CovidInCanadaAnalysis

April 25, 2020

```
In [678]: import numpy as np
          # %matplotlib inline
          import matplotlib.pyplot as plt
          from scipy.optimize import curve_fit
          from scipy import optimize
          import pandas as pd
          import seaborn as sns
          from datetime import datetime
          from datetime import timedelta
          # import datetime

In [406]: def givemedata(df, prov):
              df_temp=df[df["Province/State"]==prov]
              array = df_temp.to_numpy()
              return array[0][5:]D
          def givemedata_c(df, prov):
              df_temp=df[df["Country/Region"]==prov]
              array = df_temp.to_numpy()
              return array[0][5:]

In [436]: #CanadaFromCanada
          def CanadaData(data, prov):
              df=data
              df_temp=df[df["prname"]==prov]
              cases=df_temp["numconf"].to_numpy()
              # dt_temp_dates=df_temp["date"].to_numpy()
              return cases

          def CanadaDatatests(data, prov):
              df=data
              df_temp=df[df["prname"]==prov]
              cases=df_temp["numtested"].to_numpy()
              # dt_temp_dates=df_temp["date"].to_numpy()
              return cases
          #     return np.nan_to_num(cases)

          def CanadaDataDeaths(data, prov):
```

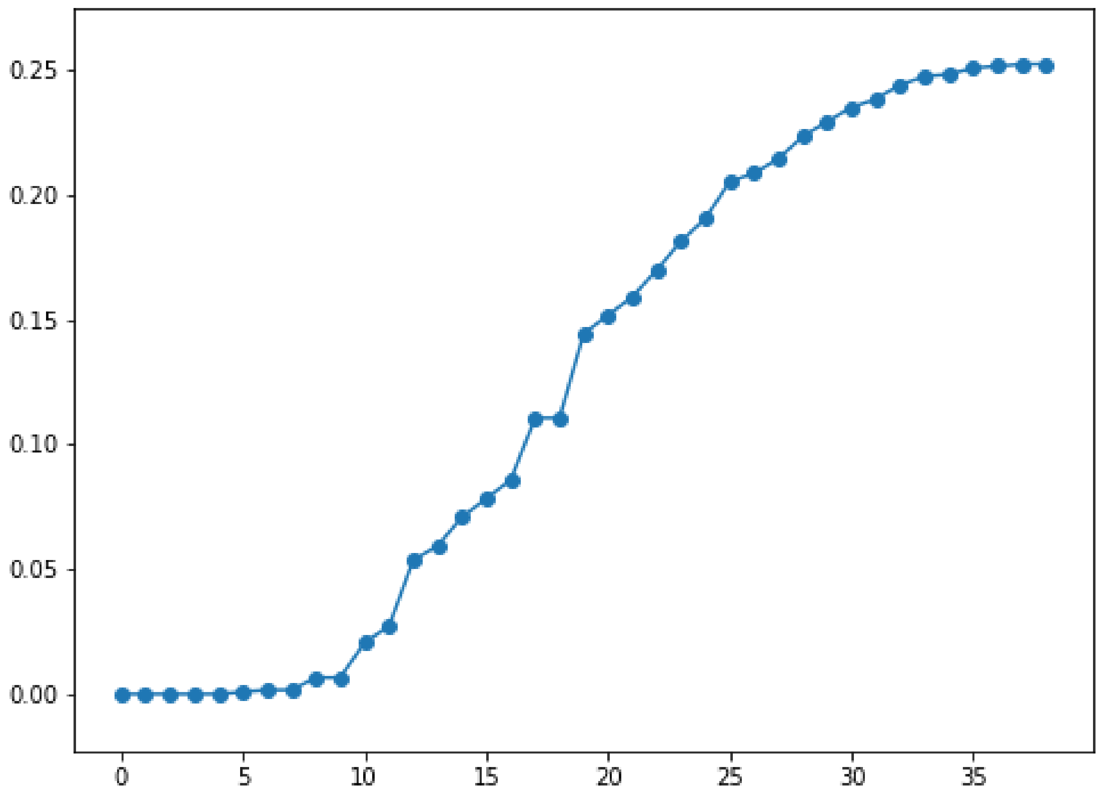
```
        df=data
        df_temp=df[df["prname"]==prov]
        cases=df_temp["numdeaths"].to_numpy()
        # dt_temp_dates=df_temp["date"].to_numpy()
        return cases
    #     return np.nan_to_num(cases)
```

```
In [408]: def func(x, L, k, M):
              return ( L /( 1 + np.exp(-1 * k * ( x - M ))))
```

```
In [409]: def Gauss(x, a, x0, sigma):
              return a * np.exp(-(x - x0)**2 / (2 * sigma**2))

          def exponenial_func(x, a, b, c):
              return a*np.exp(-b*x)+c
          def Linefunc(x, m, b):
              return m*x+b
```

```
In [410]: rawData="https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid
          data = pd.read_csv(rawData)
          # data
          dataName="British Columbia"
          dataName="Ontario"
          # ont=givemedata(data,"Ontario")
          ont=givemedata(data,dataName)
          # ont



          y_data = ont[0:-1].astype(float)
          # y_data = ont.astype(float)
          # y_data = np.insert(y_data,len(y_data), 4726) # -----------adding new data
          x_data = np.arange(len(y_data)).astype(float)
```

```
In [550]: # And plot it

          plt.figure(figsize=(8, 6))
          plt.scatter(x_data, y_data)
          plt.plot(np.arange(len(y_data)),y_data)

Out[550]: [<matplotlib.lines.Line2D at 0x1a30462710>]
```



```
In [413]: # #Let fit it bb
          # params, params_covariance = optimize.minimize(func, x_data, y_data,
          #                                                p0=[4.451E7,0.1893,121.9])

          # print(params)
```

```
In [414]: #lets goof on fitting

          #Let fit it bb
          print ("dates: ",x_data)
          print ("cases: ",y_data)
          params, params_covariance = optimize.curve_fit(func, x_data, y_data,
                                                        p0=[4.451E5,0.1893,121.9],maxfev=9000
          
          # y_data
          sigma = np.sqrt(np.diag(params_covariance))
          print("-----")
          print("params PY: ",params)
          print("sigma PY: ", sigma)
```

```
dates:  [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16. 17.
 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35.
 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53.
 54. 55. 56. 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70. 71.
 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83.]
cases:  [0.000e+00 0.000e+00 0.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
 1.000e+00 3.000e+00 3.000e+00 3.000e+00 3.000e+00 3.000e+00 3.000e+00
 3.000e+00 3.000e+00 3.000e+00 3.000e+00 3.000e+00 3.000e+00 3.000e+00
 3.000e+00 3.000e+00 3.000e+00 3.000e+00 3.000e+00 3.000e+00 3.000e+00
 3.000e+00 3.000e+00 3.000e+00 3.000e+00 4.000e+00 4.000e+00 4.000e+00
 6.000e+00 6.000e+00 1.100e+01 1.500e+01 1.800e+01 2.000e+01 2.000e+01
 2.200e+01 2.500e+01 2.800e+01 2.900e+01 3.400e+01 3.600e+01 4.100e+01
 4.200e+01 7.400e+01 7.900e+01 1.040e+02 1.770e+02 1.850e+02 2.210e+02
 2.570e+02 3.080e+02 3.770e+02 4.250e+02 5.030e+02 5.880e+02 6.880e+02
 8.580e+02 9.940e+02 1.144e+03 1.355e+03 1.706e+03 1.966e+03 2.392e+03
 2.793e+03 3.255e+03 3.630e+03 4.354e+03 4.347e+03 4.726e+03 5.276e+03
 5.759e+03 6.237e+03 6.648e+03 7.049e+03 7.470e+03 7.953e+03 8.447e+03]
-----
params PY:  [1.02531999e+04 1.87092078e-01 7.55084100e+01]
sigma PY:  [1.65517145e+02 3.10405089e-03 2.09387957e-01]
```

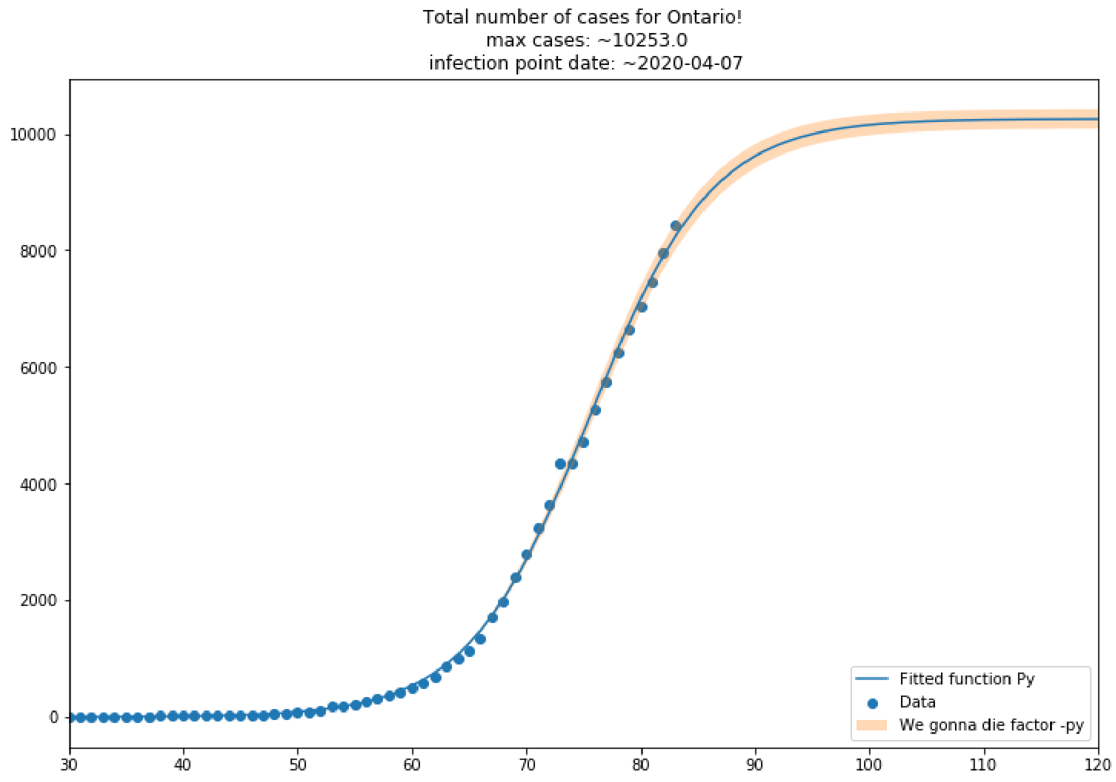
```
In [415]: # Auto Fit for: Ontraio | Number of Cases
          # y = L/(1+exp(-K*(x-M)))
          # L: 35183216.3 +/- 128.459930
          # K: 0.174521667 +/- 0.00105769512
          # M: 125.026843 +/- 0.348430865
          # Correlation: 0.999631150
          # RMSE: 13.1903203



          # paramsLOGGER=[35183216.3, 0.174521667, 124.026843]
          # sigmaLOGGER=[128.459930, 0.00105769512, 0.348430865]
          # print("params LP: ",paramsLOGGER)
          # print("sigma LP: ", sigmaLOGGER)
          # print("-----")
          # print("delta_param: ", params-paramsLOGGER)
          # print("delta_sigma: ", sigma-sigmaLOGGER)
```

```
In [416]: plt.figure(figsize=(12, 8))
          plt.scatter(x_data, y_data, label='Data')



          x_pred=np.arange(0,len(y_data)+55).astype(float)
          x_future=np.arange(0,len(y_data)+55).astype(float)
          days=0
          daysLP=days
```

```
SM=3

# print("latest data is for day: " +str(len(y_data))+ ","+ str(len(y_data)+days))
# print("sigma+: "+ str(round(func(len(y_data)+daysLP, paramsLOGGER[0]+sigmaLOGGER[0
# print("n+1: "+ str(round(func(len(y_data)+daysLP, paramsLOGGER[0], paramsLOGGER[1]
# print("sigma-: "+ str(round(func(len(y_data)+daysLP, paramsLOGGER[0]-sigmaLOGGER[0
print("----\nPY:")
print("sigma+: "+ str(round(func(len(y_data)+days, params[0]+sigma[0], params[1]+sign
print("n+1: "+ str(round(func(len(y_data)+days, params[0], params[1], params[2]))))
print("Sigma-: "+ str(round(func(len(y_data)+days, params[0]-sigma[0], params[1]-sign


# print("LP sigma: ",sigmaLOGGER)
# y_pred_plus = func(x_future, paramsLOGGER[0]+sigmaLOGGER[0]*SM, paramsLOGGER[1]+si
# y_pred_minus = func(x_future, paramsLOGGER[0]-sigmaLOGGER[0]*SM, paramsLOGGER[1]-s
# plt.fill_between(x_future, y_pred_minus,y_pred_plus, alpha=0.5,label='We gonna die
print("----\n")
print("params PY: ",params)
print("PY sigma: ",sigma)
print("max cases: ", round(params[0]))
print("infection point: ", round(params[2]))
date_start= datetime.strptime("01/22/20", "%m/%d/%y")
infectionpoint = date_start + timedelta(days=round(params[2]))
print ("infection point date: ",infectionpoint.date())




y_pred_plus = func(x_future, params[0]+sigma[0], params[1]+sigma[1], params[2]-sigma
y_pred_minus = func(x_future, params[0]-sigma[0], params[1]-sigma[1], params[2]+sigma
plt.fill_between(x_future, y_pred_minus,y_pred_plus, alpha=0.3,label='We gonna die fa

plt.plot(x_pred, func(x_pred, params[0], params[1], params[2]),
         label='Fitted function Py')
120
# plt.plot(x_pred, func(x_pred, paramsLOGGER[0], paramsLOGGER[1], paramsLOGGER[2]),
#          label='Fitted function LP')

plt.legend(loc='lower right')

plt.xlim(30, 120)
# plt.semilogy(basey=1)
# plt.yscale('log')
plt.title("Total number of cases for "+dataName+"!\n max cases: ~" + str(round(params
plt.show()
```

```
----
PY:
```

```
sigma+: 8747.0
n+1: 8515.0
Sigma-: 8283.0
----

params PY:  [1.02531999e+04 1.87092078e-01 7.55084100e+01]
PY sigma:  [1.65517145e+02 3.10405089e-03 2.09387957e-01]
max cases:  10253.0
infection point:  76.0
infection point date:  2020-04-07
```



```
In [417]: # with plt.xkcd():
          #     plt.figure(figsize=(12, 8))
          #     diff_LP = y_data - func(np.arange(0,len(y_data)).astype(float), paramsLOGGER[0
          #     # print(diff_LP)
          #     diff_PY = y_data - func(np.arange(0,len(y_data)).astype(float), params[0], para
          #     # print(diff_PY)
          #     plt.plot(x_data, diff_LP, label='diff LP')
          #     plt.plot(x_data, diff_PY, label='diff PY')
          #     plt.plot(x_data, diff_LP-diff_PY,label='diff of the two', alpha=0.5 )
          #     plt.legend(loc='upper left')
```

```
In [418]: # plt.figure(figsize=(12, 8))
          # plt.scatter(x_data, y_data, label='Data')

          # x_pred=np.arange(0,len(ont)+30).astype(float)
          # x_future=np.arange(0,len(ont)+30).astype(float)

          # print("sigma+: "+ str(func(len(ont)+1, params[0]+sigma[0], params[1]+sigma[1], para
          # print("n+1: "+ str(func(len(ont)+1, params[0], params[1], params[2])))
          # print("Sigma-: "+ str(func(len(ont)+1, params[0]-sigma[0], params[1]-sigma[1], para

          # sigma = np.sqrt(np.diag(params_covariance))
          # # print(sigma)
          # y_pred_plus = func(x_future, params[0]+sigma[0], params[1]+sigma[1], params[2]-sig
          # y_pred_minus = func(x_future, params[0]-sigma[0], params[1]-sigma[1], params[2]+si
          # plt.fill_between(x_future, y_pred_minus,y_pred_plus, alpha=0.5,label='We gonna die


          # plt.plot(x_pred, func(x_pred, params[0], params[1], params[2]),
          #          label='Fitted function')

          # plt.legend(loc='best')

          # plt.show()

In [419]: # def Gauss(x, a, x0, sigma):
          #     return a * np.exp(-(x - x0)**2 / (2 * sigma**2))


          # delta_cases=np.diff(y_data)[0:-1]
          # delta_dates=np.arange(0,len(delta_cases)).astype(float)

          # delta_x_future=np.arange(0,len(delta_cases)+55).astype(float)

          # mean = sum(delta_dates * delta_cases) / sum(delta_cases)
          # sigma = np.sqrt(sum(delta_cases * (delta_dates - mean)**2) / sum(delta_cases))

          # popt,pcov = curve_fit(Gauss, delta_dates, delta_cases, p0=[max(delta_cases), mean,


          # x_pred_delta=np.arange(0,len(delta_cases)+255).astype(float)


          # plt.figure(figsize=(8, 6))
          # plt.title("new number of cases for "+"ontario"+"!\n")
          # plt.scatter(delta_dates, delta_cases)

          # plt.plot(x_pred_delta, Gauss(x_pred_delta, popt[0], popt[1], popt[2]),
          #              label='Fitted function gauss')
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

1 AUTOMATIC

```
In [420]: date_start= datetime.strptime("01/22/20", "%m/%d/%y")
          end_date = date_start + timedelta(days=72)
          print (end_date.date())
```

```
2020-04-03
```

```
In [716]: def MakeMePrediction(where,maxfev):
              print("\n\n------------- "+where+ "-------------")
          #     rawData="https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_
          #     data = pd.read_csv(rawData)
              canada_Data="https://health-infobase.canada.ca/src/data/covidLive/covid19.csv"
              data = pd.read_csv(canada_Data)

              dataName=where
          #     ont=givemedata(data,where)
              ont=CanadaData(data,where)

              totaltested=CanadaDatatests(data,dataName)


              deaths=CanadaDataDeaths(data,dataName)
              death_dates=np.arange(0,len(deaths)).astype(float)

              delta_testes=np.diff(totaltested)[:] #total tested
              d_tested_dates=np.arange(0,len(delta_testes))

              y_data = ont[0:].astype(float)
              x_data = np.arange(len(y_data)).astype(float)


              #Fit model
              params, params_covariance = optimize.curve_fit(func, x_data, y_data,
                                                             p0=[4.451E5,0.1893,121.9],maxfev=
              params1, params_covariance1 = optimize.curve_fit(func, x_data[0:-1], y_data[0:-1
                                                             p0=[4.451E5,0.1893,121.9],maxfev=
              params3, params_covariance3 = optimize.curve_fit(func, x_data[0:-3], y_data[0:-3
                                                             p0=[4.451E5,0.1893,121.9],maxfev=
              params5, params_covariance5 = optimize.curve_fit(func, x_data[0:-5], y_data[0:-5]
                                                             p0=[4.451E5,0.1893,121.9],maxfev=9
              params10, params_covariance10 = optimize.curve_fit(func, x_data[0:-10], y_data[0
                                                             p0=[4.451E5,0.1893,121.9],maxfev=9
              param_ex, ex_cov = optimize.curve_fit(exponenial_func,x_data, y_data,
                                                   p0=[100,0.01,-2000],maxfev=9000)


              #Out put raw data
              print ("dates: ",x_data)
              print ("cases: ",y_data)
              #Calculate sigma
              sigma = np.sqrt(np.diag(params_covariance))
              sigma1 = np.sqrt(np.diag(params_covariance1))
              print("-----")
              print("params PY: ",params)
              print("sigma PY: ", sigma)

          #     #QA of data
          #     plt.figure(figsize=(8, 6))
          #     plt.title("Total number of cases for "+dataName+"!\n")
          #     plt.scatter(x_data, y_data)
          #     plt.plot(np.arange(len(y_data)),y_data)
          #     plt.grid(alpha=0.2)



              #-------------------GAUSSIAN

              delta_cases=np.diff(y_data)[:]
              delta_dates=np.arange(0,len(delta_cases)).astype(float)
              delta_x_future=np.arange(0,len(delta_cases)+75).astype(float)
              mean = sum(delta_dates * delta_cases) / sum(delta_cases)
              sigma_gfit = np.sqrt(sum(delta_cases * (delta_dates - mean)**2) / sum(delta_cases
              popt,pcov   = curve_fit(Gauss, delta_dates, delta_cases, p0=[max(delta_cases), me
              popt1,pcov1 = curve_fit(Gauss, delta_dates[0:-1], delta_cases[0:-1], p0=[max(del
              popt2,pcov2 = curve_fit(Gauss, delta_dates[0:-2], delta_cases[0:-2], p0=[max(del
              popt3,pcov3 = curve_fit(Gauss, delta_dates[0:-3], delta_cases[0:-3], p0=[max(del
              popt4,pcov4 = curve_fit(Gauss, delta_dates[0:-4], delta_cases[0:-4], p0=[max(del
              popt5,pcov5 = curve_fit(Gauss, delta_dates[0:-5], delta_cases[0:-5], p0=[max(del
              popt6,pcov6 = curve_fit(Gauss, delta_dates[0:-6], delta_cases[0:-6], p0=[max(del
              popt7,pcov7 = curve_fit(Gauss, delta_dates[0:-7], delta_cases[0:-7], p0=[max(del
              popt8,pcov8 = curve_fit(Gauss, delta_dates[0:-8], delta_cases[0:-8], p0=[max(del
              popt9,pcov9 = curve_fit(Gauss, delta_dates[0:-9], delta_cases[0:-9], p0=[max(del

              x_pred_delta=np.arange(0,len(delta_cases)+25).astype(float)
              sigmaG = np.sqrt(np.diag(pcov))
```

```
y_pred_plusG = Gauss(x_pred_delta, popt[0]+sigmaG[0], popt[1]+sigmaG[1], popt[2]+
y_pred_minusG = Gauss(x_pred_delta, popt[0]-sigmaG[0], popt[1]-sigmaG[1], popt[2


plt.figure(figsize=(12, 8))
plt.title("new number of cases by day for "+where+"!\n")
plt.xlabel("Days from Day0(see note at top)")
plt.ylabel("Cases")
plt.scatter(delta_dates, delta_cases, alpha=0.9)
plt.plot(x_pred_delta, Gauss(x_pred_delta, popt[0], popt[1], popt[2]),
             label='Fitted function gauss', color='green')
plt.plot(x_pred_delta, Gauss(x_pred_delta, popt1[0], popt1[1], popt1[2]), alpha=
             label='n-1')
plt.plot(x_pred_delta, Gauss(x_pred_delta, popt2[0], popt2[1], popt2[2]), alpha=
             label='n-2')
plt.plot(x_pred_delta, Gauss(x_pred_delta, popt3[0], popt3[1], popt3[2]), alpha=
             label='n-3')
plt.plot(x_pred_delta, Gauss(x_pred_delta, popt4[0], popt4[1], popt4[2]), alpha=
             label='n-4')
plt.plot(x_pred_delta, Gauss(x_pred_delta, popt5[0], popt5[1], popt5[2]), alpha=
             label='n-5')
plt.plot(x_pred_delta, Gauss(x_pred_delta, popt6[0], popt6[1], popt6[2]), alpha=
             label='n-6')
plt.plot(x_pred_delta, Gauss(x_pred_delta, popt7[0], popt7[1], popt7[2]), alpha=
             label='n-7')
plt.plot(x_pred_delta, Gauss(x_pred_delta, popt8[0], popt8[1], popt8[2]), alpha=
             label='n-8')
plt.plot(x_pred_delta, Gauss(x_pred_delta, popt9[0], popt9[1], popt9[2]), alpha=
             label='n-9')
plt.fill_between(x_pred_delta, y_pred_minusG,y_pred_plusG, alpha=0.3,label='1 Si
plt.legend(loc='best')
plt.grid(alpha=0.2)
plt.show()



#--------------------MODEL!!!
plt.figure(figsize=(12, 8))
plt.xlabel("Days from Day0(see note at top)")
plt.ylabel("Cases")
plt.scatter(x_data, y_data, label='Data')
plt.scatter(death_dates,deaths,label='Deaths',color='red')


x_pred=np.arange(0,len(y_data)+15).astype(float)
x_future=np.arange(0,len(y_data)+15).astype(float)
days=0
daysLP=days
```

```
    SM=3

    print("----\nPY:")
    print("sigma+: "+ str(round(func(len(y_data)+days, params[0]+sigma[0], params[1]
    print("n+1: "+ str(round(func(len(y_data)+days, params[0], params[1], params[2]))
    print("Sigma-: "+ str(round(func(len(y_data)+days, params[0]-sigma[0], params[1]


    print("----\n")
    print("params PY: ",params)
    print("PY sigma: ",sigma)
    print("max cases: ", round(params[0]))
    print("infection point: ", round(params[2]))
    date_start= datetime.strptime("01/31/20", "%m/%d/%y")
    infectionpoint = date_start + timedelta(days=round(params[2]))
    print ("infection point date: ",infectionpoint.date())


    y_pred_plus = func(x_future, params[0]+sigma[0], params[1]+sigma[1], params[2]-s
    y_pred_minus = func(x_future, params[0]-sigma[0], params[1]-sigma[1], params[2]+
    plt.fill_between(x_future, y_pred_minus,y_pred_plus, alpha=0.3,label='1 Sigma on

    y_pred_plus1 = func(x_future, params1[0]+sigma1[0], params1[1]+sigma1[1], params
    y_pred_minus1 = func(x_future, params1[0]-sigma1[0], params1[1]-sigma1[1], param
    plt.fill_between(x_future, y_pred_minus1,y_pred_plus1, alpha=0.1,label='1 Sigma 

    plt.plot(x_pred, func(x_pred, params[0], params[1], params[2]),
             label='Fitted function Py')
    plt.plot(x_pred, func(x_pred, params1[0], params1[1], params1[2]), alpha=0.7,
             label='1 Day ago')
    plt.plot(x_pred, func(x_pred, params3[0], params3[1], params3[2]), alpha=0.7,
             label='3 Days ago')
    plt.plot(x_pred, func(x_pred, params5[0], params5[1], params5[2]), alpha=0.7,
             label='5 Days ago')
    plt.plot(x_pred, func(x_pred, params10[0], params10[1], params10[2]), alpha=0.7,
             label='10 Days ago')
#     plt.plot(x_pred, exponenial_func(x_pred, param_ex[0], param_ex[1], param_ex[2]
#              label='exponenial fit')


    plt.legend(loc='best')
    plt.tick_params(labelright=True)
    plt.grid(alpha=0.2)
#     plt.xlim(-10, 80)
    # plt.semilogy(basey=1)
#     plt.yscale('log')
    plt.title("Total number of cases for "+dataName+"!\n max cases: ~" + str(round(p
    plt.show()


    #-------------------tests for cases-------------------

    x_pred_delta_tests=np.arange(0,len(delta_cases)+5).astype(float)

    tested_dates=np.arange(0,totaltested.size)
    valid = ~(np.isnan(tested_dates) | np.isnan(totaltested))
    popt_line_t, pcov_line_t = curve_fit(Linefunc, tested_dates[valid][0:-1], totalte

    plt.figure(figsize=(12,8))
    plt.xlabel("Days from Day0(see note at top)")
    plt.ylabel("Tests")
    plt.title("Total test to day in "+dataName+"!\n"+"Rate: "+str(round(popt_line_t[0
    plt.scatter(np.arange(0,totaltested.size),totaltested, label='data')
    plt.plot(x_pred_delta_tests, Linefunc(x_pred_delta_tests, popt_line_t[0], popt_li
    plt.legend(loc='best')
    plt.grid(alpha=0.2,which='both')
    plt.show()
    #--
    valid = ~(np.isnan(d_tested_dates) | np.isnan(delta_testes))
    popt_line, pcov_line = curve_fit(Linefunc, d_tested_dates[valid][0:-1], delta_tes

    plt.figure(figsize=(12, 8))
    plt.xlabel("Days from Day0(see note at top)")
    plt.ylabel("Tests")
    plt.title("Tests by day for "+dataName+"!\n"+"Rate: "+str(round(popt_line[0]))+"
    plt.scatter( d_tested_dates,delta_testes, label='tests by day')
    plt.plot(x_pred_delta_tests, Linefunc(x_pred_delta_tests, popt_line[0], popt_line
    plt.legend(loc='best')
    plt.grid(alpha=0.2,which='both')
    plt.show()

    print("Precent of positive cases pre test:", str(round ((y_data[-1]/totaltested[-
```

1.1 The Cruve for everywhere

```
In [717]: def TheCruve(AllofCanada,AllofCanadaAbv,maxfev):

              where="Ontario"

              canada_Data="https://health-infobase.canada.ca/src/data/covidLive/covid19.csv"
              data = pd.read_csv(canada_Data)

              plt.figure(figsize=(12, 8))
```

```
    plt.title("Daily cases by province\n AKA \"The Curve\"")
    plt.xlabel("Day from Day 0 of each data set (see note to the top)")
    plt.ylabel("Cases")

    for i,where in enumerate(AllofCanada):
        ont=CanadaData(data,where)
        y_data = ont[0:].astype(float)
        x_data = np.arange(len(y_data)).astype(float)

        delta_cases=np.diff(y_data)[:]
        delta_dates=np.arange(0,len(delta_cases)).astype(float)
        delta_x_future=np.arange(0,len(delta_cases)+115).astype(float)
        mean = sum(delta_dates * delta_cases) / sum(delta_cases)
        sigma_gfit = np.sqrt(sum(delta_cases * (delta_dates - mean)**2) / sum(delta_
        popt,pcov   = curve_fit(Gauss, delta_dates, delta_cases, p0=[max(delta_cases

        plt.scatter(delta_dates, delta_cases, alpha=0.3,
                    label='Data for '+AllofCanadaAbv[i])
        plt.plot(x_pred_delta, Gauss(x_pred_delta, popt[0], popt[1], popt[2]),
                 label='Gaus. '+AllofCanadaAbv[i])
    plt.legend(loc='best')
#     plt.yscale('log')
    plt.show()


    plt.figure(figsize=(12, 8))
    plt.title("Normalized Daily cases by province\n AKA \"The Curve\"")
    plt.xlabel("Day from Day 0 of each data set (see note to the top)")
    plt.ylabel("Normalized Cases")

    for i,where in enumerate(AllofCanada):
        ont=CanadaData(data,where)
        y_data = ont[0:].astype(float)
        y_data = y_data / np.sqrt(np.sum(y_data**2))
        x_data = np.arange(len(y_data)).astype(float)

        delta_cases=np.diff(y_data)[:]
        delta_dates=np.arange(0,len(delta_cases)).astype(float)
        delta_x_future=np.arange(0,len(delta_cases)+115).astype(float)
        mean = sum(delta_dates * delta_cases) / sum(delta_cases)
        sigma_gfit = np.sqrt(sum(delta_cases * (delta_dates - mean)**2) / sum(delta_
        popt,pcov   = curve_fit(Gauss, delta_dates, delta_cases, p0=[max(delta_cases

    #     plt.scatter(delta_dates, delta_cases, alpha=0.1,
    #                 label='Data for '+AllofCanadaAb[i])
        if where=="Canada":
            plt.plot(x_pred_delta, Gauss(x_pred_delta, popt[0], popt[1], popt[2]),
                     label='Gaus. '+AllofCanadaAbv[i],linewidth=4)
        else:
            plt.plot(x_pred_delta, Gauss(x_pred_delta, popt[0], popt[1], popt[2]),
                     label='Gaus. '+AllofCanadaAbv[i],alpha=0.5)
    plt.legend(loc='best')
    plt.show()
    print("The plot above shows total cases normalized to 1 for each province \n\n\n
```

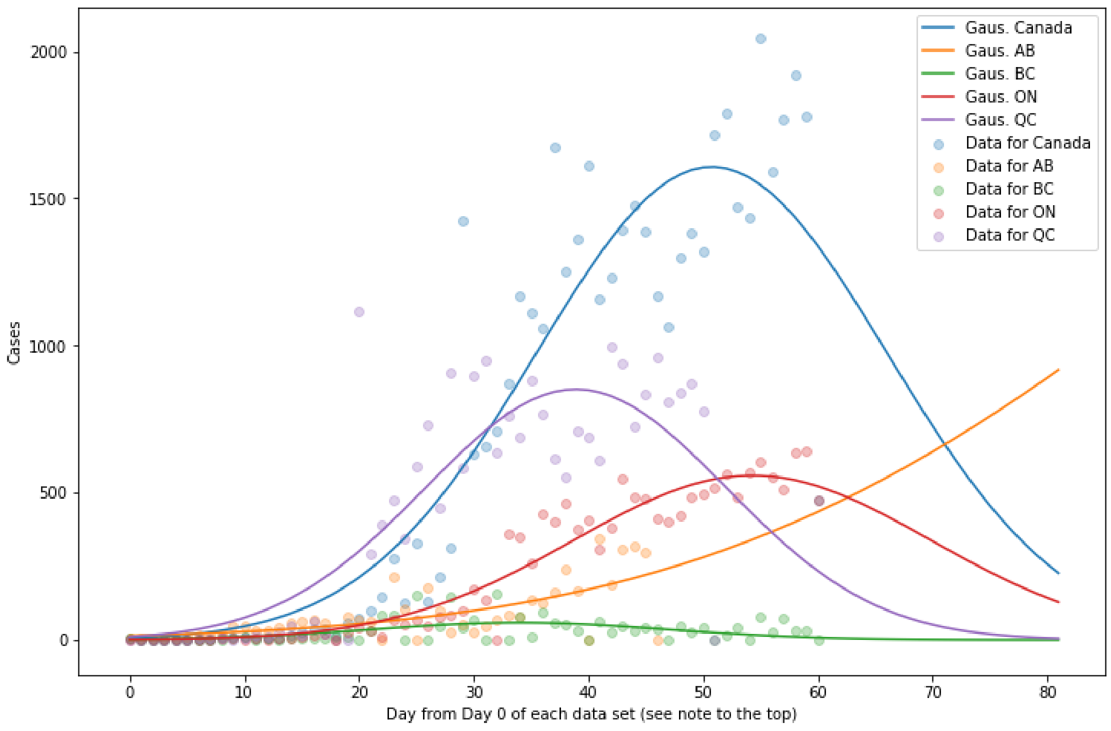
```
In [722]: %matplotlib inline
          # %time
          print ("Last updated: "+datetime.now().strftime("%Y-%m-%d %H:%M:%S"))


          note="""
              Waqar Muhammad - mwaqar@snolab.ca

              NOTE:
              All of the following was data is from Health Canada.
              The CDF fit is a logicitsts curve fit.
              The PDF fit is a simple gaussian fit.
              Day 0 is:
              -> 31-01-2020 for Canada
              -> 08-03-2020 for AB
              -> 31-01-2020 for BC
              -> 11-03-2020 for MN
              -> 11-03-2020 for NB
              -> 11-03-2020 for NL
              -> 11-03-2020 for NS
              -> 31-01-2020 for ON
              -> 11-03-2020 for PEI
              -> 01-03-2020 for QC
              -> 11-03-2020 for SW"""
          print(note)


          AllofCanada=[ "Ontario"]
          AllofCanadaAbv=["ON"]

          AllofCanada=["Canada","Alberta","British Columbia", "Ontario", "Quebec"]
          AllofCanadaAbv=["Canada","AB","BC","ON","QC"]

          # AllofCanada=["Canada","Alberta","British Columbia", "Manitoba", "New Brunswick",
          #                 "Newfoundland and Labrador", "Nova Scotia", "Ontario", "Prince Edwar
          # AllofCanadaAbv=["Canada","AB","BC", "MN", "NB",
          #                 "NL", "NS", "ON", "PEI", "QC", "SW"]

          TheCruve(AllofCanada,AllofCanadaAbv,519000)
```

```
for i in AllofCanada:
    MakeMePrediction(i,519000)
```

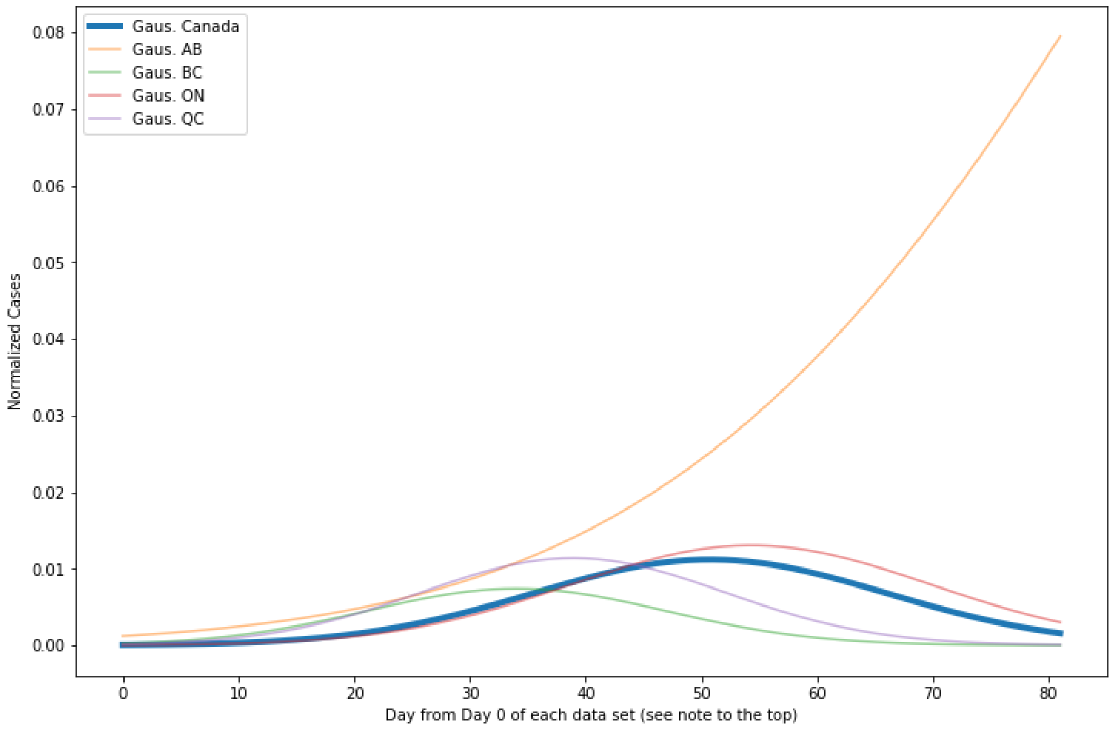
```
    # All of the following was data is from Health Canada.
    # The CDF fit is a logicitsts curve fit.
    # The PDF fit is a simple gaussian fit.
    # Day 0 is:
    # -> 31-01-2020 for Canada
    # -> 08-03-2020 for AB
    # -> 31-01-2020 for BC
    # -> 11-03-2020 for MN
    # -> 11-03-2020 for NB
    # -> 11-03-2020 for NL
    # -> 11-03-2020 for NS
    # -> 31-01-2020 for ON
    # -> 11-03-2020 for PEI
    # -> 01-03-2020 for QC
    # -> 11-03-2020 for SW
```

```
Last updated: 2020-04-25 18:59:11

    Waqar Muhammad - mwaqar@snolab.ca

    NOTE:
    All of the following was data is from Health Canada.
    The CDF fit is a logicitsts curve fit.
    The PDF fit is a simple gaussian fit.
    Day 0 is:
    -> 31-01-2020 for Canada
    -> 08-03-2020 for AB
    -> 31-01-2020 for BC
    -> 11-03-2020 for MN
    -> 11-03-2020 for NB
    -> 11-03-2020 for NL
    -> 11-03-2020 for NS
    -> 31-01-2020 for ON
    -> 11-03-2020 for PEI
    -> 01-03-2020 for QC
    -> 11-03-2020 for SW
```

Daily cases by province
AKA "The Curve"

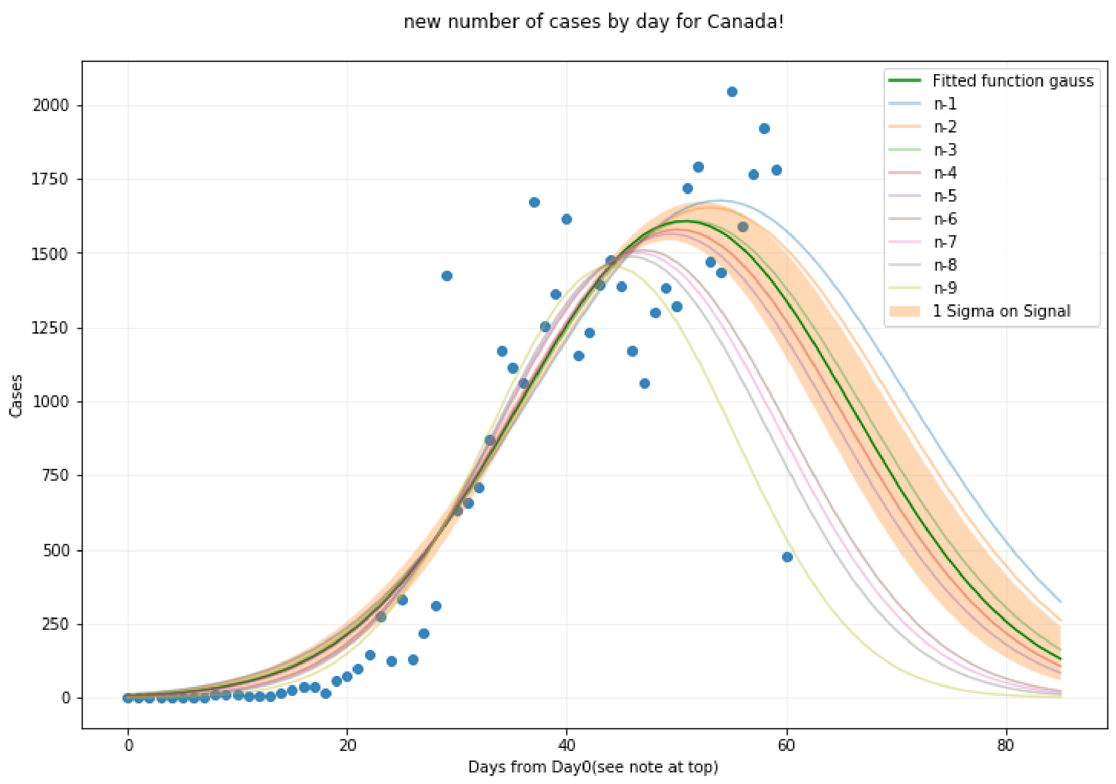


Normalized Daily cases by province
AKA "The Curve"

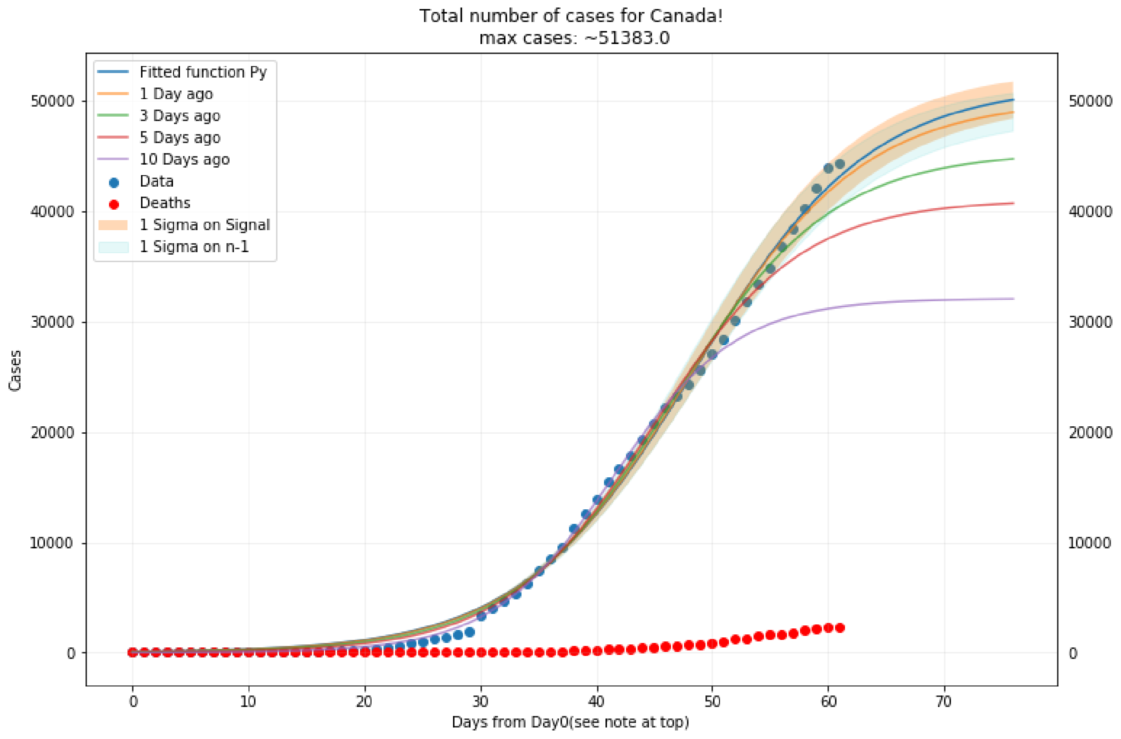


The plot above shows total cases normalized to 1 for each province

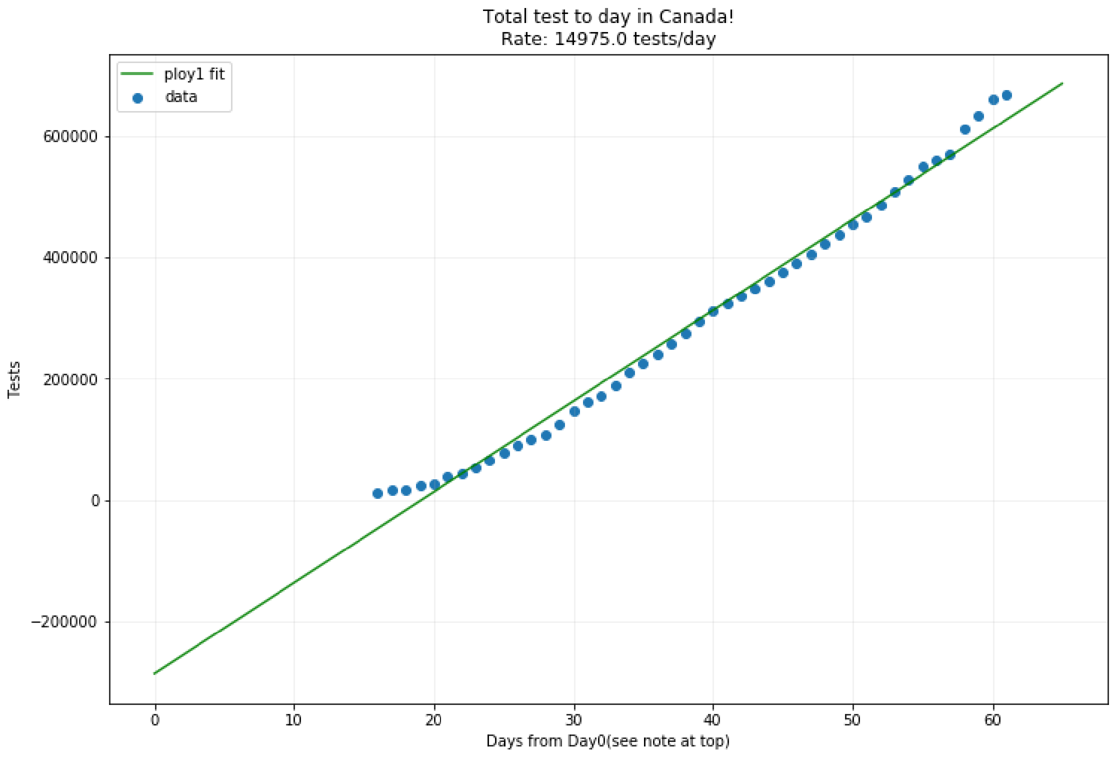
```
------------- Canada-------------
dates:  [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16. 17.
 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35.
 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53.
 54. 55. 56. 57. 58. 59. 60. 61.]
cases:  [4.0000e+00 7.0000e+00 8.0000e+00 9.0000e+00 1.0000e+01 1.1000e+01
 1.2000e+01 1.3000e+01 1.5000e+01 2.4000e+01 3.3000e+01 4.5000e+01
 5.1000e+01 5.7000e+01 6.2000e+01 7.7000e+01 1.0300e+02 1.3800e+02
 1.7600e+02 1.9300e+02 2.4900e+02 3.2400e+02 4.2400e+02 5.6900e+02
 8.4600e+02 9.7100e+02 1.3020e+03 1.4300e+03 1.6460e+03 1.9590e+03
 3.3850e+03 4.0180e+03 4.6750e+03 5.3860e+03 6.2550e+03 7.4240e+03
 8.5360e+03 9.5950e+03 1.1268e+04 1.2519e+04 1.3882e+04 1.5496e+04
 1.6653e+04 1.7883e+04 1.9274e+04 2.0748e+04 2.2133e+04 2.3301e+04
 2.4365e+04 2.5663e+04 2.7046e+04 2.8364e+04 3.0081e+04 3.1872e+04
 3.3341e+04 3.4777e+04 3.6823e+04 3.8413e+04 4.0179e+04 4.2099e+04
 4.3877e+04 4.4353e+04]
-----
params PY:  [5.13832255e+04 1.32551647e-01 4.85528592e+01]
sigma PY:  [1.47481955e+03 4.42771184e-03 5.46036439e-01]
```



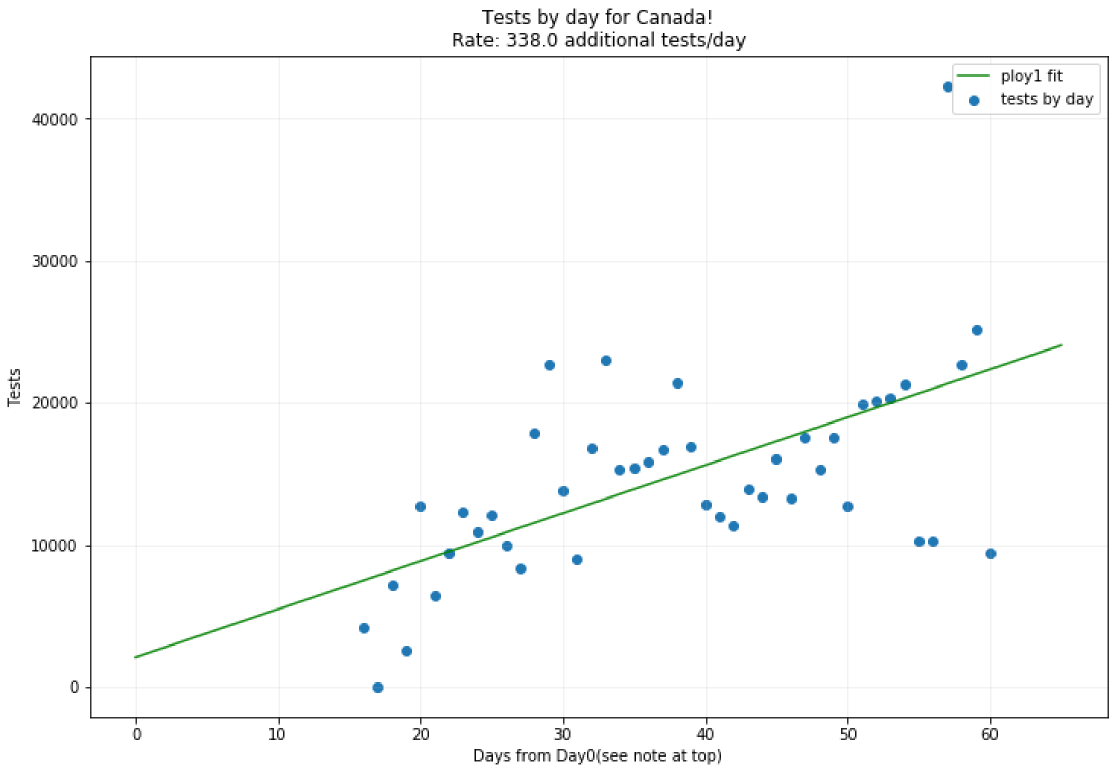
```
----
PY:
sigma+: 46081.0
n+1: 43984.0
Sigma-: 41888.0
----

params PY:  [5.13832255e+04 1.32551647e-01 4.85528592e+01]
PY sigma:  [1.47481955e+03 4.42771184e-03 5.46036439e-01]
max cases:  51383.0
infection point:  49.0
infection point date:  2020-03-20
```



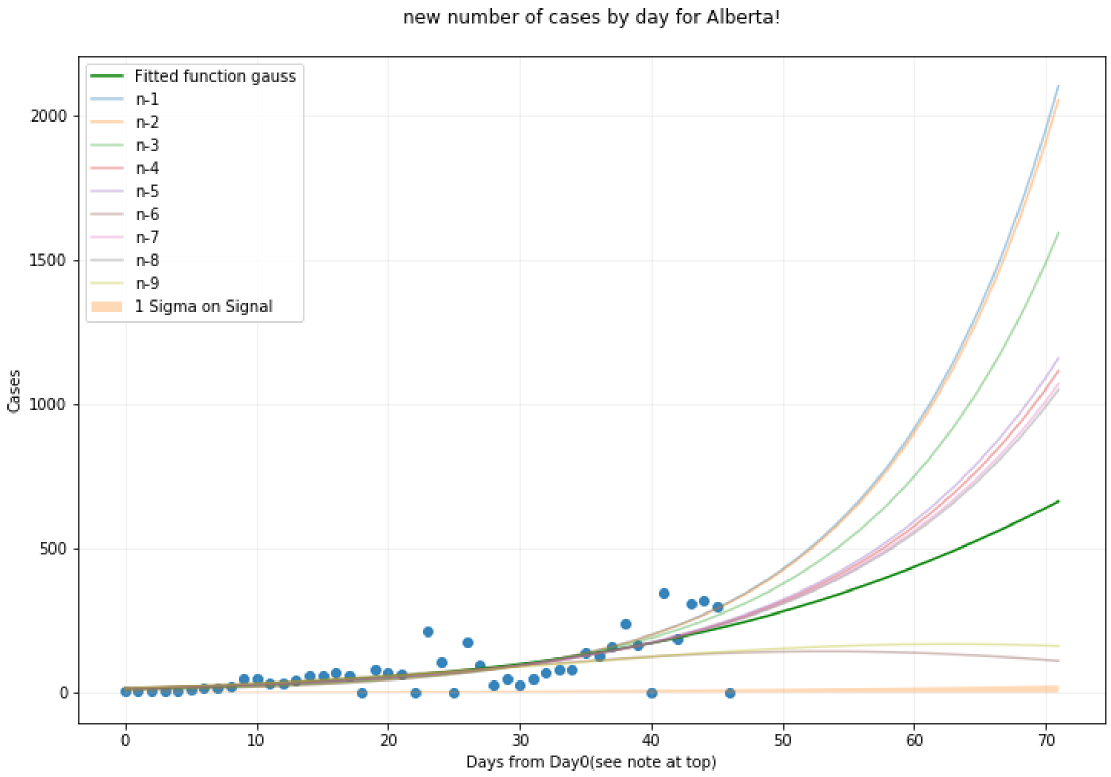
Total number of cases for Canada!
max cases: ~51383.0
Fitted function Py
1 Day ago
3 Days ago
5 Days ago
10 Days ago
Data
Deaths
1 Sigma on Signal
1 Sigma on n-1
Cases
Days from Day0(see note at top)



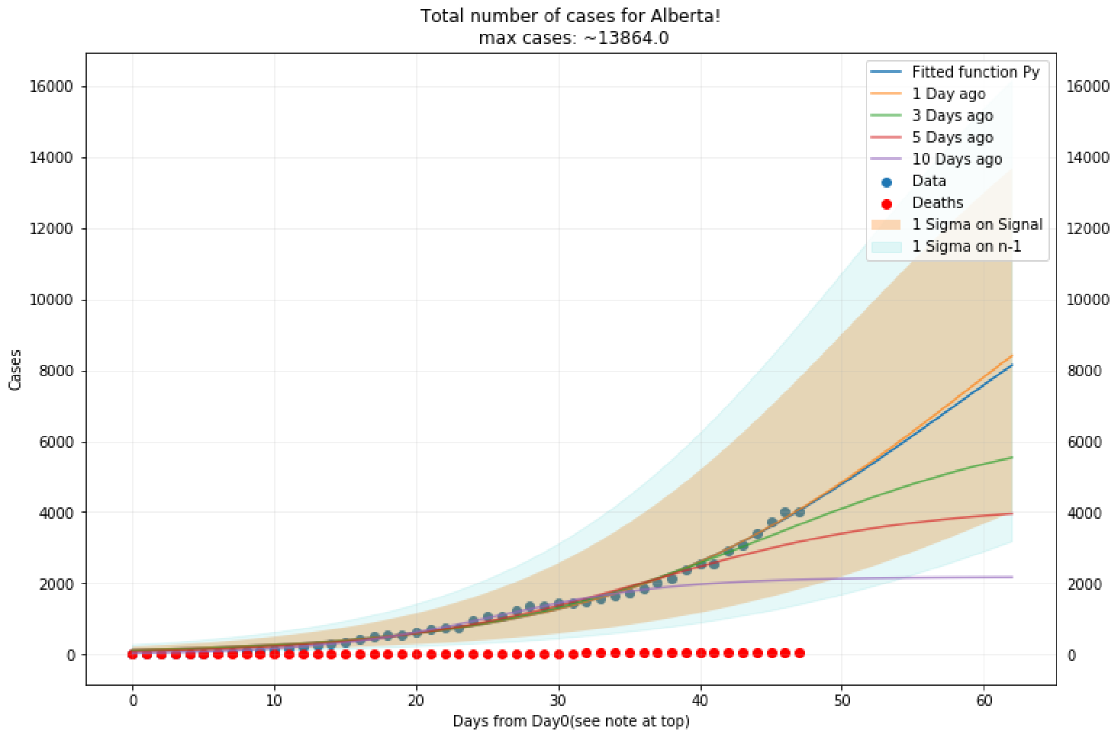
Total test to day in Canada!
Rate: 14975.0 tests/day
ploy1 fit
data
Tests
Days from Day0(see note at top)

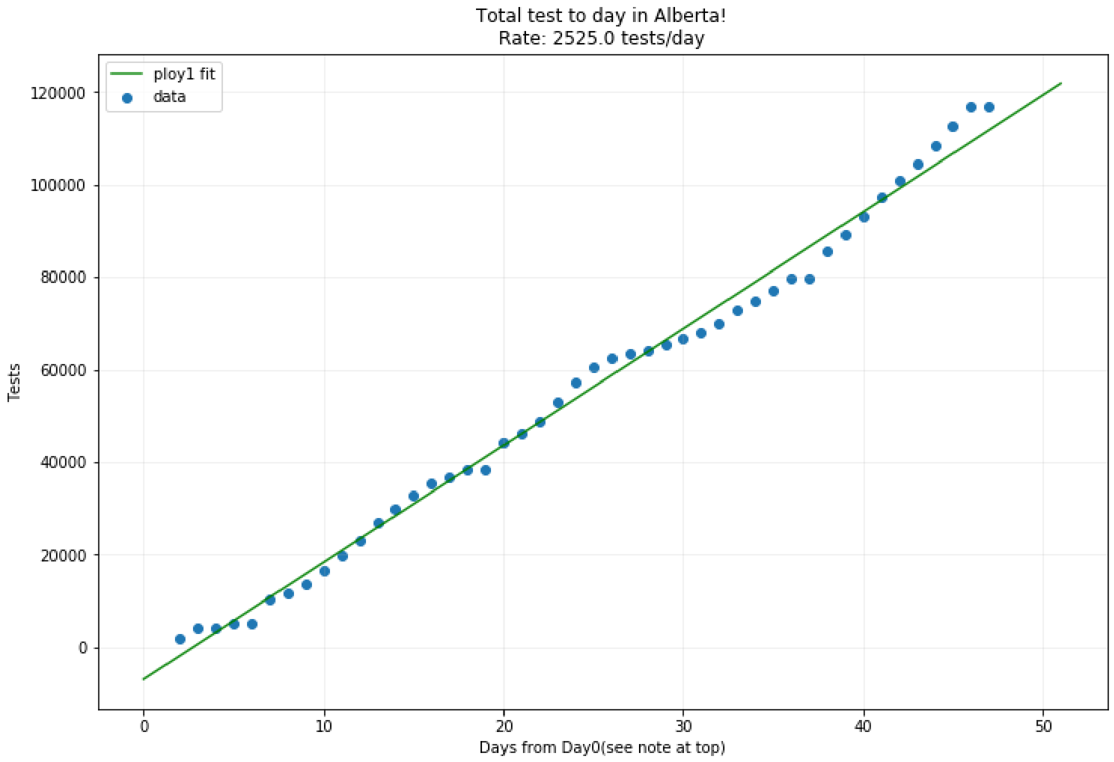


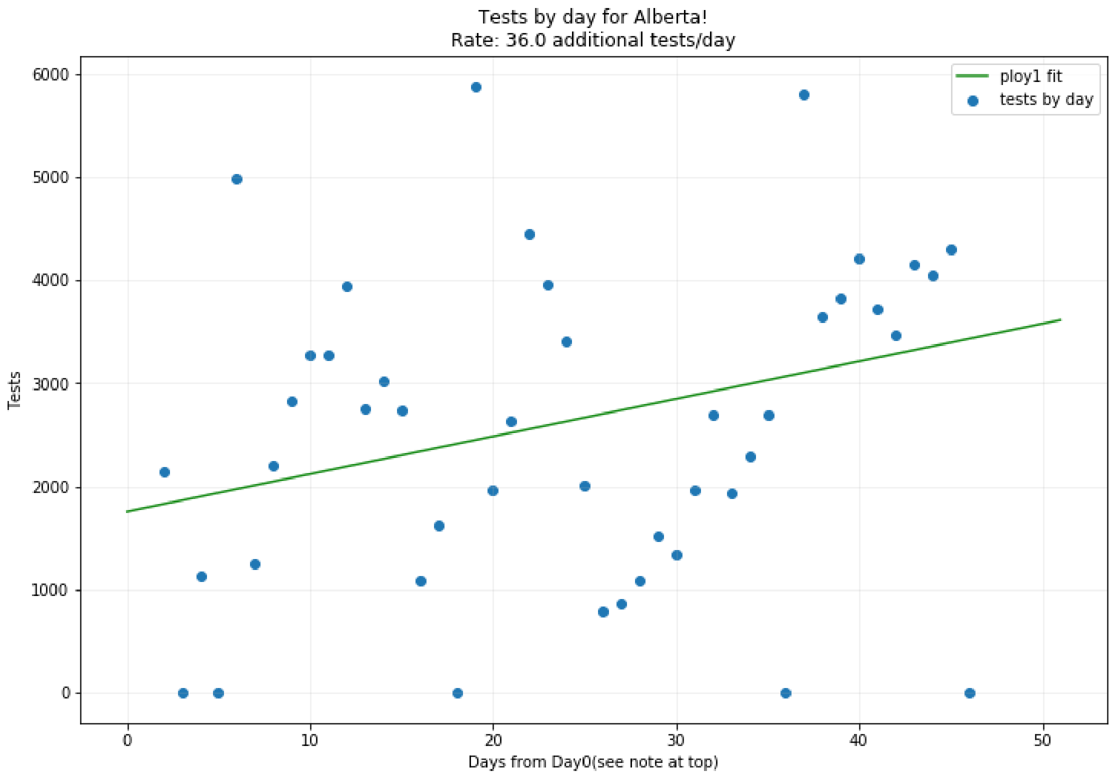
```
Precent of positive cases pre test: 6.624 %


------------- Alberta-------------
dates:  [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16. 17.
 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35.
 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47.]
cases:  [1.000e+00 7.000e+00 1.400e+01 1.900e+01 2.300e+01 2.900e+01 3.900e+01
 5.600e+01 7.400e+01 9.700e+01 1.460e+02 1.950e+02 2.260e+02 2.590e+02
 3.010e+02 3.580e+02 4.190e+02 4.860e+02 5.420e+02 5.420e+02 6.210e+02
 6.900e+02 7.540e+02 7.540e+02 9.680e+02 1.075e+03 1.075e+03 1.250e+03
 1.348e+03 1.373e+03 1.423e+03 1.451e+03 1.500e+03 1.569e+03 1.651e+03
 1.732e+03 1.870e+03 1.996e+03 2.158e+03 2.397e+03 2.562e+03 2.562e+03
 2.908e+03 3.095e+03 3.401e+03 3.720e+03 4.017e+03 4.017e+03]
-----
params PY:  [1.38643457e+04 8.25492791e-02 5.77224306e+01]
sigma PY:  [5.04271686e+03 5.45387623e-03 6.66878728e+00]
```



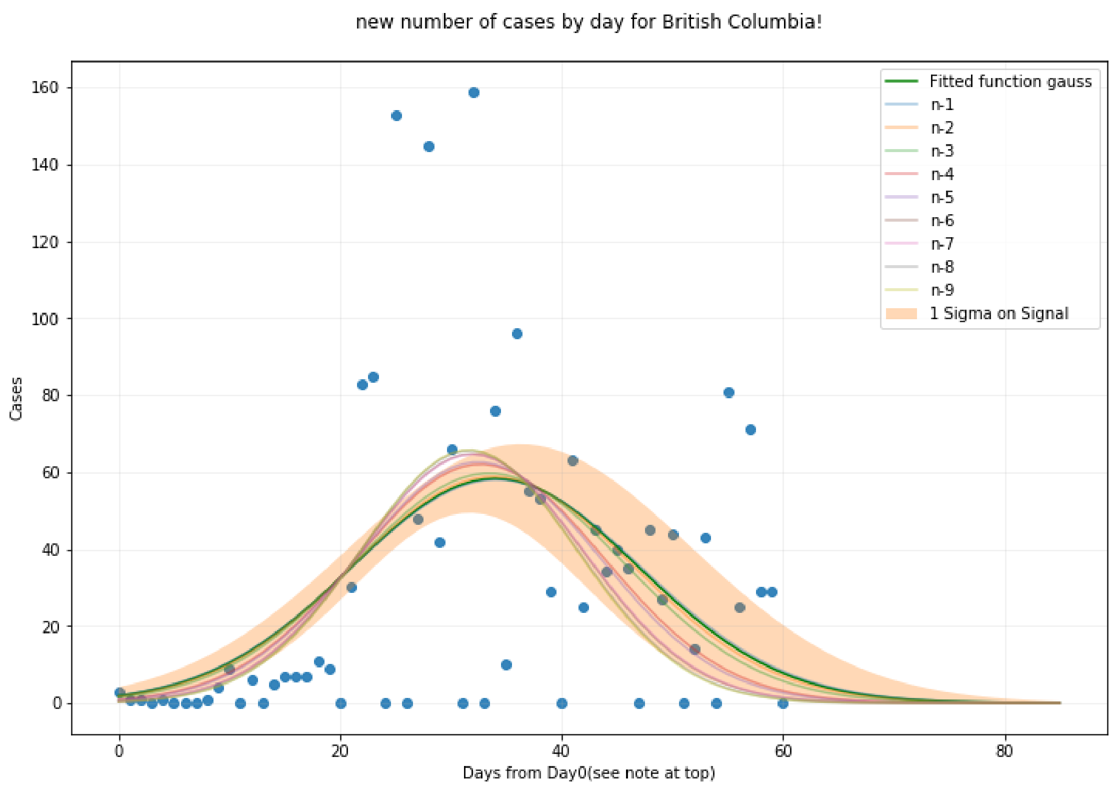
```
----
PY:
sigma+: 8191.0
n+1: 4291.0
Sigma-: 1944.0
----

params PY:  [1.38643457e+04 8.25492791e-02 5.77224306e+01]
PY sigma:  [5.04271686e+03 5.45387623e-03 6.66878728e+00]
max cases:  13864.0
infection point:  58.0
infection point date:  2020-03-29
```


Total number of cases for Alberta!
max cases: ~13864.0
Fitted function Py
1 Day ago
3 Days ago
5 Days ago
10 Days ago
Data
Deaths
1 Sigma on Signal
1 Sigma on n-1
Cases
Days from Day0(see note at top)

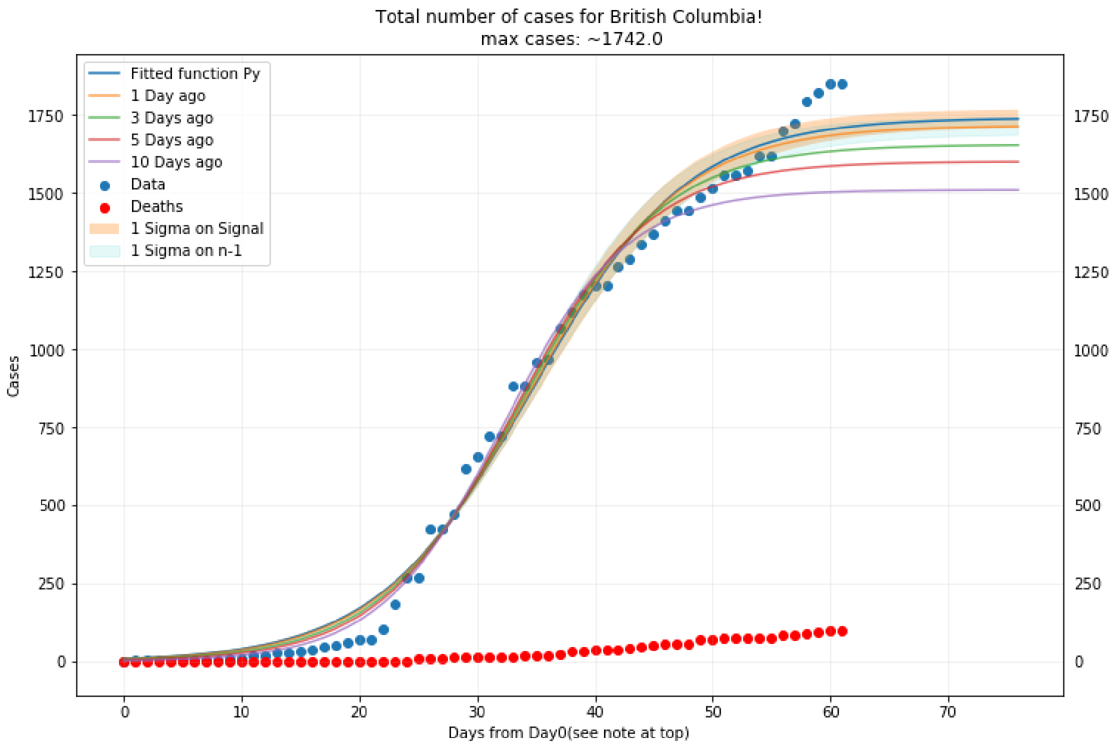

Total test to day in Alberta!
Rate: 2525.0 tests/day
ploy1 fit
data
Tests
Days from Day0(see note at top)

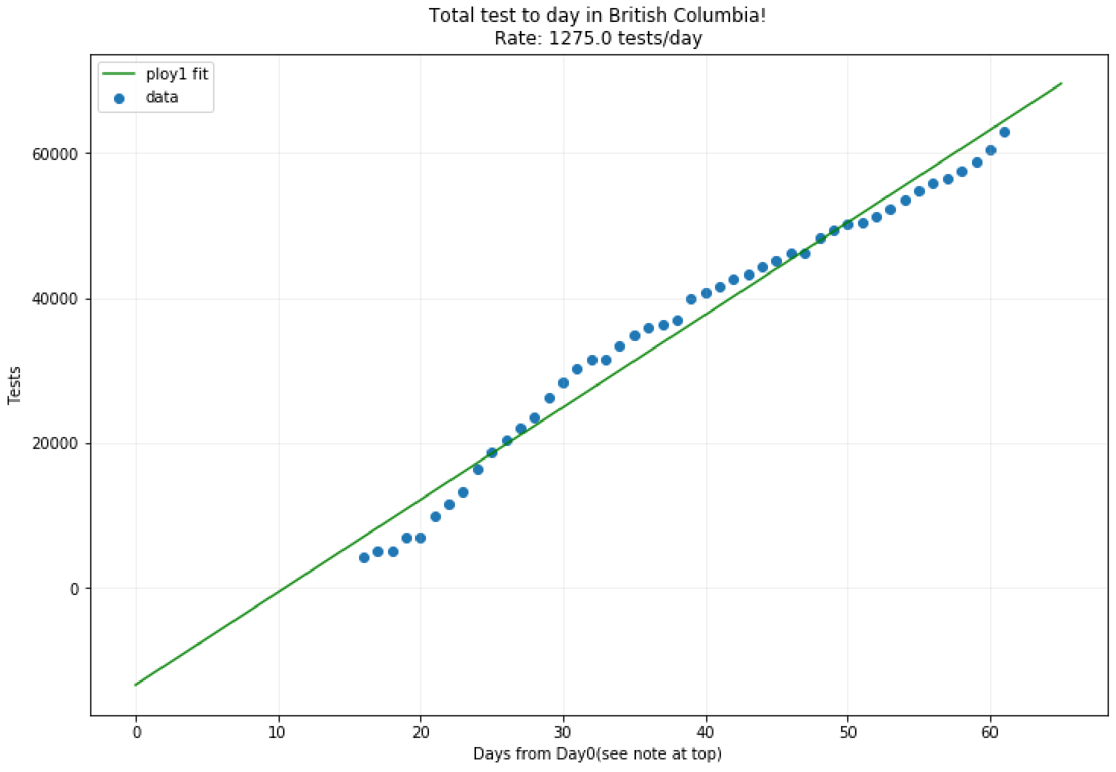


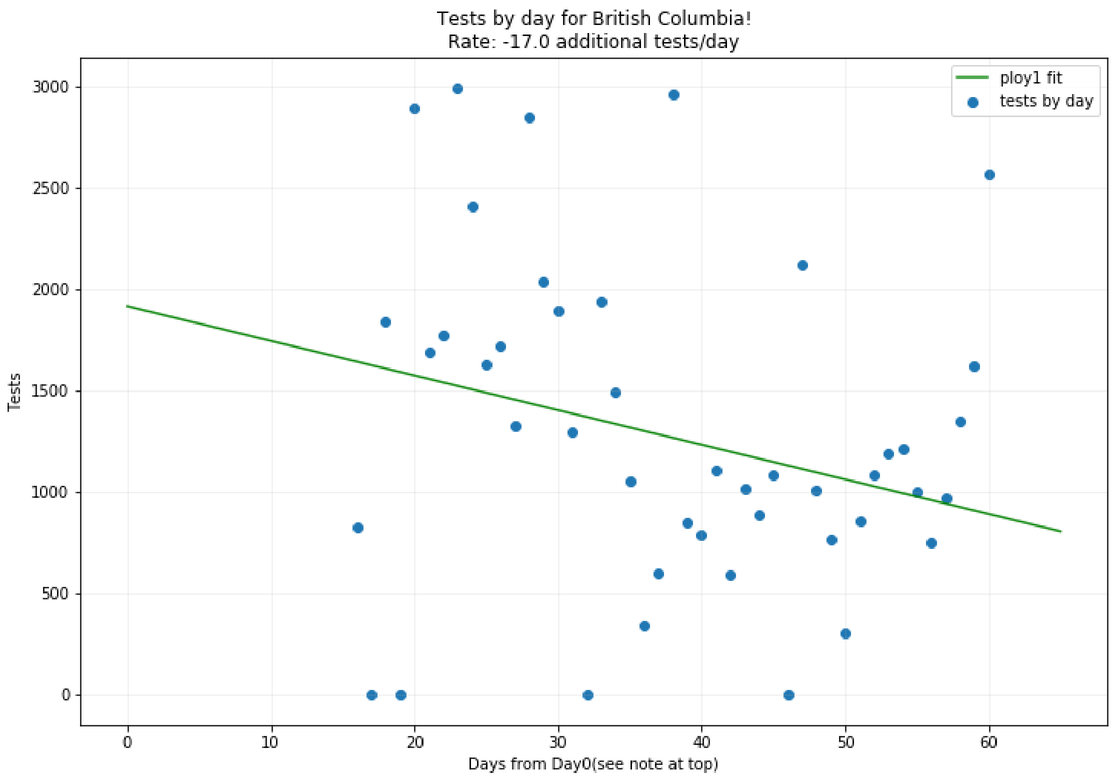
```
Precent of positive cases pre test: 3.437 %


------------- British Columbia-------------
dates:  [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16. 17.
 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35.
 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53.
 54. 55. 56. 57. 58. 59. 60. 61.]
cases:  [1.000e+00 4.000e+00 5.000e+00 6.000e+00 6.000e+00 7.000e+00 7.000e+00
 7.000e+00 7.000e+00 8.000e+00 1.200e+01 2.100e+01 2.100e+01 2.700e+01
 2.700e+01 3.200e+01 3.900e+01 4.600e+01 5.300e+01 6.400e+01 7.300e+01
 7.300e+01 1.030e+02 1.860e+02 2.710e+02 2.710e+02 4.240e+02 4.240e+02
 4.720e+02 6.170e+02 6.590e+02 7.250e+02 7.250e+02 8.840e+02 8.840e+02
 9.600e+02 9.700e+02 1.066e+03 1.121e+03 1.174e+03 1.203e+03 1.203e+03
 1.266e+03 1.291e+03 1.336e+03 1.370e+03 1.410e+03 1.445e+03 1.445e+03
 1.490e+03 1.517e+03 1.561e+03 1.561e+03 1.575e+03 1.618e+03 1.618e+03
 1.699e+03 1.724e+03 1.795e+03 1.824e+03 1.853e+03 1.853e+03]
-----
params PY:  [1.74198031e+03 1.51203429e-01 3.46895141e+01]
sigma PY:  [2.79375990e+01 6.84657441e-03 3.92174555e-01]
```



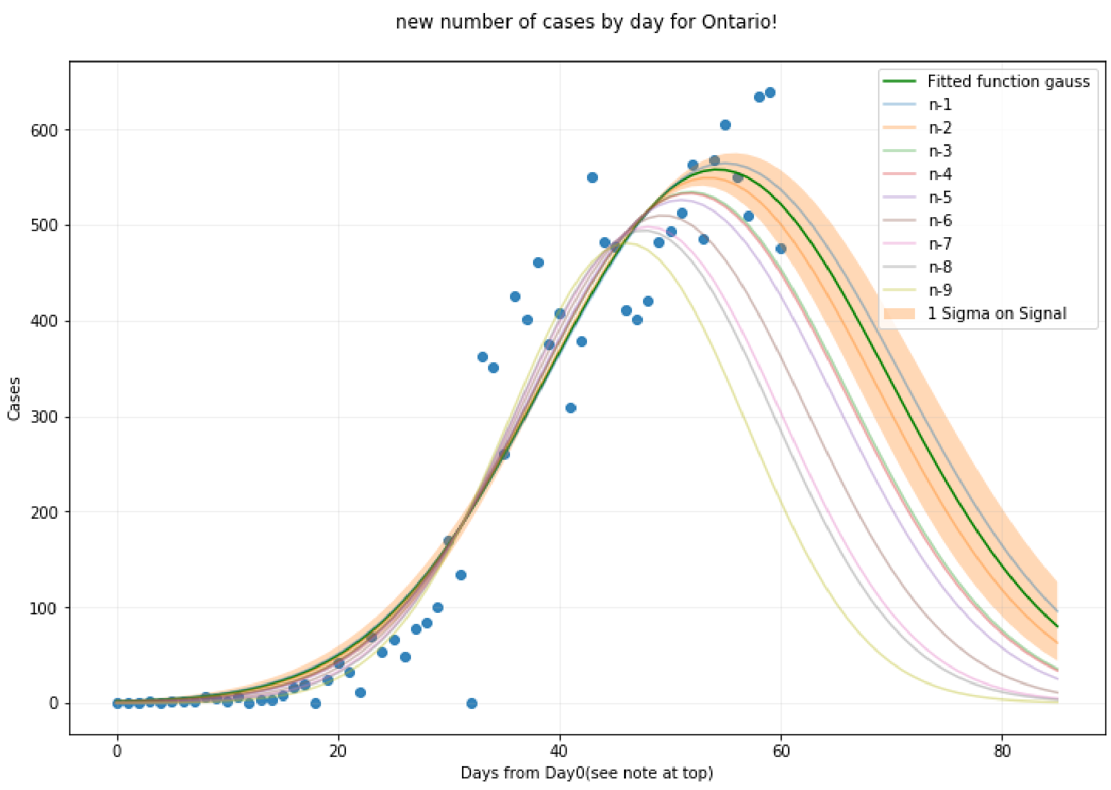
```
----
PY:
sigma+: 1748.0
n+1: 1714.0
Sigma-: 1680.0
----

params PY:  [1.74198031e+03 1.51203429e-01 3.46895141e+01]
PY sigma:  [2.79375990e+01 6.84657441e-03 3.92174555e-01]
max cases:  1742.0
infection point:  35.0
infection point date:  2020-03-06
```


Total number of cases for British Columbia!
max cases: ~1742.0
Fitted function Py
1 Day ago
3 Days ago
5 Days ago
10 Days ago
Data
Deaths
1 Sigma on Signal
1 Sigma on n-1
Cases
Days from Day0(see note at top)

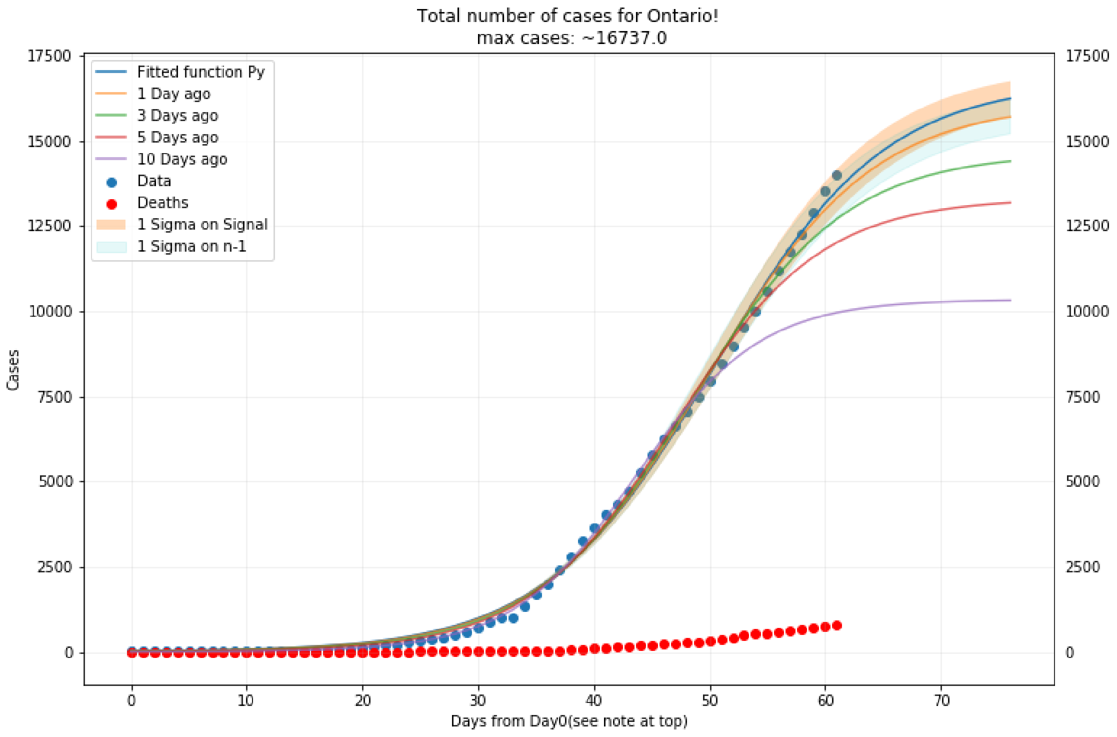

Total test to day in British Columbia!
Rate: 1275.0 tests/day
ploy1 fit
data
Tests
Days from Day0(see note at top)

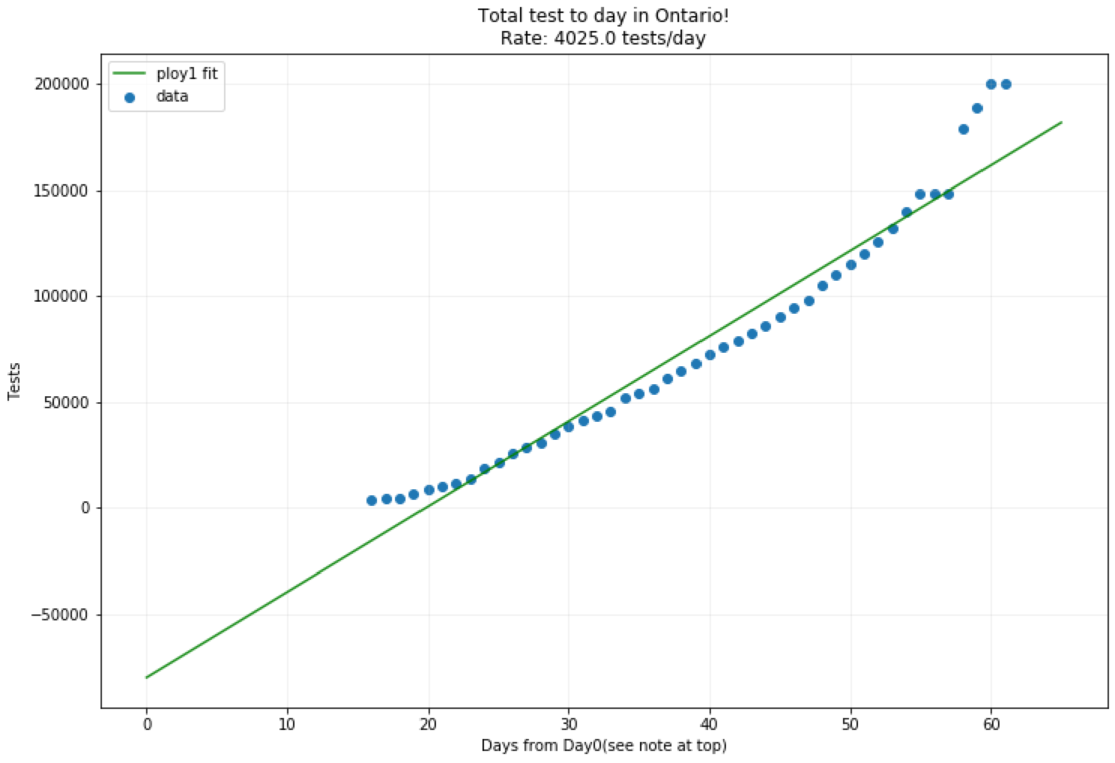


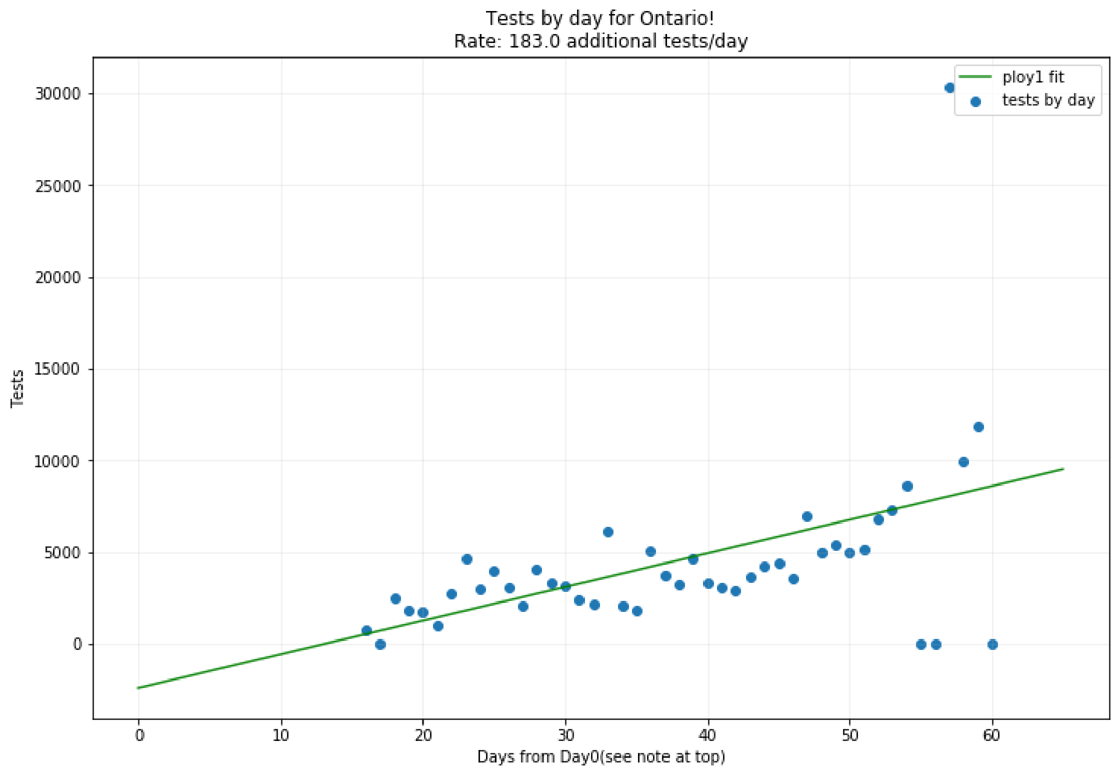
```
Precent of positive cases pre test: 2.939 %


------------- Ontario-------------
dates:  [ 0.   1.   2.   3.   4.   5.   6.   7.   8.   9.  10.  11.  12.  13.  14.  15.  16.  17.
 18.  19.  20.  21.  22.  23.  24.  25.  26.  27.  28.  29.  30.  31.  32.  33.  34.  35.
 36.  37.  38.  39.  40.  41.  42.  43.  44.  45.  46.  47.  48.  49.  50.  51.  52.  53.
 54.  55.  56.  57.  58.  59.  60.  61.]
cases:  [3.0000e+00 3.0000e+00 3.0000e+00 3.0000e+00 4.0000e+00 4.0000e+00
 5.0000e+00 6.0000e+00 8.0000e+00 1.5000e+01 2.0000e+01 2.2000e+01
 2.8000e+01 2.8000e+01 3.1000e+01 3.4000e+01 4.2000e+01 5.9000e+01
 7.9000e+01 7.9000e+01 1.0300e+02 1.4500e+02 1.7700e+02 1.8900e+02
 2.5800e+02 3.1100e+02 3.7700e+02 4.2500e+02 5.0300e+02 5.8800e+02
 6.8800e+02 8.5800e+02 9.9300e+02 9.9300e+02 1.3550e+03 1.7060e+03
 1.9660e+03 2.3920e+03 2.7930e+03 3.2550e+03 3.6300e+03 4.0380e+03
 4.3470e+03 4.7260e+03 5.2760e+03 5.7590e+03 6.2370e+03 6.6480e+03
 7.0490e+03 7.4700e+03 7.9530e+03 8.4470e+03 8.9610e+03 9.5250e+03
 1.0010e+04 1.0578e+04 1.1184e+04 1.1735e+04 1.2245e+04 1.2879e+04
 1.3519e+04 1.3995e+04]
-----
params PY:  [1.67369528e+04 1.35819148e-01 5.04192397e+01]
sigma PY:  [4.44204833e+02 3.72898713e-03 4.75104316e-01]
```



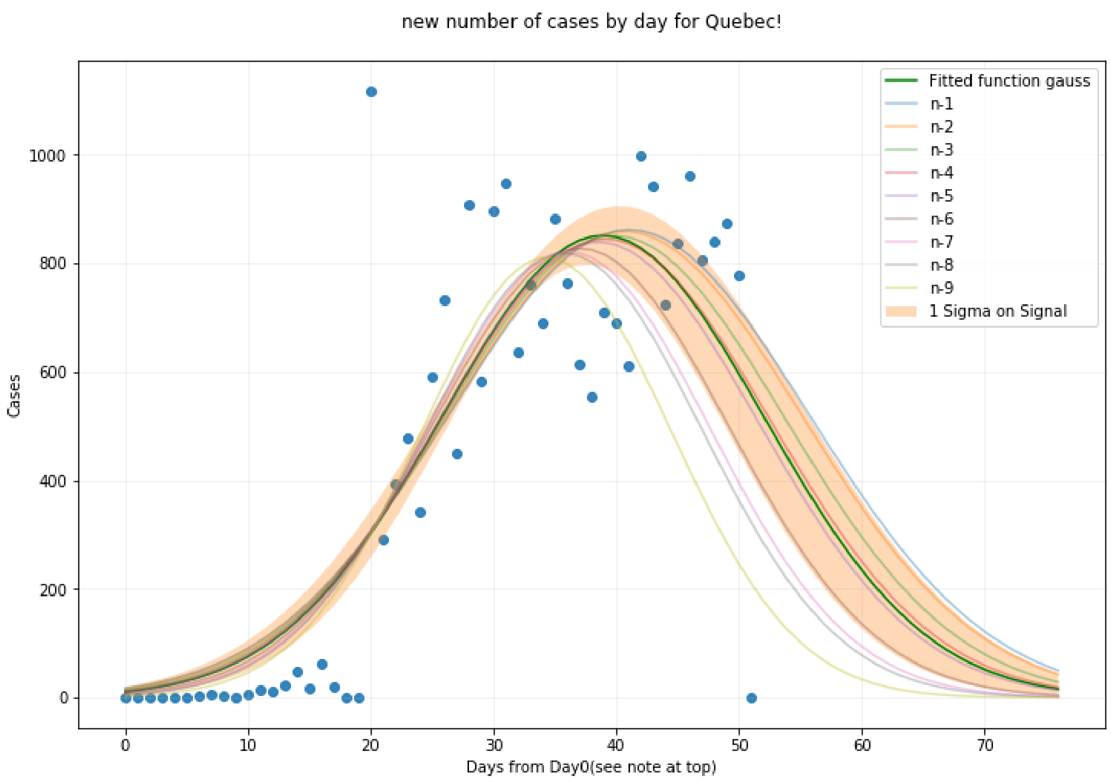
```
----
PY:
sigma+: 14487.0
n+1: 13861.0
Sigma-: 13239.0
----

params PY:  [1.67369528e+04 1.35819148e-01 5.04192397e+01]
PY sigma:  [4.44204833e+02 3.72898713e-03 4.75104316e-01]
max cases:  16737.0
infection point:  50.0
infection point date:  2020-03-21
```


Total number of cases for Ontario!
max cases: ~16737.0
Fitted function Py
1 Day ago
3 Days ago
5 Days ago
10 Days ago
Data
Deaths
1 Sigma on Signal
1 Sigma on n-1
Cases
Days from Day0(see note at top)
0
2500
5000
7500
10000
12500
15000
17500
0
10
20
30
40
50
60
70

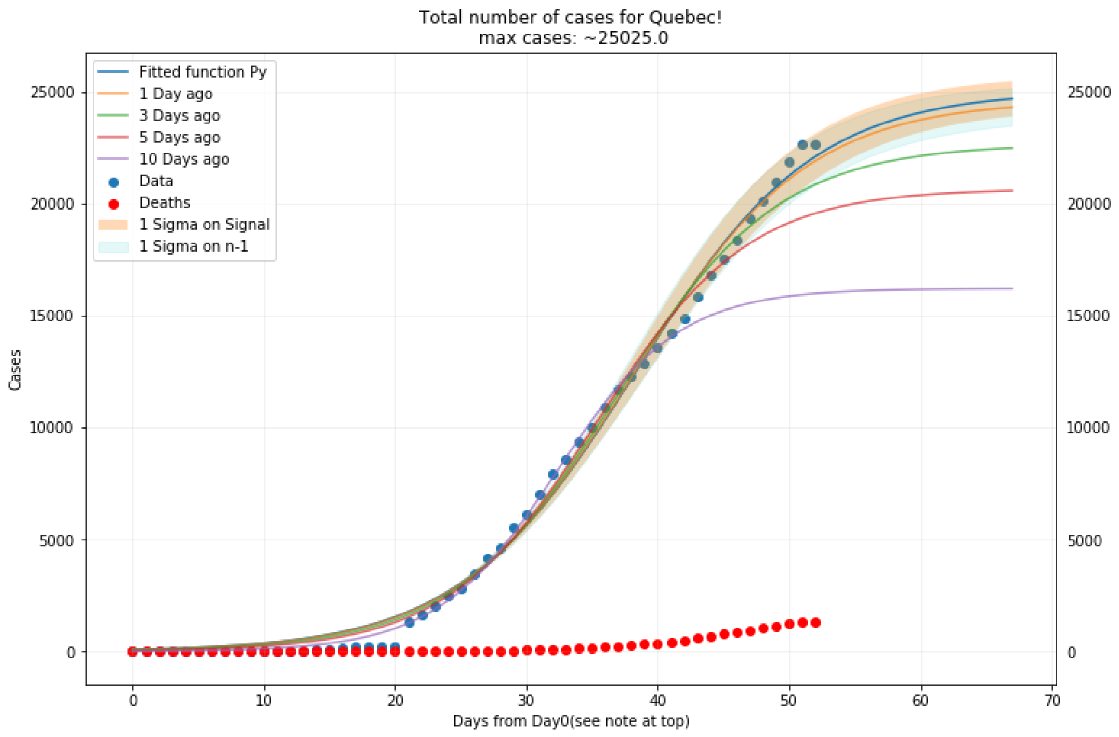

Total test to day in Ontario!
Rate: 4025.0 tests/day
ploy1 fit
data
Tests
Days from Day0(see note at top)
−50000
0
50000
100000
150000
200000
0
10
20
30
40
50
60



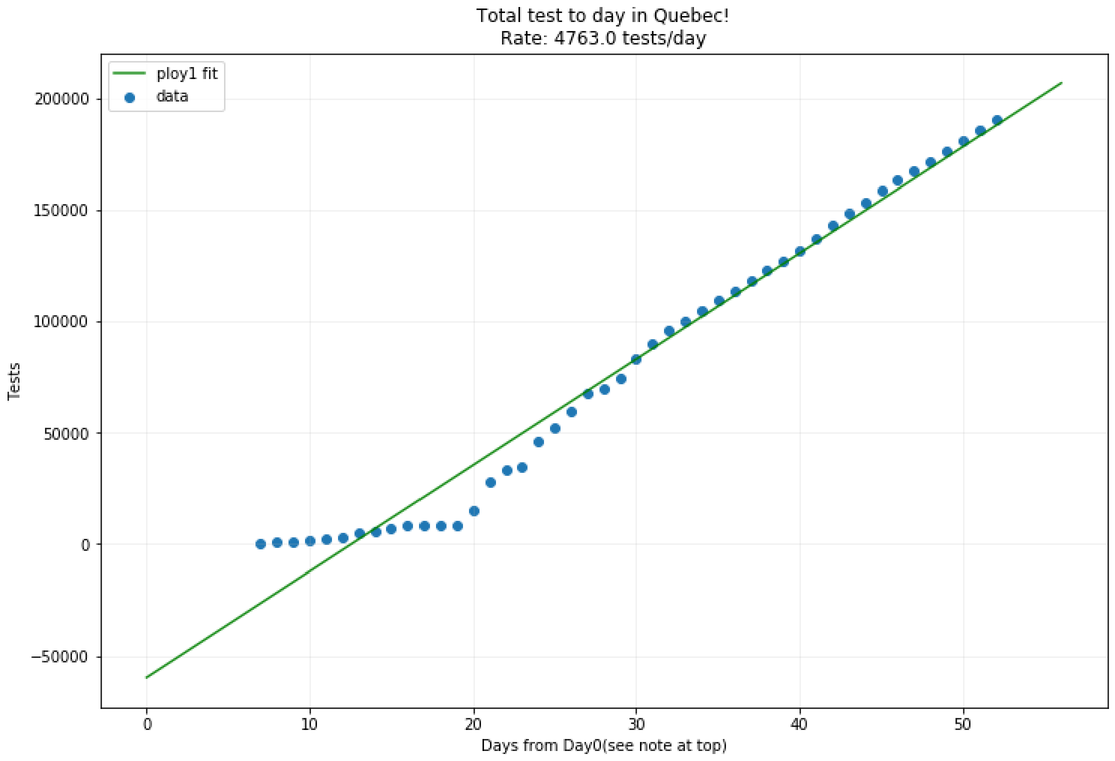
```
Precent of positive cases pre test: 6.988 %


------------- Quebec-------------
dates:  [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16. 17.
 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35.
 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52.]
cases:  [1.0000e+00 1.0000e+00 2.0000e+00 2.0000e+00 2.0000e+00 3.0000e+00
 4.0000e+00 7.0000e+00 1.3000e+01 1.7000e+01 1.7000e+01 2.4000e+01
 3.9000e+01 5.0000e+01 7.4000e+01 1.2100e+02 1.3900e+02 2.0200e+02
 2.2100e+02 2.2100e+02 2.2100e+02 1.3390e+03 1.6290e+03 2.0210e+03
 2.4980e+03 2.8400e+03 3.4300e+03 4.1620e+03 4.6110e+03 5.5180e+03
 6.1010e+03 6.9970e+03 7.9440e+03 8.5800e+03 9.3400e+03 1.0031e+04
 1.0912e+04 1.1677e+04 1.2292e+04 1.2846e+04 1.3557e+04 1.4248e+04
 1.4860e+04 1.5857e+04 1.6798e+04 1.7521e+04 1.8357e+04 1.9319e+04
 2.0126e+04 2.0965e+04 2.1838e+04 2.2616e+04 2.2616e+04]
-----
params PY:  [2.50250451e+04 1.48081185e-01 3.83765667e+01]
sigma PY:  [7.08214631e+02 5.90969513e-03 5.13177062e-01]
```



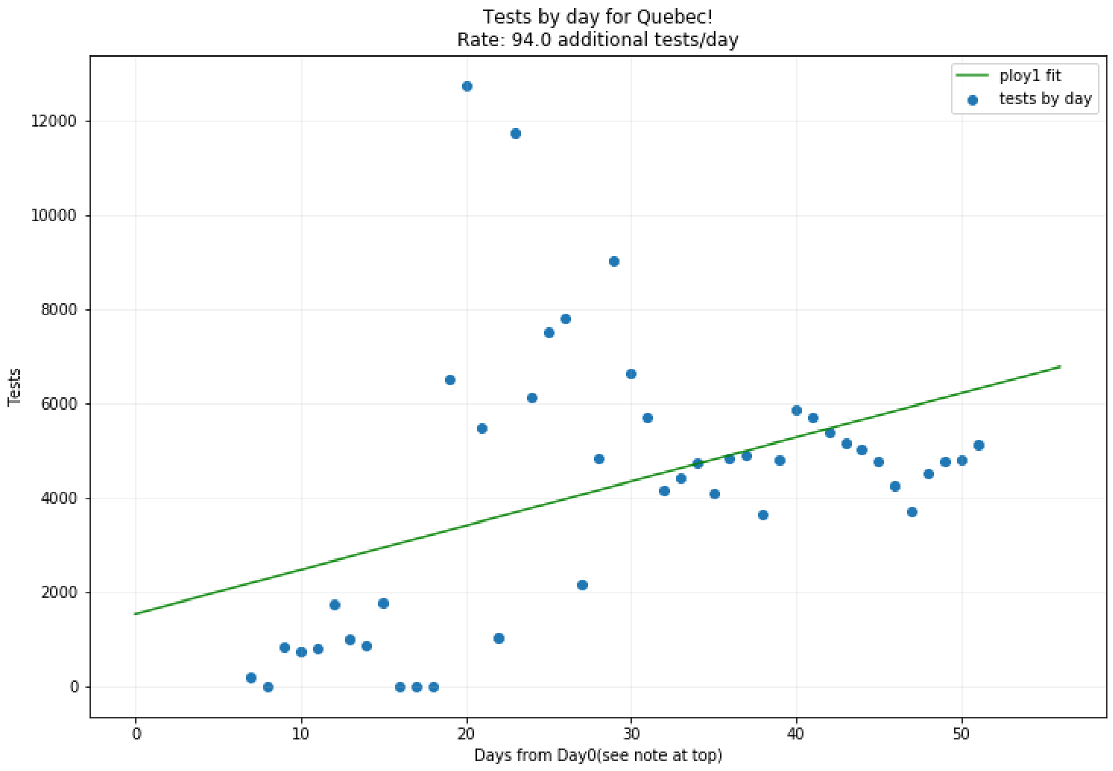
```
----
PY:
sigma+: 23453.0
n+1: 22450.0
Sigma-: 21434.0
----

params PY:  [2.50250451e+04 1.48081185e-01 3.83765667e+01]
PY sigma:  [7.08214631e+02 5.90969513e-03 5.13177062e-01]
max cases:  25025.0
infection point:  38.0
infection point date:  2020-03-09
```



Total number of cases for Quebec!
max cases: ~25025.0
Fitted function Py
1 Day ago
3 Days ago
5 Days ago
10 Days ago
Data
Deaths
1 Sigma on Signal
1 Sigma on n-1
Cases
Days from Day0(see note at top)



Total test to day in Quebec!
Rate: 4763.0 tests/day
ploy1 fit
data
Tests
Days from Day0(see note at top)



```
Precent of positive cases pre test: 11.855 %
```

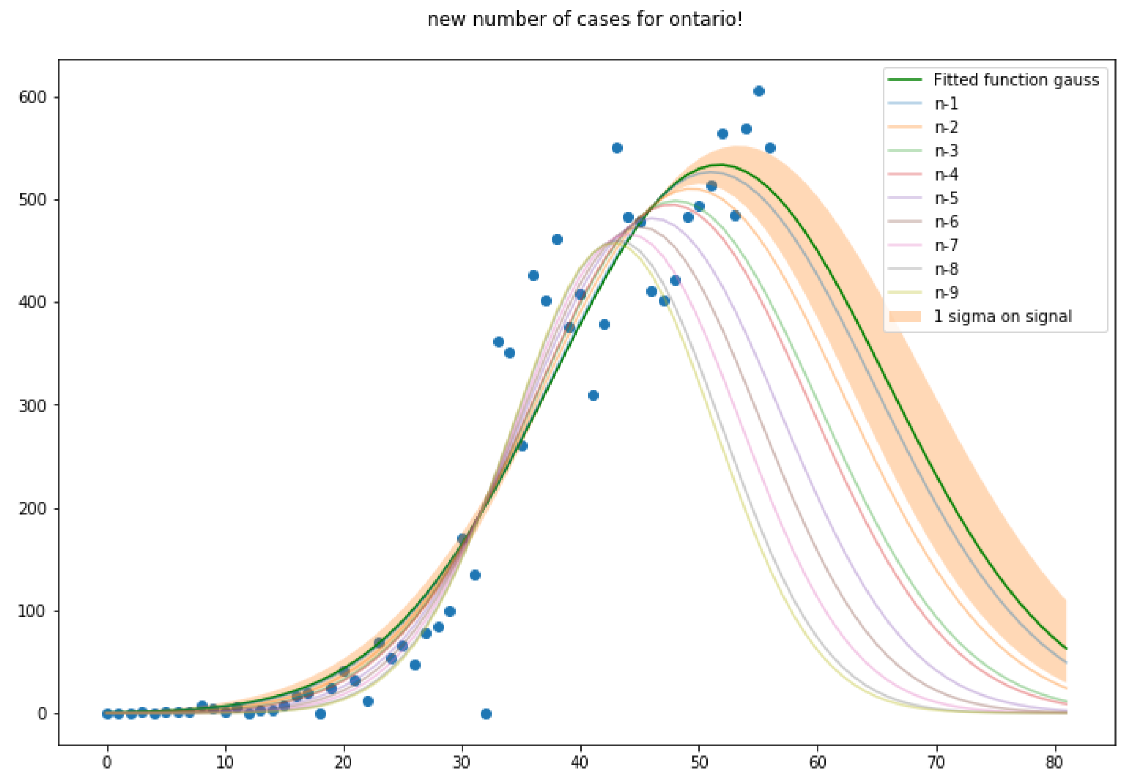
```
In [ ]:
```

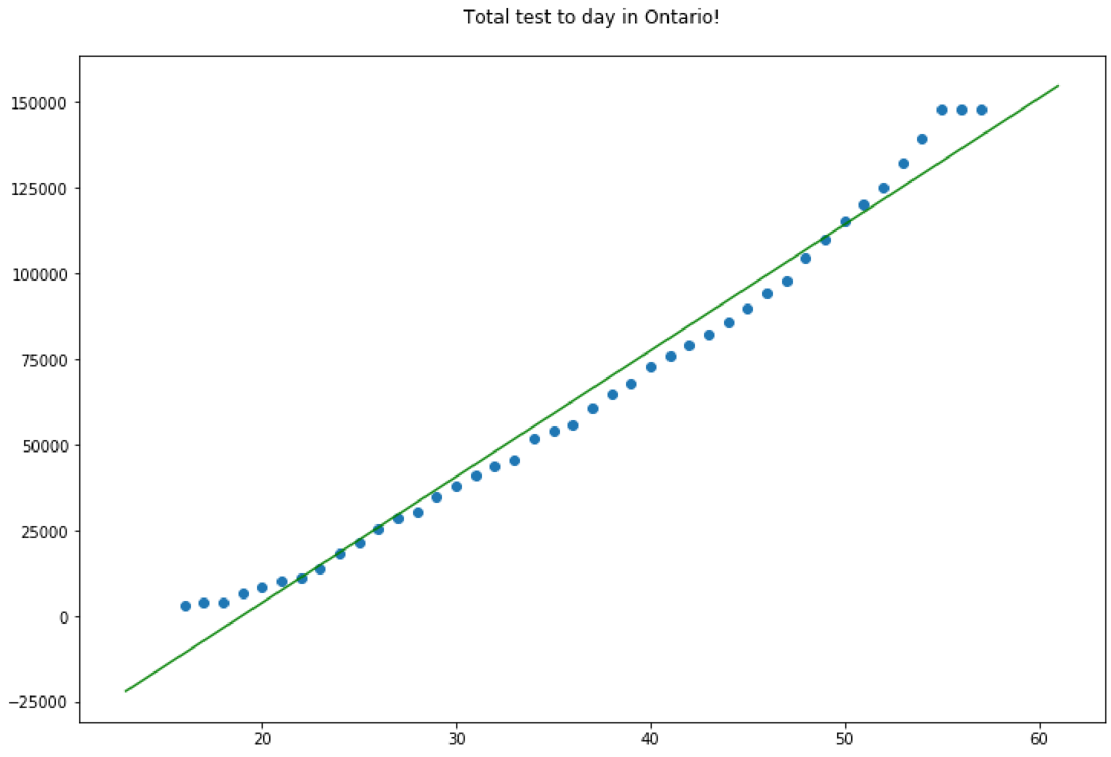
```
In [ ]:
```

1.2 Testing features

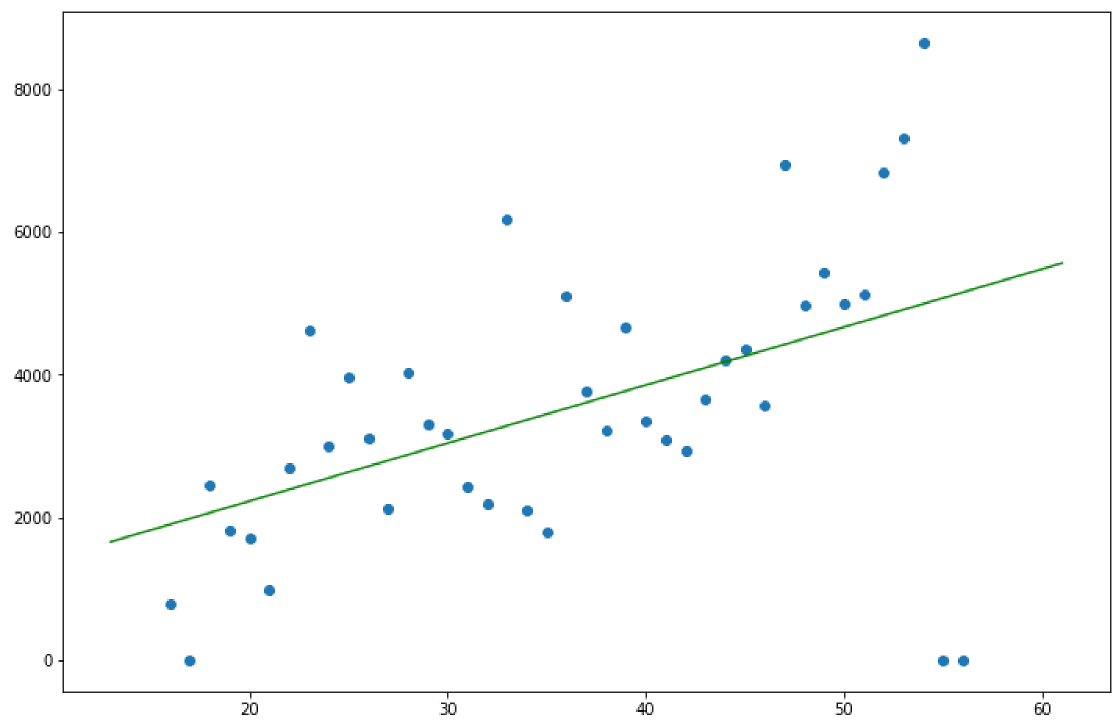
```
In [662]: def Gauss(x, a, x0, sigma):
              return a * np.exp(-(x - x0)**2 / (2 * sigma**2))


          canada_Data="https://health-infobase.canada.ca/src/data/covidLive/covid19.csv"
          data = pd.read_csv(canada_Data)

          dataName="Ontario"
          #      ont=givemedata(data,where)
          ont=CanadaData(data,dataName)
          totaltested=CanadaDatatests(data,dataName)
          deaths=CanadaDataDeaths(data,dataName)

          death_dates=np.arange(0,len(deaths)).astype(float)
```

```
delta_testes=np.diff(totaltested)[:] #total tested
d_tested_dates=np.arange(0,len(delta_testes))

delta_cases=np.diff(ont)[:] #cases
d_dates=np.arange(0,len(delta_cases))




# bins=np.arange(d_dates[0],d_dates[-1],len(d_dates)/15)

# delta_cases=np.digitize(delta_cases,25)
# print (delta_cases)
# print (bins)
# print (np.digitize(delta_cases,bins))




delta_dates=np.arange(0,len(delta_cases)).astype(float)
delta_x_future=np.arange(0,len(delta_cases)+55).astype(float)

# popt_ex, pcov_ex = curve_fit(lambda x,a,b: a*np.exp(b*x), delta_dates, delta_cases

mean = sum(delta_dates * delta_cases) / sum(delta_cases)
sigma = np.sqrt(sum(delta_cases * (delta_dates - mean)**2) / sum(delta_cases))
popt,pcov = curve_fit(Gauss, delta_dates, delta_cases, p0=[max(delta_cases), mean, s
popt1,pcov1 = curve_fit(Gauss, delta_dates[0:-1], delta_cases[0:-1], p0=[max(delta_ca
popt2,pcov2 = curve_fit(Gauss, delta_dates[0:-2], delta_cases[0:-2], p0=[max(delta_ca
popt3,pcov3 = curve_fit(Gauss, delta_dates[0:-3], delta_cases[0:-3], p0=[max(delta_ca
popt4,pcov4 = curve_fit(Gauss, delta_dates[0:-4], delta_cases[0:-4], p0=[max(delta_ca
popt5,pcov5 = curve_fit(Gauss, delta_dates[0:-5], delta_cases[0:-5], p0=[max(delta_ca
popt6,pcov6 = curve_fit(Gauss, delta_dates[0:-6], delta_cases[0:-6], p0=[max(delta_ca
popt7,pcov7 = curve_fit(Gauss, delta_dates[0:-7], delta_cases[0:-7], p0=[max(delta_ca
popt8,pcov8 = curve_fit(Gauss, delta_dates[0:-8], delta_cases[0:-8], p0=[max(delta_ca
popt9,pcov9 = curve_fit(Gauss, delta_dates[0:-9], delta_cases[0:-9], p0=[max(delta_ca



x_pred_delta=np.arange(0,len(delta_cases)+25).astype(float)
sigmaG = np.sqrt(np.diag(pcov))

y_pred_plusG = Gauss(x_pred_delta, popt[0]+sigmaG[0], popt[1]+sigmaG[1], popt[2]+sigm
y_pred_minusG = Gauss(x_pred_delta, popt[0]-sigmaG[0], popt[1]-sigmaG[1], popt[2]-si


plt.figure(figsize=(12, 8))
plt.title("new number of cases for "+"ontario"+"!\n")
plt.scatter(delta_dates, delta_cases)
plt.plot(x_pred_delta, Gauss(x_pred_delta, popt[0], popt[1], popt[2]),
             label='Fitted function gauss', color='green')

plt.plot(x_pred_delta, Gauss(x_pred_delta, popt1[0], popt1[1], popt1[2]), alpha=0.4,
             label='n-1')
plt.plot(x_pred_delta, Gauss(x_pred_delta, popt2[0], popt2[1], popt2[2]), alpha=0.4,
             label='n-2')
plt.plot(x_pred_delta, Gauss(x_pred_delta, popt3[0], popt3[1], popt3[2]), alpha=0.4,
             label='n-3')
plt.plot(x_pred_delta, Gauss(x_pred_delta, popt4[0], popt4[1], popt4[2]), alpha=0.4,
             label='n-4')
plt.plot(x_pred_delta, Gauss(x_pred_delta, popt5[0], popt5[1], popt5[2]), alpha=0.4,
             label='n-5')
plt.plot(x_pred_delta, Gauss(x_pred_delta, popt6[0], popt6[1], popt6[2]), alpha=0.4,
             label='n-6')
plt.plot(x_pred_delta, Gauss(x_pred_delta, popt7[0], popt7[1], popt7[2]), alpha=0.4,
             label='n-7')
plt.plot(x_pred_delta, Gauss(x_pred_delta, popt8[0], popt8[1], popt8[2]), alpha=0.4,
             label='n-8')
plt.plot(x_pred_delta, Gauss(x_pred_delta, popt9[0], popt9[1], popt9[2]), alpha=0.4,
             label='n-9')

plt.fill_between(x_pred_delta, y_pred_minusG,y_pred_plusG, alpha=0.3,label='1 sigma c

plt.legend(loc='best')

#-------------------tests-------------------
x_pred_delta_tests=np.arange(13,len(delta_cases)+5).astype(float)

tested_dates=np.arange(0,totaltested.size)
valid = ~(np.isnan(tested_dates) | np.isnan(totaltested))
popt_line_t, pcov_line_t = curve_fit(Linefunc, tested_dates[valid], totaltested[vali

plt.figure(figsize=(12,8))
plt.title("Total test to day in "+dataName+"!\n")
plt.scatter(np.arange(0,totaltested.size),totaltested)
plt.plot(x_pred_delta_tests, Linefunc(x_pred_delta_tests, popt_line_t[0], popt_line_

#--
valid = ~(np.isnan(d_tested_dates) | np.isnan(delta_testes))
popt_line, pcov_line = curve_fit(Linefunc, d_tested_dates[valid], delta_testes[valid]

plt.figure(figsize=(12, 8))
plt.title("Tests by day for "+dataName+"!\n")
plt.scatter( d_tested_dates,delta_testes, label='tests by day')
plt.plot(x_pred_delta_tests, Linefunc(x_pred_delta_tests, popt_line[0], popt_line[1]
Out[662]: [<matplotlib.lines.Line2D at 0x1a39dbfdd8>]
```


new number of cases for ontario!
Fitted function gauss
n-1
n-2
n-3
n-4
n-5
n-6
n-7
n-8
n-9
1 sigma on signal
0
100
200
300
400
500
600
0
10
20
30
40
50
60
70
80


Total test to day in Ontario!
−25000
0
25000
50000
75000
100000
125000
150000
20
30
40
50
60

Tests by day for Ontario!



```
In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```