Inferência Bayesiana para Quantificação de Incerteza em Estimação de Parâmetros

GUSTAVO BARBOSA LIBOTTE

Laboratório Nacional de Computação Científica
UNIDADE DE PESQUISA DO MCTI

MATERIAL SUPLEMENTAR

Python Notebook

https://github.com/gustavolibotte/eamc-2021-bayesian-inference

```
$ git clone
https://github.com/gustavolibotte/eamc-2021-bayesian-inference.git
```

launch binder **Notebook interativo online**

Obtenção de Dados de Repositório

+


```python
# Análise e manipulação de dados em larga escala
import pandas as pd
```

github.com/gustavolibotte/eamc-2021-bayesian-inference

```
data_url = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv"
confirmed_cases = pd.read_csv(data_url, sep = ",")
```

time_series_covid19_confirmed_global.csv

Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20
	Afghanistan	33.93911	67.709953	0	0	0	0	0
	Albania	41.1533	20.1683	0	0	0	0	0
	Algeria	28.0339	1.6596	0	0	0	0	0
	Andorra	42.5063	1.5218	0	0	0	0	0
	Angola	-11.2027	17.8739	0	0	0	0	0
	Antigua and Barbuda	17.0608	-61.7964	0	0	0	0	0
	Argentina	-38.4161	-63.6167	0	0	0	0	0
	Armenia	40.0691	45.0382	0	0	0	0	0
Australian Capital Territory	Australia	-35.4735	149.0124	0	0	0	0	0
New South Wales	Australia	-33.8688	151.2093	0	0	0	0	3
Northern Territory	Australia	-12.4634	130.8456	0	0	0	0	0
Queensland	Australia	-27.4698	153.0251	0	0	0	0	0
South Australia	Australia	-34.9285	138.6007	0	0	0	0	0
Tasmania	Australia	-42.8821	147.3272	0	0	0	0	0

Province/State,Country/Region,Lat,Long,1/22/20,1/23/20,1/24/20,1
2/10/20,2/11/20,2/12/20,2/13/20,2/14/20,2/15/20,2/16/20,2/17/20,
/4/20,3/5/20,3/6/20,3/7/20,3/8/20,3/9/20,3/10/20,3/11/20,3/12/20
20,3/28/20,3/29/20,3/30/20,3/31/20,4/1/20,4/2/20,4/3/20,4/4/20,4
/20,4/21/20,4/22/20,4/23/20,4/24/20,4/25/20,4/26/20,4/27/20,4/28
4/20,5/15/20,5/16/20,5/17/20,5/18/20,5/19/20,5/20/20,5/21/20,5/2
,6/7/20,6/8/20,6/9/20,6/10/20,6/11/20,6/12/20,6/13/20,6/14/20,6/
6/30/20,7/1/20,7/2/20,7/3/20,7/4/20,7/5/20,7/6/20,7/7/20,7/8/20,
,7/24/20,7/25/20,7/26/20,7/27/20,7/28/20,7/29/20,7/30/20,7/31/20
0,8/17/20,8/18/20,8/19/20,8/20/20,8/21/20,8/22/20,8/23/20,8/24/2
20,9/10/20,9/11/20,9/12/20,9/13/20,9/14/20,9/15/20,9/16/20,9/17/
2/20,10/3/20,10/4/20,10/5/20,10/6/20,10/7/20,10/8/20,10/9/20,10/
3/20,10/24/20,10/25/20,10/26/20,10/27/20,10/28/20,10/29/20,10/30
20,11/14/20,11/15/20,11/16/20,11/17/20,11/18/20,11/19/20,11/20/2
/4/20,12/5/20,12/6/20,12/7/20,12/8/20,12/9/20,12/10/20,12/11/20,
2/25/20,12/26/20,12/27/20,12/28/20,12/29/20,12/30/20,12/31/20,1/
/17/21,1/18/21,1/19/21,1/20/21,1/21/21,1/22/21,1/23/21,1/24/21,1
,Afghanistan,33.93911,67.709953,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
07,118,146,175,197,240,275,300,338,368,424,445,485,532,556,608,6
781,4042,4403,4687,4968,5227,5640,6054,6403,6665,7073,7654,8146,
22146,22894,23550,24106,24770,25531,26314,26878,27536,27882,2842
184,34356,34441,34595,34730,34984,35060,35219,35279,35453,35493,
3,37153,37260,37336,37422,37497,37542,37590,37667,37710,37750,37
38544,38572,38606,38641,38716,38772,38815,38855,38872,38897,3891
693,39703,39799,39870,39928,39994,40026,40088,40141,40200,40287,

```
country = "Germany"

init_date_cal = "2/20/20" # m/d/a
final_date_cal = "3/8/20" # m/d/a
```

```
# Obtenção dos dados do repositório
data_cases_df = confirmed_cases.loc[confirmed_cases["Country/Region"] == country, init_date_cal : final_date_cal]

# Variáveis com dados observáveis
daily_date_cal = (pd.to_datetime(data_cases_df.keys()).to_numpy())[1:]
daily_cases_cal = np.diff(data_cases_df.to_numpy()).ravel()

# Sequencia correspondente ao número de dias com dados disponíveis
data_time_cal = np.linspace(0, daily_cases_cal.size - 1, daily_cases_cal.size)
```

[5] data_cases_df:

```
     2/20/20  2/21/20  2/22/20  2/23/20  2/24/20  2/25/20  2/26/20  2/27/20  \
133       16       16       16       16       16       17       27       46

     2/28/20  2/29/20  3/1/20  3/2/20  3/3/20  3/4/20  3/5/20  3/6/20  3/7/20  \
133       48       79     130     159     196     262     482     670     799

     3/8/20
133    1040
```

```
num_days_predict = 7
init_date_pred = "3/9/20" # m/d/a
final_date_pred = datetime.strptime(final_date_cal, "%m/%d/%y") + timedelta(days = num_days_predict)
if platform.system() == "Windows":
    final_date_pred = final_date_pred.strftime("%#m/%#d/%y")
else:
    final_date_pred = final_date_pred.strftime("%-m/%-d/%y")
```

```
# Obtenção dos dados do repositório
data_cases_df = confirmed_cases.loc[confirmed_cases["Country/Region"] == country, init_date_pred : final_date_pred]

# Variáveis com dados observáveis
daily_date_pred = (pd.to_datetime(data_cases_df.keys()).to_numpy())[1:]
daily_cases_pred = np.diff(data_cases_df.to_numpy()).ravel()

# Sequencia correspondente ao número de dias com dados disponíveis
data_time_pred = np.linspace(data_time_cal[-1] + 1, data_time_cal[-1] + daily_cases_pred.size, daily_cases_pred.size)
```

[2] data_cases_df:

	3/9/20	3/10/20	3/11/20	3/12/20	3/13/20	3/14/20	3/15/20
133	1176	1457	1908	2078	3675	4585	5795

github.com/gustavolibotte/eamc-2021-bayesian-inference

```python
fig, ax = plt.subplots(figsize=[8, 4])

ax.plot(daily_date_cal, daily_cases_cal, marker="o", linewidth=0, label="Dados de calibração")
ax.plot(daily_date_pred, daily_cases_pred, marker="o", linewidth=0, label="Dados para conferência")

# Definição do formato da data no eixo horizontal
ax.xaxis.set_major_formatter(mdates.DateFormatter('%d/%m/%y'))
ax.xaxis.set_major_locator(mdates.DayLocator(interval=5))

ax.set_xlabel("Data")
ax.set_ylabel("Quantidade diária de infectados");

ax.grid()

ax.legend();
```

github.com/gustavolibotte/eamc-2021-bayesian-inference

Definição do Modelo

- A taxa de transmissão por contato entre indivíduos suscetíveis (S) e infectados (I) é dada por β.
- A taxa a que os indivíduos são removidos (R), por terem se recuperado ou morrido da doença, é dada por γ.

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -\beta\frac{SI}{N}$$

$$\frac{\mathrm{d}I}{\mathrm{d}t} = \beta\frac{SI}{N} - \gamma I$$

$$\frac{\mathrm{d}R}{\mathrm{d}t} = \gamma I$$

```julia
def SIR_model(t, Y, β, γ, N):
    S, I, R = Y
    dS = -β / N * S * I
    dI = β / N * S * I - γ * I
    dR = γ * I
    return dS, dI, dR
```

```python
# Solver para o sistema de equações diferenciais
from scipy.integrate import solve_ivp
```

```python
def model_solver(t_span, y0, t_eval, N, β, γ):
    model_pars = [β, γ, N]
    sol_ODE = solve_ivp(
        fun=lambda t, Y: SIR_model(
            t,
            Y,
            *model_pars
        ),
        t_span=t_span,
        y0=y0,
        method="LSODA",
        t_eval=t_eval
    )
    y_sol = sol_ODE.y
    return y_sol
```

github.com/gustavolibotte/eamc-2021-bayesian-inference

```python
@theano.compile.ops.as_op(
    itypes=[
        t.dvector, # time_exp
        t.dscalar, # β
        t.dscalar, # γ
        t.dscalar, # I0
        t.dscalar, # R0
        t.dscalar, # N
    ],
    otypes=[t.dvector],
)
def model_wrapper(time_exp, β, γ, I0, R0, N):
    time_span = (time_exp[0], time_exp[-1])

    S0 = N - (I0 + R0)
    ICs = (S0, I0, R0)

    args = [β, γ]

    y_model = model_solver(time_span, ICs, time_exp, N, *args)
    infected_simulated = y_model[1]

    return infected_simulated
```

⚠ **Veja mais sobre** theano **em:** https://pypi.org/project/Theano/

Estimação de Parâmetros

Inferência Bayesiana

- A principal característica da **inferência Bayesiana** é combinar novas evidências com informações anteriores através do uso do Teorema de Bayes.

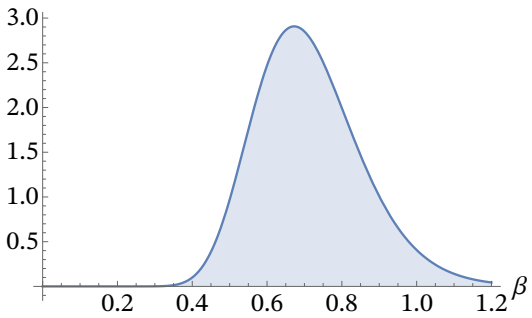
$$p_{\text{post}}(\theta \mid \mathcal{D}) \propto p_{\text{like}}(\mathcal{D} \mid \theta)\, p_{\text{prior}}(\theta)$$

  - Distribuição a priori $p_{\text{prior}}(\theta)$: reflete o conhecimento sobre o valor de θ, sem interferência dos dados $\mathcal{D}$;
  - Função de verossimilhança $p_{\text{like}}(\mathcal{D} \mid \theta)$: mensura a probabilidade do modelo de caracterizar as observações disponíveis $\mathcal{D}$, dado um conjunto de parâmetros θ;
  - Distribuição a posterior $p_{\text{post}}(\theta \mid \mathcal{D})$: Distribuição resultante de θ, que incorpora a informação inicial sobre o parâmetro e a informação oriunda dos dados

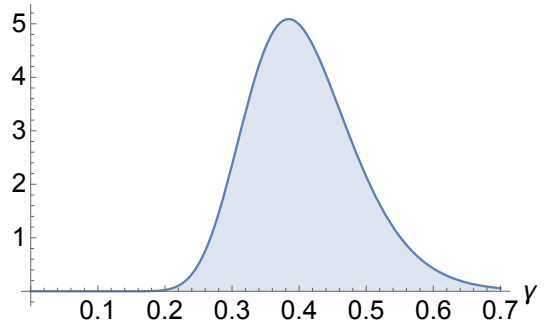
github.com/gustavolibotte/eamc-2021-bayesian-inference

- Distribuição a priori dos parâmetros do modelo:

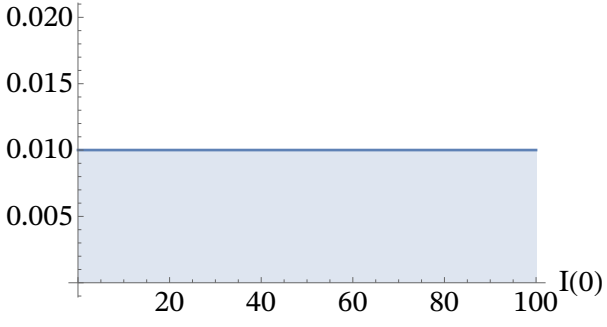
$\beta \sim \text{Lognormal}(\ln(0{,}7),\ 0{,}2)$



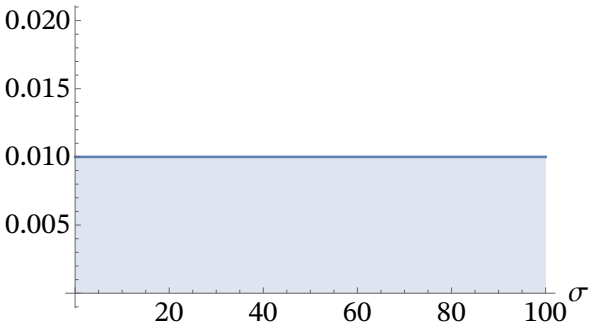
$\gamma \sim \text{Lognormal}(\ln(0{,}4),\ 0{,}2)$



$I(0) \sim \text{Uniform}(0,\ 100)$



$\sigma \sim \text{Uniform}(0,\ 100)$



- O procedimento de inferência é realizado usando a biblioteca PyMC3.
- Um dos algoritmos que a biblioteca fornece é o **Sequential Monte Carlo** (SMC).
- Algoritmo populacional que funciona passando por estágios em um esquema de recozimento.
- O estado inicial mais "quente" equivale à distribuição a priori de θ.
- Quando o sistema atinge a temperatura mais "fria", espera-se obter a distribuição a posteriori de θ.
- O resfriamento lento possibilita encontrar os picos da distribuição.

Ching, J., & Chen, Y.-C. (2007). Transitional Markov Chain Monte Carlo Method for Bayesian Model Updating, Model Class Selection, and Model Averaging. Journal of Engineering Mechanics, 133(7), 816–832. https://doi.org/10.1061/(asce)0733-9399(2007)133:7(816)

- As amostras são computadas avaliando-se $p_{\text{post}}(\theta \mid \mathcal{D})_\tau \propto p_{\text{like}}(\mathcal{D} \mid \theta)^\tau \, p_{\text{prior}}(\theta)$.
- Os passos principais do algoritmo são dados a seguir:

1. Inicialize $\tau^{(k)} = 0$ no estágio inicial $k = 0$, gere um conjunto $\left\{\theta_i^{(k)}\right\}$ de m amostras para $i = 1, \ldots, m$ e faça $k = 1$.
2. Aumente $\tau^{(k)}$ de forma que $\tau^{(k)} - \tau^{(k-1)} = \epsilon$, com $\epsilon \to 0$.
3. Compute um conjunto $\left\{\omega_i^{(k)}\right\}$ de m pesos que respeitam a relação $\omega_i^{(k)} = p_{\text{like}}\left(\mathcal{D} \mid \theta_i^{(k-1)}\right)^{\tau^{(k)} - \tau^{(k-1)}}$, para $i = 1, \ldots, m$.
4. Calcule a média dos pesos $S^{(k)} = \frac{1}{m} \sum_{i=1}^{m} \omega_i^{(k)}$ e a ponderação das amostras $\bar{\theta}^{(k)} = \sum_{i=1}^{m} \frac{\omega_i^{(k)} \theta_i^{(k-1)}}{\sum_{j=1}^{m} \omega_j^k}$.
5. Calcule a matriz de covariância por $\boldsymbol{\Sigma}^{(k)} = \frac{\eta^2}{m} \sum_{i=1}^{m} \frac{\omega_i^{(k)}}{S^{(k)}} \left(\theta_i^{(k-1)} - \bar{\theta}^{(k-1)}\right)\left(\theta_i^{(k-1)} - \bar{\theta}^{(k-1)}\right)^\top$, onde η é um fator de escala.
6. Para cada amostra, selecione $j \in \{1, \ldots, m\}$ aleatoriamente. Aceite $\theta^{\text{c}} \sim \mathcal{N}\left(\theta_j^{(k-1)}, \boldsymbol{\Sigma}^{(k)}\right)$ se $r \leq \frac{p_{\text{like}}\left(\mathcal{D} \mid \theta^{\text{c}}\right)^{\tau^{(k)}} p_{\text{prior}}\left(\theta^{\text{c}}\right)}{p_{\text{like}}\left(\mathcal{D} \mid \theta_j^{(k-1)}\right)^{\tau^{(k)}} p_{\text{prior}}\left(\theta_j^{(k-1)}\right)}$, onde r é uniformemente distribuído entre $[0, 1]$.
7. Faça $k = k + 1$ e repita os passos 2–6 enquanto $\tau^{(k)} < 1$.

- A função de verossimilhança (*likelihood*) é dada por

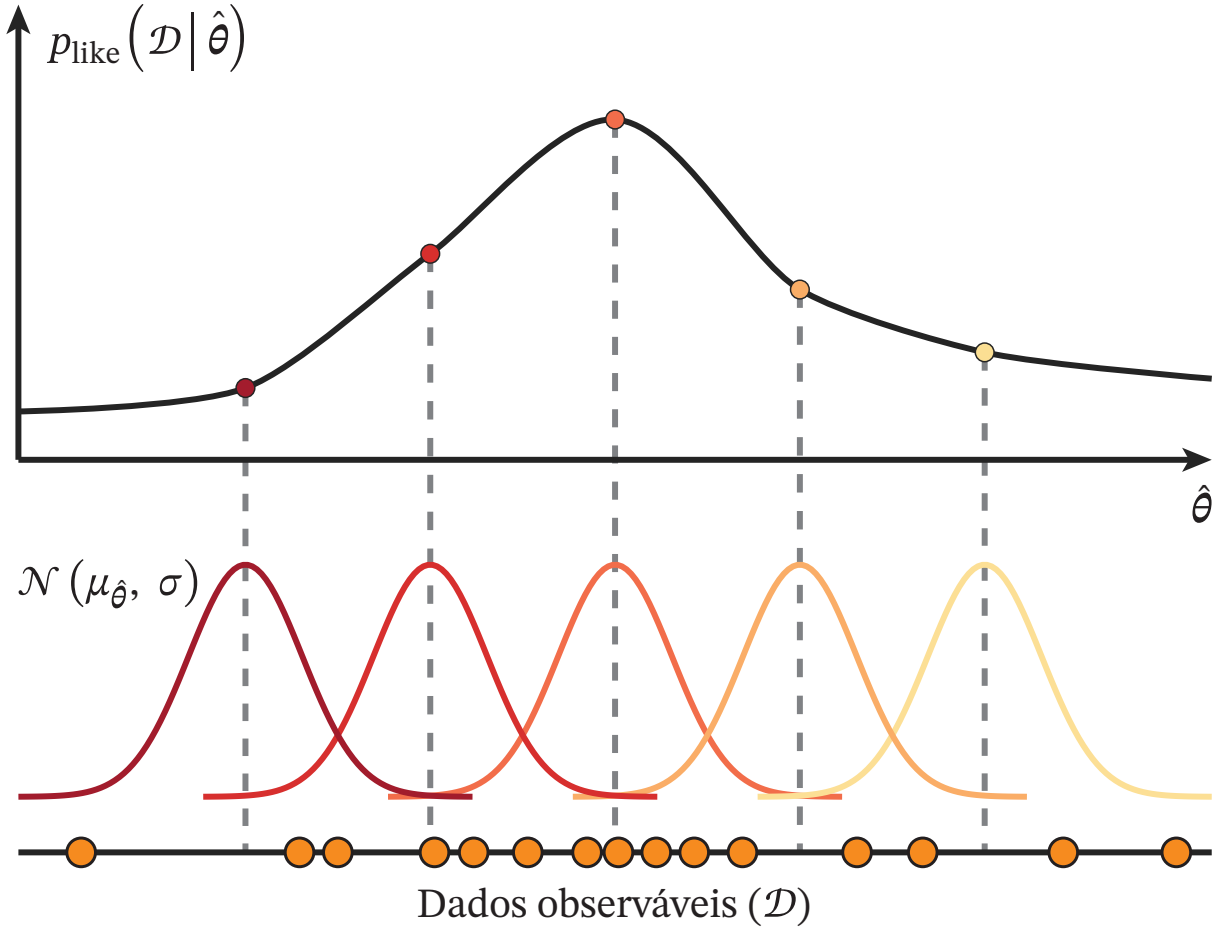
$$p_{\text{like}}\left(\mathcal{D} \mid \boldsymbol{\theta}\right) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\sum_{\ell=1}^{p} \frac{\left(\mathcal{D}\left(t_\ell\right) - y\left(t_\ell, \boldsymbol{\theta}\right)\right)^2}{2\sigma^2}\right).$$

  - t_ℓ: instante de tempo;
  - $\mathcal{D}\left(t_\ell\right)$: dado disponível em t_ℓ;
  - $y\left(t_\ell, \boldsymbol{\theta}\right)$: resposta do modelo em t_ℓ para um conjunto de parâmetros $\boldsymbol{\theta}$;
  - σ: desvio padrão (não conhecido, tratado como hiperparâmetro do modelo).

```python
# Calibração Bayesiana
import pymc3 as pm

num_draws = 3000
calibration_variable_names = ["β", "γ", "I(0)", "σ"]
```

```python
with pm.Model() as model_mcmc:
    β = pm.Lognormal(calibration_variable_names[0], mu=np.log(0.7), sigma=0.2)
    γ = pm.Lognormal(calibration_variable_names[1], mu=np.log(0.4), sigma=0.2)
    init_I = pm.Uniform(calibration_variable_names[2], lower=0, upper=100)
    σ = pm.Uniform(calibration_variable_names[3], lower=1, upper=100)

    fitting_model = pm.Deterministic(
        "SIR_model",
        model_wrapper(
            theano.shared(data_time_cal),
            β,
            γ,
            init_I,
            theano.shared(R0),
            theano.shared(N),
        ),
    )

    likelihood = pm.Normal("likelihood", mu=fitting_model, sigma=σ, observed=daily_cases_cal)

    trace_calibration = pm.sample_smc(
        draws=num_draws,
        n_steps=25,
        parallel=True,
        cores=8,
        progressbar=True,
        random_seed=12345
    )
```

github.com/gustavolibotte/eamc-2021-bayesian-inference


$p_{\text{like}}\left(\mathcal{D} \mid \hat{\theta}\right)$
$\hat{\theta}$
$\mathcal{N}\left(\mu_{\hat{\theta}},\ \sigma\right)$
Dados observáveis ($\mathcal{D}$)

- O algoritmo anterior é capaz de gerar amostras diretamente da distribuição a posteriori e, assim, o cálculo das evidências pode ser evitado.
- O termo correspondente à evidência pode ser aproximado por $p_{\text{evid}}(\mathcal{D}) \approx \prod_{k=0}^{K} S^{(k)}$, onde K é o número total de estágios.

```python
# Suporte a resultados e gráficos dos dados estatísticos
import arviz as az
```

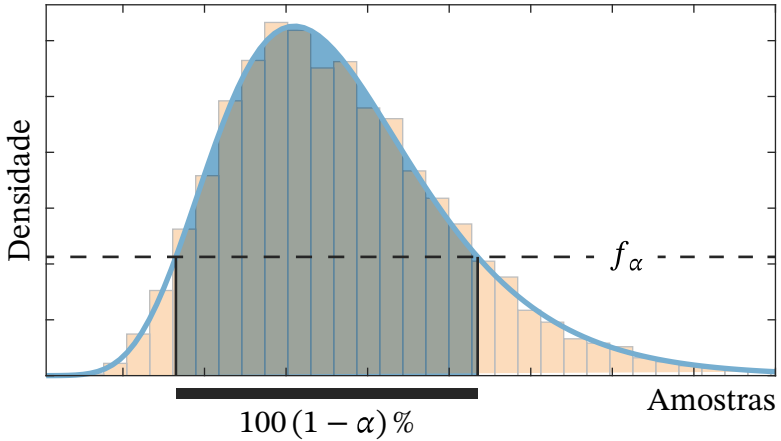
```python
trace_summary = az.summary(
    trace_calibration,
    var_names=calibration_variable_names,
    kind="stats",
    hdi_prob=0.95,
    round_to=3
)

print(trace_summary)
```

	mean	sd	hdi_2.5%	hdi_97.5%
β	0.646	0.074	0.504	0.790
γ	0.413	0.070	0.275	0.548
I(0)	6.581	4.027	0.847	14.093
σ	40.474	8.342	25.494	56.537

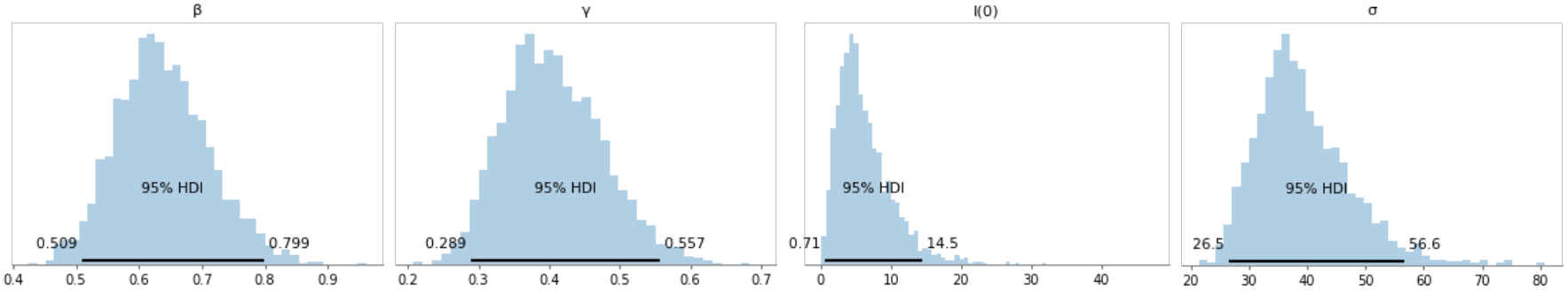
- O HDI (*highest density interval*) indica quais pontos de uma distribuição são mais confiáveis e quais cobrem a maior parte da distribuição.
- Resume a distribuição especificando um intervalo onde cada ponto tenha maior credibilidade do que qualquer ponto fora do intervalo.
- **Definição**: Seja $f(x)$ a função densidade de probabilidade de uma variável aleatória X. O intervalo de credibilidade de $100(1-\alpha)\,\%$ é o subconjunto $Q(f_\alpha)$ de amostras de X tal que

$$Q(f_\alpha) = \{x : f(x) \geq f_\alpha\},$$

onde f_α é a maior constante tal que $\mathrm{P}(X \in Q(f_\alpha)) \geq 1-\alpha$.



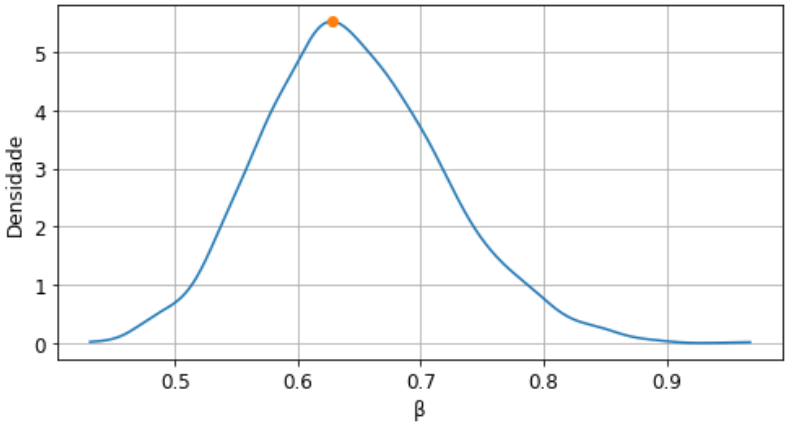
```python
for variable in calibration_variable_names:
    ax = az.plot_posterior(
        trace_calibration,
        var_names=(f"{variable}"),
        figsize=[4, 3],
        kind="hist",
        point_estimate=None,
        hdi_prob=0.95,
        round_to=3
    )
```

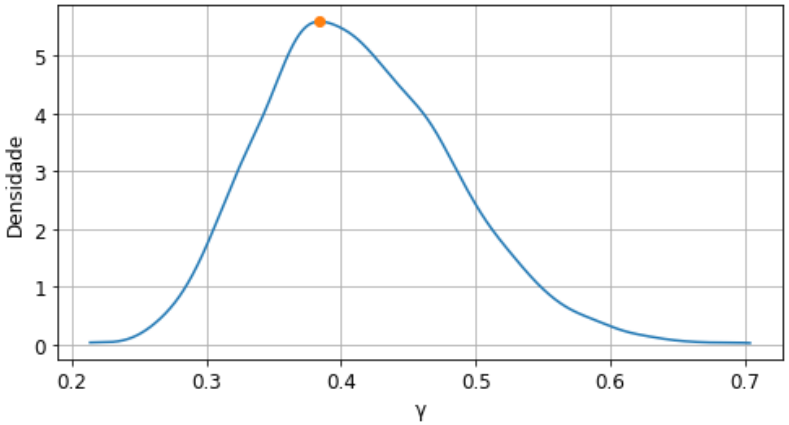


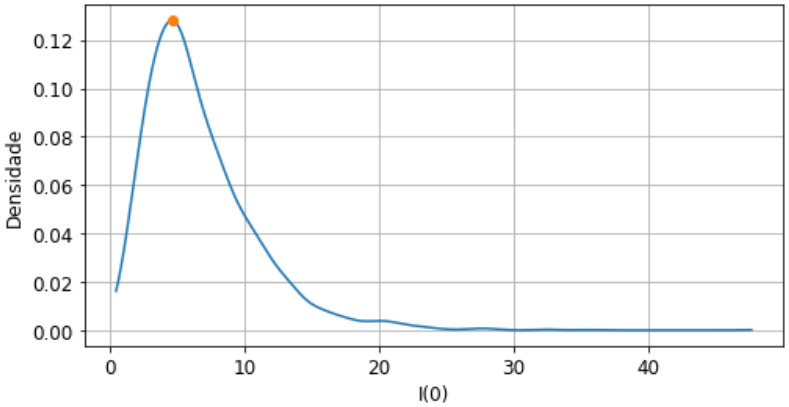
github.com/gustavolibotte/eamc-2021-bayesian-inference

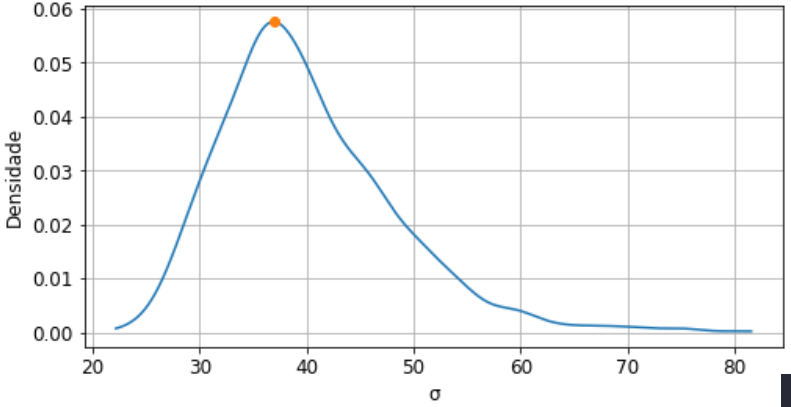
```python
map_params = {}
for param in calibration_variable_names:
    range_param = np.linspace(trace_calibration[param].min(), trace_calibration[param].max(), num_draws)

    # Kernel density estimator
    kernel = gaussian_kde(trace_calibration[param])

    # Calcula o kernel no intervalo definido
    kde = kernel(range_param)

    # Obtém o índice do ponto máximo
    kde_max_index = np.argmax(kde)

    # Obtém o valor correspondente ao índice do ponto máximo
    rv_mpv_value = range_param[kde_max_index]

    # Salva no dicionário
    MAP = rv_mpv_value
    map_params[param] = np.round(float(MAP), 5)

    # Criação da figura
    fig, ax = plt.subplots(figsize=[8, 4])
    ax.plot(range_param, kde)
    ax.plot(rv_mpv_value, kernel(MAP), "o")
    ax.set_xlabel(f"{param}");
    ax.set_ylabel("Densidade")
    ax.grid()
```

github.com/gustavolibotte/eamc-2021-bayesian-inference







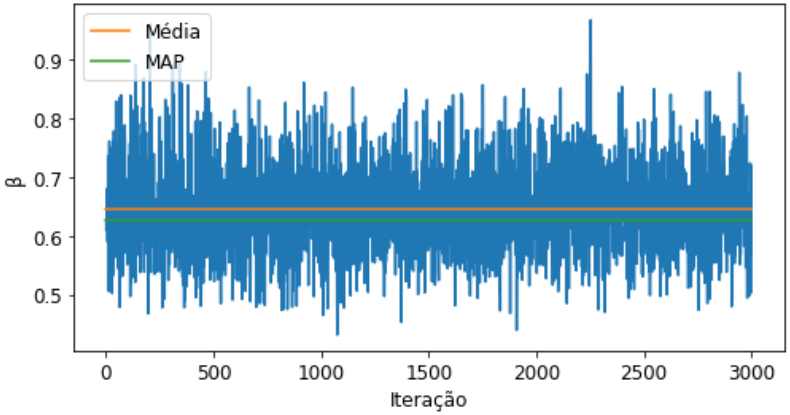


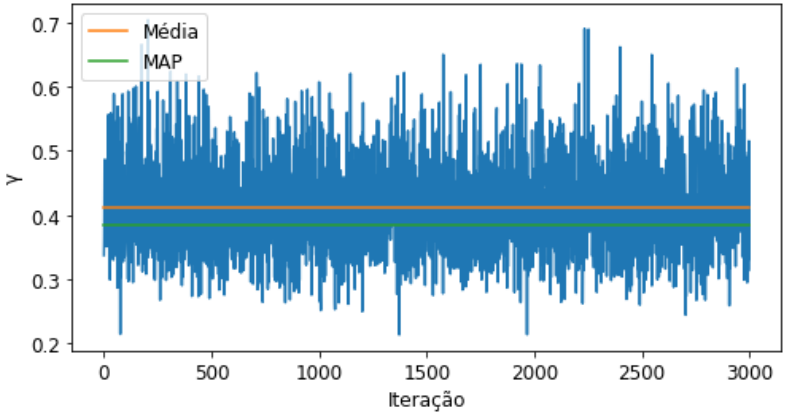
github.com/gustavolibotte/eamc-2021-bayesian-inference

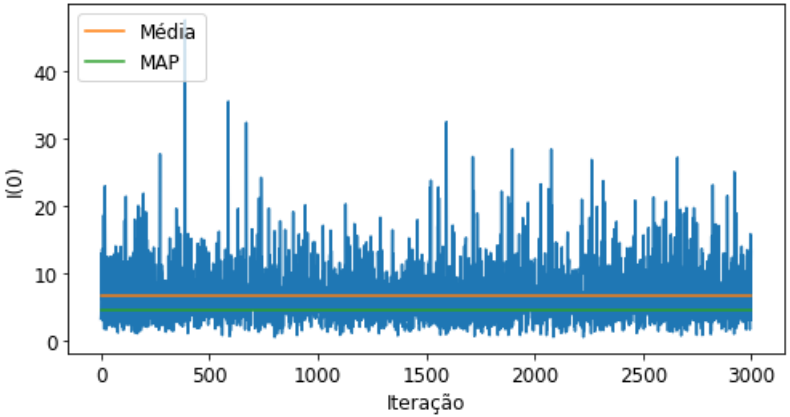
```
num_draws = 3000
calibration_variable_names = ["β", "γ", "I(0)", "σ"]
```

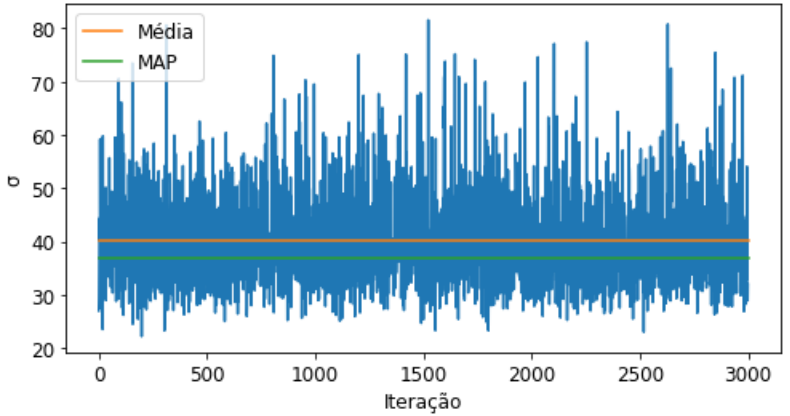
```
for variable in calibration_variable_names:
    # Criação da figura
    fig, ax = plt.subplots(figsize=[8, 4])

    # Plotagem das iterações
    ax.plot(trace_calibration[f"{variable}"], color="C0")

    # Plotagem da média
    mean_variable = trace_summary["mean"][f"{variable}"]
    ax.plot([0, num_draws], [mean_variable, mean_variable], "-", label="Média", color="C1")

    # Plotagem do MAP
    ax.plot([0, num_draws], [map_params[variable], map_params[variable]], "-", label="MAP", color="C2")

    # Atribuição dos nomes dos eixos
    ax.set_xlabel("Iteração")
    ax.set_ylabel(f"{variable}");

    # Legenda
    ax.legend(loc="upper left")
```

github.com/gustavolibotte/eamc-2021-bayesian-inference



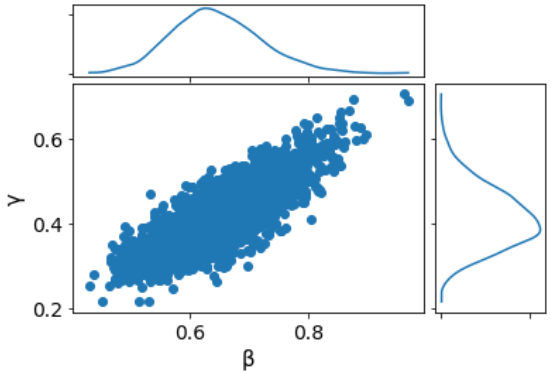


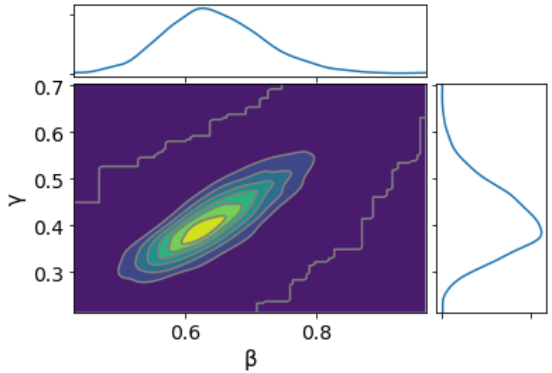


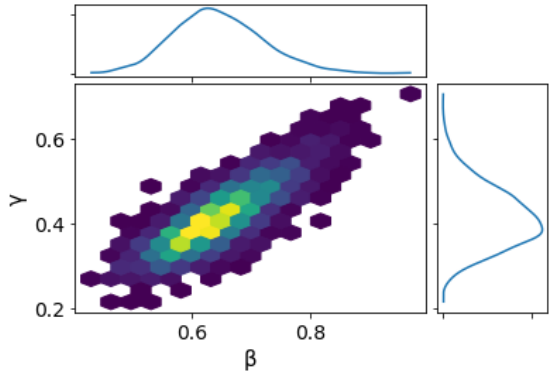


github.com/gustavolibotte/eamc-2021-bayesian-inference

```
az.plot_joint(trace_calibration, var_names=["β", "γ"], kind='scatter');
                                                # kind='scatter'|'kde'|'hexbin'
```







github.com/gustavolibotte/eamc-2021-bayesian-inference

Simulação do Modelo com Parâmetros Ótimos

```
print(map_params)
```

```
{'β': 0.62744, 'γ': 0.38436, 'I(0)': 4.61782, 'σ': 36.91728}
```

```
print(data_time_cal)
```

```
[ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16.]
```

```
time_span = (data_time_cal[0], data_time_cal[-1])
S0 = N - (map_params["I(0)"] + R0)
ICs = (S0, map_params["I(0)"], R0)
map_args = [map_params["β"], map_params["γ"]]

y_fit = model_solver(time_span, ICs, data_time_cal, N, *map_args)
infected_calibration = y_fit[1]
```

github.com/gustavolibotte/eamc-2021-bayesian-inference

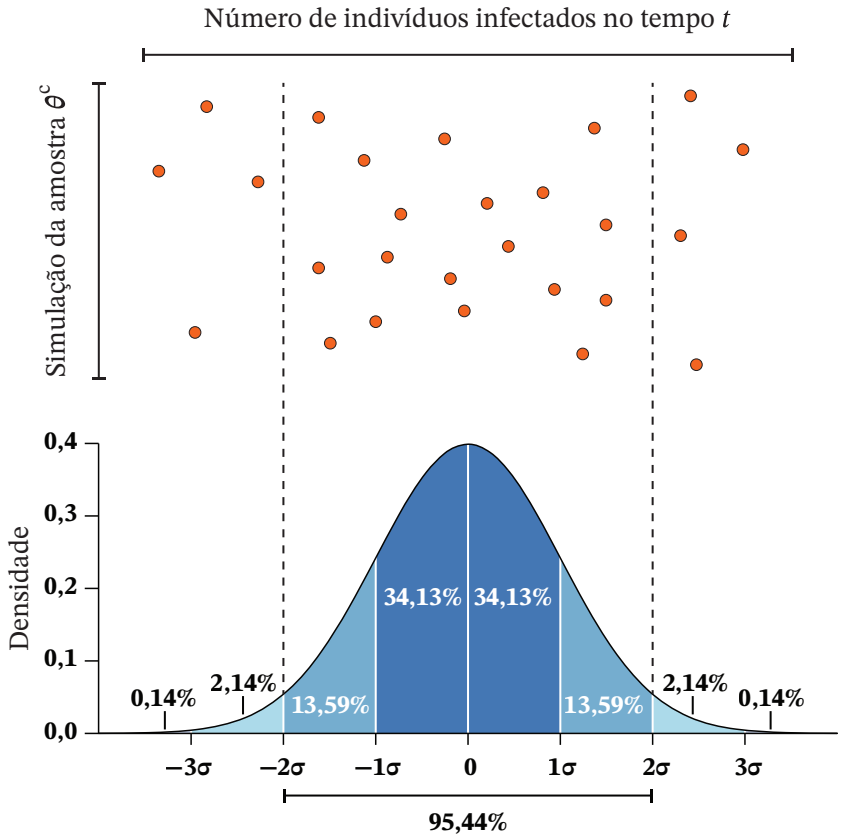
- Matriz de dimensão 3000×17 com simulações obtidas usando amostras do SMC.

```
print(trace_calibration["SIR_model"])
```

```
[[  3.21641184   4.22567472   5.55171324 ... 146.91554104 193.03695818
  253.59098235]
 [  3.8230431    4.95898362   6.43254649 ... 146.05434939 189.47583913
  245.76418688]
 [  9.16863598  11.13545758  13.52435295 ... 139.37961314 169.27831981
  205.58323833]
 ...
 [ 10.64000469  12.71631359  15.19793515 ... 129.12608272 154.33833337
  184.45890953]
 [  9.73572886  11.95016782  14.66851713 ... 171.66922523 210.73499359
  258.67390612]
 [ 13.22281886  15.70543868  18.65432762 ... 147.12419199 174.75030881
  207.56540081]]
```

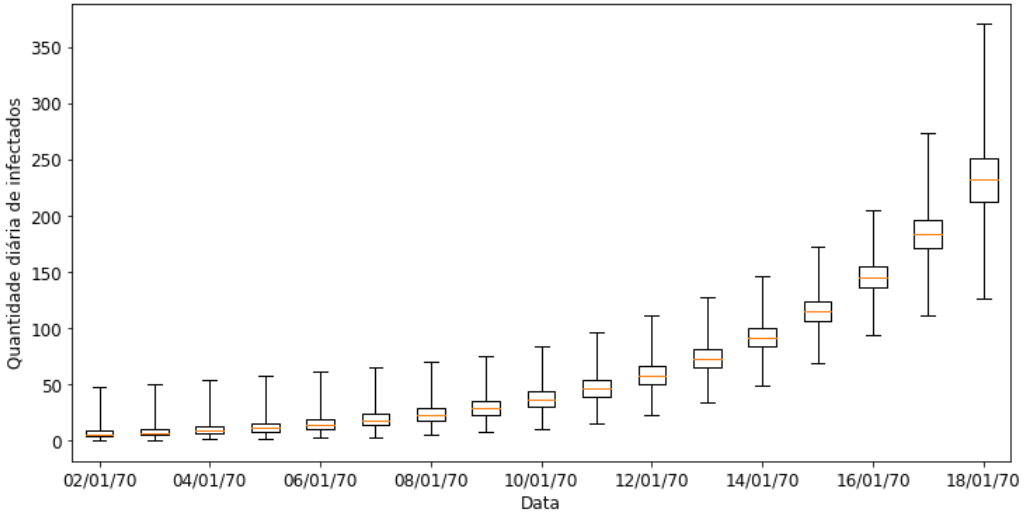
```
percentile_cut = 2.5
```

```
y_min = np.percentile(
    trace_calibration["SIR_model"],
    percentile_cut,
    axis=0
)
y_max = np.percentile(
    trace_calibration["SIR_model"],
    100 - percentile_cut,
    axis=0
)
```

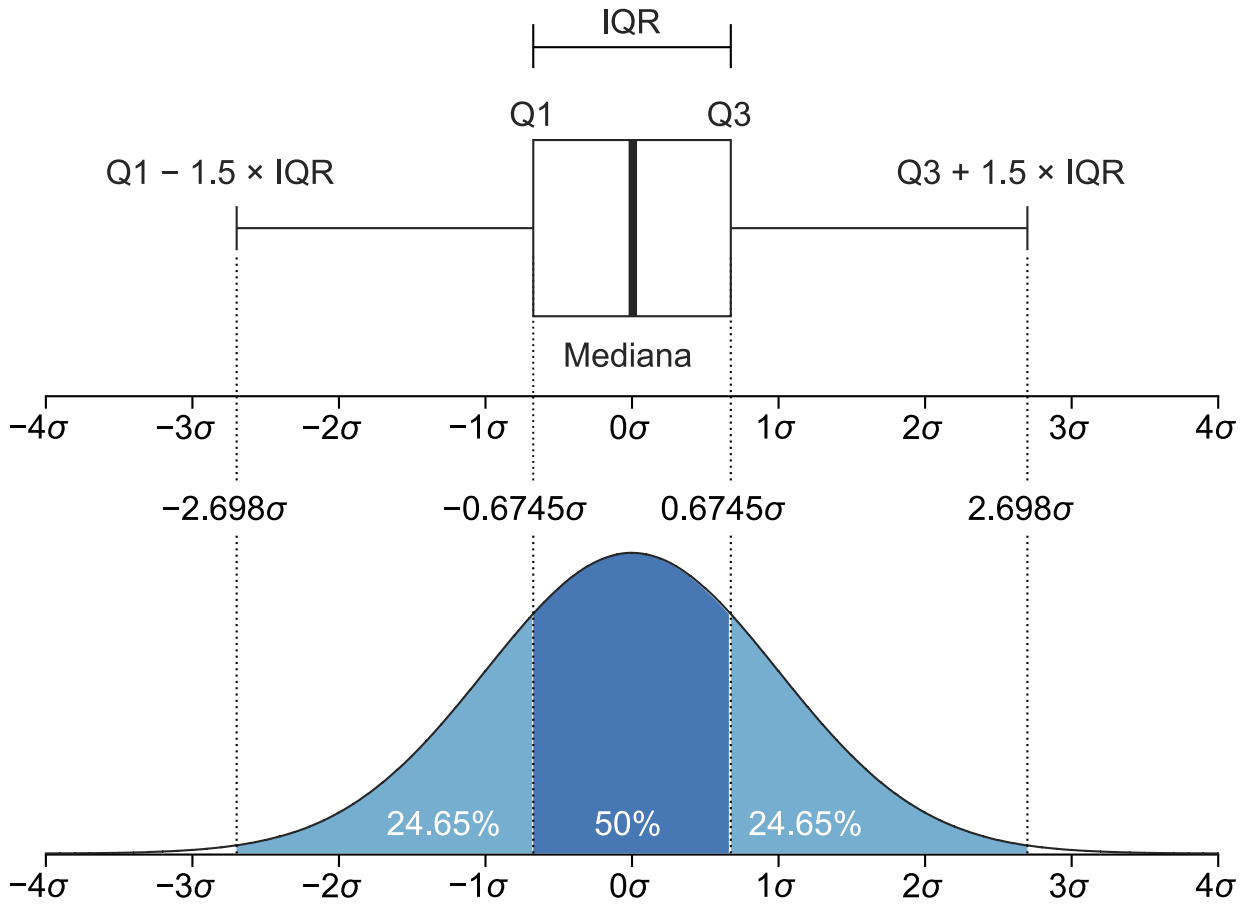


github.com/gustavolibotte/eamc-2021-bayesian-inference

```python
fig, ax = plt.subplots(figsize=[12, 6])
ax.boxplot(trace_calibration["SIR_model"], whis=(0,100));

ax.xaxis.set_major_formatter(mdates.DateFormatter('%d/%m/%y'))
ax.xaxis.set_major_locator(mdates.DayLocator(interval=2))

ax.set_xlabel("Data")
ax.set_ylabel("Quantidade diária de infectados");
```

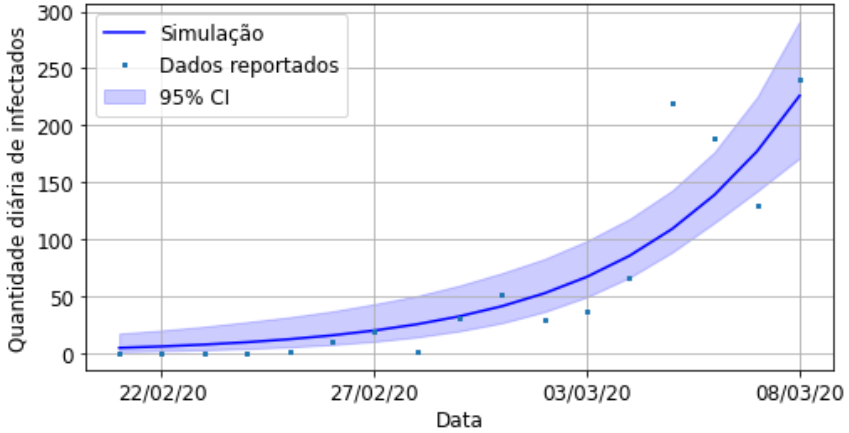


github.com/gustavolibotte/eamc-2021-bayesian-inference



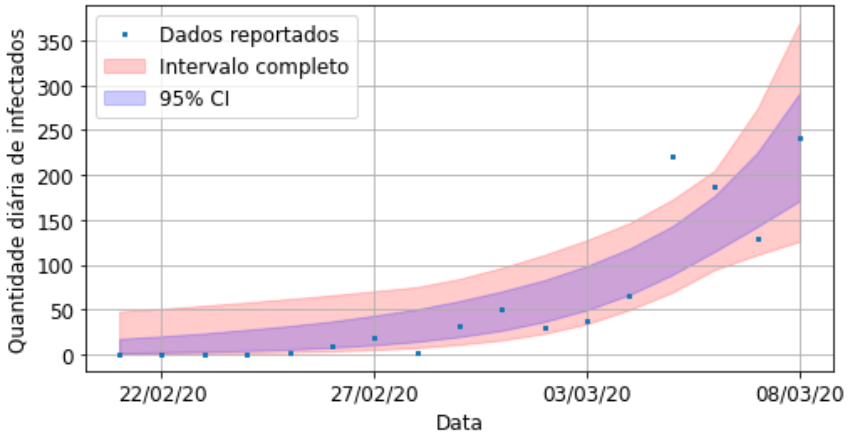
github.com/gustavolibotte/eamc-2021-bayesian-inference

```
fig, ax = plt.subplots(figsize=[8, 4])
ax.plot(daily_date_cal, infected_calibration, "b", label="Simulação", linestyle="-", markersize=2)
ax.fill_between(daily_date_cal, y_min, y_max, label="95% CI", color="b", alpha=0.2)
ax.plot(daily_date_cal, daily_cases_cal, label="Dados reportados", marker="s", linestyle="", markersize=2)

ax.xaxis.set_major_formatter(mdates.DateFormatter('%d/%m/%y'))
ax.xaxis.set_major_locator(mdates.DayLocator(interval=5))

ax.set_xlabel("Data")
ax.set_ylabel("Quantidade diária de infectados");
ax.legend()
ax.grid()
```



github.com/gustavolibotte/eamc-2021-bayesian-inference

```python
y_min_total = np.percentile(trace_calibration["SIR_model"], 0, axis=0)
y_max_total = np.percentile(trace_calibration["SIR_model"], 100, axis=0)

fig, ax = plt.subplots(figsize=[8, 4])

ax.fill_between(daily_date_cal, y_min_total, y_max_total, label="Intervalo completo", color="r", alpha=0.2);
ax.fill_between(daily_date_cal, y_min, y_max, label="95% CI", color="b", alpha=0.2);
ax.plot(daily_date_cal, daily_cases_cal, label="Dados reportados", marker="s", linestyle="", markersize=2)
```



github.com/gustavolibotte/eamc-2021-bayesian-inference

Predição

```
data_time_cal_pred = np.concatenate((data_time_cal, data_time_pred))
time_span = (data_time_cal_pred[0], data_time_cal_pred[-1])

y_fit = model_solver(time_span, ICs, data_time_cal_pred, N, *map_args)
susceptible_prediction = y_fit[0]
infected_prediction = y_fit[1]
removed_prediction = y_fit[2]
```

[1] data_time_cal_pred:

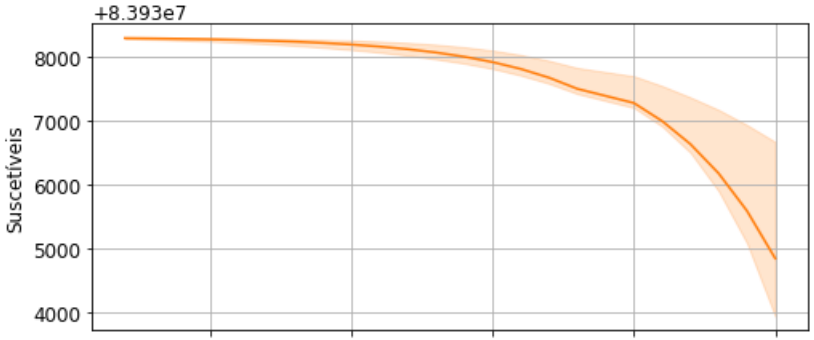
```
[ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16. 17.
 18. 19. 20. 21. 22.]
```

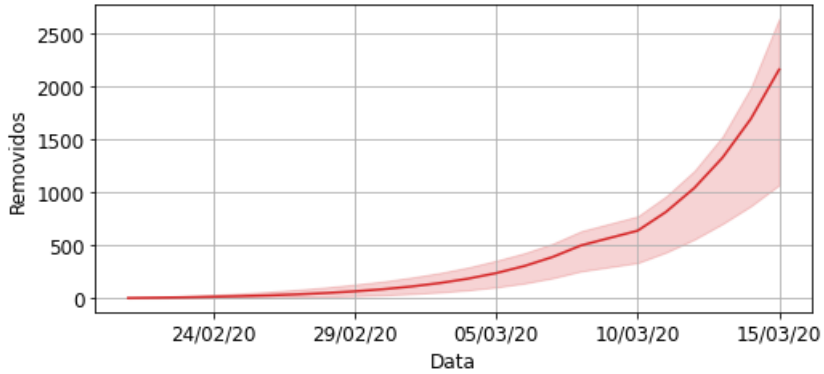
```python
susceptible_pred = np.zeros((num_draws, data_time_cal_pred.size))
infected_pred = np.zeros((num_draws, data_time_cal_pred.size))
removed_pred = np.zeros((num_draws, data_time_cal_pred.size))

for i in range(num_draws):
    draw_β = (trace_calibration["β"])[i]
    draw_γ = (trace_calibration["γ"])[i]
    draw_I0 = (trace_calibration["I(0)"])[i]

    S0 = N - (draw_I0 + R0)
    ICs = (S0, draw_I0, R0)

    y_fit = model_solver(time_span, ICs, data_time_cal_pred, N, draw_β, draw_γ)
    susceptible_pred[i] = y_fit[0]
    infected_pred[i] = y_fit[1]
    removed_pred[i] = y_fit[2]

susceptible_min = np.percentile(susceptible_pred, percentile_cut, axis=0)
susceptible_max = np.percentile(susceptible_pred, 100 - percentile_cut, axis=0)

infected_min = np.percentile(infected_pred, percentile_cut, axis=0)
infected_max = np.percentile(infected_pred, 100 - percentile_cut, axis=0)

removed_min = np.percentile(removed_pred, percentile_cut, axis=0)
removed_max = np.percentile(removed_pred, 100 - percentile_cut, axis=0)
```

github.com/gustavolibotte/eamc-2021-bayesian-inference

```python
daily_date_cal_pred = np.concatenate((daily_date_cal, daily_date_pred))
daily_cases_cal_pred = np.concatenate((daily_cases_cal, daily_cases_pred))

# Criação da figura
fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=[8, 12], sharex=True)

# Curva da simulação calculada usando os parâmetros ótimos
ax1.plot(daily_date_cal_pred, susceptible_prediction, linestyle="-", markersize=2, color="C1")
ax2.plot(daily_date_cal_pred, infected_prediction, label="Simulação", linestyle="-", markersize=2, color="C2")
ax3.plot(daily_date_cal_pred, removed_prediction, linestyle="-", markersize=2, color="C3")

# Intervalo de credibilidade
ax1.fill_between(daily_date_cal_pred, susceptible_min, susceptible_max, alpha=0.2, color="C1")
ax2.fill_between(daily_date_cal_pred, infected_min, infected_max, alpha=0.2, color="C2")
ax3.fill_between(daily_date_cal_pred, removed_min, removed_max, alpha=0.2, color="C3")

# Dados reportados
ax2.plot(daily_date_cal, daily_cases_cal, label="Dados reportados", marker="s", linestyle="", markersize=2)
ax2.plot(daily_date_pred, daily_cases_pred, label="Dados reportados", marker="8", linestyle="", markersize=2)

# Definição do formato da data no eixo horizontal
ax3.xaxis.set_major_formatter(mdates.DateFormatter('%d/%m/%y'))
ax3.xaxis.set_major_locator(mdates.DayLocator(interval=5))

# Nomes dos eixos e legenda
ax3.set_xlabel("Data")
ax1.set_ylabel("Suscetíveis");
ax2.set_ylabel("Infectados");
ax3.set_ylabel("Removidos")
```

github.com/gustavolibotte/eamc-2021-bayesian-inference





github.com/gustavolibotte/eamc-2021-bayesian-inference

MUITO OBRIGADO!

Dúvidas?

Contato:

Gustavo Libotte
glibotte@lncc.br