Repeated measurement of IQ

```
import os

if "KERAS_BACKEND" not in os.environ:
    # set this to "torch", "tensorflow", or "jax"
    os.environ["KERAS_BACKEND"] = "jax"

import matplotlib.pyplot as plt
import numpy as np
import bayesflow as bf
```

```
INFO:bayesflow:Using backend 'jax'
```

In this example we build a model that estimates the IQ score of three participants ($i \in (1, 2, 3)$), each taking part in three IQ tests ($j \in (1, 2, 3)$). The Bayesian graphical model is as follows

$$\begin{aligned} \mu_i &\sim \text{Uniform}(0, 300) \\ \sigma &\sim \text{Uniform}(0, 100) \\ x_{ij} &\sim \text{Gaussian}(\mu_i, \sigma). \end{aligned} \tag{1}$$

We use rather uninformative priors both for μ and for σ. Follow this example and the book questions will lead to you to customizing this workflow with different custom priors!

Simulator

Again, we will use handcrafted summary statistics - in this case, we just calculate the mean and standard deviation for each person.

```
def prior():
    mu=np.random.uniform(low=0, high=300, size=3)
    sigma=np.random.uniform(low=0, high=100)
    return dict(mu=mu, sigma=sigma)

def summary(x):
    mean = np.mean(x, axis=-2)
    sd = np.std(x, axis=-2)
    return dict(mean=mean, sd=sd)

def likelihood(mu, sigma):
    x=np.random.normal(loc=mu, scale=sigma, size=(3, 3))
    return summary(x)

simulator=bf.make_simulator([prior, likelihood])
```

Approximator

```
adapter = (
    bf.Adapter()
    .constrain("mu", lower=0, upper=300)
    .constrain("sigma", lower=0, upper=100)
    .standardize(include=["mu", "sigma"])
    .concatenate(["mu", "sigma"], into="inference_variables")
    .concatenate(["mean", "sd"], into="inference_conditions")
    )
```

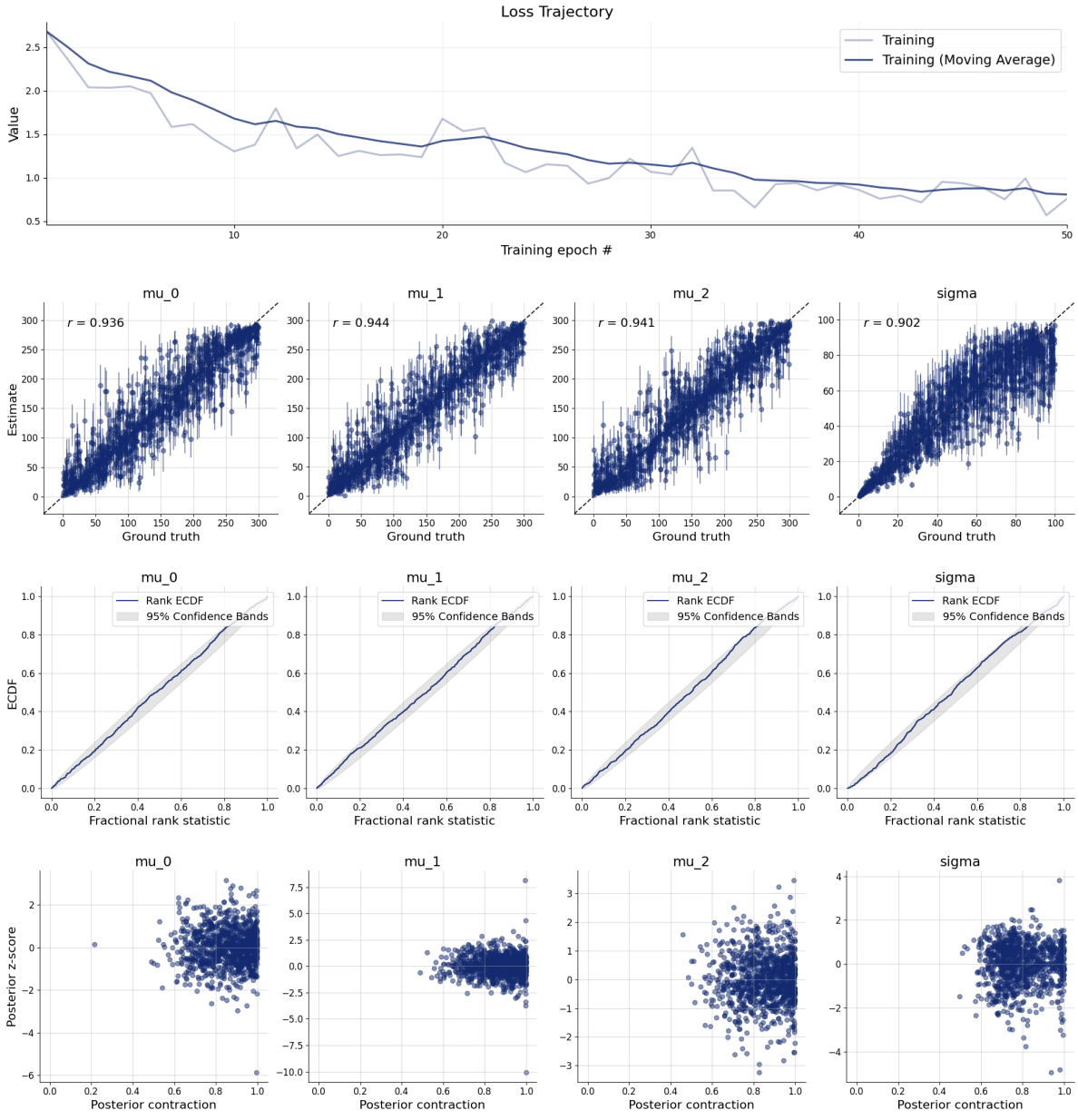
```
workflow=bf.BasicWorkflow(
    simulator=simulator,
    adapter=adapter,
    inference_network=bf.networks.CouplingFlow()
)
```

Training

```
history=workflow.fit_online(epochs=50, batch_size=512)
```

Validation

```
test_data=simulator.sample(1000)
figs=workflow.plot_default_diagnostics(test_data=test_data, num_samples=500)
```



Inference

```
x=np.array([
    [[ 90,  95, 100],
     [105, 110, 115],
     [150, 155, 160]]
    ])
x=x.swapaxes(1, 2)

inference_data=summary(x)
```

```
samples=workflow.sample(num_samples=2000, conditions=inference_data, split=True)
```

```
workflow.samples_to_data_frame(samples).describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
mu_0	2000.0	94.010524	4.384315	63.002457	91.229183	93.958546	96.775616	114.337432
mu_1	2000.0	109.797457	4.557817	85.536897	107.156925	109.449254	112.218850	137.207358
mu_2	2000.0	151.907019	12.026682	89.914878	145.126097	151.700725	157.969029	214.173878
sigma	2000.0	7.270084	2.392036	2.487769	5.650259	6.864182	8.381889	26.170801

Lee, M. D., & Wagenmakers, E.-J. (2013). *Bayesian Cognitive Modeling: A Practical Course*. Cambridge University Press.