11 - Pandas-Merge-Join-Concat

March 22, 2023

Table of Contents

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
import matplotlib as mpl
import seaborn as sns
from numpy.random import randn
```

[2]:
```
#from google.colab import drive
#drive.mount('/content/drive')
```

[3]:
```
acs =    pd.read_csv('https://raw.githubusercontent.com/jimcody2014/python-data/
 ↪main/acs2017.csv')
places = pd.read_csv('https://raw.githubusercontent.com/jimcody2014/python-data/
 ↪main/places.csv')
print(acs.shape)
print(places.shape)
```

```
(3221, 18)
(3142, 18)
```

1 Reducing rows and columns

1.1 Reducing the rows

[4]:
```
# Earlier, we used the code similar to the code below as a way of removing rows
# from the dataframe.

# In this code we are selecting the rows required

acs = acs.loc[acs['State']== 'Georgia']
acs.shape

# This code modifies the existing dataframe.
```

[4]:
```
(159, 18)
```

[5]:
```
# What if we wanted a new dataframe?

ga_a = acs.loc[acs.State == 'Georgia']
ga_a.shape
```

[5]:
```
(159, 18)
```

[6]:
```
type(ga_a)
```

[6]:
```
pandas.core.frame.DataFrame
```

1.1.1 An alternative approach

[7]:
```
places.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3142 entries, 0 to 3141
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   StateAbbr        3142 non-null   object
 1   StateDesc        3142 non-null   object
 2   CountyName       3142 non-null   object
 3   CountyFIPS       3142 non-null   int64
 4   TotalPopulation  3142 non-null   int64
 5   harthritis       3142 non-null   float64
 6   hasthma          3142 non-null   float64
 7   hbphigh          3142 non-null   float64
 8   hcancer          3142 non-null   float64
 9   hhighchol        3142 non-null   float64
 10  hkidney          3142 non-null   float64
 11  hcopd            3142 non-null   float64
 12  hchd             3142 non-null   float64
 13  hdiabetes        3142 non-null   float64
 14  hmhlth           3142 non-null   float64
 15  hphlth           3142 non-null   float64
 16  hteethlost       3142 non-null   float64
 17  hstroke          3142 non-null   float64
dtypes: float64(13), int64(2), object(3)
memory usage: 442.0+ KB
```

[8]:
```
ga_p = places[places['StateAbbr'] == 'GA']
ga_p.shape
```

[8]:
```
(159, 18)
```

[9]:
```
type(ga_p)
```

[9]:
```
pandas.core.frame.DataFrame
```

1.2 Reducing the columns

[10]:
```
drop_columns = {'Hispanic','White','Black','Native','Asian','Pacific'}
drop_columns

ga_a.drop(columns = drop_columns, inplace=True)
```

[11]:
```
ga_a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 159 entries, 387 to 545
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   CountyId          159 non-null    int64
 1   State             159 non-null    object
 2   County            159 non-null    object
 3   TotalPop          159 non-null    int64
 4   Men               159 non-null    int64
 5   Women             159 non-null    int64
 6   VotingAgeCitizen  159 non-null    int64
 7   Income            159 non-null    int64
 8   IncomePerCap      159 non-null    int64
 9   Poverty           159 non-null    float64
 10  ChildPoverty      156 non-null    float64
 11  Unemployment      159 non-null    float64
dtypes: float64(3), int64(7), object(2)
memory usage: 16.1+ KB
```

[12]:
```
ga_a.head()
```

[12]:

	CountyId	State	County	TotalPop	Men	Women	VotingAgeCitizen	Income	IncomePerCap	Poverty	ChildPoverty	Unemployment
387	13001	Georgia	Appling County	18471	9090	9381	13387	37089	19936	24.7	31.8	7.7
388	13003	Georgia	Atkinson County	8313	4112	4201	5245	33063	19904	27.4	45.6	8.4
389	13005	Georgia	Bacon County	11279	5599	5680	7903	38824	18856	23.2	34.4	3.1
390	13007	Georgia	Baker County	3251	1547	1704	2512	43867	22270	19.5	19.4	2.5
391	13009	Georgia	Baldwin County	45527	22893	22634	36104	37008	20114	27.8	37.0	9.0

[13]:
```
ga_p.head()
```

[13]:

	StateAbbr	StateDesc	CountyName	CountyFIPS	TotalPopulation	harthritis
387	GA	Georgia	Appling	13001	18507	32.2

```
388        GA   Georgia   Atkinson       13003             8297        29.0
389        GA   Georgia      Bacon       13005            11185        30.5
390        GA   Georgia      Baker       13007             3092        32.6
391        GA   Georgia    Baldwin       13009            44823        27.2

     hasthma  hbphigh  hcancer  hhighchol  hkidney  hcopd  hchd  hdiabetes  \
387     10.5     39.0      7.4       37.8      4.1   11.8  10.0       16.3
388     10.7     38.3      6.5       37.3      4.1   11.8   9.8       16.8
389     10.4     37.1      7.1       36.2      3.8   11.3   9.3       15.2
390     10.3     44.2      8.0       38.3      4.3   10.4   9.7       18.2
391     10.5     38.4      6.3       34.8      3.6    9.1   8.1       14.9

     hmhlth  hphlth  hteethlost  hstroke
387    16.7    18.2        24.6      5.1
388    17.7    18.8        28.1      5.1
389    16.8    17.6        22.9      4.8
390    14.3    16.8        21.3      5.6
391    15.9    15.0        20.8      4.5
```

2 Saving a dataframe to csv

[ ]:
```
# Save ga_a and ga_p as csv files
ga_a.to_csv('data/ga_a.csv')
ga_p.to_csv('data/ga_p.csv')
```

3 Merge, .join , Concatenate

3.1 Making the 'table' wider (i.e., adding columns from. a second source)

- merge() for combining data on common columns or indices
- .join() for combining data on a key column or an index (uses merge internally, faster because index is used)
- concat() for combining DataFrames across rows or columns

[15]:
```
df1 = {
    'location':['bolton','berlin','boyleston','charlton'],
    'apples': [3, 2, 0, 1],
    'pears': [0, 3, 7, 2]
}

df2 = {
    'location':['bolton','berlin','boyleston','charlton'],
    'blueberries': [3, 2, 0, 1],
    'strawberries': [0, 3, 7, 2]
}
```

```
d1 = pd.DataFrame(df1)
d2 = pd.DataFrame(df2)
d1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   location  4 non-null      object
 1   apples    4 non-null      int64
 2   pears     4 non-null      int64
dtypes: int64(2), object(1)
memory usage: 224.0+ bytes
```

[16]:
```
d2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   location      4 non-null      object
 1   blueberries   4 non-null      int64
 2   strawberries  4 non-null      int64
dtypes: int64(2), object(1)
memory usage: 224.0+ bytes
```

3.2 Merge

3.2.1 Basic merge

[17]:
```
pd.merge(d1,d2,how = 'inner')  # Inner, outer, left, right
# When not being explicit, the merge is based on the index for each dataframe.
```

[17]:

	location	apples	pears	blueberries	strawberries
0	bolton	3	0	3	0
1	berlin	2	3	2	3
2	boyleston	0	7	0	7
3	charlton	1	2	1	2

More info: https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html#brief-primer-on-merge-methods-relational-algebra

3.2.2 Specifying columns to use for the merge

[18]:
```
df3 = {
    'state':['MA','MA','VT','VT'],
    'location':['bolton','berlin','boyleston','berlin'],
    'apples': [3, 2, 0, 1],
    'pears': [0, 3, 7, 2]
}

df4 = {
    'state':['MA','MA','VT','VT'],
    'location':['bolton','berlin','boyleston','berlin'],
    'blueberries': [3, 2, 0, 1],
    'strawberries': [0, 3, 7, 2]
}

d3 = pd.DataFrame(df3)
d4 = pd.DataFrame(df4)
```

[19]:
```
multi = pd.merge(d3,d4, how = 'inner', on = ['state','location'])
multi
```

[19]:

	state	location	apples	pears	blueberries	strawberries
0	MA	bolton	3	0	3	0
1	MA	berlin	2	3	2	3
2	VT	boyleston	0	7	0	7
3	VT	berlin	1	2	1	2

3.3 .join

[20]:
```
# basic syntax - first_dataframe.join(to second dataframe)
d1.join(d2, lsuffix = '_1')

# d1.join(d2, lsuffix = '_1', rsuffix = '_2')
```

[20]:

	location_1	apples	pears	location	blueberries	strawberries
0	bolton	3	0	bolton	3	0
1	berlin	2	3	berlin	2	3
2	boyleston	0	7	boyleston	0	7
3	charlton	1	2	charlton	1	2

[21]:
```
# If you want to use join() and want to merge the columns, you must set them to␣
 ↪be indexes first.

d1.join(d2.set_index('location'), on='location', lsuffix = '_1', rsuffix = '_2')
```

[21]:

	location	apples	pears	blueberries	strawberries
0	bolton	3	0	3	0
1	berlin	2	3	2	3
2	boyleston	0	7	0	7
3	charlton	1	2	1	2

[22]:
```
d3.join(d4.set_index(['state','location']), on=['state','location'],␣
↪lsuffix='_3')
```

[22]:

	state	location	apples	pears	blueberries	strawberries
0	MA	bolton	3	0	3	0
1	MA	berlin	2	3	2	3
2	VT	boyleston	0	7	0	7
3	VT	berlin	1	2	1	2

pd.merge() vs dataframe.join() vs dataframe.merge()

TL;DR: `pd.merge()` is the most generic. `df.merge()` is the same as `pd.merge()` with an implicit left dataframe. Use `df.join()` for merging on index columns exclusively. `df.join` is **much faster** because it joins by index

These are **three different ways** to do merging/joining dataframes on pandas:

	pandas.merge	dataframe.join	dataframe.merge
How to call	Pandas global method	Dataframe method	Dataframe method
Join on	Join on any column	Join on index columns only	Join on any column
Performance	Slow unless using indices	**Fast!**	Slow unless using indices
Example	`pd.merge(left_df, right_df)`	`left_df.join(right_df)`	`left_df.merge(right_df)`

https://queirozf.com/entries/pandas-dataframes-merge-join-examples

3.4 Other merge/join options

3.4.1 Column names not the same

[23]:
```
df5 = {
    'state':['MA','MA','VT','VT'],
    'location':['bolton','berlin','boyleston','berlin'],
    'apples': [3, 2, 0, 1],
    'pears': [0, 3, 7, 2]
}
```

```
df6 = {
    'states':['MA','MA','VT','VT'],
    'loc':['bolton','berlin','boyleston','berlin'],
    'blueberries': [3, 2, 0, 1],
    'strawberries': [0, 3, 7, 2]
}

d5 = pd.DataFrame(df5)
d6 = pd.DataFrame(df6)
```

[24]:
```
# Use left_on, right_on instead of on.

pd.merge(d5, d6, left_on = ['state','location'], right_on = ['states','loc'])

# Would it be better or easier to just rename the columns?
```

[24]:

	state	location	apples	pears	states	loc	blueberries	strawberries
0	MA	bolton	3	0	MA	bolton	3	0
1	MA	berlin	2	3	MA	berlin	2	3
2	VT	boyleston	0	7	VT	boyleston	0	7
3	VT	berlin	1	2	VT	berlin	1	2

3.4.2 Outer join

[25]:
```
df7 = {
    'state':['MA','MA','VT','NH'],
    'location':['bolton','berlin','boyleston','berlin'],
    'apples': [3, 2, 0, 1],
    'pears': [0, 3, 7, 2]
}

df8 = {
    'state':['MA','MA','VT','ME'],
    'location':['bolton','berlin','boyleston','berlin'],
    'blueberries': [3, 2, 0, 1],
    'strawberries': [0, 3, 7, 2]
}

d7 = pd.DataFrame(df7)
d8 = pd.DataFrame(df8)
```

[26]:
```
outer = pd.merge(d7,d8, how = 'outer', on = ['state','location'])
outer
```

[26]:

	state	location	apples	pears	blueberries	strawberries
0	MA	bolton	3.0	0.0	3.0	0.0
1	MA	berlin	2.0	3.0	2.0	3.0

```
2    VT  boyleston     0.0    7.0          0.0           7.0
3    NH     berlin     1.0    2.0          NaN           NaN
4    ME     berlin     NaN    NaN          1.0           2.0
```

3.4.3 Left join

[27]:
```
left = pd.merge(d7,d8, how = 'left', on = ['state','location'])
left

# Notice the Nan values
```

[27]:

	state	location	apples	pears	blueberries	strawberries
0	MA	bolton	3	0	3.0	0.0
1	MA	berlin	2	3	2.0	3.0
2	VT	boyleston	0	7	0.0	7.0
3	NH	berlin	1	2	NaN	NaN

3.4.4 Right join

[28]:
```
right = pd.merge(d7,d8, how = 'right', on = ['state','location'])
right
```

[28]:

	state	location	apples	pears	blueberries	strawberries
0	MA	bolton	3.0	0.0	3	0
1	MA	berlin	2.0	3.0	2	3
2	VT	boyleston	0.0	7.0	0	7
3	ME	berlin	NaN	NaN	1	2

3.4.5 Same code using .join

[29]:
```
# Outer join

d7.join(d8.set_index(['state','location']), on=['state','location'],␣
 ↪lsuffix='_3',how= 'outer')
```

[29]:

	state	location	apples	pears	blueberries	strawberries
0	MA	bolton	3.0	0.0	3.0	0.0
1	MA	berlin	2.0	3.0	2.0	3.0
2	VT	boyleston	0.0	7.0	0.0	7.0
3	NH	berlin	1.0	2.0	NaN	NaN
3	ME	berlin	NaN	NaN	1.0	2.0

[30]:
```
# Left join

d7.join(d8.set_index(['state','location']), on=['state','location'],␣
 ↪lsuffix='_3',how= 'left')
```

[30]:

	state	location	apples	pears	blueberries	strawberries
0	MA	bolton	3	0	3.0	0.0
1	MA	berlin	2	3	2.0	3.0
2	VT	boyleston	0	7	0.0	7.0
3	NH	berlin	1	2	NaN	NaN

[31]:
```
# Right join

d7.join(d8.set_index(['state','location']), on=['state','location'],␣
↪lsuffix='_3',how= 'right')
```

[31]:

	state	location	apples	pears	blueberries	strawberries
0	MA	bolton	3.0	0.0	3	0
1	MA	berlin	2.0	3.0	2	3
2	VT	boyleston	0.0	7.0	0	7
3	ME	berlin	NaN	NaN	1	2

3.5 Append (aka union)

Stack datasets on top of one another

[32]:
```
df9 = {
    'month':['Oct','Oct','Oct','Oct'],
    'location':['bolton','berlin','boyleston','charlton'],
    'apples': [3, 2, 0, 1],
    'pears': [0, 3, 7, 2]
}

df10 = {
    'month':['Nov','Nov','Nov','Nov'],
    'location':['bolton','berlin','boyleston','charlton'],
    'apples': [3, 2, 0, 1],
    'pears': [0, 3, 7, 2]
}

d9 = pd.DataFrame(df9)
d10 = pd.DataFrame(df10)
```

[33]:
```
d9.append(d10)
```

```
/var/folders/bg/jzzhjp857hv08kcqg3jdptcr0000gn/T/ipykernel_38194/4057771359.py:1
: FutureWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  d9.append(d10)
```

[33]:

	month	location	apples	pears
0	Oct	bolton	3	0

```
1   Oct     berlin       2      3
2   Oct  boyleston       0      7
3   Oct   charlton       1      2
0   Nov     bolton       3      0
1   Nov     berlin       2      3
2   Nov  boyleston       0      7
3   Nov   charlton       1      2
```

3.6 Concatenate

With concatenation, your datasets are just stitched together along an axis — either the row axis or column axis

[34]:
```
pd.concat([d9,d10])  # Need to pass in a *list* of dataframes
```

[34]:
```
  month   location  apples  pears
0   Oct     bolton       3      0
1   Oct     berlin       2      3
2   Oct  boyleston       0      7
3   Oct   charlton       1      2
0   Nov     bolton       3      0
1   Nov     berlin       2      3
2   Nov  boyleston       0      7
3   Nov   charlton       1      2
```

Concat can work along either axis

[35]:
```
pd.concat([d9,d10], axis = 1)
```

[35]:
```
  month   location  apples  pears month   location  apples  pears
0   Oct     bolton       3      0   Nov     bolton       3      0
1   Oct     berlin       2      3   Nov     berlin       2      3
2   Oct  boyleston       0      7   Nov  boyleston       0      7
3   Oct   charlton       1      2   Nov   charlton       1      2
```

[36]:
```
pd.concat([d7,d8], axis = 1)  # Be careful!
```

[36]:
```
  state   location  apples  pears state   location  blueberries  strawberries
0    MA     bolton       3      0    MA     bolton            3             0
1    MA     berlin       2      3    MA     berlin            2             3
2    VT  boyleston       0      7    VT  boyleston            0             7
3    NH     berlin       1      2    ME     berlin            1             2
```

[37]:
```
pd.concat([d7,d8], join = 'inner', axis =1)
```

[37]:
```
  state   location  apples  pears state   location  blueberries  strawberries
0    MA     bolton       3      0    MA     bolton            3             0
1    MA     berlin       2      3    MA     berlin            2             3
```

```
2    VT  boyleston       0      7    VT  boyleston           0            7
3    NH     berlin       1      2    ME     berlin           1            2
```

4 Merge Exercise - 20 minutes

[38]:
```
# ga_a (acs data for Georgia) and ga_p (PLACES data for Georgia) each have 159␣
 ↪rows.
# We might assume each df contains the same list of counties but we can't be␣
 ↪sure.
# Merge the datasets together using an inner join and then an outer join.
# Do the shapes (dimensions) of the merged datasets change?
# Do they contain the same number of counties?
```

5 Return to the Beer notebook and complete part 3

[ ]:
```

```