

# Package ‘BiocIO’

October 18, 2022

**Title** Standard Input and Output for Bioconductor Packages

**Version** 1.6.0

**Description** 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

**LazyData** true

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.1.1

**Depends** R (>= 4.0)

**Imports** BiocGenerics, S4Vectors, methods, tools

**Suggests** testthat, knitr, rmarkdown, BiocStyle

**Collate** export.R import.R BiocFile.R compression.R

**VignetteBuilder** knitr

**biocViews** Annotation,DataImport

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

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

**git\_branch** RELEASE\_3\_15

**git\_last\_commit** 60c8aa1

**git\_last\_commit\_date** 2022-04-26

**Date/Publication** 2022-10-18

**Author** Martin Morgan [aut],  
 Michael Lawrence [aut],  
 Daniel Van Twisk [aut],  
 Bioconductor Package Maintainer [cre]

**Maintainer** Bioconductor Package Maintainer <maintainer@bioconductor.org>

## R topics documented:

BiocFile-class . . . . .	2
io . . . . .	3
<b>Index</b>	<b>6</b>

---

BiocFile-class	<i>BiocFile objects</i>
----------------	-------------------------

---

### Description

A BiocFile object is the base class for classes representing files accessible with rtracklayer. It wraps a resource (either a path, URL or connection). We can represent a list of BiocFile objects with a BiocFileList.

### 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.

### Coercion

`as.character(x)`: Returns the path of the file as a character vector.

### 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..

`bestFileFormat(x)`: Returns the best possible file format for a given file. This function searches through loaded packages for "File" classes that contain S4 methods for 'export' and 'import' for that class.

`decompress(x)`: Returns a decompressed representation of a CompressedFile or character object.

**Author(s)**

Michael Lawrence

**See Also**

Implementing classes include: [BigWigFile](#), [TwoBitFile](#), [BEDFile](#), [GFFFile](#), and [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)
```

## Description

The functions `import` and `export` load and save objects from and to particular file formats. The `rtracklayer` package implements support for a number of annotation and sequence formats.

## Usage

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

## Arguments

<code>object</code>	The object to export.
<code>con</code>	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 filename and a corresponding file connection is created and then closed after exporting the object. If a <a href="#">BiocFile</a> 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.
<code>format</code>	The format of the output. If missing and <code>con</code> is a filename, the format is derived from the file extension. This argument is unnecessary when <code>con</code> is a derivative of <a href="#">BiocFile</a> .
<code>text</code>	If <code>con</code> is missing, this can be a character vector directly providing the string data to import.
<code>...</code>	Parameters to pass to the format-specific method.

## 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

- \* **IO**
  - io, 3
- \* **classes**
  - BiocFile-class, 2
- \* **methods**
  - BiocFile-class, 2
- as.character, BiocFile-method  
(BiocFile-class), 2
- bed, 4
- bed15, 4
- BEDFile, 3
- bedgraph, 4
- bestFileFormat (BiocFile-class), 2
- bestFileFormat, GenomicRanges-method  
(BiocFile-class), 2
- bestFileFormat, GRangesList-method  
(BiocFile-class), 2
- bestFileFormat, IntegerRangesList-method  
(BiocFile-class), 2
- bestFileFormat, RleList-method  
(BiocFile-class), 2
- bigwig, 4
- BigWigFile, 3
- BiocFile, 4
- BiocFile-class, 2
- BiocFileList (BiocFile-class), 2
- BiocFileList-class (BiocFile-class), 2
- class:BiocFile (BiocFile-class), 2
- class:CompressedFile (BiocFile-class), 2
- CompressedFile-class (BiocFile-class), 2
- decompress (BiocFile-class), 2
- decompress, ANY-method (BiocFile-class),  
2
- decompress, character-method  
(BiocFile-class), 2
- decompress, CompressedFile-method  
(BiocFile-class), 2
- decompress, GZFile-method  
(BiocFile-class), 2
- export (io), 3
- export, ANY, character, character-method  
(io), 3
- export, ANY, character, missing-method  
(io), 3
- export, ANY, CompressedFile, missing-method  
(io), 3
- export, ANY, connection, character-method  
(io), 3
- export, ANY, connection, missing-method  
(io), 3
- export, ANY, missing, character-method  
(io), 3
- FileForFormat (BiocFile-class), 2
- fileFormat (BiocFile-class), 2
- fileFormat, BiocFile-method  
(BiocFile-class), 2
- fileFormat, character-method  
(BiocFile-class), 2
- fileFormat, CompressedFile-method  
(BiocFile-class), 2
- gff, 4
- GFFFFile, 3
- import (io), 3
- import, character, character, ANY-method  
(io), 3
- import, character, missing, ANY-method  
(io), 3
- import, CompressedFile, missing, ANY-method  
(io), 3
- import, connection, character, ANY-method  
(io), 3
- import, connection, missing, ANY-method  
(io), 3

import,missing,ANY,character-method  
    (io), 3  
io, 3  
  
path (BiocFile-class), 2  
path,BiocFile-method (BiocFile-class), 2  
  
resource (BiocFile-class), 2  
resource<- (BiocFile-class), 2  
  
show,BiocFile-method (BiocFile-class), 2  
  
TwoBitFile, 3  
  
wig, 4  
WIGFile, 3