# Package: c3dr (via r-universe)

October 11, 2026

**Title** Read and Write C3D Motion Capture Files

**Version** 0.2.0.9000

**Description** A wrapper for the 'EZC3D' library to work with C3D motion capture data.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**URL** https://github.com/ropensci/c3dr, https://docs.ropensci.org/c3dr/

**BugReports** https://github.com/ropensci/c3dr/issues

**LinkingTo** Rcpp

**Imports** Rcpp

**Suggests** testthat (>= 3.0.0), quarto

**Config/testthat/edition** 3

**VignetteBuilder** quarto

**Config/roxygen2/version** 8.0.0

**Repository** https://ropensci.r-universe.dev

**Date/Publication** 2026-09-11 07:55:10 UTC

**RemoteUrl** https://github.com/ropensci/c3dr

**RemoteRef** main

**RemoteSha** 66ca01f60d6515aeccc86458c5bed99b9d61ae23

## Contents


c3d_analog . . . . . 2
c3d_convert . . . . . 3
c3d_data . . . . . 4
c3d_example . . . . . 5
c3d_read . . . . . 6
c3d_setdata . . . . . 7

c3d_write . . . . . 8
format.c3d . . . . . 9
print.c3d . . . . . 10

**Index** **11**

---

`c3d_analog` *Get analog data from a c3d object*

---

## Description

Get the analog data of a c3d object in a data frame.

## Usage

```
c3d_analog(x)
```

## Arguments

| | |
|---|---|
| `x` | A c3d object, as imported by `c3d_read()`. |

## Details

The analog data of imported c3d objects in `c3dr` is saved as a list of lists. This is good for internal handling, but for analysis a table format (a data frame) is often more convenient.`c3d_analog()` returns the analog data from an imported c3d object as a data frame.

## Value

A data.frame with with n rows and m columns, where n is the number of frames recorded times the number of analog subframes per frame, and m as the number of recorded analog channels.

## Examples

```
# Import example data
d <- c3d_read(c3d_example())

# get analog data
a <- c3d_analog(d)
head(a)
```

c3d_convert *Convert between c3d point data formats*

## Description

Convert between different representations of point data in c3d files.

## Usage

```
c3d_convert(data, format)
```

## Arguments

| | |
|---|---|
| data | A data frame of class c3d_data as generated by c3d_data(). The data can have any of the three data formats (see Data Formats section below). |
| format | Either "wide" (default), "long", or "longest" to determine the format of the resulting data frame. See the Data Formats section for more details. |

## Value

A data frame of class c3d_data in the format specified by the format argument.

## Data Formats

### Wide:

The wide format has three numeric columns per point (x, y, z). The column names have the structure pointname_type, so for example the x-coordinate of a point named C7 has the name C7_x. Each row corresponds to one recording frame.

### Long:

The long format has one column per point. The column names correspond to the names of the points. Each recording frame corresponds to three rows of data (x, y, z). In additional to the point columns containing numeric data there are two additional columns: One that indicates the frame number (frame, numeric) and one that indicates the coordinate type (type, either x, y, or z, as a character).

### Longest:

The longest format has one data column (value, numeric). The other columns indicate the frame number (frame, numeric), the coordinate type (type, either x, y, or z, as a character), and the point name (point, character). Thus, each row of the data frame corresponds to one data entry.

## Examples

```
# Import example data
d <- c3d_read(c3d_example())
# get point data in wide format
w <- c3d_data(d, format = "wide")
# convert to long data
```

```
l <- c3d_convert(w, "long")
head(l)
```

---

`c3d_data` *Get point data from a c3d object*

---

## Description

Get the point data of an c3d object in a data frame.

## Usage

```
c3d_data(x, format = "wide")
```

## Arguments

| | |
|---|---|
| `x` | A c3d object, as imported by `c3d_read()`. |
| `format` | Either `"wide"` (default), `"long"`, or `"longest"` to determine the format of the resulting data frame. See the Data Formats section for more details. |

## Details

The point data of imported c3d objects in `c3dr` is saved as a list of lists. This is good for internal handling, but for analysis a table format (a data frame) is often more convenient.`c3d_data()` returns the point data from an imported c3d object as a data frame.

Analyses of data frames may require them to have different formats. For `c3d_data` output, different data formats (`"wide"`, `"long"`, `"longest"`) are available. See the section below for more details. You can convert between different formats with `c3d_convert()`.

## Value

A data frame of class `c3d_data` with the c3d point data. The structure of the data frame depends on the 'format' argument.

## Data Formats

### Wide:

The wide format has three numeric columns per point (x, y, z). The column names have the structure `pointname_type`, so for example the x-coordinate of a point named `C7` has the name `C7_x`. Each row corresponds to one recording frame.

### Long:

The long format has one column per point. The column names correspond to the names of the points. Each recording frame corresponds to three rows of data (x, y, z). In additional to the point columns containing numeric data there are two additional columns: One that indicates the frame number (`frame`, numeric) and one that indicates the coordinate type (`type`, either x, y, or z, as a character).

**Longest:**
The longest format has one data column (`value`, numeric). The other columns indicate the frame number (`frame`, numeric), the coordinate type (`type`, either x, y, or z, as a character), and the point name (`point`, character). Thus, each row of the data frame corresponds to one data entry.

## Examples

```
# Import example data
d <- c3d_read(c3d_example())

# wide format (default)
w <- c3d_data(d)
head(w)

# long format
l <- c3d_data(d, format = "long")
head(l)

# longest format
ll <- c3d_data(d, format = "longest")
head(ll)
```

---

`c3d_example` *Get path to c3dr example*

---

## Description

Return the file path for an example data files within the `c3dr` package.

## Usage

```
c3d_example()
```

## Details

The test data file contains a short recording of human walking using a full-body model. The test data includes analog channels (e.g., EMG) and data from two force platforms. The recording was made with a Qualisys motion capture system.

The file is taken from https://github.com/pyomeca/ezc3d-testFiles under a GPL-3.0 license.

## Value

A character vector with the absolute file path of the example file.

## Examples

```
c3d_example()
```

`c3d_read` *Read a c3d file in R*

## Description

Import a c3d file using the C++ ezc3d library.

## Usage

```
c3d_read(file)
```

## Arguments

| | |
|---|---|
| `file` | A string with the path of a c3d file. |

## Details

This function reads a c3d file with biomechanical data. It returns a c3d object, which is a list of all imported data.

The resulting c3d object has the following entries:

- **header**: A list with header parameters containing general meta data for the recording. `nframes` is the total number of frames recorded. `npoints` is the total number of points recorded. `nanalogs` is the number of analog channels. `analogperframe` is the rate of analog frames per point recording frame. `framerate` is the number of point frames per second. `nevents` is the number of recorded events.
- **parameters**: A list with meta data of the recording. The parameters are organized in groups, similarly to the original structure in the c3d file. c3dr tries to preserve the data type and data structure of each imported parameter value. One-dimensional parameters are imported as a scalar or vector, two-dimensional parameters are imported as a matrix, three-dimensional parameters are imported as a list of matrices.
- **data**: A list with the point data of the recording. Each element in the list corresponds to one frame. Use `c3d_data()` to convert the data to a data frame.
- **analog**: A list with the analog data of the recording. Each element of the list corresponds to one frame of the point recording and contains a matrix with all analog channels (as columns) for all subframes (as rows). Use `c3d_analog()` to convert the data to a data frame.
- **forceplatform**: A list with force platform data, if available. Each element in the list corresponds to one force platform. Each force platform is another list with the following elements: `forces` is a matrix of the forces. `moments` is a matrix of the moments. `tz` is a matrix of the moments on the center of pressure. `meta` is a list with further meta data of the force platform recording (`frames`, `funit` unit of force, `munit` unit of moments, `punit` unit of center of pressure position, `calmatrix` calibration matrix, `corners` position of the corners, `origin` position of the origin).

## Value

A list of class `c3d`.

## Examples

```
# get example data path
path <- c3d_example()

d <- c3d_read(path)
str(d)
```

---

`c3d_setdata` *Write data to a c3d object*

---

## Description

Set new data to an existing c3d object.

## Usage

```
c3d_setdata(x, newdata = NULL, newanalog = NULL)
```

## Arguments

| | |
|---|---|
| `x` | A `c3d` object to be modified. |
| `newdata` | The new point data that should be written to the `c3d` object. Usually a data frame of the class `c3d_data` as it is generated by `c3d_data()`. Defaults to `NULL`, which means that the point data will remain unchanged. The new point data can be in any format (wide, long, longest), but take care that the conventions of the format are met (see `c3d_data()` for details). |
| `newanalog` | The new analog data that should be written to the `c3d` object. Usually a data frame of the class `c3d_analog` as it is generated by `c3d_analog()`. Defaults to `NULL`, which means that the analog data will remain unchanged. |

## Details

This is a basic helper function to allow the modification of data within the `c3dr` package for later export. The function call updates the data (point and/or analog) and the appropriate parameters and header sections. Note that not all parameters can be updated based on insufficient information. For example, when using `c3d_setdata()` for updating the point data, the point label parameter gets updated (based on the column headers), but the point label descriptions will be unmodified. This can create minor inconsistencies in the resulting c3d object, which in the worst case can lead to corrupt data after export with `c3d_write()`. If you plan heavy modifications of the data before export make sure to manually check and update all relevant parameters as well as the residual data after calling `c3d_setdata()`.

## Value

The modified c3d object.

## Examples

```
# Import example data
d <- c3d_read(c3d_example())

# remove last frame from point data and analog data (10 subframes for analog)
d_cut <- c3d_data(d)[-340, ]
a_cut <- c3d_analog(d)[-(3391:3400), ]

# write the new c3d object
d_new <- c3d_setdata(d, newdata = d_cut, newanalog = a_cut)
d_new
```

---

`c3d_write` *Write a c3d file in R*

---

## Description

Write a c3d file using the C++ ezc3d library.

## Usage

```
c3d_write(x, file)
```

## Arguments

| | |
|---|---|
| `x` | A c3d object. |
| `file` | A string with the file path to write to. |

## Details

This function takes an c3d object in R and writes it to a c3d file. The function creates a new c3d file from scratch and inserts all point data, analog data and parameters in the file. Note that the resulting file will show minor discrepancies compared to the original file (e.g., in terms of file structure). During import and export minor rounding errors can occur.

Force platform data is exported by writing the analog channels with raw force data and the corresponding parameters of the `FORCE_PLATFORM` group for processing. This means that in the current version, modifications to the processed force platform data (`obj$forceplatform`) will not be exported. Make modifications to the analog channels holding the raw data instead. If you delete analog channels in your data, make sure that `obj$parameters$FORCE_PLATFORM$CHANNEL` still points to the correct indices.

The header parameters of the c3d object will not be exported but recreated based on the parameter section. If you want to change the header you should change the appropriate parameters instead.

Be cautious when writing a modified c3d object to an c3d file, as internal inconsistencies may lead to corrupt files. `c3d_write()` and the underlying ezc3d function perform some basic checks but may fail if, for example, parameters and data are inconsistent. You can use the helper function `c3d_setdata()` for modifying point or analog data of a c3d object. Larger modifications may requires expert knowledge of the c3d file structure and parameters.

## Value

Returns its input invisible.

## Examples

```
# read an example file
d <- c3d_read(c3d_example())

# create a temporary file
tmp <- tempfile()
on.exit(unlink(tmp))

# write c3d file
c3d_write(d, tmp)
```

---

`format.c3d` *Formatting c3d objects*

---

## Description

Formatting method for c3d objects

## Usage

```
## S3 method for class 'c3d'
format(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `list` of the class `c3d` to be formatted. |
| `...` | empty argument, currently not used. |

## Value

A character string with basic information for the c3d object.

## Examples

```
# Import example data
d <- c3d_read(c3d_example())

format(d)
```

print.c3d *Printing c3d objects*

## Description

Printing method for c3d objects

## Usage

```
## S3 method for class 'c3d'
print(x, ...)
```

## Arguments

| | |
|---|---|
| x | A list of the class c3d to be printed. |
| ... | empty argument, currently not used. |

## Details

Prints c3d objects by calling format.c3d().

## Value

The function prints basic information for the c3d object and returns it invisibly.

## Examples

```
# Import example data
d <- c3d_read(c3d_example())

print(d)
```

# Index

c3d_analog, 2
c3d_analog(), *6, 7*
c3d_convert, 3
c3d_convert(), *4*
c3d_data, 4
c3d_data(), *3*, *6, 7*
c3d_example, 5
c3d_read, 6
c3d_read(), *2*, *4*
c3d_setdata, 7
c3d_setdata(), *8*
c3d_write, 8
c3d_write(), *7*

format.c3d, 9
format.c3d(), *10*

print.c3d, 10