

# Package ‘BiocBaseUtils’

May 5, 2024

**Title** General utility functions for developing Bioconductor packages

**Version** 1.6.0

**Description** The package provides utility functions related to package development. These include functions that replace slots, and selectors for show methods. It aims to coalesce the various helper functions often re-used throughout the Bioconductor ecosystem.

**Imports** methods, utils

**Depends** R (>= 4.2.0)

**Suggests** knitr, rmarkdown, BiocStyle, tinytest

**License** Artistic-2.0

**Encoding** UTF-8

**biocViews** Software, Infrastructure

**BugReports** <https://www.github.com/Bioconductor/BiocBaseUtils/issues>

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.1

**VignetteBuilder** knitr

**Date** 2024-04-29

**git\_url** <https://git.bioconductor.org/packages/BiocBaseUtils>

**git\_branch** RELEASE\_3\_19

**git\_last\_commit** 2a04686

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.19

**Date/Publication** 2024-05-05

**Author** Marcel Ramos [aut, cre] (<<https://orcid.org/0000-0002-3242-0582>>),  
Martin Morgan [ctb],  
Hervé Pagès [ctb]

**Maintainer** Marcel Ramos <marcel.ramos@roswellpark.org>

## Contents

BiocBaseUtils-package . . . . .	2
askUserYesNo . . . . .	2
Assertions . . . . .	3
checkInstalled . . . . .	5
selectSome . . . . .	6
setSlots . . . . .	7

<b>Index</b>	<b>8</b>
--------------	----------

---

BiocBaseUtils-package *BiocBaseUtils: Utility and Internal functions for Bioconductor packages*

---

### Description

BiocBaseUtils is a package aimed at helping the typical Bioconductor developer formalize often written functions that can be seen scattered throughout the Bioconductor ecosystem. Some of these functions include the ability to replace slots in an object. Other functions work to create a nice show method output by selecting some observations.

### Author(s)

**Maintainer:** Marcel Ramos <marcel.ramos@roswellpark.org> ([ORCID](#))

Other contributors:

- Martin Morgan <martin.morgan@roswellpark.org> [contributor]
- Hervé Pagès <hpages.on.github@gmail.com> [contributor]

### See Also

Useful links:

- Report bugs at <https://www.github.com/Bioconductor/BiocBaseUtils/issues>

---

askUserYesNo *Ask user for a yes/no response*

---

### Description

Ask user for a yes/no response

### Usage

```
askUserYesNo(prompt, interactive.only = TRUE)
```

**Arguments**

`prompt` `character()` Question form prompt to display to the user without a question mark

`interactive.only` `logical(1)` If TRUE, the function will only prompt the user when the R session is interactive. If FALSE, the function will always prompt the user.

**Value**

TRUE when user replies with 'yes' to prompt, FALSE when 'no'

**Author(s)**

Martin M.

**Examples**

```
askUserYesNo("Do you want to continue")
```

---

Assertions

*Suite of helper functions to test for types*

---

**Description**

These are a group of helper functions that allow the developer to easily check for common data types in Bioconductor. These include logical, character, and numeric (& integer).

**Usage**

```
isTRUEorFALSE(x, na.ok = FALSE)
```

```
isScalarCharacter(x, na.ok = FALSE, zchar = FALSE)
```

```
isScalarInteger(x, na.ok = FALSE)
```

```
isScalarNumber(x, na.ok = FALSE, infinite.ok = FALSE)
```

```
isScalarLogical(x, na.ok = FALSE)
```

```
isCharacter(x, na.ok = FALSE, zchar = FALSE)
```

```
isZeroOneCharacter(x, na.ok = FALSE, zchar = FALSE)
```

**Arguments**

<code>x</code>	The input vector whose type is to be checked
<code>na.ok</code>	logical(1L) Whether it is acceptable to consider NA type inputs (default: FALSE).
<code>zchar</code>	logical(1L) Whether it is acceptable to consider 'zero' characters as defined by <code>nchar</code> , e.g., <code>nchar("")</code> (default: FALSE).
<code>infinite.ok</code>	logical(1L) Whether it is acceptable to consider infinite values as identified by <code>is.finite</code> (default: FALSE).

**Details**

Some functions such as `isScalarCharacter` allow exceptions to the type checks via the `na.ok` and `zchar` arguments. Others, for example `isScalarNumber` can permit `Inf` with the `infinite.ok` argument.

**Value**

Either TRUE or FALSE

**Functions**

- `isTRUEorFALSE()`: Is the input a single logical vector?
- `isScalarCharacter()`: Is the input a single character vector?
- `isScalarInteger()`: Is the input a single integer vector?
- `isScalarNumber()`: Is the input a single numeric vector?
- `isScalarLogical()`: Is the input a single logical vector?
- `isCharacter()`: Is the input a character vector?
- `isZeroOneCharacter()`: Is the input a character vector of zero or one length?

**Author(s)**

M. Morgan, H. Pagès

**Examples**

```
isTRUEorFALSE(TRUE)
isTRUEorFALSE(FALSE)
isTRUEorFALSE(NA, na.ok = TRUE)

isScalarCharacter(LETTERS)
isScalarCharacter("L")
isCharacter(LETTERS)
isCharacter(NA_character_, na.ok = TRUE)
isZeroOneCharacter("")
isZeroOneCharacter("", zchar = TRUE)

isScalarInteger(1L)
isScalarInteger(1)
```

```
isScalarNumber(1)
isScalarNumber(1:2)
```

---

checkInstalled	<i>Check packages are installed otherwise suggest</i>
----------------	---

---

### Description

checkInstalled allows to check if a package is installed. If the package is not available, a convenient copy-and-paste message is provided for package installation with BiocManager. The function is typically used within functions that check for package availability from the Suggests field.

### Usage

```
checkInstalled(pkgs)
```

### Arguments

pkgs            character() package names required for a function

### Value

TRUE if all packages are installed, otherwise stops with a message and suggests installation of missing packages

### Author(s)

M. Morgan, M. Ramos

### Examples

```
if (interactive()) {
  checkInstalled(
    c("BiocParallel", "SummarizedExperiment")
  )
}
```

---

`selectSome`*Select and return only some entries from a vector*

---

### Description

`selectSome` works well in `show` methods. It abbreviates a vector input depending on the `maxToShow` argument.

### Usage

```
selectSome(  
  obj,  
  maxToShow = 5,  
  ellipsis = "...",  
  ellipsisPos = c("middle", "end", "start"),  
  quote = FALSE  
)
```

### Arguments

<code>obj</code>	<code>character()</code> A vector to be abbreviated for display purposes
<code>maxToShow</code>	<code>numeric(1)</code> The maximum number of values to show in the output (default: 5)
<code>ellipsis</code>	<code>character(1)</code> The symbol used to abbreviate values in the vector (default: "...")
<code>ellipsisPos</code>	<code>character(1)</code> The location for the ellipsis in the output, by default in the "middle" but can be moved to either the "end" or the "start".
<code>quote</code>	<code>logical(1)</code> Whether or not to add a single quote around the <code>obj</code> input. This only works for character type inputs.

### Value

An abbreviated output of `obj`

### Author(s)

M. Morgan, H. Pagès

### Examples

```
letters  
  
selectSome(letters)
```

---

setSlots	<i>Convenience function to set slot values</i>
----------	--

---

**Description**

Given the current object, the function `setSlots` will take name-value pair inputs either as named arguments or a list and replace the values of the specified slots. This is a convenient function for updating slots in an S4 class object.

**Usage**

```
setSlots(object, ..., check = TRUE)
```

**Arguments**

<code>object</code>	An S4 object with slots to replace
<code>...</code>	Slot name and value pairs either as named arguments or a named list, e.g., <code>slotName = value</code> .
<code>check</code>	logical(1L) Whether to run <code>validObject</code> after the slot replacement

**Value**

The object input with updated slot data

**Author(s)**

H. Pagès

**Examples**

```
setClass("A", representation = representation(slotA = "character"))  
  
aclass <- new("A", slotA = "A")  
  
setSlots(aclass, slotA = "B")
```

# Index

`askUserYesNo`, [2](#)

`Assertions`, [3](#)

`BiocBaseUtils` (`BiocBaseUtils`-package), [2](#)

`BiocBaseUtils`-package, [2](#)

`checkInstalled`, [5](#)

`isCharacter` (`Assertions`), [3](#)

`isScalarCharacter` (`Assertions`), [3](#)

`isScalarInteger` (`Assertions`), [3](#)

`isScalarLogical` (`Assertions`), [3](#)

`isScalarNumber` (`Assertions`), [3](#)

`isTRUEorFALSE` (`Assertions`), [3](#)

`isZeroOneCharacter` (`Assertions`), [3](#)

`replaceSlots` (`setSlots`), [7](#)

`selectSome`, [6](#)

`setSlots`, [7](#)