

# Model diagnostics

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

Lets consider the following model for a linear regression:

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

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

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

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

```
import os
import sys
newPath = os.path.dirname(os.path.abspath(""))
if newPath not in sys.path:
    sys.path.append(newPath)
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'])
m.data_to_model(['weight', 'height'])

# 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=1000,num_chains=4)
m.summary()

```

jax.local\_device\_count 32

0%| 0/1500 [00:00<?, ?it/s]

0%| 0/1500 [00:00<?, ?it/s]

0%| 0/1500 [00:00<?, ?it/s]

0%| 0/1500 [00:00<?, ?it/s]

	mean	sd	hdi_5.5%	hdi_94.5%
a	154.65	0.28	154.23	155.13
b	5.81	0.28	5.35	6.23
s	5.14	0.20	4.83	5.46

## List of all available diagnostics

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

### Pareto-smoothed

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

Computed from 4000 posterior samples and 346 observations log-likelihood matrix.

	Estimate	SE
elpd_loo	-1058.44	14.67
p_loo	3.14	-
-----		

Pareto k diagnostic values:

		Count	Pct.
(-Inf, 0.5]	(good)	346	100.0%
(0.5, 0.7]	(ok)	0	0.0%
(0.7, 1]	(bad)	0	0.0%
(1, Inf)	(very bad)	0	0.0%

## Widely applicable information criterion

```
m.df.head()
```

	height	weight	age	male
0	151.765	0.430669	63.0	1
1	139.700	-1.326018	63.0	0
2	136.525	-2.041868	65.0	0
3	156.845	1.238745	41.0	1
4	145.415	-0.583818	51.0	0

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

Computed from 4000 posterior samples and 346 observations log-likelihood matrix.

	Estimate	SE
elpd_waic	-1058.43	14.67
p_waic	3.14	-

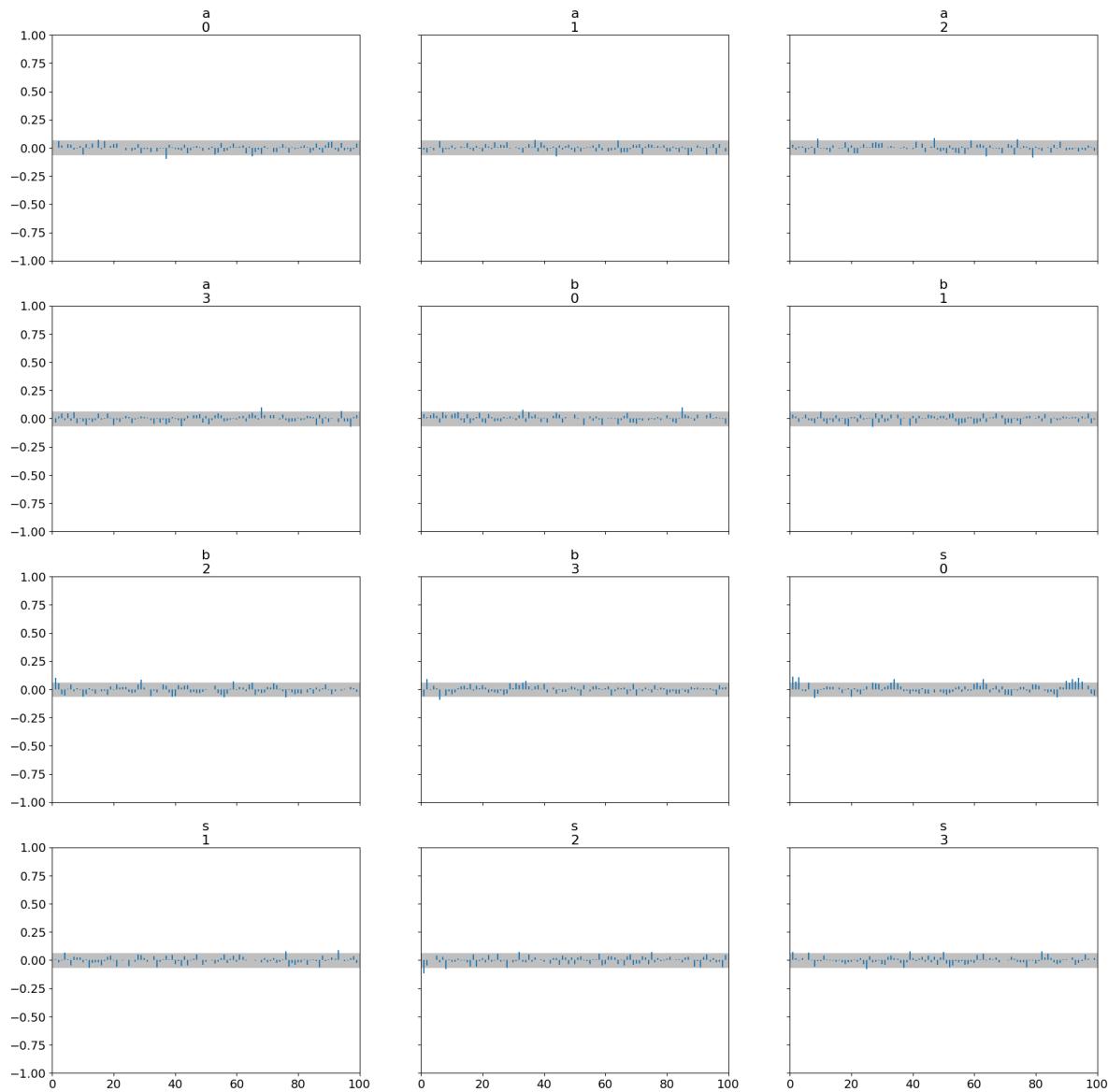
## Predictions from model base on specific data value

```
m.sample()  
m.sample(data=dict(weight=jnp.array([0.4])), remove_obs=False)
```

```
{'x': Array([149.77345571], dtype=float64)}
```

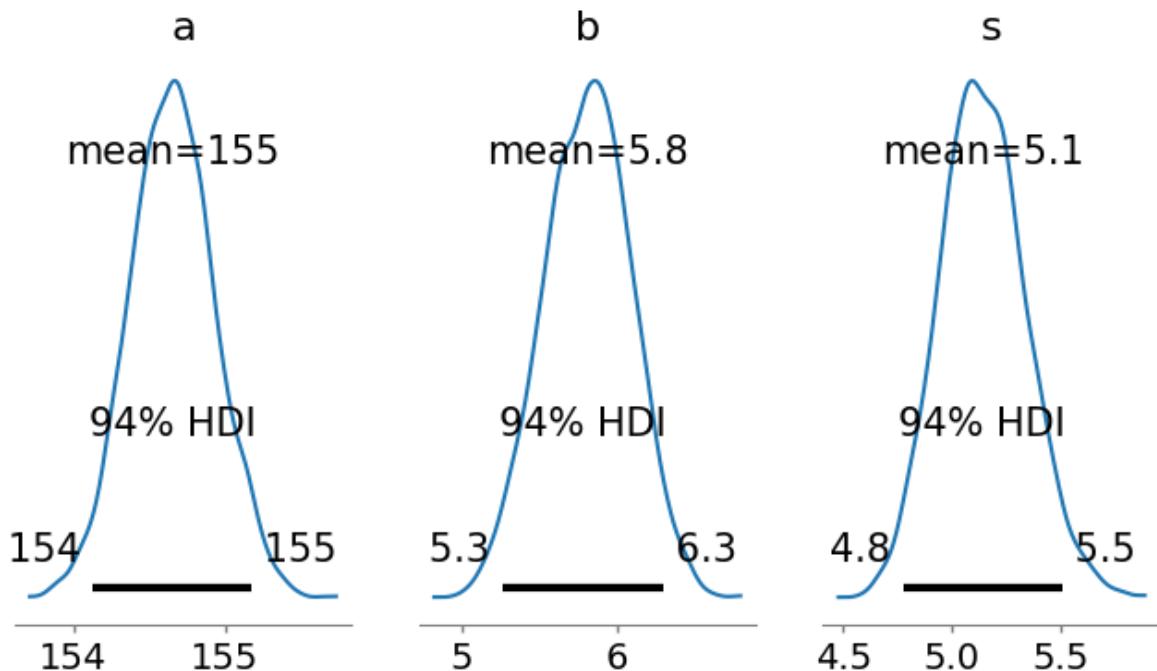
## Plot autocorrelation of the MCMC chains

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



## Posterior distribution plots

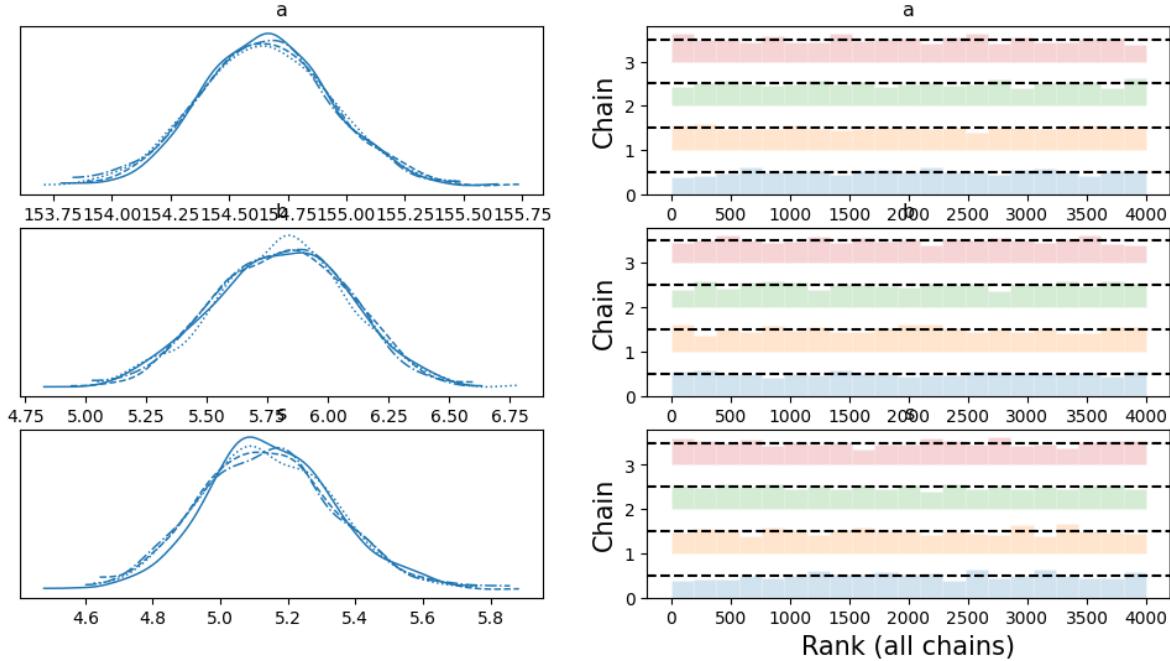
```
m.diag.posterior()
```



## Trace plots for MCMC chains

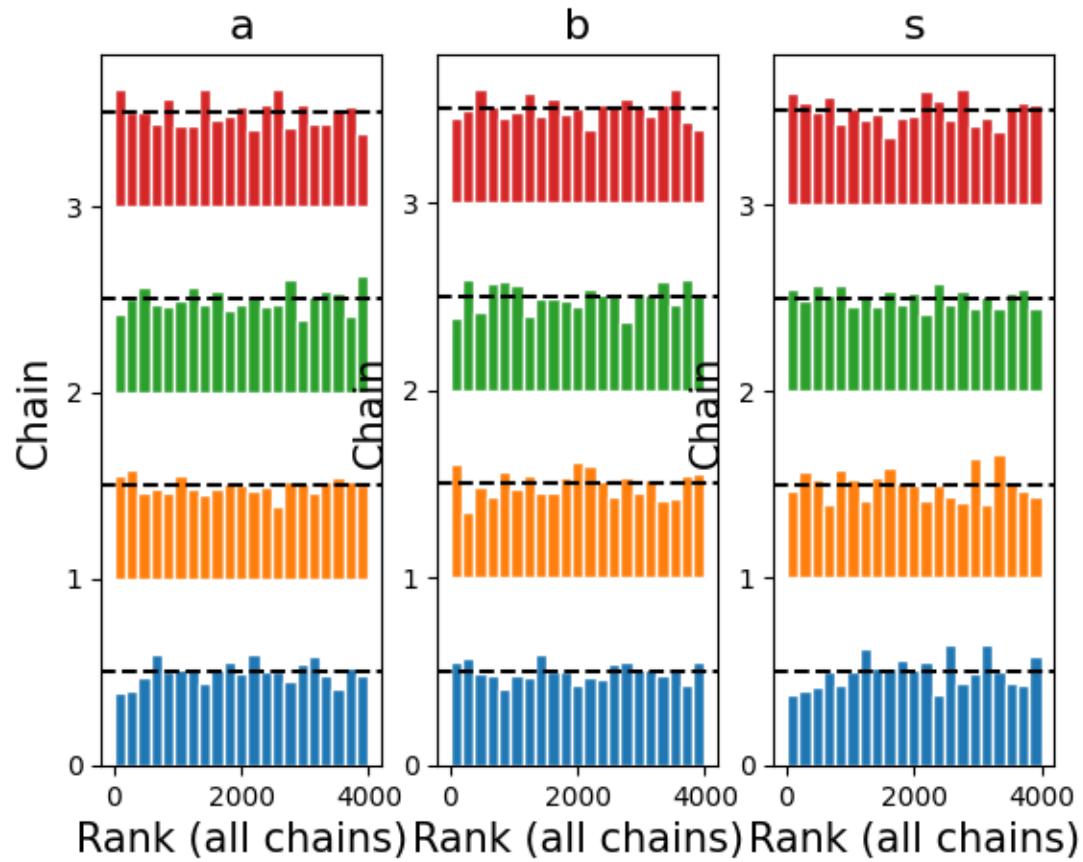
```
m.diag.plot_trace()
```

```
array([[<Axes: title={'center': 'a'}>,
       <Axes: title={'center': 'a'}, xlabel='Rank (all chains)', ylabel='Chain'>],
      [<Axes: title={'center': 'b'}>,
       <Axes: title={'center': 'b'}, xlabel='Rank (all chains)', ylabel='Chain'>],
      [<Axes: title={'center': 's'}>,
       <Axes: title={'center': 's'}, xlabel='Rank (all chains)', ylabel='Chain'>]],  
      dtype=object)
```



## Create rank plots for MCMC chains

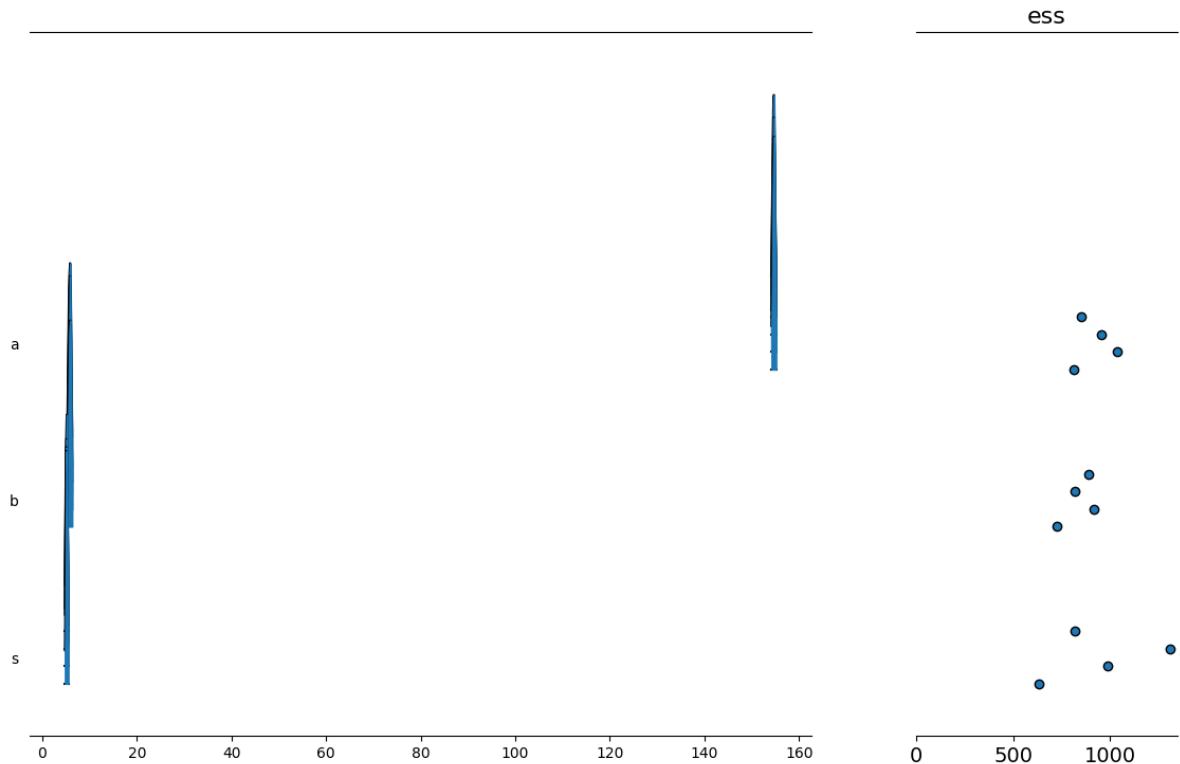
```
m.diag.rank()
```



**Forest plot of estimated values**

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

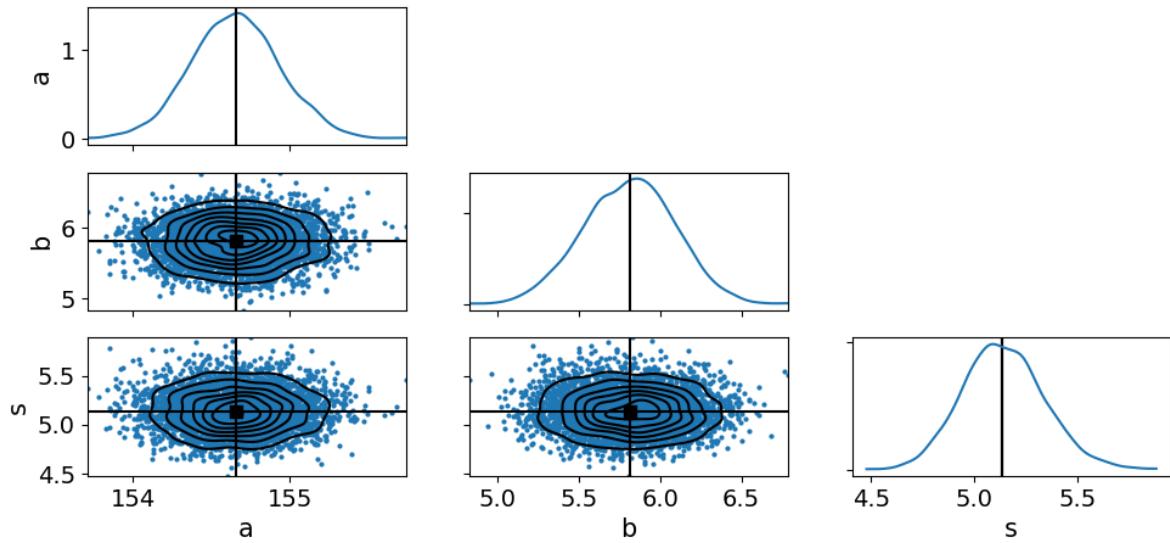
```
array([<Axes: >, <Axes: title={'center': 'ess'}>], dtype=object)
```



### Pairwise plots of the posterior distribution

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

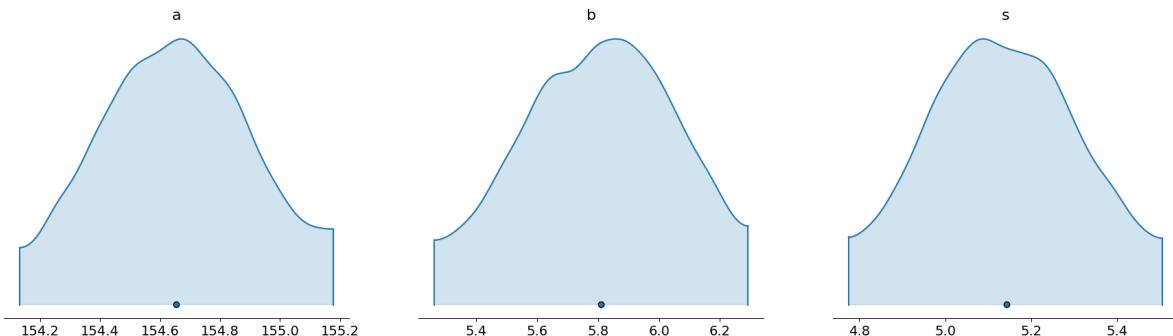
```
/home/sosa/.local/lib/python3.10/site-packages/xarray/core/utils.py:494: FutureWarning: The
  warnings.warn(
array([[<Axes: ylabel='a'>, <Axes: >, <Axes: >],
       [<Axes: ylabel='b'>, <Axes: >, <Axes: >],
       [<Axes: xlabel='a', ylabel='s'>, <Axes: xlabel='b'>,
        <Axes: xlabel='s'>]], dtype=object)
```



### Density plots of the posterior distribution

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

```
array([[[<Axes: title={'center': 'a'}>, <Axes: title={'center': 'b'}>,
       <Axes: title={'center': 's'}>]], dtype=object)
```



### Evolution of effective sample size across iterations

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

