In [1]:
```
## This script will walk through a basic analysis of the Acquisition and
## Challenge (Nov 9th and 10th 2019)
##
## Here you will find the tools necessary to open, read, and process the
## idea of the types of differing risk factors over the years. The key fo
## should be built around the "Zero_Bal_Cd" attribute. Look further into
## on the different values this field can take on.
##
## In order to run this script you need to download data from this link:
## on the site but it's free)
##    https://loanperformancedata.fanniemae.com/lppub/index.html#Portfoli
##
## Additional details about these datasets (attribute names, allowable va
## is available from here:
##    https://www.fanniemae.com/portal/funding-the-market/data/loan-perfo
##
## Download the data the 3rd Quarter for the years 2004, 2008, 2012, and
##
## Unzip the data files into the "RawData" directory and then execute thi
##
## Make sure you have all the necessary python libraries listed below ins
##
```

In [2]:
```
import os
import glob
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
/usr/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning: numpy.dt
ype size changed, may indicate binary incompatibility. Expected 96, got
88
  return f(*args, **kwds)
/usr/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning: numpy.dt
ype size changed, may indicate binary incompatibility. Expected 96, got
88
  return f(*args, **kwds)
```

In [3]:
```
## print our current working directory to be sure we're operating in the
##
##os.getcwd()
```

In [4]:
```
## create a list of the acquisition data file names
##
all_Acq_files = glob.glob(os.path.join("RawData/Acquisition*.txt"))
```

In [5]:
```
## print out the list to make sure we've got them all
##
all_Acq_files
```

Out[5]:
```
['RawData/Acquisition_2004Q3.txt',
 'RawData/Acquisition_2016Q3.txt',
 'RawData/Acquisition_2008Q3.txt',
 'RawData/Acquisition_2012Q3.txt']
```

In [6]:
```
## read the contents of each acquisition file into a data frame
##
df_from_each_file = (pd.read_csv(f,sep ="|", index_col=None, header=None)
df   = pd.concat(df_from_each_file, ignore_index=True)
```

In [7]:
```
## The files don't have names for each column so add the columns here
##
df.rename(columns={
                    0: 'Loan_ID',
                    1: 'Channel',
                    2: 'Seller',
                    3: 'Interest_Rate',
                    4: 'UPB',
                    5: 'Loan_Term',
                    6: 'Origination_Date',
                    7: 'First_Payment_Date',
                    8: 'LTV',
                    9: 'CLTV',
                    10: 'Num_Borrowers',
                    11: 'DTI',
                    12: 'Borrower_FICO',
                    13: 'First_Time_Buyer',
                    14: 'Loan_Purpose',
                    15: 'Dwelling_Type',
                    16: 'Unit_Count',
                    17: 'Occupancy',
                    18: 'State',
                    19: 'Zip',
                    20: 'Insurance%',
                    21: 'Product',
                    22: 'Co_Borrower_FICO',
                    23: 'Mortgage_Insurance_Type',
                    24: 'Relocation_Indicator'}, inplace=True)
```

In [8]:
```
## Now grab a listing of all the performance files in the RawData directo
##
all_perf_files = glob.glob(os.path.join( "RawData/Performance_*.txt"))
```

In [9]:
```
## display a listong of the performance files to make sure the year/quart
## with the acquisition files
##
all_perf_files
```

Out[9]:
```
['RawData/Performance_2012Q3.txt',
 'RawData/Performance_2004Q3.txt',
 'RawData/Performance_2016Q3.txt',
 'RawData/Performance_2008Q3.txt']
```

In [10]:
```
## read in the data from each of the performance files and concatenate th
## data together into a single dataframe names "perf_df" and while we're
## the columns we actually want for this analysis.
##
df_from_each_file = (pd.read_csv(f,sep ="|", index_col=None, header=None
                                 ,usecols=[0,1,3,4,5,11,12]
                                 , names = ['Loan_ID', 'Period', 'Current
                                            'Mod_Ind','Zero_Bal_Cd']
                                 ,dtype = { 'Loan_ID' : np.int64, 'Curren
                                            'Current_UPB': np.float64}
                                ) for f in all_perf_files)
perf_df   = pd.concat(df_from_each_file, ignore_index=True)
```

In [11]:
```
## Modify the date field ("Period") to be a number for easier manipulatio
## later on in the script
##
perf_df['Period']=perf_df['Period'].apply(str).str[6:].apply(int)*100+per

## Select the latest period in the data frame as we're concerned with the
##
idx = perf_df.groupby(['Loan_ID'])['Period'].transform(max) == perf_df['P

## Create a new data frame with just the latest period record
##
perf_df_new = perf_df[idx].copy()
```

In [12]:
```
## In looking at the FAQ dor the datasets we know that if the zero balanc
## meaning it's paid up correctly. It's not late, or paid off early, or i
## about those records as regards our analysis.
##
perf_df_new.Zero_Bal_Cd.fillna(0,inplace=True)

## Also, some of the loans are missing the UPB (unpaid balance). We can't
## so we'll just drop those loans from the dataframe
##
perf_df_new.dropna(inplace=True)

## create a mapping of the available zero_balance_code numbers and their
##
zero_bal_cd_map = {0:'Current',1:'Prepaid',2:'Third Party Sale',3:'Short
                   6:'Repurchase',9:'REO',15:'Note Sale',16:'RPL Loan Sal
perf_df_new['Zero_Bal_Cd'] = perf_df_new['Zero_Bal_Cd'].map(zero_bal_cd_m

## display a listing of the updated performance data
##
perf_df_new.head()
```

Out[12]:

| | Loan_ID | Period | Current_IR | Current_UPB | Age | Mod_Ind | Zero_Bal_Cd |
|---|---|---|---|---|---|---|---|
| **81** | 100002679724 | 201906 | 3.625 | 110549.80 | 82 | N | Current |
| **150** | 100003137281 | 201805 | 3.375 | 147781.08 | 68 | N | Prepaid |
| **234** | 100004790326 | 201906 | 4.125 | 219852.60 | 84 | N | Current |
| **268** | 100006404894 | 201504 | 3.000 | 123368.58 | 33 | N | Prepaid |
| **305** | 100008536293 | 201508 | 3.250 | 159596.31 | 36 | N | Prepaid |

In [15]:
```
## Now that we've cleaned up the acquisition and performance data, merge
## data frame that we'll call "loan_df"
##
loan_df = pd.merge(df,perf_df_new,how='inner',on='Loan_ID')

## display the first several rows from the combined dataset
##
loan_df.head()
```

Out[15]:

| | Loan_ID | Channel | Seller | Interest_Rate | UPB | Loan_Term | Origination_Date | I |
|---|---|---|---|---|---|---|---|---|
| **0** | 100001458647 | R | CITIMORTGAGE, INC. | 5.625 | 297000 | 360 | 05/2004 | |
| **1** | 100004788186 | C | BANK OF AMERICA, N.A. | 5.750 | 50000 | 180 | 08/2004 | |
| **2** | 100008528816 | R | OTHER | 5.000 | 80000 | 180 | 08/2004 | |
| **3** | 100014656651 | C | BANK OF AMERICA, N.A. | 6.300 | 55000 | 240 | 07/2004 | |
| **4** | 100021529837 | C | BANK OF AMERICA, N.A. | 5.875 | 140000 | 360 | 07/2004 | |

5 rows × 31 columns

In [16]:
```
loan_df.isnull().sum()
```

Out[16]:
```
Loan_ID                       0
Channel                       0
Seller                        0
Interest_Rate                 0
UPB                           0
Loan_Term                     0
Origination_Date              0
First_Payment_Date            0
LTV                           0
CLTV                          4
Num_Borrowers                65
DTI                       40037
Borrower_FICO              3450
First_Time_Buyer              0
Loan_Purpose                  0
Dwelling_Type                 0
Unit_Count                    0
Occupancy                     0
State                         0
Zip                           0
Insurance%              1678310
Product                       0
Co_Borrower_FICO        1017720
Mortgage_Insurance_Type 1678310
Relocation_Indicator          0
Period                        0
Current_IR                    0
Current_UPB                   0
Age                           0
Mod_Ind                       0
Zero_Bal_Cd                   0
dtype: int64
```

In [17]:
```
## Assign Defaults for the missing values in the loans dataframe
##
loan_df.Mortgage_Insurance_Type.fillna(0,inplace=True)
loan_df['Insurance%'].fillna(0,inplace=True)
loan_df.Num_Borrowers.fillna(1,inplace=True)
loan_df.CLTV.fillna(loan_df.LTV,inplace=True)
loan_df.drop('Co_Borrower_FICO',axis=1,inplace=True)
```

In [18]:
```
loan_df.isnull().sum()
```

Out[18]:
```
Loan_ID                      0
Channel                      0
Seller                       0
Interest_Rate                0
UPB                          0
Loan_Term                    0
Origination_Date             0
First_Payment_Date           0
LTV                          0
CLTV                         0
Num_Borrowers                0
DTI                      40037
Borrower_FICO             3450
First_Time_Buyer             0
Loan_Purpose                 0
Dwelling_Type                0
Unit_Count                   0
Occupancy                    0
State                        0
Zip                          0
Insurance%                   0
Product                      0
Mortgage_Insurance_Type      0
Relocation_Indicator         0
Period                       0
Current_IR                   0
Current_UPB                  0
Age                          0
Mod_Ind                      0
Zero_Bal_Cd                  0
dtype: int64
```

In [19]:
```
## Drop any records that still have null values - we don't want to inclua
##
loan_df.dropna(inplace=True)
```

In [20]:
```
## We'll do some analysis against the FICO (credit score) of the borrower
## Create several bins based on the FICO score range and add the calculat
## to each record in the dataframe
##
FICO_bins = [0,620,660,700,740,780,850]
FICO_labels = ['0-620', '620-660','660-700','700-740','740-780','780+']
loan_df['FICO_bins'] = pd.cut(loan_df['Borrower_FICO'],bins=FICO_bins,lab

Term_bins =[0,180,360]
Term_labels =['<=15 Years','<= 30 Years']
loan_df['Term_bins'] = pd.cut(loan_df['Loan_Term'],bins=Term_bins,labels=

zero_bal_cd_map = {'Current':'Current','Prepaid':'Prepaid','Third Party S
                   'Repurchase':'Underperforming','REO':'Underperforming'
loan_df['Current_Status'] = loan_df['Zero_Bal_Cd'].map(zero_bal_cd_map).a

loan_df['Origin_Month'],loan_df['Origin_Year'] = loan_df['First_Payment_D

df = loan_df[loan_df['Origin_Year'].isin(['2003','2008','2012','2016'])]
```

In [21]:
```
## Build a table showing the current status and total for each status typ
##
df.groupby(['Origin_Year','Current_Status']).agg({'Loan_ID':'count'})
```

Out[21]:

| | | Loan_ID |
|---|---|---|
| **Origin_Year** | **Current_Status** | |
| **2003** | **Current** | 265 |
| | **Prepaid** | 3759 |
| | **Underperforming** | 52 |
| **2008** | **Current** | 24646 |
| | **Prepaid** | 296382 |
| | **Underperforming** | 19573 |
| **2012** | **Current** | 383238 |
| | **Prepaid** | 330469 |
| | **Underperforming** | 1007 |
| **2016** | **Current** | 514154 |
| | **Prepaid** | 108063 |
| | **Underperforming** | 551 |

In [22]:
```
## Create a new dataframe that holds the first 100,000 records
##
df2 = df.groupby('Origin_Year').head(100000)
```

In [23]:
```
## display another table showing the total number of each status by year ##
df2.groupby(['Origin_Year','Current_Status']).agg({'Loan_ID':'count'})
```

Out[23]:

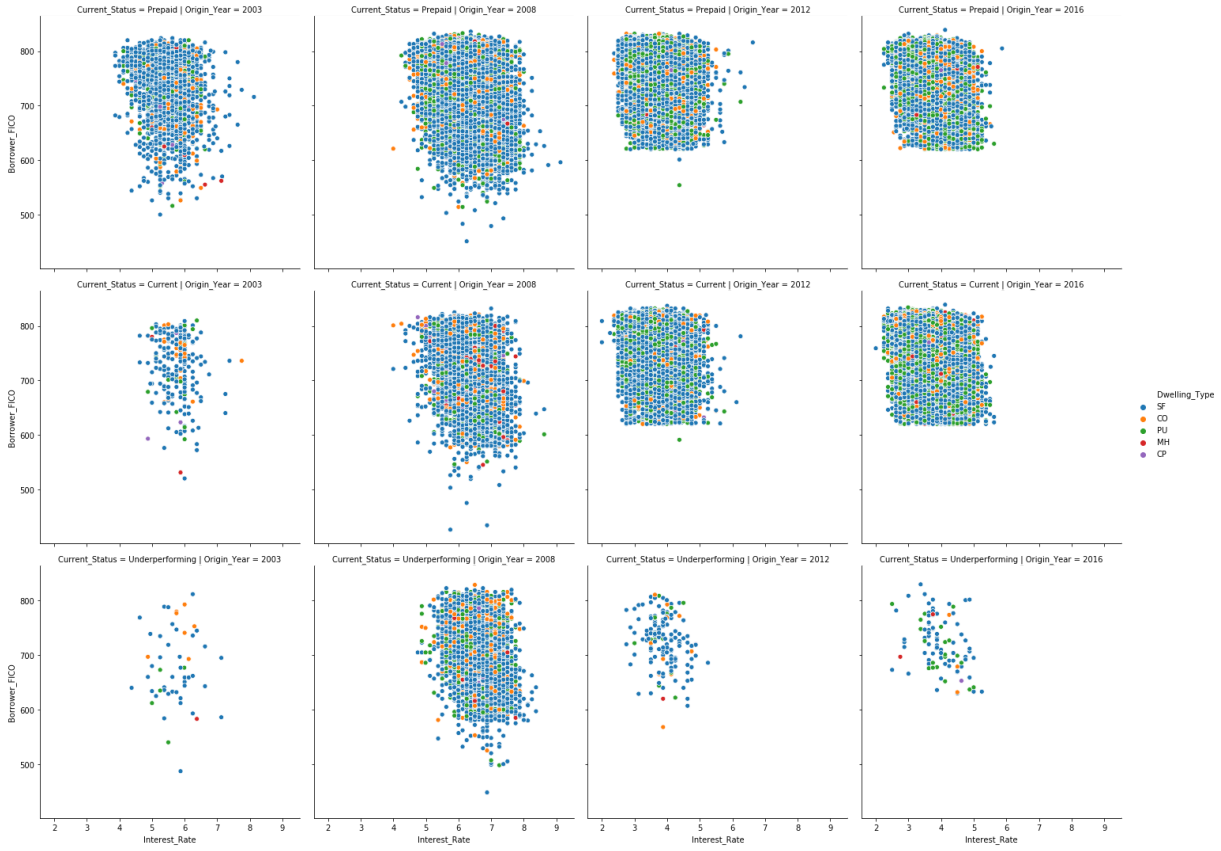
| | | Loan_ID |
|---|---|---|
| **Origin_Year** | **Current_Status** | |
| **2003** | **Current** | 265 |
| | **Prepaid** | 3759 |
| | **Underperforming** | 52 |
| **2008** | **Current** | 7212 |
| | **Prepaid** | 87017 |
| | **Underperforming** | 5771 |
| **2012** | **Current** | 53775 |
| | **Prepaid** | 46070 |
| | **Underperforming** | 155 |
| **2016** | **Current** | 82548 |
| | **Prepaid** | 17352 |
| | **Underperforming** | 100 |

In [24]:
```
## Dump some info about the attributes that make up our dataframe
##
## Out of this list of attributes, which ones (and with what values) corr
## mortages for each of the 4 years? Is there some attribute that remains
## These are the "risk factors" we're interested in having you find in th
## interesting way.
##
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 304076 entries, 49 to 1466311
Data columns (total 35 columns):
Loan_ID                    304076 non-null int64
Channel                    304076 non-null object
Seller                     304076 non-null object
Interest_Rate              304076 non-null float64
UPB                        304076 non-null int64
Loan_Term                  304076 non-null int64
Origination_Date           304076 non-null object
First_Payment_Date         304076 non-null object
LTV                        304076 non-null int64
CLTV                       304076 non-null float64
Num_Borrowers              304076 non-null float64
DTI                        304076 non-null float64
Borrower_FICO              304076 non-null float64
First_Time_Buyer           304076 non-null object
Loan_Purpose               304076 non-null object
Dwelling_Type              304076 non-null object
Unit_Count                 304076 non-null int64
Occupancy                  304076 non-null object
State                      304076 non-null object
Zip                        304076 non-null int64
Insurance%                 304076 non-null float64
Product                    304076 non-null object
Mortgage_Insurance_Type    304076 non-null float64
Relocation_Indicator       304076 non-null object
Period                     304076 non-null int64
Current_IR                 304076 non-null float64
Current_UPB                304076 non-null float64
Age                        304076 non-null int64
Mod_Ind                    304076 non-null object
Zero_Bal_Cd                304076 non-null object
FICO_bins                  304076 non-null category
Term_bins                  304076 non-null category
Current_Status             304076 non-null object
Origin_Month               304076 non-null object
Origin_Year                304076 non-null object
dtypes: category(2), float64(9), int64(8), object(16)
memory usage: 79.5+ MB
```

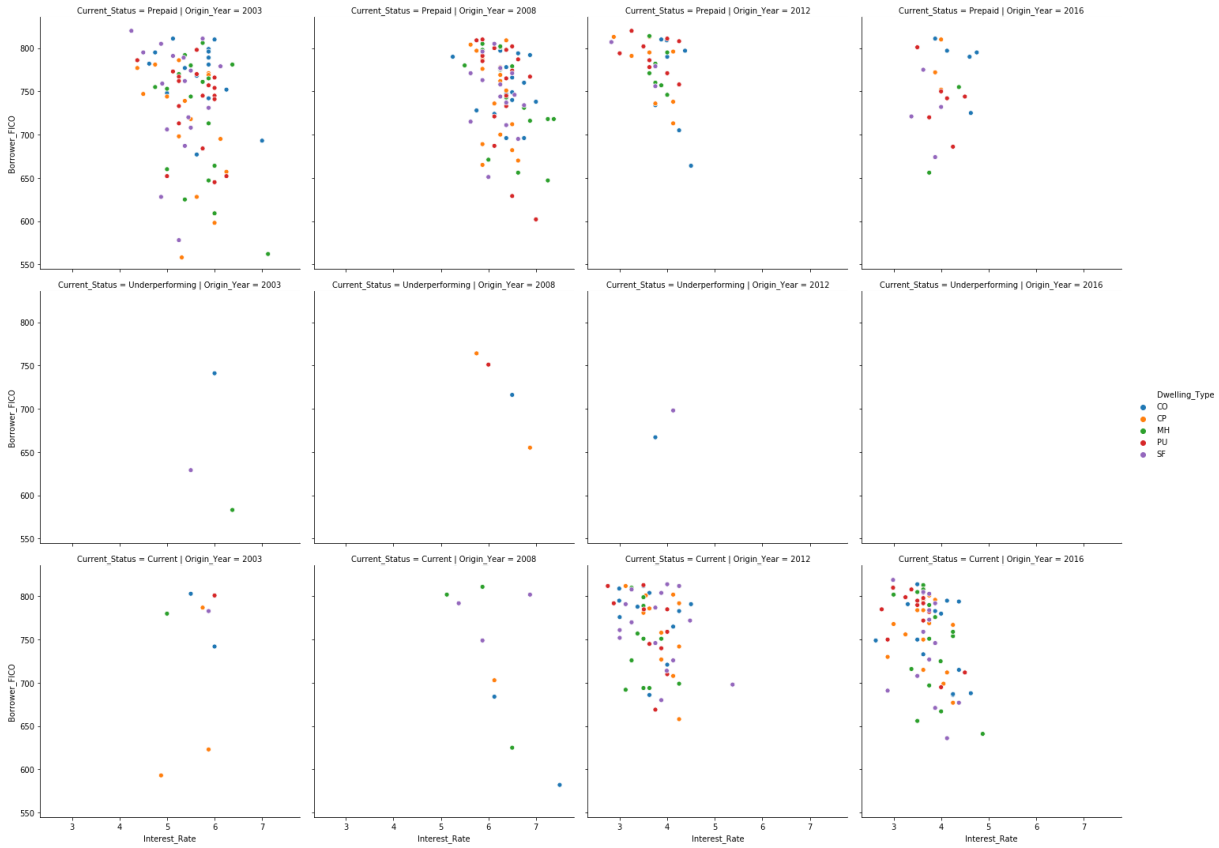
In [25]:
```
## Write our current data frame out to a file. This will allow us to pick
## analysis without going through all the previous work to clean and stru
## the data correctly.
##
df2.to_csv('Processed_loans.csv',index=False)
```

In [26]:
```
## Create and display a plot showing the distribution by year for the Bor
## current, underperforming, and prepaid (paid off early) loans
##
sns.relplot(y='Borrower_FICO', x='Interest_Rate', data=df2
            ,hue='Dwelling_Type',
            row='Current_Status', col='Origin_Year')
plt.show()
```




In [27]:
```
## Rebalance the record set by Dwelling Type into a new data frame (df3).
## Family data (individual family homes) is far larger than the other typ
## a more balanced view of the data. You could go back and do a deeper an
## factors on that basis, or by zip code, number of borrowers, etc...
##
g = df2.groupby(['Origin_Year','Dwelling_Type'])
df3 = g.apply(lambda x: x.sample(g.size().min())).reset_index(drop=True)
```

In [28]:
```
## Redisplay the graph based on the same attributes used above
##
sns.relplot(y='Borrower_FICO', x='Interest_Rate', data=df3#.query('Curren
            ,hue='Dwelling_Type',
            row='Current_Status', col='Origin_Year')
plt.show()
```




In [ ]:
```
```