow=c(1,3))    # set the plotting area into a 1*3 array
boxplot(salary~sex,data=Salaries,ylim=c(75000,175000))
boxplot(salary~rank,data=Salaries,ylim=c(75000,175000))
boxplot(salary~discipline,data=Salaries,ylim=c(75000,175000))
```



The linear regression model considers correlation between parameters based on which covariates are included. When looking at sex_model, there is a noticeable difference between the salaries for males and females, suggesting that gender makes a sufficient impact on salary. However, when compared to other covariates in model1, such as rank and discipline, are much more important factors in determining salary.

Part C

i. For robust linear regression, the following assumptions must be satisfied:

1. $E(\epsilon_i) = 0$
2. $Var(\epsilon_i) = \sigma^2$
3. $\epsilon_i \sim N(0, \sigma^2)$
4. ϵ_i independent of ϵ_k for all $i \neq k$

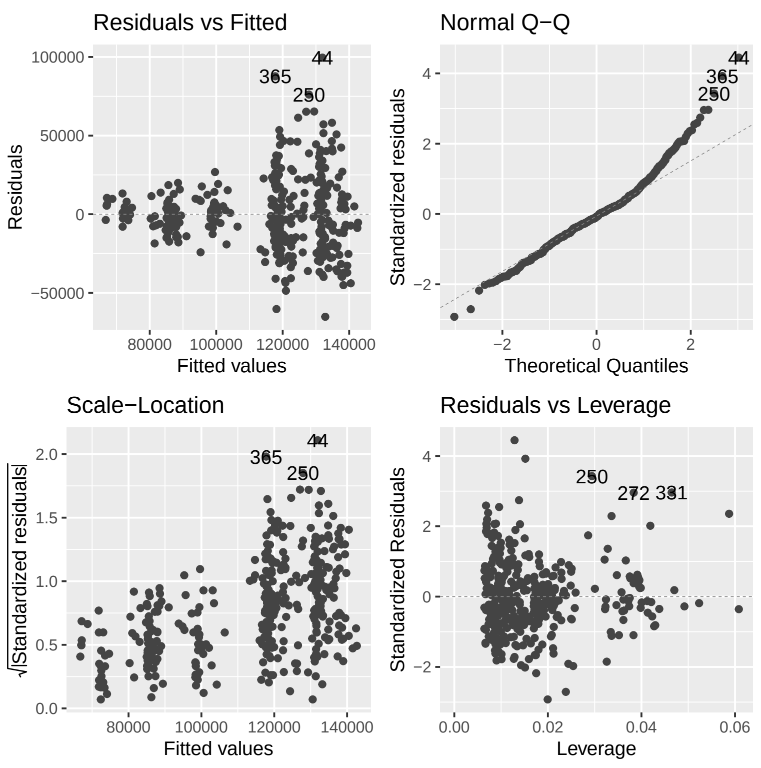
Residuals vs Fitted Plot: we can see that assumption 1 is satisfied: Since the residuals are approximately centered at 0, we can safely assumed that their expected value is also 0.

Normal Q-Q Plot: The normal Q-Q plot looks okay, maybe with a bit of deviation on the right tail. Overall, we would say that this plot indicates that assumption 3 has been satisfied decently.

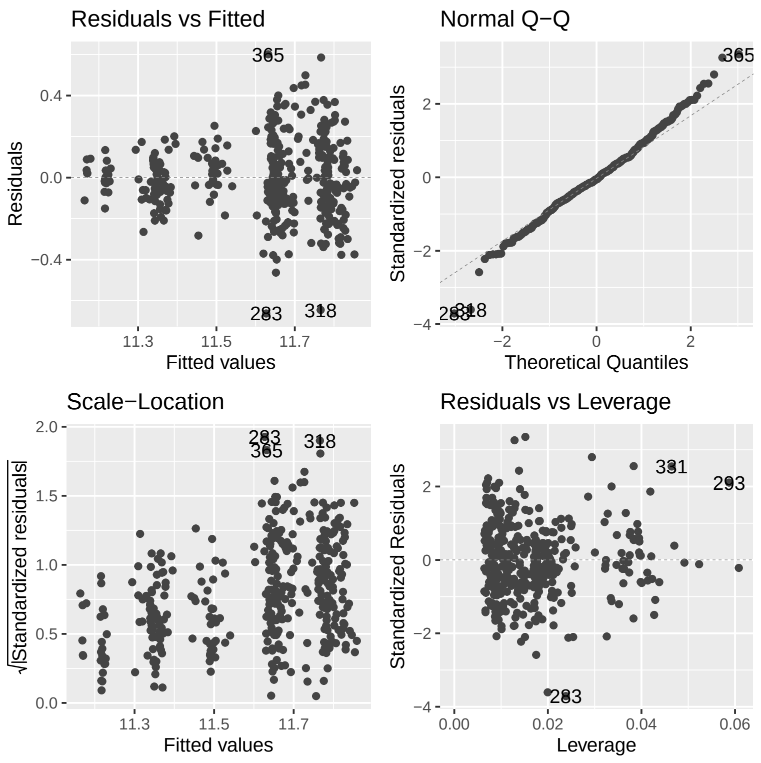
Scale-Location Plot: We are looking to see if there is any trend, and there does appear that there is a positive trend. So we conclude that the errors are not homoscedastic, violating one of our assumptions. Residuals vs Leverage: There are no points of extreme leverage to note. We would be looking for points in the top right or bottom right corners.

ii. By taking the log of the response variable (salary), we have improved the Q-Q plot a bit, but dissapointingly have not improved the scale-location plot, which was the main issue.

```r
Salaries$log_salary <- log(Salaries$salary)
model2 <- lm(log_salary ~ ., data = Salaries[,-6])
autoplot(model1, smooth.colour = NA)
```



```
autoplot(model2, smooth.colour = NA)
```



Part D

i. Here we implement a model with an interaction term between sex and yrs.since.phd

```
model3 <- lm(log_salary ~ . + sex*yrs.since.phd, data = Salaries[,-6])
```

ii. As we can see, Bernie's hypothesis is not correct. Adding an interaction between sex and yrs.since.phd does not improve the significance of the correlation between log_salary and sex.

```
summary(model2)
```

```
##
## Call:
## lm(formula = log_salary ~ ., data = Salaries[, -6])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.66236 -0.10813 -0.00914  0.09804  0.60107
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   11.164144   0.036794 303.425  < 2e-16 ***
## rankAssocProf  0.153787   0.033239   4.627 5.06e-06 ***
## rankProf       0.449463   0.033979  13.228  < 2e-16 ***
## disciplineB    0.131869   0.018786   7.019 9.94e-12 ***
## yrs.since.phd  0.003289   0.001932   1.702   0.0896 .
## yrs.service   -0.003918   0.001699  -2.305   0.0217 *
## sexMale        0.045583   0.030941   1.473   0.1415
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1807 on 390 degrees of freedom
## Multiple R-squared:  0.5248, Adjusted R-squared:  0.5175
## F-statistic: 71.79 on 6 and 390 DF,  p-value: < 2.2e-16
```

```
summary(model3)
```

```
##
## Call:
## lm(formula = log_salary ~ . + sex * yrs.since.phd, data = Salaries[,
##     -6])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.66187 -0.10831 -0.00951  0.09846  0.60143
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            11.1537511  0.0591759 188.485  < 2e-16 ***
## rankAssocProf           0.1528200  0.0335575   4.554 7.05e-06 ***
## rankProf                0.4482679  0.0344343  13.018  < 2e-16 ***
## disciplineB             0.1317818  0.0188133   7.005 1.09e-11 ***
## yrs.since.phd           0.0039500  0.0035253   1.120   0.2632
## yrs.service            -0.0038902  0.0017059  -2.280   0.0231 *
## sexMale                 0.0574914  0.0614436   0.936   0.3500
## yrs.since.phd:sexMale -0.0007049  0.0031407  -0.224   0.8225
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.1809 on 389 degrees of freedom
## Multiple R-squared:  0.5249, Adjusted R-squared:  0.5163
## F-statistic: 61.39 on 7 and 389 DF,  p-value: < 2.2e-16
```

Part E

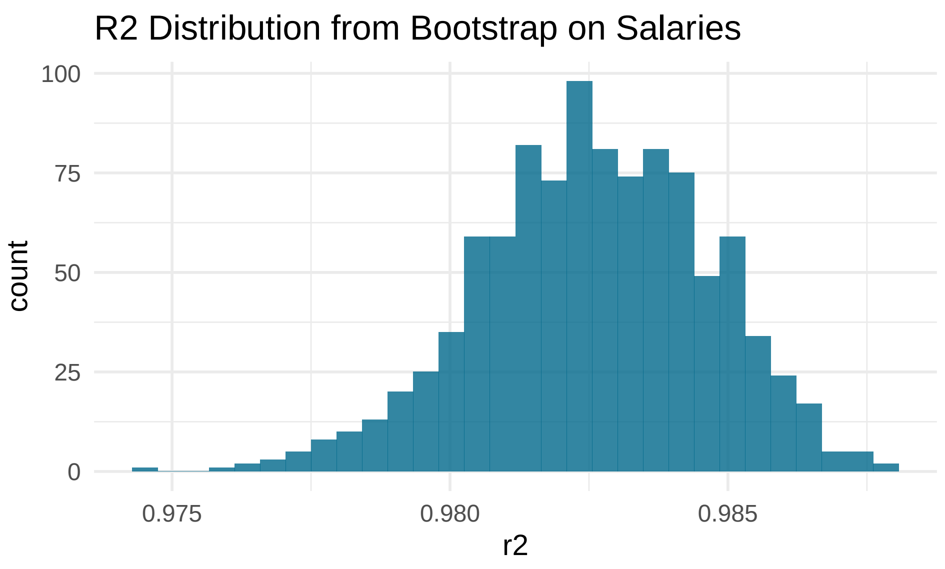
i. Here we will implement bootstrapping to get information about our uncertainty of the R^2 value in our model.

```
set.seed(4268)

# boot-strap from Salaries dataset
N = 1000
estimator = rep(NA, N)
for (i in 1:N) {
    boot_data_i <- sample_n(Salaries, length(Salaries$salary), replace = TRUE)
    model_i <- lm(salary ~ ., data = boot_data_i)
    estimator[i] = summary(model_i)$r.squared
}
sd(estimator)
```

```
## [1] 0.002063122
```

ii. Here we plot the distribution of R^2 values obtained through the bootstrap.

```
ggplot(data.frame(r2=estimator), aes(x=r2)) +
  geom_histogram(fill='deepskyblue4',alpha=0.8) +
  ggtitle("R2 Distribution from Bootstrap on Salaries") +
  theme_minimal()
```



iii. The standard error (empirical standard deviation) is ≈ 0.00206. Furthermore, the 95 percent quantile interval is approximately from 0.978 to 0.986, as computed below.

```
sd(estimator)
```

```
## [1] 0.002063122
```

```
quantile_vector <- c(0.025,0.975)
quantile(estimator,quantile_vector)
```

```
##      2.5%     97.5%
## 0.9781084 0.9863881
```

iv. This analysis bodes very well for the degree to which our model can predict the data. Even on the low end of the 95 percent quantile interval, our model captures 97.8 percent of the variation in the salary variable, which is quite good.

Part F

i. Below is Bert-Ernie's code, corrected. We have changed the interval to a prediction interval since we are only predicted one value. We have changed the confidence level to 0.9 in order to achieve a 95% one-sided confidence interval. Note that with the corrected code, Bert-Ernie can still be confident about his salary regardless of which option he chooses.

```
# Make a data frame containing two new observations, corresponding to
# Bert-Ernie's two possible futures
bert_ernie <- data.frame(rank = c("Prof", "Prof"),
                         discipline = c("A", "B"), # Theoretical, applied
                         yrs.since.phd = c(20, 20),
                         yrs.service = c(20, 20),
                         sex = c("Male", "Male"))
# Use the full model to predict his salary
preds <- predict(object = model1,
                 newdata = bert_ernie,
                 interval = 'prediction',
                 level = 0.9) # Not 0.95 since we don't care about upper limit
# Check predictions
preds
```

```
##        fit      lwr      upr
## 1 116715.6 79318.04 154113.1
## 2 131133.2 93792.87 168473.5
```

```
# Check if lower limit for salary in a theoretical field is large enough
preds[1, 2] > 75000
```

```
## [1] TRUE
```

ii. Analytical expression for the two sided, 90% prediction interval:

$$\left[\boldsymbol{x}_0^T\hat{\boldsymbol{\beta}} - t_{n-p}(1-0.1/2)\hat{\sigma}\sqrt{1+\boldsymbol{x}_0^T(X^TX)^{-1}\boldsymbol{x}_0}, \boldsymbol{x}_0^T\hat{\boldsymbol{\beta}} + t_{n-p}(1-0.1/2)\hat{\sigma}\sqrt{1+\boldsymbol{x}_0^T(X^TX)^{-1}\boldsymbol{x}_0}\right]$$

Now we can implement this interval in R, noting that we only take the lower bound for the Bert-Ernie's purposes. The result is identical to the one we received using R-functions.

```
X = model.matrix(model1)
beta_hat = coef(model1)
sigma_hat = sigma(model1)
x0 = c(1,0,1,0,20,20,1)
tval = qt(c(0.05,0.95),df=390)

PI_l = t(x0)%*%beta_hat + tval[1]*sigma_hat* sqrt(1 + t(x0)%*%solve(t(X)%*%X)%*% x0)
```

```
PI_u = t(x0)%*%beta_hat + tval[2]*sigma_hat* sqrt(1 + t(x0)%*%solve(t(X)%*%X)%*% x0)

PI = c(PI_l,PI_u)
PI
```

```
## [1]  79318.04 154113.12
```

We notice the analytical results yields the same PI.

Problem 3

Here we download and clean the data, as provided in the problem statement:

```
bigfoot_original <- readr::read_csv("https://raw.githubusercontent.com/rfordatascience/tidytuesday/maste

# Prepare the data:
bigfoot <- bigfoot_original %>%
  # Select the relevant covariates:
  dplyr::select(classification, observed, longitude, latitude, visibility) %>%
  # Remove observations of class C (these are second- or third hand accounts):
  dplyr::filter(classification != "Class C") %>%
  # Turn into 0/1, 1 = Class A, 0 = Class B:
  dplyr::mutate(class = ifelse(classification == "Class A", 1, 0)) %>%
  # Create new indicator variables for some words from the description:
  dplyr::mutate(fur = grepl("fur", observed),
                howl = grepl("howl", observed),
                saw = grepl("saw", observed),
                heard = grepl("heard", observed)) %>%
  # Remove unnecessary variables:
  dplyr::select(-c("classification", "observed")) %>%
  # Remove any rows that contain missing values:
  tidyr::drop_na()
```

Here we separate the data into training and test data, using the same seed as in the problem statement:

```
set.seed(2023)

# 70% of the sample size for training set
training_set_size <- floor(0.7 * nrow(bigfoot))

train_ind <- sample(seq_len(nrow(bigfoot)), size = training_set_size)

train <- bigfoot[train_ind, ]
test <- bigfoot[-train_ind, ]
```

Part A

i. Here we fit a logistic model.

```
bigfoot_logistic <- glm(class ~ ., data = train, family = binomial)

bigfoot_logistic_test_probs <- predict(bigfoot_logistic, newdata= test, type = "response")
bigfoot_logistic_pred <- rep("0", length(test$class))
bigfoot_logistic_pred[bigfoot_logistic_test_probs > .5] <- "1"
```

Note that 441 observations were classified in the “A” category. 299 of these were correct identifications.

ii. We multiply the odds by 3.6396744 (option 4)

Part B

i. The following code implements a QDA model:

```r
bigfoot_qda <- qda(class ~ ., data = train)
bigfoot_qda_test_probs <- predict(bigfoot_qda, test)
bigfoot_qda_pred <- predict(bigfoot_qda, test)$class
```

We note that 626 observations in the test set were identified as class "A", of which 389 were correct identifications.

ii.

1. True: The main difference between QDA and LDA is that QDA assumes that each class has its own covariance matrix.
2. False: LDA generally leads to low variance, high bias since it is less flexible than QDA. Thus we would prefer it for when there are fewer training observations so that we can reduce the variance of our model. [review logic]
3. False: If the Bayes decision boundary is linear, we would always prefer LDA
4. False: QDA does not have observations with a common mean vector.

Part C

i. The code below implements KNN classification for k=25.

```r
bigfoot_knn_pred <- knn(train=train, test=test, cl=train$class, k = 25, prob = TRUE)
```

ii. We may employ CV to choose K and test the predictive power of different values for K. As K increases, the bias increases, and variance decreases. As K decreases, the variance increases and bias decreases, resulting in a highly nonlinear decision border.

Part D

i. In this case, we are concerned with prediction, since our main goal is to develop a model so that we can assess whether future observations are truly those of bigfoot given their features. None of the models we have created are irreconcilable with this goal. However, for inference we generally want models that are more interpretable and allow us to more clearly see which predictors are associated with the response. In this sense, we would prefer linear models such as LDA, since the goal is understanding of the relation. Also, for inference we would probably want to avoid KNN, since it is very uninterpretable. For prediction, the opposite is true. We want to maximize how accurately we can predict the response, even if that sacrifices interpretability. So for that we would choose more complex models such as QDA or KNN. However, I would not say that this excludes any models, supposing that they can give good predictions.

ii. Sensitivity refers to the probability (based on the test set error) that an observation is marked as positive, given that it is actually positive. So in this case it can be stated as "the probability that we predict an observation is of bigfoot, given that it truly is." Specificity refers to the probability (based on the test set) that an observation is marked as negative, given that it is actually negative. So in this case it can be stated as "the probability that we predict an observation is not of bigfoot, given that it truly is not." For both of these statistics, we prefer them to be as close to 1 as possible. Below are the confusion matrices and their accompanying code.

```r
logistic_confusion <- table(bigfoot_logistic_pred, test$class)
rownames(logistic_confusion) <- c('pred B','pred A')
colnames(logistic_confusion) <- c('true B','true A')
logistic_sensitivity <- logistic_confusion[2,2] / sum(logistic_confusion[,2])
logistic_specificity <- logistic_confusion[1,1] / sum(logistic_confusion[,1])
logistic_vals = c(logistic_sensitivity, logistic_specificity)

qda_confusion <- table(bigfoot_qda_pred, test$class)
rownames(qda_confusion) <- c('pred B','pred A')
colnames(qda_confusion) <- c('true B','true A')
qda_sensitivity <- qda_confusion[2,2] / sum(qda_confusion[,2])
qda_specificity <- qda_confusion[1,1] / sum(qda_confusion[,1])
qda_vals = c(qda_sensitivity, qda_specificity)

knn_confusion <- table(bigfoot_knn_pred, test$class)
rownames(knn_confusion) <- c('pred B','pred A')
colnames(knn_confusion) <- c('true B','true A')
knn_sensitivity <- knn_confusion[2,2] / sum(knn_confusion[,2])
knn_specificity <- knn_confusion[1,1] / sum(knn_confusion[,1])
knn_vals = c(knn_sensitivity, knn_specificity)

names = c("Sensitivity", "Specificity")
df_ss = data.frame(names, logistic = logistic_vals, QDA = qda_vals, KNN = knn_vals)

logistic_confusion
```

```
##
## bigfoot_logistic_pred true B true A
##                pred B    323    148
##                pred A    142    299
```

```r
qda_confusion
```

```
##
## bigfoot_qda_pred true B true A
##           pred B    228     58
##           pred A    237    389
```

```r
knn_confusion
```

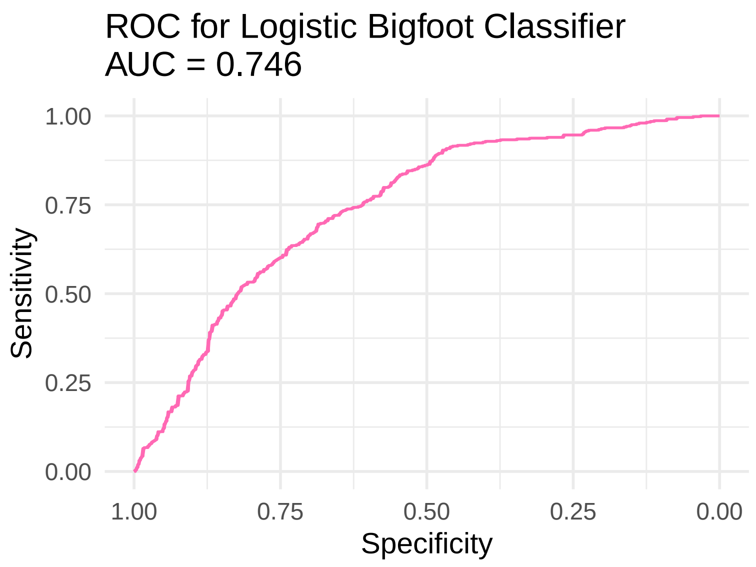
```
##
## bigfoot_knn_pred true B true A
##           pred B    386     85
##           pred A     79    362
```

```r
df_ss
```

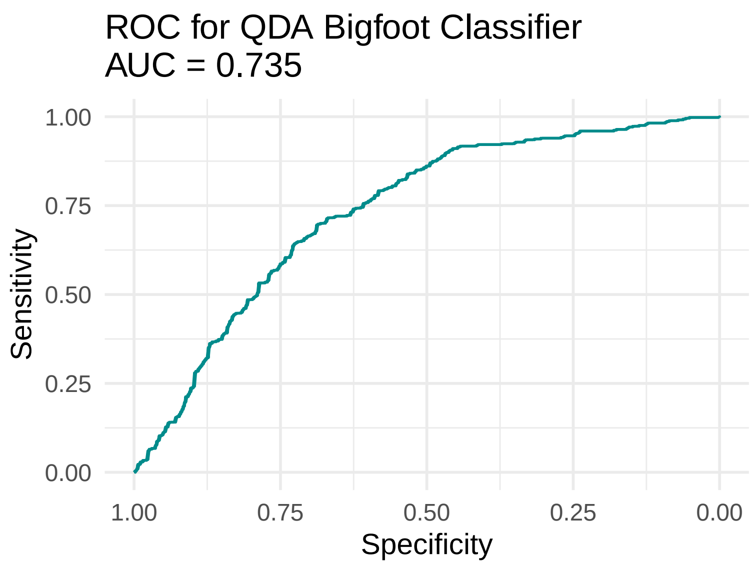
```
##         names  logistic       QDA       KNN
## 1 Sensitivity 0.6689038 0.8702461 0.8098434
## 2 Specificity 0.6946237 0.4903226 0.8301075
```

iii. Here we present the ROC plots for three models:

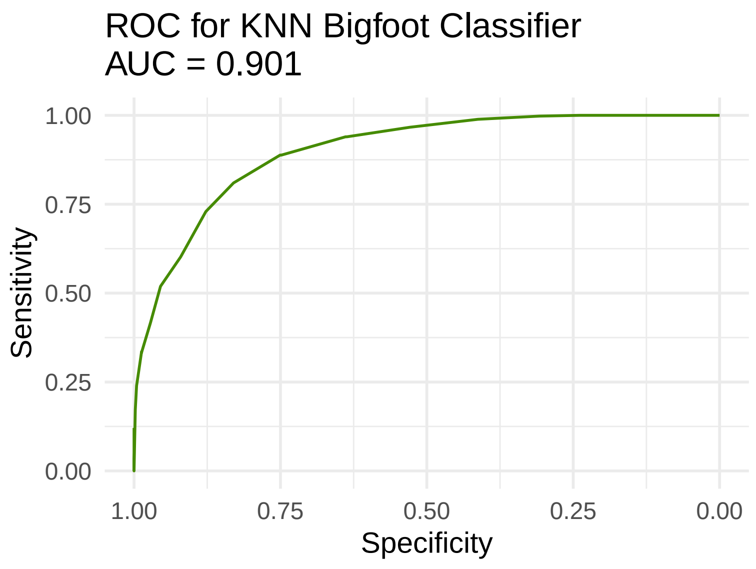
```r
bigfoot_logistic_roc <- roc(test$class ~ bigfoot_logistic_test_probs,
                            plot = FALSE, print.auc = FALSE)
bigfoot_qda_roc <- roc(test$class ~ bigfoot_qda_test_probs$posterior[,1],
                        plot = FALSE, print.auc = FALSE)

probKNN <- ifelse(bigfoot_knn_pred == 0,
                  1 - attributes(bigfoot_knn_pred)$prob,
                  attributes(bigfoot_knn_pred)$prob)
bigfoot_knn_roc <- roc(test$class ~ probKNN, plot = FALSE, print.auc = FALSE)


# make df's for each model's ROC data:
logistic_roc_df <- data.frame(specificities = bigfoot_logistic_roc$specificities, sensitivities =
                                bigfoot_logistic_roc$sensitivities)

qda_roc_df <- data.frame(specificities = bigfoot_qda_roc$specificities, sensitivities = bigfoot_qda_roc$

knn_roc_df <- data.frame(specificities = bigfoot_knn_roc$specificities, sensitivities = bigfoot_knn_roc$



# plot each model's ROC data, including AUC:
ggplot(logistic_roc_df, aes(x = specificities, y = sensitivities)) +
  geom_line(color = "hotpink") +
  ggtitle(paste("ROC for Logistic Bigfoot Classifier \nAUC =", round(bigfoot_logistic_roc$auc,3))) +
  scale_x_reverse() +
  labs(x = "Specificity", y = "Sensitivity") +
  theme_minimal()
```



```
ggplot(qda_roc_df, aes(x = specificities, y = sensitivities)) +
  geom_line(color = "darkcyan") +
  ggtitle(paste("ROC for QDA Bigfoot Classifier \nAUC =", round(bigfoot_qda_roc$auc,3))) +
  scale_x_reverse() +
  labs(x = "Specificity", y = "Sensitivity") +
  theme_minimal()
```



```
ggplot(knn_roc_df, aes(x = specificities, y = sensitivities)) +
  geom_line(color = "chartreuse4") +
  ggtitle(paste("ROC for KNN Bigfoot Classifier \nAUC =", round(bigfoot_knn_roc$auc,3))) +
  scale_x_reverse() +
  labs(x = "Specificity", y = "Sensitivity") +
  theme_minimal()
```



iv. For a classification model such as this, we want the highest possible sensitivity, specificity, and AUC. Based on these measures, we can see that the logistic model is strictly worse than QDA and KNN, with lower sensitivity and specificity. QDA and KNN are closer in their sensitivity and specificity metrics, but we can also use the AUC. In this sense, KNN (AUC = 0.901) is the clear winner, since it has a drastically higher AUC than QDA (AUC = 0.735). What this means is that when we change the classification threshold to get a higher sensitivity, we sacrifice less specificity in our KNN model than we do in our QDA model. In conclusion we would choose the KNN model, especially considering the fact that our value for k was set arbitrarily and could be optimized to get an even better AUC, sensitivity and specificity.

Problem 4

Part A

We want to calculate the error of our model when the i^{th} response is removed, i.e.

$$e_{(-i)} = y_i - \hat{y}_{(-i)} = y_i - x_i^T \hat{\beta}_{(-i)}$$

The estimator for beta with the i^{th} response removed is unknown, so we must find an expression for it. We start by noticing that

$$\hat{\beta} = (X^T X)^{-1} X^T Y \qquad \text{and}$$

$$\hat{\beta}_{(-i)} = (X_{(-i)}^T X_{(-i)})^{-1} X_{(-i)}^T y_{(-i)} \quad \text{(a)}$$

Since the estimator of beta without the i^{th} response is unknown, we need to express it in terms of known variables. First, using the Sherman-Morris formula, we find that:

$$\begin{aligned}
(X_{(-i)}^T X_{(-i)})^{-1} &= (X^T X - x_i x_i^T)^{-1} \\
&= (X^T X)^{-1} + \frac{(X^T X)^{-1} x_i x_i^T (X^T X)^{-1}}{1 - x_i^T (X^T X)^{-1} x_i} \\
&= (X^T X)^{-1} + \frac{(X^T X)^{-1} x_i x_i^T (X^T X)^{-1}}{1 - h_i} \quad \text{(b)}
\end{aligned}$$

It is also known that

$$X_{(-i)}^T y_{(-i)} = X^T Y - x_i y_i \quad \text{(c)}$$

With this we can substitute (b) and (c) into (a), giving:

$$\begin{aligned}
\hat{\beta}_{(-i)} &= \left[(X^T X)^{-1} + \frac{(X^T X)^{-1} x_i x_i^T (X^T X)^{-1}}{1 - h_i} \right] (X^T Y - x_i y_i) \\
&= \hat{\beta} - \frac{(X^T X)^{-1} x_i}{1 - hi} (y_i (1 - h_i) - x_i^T \hat{\beta} + h_i y_i) \\
&= \hat{\beta} - \frac{(X^T X)^{-1} x_i}{1 - hi} (y_i - x_i^T \hat{\beta}) \\
&= \hat{\beta} - \frac{(X^T X)^{-1} x_i}{1 - hi} (y_i - \hat{y}_i)
\end{aligned}$$

Which, when substituted into the error equation gives:

$$\begin{aligned}
y_i - \hat{y}_{(-i)} &= y_i - x_i^T \hat{\beta}_{(-i)} \\
&= y_i - x_i^T \left(\hat{\beta} - \frac{(X^T X)^{-1} x_i}{1 - hi} (y_i - \hat{y}_i) \right) \\
&= y_i - \hat{y}_i + \frac{h_i}{1 - h_i} (y_i - \hat{y}_i) \\
&= \frac{y_i - \hat{y}_i}{1 - h_i}
\end{aligned}$$

Finally, the LOOCV is calculated by the RSS of the error $e_{(-i)}$, so:

$$\text{CV} = \frac{1}{N} \sum_{i=1}^{N} e_{(-i)}^2 = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{y_i - \hat{y}_i}{1 - h_i} \right)^2$$

Part B

i. False: LOOCV results in very low bias but high variance, since it has more nearly the maximum number of observations in its training set (n-1). The bias of K-fold CV will converge (in a decreasing manner) to the bias of LOOCV as k approaches n,

ii. False: Polynomial regression is explicitly one of the conditions for which the formula in part A is valid (see p.202 ISLR)

iii. True: The formula from part A is valid for linear regression (least squares) and transforming the response variable does compromise this.

iv. False: In the validation approach, we train on the training set and then test on the test set, which is equivalent to the first step of the 2-fold CV approach. In the 2-fold CV approach we train on the training set and then test on the test set, but then we switch these sets and train on what was previously the test set and test on what was previously the training set.