

# Package ‘csvwr’

February 8, 2026

**Type** Package

**Title** Read and Write CSV on the Web (CSVW) Tables and Metadata

**Version** 0.1.7

**Author** Robin Gower

**Maintainer** Robin Gower <csvwr@infonomics.ltd.uk>

**Description** Provide functions for reading and writing CSVW - i.e. CSV tables and JSON metadata.  
The metadata helps interpret CSV by setting the types and variable names.

**License** GPL-3

**URL** <https://robsteranium.github.io/csvwr/>,  
<https://github.com/Robsteranium/csvwr>

**BugReports** <https://github.com/Robsteranium/csvwr/issues>

**Encoding** UTF-8

**Suggests** testthat (>= 3.0.0), knitr, markdown, rmarkdown

**Imports** cli, magrittr, jsonlite, purrr, readr, stringr, rlang

**Config/testthat/edition** 3

**RoxygenNote** 7.2.1

**VignetteBuilder** knitr, rmarkdown

**Language** en-GB

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2022-11-21 11:20:02 UTC

## Contents

|                           |   |
|---------------------------|---|
| base_uri . . . . .        | 2 |
| coalesce_truth . . . . .  | 3 |
| create_metadata . . . . . | 3 |
| csvwr_example . . . . .   | 4 |
| csvw_to_list . . . . .    | 4 |

|                                          |    |
|------------------------------------------|----|
| datatype_to_type . . . . .               | 5  |
| default_dialect . . . . .                | 5  |
| default_schema . . . . .                 | 6  |
| derive_metadata . . . . .                | 6  |
| derive_table_schema . . . . .            | 7  |
| extract_table . . . . .                  | 7  |
| normalise_metadata . . . . .             | 8  |
| read_csvw . . . . .                      | 8  |
| read_csvw_dataframe . . . . .            | 9  |
| read_metadata . . . . .                  | 9  |
| render_uri_templates . . . . .           | 10 |
| transform_datetime_format . . . . .      | 11 |
| type_to_datatype . . . . .               | 11 |
| validate_csvw . . . . .                  | 12 |
| validate_referential_integrity . . . . . | 12 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>13</b> |
|--------------|-----------|

---

|          |                                                 |
|----------|-------------------------------------------------|
| base_uri | <i>Retrieve the base URI from configuration</i> |
|----------|-------------------------------------------------|

---

**Description**

Retrieve the base URI from configuration

**Usage**

base\_uri()

**Value**

returns the value of csvwr\_base\_uri option, defaulting to example.net

**Examples**

```
## Not run:
base_uri() # returns default

options(csvwr_base_uri="http://www.w3.org/2013/csvw/tests/")
base_uri()

## End(Not run)
```

---

|                |                                     |
|----------------|-------------------------------------|
| coalesce_truth | <i>Coalesce value to truthiness</i> |
|----------------|-------------------------------------|

---

**Description**

Determine whether the input is true, with missing values being interpreted as false.

**Usage**

```
coalesce_truth(x)
```

**Arguments**

|   |                     |
|---|---------------------|
| x | logical, NA or NULL |
|---|---------------------|

**Value**

FALSE if x is anything but TRUE

---

|                 |                                                      |
|-----------------|------------------------------------------------------|
| create_metadata | <i>Create tabular metadata from a list of tables</i> |
|-----------------|------------------------------------------------------|

---

**Description**

The table annotations should each be a list with keys for url and tableSchema. You can use derive\_table\_schema to derive a schema from a data frame.

**Usage**

```
create_metadata(tables)
```

**Arguments**

|        |                                  |
|--------|----------------------------------|
| tables | a list of csvw:table annotations |
|--------|----------------------------------|

**Value**

a list describing a tabular metadata annotation

**Examples**

```
d <- data.frame(foo="bar")
table <- list(url="filename.csv", tableSchema=derive_table_schema(d))
create_metadata(tables=list(table))
```

---

|               |                                  |
|---------------|----------------------------------|
| csvwr_example | <i>Get path to csvwr example</i> |
|---------------|----------------------------------|

---

### Description

The csvwr package includes some example csvw files in its inst/extdata directory. You can use this function to find them.

### Usage

```
csvwr_example(path = NULL)
```

### Arguments

|      |                                                          |
|------|----------------------------------------------------------|
| path | The filename. If NULL, the example files will be listed. |
|------|----------------------------------------------------------|

### Details

Inspired by [readr::readr\\_example\(\)](#)

### Value

either a file path or a vector of filenames

### Examples

```
csvwr_example()  
csvwr_example("computer-scientists.csv")
```

---

|              |                                                     |
|--------------|-----------------------------------------------------|
| csvw_to_list | <i>Convert a csvw metadata to a list (csv2json)</i> |
|--------------|-----------------------------------------------------|

---

### Description

Convert a csvw metadata to a list (csv2json)

### Usage

```
csvw_to_list(csvw)
```

### Arguments

|      |                      |
|------|----------------------|
| csvw | a csvw metadata list |
|------|----------------------|

### Value

a list following the csv2json translation rules

**Examples**

```
## Not run:
csvw_to_list(read_csvw("example.csv"))

## End(Not run)
```

---

|                  |                                      |
|------------------|--------------------------------------|
| datatype_to_type | <i>Map csvw datatypes to R types</i> |
|------------------|--------------------------------------|

---

**Description**

Translate **csvw datatypes** to R types. This implementation currently targets **readr::cols** column specifications.

**Usage**

```
datatype_to_type(datatypes)
```

**Arguments**

datatypes      a list of csvw datatypes

**Value**

a readr::cols specification - a list of collectors

**Examples**

```
## Not run:
cspec <- datatype_to_type(list("double", list(base="date", format="yyyy-MM-dd")))
readr::read_csv(readr::readr_example("challenge.csv"), col_types=cspec)

## End(Not run)
```

---

|                 |                             |
|-----------------|-----------------------------|
| default_dialect | <i>CSVW default dialect</i> |
|-----------------|-----------------------------|

---

**Description**

The **CSVW Default Dialect specification** described in **CSV Dialect Description Format**.

**Usage**

```
default_dialect
```

**Format**

An object of class list of length 13.

**Value**

a list specifying a default csv dialect

---

|                |                                                            |
|----------------|------------------------------------------------------------|
| default_schema | Create a default table schema given a csv file and dialect |
|----------------|------------------------------------------------------------|

---

**Description**

If neither the table nor the group have a tableSchema annotation, then this default schema will be used.

**Usage**

```
default_schema(filename, dialect = default_dialect)
```

**Arguments**

- filename            a csv file
- dialect            specification of the csv's dialect (default: default\_dialect)

**Value**

a table schema

---

|                 |                                      |
|-----------------|--------------------------------------|
| derive_metadata | Derive csvw metadata from a csv file |
|-----------------|--------------------------------------|

---

**Description**

Derive csvw metadata from a csv file

**Usage**

```
derive_metadata(filename)
```

**Arguments**

- filename            a csv file

**Value**

a list of csvw metadata

**Examples**

```
derive_metadata(csvwr_example("computer-scientists.csv"))
```

---

|                     |                                                   |
|---------------------|---------------------------------------------------|
| derive_table_schema | <i>Derive csvw table schema from a data frame</i> |
|---------------------|---------------------------------------------------|

---

**Description**

Derive csvw table schema from a data frame

**Usage**

```
derive_table_schema(d)
```

**Arguments**

d                      a data frame

**Value**

a list describing a csvw:tableSchema

**Examples**

```
derive_table_schema(data.frame(a=1,b=2))
```

---

|               |                                                      |
|---------------|------------------------------------------------------|
| extract_table | <i>Extract a referenced table from CSVW metadata</i> |
|---------------|------------------------------------------------------|

---

**Description**

Extract a referenced table from CSVW metadata

**Usage**

```
extract_table(csvw, reference)
```

**Arguments**

csvw                      the metadata  
reference                  a foreign key reference expressed as a list containing either a reference attribute or a schemaReference attribute

**Value**

a csvw table

---

|                    |                           |
|--------------------|---------------------------|
| normalise_metadata | <i>Normalise metadata</i> |
|--------------------|---------------------------|

---

**Description**

The spec defines a **normalisation process**.

**Usage**

```
normalise_metadata(metadata, location)
```

**Arguments**

|          |                              |
|----------|------------------------------|
| metadata | a csvw metadata list         |
| location | the location of the metadata |

**Value**

metadata with normalised properties

---

|           |                            |
|-----------|----------------------------|
| read_csvw | <i>Read CSV on the Web</i> |
|-----------|----------------------------|

---

**Description**

If the argument to filename is a json metadata document, this will be used to find csv files for each table using the value of csvw:url.

**Usage**

```
read_csvw(filename, metadata = NULL)
```

**Arguments**

|          |                                                    |
|----------|----------------------------------------------------|
| filename | a path for a csv table or a json metadata document |
| metadata | optional user metadata                             |

**Details**

If the argument to filename is a csv file, and no metadata is provided, an attempt is made to derive metadata.

If the argument to filename is a csv file, and the metadata is provided, then the given csv will override the value of csvw:url.

The csvw metadata is returned as a list. In each table in the table group, an element named dataframe is added which provides the contents of the csv table parsed into a data frame using the table schema.

**Value**

csvw metadata list, with a dataframe property added to each table

**Examples**

```
## Not run:
read_csvw("metadata.json")
read_csvw("table.csv", "metadata.json")

## End(Not run)
```

---

|                     |                                                         |
|---------------------|---------------------------------------------------------|
| read_csvw_dataframe | <i>Read a data frame from the first table in a csvw</i> |
|---------------------|---------------------------------------------------------|

---

**Description**

Wrapper around read\_csvw convenient when you're only interested in the data and there's only one table

**Usage**

```
read_csvw_dataframe(filename, metadata = NULL)
```

**Arguments**

|          |                                                    |
|----------|----------------------------------------------------|
| filename | a path for a csv table or a json metadata document |
| metadata | optional user metadata                             |

**Value**

A data frame parsed using the table schema

---

|               |                                     |
|---------------|-------------------------------------|
| read_metadata | <i>Read and parse CSVW Metadata</i> |
|---------------|-------------------------------------|

---

**Description**

Reads in a json document as a list, transforming columns specifications into a dataframe.

**Usage**

```
read_metadata(filename)
```

**Arguments**

|          |                                     |
|----------|-------------------------------------|
| filename | a path for a json metadata document |
|----------|-------------------------------------|

**Value**

csvw metadata list

---

render\_uri\_templates    *Render URI templates*

---

**Description**

Interpolate variable bindings into a URI template.

**Usage**

```
render_uri_templates(templates, bindings = NULL, ...)
```

**Arguments**

|           |                                                               |
|-----------|---------------------------------------------------------------|
| templates | a character vector with URI templates                         |
| bindings  | a list of variable bindings to be interpolated into templates |
| ...       | further bindings specified as named function arguments        |

**Details**

This doesn't yet implement the whole of RFC 6570, just enough to make the tests pass

You can bind variables by passing a list to the explicit bindings argument, or variadically with ... by naming arguments according to the variable name you wish to bind.

**Value**

a character vector with the expanded URI

**Examples**

```
render_uri_templates("{+url}/resource?query=value", list(url="http://example.net"))
render_uri_templates("{+url}", url="http://example.net")
```

---

transform\_datetime\_format

*Transform date/time format string from Unicode TR35 to POSIX 1003.1*

---

### Description

As per the [csvw specification for date and time formats](#) we accept format strings using the [date field symbols defined in unicode TR35](#). These are converted to POSIX 1003.1 date format strings for use in [base::strptime\(\)](#) or [readr::parse\\_date\(\)/readr::parse\\_datetime\(\)](#).

### Usage

```
transform_datetime_format(format_string)
```

### Arguments

format\_string    a UAX35 date format string

### Value

a POSIX date format string

### Examples

```
## Not run:
fmt <- transform_datetime_format("dd.MM.yyyy")
strptime("01.01.2001", format=fmt)

## End(Not run)
```

---

type\_to\_datatype

*Map R types to csvw datatype*

---

### Description

Translate R types to [csvw datatypes](#). Acts as an inverse of `datatype_to_type` but doesn't provide a 1:1 correspondence.

### Usage

```
type_to_datatype(types)
```

### Arguments

types            a list of R types

**Value**

a list of csvw datatypes

---

|               |                                    |
|---------------|------------------------------------|
| validate_csvw | <i>Validate CSVW specification</i> |
|---------------|------------------------------------|

---

**Description**

Follows the **csvw table validation** procedure.

**Usage**

validate\_csvw(csvw)

**Arguments**

csvw                    a csvw metadata specification (a list)

**Value**

a validation report (list)

---

|                                |                                                                 |
|--------------------------------|-----------------------------------------------------------------|
| validate_referential_integrity | <i>Validate the referential integrity of a csvw table group</i> |
|--------------------------------|-----------------------------------------------------------------|

---

**Description**

Fails if foreign keys aren't found in the referenced tables

**Usage**

validate\_referential\_integrity(csvw)

**Arguments**

csvw                    the metadata annotation

**Value**

a list specifying any foreign key violations

# Index

- \* **datasets**
  - default\_dialect, [5](#)
- base::strptime(), [11](#)
- base\_uri, [2](#)
- coalesce\_truth, [3](#)
- create\_metadata, [3](#)
- csvw\_to\_list, [4](#)
- csvwr\_example, [4](#)
- datatype\_to\_type, [5](#)
- default\_dialect, [5](#)
- default\_schema, [6](#)
- derive\_metadata, [6](#)
- derive\_table\_schema, [7](#)
- extract\_table, [7](#)
- normalise\_metadata, [8](#)
- read\_csvw, [8](#)
- read\_csvw\_dataframe, [9](#)
- read\_metadata, [9](#)
- readr::cols, [5](#)
- readr::parse\_date(), [11](#)
- readr::parse\_datetime(), [11](#)
- readr::readr\_example(), [4](#)
- render\_uri\_templates, [10](#)
- transform\_datetime\_format, [11](#)
- type\_to\_datatype, [11](#)
- validate\_csvw, [12](#)
- validate\_referential\_integrity, [12](#)