

# Package ‘qreport’

May 9, 2026

**Type** Package

**Title** Statistical Reporting with 'Quarto'

**Version** 1.1-0

**Date** 2025-09-30

**Description** Provides statistical components, tables, and graphs that are useful in 'Quarto' and 'RMarkdown' reports and that produce 'Quarto' elements for special formatting such as tabs and marginal notes and graphs. Some of the functions produce entire report sections with tabs, e.g., the missing data report created by `missChk()`. Functions for inserting variables and tables inside 'graphviz' and 'mermaid' diagrams are included, and so are special clinical trial graphics for adverse event reporting.

**Depends** Hmisc ( $\geq 5.1-1$ ), ggplot2

**Imports** rms, Formula, viridis, knitr, htmltools, data.table

**Suggests** tth

**License** GPL ( $\geq 2$ )

**Encoding** UTF-8

**LazyLoad** Yes

**URL** <https://hbiostat.org/R/qreport/>

**RoxygenNote** 7.3.3

**NeedsCompilation** no

**Author** Frank E Harrell Jr [aut, cre]

**Maintainer** Frank E Harrell Jr <fh@fharrell.com>

**Repository** CRAN

**Date/Publication** 2025-10-02 13:50:02 UTC

## Contents

addCap	2
aePlot	3
asForm	5

conVars . . . . .	6
dataChk . . . . .	7
dataOverview . . . . .	8
disVars . . . . .	9
dNeedle . . . . .	10
getqreportOption . . . . .	10
hookaddcap . . . . .	11
hooktime . . . . .	12
htmlList . . . . .	12
htmlView . . . . .	13
htmlViewx . . . . .	14
kabl . . . . .	15
makecallout . . . . .	16
makecnote . . . . .	17
makecodechunk . . . . .	18
makecolmarg . . . . .	19
makedot . . . . .	20
makegraphviz . . . . .	22
makegvflow . . . . .	23
makemermaid . . . . .	25
maketabs . . . . .	26
missChk . . . . .	27
multDataOverview . . . . .	29
pdumpit . . . . .	30
printCap . . . . .	30
putQcap . . . . .	31
runDeriveExpr . . . . .	32
rwrap . . . . .	33
sampleFrac . . . . .	33
saveCap . . . . .	34
scplot . . . . .	35
setqreportOption . . . . .	36
spar . . . . .	37
timeMar . . . . .	38
varType . . . . .	39
vClus . . . . .	40

<b>Index</b>	<b>43</b>
--------------	-----------

---

addCap	<i>addCap</i>
--------	---------------

---

## Description

Add Figure Captions to a Dataset

**Usage**

```
addCap(label = NULL, cap = NULL, scap = NULL)
```

**Arguments**

label	figure label to use if not fetched from chunk information
cap	caption to use if not from chunk
scap	short caption to use if not from chunk

**Details**

Fetches the figure caption and optional short caption from the currently running code chunk (under knitr) and appends them to a running caption dataset named `.captions.` in the global environment. This facilitates customizing a table of figures in a report.

**Value**

invisible list with `label`, `cap`, `scap`

**Author(s)**

Frank Harrell

**Examples**

```
## Not run:  
# Called from inside a knitr chunk and all information pulled from  
# chunk information  
addCap()  
  
## End(Not run)
```

---

aePlot

*Adverse Event Plot*

---

**Description**

Generates graphics for binary event proportions

**Usage**

```
aePlot(  
  formula,  
  data = NULL,  
  subset = NULL,  
  na.action = na.retain,  
  exposure = NULL,
```

```

expunit = "",
study = " ",
refgroup = NULL,
minincidence = 0,
conf.int = 0.95,
etype = "adverse events",
head = NULL,
tail = NULL,
size = c("regular", "wide"),
popts = NULL,
label = NULL
)

```

### Arguments

formula	a formula with one or two left hand variables (the first representing major categorization and the second minor), and 1-2 right hand variables. One of the right hand variables may be enclosed in <code>id()</code> to indicate the presence of a unique subject ID. The remaining variable is treatment.
data	input data frame
subset	subsetting criteria
na.action	function for handling NAs when creating analysis frame
exposure	a numeric vector whose length is the number of treatments, with names equal to the treatment names
expunit	character string specifying the time units for exposure
study	character string identifying the study; used in multi-study reports or where distinct patient strata are analyzed separately. Used to fetch the study-specific metadata stored by <code>setqreportOption</code> . Single study reports just use <code>study=' '</code> .
refgroup	a character string specifying which treatment group is subtracted when computing risk differences. If there are two treatments the default is the first one listed in <code>qreport</code> options.
minincidence	a number between 0 and 1 specifying the minimum incidence in any stratum that must hold before an event is included in the summary. When exposure is given, <code>minincidence</code> applies to the hazard rate.
conf.int	confidence level for difference in proportions (passed to <code>dotchartpl</code> )
etype	a character string describing the nature of the events, for example "adverse events", "serious adverse events". Used in figure captions.
head	character string. Specifies initial text in the figure caption, otherwise a default is used.
tail	a character string to add to end of automatic caption
size	default is standard text body width. Set to "wide" to render plot with <code>column: page-inset-left</code> .
popts	a list of additional options to pass to <code>dotchartpl</code>
label	label for figure. <code>fig-</code> is placed in front of this label. Default uses the name of the code chunk. If a label is defined by the time the graph is produced that label will be used instead of the code chunk.

**Details**

Generates dot charts showing proportions of subjects having events (at any time). Events can be categorized by a single level or by major and minor levels (e.g., body system and preferred terms). When there are two treatments, half-width CLs of treatment differences are drawn, centered at the midpoint of the two proportions, and CLs for differences appear in hover text. Input data must contain one record per event, with this record containing the event name. If there is more than one event of a given type per subject, unique subject ID must be provided. Denominators come from `qreport` options when computing event incidence proportions. Instead, when a named vector exposure is specified, with names equal to the treatments, exposure is used as the denominator so that the exponential distribution hazard rate is computed, i.e., events per unit of exposure time. When a subject has only one event of each type, the usual interpretation holds. When a subject has multiple events, the estimate is events per person per time unit. A character variable `expunit` defines the time units. It is assumed that only randomized subjects are included in the dataset. Whenever the number of events of a given type is zero for a group, the event frequency is changed to 0.5 so that one may compute confidence intervals on the log scale as well as hazard ratios.

For an example with output see <https://hbiostat.org/rflow/descript.html#adverse-event-chart/>

**Value**

no return value, called for knitting with `knit`

**Author(s)**

Frank Harrell

**Examples**

```
# See test.Rnw in tests directory
```

---

asForm

*asForm*

---

**Description**

Convert Vector of Variables Names to a Right-Sided Formula

**Usage**

```
asForm(x)
```

**Arguments**

x                    character vector

**Details**

Given a vector of character strings, turns them into a formula with no left hand side variable.

**Value**

formula

**Author(s)**

Frank Harrell

**Examples**

```
asForm(letters[1:6])
```

---

conVars

*conVars*

---

**Description**

Find Continuous Variables

**Usage**

```
conVars(...)
```

**Arguments**

...                    passed to [varType()]

**Details**

Uses [varType()] to type the variables then retrieves the vector of names of continuous ones.

**Value**

character vector

**Author(s)**

Frank Harrell

**Examples**

```
## Not run:  
conVars(mydata)  
  
## End(Not run)
```

---

dataChk	<i>dataChk</i>
---------	----------------

---

## Description

Run a Series of Data Checks and Report

## Usage

```
dataChk(
  d,
  checks,
  id = character(),
  html = FALSE,
  omit0 = FALSE,
  byid = FALSE,
  nrows = 500
)
```

## Arguments

d	a data table
checks	a vector of expressions that if satisfied causes records to be listed
id	option vector of variable names to serve as IDs
html	set to TRUE to create HTML output and put each check in a separate tab, also creating summary tabs
omit0	set to TRUE to ignore checks finding no observations
byid	if id is given set byid=TRUE to also list a data frame with all flagged conditions, sorted by id
nrows	maximum number of rows to allow to be printed

## Details

Function to run various data checks on a data table.

Checks are run separately for each part of the expression vector checks. For each single expression, the variables listed in the output are all the variables mentioned in the expression plus optional variables whose names are in the character vector id. `%between% c(a,b)` in expressions is printed as `[a, b]`. The output format is plain text unless `html=TRUE` which also puts each table in a separate Quarto tab. See [here](#) for examples.

## Value

an invisible data frame containing variables check (the expression checked) and n (the number of records satisfying the expression)

**Author(s)**

Frank Harrell

**Examples**

```
## Not run:  
dataChk(mydata)  
  
## End(Not run)
```

---

`dataOverview``dataOverview`

---

**Description**

Produce a Data Overview Quarto Section

**Usage**

```
dataOverview(  
  d,  
  d2 = NULL,  
  id = NULL,  
  plot = c("scatter", "dot", "none"),  
  pr = nvar <= 50,  
  which = 1,  
  dec = 3  
)
```

**Arguments**

<code>d</code>	a data frame or table
<code>d2</code>	optional second dataset used for analyzing uniqueness of subject IDs
<code>id</code>	optional formula providing names of subject identifiers
<code>plot</code>	specifies type of plot, defaulting to 'scatter'
<code>pr</code>	set to FALSE to omit detailed table and present only graphics
<code>which</code>	when two datasets are given which one should be the focus
<code>dec</code>	certain summary statistics are rounded to the nearest dec places

**Details**

Produces a multi-tabbed dataset overview as exemplified [here](#). This includes provision of data about data such as variable type, symmetry, missingness, rarest and most common values.

**Value**

nothing; renders a report with Quarto/RMarkdown

**Author(s)**

Frank Harrell

**Examples**

```
## Not run:  
dataOverview(mydata, secondarydataset)  
  
## End(Not run)
```

---

disVars

*disVars*

---

**Description**

Find Discrete Variables

**Usage**

```
disVars(...)
```

**Arguments**

...            passed to [varType()]

**Details**

Uses [varType()] to type the variables then retrieves the vector of names of discrete ones.

**Value**

character vector

**Author(s)**

Frank Harrell

**Examples**

```
## Not run:  
disVars(mydata)  
  
## End(Not run)
```

---

dNeedle	<i>Draw Needles</i>
---------	---------------------

---

**Description**

Create an html base64 string from a png graphic to draw needles for current sample sizes. Uses colors set by call to setqreportOptions.

**Usage**

```
dNeedle(sf, study = " ")
```

**Arguments**

sf	output of sampleFrac
study	character string specifying study ID

**Value**

a base64 representation of a png graphic, suitable for inclusion in html

**Examples**

```
setqreportOption(tx.var='treatment', denom=c(enrolled=1000, randomized=800, a=398, b=402))
dNeedle(sampleFrac(getqreportOption('denom')))
```

---

getqreportOption	<i>Get qreport Options</i>
------------------	----------------------------

---

**Description**

Get qreport options, assigning default values of unspecified options.

**Usage**

```
getqreportOption(opts = NULL, study = " ")
```

**Arguments**

opts	character vector containing list of option names to retrieve. If only one element, the result is a scalar, otherwise a list. If opts is not specified, a list with all current option settings is returned.
study	character string specifying an optional study designation

**Value**

getching qreport options

**Examples**

```
## Not run:  
getqreportOption('tx.var')  
  
## End(Not run)
```

---

hookaddcap

*hookaddcap*

---

**Description**

Set knitr to Automatically Call addCap in Figure-Producing Chunks

**Usage**

```
hookaddcap(loc = NULL)
```

**Arguments**

loc                    if non-NULL will be used to set the knitr chunk option `fig.cap.location`

**Details**

Adds a knitr hook that takes effect before the chunk is run. The hook function retrieves figure information from the current chunk to give to addCap.

**Value**

nothing; calls knitr hook and chunk option setting functions

**Author(s)**

Frank Harrell

**Examples**

```
## Not run:  
hookaddcap()  
  
## End(Not run)
```

hooktime                    *hooktime*

---

**Description**

Create knitr Hook for Reporting Execution Time for Chunks

**Usage**

```
hooktime(all = FALSE)
```

**Arguments**

all                    set to TRUE to time every chunk without the need for time=TRUE in the chunk header

**Details**

Creates a hook called `time` that can be activated by including `time=TRUE` in a chunk header. The chunk's execution time in seconds will be printed in a very small html font at the end of the chunk's output.

**Value**

nothing

**Author(s)**

Frank Harrell

**See Also**

[this](#) and [timeMar\(\)](#)

---

htmlList                    *htmlList*

---

**Description**

Print Named List of Vectors

**Usage**

```
htmlList(x, dec = 4)
```

**Arguments**

x                    a named list  
dec                  round to this decimal place

**Details**

Function to print a simple named list of vectors in html Creates a column name from the names of the list If a vector element of the list is numeric, it is rounded to dec digits to the right of the decimal place.

**Value**

a kable

**Author(s)**

Frank Harrell

**Examples**

```
set.seed(1)
w <- list(A = runif(4), B=rnorm(3))
htmlList(w)
```

---

htmlView	<i>htmlView</i>
----------	-----------------

---

**Description**

Convert Objects to HTML and View

**Usage**

```
htmlView(...)
```

**Arguments**

...                    any number of objects for which an html method exists

**Details**

Converts a series of objects created to html. Displays these in the RStudio View pane. If RStudio is not running displays in an external browser. Assumes there is an html method for the objects (e.g., objects are result of Hmisc::describe or Hmisc::contents. User can page through the different outputs with the arrow keys in the RStudio View pane

**Value**

nothing is returned; used to launch a browser on html text

**Author(s)**

Frank Harrell

**Examples**

```
## Not run:
htmlView(contents(d1), contents(d2))
htmlView(describe(d1), describe(d2, descript='Second Dataset'))
htmlView(contents(d), describe(d))

## End(Not run)
```

---

htmlViewx

*htmlViewx*


---

**Description**

Convert to HTML and Eternally View Objects

**Usage**

```
htmlViewx(..., tab = c("notfirst", "all", "none"))
```

**Arguments**

... a series of objects for which an 'html' method exists  
 tab set to 'all' to add even the first object to an existing window.

**Details**

'htmlViewx' is similar to 'htmlView' except that an external viewer is launched, and the first object is opened in a new window. Subsequent objects are opened in a new tab in the last created window. Set 'options(vbrowser='command line to run browser')' to use a browser other than 'Vivaldi'. Defaults to opening a new window for only the first object, and adding tabs after that.

**Value**

does not return a value; launches a browser

**Author(s)**

Frank Harrell

**Examples**

```
## Not run:
options(prType='html')
htmlViewx(contents(d), describe(d))

## End(Not run)
```

---

kabl	<i>kabl</i>
------	-------------

---

**Description**

Front-end to kable and kables

**Usage**

```
kabl(..., caption = NULL, digits = 4, col.names = NA, row.names = NA)
```

**Arguments**

...	one or more objects to pass to kable
caption	overall single caption
digits	passed to kable and applies to all tables
col.names	passed to kable
row.names	passed to kable

**Details**

Calls kable() if only one table is to be printed. Calls kable() for each table and passes it to kables if more than one. Accounts for results of tapply not being a vector (is an array)

**Value**

result of kable or kables

**Author(s)**

Frank Harrell

**Examples**

```
kabl(data.frame(a=1:2, b=3:4), data.frame(x=11:13, y=21:23))
```

---

`makecallout`*makecallout*

---

## Description

General Case Handling of Quarto Callouts

## Usage

```
makecallout(...)
```

## Arguments

...

can be any of the following

- `x` object to print (if `type='print'`), or one or more formulas whose right hand sides are to be run. Left side provides labels if needed by the particular callout, and if `raw` is included on the right side any R code chunk run will have `results='asis'` in the chunk header.
- callout character string giving the Quarto callout
- label character string label if needed and if not obtained from the left side of a formula
- `type` defaults to `'print'` to print an object. Set to `'run'` to run a chunk or `'cat'` to use `cat()` to render.
- `now` set to `FALSE` to return code instead of running it
- `results` if not using formulas, specifies the formatting option to code in the code header, either `'asis'` (the default) or `'markup'`
- `close` specifies whether to close the callout or to leave it open for future calls
- parameters passed to `print`

## Details

This function generates and optionally runs markdown/R code that runs Quarto callouts such as collapsible notes or marginal notes. Before rendering `x`, `options(rawmarkup=TRUE)` is set so that `Hmisc::rendHTML` will not try to protect html in things like margins. Quarto doesn't like the surrounding html protection lines in that context. The option is set back to its original value after rendering.

## Value

if code is not executed, returns a character vector with the code to run

## Author(s)

Frank Harrell

**Examples**

```
x <- 1:3
co <- '.callout-note collapse="true'
makecallout(x, callout=co, label='# thislabel', type='print')
makecallout(thislabel ~ x, callout=co, type='print')
```

---

makecnote

*makecnote*


---

**Description**

Print an Object in a Collapsible Note

**Usage**

```
makecnote(
  x,
  label = paste0("`", deparse(substitute(x)), "`"),
  wide = FALSE,
  type = c("print", "run", "cat"),
  ...
)
```

**Arguments**

<code>x</code>	an object having a suitable print method
<code>label</code>	a character string providing a title for the tab. Default is the name of the argument passed to <code>makecnote</code> .
<code>wide</code>	set to <code>TRUE</code> to expand the width of the text body
<code>type</code>	default is to print; can also be run, cat
<code>...</code>	an optional list of arguments to be passed to print

**Details**

Prints an object in a Quarto collapsible note.

**Value**

nothing is returned, used for rendering markup

**Author(s)**

Frank Harrell

**Examples**

```
makecnote('This is some text', label='mylab', wide=TRUE)
```

makecodechunk

*makecodechunk*

---

**Description**

Create Text for Running Code Chunk

**Usage**

```
makecodechunk(  
  cmd,  
  opts = NULL,  
  results = "asis",  
  lang = "r",  
  callout = NULL,  
  h = NULL,  
  w = NULL  
)
```

**Arguments**

cmd	character string vector of commands to run inside chunk
opts	optional list of chunk options, e.g. <code>list(fig.width=6, fig.cap="This is a caption")</code> . See <a href="https://yihui.org/knitr/options/">https://yihui.org/knitr/options/</a> for a complete list of options.
results	format of results, default is 'asis'. May specify <code>results='markup'</code> .
lang	language for the chunk
callout	an optional Quarto callout to include after <code># </code> after the chunk header that affects how the result appears, e.g. <code>callout='column: margin'</code>
h	optional height to place after the chunk header after <code># </code>
w	optional width

**Details**

Creates text strings suitable for running through `knitr`. The chunk is given a random name because certain operations are not allowed by `knitr` without it.

**Value**

character vector

**Author(s)**

Frank Harrell

**Examples**

```
makecodechunk('x <- pi; print(x)')
```

---

makecolmarg	<i>makecolmarg</i>
-------------	--------------------

---

**Description**

Put an Object in the Margin

**Usage**

```
makecolmarg(x, type = c("print", "run", "cat"), ...)
```

**Arguments**

x	an object having a suitable print method
type	type of execution
...	an optional list of arguments to be passed to print

**Details**

Prints an object in a Quarto column margin.

**Value**

nothing is returned, used to render markup

**Author(s)**

Frank Harrell

**Examples**

```
makecolmarg(data.frame(x=1:3, y=4:6))
```

---

 makedot

*Create a General Quarto Graphviz Flow Diagram from Plain Text  
With Variable Insertions*


---

## Description

Takes a character string or vector and creates a graphviz dot flowchart from the input. There are two kinds of input strings or lines. One defines nodes, the syntax being `nodename:string` where `nodename` is any valid graphviz node name and `string` represents a character string with possible variable value substitutions of the form `{{variablename}}`. `string` is automatically wrapped by having newline characters inserted to keep lines within `width` characters. The second kind of input line is one specifying node connections with graphviz dot syntax, respecting node names defined in earlier strings. Unlike `makegvflow()`, this approach allows a node to connect to any number of other nodes, and allows any flow direction.

## Usage

```
makedot(
  .object.,
  ...,
  direction = c("TD", "LR"),
  shape = "box",
  fontsize = 24,
  nodesetup = NULL,
  file,
  width = 30,
  onlyprint = FALSE
)
```

## Arguments

<code>.object.</code>	character string or vector of plain text plus possible $\LaTeX$ math delimited by single dollar signs. An empty initial line is ignore, so the user need not worry about having an initial quote mark on a line by itself.
<code>...</code>	<code>name=value</code> pairs that makes values replace <code>{{name}}</code> elements in the markup
<code>direction</code>	direction of the flow chart, default is top-down
<code>shape</code>	node shape
<code>fontsize</code>	text font size
<code>nodesetup</code>	extra node setup to include after direction
<code>file</code>	name of file to hold graphviz markup after variable insertions. Run this in Quarto using a chunk to looks like what is below, which was for <code>file='graphviz.dot'</code> .
<code>width</code>	text width for word-wrapping
<code>onlyprint</code>	set to TRUE to only print the generated graphiz markup and not save it to file

```

```{dot}
//| label: fig-flow1
//| fig-height: 4
//| fig-cap: "Chart caption"
//| file: graphviz.dot
```

```

## Details

If a node label contains `\n`, all `\n` are replaced with `\n` and ordinary word wrapping is suppressed. This is how users take control of line splitting. If a node label ends in `//any text`, the text in `any text` will be copied to inside the node definition after the label is closed with double quote and after a space is added. This allows node colors and other attributes to be changed.

Precede single quotes in node labels with a single backslash. Precede double quotes with double backslashes.

The function uses `knitr::knit_expand()` to apply variable insertions before the diagram is rendered by Quarto. See [this](#) for an example. `##' @title makedot`

## Value

nothing; used to `knitr::knit_expand()` graphviz markup

## Author(s)

Frank Harrell

## See Also

[makegraphviz\(\)](#), [makegvflow\(\)](#)

## Examples

```

x <- '
A:Some string of text for {{jj}}
B:Another string of test, this one being longer and needing breaking
C:A third string defining a node//style="filled,rounded", fillcolor=lightblue
A -> B
B -> C [label="some text along connecting arc"]
C -> A'
# Can also use A -> {B, C}
makedot(x, jj='tiger', onlyprint=TRUE)

```

---

|              |                     |
|--------------|---------------------|
| makegraphviz | <i>makegraphviz</i> |
|--------------|---------------------|

---

## Description

Create a Quarto Graphviz dot Diagram Chunk With Variable Insertions

## Usage

```
makegraphviz(.object., ..., file)
```

## Arguments

|                       |  |
|-----------------------|--|
| <code>.object.</code> | character string or vector with graphviz markup  |
| <code>...</code>      | name=value pairs that makes values replace <code>{{name}}</code> elements in the markup  |
| <code>file</code>     | name of file to hold graphviz markup after variable insertions. Run this in Quarto using a chunk to looks like the following, which was for <code>file='graphviz.dot'</code> . |

```

```{dot}
//| label: fig-doverview-graphviz
//| fig-height: 4
//| fig-cap: "Consort diagram produced with `graphviz` with detailed exclusion frequenc
//| file: graphviz.dot
```

```

## Details

Takes a character string or vector and uses `knitr::knit_expand()` to apply variable insertions before the diagram is rendered by Quarto. See [this](#) for an example. Unlike mermaid, graphviz can include user-defined linkages to specific parts of a node (e.g., a single word in a line of text) to another part of the chart, and can render tables. If an inclusion is `...` is a data frame or table, it will be properly rendered inside the diagram.

## Value

nothing; used to `knitr::knit_expand()` graphviz markup

## Author(s)

Frank Harrell

## See Also

[makemermaid\(\)](#), [makegvflow\(\)](#), [makedot\(\)](#)

---

 makegvflow

*Create a Quarto Graphviz Flow Diagram from Plain Text With Variable Insertions*


---

## Description

Takes a character string or vector and creates a graphviz flowchart from the hierarchy indicated by the number of indented spaces in each line. The text for the root levels is not indented, and major, minor, and tiny levels are indented 2, 4, or 6 spaces, respectively. Default color, font size, and fill color are chosen so that the chart is suitable for presentations. Simple  $\LaTeX$  math markup is converted to simple HTML markup using `tth::tth()`. Text lines are automatically wrapped to keep text boxes from being too wide. Text lines beginning with "+" are combined with the previous major, minor, or tiny text line but separated by a double line break (single break if `lbdouble=FALSE`).

## Usage

```
makegvflow(
  .object.,
  ...,
  direction = c("TD", "LR"),
  style = "filled",
  shape = "box",
  font = "Times-Roman",
  fontsize = 18,
  fontcolor = "blue",
  fillcolor = "azure",
  penwidth = 0.1,
  arrowcolor = "blue3",
  arrowsize = 0.7,
  width = 30,
  lbdouble = TRUE,
  extracon = NULL,
  file,
  onlyprint = FALSE
)
```

## Arguments

|                        |  |
|------------------------|--|
| <code>.object.</code>  | character string or vector of plain text plus possible $\LaTeX$ math delimited by single dollar signs. An empty initial line is ignore, so the user need not worry about having an initial quote mark on a line by itself. |
| <code>...</code>       | name=value pairs that makes values replace <code>{{name}}</code> elements in the markup  |
| <code>direction</code> | direction of the flow chart, default is top-down   |
| <code>style</code>     | node style   |
| <code>shape</code>     | node shape   |
| <code>font</code>      | font for text in nodes   |

|            |  |
|------------|--|
| fontsize   | text font size   |
| fontcolor  | text color   |
| fillcolor  | node fill color  |
| penwidth   | thickness of lines for node borders  |
| arrowcolor | arrow color  |
| arrowsize  | arrow size   |
| width      | text width for word-wrapping   |
| lbdouble   | set to FALSE to use a single line break for "+" lines  |
| extracon   | one or more text strings specifying extra connections between nodes using node names nijk for major level i, minor level j, tiny level k (as many of these that are applicable). For example specify extracon=c('n1 -> n2', 'n21 -> n31'). |
| file       | name of file to hold graphviz markup after variable insertions. Run this in Quarto using a chunk to looks like what is below, which was for file='graphviz.dot'.   |
| onlyprint  | set to TRUE to only print the generated graphviz markup and not save it to file  |

```

```{dot}
//| label: fig-flow1
//| fig-height: 4
//| fig-cap: "Chart caption"
//| file: graphviz.dot
```

```

### Details

The function uses `knitr::knit_expand()` to apply variable insertions before the diagram is rendered by Quarto. See [this](#) for an example. `##' @title makegvflow`

### Value

nothing; used to `knitr::knit_expand()` graphviz markup

### Author(s)

Frank Harrell

### See Also

[makegraphviz\(\)](#)

### Examples

```

x <- '
Root text
  Major 1
    Minor 11 {{{}}}
    Minor 12
  Major 2

```

```

Minor 21
Minor 22
Minor 23 that is very very long and just keeps going
  tiny 231 and  $\alpha + \sum_{i=1}^n X_i$ 
  tiny 232
  + a second line for tiny 232 that is pretty long
  + a third line for tiny 232
Major 3
  Minor 31
    tiny 311'
makegvflow(x, extracon='n12 -> n21', jj='tiger', onlyprint=TRUE)

```

---

makemermaid

*makemermaid*


---

## Description

Create a Quarto Mermaid Diagram Chunk With Variable Insertions

## Usage

```
makemermaid(.object., ..., file)
```

## Arguments

`.object.` character string or vector with mermaid markup  
`...` name=value pairs that makes values replace `{{name}}` elements in the markup  
`file` name of file to hold mermaid markup after variable insertions. Run this in Quarto using a chunk to looks like the following, which was for `file='mermaid1.mer'`.

```

```{mermaid}
%%| fig-cap: "Consort diagram produced by `mermaid`"
%%| label: fig-mermaid1
%%| file: mermaid1.mer
```

```

## Details

Takes a character string or vector and uses `knitr::knit_expand()` to apply variable insertions before the diagram is rendered by Quarto. See [this](#) for an example.

## Value

nothing; used to `knitr::knit_expand()` mermaid markup

## Author(s)

Frank Harrell

**See Also**

[makegraphviz\(\)](#)

---

maketabs

*maketabs*

---

**Description**

Make Quarto Tabs

**Usage**

```
maketabs(
  ...,
  wide = FALSE,
  cwidth = if (wide) "column-page",
  initblank = FALSE,
  baselabel = NULL,
  cap = NULL,
  basecap = NULL,
  debug = FALSE
)
```

**Arguments**

|           |   |
|-----------|---|
| ...       | a series of formulas or a single named list. For formulas the left side is the tab label (if multiple words or other illegal R expressions enclose in backticks) and the right hand side has expressions to evaluate during chunk execution, plus optional raw, caption, and fig.size options.  |
| wide      | set to TRUE to use a Quarto column-page for the body of the text to allow it to use some of the margins   |
| cwidth    | specify a legal Quarto character string instead of wide to specify the width of the output. These are defined <a href="#">here</a> . Commonly used values are 'column-screen-right', 'column-page-left', 'column-screen-inset-shaded'.  |
| initblank | set to TRUE to create a first tab that is blank so that the report will not initially show any tabbed material  |
| baselabel | a one-word character string that provides the base name of labels for tabs with figure captions. The sequential tab number is appended to baselabel to obtain the full figure label. If using formulas the figure label may instead come from caption(.., label). If not specified it is taken to be the name of the current chunk with fig- prepended. |
| cap       | applies to the non-formula use of maketabs and is an integer vector specifying which tabs are to be given figure labels and captions.   |
| basecap   | a single character string providing the base text for captions if cap is specified.   |
| debug     | set to TRUE to output debugging information in file /tmp/z  |

**Details**

Loops through a series of formulas or elements of a named list and outputs each element into a separate Quarto tab. `wide` and `column` arguments are used to expand the width of the output outside the usual margins. An `initblank` argument creates a first tab that is empty, or you can specify a formula `~` . This allows one to show nothing until one of the other tabs is clicked. Multiple commands can be run in one chunk by including multiple right hand terms in a formula. A chunk can be marked for producing raw output by including a term `raw` somewhere in the formula's right side. It can be marked for constructing a label and caption by including `+ caption(caption string, label string)`. The tab number is appended to the label string, and if the label is not provided `baselabel` will be used.

**Value**

nothing is returned; used to render markup

**Author(s)**

Frank Harrell

**Examples**

```
X <- list(A=data.frame(x=1:2), B=data.frame(x=1:2, y=11:12))
maketabs(X)
```

---

missChk

*missChk*

---

**Description**

Produce a Report Section Detailing Missing Values in a Dataset

**Usage**

```
missChk(
  data,
  use = NULL,
  exclude = NULL,
  type = c("report", "seq"),
  maxpat = 15,
  maxcomb = 25,
  excl1pat = TRUE,
  sortpatterns = TRUE,
  prednmiss = FALSE,
  omitpred = NULL,
  baselabel = NULL,
  ...
)
```

**Arguments**

|              |   |
|--------------|---|
| data         | data frame or table to analyze  |
| use          | a formula or character vector specifying which variables to consider if not all those in data   |
| exclude      | a formula or character vector specifying which variables to exclude from consideration  |
| type         | specify 'seq' to return a summary of sequential exclusions rather than produce a report   |
| maxpat       | maximum number of missing data patterns allowed when counting occurrences of all combinations of variables' NAs                                     |
| maxcomb      | maximum number of combinations for which to produce a combination dot plot  |
| excl1pat     | set to FALSE to not list distinct combinations that only occur for one observation  |
| sortpatterns | set to FALSE to not sort patterns in decreasing frequency of missingness  |
| prednmiss    | set to TRUE to use ordinal regression to predict the number of missing variables on an observation from the values of all the non-missing variables |
| omitpred     | a formula or character vector specifying a list of predictors not to use when predicting number of missing variables                                |
| baselabel    | base label for Quarto tabs made with <code>maketabs()</code>  |
| ...          | passed to <code>Hmisc::combplotp()</code>   |

**Details**

Quantifies frequencies of missing observations on a variable and missing variables on an observaton and produces variables tables and (depending on the number of NAs) multiple graphic displays in Quarto tabs. The results are best understood by referring to [this](#).

**Value**

nothing; outputs Quarto/RMarkdown text and tabs for a full report section

**Author(s)**

Frank Harrell

**Examples**

```
## Not run:
missChk(mydata)

## End(Not run)
```

---

multDataOverview      *multDataOverview*

---

**Description**

Multiple Dataset Overview

**Usage**

```
multDataOverview(X, id = NULL)
```

**Arguments**

|    |  |
|----|--|
| X  | list object containing any number of data frames/tables                            |
| id | formula containing a single subject identifier, e.g., <code>id = patient.id</code> |

**Details**

Provides an overview of the data tables inside a giant list. The result returned (invisible) is a data table containing for each variable a comma-separated list of datasets containing that variable (other than id variables).

**Value**

invisibly, a data table

**Author(s)**

Frank Harrell

**See Also**

[dataOverview\(\)](#)

**Examples**

```
## Not run:  
multDataOverview(list(data1=mydata1, data2=mydata2), id = ~ subject.id)  
  
## End(Not run)
```

---

pdumpit

*Print to File for Debugging*

---

### Description

If options(dumpfile="...") is set, uses Hmisc::prn() to print objects for debugging

### Usage

```
pdumpit(x, txt = as.character(substitute(x)))
```

### Arguments

|     |  |
|-----|--|
| x   | input to prn                               |
| txt | text label, defaults to name of x argument |

### Value

no result, used only for printing debugging information

### Author(s)

Frank Harrell

---

printCap

*printCap*

---

### Description

Pretty Printing of Captions Dataset

### Usage

```
printCap(book = FALSE)
```

### Arguments

|      |   |
|------|---|
| book | set to TRUE to not use format='html' when running kable |
|------|---|

### Details

Uses kable to print the caption information saved in .captions..

### Value

kable object

**Author(s)**

Frank Harrell

**Examples**

```
## Not run:  
princCap()  
  
## End(Not run)
```

---

|                      |                |
|----------------------|----------------|
| <code>putQcap</code> | <i>putQcap</i> |
|----------------------|----------------|

---

**Description**

Create Quarto Figure Caption

**Usage**

```
putQcap(..., scap = NULL, label = NULL)
```

**Arguments**

- `...` one or more character strings to form the caption
- `scap` a character string subcaption
- `label` figure label

**Details**

Creates a Quarto label and caption and uses `addCap()` to add to running list of figures

**Value**

string vector with YAML components `label`, `fig-cap`, `fig-scap`

**Author(s)**

Frank Harrell

**Examples**

```
putQcap('First part of caption', 'second part', scap='subcaption', label='xx')
```

---

|               |                      |
|---------------|----------------------|
| runDeriveExpr | <i>runDeriveExpr</i> |
|---------------|----------------------|

---

## Description

Apply Derived Variable Specifications

## Usage

```
runDeriveExpr(d, derv, pr = TRUE)
```

## Arguments

|      |   |
|------|---|
| d    | a data table                                  |
| derv | a list of expressions to evaluate             |
| pr   | set pr=FALSE to suppress information messages |

## Details

Function to apply derived variable specifications derv to a data table d. Actions on d are done in place, so call the function using `runDeriveExpr(d, derv object)` and not by running `d <- runDeriveExpr(d, derv object)`. See [this](#) for an example.

## Value

nothing; used to print information and add variables to data table

## Author(s)

Frank Harrell

## Examples

```
require(data.table)
d <- data.table(ht=c(68, 60), wt=c(280, 135), chol=c(120, 150))
derived <- list(
  list(bmi = expression(703 * wt / ht ^ 2),
       label='Body Mass Index',
       units='Kg/m^2'),
  list(bsa=expression(0.007184 * (0.4536 * wt) ^ 0.425 * (2.54 * ht) ^ 0.725),
       label='Body Surface Area',
       units='m^2', drop=.q(wt, ht) ) )
runDeriveExpr(d, derived)
print(d)
contents(d)
```

---

`rwrap`*rwrap*

---

**Description**

Protecting Backticks for Illustrating In-line R Code

**Usage**

```
rwrap(x)
```

**Arguments**

`x` a character string

**Details**

This function pastes back ticks around a string so those extra back ticks don't have to appear in the user's code in a report. This prevents Quarto from intervening.

**Value**

`x` surrounded by backtick `r` and backtick

**Author(s)**

Frank Harrell

**Examples**

```
rwrap('pi')
```

---

`sampleFrac`*Compute Sample Fractions*

---

**Description**

Uses denominators stored with `setqreportOption` along with counts specified to `SampleFrac` to compute fractions of subjects in current analysis

**Usage**

```
sampleFrac(n, nobSY = NULL, table = TRUE, study = " ")
```

**Arguments**

|       |  |
|-------|--|
| n     | integer vector, named with "enrolled", "randomized" and optionally also including treatment levels.  |
| nobsY | a result of the the nobsY Hmisc function   |
| table | set to TRUE to return as an attribute "table" a character string containing an HTML table showing the pertinent frequencies created from n and the denom option, and if nobsY is present, adding another table with response variable-specific counts. |
| study | character string with study ID   |

**Value**

named vector of relative sample sizes with an attribute table with frequency counts

**Examples**

```
setqreportOption(tx.var='treatment', denom=c(enrolled=1000, randomized=800, a=398, b=402))
sampleFrac(getqreportOption('denom'))
```

---

saveCap

*saveCap*

---

**Description**

Save Caption Dataset in External File

**Usage**

```
saveCap(basename)
```

**Arguments**

basename      base file name to which -captions.rds will be appended

**Details**

Uses `base::saveRDS()` to save the .captions. dataset to a user file.

**Value**

nothing; used to create a saved RDS dataset of caption information

**Author(s)**

Frank Harrell

**Examples**

```
## Not run:
saveCap('chapter3')

## End(Not run)
```

---

|       |              |
|-------|--------------|
| splot | <i>splot</i> |
|-------|--------------|

---

**Description**

Separate Chunk Plot

**Usage**

```
splot(command, cap = NULL, scap = NULL, w = 5, h = 4, id = NULL)
```

**Arguments**

|         |   |
|---------|---|
| command | an command that causes a plot to be rendered                                      |
| cap     | long caption  |
| scap    | short caption   |
| w       | width of plot   |
| h       | height of plot  |
| id      | a string ID for the plot. Defaults to the current chunk label if knitr is running |

**Details**

Runs a plot on its own Rmarkdown/Quarto knitr Chunk. The plot will have its own caption and size, and short captions are placed in the markdown TOC

Expressions cannot be re-used, i.e., each expression must evaluate to the right quantity after the chunk in which the splot calls are made is finished, and the new constructed chunk is input. To input and run the constructed chunk: `{r child='splot.Rmd'}` preceded and following by 3 back ticks. `Hmisc::putHcap` is used to markup regular and short captions `cap`, `scap`. Short caption appears in TOC. If no `scap`, then `cap` is used for this. To change the `putHcap` subsub argument set `options(splot.subsub='## ')` for example.

**Value**

no value return; outputs R Markdown/Quarto markup

**Author(s)**

Frank Harrell

**Examples**

```
## Not run:
splot(id='chunkid') # initialize output file splot.Rmd
# or use splot() to use the current chunk name as the id
# splot(plotting expression, caption, optional short caption, w, h)
# splot(plotting expression ...)

## End(Not run)
```

---

setqreportOption      *Set qreport Options*

---

**Description**

Set qreport options, assigning certain defaults

**Usage**

```
setqreportOption(..., study = " ")
```

**Arguments**

|       |   |
|-------|---|
| ...   | a series of options for which non-default values are desired: <ul style="list-style-type: none"> <li>• tx.pch: symbols corresponding to treatments</li> <li>• tx.col: colors corresponding to treatments</li> <li>• tx.linecol: colors for lines in line plots</li> <li>• nontx.col: colors for categories other than treatments</li> <li>• tx.lty: line types corresponding to treatments</li> <li>• tx.lwd: line widths corresponding to treatments</li> <li>• tx.var: character string name of treatment variable</li> <li>• er.col: 2-vector with names "enrolled", "randomized" containing colors to use for enrolled and randomized in needle displays</li> <li>• alpha.f: single numeric specifying alpha adjustment to be applied to all colors. Default is 1 (no adjustment)</li> <li>• denom: named vector with overall sample sizes See <a href="https://github.com/plotly/plotly.py/blob/master/plotly.R">https://github.com/plotly/plotly.py/blob/master/plotly.R</a></li> </ul> |
| study | an optional study mnemonic (character string) needed when multiple studies are being analyzed (or when one study is divided into distinct strata)   |

**Value**

no returned value, used to set options()

**Examples**

```
setqreportOption(tx.var='treatment', denom=c(enrolled=1000, randomized=800, a=398, b=402))
```



|          |   |
|----------|---|
| ps       | see par   |
| mfrow    | see par   |
| axes     | see par   |
| cex.lab  | see par   |
| cex.axis | see par   |
| ...      | other parameters passed as-is to <code>graphics::par()</code> |

### Details

This function tries to set `graphics::par()` to make base graphics look more publication-ready.

### Value

nothing; side effect of setting `par()`

### Author(s)

Frank Harrell

### Examples

```
## Not run:
spar(top=2, bot=1) # leave extra space for titles

## End(Not run)
```

---

timeMar

*timeMar*

---

### Description

Time an Expression and Report in Quarto Margin

### Usage

```
timeMar(x)
```

### Arguments

x                    an expression to execute

### Details

Function to time an expression, printing the result of `base::system.time()` in the right margin, and storing the result of `system.time` in `.systime.` in the global environment so tha the user can refer to it.

**Value**

invisibly, the result of the expression

**Author(s)**

Frank Harrell

**See Also**

[hooktime\(\)](#)

**Examples**

```
## Not run:
g <- function(...){} # define a function to run slowly
result <- timeMar(g())

## End(Not run)
```

---

varType

*varType*


---

**Description**

Determine Types of Variables

**Usage**

```
varType(data, include = NULL, exclude = NULL, ndistinct = 10, nnonnum = 20)
```

**Arguments**

|           |  |
|-----------|--|
| data      | data frame or table to analyze   |
| include   | formula or vector of variable names to attend to   |
| exclude   | a formula or character vector specifying which variables to exclude from consideration     |
| ndistinct | minimum number of distinct numeric values a variable must have to be considered continuous |
| nnonnum   | maximum number of distinct values a variable can have to be considered discrete            |

**Details**

For all the variables in a data frame/table, analyzes them to determine types: continuous, nonnumeric, and discrete. include and exclude can be vector or right-side-only formulas.

**Value**

list of vectors

**Author(s)**

Frank Harrell

**Examples**

```

set.seed(1)
d <- data.frame(i=1:100, x=runif(100), y=sample(1:3, 100, TRUE),
               w=sample(c('cat','dog','giraffe'), 100, TRUE),
               v=sample(letters, 100, TRUE))

varType(d)

```

---

vClus

*cClus*


---

**Description**

Make Variable Clustering Quarto Report Section

**Usage**

```

vClus(
  d,
  exclude = NULL,
  corrmatrix = FALSE,
  redundancy = FALSE,
  spc = FALSE,
  trans = FALSE,
  rexclude = NULL,
  fracmiss = 0.2,
  maxlevels = 10,
  minprev = 0.05,
  imputed = NULL,
  horiz = FALSE,
  label = "fig-varclus",
  print = TRUE,
  redunargs = NULL,
  spcargs = NULL,
  transaceargs = NULL,
  transacefile = NULL,
  spcfile = NULL
)

```

**Arguments**

|         |  |
|---------|--|
| d       | a data frame or table  |
| exclude | formula or vector of character strings containing variables to exclude from analysis |

|              |  |
|--------------|--|
| corrmatrix   | set to TRUE to use <code>Hmisc::plotCorrM()</code> to depict a Spearman rank correlation matrix.   |
| redundancy   | set to TRUE to run <code>Hmisc::redun()</code> on non-excluded variables   |
| spc          | set to TRUE to run <code>Hmisc::princmp()</code> to do a sparse principal component analysis with the argument <code>method='sparse'</code> passed   |
| trans        | set to TRUE to run <code>Hmisc::transace()</code> to transform each predictor before running redundancy or principal components analysis. <code>transace</code> is run on the stacked filled-in data if <code>imputed</code> is given.   |
| rexclude     | extra variables to exclude from <code>transace</code> transforming-finding, redundancy analysis, and sparse principal components (formula or character vector)   |
| fracmiss     | if the fraction of NAs for a variable exceeds this the variable will not be included   |
| maxlevels    | if the maximum number of distinct values for a categorical variable exceeds this, the variable will be dropped   |
| minprev      | the minimum proportion of non-missing observations in a category for a binary variable to be retained, and the minimum relative frequency of a category before it will be combined with other small categories   |
| imputed      | an object created by <code>Hmisc::aregImpute()</code> or <code>mice::mice()</code> that contains information from multiple imputation that causes <code>vClus</code> to create all the filled-in datasets, stack them into one tall dataset, and pass that dataset to <code>Hmisc::redun()</code> or <code>Hmisc::princmp()</code> so that NAs can be handled efficiently in redundancy analysis and sparse principal components, i.e., without excluding partial records. Variable clustering and the correlation matrix are already efficient because they use pairwise deletion of NAs. |
| horiz        | set to TRUE to draw the dendrogram horizontally  |
| label        | figure label for Quarto  |
| print        | set to FALSE to not let <code>dataframeReduce</code> report details  |
| redunargs    | a <code>list()</code> of other arguments passed to <code>Hmisc::redun()</code>   |
| spcargs      | a <code>list()</code> of other arguments passed to <code>Hmisc::princmp()</code>   |
| transaceargs | a <code>list()</code> of other arguments passed to <code>Hmisc::transace()</code>  |
| transacefile | similar to <code>spcfile</code> and can be used when <code>trans=TRUE</code>   |
| spcfile      | a character string specifying an <code>.rds</code> R binary file to hold the results of sparse principal component analysis. Using <code>Hmisc::runifChanged()</code> , if the file name is specified and no inputs have changed since the last run, the result is read from the file. Otherwise a new run is made and the file is recreated if <code>spcfile</code> is specified. This is done because sparse principal components can take several minutes to run on large files.  |

## Details

Draws a variable clustering dendrogram and optionally graphically depicts a correlation matrix. See [this](#) for an example. Uses `Hmisc::varclus()`.

**Value**

makes Quarto tabs and prints output, returning nothing unless `spc=TRUE` or `trans=TRUE` are used, in which case a list with components `princmp` and/or `transace` is returned and these components can be passed to special print and plot methods for `spc` or to `ggplot_transace`. The user can put scree plots and PC loading plots in separate code chunks that use different figure sizes that way.

**Author(s)**

Frank Harrell

**See Also**

[Hmisc::varclus\(\)](#), [Hmisc::plotCorrM\(\)](#), [Hmisc::dataframeReduce\(\)](#), [Hmisc::redun\(\)](#), [Hmisc::princmp\(\)](#), [Hmisc::transace\(\)](#)

**Examples**

```
## Not run:  
vClus(mydata, exclude=.q(country, city))  
  
## End(Not run)
```

# Index

addCap, 2  
aePlot, 3  
asForm, 5  
  
base::saveRDS(), 34  
base::system.time(), 38  
  
conVars, 6  
  
dataChk, 7  
dataOverview, 8  
dataOverview(), 29  
disVars, 9  
dNeedle, 10  
  
getqreportOption, 10  
graphics::par(), 38  
  
Hmisc::aregImpute(), 41  
Hmisc::combplotp(), 28  
Hmisc::dataframeReduce(), 42  
Hmisc::plotCorrM(), 41, 42  
Hmisc::princmp(), 41, 42  
Hmisc::redun(), 41, 42  
Hmisc::runifChanged(), 41  
Hmisc::transace(), 41, 42  
Hmisc::varclus(), 41, 42  
hookaddcap, 11  
hooktime, 12  
hooktime(), 39  
htmlList, 12  
htmlView, 13  
htmlViewx, 14  
  
kabl, 15  
knitr::knit\_expand(), 21, 22, 24, 25  
  
makecallout, 16  
makecnote, 17  
makecodechunk, 18  
makecolmarg, 19  
  
makedot, 20  
makedot(), 22  
makegraphviz, 22  
makegraphviz(), 21, 24, 26  
makegvflow, 23  
makegvflow(), 20–22  
makemermaid, 25  
makemermaid(), 22  
maketabs, 26  
maketabs(), 28  
mice::mice(), 41  
missChk, 27  
multDataOverview, 29  
  
pdumpit, 30  
printCap, 30  
putQcap, 31  
  
runDeriveExpr, 32  
rwrap, 33  
  
sampleFrac, 33  
saveCap, 34  
scplot, 35  
setqreportOption, 4, 36  
spar, 37  
  
timeMar, 38  
timeMar(), 12  
tth::tth(), 23  
  
varType, 39  
vClus, 40