WestJet Summer 2018 Internship Flight Clustering Model Project Report

Written By Kofi Buahin – Revenue Analytics & Management Co-op Summer Student 2018

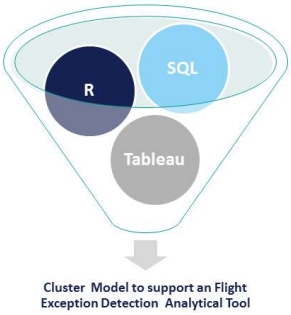
Project Led by Kofi Buahin & Yanik Lacroix Torrent

For further elaboration please don't hesitate to contact Kofi at kkbuahin@gmail.com

Contents

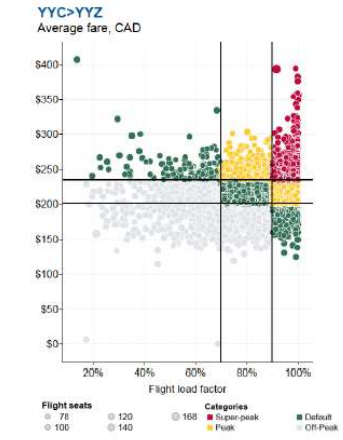
Objective & Introduction – Pre Steps:

Our goal with this project was to develop a flight clustering model that could fuel an alert system to detect flights which don't follow their expected pattern, track and visualize them, and create an avenue for the Revenue analysts to take action .

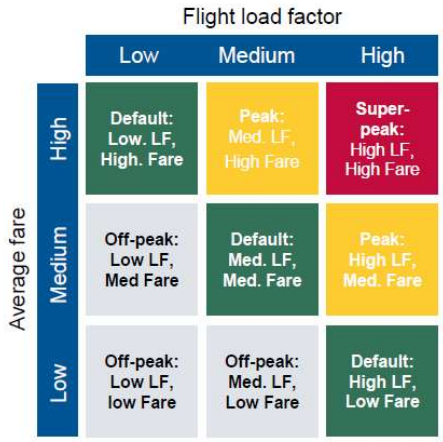


- Currently there is another company (Seabury) that implements the "Split History Optimization" cluster Model that carries out the clustering process for WestJet.
    - WestJet currently has a clustering system that serves to calculate split histories for use in PROS in order to provide analytical support to the analysts.
    - Flights are clustered into 4 different categories based on load factor and average fares:
        - Super-peak
        - Peak
        - Default
        - Off - Peak
    - The segmentation is based on a 70-90% LF threshold and average thresholds are dynamically adjusted based on a Market by Market basis. This can be seen shown in the image presented below:

Historical flights are evaluated



Outputs are grouped into four categories



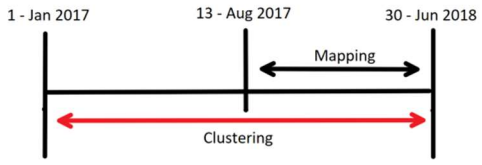
    - The current Seabury Method works because PROS requires simplicity however as far as we can see there is only a very high level of statistical theory implemented to cluster the flights and this project seeks to provide a more detailed and specific output.
- This project will look to work at a more granular level than the Seabury method and if executed properly we could take this new clustering methodology and utilize it in place of the Seabury Method.

Step 1: Data Extraction

- The first step of the process of clustering was determining the data which we wanted to collect in order to complete our clustering.
- SQL was the program used for Data Collection and we ran queries to obtain the appropriate data that would be used in the project.
- WestJet has data separated in different locations and we had to collect information from two different areas to conduct the clustering analysis.
    - **Inventory Leg Build(Inventory Fact)** → houses inventory data and facts about the flight
    - **Flown Revenue (Bookings Fact)** → contains only the bookings information and anything that pertains to price/revenue.
- We are taking data from a time frame that spans over the last year and half because it is the most relevant to our future flights and gives us ample amounts of data to establish clustering groups for all of our markets.
    - We are only taking data from the day of departure so that the NDO 0 (Capture date = FlightDate +1)
    - We also took all flight-level data network-wide from this time period.
    - The filter we used to collect this data can be seen presented below:

```
WHERE   FlightDate between '2017-01-01' AND '2018-06-30'
AND     CaptureDate = FlightDate+1
```

    - The screenshot presented below provides a breakdown of the usage of the data:



- This was also the time where we collected Revenue Data to give us even more options of Data to look at in the process of clustering. We were able to collect this data from the Access.Fact_FlownRevenue view in the DM_RevenueMgmt database in SQL.

Step 2: Data Selection

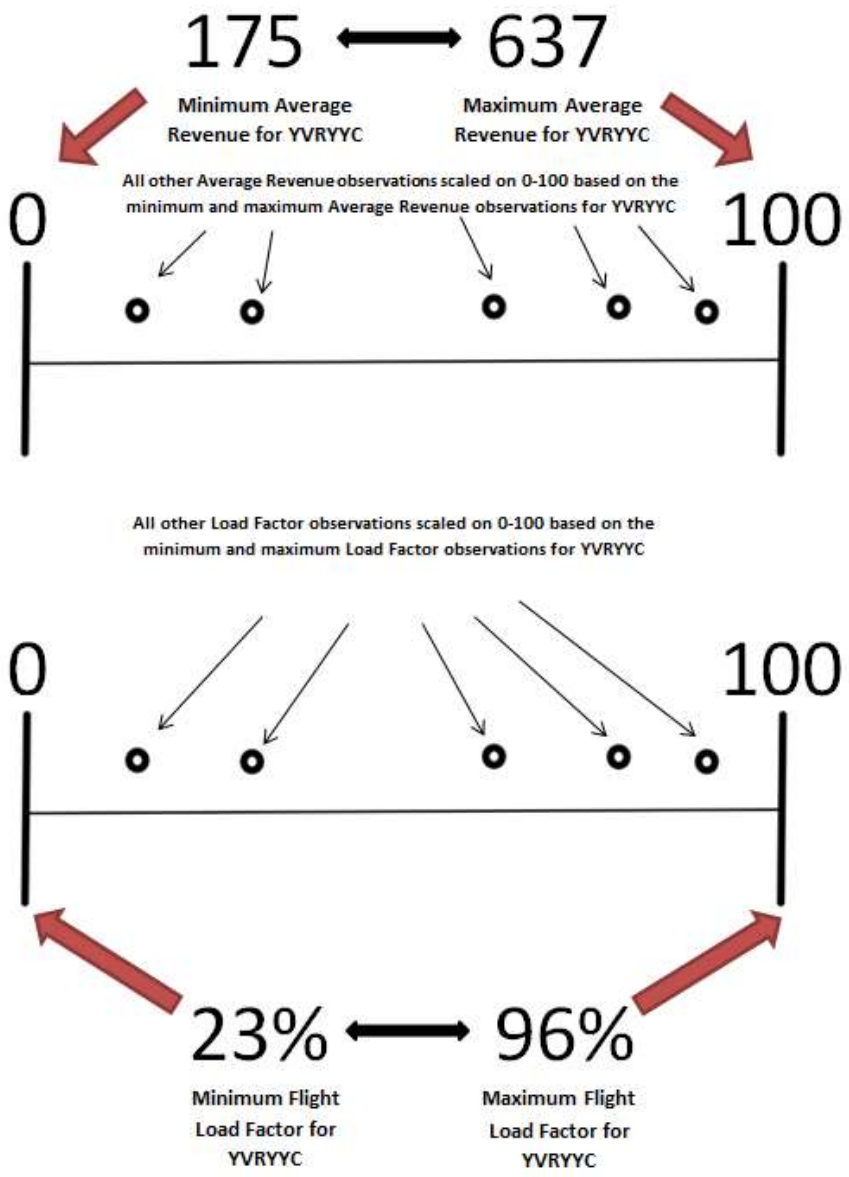
Obtaining Data

- From all the data fields collected we decided to select the following variables for our main database to be used in R. We decided they represented the most relevant fields to include in our clustering analysis for the reasons listed below:
    - **Capture Date** – This was important to look at because this represents a snapshot of any given piece of data that we are looking at for any given time.
    - **Flight Date** – This represents a key variable for flight identification and YoY mapping purposes.

- **Flight Number** – This represents a key variable for flight identification and YoY mapping purposes.
- **Departure Time** – Played a key role in the YoY mapping process for the flights we will be comparing.
- **Arrival Time** – Played a key role in the YoY mapping process for the flights we will be comparing.
- **Directional Market** – The directional market represents a pivotal identifying variable in the Data and allowed us to track the data for different routes.
- **Equipment Code** – Represents an identifier for the type of plane being flown.
- **Capacity** - Shows the seat selling potential we have and was an important variable to pay attention to.
- **Lid** – A variable that gives us an idea on the actual physical capacity available and can be crucial in an Analyst's decision making process.
- **Total Sold** – presents a good indicator of whether we are on track with the number of seats sold.
- **Number of Days out (NDO)** – A calculated field that helps us to track how a flight is doing based on the number of days a flight is away from its departure.
- **RPM & ASM** – Represents 2 key variables to calculate load factor, one of the main KPIs we used.
- **Time band** – A Calculated Field that allowed us to aggregate and look at flights around a similar time frame.
- **Load Factor** – Calculated Field that represents a potential KPI to be used to cluster the flights together.
- **Average Revenue** – A KPI that was a candidate to be one of the variables used to cluster our flights.
- **Total Revenue** – Required to calculate the Average Revenue
- **Passengers Flown** – Required to calculate the Average Revenue

Standardizing Data

- There is a great importance of scale when considering distance between two observations with cluster analysis.
- With Cluster Analysis, the scale of the different variables used to group observations plays a very important role for comparability reasons. If the scales of the variables which we are using to cluster are very different then that can skew the Euclidean distances between points and therefore have a misleading effect on the nature of our clusters.
- Our Solution to this was to standardize values for both Load Factor and Average Revenue using a constant mathematical function. This function allowed us to convert all of our data onto a scale of 0 – 100 relative to the observations within the variable we are trying to standardize.
- Our goal was to have something like what is being presented in the diagram below:



- Formula is displayed below in natural terms and as it was created in R:

$$New\ Value = (x - \text{Min}) * \frac{100}{Max - Min}$$

X = Observation in sequence being converted to standard 0 - 100 scale
Min = Lowest value on current scale
Max = Highest value on current scale

Example: Convert a flights LF of 66% in range 35-100% to a new value of 47.69 on scale 0-100

X = 66
Min = 35
Max = 100

New value = (66-35) * 100/ (100 – 35)
New value = 31 * 1.538461538
New value = 47.69

RELEVANT CODE SNIPPETS

```
27 conversion <- function(arg1, maxx, minn){
28   term1 = arg1 - minn
29   term2 = maxx - minn
30   round(term1 * (100/term2),2)
31 }
32
```

Selecting Final Variables for Clustering

- By standardizing the Load Factor and Average Revenue, we create new variables that allow for a scenario where it is more appropriate to compare our two clustering variables.
- After this, we have to come to a decision regarding which variables to use to carry out our cluster analysis.
- Initially we wanted to utilize 3 different variables in order to create a 3-Dimensional Cluster Segmentation by (NDO_20LF (this represents a measure which records the highest NDO at which Load Factor reaches 20%), Standardized Average Revenue, and Standardized Load Factor)
    - The reason we wanted to add 3 variables was to capture different factors that contribute to the performance of a flight. We also wanted to identify the characteristics of a flight.
    - Specifically with NDO_20LF, our goal was to create a way to capture the booking window for each flight observation for each market. By establishing the maximum length of NDO at which a flight hits 20% Load factor, we can get some sort of indication about how early customers are actually making ticket purchases.
    - Although we felt it was very important to take into account the booking window, we discovered that implementing NDO_20LF was sub-optimal because the clusters we ran would only cluster according to the other two variables and ignored NDO_20LF.
    - We found this was the Case because NDO and Standardized Load Factor have a very strong positive correlation and therefore are essentially telling is the say.
        - For example, the higher the NDO, it is expected that the lower the Load factor will be in most cases due to the fact that traditionally, you expect the flight to fill up more over time.
- For this reason we decided to take only 2 variables, Standardized Average Revenue and Standardized Load Factor. In doing this we are able to create clusters with variables that will have an effect on the assignment of observations to clusters.

Step 3: Database Construction

- After coming to a conclusion regarding all the variables that should be included in the cluster analysis, we had to create a database to allow us to easily access all the data that we needed to conduct the clustering.
- This entire process was carried out using SQL. More specifically we wrote a query that would create a Sandbox. Table in SQL that we could directly read into R and manipulate with code. This is done for a few reasons:
    - For simplicity. This makes it easier to work with the data and more convenient to combine the clusters for all the different routes being examined.
    - By doing this, it also stops us from hard coding and R and subsequently makes the code easier to re-use.
- Snippets of the code for the Sandbox Table can be seen displayed below:

```
/*      This section deletes existing rows for the Capture Date                */
WHILE (1=1)
BEGIN
DELETE TOP (100000)
-- SELECT *
FROM    Sandbox.ClusteringFinalDataTable
IF @@ROWCOUNT = 0 BREAK
END


INSERT INTO Sandbox.ClusteringFinalDataTable ( CaptureDate, FlightDate, FlightNumber, DOW, [MonthName], DeptTime, ArriveTime, DirMkt,
            EquipmentCode, Capacity, Lid, TotalRevenue, LFpctage, NDO_20LF, MaxFare, MinFare, TimeBand, AvgFare, Pure_AvgFare, AvgRevenue )
SELECT
       FINAL.CaptureDate, FINAL.FlightDate, FINAL.FlightNumber, FINAL.DOW, FINAL.[MonthName], FINAL.DeptTime, FINAL.ArriveTime, FINAL.DirMkt,
       FINAL.EquipmentCode, FINAL.Capacity, FINAL.Lid, RevNum.TotalRevenue, FINAL.LFpctage, LF20.NDO_20LF,
       FINAL.MaxFare, FINAL.MinFare, FINAL.TimeBand, FINAL.AvgFare, FINAL.Pure_AvgFare,
       CASE WHEN RevNum.PassengersFlown = 0 THEN 0 ELSE RevNum.TotalRevenue / RevNum.PassengersFlown END as AvgRevenue
FROM    #ClusteringDataNDO0 FINAL
       JOIN    #DEP RevNum
       ON      FINAL.FlightDate = RevNum.FlightDate
       AND     FINAL.FlightNumber = RevNum.FlightNumber
       AND     FINAL.DirMkt = RevNum.DirMkt
       JOIN    #LoadFactorGreaterThan20 LF20
       ON      LF20.FlightDate = FINAL.FlightDate
       AND     LF20.FlightNumber = FINAL.FlightNumber
       AND     LF20.DirMkt = FINAL.DirMkt
```

Step 4: Outlier Detection

- This proved to be one of the most time consuming aspects of the project however it represented a key step. We had to remove outliers because these are points that represent flights that were performing outside the norm
    - Since we are working towards establishing clustering patterns, it would be inappropriate to take into account flights that deviate from the norm as they would not present an accurate representation of what is expected for certain markets.
- Outliers in our Flight and Revenue data here at WestJet can arise for a few reasons:
    - Data Issues (Inconsistences & Inaccuracies): There were often scenarios that we found that flight data was inaccurate or didn't line up.
        - For example there were a few situations where we found load factor was 0 % or greater than 100% which is clearly inaccurate.
    - Special Events: These are one-off occurrences that cause a spike in our performance metrics that will probably not occur again.

        - For example the September 2017 Hurricanes causing flights going downwards towards affected cities to be empty and flights coming back up to be full. This would cause performances spikes in our metrics that we wouldn't want to consider in our definition of a cluster pattern.
- We decided to utilize two different methodologies to remove outliers for our 2 differing clustering variables.
    - This was done because the observations for these two different variables have differing characteristics and therefore outliers must be defined in unique ways that account for these differences.
- The outlier removal process was carried out in R for the most part, using custom written functions to eliminate unwanted observations from the data set we established in SQL.

Removing Outliers for Load Factor

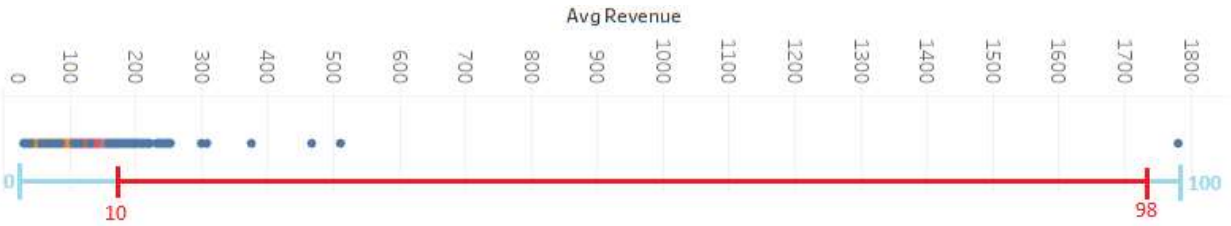
- The logic for removing Outliers in Load Factor was to set a custom upper and lower threshold for Load Factor and any observations with a Load Factor outside both of these defined thresholds will be eliminated.
- We set two different lower thresholds:
    - For routes with an average load factor < 50% ; the lower threshold for outliers is 5%
    - For all other routes the lower threshold for outliers is 10%
- We set a constant upper threshold of 100% for all routes and therefore any observations with an average load factor of 100% would be highlighted as outliers and removed from the working Data Set.
    - It's important to note that this is a very conservative method of detecting outliers and only works to remove very obvious outliers for load factor (Load Factor values lower than 10% and greater than 100% are uncommon)
        - We decided to remove outliers like this for Load Factor for simplicity and due to time constraints we faced.
        - In potential future developments of this project, we can explore filtering these outliers in a more elaborate manner.
- As a summary…
    - Routes with an average load factor < 50%: Any observations with load factor outside the range of 5 – 100 % will be removed
    - Routes with an average load factor > 50%: Any observations with load factor outside the range of 10 – 100 % will be removed
- For the R code reference, please observe the screenshot presented below.

RELEVANT CODE SNIPPETS

```
remove_outliersLF <- function(df){
  nu_df <- df
  if (nrow(df)>35){
    if (mean(df$LFpctage)<50){
      nu_df <- filter(nu_df, nu_LF > 5)
      nu_df <- filter(nu_df, nu_LF <= 100)
    } else {
      nu_df <- filter(nu_df, nu_LF > 10)
      nu_df <- filter(nu_df, nu_LF <= 100)
    }
  } else{nu_df <- df}
}
```

Removing Outliers for Average Revenue

- Removing outliers for Average Revenue posed a greater challenge and we looked at a few methods of removing outliers.
- The first method we tried to implement was the **Interquartile Range(IQR)** method of removing outliers which has been outlined below:
    - Steps to IQR outlier detection rule:
        - Sort Average revenue in order of lowest to highest for the directional market in discussion.
        - Calculate **first quartile** (Q1), **third quartile** (Q3) and the **IQR** (IQR=Q3-Q1) of the Average revenue for that directional market.
        - Compute the upper threshold and lower threshold for outlier detection
            - Lower Threshold = Q1 – (1.5 * IQR)
            - Upper Threshold = Q3 + (1.5 * IQR)
            - Anything outside this range is an outlier.
        - Summary: Anything greater than the Upper Threshold or smaller than the Lower Threshold will be deemed an outlier with this rule.
    - We didn't select this method because it assumes that the flight revenue for each directional market will be normally distributed and this will not always be the case.
        - Because of this assumption, the IQR removed a lot of flight observation in our trial and error process.
- The second method we looked into for removing outliers was the method we used to determine outliers for Load Factor.
    - With this method we planned to use standardized average revenue as the outlier detecting variable set an upper threshold of 98 and a lower threshold of 10. Anything outside this range would be determined an outlier.
    - This method didn't work because our standardization method is sensitive to the largest observation of average revenue.
        - What is meant by this is that if the greatest value of average revenue for a given route is significantly larger than the other average revenue values for that route the standardization could be skewed and that would impact this outlier removal process.
    - You can see a visual example of this sensitivity in the image presented below that represents standardized average revenue values for YVRYLW:



  - As you can see in the image presented above a lot of the flight observations have standardized average revenue closer to the beginning of spectrum due to their great difference from the maximum observation from this market.
    - With the normal range method, a lot of the flight observations that aren't actually outliers would be defined as outliers.
- After a lot of testing we decided to implement the **Median Absolute Deviation (MAD) Method to determine outliers.**
- Reasons for Implementing MAD
  - It provides a better alternative to using the mean as an indicator of central tendency because…
    - The mean and standard deviation are heavily affected by outliers
    - The mean is unlikely to be well equipped to uncovering outliers in small sample sizes.
  - Median is insensitive to outliers and therefore this method provides a way to cautiously remove outliers.
  - The MAD methodology is also completely immune to sample size which makes it good for us because we have routes with differing frequencies.
- The Median absolute deviation essentially identifies takes the Median ± MAD * Multiplier (or Absolute Deviation) to create a range for suitable observations.
  - Anything that falls outside of this range is defined an outlier and can be treated as such within the Data Set.
- The formula for the MAD can be seen presented below:

$$MAD = b * M_i\left(\left|X_i - M_j(X_j)\right|\right)$$

$\boldsymbol{X_j}$ = n original observations

$\boldsymbol{M_i}$ = median of the series

b = constant that is linked to the assumptions of normality of the data

- Once you have calculated the MAD you then plug it into the formula presented below to create your range for outlier detection:

$$\boldsymbol{Lower\ Outlier\ Detection\ Threshold} = \boldsymbol{Median} - (\boldsymbol{MAD} * \boldsymbol{Multiplier})$$

$$\boldsymbol{Upper\ Outlier\ Detection\ Threshold} = \boldsymbol{Median} + (\boldsymbol{MAD} * \boldsymbol{Multiplier})$$

Example Series: 1, 2, 3, 4, 5; Example Multiplier = 2.5

$M_j = 3$

M_i ($|X_i - M_j (X_j)|$) = Median (| (1-3), (2-3), (3-3), (4-3), (5-3)|)

$$\mathbf{M_i(|X_i - M_j(X_j)|)} = \text{Median } (0, 1, 1, 2, 2) = 1$$

$$\mathbf{B = 1}$$

$$\mathbf{MAD = 1^* 1 = 1}$$

$$\mathbf{Lower\ Threshold} = 3 - (1^*2.5) = 0.5$$
$$\mathbf{Upper\ Threshold} = 3 + (1^*2.5) = 5.5$$

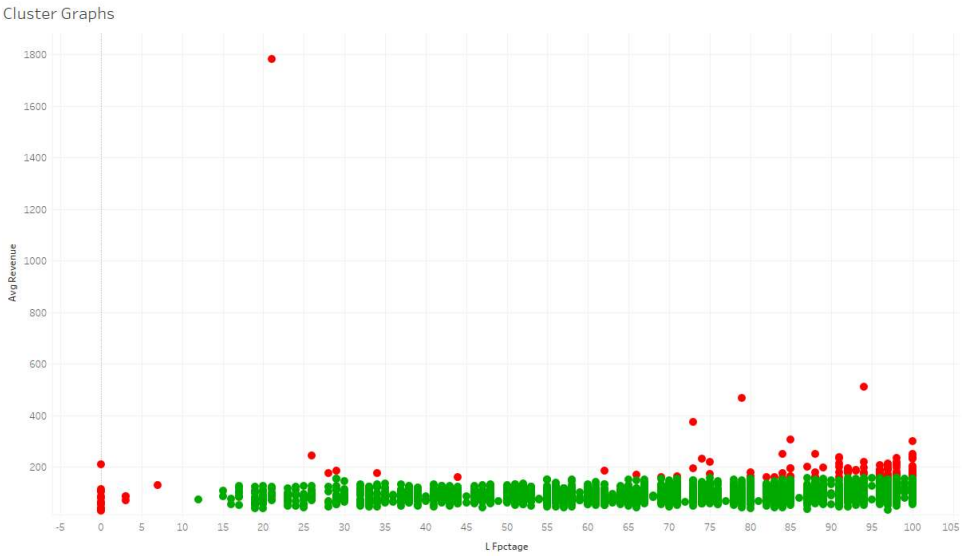
- The multiplier is set depending on how conservative or aggressive you would like to be when defining outliers.
    - The **higher** the multiplier the more conservative you are in outlier identification(the fewer outliers removed)
- To determine how conservative we wanted to be with our outlier removal, we tried working with a number of different multipliers. We came to a conclusion that we would have to implement different multipliers for different routes as it was impossible to set a constant multiplier across all routes that would remove an appropriate number of outliers.
- In order to do this, we implemented a trial and error based set of code in R that would continuously run the flights from each route through the MAD outlier detection methodology with different outliers and assign each route to a multiplier group based on whether the multiplier removed an acceptable number of observations as outliers.
    - After repeated trial an error, we reached a level where the outlier removal method would take less than 5% of flight observations for any given route. This created a situation where different routes would be assigned to different multiplier groups to fulfill this outlier detection level.
    - Once the **For Loop** implemented in R to assign each route to a multiplier group is completed, a final outlier removal is done for each assigned route using the appropriate multiplier it has been assigned to.

- After running through trial and error for different multiplier values, we established a range of 6 different multipliers to be applied to the 566 routes and a breakdown of this can be seen in the table presented below:

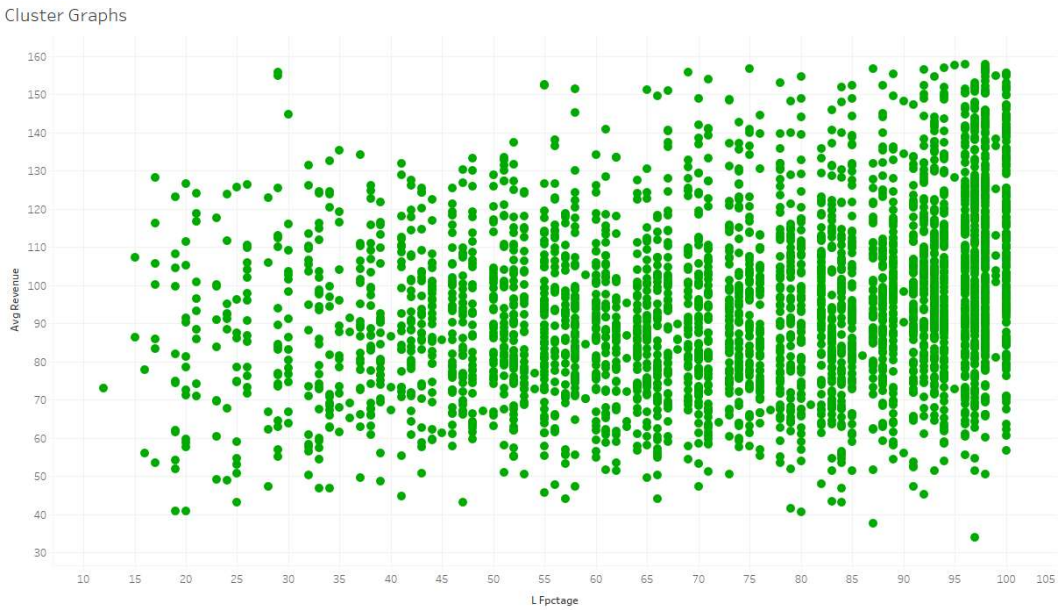
Multiplier Value	# of Routes Assigned
2	76
3	95
3.5	140
4	94

5	101
6	59

- Presented below is a visual example of how important the outlier removal process is. In this sample market YVRYLW, the outlier removal process completely changed the scope of the graph.
    - As you can see when comparing the first graph to the second, the outlier flights greatly expand the plain of the cluster graph and this can have an impact on the distances calculated between the different flight observations.



Graph 1: All Flights from the YVRYLW route coded by outliers (red means that it was an identified as an outlier and green means it was considered a normal flight)



Graph 2: Flights from the YVRYLW that were considered normal observations without the outliers included

RELEVANT CODE SNIPPETS

```
remove_outliers <- function(df, mult){
  nu_df <- df
  med <- median(df$AvgRevenue)
  absdev <- mad(df$AvgRevenue, constant = 1)

  if (nrow(df)>35){
    nu_df <- filter(nu_df,AvgRevenue < med + mult*absdev)
    nu_df <- filter(nu_df,AvgRevenue > med - mult*absdev)
  }

  df <- nu_df
}
```

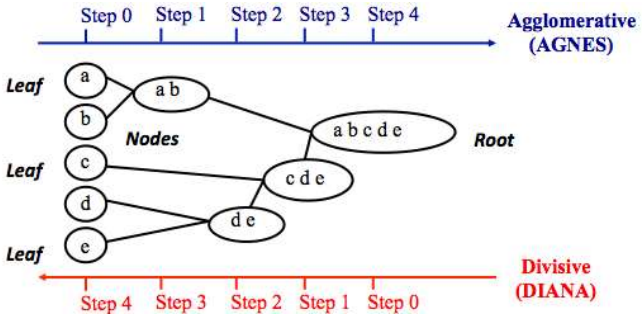
Step 5: Clustering Method Selection

- **Clustering,** when relating specifically to data, is a process of partitioning a set of data (or objects) into a set of meaningful sub-classes, called clusters.
    - In our case, the data we will be working is the flight information we have already collected and we will be grouping flights together based on their performance on our KPIs, Average Revenue and Load Factor.

Methods Explored:

- Before coming to a conclusion on which clustering methodology to implement we investigated a few options; Hierarchal clustering, PAM clustering and K-Means clustering.

- **First Method: Hierarchal Clustering** → involves creating clusters that have a predetermined ordering from top to bottom.
    - There are two forms of Hierarchical Clustering, Agglomerative and Divisive Hierarchical.
    - **Agglomerative(AGNES)** works in a bottom-up manner meaning each object(in our case flight observation) is initially considered as a single-element cluster (leaf)
        - At each step of the algorithm, the two flight clusters that are the most similar are combined into a new bigger cluster (nodes).
        - Procedure is iterated until all points are member of just one single big cluster (root).
        - Result is a tree which can be plotted as a dendrogram.
    - **Divisive (DIANA)** works in a top-down manner and therefore is an algorithm in inverse order of AGNES.
        - Begins with the root, in which all objects are included in a single cluster.
        - At each step of iteration, the most heterogeneous cluster is divided into two.
        - Process is iterated until all objects are in their own cluster



- **Second Method: PAM Clustering** → Stands for "partition around medoids" and is intended to find a sequence of objects (in our case flights) called medoids that are centrally located in clusters.
    - Flights that are tentatively defined as medoids are placed into a set ***S*** of selected Flights.
    - If ***O*** is the set of objects that the set **U = O – S** is the set of unselected objects.
    - Goal of the algorithm is to minimize the average dissimilarity of flights to their closest selected flight
    - There are two phases to the PAM method
        - Phase 1(BUILD) = a collection of k flights are selected for an initial set S.
        - Phase 2(SWAP) = one tries to improve the quality of the clustering by exchanging selected flights with unselected flights.
- **Third Method: K-Means Clustering** → It would classify flights in multiple groups (i.e., clusters), such that flights within the same cluster are as similar as possible (i.e., high intra-class similarity), whereas flights from different clusters are as dissimilar as possible.
- With this form of clustering, each group of flights in a cluster are represented by its center (i.e, centroid) which corresponds to the flight which represents the mean of all the flights assigned to assigned to the cluster.
- This approach to clustering assumes Euclidean Space and implements the Euclidean Distance formula to determine the distance and thus similarity between flight observations for any given directional market. The Euclidean distance formula can be seen presented below:

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

- With this form of clustering the total within-cluster sum of square measures the compactness (i.e goodness) of the flight clustering and the aim is for this value to be as small as possible. This Variation is expressed in the formula below:

$$tot.withiness = \sum_{k=1}^{k} W(C_k) = \sum_{k=1}^{k} \sum_{x_i \in C_k} (x_i - \mu_k)^2$$

- There are 5 general steps of what goes on in the process of k-means clustering and these steps can be seen presented below:

1. Specify the number of clusters (K) to be created (by the analyst)

2. Select randomly k flights from the route as the initial cluster centers or means

3. Assign each flight observation to their closest centroid, based on the Euclidean distance between the flight and the centroid flight

4. For each of the k clusters update the cluster centroid flight by calculating the new mean values of all the data points in the cluster.

5. Iteratively minimize the total within sum of square (Eq. 7). That is, iterate steps 3 and 4 until the cluster assignments stop changing or the maximum number of iterations is reached.

- **Step 4 Side note**: The centroid of a Kth cluster is a vector of length p containing the means of all variables for the observations in the kth cluster; p is the number of variables.
- The entire process outlined in the diagram presented above can all be automated utilizing the k-means function in R.

Final Method Selected: K-means Clustering

- We decided to implement the K-means function in R to carry out the clustering. This function takes the K means approach to clustering which aims to partition the data points such that the sum of squares from points to the assigned cluster centers is minimized.
- In R, when you run the k-means function, the output is a list with different pieces of information as listed below:

- `cluster` : A vector of integers (from 1:k) indicating the cluster to which each point is allocated.
- `centers` : A matrix of cluster centers.
- `totss` : The total sum of squares.
- `withinss` : Vector of within-cluster sum of squares, one component per cluster.
- `tot.withinss` : Total within-cluster sum of squares, i.e. sum(withinss).
- `betweenss` : The between-cluster sum of squares, i.e. $totss-tot.withinss$.
- `size` : The number of points in each cluster.

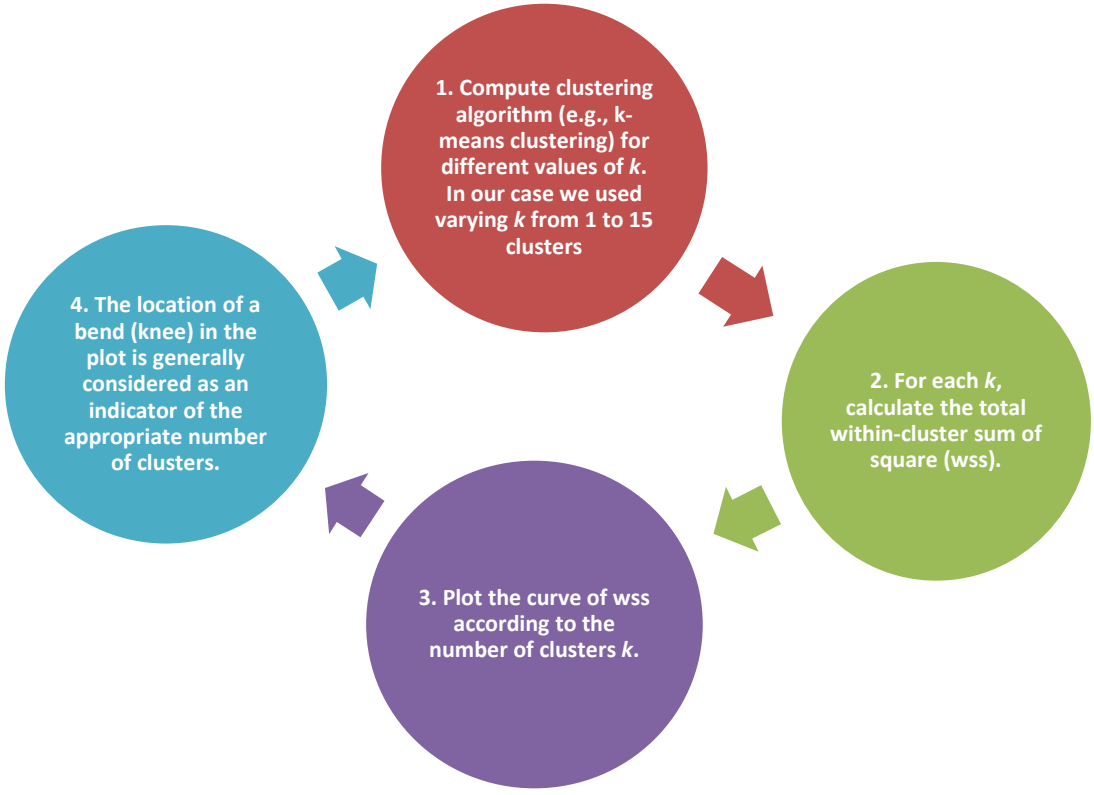
- **Reasons for selecting K-means clustering:**
  - The main reason for selecting the K-means clustering method is the scalability of the model. It represents a method that can be easily applied to large datasets and in that sense provides the better alternative for clustering the Hierarchal.
    - In our case since we operate tens of thousands of flights annually, it makes more sense to implement this method for efficiency purposes.
  - Additionally, the output provided by the K-means function provides an output that has the total within-cluster sum of squares which is optimal for selecting the number of clusters to have on a per route basis. This will be explained in the Elbow Methodology section.
- **Reasons Against Selecting Other Methodologies:**
  - The main reason we decided against utilizing Hierarchal Clustering was for logistics.
    - Hierarchal clustering is quadratic in the number of Data Objects meaning where *n* is the number of data objects it is represented as $O(n^2)$.
      - With hundreds of thousands of data observations it would be inefficient to go through hierarchical clustering.
  - P.A.M is another partitioning method of clustering and has some similarity to k-means clustering. We decided to go with k-means due to its more simple nature and the presence of a clear methodology to select an optimal k.

Selecting Optimal k Clusters: Elbow Methodology

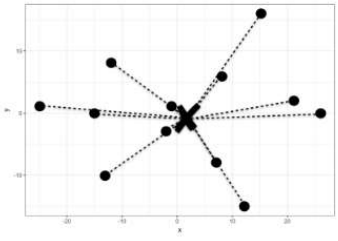
- We implemented the elbow method to determine the optimal k to assign to each route.

- This method relies on the calculation of the Total Within-Cluster Sum of Squares (tots.withinss) across every cluster
  - This refers to the sum of Euclidean distances between each observation & the centroid corresponding to the cluster to which the observation is assigned as discussed above.

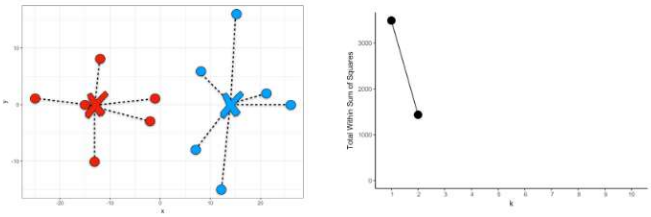
- The steps to the elbow methodology have been outlined below:



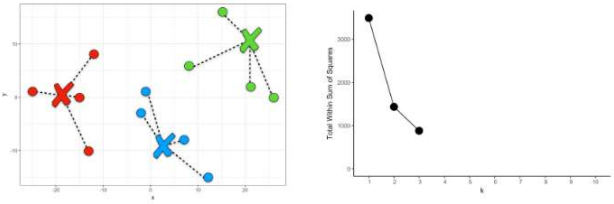
Total Within-Cluster Sum of Squares: k = 1



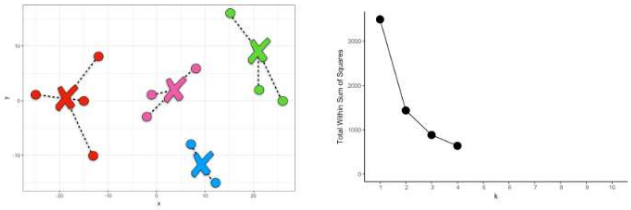
Total Within-Cluster Sum of Squares: k = 2



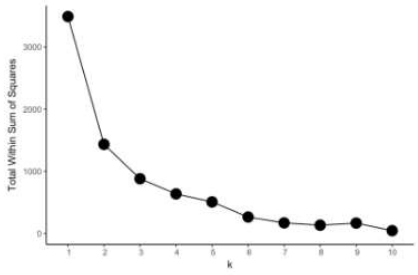
Total Within-Cluster Sum of Squares: k = 3



Total Within-Cluster Sum of Squares: k = 4



Elbow Plot



- The whole process described above can be automated in R with a combination of different functions.

Relevant Code Snippets

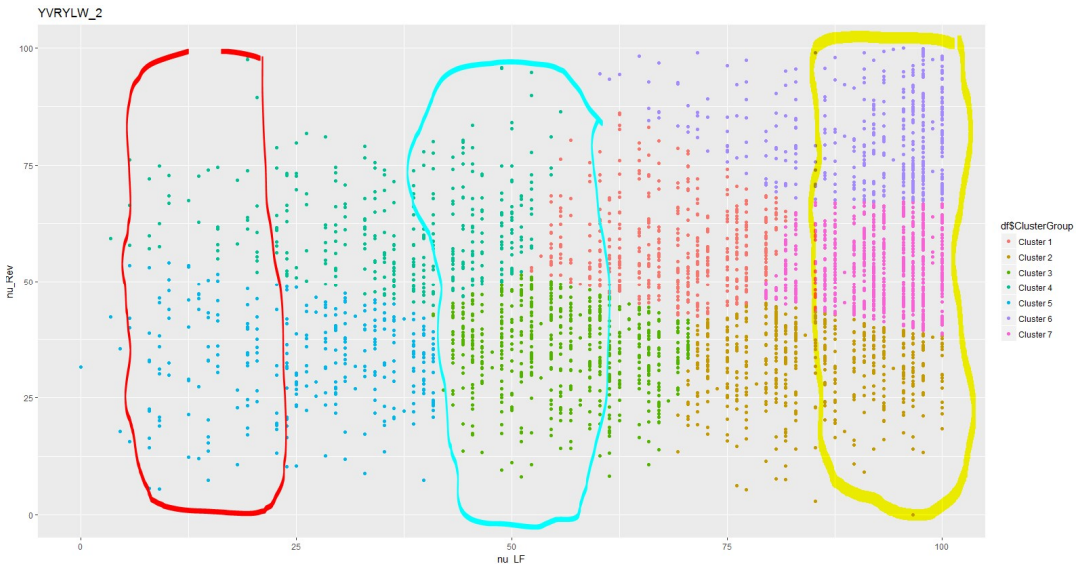
```
81  #'elbowGenerator' is a function dedicated to generati
82  elbowGenerator <- function(eval_df){
83    elbow <- map_dbl(1:15, function(k){
84      model <- kmeans(eval_df, k, nstart = 10)
85      model$tot.withinss
86    })
87    elbow
88  }
89
```

Step 6: Clustering Review

- This part of the process was dedicated to looking back on the flight groupings we conducted in the previous steps. We set out to investigate the effectiveness using Average Revenue and Load Factor as indicators to create distinct groups of similarly performing flights.
- This portion of the project consisted on a few different tasks:
    - Reviewing segmentation between flight information to see if the clusters assigned made sense.
    - Reviewing the number of cluster groups to ensure there weren't too many or too few for every directional market.
    - Reviewing the outlier removal process to make sure that the flight information used for clustering is appropriate and there were no outliers causing skewed clustering results.

Segmentation & Outlier Removal Review:

- We did the majority of this in R using a custom function designed to present the clusters graphically and improve visualization.
- In this process we were looking to see if we could find multiple different cluster groups within on specific area of a graph to show that both variables were contributing to the clustering process. The cluster graph for YVRYLW provides an example of this below:



- As you can see in the highlighted areas on the graph presented above, for a group of observations that share the same Load Factor there are multiple different clusters and this is a sign of effective clustering because it represents that they can have differing values for average revenue.
- In this review segment we were also able to determine whether our outlier removal methodologies were appropriate. By looking at the clustering graph we made above, we were also able to decipher whether the outlier removal process had done its job in removing unusual flight observations.
- We tested this out for a number of markets to make sure that the clustering was generally performing appropriately.

Review of Ks

- For this process we had to initially set a maximum number of Ks that could be assigned to any given route within our network in an effort to promote some form of consistency.
    - We decided that this number would be 8 because this would give us enough freedom to cater to the needs of larger routes that may require more clusters without becoming too redundant.
    - We also decided to set the maximum number of Ks at 8 is because it made mathematical sense. As you can see in the table shown below, the number of routes assigned to a specific K group gradually dwindles and that means that the higher number Ks are becoming less necessary.
    - Additionally, by making the number of K's reasonable and not to large, we can allow the revenue analysts to manage their flights in a more efficient manner.
- In order to do this, we had to undergo trial and error with the clustering script and make edits to our custom optimal K assignment function until the greatest K assigned to any route was 8. The breakdown of the K's assigned can be seen in the table presented below:

K	Number of DirMkts Assigned to this K	Average Frequency of DirMkt Assigned to this K
1	21	8
2	38	34
3	125	147
4	174	317
5	116	480
6	43	1438
7	39	2838
8	10	4630

- The final step of the review process was to go back and create a cluster group assignment for all the flights that had been deemed as outliers since they had been removed from the working data set.

- To identify them, we utilized R to go back in and assign each flight that had not been given a cluster to the cluster group 0

Step 7: Clustering Build Graphs and Information

- Once all of this had been done we had to export the newly clustered Data back into a Sandbox table in SQL to be further manipulated
- The next step would be to prepare the data to be aggregated by cluster per each different route from a given NDO. This would be done in order to give us the ability to look at a routes build curve.
- The first thing we had to do was to write a query in SQL that would give us every flight from 2017-01-01 to 2018-06-30 with its assigned cluster group and other facts about the flight. We collected data using the filter in the screenshot below which allowed us to collect Data from flights up to 330 days out. This was done in order to collect historical data in order to start the process of creating build curves for the clusters that we had established.

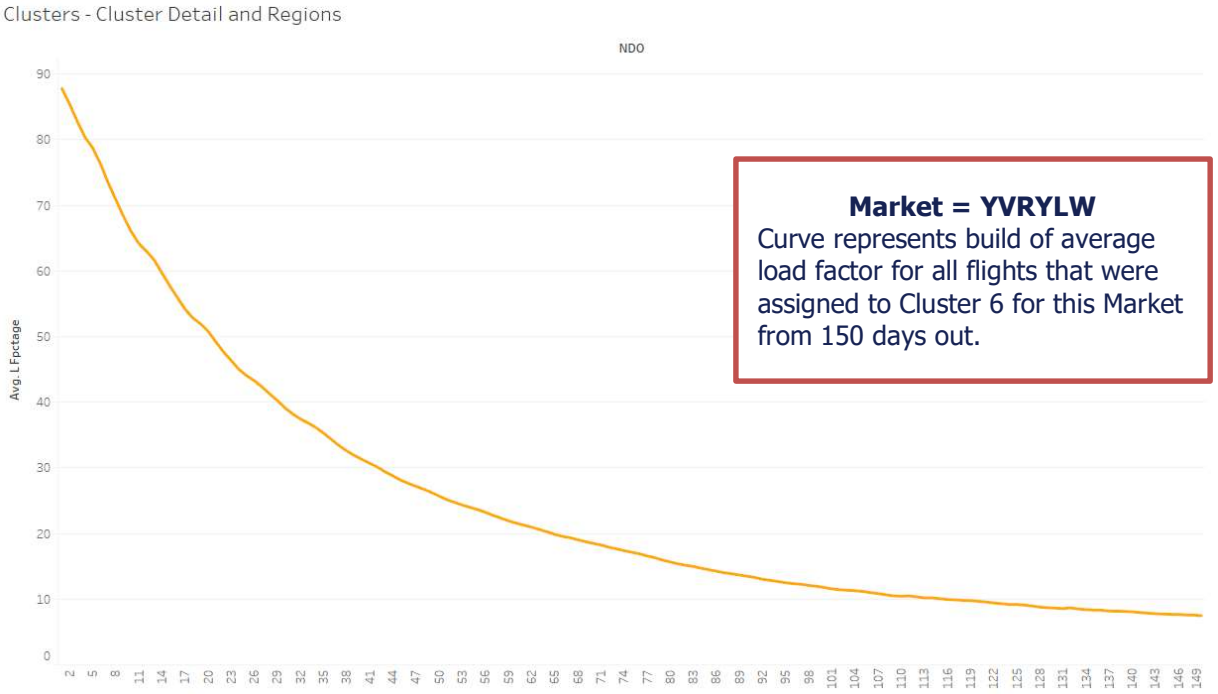
```
WHERE    CaptureDate >= '2016-01-01'
AND      FlightDate BETWEEN '2017-01-01' AND '2018-06-30'
```

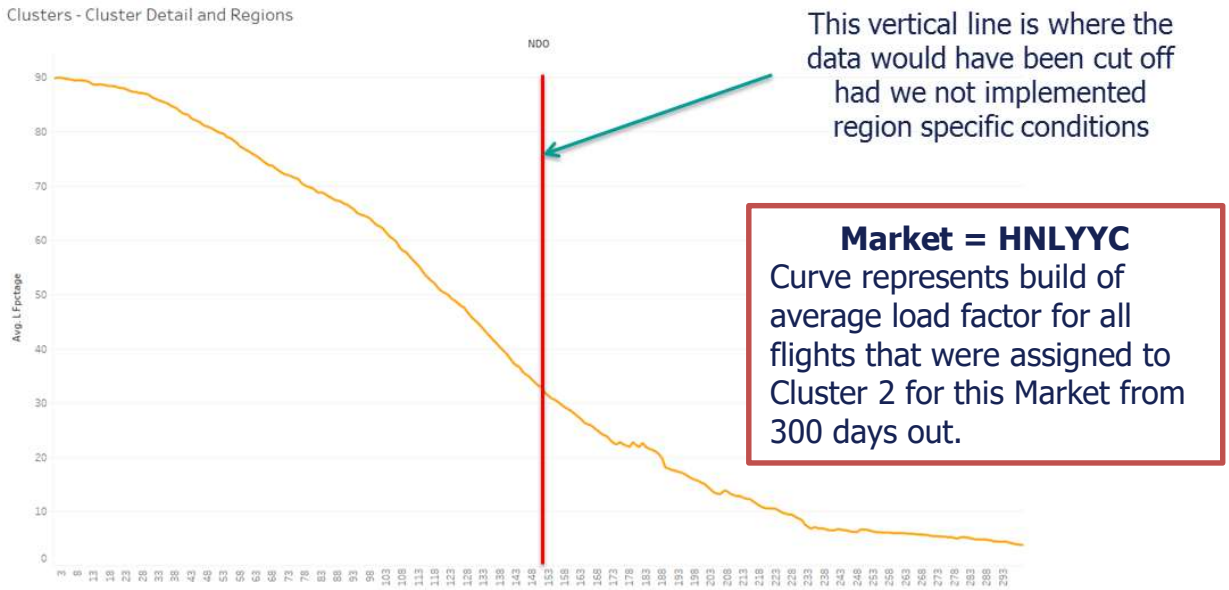
  - We decided that we would create a logic that would collect data up to different NDO's for different regions, sub-regions and some specific single route instances.
    - We did this is because every region has different characteristics.
    - This is highlighted by our consumer purchasing habits for Leisure versus. Business
      - Business Routes are more likely to have flights that sell out closer to their departure date whereas Leisure routes are more likely to sell at a greater frequency at a higher NDO.
    - For example, with the route CUNYYC in the International-Sun Region, this represents a Leisure route therefore making it more likely to fill up at a faster rate than a route in the Domestic region
    - This means that we would need to capture data from more NDO's for this route in order to get a better picture of its build.
  - The logic we decided on for NDOs to collect can be seen presented in the table below:

Number of NDOs Collected	Region/Sub-Region/Route Rule Applies To
300	International-Sun
	Transatlantic
	Hawaii
225	Florida
	Arizona
	BNAYYC
	BNAYYZ
	YYCBNA
	YYZBNA

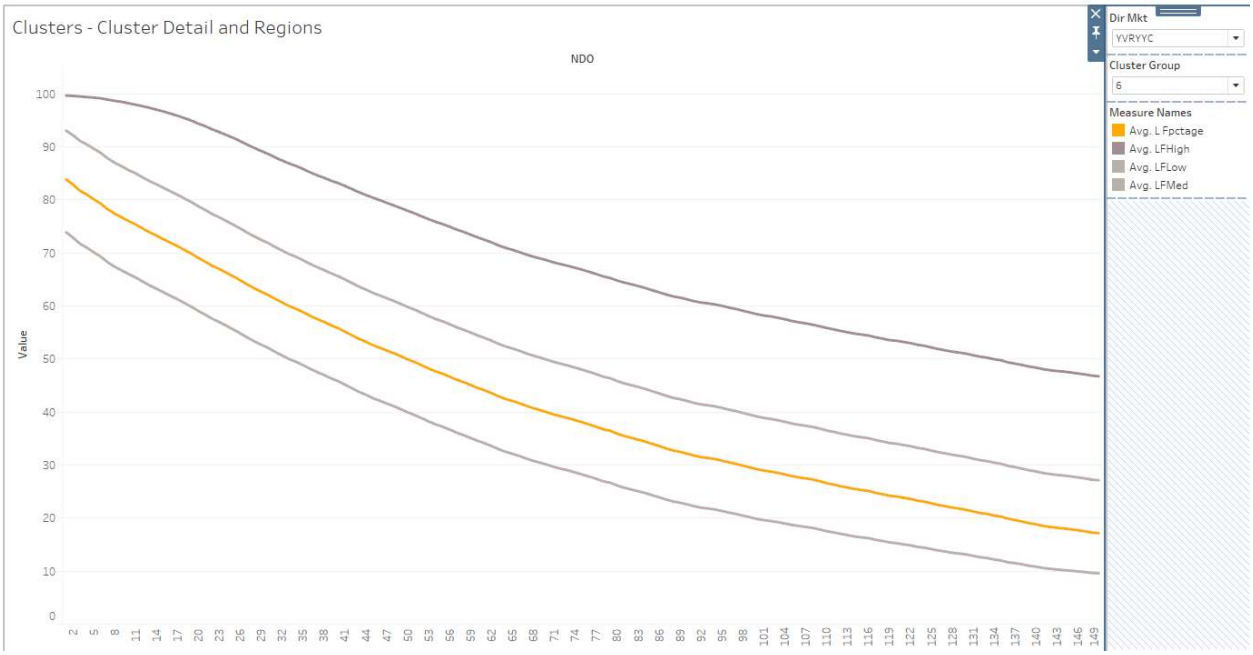
	DENYYC
	YYCDEN
150	Everything other region/sub-region/route that is not in the first two rows.

- The two screenshots presented below illustrate the importance of having differing NDOs collected for different regions.



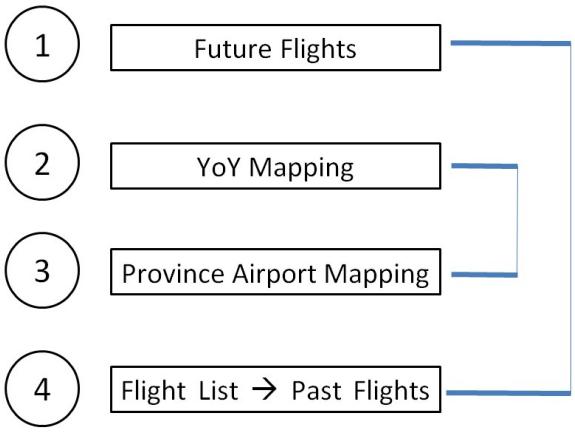


- Once this information is collected we combined the flight data with the revenue data for each of the routes per cluster for every NDO. We then set out to create some custom regions on the build curve to further help analysts with their performance tracking.
    - To create an avenue for further analysis for our revenue analysts, we created a number of regions on the build curve dashboard to provide an alert that can show which flights are performing well and which flights are underperforming.
- Listed below are the characteristics of the performance regions defined:
    - Low Performing: Flights that fall within this region have a Load Factor **less than** the Average Load Factor of their cluster – 10. These are flights that are performing poorly and need some adjustment from analysts.
    - Normal Performing: Flights that fall within this region have a Load factor **less than** the Average Load Factor of their cluster +10 **AND greater than** the Average Load Factor of their cluster – 10. These are flights that are performing in an expected range because they are relatively close to the average load factor of their cluster with 10% margin of error. These are flights that Revenue analysts can be confident are steady on track.
    - Yield Region: Flights that fall within this region have a Load factor **less than** the Average Load Factor of their cluster +30 **AND greater than** the Average Load Factor of their cluster + 10. These are flights that present high opportunity for yield.
    - Protect Region: Flights that fall within this region have a Load factor **greater than** the Average Load Factor of their cluster +30. These flights need monitoring because they are being sold at an extremely fast rate and are in danger of being sold too soon.
- With all this information now stored in SQL, we were then able to export the table to Tableau and visualize the builds of each route on a cluster basis. An example of this can be seen in the graph presented below:



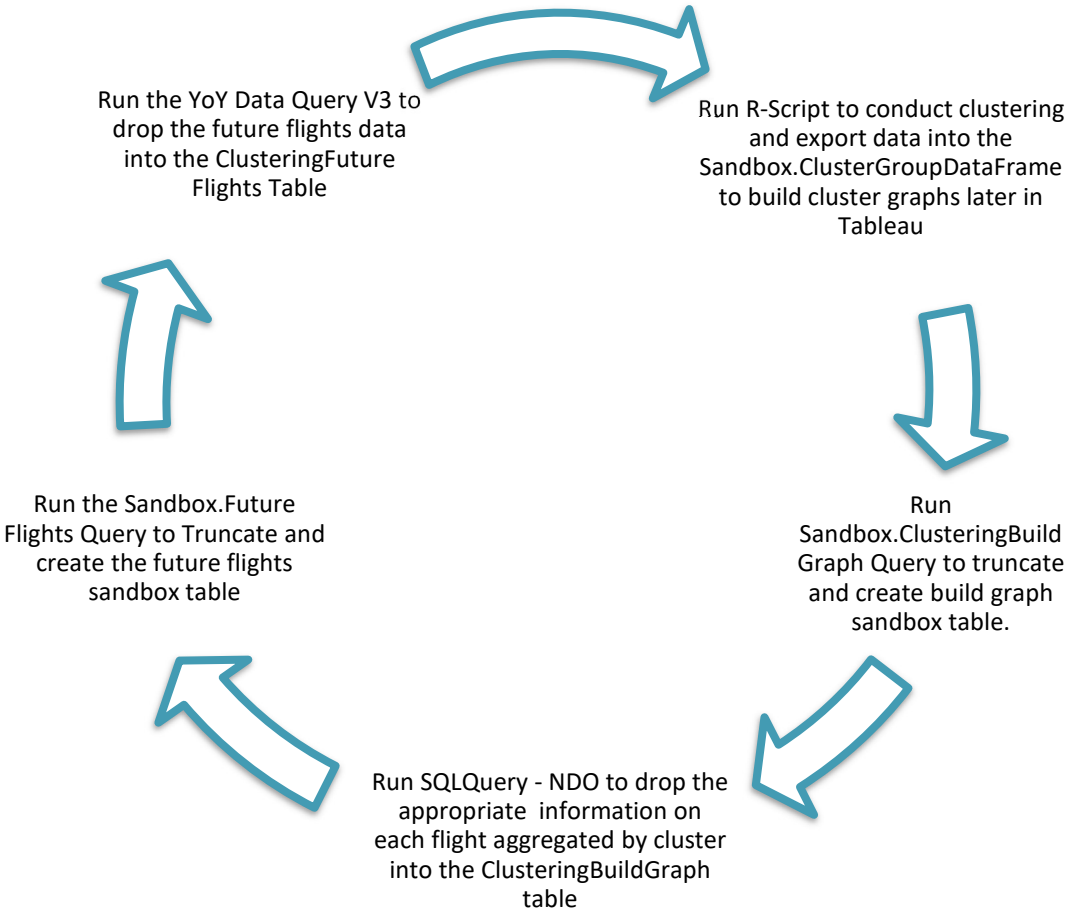
Step 8: Cluster Assignment with YoY Mapping

- The last major task we had to conduct on this project was to get data for future flights that have not taken off yet in order to make comparisons to this year on a flight level per cluster. This had a few different elements to it and represented one of the most essential parts of the presentation because it represents the area of main benefit to be received from this project.
- The first key element in doing this was to collect data from all our flights from today to the end of time, meaning all flights from the current day to the actual date of the flight in the future.
    - We collected data from almost identical data fields as those which we collected data for from past flights.
- We then had to account for date mapping in order to make this comparison accurate. From year to year, the placement of holidays on the calendar changes and therefore this will affect the different stories told in the data.
    - For example, the date for Easter can change from year to year and therefore, we have to take that into consideration when making comparison between flights on a certain holiday during that year and flights in the next year.
- To do that we also had to get the Airport Mapping of the Routes we are doing the Date mapping from because holidays can also differ from Province to Province and Country to Country.
- Once these two things had been completed, only then were we able to match flight dates from last year to this year and therefore assign future flights to the appropriate cluster group comparison in the next year.
- The diagram below represents the general process of providing a mapping for future flights with a flight from the previous year and the cluster group it belongs to:



The aim of the SQL query is to join tables 1 & 4 utilizing table2 & 3 as references to get both linked together. To do this we used the characteristics Date, Flight# and Time to identify each entry within the table.

- With the hope of having this project utilized to build a tool for clustering in the future, we took the liberty of breaking down the different scripts utilized in the entire process from start to finish in the diagram below:



Step 9: Study of Results

- After concluding the bulk of the project, we have now been able to create a combination of tableau dashboards to illustrate the potential of adding an analytical tool based off of K – means clustering of flights.
- Here we can reflect on the possible benefits and limitations associated with implementing this project in a larger and more advanced scale.
- It is also important to note that the results of this project represent the first stage in what could be a long process to implementing a clustering system that is something like what is proposed in this project.

Potential Strengths

- With a system like this in place we can better facilitate exception detecting capabilities for our analysts. By looking at flights performance with reference to the cluster they are assigned to, we can further empower analysts to make game-changing decisions regarding pricing/capacity.
- If implemented accurately we can save a lot of money by developing a clustering methodology developed in-house.
    - This system could be a potential replacement for the Seabury split history clustering methodology and could provide a more data driven method of categorizing our flight performance.
- This could also provide a strong method to visualize our flights on an even more granular manner than we do and in the process could allow our analysts to make even more informed decisions.

Limitations

- Mapping flights based on flight number and time the flight runs in the day can often be risky because these are characteristics that can change from year to year and could potentially negatively impact the accuracy of the tool.
- In this iteration of the projected we decided to remove outliers from routes with **more than 35 flights** during the time period we were collecting data for. This is because we felt that these routes already had a very low number of flights and we needed as many observations as we could have.
    - This is a sub-optimal solution because regardless of the number of flights for a route there may still be a few outliers. Due to the time constraints for conducting this project, we weren't able to outline a method to factor in outliers for low frequency routes and this is something we should definitely explore in future iterations of the project.
- Another limitation from this project is related to the definition of performance regions in step 7. We utilized a constant margin of error of 10% to define the normal performing region, yield region and low region.
    - The standard deviation could potentially be an even more accurate margin of error for these regions. We didn't use it in this iteration of the project because with markets in which you don't have a lot of flight observations the standard deviation has greater fluctuations and therefore it was better to set a constant threshold for consistency.

    - Due to time limitations we weren't able to investigate ways to make the standard deviation a feasible margin of error for the definition of these regions however, in the future we could dedicate more time to exploring this option.
- For markets with low frequencies, it's very difficult to create cluster groupings.
    - This is due to the fact that there aren't that many observations and therefore it's difficult to come up with an average because of the lack of data available.
- Our proposed analytical tool currently doesn't have a way to factor in big changes in market conditions.

Future Improvements & Developments:

- Our proposed tool could have Cluster adjustment potential for the analysts to meet performance goals. In future iterations of the project we plan to explore the possibility of having Analysts adjust the cluster build curves for their markets to factor in market condition changes and set forward looking goals as opposed to backwards looking goals.
    - Ex. Sold too soon clusters, Lowering Final Load Factor, Raising ending Load Factor goals.
- We also plan to investigate the possibility of including a snapshot feature to capture build of the future flights and their interaction with the performance regions.
    - By doing this we intend to create an avenue for analysts to see the effect of their adjustments over time.
        - By plotting a build curve for the future flights using this feature, you can see how a flight reacts to an analyst's tweaks.
    - We also intend to create a new way to measure KPIs based on flight performance (Ex. % days spent in specific performance regions)
- The definition of performance regions could be investigated further. We used a constant margin of error of 10% to define most of the regions but this doesn't take into account differences in markets and what would be considered an appropriate margin of error for different markets.
    - Potentially using standard deviation to make it more route specific
- Add more variables to the clustering to capture a more dynamic array of flight observations to our
    - Ex. Method to capture guest booking window to the model
- Efficiency of the code used to conduct the clustering portion of the tool could be improved in order to allow quicker data manipulation.

Informational Research Threads

https://uc-r.github.io/hc_clustering#algorithms

http://www.saedsayad.com/clustering_hierarchical.htm

https://www.cs.umb.edu/cs738/pam1.pdf

https://www.bvicam.ac.in/news/INDIACom%202008%20Proceedings/pdfs/papers/158.pdf