Group 205 Final Project

Seonhye Yang

8/3/2019

```
library(readr, warn.conflicts = F, quietly = T)
library(plotly, warn.conflicts = F, quietly = T)
```

INTRODUCTION

The data we are going to use involves prices of different stocks (Microsoft, Apple and Tesla). The important factor about these stock data sets is the price of stock for different days. All of the datasets include a Date (day), Open (initial price when market opens), High (highest price during the day), Low (lowest price during the day), Close (closing price when market closes), Adj.close (final price of a stock at the end of the day when banks unload their positions and adjusted price count for this) and volume (number of stocks sold in one day). The datasets are interesting because they give us different information about stock prices during the day. Stocks are time sensitive which means the price is constantly changing throughout the day before the market closes. Determining the future value of a company's stock is important because it could yield a significant amount of profit. One way to accomplish this is to use a linear regression model.

METHOD

For this project, we split the models into 3 month intervals for accuracy. Specifically, the smaller the input date range, the more accurate we will be since we are working on a smaller time scale (smaller noise).

The ANOVA test compares the predictive power of our AIC-selected models the predictive power of the full model. It is important to avoid models with too many variables because it may result in overfitting. For this project, we did not take out any outliers or influential points because these points indicate future movements in the market (outliers are common in trading). To select our model we followed a standard practice method. After we had segmented our data, we then went to create a full model for every segment or split. With this full model in hand, we applied a backwards AIC to the full model for each split. This is important, as it crafts a custom model for each split of data we have. This means that we can correctly fit our model depending on market conditions. For example, in times of high volume trading, where the price was moved simply by amount of trades, and not signals such as news, we can see that the step AIC included volume and price. But in more news oriented situations, the open and high price were included, as they represent the current state of the stock, and are more influential to the future value. With a model in hand for each spilt, we can then perform validation, and predictions with our model.

Our goal is to create a model that can predict the stock price for the following day. If you look at the dataset of MSFT_stock (Microsoft) and AAPL_stock (Apple), there's a column called "Adj.close" which stands for Adjusted Close (this is the final stock price at the end of the day). Our model predicts the final stock for the following day and in order to achieve this, we shifted up the Adj.close by one day because we want to predict the stock price for the following day. To be specific, if Adj.close is $350 on the date 2018-01-03, we would place $350 on the date 2018-01-02 in the new column, Target, and vice versa.

```
generateData <- function(dataRaw, parts=60) {
  # Create target column
  dataRaw$Target <- dataRaw$Adj.Close
  # Shift target column
  for (i in 1:nrow(dataRaw) - 1){
    dataRaw$Target[i] = dataRaw$Adj.Close[i + 1]
  }
  # Remove date column
  dataNoDate <- within(dataRaw, rm("Date"))
  # Split the data table
```

```
  splitData <- split(dataNoDate, rep(1:ceiling(nrow(dataNoDate)/parts), each=parts, length.out=nrow(data
  return(splitData)
}

MSFT <- read.csv("MSFT.csv")
head(MSFT)
```

```
##         Date  Open  High   Low Close Adj.Close   Volume
## 1 2018-01-02 86.13 86.31 85.50 85.95  83.81786 22483800
## 2 2018-01-03 86.06 86.51 85.97 86.35  84.20794 26061400
## 3 2018-01-04 86.59 87.66 86.57 87.11  84.94909 21912000
## 4 2018-01-05 87.66 88.41 87.43 88.19  86.00230 23407100
## 5 2018-01-08 88.20 88.58 87.60 88.28  86.09006 22113000
## 6 2018-01-09 88.65 88.73 87.86 88.22  86.03155 19484300
```

```
msftSplit <-generateData(MSFT)
head(msftSplit[1])
```

```
## $`1`
##     Open  High   Low Close Adj.Close   Volume   Target
## 1  86.13 86.31 85.50 85.95  83.81786 22483800 84.20794
## 2  86.06 86.51 85.97 86.35  84.20794 26061400 84.94909
## 3  86.59 87.66 86.57 87.11  84.94909 21912000 86.00230
## 4  87.66 88.41 87.43 88.19  86.00230 23407100 86.09006
## 5  88.20 88.58 87.60 88.28  86.09006 22113000 86.03155
## 6  88.65 88.73 87.86 88.22  86.03155 19484300 85.64147
## 7  87.86 88.19 87.41 87.82  85.64147 18652200 85.89502
## 8  88.13 88.13 87.24 88.08  85.89502 17808900 87.37733
## 9  88.67 89.78 88.45 89.60  87.37733 24271500 86.15833
## 10 90.10 90.79 88.01 88.35  86.15833 36599700 87.90393
## 11 89.08 90.28 88.75 90.14  87.90393 25621200 87.86492
## 12 89.80 90.67 89.66 90.10  87.86492 24159700 87.76739
## 13 90.14 90.61 89.66 90.00  87.76739 36875000 89.33746
## 14 90.00 91.62 89.74 91.61  89.33746 23601600 89.62026
## 15 91.90 92.30 91.54 91.90  89.62026 23412800 89.54225
## 16 92.55 93.43 91.58 91.82  89.54225 33277500 90.03960
## 17 92.47 93.24 91.93 92.33  90.03960 26383200 91.72669
## 18 93.12 94.06 92.58 94.06  91.72669 29172200 91.59016
## 19 95.14 95.45 93.72 93.92  91.59016 31569900 90.43942
## 20 93.30 93.66 92.10 92.74  90.43942 38635100 92.65311
## 21 93.75 95.40 93.51 95.01  92.65311 48756300 91.92172
## 22 94.79 96.07 93.58 94.26  91.92172 47227900 89.50324
## 23 93.64 93.97 91.50 91.78  89.50324 47867800 85.81701
## 24 90.56 93.24 88.00 88.00  85.81701 51031500 89.06441
## 25 86.89 91.48 85.25 91.33  89.06441 67998600 87.38708
## 26 90.49 91.77 89.20 89.61  87.38708 41107600 82.90118
## 27 89.71 89.88 84.76 85.01  82.90118 55628700 85.99255
## 28 86.30 88.93 83.83 88.18  85.99255 63499100 86.91898
## 29 88.74 89.78 87.93 89.13  86.91898 35720300 87.60161
## 30 88.93 90.00 87.80 89.83  87.60161 26407700 88.97330
## 31 88.51 90.99 88.41 90.81  88.97330 34960900 90.78590
## 32 91.21 92.72 90.62 92.66  90.78590 27823900 90.13923
## 33 92.45 93.50 91.80 92.00  90.13923 30596900 90.84467
## 34 91.48 93.06 91.01 92.72  90.84467 30911700 89.63955
```

```
## 35 92.98 93.36 91.49 91.49  89.63955 26922500 89.87470
## 36 92.05 92.73 91.36 91.73  89.87470 24392800 92.15757
## 37 93.60 94.07 92.36 94.06  92.15757 26329200 93.49005
## 38 94.40 95.45 94.25 95.42  93.49005 30199800 92.29473
## 39 95.74 95.84 94.20 94.20  92.29473 25869100 91.87344
## 40 94.84 95.71 93.63 93.77  91.87344 31167300 90.97203
## 41 93.99 94.57 91.84 92.85  90.97203 37135600 91.16800
## 42 91.58 93.15 90.86 93.05  91.16800 32830400 91.74606
## 43 92.34 94.27 92.26 93.64  91.74606 23901600 91.43253
## 44 94.34 94.49 92.94 93.32  91.43253 22175800 91.96162
## 45 93.16 93.94 92.43 93.86  91.96162 26716100 92.52008
## 46 94.27 95.10 93.77 94.43  92.52008 25887800 94.58741
## 47 95.29 96.54 95.00 96.54  94.58741 36937300 94.81275
## 48 96.50 97.21 96.04 96.77  94.81275 26073700 92.50050
## 49 97.00 97.24 93.97 94.41  92.50050 35387800 91.95181
## 50 95.12 95.41 93.50 93.85  91.95181 32132000 92.27515
## 51 93.53 94.58 92.83 94.18  92.27515 27611000 92.68665
## 52 94.68 95.38 93.92 94.60  92.68665 49081300 91.01122
## 53 93.74 93.90 92.11 92.89  91.01122 33344100 91.24638
## 54 93.05 93.77 93.00 93.13  91.24638 23075200 90.60952
## 55 92.93 94.05 92.21 92.48  90.60952 24457100 87.97393
## 56 91.27 91.75 89.66 89.79  87.97393 38604700 85.41672
## 57 89.50 90.46 87.08 87.18  85.41672 44068900 91.88322
## 58 90.61 94.00 90.40 93.78  91.88322 56396800 87.66041
## 59 94.94 95.14 88.51 89.47  87.66041 56569000 87.58202
## 60 89.82 91.23 88.87 89.39  87.58202 52501100 89.42400
```

```
AAPL <- read.csv("AAPL.csv")
aaplSplit <-generateData(AAPL)

TSLA <- read.csv("TSLA.csv")
tslaSplit <-generateData(TSLA)
```

```
generateModels <- function(inputData) {
  # Make an empty list
  ret <- c()
  retOrig <- c()
  # For every split in our dataset
  for (i in 1:length(inputData)) {
    # Create the full model for this split
    model <- lm(Target ~ ., data = inputData[[i]])
    retOrig <- c(retOrig, list(model))
    # Run step to find what the best model for this split is, and add it to the list
    stepModel <- step(model, direction = "backward", trace = FALSE)
    ret <- c(ret, list(stepModel))
  }
  # Return the list
  v <- list(initial=retOrig, step=ret)
  return(v)
}


modelsMsft <- generateModels(msftSplit)
modelsAapl <- generateModels(aaplSplit)
modelsTsla <- generateModels(tslaSplit)
```

```
generateAnova <- function(inputmodels){
  for (i in 1:length(inputmodels$step)) {
    print(anova(inputmodels$initial[[i]], inputmodels$step[[i]]))
  }
}
```

MSFT Anova

```
generateAnova(modelsMsft)
```

```
## Analysis of Variance Table
##
## Model 1: Target ~ Open + High + Low + Close + Adj.Close + Volume
## Model 2: Target ~ Close + Adj.Close
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     53 152.12
## 2     57 156.98 -4    -4.859 0.4232 0.7912
## Analysis of Variance Table
##
## Model 1: Target ~ Open + High + Low + Close + Adj.Close + Volume
## Model 2: Target ~ Low + Close + Adj.Close
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     53 69.883
## 2     56 71.784 -3   -1.9013 0.4807 0.6971
## Analysis of Variance Table
##
## Model 1: Target ~ Open + High + Low + Close + Adj.Close + Volume
## Model 2: Target ~ Low + Adj.Close
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     53 67.581
## 2     57 71.820 -4   -4.2391 0.8311 0.5114
## Analysis of Variance Table
##
## Model 1: Target ~ Open + High + Low + Close + Adj.Close + Volume
## Model 2: Target ~ Low
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     53 270.21
## 2     58 275.04 -5   -4.8291 0.1894 0.9653
## Analysis of Variance Table
##
## Model 1: Target ~ Open + High + Low + Close + Adj.Close + Volume
## Model 2: Target ~ Close + Adj.Close + Volume
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     53 108.51
## 2     56 110.99 -3   -2.4803 0.4038 0.7508
## Analysis of Variance Table
##
## Model 1: Target ~ Open + High + Low + Close + Adj.Close + Volume
## Model 2: Target ~ Adj.Close
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     53 150.39
## 2     58 162.51 -5   -12.127 0.8548 0.5175
```

```
## Analysis of Variance Table
##
## Model 1: Target ~ Open + High + Low + Close + Adj.Close + Volume
## Model 2: Target ~ Close
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     23 31.085
## 2     27 35.265 -4   -4.1807 0.7733 0.5537
```

AAPL Anova

```
generateAnova(modelsAapl)
```

```
## Analysis of Variance Table
##
## Model 1: Target ~ Open + High + Low + Close + Adj.Close + Volume
## Model 2: Target ~ Close + Adj.Close + Volume
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     53 380.58
## 2     56 383.50 -3   -2.9168 0.1354 0.9385
## Analysis of Variance Table
##
## Model 1: Target ~ Open + High + Low + Close + Adj.Close + Volume
## Model 2: Target ~ Low + Adj.Close
##   Res.Df   RSS Df Sum of Sq      F Pr(>F)
## 1     53 273.3
## 2     57 277.0 -4   -3.6963 0.1792 0.9482
## Analysis of Variance Table
##
## Model 1: Target ~ Open + High + Low + Close + Adj.Close + Volume
## Model 2: Target ~ High + Low + Close + Adj.Close + Volume
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     53 253.37
## 2     54 260.82 -1   -7.4486 1.5581 0.2174
## Analysis of Variance Table
##
## Model 1: Target ~ Open + High + Low + Close + Adj.Close + Volume
## Model 2: Target ~ Low + Close + Adj.Close
##   Res.Df     RSS Df Sum of Sq      F Pr(>F)
## 1     53  989.83
## 2     56 1001.02 -3    -11.19 0.1997 0.8961
## Analysis of Variance Table
##
## Model 1: Target ~ Open + High + Low + Close + Adj.Close + Volume
## Model 2: Target ~ Open + Low + Adj.Close
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     53 688.82
## 2     56 721.20 -3   -32.382 0.8305  0.483
## Analysis of Variance Table
##
## Model 1: Target ~ Open + High + Low + Close + Adj.Close + Volume
## Model 2: Target ~ Close + Adj.Close
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     53 555.23
## 2     57 560.04 -4   -4.8137 0.1149 0.9767
```

```
## Analysis of Variance Table
##
## Model 1: Target ~ Open + High + Low + Close + Adj.Close + Volume
## Model 2: Target ~ Close
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     23  99.81
## 2     27 108.47 -4   -8.6585 0.4988 0.7368
```

Looking at the anova test, Target ~ High + Low + Close + Adj.Close + Volume is a good model since it has a pvalue less than 0.25.

TSLA Anova

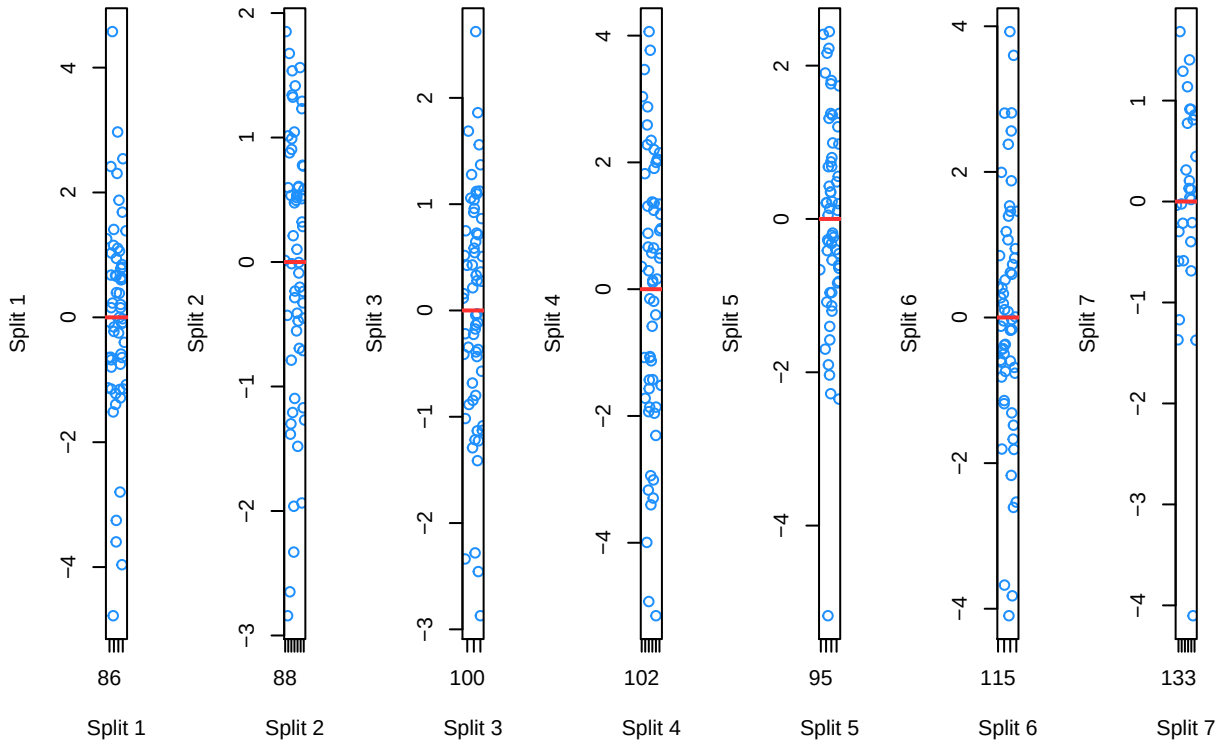
```
generateAnova(modelsTsla)
```

```
## Analysis of Variance Table
##
## Model 1: Target ~ Open + High + Low + Close + Adj.Close + Volume
## Model 2: Target ~ High + Low + Close
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     54 4056.5
## 2     56 4088.0 -2   -31.478 0.2095 0.8116
## Analysis of Variance Table
##
## Model 1: Target ~ Open + High + Low + Close + Adj.Close + Volume
## Model 2: Target ~ Close + Volume
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     54 4849.5
## 2     57 5002.3 -3   -152.87 0.5674 0.6389
## Analysis of Variance Table
##
## Model 1: Target ~ Open + High + Low + Close + Adj.Close + Volume
## Model 2: Target ~ Close
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     54 7966.0
## 2     58 8301.6 -4   -335.65 0.5688 0.6863
## Analysis of Variance Table
##
## Model 1: Target ~ Open + High + Low + Close + Adj.Close + Volume
## Model 2: Target ~ Close + Volume
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     54 8679.5
## 2     57 8855.9 -3   -176.41 0.3659 0.7779
## Analysis of Variance Table
##
## Model 1: Target ~ Open + High + Low + Close + Adj.Close + Volume
## Model 2: Target ~ High + Low + Close + Volume
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     54 5809.1
## 2     55 5819.6 -1   -10.494 0.0976  0.756
## Analysis of Variance Table
##
## Model 1: Target ~ Open + High + Low + Close + Adj.Close + Volume
## Model 2: Target ~ Open + High + Close + Volume
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
```

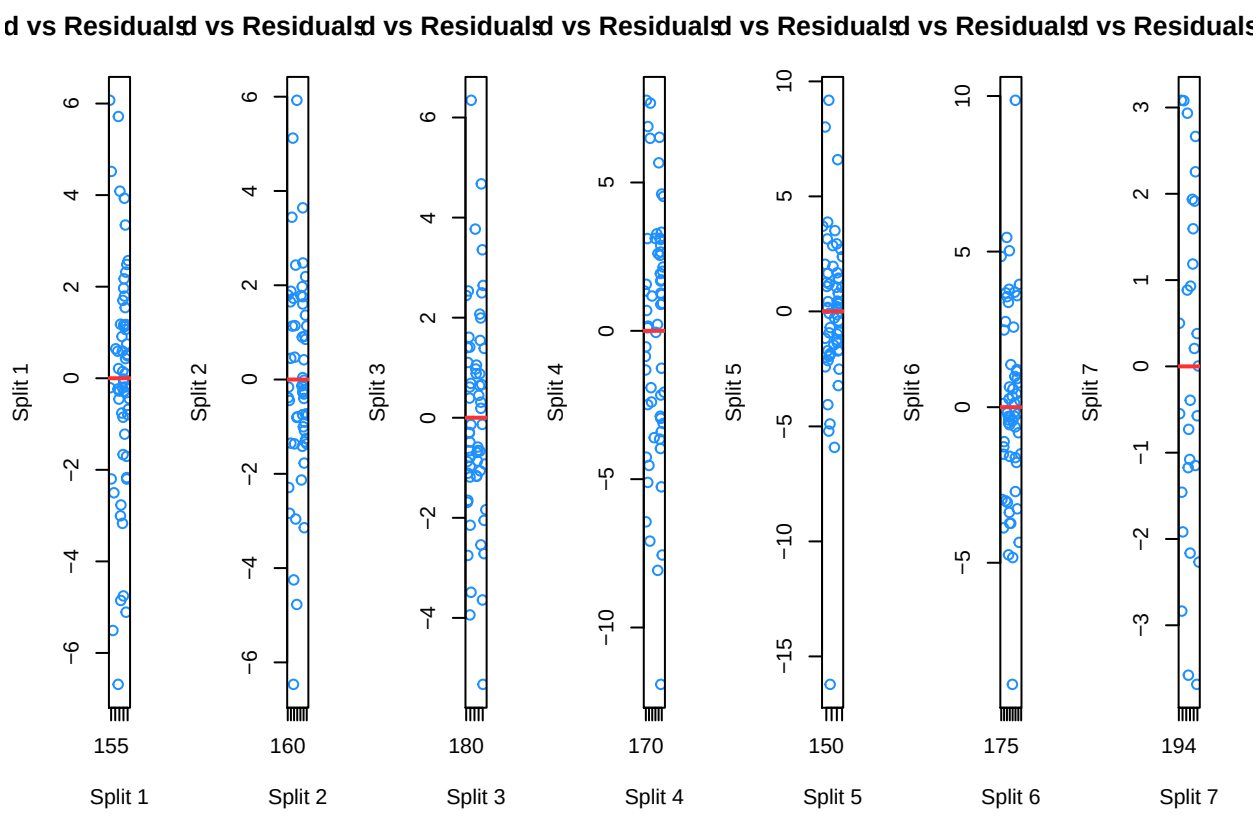
```
## 1     54 2931.2
## 2     55 2958.0 -1   -26.738 0.4926 0.4858
## Analysis of Variance Table
##
## Model 1: Target ~ Open + High + Low + Close + Adj.Close + Volume
## Model 2: Target ~ Close
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     23 551.15
## 2     27 579.12 -4   -27.973 0.2918 0.8802
```

```
generateResiduals <- function(madeModels, name) {
  par(mfrow=c(1,length(madeModels$step)))
  for (i in 1:length(madeModels$step)) {
    plot(fitted(madeModels$step[[i]]), resid(madeModels$step[[i]]), main = paste("Fitted vs Residuals",
    abline(h = 0, col = "firebrick1", lwd = 2)
  }
}

generateResiduals(modelsMsft, "MSFT")
```

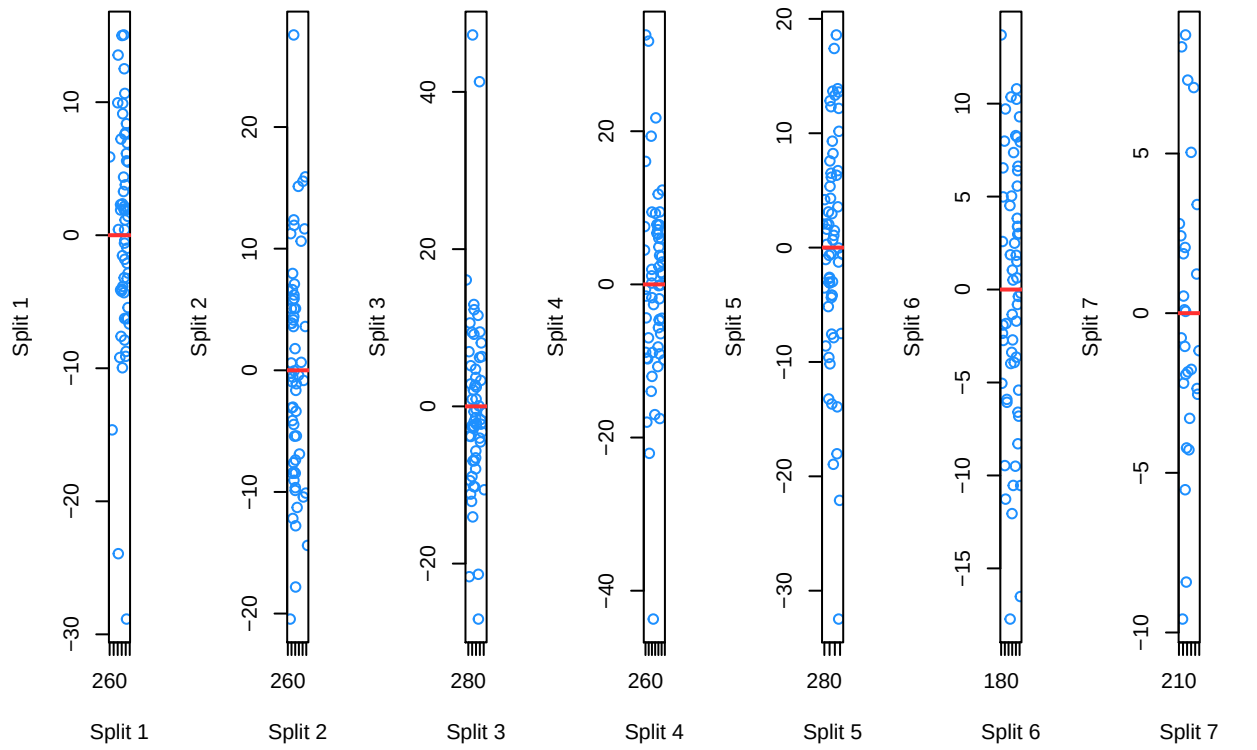
d vs Residualsd vs Residualsd vs Residualsd vs Residualsd vs Residualsd vs Residualsd vs Residuals



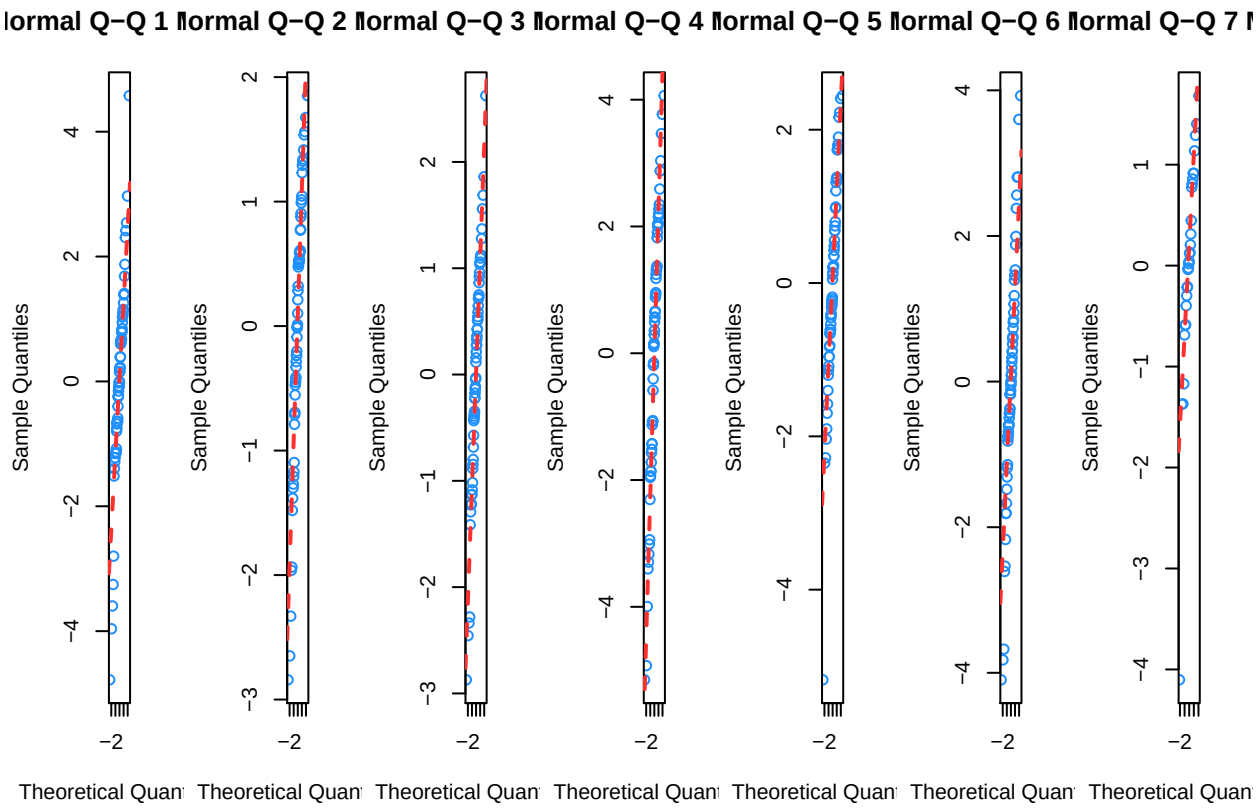

```
generateResiduals(modelsAapl, "AAPL")
```



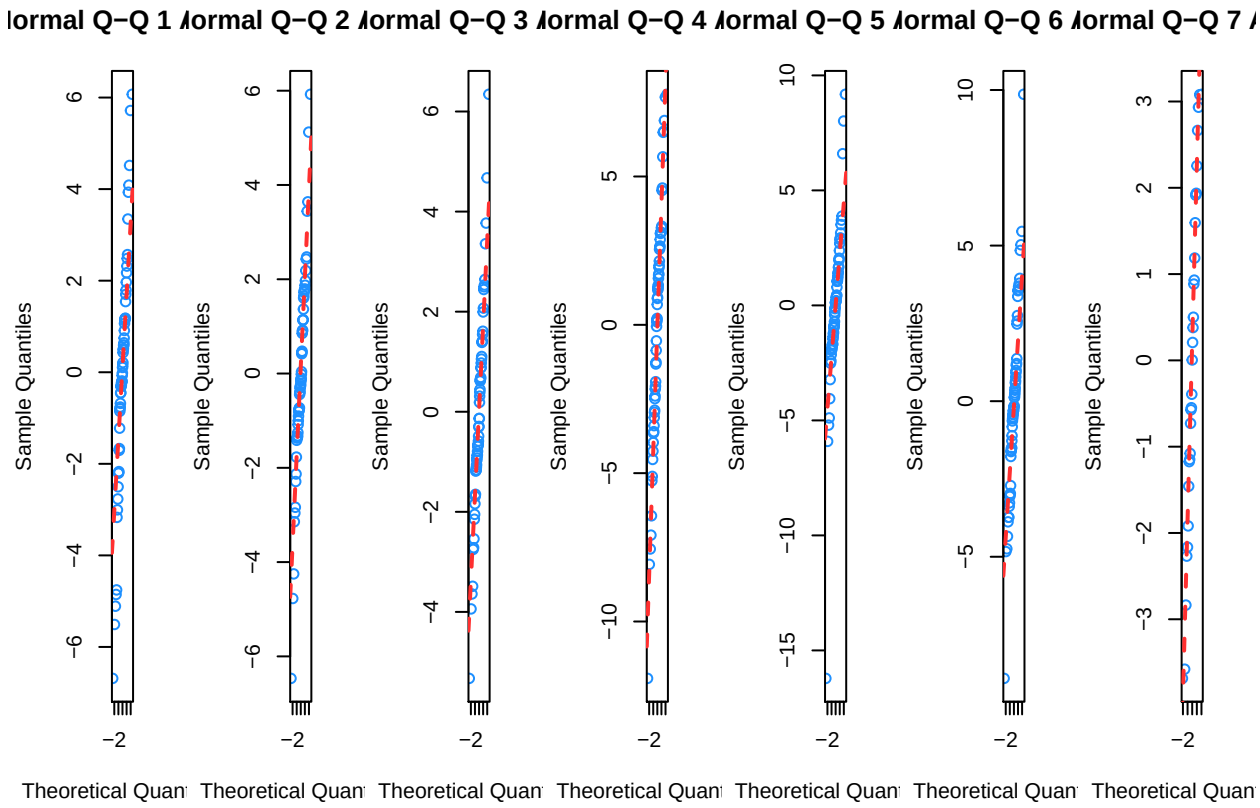

```
generateResiduals(modelsTsla, "TSLA")
```



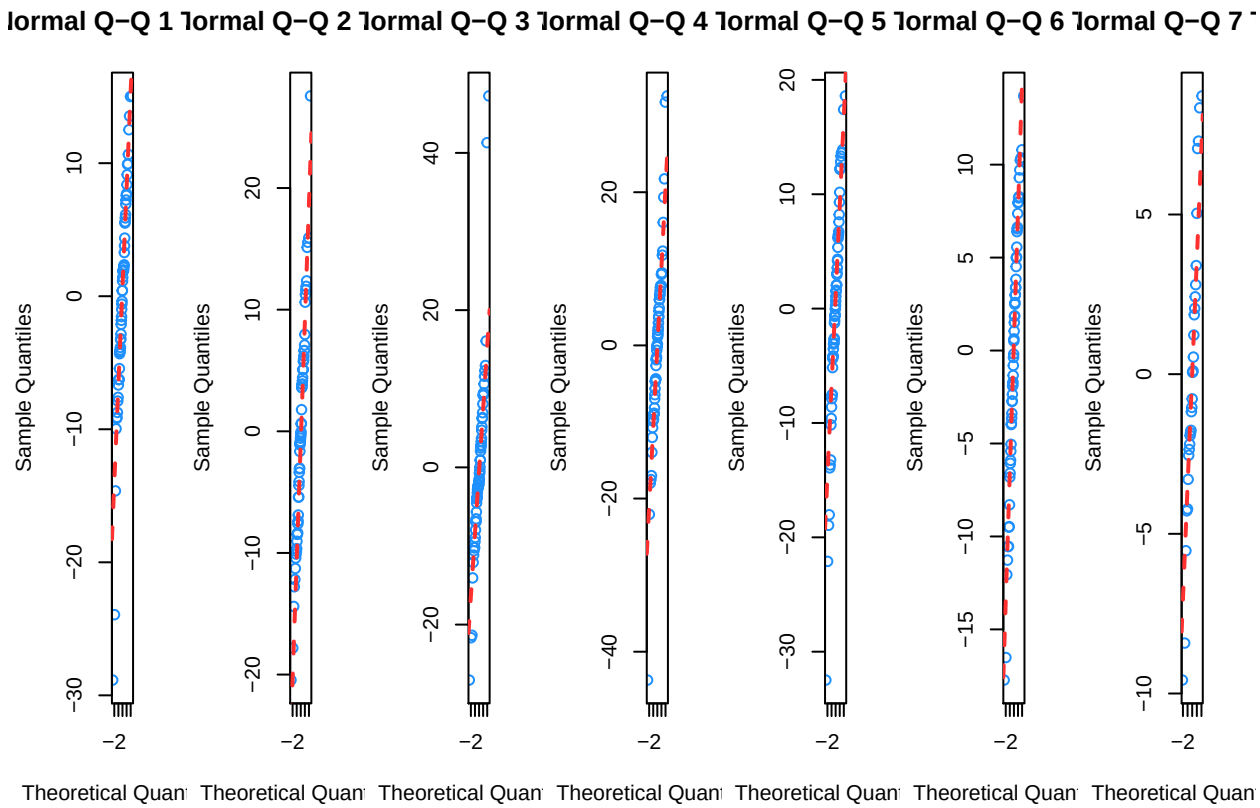
```
generateQQ <- function(madeModels, name) {
  par(mfrow=c(1,length(madeModels$step)))
  for (i in 1:length(madeModels$step)) {
    qqnorm(resid(madeModels$step[[i]]), col = "dodgerblue1", main = paste("Normal Q-Q", i, name))
    qqline(resid(madeModels$step[[i]]), lty = 2, lwd = 2, col = "firebrick1")
  }
}

generateQQ(modelsMsft, "MSFT")
```




```
generateQQ(modelsAapl, "AAPL")
```




```
generateQQ(modelsTsla, "TLSA")
```



```r
doShapiro <- function(madeModels, name) {
  print(paste(name, "Shapiro tests"))
  for (i in 1:length(madeModels$step)) {
    print(paste("Split", i))
    print(shapiro.test(resid(madeModels$step[[i]])))
  }
}
```

```r
doShapiro(modelsMsft, "MSFT")
```

```
## [1] "MSFT Shapiro tests"
## [1] "Split 1"
##
##  Shapiro-Wilk normality test
##
## data:  resid(madeModels$step[[i]])
## W = 0.95957, p-value = 0.04481
##
## [1] "Split 2"
##
##  Shapiro-Wilk normality test
##
## data:  resid(madeModels$step[[i]])
## W = 0.96214, p-value = 0.0599
##
## [1] "Split 3"
##
##  Shapiro-Wilk normality test
##
## data:  resid(madeModels$step[[i]])
## W = 0.98164, p-value = 0.5019
##
## [1] "Split 4"
##
##  Shapiro-Wilk normality test
##
## data:  resid(madeModels$step[[i]])
## W = 0.97843, p-value = 0.3658
##
## [1] "Split 5"
##
##  Shapiro-Wilk normality test
##
## data:  resid(madeModels$step[[i]])
## W = 0.95737, p-value = 0.03506
##
## [1] "Split 6"
##
##  Shapiro-Wilk normality test
##
## data:  resid(madeModels$step[[i]])
## W = 0.98168, p-value = 0.5037
##
## [1] "Split 7"
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(madeModels$step[[i]])
## W = 0.87511, p-value = 0.002614
```

```
doShapiro(modelsAapl, "AAPL")
```

```
## [1] "AAPL Shapiro tests"
## [1] "Split 1"
##
##  Shapiro-Wilk normality test
##
## data:  resid(madeModels$step[[i]])
## W = 0.97217, p-value = 0.1862
##
## [1] "Split 2"
##
##  Shapiro-Wilk normality test
##
## data:  resid(madeModels$step[[i]])
## W = 0.97436, p-value = 0.2372
##
## [1] "Split 3"
##
##  Shapiro-Wilk normality test
##
## data:  resid(madeModels$step[[i]])
## W = 0.98601, p-value = 0.7227
##
## [1] "Split 4"
##
##  Shapiro-Wilk normality test
##
## data:  resid(madeModels$step[[i]])
## W = 0.98062, p-value = 0.4551
##
## [1] "Split 5"
##
##  Shapiro-Wilk normality test
##
## data:  resid(madeModels$step[[i]])
## W = 0.87145, p-value = 1.392e-05
##
## [1] "Split 6"
##
##  Shapiro-Wilk normality test
##
## data:  resid(madeModels$step[[i]])
## W = 0.97214, p-value = 0.1855
##
## [1] "Split 7"
##
##  Shapiro-Wilk normality test
##
```

```
## data:  resid(madeModels$step[[i]])
## W = 0.96808, p-value = 0.5089
```

```
doShapiro(modelsTsla, "TSLA")
```

```
## [1] "TSLA Shapiro tests"
## [1] "Split 1"
##
##  Shapiro-Wilk normality test
##
## data:  resid(madeModels$step[[i]])
## W = 0.95004, p-value = 0.01565
##
## [1] "Split 2"
##
##  Shapiro-Wilk normality test
##
## data:  resid(madeModels$step[[i]])
## W = 0.9872, p-value = 0.7828
##
## [1] "Split 3"
##
##  Shapiro-Wilk normality test
##
## data:  resid(madeModels$step[[i]])
## W = 0.87448, p-value = 1.744e-05
##
## [1] "Split 4"
##
##  Shapiro-Wilk normality test
##
## data:  resid(madeModels$step[[i]])
## W = 0.95499, p-value = 0.02689
##
## [1] "Split 5"
##
##  Shapiro-Wilk normality test
##
## data:  resid(madeModels$step[[i]])
## W = 0.96854, p-value = 0.1237
##
## [1] "Split 6"
##
##  Shapiro-Wilk normality test
##
## data:  resid(madeModels$step[[i]])
## W = 0.98165, p-value = 0.5021
##
## [1] "Split 7"
##
##  Shapiro-Wilk normality test
##
## data:  resid(madeModels$step[[i]])
## W = 0.97189, p-value = 0.6119
```

RESULTS

The anova tests for MSFT all have pvalue greater than 0.05. If we made the rejection level alpha = 0.05, we'd end up not rejecting the null hypothesis. Instead, we choose to pick 2 models with the lowest pvalues. For instance with MSFT, we should choose Target ~ Low + Adj.Close and Target ~ Adj.Close because they have the lowest pvalues (0.5114 and 0.5175 respectively). We do this because our data has high noise so getting exact anova value is difficult. Same as before, for AAPL, we choose 2 models with the lowest pvalue which are Target ~ High + Low + Close + Adj.Close + Volume and Target ~ Open + Low + Adj.Close (pvalues 0.2174 and 0.483 respectively). Same as before, for TSLA, we choose 2 models with the lowest pvalue which are Target ~ Open + High + Close + Volume and Target ~ Close + Volume (pvalues 0.4858 and 0.6389 respectively).

In addition, looking at the fitted vs residuals for MSFT, AAPL and TSLA, none of the diagnostic plots seem to violate normality. This means that during that time, the market was behaving in a linear way. In the future, we could use this information to decide whether or not our model should be used to trading. Looking at the Normal Q-Q for MSFT, AAPL and TSLA, Normal Q-Q 1 MSFT, Normal Q-Q 7 AAPL and Normal Q-Q 7 TSLA seem to violate normality. This means that during that period, the data points were not normally distributed. However, it's really hard to judge constant variance and normality by looking at these diagnostic plots. Therefore, we performed tests that could give us the pvalue and set an appropriate significant level for rejection.

Looking at the Shapiro.Wilko test:

1. For MSFT at alpha level 0.05, 1st, 5th and 7th model does not violate normality.
2. For AAPL, the pvalues are larger than the previous one, so we'll use a larger signifigance level. If we use alpla = 0.2, the 1st, 5th and 6th models do not violate normality.
3. For TSLA, if we use signifigance level 0.05, the 1st, 3rd and 4th model do not violate normality.

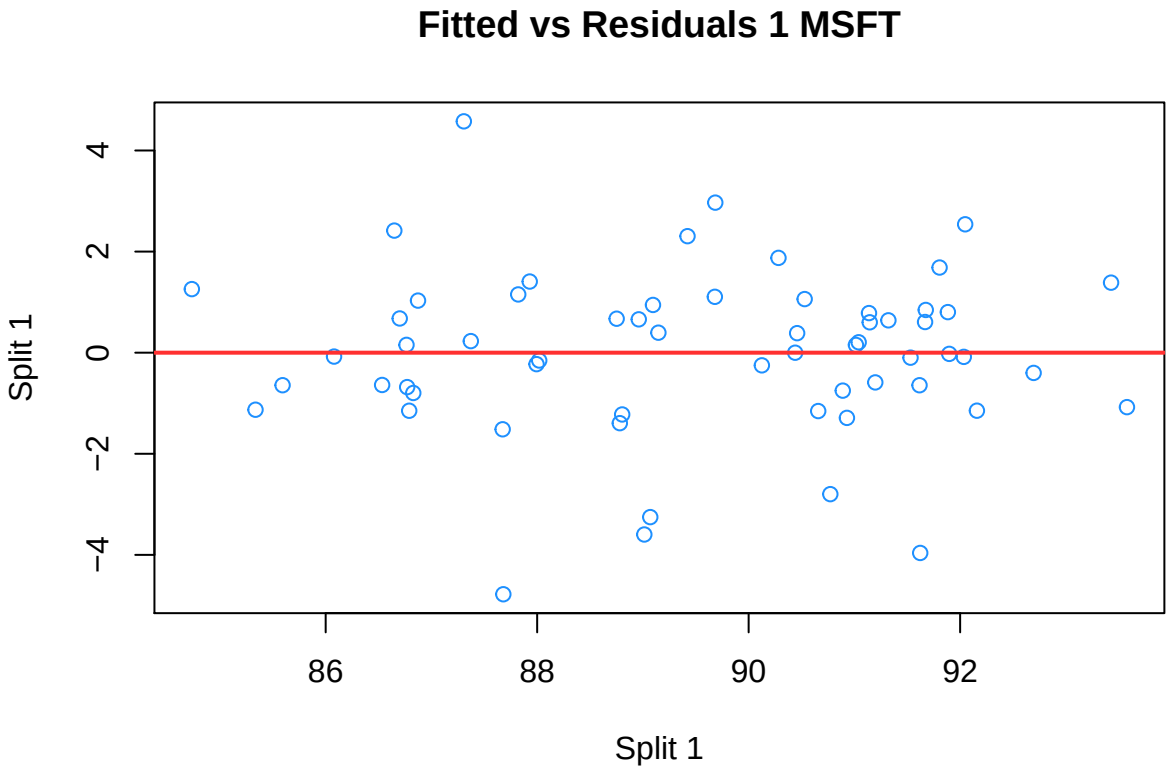
DISCUSSION

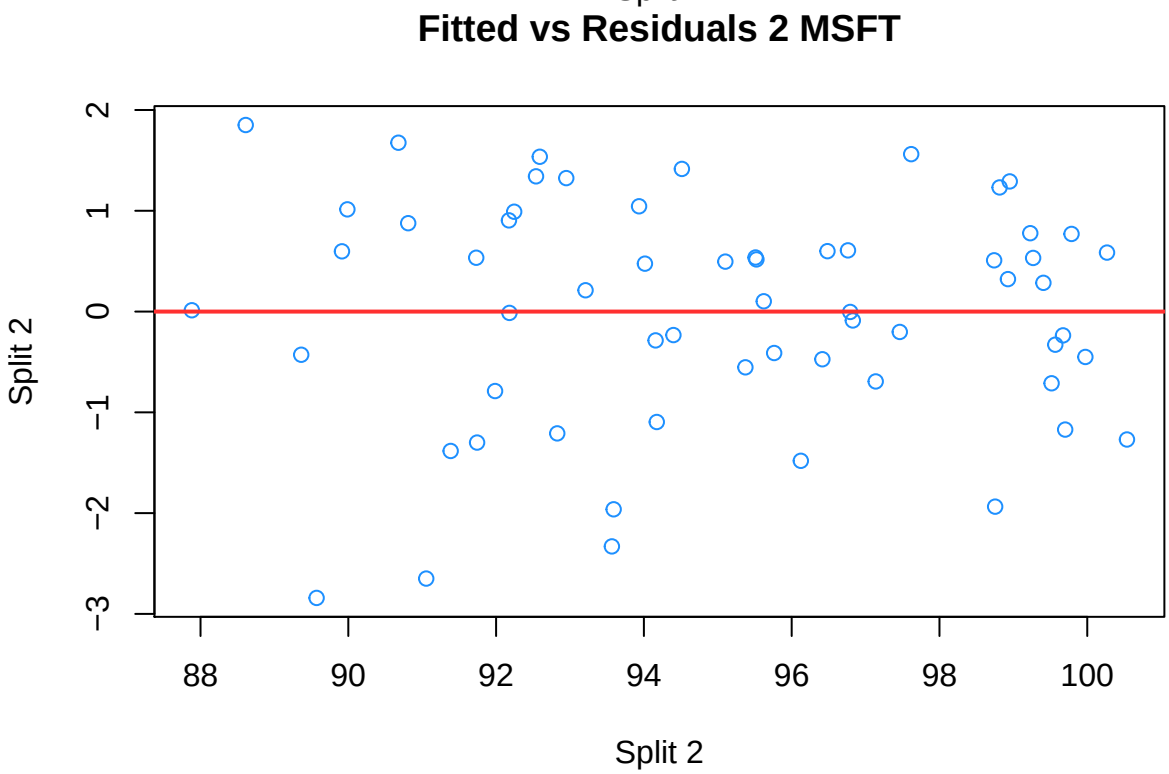
The models generated by model selection are useful because the models give a general indication on whether a stock will go up or down. Using thr data, a trader would be able to make an informed decision on whether or not to buy a stock based on what it will do in the future. In the future, we can improve the models by testing it on larger datasets that include economic downturns as well, specifically the 2008 recession and 1999 dot com bubble. Testing on datasets like that would show us that the model can operate in periods of high volatility and uncertainty. If our model shows an upward trend, you buy because the methodology is buy low sell high. And if our model shows a downward trend, you sell because it's going to lose value.

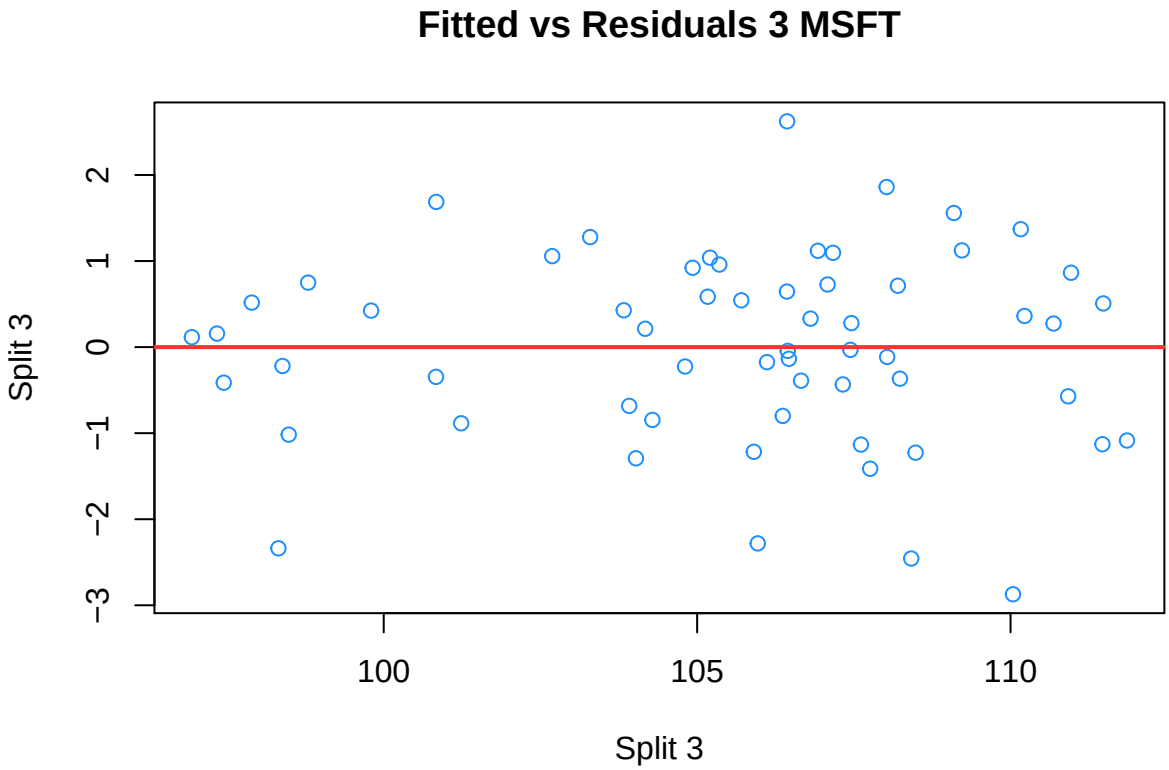
APPENDIX

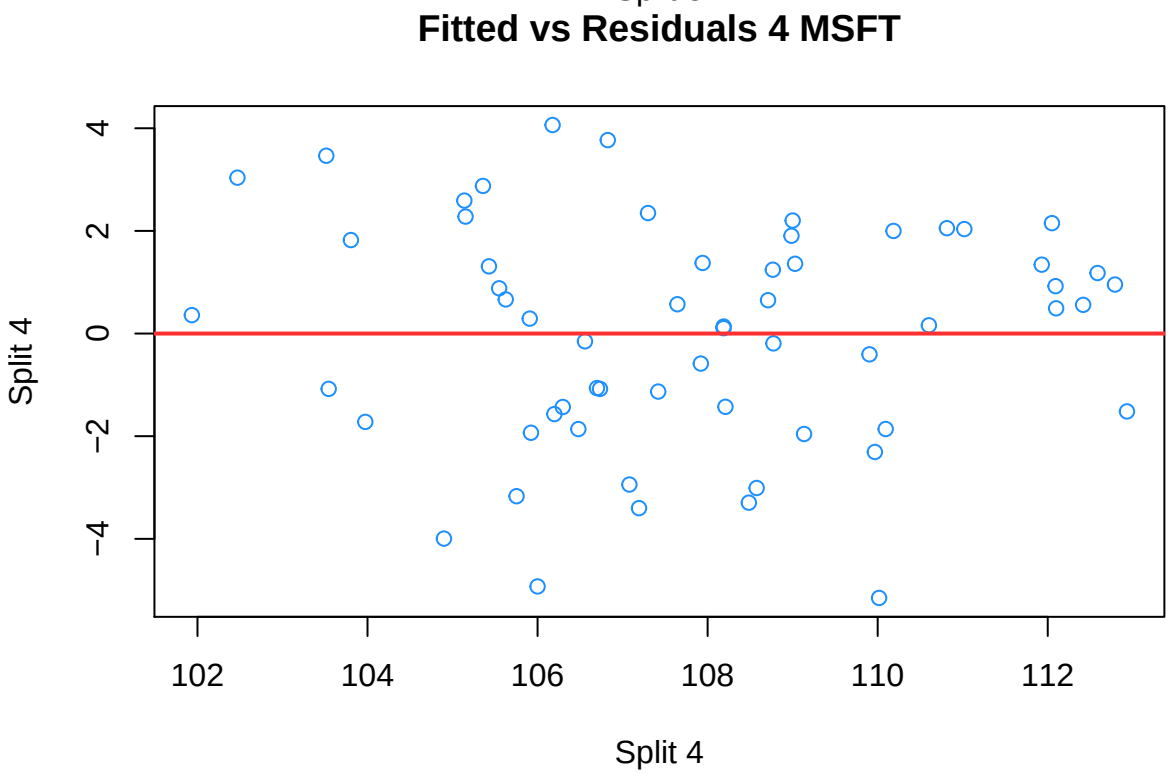
The following are full size charts as presented in the report for easier viewing.

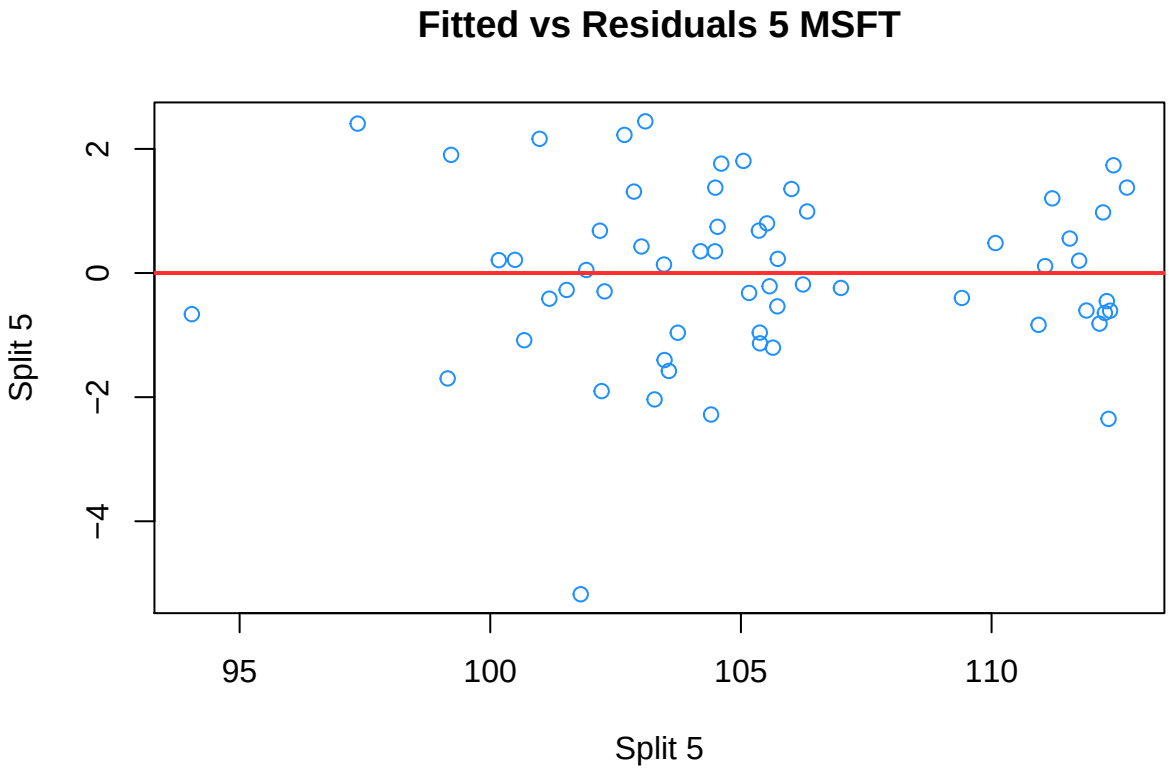
```
generateResiduals <- function(madeModels, name) {
  for (i in 1:length(madeModels$step)) {
    plot(fitted(madeModels$step[[i]]), resid(madeModels$step[[i]]), main = paste("Fitted vs Residuals",
    abline(h = 0, col = "firebrick1", lwd = 2)
  }
}

generateResiduals(modelsMsft, "MSFT")
```

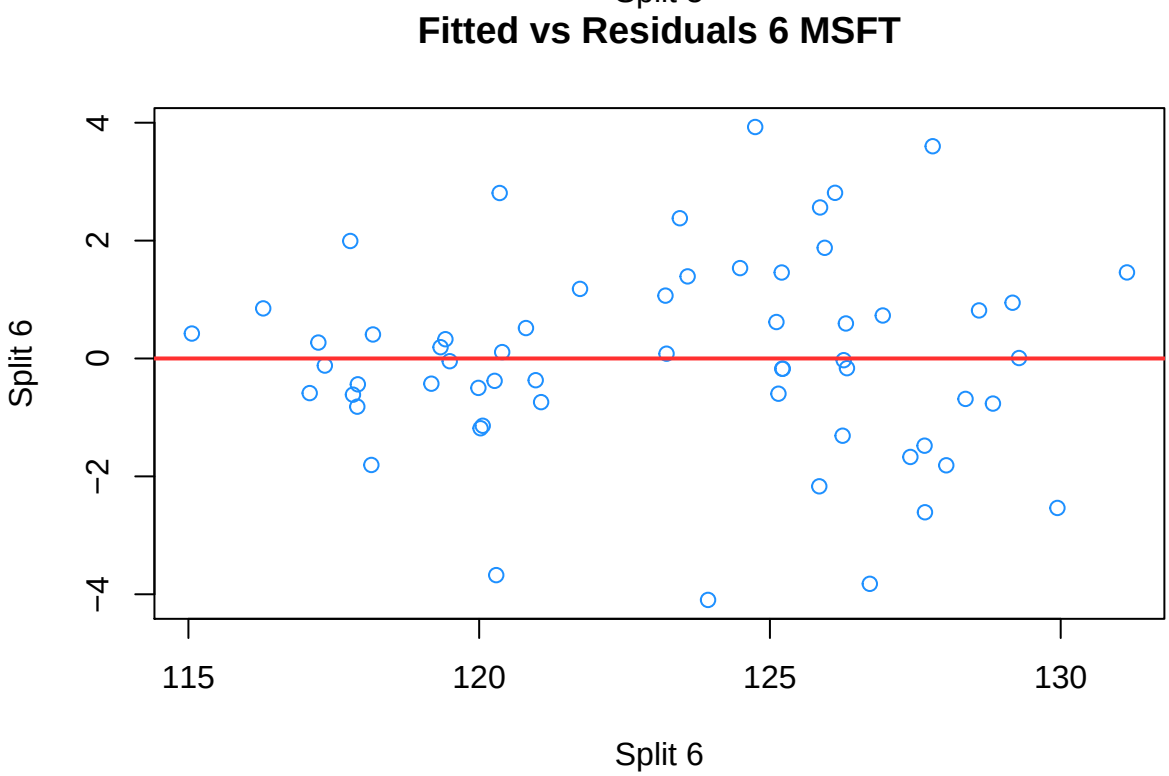

Fitted vs Residuals 1 MSFT
Split 1
4
2
0
−2
−4
86
88
90
92
Split 1

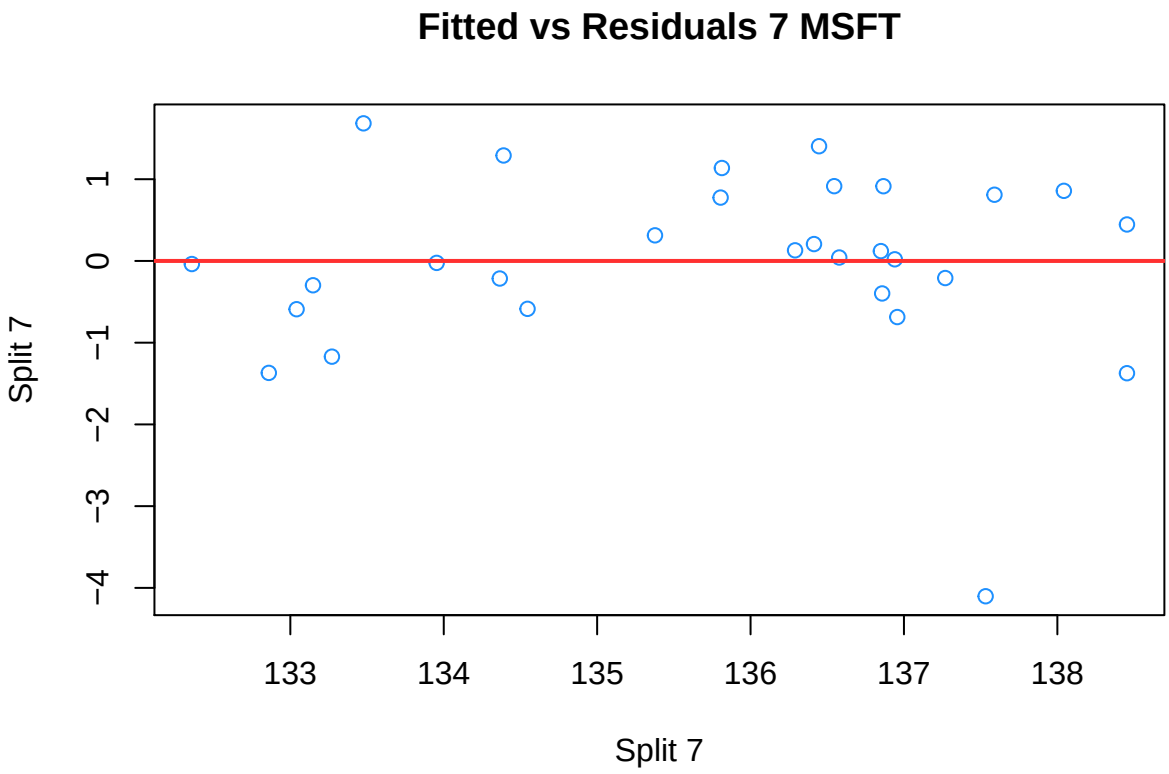

Fitted vs Residuals 2 MSFT
Split 2
2
1
0
−1
−2
−3
88
90
92
94
96
98
100
Split 2


Fitted vs Residuals 3 MSFT
Split 3
Split 3
100
105
110
−3
−2
−1
0
1
2

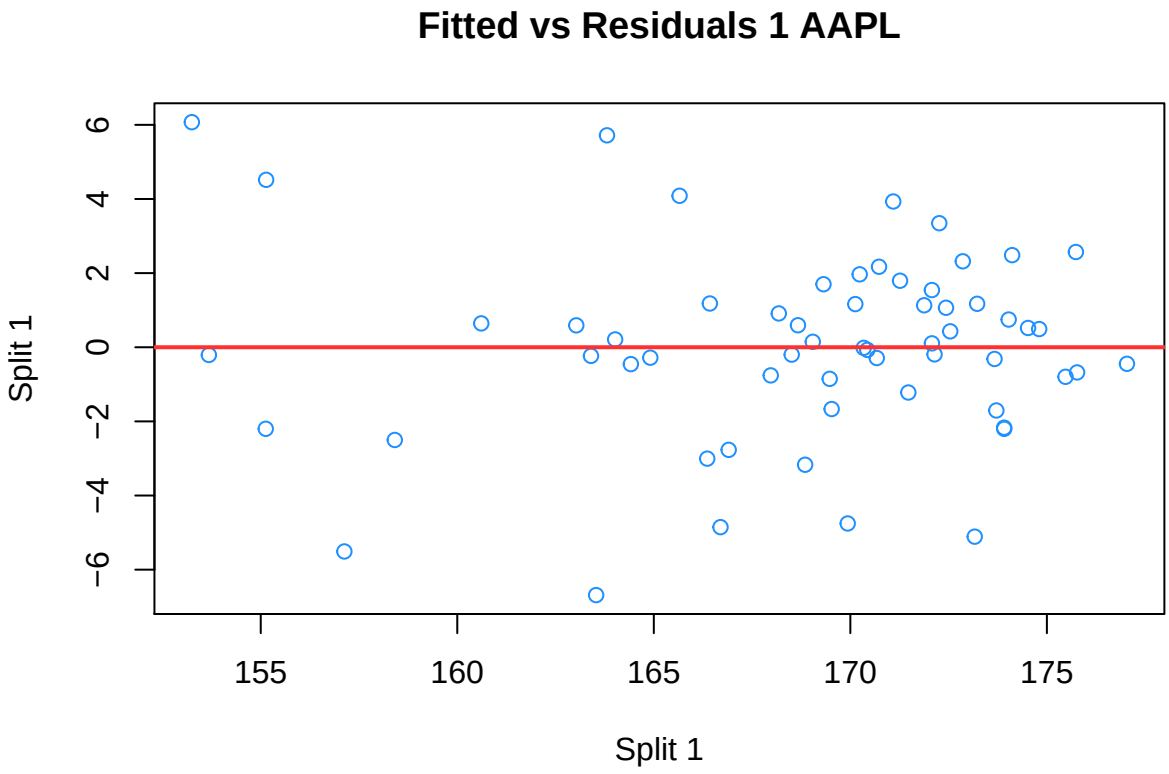

Fitted vs Residuals 4 MSFT
Split 4
Split 4
102
104
106
108
110
112
−4
−2
0
2
4

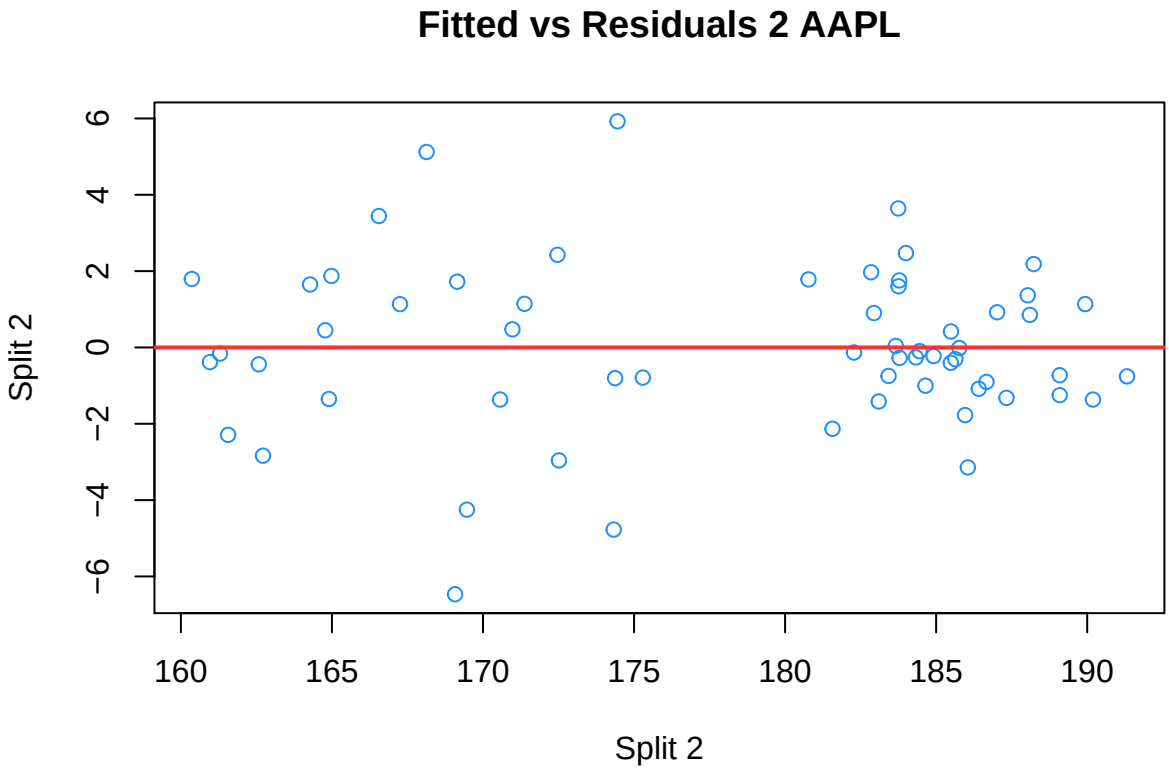

Fitted vs Residuals 5 MSFT
Split 5
2
0
−2
−4
95
100
105
110
Split 5

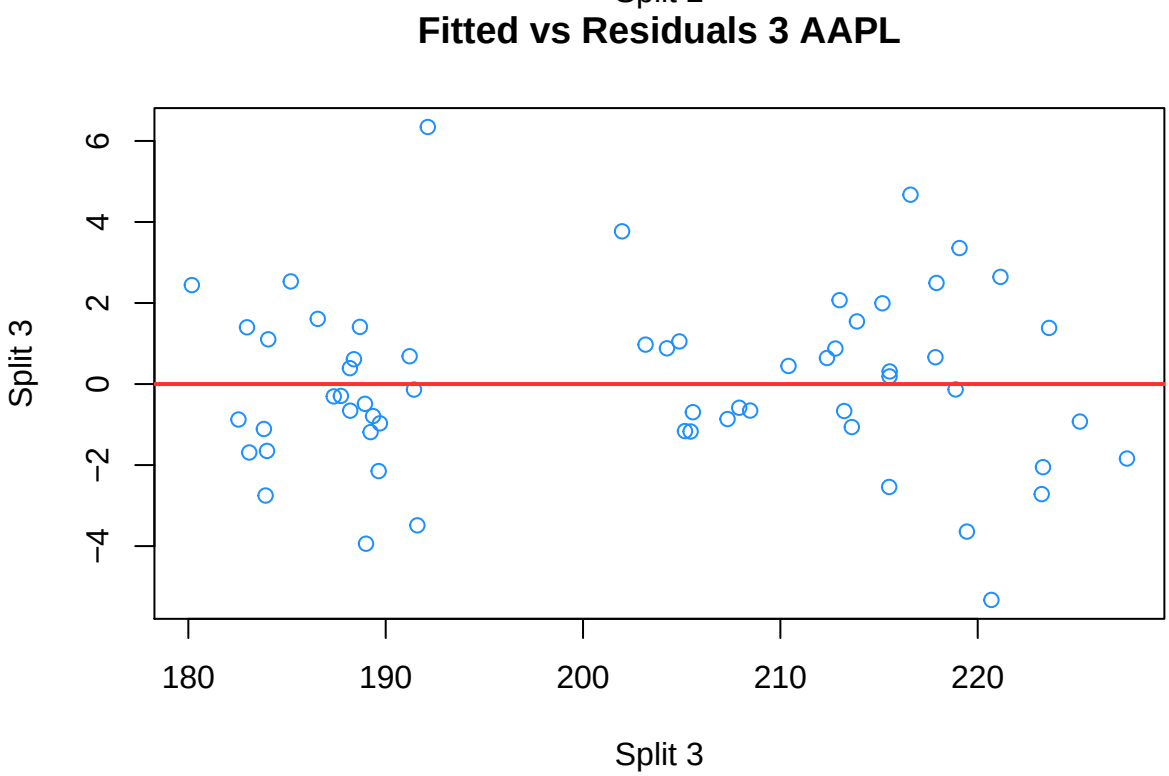

Fitted vs Residuals 6 MSFT
Split 6
4
2
0
−2
−4
115
120
125
130
Split 6

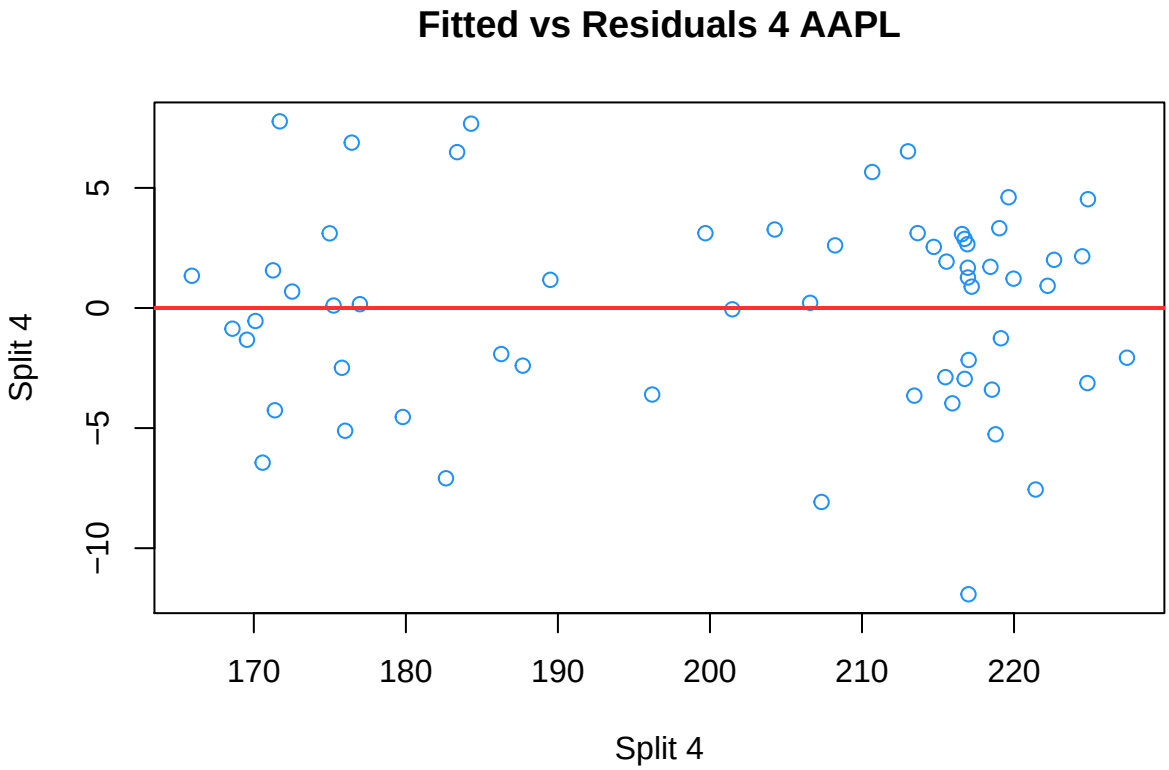


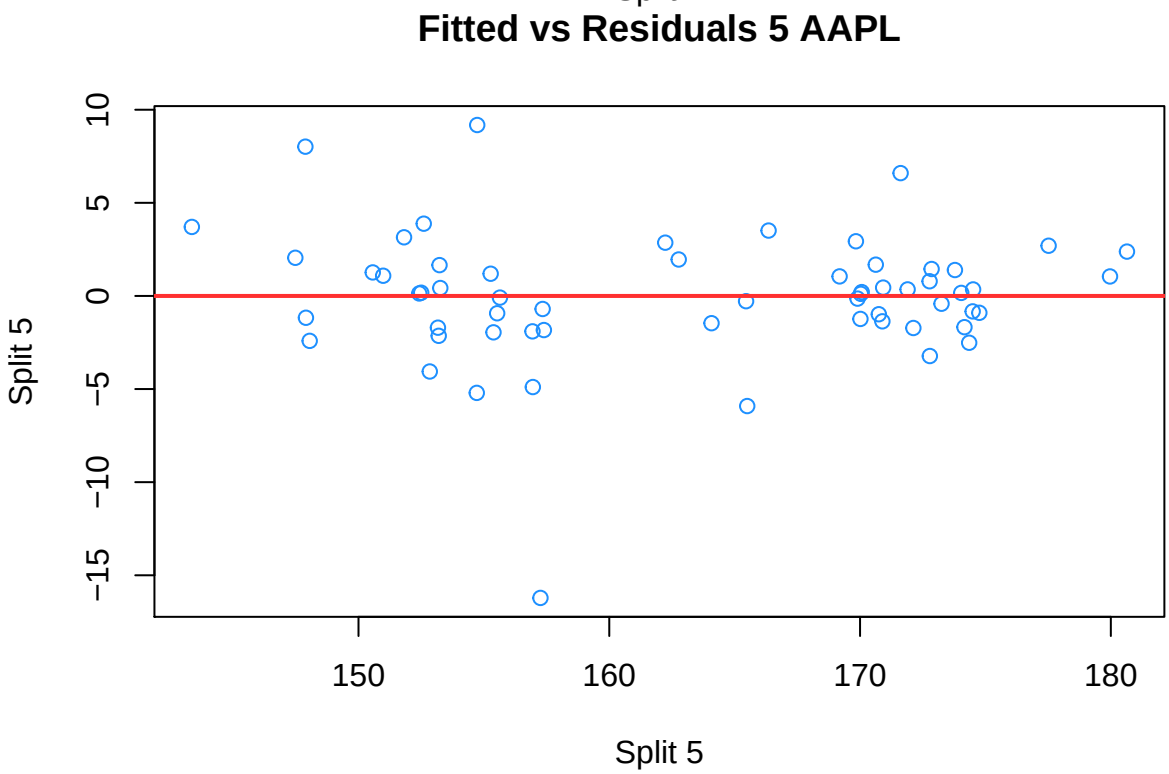

```
generateResiduals(modelsAapl, "AAPL")
```

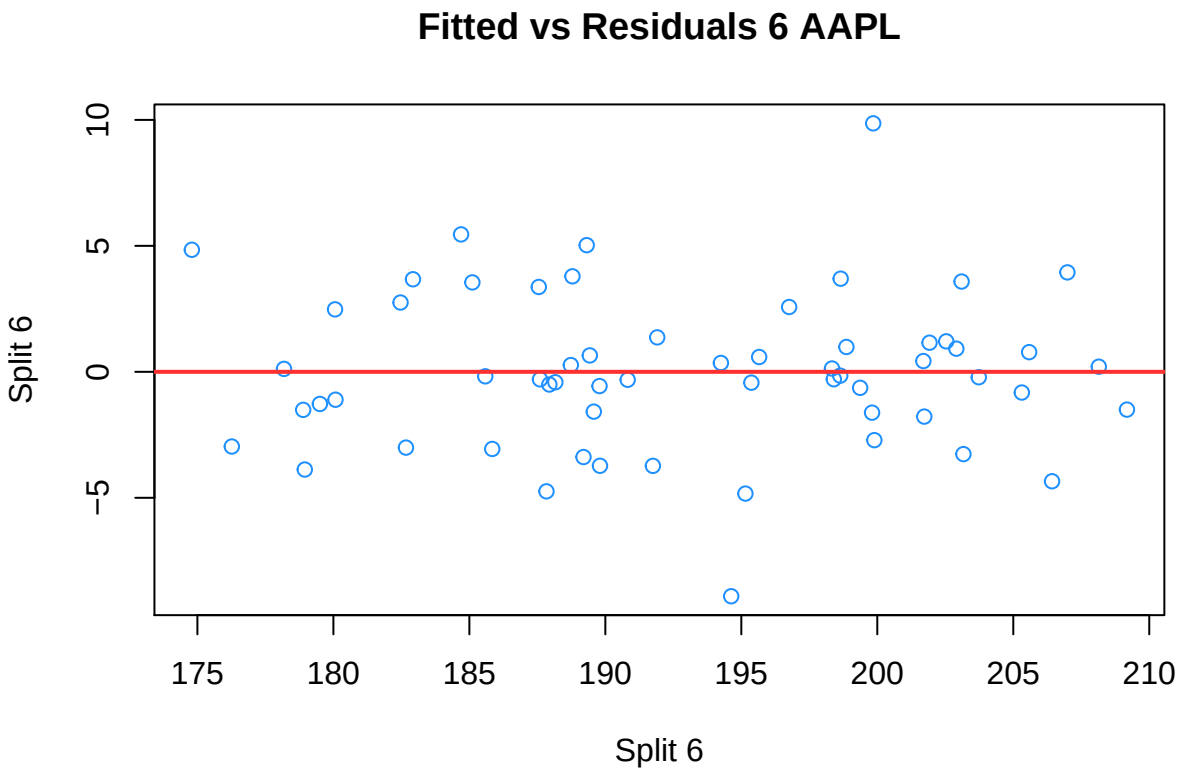


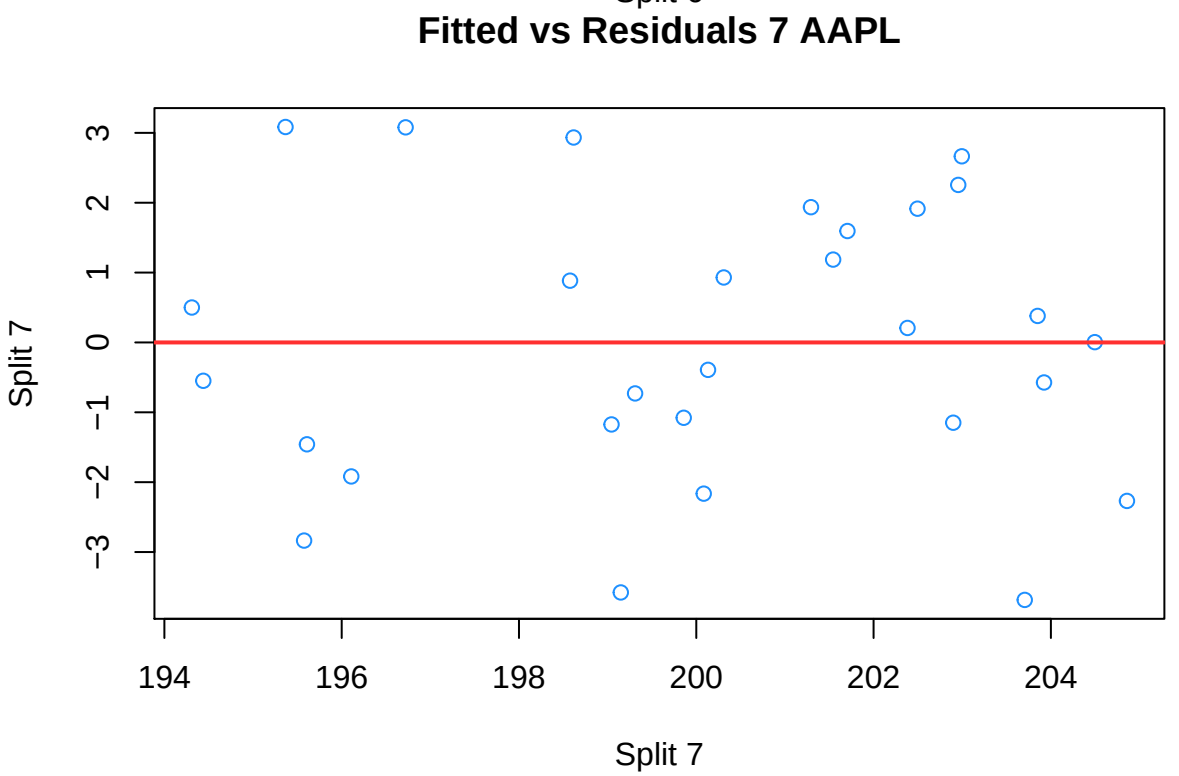

Fitted vs Residuals 2 AAPL
Split 2
6
4
2
0
−2
−4
−6
160
165
170
175
180
185
190
Split 2


Fitted vs Residuals 3 AAPL
Split 3
6
4
2
0
−2
−4
180
190
200
210
220
Split 3

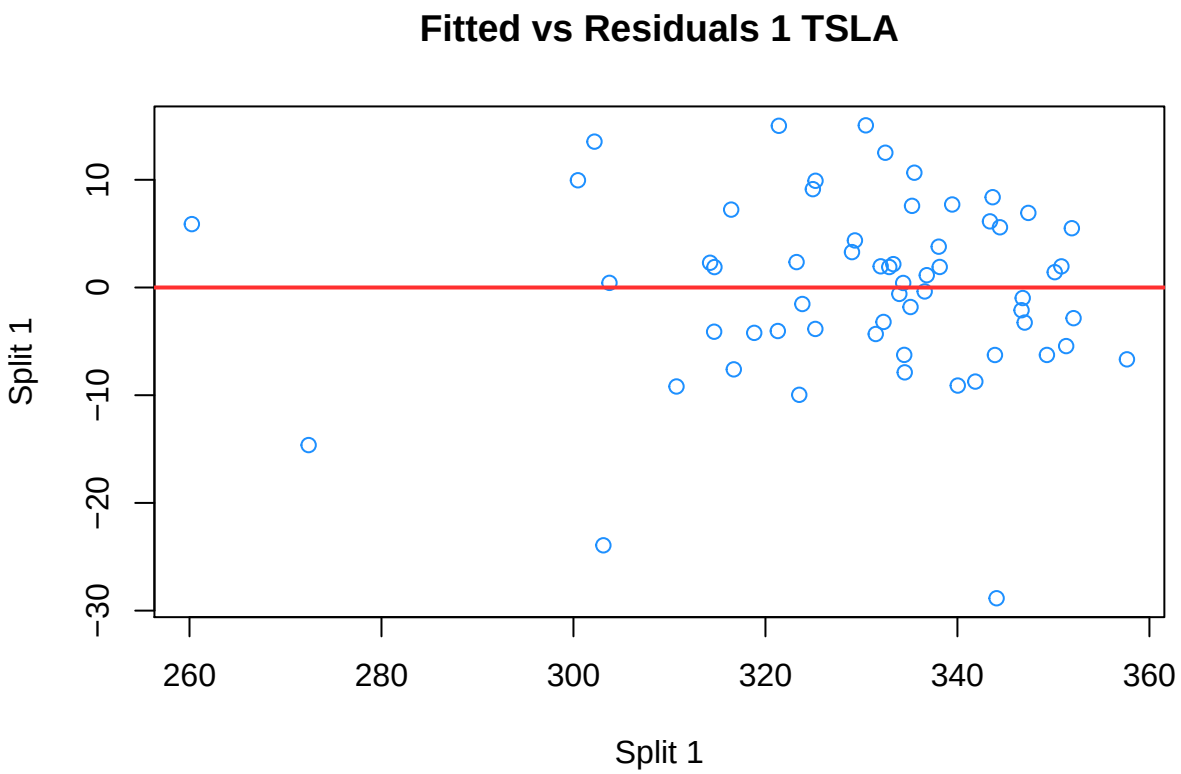

Fitted vs Residuals 4 AAPL
Split 4
Split 4
5
0
−5
−10
170
180
190
200
210
220

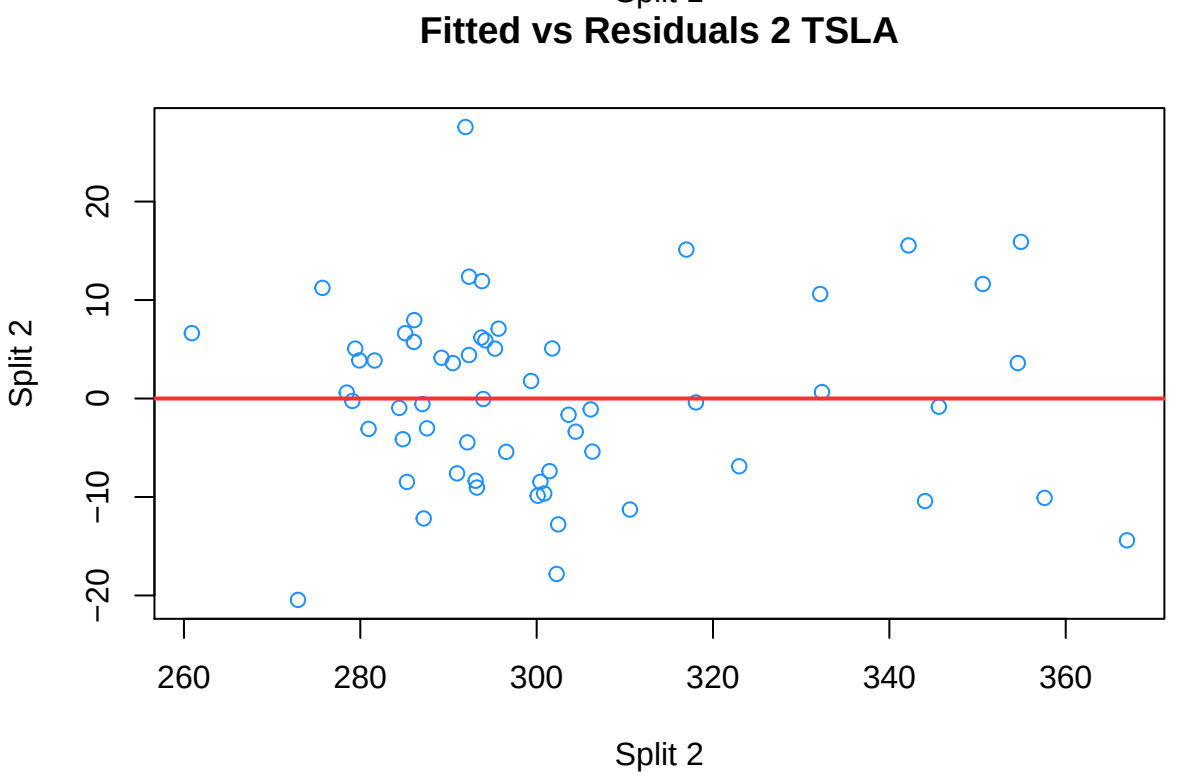

Fitted vs Residuals 5 AAPL
Split 5
Split 5
10
5
0
−5
−10
−15
150
160
170
180

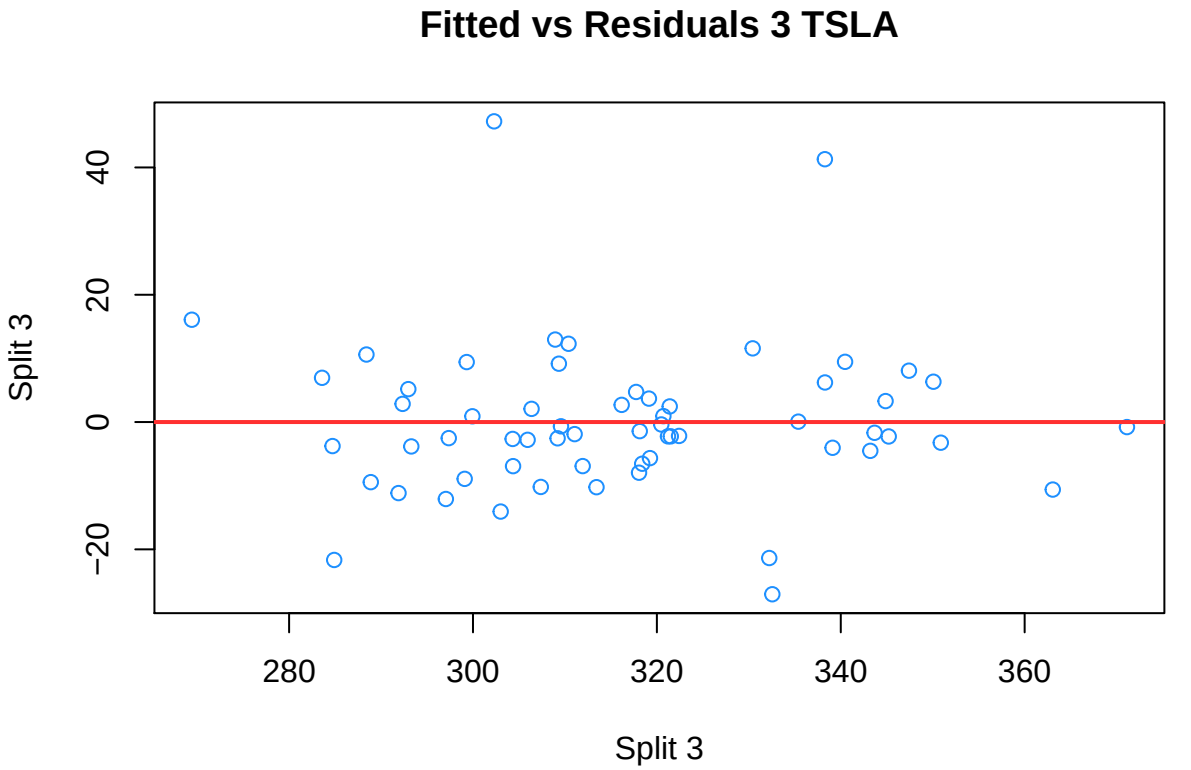


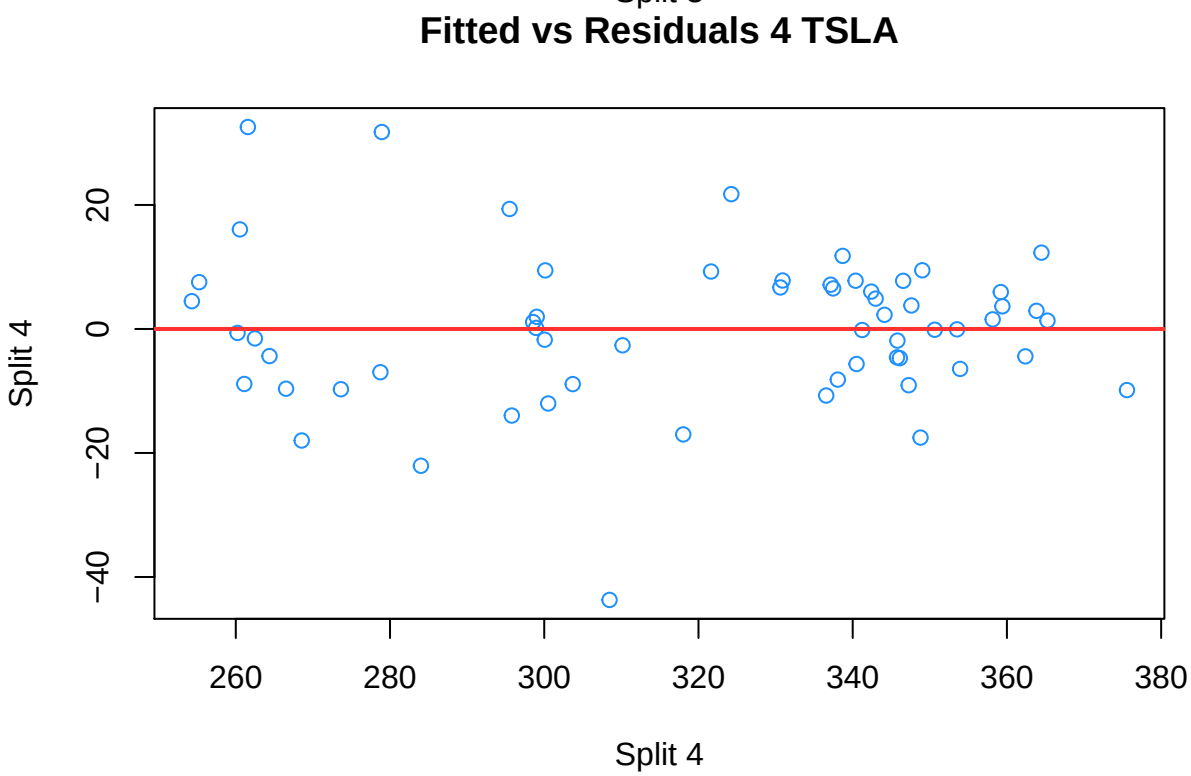


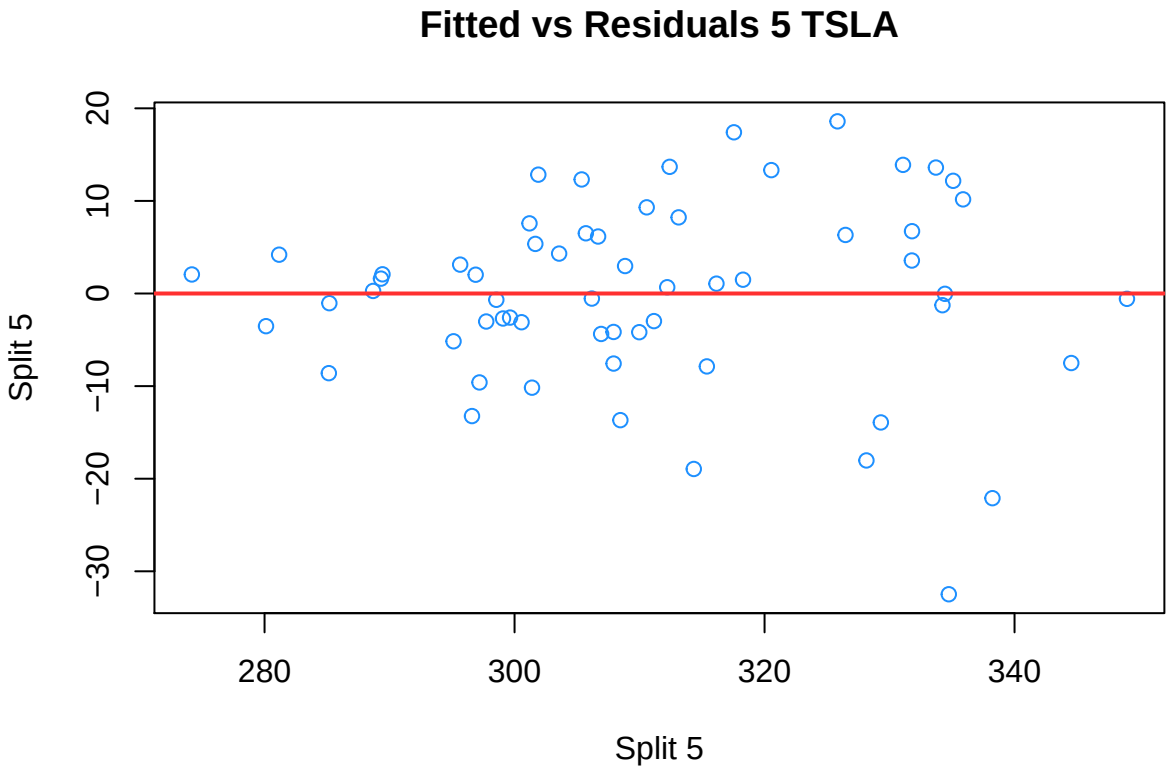

```
generateResiduals(modelsTsla, "TSLA")
```

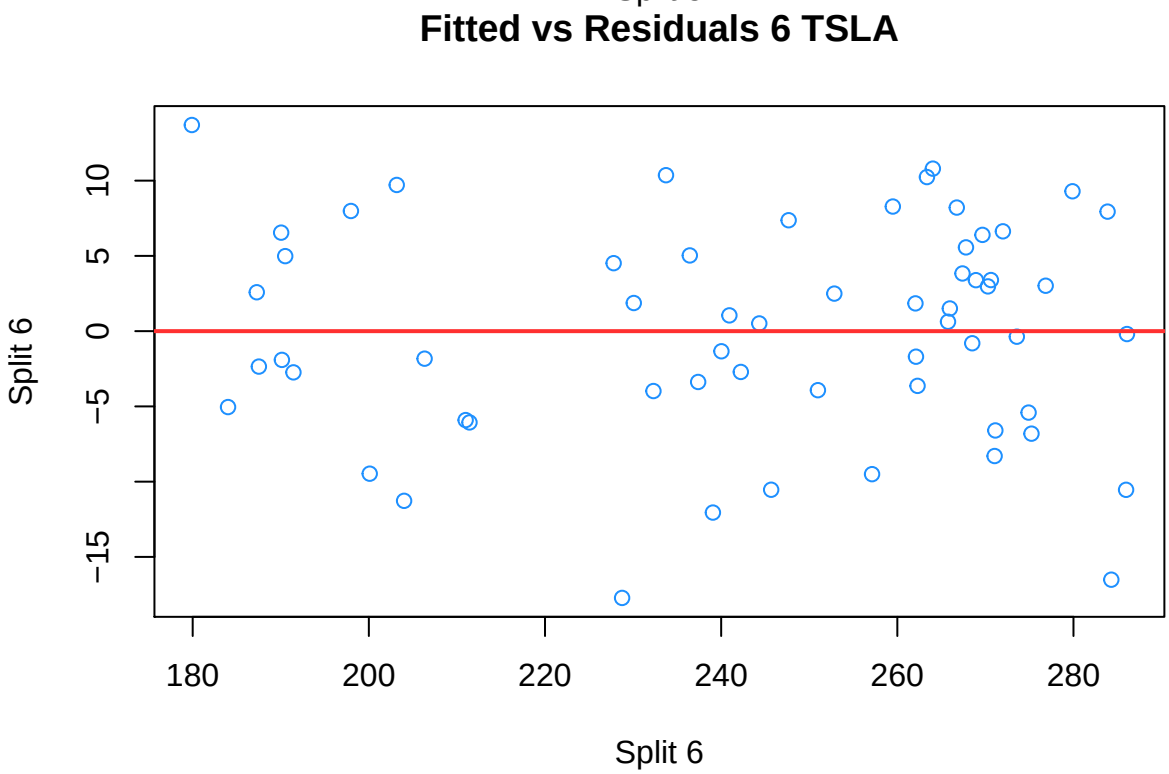

Fitted vs Residuals 1 TSLA
Split 1
10
0
−10
−20
−30
260
280
300
320
340
360
Split 1

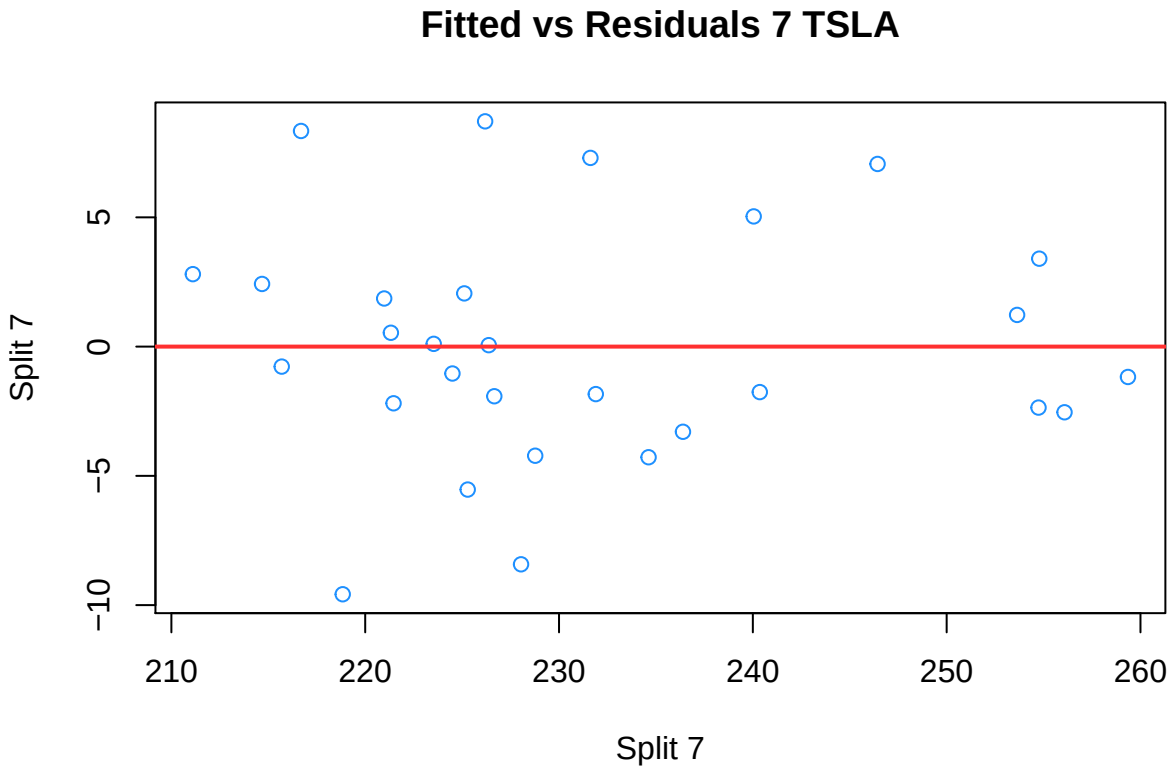

Fitted vs Residuals 2 TSLA
Split 2
20
10
0
−10
−20
260
280
300
320
340
360
Split 2


Fitted vs Residuals 3 TSLA
Split 3
40
20
0
−20
280
300
320
340
360
Split 3

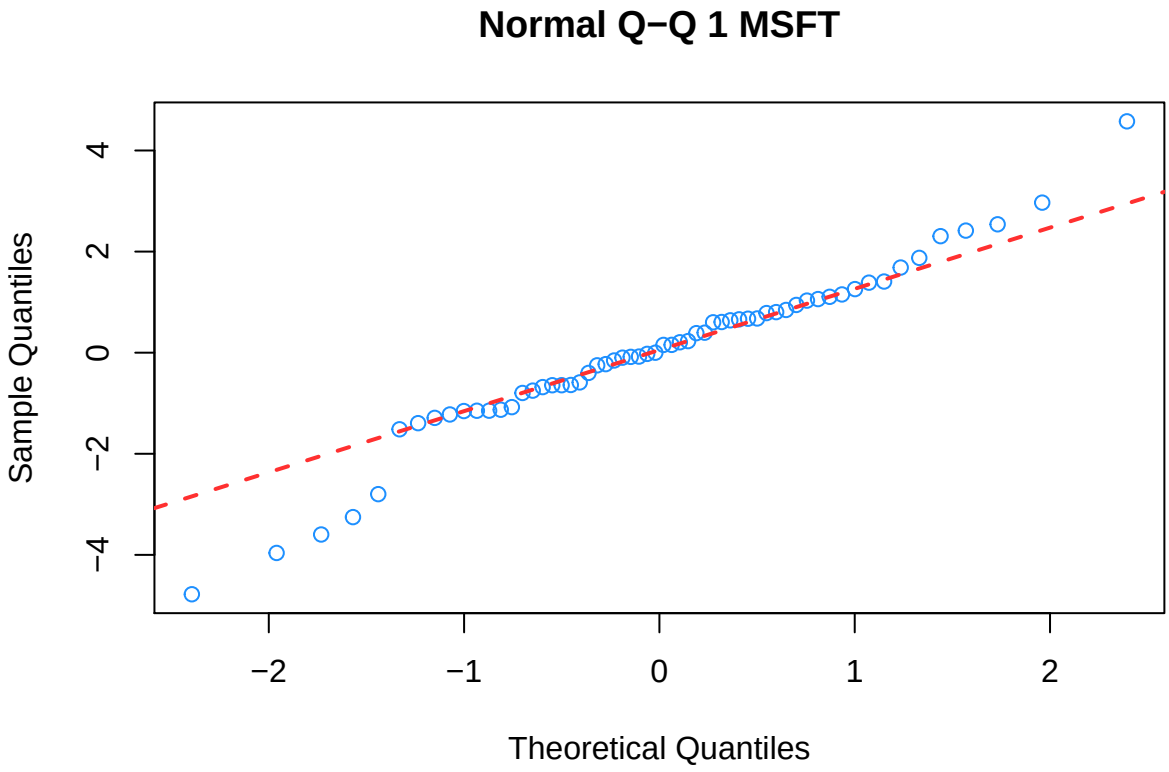

Fitted vs Residuals 4 TSLA
Split 4
20
0
−20
−40
260
280
300
320
340
360
380
Split 4

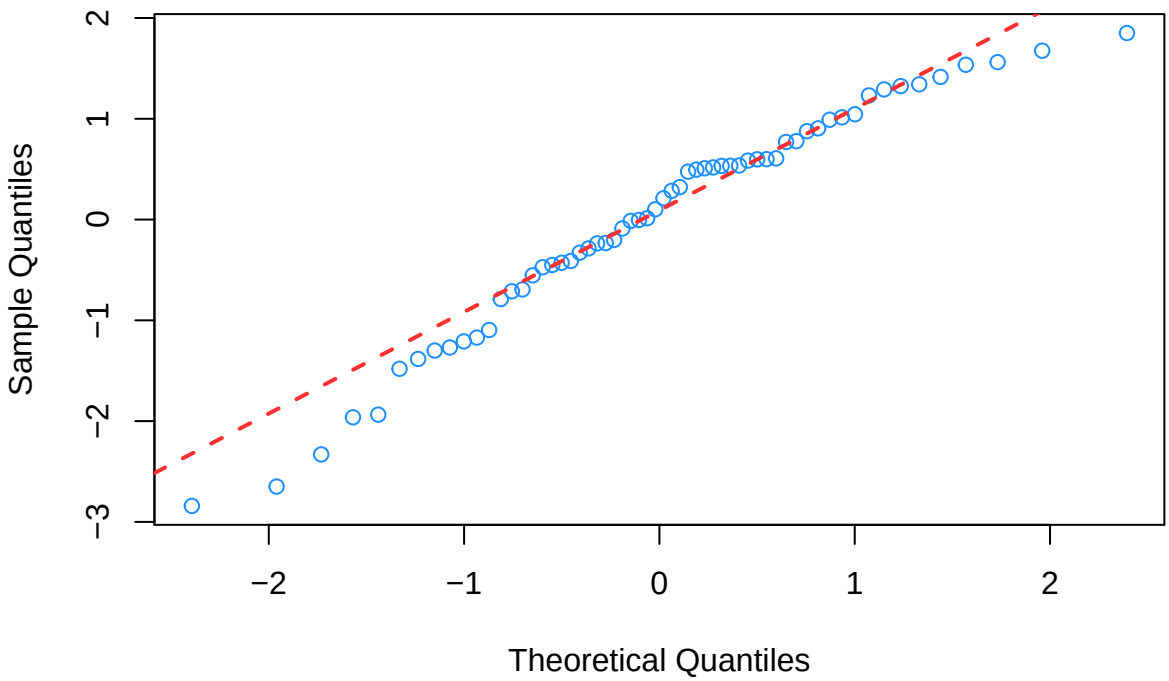

Fitted vs Residuals 5 TSLA
Split 5
20
10
0
−10
−20
−30
280
300
320
340
Split 5

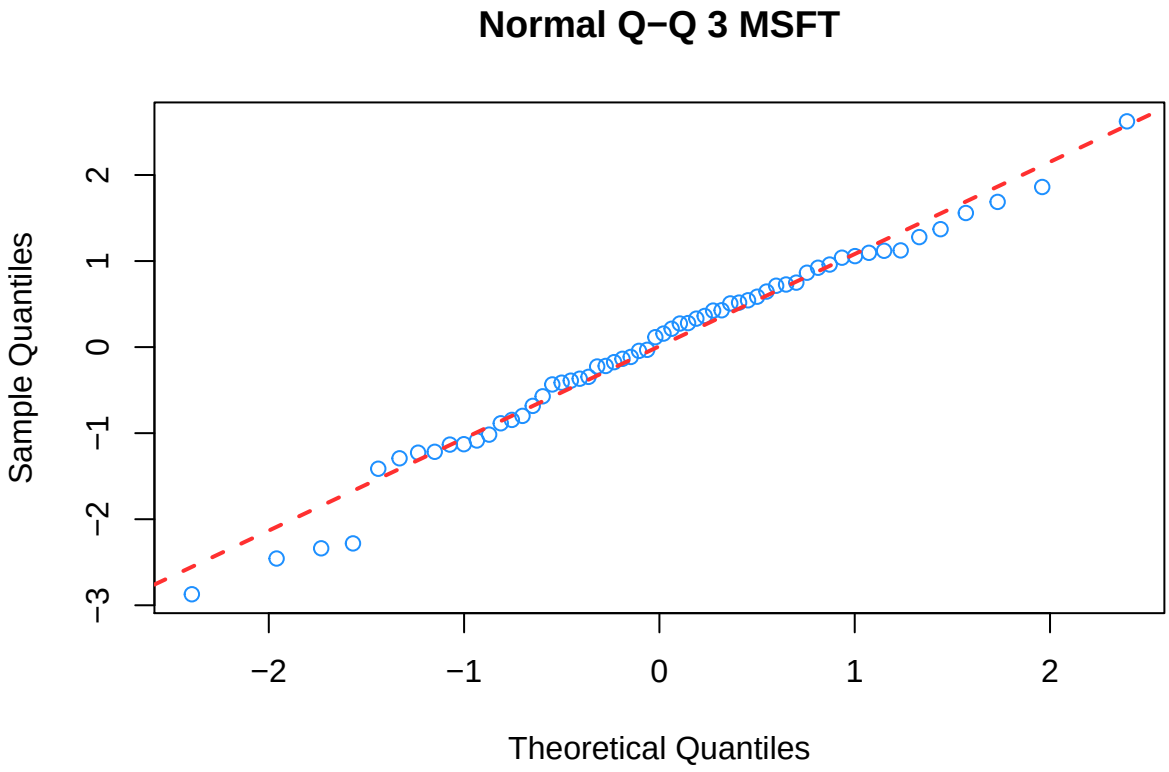

Fitted vs Residuals 6 TSLA
Split 6
10
5
0
−5
−15
180
200
220
240
260
280
Split 6

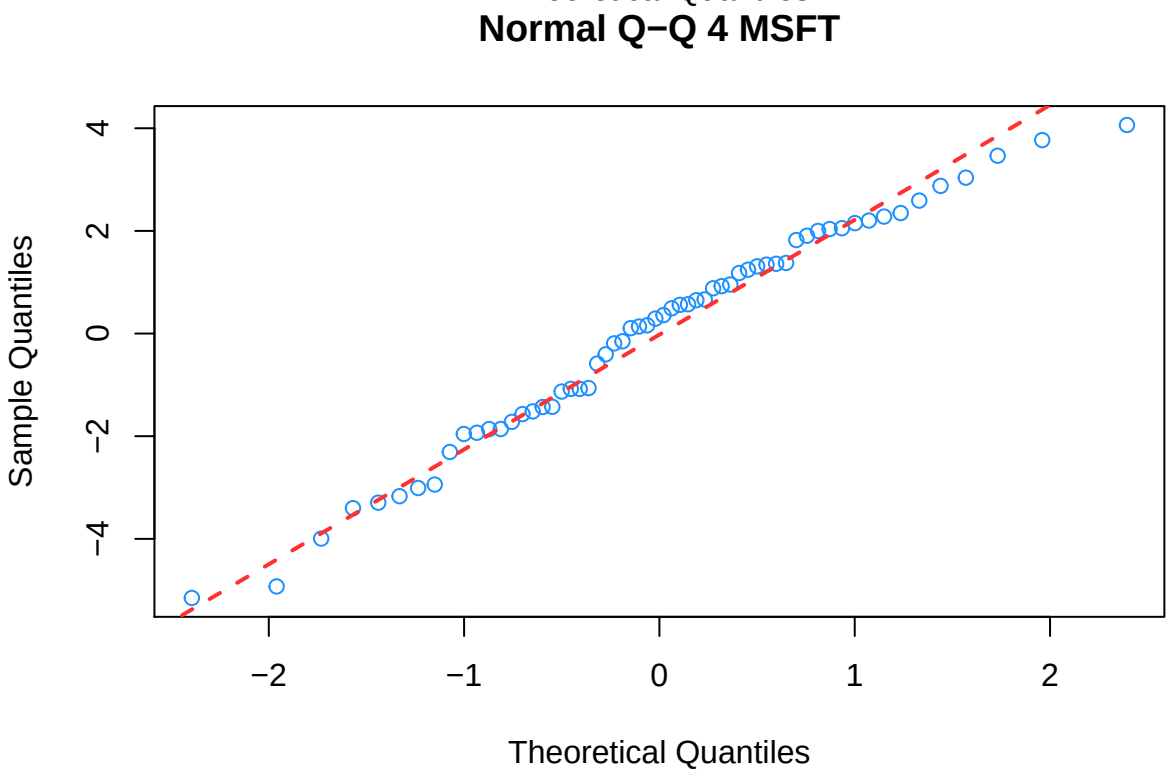


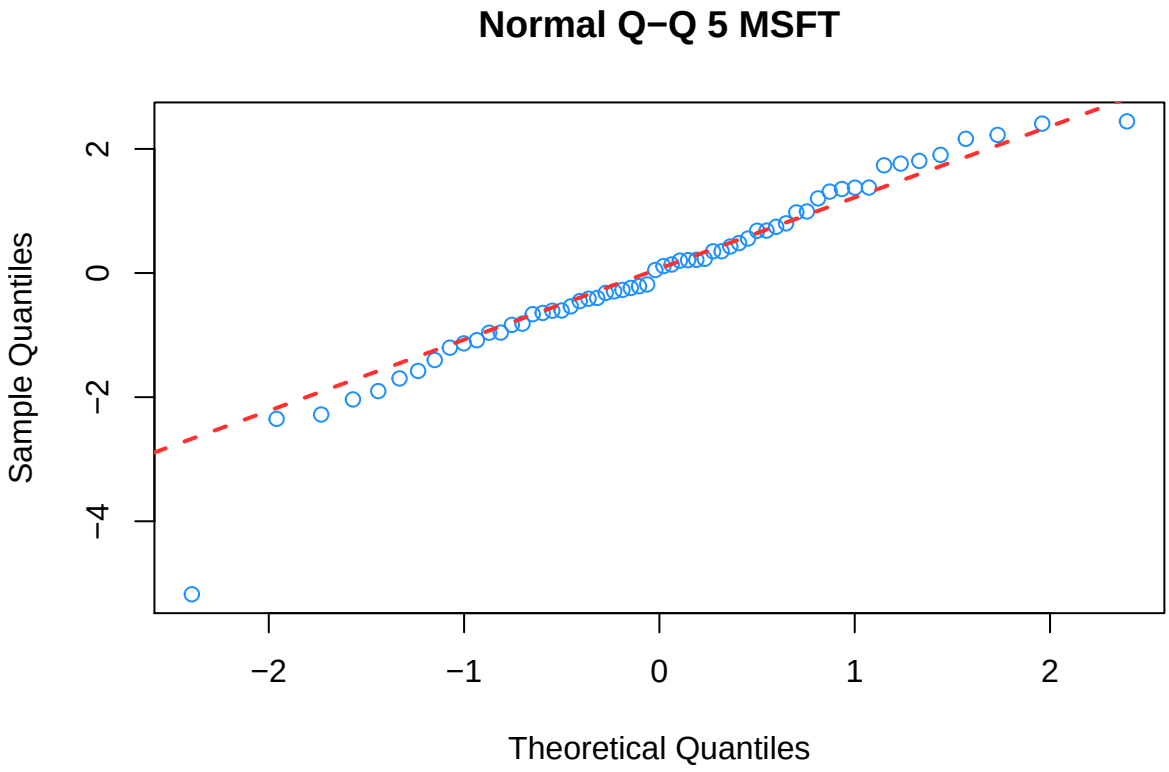

```r
generateQQ <- function(madeModels, name) {
  for (i in 1:length(madeModels$step)) {
    qqnorm(resid(madeModels$step[[i]]), col = "dodgerblue1", main = paste("Normal Q-Q", i, name))
    qqline(resid(madeModels$step[[i]]), lty = 2, lwd = 2, col = "firebrick1")
  }
}

generateQQ(modelsMsft, "MSFT")
```

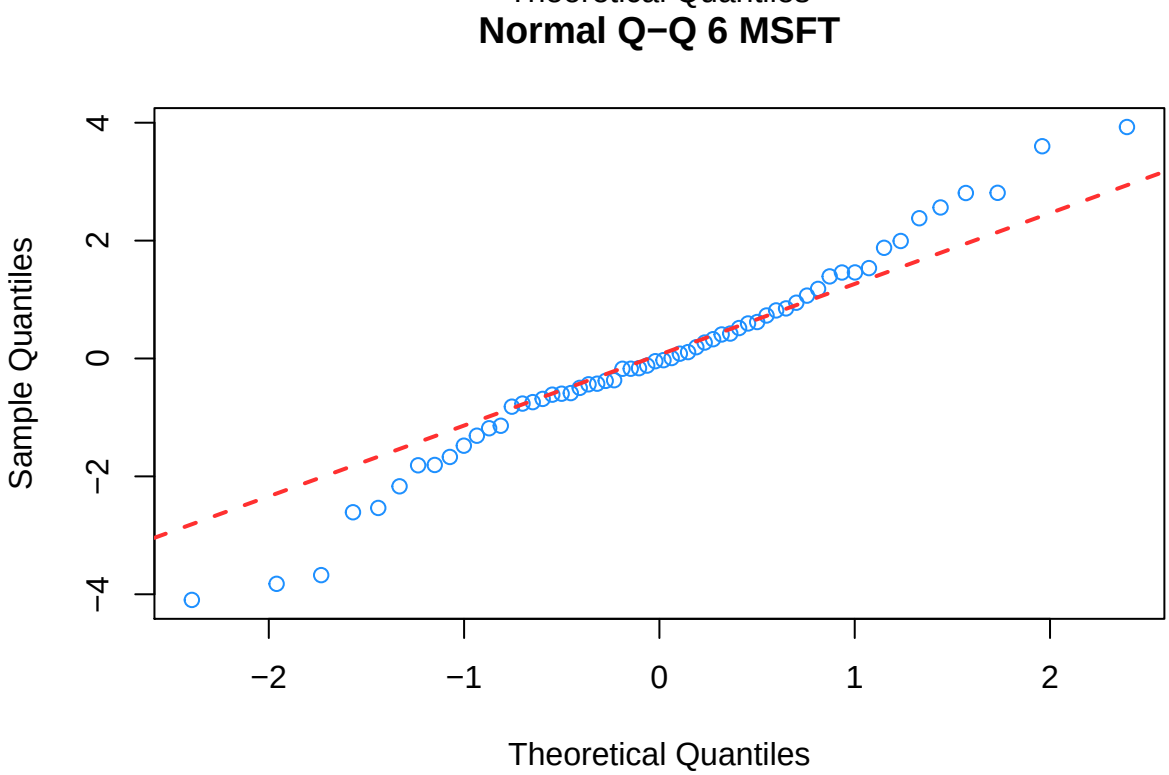

Normal Q−Q 1 MSFT
Sample Quantiles
Theoretical Quantiles
−4
−2
0
2
4
−2
−1
0
1
2

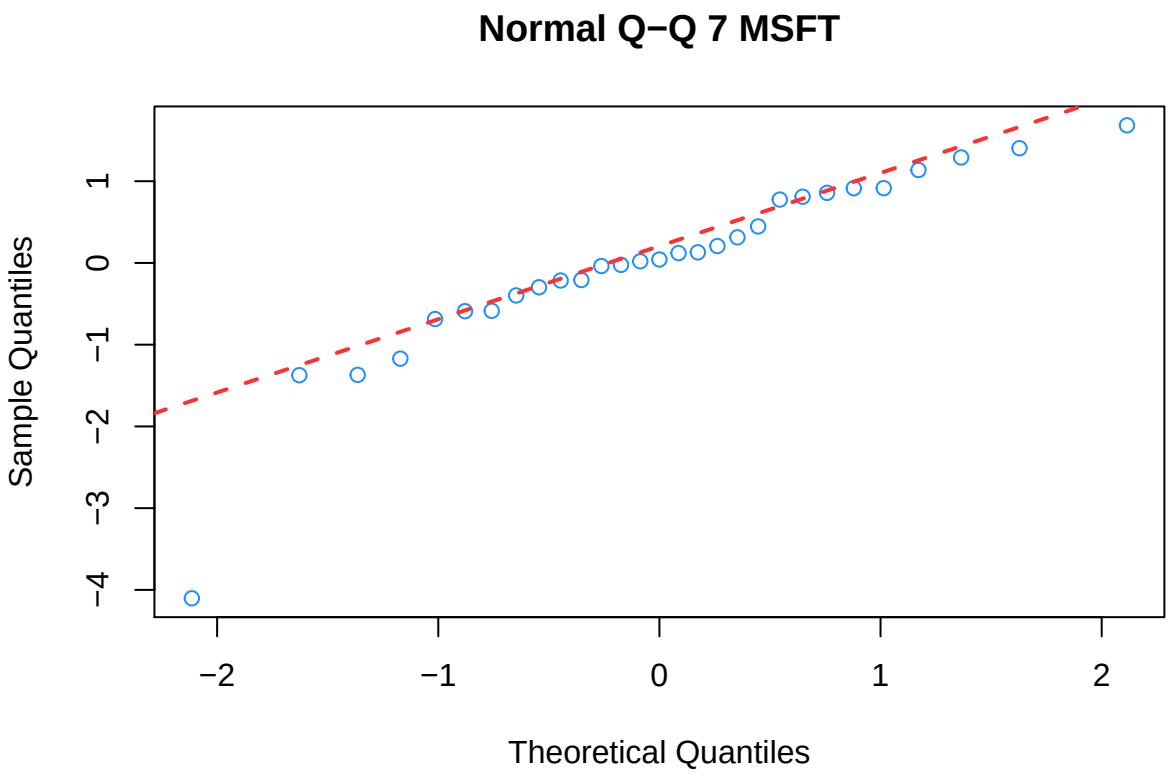

Normal Q−Q 2 MSFT
Sample Quantiles
Theoretical Quantiles
−3
−2
−1
0
1
2
−2
−1
0
1
2


Normal Q−Q 3 MSFT
Sample Quantiles
Theoretical Quantiles
−3
−2
−1
0
1
2
−2
−1
0
1
2

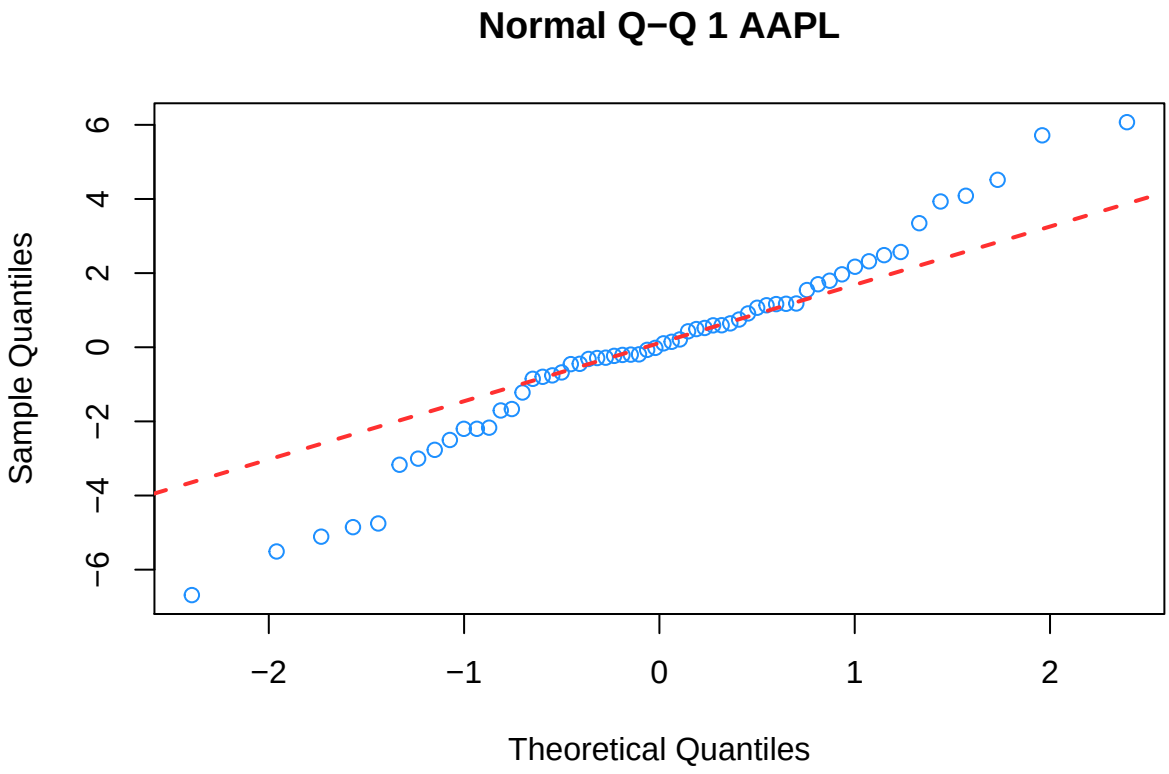

Normal Q−Q 4 MSFT
Sample Quantiles
Theoretical Quantiles
−4
−2
0
2
4
−2
−1
0
1
2

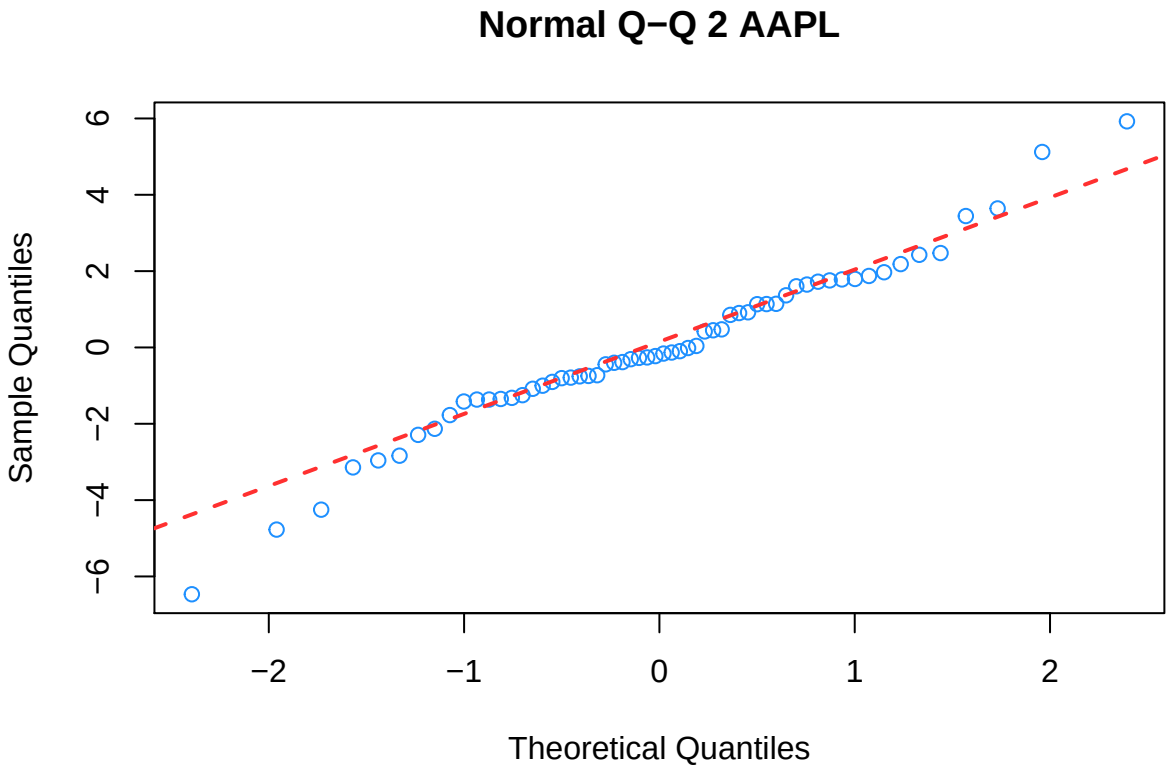

Normal Q−Q 5 MSFT
Sample Quantiles
Theoretical Quantiles
2
0
−2
−4
−2
−1
0
1
2

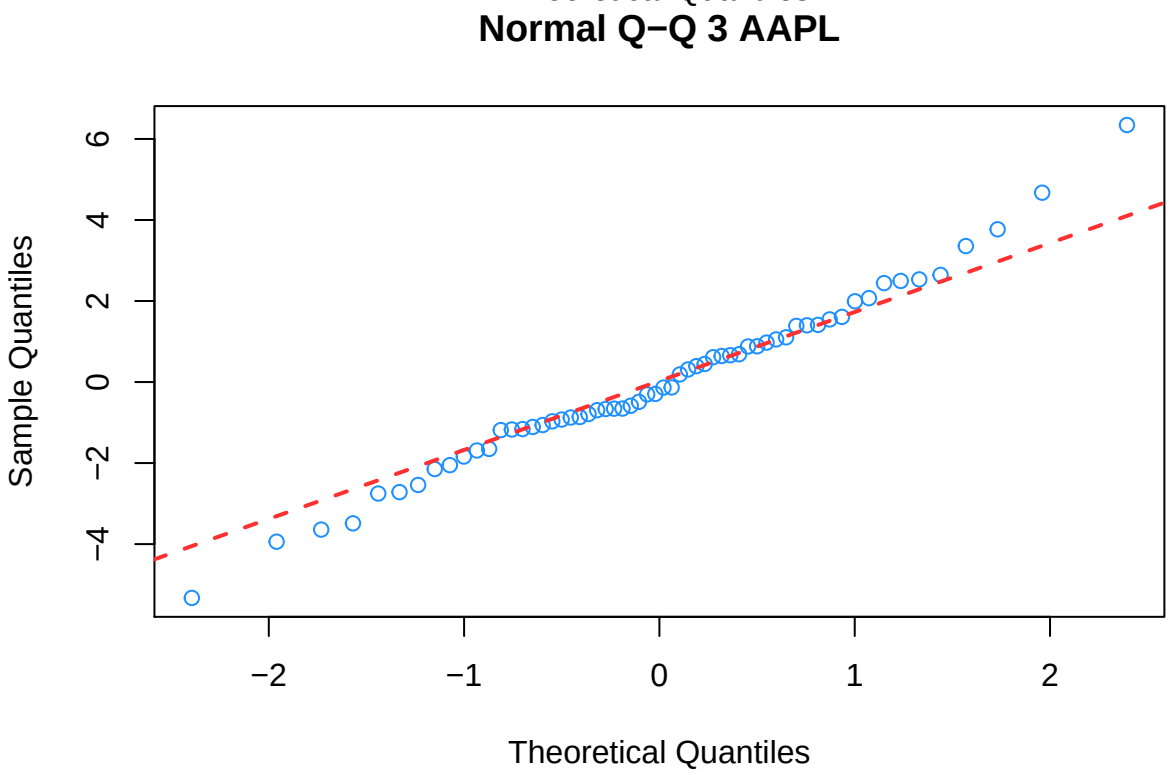

Normal Q−Q 6 MSFT
Sample Quantiles
Theoretical Quantiles
4
2
0
−2
−4
−2
−1
0
1
2

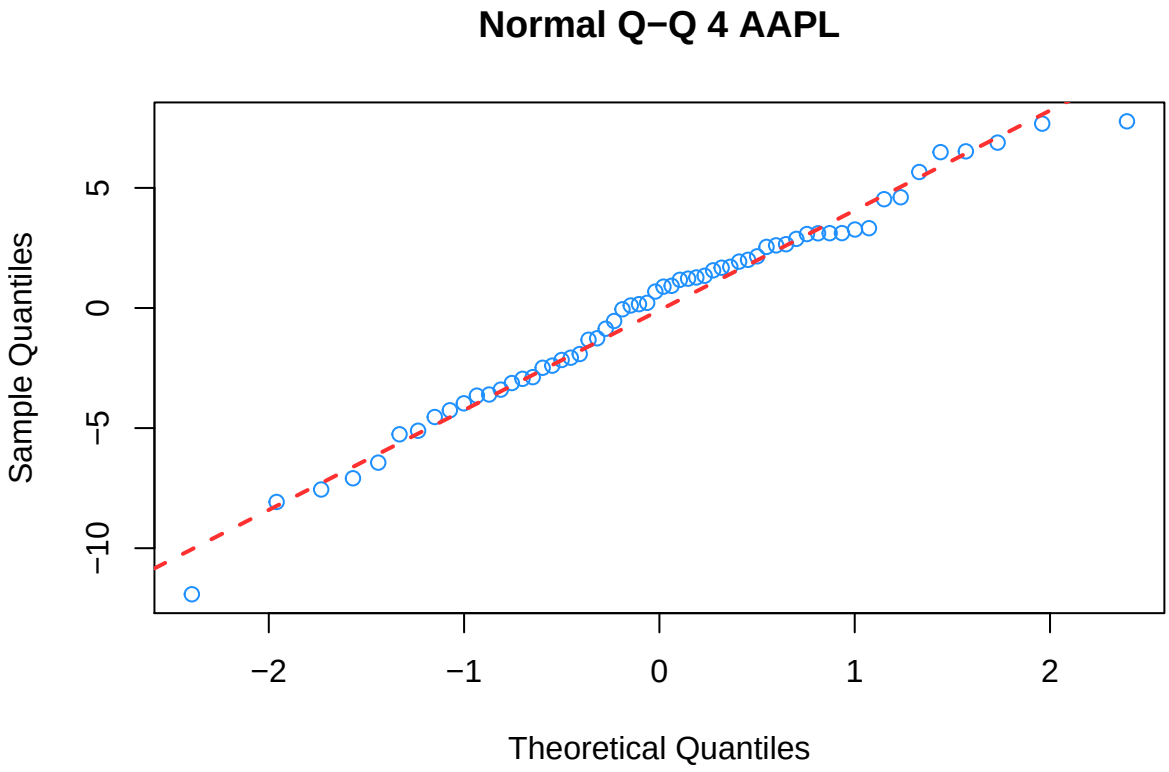


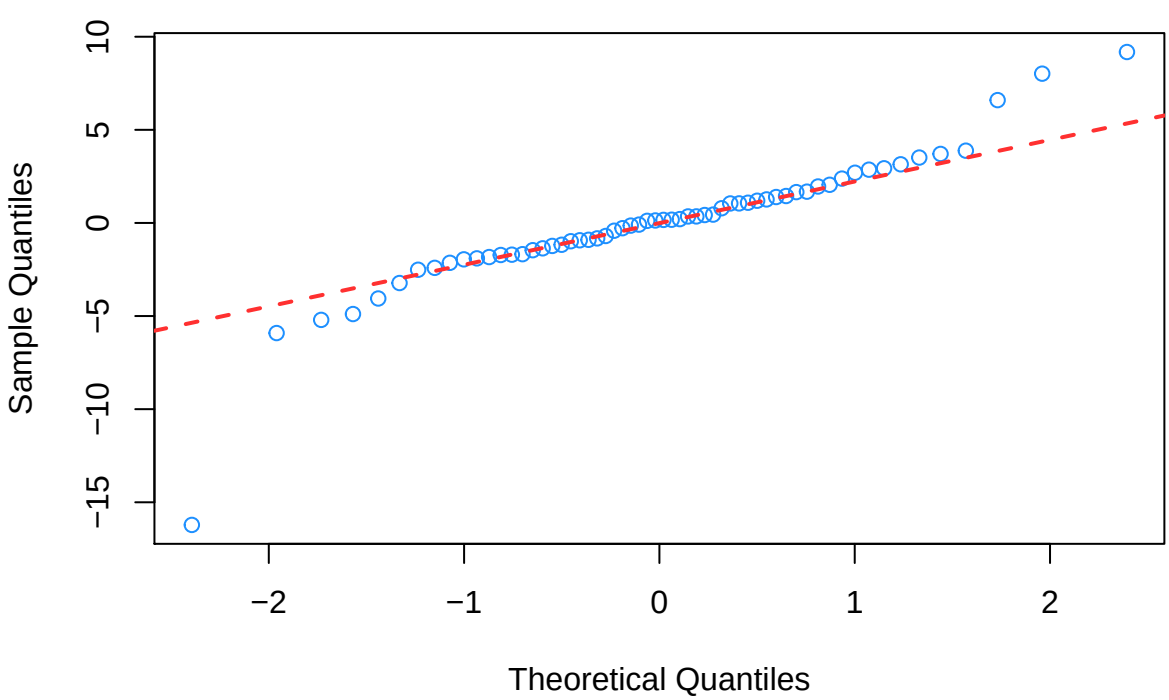

```
generateQQ(modelsAapl, "AAPL")
```

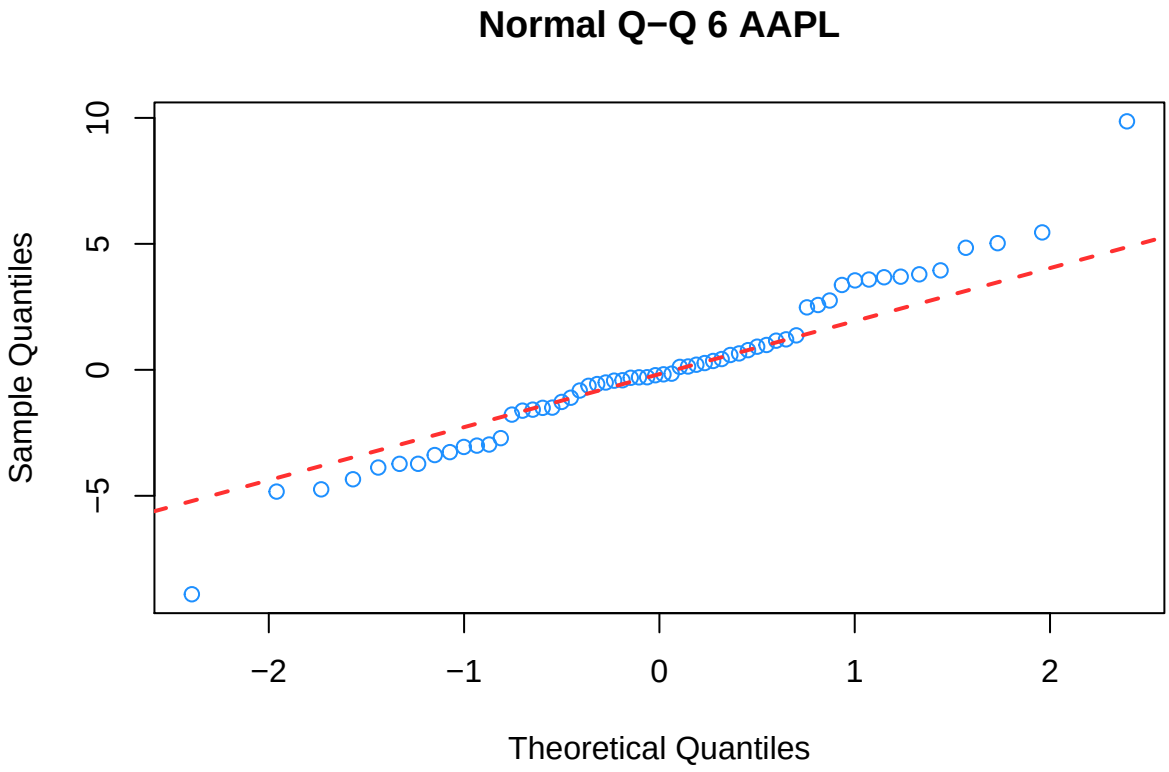


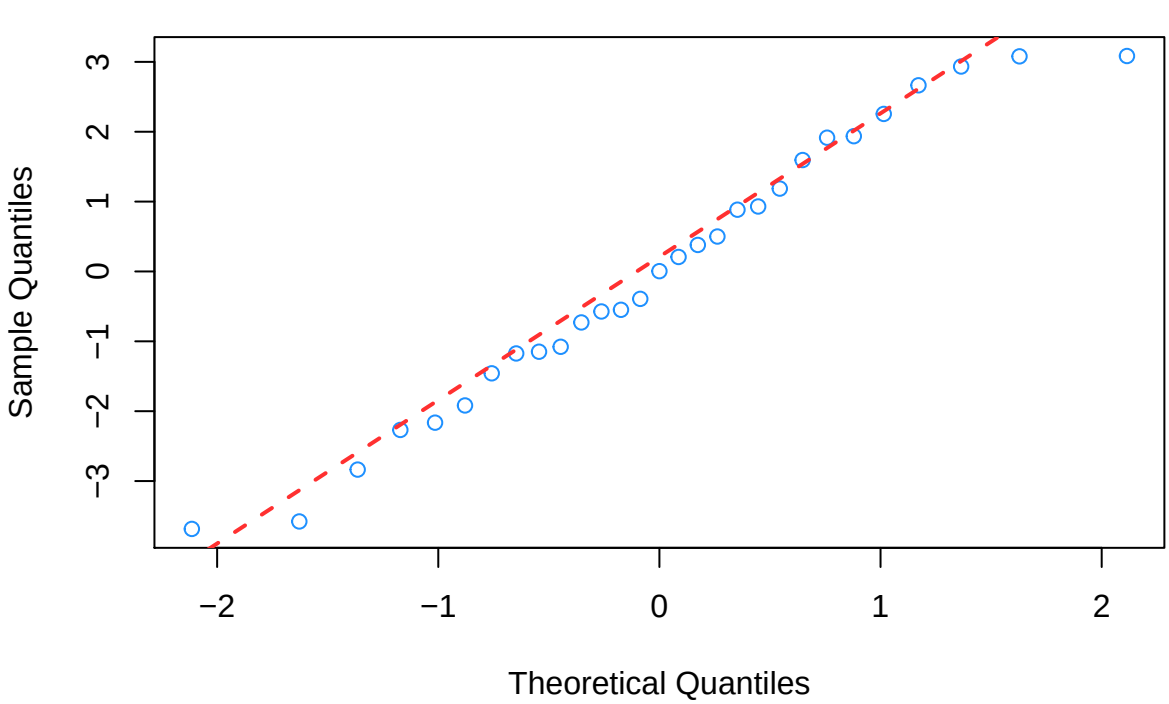

Normal Q−Q 2 AAPL
Sample Quantiles
Theoretical Quantiles
−6
−4
−2
0
2
4
6
−2
−1
0
1
2


Normal Q−Q 3 AAPL
Sample Quantiles
Theoretical Quantiles
−4
−2
0
2
4
6
−2
−1
0
1
2

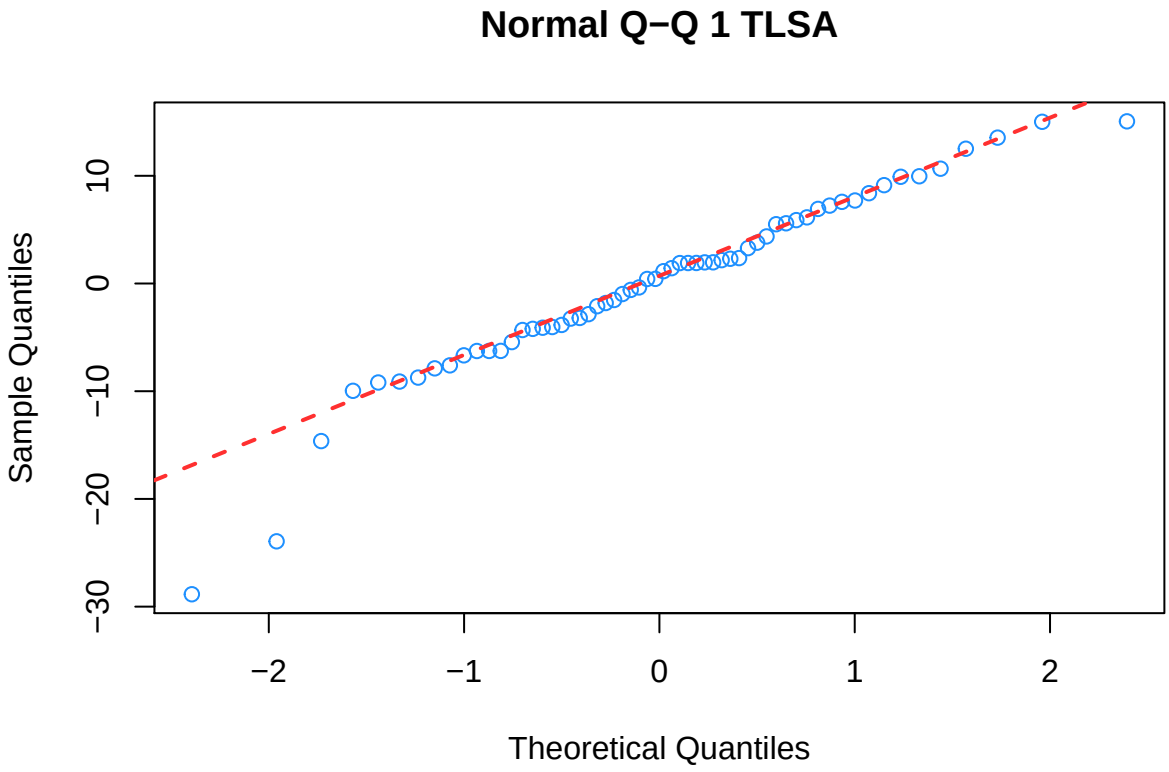

Normal Q−Q 4 AAPL
Sample Quantiles
Theoretical Quantiles
5
0
−5
−10
−2
−1
0
1
2

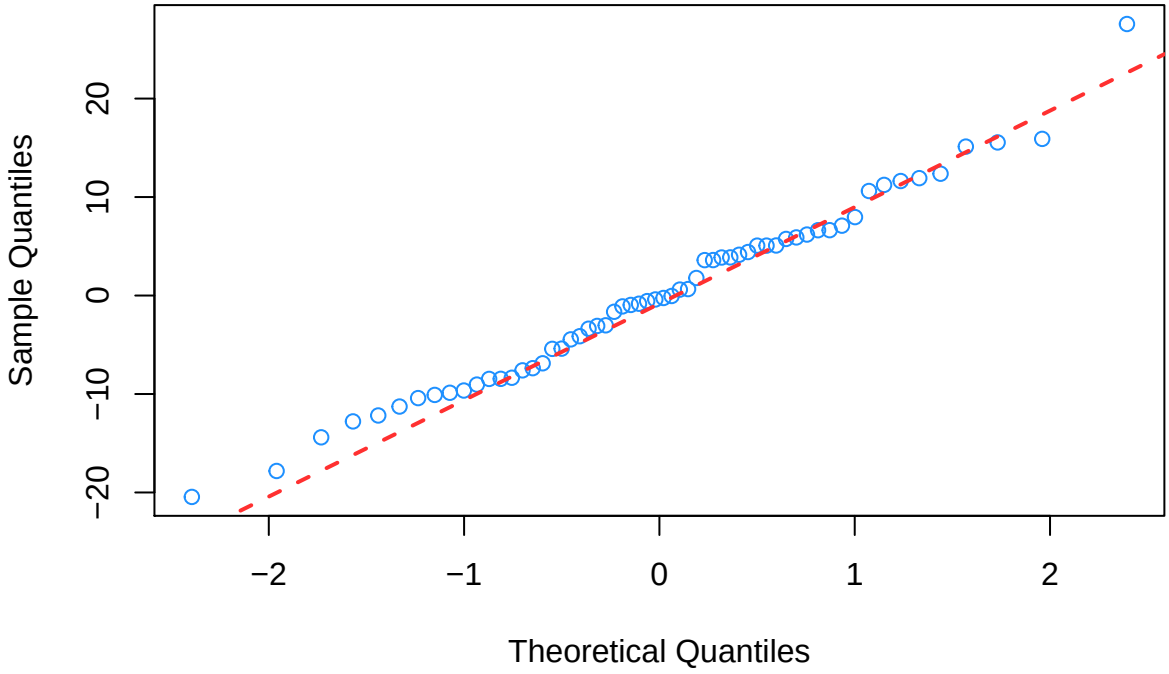

Normal Q−Q 5 AAPL
Sample Quantiles
Theoretical Quantiles
10
5
0
−5
−10
−15
−2
−1
0
1
2

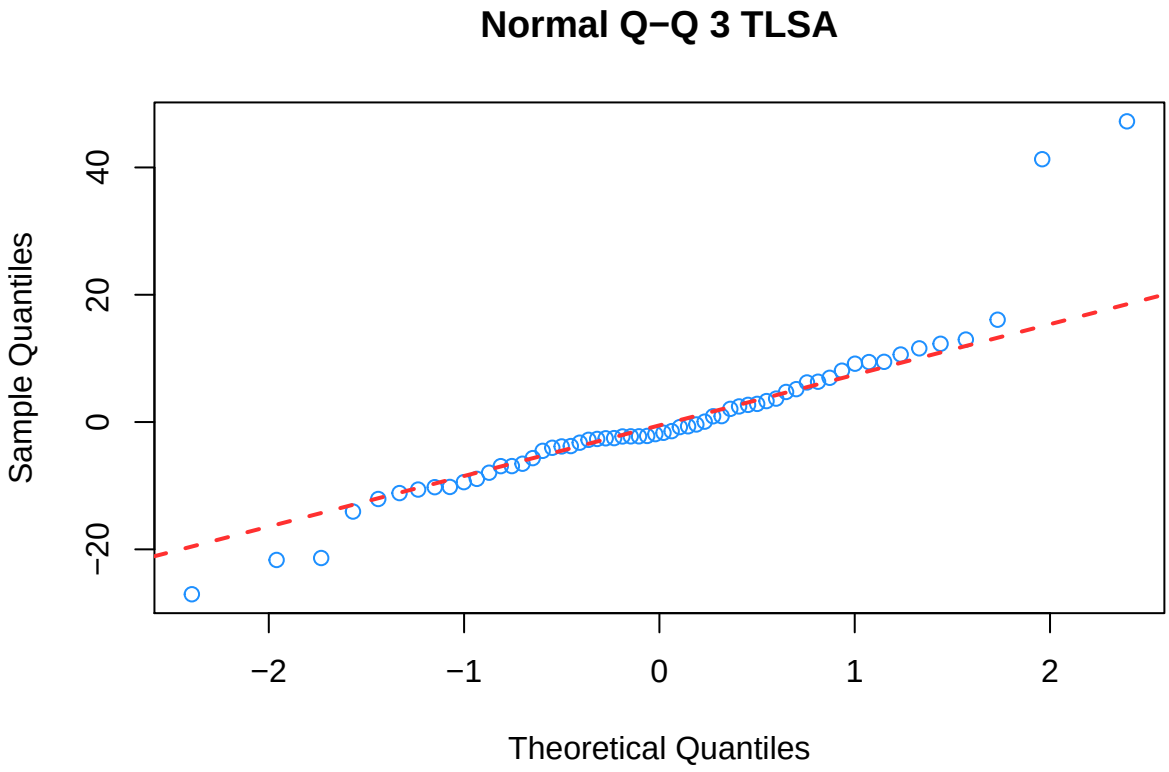


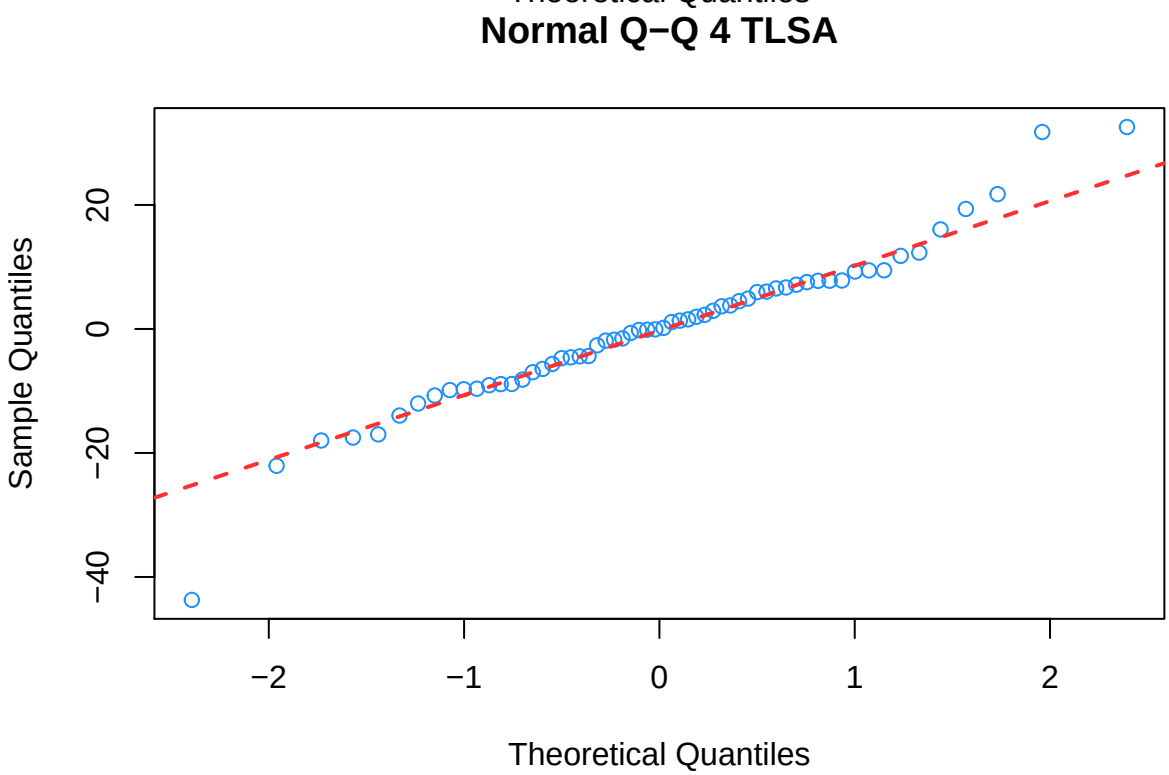


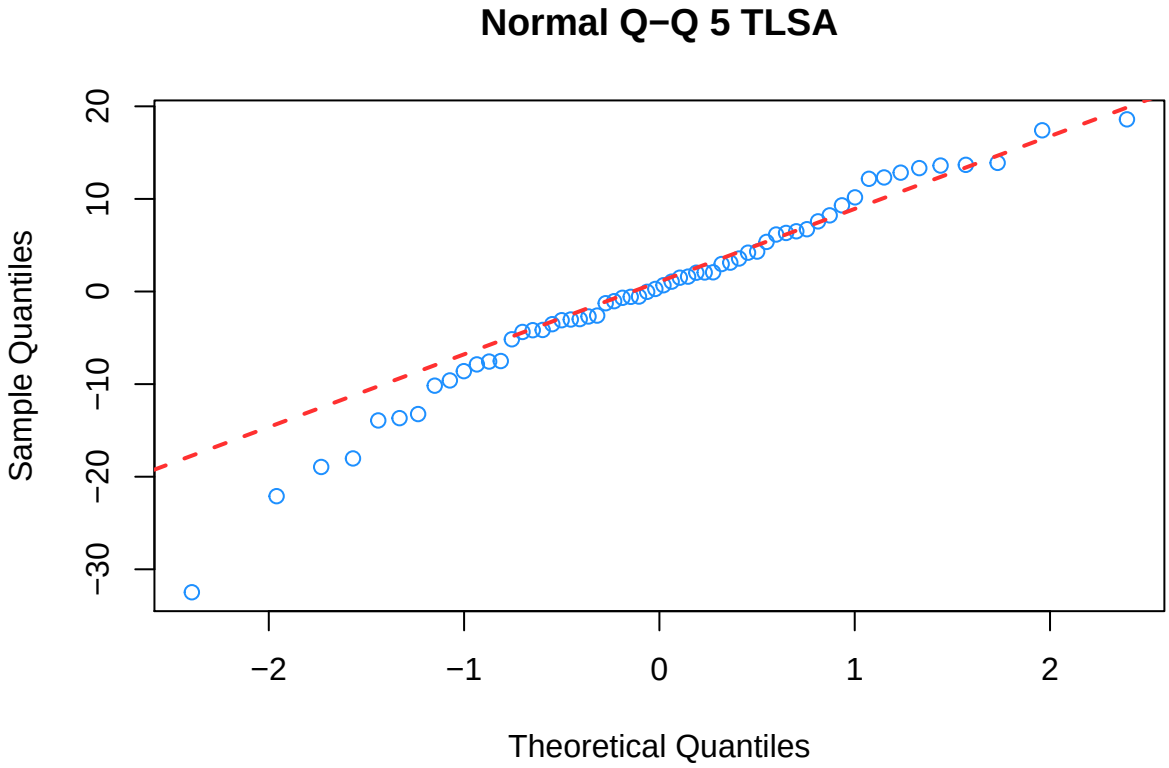

```
generateQQ(modelsTsla, "TLSA")
```

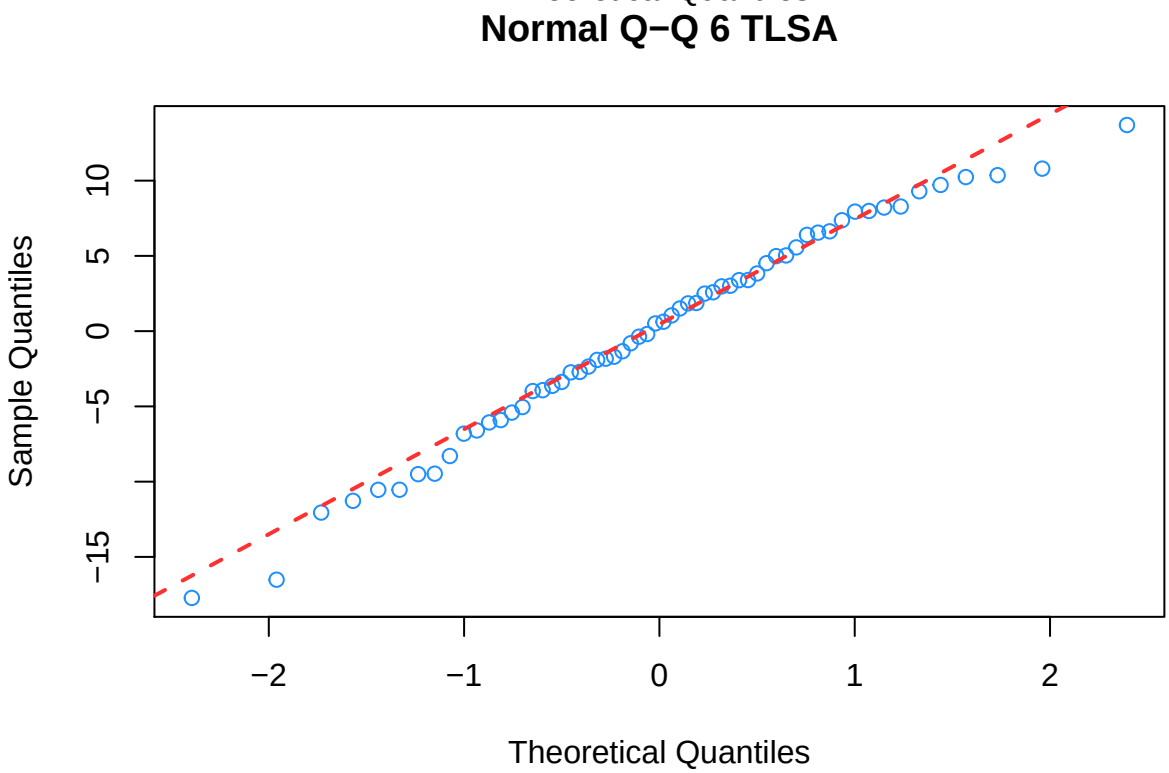

Normal Q−Q 1 TLSA
Sample Quantiles
10
0
−10
−20
−30
−2
−1
0
1
2
Theoretical Quantiles

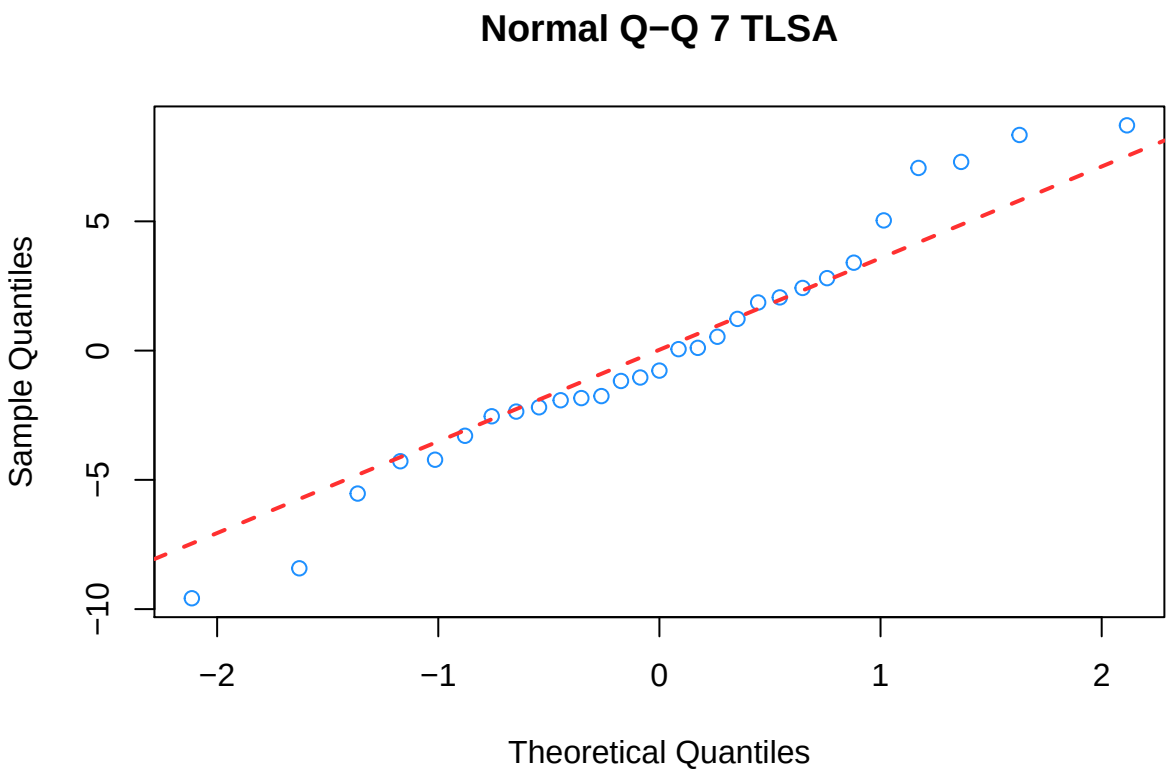

Normal Q−Q 2 TLSA
Sample Quantiles
20
10
0
−10
−20
−2
−1
0
1
2
Theoretical Quantiles


Normal Q−Q 3 TLSA
Sample Quantiles
Theoretical Quantiles
−20
0
20
40
−2
−1
0
1
2


Normal Q−Q 4 TLSA
Sample Quantiles
Theoretical Quantiles
−40
−20
0
20
−2
−1
0
1
2


Normal Q−Q 5 TLSA
Sample Quantiles
Theoretical Quantiles
20
10
0
−10
−20
−30
−2
−1
0
1
2


Normal Q−Q 6 TLSA
Sample Quantiles
Theoretical Quantiles
10
5
0
−5
−15
−2
−1
0
1
2


Normal Q−Q 7 TLSA
Sample Quantiles
Theoretical Quantiles
−10
−5
0
5
−2
−1
0
1
2