

APL64 PDIST: Probability Distributions

Overview	4
Study the  PDIST and MathNet Documentation	4
Pseudo-Random Number Generators.....	4
Pseudo-Random Number Generators.....	4
Pseudo-Random Seed	4
Repeatability	4
 PDIST Workflow.....	5
Select the Pseudo-random Number Generator	5
Create a Probability Distribution Instance	5
Review the Probability Distribution Properties.....	6
Obtain Probability Distribution Samples.....	6
Consider the Other  PDIST Actions	6
Syntax.....	6
 PDIST Object Actions Syntax	6
 PDIST Instance Actions Syntax	6
Object and Instance Example.....	6
 PDIST Actions	7
 PDIST Object Actions.....	7
Clear	7
Count.....	7
Create.....	8
Bernoulli.....	9
Beta	9
Binomial	9
Categorical	10
Cauchy.....	10
Chi	11
Chi-Squared.....	11
Continuous-Uniform	12
Conway-Maxwell-Poisson	12
Dirichlet.....	12

Discrete-Uniform.....	13
Erlang	13
Exponential	14
Fisher-Snedecor	14
Gamma.....	15
Geometric	15
Hyper-Geometric.....	16
Inverse-Gamma	16
InverseWishart	17
Laplace	17
Log-Normal.....	18
MatrixNormal.....	18
Multinomial.....	19
Negative-Binomial	19
Normal	20
NormalGamma.....	20
Pareto	21
Poisson	21
Rayleigh	22
Stable	22
Student-T.....	23
Triangular	23
Weibull	24
Wishart.....	24
Zipf	25
Delete.....	25
Help (?).....	26
Instances	27
New	27
☐ PDIST Instance Actions.....	27
CumDist.....	28
Density	28
Most Distributions.....	28

Dirichlet	29
Inverse Wishart	29
NormalGamma	30
Wishart.....	30
DensityLn.....	31
InvCumDist	31
Probability	32
Most Distributions.....	32
Multinomial distribution	32
ProbabilityLn	33
Properties.....	34
RandomSource	35
Pseudo-random Number Generators: Source Names:	35
Random Seed Names:	36
Samples	37
Most Distributions.....	37
Dirichlet.....	37
InverseWishart, MatrixNormal, Wishart.....	37
MultiNomial	38
NormalGamma.....	38
Self.....	38
☐ PDISTSELF System Variable	38

Overview

The APL64 `▢PDIST` system function is an interface to the MathNet [Probability](#) and [MathNet Random](#) toolkits. The `▢PDIST` system function provides the `Samples` action to obtain sample values for the supported probability distributions using an APL64 programmer-selected pseudo-random number generators.

Study the `▢PDIST` and MathNet Documentation

The `▢PDIST` documentation, and the Mathnet Probability and Random toolkits documentation are mandatory reading to effectively use `▢PDIST`.

Pseudo-Random Number Generators

Pseudo-Random Number Generators

`▢PDIST` requires the APL64 programmer to select an appropriate pseudo-random number generator for each probability distribution used. [▢PDIST provides a selection of pseudo-random number generators](#).

Pseudo-Random Seed

Each instance of a pseudo-random number generator requires a random seed. [▢PDIST provides a selection of random seed options](#).

Repeatability

If the same random seed is used, e.g. a fixed number, all sequences of pseudo-random numbers obtained from a pseudo-random number generator will be the same. This behavior is useful for testing and debugging of an application system but is not appropriate for a production application.

Because of this behavior of pseudo-random number generators, the structure of the `▢PDIST` system function provides for creating instances of probability distributions. Recurring requests for sample values of a probability distribution should be made to the same instance of that probability distribution.

NOTE: Some examples do not use a fixed numeric random seed, so the illustrated result may not be repeatable. Round off error in printed results can also affect the repeatability of results, even when a fixed numeric random seed is used.

Example:

```
'#▢PDist 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' '16807' 'Bernoulli' 0.5
'Bernoulli1' ▢PDist 'Samples' 2
'Bernoulli1' ▢PDist 'Samples' 2
'Bernoulli1' ▢PDist 'Samples' 2
'#▢PDist 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' '16807' 'Bernoulli' 0.5
'Bernoulli1' ▢PDist 'Samples' 2
'Bernoulli1' ▢PDist 'Samples' 2
'Bernoulli1' ▢PDist 'Samples' 2
'#▢PDist 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'Guid' 'Bernoulli' 0.5
'Bernoulli1' ▢PDist 'Samples' 2
'Bernoulli1' ▢PDist 'Samples' 2
'Bernoulli1' ▢PDist 'Samples' 2
```

```
#'PDIST 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'Guid' 'Bernoulli' 0.5
'Bernoulli1' PDIST 'Samples' 2
'Bernoulli1' PDIST 'Samples' 2
'Bernoulli1' PDIST 'Samples' 2
```

```
0      #'PDIST 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' '16807' 'Bernoulli' 0.5
1      Bernoulli1
2      'Bernoulli1' PDIST 'Samples' 2
3      1 1
4      'Bernoulli1' PDIST 'Samples' 2
5      0 0
6      'Bernoulli1' PDIST 'Samples' 2
7      1 1
8      #'PDIST 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'Guid' 'Bernoulli' 0.5
9      Bernoulli1
10     'Bernoulli1' PDIST 'Samples' 2
11     1 1
12     'Bernoulli1' PDIST 'Samples' 2
13     0 0
14     'Bernoulli1' PDIST 'Samples' 2
15     1 1
16     #'PDIST 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'Guid' 'Bernoulli' 0.5
17     Bernoulli1
18     'Bernoulli1' PDIST 'Samples' 2
19     1 1
20     'Bernoulli1' PDIST 'Samples' 2
21     1 0
22     'Bernoulli1' PDIST 'Samples' 2
23     0 1
24     #'PDIST 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'Guid' 'Bernoulli' 0.5
25     Bernoulli1
26     'Bernoulli1' PDIST 'Samples' 2
27     0 0
28     'Bernoulli1' PDIST 'Samples' 2
29     0 1
30     'Bernoulli1' PDIST 'Samples' 2
31     1 1
32
```

PDIST Workflow

Select the Pseudo-random Number Generator

Select the pseudo-random number generator and random seed used to create a PDIST probability distribution instance. See the PDIST [RandomSource](#) action for the available options.

Create a Probability Distribution Instance

Use the PDIST [Create](#) or [New](#) action to create a probability distribution instance. Creating a probability distribution instance involves selecting the random seed type, the random number generator, the probability distribution type, and providing the probability distribution initialization parameters.

Review the Probability Distribution Properties

Use the `□PDIST` [Properties](#) action to verify that the probability distribution instance is appropriate.

Obtain Probability Distribution Samples

Use the `□PDIST` [Samples](#) action to obtain an APL64 rank-1 array of probability distribution samples.

Consider the Other `□PDIST` Actions

`□PDIST` includes [additional actions](#).

Syntax

`Result←[larg] □PDIST Action ActionArg1 ...`

Result: result of the specified action. Refer to [□PDIST '?'](#) for a summary of result types

larg: text

- '#' for `□PDist` object actions
- instanceName for `□PDist` instance actions
- optional for `□PDist` instance actions, when for `□PDistSelf` is set to an instance name

The right argument of `□PDIST` is an APL64 rank-1 array containing the Action text and any necessary action arguments.

Action: Case-insensitive text indicating the user-selected action

ActionArgs1, ... required arguments for the selected Action.

Multiple instances of the same or different probability distributions may be created using the `□PDIST` system function. All such probability distribution instances are contained within the `□PDIST` object.

`□PDIST` Object Actions Syntax

Syntax: `Result← '#' □PDist Action ActionArg1 ...`

`□PDIST` Instance Actions Syntax

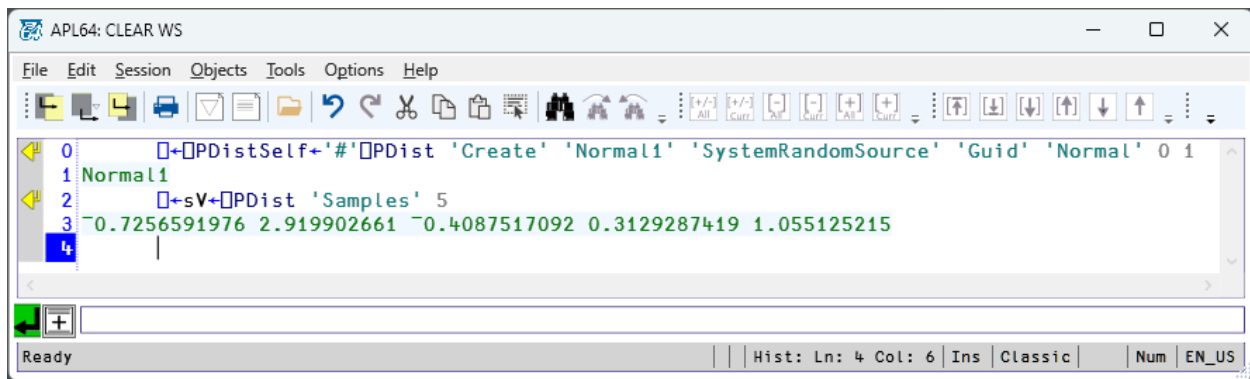
Syntax: `Result←[instanceName] □PDist Action ActionArg1 ...`

The instanceName is optional when the `□PDistSelf` system variable is set to a `□PDIST` instance name.

Object and Instance Example

The APL64 `□PDistSelf` system variable is set to a `□PDIST` instance name, no left argument to the `□PDIST` system function is necessary for actions on that instance:

<pre>□←□PDistSelf←'#'□PDist 'Create' 'Normal1' 'SystemRandomSource' 'Guid' 'Normal' 0 1 □←sV←□PDist 'Samples' 5</pre>



⎵⎵PDIST Actions

⎵⎵PDIST Object Actions

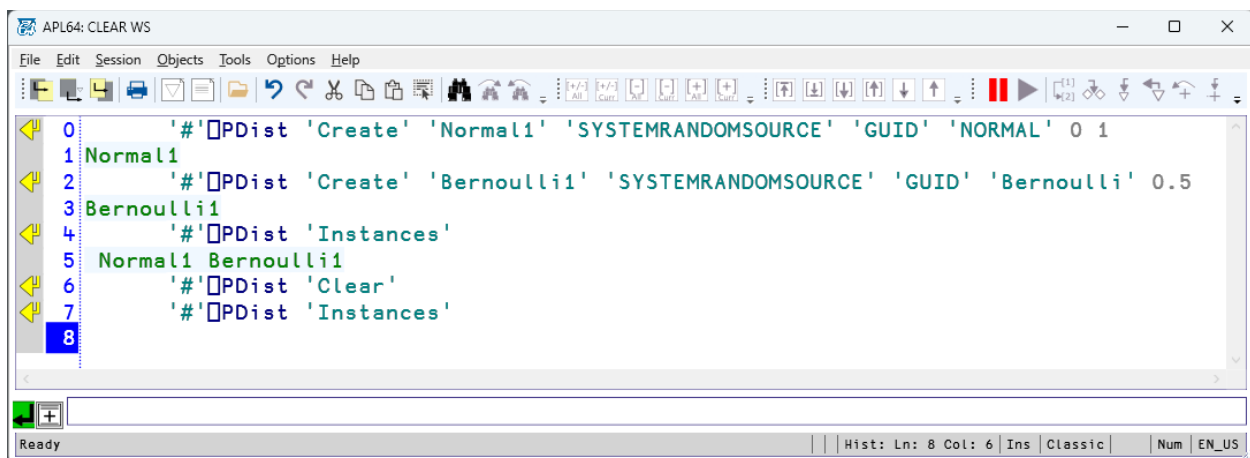
Clear

Clear all ⎵⎵PDIST instances

```

'#'⎵⎵PDIST 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
'#'⎵⎵PDIST 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
'#'⎵⎵PDIST 'Instances'
'#'⎵⎵PDIST 'Clear'
'#'⎵⎵PDIST 'Instances'

```



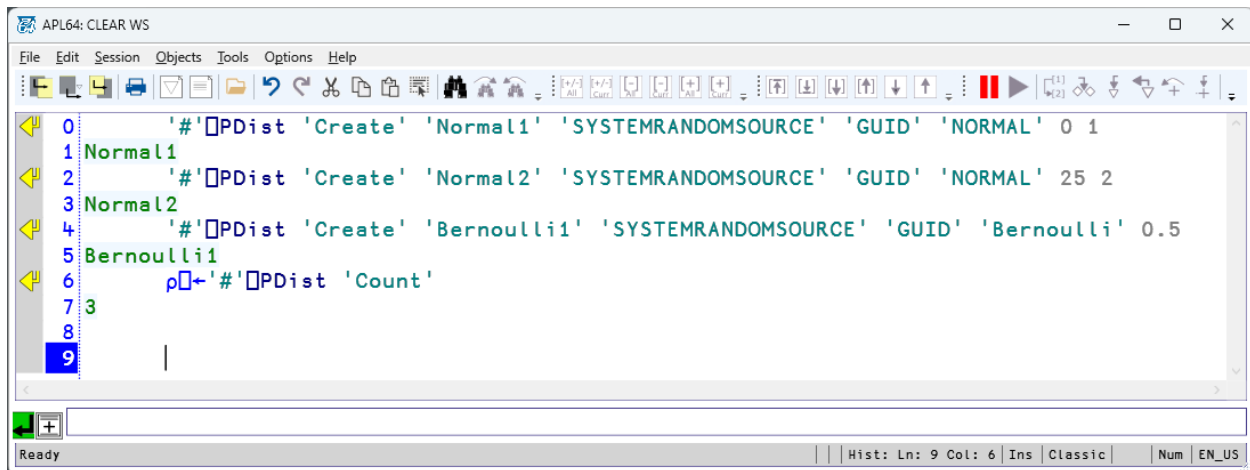
Count

Return the number of ⎵⎵PDIST instances

```

'#'⎵⎵PDIST 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
'#'⎵⎵PDIST 'Create' 'Normal2' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 25 2
'#'⎵⎵PDIST 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
ρ⎵⎵←'#'⎵⎵PDIST 'Count'

```



```
0      '#⊞PDist 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
1 Normal1
2      '#⊞PDist 'Create' 'Normal2' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 25 2
3 Normal2
4      '#⊞PDist 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
5 Bernoulli1
6      ρ⊞- '#⊞PDist 'Count'
7      3
8
9      |
```

Create

Create a new ☐PDIST probability distribution instance, possibly replacing an existing ☐PDIST instance.

Syntax: `instName←'⊞⊞PDist 'Create' instName rndSrcName rndSeedName distName param1 ...`

Each ☐PDIST probability distribution instance must have a unique instance name, `instName`.

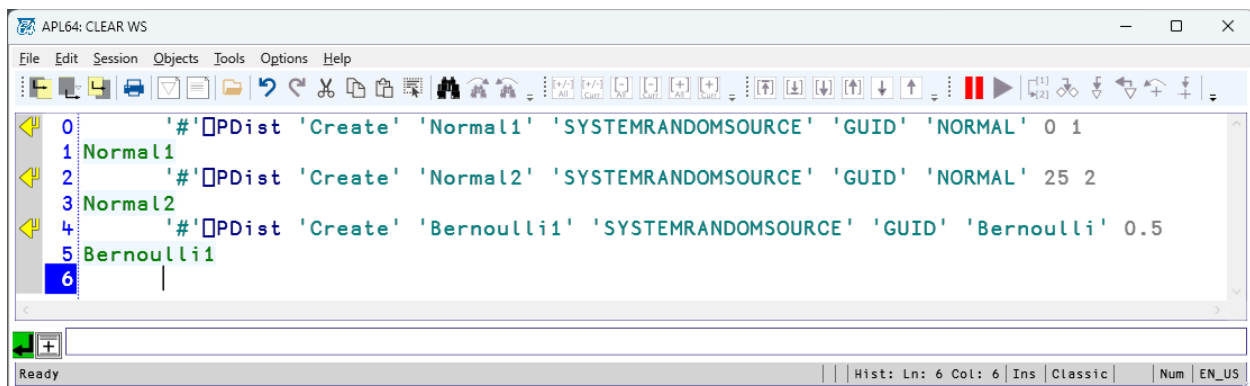
[rndSrcName](#) is the name of the selected pseudo-random number generator.

`rndSeedName` is the name of the selected random seed option.

`distName` is the name of the selected probability distribution.

`param1 ...` are the required initialization parameters for the selected probability distribution.

```
'⊞⊞PDist 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
'⊞⊞PDist 'Create' 'Normal2' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 25 2
'⊞⊞PDist 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
```



```
0      '#⊞PDist 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
1 Normal1
2      '#⊞PDist 'Create' 'Normal2' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 25 2
3 Normal2
4      '#⊞PDist 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
5 Bernoulli1
6      |
```

Each type of ☐PDIST probability distribution requires initialization parameters. The required initialization parameters differ among ☐PDIST probability distributions.

The probability distribution names are not case-sensitive.

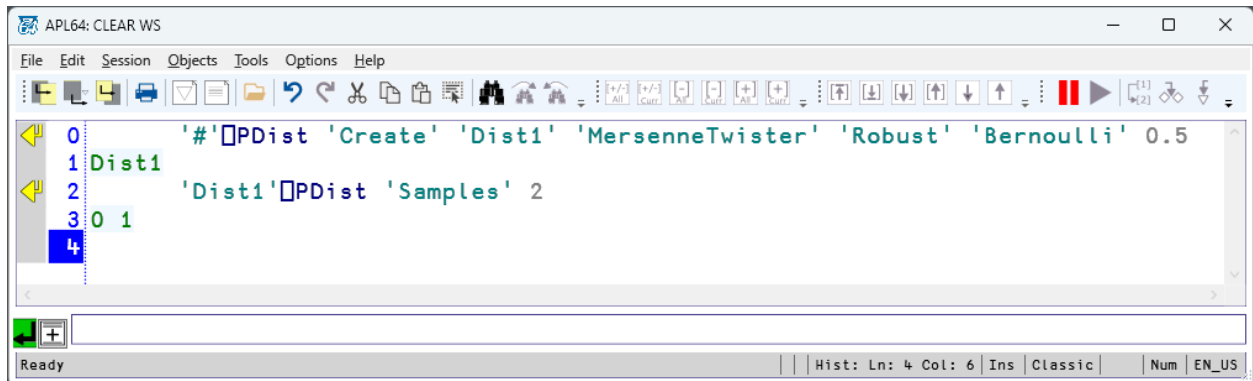
Probability distributions supported by Mathnet:

Bernoulli

Syntax: '#'☐PDist 'Create' instName rndSrcName rndSeedName 'Bernoulli' prob1

prob1: Double: Probability of generating 1 in range [0, 1]

```
'#'☐PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Bernoulli' 0.5  
'Dist1'☐PDist 'Samples' 2
```



The screenshot shows the APL64: CLEAR WS interface. The code editor contains the following code:
0 '#'☐PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Bernoulli' 0.5
1 Dist1
2 'Dist1'☐PDist 'Samples' 2
3 0 1
4
The execution results are shown in the lower pane: 0 1. The status bar at the bottom indicates 'Ready' and 'Hist: Ln: 4 Col: 6 | Ins | Classic | Num | EN_US'.

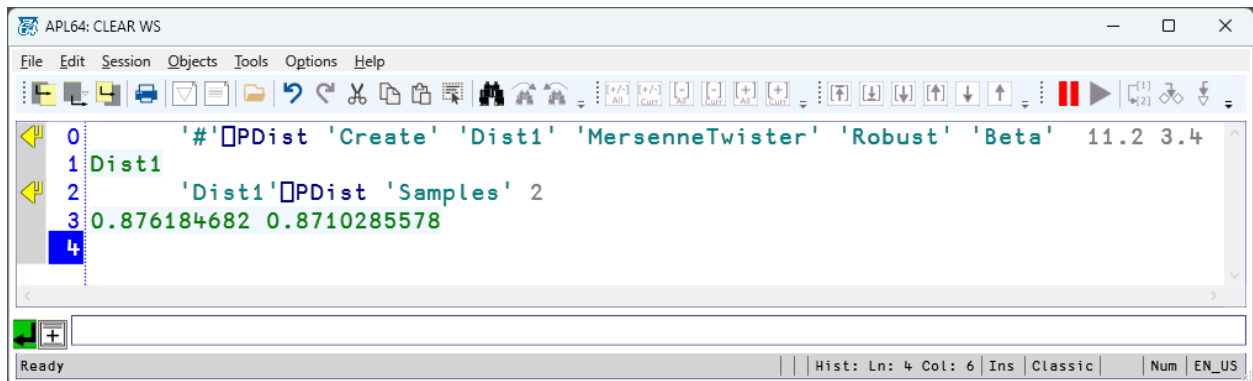
Beta

Syntax: '#'☐PDist 'Create' instName rndSrcName rndSeedName 'Beta' a b

a: Double: Alpha shape parameter in range: $0 \leq a$

b: Double: Beta shape parameter in range: $0 \leq b$

```
'#'☐PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Beta' 11.2 3.4  
'Dist1'☐PDist 'Samples' 2
```



The screenshot shows the APL64: CLEAR WS interface. The code editor contains the following code:
0 '#'☐PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Beta' 11.2 3.4
1 Dist1
2 'Dist1'☐PDist 'Samples' 2
3 0.876184682 0.8710285578
4
The execution results are shown in the lower pane: 0.876184682 0.8710285578. The status bar at the bottom indicates 'Ready' and 'Hist: Ln: 4 Col: 6 | Ins | Classic | Num | EN_US'.

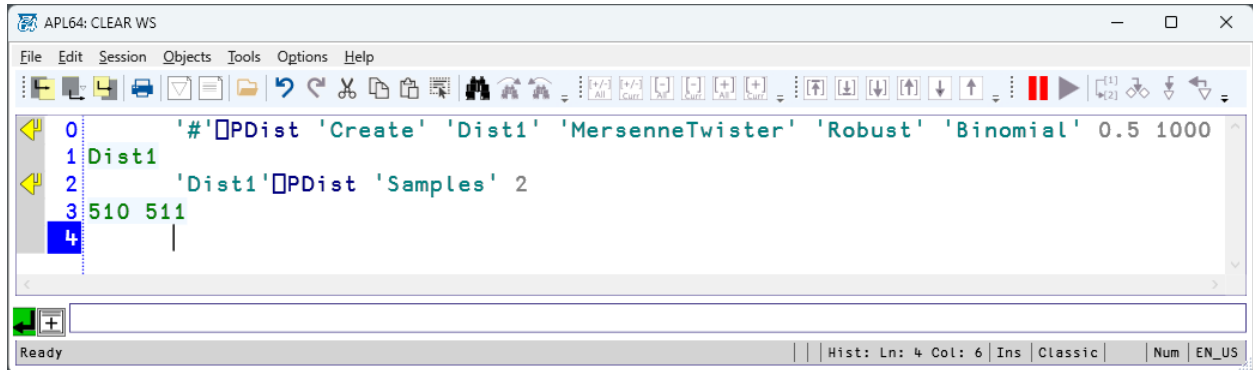
Binomial

Syntax: '#'☐PDist 'Create' instName rndSrcName rndSeedName 'Binomial' pSuccess nTrials

pSuccess: Double: Probability of success in each trial in range [0,1]

nTrials: Int32: 0≤#Trials

```
'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Binomial' 0.5 1000  
'Dist1'⎕PDist 'Samples' 2
```



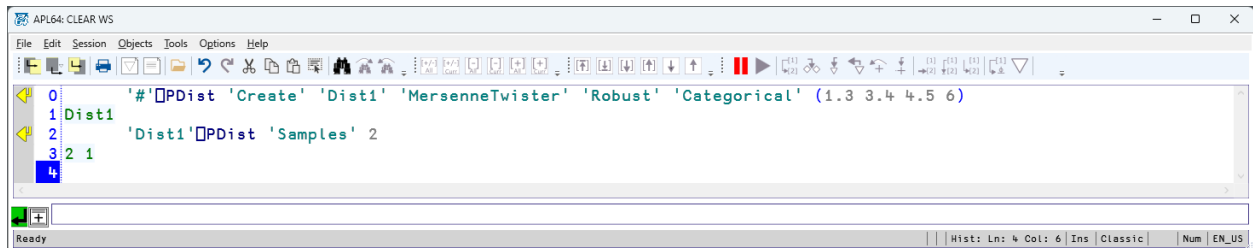
The screenshot shows the APL64: CLEAR WS interface. The code editor contains the following code:
0 '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Binomial' 0.5 1000
1 Dist1
2 'Dist1'⎕PDist 'Samples' 2
3 510 511
4
The status bar at the bottom indicates 'Ready' and 'Hist: Ln: 4 Col: 6 Ins Classic Num EN_US'.

Categorical

Syntax: '#⎕PDist 'Create' instName rndSrcName rndSeedName 'Categorical' massRatios[]

massRatios[]: Double[]: Mass ratios array: 0≤Each Element

```
'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Categorical' (1.3 3.4 4.5 6)  
'Dist1'⎕PDist 'Samples' 2
```



The screenshot shows the APL64: CLEAR WS interface. The code editor contains the following code:
0 '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Categorical' (1.3 3.4 4.5 6)
1 Dist1
2 'Dist1'⎕PDist 'Samples' 2
3 2 1
4
The status bar at the bottom indicates 'Ready' and 'Hist: Ln: 4 Col: 6 Ins Classic Num EN_US'.

Cauchy

Syntax: '#⎕PDist 'Create' instName rndSrcName rndSeedName 'Cauchy' location scale

location: Double: x0 of the distribution in range: 0≤location

scale: Double: Scale of the distribution in range: 0≤scale

```
'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Cauchy' 1.5 100  
'Dist1'⎕PDist 'Samples' 2
```

```

0      '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Cauchy' 1.5 100
1 Dist1
2      'Dist1'⎕PDist 'Samples' 2
3 123.1260319 406.0598629

```

Ready | Hist: Ln: 4 Col: 6 | Ins | Classic | Num | EN_US

Chi

Syntax: '#⎕PDist 'Create' instName rndSrcName rndSeedName 'Chi' freedom

freedom: Int32: Degrees of freedom in range: 0<freedom

```

'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Chi' 2
'Dist1'⎕PDist 'Samples' 2

```

```

0      '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Chi' 2
1 Dist1
2      'Dist1'⎕PDist 'Samples' 2
3 1.922618535 1.183203971

```

Ready | Hist: Ln: 4 Col: 6 | Ins | Classic | Num | EN_US

Chi-Squared

Syntax: '#⎕PDist 'Create' instName rndSrcName rndSeedName 'Chi-Squared' freedom

freedom: Int32: Degrees of freedom in range: 0<freedom

```

'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Chi-Squared' 2
'Dist1'⎕PDist 'Samples' 2

```

```

0      '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Chi-Squared' 2
1 Dist1
2      'Dist1'⎕PDist 'Samples' 2
3 1.475543099 0.54426793

```

Ready | Hist: Ln: 4 Col: 6 | Ins | Classic | Num | EN_US

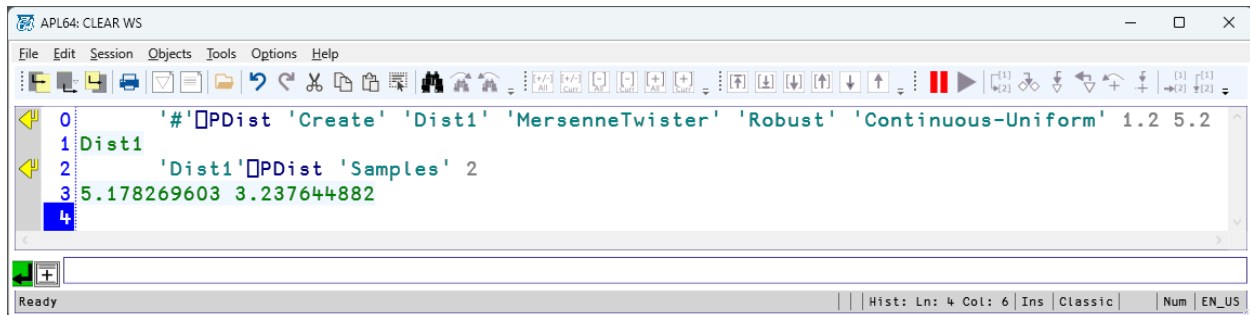
Continuous-Uniform

Syntax: '#'☐PDist 'Create' instName rndSrcName rndSeedName 'Continuous-Uniform' lower upper

lower: Double: Lower bound in range: $0 \leq \text{lower}$

upper: Double: Upper bound in range: $0 \leq \text{upper}$ and $\text{lower} < \text{upper}$

```
'#'☐PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Continuous-Uniform' 1.2 5.2
'Dist1'☐PDist 'Samples' 2
```



```
APL64: CLEAR WS
File Edit Session Objects Tools Options Help
0 '#'☐PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Continuous-Uniform' 1.2 5.2
1 Dist1
2 'Dist1'☐PDist 'Samples' 2
3 5.178269603 3.237644882
4
```

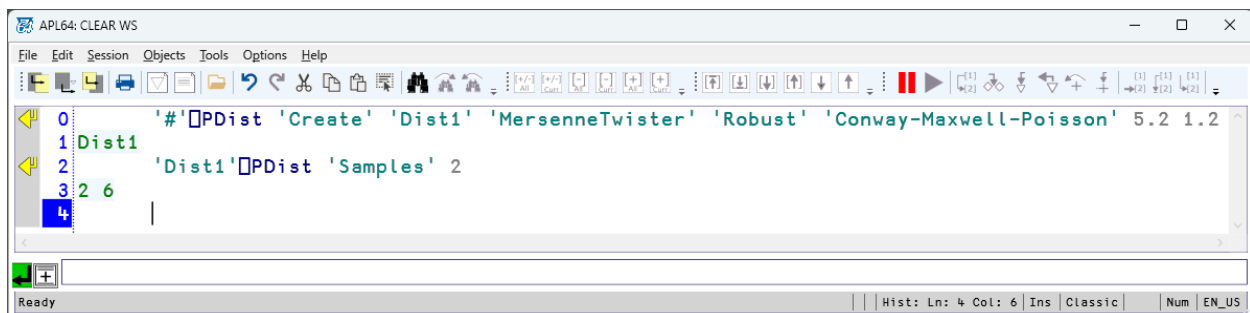
Conway-Maxwell-Poisson

Syntax: '#'☐PDist 'Create' instName rndSrcName rndSeedName 'Conway-Maxwell-Poisson' lambda nu

lambda: Double: In range: $0 < \text{lambda}$

nu: Double: Rate of decay: In range: $0 \leq \text{nu}$

```
'#'☐PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Conway-Maxwell-Poisson' 5.2 1.2
'Dist1'☐PDist 'Samples' 2
```



```
APL64: CLEAR WS
File Edit Session Objects Tools Options Help
0 '#'☐PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Conway-Maxwell-Poisson' 5.2 1.2
1 Dist1
2 'Dist1'☐PDist 'Samples' 2
3 2 6
4
```

Dirichlet

Syntax: '#'☐PDist 'Create' instName rndSrcName rndSeedName 'Dirichlet' (Double[n])

Double[n] is a vector of double numbers, where n is the dimension of the distribution

```
'#'☐PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Dirichlet' (1 2 3.4 5)
ρ''☐←'Dist1'☐PDist 'Samples' 2
```

```

0 '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Dirichlet' (1 2 3.4 5)
1 Dist1
2 p←'Dist1'⎕PDist 'Samples' 2
3 0.0894990607 0.1072925687 0.2866018256 0.516606545 0.161740018 0.1565499979 0.1884846226 0.4932253615
4 4 4
5

```

Discrete-Uniform

Syntax: '#⎕PDist 'Create' instName rndSrcName rndSeedName 'Discrete-Uniform' lower upper

lower: Int32: lower ≤ upper

upper: Int32: lower ≤ upper

```

'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Discrete-Uniform' 5 22
'Dist1'⎕PDist 'Samples' 2

```

```

0 '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Discrete-Uniform' 5 22
1 Dist1
2 'Dist1'⎕PDist 'Samples' 2
3 18 13
4

```

Erlang

Syntax: '#⎕PDist 'Create' instName rndSrcName rndSeedName 'Erlang' shape rate

shape: Double: 0 ≤ shape

rate: Double: 0 ≤ rate

```

'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Erlang' 5 1.2
'Dist1'⎕PDist 'Samples' 2

```

```

0 '# PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Erlang' 5 1.2
1 Dist1
2 'Dist1' PDist 'Samples' 2
3 3.03250138 4.022998621
4

```

Exponential

Syntax: '# PDist 'Create' instName rndSrcName rndSeedName 'Exponential' rate

rate: Double: $0 \leq \text{rate}$

```

'# PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Exponential' 0.05
'Dist1' PDist 'Samples' 2

```

```

0 '# PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Exponential' 0.05
1 Dist1
2 'Dist1' PDist 'Samples' 2
3 10.68273676 23.57557645
4

```

Fisher-Snedecor

Syntax: '# PDist 'Create' instName rndSrcName rndSeedName 'Fisher-Snedecor' d1 d2

d1: 1st degree of freedom: Double: $0 < d1$

d2: 2nd degree of freedom: Double: $0 < d2$

```

'# PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Fisher-Snedecor' 4.5 6.7
'Dist1' PDist 'Samples' 2

```

```

0 '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Fisher-SNedecor' 4.5 6.7
1 Dist1
2 'Dist1'⎕PDist 'Samples' 2
3 0.8224079573 0.428716937
4

```

Gamma

Syntax: '#⎕PDist 'Create' instName rndSrcName rndSeedName 'Gamma' shape rate

shape: Double: $0 \leq \text{shape}$

rate: Double: $0 \leq \text{rate}$

```

'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Gamma' 45.5 2.3
'Dist1'⎕PDist 'Samples' 2

```

```

0 '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Gamma' 45.5 2.3
1 Dist1
2 'Dist1'⎕PDist 'Samples' 2
3 25.27538499 14.25252853
4

```

Geometric

Syntax: '#⎕PDist 'Create' instName rndSrcName rndSeedName 'Geometric' prob1

prob1: Double: Probability of a success for a trial

```

'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Geometric' 0.5
'Dist1'⎕PDist 'Samples' 2

```

```

0 '#␣PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Geometric' 0.5
1 Dist1
2 'Dist1'␣PDist 'Samples' 2
3 4 3
4

```

Ready | Hist: Ln: 4 Col: 6 | Ins | Classic | Num | EN_US

Hyper-Geometric

Syntax: '#␣PDist 'Create' instName rndSrcName rndSeedName 'Hyper-Geometric' p s d

p: Int32: Population size: $0 < p$

s: Int32: #successes in the population: $0 < s$

d: Int32: #draws without replacement: $0 < d$

```

'#␣PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Hyper-Geometric' 20000 15000 100
'Dist1'␣PDist 'Samples' 2

```

```

0 '#␣PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Hyper-Geometric' 20000 15000 100
1 Dist1
2 'Dist1'␣PDist 'Samples' 2
3 75 71
4

```

Ready | Hist: Ln: 4 Col: 6 | Ins | Classic | Num | EN_US

Inverse-Gamma

Syntax: '#␣PDist 'Create' instName rndSrcName rndSeedName 'Inverse-Gamma' shape scale

shape: Double: $0 < \text{shape}$

scale: Double: $0 < \text{scale}$

```

'#␣PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Inverse-Gamma' 2.1 1.5
'Dist1'␣PDist 'Samples' 2

```

```

0 '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Inverse-Gamma' 2.1 1.5
1 Dist1
2 'Dist1'⎕PDist 'Samples' 2
3 0.4281692615 0.8502313942

```

InverseWishart

Syntax : '#⎕PDist 'Create' instName rndSrcName rndSeedName 'InverseWishart' df scaleMat

df: Int: degrees of freedom

scaleMat: real, positive-definite matrix

```

'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'InverseWishart' 4 (2 2p5 2 2 10)
ρ←'Dist1'⎕PDist 'Samples' 2

```

```

0 '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'InverseWishart' 4 (2 2p5 2 2 10)
1 Dist1
2 ρ←'Dist1'⎕PDist 'Samples' 2
3 6.522558703 5.519244761 0.5920042445 -0.479569263
4 5.519244761 7.619724218 -0.479569263 2.323699743
5 2 2 2 2
6

```

Laplace

Syntax: '#⎕PDist 'Create' instName rndSrcName rndSeedName 'Laplace' location scale

location: Double

scale: Double: 0<scale

```

'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' '42' 'Laplace' -2.1 1.5
'Dist1'⎕PDist 'Samples' 2

```

```

0 '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' '42' 'Laplace' ~2.1 1.5
1 Dist1
2 'Dist1'⎕PDist 'Samples' 2
3 ~2.5333637803346908 ~0.75126998936434352
4

```

Log-Normal

Syntax: '#⎕PDist 'Create' instName rndSrcName rndSeedName 'Log-Normal' mu sigma

mu: Double: Log-scale of the distribution

sigma: Double : Shape of the distribution: $0 \leq \sigma$

```

'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' '42' 'Log-Normal' ~2.1 1.5
'Dist1'⎕PDist 'Samples' 2

```

```

0 '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' '42' 'Log-Normal' ~2.1 1.5
1 Dist1
2 'Dist1'⎕PDist 'Samples' 2
3 0.056390969546584856 0.76558259423638656
4

```

MatrixNormal

Syntax: '#⎕PDist 'Create' instName rndSrcName rndSeedName 'MatrixNormal' mMat rcMat ccMat

mMat: double[n;p]: mean matrix

rcMat : double[n;n]: row covariance matrix

ccMat: double[p;p]: column covariance matrix

```

mMat←2 3⍲6
rcMat←2 2⍲1 0 0 1
ccMat←3 3⍲1 0 0 0 1 0 0 0 1
'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'MatrixNormal' mMat rcMat ccMat
ρ←'Dist1'⎕PDist 'Samples' 2

```

```

APL64: CLEAR WS
File Edit Session Objects Tools Options Help
mMat←2 3p16
rcMat←2 2p1 0 0 1
ccMat←3 3p1 0 0 0 1 0 0 0 1
'#'PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'MatrixNormal' mMat rcMat ccMat
Dist1
p''←'Dist1'PDist 'Samples' 2
2.283545405 2.439907554 3.219974263 0.9035022134 2.14319658 3.31239888
1.977668308 3.957937238 7.398773719 3.034878542 2.240913013 5.354389775
2 3 2 3

```

Multinomial

Syntax: '#'PDist 'Create' instName rndSrcName rndSeedName 'Multinomial' pRatios nTrials

pRatios: Double[]: probability ratios

nTrials: Int: #Trials

```

'#'PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Multinomial' (1 2 3 4) 10
p''←'Dist1'PDist 'Samples' 3

```

```

APL64: CLEAR WS
File Edit Session Objects Tools Options Help
'#'PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Multinomial' (1 2 3 4) 10
Dist1
p''←'Dist1'PDist 'Samples' 3
2 2 1 5 1 3 4 2 2 2 5 1
4 4 4 4

```

Negative-Binomial

Syntax: '#'PDist 'Create' instName rndSrcName rndSeedName 'Negative-Binomial' s p

s: Double: #successes required to stop the experiment: 0≤s

p: Double: probability of a trial resulting in success in range [0,1]

```

'#'PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Negative-Binomial' 5 .5
'Dist1'PDist 'Samples' 2

```

```

0 '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Negative-Binomial' 5 .5
1 Dist1
2 'Dist1'⎕PDist 'Samples' 2
3 6 10

```

Normal

Syntax: '#⎕PDist 'Create' instName rndSrcName rndSeedName 'Normal' m s

m: Double: Mean

s: Double: Standard deviation: $0 \leq s$

```

'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Normal' 0 1
'Dist1'⎕PDist 'Samples' 2

```

```

0 '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Normal' 0 1
1 Dist1
2 'Dist1'⎕PDist 'Samples' 2
3 -1.141441033 -0.495034191

```

NormalGamma

Syntax: '#⎕PDist 'Create' instName rndSrcName rndSeedName 'NormalGamma' MI Ms Ps Ips

MI: double: mean location

Ms: double: mean scale

Ps: double: precision shape

Ips: double: precision inverse scale

'Dist1'⎕PDist 'Samples' 2

```

'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'NormalGamma' 0 10 2 2
'Dist1'⎕PDist 'Samples' 2

```

```

0 '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'NormalGamma' 0 10 2 2
1 Dist1
2 'Dist1'⎕PDist 'Samples' 2
3 ~0.4567849444 0.6548883378 ~0.8361690849 0.2146275654
4

```

Ready | Hist: Ln: 4 Col: 6 | Ins | Classic | EN_US

Pareto

Syntax: '#⎕PDist 'Create' instName rndSrcName rndSeedName 'Pareto' scale shape

scale: Double: 0<scale

shape: Double : 0<shape

```

'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Pareto' 10.3 4.5
'Dist1'⎕PDist 'Samples' 2

```

```

0 '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Pareto' 10.3 4.5
1 Dist1
2 'Dist1'⎕PDist 'Samples' 2
3 17.85278372 10.35889363
4

```

Ready | Hist: Ln: 4 Col: 6 | Ins | Classic | Num | EN_US

Poisson

Syntax: '#⎕PDist 'Create' instName rndSrcName rndSeedName 'Poisson' lambda

lambda: Double: 0<lambda

```

'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Poisson' 45.7
'Dist1'⎕PDist 'Samples' 2

```

```

0 '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Poisson' 45.7
1 Dist1
2 'Dist1'⎕PDist 'Samples' 2
3 43 50
4

```

Rayleigh

Syntax: '#⎕PDist 'Create' instName rndSrcName rndSeedName 'Rayleigh' scale loc

scale: Double: $0 < \text{scale}$

loc: Double: Location:

```

'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Rayleigh' 100 23
'Dist1'⎕PDist 'Samples' 2

```

```

0 '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Rayleigh' 100 23
1 Dist1
2 'Dist1'⎕PDist 'Samples' 2
3 155.5591716 16.59329991
4

```

Stable

Syntax: '#⎕PDist 'Create' instName rndSrcName rndSeedName 'Stable' a b scale loc

a: Double: In range: $(0, 2]$

b: Double: In range: $[-1, 1]$

scale: Double: $0 < \text{scale}$

loc: Double: Location:

```

'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Stable' 1.7 0.5 23 10
'Dist1'⎕PDist 'Samples' 2

```

```

0      '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Stable' 1.7 0.5 23 10
1 Dist1
2      'Dist1'⎕PDist 'Samples' 2
3 72.21201943 -24.05582306
4

```

Ready | Hist: Ln: 4 Col: 6 | Ins | Classic | Num | EN_US

Student-T

Syntax: '#⎕PDist 'Create' instName rndSrcName rndSeedName 'Student-T' loc scale freedom

loc: Double: Location

scale: Double: 0<scale

freedom: Double: Degrees of freedom : 0<freedom

```

'#⎕PDist 'Create' 'ST1' 'MERSENNETWISTER' 'Robust' 'Student-T' 1.2 2.3 1.2
'ST1'⎕PDist 'Samples' 2

```

```

0      '#⎕PDist 'Create' 'ST1' 'MERSENNETWISTER' 'Robust' 'Student-T' 1.2 2.3 1.2
1 ST1
2      'ST1'⎕PDist 'Samples' 2
3 -0.1312188174 0.9498274857
4

```

Ready | Hist: Ln: 4 Col: 6 | Ins | Classic | Num | EN_US

Triangular

Syntax: '#⎕PDist 'Create' instName rndSrcName rndSeedName 'Triangular' lower upper mode

lower: Double: lower≤ mode ≤ upper

upper: Double:

mode: Double:

```

'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Triangular' 5.3 11.6 10.1
'Dist1'⎕PDist 'Samples' 2

```

```

0 | '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Triangular' 5.3 11.6 10.1
1 | Dist1
2 | 'Dist1'⎕PDist 'Samples' 2
3 | 10.00308735 9.837180267
4 |

```

Weibull

Syntax: '#⎕PDist 'Create' instName rndSrcName rndSeedName 'Weibull' scale shape

scale: Double: 0<scale

shape: Double: 0<shape

```

'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Weibull' 10.5 2.4
'Dist1'⎕PDist 'Samples' 2

```

```

0 | '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Weibull' 10.5 2.4
1 | Dist1
2 | 'Dist1'⎕PDist 'Samples' 2
3 | 8.976851153 7.977554739
4 |

```

Wishart

Syntax: '#⎕PDist 'Create' 'Create' instName rndSrcName rndSeedName 'Wishart' df scaleMat

df: Int: degrees of freedom

scaleMat: matrix

```

'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Wishart' 4 (2 2p5 2 2 10)
'Dist1'⎕PDist 'Samples' 2

```

```

0 '# PDIST 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Wishart' 4 (2 2p5 2 2 10)
1 Dist1
2 'Dist1' PDIST 'Samples' 2
3 11.53249402 15.43683085 6.057905577 -0.08302224777
4 15.43683085 35.57122659 -0.08302224777 39.74904875
5

```

Zipf

Syntax: '# PDIST 'Create' instName rndSrcName rndSeedName 'Zipf' s n

s: Double

n: Int32

```

'# PDIST 'Create' 'Dist1' 'MERSENNETWISTER' 'Robust' 'Zipf' 3.3 4
'Dist1' PDIST 'Samples' 2

```

```

0 '# PDIST 'Create' 'Dist1' 'MERSENNETWISTER' 'Robust' 'Zipf' 3.3 4
1 Dist1
2 'Dist1' PDIST 'Samples' 2
3 2 1
4

```

Delete

Delete an existing PDIST instance, by specifying its instance name.

```

'# PDIST 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
'# PDIST 'Create' 'Normal2' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 25 2
'# PDIST 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
'# PDIST Instances'
'# PDIST 'Delete' 'Normal2'
'# PDIST Instances'

```

```

APL64: CLEAR WS
File Edit Session Objects Tools Options Help
0 '#[]PDist 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
1 Normal1
2 '#[]PDist 'Create' 'Normal2' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 25 2
3 Normal2
4 '#[]PDist 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
5 Bernoulli1
6 '#[]PDist 'Instances'
7 Normal1 Normal2 Bernoulli1
8 '#[]PDist 'Delete' 'Normal2'
9 '#[]PDist 'Instances'
10 Normal1 Bernoulli1
11
Ready | Hist: Ln: 0 Col: 0 Ins | Classic | Num | EN_US

```

Help (?)

Return a summary of the []PDIST system function documentation.

```
'#[]PDist '?'
```

```

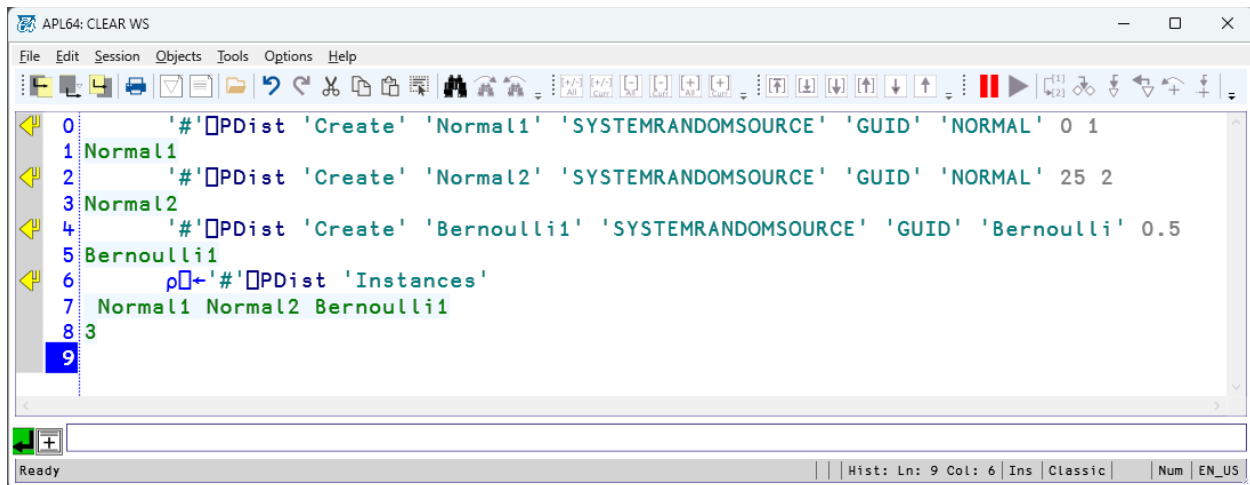
APL64: CLEAR WS
File Edit Session Objects Tools Options Help
0 '# []Pdist 'help'
1 []PDist Documentation
2 []PDist Object Actions:
3     '# []PDist 'Clear'
4 Int32 ← '# []PDist 'Count'
5 instName ← '# []PDist 'Create' instName rndSrcName rndSeedName distName param1 ...
6     '# []PDist 'Delete' instName
7 text ← '# []PDist 'Help'
8 text ← '# []PDist '?'
9 text[][] ← '# []PDist 'Instances'
10 instName ← '# []PDist 'New' instName rndSrcName rndSeedName distName param1 ...
11
12 []PDist Instance Actions
13 ns ← instName []PDist 'CumDist' ns
14 ns ← instName []PDist 'Density' ns
15 ns ← instName []PDist 'DensityLn' ns
16 ns ← instName []PDist 'InvCumDist' ns
17 ns ← instName []PDist 'Probability' ns
18 ns ← instName []PDist 'ProbabilityLn' ns
19 rank2Array ← instName []PDist 'Properties'
20 (rndSrcNm rndSeedNm) ← instName []PDist instanceName [rndSrcNm rndSeedNm]
21 ns[] ← instName []PDist 'Samples' #samples
22 instName ← instName []PDist 'Self'
23
24 Notes:
25 An exception will be thrown is the action does not apply to the distribution
26 ns: Numeric value: Int32 or Double, scalar or vector depending on the distribution
27
Ready | Hist: Ln: 27 Col: 6 Ins | Classic | EN_US

```

Instances

Return a rank-1 array of the names of the existing ☐PDIST instances.

```
'#'☐PDist 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
'#'☐PDist 'Create' 'Normal2' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 25 2
'#'☐PDist 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
ρ<←'#'☐PDist 'Instances'
```



The screenshot shows the APL64: CLEAR WS interface. The code editor contains the following code:

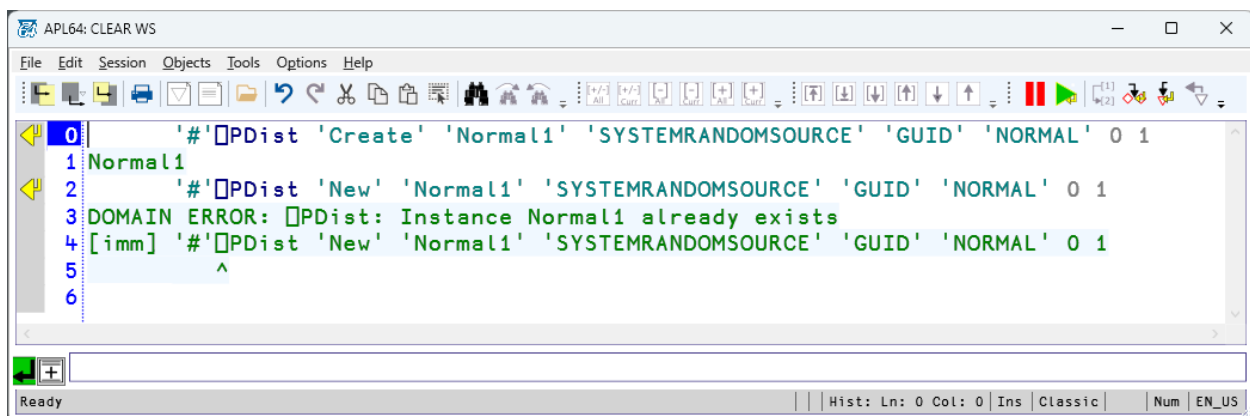
```
0 '#'☐PDist 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
1 Normal1
2 '#'☐PDist 'Create' 'Normal2' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 25 2
3 Normal2
4 '#'☐PDist 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
5 Bernoulli1
6 ρ<←'#'☐PDist 'Instances'
7 Normal1 Normal2 Bernoulli1
8 3
9
```

The output window shows the results of the execution, which are the names of the created instances: Normal1, Normal2, and Bernoulli1. The status bar at the bottom indicates "Ready" and "Hist: Ln: 9 Col: 6 Ins Classic Num EN_US".

New

Create a new ☐PDIST instance. The ☐PDIST New action is analogous to the ☐PDIST Create action, except that an existing ☐PDIST cannot be overridden by the ☐PDIST New action.

```
'#'☐PDist 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
'#'☐PDist 'New' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
```



The screenshot shows the APL64: CLEAR WS interface. The code editor contains the following code:

```
0 '#'☐PDist 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
1 Normal1
2 '#'☐PDist 'New' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
3 DOMAIN ERROR: ☐PDist: Instance Normal1 already exists
4 [imm] '#'☐PDist 'New' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
5 ^
6
```

The output window shows the error message: "DOMAIN ERROR: ☐PDist: Instance Normal1 already exists". The status bar at the bottom indicates "Ready" and "Hist: Ln: 0 Col: 0 Ins Classic Num EN_US".

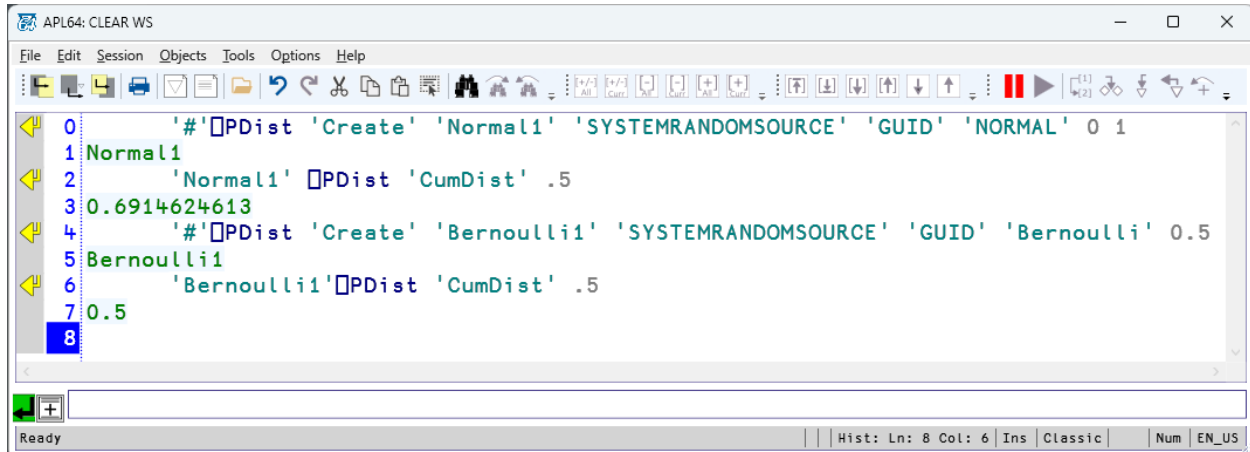
☐PDIST Instance Actions

Some ☐PDIST instance actions may not apply to some probability distributions, in which case an exception will be thrown.

CumDist

The [cumulative distribution function](#) represents the probability that the random variable, satisfying the specified probability distribution, will have a value less than or equal to a specified value.

```
'#[]PDist 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
'Normal1' []PDist 'CumDist' .5
'#[]PDist 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
'Bernoulli1' []PDist 'CumDist' .5
```



The screenshot shows the APL64: CLEAR WS window with the following code and output:

```
0      '#[]PDist 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
1 Normal1
2      'Normal1' []PDist 'CumDist' .5
3 0.6914624613
4      '#[]PDist 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
5 Bernoulli1
6      'Bernoulli1' []PDist 'CumDist' .5
7 0.5
8
```

The status bar at the bottom indicates: Ready | Hist: Ln: 8 Col: 6 | Ins | Classic | Num | EN_US

Density

The [probability density function](#) provides the relative likelihood that a random variable, satisfying the specified probability distribution, will have a value equal to a specified value.

Most Distributions

For most distributions, the argument is a scalar double. The result is a scalar double.

```
'#[]PDist 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
'Normal1' []PDist 'Density' .5
'#[]PDist 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
'Bernoulli1' []PDist 'Density' .5
```

```

APL64: CLEAR WS
File Edit Session Objects Tools Options Help
0 '#␣PDist 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
1 Normal1
2 'Normal1' ␣PDist 'Density' .5
3 0.3520653268
4 '#␣PDist 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
5 Bernoulli1
6 'Bernoulli1'␣PDist 'Density' .5
7 DOMAIN ERROR: ␣PDist: Distribution Type: Bernoulli: Density action not applicable
8 [imm] 'Bernoulli1'␣PDist 'Density' .5
9 ^
10

```

For certain multi-variate distributions, the arguments and results depend on the distribution type:

Dirichlet

The argument is a vector of doubles. The total of the argument must equal to 1. The result is a scalar double.

```

'#␣PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Dirichlet' (1 2 3.4 5)
'Dist1'␣PDist 'Density' (.1 .2 .3 .4)

```

```

APL64: CLEAR WS
File Edit Session Objects Tools Options Help
0 '#␣PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Dirichlet' (1 2 3.4 5)
1 Dist1
2 'Dist1'␣PDist 'Density' (.1 .2 .3 .4)
3 37.26724226
4

```

Inverse Wishart

The argument is a rank-2 matrix of doubles. The result is a scalar double.

```

'#␣PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'InverseWishart' 4 (2 2p5 2 2 10)
'Dist1'␣PDist 'Density' (2 2p1 0 0 1)

```

```

APL64: CLEAR WS
File Edit Session Objects Tools Options Help
0 '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'InverseWishart' 4 (2 2p5 2 2 10)
1 Dist1
2 'Dist1'⎕PDist 'Density' (2 2p1 0 0 1)
3 0.04656581296
4

```

NormalGamma

mean is a scalar double

precision is a scalar double

```

'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'NormalGamma' 0 10 2 2
mean←4.25
precision←0.5
'Dist1'⎕PDist 'Density' mean precision

```

```

APL64: CLEAR WS
File Edit Session Objects Tools Options Help
0 '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'NormalGamma' 0 10 2 2
1 Dist1
2 mean←4.25
3 precision←0.5
4 'Dist1'⎕PDist 'Density' mean precision
5 1.607016675E-20
6

```

Wishart

The argument is a rank-2 matrix of doubles. The result is a scalar double.

```

'#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Wishart' 4 (2 2p5 2 2 10)
'Dist1'⎕PDist 'Density' (2 2p1 0 0 1)

```

```

APL64: CLEAR WS
File Edit Session Objects Tools Options Help
0 '#⎕PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Wishart' 4 (2 2p5 2 2 10)
1 Dist1
2 'Dist1'⎕PDist 'Density' (2 2p1 0 0 1)
3 0.00001597480605
4

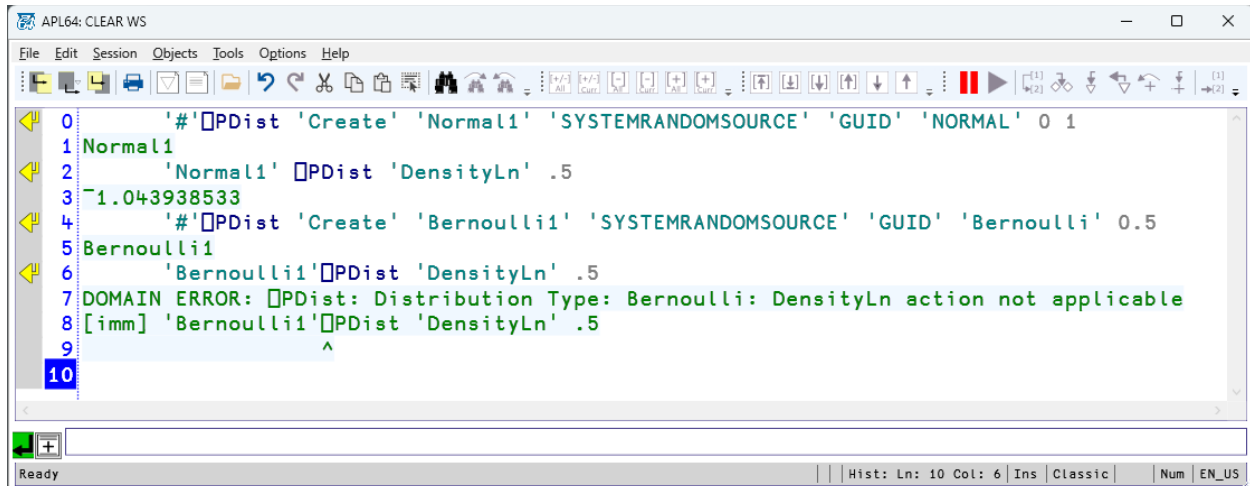
```

DensityLn

Natural logarithm of the density function value.

The arguments and results are the same as those for the Density action.

```
'#[]PDist 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
'Normal1' []PDist 'DensityLn' .5
'#[]PDist 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
'Bernoulli1' []PDist 'DensityLn' .5
```

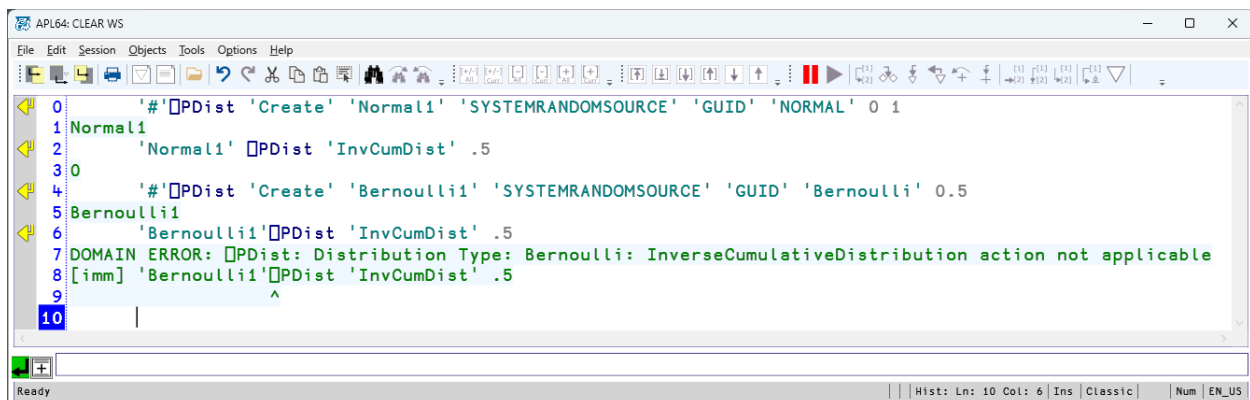


```
0 '#[]PDist 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
1 Normal1
2 'Normal1' []PDist 'DensityLn' .5
3 -1.043938533
4 '#[]PDist 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
5 Bernoulli1
6 'Bernoulli1' []PDist 'DensityLn' .5
7 DOMAIN ERROR: []PDist: Distribution Type: Bernoulli: DensityLn action not applicable
8 [imm] 'Bernoulli1' []PDist 'DensityLn' .5
9
10
```

InvCumDist

The inverse cumulative density distribution or [quantile](#) returns the value of a random variable, satisfying the specified probability distribution, which has the probability less than or equal to a specified probability.

```
'#[]PDist 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
'Normal1' []PDist 'InvCumDist' .5
'#[]PDist 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
'Bernoulli1' []PDist 'InvCumDist' .5
```



```
0 '#[]PDist 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
1 Normal1
2 'Normal1' []PDist 'InvCumDist' .5
3 0
4 '#[]PDist 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
5 Bernoulli1
6 'Bernoulli1' []PDist 'InvCumDist' .5
7 DOMAIN ERROR: []PDist: Distribution Type: Bernoulli: InverseCumulativeDistribution action not applicable
8 [imm] 'Bernoulli1' []PDist 'InvCumDist' .5
9
10
```

Probability

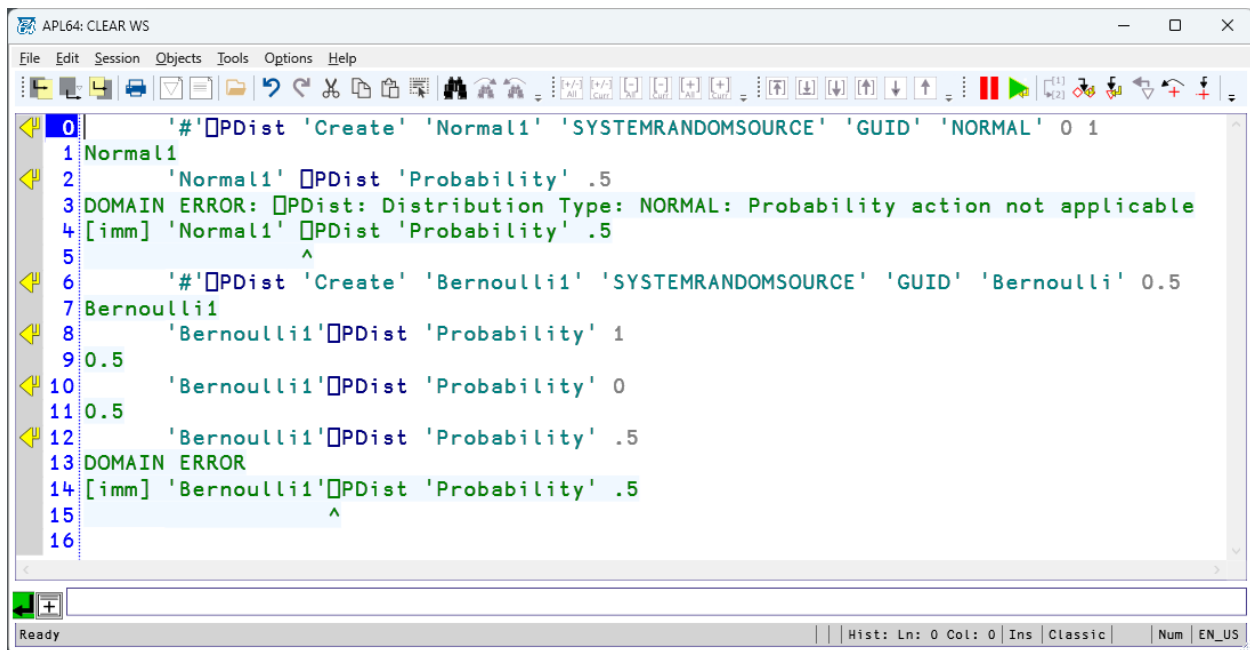
Returns the [probability](#) that random variable, satisfying the specified probability distribution, will be equal to a specified value.

Most Distributions

For continuous distributions the argument is a scalar double. For discrete distributions the argument is a scalar integer. The result is a scalar double.

Bernoulli distribution

```
'#[]PDist 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
'Normal1' []PDist 'Probability' .5
'#[]PDist 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
'Bernoulli1' []PDist 'Probability' 1
'Bernoulli1' []PDist 'Probability' 0
'Bernoulli1' []PDist 'Probability' .5
```



Multinomial distribution

```
'#[]PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Multinomial' (.2 .8) 3
'Dist1' []PDist 'Probability' (3 0)
'Dist1' []PDist 'Probability' (2 1)
'Dist1' []PDist 'Probability' (1 2)
'Dist1' []PDist 'Probability' (0 3)
```

```

APL64: CLEAR WS
File Edit Session Objects Tools Options Help
0 '#␣PDist 'Create' 'Dist1' 'MersenneTwister' 'Robust' 'Multinomial' (.2 .8) 3
1 Dist1
2 'Dist1'␣PDist 'Probability' (3 0)
3 0.008
4 'Dist1'␣PDist 'Probability' (2 1)
5 0.096
6 'Dist1'␣PDist 'Probability' (1 2)
7 0.384
8 'Dist1'␣PDist 'Probability' (0 3)
9 0.512
10

```

Ready | Hist: Ln: 10 Col: 6 | Ins | Classic | EN_US

ProbabilityLn

Natural logarithm of the probability value.

The arguments and results are the same as those for the Probability action.

```

'#␣PDist 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
'Normal1'␣PDist 'ProbabilityLn' .5
'#␣PDist 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
'Bernoulli1'␣PDist 'ProbabilityLn' 1
'Bernoulli1'␣PDist 'ProbabilityLn' 0
'Bernoulli1'␣PDist 'ProbabilityLn' .5

```

```

APL64: CLEAR WS
File Edit Session Objects Tools Options Help
0 '#␣PDist 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
1 Normal1
2 'Normal1'␣PDist 'ProbabilityLn' .5
3 DOMAIN ERROR: ␣PDist: Distribution Type: NORMAL: ProbabilityLn action not applicable
4 [imm] 'Normal1'␣PDist 'ProbabilityLn' .5
5
6 '#␣PDist 'Create' 'Bernoulli1' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
7 Bernoulli1
8 'Bernoulli1'␣PDist 'ProbabilityLn' 1
9 -0.6931471806
10 'Bernoulli1'␣PDist 'ProbabilityLn' 0
11 -0.6931471806
12 'Bernoulli1'␣PDist 'ProbabilityLn' .5
13 DOMAIN ERROR
14 [imm] 'Bernoulli1'␣PDist 'ProbabilityLn' .5
15
16

```

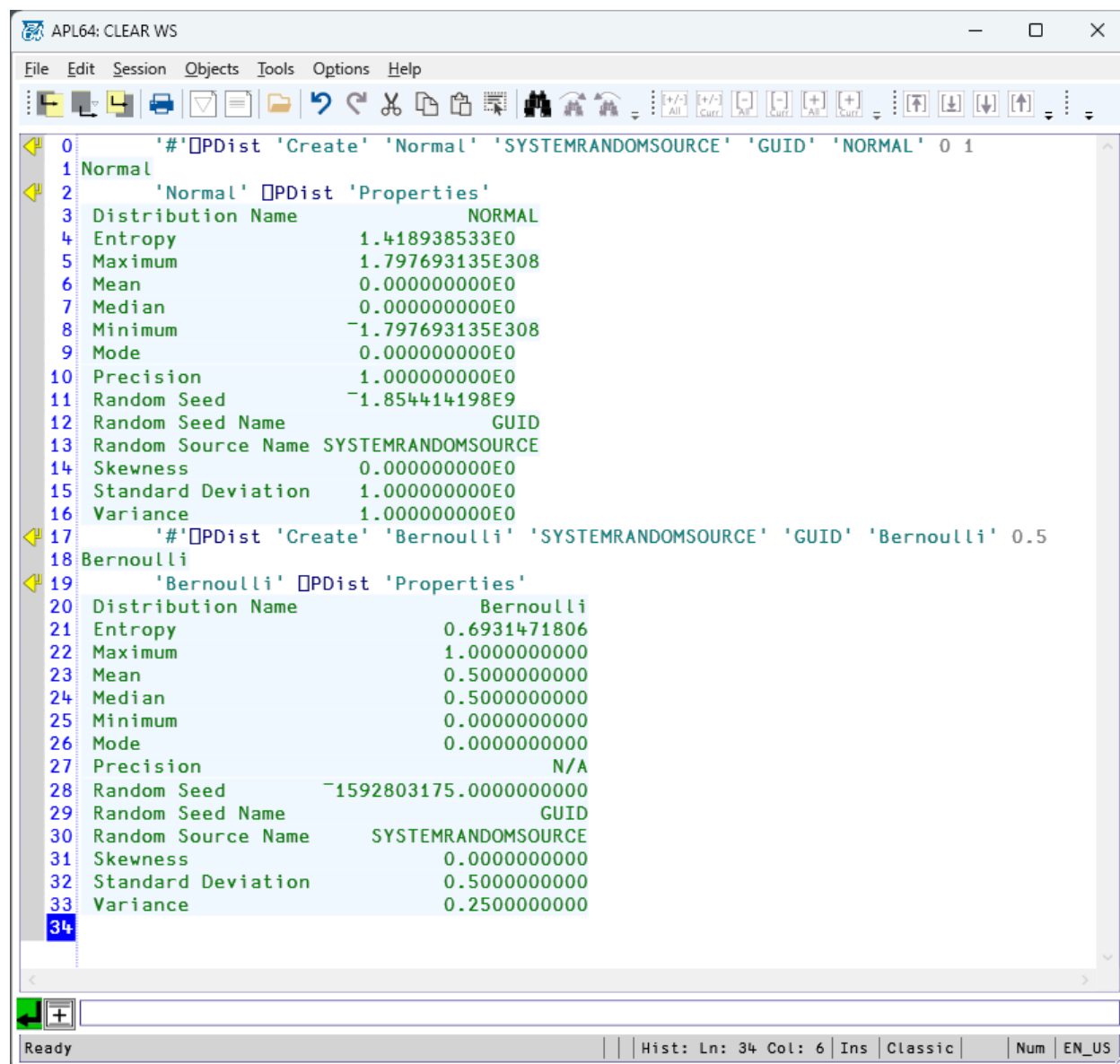
Ready | Hist: Ln: 16 Col: 6 | Ins | Classic | Num | EN_US

Properties

Returns a rank-2 array with the available properties of the selected distribution which are supported by the MathNet toolkit. Column #1 contains the property description. Column #2 contains the property value.

Some distributions do not support all properties listed.

```
'#'[ ]PDist 'Create' 'Normal' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
'Normal' [ ]PDist 'Properties'
'#'[ ]PDist 'Create' 'Bernoulli' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
'Bernoulli' [ ]PDist 'Properties'
```



```
0 '#'[ ]PDist 'Create' 'Normal' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
1 Normal
2 'Normal' [ ]PDist 'Properties'
3 Distribution Name      NORMAL
4 Entropy                1.418938533E0
5 Maximum                1.797693135E308
6 Mean                  0.000000000E0
7 Median                0.000000000E0
8 Minimum               -1.797693135E308
9 Mode                  0.000000000E0
10 Precision            1.000000000E0
11 Random Seed          -1.854414198E9
12 Random Seed Name     GUID
13 Random Source Name   SYSTEMRANDOMSOURCE
14 Skewness              0.000000000E0
15 Standard Deviation   1.000000000E0
16 Variance              1.000000000E0
17 '#'[ ]PDist 'Create' 'Bernoulli' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
18 Bernoulli
19 'Bernoulli' [ ]PDist 'Properties'
20 Distribution Name     Bernoulli
21 Entropy               0.6931471806
22 Maximum              1.0000000000
23 Mean                 0.5000000000
24 Median               0.5000000000
25 Minimum              0.0000000000
26 Mode                 0.0000000000
27 Precision            N/A
28 Random Seed          -1592803175.0000000000
29 Random Seed Name     GUID
30 Random Source Name   SYSTEMRANDOMSOURCE
31 Skewness              0.0000000000
32 Standard Deviation   0.5000000000
33 Variance              0.2500000000
34
```

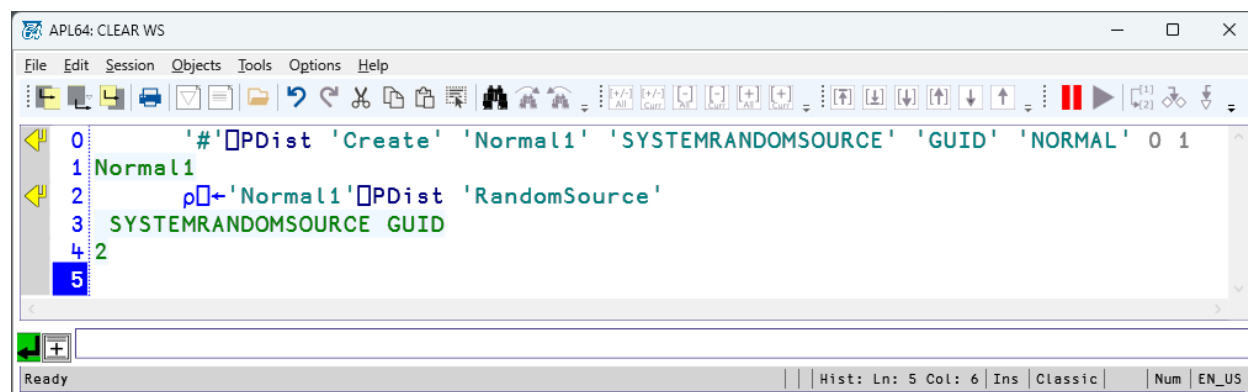
RandomSource

Use this ☐PDIST instance method to view or change the random seed and pseudo-random number generator for an existing ☐PDIST instance.

To obtain the current random source name and random seed name for a specified ☐PDIST distribution instance. The result of this format of the RandomSource action is a rank-2 array containing the existing random source name and the random seed name.

Syntax: (rndSrcNm rndSeedNm)←instName'☐PDist instanceName

```
'#'☐PDist 'Create' 'Normal1' 'SYSTEMRANDOMSOURCE' 'GUID' 'NORMAL' 0 1
ρ☐←'Normal1'☐PDist 'RandomSource'
```



To set the random source name and random seed name for a specified ☐PDIST distribution instance. The result of this format of the RandomSource action is a rank-2 array containing the existing random source name and the random seed name.

Syntax: (rndSrcNm rndSeedNm)←instName☐PDist instanceName rndSrcNm rndSeedNm

'rndSrcNm' is selected from the [pseudo-random number generators](#) supported by MathNet.

Pseudo-random Number Generators: Source Names:

Random Source Name	Description
SystemRandomSource	Wraps the .NET System.Random to provide thread-safety
MersenneTwister	Mersenne Twister 19937 generator
XorShift	Multiply-with-carry XOR-shift generator
Mcg31M1	Multiplicative congruential generator using a modulus of $2^{31}-1$ and a multiplier of 1132489760
Mcg59	Multiplicative congruential generator using a modulus of 2^{59} and a multiplier of 13^{13}
Wh1982	Wichmann-Hill's 1982 combined multiplicative congruential generator
WH2006	Wichmann-Hill's 2006 combined multiplicative congruential generator
Mrg32K3A	32-bit combined multiple recursive generator with 2 components of order 3
Palf	Parallel Additive Lagged Fibonacci generator
XOSHIRO256STARSTAR	xoshiro256** random number generator
CryptoRandomSource	Wraps the .NET RNGCryptoServiceProvider

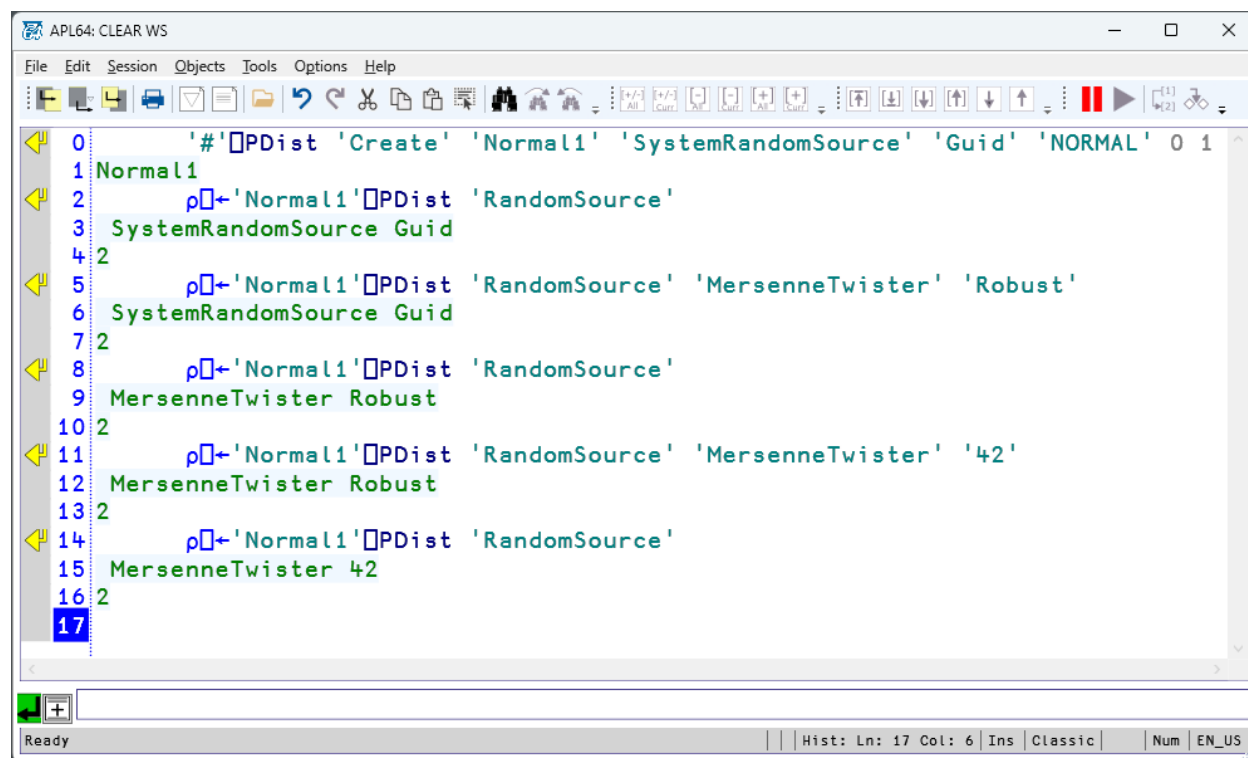
'rndSeeNm' is selected from the [random seed](#) options supported by Mathnet. The 'numericValue' random seed option is useful for testing an application using ☐ PDist, but this option should not be used for a production application.

Random Seed Names:

These are the [random seed options supported by Mathnet](#):

Random Seed Name or Explicit Value
Time
Guid
Robust
'numericValue'

```
#☐ PDist 'Create' 'Normal1' 'SystemRandomSource' 'Guid' 'NORMAL' 0 1
ρ☐ ← 'Normal1' ☐ PDist 'RandomSource'
ρ☐ ← 'Normal1' ☐ PDist 'RandomSource' 'MersenneTwister' 'Robust'
ρ☐ ← 'Normal1' ☐ PDist 'RandomSource'
ρ☐ ← 'Normal1' ☐ PDist 'RandomSource' 'MersenneTwister' '42'
ρ☐ ← 'Normal1' ☐ PDist 'RandomSource'
```



Samples

Use this `▢PDIST` instance method obtain an APL64 rank-1 array of sample values of the specified `▢PDIST` probability distribution instance. The result of the `Samples` method depends on the distribution type.

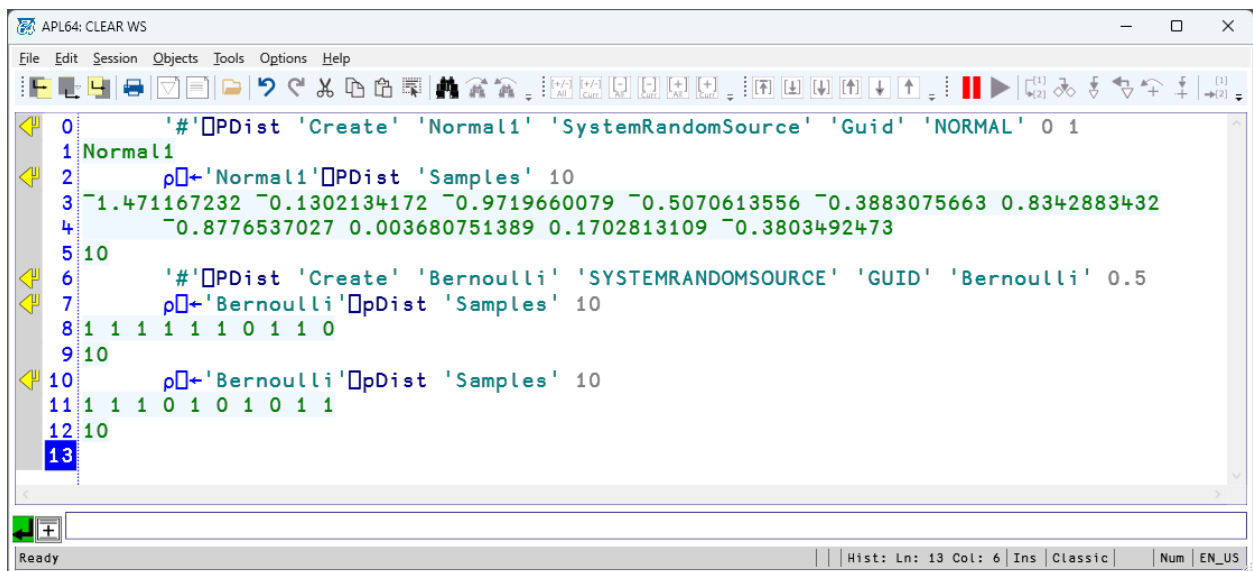
Review the documentation of the `▢PDIST` [Create action](#) for examples using the `▢PDIST` `Samples` action.

Most Distributions

Syntax: `double[] ← 'Normal1'▢PDist 'Samples' int`

For most distributions the argument is a scalar integer indicating the desired number of samples in the result. For continuous distributions the result is a vector of doubles. For discrete distributions the result is a vector of integers.

```
'#▢PDist 'Create' 'Normal1' 'SystemRandomSource' 'Guid' 'NORMAL' 0 1
ρ▢←'Normal1'▢PDist 'Samples' 10
'#▢PDist 'Create' 'Bernoulli' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
ρ▢←'Bernoulli'▢pDist 'Samples' 10
ρ▢←'Bernoulli'▢pDist 'Samples' 10
```



```
0 '#▢PDist 'Create' 'Normal1' 'SystemRandomSource' 'Guid' 'NORMAL' 0 1
1 Normal1
2 ρ▢←'Normal1'▢PDist 'Samples' 10
3 -1.471167232 -0.1302134172 -0.9719660079 -0.5070613556 -0.3883075663 0.8342883432
4 -0.8776537027 0.003680751389 0.1702813109 -0.3803492473
5 10
6 '#▢PDist 'Create' 'Bernoulli' 'SYSTEMRANDOMSOURCE' 'GUID' 'Bernoulli' 0.5
7 ρ▢←'Bernoulli'▢pDist 'Samples' 10
8 1 1 1 1 1 1 0 1 1 0
9 10
10 ρ▢←'Bernoulli'▢pDist 'Samples' 10
11 1 1 1 0 1 0 1 0 1 1
12 10
13
```

Dirichlet

Syntax: `(double[] double[] ...) ← 'Normal1'▢PDist 'Samples' int`

The argument is a scalar integer. The result is a vector of `double[]` vectors.

InverseWishart, MatrixNormal, Wishart

Syntax: `(double[;] double[;] ...) ← 'Normal1'▢PDist 'Samples' int`

The argument is a scalar integer. The result is a vector of rank-2 matrices.

MultiNomial

Syntax: (Int32[] Int32[] ...) ← 'Normal1'␣PDist 'Samples' int

The argument is a scalar integer. The result is a vector of Int32[] vectors.

NormalGamma

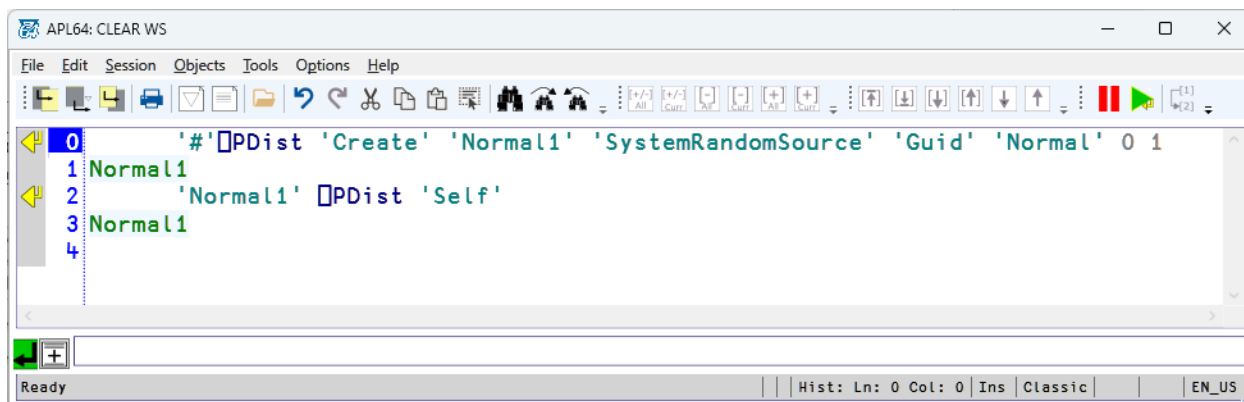
Syntax: (double[2] double[2] ...) ← 'Normal1'␣PDist 'Samples' int

The argument is a scalar integer. The result is a vector of 2-element double vectors. Each double vector contains the mean and precision of the sample.

Self

Use this ␣PDIST instance method to obtain the instance name of a selected ␣PDIST probability distribution instance.

```
#'␣PDist 'Create' 'Normal1' 'SystemRandomSource' 'Guid' 'Normal' 0 1  
'Normal1' ␣PDist 'Self'
```



␣PDISTSELF System Variable

When the ␣PDistSelf system variable is a ␣Pdist instance, the left argument of ␣Pdist is not necessary. ␣PdistSelf can be localized in an APL64 programmer defined function.

Example:

Create the FN1 function, which localizes ␣PdistSelf. Running the FN1 function will not modify the workspace value of ␣PdistSelf:

```
␣PdistSelf←'#'␣PDist 'Create' 'Normal1' 'SystemRandomSource' 'Guid' 'Normal' 0 1  
␣Pdist 'Self'  
FN1  
␣Pdist 'Self'
```

