Benchmarking performance

Import the required libraries

[1]:
```
import timeit

import numpy as np

import pyvinecopulib as pv

from collections import defaultdict


def print_results(results, name):
  print(
    f"{name}: min={min(results) * 1000:.6f}, mean={np.mean(results) * 1000:.6f},
max={max(results) * 1000:.6f}"
  )
```

Set the parameters for the benchmark

[2]:
```
# TODO: Check TLL because compiler might have large impact on performance there

n = 1000
d = 5
repeats = 5


def generate_data(n, d, seed=0):
  np.random.seed(seed)
  # Generate data
  x = np.random.normal(size=n)[:, np.newaxis] * np.ones(
    (n, d)
  ) + 0.5 * np.random.normal(size=(n, d))

  # Convert data to pseudo-observations
  u = pv.to_pseudo_obs(x)

  return u


def generate_data_bicop(n, seed):
  cop = pv.Bicop(family=pv.gaussian, parameters=np.array([[0.5]]))
  return cop.simulate(n, seeds=[seed])
```

Bivariate copula benchmark

[3]:

```python
def itau_bicop(u):
  pv.Bicop.from_data(
    u, controls=pv.FitControlsBicop(family_set=pv.itau, num_threads=1)
  )


def itau_par_method_bicop(u):
  pv.Bicop.from_data(
    u,
    controls=pv.FitControlsBicop(
      family_set=pv.itau, parametric_method="itau", num_threads=1
    ),
  )


def tll_bicop(u):
  pv.Bicop.from_data(
    u, controls=pv.FitControlsBicop(family_set=[pv.tll], num_threads=1)
  )


functions_to_benchmark_bicop = {
  "itau": itau_bicop,
  "itau_par_method": itau_par_method_bicop,
  "tll": tll_bicop,
}

# Benchmark the functions
results_bicop = defaultdict(list)
for seed in range(repeats):
  u = generate_data_bicop(n, seed)
  for name, function in functions_to_benchmark_bicop.items():
    tic = timeit.default_timer()
    function(u)
    toc = timeit.default_timer()
    results_bicop[name].append(toc - tic)


# Display benchmark results
print("Unit (milliseconds, bicop):")
for name, result in results_bicop.items():
  print_results(result, name)
```

```
Unit (milliseconds, bicop):
itau: min=134.766681, mean=227.786324, max=290.670977
itau_par_method: min=20.589506, mean=21.285831, max=23.578367
tll: min=11.247802, mean=11.617707, max=11.900096
```

Vine copula benchmark

[4]:

```python
# Various configurations of the `vinecop` function
controls_itau = pv.FitControlsVinecop(family_set=pv.itau, num_threads=1)
controls_itau_par_method = pv.FitControlsVinecop(
  family_set=pv.itau, parametric_method="itau", num_threads=1
)
controls_tll = pv.FitControlsVinecop(family_set=[pv.tll], num_threads=1)


# Define two different configurations of the `vinecop` function
def vinecop_itau(u):
  return pv.Vinecop.from_data(u, controls=controls_itau)


def vinecop_itau_par_method(u):
  return pv.Vinecop.from_data(u, controls=controls_itau_par_method)


def vinecop_tll(u):
  return pv.Vinecop.from_data(u, controls=controls_tll)


functions_to_benchmark = {
  "itau": vinecop_itau,
  "itau_par_method": vinecop_itau_par_method,
  "tll": vinecop_tll,
}

# Benchmark the functions
results = defaultdict(list)
for seed in range(repeats):
  u = generate_data(n, d, seed)
  for name, function in functions_to_benchmark.items():
    tic = timeit.default_timer()
    function(u)
    toc = timeit.default_timer()
    results[name].append(toc - tic)


# Display benchmark results
print("Unit (milliseconds):")
for name, result in results.items():
  print_results(result, name)

# # Benchmark in C++ directly
# results_cpp = defaultdict(list)
# for seed in range(repeats):
#   u = generate_data(n, d, seed)
#   results_cpp_tmp = pv.pyvinecopulib_ext.benchmark(u)
#   for key, value in zip(functions_to_benchmark.keys(), results_cpp_tmp):
#     results_cpp[key].append(value)

# # Display benchmark results
# print("Unit (milliseconds, C++):")
# for name, result in results_cpp.items():
#   print_results(result, name)
```

```
Unit (milliseconds):
itau: min=1613.363917, mean=1872.592578, max=2094.827519
itau_par_method: min=198.184067, mean=207.710620, max=212.484254
tll: min=132.396423, mean=134.724716, max=137.676346
```