Simple Linear Regression

2025-07-18

Contents

1 Load the Dataset

```
if (!"pacman" %in% installed.packages()[, "Package"]) {
install.packages("pacman", dependencies = TRUE)
library("pacman", character.only = TRUE)
}
pacman::p_load("here")
knitr::opts_knit$set(root.dir = here::here())
```

```
pacman::p_load("readr")
clv_data <- read_csv("./data/clv_data.csv")
```

```
## Rows: 500 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (2): purchase_frequency, customer_lifetime_value
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(clv_data)
```

```
## # A tibble: 6 x 2
##   purchase_frequency customer_lifetime_value
##                <dbl>                   <dbl>
## 1                  3                   110.
## 2                  7                   190.
## 3                  6                   160.
## 4                  2                    94.4
## 5                  4                   133.
## 6                  8                   223.
```

2 View the Data Types

```
sapply(clv_data, class)
```

```
##      purchase_frequency customer_lifetime_value
##               "numeric"               "numeric"
```

```
str(clv_data)
```

```
## spc_tbl_ [500 x 2] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ purchase_frequency     : num [1:500] 3 7 6 2 4 8 0 4 8 3 ...
##  $ customer_lifetime_value: num [1:500] 110.3 190.2 160 94.4 133.2 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   purchase_frequency = col_double(),
```

```
##   ..   customer_lifetime_value = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
summary(clv_data)
```

```
##  purchase_frequency customer_lifetime_value
##  Min.   :-1.000     Min.   : 26.13
##  1st Qu.: 4.000     1st Qu.:122.04
##  Median : 5.000     Median :148.21
##  Mean   : 4.914     Mean   :148.25
##  3rd Qu.: 6.000     3rd Qu.:175.88
##  Max.   :11.000     Max.   :262.04
```

3 Variance:

```
#'sapply()' is designed to apply a function to a variable in a dataset
#In this case, I used 'sapply()' to apply the 'var()' function used to compute the variance.
#High variability means that the values are less consistent, thus making it harder to make predictions.
sapply(clv_data[,], var)
```

```
##      purchase_frequency customer_lifetime_value
##                4.146898             1642.315996
```

4 Standard Deviation:

```
sapply(clv_data[,],sd)
```

```
##      purchase_frequency customer_lifetime_value
##                2.036393               40.525498
```

5 Kurtosis

```
#Informs how often outliers occur
#Different formulas for calculating hence we specify type 2 which is used in other software
#Kurtosis = 3 -> medium no. of outliers
#Kurtosis<3 -> low no. of ouliers and vice versa
pacman::p_load("e1071")
sapply(clv_data[,],kurtosis, type=2)
```

```
##      purchase_frequency customer_lifetime_value
##              -0.1220038             -0.1484811
```

6 Skewness

```
#Used to ID the asymmetry of distribution of results
#Similar to kurtosis we have type 2 which is widely used by other apps :)
#-0.4<Skewness<0.4 inclusive implies no skew i.e it is a normal distribution
#Above 0.4 implies +ve skew
#below -0.4 implies -ve skew: a left-skewed distribution
sapply(clv_data[,], skewness, type = 2)
```

```
##      purchase_frequency customer_lifetime_value
##             -0.04021915             -0.01608242
```

7 Covariance

```
#Indicates the direction of the linear relationship betweeen 2 variables
#Assesses whether increase in one leads to an increase in the other
#+ve covariance -> when one increases the other increases
#-ve  covariance -> when one increases the other decreases
#Zero covariance -> no relationship
#Shows direction of relationship but not strength
cov(clv_data, method = "spearman")
```

```
##                         purchase_frequency customer_lifetime_value
## purchase_frequency                 20409.91                20235.73
## customer_lifetime_value            20235.73                20874.99
```

8 Correlation

```
#Strong correlation enables better prediction of independent variable
#Only useful if there is linear association/strong correlation
#Spearman's rank correlation rho is used to measure statistical significance of the correlation
#Monotomic relationship -> one var increases and the other either increases consistently or consistentl
#Rate of change may vary but direction is preserved
cor.test(clv_data$customer_lifetime_value, clv_data$purchase_frequency, method = "spearman")
```

```
##
##  Spearman's rank correlation rho
##
## data:  clv_data$customer_lifetime_value and clv_data$purchase_frequency
## S = 409190, p-value < 2.2e-16
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.9803588
```

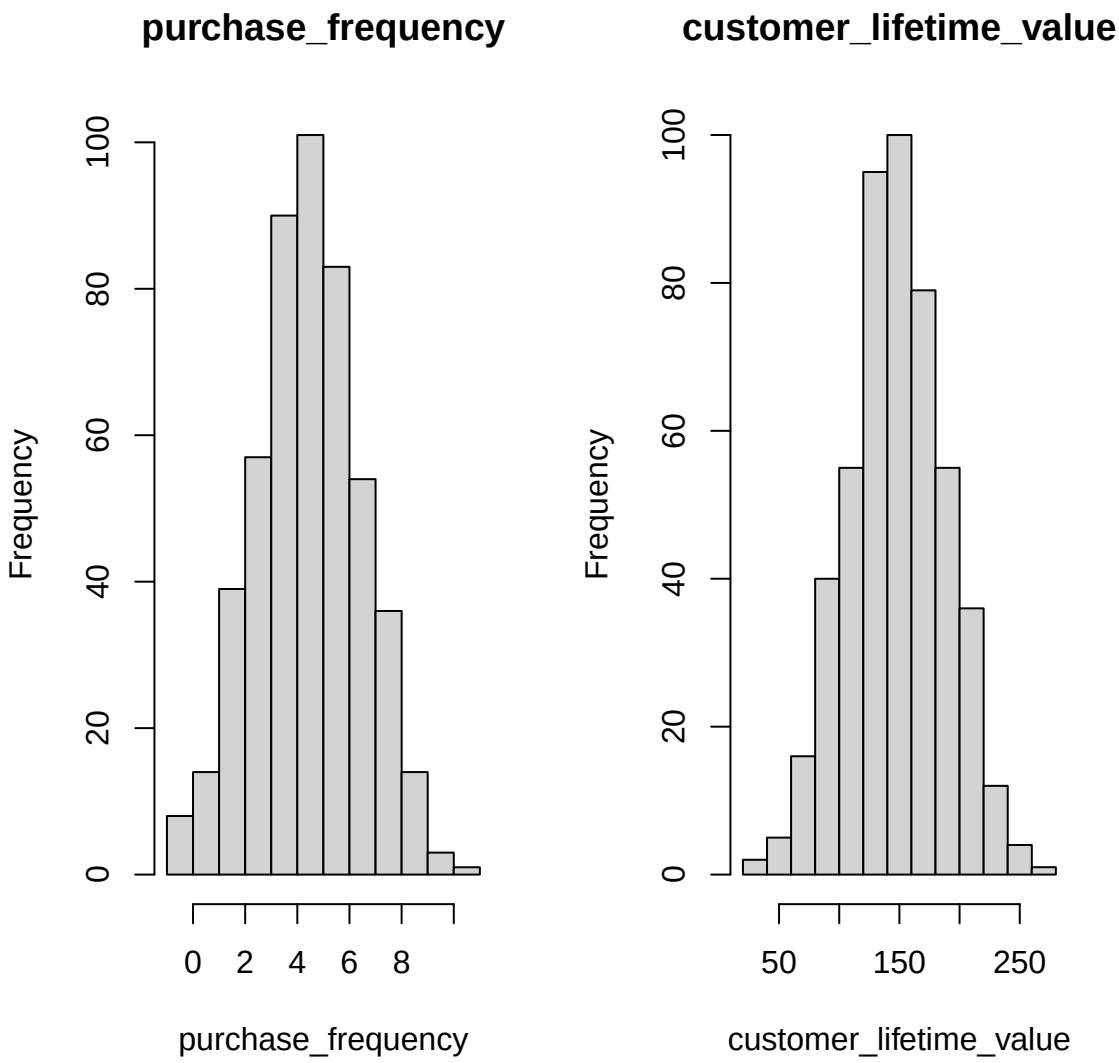
To view correlation of all variables

```
cor(clv_data, method = "spearman")
```

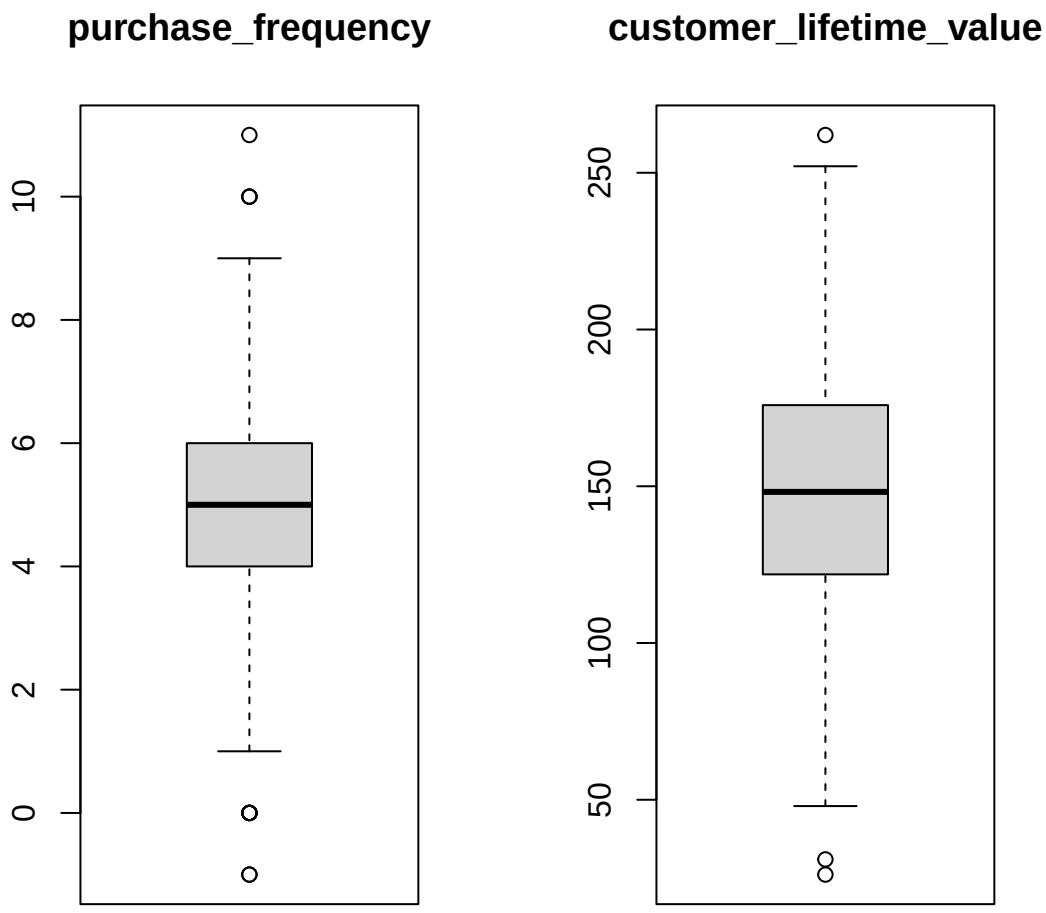
```
##                          purchase_frequency customer_lifetime_value
## purchase_frequency               1.0000000               0.9803588
## customer_lifetime_value          0.9803588               1.0000000
```

9 Basic Visualizations

```
# par(mfrow = c(1, 2)) This  is used to divide the area used to plot the visualization into a 1 row by
# for (i in 1:2) This is used to identify the variable (column) that is being processed
# clv_data[[i]] This is used to extract the i-th column as a vector
# hist() This is the fnctn used to plot the histogram
par(mfrow = c(1, 2))
for (i in 1:2) {
  if (is.numeric(clv_data[[i]])){
    hist(clv_data[[i]],
         main = names(clv_data)[i],
         xlab = names(clv_data)[i])
  } else {
    message(paste("Column", names(clv_data)[i], "is not numeric and will be skipped"))
  }
}
```

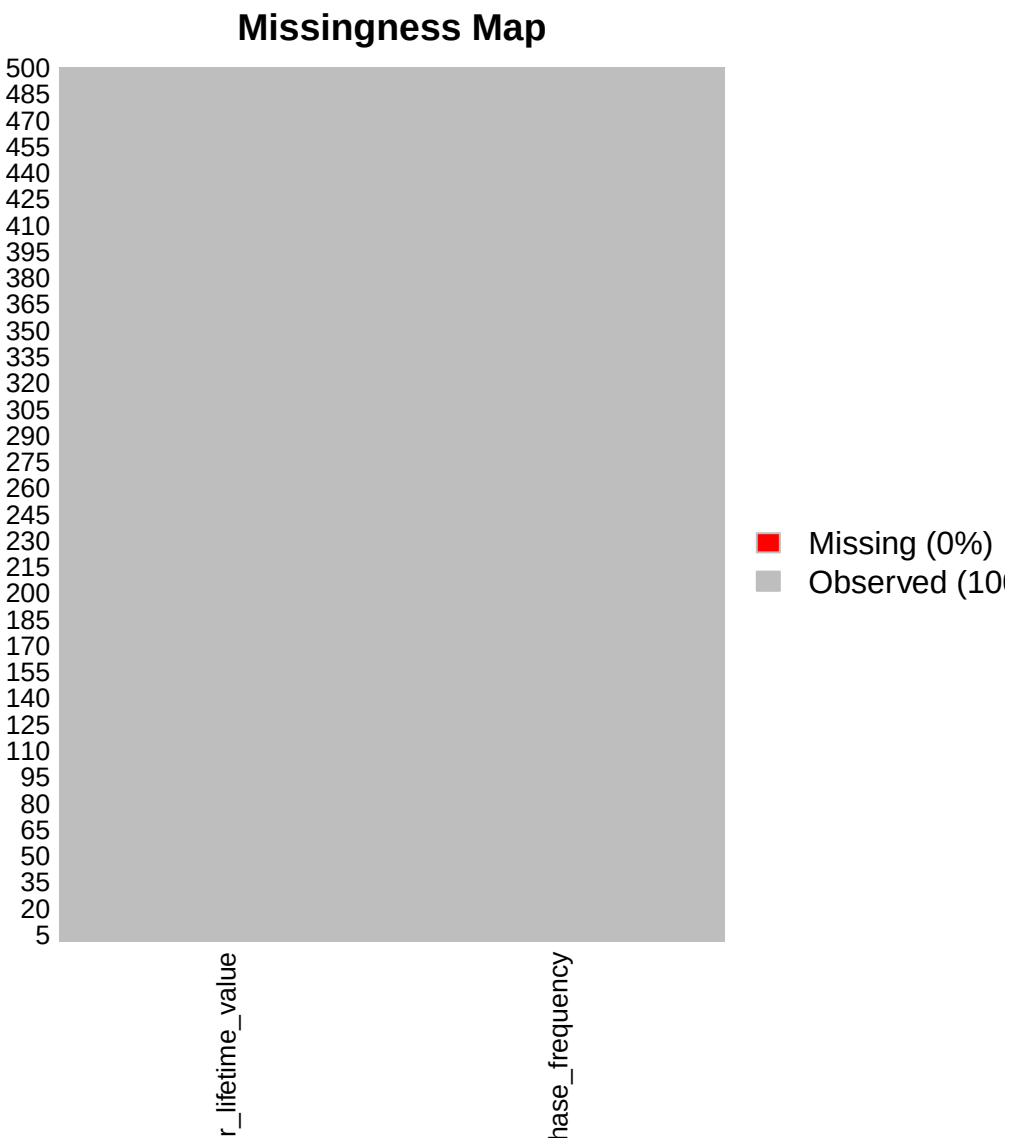



```
par(mfrow = c(1, 2))
for (i in 1:2) {
  if (is.numeric(clv_data[[i]])) {
    boxplot(clv_data[[i]], main = names(clv_data)[i])
  } else {
    message(paste("Column", names(clv_data)[i], "is not numeric and will be skipped"))
  }
}
```



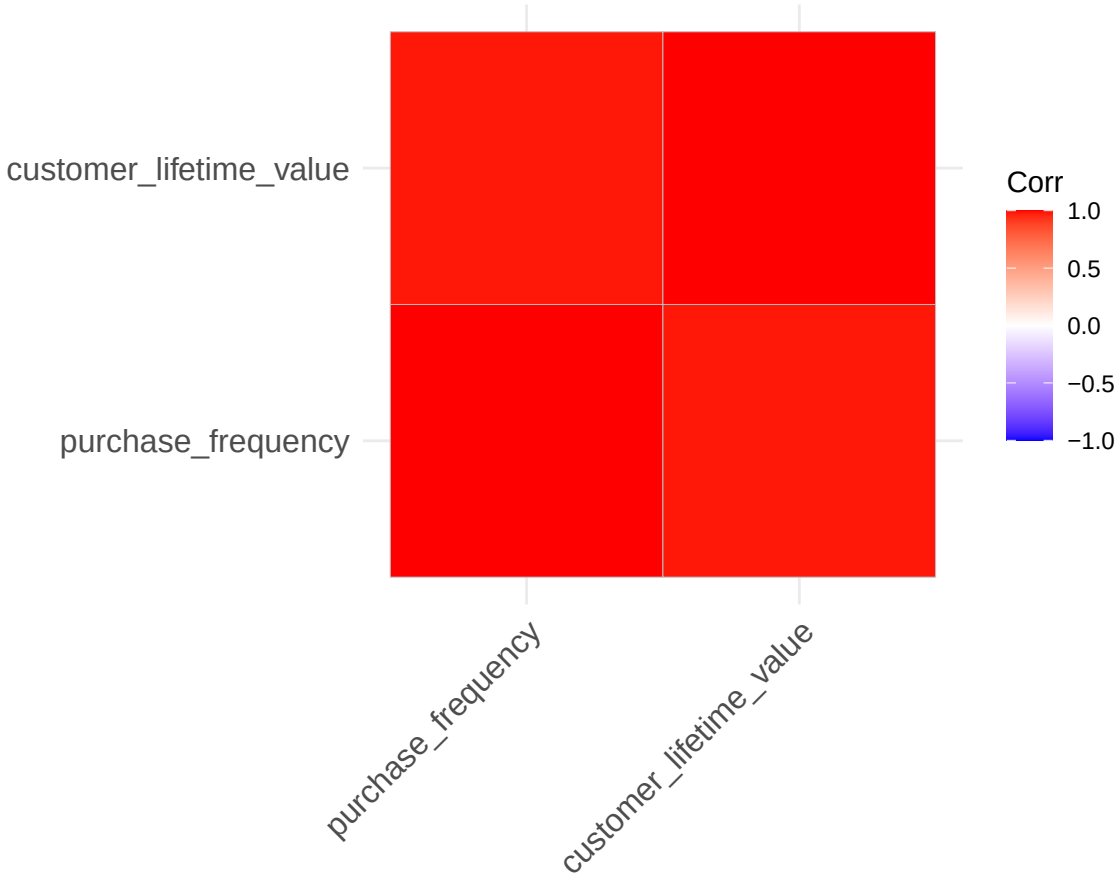

```
pacman::p_load("Amelia")

missmap(clv_data, col = c("red", "grey"), legend = TRUE)
```



10 Correlation Plot

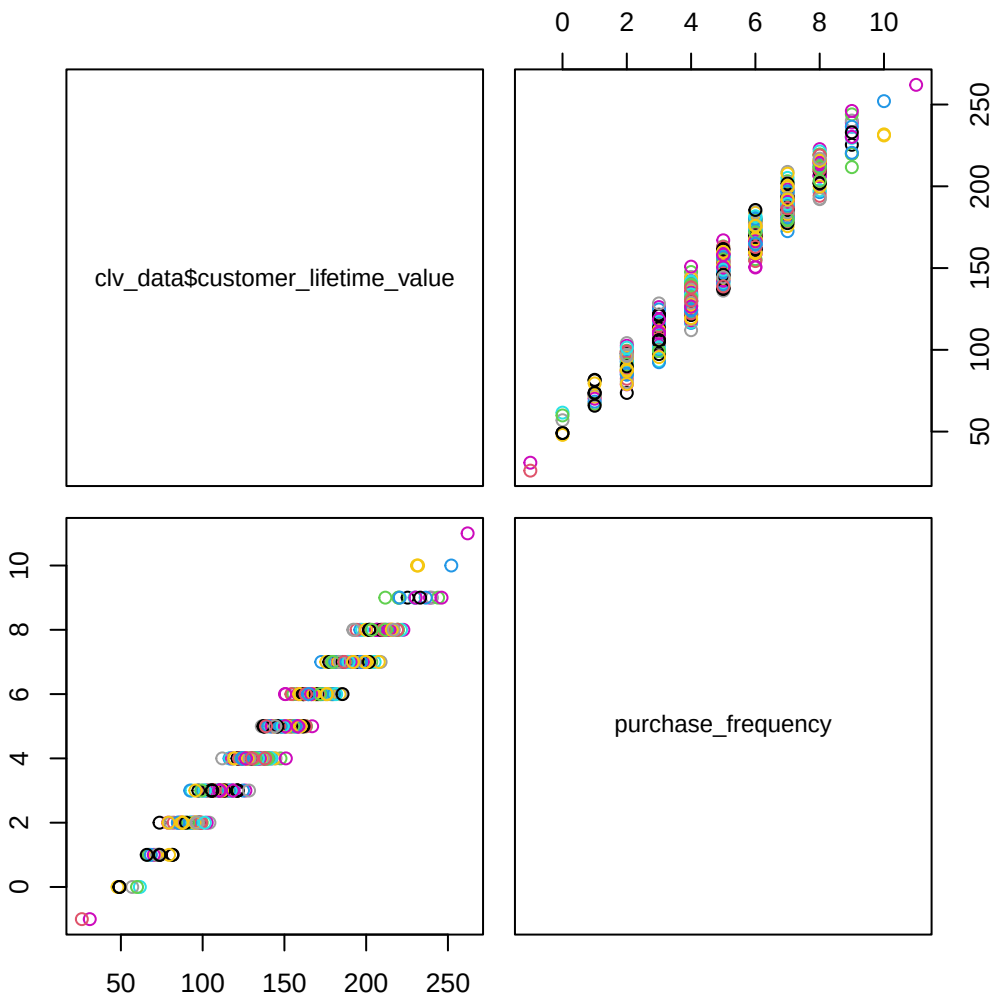
```
pacman::p_load("ggcorrplot")
ggcorrplot(cor(clv_data[,]))
```



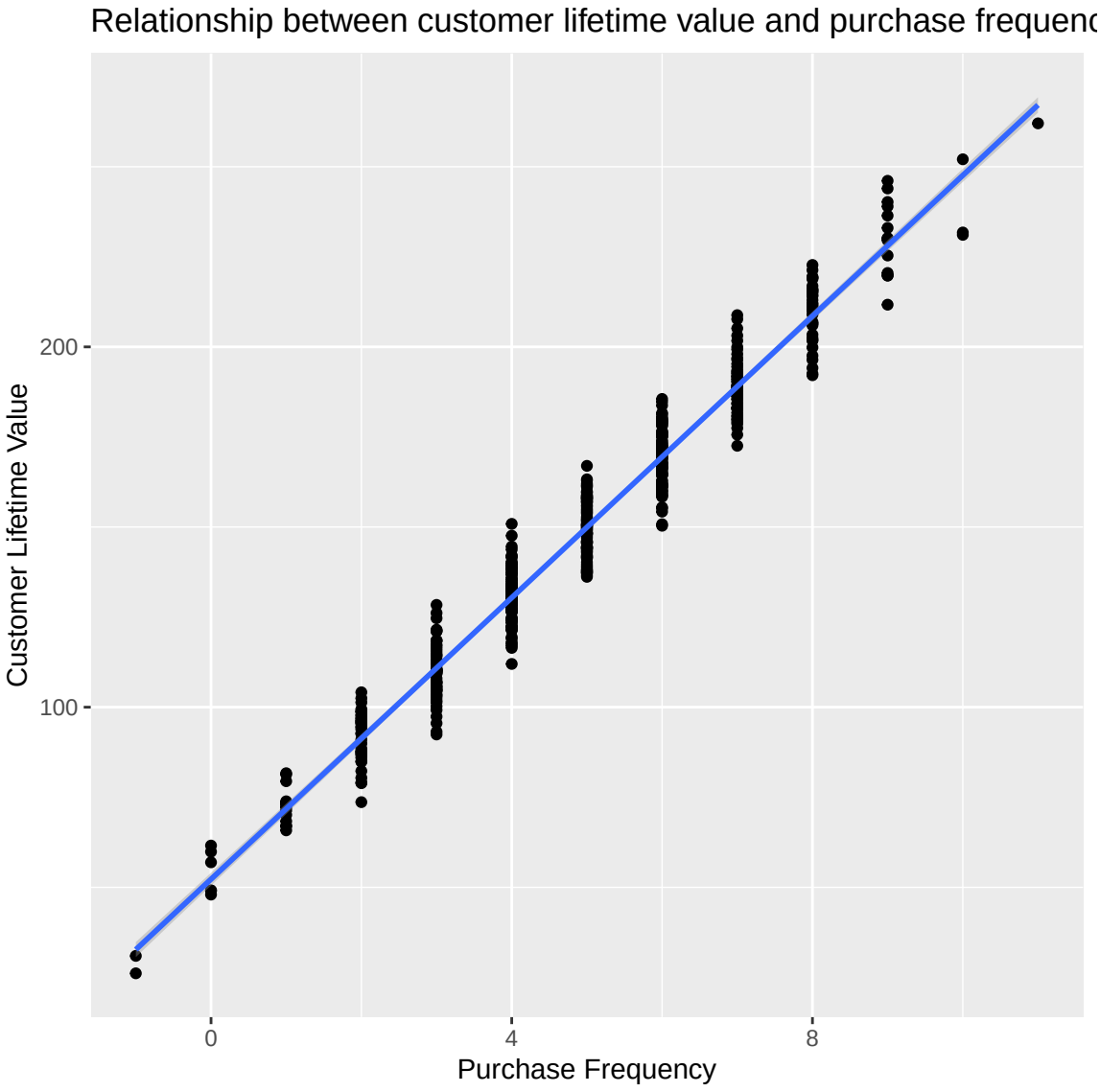
11 Scatter Plot

```
pacman::p_load("corrplot")

pairs(clv_data$customer_lifetime_value ~ . , data = clv_data, col = clv_data$customer_lifetime_value)
```




```
pacman::p_load("ggplot2")
ggplot(clv_data,
       aes(x = purchase_frequency, y = customer_lifetime_value)) +
  geom_point() +
geom_smooth(method = lm) +
  labs(
    title = "Relationship between customer lifetime value and purchase frequency",
    x = "Purchase Frequency",
    y = "Customer Lifetime Value"
  )
```



12 Statistical test of Linear Regression

```
slr_test <- lm(customer_lifetime_value ~ purchase_frequency, data = clv_data)

#To view result
summary(slr_test)
```

```
##
## Call:
## lm(formula = customer_lifetime_value ~ purchase_frequency, data = clv_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.1176  -5.6169  -0.0491   5.6618  20.4837
```

```
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          52.2538     0.9042   57.79   <2e-16 ***
## purchase_frequency  19.5356     0.1700  114.91   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.734 on 498 degrees of freedom
## Multiple R-squared:  0.9637, Adjusted R-squared:  0.9636
## F-statistic: 1.32e+04 on 1 and 498 DF,  p-value: < 2.2e-16
```

```
#Obtain confidence intervals
confint(slr_test, level = 0.95)
```

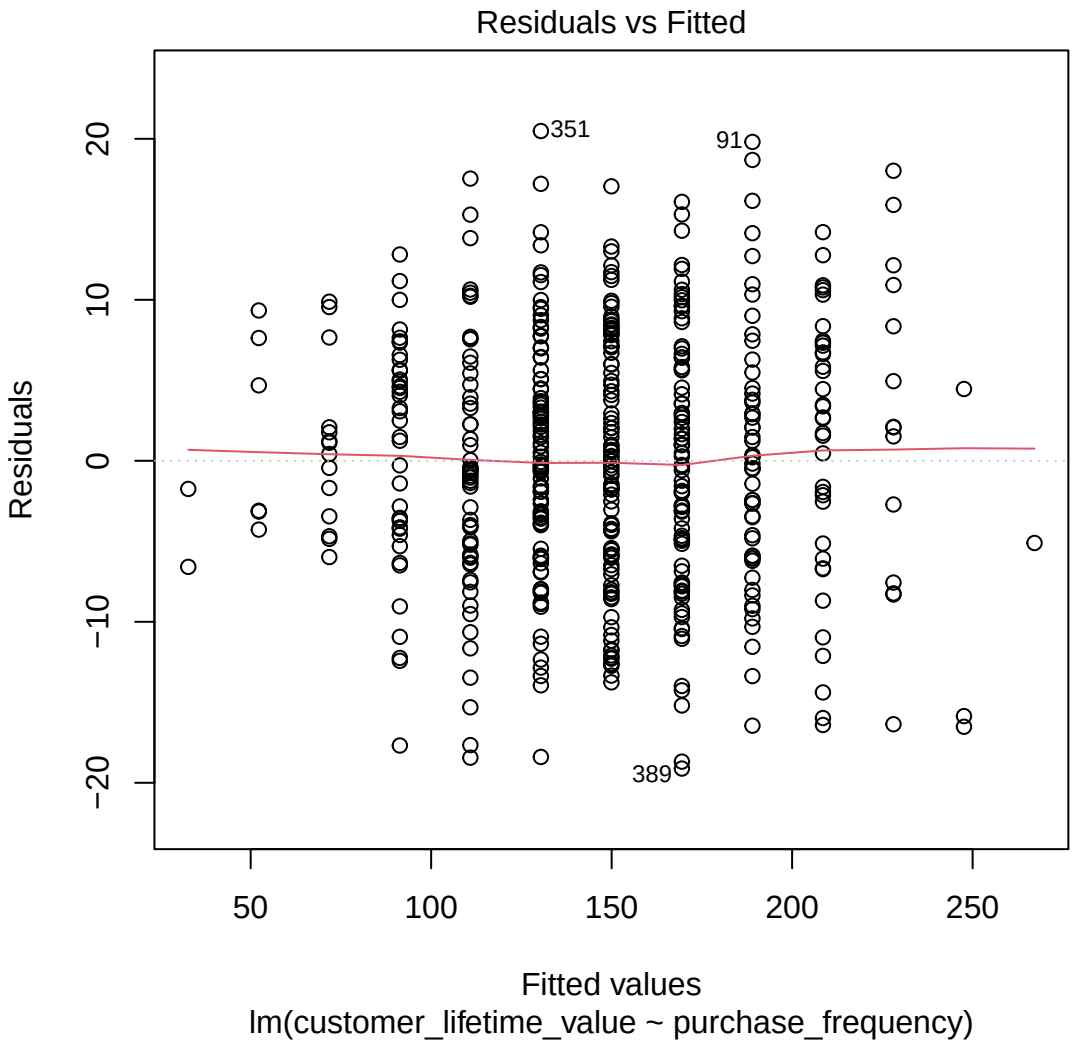
```
##                       2.5 %   97.5 %
## (Intercept)        50.47731 54.03036
## purchase_frequency 19.20159 19.86965
```

13 Diagnostic EDA

Diagnostic EDA tests validity of the model's assumptions before interpreting results. This helps prevent incorrect conclusions

13.1 Test of Linearity

```
plot(slr_test, which = 1)
```




```
# Tests whether relationship between dependent and independent variables is linear
# A plot of residuals vs fitted values enables test for linearity
# For the model to pass there should be no pattern in the distribution of residuals and the residuals s
# i.e the residuals should randomly vary around the mean of the value of the response variable
```

13.2 Test of Independence of Errors

This test is necessary to confirm each observation is independent of each other.

It helps to identify autocorrelation which occurs when data is collected over a close period of time or when an observation is related to another.

Autocorrelation leads to underestimated standard errors and inflated t-statistics / findings appear bigger than they actually are.

Durbin Watson Test

- H0 -> There is no autocorrelation (null hypothesis)

- H1 -> There is autocorrelation

If the p-value > 5, no evidence to reject null hypothesis "There is no autocorrelation"

```
pacman::p_load("lmtest")
dwtest(slr_test)
```

```
##
##  Durbin-Watson test
##
## data:  slr_test
## DW = 1.9104, p-value = 0.1573
## alternative hypothesis: true autocorrelation is greater than 0
```

```
#The results show a p-value of 0.1573 therefore the test of independence of errors around the regressio
```

13.3 Test of Normality

It assesses whether the residuals are normally distributed i.e most residuals(errors) are close to zero and large errors are rare

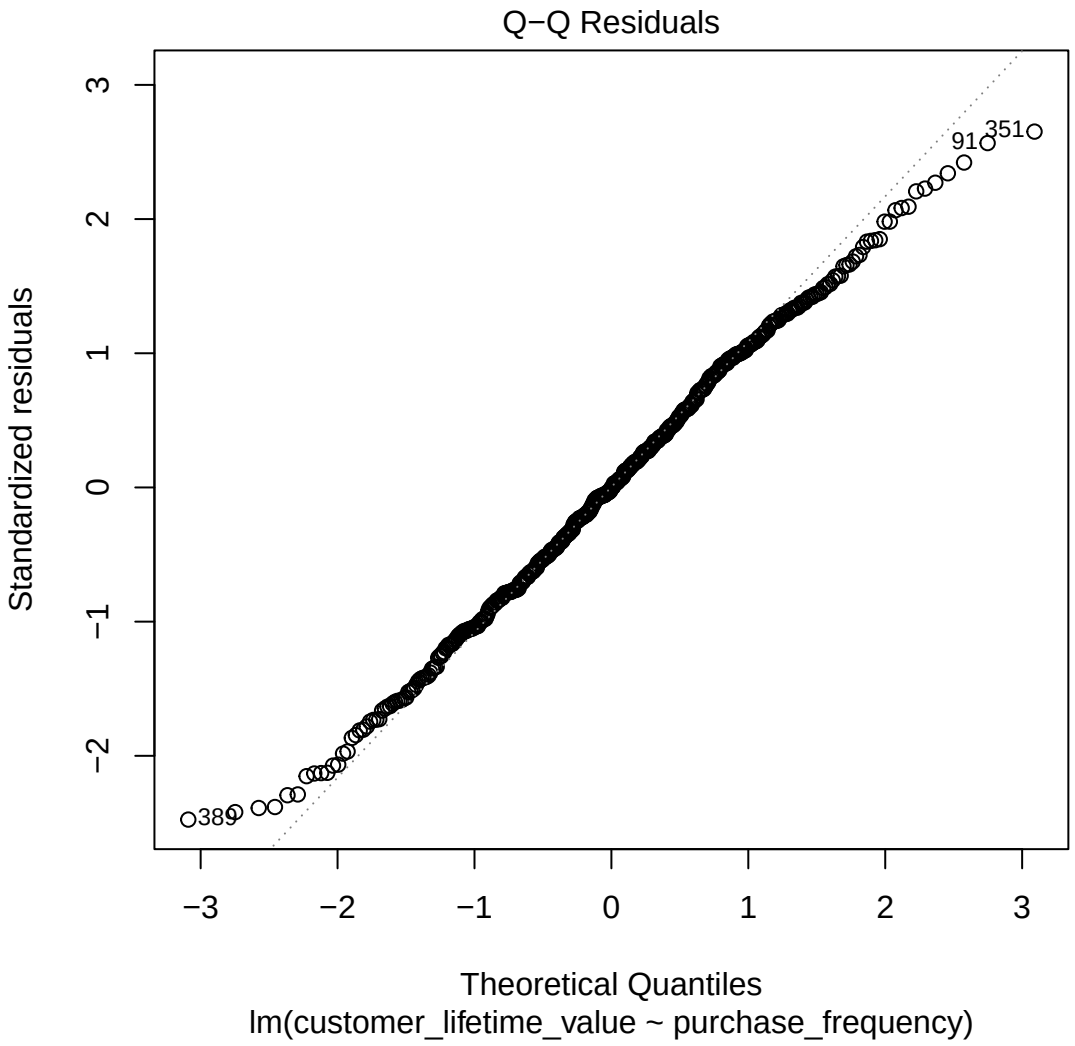
A Q-Q plot can be used for this

It is a scatter-plot of the quantities of the residuals against quantiles of a normal distribution

Quantiles are statistical values that divide a data set or probability into equal-sized intervals e.g quartiles, percentiles, deciles(10 equal parts) etc

If the points in the plot fall along a straight line, then the normality assumption is satisfied.

```
plot(slr_test, which = 2)
```



14 Test of Homoscedasticity

Homoscedasticity requires that the spread of residuals should be constant across all levels of the independent variable. A scale-location plot (a.k.a. spread-location plot) can be used to conduct a test of homoscedasticity.

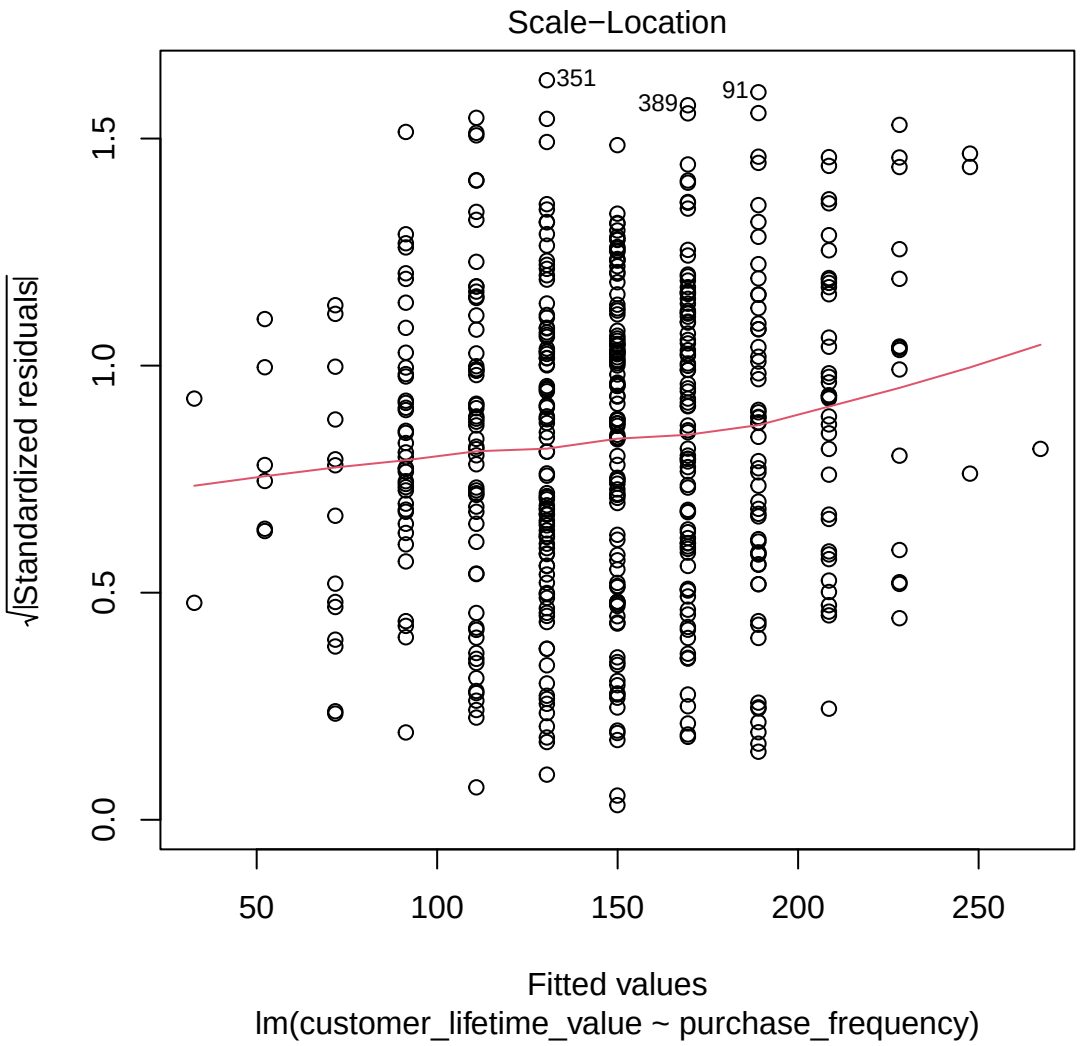
The x-axis shows the fitted (predicted) values from the model and the y-axis shows the square root of the standardized residuals. The red line is added to help visualize any patterns.

In a model with homoscedastic errors (equal variance across all predicted values):

- Points should be randomly scattered around a horizontal line
- The smooth line should be approximately horizontal
- The vertical spread of points should be roughly equal across all fitted values
- No obvious patterns, funnels, or trends should be visible

Points forming a cone shape that widens from left to right suggests heteroscedasticity with increasing variance for larger fitted values.

```
plot(slr_test, which = 3)
```



15 Quantitative Validation of Assumptions

The graphical representations of the various tests of assumptions should be accompanied by quantitative values. The gvlma package(Global Validation of Linear Models Assumptions) is useful for this purpose.

```
pacman::p_load("gvlma")
gvlma_results <- gvlma(slr_test)
summary(gvlma_results)
```

```
##
```

```
## Call:
## lm(formula = customer_lifetime_value ~ purchase_frequency, data = clv_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.1176  -5.6169  -0.0491   5.6618  20.4837
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         52.2538     0.9042   57.79   <2e-16 ***
## purchase_frequency  19.5356     0.1700  114.91   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.734 on 498 degrees of freedom
## Multiple R-squared:  0.9637, Adjusted R-squared:  0.9636
## F-statistic: 1.32e+04 on 1 and 498 DF,  p-value: < 2.2e-16
##
##
## ASSESSMENT OF THE LINEAR MODEL ASSUMPTIONS
## USING THE GLOBAL TEST ON 4 DEGREES-OF-FREEDOM:
## Level of Significance =  0.05
##
## Call:
##  gvlma(x = slr_test)
##
##                      Value p-value                Decision
## Global Stat        5.08943 0.27824 Assumptions acceptable.
## Skewness           0.03973 0.84201 Assumptions acceptable.
## Kurtosis           3.61252 0.05735 Assumptions acceptable.
## Link Function      0.01459 0.90385 Assumptions acceptable.
## Heteroscedasticity 1.42258 0.23298 Assumptions acceptable.
```