Machine learning assignment

Covid-19 Outbreak Prediction

Dataset: https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset (https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset)

Objective: -to predict the spread of corona virus across the region for China, US and Malaysia -to analyses the growth rates and types of mitigation applied based on China, US and Malaysia

In [2]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels as sm
from sklearn.impute import SimpleImputer
sns.set()
%matplotlib inline
pd.options.plotting.backend
pd.plotting.register_matplotlib_converters()
```

Data cleaning

In this part of notebook, I take csv files covid_19_data

Clean data from covid_19_data.csv file and only select region for China, US and Malaysia

In [3]:

```
covid = pd.read_csv('covid_19_data.csv')
```

In [4]:

```
#see data
covid.head()
```

Out[4]:

	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Rec
0	1	01/22/2020	Anhui	Mainland China	1/22/2020 17:00	1.0	0.0	
1	2	01/22/2020	Beijing	Mainland China	1/22/2020 17:00	14.0	0.0	
2	3	01/22/2020	Chongqing	Mainland China	1/22/2020 17:00	6.0	0.0	
3	4	01/22/2020	Fujian	Mainland China	1/22/2020 17:00	1.0	0.0	
4	5	01/22/2020	Gansu	Mainland China	1/22/2020 17:00	0.0	0.0	

In [5]:

```
# data information
covid.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 172480 entries, 0 to 172479
Data columns (total 8 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   SNo              172480 non-null  int64  
 1   ObservationDate  172480 non-null  object 
 2   Province/State   124597 non-null  object 
 3   Country/Region   172480 non-null  object 
 4   Last Update      172480 non-null  object 
 5   Confirmed        172480 non-null  float64
 6   Deaths           172480 non-null  float64
 7   Recovered        172480 non-null  float64
dtypes: float64(3), int64(1), object(4)
memory usage: 10.5+ MB
```

In [6]:

```
# check if there exist a missing value
mis = covid.isnull().sum()
mis[mis>0]
```

Out[6]:

```
Province/State    47883
dtype: int64
```

In [8]:

```
#Only Province/State have a missing value. I can impute it because this variable is nec
essary for visualizing a data.

imputer = SimpleImputer(strategy='constant')#here I use constant because I cannot put a
nother Province/State
#that we do not know or that does not correspond to his country/region
impute_covid = pd.DataFrame(imputer.fit_transform(covid), columns=covid.columns)
impute_covid.head()
```

Out[8]:

	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Rec
0	1	01/22/2020	Anhui	Mainland China	1/22/2020 17:00	1	0	
1	2	01/22/2020	Beijing	Mainland China	1/22/2020 17:00	14	0	
2	3	01/22/2020	Chongqing	Mainland China	1/22/2020 17:00	6	0	
3	4	01/22/2020	Fujian	Mainland China	1/22/2020 17:00	1	0	
4	5	01/22/2020	Gansu	Mainland China	1/22/2020 17:00	0	0	

In [9]:

```
#convert ObservationDate and Last Update object to datetime
#convert confirmed, recovered, death to numeric
impute_covid['ObservationDate'] = pd.to_datetime(impute_covid['ObservationDate'])
impute_covid['Last Update'] = pd.to_datetime(impute_covid['Last Update'])
impute_covid['Confirmed'] = pd.to_numeric(impute_covid['Confirmed'], errors='coerce')
impute_covid['Recovered'] = pd.to_numeric(impute_covid['Recovered'], errors='coerce')
impute_covid['Deaths'] = pd.to_numeric(impute_covid['Deaths'], errors='coerce')
```

In [10]:

```
#check
#impute_covid.to_csv('covid_19_data_clean.csv', index=False)
impute_covid.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 172480 entries, 0 to 172479
Data columns (total 8 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   SNo              172480 non-null  object
 1   ObservationDate  172480 non-null  datetime64[ns]
 2   Province/State   172480 non-null  object
 3   Country/Region   172480 non-null  object
 4   Last Update      172480 non-null  datetime64[ns]
 5   Confirmed        172480 non-null  float64
 6   Deaths           172480 non-null  float64
 7   Recovered        172480 non-null  float64
dtypes: datetime64[ns](2), float64(3), object(3)
memory usage: 10.5+ MB
```

we are finishing data cleaning

Feature Statistics and Visualization impute_covid: we are going to visualize this data and make some statistics to find the relevant information.

In [11]:

```
# see again data table
impute_covid.head(3)
```

Out[11]:

	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Reco
0	1	2020-01-22	Anhui	Mainland China	2020-01-22 17:00:00	1.0	0.0	
1	2	2020-01-22	Beijing	Mainland China	2020-01-22 17:00:00	14.0	0.0	
2	3	2020-01-22	Chongqing	Mainland China	2020-01-22 17:00:00	6.0	0.0	

In [12]:

```
# we compute the active_confirmed
impute_covid['active_confirmed'] = impute_covid['Confirmed'].values - \
(impute_covid['Deaths'].values+impute_covid['Recovered'].values)
```

In [13]:

```
#check if all is ok
impute_covid.isnull().sum()[impute_covid.isnull().sum()>0]
```

Out[13]:

```
Series([], dtype: int64)
```

In [14]:

```
#ok we have no problem see table data
impute_covid.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 172480 entries, 0 to 172479
Data columns (total 9 columns):
 #   Column            Non-Null Count   Dtype         
---  ------            --------------   -----         
 0   SNo               172480 non-null  object        
 1   ObservationDate   172480 non-null  datetime64[ns]
 2   Province/State    172480 non-null  object        
 3   Country/Region    172480 non-null  object        
 4   Last Update       172480 non-null  datetime64[ns]
 5   Confirmed         172480 non-null  float64       
 6   Deaths            172480 non-null  float64       
 7   Recovered         172480 non-null  float64       
 8   active_confirmed  172480 non-null  float64       
dtypes: datetime64[ns](2), float64(4), object(3)
memory usage: 11.8+ MB
```

China

In [17]:

```
china = impute_covid[impute_covid['Country/Region'] == 'Mainland China']
chstar_date = china.ObservationDate.min()
chend_date = china.ObservationDate.max()
print('Novel covid-19 China:\n start date = {}\n end date = {}'.format(chstar_date, che
nd_date))
```

```
Novel covid-19 China:
 start date = 2020-01-22 00:00:00
 end date = 2020-12-06 00:00:00
```

In [18]:

```
lastChina = china[china['ObservationDate'] == chend_date]
lastChina.head()
```

Out[18]:

	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Death
171912	171913	2020-12-06	Anhui	Mainland China	2020-12-07 05:26:14	992.0	6
171946	171947	2020-12-06	Beijing	Mainland China	2020-12-07 05:26:14	952.0	9
171996	171997	2020-12-06	Chongqing	Mainland China	2020-12-07 05:26:14	590.0	6
172038	172039	2020-12-06	Fujian	Mainland China	2020-12-07 05:26:14	500.0	1
172043	172044	2020-12-06	Gansu	Mainland China	2020-12-07 05:26:14	182.0	2

In [19]:

```
print('================ China report ====================================')
print('== Information to {} on novel COVID-19 =========\n'.format(chend_date))
print('Tota confirmed: {}\nTotal Deaths: {}\nTotal Recovered: {}\nTotal active confirme
d: {}\n'.format(\
lastChina.Confirmed.sum(), lastChina.Deaths.sum(), lastChina.Recovered.sum(), lastChina
.active_confirmed.sum()))
print('================================================================')
```

```
================ China report ====================================
== Information to 2020-12-06 00:00:00 on novel COVID-19 =========

Tota confirmed: 86634.0
Total Deaths: 4634.0
Total Recovered: 81718.0
Total active confirmed: 282.0

================================================================
```

In [20]:

```
lastChina[['Province/State', 'Confirmed', 'Deaths', 'Recovered', 'active_confirmed']].s
tyle.\
background_gradient(cmap='viridis')
```

Out[20]:

	Province/State	Confirmed	Deaths	Recovered	active_confirmed
171912	Anhui	992.000000	6.000000	986.000000	0.000000
171946	Beijing	952.000000	9.000000	939.000000	4.000000
171996	Chongqing	590.000000	6.000000	583.000000	1.000000
172038	Fujian	500.000000	1.000000	453.000000	46.000000
172043	Gansu	182.000000	2.000000	180.000000	0.000000
172061	Guangdong	2004.000000	8.000000	1960.000000	36.000000
172062	Guangxi	263.000000	2.000000	260.000000	1.000000
172065	Guizhou	147.000000	2.000000	145.000000	0.000000
172068	Hainan	171.000000	6.000000	165.000000	0.000000
172074	Hebei	373.000000	6.000000	367.000000	0.000000
172075	Heilongjiang	949.000000	13.000000	936.000000	0.000000
172076	Henan	1295.000000	22.000000	1266.000000	7.000000
172085	Hubei	68149.000000	4512.000000	63633.000000	4.000000
172087	Hunan	1020.000000	4.000000	1016.000000	0.000000
172095	Inner Mongolia	336.000000	1.000000	308.000000	27.000000
172109	Jiangsu	680.000000	0.000000	674.000000	6.000000
172110	Jiangxi	935.000000	1.000000	934.000000	0.000000
172111	Jilin	157.000000	2.000000	155.000000	0.000000
172159	Liaoning	289.000000	2.000000	287.000000	0.000000
172243	Ningxia	75.000000	0.000000	75.000000	0.000000
172308	Qinghai	18.000000	0.000000	18.000000	0.000000
172353	Shaanxi	502.000000	3.000000	478.000000	21.000000
172354	Shandong	857.000000	7.000000	841.000000	9.000000
172355	Shanghai	1366.000000	7.000000	1294.000000	65.000000
172356	Shanxi	222.000000	0.000000	218.000000	4.000000
172360	Sichuan	812.000000	3.000000	779.000000	30.000000
172392	Tianjin	301.000000	3.000000	292.000000	6.000000
172393	Tibet	1.000000	0.000000	1.000000	0.000000
172463	Xinjiang	980.000000	3.000000	977.000000	0.000000
172471	Yunnan	221.000000	2.000000	210.000000	9.000000
172477	Zhejiang	1295.000000	1.000000	1288.000000	6.000000

United State

In [21]:

```
us = impute_covid[impute_covid['Country/Region'] == 'US']
usstar_date = us.ObservationDate.min()
usend_date = us.ObservationDate.max()
print('Novel covid-19 US:\n start date = {}\n end date = {}'.format(usstar_date, usend_
date))
```

```
Novel covid-19 US:
 start date = 2020-01-22 00:00:00
 end date = 2020-12-06 00:00:00
```

In [23]:

```
lastUs = us[us['ObservationDate'] == usend_date]
lastUs.head()
```

Out[23]:

	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deatl
171896	171897	2020-12-06	Alabama	US	2020-12-07 05:26:14	269877.0	3889
171898	171899	2020-12-06	Alaska	US	2020-12-07 05:26:14	37036.0	143
171923	171924	2020-12-06	Arizona	US	2020-12-07 05:26:14	364276.0	6950
171924	171925	2020-12-06	Arkansas	US	2020-12-07 05:26:14	170924.0	2660
171967	171968	2020-12-06	California	US	2020-12-07 05:26:14	1366673.0	19928

In [24]:

```
print('================ US report ==================================')
print('== Information to {} on novel COVID-19 =========\n'.format(usend_date))
print('Total confirmed: {}\nTotal Deaths: {}\nTotal Recovered: {}\nTotal active confirm
ed: {}\n'.format(\
lastUs.Confirmed.sum(), lastUs.Deaths.sum(), lastUs.Recovered.sum(), lastUs.active_conf
irmed.sum()))
print('================================================================')
```

```
================ US report ==================================
== Information to 2020-12-06 00:00:00 on novel COVID-19 =========

Total confirmed: 14757000.0
Total Deaths: 282299.0
Total Recovered: 5624444.0
Total active confirmed: 8850257.0

================================================================
```

In [25]:

```
lastUs[['Province/State', 'Confirmed', 'Deaths', 'Recovered', 'active_confirmed']].styl
e.\
background_gradient(cmap='viridis')
```

Out[25]:

	Province/State	Confirmed	Deaths	Recovered	active_confirmed
171896	Alabama	269877.000000	3889.000000	0.000000	265988.000000
171898	Alaska	37036.000000	143.000000	0.000000	36893.000000
171923	Arizona	364276.000000	6950.000000	0.000000	357326.000000
171924	Arkansas	170924.000000	2660.000000	0.000000	168264.000000
171967	California	1366673.000000	19928.000000	0.000000	1346745.000000
172002	Colorado	260581.000000	3356.000000	0.000000	257225.000000
172003	Connecticut	127715.000000	5146.000000	0.000000	122569.000000
172013	Delaware	39912.000000	793.000000	0.000000	39119.000000
172016	Diamond Princess cruise ship	49.000000	0.000000	0.000000	49.000000
172017	District of Columbia	23136.000000	697.000000	0.000000	22439.000000
172033	Florida	1058074.000000	19177.000000	0.000000	1038897.000000
172046	Georgia	501405.000000	9806.000000	0.000000	491599.000000
172054	Grand Princess	103.000000	3.000000	0.000000	100.000000
172059	Guam	6959.000000	113.000000	0.000000	6846.000000
172073	Hawaii	18842.000000	262.000000	0.000000	18580.000000
172091	Idaho	110510.000000	1035.000000	0.000000	109475.000000
172092	Illinois	787573.000000	14116.000000	0.000000	773457.000000
172093	Indiana	381617.000000	6242.000000	0.000000	375375.000000
172096	Iowa	244691.000000	2717.000000	0.000000	241974.000000
172123	Kansas	171364.000000	1786.000000	0.000000	169578.000000
172128	Kentucky	200631.000000	2072.000000	0.000000	198559.000000
172170	Louisiana	251123.000000	6584.000000	0.000000	244539.000000
172182	Maine	13348.000000	227.000000	0.000000	13121.000000
172189	Maryland	215027.000000	4846.000000	0.000000	210181.000000
172190	Massachusetts	256844.000000	11004.000000	0.000000	245840.000000
172201	Michigan	426576.000000	10321.000000	0.000000	416255.000000
172205	Minnesota	350862.000000	4043.000000	0.000000	346819.000000
172206	Mississippi	164931.000000	3961.000000	0.000000	160970.000000
172207	Missouri	329420.000000	4284.000000	0.000000	325136.000000
172212	Montana	67875.000000	736.000000	0.000000	67139.000000
172230	Nebraska	139834.000000	1205.000000	0.000000	138629.000000
172232	Nevada	168140.000000	2315.000000	0.000000	165825.000000
172235	New Hampshire	24888.000000	564.000000	0.000000	24324.000000
172236	New Jersey	368016.000000	17321.000000	0.000000	350695.000000
172237	New Mexico	108088.000000	1749.000000	0.000000	106339.000000
172239	New York	705827.000000	34958.000000	0.000000	670869.000000

	Province/State	Confirmed	Deaths	Recovered	active_confirmed
172250	North Carolina	394990.000000	5543.000000	0.000000	389447.000000
172251	North Dakota	82981.000000	1019.000000	0.000000	81962.000000
172254	Northern Mariana Islands	109.000000	2.000000	0.000000	107.000000
172267	Ohio	475024.000000	6959.000000	0.000000	468065.000000
172271	Oklahoma	216486.000000	1896.000000	0.000000	214590.000000
172275	Oregon	84496.000000	1033.000000	0.000000	83463.000000
172288	Pennsylvania	423100.000000	11255.000000	0.000000	411845.000000
172302	Puerto Rico	56671.000000	1192.000000	0.000000	55479.000000
172315	Recovered	0.000000	0.000000	5624444.000000	-5624444.000000
172319	Rhode Island	62137.000000	1413.000000	0.000000	60724.000000
172371	South Carolina	232099.000000	4566.000000	0.000000	227533.000000
172372	South Dakota	85991.000000	1110.000000	0.000000	84881.000000
172388	Tennessee	400594.000000	4943.000000	0.000000	395651.000000
172390	Texas	1322738.000000	23137.000000	0.000000	1299601.000000
172427	Utah	215407.000000	939.000000	0.000000	214468.000000
172442	Vermont	5015.000000	79.000000	0.000000	4936.000000
172446	Virgin Islands	1633.000000	23.000000	0.000000	1610.000000
172447	Virginia	255053.000000	4200.000000	0.000000	250853.000000
172456	Washington	177447.000000	2925.000000	0.000000	174522.000000
172459	West Virginia	54997.000000	838.000000	0.000000	54159.000000
172461	Wisconsin	441067.000000	3952.000000	0.000000	437115.000000
172462	Wyoming	36218.000000	266.000000	0.000000	35952.000000

Malaysia

In [26]:

```
my = impute_covid[impute_covid['Country/Region'] == 'Malaysia']
mystar_date = my.ObservationDate.min()
myend_date = my.ObservationDate.max()
print('Novel covid-19 Malaysia:\n start date = {}\n end date = {}'.format(mystar_date,
myend_date))
```

```
Novel covid-19 Malaysia:
 start date = 2020-01-23 00:00:00
 end date = 2020-12-06 00:00:00
```

In [28]:

```
lastMy = my[my['ObservationDate'] == myend_date]
lastMy.head()
```

Out[28]:

	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Death
171815	171816	2020-12-06	missing_value	Malaysia	2020-12-07 05:26:14	72694.0	382

In [29]:

```
print('================ Malaysia report ==================================')
print('== Information to {} on novel COVID-19 =========\n'.format(myend_date))
print('Total confirmed: {}\nTotal Deaths: {}\nTotal Recovered: {}\nTotal active confirm
ed: {}\n'.format(\
lastMy.Confirmed.sum(), lastMy.Deaths.sum(), lastMy.Recovered.sum(), lastMy.active_conf
irmed.sum()))
print('==================================================================')
```

```
================ Malaysia report ==================================
== Information to 2020-12-06 00:00:00 on novel COVID-19 =========

Total confirmed: 72694.0
Total Deaths: 382.0
Total Recovered: 61273.0
Total active confirmed: 11039.0

==================================================================
```

In [30]:

```
lastMy[['Province/State', 'Confirmed', 'Deaths', 'Recovered', 'active_confirmed']].styl
e.\
background_gradient(cmap='viridis')
```

Out[30]:

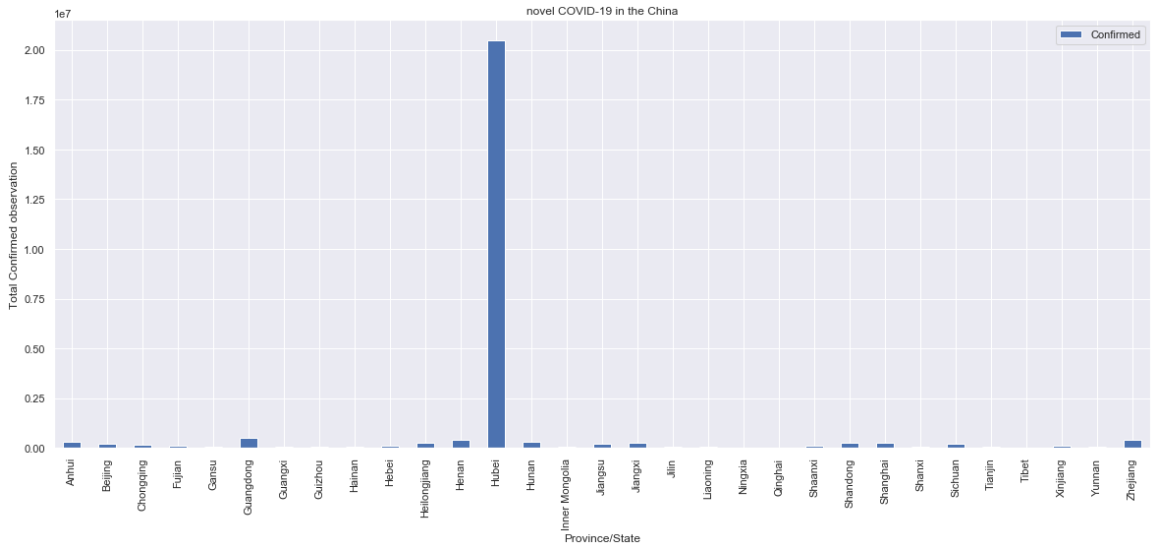
	Province/State	Confirmed	Deaths	Recovered	active_confirmed
171815	missing_value	72694.000000	382.000000	61273.000000	11039.000000

In [89]:

```
time_obs = china.groupby('Province/State')['Confirmed'].aggregate([np.sum])
time_obs.columns = ['Confirmed']
time_obs.plot(figsize=(20,8), title='novel COVID-19 in the China', kind='bar')
plt.ylabel('Total Confirmed observation')
```

Out[89]:

```
Text(0, 0.5, 'Total Confirmed observation')
```



1) Predict the spread of corona virus across the region for China, US and Malaysia using Linear Regression

In [95]:

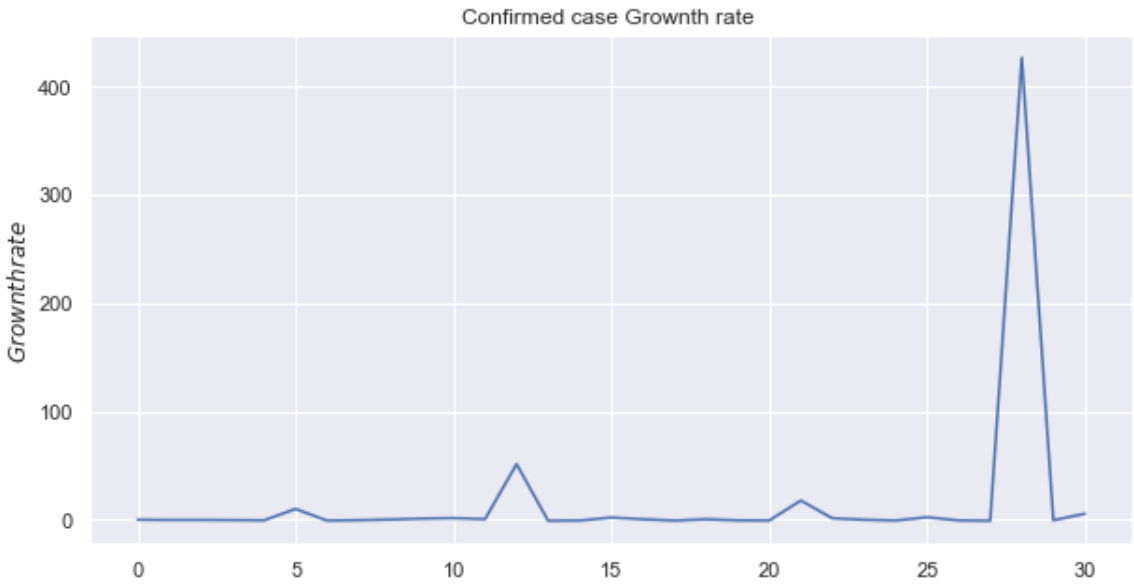
```
x = []
x.append(0)
for i in range(time_obs.shape[0]-1):
    a = time_obs.iloc[i+1,0]-time_obs.iloc[i,0]
    x.append(a/time_obs.iloc[i,0])

grown_rate = time_obs.reset_index()
grown_rate['grownRate'] = x
grown_rate.head()

grown_rate.grownRate.plot(figsize=(10,5))
plt.title('Confirmed case Grownth rate ')
plt.ylabel('$Grownth rate$')
```

Out[95]:

```
Text(0, 0.5, '$Grownth rate$')
```



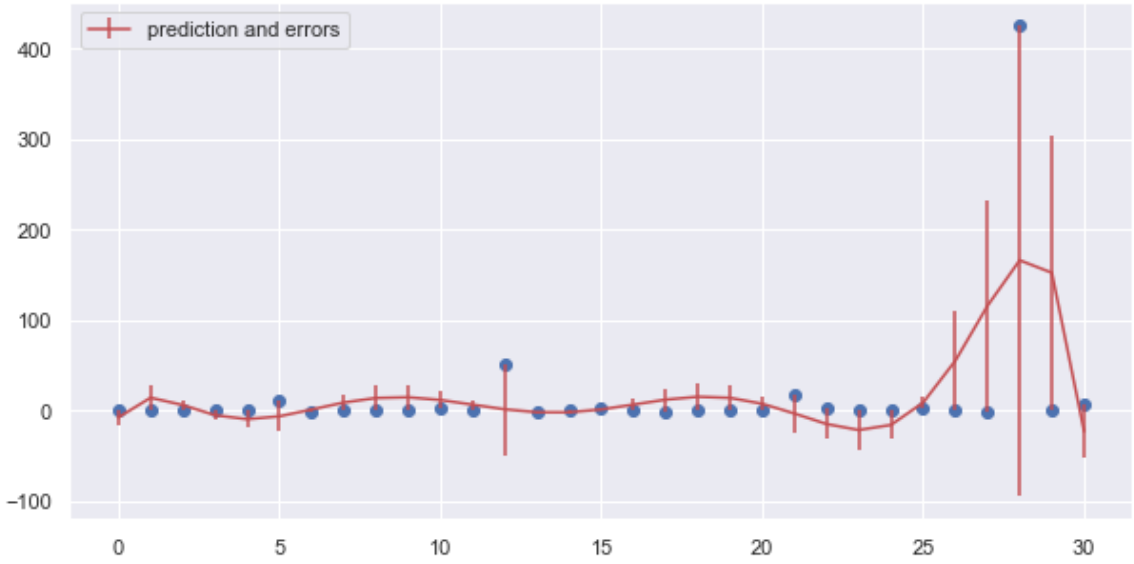
Linear Regression

In [93]:

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split

trend_model = make_pipeline(PolynomialFeatures(8), LinearRegression(normalize=True, fit
_intercept=True))
trend_model.fit(np.array(grown_rate.index).reshape((-1,1)), grown_rate['grownRate'])

dt =np.array(grown_rate.index).reshape((-1,1))
fit_grown = trend_model.predict(dt)
errors = grown_rate['grownRate'] - fit_grown
upperlimits = [True, False]
lowerlimits = [False, True]
plt.figure(figsize=(10,5))
plt.scatter(dt, grown_rate['grownRate'])
plt.errorbar(dt, fit_grown,yerr = errors,  color='r', label='prediction and errors')
plt.legend(loc='best')
plt.show()
```



In [ ]:

```

```