STAT 302: Homework 8

Jaiden Atterbury

Due 06-02-2023 at 11:59 PM

NOTE: In class we were told to not check model assumptions or interpret parameter meanings in the context of the problem. I will however still test the normality assumption in multiple linear regression models as well as write out the model equations.

1. Using **Handout 1 dataset** on Canvas. Estimate the prediction error of the model using (i) Training and Testing and (ii) 5-fold cross-validation.

In this problem we will use the Handout 1 dataset on Canvas and use the COMMIT variable as our dependent variable and do some diagnostics to choose our predictor variables.

Since we know there is no missing data and quite a few data points we will keep all of our categorical variables as independent variables. However, for our continuous variables we will make a correlation matrix to assess which variables we'll keep.

```
# Remove the categorical variables:
cat_vars_removed <- Handout_1[, -c(3, 4, 5)]

# Create a correlation matrix with all of the continuous variables:
cor(cat_vars_removed)
```

```
##                COMMIT         AGE      SALARY   CLASSSIZE   RESOURCES   AUTONOMY
## COMMIT     1.0000000  0.18094973  0.38070161 -0.36595143  0.32994994 0.30979764
## AGE        0.1809497  1.00000000  0.10563458  0.04912606  0.17689868 0.08721293
## SALARY     0.3807016  0.10563458  1.00000000 -0.52143944  0.39674186 0.01661954
## CLASSSIZE -0.3659514  0.04912606 -0.52143944  1.00000000 -0.21293023 0.25116551
## RESOURCES  0.3299499  0.17689868  0.39674186 -0.21293023  1.00000000 0.03983696
## AUTONOMY   0.3097976  0.08721293  0.01661954  0.25116551  0.03983696 1.00000000
## CLIMATE    0.4271400 -0.12804799  0.23932435 -0.24608641  0.22124426 0.31358756
## SUPPORT    0.2424249  0.02112233  0.32526239 -0.30817162  0.01128116 0.17733753
##               CLIMATE     SUPPORT
## COMMIT     0.4271400  0.24242491
## AGE       -0.1280480  0.02112233
## SALARY     0.2393244  0.32526239
## CLASSSIZE -0.2460864 -0.30817162
## RESOURCES  0.2212443  0.01128116
## AUTONOMY   0.3135876  0.17733753
## CLIMATE    1.0000000  0.28470265
## SUPPORT    0.2847027  1.00000000
```

As can be seen from the above correlation matrix, there are no variables that are strongly correlated with the commitment score of a teacher, however SALARY, CLASSSIZE, and CLIMATE all have correlation

coefficients that are higher than 0.35, which is better than the other variables. Thus we will also add those to our list predictor variables. In the following code chunk we will create our new dataset with only our variables of interest

```
# Select our variables of interest:
Handout_1_new <- Handout_1 %>%
  select(COMMIT, SEX, SCHTYPE, SCHLEVEL, SALARY, CLASSSIZE, CLIMATE)
```

We will now test the normality of our dependent variable to see if we need to make a transformation or simply proceed with the regression.

```
# Run a Shapiro test to assess the normality of the pendent variable:
shapiro.test(Handout_1_new$COMMIT)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Handout_1_new$COMMIT
## W = 0.98438, p-value = 0.08778
```

As seen from the above Shapiro-Wilk normality test, the p-value is 0.08778, thus we fail to reject the null hypothesis at the 5% level of significance. Hence we assume that the dependent variable is approximately normally distributed and we can continue on with our regression normally.

Training and testing:
Now we will split our new data set into a training and testing set in order to evaluate our model accuracy.

```
# Set seed for reproducibility:
set.seed(1)

# Split the data set into 80% training and 20% testing:
index <- createDataPartition(Handout_1_new$SEX, p=.8, list=FALSE)
```

Below we will officially split our data set into the training and testing set. Furthermore, we will check the number of observations in each data set to see if splitting worked correctly.

```
# Create the training set:
train_data  <- Handout_1_new[index, ]

# Create the testing set:
test_data <- Handout_1_new[-index, ]

# Test and see if the split worked correctly:
nrow(train_data)
```

```
## [1] 120
```

```
nrow(test_data)
```

```
## [1] 30
```

As we can see from the above output, we had 120 observations in our training set and 30 observations in our testing set which is exactly what we would expect from an 80/20 split of 150 observations.

We will now construct our model using our training data that we created above.

```
# Create the model on the training set:
model_1 <- lm(COMMIT ~ factor(SEX) + factor(SCHTYPE) + factor(SCHLEVEL) + SALARY
              + CLASSSIZE + CLIMATE, data = train_data)

# Get a model summary:
summary(model_1)
```

```
##
## Call:
## lm(formula = COMMIT ~ factor(SEX) + factor(SCHTYPE) + factor(SCHLEVEL) +
##     SALARY + CLASSSIZE + CLIMATE, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -25.2519  -9.8949   0.8926   8.7568  27.6590
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        18.7636    25.6721   0.731   0.4664
## factor(SEX)2        1.8785     2.9539   0.636   0.5261
## factor(SCHTYPE)2    5.2398     3.2549   1.610   0.1103
## factor(SCHTYPE)3    6.8303     4.7615   1.434   0.1543
## factor(SCHLEVEL)2  -5.4967     2.9703  -1.851   0.0669 .
## factor(SCHLEVEL)3  -6.5230     3.2574  -2.003   0.0477 *
## SALARY              0.3837     0.5563   0.690   0.4918
## CLASSSIZE          -0.4742     0.3316  -1.430   0.1555
## CLIMATE             2.0679     0.4920   4.203 5.35e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.71 on 111 degrees of freedom
## Multiple R-squared:  0.346,  Adjusted R-squared:  0.2989
## F-statistic: 7.342 on 8 and 111 DF,  p-value: 8.425e-08
```

As can be seen from the above model output ignoring the significance of variables and model assumptions since that isn't the aim of this problem the regression equation is $\widehat{\text{COMMIT}} = 18.8 + 1.9 \cdot \text{Female} + 5.2 \cdot \text{Private} + 6.8 \cdot \text{Charter} - 5.5 \cdot \text{Middle} - 6.5 \cdot \text{High} + 0.4 \cdot \text{SALARY} - 0.5 \cdot \text{CLASSSIZE} + 2.1 \cdot \text{CLIMATE}$

Where Female refers to SEX, Private and Charter refer to SCHTYPE, and Middle and High refer to SCHLEVEL.

Lastly we will analyze the estimated prediction error of the model.

```
# Create the prediction vector:
predictions_1 <- model_1 %>% predict(test_data)

# Print a data frame of estimated prediction error:
data.frame(R2 = R2(predictions_1, test_data$COMMIT),
           RMSE = RMSE(predictions_1, test_data$COMMIT),
           MAE = MAE(predictions_1, test_data$COMMIT))
```

```
##          R2     RMSE      MAE
## 1 0.2900786 12.03883 10.24386
```

As can be seen the root mean squared error (RMSE) is 12.03883, the mean absolute error (MAE) is 10.24386 and the R squared value is 0.29. Based on the size of the sample these aren't the greatest error value, but they don't have too much meaning unless compared with another model. Which we will create using 5-fold cross-validation below.

5-fold cross-validation:
Below we will fit a new model onto the full data set using 5-fold cross-validation.

```
# Set seed for reproducibility:
set.seed(1)

# Specify the cross-validation method:
ctrl <- trainControl(method = "cv", number = 5)

# Build the model on the full data set:
model_2 <- train(COMMIT ~ factor(SEX) + factor(SCHTYPE) + factor(SCHLEVEL)
                 + SALARY + CLASSSIZE + CLIMATE, data = Handout_1_new,
                 method = "lm", trControl = ctrl)

# Get the model summary:
summary(model_2)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -25.3807  -9.6641   0.6781   8.5344  26.6122
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)           4.9426    22.6716   0.218   0.8277
## `factor(SEX)2`        2.9086     2.5805   1.127   0.2616
## `factor(SCHTYPE)2`    4.4594     2.8348   1.573   0.1179
## `factor(SCHTYPE)3`    4.4128     4.3168   1.022   0.3084
## `factor(SCHLEVEL)2`  -4.3818     2.5694  -1.705   0.0903 .
## `factor(SCHLEVEL)3`  -6.4128     2.7821  -2.305   0.0226 *
## SALARY                0.8168     0.4792   1.705   0.0905 .
## CLASSSIZE            -0.5276     0.2918  -1.808   0.0727 .
## CLIMATE               1.7229     0.4101   4.202 4.68e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.49 on 141 degrees of freedom
## Multiple R-squared:  0.3436, Adjusted R-squared:  0.3064
## F-statistic: 9.228 on 8 and 141 DF,  p-value: 3.588e-10
```

As can be seen from the above model output ignoring the significance of variables and model assumptions since that isn't the aim of this problem the regression equation is $\widehat{\text{COMMIT}} = 4.9 + 2.9 \cdot \text{Female} + 4.5 \cdot \text{Private} + 4.4 \cdot \text{Charter} - 4.4 \cdot \text{Middle} - 6.4 \cdot \text{High} + 0.8 \cdot \text{SALARY} - 0.5 \cdot \text{CLASSSIZE} + 1.7 \cdot \text{CLIMATE}$

Where Female refers to SEX, Private and Charter refer to SCHTYPE, and Middle and High refer to SCHLEVEL. This equation didn't change much from the one made with the training set, the only noticeable difference is in the model intercept which is usually the most sensitive to small changes in the data.

Lastly we will analyze the estimated prediction error of the model.

```
# Get the estimated prediction errors:
print(model_2)
```

```
## Linear Regression
##
## 150 samples
##   6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 121, 121, 120, 119, 119
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   12.53976  0.3132769  10.57397
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

As can be seen the root mean squared error (RMSE) is 12.53976, the mean absolute error (MAE) is 10.57397 and the R squared value is 0.3132769. Based on the size of the sample these aren't the greatest error values. Furthermore, they are very similar to those computed with the training and test set. In particular this data set has slightly higher prediction error but also a higher R squared value. In conclusion both of the above models aren't the greatest for predicting COMMIT, however when compared to each other they are virtually indistinguishable.

2. Using the **Fish consumption dataset** on Canvas. Evaluate the accuracy of the model using K-fold cross-validation and repeated K-fold cross-validation. Use obesity as your outcome variable and choose your own k (usually between 5 to 10). Use independent variables: age, experience, stress, sex and job status.

In this problem we will use the Fish consumption data set on Canvas. Using the obesity variable as our dependent variable and age, experience, stress, sex and job status as our independent variables, we will use K-fold cross-validation and repeated K-fold cross-validation to evaluate the training and testing accuracy of the model. Specifically, we will be using a k of 10 for our cross-validation. Like the last problem we won't be testing any assumptions, however we will still present the regression equation. First we will select only the columns of importance from the data set.

```
# Select the dependent and independent variables:
FC_edited_new <- FC_edited %>%
  select(Obese, Age, Experience, Stress, Sex, Job.Status)
```

10-fold cross-validation:
Since this is a logistic regression model (obesity is binary), we will do 10-fold cross-validation as well as using a training and testing set (80/20 split) to get training and testing accuracy.

```
# Set the seed for reproducibility:
set.seed(1)

# Partition data and create index matrix of selected values:
index_2 <- createDataPartition(FC_edited$Obese, p=.8, list=FALSE, times=1)
```

We will now officially split the data into a training and testing set and check to see if this split was done correctly.

```
# Create the training set:
train_data_2  <- FC_edited_new[index_2, ]

# Create the testing set:
test_data_2 <- FC_edited_new[-index_2, ]

# Test and see if the split worked correctly:
nrow(train_data_2)
```

```
## [1] 160
```

```
nrow(test_data_2)
```

```
## [1] 39
```

As we can see from the above output, we had 160 observations in our training set and 39 observations in our testing set which is exactly what we would expect from an 80/20 split of 199 observations.

We will now construct our model using our training data that we created above.

```
# Set the seed for reproducibility:
set.seed(1)

# Set the type of cross-validation
ctrlspecs_2 <- trainControl(method="cv",
                            number=10,
                            savePredictions="all",
                            classProbs=TRUE)

# Build the model on the training data:
model_3 <- train(Obese ~ Age + Stress + Experience + factor(Job.Status)
                 + factor(Sex), data=train_data_2,  method="glm",
                 family = "binomial", trControl=ctrlspecs_2)

# Get the model summary:
summary(model_3)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
```

```
## -1.7316  -0.2794  -0.1429   0.7486   2.5451
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -1.15682    1.43199  -0.808   0.4192
## Age                      -0.04463    0.03410  -1.309   0.1906
## Stress                   -0.05713    0.02684  -2.128   0.0333 *
## Experience               -0.02002    0.02863  -0.699   0.4844
## `factor(Job.Status)Part` -0.05744    0.63011  -0.091   0.9274
## `factor(Sex)Male`         4.66713    1.04768   4.455  8.4e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 205.92  on 159  degrees of freedom
## Residual deviance: 122.80  on 154  degrees of freedom
## AIC: 134.8
##
## Number of Fisher Scoring iterations: 7
```

As can be seen from the above model output ignoring the significance of variables and model assumptions since that isn't the aim of this problem. the regression equation is $\log\left(\frac{P(\text{Obesity})}{1-P(\text{Obesity})}\right) = -1.2 - 0.04 \cdot \text{Age} - 0.06 \cdot \text{Stress} - 0.02 \cdot \text{Experience} - 0.06 \cdot \text{Part} + 4.7 \cdot \text{Male}$. Where Part stands for part time job status.

Now we will compute the training and testing accuracy of this model.

```
# Get the model training accuracy:
print(model_3)
```

```
## Generalized Linear Model
##
## 160 samples
##   5 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 145, 144, 144, 144, 143, 144, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7441422  0.4597091
```

The training accuracy of this model was 74.41%, meaning that the model correctly classifies a person as obese or not obese 74.41% of the time. (on the training data).

```
# Get testing set predictions:
predictions <- predict(model_3, newdata=test_data_2)

# Turn the Obese variable as a factor in the testing set to get accuracy.
test_data_2$Obese <- as.factor(test_data_2$Obese)
```

```r
# Get testing set accuracy:
confusionMatrix(data=predictions, test_data_2$Obese)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  23   4
##        Yes  3   9
##
##                Accuracy : 0.8205
##                  95% CI : (0.6647, 0.9246)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.02654
##
##                   Kappa : 0.5882
##
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 0.8846
##             Specificity : 0.6923
##          Pos Pred Value : 0.8519
##          Neg Pred Value : 0.7500
##              Prevalence : 0.6667
##          Detection Rate : 0.5897
##    Detection Prevalence : 0.6923
##       Balanced Accuracy : 0.7885
##
##        'Positive' Class : No
##
```

The testing accuracy of this model was 82.05%, meaning that the model correctly classifies a person as obese or not obese 82.05% of the time. (on the testing data). We will now do the exact same process but for repeated 10-fold cross-validation.

Repeated 10-fold cross-validation:
We will now construct our model using our training data that we created above using 3 repeated of 10-fold cross-validation.

```r
# Set the seed for reproducibility:
set.seed(1)

# Set the type of cross-validation
ctrlspecs_3 <- trainControl(method="repeatedcv",
                            number=10,
                            repeats=3,
                            savePredictions="all",
                            classProbs=TRUE)

# Build the model on the training data:
model_4 <- train(Obese ~ Age + Stress + Experience + factor(Job.Status)
                 + factor(Sex), data=train_data_2,  method="glm",
                 family = "binomial", trControl=ctrlspecs_3)
```

```r
# Get the model summary:
summary(model_4)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7316  -0.2794  -0.1429   0.7486   2.5451
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -1.15682    1.43199  -0.808   0.4192
## Age                       -0.04463    0.03410  -1.309   0.1906
## Stress                    -0.05713    0.02684  -2.128   0.0333 *
## Experience                -0.02002    0.02863  -0.699   0.4844
## `factor(Job.Status)Part`  -0.05744    0.63011  -0.091   0.9274
## `factor(Sex)Male`          4.66713    1.04768   4.455  8.4e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 205.92  on 159  degrees of freedom
## Residual deviance: 122.80  on 154  degrees of freedom
## AIC: 134.8
##
## Number of Fisher Scoring iterations: 7
```

As can be seen from the above model output ignoring the significance of variables and model assumptions since that isn't the aim of this problem. the regression equation is $\log\left(\frac{P(\text{Obesity})}{1-P(\text{Obesity})}\right) = -1.2 - 0.04 \cdot \text{Age} - 0.06 \cdot \text{Stress} - 0.02 \cdot \text{Experience} - 0.06 \cdot \text{Part} + 4.7 \cdot \text{Male}$. Where Part stands for part time job status. This is exactly the same as the last model, meaning any coefficient changes were very small (past 5 decimal places). We can also see there was relatively no change since the AIC also remained unchanged from the previous model,

Now we will compute the training and testing accuracy of this model.

```r
# Get the model training accuracy:
print(model_4)
```

```
## Generalized Linear Model
##
## 160 samples
##   5 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 145, 144, 144, 144, 143, 144, ...
## Resampling results:
```

```
##
##   Accuracy   Kappa
##   0.7445833  0.4535543
```

The training accuracy of this model was 74.46%, meaning that the model correctly classifies a person as obese or not obese 74.46% of the time (on the training data). This is slightly better than the previous model without repetition.

```r
# Get testing set predictions:
predictions_2 <- predict(model_4, newdata=test_data_2)

# Get testing set accuracy:
confusionMatrix(data=predictions_2, test_data_2$Obese)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  23   4
##        Yes  3   9
##
##                Accuracy : 0.8205
##                  95% CI : (0.6647, 0.9246)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.02654
##
##                   Kappa : 0.5882
##
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 0.8846
##             Specificity : 0.6923
##          Pos Pred Value : 0.8519
##          Neg Pred Value : 0.7500
##              Prevalence : 0.6667
##          Detection Rate : 0.5897
##    Detection Prevalence : 0.6923
##       Balanced Accuracy : 0.7885
##
##        'Positive' Class : No
##
```

The testing accuracy of this model was 82.05%, meaning that the model correctly classifies a person as obese or not obese 82.05% of the time. (on the testing data). The testing accuracy is exactly the same as with normal 10-fold cross-validation. As can be seen, there is not much difference between normal and repeated 10-fold cross validation on this particular split of training and testing data.

3. Define a function that loops through each element of a matrix and replaces each element with the row index minus the column index. Create a matrix and demonstrate the use of this function on your matrix. **Hint:** use the functions *ncol()* and *nrow()* to find the number of columns and number of rows in your matrix input.

In this problem, we will define a function that loops through each element of a matrix and replaces each element with the row index minus the column index.

Defining the function:

```
# Define the function:
replace_element <- function(matrix) {
  # Loop through each row element:
  for (row_index in 1:nrow(matrix)) {

    # Loop through each column element:
    for (column_index in 1:ncol(matrix)) {

      # Replace the number at [row_index, column_index] with
      # row_index - column_index:
      matrix[row_index, column_index] <- row_index - column_index
    }
  }
  # Return the altered matrix:
  return(matrix)
}
```

Now that we have defined a function that loops through each element of a matrix and replaces each element with the row index minus the column index, we will create a matrix and demonstrate the use of this function on that matrix.

Test the function:

```
# Create a matrix to test the function on:
test_matrix <- matrix(1:16, nrow=4, ncol=4, byrow=F)

# Test the function on this matrix:
replace_element(test_matrix)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    0   -1   -2   -3
## [2,]    1    0   -1   -2
## [3,]    2    1    0   -1
## [4,]    3    2    1    0
```

As can be seen from the above output, every element of the input matrix has changed, and in particular, every element is what we would expect when taking the row index minus the column index.

4. Answer the following questions:

(a) Write a function called 'isPassingGrade' whose input x is a number, and which returns FALSE if x is lower than 50 and TRUE otherwise.

In this part of the problem, we will write a function called 'isPassingGrade' whose input x is a number, and which returns FALSE if x is lower than 50 and TRUE otherwise.

Create the function isPassingGrade:

```
# Define the function:
isPassingGrade <- function(x) {
  # If x is less than 50, return FALSE:
```

```
  if (x < 50) {
    return(FALSE)
  }
  # Otherwise, return TRUE
  return(TRUE)
}
```

Since the function in part (b) requires the implementation of 'isPassingGrade', we will wait to test the function until after the function in part (b) is implemented.

(b) Write a function called 'sendMessage' whose input x is a number, and which prints 'Congratulations' if 'isPassingGrade(x)' is TRUE and prints Oh no! if 'isPassingGrade(x)' is FALSE.

In this part of the problem, we will write a function called 'sendMessage' whose input x is a number, and which prints 'Congratulations' if 'isPassingGrade(x)' is TRUE and prints Oh no! if 'isPassingGrade(x)' is FALSE.

Create the function sendMessage:

```
# Define the function:
sendMessage <- function(x) {
  # If isPassingGrade(x) is TRUE, print 'Congratulations':
  if (isPassingGrade(x)) {
    print('Congratulations')
  }
  # Else, isPassingGrade(x) is FALSE, print 'Oh no!':
  else {
    print('Oh no!')
  }
}
```

(c) Write a function called 'gradeSummary' whose input x is a number. Your function will return a list with two elements, named 'letter.grade' and 'passed'. The letter grade will be ' "A" ' if x is at least '90'. The letter grade will be ' "B" ' if x is between 80 and 90. The letter grade will be ' "F" ' if x is lower than 80. If the student's letter grade is an A or B, 'passed' should be TRUE; 'passed' should be FALSE otherwise.

We will now test the function, and the function in part (a), for the two different cases.

Testing case 1: $x < 50$:

```
# Test case 1:
sendMessage(49)
```

```
## [1] "Oh no!"
```

As can be seen, since $x = 49 < 50$, 'isPassingGrade' returned FALSE, and thus 'sendMessage' returned 'Oh no!' as expected.

Testing case 1: $x \geq 50$:

```r
# Test case 2:
sendMessage(50)
```

```
## [1] "Congratulations"
```

As can be seen, since $x = 50 \geq 50$, 'isPassingGrade' returned TRUE, and thus 'sendMessage' returned 'Congratulations' as expected.

Thus as can be seen from the above tests, 'isPassingGrade' and 'sendMessage' were implemented correctly based on the instructions presented in the problem.

In this part of the problem, we will write a function called 'gradeSummary' whose input x is a number. Your function will return a list with two elements, named 'letter.grade' and 'passed'. The letter grade will be ' "A" ' if x is at least '90'. The letter grade will be ' "B" ' if x is between 80 and 90. The letter grade will be ' "F" ' if x is lower than 80. If the student's letter grade is an A or B, 'passed' should be TRUE; 'passed' should be FALSE otherwise.

Create the function gradeSummary:

```r
# Define the function:
gradeSummary <- function(x) {
  # Determine the letter grade:
  if (x >= 90) {
    letter.grade <- "A"
  } else if (x >= 80 & x < 90) {
    letter.grade <- "B"
  } else {
    letter.grade <- "F"
  }

  # Determine if student passed or failed:
  if (letter.grade %in% c("A", "B")) {
    passed <- TRUE
  } else {
    passed <- FALSE
  }

  # Define the return list:
  grade_summary <- list(letter.grade, passed)

  # Give the return list names:
  names(grade_summary) <- c("letter.grade", "passed")

  # Return the list:
  return(grade_summary)
}
```

Below we will test our function implementation for all three different possibilities of x.

Testing case 1: x $\geq$ 90:

```r
gradeSummary(90)
```

```
## $letter.grade
```

```
## [1] "A"
##
## $passed
## [1] TRUE
```

As can be seen from the above function call, since $x = 90 \geq 90$ we got a letter.grade of "A" and a passed value of TRUE. These are the expected values from this specific input.

Testing case 2: $80 \leq x < 90$:

```
gradeSummary(80)
```

```
## $letter.grade
## [1] "B"
##
## $passed
## [1] TRUE
```

As can be seen from the above function call, since $x = 80 \geq 80$ we got a letter.grade of "B" and a passed value of TRUE. These are the expected values from this specific input.

Testing case 3: $x < 80$:

```
gradeSummary(79)
```

```
## $letter.grade
## [1] "F"
##
## $passed
## [1] FALSE
```

As can be seen from the above function call, since $x = 79 < 80$ we got a letter.grade of "F" and a passed value of FALSE. These are the expected values from this specific input.

Thus as can be seen from the above tests, 'gradeSummary' was implemented correctly based on the instructions for the problem.

5. Find the maximum likelihood estimator (MLE) of σ from the normal distribution, assuming μ is constant.

In this problem we will find the maximum likelihood estimator (MLE) of σ from the normal distribution, assuming μ is constant. If we suppose that $X_1, X_2, \dots X_n \overset{i.i.d.}{\sim} Norm(\mu_0, \sigma)$, assuming that μ_0 is known. Hence the PDF of the X_i's can be written as $f(x_i) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{\frac{(x_i-\mu_0)^2}{2\sigma^2}}$, $-\infty < x_i < \infty$. With that being said, to find the maximum likelihood estimator of σ, we must find the likelihood function, take the natural log of the likelihood function, find the critical point of the log-likelihood function, and lastly run the second derivative test to show the critical point is a maximum. First off, we will compute the likelihood function as shown below.

$$\begin{aligned} L(\sigma) &= f(x_1) \times f(x_2) \times \cdots \times f(x_n) \\ &= \frac{1}{(2\pi\sigma^2)^{1/2}} e^{\frac{(x_1-\mu_0)^2}{2\sigma^2}} \times \frac{1}{(2\pi\sigma^2)^{1/2}} e^{\frac{(x_2-\mu_0)^2}{2\sigma^2}} \times \cdots \times \frac{1}{(2\pi\sigma^2)^{1/2}} e^{\frac{(x_n-\mu_0)^2}{2\sigma^2}} \\ &= \frac{1}{(2\pi\sigma^2)^{n/2}} e^{\frac{\sum_{i=1}^{n}(x_i-\mu_0)^2}{2\sigma^2}}, \ \sigma \geq 0 \end{aligned}$$

Thus, as computed above, the likelihood function of σ is $L(\sigma) = \frac{1}{(2\pi\sigma^2)^{n/2}} e^{\frac{\sum_{i=1}^{n}(x_i-\mu_0)^2}{2\sigma^2}},\ \sigma \geq 0$. However, notice that taking the derivative of this function will not be easy, thus we will take the natural log of the likelihood function to turn multiplication into addition. This process is shown below.

$$\begin{aligned}
\ell(\sigma) &= \ln(L(\sigma)) \\
&= \ln\left(\frac{1}{(2\pi\sigma^2)^{n/2}} e^{\frac{\sum_{i=1}^{n}(x_i-\mu_0)^2}{2\sigma^2}}\right) \\
&= \ln\left(\frac{1}{(2\pi\sigma^2)^{n/2}}\right) + \ln\left(e^{\frac{\sum_{i=1}^{n}(x_i-\mu_0)^2}{2\sigma^2}}\right) \\
&= \ln(1) - \ln\left(\left(2\pi\sigma^2\right)^{n/2}\right) - \frac{\sum_{i=1}^{n}\left(x_i-\mu_0\right)^2}{2\sigma^2} \\
&= \frac{-n}{2}\ln\left(2\pi\sigma^2\right) - \frac{\sum_{i=1}^{n}\left(x_i-\mu_0\right)^2}{2\sigma^2} \\
&= \frac{-n}{2}\ln\left(2\pi\right) - \frac{n}{2}\ln\left(\sigma^2\right) - \frac{\sum_{i=1}^{n}\left(x_i-\mu_0\right)^2}{2\sigma^2} \\
&= \frac{-n}{2}\ln\left(2\pi\right) - n\ln\left(\sigma^2\right) - \frac{\sum_{i=1}^{n}\left(x_i-\mu_0\right)^2}{2\sigma^2},\ \sigma \geq 0
\end{aligned}$$

As computed above, the log-likelihood function is $\ell(\sigma) = \frac{-n}{2}\ln\left(2\pi\right) - n\ln\left(\sigma^2\right) - \frac{\sum_{i=1}^{n}(x_i-\mu_0)^2}{2\sigma^2},\ \sigma \geq 0$. Now we must find the critical points of this function in order to find the candidates for the maximum likelihood estimator of σ. This derivative calculation is shown below.

$$\begin{aligned}
\frac{d}{d\sigma}\ell(\sigma) &= \frac{d}{d\sigma}\left(\frac{-n}{2}\ln\left(2\pi\right) - n\ln\left(\sigma^2\right) - \frac{\sum_{i=1}^{n}\left(x_i-\mu_0\right)^2}{2\sigma^2}\right) \\
&= \frac{-n}{\sigma} + \frac{\sum_{i=1}^{n}\left(x_i-\mu_0\right)^2}{\sigma^3},\ \sigma \geq 0
\end{aligned}$$

In order to find the critical points, we must find the values of σ such that this derivative is equal to zero.

$$\begin{aligned}
0 &= \frac{-n}{\sigma} + \frac{\sum_{i=1}^{n}\left(x_i-\mu_0\right)^2}{\sigma^3} \\
\frac{n}{\sigma} &= \frac{\sum_{i=1}^{n}\left(x_i-\mu_0\right)^2}{\sigma^3} \\
\sigma^2 &= \frac{\sum_{i=1}^{n}\left(x_i-\mu_0\right)^2}{n} \\
\sigma &= \sqrt{\frac{\sum_{i=1}^{n}\left(x_i-\mu_0\right)^2}{n}}
\end{aligned}$$

As computed above, the critical point $\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i-\mu_0)^2}{n}}$ is a candidate for the maximum likelihood estimator for σ we will now compute the second derivative test below to show that this critical point is in fact a maximum.

$$\begin{aligned}
\frac{d^2}{d\sigma^2}\ell(\sigma) &= \frac{d^2}{d\sigma^2}\left(\frac{-n}{\sigma} + \frac{\sum_{i=1}^{n}\left(x_i-\mu_0\right)^2}{\sigma^3}\right) \\
&= \frac{n}{\sigma^2} - \frac{3\sum_{i=1}^{n}\left(x_i-\mu_0\right)^2}{\sigma^4},\ \sigma \geq 0
\end{aligned}$$

Plugging in our value of σ into the second derivative of the log-likelihood function we obtain $\frac{n}{\left(\sqrt{\frac{\sum_{i=1}^{n}(x_i-\mu_0)^2}{n}}\right)^2}-\frac{3\sum_{i=1}^{n}(x_i-\mu_0)^2}{\left(\sqrt{\frac{\sum_{i=1}^{n}(x_i-\mu_0)^2}{n}}\right)^4}=\frac{n}{\frac{\sum_{i=1}^{n}(x_i-\mu_0)^2}{n}}-\frac{3\sum_{i=1}^{n}(x_i-\mu_0)^2}{\frac{\sum_{i=1}^{n}(x_i-\mu_0)^4}{n^2}}=\frac{n^2}{\sum_{i=1}^{n}(x_i-\mu_0)^2}-\frac{3n^2}{\sum_{i=1}^{n}(x_i-\mu_0)^2}=\frac{-2n^2}{\sum_{i=1}^{n}(x_i-\mu_0)^2}$. Since $n>0$, $\sum_{i=1}^{n}(x_i-\mu_0)^2>0$, it follows that $\frac{-2n^2}{\sum_{i=1}^{n}(x_i-\mu_0)^2}<0, \quad \forall\sigma\geq 0$. Since $\frac{-2n^2}{\sum_{i=1}^{n}(x_i-\mu_0)^2}<0, \quad \forall\sigma\geq 0$, it follows that $\sigma=\sqrt{\frac{\sum_{i=1}^{n}(x_i-\mu_0)^2}{n}}$ is a local maximum. However, since $\sigma=\sqrt{\frac{\sum_{i=1}^{n}(x_i-\mu_0)^2}{n}}$ is the only critical point, it turns out that it is in fact a global maximum. Hence we have shown that $\widehat{\sigma}^{mle}=\sqrt{\frac{\sum_{i=1}^{n}(x_i-\mu_0)^2}{n}}$ is the MLE.