## NAME

rdbstats - compute statistics on an rdb table

## SYNOPSIS

**rdbstats** [*options*] [*columns*]

## OPTIONS

Options may be abbreviated. Options which take values may be separated from their values either by white space or the = character.

**--help**

Print this help information and exit.

**--version**

Print the program version and exit.

**--group** column1[,column2[, ... ]]

Set the break column(s) to the specified column name(s). Consecutive rows with the same value in the break column(s) are treated as independent data sets. This option cannot be used in conjunction with the **--rows** option. This option may be repeated. For example:

```
rdbstats --group groupie < data.rdb
rdbstats --group=groupie < data.rdb
rdbstats --group=groupie,groupy < data.rdb
rdbstats -g groupie,groupy < data.rdb
rdbstats -g groupie -g groupy < data.rdb
```

**--normalize** `ave|median`

This specifies that the output statistics should be normalized either by the median or the average. If `median` is specified, the **--quartiles** option is implicitly turned on unless **--percentiles** is specified. Normalization is done in the following fashion

```
Q  / abs(ave or median)
```

where `Q` is the statistic to be normalized. For example:

```
rdbstats --normalize ave < data.rdb
rdbstats --normalize=ave < data.rdb
rdbstats -n ave < data.rdb
```

**--percentiles** *percentile list*

Generate percentile statistics, including the median, as well as the normal statistics. The percentile list is a comma separated list of percentages. For example, `--percentiles=33` will output the interpolated value which is greater than 33 percent of the data. Because all of the data must be stored in memory for all of the columns for these statistics, this is a rather memory intensive option for large data sets. For example:

```
rdbstats --percentiles 10,20,30 --percentiles 15,35 < data.rdb
rdbstats --percentiles=10,20,30 --percentiles=15,35 < data.rdb
rdbstats -p 10,20,30 -p 15,35 < data.rdb
```

**--quartiles**

Generate quartile statistics (first and last quartile and median) as well as the normal statistics.

Because all of the data must be stored in memory for all of the columns for these statistics, this is a rather memory intensive option for large data sets. For example:

```
rdbstats --quartiles < data.rdb
rdbstats -q < data.rdb
```

**--rows** *range list*

Specify the ranges of rows to operate on. This option may be used multiple times. This option cannot be used in conjunction with the **--group** option.

**--all**

```
rdbstats --all < data.rdb
rdbstats -a < data.rdb
rdbstats < data.rdb
```

Generate statistics for all numerical columns.

**--d** *column*[,*def*]

override the column type definition for *column* in the rdb file, setting it to *def*, if specified, or the opposite of the current definition, if *def* is not specified.

```
rdbstats -d groupie < data.rdb
rdbstats -d groupie,N < data.rdb
```

## DESCRIPTION

**rdbstats** generates statistics for columns in an rdb table. It reads the rdbtable from STDIN and writes an rdbtable to STDOUT. By default it generates the sum, average, standard deviation, minimum, and maximum values of the data. It can optionally generate the first and last quartiles and the median.

**rdbstats** can operate on more than one column. It normally operates on the columns specified on the command line, but if the **-all** option is specified, it works on all numeric columns. New columns containing the data products are created by appending suffixes to the names of the source columns. The suffices, and the contents of the columns are:

`_n`
: the number of data items

`_sum`
: the sum of the data

`_ave`
: the average of the data

`_dev`
: the standard deviation of the data

`_min`
: the minimum data value

`_max`
: the maximum data value

`_rss`

the square root of the sum of the squares of the data

`_fq`

the first quartile

`_median`

the median

`_max`

the last quartile

`_pN`

the percentile `N`, where `N` is specified by the **--percentiles** options

## Subsets of the data

The input data may be split into independent subsets in one of two ways. Subsets may be identified by one or more *break* columns. Contiguous data with the same value in the break columns are treated as a subset. For example, given the rdb database,

```
row     dataset data
N       N       N
1       1       3.1
2       1       4.9
3       1       62.2
4       2       122
5       2       233
6       2       232
```

If `dataset` is the break column, rows 1-3 are a subset and rows 4-6 are a subset. Note that the break column may be either numeric or string. Its value need have no intrinsic meaning.

The second method is to specify ranges of the record numbers of the rows to be included. Unlike with the break column, subsets need not be contiguous; they may not, however, overlap. Row range lists are specified with the **--rows** option; multiple instances of the option are allowed to specify multiple subsets. Row ranges have the following syntax:

n

{ n }

a:b

{x | a<=x && x<=b}

a,b

{x | x=a, x=b }

Range lists are composed of ranges, separated by commas. Ranges in a range list must be disjoint, and must be listed in increasing order.

Valid characters in a range are 0-9, '(', ')', '-' and ','. White space and underscore (_) are ignored. Other characters are not allowed.

Here are some range list examples:

-

| | |
|---|---|
| | { } |
| `1` | { 1 } |
| `1-2` | { 1, 2 } |
| `(-)` | the integers |
| `1-3,4,18-21` | { 1, 2, 3, 4, 18, 19, 20, 21 } |

Note that record numbers begin with 1.