Package ‘BiocIO’

January 15, 2026

Title Standard Input and Output for Bioconductor Packages

Version 1.21.0

Description The `BiocIO` package contains high-level abstract classes and generics used by developers to build IO funcionality within the Bioconductor suite of packages. Implements `import()` and `export()` standard generics for importing and exporting biological data formats. `import()` supports whole-file as well as chunk-wise iterative import. The `import()` interface optionally provides a standard mechanism for 'lazy' access via `filter()` (on row or element-like components of the file resource), `select()` (on column-like components of the file resource) and `collect()`. The `import()` interface optionally provides transparent access to remote (e.g. via https) as well as local access. Developers can register a file extension, e.g., `.loom` for dispatch from character-based URIs to specific `import()` / `export()` methods based on classes representing file types, e.g., `LoomFile()`.

License Artistic-2.0

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 4.3.0)

Imports BiocGenerics, S4Vectors, methods, tools

Suggests testthat, knitr, rmarkdown, BiocStyle

Collate 'BiocFile.R' 'import_export.R' 'compression.R' 'utils.R'

VignetteBuilder knitr

biocViews Annotation,DataImport

BugReports https://github.com/Bioconductor/BiocIO/issues

Date 2024-11-21

git_url https://git.bioconductor.org/packages/BiocIO

git_branch devel

git_last_commit 884e288

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2026-01-15

Author Martin Morgan [aut],
Michael Lawrence [aut],
Daniel Van Twisk [aut],
Marcel Ramos [cre] (ORCID: <`https://orcid.org/0000-0002-3242-0582`>)

Maintainer Marcel Ramos <`marcel.ramos@sph.cuny.edu`>

Contents

`BiocFile-class` *BiocFile class objects*

Description

`BiocFile` is the base virtual class for high-level file abstractions where subclasses are associated with a particular file format or type. It wraps a low-level representation of a file, currently either a path, URL, or connection object. We can represent a list of `BiocFile` objects with a `BiocFileList`.

Usage

```
BiocFileList(files)

resource(x)

resource(x) <- value

## S4 method for signature 'BiocFile'
resource(x)

## S4 replacement method for signature 'BiocFile,character_OR_connection'
resource(x) <- value

fileFormat(x)

## S4 method for signature 'character'
fileFormat(x)
```

```
## S4 method for signature 'BiocFile'
fileFormat(x)

## S4 method for signature 'BiocFile'
path(object, ...)

## S4 method for signature 'BiocFile'
show(object)

FileForFormat(path, format = file_ext(path), prefix = NULL, suffix = "File")

## S4 method for signature 'BiocFile'
as.character(x)
```

Arguments

`files`	`character()` A vector of file paths for the `BiocFileList` constructor
`x`	A `BiocFile` instance
`object`	A `BiocFile` instance
`...`	additional arguments to lower-level functions, not used.
`path, value`	Either a `character` or `connection` object to replace the original resource
`format`	`character(1)` The file extension conducive to a file class name, e.g., `CSVFile`
`prefix`	`character(1)` The prefix to prepend to the format class name, e.g., `Spatial` for a class `SpatialCSV`.
`suffix`	`character(1)` The suffix to append to the format class name, e.g., `File` for a class `CSVFile`.

Value

For constructors, an instance of that class. For extractors such as `resource` and `path`, typically a `character` vector of the file path. For `FileForFormat`, a convenient instance of the class for which the input file corresponds to.

Accessor Methods

In the code snippets below, `x` represents a `BiocFile` object.

- `path(x)`: Gets the path, as a `character` vector, to the resource represented by the `BiocFile` object, if possible.
- `resource(x)`: Gets the low-level resource, either a character vector (a path or URL) or a connection.
- `fileFormat(x)`: Gets a string identifying the file format. Can also be called directly on a character file path, in which case it uses a heuristic based on the file extension.

FileForFormat

The `prefix` and `suffix` arguments are used to filter the class names to those that match the pattern `paste0(prefix, format, suffix)`. If either `prefix` or `suffix` are `NULL`, they are ignored. Note that the search is case insensitive and does require the `format` to be in the name of the class.

Author(s)

Michael Lawrence

See Also

Implementing classes include: BigWigFile, TwoBitFile, BEDFile, GFFFile, WIGFile

Examples

```
## For our examples, we create a class called CSVFILE that extends BiocFile
.CSVFile <- setClass("CSVFile", contains = "BiocFile")

## Constructor
CSVFile <- function(resource) {
    .CSVFile(resource = resource)
}

setMethod("import", "CSVFile", function(con, format, text, ...) {
    read.csv(resource(con), ...)
})

## Define export
setMethod("export", c("data.frame", "CSVFile"),
    function(object, con, format, ...) {
        write.csv(object, resource(con), ...)
    }
)

## Recommend CSVFile class for .csv files
temp <- tempfile(fileext = ".csv")
FileForFormat(temp)

## Create CSVFile
csv <- CSVFile(temp)

## Display path of file
path(csv)

## Display resource of file
resource(csv)
```

`compression` *File compression*

Description

Methods and generics for file compression strategies.

Usage

```
decompress(manager, con, ...)

## S4 method for signature 'ANY'
decompress(manager, con, ...)

## S4 method for signature 'CompressedFile'
decompress(manager, con, ...)

## S4 method for signature 'character'
decompress(manager, con, ...)

## S4 method for signature 'CompressedFile'
fileFormat(x)
```

Arguments

`manager`	The connection manager, defaults to the internal `manager` class
`con`	The connection from which data is loaded or to which data is saved. If this is a `character` vector, it is assumed to be a file name and a corresponding file connection is created and then closed after exporting the object. If it is a BiocFile derivative, the data is loaded from or saved to the underlying resource. If missing, the function will return the output as a character vector, rather than writing to a connection.
`...`	Parameters to pass to the format-specific method.
`x`	A `BiocFile` instance

Value

A decompressed representation of a `CompressedFile` or `character` object

Related functions

- `FileForFormat(path, format = file_ext(path))`: Determines the file type of `path` and returns a high-level file object such as BamFile, BEDFile, BigWigFile, etc.

Examples

```
file <- tempfile(fileext = ".gzip")
decompress(con = file)
```

`IO` *Import and export*

Description

The functions `import` and `export` load and save objects from and to particular file formats.

Usage

```
import(con, format, text, ...)

## S4 method for signature 'connection,character,ANY'
import(con, format, text, ...)

## S4 method for signature 'connection,missing,ANY'
import(con, format, text, ...)

## S4 method for signature 'character,missing,ANY'
import(con, format, text, ...)

## S4 method for signature 'character,character,ANY'
import(con, format, text, ...)

## S4 method for signature 'missing,ANY,character'
import(con, format, text, ...)

export(object, con, format, ...)

## S4 method for signature 'ANY,connection,character'
export(object, con, format, ...)

## S4 method for signature 'ANY,connection,missing'
export(object, con, format, ...)

## S4 method for signature 'ANY,missing,character'
export(object, con, format, ...)

## S4 method for signature 'ANY,character,missing'
export(object, con, format, ...)

## S4 method for signature 'ANY,character,character'
export(object, con, format, ...)

## S4 method for signature 'CompressedFile,missing,ANY'
import(con, format, text, ...)

## S4 method for signature 'ANY,CompressedFile,missing'
```

```
export(object, con, format, ...)
```

Arguments

`con`	The connection from which data is loaded or to which data is saved. If this is a `character` vector, it is assumed to be a file name and a corresponding file connection is created and then closed after exporting the object. If it is a BiocFile derivative, the data is loaded from or saved to the underlying resource. If missing, the function will return the output as a character vector, rather than writing to a connection.
`format`	The format of the output. If missing and con is a file name, the format is derived from the file extension. This argument is unnecessary when con is a derivative of BiocFile.
`text`	If con is missing, this can be a character vector directly providing the string data to import.
`...`	Parameters to pass to the format-specific method.
`object`	The object to export.

Value

If con is missing, a character vector containing the string output. Otherwise, nothing is returned.

Author(s)

Michael Lawrence

See Also

Format-specific options for the popular formats: GFF, BED, Bed15, bedGraph, WIG, BigWig

Examples

```
## To illustrate export(), import(), and yeild(), we create a class, CSVFILE
.CSVFile <- setClass("CSVFile", contains = "BiocFile")

## Constructor
CSVFile <- function(resource) {
    .CSVFile(resource = resource)
}

## Define import
setMethod("import", "CSVFile",
    function(con, format, text, ...) {
        read.csv(resource(con), ...)
    }
)

## Define export
setMethod("export", c("data.frame", "CSVFile"),
    function(object, con, format, ...) {
```

```
        write.csv(object, resource(con), ...)
    }
)

## Usage
temp <- tempfile(fileext = ".csv")
csv <- CSVFile(temp)

export(mtcars, csv)
df <- import(csv)
```

Index