In [10]:
```
## This script will walk through a basic analysis of the Acquisition a
nd Performance data for the Data Science
## Challenge (Nov 9th and 10th 2019)
##
## Here you will find the tools necessary to open, read, and process t
he data as well as give you some
## idea of the types of differing risk factors over the years. The key
focus of your your analysis
## should be built around the "Zero_Bal_Cd" attribute. Look further in
to the script for more details
## on the different values this field can take on.
##
## In order to run this script you need to download data from this lin
k: (you will have to create an account
## on the site but it's free)
##    https://loanperformancedata.fanniemae.com/lppub/index.html#Portf
olio
##
## Additional details about these datasets (attribute names, allowable
values, definitions, etc:
## is available from here:
##    https://www.fanniemae.com/portal/funding-the-market/data/loan-pe
rformance-data.html
##
## Download the data the 3rd Quarter for the years 2004, 2008, 2012, a
nd 2016.
## Unzip the data files into the "RawData" directory and then execute
 this script.
## (you will need to have the included libraries installed (see the li
st below))
##
```

In [87]:
```
import os
import glob
import pandas as pd
import numpy as np

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

In [88]:
```
## print our current working directory to be sure we're operating in t
he right place
##
##os.getcwd()
```

In [63]:
```
## create a list of the acquisition data file names
##
all_Acq_files = glob.glob(os.path.join("RawData/Acquisition*.txt"))
```

In [64]:
```
## print out the list to make sure we've got them all
##
all_Acq_files
```

Out[64]:
```
['RawData/Acquisition_2004Q3.txt',
 'RawData/Acquisition_2016Q3.txt',
 'RawData/Acquisition_2008Q3.txt',
 'RawData/Acquisition_2012Q3.txt']
```

In [65]:
```
## read the contents of each acquisition file into a data frame
##
df_from_each_file = (pd.read_csv(f,sep ="|", index_col=None, header=No
ne) for f in all_Acq_files)
df   = pd.concat(df_from_each_file, ignore_index=True)
```

In [66]:
```
## 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 [67]:
```
## Now grab a listing of all the performance files in the RawData dire
ctory
##
all_perf_files = glob.glob(os.path.join( "RawData/Performance_*.txt"))
```

In [68]:
```
## display a listong of the performance files to make sure the year/qu
arter aligns
## with the acquisition files
##
all_perf_files
```

Out[68]:
```
['RawData/Performance_2012Q3.txt',
 'RawData/Performance_2004Q3.txt',
 'RawData/Performance_2016Q3.txt',
 'RawData/Performance_2008Q3.txt']
```

In [69]:
```
## read in the data from each of the performance files and concatenate
the
## data together into a single dataframe names "perf_df"
##
df_from_each_file = (pd.read_csv(f,sep ="|", index_col=None, header=No
ne
                                 ,usecols=[0,1,3,4,5,11,12]
                                 , names = ['Loan_ID', 'Period', 'Curr
ent_IR','Current_UPB', 'Age',
                                            'Mod_Ind','Zero_Bal_Cd']
                                 ,dtype = { 'Loan_ID' : np.int64, 'Cur
rent_IR' : np.float64,
                                           'Current_UPB': np.float64}
                                ) for f in all_perf_files)
perf_df   = pd.concat(df_from_each_file, ignore_index=True)
```

In [70]:
```
## Modify the date field ("Period") to be a number for easier manipula
tion
## later on in the script
##
perf_df['Period']=perf_df['Period'].apply(str).str[6:].apply(int)*100+
perf_df['Period'].apply(str).str[:2].apply(int)

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

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

In [71]:
```
## In looking at the FAQ dor the datasets we know that if the zero bal
ance code is null then the loan is current
## meaning it's paid up correctly. It's not late, or paid off early, o
r in default
##
perf_df_new.Zero_Bal_Cd.fillna(0,inplace=True)

## Also, some of the loans don't have a UPB (unpaid balance). We can't
use that data in building our model
## 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 the
ir meanings
##
zero_bal_cd_map = {0:'Current',1:'Prepaid',2:'Third Party Sale',3:'Sho
rt Sale',
                   6:'Repurchase',9:'REO',15:'Note Sale',16:'RPL Loan
 Sale'}
perf_df_new['Zero_Bal_Cd'] = perf_df_new['Zero_Bal_Cd'].map(zero_bal_c
d_map).apply(str)

perf_df_new
```

Out[71]:

	Loan_ID	Period	Current_IR	Current_UPB	Age	Mod_Ind	Zero_Bal_(
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
387	100008741734	201906	3.750	135291.59	81	N	Current
469	100010567729	201906	2.875	116108.33	80	N	Current
493	100011023127	201408	4.375	107736.31	23	N	Prepaid
577	100012697764	201906	4.250	231085.49	84	N	Current
661	100013153592	201906	4.250	64765.21	83	N	Current
743	100013617784	201906	3.000	176560.81	81	N	Current
787	100014015206	201603	3.500	203397.26	43	N	Prepaid
837	100014589820	201609	3.990	355501.78	51	N	Prepaid
921	100015546042	201906	3.750	353354.50	83	N	Current
980	100016027477	201706	3.375	178309.60	58	N	Prepaid
1062	100016250604	201906	2.750	139056.65	80	N	Current
1146	100018967703	201906	3.625	126891.95	82	N	Current
1203	100018967858	201703	4.000	36262.01	56	N	Prepaid
1253	100019243370	201608	3.750	274493.44	48	N	Prepaid
1308	100019459756	201701	3.500	92149.66	54	N	Prepaid
1320	100020496125	201307	4.125	548217.15	12	N	Prepaid
1403	100022449231	201906	3.625	295859.83	82	N	Current
1452	100022930706	201607	3.125	78612.84	48	N	Prepaid
1536	100023123707	201906	2.875	123482.38	82	N	Current
1581	100023201264	201603	4.375	281224.08	44	N	Prepaid
1598	100024021987	201311	3.000	270495.60	16	N	Prepaid
1680	100025839051	201906	3.750	185647.02	82	N	Current
1728	100026964101	201606	3.875	181086.32	48	N	Prepaid
1812	100027593502	201906	3.250	148254.28	83	N	Current
1872	100028009064	201706	2.990	82625.19	58	N	Prepaid
...	...	...	...	...	...	...	...
109357264	999927484807	200912	6.250	119016.53	16	N	Prepaid

	Loan_ID	Period	Current_IR	Current_UPB	Age	Mod_Ind	Zero_Bal_(
109357269	999930047433	200812	6.000	584000.00	4	N	Prepaid
109357400	999934658336	201906	6.250	29552.28	130	N	Current
109357482	999940323683	201504	6.000	149210.63	83	N	Prepaid
109357498	999943580764	200912	6.875	501303.39	15	N	Prepaid
109357541	999950451401	201201	5.875	80607.09	43	N	Prepaid
109357550	999953073866	200903	5.875	413979.90	8	N	Prepaid
109357599	999954192423	201208	5.500	83096.94	48	N	Prepaid
109357729	999955147903	201906	6.250	148778.58	129	N	Current
109357782	999956315763	201212	6.000	278435.80	53	N	Prepaid
109357844	999956682737	201310	6.000	72647.35	60	N	Prepaid
109357855	999957697918	200906	6.000	148923.50	10	N	Prepaid
109357985	999959338653	201906	6.250	53802.13	128	N	Current
109358030	999961174075	201204	6.250	356115.80	45	N	Prepaid
109358078	999961282690	201208	7.500	46048.07	48	N	Repurchase
109358101	999964227722	201007	6.000	317061.54	21	N	Prepaid
109358149	999965805672	201206	7.125	62220.20	47	N	Prepaid
109358172	999967109281	201007	6.500	262613.24	22	N	Prepaid
109358199	999969386579	201010	6.750	116570.79	25	N	Prepaid
109358276	999972374984	201411	2.000	183070.82	76	Y	Short Sale
109358321	999972656036	201203	6.125	105635.18	45	N	Prepaid
109358359	999978594909	201109	5.000	80982.05	38	N	Prepaid
109358371	999979386817	200908	6.000	146327.15	11	N	Prepaid
109358376	999984148064	200812	6.375	43000.00	5	N	Prepaid
109358387	999991636992	200906	5.999	395352.03	10	N	Prepaid
109358396	999995504903	200904	6.250	236909.47	9	N	Prepaid
109358406	999996916020	200905	7.125	413926.48	10	N	Prepaid
109358421	999998818599	200909	5.625	232088.77	15	N	Prepaid
109358468	999998855421	201207	5.990	177008.35	48	N	Prepaid
109358476	999999246774	200903	5.875	325624.50	8	N	Prepaid

2081532 rows × 7 columns

In [72]:
```
## Now that we've cleaned up the acquisition and performance data merg
e them into a single, integrated
## 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[72]:

	Loan_ID	Channel	Seller	Interest_Rate	UPB	Loan_Term	Origina
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 [73]:
```
loan_df.isnull().sum()
```

Out[73]:
```
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 [74]:
```
## 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 [75]:
```
loan_df.isnull().sum()
```

Out[75]:
```
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 [76]:
```
## Drop any records that have null values - we don't want to include t
hem in the model / analysis
##
loan_df.dropna(inplace=True)
```

In [79]:
```
## We'll do some analysis against the FICO (credit score) of the borro
wer
## Create several bins based on the FICO score range and add the calcu
lated FICO bin score
## 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,
labels=FICO_labels)

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

zero_bal_cd_map = {'Current':'Current','Prepaid':'Prepaid','Third Part
y Sale':'Underperforming','Short Sale':'Underperforming',
                   'Repurchase':'Underperforming','REO':'Underperformi
ng','Note Sale':'Underperforming','RPL Loan Sale':'Underperforming'}
loan_df['Current_Status'] = loan_df['Zero_Bal_Cd'].map(zero_bal_cd_map
).apply(str)

loan_df['Origin_Month'],loan_df['Origin_Year'] = loan_df['First_Paymen
t_Date'].str.split('/', 1).str

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

In [80]:
```
## Build a table showing the current status and total for each status
 type by year
##
df.groupby(['Origin_Year','Current_Status']).agg({'Loan_ID':'count'})
```

Out[80]:

		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 [30]:
```
## Create a new dataframe that holds the first 100,000 records
##
df2 = df.groupby('Origin_Year').head(100000)
```

In [81]:
```
## display another table showing the total number of each status by ye
ar
##
df2.groupby(['Origin_Year','Current_Status']).agg({'Loan_ID':'count'})
```

Out[81]:

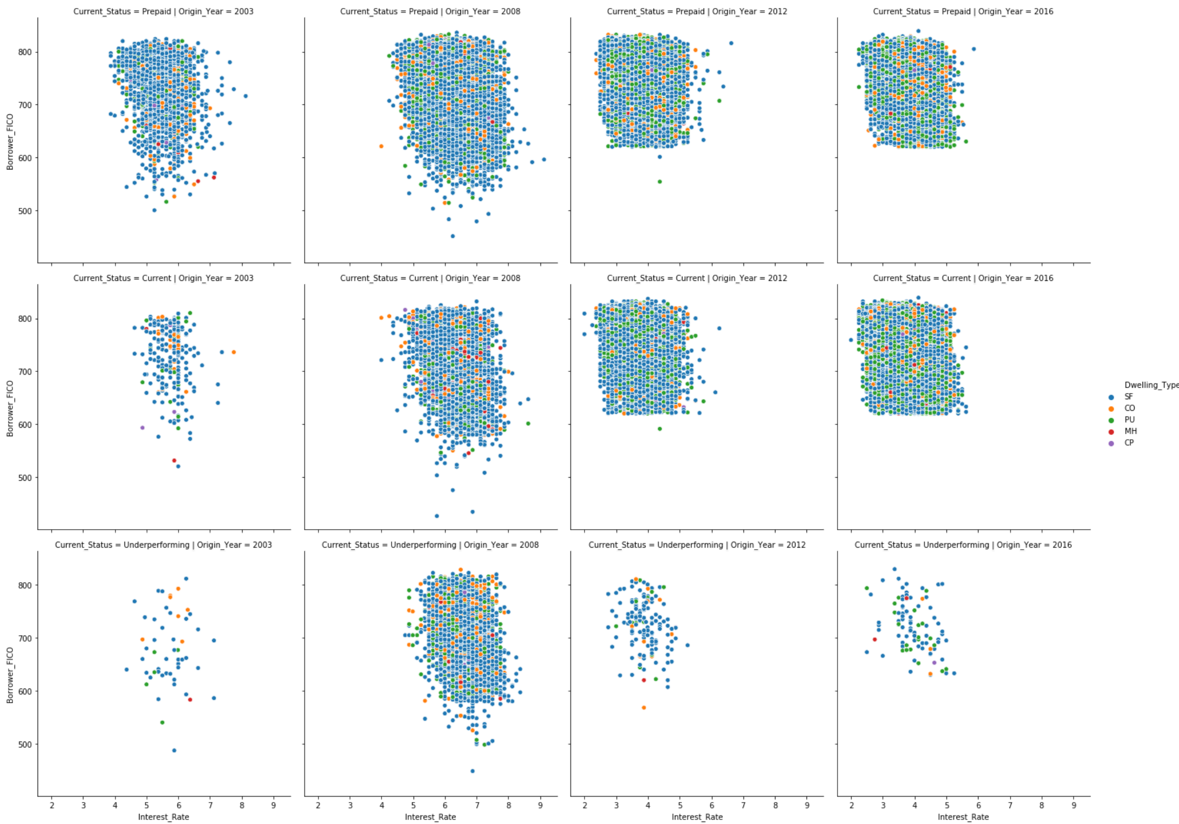
		Loan_ID
Origin_Year	**Current_Status**	
2003	**Current**	265
	Prepaid	3759
	Underperforming	52
2008	**Current**	7212
	Prepaid	87017
	Underperforming	5771
2012	**Current**	53775
	Prepaid	46070
	Underperforming	155
2016	**Current**	82548
	Prepaid	17352
	Underperforming	100

In [82]:
```
## dump some info about the attributes that make up our dataframe
##
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 304076 entries, 49 to 1466311
Data columns (total 34 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
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(7), object(16)
memory usage: 77.1+ MB
```

In [84]:
```
## Write our current data frame out to a file. This will allow us to p
ick up and continue our
## analysis without going through all the previous work to clean and s
tructure
## the data correctly.
##
df2.to_csv('Processed_loans.csv',index=False)
```

In [39]:
```
## Create and display a plot showing the distribution by year for the
 Borrower FICO score vs
## 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 [85]:
```
## Rebalance the record set by Dwelling Type into a new data frame (df
3)
##
g = df2.groupby(['Origin_Year','Dwelling_Type'])
df3 = g.apply(lambda x: x.sample(g.size().min())).reset_index(drop=Tru
e)
```

In [86]:
```
## Redisplay the graph based on the same attributes used above
##
sns.relplot(y='Borrower_FICO', x='Interest_Rate', data=df3#.query('Current_Status != "Prepaid"'), #kind='line',
            ,hue='Dwelling_Type',
            row='Current_Status', col='Origin_Year')
plt.show()
```