Package: RcppCNPy (via r-universe)

October 12, 2026

Type Package

Title Read-Write Support for 'NumPy' Files via 'Rcpp'

Version 0.2.15

Date 2026-03-13

Description The 'cnpy' library written by Carl Rogers provides read and write facilities for files created with (or for) the 'NumPy' extension for 'Python'. Vectors and matrices of numeric types can be read or written to and from files as well as compressed files. Support for integer files is available if the package has been built with as C++11 which should be the default on all platforms since the release of R 3.3.0.

URL https://github.com/eddelbuettel/rcppcnpy, https://dirk.eddelbuettel.com/code/rcpp.cnpy.html

BugReports https://github.com/eddelbuettel/rcppcnpy/issues

License GPL (>= 2)

Depends R (>= 3.1.0)

Imports methods, Rcpp

LinkingTo Rcpp

Suggests reticulate, rbenchmark

NeedsCompilation yes

Author Dirk Eddelbuettel [aut, cre] (ORCID: <https://orcid.org/0000-0001-6419-907X>), Wush Wu [aut] (ORCID: <https://orcid.org/0000-0001-5180-0567>), Carl Rogers [aut] (Author of CNPy)

Maintainer Dirk Eddelbuettel <edd@debian.org>

Repository https://cran.r-universe.dev

Date/Publication 2026-03-13 13:10:27 UTC

RemoteUrl https://github.com/cran/RcppCNPy

RemoteRef HEAD

RemoteSha 17eef9d12e004a3e02de3127c0b69f1d070b957a

Contents

RcppCNPy-package 2

Index **4**

`RcppCNPy-package` *File access to data files written by (or for) NumPy (Numeric Python) modules*

Description

This package provides access to the `cnpy` library by Carl Rogers which provides read and write facilities for files created with (or for) the NumPy extension for Python.

Support is provided to reading and writing of either vectors or matrices of numeric or integer types.

Files with `gzip` compression can be transparently read and written as well.

Usage

```
npyLoad(filename, type="numeric", dotranspose=TRUE)
npySave(filename, object, mode="w", checkPath=TRUE)
npyHasIntegerSupport()
```

Arguments

`filename`	string with (path and) filename for a `npy` object file. If the string ends with `.gz`, compressed files can be read or written.
`type`	string with type 'numeric' (default) or 'integer'.
`object`	an R object, currently limited to a vector or matrix of either integer or numeric type
`dotranspose`	a boolean variable indicating whether a two-dimensional object should be transposed after reading, default is true
`mode`	a one-character string indicating whether files are appended to ("a") or written ("w", the default). In case of writing `gzip`-ed file, this option is not supported as such files can only be (over-)written, and bot appended.
`checkPath`	a boolean variable indicating whether a path implied in the `filename` argument is to be checked for existing directories, default is true.

Details

The package uses Rcpp modules to provide R bindings `npyLoad()` and `npySave()` which wrap the `npy_load()` and `npy_save()` functions. Currently, only one- and two-dimensional vectors and matrices are suppported; higher-dimensional arrays could be added.

Integer support requires access to the `long long` type which is available if the package is built using the C++11 standard; this is the default since release 0.2.3 which came out after R 3.1.0 permitted use of C++11 in CRAN packages.

Note that R uses only one `integer` type (which uses 32 bits) and one `double` floating point type (which uses 64 bits). If Python data of either type with a different bitsize is to be shared with R, it has be cast to the corresponding width used by R first.

Author(s)

Dirk Eddelbuettel provided the binding to R (using the Rcpp package).

Carl Rogers wrote the underlying `cnpy` library, which is released under the MIT license.

Maintainer: Dirk Eddelbuettel <edd@debian.org>

References

Rcpp, in particular the Rcpp modules documentation.

The `cnpy` repository: https://github.com/rogersce/cnpy

See Also

Rcpp

Examples

```
## Not run:
    library(RcppCNPy)

    ## load NumPy file with floating-point data
    fmat <- npyLoad("fmat.npy")

    ## load NumPy file with integer data
    imat <- npyLoad("imat.npy", "integer")

    ## save floating-point data: matrix and vector
    M <- matrix(0:11, 3, 4, byrow=TRUE) * 1.1
    v <- v <- 0:4 * 1.1
    npySave("fmat.npy", M)
    npySave("fvec.npy", v)

    ## save integer data: matrix and vector
    M <- matrix(0:11, 3, 4, byrow=TRUE)
    v <- v <- 0:4
    npySave("imat.npy", M)
    npySave("ivec.npy", v)


## End(Not run)
```

Index

∗ **package**
RcppCNPy-package, 2

npyHasIntegerSupport
*(*RcppCNPy-package*)*, 2
npyLoad *(*RcppCNPy-package*)*, 2
npySave *(*RcppCNPy-package*)*, 2

Rcpp, *3*
RcppCNPy *(*RcppCNPy-package*)*, 2
RcppCNPy-package, 2