Matthew Bolding
Dr. Nelis Potgieter
MATH 40853–015
March 10th, 2023

Regression and Time Series: Project 1

1 Training Linear Models

Before beginning to train any models from the `txhousing` dataset, we must first split the housing data for three selected markets, those being Fort Worth, Austin, and Houston. Many of the following operations, like this one, involve R code behind the scenes. This part's code may be found here.

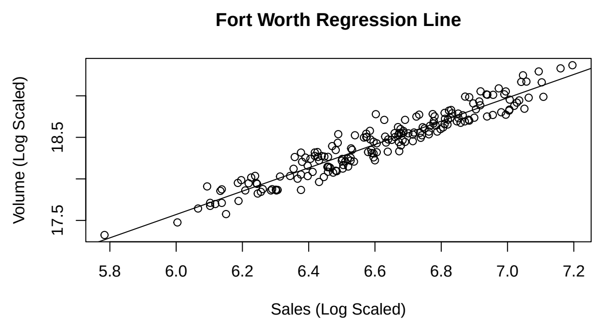
1.1 Fort Worth Housing Market

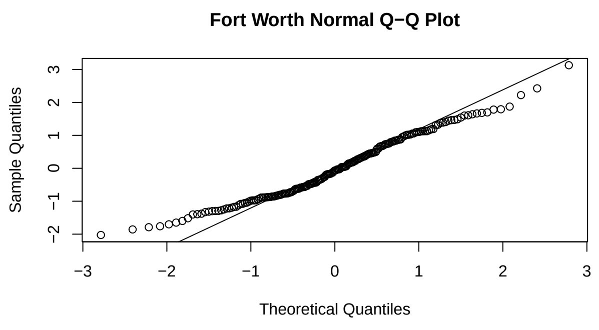
The model for the Fort Worth housing market—trained in this way—may be written in the form of

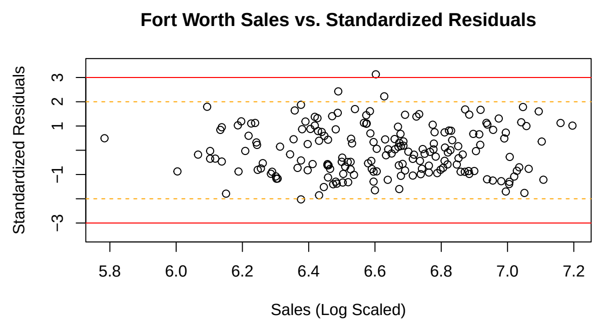
$$\widehat{\text{Volume}}_{\texttt{Fort Worth}} = 9.14352 + 1.40460\,\text{Sales}$$

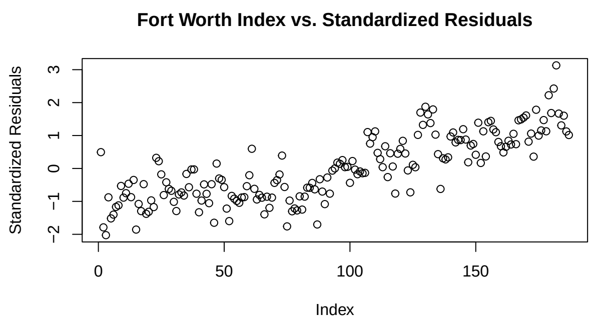
where both the predictor and the outcome variable are log scaled. Running the `summary` command on the trained linear model, we can recover the R^2 metric with the `Multiple R-squared` value, which is the proportion of variability in the dataset explained by the model. This model's R^2 value is 0.9166023.

Now we proceed with a residual analysis, a crucial task in validating the model for the dataset. A residual is what has been left unexplained by the model, i.e., $e_i = y_i - \hat{y}_i$, and residuals may be standardized by the expression $e_i^* = e_i(\text{MSE})^{-1/2}$, where MSE is the mean square error of the model—an estimator for the error variance term σ_ϵ^2. In the case of simple linear regression, $\text{MSE} = \frac{1}{n-2}\sum_{i=1}^{n} e_i^2$. Hence, the greater the e_i, the higher the model's MSE. These functions return R^2 and MSE values.









First, we inspect the **Fort Worth Normal Q-Q Plot** plot. (See the function to create all plots here.) Observe that the superimposed line passes through the quantile points. We may then make a reasonable assumption that the data follows a normal distribution.

Consider the **Fort Worth Sales vs. Standardized Residuals** plot. Before making any inferences from the plot, recall the 68–95–99.7 rule which stipulates that within one standard deviation from the mean in a normal distribution, we should expect to see 68% of the values, within two standard deviations, 95%, and within three standard deviations, 99.7%. Therefore, having values that are differ from the mean by more than 2 or 3 standard deviations aren't a problem in and of themselves—in fact, they might contains valuable information about the dataset.

Recall e_i^*, and note that any observation with $|e_i^*| \geq 2$ is a potential outlier, and since the regression coefficients b_0 and b_1 for intercept and slope, respectively, are weighted values, these observations might have undue influence on the model's accuracy, slope, and intercept terms. We see that 4 of 187 e_i^* are not in the interval $[-2, 2]$, which makes up merely 2.14% of the dataset. There exists 1 $e_i^* \notin [-3, 3]$, but even still, this singular data point consists of only 0.5% of all values, so these anomalies should not be cause for concern since n is large.

Viewing **Fort Worth Index vs. Standardized Residuals** shows that there might be some relationship between the order in which the data samples were collected and the error of the model. The model, trained on chronologically sorted data, underestimates the home volume for earlier data points but overestimates for more recent ones. And what's more, there's a non-trivial correlation between the `date` of collection and the `volume`: 0.6668106.

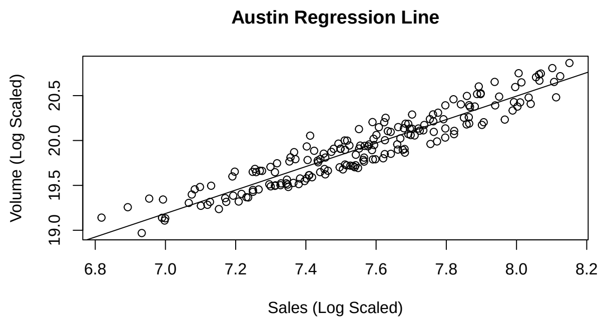
Although this relationship exists, this fact does not undermine the model as a whole—the model does not suffer from heteroscedasticity, a situation in which variance varies by observation. While the ϵ_i, the true residuals, might not be completely independent, the individual observations maintain random variance. This observation might suggest that one month's housing market appears to influence the next, so this problem could likely benefit from a time series analysis.

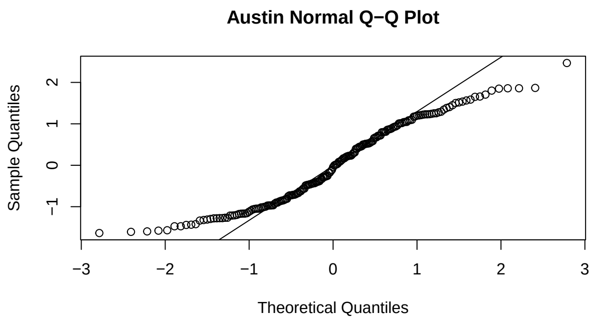
1.2 Austin Housing Market

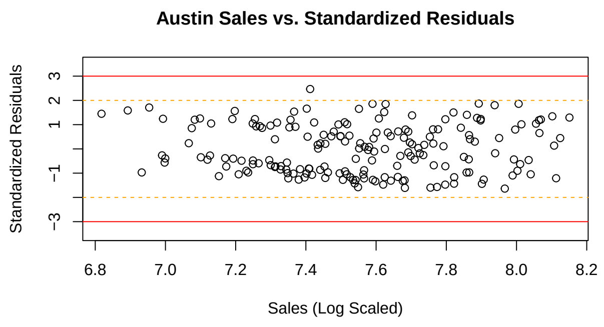
Like for Fort Worth, a model may be trained for the Austin housing market, and the equation of the trained model, where the predictor and outcome variables are log-scaled, is

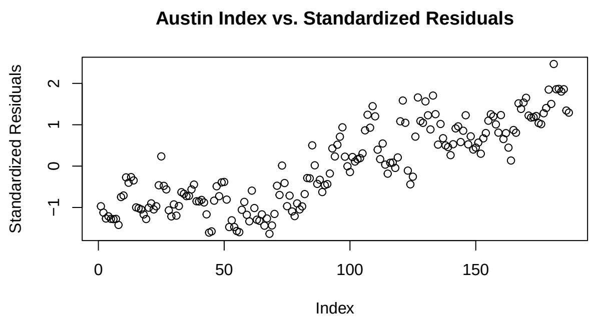
$$\widehat{\text{Volume}}_{\text{Austin}} = 10.03255 + 1.30772\,\text{Sales}.$$

Similarly to `fortWorthModel`, we may recover the a metric that assesses the model's accuracy, the R^2 value, with the `Multiple-R-squared` field; this model's R^2 value is 0.8918986. This model is similarly predictive.









In terms of the residual analysis, we see much of the same. The **Austin Normal Q-Q Plot** shows a good fit to normality, and **Austin Sales vs. Standardized Residuals** shows that there's only one observation which could be an outlier, and no $|e_i^*| \geq 3$. There's no overwhelming evidence that the data's not normal, so we will assume it fits to the normal distribution.

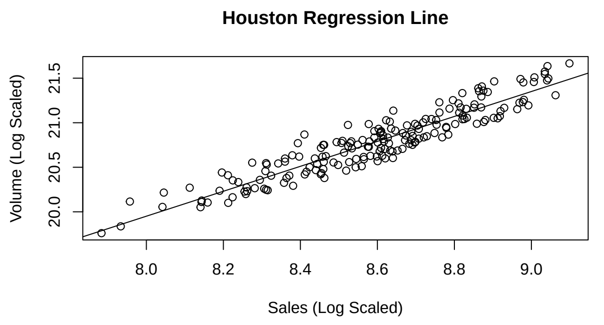
And in the **Austin Sales vs. Standardized Residuals** graph, we again see a relationship between the standardized residuals and their index. Like the corresponding graph for Fort Worth, we do not see the variability of the dataset change, however; the variance of the e_i^* appear to remain constant.

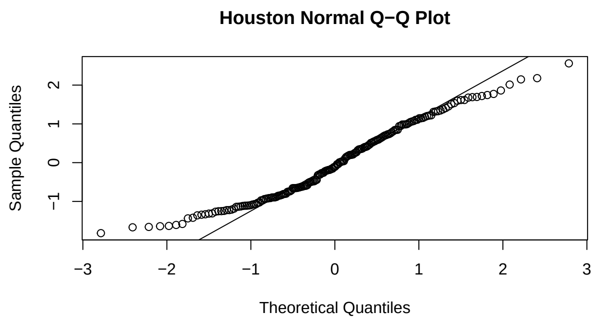
1.3 Houston Housing Market

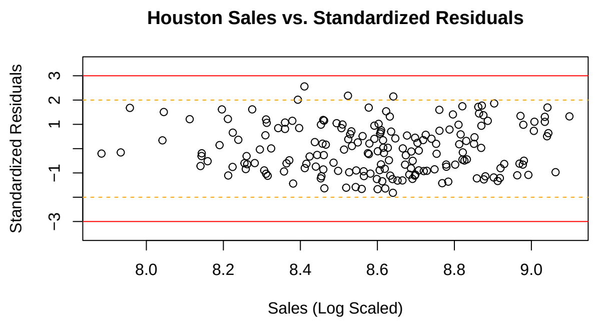
A model may be trained for the Houston housing market and its equation, where both `Volume` and `Sales` are log scaled, is

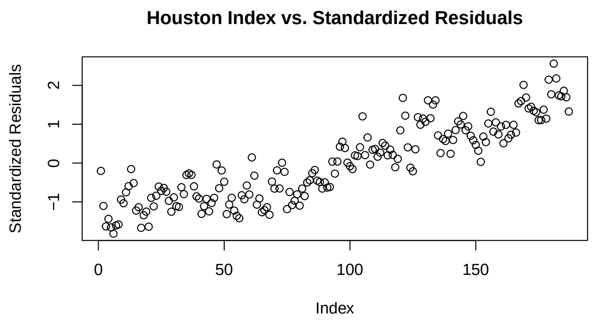
$$\widehat{\text{Volume}}_{\text{Houston}} = 8.75218 + 1.39982\,\text{Sales}.$$

According to the `Multiple R-squared` statistic, the model has an R^2 value of 0.8741417, so the model explains 87.41% of the variability in the Again, the model predicts the dataset well according to this metric.









The residual analysis plot **Houston Sales vs. Standardized Residuals** for Houston's housing market model shows much of the same. Although there are 4 potential outliers, these are not of great concern; they make up a small portion of the dataset. Along with the **Houston Normal Q-Q Plot**'s display of normality, we can assume that the data fits the normal distribution.

As we would almost come to expect at this point, the **Houston Index vs. Standardized Residuals** scatter plot shows a relationship between an observation's index and its standardized residual. The variance between observations appears to stay the same, however, so the model retains homoscedasticity.

2 Slope Parameter Comparison

We might want to compare the slopes of all the models to determine if there's a similar relationship between the three housing markets analyzed. To perform the comparison, we will construct a 95% confidence interval, and if there's overlap between the intervals, then one could reasonably say that the relationship between sales and volume within one market is similar to relationships in other markets.

We can calculate these confidence intervals with the following equation $\left(b_1 \pm t_{\alpha/2,n-2} \cdot se(b_1)\right)$, where $se(b_1)$ is the standard error of the slope term. Here, we use the t-distribution since we assumed normality for all the markets analyzed.

See the function definition for `confidenceInterval95` here and the one for `printCIInterval` here.

```
printCIInterval("Fort Worth", confidenceInterval95(fortWorthModel, fortWorthHousing))
printCIInterval("Austin", confidenceInterval95(austinModel, austinHousing))
printCIInterval("Houston", confidenceInterval95(houstonModel, houstonHousing))
```

```
## Fort Worth 95% Confidence Interval: (1.3431, 1.4661)
## Austin 95% Confidence Interval: (1.2417, 1.3738)
## Houston 95% Confidence Interval: (1.3228, 1.4769)
```

Observe that within the 95% confidence intervals for the slopes of all models across the markets, an overlap does exist. Hence, the relationship between sales and volume is comparable across markets.

3 95% Prediction Interval for `volume`

We will now calculate the 95% prediction intervals for each of the markets with sales totaling 500 units. The equation for the confidence interval, $\left(\widehat{\text{Volume}}^* \pm t_{\alpha/2,n-2} \cdot se(\text{pred})\right)$, but it uses the standard error of the prediction $\widehat{\text{Volume}}^*$ is the predicted value. Again, we use the t-distribution for the same reason as we did for the confidence interval calculations.

See the function definition for `prediciton Interval95` here, and go here to see the definition for `printPIInterval`.

```
printPIInterval("Fort Worth", predictionInterval95(fortWorthModel, fortWorthHousing, 500))
printPIInterval("Austin", predictionInterval95(austinModel, austinHousing, 500))
printPIInterval("Houston", predictionInterval95(houstonModel, houstonHousing, 500))
```

```
## Fort Worth 95% Prediction Interval: (27899414.967, 119757325.516)
## Austin 95% Prediction Interval: (36454312.589, 162700473.909)
## Houston 95% Prediction Interval: (18252380.107, 78858507.721)
```

4 Appendix

Here is all the code I used to compile this report.

Importing and splitting the data into the various markets.

```
library(tidyverse)

fortWorthHousing <- txhousing[txhousing$city == "Fort Worth", ]
austinHousing <- txhousing[txhousing$city == "Austin", ]
houstonHousing <- txhousing[txhousing$city == "Houston", ]
```

Training models for each of the selected markets.

```
fortWorthModel <- lm(log(volume) ~ log(sales), data = fortWorthHousing)
austinModel <- lm(log(volume) ~ log(sales), data = austinHousing)
houstonModel <- lm(log(volume) ~ log(sales), data = houstonHousing)
```

Recovering the Mean Square Error and R^2 values.

```
getMSE <- function(model) {
  return(summary(model)$sigma^2)
}

getR2 <- function(model) {
  return(summary(model)$r.squared)
}
```

Plotting four graphs with a single function: the regression line, the Normal Q-Q plot, sales verses standardized residuals, and index verses standardized residuals.

```
createPlots <- function(city, model, housingData) {
  # Regression Line Plot
  plot(log(housingData$volume) ~ log(housingData$sales),
       xlab = "Sales (Log Scaled)",
       ylab = "Volume (Log Scaled)",
       main = paste(city, "Regression Line"))
  abline(model, untf=F)

  # Calculate the Standardized Residuals
  residuals <- model$residuals
  standardizedResiduals <- residuals/summary(model)$sigma

  # QQ Plot
  qqnorm(standardizedResiduals,
         main = paste(city, "Normal Q-Q Plot"))
  qqline(standardizedResiduals)

  # Plot scaled Sales vs. Standardized Residuals
  plot(log(housingData$sales), standardizedResiduals,
       xlab = "Sales (Log Scaled)",
       ylab = "Standardized Residuals",
       main =  paste(city, "Sales vs. Standardized Residuals"),
       ylim = c(-3.5, 3.5))

  abline(a = -2, b = 0, lty = 2, col = "orange")
  abline(a = 2, b = 0, lty = 2, col = "orange")
  abline(a = -3, b = 0, col = "red")
  abline(a = 3, b = 0, col = "red")

  # Plot Index vs. Standardized Residuals
  plot(standardizedResiduals,
       ylab = "Standardized Residuals",
       main = paste(city, "Index vs. Standardized Residuals"))
}
```

Function definitions to print the well-formatted intervals.

```
printPIInterval <- function(city, pair) {
  cat(paste(city, " 95% Prediction Interval: ", "(", round(pair[1], 3),
            ", ", round(pair[2], 3), ")\n", sep = ""))
}

printCIInterval <- function(city, pair) {
  cat(paste(city, " 95% Confidence Interval: ", "(", round(pair[1], 4),
            ", ", round(pair[2], 4), ")\n", sep = ""))
}
```

Function definition to calculate the 95% confidence intervals.

```
confidenceInterval95 <- function(model, housingData) {
  b1 <- model$coefficients[[2]]

  # At the time of writing this function, I did not recall that the summary
  # statistics for a linear model outputs the standard error for the slope
  # parameter... I suppose this could be simpler!

  std.err.b1 <- sqrt(summary(model)$sigma^2
                     /((nrow(housingData) - 1) * sd(log(housingData$sales))^2))
  t.value <- qt(0.975, nrow(housingData) - 2)

  upperBound <- b1 - (std.err.b1 * t.value)
  lowerBound <- b1 + (std.err.b1 * t.value)
  return(c(upperBound, lowerBound))
}
```

Function definition to calculate the 95% prediction intervals.

```
predictionInterval95 <- function(model, housingData, value) {
  prediction <- predict(model, newdata = data.frame(sales = value))[[1]]
  mse <- summary(model)$sigma^2
  variance <- sd(log(housingData$sales))^2
  size <- nrow(housingData)
  mean <- mean(log(housingData$sales))
  t.value <- qt(0.975, size - 2)

  se.pred <- sqrt(mse * (1 + (1/size) + ((prediction - mean)^2/((size - 1) * variance))))

  upperBound <- prediction + (se.pred * t.value)
  lowerBound <- prediction - (se.pred * t.value)
  return(c(exp(lowerBound), exp(upperBound)))
}
```