

Package: apaTables (via r-universe)

October 9, 2026

Version 4.0.0

Title Create American Psychological Association (APA) Style Tables

Description A common task faced by researchers is the creation of APA style (i.e., American Psychological Association style) tables from statistical output. In R a large number of function calls are often needed to obtain all of the desired information for a single APA style table, and manually re-creating that table in a word processor is prone to transcription errors. This package builds complete APA style tables directly from analysis output. A single command saves a finished table as a 'Microsoft Word' (.docx) file, ready for a manuscript. Tables can also be placed in 'Quarto' documents (via the bundled 'apaquarto' extension), so a rendered 'Word' or PDF manuscript comes out with its tables already in place — numbered, titled, and with notes. The same tables can be shown on 'reveal.js' slides, and HTML output is also supported. Using this package minimizes transcription errors and reduces the number of commands needed by the user. The approach is described in Stanley and Spence (2018) <[doi:10.1177/2515245918773743](https://doi.org/10.1177/2515245918773743)>.

URL <https://github.com/dstanley4/apaTables>,
<https://dstanley4.github.io/apaTables/>

BugReports <https://github.com/dstanley4/apaTables/issues>

Depends R (>= 4.1.0)

Imports stats, utils, car, broom, dplyr, flextable (>= 0.9.0), knitr, officer

Suggests testthat, MBESS, afex, tidyr, rmarkdown, tinytex, pkgload, withr

VignetteBuilder knitr

Encoding UTF-8

License MIT + file LICENSE

Copyright See the file inst/COPYRIGHTS for the copyright and licence of the bundled third-party components.

LazyData true
Date 2026-08-09
Config/testthat/edition 3
Config/roxygen2/version 8.0.0
Config/pak/sysreqs
libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev make libharfbuzz-
dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libssl-dev
Repository https://dstanley4.r-universe.dev
Date/Publication 2026-08-10 13:25:21 UTC
RemoteUrl https://github.com/dstanley4/apatables
RemoteRef HEAD
RemoteSha 3e4714b44dbff17de567eede5bc34800b12c5855

Contents

apaTables-package	3
album	4
apa.1way.table	5
apa.2way.table	9
apa.afex.table	14
apa.aov.table	20
apa.cor.table	25
apa.d.table	30
apa.quarto.example	35
apa.quarto.new	37
apa.quarto.revealjs	39
apa.quarto.setup	43
apa.quarto.template	45
apa.reg.table	49
apa.save	54
apa_caption	56
apa_note_text	57
dating_wide	58
drink_attitude_wide	59
Eysenck	60
fidler_thompson	60
get.ci.partial.eta.squared	61
goggles	62
viagra	62
Index	64

apaTables-package	<i>apaTables: Create American Psychological Association (APA) Style Tables</i>
-------------------	--

Description

A common task faced by researchers is the creation of APA style (i.e., *American Psychological Association* style) tables from statistical output. In R a large number of function calls are often needed to obtain all of the desired information for a single APA style table, and manually re-creating that table in a word processor is prone to transcription errors. This package builds complete APA style tables directly from analysis output. A single command saves a finished table as a Microsoft Word (.docx) file, ready for a manuscript. Tables can also be placed in Quarto documents (via the bundled apaquarto extension), so a rendered Word or PDF manuscript comes out with its tables already in place – numbered, titled, and with notes. The same tables can be shown on reveal.js slides, and HTML output is also supported. Using this package minimizes transcription errors and reduces the number of commands needed by the user.

Bugs and feature requests can be reported at: <https://github.com/dstanley4/apaTables/issues>

Tutorial at: <https://dstanley4.github.io/apaTables/articles/apaTables.html>

Currently, the following tables can be created:

- Correlation tables - Correlation tables (with confidence intervals and descriptive statistics) are created from data frames using [apa.cor.table](#).
- Single "block" regression tables - Single "block" regression tables are created from a regression object using [apa.reg.table](#).
- Multiple "block" regression tables - Multiple "block" regression tables are created from regression objects using [apa.reg.table](#).
- ANOVA tables - An ANOVA F-table can be created via [apa.aov.table](#) from a regression object (i.e. lm output or aov output). Cell mean/standard deviation tables for 1- and 2-way designs are created from data frames using [apa.1way.table](#) and [apa.2way.table](#).
- ANOVA tables from afex package - An ANOVA F-table from `afex::aov_ez()` output (between, within, or mixed designs) can be created via [apa.afex.table](#).
- Standardized mean difference (i.e., *d*-value) tables (with confidence intervals and descriptive statistics) illustrating all possible paired comparisons using a single independent variable are created from data frames using [apa.d.table](#).

Author(s)

Maintainer: David J. Stanley <dstanley@uoguelph.ca> [copyright holder]

Authors:

- David J. Stanley <dstanley@uoguelph.ca> [copyright holder]

Other contributors:

- W. Joel Schneider (author of the apaquarto Quarto extension, bundled under CC0) [contributor]

- Brenton M. Wiernik (author of apa.csl, bundled under CC BY-SA 3.0) [contributor, copyright holder]

See Also

Useful links:

- <https://github.com/dstanley4/apaTables>
- <https://dstanley4.github.io/apaTables/>
- Report bugs at <https://github.com/dstanley4/apaTables/issues>

album

album data from textbook

Description

A data set from Field et al (2012)

Usage

```
data(album)
```

Format

A data frame with 200 rows and 4 variables:

adverts Amount spent of adverts, thousands of pounds

sales Album sales in thousands

airplay Number of times songs from album played on radio week prior to release

attract Attractiveness rating of band members

Source

<https://studysites.sagepub.com/dsur/study/>

References

Field, A., Miles, J., & Field, Z. (2012). Discovering statistics using R. Sage.

apa.lway.table

*Create an APA-style table of means for a one-way design***Description**

apa.lway.table() creates a table of means and standard deviations of a dependent variable for each level of a grouping variable, in APA style. Optionally, the 95% confidence interval for each mean is displayed. The table prints to the console. The most common next step is saving the table as a Word file to use in a manuscript:

Usage

```
apa.lway.table(
  iv,
  dv,
  data,
  filename = NA,
  table.number = 1,
  show.conf.interval = FALSE,
  landscape = NA
)
```

Arguments

iv	Name of the grouping (independent) variable column in data, given without quotation marks (for example, iv = dose, not iv = "dose").
dv	Name of the dependent variable column in data, given without quotation marks (for example, dv = libido, not dv = "libido").
data	A data frame.
filename	Optional file name. If supplied, the table is also saved as a Word (.docx) or HTML (.html) file, according to the file extension. Legacy .doc/.rtf names are saved as .docx.
table.number	Number for the "Table N" line of the output. Defaults to 1. Use 0 (or NA, or NULL) for a table with no number: none is then shown at the console or written to the Word file. When apa.save writes several tables to one file, a table left at the default is numbered by its position in that file, while a number given here is kept. In a Quarto document that uses the apaquarto extension this value is ignored: apaquarto numbers the tables itself, in the order they appear in the document.
show.conf.interval	If TRUE, display the 95% confidence interval for each mean. Defaults to FALSE.
landscape	Whether this table is placed on a rotated (landscape) page. The default, NA, means apaTables decides by measuring the table against the page. Pass TRUE to insist on a rotated page or FALSE to insist on an upright one. An explicit value is honoured wherever the destination can carry it out; in a Quarto PDF

that means the chunk needs a `tbl-` label (see [apa.quarto.pdf](#)). A table too wide for the page it is given is shrunk to fit rather than having a column cut off. Landscape is a property of a PAGE, so it applies to PDF and Word output and is ignored, without complaint, for HTML and slides. See **Rotated (landscape) pages** below.

Details

```
table1 <- apa.1way.table(iv = dose, dv = libido, data = viagra)
apa.save("Table1.docx", table1)
```

The file appears in your working directory – the folder R is currently pointing at. See the *Word files (apa.save)* section below. The table can also be placed in a Quarto document, so a rendered manuscript comes out with the table already in place – see the *Quarto documents* section below.

Value

A table object (class `apa_table`). Typing its name at the console displays the table. Pass it to [apa.save](#) to create a Word or HTML file. Or, if you are working in a Quarto document, use [apa.quarto.pdf](#) or [apa.quarto.docx](#) to make PDF and Word manuscripts, respectively.

Word files (apa.save)

[apa.save](#) writes the finished table to a Word file you can open, copy from, or send – the most common way to use an `apaTables` table:

```
apa.save("Table1.docx", table1)
```

The file appears in your working directory – the folder R is currently pointing at. Several tables can go into one file, each on its own page, numbered Table 1, Table 2, and so on in the order you list them:

```
apa.save("AllTables.docx", table1, table2)
```

The title above each table and the note below it are written by `apaTables` from the table itself; on this route there is no way to supply your own wording at this time. Tables are set in 12 point, and a table too wide for an upright page gets a rotated (landscape) page of its own rather than smaller type – so a wide table coming out sideways is expected, not an error.

Quarto documents

Quarto (<https://quarto.org>) is a free system for writing documents that combine text and R code. The `apaquarto` extension makes a Quarto document follow APA style: place this table in such a document and it appears as a fully formatted APA table – number, title, and note included – in Word, PDF, or HTML. The extension must be installed beside the document, and one command does it. `apa.quarto.new("my-paper")` creates a new manuscript folder with the extension already installed; for a folder you already have, `apa.quarto.setup()` installs the extension there instead. Tables go into the document inside a "code chunk": a small block of R code with a few settings lines at the top. "Rendering" the document turns it into the finished Word, PDF, or HTML file.

Every apaquarto document begins with a YAML header: the settings block between `---` lines at the very top of the document. apaquarto needs an author there – with no author, or an empty one, the render stops with a LaTeX error – and it reports a problem unless that author has an affiliation and is marked `corresponding: true`. This header is enough to render a PDF manuscript:

```
---
title: "My Paper"
author:
  - name: Jane Doe
    corresponding: true
    affiliations:
      - name: University of Guelph
format: apaquarto-pdf
---
```

For a Word manuscript, the same header with one line changed:

```
---
title: "My Paper"
author:
  - name: Jane Doe
    corresponding: true
    affiliations:
      - name: University of Guelph
format: apaquarto-docx
---
```

There are four format-named functions for this – [apa.quarto.pdf](#), [apa.quarto.docx](#), [apa.quarto.html](#) and [apa.quarto.revealjs](#) – one for each Quarto destination, and they behave identically: the document's own YAML `format:` line decides what is actually produced, so the name only states intent. This page's examples use `apa.quarto.pdf()` to match the `format: apaquarto-pdf` shown above; swap in `apa.quarto.docx()` or `apa.quarto.html()` together with the matching YAML line for Word or HTML.

In the document, the table lives inside a code chunk like this one. [apa.quarto.pdf](#) displays the table body with its title and note included automatically, so this chunk is complete – there is no title or note line to write, and the wording stays correct when the analysis changes. The `library(apaTables)` line is part of the chunk: without it the document cannot find `apa.1way.table()` and the render stops.

```
```{r}
#| label: tbl-means
#| message: false
#| warning: true
library(apaTables)
table1 <- apa.1way.table(iv = dose, dv = libido, data = viagra)
apa.quarto.pdf(table1)
```
```

That `#| label: tbl-` means line is not optional: without a label starting `tbl-`, Quarto builds no float for the chunk, so there is no number, no title, and no note, however you try to supply them.

Wide and tall tables. If this table turns out wider than the upright (portrait) PDF page, APA style places it on a rotated (landscape) page – and `apaTables` does that for you. It measures the table and asks `apaquarto` for a rotated page when one is needed, whether your tables sit in the text or are collected at the end of the manuscript. There is nothing to add to the chunk and nothing to add to the document's YAML header. A table wider still – wider than even a rotated page – is printed smaller instead, as small as it has to be so that no column is cut off. If that takes the type below 6 points, a warning in the rendered draft says so and points to [apa.save](#) for a full-size Word version. A table too tall for its page needs nothing either: it continues onto the next page by itself, with the column headings repeated at the top of each continuation and the note below the last piece. None of this applies to [apa.save](#) files: there, each table sits on a page of its own, always at full size.

To see this table (and every `apaTables` table type) in a complete working document, type `apa.quarto.example("apaTables-` in the R console. Two ready-to-render example documents appear in your working directory, with the `apaquarto` extension installed beside them – one per destination:

```
apaTables-examples-pdf.qmd
apaTables-examples-docx.qmd
```

Render the `-pdf` file for a PDF manuscript, the `-docx` file for a Word manuscript.

Rotated (landscape) pages

A table too wide to stand upright is placed on a rotated page automatically. You can overrule that per table with `landscape = TRUE` or `FALSE`, and per chunk in a Quarto document with `#| apa-landscape: true` or `false`, which wins over the argument.

The same table can be rotated in one output and upright in another, and both are right. `apa.save()` sets tables in 12 point on a page with half inch side margins; a Quarto manuscript sets them in 10 point inside a one inch text block. The table really is wider in the Word file, so it can need a rotated page there while fitting upright in the PDF. Each destination measures the table it is actually going to draw.

See Also

New to Quarto? [apa.quarto.example](#) writes a complete, ready-to-render example document with one `apaTables` table per page – the fastest way to see every table type working in an `apaquarto` document. In the R console, type:

```
apa.quarto.example("apaTables-examples.qmd")
```

Prefer a written guide? A step-by-step guide to placing these tables in Quarto documents comes installed with the package. To open it, type:

```
vignette("apaquarto-usage", package = "apaTables")
```

The package website has all of this documentation with rendered examples: <https://dstanley4.github.io/apaTables/>.

Other functions used with this table: [apa.save](#) to save it as a Word or HTML file; [apa.quarto.template](#) and [apa.quarto.pdf](#) to place it in a Quarto document.

Examples

```
library(apaTables)

# Means and standard deviations of libido for each dose
# (viagra data from Field et al. (2012), Discovering Statistics Using R)
table1 <- apa.1way.table(iv = dose, dv = libido,
                        data = viagra, table.number = 1)

table1

# Word documents -----
# Save the table as a .docx file (here, in a temporary
# folder; for real use give an ordinary name, such as
# apa.save("Table1.docx", table1), and the file appears in your
# working directory)
apa.save(file.path(tempdir(), "table1.docx"), table1)

# Quarto documents -----
# Print a ready-to-paste apaquarto code chunk:
apa.quarto.template(table1)

# Or write your own chunk; apa.quarto.pdf() displays the table body:
apa.quarto.pdf(table1)

# Clean up
unlink(file.path(tempdir(), "table1.docx"))
```

apa.2way.table

Create an APA-style table of means for a two-way design

Description

apa.2way.table() creates a table of cell means and standard deviations of a dependent variable for a two-way (factorial) design, in APA style. Optionally, the 95% confidence interval for each cell mean or the marginal means can be displayed. The table prints to the console. The most common next step is saving the table as a Word file to use in a manuscript:

Usage

```
apa.2way.table(
  iv1,
  iv2,
  dv,
  data,
  filename = NA,
  table.number = 1,
  show.conf.interval = FALSE,
  show.marginal.means = FALSE,
  landscape = NA
)
```

Arguments

| | |
|---------------------|---|
| iv1 | Name of the first grouping (independent) variable column in data, given without quotation marks (for example, iv1 = gender, not iv1 = "gender"). |
| iv2 | Name of the second grouping (independent) variable column in data, given without quotation marks. |
| dv | Name of the dependent variable column in data, given without quotation marks. |
| data | A data frame. |
| filename | Optional file name. If supplied, the table is also saved as a Word (.docx) or HTML (.html) file, according to the file extension. Legacy .doc/.rtf names are saved as .docx. |
| table.number | Number for the "Table N" line of the output. Defaults to 1. Use 0 (or NA, or NULL) for a table with no number: none is then shown at the console or written to the Word file. When apa.save writes several tables to one file, a table left at the default is numbered by its position in that file, while a number given here is kept. In a Quarto document that uses the <code>apaquarto</code> extension this value is ignored: <code>apaquarto</code> numbers the tables itself, in the order they appear in the document. |
| show.conf.interval | If TRUE, display the 95% confidence interval for each cell mean; marginal means are then not shown. Defaults to FALSE. |
| show.marginal.means | If TRUE, add the marginal means for each level of iv1 and iv2. Used only when <code>show.conf.interval = FALSE</code> . Defaults to FALSE. |
| landscape | Whether this table is placed on a rotated (landscape) page. The default, NA, means <code>apaTables</code> decides by measuring the table against the page. Pass TRUE to insist on a rotated page or FALSE to insist on an upright one. An explicit value is honoured wherever the destination can carry it out; in a Quarto PDF that means the chunk needs a <code>tbl-</code> label (see apa.quarto.pdf). A table too wide for the page it is given is shrunk to fit rather than having a column cut off. Landscape is a property of a PAGE, so it applies to PDF and Word output and is ignored, without complaint, for HTML and slides. See Rotated (landscape) pages below. |

Details

```
table1 <- apa.2way.table(iv1 = gender, iv2 = alcohol,
                        dv = attractiveness, data = goggles)
apa.save("Table1.docx", table1)
```

The file appears in your working directory – the folder R is currently pointing at. See the *Word files (apa.save)* section below. The table can also be placed in a Quarto document, so a rendered manuscript comes out with the table already in place – see the *Quarto documents* section below.

Value

A table object (class `apa_table`). Typing its name at the console displays the table. Pass it to [apa.save](#) to create a Word or HTML file. Or, if you are working in a Quarto document, use [apa.quarto.pdf](#) or [apa.quarto.docx](#) to make PDF and Word manuscripts, respectively.

Word files (apa.save)

`apa.save` writes the finished table to a Word file you can open, copy from, or send – the most common way to use an apaTables table:

```
apa.save("Table1.docx", table1)
```

The file appears in your working directory – the folder R is currently pointing at. Several tables can go into one file, each on its own page, numbered Table 1, Table 2, and so on in the order you list them:

```
apa.save("AllTables.docx", table1, table2)
```

The title above each table and the note below it are written by apaTables from the table itself; on this route there is no way to supply your own wording at this time. Tables are set in 12 point, and a table too wide for an upright page gets a rotated (landscape) page of its own rather than smaller type – so a wide table coming out sideways is expected, not an error.

Quarto documents

Quarto (<https://quarto.org>) is a free system for writing documents that combine text and R code. The apaquarto extension makes a Quarto document follow APA style: place this table in such a document and it appears as a fully formatted APA table – number, title, and note included – in Word, PDF, or HTML. The extension must be installed beside the document, and one command does it. `apa.quarto.new("my-paper")` creates a new manuscript folder with the extension already installed; for a folder you already have, `apa.quarto.setup()` installs the extension there instead. Tables go into the document inside a "code chunk": a small block of R code with a few settings lines at the top. "Rendering" the document turns it into the finished Word, PDF, or HTML file.

Every apaquarto document begins with a YAML header: the settings block between --- lines at the very top of the document. apaquarto needs an author there – with no author, or an empty one, the render stops with a LaTeX error – and it reports a problem unless that author has an affiliation and is marked corresponding: true. This header is enough to render a PDF manuscript:

```
---
title: "My Paper"
author:
  - name: Jane Doe
    corresponding: true
    affiliations:
      - name: University of Guelph
format: apaquarto-pdf
---
```

For a Word manuscript, the same header with one line changed:

```
---
title: "My Paper"
author:
```

```

- name: Jane Doe
  corresponding: true
  affiliations:
    - name: University of Guelph
format: apaquarto-docx
---
```

There are four format-named functions for this – [apa.quarto.pdf](#), [apa.quarto.docx](#), [apa.quarto.html](#) and [apa.quarto.revealjs](#) – one for each Quarto destination, and they behave identically: the document’s own YAML format: line decides what is actually produced, so the name only states intent. This page’s examples use `apa.quarto.pdf()` to match the format: `apaquarto-pdf` shown above; swap in `apa.quarto.docx()` or `apa.quarto.html()` together with the matching YAML line for Word or HTML.

In the document, the table lives inside a code chunk like this one. [apa.quarto.pdf](#) displays the table body with its title and note included automatically, so this chunk is complete – there is no title or note line to write, and the wording stays correct when the analysis changes. The `library(apaTables)` line is part of the chunk: without it the document cannot find `apa.2way.table()` and the render stops.

```

```{r}
#| label: tbl-means
#| message: false
#| warning: true
library(apaTables)
table1 <- apa.2way.table(iv1 = gender, iv2 = alcohol,
 dv = attractiveness, data = goggles)
apa.quarto.pdf(table1)
```
```

That `#| label: tbl-means` line is not optional: without a label starting `tbl-`, Quarto builds no float for the chunk, so there is no number, no title, and no note, however you try to supply them.

Wide and tall tables. If this table turns out wider than the upright (portrait) PDF page, APA style places it on a rotated (landscape) page – and `apaTables` does that for you. It measures the table and asks `apaquarto` for a rotated page when one is needed, whether your tables sit in the text or are collected at the end of the manuscript. There is nothing to add to the chunk and nothing to add to the document’s YAML header. A table wider still – wider than even a rotated page – is printed smaller instead, as small as it has to be so that no column is cut off. If that takes the type below 6 points, a warning in the rendered draft says so and points to [apa.save](#) for a full-size Word version. A table too tall for its page needs nothing either: it continues onto the next page by itself, with the column headings repeated at the top of each continuation and the note below the last piece. None of this applies to [apa.save](#) files: there, each table sits on a page of its own, always at full size.

To see this table (and every `apaTables` table type) in a complete working document, type `apa.quarto.example("apaTables-` in the R console. Two ready-to-render example documents appear in your working directory, with the `apaquarto` extension installed beside them – one per destination:

```

apaTables-examples-pdf.qmd
apaTables-examples-docx.qmd
```

Render the `-pdf` file for a PDF manuscript, the `-docx` file for a Word manuscript.

Rotated (landscape) pages

A table too wide to stand upright is placed on a rotated page automatically. You can overrule that per table with `landscape = TRUE` or `FALSE`, and per chunk in a Quarto document with `#| apa-landscape: true` or `false`, which wins over the argument.

The same table can be rotated in one output and upright in another, and both are right. `apa.save()` sets tables in 12 point on a page with half inch side margins; a Quarto manuscript sets them in 10 point inside a one inch text block. The table really is wider in the Word file, so it can need a rotated page there while fitting upright in the PDF. Each destination measures the table it is actually going to draw.

See Also

New to Quarto? [apa.quarto.example](#) writes a complete, ready-to-render example document with one `apaTables` table per page – the fastest way to see every table type working in an `apaquarto` document. In the R console, type:

```
apa.quarto.example("apaTables-examples.qmd")
```

Prefer a written guide? A step-by-step guide to placing these tables in Quarto documents comes installed with the package. To open it, type:

```
vignette("apaquarto-usage", package = "apaTables")
```

The package website has all of this documentation with rendered examples: <https://dstanley4.github.io/apaTables/>.

Other functions used with this table: [apa.save](#) to save it as a Word or HTML file; [apa.quarto.template](#) and [apa.quarto.pdf](#) to place it in a Quarto document.

Examples

```
library(apaTables)

# Cell means for a 2-way design from Fidler & Thompson (2001)
table2 <- apa.2way.table(iv1 = a, iv2 = b, dv = dv,
                        data = fidler_thompson,
                        table.number = 2)

table2

# Cell means for the goggles data
# (from Field et al. (2012), Discovering Statistics Using R)
table3 <- apa.2way.table(iv1 = gender, iv2 = alcohol, dv = attractiveness,
                        data = goggles, table.number = 3)

table3

# Word documents -----
# apa.save() can write several tables into one Word document
# (here, in a temporary
# folder; for real use give an ordinary name, such as
# apa.save("Table1.docx", table1), and the file appears in your
```

```
# working directory)
apa.save(file.path(tempdir(), "my_tables.docx"), table2, table3)

# Quarto documents -----
# Print a ready-to-paste apaquarto code chunk:
apa.quarto.template(table2)

# Or write your own chunk; apa.quarto.pdf() displays the table body:
apa.quarto.pdf(table2)

# Clean up
unlink(file.path(tempdir(), "my_tables.docx"))
```

| | |
|----------------|--|
| apa.afex.table | Create an APA-style ANOVA table from afex output |
|----------------|--|

Description

`apa.afex.table()` creates an ANOVA table in APA style from the result of `afex::aov_ez()`. It handles between-participant, within-participant, and mixed designs. For each effect the table reports the numerator and denominator degrees of freedom, mean square error, F, p, and generalized eta-squared; for within-participant effects with a sphericity correction, the epsilon multiplier is also reported. The table prints to the console. The most common next step is saving the table as a Word file to use in a manuscript:

Usage

```
apa.afex.table(
  afex.output,
  correction = "GG",
  table.title = "",
  filename = NA,
  table.number = 1,
  landscape = NA
)
```

Arguments

| | |
|--------------------------|--|
| <code>afex.output</code> | The result returned by <code>afex::aov_ez()</code> (an <code>afex_aov</code> object). |
| <code>correction</code> | Sphericity correction for within-participant effects: "GG" (Greenhouse-Geisser, the default), "HF" (Huynh-Feldt), or "none". Degrees of freedom and p-values in the table incorporate the correction. |
| <code>table.title</code> | Optional string with the table title. When left empty (the default) the title is "ANOVA Results". The title given here is used everywhere the table appears: at the console, in a Word or HTML file, and in a Quarto document. |
| <code>filename</code> | Optional file name. If supplied, the table is also saved as a Word (.docx) or HTML (.html) file, according to the file extension. Legacy .doc/.rtf names are saved as .docx. |

| | |
|--------------|---|
| table.number | Number for the "Table N" line of the output. Defaults to 1. Use 0 (or NA, or NULL) for a table with no number: none is then shown at the console or written to the Word file. When apa.save writes several tables to one file, a table left at the default is numbered by its position in that file, while a number given here is kept. In a Quarto document that uses the apaquarto extension this value is ignored: apaquarto numbers the tables itself, in the order they appear in the document. |
| landscape | Whether this table is placed on a rotated (landscape) page. The default, NA, means apaTables decides by measuring the table against the page. Pass TRUE to insist on a rotated page or FALSE to insist on an upright one. An explicit value is honoured wherever the destination can carry it out; in a Quarto PDF that means the chunk needs a tbl- label (see apa.quarto.pdf). A table too wide for the page it is given is shrunk to fit rather than having a column cut off. Landscape is a property of a PAGE, so it applies to PDF and Word output and is ignored, without complaint, for HTML and slides. See Rotated (landscape) pages below. |

Details

```
goggles_results <- afex::aov_ez(id = "participant",
  dv = "attractiveness", data = goggles,
  between = c("gender", "alcohol"))
table1 <- apa.afex.table(goggles_results)
apa.save("Table1.docx", table1)
```

The file appears in your working directory – the folder R is currently pointing at. See the *Word files (apa.save)* section below. The table can also be placed in a Quarto document, so a rendered manuscript comes out with the table already in place – see the *Quarto documents* section below.

Value

A table object (class `apa_table`). Typing its name at the console displays the table. Pass it to [apa.save](#) to create a Word or HTML file. Or, if you are working in a Quarto document, use [apa.quarto.pdf](#) or [apa.quarto.docx](#) to make PDF and Word manuscripts, respectively.

Word files (apa.save)

[apa.save](#) writes the finished table to a Word file you can open, copy from, or send – the most common way to use an apaTables table:

```
apa.save("Table1.docx", table1)
```

The file appears in your working directory – the folder R is currently pointing at. Several tables can go into one file, each on its own page, numbered Table 1, Table 2, and so on in the order you list them:

```
apa.save("AllTables.docx", table1, table2)
```

The title above each table and the note below it are written by apaTables from the table itself; on this route there is no way to supply your own wording at this time. Tables are set in 12 point, and a table too wide for an upright page gets a rotated (landscape) page of its own rather than smaller type – so a wide table coming out sideways is expected, not an error.

Quarto documents

Quarto (<https://quarto.org>) is a free system for writing documents that combine text and R code. The apaquarto extension makes a Quarto document follow APA style: place this table in such a document and it appears as a fully formatted APA table – number, title, and note included – in Word, PDF, or HTML. The extension must be installed beside the document, and one command does it. `apa.quarto.new("my-paper")` creates a new manuscript folder with the extension already installed; for a folder you already have, `apa.quarto.setup()` installs the extension there instead. Tables go into the document inside a "code chunk": a small block of R code with a few settings lines at the top. "Rendering" the document turns it into the finished Word, PDF, or HTML file.

Every apaquarto document begins with a YAML header: the settings block between `---` lines at the very top of the document. apaquarto needs an author there – with no author, or an empty one, the render stops with a LaTeX error – and it reports a problem unless that author has an affiliation and is marked `corresponding: true`. This header is enough to render a PDF manuscript:

```
---
title: "My Paper"
author:
  - name: Jane Doe
    corresponding: true
    affiliations:
      - name: University of Guelph
format: apaquarto-pdf
---
```

For a Word manuscript, the same header with one line changed:

```
---
title: "My Paper"
author:
  - name: Jane Doe
    corresponding: true
    affiliations:
      - name: University of Guelph
format: apaquarto-docx
---
```

Two other packages must be installed first. This table is built from the output of `afex::aov_ez()`, so the afex package has to be installed; and repeated-measures data usually arrive with one row per participant, whereas `aov_ez()` needs one row per observation, so `tidyr::pivot_longer()` is normally used to reshape them. Neither package is installed with apaTables. In the R console, type:

```
install.packages(c("afex", "tidyr"))
```

Without them the render stops at the chunk below with there is no package called 'afex'.

There are four format-named functions for this – [apa.quarto.pdf](#), [apa.quarto.docx](#), [apa.quarto.html](#) and [apa.quarto.revealjs](#) – one for each Quarto destination, and they behave identically: the document's own YAML format: line decides what is actually produced, so the name only states intent. This page's examples use `apa.quarto.pdf()` to match the format: `apaquarto-pdf` shown above; swap in `apa.quarto.docx()` or `apa.quarto.html()` together with the matching YAML line for Word or HTML.

In the document, the table lives inside a code chunk like this one. [apa.quarto.pdf](#) displays the table body with its title and note included automatically, so this chunk is complete – there is no title or note line to write, and the wording stays correct when the analysis changes. The `library(apaTables)` line is part of the chunk: without it the document cannot find `apa.afex.table()` and the render stops.

```
```{r}
#| label: tbl-anova
#| message: false
#| warning: true
library(apaTables)
goggles_results <- afex::aov_ez(id = "participant",
 dv = "attractiveness", data = goggles,
 between = c("gender", "alcohol"))
table1 <- apa.afex.table(goggles_results)
apa.quarto.pdf(table1)
```
```

That `#| label: tbl-anova` line is not optional: without a label starting `tbl-`, Quarto builds no float for the chunk, so there is no number, no title, and no note, however you try to supply them.

Wide and tall tables. If this table turns out wider than the upright (portrait) PDF page, APA style places it on a rotated (landscape) page – and `apaTables` does that for you. It measures the table and asks `apaquarto` for a rotated page when one is needed, whether your tables sit in the text or are collected at the end of the manuscript. There is nothing to add to the chunk and nothing to add to the document's YAML header. A table wider still – wider than even a rotated page – is printed smaller instead, as small as it has to be so that no column is cut off. If that takes the type below 6 points, a warning in the rendered draft says so and points to [apa.save](#) for a full-size Word version. A table too tall for its page needs nothing either: it continues onto the next page by itself, with the column headings repeated at the top of each continuation and the note below the last piece. None of this applies to [apa.save](#) files: there, each table sits on a page of its own, always at full size.

To see this table (and every `apaTables` table type) in a complete working document, type `apa.quarto.example("apaTables-` in the R console. Two ready-to-render example documents appear in your working directory, with the `apaquarto` extension installed beside them – one per destination:

```
apaTables-examples-pdf.qmd
apaTables-examples-docx.qmd
```

Render the `-pdf` file for a PDF manuscript, the `-docx` file for a Word manuscript.

Rotated (landscape) pages

A table too wide to stand upright is placed on a rotated page automatically. You can overrule that per table with `landscape = TRUE` or `FALSE`, and per chunk in a Quarto document with `#|apa-landscape: true` or `false`, which wins over the argument.

The same table can be rotated in one output and upright in another, and both are right. `apa.save()` sets tables in 12 point on a page with half inch side margins; a Quarto manuscript sets them in 10 point inside a one inch text block. The table really is wider in the Word file, so it can need a rotated page there while fitting upright in the PDF. Each destination measures the table it is actually going to draw.

See Also

New to Quarto? [apa.quarto.example](#) writes a complete, ready-to-render example document with one `apaTables` table per page – the fastest way to see every table type working in an `apaquarto` document. In the R console, type:

```
apa.quarto.example("apaTables-examples.qmd")
```

Prefer a written guide? A step-by-step guide to placing these tables in Quarto documents comes installed with the package. To open it, type:

```
vignette("apaquarto-usage", package = "apaTables")
```

The package website has all of this documentation with rendered examples: <https://dstanley4.github.io/apaTables/>.

Other functions used with this table: [apa.save](#) to save it as a Word or HTML file; [apa.quarto.template](#) and [apa.quarto.pdf](#) to place it in a Quarto document.

Examples

```
if (requireNamespace("afex", quietly = TRUE)) {

#
# ** Example 1: Between Participant Predictors
#

goggles <- apaTables::goggles

goggles_results <- afex::aov_ez(id = "participant", dv = "attractiveness",
                              data = goggles,
                              between = c("gender", "alcohol"))

# Make APA table (all three tables are saved together at the end)
goggles_table <- apa.afex.table(afex.output = goggles_results)
goggles_table

# Quarto documents -----
# Print a ready-to-paste apaquarto code chunk:
apa.quarto.template(goggles_table)
```

```

# Or write your own chunk; apa.quarto.pdf() displays the table body:
apa.quarto.pdf(goggles_table)
}

if (requireNamespace("afex", quietly = TRUE) &&
    requireNamespace("tidyr", quietly = TRUE)) {

#
# ** Example 2: Within Participant Predictors
#

drink_attitude_wide <- apaTables::drink_attitude_wide

# Convert data from wide format to long format where one row represents
# one OBSERVATION. Wide format column names MUST represent levels of each
# variable separated by an underscore. See vignette for further details.

drink_attitude_long <- tidyr::pivot_longer(drink_attitude_wide,
                                           cols = beer_positive:water_neutral,
                                           names_to = c("drink", "imagery"),
                                           names_sep = "_",
                                           values_to = "attitude")

drink_attitude_long$drink <- as.factor(drink_attitude_long$drink)
drink_attitude_long$imagery <- as.factor(drink_attitude_long$imagery)

drink_attitude_results <- afex::aov_ez(id = "participant", dv = "attitude",
                                       data = drink_attitude_long,
                                       within = c("drink", "imagery"))

# Make APA table (saved together with the others below)
drink_table <- apa.afex.table(afex.output = drink_attitude_results)

#
# ** Example 3: Between and Within Participant Predictors
#

dating_wide <- apaTables::dating_wide

# Convert data from wide format to long format where one row represents
# one OBSERVATION. Wide format column names MUST represent levels of each
# variable separated by an underscore. See vignette for further details.

dating_long <- tidyr::pivot_longer(dating_wide,
                                   cols = attractive_high:ugly_none,
                                   names_to = c("looks", "personality"),
                                   names_sep = "_",
                                   values_to = "date_rating")

dating_long$looks <- as.factor(dating_long$looks)
dating_long$personality <- as.factor(dating_long$personality)

```

```

dating_results <- afex::aov_ez(id = "participant", dv = "date_rating",
                              data = dating_long,
                              between = "gender",
                              within = c("looks", "personality"))

# Make APA table (saved together with the others below)
dating_table <- apa.afex.table(afex.output = dating_results)

# -----
# Save the tables to a Microsoft Word (.docx) file
# -----
# apa.save() can write several tables into one Word document. This
# example writes to a temporary folder (tempdir()) so it leaves no
# files on your computer; for real use give an ordinary name such as
# "Tables_afex.docx".
apa.save(filename = file.path(tempdir(), "tables_afex.docx"),
         goggles_table,
         drink_table,
         dating_table)

# delete demo file
unlink(file.path(tempdir(), "tables_afex.docx"))
}

```

apa.aov.table

Create an APA-style fixed-effects ANOVA table

Description

apa.aov.table() creates a fixed-effects ANOVA table in APA style from an [lm](#) result. For each predictor the table reports the sum of squares, degrees of freedom, mean square, F, p, and partial eta-squared with its confidence interval. The table prints to the console. The most common next step is saving the table as a Word file to use in a manuscript:

Usage

```

apa.aov.table(
  lm_output,
  filename = NA,
  table.number = 1,
  conf.level = 0.9,
  type = 3,
  landscape = NA
)

```

Arguments

| | |
|--------------|---|
| lm_output | A result from lm in which all predictors are factors (categorical variables). |
| filename | Optional file name. If supplied, the table is also saved as a Word (.docx) or HTML (.html) file, according to the file extension. Legacy .doc/.rtf names are saved as .docx. |
| table.number | Number for the "Table N" line of the output. Defaults to 1. Use 0 (or NA, or NULL) for a table with no number: none is then shown at the console or written to the Word file. When apa.save writes several tables to one file, a table left at the default is numbered by its position in that file, while a number given here is kept. In a Quarto document that uses the apaquarto extension this value is ignored: apaquarto numbers the tables itself, in the order they appear in the document. |
| conf.level | Confidence level (.90, the default, or .95) for the interval around partial eta-squared. Using .90 keeps conclusions based on this interval consistent with the p-value. |
| type | Sum-of-squares type: 2 (Type II) or 3 (Type III, the default). |
| landscape | Whether this table is placed on a rotated (landscape) page. The default, NA, means apaTables decides by measuring the table against the page. Pass TRUE to insist on a rotated page or FALSE to insist on an upright one. An explicit value is honoured wherever the destination can carry it out; in a Quarto PDF that means the chunk needs a tbl- label (see apa.quarto.pdf). A table too wide for the page it is given is shrunk to fit rather than having a column cut off. Landscape is a property of a PAGE, so it applies to PDF and Word output and is ignored, without complaint, for HTML and slides. See Rotated (landscape) pages below. |

Details

```
options(contrasts = c("contr.sum", "contr.poly"))
lm_output <- lm(libido ~ dose, data = viagra)
table1 <- apa.aov.table(lm_output)
apa.save("Table1.docx", table1)
```

The file appears in your working directory – the folder R is currently pointing at. See the *Word files (apa.save)* section below. The table can also be placed in a Quarto document, so a rendered manuscript comes out with the table already in place – see the *Quarto documents* section below.

Confidence intervals for partial eta-squared are calculated with the noncentral approach described by Smithson (2001) and Fidler and Thompson (2001). For Type III sums of squares to match programs such as SPSS, factor contrasts must sum to zero; the examples set `options(contrasts = c("contr.sum", "contr.poly"))` for this reason.

Value

A table object (class `apa_table`). Typing its name at the console displays the table. Pass it to [apa.save](#) to create a Word or HTML file. Or, if you are working in a Quarto document, use [apa.quarto.pdf](#) or [apa.quarto.docx](#) to make PDF and Word manuscripts, respectively.

Word files (apa.save)

`apa.save` writes the finished table to a Word file you can open, copy from, or send – the most common way to use an `apaTables` table:

```
apa.save("Table1.docx", table1)
```

The file appears in your working directory – the folder R is currently pointing at. Several tables can go into one file, each on its own page, numbered Table 1, Table 2, and so on in the order you list them:

```
apa.save("AllTables.docx", table1, table2)
```

The title above each table and the note below it are written by `apaTables` from the table itself; on this route there is no way to supply your own wording at this time. Tables are set in 12 point, and a table too wide for an upright page gets a rotated (landscape) page of its own rather than smaller type – so a wide table coming out sideways is expected, not an error.

Quarto documents

Quarto (<https://quarto.org>) is a free system for writing documents that combine text and R code. The `apaquarto` extension makes a Quarto document follow APA style: place this table in such a document and it appears as a fully formatted APA table – number, title, and note included – in Word, PDF, or HTML. The extension must be installed beside the document, and one command does it. `apa.quarto.new("my-paper")` creates a new manuscript folder with the extension already installed; for a folder you already have, `apa.quarto.setup()` installs the extension there instead. Tables go into the document inside a "code chunk": a small block of R code with a few settings lines at the top. "Rendering" the document turns it into the finished Word, PDF, or HTML file.

Every `apaquarto` document begins with a YAML header: the settings block between `---` lines at the very top of the document. `apaquarto` needs an author there – with no author, or an empty one, the render stops with a LaTeX error – and it reports a problem unless that author has an affiliation and is marked corresponding: `true`. This header is enough to render a PDF manuscript:

```
---
title: "My Paper"
author:
  - name: Jane Doe
    corresponding: true
    affiliations:
      - name: University of Guelph
format: apaquarto-pdf
---
```

For a Word manuscript, the same header with one line changed:

```
---
title: "My Paper"
author:
```

```

- name: Jane Doe
  corresponding: true
  affiliations:
    - name: University of Guelph
format: apaquarto-docx
---
```

There are four format-named functions for this – [apa.quarto.pdf](#), [apa.quarto.docx](#), [apa.quarto.html](#) and [apa.quarto.revealjs](#) – one for each Quarto destination, and they behave identically: the document’s own YAML format: line decides what is actually produced, so the name only states intent. This page’s examples use `apa.quarto.pdf()` to match the format: `apaquarto-pdf` shown above; swap in `apa.quarto.docx()` or `apa.quarto.html()` together with the matching YAML line for Word or HTML.

In the document, the table lives inside a code chunk like this one. [apa.quarto.pdf](#) displays the table body with its title and note included automatically, so this chunk is complete – there is no title or note line to write, and the wording stays correct when the analysis changes. Two lines of the chunk are easy to leave out and both matter. The first is `library(apaTables)`: without it the document cannot find `apa.aov.table()` and the render stops. The second is the `options(contrasts = ...)` line: without it the Type III sums of squares are computed from R’s default treatment contrasts, and the table comes out with different – wrong – F values and p-values, with no warning anywhere. Set the contrasts before fitting the model:

```

```{r}
#| label: tbl-anova
#| message: false
#| warning: true
library(apaTables)
options(contrasts = c("contr.sum", "contr.poly"))
lm_output <- lm(libido ~ dose, data = viagra)
table1 <- apa.aov.table(lm_output)
apa.quarto.pdf(table1)
```
```

That `#| label: tbl-anova` line is not optional: without a label starting `tbl-`, Quarto builds no float for the chunk, so there is no number, no title, and no note, however you try to supply them.

Wide and tall tables. If this table turns out wider than the upright (portrait) PDF page, APA style places it on a rotated (landscape) page – and `apaTables` does that for you. It measures the table and asks `apaquarto` for a rotated page when one is needed, whether your tables sit in the text or are collected at the end of the manuscript. There is nothing to add to the chunk and nothing to add to the document’s YAML header. A table wider still – wider than even a rotated page – is printed smaller instead, as small as it has to be so that no column is cut off. If that takes the type below 6 points, a warning in the rendered draft says so and points to [apa.save](#) for a full-size Word version. A table too tall for its page needs nothing either: it continues onto the next page by itself, with the column headings repeated at the top of each continuation and the note below the last piece. None of this applies to [apa.save](#) files: there, each table sits on a page of its own, always at full size.

To see this table (and every `apaTables` table type) in a complete working document, type `apa.quarto.example("apaTables-` in the R console. Two ready-to-render example documents appear in your working directory, with the `apaquarto` extension installed beside them – one per destination:

```
apaTables-examples-pdf.qmd
apaTables-examples-docx.qmd
```

Render the `-pdf` file for a PDF manuscript, the `-docx` file for a Word manuscript.

Rotated (landscape) pages

A table too wide to stand upright is placed on a rotated page automatically. You can overrule that per table with `landscape = TRUE` or `FALSE`, and per chunk in a Quarto document with `#| apa-landscape: true` or `false`, which wins over the argument.

The same table can be rotated in one output and upright in another, and both are right. `apa.save()` sets tables in 12 point on a page with half inch side margins; a Quarto manuscript sets them in 10 point inside a one inch text block. The table really is wider in the Word file, so it can need a rotated page there while fitting upright in the PDF. Each destination measures the table it is actually going to draw.

References

Smithson, M. (2001). Correct confidence intervals for various regression effect sizes and parameters: The importance of noncentral distributions in computing intervals. *Educational and Psychological Measurement*, 61(4), 605-632.

Fidler, F., & Thompson, B. (2001). Computing correct confidence intervals for ANOVA fixed- and random-effects effect sizes. *Educational and Psychological Measurement*, 61(4), 575-604.

See Also

New to Quarto? [apa.quarto.example](#) writes a complete, ready-to-render example document with one `apaTables` table per page – the fastest way to see every table type working in an `apaquarto` document. In the R console, type:

```
apa.quarto.example("apaTables-examples.qmd")
```

Prefer a written guide? A step-by-step guide to placing these tables in Quarto documents comes installed with the package. To open it, type:

```
vignette("apaquarto-usage", package = "apaTables")
```

The package website has all of this documentation with rendered examples: <https://dstanley4.github.io/apaTables/>.

Other functions used with this table: [apa.save](#) to save it as a Word or HTML file; [apa.quarto.template](#) and [apa.quarto.pdf](#) to place it in a Quarto document.

Examples

```
library(apaTables)

op <- options() # save the current settings, restored at the end
```

```

# 1-way ANOVA of libido by dose
# (viagra data from Field et al. (2012), Discovering Statistics Using R)
# Contrasts must sum to zero for correct Type III sums of squares:
options(contrasts = c("contr.sum", "contr.poly"))
lm_output <- lm(libido ~ dose, data = viagra)
table1 <- apa.aov.table(lm_output, table.number = 4)
table1

# 2-way ANOVA from Fidler & Thompson (2001)
options(contrasts = c("contr.sum", "contr.poly"))
lm_output <- lm(dv ~ a*b, data = fidler_thompson)
table2 <- apa.aov.table(lm_output, table.number = 5)

# 2-way ANOVA for the goggles data
# (from Field et al. (2012), Discovering Statistics Using R)
options(contrasts = c("contr.sum", "contr.poly"))
lm_output <- lm(attractiveness ~ gender*alcohol, data = goggles)
table3 <- apa.aov.table(lm_output, table.number = 6)

# Word documents -----
# apa.save() can write several tables into one Word document
# (here, in a temporary
# folder; for real use give an ordinary name, such as
# apa.save("Table1.docx", table1), and the file appears in your
# working directory)
apa.save(file.path(tempdir(), "my_tables.docx"), table1, table2, table3)

# Quarto documents -----
# Print a ready-to-paste apaquarto code chunk:
apa.quarto.template(table1)

# Or write your own chunk; apa.quarto.pdf() displays the table body:
apa.quarto.pdf(table1)

# Clean up
unlink(file.path(tempdir(), "my_tables.docx"))
options(op)

```

apa.cor.table

Create an APA-style correlation table

Description

apa.cor.table() creates a table of means, standard deviations, and Pearson correlations in APA style. Each correlation is reported with its 95% confidence interval and, optionally, significance stars and a p-value. Typing apa.cor.table(attitude) prints the finished table at the console. The most common next step is saving the table as a Word file to use in a manuscript:

Usage

```
apa.cor.table(
  data,
  filename = NA,
  table.number = 1,
  show.conf.interval = TRUE,
  show.sig.stars = TRUE,
  show.pvalue = TRUE,
  landscape = NA
)
```

Arguments

| | |
|--------------------|--|
| data | A data frame. Non-numeric columns are dropped before the table is constructed. |
| filename | Optional file name. If supplied, the table is also saved as a Word (.docx) or HTML (.html) file, according to the file extension. Legacy .doc/.rtf names are saved as .docx. |
| table.number | Number for the "Table N" line of the output. Defaults to 1. Use 0 (or NA, or NULL) for a table with no number: none is then shown at the console or written to the Word file. When apa.save writes several tables to one file, a table left at the default is numbered by its position in that file, while a number given here is kept. In a Quarto document that uses the apaquarto extension this value is ignored: apaquarto numbers the tables itself, in the order they appear in the document. |
| show.conf.interval | Deprecated and ignored; confidence intervals are always displayed. This argument will be removed in a later version. |
| show.sig.stars | If TRUE (the default), correlations are marked with stars for significance (* p < .05, ** p < .01). |
| show.pvalue | If TRUE (the default), the p-value for each correlation is displayed below its confidence interval. |
| landscape | Whether this table is placed on a rotated (landscape) page. The default, NA, means apaTables decides by measuring the table against the page. Pass TRUE to insist on a rotated page or FALSE to insist on an upright one. An explicit value is honoured wherever the destination can carry it out; in a Quarto PDF that means the chunk needs a <code>tbl-</code> label (see apa.quarto.pdf). A table too wide for the page it is given is shrunk to fit rather than having a column cut off. Landscape is a property of a PAGE, so it applies to PDF and Word output and is ignored, without complaint, for HTML and slides. See Rotated (landscape) pages below. |

Details

```
table1 <- apa.cor.table(attitude)
apa.save("Table1.docx", table1)
```

The file appears in your working directory – the folder R is currently pointing at. See the *Word files (apa.save)* section below. The table can also be placed in a Quarto document, so a rendered manuscript comes out with the table already in place – see the *Quarto documents* section below.

For each variable the table reports the number of cases (*N*), mean (*M*), and standard deviation (*SD*), computed after removing missing values for that variable. Correlations and their confidence intervals are computed pairwise with `cor.test`, so each correlation uses all cases that are complete for that pair of variables.

Value

A table object (class `apa_table`). Typing its name at the console displays the table. Pass it to `apa.save` to create a Word or HTML file. Or, if you are working in a Quarto document, use `apa.quarto.pdf` or `apa.quarto.docx` to make PDF and Word manuscripts, respectively.

Word files (apa.save)

`apa.save` writes the finished table to a Word file you can open, copy from, or send – the most common way to use an `apaTables` table:

```
table1 <- apa.cor.table(attitude)
apa.save("Table1.docx", table1)
```

The file appears in your working directory – the folder R is currently pointing at. Several tables can go into one file, each on its own page, numbered Table 1, Table 2, and so on in the order you list them:

```
apa.save("AllTables.docx", table1, table2)
```

The title above each table and the note below it are written by `apaTables` from the table itself; on this route there is no way to supply your own wording at this time. Tables are set in 12 point, and a table too wide for an upright page gets a rotated (landscape) page of its own rather than smaller type – so a wide correlation table coming out sideways is expected, not an error.

Quarto documents

Quarto (<https://quarto.org>) is a free system for writing documents that combine text and R code. The `apaquarto` extension makes a Quarto document follow APA style: place this table in such a document and it appears as a fully formatted APA table – number, title, and note included – in Word, PDF, or HTML. The extension must be installed beside the document, and one command does it. `apa.quarto.new("my-paper")` creates a new manuscript folder with the extension already installed; for a folder you already have, `apa.quarto.setup()` installs the extension there instead. Tables go into the document inside a "code chunk": a small block of R code with a few settings lines at the top. "Rendering" the document turns it into the finished Word, PDF, or HTML file.

Every `apaquarto` document begins with a YAML header: the settings block between `---` lines at the very top of the document. `apaquarto` needs an author there – with no author, or an empty one, the render stops with a LaTeX error – and it reports a problem unless that author has an affiliation and is marked `corresponding: true`. This header is enough to render a PDF manuscript:

```

---
title: "My Paper"
author:
  - name: Jane Doe
    corresponding: true
    affiliations:
      - name: University of Guelph
format: apaquarto-pdf
---

```

For a Word manuscript, the same header with one line changed:

```

---
title: "My Paper"
author:
  - name: Jane Doe
    corresponding: true
    affiliations:
      - name: University of Guelph
format: apaquarto-docx
---

```

There are four format-named functions for this – [apa.quarto.pdf](#), [apa.quarto.docx](#), [apa.quarto.html](#) and [apa.quarto.revealjs](#) – one for each Quarto destination, and they behave identically: the document’s own YAML format: line decides what is actually produced, so the name only states intent. This page’s examples use `apa.quarto.pdf()` to match the format: `apaquarto-pdf` shown above; swap in `apa.quarto.docx()` or `apa.quarto.html()` together with the matching YAML line for Word or HTML.

In the document, the table lives inside a code chunk like this one. [apa.quarto.pdf](#) displays the table body with its title and note included automatically, so this chunk is complete – there is no title or note line to write, and the wording stays correct when the analysis changes. The `library(apaTables)` line is part of the chunk: without it the document cannot find `apa.cor.table()` and the render stops.

```

```{r}
#| label: tbl-correlations
#| message: false
#| warning: true
library(apaTables)
table1 <- apa.cor.table(attitude)
apa.quarto.pdf(table1)
```

```

That `#| label: tbl-correlations` line is not optional: without a label starting `tbl-`, Quarto builds no float for the chunk, so there is no number, no title, and no note, however you try to supply them.

Wide and tall tables. If this table turns out wider than the upright (portrait) PDF page, APA style places it on a rotated (landscape) page – and `apaTables` does that for you. It measures the table

and asks apaquarto for a rotated page when one is needed, whether your tables sit in the text or are collected at the end of the manuscript. There is nothing to add to the chunk and nothing to add to the document's YAML header. A table wider still – wider than even a rotated page – is printed smaller instead, as small as it has to be so that no column is cut off. If that takes the type below 6 points, a warning in the rendered draft says so and points to [apa.save](#) for a full-size Word version. A table too tall for its page needs nothing either: it continues onto the next page by itself, with the column headings repeated at the top of each continuation and the note below the last piece. None of this applies to [apa.save](#) files: there, each table sits on a page of its own, always at full size.

To see this table (and every apaTables table type) in a complete working document, type `apa.quarto.example("apaTables-` in the R console. Two ready-to-render example documents appear in your working directory, with the apaquarto extension installed beside them – one per destination:

```
apaTables-examples-pdf.qmd
apaTables-examples-docx.qmd
```

Render the `-pdf` file for a PDF manuscript, the `-docx` file for a Word manuscript.

Rotated (landscape) pages

A table too wide to stand upright is placed on a rotated page automatically. You can overrule that per table with `landscape = TRUE` or `FALSE`, and per chunk in a Quarto document with `#| apa-landscape: true` or `false`, which wins over the argument.

The same table can be rotated in one output and upright in another, and both are right. `apa.save()` sets tables in 12 point on a page with half inch side margins; a Quarto manuscript sets them in 10 point inside a one inch text block. The table really is wider in the Word file, so it can need a rotated page there while fitting upright in the PDF. Each destination measures the table it is actually going to draw.

See Also

New to Quarto? [apa.quarto.example](#) writes a complete, ready-to-render example document with one apaTables table per page – the fastest way to see every table type working in an apaquarto document. In the R console, type:

```
apa.quarto.example("apaTables-examples.qmd")
```

Prefer a written guide? A step-by-step guide to placing these tables in Quarto documents comes installed with the package. To open it, type:

```
vignette("apaquarto-usage", package = "apaTables")
```

The package website has all of this documentation with rendered examples: <https://dstanley4.github.io/apaTables/>.

Other functions used with this table: [apa.save](#) to save it as a Word or HTML file; [apa.quarto.template](#) and [apa.quarto.pdf](#) to place it in a Quarto document.

Examples

```
library(apaTables)

# Means, standard deviations, and correlations for the attitude data
table1 <- apa.cor.table(attitude)
table1

# Word documents -----
# Save the table as a .docx file. This example writes to a temporary
# folder; for real use give an ordinary name, such as
# apa.save("Table1.docx", table1), and the file appears in your
# working directory.
apa.save(file.path(tempdir(), "table1.docx"), table1)

# Quarto documents -----
# Print a ready-to-paste apaquarto code chunk:
apa.quarto.template(table1)

# Or write your own chunk; apa.quarto.pdf() displays the table body:
apa.quarto.pdf(table1)

# Clean up
unlink(file.path(tempdir(), "table1.docx"))
```

apa.d.table

Create an APA-style table of pairwise d-values

Description

`apa.d.table()` creates a table of standardized mean differences (Cohen's *d*) for all pairwise comparisons of the levels of a grouping variable, in APA style. The table also reports the mean and standard deviation for each group, and each *d*-value is accompanied by its 95% confidence interval. The table prints to the console. The most common next step is saving the table as a Word file to use in a manuscript:

Usage

```
apa.d.table(
  iv,
  dv,
  data,
  filename = NA,
  table.number = 1,
  show.conf.interval = TRUE,
  landscape = NA
)
```

Arguments

| | |
|--------------------|---|
| iv | Name of the grouping (independent) variable column in data, given without quotation marks (for example, <code>iv = dose</code> , not <code>iv = "dose"</code>). |
| dv | Name of the dependent variable column in data, given without quotation marks (for example, <code>dv = libido</code> , not <code>dv = "libido"</code>). |
| data | A data frame. |
| filename | Optional file name. If supplied, the table is also saved as a Word (.docx) or HTML (.html) file, according to the file extension. Legacy .doc/.rtf names are saved as .docx. |
| table.number | Number for the "Table N" line of the output. Defaults to 1. Use 0 (or NA, or NULL) for a table with no number: none is then shown at the console or written to the Word file. When apa.save writes several tables to one file, a table left at the default is numbered by its position in that file, while a number given here is kept. In a Quarto document that uses the <code>apaquarto</code> extension this value is ignored: <code>apaquarto</code> numbers the tables itself, in the order they appear in the document. |
| show.conf.interval | Deprecated and ignored; confidence intervals are always displayed when the MBESS package is installed. This argument will be removed in a later version. |
| landscape | Whether this table is placed on a rotated (landscape) page. The default, NA, means <code>apaTables</code> decides by measuring the table against the page. Pass TRUE to insist on a rotated page or FALSE to insist on an upright one. An explicit value is honoured wherever the destination can carry it out; in a Quarto PDF that means the chunk needs a <code>tbl-</code> label (see apa.quarto.pdf). A table too wide for the page it is given is shrunk to fit rather than having a column cut off. Landscape is a property of a PAGE, so it applies to PDF and Word output and is ignored, without complaint, for HTML and slides. See Rotated (landscape) pages below. |

Details

```
table1 <- apa.d.table(iv = dose, dv = libido, data = viagra)
apa.save("Table1.docx", table1)
```

The file appears in your working directory – the folder R is currently pointing at. See the *Word files (apa.save)* section below. The table can also be placed in a Quarto document, so a rendered manuscript comes out with the table already in place – see the *Quarto documents* section below.

The d-values are calculated with the pooled-variance formulas 4.18 and 4.19 of Borenstein, Hedges, Higgins, and Rothstein (2009) – that is, the mean difference divided by the pooled within-groups standard deviation, with no small-sample correction applied; a d-value is not calculated when unequal variances prevent pooling. Each confidence interval is the exact noncentral t interval computed by `MBESS::ci.smd()`; the **MBESS** package is therefore required for the intervals, and when **MBESS** is not installed the table is created without them.

Value

A table object (class `apa_table`). Typing its name at the console displays the table. Pass it to [apa.save](#) to create a Word or HTML file. Or, if you are working in a Quarto document, use

`apa.quarto.pdf` or `apa.quarto.docx` to make PDF and Word manuscripts, respectively.

Word files (`apa.save`)

`apa.save` writes the finished table to a Word file you can open, copy from, or send – the most common way to use an `apaTables` table:

```
apa.save("Table1.docx", table1)
```

The file appears in your working directory – the folder R is currently pointing at. Several tables can go into one file, each on its own page, numbered Table 1, Table 2, and so on in the order you list them:

```
apa.save("AllTables.docx", table1, table2)
```

The title above each table and the note below it are written by `apaTables` from the table itself; on this route there is no way to supply your own wording at this time. Tables are set in 12 point, and a table too wide for an upright page gets a rotated (landscape) page of its own rather than smaller type – so a wide table coming out sideways is expected, not an error.

Quarto documents

Quarto (<https://quarto.org>) is a free system for writing documents that combine text and R code. The `apaquarto` extension makes a Quarto document follow APA style: place this table in such a document and it appears as a fully formatted APA table – number, title, and note included – in Word, PDF, or HTML. The extension must be installed beside the document, and one command does it. `apa.quarto.new("my-paper")` creates a new manuscript folder with the extension already installed; for a folder you already have, `apa.quarto.setup()` installs the extension there instead. Tables go into the document inside a "code chunk": a small block of R code with a few settings lines at the top. "Rendering" the document turns it into the finished Word, PDF, or HTML file.

Every `apaquarto` document begins with a YAML header: the settings block between `---` lines at the very top of the document. `apaquarto` needs an author there – with no author, or an empty one, the render stops with a LaTeX error – and it reports a problem unless that author has an affiliation and is marked `corresponding: true`. This header is enough to render a PDF manuscript:

```
---
title: "My Paper"
author:
  - name: Jane Doe
    corresponding: true
    affiliations:
      - name: University of Guelph
format: apaquarto-pdf
---
```

For a Word manuscript, the same header with one line changed:

```

---
title: "My Paper"
author:
  - name: Jane Doe
    corresponding: true
    affiliations:
      - name: University of Guelph
format: apaquarto-docx
---

```

There are four format-named functions for this – [apa.quarto.pdf](#), [apa.quarto.docx](#), [apa.quarto.html](#) and [apa.quarto.revealjs](#) – one for each Quarto destination, and they behave identically: the document’s own YAML format: line decides what is actually produced, so the name only states intent. This page’s examples use `apa.quarto.pdf()` to match the format: `apaquarto-pdf` shown above; swap in `apa.quarto.docx()` or `apa.quarto.html()` together with the matching YAML line for Word or HTML.

In the document, the table lives inside a code chunk like this one. [apa.quarto.pdf](#) displays the table body with its title and note included automatically, so this chunk is complete – there is no title or note line to write, and the wording stays correct when the analysis changes. The `library(apaTables)` line is part of the chunk: without it the document cannot find `apa.d.table()` and the render stops.

```

```{r}
#| label: tbl-dvalues
#| message: false
#| warning: true
library(apaTables)
table1 <- apa.d.table(iv = dose, dv = libido, data = viagra)
apa.quarto.pdf(table1)
```

```

That `#| label: tbl-dvalues` line is not optional: without a label starting `tbl-`, Quarto builds no float for the chunk, so there is no number, no title, and no note, however you try to supply them.

Wide and tall tables. If this table turns out wider than the upright (portrait) PDF page, APA style places it on a rotated (landscape) page – and `apaTables` does that for you. It measures the table and asks `apaquarto` for a rotated page when one is needed, whether your tables sit in the text or are collected at the end of the manuscript. There is nothing to add to the chunk and nothing to add to the document’s YAML header. A table wider still – wider than even a rotated page – is printed smaller instead, as small as it has to be so that no column is cut off. If that takes the type below 6 points, a warning in the rendered draft says so and points to [apa.save](#) for a full-size Word version. A table too tall for its page needs nothing either: it continues onto the next page by itself, with the column headings repeated at the top of each continuation and the note below the last piece. None of this applies to [apa.save](#) files: there, each table sits on a page of its own, always at full size.

To see this table (and every `apaTables` table type) in a complete working document, type `apa.quarto.example("apaTables-` in the R console. Two ready-to-render example documents appear in your working directory, with the `apaquarto` extension installed beside them – one per destination:

```
apaTables-examples-pdf.qmd
apaTables-examples-docx.qmd
```

Render the `-pdf` file for a PDF manuscript, the `-docx` file for a Word manuscript.

Rotated (landscape) pages

A table too wide to stand upright is placed on a rotated page automatically. You can overrule that per table with `landscape = TRUE` or `FALSE`, and per chunk in a Quarto document with `#| apa-landscape: true` or `false`, which wins over the argument.

The same table can be rotated in one output and upright in another, and both are right. `apa.save()` sets tables in 12 point on a page with half inch side margins; a Quarto manuscript sets them in 10 point inside a one inch text block. The table really is wider in the Word file, so it can need a rotated page there while fitting upright in the PDF. Each destination measures the table it is actually going to draw.

References

Borenstein, M., Hedges, L. V., Higgins, J. P. T., & Rothstein, H. R. (2009). Introduction to meta-analysis. John Wiley & Sons.

See Also

New to Quarto? apa.quarto.example writes a complete, ready-to-render example document with one `apaTables` table per page – the fastest way to see every table type working in an `apaquarto` document. In the R console, type:

```
apa.quarto.example("apaTables-examples.qmd")
```

Prefer a written guide? A step-by-step guide to placing these tables in Quarto documents comes installed with the package. To open it, type:

```
vignette("apaquarto-usage", package = "apaTables")
```

The package website has all of this documentation with rendered examples: <https://dstanley4.github.io/apaTables/>.

Other functions used with this table: [apa.save](#) to save it as a Word or HTML file; [apa.quarto.template](#) and [apa.quarto.pdf](#) to place it in a Quarto document.

Examples

```
library(apaTables)

# Pairwise d-values for the effect of dose on libido
# (viagra data from Field et al. (2012), Discovering Statistics Using R)
table1 <- apa.d.table(iv = dose, dv = libido, data = viagra)
table1

# Word documents -----
```

```
# Save the table as a .docx file (here, in a temporary
# folder; for real use give an ordinary name, such as
# apa.save("Table1.docx", table1), and the file appears in your
# working directory)
apa.save(file.path(tempdir(), "table1.docx"), table1)

# Quarto documents -----
# Print a ready-to-paste apaquarto code chunk:
apa.quarto.template(table1)

# Or write your own chunk; apa.quarto.pdf() displays the table body:
apa.quarto.pdf(table1)

# Clean up
unlink(file.path(tempdir(), "table1.docx"))
```

| | |
|--------------------|--|
| apa.quarto.example | <i>Write two ready-to-render example documents showing every apaTables table</i> |
|--------------------|--|

Description

The fastest way to learn the Quarto workflow is to see it working. `apa.quarto.example()` writes two complete example manuscripts into your working directory, with everything they need already installed beside them:

```
apa.quarto.example("apaTables-examples.qmd")
```

Two files appear:

```
apaTables-examples-pdf.qmd
apaTables-examples-docx.qmd
```

Both show every apaTables table type, one table per page, each with its APA number, title, and note. They are the same document set up for different destinations: render the `-pdf` file and you get a PDF manuscript (this needs a LaTeX engine such as TinyTeX); render the `-docx` file and you get a Word manuscript (no LaTeX needed). Render with the Preview button in Positron, or in a terminal:

```
quarto render apaTables-examples-docx.qmd
```

Then read the finished manuscript side by side with its `.qmd` source file. Every table in the output was made by a short code chunk you can find in the source, copy, and adapt to your own data – that comparison is the whole lesson. Along the way you will see two tables arrive on rotated (landscape) pages: nothing in the source asks for this. apaTables measures each table as the document renders and rotates the ones too wide to stand upright.

One page in each document (the repeated-measures ANOVA) uses the `afex` and `tidyr` packages, which do not come with apaTables. Install them once with `install.packages(c("afex", "tidyr"))`,

or the render stops on that page. The same instruction is repeated in a comment at the top of each written file.

The apaquarto extension – the part that makes a rendered document follow APA style – is installed in an `_extensions` folder beside the two files automatically, which is why they render with no other setup.

Usage

```
apa.quarto.example(filename = "apaTables-examples.qmd", overwrite = FALSE)
```

Arguments

| | |
|-----------|--|
| filename | Path used as the stem for both files. A <code>.qmd</code> extension, if present, is stripped before deriving the two output names: <code>apa.quarto.example("apaTables-examples.qmd")</code> writes <code>apaTables-examples-pdf.qmd</code> and <code>apaTables-examples-docx.qmd</code> . A bare name (the default) is written in the current working directory (the folder R is currently using); an absolute or <code>~</code> path is written wherever it points. Alongside the two files, this also creates (or repairs, via apa.quarto.setup) an <code>_extensions/wjschne/apaquarto</code> folder in the same directory, so both files render without any extra setup. |
| overwrite | Logical. When <code>FALSE</code> (default) the function stops if either output file already exists; set <code>TRUE</code> to replace existing files. |

Value

A named character vector `c(pdf = ..., docx = ...)` with the paths to the two written files, invisibly. Called for its side effects: it also writes (or repairs) an `_extensions/wjschne/apaquarto` folder beside them, as described under `filename`.

See Also

[apa.quarto.new\("my-paper"\)](#) starts an APA manuscript with the apaquarto extension already installed; [apa.quarto.setup\(\)](#) installs the extension into a project you already have. A step-by-step guide to placing apaTables tables in Quarto documents also comes installed with the package. To open it, type:

```
vignette("apaquarto-usage", package = "apaTables")
```

Related functions: [apa.quarto.pdf](#), [apa.quarto.template](#).

Examples

```
# Write both example documents to a temporary folder
out <- file.path(tempdir(), "apaTables-examples.qmd")
apa.quarto.example(filename = out)

# (delete the demo files, and the extension folder installed beside them)
unlink(file.path(tempdir(),
                  c("apaTables-examples-pdf.qmd",
                    "apaTables-examples-docx.qmd")))
unlink(file.path(tempdir(), "_extensions"), recursive = TRUE)
```

apa.quarto.new

Start a new APA manuscript that renders on the first try

Description

One command gives you a working APA manuscript:

```
apa.quarto.new("my-paper")
```

This creates a my-paper folder holding my-paper.qmd – a Quarto document with a working APA header and one worked table – and, beside it, an _extensions folder holding the extension that makes the rendered document follow APA style. Open my-paper.qmd and click Render (the Preview button in Positron), or run quarto render in that folder: the very first render produces a complete APA PDF. The starter is set up for PDF; for a Word manuscript, change its format: apaquarto-pdf line to format: apaquarto-docx.

What gets installed is not the standard apaquarto release. It is apaTables' own tested copy: apaquarto plus table fixes that are not yet in the standard release. So always install it with this function (or [apa.quarto.setup](#) for a folder you already have), never with Quarto's own quarto add, which gets the standard release instead.

The _extensions folder is the part you cannot write by hand: a document's settings block only names a format, and the folder is what supplies it. Without it the document does not render at all – Quarto stops with Unable to read the extension 'apaquarto' before any R code runs, so apaTables never gets the chance to install it for you.

Usage

```
apa.quarto.new(path, filename = NULL, overwrite = FALSE)
```

Arguments

| | |
|-----------|---|
| path | Folder to create. It is created if it does not exist. |
| filename | Name for the document. Defaults to the folder's own name with a .qmd extension. |
| overwrite | Overwrite an existing document of that name? |

Value

Invisibly, a list with path, document and setup (what [apa.quarto.setup](#) did).

Quarto documents

Quarto (<https://quarto.org>) is a free system for writing documents that combine text and R code. apaquarto is a Quarto extension that makes such a document follow APA style – tables get their number, italic title, and *Note*. line automatically, in Word, PDF, or HTML. In the R console, type:

```
library(apaTables)
apa.quarto.new("my-paper")
```

This creates a folder called `my-paper` containing `my-paper.qmd`. The document's YAML header (the settings block between `---` lines at the top) already has everything `apaquarto` needs to render: a title, an author with `corresponding: true` and an affiliation (`apaquarto` stops with a LaTeX error if the author is missing or has neither), and `format: apaquarto-pdf` –

```
---
title: "My Paper"
shorttitle: "My Paper"
author:
  - name: Your Name
    corresponding: true
    email: you@example.org
    affiliations:
      - name: Your University
format: apaquarto-pdf
---
```

plus one worked `apaTables` table already in the body. Beside the document, `apa.quarto.new()` also installs an `_extensions/wjschne/apaquarto` folder (via [apa.quarto.setup](#)), so opening `my-paper/my-paper.qmd` and clicking `Render` – or running `quarto render` in that folder – produces a complete APA PDF on the first try, with no terminal command to install anything first.

Do **not** run `quarto add wjschne/apaquarto` or `quarto update extension wjschne/apaquarto` in the folder this creates: both commands silently replace the patched extension with the plain, unpatched one, while reporting `[No Change]`. That reintroduces bugs in table numbering and in where a long table's note lands. If you have already run one of those commands, running [apa.quarto.setup\(\)](#) again puts the tested copy back.

Sending a coauthor or reviewer the whole project folder carries `_extensions/` with it, so they do nothing extra. Sending only the `.qmd` file does not: they must run `apa.quarto.setup()` in that folder themselves before rendering, because with the extension folder absent, Quarto fails before any R code runs.

See Also

A step-by-step guide to placing `apaTables` tables in Quarto documents comes installed with the package – see its "Installing the `apaquarto` extension" section. To open it, type:

```
vignette("apaquarto-usage", package = "apaTables")
```

Related functions: [apa.quarto.setup](#) to install or repair the extension in a project you already have, [apa.quarto.example](#).

Examples

```
# Start a new manuscript in a temporary folder
proj <- file.path(tempdir(), "apaTables-quarto-new-example")
```

```
apa.quarto.new(proj)

# (delete the demo folder)
unlink(proj, recursive = TRUE)
```

apa.quarto.revealjs *Display an apaTables table in a Quarto document*

Description

You have made a table, and your manuscript is a Quarto document. Put the table in with a short code chunk, using the function named for the document's format: line:

Usage

```
apa.quarto.revealjs(x, title = NULL, note = NULL, font.size = NULL)

apa.quarto.pdf(x, title = NULL, note = NULL, font.size = NULL)

apa.quarto.docx(x, title = NULL, note = NULL, font.size = NULL)

apa.quarto.html(x, title = NULL, note = NULL, font.size = NULL)
```

Arguments

| | |
|-----------|---|
| x | An apaTables table (an apa.*.table() result or an apa_table_spec). |
| title | Optional table title, overriding the one apaTables generated. A plain character string, or markdown (e.g. "Study *N* by Group"). A tbl-cap settings line in the chunk still takes precedence over this argument. Default NULL uses the generated title. On apa.quarto.revealjs() this is different , because a slide reads no tbl-cap: the default NULL puts no title on the slide at all, and TRUE (rather than a string) requests the generated title – see Slides are different below. |
| note | Optional table note, overriding the one apaTables generated. A plain character string, or markdown. An apa-note settings line in the chunk still takes precedence over this argument. Default NULL uses the generated note. A note given here is one paragraph; apaquarto starts a new paragraph wherever the text contains a double quote, a comma and a double quote in a row, so a note written with that sequence in it is split there. This is long-standing apaquarto behaviour, and the same is true of a hand-written apa-note settings line. A generated note is unaffected. On apa.quarto.revealjs() this is different , because a slide reads no apa-note: the default NULL puts no note on the slide at all, and TRUE (rather than a string) requests the generated note – see Slides are different below. |
| font.size | Optional explicit type size in points for the table body, overriding the automatic rotate-then-shrink sequence. The title above the table and the note below it keep their usual size. PDF, HTML and slides use the size exactly, fractions included. |

Word is different: it takes a table's type size from a style in the document template, so it uses the requested size rounded down to a whole point, kept between 6 and 12, and says in the rendered draft when it had to adjust. A size below 6 points, where an APA table stops being comfortable to read, warns on every destination. Nothing is ever refused or cut off. Default NULL lets apaTables choose.

Details

```
```{r}
#| label: tbl-correlations
#| message: false
#| warning: true
library(apaTables)
table1 <- apa.cor.table(attitude)
apa.quarto.pdf(table1)
```
```

Render the document, and the table appears fully APA-formatted – number, italic title, and *Note*. included – with nothing else written. (New to Quarto? It is a free system for writing documents that combine text and R code, at <https://quarto.org>, and `apa.quarto.new("my-paper")` creates a working APA manuscript with everything already installed. A slide deck works differently; see **Slides are different** below.)

There are four of these functions, one named for each destination – `apa.quarto.pdf()`, `apa.quarto.docx()`, `apa.quarto.html()`, `apa.quarto.revealjs()`. The first three put a table on a manuscript page and behave identically: put one of them on the last line of a table's code chunk and the table appears in APA style, number, title and note included, with nothing else written. **The name states intent; the document's own YAML format: line decides what is actually produced.** A document rendered with `format: apaquarto-docx` gets a correct Word table even from a chunk that calls `apa.quarto.pdf()`. Quarto's output format is a property of the document, not of a chunk, so a function name can only ever state what you meant. If the name and the render disagree, the rendered draft carries one line saying exactly which YAML line would make them agree, for example:

This document renders to ``apaquarto-pdf``. To produce Word, set ``format: apaquarto-docx`` in the YAML.

The table itself is unaffected either way – the message is guidance only. `apa.quarto.template` writes the whole chunk for you, and its `target` argument picks which of the four names it prints.

`apa.quarto.revealjs()` goes on a slide deck rather than a manuscript page, and differs from the other three in several ways that matter – see its own **Slides are different** section below.

The title and note come from the table itself by default on the three manuscript destinations. Three things can override them, each winning over the one after it: the chunk's own `tbl-cap` / `apa-note` settings lines, then the `title` / `note` arguments below, then the generated value. Supplying both a chunk option and the matching argument is not an error, but only the chunk option is used. This is identical across the three manuscript destinations; `apa.quarto.revealjs()` does not use `tbl-cap` or `apa-note` at all, because a slide deck has no float for either to attach to (see **Slides are different** below). `apa.quarto.template` writes the whole chunk for you to copy; the chunk it prints has no `tbl-cap` or `apa-note` line at all, because both are automatic – add one only for your own wording.

A chunk needs a label to become a table at all. Give it a settings line `#| label: tbl-something` – `apa.quarto.template` always writes one. Without a `tbl-` label, Quarto builds no float for the chunk, so there is no number, no title (generated or written by hand), and no note, no matter what `tbl-cap`, `apa-note`, `title`, or `note` say. (This does not apply to `apa.quarto.revealjs()`: a slide deck builds no float at all, with or without a label, so a table appears regardless – see **Slides are different** below for how its title and note are requested instead.)

Nothing needs to be said about the size of the table or the shape of the page. When a table is too wide for an upright (portrait) page, it is placed on a rotated (landscape) page automatically, at full size. That is what APA style asks for: "if a table is too wide to fit on one page, use landscape orientation on the page with the wide table." A table too wide to fit even at its smallest readable size is printed smaller still instead – as small as it has to be, so that no column is cut off. When that takes the type below 6 points, where an APA table stops being comfortable to read, a warning in the rendered draft says so and points to `apa.save` for a full-size Word version; the table itself is still complete. A table taller than the page continues onto the next one, with the column headings repeated and the note below the last piece. (On an HTML render there is no page at all, so none of this applies: nothing rotates and nothing shrinks – the table renders at full size regardless of width. On a slide render there is no page either, so nothing rotates there. But a slide's fixed viewport means shrinking IS meaningful, and `apa.quarto.revealjs()` uses its own separate automatic sizing to fit the table on the slide rather than this rotate-then-shrink sequence; see **Slides are different** below.)

There is one table shape `apaTables` cannot rescue. The space between columns is a fixed 4 points and does not shrink with the type, so a table of more than roughly 136 columns is wider than a rotated page at every possible font size. Such a table is warned about explicitly. No `apaTables` function produces one from real data.

Automatic measurement can be overruled per table with the `landscape` argument every builder accepts, and per chunk with a settings line, which wins over the argument. Add `#| apa-landscape: true` to a code chunk to place that table on a rotated page even though it would otherwise stand upright, or `#| apa-landscape: false` to forbid rotation and keep it on an upright page instead. A wide table is then shrunk to fit rather than losing a column.

Rotating a page needs a label. Give every table chunk a settings line `#| label: tbl-something` – the label a table needs anyway for its number and for references to it. A chunk without one is not a table as far as the document is concerned, so its page cannot be turned. A wide table in such a chunk is printed smaller on the upright page instead, with every column kept. If the chunk asked for a rotated page, the rendered draft says the label is what is missing.

Value

The table, ready for the document (a `knit_asis` object).

Slides are different

`apa.quarto.revealjs()` places a table on a slide rather than a manuscript page, and it differs from `apa.quarto.pdf()` / `apa.quarto.docx()` / `apa.quarto.html()` in ways worth knowing before you rely on it.

revealjs is the supported slide format. Set `format: revealjs` in the document's YAML – everything below describes that format. **PowerPoint is not supported yet.** The obstacle is not something a PowerPoint template or setting can fix. Quarto renders PowerPoint through pandoc, and pandoc's PowerPoint writer gives no way to set a table's text size from the document, so a table placed on

a PowerPoint slide comes out at PowerPoint's own type size. Even at a smaller size, a larger table (a two-block regression table, for instance) is taller than a slide. Support for PowerPoint is planned for a future release; apaTables has no working route to a PowerPoint table yet. If a document using `apa.quarto.revealjs()` renders to PowerPoint anyway, the table still appears – styled only by PowerPoint's own default table look, not APA format – and a warning says PowerPoint is not yet supported and names format: `revealjs` as the supported destination.

There is no table number, ever. A manuscript has a running list of tables to number against – "Table 1", "Table 2", and so on. A slide deck has no such list, so `apa.quarto.revealjs()` never adds a number.

Title and note are opt-in, not automatic. `apa.quarto.revealjs(x)` on its own puts the table alone on the slide, nothing else. Pass `title = TRUE` and/or `note = TRUE` to add the title and note apaTables generated for the table – the same arguments work here as on `apa.quarto.pdf()` and its siblings. This matters most for a correlation table: its cells mark a significant correlation with **, and the sentence explaining what ** means (* indicates $p < .05$. ** indicates $p < .01$.) lives in the note. Left at the default, a correlation table on a slide shows the stars with their meaning nowhere on the slide – `note = TRUE` is the remedy:

```
my_table <- apa.cor.table(attitude)
apa.quarto.revealjs(my_table, note = TRUE)
```

An explicit font.size is honoured exactly, the same value you typed, though a size below 6 points – where an APA table stops being comfortable to read – warns in the rendered draft. Left at the default, apaTables shrinks the table automatically to fit the slide.

Apart from the number, title, note, and sizing, a slide table is the same table every other destination draws: identical values, identical formatting.

Journal mode (a PDF manuscript setting) is not supported

This section concerns PDF manuscripts only. A slide deck never uses `apaquarto`, so it is unaffected; so are Word and HTML output.

`apaquarto` can typeset a manuscript as a printed journal article (`documentmode: jou`), in two narrow columns. Do not put apaTables tables in such a document. A two-column page cannot break a table across pages, so rows, columns and whole notes can be missing from the rendered PDF with nothing reported anywhere. A document of four ordinary tables came out with two of the notes and several variable names simply absent. A warning says so during the render, but it is a warning and not a repair: there is no setting that makes journal mode safe. Render the manuscript with `documentmode: man` instead, which is what a journal is submitted in, or save the table as a Word file with [apa.save](#).

See Also

[apa.quarto.new\("my-paper"\)](#) starts an APA manuscript with the `apaquarto` extension already installed; [apa.quarto.setup\(\)](#) installs the extension into a project you already have. A step-by-step guide to placing apaTables tables in Quarto documents also comes installed with the package. To open it, type:

```
vignette("apaquarto-usage", package = "apaTables")
```

A table leaves apaTables by one of two routes, and this is one of them. `apa.quarto.pdf()` / `apa.quarto.docx()` / `apa.quarto.html()` / `apa.quarto.revealjs()` put a table inside a Quarto document (a manuscript page or a slide deck); `apa.save` writes a table in a Word (.docx) or HTML file of its own. `apa.quarto.template` writes the whole code chunk for you to copy; `apa.quarto.new` creates a Quarto project that renders an APA manuscript on the first try.

Examples

```
# Build the table once; display it with the entry point named for the
# document's own format: line
my_table <- apa.cor.table(attitude)

# In a document whose YAML says format: apaquarto-pdf
apa.quarto.pdf(my_table)

# format: apaquarto-docx
apa.quarto.docx(my_table)

# format: apaquarto-html
apa.quarto.html(my_table)

# format: revealjs -- a slide has no automatic title or note, so pass
# note = TRUE to show the note that explains the significance stars
apa.quarto.revealjs(my_table)
apa.quarto.revealjs(my_table, note = TRUE)
```

apa.quarto.setup

Install or repair this project's apaquarto extension

Description

You already have a project folder, and you want apaTables tables to come out APA-formatted when its documents render. Run this once, in the R console, from that folder:

```
apa.quarto.setup()
```

It puts the apaquarto extension into an `_extensions` folder beside your document. **What gets installed is not the standard apaquarto release:** it is apaTables' own tested copy – apaquarto plus table fixes that are not yet in the standard release. That is also why this command exists at all. Running Quarto's own `quarto add` or `quarto update` in a project silently replaces the tested copy with the standard one – while reporting [No Change] – which brings back bugs in table numbering and in where a long table's note lands. Running `apa.quarto.setup()` again puts the tested copy back.

It is safe to run at any time: on a project that is already correct it does nothing and says nothing. When it does repair something during a render, the report reaches the render log rather than the manuscript itself, so a repaired document still renders with nothing extra typeset into it.

Usage

```
apa.quarto.setup(path = ".", auto = TRUE, force = FALSE)
```

Arguments

| | |
|-------|---|
| path | Project folder. Defaults to the current directory, or, during a render, the folder whose <code>_extensions</code> the render is using. |
| auto | Replace a copy that is not ours (the standard <code>apaquarto</code> , or one restored by <code>quarto update</code>)? Defaults to <code>TRUE</code> . |
| force | Overwrite a copy that IS ours but has been edited locally? Defaults to <code>FALSE</code> , so your own edits survive. |

Value

Invisibly, a list with `action`, `version`, `found` and `path`.

Quarto documents

Quarto (<https://quarto.org>) is a free system for writing documents that combine text and R code. `apaquarto` is a Quarto extension that makes such a document follow APA style: `apaTables` tables placed in it appear as fully formatted APA tables – number, title, and note included – in Word, PDF, or HTML. `apaquarto` is installed per project folder, not into R itself, which is what `apa.quarto.setup()` does. Call it once, in the R console, from the folder that contains – or will contain – your `.qmd` document:

```
library(apaTables)
apa.quarto.setup()
```

This puts an `_extensions/wjschne/apaquarto` folder next to your document, and its YAML header – the settings block between `---` lines at the very top of the document – can then select an `apaquarto` format. `apaquarto` needs an author in that header – with no author, or an empty one, the render stops with a LaTeX error – and it reports a problem unless that author has an affiliation and is marked `corresponding: true`. This header is enough to render:

```
---
title: "My Paper"
author:
  - name: Jane Doe
    corresponding: true
    affiliations:
      - name: University of Guelph
format: apaquarto-pdf
---
```

`apa.quarto.setup()` is safe to run at any time – on a project whose copy is already correct, it does nothing and says nothing. Do **not** run `quarto add wjschne/apaquarto` or `quarto update extension wjschne/apaquarto` in one of these folders: both commands silently replace the patched copy with the plain, unpatched extension, while reporting `[No Change]`. That reintroduces bugs in

table numbering and in where a long table's note lands. If you have already run one of those commands, running `apa.quarto.setup()` again puts the tested copy back.

Sending a coauthor or reviewer the whole project folder carries `_extensions/` with it, so they do nothing extra. Sending only the `.qmd` file does not: they must run `apa.quarto.setup()` in that folder themselves before rendering, because with the extension folder absent, Quarto fails before any R code runs, so nothing can repair itself automatically.

See Also

A step-by-step guide to placing apaTables tables in Quarto documents comes installed with the package – see its "Installing the apaquarto extension" section. To open it, type:

```
vignette("apaquarto-usage", package = "apaTables")
```

Related functions: [apa.quarto.new](#) to start a whole new manuscript in one command, [apa.quarto.template](#), [apa.quarto.example](#).

Examples

```
# Install the apaquarto extension into a temporary project folder
proj <- file.path(tempdir(), "apaTables-quarto-setup-example")
dir.create(proj, showWarnings = FALSE)
apa.quarto.setup(proj)

# Running it again on the same, already-correct folder does nothing
apa.quarto.setup(proj)

# (delete the demo folder)
unlink(proj, recursive = TRUE)
```

`apa.quarto.template` *Print a ready-to-paste apaquarto chunk for a table*

Description

You have made a table and want it in your Quarto document, but you would rather not write the code chunk yourself. Ask for it, in the R console:

Usage

```
apa.quarto.template(
  x,
  label = NULL,
  code = NULL,
  target = c("pdf", "docx", "html", "revealjs")
)
```

Arguments

| | |
|---------------------|--|
| <code>x</code> | An apaTables table (an <code>apa.*.table()</code> result or an <code>apa_table_spec</code>). |
| <code>label</code> | A length-1 character string naming the chunk. By default the name is built from the name of the table given for <code>x</code> , so <code>apa.quarto.template(table3)</code> names the chunk <code>tbl-table3</code> ; when <code>x</code> is not a stored table but an expression written out in full, the name is <code>tbl-mytable</code> . A label you supply yourself is used exactly as given, so make it begin with <code>"tbl-"</code> – Quarto only treats a chunk as a table when its label does. Writing <code>@tbl-table3</code> in your document text then produces that table’s number (for example, “Table 1”). |
| <code>code</code> | A length-1 character string with the R code that produces the table, used in the display line of the chunk. You can usually ignore this argument: by default, whatever you typed for <code>x</code> is reused in the chunk, so <code>apa.quarto.template(table1)</code> prints the line <code>apa.quarto.pdf(table1)</code> . |
| <code>target</code> | Which of the format-named entry points to print the chunk for – “pdf” (the default), “docx”, “html”, or “revealjs”. This only picks which name the printed chunk uses; because that name is forgiving, the chunk still renders correctly however the document’s own YAML format: line is actually set. Picking the matching name simply keeps the printed chunk consistent with the YAML, and changes the wide-table advice below to name the right page (a docx table rotates like a PDF one; an HTML table has no page at all, so nothing rotates and nothing shrinks – it renders at full width regardless, and nothing is printed about landscape). <code>target = “revealjs”</code> is different in one more way: a slide deck has no float, so the chunk it prints has no <code># label: tbl-...</code> line either – see apa.quarto.revealjs for why a deck has no table number, and why title and note are opt-in there instead of automatic. |

Details

```
table1 <- apa.cor.table(attitude)
apa.quarto.template(table1)
```

This prints a finished chunk, named after your table:

```
```{r}
#| label: tbl-table1
#| message: false
#| warning: true
table1 <- apa.cor.table(...) # <- the line that builds your table
apa.quarto.pdf(table1)
```
```

Copy it into your .qmd document, replace the comment with the line that builds `table1` – `library(apaTables)` with it – and render. The chunk needs nothing more: the table’s title and note are written from the table itself, every time you render, so there is no `tbl-cap` or `apa-note` line to fill in and nothing that can quietly stop matching your analysis.

The display line matches your document’s format: line – the `target` argument below picks [apa.quarto.pdf](#) (the default), [apa.quarto.docx](#), [apa.quarto.html](#), or [apa.quarto.revealjs](#). `target = “revealjs”`

prints a chunk showing the table alone, because a slide adds no title or note unless asked (see **Slides are different** on [apa.quarto.revealjs](#)'s page). The printed output is self-documenting either way: it says where to put the chunk. Call this function in the R console – not inside the Quarto document itself – and copy the chunk it prints into your .qmd file.

The chunk is always two lines: build the table, then display it. If you passed the table-building call itself – `apa.quarto.template(apa.cor.table(attitude))` – both lines are ready to run. If you passed a table you had already built, by name, the first line is a *comment* showing where your own table-building line belongs; replace it with that line, or delete it if the table is built in an earlier chunk.

The chunk's #| label: line is not optional on the three manuscript destinations, unlike the optional `tbl-cap` and `apa-note` lines this function no longer prints. A chunk without a label beginning `tbl-` is not a table as far as Quarto is concerned, so it gets no number, no title – generated or written by hand – and no note. This function always writes one for you on those three destinations; keep it if you write the chunk yourself. A `target = "revealjs"` chunk has no `#| label:` line at all, and needs none: a slide deck has no float, so a label would buy nothing.

Every apaquarto document begins with a YAML header: the settings block between `---` lines at the very top of the document. apaquarto needs an author there – with no author, or an empty one, the render stops with a LaTeX error – and it reports a problem unless that author has an affiliation and is marked `corresponding: true`. This header is enough to render:

```
---
title: "My Paper"
author:
  - name: Jane Doe
    corresponding: true
    affiliations:
      - name: University of Guelph
format: apaquarto-pdf
---
```

In the R console, create the table and then ask for its Quarto code:

```
library(apaTables)
table1 <- apa.cor.table(attitude)
apa.quarto.template(table1)
```

The printed chunk displays `table1` but does not create it, so the chunk you paste into your Quarto document must also carry the lines that do – `library(apaTables)` among them, without which the render stops with `could not find function "apa.cor.table"`. This particular table fits an upright (portrait) PDF page, and a table that does not is handled without anything being added to the chunk – see *wide tables* below:

```
```{r}
#| label: tbl-table1
#| message: false
#| warning: true
library(apaTables)
```

```
table1 <- apa.cor.table(attitude)
apa.quarto.pdf(table1)
```

```

Nothing in that chunk mentions the title or the note, and nothing needs to: `apa.quarto.pdf()` writes both from the table itself, with the text styling intact (italics and super/subscripts render as they would anywhere else in the document, and a multi-part note starts each part, such as the significance-star sentence, on its own line). They are written again on every render, so they stay correct when the analysis changes. To use your own wording instead, add a `tbl-cap` line for the title or an `apa-note` line for the note, or pass `title = / note =` to the entry point function – whichever is supplied wins over the generated version. [apa_caption](#) returns the generated title if you would rather start from it than write one from scratch.

Wide tables. When the table is too wide for an upright (portrait) PDF page, APA style places it on a rotated (landscape) page, and that happens by itself: `apaTables` measures the table with the same calculation the chunk will use, and asks `apaquarto` for a rotated page when one is needed. This works whether your tables sit in the text or are collected at the end of the manuscript, and there is nothing to add to the chunk and nothing to add to the document's YAML header. When the table given here is a wide one, this function says so and reports its width, so you know why a page in your document will be turned. A document that already wraps a chunk in `:: {.landscape}` markers keeps rendering exactly as before.

Tall tables. A table too tall for the page it goes on continues onto the next one by itself, with the column headings repeated at the top of each continuation and the note below the last piece. There is no settings line to add: every `apaTables` table is written in a form that can be split across pages, and `apaquarto` places it accordingly.

Run this function in the R console and copy the printed chunk into your Quarto document.

Every chunk in a Quarto document must have a different name, so the chunk's name is built from the name of the table you asked for: `apa.quarto.template(table3)` names the chunk `tbl-table3`. Two different tables therefore give two chunks that can both be pasted into the same document. Dots and underscores in the table's name become hyphens (`my.table` gives `tbl-my-table`), because a dot inside a name would break the `@tbl-` reference that produces the table's number in your text. When there is no name to use – because the table was written out in full rather than stored under a name – the chunk is named `tbl-mytable`, and a second such chunk in the same document needs its name changed by hand. Give the `label` argument to choose the name yourself.

Value

The chunk text as a single character string, returned invisibly (most users simply copy the printed output; the surrounding instructions are not part of the returned text).

See Also

[apa.quarto.new\("my-paper"\)](#) starts an APA manuscript with the `apaquarto` extension already installed; [apa.quarto.setup\(\)](#) installs the extension into a project you already have. A step-by-step guide to placing `apaTables` tables in Quarto documents also comes installed with the package. To open it, type:

```
vignette("apaquarto-usage", package = "apaTables")
```

Examples

```
# Print a ready-to-paste chunk for the table (PDF document by default)
table1 <- apa.cor.table(attitude)
apa.quarto.template(table1)

# The same chunk for a Word document
apa.quarto.template(table1, target = "docx")
```

apa.reg.table

Create an APA-style regression table

Description

`apa.reg.table()` creates a regression table in APA style from one or more [lm](#) results. For each predictor the table reports the unstandardized regression weight (b), the standardized regression weight (beta), the semi-partial correlation squared, and the zero-order correlation, each with its confidence interval, along with model fit (R-squared). When several `lm` results are supplied (blocks in a hierarchical regression), the table also reports the change in R-squared for each block. The table prints to the console. The most common next step is saving the table as a Word file to use in a manuscript:

Usage

```
apa.reg.table(
  ...,
  filename = NA,
  table.number = 1,
  prop.var.conf.level = 0.95,
  landscape = NA
)
```

Arguments

| | |
|---------------------------|--|
| <code>...</code> | One or more results from lm – one for each block in a hierarchical regression. All must use the same criterion variable. |
| <code>filename</code> | Optional file name. If supplied, the table is also saved as a Word (<code>.docx</code>) or HTML (<code>.html</code>) file, according to the file extension. Legacy <code>.doc/.rtf</code> names are saved as <code>.docx</code> . |
| <code>table.number</code> | Number for the "Table N" line of the output. Defaults to 1. Use 0 (or NA, or NULL) for a table with no number: none is then shown at the console or written to the Word file. When apa.save writes several tables to one file, a table left at the default is numbered by its position in that file, while a number given here is kept. In a Quarto document that uses the <code>apaquarto</code> extension this value is ignored: <code>apaquarto</code> numbers the tables itself, in the order they appear in the document. |

| | |
|----------------------------------|---|
| <code>prop.var.conf.level</code> | Confidence level (.90 or .95, the default) for the intervals around the proportion-of-variance measures (semi-partial correlation squared, R-squared, and change in R-squared). Using .90 keeps conclusions based on these intervals consistent with the p-value for the block. |
| <code>landscape</code> | Whether this table is placed on a rotated (landscape) page. The default, NA, means <code>apaTables</code> decides by measuring the table against the page. Pass TRUE to insist on a rotated page or FALSE to insist on an upright one. An explicit value is honoured wherever the destination can carry it out; in a Quarto PDF that means the chunk needs a <code>tbl-</code> label (see apa.quarto.pdf). A table too wide for the page it is given is shrunk to fit rather than having a column cut off. Landscape is a property of a PAGE, so it applies to PDF and Word output and is ignored, without complaint, for HTML and slides. See Rotated (landscape) pages below. |

Details

```
blk1 <- lm(sales ~ adverts + airplay, data = album)
table1 <- apa.reg.table(blk1)
apa.save("Table1.docx", table1)
```

The file appears in your working directory – the folder R is currently pointing at. See the *Word files (apa.save)* section below. The table can also be placed in a Quarto document, so a rendered manuscript comes out with the table already in place – see the *Quarto documents* section below.

Confidence intervals for the change in R-squared, and for the semi-partial correlation squared of each predictor when the model has two or more predictors, are calculated with the asymptotic approach of Alf and Graf (1999). Note that Algina, Keselman, and Penfield (2008) found this approach can under some circumstances lead to inaccurate confidence intervals on proportion-of-variance values.

The confidence interval for R-squared – and, when a model has a single predictor, the interval for that predictor’s semi-partial correlation squared, which then equals R-squared – is computed by `MBESS::ci.R2()` using the noncentral F distribution for fixed predictors.

Value

A table object (class `apa_table`). Typing its name at the console displays the table. Pass it to [apa.save](#) to create a Word or HTML file. Or, if you are working in a Quarto document, use [apa.quarto.pdf](#) or [apa.quarto.docx](#) to make PDF and Word manuscripts, respectively.

Word files (apa.save)

[apa.save](#) writes the finished table to a Word file you can open, copy from, or send – the most common way to use an `apaTables` table:

```
apa.save("Table1.docx", table1)
```

The file appears in your working directory – the folder R is currently pointing at. Several tables can go into one file, each on its own page, numbered Table 1, Table 2, and so on in the order you list them:

```
apa.save("AllTables.docx", table1, table2)
```

The title above each table and the note below it are written by `apaTables` from the table itself; on this route there is no way to supply your own wording at this time. Tables are set in 12 point, and a table too wide for an upright page gets a rotated (landscape) page of its own rather than smaller type – so a wide table coming out sideways is expected, not an error.

Quarto documents

Quarto (<https://quarto.org>) is a free system for writing documents that combine text and R code. The `apaquarto` extension makes a Quarto document follow APA style: place this table in such a document and it appears as a fully formatted APA table – number, title, and note included – in Word, PDF, or HTML. The extension must be installed beside the document, and one command does it. `apa.quarto.new("my-paper")` creates a new manuscript folder with the extension already installed; for a folder you already have, `apa.quarto.setup()` installs the extension there instead. Tables go into the document inside a "code chunk": a small block of R code with a few settings lines at the top. "Rendering" the document turns it into the finished Word, PDF, or HTML file.

Every `apaquarto` document begins with a YAML header: the settings block between `---` lines at the very top of the document. `apaquarto` needs an author there – with no author, or an empty one, the render stops with a LaTeX error – and it reports a problem unless that author has an affiliation and is marked corresponding: `true`. This header is enough to render a PDF manuscript:

```
---
title: "My Paper"
author:
  - name: Jane Doe
    corresponding: true
    affiliations:
      - name: University of Guelph
format: apaquarto-pdf
---
```

For a Word manuscript, the same header with one line changed:

```
---
title: "My Paper"
author:
  - name: Jane Doe
    corresponding: true
    affiliations:
      - name: University of Guelph
format: apaquarto-docx
---
```

There are four format-named functions for this – `apa.quarto.pdf`, `apa.quarto.docx`, `apa.quarto.html` and `apa.quarto.revealjs` – one for each Quarto destination, and they behave identically: the document's own YAML `format` line decides what is actually produced, so the name only states intent. This page's examples use `apa.quarto.pdf()` to match the `format: apaquarto-pdf` shown above;

swap in `apa.quarto.docx()` or `apa.quarto.html()` together with the matching YAML line for Word or HTML.

In the document, the table lives inside a code chunk like this one. [apa.quarto.pdf](#) displays the table body with its title and note included automatically, so this chunk is complete – there is no title or note line to write, and the wording stays correct when the analysis changes. The `library(apaTables)` line is part of the chunk: without it the document cannot find `apa.reg.table()` and the render stops.

```
```{r}
#| label: tbl-regression
#| message: false
#| warning: true
library(apaTables)
blk1 <- lm(sales ~ adverts + airplay, data = album)
table1 <- apa.reg.table(blk1)
apa.quarto.pdf(table1)
```
```

That `#| label: tbl-regression` line is not optional: without a label starting `tbl-`, Quarto builds no float for the chunk, so there is no number, no title, and no note, however you try to supply them.

Wide and tall tables. If this table turns out wider than the upright (portrait) PDF page, APA style places it on a rotated (landscape) page – and `apaTables` does that for you. It measures the table and asks `apaquarto` for a rotated page when one is needed, whether your tables sit in the text or are collected at the end of the manuscript. There is nothing to add to the chunk and nothing to add to the document’s YAML header. A table wider still – wider than even a rotated page – is printed smaller instead, as small as it has to be so that no column is cut off. If that takes the type below 6 points, a warning in the rendered draft says so and points to [apa.save](#) for a full-size Word version. A table too tall for its page needs nothing either: it continues onto the next page by itself, with the column headings repeated at the top of each continuation and the note below the last piece. None of this applies to [apa.save](#) files: there, each table sits on a page of its own, always at full size.

To see this table (and every `apaTables` table type) in a complete working document, type `apa.quarto.example("apaTables-` in the R console. Two ready-to-render example documents appear in your working directory, with the `apaquarto` extension installed beside them – one per destination:

```
apaTables-examples-pdf.qmd
apaTables-examples-docx.qmd
```

Render the `-pdf` file for a PDF manuscript, the `-docx` file for a Word manuscript.

Rotated (landscape) pages

A table too wide to stand upright is placed on a rotated page automatically. You can overrule that per table with `landscape = TRUE` or `FALSE`, and per chunk in a Quarto document with `#| apa-landscape: true` or `false`, which wins over the argument.

The same table can be rotated in one output and upright in another, and both are right. `apa.save()` sets tables in 12 point on a page with half inch side margins; a Quarto manuscript sets them in 10 point inside a one inch text block. The table really is wider in the Word file, so it can need a rotated page there while fitting upright in the PDF. Each destination measures the table it is actually going to draw.

References

- Alf Jr, E. F., & Graf, R. G. (1999). Asymptotic confidence limits for the difference between two squared multiple correlations: A simplified approach. *Psychological Methods*, 4(1), 70-75.
- Algina, J., Keselman, H. J., & Penfield, R. J. (2008). Note on a confidence interval for the squared semipartial correlation coefficient. *Educational and Psychological Measurement*, 68(5), 734-741.
- Kelley, K. (2007). Constructing confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software*, 20(8), 1-24.
- Steiger, J. H., & Fouladi, R. T. (1992). R2: A computer program for interval estimation, power calculation, and hypothesis testing for the squared multiple correlation. *Behavior Research Methods, Instruments, & Computers*, 24, 581-582.

See Also

New to Quarto? [apa.quarto.example](#) writes a complete, ready-to-render example document with one apaTables table per page – the fastest way to see every table type working in an apaquarto document. In the R console, type:

```
apa.quarto.example("apaTables-examples.qmd")
```

Prefer a written guide? A step-by-step guide to placing these tables in Quarto documents comes installed with the package. To open it, type:

```
vignette("apaquarto-usage", package = "apaTables")
```

The package website has all of this documentation with rendered examples: <https://dstanley4.github.io/apaTables/>.

Other functions used with this table: [apa.save](#) to save it as a Word or HTML file; [apa.quarto.template](#) and [apa.quarto.pdf](#) to place it in a Quarto document.

Examples

```
library(apaTables)

# Single-block regression predicting album sales
# (album data from Field et al. (2012), Discovering Statistics Using R)
blk1 <- lm(sales ~ adverts + airplay, data = album)
table1 <- apa.reg.table(blk1, table.number = 1)
table1

# Word documents -----
# Save the table as a .docx file (here, in a temporary
# folder; for real use give an ordinary name, such as
# apa.save("Table1.docx", table1), and the file appears in your
# working directory)
apa.save(file.path(tempdir(), "table1.docx"), table1)

# Quarto documents -----
# Print a ready-to-paste apaquarto code chunk:
```

```

apa.quarto.template(table1)

# Or write your own chunk; apa.quarto.pdf() displays the table body:
apa.quarto.pdf(table1)

# Clean up
unlink(file.path(tempdir(), "table1.docx"))

# Hierarchical regression -----
# Two blocks; more than two can be used
blk1 <- lm(sales ~ adverts, data = album)
blk2 <- lm(sales ~ adverts + airplay + attract, data = album)
table2 <- apa.reg.table(blk1, blk2, table.number = 2)

# Interaction product-term test with blocks
blk1 <- lm(sales ~ adverts + airplay, data = album)
blk2 <- lm(sales ~ adverts + airplay + I(adverts * airplay), data = album)
table3 <- apa.reg.table(blk1, blk2, table.number = 3)

# Interaction product-term test with blocks and additional product terms
blk1 <- lm(sales ~ adverts + airplay, data = album)
blk2 <- lm(sales ~ adverts + airplay + I(adverts*adverts) + I(airplay*airplay),
           data = album)
blk3 <- lm(sales ~ adverts + airplay + I(adverts*adverts) + I(airplay*airplay) +
           I(adverts*airplay), data = album)
table4 <- apa.reg.table(blk1, blk2, blk3, table.number = 4)

# Interaction product-term test with a single regression
# (i.e., semi-partial correlation focus)
blk1 <- lm(sales ~ adverts + airplay + I(adverts * airplay), data = album)
table5 <- apa.reg.table(blk1, table.number = 5)

# apa.save() can write several tables into one Word document
apa.save(file.path(tempdir(), "regression_tables.docx"),
         table1, table2, table3, table4, table5)

# Clean up
unlink(file.path(tempdir(), "regression_tables.docx"))

```

apa.save

Save apaTables tables to a Word (.docx) or HTML (.html) file

Description

apa.save() writes one or more finished tables to a single file – the way to get an apaTables table into a manuscript you write in Word, or to give a journal its tables as a file of their own:

Usage

```
apa.save(filename, ..., paper = "us")
```

Arguments

| | |
|----------|--|
| filename | Output filename ending in .docx or .html (a .doc/.rtf name is written as .docx). |
| ... | One or more apaTables table objects to save. |
| paper | Page size for .docx: "us" (default) or "a4". Capitalization is forgiven ("A4" works); any other value is an error. |

Details

```
table1 <- apa.cor.table(attitude)
apa.save("Table1.docx", table1)
```

The file appears in your working directory – the folder R is currently pointing at. Open it in Word and the finished APA table is there: the "Table N" line, the italic title, the body, and the *Note*. below it. Copy it into your manuscript, or send the file as it is. The format is chosen by the file extension: .docx for Word, .html for a web page (a legacy .doc/.rtf name is written as .docx). `apa.save()` does not write PDF files.

Several tables can go into one file, each on a page of its own, in the order you list them:

```
apa.save("AllTables.docx", table1, table2, table3)
```

A table left with its default number is numbered by its position in the file – Table 1, Table 2, Table 3 above – while a number set when the table was built is kept.

Tables are set in 12 point, matching the manuscript body type APA style expects. Each page is upright (portrait) unless the table is too wide for it: `apa.save()` measures every table, and one that will not fit upright is placed on a landscape page at full size – columns are never shrunk or cut off. To decide the orientation yourself, use the `landscape` argument of the function that built the table – for example `apa.cor.table(attitude, landscape = TRUE)`; an explicit value always wins over measurement.

The title and the note are written from the table itself, so they stay correct when the analysis changes; there is no title or note argument here at this time. If you need particular wording, edit the finished Word file.

Value

Invisibly returns the filename actually written.

See Also

The functions that make the tables this saves: [apa.cor.table](#), [apa.reg.table](#), [apa.aov.table](#), [apa.1way.table](#), [apa.2way.table](#), [apa.afex.table](#), and [apa.d.table](#).

If your manuscript lives in a Quarto document – text and analysis in one file – place tables with [apa.quarto.pdf](#) or [apa.quarto.docx](#) there instead of saving separate files.

The package website has all of this documentation with rendered examples: <https://dstanley4.github.io/apaTables/>.

Examples

```
library(apaTables)

table1 <- apa.1way.table(iv = dose, dv = libido,
                        data = viagra, table.number = 1)

apa.save(filename = file.path(tempdir(), "Table1.docx"), table1)

# delete demo file
unlink(file.path(tempdir(), "Table1.docx"))
```

apa_caption

Caption text for a table (for the apaquarto tbl-cap settings line)

Description

Returns the table title as a plain-text string, suitable for pasting into the `tbl-cap` settings line of a table chunk (a small block of R code) in an apaquarto document (a Quarto document using the apaquarto APA-style extension). Returns "" if the table has no title. Does not include "Table N".

Usage

```
apa_caption(x)
```

Arguments

`x` An apaTables table (an `apa.*.table()` result or an `apa_table_spec`).

Value

A length-1 character string.

See Also

[apa.quarto.new\("my-paper"\)](#) starts an APA manuscript with the apaquarto extension already installed; [apa.quarto.setup\(\)](#) installs the extension into a project you already have. A step-by-step guide to placing apaTables tables in Quarto documents also comes installed with the package. To open it, type:

```
vignette("apaquarto-usage", package = "apaTables")
```

Examples

```
my_table <- apa.cor.table(attitude)
apa_caption(my_table)
```

| | |
|---------------|---|
| apa_note_text | <i>Note text for a table (for the apaquarto apa-note settings line)</i> |
|---------------|---|

Description

Returns the table note as a plain-text string, suitable for pasting into the apa-note settings line of a table chunk (a small block of R code) in an apaquarto document (a Quarto document using the apaquarto APA-style extension). Returns "" if the table has no notes. Does not include "Note."

Usage

```
apa_note_text(x)
```

Arguments

x An apaTables table (an apa.*.table() result or an apa_table_spec).

Value

A length-1 character string.

See Also

[apa.quarto.new\("my-paper"\)](#) starts an APA manuscript with the apaquarto extension already installed; [apa.quarto.setup\(\)](#) installs the extension into a project you already have. A step-by-step guide to placing apaTables tables in Quarto documents also comes installed with the package. To open it, type:

```
vignette("apaquarto-usage", package = "apaTables")
```

Examples

```
my_table <- apa.cor.table(attitude)
apa_note_text(my_table)

# Extend the generated wording rather than replace it
paste(apa_note_text(my_table), "Data collected in 2026.")
```

 dating_wide

dating data from textbook

Description

A data set from Field et al (2012)

Usage

```
data(dating_wide)
```

Format

A data frame with 20 rows and 11 columns. Gender is a between subjects variable. Looks and Personality are within subject variables. Both gender and participant are factors.

participant Factor: Participant ID number

gender Factor: Gender: Male/Female

attractive_high Date rating where looks=attractive and personality=high

average_high Date rating where looks=average and personality=high

ugly_high Date rating where looks=ugly and personality=high

attractive_some Date rating where looks=attractive and personality=some

average_some Date rating where looks=average and personality=some

ugly_some Date rating where looks=ugly and personality=some

attractive_none Date rating where looks=attractive and personality=none

average_none Date rating where looks=average and personality=none

ugly_none Date rating where looks=ugly and personality=none

Source

<https://studysites.sagepub.com/dsur/study/>

References

Field, A., Miles, J., & Field, Z. (2012). Discovering statistics using R. Sage.

| | |
|---------------------|--|
| drink_attitude_wide | <i>drink attitude data from textbook</i> |
|---------------------|--|

Description

A data set from Field et al (2012)

Usage

```
data(drink_attitude_wide)
```

Format

A data frame with 20 rows and 10 columns. Drink and Imagery are within subject variables. Participant is a factor.

participant Factor: Participant ID number

beer_positive Attitude where drink=beer and imagery=positive

beer_negative Attitude where drink=beer and imagery=negative

beer_neutral Attitude where drink=beer and imagery=neutral

wine_positive Attitude where drink=wine and imagery=positive

wine_negative Attitude where drink=wine and imagery=negative

wine_neutral Attitude where drink=wine and imagery=neutral

water_positive Attitude where drink=water and imagery=positive

water_negative Attitude where drink=water and imagery=negative

water_neutral Attitude where drink=water and imagery=neutral

Source

<https://studysites.sagepub.com/dsur/study/>

References

Field, A., Miles, J., & Field, Z. (2012). Discovering statistics using R. Sage.

| | |
|---------|---------------------|
| Eysenck | <i>Eysenck data</i> |
|---------|---------------------|

Description

A data set from Howell (2012)

Usage

```
data(Eysenck)
```

Format

A data frame with 100 rows and 3 variables:

Age Young or Old

Condition Experimental learning condition

Recall Level of word recall

Source

<https://www.uvm.edu/~statdhtx/methods8/DataFiles/Tab13-2.dat>

References

Howell, D. (2012). Statistical methods for psychology. Cengage Learning.

| | |
|-----------------|--|
| fidler_thompson | <i>Fidler & Thompson (2001) Fixed-Effects ANOVA data</i> |
|-----------------|--|

Description

A data set from Fidler & Thompson (2001)

Usage

```
data(fidler_thompson)
```

Format

A data frame with 24 rows and 3 variables:

a Independent variable: a

b Independent variable: b

dv Dependent variable: dv

References

Fidler, F., & Thompson, B. (2001). Computing correct confidence intervals for ANOVA fixed- and random-effects effect sizes. *Educational and Psychological Measurement*, 61(4), 575-604.

```
get.ci.partial.eta.squared
```

Calculates confidence interval for partial eta-squared in a fixed-effects ANOVA

Description

Calculates confidence interval for partial eta-squared in a fixed-effects ANOVA

Usage

```
get.ci.partial.eta.squared(F.value, df1, df2, conf.level = 0.9)
```

Arguments

| | |
|------------|---|
| F.value | The F-value for the fixed-effect |
| df1 | Degrees of freedom for the fixed-effect |
| df2 | Degrees of freedom error |
| conf.level | Confidence level (0 to 1). For partial eta-squared a confidence level of .90 is traditionally used rather than .95. |

Value

List with confidence interval values (LL and UL)

Examples

```
# Smithson (2001) p. 619
get.ci.partial.eta.squared(F.value=6.00, df1=1, df2=42, conf.level=.90)
get.ci.partial.eta.squared(F.value=2.65, df1=6, df2=42, conf.level=.90)
get.ci.partial.eta.squared(F.value=2.60, df1=6, df2=42, conf.level=.90)

# Fidler & Thompson (2001) Fixed Effects 2x4 p. 594 (Table 6) / p. 596 (Table 8)
get.ci.partial.eta.squared(F.value=1.50, df1=1, df2=16, conf.level=.90)
get.ci.partial.eta.squared(F.value=4.00, df1=3, df2=16, conf.level=.90)
get.ci.partial.eta.squared(F.value=1.50, df1=3, df2=16, conf.level=.90)
```

| | |
|----------------------|-----------------------------------|
| <code>goggles</code> | <i>goggles data from textbook</i> |
|----------------------|-----------------------------------|

Description

A data set from Field et al (2012)

Usage

```
data(goggles)
```

Format

A data frame with 48 rows and 4 variables:

participant Participant identification number

gender Gender of participant

alcohol Amount alcohol consumed

attractiveness Perceived attractiveness

Source

<https://studysites.sagepub.com/dsur/study/>

References

Field, A., Miles, J., & Field, Z. (2012). Discovering statistics using R. Sage.

| | |
|---------------------|----------------------------------|
| <code>viagra</code> | <i>viagra data from textbook</i> |
|---------------------|----------------------------------|

Description

A data set from Field et al (2012)

Usage

```
data(viagra)
```

Format

A data frame with 15 rows and 2 variables:

dose Level of viagra dose

libido Libido after taking viagra

Source

<https://studysites.sagepub.com/dsur/study/>

References

Field, A., Miles, J., & Field, Z. (2012). *Discovering statistics using R*. Sage.

Index

* datasets

- album, 4
 - dating_wide, 58
 - drink_attitude_wide, 59
 - Eysenck, 60
 - fidler_thompson, 60
 - goggles, 62
 - viagra, 62
- album, 4
- apa.1way.table, 3, 5, 55
- apa.2way.table, 3, 9, 55
- apa.afex.table, 3, 14, 55
- apa.aov.table, 3, 20, 55
- apa.cor.table, 3, 25, 55
- apa.d.table, 3, 30, 55
- apa.quarto.docx, 6, 7, 10, 12, 15, 17, 21, 23, 27, 28, 32, 33, 46, 50, 51, 55
- apa.quarto.docx (apa.quarto.revealjs), 39
- apa.quarto.example, 8, 13, 18, 24, 29, 34, 35, 38, 45, 53
- apa.quarto.html, 7, 12, 17, 23, 28, 33, 46, 51
- apa.quarto.html (apa.quarto.revealjs), 39
- apa.quarto.new, 6, 11, 16, 22, 27, 32, 36, 37, 40, 42, 43, 45, 48, 51, 56, 57
- apa.quarto.pdf, 6–8, 10, 12, 13, 15, 17, 18, 21, 23, 24, 26–29, 31–34, 36, 46, 50–53, 55
- apa.quarto.pdf (apa.quarto.revealjs), 39
- apa.quarto.revealjs, 7, 12, 17, 23, 28, 33, 39, 46, 47, 51
- apa.quarto.setup, 6, 11, 16, 22, 27, 32, 36–38, 42, 43, 48, 51, 56, 57
- apa.quarto.template, 8, 13, 18, 24, 29, 34, 36, 40, 41, 43, 45, 45, 53
- apa.reg.table, 3, 49, 55
- apa.save, 5, 6, 8, 10–13, 15, 17, 18, 21–24, 26, 27, 29, 31–34, 41–43, 49, 50, 52, 53, 54
- apa_caption, 48, 56
- apa_note_text, 57
- apaTables (apaTables-package), 3
- apaTables-package, 3
- cor.test, 27
- dating_wide, 58
- drink_attitude_wide, 59
- Eysenck, 60
- fidler_thompson, 60
- get.ci.partial.eta.squared, 61
- goggles, 62
- lm, 20, 21, 49
- viagra, 62