Answer 1

We know from Bayes Theorem, where y is the data and θ represents the parameters of the model that will define the data y

$$P(\theta|y) = (P(y|\theta).P(\theta)) / P(y)$$

=> $P(\theta|y) \propto P(y|\theta).P(\theta)$ => $P(\theta|y) \propto L(\theta).P(\theta)$ => Posterior ∝ Liklihood . Prior

From the question we know,

Prior is defined by $P(\theta) = Beta(\alpha,\beta) = B(\alpha,\beta)^{-1} . \theta^{\alpha-1} . (1-\theta)^{\beta-1}$

Likelihood function is defined by = $Bin(ni,\theta) = ({}^{ni}C_{yi}) . \theta^{yi} . (1-\theta)^{ni-yi}$

Therefore, the posterior is given by

$$\prod(({}^{ni}C_{yi}) . \theta^{yi} . (1-\theta)^{ni-yi}).(B(\alpha,\beta)^{-1} . \theta^{\alpha-1} . (1-\theta)^{\beta-1})$$

=> $P(\theta|y1,,,,,,yn) \propto (\theta^{\sum yi} . (1-\theta)^{\sum(ni-yi)}) \times (\theta^{\alpha-1} . (1-\theta)^{\beta-1})$

=> $P(\theta|y1,,,,,,yn) \propto (\theta^{\alpha + \sum yi -1} . (1-\theta)^{\beta + \sum(ni-yi) -1})$

=> $P(\theta|y1,,,,,,yn) \propto (\theta^{\alpha + \sum yi -1} . (1-\theta)^{\beta + \sum(ni-yi) -1})$

=> $P(\theta|y1,,,,,,yn) \propto$ **$Beta(\alpha + \sum yi, \beta + \sum(ni-yi))$** = Posterior Distribution

Answer 2

We know from Bayes Theorem, where y is the data and θ represents the parameters of the model that will define the data y

$$P(\theta|y) = (P(y|\theta).P(\theta)) / P(y)$$

=> $P(\theta|y) \propto P(y|\theta).P(\theta)$ => $P(\theta|y) \propto L(\theta).P(\theta)$ => Posterior ∝ Liklihood . Prior

From the question we know,

Prior is defined by $P(\theta) = Beta(\alpha,\beta) = B(\alpha,\beta)^{-1} . \theta^{\alpha-1} . (1-\theta)^{\beta-1}$

Likelihood function is defined by = $Geom(\theta) = (1-\theta)^{y-1} . \theta$

Therefore, the posterior is given by

$$\prod((1-\theta)^{yi-1} . \theta).(B(\alpha,\beta)^{-1} . \theta^{\alpha-1} . (1-\theta)^{\beta-1})$$

=> $P(\theta|y1,,,,,,yn) \propto (\theta^{n} . (1-\theta)^{\sum(yi-1)}) \times (\theta^{\alpha-1} . (1-\theta)^{\beta-1})$

=> $P(\theta|y1,,,,,,yn) \propto (\theta^{\alpha+n-1} . (1-\theta)^{\beta + \sum(yi-1) -1})$

=> $P(\theta|y1,,,,,,yn) \propto$ **Beta(α+n, β + ∑(yi-1))** = Posterior Distribution

Answer 3

a) From Q1, we know that Posterior Distribution

$$P(\theta|y1,,,,,,yn) \propto Beta(\alpha + \sum yi, \beta + \sum(ni-yi))$$

Treatment	Total (n)	Successes (y)
Placebo only	160	30
Nicotine Patch	244	52
Zyban	244	85
Zyban and Nicotine Patch	245	95

Using the prior mentioned **θ∼Beta(1,1) where α = 1, β = 1**
The Posterior Distribution, becomes **P(θ|y) ∝ Beta(y + 1, n – y + 1)**

For Placebo, $P(\theta_1|y_1) \propto Beta(30 + 1, 160 – 30 + 1)$ = **Beta(31, 131)**
For Nicotine Patch, $P(\theta_2|y_2) \propto Beta(52 + 1, 244 – 52 + 1)$ = **Beta(53, 193)**
For Zyban, $P(\theta_3|y_3) \propto Beta(85 + 1, 244 – 85 + 1)$ = **Beta(86, 160)**
For Zyban and Nicotine Patch, $P(\theta_4|y_4) \propto Beta(95 + 1, 245 – 95 + 1)$ = **Beta(96, 151)**

b) When calculating the credible interval of 95% = 0.95
The alpha will be set to 1 – 95% = 1 – 0.95 = 0.05
For calculating the 95% credible interval, define the [lower limit, upper limit]
[lower limit, upper limit] = [alpha/2, 1 - alpha/2] = [0.025, 0.975]
To find the credible interval, we have used scipy **(Code attached in APPENDIX)**

group	Beta parameters	lower	upper
placebo	(31, 131)	0.134734	0.255194
nicotine	(53, 193)	0.166444	0.268831
zyban	(86, 160)	0.291345	0.410159
zyban & nicotine	(96, 151)	0.328908	0.450128

c)

From the calculations, we can see that Bayesian credible interval and confidence interval lie close to each other. For the groups of placebo only and nicotine only, the credible interval is slightly bigger in scale. For zyban only group, the credible interval and confidence interval are same.

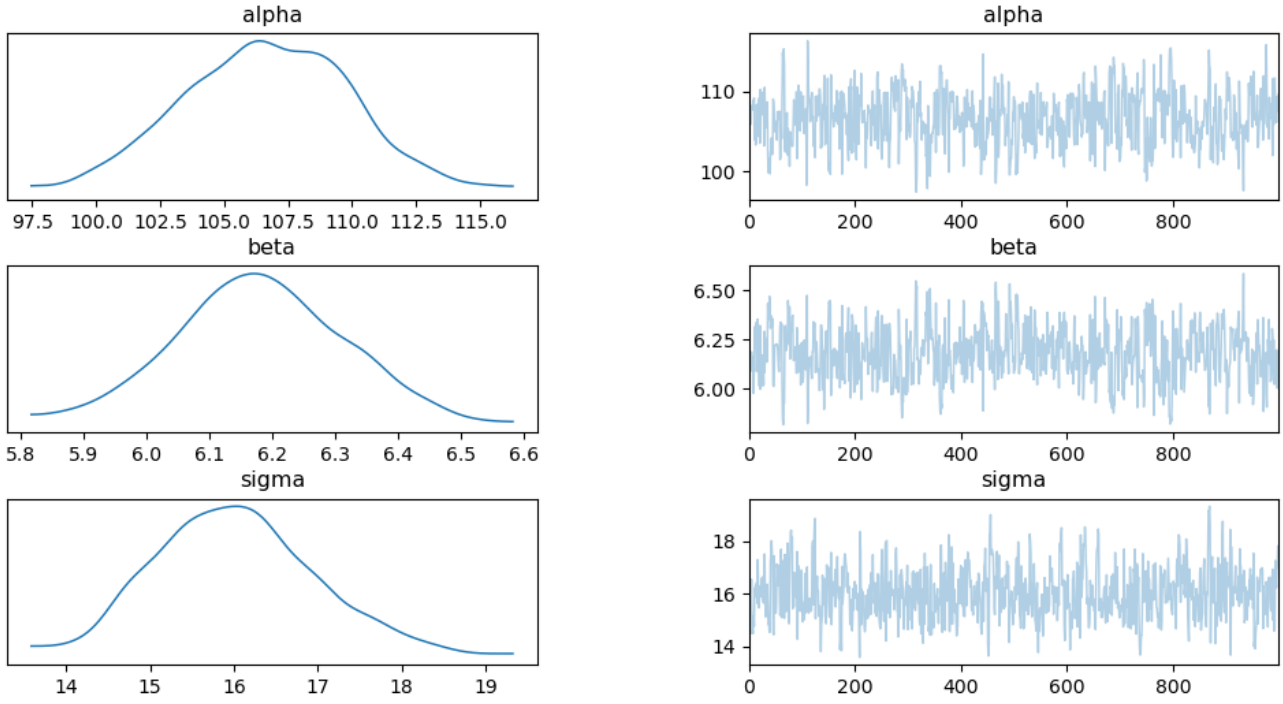
For the group of Zyban and nicotine patch the credible interval is wider than the confidence interval. When the credible interval is wider than the confidence interval, it means that the Bayesian method is incorporating more uncertainty because it includes prior information. This means the model is less sure of the estimated value. Hence it is presenting a wider range to cover all outcomes. This happens due to the data not being sure of the output value but we know that credible interval presents the true value given the data and the presented priors.

Hence, I agree with authors assumption that zyban only has the best success.

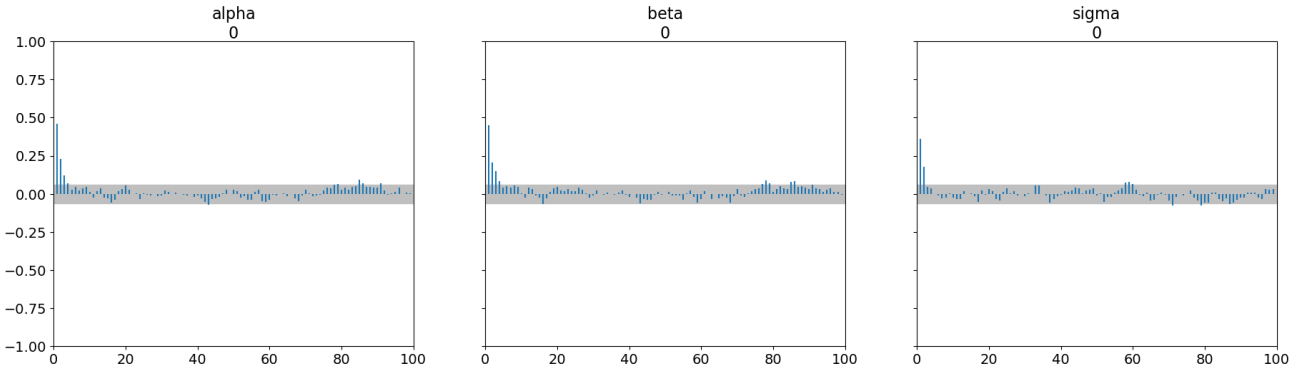
Answer 4

a) **(Code attached in APPENDIX)**
b) **(Code attached in APPENDIX)**
c) **(Code attached in APPENDIX)**

Trace Plot



Autocorrelation Plot



The trace plot show variability in the 3 parameters and also show that the parameter space is covered well. The existence of a unimodal structure also behaves well for the Posterior distribution. For the three parameters the graph shows no patterns as well as the fluctuations lie close to their respective means.

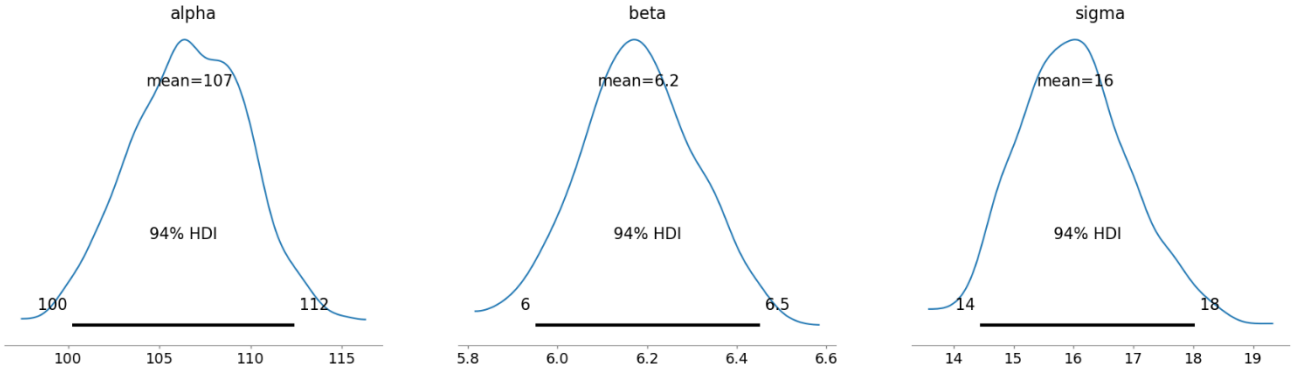
The bell-shaped curve shows that the 3 parameters have converged on a normal distribution as required for the Posterior distribution. This is good for convergence.

The autocorrelation plot shows the independence of the samples from each other. In case of autocorrelation plots the objective is that the value should fall close to zero as fast as possible. For all the 3 parameters the autocorrelation is high at the beginning but decreases near to 0. The existence of high autocorrelation means that sampled data is not as independent. This negatively affects the convergence.

Summary Statistics

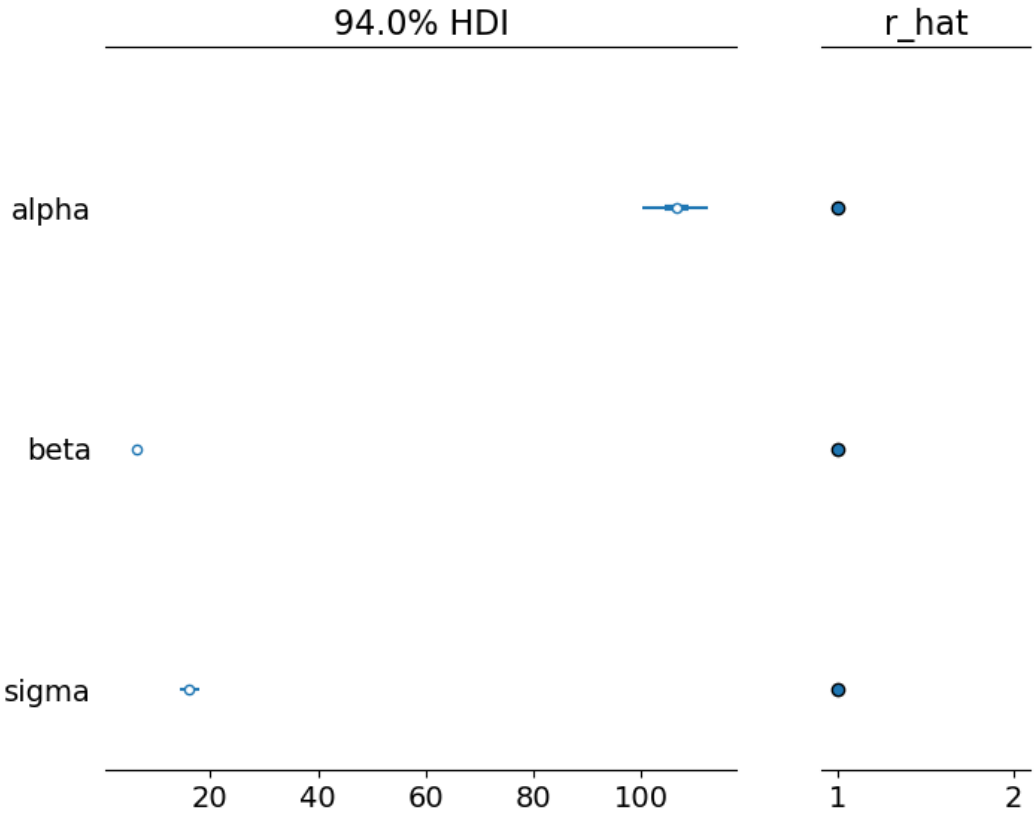
	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
alpha	106.638	3.257	100.237	112.405	0.181	0.128	323.0	370.0	NaN
beta	6.179	0.134	5.951	6.452	0.008	0.005	311.0	346.0	NaN
sigma	16.032	0.973	14.440	18.023	0.046	0.033	473.0	369.0	NaN

Density Plots



d) **(Code attached in APPENDIX)**

Gelman-Rubin diagnostic plot



From the Graph we can see that the value for r_hat calculated in Gelman-Rubin Plot –

alpha 1.001
beta 1.002
sigma 1.0

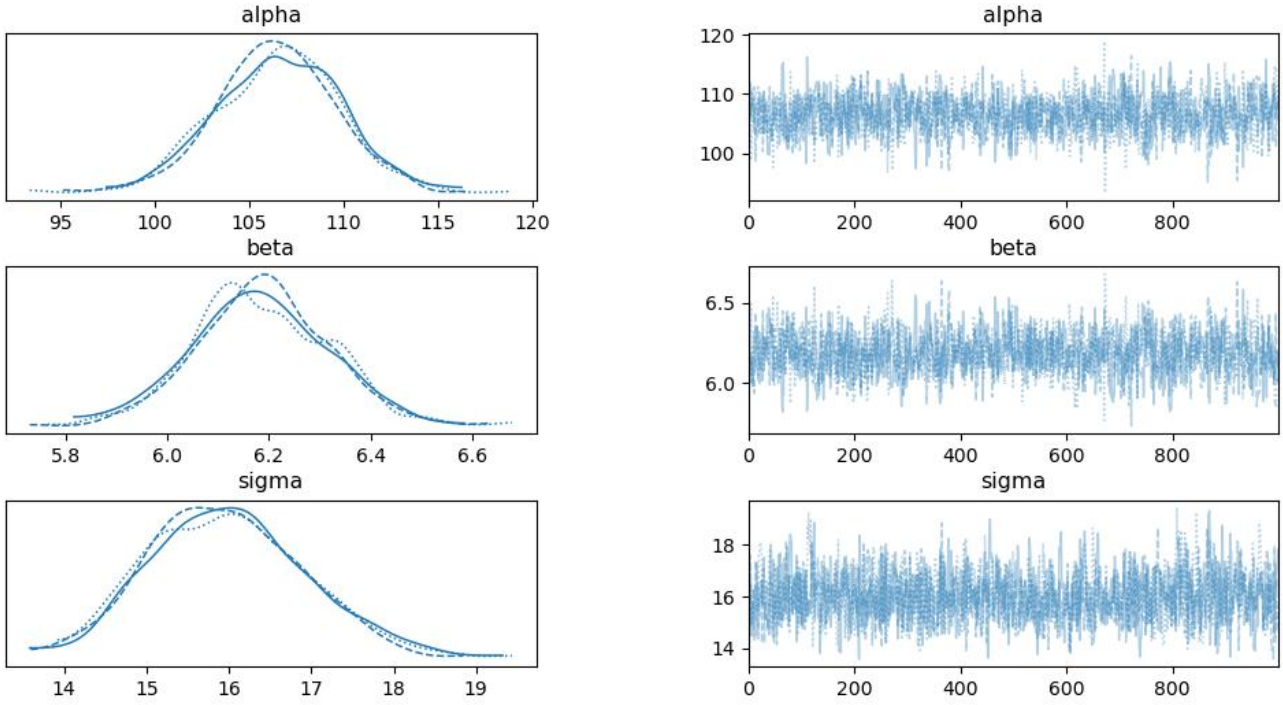
These values suggest good convergence as value should be close to 1 and less than 1.1. This suggests that the chains have converged to a stable distribution. The HDI value also suggests that parameter estimates are also reliable and model is stable.

e) **(Code attached in APPENDIX)**

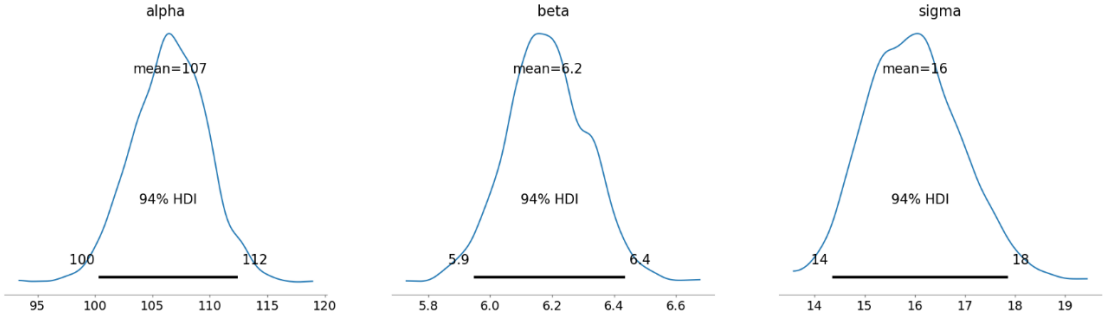
Summary

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
alpha	106.502	3.208	100.328	112.423	0.114	0.081	784.0	1204.0	1.0
beta	6.185	0.131	5.947	6.437	0.005	0.003	801.0	1070.0	1.0
sigma	15.992	0.951	14.352	17.852	0.026	0.019	1363.0	1122.0	1.0

Trace Plot



Density Plot



In Trace Plot, the normal bell-shaped curve shows that the 3 parameters have converged on a normal distribution as required for the Posterior distribution. The existence of a unimodal structure also behaves well for the Posterior distribution. This is good for convergence.
The parameter space also seems to be well covered. For the three parameters, the graph shows no patterns as well as the fluctuations lie close to their respective means meaning less variability.
The total of 3 chains and after their respective iterations seems to have converged.

Importing Libraries

In [1]:

```
import pymc3 as pm
import arviz as az
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import beta

import warnings
warnings.filterwarnings("ignore")
```

```
WARN: Could not locate executable g77
WARN: Could not locate executable f77
WARN: Could not locate executable ifort
WARN: Could not locate executable ifl
WARN: Could not locate executable f90
WARN: Could not locate executable DF
WARN: Could not locate executable efl
WARN: Could not locate executable gfortran
WARN: Could not locate executable f95
WARN: Could not locate executable g95
WARN: Could not locate executable efort
WARN: Could not locate executable efc
WARN: Could not locate executable flang
WARN: don't know how to compile Fortran code on platform 'nt'
```

```
WARNING (theano.configdefaults): g++ not available, if using conda: `conda install m2w64-
toolchain`
WARNING (theano.configdefaults): g++ not detected ! Theano will be unable to execute opti
mized C-implementations (for both CPU and GPU) and will default to Python implementations
. Performance will be severely degraded. To remove this warning, set Theano flags cxx to
an empty string.
WARNING (theano.tensor.blas): Using NumPy C-API based implementation for BLAS functions.
```

Question 3

In [2]:

```
df = pd.DataFrame([{"group": "placebo", "parameters": (31, 131)},
                   {"group": "nicotine", "parameters": (53, 193)},
                   {"group": "zyban", "parameters": (86, 160)},
                   {"group": "zyban & nicotine", "parameters": (96, 151)}])
```

In [3]:

```
alpha=0.05
for index, row in df.iterrows():
    df.loc[df["group"] == row["group"], "lower"] = beta.ppf(alpha/2, row["parameters"][0
], row["parameters"][1])
    df.loc[df["group"] == row["group"], "upper"] = beta.ppf(1 - (alpha/2), row["paramete
rs"][0], row["parameters"][1])
df
```

Out[3]:

	group	parameters	lower	upper
0	placebo	(31, 131)	0.134734	0.255194
1	nicotine	(53, 193)	0.166444	0.268831
2	zyban	(86, 160)	0.291345	0.410159
3	zyban & nicotine	(96, 151)	0.328908	0.450128

group parameters lower upper

Question 4

In [4]:

```
df = pd.read_csv("Rats.csv", header=None, names=[8, 15, 22, 29, 36])
df.index = range(1, df.shape[0]+1)

x = []
y = []
for column in df.columns:
    for value in df[column]:
        x.append(column)
        y.append(value)
```

a) and b)

In [5]:

```
with pm.Model() as model:
    alpha = pm.Normal("alpha", mu=0, sigma=1000)
    beta = pm.Normal("beta", mu=0, sigma=1000)
    sigma = pm.Gamma("sigma", alpha=0.001, beta=0.001)

    mu = alpha + (beta * x)
    observations = pm.Normal("observations", mu=mu, sigma=sigma, observed=y)
    trace = pm.sample(draws=1000, chains=3, random_seed=42, return_inferencedata=True)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (3 chains in 4 jobs)
NUTS: [sigma, beta, alpha]
```

100.00% [6000/6000 14:24<00:00 Sampling 3 chains, 0 divergences]

```
Sampling 3 chains for 1_000 tune and 1_000 draw iterations (3_000 + 3_000 draws total) to
ok 878 seconds.
The acceptance probability does not match the target. It is 0.9031790924798925, but shoul
d be close to 0.8. Try to increase the number of tuning steps.
The acceptance probability does not match the target. It is 0.899352879507454, but should
be close to 0.8. Try to increase the number of tuning steps.
```

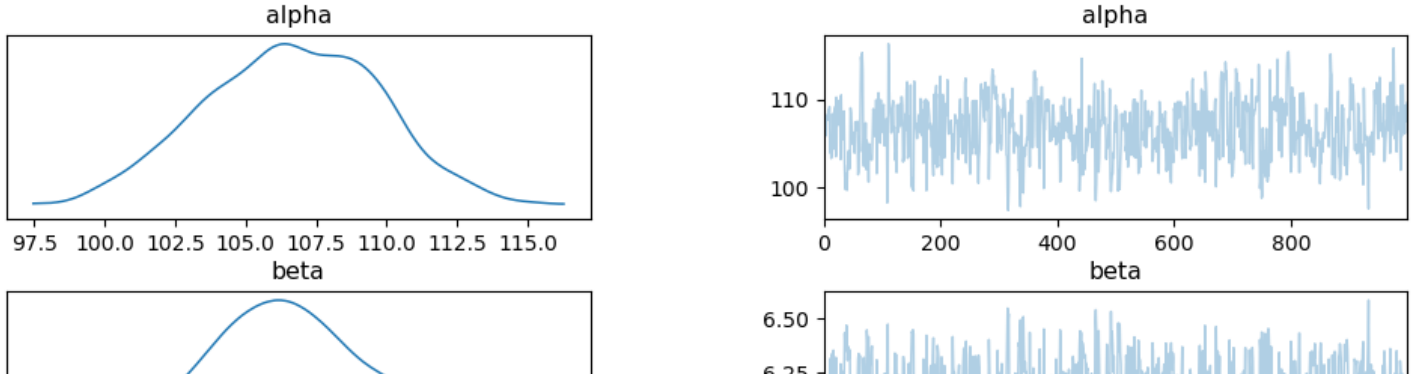
c)

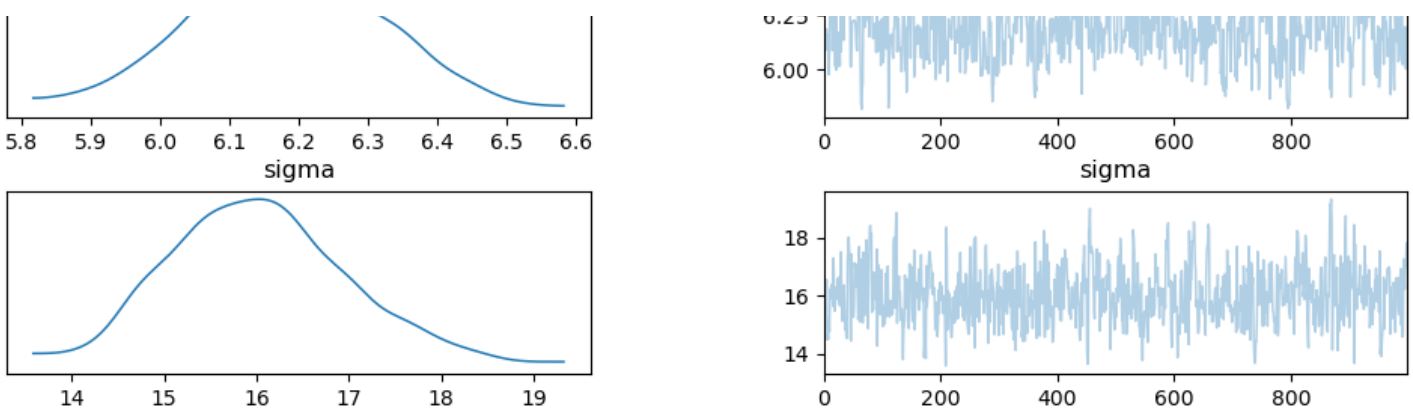
In [6]:

```
first_chain=trace.sel(chain=[0], drop=False)
```

In [7]:

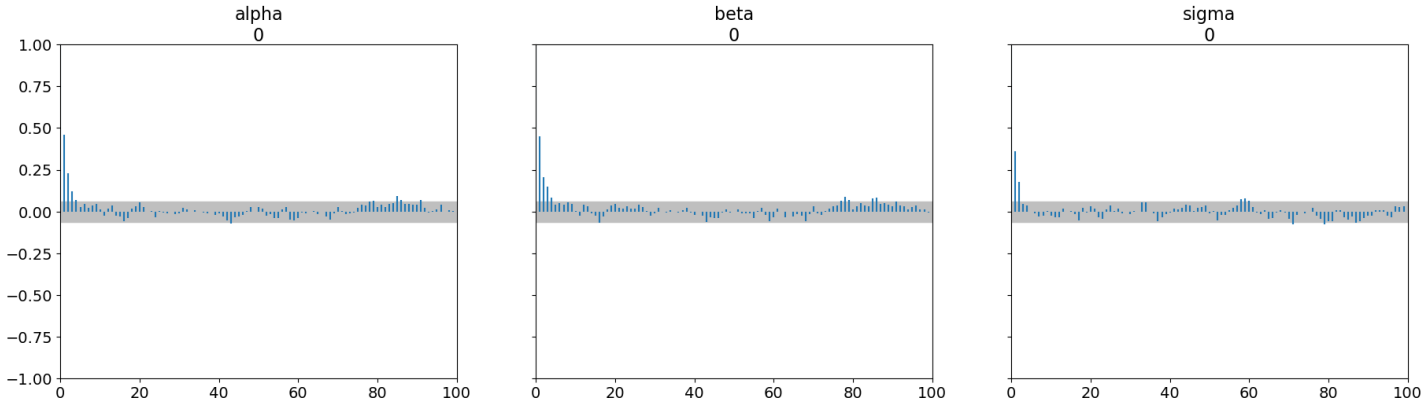
```
az.plot_trace(first_chain)
plt.subplots_adjust(hspace=0.4, wspace=0.4)
plt.show()
```





In [8]:

```
az.plot_autocorr(first_chain)
plt.show()
```



In [9]:

```
az.summary(first_chain)
```

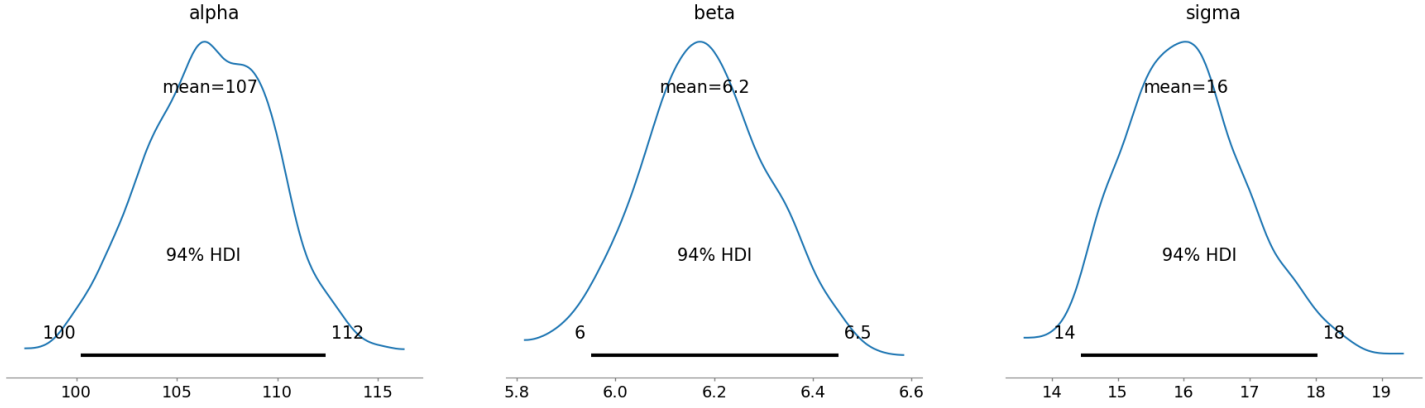
```
arviz - WARNING - Shape validation failed: input_shape: (1, 1000), minimum_shape: (chains=2, draws=4)
```

Out[9]:

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
alpha	106.638	3.257	100.237	112.405	0.181	0.128	323.0	370.0	NaN
beta	6.179	0.134	5.951	6.452	0.008	0.005	311.0	346.0	NaN
sigma	16.032	0.973	14.440	18.023	0.046	0.033	473.0	369.0	NaN

In [10]:

```
az.plot_posterior(first_chain)
plt.show()
```



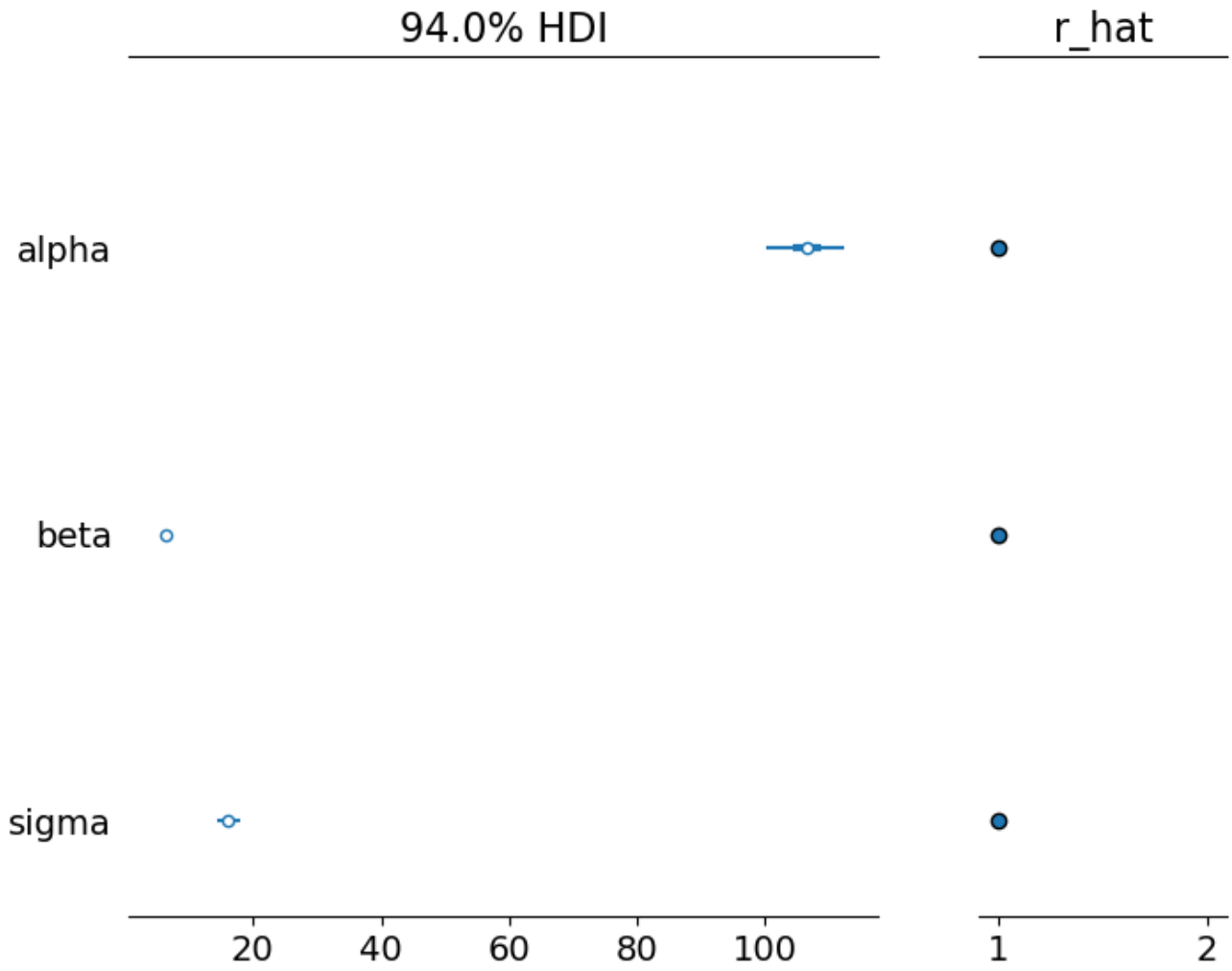
d)

In [11]:

```
gelman_rubin = az.rhat(trace)
print(gelman_rubin)

az.plot_forest(trace, r_hat=True, combined=True)
plt.show()
```

```
<xarray.Dataset>
Dimensions:  ()
Data variables:
    alpha    float64 1.001
    beta     float64 1.002
    sigma    float64 1.0
```



e)

In [12]:

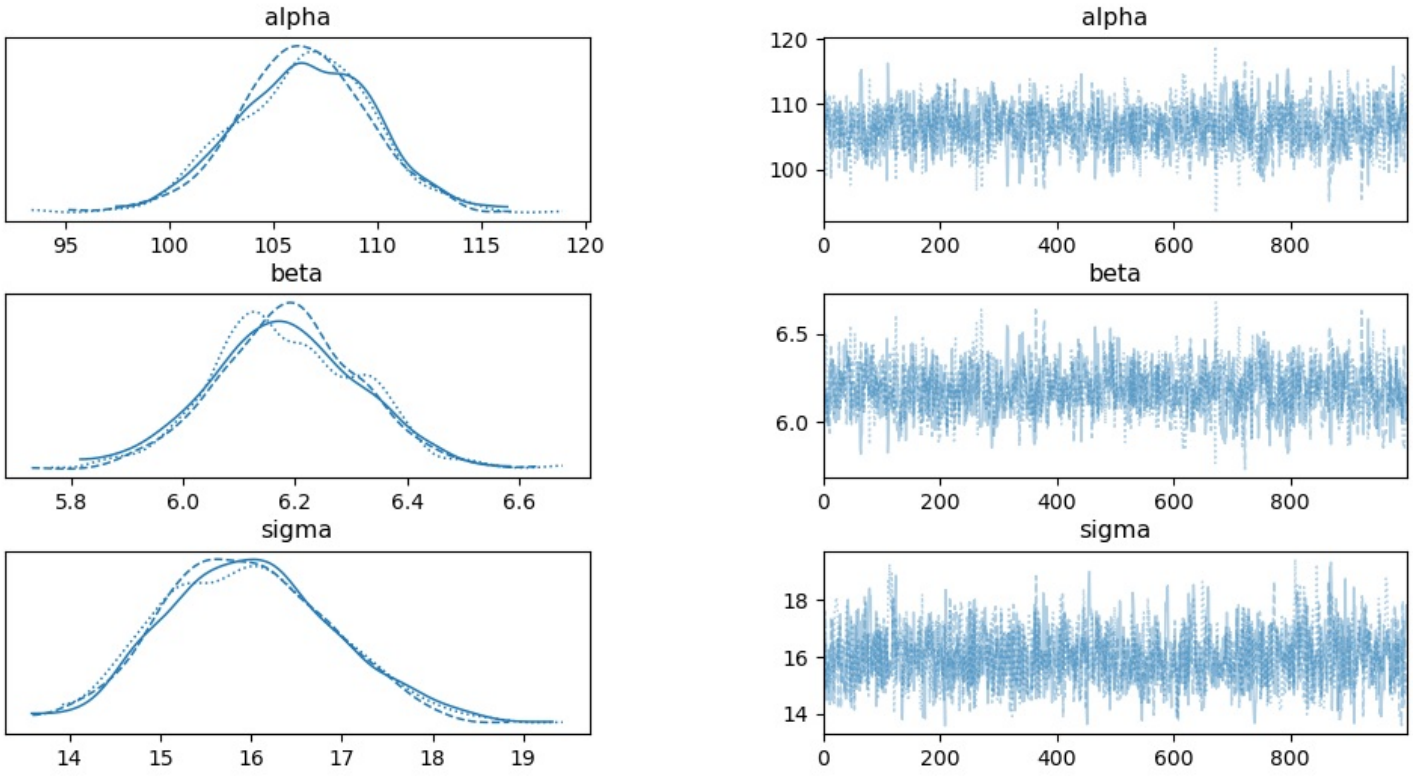
```
az.summary(trace)
```

Out[12]:

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
alpha	106.502	3.208	100.328	112.423	0.114	0.081	784.0	1204.0	1.0
beta	6.185	0.131	5.947	6.437	0.005	0.003	801.0	1070.0	1.0
sigma	15.992	0.951	14.352	17.852	0.026	0.019	1363.0	1122.0	1.0

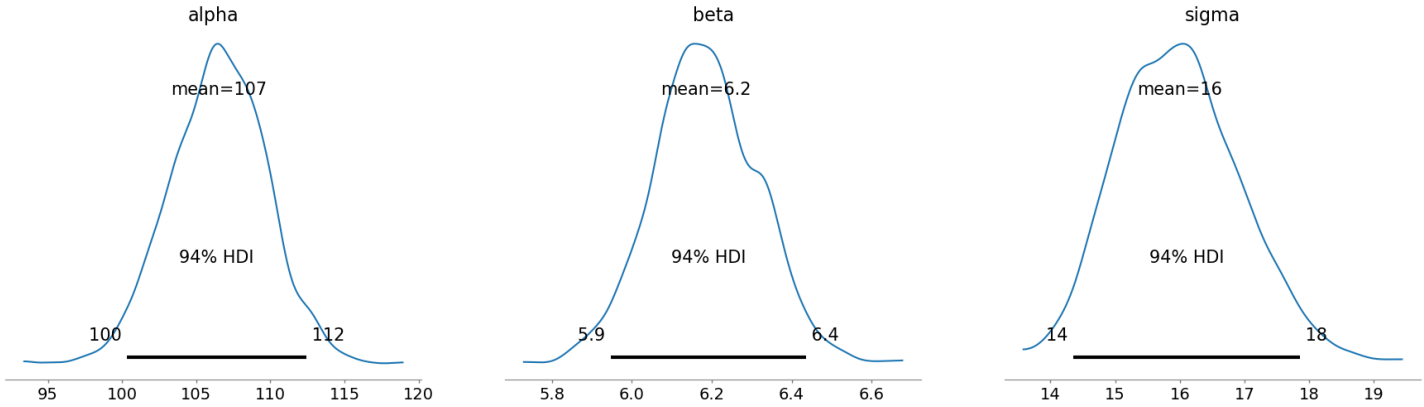
In [13]:

```
az.plot_trace(trace)
plt.subplots_adjust(hspace=0.4, wspace=0.4)
plt.show()
```



In [14]:

```
az.plot_posterior(trace)
plt.show()
```



In [ ]: