Inferring a rate

```
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'
```

The first problem in this chapter involves inferring a binomial rate:

$$\begin{aligned} \theta &\sim \text{Beta}(1, 1) \\ k &\sim \text{Binomial}(\theta, n). \end{aligned} \tag{1}$$

Simulator

We will amortize over different sample sizes, so we will also draw randonly n during simulations.

```
def context():
    return dict(n=np.random.randint(1, 101))

def prior():
    return dict(theta=np.random.beta(a=1, b=1))

def likelihood(n, theta):
    return dict(k=np.random.binomial(n=n, p=theta))
```

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

Approximator

```
adapter = (
    bf.Adapter()
    .constrain("theta", lower=0, upper=1)
    .rename("theta", "inference_variables")
    .concatenate(["k", "n"], into="inference_conditions")
)
```

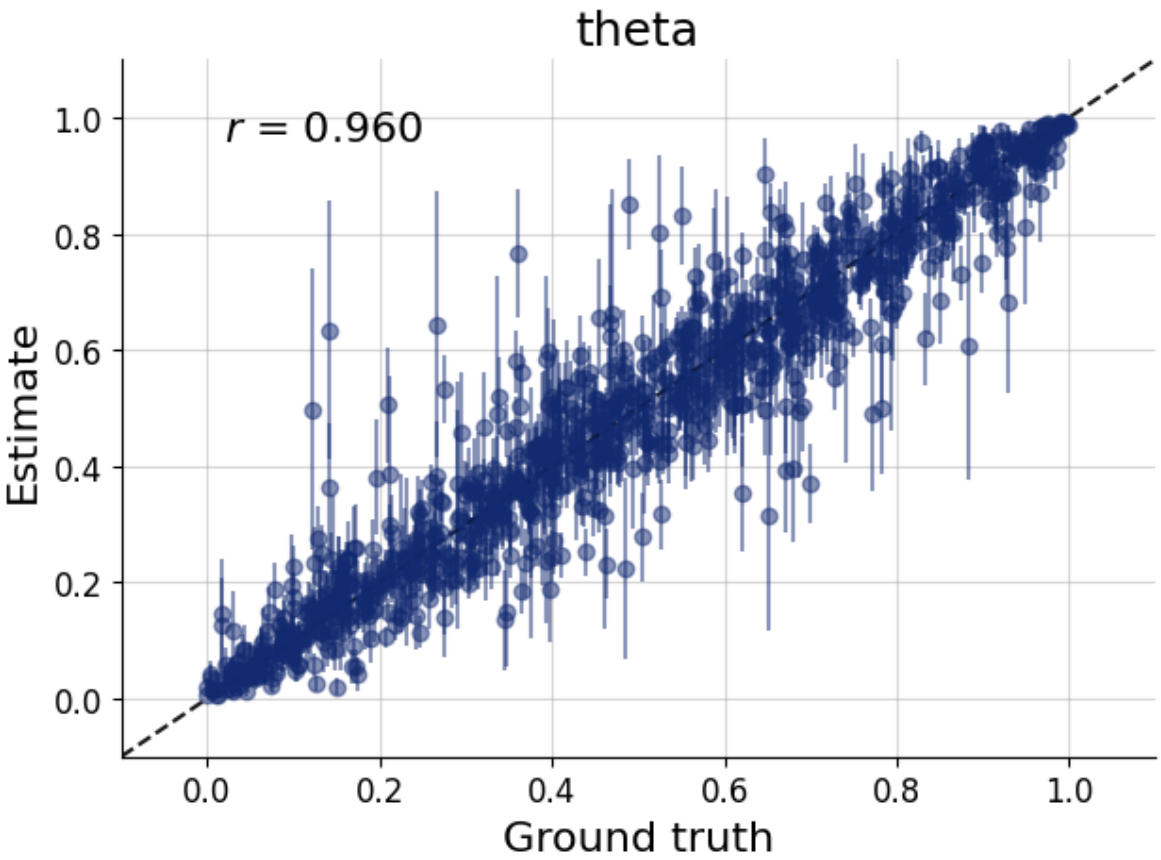
```
workflow=bf.BasicWorkflow(
    simulator=simulator,
    adapter=adapter,
    inference_network=bf.networks.CouplingFlow()
)
```

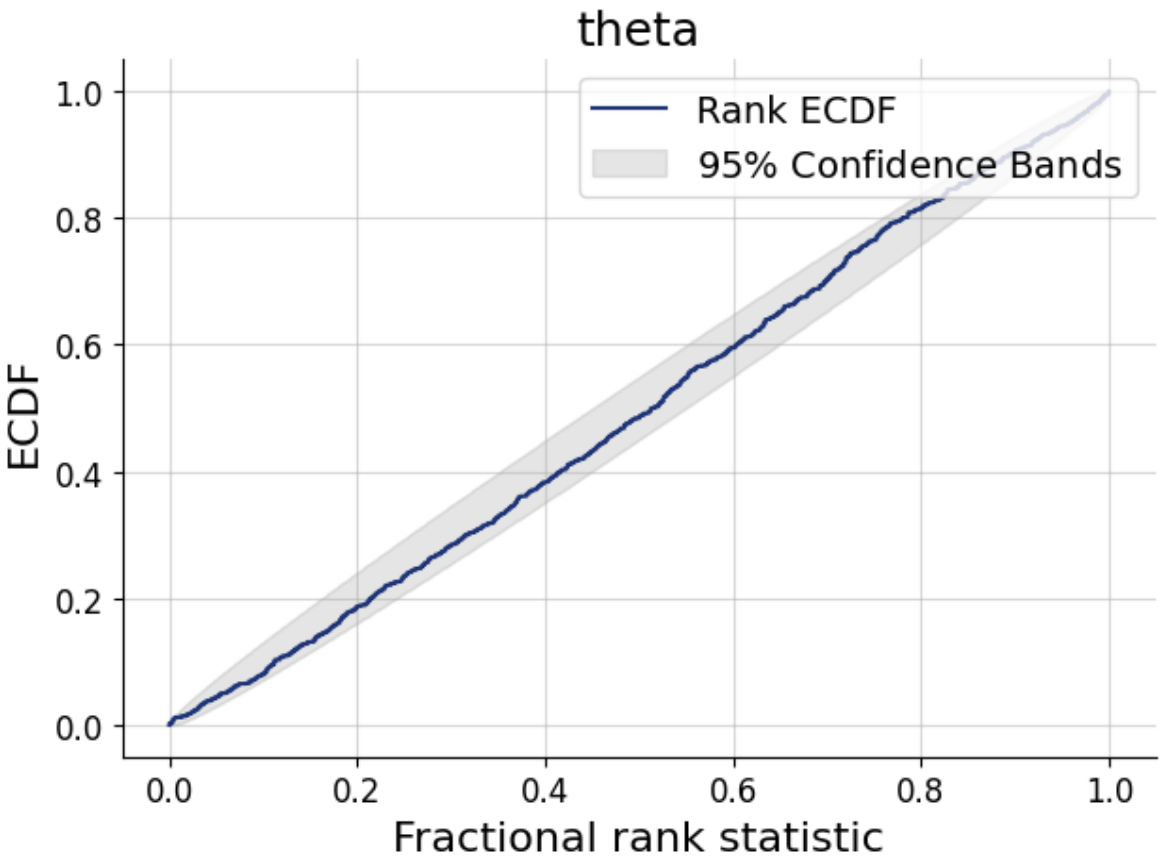
Training

```
history=workflow.fit_online(epochs=20, batch_size=512)
```

Validation

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


theta
r = 0.960
Estimate
Ground truth
0.0
0.2
0.4
0.6
0.8
1.0


theta
Rank ECDF
95% Confidence Bands
ECDF
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Fractional rank statistic

Inference

Now we obtain the approximation of the posterior distribution of θ given $k = 5$ and $n = 10$

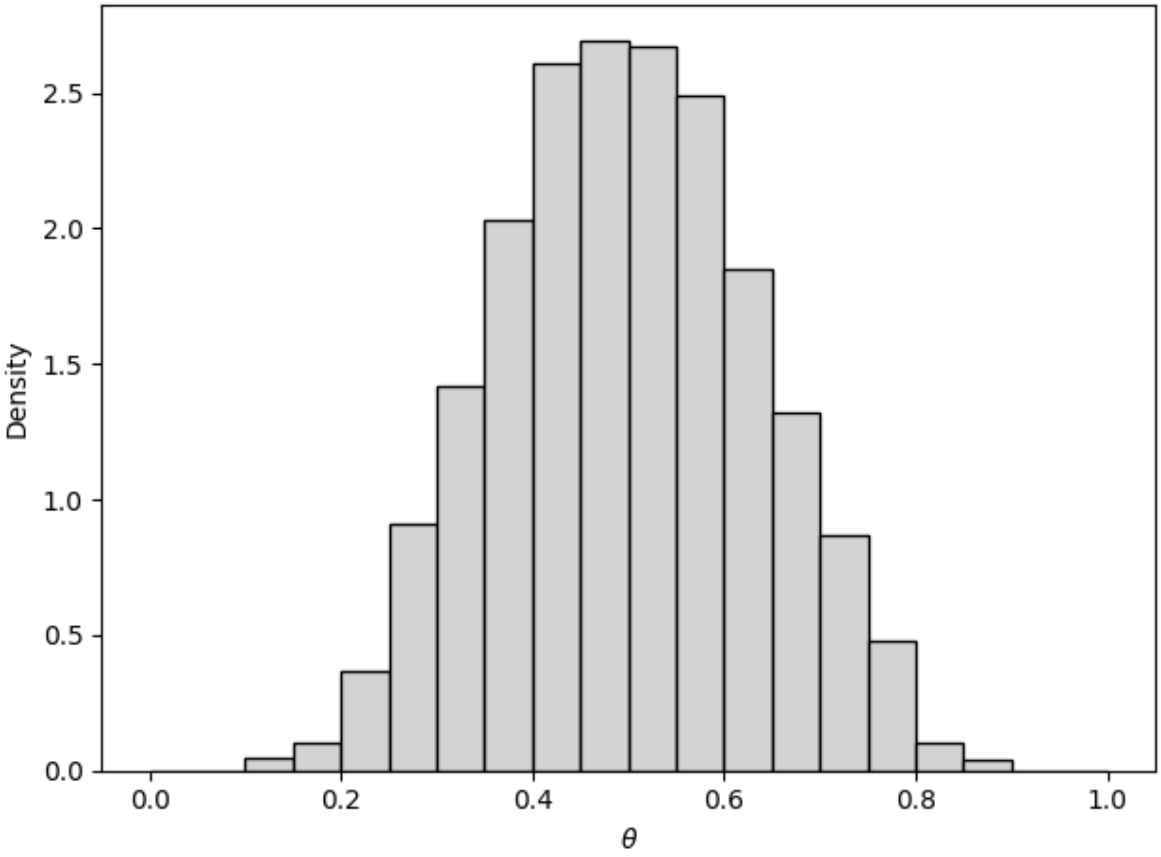
```
inference_data = dict(
    k = np.array([[5]]),
    n = np.array([[10]])
)
```

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

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

	theta
count	2000.000000
mean	0.498038
std	0.134124
min	0.125993
25%	0.401482
50%	0.496852
75%	0.591911
max	0.870781

```
plt.hist(samples["theta"].flatten(), density=True, color="lightgray", edgecolor="black", bins
plt.xlabel(r"$\theta$")
plt.ylabel("Density")
plt.tight_layout()
```



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