(index.html)

Home (index.html)

Munge (munge.html)

Aggregate (aggregate.html)

Visualize (visualize.html)

Time Series (timeseries.html)

Using Pandas for Analyzing Data - Data Munging

wavedatalab.github.io (http://wavedatalab.github.io/datawithpython/index.html)

In [2]:
```
%matplotlib
import numpy as np
import pandas as pd
```

```
Using matplotlib backend: Qt4Agg
```

Read the csv file of your choice using Pandas

In [3]:
```
ver=pd.read_csv("ver.csv")
```

View the first few rows of the file and set the number of columns displayed. Note: Using ver.head() will display the first five rows.

In [5]:
```
pd.set_option('display.max_columns', 80)
ver.head(3)
```

Out[5]:

	action_taken	action_taken_name	agency_code	agency_abbr	agency_name	applicant_ethnicity	applicant_ethnicity_n
0	1	Loan originated	1	OCC	Office of the Comptroller of the Currency	2	Not Hispanic or Latino
1	1	Loan originated	1	OCC	Office of the Comptroller of the Currency	2	Not Hispanic or Latino
2	1	Loan originated	5	NCUA	National Credit Union Administration	2	Not Hispanic or Latino

Determine the number of rows and columns in the dataset

In [6]:
```
ver.shape
```

Out[6]:
```
(34573, 46)
```

Find the number of rows in the dataset

In [7]:
```
len(ver)
```

Out[7]:
```
34573
```

Get the names of the columns

In [9]:
```
ver.columns
```

Out[9]:
```
Index([u'action_taken', u'action_taken_name', u'agency_code', u'agency_abbr
', u'agency_name', u'applicant_ethnicity', u'applicant_ethnicity_name', u'a
pplicant_income_000s', u'applicant_race_1', u'applicant_race_name_1', u'app
licant_sex', u'applicant_sex_name', u'census_tract_number', u'co_applicant_
ethnicity', u'co_applicant_ethnicity_name', u'co_applicant_race_1', u'co_ap
plicant_race_name_1', u'co_applicant_sex', u'co_applicant_sex_name', u'coun
ty_code', u'county_name', u'hoepa_status', u'hoepa_status_name', u'lien_sta
tus', u'lien_status_name', u'loan_purpose', u'loan_purpose_name', u'loan_ty
pe', u'loan_type_name', u'owner_occupancy', u'owner_occupancy_name', u'prea
pproval', u'preapproval_name', u'property_type', u'property_type_name', u'p
urchaser_type', u'purchaser_type_name', u'hud_median_family_income', u'loan
_amount_000s', u'number_of_1_to_4_family_units', u'number_of_owner_occupied
_units', u'minority_population', u'population', u'tract_to_msamd_income', u
'logloanamt', u'logincome'], dtype='object')
```

Get the first five rows of a column by name

In [18]:
```
ver['action_taken'][:5]
```

Out[18]:
```
0    1
1    1
2    1
3    1
4    1
Name: action_taken, dtype: int64
```

Create categorical ranges for numerical data. Note that that you can specifiy the number of ranges you wish.

In [19]:
```
incomeranges = pd.cut(ver['applicant_income_000s'], 14)
incomeranges[:5]
```

Out[19]:
```
0    (-8.998, 715.143]
1    (-8.998, 715.143]
2    (-8.998, 715.143]
3    (-8.998, 715.143]
4    (-8.998, 715.143]
Name: applicant_income_000s, dtype: category
Categories (14, object): [(-8.998, 715.143] < (715.143, 1429.286] < (1429.
286, 2143.429] < (2143.429, 2857.571] ... (7142.429, 7856.571] < (7856.571
, 8570.714] < (8570.714, 9284.857] < (9284.857, 9999]]
```

Look at the value counts in the ranges created above

In [20]:
```
pd.value_counts(incomeranges)
```

Out[20]:
```
(-8.998, 715.143]       34306
(715.143, 1429.286]       200
(1429.286, 2143.429]       33
(2143.429, 2857.571]       21
(2857.571, 3571.714]        6
(3571.714, 4285.857]        4
(5714.143, 6428.286]        2
(9284.857, 9999]            1
(8570.714, 9284.857]        0
(7856.571, 8570.714]        0
(7142.429, 7856.571]        0
(6428.286, 7142.429]        0
(5000, 5714.143]            0
(4285.857, 5000]            0
dtype: int64
```

Index into the first six columns of the first row

In [21]:
```
ver.ix[0,0:6]
```

Out[21]:
```
action_taken                                                1
action_taken_name                                Loan originated
agency_code                                                 1
agency_abbr                                               OCC
agency_name           Office of the Comptroller of the Currency
applicant_ethnicity                                         2
Name: 0, dtype: object
```

Order the data by specified column

In [22]:
```
ver['loan_amount_000s'].order()[:5]
```

Out[22]:
```
4193     1
32737    1
32965    1
34335    1
34342    1
Name: loan_amount_000s, dtype: int64
```

Sort by a column and that obtain a cross-section of that data

In [23]:
```
sorteddata = ver.sort(['loan_amount_000s'])
sorteddata.ix[:,0:6].head(3)
```

Out[23]:

	action_taken	action_taken_name	agency_code	agency_abbr	agency_name	applicant_ethn
4193	1	Loan originated	5	NCUA	National Credit Union Administration	2
32737	6	Loan purchased by the institution	9	CFPB	Consumer Financial Protection Bureau	3
32965	6	Loan purchased by the institution	9	CFPB	Consumer Financial Protection Bureau	3

Obtain the first three rows and first three columns of the sorted data

In [24]:
```
sorteddata.iloc[0:3,0:3]
```

Out[24]:

	action_taken	action_taken_name	agency_code
4193	1	Loan originated	5
32737	6	Loan purchased by the institution	9
32965	6	Loan purchased by the institution	9

Obtain value counts of specifiec column

In [25]:
```
ver['action_taken_name'].value_counts()
```

Out[25]:
```
Loan originated                                21852
Application denied by financial institution     4894
Application withdrawn by applicant              2759
Loan purchased by the institution               2671
Application approved but not accepted           1421
File closed for incompleteness                   976
dtype: int64
```

A way to obtain the datatype for every column

In [26]:
```
zip(ver.columns, [type(x) for x in ver.ix[0,:]])
```

Out[26]:
```
[('action_taken', numpy.int64),
 ('action_taken_name', str),
 ('agency_code', numpy.int64),
 ('agency_abbr', str),
 ('agency_name', str),
 ('applicant_ethnicity', numpy.int64),
 ('applicant_ethnicity_name', str),
 ('applicant_income_000s', numpy.int64),
 ('applicant_race_1', numpy.int64),
 ('applicant_race_name_1', str),
 ('applicant_sex', numpy.int64),
 ('applicant_sex_name', str),
 ('census_tract_number', numpy.float64),
 ('co_applicant_ethnicity', numpy.int64),
 ('co_applicant_ethnicity_name', str),
 ('co_applicant_race_1', numpy.int64),
 ('co_applicant_race_name_1', str),
 ('co_applicant_sex', numpy.int64),
 ('co_applicant_sex_name', str),
 ('county_code', numpy.int64),
 ('county_name', str),
 ('hoepa_status', numpy.int64),
 ('hoepa_status_name', str),
 ('lien_status', numpy.int64),
 ('lien_status_name', str),
 ('loan_purpose', numpy.int64),
 ('loan_purpose_name', str),
 ('loan_type', numpy.int64),
 ('loan_type_name', str),
 ('owner_occupancy', numpy.int64),
 ('owner_occupancy_name', str),
 ('preapproval', numpy.int64),
 ('preapproval_name', str),
 ('property_type', numpy.int64),
 ('property_type_name', str),
 ('purchaser_type', numpy.int64),
 ('purchaser_type_name', str),
 ('hud_median_family_income', numpy.int64),
 ('loan_amount_000s', numpy.int64),
 ('number_of_1_to_4_family_units', numpy.int64),
 ('number_of_owner_occupied_units', numpy.int64),
 ('minority_population', numpy.float64),
 ('population', numpy.int64),
 ('tract_to_msamd_income', numpy.float64),
 ('logloanamt', numpy.float64),
 ('logincome', numpy.float64)]
```

The Pandas way to obtain datatypes for every column

In [5]:
```
ver.dtypes
```

Out[5]:
```
action_taken                      int64
action_taken_name                object
agency_code                       int64
agency_abbr                      object
agency_name                      object
applicant_ethnicity               int64
applicant_ethnicity_name         object
applicant_income_000s             int64
applicant_race_1                  int64
applicant_race_name_1            object
applicant_sex                     int64
applicant_sex_name               object
census_tract_number             float64
co_applicant_ethnicity            int64
co_applicant_ethnicity_name      object
co_applicant_race_1               int64
co_applicant_race_name_1         object
co_applicant_sex                  int64
co_applicant_sex_name            object
county_code                       int64
county_name                      object
hoepa_status                      int64
hoepa_status_name                object
lien_status                       int64
lien_status_name                 object
loan_purpose                      int64
loan_purpose_name                object
loan_type                         int64
loan_type_name                   object
owner_occupancy                   int64
owner_occupancy_name             object
preapproval                       int64
preapproval_name                 object
property_type                     int64
property_type_name               object
purchaser_type                    int64
purchaser_type_name              object
hud_median_family_income          int64
loan_amount_000s                  int64
number_of_1_to_4_family_units     int64
number_of_owner_occupied_units    int64
minority_population             float64
population                        int64
tract_to_msamd_income           float64
logloanamt                      float64
logincome                       float64
dtype: object
```

Get the unique values for a column by name.

In [27]:
```
ver['county_name'].unique()
```

Out[27]:
```
array(['Grand Isle County', 'Chittenden County', 'Washington Co
       'Franklin County', 'Caledonia County', 'Addison County',
       'Orange County', 'Orleans County', 'Rutland County',
       'Windham County', 'Lamoille County', 'Windsor County',
       'Bennington County', 'Essex County'], dtype=object)
```

Get a count of the unique values of a column

In [28]:
```
len(ver['county_name'].unique())
```

Out[28]:
```
14
```

Index into a column and get the first four rows

In [31]:
```
ver.ix[0:3,'preapproval_name']
```

Out[31]:
```
0              Not applicable
1              Not applicable
2    Preapproval was requested
3              Not applicable
Name: preapproval_name, dtype: object
```

Obtain binary values

```
In [32]: ver.ix[0:3,'preapproval_name'] == "Preapproval was rec
```

```
Out[32]: 0    False
         1    False
         2     True
         3    False
         Name: preapproval_name, dtype: bool
```