

Four fitting models comparison for SARS-nCoV-2 total cumulative cases curves

Max Pierini*

May 3, 2020

1 Abstract

Four different models have been compared for fitting SARS-nCoV-2 total cumulative cases curves in 187 countries over a period of 102 days. Evaluated models have been: *Simple Logistic Function* (**SLF**), *Simple Gompertz Function* (**SGF**), *Double Logistic Function* (**DLF**) and *Double Gompertz Function* (**DGF**). **DGF** model showed lower MSE, RMSE, NRMSE, MAE, NAE and higher Pearson R compared to the others. R^2 ($p > .99$), R^2_{adj} ($p > .99$), ΔAIC (> 0) and ΔBIC ($p < .5$) showed higher percentages for **DGF** compared with the others.

Results suggest that *Double Gompertz Function* may be a good fitting model for SARS-nCoV-2 cumulative cases curve.

2 Methods

2.1 Data

SARS-nCoV-2 total cumulative cases data have been gathered from Johns Hopkins University GitHub repository [1] and summed into single countries where regional level was provided [2]. Data have been used “as is” without rejecting any outlier and/or error (negative daily Δ). Data and results have been stored in a **pandas** n-dimensional **DataFrame**.

Raw data contained 187 countries and daily cumulative confirmed cases for 102 days, from 2020-01-22 to 2020-05-03.

2.2 Models

Models have been defined with **lmfit** (implementation of classical **curve_fit** in **scipy**) using Nelder-Mead method for fitting [2].

Total residual from each function have been initially compared (unsorted, sorted, gaussian distribution) to find the model with residual μ closer to 0 and shorter σ . *Akaike Information Criterion* differences (ΔAIC) in relative probability density space, *Bayesian Information Criterion* differences (ΔBIC), R^2 and R^2_{adj} coefficients mean have been used to find the likely better fitting model that has been finally compared, country by country. See Section 7 for formulae.

Models have been defined as follow:

*info@maxpierini.it

- Simple Logistic Function (**SLF**):

```
def simple_logistic_function(x, a, b, k, e):
    d = k * (b - np.array(x))
    return (a / (1 + np.exp(d))) + e
```

$$f(t) = \frac{a}{1 + e^{k(b-t)}} + \varepsilon$$

- Double Logistic Function (**DLF**):

```
def double_logistic_function(x, a1, b1, k1, a2, b2, k2, e):
    d1 = k1 * (b1 - np.array(x))
    g1 = a1 / (1 + np.exp(d1))
    d2 = k2 * (b2 - np.array(x))
    g2 = (a2 - a1) / (1 + np.exp(d2))
    return g1 + g2 + e
```

$$f(t) = \frac{a_1}{1 + e^{k_1(b_1-t)}} + \frac{a_2 - a_1}{1 + e^{k_2(b_2-t)}} + \varepsilon$$

- Simple Gompertz Function (**SGF**):

```
def simple_gompertz_function(x, a, b, k, e):
    exp = - np.exp(k * (b - x))
    return a * np.exp(exp) + e
```

$$f(t) = a \cdot e^{-e^{k(b-t)}} + \varepsilon$$

- Double Gompertz Function (**DGF**):

```
def double_gompertz_function(x, a1, b1, k1, a2, b2, k2, e):
    exp1 = - np.exp(k1 * (b1 - x))
    g1 = a1 * np.exp(exp1)
    exp2 = - np.exp(k2 * (b2 - x))
    g2 = (a2 - a1) * np.exp(exp2)
    return g1 + g2 + e
```

$$f(t) = a_1 \cdot e^{-e^{k_1(b_1-t)}} + (a_2 - a_1) \cdot e^{-e^{k_2(b_2-t)}} + \varepsilon$$

3 Model fitting

Fitting has been performed with `lmfit` using Nelder-Mead method

```
model = lmfit.Model(function)
result = model.fit(data=y, params=p, x=x, method='Nelder', nan_policy='omit')
```

initial parameters `p` have been guessed as follows (where y are observed values):

- **SLF** and **SGF**

```

p = model.make_params(
    a=y[-1],
    b=max_y_i,
    k=.1,
    e=y[0]
)

```

$$a = y_{-1}$$

$$b = x_{\max(dy)}$$

$$k = 0.1$$

$$\varepsilon = y_0$$

- **DLF and DGF**

```

p = model.make_params(
    a1=y[max_y_i] * 2,
    b1=max_y_i,
    k1=.1,
    a2=max(y),
    b2=len(y),
    k2=.1,
    e=y[0]
)

```

$$a_1 = 2y_{\max(dy)}$$

$$b_1 = x_{\max(dy)}$$

$$k_1 = 0.1$$

$$a_2 = \max(y)$$

$$b_2 = x_{y_{-1}}$$

$$k_2 = 0.1$$

$$\varepsilon = y_0$$

Fitting failed for 0 countries returning best fit information from 187 countries, for a total of 19074 observed and 76296 predicted values.

Complete `python` backend for data gathering, fitting and analysis along with a `pickle` saved dataframe of all measured data and results is online available [2].

Fitting examples are reported in figures [REF] [REF] [REF].

4 Analysis

Several skill score have been used to evaluate to average skill and skill interpolating extreme values (outliers).

Mean absolute error (**MAE**) is a natural, unambiguous, measure of average error [3]. It shows the errors in the same unit as variables themselves. MAE is bounded below by 0 (best case) and unbounded above. Advantage over Mean Bias Error (**MBE**) is that, taking absolute error values, positive and negative errors can't cancel out [4].

Normalized Absolute Error (**NAE**), bounded below 0 (best case) and “virtually” unbounded above. If more than 1 errors are greater than observed values themselves. Advantage over Normalized Bias Error (**NBE**) is that, taking absolute error values, positive and negative errors can't cancel out. [REF]

Mean Squared Error (**MSE**), variance, taking the square of residual is highly sensitive to large outliers [4]. Bounded below 0 (best case) and unbounded above.

Root Mean Squared Error (**RMSE**) is very commonly used as a measure of deviation from the observed value. Although it has been criticized as being ambiguous [3] and its dependence on the squared error means that it is not resistant to outliers deviating from a Gaussian distribution. It has been included because of its sensitivity to large outliers. Bounded below 0 (best case) and unbounded above.

Normalized Root Mean Squared Error (**NRMSE**) allows to compare **RMSE** of different models, normalized on observed values, on a $(0, 1]$ scale [REF].

Pearson Correlation coefficient (**Pearson R**) depends on squared deviations and so is not a resistant measure. However, this statistic removes the effect of any bias in the interpolated data. Problems with correctly capturing the variance will not be highlighted as the measure normalizes the observed and modeled values by their standard deviations [4]. The statistic is standardized. However, because of its insensitivity to biases and errors in variance, the correlation coefficient should be considered as a measure of potential skill [5].

R^2 coefficient test ... Bounded from 0 to 1, $R^2 > .99$ has been fixed to evaluate model H_0 against alternative model H_1 [6].

R_{adj}^2 coefficient test ... Bounded from 0 to 1, $R_{adj}^2 > .99$ has been fixed to evaluate model H_0 against alternative model H_1 [6].

Akaike Information Criterion (**AIC**), its score (or weight) and its relative probability distribution space ... Within the probability distribution space, bounded from 0 to 1, a $\Delta\mathbf{AIC}_p$ value less than 0.5 means model H_0 has more chances to be better fitting than alternative model H_1 [7].

Bayesian Information Criterion (**BIC**) and its delta ... A $\Delta\mathbf{BIC} > 0$ means model H_0 has evidence to be better fitting than alternative model H_1 . Look in Appendix for formulae [8].

4.1 Analysis Example

Example of analysis of two fitting models for a noisy observed sample.

Observed o values are generated by

$$o = f(x) = a \cdot x^2 + b + \varepsilon_x$$

where $x = (0, 100]$, $a = 2.3$, $b = 3000$ and ε_x is a random noise $(0, 1000]$.

Observed are fitted with two models using `scipy.optimize.curve_fit` (least squares method):

$$e_1(x) = a \cdot x^2$$

and

$$e_2(x) = a \cdot x^2 + b$$

Model e_2 is taken as null hypothesis H_0 .

```
import numpy as np
from scipy import stats as sts
from scipy.optimize import curve_fit
import sklearn.metrics as skl
from matplotlib import pyplot as plt

def AIC_test(diff):
    try:
        return np.exp(-.5 * diff) / (1 + np.exp(-.5 * diff))
    except Exception as err:
        if diff > 0:
            return 1.
        return 0.

def func_noise(x, a, b, err=False):
    noise = np.random.normal(0, 1000, len(x))
    return a * np.array(x) ** 2 + b + (noise if err else 0)

def func(x, a):
    return a * np.array(x) ** 2

def func2(x, a, b):
    return b + a * np.array(x) ** 2

def report(o, E, P):
    mae, nae, mbe, nbe, mse, rmse, nrmse, r, r2, r2a, aic, bic = (
        [] for _ in range(12)
    )
    print("
                e1 H1                e2 H0")
    print("-----")
    for i, e in enumerate(E):
```

```

mae.append(skl.mean_absolute_error(o, e))
nae.append(np.sum(np.abs(o - e)) / np.sum(o))
mbe.append(np.sum(o - e) / len(x))
nbe.append(np.sum(o - e) / np.sum(o))
_mse = skl.mean_squared_error(o, e)
mse.append(_mse)
_rmse = np.sqrt(_mse)
rmse.append(_rmse)
_nrmse = _rmse / np.sum(o)
nrmse.append(_nrmse)
r.append(sts.pearsonr(o, e)[0])
r2.append(skl.r2_score(o, e))
rss = np.sum(np.abs(o - e) ** 2)
tss = np.sum(np.abs(o - np.mean(o)) ** 2)
r2a.append(1 - (rss / tss) * ((len(x) - 1) / (len(x) - P[i] - 1)))
aic.append(len(x) * np.log(rss / len(x)) + 2 * 1)
bic.append(len(x) * np.log(rss / len(x)) + np.log(len(x)) * P[i])
print(f" MAE: {mae[0]:>14.6f} {mae[1]:>14.6f}")
print(f" NAE: {nae[0]:>14.6f} {nae[1]:>14.6f}")
print(f" BME: {mbe[0]:>14.9f} {mbe[1]:>14.6e}")
print(f" NBE: {nbe[0]:>14.9f} {nbe[1]:>14.6e}")
print(f" MSE: {mse[0]:>14.6f} {mse[1]:>14.5f}")
print(f" RMSE: {rmse[0]:>14.6f} {rmse[1]:>14.6f}")
print(f" NRMSE: {nrmse[0]:>14.9f} {nrmse[1]:>14.9f}")
print(f" R: {r[0]:>14.9f} {r[1]:>14.9f}")
print(f" R2: {r2[0]:>14.9f} {r2[1]:>14.9f}")
print(f" R2adj: {r2a[0]:>14.9f} {r2a[1]:>14.9f}")
print(f" AIC: {aic[0]:>14.6f} {aic[1]:>14.6f} p:{AIC_test(aic[0]-aic[1]):.3e}")
print(f" BIC: {bic[0]:>14.6f} {bic[1]:>14.6f} \U00000394:{bic[0] - bic[1]:.5f}")

x = np.arange(0, 100)
o = func_noise(x, 2.3, 3e3, err=True)
popt, pcov = curve_fit(func, x, o)
e1 = func(x, *popt)
popt, pcov = curve_fit(func2, x, o)
e2 = func2(x, *popt)

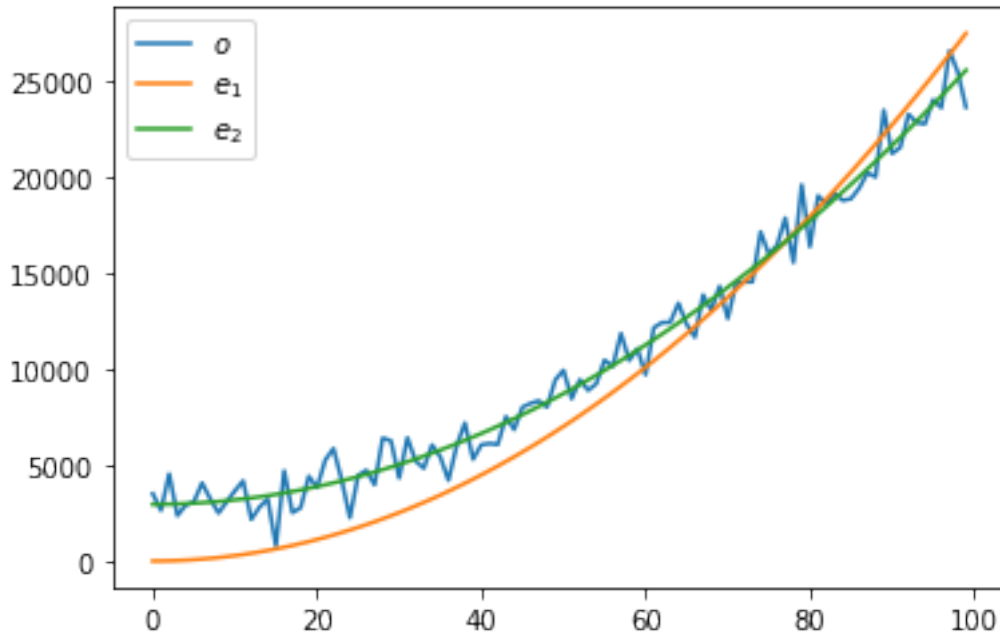
report(o, [e1, e2], [1, 2])

plt.plot(x, o, label="o")
plt.plot(x, e1, label="e1")
plt.plot(x, e2, label="e2")
plt.legend(loc="best")
plt.show()

```

Fitting and analysis report (figure [REF]).

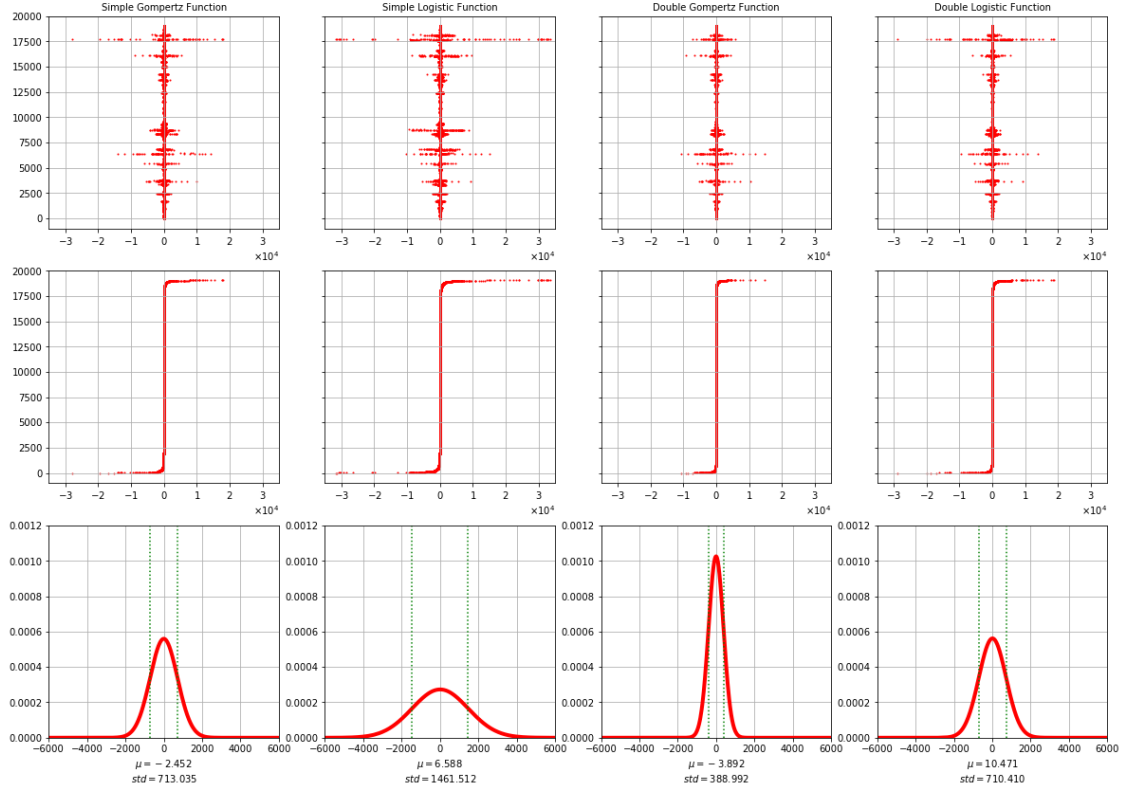
	e1 H1	e2 H0	
MAE:	1885.220348	740.113763	
NAE:	0.179196	0.070350	
BME:	1319.888763584	-1.578101e-08	
NBE:	0.125459752	-1.500037e-12	
MSE:	4783806.846386	888251.64274	
RMSE:	2187.191543	942.471030	
NRMSE:	0.002078997	0.000895850	
R:	2.2323166e-86	2.2323166e-86	Δ : -1.279074e-98
R2:	0.898722476	0.981194908	
R2adj:	0.897689032	0.980807174	
AIC:	1540.074720	1371.701036	p: 2.742327e-37
BIC:	1542.679890	1378.911377	Δ : 163.7685134



4.2 Residual

Total residual from each model have been collected and compared to get a first “rough” evidence of the most likely better fitting model [FIG].

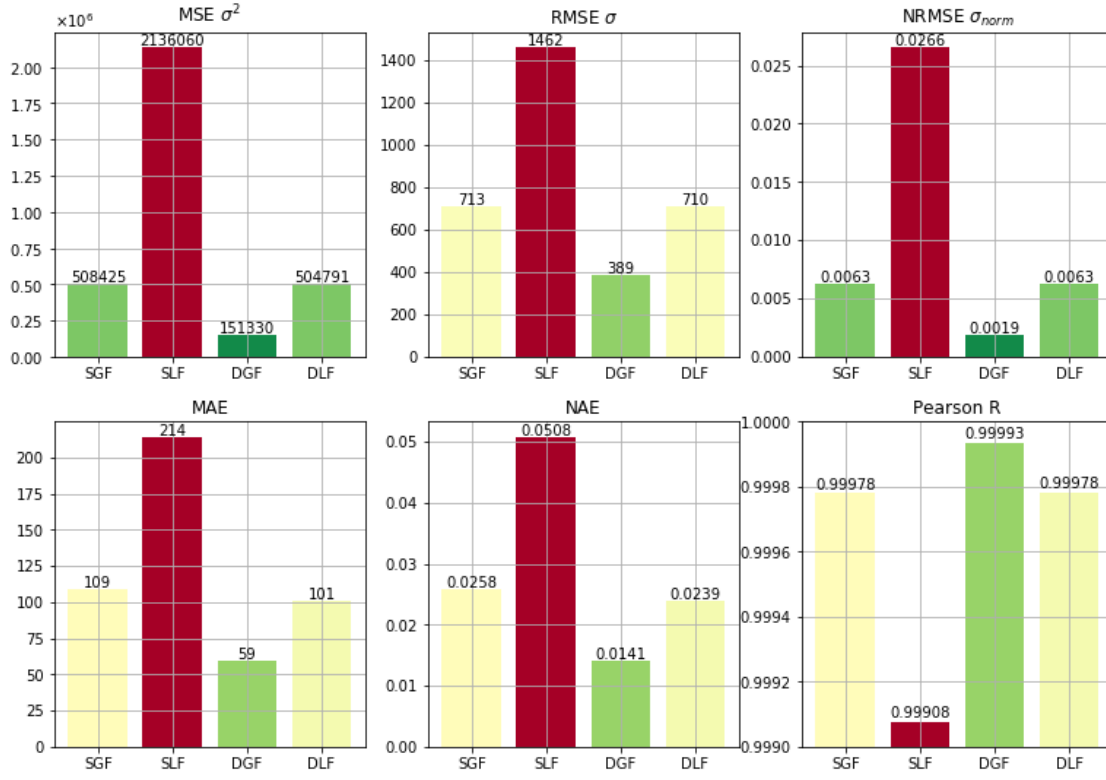
Models Residual



DGF showed the lower residual standard deviation while the mean of all four models has been very close to 0.

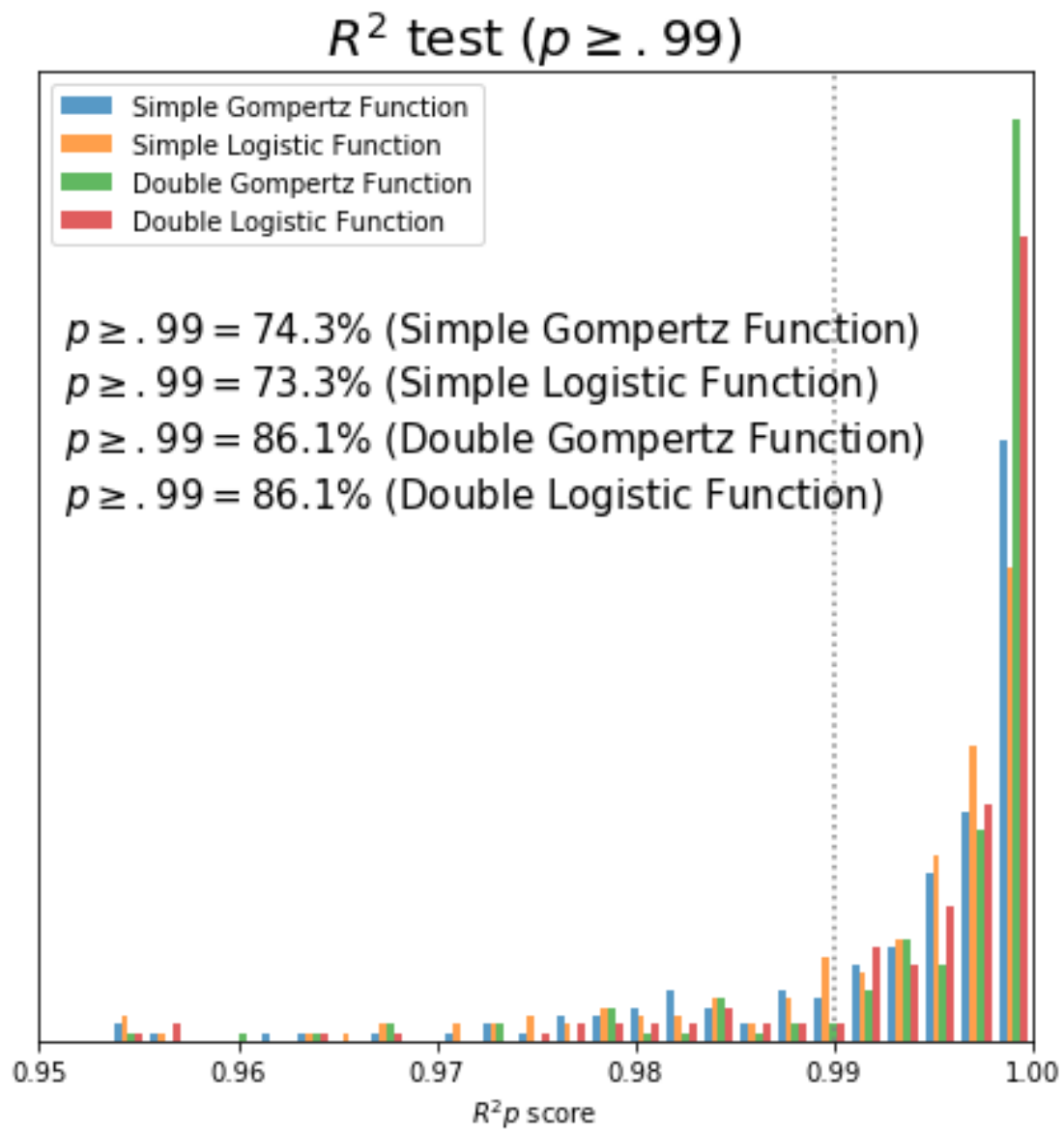
Mean Squared Error (**MSE**, Variance), Root Mean Squared Error (**RMSE**, Standard Deviation), Normalize Root Mean Squared Error (**NRMSE**, Normalized Standard Deviation), Mean Absolute Error (**MAE**), Normalized Absolute Error (**NAE**) and Pearson Correlation Coefficient (**Pearson R**) have been computed for all models residual Section 4.3 (see Appendix for formulae Section 7). **DGF** showed the best results for all values confirming the first null hypothesis that could have been the best fitting model among the chosen ones.

4.3 Errors tests

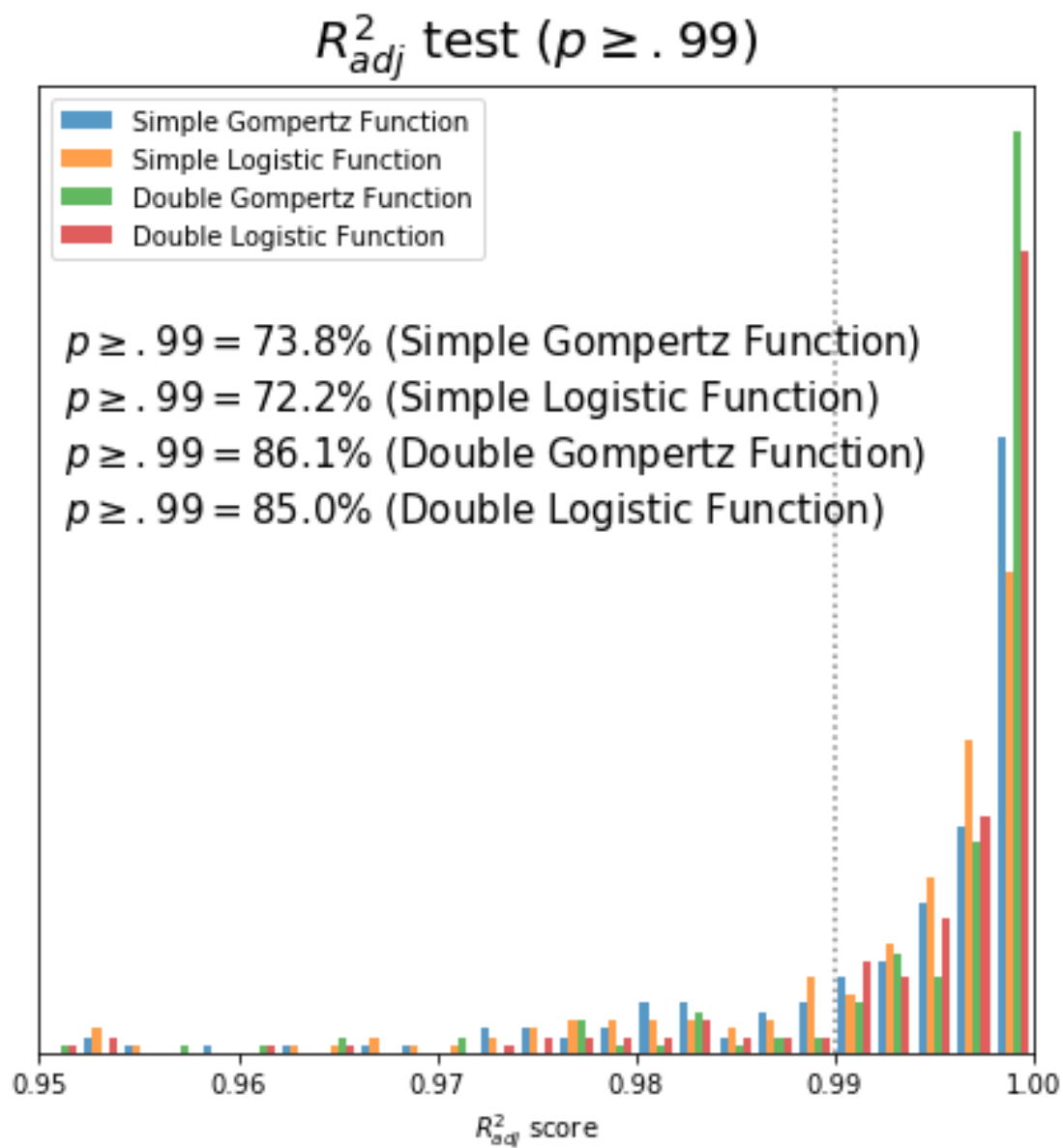


Coefficient of Determination R^2 , Adjusted Coefficient of Determination R^2_{adj} , Akaike Information Criterion (**AIC**) and Bayesian Information Criterion (**BIC**) have computed and collected from all fits and compared with each other.

4.4 R²

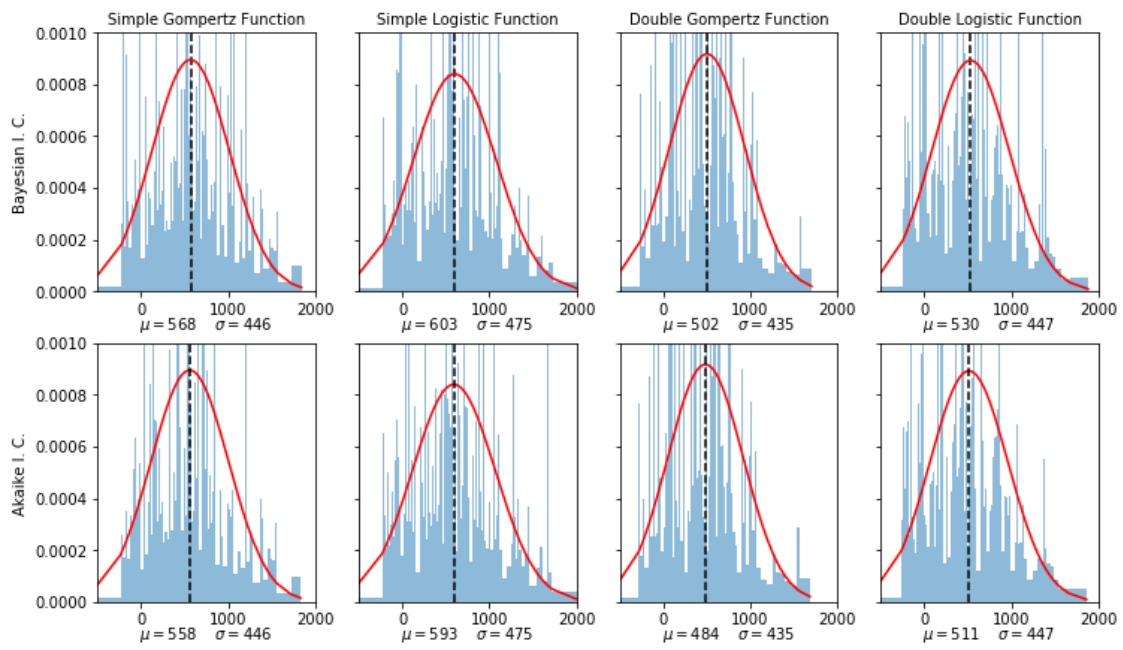


4.5 Adjusted R2



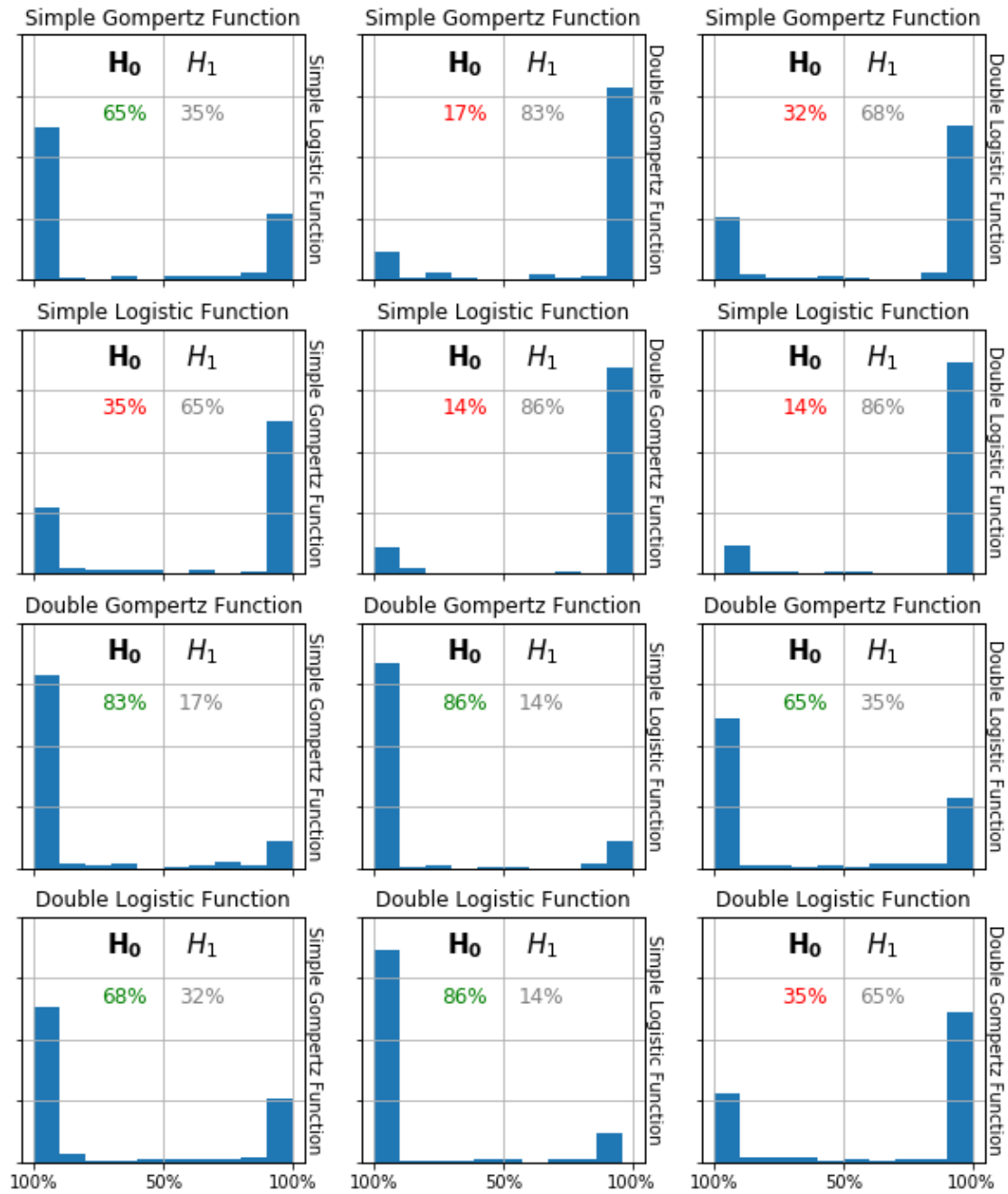
4.6 Information Criteria

Models Statistical Tests



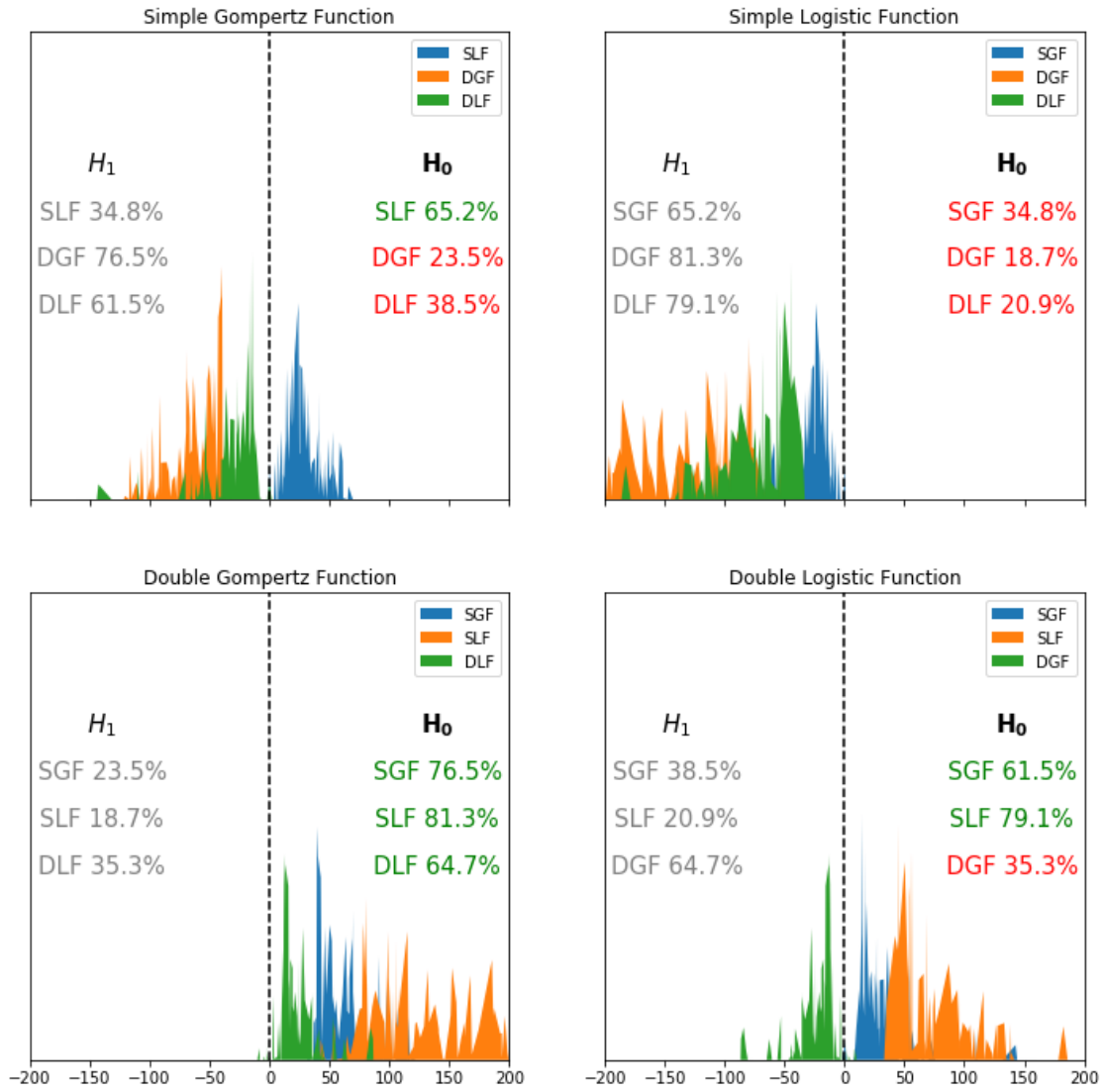
4.6.1 AIC

ΔAIC p density



4.6.2 BIC

Δ BIC test



4.7 Results

All tests strongly confirmed *Double Gompertz Function* as the better fitting model for SARS-nCoV-2 cumulative cases curve fitting. Results also showed that **DGF** is not only much more fitting than models with less parameters (**SLF** and **SGF**) as expected but also compared to *Double Logistic Function* with the same degrees of freedom.

	MAE	NAE	MSE	RMSE	NRMSE	R	$R^2 > .99$	$R^2_{adj} > .99$	$\Delta AIC < .5$	$\Delta BIC > 0$	H_1
									65.24%	65.24%	SLF
SGF	109	0.02578	508425	713	0.00632	0.99978	74.33%	73.80%	16.58%	23.53%	DGF
									31.55%	38.50%	DLF
									34.76%	34.76%	SGF
SLF	214	0.05076	2136060	1462	0.02656	0.99908	73.26%	72.19%	13.90%	18.72%	DGF
									13.90%	20.86%	DLF
									83.42%	76.47%	SGF
DGF	59	0.01408	151330	389	0.00188	0.99993	86.10%	86.10%	86.10%	81.28%	SLF
									64.71%	64.71%	DLF
									68.45%	61.50%	SGF
DLF	101	0.02389	504791	710	0.00628	0.99978	86.10%	85.03%	86.10%	79.14%	SLF
									35.29%	35.29%	DGF

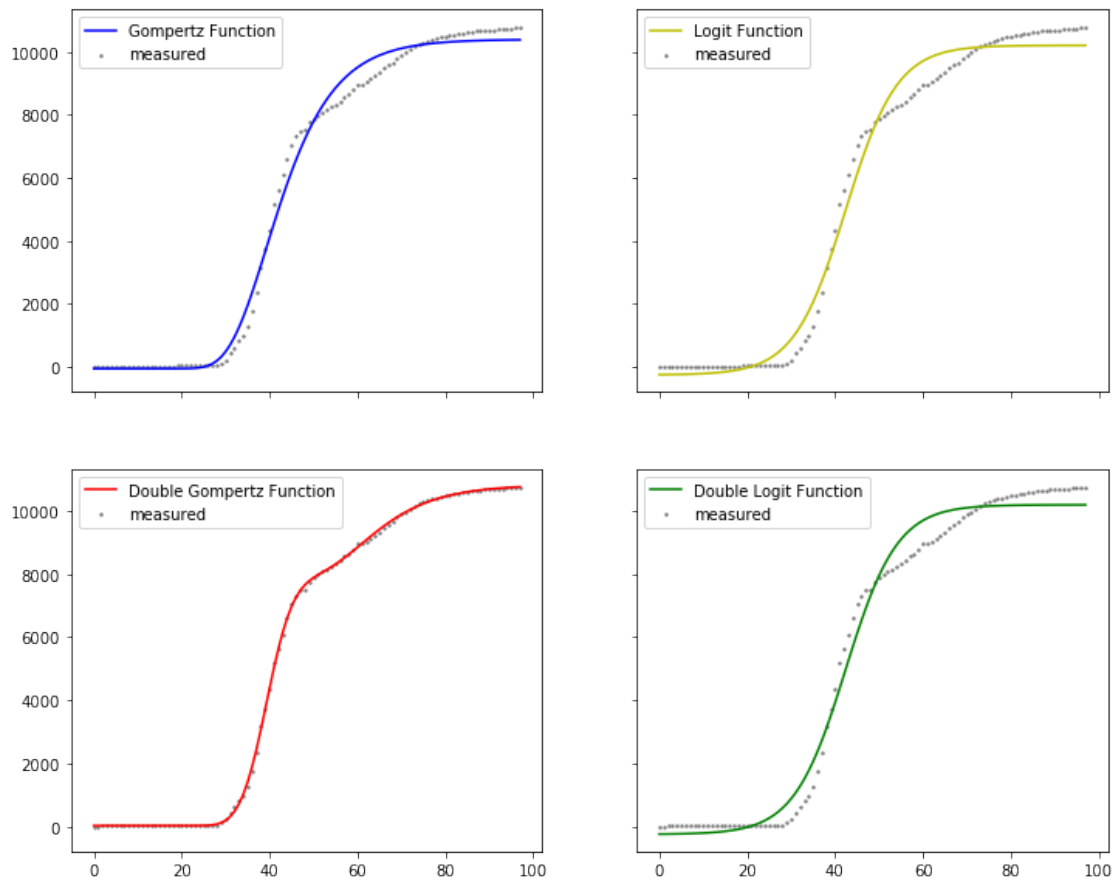
5 Conclusions

Among the compared models *Double Gompertz Function* has showed the best results and scores fitting data of SARS-nCoV-2 cumulative cases, suggesting that this model should be studied more deeply (possibly improved) and compared to other existing models for further analysis, including forecasting capabilities.

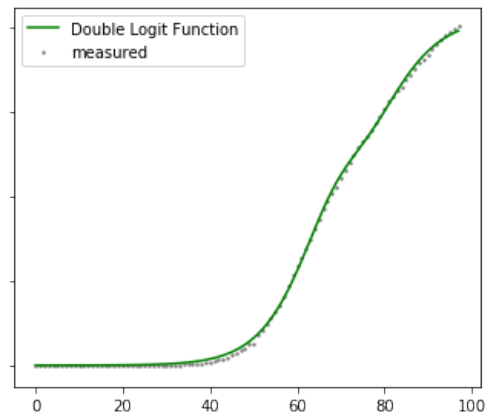
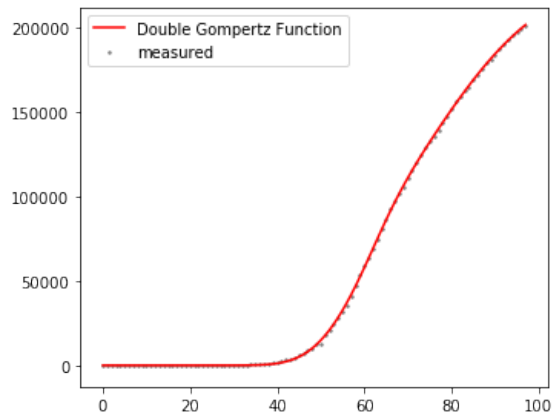
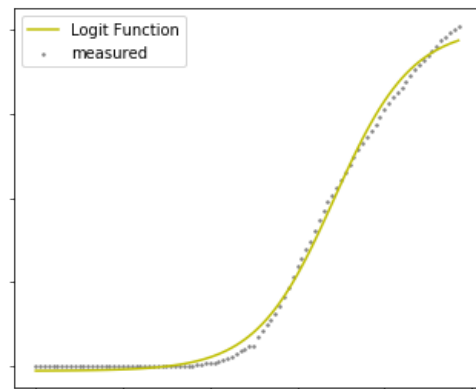
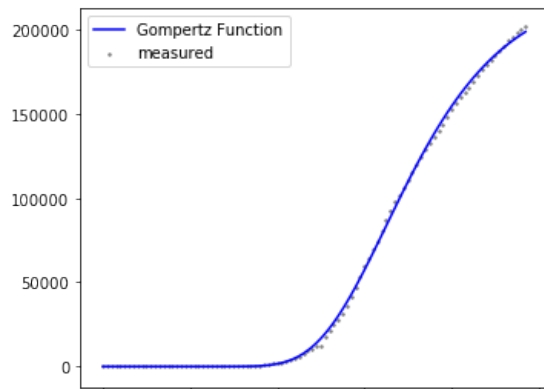
6 Plots

6.1 Fit examples

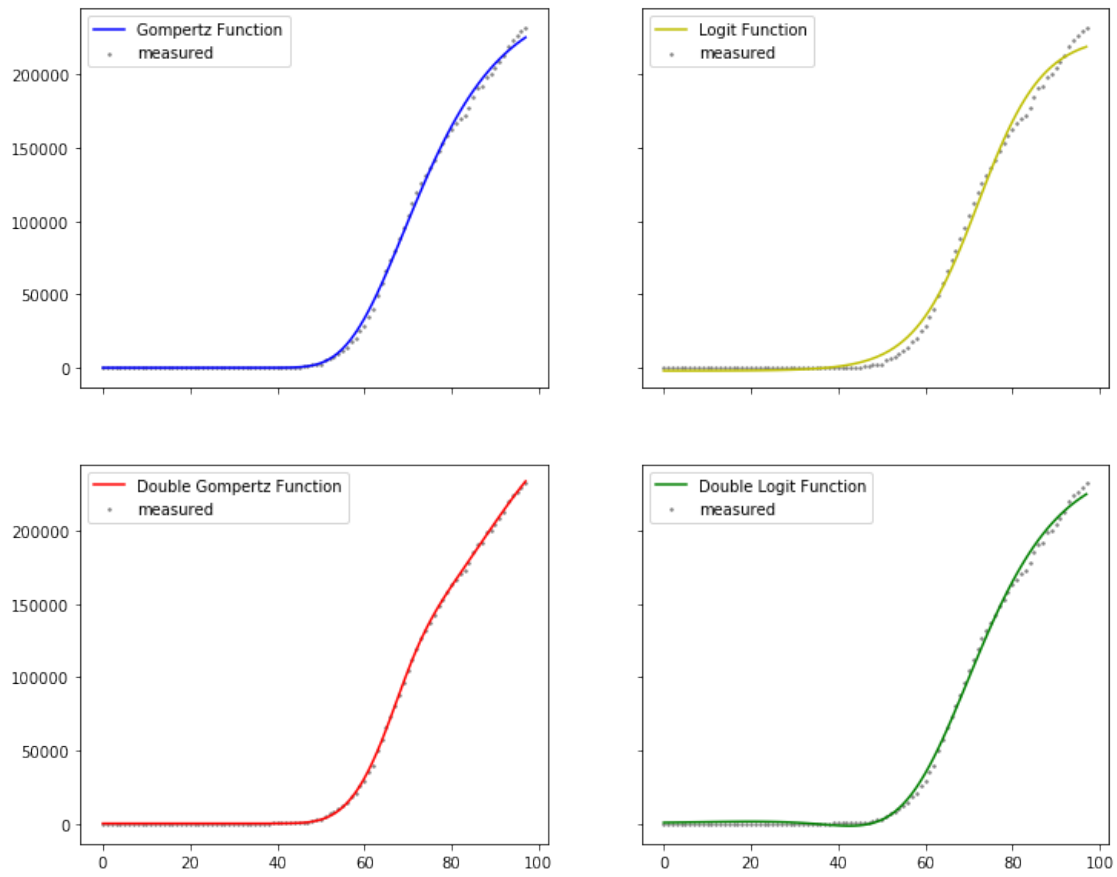
Korea, South



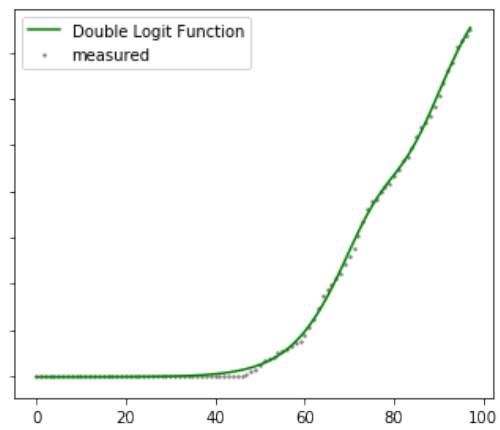
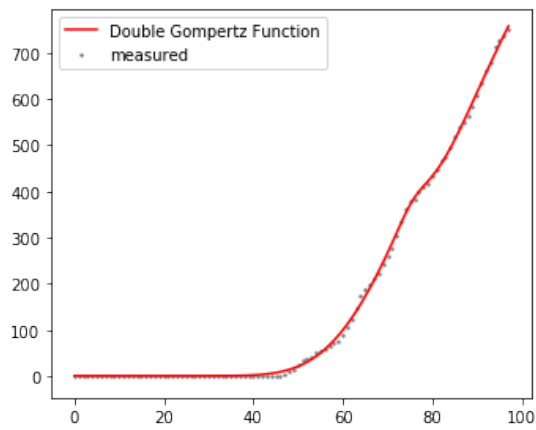
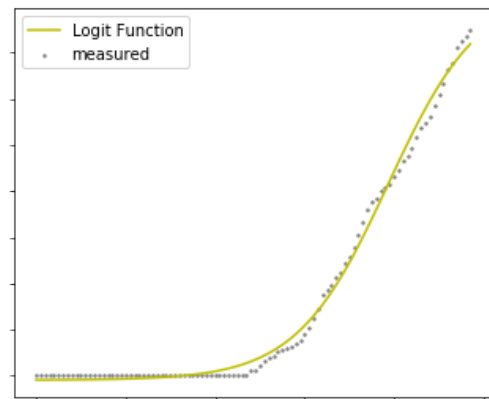
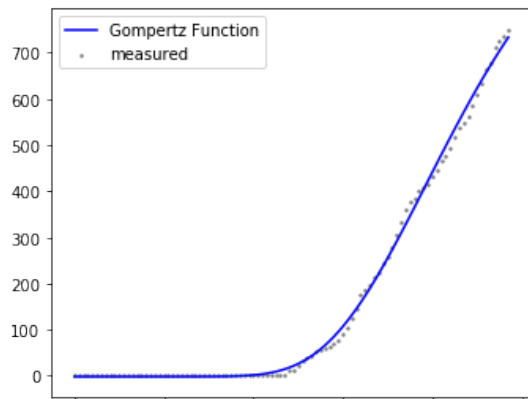
Italy



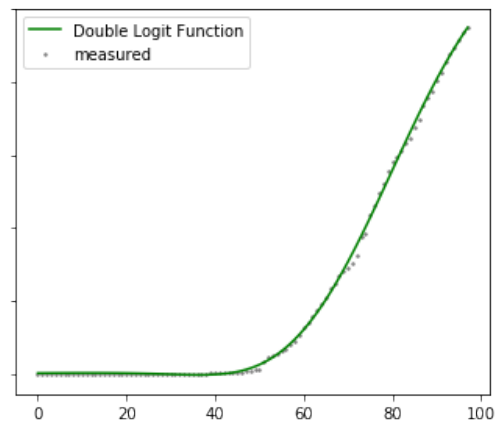
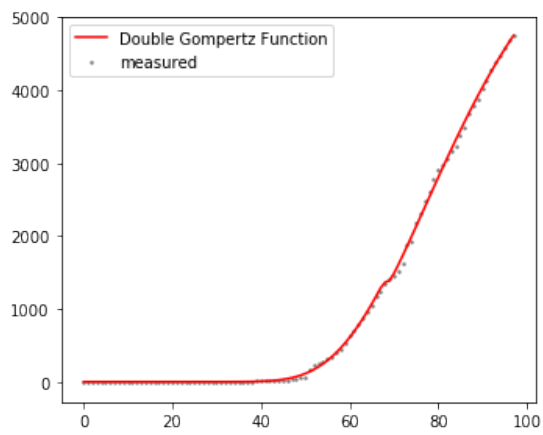
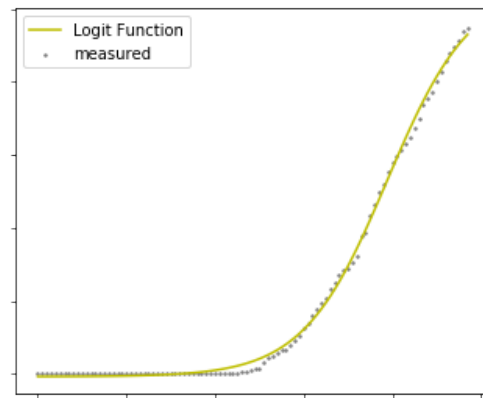
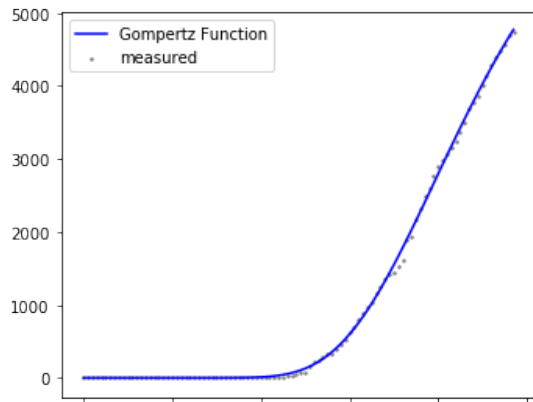
Spain



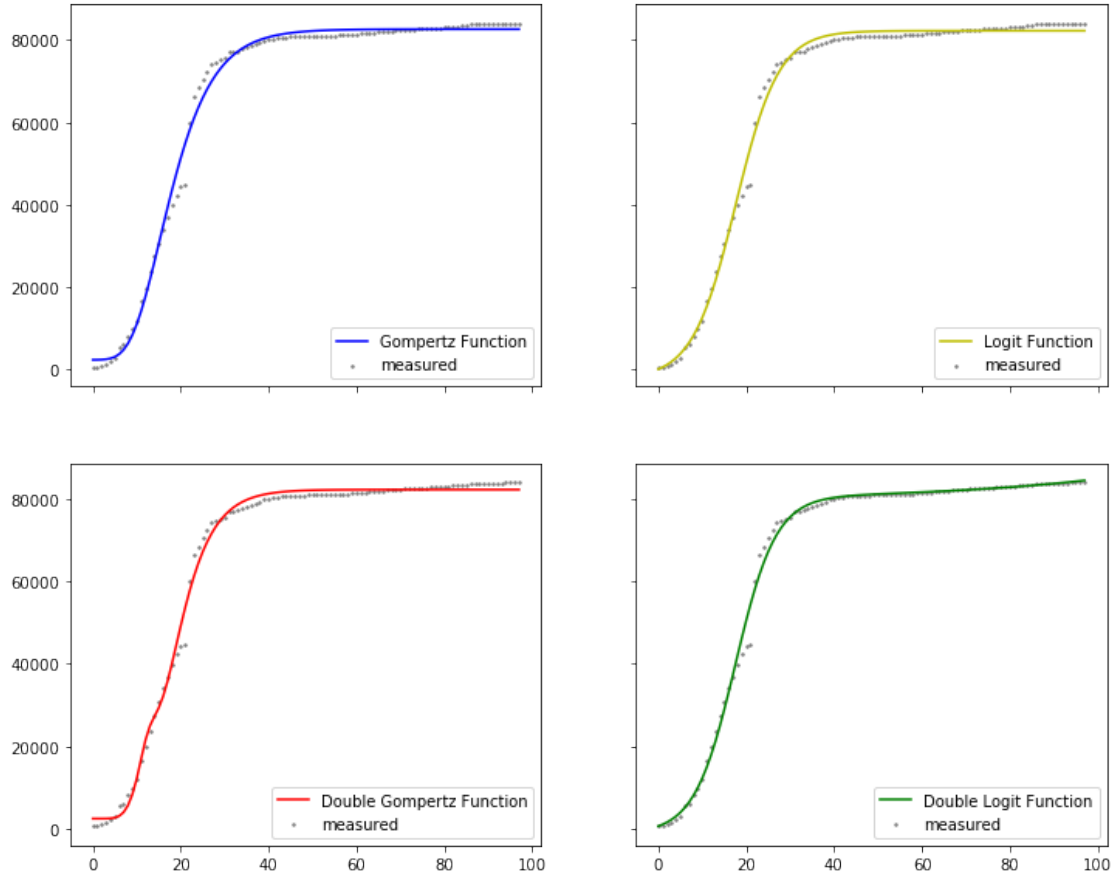
Albania



Finland



China



7 Appendix

7.1 Statistics formulae

In all formulae we assume: y as the observed (measured) values and \hat{y} as expected (predicted by fitting model) values; n is the number of values and n_{var} the number of model's variable parameters.

- **MBE:** Mean Bias Error

$$\mathbf{MBE} = \frac{\sum (y - \hat{y})}{n}$$

- **MAE:** Mean Absolute Error (aka Mean Deviation)

$$\mathbf{MAE} = \frac{\sum |y - \hat{y}|}{n}$$

- **NBE**: Normalized Bias Error

$$\mathbf{NBE} = \frac{\sum (y - \hat{y})}{\sum y}$$

- **NAE**: Normalized Absolute Error (aka Normalized Mean Deviation)

$$\mathbf{NAE} = \frac{\sum |y - \hat{y}|}{\sum y}$$

- **RSS**: Residual Sum of Squares

$$\mathbf{RSS} = \sum (y - \hat{y})^2$$

- **TSS**: Total Sum of Squares

$$\mathbf{TSS} = \sum \left(y - \frac{\sum y}{n} \right)^2$$

- **MSE**: Mean Squared Error (aka Variance)

$$\mathbf{MSE} = \sigma^2 = \frac{\mathbf{RSS}}{n}$$

- **RMSE**: Root Mean Squared Error (aka Standard Deviation)

$$\mathbf{RMSE} = \sigma = \sqrt{\mathbf{MSE}}$$

- **NRMSE**: Normalized Root Mean Squared Error (aka Normalized Standard Deviation)

$$\mathbf{NRMSE} = \sigma_\nu = \frac{\mathbf{RMSE}}{\sum y}$$

- R^2 : Coefficient of Determination:

$$R^2 = 1 - \frac{\mathbf{RSS}}{\mathbf{TSS}}$$

- R_{adj}^2 : Adjusted Coefficient of Determination:

$$R_{adj}^2 = 1 - \frac{\mathbf{RSS}}{\mathbf{TSS}} \left(\frac{n-1}{n-n_{var}-1} \right)$$

- **Pearson R**: Pearson Correlation Coefficient

$$\mathbf{R} = \frac{\sum y\hat{y} - \frac{1}{n} \sum y \sum \hat{y}}{\sqrt{\sum y^2 - \frac{1}{n} (\sum y)^2} \sqrt{\sum \hat{y}^2 - \frac{1}{n} (\sum \hat{y})^2}}$$

- **AIC**: Aikake Information Criterion

$$\mathbf{AIC} = n \ln \left(\frac{\mathbf{RSS}}{n} \right) + 2n_{var}$$

- **AIC p**: Aikake Information Criterion score (or weight) in **AIC** relative probability distribution space [FIG]:

$$\mathbf{AIC}_p = \frac{e^{-0.5 \cdot (\mathbf{AIC}_1 - \mathbf{AIC}_0)}}{1 + e^{-0.5 \cdot (\mathbf{AIC}_1 - \mathbf{AIC}_0)}}$$

$$\mathbf{AIC}_p < .5 \Rightarrow P(H_0) > P(H_1)$$

- **BIC**: Bayesian Information Criterion

$$\mathbf{BIC} = n \ln \left(\frac{\mathbf{RSS}}{n} \right) + \ln(n)n_{var}$$

- $\Delta\mathbf{BIC}$: Bayesian Information Criterion difference [FIG]:

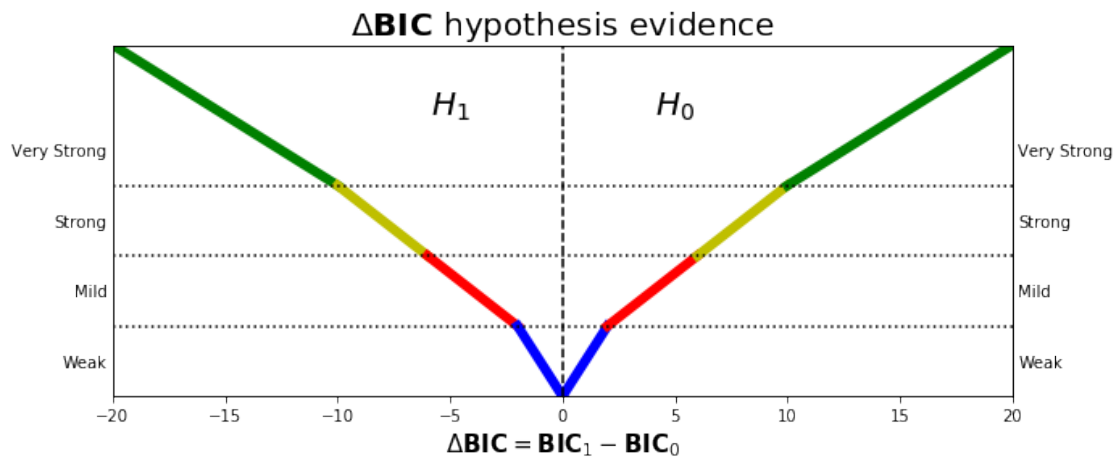
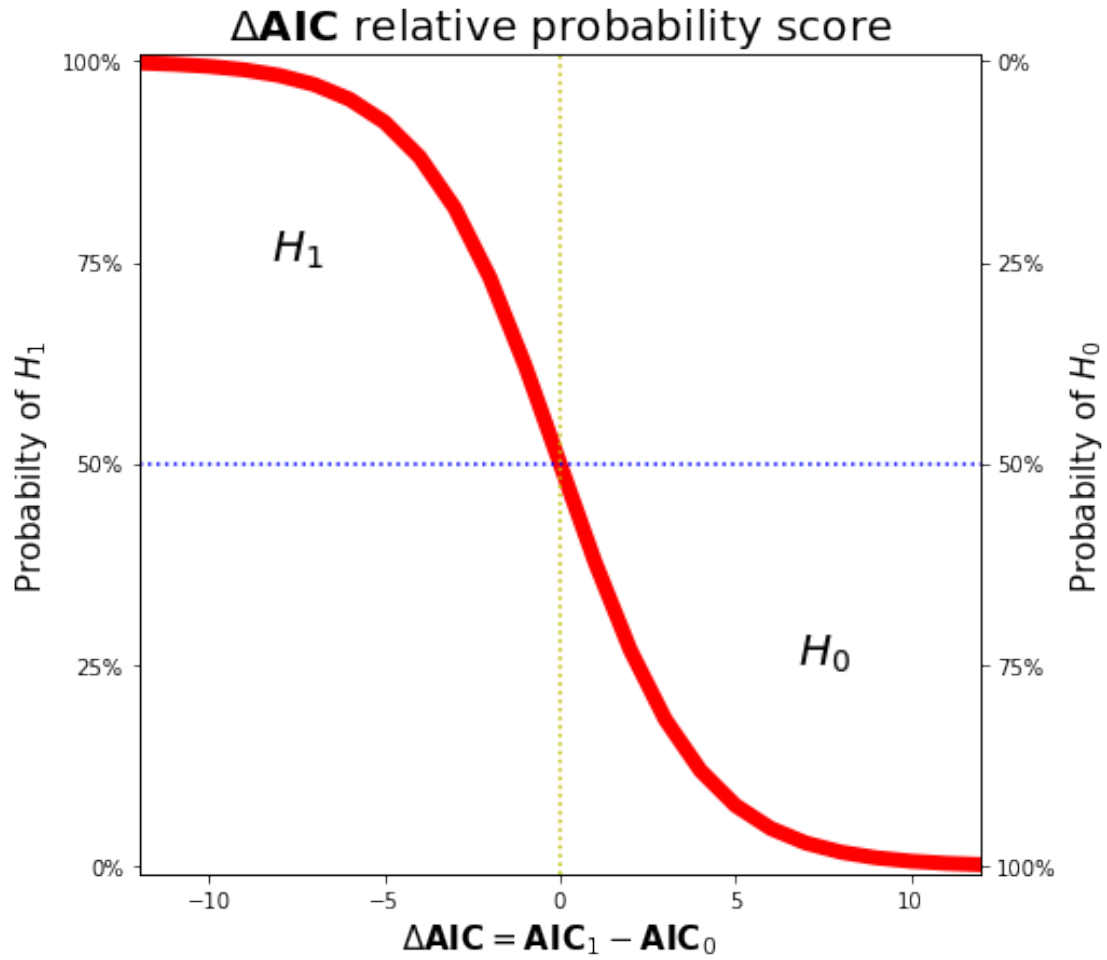
$$\Delta\mathbf{BIC} = \mathbf{BIC}_1 - \mathbf{BIC}_0$$

$$\Delta\mathbf{BIC} > 10 \Rightarrow H_0 \text{ Very strong evidence}$$

$$\Delta\mathbf{BIC} = (6, 10] \Rightarrow H_0 \text{ Strong evidence}$$

$$\Delta\mathbf{BIC} = (2, 6] \Rightarrow H_0 \text{ Mild evidence}$$

$$\Delta\mathbf{BIC} = (0, 2] \Rightarrow H_0 \text{ Weak evidence}$$



References

- [1] J. H. University, "Johns hopkins university covid-19 github repository," *GitHub*, 2020. [Online]. Available: <https://github.com/CSSEGISandData/COVID-19>

- [2] M. Pierini, “Max pierini covid-19 github repository,” *GitHub*, 2020. [Online]. Available: <https://github.com/maxdevblock/covid-19-time-series>
- [3] C. J. Willmott and K. Matsuura, “On the use of dimensioned measures of error to evaluate the performance of spatial interpolators,” *International Journal of Geographical Information Science*, vol. 20, no. 1, pp. 89–102, 2006.
- [4] N. Hofstra, M. Haylock, M. New, P. Jones, and C. Frei, “Comparison of six methods for the interpolation of daily, european climate data,” *Journal of Geophysical Research: Atmospheres*, vol. 113, no. D21, 2008.
- [5] A. H. Murphy and E. S. Epstein, “Skill scores and correlation coefficients in model verification,” *Monthly weather review*, vol. 117, no. 3, pp. 572–582, 1989.
- [6] A. Zhab Bhat, “Curve fitting: Linear, cubic, polynomial (1-5), pie- cewise, goodness of fit and regression analysis in python.” [Online]. Available: <https://projects.skill-lync.com/projects/Curve-Fitting-In-Python-Assignment-89826>
- [7] H. Motulsky and A. Christopoulos, “Comparing models and curves, graphpad software,” 2002.
- [8] F. J. Fabozzi, S. M. Focardi, S. T. Rachev, and B. G. Arshanapalli, *The basics of financial econometrics: Tools, concepts, and asset management applications*. John Wiley & Sons, 2014.