Project: Investigate Forest Forest Coverage and its relationship to Those Affected by Droughts, Extreme Temperatures and Flooods

Table of Contents

Introduction

In this project I am exploring whether or not countries with a higher percentage of forest coverage fare better during certain natural disasters, specifically floods, droughts and extreme temperatures. Data used in this project will be pulled from Gapminder.org, from their forestry datasets and disaster datasets under their environment category. I have also pulled a dataset on total population so that comparison between affected can be drawn on per/capita in each country.

Questions to answer:

1. Do countries with a higher percentage of forest coverage have a lower number of affected from floods, droughts and extreme temperatures?
2. Which natural disaster type affects the most countries on a per/capita basis in countries with the lowest percentage of forest coverage?
3. Over time, do countries where forest coverage increases have decreaseing number of those affected by disasters?

The following datasets from Gapminder.org were used in compiling the data used in this report:

- Forest coverage(%)
- Total Population
- Drought affected annual number
- Extreme temperature affected annual number
- Flood affected annual number

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Data Wrangling

Before analyzing data I will examine each data set to understand and evaluate what data is available to me, what is missing and develop a plan for preparing data for for exploration. I will determine where data is available across all datasets for comparison.

To asses data I will look at each dataset's date range in years (columns) and countries in which data is available (rows). I can then establish where data overlaps in all datasets for further exploration.

I will also asses null values in each data set and look to see if there is a pattern for missing values or if gaps in data share any similarities. I will then address missing values by either inputting estimated values based on surrounding data or drop the row or columns from the series.

Lastly, I will check for duplicated data in each data set, and if necessary, drop it.

The 5 datasets I plan to use for this project are:

- Forest coverage - A percentagre of a countries total land area covered with forest during the given year
- Total Population - The total population for a country in a given year
- Drought Affected - Total number of people affected, injured or killed in drought during the given year
- Extreme Temperature Affected - Total number of people affected, injured or killed in extreme temperatures during the given year
- Flood Affected - Total number of people afftected, injured or killed in flood during the given year

I have identified forest coverage and total population as my independent variables and drought affected, extreme temperature affected and flood affected as my dependent variables

In [3]:
```
forest_coverage = pd.read_csv('forest_coverage_percent.csv')
total_pop = pd.read_csv('population_total.csv')

drought = pd.read_csv('drought_affected_annual_number.csv')
ext_temp = pd.read_csv('extreme_temperature_affected_annual_number.csv')
flood = pd.read_csv('flood_affected_annual_number.csv')
```

Forest Coverage

In [4]:
```
forest_coverage.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 192 entries, 0 to 191
Data columns (total 27 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   country  192 non-null    object 
 1   1990     162 non-null    float64
 2   1991     165 non-null    float64
 3   1992     184 non-null    float64
 4   1993     188 non-null    float64
 5   1994     188 non-null    float64
 6   1995     188 non-null    float64
 7   1996     188 non-null    float64
 8   1997     188 non-null    float64
 9   1998     188 non-null    float64
 10  1999     188 non-null    float64
 11  2000     190 non-null    float64
 12  2001     190 non-null    float64
 13  2002     190 non-null    float64
 14  2003     190 non-null    float64
 15  2004     190 non-null    float64
 16  2005     190 non-null    float64
 17  2006     192 non-null    float64
 18  2007     192 non-null    float64
 19  2008     192 non-null    float64
 20  2009     192 non-null    float64
 21  2010     192 non-null    float64
 22  2011     191 non-null    float64
 23  2012     191 non-null    float64
 24  2013     191 non-null    float64
 25  2014     191 non-null    float64
 26  2015     191 non-null    float64
dtypes: float64(26), object(1)
memory usage: 40.6+ KB
```

In [5]:
```
forest_coverage.duplicated().sum()
```

Out[5]:
```
0
```

In [6]:
```
forest_coverage.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 192 entries, 0 to 191
Data columns (total 27 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   country  192 non-null    object 
 1   1990     162 non-null    float64
 2   1991     165 non-null    float64
 3   1992     184 non-null    float64
 4   1993     188 non-null    float64
 5   1994     188 non-null    float64
 6   1995     188 non-null    float64
 7   1996     188 non-null    float64
 8   1997     188 non-null    float64
 9   1998     188 non-null    float64
 10  1999     188 non-null    float64
 11  2000     190 non-null    float64
 12  2001     190 non-null    float64
 13  2002     190 non-null    float64
 14  2003     190 non-null    float64
 15  2004     190 non-null    float64
 16  2005     190 non-null    float64
 17  2006     192 non-null    float64
 18  2007     192 non-null    float64
 19  2008     192 non-null    float64
 20  2009     192 non-null    float64
 21  2010     192 non-null    float64
 22  2011     191 non-null    float64
 23  2012     191 non-null    float64
 24  2013     191 non-null    float64
 25  2014     191 non-null    float64
 26  2015     191 non-null    float64
dtypes: float64(26), object(1)
memory usage: 40.6+ KB
```

In [7]:
```
forest_coverage.isnull().sum()
```

Out[7]:
```
country     0
1990       30
1991       27
1992        8
1993        4
1994        4
1995        4
1996        4
1997        4
1998        4
1999        4
2000        2
2001        2
2002        2
2003        2
2004        2
2005        2
2006        0
2007        0
2008        0
2009        0
2010        0
2011        1
2012        1
2013        1
2014        1
2015        1
dtype: int64
```

In [8]:
```
missing_values = forest_coverage[forest_coverage.isna().any(axis=1)]
missing_values
```

Out[8]:

	country	1990	1991	1992	1993	1994	1995	1996	1997	1998	...	200
7	Armenia	NaN	NaN	0.1180	0.1180	0.1170	0.1170	0.1170	0.1170	0.1170	...	0.117
10	Azerbaijan	NaN	NaN	0.1030	0.1030	0.1030	0.1040	0.1040	0.1040	0.1040	...	0.109
15	Belarus	NaN	NaN	0.3880	0.3910	0.3930	0.3960	0.3980	0.4010	0.4030	...	0.417
16	Belgium	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	0.223
21	Bosnia and Herzegovina	NaN	NaN	0.4310	0.4300	0.4300	0.4290	0.4290	0.4280	0.4280	...	0.427
42	Croatia	NaN	NaN	0.3320	0.3330	0.3330	0.3340	0.3340	0.3350	0.3360	...	0.341
45	Czech Republic	NaN	NaN	NaN	0.3400	0.3410	0.3410	0.3410	0.3410	0.3410	...	0.343
54	Eritrea	NaN	NaN	NaN	0.1590	0.1590	0.1580	0.1580	0.1570	0.1570	...	0.153
55	Estonia	NaN	NaN	0.5220	0.5230	0.5240	0.5250	0.5260	0.5260	0.5270	...	0.530
57	Ethiopia	NaN	NaN	NaN	0.1470	0.1460	0.1440	0.1430	0.1410	0.1400	...	0.129
63	Georgia	NaN	NaN	0.3960	0.3960	0.3960	0.3970	0.3970	0.3970	0.3970	...	0.400
86	Kazakhstan	NaN	NaN	0.0126	0.0126	0.0126	0.0126	0.0125	0.0125	0.0125	...	0.012
90	Kyrgyz Republic	NaN	NaN	0.0438	0.0440	0.0441	0.0442	0.0443	0.0444	0.0445	...	0.043
92	Latvia	NaN	NaN	0.5120	0.5130	0.5150	0.5160	0.5170	0.5180	0.5190	...	0.532
98	Lithuania	NaN	NaN	0.3130	0.3140	0.3150	0.3160	0.3180	0.3190	0.3200	...	0.340
99	Luxembourg	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	0.335
106	Marshall Islands	NaN	0.702	0.7020	0.7020	0.7020	0.7020	0.7020	0.7020	0.7020	...	0.702
110	Micronesia, Fed. Sts.	NaN	0.909	0.9090	0.9100	0.9100	0.9100	0.9110	0.9110	0.9110	...	0.915
111	Moldova	NaN	NaN	0.0974	0.0973	0.0974	0.0976	0.0978	0.0980	0.0981	...	0.112
113	Montenegro	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	0.495
126	North Macedonia	NaN	NaN	0.3620	0.3640	0.3660	0.3680	0.3700	0.3710	0.3730	...	0.385
130	Palau	NaN	0.833	0.8360	0.8390	0.8420	0.8450	0.8480	0.8510	0.8540	...	0.876
141	Russia	NaN	NaN	0.4940	0.4940	0.4940	0.4940	0.4940	0.4940	0.4940	...	0.495
148	Serbia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	0.289
152	Slovak Republic	NaN	NaN	NaN	0.4000	0.4000	0.4000	0.4000	0.3990	0.3990	...	0.402
153	Slovenia	NaN	NaN	0.5940	0.5970	0.5990	0.6010	0.6030	0.6050	0.6080	...	0.618
163	Sudan	0.129	0.129	0.1280	0.1270	0.1260	0.1260	0.1250	0.1240	0.1230	...	0.118
168	Tajikistan	NaN	NaN	0.0292	0.0292	0.0292	0.0292	0.0292	0.0293	0.0293	...	0.029
177	Turkmenistan	NaN	NaN	0.0878	0.0878	0.0878	0.0878	0.0878	0.0878	0.0878	...	0.087
180	Ukraine	NaN	NaN	0.1610	0.1610	0.1620	0.1620	0.1630	0.1630	0.1630	...	0.165
185	Uzbekistan	NaN	NaN	0.0724	0.0728	0.0731	0.0735	0.0739	0.0743	0.0747	...	0.077

31 rows × 27 columns

Forest coverage data covers years 1990 to 2015, which corresponds with the date ranges for Biomass coverage. 1990 and 1991 are missing quite a few values across several countries. There aslo appear to be 5 countries with more than 3 consecutive null values outside of the 1990-1991 nulls.

There is no duplicated data.

There are 192 countries covered in this dataset.

Population Data

In [9]:
```
total_pop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Columns: 302 entries, country to 2100
dtypes: int64(301), object(1)
memory usage: 460.2+ KB
```

In [10]:
```
total_pop.head(15)
```

Out[10]:

	country	1800	1801	1802	1803	1804	1805	1806	1
0	Afghanistan	3280000	3280000	3280000	3280000	3280000	3280000	3280000	3280
1	Albania	400000	402000	404000	405000	407000	409000	411000	413
2	Algeria	2500000	2510000	2520000	2530000	2540000	2550000	2560000	2560
3	Andorra	2650	2650	2650	2650	2650	2650	2650	2
4	Angola	1570000	1570000	1570000	1570000	1570000	1570000	1570000	1570
5	Antigua and Barbuda	37000	37000	37000	37000	37000	37000	37000	37
6	Argentina	534000	520000	506000	492000	479000	466000	453000	441
7	Armenia	413000	413000	413000	413000	413000	413000	413000	413
8	Australia	200000	205000	211000	216000	222000	227000	233000	239
9	Austria	3000000	3020000	3040000	3050000	3070000	3090000	3110000	3120
10	Azerbaijan	880000	880000	880000	880000	880000	880000	880000	880
11	Bahamas	27400	27400	27400	27400	27400	27400	27400	27
12	Bahrain	64500	64500	64500	64500	64500	64500	64500	64
13	Bangladesh	19200000	19200000	19300000	19300000	19300000	19400000	19400000	19500
14	Barbados	81700	81700	81700	81700	81700	81700	81700	81

15 rows × 302 columns

In [11]:
```
total_pop.isnull().sum().sum()
```

Out[11]: 0

In [12]:
```
total_pop.duplicated().sum()
```

Out[12]: 0

Looking at total population there is a siginificant amount of data going back to 1800 covering 195 countries. There are also values for predicted population until the year 2100.

There is no duplicated data.

There appear to be no null values in the data.

Drought Data Affected

In [13]:
```
drought.head(10)
```

Out[13]:

	country	1970	1971	1972	1973	1974	1975	1976	1977	1978	...	1999	2000	2001
0	Afghanistan	0	0	0	0	0	0	0	0	0	...	0	2580000	0
1	Albania	0	0	0	0	0	0	0	0	0	...	0	0	0
2	Algeria	0	0	0	0	0	0	0	0	0	...	0	0	0
3	Angola	0	0	0	0	0	0	0	0	0	...	0	0	58
4	Antigua and Barbuda	0	0	0	0	0	0	0	0	0	...	0	0	0
5	Argentina	0	0	0	0	0	0	0	0	0	...	0	0	0
6	Armenia	0	0	0	0	0	0	0	0	0	...	0	297000	0
7	Australia	0	0	0	0	0	0	0	0	0	...	0	0	0
8	Azerbaijan	0	0	0	0	0	0	0	0	0	...	0	0	0
9	Bangladesh	0	0	0	0	0	0	0	0	0	...	0	0	0

10 rows × 40 columns

In [14]:
```
drought.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 123 entries, 0 to 122
Data columns (total 40 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   country  123 non-null    object
 1   1970     123 non-null    int64 
 2   1971     123 non-null    int64 
 3   1972     123 non-null    int64 
 4   1973     123 non-null    int64 
 5   1974     123 non-null    int64 
 6   1975     123 non-null    int64 
 7   1976     123 non-null    int64 
 8   1977     123 non-null    int64 
 9   1978     123 non-null    int64 
 10  1979     123 non-null    int64 
 11  1980     123 non-null    int64 
 12  1981     123 non-null    int64 
 13  1982     123 non-null    int64 
 14  1983     123 non-null    int64 
 15  1984     123 non-null    int64 
 16  1985     123 non-null    int64 
 17  1986     123 non-null    int64 
 18  1987     123 non-null    int64 
 19  1988     123 non-null    int64 
 20  1989     123 non-null    int64 
 21  1990     123 non-null    int64 
 22  1991     123 non-null    int64 
 23  1992     123 non-null    int64 
 24  1993     123 non-null    int64 
 25  1994     123 non-null    int64 
 26  1995     123 non-null    int64 
 27  1996     123 non-null    int64 
 28  1997     123 non-null    int64 
 29  1998     123 non-null    int64 
 30  1999     123 non-null    int64 
 31  2000     123 non-null    int64 
 32  2001     123 non-null    int64 
 33  2002     123 non-null    int64 
 34  2003     123 non-null    int64 
 35  2004     123 non-null    int64 
 36  2005     123 non-null    int64 
 37  2006     123 non-null    int64 
 38  2007     123 non-null    int64 
 39  2008     123 non-null    int64 
dtypes: int64(39), object(1)
memory usage: 38.6+ KB
```

In [15]:
```
drought.isnull().sum().sum()
```

Out[15]:
```
0
```

In [16]:
```
drought.duplicated().sum()
```
Out[16]:
```
0
```

Drought data covers years 1970 to 2008 and only 123 countries. There are a significant number of zero values in the data, but no null values.

Since natural disasters happen infrequently and not year to year and impact some countries more than others it would be within expectations that many years for countries had zero affected for particular natural disasters and that these zero values are accurate.

There is no duplicated data.

Extreme Temperature Affected

```
In [17]: ext_temp.head(30)
```

Out[17]:

	country	1971	1972	1973	1974	1975	1976	1977	1978	1979	...	1999	2000	200
0	Afghanistan	0	0	0	NaN	0	NaN	0	0	0	...	0	0	20000
1	Albania	0	0	0	NaN	0	NaN	0	0	0	...	0	0	
2	Algeria	0	0	0	NaN	0	NaN	0	0	0	...	0	0	
3	Argentina	0	100	0	NaN	0	NaN	0	0	0	...	0	315	301
4	Australia	0	0	0	NaN	0	NaN	0	0	0	...	0	0	
5	Austria	0	0	0	NaN	0	NaN	0	0	0	...	0	0	
6	Bangladesh	0	0	0	NaN	0	NaN	0	0	0	...	0	49	201
7	Belarus	0	0	0	NaN	0	NaN	0	0	0	...	0	0	
8	Belgium	0	0	0	NaN	0	NaN	0	0	0	...	0	0	
9	Belize	0	0	0	NaN	0	NaN	0	0	0	...	0	0	
10	Bolivia	0	0	0	NaN	0	NaN	0	0	0	...	0	25300	1
11	Bosnia and Herzegovina	0	0	0	NaN	0	NaN	0	0	0	...	0	0	
12	Brazil	0	0	0	NaN	726	NaN	0	0	0	...	0	7	
13	Bulgaria	0	0	0	NaN	0	NaN	0	0	0	...	0	7	
14	Canada	0	0	0	NaN	0	NaN	0	0	0	...	0	0	
15	Chile	0	0	0	NaN	0	NaN	0	0	0	...	0	0	
16	China	0	0	0	NaN	0	NaN	0	0	0	...	0	0	
17	Croatia	0	0	0	NaN	0	NaN	0	0	0	...	0	240	
18	Cyprus	0	0	0	NaN	0	NaN	0	0	0	...	0	405	
19	Czech Republic	0	0	0	NaN	0	NaN	0	0	0	...	0	0	
20	Egypt	0	0	0	NaN	0	NaN	0	0	0	...	0	108	
21	El Salvador	0	0	0	NaN	0	NaN	0	0	0	...	0	0	
22	Estonia	0	0	0	NaN	0	NaN	0	0	0	...	0	0	
23	France	0	0	0	NaN	0	NaN	0	0	0	...	0	0	
24	Germany	0	0	0	NaN	0	NaN	0	0	0	...	0	0	
25	Greece	0	0	0	NaN	0	NaN	0	0	0	...	0	203	
26	Guatemala	0	0	0	NaN	0	NaN	0	0	0	...	0	0	185
27	Hungary	0	0	0	NaN	0	NaN	0	0	0	...	0	0	8
28	India	0	0	261	NaN	0	NaN	0	150	400	...	140	282	17
29	Israel	0	0	0	NaN	0	NaN	0	0	0	...	0	0	

30 rows × 39 columns

In [18]:
```
ext_temp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 68 entries, 0 to 67
Data columns (total 39 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   country  68 non-null     object 
 1   1971     68 non-null     int64  
 2   1972     68 non-null     int64  
 3   1973     68 non-null     int64  
 4   1974     0 non-null      float64
 5   1975     68 non-null     int64  
 6   1976     0 non-null      float64
 7   1977     68 non-null     int64  
 8   1978     68 non-null     int64  
 9   1979     68 non-null     int64  
 10  1980     68 non-null     int64  
 11  1981     68 non-null     int64  
 12  1982     68 non-null     int64  
 13  1983     68 non-null     int64  
 14  1984     68 non-null     int64  
 15  1985     68 non-null     int64  
 16  1986     68 non-null     int64  
 17  1987     68 non-null     int64  
 18  1988     68 non-null     int64  
 19  1989     68 non-null     int64  
 20  1990     68 non-null     int64  
 21  1991     68 non-null     int64  
 22  1992     68 non-null     int64  
 23  1993     68 non-null     int64  
 24  1994     68 non-null     int64  
 25  1995     68 non-null     int64  
 26  1996     68 non-null     int64  
 27  1997     68 non-null     int64  
 28  1998     68 non-null     int64  
 29  1999     68 non-null     int64  
 30  2000     68 non-null     int64  
 31  2001     68 non-null     int64  
 32  2002     68 non-null     int64  
 33  2003     68 non-null     int64  
 34  2004     68 non-null     int64  
 35  2005     68 non-null     int64  
 36  2006     68 non-null     int64  
 37  2007     68 non-null     int64  
 38  2008     68 non-null     int64  
dtypes: float64(2), int64(36), object(1)
memory usage: 20.8+ KB
```

In [19]:
```
ext_temp.duplicated().sum()
```

Out[19]:
```
0
```

Extreme Temperature data covers years 1971 to 2008, and only 68 countries. There are no no-null values for years 1974 and 1976, but all other years have recorded data for all countries in the dataset.

There is no duplicated data.

Like the drought data, there is a significant amount of zero values.

Flood Affected

In [20]:
```
flood.head(15)
```

Out[20]:

	country	1970	1971	1972	1973	1974	1975	1976	1977	1978	...
0	Afghanistan	0	0	250000	0	0	0	80100	0	272000	...
1	Albania	0	0	0	0	0	0	0	0	0	...
2	Algeria	0	0	0	146000	20000	0	0	0	0	...
3	Angola	0	0	0	0	0	0	0	0	0	...
4	Argentina	36	0	0	0	0	0	0	2500	1600	...
5	Armenia	0	0	0	0	0	0	0	0	0	...
6	Australia	0	27	0	12000	14	0	10000	0	7	...
7	Austria	0	0	0	0	0	0	0	0	0	...
8	Azerbaijan	0	0	0	0	0	0	0	0	0	...
9	Bahamas	0	0	0	0	0	0	0	0	0	...
10	Bangladesh	10000000	0	50	427	38000000	0	4000000	214000	400000	...
11	Barbados	213	0	0	0	0	0	0	0	0	...
12	Belarus	0	0	0	0	0	0	0	0	0	...
13	Belgium	0	600	0	0	0	0	0	0	0	...
14	Belize	0	0	0	0	0	0	0	0	0	...

15 rows × 40 columns

In [21]:
```
flood.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 166 entries, 0 to 165
Data columns (total 40 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   country  166 non-null    object
 1   1970     166 non-null    int64 
 2   1971     166 non-null    int64 
 3   1972     166 non-null    int64 
 4   1973     166 non-null    int64 
 5   1974     166 non-null    int64 
 6   1975     166 non-null    int64 
 7   1976     166 non-null    int64 
 8   1977     166 non-null    int64 
 9   1978     166 non-null    int64 
 10  1979     166 non-null    int64 
 11  1980     166 non-null    int64 
 12  1981     166 non-null    int64 
 13  1982     166 non-null    int64 
 14  1983     166 non-null    int64 
 15  1984     166 non-null    int64 
 16  1985     166 non-null    int64 
 17  1986     166 non-null    int64 
 18  1987     166 non-null    int64 
 19  1988     166 non-null    int64 
 20  1989     166 non-null    int64 
 21  1990     166 non-null    int64 
 22  1991     166 non-null    int64 
 23  1992     166 non-null    int64 
 24  1993     166 non-null    int64 
 25  1994     166 non-null    int64 
 26  1995     166 non-null    int64 
 27  1996     166 non-null    int64 
 28  1997     166 non-null    int64 
 29  1998     166 non-null    int64 
 30  1999     166 non-null    int64 
 31  2000     166 non-null    int64 
 32  2001     166 non-null    int64 
 33  2002     166 non-null    int64 
 34  2003     166 non-null    int64 
 35  2004     166 non-null    int64 
 36  2005     166 non-null    int64 
 37  2006     166 non-null    int64 
 38  2007     166 non-null    int64 
 39  2008     166 non-null    int64 
dtypes: int64(39), object(1)
memory usage: 52.0+ KB
```

In [22]:
```
flood.isnull().sum().sum()
```

Out[22]:
```
0
```

In [23]:
```
ext_temp.duplicated().sum()
```
Out[23]:
```
0
```

Flood data also covers years 1970 to 2008, and covers 166 countries. There are no null values in the data, though like the datasets for droughts and extreme temperatures, there are a significant number of zero values.

There is no duplicated data

Data Wrangling Assesment

The 5 datasets are all arranged by country and year which will make analysis simple once data has been cleaned.

Date ranges vary from each dataset and ranges not available in all tables will need to be dropped. The independent variable with a limiting range is forest coverage, which only has data available back to 1990. All three dependent variables have data as recent as 2008.

Therefore, the years for exploration will be 1990 to 2008. All other columns will be dropped.

Another limiting factor across all dataset are the countries in which data is available. Of the two independent variables total population has data for all 195 countries and forest coverage has data for 192. Since these 2 data sets need to match for comparison across the dependent variable data sets I will drop countries that are not available in both.

Drought affected contains data from 123 countries, extreme temperatures contains data from 68 countries, and flood affected contains data from 166 countries. Any countries in these datasets that are not represented in the forest coverage dataset will be dropped. Beyond that, since these data sets will be examined seperately against the independent variables, the existence of countries in one natural disaster dataset and not another will not impact the conclusion of analysis of these variables.

However, to determine the impact of forest coverage across all 3 natural disasters in total, I will create a seperate dataset of total natural disaster affected that will only include countries in which data is available across all datasets.

Finally, null data is present in several datasets. Dropping date ranges from all data sets as mentioned above will remove any null values from our dependent variables data sets. Since total population has zero null values, the only remaining nulls wil be in forest coverage. Four countries have large gaps in data, and will be dropped. The years 1991 and 1992 in the remaining rows can be filled by backfilling data from following years in each country.

Data Cleaning

Forest Coverage Cleaning

Removing years outside of the establish range for this project will remove several null values from consideration.

In [24]:
```
forest_coverage.drop(forest_coverage.columns.to_series()["2009":], axis=1, inplace=True)
```

Next I'll find and remove countries with large gaps in data, which I'll set as any country missing 4 or more consecutives years of data.

In [25]:
```
missing_values = forest_coverage[forest_coverage.isna().any(axis=1)]
missing_values.head(35)
```

Out[25]:

	country	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	
7	Armenia	NaN	NaN	0.1180	0.1180	0.1170	0.1170	0.1170	0.1170	0.1170	0.1170	0
10	Azerbaijan	NaN	NaN	0.1030	0.1030	0.1030	0.1040	0.1040	0.1040	0.1040	0.1050	0
15	Belarus	NaN	NaN	0.3880	0.3910	0.3930	0.3960	0.3980	0.4010	0.4030	0.4050	0
16	Belgium	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0
21	Bosnia and Herzegovina	NaN	NaN	0.4310	0.4300	0.4300	0.4290	0.4290	0.4280	0.4280	0.4270	0
42	Croatia	NaN	NaN	0.3320	0.3330	0.3330	0.3340	0.3340	0.3350	0.3360	0.3360	0
45	Czech Republic	NaN	NaN	NaN	0.3400	0.3410	0.3410	0.3410	0.3410	0.3410	0.3410	0
54	Eritrea	NaN	NaN	NaN	0.1590	0.1590	0.1580	0.1580	0.1570	0.1570	0.1570	0
55	Estonia	NaN	NaN	0.5220	0.5230	0.5240	0.5250	0.5260	0.5260	0.5270	0.5280	0
57	Ethiopia	NaN	NaN	NaN	0.1470	0.1460	0.1440	0.1430	0.1410	0.1400	0.1380	0
63	Georgia	NaN	NaN	0.3960	0.3960	0.3960	0.3970	0.3970	0.3970	0.3970	0.3970	0
86	Kazakhstan	NaN	NaN	0.0126	0.0126	0.0126	0.0126	0.0125	0.0125	0.0125	0.0125	0
90	Kyrgyz Republic	NaN	NaN	0.0438	0.0440	0.0441	0.0442	0.0443	0.0444	0.0445	0.0446	0
92	Latvia	NaN	NaN	0.5120	0.5130	0.5150	0.5160	0.5170	0.5180	0.5190	0.5200	0
98	Lithuania	NaN	NaN	0.3130	0.3140	0.3150	0.3160	0.3180	0.3190	0.3200	0.3210	0
99	Luxembourg	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0
106	Marshall Islands	NaN	0.702	0.7020	0.7020	0.7020	0.7020	0.7020	0.7020	0.7020	0.7020	0
110	Micronesia, Fed. Sts.	NaN	0.909	0.9090	0.9100	0.9100	0.9100	0.9110	0.9110	0.9110	0.9120	0
111	Moldova	NaN	NaN	0.0974	0.0973	0.0974	0.0976	0.0978	0.0980	0.0981	0.0983	0
113	Montenegro	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
126	North Macedonia	NaN	NaN	0.3620	0.3640	0.3660	0.3680	0.3700	0.3710	0.3730	0.3750	0
130	Palau	NaN	0.833	0.8360	0.8390	0.8420	0.8450	0.8480	0.8510	0.8540	0.8570	0
141	Russia	NaN	NaN	0.4940	0.4940	0.4940	0.4940	0.4940	0.4940	0.4940	0.4940	0
148	Serbia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
152	Slovak Republic	NaN	NaN	NaN	0.4000	0.4000	0.4000	0.4000	0.3990	0.3990	0.3990	0
153	Slovenia	NaN	NaN	0.5940	0.5970	0.5990	0.6010	0.6030	0.6050	0.6080	0.6100	0
168	Tajikistan	NaN	NaN	0.0292	0.0292	0.0292	0.0292	0.0292	0.0293	0.0293	0.0293	0
177	Turkmenistan	NaN	NaN	0.0878	0.0878	0.0878	0.0878	0.0878	0.0878	0.0878	0.0878	0
180	Ukraine	NaN	NaN	0.1610	0.1610	0.1620	0.1620	0.1630	0.1630	0.1630	0.1640	0
185	Uzbekistan	NaN	NaN	0.0724	0.0728	0.0731	0.0735	0.0739	0.0743	0.0747	0.0751	0

In [26]:

```python
missing_values = missing_values.copy()
missing_values.loc[:,'N_V'] = missing_values.isna().sum(axis=1)
missing_fourplus = missing_values.query('N_V >= 4')
missing_fourplus
```

Out[26]:

	country	1990	1991	1992	1993	1994	1995	1996	1997	1998	...	2000	2001	200
16	Belgium	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	0.220	0.221	0.22
99	Luxembourg	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	0.335	0.335	0.33
113	Montenegro	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	Na
148	Serbia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	Na

4 rows × 21 columns

In [27]:

```python
missing_fourplus_ind = missing_fourplus.index.values
forest_coverage.drop(missing_fourplus_ind, inplace=True)
forest_coverage.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 188 entries, 0 to 191
Data columns (total 20 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   country  188 non-null    object
 1   1990     162 non-null    float64
 2   1991     165 non-null    float64
 3   1992     184 non-null    float64
 4   1993     188 non-null    float64
 5   1994     188 non-null    float64
 6   1995     188 non-null    float64
 7   1996     188 non-null    float64
 8   1997     188 non-null    float64
 9   1998     188 non-null    float64
 10  1999     188 non-null    float64
 11  2000     188 non-null    float64
 12  2001     188 non-null    float64
 13  2002     188 non-null    float64
 14  2003     188 non-null    float64
 15  2004     188 non-null    float64
 16  2005     188 non-null    float64
 17  2006     188 non-null    float64
 18  2007     188 non-null    float64
 19  2008     188 non-null    float64
dtypes: float64(19), object(1)
memory usage: 30.8+ KB
```

Now I'll fill data for the missing 1990 and 1991 values based on values for following years using backfill

In [28]:
```
forest_coverage.loc[:, '1990':'2008'] = forest_coverage.loc[:, '1990':'2008'].
fillna(method='bfill',axis=1)
```

In [29]:
```
forest_coverage.isnull().sum().sum()
```

Out[29]: 0

In [30]:
```
forest_coverage.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 188 entries, 0 to 191
Data columns (total 20 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   country  188 non-null    object 
 1   1990     188 non-null    float64
 2   1991     188 non-null    float64
 3   1992     188 non-null    float64
 4   1993     188 non-null    float64
 5   1994     188 non-null    float64
 6   1995     188 non-null    float64
 7   1996     188 non-null    float64
 8   1997     188 non-null    float64
 9   1998     188 non-null    float64
 10  1999     188 non-null    float64
 11  2000     188 non-null    float64
 12  2001     188 non-null    float64
 13  2002     188 non-null    float64
 14  2003     188 non-null    float64
 15  2004     188 non-null    float64
 16  2005     188 non-null    float64
 17  2006     188 non-null    float64
 18  2007     188 non-null    float64
 19  2008     188 non-null    float64
dtypes: float64(19), object(1)
memory usage: 30.8+ KB
```

Since removing countries have have created gaps in the index series I will reset the index so that future comparison against other dataframes won't error out on missing indices.

In [31]:
```
forest_coverage.reset_index(drop=True, inplace=True)
```

Total Population Cleaning

Total population has no null values to remove. To prepare this data I will remove years not in our date range for comparison and countries not present in forest coverage data.

Remove years outside of date range (1990-2008)

In [32]:
```
total_pop.drop(total_pop.columns.to_series()["1800":"1989"], axis=1, inplace=True)
total_pop.drop(total_pop.columns.to_series()["2009":"2100"], axis=1, inplace=True)
```

Remove countries from total population not present in Forest Coverage

Find indices of countries in total_pop, but not in forest_coverage and store as missing_countries_ind

In [33]:
```
missing_countries_ind = (pd.merge(total_pop, forest_coverage, on='country', indicator=True, how='outer').query('_merge == "left_only"')[['country']].index.values)
```

Using the indices of missing countries, drop the missing country rows from total_pop dataset

In [34]:
```
total_pop.drop(missing_countries_ind, inplace=True)
```

Reset the index for total population so that there is consistency between datasets on indices to countrie

In [35]:
```
total_pop.reset_index(drop=True, inplace=True)
```

Confirm countries in total_pop and forest_coverage are identical

In [36]:
```
total_pop.country.equals(forest_coverage.country)
```

Out[36]:
```
True
```

Drought Cleaning

There are no null values in the drought dataframe so the only thing necessary to prepare it for exploration is to drop date ranges outside of 1990-2008.

In [37]:
```
drought.drop(drought.columns.to_series()["1970":"1989"], axis=1, inplace=True)
```

```
In [38]: drought.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 123 entries, 0 to 122
Data columns (total 20 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   country  123 non-null    object
 1   1990     123 non-null    int64 
 2   1991     123 non-null    int64 
 3   1992     123 non-null    int64 
 4   1993     123 non-null    int64 
 5   1994     123 non-null    int64 
 6   1995     123 non-null    int64 
 7   1996     123 non-null    int64 
 8   1997     123 non-null    int64 
 9   1998     123 non-null    int64 
 10  1999     123 non-null    int64 
 11  2000     123 non-null    int64 
 12  2001     123 non-null    int64 
 13  2002     123 non-null    int64 
 14  2003     123 non-null    int64 
 15  2004     123 non-null    int64 
 16  2005     123 non-null    int64 
 17  2006     123 non-null    int64 
 18  2007     123 non-null    int64 
 19  2008     123 non-null    int64 
dtypes: int64(19), object(1)
memory usage: 19.3+ KB
```

Extreme Temperature Cleaning

Null values are present in this dataframe, but only in years outside of the established date range. Dropping years before 1990 will remove any null values and finish preparation of this dataframe for exploration.

```
In [39]: ext_temp.drop(ext_temp.columns.to_series()["1971":"1989"], axis=1, inplace=True)
```

In [40]:
```
ext_temp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 68 entries, 0 to 67
Data columns (total 20 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   country  68 non-null     object
 1   1990     68 non-null     int64 
 2   1991     68 non-null     int64 
 3   1992     68 non-null     int64 
 4   1993     68 non-null     int64 
 5   1994     68 non-null     int64 
 6   1995     68 non-null     int64 
 7   1996     68 non-null     int64 
 8   1997     68 non-null     int64 
 9   1998     68 non-null     int64 
 10  1999     68 non-null     int64 
 11  2000     68 non-null     int64 
 12  2001     68 non-null     int64 
 13  2002     68 non-null     int64 
 14  2003     68 non-null     int64 
 15  2004     68 non-null     int64 
 16  2005     68 non-null     int64 
 17  2006     68 non-null     int64 
 18  2007     68 non-null     int64 
 19  2008     68 non-null     int64 
dtypes: int64(19), object(1)
memory usage: 10.8+ KB
```

Flood Cleaning

There are no null values in the flood dataframe so the only thing necessary to prepare it for exploration is to drop date ranges outside of 1990-2008.

In [41]:
```
flood.drop(flood.columns.to_series()["1970":"1989"], axis=1, inplace=True)
```

In [42]:
```
flood.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 166 entries, 0 to 165
Data columns (total 20 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   country  166 non-null    object
 1   1990     166 non-null    int64 
 2   1991     166 non-null    int64 
 3   1992     166 non-null    int64 
 4   1993     166 non-null    int64 
 5   1994     166 non-null    int64 
 6   1995     166 non-null    int64 
 7   1996     166 non-null    int64 
 8   1997     166 non-null    int64 
 9   1998     166 non-null    int64 
 10  1999     166 non-null    int64 
 11  2000     166 non-null    int64 
 12  2001     166 non-null    int64 
 13  2002     166 non-null    int64 
 14  2003     166 non-null    int64 
 15  2004     166 non-null    int64 
 16  2005     166 non-null    int64 
 17  2006     166 non-null    int64 
 18  2007     166 non-null    int64 
 19  2008     166 non-null    int64 
dtypes: int64(19), object(1)
memory usage: 26.1+ KB
```

Data Cleaning Assesment

Both independent variables are now equal in years and countries covered. There are no null values in either.

All three dependent variables are free of null values. The data for years outside of our established range for each have been dropped so that all five dataframes have identical date ranges. They range in number of countries covered, but seeing as they will each be compared seperately against the independent variables they will not be changed.

Exploratory Data Analysis

Do countries with a higher percentage of forest coverage have a lower number of affected from floods, droughts and extreme temperatures?

To answer this question I will compare forest coverage and natural disaster affected in each dataframe on a 6 year average for top countries in the given 6 years.

Top countries will include the 5 countries with the highest 6 year average forest coverage percentage.

I will then create a baseline to compare against by averaging forest coverage across all countries and plotting that against average natural disaster affected of all countries.

In [43]:
```
# Create 3 new columns in forest_coverage for 90-96 average, 97-02 average, an
d 03-08 average
fc_avg = forest_coverage.copy()
fc_avg['90_96_avg_fc'] = fc_avg.loc[:,'1990':'1996'].mean(axis=1)
fc_avg['97_02_avg_fc'] = fc_avg.loc[:,'1997':'2002'].mean(axis=1)
fc_avg['03_08_avg_fc'] = fc_avg.loc[:,'2003':'2008'].mean(axis=1)
fc_avg.drop(fc_avg.loc[:,'1990':'2008'], axis=1, inplace=True)
```

In [44]:
```
#Create 3 new columns in total_pop for 90-96 average, 97-02 average, and 03-08
average
tp_avg = total_pop.copy()
tp_avg['90_96_avg_tp'] = tp_avg.loc[:,'1990':'1996'].mean(axis=1)
tp_avg['97_02_avg_tp'] = tp_avg.loc[:,'1997':'2002'].mean(axis=1)
tp_avg['03_08_avg_tp'] = tp_avg.loc[:,'2003':'2008'].mean(axis=1)
tp_avg.drop(tp_avg.loc[:,'1990':'2008'], axis=1, inplace=True)
```

In [45]:
```
# Create 3 new columns in flood for 90-96 average, 97-02 average, and 03-08 av
erage
flood_avg = flood.copy()
flood_avg['90_96_avg_fl'] = flood_avg.loc[:,'1990':'1996'].mean(axis=1)
flood_avg['97_02_avg_fl'] = flood_avg.loc[:,'1997':'2002'].mean(axis=1)
flood_avg['03_08_avg_fl'] = flood_avg.loc[:,'2003':'2008'].mean(axis=1)
flood_avg.drop(flood_avg.loc[:,'1990':'2008'], axis=1, inplace=True)
```

In [46]:
```
# Create 3 new columns in extreme_temp for 90-96 average, 97-02 average, and 0
3-08 average
ext_temp_avg = ext_temp.copy()
ext_temp_avg['90_96_avg_et'] = ext_temp_avg.loc[:,'1990':'1996'].mean(axis=1)
ext_temp_avg['97_02_avg_et'] = ext_temp_avg.loc[:,'1997':'2002'].mean(axis=1)
ext_temp_avg['03_08_avg_et'] = ext_temp_avg.loc[:,'2003':'2008'].mean(axis=1)
ext_temp_avg.drop(ext_temp_avg.loc[:,'1990':'2008'], axis=1, inplace=True)
```

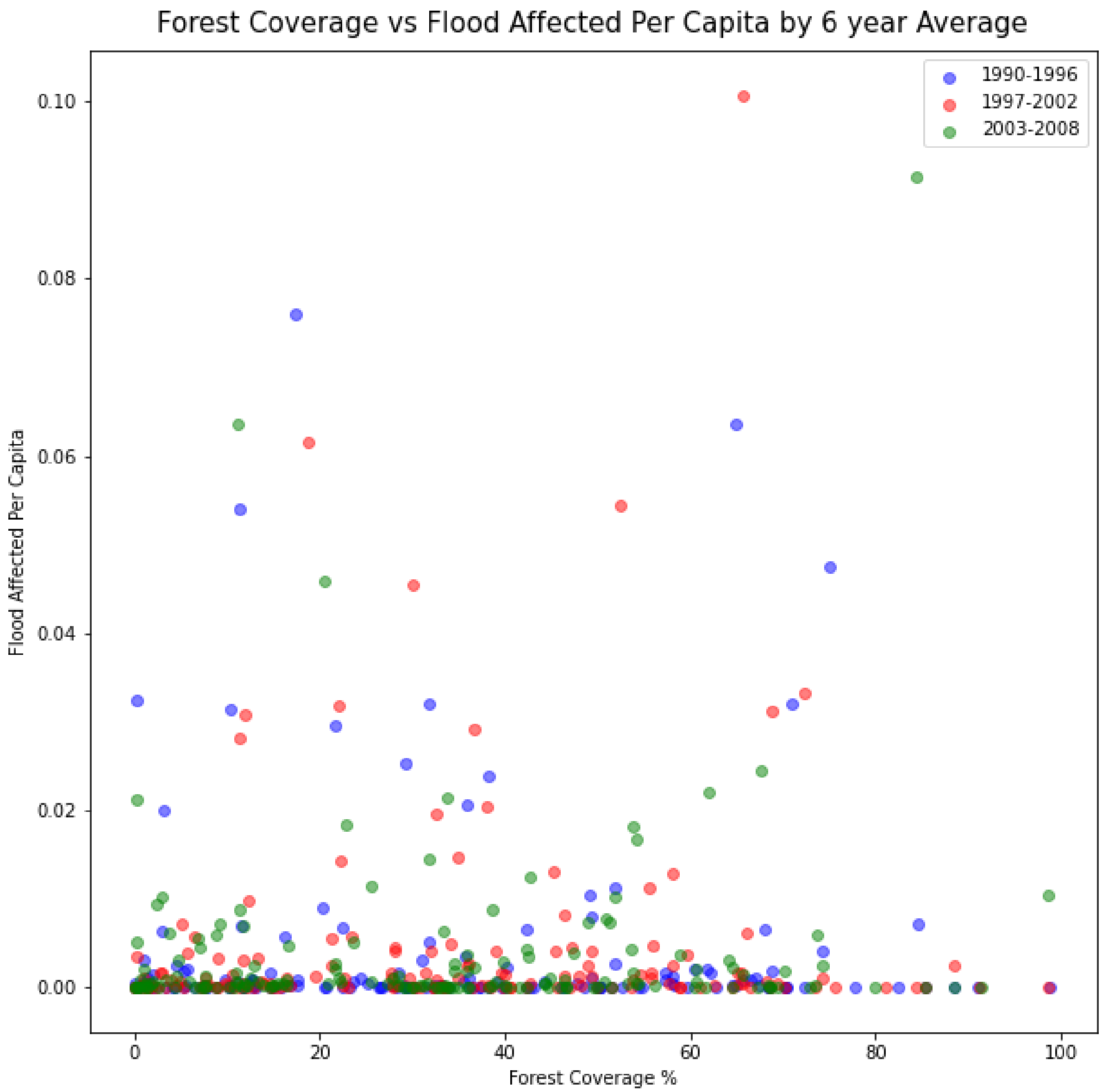
In [47]:
```
# Create 3 new columns in drought for 90-96 average, 97-02 average, and 03-08
 average
drought_avg = drought.copy()
drought_avg['90_96_avg_dr'] = drought_avg.loc[:,'1990':'1996'].mean(axis=1)
drought_avg['97_02_avg_dr'] = drought_avg.loc[:,'1997':'2002'].mean(axis=1)
drought_avg['03_08_avg_dr'] = drought_avg.loc[:,'2003':'2008'].mean(axis=1)
drought_avg.drop(drought_avg.loc[:,'1990':'2008'], axis=1, inplace=True)
```

In [48]:
```
# Create df of forest_coverage and flood average columns where country data is
available in both
fc_v_flood = pd.merge(fc_avg, flood_avg, on='country', how='inner')

#Merge with total population to allow fo per capita calculations
fc_v_flood_pc = pd.merge(fc_v_flood, tp_avg, on='country', how='inner')
```

In [49]:
```
#Add a per capita affected column for each 6 year average in flood by dividing
total affected by total population
fc_v_flood_pc['af_90_96_pc'] = fc_v_flood_pc['90_96_avg_fl']/fc_v_flood_pc['90
_96_avg_tp']
fc_v_flood_pc['af_97_02_pc'] = fc_v_flood_pc['97_02_avg_fl']/fc_v_flood_pc['97
_02_avg_tp']
fc_v_flood_pc['af_03_08_pc'] = fc_v_flood_pc['03_08_avg_fl']/fc_v_flood_pc['03
_08_avg_tp']
```

In [50]:

```
plt.figure(figsize=(10,10))

#1990-1996 Comparison
y_1 = fc_v_flood_pc['af_90_96_pc'] #Flood Affected Average of 1990-1996 / Tota
l Population Average of 1990-1996
x_1 = fc_v_flood_pc['90_96_avg_fc']*100 #Forest Coverage Average from 1990-199
6 times 100 for percentage

#1997-2002 Comparison
y_2 = fc_v_flood_pc['af_97_02_pc'] #Flood Affected Average of 1997-2002 / Tota
l Population Average of 1997-2002
x_2 = fc_v_flood_pc['97_02_avg_fc']*100 #Forest Coverage Average from 1997-200
2 times 100 for percentage

#2003-2008 Comparison
y_3 = fc_v_flood_pc['af_03_08_pc'] #Flood Affected Average of 2003-2008 / Tota
l Population Average of 2003-2008
x_3 = fc_v_flood_pc['03_08_avg_fc']*100 #Forest Coverage Average from 2003-200
8 times 100 for percentage

plt.scatter(x_1, y_1, c='blue', alpha=0.5, label='1990-1996')
plt.scatter(x_2, y_2, c='red', alpha=0.5, label='1997-2002')
plt.scatter(x_3, y_3, c='green', alpha=0.5, label='2003-2008')

plt.title("Forest Coverage vs Flood Affected Per Capita by 6 year Average", fo
ntsize=15, pad=10)
plt.legend()
plt.ylabel('Flood Affected Per Capita')
plt.xlabel('Forest Coverage %');
```



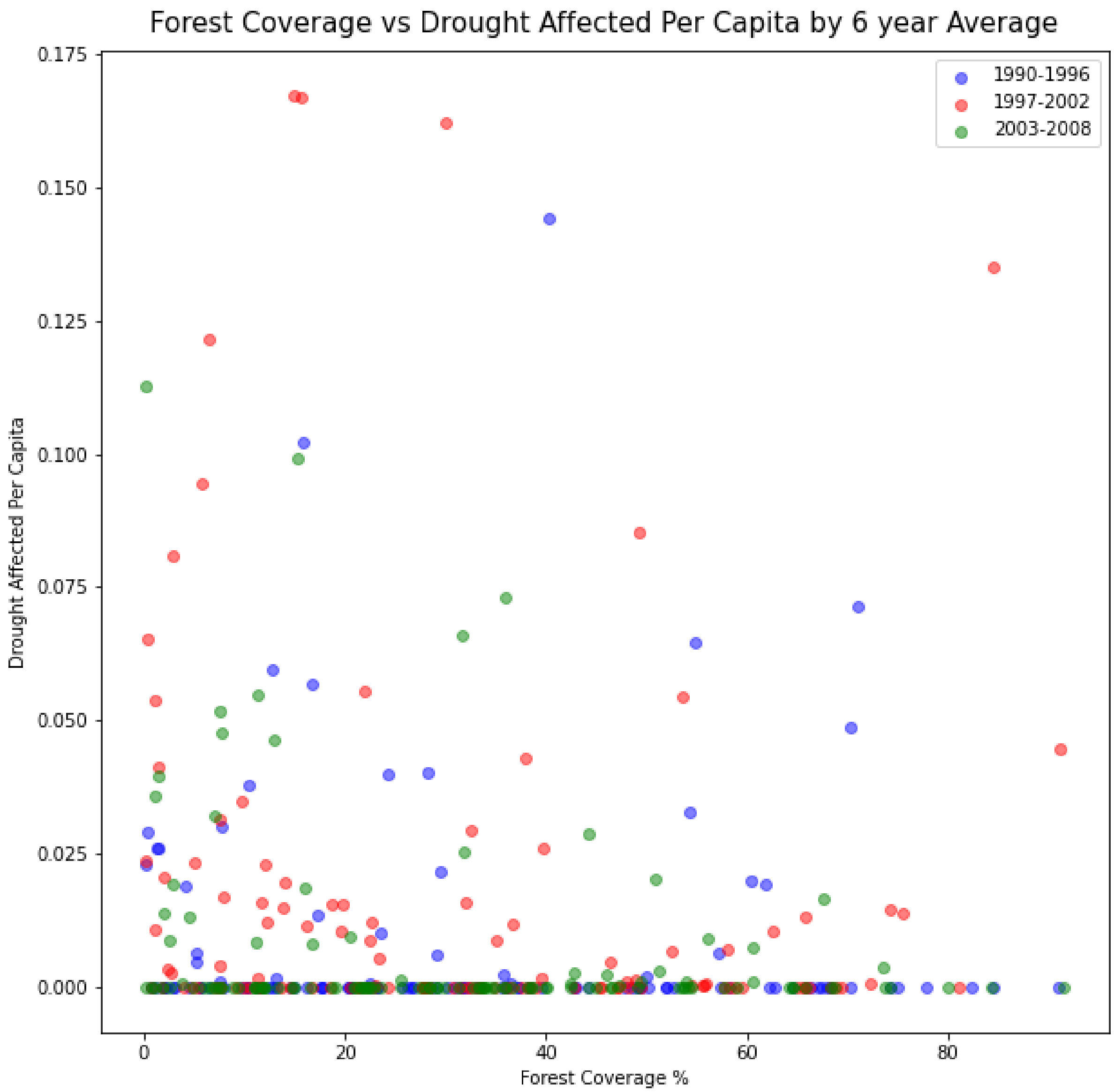
The data seems to be only slightly right skewed.

In [51]:
```
# Create df of forest_coverage and drought average columns where country data
 is available in both
fc_v_drought = pd.merge(fc_avg, drought_avg, on='country', how='inner')

#Merge with total population to allow fo per capita calculations
fc_v_drought_pc = pd.merge(fc_v_drought, tp_avg, on='country', how='inner')
```

In [52]:
```
#Add a per capita affected column for each 6 year average in drought by dividi
ng total affected by total population
fc_v_drought_pc['af_90_96_pc'] = fc_v_drought_pc['90_96_avg_dr']/fc_v_drought_
pc['90_96_avg_tp']
fc_v_drought_pc['af_97_02_pc'] = fc_v_drought_pc['97_02_avg_dr']/fc_v_drought_
pc['97_02_avg_tp']
fc_v_drought_pc['af_03_08_pc'] = fc_v_drought_pc['03_08_avg_dr']/fc_v_drought_
pc['03_08_avg_tp']
```

In [53]:

```
plt.figure(figsize=(10,10))

#1990-1996 Comparison
y_1 = fc_v_drought_pc['af_90_96_pc'] #Drought Affected Average of 1990-1996 /
 Total Population Average of 1990-1996
x_1 = fc_v_drought_pc['90_96_avg_fc']*100 #Forest Coverage Average from 1990-1
996 times 100 for percentage

#1997-2002 Comparison
y_2 = fc_v_drought_pc['af_97_02_pc'] #Drought Affected Average of 1997-2002 /
 Total Population Average of 1997-2002
x_2 = fc_v_drought_pc['97_02_avg_fc']*100 #Forest Coverage Average from 1997-2
002 times 100 for percentage

y_3 = fc_v_drought_pc['af_03_08_pc'] #Drought Affected Average of 2003-2008 /
 Total Population Average of 2003-2008
x_3 = fc_v_drought_pc['03_08_avg_fc']*100 #Forest Coverage Average from 2003-2
008 times 100 for percentage

plt.scatter(x_1, y_1, c='blue', alpha=0.5, label='1990-1996')
plt.scatter(x_2, y_2, c='red', alpha=0.5, label='1997-2002')
plt.scatter(x_3, y_3, c='green', alpha=0.5, label='2003-2008')

plt.title("Forest Coverage vs Drought Affected Per Capita by 6 year Average",
fontsize=15, pad=10)
plt.legend()
plt.ylabel('Drought Affected Per Capita')
plt.xlabel('Forest Coverage %');
```



The data for drought affected per capital appears to skew slightly more to the right than flood.

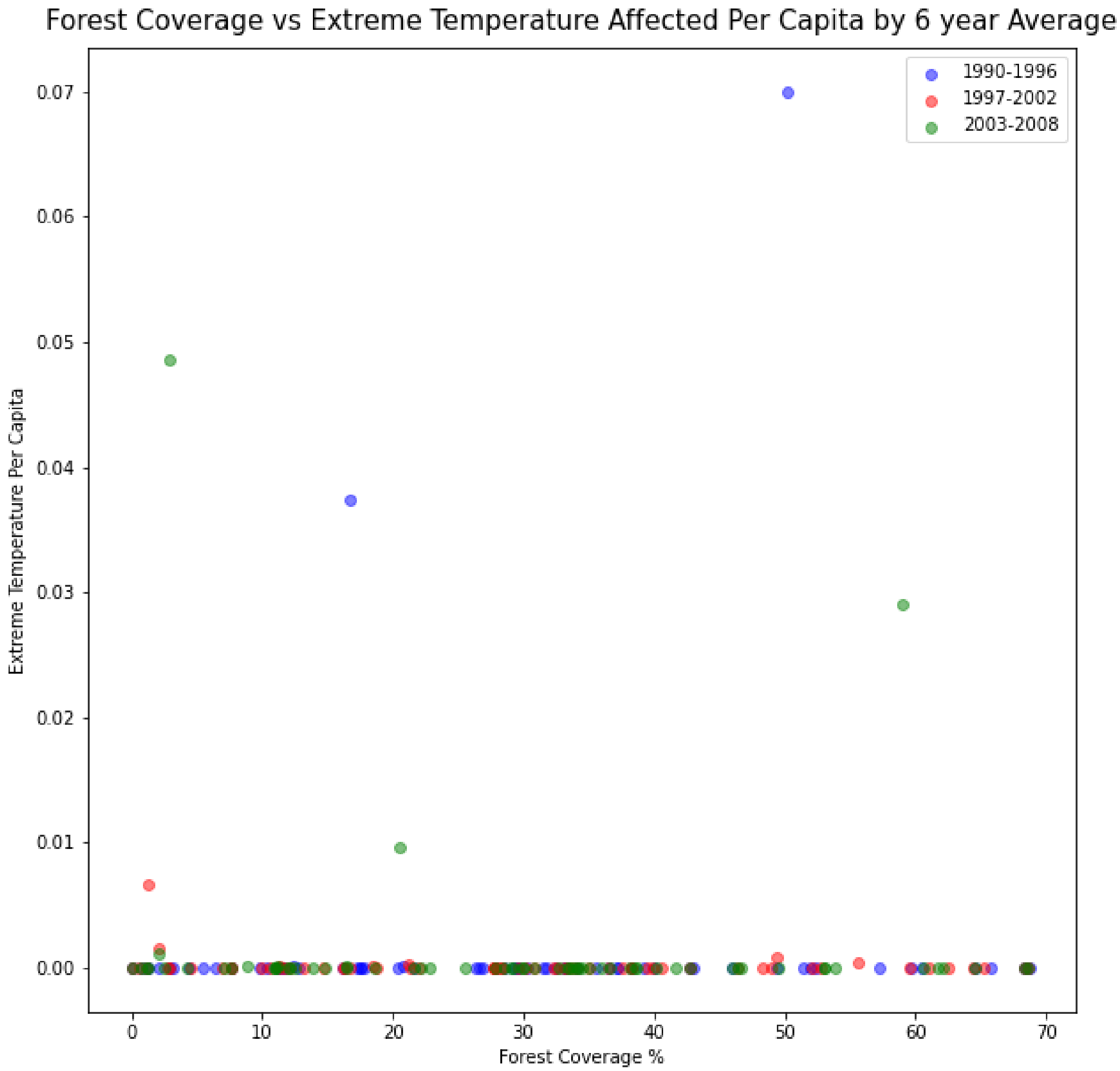
In [54]:
```
# Create df of forest_coverage and extreme temperature average columns where c
ountry data is available in both
fc_v_ext_temp = pd.merge(fc_avg, ext_temp_avg, on='country', how='inner')

#Merge with total population to allow fo per capita calculations
fc_v__ext_temp_pc = pd.merge(fc_v_ext_temp, tp_avg, on='country', how='inner')
```

In [55]:
```
#Add a per capita affected column for each 6 year average in extreme temperatu
re by dividing total affected by total population
fc_v__ext_temp_pc['af_90_96_pc'] = fc_v__ext_temp_pc['90_96_avg_et']/fc_v__ext
_temp_pc['90_96_avg_tp']
fc_v__ext_temp_pc['af_97_02_pc'] = fc_v__ext_temp_pc['97_02_avg_et']/fc_v__ext
_temp_pc['97_02_avg_tp']
fc_v__ext_temp_pc['af_03_08_pc'] = fc_v__ext_temp_pc['03_08_avg_et']/fc_v__ext
_temp_pc['03_08_avg_tp']
```

In [56]:

```
plt.figure(figsize=(10,10))

#1990-1996 Comparison
y_1 = fc_v__ext_temp_pc['af_90_96_pc'] #Extreme temperature Affected Average o
f 1990-1996 / Total Population Average of 1990-1996
x_1 = fc_v__ext_temp_pc['90_96_avg_fc']*100 #Forest Coverage Average from 1990
-1996 times 100 for percentage

#1997-2002 Comparison
y_2 = fc_v__ext_temp_pc['af_97_02_pc'] #Extreme temperature Affected Average o
f 1997-2002 / Total Population Average of 1997-2002
x_2 = fc_v__ext_temp_pc['97_02_avg_fc']*100 #Forest Coverage Average from 1997
-2002 times 100 for percentage

y_3 = fc_v__ext_temp_pc['af_03_08_pc'] #Extreme temperature Affected Average o
f 2003-2008 / Total Population Average of 2003-2008
x_3 = fc_v__ext_temp_pc['03_08_avg_fc']*100 #Forest Coverage Average from 2003
-2008 times 100 for percentage

plt.scatter(x_1, y_1, c='blue', alpha=0.5, label='1990-1996')
plt.scatter(x_2, y_2, c='red', alpha=0.5, label='1997-2002')
plt.scatter(x_3, y_3, c='green', alpha=0.5, label='2003-2008')

plt.title("Forest Coverage vs Extreme Temperature Affected Per Capita by 6 yea
r Average", fontsize=15, pad=10)
plt.legend()
plt.ylabel('Extreme Temperature Per Capita')
plt.xlabel('Forest Coverage %')
;
```

Out[56]: ''



There appears to be no relationship between extreme temperatures and forest coverage. Almost all information of extreme temperatue per capita in each 6 year average are significantly lower than in either flood or drought affected datasets.

Answer:

The comparison of each dataset for flood, drought and extreme temperature does not show a strong relationship between percentage of forest coverage and any natural disaster.

However, between the three datasets, flood and drought affected had a slight skew to the right. The scatter plot for these data sets show countries with higher percentage of forest coverage had slightly fewer affected per capita.

Which natural disaster type affects the most countries on a per/capita basis in countries with the lowest percentage of forest coverage?

To find the answer to this questions I will compare the average number of affected from each disaster in each country across the data range for countries in which data for all three natural disasters is available.

To establish the countries with the lowest percentage of forest coverage I will pull from that data the 10 countries with the lowest average forest coverage across the date range.

In [57]:
```
drought.head()
```

Out[57]:

	country	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001
0	Afghanistan	0	0	0	0	0	0	0	0	0	0	2580000	0
1	Albania	0	0	0	0	0	0	0	0	0	0	0	0
2	Algeria	0	0	0	0	0	0	0	0	0	0	0	0
3	Angola	0	0	0	0	0	0	0	105000	0	0	0	58
4	Antigua and Barbuda	0	0	0	0	0	0	0	0	0	0	0	0

In [58]:

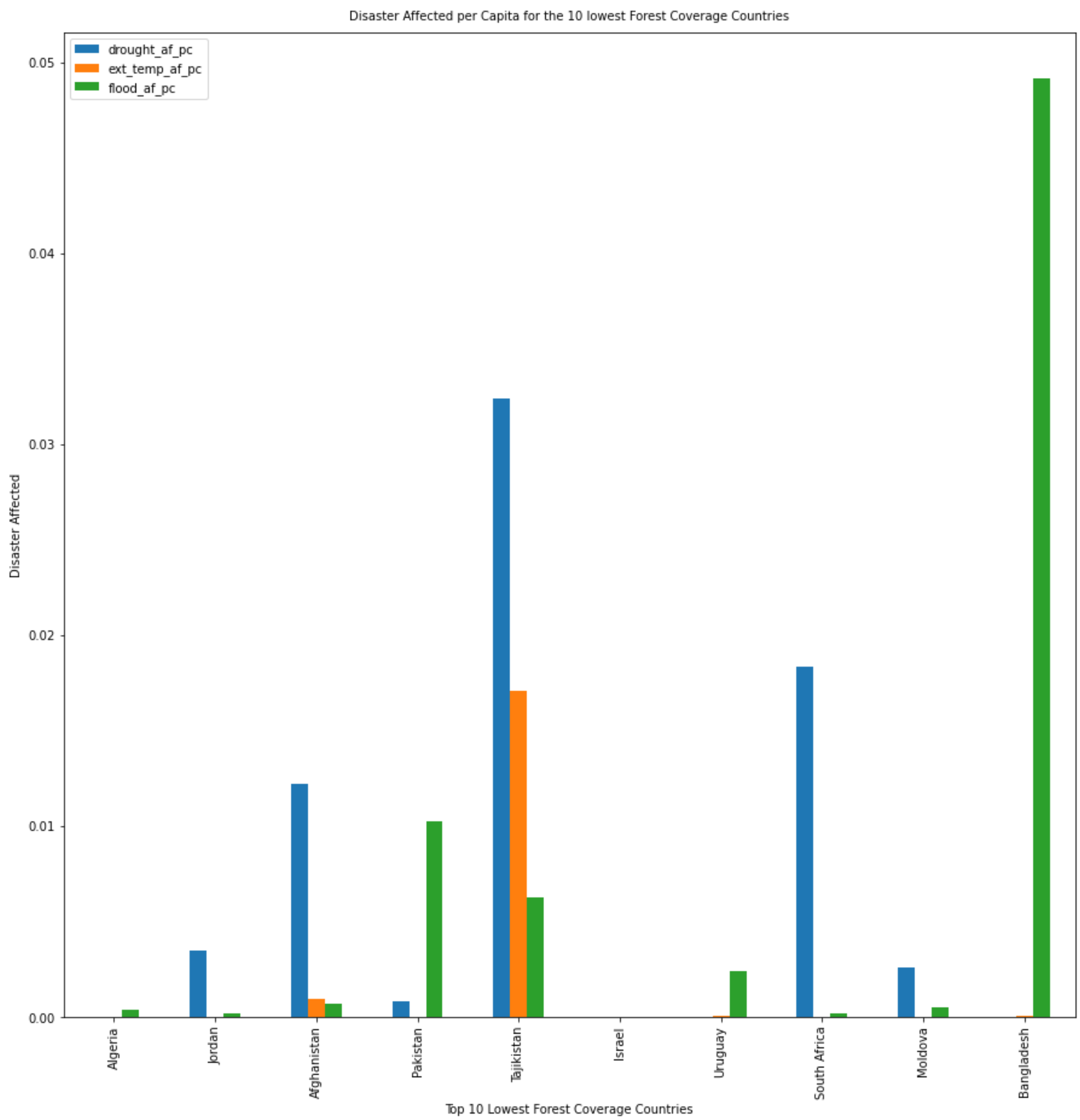
```
#Establish average affected for each country
drought_avg = drought.copy()
drought_avg['mean_drought'] = drought_avg.mean(axis=1)
drought_avg.drop(drought_avg.loc[:, '1990':'2008'], inplace = True, axis = 1)

ext_temp_avg = ext_temp .copy()
ext_temp_avg['mean_ext_temp'] = ext_temp_avg.mean(axis=1)
ext_temp_avg.drop(ext_temp_avg.loc[:, '1990':'2008'], inplace = True, axis = 1
)

flood_avg = flood.copy()
flood_avg['mean_flood'] = flood_avg.mean(axis=1)
flood_avg.drop(flood_avg.loc[:, '1990':'2008'], inplace = True, axis = 1)

#Establish average forest coverage for each country
forest_coverage_avg = forest_coverage.copy()
forest_coverage_avg.loc[:, '1990':'2008'] = forest_coverage_avg.loc[:, '1990':
'2008'].astype(float)
forest_coverage_avg['mean_coverage'] = forest_coverage_avg.loc[:, '1990':'200
8'].mean(axis=1)
forest_coverage_avg.drop(forest_coverage_avg.loc[:, '1990':'2008'], inplace =
True, axis = 1)

#Establish average population for each county
total_pop_avg = total_pop.copy()
total_pop_avg['mean_population'] = total_pop_avg.mean(axis=1)
total_pop_avg.drop(total_pop_avg.loc[:, '1990':'2008'], inplace = True, axis =
1)
```

In [59]:
```python
#Merge the dataframes together
from functools import reduce
dataframes = [drought_avg, ext_temp_avg, flood_avg, forest_coverage_avg, total
_pop_avg]
disaster_avg = reduce(lambda left,right: pd.merge(left,right,on='country'), da
taframes)
disaster_avg = disaster_avg.sort_values('mean_coverage').head(10)

#Create columns for affected per capita
disaster_avg['drought_af_pc'] = disaster_avg['mean_drought']/disaster_avg['mea
n_population']
disaster_avg['ext_temp_af_pc'] = disaster_avg['mean_ext_temp']/disaster_avg['m
ean_population']
disaster_avg['flood_af_pc'] = disaster_avg['mean_flood']/disaster_avg['mean_po
pulation']

disaster_avg.set_index("country", inplace=True)
disaster_avg
```

Out[59]:

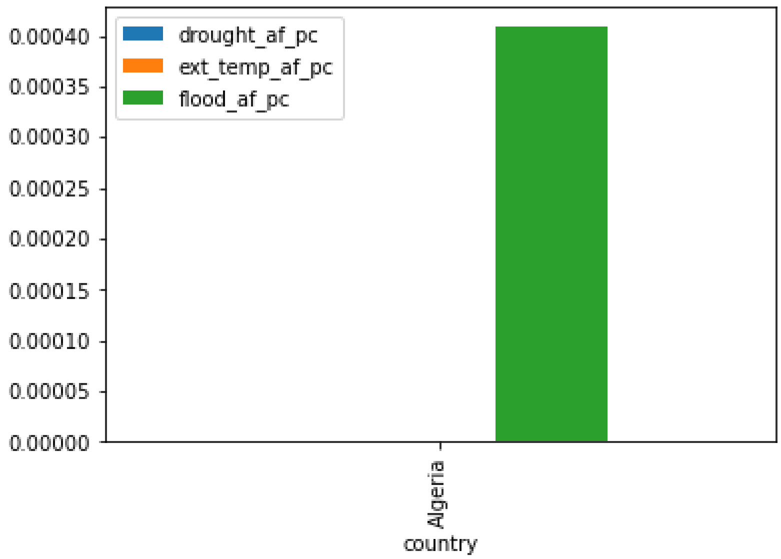
	mean_drought	mean_ext_temp	mean_flood	mean_coverage	mean_population	dr
country						
Algeria	0.631579	2.105263	1.243405e+04	0.006784	3.045263e+07	2
Jordan	17368.421053	1.421053	9.474737e+02	0.011000	5.016316e+06	3
Afghanistan	250526.315789	19602.368421	1.486368e+04	0.020700	2.051053e+07	1
Pakistan	115789.473684	93.947368	1.422206e+06	0.027932	1.388421e+08	8
Tajikistan	200000.000000	105263.157895	3.858947e+04	0.029263	6.173684e+06	3
Israel	0.000000	0.000000	5.342105e+01	0.068153	5.765789e+06	0
Uruguay	0.000000	126.684211	7.800000e+03	0.072305	3.265789e+06	0
South Africa	805263.157895	2.736842	8.110421e+03	0.076200	4.390000e+07	1
Moldova	11052.736842	0.684211	2.108474e+03	0.102274	4.250000e+06	2
Bangladesh	0.000000	9950.263158	6.126642e+06	0.113000	1.247368e+08	0

In [60]:
```
disaster_avg[['drought_af_pc', 'ext_temp_af_pc', 'flood_af_pc']].plot(kind='bar',figsize=(15,15), width = .5)

plt.title("Disaster Affected per Capita for the 10 lowest Forest Coverage Countries", fontsize=10, pad=10)
#plt.legend(disaster_avg['drought_af_pc'],['test'])
plt.ylabel('Disaster Affected')
plt.xlabel('Top 10 Lowest Forest Coverage Countries')

plt.show();
```

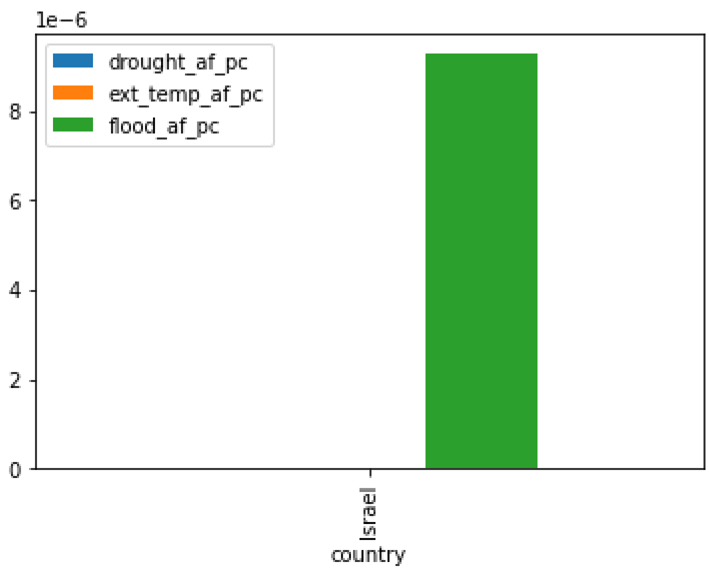


In 5 out of 10 countries its apparent that Drought is the disaster with the most affected per capita. In 3 out of 10 its clear that floods are the disaster with the most affected per capita, and with the remaining 2 countries, Israel and Algeria, the data needs to be looked at on an individual graph to find conclusive information.

In [61]:
```
disaster_avg.loc[['Algeria']][['drought_af_pc', 'ext_temp_af_pc', 'flood_af_p
c']].plot(kind='bar')

plt.show();
```



In [62]:
```
disaster_avg.loc[['Israel']][['drought_af_pc', 'ext_temp_af_pc', 'flood_af_pc'
]].plot(kind='bar')
plt.show()
```



Both Algeria and Israel have significantly more flood affected per capita than either other disaster, which has zero or near zero affected by drought or extreme temperatue in the 18 year range on average.

The addition of Isreal and Algeria to the count of the countries most affected by floods makes it 5 out of the 10 countries, or 50%.

Answer:

For the 10 countries with the lowest forest coverage area, 50% are most affected by droughts on average 50% are most affected by floods.

Over time, do countries where forest coverage increases have decreaseing number of those affected by disasters?

To answer this question I will look at two factors in the datasets. First, for each country how much has forest coverage changed over the course of the 18 years in the date ranges available.

Next, I'll compare that against the change in total disaster affected over time.

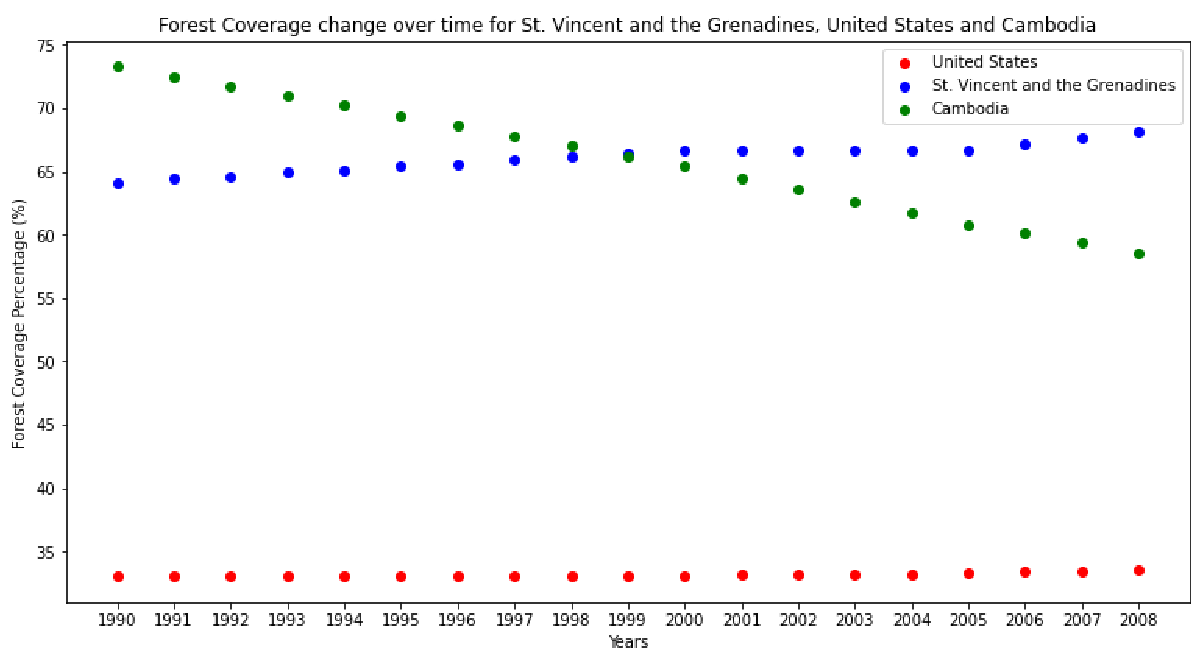
To find how forest coverage and total disaster affected has changed over time I will use linear regression to find trendlines for each and then look for values of the slope of each, positive values will show increasing values over time and and negative shows a decreasing.

The answer to the question will be determing on whether countries with a positive value for forest coverage will show a negative value for disaster affected.

In [63]:

```
fc = forest_coverage
fc.set_index('country', inplace=True)
```

In [64]:
```
fig = plt.figure(figsize = (10,5))
ax =fig.add_axes([0,0,1,1])

years = fc.columns
year_int = fc.columns.astype(int)


#United States
ax.scatter(years, fc.loc['United States']*100, color='r', label = 'United Stat
es')


#St. Vincent and the Grenadines
ax.scatter(years, fc.loc['St. Vincent and the Grenadines']*100, color='b', lab
el = 'St. Vincent and the Grenadines')

#Cambodia
ax.scatter(years, fc.loc['Cambodia']*100, color='g', label = 'Cambodia')

#Labels & Legends
ax.set_xlabel('Years')
ax.set_ylabel('Forest Coverage Percentage (%)')
ax.set_title('Forest Coverage change over time for St. Vincent and the Grenadi
nes, United States and Cambodia')
plt.legend()



plt.show()
```



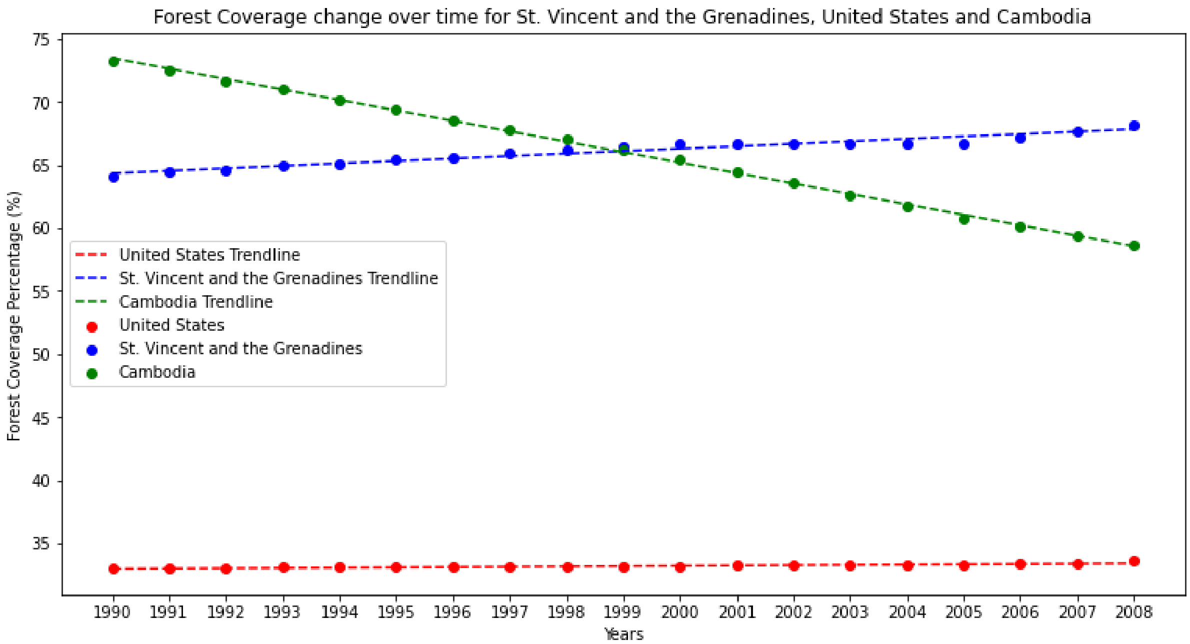
Using The United States, Cambodia and St. Vincent and the Grenadines as an example it can be seen that these three countries have different outcomes in changes of forestation over time. The United States stays relatively the same, St. Vincent and the Grenadines increases forest coverage over time, and Cambodia decreases forest coverage over time.

Using linear regression to create trendlines on each of these datasets will clearly illustrate the direction of change over time.

I will use numpy's polyfit function to create trendlines for each of the 3 example countries

In [65]:
```
fig = plt.figure(figsize = (10,5))
ax =fig.add_axes([0,0,1,1])

years = fc.columns
year_int = fc.columns.astype(int)


#United States

## Scatter Plot of Forest Coverage
ax.scatter(years, fc.loc['United States']*100, color='r', label = 'United Stat
es')

##Trendline Plot of Forest Coverage
y_us = fc.loc['United States']*100

z_us = np.polyfit(year_int, y_us, 1)
p_us = np.poly1d(z_us)
plt.plot(years,p_us(year_int),"r--", label='United States Trendline')



#Belize

## Scatter Plot of Forest Coverage
ax.scatter(years, fc.loc['St. Vincent and the Grenadines']*100, color='b', lab
el = 'St. Vincent and the Grenadines')

##Trendline Plot of Forest Coverage
y_svg = fc.loc['St. Vincent and the Grenadines']*100

z_svg = np.polyfit(year_int, y_svg, 1)
p_svg = np.poly1d(z_svg)
plt.plot(years,p_svg(year_int),"b--", label='St. Vincent and the Grenadines Tr
endline')



#Cambodia

## Scatter Plot of Forest Coverage
ax.scatter(years, fc.loc['Cambodia']*100, color='g', label = 'Cambodia')

##Trendline Plot of Forest Coverage
y_cam = fc.loc['Cambodia']*100

z_cam = np.polyfit(year_int, y_cam, 1)
p_cam = np.poly1d(z_cam)

plt.plot(years,p_cam(year_int),"g--", label='Cambodia Trendline')

#Labels & Legends
ax.set_xlabel('Years')
ax.set_ylabel('Forest Coverage Percentage (%)')
ax.set_title('Forest Coverage change over time for St. Vincent and the Grenadi
nes, United States and Cambodia')
```

```
plt.legend()



plt.show()
```



The trendlines further illustrate the pattern for the data in each country towards increasing, decreasing, or maintaining forest coverage.

As a function, polyfit will calculate the slope (at index [0]) and y intercept (at index [1]) of each line. The slope of the line will show whether a country is increasing forest coverage if the slope value is greater than 0, decreasing forest coverage if the slope value is less than zero, or not changing forest coverage if the slope value is equal to zero

In [66]:
```
s_us = np.polyfit(year_int, y_us, 1).round(decimals=6)
s_cam = np.polyfit(year_int, y_cam, 1).round(decimals=6)
s_svg = np.polyfit(year_int, y_svg, 1).round(decimals=6)
print("The slope of the United States trendline is: ", s_us[0])
print("The slope of the Cambodia trendline is: ", s_cam[0])
print("The slope of the St. Vincent and the Grenadines trendline is: ", s_svg[
0])
```

```
The slope of the United States trendline is:  0.024561
The slope of the Cambodia trendline is:  -0.828772
The slope of the St. Vincent and the Grenadines trendline is:  0.194561
```

To apply this function to every row of the dataset, I will create a function "get_slope" and using pandas .apply() to create a new column in each row with the slope value inputted.

In [67]:
```
def get_slope_fc(x):
    return np.polyfit(fc.loc[:, '1990':'2008'].columns.astype(int), x, 1)[0]*1
00
```

In [68]:
```
get_slope_fc(fc.loc['Cambodia'])
```
Out[68]:
```
-0.8287719298245856
```

A check of the function using Cambodia as a previously known slope value shows its working as intended.

In [69]:
```
slope_values = fc.apply(get_slope_fc, axis=1)

fc.loc[:,'slopes'] = slope_values
```

In [70]:
```
fc.loc['Cambodia','slopes']
```
Out[70]:
```
-0.8287719298245856
```

A check of the dataframe with the new column shows the values are correct.

In [71]:
```
fc.head()
```
Out[71]:

	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	20
country											
Afghanistan	0.0207	0.0207	0.0207	0.0207	0.0207	0.0207	0.0207	0.0207	0.0207	0.0207	0.02
Albania	0.2880	0.2870	0.2860	0.2860	0.2850	0.2840	0.2840	0.2830	0.2820	0.2810	0.28
Algeria	0.0070	0.0070	0.0069	0.0069	0.0069	0.0068	0.0068	0.0067	0.0067	0.0067	0.00
Andorra	0.3400	0.3400	0.3400	0.3400	0.3400	0.3400	0.3400	0.3400	0.3400	0.3400	0.34
Angola	0.4890	0.4880	0.4870	0.4860	0.4850	0.4840	0.4830	0.4820	0.4810	0.4800	0.47

Next, to see how each countries total number of affected changed over time, I'll create a new dataframe that is a merge of all 3 disasters, with a total for each year of those affected. Any countries for which data is not available for all 3 disaster types will be excluded.

To get a trendline slope of change over time, each column needs to have a total of affected for all 3 disaster types in each year. Merging the dataframes together will result in a new dataframe with data from only the countries that have available data for all 3 disasters, but will result in multiple columns for each year, rather than 1 column with a total of all 3 disasters.

Concatenating the 3 dataframes together, and using groupby to sum up the totals of all 3 disasters in each country for each year would solve this, but before merging would result in several null values for countries not present in all 3 dataframes.

To solve this, I will a create list of countries present in all 3 dataframes, then drop the rest of the countries from the 3 dataframes. Once that is accomplished, I can concantenate the 3 together, using groupby and sum to get the total affected of all 3 disasters in each year for each country.

Using that new dataframe, I can then calculate the change in affected over time, compare that against the rate of change in forest coverage and explore whether increasing forest coverage correlated to a decrease in total affected over time.

In [72]:
```
disaster_dataframes = [flood, drought, ext_temp]
included_countries = reduce(lambda left,right: pd.merge(left,right,on='countr
y'), disaster_dataframes).loc[:,'country']

flood_included = pd.merge(flood, included_countries, how='inner')
drought_included = pd.merge(drought, included_countries, how='inner')
ext_temp_included = pd.merge(ext_temp, included_countries, how='inner')

included_dfs = [flood_included, drought_included, ext_temp_included]

total_disasters = pd.concat(included_dfs).groupby(['country']).sum().reset_ind
ex()
total_disasters.set_index('country', inplace=True)
```

In [73]:
```
def get_slope_td(x):
    return np.polyfit(total_disasters.loc[:, '1990':'2008'].columns.astype(int
), x, 1)[0]
```

In [74]:
```
get_slope_td(total_disasters.loc['Albania'])
```

Out[74]:
```
-100.99473684214965
```

A check of the function using Albania shows slope is calculated as intended.

In [75]:
```
slope_values = total_disasters.apply(get_slope_td, axis=1)

total_disasters.loc[:,'slopes_td'] = slope_values
```

In [76]:
```
total_disasters.loc['Albania','slopes_td']
```

Out[76]:
```
-100.99473684214965
```

A check of the dataframe with the new column shows the values are correct and in place.

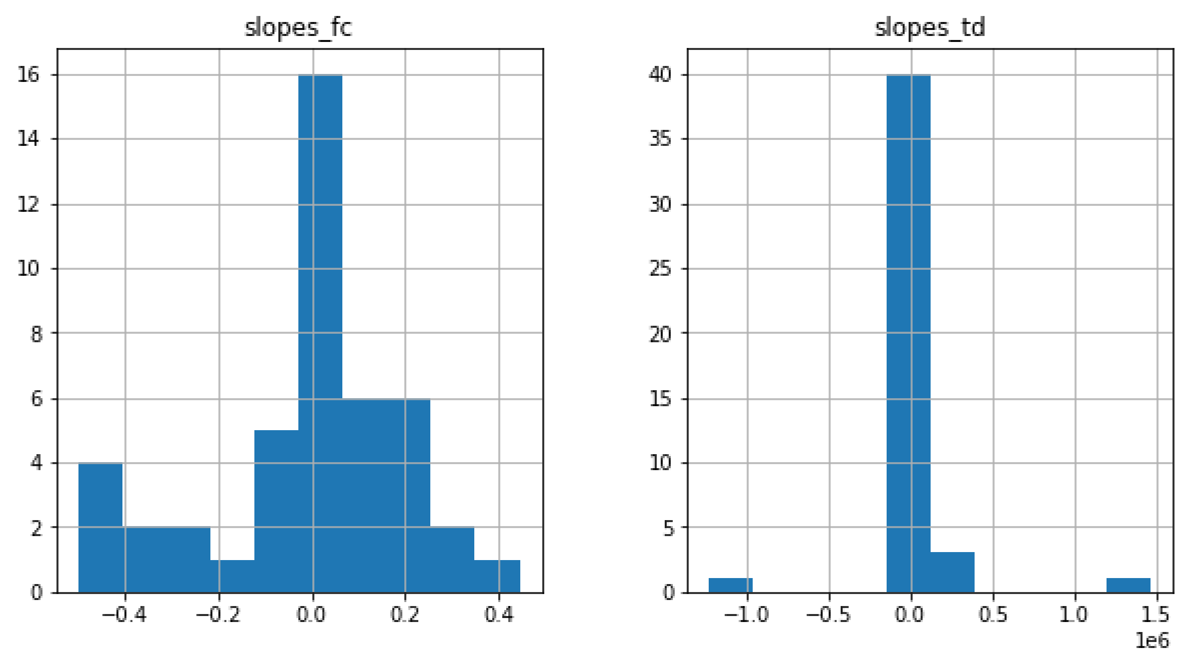
Now that I have slope values for trend lines in both datasets, I can plot that information to compare forest coverage change over time in the fc dataset with the change in affected over time in the total_disaster dataset.

To ensure I'm only comparing countries where I have change information for both disasters and forest coverage, I will drop all unneccessary columns and then join the two together on an inner join with merge.

In [77]:
```
fc.drop(fc.columns.to_series()[:"2008"], axis = 1, inplace = True)
fc.rename(columns={'slopes': 'slopes_fc'}, inplace=True)
total_disasters.drop(total_disasters.columns.to_series()[:"2008"], axis = 1, i
nplace = True)
```

In [78]:
```
change = pd.merge(total_disasters, fc, how = "inner", on='country')
```

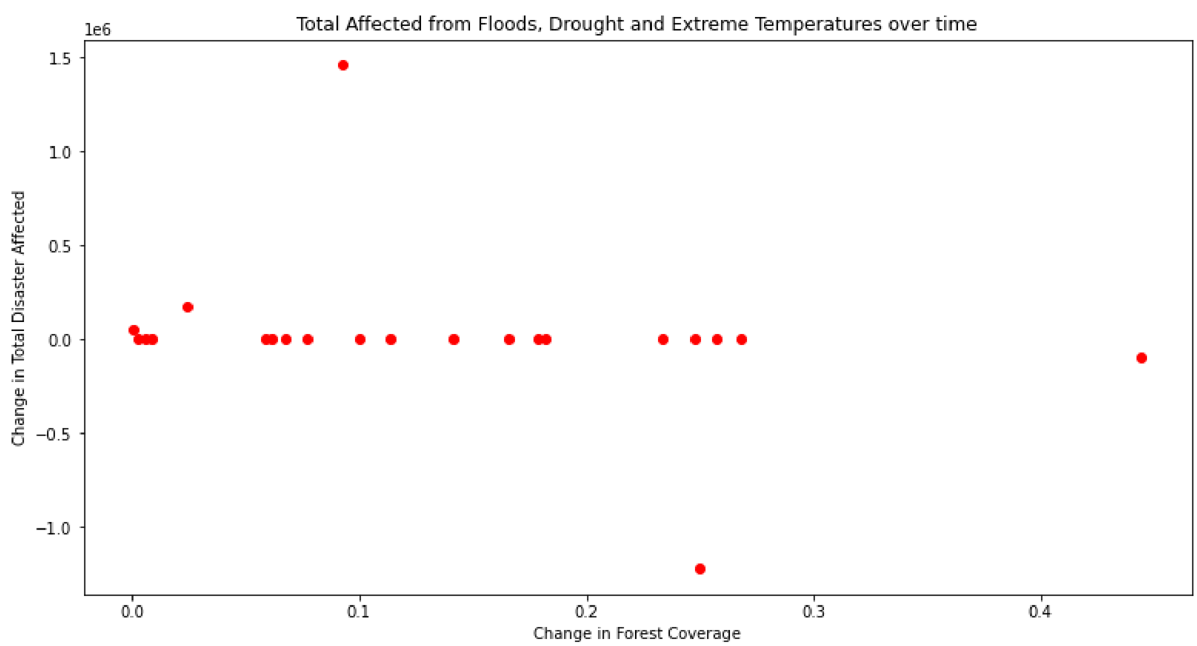
In [85]:
```
change.hist(figsize =(10,5));
```



Histograms of each variable show that in forest coverage (slopes_fc) a large portion of countries showed zero value meaning they experience no or little change in coverage over time, but a significant enough did either increase of decrease their forest coverage between 1990 and 1995.

Total disaster affected shows almost all countries experience no or little change over time with most values clustered right at/around zero. A further look at the two variables in a scatter plot should show any variance in the near zero data.

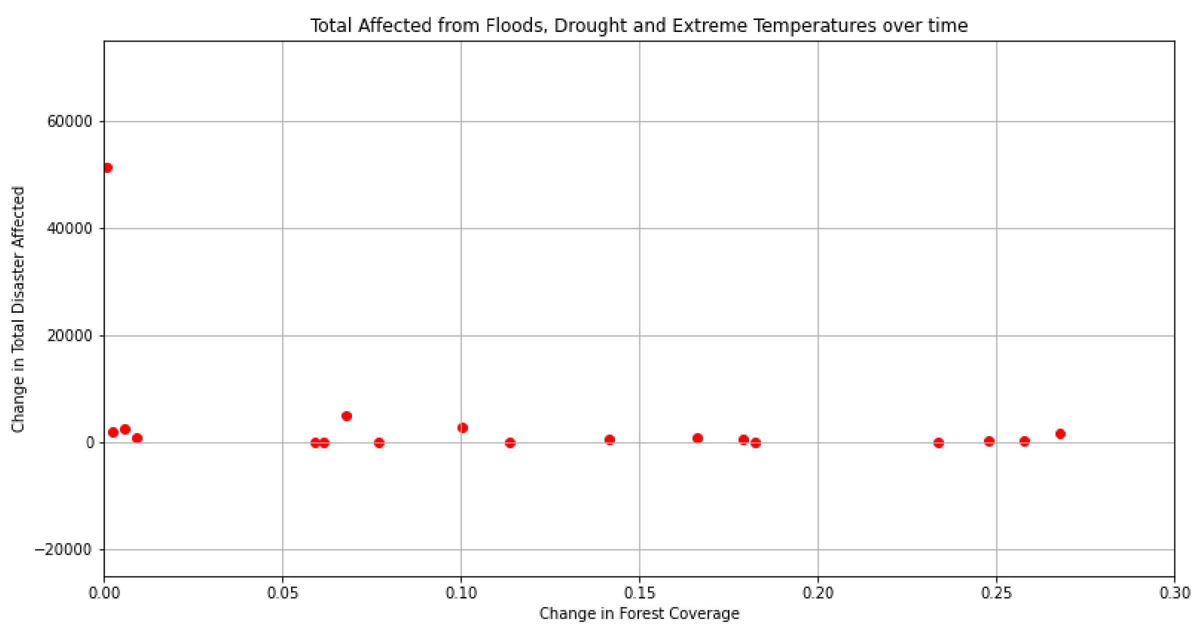
In [80]:
```
fig = plt.figure(figsize = (10,5))
ax =fig.add_axes([0,0,1,1])

greaterthan = change.query('slopes_fc > 0')


x = greaterthan.slopes_fc #change in forest coverage
y = greaterthan.slopes_td #change in total disaster affected

ax.scatter(x, y, color='r', label = 'Change')

#Labels & Legends
#plt.ylim(-100000, 100000)
ax.set_xlabel('Change in Forest Coverage')
ax.set_ylabel('Change in Total Disaster Affected')
ax.set_title('Total Affected from Floods, Drought and Extreme Temperatures ove
r time')


plt.show()
```



There are 3 major outliers that when plotted with the rest of the data flatten the results making it difficult to observe any trends that may be present.

I'll replot the data, but with restrictions on the x and y axis so that the graph only include the large grouping of data for visualization.

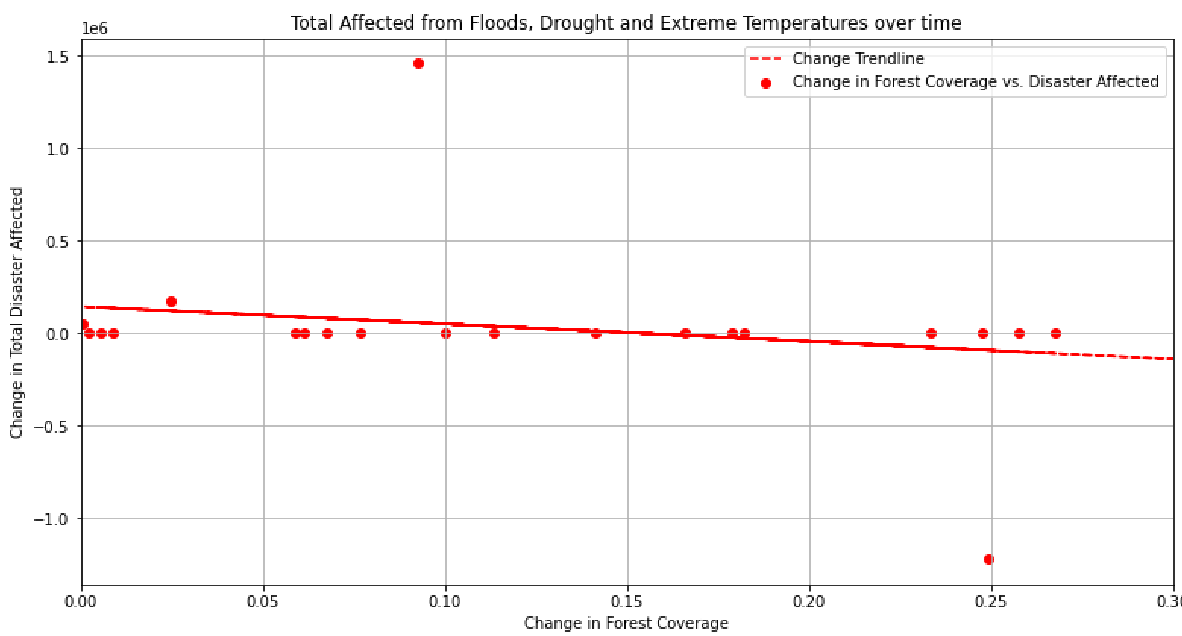
In [81]:
```
fig = plt.figure(figsize = (10,5))
ax =fig.add_axes([0,0,1,1])

greaterthan = change.query('slopes_fc > 0')


x = greaterthan.slopes_fc #change in forest coverage
y = greaterthan.slopes_td #change in total disaster affected

ax.scatter(x, y, color='r', label = 'Change')






#Labels & Legends
plt.xlim(0,.3)
plt.ylim(-25000, 75000)
plt.grid()
ax.set_xlabel('Change in Forest Coverage')
ax.set_ylabel('Change in Total Disaster Affected')
ax.set_title('Total Affected from Floods, Drought and Extreme Temperatures ove
r time')


plt.show()
```



With this view of the data minus the outliers, a trend is not apparent to show that increasing forest coverage over time coincides with a decrease in disaster affected.

In [82]:

```
fig = plt.figure(figsize = (10,5))
ax =fig.add_axes([0,0,1,1])

greaterthan = change.query('slopes_fc > 0')


x = greaterthan.slopes_fc #change in forest coverage
y = greaterthan.slopes_td #change in total disaster affected

ax.scatter(x, y, color='r', label = 'Change in Forest Coverage vs. Disaster Af
fected')

#adding a trendline
# plot the data itself
#ax.plot(x,y)

# calc the trendline
z = np.polyfit(x, y, 1)
p = np.poly1d(z)
ax.plot(x,p(x),"r--", label = 'Change Trendline')



#Labels & Legends
plt.xlim(0,.3)
#plt.ylim(-25000, 75000)
plt.grid()
ax.set_xlabel('Change in Forest Coverage')
ax.set_ylabel('Change in Total Disaster Affected')
ax.set_title('Total Affected from Floods, Drought and Extreme Temperatures ove
r time')
ax.legend()

plt.show()
```



A trendline plotted with all the available data shows that there is a slight trend. This trend shows that as forest coverage increases over time, there is a decrease in those affected by disaster.

Conclusion

In this project I looked at the effects of forest coverage on the number of people affected by natural disasters. With the data available I was limited to information on floods, droughts and extreme temperatures from a range of countries in years 1995-2008.

There were a few limitations on data available that need to be considered in determining any kind of causation or direct relationship. Since the total number of affected by natural disasters is the result of many factors, not just in the frequency of those disaster types, but government response, population density, natural geography, local climate, etc. this study would not be thorough enough to establish causation, but instead examine a correlation between forest coverage and disaster affected.

Also, there are a wide range of natural disasters but data in this study only looks at the three types for which relevant data is available, flood, drought and extreme temperatures. For a broader, more expansive study, other natural disaster types that would have a relationship with forest coverage would need to be considered such as those affected by wildfires, high winds, etc.

Laslty, while forest coverage and total population data was fairly complete for all countries, the data for each disaster was limited to certain countries, and when comparing data across all three natural disaster types, there was only enough data to compare 45 countries when considering comparisons of total disasters, or comparing disaster affected in each country by type.

With those limitations however, the analysis of the data available showed enough of a correlation in the data to answer the stated questions:

Do countries with a higher percentage of forest coverage have a lower number of affected from floods, droughts and extreme temperatures?

The comparison of each dataset for flood, drought and extreme temperature does not show a strong relationship between percentage of forest coverage and any natural disaster.

However, between the three datasets, flood and drought affected had a slight skew to the right. The scatter plot for these data sets show countries with higher percentage of forest coverage had slightly fewer affected per capita.

Which natural disaster type affects the most countries on a per/capita basis in countries with the lowest percentage of forest coverage?

For the 10 countries with the lowest forest coverage area, 50% are most affected by droughts on average 50% are most affected by floods.

Over time, do countries where forest coverage increases have decreaseing number of those affected by disasters?

The plotted information and trendliine showed that yes, increasing forest coverage over time resulted in fewer affected by disasters over time.

```
In [ ]:
```