

# Package: BiocIO (via r-universe)

October 12, 2026

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

**Version** 1.23.3

**Description** The ``BiocIO`` package contains high-level abstract classes and generics used by developers to build IO functionality 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** 7.3.3

**Depends** R (>= 4.3.0)

**Imports** BiocBaseUtils, 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** 2026-04-28

**Repository** <https://link-ny.r-universe.dev>

**Date/Publication** 2026-04-28 22:14:47 UTC

**RemoteUrl** <https://github.com/Bioconductor/BiocIO>  
**RemoteRef** HEAD  
**RemoteSha** 145536de8f8d3c75c341455abc4e83191be6cc04

**Contents**

|                          |          |
|--------------------------|----------|
| BiocFile-class . . . . . | 2        |
| compression . . . . .    | 4        |
| IO . . . . .             | 5        |
| <b>Index</b>             | <b>9</b> |

---

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

---

**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

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

### 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", c(con = "CSVFile"), function(con, format, text, ...) {
  read.csv(resource(con), ...)
})

## Define export
setMethod("export", c(object = "data.frame", con = "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 <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. |
| ...     | 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)
```

### 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 <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.                                                                                                                                                                        |
| 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 <a href="#">BiocFile</a> . |
| 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", c(con = "CSVFile"),
  function(con, format, text, ...) {
    read.csv(resource(con), ...)
  }
)

## Define export
setMethod("export", c(object = "data.frame", con = "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, [5](#)
- \* **classes**
  - BiocFile-class, [2](#)
- \* **methods**
  - BiocFile-class, [2](#)
- as.character, BiocFile-method (BiocFile-class), [2](#)
- BED, [7](#)
- Bed15, [7](#)
- BEDFile, [4](#)
- bedGraph, [7](#)
- BigWig, [7](#)
- BigWigFile, [4](#)
- BiocFile, [5–7](#)
- BiocFile (BiocFile-class), [2](#)
- BiocFile-class, [2](#)
- BiocFileList (BiocFile-class), [2](#)
- BiocFileList-class (BiocFile-class), [2](#)
- compress (compression), [4](#)
- CompressedFile-class (compression), [4](#)
- compression, [4](#)
- decompress (compression), [4](#)
- decompress, ANY-method (compression), [4](#)
- decompress, character-method (compression), [4](#)
- decompress, CompressedFile-method (compression), [4](#)
- decompress, GZFile-method (compression), [4](#)
- export (IO), [5](#)
- export, ANY, character, character-method (IO), [5](#)
- export, ANY, character, missing-method (IO), [5](#)
- export, ANY, CompressedFile, missing-method (IO), [5](#)
- export, ANY, connection, character-method (IO), [5](#)
- export, ANY, connection, missing-method (IO), [5](#)
- export, ANY, missing, character-method (IO), [5](#)
- FileForFormat (BiocFile-class), [2](#)
- fileFormat (BiocFile-class), [2](#)
- fileFormat, BiocFile-method (BiocFile-class), [2](#)
- fileFormat, character-method (BiocFile-class), [2](#)
- fileFormat, CompressedFile-method (compression), [4](#)
- GFF, [7](#)
- GFFFile, [4](#)
- import (IO), [5](#)
- import, character, character, ANY-method (IO), [5](#)
- import, character, missing, ANY-method (IO), [5](#)
- import, CompressedFile, missing, ANY-method (IO), [5](#)
- import, connection, character, ANY-method (IO), [5](#)
- import, connection, missing, ANY-method (IO), [5](#)
- import, missing, ANY, character-method (IO), [5](#)
- IO, [5](#)
- path (BiocFile-class), [2](#)
- path, BiocFile-method (BiocFile-class), [2](#)
- resource (BiocFile-class), [2](#)

resource,BiocFile-method  
    (BiocFile-class), [2](#)  
resource<- (BiocFile-class), [2](#)  
resource<-,BiocFile,character\_OR\_connection-method  
    (BiocFile-class), [2](#)  
  
show,BiocFile-method (BiocFile-class), [2](#)  
  
TwoBitFile, [4](#)  
  
WIG, [7](#)  
WIGFile, [4](#)