Compulsory exercise 1: Group 11

TMA4268 Statistical Learning V2023

Torbjørn Vatne, Ludvik Braathen and Johan Bjerkem

17 February, 2023

```r
install.packages("knitr")     # probably already installed
install.packages("rmarkdown") # probably already installed
install.packages("ggplot2")   # plotting with ggplot2
install.packages("dplyr")     # for data cleaning and preparation
install.packages("tidyr")     # also data preparation
install.packages("carData")   # dataset
install.packages("class")     # for KNN
install.packages("pROC")      # calculate roc
install.packages("plotROC")   # plot roc
install.packages("ggmosaic")  # mosaic plot
install.packages("knitr")
install.packages("formatR")
library(knitr)
install.packages("tinytex")
tinytex::install_tinytex()
```

Problem 1 (9P)

a) (1P)

Here we find the expected value:

$$
\begin{aligned}
E(\widetilde{\boldsymbol{\beta}}) &= E((\mathbf{X}^\top\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{Y}) && (1)\\
&= (\mathbf{X}^\top\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^\top E(\mathbf{Y}) && (2)\\
&= (\mathbf{X}^\top\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^\top E(\mathbf{X}\boldsymbol{\beta}+\varepsilon) && (3)\\
&= (\mathbf{X}^\top\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^\top (\mathbf{X}\boldsymbol{\beta}+\mathbf{0}) && (4)\\
&= (\mathbf{X}^\top\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^\top \mathbf{X}\boldsymbol{\beta} && (5)
\end{aligned}
$$

Here we find the variance-covarince matrix:

$$
\begin{aligned}
Cov(\widetilde{\boldsymbol{\beta}}) &= Cov((X^TX+\lambda I)^{-1}X^TY) && (6)\\
&= (X^TX+\lambda I)^{-1}X^TCov(Y)((X^TX+\lambda I)^{-1}X^T)^T && (7)\\
&= (X^TX+\lambda I)^{-1}X^T\sigma^2I((X^TX+\lambda I)^{-1}X^T)^T && (8)
\end{aligned}
$$

b) (2P)

First we find the expected value:

$$E(\mathbf{x}_0^T\widetilde{\boldsymbol{\beta}}) = \mathbf{x}_0^T E(\widetilde{\boldsymbol{\beta}}) \tag{9}$$
$$= \mathbf{x}_0^T(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} \tag{10}$$

Then we find the variance: (Here we need to remember the rule for variance where we need to multiply the same term transposed at the back of the variance term.)

$$Var(\mathbf{x}_0^T\widetilde{\boldsymbol{\beta}}) = \mathbf{x}_0^T Var(\widetilde{\boldsymbol{\beta}})\mathbf{x}_0 \tag{11}$$
$$= \mathbf{x}_0^T Var(\widetilde{\boldsymbol{\beta}})\mathbf{x}_0 \tag{12}$$
$$= \mathbf{x}_0^T(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\sigma^2((\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top)^\top\mathbf{x}_0 \tag{13}$$

c) (1P)

Explain with words how we can interpret the three terms: bias, variance and irreducible error.

Bias is the error that comes by oversimplifying the model. This means underfitting the model, meaning high bias will over simplify the relationships in the data. Variance, on the other hand, is the error that comes by creating a too complex mode. This means overfitting the model, where the model tries to hard to fit the function to the noise of the data. There is a famous plot of the relationship between bias and variance. When we train a model we start with a high bias that decays as we train the model. The variance starts low and increases as we train the model. We want to hit the sweet spot where bias and variance intercepts, as this is the point with the minimal total error. Irreducible error is the error that comes with the data. As the name suggests this error cannot be eliminated by any model, as it is inherit by the data and is not due to bias or variance.

d) (2P)

$$E[(y_0 - \widetilde{f}(\mathbf{x}_0))^2] = Var(\widetilde{f}(\mathbf{x}_0)) + (Bias(\widetilde{f}(\mathbf{x}_0)))^2 + Var(\varepsilon) \tag{14}$$

$$Var(\widetilde{f}(\mathbf{x}_0)) = \mathbf{x}_0^T(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\sigma^2((\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top)^\top\mathbf{x}_0 \tag{15}$$

$$(Bias(\widetilde{f}(\mathbf{x}_0)))^2 = (E(\mathbf{y}_0 - \widetilde{f}(\mathbf{x}_0)))^2 \tag{16}$$
$$(Bias(\widetilde{f}(\mathbf{x}_0)))^2 = (E(\mathbf{y}_0) - E(\widetilde{f}(\mathbf{x}_0)))^2 \tag{17}$$
$$(Bias(\widetilde{f}(\mathbf{x}_0)))^2 = (\mathbf{x}_0^T\boldsymbol{\beta} - \mathbf{x}_0^T(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta})^2 \tag{18}$$

$$Var(\varepsilon) = \sigma^2 \tag{19}$$

$$\begin{aligned} E[(y_0 - \widetilde{f}(\mathbf{x}_0))^2] = {} & \mathbf{x}_0^T(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\sigma^2((\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top)^\top\mathbf{x}_0 + \\ & (\mathbf{x}_0^T\boldsymbol{\beta} - \mathbf{x}_0^T(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta})^2 + \sigma^2 \end{aligned} \tag{20}$$

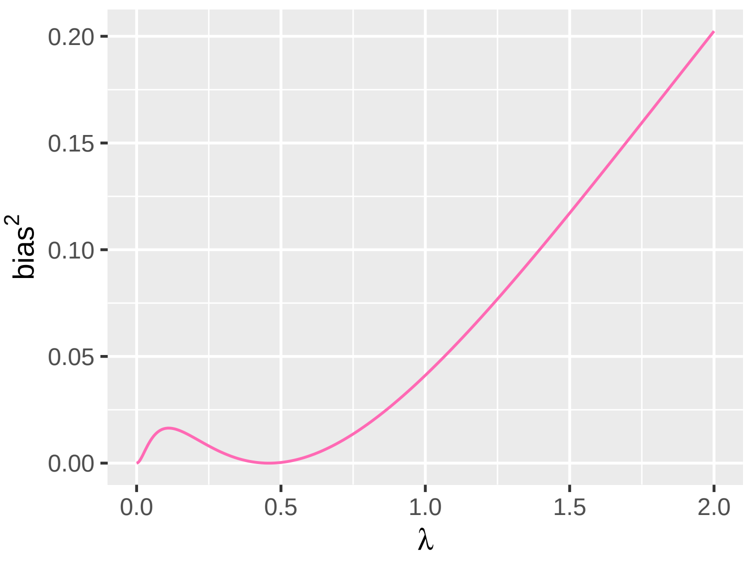
Plotting the bias-variance trade-off

The estimator $\widetilde{\beta}$ is a function of the tuning parameter λ, which controls the bias-variance trade-off. Using the decomposition derived in c) we will plot the three elements (bias, variance and irreducible error) using the values in the code chunk below. `values` is a list with the design matrix `X`, the vector $\mathbf{x}_0$ as `x0`, the parameters vector `beta` and the irreducible error `sigma`.

```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
```

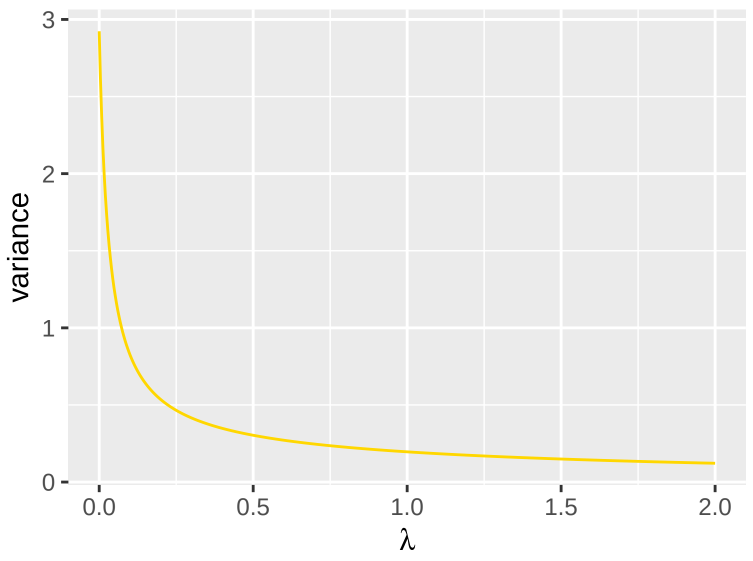
e) (1P)

```r
library(ggplot2)
bias <- function(lambda, X, x0, beta) {
  p <- ncol(X)
  inv <- solve(t(X) %*% X + lambda * diag(p))
  value <- (t(x0)%*%beta - t(x0)%*%inv%*%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))
```



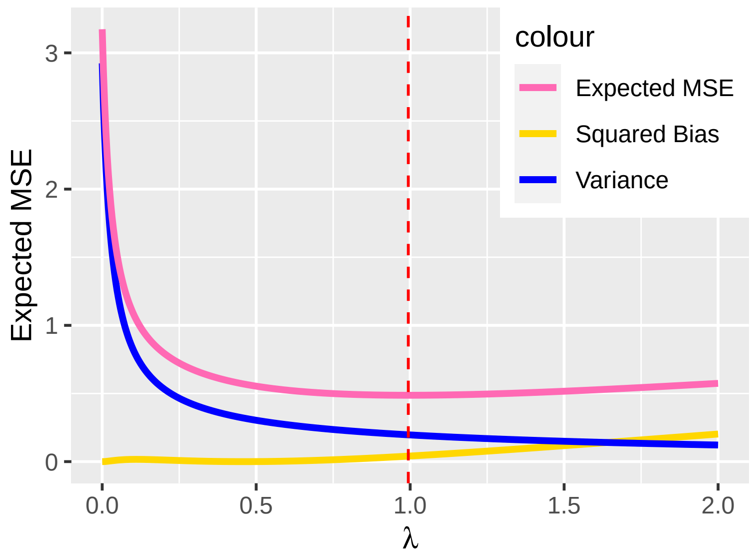
f) (1P)

```
variance <- function(lambda, X, x0, sigma) {
  p <- ncol(X)
  inv <- solve(t(X) %*% X + lambda * diag(p))
  value <- sigma^2*t(x0)%*%inv%*%t(X) %*% t(inv%*%t(X))%*%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")
```



g) (1P)

```
exp_mse <- BIAS + VAR + sigma^2
best_lambda <- lambdas[which.min(exp_mse)]
df <- merge(dfBias, dfVar, by = "lambdas")
df$exp_mse <- exp_mse
ggplot(df, aes(x = lambdas)) +
  geom_line(aes(y = bias, color = "Squared Bias"), linewidth = 1.2) +
  geom_line(aes(y = var, color = "Variance"), linewidth = 1.2) +
  geom_line(aes(y = exp_mse, color = "Expected MSE"), linewidth = 1.2) +
  geom_vline(xintercept = best_lambda, color = "red", linetype = "dashed") +
  scale_color_manual(values = c("hotpink", "gold", "blue")) +
  xlab(expression(lambda)) +
  ylab("Expected MSE") +
  theme(legend.position = c(1, 1), legend.justification = c(1, 1))
```



Problem 2 (15P)

```
model1 <- lm(salary ~ rank + discipline + yrs.since.phd + yrs.service + sex, data = Salaries)
```

a) Categorical variables (2P)

i) The $lm()$ function by default uses the first of the three levels (of the covariate rank), `AsstProf`, as a reference level, and then converts the other levels, `AssocProf` and `Prof`, to a set of 'dummy variables'. These dummy variables are represented as binary variables. Naturally, only one of the levels can be 'active' at once, as a professor can only be either of the three different types of professors. The estimated coefficient for associate professor represents the difference in mean salary between associate professors and assistant professors (reference level), while the estimated coefficient for a full professor represents the difference in mean salary between full professors and assistant professors. In other words, associate professors are expected to earn at average 12907 more than an assistant professor, and full professors are expected to earn at average 45066 more than an assistant professor. ***FIKS FORMATERING***

ii)

```
library(carData)
r.salary.rank <- lm(salary ~ yrs.service + yrs.since.phd + rank , data = Salaries)
r.salary.no.rank <- lm(salary ~ yrs.service + yrs.since.phd, data = Salaries)
anova(r.salary.rank, r.salary.no.rank)
```

```
## Analysis of Variance Table
##
## Model 1: salary ~ yrs.service + yrs.since.phd + rank
## Model 2: salary ~ yrs.service + yrs.since.phd
##   Res.Df        RSS Df  Sum of Sq      F    Pr(>F)
## 1    392 2.1839e+11
## 2    394 2.9487e+11 -2 -7.6488e+10 68.648 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

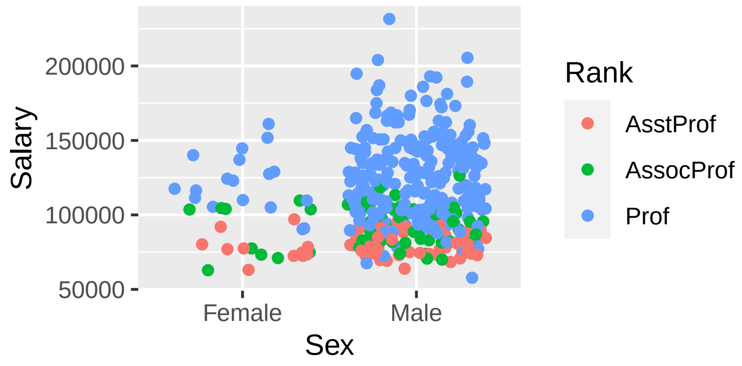
Here we've done a comparison of two models, Model 1 and Model 2. In Model 1 we've included the variables: `yrs.service`, `yrs.since.PhD`, and `rank`. In Model 2 we also include `yrs.service` and `yrs.since.PhD`, but excluded the `rank`-variable. Running *anova*() on these two models, gives us an F-value of 68.648 and a p-value of 2.2e-16. Determining which values of an F-test that are considered 'good' is not easy. In this case, with a relatively high F-value, together with a p-value which is practically 0, we consider that there is enough evidence that `rank` as a whole does have an impact on `salary`.

b) Conflicting model results? (2P)

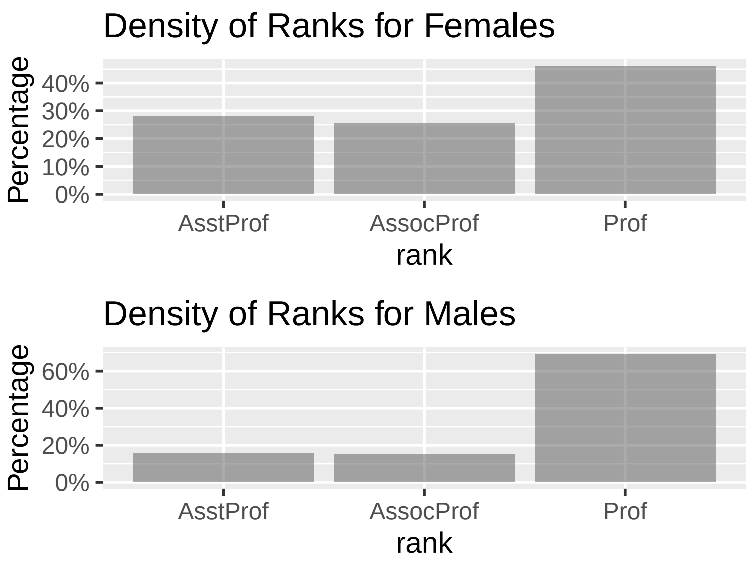
```r
library(ggplot2)
sex_model <- lm(salary ~ sex, data = Salaries)
coef(summary(sex_model))
```

```
##              Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 101002.41   4809.386 21.001103 2.683482e-66
## sexMale      14088.01   5064.579  2.781674 5.667107e-03
```

```r
ggplot(Salaries, aes(x = sex, y = salary, col = rank)) +
  geom_jitter() +
  labs(x = "Sex", y = "Salary", color = "Rank") +
  scale_x_discrete(labels = c("Female", "Male"))
```



```r
library(gridExtra)
p1 = ggplot(data = Salaries[Salaries$sex == 'Female',], aes(x = rank, fill = rank)) +
  geom_bar(position = "identity", alpha = 0.5, aes(y = ..prop.., group = 1)) +
  ggtitle("Density of Ranks for Females") +
  ylab("Percentage") +
  scale_y_continuous(labels = scales::percent)

p2 = ggplot(data = Salaries[Salaries$sex == 'Male',], aes(x = rank, fill = rank)) +
  geom_bar(position = "identity", alpha = 0.5, aes(y = ..prop.., group = 1)) +
  ggtitle("Density of Ranks for Males") +
  ylab("Percentage") +
  scale_y_continuous(labels = scales::percent)

grid.arrange(p1, p2, nrow = 2)
```



There can be two reasons:

1. There are relatively few observations on females compared to men, which makes the error variance higher, which then increases the p-value of the predictor. Had there been more female observations, it is likely that Std. error would decrease, which then would make the p-value more significant. With a sexMale parameter of 4783.5 it seems that sex correlates with salary. This is even more clear in the model with sex as the only predictor, where the p-value also is more significant.
2. Another possible reason why sex seems insignificant in model1 might be that there is another variable explaining the difference in salaries for the two sexes; Rank. There is a high correlation between Rank and salary, and looking at the barplots, we see that a high percentage of men are professors, however a much lower percentage of females are professors.

c) Model assumptions and transformation (2P)

```
library(ggplot2)
library(ggfortify)
autoplot(model1, smooth.colour = NA)
```

i) In the Residuals vs fitted plot, we can clearly see that the expected value of the residuals are zero, which fulfills the first assumption. However, the variance of the residuals is not constant over the fitted values, meaning there is no homoscedasticity. In other words, the residuals are not normally distributed with a constant variance. This can be seen in the Scale-Location and Normal Q-Q plots too. The assumption that there are no influentual outliers is unfulfilled as there are high-residual points with high leverage.

ii)

```
log_salary <- log(Salaries$salary)
model2 <- lm(log_salary ~ rank + discipline + yrs.since.phd + yrs.service + sex - salary, data = Salari
library(ggplot2)
autoplot(model2, smooth.colour = NA)
```

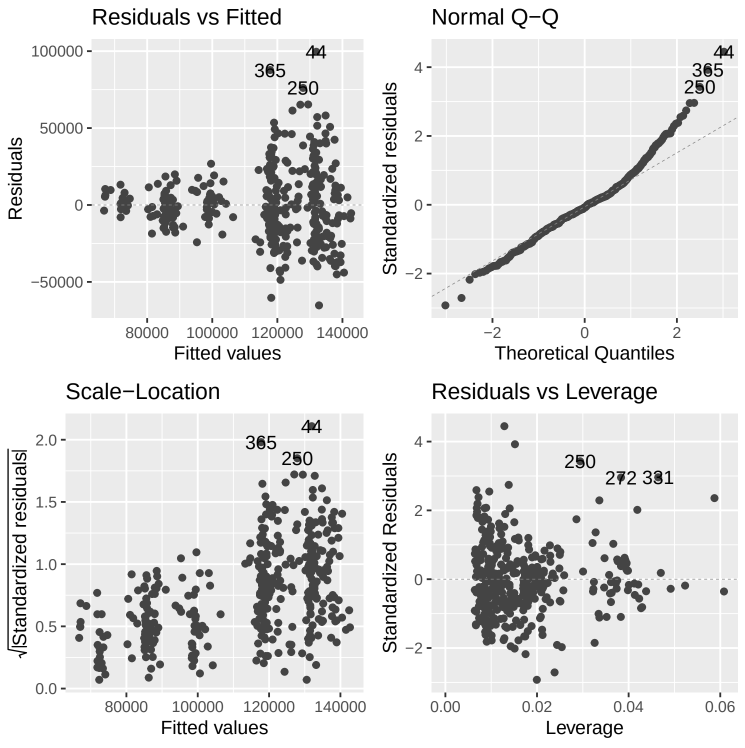


Figure 1: Diagnostic plots for ‘model1‘.

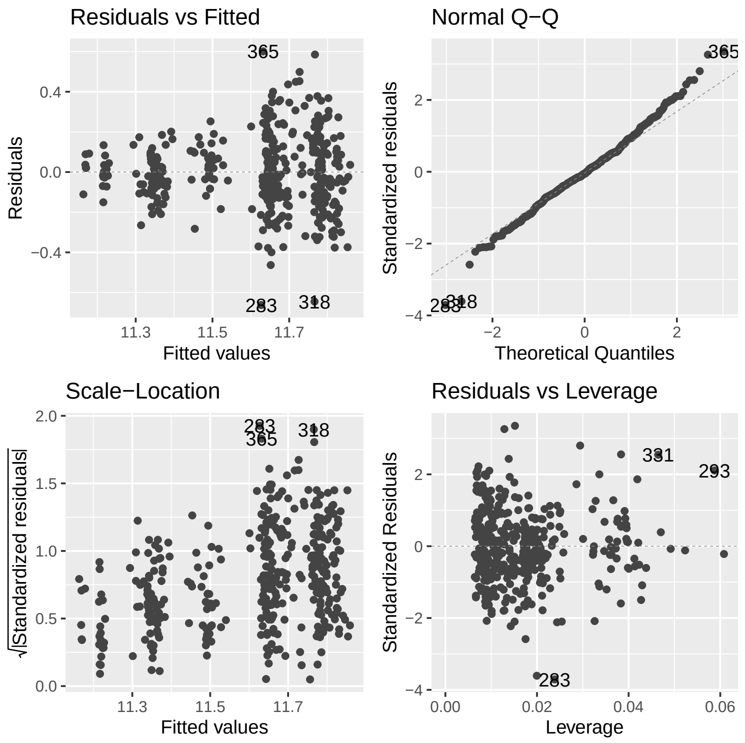


Figure 2: Diagnostic plots for ‘model2‘.

The residual variance has become somewhat more constant, but not much better. Influentual outliers does not seem to have improved.

d) Interactions (2P)

i)

```
model3 <- lm(log_salary ~ rank + discipline + yrs.since.phd + yrs.service + sex + sex:yrs.since.phd -sa
coef(summary(model3))[8,]
```

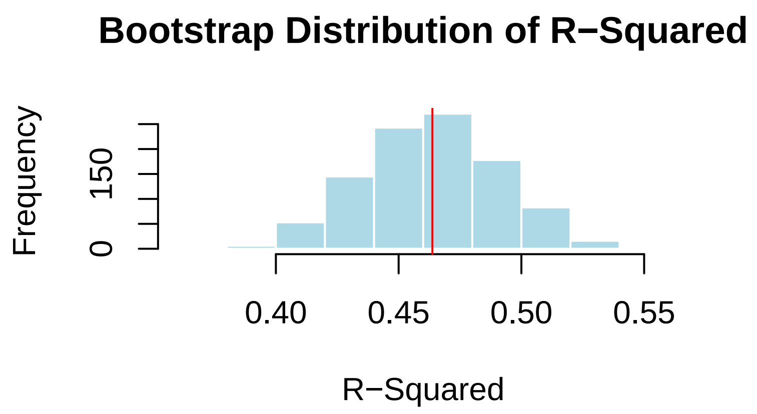
```
##      Estimate    Std. Error       t value      Pr(>|t|)
## -0.0007048561  0.0031406996 -0.2244264630  0.8225433134
```

ii) From the summary of the linear model, we can see that the p-value for the interaction term is very high, and we conclude with Bert-Ernie's hypothesis being incorrect.

e) Bootstrap (4P)

```
# (i)
set.seed(4268)
getR2 <- function(data, indices) {
  fit <- lm(salary ~ rank + discipline + yrs.since.phd + yrs.service + sex, data = data[indices,])
  summary(fit)$r.squared
}
library(boot)
boot_results <- boot(data = Salaries, statistic = getR2, R = 1000, strata = Salaries$rank)
```

```
# (ii)
hist(boot_results$t, main = "Bootstrap Distribution of R-Squared",
     xlab = "R-Squared", col = "lightblue", border = "white")
abline(v = mean(boot_results$t), col = "red")
```



```r
# (iii)
sd(boot_results$t)
```

```
## [1] 0.02803583
```

```r
quant = boot_results$t[25:975] # 95%  quantile, removing bottom and top 2.5%.
summary(quant)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3761  0.4440  0.4644  0.4637  0.4825  0.5470
```

(iv) Original R-squared was calculated to 0.5249 and bootstrap estimates 0.4546766 with a 95% confidence interval of [0.3761, 0.5470]. From this we see that the original R-squared calculated lies within the 95% confidence interval.

f) Picking a field (3P)

```r
# i)
bert_ernie <- data.frame(rank=c("Prof", "Prof"), discipline=c("A", "B"),
                         yrs.since.phd=c(20, 20), yrs.service=c(20, 20),
                         sex=c("Male", "Male"))
preds <- predict(object=model1, newdata=bert_ernie, interval="prediction", level=0.95)
# 1. Corrected confidence to prediction
# 2. Corrected 0.975 to 0.95
preds[1, 2] > 75000
```

```
## [1] FALSE
```

Many sleepless nights of debugging R code are looming in his future unfortunately.

ii) The analytic expression for the lower limit of the prediction interval:

$$PI_{lower} = \boldsymbol{x}_0^T \hat{\boldsymbol{\beta}} - t_{n-p}(1-\alpha/2)\hat{\sigma}\sqrt{1 + \boldsymbol{x}_0^T (X^T X)^{-1} \boldsymbol{x}_0}$$

```r
x_0 = c(1, 0, 1, 0, 20, 20, 1)
beta_hat = coef(model1)
alpha = 0.05
sd_hat = summary(model1)$sigma
X <- model.matrix(~rank+discipline+yrs.since.phd+yrs.service+sex, data=Salaries)
n = nrow(Salaries)
p = ncol(X)
PI_lower = t(x_0)%*%beta_hat - qt(1-alpha/2,df=n-p) * sd_hat *
  sqrt(1+t(x_0)%*%solve(t(X)%*%X)%*%x_0)
PI_lower
```

```
##          [,1]
## [1,] 72121.12
```

```
PI_lower == preds[1,2]
```

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

Problem 3 (13P)

```
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, ]
```

a) (2P)

```
glm.bigfoot = glm(class ~ ., data = train, family=binomial)
glm.probs <- predict(glm.bigfoot, test, type = "response")
glm.pred <- rep(0, 912)
glm.pred[glm.probs > 0.5] <- 1
table(glm.pred)
```

```
## glm.pred
##   0   1
## 471 441
```

```
print("441 sights classified as clear sightings.")
```

```
## [1] "441 sights classified as clear sightings."
```

(ii) (1P) Single choice:

4) We multiply by 3.641.

b) (3P)

```
require(MASS)
qda.fit <- qda(class ~ ., data = train)

qda.preds <- predict(qda.fit, test)
table(qda.preds$class)
```

```
##
##   0   1
## 286 626
```

```r
print("626 sights classified as clear sightings.")
```

```
## [1] "626 sights classified as clear sightings."
```

(ii) (2P)

True, False, False, False

c) (2P)

```r
require(class)
knnMod <- knn(train = train, test = test, cl = train$class, k = 25, prob = TRUE)
probKNN <- ifelse(knnMod == 0,
1 - attributes(knnMod)$prob,
attributes(knnMod)$prob)

table(knnMod)
```

```
## knnMod
##   0   1
## 471 441
```

```r
print("441 sights classified as clear sightings.")
```

```
## [1] "441 sights classified as clear sightings."
```

(ii) (1P)

Choosing an optimal value for k comes down to the bias-variance trade-off. A low k will lead to a flexible fit, for example for $k = 1$ all the model will interpolate all of the training data. Here the variance is high and the bias is low. Increasing k will increase to higher bias and lower variance. This means the outliers will effect the model less.

There are no predefined statisticial models for choosing k.I would create a plot between different k values and their errors. Based on this plot i would choose the k that leads to the lowest error.

d) (6P)

i) (2P)

Prediction. Given a new measurement, we want to use an existing data set to build a model that reliably chooses the correct classification. If we wanted to do inference we would be interested in the interpretability of the model. In the case of inference QDA and KNN would be less interesting models. As the goal is prediction all the models are interesting, but we would choose the model with the lowest error for our predicting (our a combination of the models that produces the best result).

ii) (3P)

Confusion matrix for logistic regression:

```
table(glm.pred, test$class)
```

```
##
## glm.pred   0   1
##        0 323 148
##        1 142 299
```

```
299/(299+148)
```

```
## [1] 0.6689038
```

```
323/(323+142)
```

```
## [1] 0.6946237
```

Sensitivity = 299 / (299 + 148) = 66.9% Specificity = 323 / (323 + 142) = 69.5%

Confusion matrix for QDA:

```
table(qda.preds$class, test$class)
```

```
##
##       0   1
##   0 228  58
##   1 237 389
```

```
389/(389+58)
```

```
## [1] 0.8702461
```

```
228/(228+237)
```

```
## [1] 0.4903226
```

Sensitivity = 389 / (389 + 58) = 87.0% Specificity = 228 / (228 + 237) = 49.0%

Confusion matrix for KNN:

```
table(knnMod, test$class)
```

```
##
## knnMod   0   1
##      0 386  85
##      1  79 362
```

```
362/(362+85)
```

```
## [1] 0.8098434
```

```r
386/(386+79)
```

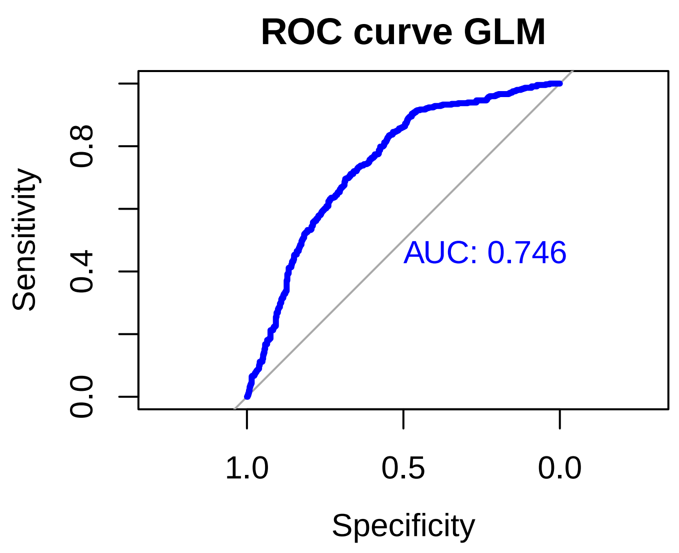
```
## [1] 0.8301075
```

Sensitivity = 362 / (362 + 85) = 81.0% Specificity = 386 / (386 + 79) = 83.0%

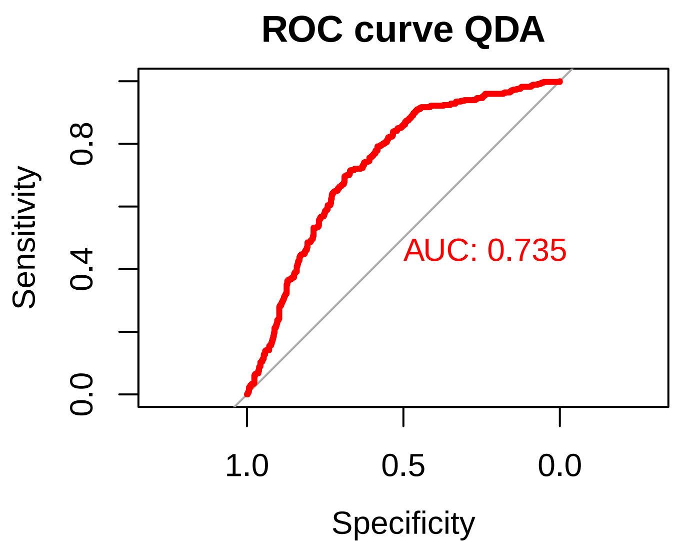
Sensitivity means how often is a true bigfoot sighting actually classified as Class A by the model. Specificity mean how often is a false bigfoot sighting actually classified as Class B by the model. Both are measures of accuracy. Good percentages for each score depends on the situation, there is no given standard that is universal.

iii) (1P).

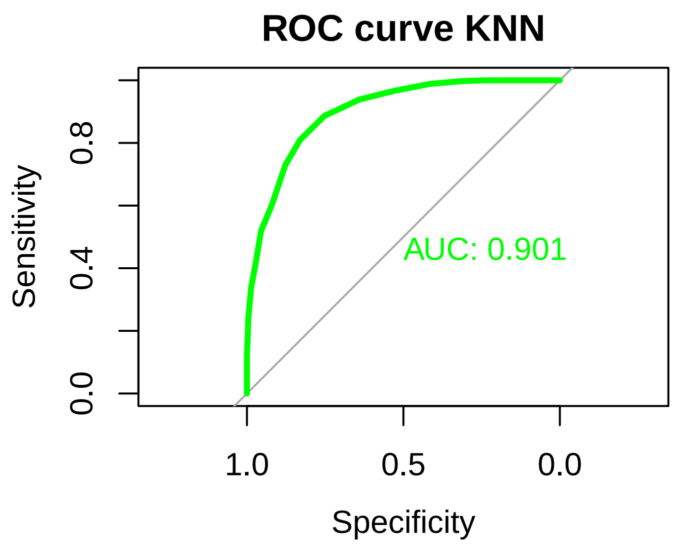
```r
library(pROC)


roc_glm <- roc(response = test$class, predictor = glm.probs)
plot(roc_glm, col="blue", lwd=3, print.auc=TRUE, main="ROC curve GLM")
```



```r
roc_qda <- roc(response = test$class, predictor = qda.preds$posterior[,"1"])
plot(roc_qda, col="red", lwd=3, print.auc=TRUE, main="ROC curve QDA")
```



```
roc_knn <- roc(response = test$class, predictor = probKNN)
plot(roc_knn, col="green", lwd=3, print.auc=TRUE, main="ROC curve KNN")
```



iv) (1P)

QDA and logistic regression performs pretty equal, but the ROC plot for KNN is substantially better. Because of this i would choose the KNN model.

Problem 4 (6P)

a) General formula for CV:

$$CV_K = \sum_{i=1}^{n} \frac{n_k}{N} MSE_k \ , \qquad MSE_k = \sum_{i \in C_k} (y_i - \hat{y}_{(-i)})^2$$

In LOOCV we have $K = N$, which means that $C_k = k$, and $n_k = 1$. This gives

$$CV = \frac{1}{N}\sum_{i=1}^{n}(y_i - \hat{y}_{(-i)})^2$$

We also have

$$\hat{y}_{(-i)} = \boldsymbol{x}_i^\top \hat{\boldsymbol{\beta}}_{(-i)} \hat{\boldsymbol{\beta}}_{(-i)} = (\mathbf{X}_{(-i)}^\top \mathbf{X}_{(-i)})^{-1}\mathbf{X}_{(-i)}^\top \mathbf{y}_{(-i)}$$

Hint 2 and 3 then gives us

$$\hat{\boldsymbol{\beta}}_{(-i)} = (\mathbf{X}^\top\mathbf{X} - \mathbf{x}_i\mathbf{x}_i^\top)^{-1}(\mathbf{X}^\top\mathbf{y} - \mathbf{x}_i y_i)$$

The Sherman-Morrison formula then gives

$$\hat{\boldsymbol{\beta}}_{(-i)} = \left((\mathbf{X}^\top\mathbf{X})^{-1} - \frac{(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{x}_i\mathbf{x}_i^\top(\mathbf{X}^\top\mathbf{X})^{-1}}{1 + \mathbf{x}_i^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{x}_i}\right)(\mathbf{X}^\top\mathbf{y} - \mathbf{x}_i y_i) \tag{21}$$

$$\Rightarrow \hat{y}_{(-i)} = \boldsymbol{x}_i^\top\left((\mathbf{X}^\top\mathbf{X})^{-1} - \frac{(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{x}_i\mathbf{x}_i^\top(\mathbf{X}^\top\mathbf{X})^{-1}}{1 + h_i}\right)(\mathbf{X}^\top\mathbf{y} - \mathbf{x}_i y_i) \tag{22}$$

$$\Rightarrow y_i - \hat{y}_{(-i)} = y_i - \boldsymbol{x}_i^\top\left((\mathbf{X}^\top\mathbf{X})^{-1} - \frac{(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{x}_i\mathbf{x}_i^\top(\mathbf{X}^\top\mathbf{X})^{-1}}{1 + h_i}\right)(\mathbf{X}^\top\mathbf{y} - \mathbf{x}_i y_i) =^{?} \frac{y_i - \hat{y}_i}{1 - h_i} \tag{23}$$

b) FALSE, FALSE, TRUE, FALSE