In [20]:
```
from __future__ import print_function
__Author__ = 'dp1618'
import pylab as pl
import numpy as np
import pandas as pd
import statsmodels.formula.api as sm
import statsmodels.api as sm1
import csv
from scipy import stats
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [2]:
```
#Import Files
dfpop = pd.read_csv('Population by FIPS County.csv')
dfpop['FIPS'] = dfpop['County FIPS']
dfpop.columns
#print (dfpop)
```

Out[2]:
```
Index([u'Area Name-Legal/Statistical Area Description', u'Qualifying Name',
       u'Area (Land)', u'Area (Water)', u'Summary Level',
       u'Geographic Component', u'Region', u'Division', u'County FIPS',
       u'State (FIPS)', u'County', u'Total Population', u'Area Total',
       u'Area Total: Area (Land)', u'Area Total: Area (Water)', u'FIPS'],
      dtype='object')
```

In [3]:
```
dfemit = pd.read_csv('CountiesBySector.csv', skiprows = 14, header = True)
dfemit.columns
#print (dfemit)
```

Out[3]:
```
Index([u'State', u' County', u'FIPS', u' Total', u'Unnamed: 4', u'Commercial',
       u'Industrial', u'Residential', u'Electricity Prod', u'Onroad',
       u'Cement', u'Aircraft', u'Airborne', u'Nonroad'],
      dtype='object')
```

In [4]:
```
dfMSA = pd.read_csv('CountiesMSACodes.csv', skiprows = 2)
dfMSA.columns
#print (dfMSA)
```

Out[4]:
```
Index([u'CBSA Code', u'Metro Division Code', u'CSA Code', u'CBSA Title',
       u'Level of CBSA', u'Status, 1=metro 2=micro',
       u'Metropolitan Division Title', u'CSA Title', u'Component Name',
       u'State', u'FIPS'],
      dtype='object')
```

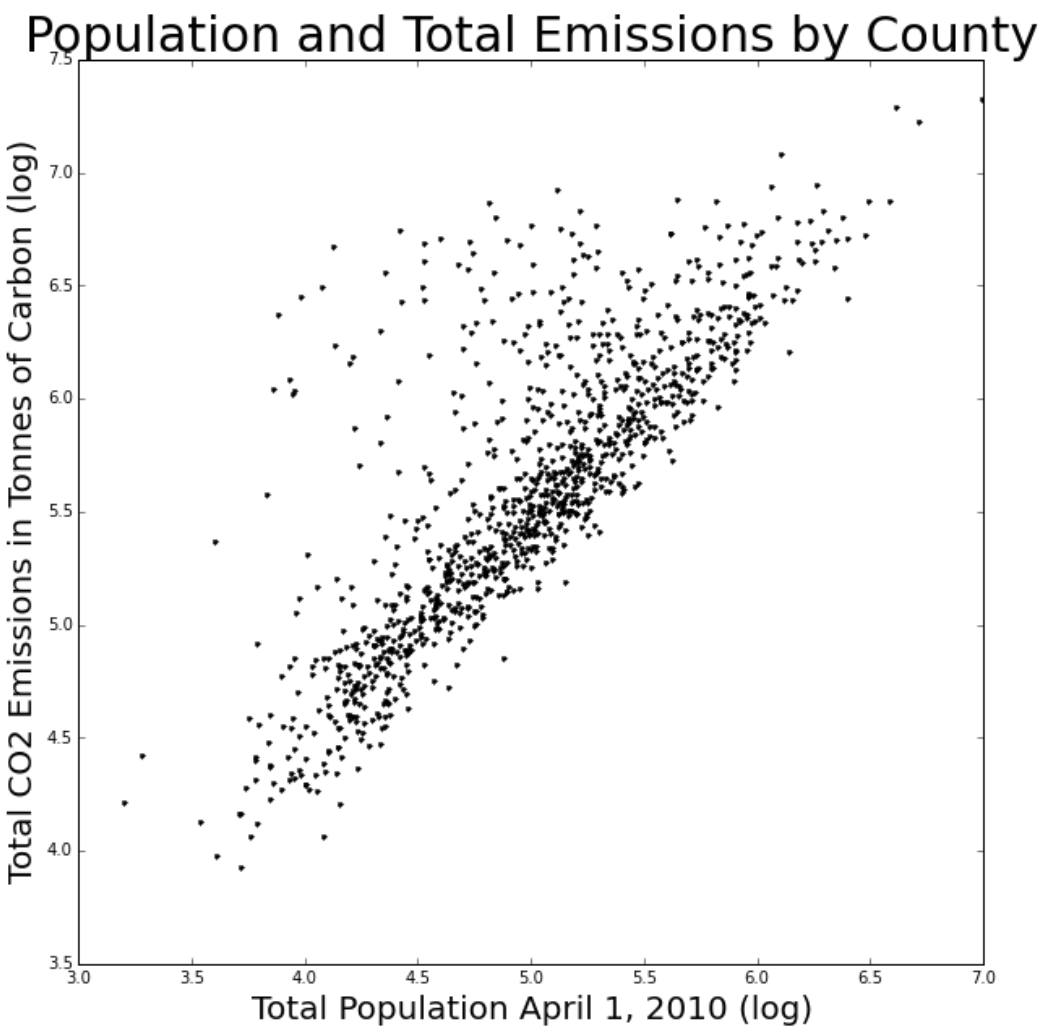
In [5]:
```
#Join the counties by sector and the counties with MSA Codes.
#Join over FIPS Column

merge1 = dfemit.merge(dfMSA, on='FIPS')
#merge1
```

In [6]:
```
#Merge population per CBSA Code
EmitandPop = pd.merge(merge1, dfpop, on = 'FIPS')
#EmitandPop = dfpop.merge(merge1, on = 'CBSA Code')
#print(EmitandPop)
EmitandPop[' Total'] = 1000000*EmitandPop[' Total']
```

In [7]:
```
EmitandPop = EmitandPop.drop(EmitandPop[EmitandPop['Status, 1=metro 2=micro'] == 2].index)
```

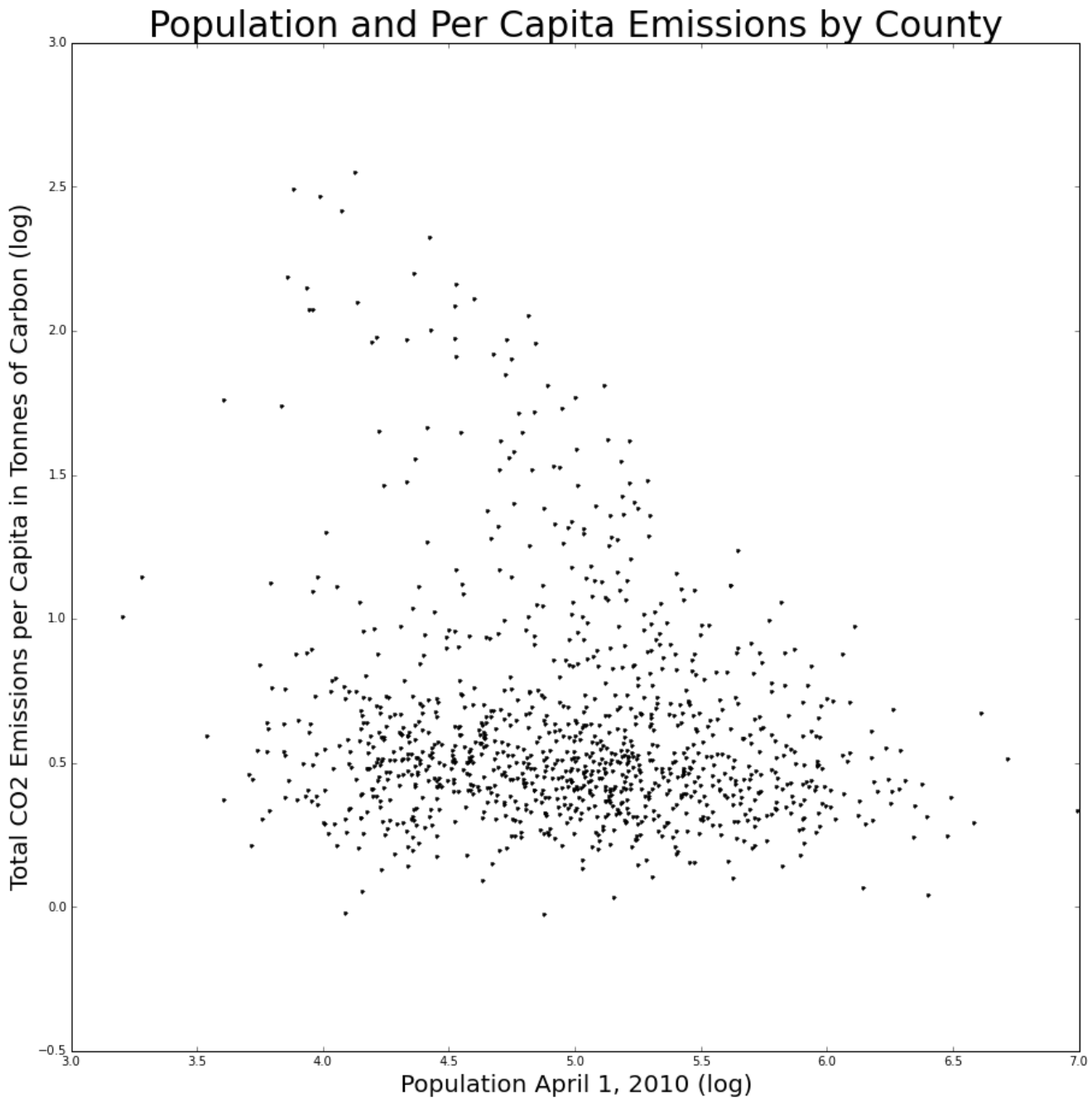
In [32]:
```
#Plot - Total CO2 Emissions and Population
pl.figure(figsize = (10,10))
pl.plot(np.log10(EmitandPop['Total Population']), np.log10(EmitandPop[' Total']), 'k.')
pl.xlabel('Total Population April 1, 2010 (log)', fontsize = 20)
pl.ylabel('Total CO2 Emissions in Tonnes of Carbon (log)', fontsize = 20)
pl.title ('Population and Total Emissions by County', fontsize = 30)

#Correlation Analysis:
cor = stats.pearsonr(EmitandPop['Total Population'], EmitandPop[' Total'])
print (cor)

#Strong positive correlation.  Pearsons Value of .755, where zero indicates no correlation and
#1 and -1 indicate strong correlation
```

```
(0.75541073023113547, 5.7151620561426247e-202)
```



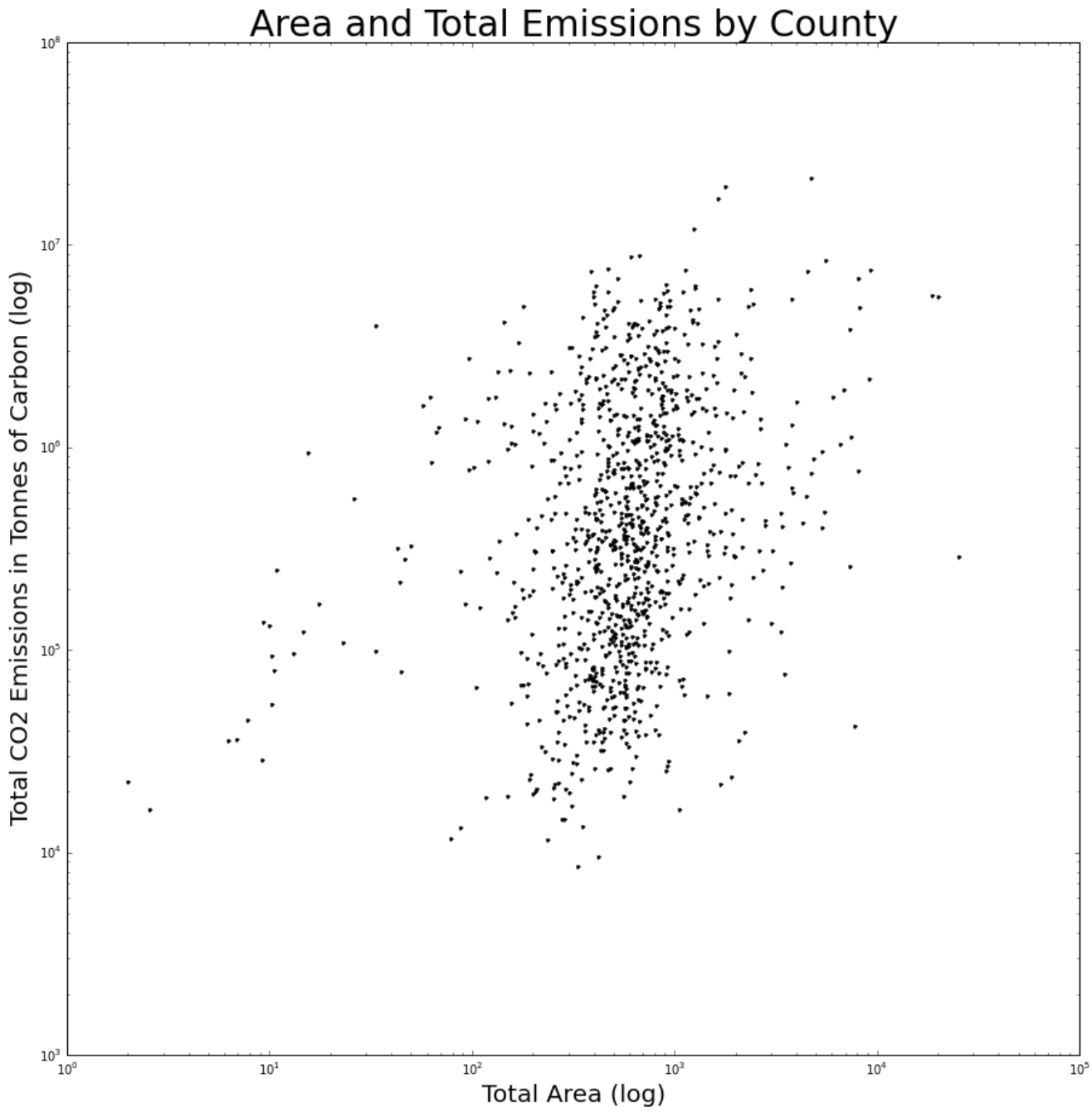
In [38]:
```
#Normalize CO2 Emissions per Capita and Plot CO2 Per Capita and Population

EmitandPop['CO2perCapita_Tonnes'] = ((EmitandPop[' Total']).astype(float) /
                                     (EmitandPop['Total Population']).astype(float))
pl.figure(figsize = (15,15))
pl.plot(log10(EmitandPop['Total Population']), log10(EmitandPop['CO2perCapita_Tonnes']), 'k.')
pl.xlabel('Population April 1, 2010 (log)', fontsize = 20)
pl.ylabel('Total CO2 Emissions per Capita in Tonnes of Carbon (log)', fontsize = 20)
pl.title ('Population and Per Capita Emissions by County', fontsize = 30)

#Correlation Analysis:
cor = stats.pearsonr(EmitandPop['Total Population'], EmitandPop['CO2perCapita_Tonnes'])
print (cor)
```

```
(-0.081107116374212807, 0.0073819052325340339)
```



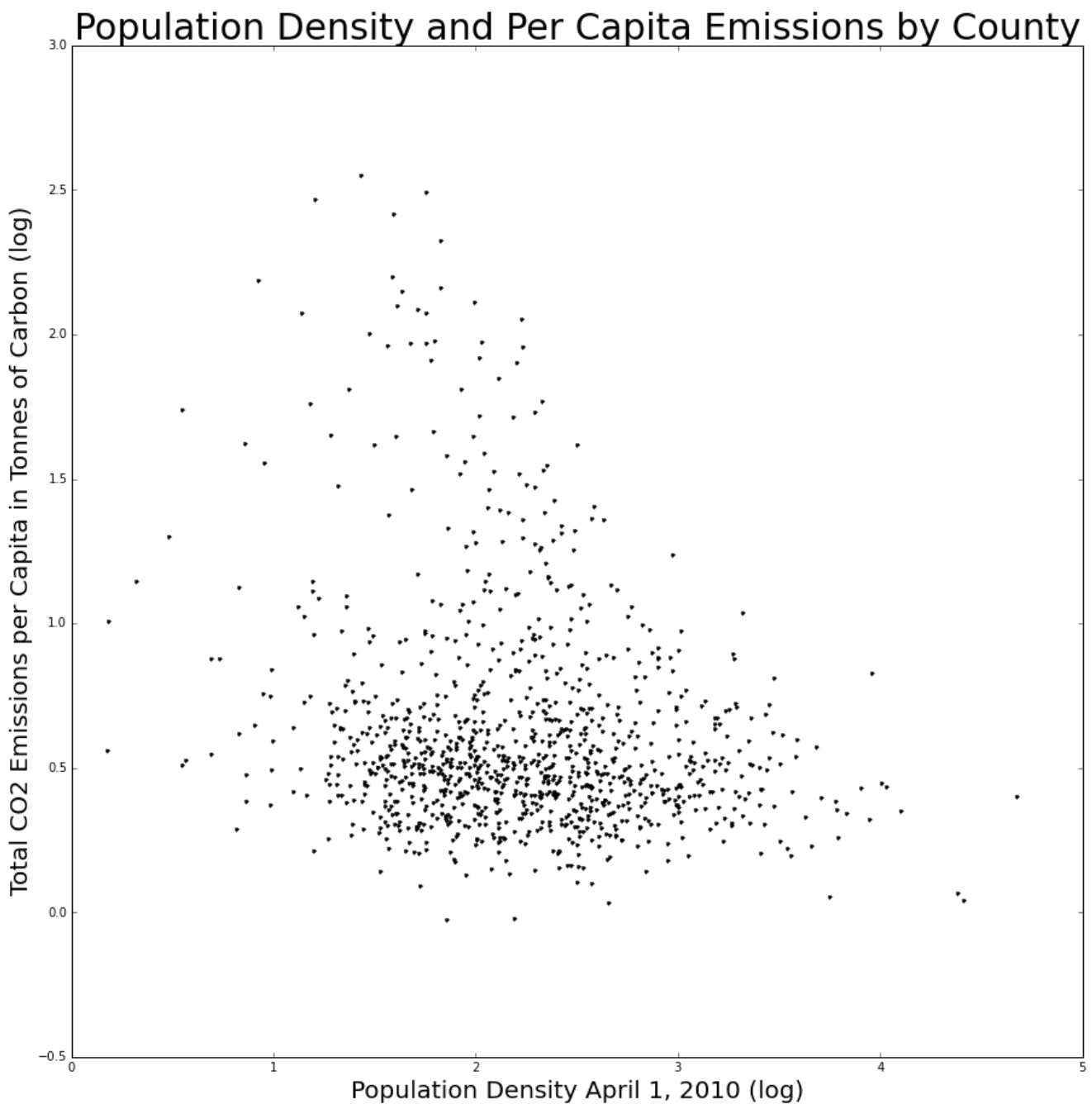
In [10]:
```
pl.figure(figsize = (15,15))
pl.loglog(EmitandPop['Area Total'], EmitandPop[' Total'], 'k.')
pl.xlabel('Total Area (log)', fontsize = 20)
pl.ylabel('Total CO2 Emissions in Tonnes of Carbon (log)', fontsize = 20)
pl.title ('Area and Total Emissions by County', fontsize = 30)
```

Out[10]: <matplotlib.text.Text at 0x10b22f650>



In [41]:
```
EmitandPop['PopDensity'] = ((EmitandPop['Total Population']).astype(float) /
                                        (EmitandPop['Area Total']).astype(float))
pl.figure(figsize = (15,15))
pl.plot(log10(EmitandPop['PopDensity']), log10(EmitandPop['CO2perCapita_Tonnes']), 'k.')
pl.xlabel('Population Density April 1, 2010 (log)', fontsize = 20)
pl.ylabel('Total CO2 Emissions per Capita in Tonnes of Carbon (log)', fontsize = 20)
pl.title ('Population Density and Per Capita Emissions by County', fontsize = 30)
cor = stats.pearsonr(EmitandPop['Total Population'], EmitandPop['CO2perCapita_Tonnes'])
print (cor)
```

```
(-0.081107116374212807, 0.0073819052325340339)
```



In [12]:
```
EmitandPop.columns
```

Out[12]:
```
Index([u'State_x', u' County', u'FIPS', u' Total', u'Unnamed: 4',
       u'Commercial', u'Industrial', u'Residential', u'Electricity Prod',
       u'Onroad', u'Cement', u'Aircraft', u'Airborne', u'Nonroad',
       u'CBSA Code', u'Metro Division Code', u'CSA Code', u'CBSA Title',
       u'Level of CBSA', u'Status, 1=metro 2=micro',
       u'Metropolitan Division Title', u'CSA Title', u'Component Name',
       u'State_y', u'Area Name-Legal/Statistical Area Description',
       u'Qualifying Name', u'Area (Land)', u'Area (Water)', u'Summary Level',
       u'Geographic Component', u'Region', u'Division', u'County FIPS',
       u'State (FIPS)', u'County', u'Total Population', u'Area Total',
       u'Area Total: Area (Land)', u'Area Total: Area (Water)',
       u'CO2perCapita_Tonnes', u'PopDensity'],
      dtype='object')
```

In [13]:
```
#Sort by CO2 emissions per capita
EmitandPoplimited = EmitandPop[['State_x', ' County', 'FIPS', ' Total', 'CBSA Code',
                                'CBSA Title', 'Total Population', 'Area Total',
                                'CO2perCapita_Tonnes', 'PopDensity']]
Sorted = EmitandPoplimited.sort(columns = 'CO2perCapita_Tonnes')
Sorted.head()
```

Out[13]:

	State_x	County	FIPS	Total	CBSA Code	CBSA Title	Total Population	Area Total	CO2perCapita_Tonnes	PopD
1461	TX	Coryell	48099	71437.166	28660	Killeen-Temple-Fort Hood, TX	75388	1056.756000	0.947593	71.33
1650	VA	Poquoson	51735	11658.060	47260	Virginia Beach-Norfolk-Newport News, VA-NC	12150	78.425670	0.959511	154.9
300	GA	Paulding	13223	154374.859	12060	Atlanta-Sandy Springs-Marietta, GA	142324	314.341700	1.084672	452.7
1014	NY	Kings	36047	2762535.626	35620	New York-Northern New Jersey-Long Island, NY-N...	2504700	96.917300	1.102941	2584
1645	VA	Manassas Park	51685	16285.060	47900	Washington-Arlington-Alexandria, DC-VA-MD-WV	14273	2.534767	1.140970	5630.

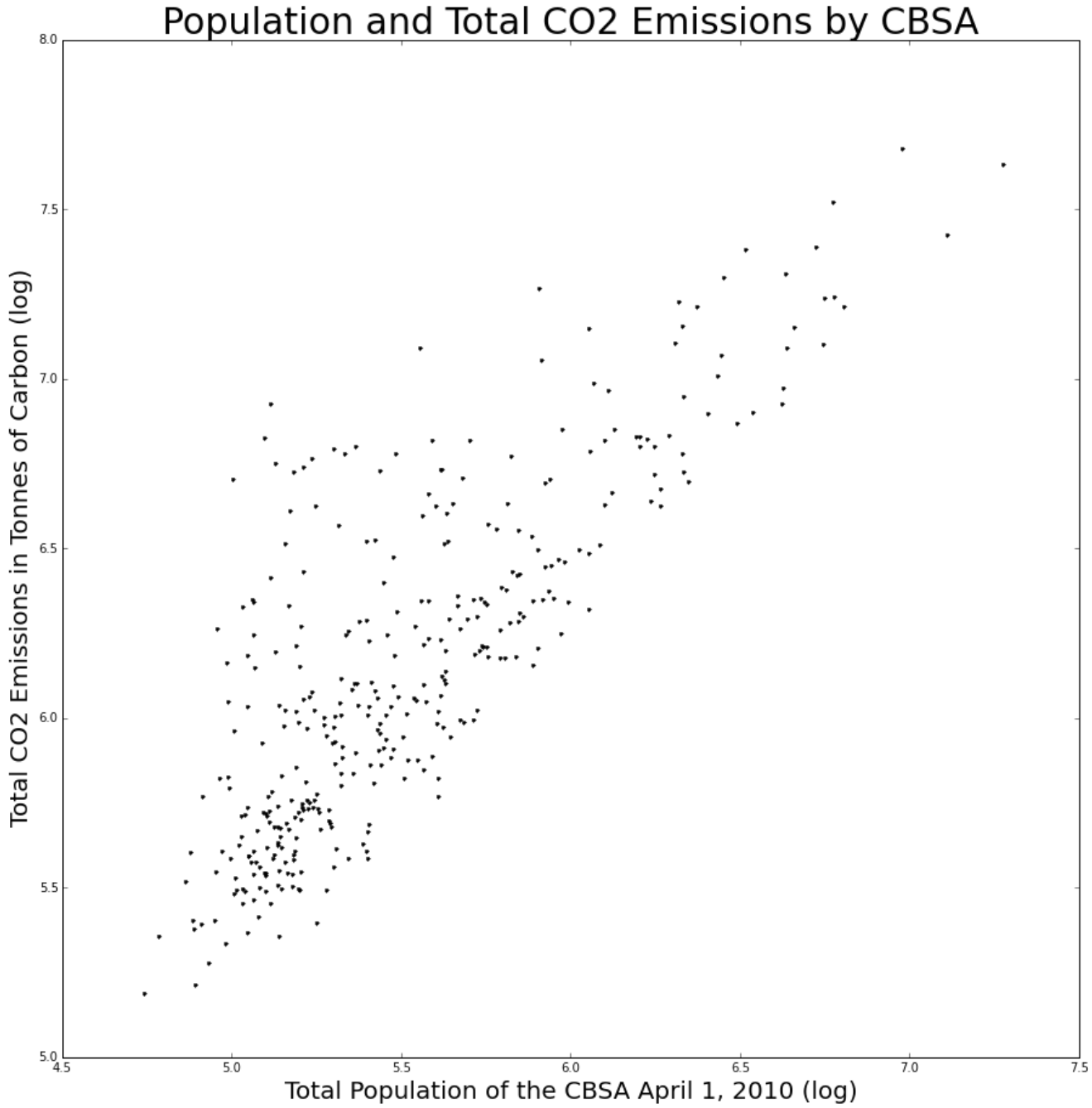
In [14]:
```
#Sum the Emissions over CBSA Code and re-run the analysis
EmitandPoplimited2 = EmitandPop[['State_x', ' County', ' Total', 'CBSA Code',
                                 'CBSA Title', 'Total Population', 'Area Total']]
EmitandPopbyCBSA = EmitandPoplimited2.groupby(['CBSA Title', 'CBSA Code']).sum()
EmitandPopbyCBSA.head()
```

Out[14]:

		Total	Total Population	Area Total
CBSA Title	**CBSA Code**			
Abilene, TX	**10180**	649929.762	165252	2757.7055
Akron, OH	**10420**	2048256.942	703200	924.1192
Albany, GA	**10500**	977521.540	157308	1957.9622
Albany-Schenectady-Troy, NY	**10580**	2816506.060	870716	2878.2170
Albuquerque, NM	**10740**	2265460.755	887077	9297.1590

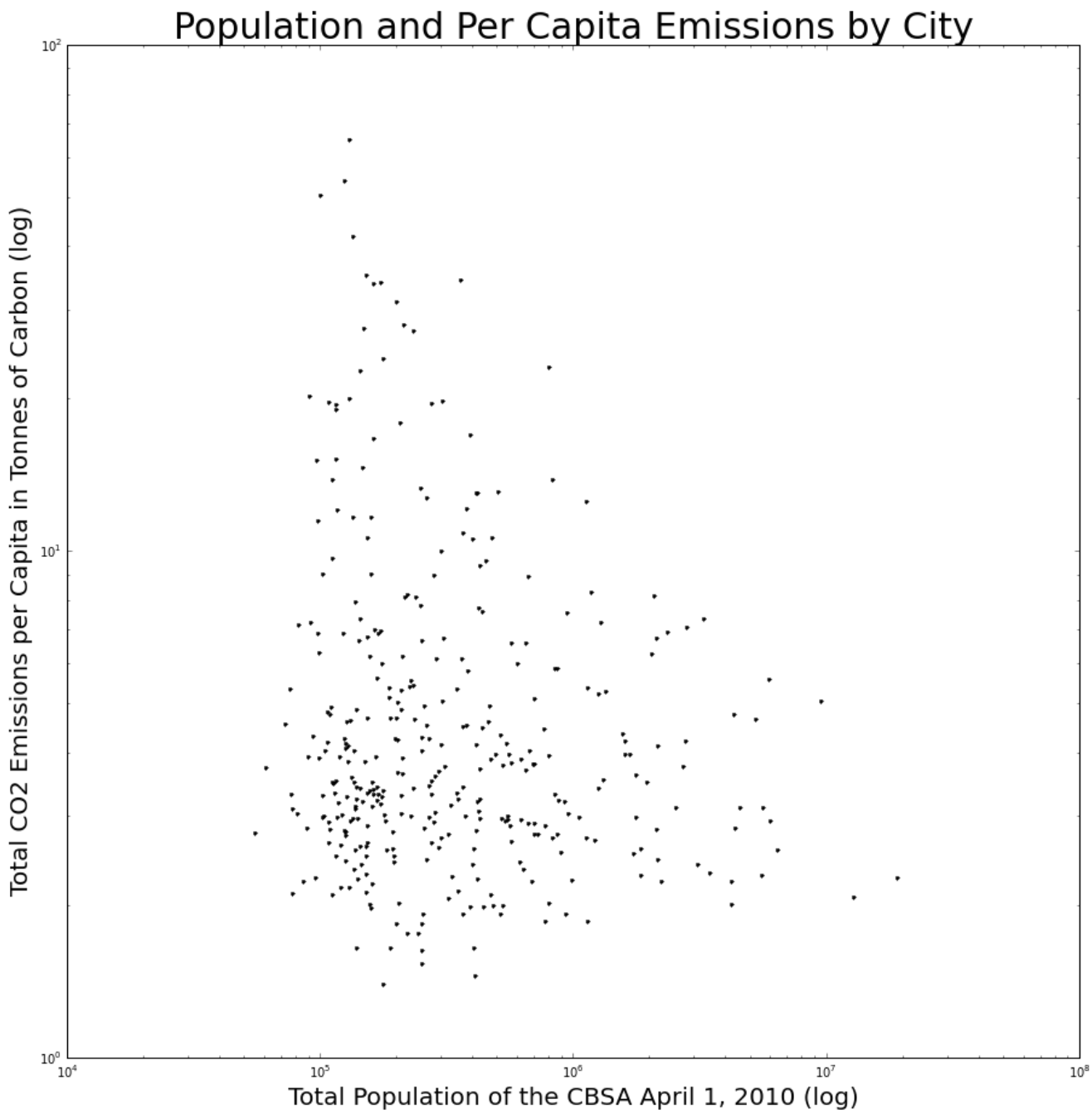
In [37]:
```
#plot grouped by CBSA
pl.figure(figsize = (15,15))
pl.plot(log10(EmitandPopbyCBSA['Total Population']), log10(EmitandPopbyCBSA[' Total']), 'k.')
pl.xlabel('Total Population of the CBSA April 1, 2010 (log)', fontsize = 20)
pl.ylabel('Total CO2 Emissions in Tonnes of Carbon (log)', fontsize = 20)
pl.title('Population and Total CO2 Emissions by CBSA', fontsize = 30)
#Correlation Analysis:
cor = stats.pearsonr(EmitandPopbyCBSA['Total Population'], EmitandPopbyCBSA[' Total'])
print (cor)
```

```
(0.85144115032796652, 6.1150188819454323e-103)
```



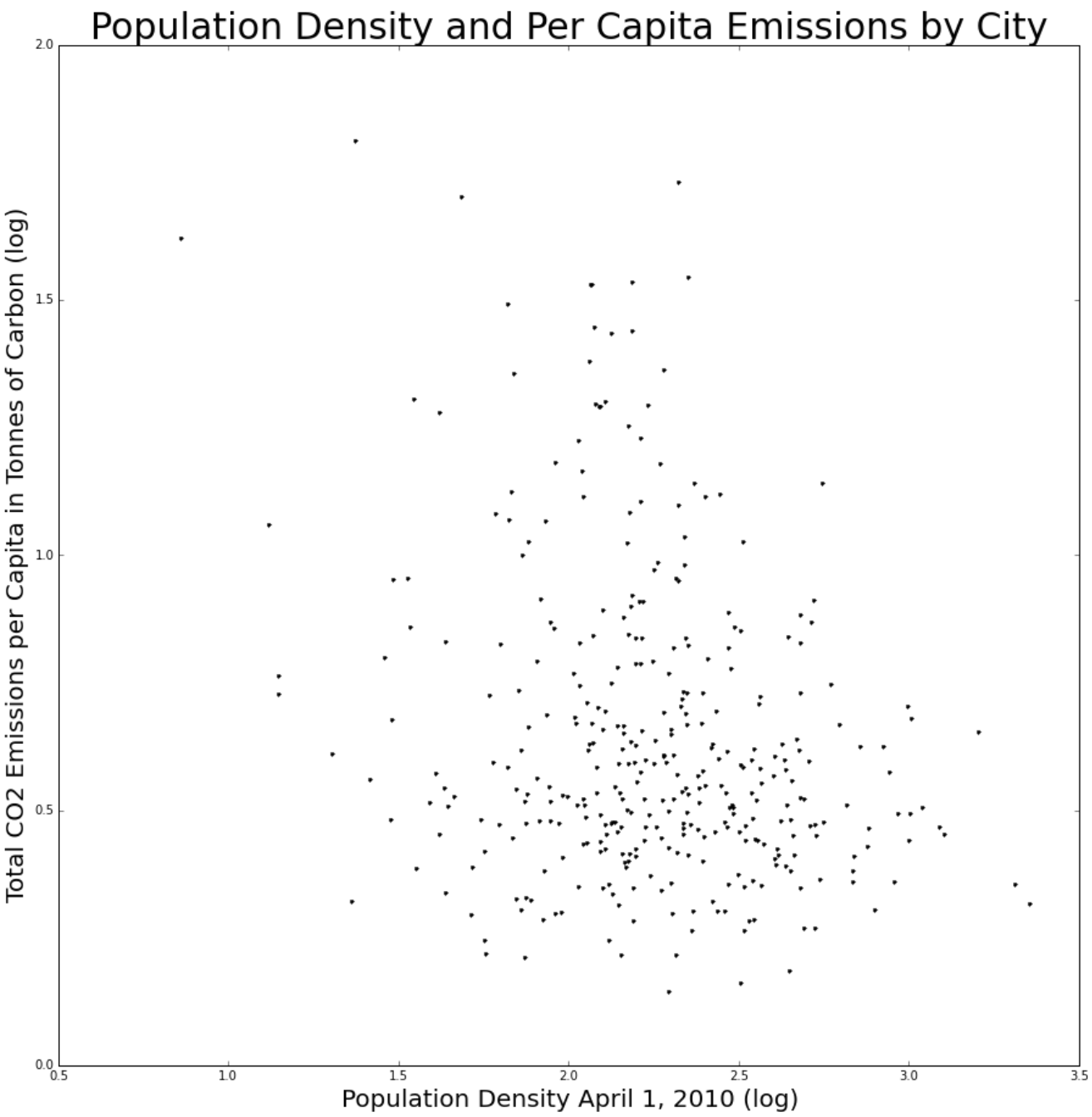
In [40]:
```
#Normalize CO2 Emissions per Capita and Plot CO2 Per Capita and Population

EmitandPopbyCBSA['CO2perCapita_Tonnes'] = ((EmitandPopbyCBSA[' Total']).astype(float) /
                                           (EmitandPopbyCBSA['Total Population']).astype(float))
pl.figure(figsize = (15,15))
pl.loglog(EmitandPopbyCBSA['Total Population'], EmitandPopbyCBSA['CO2perCapita_Tonnes'], 'k.')
pl.xlabel('Total Population of the CBSA April 1, 2010 (log)', fontsize = 20)
pl.ylabel('Total CO2 Emissions per Capita in Tonnes of Carbon (log)', fontsize = 20)
pl.title ('Population and Per Capita Emissions by City', fontsize = 30)
#Correlation Analysis:
cor = stats.pearsonr(EmitandPopbyCBSA['Total Population'],
                     EmitandPopbyCBSA['CO2perCapita_Tonnes'])
print (cor)
```

```
(-0.10455157895091345, 0.046833759702735837)
```



In [42]:
```
EmitandPopbyCBSA['PopDensity'] = ((EmitandPopbyCBSA['Total Population']).astype(float) /
                                  (EmitandPopbyCBSA['Area Total']).astype(float))
pl.figure(figsize = (15,15))
pl.plot(log10(EmitandPopbyCBSA['PopDensity']), log10(EmitandPopbyCBSA['CO2perCapita_Tonnes']), 'k.
')
pl.xlabel('Population Density April 1, 2010 (log)', fontsize = 20)
pl.ylabel('Total CO2 Emissions per Capita in Tonnes of Carbon (log)', fontsize = 20)
pl.title ('Population Density and Per Capita Emissions by City', fontsize = 30)
cor = stats.pearsonr(EmitandPopbyCBSA['Total Population'], EmitandPopbyCBSA['CO2perCapita_Tonnes']
)
print (cor)
```

```
(-0.10455157895091345, 0.046833759702735837)
```



In [17]:
```
Sorted2 = EmitandPopbyCBSA.sort(columns = 'CO2perCapita_Tonnes')
Sorted2.head()
```

Out[17]:

		Total	Total Population	Area Total	CO2perCapita_Tonnes
CBSA Title	**CBSA Code**				
Jacksonville, NC	**27340**	248844.833	177772	905.9130	1.399798
Brownsville-Harlingen, TX	**15180**	589178.470	406220	1276.4580	1.450393
Bremerton-Silverdale, WA	**14740**	385786.439	251133	565.9188	1.536184
Laredo, TX	**29700**	407319.151	250304	3375.5900	1.627298
Greenville, NC	**24780**	312057.515	189510	921.2242	1.646655

In [ ]:
```
```