The Effect of Pro Sports teams on a City's Economy

By Jae Hyun Lee, Jordan Bales, Ben Rogers

Executive Summary

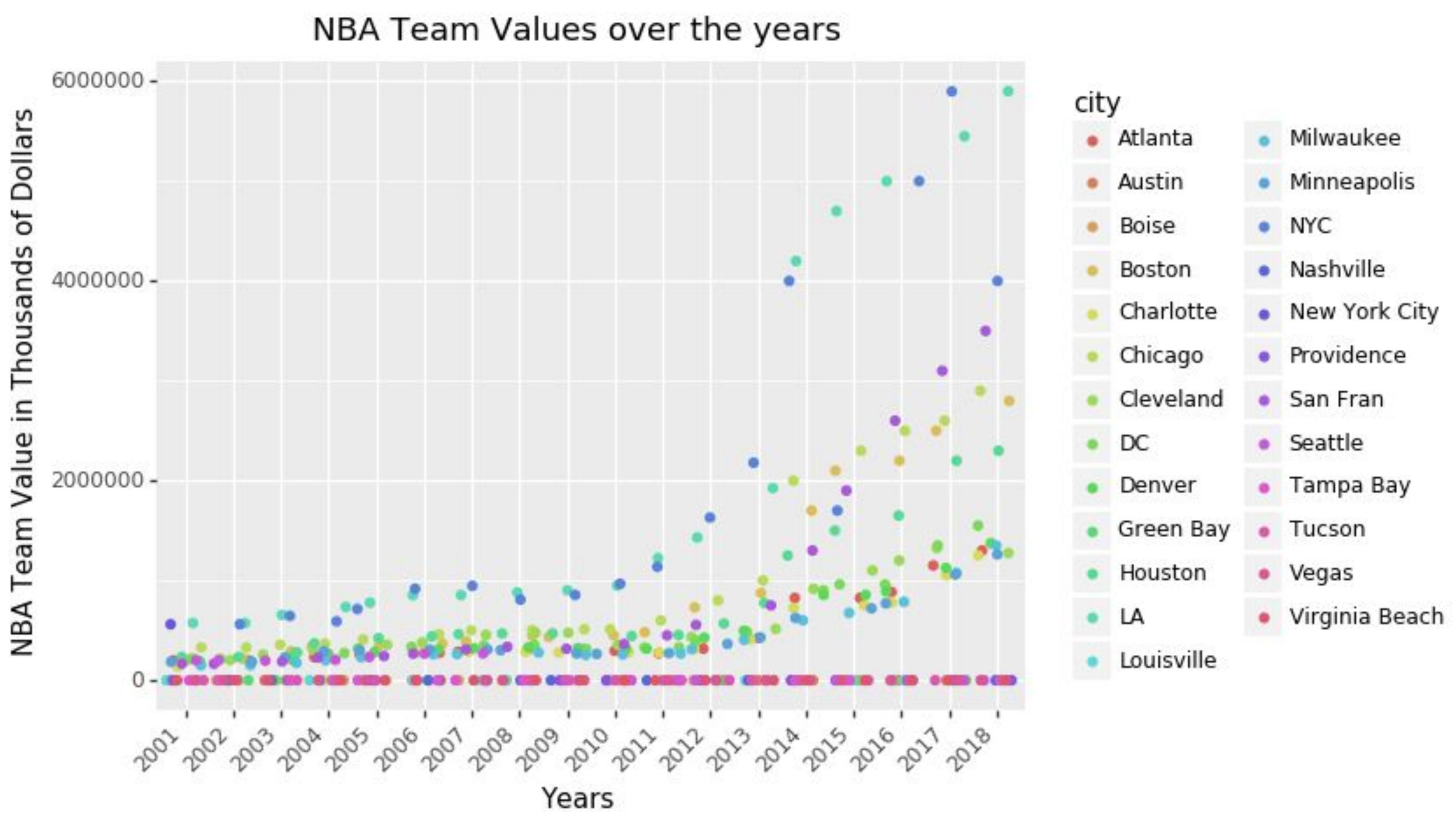
1. Does a city having a sports team affect its GDP?
2. Does market size matter to economic impact?
3. Does a city purchasing or relocating a sports team affect its GDP?

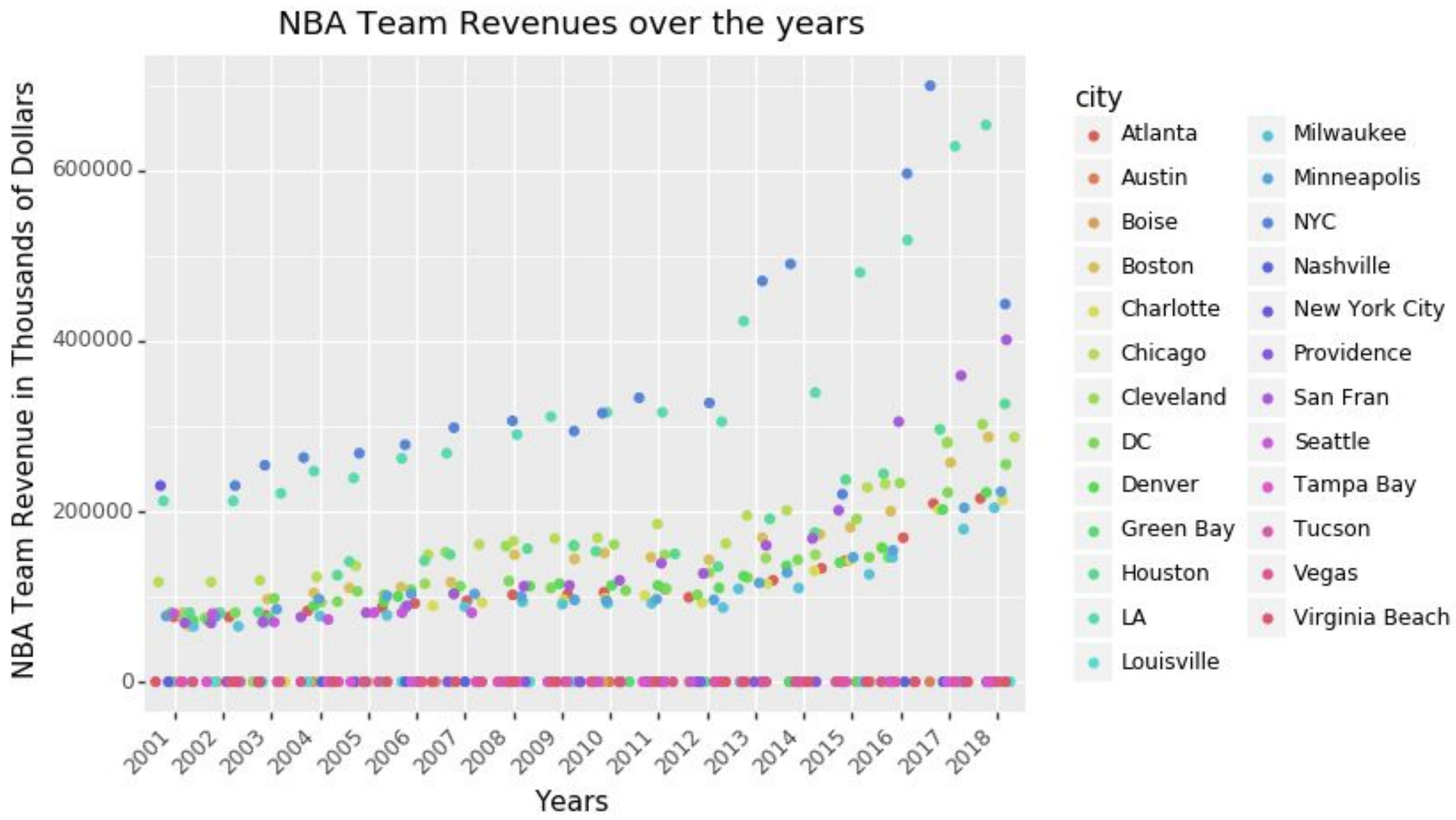
Data Summary

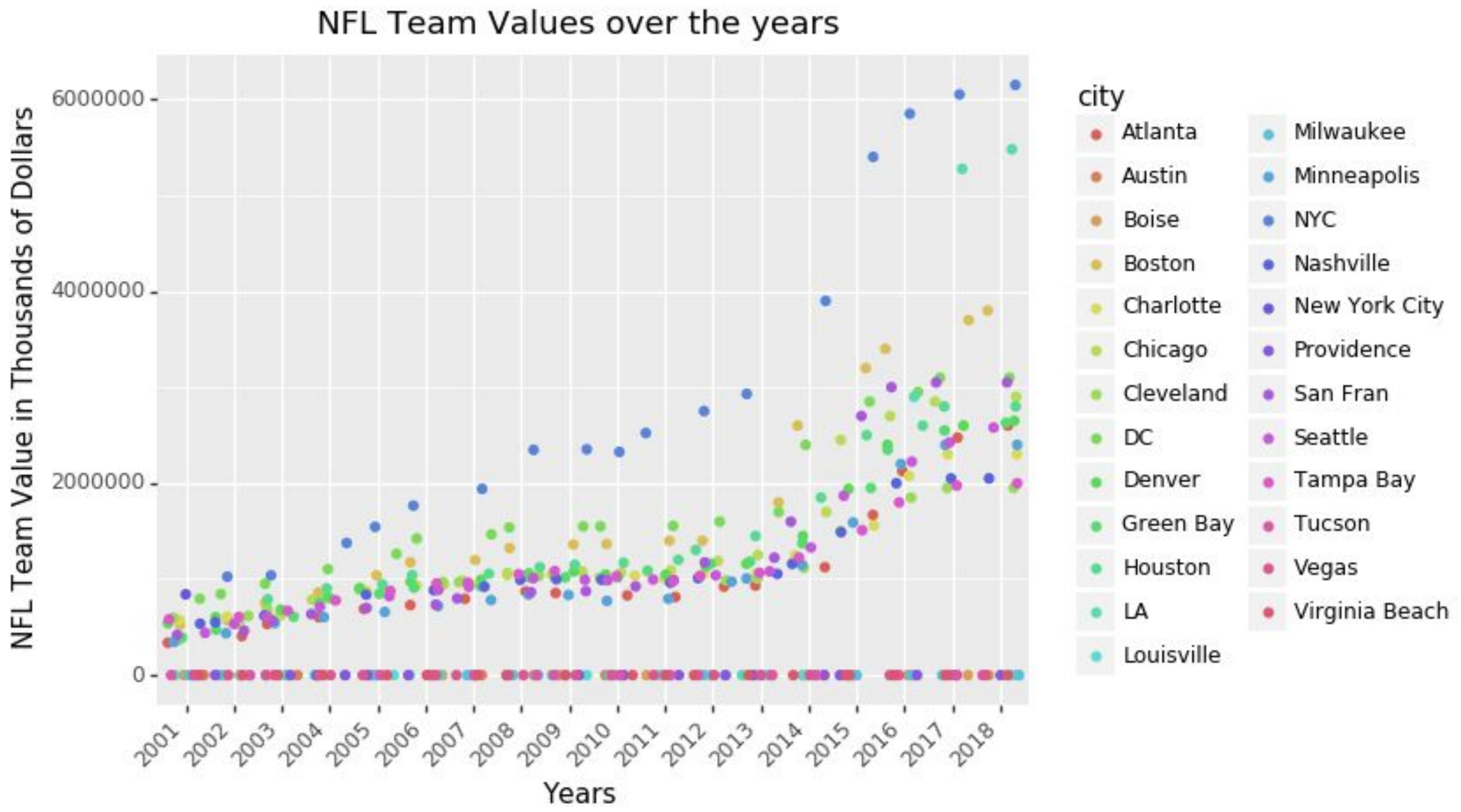
- Data from Bureau of Economic Analysis, Forbes, Sports Reference, ESPN
- Mixture of scrapped, CSV, and manually imputed

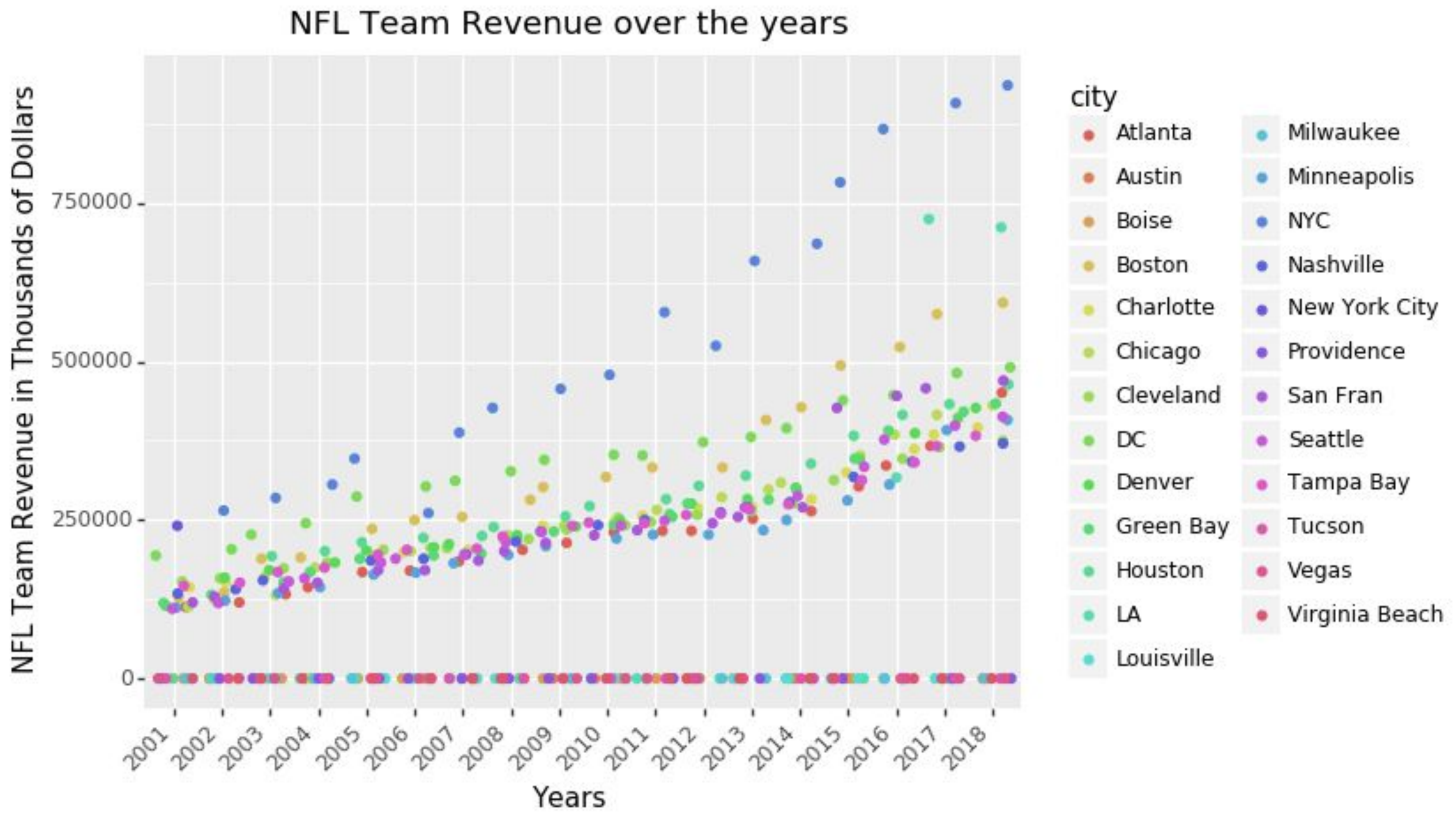
Data Summary

- Data from 24 cities from 2001 to 2018
    - 6 Large Market, 6 Medium Market
    - 6 Small Market, 6 No Teams
- 96 total variables

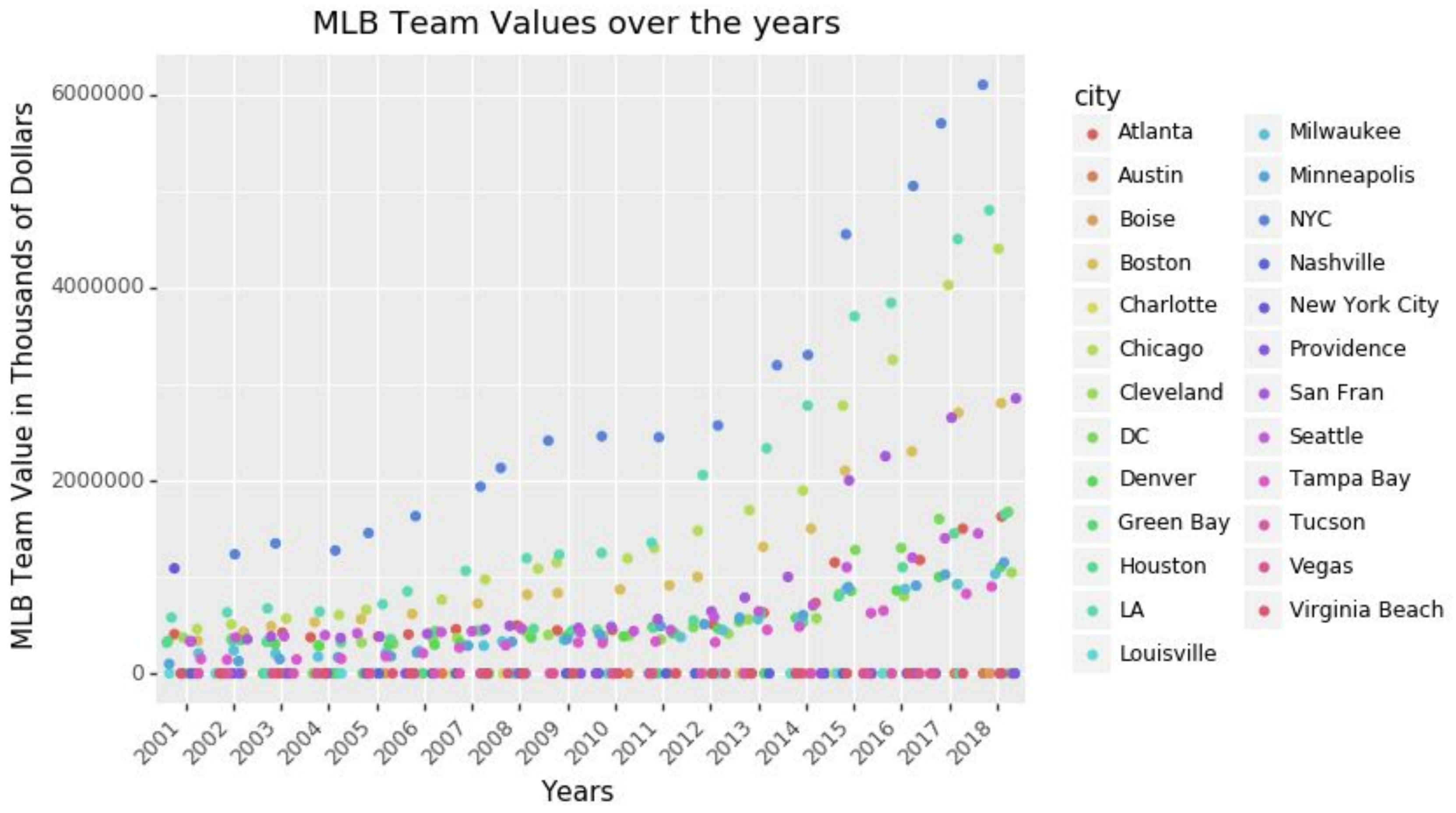

NBA Team Values over the years
NBA Team Value in Thousands of Dollars
6000000
4000000
2000000
0
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
Years
city
Atlanta
Austin
Boise
Boston
Charlotte
Chicago
Cleveland
DC
Denver
Green Bay
Houston
LA
Louisville
Milwaukee
Minneapolis
NYC
Nashville
New York City
Providence
San Fran
Seattle
Tampa Bay
Tucson
Vegas
Virginia Beach


NBA Team Revenues over the years
NBA Team Revenue in Thousands of Dollars
0
200000
400000
600000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
Years
city
Atlanta
Austin
Boise
Boston
Charlotte
Chicago
Cleveland
DC
Denver
Green Bay
Houston
LA
Louisville
Milwaukee
Minneapolis
NYC
Nashville
New York City
Providence
San Fran
Seattle
Tampa Bay
Tucson
Vegas
Virginia Beach

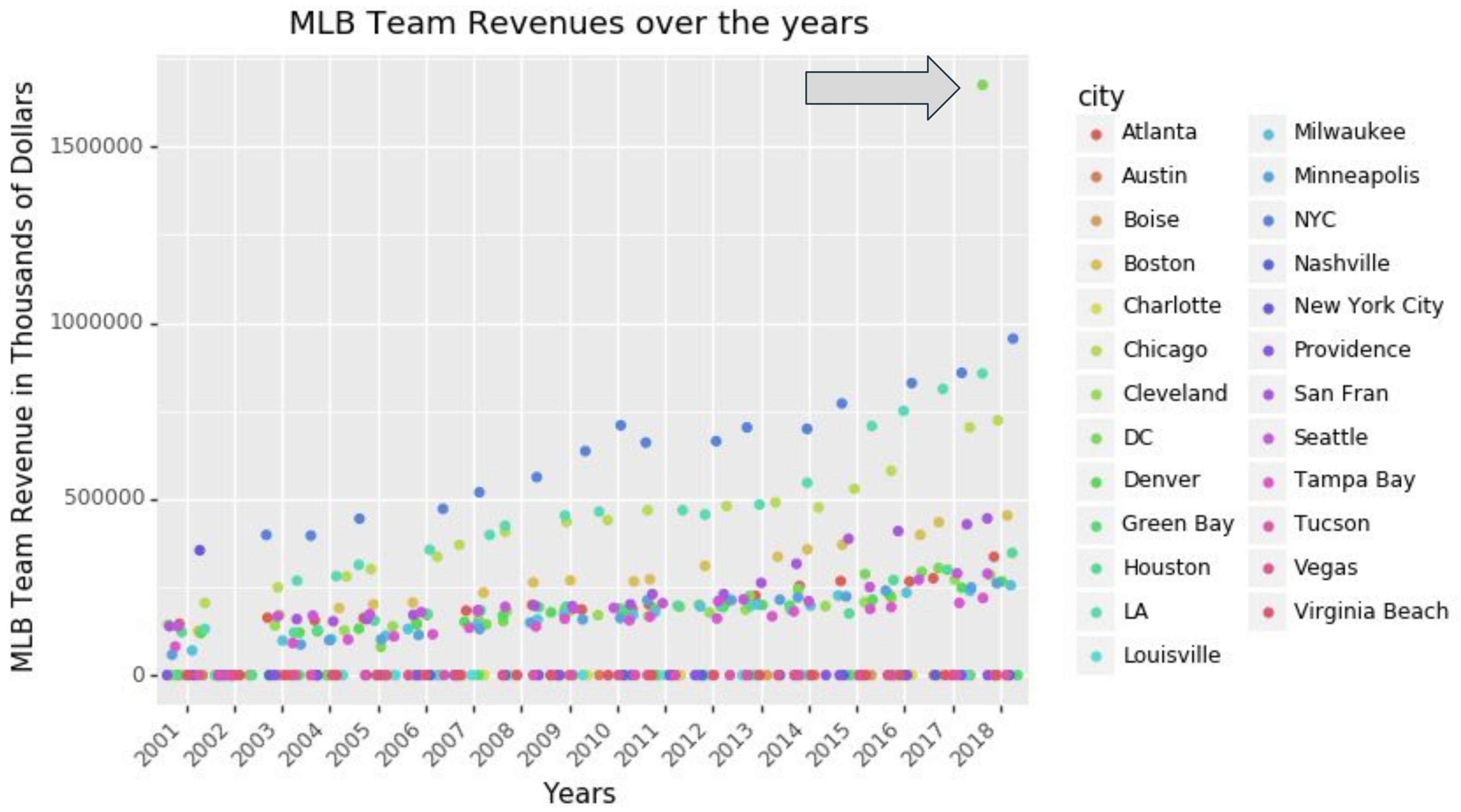

NFL Team Values over the years
NFL Team Value in Thousands of Dollars
6000000
4000000
2000000
0
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
Years
city
Atlanta
Austin
Boise
Boston
Charlotte
Chicago
Cleveland
DC
Denver
Green Bay
Houston
LA
Louisville
Milwaukee
Minneapolis
NYC
Nashville
New York City
Providence
San Fran
Seattle
Tampa Bay
Tucson
Vegas
Virginia Beach



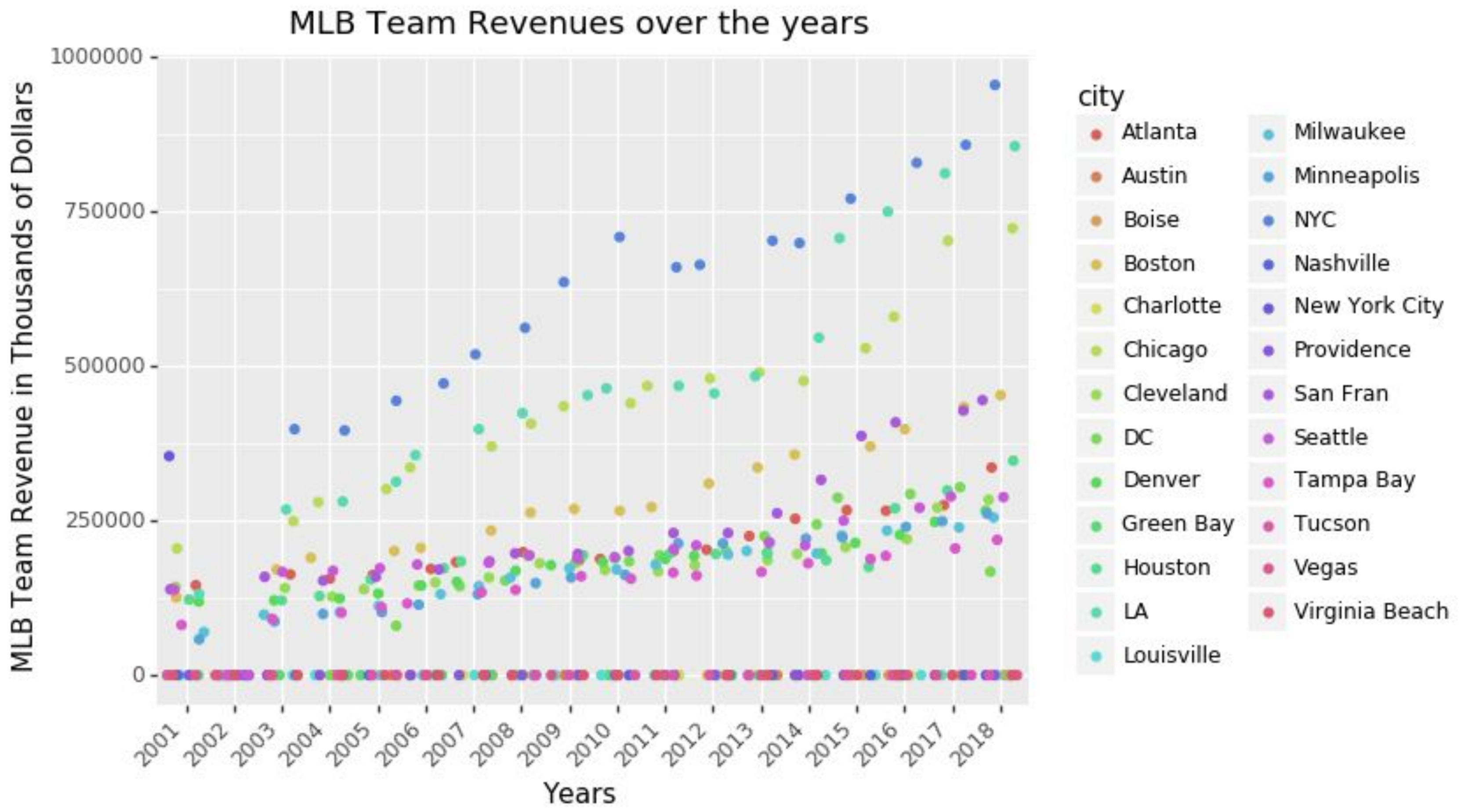
NFL Team Revenue over the years
NFL Team Revenue in Thousands of Dollars
750000
500000
250000
0
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
Years
city
Atlanta
Austin
Boise
Boston
Charlotte
Chicago
Cleveland
DC
Denver
Green Bay
Houston
LA
Louisville
Milwaukee
Minneapolis
NYC
Nashville
New York City
Providence
San Fran
Seattle
Tampa Bay
Tucson
Vegas
Virginia Beach



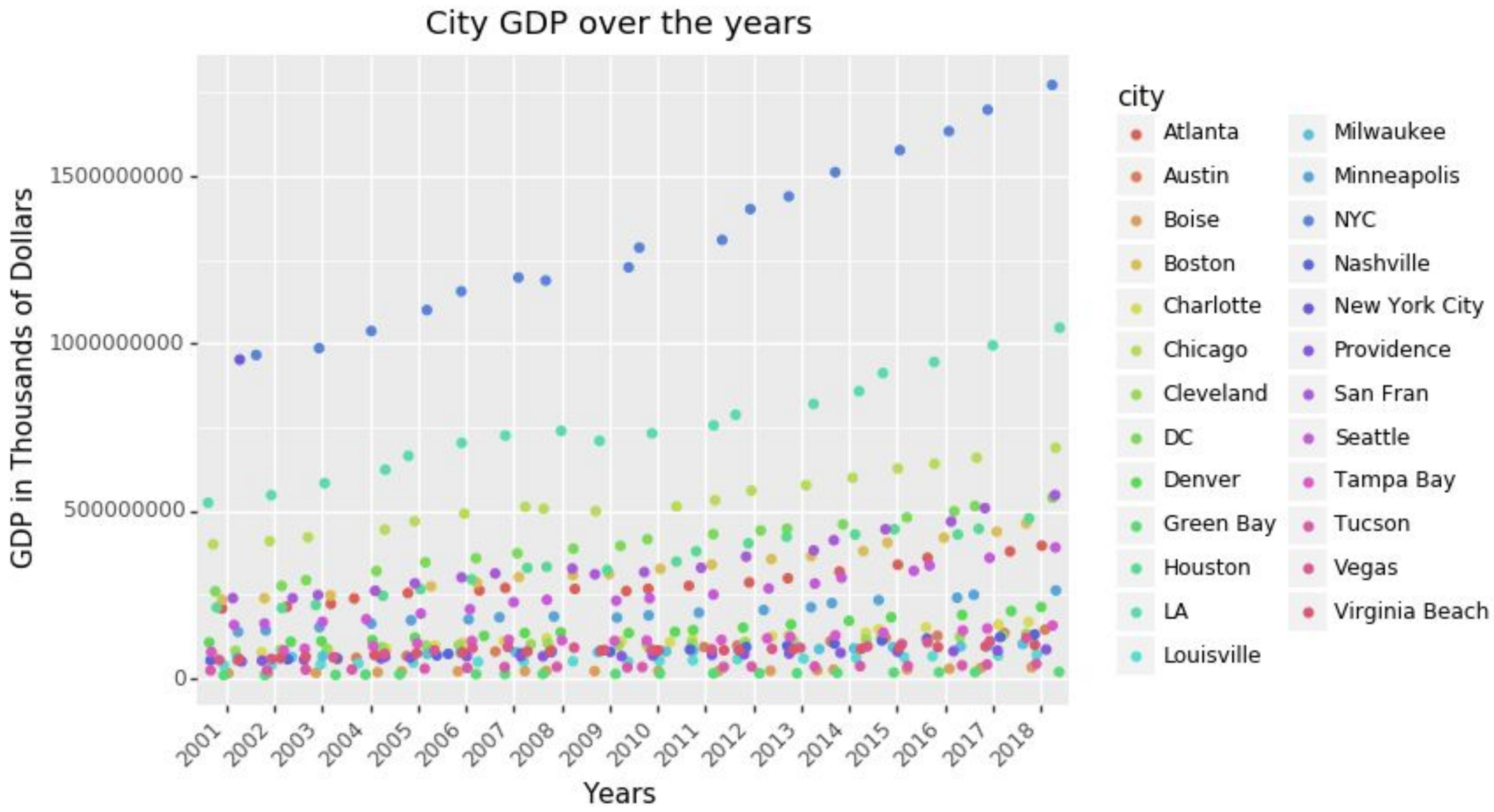
MLB Team Values over the years
MLB Team Value in Thousands of Dollars
6000000
4000000
2000000
0
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
Years
city
Atlanta
Austin
Boise
Boston
Charlotte
Chicago
Cleveland
DC
Denver
Green Bay
Houston
LA
Louisville
Milwaukee
Minneapolis
NYC
Nashville
New York City
Providence
San Fran
Seattle
Tampa Bay
Tucson
Vegas
Virginia Beach



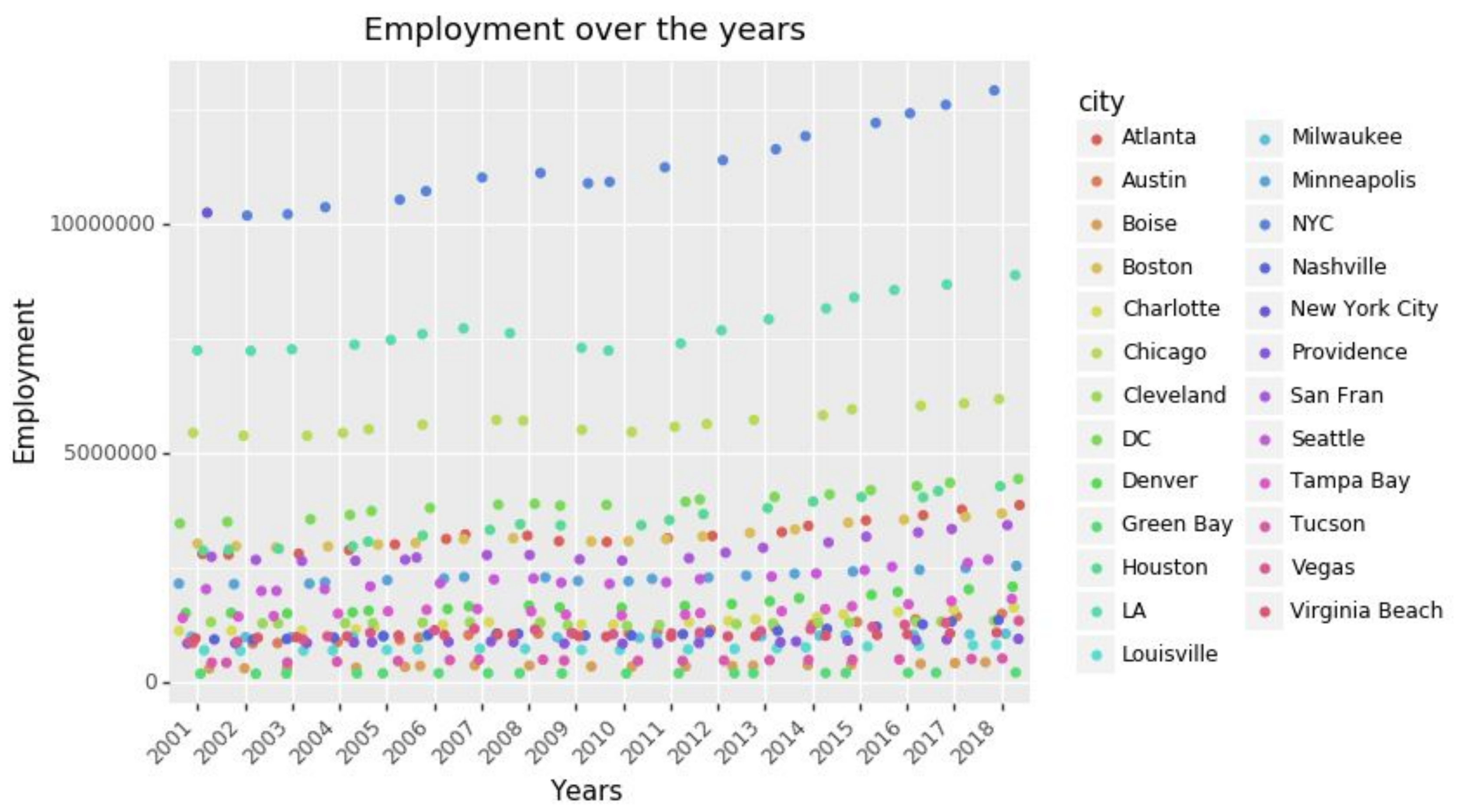
MLB Team Revenues over the years
MLB Team Revenue in Thousands of Dollars
0
500000
1000000
1500000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
Years
city
Atlanta
Austin
Boise
Boston
Charlotte
Chicago
Cleveland
DC
Denver
Green Bay
Houston
LA
Louisville
Milwaukee
Minneapolis
NYC
Nashville
New York City
Providence
San Fran
Seattle
Tampa Bay
Tucson
Vegas
Virginia Beach



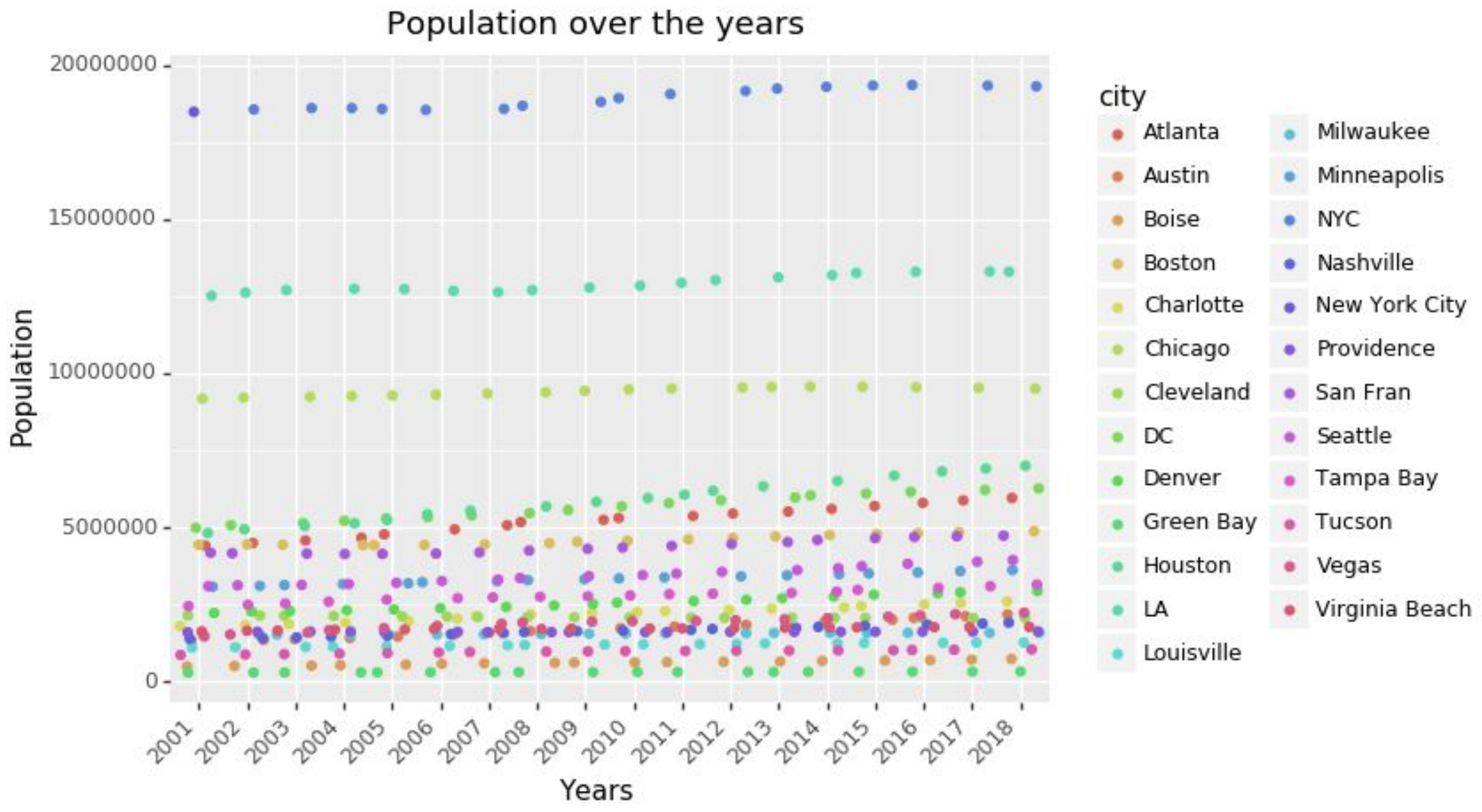
MLB Team Revenues over the years
MLB Team Revenue in Thousands of Dollars
1000000
750000
500000
250000
0
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
Years
city
Atlanta
Austin
Boise
Boston
Charlotte
Chicago
Cleveland
DC
Denver
Green Bay
Houston
LA
Louisville
Milwaukee
Minneapolis
NYC
Nashville
New York City
Providence
San Fran
Seattle
Tampa Bay
Tucson
Vegas
Virginia Beach



City GDP over the years
GDP in Thousands of Dollars
1500000000
1000000000
500000000
0
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
Years
city
Atlanta
Austin
Boise
Boston
Charlotte
Chicago
Cleveland
DC
Denver
Green Bay
Houston
LA
Louisville
Milwaukee
Minneapolis
NYC
Nashville
New York City
Providence
San Fran
Seattle
Tampa Bay
Tucson
Vegas
Virginia Beach



Employment over the years
Employment
10000000
5000000
0
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
Years
city
Atlanta
Austin
Boise
Boston
Charlotte
Chicago
Cleveland
DC
Denver
Green Bay
Houston
LA
Louisville
Milwaukee
Minneapolis
NYC
Nashville
New York City
Providence
San Fran
Seattle
Tampa Bay
Tucson
Vegas
Virginia Beach


Population over the years
Population
20000000
15000000
10000000
5000000
0
Years
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
city
Atlanta
Austin
Boise
Boston
Charlotte
Chicago
Cleveland
DC
Denver
Green Bay
Houston
LA
Louisville
Milwaukee
Minneapolis
NYC
Nashville
New York City
Providence
San Fran
Seattle
Tampa Bay
Tucson
Vegas
Virginia Beach

Variable Transformation and Preprocessing

Import Libraries

```python
[1]: import os
     import pandas as pd
     import numpy as np
```

Import Data

```python
[2]: df = pd.read_csv("City_combined_v4.csv")
```

Variable Transformation and Preprocessing

Quick Summary

[3]:
```
df.head()
```

[3]:

	city	gdp	private_industries	agriculture	mining	utilities	construction	manufacturing	durable_goods_manufacturing	nondurable_goods_manufacturing	...	super_bowl_winner	nfl
0	Atlanta	209741698	191749470	366262	331505	2100718	(D)	(D)	(D)	(D)	...	0	
1	Austin	55307638	47708348	21993	205947	381802	4324915	7829516	(D)	(D)	...	0	
2	Boise	16435618	14448912	(D)	13511	57379	1174578	3196453	(D)	(D)	...	0	
3	Boston	236051306	213403308	(D)	87120	2666446	9560212	26949729	(D)	(D)	...	1	
4	Charlotte	76027214	69766921	(D)	105937	1805777	3895799	14555666	(D)	(D)	...	0	

5 rows × 97 columns

[4]:
```
df.describe()
```

[4]:

	gdp	private_industries	finance_total	government_and_government_enterprises	personal_income_total	net_earnings_by_place_of_residence	personal_current_transfer_receipts	incom
count	4.320000e+02	4.320000e+02	4.320000e+02	4.320000e+02	4.320000e+02	4.320000e+02	4.320000e+02	
mean	2.452478e+08	2.186487e+08	5.642425e+07	2.659914e+07	1.998965e+08	1.348661e+08	2.614331e+07	
std	2.928506e+08	2.652310e+08	8.719089e+07	3.019768e+07	2.384918e+08	1.566991e+08	3.434414e+07	
min	1.091764e+07	9.892020e+06	1.855698e+06	1.025624e+06	8.653562e+06	6.047825e+06	9.235270e+05	
25%	7.513007e+07	6.255351e+07	1.369553e+07	8.118968e+06	6.317938e+07	4.232933e+07	8.789657e+06	
50%	1.227686e+08	1.106198e+08	2.684385e+07	1.432709e+07	1.036139e+08	6.841517e+07	1.485404e+07	
75%	3.188245e+08	2.803013e+08	6.003646e+07	3.140421e+07	2.529817e+08	1.735379e+08	2.797278e+07	
max	1.772320e+09	1.611478e+09	5.751021e+08	1.608420e+08	1.480233e+09	9.413716e+08	2.100819e+08	

8 rows × 64 columns

Variable Transformation and Preprocessing

```
[5]: numRow = len(df)
     numCol = len(df.columns)
     print("Total number of rows: {}".format(numRow))
     print("Total number of columns: {}".format(numCol))
```

```
Total number of rows: 432
Total number of columns: 97
```

```
[6]: pd.set_option('display.max_rows', None)
     df.dtypes
```

```
[6]: city                                                                      object
     gdp                                                                        int64
     private_industries                                                         int64
     agriculture                                                               object
     mining                                                                    object
     utilities                                                                 object
     construction                                                              object
     manufacturing                                                             object
     durable_goods_manufacturing                                               object
     nondurable_goods_manufacturing                                            object
     wholesale_trade                                                           object
     retail_trade                                                              object
     transportation_and_warehousing                                            object
     information                                                               object
     finance_total                                                              int64
     finance                                                                   object
     real_estate                                                               object
     professional_and_business_services                                        object
     professional_scientific_and_technical_services                            object
     management_of_companies_and_enterprises                                   object
     administrative_and_support_and_waste_management_and_remediation services  object
     educational_services_health_care_and_social_assistance                    object
     educational_services                                                      object
     health_care_and_social_assistance                                         object
     arts_total                                                                object
```

Variable Transformation and Preprocessing

Preprocessing

[7]:
```
# Replace all missing data with np.NaN
df = df.replace(to_replace='(D)', value=np.NaN)
df = df.replace(to_replace='(L)', value=np.NaN)
df = df.replace(to_replace='na', value=np.NaN)
```

[8]:
```
# Find percentage of np.NaN per column
x = df.isnull().sum().sort_values(ascending=False) / len(df)
```

[9]:
```
# Drop any column with 25% or more missing data
df = df.loc[:, x < .25]
```

[10]:
```
# Expected to drop 20 columns
numColNew = len(df.columns)
print("{} column(s) have been dropped; {} columns remain.".format(numCol - numColNew, numColNew))
```

```
20 column(s) have been dropped; 77 columns remain.
```

Variable Transformation and Preprocessing

[11]:
```
# Check dtypes of columns
df.dtypes
```

[11]:
```
city                                                 object
gdp                                                   int64
private_industries                                    int64
mining                                               object
construction                                         object
manufacturing                                        object
retail_trade                                         object
finance_total                                         int64
finance                                              object
real_estate                                          object
professional_and_business_services                   object
educational_services_health_care_and_social_assistance    object
arts_total                                           object
other_services                                       object
government_and_government_enterprises                 int64
personal_income_total                                 int64
net_earnings_by_place_of_residence                    int64
personal_current_transfer_receipts                    int64
income_maintenance_benefits                           int64
unemployment_insurance_compensation                   int64
retirement_and_other                                  int64
dividends_interest_and_rent                           int64
population                                            int64
per_capita_personal_income                            int64
per_capita_net_earnings                               int64
```

Variable Transformation and Preprocessing

```
[12]: # Replace all np.NaN with 0
      df = df.replace(to_replace=np.NaN, value=int(0))
```

```
[13]: # Change dtype of all columns except the first (city) to numeric. In order to avoid muddling regression results
      df.iloc[:,1:] = df.iloc[:,1:].apply(pd.to_numeric)
      df.dtypes
```

```
[13]: city                                                  object
      gdp                                                    int64
      private_industries                                     int64
      mining                                                 int64
      construction                                           int64
      manufacturing                                          int64
      retail_trade                                           int64
      finance_total                                          int64
      finance                                                int64
      real_estate                                            int64
      professional_and_business_services                     int64
      educational_services_health_care_and_social_assistance int64
      arts_total                                             int64
      other_services                                         int64
      government_and_government_enterprises                  int64
      personal_income_total                                  int64
      net_earnings_by_place_of_residence                     int64
      personal_current_transfer_receipts                     int64
      income_maintenance_benefits                            int64
      unemployment_insurance_compensation                    int64
      retirement_and_other                                   int64
      dividends_interest_and_rent                            int64
      population                                             int64
      per_capita_personal_income                             int64
      per_capita_net_earnings                                int64
```

Variable Transformation and Preprocessing

```
[14]: # Correct column names
df = df.rename(columns={'total _team _revenue_mlb': 'total_team_revenue_mlb', 'division _title_mlb': 'division_title_mlb'})

[15]: # Convert categorical columns to type category to make our model functions easier to deal with
df[["large_market", "medium_market","small_market","year","city","no_teams","super_bowl_winner", \
    "nfl_division_champion","nfl_playoff_teams","world_series_title","division_title_mlb"]] \
= df[["large_market", "medium_market","small_market","year","city","no_teams","super_bowl_winner", \
    "nfl_division_champion","nfl_playoff_teams","world_series_title","division_title_mlb"]].astype('category')

df.dtypes

[15]: city                                                        category
      gdp                                                            int64
      private_industries                                             int64
      mining                                                         int64
      construction                                                   int64
      manufacturing                                                  int64
      retail_trade                                                   int64
      finance_total                                                  int64
      finance                                                        int64
      real_estate                                                    int64
      professional_and_business_services                             int64
      educational_services_health_care_and_social_assistance         int64
      arts_total                                                     int64
      other_services                                                 int64
      government_and_government_enterprises                          int64
```

Models Constructed

```
[16]: from pyspark.sql import SparkSession
      from pyspark.mllib.regression import LabeledPoint, LinearRegressionWithSGD
      from pyspark.ml.regression import LinearRegression

      spark = SparkSession.builder \
              .master("local") \
              .appName("mllib_classifier") \
              .getOrCreate()
      sc = spark.sparkContext
```

```
[17]: # Going to export our formatted file into csv so we can use a fresh copy for each question and algorithm iteration
      df.to_csv('FormattedData.csv',index=False)
```

```
[18]: from pyspark import SparkConf, SparkContext
      from pyspark.sql import SQLContext

      sqlContext = SQLContext(sc)

      SparkDF = sqlContext.read.format('com.databricks.spark.csv').options(header='true', inferschema='true').load('FormattedData.csv')
      SparkDF.take(1)
```

```
[18]: [Row(city='Atlanta', gdp=209741698, private_industries=191749470, mining=331505, construction=0, manufacturing=0, retail_trade=13217659, finance_total=41922449, fin
      ance=0, real_estate=0, professional_and_business_services=26620247, educational_services_health_care_and_social_assistance=10466645, arts_total=0, other_services=41
      45753, government_and_government_enterprises=17992228, personal_income_total=153691997, net_earnings_by_place_of_residence=117319662, personal_current_transfer_rece
      ipts=12145861, income_maintenance_benefits=1204893, unemployment_insurance_compensation=316462, retirement_and_other=10624506, dividends_interest_and_rent=24226474,
      population=4402455, per_capita_personal_income=34911, per_capita_net_earnings=26649, per_capita_personal_current_transfer_receipts=2759, per_capita_income_maintenan
      ce_benefits=274, per_capita_unemployment_insurance_compensation=72, per_capita_retirement_and_other=2413, per_capita_dividends_interest_and_rent=5503, earnings_by_p
      lace_of_work=132211508, wages_and_salaries=97270037, supplements_to_wages_and_salaries=17607619, employer_contributions_for_employee_pension_and_insurance_funds=112
      74319, employer_contributions_for_government_social_insurance=6333300, proprietors_income=17333852, farm_proprietors_income=188311, nonfarm_proprietors_income=17145
      541, total_employment=2809261, wage_and_salary_employment=2381096, proprietors_employment=428165, farm_proprietors_employment=9902, nonfarm_proprietors_employment=4
      18263, average_earnings_per_job=47063, average_wages_and_salaries=40851, average_nonfarm_proprietors_income=40992, total_mlb_teams=1, total_mlb_team_value=407000.0,
      total_team_revenue_mlb=145500, home_games_mlb=81, reg_season_wins_mlb=88, home_wins_mlb=40, world_series_title=0, division_title_mlb=1, attendance_mlb=2823530, atte
      ndance_per_game_mlb=34858.39506, payroll_mlb=91936166, total_nfl_teams=1, nfl_team_values=338000, nfl_team_revenue=113000, total_nba_team=1, nba_team_value=199000.
      0, nba_team_revenue=76000.0, large_market=1, medium_market=0, small_market=0, no_teams=0, super_bowl_winner=0, nfl_division_champion=0, nfl_playoff_teams=0, nfl_win
      _percentage=0.4375, nfl_wins=7, nfl_home_attendance=425717, nfl_attendance_per_game=53214.625, year=2001, team_relocated=0, team_purchased=0)]
```

Models Constructed

```
[19]: # Check schema
      SparkDF.cache()
      SparkDF.printSchema()

root
 |-- city: string (nullable = true)
 |-- gdp: integer (nullable = true)
 |-- private_industries: integer (nullable = true)
 |-- mining: integer (nullable = true)
 |-- construction: integer (nullable = true)
 |-- manufacturing: integer (nullable = true)
 |-- retail_trade: integer (nullable = true)
 |-- finance_total: integer (nullable = true)
 |-- finance: integer (nullable = true)
 |-- real_estate: integer (nullable = true)
 |-- professional_and_business_services: integer (nullable = true)
 |-- educational_services_health_care_and_social_assistance: integer (nullable = true)
 |-- arts_total: integer (nullable = true)
 |-- other_services: integer (nullable = true)
 |-- government_and_government_enterprises: integer (nullable = true)
 |-- personal_income_total: integer (nullable = true)
 |-- net_earnings_by_place_of_residence: integer (nullable = true)
 |-- personal_current_transfer_receipts: integer (nullable = true)
 |-- income_maintenance_benefits: integer (nullable = true)
 |-- unemployment_insurance_compensation: integer (nullable = true)
```

Models Constructed

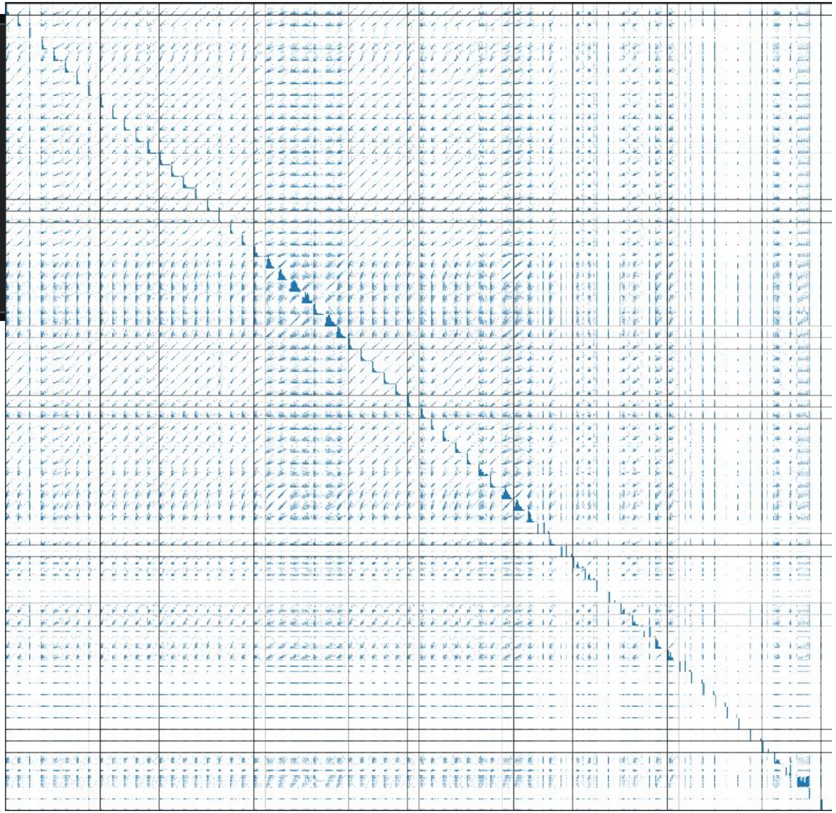
```
[20]: SparkDF.describe().toPandas().transpose()
```

[20]:

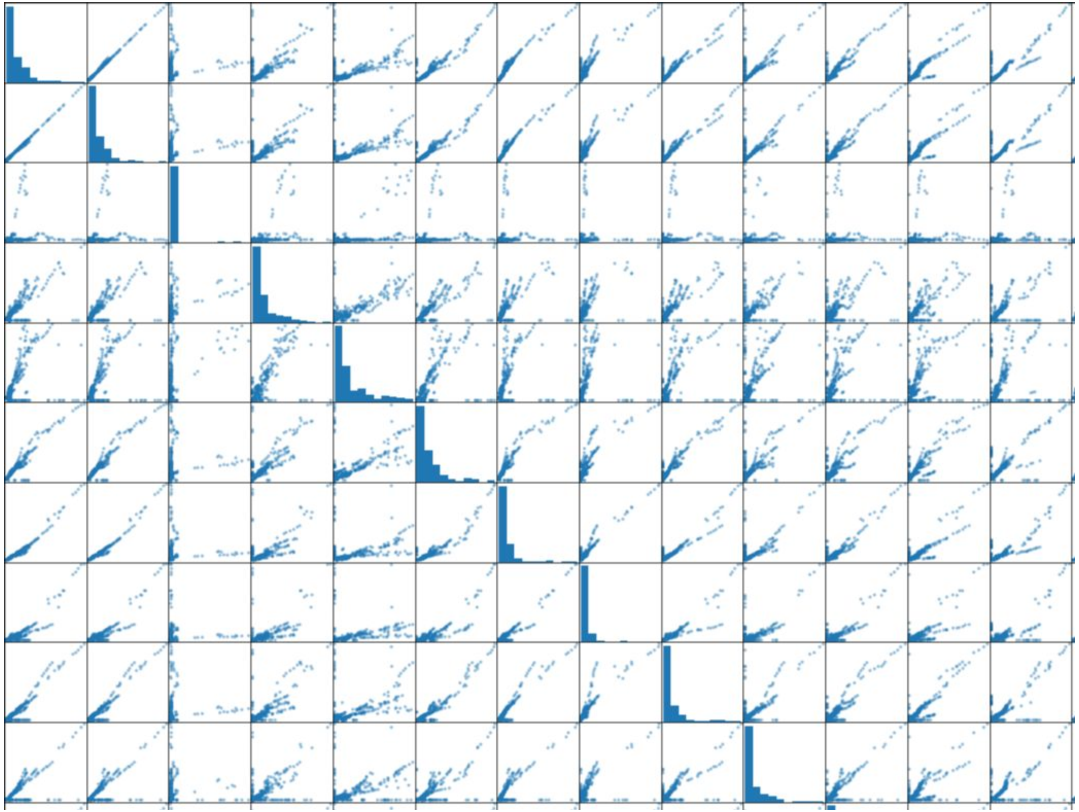
	0	1	2	3	4
summary	count	mean	stddev	min	max
city	432	None	None	Atlanta	Virginia Beach
gdp	432	2.4524781537962964E8	2.9285056394180304E8	10917643	1772319824
private_industries	432	2.1864867616203704E8	2.652309847794597E8	9892020	1611477854
mining	432	1267060.625	4488817.401528948	0	33066973
construction	432	7558340.398148148	9308089.068500169	0	55829143
manufacturing	432	1.9947657002314813E7	2.3264202645681888E7	0	93939097
retail_trade	432	1.2508954724537037E7	1.4182950630311672E7	0	74526288
finance_total	432	5.6424249083333336E7	8.719088511636335E7	1855698	575102095
finance	432	2.090223631712963E7	4.454914393066328E7	0	312484712
real_estate	432	2.942935178935185E7	4.689620853421105E7	0	262617382
professional_and_business_services	432	2.6076587840277776E7	3.942271354493554E7	0	262520531
educational_services_health_care_and_social_assistance	432	1.7408494321759257E7	2.451222136616712E7	0	156446504
arts_total	432	9100840.372685185	1.206079294849609E7	0	74418929
other_services	432	4707902.766203703	6286300.79037286	0	33801689
government_and_government_enterprises	432	2.659913920601852E7	3.0197679797612283E7	1025624	160841970
personal_income_total	432	1.998965341597222E8	2.3849175413430935E8	8653562	1480232981

Models Constructed

```
[ ]: numeric_features = [t[0] for t in SparkDF.dtypes if t[1] == 'int' or t[1] == 'float']
     sampled_data = SparkDF.select(numeric_features).sample(False, 0.8).toPandas()
     axs = pd.plotting.scatter_matrix(sampled_data, figsize=(100, 100))
     n = len(sampled_data.columns)
     for i in range(n):
         v = axs[i, 0]
         v.yaxis.label.set_rotation(0)
         v.yaxis.label.set_ha('right')
         v.set_yticks(())
         h = axs[n-1, i]
         h.xaxis.label.set_rotation(90)
         h.set_xticks(())
```



Models Constructed



Models Constructed

```
[21]: # Explore variable correlation
      import six
      for i in SparkDF.columns:
          if not( isinstance(SparkDF.select(i).take(1)[0][0], six.string_types)):
              print( "GDP for ", i, SparkDF.stat.corr('gdp',i))
```

```
GDP for  gdp 1.0
GDP for  private_industries 0.9990404733420539
GDP for  mining 0.1150729440967743
GDP for  construction 0.7514387855785662
GDP for  manufacturing 0.46107496971927375
GDP for  retail_trade 0.9561629708376779
GDP for  finance_total 0.9762518593797699
GDP for  finance 0.8947362589491431
GDP for  real_estate 0.9471521873365634
GDP for  professional_and_business_services 0.7529513307902886
GDP for  educational_services_health_care_and_social_assistance 0.9377869678321251
GDP for  arts_total 0.8496316631288121
GDP for  other_services 0.8945359056247975
GDP for  government_and_government_enterprises 0.923053544270096
GDP for  personal_income_total 0.9989813064962608
GDP for  net_earnings_by_place_of_residence 0.9984149213329097
GDP for  personal_current_transfer_receipts 0.9774870639814784
GDP for  income_maintenance_benefits 0.965363388732667
GDP for  unemployment_insurance_compensation 0.7315864750439102
GDP for  retirement_and_other 0.9775407353600813
GDP for  dividends_interest_and_rent 0.9891758734953132
GDP for  population 0.9682150952291451
GDP for  per_capita_personal_income 0.5169426852139307
GDP for  per_capita_net_earnings 0.501291222954431
GDP for  per_capita_personal_current_transfer_receipts 0.2761941748396393
GDP for  per_capita_income_maintenance_benefits 0.4031884361561063
GDP for  per_capita_unemployment_insurance_compensation 0.09914192977915737
GDP for  per_capita_retirement_and_other 0.24804122053057684
GDP for  per_capita_dividends_interest_and_rent 0.46759916300949883
```

Models Constructed

```
[58]: # Select columns to keep
      col_names = (list(df.columns)[-31:])
      col_names.extend(['per_capita_personal_income'])

      remove_cols = ['attendance_mlb', 'nfl_home_attendance','attendance_per_game_mlb','home_games_mlb','total_nba_team','total_mlb_teams','total_nfl_teams','reg_season_wins_mlb',
                    'home_wins_mlb','world_series_title','division_title_mlb','super_bowl_winner','nfl_division_champion','nfl_playoff_teams','nfl_win_percentage','nfl_wins',
                    'nfl_attendance_per_game','payroll_mlb']
      col_names = [col for col in col_names if col not in remove_cols]
```

```
[59]: # Transform into vector features
      from pyspark.ml.feature import VectorAssembler
      vectorAssembler = VectorAssembler(inputCols = col_names, outputCol = 'features')
      vSparkDF = vectorAssembler.transform(SparkDF)
      vSparkDF = vSparkDF.select(['features', 'gdp'])
      vSparkDF.show(3)
```

```
+--------------------+---------+
|            features|      gdp|
+--------------------+---------+
|[407000.0,145500....|209741698|
|(14,[9,10,13],[1....| 55307638|
|(14,[9,10,13],[1....| 16435618|
+--------------------+---------+
only showing top 3 rows
```

```
[60]: # Split data into train and test set
      splits = vSparkDF.randomSplit([0.7, 0.3])
      train_df = splits[0]
      test_df = splits[1]
```

Models Constructed

```
[61]: # Linear Regression
      from pyspark.ml.regression import LinearRegression
      lr = LinearRegression(featuresCol = 'features', labelCol='gdp', maxIter=10, regParam=0.3, elasticNetParam=0.8)
      lr_model = lr.fit(train_df)
      print("Coefficients: " + str(lr_model.coefficients))
      print("Intercept: " + str(lr_model.intercept))

      Coefficients: [97.97768857870597,268.24153858837775,13.180338120907132,153.78697575257442,-0.0,683.73012986315,138133408.13347384,-57408676.30342582,-94051875.87989247,-433397.603182999
      2,-7616017.523846555,-122000519.75638849,25315267.21949936,-504.09989097273296]
      Intercept: 15381737250.28748

[63]: trainingSummary = lr_model.summary
      print("RMSE: %f" % trainingSummary.rootMeanSquaredError)
      print("r2: %f" % trainingSummary.r2)
      print("Adjusted r2: %f" % trainingSummary.r2adj)

      RMSE: 124594860.634852
      r2: 0.845292
      Adjusted r2: 0.837875

[64]: train_df.describe().show()

      +-------+--------------------+
      |summary|                 gdp|
      +-------+--------------------+
      |  count|                 307|
      |   mean|2.621812232084690 6E8|
      | stddev| 3.172870083581394E8|
      |    min|            10917643|
      |    max|          1772319824|
      +-------+--------------------+
```

Models Constructed

```
[65]: # analyze coefficients of features
c = lr_model.coefficients
d = {'features': col_names, 'coefficients': c}
features = pd.DataFrame(data=d)
features
```

[65]:

	features	coefficients
0	total_mlb_team_value	9.797769e+01
1	total_team_revenue_mlb	2.682415e+02
2	nfl_team_values	1.318034e+01
3	nfl_team_revenue	1.537870e+02
4	nba_team_value	-0.000000e+00
5	nba_team_revenue	6.837301e+02
6	large_market	1.381334e+08
7	medium_market	-5.740868e+07
8	small_market	-9.405188e+07
9	no_teams	-4.333976e+05
10	year	-7.616018e+06
11	team_relocated	-1.220005e+08
12	team_purchased	2.531527e+07
13	per_capita_personal_income	-5.040999e+02

Models Constructed

```
[66]: lr_predictions = lr_model.transform(test_df)
      lr_predictions.select("prediction","gdp","features").show(5)
      from pyspark.ml.evaluation import RegressionEvaluator
      lr_evaluator = RegressionEvaluator(predictionCol="prediction", \
                       labelCol="gdp",metricName="r2")
      print("R Squared (R2) on test data = %g" % lr_evaluator.evaluate(lr_predictions))

      +--------------------+---------+--------------------+
      |          prediction|      gdp|            features|
      +--------------------+---------+--------------------+
      |1.2359962999925423E8|112796544|(14,[0,1,2,3,7,10...|
      | 1.261586847247982E8|114389181|(14,[0,1,2,3,7,10...|
      |1.0880535902512741E8|120334797|(14,[0,1,2,3,7,10...|
      |1.4053713604518318E8|233226865|(14,[0,1,2,3,7,10...|
      |1.1918326277298927E8|129677300|(14,[0,1,2,3,7,10...|
      +--------------------+---------+--------------------+
      only showing top 5 rows

      R Squared (R2) on test data = 0.801138

[67]: test_result = lr_model.evaluate(test_df)
      lr_rmse = test_result.rootMeanSquaredError
      print("Linear Regression: Root Mean Squared Error (RMSE) on test data = %g" % lr_rmse)

      Linear Regression: Root Mean Squared Error (RMSE) on test data = 9.65013e+07

[68]: print("numIterations: %d" % trainingSummary.totalIterations)
      print("objectiveHistory: %s" % str(trainingSummary.objectiveHistory))
      trainingSummary.residuals.show()

      numIterations: 11
      objectiveHistory: [0.5, 0.4000603758181479, 0.1888294343316928, 0.12766569691331303, 0.1067952248516049, 0.09681861964038191, 0.09023375397947249, 0.085
      92, 0.07793815609225353, 0.07735397530143193]
      +--------------------+
      |           residuals|
      +--------------------+
      | 2.087052418919735E8|
      |-5.23719960187282...|
      | 9.792833400663185E7|
      |-3.71654276974449...|
      | -5.63367761282177E7|
```

Models Constructed

```
[69]: predictions = lr_model.transform(test_df)
      predictions.select("prediction","gdp","features").show()

+--------------------+---------+--------------------+
|          prediction|      gdp|            features|
+--------------------+---------+--------------------+
|1.2359962999925423E8|112796544|(14,[0,1,2,3,7,10...|
| 1.261586847247982E8|114389181|(14,[0,1,2,3,7,10...|
|1.0880535902512741E8|120334797|(14,[0,1,2,3,7,10...|
|1.4053713604518318E8|233226865|(14,[0,1,2,3,7,10...|
|1.1918326277298927E8|129677300|(14,[0,1,2,3,7,10...|
|1.3587489805522537E8|138118159|(14,[0,1,2,3,7,10...|
| 1.582710464039402E8|150163428|(14,[0,1,2,3,7,10...|
|2.3559198463868332E8|360940192|(14,[0,1,2,3,7,10...|
|2.3442978188765144E8|392036945|(14,[0,1,2,3,7,10...|
|   5.4751426684062E8|665296297|(14,[0,1,4,5,6,10...|
| 6.332436414239159E8|739857938|(14,[0,1,4,5,6,10...|
| 6.542940890929108E8|756470973|(14,[0,1,4,5,6,10...|
| 8.110207117820358E8|820353615|(14,[0,1,4,5,6,10...|
|1.0252974920246525E9|912384865|(14,[0,1,4,5,6,10...|
| 1.867524293989296E8|213096390|(14,[0,1,4,5,7,10...|
| 1.328072569291668E8| 90850236|(14,[0,1,4,5,8,10...|
|1.0335512075218582E8| 70579101|(14,[0,1,4,5,8,10...|
|1.0651210590215683E8| 77618804|(14,[0,1,4,5,8,10...|
| 3.559821363691788E8|240353006|(14,[0,2,3,4,5,6,...|
| 1.069674357549324E8| 84628076|(14,[0,2,3,7,10,1...|
+--------------------+---------+--------------------+
only showing top 20 rows
```

Models Constructed

```
[70]: # Decision Tree Regression
from pyspark.ml.regression import DecisionTreeRegressor
dt = DecisionTreeRegressor(featuresCol ='features', labelCol = 'gdp')
dt_model = dt.fit(train_df)
dt_predictions = dt_model.transform(test_df)
dt_evaluator = RegressionEvaluator(
    labelCol="gdp", predictionCol="prediction", metricName="rmse")
dt_rmse = dt_evaluator.evaluate(dt_predictions)
print("Decision Tree: Root Mean Squared Error (RMSE) on test data = %g" % dt_rmse)
```

```
Decision Tree: Root Mean Squared Error (RMSE) on test data = 6.90014e+07
```

```
[71]: # Feature Importance
a = dt_model.featureImportances.toArray()
d = {'features': col_names, 'importance': a}
fi = pd.DataFrame(data=d)
fi.sort_values(by='importance', ascending=False)
```

[71]:

	features	importance
1	total_team_revenue_mlb	0.645348
0	total_mlb_team_value	0.133827
3	nfl_team_revenue	0.047209
5	nba_team_revenue	0.040221
6	large_market	0.037229
2	nfl_team_values	0.024212
10	year	0.021677
4	nba_team_value	0.019270
7	medium_market	0.013007
13	per_capita_personal_income	0.012476
11	team_relocated	0.002765
12	team_purchased	0.001814
8	small_market	0.000946
9	no_teams	0.000000

Models Constructed

```
[72]: # Gradient-boosted Tree Regression
from pyspark.ml.regression import GBTRegressor
gbt = GBTRegressor(featuresCol = 'features', labelCol = 'gdp', maxIter=10)
gbt_model = gbt.fit(train_df)
gbt_predictions = gbt_model.transform(test_df)
gbt_predictions.select('prediction', 'gdp', 'features').show(5)

+-------------------+---------+--------------------+
|         prediction|      gdp|            features|
+-------------------+---------+--------------------+
|1.0580206106817895E8|112796544|(14,[0,1,2,3,7,10...|
| 9.958060905157915E7|114389181|(14,[0,1,2,3,7,10...|
| 9.283220574138588E7|120334797|(14,[0,1,2,3,7,10...|
|2.4881729941875306E8|233226865|(14,[0,1,2,3,7,10...|
| 1.750199646928488E8|129677300|(14,[0,1,2,3,7,10...|
+-------------------+---------+--------------------+
only showing top 5 rows

[73]: gbt_evaluator = RegressionEvaluator(
    labelCol="gdp", predictionCol="prediction", metricName="rmse")
gbt_rmse = gbt_evaluator.evaluate(gbt_predictions)
print("GBT: Root Mean Squared Error (RMSE) on test data = %g" % gbt_rmse)

GBT: Root Mean Squared Error (RMSE) on test data = 5.87912e+07
```

Models Constructed

```python
def compareModels(col_names):
    from pyspark.ml.feature import VectorAssembler
    vectorAssembler = VectorAssembler(inputCols = col_names, outputCol = 'features')
    vSparkDF = vectorAssembler.transform(SparkDF)
    vSparkDF = vSparkDF.select(['features', 'gdp'])
    # vSparkDF.show(3)

    splits = vSparkDF.randomSplit([0.7, 0.3])
    train_df = splits[0]
    test_df = splits[1]

    # Linear Regression
    from pyspark.ml.regression import LinearRegression
    lr = LinearRegression(featuresCol = 'features', labelCol='gdp', maxIter=10, regParam=0.3, elasticNetParam=0.8)
    lr_model = lr.fit(train_df)
    # print("Coefficients: " + str(lr_model.coefficients))
    # print("Intercept: " + str(lr_model.intercept))

    # trainingSummary = lr_model.summary
    # print("RMSE: %f" % trainingSummary.rootMeanSquaredError)
    # print("r2: %f" % trainingSummary.r2)

    lr_predictions = lr_model.transform(test_df)
    # lr_predictions.select("prediction","gdp","features").show(5)
    from pyspark.ml.evaluation import RegressionEvaluator
    lr_evaluator = RegressionEvaluator(predictionCol="prediction", \
                     labelCol="gdp",metricName="r2")
    print("Linear Regression: R Squared (R2) on test data = %g" % lr_evaluator.evaluate(lr_predictions))

    test_result = lr_model.evaluate(test_df)
    lr_rmse = test_result.rootMeanSquaredError
    print("Linear Regression: Root Mean Squared Error (RMSE) on test data = %g" % lr_rmse)

    from pyspark.ml.regression import DecisionTreeRegressor
```

```python
def featuresCoeff(col_names):
    from pyspark.ml.feature import VectorAssembler
    vectorAssembler = VectorAssembler(inputCols = col_names, outputCol = 'features')
    vSparkDF = vectorAssembler.transform(SparkDF)
    vSparkDF = vSparkDF.select(['features', 'gdp'])
    # vSparkDF.show(3)

    splits = vSparkDF.randomSplit([0.7, 0.3])
    train_df = splits[0]
    test_df = splits[1]

    # Linear Regression
    from pyspark.ml.regression import LinearRegression
    lr = LinearRegression(featuresCol = 'features', labelCol='gdp', maxIter=10, regParam=0.3, elasticNetParam=0.8)
    lr_model = lr.fit(train_df)
    # print("Coefficients: " + str(lr_model.coefficients))
    # print("Intercept: " + str(lr_model.intercept))

    # trainingSummary = lr_model.summary
    # print("RMSE: %f" % trainingSummary.rootMeanSquaredError)
    # print("r2: %f" % trainingSummary.r2)

    c = lr_model.coefficients
    d = {'features': col_names, 'coefficients': c}
    features = pd.DataFrame(data=d)

    return features
```

Models Constructed

```
[76]: col_names2 = (list(df.columns)[-31:])
      col_names2.extend(['per_capita_personal_income'])

      remove_cols = ['attendance_mlb', 'nfl_home_attendance', 'nfl_wins','total_mlb_teams','home_wins_mlb',\
                     'reg_season_wins_mlb','nfl_win_percentage','nfl_home_attendance','home_games_mlb']
      col_names2 = [col for col in col_names2 if col not in remove_cols]

[77]: compareModels(col_names2)

      Linear Regression: R Squared (R2) on test data = 0.823725
      Linear Regression: Root Mean Squared Error (RMSE) on test data = 1.42175e+08
      Decision Tree: R Squared (R2) on test data = 0.959344
      Decision Tree: Root Mean Squared Error (RMSE) on test data = 6.82793e+07
      GBT: R Squared (R2) on test data = 0.966251
      GBT: Root Mean Squared Error (RMSE) on test data = 6.22097e+07

[78]: col_names3 = (list(df.columns)[-31:])
      col_names3.extend(['per_capita_personal_income'])

      remove_cols = ['attendance_mlb', 'nfl_home_attendance', 'nfl_wins','total_mlb_teams','home_wins_mlb', \
                     'reg_season_wins_mlb','nfl_win_percentage','nfl_home_attendance', \
                     'home_games_mlb', 'large_market','medium_market','small_market']
      col_names3 = [col for col in col_names3 if col not in remove_cols]

[79]: compareModels(col_names3)

      Linear Regression: R Squared (R2) on test data = 0.806799
      Linear Regression: Root Mean Squared Error (RMSE) on test data = 1.04353e+08
      Decision Tree: R Squared (R2) on test data = 0.919527
      Decision Tree: Root Mean Squared Error (RMSE) on test data = 6.73484e+07
      GBT: R Squared (R2) on test data = 0.930767
      GBT: Root Mean Squared Error (RMSE) on test data = 6.24679e+07

[80]: featuresCoeff(col_names3)

[80]:
```

	features	coefficients
0	total_mlb_team_value	9.917152e+01
1	total_team_revenue_mlb	1.566667e+02
2	world_series_title	-3.993886e+07
3	division_title_mlb	-0.000000e+00
4	attendance_per_game_mlb	[illegible]

Model Performance

Once our first model, Sports Teams effect on GDP, was developed the other two fell into place rather easy

- Initially gave an amazing adjusted R-squared of 0.970195 as expected
- After removing this nuclear variable we achieved a much more realistic, but sufficient results with an adjusted R-squared of 0.837875.
- Noted the positive coef of “large_market” and “no_team”

Model Performance

- Our second model focused on the impact market size played
  - This returned only one positive coefficient being “Large_Market” at 3.64e+08
    - “Small_Market” at -1.72e+08
    - “Medium_Market” at -1.87e+07
    - “No_Team” at -1.92e+08
- Our last model looked at the ownership changed or initialization of a team.
  - Focusing on the binary variables “team_relocated” and “team_ownership”
  - Adjusted R-squared of 0.806118
  - root mean squared error of 1.04525e+08.

Conclusion and Future Research

- Ultimately we concluded a city having a sports team can impact GDP, but only when market size is large such as New York City or Los Angeles.

- We were also able to conclude that a team relocating or joining a league during its first year actually had a negative impact on GDP.

- The change in ownership also had a slight negative impact on GDP however it was much less than the initialization of a team or relocation.

Conclusion and Future Research

- Does it make economic sense for a city to recruit a professional sports team or an additional sports team to their market; is the return on investment there?
- Does it have an impact on things that aren’t as easy to measure such as happiness of the residents and social opportunities?
- Researching the localized effect of sports teams on restaurants and bars.
- What impact does college sports have on their localities?