lect6 Pandas

September 20, 2016

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

```
In [2]: !head primary_results.csv
```

```
state,state_abbreviation,county,fips,party,candidate,votes,fraction_votes
Alabama,AL,Autauga,1001.0,Democrat,Bernie Sanders,544,0.182
Alabama,AL,Autauga,1001.0,Democrat,Hillary Clinton,2387,0.8
Alabama,AL,Baldwin,1003.0,Democrat,Bernie Sanders,2694,0.32899999999999996
Alabama,AL,Baldwin,1003.0,Democrat,Hillary Clinton,5290,0.647
Alabama,AL,Barbour,1005.0,Democrat,Bernie Sanders,222,0.078
Alabama,AL,Barbour,1005.0,Democrat,Hillary Clinton,2567,0.9059999999999999
Alabama,AL,Bibb,1007.0,Democrat,Bernie Sanders,246,0.19699999999999998
Alabama,AL,Bibb,1007.0,Democrat,Hillary Clinton,942,0.755
Alabama,AL,Blount,1009.0,Democrat,Bernie Sanders,395,0.386
```

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

Out[3]:

	fips	area_name	state_abbreviation	PST045214	PST040210	\
0	0	United States	NaN	318857056	308758105	
1	1000	Alabama	NaN	4849377	4780127	
2	1001	Autauga County	AL	55395	54571	
3	1003	Baldwin County	AL	200111	182265	
4	1005	Barbour County	AL	26887	27457	
5	1007	Bibb County	AL	22506	22919	
6	1009	Blount County	AL	57719	57322	
7	1011	Bullock County	AL	10764	10915	
8	1013	Butler County	AL	20296	20946	
9	1015	Calhoun County	AL	115916	118586	
10	1017	Chambers County	AL	34076	34170	
11	1019	Cherokee County	AL	26037	25986	
12	1021	Chilton County	AL	43931	43631	
13	1023	Choctaw County	AL	13323	13858	
14	1025	Clarke County	AL	24945	25840	

15	1027	Clay County	AL	13552	13932
16	1029	Cleburne County	AL	15080	14972
17	1031	Coffee County	AL	50909	49948
18	1033	Colbert County	AL	54543	54428
19	1035	Conecuh County	AL	12670	13228
20	1037	Coosa County	AL	10886	11758
21	1039	Covington County	AL	37914	37765
22	1041	Crenshaw County	AL	13977	13906
23	1043	Cullman County	AL	81289	80410
24	1045	Dale County	AL	49484	50251
25	1047	Dallas County	AL	41711	43820
26	1049	DeKalb County	AL	71065	71115
27	1051	Elmore County	AL	80977	79296
28	1053	Escambia County	AL	37733	38319
29	1055	Etowah County	AL	103531	104427
...	...	...	...	...	...
3165	55131	Washington County	WI	133251	131885
3166	55133	Waukesha County	WI	395118	389938
3167	55135	Waupaca County	WI	52066	52410
3168	55137	Waushara County	WI	24178	24496
3169	55139	Winnebago County	WI	169511	166994
3170	55141	Wood County	WI	73608	74749
3171	56000	Wyoming	NaN	584153	563767
3172	56001	Albany County	WY	37811	36299
3173	56003	Big Horn County	WY	11930	11668
3174	56005	Campbell County	WY	48320	46133
3175	56007	Carbon County	WY	15854	15885
3176	56009	Converse County	WY	14097	13833
3177	56011	Crook County	WY	7248	7083
3178	56013	Fremont County	WY	40703	40123
3179	56015	Goshen County	WY	13514	13247
3180	56017	Hot Springs County	WY	4816	4812
3181	56019	Johnson County	WY	8573	8569
3182	56021	Laramie County	WY	96389	91881
3183	56023	Lincoln County	WY	18567	18106
3184	56025	Natrona County	WY	81624	75450
3185	56027	Niobrara County	WY	2463	2484
3186	56029	Park County	WY	28989	28205
3187	56031	Platte County	WY	8799	8667
3188	56033	Sheridan County	WY	30032	29116
3189	56035	Sublette County	WY	10057	10247
3190	56037	Sweetwater County	WY	45010	43806
3191	56039	Teton County	WY	22930	21294
3192	56041	Uinta County	WY	20904	21118
3193	56043	Washakie County	WY	8322	8533
3194	56045	Weston County	WY	7201	7208

```
      PST120214  POP010210  AGE135214  AGE295214  AGE775214    ...      \
```

0	3.3	308745538	6.2	23.1	14.5	...
1	1.4	4779736	6.1	22.8	15.3	...
2	1.5	54571	6.0	25.2	13.8	...
3	9.8	182265	5.6	22.2	18.7	...
4	-2.1	27457	5.7	21.2	16.5	...
5	-1.8	22915	5.3	21.0	14.8	...
6	0.7	57322	6.1	23.6	17.0	...
7	-1.4	10914	6.3	21.4	14.9	...
8	-3.1	20947	6.1	23.6	18.0	...
9	-2.3	118572	5.7	22.2	16.0	...
10	-0.3	34215	5.9	21.4	18.3	...
11	0.2	25989	4.8	20.4	20.9	...
12	0.7	43643	6.4	24.2	15.2	...
13	-3.9	13859	4.9	20.6	20.8	...
14	-3.5	25833	5.6	22.6	18.0	...
15	-2.7	13932	5.3	21.6	19.4	...
16	0.7	14972	5.7	23.3	18.4	...
17	1.9	49948	6.1	23.7	15.9	...
18	0.2	54428	5.7	21.7	18.5	...
19	-4.2	13228	5.5	21.4	20.4	...
20	-7.4	11539	4.8	18.3	18.5	...
21	0.4	37765	5.9	22.1	19.8	...
22	0.5	13906	6.1	22.7	18.0	...
23	1.1	80406	5.9	22.5	17.6	...
24	-1.5	50251	6.3	23.6	15.4	...
25	-4.8	43820	6.7	25.2	15.6	...
26	-0.1	71109	6.2	24.7	15.7	...
27	2.1	79303	5.9	23.0	14.0	...
28	-1.5	38319	6.1	22.0	16.7	...
29	-0.9	104430	5.7	22.1	17.3	...
...	...	...	...	...	...	...
3165	1.0	131887	5.3	23.0	15.7	...
3166	1.3	389891	5.1	22.4	16.5	...
3167	-0.7	52410	5.2	21.2	19.5	...
3168	-1.3	24496	4.8	18.7	22.5	...
3169	1.5	166994	5.6	20.8	14.7	...
3170	-1.5	74749	5.7	21.9	18.6	...
3171	3.6	563626	6.5	23.7	14.0	...
3172	4.2	36299	5.2	16.5	9.8	...
3173	2.2	11668	6.5	25.3	19.2	...
3174	4.7	46133	8.0	28.0	7.1	...
3175	-0.2	15885	6.2	23.9	13.8	...
3176	1.9	13833	6.9	24.7	14.1	...
3177	2.3	7083	6.6	23.5	18.0	...
3178	1.4	40123	7.5	25.5	16.2	...
3179	2.0	13249	5.6	20.1	20.5	...
3180	0.1	4812	4.4	20.8	23.6	...
3181	0.0	8569	5.8	22.0	20.6	...

3182	4.9	91738	6.5	23.5	14.3	...
3183	2.5	18106	6.6	27.0	14.8	...
3184	8.2	75450	6.8	23.9	12.8	...
3185	-0.8	2484	4.7	16.6	21.2	...
3186	2.8	28205	5.5	20.5	20.1	...
3187	1.5	8667	4.9	20.4	22.7	...
3188	3.1	29116	5.6	21.6	18.3	...
3189	-1.9	10247	5.8	24.0	12.6	...
3190	2.7	43806	7.3	27.0	9.5	...
3191	7.7	21294	5.7	19.1	12.2	...
3192	-1.0	21118	7.6	29.8	11.0	...
3193	-2.5	8533	5.5	23.9	20.1	...
3194	-0.1	7208	6.5	21.6	18.1	...

	SBO415207	SBO015207	MAN450207	WTN220207	RTN130207	RTN131207
0	8.3	28.8	5319456312	4174286516	3917663456	12990
1	1.2	28.1	112858843	52252752	57344851	12364
2	0.7	31.7	0	0	598175	12003
3	1.3	27.3	1410273	0	2966489	17166
4	0.0	27.0	0	0	188337	6334
5	0.0	0.0	0	0	124707	5804
6	0.0	23.2	341544	0	319700	5622
7	0.0	38.8	0	0	43810	3995
8	0.0	0.0	399132	56712	229277	11326
9	0.5	24.7	2679991	0	1542981	13678
10	0.0	29.3	667283	0	264650	7620
11	0.0	14.5	307439	62293	186321	7613
12	0.0	0.0	0	155139	359910	8496
13	0.0	44.1	0	52904	84633	5969
14	0.0	28.6	571454	85803	344311	13034
15	0.0	0.0	330878	0	70558	5123
16	0.0	17.7	0	5888	133407	9080
17	0.7	29.3	613758	131594	639623	13665
18	0.0	22.8	2083166	351388	821284	15040
19	0.0	23.0	183544	76736	71232	5430
20	0.0	0.0	89585	31647	20531	1904
21	0.0	0.0	561168	338692	424873	11544
22	0.0	30.2	270809	131874	75456	5491
23	0.0	23.9	1393437	0	867058	10757
24	0.0	0.0	114991	69341	332503	6889
25	0.0	28.8	1174763	0	437359	10223
26	0.0	25.8	1831626	0	635185	9303
27	0.0	28.3	766344	104885	618313	7982
28	0.0	21.3	665336	93125	379587	10089
29	0.6	25.7	1252962	518424	1114841	10801
...	...	...	...	...	...	...
3165	0.4	19.9	3393434	2800492	1759993	13729
3166	1.0	24.7	15663482	6882092	5955006	15712

3167	0.8	24.6	2133603	86158	516629	9929
3168	0.0	23.9	111250	49841	203004	8226
3169	0.8	23.4	9198861	1109192	1796479	11126
3170	0.0	25.3	2562632	893219	1002909	13585
3171	2.8	25.5	8834810	6352890	8957553	17114
3172	3.9	32.2	0	113498	468773	14433
3173	0.0	30.3	0	0	71008	6242
3174	2.1	28.2	262454	917314	775117	19157
3175	0.0	14.7	0	51494	295340	19182
3176	2.6	26.6	0	0	134349	10436
3177	0.0	16.8	0	14112	56351	8842
3178	2.4	22.3	0	76180	558410	14944
3179	0.0	23.8	147899	52428	116123	9614
3180	0.0	28.2	0	0	38266	8502
3181	0.0	0.0	0	0	71779	8868
3182	4.7	29.8	2420198	628089	1720451	19866
3183	0.0	21.2	0	26778	221712	13705
3184	2.0	24.0	1230635	2889434	1377032	19174
3185	0.0	0.0	0	0	19649	8408
3186	1.8	22.5	62727	155545	388380	14271
3187	0.0	33.0	0	34092	93405	11163
3188	0.0	21.7	0	0	478180	17110
3189	0.0	17.3	0	0	82792	10563
3190	3.8	27.2	0	437493	898189	22843
3191	3.3	25.3	0	0	515644	25688
3192	2.2	15.9	0	159375	413983	20626
3193	0.0	26.9	0	12128	98308	12596
3194	0.0	29.3	0	11540	64312	9395

	AFN120207	BPS030214	LND110210	POP060210
0	613795732	1046363	3531905.43	87.4
1	6426342	13369	50645.33	94.4
2	88157	131	594.44	91.8
3	436955	1384	1589.78	114.6
4	0	8	884.88	31.0
5	10757	19	622.58	36.8
6	20941	3	644.78	88.9
7	3670	1	622.81	17.5
8	28427	2	776.83	27.0
9	186533	114	605.87	195.7
10	23237	8	596.53	57.4
11	13948	2	553.70	46.9
12	34073	78	692.85	63.0
13	11345	0	913.50	15.2
14	23596	11	1238.47	20.9
15	4352	0	603.96	23.1
16	4134	1	560.10	26.7
17	50080	58	678.97	73.6

```
18        61935        143      592.62       91.8
19         8667          3      850.16       15.6
20          575          0      650.93       17.7
21        30271          8     1030.46       36.6
22         5640          2      608.84       22.8
23       102291         77      734.84      109.4
24        37229         17      561.15       89.6
25        33708         17      978.70       44.8
26        67915         31      777.09       91.5
27        47884        259      618.49      128.2
28        31909         26      945.08       40.5
29       137966         79      534.99      195.2
...         ...        ...         ...        ...
3165     152903        337      430.70      306.2
3166     616641       1020      549.57      709.4
3167      55597         88      747.71       70.1
3168      25080         54      626.15       39.1
3169     212069        511      434.49      384.3
3170      84291        236      793.12       94.2
3171    1469008       1901    97093.14        5.8
3172      65787         99     4273.84        8.5
3173       5611         13     3137.10        3.7
3174      89245        163     4802.71        9.6
3175      42949         31     7897.58        2.0
3176      20947         43     4254.88        3.3
3177       6135          5     2854.41        2.5
3178      66975         18     9183.81        4.4
3179      11779          1     2225.39        6.0
3180      10368          0     2004.09        2.4
3181      20749         12     4154.15        2.1
3182     173722        403     2685.91       34.2
3183      21115         68     4076.13        4.4
3184     167974        330     5340.35       14.1
3185       3071          1     2626.04        0.9
3186     132951        131     6942.08        4.1
3187      12103         22     2084.21        4.2
3188      66787        117     2523.99       11.5
3189      19746         27     4886.54        2.1
3190     150439        227    10426.65        4.2
3191     327363        145     3995.38        5.3
3192      35497         40     2081.26       10.1
3193      10175          4     2238.55        3.8
3194       7520          1     2398.09        3.0

[3195 rows x 54 columns]
```

```
In [4]: facts.shape
```

```
Out[4]: (3195, 54)
```

In [5]:
```
facts.head()
```

Out[5]:

	fips	area_name	state_abbreviation	PST045214	PST040210	PST120214
0	0	United States	NaN	318857056	308758105	3.3
1	1000	Alabama	NaN	4849377	4780127	1.4
2	1001	Autauga County	AL	55395	54571	1.5
3	1003	Baldwin County	AL	200111	182265	9.8
4	1005	Barbour County	AL	26887	27457	-2.1

	POP010210	AGE135214	AGE295214	AGE775214	...	SBO415207	\
0	308745538	6.2	23.1	14.5	...	8.3	
1	4779736	6.1	22.8	15.3	...	1.2	
2	54571	6.0	25.2	13.8	...	0.7	
3	182265	5.6	22.2	18.7	...	1.3	
4	27457	5.7	21.2	16.5	...	0.0	

	SBO015207	MAN450207	WTN220207	RTN130207	RTN131207	AFN120207	\
0	28.8	5319456312	4174286516	3917663456	12990	613795732	
1	28.1	112858843	52252752	57344851	12364	6426342	
2	31.7	0	0	598175	12003	88157	
3	27.3	1410273	0	2966489	17166	436955	
4	27.0	0	0	188337	6334	0	

	BPS030214	LND110210	POP060210
0	1046363	3531905.43	87.4
1	13369	50645.33	94.4
2	131	594.44	91.8
3	1384	1589.78	114.6
4	8	884.88	31.0

[5 rows x 54 columns]

In [6]:
```
fact_dict = pd.read_csv("county_facts_dictionary.csv")
fact_dict
```

Out[6]:

	column_name	description
0	PST045214	Population, 2014 estimate
1	PST040210	Population, 2010 (April 1) estimates base
2	PST120214	Population, percent change - April 1, 2010 to ...
3	POP010210	Population, 2010
4	AGE135214	Persons under 5 years, percent, 2014
5	AGE295214	Persons under 18 years, percent, 2014
6	AGE775214	Persons 65 years and over, percent, 2014
7	SEX255214	Female persons, percent, 2014
8	RHI125214	White alone, percent, 2014
9	RHI225214	Black or African American alone, percent, 2014
10	RHI325214	American Indian and Alaska Native alone, perce...
11	RHI425214	Asian alone, percent, 2014

```
12   RHI525214  Native Hawaiian and Other Pacific Islander alo...
13   RHI625214                  Two or More Races, percent, 2014
14   RHI725214                Hispanic or Latino, percent, 2014
15   RHI825214  White alone, not Hispanic or Latino, percent, ...
16   POP715213  Living in same house 1 year & over, percent, 2...
17   POP645213           Foreign born persons, percent, 2009-2013
18   POP815213  Language other than English spoken at home, pc...
19   EDU635213  High school graduate or higher, percent of per...
20   EDU685213  Bachelor's degree or higher, percent of person...
21   VET605213                                Veterans, 2009-2013
22   LFE305213  Mean travel time to work (minutes), workers ag...
23   HSG010214                                Housing units, 2014
24   HSG445213                    Homeownership rate, 2009-2013
25   HSG096213  Housing units in multi-unit structures, percen...
26   HSG495213  Median value of owner-occupied housing units, ...
27   HSD410213                             Households, 2009-2013
28   HSD310213                  Persons per household, 2009-2013
29   INC910213  Per capita money income in past 12 months (201...
30   INC110213              Median household income, 2009-2013
31   PVY020213   Persons below poverty level, percent, 2009-2013
32   BZA010213           Private nonfarm establishments, 2013
33   BZA110213             Private nonfarm employment,  2013
34   BZA115213  Private nonfarm employment, percent change, 20...
35   NES010213                  Nonemployer establishments, 2013
36   SBO001207                       Total number of firms, 2007
37   SBO315207                  Black-owned firms, percent, 2007
38   SBO115207  American Indian- and Alaska Native-owned firms...
39   SBO215207                  Asian-owned firms, percent, 2007
40   SBO515207  Native Hawaiian- and Other Pacific Islander-ow...
41   SBO415207               Hispanic-owned firms, percent, 2007
42   SBO015207                  Women-owned firms, percent, 2007
43   MAN450207            Manufacturers shipments, 2007 ($1,000)
44   WTN220207          Merchant wholesaler sales, 2007 ($1,000)
45   RTN130207                      Retail sales, 2007 ($1,000)
46   RTN131207                    Retail sales per capita, 2007
47   AFN120207  Accommodation and food services sales, 2007 ($...
48   BPS030214                                Building permits, 2014
49   LND110210                 Land area in square miles, 2010
50   POP060210                 Population per square mile, 2010
```

In [7]:
```
# set the column name to the index, so we can index by it
fact_dict = fact_dict.set_index('column_name')
fact_dict.head()
```

Out[7]:

column_name	description
PST045214	Population, 2014 estimate
PST040210	Population, 2010 (April 1) estimates base

```
PST120214    Population, percent change - April 1, 2010 to ...
POP010210                                        Population, 2010
AGE135214                  Persons under 5 years, percent, 2014
```

```
In [8]: fact_dict.loc['AGE135214'].description
```

```
Out[8]: 'Persons under 5 years, percent, 2014'
```

```
In [9]: results = pd.read_csv("primary_results.csv")
        results.shape
```

```
Out[9]: (24611, 8)
```

```
In [10]: results.head()
```

Out[10]:

	state	state_abbreviation	county	fips	party	candidate	votes	fraction_votes
0	Alabama	AL	Autauga	1001.0	Democrat	Bernie Sanders	544	0.182
1	Alabama	AL	Autauga	1001.0	Democrat	Hillary Clinton	2387	0.800
2	Alabama	AL	Baldwin	1003.0	Democrat	Bernie Sanders	2694	0.329
3	Alabama	AL	Baldwin	1003.0	Democrat	Hillary Clinton	5290	0.647
4	Alabama	AL	Barbour	1005.0	Democrat	Bernie Sanders	222	0.078

```
In [11]: results.candidate.unique()
```

```
Out[11]: array(['Bernie Sanders', 'Hillary Clinton', 'Ben Carson', 'Donald Trump',
                'John Kasich', 'Marco Rubio', 'Ted Cruz', ' Uncommitted',
                "Martin O'Malley", 'Carly Fiorina', 'Chris Christie', 'Jeb Bush',
                'Mike Huckabee', 'Rand Paul', 'Rick Santorum', ' No Preference'], c
```

```
In [12]: # each column of a DataFrame is a Series
         candidates = results.candidate
         type(candidates)
```

```
Out[12]: pandas.core.series.Series
```

1 Using Dictionaries

We now use a dictionary to count how the number of counties in which each candidate received votes

```
In [13]: # First, let's explore how dictionaries work.
         dict = {}
         dict['Dan'] = 0
         dict['Sahand'] = 1
         dict
```

```
Out[13]: {'Dan': 0, 'Sahand': 1}

In [14]: dict['Sahand']

Out[14]: 1

In [15]: dict['Natalie']


         ---------------------------------------------------------------------------
         KeyError                                  Traceback (most recent call last)

         <ipython-input-15-f12b264cef27> in <module>()
     ----> 1 dict['Natalie']


         KeyError: 'Natalie'


In [16]: dict.has_key('Natalie')

Out[16]: False

In [17]: dict = {}
          for c in candidates:
              if dict.has_key(c):
                  dict[c] += 1
              else:
                  dict[c] = 1
          dict

Out[17]: {' No Preference': 351,
           ' Uncommitted': 99,
           'Ben Carson': 1669,
           'Bernie Sanders': 4205,
           'Carly Fiorina': 109,
           'Chris Christie': 109,
           'Donald Trump': 3586,
           'Hillary Clinton': 4205,
           'Jeb Bush': 155,
           'John Kasich': 3586,
           'Marco Rubio': 2555,
           "Martin O'Malley": 99,
           'Mike Huckabee': 99,
           'Rand Paul': 99,
           'Rick Santorum': 99,
           'Ted Cruz': 3586}

In [18]: type(dict)
```

```
Out[18]: dict
```

Sometimes dictionaries are incredibly useful. But, we will try to use DataFrame operations as much as possible.

1.1 Cleaning up facts

```
In [19]: facts.state_abbreviation.isnull()
```

```
Out[19]: 0        True
         1        True
         2       False
         3       False
         4       False
         5       False
         6       False
         7       False
         8       False
         9       False
         10      False
         11      False
         12      False
         13      False
         14      False
         15      False
         16      False
         17      False
         18      False
         19      False
         20      False
         21      False
         22      False
         23      False
         24      False
         25      False
         26      False
         27      False
         28      False
         29      False
                 ...
         3165    False
         3166    False
         3167    False
         3168    False
         3169    False
         3170    False
         3171     True
         3172    False
```

```
3173    False
3174    False
3175    False
3176    False
3177    False
3178    False
3179    False
3180    False
3181    False
3182    False
3183    False
3184    False
3185    False
3186    False
3187    False
3188    False
3189    False
3190    False
3191    False
3192    False
3193    False
3194    False
Name: state_abbreviation, dtype: bool
```

```
In [20]: facts[facts.state_abbreviation.isnull()]
```

Out[20]:

	fips	area_name	state_abbreviation	PST045214	PST040210
0	0	United States	NaN	318857056	308758105
1	1000	Alabama	NaN	4849377	4780127
69	2000	Alaska	NaN	736732	710249
99	4000	Arizona	NaN	6731484	6392310
115	5000	Arkansas	NaN	2966369	2915958
191	6000	California	NaN	38802500	37254503
250	8000	Colorado	NaN	5355866	5029324
315	9000	Connecticut	NaN	3596677	3574096
324	10000	Delaware	NaN	935614	897936
328	11000	District Of Columbia	NaN	658893	601767
330	12000	Florida	NaN	19893297	18804623
398	13000	Georgia	NaN	10097343	9688681
558	15000	Hawaii	NaN	1419561	1360301
564	16000	Idaho	NaN	1634464	1567652
609	17000	Illinois	NaN	12880580	12831587
712	18000	Indiana	NaN	6596855	6484192
805	19000	Iowa	NaN	3107126	3046869
905	20000	Kansas	NaN	2904021	2853132
1011	21000	Kentucky	NaN	4413457	4339349
1132	22000	Louisiana	NaN	4649676	4533479
1197	23000	Maine	NaN	1330089	1328361

1214	24000	Maryland	NaN	5976407	5773785
1239	25000	Massachusetts	NaN	6745408	6547817
1254	26000	Michigan	NaN	9909877	9884133
1338	27000	Minnesota	NaN	5457173	5303925
1426	28000	Mississippi	NaN	2994079	2968103
1509	29000	Missouri	NaN	6063589	5988923
1625	30000	Montana	NaN	1023579	989417
1682	31000	Nebraska	NaN	1881503	1826341
1776	32000	Nevada	NaN	2839099	2700692
1794	33000	New Hampshire	NaN	1326813	1316466
1805	34000	New Jersey	NaN	8938175	8791936
1827	35000	New Mexico	NaN	2085572	2059192
1861	36000	New York	NaN	19746227	19378112
1924	37000	North Carolina	NaN	9943964	9535691
2025	38000	North Dakota	NaN	739482	672591
2079	39000	Ohio	NaN	11594163	11536725
2168	40000	Oklahoma	NaN	3878051	3751616
2246	41000	Oregon	NaN	3970239	3831073
2283	42000	Pennsylvania	NaN	12787209	12702884
2351	44000	Rhode Island	NaN	1055173	1052931
2357	45000	South Carolina	NaN	4832482	4625401
2404	46000	South Dakota	NaN	853175	814191
2471	47000	Tennessee	NaN	6549352	6346275
2567	48000	Texas	NaN	26956958	25146104
2822	49000	Utah	NaN	2942902	2763885
2852	50000	Vermont	NaN	626562	625745
2867	51000	Virginia	NaN	8326289	8001023
3002	53000	Washington	NaN	7061530	6724543
3042	54000	West Virginia	NaN	1850326	1853033
3098	55000	Wisconsin	NaN	5757564	5687289
3171	56000	Wyoming	NaN	584153	563767

	PST120214	POP010210	AGE135214	AGE295214	AGE775214	...	\
0	3.3	308745538	6.2	23.1	14.5	...	
1	1.4	4779736	6.1	22.8	15.3	...	
69	3.7	710231	7.4	25.3	9.4	...	
99	5.3	6392017	6.4	24.1	15.9	...	
115	1.7	2915918	6.5	23.8	15.7	...	
191	4.2	37253956	6.5	23.6	12.9	...	
250	6.5	5029196	6.3	23.3	12.7	...	
315	0.6	3574097	5.3	21.6	15.5	...	
324	4.2	897934	6.0	21.8	16.4	...	
328	9.5	601723	6.5	17.5	11.3	...	
330	5.8	18801310	5.5	20.4	19.1	...	
398	4.2	9687653	6.6	24.7	12.4	...	
558	4.4	1360301	6.4	21.7	16.1	...	
564	4.3	1567582	7.0	26.4	14.3	...	
609	0.4	12830632	6.1	23.2	13.9	...	

712	1.7	6483802	6.4	24.0	14.3	...
805	2.0	3046355	6.3	23.4	15.8	...
905	1.8	2853118	6.9	24.9	14.3	...
1011	1.7	4339367	6.3	22.9	14.8	...
1132	2.6	4533372	6.6	23.9	13.6	...
1197	0.1	1328361	4.9	19.5	18.3	...
1214	3.5	5773552	6.2	22.6	13.8	...
1239	3.0	6547629	5.4	20.6	15.1	...
1254	0.3	9883640	5.8	22.4	15.4	...
1338	2.9	5303925	6.4	23.5	14.3	...
1426	0.9	2967297	6.5	24.4	14.3	...
1509	1.2	5988927	6.2	23.0	15.4	...
1625	3.5	989415	6.0	22.0	16.7	...
1682	3.0	1826341	6.9	24.8	14.4	...
1776	5.1	2700551	6.2	23.4	14.2	...
1794	0.8	1316470	4.9	20.1	15.9	...
1805	1.7	8791894	6.0	22.5	14.7	...
1827	1.3	2059179	6.6	24.1	15.3	...
1861	1.9	19378102	6.0	21.4	14.7	...
1924	4.3	9535483	6.1	23.0	14.7	...
2025	9.9	672591	6.9	22.8	14.2	...
2079	0.5	11536504	6.0	22.8	15.5	...
2168	3.4	3751351	6.8	24.6	14.5	...
2246	3.6	3831074	5.8	21.6	16.0	...
2283	0.7	12702379	5.6	21.1	16.7	...
2351	0.2	1052567	5.2	20.2	15.7	...
2357	4.5	4625364	6.0	22.4	15.8	...
2404	4.8	814180	7.1	24.7	15.3	...
2471	3.2	6346105	6.1	22.8	15.1	...
2567	7.2	25145561	7.3	26.4	11.5	...
2822	6.5	2763885	8.6	30.7	10.0	...
2852	0.1	625741	4.9	19.4	16.9	...
2867	4.1	8001024	6.2	22.4	13.8	...
3002	5.0	6724540	6.3	22.7	14.1	...
3042	−0.1	1852994	5.5	20.5	17.8	...
3098	1.2	5686986	5.9	22.6	15.2	...
3171	3.6	563626	6.5	23.7	14.0	...

	SBO415207	SBO015207	MAN450207	WTN220207	RTN130207	RTN131207
0	8.3	28.8	5319456312	4174286516	3917663456	12990
1	1.2	28.1	112858843	52252752	57344851	12364
69	0.0	25.9	8204030	4563605	9303387	13635
99	10.7	28.1	57977827	57573459	86758801	13637
115	2.3	24.5	60735582	29659789	32974282	11602
191	16.5	30.3	491372092	598456486	455032270	12561
250	6.2	29.2	46331953	53598986	65896788	13609
315	4.2	28.1	58404898	107917037	52165480	14953
324	2.1	26.1	25679939	5727401	14202083	16421

328	6.1	34.5	332844	2117990	3843716	6555
330	22.4	28.9	104832907	221641518	262341127	14353
398	3.6	30.9	144280774	141962359	117516907	12326
558	3.6	31.0	8799266	8894672	17611851	13793
564	2.6	23.5	18010976	14286715	20526631	13691
609	5.0	30.5	257760713	231082768	165450520	12947
712	1.8	26.8	221877814	67634947	78745589	12408
805	0.9	25.5	97592051	41068338	39234649	13172
905	2.4	27.5	76751828	45863865	34538332	12444
1011	1.1	25.6	119105421	74680759	50405925	11843
1132	2.9	27.4	205054723	51415553	56543203	12921
1197	0.7	25.6	16363192	8823719	20444031	15520
1214	4.9	32.6	41456097	51276797	75664186	13429
1239	3.3	29.8	86428959	95275672	88082966	13553
1254	1.3	30.4	234455768	107109349	109102594	10855
1338	1.0	26.8	107563060	82878056	71384103	13751
1426	0.8	26.9	59869456	23003585	33751407	11552
1509	1.2	26.1	110907604	81032913	76575216	12957
1625	1.0	24.6	10638145	8202782	14686854	15343
1682	1.9	25.7	40157999	24019868	26486612	14965
1776	8.1	28.6	15735787	19255893	37433983	14579
1794	1.0	25.8	18592406	14564458	25353874	19246
1805	8.7	27.3	116608094	233413004	124813580	14453
1827	23.6	31.7	17122725	10589286	24469997	12429
1861	9.9	30.4	162720173	313461904	230718065	11879
1924	2.7	28.2	205867299	88795885	114578173	12641
2025	0.5	24.8	11349799	13099348	10527300	16495
2079	1.1	27.7	295890890	135575279	138816008	12049
2168	2.3	25.3	60681358	48074682	43095353	11931
2246	3.3	29.8	66880653	51910777	50370919	13494
2283	2.3	27.0	234840418	142859202	166842778	13323
2351	6.0	27.3	12061517	9182788	12286485	11646
2357	1.7	27.6	93977455	40498047	54298410	12273
2404	0.8	22.2	13051128	11400476	12266218	15390
2471	1.6	25.9	140447760	80116528	77547291	12563
2567	20.7	28.2	593541502	424238194	311334781	13061
2822	3.7	24.9	42431657	25417368	36574240	13730
2852	0.6	26.0	10751461	5121694	9310119	15005
2867	4.5	30.1	92417797	60513396	105663299	13687
3002	3.2	28.7	112053283	76790966	92968519	14380
3042	0.7	28.1	25080573	11036467	20538829	11340
3098	1.3	25.9	163563195	59996244	72283321	12904
3171	2.8	25.5	8834810	6352890	8957553	17114

	AFN120207	BPS030214	LND110210	POP060210
0	613795732	1046363	3531905.43	87.4
1	6426342	13369	50645.33	94.4
69	1851293	1518	570640.95	1.2

99	13268514	26997	113594.08	56.3
115	3559795	7666	52035.48	56.0
191	80852787	83645	155779.22	239.1
250	11440395	28686	103641.89	48.5
315	9138437	5329	4842.36	738.1
324	1910770	5194	1948.54	460.8
328	4278171	4189	61.05	9856.5
330	41922059	84075	53624.76	350.6
398	16976235	39423	57513.49	168.4
558	8042210	3066	6422.63	211.8
564	2415951	8797	82643.12	19.0
609	25469026	20579	55518.93	231.1
712	11669759	17816	35826.11	181.0
805	4737719	10256	55857.13	54.5
905	4192347	7459	81758.72	34.9
1011	6300866	9536	39486.34	109.9
1132	9729869	15255	43203.90	104.9
1197	2515827	3242	30842.92	43.1
1214	10758428	16331	9707.24	594.8
1239	14917210	14486	7800.06	839.4
1254	14536648	15933	56538.90	174.8
1338	10423660	16990	79626.74	66.6
1426	7045097	6871	46923.27	63.2
1509	11070634	16003	68741.52	87.1
1625	2079426	3884	145545.80	6.8
1682	2685580	7605	76824.17	23.8
1776	28815533	13016	109781.18	24.6
1794	2630968	3403	8952.65	147.0
1805	19993613	28155	7354.22	1195.5
1827	3734300	4799	121298.15	17.0
1861	39813499	36286	47126.40	411.2
1924	16126939	49911	48617.91	196.1
2025	1214201	12178	69000.80	9.7
2079	17779905	19872	40860.69	282.3
2168	5106585	14179	68594.92	54.7
2246	7555764	16645	95988.01	39.9
2283	19625449	25059	44742.70	283.9
2351	2148674	952	1033.81	1018.1
2357	8383463	27537	30060.70	153.9
2404	1622751	4722	75811.00	10.7
2471	10626759	27632	41234.90	153.9
2567	42054592	166982	261231.71	96.3
2822	3980570	17510	82169.62	33.6
2852	1367630	1546	9216.66	67.9
2867	15340483	28682	39490.09	202.6
3002	12389422	33898	66455.52	101.2
3042	2553258	2677	24038.21	77.1
3098	9247311	14622	54157.80	105.0

```
3171    1469008       1901    97093.14        5.8

[52 rows x 54 columns]
```

```
In [21]: print facts.shape
         facts = facts[facts.state_abbreviation.notnull()]
         print facts.shape
         facts.head()
```

```
(3195, 54)
(3143, 54)
```

Out[21]:

	fips	area_name	state_abbreviation	PST045214	PST040210	PST12021
2	1001	Autauga County	AL	55395	54571	1.
3	1003	Baldwin County	AL	200111	182265	9.
4	1005	Barbour County	AL	26887	27457	-2.
5	1007	Bibb County	AL	22506	22919	-1.
6	1009	Blount County	AL	57719	57322	0.

	POP010210	AGE135214	AGE295214	AGE775214	...	SBO415207	\
2	54571	6.0	25.2	13.8	...	0.7	
3	182265	5.6	22.2	18.7	...	1.3	
4	27457	5.7	21.2	16.5	...	0.0	
5	22915	5.3	21.0	14.8	...	0.0	
6	57322	6.1	23.6	17.0	...	0.0	

	SBO015207	MAN450207	WTN220207	RTN130207	RTN131207	AFN120207	\
2	31.7	0	0	598175	12003	88157	
3	27.3	1410273	0	2966489	17166	436955	
4	27.0	0	0	188337	6334	0	
5	0.0	0	0	124707	5804	10757	
6	23.2	341544	0	319700	5622	20941	

	BPS030214	LND110210	POP060210
2	131	594.44	91.8
3	1384	1589.78	114.6
4	8	884.88	31.0
5	19	622.58	36.8
6	3	644.78	88.9

[5 rows x 54 columns]

```
In [23]: ## let's look at the population distributions
         pop = facts['PST045214']
         type(pop)
```

```
Out[23]: pandas.core.series.Series
```

```
In [24]: # convert to an array, sort and plot
         popv = pop.as_matrix()
         type(popv)

Out[24]: numpy.ndarray

In [25]: plt.plot(sorted(popv),'.')

<IPython.core.display.Javascript object>


<IPython.core.display.HTML object>


Out[25]: [<matplotlib.lines.Line2D at 0x113a53c50>]

In [26]: plt.figure()
         plt.semilogy(sorted(popv),'.')

<IPython.core.display.Javascript object>


<IPython.core.display.HTML object>


Out[26]: [<matplotlib.lines.Line2D at 0x113a014d0>]

In [27]: plt.ylabel('County Size')
         plt.xlabel('Rank')

Out[27]: <matplotlib.text.Text at 0x113f6a950>
```

Is this a log-normal distribution?

2 Groupby

Groupby is a very useful operation on tables. It collects together parts of the DataFrame by the values in a column. To then produce a table, we must provide some way of aggregating the groups. `size()` is one such aggregator

```
In [28]: # make a DataFrame containing only two columns
         pop_facts = facts[['state_abbreviation', 'PST045214']]
         pop_facts.head()

Out[28]:   state_abbreviation  PST045214
         2                 AL      55395
         3                 AL     200111
         4                 AL      26887
         5                 AL      22506
         6                 AL      57719
```

In [29]:
```
# Here, we compute the population of each state by summing the population
# This way of doing it produces a DataFrame
state_pop = pop_facts.groupby('state_abbreviation').sum()
state_pop
```

Out[29]:

state_abbreviation	PST045214
AK	736732
AL	4849377
AR	2966369
AZ	6731484
CA	38802500
CO	5355866
CT	3596677
DC	658893
DE	935614
FL	19893297
GA	10097343
HI	1419561
IA	3107126
ID	1634464
IL	12880580
IN	6596855
KS	2904021
KY	4413457
LA	4649676
MA	6745408
MD	5976407
ME	1330089
MI	9909877
MN	5457173
MO	6063589
MS	2994079
MT	1023579
NC	9943964
ND	739482
NE	1881503
NH	1326813
NJ	8938175
NM	2085572
NV	2839099
NY	19746227
OH	11594163
OK	3878051
OR	3970239
PA	12787209
RI	1055173
SC	4832482

```
SD                      853175
TN                     6549352
TX                    26956958
UT                     2942902
VA                     8326289
VT                      626562
WA                     7061530
WI                     5757564
WV                     1850326
WY                      584153
```

```
In [30]: type(state_pop)
```

```
Out[30]: pandas.core.frame.DataFrame
```

```
In [31]: # how many counties per state
         # this way of doing it produces a Series.
         num_counties = facts.groupby('state_abbreviation').size()
         num_counties
```

```
Out[31]: state_abbreviation
         AK     29
         AL     67
         AR     75
         AZ     15
         CA     58
         CO     64
         CT      8
         DC      1
         DE      3
         FL     67
         GA    159
         HI      5
         IA     99
         ID     44
         IL    102
         IN     92
         KS    105
         KY    120
         LA     64
         MA     14
         MD     24
         ME     16
         MI     83
         MN     87
         MO    115
         MS     82
         MT     56
         NC    100
```

```
ND     53
NE     93
NH     10
NJ     21
NM     33
NV     17
NY     62
OH     88
OK     77
OR     36
PA     67
RI      5
SC     46
SD     66
TN     95
TX    254
UT     29
VA    134
VT     14
WA     39
WI     72
WV     55
WY     23
dtype: int64
```

```
In [32]: # A series is a column of a DataFrame
         type(num_counties)
```

```
Out[32]: pandas.core.series.Series
```

```
In [33]: # let's rename the population column,
         # and add the number of counties to the DataFrame
         state_pop = state_pop.rename(columns={'PST045214':'pop'})
         state_pop['counties'] = num_counties
         state_pop
```

Out[33]:

state_abbreviation	pop	counties
AK	736732	29
AL	4849377	67
AR	2966369	75
AZ	6731484	15
CA	38802500	58
CO	5355866	64
CT	3596677	8
DC	658893	1
DE	935614	3
FL	19893297	67
GA	10097343	159

```
HI                    1419561         5
IA                    3107126        99
ID                    1634464        44
IL                   12880580       102
IN                    6596855        92
KS                    2904021       105
KY                    4413457       120
LA                    4649676        64
MA                    6745408        14
MD                    5976407        24
ME                    1330089        16
MI                    9909877        83
MN                    5457173        87
MO                    6063589       115
MS                    2994079        82
MT                    1023579        56
NC                    9943964       100
ND                     739482        53
NE                    1881503        93
NH                    1326813        10
NJ                    8938175        21
NM                    2085572        33
NV                    2839099        17
NY                   19746227        62
OH                   11594163        88
OK                    3878051        77
OR                    3970239        36
PA                   12787209        67
RI                    1055173         5
SC                    4832482        46
SD                     853175        66
TN                    6549352        95
TX                   26956958       254
UT                    2942902        29
VA                    8326289       134
VT                     626562        14
WA                    7061530        39
WI                    5757564        72
WV                    1850326        55
WY                     584153        23
```

In [34]:
```
# here is a scatter plot of these.  This is Pandas using a Matplotlib plot
state_pop.plot.scatter(x='counties',y='pop')
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x116903e50>

In [35]: # here are the 10 largest counties
         facts.nlargest(10,'PST045214')
```

Out[35]:

	fips	area_name	state_abbreviation	PST045214	PST040210
210	6037	Los Angeles County	CA	10116705	9818664
625	17031	Cook County	IL	5246456	5195060
2668	48201	Harris County	TX	4441370	4093011
107	4013	Maricopa County	AZ	4087191	3817357
228	6073	San Diego County	CA	3263431	3095308
221	6059	Orange County	CA	3145515	3010269
373	12086	Miami-Dade County	FL	2662874	2498017
1885	36047	Kings County	NY	2621793	2504709
2624	48113	Dallas County	TX	2518638	2367636
224	6065	Riverside County	CA	2329271	2189757

	PST120214	POP010210	AGE135214	AGE295214	AGE775214	...	\
210	3.0	9818605	6.4	22.8	12.2	...	
625	1.0	5194675	6.5	22.6	12.9	...	
2668	8.5	4092459	7.7	27.1	9.2	...	
107	7.1	3817117	6.7	25.1	13.8	...	
228	5.4	3095313	6.6	22.3	12.7	...	
221	4.5	3010232	6.1	23.0	13.1	...	
373	6.6	2496435	5.8	20.6	15.2	...	
1885	4.7	2504700	7.6	23.3	12.1	...	
2624	6.4	2368139	7.7	26.9	9.7	...	
224	6.4	2189641	6.8	26.3	13.2	...	

	SBO415207	SBO015207	MAN450207	WTN220207	RTN130207	RTN131207	\
210	21.6	30.2	153343705	198435837	119111840	12236	
625	7.2	32.3	77932858	83964561	60585557	11571	
2668	23.0	29.2	169275136	205478751	51899053	13276	
107	9.2	27.3	40182099	49760315	58688328	15153	
228	15.2	29.9	27541073	33704921	38710620	13009	
221	12.4	28.4	49131942	97963621	45022513	15221	
373	60.5	28.9	9347116	60760055	34530470	14074	
1885	10.8	33.6	4555642	13802231	15431858	6077	
2624	15.3	28.1	39047030	58165146	33177208	13929	
224	24.0	30.4	13623526	16912263	24146447	11745	

	AFN120207	BPS030214	LND110210	POP060210
210	20238148	17659	4057.88	2419.6
625	13094372	7753	945.33	5495.1
2668	7874724	40060	1703.48	2402.4
107	8408940	18597	9200.14	414.9
228	9551513	6875	4206.63	735.8
221	8247828	9291	790.57	3807.7

```
373     6005856       7731    1897.72     1315.5
1885    1544409       7551      70.82    35369.1
2624    5705137      14360     871.28     2718.0
224     4835331       6761    7206.48      303.8

[10 rows x 54 columns]
```

2.1 make DataFrames for Clinton and Trump

```
In [36]: trump = results[results.candidate == 'Donald Trump']
         trump.shape

Out[36]: (3586, 8)

In [37]: trump.head()

Out[37]:        state state_abbreviation   county    fips       party     candidate
         135  Alabama                AL  Autauga  1001.0  Republican  Donald Trump
         140  Alabama                AL  Baldwin  1003.0  Republican  Donald Trump
         145  Alabama                AL  Barbour  1005.0  Republican  Donald Trump
         150  Alabama                AL     Bibb  1007.0  Republican  Donald Trump
         155  Alabama                AL   Blount  1009.0  Republican  Donald Trump

              votes  fraction_votes
         135   5387           0.445
         140  23618           0.469
         145   1710           0.501
         150   1959           0.494
         155   7390           0.487

In [38]: # this deletes a column from a DataFrame
         del trump['state']

In [39]: trump.head()

Out[39]:     state_abbreviation   county    fips       party     candidate  votes
         135                 AL  Autauga  1001.0  Republican  Donald Trump   5387
         140                 AL  Baldwin  1003.0  Republican  Donald Trump  23618
         145                 AL  Barbour  1005.0  Republican  Donald Trump   1710
         150                 AL     Bibb  1007.0  Republican  Donald Trump   1959
         155                 AL   Blount  1009.0  Republican  Donald Trump   7390

              fraction_votes
         135           0.445
         140           0.469
         145           0.501
         150           0.494
         155           0.487
```

```
In [40]: del trump['candidate']
         del trump['party']
         trump.head()
```

Out[40]:

	state_abbreviation	county	fips	votes	fraction_votes
135	AL	Autauga	1001.0	5387	0.445
140	AL	Baldwin	1003.0	23618	0.469
145	AL	Barbour	1005.0	1710	0.501
150	AL	Bibb	1007.0	1959	0.494
155	AL	Blount	1009.0	7390	0.487

```
In [41]: clinton = results[results.candidate == 'Hillary Clinton']
         clinton.shape
```

Out[41]: (4205, 8)

```
In [42]: del clinton['candidate']
         del clinton['party']
         del clinton['state']
         clinton.head()
```

Out[42]:

	state_abbreviation	county	fips	votes	fraction_votes
1	AL	Autauga	1001.0	2387	0.800
3	AL	Baldwin	1003.0	5290	0.647
5	AL	Barbour	1005.0	2567	0.906
7	AL	Bibb	1007.0	942	0.755
9	AL	Blount	1009.0	564	0.551

```
In [43]: # give unique names to the columns so that we can merge them together
         clinton = clinton.rename(columns={'fraction_votes':'c_frac', 'votes':'c_vo
         clinton.head()
```

Out[43]:

	state_abbreviation	county	fips	c_votes	c_frac
1	AL	Autauga	1001.0	2387	0.800
3	AL	Baldwin	1003.0	5290	0.647
5	AL	Barbour	1005.0	2567	0.906
7	AL	Bibb	1007.0	942	0.755
9	AL	Blount	1009.0	564	0.551

```
In [44]: trump = trump.rename(columns={'fraction_votes':'t_frac', 'votes':'t_votes'
         trump.head()
```

Out[44]:

	state_abbreviation	county	fips	t_votes	t_frac
135	AL	Autauga	1001.0	5387	0.445
140	AL	Baldwin	1003.0	23618	0.469
145	AL	Barbour	1005.0	1710	0.501
150	AL	Bibb	1007.0	1959	0.494
155	AL	Blount	1009.0	7390	0.487

```
In [45]: # We want to use the fips number to combine all these tables.
         # but, we will see that it is a float64 in trump, and an int64 in facts
         # we will need to fix this
         trump.dtypes

Out[45]: state_abbreviation     object
         county                 object
         fips                  float64
         t_votes                 int64
         t_frac                float64
         dtype: object

In [46]: facts.dtypes

Out[46]: fips                    int64
         area_name              object
         state_abbreviation     object
         PST045214               int64
         PST040210               int64
         PST120214             float64
         POP010210               int64
         AGE135214             float64
         AGE295214             float64
         AGE775214             float64
         SEX255214             float64
         RHI125214             float64
         RHI225214             float64
         RHI325214             float64
         RHI425214             float64
         RHI525214             float64
         RHI625214             float64
         RHI725214             float64
         RHI825214             float64
         POP715213             float64
         POP645213             float64
         POP815213             float64
         EDU635213             float64
         EDU685213             float64
         VET605213               int64
         LFE305213             float64
         HSG010214               int64
         HSG445213             float64
         HSG096213             float64
         HSG495213               int64
         HSD410213               int64
         HSD310213             float64
         INC910213               int64
         INC110213               int64
```

```
PVY020213             float64
BZA010213               int64
BZA110213               int64
BZA115213             float64
NES010213               int64
SBO001207               int64
SBO315207             float64
SBO115207             float64
SBO215207             float64
SBO515207             float64
SBO415207             float64
SBO015207             float64
MAN450207               int64
WTN220207               int64
RTN130207               int64
RTN131207               int64
AFN120207               int64
BPS030214               int64
LND110210             float64
POP060210             float64
dtype: object
```

```
In [47]: # int converts a float to an int
         int(3.1)
```

```
Out[47]: 3
```

```
In [48]: # let's try applying int to each fips entry in trump
         # this will cause an error!
         trump.fips = trump.fips.apply(int)
```

```
---------------------------------------------------------------------------

ValueError                                Traceback (most recent call last)

<ipython-input-48-d2d3ae290c2a> in <module>()
  1 # let's try applying int to each fips entry in trump
  2 # this will cause an error!
----> 3 trump.fips = trump.fips.apply(int)


/Users/spielman/anaconda2/lib/python2.7/site-packages/pandas/core/series.py
2218         else:
2219             values = self.asobject
-> 2220             mapped = lib.map_infer(values, f, convert=convert_dtype)
2221
2222         if len(mapped) and isinstance(mapped[0], Series):
```

```
        pandas/src/inference.pyx in pandas.lib.map_infer (pandas/lib.c:62658)()

        ValueError: cannot convert float NaN to integer
```

```
In [49]: # because there are null values.  NaN stands for Not a Number!
         sum(trump.fips.isnull())
```

```
Out[49]: 10
```

```
In [50]: # let's look at them
         # this notation selects certain rows of the table
         trump[trump.fips.isnull()]
```

Out[50]:

	state_abbreviation	county	fips	t_votes	t_frac
14610	NH	Belknap	NaN	5505	0.360110
14618	NH	Carroll	NaN	4182	0.345391
14626	NH	Cheshire	NaN	4543	0.344951
14634	NH	Coos	NaN	2183	0.382915
14642	NH	Grafton	NaN	4898	0.304280
14650	NH	Hillsborough	NaN	28981	0.353724
14658	NH	Merrimack	NaN	10966	0.338175
14666	NH	Rockingham	NaN	28716	0.393710
14674	NH	Strafford	NaN	7352	0.343760
14682	NH	Sullivan	NaN	3080	0.380952

```
In [51]: facts[facts.state_abbreviation=='NH']
```

Out[51]:

	fips	area_name	state_abbreviation	PST045214	PST040210
1795	33001	Belknap County	NH	60305	60092
1796	33003	Carroll County	NH	47399	47820
1797	33005	Cheshire County	NH	76115	77117
1798	33007	Coos County	NH	31653	33052
1799	33009	Grafton County	NH	89658	89114
1800	33011	Hillsborough County	NH	405184	400721
1801	33013	Merrimack County	NH	147171	146442
1802	33015	Rockingham County	NH	300621	295220
1803	33017	Strafford County	NH	125604	123146
1804	33019	Sullivan County	NH	43103	43742

	PST120214	POP010210	AGE135214	AGE295214	AGE775214	...	\
1795	0.4	60088	4.7	19.5	19.9	...	
1796	-0.9	47818	3.8	16.8	24.5	...	
1797	-1.3	77117	4.7	18.6	17.3	...	
1798	-4.2	33055	4.2	17.7	21.8	...	
1799	0.6	89118	4.3	17.2	18.0	...	
1800	1.1	400721	5.4	21.7	13.9	...	

```
1801        0.5     146445        4.8       19.9       16.1    ...
1802        1.8     295223        4.6       20.6       15.2    ...
1803        2.0     123143        5.0       19.5       13.8    ...
1804       -1.5      43742        4.7       19.7       18.9    ...

      SBO415207  SBO015207  MAN450207  WTN220207  RTN130207  RTN131207  \
1795        0.0       23.7     687068          0    1348651      22099
1796        0.4       19.6     213380          0     889015      18569
1797        0.6       22.6    1095187          0    1649001      21293
1798        0.0        0.0     295901          0     602906      18656
1799        0.3       24.0    1314279          0    1961272      22919
1800        2.0       26.3    7707601    3873122    7647259      18989
1801        0.9       24.6    1539636    2812352    2605617      17536
1802        0.9       27.8    3912579    5844817    6414862      21603
1803        0.8       26.8    1185999     351111    1672899      13701
1804        0.0       24.8     640776          0     562392      13154

      AFN120207  BPS030214  LND110210  POP060210
1795     151572        198     400.23      150.1
1796     214906        197     931.06       51.4
1797     110289        141     706.66      109.1
1798      85674         56    1794.69       18.4
1799     272818        187    1708.75       52.2
1800     732310        968     876.14      457.4
1801     223534        312     934.12      156.8
1802     660746        875     694.72      425.0
1803     146812        416     368.98      333.7
1804      32307         53     537.31       81.4

[10 rows x 54 columns]
```

In [52]:
```
nhfips = facts[facts.state_abbreviation=='NH'].fips
nhfips
```

Out[52]:
```
1795    33001
1796    33003
1797    33005
1798    33007
1799    33009
1800    33011
1801    33013
1802    33015
1803    33017
1804    33019
Name: fips, dtype: int64
```

In [53]:
```
# I should probably use some string matching on county names to transfer
# the fips from facts to trump.  But, instead I'll observe that they are s
```

```
        # alphabetically, so I can just use a loop to move one to the other

        ind = trump[trump.fips.isnull()].index
        for i in range(10):
            trump.set_value(ind[i],'fips',nhfips.iloc[i])
        trump[trump.state_abbreviation=='NH']
```

Out[53]:

	state_abbreviation	county	fips	t_votes	t_frac
14610	NH	Belknap	33001.0	5505	0.360110
14618	NH	Carroll	33003.0	4182	0.345391
14626	NH	Cheshire	33005.0	4543	0.344951
14634	NH	Coos	33007.0	2183	0.382915
14642	NH	Grafton	33009.0	4898	0.304280
14650	NH	Hillsborough	33011.0	28981	0.353724
14658	NH	Merrimack	33013.0	10966	0.338175
14666	NH	Rockingham	33015.0	28716	0.393710
14674	NH	Strafford	33017.0	7352	0.343760
14682	NH	Sullivan	33019.0	3080	0.380952

```
In [54]: clinton[clinton.fips.isnull()]
```

Out[54]:

	state_abbreviation	county	fips	c_votes	c_frac
14588	NH	Belknap	NaN	3490	0.368143
14590	NH	Carroll	NaN	3230	0.363534
14592	NH	Cheshire	NaN	5166	0.292907
14594	NH	Coos	NaN	2013	0.356283
14596	NH	Grafton	NaN	6918	0.326691
14598	NH	Hillsborough	NaN	28099	0.420990
14600	NH	Merrimack	NaN	12209	0.403137
14602	NH	Rockingham	NaN	22829	0.423473
14604	NH	Strafford	NaN	8801	0.356807
14606	NH	Sullivan	NaN	2497	0.297156

```
In [55]: ind = clinton[clinton.fips.isnull()].index
         for i in range(10):
             clinton.set_value(ind[i],'fips',nhfips.iloc[i])
         clinton[clinton.state_abbreviation=='NH']
```

Out[55]:

	state_abbreviation	county	fips	c_votes	c_frac
14588	NH	Belknap	33001.0	3490	0.368143
14590	NH	Carroll	33003.0	3230	0.363534
14592	NH	Cheshire	33005.0	5166	0.292907
14594	NH	Coos	33007.0	2013	0.356283
14596	NH	Grafton	33009.0	6918	0.326691
14598	NH	Hillsborough	33011.0	28099	0.420990
14600	NH	Merrimack	33013.0	12209	0.403137
14602	NH	Rockingham	33015.0	22829	0.423473
14604	NH	Strafford	33017.0	8801	0.356807
14606	NH	Sullivan	33019.0	2497	0.297156

```
In [56]: trump.fips = trump.fips.apply(int)
         trump.head()
```

Out[56]:

	state_abbreviation	county	fips	t_votes	t_frac
135	AL	Autauga	1001	5387	0.445
140	AL	Baldwin	1003	23618	0.469
145	AL	Barbour	1005	1710	0.501
150	AL	Bibb	1007	1959	0.494
155	AL	Blount	1009	7390	0.487

```
In [57]: clinton.fips = clinton.fips.apply(int)
         clinton.head()
```

Out[57]:

	state_abbreviation	county	fips	c_votes	c_frac
1	AL	Autauga	1001	2387	0.800
3	AL	Baldwin	1003	5290	0.647
5	AL	Barbour	1005	2567	0.906
7	AL	Bibb	1007	942	0.755
9	AL	Blount	1009	564	0.551

3 Merge Tables. This is a SQL Join.

```
In [58]: election = pd.merge(facts, trump, on="fips", how="inner")
```

```
In [59]: election.head()
```

Out[59]:

	fips	area_name	state_abbreviation_x	PST045214	PST040210	PST120
0	1001	Autauga County	AL	55395	54571	
1	1003	Baldwin County	AL	200111	182265	
2	1005	Barbour County	AL	26887	27457	-
3	1007	Bibb County	AL	22506	22919	-
4	1009	Blount County	AL	57719	57322	

	POP010210	AGE135214	AGE295214	AGE775214	...	RTN130207	RTN13120
0	54571	6.0	25.2	13.8	...	598175	1200
1	182265	5.6	22.2	18.7	...	2966489	1716
2	27457	5.7	21.2	16.5	...	188337	633
3	22915	5.3	21.0	14.8	...	124707	580
4	57322	6.1	23.6	17.0	...	319700	562

	AFN120207	BPS030214	LND110210	POP060210	state_abbreviation_y	cour
0	88157	131	594.44	91.8	AL	Autau
1	436955	1384	1589.78	114.6	AL	Baldw
2	0	8	884.88	31.0	AL	Barbo
3	10757	19	622.58	36.8	AL	Bi
4	20941	3	644.78	88.9	AL	Blou

	t_votes	t_frac

```
0     5387   0.445
1    23618   0.469
2     1710   0.501
3     1959   0.494
4     7390   0.487

[5 rows x 58 columns]
```

In [60]:
```
# Let's try that again, but only grabbing the columns from trump that we
election = pd.merge(facts, trump[['fips','t_votes','t_frac']], on="fips",
election.head()
```

Out[60]:
```
   fips       area_name state_abbreviation  PST045214  PST040210  PST12021
0  1001  Autauga County                AL      55395      54571        1.
1  1003  Baldwin County                AL     200111     182265        9.
2  1005  Barbour County                AL      26887      27457       -2.
3  1007     Bibb County                AL      22506      22919       -1.
4  1009   Blount County                AL      57719      57322        0.

   POP010210  AGE135214  AGE295214  AGE775214   ...    MAN450207  WTN22020
0      54571        6.0       25.2       13.8   ...            0
1     182265        5.6       22.2       18.7   ...      1410273
2      27457        5.7       21.2       16.5   ...            0
3      22915        5.3       21.0       14.8   ...            0
4      57322        6.1       23.6       17.0   ...       341544

   RTN130207  RTN131207  AFN120207  BPS030214  LND110210  POP060210  t_vot
0     598175      12003      88157        131     594.44       91.8     53
1    2966489      17166     436955       1384    1589.78      114.6    236
2     188337       6334          0          8     884.88       31.0     17
3     124707       5804      10757         19     622.58       36.8     19
4     319700       5622      20941          3     644.78       88.9     73

   t_frac
0   0.445
1   0.469
2   0.501
3   0.494
4   0.487

[5 rows x 56 columns]
```

In [61]:
```
election = pd.merge(election, clinton[['fips','c_votes','c_frac']], on="fi
```

In [62]:
```
election.head()
```

Out[62]:
```
   fips       area_name state_abbreviation  PST045214  PST040210  PST12021
0  1001  Autauga County                AL      55395      54571        1.
1  1003  Baldwin County                AL     200111     182265        9.
```

```
   2  1005  Barbour County                AL      26887      27457       -2.
   3  1007     Bibb County                AL      22506      22919       -1.
   4  1009   Blount County                AL      57719      57322        0.

      POP010210  AGE135214  AGE295214  AGE775214   ...    RTN130207  RTN13120
   0      54571        6.0       25.2       13.8   ...       598175      1200
   1     182265        5.6       22.2       18.7   ...      2966489      1716
   2      27457        5.7       21.2       16.5   ...       188337       633
   3      22915        5.3       21.0       14.8   ...       124707       580
   4      57322        6.1       23.6       17.0   ...       319700       562

      AFN120207  BPS030214  LND110210  POP060210  t_votes  t_frac  c_votes  \
   0      88157        131     594.44       91.8     5387   0.445     2387
   1     436955       1384    1589.78      114.6    23618   0.469     5290
   2          0          8     884.88       31.0     1710   0.501     2567
   3      10757         19     622.58       36.8     1959   0.494      942
   4      20941          3     644.78       88.9     7390   0.487      564

      c_frac
   0   0.800
   1   0.647
   2   0.906
   3   0.755
   4   0.551

   [5 rows x 58 columns]
```

In [63]:
```
print clinton.shape
print trump.shape
print facts.shape
print election.shape
```

```
(4205, 5)
(3586, 5)
(3143, 54)
(2721, 58)
```

We lost a lot of rows. We will figure out why later.

In [64]:
```
# let's plot some tends we have been told to expect
election.plot.scatter(x='RHI825214',y='c_frac')
plt.xlabel(fact_dict.loc['RHI825214'].description)
plt.ylabel('fraction of vote for Clinton')
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
Out[64]: <matplotlib.text.Text at 0x116f81650>

In [65]: ## let's compute the best fit line, using least squares
         A = election['RHI825214'].as_matrix()
         A.shape

Out[65]: (2721,)

In [66]: A = np.reshape(A,(A.size,1))
         A.shape

Out[66]: (2721, 1)

In [67]: # append an all-1s column to A
         A = np.hstack((A, np.ones((2721,1))))
         b = election['c_frac'].as_matrix()
         b

Out[67]: array([ 0.8  ,  0.647,  0.906, ...,  0.438,  0.385,  0.395])

In [68]: Atrans = A.transpose()
         x = np.linalg.inv(Atrans.dot(A)).dot(Atrans.dot(b))
         x

Out[68]: array([-0.00411099,  0.85302569])

In [69]: xrange = np.array([min(A[:,0]), max(A[:,0])])
         xrange

Out[69]: array([  3.1,  98.6])

In [70]: # We now add a the best-fit line to that plot.
         # This is why we keep the plots around.
         h = plt.plot(xrange, xrange*x[0] + x[1],color='red',linewidth=2)
```

3.0.1 Search to see if there are any other identifiable trends.

By doing the same thing for every column, and checking the slopes of the lines we get.

```
In [71]: cols = election.columns[3:-4]
         cols

Out[71]: Index([u'PST045214', u'PST040210', u'PST120214', u'POP010210', u'AGE135214
                u'AGE295214', u'AGE775214', u'SEX255214', u'RHI125214', u'RHI225214
                u'RHI325214', u'RHI425214', u'RHI525214', u'RHI625214', u'RHI725214
                u'RHI825214', u'POP715213', u'POP645213', u'POP815213', u'EDU635213
                u'EDU685213', u'VET605213', u'LFE305213', u'HSG010214', u'HSG445213
                u'HSG096213', u'HSG495213', u'HSD410213', u'HSD310213', u'INC910213
                u'INC110213', u'PVY020213', u'BZA010213', u'BZA110213', u'BZA115213
                u'NES010213', u'SBO001207', u'SBO315207', u'SBO115207', u'SBO215207
                u'SBO515207', u'SBO415207', u'SBO015207', u'MAN450207', u'WTN220207
                u'RTN130207', u'RTN131207', u'AFN120207', u'BPS030214', u'LND110210
                u'POP060210'],
               dtype='object')
```

```
In [72]: nc = len(cols)
         nc

Out[72]: 51

In [73]: slopes = np.zeros(nc)
         lines = np.zeros((nc,2))
         xranges = np.zeros((nc,2))
         for i in range(nc):
             a = election[cols[i]].as_matrix()
             A = np.reshape(a,(a.size,1))
             A = np.hstack((A, np.ones((2721,1))))
             b = election['c_frac'].as_matrix()
             Atrans = A.transpose()
             x = np.linalg.inv(Atrans.dot(A)).dot(Atrans.dot(b))
             lines[i,:] = x
             xranges[i,:] = [max(a), min(a)]
             slopes[i] = abs(x[0])*(max(a)-min(a))
         slopes

Out[73]: array([ 0.21897135,  0.22045563,  0.08585199,  0.22041821,  0.1293298 ,
                 0.07261254,  0.15784183,  0.22731679,  0.51804161,  0.62312866,
                 0.29531154,  0.04508903,  0.413932  ,  0.76914228,  0.09768342,
                 0.3925999 ,  0.13161584,  0.08048621,  0.09936856,  0.48934286,
                 0.18924382,  0.12879959,  0.18516068,  0.20748336,  0.17550077,
                 0.03693421,  0.22084628,  0.20107394,  0.26448777,  0.22847823,
                 0.22654217,  0.38745749,  0.17895536,  0.19374683,  0.07415944,
                 0.27140234,  0.23020625,  0.61273306,  0.24546104,  0.10803073,
                 0.30950038,  0.19865137,  0.07118098,  0.14552515,  0.1962401 ,
                 0.18263682,  0.16977012,  0.15761724,  0.19275358,  0.36124485,
                 0.31239258])

In [74]: ind = np.argmax(slopes)
         ind

Out[74]: 13

In [75]: col = cols[ind]
         col

Out[75]: 'RHI625214'

In [76]: slopes[ind]

Out[76]: 0.76914227854941675

In [77]: election.plot.scatter(x=col,y='c_frac')
         plt.xlabel(fact_dict.loc[col].description)
         plt.ylabel('fraction of vote for Clinton')
```

```
        x = lines[ind,:]
        xrange = xranges[ind,:]
        h = plt.plot(xrange, xrange*x[0] + x[1],color='red',linewidth=2)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
In [ ]:
```

3.1 This time, for Trump

```
In [78]: slopes = np.zeros(nc)
         lines = np.zeros((nc,2))
         xranges = np.zeros((nc,2))
         for i in range(nc):
             a = election[cols[i]].as_matrix()
             A = np.reshape(a,(a.size,1))
             A = np.hstack((A, np.ones((2721,1))))
             b = election['t_frac'].as_matrix()
             Atrans = A.transpose()
             x = np.linalg.inv(Atrans.dot(A)).dot(Atrans.dot(b))
             lines[i,:] = x
             xranges[i,:] = [max(a), min(a)]
             slopes[i] = abs(x[0])*(max(a)-min(a))
         slopes
```

```
Out[78]: array([ 0.14611112,  0.16300784,  0.27363822,  0.16308321,  0.26871044,
                 0.34431267,  0.39096012,  0.08649924,  0.00247037,  0.05510525,
                 0.25835921,  0.09668882,  0.04681471,  0.09879784,  0.11823176,
                 0.06091184,  0.28962715,  0.05333837,  0.0905106 ,  0.03148675,
                 0.13202451,  0.08385681,  0.05299727,  0.11662331,  0.02579914,
                 0.07642623,  0.19815917,  0.1284903 ,  0.23288672,  0.04748565,
                 0.12208201,  0.08012713,  0.12922958,  0.0070506 ,  0.40047241,
                 0.09438274,  0.13492366,  0.04685062,  0.02851623,  0.14338602,
                 0.01192722,  0.01431125,  0.0114247 ,  0.08657923,  0.04325305,
                 0.10157566,  0.12537203,  0.09986729,  0.23201665,  0.39981289,
                 0.04976635])
```

```
In [79]: ind = np.argmax(slopes)
         ind
```

```
Out[79]: 34
```

```
In [80]: col = cols[ind]
         col
```

```
Out[80]: 'BZA115213'

In [81]: election.plot.scatter(x=col,y='t_frac')
         plt.xlabel(fact_dict.loc[col].description)
         plt.ylabel('fraction of vote for Trump')
         x = lines[ind,:]
         xrange = xranges[ind,:]
         h = plt.plot(xrange, xrange*x[0] + x[1],color='red',linewidth=2)

<IPython.core.display.Javascript object>


<IPython.core.display.HTML object>


In [ ]:
```

3.2 Regression using all the data

```
In [82]: # We divide our data into training data and test data.
         # I only believe regression if it gives good results on test data (on whic
         ncounties = election.shape[0]
         select = np.random.rand(ncounties) < 0.9
         print sum(select)
         train = election[select]
         test = election[~select]
         print train.shape
         print test.shape

2467
(2467, 58)
(254, 58)


In [83]: # just take the columns that represent percents
         pcols = cols[4:21]
         pcols

Out[83]: Index([u'AGE135214', u'AGE295214', u'AGE775214', u'SEX255214', u'RHI125214
                u'RHI225214', u'RHI325214', u'RHI425214', u'RHI525214', u'RHI625214
                u'RHI725214', u'RHI825214', u'POP715213', u'POP645213', u'POP815213
                u'EDU635213', u'EDU685213'],
               dtype='object')

In [84]: A = train[pcols].as_matrix()
         A = np.hstack((A, np.ones((train.shape[0],1))))
         print type(A)
         print A.shape
```

```
<type 'numpy.ndarray'>
(2467, 18)
```

```
In [85]: b = train['c_frac'].as_matrix()
         print type(b)
         print b.shape
```

```
<type 'numpy.ndarray'>
(2467,)
```

```
In [86]: # solve the linear regression problem
         Atrans = A.transpose()
         x = np.linalg.inv(Atrans.dot(A)).dot(Atrans.dot(b))

         # and, report the average square error
         np.mean((A.dot(x)-b)**2)
```

```
Out[86]: 0.010776691943377418
```

```
In [87]: # compre this to the average error if we just guessed the average
         np.mean((np.mean(b)-b)**2)
```

```
Out[87]: 0.02541508053756095
```

```
In [88]: # Now, try it on the test data
         Atest = test[pcols].as_matrix()
         Atest = np.hstack((Atest, np.ones((test.shape[0],1))))
         btest = test['c_frac'].as_matrix()

         print "using x", np.mean((Atest.dot(x) - btest)**2)
         print "using mean", np.mean((np.mean(btest) - btest)**2)
```

```
using x 0.0114414544196
using mean 0.0258314206708
```

```
In [89]: x_meaning = pd.Series(x[:-1], index=pcols)
         x_meaning
```

```
Out[89]: AGE135214   -0.015200
         AGE295214    0.006229
         AGE775214    0.004397
         SEX255214    0.004647
         RHI125214   -0.009741
         RHI225214   -0.010462
         RHI325214   -0.017167
         RHI425214   -0.008753
```

```
RHI525214   -0.008878
RHI625214   -0.040117
RHI725214   -0.005027
RHI825214   -0.008057
POP715213    0.000089
POP645213   -0.002552
POP815213   -0.000341
EDU635213   -0.002562
EDU685213   -0.000096
dtype: float64
```

```
In [90]: x_meaning.to_frame().join(fact_dict)
```

Out[90]:

	0	description
AGE135214	-0.015200	Persons under 5 years, percent, 2014
AGE295214	0.006229	Persons under 18 years, percent, 2014
AGE775214	0.004397	Persons 65 years and over, percent, 2014
SEX255214	0.004647	Female persons, percent, 2014
RHI125214	-0.009741	White alone, percent, 2014
RHI225214	-0.010462	Black or African American alone, percent, 2014
RHI325214	-0.017167	American Indian and Alaska Native alone, perce...
RHI425214	-0.008753	Asian alone, percent, 2014
RHI525214	-0.008878	Native Hawaiian and Other Pacific Islander alo...
RHI625214	-0.040117	Two or More Races, percent, 2014
RHI725214	-0.005027	Hispanic or Latino, percent, 2014
RHI825214	-0.008057	White alone, not Hispanic or Latino, percent, ...
POP715213	0.000089	Living in same house 1 year & over, percent, 2...
POP645213	-0.002552	Foreign born persons, percent, 2009-2013
POP815213	-0.000341	Language other than English spoken at home, pc...
EDU635213	-0.002562	High school graduate or higher, percent of per...
EDU685213	-0.000096	Bachelor's degree or higher, percent of person...

4 Figuring out which counties were missing

```
In [91]: # here is an example of using isin (and in)
         NewEngland = ['CT','MA','NH','ME', 'VT', 'RI']
         'CT' in NewEngland
```

```
Out[91]: True
```

```
In [92]: state_pop = state_pop.reset_index()
         state_pop.head()
```

Out[92]:

	state_abbreviation	pop	counties
0	AK	736732	29
1	AL	4849377	67
2	AR	2966369	75
3	AZ	6731484	15
4	CA	38802500	58

In [93]:
```
state_pop[state_pop.state_abbreviation.isin(NewEngland)]
```

Out[93]:

	state_abbreviation	pop	counties
6	CT	3596677	8
19	MA	6745408	14
21	ME	1330089	16
30	NH	1326813	10
39	RI	1055173	5
46	VT	626562	14

In [94]:
```
# lets see which rows from clinton did not make it into election.
# the ~ (tilde) symbol takes the negation.
clinton[~clinton.fips.isin(election.fips)]
```

Out[94]:

	state_abbreviation	county	fips	c_votes	c_fr
470	AK	State House District 1	90200101	3	0.2
472	AK	State House District 10	90200110	2	0.2
474	AK	State House District 11	90200111	2	0.1
476	AK	State House District 12	90200112	0	0.0
478	AK	State House District 13	90200113	2	0.2
480	AK	State House District 14	90200114	4	0.3
482	AK	State House District 15	90200115	1	0.1
484	AK	State House District 16	90200116	3	0.2
486	AK	State House District 17	90200117	4	0.2
488	AK	State House District 18	90200118	4	0.2
490	AK	State House District 19	90200119	4	0.3
492	AK	State House District 2	90200102	1	0.1
494	AK	State House District 20	90200120	6	0.3
496	AK	State House District 21	90200121	6	0.3
498	AK	State House District 22	90200122	3	0.2
500	AK	State House District 23	90200123	0	0.0
502	AK	State House District 24	90200124	5	0.3
504	AK	State House District 25	90200125	4	0.2
506	AK	State House District 26	90200126	4	0.2
508	AK	State House District 27	90200127	4	0.2
510	AK	State House District 28	90200128	5	0.2
512	AK	State House District 29	90200129	2	0.2
514	AK	State House District 3	90200103	0	0.0
516	AK	State House District 30	90200130	2	0.2
518	AK	State House District 31	90200131	0	0.0
520	AK	State House District 32	90200132	0	0.0
522	AK	State House District 33	90200133	4	0.1
524	AK	State House District 34	90200134	4	0.2
526	AK	State House District 35	90200135	4	0.2
528	AK	State House District 36	90200136	2	0.1
...	...	...	...	...	.
21513	VT	Winhall	95000240	44	0.2
21515	VT	Winooski	95000241	156	0.1

```
21517                VT                Wolcott  95000242       23   0.0
21519                VT               Woodbury  95000243       11   0.0
21521                VT               Woodford  95000244       10   0.1
21523                VT              Woodstock  95000245      199   0.2
21525                VT              Worcester  95000246       24   0.0
24518                WY                 Albany     56001        8   0.2
24520                WY               Big Horn     56003        2   0.5
24522                WY               Campbell     56005        6   0.3
24524                WY                 Carbon     56007        4   0.5
24526                WY              Converse     56009        2   0.4
24528                WY                 Crook     56011        1   0.5
24530                WY               Fremont     56013       10   0.5
24532                WY                Goshen     56015        3   0.6
24534                WY           Hot Springs     56017        1   0.5
24536                WY               Johnson     56019        1   0.5
24538                WY               Laramie     56021       26   0.5
24540                WY               Lincoln     56023        3   0.5
24542                WY               Natrona     56025       20   0.5
24544                WY              Niobrara     56027        0   0.0
24546                WY                  Park     56029        5   0.3
24548                WY                Platte     56031        1   0.3
24550                WY               Sheridan     56033        7   0.5
24552                WY               Sublette     56035        1   0.2
24554                WY              Sweetwater     56037       10   0.4
24556                WY                 Teton     56039        6   0.4
24558                WY                 Uinta     56041        4   0.4
24560                WY               Washakie     56043        2   0.6
24562                WY                 Weston     56045        1   0.5

[1484 rows x 5 columns]
```

In the case of VT, it seems to be due to an error in the fips number that we used to merge the data.

```
In [95]: facts[facts.state_abbreviation=='VT']
```

```
Out[95]:        fips          area_name state_abbreviation  PST045214  PST040210  \
2853  50001     Addison County                VT      37009      36824
2854  50003  Bennington County                VT      36445      37125
2855  50005   Caledonia County                VT      30981      31226
2856  50007  Chittenden County                VT     160531     156540
2857  50009      Essex County                VT       6125       6306
2858  50011    Franklin County                VT      48642      47752
2859  50013  Grand Isle County                VT       6994       6970
2860  50015    Lamoille County                VT      25082      24475
2861  50017     Orange County                VT      28859      28936
2862  50019    Orleans County                VT      27082      27231
2863  50021    Rutland County                VT      60086      61646
```

2864	50023	Washington County	VT	58998	59535
2865	50025	Windham County	VT	43714	44513
2866	50027	Windsor County	VT	56014	56666

	PST120214	POP010210	AGE135214	AGE295214	AGE775214	...	\
2853	0.5	36821	4.3	18.2	16.9	...	
2854	-1.8	37125	4.7	19.5	21.2	...	
2855	-0.8	31227	4.8	20.4	17.8	...	
2856	2.5	156545	4.9	18.7	13.2	...	
2857	-2.9	6306	4.3	17.6	23.3	...	
2858	1.9	47746	6.0	23.1	13.9	...	
2859	0.3	6970	4.6	18.9	17.9	...	
2860	2.5	24475	5.2	21.2	15.2	...	
2861	-0.3	28936	4.8	19.7	17.6	...	
2862	-0.5	27231	4.9	20.1	20.3	...	
2863	-2.5	61642	4.6	18.4	19.4	...	
2864	-0.9	59534	4.9	19.4	17.0	...	
2865	-1.8	44513	4.6	18.9	19.3	...	
2866	-1.2	56670	4.6	18.8	20.7	...	

	SBO415207	SBO015207	MAN450207	WTN220207	RTN130207	RTN131207	\
2853	0.0	22.1	538957	101400	464847	12657	
2854	0.6	24.8	571644	0	833024	22861	
2855	0.0	28.0	339060	66247	412719	13565	
2856	1.0	28.2	4823542	2392465	2740931	18188	
2857	0.0	23.8	62793	0	15031	2325	
2858	0.0	23.6	0	413283	545346	11383	
2859	0.0	0.0	0	9066	39465	5229	
2860	0.0	31.4	0	0	313467	12412	
2861	0.0	24.7	161917	0	237240	8181	
2862	0.0	18.2	0	73373	324145	11858	
2863	0.5	23.9	640754	267644	1084887	17136	
2864	1.0	23.0	1011871	0	946580	16125	
2865	0.9	30.2	375077	867141	660735	15120	
2866	0.3	24.5	406465	327739	691702	12147	

	AFN120207	BPS030214	LND110210	POP060210
2853	48164	101	766.33	48.0
2854	106659	35	674.98	55.0
2855	29922	49	648.86	48.1
2856	360756	533	536.58	291.7
2857	3090	9	663.60	9.5
2858	39606	152	633.71	75.3
2859	6528	21	81.81	85.2
2860	174189	66	458.80	53.3
2861	26208	39	687.03	42.1
2862	36307	200	693.27	39.3
2863	124431	50	929.82	66.3

```
2864     118956        127     687.23       86.6
2865     138405         91     785.31       56.7
2866     154409         73     969.34       58.5

[14 rows x 54 columns]
```

```
In [96]: clinton[clinton.state_abbreviation=='VT']
```

Out[96]:

	state_abbreviation	county	fips	c_votes	c_frac
21035	VT	Addison	95000001	40	0.137
21037	VT	Albany	95000002	16	0.103
21039	VT	Alburgh	95000003	31	0.089
21041	VT	Andover	95000004	14	0.122
21043	VT	Arlington	95000005	84	0.174
21045	VT	Athens	95000006	5	0.081
21047	VT	Bakersfield	95000007	22	0.083
21049	VT	Baltimore	95000008	1	0.023
21051	VT	Barnard	95000009	46	0.177
21053	VT	Barnet	95000010	31	0.091
21055	VT	Barre	95000011	141	0.117
21057	VT	Barre Town	95000012	149	0.121
21059	VT	Barton	95000013	25	0.073
21061	VT	Belvidere	95000014	3	0.061
21063	VT	Bennington	95000015	400	0.158
21065	VT	Benson	95000016	19	0.107
21067	VT	Berkshire	95000017	11	0.057
21069	VT	Berlin	95000018	53	0.103
21071	VT	Bethel	95000019	43	0.103
21073	VT	Bloomfield	95000020	6	0.125
21075	VT	Bolton	95000021	33	0.101
21077	VT	Bradford	95000022	57	0.120
21079	VT	Braintree	95000023	18	0.075
21081	VT	Brandon	95000024	103	0.133
21083	VT	Brattleboro	95000025	422	0.126
21085	VT	Bridgewater	95000026	24	0.127
21087	VT	Bridport	95000027	26	0.120
21089	VT	Brighton	95000028	35	0.157
21091	VT	Bristol	95000029	76	0.085
21093	VT	Brookfield	95000030	36	0.115
...	...	...	...	...	...
21467	VT	Waterbury	95000217	174	0.124
21469	VT	Waterford	95000218	21	0.096
21471	VT	Waterville	95000219	6	0.049
21473	VT	Weathersfield	95000220	78	0.127
21475	VT	Wells	95000221	26	0.139
21477	VT	West Fairlee	95000222	9	0.071
21479	VT	West Haven	95000225	5	0.086
21481	VT	West Rutland	95000229	52	0.154

```
21483                VT   West Windsor  95000230       53   0.165
21485                VT      Westfield  95000223        8   0.071
21487                VT       Westford  95000224       66   0.111
21489                VT    Westminster  95000226      102   0.106
21491                VT       Westmore  95000227        4   0.057
21493                VT         Weston  95000228       26   0.144
21495                VT      Weybridge  95000231       61   0.210
21497                VT       Wheelock  95000232        9   0.063
21499                VT        Whiting  95000233        8   0.092
21501                VT     Whitingham  95000234       24   0.127
21503                VT   Williamstown  95000235       55   0.112
21505                VT      Williston  95000236      392   0.174
21507                VT     Wilmington  95000237       70   0.159
21509                VT        Windham  95000238       15   0.120
21511                VT        Windsor  95000239      108   0.132
21513                VT        Winhall  95000240       44   0.216
21515                VT       Winooski  95000241      156   0.102
21517                VT        Wolcott  95000242       23   0.070
21519                VT       Woodbury  95000243       11   0.056
21521                VT       Woodford  95000244       10   0.116
21523                VT      Woodstock  95000245      199   0.220
21525                VT      Worcester  95000246       24   0.072

[246 rows x 5 columns]
```

Before believing any of our results, we should fix the problem of missing data! This would involve understanding the (possibly many) reasons each county didn't make it in

In [ ]: