3. Joining data in pandas

February 6, 2021

1 3. Joining data in pandas

In this notebook, we'll use pandas to join some relational data: - `../data/country-codes.csv` – a table of ISO country codes and country names - `../data/country-population.csv` – country population data from the U.N.

Read more about the `merge` method for joining dataframes

[8]:
```
import pandas as pd
```

When we read in the CSVs, we need to make sure that pandas doesn't parse the ISO codes as numbers, because we want to keep any leading zeroes. So in addition to the path to the CSV, we'll also use an argument called `dtype` to specify that the `code` columns need to be parsed as a string.

You can find more information on the `dtype` argument in the documentation for the `read_csv()` method

[9]:
```
country_codes = pd.read_csv('../data/country-codes.csv', dtype={'code': str})
```

[10]:
```
country_codes.head()
```

[10]:
```
  code   country
0  108   Burundi
1  174   Comoros
2  262  Djibouti
3  232   Eritrea
4  231  Ethiopia
```

[11]:
```
country_pop = pd.read_csv('../data/country-population.csv', dtype={'code': str})
```

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

[12]:
```
  code  pop2000  pop2001  pop2002  pop2003  pop2004  pop2005  pop2006  \
0  108   6401.0   6556.0   6742.0   6953.0   7182.0   7423.0   7675.0
1  174    542.0    556.0    569.0    583.0    597.0    612.0    626.0
2  262    718.0    733.0    746.0    759.0    771.0    783.0    796.0
3  232   3393.0   3497.0   3615.0   3738.0   3859.0   3969.0   4067.0
4  231  66537.0  68492.0  70497.0  72545.0  74624.0  76727.0  78851.0
```

```
   pop2007  pop2008  pop2009  pop2010  pop2011  pop2012  pop2013  pop2014  \
0   7940.0   8212.0   8489.0   8767.0   9044.0   9320.0   9600.0   9892.0
1    642.0    657.0    673.0    690.0    707.0    724.0    742.0    759.0
2    809.0    823.0    837.0    851.0    866.0    881.0    897.0    912.0
3   4153.0   4233.0   4310.0   4391.0   4475.0   4561.0   4651.0   4746.0
4  81000.0  83185.0  85416.0  87703.0  90047.0  92444.0  94888.0  97367.0

   pop2015
0  10199.0
1    777.0
2    927.0
3   4847.0
4  99873.0
```

1.0.1 Join the data with the country codes lookup table

To join data in pandas, we can use the `merge()` method. At minimum, you need to hand this method the two dataframes to join, plus specify the name of the column to join `on`. (If the columns have different names, you can use the `left_on` and `right_on` arguments – the "left" dataframe is the first one you hand to the `merge` method.)

[13]:
```
merged = pd.merge(country_pop,
                  country_codes,
                  on='code')
```

[14]:
```
merged.head()
```

[14]:
```
  code  pop2000  pop2001  pop2002  pop2003  pop2004  pop2005  pop2006  \
0  108   6401.0   6556.0   6742.0   6953.0   7182.0   7423.0   7675.0
1  174    542.0    556.0    569.0    583.0    597.0    612.0    626.0
2  262    718.0    733.0    746.0    759.0    771.0    783.0    796.0
3  232   3393.0   3497.0   3615.0   3738.0   3859.0   3969.0   4067.0
4  231  66537.0  68492.0  70497.0  72545.0  74624.0  76727.0  78851.0

   pop2007  pop2008  pop2009  pop2010  pop2011  pop2012  pop2013  pop2014  \
0   7940.0   8212.0   8489.0   8767.0   9044.0   9320.0   9600.0   9892.0
1    642.0    657.0    673.0    690.0    707.0    724.0    742.0    759.0
2    809.0    823.0    837.0    851.0    866.0    881.0    897.0    912.0
3   4153.0   4233.0   4310.0   4391.0   4475.0   4561.0   4651.0   4746.0
4  81000.0  83185.0  85416.0  87703.0  90047.0  92444.0  94888.0  97367.0

   pop2015   country
0  10199.0   Burundi
1    777.0   Comoros
2    927.0  Djibouti
3   4847.0   Eritrea
```

```
4  99873.0  Ethiopia
```

1.0.2 Your turn

In the cells below, read in these two datasets and merge them: - `../data/sdr-maintable.csv`: The main table of information from the Service Difficulty Reporting database maintained by the FAA. - `../data/sdr-opcode.csv`: The lookup table that maps airline codes to airline names.

You'll want to join on the `OPCODE` column in the `sdr-maintable.csv` file and on the `CODE` column for the `sdr-opcode.csv` file, so you'll need to use the `left_on` and `right_on` arguments rather than `on`. Assign your newly joined dataframe to a new variable name.

Then: - Select the columns you'd like to export to file - Export the joined file to a CSV

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

1.0.3 Joining on multiple columns

You can join on multiple columns, which can be useful when conducting an enterprise join to hunt for leads. Just pass in a list to the `on`/`left_on`/`right_on` arguments instead of a string, like this:

```
merged = pd.merge(df1,
                  df2,
                  left_on=['lname', 'fname', 'zipcode'],
                  right_on=['last_name', 'first_name', 'zip'])
```