

simulate__continous__network

June 12, 2025

1 Simulate Continuous Network

```
[1]: import pandas as pd
import pysmile
import pysmile_license
import numpy as np
import os
```

1.1 Create simulated data from a network with continuous variables

```
[2]: output_folder = "./simulated_data"
os.makedirs(output_folder, exist_ok=True)

simulated_dataset_path = "./simulated_data/simulated_cont_dataset.csv"
```

```
[3]: # v is arrow

#      C
#      /
#      v
#      A      B
#      \      /
#      v v
#      D

# Number of samples
n_samples = 50000

# Set random seed for reproducibility
np.random.seed(42)

# Node D: Independent Gaussian
C = np.random.normal(loc=0, scale=1, size=n_samples)

# Node A: Depends on D
epsilon_A = np.random.normal(loc=0, scale=1, size=n_samples)
A = 20 * C + epsilon_A
```

```

# Node B: Independent
B = np.random.normal(loc=1, scale=1, size=n_samples)

# Node C: Deterministic sum of A and B
D = A + B

# Combine into DataFrame for inspection or further use
data = pd.DataFrame({'C': C, 'A': A, 'B': B, 'D': D})

print(data.head())

data.to_csv(simulated_dataset_path, index=False)

```

	C	A	B	D
0	0.496714	10.032489	2.030595	12.063084
1	-0.138264	-2.829394	-0.155355	-2.984749
2	0.647689	13.905562	1.575437	15.480999
3	1.523030	31.993428	0.380762	32.374190
4	-0.234153	-3.996221	0.672597	-3.323623

1.2 Structure learning from simulated dataset

```

[4]: cont_ds = pysmile.learning.DataSet()

try:
    cont_ds.read_file(simulated_dataset_path)
except pysmile.SMILEException:
    print("Dataset load failed")
#endtry

print(f"Dataset has {cont_ds.get_variable_count()} variables (columns) "
      + f"and {cont_ds.get_record_count()} records (rows)")

```

Dataset has 4 variables (columns) and 50000 records (rows)

```

[5]: bayes_search = pysmile.learning.BayesianSearch()
bayes_search.set_iteration_count(50)
bayes_search.set_rand_seed(9876543)

## (1)
try:
    net1 = bayes_search.learn(cont_ds)
    print(f"1st Bayesian Search finished, structure score: {bayes_search.
    ↪get_last_score()}")
    net1.write_file("./simulated_data/learned_cont_net_bayes_1.xdsl")
except pysmile.SMILEException:
    print("Bayesian Search failed")

```

```

#endtry

## (2)
bayes_search = pysmile.learning.BayesianSearch()
bayes_search.set_iteration_count(50)
bayes_search.set_rand_seed(3456789)
try:
    net2 = bayes_search.learn(cont_ds)
    print(f"2nd Bayesian Search finished, structure score: {bayes_search.
    ↪get_last_score()}")
    net2.write_file("./simulated_data/learned_cont_net_bayes_2.xdsl")
except pysmile.SMILEException:
    print("Bayesian Search failed")
#endtry

## (3)
pc = pysmile.learning.PC()
try:
    pattern = pc.learn(cont_ds)
    net5 = pattern.make_network(cont_ds)
    print("PC finished, proceeding to parameter learning")
    net5.write_file("./simulated_data/learned_cont_net_pc.xdsl")
except pysmile.SMILEException:
    print("PC failed")
#endtry

```

Bayesian Search failed

Bayesian Search failed

PC finished, proceeding to parameter learning

1.3 Construct predefined network with continuous variables

```

[6]: def create_cont_node(net, id, name, x_pos, y_pos):

    handle = net.add_node(pysmile.NodeType.EQUATION, id)
    net.set_node_name(handle, name)
    net.set_node_position(handle, x_pos, y_pos, 85, 55)

    return handle

```

```

[7]: cont_net = pysmile.Network()

A = create_cont_node(cont_net, "A", "A", 10, 20)
B = create_cont_node(cont_net, "B", "B", 10, 30)
C = create_cont_node(cont_net, "C", "C", 10, 40)
D = create_cont_node(cont_net, "D", "D", 10, 50)

```

```
cont_net.add_arc(C, A)
cont_net.add_arc(A, D)
cont_net.add_arc(B, D)

cont_net.write_file("./simulated_data/predefined_cont_net.xdsl")
```

1.4 Learn parameter for predefined network

```
[8]: em = pysmile.learning.EM()

try:
    matching = cont_ds.match_network(cont_net)
except pysmile.SMILEException:
    print("Can't automatically match network with dataset")
#endtry

em.set_uniformize_parameters(False)
em.set_randomize_parameters(False)
em.set_eq_sample_size(0)

try:
    em.learn(cont_ds, cont_net, matching)
except pysmile.SMILEException:
    print("EM failed")
#endtry

print("EM finished")
cont_net.write_file("./simulated_data/simulated_data_em_cont.xdsl")
print("Complete.")
```

EM finished
Complete.