Censored data

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

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

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

A person attempts an exam that consists of 50 four-choice questions. In order to pass the exam, one must answer at least 30 correctly. If the person fails, they retake the exam. Once they pass the exam, we obtain only the following information: how many attempts they took, what was their score on the final, successfull attempt, and what was the range of their scores on the attempts that came before.

The goal is to estimate the probability θ of answering any of the test questions correctly.

Simulator

```
def prior():
    return dict(
        theta = np.random.uniform(low=0.25, high=1)
    )

def summary(scores):
    attempts = len(scores)
    if attempts == 1:
        return dict(
```

```
            attempts=attempts,
            score=scores[0],
            min=-1,
            max=-1
        )
    else:
        return dict(
            attempts=attempts,
            score=scores[-1],
            min=np.min(scores[:-1]),
            max=np.max(scores[:-1])
        )

def likelihood(theta, n_questions=50, min_correct=30):
    scores = []
    score = 0
    while score < min_correct:
        score = np.random.binomial(n=n_questions, p=theta)
        scores.append(score)
    scores = np.array(scores)

    return summary(scores)

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

Approximator

```
adapter = (
    bf.Adapter()
    .constrain("theta", lower=0.25, upper=1)
    .rename("theta", "inference_variables")
    .concatenate(["attempts", "score", "min", "max"], into="inference_conditions")
)
```

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

Training

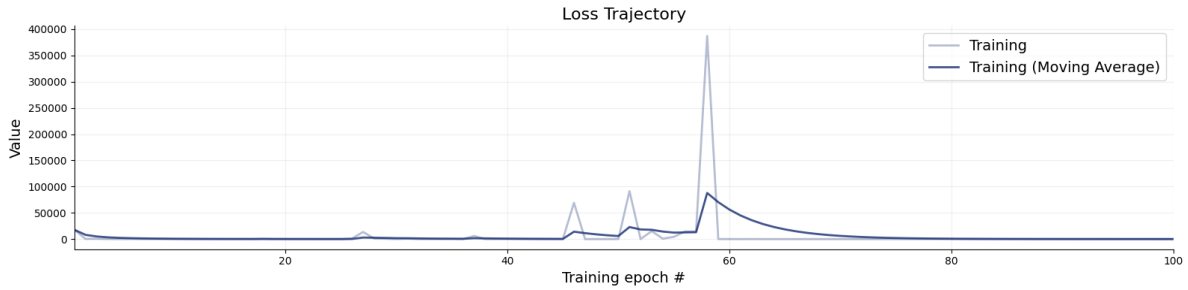
```
train_data = simulator.sample(1000)
val_data = simulator.sample(100)
```

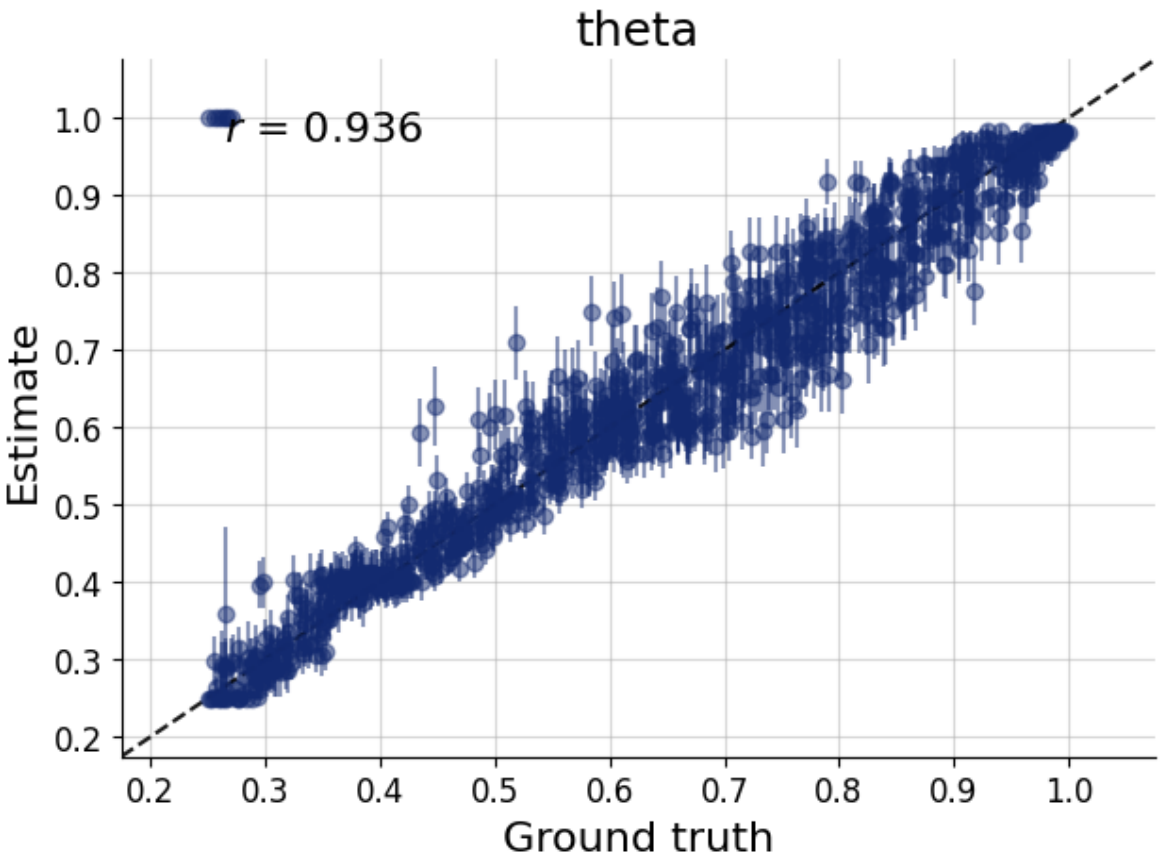
```
history=workflow.fit_offline(data = train_data, batch_size=100, validation_data=val_data)
```

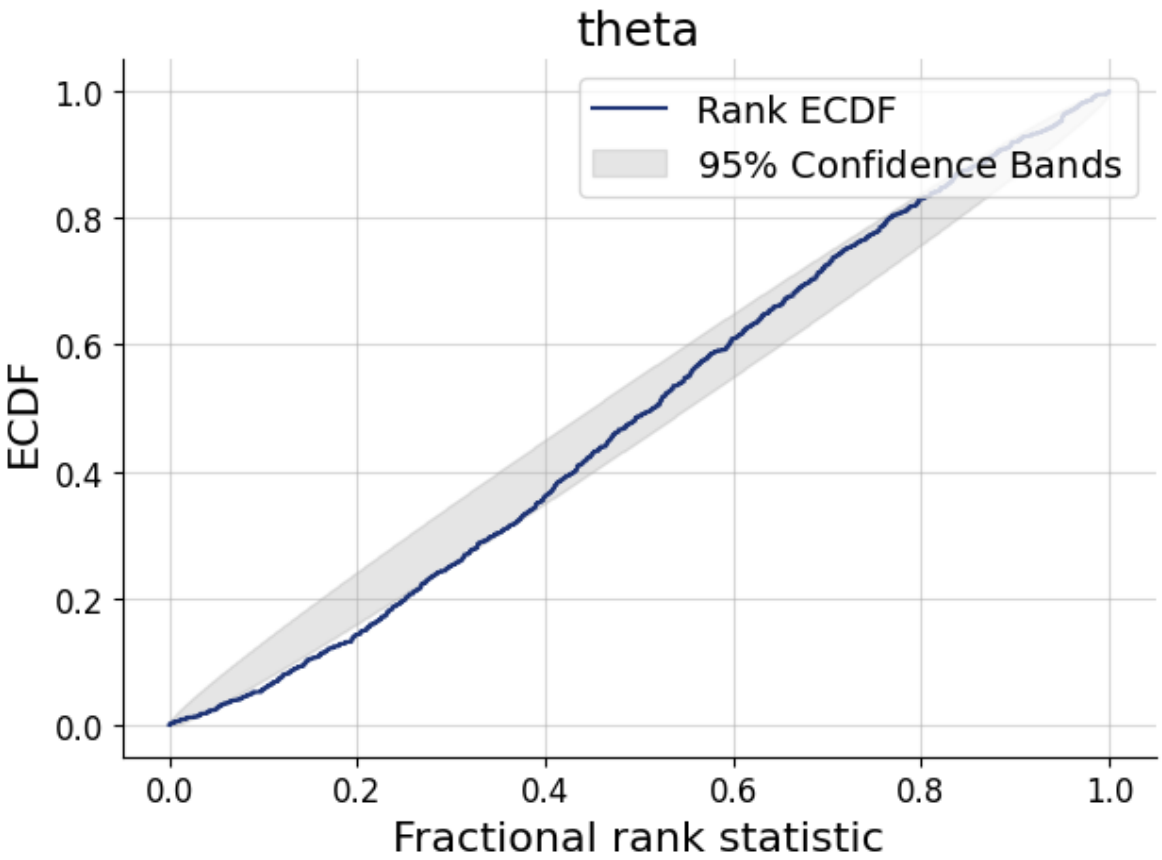
Validation

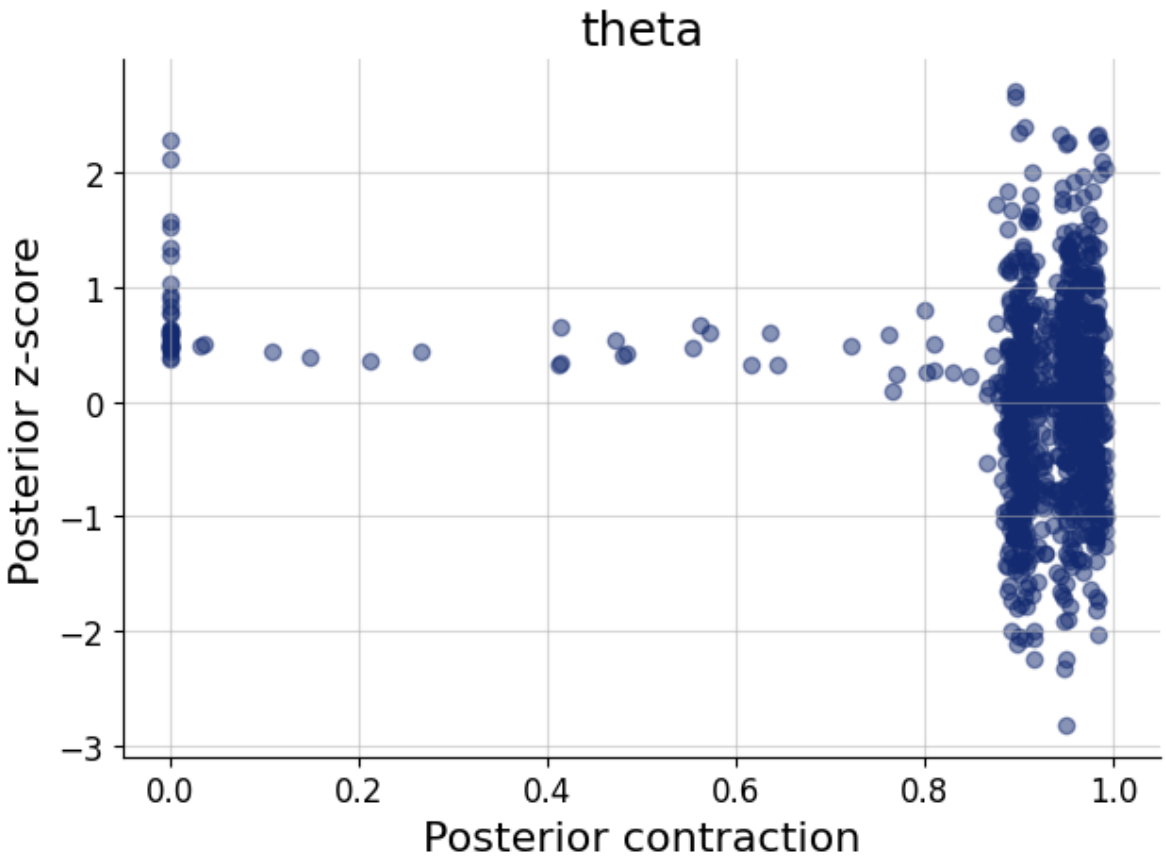
```
test_data = simulator.sample(1000)
```

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




theta
r = 0.936
Estimate
Ground truth
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
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



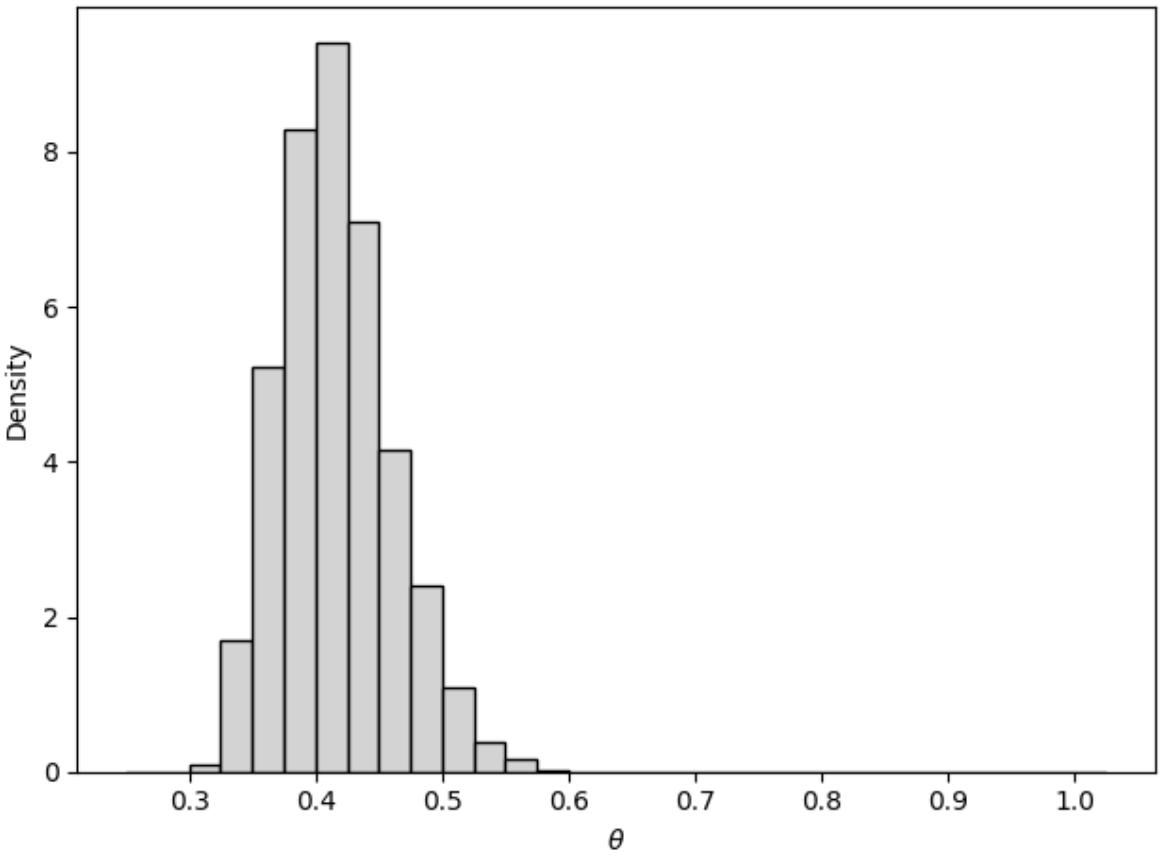
Inference

We will make an inference for Cha Sa-soon (Lee & Wagenmakers, 2013), who took 950 attempts to pass the test with a final score 30 correct questions. The range of the previous 949 attempts was between 15 and 25.

```
inference_data = dict(
    attempts = np.array([[950]]),
    score=np.array([[30]]),
    min=np.array([[15]]),
    max=np.array([[25]])
)
```

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

```
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.