Compulsory exercise 1: Group 3

TMA4268 Statistical Learning V2023

Helle Villmones Haug, Hjalmar Jacob Vinje and Sanna Baug Warholm

23 February, 2023

Problem 1

For this problem you will need to include some LaTex code. Please install latex on your computer and then consult Compulsor1.Rmd for hints how to write formulas in LaTex.

a)

We know

$$Y = f(\mathbf{x}) + \varepsilon \mathrm{E}(\varepsilon) = 0 f(\mathbf{x}) = \mathbf{x}^T \boldsymbol{\beta}$$

so

$$\mathrm{E}(\widetilde{\boldsymbol{\beta}}) = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathrm{E}(\mathbf{Y}) = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathrm{E}(\mathbf{X}\beta + \epsilon) = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{X}\beta + 0 = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{X}\beta$$

And the covariance matrix is then

$$\mathrm{Cov}(\widetilde{\boldsymbol{\beta}}) = \mathrm{Cov}((\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{Y}) = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathrm{Cov}(\mathbf{Y})((\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-\mathbf{1}} \mathbf{X}^\top)^\top$$

Cov(Y) would be only the variance:

$$= (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \sigma^2 \mathbf{I}((\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top)^\top$$

b)

Expected value:

$$\mathrm{E}(\widetilde{f}(x_0)) = \mathrm{E}(\mathbf{x}_0^\top \widetilde{\boldsymbol{\beta}}) = \mathbf{x}_0^\top \mathrm{E}(\widetilde{\boldsymbol{\beta}}) = \mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{X}\beta$$

Variance:

$$\mathrm{Var}(\widetilde{f}(x_0)) = \mathrm{Var}(\mathbf{x}_0^\top \widetilde{\boldsymbol{\beta}}) = \mathbf{x}_0^\top \mathrm{Var}(\widetilde{\boldsymbol{\beta}}) \mathbf{x}_0 = \mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathrm{Cov}(\mathbf{Y})((\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-\mathbf{1}} \mathbf{X}^\top)^\top \mathbf{x_0}$$

Cov(Y) would be only the variance:

$$= \mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \sigma^2 \mathbf{I}((\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top)^\top \mathbf{x}_0$$

c)

Bias is a systematic error, in statistics it is an estimate of the systematic difference from the true value. If a model is inflexible, it could be underfitted and have a high bias.

Variance measures uncertainty. If variance is high, the estimate of the distribution function is likely to change for different input data. Very complex models might overfit data and get a high variance.

Irreducable error is the variance of a "unknown" variable which adds uncorrelated noise with a mean of 0 ("random", or unobserved influences), which means that it cannot be reduced by improving the fit of the model.

d)

Bias:

$$f(x_0) - E[\widetilde{f}(x_0)] = \mathbf{x}_0^\top \widetilde{\boldsymbol{\beta}} - \mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{X} \beta$$

Variance (using previous calculations):

$$Var(\widetilde{f}(x_0)) = \mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathrm{Cov}(\mathbf{Y}) ((\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-\mathbf{1}} \mathbf{X}^\top)^\top \mathbf{x_0}$$

Irreducible error:

$$Var(\varepsilon) = \sigma^2$$

MSE:

$MSE =$ irreducable error + variance + squared bias $= Var(\varepsilon) + \mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathrm{Cov}(\mathbf{Y}) ((\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-\mathbf{1}} \mathbf{X}^\top)^\top \mathbf{x_0} + (\mathbf{x_0}^\top \widetilde{\boldsymbol{\beta}}$-

e)

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

```
## [1] 100  81
```

```
x0 <- values$x0
dim(x0)
```

```
## [1] 81  1
```

```
beta <- values$beta
dim(beta)
```

```
## [1] 81  1
```

```
sigma <- values$sigma
sigma
```

```
## [1] 0.5
```

```
library(ggplot2)
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))
```

f)

Now we will create the variance function which takes the same inputs as the squared bias. As in e) you have to fill only the `value <- ...` and run the code to plot the variance.

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

Fill in the `exp_mse` of the following code to calculate the expected MSE for the same lambda values that you plugged in above. Then plot all the components together and find the value of λ which minimizes the expected MSE.

```
irrErr <- sigma^2
exp_mse <- BIAS + VAR + irrErr
min_lambda <- lambdas[which.min(exp_mse)]
dfMSE <- data.frame(lambdas = lambdas, mse = exp_mse, bias = BIAS, var = VAR, irrErr = irrErr)
```

```
dfBVI <- merge(dfBias, dfVar, by = "lambdas")

# <!-- ggplot(dfMSE, aes(x = lambdas)) +  -->
# <!--   geom_line(aes(y = bias, color = "BIAS squared")) +  -->
# <!--   geom_line(aes(y = var, color = "VAR")) +  -->
# <!--   geom_line(aes(y = irrErr, color = "irrErr")) +  -->
# <!--   geom_line(aes(y = mse, color = "Exp. MSE")) + -->
# <!--   scale_color_manual(values = c("BIAS" = "blue", "VAR" = "orange", "irrErr" = "pink", "Exp. MSE"
# <!--   xlab("lambda") + -->
# <!--   ylab("Exp. MSE") + -->
# <!--   ggtitle(paste("the lambda that minimize expected MSE is ", round(min_lambda, 3))) -->
```

Problem 2

a)

i)

The lm() function creates a variable to be estimatied which is multiplied with the binary variable of rankAssocProf and rankProf. The interpretation oof the number corresponding to the variables is that holding all other variables constant going from a AsstProf to AssocProf is assosiated with an increase in salary of 12,907.6 and going from AsstProf to Prof is assosiated with an increase in salary of 45,066.0 holding all other variables constant.

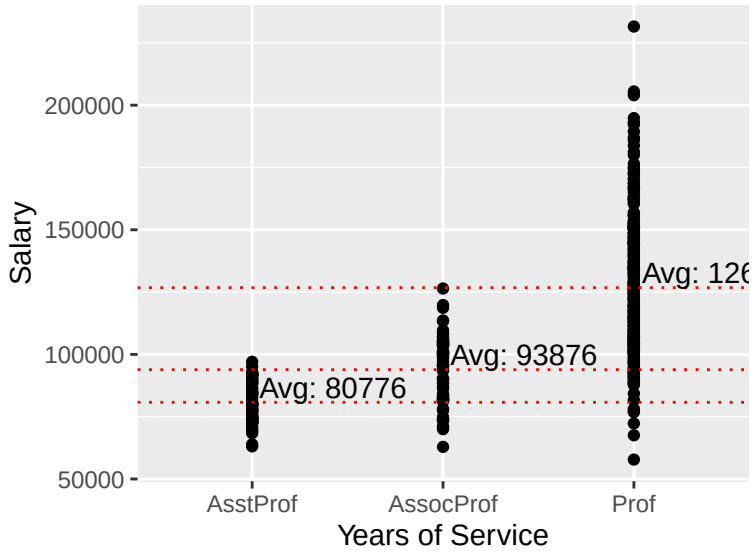
ii)

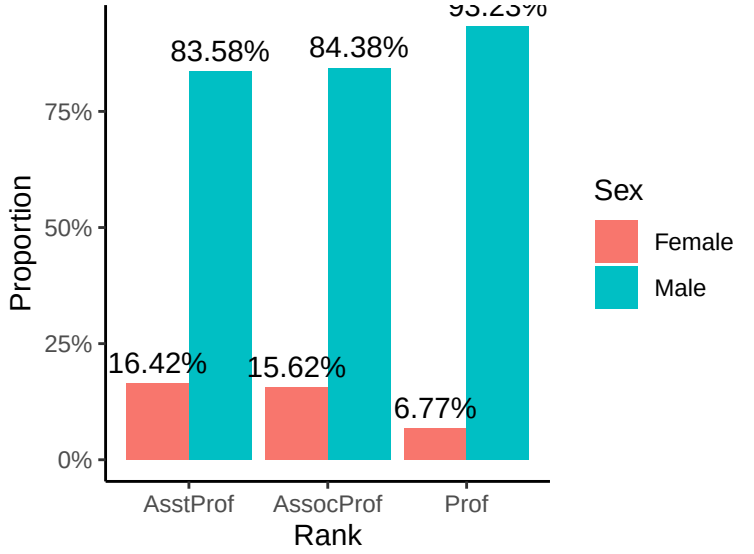
To test the whole categorical variable Rank, you can use an ANOVA test to compare the means of salary for each category of Rank.

```
             Df    Sum Sq   Mean Sq F value Pr(>F)
rank          2 1.432e+11 7.162e+10   128.2 <2e-16 ***
Residuals   394 2.201e+11 5.586e+08
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

This shows that rank as a whole has an impact on salary at the 99% confidence level.

b)





The first graph shows that professors earn more than assistante professors and associate professors. The second shows that women make up a higher proportion of AsstProf than Prof.

This is the reason why the regression with sex as the only covariate shows that sex is statistically significant in trems of salary, but it is no longer significant when controlling for rank and the other covariates.

c)

```
model1 <- lm(salary ~ ., data = Salaries)
options(repos = c(CRAN = "https://cran.rstudio.com/"))
install.packages("ggfortify")
```

```
##
## The downloaded binary packages are in
##  /var/folders/wk/x86_p6511195p594k6qnb98h0000gn/T//RtmplJvveo/downloaded_packages
```

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

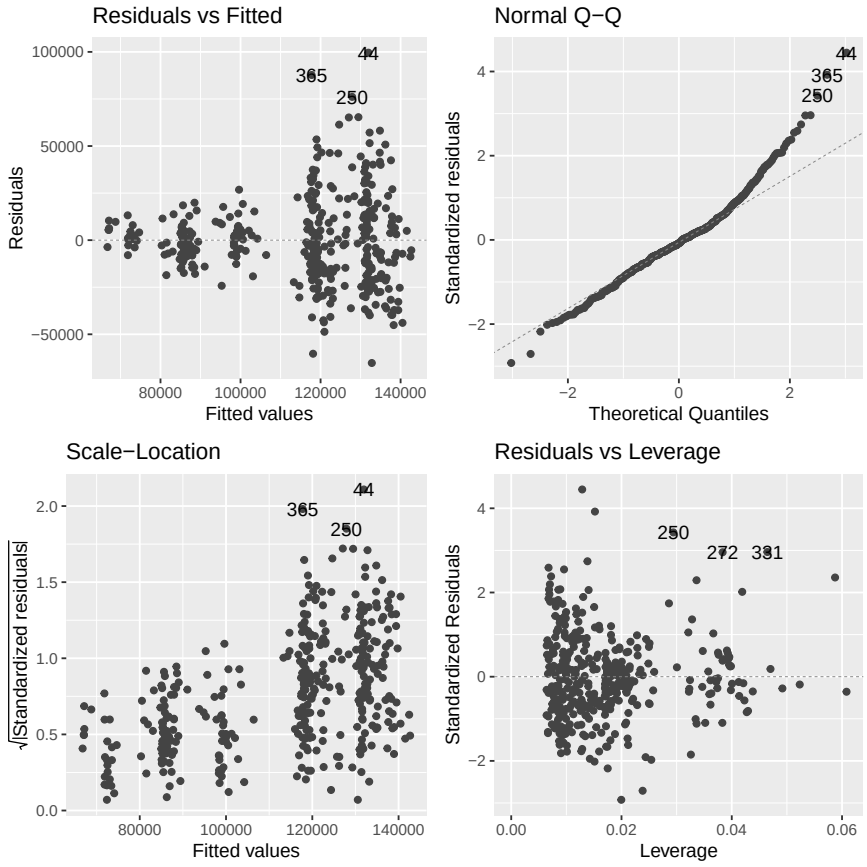


Figure 1: Diagnostic for model1

The first plot showing residuals vs fitted values shows that the data is heteroskedastic, meaning that the variance is not constant. This breaks the assumption of homoscedasticity.

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)
```

The residual variance is still heteroskedastic.

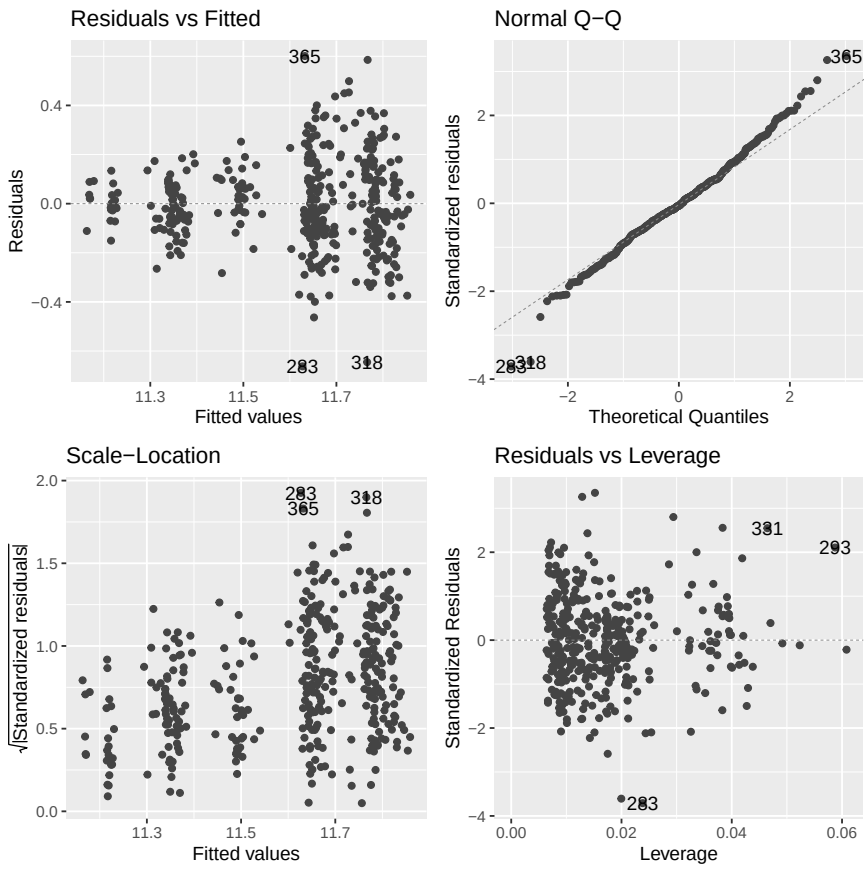


Figure 2: Diagnostic for model2

```
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)
```

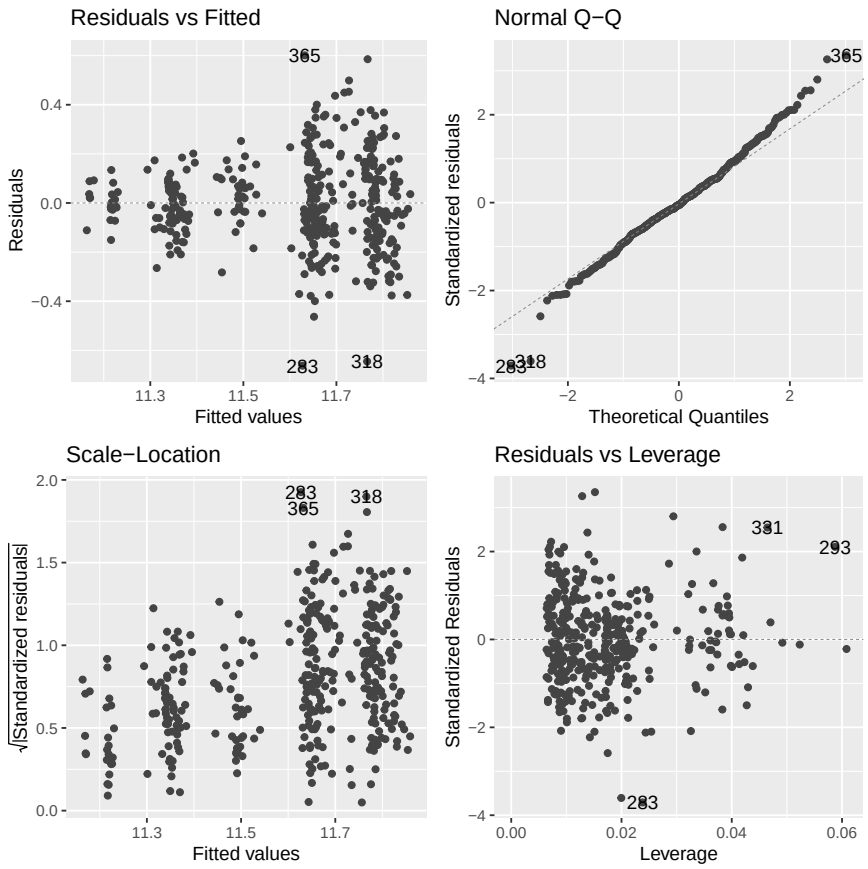


Figure 3: Diagnostic for model3

d)

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

```
##
## Call:
## lm(formula = log_salary ~ rank + discipline + yrs.since.phd +
##     yrs.service + sex + sex:yrs.since.phd - salary, data = Salaries)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.66187 -0.10831 -0.00951  0.09846  0.60143
##
```

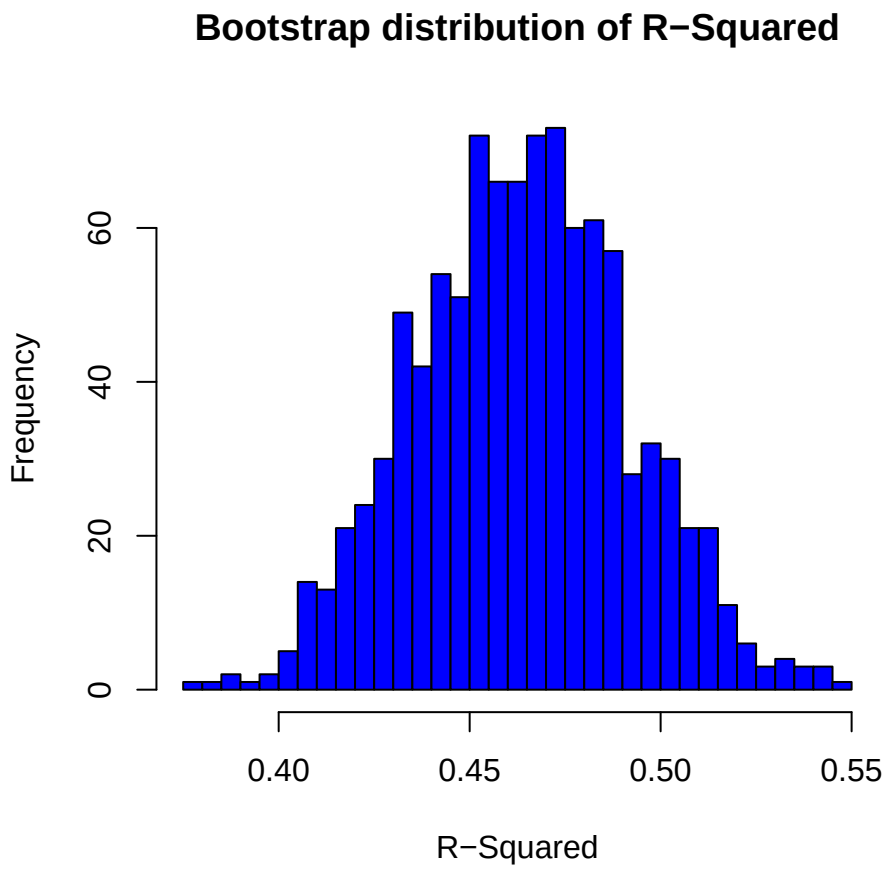
```
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            11.1537511  0.0591759 188.485  < 2e-16 ***
## rankAssocProf           0.1528200  0.0335575   4.554 7.05e-06 ***
## rankProf                0.4482679  0.0344343  13.018  < 2e-16 ***
## disciplineB             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
```

ii) The interaction term is not statistically significant, so we can not conclude that Bert-Ernie is correct.

e)

```r
# 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)
```

```r
# ii)
hist(boot_results$t, main = "Bootstrap distribution of R-Squared",
     xlab = "R-Squared", col = "blue", border = "black", breaks = 30)
```




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

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

```
quant = boot_results$t[25:975]
summary(quant)
```

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

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

f)

```
# 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
```

He can now no longer be confident with 95% certainty that we will earn at least $75 000 at this time.

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

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

```
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

The Bigfoot Field Researchers Organization (BFRO)-problem, using the suggested code

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

Plus the data preparation, not included in the pdf (but shown in the Rmarkdown-file)

Next, setting seed and creating training- and test-sets:

```
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, ]
```

Task a)

(i)

```
model <- glm(class~longitude+latitude+visibility+fur+howl+saw+heard, family="binomial", data=train)

glm_probabilities <- predict(model, test, type="response")
no_classified = sum(glm_probabilities >= 0.5)
no_classified # Number of reports classified as clear sightings: 441
```

```
## [1] 441
```

Number of clear sightings: 441

(ii)

```
summary(model)
```

```
##
## Call:
## glm(formula = class ~ longitude + latitude + visibility + fur +
##     howl + saw + heard, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0710  -1.0149  -0.4291   1.0007   2.1469
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.989051   0.422048   2.343 0.019106 *
## longitude   -0.003112   0.003460  -0.900 0.368374
## latitude    -0.036988   0.009849  -3.756 0.000173 ***
## visibility  -0.005681   0.023686  -0.240 0.810449
## furTRUE      0.575172   0.136328   4.219 2.45e-05 ***
## howlTRUE    -0.792152   0.189803  -4.174 3.00e-05 ***
## sawTRUE      1.291894   0.097630  13.233  < 2e-16 ***
## heardTRUE   -1.075540   0.099634 -10.795  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2948.6  on 2126  degrees of freedom
## Residual deviance: 2509.9  on 2119  degrees of freedom
## AIC: 2525.9
##
## Number of Fisher Scoring iterations: 4
```

The coefficients for sawTRUE is 1.29, which means that the average change in log odds with one unit increase of the value.

```
change <- exp(1.29)
print(change)
```

```
## [1] 3.632787
```

The answer is therefore D) Multiply by 3.64")

Task b)

(i)

```
require(MASS)
qda_model <- qda(class~longitude+latitude+visibility+fur+howl+saw+heard, data=train)
qda_predicted <- predict(qda_model, test)
table(qda_predicted$class)
```

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

Number of clear sightings: 626

(ii)

1): True, 2): False, 3): False, 4): False

Task c)

(i)

```
require(class)
?knn()
knn_model <- knn(train=train, test=test, cl=train$class, k=25, prob=TRUE)

table(knn_model)
```

```
## knn_model
##   0   1
## 471 441
```

Number of clear sightings: 441

Task c)

(ii)

Trade-off between bias and variance, higher k -> less variance and more bias. How to tune the k-parameter in a better way: I could create plots for different k-values and choose the k-value with the lowest error.

Task d)

(i)

Prediction, because we use existing data for creating a model that will classify a new instance correctly as often as possible. With inference, we are more interested in evaluating the relationship between the response variables and the predictor, i.e. the interepretability of the model. All models are interesting with predicting, but KNN and QDA would not been as relevant for inference.

(ii)

Sensitivity: True positive value, probability of a positive test result, given that instance truly is positive.
Specificity: True negative value, probability of a negative test result, given that instance tryly is negative.

For all confusion matrices: rows show prediction values and columns show true values.

```
# Confusion matrix Glm
glm_predicted <- rep(0, 912)
glm_predicted[glm_probabilities > 0.5] <- 1
table(glm_predicted, test$class)
```

```
##
## glm_predicted   0   1
##             0 323 148
##             1 142 299
```

```
glm_sensitivity <- 299/(299+148)
glm_specificity <- 323/(323+142)
glm_sensitivity
```

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

```
glm_specificity
```

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

Glm sensitivity is 66,9 % and specificity is 69,5 %

```
# Confusion matrix QDA
table(qda_predicted$class, test$class)
```

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

```
qda_sensitivity <- 389/(389+58)
qda_specificity <- 228/(228+237)
qda_sensitivity
```

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

```
qda_specificity
```

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

QDA sensitivity is 87,0 % and specificity is 49,0 %

```
# Confusion matrix KNN
table(knn_model, test$class)
```

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

```
knn_sensitivity <- 362/(362+85)
knn_specificity <- 386/(386+79)
knn_sensitivity
```

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

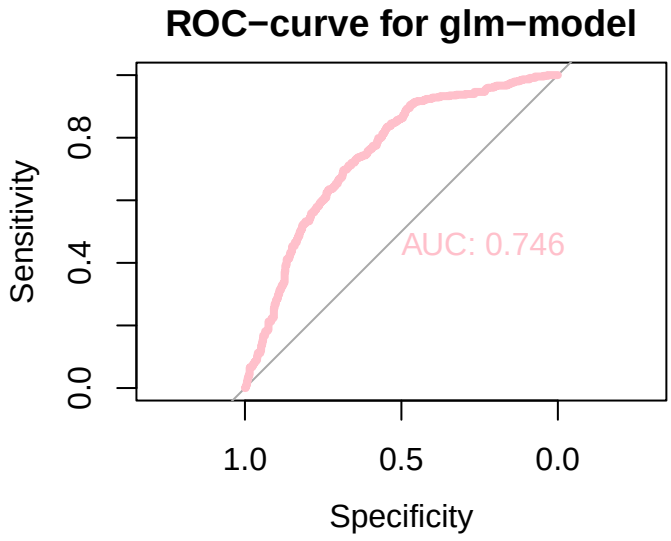
```
knn_specificity
```

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

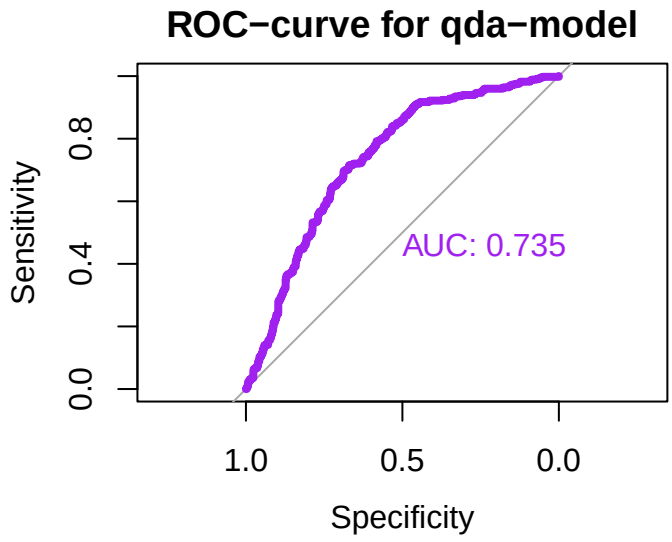
KNN sensitivity is 81,0 % and specificity is 83,0 %

(iii)

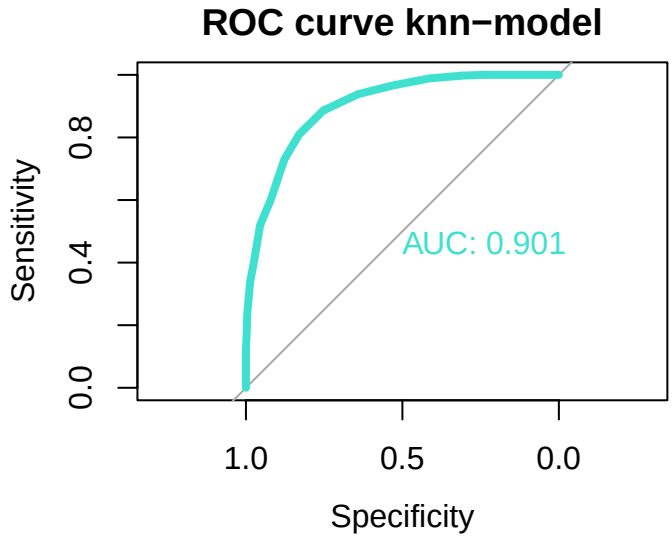
```
library(pROC)

glm_roc <- roc(response = test$class, predictor = glm_probabilities)
plot(glm_roc, col="pink", lwd=4, print.auc=TRUE, main="ROC-curve for glm-model")
```



```
qda_roc <- roc(response = test$class, predictor = qda_predicted$posterior[,"1"])
plot(qda_roc, col="purple", lwd=4, print.auc=TRUE, main="ROC-curve for qda-model")
```




```
knn_probabilities <- ifelse(knn_model == 0, 1 - attributes(knn_model)$prob,attributes(knn_model)$prob)
knn_roc <- roc(response = test$class, predictor = knn_probabilities)
plot(knn_roc, col="turquoise", lwd=4, print.auc=TRUE, main="ROC curve knn-model")
```



(iv)

Glm and QDA performs similar for ROC, while KKN performs significantly better. Would therefore choose the KNN-classifier for this problem.

Problem 4

a)

b)

(i): False, (ii): False, (iii): True, (iv): False