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

Max Pierini*

May 3, 2020

[1]:
```
import pandas as pd
import datetime as dt
from IPython.display import display, Markdown
```

[2]:
```
def simple_gompertz_function(x, a, b, k, e):
    exp = - np.exp(k * (b - x))
    return a * np.exp(exp) + e

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

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

def double_logistic_function(x, a1, b1, k1, a2, b2, k2, e):
    d1 = k1 * (b1 - np.array(x))
    l1 = a1 / (1 + np.exp(d1))
    d2 = k2 * (b2 - np.array(x))
    l2 = (a2 - a1) / (1 + np.exp(d2))
    return l1 + l2 + e
```

[3]:
```
today = dt.datetime.now().strftime("%Y-%m-%d")
df = pd.read_pickle(f"data-confirmed-{today}.pkl")

countries = df["measured"].size
start = pd.Timestamp("2020-01-22")
period = df["measured"]["China"].size
end = start + pd.Timedelta(days=period)
```

*info@maxpierini.it

```
errors = []
values = 0
for f in df.columns:
    if f == "measured":
        continue
    for c in df[f].index:
        if not df[f][c]:
            errors.append(c)
        values += df[f][c].best_fit.size
```

1 Abstract

[4]:
```
display(
    Markdown(f"""
Four different models have been compared for fitting SARS-nCoV-2 total␣
 ↪cumulative cases curves in
{countries} countries over a period of {period} 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.
$R^2$ ($p>.99$), $R^2_{{adj}}$ ($p>.99$), $\Delta$AIC ($>0$) and $\Delta$BIC␣
 ↪($p<.5$) 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.
    """
    )
)
```

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

[5]:
```
display(
    Markdown(f"""
SARS-nCoV-2 total cumulative cases data have been gathered from Johns Hopkins␣
 ↪University GitHub
repository [REF] and summed into single countries where regional level was␣
 ↪provided [REF].
Data have been used "as is" without rejecting any outlier and/or error␣
 ↪(negative daily $\Delta$).
Data and results have been stored in a `pandas` n-dimensional `DataFrame`.

Raw data contained {countries} countries and daily cumulative confirmed cases␣
 ↪for
{period} days, from 2020-01-22 to {end.strftime('%Y-%m-%d')}.
    """)
)
```

SARS-nCoV-2 total cumulative cases data have been gathered from Johns Hopkins University GitHub repository [REF] and summed into single countries where regional level was provided [REF]. 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 [REF].

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 (Δ**AIC**) in relative probability density space, *Bayesian Information Criterion* differences (Δ**BIC**), 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 Appendix [REF] for formulas.

Models have been defined as follow:

- Simple Logisic 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 Logisic 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]
)
```

$$a = y_{-1}$$

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

$$k = 0.1$$

$$\varepsilon = y_0$$

- **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]
)
```

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

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

$$k_1 = 0.1$$

$$a_2 = \mathbf{max}(y)$$

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

$$k_2 = 0.1$$

$$\varepsilon = y_0$$

[6]:
```
display(
    Markdown(f"""
Fitting failed for {len(errors)} countries returning best fit information from␣
 ↪{countries} countries,
for a total of {countries*period} observed and {values} 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
avalaible at [REF].
    """)
)
```

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 avalaible at [REF].

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

4 Analysis

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

Mean absolute error (**MAE**) is a natural, unambiguous, measure of average error [Willmott and Matsuura, 2006]. 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.

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 [REF]. 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 [Willmott and Matsuura, 2006] 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. 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 [Murphy and Epstein, 1989; Wilks, 2006].

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 [REF].

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

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 [REF].

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 formulas [REF].

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):
```

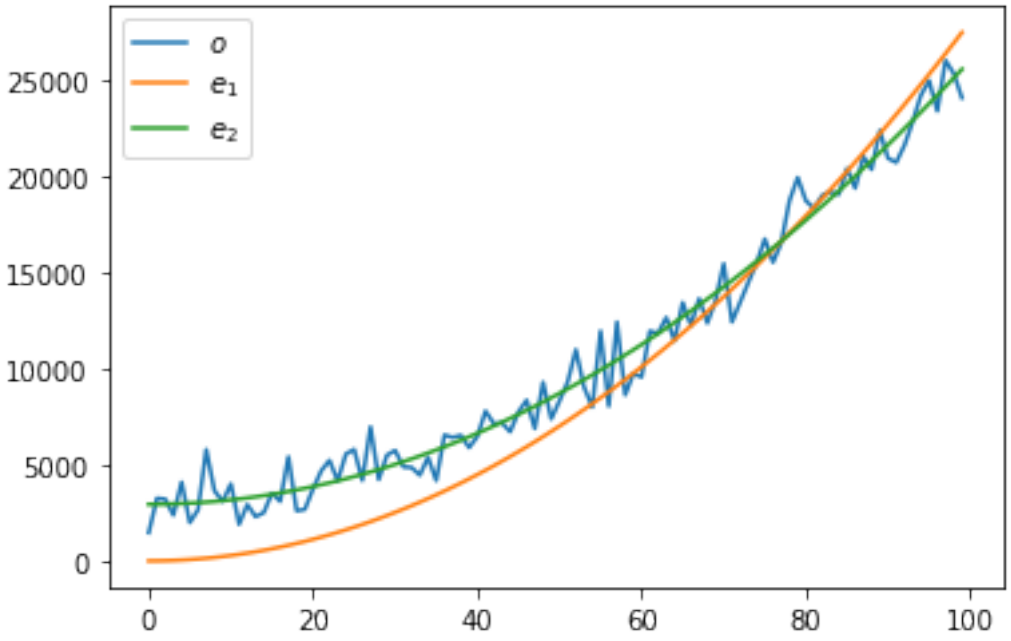
```
        mae.append(skl.mean_absolute_error(o, e))
        nae.append(np.sum(np.abs(o - e)) / np.sum(o))
        mbe.append(np.sum(o - e) / len(x))
        nbe.append(np.sum(o - e) / np.sum(o))
        _mse = skl.mean_squared_error(o, e)
        mse.append(_mse)
        _rmse = np.sqrt(_mse)
        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]).

[7]:
```
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)[1])
        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)
```

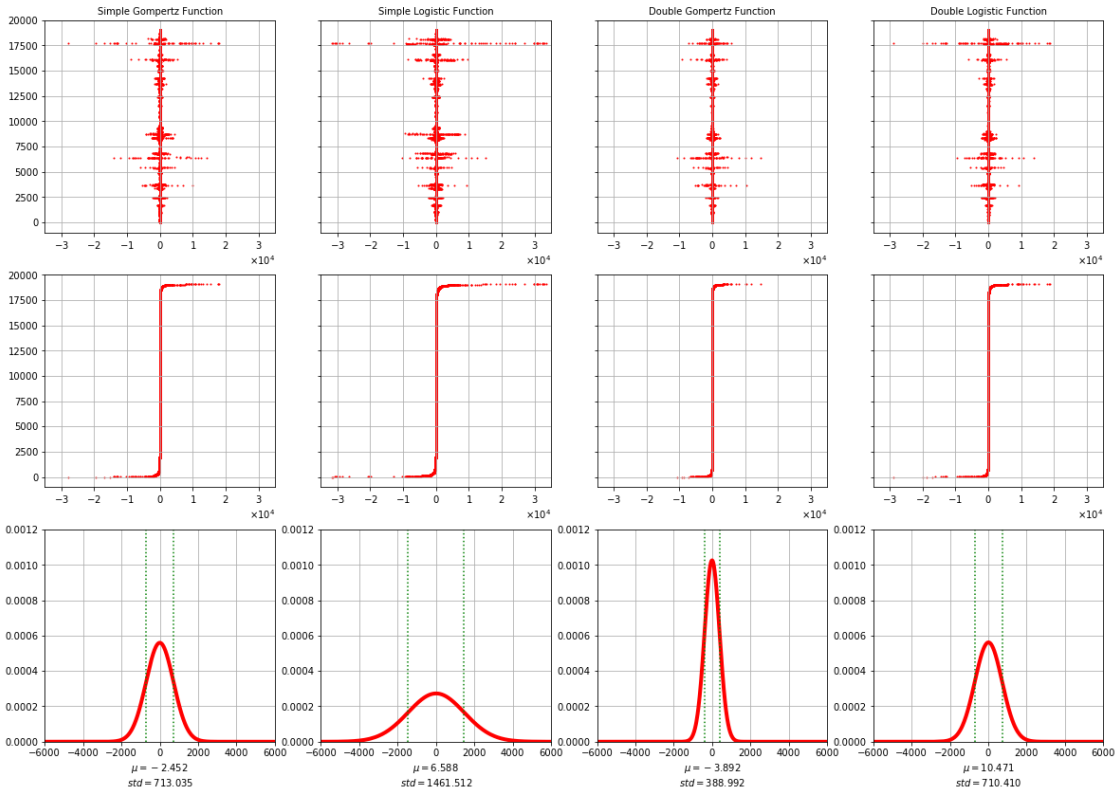
```
        bic.append(len(x) * np.log(rss / len(x)) + np.log(len(x)) * P[i])
    print(f"  MAE: {mae[0]:>14.6f} {mae[1]:>14.6f}")
    print(f"  NAE: {nae[0]:>14.6f} {nae[1]:>14.6f}")
    print(f"  BME: {mbe[0]:>14.9f} {mbe[1]:>14.6e}")
    print(f"  NBE: {nbe[0]:>14.9f} {nbe[1]:>14.6e}")
    print(f"  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]:>14.7e} {r[1]:>14.7e}   \U00000394:{r[0]-r[1]:.6e}")
    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]):
↪.6e}")
    print(f"  BIC: {bic[0]:>14.6f} {bic[1]:>14.6f}   \U00000394:{bic[0] -␣
↪bic[1]:.7f}")

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="$e_1$")
plt.plot(x, e2, label="$e_2$")
plt.legend(loc="best")
plt.show()
```

```
                e1 H1           e2 H0
------  -------------  -------------
  MAE:    1862.734150     856.051098
  NAE:       0.176796       0.081250
  BME: 1317.558145262  -1.581981e-08
  NBE:    0.125052338  -1.501493e-12
  MSE: 5002073.344917  1120263.29439
 RMSE:    2236.531543    1058.424912
NRMSE:    0.002122741    0.001004574
    R:  1.1738144e-81  1.1738144e-81   Δ:0.000000e+00
   R2:    0.895183791    0.976525384
R2adj:    0.894114238    0.976041371
  AIC:    1544.536305    1394.907430   p:3.224798e-33
  BIC:    1547.141476    1402.117770   Δ:145.0237052
```



4.2 Residual

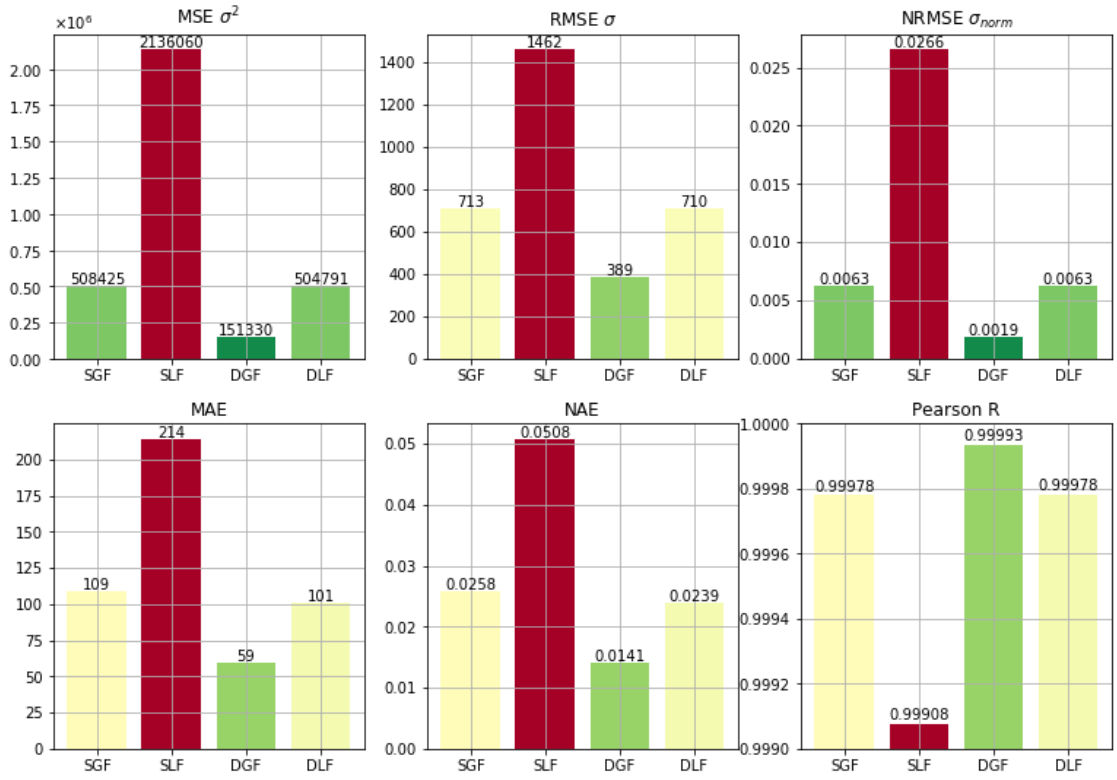
Total residual from each model have been collected and compared to get a first "rough" evidence of the most likely better fitting model [FIG].



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

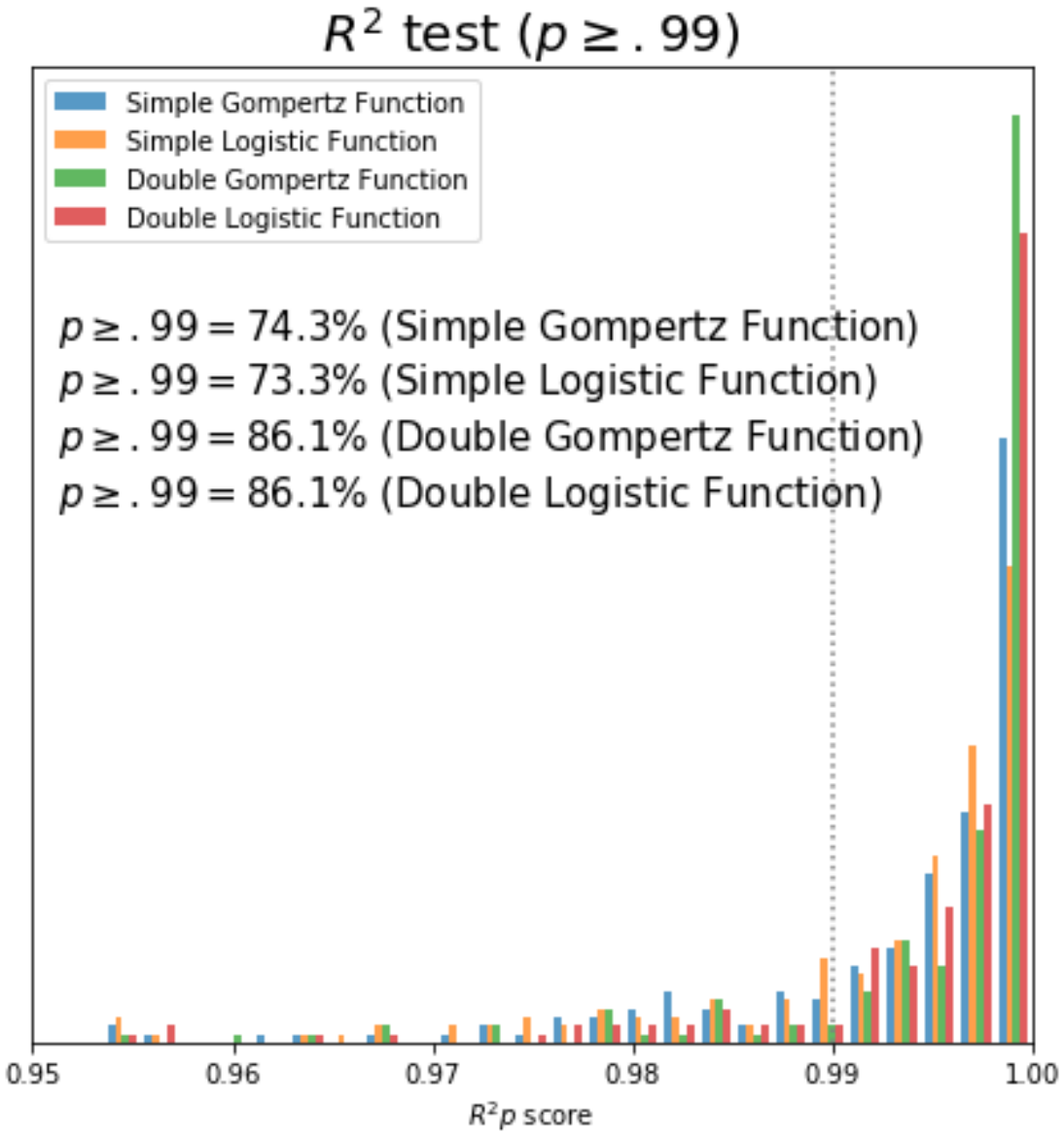
Mean Squared Error (**MSE**, Variance), Root Mean Squared Error (**RMSE**, Standard Deviation), Normalize Root Mean Squared Error (**NRMSE**, Normalized Standard Deviation), Mean Absolute Error (**MAE**), Normalized Absolute Error (**NAE**) and Pearson Correlation Coefficient (**Pearson R**) have been computed for all models residual [FIG] (see Appendix for formulas [REF]). **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

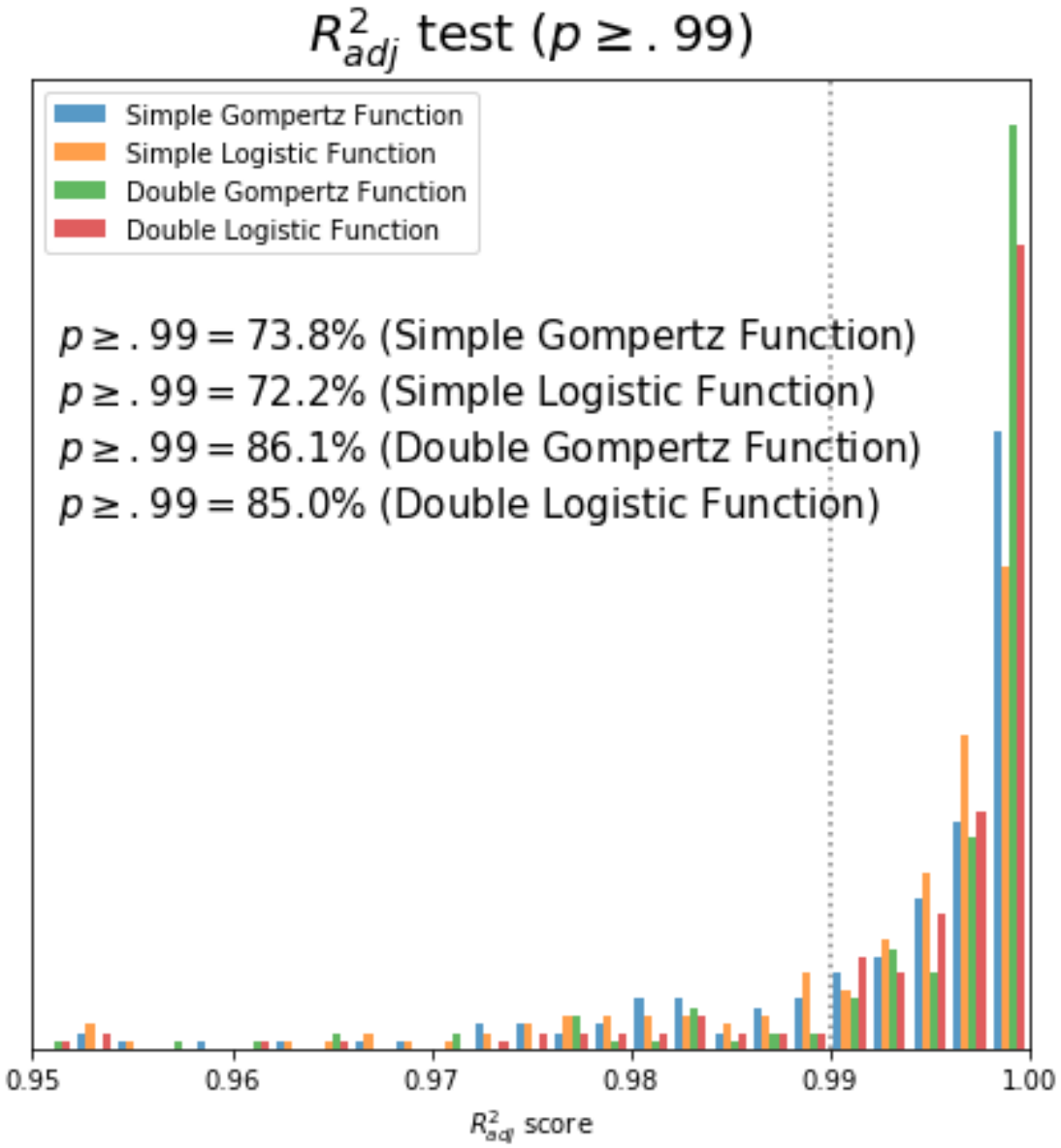


Coefficient of Determination R^2, Adjusted Coefficient of Determination R^2_{adj}, Akaike Information Criterion (**AIC**) and Bayesian Information Criterion (**BIC**) have computed and collected from all fits and compared with each other.

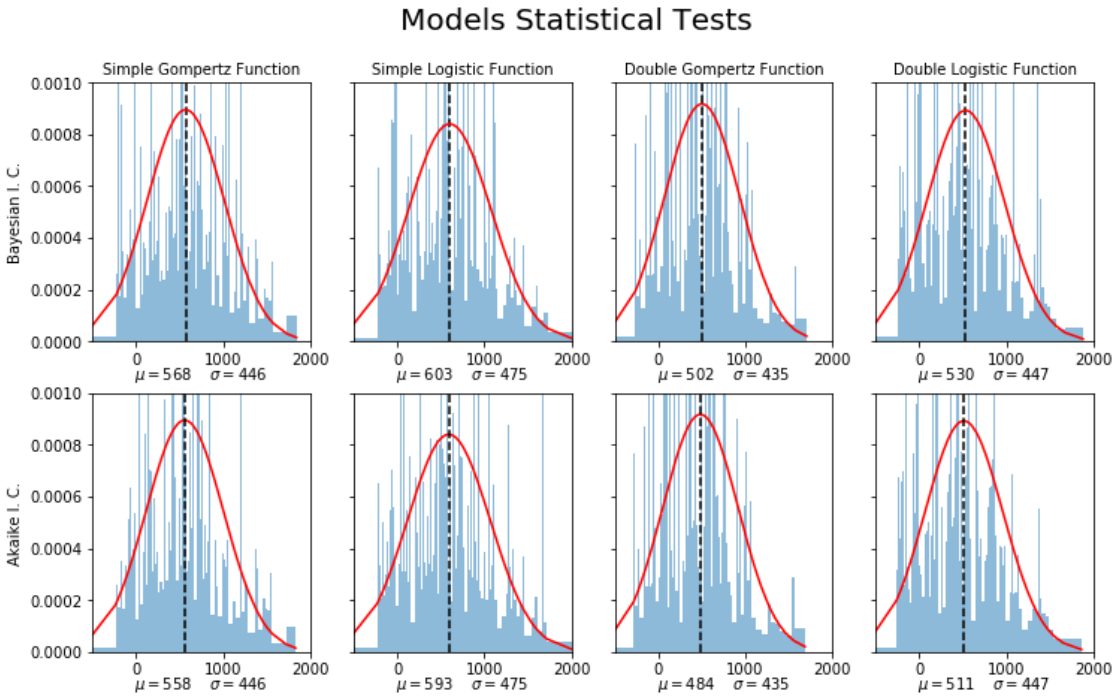
4.4 R2


R^2 test ($p \geq .99$)
Simple Gompertz Function
Simple Logistic Function
Double Gompertz Function
Double Logistic Function
$p \geq .99 = 74.3\%$ (Simple Gompertz Function)
$p \geq .99 = 73.3\%$ (Simple Logistic Function)
$p \geq .99 = 86.1\%$ (Double Gompertz Function)
$p \geq .99 = 86.1\%$ (Double Logistic Function)
0.95
0.96
0.97
0.98
0.99
1.00
R^2p score

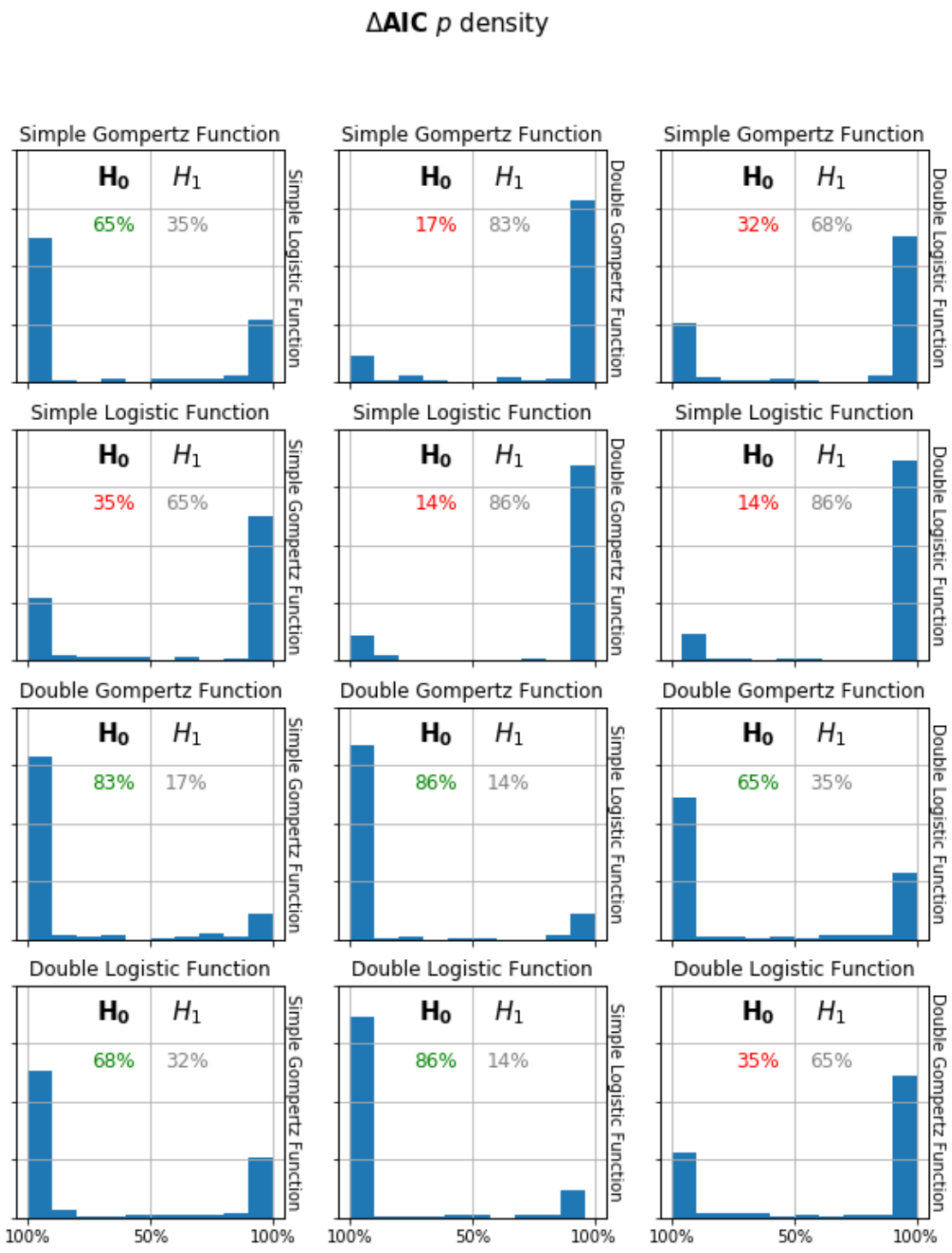
4.5 Adjusted R2



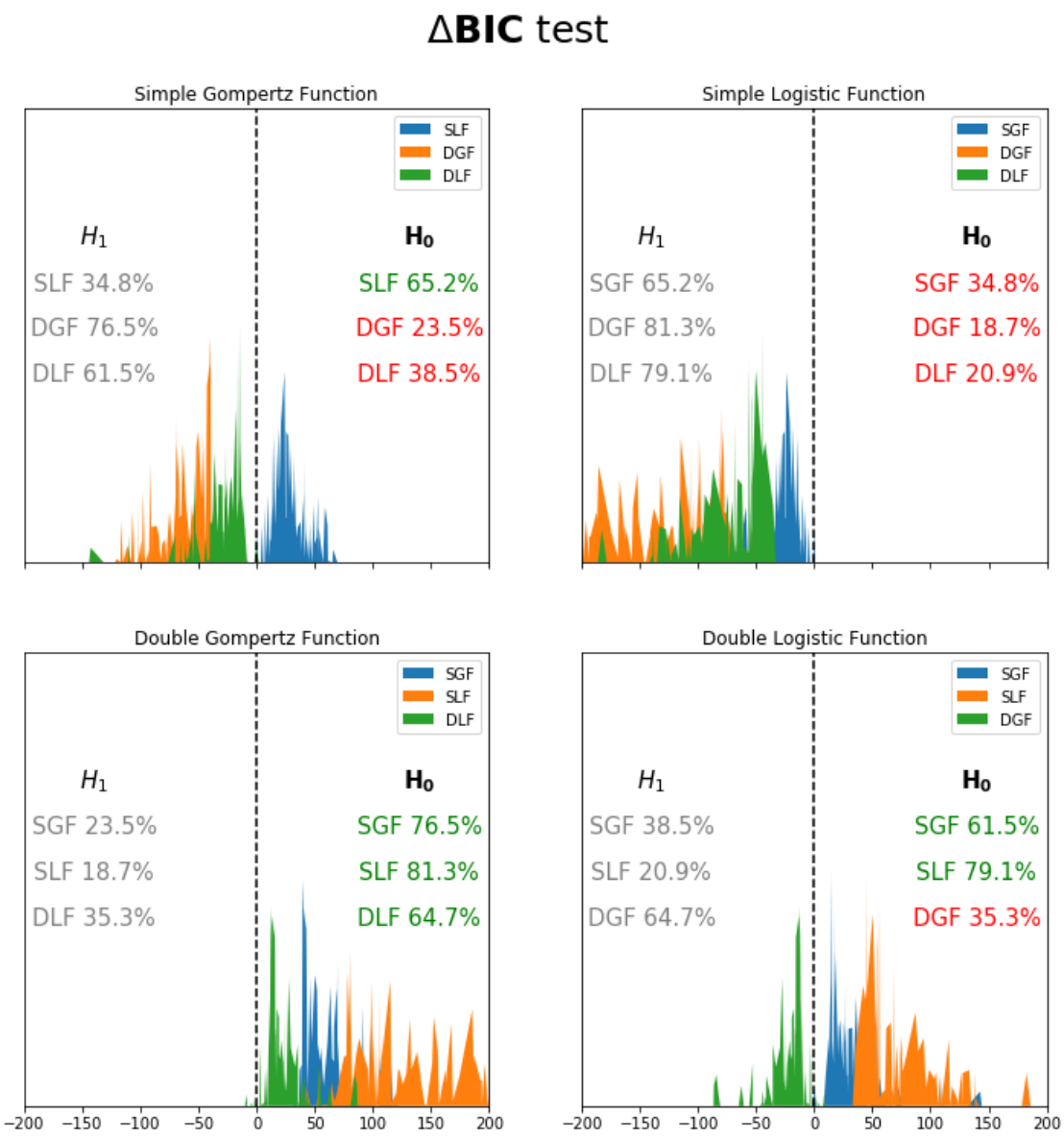
4.6 Information Criteria



4.6.1 AIC


ΔAIC p density
Simple Gompertz Function
H0 H1
65% 35%
Simple Logistic Function
Simple Gompertz Function
H0 H1
17% 83%
Double Gompertz Function
Simple Gompertz Function
H0 H1
32% 68%
Double Logistic Function
Simple Logistic Function
H0 H1
35% 65%
Simple Gompertz Function
Simple Logistic Function
H0 H1
14% 86%
Double Gompertz Function
Simple Logistic Function
H0 H1
14% 86%
Double Logistic Function
Double Gompertz Function
H0 H1
83% 17%
Simple Gompertz Function
Double Gompertz Function
H0 H1
86% 14%
Simple Logistic Function
Double Gompertz Function
H0 H1
65% 35%
Double Logistic Function
Double Logistic Function
H0 H1
68% 32%
Simple Gompertz Function
Double Logistic Function
H0 H1
86% 14%
Simple Logistic Function
Double Logistic Function
H0 H1
35% 65%
Double Gompertz Function
100% 50% 100%
100% 50% 100%
100% 50% 100%

4.6.2 BIC



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	$R^2 > .99$	$R^2_{adj} > .99$	Δ**AIC** $< .5$	Δ**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

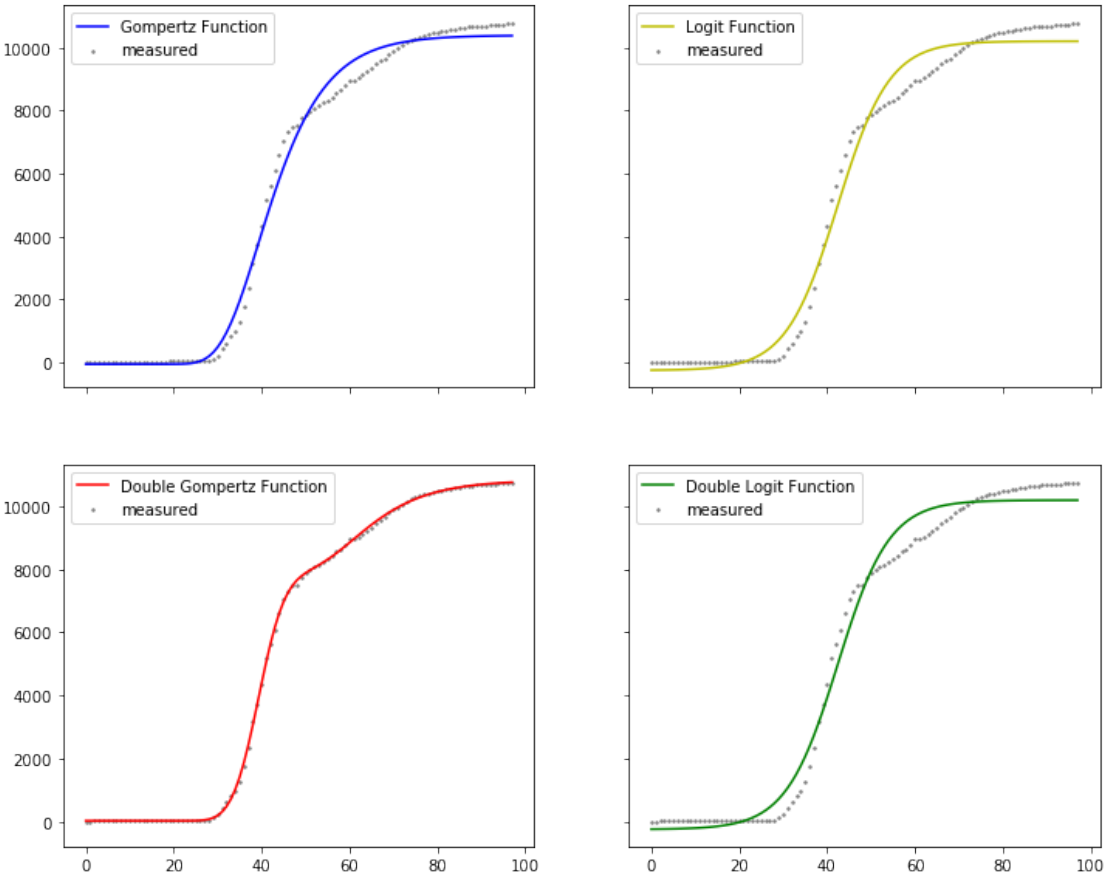
Among the compared models *Double Gompertz Function* has showed the best results and scores fitting data of SARS-nCoV-2 cumulative cases, suggesting that this model should be studied more deeply (possibly improved) and compared to other existing models for further analysis, including forecasting capabilities.

6 Plots

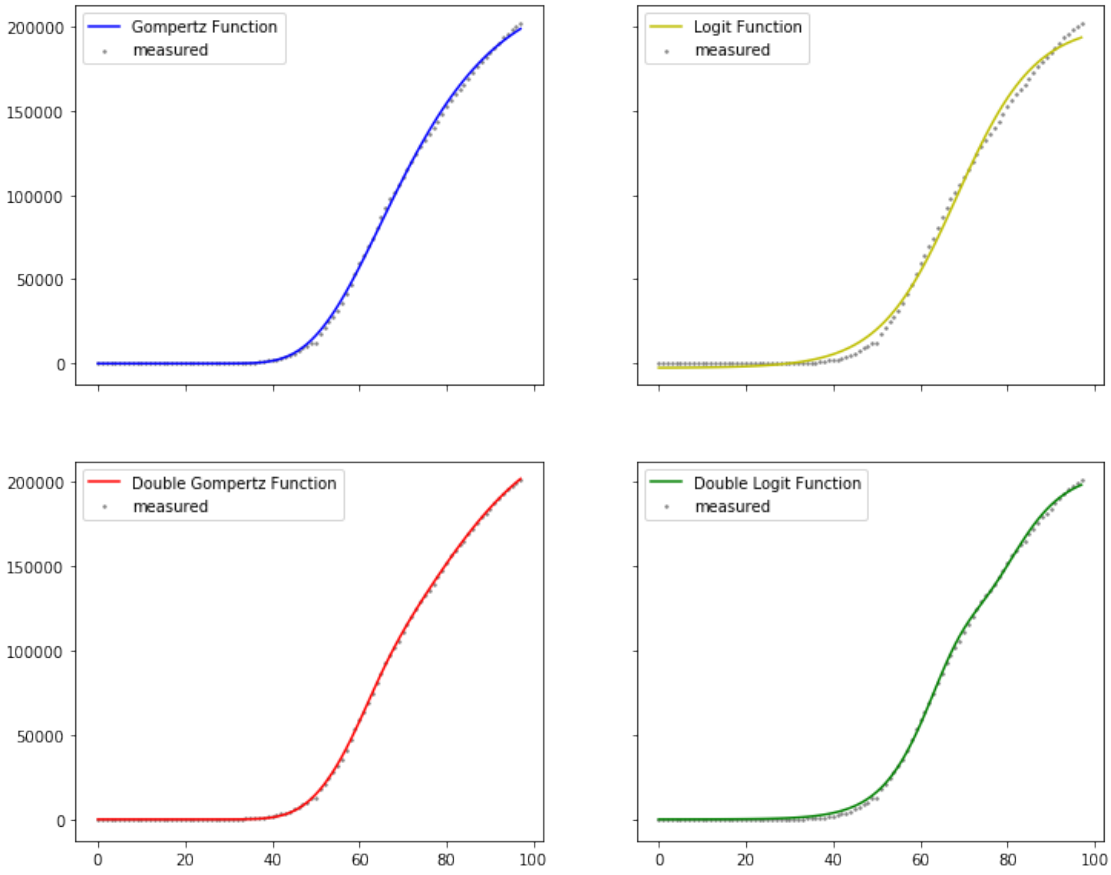
6.1 Fit examples

[8]:
```
from IPython.display import display, Markdown
for j in range(6):
    display(Markdown(f"![img](fit{j+1}.png)\n\n***\n\n"))
```

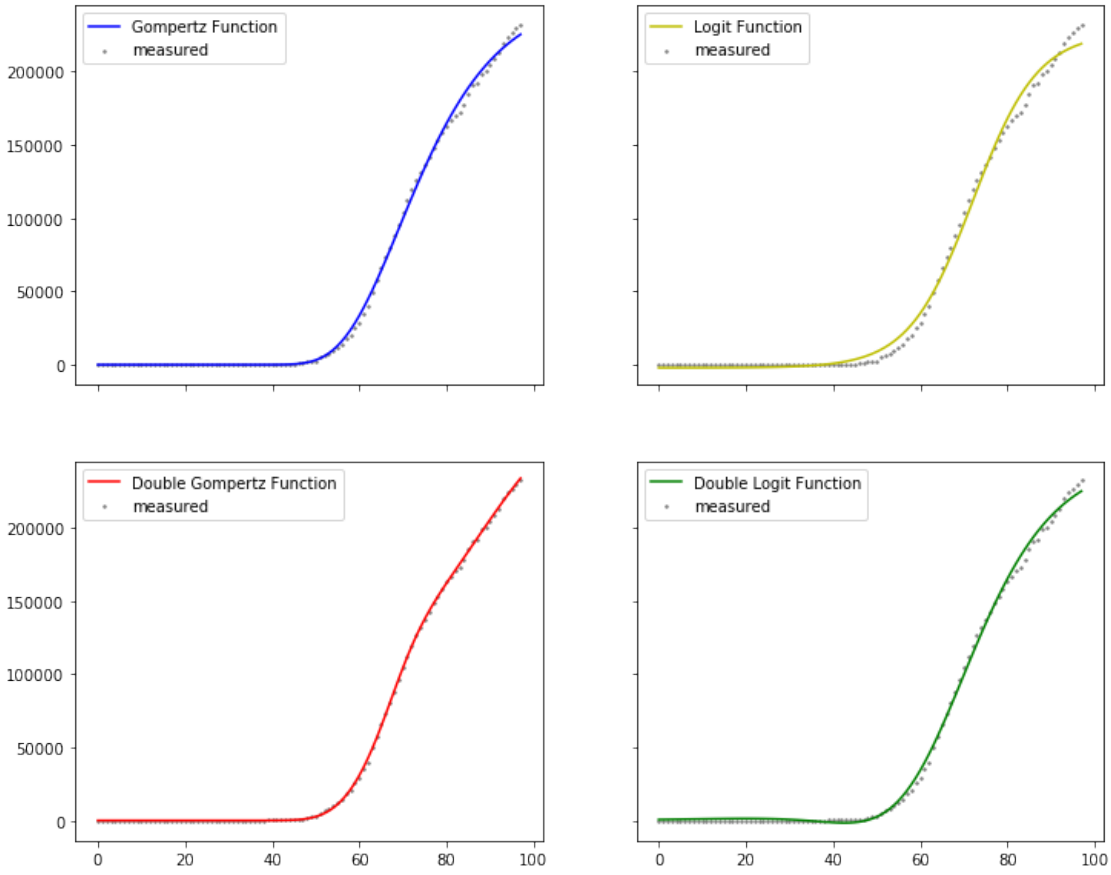
Korea, South


Gompertz Function
measured
Logit Function
measured
Double Gompertz Function
measured
Double Logit Function
measured
0
2000
4000
6000
8000
10000
0
20
40
60
80
100

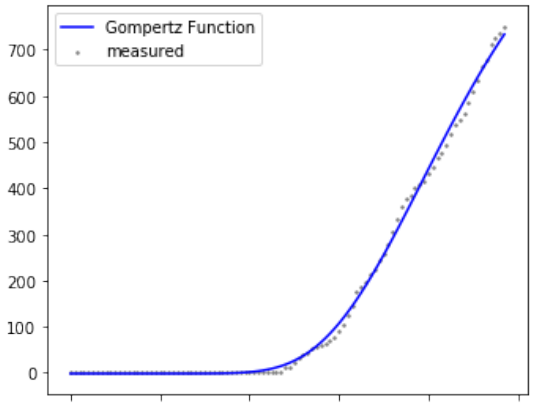
Italy

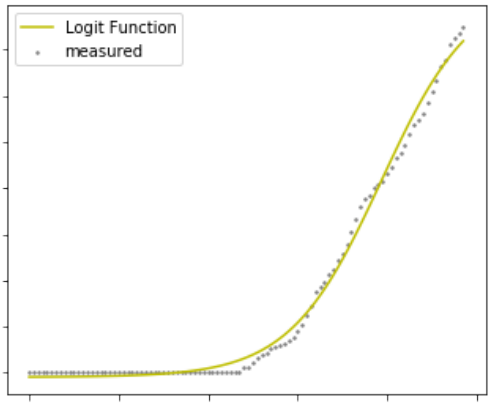

Gompertz Function
measured
Logit Function
measured
Double Gompertz Function
measured
Double Logit Function
measured
200000
150000
100000
50000
0
0
20
40
60
80
100

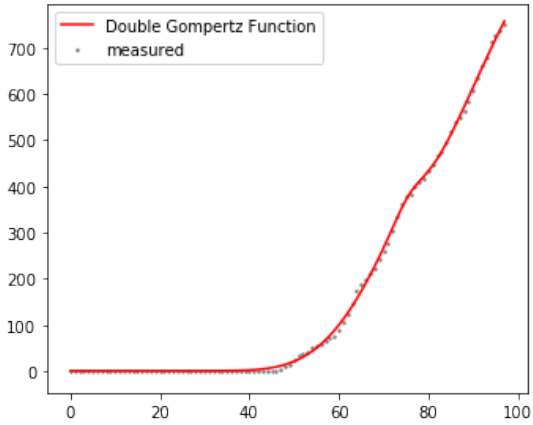
Spain

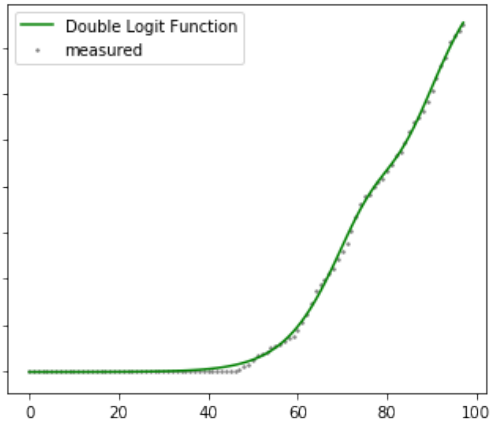

Gompertz Function
measured
Logit Function
measured
Double Gompertz Function
measured
Double Logit Function
measured
200000
150000
100000
50000
0
0
20
40
60
80
100

Albania

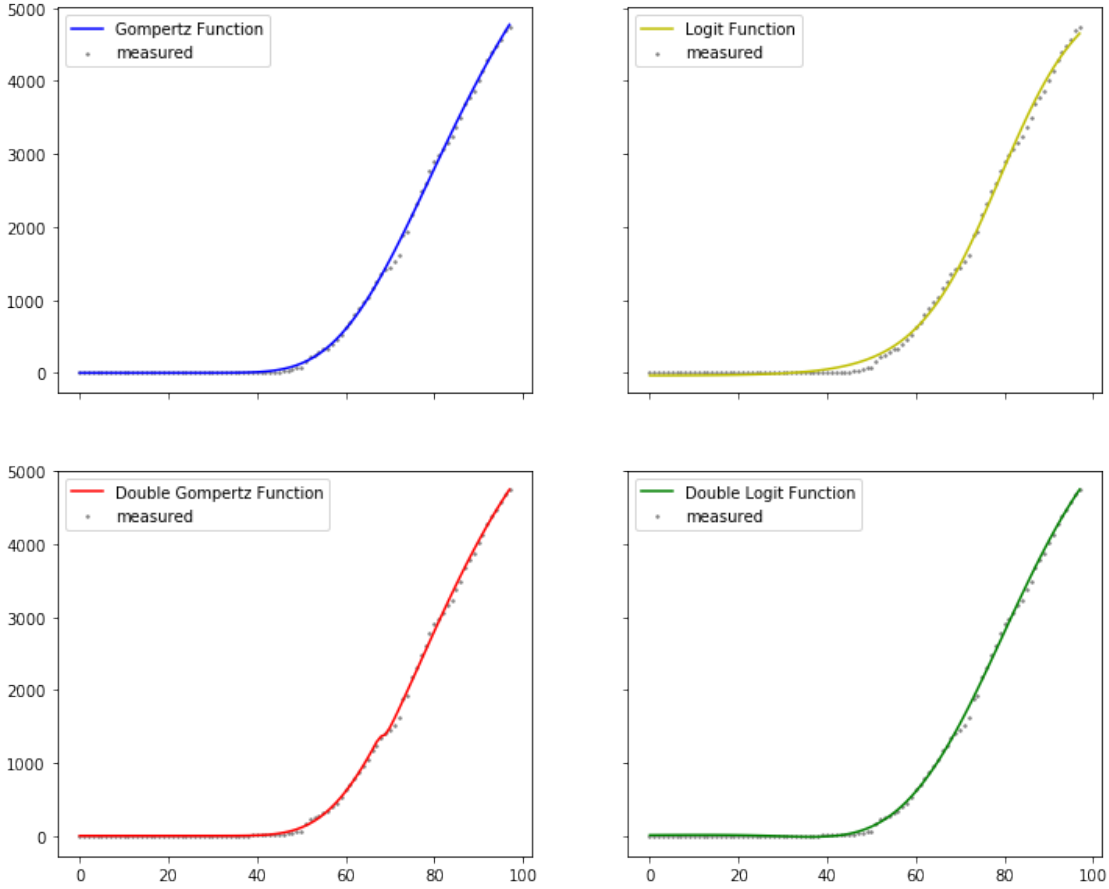

Gompertz Function
measured
700
600
500
400
300
200
100
0

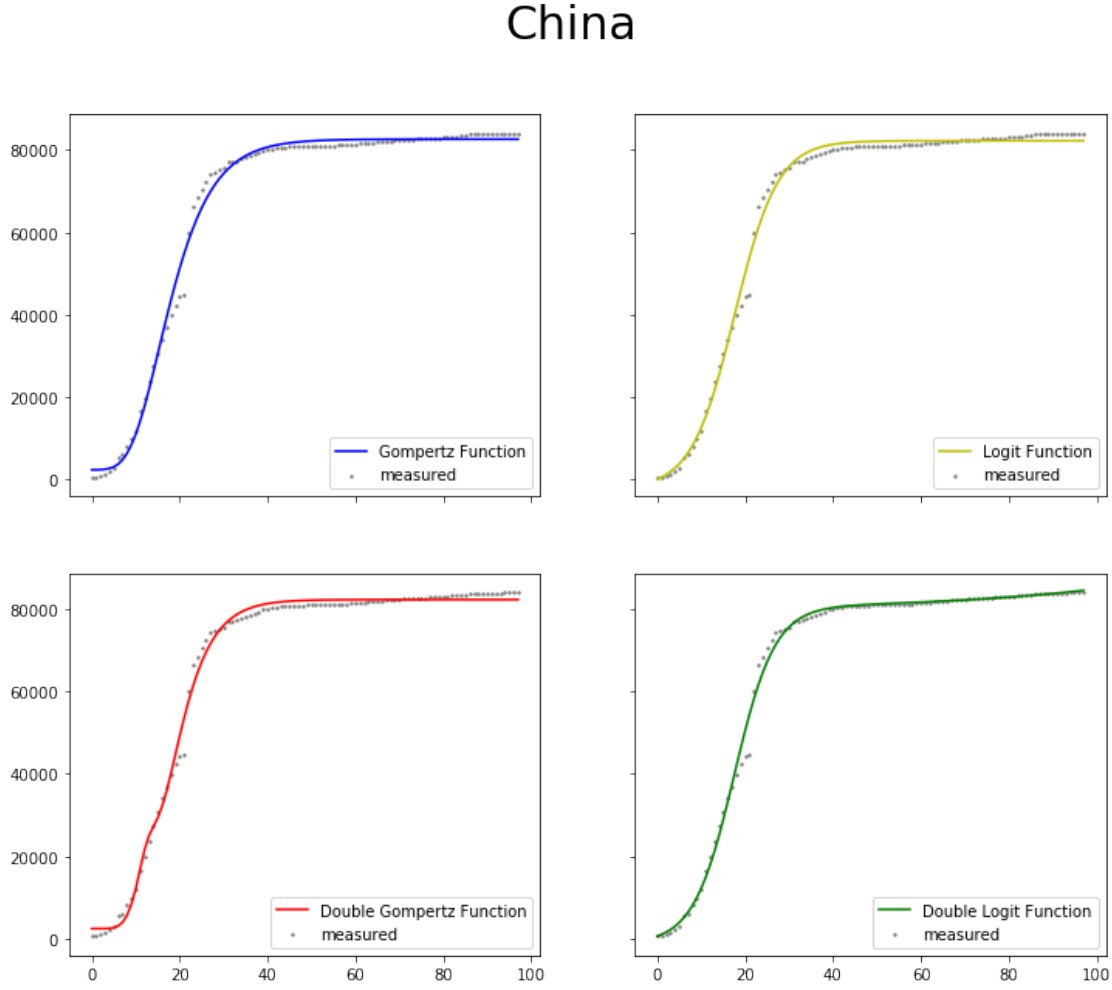

Logit Function
measured


Double Gompertz Function
measured
700
600
500
400
300
200
100
0
0
20
40
60
80
100


Double Logit Function
measured
0
20
40
60
80
100

Finland


Gompertz Function
measured
Logit Function
measured
Double Gompertz Function
measured
Double Logit Function
measured
5000
4000
3000
2000
1000
0
0
20
40
60
80
100



7 Appendix

7.1 Statistics formulas

In all formulas 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^2_{adj}: Adjusted Coefficient of Determination:

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

- **Pearson R**: Pearson Correlation Coefficient

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

- **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

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

- **ΔBIC**: Bayesian Information Criterion difference [FIG]:

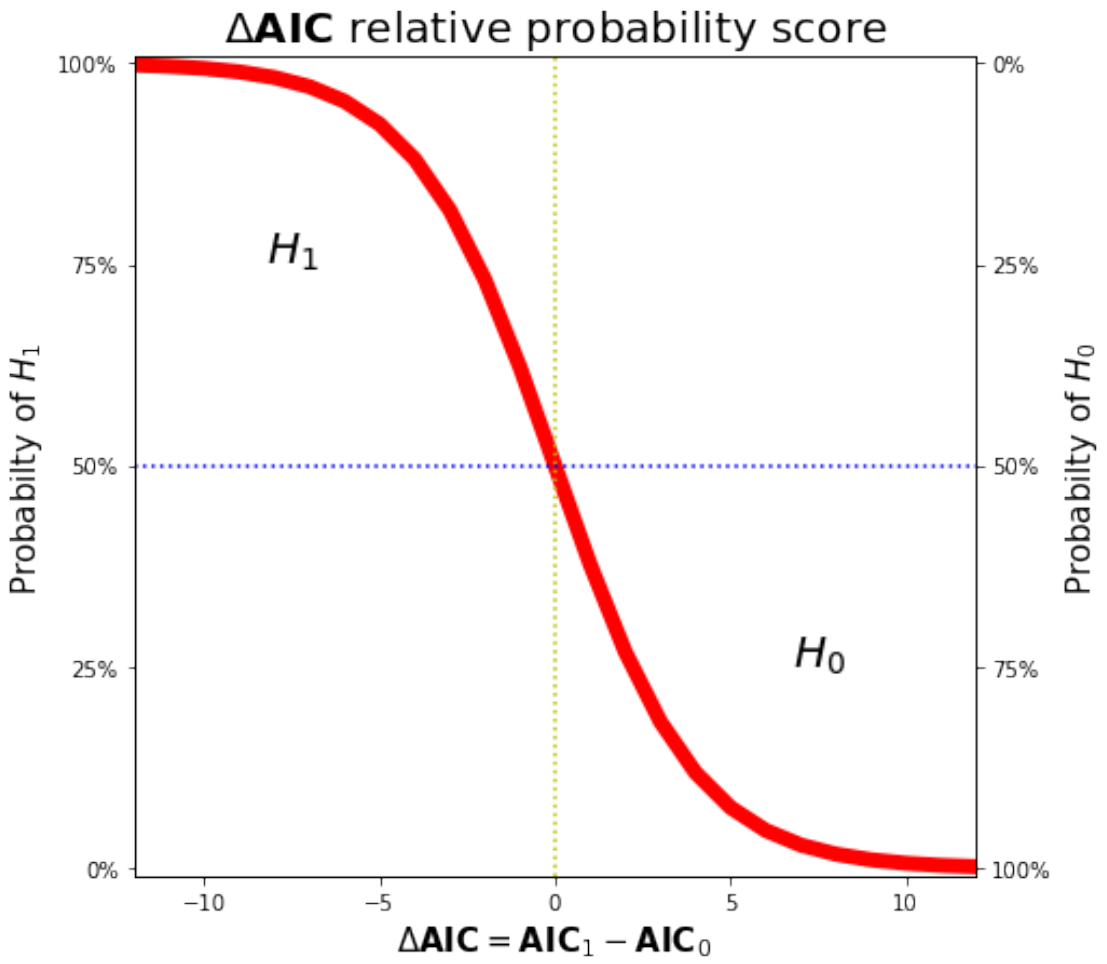
$$\mathbf{\Delta BIC} = \mathbf{BIC}_1 - \mathbf{BIC}_0$$

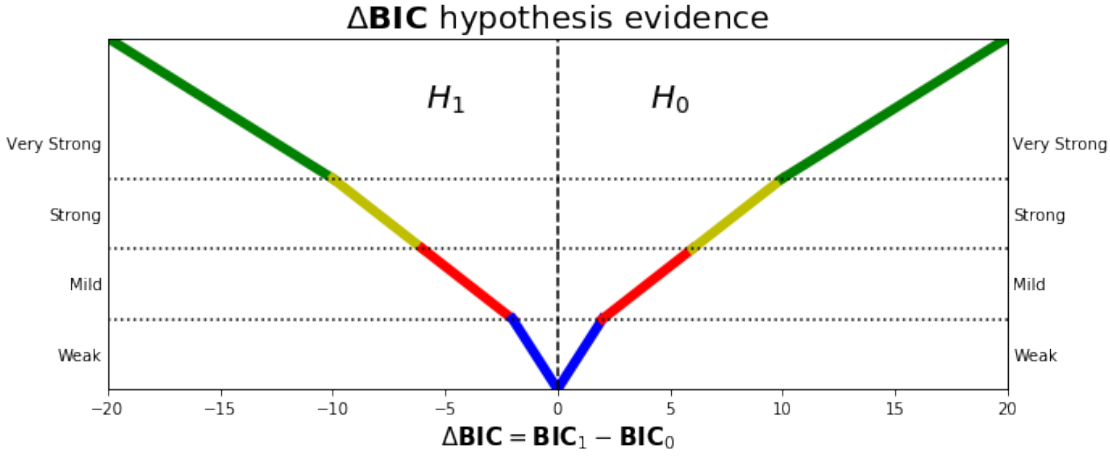
$$\mathbf{\Delta BIC} > 10 \Rightarrow H_0 \text{ Very strong evidence}$$

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

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

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


ΔAIC relative probability score
100%
75%
50%
25%
0%
0%
25%
50%
75%
100%
H_1
H_0
Probability of H_1
Probability of H_0
−10
−5
0
5
10
$\Delta\mathbf{AIC} = \mathbf{AIC}_1 - \mathbf{AIC}_0$


ΔBIC hypothesis evidence
H_1
H_0
Very Strong
Strong
Mild
Weak
Very Strong
Strong
Mild
Weak
−20
−15
−10
−5
0
5
10
15
20
$\Delta\mathbf{BIC} = \mathbf{BIC}_1 - \mathbf{BIC}_0$

[ ]: