Logistic Regression - Chapter 9

AUTHOR
Aaron Younger

BUSINESS UNDERSTANDING

The marketing department of a grocery retailer wants to understand the factors that contributed to an email campaign response. They would like to use this model to select customers for future similar email campaigns.

DATA UNDERSTANDING AND DATA PREP

Each row represents a customer who received an email as part of a campaign (tidy format).

Variables response = binary indicating whether the customer responded to the email campaign (made a purchase)
recency = number of days since last purchase prior to email campaign start
frequency = number of purchases (all channels) in the last year prior to the start of the email campaign
total_spend = total dollars spent by customer customer (all changes) in last year prior to campaign
NumWebVisits = number of website visits in the month prior to the campaign
kidhome = number of kids in the home

```r
## Libraries
library(readxl)
library(tidyverse)
library(janitor)
library(DataExplorer)
library(caret)
library(ggplot2)
```

```r
options(scipen = 9999, digits=6)

load("retail_DAT4253.RData")

retail_data <- clean_names(retail)
View(retail_data)
```

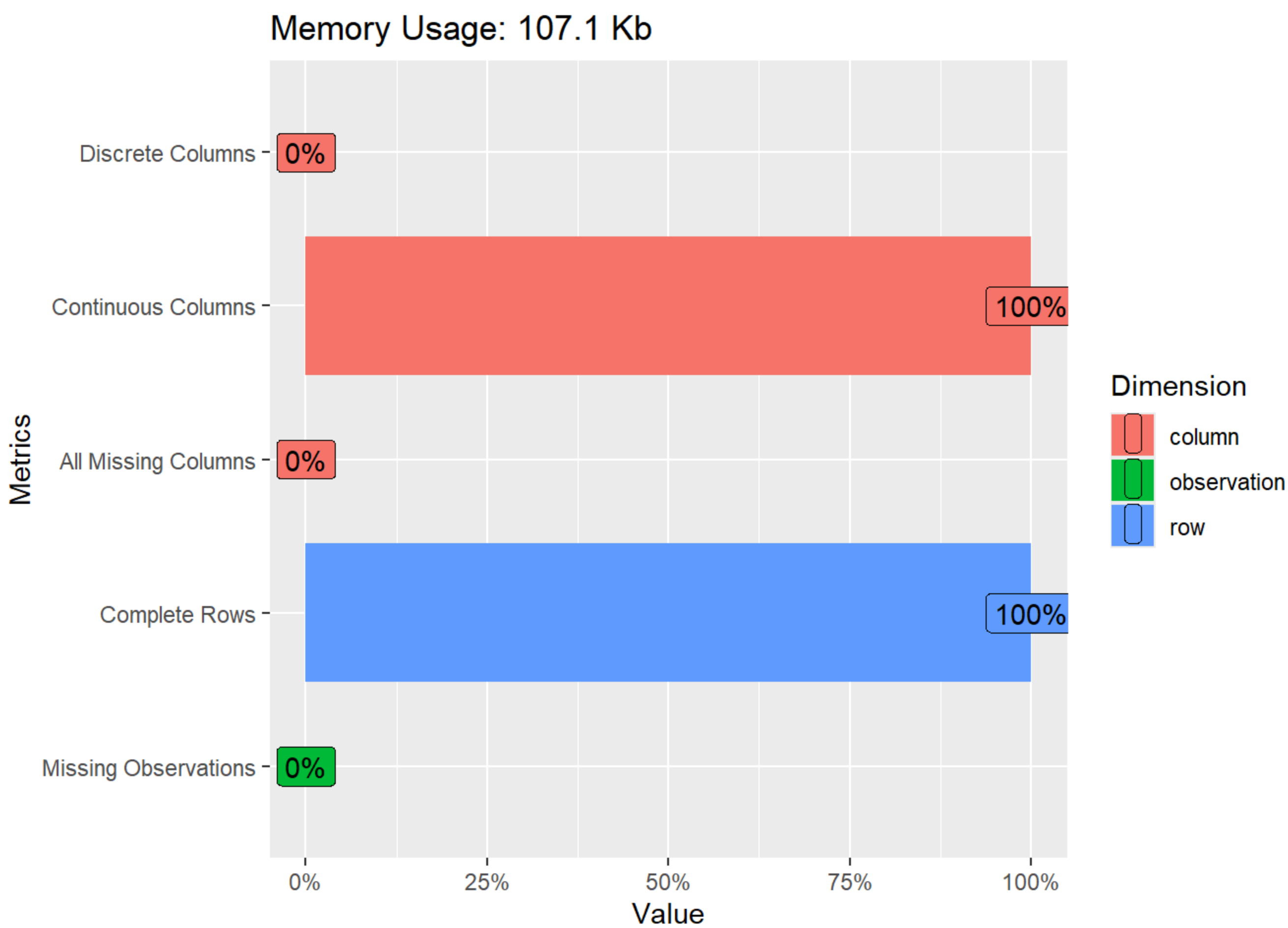
Basic EDA

Conduct basic EDA using `DataExplorer`
Specifically look at barcharts, histograms, and the numerics by the depvar in boxplots.

```r
## Dataset EDA

retail_data %>% str()
```

```
tibble [2,240 × 6] (S3: tbl_df/tbl/data.frame)
 $ response           : num [1:2240] 1 0 0 0 0 0 0 0 1 0 ...
 $ frequency          : num [1:2240] 25 6 21 8 19 22 21 10 6 2 ...
 $ recency            : num [1:2240] 58 38 26 26 94 16 34 32 19 68 ...
 $ total_spend        : num [1:2240] 1617 27 776 53 422 ...
 $ kidhome            : num [1:2240] 0 1 0 1 1 0 0 1 1 1 ...
 $ num_web_visits_month: num [1:2240] 7 5 4 6 5 6 6 8 9 20 ...
```

```r
retail_data %>% plot_intro()
```




```r
retail_data %>% head()
```

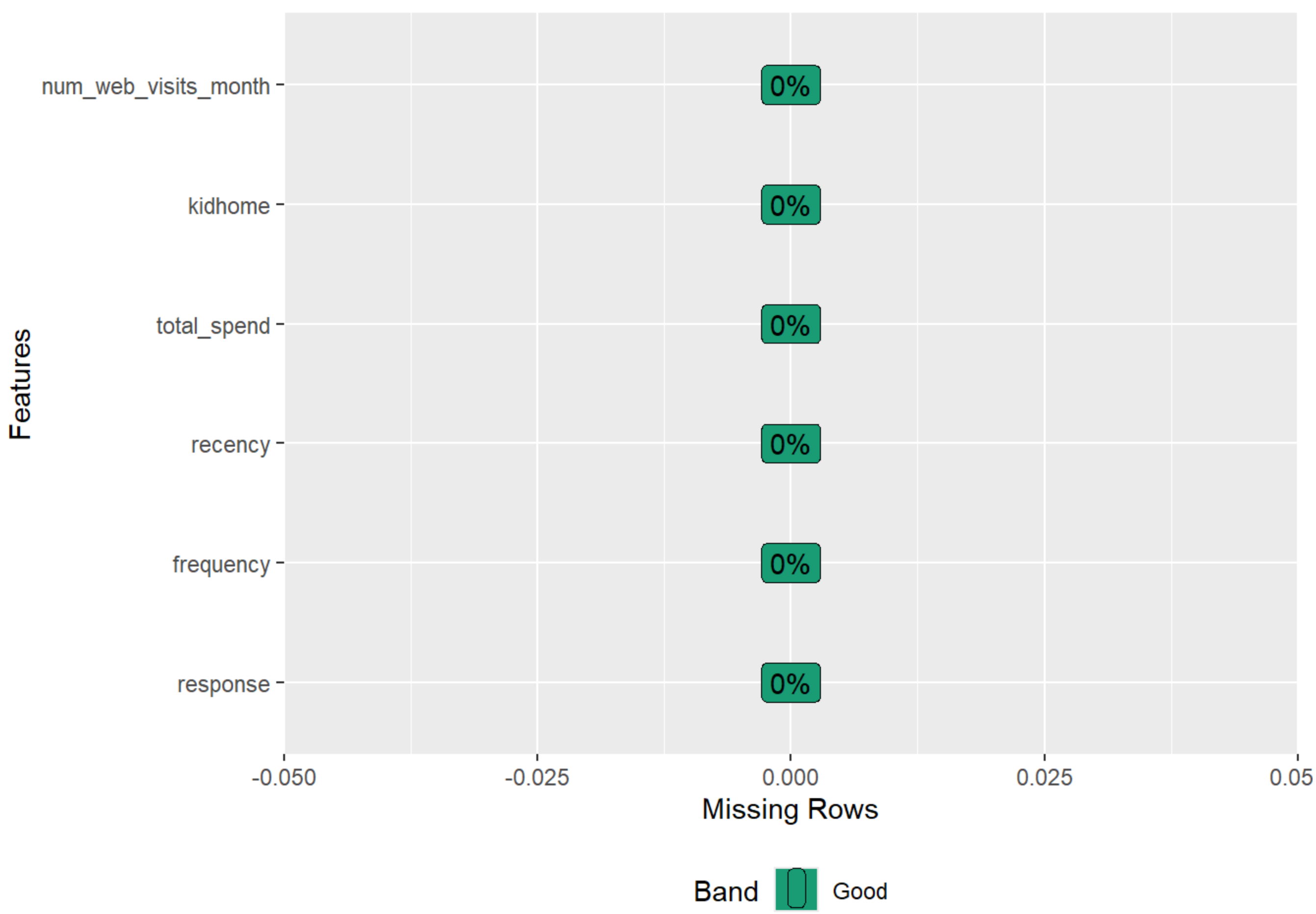
```
# A tibble: 6 × 6
  response frequency recency total_spend kidhome num_web_visits_month
     <dbl>     <dbl>   <dbl>       <dbl>   <dbl>                <dbl>
1        1        25      58        1617       0                    7
2        0         6      38          27       1                    5
3        0        21      26         776       0                    4
```

```
4         0          8       26           53        1                      6
5         0         19       94          422        1                      5
6         0         22       16          716        0                      6
```

```
retail_data %>% tail()
```

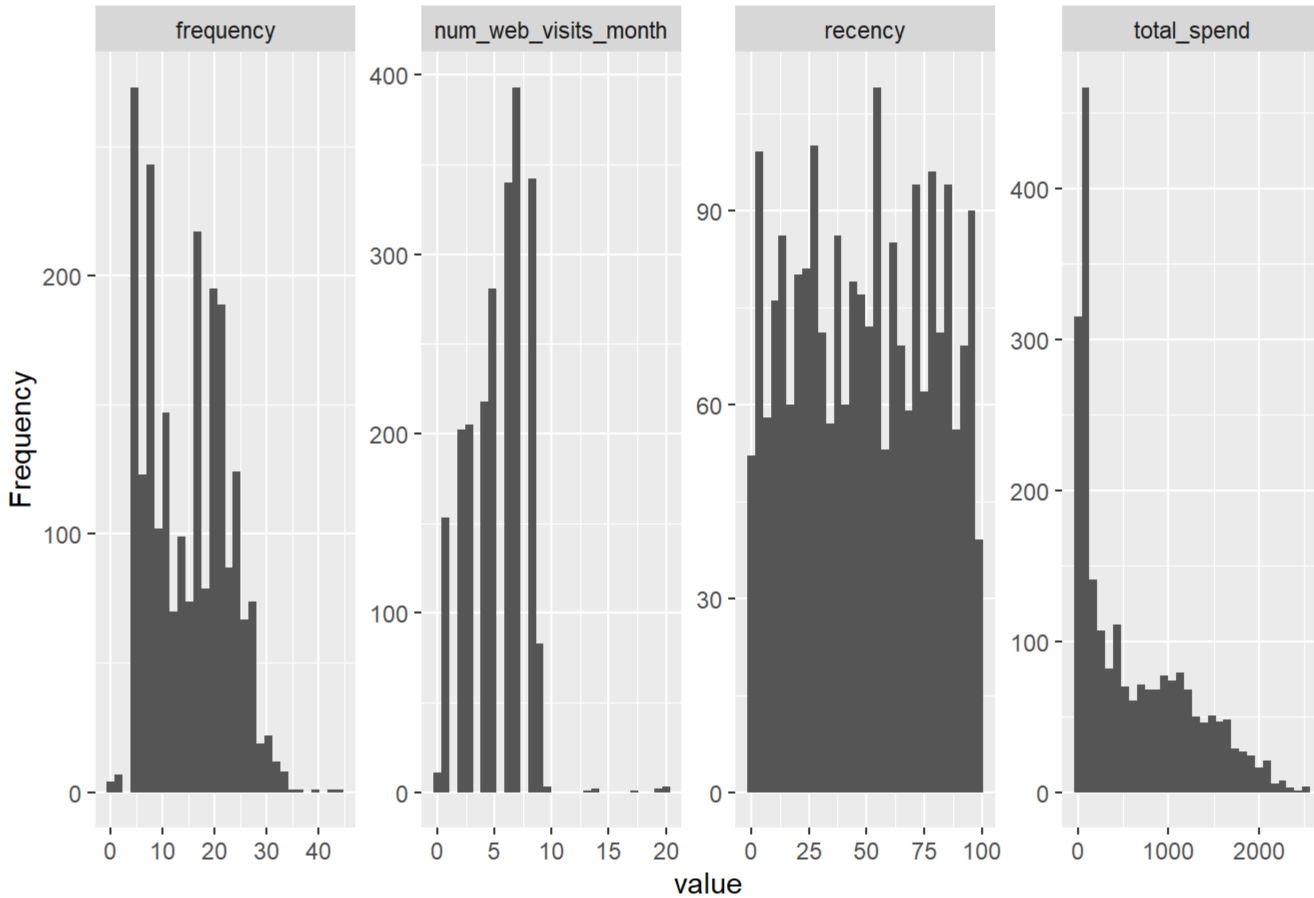
```
# A tibble: 6 × 6
  response frequency recency total_spend kidhome num_web_visits_month
     <dbl>     <dbl>   <dbl>       <dbl>   <dbl>                <dbl>
1        0         4      81          30       1                    7
2        0        18      46        1341       0                    5
3        0        22      56         444       2                    7
4        0        19      91        1241       0                    6
5        0        23       8         843       0                    3
6        1        11      40         172       1                    7
```

```
retail_data %>% plot_missing() ## No missing values in this dataset
```

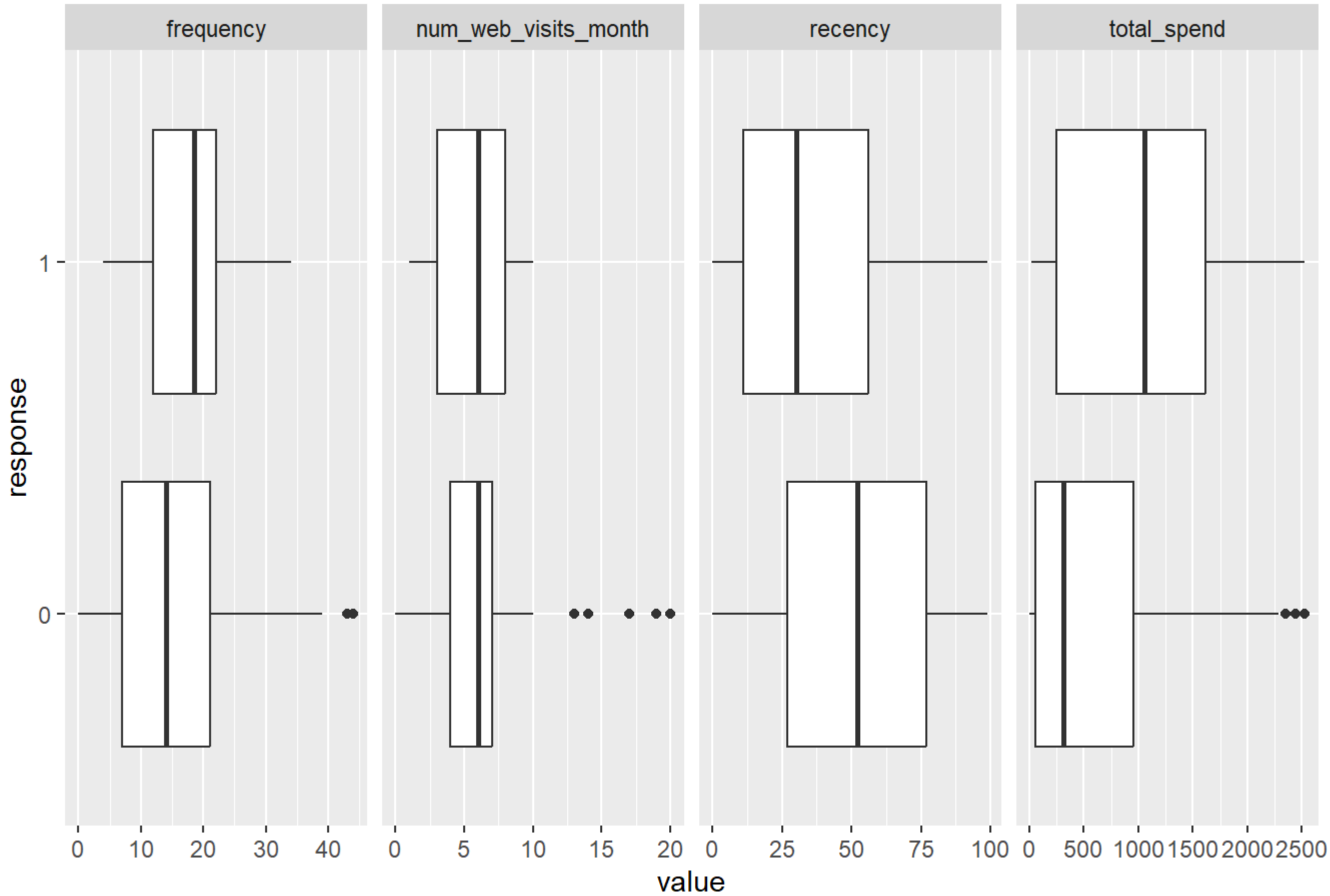



```
## Variable EDA

## Change data type for certain variables
retail_data$response <- as.factor(retail_data$response)
```

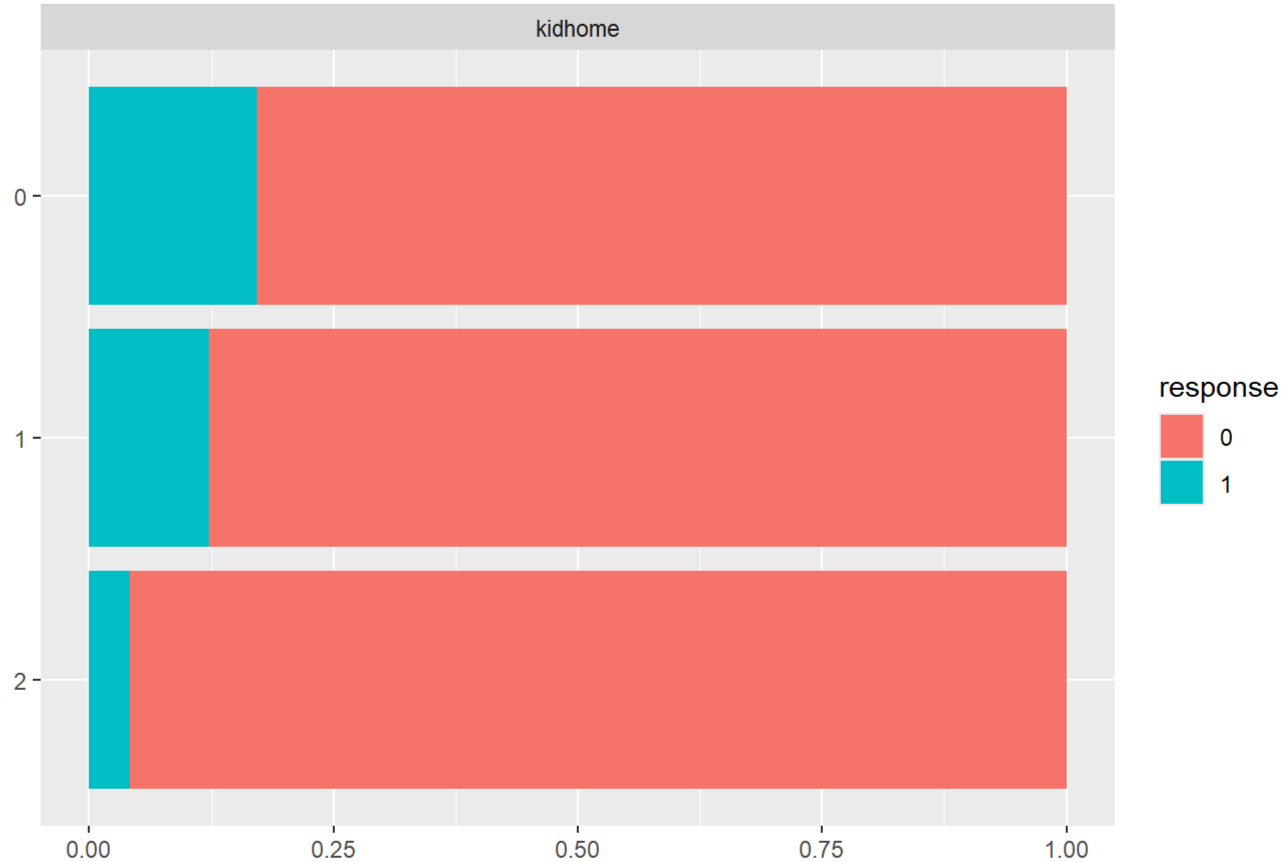
```
retail_data$kidhome <- as.factor(retail_data$kidhome)


retail_data %>% plot_histogram()
```




```
retail_data %>% plot_boxplot(by="response")
```




```
retail_data %>% plot_bar(by = "response")
```

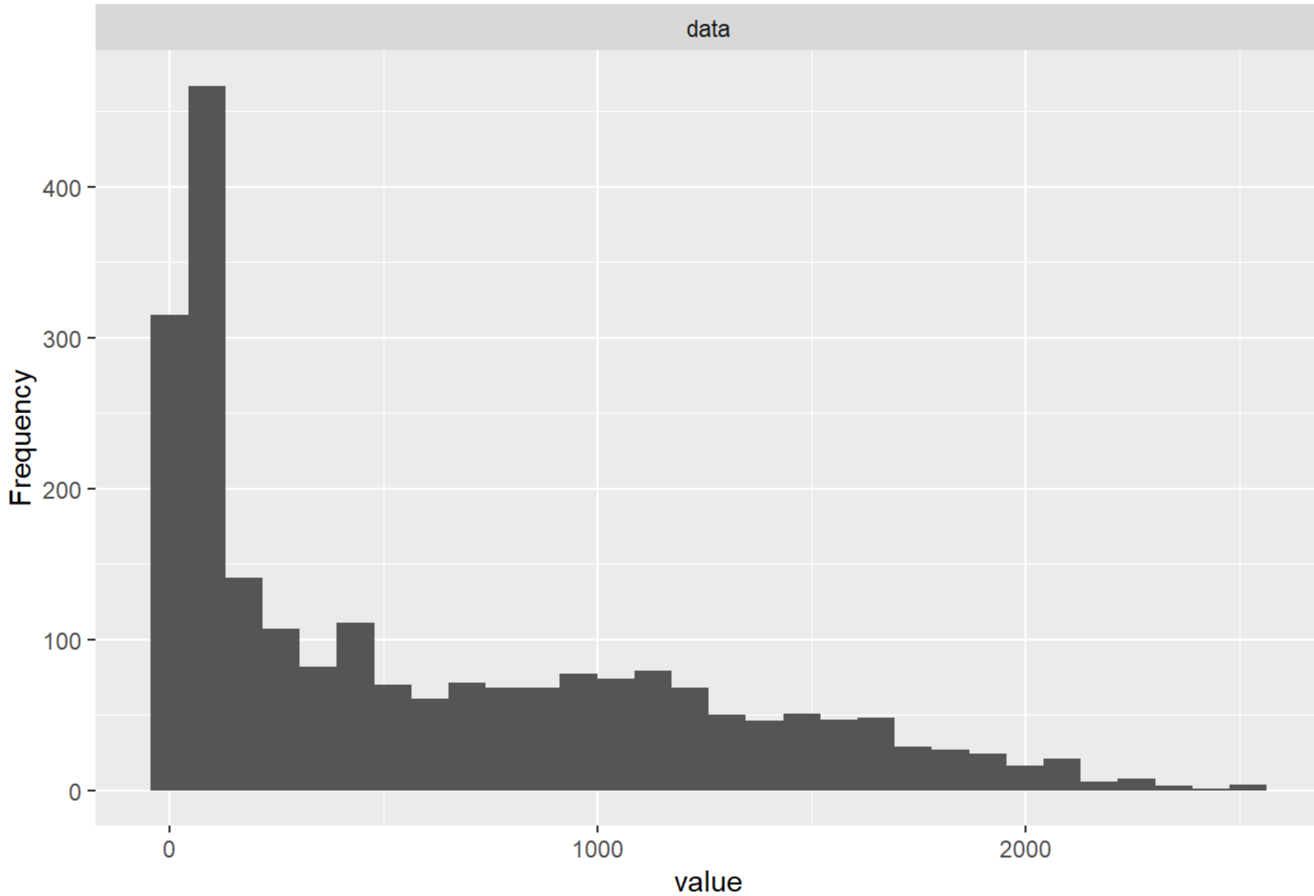


Comment: This dataset has six variables with one of those variables being the dependent variable, response. This dataset has no missing values. The distribution of spend is right skewed and will need to be addressed. Based off of box plots number of website visits and frequency have a few outliers. The Kid at home variable has all cases of the dependent variable, meaning there is both yes and no responses for all number of kids in the kids home variable. Their are instances of 0, 1, and 2 kids at home in the kids home variable.

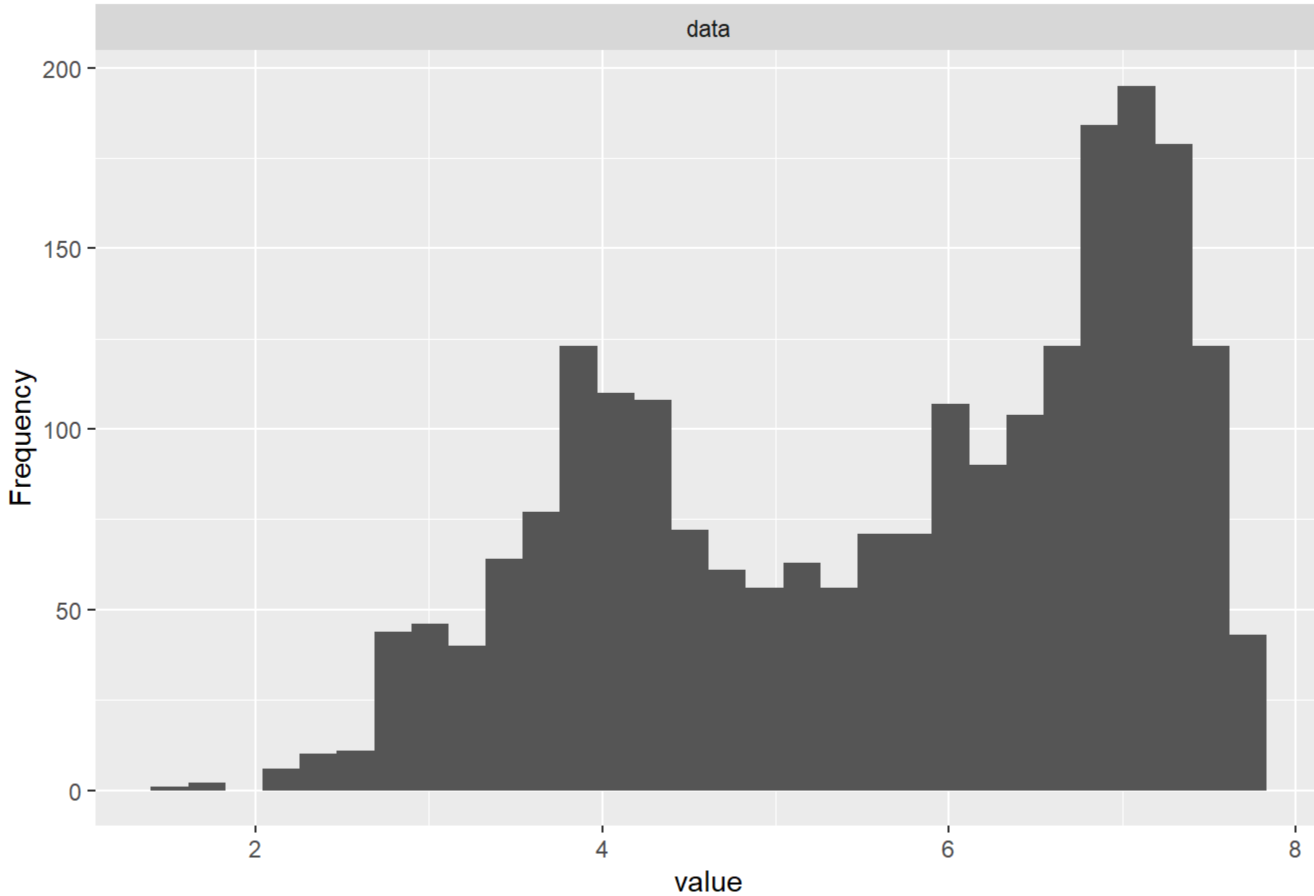
Munge: Variable Transformation

Total_Spend has a wide distribution. Create a new variable that is the log of Total Spend, and then delete (drop) the original Total_Spend from the dataset. Such a transformation is good practice for a sales for dollar variable.

```
## Distribution of Spend
## Spend is not normally distributed, it is right skewed
retail_data$total_spend %>%  plot_histogram()
```




```r
retail_data$log_total_spend <- log(retail_data$total_spend)
retail_data <- retail_data %>%
  select(-total_spend)
View(retail_data)

retail_data$log_total_spend %>% plot_histogram()
```



Comment: To deal with the right skewness of total spend, the log of total spend was taken. This helped reduce the influence of extreme spenders, making the distribution more symmetric and more reliable for the model.

Summary Statistics

Run descriptive statistics and a correlation matrix. Comment on your findings.

```
library(skimr)
skim(retail_data)
```

Data summary

Name	retail_data
Number of rows	2240
Number of columns	6
_______________________	
Column type frequency:	
factor	2

numeric	4

Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
response	0	1	FALSE	2	0: 1906, 1: 334
kidhome	0	1	FALSE	3	0: 1293, 1: 899, 2: 48

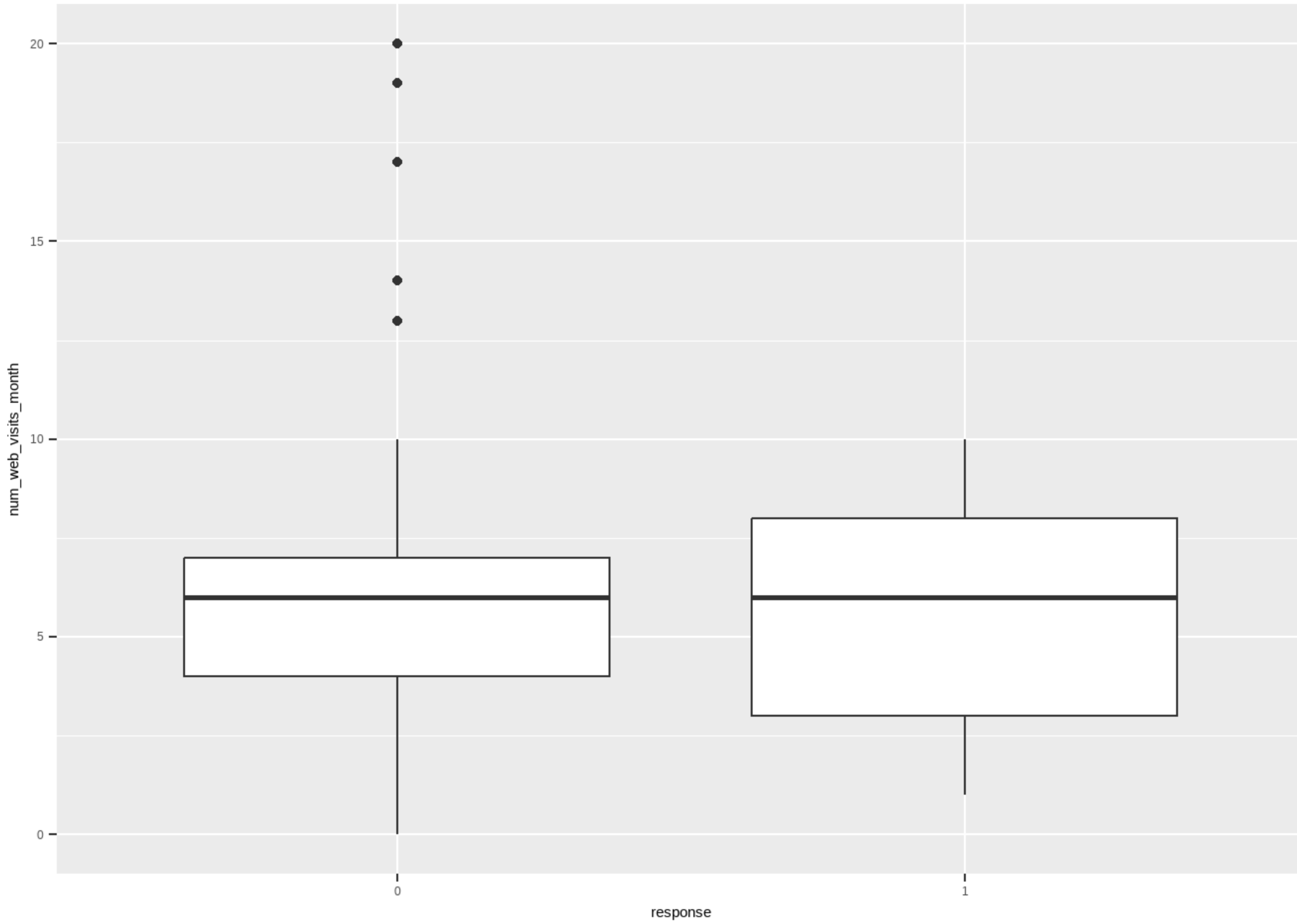
Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
frequency	0	1	14.86	7.68	0.00	8.00	15.00	21.00	44.00	
recency	0	1	49.11	28.96	0.00	24.00	49.00	74.00	99.00	
num_web_visits_month	0	1	5.32	2.43	0.00	3.00	6.00	7.00	20.00	
log_total_spend	0	1	5.61	1.48	1.61	4.23	5.98	6.95	7.83	

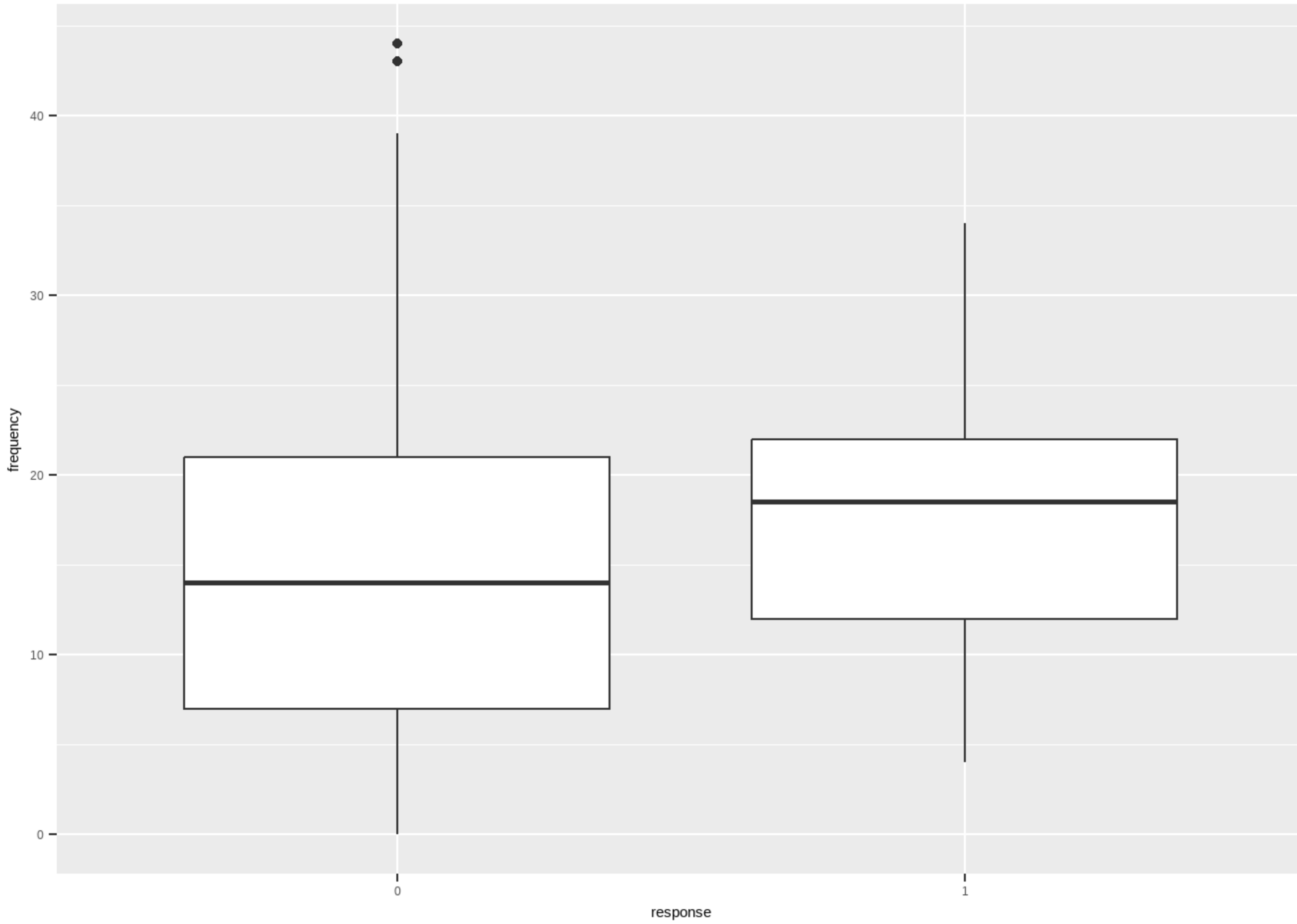
```r
##outliers
library(dlookr)
retail_data %>% diagnose_outlier
```

```
# A tibble: 4 × 6
  variables     outliers_cnt outliers_ratio outliers_mean with_mean without_mean
  <chr>                <int>          <dbl>         <dbl>     <dbl>        <dbl>
1 frequency                2         0.0893          43.5     14.9         14.8
2 recency                  0         0              NaN       49.1         49.1
3 num_web_visi…            8         0.357           17.9      5.32         5.27
4 log_total_sp…            0         0              NaN        5.61         5.61
```

```r
## Visualize outliers
ggplot(retail_data, aes(x=response, y = num_web_visits_month)) +
  geom_boxplot()
```



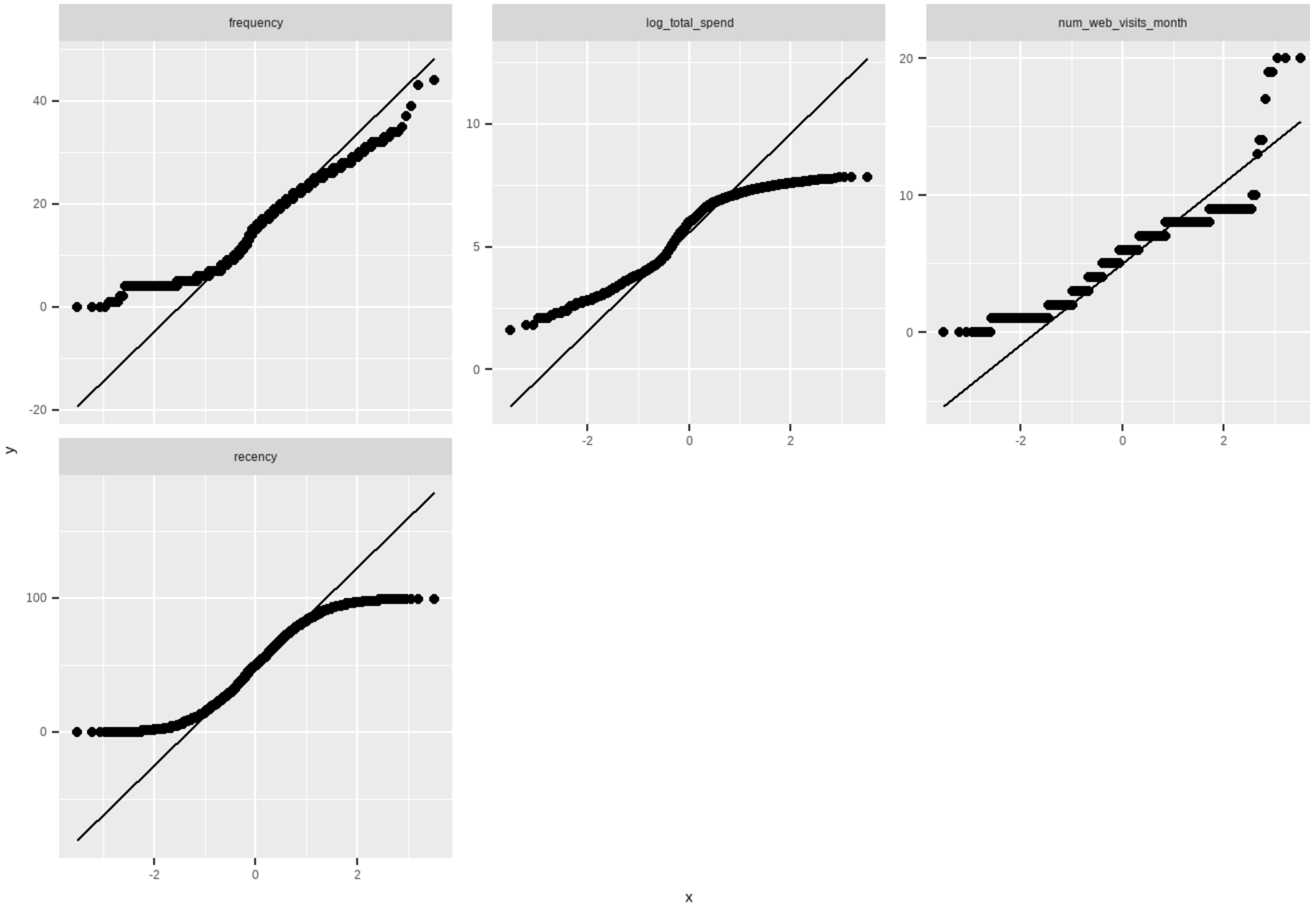

```r
ggplot(retail_data, aes(x=response, y = frequency))+
  geom_boxplot()
```



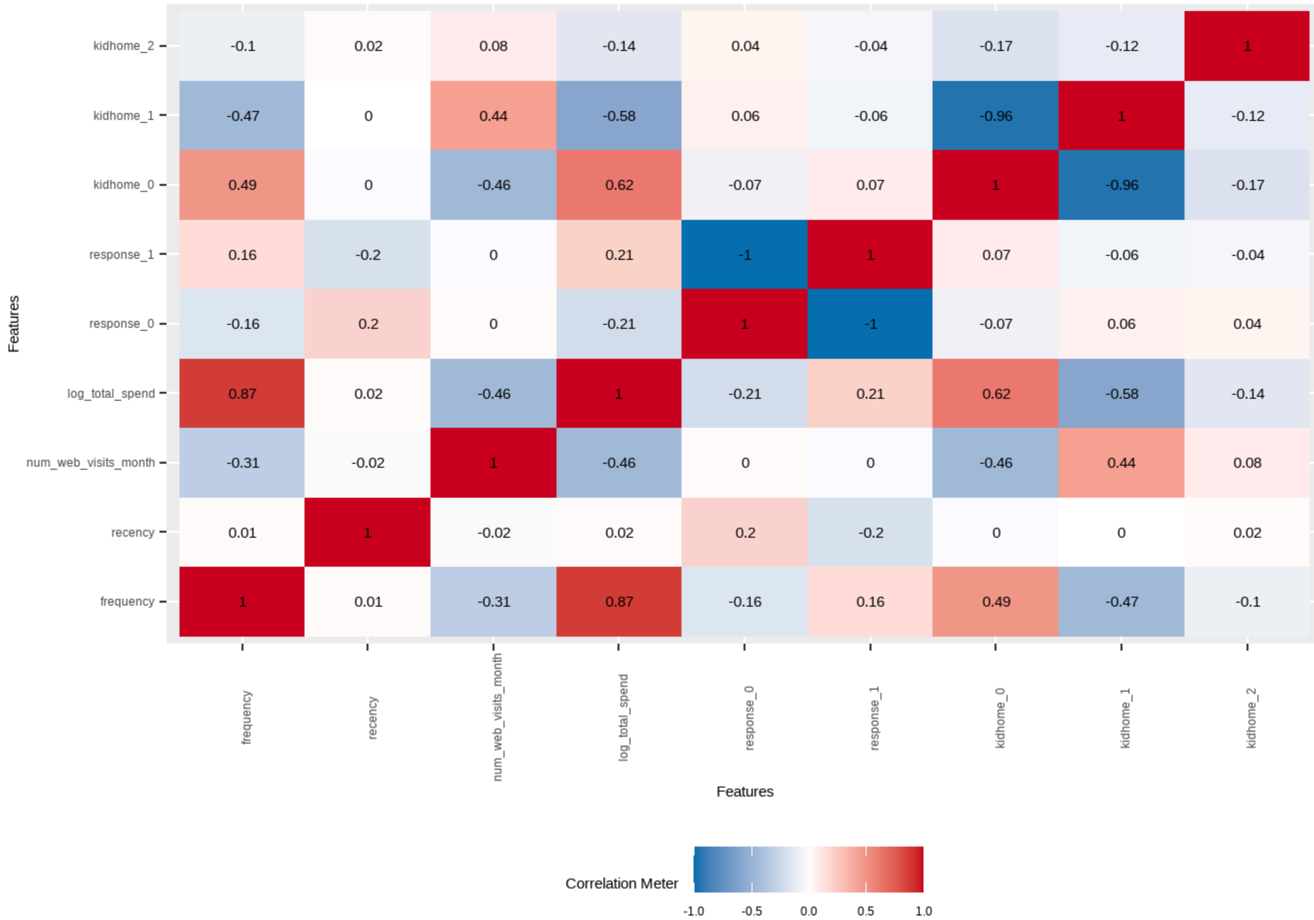

```
## Mean of independent variables by depvar
retail_data %>%
  group_by(response) %>%
  summarise(mean_frequency = mean(frequency),
            mean_TotSpend = mean(log_total_spend),
            mean_webvis = mean(num_web_visits_month),
            mean_rec = mean(recency),
            mean_kidshome = mean(as.numeric(kidhome)))
```

```
# A tibble: 2 × 6
  response mean_frequency mean_TotSpend mean_webvis mean_rec mean_kidshome
  <fct>             <dbl>         <dbl>       <dbl>    <dbl>         <dbl>
1 0                  14.4          5.48        5.32     51.5          1.46
2 1                  17.7          6.37        5.29     35.4          1.34
```

```
## Normality qq-plots
retail_data %>% plot_qq()
```




```r
## Correlation of independent by dependent variables
retail_data %>% plot_correlation()
```



Comment:

Assess depvar balance

Create a table that shows the prevalence of the depvar, response, in the dataset. Comment on your findings.

Outliers for numeric variables were looked at. There were only two variables that had outliers, freqeuncy and number of website visits. The affect that the outliers had on both these variables was very minimal so outliers were kept in the dataset. The mean was then found for each variable relating to the two cases of the dependent variable. For these metrics it looked like a higher frequency, total spend, and lower recency led to a yes (1) response. Normality was then checked among the numeric variables using a qq plot. No numeric variable was linear as each numeric variable had some bending at the tails. Lastly, a correlation plot was made to see correlation of variables to the dependent variable. This highlighted that frequency and total spend had a weak positive correlation to yes (1) responses and recency had a weak negative correlation to yes (1) responses.

```
cat("Proportion of Responses \n")
```

```
Proportion of Responses
```

```
prop.table(table(retail_data$response)) * 100
```

```
      0       1
85.0893 14.9107
```

Comment: The dependent variable is imbalanced in this dataset, with approximately 85% of responses being 0 and approximately 15% of responses being 1. This imbalance is important to note, as it can bias the models predicting power and lead to misleading overall accuracy.

Partition 60/40

Since our dataset is unbalanced, check our partitions to see that they're similar.
In this case we are using set.seed to 4.
Check the partitions for balance compared to the full dataset. They should be close.

```
library(caret)
set.seed(4)
myIndex <- createDataPartition(retail_data$response, p=0.6, list = FALSE)
trainset <- retail_data[myIndex,]
testset <- retail_data[-myIndex,]
prop.table(table(trainset$response))
```

```
       0        1
0.850558 0.149442
```

```
prop.table(table(testset$response))
```

```
       0        1
0.851397 0.148603
```

```
## compare to original dataset
cat("Proportion of Responses \n")
```

```
Proportion of Responses
```

```
prop.table(table(retail_data$response)) * 100
```

```
      0       1
85.0893 14.9107
```

Comment: Before modeling I partitioned the dataset into a 60/40 split. 60 percent of the data was put into the training set and 40% was put into the test set. Since this dataset is imbalanced I checked to make sure the partitions were balanced close to the full dataset. The Partitions were very close to the balance of the original dataset.

MODELING

Logistic Regression Model

Use the `caret` package to train a LR model using 10-fold cross-validation.
Look at the output and check for multicollinearity.

```
myCtrl <- trainControl(method="cv", number=10)
set.seed(4)
logit <- train(as.factor(response)~., data=trainset, method="glm", family="binomial", trControl
         = myCtrl)
summary(logit)
```

```
Call:
NULL

Coefficients:
                      Estimate Std. Error z value             Pr(>|z|)    
(Intercept)           -7.49924    0.78965   -9.50 < 0.0000000000000002 ***
frequency             -0.06865    0.02036   -3.37              0.00075 ***
recency               -0.02238    0.00302   -7.40   0.0000000000001339 ***
kidhome1               0.73031    0.22342    3.27              0.00108 ** 
kidhome2             -12.98386  428.60250   -0.03              0.97583    
num_web_visits_month   0.16741    0.03903    4.29   0.0000179454760621 ***
log_total_spend        1.12415    0.14128    7.96   0.0000000000000018 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1134.48  on 1344  degrees of freedom
Residual deviance:  964.74  on 1338  degrees of freedom
AIC: 978.7

Number of Fisher Scoring iterations: 15
```

```
library(car)
vif(logit$finalModel)
```

```
           frequency              recency             kidhome1 
             2.74284              1.01199              1.68278 
```

```
           kidhome2 num_web_visits_month      log_total_spend
            1.00000              1.50809              4.06506
```

Comment:
The model showed that several of the predictor variables were significant. Those being frequency, recency, 1 kid at home, number of website visits, and log of total spend. Multicollinearity was also checked and log of total spend had a high vif value meaning it could be affected by multicollinearity.

Calculate odds ratios

Interpret the odds ratio for all five predictors.

```
table <- tibble(
  predictors = names(coef(logit$finalModel)),
  odds_ratio = exp(coef(logit$finalModel)),
  p_value    = round(summary(logit$finalModel)$coefficients[, "Pr(>|z|)"], 4))
knitr::kable(table, caption = "Baseline Logistic Regression Measures")
```

Baseline Logistic Regression Measures

predictors	odds_ratio	p_value
(Intercept)	0.000554	0.0000
frequency	0.933652	0.0007
recency	0.977865	0.0000
kidhome1	2.075720	0.0011
kidhome2	0.000002	0.9758
num_web_visits_month	1.182235	0.0000
log_total_spend	3.077601	0.0000

Fully interpret each of the coefficients. Do they make sense? Explain.

Odds Ratio to outcome (response):

Odds Ratio = 1 -> no effect on outcome Odds Ratio > 1 -> Predictor increases the odds of the outcome happening, positive effect Odds Ratio < 1 -> Predictor decreases the odds of the outcome happening, negative effect

Odds Ratio is calculated by exponentiation the regression coefficients. response = binary indicating whether the customer responded to the email campaign (made a purchase)
recency = number of days since last purchase prior to email campaign start
frequency = number of purchases (all channels) in the last year prior to the start of the email campaign
total_spend = total dollars spent by customer customer (all changes) in last year prior to campaign
NumWebVisits = number of website visits in the month prior to the campaign
kidhome = number of kids in the home

Frequency: The odds ratio number for frequency is 0.933652. Since the odds ratio is < 1 frequency has a negative effect on response happening. For each unit increase in frequency there is a 6.6% decrease in odds of response. (1-0.934) * 100 = 6.6%. Since response is measured by whether a customer made a purchase it is suprising that frequency does not have a positive effect on response. The negative affect could suggest that customers with already high buying frequency are less responsive to additional marketing.

Recency: The odds ratio number for recency is 0.977865. Since the odds ratio is < 1 recency has a negative effect on response happening. For each unit increase in recency there is a 2.2% decrease in the odds of response. (1-0.978) * 100 = 2.2%. Recency's effect on response makes sense. The longer its been since a customer's last purchase, the less likely they are to respond to an email.

Kids at home: The odds ratio for kids at home is calculated using no kids at home as a baseline. The odds ratio for 1 kid at home is 2.075720. This means that houses with 1 kid have double the odds to respond compared to houses with no kids. This is significant as it shows who to focus on when marketing who to send these emails to. However houses with 2 kids show no affect to the odds of response when compared to houses with zero kids. The odds ratio for 0 kids is 0.000002 and this result is not statistically significant as the associated p-value is 0.9758. Households with zero kids served as the reference group.

Web Visits: The odds ratio for Web Visits is 1.182235. Since the odds ratio is > 1 web visits has a positive effect on response happening. For each unit increase in web visits there is an 18.2% increase in the odds of response. (1.182-1) * 100 = 18.2%. This makes sense as higher web visits should increase the likelihood of a customer responding.

Log of Total Spend: The odds ratio for total spend is 3.077601. Since the odds ratio is > 1 total spend has a positive effect on response happening. For each one unit increase in total spend the odds of a customer responding multiplies by about 3x. This makes sense as a business would expect higher paying customers more likely to respond.

Variable Importance

```r
library(caret)
caret::varImp(logit$finalModel)
```

```
                      Overall
frequency           3.3721584
recency             7.4022250
kidhome1            3.2687823
kidhome2            0.0302935
num_web_visits_month 4.2890305
log_total_spend     7.9567613
```

Comment: Variable Importance ranks predictive power of predictors to response (dependent variable). The Variable Importance of this model shows Total Spend and Recency are the strongest predictors to customer response whereas kidhome2 shows almost no predictive power in this model.

Assess LR Model Fit

Global likelihood ratio test (LRT)

Run the LRT test. Similar to the example, you will need to re-run the logit without cross validation just for this test, because `caret` doesn't store everything you need.

```
logit_full <- glm(response ~., family = binomial(link="logit"), data = trainset)
logit_null <- glm(response ~ 1, family = binomial(link="logit"), data = trainset)

anova(logit_full, logit_null, test="LRT")
```

```
Analysis of Deviance Table

Model 1: response ~ frequency + recency + kidhome + num_web_visits_month +
    log_total_spend
Model 2: response ~ 1
  Resid. Df Resid. Dev Df Deviance            Pr(>Chi)
1      1338      964.7
2      1344     1134.5 -6   -169.7 <0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Comment: The likelihood ratio test (LRT) is a test of model fit. The LRT compares two models, one model with just the intercept and the other model with predictors. In this case the predictors as a group significantly improve model fit compared to just the intercept. The low p value support the predictors significantly improve model fit.

Model Fit Metrics

Again, `caret` doesn't retain content we need to calculate PseudoR2, so use the output you created above using glm().

```
library(DescTools)
table1 <- tibble(
  McFadden = formattable::comma(PseudoR2(logit_full, "McFadden"), digits = 4),
  Nagel = formattable::comma(PseudoR2(logit_full, "Nagel"), digits = 4),
  AIC = logit$finalModel$aic
)
knitr::kable(table1, caption="Goodness of fit measures \n")
```

Goodness of fit measures

McFadden	Nagel	AIC
0.1496	0.2081	978.738

Comment on each of the tests metrics: **McFadden:** For this logistic regression model the McFadden value is 0.1496 or approximately 0.15. The McFadden variable measures how much better a model explains data compared to a null model. An excellent fit for a logistic regression model is indicated by a McFadden value from 0.2-0.4. So the value of 0.15 indicates the model has a moderate fit.

Nagelkerke: For this logistic regression model the Nagelkerke value is 0.2081 or approximately 0.21. This measure is an adjusted version of the McFadden measure. This measure is on a scale of 0-1. A value of 0.21 suggests the model accounts for about 21% of the variation in the outcome. This model does have some explanatory power but their is still variation unexplained. **AIC:** This value by itself is meaningless as it is a value used to cross compare models. A lower AIC number indicates a better model.

Comparing probabilities

What are the probabilities that a customer will respond to an email campaign if they had a recency=0, frequency=10, 5 web visits, log of total spend = 5.6, and either 0 OR 1 kid at home?

!!Note that you will need to use income=as.numeric(c(NA,NA)) for the income variable, since it is stored still as a predictor, even if it's not used in the model.

```
predict(logit_full, data.frame(kidhome= factor(c(0,1), levels = levels(trainset$kidhome)),
                               recency=c(0,0),
                               frequency=c(10,10),
                               num_web_visits_month=c(5,5),
                               log_total_spend=c(5.6,5.6),
                               income=as.numeric(c(NA,NA))),
                type="response")
```

```
       1        2 
0.258557 0.419903 
```

Comment: Holding predictors constant (recency = 0, frequency = 10, web visits = 5, log spend = 5.6, income missing) a household with no kids has a 26% chance of response whereas a houssehold with one kid has a 42% chance of response.

EVALUATION

Confusion Matrix – Default Cutoff of 50%

Create the CM at the default. Compute the F1 Score.

```
logit_predict_base <- predict(logit, newdata = testset, type="raw")
CM_default <- confusionMatrix(logit_predict_base, as.factor(testset$response), positive = '1')
CM_default
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
```

```
          0 754 125
          1   8   8

               Accuracy : 0.851
                 95% CI : (0.826, 0.874)
    No Information Rate : 0.851
    P-Value [Acc > NIR] : 0.523

                  Kappa : 0.078

 Mcnemar's Test P-Value : <0.0000000000000002

            Sensitivity : 0.06015
            Specificity : 0.98950
         Pos Pred Value : 0.50000
         Neg Pred Value : 0.85779
             Prevalence : 0.14860
         Detection Rate : 0.00894
   Detection Prevalence : 0.01788
      Balanced Accuracy : 0.52483

       'Positive' Class : 1
```

```r
precision <- unname(CM_default$byClass['Pos Pred Value'])
recall <- unname(CM_default$byClass['Sensitivity'])
f1_score_default <- 2 * ((precision * recall) / (precision + recall))
cat("F1 Score: ", f1_score_default, "\n")
```

```
F1 Score:  0.107383
```

Comment: It is important to note that this dataset was imbalanced and the majority class was 0, and the positive class for this confusion matrix is 1.

Model Performance:

Accuracy = 85.1%; this shows the proportion of all correct predictions made by the model. However, since the dataset is imbalanced, accuracy can be misleading.

95% CI: with 95% confidence, the true accuracy of the model lies between 82.6% and 87.4%.

No Information Rate = 0.851: This shows what the accuracy of the model would be if it always predicted the majority class, 0 (No Response). Since this rate is the exact same as the accuracy this model does not perform better than trivial guessing.

The P-Value = 0.523; This shows the model accuracy is not statistically significant better than the No Information Rate.

Kappa = 0.078; This indicates the model is barely performing better than random chance when adjusting for imbalanced data. Kappa helps reveal the model is not learning the minority class well.

Mcnemar's Test P-Value = <0.0000000000000002; This is a measure of significance to see if the model is making one kind of mistake more often. Since the p-value is very low this indicates that the model misclassifies towards one side, that being more false negatives than false positives. In other words the model will more likely miss actual 1's than incorrect 0's.

Class Specific Metrics
Sensitivity = 0.06; The model correctly identifies only 6% of actual positives. The model fails to capture the minority class well.

Specificity = 0.989; The model correctly identifies 99% of actual negatives. The model is heavily biased towards predicting the majority class.

Positive Predictive Value = 0.5; Of all the predicted positive values, only 50% were true positives. This shows weak precision for the positive class.

Negative Predictive Value = 0.858; Of all the predicted negative values, 86% were truly negative.

Prevalence = 0.149; About 15% of the dataset is the positive class '1', this further supports the dataset being imbalanced.

Detection Rate = 0.009; less than 1% of records were correctly predicted as the positive class.

Detection Prevalence = 0.018; About 1.8% of cases were predicted as positive which is much lower than the prevalence value ~ 15%.

Balanced Accuracy = 0.525; The average sensitivity and specificity, this shows weak balance compared to the Accuracy value due to the sensitivity being poor.

F1 Score = 0.107; The F1 score combines precision and recall into a single number which means this measure focuses on the positive class. A value of 0.107 suggests the model struggles to balance identifying positives and being accurate when it predicts positives.

Confusion Matrix – Threshold Tuning Method (optimal Cutoff Value)

If you think you have an issue with imbalance, complete this step to use the Threshold Tuning Method to address it.
Find the Optimal cutoff (due to depvar imbalance).
Recalculate the Confusion Matrix with the new cutoff value.

```
library(pROC)

# score the model to get the predicted probabilities (not the predicted class) from the model
logit_class_prob <- predict(logit, newdata = testset, type = 'prob')

# create the roc_curve (logit_class_prob has 2 columns, take the second one that has the
        probability of 1)
roc_curve <- roc(testset$response, logit_class_prob[,2])
```

```
# find and report the optimal cut off
optimal_cutoff <- coords(roc_curve, "best", ret = "threshold") #best uses Youden method
         max(sensitivity+specificity) to balance
cat("CONFUSION MATRIX AT OPTIMAL CUTOFF VALUE OF:", optimal_cutoff$threshold,"\n\n")
```

```
CONFUSION MATRIX AT OPTIMAL CUTOFF VALUE OF: 0.13988
```

```
# rescore using the new optimal cutoff
logit_class_opt <- ifelse(logit_class_prob[,2] >= optimal_cutoff$threshold, 1, 0)

# New Confusion Matrix
CM_base_opt <- confusionMatrix(as.factor(logit_class_opt), as.factor(testset$response), positive
         = '1')
CM_base_opt
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 501  30
         1 261 103
                                             
               Accuracy : 0.675              
                 95% CI : (0.643, 0.705)     
    No Information Rate : 0.851              
    P-Value [Acc > NIR] : 1                  
                                             
                  Kappa : 0.252              
                                             
 Mcnemar's Test P-Value : <0.0000000000000002
                                             
            Sensitivity : 0.774              
            Specificity : 0.657              
         Pos Pred Value : 0.283              
         Neg Pred Value : 0.944              
             Prevalence : 0.149              
         Detection Rate : 0.115              
   Detection Prevalence : 0.407              
      Balanced Accuracy : 0.716              
                                             
       'Positive' Class : 1                  
```

```
# Calculate the F1 Score
precision <- unname(CM_base_opt$byClass['Pos Pred Value']) # synonyms
recall    <- unname(CM_base_opt$byClass['Sensitivity'])   # synonyms
f1_score_base_opt <- 2 * ((precision * recall) / (precision + recall))
cat("F1 Score: ", f1_score_base_opt, "\n")
```

```
F1 Score:  0.414487
```

Comments: Threshold tuning with the optimal cut off value lowered model accuracy but improved the model’s class identification with an increases F1 score and sensitivity.

Confusion Matrix – Weighted Classes

```r
class_counts <- table(trainset$response)
wts <- ifelse(trainset$response == 1,
              (1 / class_counts[2]) * 0.5 * sum(class_counts),
              (1 / class_counts[1]) * 0.5 * sum(class_counts))

# rerun the model using these weights
set.seed(4)
logit_weighted <- train(as.factor(response) ~ .,
                        data = trainset,
                        method = "glm", family = "binomial",
                        trControl = myCtrl,
                        weights = wts)
summary(logit_weighted)
```

```

Call:
NULL

Coefficients:
                     Estimate Std. Error z value             Pr(>|z|)    
(Intercept)           -6.2306     0.6103  -10.21 < 0.0000000000000002 ***
frequency             -0.0735     0.0164   -4.49        0.00000725130 ***
recency               -0.0220     0.0022  -10.00 < 0.0000000000000002 ***
kidhome1               0.7575     0.1804    4.20        0.00002683821 ***
kidhome2             -13.4816   298.9000   -0.05                 0.96    
num_web_visits_month   0.2031     0.0322    6.30        0.00000000029 ***
log_total_spend        1.1804     0.1120   10.54 < 0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1864.6  on 1344  degrees of freedom
Residual deviance: 1527.2  on 1338  degrees of freedom
AIC: 2009

Number of Fisher Scoring iterations: 14
```

```r
library(car)
vif(logit_weighted$finalModel) ## Multicollinearity issue with log total spend and possibly
        frequency.
```

```
      frequency            recency           kidhome1
        3.68129            1.02233            1.95676
       kidhome2 num_web_visits_month     log_total_spend
        1.00000            1.74676            5.80572
```

```
table2 <- tibble(
  predictors = names(coef(logit_weighted$finalModel)),
  odds_ratio_weighted = exp(coef(logit_weighted$finalModel)),
  odds_ratio_unweighted = exp(coef(logit$finalModel)),
  VarImp_weighted = round(abs(summary(logit_weighted$finalModel)$coefficients[, "z value"]),2),
  VarImp_baseline = round(abs(summary(logit$finalModel)$coefficients[, "z value"]),2)

)
knitr::kable(table2, caption = "Weighted Regression Measures")
```

Weighted Regression Measures

predictors	odds_ratio_weighted	odds_ratio_unweighted	VarImp_weighted	VarImp_baseline
(Intercept)	0.001968	0.000554	10.21	9.50
frequency	0.929139	0.933652	4.49	3.37
recency	0.978224	0.977865	10.00	7.40
kidhome1	2.133008	2.075720	4.20	3.27
kidhome2	0.000001	0.000002	0.05	0.03
num_web_visits_month	1.225166	1.182235	6.30	4.29
log_total_spend	3.255704	3.077601	10.54	7.96

```
## Test weighted model
logit_class_w <- predict(logit_weighted, newdata = testset, type="raw")
CM_weight_def <- confusionMatrix(logit_class_w, as.factor(testset$response), positive = '1')
CM_weight_def
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 523  37
         1 239  96

               Accuracy : 0.692
                 95% CI : (0.66, 0.722)
    No Information Rate : 0.851
    P-Value [Acc > NIR] : 1

                  Kappa : 0.251
```

```
Mcnemar's Test P-Value : <0.0000000000000002

            Sensitivity : 0.722
            Specificity : 0.686
         Pos Pred Value : 0.287
         Neg Pred Value : 0.934
             Prevalence : 0.149
         Detection Rate : 0.107
   Detection Prevalence : 0.374
      Balanced Accuracy : 0.704

       'Positive' Class : 1
```

```
# F1 Score
precision <- unname(CM_weight_def$byClass['Pos Pred Value'])
recall    <- unname(CM_weight_def$byClass['Sensitivity'])
f1_score_weight_def <- 2 * ((precision * recall) / (precision + recall))
cat("F1 Score: ", f1_score_weight_def, "\n")
```

```
F1 Score:  0.410256
```

Comment: The weighted logistic model is showing multicollinearity issues as log of total spend is >5. The weighted model is showing similar odds ratio and variable importance for predictors compared to the baseline logistic model.

Confusion Matrix – Optimal cutoff value + weighted classes

```
library(pROC)

# score the model to get the predicted probabilities (not the predicted class) from the model
logit_class_prob <- predict(logit_weighted, newdata = testset, type = 'prob')

# create the roc_curve (logit_class_prob has 2 columns, take the second one that has the
         probability of 1)
roc_curve <- roc(testset$response, logit_class_prob[,2])

# find and report the optimal cut off
optimal_cutoff <- coords(roc_curve, "best", ret = "threshold") #best uses Youden method
         max(sensitivity+specificity) to balance
cat("CONFUSION MATRIX AT OPTIMAL CUTOFF VALUE OF:", optimal_cutoff$threshold,"\n\n")
```

```
CONFUSION MATRIX AT OPTIMAL CUTOFF VALUE OF: 0.468879
```

```
# rescore using the new optimal cutoff
logit_class_opt <- ifelse(logit_class_prob[,2] >= optimal_cutoff$threshold, 1, 0)

# New Confusion Matrix
```

```r
CM_base_opt <- confusionMatrix(as.factor(logit_class_opt), as.factor(testset$response), positive
        = '1')
CM_base_opt
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 483  27
         1 279 106

               Accuracy : 0.658
                 95% CI : (0.626, 0.689)
    No Information Rate : 0.851
    P-Value [Acc > NIR] : 1

                  Kappa : 0.242

 Mcnemar's Test P-Value : <0.0000000000000002

            Sensitivity : 0.797
            Specificity : 0.634
         Pos Pred Value : 0.275
         Neg Pred Value : 0.947
             Prevalence : 0.149
         Detection Rate : 0.118
   Detection Prevalence : 0.430
      Balanced Accuracy : 0.715

       'Positive' Class : 1
```

```r
# Calculate the F1 Score
precision <- unname(CM_base_opt$byClass['Pos Pred Value']) # synonyms
recall    <- unname(CM_base_opt$byClass['Sensitivity'])   # synonyms
f1_score_base_opt <- 2 * ((precision * recall) / (precision + recall))
cat("F1 Score: ", f1_score_base_opt, "\n")
```

```
F1 Score:  0.409266
```

Optimal Model Selection: Weighted Classes logistic Regression Model

```r
## Confusion Matrix of weighted Classes Logistic Regression
logit_class_w <- predict(logit_weighted, newdata = testset, type="raw")
CM_weight_def <- confusionMatrix(logit_class_w, as.factor(testset$response), positive = '1')
CM_weight_def
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 523  37
         1 239  96

               Accuracy : 0.692
                 95% CI : (0.66, 0.722)
    No Information Rate : 0.851
    P-Value [Acc > NIR] : 1

                  Kappa : 0.251

 Mcnemar's Test P-Value : <0.0000000000000002

            Sensitivity : 0.722
            Specificity : 0.686
         Pos Pred Value : 0.287
         Neg Pred Value : 0.934
             Prevalence : 0.149
         Detection Rate : 0.107
   Detection Prevalence : 0.374
      Balanced Accuracy : 0.704

       'Positive' Class : 1
```

```r
# F1 Score
precision <- unname(CM_weight_def$byClass['Pos Pred Value'])
recall    <- unname(CM_weight_def$byClass['Sensitivity'])
f1_score_weight_def <- 2 * ((precision * recall) / (precision + recall))
cat("F1 Score: ", f1_score_weight_def, "\n")
```

```
F1 Score:  0.410256
```

Comment: **Model Performance**

Accuracy = 69.2%: This shows the overall proportion of correct predictions made by the model. While reasonable, accuracy on its own can be misleading because the dataset is imbalanced.

95% CI (0.66 – 0.722): With 95% confidence, the true accuracy of the model lies between 66% and 72%. This indicates that performance is fairly consistent but below the majority-class benchmark.

No Information Rate = 0.851: This metric is saying that if the model picked the majority class every time it would get an accuracy of 85.1%. Since the model's actual accuracy (69.2%) is below this baseline, the model performs worse than trivial guessing.

P-Value [Acc > NIR] = 1.0: The very high p-value indicates that the model's accuracy is not statistically better than the No Information Rate.

Kappa = 0.251: This metric accounts for the imbalanced dataset and shows the level of agreement between predicted and actual classes beyond chance. A value of 0.25 reflects only fair agreement — stronger than random guessing, but still limited.

McNemar's Test P-Value < 0.0000000000000002: This significance test checks whether the model makes one type of misclassification more often than the other. The very low p-value indicates the errors are asymmetric, with the model producing far more false negatives than false positives. In other words, the model tends to miss actual 1's more often than it incorrectly predicts 1's.

Class-Specific Metrics

Sensitivity = 0.722: The model correctly identifies about 72% of the actual positive cases. This is a strong recall for the minority class, showing the model does a better job of capturing positives compared to before.

Specificity = 0.686: The model correctly identifies about 69% of the negatives. This is weaker than sensitivity and suggests the model sometimes misclassifies negatives as positives.

Positive Predictive Value (Precision) = 0.287: Of all the predicted positives, only 28.7% are truly positive. This indicates the model struggles with precision and produces many false positives.

Negative Predictive Value = 0.934: Of all the predicted negatives, 93.4% are truly negative. This shows the model is much more reliable when predicting the majority class.

Prevalence = 0.149: About 15% of the dataset belongs to the positive class ('1'), confirming the imbalance in the data.

Detection Rate = 0.107: About 10.7% of all records were correctly identified as positives, which is below the actual prevalence but better than the model before.

Detection Prevalence = 0.374: About 37.4% of cases were predicted as positive, which is much higher than the actual prevalence (~15%), reflecting the high number of false positives.

Balanced Accuracy = 0.704: This is the average of sensitivity and specificity. At 70.4%, it suggests the model is better balanced between detecting positives and negatives than raw accuracy alone would suggest.

F1 Score = 0.410: The F1 score combines precision and recall into a single metric for the positive class. The score of 0.41 suggests that while recall is decent, poor precision drags down the balance between capturing positives and being correct when predicting them.

Classification Model Performance Charts

GAINS TABLE

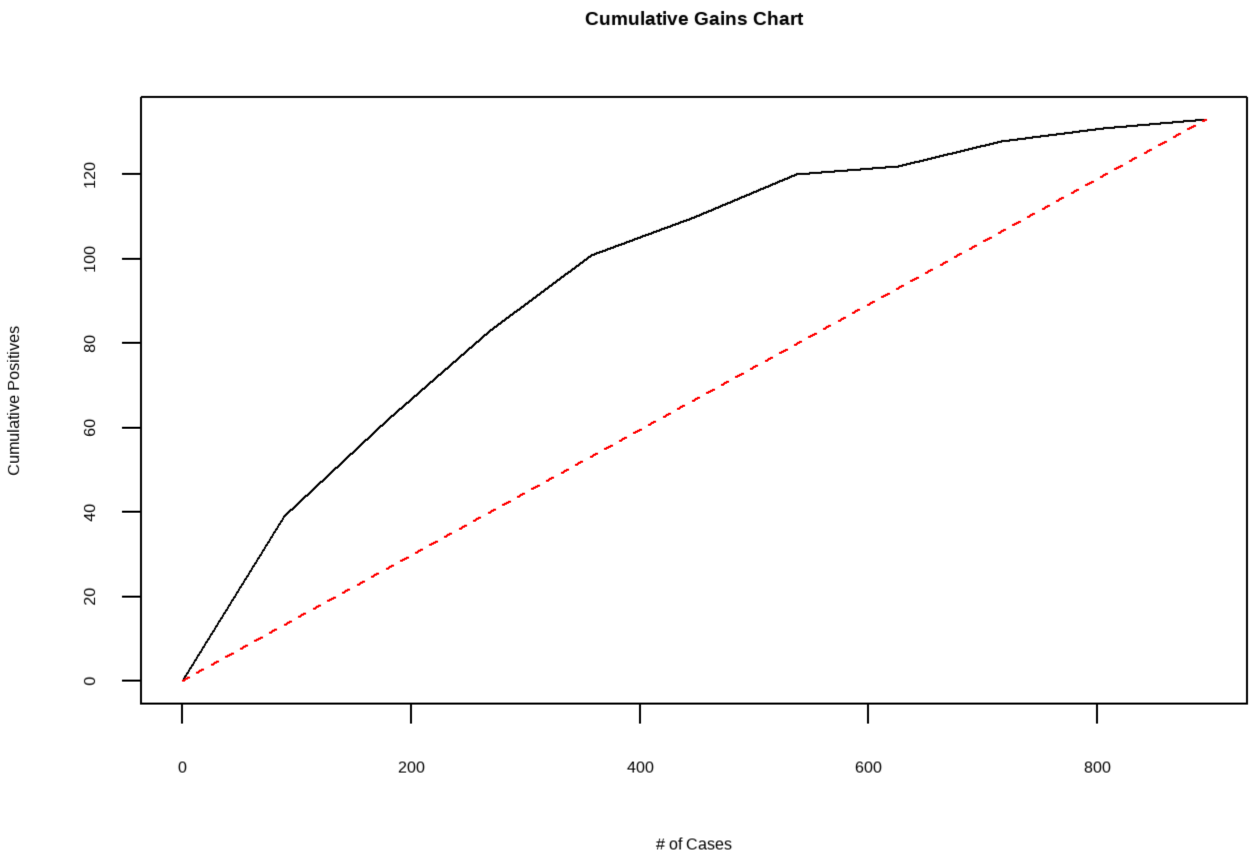
```
library(gains)
logit_class_w <- predict(logit_weighted, newdata = testset, type = 'prob') #!!!!!!! change model
        here
```

```r
testset$response <- as.numeric(as.character(testset$response))
gains_table <- gains(testset$response, logit_class_prob[,2])
gains_table
```

```
Depth                            Cume   Cume Pct                     Mean
 of           Cume     Mean      Mean   of Total    Lift    Cume     Model
File     N      N      Resp      Resp       Resp    Index   Lift     Score
-------------------------------------------------------------------------
  10    89     89      0.44      0.44      29.3%     295     295      0.81
  20    91    180      0.25      0.34      46.6%     170     232      0.70
  30    88    268      0.24      0.31      62.4%     161     208      0.60
  40    90    358      0.20      0.28      75.9%     135     190      0.52
  50    90    448      0.10      0.25      82.7%      67     165      0.46
  60    89    537      0.11      0.22      90.2%      76     150      0.39
  70    89    626      0.02      0.19      91.7%      15     131      0.31
  80    90    716      0.07      0.18      96.2%      45     120      0.24
  90    89    805      0.03      0.16      98.5%      23     110      0.16
 100    90    895      0.02      0.15     100.0%      15     100      0.06
```

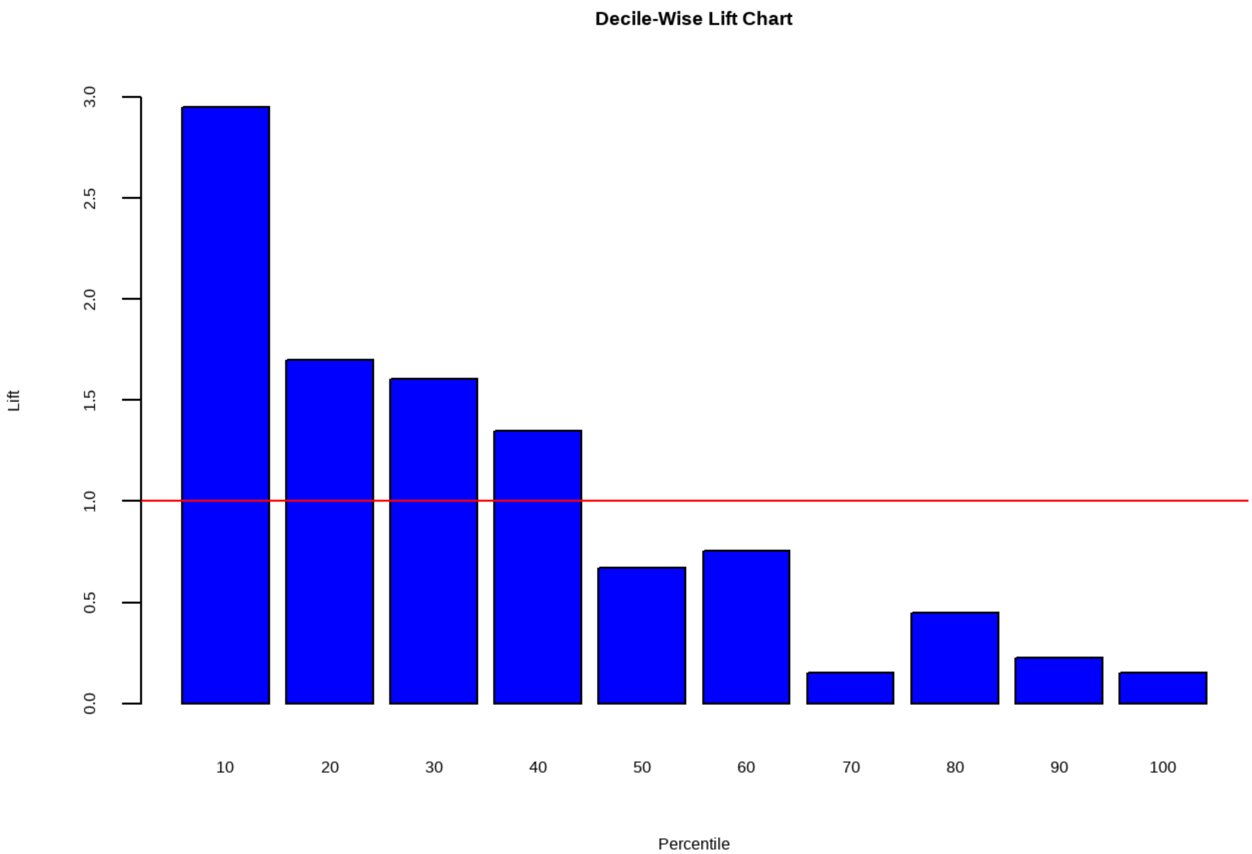
CUMULATIVE GAINS CHART

```r
plot(c(0, gains_table$cume.pct.of.total * sum(testset$response)) ~ c(0, gains_table$cume.obs),
     xlab = '# of Cases',
     ylab = "Cumulative Positives",
     type = "l",
     main = "Cumulative Gains Chart")
lines(c(0, sum(testset$response)) ~ c(0, dim(testset)[1]), col = "red", lty = 2)
```



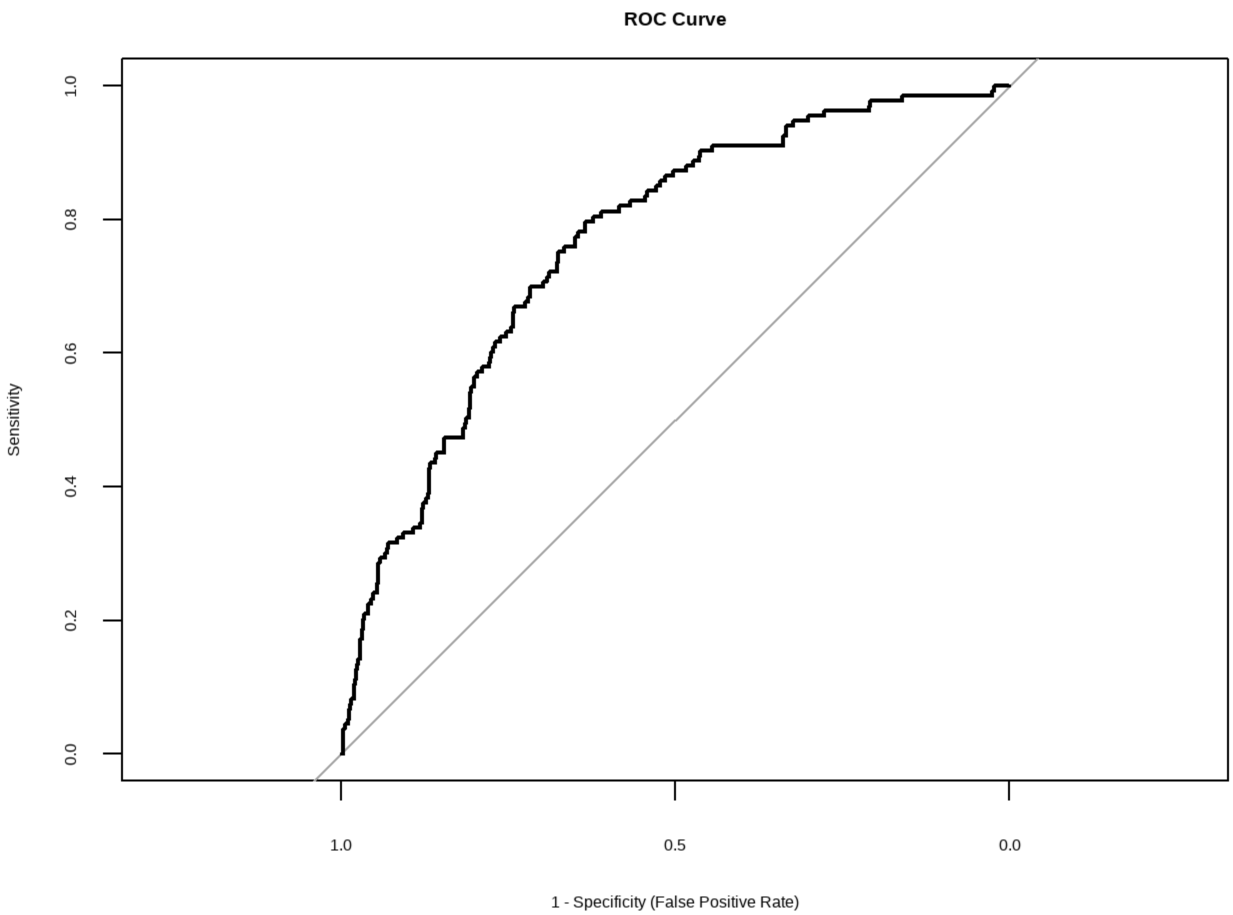
BARPLOT DECILE-WISE LIFT CHART

```
barplot(gains_table$mean.resp/mean(testset$response), names.arg=gains_table$depth,
        xlab="Percentile", ylab="Lift", ylim=c(0,3), col="blue", main="Decile-Wise Lift Chart")
abline(h=c(1),col="red")
```



ROC Curve with AUC

```
library(pROC)
roc_object <- roc(testset$response, logit_class_prob[,2])
plot.roc(roc_object, main="ROC Curve", xlab = "1 - Specificity (False Positive Rate)")
```




```
roc_object$auc
```

```
Area under the curve: 0.764
```

Comment:
The Classification performance charts show that the model does have predictive power. When looking at the gains chart we see that the first 50% of the dataset captures about 83% of all responders. The cumulative lift chart shows that the model is better than random chance when determining the likelihood of responses. This is because the model curve (black line) is above the random baseline (red dashed line). The decile-wise lift chart shows that the first four deciles have a lift value greater than 1, this means that those deciles identify more correct responses than would be expected under random guessing. Finally the ROC curve shows decent discriminative ability with the curve bending above the mid line towards the left corner. The AUC value is 0.764 which supports the fact that this model can distinguish between the two classes with reasonable accuracy.

DEPLOYMENT

Although this model is not perfect it does hold value in helping predict which customers will respond to email campaigns. This model can be used to score customer databases regularly with a continually focus to improve the models accuracy and F1 score. Future refinements to the model should also be considered such as recalculating best cut off value or re weighting classes to optimally battle imbalanced datasets.