

# Model Diagnostic

The BI class can compute model diagnostics for a given model.

Lets consider the following model for a linear regression:

$$Y_i \sim \text{Normal}(\alpha + \beta X_i, \sigma)$$

$$\alpha \sim \text{Normal}(0, 1)$$

$$\beta \sim \text{Normal}(0, 1)$$

$$\sigma \sim \text{Uniform}(0, 50)$$

```
from BI import bi
import jax.numpy as jnp
# setup platform-----
m = bi(platform='cpu')

# import data -----
m.data('Howell1.csv', sep=';')
m.df = m.df[m.df.age > 18]
m.scale(data=['weight'])

# define model -----
def model(weight, height):
    a = m.dist.normal( 178, 20, name = 'a')
    b = m.dist.log_normal( 0, 1, name = 'b')
    s = m.dist.uniform( 0, 50, name = 's')
    m.dist.normal(a + b * weight , s, obs=height, shape=(weight.shape[0],))
```

```
# Run sampler -----
m.fit(model, num_samples=500,num_chains=4)
m.summary()
```

```
jax.local_device_count 16
```

```
0%|          | 0/1000 [00:00<?, ?it/s]
```

```
0%|          | 0/1000 [00:00<?, ?it/s]
```

```
0%|          | 0/1000 [00:00<?, ?it/s]
```

```
0%|          | 0/1000 [00:00<?, ?it/s]
```

	mean	sd	hdi_5.5%	hdi_94.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
a	154.64	0.29	154.16	155.07	0.01	0.01	1711.30	1283.42	1.0
b	5.82	0.29	5.33	6.23	0.01	0.01	1999.81	1298.15	1.0
s	5.14	0.20	4.82	5.45	0.00	0.00	2088.50	1526.51	1.0

## List of all available diagnostics

For additional documentation check the [diagnostics API reference](#)

## Predictions from model based on specific data value

```
m.sample() # Predictions from model base on data in data_on_model
m.sample(data=dict(weight=jnp.array([0.4])), remove_obs=False)# Predictions from a given value
```

```
/home/sosa/work/BI/BI/Main/main.py:417: UserWarning:
```

```
Sample's batch dimension size 2000 is different from the provided 1 num_samples argument. De
```

```
{'x': Array([[159.34970727],  
            [163.37257718],  
            [158.07636766],  
            ...,  
            [162.42332523],  
            [147.89303169],  
            [161.40022364]]], dtype=float64)}
```

### Forest plot of estimated values

```
m.diag.forest()
```

Unable to display output for mime type(s): text/html

Unable to display output for mime type(s): text/html

### Density plots of the posterior distribution

```
m.diag.density()
```

Unable to display output for mime type(s): text/html

### Posterior distribution plots

```
m.diag.posterior()
```

Unable to display output for mime type(s): text/html

### Trace plots for MCMC chains

```
m.diag.plot_trace()
```

Unable to display output for mime type(s): text/html

## Pairwise plots of the posterior distribution

```
m.diag.pair()
```

Unable to display output for mime type(s): text/html

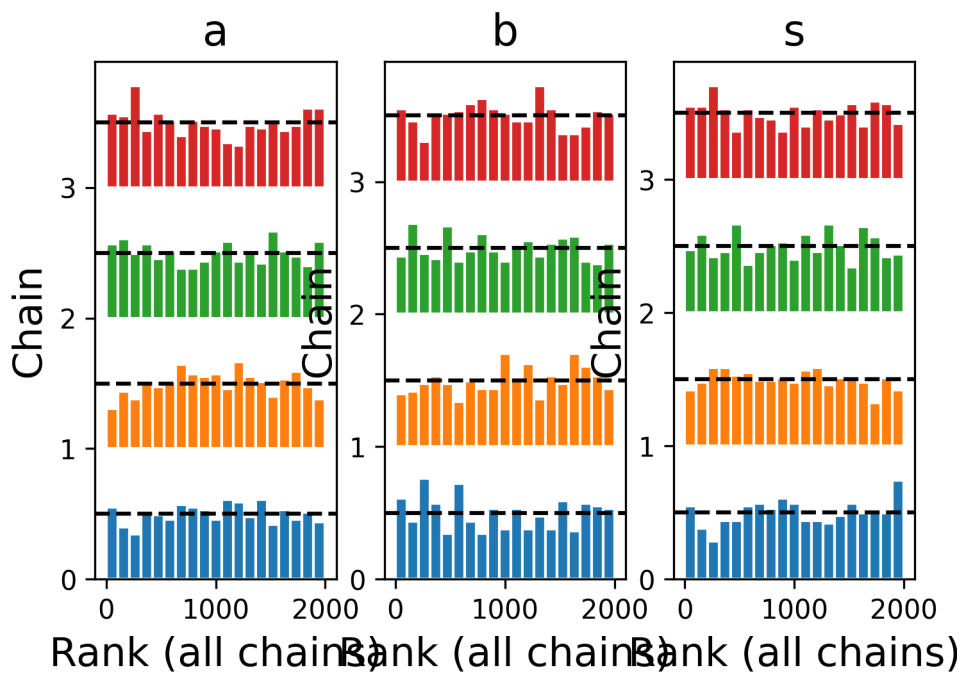
## Plot autocorrelation of MCMC chains

```
m.diag.autocor()
```

Unable to display output for mime type(s): text/html

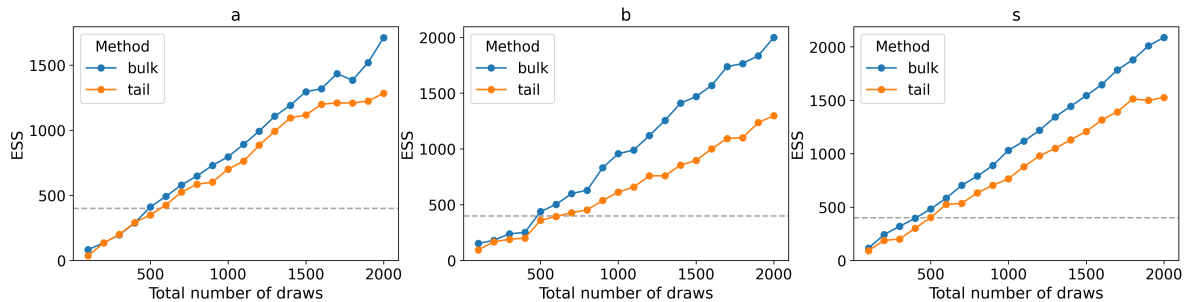
## Create rank plots for MCMC chains

```
m.diag.rank()
```



## Evolution of effective sample size across iterations

```
m.diag.plot_ess()
```



## Pareto-smoothed

```
m.diag.loo()
```

Computed from 2000 posterior samples and 346 observations log-likelihood matrix.

	Estimate	SE
elpd_loo	-1058.55	14.72
p_loo	3.26	-

-----

Pareto k diagnostic values:

		Count	Pct.
(-Inf, 0.70]	(good)	346	100.0%
(0.70, 1]	(bad)	0	0.0%
(1, Inf)	(very bad)	0	0.0%

## Widely applicable information criterion

```
m.diag.WAIC()
```

Computed from 2000 posterior samples and 346 observations log-likelihood matrix.

	Estimate	SE
elpd_waic	-1058.54	14.72
p_waic	3.26	-