Dppsampl Package

11th November 2023

Authors:

Ada Gąssowska <ada.gassowska.stud@pw.edu.pl>

Katarzyna Solawa <katarzyna.solawa.stud@pw.edu.pl>

Kacper Kurowski <kacper.kurowski.dokt@pw.edu.pl>

Description:

This package provides algorithms for sampling from the Determinantal Point Processes (DPPs) for SAS.

Version:

0.5

Modules:

- `sample_from_beta_ensemble_full` Function
- `sample_from_beta_ensemble_banded` Function

Overview:

The dppsampl package provides an implementation of the algorithms for sampling from the Determinanatal Point Processes (DPPs) written natively in SAS. DPPs are stochastic point processes (so, their result is a subset of that set) which respect the diversity present in the set.

The package offers ways to sample from Finite, Continuous and the so-called Exotic DPPs. For Finite DPPs the set of possible values is finite, for continuous it is infinite and the samples should approximate some continuous distribution. The exotic DPPs however, are a special kind of DPPs which were analyzed for different reasons and one needed a change-of-perspective to deduce that they can be thought of as DPPs.

The available continuous DPPs come from a family of distributions called Beta ensembles. They were initially studied in Physics, as they can be thought of as models of a Coulomb gas.

Before you use the dppsampl package, you must install it by using the `package install` statement. For example, if the ZIP file is located in the directory `C:\Packages`, then the following statement installs the package:

```
proc iml;
package install "C:\Packages\dppsampl.zip";
quit;
```

Data Sets

We provide four test datasets which can be used to test the implementation of finite DPPs. All datasets have been constructed from the `k.sas7bdat` one.

- `k.sas7bdat` — correlation kernel which is a 10×10 projection matrix with rank of 8,
- `l.sas7bdat` — matrix created from `k.sas7bdat` using the relation $K = L(I + L)^{-1}$,
- `eig_vals.sas7bdat` — set of eigenvalues of `k.sas7bdat`,
- `eig_vals.sas7bdat` — set of eigenvectors of `k.sas7bdat`.

Available Functions:

sample_from_beta_ensemble_full Function

Syntax:

```
sample_from_beta_ensemble_full(
    ensemble_version,
    M_1, M_2,
    size=10,
    beta=2,
    normalize=1,
    haar_mode="Hermite",
    heurestic_fix=1,
    random_state=1618
);
```

Parameters

`ensemble_version`	Version of Beta ensemble to use. Available values are `"Hermite"`, `"Laguerre"`, `"Jacobi"`, `"Circular"`, and "`"Ginibre"`
`M_1`	Distribution parameter for the `"Laguerre"` and `"Jacobi"` `ensemble_version`s. Should be greater or equal to `size`.
`M_1`	Distribution parameter for the `"Jacobi"` `ensemble_version`. Should be greater or equal to `size`.
`size`	Size of the sampled subset.
`beta`	Beta parameter. Should be 1, 2, or 4.
`normalize`	Parameter which states whether the sample should be normalized to fit one of the more known distributions.
`haar_mode`	Which Haar measure mode to use. Can be `"Hermite"` or `"QR"`. Influences the result only for the Circular Ensemble. (Should be 1 or 0).
`heurestic_fix`	Whether to apply the heuresis to fix the oversampling problem present in the Circular and Ginibre ensembles. Should be 1 or 0.
`random_state`	Seed for the randomness. Should be a positive integer.

Description

The function provides the method for sampling from beta ensemble using the full matrix method. There are five versions of Beta ensembles that have been implemented. "Hermite", "Laguerre", "Jacobi", "Circular", and "Ginibre". For the first three, the result is a one-column sample. For the next two, it is a two-column sample.

Example:

```
ensemble_version = "Circular";
size=10;
beta=4;
M_1=10; M_2 = 10;
haar_mode="Hermite";
normalize=0;
heurestic_fix=0;
random_state=1618;


sample = sample_from_beta_ensemble_full(
        ensemble_version,
        M_1, M_2,
        size,
        beta,
        normalize,
        haar_mode,
        heurestic_fix,
        random_state
);

run scatter(sample[,1], sample[,2]);
```

sample_from_beta_ensemble_banded Function

Syntax

```
sample_from_beta_ensemble_banded(
    ensemble_version,
    size=10,
    beta=2,
    loc=0.0,
    scale=1.0,
    shape = 1.0,
    a = 1.0, b = 1.0,
    normalize=1,
    heurestic_fix=1,
    random_state=1618
);
```

Parameters

`ensemble_version`	Version of Beta ensemble to use. Available values are `"Hermite"`, `"Laguerre"`, `"Jacobi"`, and `"Ginibre"`
`size`	Size of the sampled subset.
`beta`	Beta parameter. Should be positive integer.
`loc`	Location parameter for the standard deviation for the `"Hermite"` Beta ensemble.
`scale`	Scale parameter for the expected value for the `"Hermite"` and `"Laguerre"` Beta ensembles.
`shape`	Shape parameter for the `"Laguerre"` Beta ensemble.
`a`	Parameter for the `"Jacobi"` Beta ensemble. Related to the Beta distribution.
`b`	Parameter for the `"Jacobi"` Beta ensemble. Related to the Beta distribution.
`normalize`	Parameter which states whether the sample should be normalized to fit one of the more known distributions.
`heurestic_fix`	Whether to apply the heuresis to fix the oversampling problem present in the Circular and Ginibre ensembles. Should be 1 or 0.
`random_state`	Seed for the randomness. Should be a positive integer.

Description

The function provides the method for sampling from beta ensemble using the banded matrix method. There are five versions of Beta ensembles that have been implemented. `"Hermite"`, `"Laguerre"`, `"Jacobi"`, and `"Ginibre"`. For the first three, the result is a one-column sample. For the Ginibre ensemble, it is a two-column sample.

Example

```
ensemble_version = "Hermite";
size=1000;
beta=2;
loc=0.0;
scale=1.0;
shape = 1.0;
a = 1.0;
b = 1.0;
normalize=0;
heurestic_fix=0;
random_state=1618;


sample = sample_from_beta_ensemble_banded(
    ensemble_version,
    size,
    beta,
    loc,
    scale,
    shape,
    a, b,
    normalize,
    heurestic_fix,
    random_state
);

run histogram(sample);
```