

Package ‘rredis’

May 15, 2012

Type Package

Title Redis client for R

Version 1.6.6

Date 2012-05-14

Author B. W. Lewis

Maintainer B. W. Lewis <blewis@illposed.net>

Description An R client for the Redis persistent key-value database available from <http://redis.io>.

License Apache License 2.0

LazyLoad yes

Repository CRAN

Date/Publication 2012-05-15 06:38:23

R topics documented:

redis-package	3
redisAuth	4
redisBgRewriteAOF	5
redisBgSave	6
redisBLPop	6
redisBRPop	7
redisClose	9
redisCmd	10
redisConnect	11
redisDBSize	12
redisDecr	13
redisDecrBy	14
redisDelete	15
redisDiscard	16
redisExec	16

redisExists	17
redisExpire	18
redisExpireAt	19
redisFlushAll	20
redisFlushDB	20
redisGet	21
redisGetResponse	22
redisGetSet	23
redisHDel	25
redisHExists	26
redisHFields	27
redisHGet	28
redisHGetAll	29
redisHIncrBy	30
redisHKeys	31
redisHLen	32
redisHMGet	33
redisHMSet	34
redisHSet	35
redisHVals	36
redisIncr	37
redisIncrBy	38
redisInfo	39
redisKeys	39
redisLIndex	40
redisLLen	41
redisLPop	42
redisLPush	43
redisLRange	44
redisLRem	45
redisLSet	46
redisLTrim	47
redisMGet	48
redisMonitorChannels	49
redisMove	50
redisMSet	51
redisMulti	52
redisPublish	53
redisRandomKey	54
redisRename	55
redisRPop	56
redisRPopLPush	57
redisRPush	58
redisSAdd	59
redisSave	60
redisSCard	61
redisSDiff	62
redisSDiffStore	63

redisSelect	64
redisSet	65
redisSetBlocking	66
redisSetContext	67
redisShutdown	68
redisSInter	69
redisSInterStore	70
redisSIsMember	71
redisSlaveOf	72
redisSMembers	73
redisSMove	74
redisSort	75
redisSPop	76
redisSRandMember	77
redisSRem	78
redisSubscribe	79
redisSUnion	80
redisSUnionStore	81
redisTTL	82
redisType	83
redisUnsubscribe	84
redisUnwatch	86
redisWatch	87
redisZAdd	87
redisZCard	88
redisZIncrBy	89
redisZInterStore	89
redisZRange	90
redisZRangeByScore	91
redisZRank	92
redisZRem	93
redisZRemRangeByRank	93
redisZRemRangeByScore	94
redisZScore	95
redisZUnionStore	96

Index	97
--------------	-----------

redis-package	<i>An R client for Redis</i>
---------------	------------------------------

Description

Redis is a high-performance, in-memory, networked, key/value-like database. The redis package for R provides a native R language interface to Redis. The R client can store and retrieve arbitrary serializable R objects as well as raw data objects like strings and binary blobs from other clients. Redis supports a number of innovative value types beyond arbitrary binary data, including sets and lifo/fifo-like lists, database replication, database persistence, high-performance virtual memory

management and much more. Please see the `redis` vignette for more information and examples on using Redis and R together.

Details

Package: redis
Type: Package
Version: 0.01
Date: 2010-01-26
License: Apache
LazyLoad: yes

Author(s)

B. W. Lewis, with substantial assistance from Pat Sheilds

Maintainer: B. W. Lewis <blewis@illposed.net>

References

<http://redis.io/commands>

Examples

```
## Not run:  
redisConnect()  
redisSet('x',runif(5))  
redisGet('x')  
redisClose()  
  
## End(Not run)
```

redisAuth

Redis authentication.

Description

Redis supports a trivially simple and insecure authentication method. This function implements it.

Usage

```
redisAuth(pwd)
```

Arguments

pwd The (required) password.

Details

You should not use this function. If you need a secure key/value database, it's best not to use Redis for now.

Value

TRUE if successful, FALSE otherwise.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

redisBgRewriteAOF *redisBgRewriteAOF*

Description

Re-write the Redis append-only file in the background.

Usage

`redisBgRewriteAOF()`

Details

BGREWRITEAOF rewrites the Append Only File in the background when it gets too big. The Redis Append Only File is a Journal, so every operation modifying the dataset is logged in the Append Only File (and replayed at startup). This means that the Append Only File always grows. In order to rebuild its content the BGREWRITEAOF creates a new version of the append only file starting directly from the dataset in memory in order to guarantee the generation of the minimal number of commands needed to rebuild the database. (These details are copied verbatim from the Redis master documentation, see the references below.)

Value

Nothing is returned.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

redisBgSave	<i>redisBgSave</i>
-------------	--------------------

Description

Asynchronously save that database to disk.

Usage

```
redisBgSave()
```

Details

Force Redis to save the database(s) to disk in the background.

Value

Nothing is returned. Check the UNIX time of the last completed save operation with the `redisLastsave` function.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

redisBLPop	<i>Blocking List Pop</i>
------------	--------------------------

Description

Pop the first available value from a key or list of keys, blocking until a value is available.

Usage

```
redisBLPop(keys, timeout = 0)
```

Arguments

keys	A character key value or vector of key values to monitor.
timeout	A timeout in seconds after which, if no value is available, the function returns NULL. A value of zero indicates block indefinitely.

Details

redisBLPop blocks until at least one of the keys exists and contains a non-empty value, or until the specified timeout period is reached, whichever comes first. Keys are scanned in the order that they are specified.

Value

redisBLPop returns NULL after the timeout period, or a list containing:

key	The first key encountered with an available value,
value	The corresponding value.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

redisBRpop

Examples

```
## Not run:
redisConnect()
redisBLpop('x', 5)
redisLPush('x', runif(5))
redisBLPop('x')

## End(Not run)
```

redisBRPop

Blocking List Pop

Description

Pop the first available value from a key or list of keys, blocking until a value is available.

Usage

```
redisBRPop(keys, timeout = 0)
```

Arguments

keys	A character key value or vector of key values to monitor.
timeout	A timeout in seconds after which, if no value is available, the function returns NULL. A value of zero indicates block indefinitely.

Details

redisBRPop blocks until at least one of the keys exists and contains a non-empty value, or until the specified timeout period is reached, whichever comes first. Keys are scanned in the order that they are specified.

Value

redisBRPop returns NULL after the timeout period, or a list containing:

key	The first key encountered with an available value,
value	The corresponding value.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

redisBLpop

Examples

```
## Not run:
redisConnect()
redisBLpop('x', 5)
redisLPush('x',runif(5))
redisBRPop('x')

## End(Not run)
```

redisClose	<i>Close an open connection to a Redis server.</i>
------------	--

Description

The redisClose function closes any open connection to a Redis server.

Usage

```
redisClose()
```

Details

A running instance of a Redis server is required.

Value

Nothing is returned. Errors are displayed if the function fails to close the connection to the Redis server, or if the connection is invalid.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisConnect](#)

Examples

```
## Not run:  
redisConnect()  
redisSet('x',runif(5))  
redisGet('x')  
redisClose()  
  
## End(Not run)
```

`redisCmd`*General Redis Interface Function*

Description

Perform any Redis command.

Usage

```
redisCmd(CMD, ..., raw = FALSE)
```

Arguments

<code>CMD</code>	The (required) Redis command (character string).
<code>...</code>	Redis command arguments.
<code>raw</code>	If TRUE, return results in raw byte form.

Details

Use this low-level function to perform any Redis operation. The ... argument(s) not already in raw format will be converted to raw byte format, and non-character values will be serialized.

Value

Output from the Redis command—varies depending on command.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

Examples

```
## Not run:  
redisCmd('set', 'x', runif(5))  
redisCmd('get', 'x')  
  
## End(Not run)
```

redisConnect	<i>Connect to a Redis server.</i>
--------------	-----------------------------------

Description

Connect to an available Redis server on the specified port.

Usage

```
redisConnect(host = "localhost", port = 6379, returnRef = FALSE)
```

Arguments

host	The Redis server host name or inet address (optional, character)
port	The Redis port number (optional, numeric or integer)
returnRef	Set returnRef=TRUE to return a list describing the Redis connection (not presently useful).

Details

A running instance of a Redis server is required.

Value

Nothing is returned by default. Errors are displayed if the function fails to connect to the specified Redis server. Disconnect from a connected server with `redisClose`.

If `returnRef` is set to `TRUE` and no error occurs, a list describing the Redis connection will be returned. A future version of the package will use this feature to support multiple Redis connections with the `attachRedis` function.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisClose](#)

Examples

```
## Not run:
redisConnect()
redisSet('x',runif(5))
redisGet('x')
redisClose()

## End(Not run)
```

redisDBSize	<i>Return the number of keys in the current Redis database.</i>
-------------	---

Description

Return the number of keys in the current Redis database.

Usage

```
redisDBSize()
```

Details

Use `redisSelect` to choose a current database from among the available Redis databases.

Value

The number of keys in the current database.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSelect](#)

Examples

```
## Not run:
redisDBSize()

## End(Not run)
```

redisDecr	<i>Decrement a value by one.</i>
-----------	----------------------------------

Description

Decrement the value corresponding to the specified key by one.

Usage

```
redisDecr(key)
```

Arguments

key A key corresponding to the value to decrement.

Details

The value should be a character representation of an integer. If the key does not exist or contains a value of a wrong type, set the key to the value of "0" before to perform the operation.

Value

The new value of key after the decrement, returned as a character string.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisIncr](#)

Examples

```
## Not run:  
redisSet('x', '5')  
redisDecr('x')  
  
## End(Not run)
```

`redisDecrBy`*Decrement a value by the specified value.*

Description

Decrement the value corresponding to the specified key.

Usage

```
redisDecrBy(key, value)
```

Arguments

<code>key</code>	A key corresponding to the value to decrement.
<code>value</code>	The value to decrement by (integer, numeric, or character).

Details

The value should be a character representation of an integer. If the key does not exist or contains a value of a wrong type, set the key to the value of "0" before to perform the operation.

Value

The new value of key after the decrement, returned as a character string.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisIncr](#)

Examples

```
## Not run:  
redisSet('x', '5')  
redisDecrBy('x', '3')  
  
## End(Not run)
```

`redisDelete`*Delete a key and associated value from Redis.*

Description

Delete a key and associated value from Redis.

Usage

```
redisDelete(key)
```

Arguments

`key` The (required) character identifier to be looked up.

Details

The key must not contain spaces or newline characters (otherwise an error will be thrown).

Value

Nothing is returned if the key/value pair is successfully deleted. A warning is thrown if the they key could not be found.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSet](#)

Examples

```
## Not run:  
redisSet('x',runif(5))  
redisDelete('x')  
  
## End(Not run)
```

redisDiscard	<i>redisDiscard</i>
--------------	---------------------

Description

Discard the current transaction block.

Usage

```
redisDiscard()
```

Details

Discard the current transaction block.

Value

"OK"

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

redisExec	<i>redisExec</i>
-----------	------------------

Description

End a transaction block.

Usage

```
redisExec()
```

Details

End a transaction block. All Redis statements issued after `redisMulti` will be queued locally and then sent to the Redis server en masse once the `redisExec` function is issued.

Value

The results of all queued functions are returned in a list.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

redisExists	<i>Test the existence of a key in the Redis database.</i>
-------------	---

Description

Test the existence of a key in the Redis database.

Usage

```
redisExists(key)
```

Arguments

key The (required) character identifier to be looked up.

Details

The key must not contain spaces or newline characters (otherwise an error will be thrown).

Value

Returns FALSE if no matching key, TRUE if matching key exists, and NULL if an error occurred.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSet](#)

Examples

```
## Not run:  
redisSet('x',runif(5))  
redisExists('x')
```

```
## End(Not run)
```

redisExpire	<i>Set a timeout on the specified key.</i>
-------------	--

Description

Set a timeout on the specified key, after which the key and corresponding value will be deleted.

Usage

```
redisExpire(key, seconds)
```

Arguments

key	The character key on which to set the timeout.
seconds	The integer timeout in seconds.

Details

Operations that modify value(s) corresponding to a key subsequent to the redisExpire function clear the timeout, removing the expiration. The redisExpire function can't set a new timeout value once a timeout has been set on a key.

Value

Boolean TRUE if the timeout command was successful, FALSE otherwise.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisExpireAt](#)

Examples

```
## Not run:  
redisConnect()  
redisLPush('x',runif(5))  
redisExpire('x', 3)  
  
## End(Not run)
```

redisExpireAt	<i>Set a timeout on the specified key.</i>
---------------	--

Description

Set a timeout on the specified key, after which time the key and corresponding value will be deleted.

Usage

```
redisExpireAt(key, time)
```

Arguments

key	The character key on which to set the timeout.
time	The UNIX time of expiration.

Details

Operations that modify value(s) corresponding to a key subsequent to the redisExpireAt function clear the timeout, removing the expiration. The redisExpireAt function can't set a new timeout value once a timeout has been set on a key.

Value

Boolean TRUE if the timeout command was successful, FALSE otherwise.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisExpire](#)

Examples

```
## Not run:  
redisConnect()  
redisLPush('x',runif(5))  
redisExpireAt('x', 1266209144)  
  
## End(Not run)
```

redisFlushAll *Delete all keys and values from all databases.*

Description

Delete all keys and values from all databases, not just the currently selected one.

Usage

```
redisFlushAll()
```

Value

Returns TRUE if successful, FALSE otherwise.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisFlushDB](#)

Examples

```
## Not run:  
redisConnect()  
redisFlushAll()  
  
## End(Not run)
```

redisFlushDB *Delete all keys and values from the current database.*

Description

Delete all keys and values from the currently selected database.

Usage

```
redisFlushDB()
```

Value

Returns TRUE if successful, FALSE otherwise.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisFlushAll](#)

Examples

```
## Not run:  
redisConnect()  
redisFlushDB()  
  
## End(Not run)
```

redisGet	<i>Retrieve a value from Redis.</i>
----------	-------------------------------------

Description

Retrieve a value identified by a key from the Redis database.

Usage

```
redisGet(key, raw = FALSE)
```

Arguments

key	The (required) character identifier for the value to be looked up.
raw	Set raw=TRUE to skip de-serialization of the data.

Details

The key must not contain spaces or newline characters (otherwise an error will be thrown). The raw option is used to retrieve binary data from other languages.

Value

The value corresponding to the specified key, or NULL if the matching key contained no value or if no matching key was found.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSet](#)

Examples

```
## Not run:
redisSet('x',runif(5))
redisGet('x')

## End(Not run)
```

redisGetResponse	<i>redisGetResponse</i>
------------------	-------------------------

Description

Service messages from all subscribed Redis message channels.

Usage

```
redisGetResponse(all=TRUE)
```

Arguments

all The all argument is ignored. It's left there for backwards compatibility with code based on older package versions.

Details

(From the Redis.io documentation): `redisSubscribe`, `redisUnsubscribe` and `redisPublish` implement the Publish/Subscribe messaging paradigm where (citing Wikipedia) senders (publishers) are not programmed to send their messages to specific receivers (subscribers). Rather, published messages are characterized into channels, without knowledge of what (if any) subscribers there may be. Subscribers express interest in one or more channels, and only receive messages that are of interest, without knowledge of what (if any) publishers there are.

The `redisGetResponse` function may be called to service messages on all subscribed channels. When a message is received the a list of three elements is returned containing: the character string 'message', the name of receiving channel, and the message content.

WARNING: The `redisGetResponse` function blocks indefinitely until a message is received.

Value

A list containing the string 'message', the channel name from which the message was received, and the message data.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSubscribe](#) [redisPublish](#) [redisUnsubscribe](#) [redisMonitorChannels](#)

Examples

```
## Not run:
redisConnect()
# Define a callback function to process messages from channel 1:
channel1 <- function(x) {
  cat("Message received from channel 1: ",x,"\n")
}
# Define a callback function to process messages from channel 2:
channel2 <- function(x) {
  cat("Message received from channel 2: ",x,"\n")
}
redisSubscribe(c('channel1','channel2'))
# Monitor channels for at least one minute (and indefinitely until
# a message is received):
t1 <- proc.time()[[3]]
while(proc.time()[[3]] - t1 < 60)
{
  print(redisGetResponse())
}
redisUnsubscribe(c('channel1','channel2'))

## End(Not run)
```

redisGetSet

Store a value in Redis, returning the previously defined value.

Description

Store a value identified by a character key name in the Redis database, returning the previously defined value or NULL if the key was not already associated with a value.

Usage

```
redisGetSet(key, value)
```

Arguments

key	The (required) character identifier for the value to be stored.
value	The (required) object to associate with the key.

Details

The key must not contain spaces or newline characters (otherwise an error will be thrown).

The value object is copied to the Redis server. The value to be stored may can be any serializable R object up to the Redis maximum object size (excluding, for example, external pointer references). References to other R objects or environments inside the value are not guaranteed to be preserved.

In order to store strings that can easily be read by other clients, first convert the character object using the [charToRaw](#) function as shown in the examples.

Value

The previous value associated with key or NULL if no previous association exists.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisGet](#)

Examples

```
## Not run:  
# Store an R object with the key 'x':  
redisGetSet('x',runif(5))  
  
## End(Not run)
```

redisHDel	<i>Delete a hash value.</i>
-----------	-----------------------------

Description

Delete the value associated with the given key/field combination.

Usage

```
redisHDel(key, field)
```

Arguments

key	A key name.
field	A field name.

Value

Nothing is returned if the key/value pair is successfully deleted. A warning is thrown if the they key could not be found.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisHSet](#)

Examples

```
## Not run:  
redisHSet('A',list(x=1,y=2,z=3))  
redisHDel('A','x')  
  
## End(Not run)
```

redisHExists	<i>Test the existence of a hash.</i>
--------------	--------------------------------------

Description

Test the existence of a hash combination in the Redis database.

Usage

```
redisHExists(key, field)
```

Arguments

key	A key name.
field	A field name.

Value

Returns FALSE if no matching key/field combination, TRUE if matching entry exists, and NULL if an error occurred.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisHSet](#)

Examples

```
## Not run:  
redisHSet('A','x',runif(5))  
redisHExists('A','x')  
  
## End(Not run)
```

redisHFields	<i>Redis hash fields.</i>
--------------	---------------------------

Description

Return the fields associated with the given key.

Usage

```
redisHFields(key)
```

Arguments

key	The key to look up.
-----	---------------------

Details

Returns the fields in the Redis hash associated with key. If the key is not found, or if the hash is empty, NULL is returned. If the key is associated with a value of type other than 'hash,' an error is thrown.

Value

A list of fields defined for the given key.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisHSet](#)

Examples

```
## Not run:  
redisHSet('A',list(x=1,y=2,z=3))  
redisHFields('A')  
  
## End(Not run)
```

redisHGet	<i>Retrieve a hashed value from Redis.</i>
-----------	--

Description

Retrieve a value identified by a key and field from the Redis database.

Usage

```
redisHGet(key, field, raw = FALSE)
```

Arguments

key	A key name.
field	A field name.
raw	Set raw=TRUE to skip de-serialization of the data.

Details

Redis hash values store values in one or more fields associated with a single key name.

Value

The value corresponding to the specified key/field, or NULL if the matching key/field hash contained no value or if no matching key or field was found.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisHSet](#)

Examples

```
## Not run:  
redisHSet('A', 'x', runif(5))  
redisHGet('A', 'x')  
  
## End(Not run)
```

redisHGetAll	<i>Redis hash fields and values.</i>
--------------	--------------------------------------

Description

Return all fields and values associated with the given key.

Usage

```
redisHGetAll(key)
```

Arguments

key	The key to look up.
-----	---------------------

Details

Returns all the fields and their values in the Redis hash associated with key. If the key is not found, or if the hash is empty, NULL is returned. If the key is associated with a value of type other than 'hash,' an error is thrown.

Value

A list of values defined for the given key, named by the field names.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisHSet](#)

Examples

```
## Not run:  
redisHSet('A',list(x=1,y=2,z=3))  
redisHGetAll('A')  
  
## End(Not run)
```

redisHIncrBy	<i>Increment a value.</i>
--------------	---------------------------

Description

Increment the value corresponding to the given key/field combination by the specified value.

Usage

```
redisHIncrBy(key, field, value)
```

Arguments

key	A key name.
field	A field name.
value	The value to increment by (integer, numeric, or character).

Details

The value should be a character representation of an integer. If the key/field value does not exist or contains a value of a wrong type, set the key to the value of "0" before to perform the operation.

Value

The new value of key after the increment, returned as a character string.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisHSet](#)

Examples

```
## Not run:  
redisHSet('A','x',5)  
redisHIncrBy('A','x',3)  
  
## End(Not run)
```

redisHKeys	<i>Redis hash fields.</i>
------------	---------------------------

Description

Return the fields associated with the given key.

Usage

```
redisHKeys(key)
```

Arguments

key	The key to look up.
-----	---------------------

Details

Returns the fields in the Redis hash associated with key. If the key is not found, or if the hash is empty, NULL is returned. If the key is associated with a value of type other than 'hash,' an error is thrown.

Value

A list of fields defined for the given key.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisHSet](#)

Examples

```
## Not run:  
redisHSet('A',list(x=1,y=2,z=3))  
redisHKeys('A')  
  
## End(Not run)
```

redisHLen	<i>Redis hash length.</i>
-----------	---------------------------

Description

Return the number of fields associated with the given key.

Usage

```
redisHLen(key)
```

Arguments

key	The key to look up.
-----	---------------------

Details

Returns the number of fields in the Redis hash associated with key. If the key is not found, or if the hash is empty, 0 is returned. If the key is associated with a value of type other than 'hash,' an error is thrown.

Value

The number of fields defined for the given key.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisHSet](#)

Examples

```
## Not run:  
redisHSet('A',list(x=1,y=2,z=3))  
redisHLen('A')  
  
## End(Not run)
```

redisHGet	<i>Retrieve a list of hash values.</i>
-----------	--

Description

Retrieve a list of hash values from specified key/field pairs at once.

Usage

```
redisHGet(key, fields)
```

Arguments

key	The (required) character identifier for the key name.
fields	An R string vector of hash fields to retrieve.

Details

Redis hash values store values in one or more fields associated with a single key name. The redisHGet function retrieves several fields associated with one key at once. Fields not present return NULL.

Value

A named list of retrieved values.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisHSet](#), [redisHSet](#)

Examples

```
## Not run:  
redisHSet('A', list(x=1,y=2,z=3))  
redisHGet('A',c('y','z'))  
  
## End(Not run)
```

redisHMSets	<i>Store a list of hash values.</i>
-------------	-------------------------------------

Description

Store a list of hash values at once.

Usage

```
redisHMSets(key, values)
```

Arguments

key	The (required) character identifier for the key name.
values	An R list of values to be stored. The list names are used as field names and must be nonempty.

Details

Redis hash values store values in one or more fields associated with a single key name. The `redisHMSets` function stores several fields associated with one key at once. Values already occupying any specified field slots are replaced.

Value

TRUE is returned on success.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisHGet](#)

Examples

```
## Not run:  
redisHMSets('A', list(x=1,y=2,z=3))  
redisHGet('A','y')  
  
## End(Not run)
```

redisHSet	<i>Store a hash value in Redis.</i>
-----------	-------------------------------------

Description

Store a value identified by a character key name and field in the Redis database. Any existing value with the same key/field will be replaced by the new value unless NX is set to TRUE.

Usage

```
redisHSet(key, field, value, NX = FALSE)
```

Arguments

key	The (required) character identifier for the key name.
field	The (required) character identifier for the field name.
value	The (required) object to associate with the key/field.
NX	If NX = TRUE, existing values will not be replaced.

Details

Redis hash values store values in one or more fields associated with a single key name.

The value to be stored can be any serializable R object up to the Redis maximum object size (excluding, for example, external pointer references). References to other R objects or environments inside the value are not guaranteed to be preserved.

In order to store strings that can easily be read by other clients, first convert the character object using the [charToRaw](#) function as shown in the examples.

Value

TRUE is returned on success. If NX = FALSE and a value already exists, the value will not be replaced and FALSE will be returned.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisHGet](#)

Examples

```
## Not run:
redisHSet('A', 'x',runif(5))

## End(Not run)
```

redisHVals	<i>Redis hash values.</i>
------------	---------------------------

Description

Return all the values associated with the given hash key.

Usage

```
redisHVals(key)
```

Arguments

key The key to look up.

Details

Returns the values in the Redis hash associated with key. If the key is not found, or if the list is empty, NULL is returned. If the key is associated with a value of type other than 'hash,' an error is thrown.

Value

A list of values defined for the given key.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisHSet](#)

Examples

```
## Not run:
redisHSet('A',list(x=1,y=2,z=3))
redisHVals('A')

## End(Not run)
```

redisIncr	<i>Increment a value by one.</i>
-----------	----------------------------------

Description

Increment the value corresponding to the specified key by one.

Usage

```
redisIncr(key)
```

Arguments

key A key corresponding to the value to increment.

Details

The value should be a character representation of an integer. If the key does not exist or contains a value of a wrong type, set the key to the value of "0" before to perform the operation.

Value

The new value of key after the increment, returned as a character string.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisDecr](#)

Examples

```
## Not run:  
redisSet('x', '5')  
redisIncr('x')  
  
## End(Not run)
```

redisIncrBy	<i>Increment a value by the specified value.</i>
-------------	--

Description

Increment the value corresponding to the specified key.

Usage

```
redisIncrBy(key, value)
```

Arguments

key	A key corresponding to the value to increment.
value	The value to increment by (integer, numeric, or character).

Details

The value should be a character representation of an integer. If the key does not exist or contains a value of a wrong type, set the key to the value of "0" before to perform the operation.

Value

The new value of key after the increment, returned as a character string.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisIncr](#)

Examples

```
## Not run:  
redisSet('x', '5')  
redisIncrBy('x', '3')  
  
## End(Not run)
```

redisInfo	<i>redisInfo</i>
-----------	------------------

Description

Return system information about Redis.

Usage

```
redisInfo()
```

Value

A list of various Redis system parameters is returned, including at least: the Redis version, connected clients, connected slaves, used memory, changes since last save, last save time (UNIX time), total connections received, total commands processed, uptime in seconds, uptime in days.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

redisKeys	<i>Return a list of all keys in the active Redis database.</i>
-----------	--

Description

Return a list of all keys in the active Redis database that match the specified pattern.

Usage

```
redisKeys(pattern = "*")
```

Arguments

pattern The character string pattern used to match keys.

Details

Basic string-matching wildcards are supported. Use '?' to match any single character and '*' to match zero or more characters.

Value

A space-delimited character string containing all keys that match the pattern in the active Redis database.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSelect](#)

Examples

```
## Not run:  
redisKeys()  
  
## End(Not run)
```

redisLIndex

Retrieve a value from a Redis 'list.'

Description

Retrieve a value from a Redis 'list' at the specified index without removing it from the list.

Usage

```
redisLIndex(key, index)
```

Arguments

key	The key (whose value is of the type 'list').
index	The list index to retrieve.

Details

List indices begin at 0. Negative indexes are supported, for example -1 is the last element, -2 the penultimate and so on. If the value stored at key is not of the 'list' type an error is returned. If the index is out of range an empty string is returned.

Value

The corresponding value or an empty string if the index is out of bounds.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisLPop](#)

Examples

```
## Not run:  
redisConnect()  
  
## End(Not run)
```

redisLLen	<i>Redis list length.</i>
-----------	---------------------------

Description

Return the length of the Redis list associated with the specified key.

Usage

```
redisLLen(key)
```

Arguments

key The key to look up.

Details

Returns the length of the Redis list associated with key. If the key is not found, or if the list is empty, 0 is returned. If the key is associated with a value of type other than 'list,' an error is thrown.

Value

The length if the list.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

redisBRpop

Examples

```
## Not run:  
redisConnect()  
redisLPush('list',1)  
redisLPush('list',2)  
redisLPush('list',3)  
redisLLen('list')  
  
## End(Not run)
```

redisLPop

Remove the first element from a list.

Description

Atomically return and remove the first element of the list. For example if the list contains the elements "a","b","c" redisLPop will return "a" and the list will become "b","c".

Usage

```
redisLPop(key)
```

Arguments

key The desired key associated with a list.

Value

The first element of the list associated with the specified key, or NULL if the list is empty.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisRPop](#), [redisLPush](#)

Examples

```
## Not run:  
redisConnect()  
redisLPush('a',1)  
redisLPush('a',2)  
redisLPush('a',3)  
redisLPop('a')  
  
## End(Not run)
```

redisLPush	<i>Add a value to the tail of a list.</i>
------------	---

Description

Add a value to the tail of a list corresponding to the specified key.

Usage

```
redisLPush(key, value)
```

Arguments

key	The desired key corresponding to a list.
value	The element to add to the list.

Details

If the key does not exist an empty list is created just before the append operation. If the key exists but is not a list an error is returned.

Value

The length of the list after the push operation.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisRPush](#)

Examples

```
## Not run:
redisConnect()
redisLPush('x',1)
redisLPush('x',2)
redisLPush('x',3)

## End(Not run)
```

redisLRange	<i>Copy values from a list.</i>
-------------	---------------------------------

Description

Return the elements of a list between the start and end indices, inclusively.

Usage

```
redisLRange(key, start, end)
```

Arguments

key	A key corresponding to a list.
start	The beginning index from which to read list elements.
end	The ending index.

Details

List indices begin with 0. If the start or end indices are beyond the bounds of the list, return the list elements up to the index bounds.

Value

An indexed list containing the returned values. An error will be thrown if the specified key does not correspond to a list or if the key can't be found.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisLPop](#)

Examples

```
## Not run:
redisConnect()
redisLPush('x',1)
redisLPush('x',2)
redisLPush('x',3)
redisLRange('x',0,2)

## End(Not run)
```

redisLRem	<i>Remove elements from a list.</i>
-----------	-------------------------------------

Description

Remove the first count occurrences of the given value from a list corresponding to the specified key.

Usage

```
redisLRem(key, count, value)
```

Arguments

key	The desired key corresponding to a list.
count	The maximum number of occurrences to remove.
value	Remove elements matching this value from the list.

Details

Remove the first count occurrences of the value element from the list. If count is zero all the elements are removed. If count is negative elements are removed from tail to head, instead of the default removal behavior from head to tail. So for example `redisLRem` with count -2 and value "hello" applied to the list (a,b,c,hello,x,hello,hello) will yield the list (a,b,c,hello,x). The number of removed elements is returned as an integer upon success. Note that non-existing keys are considered as empty lists.

Value

The number of occurrences removed.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also[redisLPush](#)**Examples**

```
## Not run:  
redisConnect()  
  
## End(Not run)
```

redisLSet	<i>Set a value within a list.</i>
-----------	-----------------------------------

Description

Set the value of an element at the given index in a list corresponding to the specified key.

Usage

```
redisLSet(key, index, value)
```

Arguments

key	A key corresponding to a Redis list.
index	The index within the list of the element to write to.
value	The value to set.

Details

Indices begin at zero. Out of range indices throw an error.

Value

TRUE if successful, FALSE otherwise.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also[redisLPush](#)

Examples

```
## Not run:
redisConnect()
redisLPush('list',pi)
redisLSet('list',0,5)
redisLIndex('list',0)

## End(Not run)
```

redisLTrim	<i>Trim a list.</i>
------------	---------------------

Description

Trim an existing list so that it will contain only the specified range of elements specified.

Usage

```
redisLTrim(key, start, end)
```

Arguments

key	A key corresponding to a Redis list object.
start	The starting list index at which to trim.
end	The ending list index at which to trim.

Details

Start and end are zero-based indexes. 0 is the first element of the list (the list head), 1 the next element and so on.

Value

TRUE if successful. An error is thrown if the key does not exist or correspond to a list value.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisLPush](#)

Examples

```
## Not run:
redisConnect()
redisLPush('x',1)
redisLPush('x',2)
redisLPush('x',3)
redisLTrim('x',0,1)
redisLRange('x',0,99)

## End(Not run)
```

redisMGet	<i>Retrieve one or more values from Redis.</i>
-----------	--

Description

Retrieve values corresponding to the specified list of keys.

Usage

```
redisMGet(keys, raw = FALSE)
```

Arguments

keys	A vector or list of character strings corresponding to keys to retrieve.
raw	Set raw=TRUE to skip de-serialization of the returned values.

Details

Values are returned in a list with names corresponding to keys. The raw argument is used to retrieve binary data from other languages.

Value

A list of retrieved key/value pairs. Missing values return NULL.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisMSet](#)

Examples

```
## Not run:
redisSet('x',runif(5))
redisSet('y',runif(5))
redisMGet(c('x','y'))

## End(Not run)
```

redisMonitorChannels *redisMonitorChannels*

Description

Subscribe to one or more Redis message channels.

Usage

```
redisMonitorChannels()
```

Details

(From the Redis.io documentation): `redisSubscribe`, `redisUnsubscribe` and `redisPublish` implement the Publish/Subscribe messaging paradigm where (citing Wikipedia) senders (publishers) are not programmed to send their messages to specific receivers (subscribers). Rather, published messages are characterized into channels, without knowledge of what (if any) subscribers there may be. Subscribers express interest in one or more channels, and only receive messages that are of interest, without knowledge of what (if any) publishers there are.

The `redisMonitorChannels` function may be called repeatedly in an event loop to service messages on all subscribed channels. When a message is received, the `redisMonitorChannels` function will attempt to evaluate a callback function with same name as the channel, with the message as its single argument. If no such function can be found, the message is returned. See the help page for `redisGetResponse` for a description of the message format.

WARNING: The `redisMonitorChannels` function blocks indefinitely until a message is received. Use the lower-level `redisGetResponse` function to simply poll channels for messages without evaluating function callbacks.

Value

The result of an evaluated function callback message, or if no matching callback exists, the message.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSubscribe](#) [redisPublish](#) [redisUnsubscribe](#) [redisGetResponse](#) [redisMonitorChannels](#)

Examples

```
## Not run:
redisConnect()
# Define a callback function to process messages from channel 1:
channel1 <- function(x) {
  cat("Message received from channel 1: ",x,"\n")
}
# Define a callback function to process messages from channel 2:
channel2 <- function(x) {
  cat("Message received from channel 2: ",x,"\n")
}
redisSubscribe(c('channel1','channel2'))
# Monitor channels for at least 1 minute:
t1 <- proc.time()[[3]]
while(proc.time()[[3]] - t1 < 60)
{
  redisMonitorChannels()
  Sys.sleep(0.05)
}
redisUnsubscribe(c('channel1','channel2'))

## End(Not run)
```

redisMove

Move the specified key/value pair to another database.

Description

Move the specified key/value pair in the currently selected database to another database.

Usage

```
redisMove(key, dbindex)
```

Arguments

key The key to move.
dbindex The destination database index number.

Details

This command returns TRUE only if the key was successfully moved, and FALSE if the target key was already there or if the source key was not found at all. It is possible to use redisMove as a locking primitive.

Value

Returns TRUE if the key/value pair was moved, or FALSE otherwise.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSelect](#)

Examples

```
## Not run:
redisConnect()
redisSelect(1)
redisSet('x',5)
redisMove('x',2)
redisSelect(2)
redisGet('x')

## End(Not run)
```

redisMSet	<i>Set one or more key/value pairs in the Redis database.</i>
-----------	---

Description

Set one or more key/value pairs in the Redis database.

Usage

```
redisMSet(keyvalues, NX = FALSE)
```

Arguments

keyvalues	A list of values to set of the form list(key1=value1, key2=value2, ...)
NX	If NX = TRUE, existing values will not be replaced, otherwise they will be.

Details

Set one or more key/value pairs in the Redis database. Existing values will be replaced.

Value

“OK” upon success.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisGet](#)

Examples

```
## Not run:  
redisMSet(list(x=5, y=6))  
  
## End(Not run)
```

redisMulti

redisMulti

Description

Start a transaction block.

Usage

```
redisMulti()
```

Details

Begin a transaction block. All Redis statements issued after `redisMulti` will be queued locally and then sent to the Redis server en masse once the `redisExec` function is issued.

Value

"OK" indicates the transaction has begun.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

redisPublish	<i>redisPublish</i>
--------------	---------------------

Description

Publish data to a Redis message channel.

Usage

```
redisPublish(channel, message)
```

Arguments

channel	The channel name (character).
message	The message (any type, use RAW type for communication with non-R subscribers).

Details

(From the Redis.io documentation): `redisSubscribe`, `redisUnsubscribe` and `redisPublish` implement the Publish/Subscribe messaging paradigm where (citing Wikipedia) senders (publishers) are not programmed to send their messages to specific receivers (subscribers). Rather, published messages are characterized into channels, without knowledge of what (if any) subscribers there may be. Subscribers express interest in one or more channels, and only receive messages that are of interest, without knowledge of what (if any) publishers there are.

Use the Redis function `redisUnsubscribe` to unsubscribe from one or more channels. Service incoming messages on the channels with either `redisGetResponse` or `redisMonitorChannels`.

Use of any other Redis after `redisSubscribe` prior to calling `redisUnsubscribe` will result in an error.

Value

A list conforming to the Redis subscribe response message. Each subscribed channel corresponds to three list elements, the header 'subscribe' followed by the channel name and a count indicating the total number of subscriptions.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSubscribe](#) [redisUnsubscribe](#) [redisPublish](#) [redisGetResponse](#) [redisMonitorChannels](#)

Examples

```
## Not run:
redisConnect()
redisPublish('channel1', charToRaw('A raw character data message example'))

## End(Not run)
```

redisRandomKey	<i>Return a randomly selected key from the currently selected database.</i>
----------------	---

Description

Return a randomly selected key from the currently selected database.

Usage

```
redisRandomKey()
```

Details

Note that this function returns key names, not values.

Value

A character string corresponding to a randomly-selected key from the currently selected database, or a zero-length string if no keys are defined.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

Examples

```
## Not run:
redisConnect()
redisRandomKey()

## End(Not run)
```

redisRename	<i>Rename a key.</i>
-------------	----------------------

Description

Atomically rename a key.

Usage

```
redisRename(old, new, NX = FALSE)
```

Arguments

old	Original key name.
new	New key name.
NX	Set NX = TRUE to prevent overwriting a key that already exists.

Details

If the source and destination name are the same an error is returned. If newkey already exists it is overwritten unless NX = TRUE, in which case the operation fails.

Value

Returns the Redis status "OK" unless NX = TRUE. When NX = TRUE, returns 1 if successful, 0 otherwise.

Author(s)

B. W. Lewis

References

<http://redis.io>

See Also

[redisKeys](#)

Examples

```
## Not run:  
redisConnect()  
redisSet('x', 5)  
redisRename('x', 'y')  
  
## End(Not run)
```

`redisRPop`*Remove the last element from a list.*

Description

Atomically return and remove the last element of the list associated with the specified key. For example if the list contains the elements "a","b","c" `redisRPop` will return "c" and the list will become "a","b".

Usage

```
redisRPop(key)
```

Arguments

`key` The desired key associated with a list.

Value

The first element of the list associated with the specified key, or NULL if the list is empty.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisLPop](#), [redisLPush](#)

Examples

```
## Not run:
redisConnect()
redisLPush('a',1)
redisLPush('a',2)
redisLPush('a',3)
redisRPop('a')

## End(Not run)
```

redisRPopLPush	<i>Remove the tail from a list, pushing to another.</i>
----------------	---

Description

Atomically return and remove the last (tail) element of the src list, and push the element as the first (head) element of the dst list.

Usage

```
redisRPopLPush(src, dest)
```

Arguments

src	A key corresponding to the source list.
dest	A key corresponding to the destination list.

Details

Atomically return and remove the last (tail) element of the src list, and push the element as the first (head) element of the dst list. For example if the source list contains the elements "a","b","c" and the destination list contains the elements "foo","bar" after a `redisRPopLPush` command the content of the two lists will be "a","b" and "c","foo","bar".

If the key does not exist or the list is already empty the special value `NULL` is returned. If the `srckey` and `dstkey` are the same the operation is equivalent to removing the last element from the list and pushing it as first element of the list, so it's a "list rotation" command.

See the Redis reference below for programming examples and discussion.

Value

The value moved or rotated across lists, or `NULL` if the source key does not exist or corresponds to an empty list. An error is thrown if either of the keys does not correspond to a value of 'list' type.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisLPush](#)

Examples

```
## Not run:  
redisConnect()  
redisLPush('x',1)  
redisLPush('x',2)  
redisLPush('x',3)  
redisRPopLPush('x','x')  
redisRPopLPush('x','x')  
  
## End(Not run)
```

redisRPush	<i>Add a value to the head of a list.</i>
------------	---

Description

Add a value to the head of a list corresponding to the specified key.

Usage

```
redisRPush(key, value)
```

Arguments

key	The desired key corresponding to a list.
value	The element to add to the list.

Details

If the key does not exist an empty list is created just before the append operation. If the key exists but is not a list an error is returned.

Value

The length of the list after the successful push operation.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisRPush](#)

Examples

```
## Not run:  
redisConnect()  
redisRPush('x',1)  
redisRPush('x',2)  
redisRPush('x',3)  
  
## End(Not run)
```

redisSAdd	<i>Add an element to a set.</i>
-----------	---------------------------------

Description

Add the element to the specified set.

Usage

```
redisSAdd(set, element)
```

Arguments

set	The string set identifier.
element	The object to add to the set.

Details

If member is already a member of the set no operation is performed. If key does not exist a new set with the specified member as sole member is created. If the key exists but does not hold a set value an error is returned.

Value

TRUE if the element was successfully added, FALSE otherwise (including cases in which the element already belonged to the set).

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSRem](#)

Examples

```
## Not run:  
redisConnect()  
redisSAdd("set", 5)  
  
## End(Not run)
```

redisSave

redisSave

Description

Synchronously save that database to disk.

Usage

```
redisSave()
```

Details

Force Redis to save the database(s) to disk.

Value

Nothing is returned. Check the UNIX time of the last completed save operation with the `redisLastsave` function.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

redisSCard	<i>Set cardinality</i>
------------	------------------------

Description

Return the number of elements contained in the specified set.

Usage

```
redisSCard(set)
```

Arguments

set	A string set identifier.
-----	--------------------------

Value

The function returns an integer corresponding to the number of elements in the set. Zero is returned if the set is empty or if the set does not exist.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSAdd](#)

Examples

```
## Not run:  
redisConnect()  
redisSAdd("set",5)  
redisSCard("set")  
  
## End(Not run)
```

redisSDiff *Return the difference of two or more sets.*

Description

Return the (set) difference of two or more sets.

Usage

```
redisSDiff(keys, ...)
```

Arguments

keys	A vector or list of keys corresponding to sets. May also be a single key.
...	Additional keys corresponding to sets. See the examples below.

Details

The first argument may be a vector of set names, a list of set names, or a single set name. If only a single set name is specified, specify more sets as additional function arguments as shown in the examples.

Value

A list of elements in the difference of the specified sets, or NULL if the intersection is the empty set.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSDiff](#)

Examples

```
## Not run:
redisConnect()
redisSAdd('A',1)
redisSAdd('A',2)
redisSAdd('B',4)
redisSDiff('A','B')
redisSDiff(c('A','B'))
redisSDiff(list('A','B'))
```

```
## End(Not run)
```

redisSDiffStore	<i>Store the difference of two or more sets.</i>
-----------------	--

Description

Store the (set) difference of two or more sets in the specified set.

Usage

```
redisSDiffStore(dest, keys, ...)
```

Arguments

dest	The destination set in which to store the result.
keys	A vector or list of keys corresponding to sets. May also be a single key.
...	Additional keys corresponding to sets. See the examples below.

Details

The keys argument may be a vector of set names, a list of set names, or a single set name. If only a single set name is specified, specify more sets as additional function arguments as shown in the examples.

Value

A redis status code.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSDiff](#)

Examples

```
## Not run:
redisConnect()
redisSAdd('A',1)
redisSAdd('A',2)
redisSAdd('A',3)
redisSAdd('B',2)
redisSDiffStore('C','A','B')

## End(Not run)
```

redisSelect	<i>Select a redis database.</i>
-------------	---------------------------------

Description

Select a database by numeric index value.

Usage

```
redisSelect(index)
```

Arguments

index The nonnegative integer value of the database to select.

Details

Redis supports multiple databases indexed by nonnegative integers. The number of supported databases is configurable via the redis server configuration file. New connections are assigned to database 0 by default.

Value

A character string status code. An error is thrown if the index value is invalid.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

Examples

```
## Not run:
redisConnect()
redisSelect(1)

## End(Not run)
```

redisSet	<i>Store a value in Redis.</i>
----------	--------------------------------

Description

Store a value identified by a character key name in the Redis database. Any existing value with the same key will be replaced by the new value unless NX is set to TRUE.

Usage

```
redisSet(key, value, NX = FALSE)
```

Arguments

key	The (required) character identifier for the value to be stored.
value	The (required) object to associate with the key.
NX	If NX = TRUE, existing values will not be replaced.

Details

The value is copied to the Redis server. The value to be stored can be any serializable R object up to the Redis maximum object size (excluding, for example, external pointer references). References to other R objects or environments inside the value are not guaranteed to be preserved.

In order to store strings that can easily be read by other clients, first convert the character object using the [charToRaw](#) function as shown in the examples.

Value

The value “OK” is returned upon success (conforming to the usual Redis behavior).

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisGet](#)

Examples

```
## Not run:
# Store an R object with the key 'x':
redisSet('x',runif(5))

## End(Not run)
## Not run:
# Store a string that can be easily read by other clients:
redisSet('x',charToRaw('Hello Redis clients'))

## End(Not run)
```

redisSetBlocking	<i>Set the Redis message blocking state.</i>
------------------	--

Description

Use `redisSetBlocking` to set the `rredis` client to blocking (default) or non-blocking Redis communication mode.

Usage

```
redisSetBlocking( value = TRUE )
```

Arguments

`value` TRUE indicates Redis blocking mode, FALSE non-blocking mode.

Details

The `rredis` client blocks for a response from a connected Redis server after each transaction in blocking mode. Blocking mode is the default mode. When in non-blocking mode, transactions are issued without servicing Redis server responses, and server responses must be manually serviced with the `redisGetResponse` function.

Non-blocking mode can improve performance in some circumstances like lots of repeated `redisSet` operations. When using non-blocking mode, don't forget to periodically service responses from the Redis server.

Note that use of commands like `redisMSet` can usually obviate the need to use non-blocking mode.

Value

The new blocking mode (TRUE/FALSE) is invisibly returned.

Author(s)

B. W. Lewis

See Also[redisGetResponse](#)**Examples**

```

## Not run:
redisConnect()
redisSetBlocking(FALSE)
redisLPush('x',pi)
redisLPush('x',exp(1))
redisGetResponse()
redisSetBlocking(TRUE)

## End(Not run)

```

redisSetContext	<i>redisSetContext</i>
-----------------	------------------------

Description

Get or set the current active Redis connection environment.

Usage

```

redisSetContext( e = NULL)
redisGetContext()

```

Arguments

e An environment representing the new Redis server connection context returned by `redisConnect(returnRef=TRUE)`. The default value of `NULL` sets the context to the most recently established connection.

Details

The `redis` package stores information associated with a connection to a Redis server in an environment. The `redisSetContext` and `redisGetContext` functions help manage simultaneous connection to multiple Redis servers.

The `redisSetContext` function returns an environment representing the current active Redis connection. If there is no current active Redis connection, an environment is still returned, but without connection information.

The `redisSetContext` function replaces the current active environment. Any number of simultaneous connections to multiple Redis servers may be managed in this manner.

Value

`NULL` is invisibly returned.

Author(s)

B. W. Lewis

See Also

[redisGetContext](#) [redisConnect](#)

Examples

```
## Not run:
# Open a connection to a Redis server on HOST1 and store its context:
HOST1 <- redisConnect(host='HOST1', returnRef=TRUE)
print(redisInfo())

# Initiate a new Redis context:
HOST2 <- redisConnect(host='HOST2', returnRef=TRUE)
# The connection to HOST2 is now active:
print(redisInfo())

# We may now switch back and forth between the two active connections:
redisSetContext(HOST1)
print(redisInfo())
redisSetContext(HOST2)
print(redisInfo())
redisClose()
redisSetContext(HOST1)
redisClose()

## End(Not run)
```

redisShutdown

redisShutdown

Description

Request the currently currently connected Redis server to shutdown, then close the connection.

Usage

```
redisShutdown()
```

Details

This will terminate the connected Redis server process in an orderly way.

Value

Nothing is returned. An error is thrown if no server is connected.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

redisSInter

Find and return the intersection of two or more sets.

Description

Find and return the intersection of two or more sets.

Usage

```
redisSInter(keys, ...)
```

Arguments

keys	A vector or list of keys corresponding to sets. May also be a single key.
...	Additional keys corresponding to sets. See the examples below.

Details

The first argument may be a vector of set names, a list of set names, or a single set name. If only a single set name is specified, specify more sets as additional function arguments as shown in the examples.

Value

A list of elements in the intersection of the specified sets, or NULL if the intersection is the empty set.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSUnion](#)

Examples

```
## Not run:
redisConnect()
redisSAdd('A',1)
redisSAdd('A',2)
redisSAdd('A',3)
redisSAdd('B',2)
redisSInter('A','B')
redisSInter(c('A','B'))
redisSInter(list('A','B'))

## End(Not run)
```

redisSInterStore	<i>Store intersection of two or more sets.</i>
------------------	--

Description

Find the intersection of two or more sets, storing the result in the specified set.

Usage

```
redisSInterStore(dest, keys, ...)
```

Arguments

dest	The destination set in which to store the result.
keys	A vector or list of keys corresponding to sets. May also be a single key.
...	Additional keys corresponding to sets. See the examples below.

Details

The keys argument may be a vector of set names, a list of set names, or a single set name. If only a single set name is specified, specify more sets as additional function arguments as shown in the examples.

Value

A redis status code.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also[redisSUnionStore](#)**Examples**

```
## Not run:
redisConnect()
redisSAdd('A',1)
redisSAdd('A',2)
redisSAdd('A',3)
redisSAdd('B',2)
redisSInterStore('C','A','B')

## End(Not run)
```

redisSIsMember	<i>Test for set membership</i>
----------------	--------------------------------

Description

Test if an element belongs to a set.

Usage

```
redisSIsMember(set, element)
```

Arguments

set	A string set identifier.
element	The element whose membership is to be tested.

Value

TRUE if the element is a member of the set, FALSE otherwise.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also[redisSAdd](#)

Examples

```
## Not run:
redisConnect()
redisSAdd("set",5)
redisSIsMember("set",5)

## End(Not run)
```

 redisSlaveOf

redisSlaveOf

Description

Change Redis replication settings.

Usage

```
redisSlaveOf(host, port)
```

Arguments

host	The host name of a master Redis server.
port	The port of a master Redis server.

Details

(The following details are adapted from the Redis Wiki manual referenced below.)

The `redisSlaveOf` command can change the replication settings of a slave on the fly. If a Redis server is already acting as slave, the function `redisSlaveOf(host="no",port="one")` will turn off the replication turning the Redis server into a MASTER. In the proper form `redisSlaveOf(hostname,port)` will make the server a slave of the specific server listening at the specified hostname and port.

If a server is already a slave of some master, `redisSlaveOf(hostname,port)` will stop the replication against the old server and start the synchronization against the new one discarding the old dataset.

The form `redisSlaveOf(host="no",port="one")` will stop replication, turning the server into a MASTER but will not discard the replication. So if the old master stop working it is possible to turn the slave into a master and set the application to use the new master in read/write. Later when the other Redis server will be fixed it can be configured in order to work as slave.

Value

A Redis status message is returned.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

redisSMembers	<i>List elements of a set.</i>
---------------	--------------------------------

Description

Return a list containing all the members (elements) of the specified set.

Usage

```
redisSMembers(set)
```

Arguments

set	The set key name (character).
-----	-------------------------------

Value

A list containing the set elements. An error is thrown if the set does not exist.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSAdd](#)

Examples

```
## Not run:  
redisConnect()  
redisSAdd('set',runif(5))  
redisSMembers('set')  
  
## End(Not run)
```

redisSMove	<i>Move a set element.</i>
------------	----------------------------

Description

Move the specified element from one set to another.

Usage

```
redisSMove(setA, setB, element)
```

Arguments

setA	The set to move from (character identifier).
setB	The set to move to (character identifier).
element	The element to move.

Details

Note that the set element is specified as the actual set element object (aka set member), not as an index value.

Value

TRUE if successful, FALSE otherwise.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSAdd](#)

`redisSort`*redisSort*

Description

Sort a list, set or sorted set (zset).

Usage

```
redisSort(key, decreasing = FALSE, alpha = FALSE, by = NULL, start = NULL, count = NULL, get = NULL, store = NULL)
```

Arguments

<code>key</code>	The key name of the list, set or zset to be sorted.
<code>decreasing</code>	Set the direction of the sort.
<code>alpha</code>	Lexicographically sort if true, otherwise try numeric sorting.
<code>by</code>	The BY option takes a pattern that is used in order to generate the key names of the weights used for sorting. Weight key names are obtained substituting the first occurrence of * with the actual value of the elements on the list. See the Redis documentation for examples.
<code>start</code>	Starting index of the sort.
<code>count</code>	Number of entries past start to use for the sort.
<code>get</code>	Retrieve external keys. See the Redis documentation.
<code>store</code>	Store the results in the specified key.

Details

Sort the elements contained in the List, Set, or Sorted Set value at key. By default sorting is numeric with elements being compared as double precision floating point numbers.

Value

A list of sorted values, unless `store` is specified in which case the results are stored in the specified key and TRUE is returned.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

`redisSPop`*Remove and return an element from a set.*

Description

Remove and return an element, chosen at random, from the specified set.

Usage

```
redisSPop(set)
```

Arguments

`set` The set name (character).

Value

A random element of the set, or NULL if the set is empty or does not exist.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSAdd](#)

Examples

```
## Not run:
redisConnect()
redisSAdd('set',runif(2))
redisSAdd('set',runif(3))
redisSAdd('set',runif(4))
redisSPop('set')
redisSPop('set')
redisSPop('set')
redisSPop('set')

## End(Not run)
```

redisSRandMember	<i>Choose a random element from a set</i>
------------------	---

Description

Select and return an element of a set at random.

Usage

```
redisSRandMember(set)
```

Arguments

set	The set key name (character).
-----	-------------------------------

Value

An element from the set selected randomly. An error is thrown if the set does not exist.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSAdd](#)

Examples

```
## Not run:  
redisConnect()  
redisSAdd('set',1)  
redisSAdd('set',2)  
redisSAdd('set',3)  
redisSRandMember('set')  
  
## End(Not run)
```

redisSRem

Remove an element from a set.

Description

Remove an element from a set.

Usage

```
redisSRem(set, element)
```

Arguments

set	The set name (character) from which to remove the element.
element	The element to remove (not an index, but the actual element value).

Details

The set element matching the specified element will be removed from the set.

Value

TRUE upon successful removal, FALSE otherwise.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSAdd](#)

Examples

```
## Not run:
redisConnect()
redisSAdd('set', 5)
redisSAdd('set', 7)
redisSMembers('set')
redisSRem('set', 5)
redisSMembers('set')

## End(Not run)
```

redisSubscribe	<i>redisSubscribe</i>
----------------	-----------------------

Description

Subscribe to one or more Redis message channels.

Usage

```
redisSubscribe(channels, pattern=FALSE)
```

Arguments

channels	A character vector or list of channel names to subscribe to.
pattern	If TRUE, allow wildcard pattern matching in channel names, otherwise names indicate full channel names.

Details

(From the Redis.io documentation): `redisSubscribe`, `redisUnsubscribe` and `redisPublish` implement the Publish/Subscribe messaging paradigm where (citing Wikipedia) senders (publishers) are not programmed to send their messages to specific receivers (subscribers). Rather, published messages are characterized into channels, without knowledge of what (if any) subscribers there may be. Subscribers express interest in one or more channels, and only receive messages that are of interest, without knowledge of what (if any) publishers there are.

Use the Redis function `redisUnsubscribe` to unsubscribe from one or more channels. Service incoming messages on the channels with either `redisGetResponse` or `redisMonitorChannels`.

Use of any other Redis after `redisSubscribe` prior to calling `redisUnsubscribe` will result in an error.

Value

A list conforming to the Redis subscribe response message. Each subscribed channel corresponds to three list elements, the header 'subscribe' followed by the channel name and a count indicating the total number of subscriptions.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSubscribe](#) [redisPublish](#) [redisUnsubscribe](#) [redisGetResponse](#) [redisMonitorChannels](#)

Examples

```
## Not run:
redisConnect()
# Define a callback function to process messages from channel 1:
channel1 <- function(x) {
  cat("Message received from channel 1: ",x,"\n")
}
# Define a callback function to process messages from channel 2:
channel2 <- function(x) {
  cat("Message received from channel 2: ",x,"\n")
}
redisSubscribe(c('channel1','channel2'))
# Monitor channels for at least 1 minute:
t1 <- proc.time()[[3]]
while(proc.time()[[3]] - t1 < 60)
{
  redisMonitorChannels()
  Sys.sleep(0.05)
}
redisUnsubscribe(c('channel1','channel2'))

## End(Not run)
```

redisUnion

Return the union of two or more sets.

Description

Return the union of two or more sets.

Usage

```
redisUnion(keys, ...)
```

Arguments

keys A vector or list of keys corresponding to sets. May also be a single key.
... Additional keys corresponding to sets. See the examples below.

Details

The first argument may be a vector of set names, a list of set names, or a single set name. If only a single set name is specified, specify more sets as additional function arguments as shown in the examples.

Value

A list of elements in the union of the specified sets, or NULL if the intersection is the empty set.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSInter](#)

Examples

```
## Not run:
redisConnect()
redisSAdd('A',1)
redisSAdd('A',2)
redisSAdd('B',4)
redisSUnion('A','B')
redisSUnion(c('A','B'))
redisSUnion(list('A','B'))

## End(Not run)
```

redisSUnionStore	<i>Store the union of two or more sets.</i>
------------------	---

Description

Store the union of two or more sets in the specified set.

Usage

```
redisSUnionStore(dest, keys, ...)
```

Arguments

dest	The destination set in which to store the result.
keys	A vector or list of keys corresponding to sets. May also be a single key.
...	Additional keys corresponding to sets. See the examples below.

Details

The keys argument may be a vector of set names, a list of set names, or a single set name. If only a single set name is specified, specify more sets as additional function arguments as shown in the examples.

Value

A redis status code.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSInterStore](#)

Examples

```
## Not run:
redisConnect()
redisSAdd('A',1)
redisSAdd('A',2)
redisSAdd('A',3)
redisSAdd('B',2)
redisSUnionStore('C','A','B')

## End(Not run)
```

redisTTL

Return the time to live for a key set to expire.

Description

She the time left to live in seconds for the specified key.

Usage

```
redisTTL(key)
```

Arguments

key The key to look up.

Details

Redis keys may be set to expire at or after a given time with the `redisExpire` and `redisExpireAt` functions. This function shows the time left before expiration in seconds for such a key.

Value

Integer time left to live or -1 if the key is not set to expire or not found.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisExpire](#), [redisExpireAt](#)

Examples

```
## Not run:
redisConnect()
redisSet('x', 5)
redisExpire('x', 100)
redisTTL('x')

## End(Not run)
```

redisType

Query a Redis value type.

Description

Return a description of the type of the value corresponding to the specified key.

Usage

```
redisType(key)
```

Arguments

key The key to look up.

Details

The Redis database differentiates three types of values. The value types determine how Redis operates on the key/value pairs, not what kind of data may be contained within. Each value type may store generic R objects, binary blobs, or character strings. The Redis value types are:

string: A single value is associated with the key

list: A stack of values is associated with the key

set: A set of values is associated with the key.

Values of "list" and "set" types have available to them special stack and set operations, respectively.

Value

A descriptive character string that may be one of "none", "string", "list", or "set". "none" is returned if the key does not exist.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisKeys](#)

Examples

```
## Not run:  
redisConnect()  
redisSet('x',5)  
redisLPush('y',6)  
redisType('x')  
redisType('y')  
  
## End(Not run)
```

<code>redisUnsubscribe</code>	<i>redisUnsubscribe</i>
-------------------------------	-------------------------

Description

Subscribe to one or more Redis message channels.

Usage

```
redisUnsubscribe(channels, pattern=FALSE)
```

Arguments

<code>channels</code>	A character vector or list of channel names to subscribe to.
<code>pattern</code>	If TRUE, allow wildcard pattern matching in channel names, otherwise names indicate full channel names.

Details

(From the Redis.io documentation): `redisSubscribe`, `redisUnsubscribe` and `redisPublish` implement the Publish/Subscribe messaging paradigm where (citing Wikipedia) senders (publishers) are not programmed to send their messages to specific receivers (subscribers). Rather, published messages are characterized into channels, without knowledge of what (if any) subscribers there may be. Subscribers express interest in one or more channels, and only receive messages that are of interest, without knowledge of what (if any) publishers there are.

Use the Redis function `redisUnsubscribe` to unsubscribe from one or more channels. Service incoming messages on the channels with either `redisGetResponse` or `redisMonitorChannels`.

Use of any other Redis after `redisSubscribe` prior to calling `redisUnsubscribe` will result in an error.

Value

A list conforming to the Redis subscribe response message. Each subscribed channel corresponds to three list elements, the header element 'unsubscribe' followed by the channel name and a count indicating the total number of subscriptions.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

See Also

[redisSubscribe](#) [redisUnsubscribe](#) [redisPublish](#) [redisGetResponse](#) [redisMonitorChannels](#)

Examples

```
## Not run:
redisConnect()
# Define a callback function to process messages from channel 1:
channel1 <- function(x) {
  cat("Message received from channel 1: ",x,"\n")
}
# Define a callback function to process messages from channel 2:
channel2 <- function(x) {
  cat("Message received from channel 2: ",x,"\n")
}
redisSubscribe(c('channel1','channel2'))
# Monitor channels for at least 1 minute:
t1 <- proc.time()[[3]]
while(proc.time()[[3]] - t1 < 60)
{
  redisMonitorChannels()
  Sys.sleep(0.05)
}
```

```
redisUnsubscribe(c('channel1', 'channel2'))  
## End(Not run)
```

redisUnwatch	<i>redisUnwatch</i>
--------------	---------------------

Description

Stop watching the specified keys for changes.

Usage

```
redisUnwatch(keys)
```

Arguments

keys A character vector or list of key names to stop watching.

Details

Stop watching the specified keys. See `redisWatch` for details.

Value

"OK"

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

`redisWatch`*redisWatch*

Description

Condition a transaction block on value changes.

Usage

```
redisWatch(keys)
```

Arguments

keys A character vector or list of key names to watch for changes.

Details

Watch the specified keys for changed values. If any of the values change for the watched keys, discard the current transaction block and unwatch all watched keys.

Value

"OK"

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

`redisZAdd`*redisZAdd*

Description

Add an element to a Redis sorted set. Sorted sets order their elements by numeric weights.

Usage

```
redisZAdd(key, score, member)
```

Arguments

key	The set name.
score	The numeric score associated with the new element (member).
member	The new object to add to the set.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

redisZCard

redisZCard.Rd

Description

Report the number of elements in a sorted set.

Usage

redisZCard(key)

Arguments

key	The name of the set.
-----	----------------------

Details

See the Redis documentation for details.

Value

The number of elements in the set or zero if the set does not exist. An error is thrown if the key name exists but is not a set.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

redisZIncrBy	<i>redisZIncrBy</i>
--------------	---------------------

Description

Increment the weight associated with an ordered set element.

Usage

```
redisZIncrBy(key, member, increment)
```

Arguments

key	The set name.
member	The set element.
increment	The numeric value by which to increase the weight associated with the element.

Details

See the Redis documentation for details.

Value

The new weight associated with the element.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

redisZInterStore	<i>redisZInterStore</i>
------------------	-------------------------

Description

Store the intersection of two sorted sets into another sorted set.

Usage

```
redisZInterStore(dstkey, keys, weights = c(), aggregate = NULL)
```

Arguments

dstkey	The destination set name.
keys	A vector or list of sorted set names.
weights	A set of weights equal in length to the keys argument, or NULL (default). If specified, the weights associated with the members of each set will be multiplied by the specified weight values.
aggregate	A character value of either "SUM" "MIN" or "MAX" indicating how the resulting weights should be aggregated.

Details

See the Redis documentation for details.

Value

The number of elements stored in the destination set.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

redisZRange

redisZRange

Description

Return a subset of an ordered set.

Usage

```
redisZRange(key, start = 0, end = -1, decreasing = FALSE, withscores = FALSE)
```

Arguments

key	The set name.
start	Starting index value.
end	Ending index value.
decreasing	Set TRUE to sort in decreasing order of scores.
withscores	If TRUE, also return set element scores.

Details

Return the specified elements of the sorted set at the specified key. The elements are considered sorted from the lowest to the highest score if `decreasing = FALSE`, and from highest to lowest score if `decreasing = TRUE`. Start and end are zero-based indexes. 0 is the first element of the sorted set (the one with the lowest score), 1 the next element by score and so on. start and end can also be negative numbers indicating offsets from the end of the sorted set. For example -1 is the last element of the sorted set, -2 the penultimate element and so on.

Indexes out of range will not produce an error: if start is over the end of the sorted set, or start is greater than end, an empty list is returned. If end is over the end of the sorted set Redis will treat it just like the last element of the sorted set.

Value

A list of returned set elements. If `withscores = TRUE`, two lists called 'elements' and 'scores' are returned.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

redisZRangeByScore *redisZRangeByScore*

Description

Return all the elements in the sorted set at key with a score between min and max (including elements with score equal to min or max).

Usage

```
redisZRangeByScore(key, min, max, offset = NULL, count = NULL, withscores = FALSE)
```

Arguments

key	The set name.
min	Minimum score.
max	Maximum score.
offset	Limit the results by excluding the first offset items. Requires also the use of count.
count	Limit the results to the first count items. Requires also the use of offset.
withscores	Also return the scores.

Details

The offset and count arguments must both be specified if used.

Value

See the Redis documentation for more information.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

redisZRank

redisZRank

Description

Return the rank of an element of an ordered set.

Usage

```
redisZRank(key, member, decreasing = FALSE)
```

Arguments

key	The set name.
member	The set element.
decreasing	Set to TRUE to rank in reverse order of score.

Details

Returns the rank of the member in the sorted set, with scores ordered from low to high by default, or from high to low if decreasing = TRUE. The returned rank (or index) of the member is 0-based.

Value

The numeric rank of the set element.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

`redisZRem`*redisZRem*

Description

Delete an element from an ordered set.

Usage

```
redisZRem(key, member)
```

Arguments

key	The set name.
member	The element to delete.

Details

See the Redis documentation for details.

Value

0 or 1: 0 indicates that the element could not be removed, 1 that it was removed.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

`redisZRemRangeByRank`*redisZRemRangeByRank.Rd*

Description

```
redisZRemRangeByRank.Rd
```

Usage

```
redisZRemRangeByRank(key, start, end)
```

Arguments

key
start
end

Details

See the Redis documentation for details.

Value

See the Redis documentation for more information.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

redisZRemRangeByScore *redisZRemRangeByScore.Rd*

Description

redisZRemRangeByScore.Rd

Usage

redisZRemRangeByScore(key, min, max)

Arguments

key
min
max

Details

See the Redis documentation for details.

Value

See the Redis documentation for more information.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

redisZScore	<i>redisZScore</i>
-------------	--------------------

Description

Return the score of an ordered set element.

Usage

redisZScore(key, element)

Arguments

key	The set name.
element	The set element.

Value

The numeric score.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

redisZUnionStore *redisZUnionStore*

Description

Store the union of two sorted sets into another sorted set.

Usage

```
redisZUnionStore(dstkey, keys, weights = c(), aggregate = NULL)
```

Arguments

dstkey	The destination set name.
keys	A vector or list of sorted set names.
weights	A set of weights equal in length to the keys argument, or NULL (default). If specified, the weights associated with the members of each set will be multiplied by the specified weight values.
aggregate	A character value of either "SUM" "MIN" or "MAX" indicating how the resulting weights should be aggregated.

Details

See the Redis documentation for details.

Value

The number of elements stored in the destination set.

Author(s)

B. W. Lewis

References

<http://redis.io/commands>

Index

*Topic **package**

- redis-package, 3
- charToRaw, 24, 35, 65
- redis (redis-package), 3
- redis-package, 3
- redisAuth, 4
- redisBgRewriteAOF, 5
- redisBgSave, 6
- redisBLPop, 6
- redisBRPop, 7
- redisClose, 9, 11
- redisCmd, 10
- redisConnect, 9, 11, 68
- redisDBSize, 12
- redisDecr, 13, 37
- redisDecrBy, 14
- redisDelete, 15
- redisDiscard, 16
- redisExec, 16
- redisExists, 17
- redisExpire, 18, 19, 83
- redisExpireAt, 18, 19, 83
- redisFlushAll, 20, 21
- redisFlushDB, 20, 20
- redisGet, 21, 24, 52, 65
- redisGetContext, 68
- redisGetContext (redisSetContext), 67
- redisGetResponse, 22, 50, 53, 67, 79, 85
- redisGetSet, 23
- redisHDel, 25
- redisHExists, 26
- redisHFields, 27
- redisHGet, 28, 34, 35
- redisHGetAll, 29
- redisHIncrBy, 30
- redisHKeys, 31
- redisHLen, 32
- redisHMGet, 33
- redisHMSet, 33, 34
- redisHSet, 25–33, 35, 36
- redisHVals, 36
- redisIncr, 13, 14, 37, 38
- redisIncrBy, 38
- redisInfo, 39
- redisKeys, 39, 55, 84
- redisLIndex, 40
- redisLLen, 41
- redisLPop, 41, 42, 44, 56
- redisLPush, 42, 43, 46, 47, 56, 57
- redisLRange, 44
- redisLRem, 45
- redisLSet, 46
- redisLTrim, 47
- redisMGet, 48
- redisMonitorChannels, 23, 49, 50, 53, 79, 85
- redisMove, 50
- redisMSet, 48, 51
- redisMulti, 52
- redisPublish, 23, 50, 53, 53, 79, 85
- redisRandomKey, 54
- redisRename, 55
- redisRPop, 42, 56
- redisRPopLPush, 57
- redisRPush, 43, 58, 58
- redisSAdd, 59, 61, 71, 73, 74, 76–78
- redisSave, 60
- redisSCard, 61
- redisSDiff, 62, 62, 63
- redisSDiffStore, 63
- redisSelect, 12, 40, 51, 64
- redisSet, 15, 17, 22, 65
- redisSetBlocking, 66
- redisSetContext, 67
- redisShutdown, 68
- redisSInter, 69, 81
- redisSInterStore, 70, 82
- redisSIsMember, 71

redisSlaveOf, [72](#)
redisSMembers, [73](#)
redisSMove, [74](#)
redisSort, [75](#)
redisSPop, [76](#)
redisSRandMember, [77](#)
redisSRem, [59](#), [78](#)
redisSubscribe, [23](#), [50](#), [53](#), [79](#), [79](#), [85](#)
redisSUnion, [69](#), [80](#)
redisSUnionStore, [71](#), [81](#)
redisTTL, [82](#)
redisType, [83](#)
redisUnsubscribe, [23](#), [50](#), [53](#), [79](#), [84](#), [85](#)
redisUnwatch, [86](#)
redisWatch, [87](#)
redisZAdd, [87](#)
redisZCard, [88](#)
redisZIncrBy, [89](#)
redisZInterStore, [89](#)
redisZRange, [90](#)
redisZRangeByScore, [91](#)
redisZRank, [92](#)
redisZRem, [93](#)
redisZRemRangeByRank, [93](#)
redisZRemRangeByScore, [94](#)
redisZScore, [95](#)
redisZUnionStore, [96](#)