DBNorm

# Build-in dataset

DBNorm library provides four build-in data arrays for testing and they are DArray1 (22,277), DArray2 (22,277), DArray3 (54,675) and DArray4 (33,297). These four data arrays can be loaded via function data() or loadData().

|  |
| --- |
| # load build-in data arrays by data()  data(DArray1)  data(DArray2)  data(DArray3)  data(DArray4)  # or load build-in data arrays by loadData()  loadData(0) |

# Define standard distribution

The library allows user to define standard distributions for normalizing arbitrary distributions into standard ones.

|  |
| --- |
| # define a normal distribution  DBdata5 <- defineDist(Norm(mean=0, sd=1), -5, 5) |

# Visualizing distributions

The distributions of data arrays can be visualized by frequency or probability.

|  |  |
| --- | --- |
| **Frequency distribution** | **Probability distribution** |
|  |  |
| # Frequency distribution of DArray1  visDistData(DBdata1, "F", "DArray1", "Range", "Frequency")  # Probability distribution of DArray1  visDistData(DBdata1, "P", "DArray1", "Range", "Probability") | |

# Fitting distributions and visualization

The library provides several ways of fitting distributions, visualizing fitting results on distributions and generate fitting formula for normalization.

## Polynomial fitting

Users can define the degree of polynomial equations to fit. For example, n-degree polynomial equation is defined as

|  |  |
| --- | --- |
| **A 9-degree polynomial fitting** | **Formula** |
|  | y = (-0.000620900046122378)\*x^9+(-0.00124834884502108)\*x^8+(0.00662249052719529)\*x^7+(-0.0084800462241037)\*x^6+(0.00310757525823036)\*x^5+(-0.00428172357339282)\*x^4+(0.0130352656779109)\*x^3+(-0.00443148578371705)\*x^2+(-0.0320215363793821)\*x^1+(0.00199999999999999) |

|  |
| --- |
| # Fitting DArray’s distribution by a 9-degree polynomial equation  DBdata1 <- polyFit(DBdata1, 9)  # Visualize fitting results  visFitting(DBdata1, "DArray1", "Range", "Probability")  # Fitting equation  DBdata1$equ |

## Fourier fitting

Users can define the degree of Fourier equations to fit. For example, n-degree Fourier equation is defined as

|  |  |
| --- | --- |
| **A 3-degree Fourier fitting** | **Formula** |
|  | y = (0.00198199303768806) + (-0.00037946304847435)\*sin(1 \* x) + (-0.000156736257249885)\*cos(1 \* x) + (0.000179656025752176)\*sin(2 \* x) + (-0.000147268693497361)\*cos(2 \* x) + (-5.26254326598022e-05)\*sin(3 \* x) + (0.000157701465973085)\*cos(3 \* x) |

|  |
| --- |
| # Fitting DArray’s distribution by a 3-degree Fourier equation  DBdata1 <- fourierFit(DBdata1, 3)  # Visualize fitting results  visFitting(DBdata1, "DArray1", "Range", "Probability")  # Fitting equation  DBdata1$equ |

## Gaussian fitting

Users can use Gaussian equations to fit. For example, the Gaussian equation in the library is defined as

|  |  |
| --- | --- |
| **Gaussian fitting** | **Formula** |
|  | y = (0.0293110169514227)\*exp^(-(2.86021073423377)\*(x-5.54082132297444)^2) |

|  |
| --- |
| # Fitting DArray’s distribution by Gaussian equation  DBdata1 <- gaussianFit(DBdata1)  # Visualize fitting results  visFitting(DBdata1, "DArray1", "Range", "Probability")  # Fitting equation  DBdata1$equ |

## Customized fitting

Users can define an equation to fit. For example, we define an equation as

|  |  |
| --- | --- |
| **Gaussian fitting** | **Formula** |
|  | y = (0.0057963567671893) + (-0.000426884293350865)\*x + (0.000221349815448716)\*cos(x) |

|  |
| --- |
| # Fitting DArray’s distribution by a customized equation  DBdata1 = custFit(DBdata1, "y ~ x + cos(x)")  # Visualize fitting results  visFitting(DBdata1, "DArray1", "Range", "Probability")  # Fitting equation  DBdata1$equ |

# Normalization

The library offers the function to normalize an arbitrary data array to another one or a standard distribution so as to let them follow the same distribution.

## Normalizing between two arbitrary distribution

|  |  |
| --- | --- |
| **Before normalization** | |
| **DArray1** | **DArray3** |
|  |  |
| **After normalization** | |
| **DArray1** | **DArray3** |
|  |  |

|  |
| --- |
| # load build-in data arrays  data(DArray1)  data(DArray3)  # Capturing distribution information  DBdata1 <- genDistData(DArray1, 500)  DBdata3 <- genDistData(DArray3, 500)  # Poly Fitting  DBdata1 <- polyFit(DBdata1, 9)  DBdata3 <- polyFit(DBdata3, 9)  **# Normalization**  conNormalizer(DBdata1, DBdata3) |

## Normalizing an arbitrary distribution into a standard one

|  |  |
| --- | --- |
| **Before normalization** | |
| **DArray1** | **Norm(0, 1)** |
|  |  |
| **After normalization** | |
| **DArray1** | **Norm(0,1)** |
|  |  |

|  |
| --- |
| # load build-in data arrays  data(DArray1)  DBdata5 <- defineDist(Norm(mean=0, sd=1), -5, 5)  # Capturing distribution information  DBdata1 <- genDistData(DArray1, 500)  # Poly Fitting  DBdata1 <- polyFit(DBdata1, 9)  **# Normalization**  conNormalizer(DBdata1, DBdata5) |