Package 'curriculr'

May 13, 2026

Title Data-Driven CVs with 'Quarto' and 'Typst'

Version 0.3.0

Date 2026-05-04

Description Provides tools for producing data-driven curriculum vitae documents from structured data stored in an Excel workbook. The core workflow reads CV content from a workbook, converts it into 'Typst' layout blocks, and renders a polished PDF via the 'Quarto' publishing system. Includes functions for reading and cleaning CV data, building 'Typst' section headings and entries, rendering CV sections from data frames, and scaffolding new CV projects with a standard folder structure and template workbook. Designed to separate content from layout: CV data lives in the spreadsheet, rendering configuration lives in 'Quarto', and transformation logic lives in small, reusable R functions. See the 'Typst' typesetting system at <https://typst.app> and the 'Quarto' publishing system at <https://quarto.org>. Inspired by the 'vitae' package <https://CRAN.R-project.org/package=vitae> and the 'Awesome CV' LaTeX template <https://github.com/posquit0/Awesome-CV>.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

URL https://github.com/erwinlares/curriculr, https://erwinlares.github.io/curriculr/, https://doi.org/10.5281/zenodo.19930400

BugReports https://github.com/erwinlares/curriculr/issues

Depends R (>= 4.2.0)

Imports cli, fs, lifecycle, quarto, readr, openxlsx2

Suggests knitr, rmarkdown, spelling, withr, pdftools, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Erwin Lares [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-3284-828X>)

Maintainer Erwin Lares <erwin.lares@wisc.edu>

Repository CRAN

Date/Publication 2026-05-13 08:00:08 UTC

Contents

`add_section` *Add a new section to a curriculr workbook*

Description

Adds a new sheet to an existing curriculr-formatted Excel workbook and registers it in the `sections` control sheet. The new sheet is pre-populated with the standard column spine so the user can start entering data immediately without worrying about column names.

Usage

```
add_section(
  workbook,
  section,
  label = section,
  date_fun = "year_only",
  title_col = "title",
  org_col = "unit",
  detail_col = NA,
  where_col = "where",
  overwrite = FALSE
)
```

Arguments

workbook	A character string. Path to an existing curriculr Excel workbook.
section	A character string. Internal name of the new section. Must be a valid Excel sheet name (no more than 31 characters, no special characters). This name must match the sheet name exactly when referenced elsewhere in the workbook or in cap arguments to create_cv().
label	A character string. Display label shown as the section heading in the rendered CV. Defaults to section, which works when the section name is already human-readable. Supply a different value when the internal name and the display label differ, e.g. section = "invited_talks", label = "Invited Talks".
date_fun	A character string. Token controlling date formatting for this section. One of "date", "year", "month_year", "year_only", or "none". Defaults to "year_only". See resolve_date_fun() for token definitions.
title_col	A character string. Name of the column used as the primary entry label in the rendered CV. Defaults to "title".
org_col	A character string or NA. Name of the column used as the secondary organization or venue line. Defaults to "unit". Pass NA to omit the organization line for this section.
detail_col	A character string or NA. Name of the column used as the detail line. Defaults to NA (omitted). Pass a column name to include a detail line.
where_col	A character string or NA. Name of the column used as the location. Defaults to "where". Pass NA to omit location for this section.
overwrite	A logical. Whether to overwrite an existing sheet of the same name. Defaults to FALSE. When FALSE, an existing sheet causes an informative error. This is a destructive operation — use with care.

Details

add_section() performs the following steps:

1. Validates that workbook exists and that section is not already present (unless overwrite = TRUE).
2. Appends a new sheet named section with the standard column spine: title | unit | startMonth | startYear | endMonth | endYear | where | detail | include_in_resume.
3. Appends a new row to the sections sheet registering the new section with the supplied metadata. When overwrite = TRUE, any existing row for this section is replaced.
4. Writes the modified workbook back to workbook in place.

The workbook is modified in place. There is no undo. Consider keeping a backup copy before calling add_section() if the workbook contains data you cannot reconstruct.

Control sheets (profile, sections, theme, readme) cannot be used as section names.

Value

Invisibly returns workbook. Called primarily for its side effect of modifying the workbook on disk.

Examples

```
# Copy the template to a temp directory and add a section
tmp <- file.path(tempdir(), "cv-data.xlsx")
file.copy(
  system.file("extdata", "cv-data-template.xlsx", package = "curriculr"),
  tmp
)
add_section(tmp, section = "patents")


## Not run:
# Add a section with a display label that differs from the sheet name
add_section("cv-data.xlsx",
            section  = "invited_talks",
            label    = "Invited Talks",
            date_fun = "month_year")

# Add a section without an organization or location line
add_section("cv-data.xlsx",
            section   = "languages",
            label     = "Languages",
            date_fun  = "none",
            org_col   = NA,
            where_col = NA)

## End(Not run)
```

`create_cv` *Generate a CV document from a curriculr workbook*

Description

Called with no arguments, `create_cv()` runs in **scaffold mode**: it copies the template Excel workbook and placeholder profile image to the current working directory and prints instructions for the next step. No rendering takes place.

Usage

```
create_cv(
  data = NULL,
  photo = NULL,
  output_file = "CV.pdf",
  overwrite = FALSE,
  variant = "cv",
  use_icons = "fontawesome"
)
```

Arguments

data	A character string or NULL. Path to the Excel workbook. Defaults to NULL, which triggers scaffold mode.
photo	A character string or NULL. Path to the profile image. Defaults to NULL, which renders the CV without a profile photo using a single-column header layout. Supply a path to use a photo with the two-column header layout.
output_file	A character string. Name of the output PDF file. Defaults to "CV.pdf". Ignored in scaffold mode.
overwrite	A logical. Whether to overwrite existing files. Defaults to FALSE.
variant	A character string. Controls content scope. "cv" (the default) renders all rows from every section. "resume" renders only rows where include_in_resume is checked in the workbook.
use_icons	A character string. "fontawesome" (the default) renders contact fields in the CV header with Font Awesome icons via the Typst @preview/fontawesome package. "none" renders plain text.

Details

Called with data and photo arguments, create_cv() runs in **render mode**: it reads the workbook, generates CV.qmd, and renders it to PDF using Quarto's Typst engine. Both CV.qmd and CV.pdf are written to the same directory as the workbook.

Scaffold mode (no arguments):

1. Copies cv-data-template.xlsx to getwd().
2. Copies placeholder.png to getwd().
3. Prints instructions for editing the workbook and rendering the CV.

Render mode (data supplied):

1. Resolves and validates the workbook and photo paths.
2. Reads the workbook with read_cv_data(), applying variant filtering.
3. Resolves theme values from the workbook or built-in defaults.
4. Writes CV.qmd by injecting all resolved values into the package template via sentinel substitution.
5. Calls quarto::quarto_render() to produce the PDF.

When photo = NULL, the CV header renders as a single full-width column containing the name, contact line, address, and profile statement. When a photo path is supplied, the header uses a two-column layout with the photo on the left.

When variant = "resume", row-level filtering is controlled entirely by the include_in_resume column in each section sheet. Check the rows you want included in the resume and leave the rest unchecked.

Theme values (fonts, colors, page layout) are read from the theme sheet in the workbook. If the theme sheet is absent, built-in defaults are used. Individual keys missing from a partial theme sheet are filled from defaults.

Value

In scaffold mode, invisibly returns the path to the directory where files were copied. In render mode, invisibly returns the path to the rendered PDF.

Examples

```
# Scaffold mode — copy template files to a temp directory
withr::with_dir(tempdir(), create_cv())


## Not run:
# Render mode — requires cv-data.xlsx, Quarto, and Typst
create_cv(
  data  = "~/my_cv/cv-data.xlsx",
  photo = "~/my_cv/me.jpeg"
)

# Render mode — no photo, single-column header
create_cv(
  data = "~/my_cv/cv-data.xlsx"
)

# Render mode — resume variant
create_cv(
  data        = "~/my_cv/cv-data.xlsx",
  photo       = "~/my_cv/me.jpeg",
  variant     = "resume",
  output_file = "resume.pdf"
)

# Render mode — plain text contact line, custom output filename
create_cv(
  data        = "~/my_cv/cv-data.xlsx",
  photo       = "~/my_cv/me.jpeg",
  use_icons   = "none",
  output_file = "erwin-lares-cv.pdf"
)

## End(Not run)
```

---

cv_contact_line *Build the contact line for the CV header*

Description

Assembles a Typst-formatted contact line from a profile vector. When `use_icons = "fontawesome"`, known contact fields are rendered with their Font Awesome icon via the Typst `@preview/fontawesome` package. Fields with no icon equivalent fall back to plain text with a warning. When `use_icons = "none"`, all fields render as plain text.

Usage

```
cv_contact_line(profile, use_icons = "fontawesome")
```

Arguments

profile	A named character vector as returned by the profile element of read_cv_data().
use_icons	A character string. "fontawesome" (the default) renders contact fields with Font Awesome icons. "none" renders plain text.

Details

This function is called inside CV.qmd and is exported so that users building custom Quarto templates can call it directly.

Value

A character string of raw Typst markup for the contact line.

---

cv_render_section *Render a CV section from a data frame*

Description

Iterates over a CV data frame and writes each row as a Typst CV entry by calling .cv_entry() for each row and passing the result to base::cat().

Usage

```
cv_render_section(
  data,
  title_col,
  org_col = NULL,
  detail_col = NULL,
  date_fun = .cv_date_range,
  where_col = "where"
)
```

Arguments

data	A data frame containing CV entries. Typically one element of the list returned by read_cv_data(), e.g. cv$experience.
title_col	A character string. Name of the column to use as the entry title. Required.
org_col	A character string or NULL. Name of the column to use as the organization or secondary text. Defaults to NULL.
detail_col	A character string or NULL. Name of the column to use as additional detail. Defaults to NULL.

date_fun	A function or NULL. Called with each row to produce the date string. Defaults to .cv_date_range(). Pass NULL for sections without dates.
where_col	A character string or NULL. Name of the column to use as the location. Defaults to "where". Pass NULL to omit location.

Details

This function is intended to be called inside a Quarto document chunk with results = 'asis'. The cat() call writes raw Typst blocks directly into the document output stream. Nothing is returned — the function is called entirely for its side effect.

For sections that use dates, pass one of .cv_date_range() or .cv_year_range() as date_fun. For sections where dates are not relevant (skills, affiliations), pass date_fun = NULL.

Value

Invisibly returns NULL. Called for its side effect of writing Typst blocks to the Quarto document output stream.

Examples

```
# Load sample data and render the experience section
cv <- read_cv_data(
  system.file("extdata", "cv-data-template.xlsx", package = "curriculr")
)
cv_render_section(cv$experience,
                  title_col  = "title",
                  org_col    = "unit",
                  detail_col = "detail")


## Not run:
# The following examples are intended to be called inside a Quarto chunk
# with results = 'asis'. They require internal helpers and a loaded cv object.

# Year-only dates
cat(.cv_section("Education"))
cv_render_section(cv$education,
                  title_col  = "title",
                  org_col    = "institution",
                  detail_col = "detail",
                  date_fun   = .cv_year_range)

# No dates, no location
cat(.cv_section("Skills"))
cv_render_section(cv$skills,
                  title_col = "title",
                  org_col   = "unit",
                  date_fun  = NULL,
                  where_col = NULL)

# Custom inline date function
```

```
cat(.cv_section("Presentations"))
cv_render_section(cv$presentations,
                  title_col = "unit",
                  org_col   = "title",
                  date_fun  = function(row) {
                    trimws(paste(.cv_value(row, "startMonth"),
                                 .cv_value(row, "startYear")))
                  })

## End(Not run)
```

`cv_section` *Create a Typst CV section heading*

Description

Generates a raw Typst block for a CV section heading. The first letter of the section title is styled with the CV accent color. The heading is followed by a horizontal rule that fills the remaining line width.

Usage

```
cv_section(title)
```

Arguments

`title`	A character string. The section title to display, e.g. `"Education"` or `"Publications"`.

Details

This function is called inside `CV.qmd` to emit section headings. It is exported so that users building custom Quarto templates can call it directly without using `:::`.

Value

A character string of raw Typst markup.

read_cv_data *Read CV data from an Excel workbook*

Description

Reads all sheets from a curriculr-formatted Excel workbook and returns them as a named list of data frames. Each sheet becomes one list element, named after the sheet. The `profile` sheet is returned as a named character vector for convenient scalar access. The `theme` sheet is returned as a named character vector keyed by the `key` column. The `sections` sheet is returned as a data frame in the order the rows appear in the workbook — row order controls section render order and must not be sorted.

Usage

```
read_cv_data(path = "data/cv-data.xlsx", variant = "cv")
```

Arguments

`path`	A character string. Path to the Excel workbook. Defaults to `"data/cv-data.xlsx"`.
`variant`	A character string. Controls which rows are included from each section sheet. `"cv"` (the default) returns all rows. `"resume"` returns only rows where `include_in_resume` is `TRUE`. Sections that lack an `include_in_resume` column are included in full regardless of `variant`.

Details

Sheets containing a `startYear` column are sorted in descending order by `startYear` so that the most recent entries appear first. The `profile`, `theme`, and `sections` sheets are exempt from sorting.

The workbook must follow the curriculr schema. Every section sheet should contain a `title` column as the primary entry label. The `profile` sheet must contain `field` and `value` columns. The `sections` sheet must contain at minimum `section` and `label` columns. The `theme` sheet, if present, must contain `key` and `value` columns.

All cell values are read as character strings after import. Numeric columns such as `startYear` and `endYear` are coerced to character so that downstream rendering treats them uniformly. The `include_in_resume` column is read as a logical before coercion and used for row filtering when `variant = "resume"`.

Empty cells and cells containing the literal string `"NA"` are both converted to `NA`.

If the `theme` sheet is absent, `cv$theme` is `NULL` and `create_cv()` will fall back to built-in defaults.

Value

A named list with one element per sheet in the workbook. The `profile` element is a named character vector; the `theme` element is a named character vector (or `NULL` if the `theme` sheet is absent); all other elements are data frames. The `include_in_resume` column is dropped from returned data frames — it is used for filtering only and is not passed to the rendering pipeline. Access sections as `cv$education`, `cv$experience`, etc. Access profile fields as `cv$profile[["first_name"]]`.

Access theme values as `cv$theme[["accent_color"]]`. Access the sections control sheet as `cv$sections`.

Examples

```
# Read the sample data shipped with the package
cv <- read_cv_data(
  system.file("extdata", "cv-data-template.xlsx", package = "curriculr")
)
cv$education
cv$profile[["first_name"]]


## Not run:
# Read a user-supplied file
cv <- read_cv_data("~/my_cv/cv-data.xlsx")

# Resume variant
cv <- read_cv_data("~/my_cv/cv-data.xlsx", variant = "resume")

## End(Not run)
```

---

resolve_date_fun *Resolve a date_fun token to a function*

Description

Maps a string token from the `sections` sheet to the corresponding date formatting function used by `cv_render_section()`. This allows date formatting behaviour to be controlled from the Excel workbook rather than hardcoded in the Quarto template.

Usage

```
resolve_date_fun(token)
```

Arguments

`token`	A character string. One of `"date"`, `"year"`, `"month_year"`, `"year_only"`, or `"none"`.

Value

A function suitable for passing to the `date_fun` argument of `cv_render_section()`, or `NULL` when `token` is `"none"`.

`typst_escape` *Escape text for safe use in Typst markup*

Description

Converts an input value to a Typst-safe character string. Removes simple HTML line break tags, collapses repeated whitespace, escapes Typst special characters, and trims leading and trailing whitespace.

Usage

```
typst_escape(x)
```

Arguments

`x`	A value or vector to escape.

Details

CV content comes from Excel and may contain characters that Typst treats as markup: `#`, `$`, `%`, `&`, `~`, `_`, `^`, `{`, `}`, `[`, `]`, or `@`. The `@` character is included because Typst may interpret email addresses as references to labels.

Value

A character vector with Typst-sensitive characters escaped.

Index