

Mille Lacs Corporate Ventures

Knowledge Transfer Documentation

MSBA 6410 - Section 092

Group 1

Sarah Black, Michael Deguire, Anthony Meyers, Danny Moncada, Jonathan Watkins



Contents

1. Overview	- 2 -
1.1 Background & Context	- 2 -
1.2 Problem Statement	- 2 -
1.3 Data Sources	- 2 -
2. Technical Specifications	- 3 -
2.1 Software & Tools	- 4 -
2.2 Environment	- 4 -
3. Solution Overview	- 4 -
3.1 Summary	- 4 -
4. Data Engineering	- 6 -
4.1 Data Extraction	- 6 -
4.2 Data Preparation	- 7 -
5. Analysis	- 7 -
5.1 Player Demographics	- 8 -
5.2 Geospatial & Location Demographics	- 25 -
5.3 Effectiveness of Promotion Campaigns	- 39 -
5.4 Clustering & Customer Segmentation	- 50 -
5.5 Association Rules	- 56 -
5.6 Sentiment Analysis (with Yelp reviews)	- 66 -
5.7 Assumptions	- 74 -
6. Results, Findings, Insights	- 74 -
7. Appendix	- 75 -
8. References	- 76 -

1. Overview

1.1 Background & Context

Mille Lacs Corporate Ventures strives to improve the quality of life for the members of the Mille Lacs Band and those in their community. They want to be a force that improves businesses and communities by infusing passion and ideas. MLCV invests in gaming, hospitality, marketing and technology, and local businesses.

We have been provided with data on headcount, player demographics, and promotion redemptions and through our exploration and analysis, we have looked into MLCV's questions and have some suggestions to help reverse the downward trends in headcount and promotion redemption. In addition to the data provided by MLCV, our team looked at public Yelp reviews and performed sentiment analysis to gain an understanding of what people liked and didn't like about their experience at the properties because we believe that in order to attract new customers MLCV will need to provide an excellent experience value for money.

1.2 Problem Statement

MLCV partnered with the Carlson MSBA program to help analyze data from their Mille Lacs and Hinckley casinos. Since 2015, headcount has been declining on average 9% per year and promotional redemptions have declined at a rate of 1.2%. MLCV would like to use insights from the data to reverse these trends and have also provided some particular questions that they would like to see answered.

Questions to be answered:







- Which promotions are popular or not? Which promotions drive trip decisions?
- What are the demographics behind offer utilization?
- How do we use promotions to improve the headcounts?

1.3 Data Sources

Data provided by Mille Lacs Corporate Ventures::

- **Player Dimension** - demographics & attributes describing players

- **Coupon Dimension** - attributes describing gaming coupon
- **Coupons Redeemed Fact** - metrics related to all coupons that were redeemed
- **Coupon Group Fact** - metrics related to all coupons that were issued
- **Player Day Fact** - metrics related to all player visits and their gaming history

 CouponGroup.csv	11/1/2019 11:53 AM	Microsoft Excel C...	10,707,725 ...
 DimCoupon.csv	10/25/2019 12:30 ...	Microsoft Excel C...	2,296 KB
 DimPlayer.csv	10/28/2019 2:31 PM	Microsoft Excel C...	76,424 KB
 FactCouponRedeem.csv	10/25/2019 12:28 ...	Microsoft Excel C...	372,312 KB
 Hotel Data.xlsx	11/22/2019 10:57 ...	Microsoft Excel W...	89 KB
 PlayerDay.csv	10/31/2019 3:59 PM	Microsoft Excel C...	982,635 KB

The data files provided by MLCV were .csv or .xlsx formatted files.

External Data Sources utilized by the team:

- Yelp Reviews - customer reviews of experience at MLCV
- American Community Survey - population densities
- Forcluster_output.csv - for cluster analysis

2. Technical Specifications

2.1 Software & Tools

Technical specifications for the R environment:

```
R version 3.6.1 (2019-07-05)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows Server x64 (build 14393)
```

```
Matrix products: default
```

```
locale:
```

```
[1] LC_COLLATE=English_United States.1252
[2] LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.1252
```

```
attached base packages:
```

```
[1] stats      graphics  grDevices  utils      datasets  methods    base
```

```
other attached packages:
```

```
[1] arules_1.6-4  Matrix_1.2-17 dplyr_0.8.3   ggplot2_3.1.1 tidyr_1.0.0
```

```
loaded via a namespace (and not attached):
```

```
[1] Rcpp_1.0.1      pillar_1.3.1    compiler_3.6.1  plyr_1.8.4
[5] base64enc_0.1-3 tools_3.6.1      zeallot_0.1.0   digest_0.6.18
[9] uuid_0.1-2      jsonlite_1.6     evaluate_0.13    tibble_2.1.1
[13] lifecycle_0.1.0 gtable_0.3.0     lattice_0.20-38  pkgconfig_2.0.2
[17] rlang_0.4.0     IRdisplay_0.7.0 IRkernel_0.8.15  repr_0.19.2
[21] withr_2.1.2     vctrs_0.2.0      grid_3.6.1       tidyrselect_0.2.5
[25] glue_1.3.1      R6_2.4.0         pbdZMQ_0.3-3     purrr_0.3.2
[29] magrittr_1.5     backports_1.1.4  scales_1.0.0     htmltools_0.3.6
[33] assertthat_0.2.1 colorspace_1.4-1 lazyeval_0.2.2   munsell_0.5.0
[37] crayon_1.3.4
```

Python Environment Details

Here are the environment details...

```
C:\Python\envs\MSBA2020\python.exe
3.7.1 (default, Oct 28 2018, 08:39:03) [MSC v.1912 64 bit (AMD64)]
sys.version_info(major=3, minor=7, micro=1, releaselevel='final', serial=0)
```

```
This notebook is using pandas version: 0.25.3.
This notebook is using pandas version: 0.25.3.
This notebook is using numpy version: 1.16.4.
This notebook is using seaborn version: 0.9.0.
This notebook is using matplotlib version: 3.1.2.
```

Geospatial Analysis was performed using:

```
[1] "maptools" "maps" "zipcode" "forcats" "stringr" "dplyr" "purrr"
    "tidyr" "tibble" "ggplot2" "sp" "base" "utils"
    "tidyverse" "RevoUtils" "stats" "methods" "graphics"
    "grDevices" "datasets" "RevoUtilsMath" "readr"
```

The Shapefile preprocessing steps were performed using:

[1] QGIS Version 2.18.2-Las Palmas (exported), © 2002-2016 QGIS Development Team

Clustering Analysis of Players Redeeming Promotions performed using:

Sentiment Analysis of Yelp! Reviews was performed using:

```
NLTK v 3.4.5
Pandas v 0.24.2
BeautifulSoup v 4.4.0
SKLearn v 0.21.0
```

2.2 Environment

Here are the specifications for the environment that was used:

```
Platform: x86_64-mingw32/x64 (64-bit)
Running under: Windows Server x64 (build 14393)
```

3. Solution Overview

3.1 Solution Summary

- Adverse macroeconomic conditions will put further pressure on the casino industry in the coming years.
- Postal codes near casinos are relatively saturated, but the Twin Cities metro area is under visiting.
- Cluster analysis shows five distinct groupings of player that are redeeming promotions
- Cash, Hotel and Free Slot Gaming promotions boast the highest redemption rates
- Sentiment analysis of Yelp! reviews indicates [smoking] was a frequent indicator of a negative review.
- Sports betting offers untapped micro-segments that target a new customer base.

4. Data Engineering

4.1 Data Extraction

MSBA 6410 - Exploratory Analytics

Mille Lacs Corporate Ventures Live Case

```
## Load in the fact table - just do it in one shot
## We use low_memory = False because it is mixed data types
## I hope your machines aren't as heavily used as our VMs were to generate these analyses!
```

```
coupon_fact_df = pd.read_csv(team_folder+coupon_fact, low_memory = False,
                             encoding = "cp1252")
```

```
## Generate some summary statistics of the dataframe, just to see if there are missing values
## Or any strange abnormalities in the data
```

```
coupon_fact_df.describe()
```

	CouponID	PlayerID	RedeemValue	RedeemCount	SiteID	RedemptionNumber	ValidMonth	ValidYear
count	2.777358e+06	2.777358e+06	2.777358e+06	2.777358e+06	2.777358e+06	1.907100e+05	2.777358e+06	2.777358e+06
mean	9.274712e+04	7.053240e+08	2.386644e+01	1.004516e+00	1.578012e+00	1.858328e+07	6.218163e+00	1.738322e+01
std	1.486246e+04	2.599271e+08	2.688272e+01	8.191522e-02	4.938768e-01	1.110173e+06	3.262830e+00	1.891812e+00
min	1.000000e+00	3.300000e+01	0.000000e+00	0.000000e+00	1.000000e+00	1.666513e+07	1.000000e+00	5.000000e+00
25%	9.124200e+04	5.303723e+08	1.000000e+01	1.000000e+00	1.000000e+00	1.764133e+07	4.000000e+00	1.700000e+01
50%	9.484000e+04	7.103038e+08	2.300000e+01	1.000000e+00	2.000000e+00	1.867601e+07	6.000000e+00	1.800000e+01
75%	9.906600e+04	1.000156e+09	2.900000e+01	1.000000e+00	2.000000e+00	1.961201e+07	9.000000e+00	1.800000e+01
max	1.034460e+05	1.000805e+09	5.000000e+03	1.000000e+01	2.000000e+00	2.029006e+07	1.200000e+01	2.200000e+01

4.2 Data Preparation

To prepare the data for analysis, we used the following methods:

- Aggregation
- Removal of outliers
- Binning (grouping like items together)
- Removal of null or N/A values (which happened often)
- Feature engineering / creation
- Normalization

Examples of each of these common data preparation tasks can be seen in the analysis below.

5. Analysis

5.1 Player Demographics

We start with just the **"Player"** dimension for a few reasons:

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

- This is the only place where we have some attributes related to the player aside from the ID.
- Bringing in the other facts or dimensions will just populate the table with missing values
- We can do an analysis over the whole population, determine what groups stand out, and then go back to the **Redeemed Coupon fact** to get more information about them.

```
## Look at the column names
```

```
player_dim_df.columns
```

```
Index(['PrimaryPropertyID', 'PrimaryPropertyName', 'PlayerID', 'City', 'State',  
      'ZipCode', 'TierID', 'TierName', 'DistFromML', 'DistFromHK',  
      'LastPlayDate', 'Description', 'BirthMonth', 'BirthYear',  
      'PlayerStatus', 'Prison Flag', 'Deceased Flag', 'DMA Flag',  
      'Person # Marital Status', 'Estimated Current Home Value',  
      'Children Presence of Children 0-18',  
      'DSE Discretionary Spend Estimate', 'Est Household Income V6',  
      'PIQ Match Type', 'NCOA MOVE DATE', 'NCOA Move Type'],  
      dtype='object')
```

```
## Observe the first five rows to see the structure of the table
```

```
player_dim_df.head()
```

	PrimaryPropertyID	PrimaryPropertyName	PlayerID	City	State	ZipCode	TierID	TierName	DistFromML	DistFromHK
0	1	Grand Casino ML	33	ELK RIVER	MN	55330	25	Preferred	53.0	60.0
1	1	Grand Casino ML	10014	GARFIELD	MN	56332	25	Preferred	89.0	126.0
2	1	Grand Casino ML	10025	ELK RIVER	MN	55330	25	Preferred	53.0	60.0
3	1	Grand Casino ML	10063	TALMOON	MN	56637	25	Preferred	104.0	115.0
4	1	Grand Casino ML	10187	PEQUOT LAKES	MN	56472	25	Preferred	51.0	82.0

Right off the bat, we notice there is no "Age" column. So we will have to generate one ourselves. We can also choose to focus only on columns that are required for our population analysis. Let's include:

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

- PrimaryPropertyName
- City
- State
- ZipCode
- TierName
- LastPlayDate
- BirthMonth / BirthYear
- Person # Marital Status
- Est Household Income V6
- Children Presence of Children 0-18
- Person # Marital Status

```
## Since these column names aren't very descriptive, we're going to modify them too

## There's an extra space in the Est Household Income column! What fun it was to debug that!!!!
## Pull out on the columns we need
player_columns_to_include = \
    ["PrimaryPropertyName", "City", "State", "ZipCode", "TierName", "LastPlayDate", "BirthMonth", "BirthYear",
     "Person # Marital Status", "Est Household Income V6", "Children Presence of Children 0-18",
     "Person # Marital Status"]

new_col_names = ["Casino Site", "City", "State", "Zip Code", "Player Tier", "Last Gamble Date", "Month", "Year",
                 "Marital Status", "EstimatedHousehold Income", "No. of Dependents", "Marital Status"]

## Subset the player information based on the selected columns
player_dim_df_subset = player_dim_df[player_columns_to_include]

## Give the subsetted dataframe better/more descriptive column names
player_dim_df_subset.columns = new_col_names
```

```
## Show the first five rows of the new cleaned up table
```

```
player_dim_df_subset.head(5)
```

	Casino Site	City	State	Zip Code	Player Tier	Last Gamble Date	Month	Year	Marital Status	EstimatedHousehold Income
0	Grand Casino ML	ELK RIVER	MN	55330	Preferred	2019-10-16 18:40:18.000	7	1961	Married Extremely Likely	138
1	Grand Casino ML	GARFIELD	MN	56332	Preferred	2019-10-24 07:22:04.000	9	1935	Married Extremely Likely	36
2	Grand Casino ML	ELK RIVER	MN	55330	Preferred	2019-10-23 07:46:00.000	10	1938	Unknown Scored	18
3	Grand Casino ML	TALMOON	MN	56637	Preferred	2019-09-29 09:41:27.000	12	1951	Married Extremely Likely	49

Player Age Distribution

Here we perform our first transformation of this table and transform the Month/Year columns in a format that will allow us to calculate Age:

MSBA 6410 - Exploratory Analytics

Mille Lacs Corporate Ventures Live Case

```
player_dim_df_subset["Birth Date"] = \
    player_dim_df_subset["Month"].astype(str) + "-" + player_dim_df_subset["Year"].astype(str)

## Custom function to get the birth month and year in right format needed to perform our Age calculation

player_dim_df_subset["Birth Date"] = \
    player_dim_df_subset["Birth Date"].apply(lambda x: dt.datetime.strptime(x, "%m-%Y"))

## Create a variable for today's date

today = dt.datetime.today()

## Create the Age column by performing a few steps
## Use the today variable to subtract the birth date column
## Transform the calculation to days and divide by 365.25 to account for Leap years - and output a rounded
player_dim_df_subset["Age"] = \
    round((today - player_dim_df_subset["Birth Date"]).apply(lambda x: x.days) / 365.25, 0).astype(int)

## Confirm our transformation was successful by showing the first 5 rows

player_dim_df_subset.Age.head(5)

0    58
1    84
2    81
3    68
4    55
Name: Age, dtype: int32

## Anyone below the age of 18 is automatically ruled out - they are ILLEGAL
## There's only four instances of these players - we'll just scrap them for this analysis

player_dim_df_subset.groupby("Age").agg({"State": "count"}).head(6)
```

State	
Age	
-5176	1
-3173	1
-2174	1
-399	1
18	535
19	3309

A few observations on the Age column. Likely due to the information in the "Year" column in the dimension, we are seeing some strange Ages, a few negative values (meaning some of the gamblers haven't been born yet) and some very very large Age outliers, means we will take these values out prior to doing the histogram of Ages. 120 has 64 players so this is where we will set the cut off.

MSBA 6410 - Exploratory Analytics

Mille Lacs Corporate Ventures Live Case

```
## Filter on the strange Ages to remove them before our first analysis, where having a clean distribution helps
player_dim_df_subset = \
    player_dim_df_subset[(player_dim_df_subset.Age > 0) & (player_dim_df_subset.Age <= 120)]
```

```
## Show the two unique casino sites so we know what to subset
```

```
player_dim_df_subset["Casino Site"].unique()
array(['Grand Casino ML', 'Grand Casino HK'], dtype=object)
```

```
## Create a player dimensio for each Casino
```

```
grand_casino_ml_player_dim = \
    player_dim_df_subset[player_dim_df_subset["Casino Site"] == "Grand Casino ML"]
grand_casino_hk_player_dim = \
    player_dim_df_subset[player_dim_df_subset["Casino Site"] == "Grand Casino HK"]
```

```
## Bring in some statistical packages to add a normal curve to the data set
```

```
from scipy.stats import norm
```

```
## Grab the first column we will use to plot
```

```
ml_casino_age = grand_casino_ml_player_dim["Age"]
hk_casino_age = grand_casino_hk_player_dim["Age"]
```

```
sns.set_style("white")
plt.figure(figsize= (10, 6))

## Plot Age Distribution for Grand Casino ML
plt.hist(ml_casino_age, 50, color = 'g')

## Set the xlabels to show more buckets to help identify things better
plt.xticks(np.arange(10, 110, step = 10))

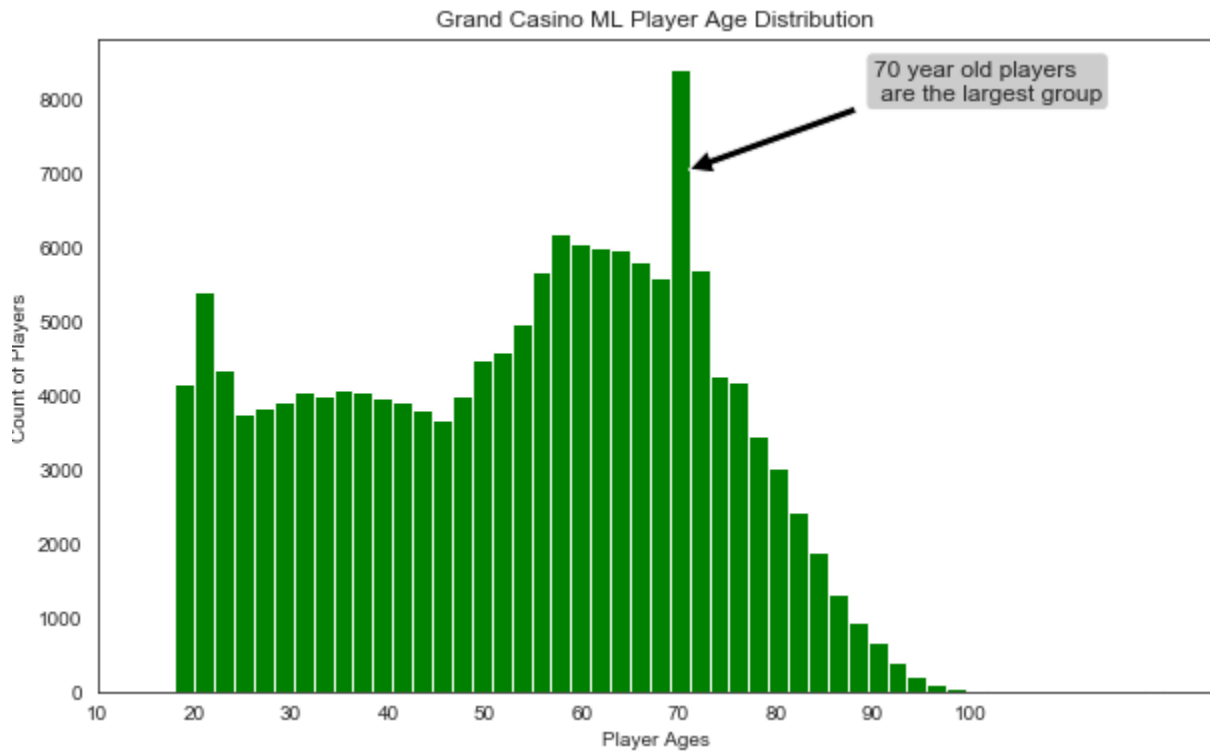
## Add more descriptive labels
plt.xlabel("Player Ages")
plt.ylabel("Count of Players")

plt.annotate("70 year old players\n are the largest group",
             xy = (70.0, 7000.0),
             xytext = (90.0, 8000.0),
             size = 12,
             bbox = dict(boxstyle="round", fc="0.8"),
             arrowprops=dict(facecolor="black", shrink = 0.05))

## Add a descriptive title
plt.title("Grand Casino ML Player Age Distribution")

## Show the beautiful graph!
plt.show()
```

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case



We pull in a special plot from the seaborn library known as **distplot**. This allows us to include a normal distribution curve overlaid with the histogram above. The y-axis is the probability density function for the kernel density estimation. However, this is the probability density and not a probability. It is helpful for relative comparison between the different buckets of Age.

```
## Import a normal distribution from scipy
from scipy.stats import norm

## Build a new plot
plt.figure(1, figsize=(10, 6))

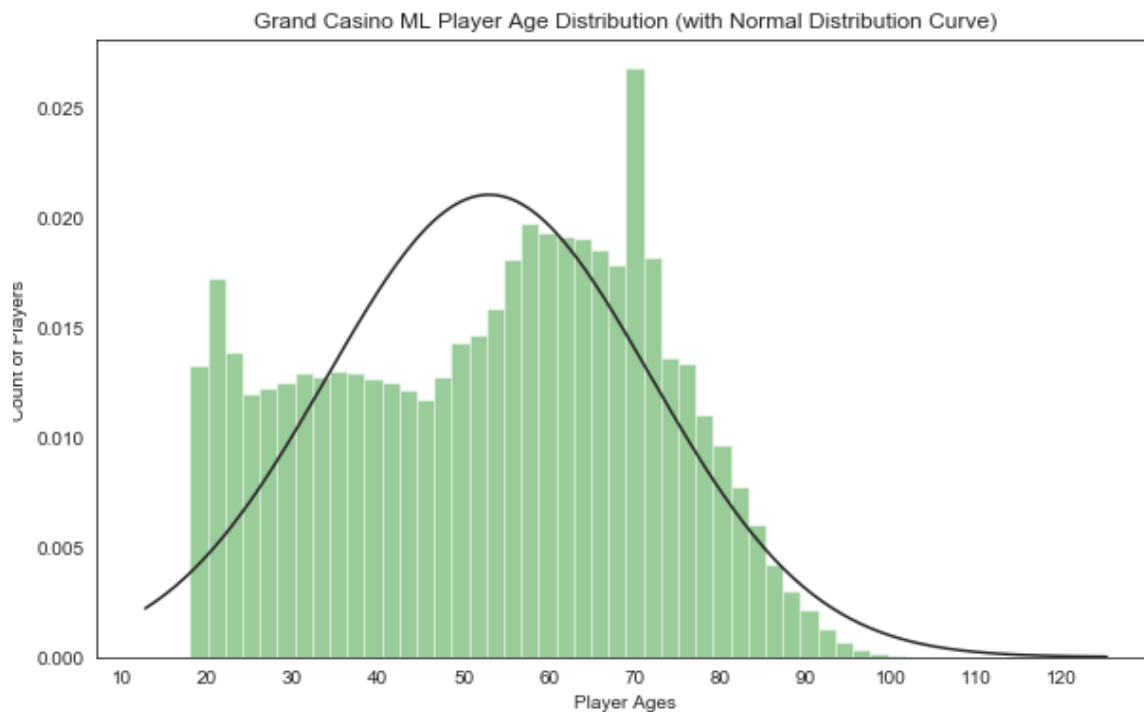
## Use a special function in seaborn to build a plot with the normal curve attached
sns.distplot(ml_casino_age, hist = True, fit = norm, kde = False, norm_hist = True,
             bins = 50, color = "green")

## Set the xlabels to show more buckets to help identify things better
plt.xticks(np.arange(10, 130, step = 10))

## Set better labels for the graph
plt.xlabel("Player Ages")
plt.ylabel("Count of Players")

## Set a title for the plot
plt.title("Grand Casino ML Player Age Distribution (with Normal Distribution Curve)")

## Show the plot
plt.show()
```



Examples for the other casino can be found in the ***Group1_MCLV_LiveCase_PlayerDemographics_PromotionalCampaigns_Assocation Rules_Visualizations.ipynb*** file.

Player Estimated Household Income Distribution

- Drop na/empty fields
- Only include income values that are digit or numeric

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

```
## Remove NA values from the income field as a first step to clean up
```

```
ml_casino_income = grand_casino_ml_player_dim["EstimatedHousehold Income"].dropna()
```

```
hk_casino_income = grand_casino_hk_player_dim["EstimatedHousehold Income"].dropna()
```

```
## In these steps, we're subsetting the data to only include Income values that are digit or numeric  
## We noticed that there were a lot of values that did not fit in that structure  
## We only want to plot legitimate values - so we subset them first and then convert them to integer
```

```
ml_casino_income = ml_casino_income[ml_casino_income.str.isdigit()]
```

```
ml_casino_income = ml_casino_income.astype(int)
```

```
hk_casino_income = hk_casino_income[hk_casino_income.str.isdigit()]
```

```
hk_casino_income = hk_casino_income.astype(int)
```

```
## Here we confirm that our transformation was successful
```

```
print(ml_casino_income.groupby(ml_casino_income).count().head())
```

```
print()
```

```
print(ml_casino_income.groupby(ml_casino_income).count().tail())
```

```
## Here we confirm that our transformation was successful
```

```
print(ml_casino_income.groupby(ml_casino_income).count().head())
```

```
print()
```

```
print(ml_casino_income.groupby(ml_casino_income).count().tail())
```

```
EstimatedHousehold Income
```

```
6      449
```

```
7      495
```

```
8      521
```

```
9      636
```

```
10     751
```

```
Name: EstimatedHousehold Income, dtype: int64
```

```
EstimatedHousehold Income
```

```
23647    1
```

```
24024    1
```

```
24286    1
```

```
24653    1
```

```
24981    1
```

```
Name: EstimatedHousehold Income, dtype: int64
```

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

```
## Create some buckets to better plot the distribution of estimated household income
## The spread of values is quite significant

slices = [15, 20, 30, 40, 50, 60, 70, 80, 100, 110, 120, 150, 200, 250, 300, 350, 400, 450, 500,
          550, 600, 650, 700, 750, 800, 850, 900, 950, 1000, 1100, 1300, 1500, 1700, 1900,
          2000, 3000, 4000, 5000, 6000, 25500]

## Save the sliced up income to a variable
income_dissected = pd.cut(ml_casino_income, slices)
```

(15, 20]	4052	(1000, 1100]	0
(20, 30]	7718	(1100, 1300]	0
(30, 40]	9267	(1300, 1500]	0
(40, 50]	10453	(1500, 1700]	0
(50, 60]	8021	(1700, 1900]	0
(60, 70]	12211	(1900, 2000]	0
(70, 80]	9077	(2000, 3000]	0
(80, 100]	14980	(3000, 4000]	0
(100, 110]	3464	(4000, 5000]	0
(110, 120]	6063	(5000, 6000]	1
(120, 150]	8202	(6000, 25500]	55
(150, 200]	4816		
(200, 250]	3551		

Name: EstimatedHousehold Income, dtype: int64

What an interesting observation on the incomes! There is a large gap between the household incomes below \$250K a year and above that line - and only 56 observations with household income above 6M a year.

So we will subset this dataset one more time for household incomes below \$250K.

```
## Subset the incomes below $250000

ml_casino_income_below_250 = ml_casino_income[ml_casino_income < 250]
hk_casino_income_below_250 = hk_casino_income[hk_casino_income < 250]
```

Now we're ready to generate the plots:

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

```
## Create a new plot figure
plt.figure(figsize= (14, 10))

## Plot Estimated Household Income Distribution for Grand Casino HK
plt.hist(ml_casino_income_below_250, 50, color = 'maroon')

plt.xticks(np.arange(10, 260, step = 10))

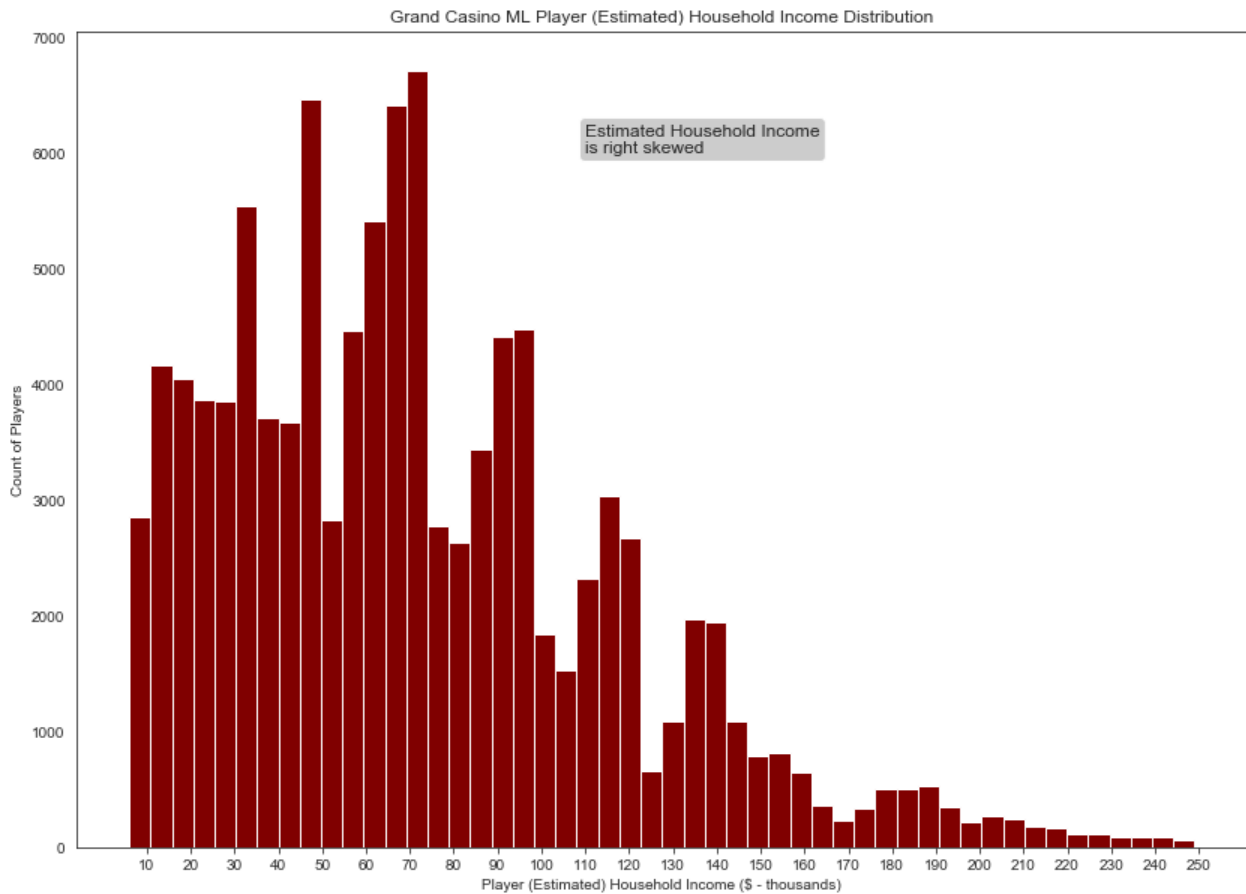
plt.tick_params(bottom = True)

## Set better labels for the graph
plt.xlabel("Player (Estimated) Household Income ($ - thousands)")
plt.ylabel("Count of Players")

plt.annotate("Estimated Household Income\nis right skewed",
            xy = (120.0, 6000),
            xytext = (110.0, 6000.0),
            size = 12,
            bbox = dict(boxstyle="round", fc="0.8"))

## Set a title for the plot
plt.title("Grand Casino ML Player (Estimated) Household Income Distribution")

## Show the plot
plt.show()
```



MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

```
## Display the same summary statistics

print()
print("The average (estimated) household income of a player in Grand Casino ML is ${},000.".format( \
    ml_casino_income_below_250.mean().astype(int)))
print()
print("The median (estimated) household income of a player in Grand Casino ML is ${},000.".format( \
    ml_casino_income_below_250.median().astype(int)))

The average (estimated) household income of a player in Grand Casino ML is $72,000.

The median (estimated) household income of a player in Grand Casino ML is $67,000.
```

Additional examples for the income distribution and for the other casino location can be found in the

Group1_MCLV_LiveCase_PlayerDemographics_PromotionalCampaigns_Assocation Rules_Visualizations.ipynb file.

Player and Tier Distribution

- Start with Python kernel
- Load the appropriate libraries
- Load the data files
- Begin the initial exploration
- Generate plots
- Assess and draw conclusions

We want to take a look at the distributions of players by their tier status to see if there's any patterns to be observed.

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

```
## Show the length of the original dataframes for the player dim and coupon fact
print(len(player_dim_df_subset))

print()

print(len(coupon_fact_df))
```

397620

2777358

```
## Confirm we don't have any "strange" Ages in the dataframe

player_ages = player_dim_df_subset.Age

print(min(player_ages))
print()

print(max(player_ages))

## Show the first few lines of the dataframe

player_dim_df_subset.head()
```

18

120

```
## Group the dataframe by Player Tier and Age to build a histogram

player_tier_age_count_df = \
    player_dim_df_subset.groupby(["Player Tier", "Age"]).agg({"Month": "count"}).reset_index()

## Rename our "count" column so that the table makes more sense
player_tier_age_count_df.rename(columns = {"Month": "Player Count"}, inplace = True)

## Sort by count to see what our biggest Player Tier and Age Group combinations are
player_tier_age_count_df.sort_values("Player Count", ascending = False, inplace = True)

## Show the first 5 rows of the new dataframe
player_tier_age_count_df.head(20)
```

	Player Tier	Age	Player Count
359	Preferred	59	7488
358	Preferred	58	7481
362	Preferred	62	7391

MSBA 6410 - Exploratory Analytics

Mille Lacs Corporate Ventures Live Case

```
## Sort the values of the dataframe by Age, to save us this step on all the subsequent dataframes
```

```
player_tier_age_count_df.sort_values(by = "Age", inplace = True)
```

```
## See what unique player tiers we have
```

```
player_tier_age_count_df["Player Tier"].unique()
```

```
array(['Preferred', 'Associate Gaming', 'Silver', 'Gold', 'Vendor 250',  
      'Vendor 40', 'AssociateTest', 'Platinum', 'Diamond', 'Admin'],  
      dtype=object)
```

```
## Save the tiers we want to overwrite to a List
```

```
tiers_to_replace = ["Associate Gaming", "Vendor 250", "Admin", "AssociateTest", "Vendor 40"]
```

```
## Use the dataframe replace function to overwrite the rows with our new category and then save over our original dataframe
```

```
player_tier_age_count_df = player_tier_age_count_df.replace(tiers_to_replace, "All Other Tiers")
```

Curiosity strikes again - how many players belong to each group? Are there any that may not provide any additional context or clarity?

```
## Create an aggregate dataframe by grouping by Player Tier - we then summarize the player count field
```

```
## We then sort the values by player count in descending order
```

```
player_tier_age_aggregate_df = \  
    player_tier_age_count_df.groupby("Player Tier").agg({"Player Count": "sum"}).reset_index(). \  
    sort_values(by = "Player Count", ascending = False)
```

```
## Create a summarization of the player count field, we will use to calculate the percent of the total for each
```

```
total_player_count = player_tier_age_aggregate_df["Player Count"].sum()
```

```
## Create two new columns
```

```
## Calculate the % of redemptions using the entire data set
```

```
player_tier_age_aggregate_df["% of Overall Total"] = \  
    player_tier_age_aggregate_df["Player Count"] / total_player_count * 100
```

```
## Calculate a running sum of the % to see which ones make up the largest portions
```

```
player_tier_age_aggregate_df["Cumulative % of Overall Total"] = \  
    player_tier_age_aggregate_df["% of Overall Total"].cumsum()
```

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

```
### Display a nice summary table
```

```
HTML(player_tier_age_aggregate_df.to_html(index = False))
```

Player Tier	Player Count	% of Overall Total	Cumulative % of Overall Total
Preferred	360752	90.727831	90.727831
Silver	19668	4.946431	95.674262
Gold	12375	3.112268	98.786530
All Other Tiers	2202	0.553795	99.340325
Platinum	1776	0.446658	99.786983
Diamond	847	0.213017	100.000000

We'll do one additional analysis, to bring in the redeemed coupon information.

```
## Replace each row that has a tier other than the top 7 and replace it with "All Other Tiers"  
## This will replace the values IN PLACE, so we only want to do this once - we won't be able to revert back without having to  
## recreate the dataframe from scratch.
```

```
## Save the tiers we want to overwrite to a list  
tiers_to_replace = ["Associate Gaming", "Vendor 250", "Admin", "AssociateTest", "Vendor 40"]
```

```
## Use the dataframe replace function to overwrite the rows with our new category and then save over our original dataframe  
player_tier_subset = player_tier_subset.replace(tiers_to_replace, "All Other Tiers")
```

```
## Merge our fact table with the player subset
```

```
merged_player_fact_df = coupon_fact_subset_df.merge(player_tier_subset, how = "left", on = "PlayerID")
```

```
merged_player_fact_aggregate_df = \  
    merged_player_fact_df.groupby("TierName").agg({"RedeemCount": "count"}).reset_index()\  
    .sort_values(by = "RedeemCount", ascending = False)
```

```
total_redeem_count = merged_player_fact_aggregate_df["RedeemCount"].sum()
```

```
total_redeem_count
```

```
2777241
```

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

```
## Create two new columns
## Calculate the % of redemptions using the entire data set
merged_player_fact_aggregate_df["% of Overall Total"] = \
    merged_player_fact_aggregate_df["RedeemCount"] / total_redeem_count * 100

## Calculate a running sum of the % to see which ones make up the largest portions
merged_player_fact_aggregate_df["Cumulative % of Overall Total"] = \
    merged_player_fact_aggregate_df["% of Overall Total"].cumsum()

HTML(merged_player_fact_aggregate_df.to_html(index = False))
```

TierName	RedeemCount	% of Overall Total	Cumulative % of Overall Total
Gold	880257	31.695377	31.695377
Preferred	861015	31.002531	62.697908
Silver	543839	19.581988	82.279896
Platinum	275793	9.930467	92.210363
Diamond	209097	7.528947	99.739310
All Other Tiers	7240	0.260690	100.000000

```
## Create a new List with the tiers we are going to focus on

tier_list = ["Preferred", "Silver", "Gold", "Associate Gaming", "Platinum", "Diamond", "All Other Tiers"]

## Create a new list to contain our dataframes needed for the visualization
tier_dfs = []

## Loop through the tiers we identified in our list above
for i in tier_list:
    ## Generate a new dataframe for each tier
    i_df = player_tier_age_count_df[player_tier_age_count_df["Player Tier"] == i]
    ## Append each dataframe to our list - we will pull these out in a second
    tier_dfs.append(i_df)
```

```
## Aggregate the table again now that we have new categories and save it to a new variable

player_tier_aggregate_df = \
    player_tier_age_count_df.groupby("Player Tier").agg({"Player Count": "sum"}).reset_index(). \
        sort_values(by = "Player Count", ascending = False)

## Reformat our table so that is more presentable
player_tier_aggregate_df["Player Count"] = player_tier_aggregate_df["Player Count"].map("{:,}".format)
```

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

```
## Show the new table!  
HTML(player_tier_aggregate_df.to_html(index = False))
```

Player Tier	Player Count
Preferred	360,752
Silver	19,668
Gold	12,375
All Other Tiers	2,202
Platinum	1,776
Diamond	847

```
## We're going to do a similar loop through this new list - extract the "X" and "Y" variables that we will need  
## for our visualization  
  
player_tier_x_vars = []  
player_tier_y_vars = []  
  
count = 0  
  
for i in tier_dfs:  
    player_tier_x_vars.append(i.Age)  
    player_tier_y_vars.append(i["Player Count"])
```

We generate a new graph - to demonstrate Player Tier by Age (All Tiers).

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

```
### Reset seaborn to the default background - for better viewing
sns.set_style("white")

## Plot scores on each epoch of our first model

plt.figure(1, figsize = (14, 10))

## Plot the first three type / subtype combinations
plt.plot(player_tier_x_vars[0], player_tier_y_vars[0], color = "black", lw = 2, label = "'Preferred' Tier")
plt.plot(player_tier_x_vars[1], player_tier_y_vars[1], color = "silver", lw = 2)
plt.plot(player_tier_x_vars[2], player_tier_y_vars[2], color = "gold", lw = 2)
plt.plot(player_tier_x_vars[3], player_tier_y_vars[3], color = "green",
         lw = 2)
plt.plot(player_tier_x_vars[4], player_tier_y_vars[4], color = "thistle", lw = 2)
plt.plot(player_tier_x_vars[5], player_tier_y_vars[5], color = "royalblue", lw = 2)

plt.plot(player_tier_x_vars[6], player_tier_y_vars[6], color = "red", lw = 2)

## Change the y-label for better descriptive text
## Set legend for first plot
## Set a descriptive title

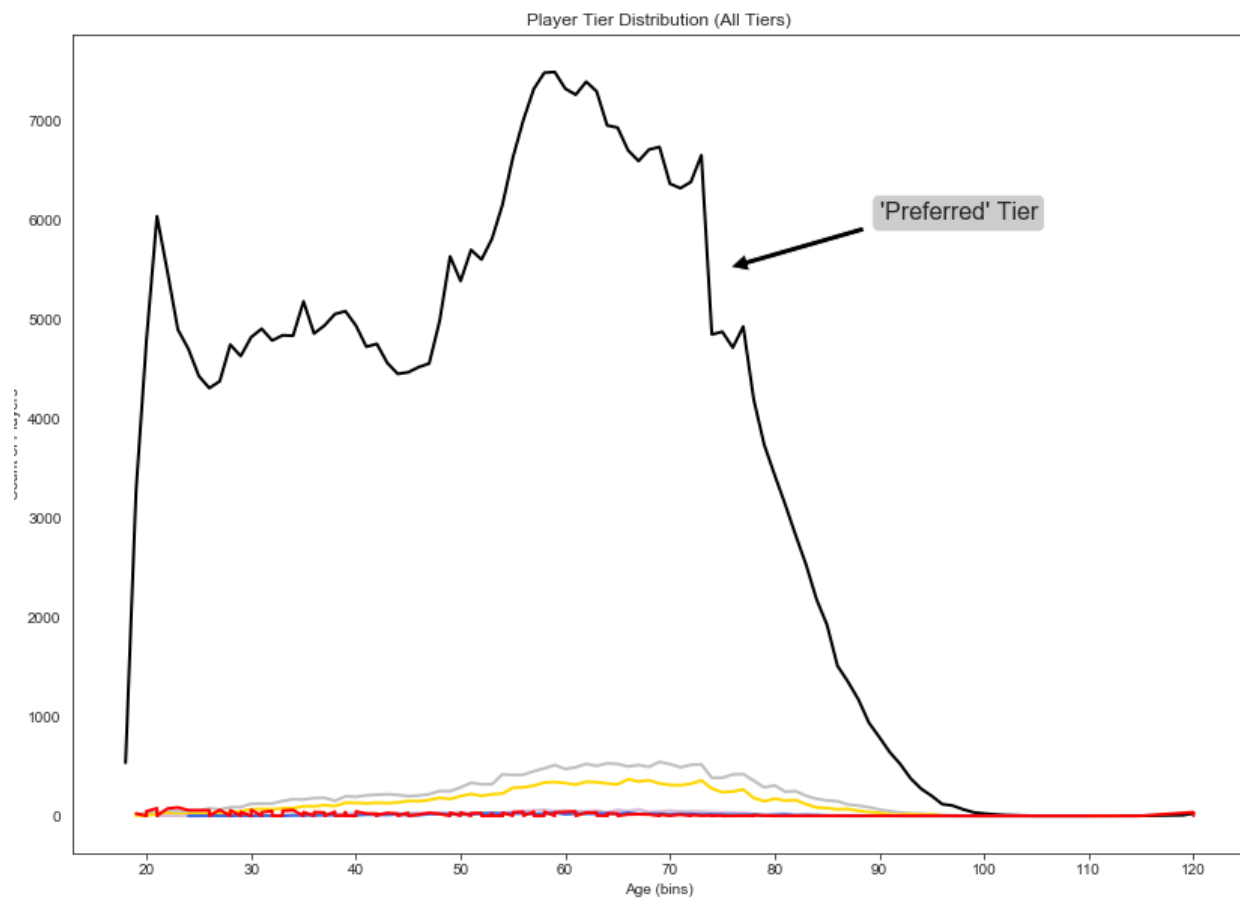
plt.xlabel("Age (bins)")
plt.ylabel("Count of Players")
plt.title("Player Tier Distribution (All Tiers)")
plt.xticks(np.arange(20, 130, step = 10))

plt.annotate("'Preferred' Tier",
            xy = (75.0, 5500.0),
            xytext = (90.0, 6000.0),
            size = 16,
            bbox = dict(boxstyle="round", fc="0.8"),
            arrowprops=dict(facecolor="black", shrink = 0.05))

ax = plt.gca()
ax.tick_params(bottom = True)

## Show the graph!
plt.show()
```


MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case



Examples for the other tiers can be found in the ***Group1_MCLV_LiveCase_PlayerDemographics_PromotionalCampaigns_Association Rules_Visualizations.ipynb*** file.

5.2 Geospatial & Location Demographics

Preparing the Zip Code Shapefile

To add a geospatial component to our analysis, we utilized the player zip codes found within *dimplayer.csv*. In addition to examining the raw counts of individuals within a given zip code, we integrated U.S. Census Bureau data from the American Community Survey. While this process would typically be performed using an API service, such as [ggmap](#), to abide by the security protocols placed on our virtual environment we developed our mapping architecture (i.e. shapefile) externally, and worked with the CSOM Help Desk to have our empty shapefile imported.

Working with QGIS

To download our base shapefile, we visited:

<https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-line-file.html>

While, the shapefile for [zip codes](#) is only available at the national level, the process of filtering the given shapefile in QGIS, a free and open-source cross-platform desktop geographic information system application, is relatively straightforward.

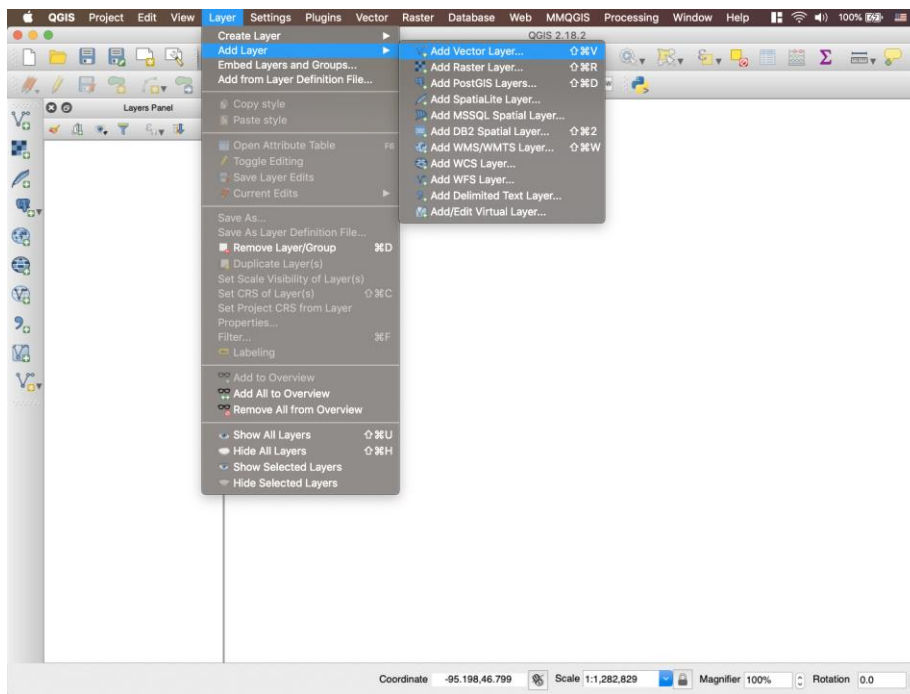
QGIS Steps

1. Download [QGIS](#). (Note: ArcGIS is a proprietary alternative).
2. Download the aforementioned zip code shapefile.

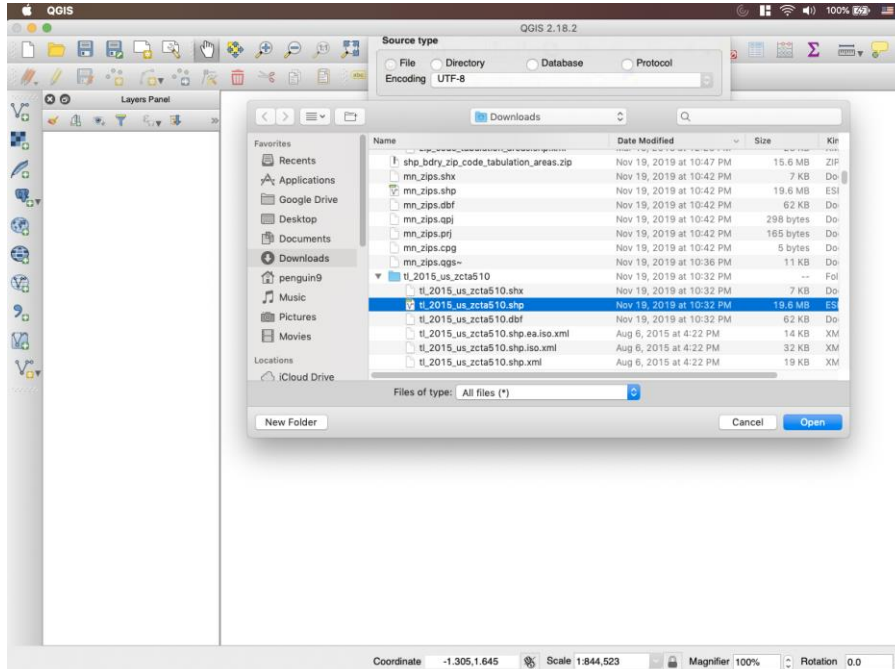


3. Open QGIS and import the zip code shapefile by (1) selecting *Layer*, (2) selecting *Add Layer*, and (3) selecting *Add Vector Layer...* in the top taskbar.

MSBA 6410 - Exploratory Analytics Mille Lacs Corporate Ventures Live Case

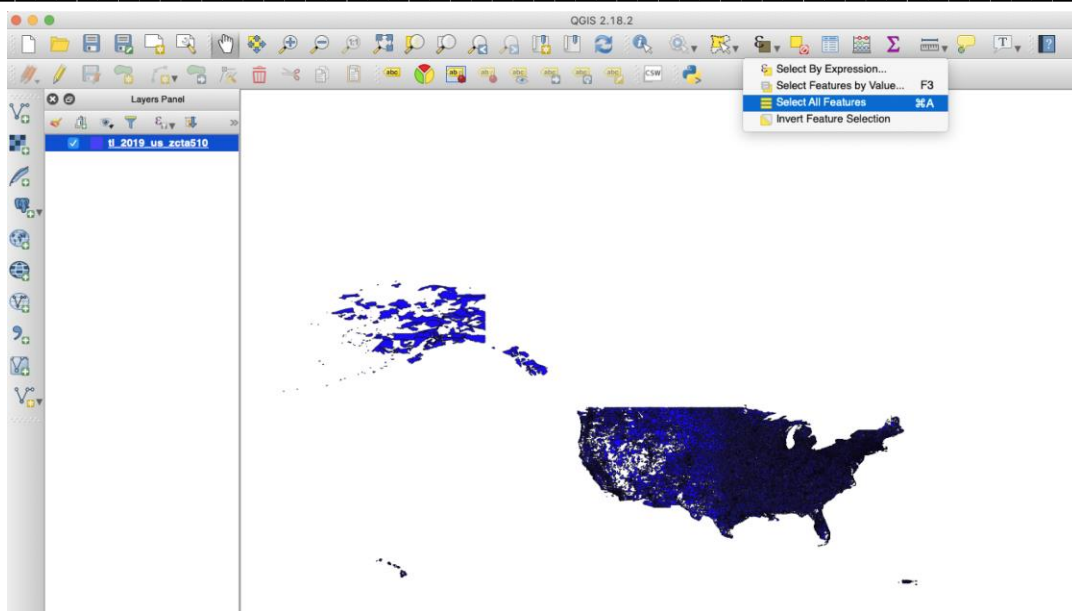


4. After you have selected *Add Vector Layer...*, navigate to the location of the appropriate shapefile.

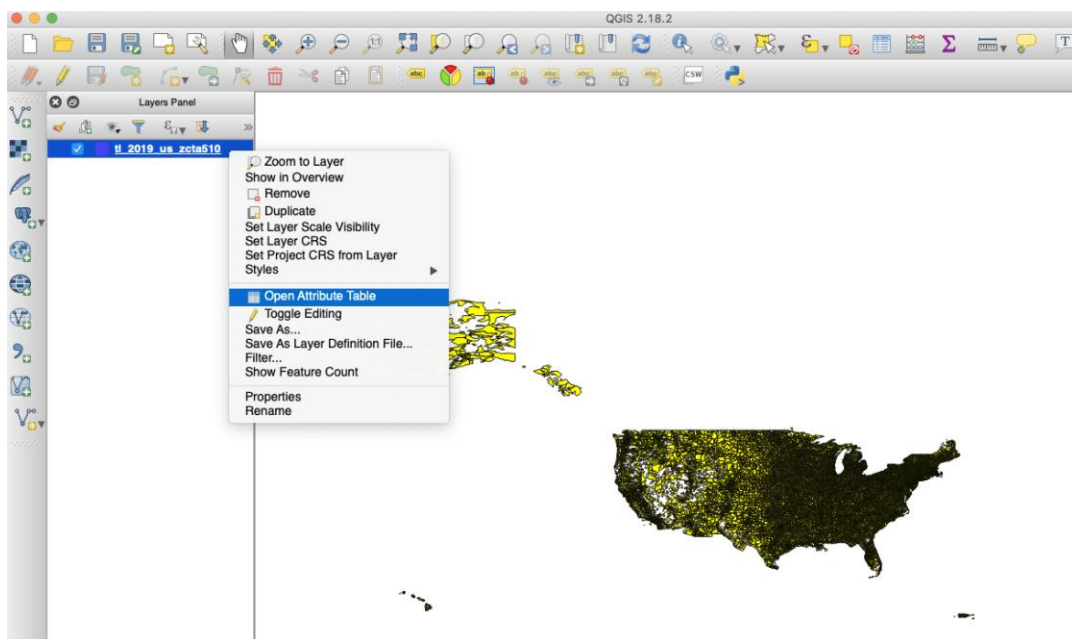


5. Within the tool bar navigate to the selection toggle (identified by the letter epsilon) and select *Select All Features*.

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

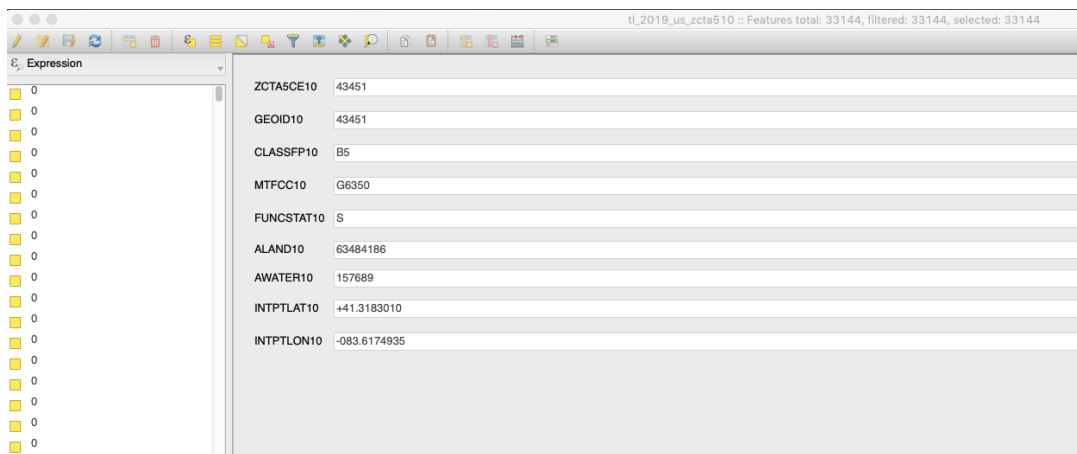


6. Navigate to the *Layers Panel* and right-click on *tl_2019_us_zcta510* to select *Open Attribute Table* from the dropdown menu.

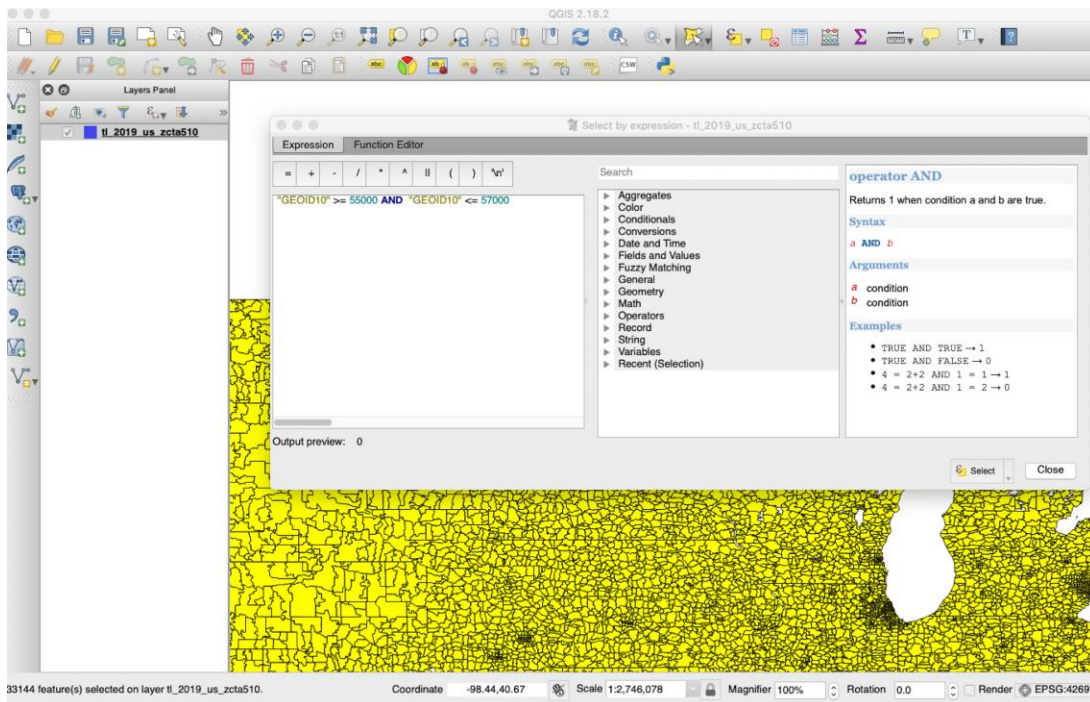


7. Navigate to *Select features using an attribute*. (Identified the letter epsilon).

MSBA 6410 - Exploratory Analytics Mille Lacs Corporate Ventures Live Case

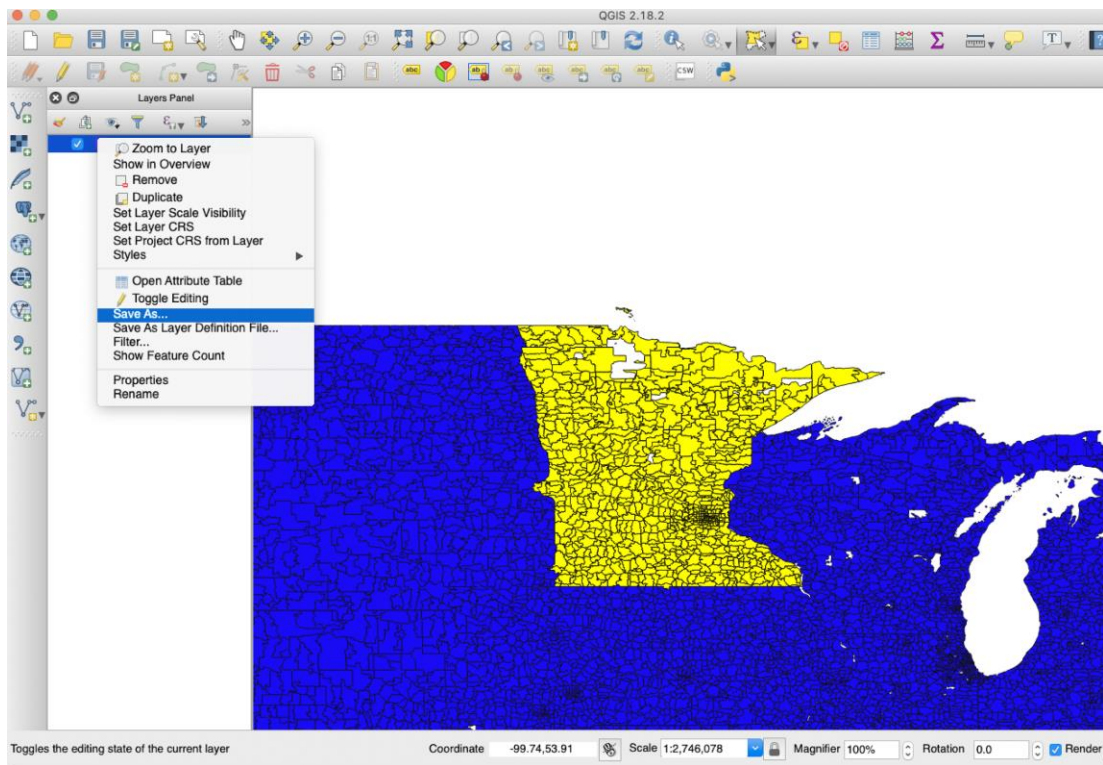


8. Within the expression field, enter "GEOID10" >55000 AND "GEOID10" < 57000 . This takes advantage of the fact that Minnesota's zip codes begin with a "55" or "56", exclusively, to invert our selections and isolate Minnesota alone.



9. Next, we return to the *Layers Panel* and right-click *tl_2019_us_zcta510* again. Within the dropdown select *Save As*.

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case



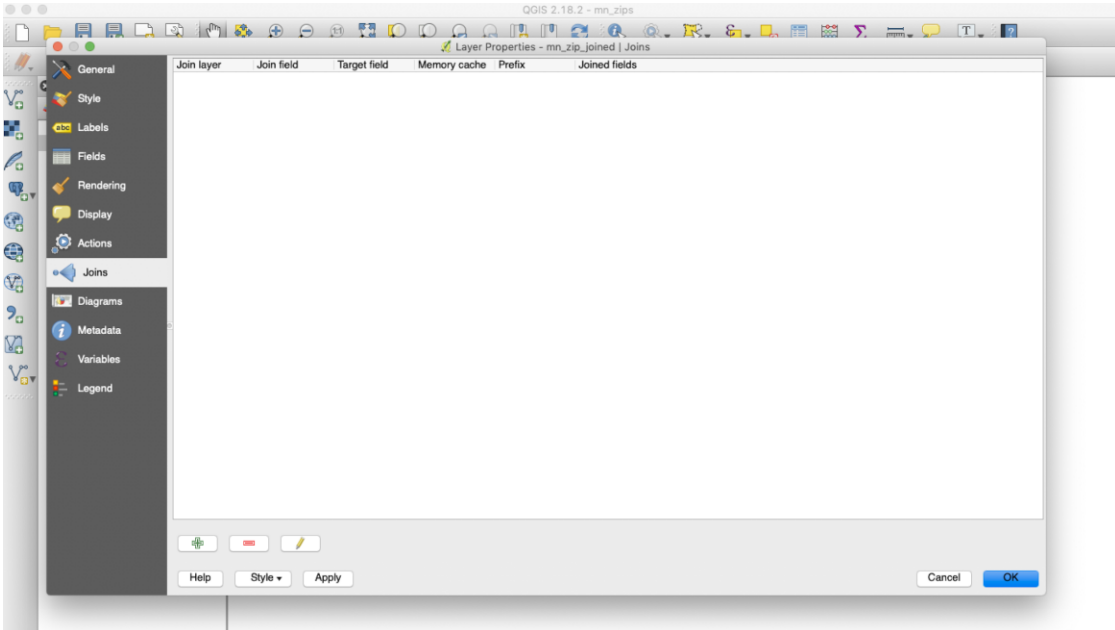
10. Within the next window, ensure you check *Save only selected features*. From there, select a file name and select okay. Congratulations, you've created the necessary shapefile.
11. For our next step, we will join some census data to our shapefile. To begin, you must download the appropriate file from <https://www.census.gov/programs-surveys/acs/technical-documentation/table-and-geography-changes/2017/5-year.html>

In our case, we are interested in obtaining ACS 17 5YR DP05..

12. Once we have obtained our file, we need to import it. To do that select the comma icon located to the left of the *Layers Panel*. Next, select browse, and select browse.

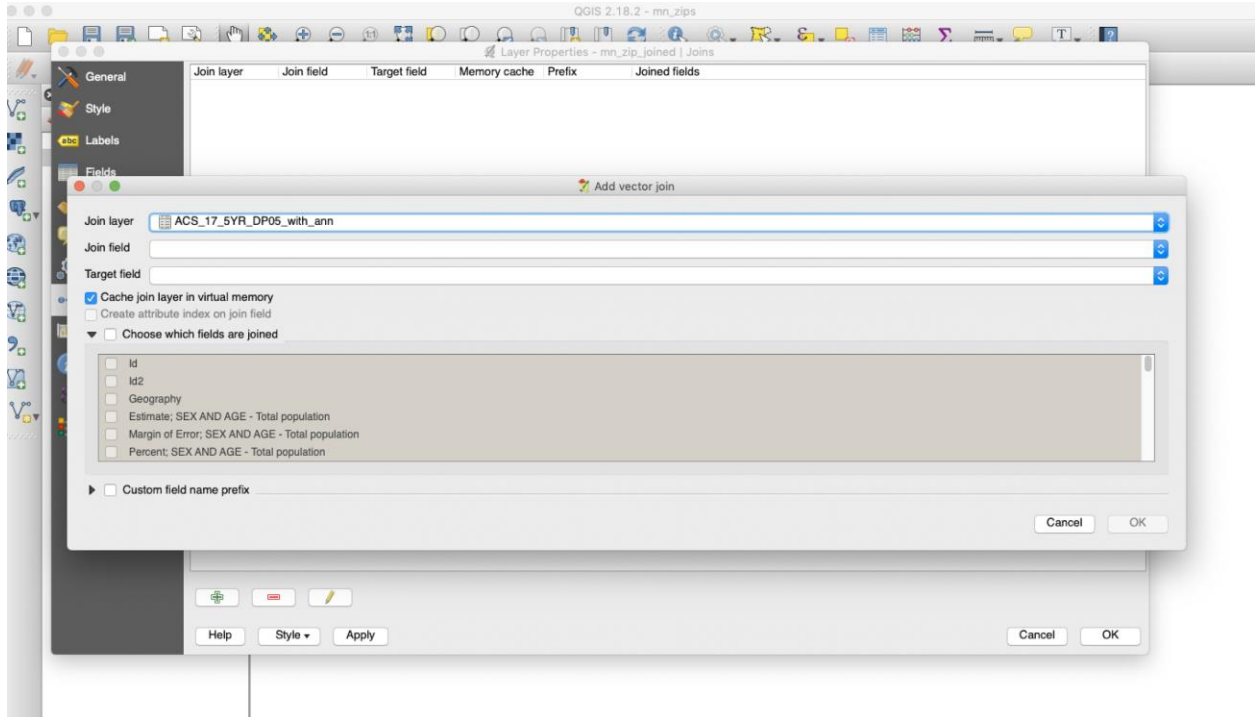


13. For the next step, remove the previous map from the *Layers Panel* and right-click on the name of your new Minnesota map and select properties from the dropdown. Next, select *Joins* from the left panel and select the green "+" button at the bottom of the window.



MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

14. Now, we select the layer that we would like to join, and select our *Join field* and *Target field*. In the dropdown, select *GEOID10* for both items. Once the join is complete, save your new shapefile and proceed to R to begin mapping.



Mapping in R

```
# Importing Necessary Libraries
library(tidyverse)
library(zipcode)
library(maps)
library(maptools)
library(RColorBrewer)
```

```
# Reading in Required Shapefile of U.S. Census Bureau Zip Code Tabulation Areas
map_shp <- readShapePoly("D://Group Folder/MLCV_Zip/MLCV_Zip/mn_zip_joined.shp")
# from https://catalog.data.gov/; initial data preprocessing performed using QGIS to
# filter data to include State: 027.
```

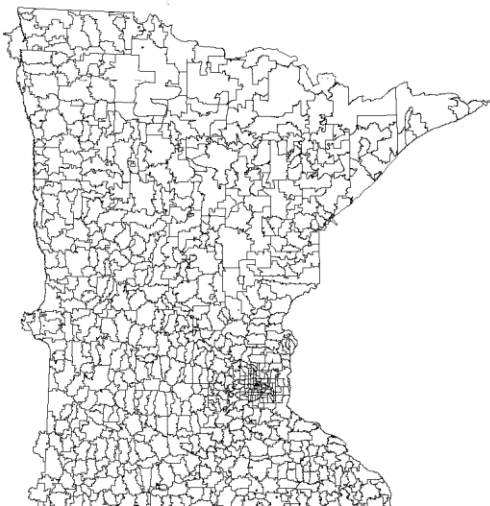
What are Zip Code Tabulation Areas (ZCTAs)? ZCTAs are generalized representations of a U.S. Postal Service Zip Code service areas, which have been revised to better align with the geographic boundaries used by the U.S. Census Bureau. Due to the resulting approximations, the results of our analysis should, in turn, be treated as an approximation. For more info, please refer to [census.gov/programs-surveys/geography/guidance/geo-areas/zctas.html](https://www.census.gov/programs-surveys/geography/guidance/geo-areas/zctas.html).

```
# Reviewing first 5 rows of data within the shapefile
head(map_shp@data)
```

	ZCTA5CE10 <chr>	GEOID10 <chr>	CLASSFP10 <chr>	MTECC10 <chr>	FUNCSTAT10 <chr>	ALAND10 <dbl>
0	56060	56060	B5	G6350	S	84864241
1	56062	56062	B5	G6350	S	325735997
2	56063	56063	B5	G6350	S	109203847
3	56065	56065	B5	G6350	S	301890699
4	56068	56068	B5	G6350	S	169949960

5 rows | 1-10 of 369 columns

```
plot(map_shp)
```



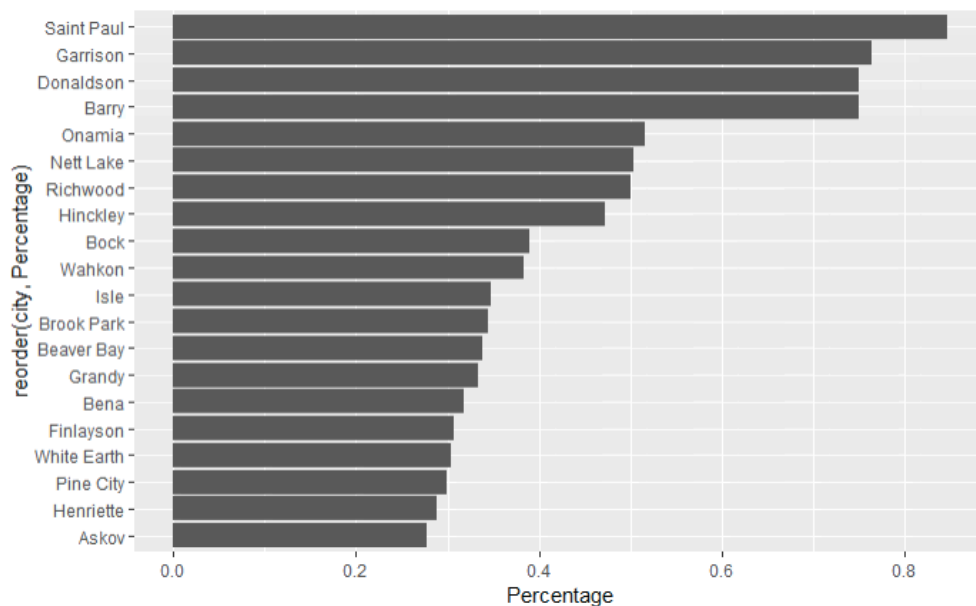
MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

```
# Importing Tabular Data

# Extracting national-level data from zipcode library
data(zipcode)
# Table of player characteristics
dim_player <- read.csv("D://Group Folder/Datasets/DimPlayer.csv")
# Isolating and counting zip codes
zip_data <- dim_player %>% count(ZipCode) %>% arrange(desc(n))
# Cleaning zip codes using the zipcode library
zip_data$ZipCode <- zipcode::clean.zipcodes(zip_data$ZipCode)
# Joining national-level data to zip code counts
df <- left_join(zip_data, zipcode, by=c("ZipCode"="zip"))
# Importing American Community Survey (ACS) 5-Year Estimates;
# Obtained from https://census.gov/programs-surveys/acs/
acs_data <- read.csv(paste0("D://Group Folder/MLCV_Zip/MLCV_Zip/",
                           "ACS_17_5YR_DP05_with_ann.csv"), skip = 1)[1:5]

# Changing dataframe names
names(acs_data) <- c("Id", "Id2", "Geography", "Estimate", "Margin")
# Ensuring IDs are interpreted as char rather than numeric
acs_data$Id2 <- as.character(acs_data$Id2)
# Joining ACS Data to zipcode/count dataframes
all_data <- left_join(df, acs_data, by=c("ZipCode"="Id2"))
# (Important Step) Standardizing zip code counts
all_data <- all_data %>% mutate(Percentage = n / Estimate)
# Safeguarding against duplicates resulting from joins
all_data <- all_data[complete.cases(all_data),]
# Eliminating Null/NA Joins
all_data <- all_data %>% filter(n > 1)
# Ensuring GEOIDs are interpreted as char rather than numeric
map_shp@data$GEOID10 <- as.character(map_shp@data$GEOID10)
# merging tabulated data to the shapefile for mapping
map_shp2 <- merge(map_shp, all_data, by.x="GEOID10", by.y="ZipCode")

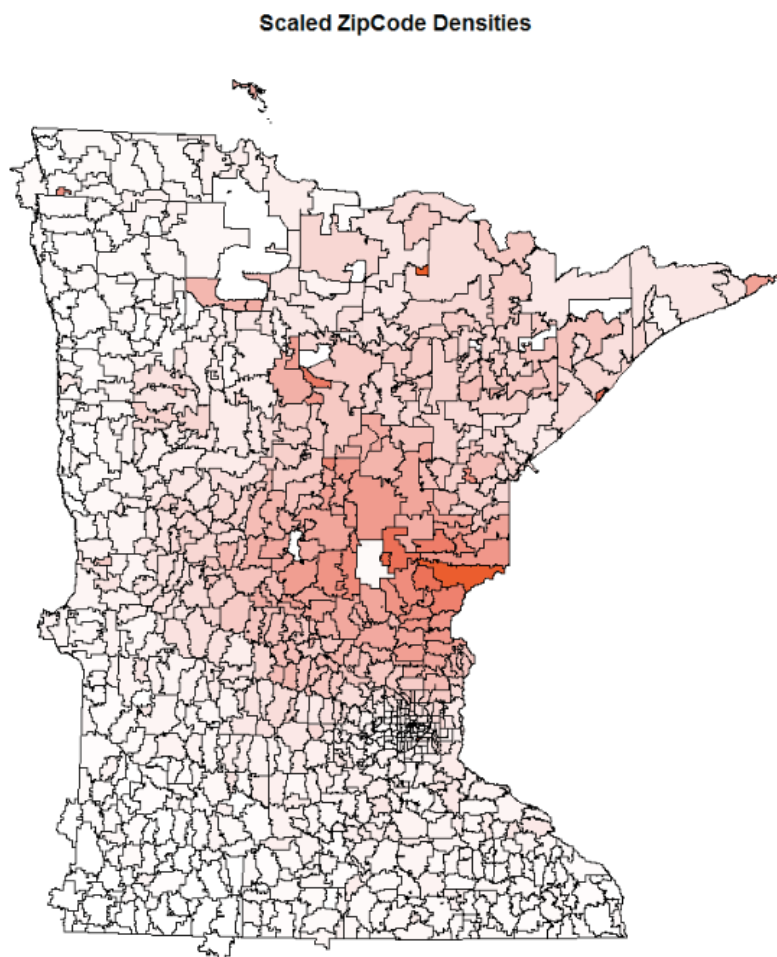
# Preliminary data analysis; zip codes per county
all_data_20 <- all_data %>% filter(Estimate > 0) %>% arrange(desc(Percentage)) %>% top_n(20)
ggplot(all_data_20, aes(x=reorder(city, Percentage), y=Percentage)) + geom_col() + coord_flip()
ggplot(all_data_20, aes(x=reorder(ZipCode, Percentage), y=Percentage)) + geom_col() + coord_flip()
```



Plotting zip codes with standardization

```
# Creating a color palette and assigning weights
p <- colorRampPalette((c("white", "red")))(100)
palette(p)
pop <- map_shp2@data$Percentage * 200

# plotting standardized zip code counts (Important Figure)
plot(map_shp, col = pop)
# adding titles
title(main="Scaled ZipCode Densities")
```

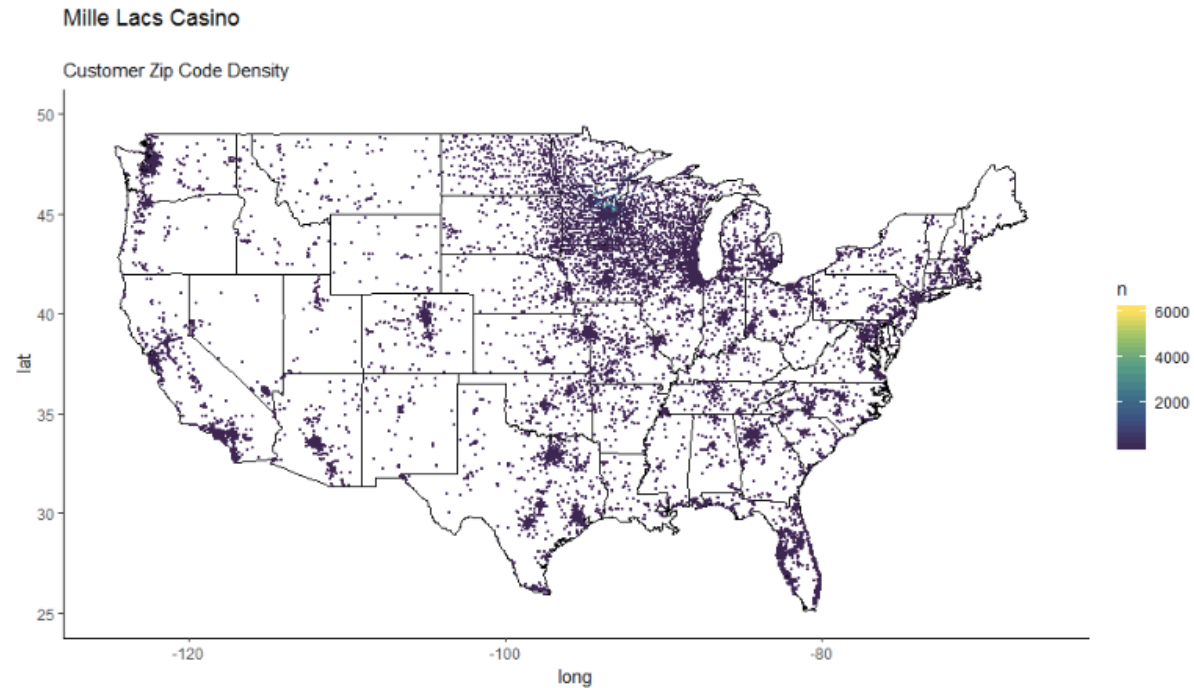


Plotting national zip codes without standardization

```
# Preparing necessary the underlying maps
county <- map_data("county", region = "minnesota")
mn <- map_data("state", region = "minnesota")
us <- map_data("state")
```

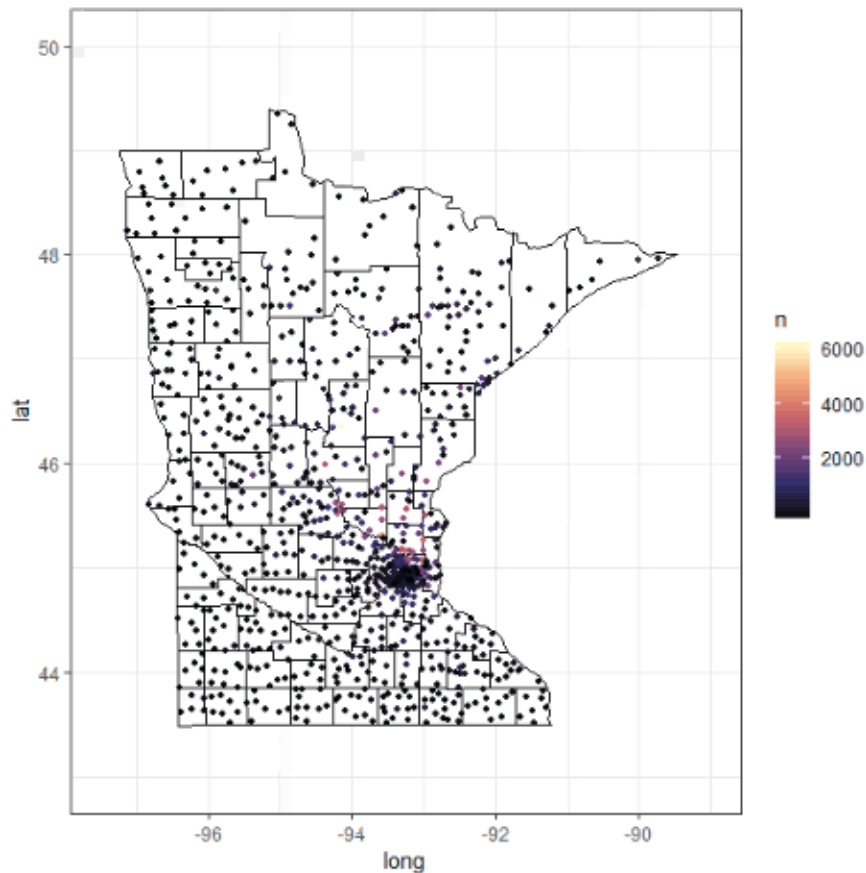
MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

```
# Nation-wide plot of zip code densities
ggplot(us) +
  geom_polygon(aes(x=long, y=lat, group = group), fill="white", color="black") +
  coord_quickmap() +
  stat_density2d(aes(x=longitude, y=latitude, fill=..level.., alpha=..level..),
    bins=9, geom="polygon", data=df) +
  scale_fill_gradient(low="black", high="red") +
  theme_bw() + theme(legend.position = 'none') +
  labs(title = "Mille Lacs Casino\n", subtitle = "Customer Zip Code Density")
```



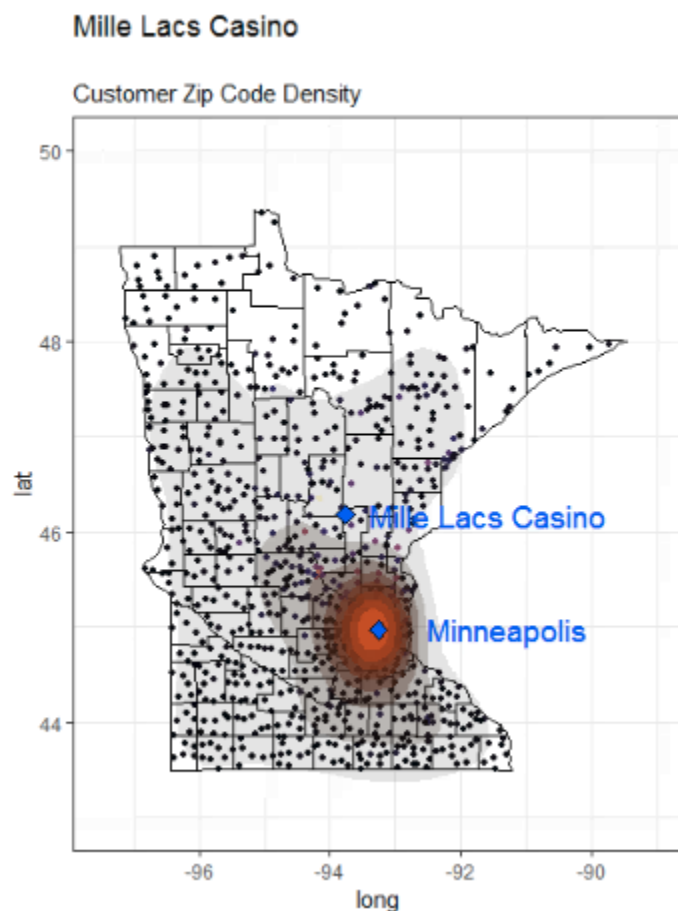
```
# Plotting nation-wide zip codes with density gradients
geom_polygon(aes(x=long, y=lat, group = group), fill="white", color="black") +
coord_quickmap() +
geom_point(aes(x=longitude, y=latitude, color=n), size = 0.2, data=df) +
scale_x_continuous(limits = c(-125, -66)) +
scale_y_continuous(limits = c(25, 50)) +
scale_color_viridis_c() +
scale_fill_gradient(low="black", high="red") +
theme_classic() +
labs(title = "Mille Lacs Casino\n", subtitle = "Customer Zip Code Density")
```

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case



```
# Plotting state-wide zip code densities with reference locations
mpls <- data.frame(longitude=c(-93.2650), latitude=c(44.9778), label=c("MPLS"))
mlc <- data.frame(longitude=c(-93.7591), latitude=c(46.1799), label=c("MLC"))
df_mn <- df %>% filter(state=="MN")

# Mapping Zip Codes
ggplot(county) +
  geom_polygon(aes(x=long, y=lat, group=group), fill="white", color="black") +
  coord_quickmap() +
  geom_point(aes(x=longitude, y=latitude, color=n), size=1, data=df_mn) +
  scale_x_continuous(limits = c(-97.5, -89)) +
  scale_y_continuous(limits = c(43, 50)) +
  scale_color_viridis_c(option="magma") +
  scale_fill_gradient(low="black", high="red") +
  stat_density2d(aes(x=longitude, y=latitude, fill=..level.., alpha=..level..),
    bins=9, geom="polygon", data=df_mn) +
  geom_point(aes(x=longitude, y=latitude), size=3, fill="blue", shape=23, data=mpls) +
  geom_text(data=mpls, aes(x=longitude, y=latitude), color="blue", size=5,
    label="Minneapolis", hjust=-.3) +
  geom_point(aes(x=longitude, y=latitude), size=3, fill="blue", shape=23, data=mlc) +
  geom_text(data=mlc, aes(x=longitude, y=latitude), color="blue", size=5,
    label="Mille Lacs Casino", hjust=-.1) +
  theme_bw() + theme(legend.position = 'none') +
  labs(title = "Mille Lacs Casino\n", subtitle = "Customer Zip Code Density")
```

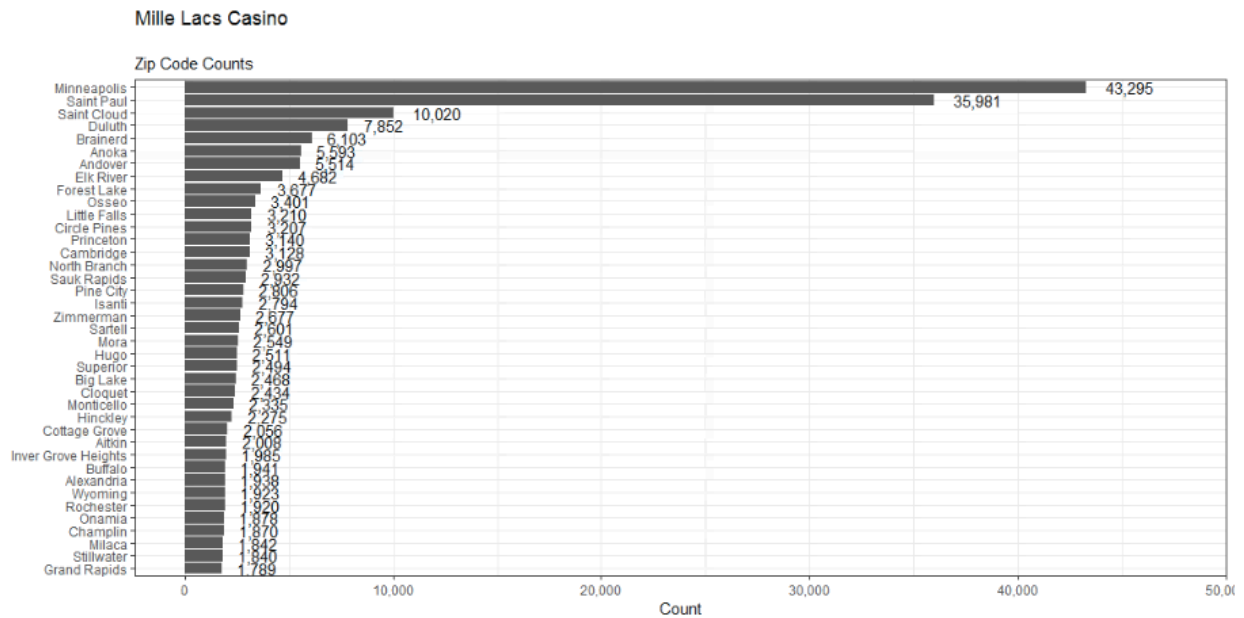


State-level Descriptive Statistics

```
# Zip code counts by city
df_temp <- df %>%
  group_by(city) %>%
  summarise(total_n = sum(n)) %>%
  arrange(desc(total_n)) %>%
  top_n(40) %>% filter(!is.na(city))

ggplot(df_temp, aes(x=reorder(city, total_n), y=total_n)) +
  geom_col() + coord_flip() +
  geom_text(label=scales::comma(df_temp$total_n), hjust=-.4) +
  scale_y_continuous(limits = c(0, 1.1 * max(df_temp$total_n)), labels = scales::comma) +
  labs(title="Mille Lacs Casino\n", subtitle="Zip Code Counts", x="", y="Count") +
  theme_bw()
```

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case



5.3 Effectiveness of Promotional Campaigns

- Start with Python kernel
- Load the appropriate libraries
- Load the data files
- Begin the initial exploration
- Generate plots
- Assess and draw conclusions

We cover the pre-processing steps in an earlier section.

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

```
## View the first five rows of the table
```

```
coupon_fact_df.head(5)
```

	CouponID	Description	PlayerID	AccountingDate	RedeemValue	RedeemCount	SiteID	RedemptionNumber	V
0	89729	M1216VHNY-VIP NYE HTL	530342410	2017-01-01	29.0	1	1	NaN	
1	89729	M1216VHNY-VIP NYE HTL	710328524	2017-01-01	29.0	1	1	NaN	
2	89093	C1016OGDO1E-Decline GP WKND 5	1000600137	2017-01-01	5.0	1	2	NaN	
3	89729	M1216VHNY-VIP NYE HTL	710004002	2017-01-01	29.0	1	1	NaN	
4	4	HNEW MEMBER TRACKING	1000681589	2017-01-01	0.0	1	2	NaN	

5 rows x 21 columns

```
## Determine if there are any records in the fact table with 0 in the "RedeemCount" field
## Anything with zero means it did not get redeemed and should be excluded
```

```
coupon_fact_df.RedeeCount.unique()
```

```
array([ 1,  2,  0,  4,  3,  5, 10,  8,  6,  9], dtype=int64)
```

```
## Remove those rows with nothing in the RedeemCount - so we just look at redeemed coupons
```

```
redeemed_coupon_df = coupon_fact_df[coupon_fact_df.RedeeCount != 0]
```

```
## Create some variables to show the length of the two facts, the original and the new one.
```

```
original_fact_row_count = len(coupon_fact_df)
```

```
reduced_fact_row_count = len(redeemed_coupon_df)
```

```
print("We removed {} rows by dropping any coupons with a redeem count of 0."\
      .format(np.abs(reduced_fact_row_count - original_fact_row_count)))
```

We removed 227 rows by dropping any coupons with a redeem count of 0.

MSBA 6410 - Exploratory Analytics

Mille Lacs Corporate Ventures Live Case

```
redeemed_coupon_type_analysis = redeemed_coupon_df.groupby(["TypeDescription",
    "SubTypeDescription"]) \
    .agg({"RedeemCount": ["sum"]}).reset_index() ## Only sum is required, the other metrics
    ## do not provide any additional meaning
    ## or context.
```

```
## Rename the columns so they are more intuitive for the analyst
```

```
redeemed_coupon_type_analysis.columns = ["Coupon Type", "Coupon Subtype", "Redemption Count"]
```

```
## Get the first 20 promotion type / sub type combinations based on the total number of redemptions
```

```
top_twenty_redeemed_promotion_types = \
    redeemed_coupon_type_analysis.sort_values(by = "Redemption Count", ascending = False).head(20)
```

```
## Create two new columns
```

```
## Calculate the % of redemptions using the entire data set
```

```
top_twenty_redeemed_promotion_types["% of Overall Total"] = \
    top_twenty_redeemed_promotion_types["Redemption Count"] / total_redemptions * 100
```

```
## Calculate a running sum of the % to see which ones make up the largest portions
```

```
top_twenty_redeemed_promotion_types["Cumulative % of Overall Total"] = \
    top_twenty_redeemed_promotion_types["% of Overall Total"].cumsum()
```

```
## Display an nice presentation of the data table
```

```
HTML(top_twenty_redeemed_promotion_types.to_html(index = False))
```

Coupon Type	Coupon Subtype	Redemption Count	% of Overall Total	Cumulative % of Overall Total
Cash	Bonus	688847	24.700808	24.700808
Hotel	All	638501	22.895492	47.596300
Free Slot Gaming	Grand Play	290835	10.428817	58.025117
Cash	Mystery	290224	10.406908	68.432025
Cash	Cash	187025	6.706378	75.138404
Hotel	HK	135938	4.874491	80.012895
Other	Other	116120	4.163853	84.176748
Gift	Other	84195	3.019081	87.195828
Points	Match Point	79035	2.834052	90.029881
Hotel	ML	63634	2.281800	92.311681
POS	Food	63421	2.274162	94.585843
Free Slot Gaming	Bonus	32627	1.169945	95.755788
Free Table Gaming	Pull-tabs	30503	1.093782	96.849571

We see that these coupon type & sub type combinations

- Cash - Bonus
- Hotel - All

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

- Free Slot Gaming - Grand Play
- Cash - Mystery
- Cash - Cash

make up 75% of redeemed coupons. How do these change over time? Let's return to our original dataframe for this analysis, where we had observations over time. Let's do some additional clean up here, to rename some columns and only include what we need for the time series analysis.

```
## Rename the column names so they are more understandable
redeemed_coupon_aggregation.columns = ["Month", "Year", "Casino Site", "Coupon Type",
                                       "Coupon Sub Type", "Redemption Count", "Remove Count",
                                       "Remove Mean", "Remove Min", "Remove Max"]

## Subset the dataframe for the columns that we required for the next analysis
## We are overwriting the original variable name with this transformation - we have to be sure that we want this
## change!
redeemed_coupon_aggregation = \
    redeemed_coupon_aggregation[["Month", "Year", "Casino Site", "Coupon Type",
                                "Coupon Sub Type", "Redemption Count"]]
```

```
## Now the table is ready for the next analysis!
```

```
redeemed_coupon_aggregation.head()
```

	Month	Year	Casino Site	Coupon Type	Coupon Sub Type	Redemption Count
0	1	2017	1	Cash	Bonus	12919
1	1	2017	1	Cash	Cash	6753
2	1	2017	1	Cash	Mystery	3088
3	1	2017	1	Event	Concert	2
4	1	2017	1	Free Slot Gaming	Bonus	791

As above, we want to just focus on the five combinations to see if there are any seasonal shifts in redemption. These might coincide with the headcount fluctuations at the casino.

```
## Create a key column - to save a little bit of time subsetting the data on these five combinations
```

```
redeemed_coupon_aggregation["Coupon Type / Subtype"] = \
    redeemed_coupon_aggregation["Coupon Type"] + " - " + redeemed_coupon_aggregation["Coupon Sub Type"]
```

```
## Create a list of the five combinations to subset
combinations = ["Cash - Bonus", "Cash - Cash", "Cash - Mystery",
               "Free Slot Gaming - Grand Play", "Hotel - All"]
```

```
top_five_coupon_combinations = \
    redeemed_coupon_aggregation[redeemed_coupon_aggregation["Coupon Type / Subtype"].isin(combinations)]
```

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

```
## Confirm transformation completed as expected
```

```
top_five_coupon_combinations.head()
```

	Month	Year	Casino Site	Coupon Type	Coupon Sub Type	Redemption Count	Coupon Type / Subtype
0	1	2017	1	Cash	Bonus	12919	Cash - Bonus
1	1	2017	1	Cash	Cash	6753	Cash - Cash
2	1	2017	1	Cash	Mystery	3088	Cash - Mystery
5	1	2017	1	Free Slot Gaming	Grand Play	4977	Free Slot Gaming - Grand Play
8	1	2017	1	Hotel	All	7857	Hotel - All

Now we build a very intuitive analysis utilizing time series information. How do the redemption counts fluctuate through time? For that, we bring in our favorite friend for visualizing, `seaborn`!

```
## Build a pivot table so that each combination is on its own line - this should allow us to plot it over time

top_five_coupon_pivot_table = \
    pd.pivot_table(top_five_coupon_combinations, values = "Redemption Count", index = ["Month", "Year"],
                    columns = ["Coupon Type / Subtype"], aggfunc = np.sum).reset_index()
```

```
## Show the first five rows of the pivotted table
```

```
top_five_coupon_pivot_table.head()
```

Coupon Type / Subtype	Month	Year	Cash - Bonus	Cash - Cash	Cash - Mystery	Free Slot Gaming - Grand Play	Hotel - All
	0	1 2017	36923	16235	7395	14738	21909
	1	1 2018	28412	10218	5697	2290	18628
	2	1 2019	13116	2259	5104	2545	12999
	3	2 2017	34935	13453	8902	15452	22455
	4	2 2018	10422	3550	3376	2378	17561

```
## A few more steps - we'll create a month label with the actual month name
```

```
top_five_coupon_pivot_table["Month (name)"] = \
    top_five_coupon_pivot_table["Month"].apply(lambda x: calendar.month_name[x])
```

```
## Combine month and year, then sort the table by year/month
```

```
top_five_coupon_pivot_table["Month & Year"] = top_five_coupon_pivot_table["Month (name)"] + " " + \
    top_five_coupon_pivot_table["Year"].astype(str)
```

```
## Sort the values by calendar year - and we should be done!
```

```
top_five_coupon_pivot_table.sort_values(by = ["Year", "Month"], inplace = True)
```

MSBA 6410 - Exploratory Analytics

Mille Lacs Corporate Ventures Live Case

```
## A few more steps - we'll create a month label with the actual month name

top_five_coupon_pivot_table["Month (name)"] = \
    top_five_coupon_pivot_table["Month"].apply(lambda x: calendar.month_name[x])

## Combine month and year, then sort the table by year/month
top_five_coupon_pivot_table["Month & Year"] = top_five_coupon_pivot_table["Month (name)"] + " " + \
    top_five_coupon_pivot_table["Year"].astype(str)

## Sort the values by calendar year - and we should be done!
top_five_coupon_pivot_table.sort_values(by = ["Year", "Month"], inplace = True)

## Show the first five rows again following the transformations above to confirm the new data frame
## is in the right structure and ready for our plots

top_five_coupon_pivot_table.head()
```

Coupon Type / Subtype	Month	Year	Cash - Bonus	Cash - Cash	Cash - Mystery	Free Slot Gaming - Grand Play	Hotel - All	Month (name)	Month & Year
0	1	2017	36923	16235	7395	14738	21909	January	January 2017
3	2	2017	34935	13453	8902	15452	22455	February	February 2017
6	3	2017	25034	10327	11198	13574	26705	March	March 2017
9	4	2017	19901	1448	9613	16018	25783	April	April 2017
12	5	2017	20501	1466	15313	21710	26321	May	May 2017

```
## Create a quick function to put all of the columns we want to plot into variables
## Separate each coupon type/subtype combination

y1 = top_five_coupon_pivot_table["Cash - Bonus"]
y2 = top_five_coupon_pivot_table["Cash - Cash"]
y3 = top_five_coupon_pivot_table["Cash - Mystery"]
y4 = top_five_coupon_pivot_table["Free Slot Gaming - Grand Play"]
y5 = top_five_coupon_pivot_table["Hotel - All"]

## Use the month and year for our x-axis
x = top_five_coupon_pivot_table["Month & Year"]
```

```
### Reset seaborn to the default background - for better viewing
sns.set_style("darkgrid")

## Generate a new figure object
plt.figure(1, figsize = (10, 6))

## Plot the first type / subtype combination
plt.plot(x, y1, 'g-o', label = "Cash - Bonus")

## Change the y-label for better descriptive text
plt.ylabel("Total Redeemed Coupons (Monthly)")

## Set a descriptive title
plt.title(" 'Cash - Bonus' Total Redeemed Coupons by Month and Year")
## Set legend for first plot
plt.legend()

## Create an axes object for the next step
ax = plt.gca()

every_nth = 3
## Loop through the labels and only show every 4 months for easier readability
for n, label in enumerate(ax.xaxis.get_ticklabels()):
    label.set_rotation(45)
    if n % every_nth != 0:
        label.set_visible(False)
ax.tick_params(bottom = True)

## Show the graph!
plt.show()
```

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case



Examples for the other coupon type & subtype combinations can be found in the ***Group1_MCLV_LiveCase_PlayerDemographics_PromotionalCampaigns_Assocation Rules_Visualizations.ipynb*** file.

- What to make of this? We see that cash rewards tend to follow the same pattern, and had a huge spike in redemptions starting in early 2017 and then coming right back down towards the end of the year. However, none of those rewards reach the same level of activity again - what caused them to be so popular during that time frame?
- These seasonal peaks generated an additional question for our group - does this happen across ALL coupons? Do we see this seasonality fluctuation with ALL redeemed coupons, regardless of type?

MSBA 6410 - Exploratory Analytics Mille Lacs Corporate Ventures Live Case

```
## We come back to our original aggregation table - we can simplify this one a bit by doing another group by
## on month and year and only selecting the Count field - irregardless of the coupon type
```

```
redeemed_coupon_aggregation.head()
```

	Month	Year	Casino Site	Coupon Type	Coupon Sub Type	Redemption Count	Coupon Type / Subtype
0	1	2017	1	Cash	Bonus	12919	Cash - Bonus
1	1	2017	1	Cash	Cash	6753	Cash - Cash
2	1	2017	1	Cash	Mystery	3088	Cash - Mystery
3	1	2017	1	Event	Concert	2	Event - Concert
4	1	2017	1	Free Slot Gaming	Bonus	791	Free Slot Gaming - Bonus

```
## Start with making a copy of the dataframe, to not overwrite the original aggregation table
## This is best practice to ensure that this can be run from any environment/machine
```

```
all_coupon_redeemed_by_month_year_df = redeemed_coupon_aggregation.copy()
```

```
## Confirm dataframe length and head match the original and show the copied table
```

```
print("There are this many rows in the original dataframe: {}".format(len(redeemed_coupon_aggregation)))
print()
```

```
print("There are this many rows in the new dataframe: {}".format(len(all_coupon_redeemed_by_month_year_df)))
print()
```

```
all_coupon_redeemed_by_month_year_df.head()
```

There are this many rows in the original dataframe: 987.

There are this many rows in the new dataframe: 987.

	Month	Year	Casino Site	Coupon Type	Coupon Sub Type	Redemption Count	Coupon Type / Subtype
	1	2017	1	Cash	Bonus	12919	Cash - Bonus
	1	2017	1	Cash	Cash	6753	Cash - Cash
	1	2017	1	Cash	Mystery	3088	Cash - Mystery
	1	2017	1	Event	Concert	2	Event - Concert
	1	2017	1	Free Slot Gaming	Bonus	791	Free Slot Gaming - Bonus

MSBA 6410 - Exploratory Analytics

Mille Lacs Corporate Ventures Live Case

```
## We start by renaming the columns to remove the original formatting from aggregating them together
## Then we drop the columns that we don't need

all_coupon_redeemed_by_month_year_df.columns = ["Month", "Year", "Casino", "Coupon Type", "Coupon Sub Type",
                                                "Coupons Redeemed (Count)", "Type / Subtype"]

all_coupon_redeemed_by_month_year_df = all_coupon_redeemed_by_month_year_df[["Month",
                                     "Year", "Coupons Redeemed (Count)"]]

## Confirm that our transformations completed successfully by look at the first 10 rows
all_coupon_redeemed_by_month_year_df.head(10)
```

	Month	Year	Coupons Redeemed (Count)
0	1	2017	12919
1	1	2017	6753
2	1	2017	3088
3	1	2017	2
4	1	2017	791
5	1	2017	4977
6	1	2017	3
7	1	2017	3120
8	1	2017	7857
9	1	2017	3

```
## Group them by Month and Year to get a summarized count, and then we will be ready to plot!

all_redeemed_coupon_by_month_year_df = \
    all_coupon_redeemed_by_month_year_df.groupby(["Month", "Year"]) \
    .agg({"Coupons Redeemed (Count)": "sum"}).reset_index()

all_redeemed_coupon_by_month_year_df.rename(columns = {"Coupons Redeemed (Count)": "Coupons Redeemed (Sum)"},
                                           inplace = True)
```

```
## A few more steps - we'll create a month label with the actual month name

all_redeemed_coupon_by_month_year_df["Month (name)"] = \
    all_redeemed_coupon_by_month_year_df["Month"].apply(lambda x: calendar.month_name[x])

## Combine month and year, then sort the table by year/month
all_redeemed_coupon_by_month_year_df["Month & Year"] = all_redeemed_coupon_by_month_year_df["Month (name)"] + " " + \
    all_redeemed_coupon_by_month_year_df["Year"].astype(str)

## Sort the values by calendar year - and we should be done!
all_redeemed_coupon_by_month_year_df.sort_values(by = ["Year", "Month"], inplace = True)
```


MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

```
## Show the first five rows to confirm the final table is ready
all_redeemed_coupon_by_month_year_df.head()
```

	Month	Year	Coupons Redeemed (Sum)	Month (name)	Month & Year
0	1	2017	118336	January	January 2017
3	2	2017	123263	February	February 2017
6	3	2017	109829	March	March 2017
9	4	2017	98263	April	April 2017
12	5	2017	117856	May	May 2017

```
## Use the sum of redeemed coupons for our y-axis
all_redeemed_coupons_y = all_redeemed_coupon_by_month_year_df["Coupons Redeemed (Sum)"]

## Use the month and year for our x-axis
all_redeemed_coupons_x = all_redeemed_coupon_by_month_year_df["Month & Year"]
```

```
sns.set_style("darkgrid")

plt.figure(1, figsize = (10, 6))

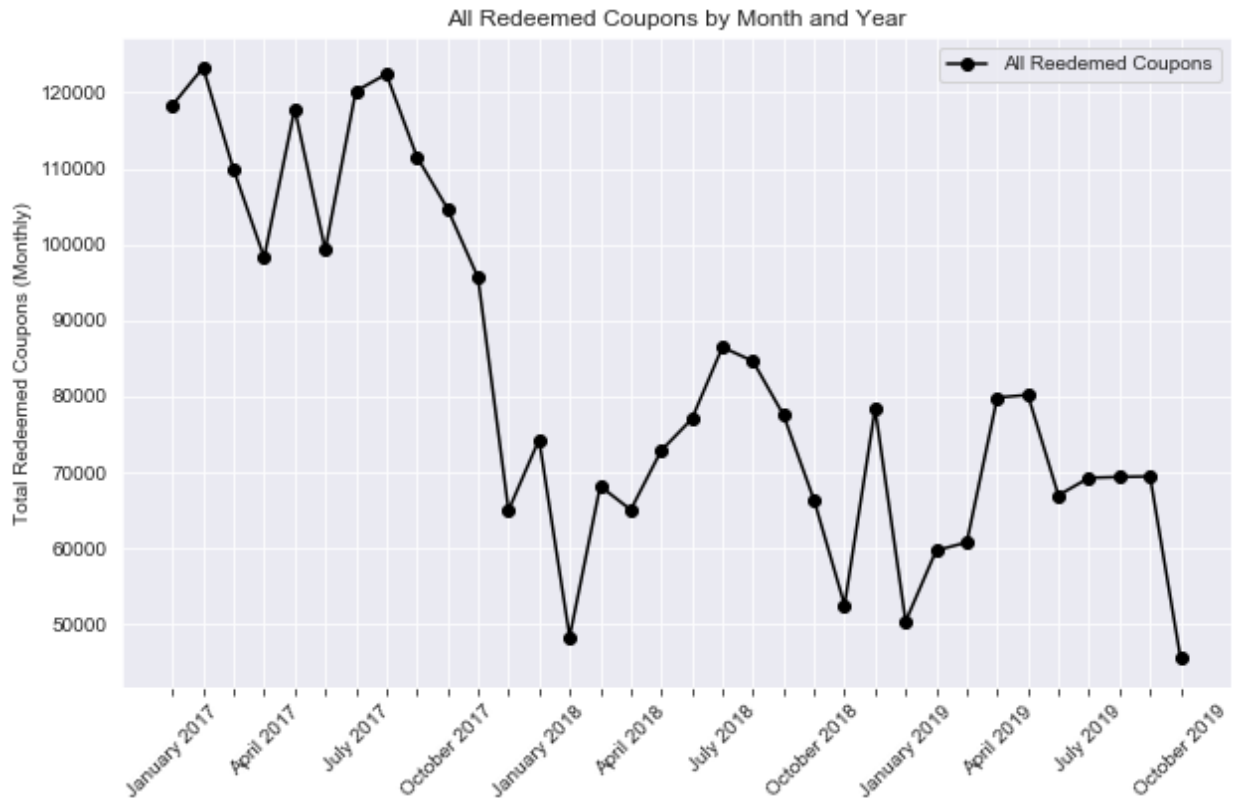
## Plot the first three type / subtype combinations
plt.plot(all_redeemed_coupons_x, all_redeemed_coupons_y, 'k-o', label = "All Redeemed Coupons")

plt.ylabel("Total Redeemed Coupons (Monthly)")
plt.title("All Redeemed Coupons by Month and Year")
plt.legend()

ax = plt.gca()

every_nth = 3
for n, label in enumerate(ax.xaxis.get_ticklabels()):
    label.set_rotation(45)
    if n % every_nth != 0:
        label.set_visible(False)
ax.tick_params(bottom = True)

## Show the graph!
plt.show()
```



Almost perfectly mirrors some of our earlier observations/plots. It's pretty fascinating to see - what could have led to such a precipitous drop? This is a question back to our business partners at MCLV.

5.4 Clustering & Customer Segmentation

- Start with Python Kernel
- Import the DimPlayer and Coupon Redeem tables
- Merge the files together using a left join, drop rows with null features, and extract only the demographic columns of interest.
- Update the data types of several features

```
play_redeem = pd.merge(dim_play, dim_cpnrredeem, on = 'PlayerID')
```

MSBA 6410 - Exploratory Analytics

Mille Lacs Corporate Ventures Live Case

```
forcluster = pd.DataFrame(play_redeem[['City', 'State', 'ZipCode', 'DistFromML', 'DistFromHK',  
    'BirthYear', 'PlayerStatus', 'Prison Flag', 'DMA Flag',  
    'Person # Marital Status', 'Estimated Current Home Value',  
    'DSE Discretionary Spend Estimate', 'Est Household Income V6']])  
  
forcluster = forcluster.dropna()  
  
forcluster['City'] = forcluster['City'].astype('category')  
forcluster['State'] = forcluster['State'].astype('category')  
forcluster['ZipCode'] = forcluster['ZipCode'].astype('category')  
forcluster['BirthYear'] = forcluster['BirthYear'].astype('float64')  
forcluster['Est Household Income V6'] = forcluster['Est Household Income V6'].astype('float64')  
  
forcluster.to_csv(r"D:\Group Folder\forcluster.csv")
```

- Switch to R Kernel

We leverage the *cluster* package to perform a K-medoids clustering analysis on the demographic information of players who are redeeming the promotions. The logic is to determine what characteristics about players we can extract. Understanding the clusters will allow MLCV to focus on the customer experience that each group requires in order to tailor the experience with a downstream strategy of attracting similar populations that aren't going to the casino.

```
library(cluster)  
library(tidyverse)  
library(lubridate)  
library(readr)  
library(Rtsne)
```

Read in the data file that we created with the Python kernel

```
data = read_csv('D:\\Group Folder\\forcluster.csv')
```

We have to update a number of data types in order for the k-medoid algorithm to function properly:

```
colnames(data)[11] <- 'MaritalStatus'  
  
data$X1 <- NULL  
names(data) = make.names(names(data))  
  
data = data %>% drop_na()  
  
data$City = as.factor(data$City)  
data$State = as.factor(data$State)  
data$ZipCode = as.factor(data$ZipCode)  
data$PlayerStatus = as.factor(data$PlayerStatus)  
data$Prison.Flag = as.factor(data$Prison.Flag)  
data$DMA.Flag = as.factor(data$DMA.Flag)  
data$MaritalStatus = as.factor(data$MaritalStatus)
```

Given the computational complexity of the data, we sample the data set to get a more manageable data set.

```
set.seed(7171989)

data_s = sample_n(data, 25000)
```

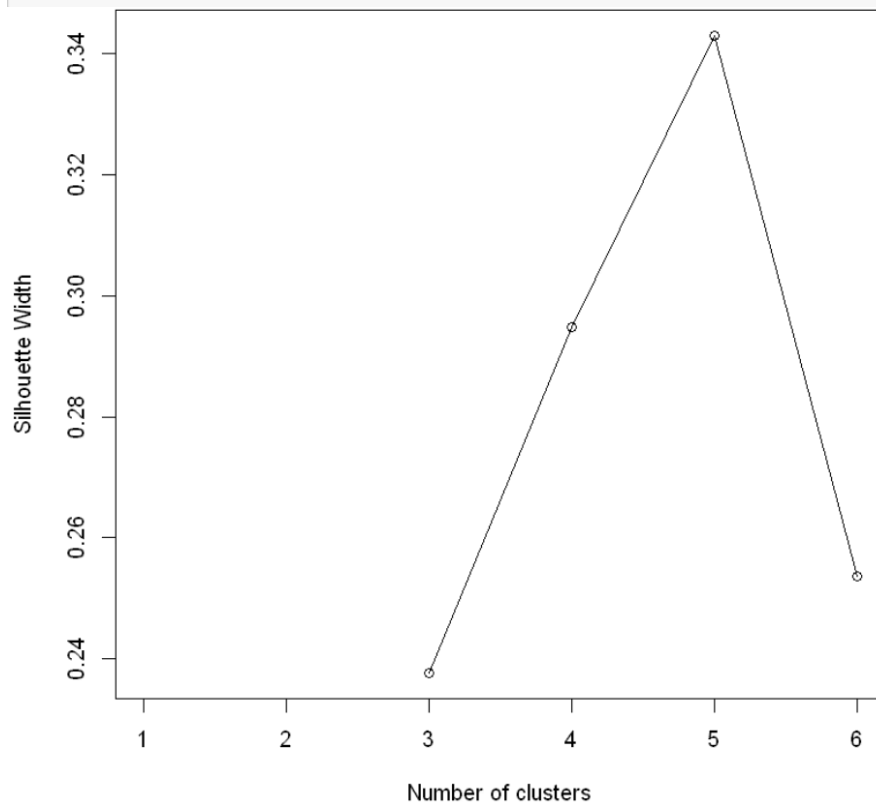
We leverage the gower distance calculation to determine the degrees or differences between each of the data observations.

```
gower_dist = daisy(data_s, metric = "gower")
```

We are still not sure what our clustering value should be so we leverage an iterative silhouette coefficient calculation to determine the ideal number of clusters the algorithm should be producing.

```
sil_width <- c(NA)
for(i in 3:6){
  pam_fit <- pam(gower_dist2, diss = TRUE, k = i)
  sil_width[i] <- pam_fit$silinfo$avg.width
}

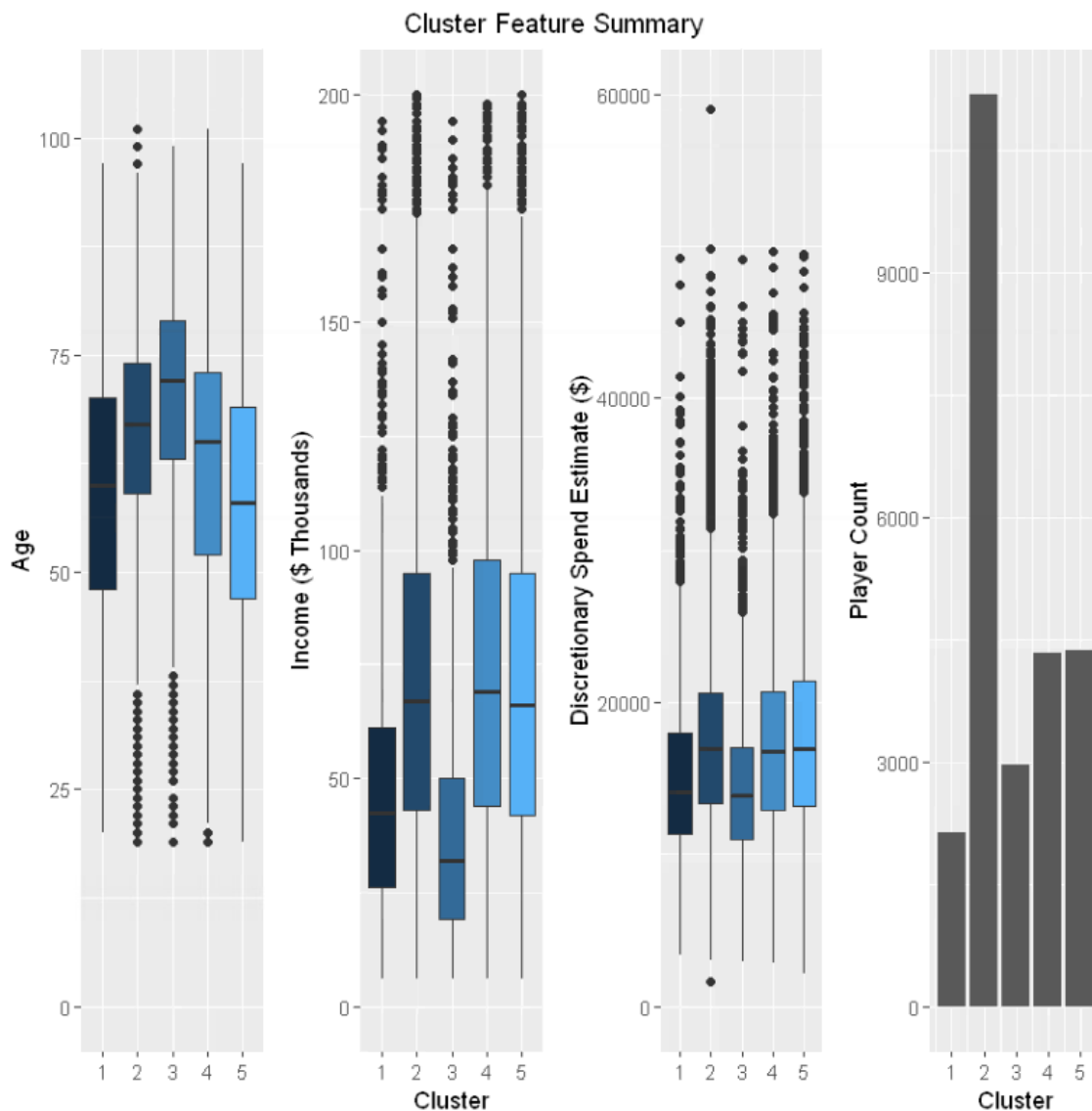
plot(1:6, sil_width,
     xlab = "Number of clusters",
     ylab = "Silhouette Width")
lines(1:6, sil_width)
```



One can see from the chart that the maximum silhouette width is shown at $k = 5$, so we leverage the *cluster* library to generate 5 clusters within our sampled data.

```
k = 5
pam_fit <- pam(gower_dist, diss = TRUE, k)
pam_results <- data_s %>% mutate(cluster = pam_fit$clustering) %>% group_by(cluster) %>% do(the_summary = summary(.))
pam_results$the_summary
```

We highlight a number of fields that are important when looking at the demographics of players and their casino habits, namely age, income, and discretionary spend (an estimate provided by MLCV). We also highlight the number of players within the clusters to show some relativity when looking at the three metrics.



From left to right, one can see clear differences in the age ranges, income ranges, and discretionary spend metrics between the clusters.

Cluster 1 appears to be a generally younger group with low amounts of income. These groups may respond better to value based promotions, that is, promotions where they can see the value of the money that they are spending.

Contrasting, clusters 2, 4, and 5 show high income levels with varying age groups and discretionary spending amounts. These players may be more interested in high stakes type promotions, such as free play on slots or table games with high betting levels. They may not be as receptive to value for money strategies compared to group one but understanding what these groups are looking for in terms of experience and value can help drive significant increases to revenue generation, profit, and visitorship.

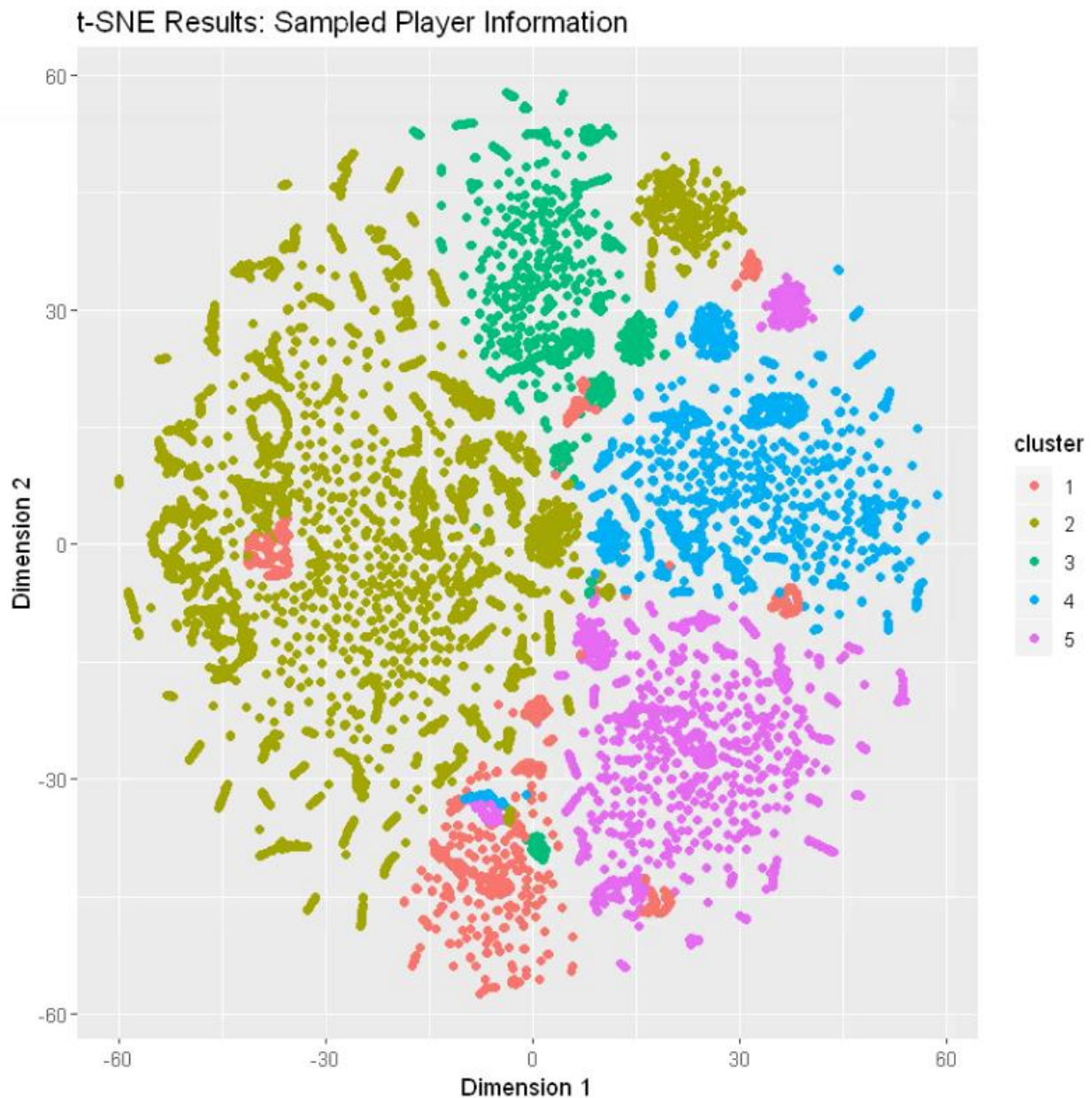
We can see clearly from the data that there exist distinct clusters of players that are redeeming the promotions. But a standard question to ask would be, how well did they cluster? One method of answering this question is through a tSNE plot (t-distributed stochastic neighbor embedding). tSNE plots allow one to visually see the separation within the clustering to help the validation process. tSNE is a dimensionality reduction technique that is useful in visualizing large datasets.

We leverage the Rtsne package to perform the analysis:

```
tsne_obj <- Rtsne(gower_dist2, is_distance = TRUE)
```

```
tsne_data <- tsne_obj$Y %>%  
  data.frame() %>%  
  setNames(c("X", "Y")) %>%  
  mutate(cluster = factor(pam_fit$clustering))
```

```
ggplot(aes(x = X, y = Y), data = tsne_data) +  
  geom_point(aes(color = cluster)) +  
  ggtitle("t-SNE Results: Sampled Player Information") +  
  labs(x="Dimension 1", y = "Dimension 2")
```



We can see from the tSNE plot that the clustering does appear to be strong, with clearly defined separation boundaries. We do see some pockets of color overlapping with other clusters, this is likely driven by the sample size and further analysis on a larger dataset will likely drive these decision boundaries to be cleaner and clearer.

5.5 Association Rules

We ran an association rule learning algorithm on a merged data table, combining all redeemed coupons and tying them to each player and that player's attributes. The *apriori* algorithm uses a process to mine & discover frequent itemsets/patterns by identifying frequent individual items in a data table and then extending these item sets larger and larger.

- Start with R kernel
- Load the appropriate libraries
- Load the data files
- Begin the initial exploration
- Transform the data to the appropriate format for association rules algorithm
- Assess and draw conclusions

```
## Bring in the libraries we need for the analysis
## Suppress the messages because they clog the notebook

suppressMessages(library(tidyr))
suppressMessages(library(ggplot2))
suppressMessages(library(dplyr))
suppressMessages(library(arules))
```

```
## Set the directory to load the data file - the fact table

data.dir = "D:/Group Folder/Datasets/"

## Set the file name

data.file = "FactCouponRedeem.csv"
```

```
## Create a look up to the CSV file

coupon_fact_table = paste(data.dir, data.file, sep = "")
```

Carve out the columns that we require for the association rules analysis:

- CouponID
- Month
- Year
- SiteID
- TargetPop
- ProgramType
- TypeDescription
- SubtypeDescription
- RedeemValue

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

- RedeemCount

```
## Read in the fact table
```

```
suppressWarnings(suppressMessages(coupon_fact_df <- readr::read_csv(file = coupon_fact_table)))
```

```
## Confirm we loaded in every row to the table
```

```
nrow(coupon_fact_df)
```

2777358

```
## Attach the df to R so we don't have to load it into memory everytime
```

```
attach(coupon_fact_df)
```

```
## Create a List in R with columns we want to subset on
```

```
columns_required <- c("CouponID", "PlayerID", "ValidMonth", "ValidYear", "SiteID", "TargetPop", "ProgramType",  
  "TypeDescription", "SubTypeDescription", "RedeemValue",  
  "RedeemCount")
```

```
## Subset the dataframe to only the columns we need for the association rules analysis
```

```
## We label it a transaction table since each couponID getting redeemed counts as a transaction
```

```
transaction_table <- coupon_fact_df[, columns_required]
```

```
## Show the structure of the "transaction" table
```

```
head(transaction_table)
```

CouponID	PlayerID	ValidMonth	ValidYear	SiteID	TargetPop	ProgramType	TypeDescription	SubTypeDescription	RedeemValue	RedeemCount
89729	530342410	12	16	1	VIP	Not Applicable	Hotel	ML	29	1
89729	710328524	12	16	1	VIP	Not Applicable	Hotel	ML	29	1
89093	1000600137	10	16	2	Not Applicable	Not Applicable	Free Slot Gaming	Grand Play	5	1
89729	710004002	12	16	1	VIP	Not Applicable	Hotel	ML	29	1
4	1000681589	4	7	2	New Member	Not Applicable	Other	Other	0	1
71400	1000681589	4	14	2	Not Applicable	Not Applicable	Free Slot Gaming	Grand Play	10	1

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

```
## Filter the table a little more - to mirror our earlier work of removing any rows with a  
## Redeem Count = 0
```

```
transaction_table_filtered <-  
  transaction_table %>%  
    filter(RedeemCount != 0) ## Remove any rows that have a zero redeem count
```

```
## Matching our row count from the first set of analyses in Python -success!
```

```
nrow(transaction_table_filtered)
```

2777131

```
## Create a unique identifier for every redeemed coupon by combining the coupon id and playerid  
## This should allow us to differentiate between different "transactions"
```

```
transaction_table_filtered$TransactionID <-  
  paste(transaction_table_filtered$CouponID, transaction_table_filtered$PlayerID)
```

```
## Subset the transaction table into Casino 1 and Casino 2 - likely differences between the two
```

```
casino_1_df <-  
  transaction_table_filtered %>%  
    filter(SiteID == 1)
```

```
casino_2_df <-  
  transaction_table_filtered %>%  
    filter(SiteID == 2)
```

```
## Move the transactionID to the first column in the dataframe  
## Remove SiteID since we don't really require it for this analysis
```

```
casino_1_df <- casino_1_df %>%  
  select(TransactionID, everything(), -SiteID)
```

```
## Look at a certain subsection of the data, that we will use to generate the Association rules
```

```
head(casino_1_df[, 4:10])
```

ValidMonth	ValidYear	TargetPop	ProgramType	TypeDescription	SubTypeDescription	RedeemValue
12	16	VIP	Not Applicable	Hotel	ML	29
12	16	VIP	Not Applicable	Hotel	ML	29
12	16	VIP	Not Applicable	Hotel	ML	29
12	16	VIP	Not Applicable	Hotel	ML	29
12	16	VIP	Not Applicable	Hotel	ML	29
12	16	VIP	Not Applicable	Hotel	ML	29

- The arules library in R requires the data to undergo a significant transformation before it can be fed into the algorithm for generating the rules.

```
#### We'll try this with one table first

trans_id <- as.character(casino_1_df[["TransactionID"]])
casino_1_df <- casino_1_df[, 4:10]

for (i in 1:ncol(casino_1_df)) casino_1_df[[i]] <- as.factor(casino_1_df[[i]])

trans <- as(casino_1_df, "transactions")

transactionInfo(trans)[["transactionID"]] <- trans_id
```

```
## Confirm the first set of transactions generated successfully

head(trans)
```

```
transactions in sparse format with
6 transactions (rows) and
414 items (columns)
```

- Use the apriori function to generate a set of rules based on the "transactions" we created above. We just want to see if there's anything that could provide useful information from mining the coupon redemptions for association rules.

```
## Create a new set of rules based on the transactions above
```

```
casino_1_rules <- apriori(trans, parameter = list(supp = 0.2,  
                                                  conf = 0.4, target = "rules", minlen=2))
```

Apriori

Parameter specification:

```
confidence minval smax arem aval originalSupport maxtime support minlen  
          0.4    0.1    1 none FALSE              TRUE        5    0.2    2  
maxlen target  ext  
      10  rules FALSE
```

Algorithmic control:

```
filter tree heap memopt load sort verbose  
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE
```

Absolute minimum support count: 234377

```
set item appearances ...[0 item(s)] done [0.00s].  
set transactions ...[414 item(s), 117188 transaction(s)] done [0.64s].  
sorting and recoding items ... [8 item(s)] done [0.05s].  
creating transaction tree ... done [0.56s].  
checking subsets of size 1 2 3 4 done [0.00s].  
writing ... [36 rule(s)] done [0.00s].  
creating S4 object ... done [0.12s].
```

```
## Convert these rules into a dataframe, so that we can have them in a format that is easier to  
## read and interpret
```

```
casino_1_arules_df <- DATAFRAME(casino_1_rules)
```

```
## Re-order the dataframe by lift
```

```
casino_1_arules_df <- casino_1_arules_df %>%  
  arrange(desc(lift))
```

Association rules produces the following metrics:

- **Support:** how often do these items appear together?
 - count of itemsets (X, Y) / all transactions containing (X, Y)
- **Confidence (strength of association):** given my LHS, how often do we see our RHS?
 - $P(Y | X)$
- **Lift (co-occurrence vs pure chance):** how often do I see LHS with RHS, compared to pure chance or coincidence?
 - $\text{supp}(X \rightarrow Y) / s(X) * s(Y)$
- **Count:** a raw count of occurrences of the itemsets

MSBA 6410 - Exploratory Analytics

Mille Lacs Corporate Ventures Live Case

```
## Show the first 20 rows of the dataframe with our new set of rules
```

```
head(casino_1_arules_df, 20)
```

	LHS	RHS	support	confidence	lift	count
	{ProgramType=Not Applicable,TypeDescription=Cash}	{SubTypeDescription=Bonus}	0.2275516	0.5068048	2.080282	266665
	{TargetPop=Not Applicable,TypeDescription=Cash}	{SubTypeDescription=Bonus}	0.2102803	0.5067939	2.080237	246425
	{TargetPop=Not Applicable,ProgramType=Not Applicable,TypeDescription=Cash}	{SubTypeDescription=Bonus}	0.2102803	0.5067939	2.080237	246425
	{SubTypeDescription=Bonus}	{TypeDescription=Cash}	0.2275610	0.9340698	2.080135	266676
	{TypeDescription=Cash}	{SubTypeDescription=Bonus}	0.2275610	0.5067689	2.080135	266676

- The first set of rules we generated didn't provide much meaningful context into what patterns could be uncovered using this approach.
- As successful data scientists, we don't stop here though! We think about what other meaningful information could be added that would add value.
- Therefore, we bring in player information using the "Player Day" table; we can use this to see what "tier" each player belongs to and if this has any influence on what coupons are being redeemed.

```
## Set up player file
```

```
player.file = "PlayerDay.csv"
```

```
## Set up player day table
```

```
player_day_table = paste(data.dir, player.file, sep = "")
```

```
## Read in the player day table - suppress warnings because they are annoying!
```

```
suppressWarnings(suppressMessages(player_day_df <- readr::read_csv(file = player_day_table)))
```

```
## Confirm we loaded in every row to the table
```

```
nrow(player_day_df)
```

```
7292910
```

```
## Attach the df to R so we don't have to load it into memory everytime
```

```
## Especially since our Virtual Machines were getting hammered daily!!
```

```
suppressMessages(attach(player_day_df))
```

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

```
## Pull out the columns need for merging

player_columns <- c("PlayerID", "TierName")

player_tier_df <- player_day_df[player_columns]
```

```
## Confirm this matches our original loading of the data table, with just the two columns we need
nrow(player_tier_df)
```

7292910

```
## De-duplicate the rows in this table so we have only unique player IDs and their tier information

player_tier_df_no_dupes <-
  player_tier_df %>%
    distinct(PlayerID, TierName)
```

```
## Print the de-duplicated row count and show the first few rows - to confirm its ready to be merged

print(nrow(player_tier_df_no_dupes))

head(player_tier_df_no_dupes)
```

[1] 395660

PlayerID	TierName
1000382937	Preferred
530533484	Preferred
530086340	Silver
710028836	Preferred
1000611579	Silver
710015699	Silver

```
## Create a new table and include the tier information from the de-duplicated PlayerDay table
## Now we overwrote the old file, but leave the test output to confirm the test was successful
```

```
transaction_table_filtered <-
  left_join(transaction_table_filtered, player_tier_df_no_dupes, by = "PlayerID")
```

```
## Success! We will overwrite the old DF with the new one including Tier information
```

```
head(transaction_table_filtered)
```

CouponID	PlayerID	ValidMonth	ValidYear	SitID	TargetPop	ProgramType	TypeDescription	SubTypeDescription
89729	530342410	12	16	1	VIP	Not Applicable	Hotel	ML
89729	710328524	12	16	1	VIP	Not Applicable	Hotel	ML
89093	1000600137	10	16	2	Not Applicable	Not Applicable	Free Slot Gaming	Grand Play
89729	710004002	12	16	1	VIP	Not Applicable	Hotel	ML
4	1000681589	4	7	2	New Member	Not Applicable	Other	Other
71400	1000681589	4	14	2	Not Applicable	Not Applicable	Free Slot Gaming	Grand Play

- As mentioned above, the Type Description = "Not Applicable" added a lot of background "noise" to the original set of rules.
- We remove these rows here as an extra step, before we continue with the generation of a new set of rules.

```
## Remove the rows with Not Applicable to generate the correct set of rules
```

```
transaction_table_filtered <-  
  transaction_table_filtered %>%  
    filter(ProgramType != "Not Applicable")
```

```
## Confirm lower row count
```

```
nrow(transaction_table_filtered)
```

517888

```
## Create a unique identifier for every redeemed coupon by combining the coupon id and playerid  
## This should allow us to differentiate between different "transactions"
```

```
transaction_table_filtered$TransactionID <-  
  paste(transaction_table_filtered$CouponID, transaction_table_filtered$PlayerID)
```

- We ran an association rule learning algorithm on the merged data table, combining all redeemed coupons and tying them to each player and that player's attributes. The apriori algorithm uses a process to mine & discover frequent itemsets/patterns by identifying frequent individual items in a data table and then extending these item sets larger and larger.
- As a very final step, we will subset each dataframe by year as well, to generate a more concise and specific set of rules.

```
## Subset the transaction table into Casino 1 and Casino 2
```

```
casino_1_df <-  
  transaction_table_filtered %>%  
    filter(SiteID == 1)
```

```
casino_2_df <-  
  transaction_table_filtered %>%  
    filter(SiteID == 2)
```

MSBA 6410 - Exploratory Analytics
Mille Lacs Corporate Ventures Live Case

```
casino_1_df_17 <-  
  casino_1_df %>%  
    filter(ValidYear == 17)  
  
casino_1_df_18 <-  
  casino_1_df %>%  
    filter(ValidYear == 18)  
  
casino_1_df_19 <-  
  casino_1_df %>%  
    filter(ValidYear == 19)
```

```
nrow(casino_1_df_17)  
  
nrow(casino_1_df_18)  
  
nrow(casino_1_df_19)
```

59230

57214

50720

ValidYear	SiteID	TargetPop	ProgramType	TypeDescription	SubTypeDescription	RedeemValue	RedeemCount	TransactionID	TierName
17	1	Loyalty	Gazette	Hotel	All	29	1	89502 24930	Gold
17	1	Loyalty	Gazette	Hotel	All	29	1	89502 530005985	Gold
17	1	Loyalty	Gazette	Cash	Cash	25	1	89301 530086721	Diamond
17	1	Loyalty	Gazette	Hotel	All	29	1	89502 530517544	Silver
17	1	Loyalty	Gazette	Hotel	All	29	1	89503 530517544	Silver
17	1	Loyalty	Gazette	Cash	Cash	25	1	89301 530483440	Diamond

```
trans_id_2 <- as.character(casino_1_df_17[["TransactionID"]])  
casino_1_df_17 <- casino_1_df_17[, 4:11]  
  
for (i in 1:ncol(casino_1_df_17)) casino_1_df_17[[i]] <- as.factor(casino_1_df_17[[i]])  
  
trans_2 <- as(casino_1_df_17, "transactions")  
  
transactionInfo(trans_2)[["transactionID"]] <- trans_id_2
```

```
## Confirm the rules generated successfully
```

```
head(trans_2)
```

```
transactions in sparse format with  
6 transactions (rows) and  
23 items (columns)
```



```
casino_1_rules_17 <- apriori(trans_2, parameter = list(supp = 0.2,  
  conf = 0.4, target = "rules", minlen=3))
```

Apriori

Parameter specification:

```
confidence minval smax arem aval originalSupport maxtime support minlen  
0.4 0.1 1 none FALSE TRUE 5 0.2 3  
maxlen target ext  
10 rules FALSE
```

Algorithmic control:

```
filter tree heap memopt load sort verbose  
0.1 TRUE TRUE FALSE TRUE 2 TRUE
```

Absolute minimum support count: 11846

```
set item appearances ...[0 item(s)] done [0.00s].  
set transactions ...[23 item(s), 59230 transaction(s)] done [0.03s].  
sorting and recoding items ... [8 item(s)] done [0.00s].  
creating transaction tree ... done [0.01s].  
checking subsets of size 1 2 3 4 5 6 7 8 done [0.00s].  
writing ... [960 rule(s)] done [0.00s].  
creating S4 object ... done [0.00s].
```

```
## Convert our completed rules into a dataframe
```

```
casino_1_arules_17_df <- DATAFRAME(casino_1_rules_17)
```

```
## Re-order the dataframe by Lift
```

```
casino_1_arules_17_df <- casino_1_arules_17_df %>%  
  arrange(desc(lift))
```

```
## Show the first 20 rows of the dataframe
```

```
head(casino_1_arules_17_df, 5)
```

	LHS	RHS	support	confidence	lift	count
	{ProgramType=Gazette,RedeemValue=29}	{SubTypeDescription=All}	0.8647138	1	1.001268	51217
	{TypeDescription=Hotel,RedeemValue=29}	{SubTypeDescription=All}	0.8821712	1	1.001268	52251
	{RedeemValue=29,RedeemCount=1}	{SubTypeDescription=All}	0.8821543	1	1.001268	52250
	{SiteID=1,RedeemValue=29}	{SubTypeDescription=All}	0.8821712	1	1.001268	52251
	{TargetPop=Loyalty,RedeemValue=29}	{SubTypeDescription=All}	0.8821712	1	1.001268	52251

MSBA 6410 - Exploratory Analytics

Mille Lacs Corporate Ventures Live Case

```
deduped_rules_casino_2_19_df <-
  casino_2_arules_19_df[ !duplicated(casino_2_arules_19_df[, c("lift")], fromLast = T), ]

deduped_rules_casino_2_19_final_df <-
  deduped_rules_casino_2_19_df[ !duplicated(deduped_rules_casino_2_19_df[, c("count")], fromLast = T), ]

head(deduped_rules_casino_2_19_final_df)
```

	LHS	RHS	support	confidence	lift	count
	{ValidYear=19,SitelD=2,TypeDescription=Hotel,SubTypeDescription=All}	{RedeemValue=29}	0.5572722	0.7744302	1.160241	63772
	{ValidYear=19,SitelD=2,TargetPop=Loyalty}	{ProgramType=Gazette}	0.8663532	0.9501184	1.096687	99142
	{ValidYear=19,SitelD=2,TypeDescription=Hotel,SubTypeDescription=All}	{TargetPop=Loyalty}	0.7121710	0.9896900	1.085380	81498
	{ValidYear=19,SitelD=2,TypeDescription=Hotel,RedeemValue=29}	{TargetPop=Loyalty}	0.6587438	0.9869212	1.082344	75384
dYear=19,SitelD=2,TypeDescription=Hotel,SubTypeDescription=All,RedeemValue=29}		{TargetPop=Loyalty}	0.5498532	0.9866869	1.082087	62923
	{ValidYear=19,SitelD=2,TypeDescription=Hotel,RedeemValue=29}	{ProgramType=Gazette}	0.6145968	0.9207808	1.062824	70332

- The analysis tells us that for all of the redeemed coupons for 2019, these characteristics kept materializing or cropping up: they belonged to Hotel, and their redemption value was \$29 (first row).
- Another slight variation to that "rule" is that many of the players redeeming these coupons belonged to the Gold tier, and that they were targeted towards "Loyalty" players.

Examples for the other association rules that were generated as part of this project can be found in

Group1_MCLV_LiveCase_PlayerDemographics_PromotionalCampaigns_Association Rules_Visualizations.ipynb file.

5.6 Sentiment Analysis

Sentiment analysis refers to the use of [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing), [text analysis](https://en.wikipedia.org/wiki/Computational_linguistics), [computational linguistics](https://en.wikipedia.org/wiki/Computational_linguistics), and [biometrics](https://en.wikipedia.org/wiki/Biometrics) to systematically identify, extract, quantify, and study affective states and subjective information.
[\(\[https://en.wikipedia.org/wiki/Sentiment_analysis\]\(https://en.wikipedia.org/wiki/Sentiment_analysis\)\)](https://en.wikipedia.org/wiki/Sentiment_analysis)

Sentiment Analysis is an incredibly useful approach to understanding customer reviews and applying analytics, such as a support vector machine (SVM), to analyze key words or n-grams to extract meaningful insight from a corpus of customer reviews. Our approach extracts customer reviews from the Yelp! platform and utilizes a TF-IDF trigram methodology to create a predictive model to define the key trigrams within the features.

- Start with Python kernel

We leverage the BeautifulSoup and urllib packages to scrape the Yelp! Source for the Onamia and Hinckley locations

```
import re
import nltk
import pandas as pd
from bs4 import BeautifulSoup
import urllib.request
```

```
#https://www.octoparse.com/blog/web-scraping-using-python
urls = ['https://www.yelp.com/biz/grand-casino-hinckley',
        'https://www.yelp.com/biz/grand-casino-hinckley?start=20',
        'https://www.yelp.com/biz/grand-casino-hinckley?start=40']

clean_reviews_hinckley = []
clean_stars_hinckley = []

for url in urls:
    review = []
    stars = []
    ourUrl = urllib.request.urlopen(url)
    soup = BeautifulSoup(ourUrl, 'html.parser')

    for i in soup.find_all("li"):
        per_review = i.find_all("div", {"class": "review-content"})
        for z in per_review:
            text = z.find("p")
            review.append(text)

    for i in soup.find_all('img', alt = True):
        if "star" in str(i):
            stars.append(i['alt'].replace(' star rating', ''))

    for each in review:
        new_each = str(each).replace('<br/>', '').replace('\xa0', '')
        new_each = new_each[13:-4]
        clean_reviews_hinckley.append(new_each)

    clean_stars_hinckley.append(stars[1:-2])

stars_hinckley = [item for sublist in clean_stars_hinckley for item in sublist]

stars_hinckley.pop(51)
stars_hinckley.pop(38)
stars_hinckley.pop(37)
stars_hinckley.pop(36)
stars_hinckley.pop(13)
stars_hinckley.pop(6)
stars_hinckley.pop(5)
```

```
#https://www.octoparse.com/blog/web-scraping-using-python
urls = ['https://www.yelp.com/biz/grand-casino-onamia-2',
        'https://www.yelp.com/biz/grand-casino-onamia-2?start=20',
        'https://www.yelp.com/biz/grand-casino-onamia-2?start=40']

clean_reviews_onamia = []
clean_stars_onamia = []

for url in urls:
    review = []
    stars = []
    ourUrl = urllib.request.urlopen(url)
    soup = BeautifulSoup(ourUrl, 'html.parser')

    for i in soup.find_all("li"):
        per_review = i.find_all("div", {"class": "review-content"})
        for z in per_review:
            text = z.find("p")
            review.append(text)

    for i in soup.find_all('img', alt = True):
        if "star" in str(i):
            stars.append(i['alt'].replace(' star rating', ''))

    for each in review:
        new_each = str(each).replace('<br/>', '').replace('\xa0', '')
        new_each = new_each[13:-4]
        clean_reviews_onamia.append(new_each)

    clean_stars_onamia.append(stars[1:-2])

stars_onamia = [item for sublist in clean_stars_onamia for item in sublist]
stars_onamia.pop(7) #this is an updated one
```

In order to simplify the predictive model we will create, and because we want to simplify the classification problem to a binary classification problem, we assign a flag to the reviews to indicate whether the review was a "good review", greater than or equal to four stars, or a "bad review", less than four stars.

We choose these splits with general rating based data handling in mind, with companies such as Lyft requiring their rideshare drivers to maintain a 4.6 rating to stay active on the platform.

```
binary_stars_onamia = []
for x in stars_onamia_int:
    if x >= 4:
        binary_stars_onamia.append(1)
    else:
        binary_stars_onamia.append(0)

binary_stars_hinckley = []
for x in stars_hinckley_int:
    if x >= 4:
        binary_stars_hinckley.append(1)
    else:
        binary_stars_hinckley.append(0)
```

We then combine the Hinckley and Onamia sets in order to create a more diverse data set, but in reality it may be useful to keep these separate and perform the analysis on each of the locations individually.

```
data_hinckley = list(zip(clean_reviews_hinckley, binary_stars_hinckley))
data_onamia = list(zip(clean_reviews_onamia, binary_stars_onamia))

data = data_hinckley + data_onamia
```

In order to extract meaningful weights from the SVM model that we will build, we chose to boot strap, with replacement, the reviews and their scores in order for the SVM hyperplane to generate more meaningful output. We boot strap the number of reviews up to 1,000 reviews.

```
from sklearn.utils import resample
import numpy as np

boot = resample(data, replace=True, n_samples=1000, random_state=1989)
boot = np.asarray(boot)

features = boot[:,0]
labels = boot[:,1]
```

Text analytics requires a great deal of preprocessing of the features in order to remove special characters, single characters, spaces, as well as case in order to ensure the text is normalized sufficiently to avoid undersampling or any outsized features.

```
processed_features = []

for sentence in range(0, len(features)):
    # Remove all the special characters
    processed_feature = re.sub(r'\W', ' ', str(features[sentence]))

    # remove all single characters
    processed_feature = re.sub(r'\s+[a-zA-Z]\s+', ' ', processed_feature)

    # Remove single characters from the start
    processed_feature = re.sub(r'^[a-zA-Z]\s+', ' ', processed_feature)

    # Substituting multiple spaces with single space
    processed_feature = re.sub(r'\s+', ' ', processed_feature, flags=re.I)

    # Removing prefixed 'b'
    processed_feature = re.sub(r'^b\s+', '', processed_feature)

    # Converting to Lowercase
    processed_feature = processed_feature.lower()

    processed_features.append(processed_feature)
```

The python NLTK (<https://www.nltk.org/>) Natural Language Toolkit is a powerful platform for text analytics research and applications. We leverage the NLTK package for tokenization and lemmatization of the features, review text.

```
import nltk
nltk.download('punkt')
from nltk.text import Text
from nltk.tokenize import sent_tokenize, word_tokenize

#ha and wa show up too often. Let's correct these. Ahead of Lemmatizing
import re
for x in range(0, len(processed_features)):
    processed_features[x] = re.sub(r'\bha\b', 'has', processed_features[x])
    processed_features[x] = re.sub(r'\bwa\b', 'was', processed_features[x])

nltk.download('wordnet')

def get_lemmatized_text(corpus):
    from nltk.stem import WordNetLemmatizer
    lemmatizer = WordNetLemmatizer()
    return ' '.join([lemmatizer.lemmatize(word) for word in review.split()])
    for review in corpus]

processed_features = get_lemmatized_text(processed_features)
```

We must also remove stop words from the review text, a common stage of feature processing in NLP analytics. We select a few sets of hyperparameters and also create tri-grams (three word combinations) from the processed review text.

```
from nltk.corpus import stopwords
nltk.download('stopwords')
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.util import ngrams

vectorizer = TfidfVectorizer(max_features=1000, min_df=7, max_df=0.8,
                             ngram_range=(3,3), analyzer = 'word',
                             stop_words=stopwords.words('english'))
processed_features_vec = vectorizer.fit_transform(processed_features).toarray()
```

To extract the importance, or usefulness, of the tri-grams, we utilize a support vector machine (SVM) classification model and extract the feature importance from the model.

The feature importance defines the weights that the SVM model generates in order to train the prediction model.

We first split our bootstrapped data into training and testing data sets using a typical approach (sklearn).

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    processed_features_vec, labels, test_size=0.2, random_state=0)
```

The classification model leverages the sklearn package and stochastic gradient descent in order to accelerate the training process, and increase its overall accuracy.

```
from sklearn import linear_model
clf = linear_model.SGDClassifier(max_iter=500, tol=1e-3)
clf.fit(X_train, y_train)

predictions = clf.predict(X_test)
```

```
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import accuracy_score, roc_auc_score

print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
print(roc_auc_score(y_test.astype(int), predictions.astype(int)))
print(accuracy_score(y_test, predictions))
```

```
[[147  0]
 [ 30 23]]
```

	precision	recall	f1-score	support
0	0.83	1.00	0.91	147
1	1.00	0.43	0.61	53
avg / total	0.88	0.85	0.83	200

```
0.7169811320754718
0.85
```

One can see from the output above, that the model achieves fairly good performance, with an AUC score of 0.71 and an overall accuracy of 85%. These metrics are not used, but do provide an indication as to how reliable the model is when we extract the feature importance from the models.

We extract the feature importance by combining the feature names as well as the coefficients that the model generated. In order to make them usable, we also order them for extraction.

```
output = pd.DataFrame(list(zip(vectorizer.get_feature_names(),
                                list(clf.coef_.flatten()))))

output = output.sort_values(by=[1])
```

For the purposes of this document, we elect to focus on the worst of the tri-grams in order to focus on areas of improvement that could be made at both the Onamia and Hinckley Locations.

If one were to execute the code, one would also be able to focus on tri-grams most predictive of positive reviews, such as the buffet and the cleanliness of the casino floors at the two locations.

If we extract features with a model coefficient (weight) or -2 or less, we can generate a list of the tri-grams of interest:


```
output[output[1]<=-2]
```

		0	1
842	think grand casino	-8.430190	
927	wa good hotel	-6.354808	
791	smoky nursing home	-5.281178	
93	black jack dealer	-4.546143	
525	live music event	-4.546143	
832	sure lung cancer	-4.331710	
146	casino grand casino	-4.146251	
25	able get room	-4.070226	
775	slot video poker	-4.057301	
624	penny nickel slot	-4.017469	
404	go back casino	-3.448243	
508	lake mille lac	-3.337959	
566	non smoking room	-3.259448	
432	grand casino hinckley	-3.030762	
591	one person get	-2.618057	
565	non smoking area	-2.153411	
709	room wa clean	-2.005981	
647	pillow smelled like	-2.001607	

We see a couple of common themes here, including things related to smoking being a frequent and large negative indicator, but also tri-grams that contains words such as 'was' and 'think' potentially indicating that the experience consumers are having on site is not what they are looking for as they reflecting on past experiences or have opinions on things to change.

NLP is quite an art, often more so than a science, but consumer reviews should be regarded very highly (both positive and negative reviews) as they are a pure ground truth source of data that experienced practitioners of NLP and text analytics can use to leverage incredible insight.

5.7 Assumptions

- Decreased smoking would lead to an increase in young gamblers
- Increasing promotions of high value (Cash, Hotel, Free Slot Gaming) would more than compensate for stopping low value promotions (Events, Gifts, Free Table Gaming)
- The trends with sports betting has continued to grow, we expect we would see similar results in Minnesota if regulations change
- Traditional gambling has decreased overtime due to the major macroeconomic influences mentioned above
- Customers from the metro area would be willing to travel if the promotions were tailored correctly

6. Results, Findings, Insights

We think the most important place for MLCV to focus on is the customer experience, modern customers value experience incredibly highly. We think that creating an experience that works for your current base and also resonates with your desired areas for growth. The experience value for dollar is particularly important for younger customers.

Existing customer base is on average: old, fairly well off, and not traveling particularly far to get to the casino. It might be difficult to find more people like them, so increasing frequency through promotions and appealing to younger potential players will be necessary for growth. To appeal to new customers, addressing negative reviews could be a powerful step. This could become a differentiating factor to gain players currently going elsewhere. We've seen research that it takes 40 positive reviews to counter one negative and that dissatisfied customers tell 9-15 people about their experience. While it must be done carefully to not alienate current customers, it will be important to growth and retention.

Narrowing the promotional offerings to those most popular and strategically offering them to the members as they advance through the levels may reduce costs and increase engagement. From analyzing the data provided, we recommend increasing Free Slot Gaming and Hotel promotions to Preferred and Silver tiers as the redemption rate for that promotion type is relatively high compared to others. This will get the "lower spend" players in the door and increase frequency. To entice the Platinum and Diamond player we recommend focusing on Cash promotions. Finally, Event, Gift, and Free Table Gaming promotions should be dramatically reduced or potentially even stopped. The redemption rate on these promotions is very low.

One of the growing trends in gambling is sports betting, we think this could be an opportunity to grow the base of loyal members. The model that Iowa has implemented allows for digital distribution which we think would be helpful for MLVC since players only need to visit the location once to play, however, this could also be a driver for sporting event watch weekends such as the World Cup or March Madness. We would recommend looking at what effect this is having in Iowa and seeing if this would make sense to implement here both an opportunity for growth and to prevent these players registering at casinos in Iowa.

Applying the theory from Airbnb where they aim to give people an experience so good that they tell everyone they know about how great it was, diving deeply into quantitative and qualitative data on how to improve the customer experience will be critical.

Conclusions:

- Cultivating the customer experience should be a key improvement area
- Opportunity for targeting customers from the Twin Cities
- Negative returns on Event, Gift, and Free Table Gaming Promotions
- Massive drop in redemptions entering 2018
- Smoking is damaging customers' views of the casinos
- Opportunity for entering new and growing markets in the gaming industry (sports betting)

Describe how the work can be carried forward, including steps for further analysis, how to scale up or extend the analysis.

7. Appendices

Supplementary and reference material may be best delivered in an appendix.

Examples include:

A. Group1_MCLV_LiveCase_PlayerDemographics_PromotionalCampaigns_AssociationRules_Visualizations.ipynb

This notebook contains all of the code used to generate the plots for Player Demographics, Promotional Campaigns, & Association Rules.

B. MCLV Geospatial2.ipynb

This notebook contains all of the code used to generate the analysis and visualizations related to the Geospatial & Location Demographics.

C. MCLV_ATM_Clustering.ipynb

This notebook contains all of the code used to generate the cluster analysis and the graphing of the tSNE plot.

D. MCLV_ATM_SentimentAnalysis.ipynb

This notebook contains all of the code used to scrape the Yelp! platform and perform the sentiment analysis.

8. References

- [1] Licata, A. (2019, August 2). 42 states have or are moving towards legalizing sports betting – here are the states where sports betting is legal. *Business Insider*. Retrieved from <https://www.businessinsider.com/states-where-sports-betting-legal-usa-2019-7>
- [2] Thomas, A. (2018, February 26). The Secret Ratio That Proves Why Customer Reviews Are So Important. *Inc.* Retrieved from <https://www.inc.com/andrew-thomas/the-hidden-ratio-that-could-make-or-break-your-company.html>
- [3] Williamson, P.J. & Zheng, M. (2009, March 1). Value-for-Money Strategies for Recessionary Times. *Harvard Business Review*. Retrieved from <https://hbr.org/2009/03/value-for-money-strategies-for-recessionary-times>
- [4] Hoon Lim, S. (2017, September 12). Spotlight on Economics: Do Casinos Have a Positive Effect on Economic Growth? North Dakota State University. Retrieved from <https://www.ag.ndsu.edu/news/columns/spotlight-on-economics/spotlight-on-economics-do-casinos-have-a-positive-effect-on-economic-growth/>
- [5] Kadimisetty, Avinash. (2018, May 11). Association Rule Mining in R. *Towards Data Science*. Retrieved from <https://towardsdatascience.com/association-rule-mining-in-r-ddf2d044ae50>