ANZ Exploratory Analysis and Predictive Analytics

Ben Crabtree

05/01/2020

Exploratory Data Analysis

From the description provided:

This is a "synthesised transaction dataset containing 3 months' worth of transactions for 100 hypothetical customers. It contains purchases, recurring transactions, and salary transactions. The dataset is designed to simulate realistic transaction behaviours that are observed in ANZ's real transaction data."

The data set contains 12,043 observations with 23 variables.

Loading and Cleaning Data

```
library(tidyverse)
library(readxl)
data = read_excel('ANZ synthesised transaction dataset.xlsx', na = c("NA",""," ", "n/a"));
```

```
clean_data = janitor::clean_names(data)
```

Some Descriptive Statistics

```
amount = clean_data$amount
mean(amount)
```

```
## [1] 187.9336
```

```
median(amount)
```

```
## [1] 29
```

```
max(amount)
```

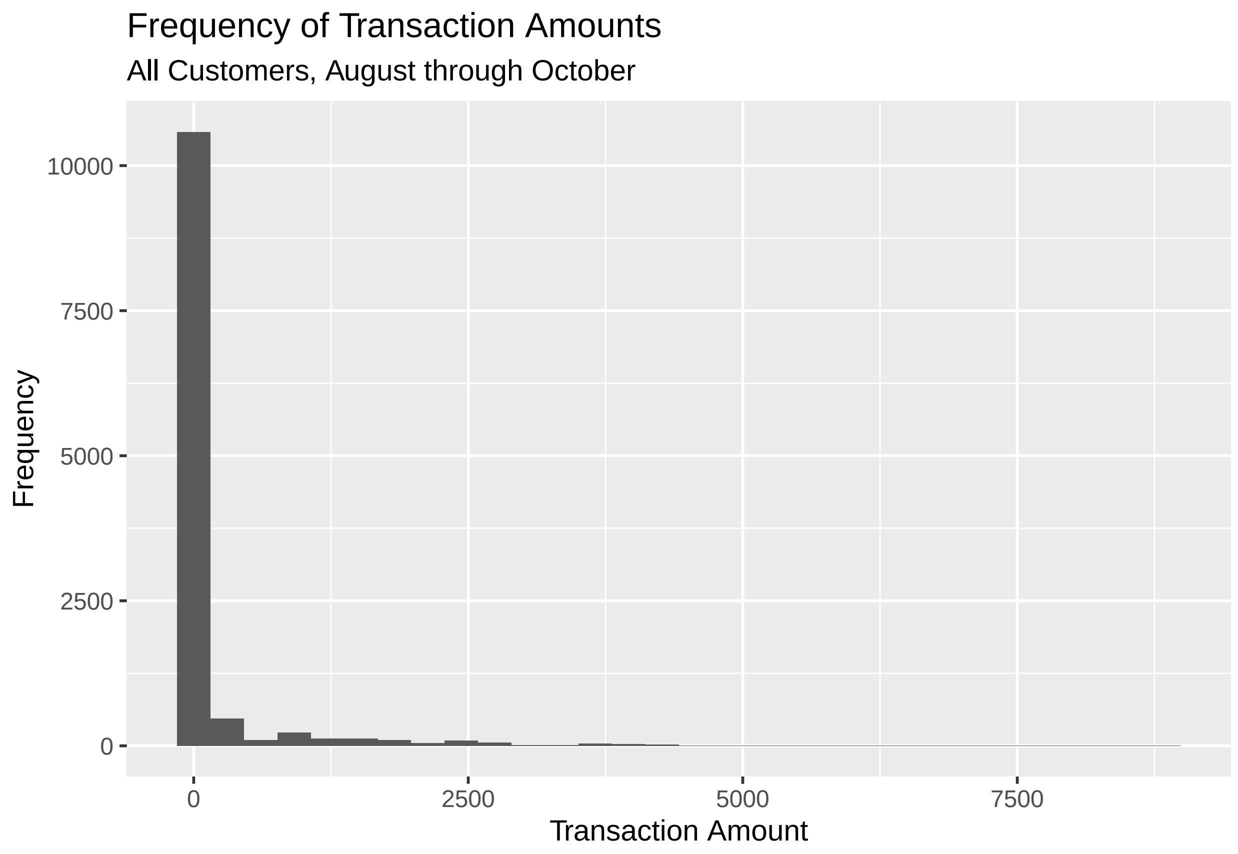
```
## [1] 8835.98
```

```
min(amount)
```

```
## [1] 0.1
```

The mean transaction amount is \$187.93 over all customers across all three months, however the median amount is \$29, suggesting that there are large outliers skewing the mean. Indeed, we see the maximum transaction amount is \$8835.98, while the minimum is \$0.1. Visualising the transaction amount frequencies we see that very small transactions are by far the most frequent.

```r
amount %>% as.data.frame() %>%
              ggplot() +
              geom_histogram(aes(x=amount)) +
  labs(x = 'Transaction Amount',
    y = 'Frequency',
    title = "Frequency of Transaction Amounts",
    subtitle = 'All Customers, August through October')
```



In August there were 3943 transactions, in September there were 4013 and in October there were 4087. So on average there were 4014.33 transactions per month. The average number of transactions per customer over the three months is 120.43 (num obs / num unique account numbers). The average number of transactions per customer in August was 39.43, while the average in September was 40.13 and the average in October was 40.87.

Number of transactions per month:

```r
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 3.6.2

##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

```
transactions_bymonth = clean_data %>%
    group_by(month = floor_date(date, "month")) %>%
    summarize(num_transactions = length(amount))

transactions_bymonth
```

```
## # A tibble: 3 x 2
##   month               num_transactions
##   <dttm>                         <int>
## 1 2018-08-01 00:00:00             3943
## 2 2018-09-01 00:00:00             4013
## 3 2018-10-01 00:00:00             4087
```

Number of transactions by account for each month:

```
numtransbyaccount_aug = clean_data %>%
                            filter(date >= as.Date("2018-08-01")
                                    & date <= as.Date("2018-08-31")) %>%
                            group_by(account) %>%
                            summarize(num_transactions = length(amount))

head(numtransbyaccount_aug)
```

```
## # A tibble: 6 x 2
##   account        num_transactions
##   <chr>                     <int>
## 1 ACC-1037050564               88
## 2 ACC-1056639002               36
## 3 ACC-1199531521               34
## 4 ACC-1217063613                3
## 5 ACC-1222300524               94
## 6 ACC-1243371644               28
```

```
sum(numtransbyaccount_aug$num_transactions) / length(unique(numtransbyaccount_aug$account))
```

```
## [1] 39.43
```

```
numtransbyaccount_sep = clean_data %>%
                            filter(date >= as.Date("2018-09-01")
                                    & date <= as.Date("2018-09-30")) %>%
                            group_by(account) %>%
                            summarize(num_transactions = length(amount))

head(numtransbyaccount_sep)
```

```
## # A tibble: 6 x 2
##   account        num_transactions
##   <chr>                     <int>
## 1 ACC-1037050564               97
## 2 ACC-1056639002               36
## 3 ACC-1199531521                24
## 4 ACC-1217063613                 1
## 5 ACC-1222300524               104
## 6 ACC-1243371644                16
```

```
sum(numtransbyaccount_sep$num_transactions) / length(unique(numtransbyaccount_sep$account))
```

```
## [1] 40.13
```

```
numtransbyaccount_oct = clean_data %>%
                            filter(date >= as.Date("2018-10-01")
                                   & date <= as.Date("2018-10-31")) %>%
                            group_by(account) %>%
                            summarize(num_transactions = length(amount))

head(numtransbyaccount_oct)
```

```
## # A tibble: 6 x 2
##   account        num_transactions
##   <chr>                     <int>
## 1 ACC-1037050564               74
## 2 ACC-1056639002               14
## 3 ACC-1199531521               19
## 4 ACC-1217063613               21
## 5 ACC-1222300524              105
## 6 ACC-1243371644               36
```

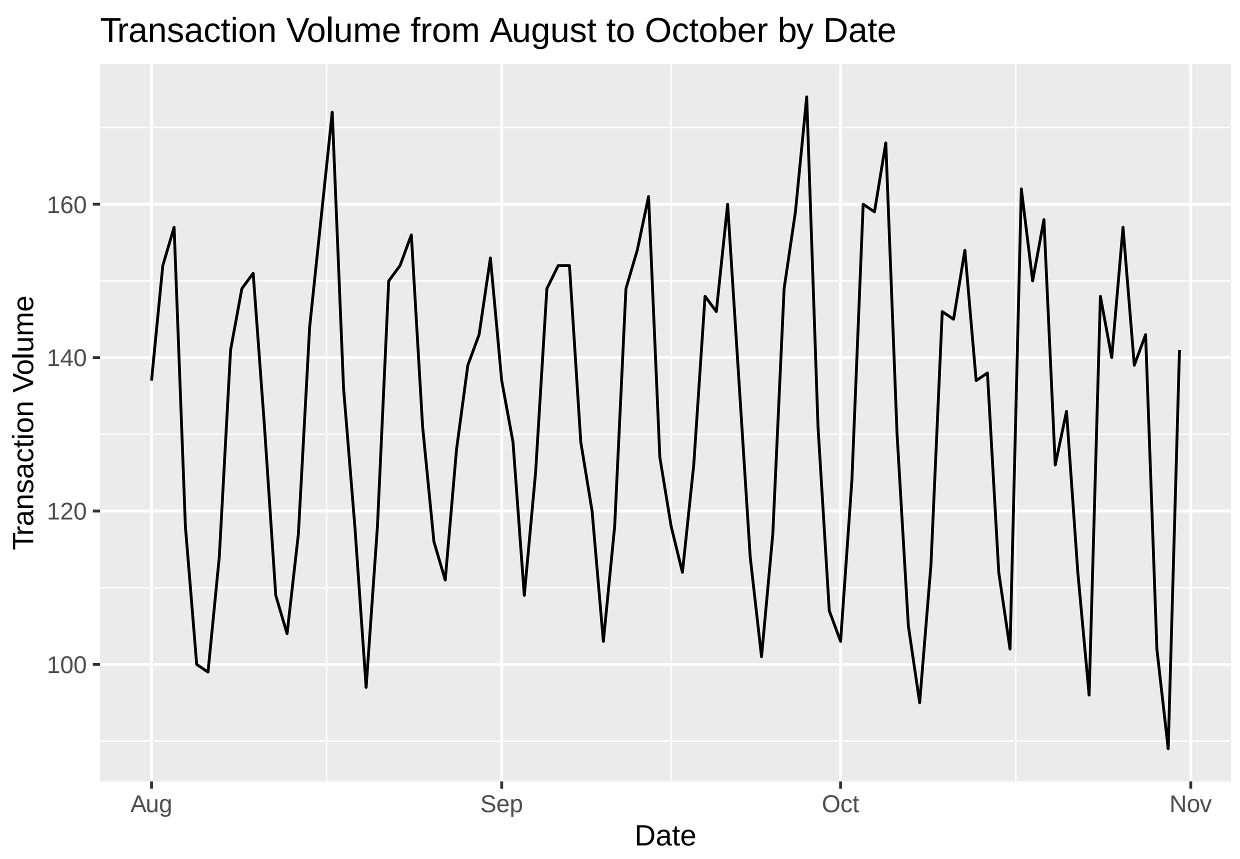
```
sum(numtransbyaccount_oct$num_transactions) / length(unique(numtransbyaccount_oct$account))
```

```
## [1] 40.87
```

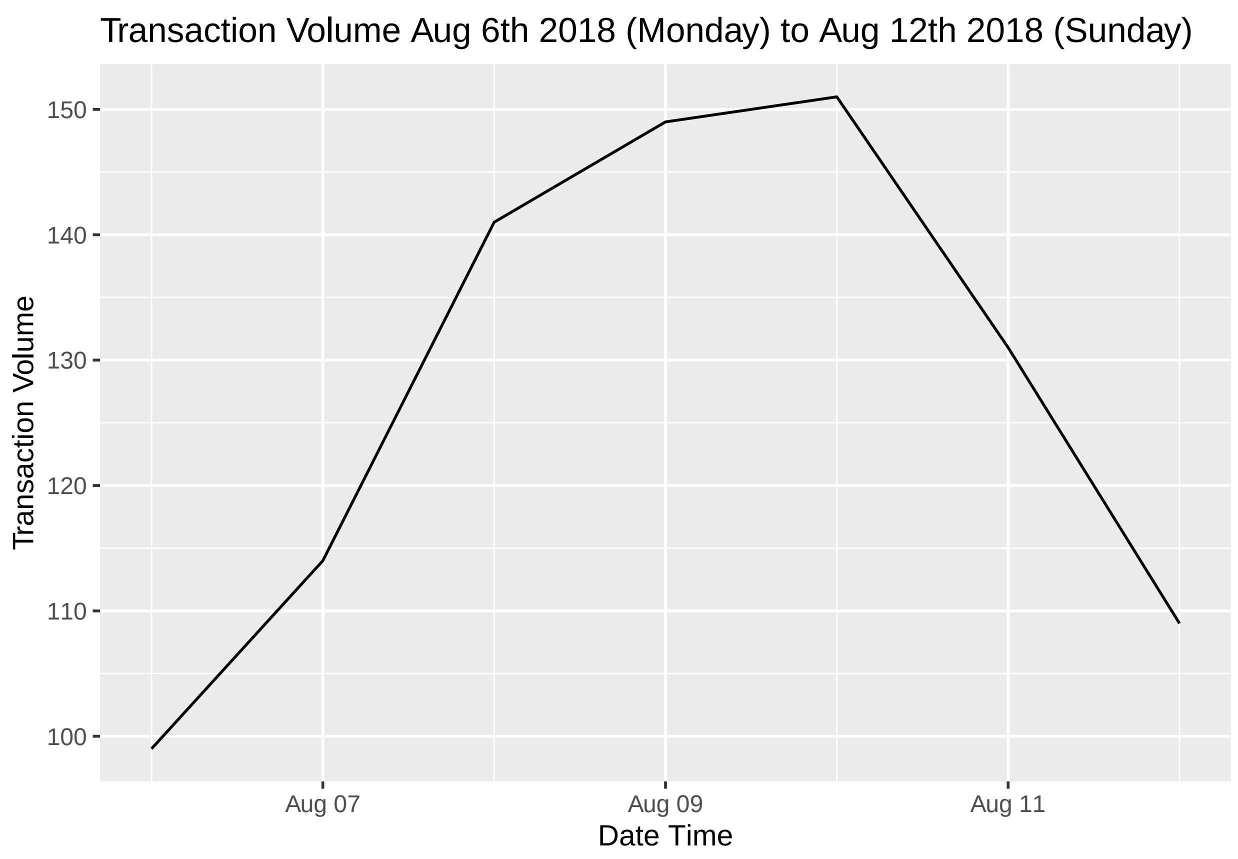
Transaction Volume

When plotting transaction volume from August to October by date, we see a cyclical pattern of peaks and troughs. Selecting a week from each month and plotting the volume by day, we see that weekly, transaction volume starts low at around 100 transactions on Monday, rising daily until a peak of around 150 by Friday and then dropping back to around 110 by Sunday, ready for the cycle to repeat. This aligns with the time series over the three months as we see four peak cycles per month, one for each week.

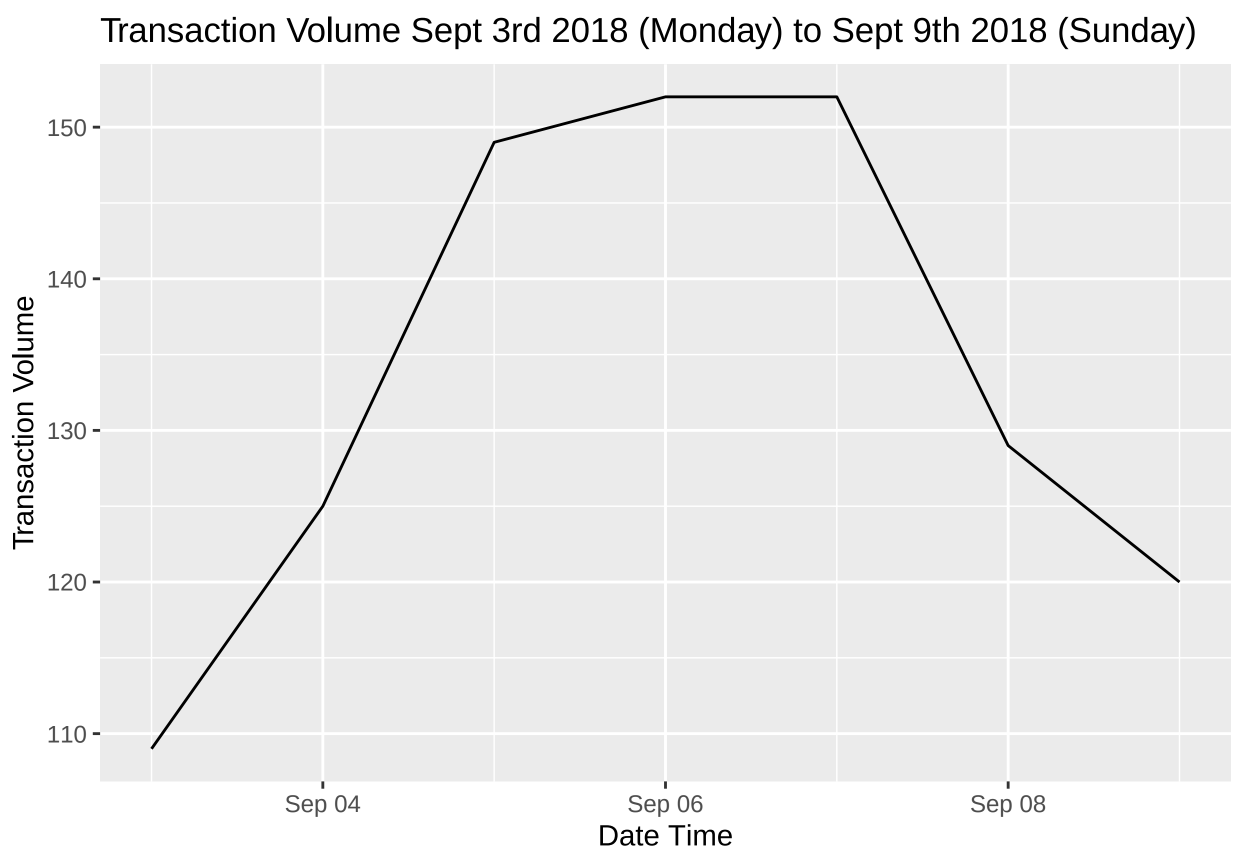
```
volume_date = clean_data %>%
    group_by(date = date) %>%
    summarize(volume = length(amount))

volume_date %>% ggplot(aes(x = date, y = volume)) +
  geom_line() +
  labs(x = "Date",
    y = "Transaction Volume",
    title = "Transaction Volume from August to October by Date")
```



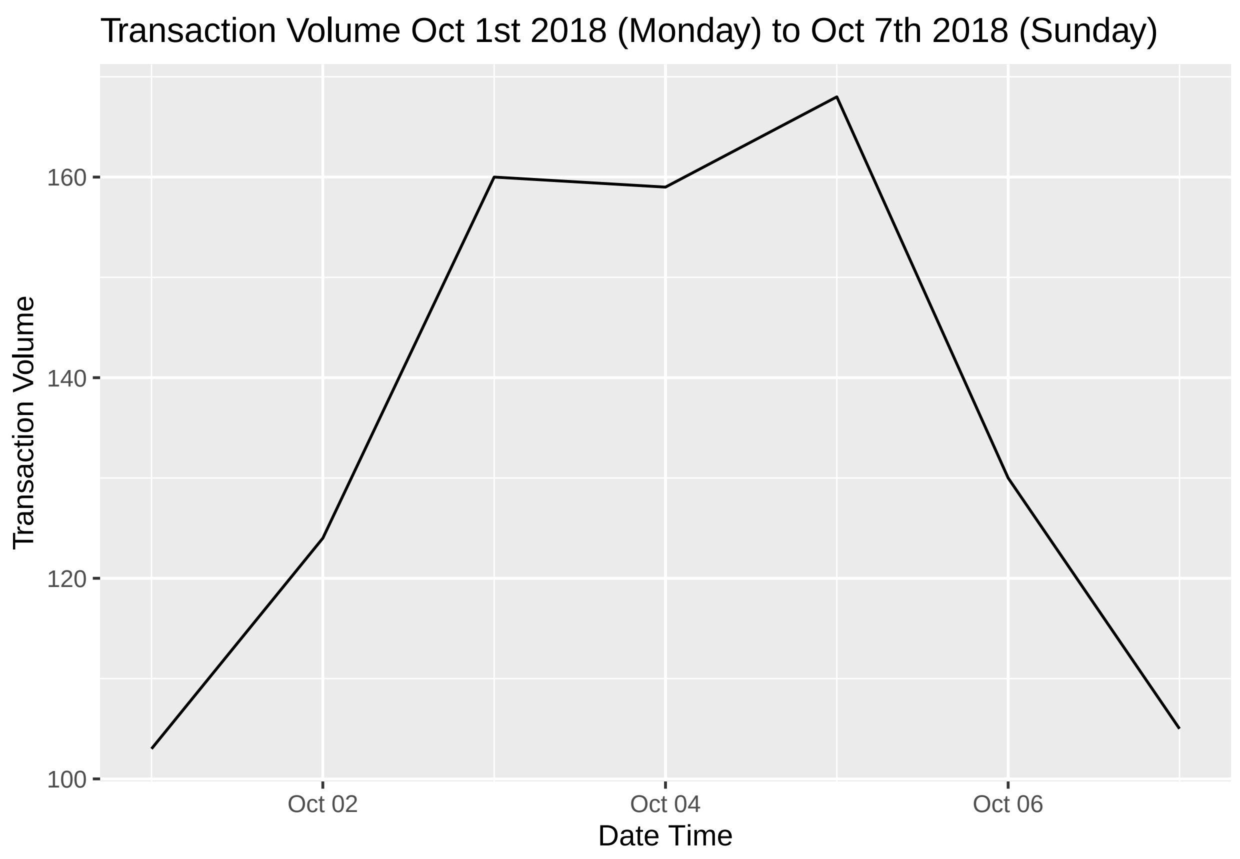
```r
volume_Aug1toAug7 = clean_data %>%
                            filter(date >= as.Date("2018-08-06")
                                    & date <= as.Date("2018-08-12")) %>%
                            group_by(date_time = date) %>%
                            summarize(volume = length(amount))

volume_Aug1toAug7 %>% ggplot(aes(x = date_time, y = volume)) +
  geom_line() +
  labs(x = "Date Time",
    y = "Transaction Volume",
    title = "Transaction Volume Aug 6th 2018 (Monday) to Aug 12th 2018 (Sunday)")
```



```r
volume_Sep3toSep9 = clean_data %>%
                            filter(date >= as.Date("2018-09-03")
                                    & date <= as.Date("2018-09-09")) %>%
                            group_by(date_time = date) %>%
                            summarize(volume = length(amount))

volume_Sep3toSep9 %>% ggplot(aes(x = date_time, y = volume)) +
  geom_line() +
  labs(x = "Date Time",
    y = "Transaction Volume",
    title = "Transaction Volume Sept 3rd 2018 (Monday) to Sept 9th 2018 (Sunday)")
```



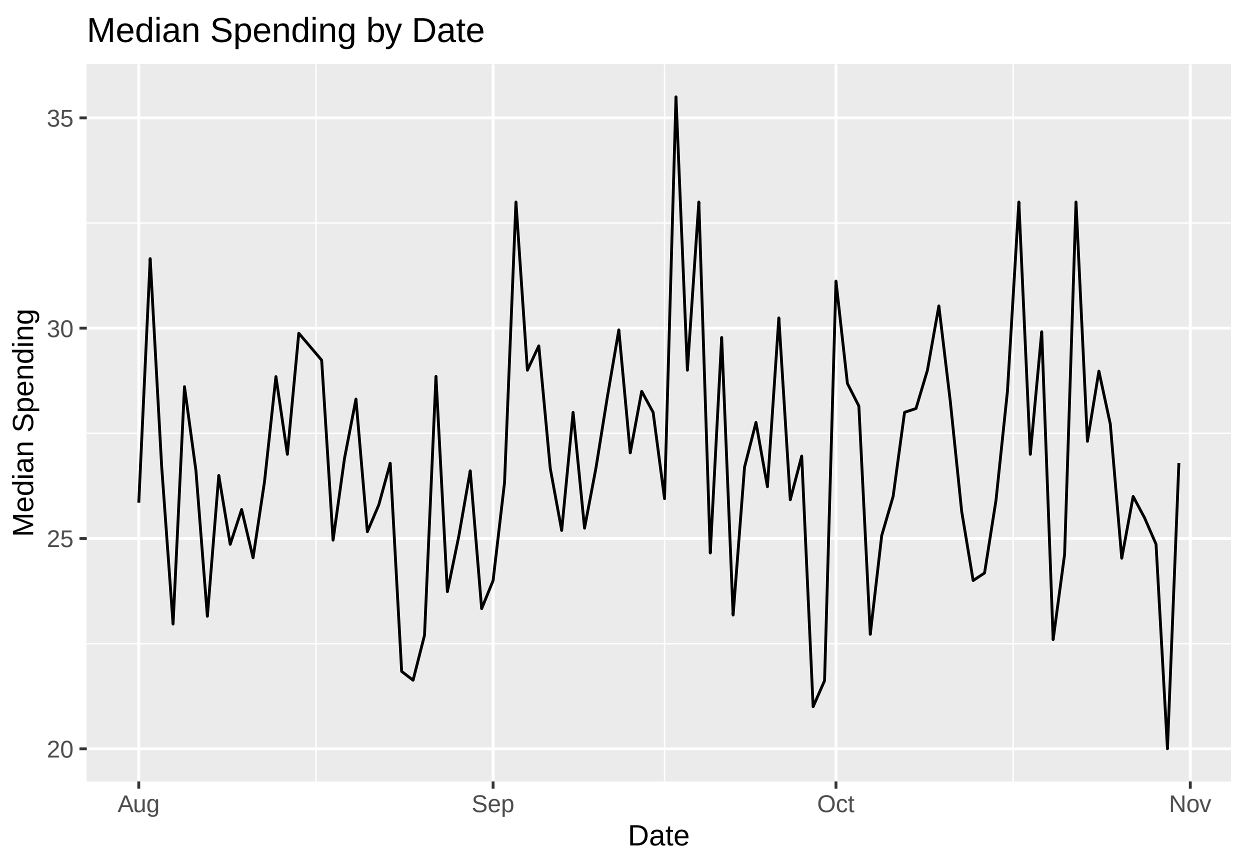
```
volume_Oct1toOct7 = clean_data %>%
                            filter(date >= as.Date("2018-10-01")
                                    & date <= as.Date("2018-10-07")) %>%
                            group_by(date_time = date) %>%
                            summarize(volume = length(amount))

volume_Oct1toOct7 %>% ggplot(aes(x = date_time, y = volume)) +
  geom_line() +
  labs(x = "Date Time",
    y = "Transaction Volume",
    title = "Transaction Volume Oct 1st 2018 (Monday) to Oct 7th 2018 (Sunday)")
```



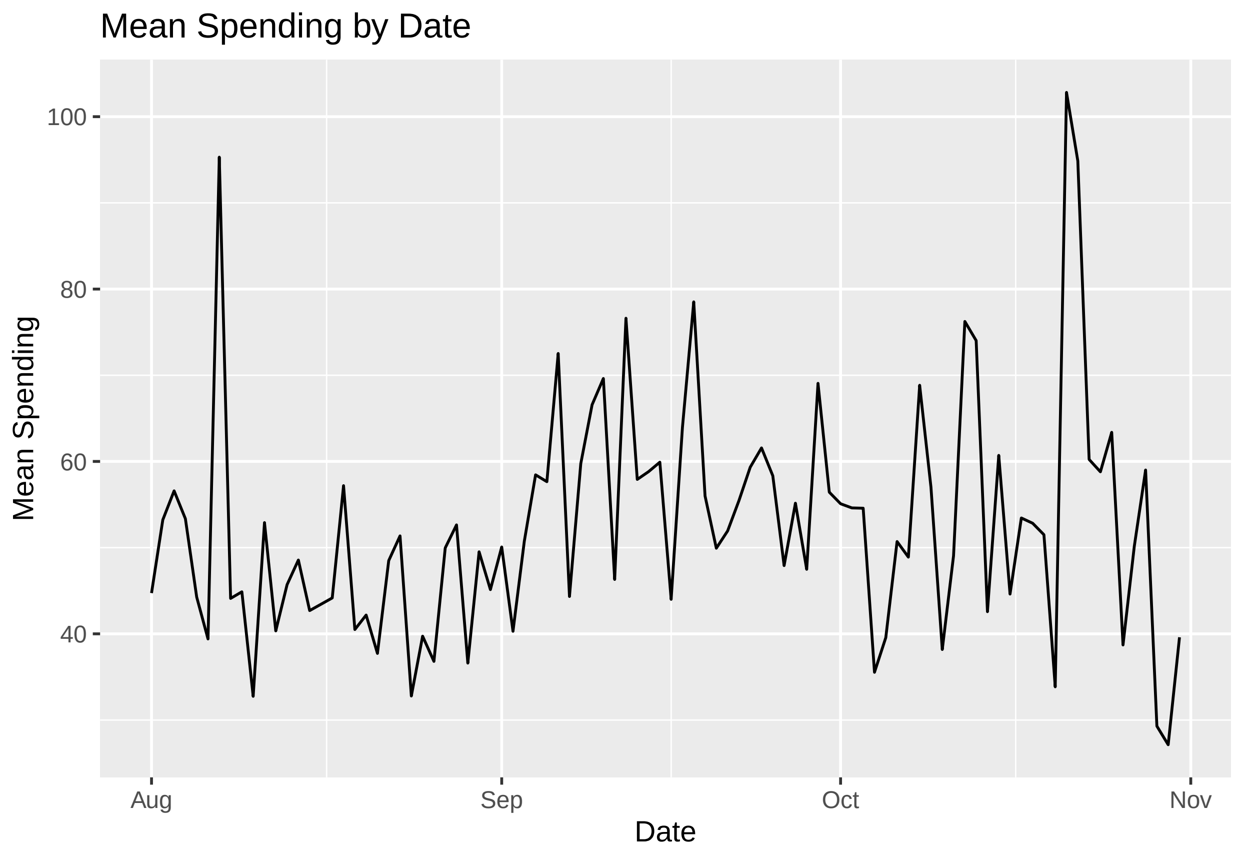
Spending

When looking at a time series of spending by date we see fairly different results when looking at median daily spending and mean daily spending, due large outliers skewing the mean. The result is that mean daily spending looks a lot more volatile, ranging between around $10 and over $100. Median daily spending however looks more stable, fluctuating between around $20 and $35.

```r
spending_date = clean_data %>%
    filter(movement == 'debit') %>%
    group_by(date = date) %>%
    summarize(median_amount = median(amount))

spending_date %>% ggplot(aes(x = date, y = median_amount)) +
  geom_line() +
  labs(x = "Date",
    y = "Median Spending",
    title = "Median Spending by Date")
```

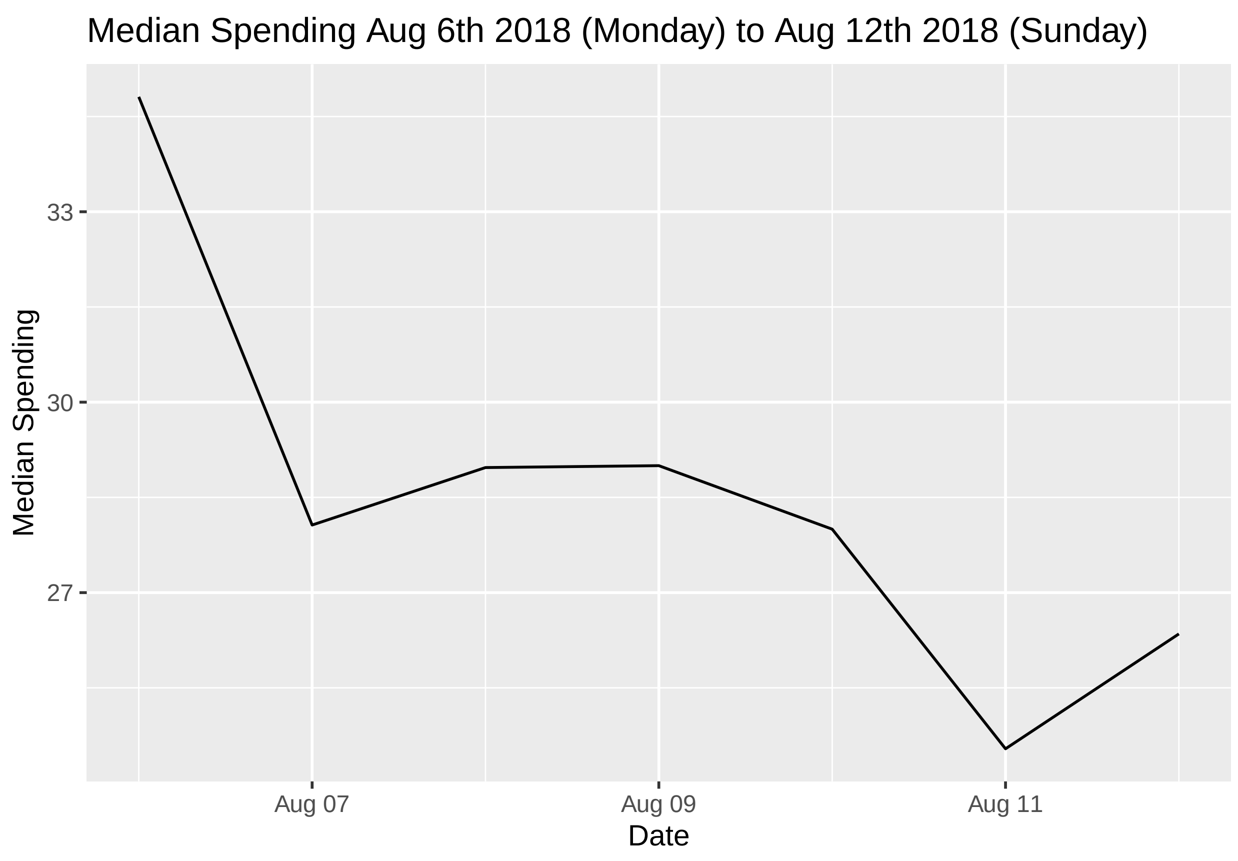


```
mean_spending_date = clean_data %>%
    filter(movement == 'debit') %>%
    group_by(date = date) %>%
    summarize(mean_amount = mean(amount))

mean_spending_date %>% ggplot(aes(x = date, y = mean_amount)) +
  geom_line() +
  labs(x = "Date",
    y = "Mean Spending",
    title = "Mean Spending by Date")
```

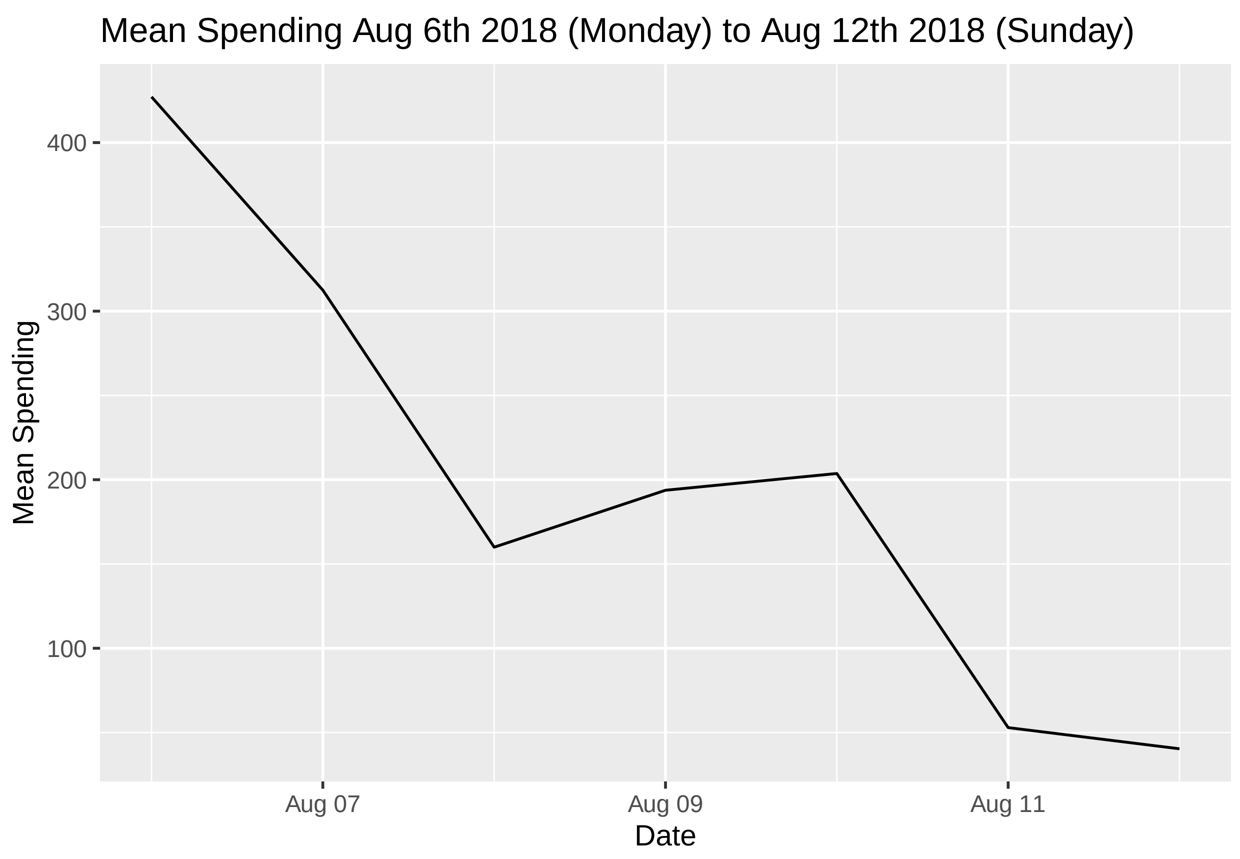


As there do not appear to be regular cyclical patterns in daily spending by week, whether aggregated by mean or median, it is not possible to choose a representative week, but examining the first Monday to Sunday in August again, we see what appears to be high spending on the Monday (median and mean) with decreasing spending through the rest of the week. Surprisingly, the lowest amount of spending is over the weekend.

```
medianspending_Aug1toAug7 = clean_data %>%
                            filter(date >= as.Date("2018-08-06")
                                    & date <= as.Date("2018-08-12")) %>%
                            group_by(date_time = date) %>%
                            summarize(median_spending = median(amount))

medianspending_Aug1toAug7 %>% ggplot(aes(x = date_time, y = median_spending)) +
  geom_line() +
  labs(x = "Date",
    y = "Median Spending",
    title = "Median Spending Aug 6th 2018 (Monday) to Aug 12th 2018 (Sunday)")
```



```r
meanspending_Aug1toAug7 = clean_data %>%
                            filter(date >= as.Date("2018-08-06")
                                    & date <= as.Date("2018-08-12")) %>%
                            group_by(date_time = date) %>%
                            summarize(mean_spending = mean(amount))

meanspending_Aug1toAug7 %>% ggplot(aes(x = date_time, y = mean_spending)) +
  geom_line() +
  labs(x = "Date",
    y = "Mean Spending",
    title = "Mean Spending Aug 6th 2018 (Monday) to Aug 12th 2018 (Sunday)")
```



Predictive Analytics

Annual Salary

Using the same transaction dataset, identify the annual salary for each customer

Annual salary for each customer can be estimated by filtering on the txn_description column and looking only at those rows marked 'PAY/SALARY'. Multiplying the amount paid by the frequency of payment gives the tri-monthly payment. Multiplying by 4 gives the yearly salary.

```r
salary_credits = clean_data %>% filter(txn_description == 'PAY/SALARY')

yearly_salary = salary_credits %>% group_by(account) %>% summarize(age = mean(age), gender = unique(gend

head(yearly_salary)
```

```
## # A tibble: 6 x 4
##   account          age gender annual_salary
##   <chr>          <dbl> <chr>          <dbl>
## 1 ACC-1037050564    40 F             46389.
## 2 ACC-1056639002    22 M             76680.
## 3 ACC-1199531521    52 M            106002.
## 4 ACC-1217063613    27 F             38909.
## 5 ACC-1222300524    38 M             52111.
## 6 ACC-1243371644    42 M             40358.
```

Explore correlations between annual salary and various customer attributes (e.g. age). These attributes could be those that are readily available in the data (e.g. age) or those that you construct or derive yourself (e.g. those relating to purchasing behaviour). Visualise any interesting correlations using a scatter plot.

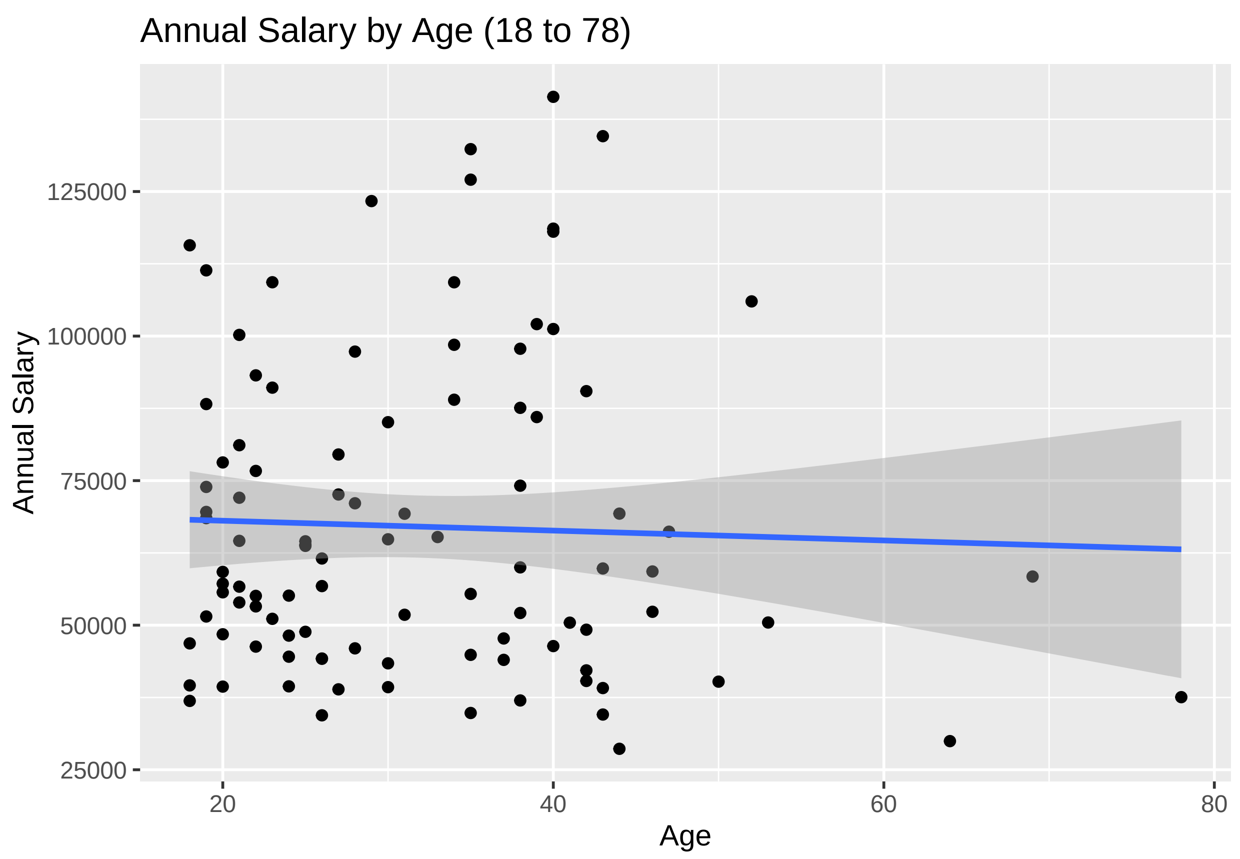
Age

As age ranges from 18 to 78 in this dataset - that is, from people just starting their careers to retirees - there is a somewhat non-linear relationship between age and annual salary when considering the entire age range. One reason is that those customers beyond the age of 60 all have incomes on the lower end of the spectrum. Annual salaries range between $28,623.84 and $141,375.68, but there appears to be quite a lot of variance in salary across the 18 to 60 age range with some 18 year olds making over $100,000 and some 50 year olds making around $50,000.

Performing a linear regression of annual salary on age reveals a weak negative trend, with a correlation coefficient of -0.037, likely due to the depressed earnings of those in the above 60 range. Removing those customers in that age range and performing linear regression again reveals a weak positive correlation between age and annual salary, with a correlation coefficient of 0.076, suggesting that there is some evidence that annual salary rises with age, until the age of 60 (around retirement), after which it drops.

Due to the fairly non-linear relationship of age and annual salary across the entire age range, it is questionable whether age will make a good predictor of annual salary in a multiple regression model.

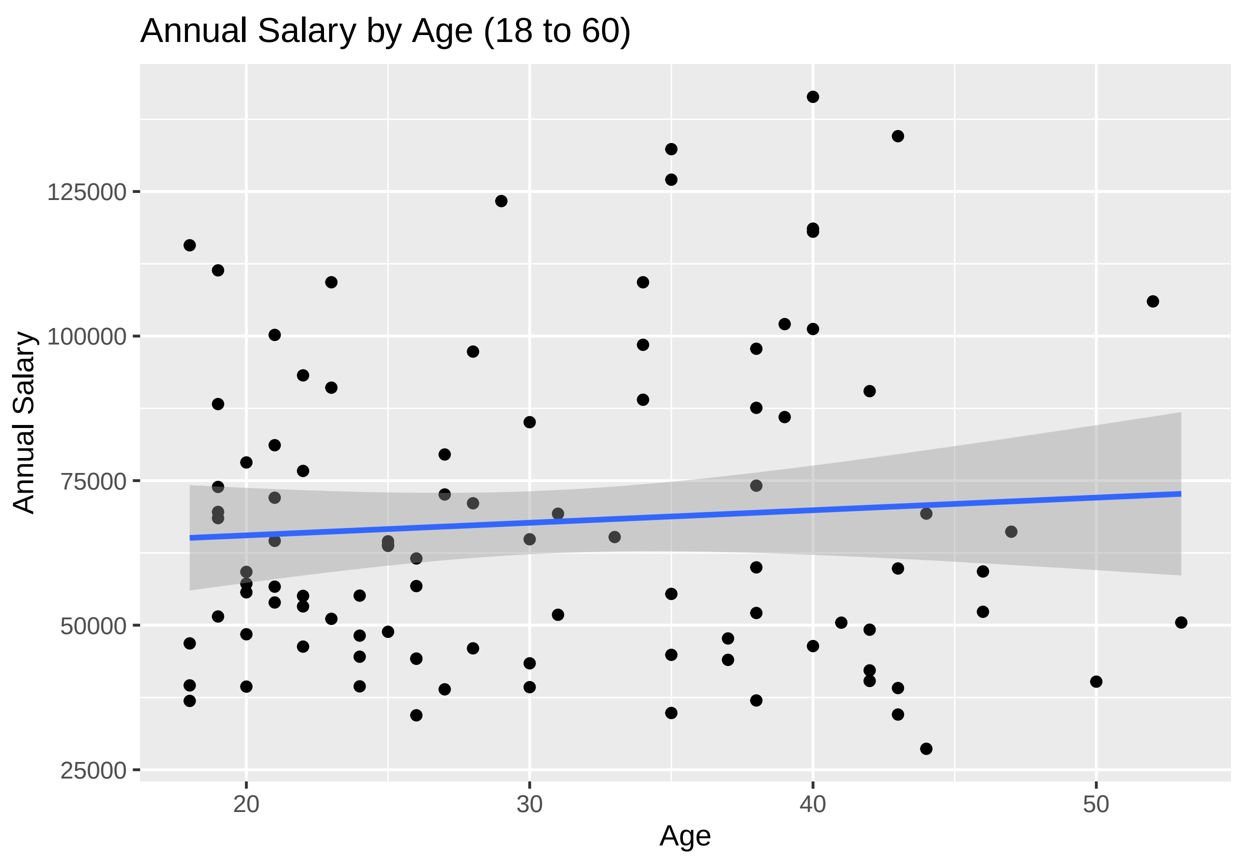
```r
yearly_salary %>% ggplot(aes(x = age, y = annual_salary)) + geom_point() +
  geom_smooth(method=lm) +
  theme(legend.position = "none") +
  labs(y = "Annual Salary",
       x = "Age",
       title = 'Annual Salary by Age (18 to 78)')
```



```r
cor(yearly_salary$annual_salary, yearly_salary$age)
```

```
## [1] -0.0365039
```

```r
yearly_salary_18to60 = yearly_salary %>% filter(yearly_salary$age <= 60)

yearly_salary_18to60 %>% ggplot(aes(x = age, y = annual_salary)) + geom_point() +
  geom_smooth(method=lm) +
  theme(legend.position = "none") +
  labs(y = "Annual Salary",
       x = "Age",
       title = 'Annual Salary by Age (18 to 60)')
```



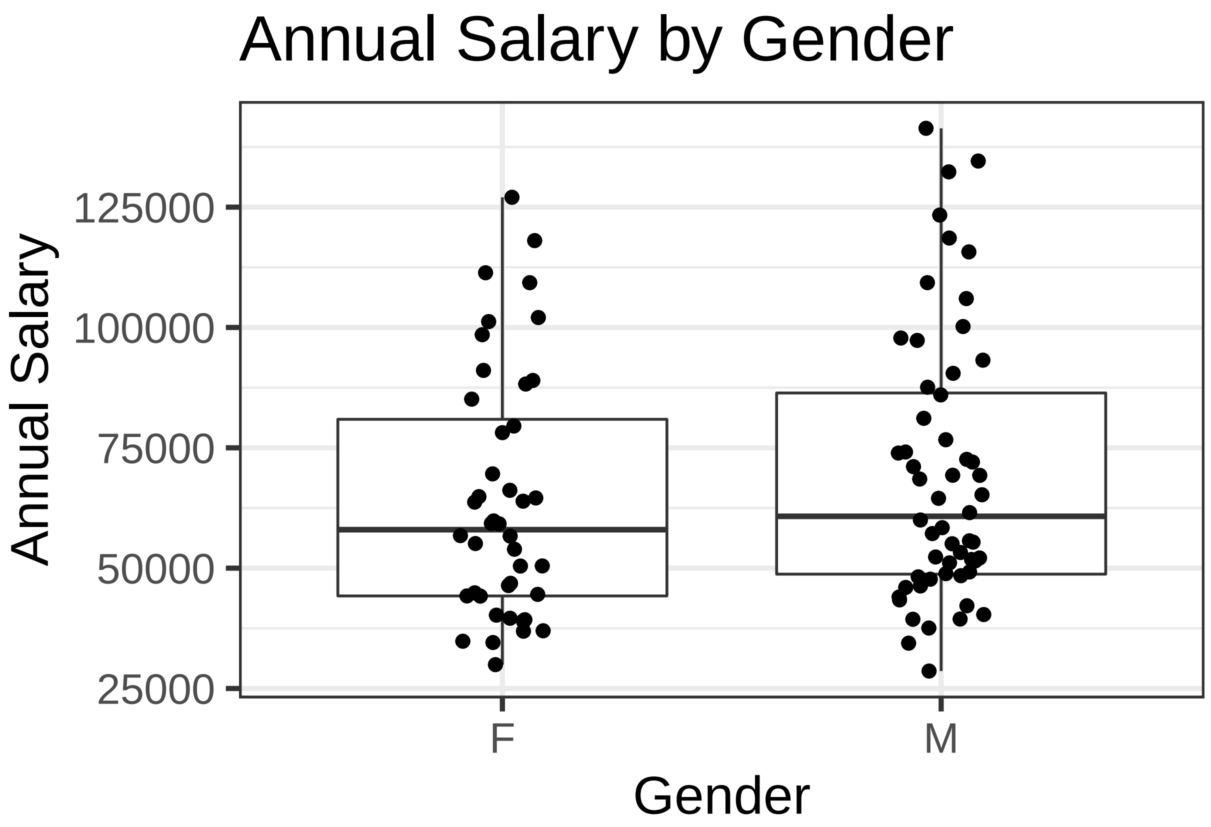
```
cor(yearly_salary_18to60$annual_salary, yearly_salary_18to60$age)
```

```
## [1] 0.07595699
```

Gender

It appears from plotting annual salary by gender that men may make more than women, but this question can be formalised by a hypothesis test. We would like to use a two sample t-test, but first we must check the assumption of normally distributed data is met.

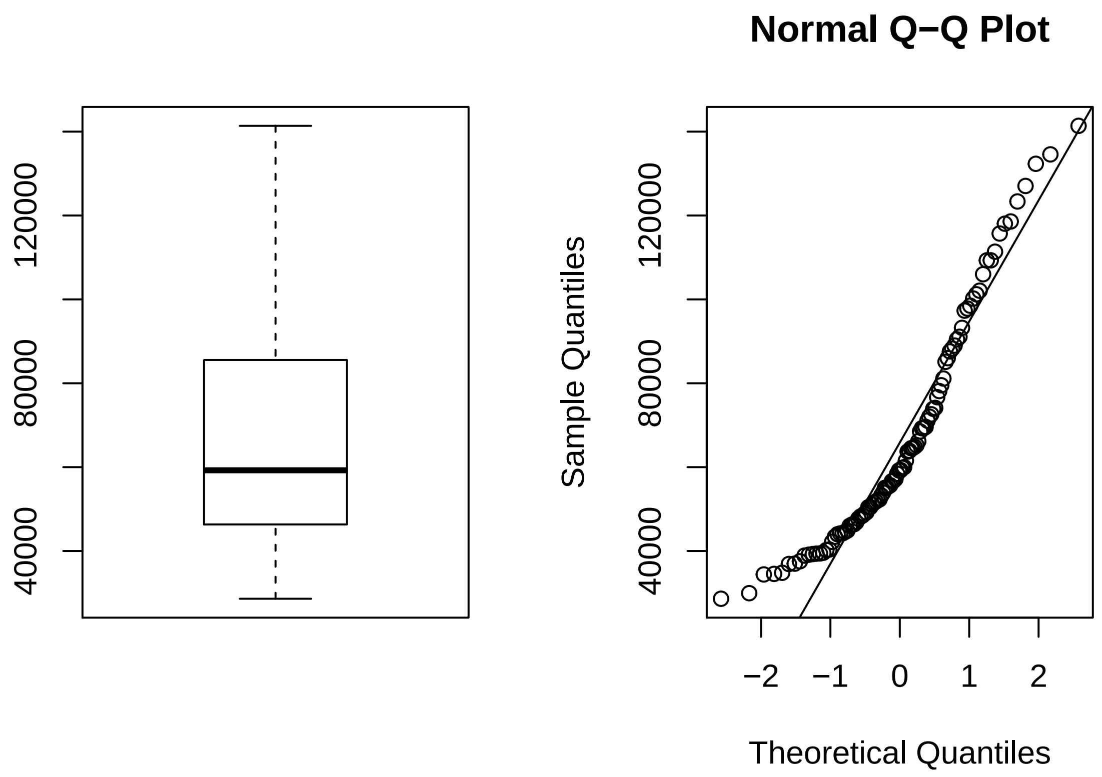
```
yearly_salary %>% ggplot(aes(y = annual_salary, x = gender)) +
  geom_boxplot(coef = 10) +
  geom_jitter(width=0.1, size = 2) +
  theme_bw(base_size = 20) +
  theme(legend.position = "none") +
  labs(y = "Annual Salary",
       x = "Gender",
       title = 'Annual Salary by Gender')
```



Testing Normality Assumption:

Plotting the annual salary data in a box plot and Q-Q plot to test for normality, we see that the box plot does not look very symmetric indicating that the median is in the lower part of the range. The Q-Q plot looks fairly linear except for the lower theoretical quantiles. From this we can conclude that the data seems fairly normally distributed, and the number of observations (100) allows us to rely on the central limit theorem for the normal distribution of sample means. So we will conclude that the assumptions necessary to perform a t-test are met.

```r
par(mfrow = c(1,2))
boxplot(yearly_salary$annual_salary)
qqnorm(yearly_salary$annual_salary)
qqline(yearly_salary$annual_salary)
```



Two Sample t-test:

We will test the following hypotheses at the 5% significance level:

Null Hypothesis: The mean annual salary of men is no different from the mean annual salary of women.

Alternate Hypothesis: The mean annual salary of men is greater than the mean annual salary of women.

```
salary_men = as_vector(yearly_salary %>% filter(gender == 'M') %>% select(annual_salary))
salary_women = as_vector(yearly_salary %>% filter(gender == 'F') %>% select(annual_salary))

t.test(salary_men, salary_women, alternative = 'greater')
```

```
##
##  Welch Two Sample t-test
##
## data:  salary_men and salary_women
## t = 1.0279, df = 95.592, p-value = 0.1533
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  -3403.147       Inf
## sample estimates:
## mean of x mean of y
##  69494.33  63968.75
```

With a test statistic of 1.0279 giving a p-value of 0.1533, at the 5% significance level, there is evidence to suggest that we should not reject the null hypothesis that the mean annual salary of men is no different from the mean annual salary of women.

Considering this result, it is debatable whether gender would be a good predictor of annual salary in a regression model.

Spending Habits

We will consider total yearly spending for each customer as a metric summarising their spending habits. Frequency of debit transactions arguably does not matter as much as the total amount spent each year as it is possible to engage in very frequent small transactions yet not spend much overall, or to engage in fewer but larger transactions and spend much more. We will consider yearly spending to keep the time scale consistent with yearly salary.

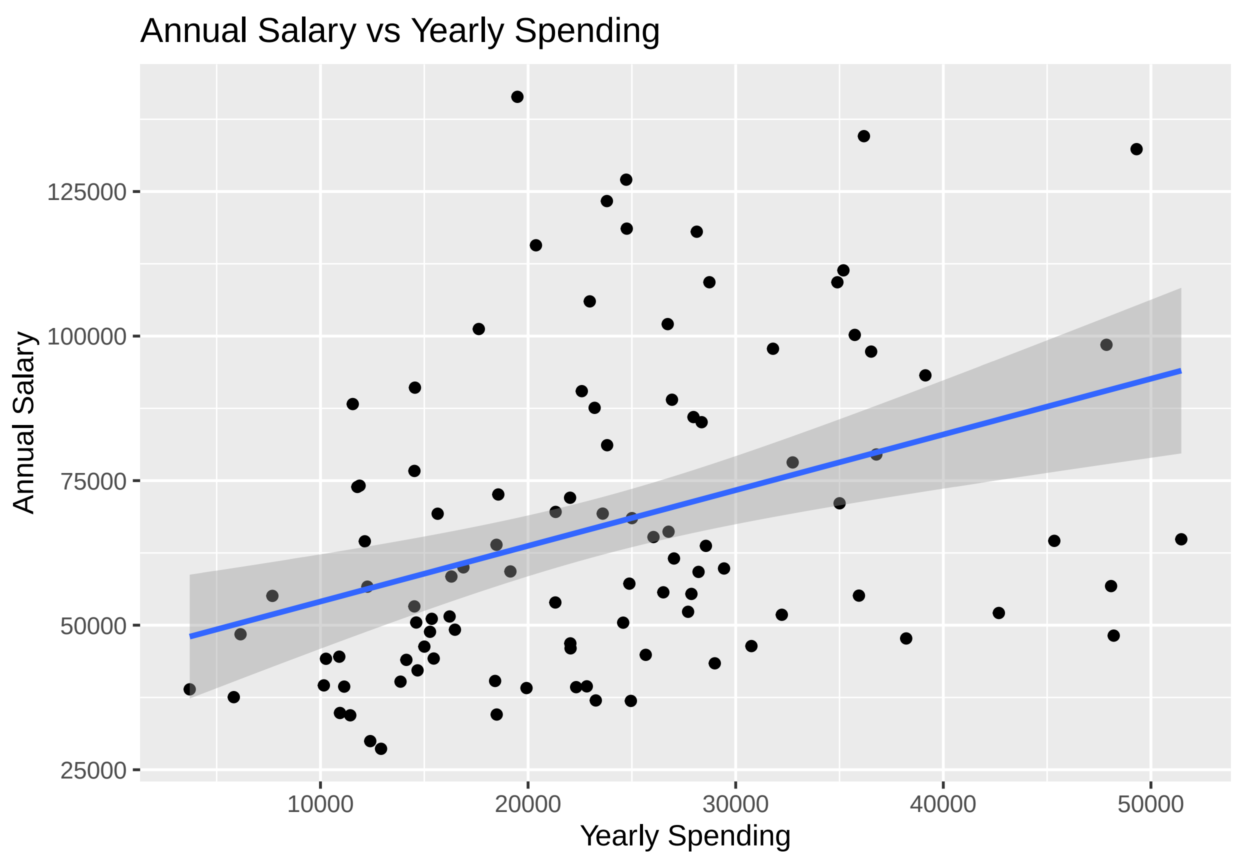
```
spending = clean_data %>% filter(movement == 'debit') %>% group_by(account) %>% summarize(debits_per_mon

salary_metrics = merge(yearly_salary, spending)

head(salary_metrics)
```

```
##          account age gender annual_salary debits_per_month
## 1 ACC-1037050564  40      F      46388.68        82.000000
## 2 ACC-1056639002  22      M      76680.24        26.666667
## 3 ACC-1199531521  52      M     106001.84        23.333333
## 4 ACC-1217063613  27      F      38908.96         7.666667
## 5 ACC-1222300524  38      M      52110.76        96.666667
## 6 ACC-1243371644  42      M      40357.92        24.666667
##   monthly_spending yearly_spending
## 1        2563.0900        30757.08
## 2        1210.5100        14526.12
## 3        1914.2367        22970.84
## 4         308.4933         3701.92
## 5        3556.2533        42675.04
## 6        1534.2533        18411.04
```

Plotting yearly salary against yearly spending for each customer and fitting a regression line, we see that there is a somewhat strong positive correlation between the two, with a correlation coefficient of 0.37.

Arguably, yearly spending is likely to be the best predictor of annual salary we have considered yet.

```
salary_metrics %>% ggplot(aes(x = yearly_spending, y = annual_salary)) + geom_point() +
  geom_smooth(method=lm) +
  theme(legend.position = "none") +
  labs(y = "Annual Salary",
       x = "Yearly Spending",
       title = 'Annual Salary vs Yearly Spending')
```



```r
cor(salary_metrics$annual_salary, salary_metrics$yearly_spending)
```

```
## [1] 0.3734772
```

Build a simple regression model to predict the annual salary for each customer using the attributes you identified above

Multiple Regression Model

We will fit a multiple regression model using the three metrics discussed - age, gender and spending - and then assess its accuracy.

First, gender must be recoded as a binary numeric variable. We will use '1' for male and '0' for female.

```r
salary_metrics$gender = ifelse(salary_metrics$gender == "M", 1, 0)
```

Now, we fit the model using 60% of the dataset as training data. This will allow us to test the predictive accuracy of the model using the remaining 40%.

```r
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 3.6.2
```

```r
set.seed(101)
sample = sample.split(names(salary_metrics), SplitRatio = 0.6)
train = subset(salary_metrics, sample == TRUE)
test  = subset(salary_metrics, sample == FALSE)

lin_mod = lm(annual_salary ~ age + gender + yearly_spending, data = train)
summary(lin_mod)
```

```
##
## Call:
## lm(formula = annual_salary ~ age + gender + yearly_spending,
##     data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -35913 -19669  -6304  11643  79491
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     58757.8228 14232.3433   4.128 0.000127 ***
## age              -254.7675   311.1831  -0.819 0.416553
## gender           1004.6403  6835.1130   0.147 0.883693
## yearly_spending     0.6319     0.3601   1.755 0.084950 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25020 on 54 degrees of freedom
## Multiple R-squared:  0.06874,    Adjusted R-squared:  0.017
## F-statistic: 1.329 on 3 and 54 DF,  p-value: 0.2746
```

As suggested by the initial tests for relationships between the three predictors and annual salary, age and gender are not significant predictors of annual salary with high p-values of 0.416553 and 0.883693 respectively. However, yearly spending is a highly significant predictor of annual salary with a lower p-value of 0.08. This suggests that it would be profitable to drop age and gender from the model and simply use yearly spending to predict annual salary.

How accurate is your model? Should ANZ use it to segment customers (for whom it does not have this data) into income brackets for reporting purposes?

Accuracy of Model

We will use two metrics to measure the predictive accuracy of the model - Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). We will test the accuracy predicting both in sample, using the original training set, and out of sample using the test set.

In Sample Accuracy:

```r
predictions_in = predict(lin_mod, newdata = train, interval = "prediction", level = 0.90)

mse_in = mean((train$annual_salary - predictions_in)^2)
sqrt(mse_in)
```

```
## [1] 42802.34
```

```
mae_in = mean(abs(train$annual_salary - predictions_in))
mae_in
```

```
## [1] 35926.01
```

Out of Sample Accuracy:

```
predictions_out = predict(lin_mod, newdata = test, interval = "prediction", level = 0.90)

mse_out = mean((test$annual_salary - predictions_out)^2)
sqrt(mse_out)
```

```
## [1] 44554.84
```

```
mae_out = mean(abs(test$annual_salary - predictions_out))
mae_out
```

```
## [1] 37332.19
```

In sample and out of sample RMSE are fairly similar, as are in and out of sample MAE, indicating that the model generalises fairly well to new data. However, both of these metrics appear quite large, indicating that there is generally quite a large prediction error using this model.

Upon reflection though, both of them are in the 35,000 to 45,000 range. As income is usually segmented into brackets of around $50,000, this is perhaps an acceptable level of error when attempting to predict a customer's income bracket. However, there is room for improvement, perhaps by finding other predictors with higher correlation and incorporating them into the model for better accuracy.

Decision Tree Based Model

For a challenge: build a decision-tree based model to predict salary. Does it perform better? How would you accurately test the performance of this model?

To fit a regression tree, we use the same formula as in the multiple regression, predicting annual salary using age, gender and yearly spending. We use the same training set to fit the model.

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.6.2
```

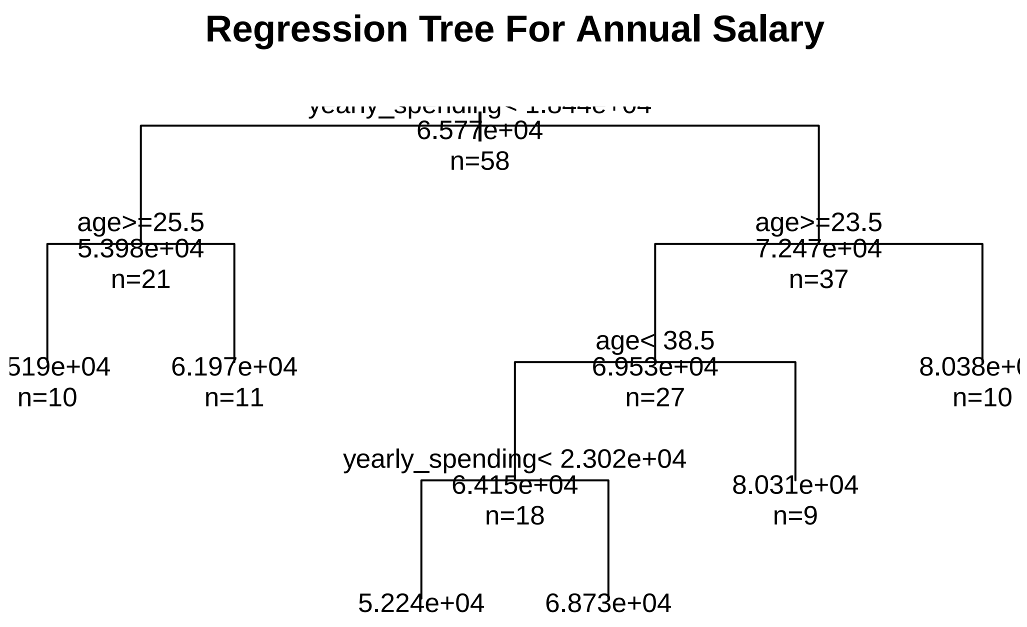
```
tree = rpart(annual_salary ~ age + gender + yearly_spending, data = train, method = 'anova', control = 

printcp(tree)
```

```
##
## Regression tree:
## rpart(formula = annual_salary ~ age + gender + yearly_spending,
##     data = train, method = "anova", control = rpart.control(minsplit = 14))
##
## Variables actually used in tree construction:
```

```
## [1] age             yearly_spending
##
## Root node error: 3.6294e+10/58 = 625753558
##
## n= 58
##
##         CP nsplit rel error xerror    xstd
## 1 0.126144      0   1.00000 1.0545 0.21697
## 2 0.040649      1   0.87386 1.1695 0.24205
## 3 0.033414      2   0.83321 1.3500 0.25925
## 4 0.027027      4   0.76638 1.3010 0.24365
## 5 0.010000      5   0.73935 1.2827 0.24296
```

Specifying that the minimum number of observations in a node be 14 results in a tree that splits annual salary into six values, which seems reasonable as this is similar to the number of brackets the ATO splits income into. This also seems reasonable as it suggests we are not over fitting the model to the data.

```
plot(tree, uniform=TRUE,
   main="Regression Tree For Annual Salary")
text(tree, use.n=TRUE, all=TRUE, cex=.8)
```

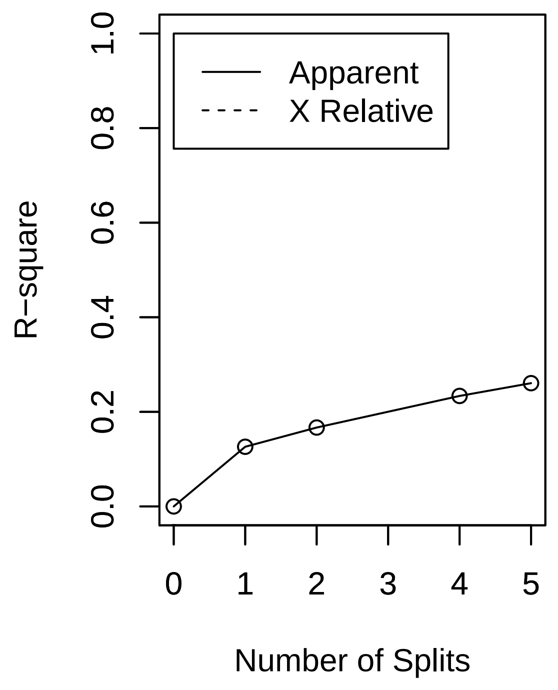


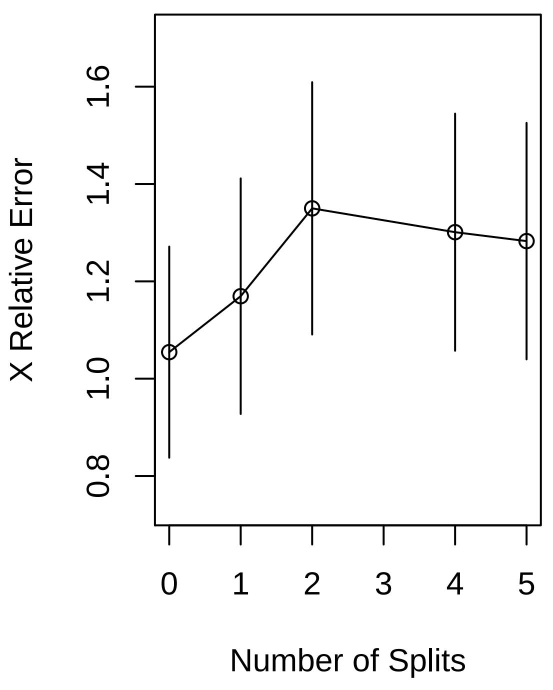
Accuracy of Model

A plot of the approximate R-squared and relative error show that R-squared (measure of total variance explained by model) increases with the number of splits as expected, and error seems to settle at 1.5 with 5 splits.

```r
par(mfrow=c(1,2)) # two plots on one page
rsq.rpart(tree)
```

```
##
## Regression tree:
## rpart(formula = annual_salary ~ age + gender + yearly_spending,
##     data = train, method = "anova", control = rpart.control(minsplit = 14))
##
## Variables actually used in tree construction:
## [1] age             yearly_spending
##
## Root node error: 3.6294e+10/58 = 625753558
##
## n= 58
##
##         CP nsplit rel error xerror    xstd
## 1 0.126144      0   1.00000 1.0545 0.21697
## 2 0.040649      1   0.87386 1.1695 0.24205
## 3 0.033414      2   0.83321 1.3500 0.25925
## 4 0.027027      4   0.76638 1.3010 0.24365
## 5 0.010000      5   0.73935 1.2827 0.24296
```





As before, we measure in sample accuracy by predicting annual salary from the training set and out of sample accuracy by predicting annual salary from the test set. We use RMSE and MAE as measures of predictive accuracy.

In Sample Accuracy:

```r
tree_pred_in = predict(tree, newdata = train, interval = "prediction", level = 0.90)

tree_mse_in = mean((train$annual_salary - tree_pred_in)^2)
sqrt(tree_mse_in)
```

```
## [1] 21509.35
```

```r
tree_mae_in = mean(abs(train$annual_salary - tree_pred_in))
tree_mae_in
```

```
## [1] 17448.14
```

Out of Sample Accuracy:

```r
tree_pred_out = predict(tree, newdata = test, interval = "prediction", level = 0.90)

tree_mse_out = mean((test$annual_salary - tree_pred_out)^2)
sqrt(tree_mse_out)
```

```
## [1] 27745.39
```

```r
tree_mae_out = mean(abs(test$annual_salary - tree_pred_out))
tree_mae_out
```

```
## [1] 22634.93
```

Both RMSE and MAE are fairly similar in and out of sample, and furthermore they are both significantly lower for this model than for the liner model, in the range between 15,000 and 30,000. This suggests that as suspected, the relationships of annual salary with some of the predictors being used (such as age) were not actually linear. So using a non-linear tree-based method has yielded a more accurate predictive model.

Conclusion

The tree based model has greater predictive accuracy, so it is the recommended model for ANZ to segment customers into income brackets for reporting purposes. Further gains in accuracy may be possible through techniques like pruning, cross validation, experimenting with parameters (e.g.. number of observations needed for a split, different cost complexity factor values), and consensus tree-based models like random forest.