Foundations of Data Science, Fall 2015, UC Berkeley
Examples from Lecture 9/4/15
A. Adhikari

In [3]:

```
census_url = 'http://www.census.gov/popest/data/national/asrh/2014/files/NC-EST2014-AGESEX-RES.csv'
```

In [4]:

```
full_table = Table.read_table(census_url)
full_table
```

Out[4]:

SEX	AGE	CENSUS2010POP	ESTIMATESBASE2010	POPESTIMATE2010	POPESTIMA
0	0	3944153	3944160	3951330	3963071
0	1	3978070	3978090	3957888	3966510
0	2	4096929	4096939	4090862	3971573
0	3	4119040	4119051	4111920	4102501
0	4	4063170	4063186	4077552	4122303
0	5	4056858	4056872	4064653	4087713
0	6	4066381	4066412	4073013	4074979
0	7	4030579	4030594	4043047	4083240
0	8	4046486	4046497	4025604	4053206
0	9	4148353	4148369	4125415	4035769

... (296 rows omitted)

In [ ]:

```
full_table.show()
# shows all rows of the table; output omitted from PDF
```

The description of the variables, provided by the Census Bureau, includes the following codes:

SEX: 0 = both males and females; 1 = males; 2 = females

AGE: Rows labeled 999 are the total in their category. So the row whose SEX entry is 1 and AGE entry is 999 contains the totals for males of all ages.

In [6]:

```
census = full_table.select(['SEX', 'AGE', 'CENSUS2010POP', 'POPESTIMATE2014'])
census
```

Out[6]:

SEX	AGE	CENSUS2010POP	POPESTIMATE2014
0	0	3944153	3948350
0	1	3978070	3962123
0	2	4096929	3957772
0	3	4119040	4005190
0	4	4063170	4003448
0	5	4056858	4004858
0	6	4066381	4134352
0	7	4030579	4154000
0	8	4046486	4119524
0	9	4148353	4106832

... (296 rows omitted)

In [7]:

```
len(census)             # number of columns
```

Out[7]:

```
4
```

In [8]:

```
census.relabel('CENSUS2010POP', 'CEN2010')
census.relabel('POPESTIMATE2014', 'EST2014')
census
```

Out[8]:

SEX	AGE	CEN2010	EST2014
0	0	3944153	3948350
0	1	3978070	3962123
0	2	4096929	3957772
0	3	4119040	4005190
0	4	4063170	4003448
0	5	4056858	4004858
0	6	4066381	4134352
0	7	4030579	4154000
0	8	4046486	4119524
0	9	4148353	4106832

... (296 rows omitted)

In [ ]:

```
census.show()
# shows all rows of the table; output omitted from PDF
```

In [10]:

```
len(census)
```

Out[10]:

4

In [11]:

```
census['Change'] = census['EST2014'] - census['CEN2010']
census['Growth'] = census['Change'] / census['CEN2010']
census
```

Out[11]:

SEX	AGE	CEN2010	EST2014	Change	Growth
0	0	3944153	3948350	4197	0.00106411
0	1	3978070	3962123	-15947	-0.00400873
0	2	4096929	3957772	-139157	-0.0339662
0	3	4119040	4005190	-113850	-0.0276399
0	4	4063170	4003448	-59722	-0.0146984
0	5	4056858	4004858	-52000	-0.0128178
0	6	4066381	4134352	67971	0.0167154
0	7	4030579	4154000	123421	0.0306212
0	8	4046486	4119524	73038	0.0180497
0	9	4148353	4106832	-41521	-0.010009

... (296 rows omitted)

In [12]:

```
census.set_format('Growth', PercentFormatter)
census
```

Out[12]:

SEX	AGE	CEN2010	EST2014	Change	Growth
0	0	3944153	3948350	4197	0.11%
0	1	3978070	3962123	-15947	-0.40%
0	2	4096929	3957772	-139157	-3.40%
0	3	4119040	4005190	-113850	-2.76%
0	4	4063170	4003448	-59722	-1.47%
0	5	4056858	4004858	-52000	-1.28%
0	6	4066381	4134352	67971	1.67%
0	7	4030579	4154000	123421	3.06%
0	8	4046486	4119524	73038	1.80%
0	9	4148353	4106832	-41521	-1.00%

... (296 rows omitted)

In [13]:

```
census.column_labels
```

Out[13]:

```
('SEX', 'AGE', 'CEN2010', 'EST2014', 'Change', 'Growth')
```

In [14]:

```
len(census)
```

Out[14]:

```
6
```

In [15]:

```
len(census.rows)              # number of rows
```

Out[15]:

```
306
```

In [16]:

```
census.rows[0]
```

Out[16]:

```
Row(SEX=0, AGE=0, CEN2010=3944153, EST2014=3948350, Change=41
97, Growth=0.0010641067930174108)
```

In [17]:

```
census.columns[2]
```

Out[17]:

```
array([  3944153,   3978070,   4096929,   4119040,   4063170,
4056858,
         4066381,   4030579,   4046486,   4148353,   4172541,
4114415,
         4106243,   4118013,   4165982,   4242820,   4316139,
4395295,
         4500855,   4585234,   4519129,   4354294,   4264642,
4198571,
         4249363,   4262350,   4152305,   4248869,   4215249,
4223076,
         4285668,   3970218,   3986847,   3880150,   3839216,
3956434,
         3802087,   3934445,   4121880,   4364796,   4383274,
4114985,
         4076104,   4105105,   4211496,   4508868,   4519761,
4535265,
         4538796,   4605901,   4660295,   4464631,   4500846,
```

4380354,
        4291999,   4254709,   4037513,   3936386,   3794928,
3641269,
        3621131,   3492596,   3563182,   3483884,   2657131,
2680761,
        2639141,   2649365,   2323672,   2142324,   2043121,
1949323,
        1864275,   1736960,   1684487,   1620077,   1471070,
1455330,
        1400123,   1371195,   1308511,   1212865,   1161421,
1074809,
         985721,   914723,   814211,   712908,   640619,
537998,
         435563,   344987,   281389,   216978,   169449,
129717,
          95223,   68138,   45900,   32266,   53364,
308745538,
        2014276,   2030853,   2092198,   2104550,   2077550,
2072094,
        2075319,   2057076,   2065453,   2119696,   2135996,
2103264,
        2100145,   2104914,   2135543,   2177022,   2216034,
2263153,
        2305473,   2341984,   2308319,   2223198,   2177797,
2140799,
        2164063,   2161308,   2097088,   2140651,   2118605,
2117939,
        2160802,   1988155,   1994476,   1936863,   1916204,
1980916,
        1890595,   1953386,   2049720,   2167405,   2191249,
2047818,
        2028653,   2035990,   2090267,   2237450,   2230982,
2238248,
        2237734,   2264671,   2300354,   2190766,   2207246,
2141354,
        2093554,   2073473,   1956141,   1905355,   1834808,
1753871,
        1745507,   1679077,   1712692,   1672329,   1267895,
1273310,
        1248276,   1248906,   1087296,   994759,   945611,
900148,
         853726,   787863,   756624,   721008,   647804,
631884,
         602458,   579234,   543559,   494870,   462983,
419831,
         373131,   336819,   293120,   249803,   217436,
176689,
         136948,   103799,   81072,   59037,   43531,
30951,
          21424,   14556,   9259,   6073,   9162,
151781326,
        1929877,   1947217,   2004731,   2014490,   1985620,
1984764,

```
        1991062,   1973503,   1981033,   2028657,   2036545,
2011151,
        2006098,   2013099,   2030439,   2065798,   2100105,
2132142,
        2195382,   2243250,   2210810,   2131096,   2086845,
2057772,
        2085300,   2101042,   2055217,   2108218,   2096644,
2105137,
        2124866,   1982063,   1992371,   1943287,   1923012,
1975518,
        1911492,   1981059,   2072160,   2197391,   2192025,
2067167,
        2047451,   2069115,   2121229,   2271418,   2288779,
2297017,
        2301062,   2341230,   2359941,   2273865,   2293600,
2239000,
        2198445,   2181236,   2081372,   2031031,   1960120,
1887398,
        1875624,   1813519,   1850490,   1811555,   1389236,
1407451,
        1390865,   1400459,   1236376,   1147565,   1097510,
1049175,
        1010549,    949097,    927863,    899069,    823266,
823446,
         797665,    791961,    764952,    717995,    698438,
654978,
         612590,    577904,    521091,    463105,    423183,
361309,
         298615,    241188,    200317,    157941,    125918,
98766,
          73799,     53582,     36641,     26193,     44202,
156964212])
```

In [18]:

```
census.rows[0][2]
```

Out[18]:

```
3944153
```

In [19]:

```
census.columns[2][0]
```

Out[19]:

```
3944153
```

In [20]:

```
census.where('AGE', 999)          # selecting rows
```

Out[20]:

SEX	AGE	CEN2010	EST2014	Change	Growth
0	999	308745538	318857056	10111518	3.28%
1	999	151781326	156936487	5155161	3.40%
2	999	156964212	161920569	4956357	3.16%

In [28]:

```
males = census.where('SEX', 1)

""" males.sort('Growth', descending=True).show()
would show the full table,
but we'll just look at the first few rows."""

males.sort('Growth', descending=True)
```

Out[28]:

SEX	AGE	CEN2010	EST2014	Change	Growth
1	100	9162	13729	4567	49.85%
1	99	6073	9037	2964	48.81%
1	98	9259	13649	4390	47.41%
1	96	21424	31235	9811	45.79%
1	93	59037	85980	26943	45.64%
1	94	43531	62130	18599	42.73%
1	97	14556	20479	5923	40.69%
1	95	30951	42824	11873	38.36%
1	92	81072	109873	28801	35.53%
1	91	103799	138080	34281	33.03%

... (92 rows omitted)

In [29]:

```
"""The corresponding calculation for females.
Compare the AGE column in this table
with the males' AGE column above.
The two columns look quite different."""

females = census.where('SEX', 2)

females.sort('Growth', descending=True)
```

Out[29]:

SEX	AGE	CEN2010	EST2014	Change	Growth
2	100	44202	58468	14266	32.27%
2	64	1389236	1826662	437426	31.49%
2	67	1400459	1832245	431786	30.83%
2	71	1049175	1350392	301217	28.71%
2	98	36641	46536	9895	27.01%
2	93	157941	200353	42412	26.85%
2	66	1390865	1758649	367784	26.44%
2	65	1407451	1776761	369310	26.24%
2	99	26193	32791	6598	25.19%
2	94	125918	156525	30607	24.31%

... (92 rows omitted)

The growth rates are complicated to compare across categories like age and gender. They are affected not only by the numerator, which is the amount of change, but also by the denominator, which is the count in 2010. If that count is small, which is the case for example with the number of males aged 99, then even small changes might result in large rates.

It is easier to compare the changes, without converting them to rates. The amount of change, or in other words, the number of people in the difference, matters in the determination of resources for health and human services, transportation, Social Security, etc.

In [30]:

```
males.sort('Change', descending=True)
```

Out[30]:

SEX	AGE	CEN2010	EST2014	Change	Growth
1	999	151781326	156936487	5155161	3.40%
1	67	1248906	1653257	404351	32.38%
1	64	1267895	1661474	393579	31.04%
1	66	1248276	1589127	340851	27.31%
1	65	1273310	1607688	334378	26.26%
1	34	1916204	2192455	276251	14.42%
1	71	900148	1169356	269208	29.91%
1	23	2140799	2399883	259084	12.10%
1	59	1753871	2006900	253029	14.43%
1	24	2164063	2391398	227335	10.51%

... (92 rows omitted)

In [34]:

```
"""The corresponding calculation for females.
Compare the AGE column in this table
with the males' AGE column above.
The two columns look quite similar."""

females.sort('Change', descending=True)
```

Out[34]:

SEX	AGE	CEN2010	EST2014	Change	Growth
2	999	156964212	161920569	4956357	3.16%
2	64	1389236	1826662	437426	31.49%
2	67	1400459	1832245	431786	30.83%
2	65	1407451	1776761	369310	26.24%
2	66	1390865	1758649	367784	26.44%
2	71	1049175	1350392	301217	28.71%
2	59	1887398	2148517	261119	13.83%
2	34	1923012	2170440	247428	12.87%
2	23	2057772	2298701	240929	11.71%
2	70	1097510	1317238	219728	20.02%

... (92 rows omitted)