Anomaly Detection

Taylor Shrode

December 7, 2020

Introduction

An outlier (anomaly) is an object that deviates considerably from normal objects (World Class FTE Week 7). Outliers can be caused by data entry errors (human errors), measurement errors (instrument errors), experimental errors (data extraction or experiment planning/executing errors), data processing errors (data manipulation or dataset unintended mutations), natural errors (not an error, novelties in data), or can be intentional (dummy outliers made to test detection methods) (Santoyo, 2017).

Outliers and noise data are not the same. Noise is random error in a measured variable and should be removed, whereas, outliers are interesting (World Class FTE Week 7). Applications such as fraud detection, unusual patient symptoms, intrusion detection for cyber-security, and military surveillance for enemy activities utilize anomaly detections (World Class FTE Week 7).

Generally, there are three types of outliers: global, contextual, and collective outliers (World Class FTE Week 7). An individual object the deviates significantly from the rest of the dataset or not in the boundary of the normal data points is known as a *global outlier (point anomaly)*. An object that has an unusual pattern or deviates significantly based on a specific context or condition is known as a *contextual outlier (conditional outlier)*. There are two attributes associated with contextual outliers (World Class FTE Week 7):

- Contextual attributes: defines context such as time and location.
- Behavioral attributes: defines characteristics of the object for outlier evaluation such as temperature.

Finally, *collective anomalies* usually occur in datasets where data instance is related. "An individual data instance may not be anomaly by itself but when it occurs together as a collection, it becomes anomaly (World Class FTE Week 7)."

We will be using a dataset that contains the count of earthquakes of magnitude 7 or greater that occurred in each year from 1900-1998. There are 99 observations with 2 numerical variables: **year** and **earthquakes**. Using this dataset, we will be exploring various outlier detection techniques and various outlier detection tests to identify outliers in this dataset. The techniques we will be using include:

1. Visualization using Box-plots
2. Percentiles
3. Grubb's Test

4. Chi-Square Test
5. Autoencoders using H2O
6. Detect Anomalies using AnomalyDetection

More information for each technique will be provided later. Now, we need to load the necessary libraries and our dataset.

Load Libraries and dataset

The libraries needed for this assignment can be found below.

```
library(DataExplorer)
library(ggplot2)
library(dplyr)
library(outliers)
library(car)
library(AnomalyDetection)
library(h2o)
```

The **DataExplorer** package is loaded to perform exploratory data analysis and the **ggplot2** package is loaded to create various plots. The **dplyr** package is loaded for any general data wrangling needs. The **outliers** package is used to apply the Grubb's test and the Chi-Square test. The **car** package is loaded so we can test our data for normality. The **AnomalyDetection** package is loaded to detect anomalies from a statistical standpoint, in the presence of seasonality and an underlying trend (Twitter/anomalydetection, 2020). Note: to install the **AnomalyDetection** package, we need to use the **devtools** package and then use the **install_github("twitter/AnomalyDetection")** command. Finally, the **h2o** package is loaded to detect anomalies using autoencoders and deep learning.

To load our data, we first need to define the path to our file, and then we can load the data.

```
earthquake <- read.csv(file_path)
head(earthquake)
```

```
##   year earthquakes
## 1 1900          13
## 2 1901          14
## 3 1902           8
## 4 1903          10
## 5 1904          16
## 6 1905          26
```

Now, we can explore our data.

Exploratory Data Analysis

First, we will view the structure and a summary of our data.

```
str(earthquake)
```

```
## 'data.frame':    99 obs. of  2 variables:
##  $ year       : int  1900 1901 1902 1903 1904 1905 1906 1907 1908 1909 ...
##  $ earthquakes: int  13 14 8 10 16 26 32 27 18 32 ...
```

```
summary(earthquake)
```

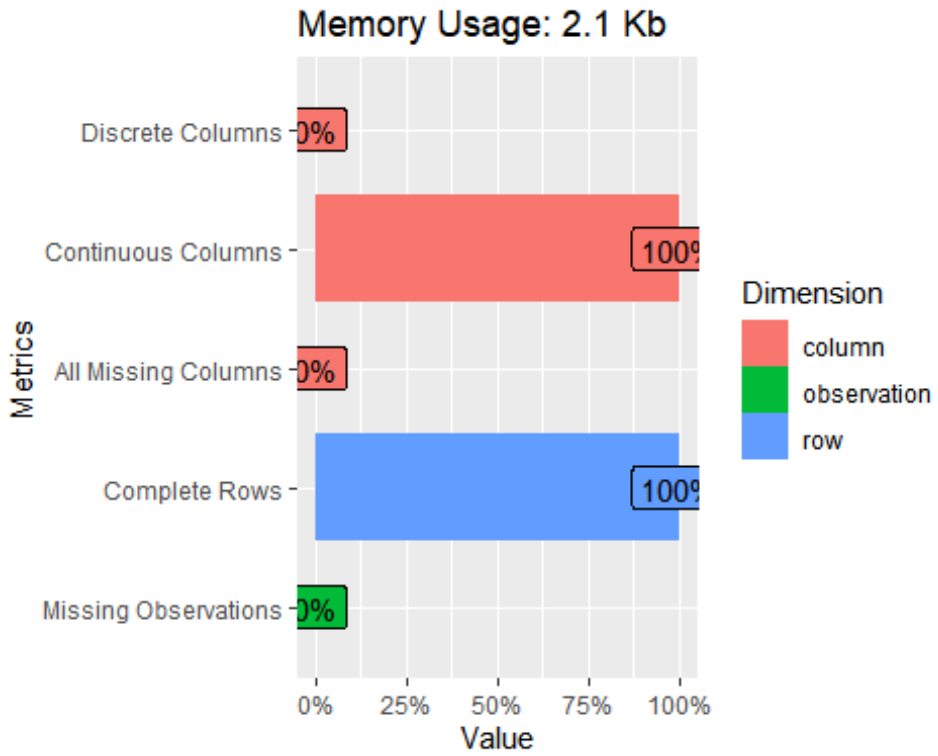
```
##       year       earthquakes
##  Min.   :1900   Min.   : 6.00
##  1st Qu.:1924   1st Qu.:15.00
##  Median :1949   Median :20.00
##  Mean   :1949   Mean   :20.02
##  3rd Qu.:1974   3rd Qu.:24.00
##  Max.   :1998   Max.   :41.00
```

From our output above, we can confirm that our dataset contains 99 observations of 2 integer variables (**year** and **earthquakes**). We can also confirm that the **year** attribute ranges from 1900 to 1998, while the **earthquakes** attribute ranges from 6 to 41. Now, we need to confirm that our dataset does not contain any null values.

```
introduce(earthquake)
```

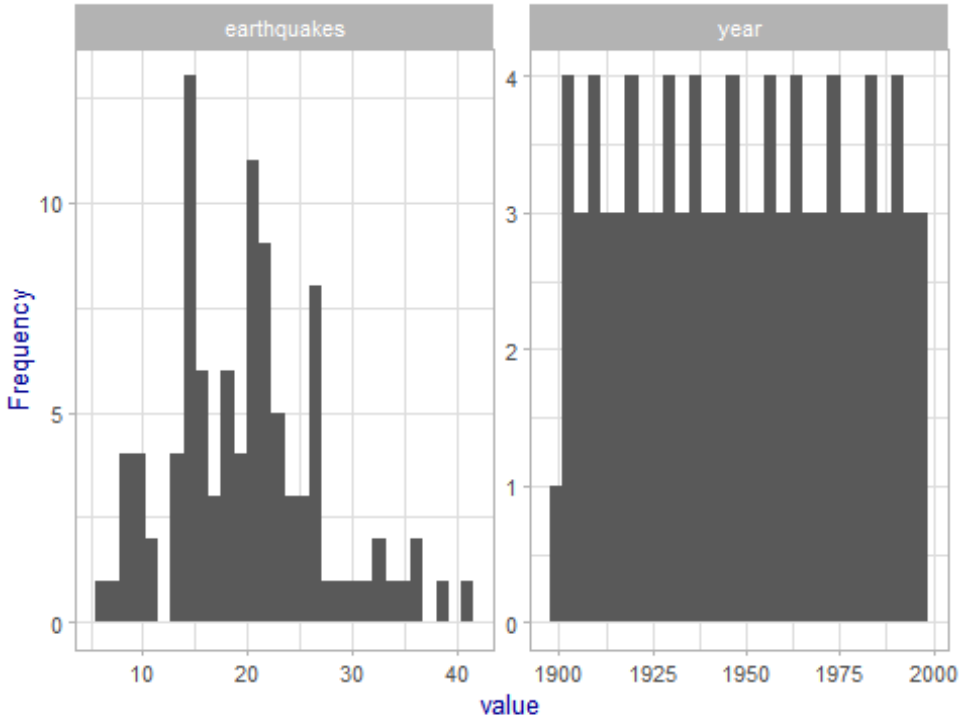
```
##   rows columns discrete_columns continuous_columns all_missing_columns
## 1   99       2                0                   2                   0
##   total_missing_values complete_rows total_observations memory_usage
## 1                    0            99                198         2152
```

```
plot_intro(earthquake)
```



The output above confirms that our data does not contain any null values. Next, we will view the distribution of our continuous features.

```
plot_histogram(earthquake, ggtheme = theme_light(base_size = 10),
theme_config = list("text" = element_text(color = "darkblue"))) #plot
continuous features
```



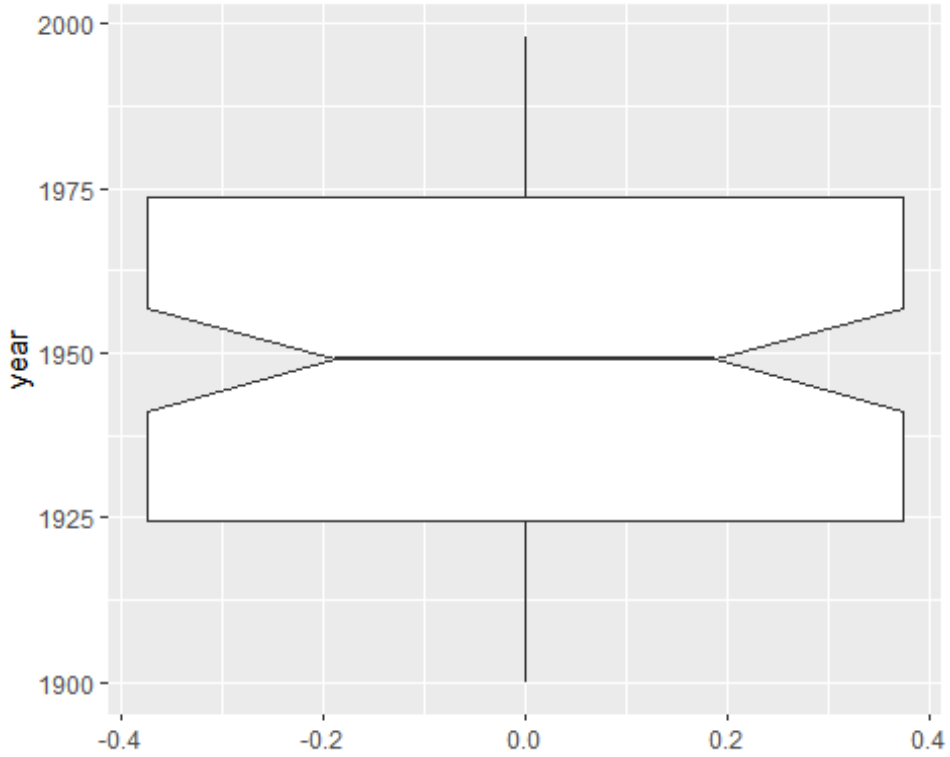
From the plots above, there does not seem to be any obvious outliers in the **year** column, but there may be outliers in the **earthquakes** column for the values greater than ~35.

Detecting Outliers using Box-plots and Percentiles

Box-plots are useful when detecting potential outliers. A *box-plot* helps visualize a quantitative variable by displaying the minimum, median, first and third quartiles and maximum (Outliers detection in R, n.d.). Observations are classified as potential outliers by using the *interquartile range (IQR)* criterion. This means that all observations above $q_{0.75} + 1.5 * IQR$ or below $q_{0.25} - 1.5 * IQR$ are considered potential outliers, where $q_{0.25}, q_{0.75}$ corresponds to the first and third quartile, respectively and IQR is the difference between the third and first quartile (Outliers detection in R, n.d.). The observations considered potential outliers using the IQR criterion are displayed as points.

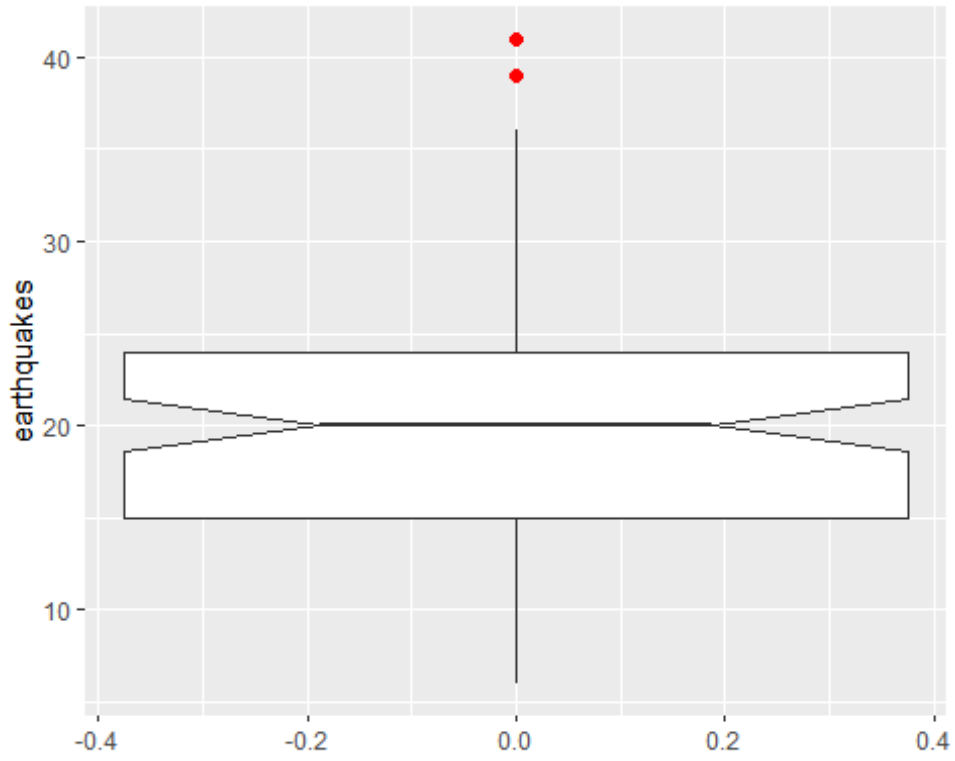
First, we will create a box-plot for **year**.

```
ggplot(earthquake, aes(y=year)) +
geom_boxplot(outlier.colour="red", outlier.shape=16, outlier.size=1,
notch=TRUE)
```



There does not seem to be any potential outliers in the **year** column. Now, we create a box-plot for the **earthquakes** column.

```
ggplot(earthquake, aes(y=earthquakes)) +
  geom_boxplot(outlier.colour="red", outlier.shape=16, outlier.size=2,
notch=TRUE)
```



From the box-plot above, there seems to be two potential outliers. To extract the values of the potential outliers, we can use the **boxplot.stats()$out** function (Outliers detection in R, n.d.).

```
boxplot.stats(earthquake$earthquakes)$out
```

```
## [1] 41 39
```

The two values considered as potential outliers are 41 and 39. Recall, the histogram above indicated that potential outliers could be values greater than ~35.

Using the **which()** function, we can extract the row numbers corresponding to the outliers (Outliers detection in R, n.d.).

```
rows <- which(earthquake$earthquakes %in%
c(boxplot.stats(earthquake$earthquakes)$out))
rows
```

```
## [1] 44 51
```

Now, we can print the variables for these outliers.

```
earthquake[rows,]
```

```
##    year earthquakes
## 44 1943          41
## 51 1950          39
```

Another method to detect potential outliers is the *percentiles* method, where all observations that lie outside the interval formed by the 2.5 and 97.5 percentiles are considered potential outliers (Outliers detection in R, n.d.). The percentiles 1 and 99, or the 5 and 95 can also be used to construct the interval.

First, we can define the lower and upper percentiles for the **year** attribute by using the **quantile()** function and utilizing the 2.5 and 97.5 percentiles (Outliers detection in R, n.d.).

```
lower_bound_year <- quantile(earthquake$year, 0.025)
lower_bound_year

##    2.5%
## 1902.45

upper_bound_year <- quantile(earthquake$year, 0.975)
upper_bound_year

##   97.5%
## 1995.55
```

According to this method, any observation greater than 1995.55 and below 1902.45 are considered potential outliers. Using the **which()** function, we can extract the row numbers for the potential outliers and then print their values (Outliers detection in R, n.d.).

```
outlier_year <- which(earthquake$year < lower_bound_year | earthquake$year  >
upper_bound_year) #gives row numbers
earthquake[outlier_year,]

##    year earthquakes
## 1  1900          13
## 2  1901          14
## 3  1902           8
## 97 1996          22
## 98 1997          20
## 99 1998          16
```

From the output above, the years 1900, 1901, 1902, 1996, 1997, and 1998 are considered potential outliers. We can use a similar process to extract potential outliers for the **earthquakes** column.

```
outlier_earthquakes <- which(earthquake$earthquakes <
quantile(earthquake$earthquakes, 0.025) | earthquake$earthquakes  >
quantile(earthquake$earthquakes, 0.975)) #gives row numbers
earthquake[outlier_earthquakes, ]

##    year earthquakes
## 44 1943          41
## 51 1950          39
## 87 1986           6
## 90 1989           7
```

From this method, we have identified a total of 10 potential outliers. To reduce this number, we can set the percentiles to 1 and 99 (Outliers detection in R, n.d.). First, we look at **year**.

```
outlier_year2 <- which(earthquake$year < quantile(earthquake$year, 0.01) |
earthquake$year > quantile(earthquake$year, 0.99))
earthquake[outlier_year2, ]

##    year earthquakes
## 1  1900          13
## 99 1998          16
```

Now, we can use similar steps for **earthquakes**.

```
outlier_earthquakes2 <- which(earthquake$earthquakes <
quantile(earthquake$earthquakes, 0.01) | earthquake$earthquakes  >
quantile(earthquake$earthquakes, 0.99))
earthquake[outlier_earthquakes2, ]

##    year earthquakes
## 44 1943          41
## 87 1986           6
```

Setting the percentiles to 1 and 99 give similar potential outliers that IQR criterion did for **earthquakes**. There were an additional two potential outliers found for the **year** column that the IQR criterion found.

Detecting Outliers using Grubb's Test and Chi-Squared Test

There are various statistical tests that can be used to identify the presence of potential outliers in a dataset. The first test we will use is Grubb's Test. The *Grubb's Test* allows us to detect whether the highest or lowest value in a dataset is an outlier (Outliers detection in R, n.d.). This test detects one outlier at a time, the highest or lowest values, so the null hypotheses are as follows (Outliers detection in R, n.d.):

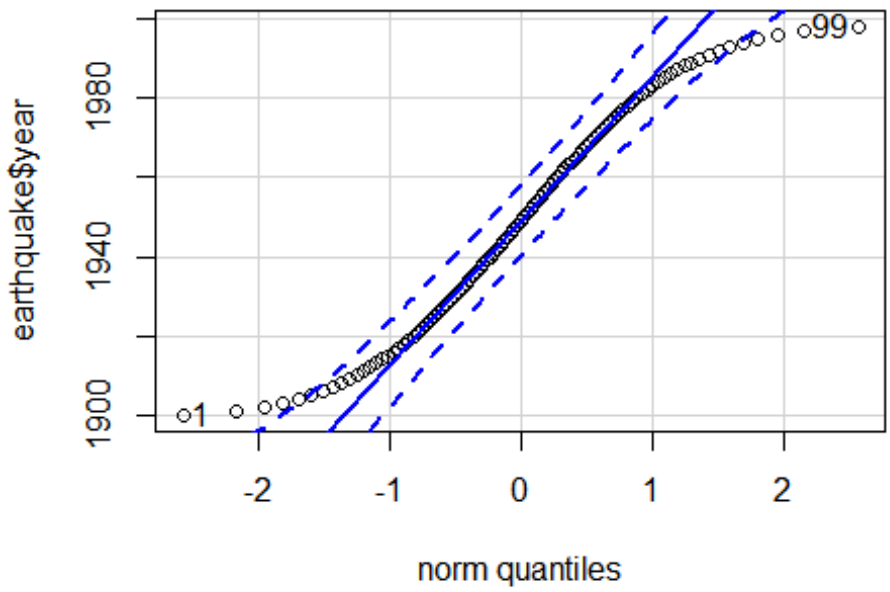
1. To test the **lowest** value:
   - H_0 The lowest value is **not** an outlier
   - H_A The lowest value is an outlier
2. To test the **highest** value
   - H_0 The highest value is **not** an outlier
   - H_A The highest value is an outlier

As with any statistical test, if the p-value is less than the level of significance (generally $\alpha = 0.05$), then the null hypothesis is rejected and we can conclude that the highest/lowest value is an outlier (Outliers detection in R, n.d.).

The Grubb's test is approriate when the data is normally distributed (without any outliers). To test for normality, we can use histograms, density plots, QQ-plots, or a normality test.

Below, we will use a QQ-plot, which allow us to visually evaluate the normality assumption (Does my data follow a normal distribution?, n.d.). We will also use the Shapiro-Wilk's test, which formally tests whether that data follows a normal distribution (Does my data follow a normal distribution?, n.d.). First, we will test our **year** column using a QQ-plot and the Shapiro-Wilks test.

```
qqPlot(earthquake$year)
```




```
## [1]  1 99
```

While many of the points are close to the reference line and within the confidence bands (other than the outliers), the normality assumption cannot be considered as met. We also see that the QQ-plot has identified two outliers: the first and last row. Now, we will use the Shapiro-Wilk test, where the hypotheses go as follows (Does my data follow a normal distribution?, n.d.):

- H_0 The data follow a normal distribution
- H_A The data do not follow a normal distribution

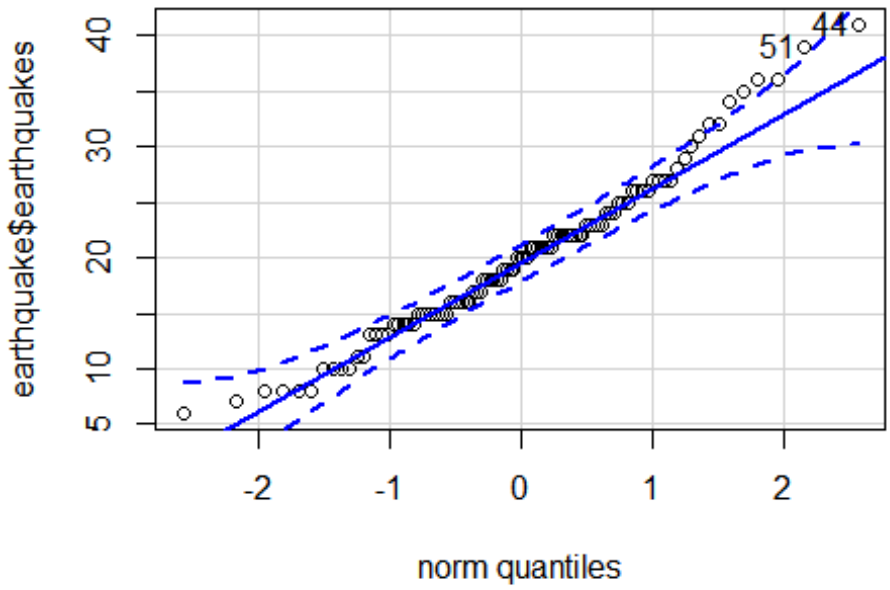
```
shapiro.test(earthquake$year)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  earthquake$year
## W = 0.95473, p-value = 0.001838
```

From the Shapiro-Wilk test, we can reject our null hypothesis and confirm that our data does not follow a normal distribution. It should be noted that "normality tests are often considered as too conservative in the sense that for large sample sizes ($n > 50$), a small deviation from the normality may cause the normality condition to be violated (Does my data follow a normal distribution?, n.d.)." In other words, as the number of observations increases, the Shapiro-Wilk test becomes very sensitive even to a small deviation from normality (Does my data follow a normal distribution?, n.d.).

Now, we look at the **earthquakes** column.

```
qqPlot(earthquake$earthquakes)
```




```
## [1] 44 51

shapiro.test(earthquake$earthquakes)

##
##  Shapiro-Wilk normality test
##
## data:  earthquake$earthquakes
## W = 0.97538, p-value = 0.0601
```

From the outputs above, we can confirm that our **earthquakes** column data follows a normal distribution. Also, from the plot above, there are two potential outliers in the row numbers 44 and 51. To use the Grubbs test, we will assume that both of our features follow a normal distribution. First, we will test the highest value in the **earthquakes** column is an

outlier. We use **type = 10** to test if the maximum value is an outlier (How to perform grubbs' test in r, 2020).

```
grubbs.test(earthquake$earthquakes, type = 10)
```

```
##
##  Grubbs test for one outlier
##
## data:  earthquake$earthquakes
## G = 2.88849, U = 0.91399, p-value = 0.1594
## alternative hypothesis: highest value 41 is an outlier
```

Our p-value is greater than 0.05, so we can not reject the null hypothesis and can conclude that the highest value of 41 is an outlier. Now, we can test the lowest value by using the parameter **opposite = TRUE** (How to perform grubbs' test in r, 2020).

```
grubbs.test(earthquake$earthquakes, opposite = TRUE, type = 10)
```

```
##
##  Grubbs test for one outlier
##
## data:  earthquake$earthquakes
## G = 1.93030, U = 0.96159, p-value = 1
## alternative hypothesis: lowest value 6 is an outlier
```

Our p-value is greater than 0.05, so we can not reject the null hypothesis and can conclude that the lowest value of 6 is an outlier. To test two high values, we can sample our data and use **type = 20**.

```
sample.quake <- earthquake[sample(nrow(earthquake), 30),]
grubbs.test(sample.quake$earthquakes, type = 20)
```

```
##
##  Grubbs test for two outliers
##
## data:  sample.quake$earthquakes
## U = 0.53365, p-value = 0.01171
## alternative hypothesis: highest values 39 , 41 are outliers
```

From the output above, the two highest values are outliers. To test if both minimum and maximum values are outliers, we can use **type = 11**.

```
grubbs.test(earthquake$year, type = 11)
```

```
##
##  Grubbs test for two opposite outliers
##
## data:  earthquake$year
## G = 3.41192, U = 0.94061, p-value = 1
## alternative hypothesis: 1900 and 1998 are outliers
```

For the **year** column, the values 1900 and 1998 are outliers. From the Grubb's tests above, we have found that the values 6, 41 (**earthquakes**) and 1900, and 1998 (**year**) are potential outliers.

The next test we will use to detect potential outliers is the Chi-Squared test for outliers. The *Chi-Squared* test for outliers performs a chi-squared test to detect one outlier in a vector (R: Chi-squared test for outlier, n.d.). The hypotheses for these test are as follows:

1. To test the **lowest** value:
   - H_0 The lowest value is **not** an outlier
   - H_A The lowest value is an outlier
2. To test the **highest** value
   - H_0 The highest value is **not** an outlier
   - H_A The highest value is an outlier

To perform this test, we can use the **chisq.out.test()** function. First, we will test the highest and lowest values in the **year** column.

```
chisq.out.test(earthquake$year)
```

```
##
##  chi-squared test for outlier
##
## data:  earthquake$year
## X-squared = 2.9103, p-value = 0.08802
## alternative hypothesis: highest value 1998 is an outlier
```

```
chisq.out.test(earthquake$year, opposite = TRUE)
```

```
##
##  chi-squared test for outlier
##
## data:  earthquake$year
## X-squared = 2.9103, p-value = 0.08802
## alternative hypothesis: lowest value 1900 is an outlier
```

The p-value for the highest value is 0.08802, which is greater than 0.05. Thus, we can not reject the null hypothesis and conclude that 1998 is not an outlier. The p-value for the lowest value is 0.08802, which is greater than 0.05. Thus, we can not reject the null hypothesis and conclude that the value 1900 is not an outlier.

Now, we can test the **earthquakes** column.

```
chisq.out.test(earthquake$earthquakes)
```

```
##
##  chi-squared test for outlier
##
## data:  earthquake$earthquakes
```

```
## X-squared = 8.3434, p-value = 0.003871
## alternative hypothesis: highest value 41 is an outlier

chisq.out.test(earthquake$earthquakes, opposite = TRUE)

##
##  chi-squared test for outlier
##
## data:  earthquake$earthquakes
## X-squared = 3.726, p-value = 0.05357
## alternative hypothesis: lowest value 6 is an outlier
```

The p-value for the highest value is 0.003871, which is less than 0.05. Thus, we can reject the null hypothesis and conclude that 41 is an outlier. The p-value for the lowest value is 0.05357, which is greater than 0.05. Thus, we can not reject the null hypothesis and conclude that the value 6 is not an outlier.

This test is often not recommended for routine use, because several more powerful tests are implemented, such as the Dixon test for outliers (R: Chi-squared test for outlier, n.d.).

Detecting Outliers Using Autoencoders

"Autoencoders is an unsupervised version of neural network that is used for data encoding and is mainly used to learn the representation of data that can be used for dimensionality reduction by training network to ignore noise (Nagdev, 2020)." Anomaly detection depends on unsupervised techniques since we tend to determine the anomalies (unknown) from the known data (Anomaly detection using h2o deep learning, n.d.).

We can use the **h2o.deeplearning()** function and setting **autoencoder = TRUE** to train an autoencoder model. Then, the **h2o.anomaly()** function can be used to detect the anomalies in the dataset (Anomaly detection using h2o deep learning, n.d.). "This function reconstructs the original dataset using the model and calculates the Mean Squared Error(MSE) for each data point (Anomaly detection using h2o deep learning, n.d.)."

First, we need to create/start a connection with H2O.

```
h2o.init()

##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         1 hours 18 minutes
##     H2O cluster timezone:       America/Denver
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.32.0.1
##     H2O cluster version age:    1 month and 29 days
##     H2O cluster name:           H2O_started_from_R_07hoc_fzb156
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   3.94 GB
```

```
##     H2O cluster total cores:    8
##     H2O cluster allowed cores:  8
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Amazon S3, Algos, AutoML, Core V3,
TargetEncoder, Core V4
##     R Version:                  R version 4.0.3 (2020-10-10)
```

Now, we need to convert our dataframe to a H2O object and convert the **year** column to a factor (Nagdev, 2020).

```
h2o.no_progress()
earthquake_h2o <- as.h2o(earthquake)
earthquake_h2o$year <- as.factor(earthquake_h2o$year)
```

To build an anomaly detection model using H2O, we need to create a training and testing dataset (Nagdev, 2020). We can partition our data using the **h2o.splitFrame()** function (H2o. Splitframe function, n.d.).

```
splits <- h2o.splitFrame(earthquake_h2o, ratios = 0.75, seed = 789)

train <- splits[[1]]
dim(train)

## [1] 75  2

test <- splits[[2]]
dim(test)

## [1] 24  2
```

The **h20.deeplearning()** function fits H2O's deep learning model. This function has many parameters, but the ones we will be using include (H2o. Deeplearning, n.d.):

- x: A vector containing the names or indices of the predictor variables to use in building the model
- *training_frame*: Id of the training data frame
- *autoencoder*: Auto-Encoder. Defaults to FALSE
- *reproducible*: Force reproducibility on small data
- *seed*: Seed for random numbers
- *ignore_constant_cols*: Ignore constant columns
- *model_id*: Destination id for this model
- *activation*: Activation function. Must be one of: "Tanh", "TanhWithDropout", "Rectifier", "RectifierWithDropout", "Maxout", "MaxoutWithDropout"
- *hidden*: Hidden layer sizes (e.g. [100, 100]). Defaults to c(200, 200).

- *epochs*: How many times the dataset should be iterated (streamed), can be fractional. Defaults to 10.

```
h2o.no_progress()
anomaly_model <- h2o.deeplearning(x = "year",
                                  training_frame = train,
                                  autoencoder = TRUE,
                                  reproducible = TRUE,
                                  seed = 789,
                                  ignore_const_cols = FALSE,
                                  model_id = "Test01",
                                  activation ="Tanh",
                                  hidden = c(200, 200),
                                  epochs = 10
)
anomaly_model
```

```
## Model Details:
## ==============
##
## H2OAutoEncoderModel: deeplearning
## Model ID:  Test01
## Status of Neuron Layers: auto-encoder, gaussian distribution, Quadratic
loss, 80,500 weights/biases, 953.2 KB, 750 training samples, mini-batch size
1
##   layer units  type dropout       l1       l2 mean_rate rate_rms momentum
## 1     1   100 Input  0.00 %       NA       NA        NA       NA       NA
## 2     2   200  Tanh  0.00 % 0.000000 0.000000  0.257482 0.428653 0.000000
## 3     3   200  Tanh  0.00 % 0.000000 0.000000  0.052753 0.013264 0.000000
## 4     4   100  Tanh      NA 0.000000 0.000000  0.043166 0.022721 0.000000
##   mean_weight weight_rms mean_bias bias_rms
## 1          NA         NA        NA       NA
## 2    0.000087   0.082792 -0.000883 0.009908
## 3   -0.000116   0.070710  0.002311 0.017744
## 4    0.000583   0.077726 -0.010907 0.017493
##
##
## H2OAutoEncoderMetrics: deeplearning
## ** Reported on training data. **
##
## Training Set Metrics:
## =====================
##
## MSE: (Extract with `h2o.mse`) 0.002383745
## RMSE: (Extract with `h2o.rmse`) 0.04882361
```

Now, we can detect anomalies in an H2O dataset using the H2O deep learning model with autoencoding trained previously (Nagdev, 2020).

```
anomaly_score <- as.data.frame(h2o.anomaly(anomaly_model,
                                           train,
```

```
                                          per_feature = FALSE
))
anomaly_score$y = 0
head(anomaly_score)

##   Reconstruction.MSE y
## 1        0.001915021 0
## 2        0.002411092 0
## 3        0.002761087 0
## 4        0.001982898 0
## 5        0.002801945 0
## 6        0.002255096 0
```

Next, we can calculate the threshold value for the train anomaly scores (Nagdev, 2020). Various methods can be used to calculate the threshold such as calculating the quantiles, maximum, median, or minimum values. Here, we use the quantile with probability of 97.5% (Nagdev, 2020).

```
threshold = quantile(anomaly_score$Reconstruction.MSE, probs = 0.975) #
Calculate the threshold value for train anomaly scores
```

Now that we have the anomaly scores for the training dataset and its thresholds, we can predict the new anomaly scores for the test data (Nagdev, 2020). Then, we can plot the train data and test data to see how the test data differs from the train data.

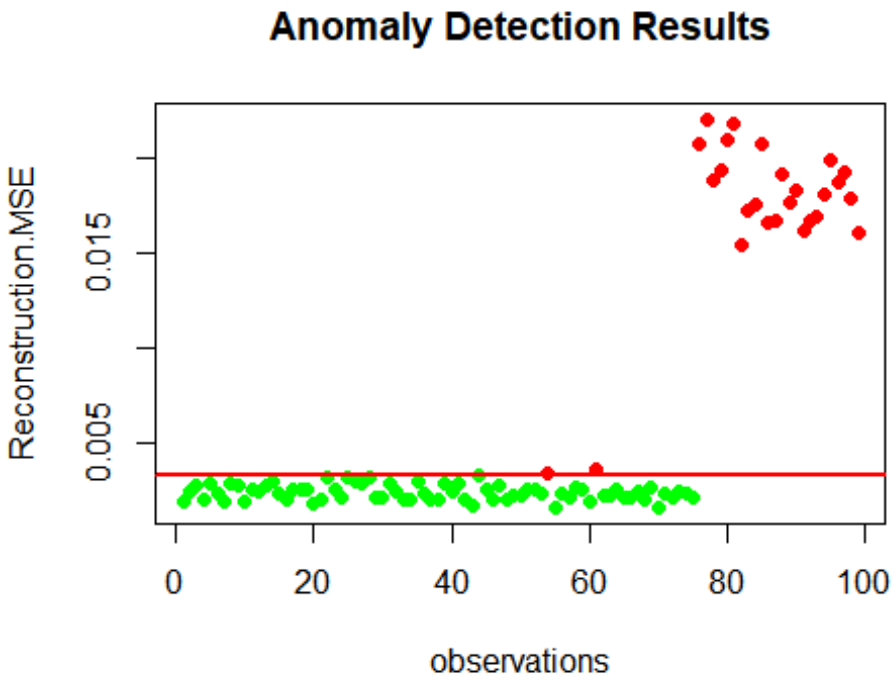
```
test_score <- as.data.frame(h2o.anomaly(anomaly_model,
                                        test,
                                        per_feature = FALSE
))

test_score$y = 1
results <- data.frame(rbind(anomaly_score,test_score), threshold)
head(results)

##   Reconstruction.MSE y   threshold
## 1        0.001915021 0 0.003266409
## 2        0.002411092 0 0.003266409
## 3        0.002761087 0 0.003266409
## 4        0.001982898 0 0.003266409
## 5        0.002801945 0 0.003266409
## 6        0.002255096 0 0.003266409
```

We can plot the results using the code below (Nagdev, 2020). Here, the x-axis is the observations and the y-axis is the anomaly score. Green points are the training data and the red are the testing data (Nagdev, 2020).

```
plot(results$Reconstruction.MSE, type = 'n', xlab='observations',
ylab='Reconstruction.MSE', main = "Anomaly Detection Results")
points(results$Reconstruction.MSE, pch=19,
col=ifelse(results$Reconstruction.MSE < threshold, "green", "red"))
abline(h=threshold, col='red', lwd=2)
```



Here, we see that only two testing data points lie below our threshold, which indicate potential outliers.

Detecting Outliers Using the AnomalyDetection Package

The **AnomalyDetection** package is used to detect anomalies in the presence of seasonality and an underlying trend (Twitter/anomalydetection, 2020). The package can also be used to detect anomalies in a vector of numerical values. The underlying algorithm (Seasonal Hybrid ESD (S-H-ESD)) builds upon the Generalized ESD test for detecting anomalies (Twitter/anomalydetection, 2020).

Before we apply the **AnomalyDetectionTs()** function, we need to convert our **year** column to time/date data as a POSIXct object by using the **as.POSIXct()** function.

```
earthquake$year <- as.POSIXct(as.character(earthquake$year), format = '%Y')
head(earthquake$year)
```

```
## [1] "1900-12-07 MST" "1901-12-07 MST" "1902-12-07 MST" "1903-12-07 MST"
## [5] "1904-12-07 MST" "1905-12-07 MST"
```

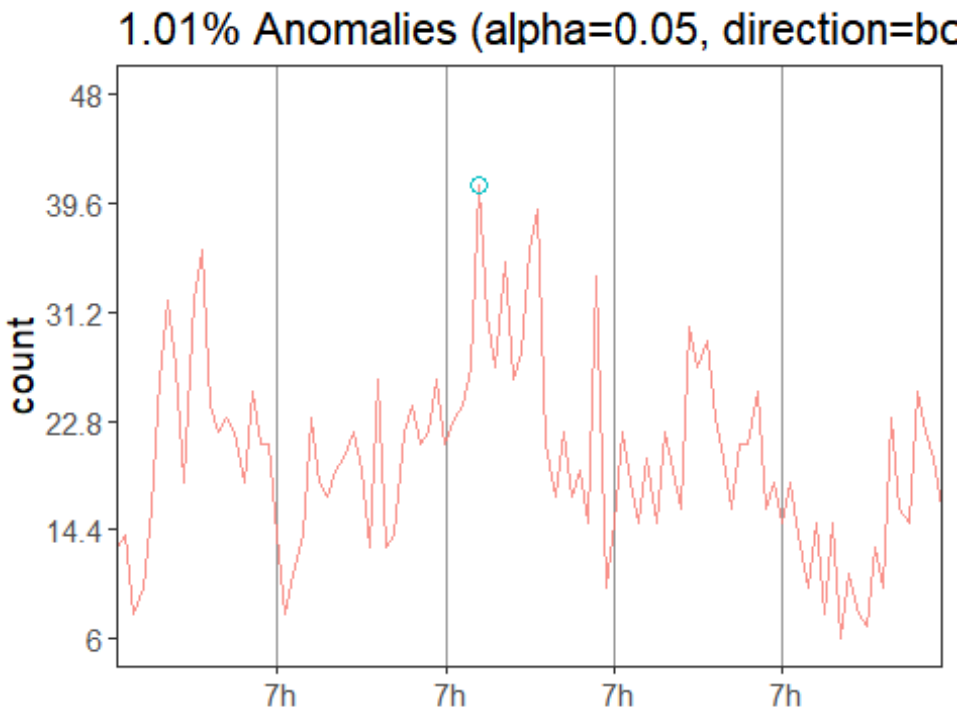
The function **AnomalyDetectionTs()** is called to detect one or more statistically significant anomalies in the input time series (Twitter/anomalydetection, 2020). The function **AnomalyDetectionVec()** is called to detect one or more statistically significant anomalies in a vector of observations. First, we will use the **AnomalyDetectionTs()** function. The parameters we will use include (Anomaly detection, n.d.):

- *max_anoms*: Maximum number of anomalies that S-H-ESD will detect as a percentage of the data
- *alpha*: The level of statistical significance with which to accept or reject anomalies
- *direction*: Directionality of the anomalies to be detected

```
anomaly_1 <- AnomalyDetectionTs(earthquake, max_anoms=0.02, alpha = 0.05,
direction='both', plot=TRUE)
anomaly_1$anoms

##    timestamp anoms
## 1 1943-12-07    41

anomaly_1$plot
```

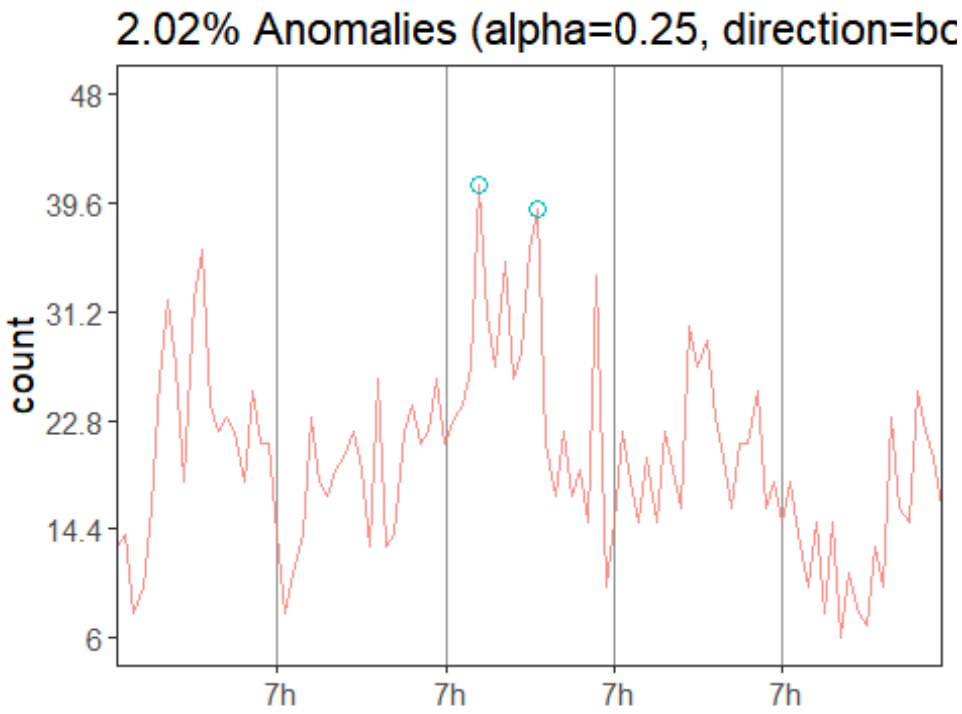


Using the parameters above, we have found one outlier. Using the **anomaly_1$anoms** command, we can determine that the outlier lies at the timestamp 1943, which has a value of 41 earthquakes. If we alter the **max_anoms** and **alpha** parameters, we can detect more potential outliers.

```
anomaly_2 <- AnomalyDetectionTs(earthquake, max_anoms=0.1, alpha = 0.25,
direction='both', plot=TRUE)
anomaly_2$anoms

##    timestamp anoms
## 1 1943-12-07    41
## 2 1950-12-07    39

anomaly_2$plot
```



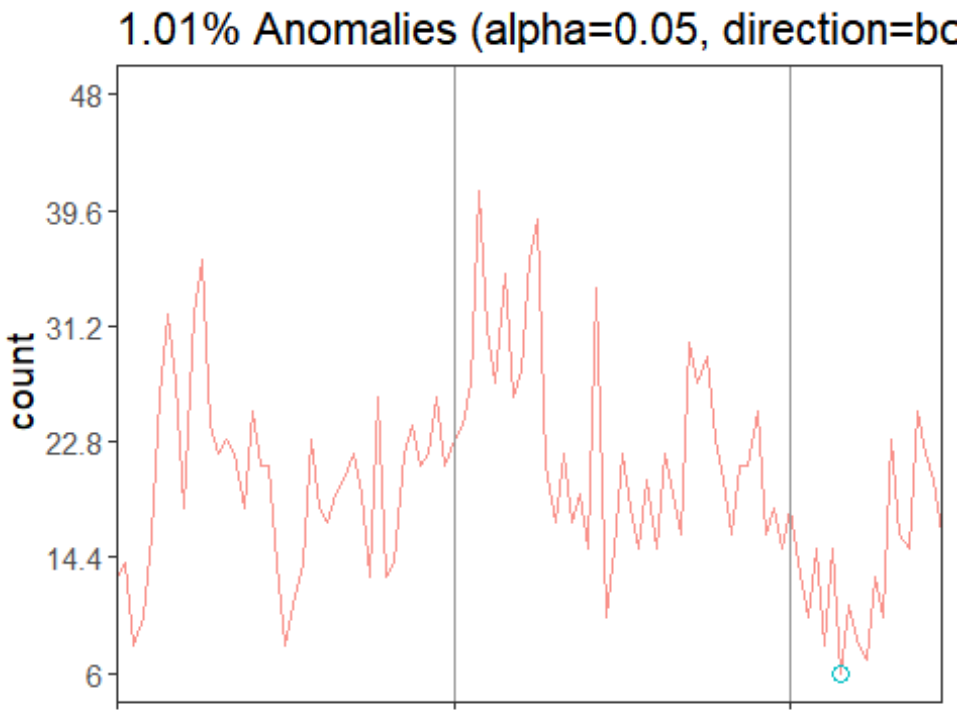
Notice, we have detected two potential outliers (39 and 41), which are the same outliers detected above with the other methods used. Now, we can use **AnomalyDetectionVec()** function with the following parameters (Anomalydetectionvec function, n.d.):

- *max_anoms*: Maximum number of anomalies that S-H-ESD will detect as a percentage of the data
- *alpha*: The level of statistical significance with which to accept or reject anomalies
- *direction*: Directionality of the anomalies to be detected
- *period*: Defines the number of observations in a single period, and used during seasonal decomposition
- *only_last*: Find and report anomalies only within the last period in the time series

```
anomaly_vec <- AnomalyDetectionVec(earthquake$earthquakes, max_anoms=0.02,
period=40, direction='both', only_last=FALSE, plot=TRUE)
anomaly_vec$anoms

##   index anoms
## 1    87     6

anomaly_vec$plot
```



The output above has indicated a single outlier located in row number 87, which has a value of 6 earthquakes.

Conclusion

For this assignment, we used various anomaly detection methods to detect potential outliers in our dataset. The dataset used contained information regarding the number of earthquakes of 7 magnitude or higher. We used visualization tools, outlier tests, and other outlier techniques to detect anomalies. From our box-plots, we found a total of two potential outliers for the **earthquakes** column. Using percentiles, we initially found a total of ten potential outliers. After adjusting the percentiles, we found a total of four potential outliers. Using the Grubb's test, we found five potential outliers and the Chi-Square test indicated that there was a total of one potential outlier. Autoencoders yielded two potential outliers. Finally, the **AnomalyDetection** package indicated that there were two potential outliers, while the **AnomalyDetectionVec()** function indicated that there was one potential outlier.

Overall, the potential outliers found across the various methods include:

- From the **year** column: 1900, 1901, 1902, 1943, 1950, 1986, 1996, 1997, 1998
- From the **earthquakes** column: 6, 7, 39, 41,

The outliers that appeared the most often, and are the most appropriate, include:

- From the **year** column: 1943, 1950

- From the **earthquakes** column: 6, 39, 41

It is most likely that the years 1943 and 1950 were likely anomalies due to the number of earthquakes.

Overall, the methods to detect potential outliers that were the most effective and appropriate for this dataset include visualization tools such as box-plots, Grubb's test, autoencoders, and the **AnomalyDetection** package.

Resources

Anomaly detection . (n.d.). Retrieved December 7, 2020, from https://rdrr.io/github/ivanliu1989/RQuant/man/AnomalyDetectionTs.html

Anomaly detection using h2o deep learning. (n.d.). Dzone.Com. Retrieved December 7, 2020, from https://dzone.com/articles/dive-deep-into-deep-learning-using-h2o-1

Anomalydetectionvec function. (n.d.). Retrieved December 7, 2020, from https://www.rdocumentation.org/packages/AnomalyDetection/versions/1.0/topics/AnomalyDetectionVec

Does my data follow a normal distribution? (n.d.). Stats and R. Retrieved December 7, 2020, from https://www.statsandr.com/blog/do-my-data-follow-a-normal-distribution-a-note-on-the-most-widely-used-distribution-and-how-to-test-for-normality-in-r/

H2o. Deeplearning. (n.d.). Retrieved December 8, 2020, from https://rdrr.io/cran/h2o/man/h2o.deeplearning.html

H2o. Splitframe function. (n.d.). Retrieved December 8, 2020, from https://www.rdocumentation.org/packages/h2o/versions/3.32.0.1/topics/h2o.splitFrame

How to perform grubbs' test in r. (2020, April 7). Statology. https://www.statology.org/grubbs-test-r/

Nagdev. (2020, March 5). Auto encoders for anomaly detection in predictive maintenance. R-Bloggers. https://www.r-bloggers.com/2020/03/auto-encoders-for-anomaly-detection-in-predictive-maintenance/

Outliers detection in R. (n.d.). Stats and R. Retrieved December 7, 2020, from https://www.statsandr.com/blog/outliers-detection-in-r/

R: Chi-squared test for outlier. (n.d.). Retrieved December 7, 2020, from http://finzi.psych.upenn.edu/R/library/outliers/html/chisq.out.test.html

Santoyo, S. (2017, November 24). A brief overview of outlier detection techniques. Medium. https://towardsdatascience.com/a-brief-overview-of-outlier-detection-techniques-1e0b2c19e561

Twitter/anomalydetection. (2020). [R]. Twitter. https://github.com/twitter/AnomalyDetection (Original work published 2014)

World Class From The Expert: Week 7