Table of Contents

In [2]:
```
import pandas as pd
import numpy as np
import cartopy.crs as ccrs
from cartopy.mpl.ticker import LongitudeFormatter, LatitudeFormatter
import matplotlib.pyplot as plt
from scipy import stats
```

1. Significant earthquakes since 2150 B.C.

Total deaths

Compute the total number of deaths caused by earthquakes since 2150 B.C. in each country, and then print the top ten countries along with the total number of deaths.

In [3]:
```
# Read .tsv data using pd.read_csv
# Setting sep='\t'
ds = pd.read_csv('Data/PS2/earthquakes-2021-10-13_19-04-37_+0800.tsv',sep='\t',encoding='utf-8',)
```

In [4]:
```
ds.head()
```

Out[4]:

	Search Parameters	Year	Mo	Dy	Hr	Mn	Sec	Tsu	Vol	Country	...	Total Missing	Total Missing Description	Total Injuries	Total Injuries Description	Total Damage ($Mil)
0	[]	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
1	NaN	-2150.0	NaN	NaN	NaN	NaN	0.0	NaN	NaN	JORDAN	...	NaN	NaN	NaN	NaN	NaN
2	NaN	-2000.0	NaN	NaN	NaN	NaN	NaN	1.0	NaN	SYRIA	...	NaN	NaN	NaN	NaN	NaN
3	NaN	-2000.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	TURKMENISTAN	...	NaN	NaN	NaN	NaN	NaN
4	NaN	-1610.0	NaN	NaN	NaN	NaN	NaN	3.0	1351.0	GREECE	...	NaN	NaN	NaN	NaN	NaN

5 rows × 48 columns

In [5]:
```
country_group = ds.groupby(by='Country').sum()
country_group.sort_values(by='Deaths',na_position='last',ascending=False)['Deaths']
```

Out[5]:
```
Country
CHINA                                   2074900.0
TURKEY                                  1074769.0
IRAN                                    1011437.0
SYRIA                                    439224.0
ITALY                                    434863.0
                                             ...
PALAU                                         0.0
SAINT VINCENT AND THE GRENADINES              0.0
SAMOA                                         0.0
SAUDI ARABIA                                  0.0
ZAMBIA                                        0.0
Name: Deaths, Length: 156, dtype: float64
```

Total earthquakes (m>6)

Compute the total number of earthquakes with magnitude larger than 6.0 (use column Mag as the magnitude) worldwide each year, and then plot the time series. Do you observe any trend? Explain why or why not?

In [11]:
```
Earthquake_Year=pd.DataFrame({'Year':ds[ds.Mag>6].Year,'Count':np.ones(len(ds[ds.Mag>6]))})
Earthquake_Year.head()
```

Out[11]:

	Year	Count
1	-2150.0	1.0
3	-2000.0	1.0
8	-1250.0	1.0
9	-1050.0	1.0
15	-479.0	1.0

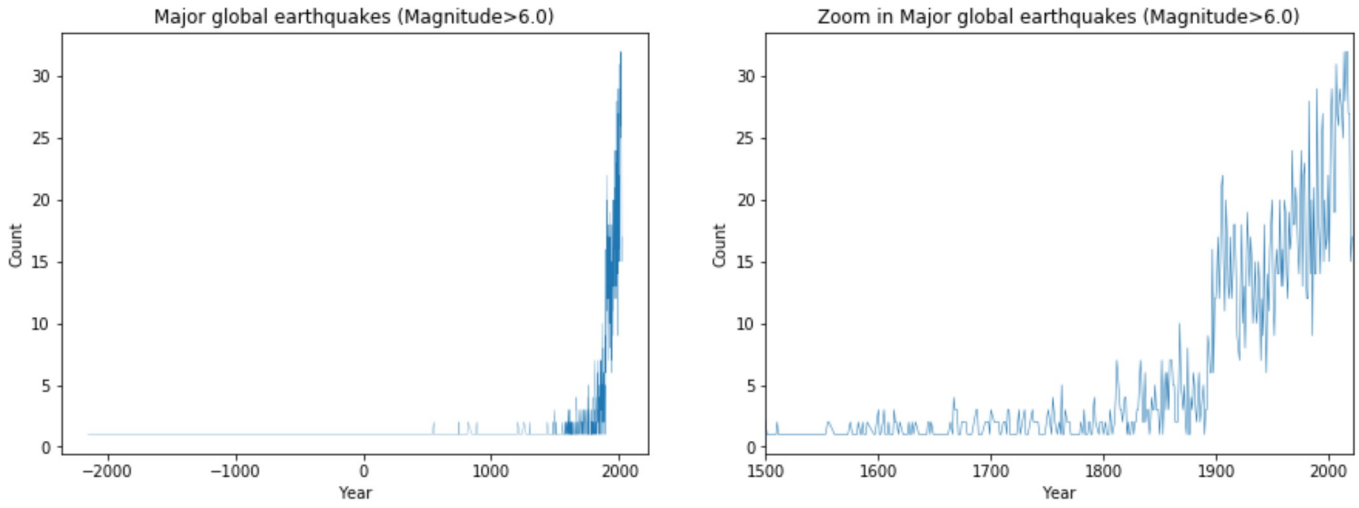
In [12]:
```
Earthquakes_group = Earthquake_Year.groupby(by='Year').sum()
Earthquakes_group
```

Out[12]:

	Count
Year	
-2150.0	1.0
-2000.0	1.0
-1250.0	1.0
-1050.0	1.0
-479.0	1.0
...	...
2017.0	32.0
2018.0	27.0
2019.0	27.0
2020.0	15.0
2021.0	17.0

530 rows × 1 columns

In [117]:
```
fig,ax = plt.subplots(1,2,figsize=(15,5))
ax[0].plot(Earthquakes_group,lw=0.25)
ax[0].set_title('Major global earthquakes (Magnitude>6.0)')
ax[1].plot(Earthquakes_group,lw=0.5)
ax[1].set_xlim(1500,2022)
ax[1].set_title('Zoom in Major global earthquakes (Magnitude>6.0)')
for a in ax:
    a.set_xlabel('Year')
    a.set_ylabel('Count')
```



There is a clear increasing trend in global major earthquakes (magnitude > 6.0). One main reason is the earthquake records are more in the recent days. The historical data is often deficient in available records.

CountEq_LargestEq

Write a function CountEq_LargestEq that returns both (1) the total number of earthquakes since 2150 B.C. in a given country AND (2) the date of the largest earthquake ever happened in this country. Apply CountEq_LargestEq to every country in the file, report your results in a descending order.

Give a country's name then return the largest eathquake occurrence time total earthquakes

In [215]:
```
def CountEq_LargestEq():
    country=input('Country Name (capitalize): ')
    data=pd.DataFrame({'Year':ds.Year,
                'Mo':ds.Mo,
                'Dy':ds.Dy,
                'Hr':ds.Hr,
                'Mn':ds.Mn,
                'Sec':ds.Sec,
                'Mag':ds.Mag,
                'Country':ds.Country,
                'Count':np.ones(len(ds))})
    data_max = data.groupby(by='Country').max()
    data_count = data.groupby(by='Country').sum()['Count']

    Eq_max_date= 'Largest Earthquake Occurrence Date: %4d-'%data_max.loc[country][0]+'%2d-'%data_max.lo
c[country][1]+'%2d '%data_max.loc[country][2]+'%2d:'%data_max.loc[country][3]+'%2d:'%data_max.loc[count
ry][4]+'%2d'%data_max.loc[country][5]
    Eq_tot_count='Total number of earthquakes since 2150 B.C.: %s'%data_count.loc[country]
    return Eq_max_date,Eq_tot_count
```

In [217]:
```
CountEq_LargestEq()
```

```
Country Name (capitalize): CHINA
```

Out[217]:
```
('Largest Earthquake Occurrence Date: 2021-12-31 23:59:59',
 'Total number of earthquakes since 2150 B.C.: 610.0')
```

Apply CountEq_LargestEq to every country

In [332]:
```
def CountEq_LargestEq():
    #country=input('Country Name (capitalize): ')
    data=pd.DataFrame({'Year':ds.Year,
                'Mo':ds.Mo,
                'Dy':ds.Dy,
                'Hr':ds.Hr,
                'Mn':ds.Mn,
                'Sec':ds.Sec,
                'Mag':ds.Mag,
                'Country':ds.Country,
                'Count':np.ones(len(ds))})
    global data_max
    data_max = data.groupby(by='Country').max()
    global data_count
    data_count = data.groupby(by='Country').sum()['Count']
    ds_out=pd.merge(data_max.iloc[:,0:4],pd.DataFrame({'sum':data_count}),right_index=True,left_index=T
rue)
    return ds_out
```

In [333]:
```
All_country=CountEq_LargestEq()
```

In [334]:
```
All_country.sort_values(by='Year',na_position='last',ascending=False)
```

Out[334]:

Country	Year	Mo	Dy	Hr	sum
HAITI	2021.0	12.0	29.0	21.0	17.0
PAKISTAN	2021.0	12.0	30.0	23.0	53.0
PERU	2021.0	12.0	31.0	23.0	185.0
PHILIPPINES	2021.0	12.0	31.0	23.0	221.0
CHINA	2021.0	12.0	31.0	23.0	610.0
...	...	...	...	...	...
NORWAY	1819.0	8.0	31.0	NaN	1.0
CANARY ISLANDS	1810.0	3.0	20.0	NaN	2.0
SIERRA LEONE	1795.0	5.0	20.0	22.0	1.0
NORTH KOREA	1727.0	8.0	31.0	NaN	6.0
IRELAND	1490.0	NaN	NaN	NaN	1.0

156 rows × 5 columns

"Year, Mo, Dy, Hr": the time when the largest earthquake ever happened in this country
"sum": the total number of earthquakes ever happened in this country since 2150 B.C.

A failed attempt: pd.to_datetime error

For example,

In [370]:
```
All_country.iloc[0,:3]
```

Out[370]:
```
Year    2018.0
Mo        12.0
Dy        31.0
Name: AFGHANISTAN, dtype: float64
```

In [371]:
```
dt=''
for dt_i in All_country.iloc[0,:3]:
    dt+=str(int(dt_i))
```

In [372]:
```
pd.to_datetime(dt)
```

Out[372]:
```
Timestamp('2018-12-31 00:00:00')
```

Changing time to datetime usually simplifies the data analysis

But the method pd.to_datetime cannot be used for the entire data

In [ ]:
```
dtime=[]
for i in range(len(data_max)):
    dt=''
    dt_i = data_max.iloc[i,:3].ravel()
    if (np.isnan(dt_i).any()==True):
        if (np.argwhere(np.isnan(dt_i))==0):
            dt=np.nan
        if (np.argwhere(np.isnan(dt_i))==1):
            dt=dt_i[0]
            dtime.append(pd.to_datetime(dt,format='%Y'))
        if (np.argwhere(np.isnan(dt_i))==2):
            dt=dt_i[0]+dt_i[1]
            dtime.append(pd.to_datetime(dt,format='%Y%m'))
    else:
        for m in dt_i:
            dt+=str(int(m))
        dtime.append(pd.to_datetime(dt,format='%Y%m%d'))
    print(dtime[i])
```

ERROR Here we actually get the Out of bounds error, because the method pd.to_datetime(args**),for "origin='unix'", the time origin is set to 1970-01-01

Wind speed in Shenzhen during the past 10 years

In [4]:
```
# Read .tsv data using pd.read_csv
ds = pd.read_csv('Data/PS2/2281305.csv')
```

```
/home/andrea/anaconda3/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3063: DtypeWarnin
g: Columns (4,8,9,12,15,21,22,24,26,31,33,34) have mixed types.Specify dtype option on import or set
low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

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

Out[5]:

	STATION	DATE	SOURCE	REPORT_TYPE	CALL_SIGN	QUALITY_CONTROL	AA1	AA2	AA3	AJ1	...	C
0	59493099999	2010-01-02T00:00:00	4	SY-MT	ZGSZ	V020	06,0000,2,1	24,0000,2,1	NaN	NaN	...	I
1	59493099999	2010-01-02T01:00:00	4	FM-15	ZGSZ	V020	NaN	NaN	NaN	NaN	...	I
2	59493099999	2010-01-02T02:00:00	4	FM-15	ZGSZ	V020	NaN	NaN	NaN	NaN	...	I
3	59493099999	2010-01-02T03:00:00	4	SY-MT	ZGSZ	V020	NaN	NaN	NaN	NaN	...	I
4	59493099999	2010-01-02T04:00:00	4	FM-15	ZGSZ	V020	NaN	NaN	NaN	NaN	...	I

5 rows × 43 columns

DATE to Datetime

In [6]:
```
DATE=ds.DATE.apply(lambda x:pd.to_datetime(x))
```

Split and merge WND

In [7]:
```
ds.WND
```

Out[7]:
```
0          040,1,N,0020,1
1          999,9,V,0010,1
2          999,9,C,0000,1
3          140,1,N,0010,1
4          300,1,N,0040,1
                 ...
111979     170,1,N,0030,1
111980     180,1,N,0040,1
111981     220,1,V,0030,1
111982     260,1,N,0030,1
111983     310,1,V,0020,1
Name: WND, Length: 111984, dtype: object
```

In [8]:
```
dfs=[ds.WND.apply(lambda x:int(x.split(',')[0])).to_frame(name='angel'),
ds.WND.apply(lambda x:int(x.split(',')[1])).to_frame(name='Q_A'),
ds.WND.apply(lambda x:x.split(',')[2]).to_frame(name='Type'),
ds.WND.apply(lambda x:float(x.split(',')[3])/10).to_frame(name='speed'),
ds.WND.apply(lambda x:int(x.split(',')[4])).to_frame(name='Q_S')]
```

In [9]:
```
WND = pd.merge(DATE,dfs[0],right_index=True,left_index=True)
for i in range(4):
    WND = pd.merge(WND,dfs[i+1],right_index=True,left_index=True)
```

In [10]:
```
WND=WND.set_index('DATE')
```

In [11]:
```
WND
```

Out[11]:

DATE	angel	Q_A	Type	speed	Q_S
2010-01-02 00:00:00	40	1	N	2.0	1
2010-01-02 01:00:00	999	9	V	1.0	1
2010-01-02 02:00:00	999	9	C	0.0	1
2010-01-02 03:00:00	140	1	N	1.0	1
2010-01-02 04:00:00	300	1	N	4.0	1
...	...	...	...	...	...
2020-09-11 17:00:00	170	1	N	3.0	1
2020-09-11 18:00:00	180	1	N	4.0	1
2020-09-11 19:00:00	220	1	V	3.0	1
2020-09-11 20:00:00	260	1	N	3.0	1
2020-09-11 21:00:00	310	1	V	2.0	1

111984 rows × 5 columns

Quality check

Q_S: quality code for WIND-OBSERVATION speed quality
1=Passed all quality control checks

In [12]:
```
WND_pass=WND[WND.Q_S==1]
WND_comp=WND_pass['speed'].replace(999.9,np.nan).dropna()
# Completed wind speed check
WND_comp
```

Out[12]:
```
DATE
2010-01-02 00:00:00    2.0
2010-01-02 01:00:00    1.0
2010-01-02 02:00:00    0.0
2010-01-02 03:00:00    1.0
2010-01-02 04:00:00    4.0
                      ...
2020-09-11 17:00:00    3.0
2020-09-11 18:00:00    4.0
2020-09-11 19:00:00    3.0
2020-09-11 20:00:00    3.0
2020-09-11 21:00:00    2.0
Name: speed, Length: 111345, dtype: float64
```

1. Calculate monthly averaged wind speed for each year

In [13]:
```
mon_WND=WND_comp.resample('M').mean()
```

In [27]:
```
mon_WND
```

Out[27]:
```
DATE
2010-01-31    2.756267
2010-02-28    3.388060
2010-03-31    3.360700
2010-04-30    3.191341
2010-05-31    3.293640
                 ...
2020-05-31    4.362198
2020-06-30    5.575800
2020-07-31    5.459140
2020-08-31    3.733608
2020-09-30    3.085019
Freq: M, Name: speed, Length: 129, dtype: float64
```

Trend analysis: linear regression

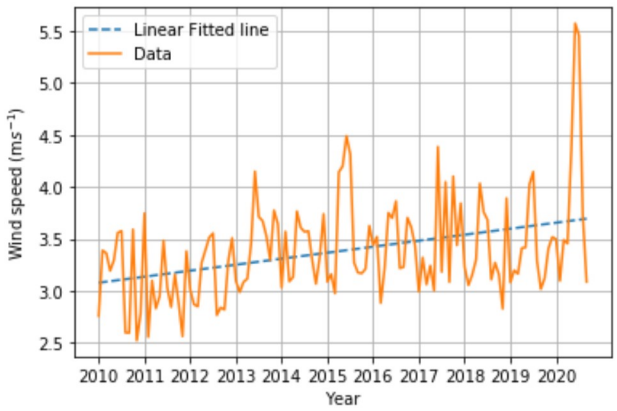
In [25]:
```
lin = stats.linregress(np.arange(len(mon_WND)),mon_WND.values)
lin
```

Out[25]:
```
LinregressResult(slope=0.0048177834642934465, intercept=3.0766291708500777, rvalue=0.367969901825728
2, pvalue=1.7853563483119417e-05, stderr=0.0010802897877188007)
```

In [45]:
```
fig,ax = plt.subplots()
f = lambda a,b,x: b+a*x
x=np.arange(len(mon_WND))

ax.plot(x,f(lin.slope,lin.intercept,x),'--',label='Linear Fitted line')
ax.plot(x,mon_WND,label='Data')
ax.grid()
ax.set_xticks(np.arange(0,len(mon_WND),12))
ax.set_xticklabels(np.arange(2010,2021))
ax.set_xlabel('Year')
ax.set_ylabel('Wind speed (m$s^{-1}$)')
ax.legend()
```

Out[45]:
```
<matplotlib.legend.Legend at 0x7f015da96790>
```



There is a weak increasing trend in monthly mean wind speed in Shenzhen for the past 10 years.

Explore a data set

Dataset: NOAA Integrated Surface Dataset
https://www.ncdc.noaa.gov/data-access/land-based-station-data/land-based-datasets (https://www.ncdc.noaa.gov/data-access/land-based-station-data/land-based-datasets)

Station: Los Angeles Downtown, CA USA
Variables:
AA1:Liquid precipitation (mm)
DEW:Dew point temperature (Celsius degree)
VIS:Visibility (meters)

In [46]:
```
# Read .tsv data using pd.read_csv
ds = pd.read_csv('Data/PS2/2749432.csv',)
ds.head()
```

```
/home/andrea/anaconda3/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3063: DtypeWarnin
g: Columns (6,12,13) have mixed types.Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[46]:

	STATION	NAME	LATITUDE	LONGITUDE	ELEVATION	DATE	SOURCE	REPORT_TYPE	CALL_SIGN	QUALITY_
0	72287493134	LOS ANGELES DOWNTOWN USC, CA US	34.0236	-118.2911	54.6	2000-01-01T00:00:00	5	FM-15	KCQT	
1	72287493134	LOS ANGELES DOWNTOWN USC, CA US	34.0236	-118.2911	54.6	2000-01-01T01:00:00	5	FM-15	KCQT	
2	72287493134	LOS ANGELES DOWNTOWN USC, CA US	34.0236	-118.2911	54.6	2000-01-01T02:00:00	5	FM-15	KCQT	
3	72287493134	LOS ANGELES DOWNTOWN USC, CA US	34.0236	-118.2911	54.6	2000-01-01T03:00:00	5	FM-15	KCQT	
4	72287493134	LOS ANGELES DOWNTOWN USC, CA US	34.0236	-118.2911	54.6	2000-01-01T04:00:00	5	FM-15	KCQT	

In [47]:
```
ds_sel = ds[['DATE','AA1','DEW','VIS']].dropna(axis=0)
ds_sel.head()
```

Out[47]:

	DATE	AA1	DEW	VIS
8929	2001-01-01T09:00:00	01,0000,9,5	+0122,5	002400,5,9,9
8930	2001-01-01T10:00:00	01,0000,9,5	+0122,5	002800,5,9,9
8931	2001-01-01T11:00:00	01,0000,9,5	+0117,5	002800,5,9,9
8932	2001-01-01T12:00:00	01,0000,9,5	+0111,5	003200,5,9,9
8933	2001-01-01T13:00:00	01,0000,9,5	+0094,5	004800,5,9,9

Convert and index data

In [49]:
```
DATE=ds_sel.DATE.apply(lambda x:pd.to_datetime(x)).to_frame(name='DATE')
```

In [50]:
```
LAX=DATE.copy()
```

In [51]:
```
VAR = ['AA1','DEW','VIS']
fields=['Hours','Depth','QP_1','QP_2','Dew','QD','VIS','QV']
```

In [52]:
```
#for var in VAR:
for a,b in zip([0,1],[2,3]):
    dfs=ds_sel[VAR[0]].apply(lambda x:int(x.split(',')[a])).to_frame(name=fields[a])
    LAX=pd.merge(LAX,dfs,right_index=True,left_index=True)

    dfs=ds_sel[VAR[0]].apply(lambda x:(x.split(',')[b])).to_frame(name=fields[b])
    LAX=pd.merge(LAX,dfs,right_index=True,left_index=True)

for x,y,z in zip(VAR[1:],[4,6],[5,7]):
    dfs=ds_sel[x].apply(lambda x:int(x.split(',')[0])).to_frame(name=fields[y])
    LAX=pd.merge(LAX,dfs,right_index=True,left_index=True)

    dfs=ds_sel[x].apply(lambda x:(x.split(',')[1])).to_frame(name=fields[z])
    LAX=pd.merge(LAX,dfs,right_index=True,left_index=True)
```

In [53]:
```
LAX = LAX.set_index(LAX.DATE)
LAX.head()
```

Out[53]:

DATE	DATE	Hours	QP_1	Depth	QP_2	Dew	QD	VIS	QV
2001-01-01 09:00:00	2001-01-01 09:00:00	1	9	0	5	122	5	2400	5
2001-01-01 10:00:00	2001-01-01 10:00:00	1	9	0	5	122	5	2800	5
2001-01-01 11:00:00	2001-01-01 11:00:00	1	9	0	5	117	5	2800	5
2001-01-01 12:00:00	2001-01-01 12:00:00	1	9	0	5	111	5	3200	5
2001-01-01 13:00:00	2001-01-01 13:00:00	1	9	0	5	94	5	4800	5

Quality check

QP_2: LIQUID-PRECIPITATION quality code
1=Passed all quality control checks
5=Passed all quality control checks, data originate from an NCEI data source

Filtering Method 1: data passed all quality checks

In [54]:
```
Depth_pass=LAX.where(LAX.QP_2.isin(['1','5']))
len(Depth_pass)
```

Out[54]: 197994

Filtering Method 2: drop all missing values

In [55]:
```
Depth_pass=LAX['Depth'].replace(9999,np.nan).dropna()
len(Depth_pass)
```

Out[55]: 197992

Resample hourly rainfall to daily data

In [56]:
```
Depth_day=Depth_pass.resample('D').sum()
Depth_day
```

Out[56]:
```
DATE
2001-01-01    0.0
2001-01-02    3.0
2001-01-03    0.0
2001-01-04    0.0
2001-01-05    0.0
              ...
2021-10-10    0.0
2021-10-11    0.0
2021-10-12    0.0
2021-10-13    0.0
2021-10-14    0.0
Freq: D, Name: Depth, Length: 7592, dtype: float64
```

Extreme rainfall (95th percentile)

Define extreme rainfall to be the 95th percentile of wet day precipitation
Define wet days as daily precipitation larger than 1.0 mm

In [57]:
```
Depth_wet=Depth_day[Depth_day>1]
```

Group by year

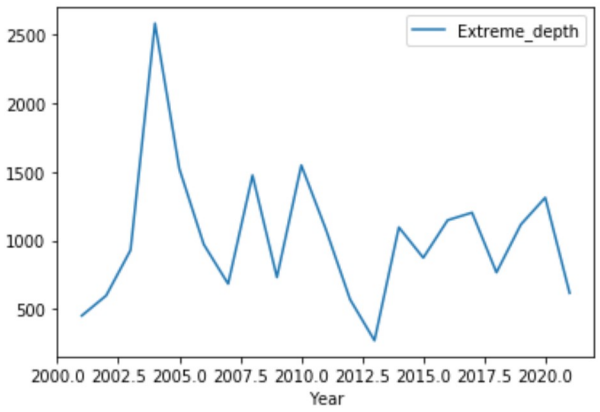
In [58]:
```
year_group=np.asarray(list(Depth_wet.groupby(Depth_wet.index.year))).T[1]
year_group[0].head()
```

Out[58]:
```
DATE
2001-01-02      3.0
2001-01-08     58.0
2001-01-09     11.0
2001-01-10     16.0
2001-01-11    958.0
Name: Depth, dtype: float64
```

In [72]:
```
extreme_wet_95=pd.DataFrame({
    'Year':np.arange(2001,2022),
    'Extreme_depth':list(map(lambda data:np.percentile(data,95),year_group)),
})
extreme_wet_95=extreme_wet_95.set_index('Year')
```

In [121]:
```
extreme_wet_95.plot()
```

Out[121]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f015605f1d0>
```

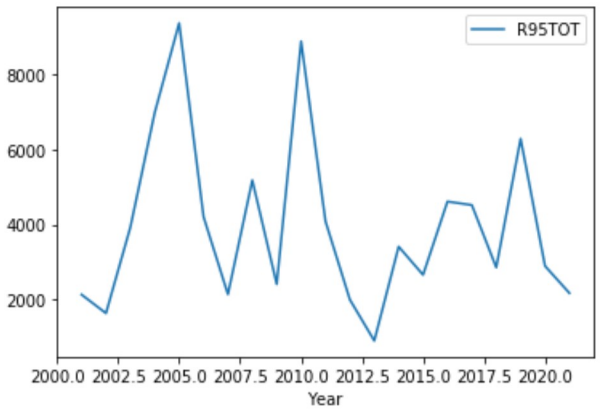


R95TOT: total amount of rainfall that is greater than 95th percentile rainfall baseline

In [133]:
```
R95TOT=pd.DataFrame({
    'Year':np.arange(2001,2022),
    'R95TOT':
    list(map(lambda data,base:data[data>float(base)].sum(),year_group,extreme_wet_95.values))
})
R95TOT=R95TOT.set_index('Year')
```

In [120]:
```
R95TOT.plot()
```

Out[120]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f01560d9610>
```



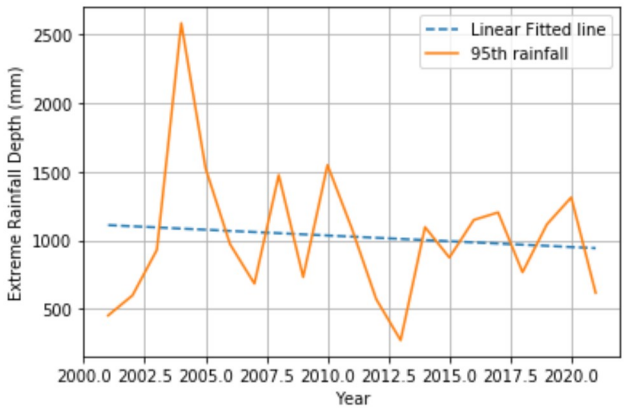
Extreme rainfall (95th percentile) trend

In [132]:
```
lin = stats.linregress(np.arange(2001,2022),extreme_wet_95['Extreme_depth'].values)

fig,ax = plt.subplots()
f = lambda a,b,x: b+a*x
x=np.arange(2001,2022)

ax.plot(x,f(lin.slope,lin.intercept,x),'--',label='Linear Fitted line')
ax.plot(x,extreme_wet_95['Extreme_depth'],label='95th rainfall')
ax.grid()
ax.set_xlabel('Year')
ax.set_ylabel('Extreme Rainfall Depth (mm)')
ax.legend()
```

Out[132]:
```
<matplotlib.legend.Legend at 0x7f015a9e1810>
```



Precipitation climatology mean for 2001-2011

Select data from 2001-2011

In [122]:
```
Depth_sel=Depth_day.where(Depth_day.index.year.isin(np.arange(2001,2012))).dropna()
Depth_sel.head()
```

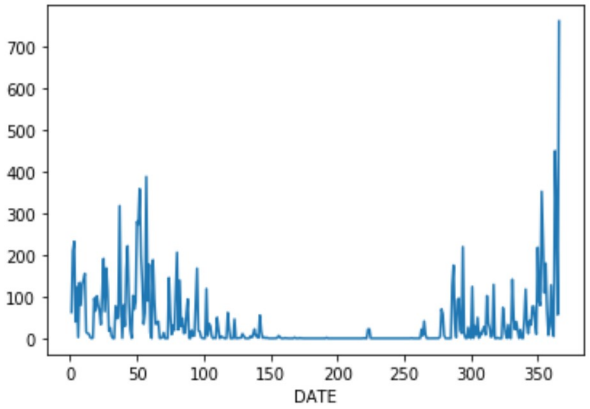
Out[122]:
```
DATE
2001-01-01    0.0
2001-01-02    3.0
2001-01-03    0.0
2001-01-04    0.0
2001-01-05    0.0
Freq: D, Name: Depth, dtype: float64
```

In [123]:
```
Depth_clim=Depth_sel.groupby(Depth_sel.index.dayofyear).mean()
Depth_clim
```

Out[123]:
```
DATE
1       62.636364
2      209.363636
3      232.454545
4       39.727273
5      122.727273
          ...
362      4.818182
363    450.272727
364    290.636364
365     56.909091
366    761.500000
Name: Depth, Length: 366, dtype: float64
```

In [130]:
```
Depth_clim.plot()
```

Out[130]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f015695c050>
```



Group and reshape data

In [137]:
```
year_group=np.asarray(list(Depth_day.groupby(Depth_day.index.year))).T[1]
year_group[0]
```

Out[137]:
```
DATE
2001-01-01      0.0
2001-01-02      3.0
2001-01-03      0.0
2001-01-04      0.0
2001-01-05      0.0
               ...
2001-12-27      0.0
2001-12-28      0.0
2001-12-29    101.0
2001-12-30     75.0
2001-12-31     54.0
Freq: D, Name: Depth, Length: 365, dtype: float64
```

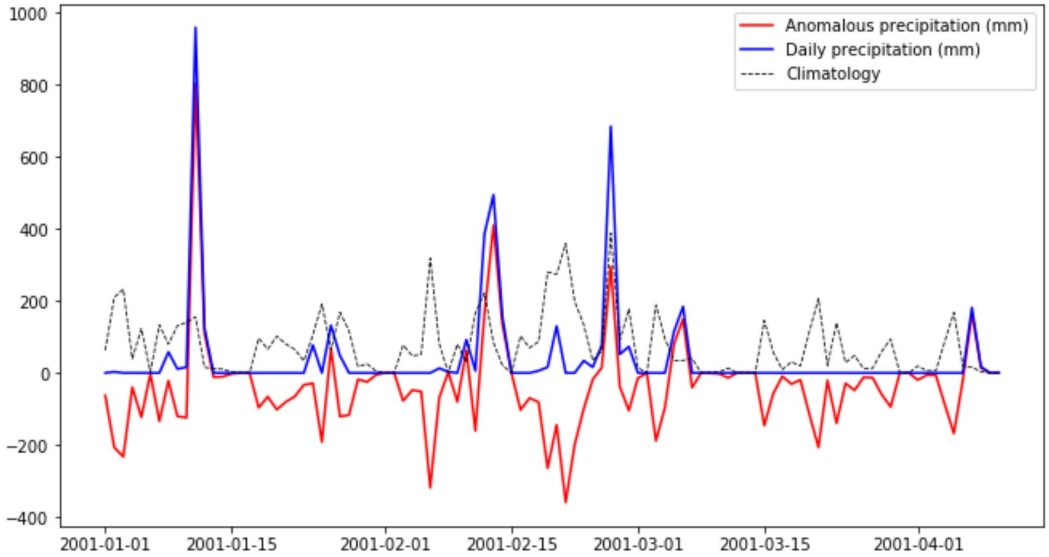
Precipitation anomaly relative to 2001-2011

In [138]:
```
Depth_anom=pd.concat(
    list(map(
        lambda data:data-Depth_clim.where(Depth_clim.index.isin(data.index.dayofyear)).dropna().values
        ,year_group)
        )
)
Depth_anom
```

Out[138]:
```
DATE
2001-01-01    -62.636364
2001-01-02   -206.363636
2001-01-03   -232.454545
2001-01-04    -39.727273
2001-01-05   -122.727273
                 ...
2021-10-10      0.000000
2021-10-11      0.000000
2021-10-12      0.000000
2021-10-13   -126.181818
2021-10-14   -175.272727
Freq: D, Name: Depth, Length: 7592, dtype: float64
```

In [139]:
```
fig,ax=plt.subplots(figsize=(11,6))
ax.plot(Depth_anom[:100],color='r',label='Anomalous precipitation (mm)')
ax.plot(Depth_day[:100],color='b',label='Daily precipitation (mm)')
ax.plot(Depth_anom.index[:100],Depth_clim.values[:100],lw=0.8,linestyle='--',color='k',label='Climatology')
plt.legend()
```

Out[139]:
```
<matplotlib.legend.Legend at 0x7f015aaccf90>
```



In [ ]: