

# Package: SciViews (via r-universe)

September 1, 2024

**Type** Package

**Version** 1.8.0

**Title** 'SciViews' - Data Processing and Visualization with the  
'SciViews::R' Dialect

**Description** The 'SciViews::R' dialect provides a set of functions that streamlines data input, process, analysis and visualization especially, but not exclusively, for beginners or occasional users. It mixes base R and tidyverse, plus another set of CRAN packages for an easy and coherent use of R.

**Maintainer** Philippe Grosjean <phgrosjean@sciviews.org>

**Depends** R (>= 4.2.0)

**Imports** cli (>= 3.6.1), crayon (>= 1.5.2), ellipse (>= 0.4.5),  
graphics (>= 4.2.0), grDevices (>= 4.2.0), purrr (>= 1.0.1),  
rstudioapi (>= 0.14), stats (>= 4.2.0), svBase (>= 1.4.0),  
tabularise (>= 0.5.0), utils (>= 4.2.0), httr2 (>= 1.0.1),  
roxygen2 (>= 7.3.1)

**Suggests** broom (>= 1.0.4), chart (>= 1.5.0), collapse (>= 2.0.12),  
data.io (>= 1.5.0), data.table (>= 1.15.4), dbplyr (>= 2.3.2),  
dplyr (>= 1.1.4), dtplyr (>= 1.3.1), forcats (>= 1.0.0), fs (>= 1.6.1), ggplot2 (>= 3.4.2), googledrive (>= 2.1.0),  
googlesheets4 (>= 1.1.0), haven (>= 2.5.2), hms (>= 1.1.3),  
httr (>= 1.4.5), jsonlite (>= 1.8.4), lubridate (>= 1.9.2),  
magrittr (>= 2.0.3), MASS (>= 7.3.58.3), modelr (>= 0.1.11),  
pillar (>= 1.9.0), readr (>= 2.1.4), readxl (>= 1.4.2), reprex  
(>= 2.0.2), rlang (>= 1.1.1), rvest (>= 1.0.3), stringr (>= 1.5.0), svFlow (>= 1.2.0), svMisc (>= 1.4.0), tibble (>= 3.2.1), tidyr (>= 1.3.0), tidyverse (>= 2.0.0), xml2 (>= 1.3.3), knitr (>= 1.42), rmarkdown (>= 2.21), spelling (>= 2.2.1), testthat (>= 3.0.0)

**Remotes** SciViews/chart, SciViews/data.io, SciViews/svBase,  
SciViews/svFlow, SciViews/svMisc, SciViews/tabularise

**Enhances** base

**License** GPL-2

**URL** <https://github.com/SciViews/SciViews>,  
<https://www.sciviews.org/SciViews/>

**BugReports** <https://github.com/SciViews/SciViews/issues>

**Roxygen** list(markdown = TRUE)

**RxygenNote** 7.3.1

**VignetteBuilder** knitr

**Encoding** UTF-8

**Language** en-US

**ByteCompile** yes

**Config/testthat/edition** 3

**Repository** <https://sciviews.r-universe.dev>

**RemoteUrl** <https://github.com/SciViews/SciViews>

**RemoteRef** HEAD

**RemoteSha** 41b9bfa1d9656ae525a9da3d714fe4ace5d99078

## Contents

SciViews-package . . . . .	3
ai_ask . . . . .	4
chatbot1 . . . . .	7
chatbot2 . . . . .	7
chatbot3 . . . . .	7
colors . . . . .	8
correlation . . . . .	9
enum . . . . .	12
ln . . . . .	13
nr . . . . .	14
panels . . . . .	15
panels.diag . . . . .	19
pcomp . . . . .	23
SciViews_packages . . . . .	28
SciViews_R . . . . .	29
timing . . . . .	30
vectorplot . . . . .	31

<b>Index</b>	<b>33</b>
--------------	-----------

---

SciViews-package	<i>'SciViews::R' Dialect for Data Processing and Visualization</i>
------------------	--

---

## Description

The SciViews::R dialect is base R + tidyverse + a series of additional SciViews packages like data.io, svBase, svFlow, tabularise or chart.

## Important functions

- `R()` for loading the ‘SciViews::R’ packages,
- `pcomp()` for a PCA analysis (unifying various methods),
- `correlation()` to calculate and plot a correlation matrix,
- `panel_reg()` and others to plot panels in pairs or coplot graphs,
- `panel_boxplot()` and others for univariate panels in pairs plots.
- `rwb_colors()` and others to generate color palettes.
- `enum()` to enumerate items in a vector,
- `timing()` to determine the time required to run an R expression,
- `nr()` and `co` as convenient shorthand to columns and rows,
- `ln()` and others for natural logarithm.

## Author(s)

**Maintainer:** Philippe Grosjean <phgrosjean@sciviews.org> ([ORCID](#))

## See Also

Useful links:

- <https://github.com/SciViews/SciViews>
- <https://www.sciviews.org/SciViews/>
- Report bugs at <https://github.com/SciViews/SciViews/issues>

ai\_ask

*Ask a question to a chatbot***Description**

The question is sent to the chatbot server, and the answer is returned. The default chatbot server is ollama running locally, and the default model is codestral:latest. It was also tested with mistral:7b-instruct-v0.2-q6\_K for an even smaller model.

**Usage**

```
ai_ask(
    question,
    context = NULL,
    max_tokens = getOption("SciViews.chatbot.max_tokens",
        Sys.getenv("SCIVIEWS_CHATBOT_MAX_TOKENS", 1000L)),
    lang = getOption("data.io_lang", "en"),
    url = getOption("SciViews.chatbot.url", Sys.getenv("SCIVIEWS_CHATBOT_URL",
        "http://localhost:11434/api/chat")),
    model = getOption("SciViews.chatbot.model", Sys.getenv("SCIVIEWS_CHATBOT_MODEL",
        "codestral:latest")),
    api_key = Sys.getenv("CONNECT_API_KEY", ""),
    verbose = FALSE
)

ai_explain_term(term, lang = getOption("data.io_lang", "en"), ...)

ai_explain_function(
    fun,
    package = NULL,
    lang = getOption("data.io_lang", "en"),
    ...
)

ai_explain_code(code, lang = getOption("data.io_lang", "en"), ...)

ai_explain_error(
    code = NULL,
    error = NULL,
    lang = getOption("data.io_lang", "en"),
    ...
)
```

**Arguments**

question            A character string with the question to ask.

context	An R object used as context (usually a data frame). This is not used yet, but it should be implemented in the future.
max_tokens	The maximum number of tokens to return in the answer. By default, it is 1000.
lang	The language to use for the answer. Default is "en". You can also use "fr" for instance.
url	The URL of the chatbot server. Default is <code>http://localhost:11434/api/chat</code>
model	The LLM (large language model) to use. Default is <code>codestral</code> . Make sure you complies to its license (see <a href="https://mistral.ai/news/mistral-ai-non-production-license-mnpl/">https://mistral.ai/news/mistral-ai-non-production-license-mnpl/</a> ), or switch to another model that better suits your requirements.
api_key	The API key to use for connecting to the chatbot server (optional, see your server administrator).
verbose	Should more information be printed? FALSE by default.
term	The term to describe.
...	Further arguments passed to <code>ai_ask()</code> .
fun	The R function to explain.
package	The R package that provides the function.
code	A small chunk of R code to explain.
error	The error message that R returns.

## Value

The answer is returned invisibly. The function is used for its side-effect of displaying the chatbot help page with the question, answer and examples

## Examples

```
## Not run:

# Basic questions
ai_ask("Who are you?")
ai_ask("What is a chatbot?")
ai_ask("Qui es-tu ?")
ai_ask("Qu'est-ce que R ?")
ai_ask("Qu'est-ce que RStudio ?")
ai_ask("What is GitHub?")
ai_ask("Qu'est-ce que le R Markdown ?")
ai_ask("What is data science?")

# Inappropriate questions
ai_ask("Qu'est ce qu'un Acanthurus sp ?")
ai_ask("Raconte-moi une bonne blague.")
ai_ask("Va te faire voir !")

# Now, more complex questions
ai_ask("Comment filtrer un data frame en R?")
ai_ask("Write R code to filter a data frame.")
```

```

ai_ask("Que fait AIC()? Donne un exemple.")
ai_ask("Qu'est ce que l'hétéroscédasticité et comment la détecter dans une ANOVA à un facteur ?")
ai_ask("How to determine which model is better using an ANOVA for nested linear models?")

# Explain terms
ai_explain_term("True positive")
ai_explain_term("percentile", lang = "fr")
ai_explain_term("git push")
ai_explain_term("Quarto", lang = "fr")
ai_explain_term("boite à moustaches") # Language mismatch
ai_explain_term("boîtes à moustaches parallèles", lang = "fr")

# Explain R functions
ai_explain_function("mean")
ai_explain_function("fmean", lang = "fr")
ai_explain_function("collapse::fmean", lang = "fr")
ai_explain_function("glm", package = "stats", lang = "fr")
ai_explain_function("replace_na", "tidyr", lang = "fr") # collapse::replace_na() used instead!
try(ai_explain_function("nonexistingfunction")) # Error
try(ai_explain_function("apropos", package = "stats")) # Wrong package
try(ai_explain_function("apropos", package = "unknownpkg")) # Unknown package

# Explain R code
ai_explain_code("y <- c(1, 5, 7, NA, -Inf, 8)")
ai_explain_code(r"-[
mtcars |>
  filter(cyl == 4) |>
  summarise(mean_hp = mean(hp), median_disp = median(displacement))]-")
ai_explain_code(r"-[
mtcars %>%
  sfilter(., cyl == 4) %>%
  ssummarise(., mean_hp = fmean(hp), median_disp = fmedian(displacement))
]-")
ai_explain_code(r"-[
  chart(data = trees, Volume ~ Girth) +
    geom_point() +
    geom_smooth()
]-")
ai_explain_error(error = "longer object length is not a multiple of shorter object length")
ai_explain_error(error = "Error: object 'mydata' not found")
ai_explain_error(error = "Error in lenght(1:10) : could not find function \"lenght\"", lang = "fr")
ai_explain_error(code = r"-[y <- c(1, 5, 7, NA, -Inf, 8, )]-", lang = "fr")
ai_explain_error(code = r"-[trees %>% filter(Girth > 10)]-", lang = "fr")
ai_explain_error(code = r"-[
urchin <- read("urchin", package = "data.io")
]-", error = r"-[
Error in read("urchin", package = "data.io") :
  dataset 'urchin' not found in package 'data.io'
]-", lang = "fr")

## End(Not run)

```

---

chatbot1SciViews chatbot

---

**Description***A chatbot does not always provide reliable results. Take this with a grain of salt!*

Question: No question

**Answer**

No answer

**Examples**# No example

---

---

chatbot2SciViews chatbot

---

**Description***A chatbot does not always provide reliable results. Take this with a grain of salt!*

Question: No question

**Answer**

No answer

**Examples**# No example

---

---

chatbot3SciViews chatbot

---

**Description***A chatbot does not always provide reliable results. Take this with a grain of salt!*

Question: No question

**Answer**

No answer

**Examples**

# No example

---

colors	<i>Various color palettes</i>
--------	-------------------------------

---

**Description**

Create vectors of n colors.

**Usage**

```
rwb_colors(n, alpha = 1, s = 0.9, v = 0.9)
```

```
rwb.colors(n, alpha = 1, s = 0.9, v = 0.9)
```

```
rwg_colors(n, alpha = 1, s = 0.9, v = 0.9)
```

```
rwg.colors(n, alpha = 1, s = 0.9, v = 0.9)
```

```
ryg_colors(n, alpha = 1, s = 0.9, v = 0.9)
```

```
ryg.colors(n, alpha = 1, s = 0.9, v = 0.9)
```

```
cwm_colors(n, alpha = 1, s = 0.9, v = 0.9)
```

```
cwm.colors(n, alpha = 1, s = 0.9, v = 0.9)
```

**Arguments**

- |       |  |
|-------|--|
| n     | The number of colors ( $\geq 1$ ) to be in the palette.                                      |
| alpha | The alpha transparency, a number in $[0, 1]$ , see argument alpha in <a href="#">hsv()</a> . |
| s     | The 'saturation' to be used to complete the HSV color descriptions.                          |
| v     | The 'value' to use for the HSV color descriptions.   |

**Details**

`cwm_colors(s = 0.5, v = 1)` gives very similar colors to `cm.colors()`. `ryg_colors()` is similar to `rainbow(start = 0, end = 2/6)`. The `xxx_colors()` (tidyverse name-compatible) and `xxx.colors()` (grDevices name-compatible) functions are synonyms.

**See Also**

[cm.colors\(\)](#), [colorRampPalette\(\)](#)



**Examples**

```
# Draw color wheels with various palettes
opar <- par(mfrow = c(2, 2))
pie(rep(1, 11), col = cwm.colors(11), main = "Cyan - white - magenta")
pie(rep(1, 11), col = rwg.colors(11), main = "Red - white - blue")
pie(rep(1, 11), col = rwg.colors(11), main = "Red - white - green")
pie(rep(1, 11), col = ryg.colors(11), main = "Red - yellow - green")
par(opar)
```

correlation

*Correlation matrices***Description**

Compute the correlation matrix between all columns of a matrix or data frame.

**Usage**

```
correlation(x, ...)
```

```
Correlation(x, ...)
```

```
## S3 method for class 'formula'
correlation(formula, data = NULL, subset, na.action, ...)
```

```
## Default S3 method:
correlation(
  x,
  y = NULL,
  use = "everything",
  method = c("pearson", "kendall", "spearman"),
  ...
)
```

```
is.Correlation(x)
```

```
is.correlation(x)
```

```
as.Correlation(x)
```

```
as.correlation(x)
```

```
## S3 method for class 'Correlation'
print(x, digits = 3, cutoff = 0, ...)
```

```
## S3 method for class 'Correlation'
summary(
```

```

    object,
    cutpoints = c(0.3, 0.6, 0.8, 0.9, 0.95),
    symbols = c(" ", ".", ",", "+", "*", "B"),
    ...
)

## S3 method for class 'summary.Correlation'
print(x, ...)

## S3 method for class 'Correlation'
plot(
  x,
  y = NULL,
  outline = TRUE,
  cutpoints = c(0.3, 0.6, 0.8, 0.9, 0.95),
  palette = rwb.colors,
  col = NULL,
  numbers = TRUE,
  digits = 2,
  type = c("full", "lower", "upper"),
  diag = (type == "full"),
  cex.lab = par("cex.lab"),
  cex = 0.75 * par("cex"),
  ...
)

## S3 method for class 'Correlation'
lines(
  x,
  choices = 1L:2L,
  col = par("col"),
  lty = 2,
  ar.length = 0.1,
  pos = NULL,
  cex = par("cex"),
  labels = rownames(x),
  ...
)

```

### Arguments

<code>x</code>	A numeric vector, matrix or data frame (or any object for <code>is.Correlation()</code> or <code>as.Correlation()</code> ).
<code>...</code>	Further arguments passed to functions.
<code>formula</code>	A formula with no response variable, referring only to numeric variables.
<code>data</code>	An optional data frame (or similar, see <code>model.frame()</code> ) containing the variables in the formula. By default the variables are taken from <code>environment(formula)</code> .
<code>subset</code>	An optional vector used to select rows (observations) of the data matrix <code>x</code> .

<code>na.action</code>	A function which indicates what should happen when the data contain NAs. The default is set by the <code>na.action</code> setting of <code>options()</code> and <code>na.fail()</code> is used if that is not set. The 'factory-fresh' default is <code>na.omit()</code> .
<code>y</code>	NULL (default), or a vector, matrix or data frame with compatible dimensions to <code>x</code> for <code>Correlation()</code> . The default is equivalent to <code>x = y</code> , but more efficient.
<code>use</code>	An optional character string giving a method for computing correlations in the presence of missing values. This must be (an abbreviation of) one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs".
<code>method</code>	A character string indicating which correlation coefficient is to be computed. One of "pearson" (default), "kendall", or "spearman", can be abbreviated.
<code>digits</code>	Digits to print after the decimal separator.
<code>cutoff</code>	Correlation coefficients lower than this (in absolute value) are suppressed.
<code>object</code>	A 'Correlation' object.
<code>cutpoints</code>	The cut points to use for categories. Specify only positive values (absolute value of correlation coefficients are summarized, or negative equivalents are automatically computed for the graph. Do not include 0 or 1 in the cutpoints).
<code>symbols</code>	The symbols to use to summarize the correlation matrix.
<code>outline</code>	Do we draw the outline of the ellipse?
<code>palette</code>	A function that can produce a palette of colors.
<code>col</code>	Color of the ellipse. If NULL (default), the colors will be computed using <code>cutpoints</code> and <code>palette</code> .
<code>numbers</code>	Do we print correlation values in the center of the ellipses?
<code>type</code>	Do we plot a complete matrix, or only lower or upper triangle?
<code>diag</code>	Do we plot items on the diagonal? They have always a correlation of one.
<code>cex.lab</code>	The expansion factor for labels.
<code>cex</code>	The expansion factor for text.
<code>choices</code>	The items to select.
<code>lty</code>	The line type to draw.
<code>ar.length</code>	The length of the arrow head.
<code>pos</code>	The position relative to arrows.
<code>labels</code>	The label to draw near the arrows.

## Value

`Correlation()` and `as.Correlation()` create a 'Correlation' object, while `is.Correlation()` tests for it.

There are `print()` and `summary()` methods for the 'Correlation' object that differ in the symbolic encoding of the correlations, (using `symnum()` for `summary()`), which makes large correlation matrices more readable.

The `plot()` method draws ellipses on a graph to represent the correlation matrix visually. This is essentially the `plotcorr()` function from package **ellipse**, with slightly different default arguments and with default cutpoints equivalent to those used in the `summary()` method.

**Author(s)**

Philippe Grosjean [phgrosjean@sciviews.org](mailto:phgrosjean@sciviews.org), wrapping code in package **ellipse**, function `plotcorr()` for the `plot.Correlation()` method.

**See Also**

`cov()`, `cov2cor()`, `cov.wt()`, `symnum()`, `plotcorr()` and look also at `panel_cor()`

**Examples**

```
# This is a simple correlation coefficient
cor(rnorm(10), runif(10))
Correlation(rnorm(10), runif(10))

# 'Correlation' objects allow better inspection of the correlation matrices
# than the output of default R cor() function
(longley.cor <- Correlation(longley))
summary(longley.cor) # Synthetic view of the correlation matrix
plot(longley.cor)    # Graphical representation

# Use of the formula interface
(mtcars.cor <- Correlation(~ mpg + cyl + disp + hp, data = mtcars,
  method = "spearman", na.action = "na.omit"))

mtcars.cor2 <- Correlation(mtcars, method = "spearman")
print(mtcars.cor2, cutoff = 0.6)
summary(mtcars.cor2)
plot(mtcars.cor2, type = "lower")

mtcars.cor2["mpg", "cyl"] # Extract a correlation from the correlation matrix
```

---

enum

---

*Enumerate items in an object*


---

**Description**

`enum()` creates a vector of integers from 1 to length of the object (it enumerates items in the object), except if the object is empty. It is particularly useful in the `for(i in enum(object))` construct.

**Usage**

```
enum(x)
```

**Arguments**

x                      Any object.

**Note**

The pattern `for(i in 1:length(object))` is often found, but it fails in case `length(object) == 0`! `enum()` is indeed a synonym of `seq_along()`, but the later one is less expressive in the context.

**See Also**

[seq\\_along\(\)](#)

**Examples**

```
enum(letters)
enum(numeric(0))
# Compare with:
1:length(numeric(0))
enum(NULL)
letters5 <- letters[1:5]
for (i in enum(letters5)) cat("letter", i, "=", letters5[i], "\n")
```

---

ln

---

*Logarithmic and exponential functions*


---

**Description**

`ln()` computes natural logarithm, `lg()` computes base 10 logarithm, and `lb()` computes binary (base 2) logarithm.

`ln1p()` and `lg1p()` computes  $\ln(x + 1)$  and  $\lg(x + 1)$  accurately also for  $|x| \ll 1$ .

`E` is the Euler constant and is equal to `exp(1)`.

**Usage**

`ln(x)`

`lg(x)`

`lb(x)`

`ln1p(x)`

`lg1p(x)`

`E`

**Arguments**

`x`                      A numeric or complex vector.

**Format**

An object of class `numeric` of length 1.

**Details**

Those functions are synonyms of `log()`, `log10()`, `log2()`, `log1p()` for those who prefer the shorter notation. Beginners sometimes make confusion between `log()` and `log10()`. Using `ln()` for natural logarithms instead of `log()` eliminates this confusion. `E` is provided for convenience as `exp(1)`, although the use of `exp()` is usually familiar enough to everyone.

**See Also**

[log\(\)](#)

**Examples**

```
ln(exp(3))          # Same as log(exp(3))
lg(10^3)            # Same as log10(10^3)
lb(1:3)             # Wrapper for log2()

ln1p(c(0, 1, 10, 100)) # Wrapper for log1p()
lg1p(c(0, 1, 10, 100)) # log10(x + 1), but optimized for x << 1

E^4                 # Similar to exp(4), but different calculation!
```

---

nr

*Convenience functions for rows or columns manipulations*


---

**Description**

`nr()` and `nc()` are synonyms of the ugly `NROW()` or `NCOL()` that get the number of row and columns in a matrix or data frame, but also in a vector (they return a value even if the `dim` attribute of the object is not set, on the contrary to `nrow()` or `ncol()`).

`ROWS` and `COLS` are constants that makes call to `apply()` more expressive. `ROWS = 1L` and `COLS = 2L`.

**Usage**

```
nr(x)
```

```
nc(x)
```

```
ROWS
```

```
COLS
```

**Arguments**

```
x          Any object.
```

**Format**

An object of class integer of length 1.

An object of class integer of length 1.

**See Also**

[nrow\(\)](#)

**Examples**

```
mm <- matrix(1:6, nrow = 3)
nr(mm)
nc(mm)

vv <- 1:6
nr(vv)
nc(vv)

# ROWS and COLS constants used with apply()
apply(mm, ROWS, mean) # Idem apply(mm, 1, mean)
apply(mm, COLS, mean) # Idem apply(mm, 2, mean)
```

---

panels

*More panel plots*

---

**Description**

Several panel plots that can be used with [coplot\(\)](#) and [pairs\(\)](#).

**Usage**

```
panel_reg(
  x,
  y,
  col = par("col"),
  bg = par("bg"),
  pch = par("pch"),
  cex = par("cex"),
  lwd = par("lwd"),
  line.reg = lm,
  line.col = "red",
  line.lwd = lwd,
  untf = TRUE,
  ...
)

panel.reg(
```

```
x,  
y,  
col = par("col"),  
bg = par("bg"),  
pch = par("pch"),  
cex = par("cex"),  
lwd = par("lwd"),  
line.reg = lm,  
line.col = "red",  
line.lwd = lwd,  
untf = TRUE,  
...  
)  
  
panel_ellipse(  
  x,  
  y,  
  col = par("col"),  
  bg = par("bg"),  
  pch = par("pch"),  
  cex = par("cex"),  
  el.level = 0.7,  
  el.col = "cornsilk",  
  el.border = "red",  
  major = TRUE,  
  ...  
)  
  
panel.ellipse(  
  x,  
  y,  
  col = par("col"),  
  bg = par("bg"),  
  pch = par("pch"),  
  cex = par("cex"),  
  el.level = 0.7,  
  el.col = "cornsilk",  
  el.border = "red",  
  major = TRUE,  
  ...  
)  
  
panel_cor(  
  x,  
  y,  
  use = "everything",  
  method = c("pearson", "kendall", "spearman"),  
  alternative = c("two.sided", "less", "greater"),
```



```
    digits = 2,
    prefix = "",
    cex = par("cex"),
    cor.cex = cex,
    stars.col = "red",
    ...
)

panel.cor(
  x,
  y,
  use = "everything",
  method = c("pearson", "kendall", "spearman"),
  alternative = c("two.sided", "less", "greater"),
  digits = 2,
  prefix = "",
  cex = par("cex"),
  cor.cex = cex,
  stars.col = "red",
  ...
)

panel_smooth(
  x,
  y,
  col = par("col"),
  bg = NA,
  pch = par("pch"),
  cex = 1,
  col.smooth = 2,
  span = 2/3,
  iter = 3,
  ...
)
```

### Arguments

x	A numeric vector.
y	A numeric vector of same length as x.
col	The color of the points.
bg	The background color for symbol used for the points.
pch	The symbol used for the points.
cex	The expansion factor used for the points.
lwd	The line width.
line.reg	A function that calculates coefficients of a straight line, for instance, <code>lm()</code> , or <code>r1m()</code> for robust linear regression.

<code>line.col</code>	The color of the line.
<code>line.lwd</code>	The width of the line.
<code>untf</code>	Logical asking whether to untransform the straight line in case one or both axis are in log scale.
<code>...</code>	Further arguments to plot functions.
<code>el.level</code>	The confidence level for the bivariate normal ellipse around data; the default value of 0.7 draws an ellipse of roughly $\pm 1$ sd.
<code>el.col</code>	The color used to fill the ellipse.
<code>el.border</code>	The color used to draw the border of the ellipse and the standardized major axis.
<code>major</code>	If TRUE, the standardized major axis is also drawn.
<code>use</code>	One of "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs" (can be abbreviated). Defines how the <code>cor()</code> function behaves with missing observations.
<code>method</code>	One of the three correlation coefficients "pearson" (default), "kendall", or "spearman". Can be abbreviated.
<code>alternative</code>	The alternative hypothesis in correlation test, see <code>cor.test()</code> .
<code>digits</code>	The number of decimal digits to print when the correlation coefficient is printed in the graph.
<code>prefix</code>	A prefix (character string) to use before the correlation coefficient printed in the graph.
<code>cor.cex</code>	Expansion coefficient for text in printing correlation coefficients.
<code>stars.col</code>	The color used for significance stars (with: *** $p < 0.001$ , ** $p < 0.1$ , * $p < 0.05$ , . $p < 0.1$ ).
<code>col.smooth</code>	Color to be used by lines for drawing the smooths.
<code>span</code>	Smoothing parameter $f$ for <code>lowess()</code> , see there.
<code>iter</code>	Number of robustness iterations for <code>lowess()</code> .

### Details

Theses functions should be used outside of the diagonal in `pairs()`, or with `coplot()`, as they are bivariate plots.

### Value

These functions return nothing and are used for their side effect of plotting in panels of composite plots.

### Author(s)

Philippe Grosjean [phgrosjean@sciviews.org](mailto:phgrosjean@sciviews.org), but code inspired from `panel.smooth()` in **graphics** and `panel.car()` in package **car**.

### See Also

`coplot()`, `pairs()`, `panel.smooth()`, `lm()`, `ellipse()`, `cor()` and `cor.test()`

## Examples

```
# Smooth lines in lower graphs and straight lines in upper graphs
pairs(trees, lower.panel = panel_smooth, upper.panel = panel_reg)
# Robust regression lines
library(MASS) # For rlm()
pairs(trees, panel = panel_reg, diag.panel = panel_boxplot,
      reg.line = rlm, line.col = "blue", line.lwd = 2)
# A Double log graph
pairs(trees, lower.panel = panel_smooth, upper.panel = panel_reg, log = "xy")

# Graph suitables to explore correlations (take care there are potentially
# many simultaneous tests done here... So, you loose much power in the whole
# analysis... use it just as an indication!)
# Pearson's r
pairs(trees, lower.panel = panel_ellipse, upper.panel = panel_cor)
# Spearman's rho (ellipse and straight lines not suitable here!)
pairs(trees, lower.panel = panel_smooth, upper.panel = panel_cor,
      method = "spearman", span = 1)
# Several groups (visualize how bad it is to consider the whole set at once!)
pairs(iris[, -5], lower.panel = panel_smooth, upper.panel = panel_cor,
      method = "kendall", span = 1,
      col = c("red3", "blue3", "green3")[iris$Species])
# Now analyze correlation for one species only
pairs(iris[iris$Species == "virginica", -5], lower.panel = panel_ellipse,
      upper.panel = panel_cor)

# A coplot with custom panes
coplot(Petal.Length ~ Sepal.Length | Species, data = iris,
       panel = panel_ellipse)
```

---

panels.diag

*More univariate panel plots*

---

## Description

Several panel plots that can be used with `pairs()`.

## Usage

```
panel_boxplot(x, col = par("col"), box.col = "cornsilk", ...)
```

```
panel.boxplot(x, col = par("col"), box.col = "cornsilk", ...)
```

```
panel_density(
  x,
  adjust = 1,
  rug = TRUE,
  col = par("col"),
  lwd = par("lwd"),
```

```
    line.col = col,
    line.lwd = lwd,
    ...
)

panel.density(
  x,
  adjust = 1,
  rug = TRUE,
  col = par("col"),
  lwd = par("lwd"),
  line.col = col,
  line.lwd = lwd,
  ...
)

panel_hist(
  x,
  breaks = "Sturges",
  hist.col = "cornsilk",
  hist.border = NULL,
  hist.density = NULL,
  hist.angle = 45,
  ...
)

panel.hist(
  x,
  breaks = "Sturges",
  hist.col = "cornsilk",
  hist.border = NULL,
  hist.density = NULL,
  hist.angle = 45,
  ...
)

panel_qqnorm(
  x,
  pch = par("pch"),
  col = par("col"),
  bg = par("bg"),
  cex = par("cex"),
  lwd = par("lwd"),
  qq.pch = pch,
  qq.col = col,
  qq.bg = bg,
  qq.cex = cex,
  qqline.col = qq.col,
```

```

    qqline.lwd = lwd,
    ...
)

panel.qqnorm(
  x,
  pch = par("pch"),
  col = par("col"),
  bg = par("bg"),
  cex = par("cex"),
  lwd = par("lwd"),
  qq.pch = pch,
  qq.col = col,
  qq.bg = bg,
  qq.cex = cex,
  qqline.col = qq.col,
  qqline.lwd = lwd,
  ...
)

```

### Arguments

<code>x</code>	A numeric vector.
<code>col</code>	The color of the points.
<code>box.col</code>	The filling color of the boxplots.
<code>...</code>	Further arguments to plot functions, or functions that construct items, like <code>density()</code> , depending on the context.
<code>adjust</code>	The bandwidth adjustment factor, see <code>density()</code> .
<code>rug</code>	Do we add a rug representation (1-d plot) of the points too?
<code>lwd</code>	The line width.
<code>line.col</code>	The color of the line.
<code>line.lwd</code>	The width of the line.
<code>breaks</code>	The number of breaks, the name of a break algorithm, a vector of breakpoints, or any other acceptable value for breaks argument of <code>hist()</code> .
<code>hist.col</code>	The filling color for the histograms.
<code>hist.border</code>	The border color for the histograms.
<code>hist.density</code>	The density for filling lines in the histograms.
<code>hist.angle</code>	The angle for filling lines in the histograms.
<code>pch</code>	The symbol used for the points.
<code>bg</code>	The background color for symbol used for the points.
<code>cex</code>	The expansion factor used for the points.
<code>qq.pch</code>	The symbol used to plot points in the QQ-plots.
<code>qq.col</code>	The color of the symbol used to plot points in the QQ-plots.

qq.bg	The background color of the symbol used to plot points in the QQ-plots.
qq.cex	The expansion factor for points in the QQ-plots.
qqline.col	The color for the QQ-plot lines.
qqline.lwd	The width for the QQ-plot lines.

## Details

Panel functions `panel_boxplot()`, `panel_density()`, `panel_hist()` and `panel_qqnorm()` should be used only to plot univariate data on the diagonals of `pairs()` plots (or scatterplot matrix).

## Value

These functions return nothing and are used for their side effect of plotting in panels of composite plots.

## Author(s)

Philippe Grosjean [phgrosjean@sciviews.org](mailto:phgrosjean@sciviews.org), but code inspired from `spm()` in package **car**.

## See Also

`pairs()`, `boxplot()`, `hist()`, `density()`, `qqnorm()`

## Examples

```
# Example of scatterplot matrices with custom plots on the diagonal

# Boxplots
pairs(trees, panel = panel_smooth, diag.panel = panel_boxplot)
pairs(trees, diag.panel = panel_boxplot, box.col = "gray")

# Densities
pairs(trees, panel = panel_smooth, diag.panel = panel_density)
pairs(trees, diag.panel = panel_density, line.col = "red", adjust = 0.5)

# Histograms
pairs(trees, panel = panel_smooth, diag.panel = panel_hist)
pairs(trees, diag.panel = panel_hist, hist.col = "gray", breaks = "Scott")

# QQ-plots against Normal theoretical distribution
pairs(trees, panel = panel_smooth, diag.panel = panel_qqnorm)
pairs(trees, diag.panel = panel_qqnorm, qqline.col = 2, qq.cex = .5, qq.pch = 3)
```

pcomp

*Principal Components Analysis***Description**

Perform a principal components analysis (PCA) on a matrix or data frame and return a pcomp object.

**Usage**

```
pcomp(x, ...)
```

## S3 method for class 'formula'

```
pcomp(formula, data = NULL, subset, na.action, method = c("svd", "eigen"), ...)
```

## Default S3 method:

```
pcomp(
  x,
  method = c("svd", "eigen"),
  scores = TRUE,
  center = TRUE,
  scale = TRUE,
  tol = NULL,
  covmat = NULL,
  subset = rep(TRUE, nrow(as.matrix(x))),
  ...
)
```

## S3 method for class 'pcomp'

```
print(x, ...)
```

## S3 method for class 'pcomp'

```
summary(object, loadings = TRUE, cutoff = 0.1, ...)
```

## S3 method for class 'summary.pcomp'

```
print(x, digits = 3, loadings = x$print.loadings, cutoff = x$cutoff, ...)
```

## S3 method for class 'pcomp'

```
plot(
  x,
  which = c("screeplot", "loadings", "correlations", "scores"),
  choices = 1L:2L,
  col = par("col"),
  bar.col = "gray",
  circle.col = "gray",
  ar.length = 0.1,
  pos = NULL,
  labels = NULL,
```

```

    cex = par("cex"),
    main = paste(deparse(substitute(x)), which, sep = " - "),
    xlab,
    ylab,
    ...
)

```

```

## S3 method for class 'pcomp'
screepplot(
  x,
  npcs = min(10, length(x$sdev)),
  type = c("barplot", "lines"),
  col = "cornsilk",
  main = deparse(substitute(x)),
  ...
)

```

```

## S3 method for class 'pcomp'
points(
  x,
  choices = 1L:2L,
  type = "p",
  pch = par("pch"),
  col = par("col"),
  bg = par("bg"),
  cex = par("cex"),
  ...
)

```

```

## S3 method for class 'pcomp'
lines(
  x,
  choices = 1L:2L,
  groups,
  type = c("p", "e"),
  col = par("col"),
  border = par("fg"),
  level = 0.9,
  ...
)

```

```

## S3 method for class 'pcomp'
text(
  x,
  choices = 1L:2L,
  labels = NULL,
  col = par("col"),
  cex = par("cex"),

```



```

    pos = NULL,
    ...
)

## S3 method for class 'pcomp'
biplot(x, choices = 1L:2L, scale = 1, pc.biplot = FALSE, ...)

## S3 method for class 'pcomp'
pairs(
  x,
  choices = 1L:3L,
  type = c("loadings", "correlations"),
  col = par("col"),
  circle.col = "gray",
  ar.col = par("col"),
  ar.length = 0.05,
  pos = NULL,
  ar.cex = par("cex"),
  cex = par("cex"),
  ...
)

## S3 method for class 'pcomp'
predict(object, newdata, dim = length(object$sdev), ...)

## S3 method for class 'pcomp'
correlation(x, newvars, dim = length(x$sdev), ...)

scores(x, ...)

## S3 method for class 'pcomp'
scores(x, labels = NULL, dim = length(x$sdev), ...)

```

## Arguments

x	A matrix or data frame with numeric data.
...	Arguments passed to or from other methods. If x is a formula one might specify scale, tol or covmat.
formula	A formula with no response variable, referring only to numeric variables.
data	An optional data frame (or similar, see <code>model.frame()</code> ) containing the variables in the formula. By default the variables are taken from <code>environment(formula)</code> .
subset	An optional vector used to select rows (observations) of the data matrix x.
na.action	A function which indicates what should happen when the data contain NAs. The default is set by the <code>na.action</code> setting of <code>options()</code> , and is <code>na.fail()</code> if that is not set. The 'factory-fresh' default is <code>na.omit()</code> .
method	Either "svd" (using <code>prcomp()</code> ), "eigen" (using <code>princomp()</code> ), or an abbreviation.

scores	A logical value indicating whether the score on each principal component should be calculated.
center	A logical value indicating whether the variables should centered. Alternately, a vector of length equal the number of columns of <code>x</code> can be supplied. The value is passed to <code>scale</code> . Note that this argument is ignored for <code>method = "eigen"</code> and the dataset is always centered in this case.
scale	A logical value indicating whether the variables should be scaled to have unit variance before the analysis takes place. The default is <code>TRUE</code> , which in general, is advisable. Alternatively, a vector of length equal the number of columns of <code>x</code> can be supplied. The value is passed to <code>scale()</code> .
tol	Only when <code>method = "svd"</code> . A value indicating the magnitude below which components should be omitted. (Components are omitted if their standard deviations are less than or equal to <code>tol</code> times the standard deviation of the first component.) With the default null setting, no components are omitted. Other settings for <code>tol</code> could be <code>tol = 0</code> or <code>tol = sqrt(.Machine\$double.eps)</code> , which would omit essentially constant components.
covmat	A covariance matrix, or a covariance list as returned by <code>cov.wt()</code> (and <code>cov.mve()</code> or <code>cov.mcd()</code> from package <b>MASS</b> ). If supplied, this is used rather than the covariance matrix of <code>x</code> .
object	A 'pcomp' object.
loadings	Do we also summarize the loadings?
cutoff	The cutoff value below which loadings are replaced by white spaces in the table. That way, larger values are easier to spot and to read in large tables.
digits	The number of digits to print.
which	The graph to plot.
choices	Which principal axes to plot. For 2D graphs, specify two integers.
col	The color to use in graphs.
bar.col	The color of bars in the screeplot.
circle.col	The color for the circle in the loadings or correlations plots.
ar.length	The length of the arrows in the loadings and correlations plots.
pos	The position of text relative to arrows in loadings and correlation plots.
labels	The labels to write. If <code>NULL</code> default values are computed.
cex	The factor of expansion for text (labels) in the graphs.
main	The title of the graph.
xlab	The label of the x-axis.
ylab	The label of the y-axis.
npcs	The number of principal components to represent in the screeplot.
type	The type of screeplot (" <code>barplot</code> " or " <code>lines</code> ") or pairs plot (" <code>loadings</code> " or " <code>correlations</code> ").
pch	The type of symbol to use.
bg	The background color for symbols.

groups	A grouping factor.
border	The color of the border.
level	The probability level to use to draw the ellipse.
pc.biplot	Do we create a Gabriel's biplot (see <a href="#">biplot()</a> )?
ar.col	Color of arrows.
ar.cex	Expansion factor for text on arrows.
newdata	New individuals with observations for the same variables as those used for calculating the PCA. You can then plot these additional individuals in the scores plot.
dim	The number of principal components to keep.
newvars	New variables with observations for same individuals as those used for calculating the PCA. Correlation with PCs is calculated. You can then plot these additional variables in the correlation plot.

### Details

`pcomp()` is a generic function with "formula" and "default" methods. It is essentially a wrapper around [prcomp\(\)](#) and [princomp\(\)](#) to provide a coherent interface and object for both methods.

A 'pcomp' object is created. It inherits from 'pca' (as in **labdsv** package, but not compatible with the version of 'pca' in **ade4**) and of 'princomp'.

For more information on algorithms, refer to [prcomp\(\)](#) for `method = "svd"` or [princomp\(\)](#) for `method = "eigen"`.

### Value

A `c("pcomp", "pca", "princomp")` object.

### Note

The signs of the columns for the loadings and scores are arbitrary. So, they could differ between functions for PCA, and even between different builds of R.

### Author(s)

Philippe Grosjean [phgrosjean@sciviews.org](mailto:phgrosjean@sciviews.org), but the core code is indeed in package **stats**.

### See Also

[prcomp\(\)](#), [princomp\(\)](#), [loadings\(\)](#), [vectorplot\(\)](#), [Correlation\(\)](#)

### Examples

```
# Let's analyze mtcars without the Mercedes data (rows 8:14)
data(mtcars)
cars.pca <- pcomp(~ mpg + cyl + disp + hp + drat + wt + qsec,
  data = mtcars, subset = -(8:14))
cars.pca
```

```
summary(cars.pca)
screeplot(cars.pca)

# Loadings are extracted and plotted this way:
(cars.ldg <- loadings(cars.pca))
plot(cars.pca, which = "loadings") # Equivalent to vectorplot(cars.ldg)

# Similarly, correlations of variables with PCs are extracted and plotted:
(cars.cor <- Correlation(cars.pca))
plot(cars.pca, which = "correlations") # Equivalent to vectorplot(cars.cor)
# One can add supplementary variables on this graph
lines(Correlation(cars.pca,
  newvars = mtcars[-(8:14), c("vs", "am", "gear", "carb")]))

# Plot the scores:
plot(cars.pca, which = "scores", cex = 0.8) # Similar to plot(scores(x)[, 1:2])
# Add supplementary individuals to this plot (labels), also points() or lines()
text(predict(cars.pca, newdata = mtcars[8:14, ]),
  labels = rownames(mtcars[8:14, ]), col = "gray", cex = 0.8)

# Pairs plot for 3 PCs
iris.pca <- pcomp(iris[, -5])
pairs(iris.pca, col = (2:4)[iris$Species])
```

---

SciViews\_packages

Give the list of SciViews::R packages and check for conflicts

---

## Description

List required packages or conflicting functions. These functions are inspired by `tidyverse::tidyverse_packages()` and `tidyverse::tidyverse_conflicts()`, but adapted to the SciViews::R context.

## Usage

```
SciViews_packages(..., all = FALSE)

SciViews_packages_topics(all = FALSE)

SciViews_conflicts(all = TRUE)

## S3 method for class 'SciViews_conflicts'
print(x, ..., startup = FALSE)
```

## Arguments

<code>...</code>	Further topics to consider in SciViews::R. Currently, "infer", "model", "explore", "ml", "ts" or "spatial".
<code>all</code>	Should all packages be listed (TRUE) or only those that are attached to the search path (FALSE).

x	A SciViews_conflicts object
startup	Should the message be printed at startup?

**Value**

A list of packages for `SciViews_packages()`, or a **SciViews\_conflicted** object with a `print()` method for `SciViews_conflicts()`.

**Examples**

```
# List of packages attached to the search path with SciViews::R
SciViews_packages()
# More complete list of packages used by SciViews::R
SciViews_packages(all = TRUE)
# Even more packages, by adding also 'model' and 'ml' topics
SciViews_packages("model", "ml", all = TRUE)
# Conflicts
SciViews_conflicts()
```

---

SciViews\_R

---

*Configure R for the SciViews::R dialect*


---

**Description**

Load required packages like `data.table`, `collapse`, `ggplot2`, `dplyr`, `svMisc`, ... to get a fully functional `SciViews::R` dialect environment.

**Usage**

```
R(..., lang = NULL, dtx = NULL, threads.percent = 75, silent = TRUE)
```

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

**Arguments**

...	Further topics to include to configure R (load more packages). Currently, "infer", "model", "explore", "ml", "ts" or "spatial"
lang	What is the default natural language to use, e.g., "en" or "fr", with uppercase versions "EN" or "FR" convert even more strings, for instance, <code>data.io::read()</code> does not convert factor levels in the corresponding language for supported data sets unless the uppercase version is specified. If NULL (by default), current configuration is not changed.
dtx	Which dtx object is to be used by default? "dtt" or "data.table" for <code>data.table</code> , "dtf" or "data.frame" for <code>data.frame</code> , "dtbl", "tibble" or "tbl_df" for <code>tibble</code> 's <code>tbl_df</code> , the name of a function to use to convert a <code>data.frame</code> object, or NULL (by default) to keep current settings.

<code>threads.percent</code>	The percentage of threads to use for <code>{data.table}</code> and <code>{collapse}</code> parallel code (number of threads depend on how many are available, and the value is rounded towards zero).
<code>silent</code>	If TRUE (by default), no report is printed about loaded packages and conflicts.
<code>x</code>	An object to print.

**Note**

Use `SciViews::R` instruction in the beginning of an R script, or in the setup or first chunk of an R Markdown/Notebook to ensure the `SciViews::R` dialect is correctly installed. The report indicating attached packages and conflicts is largely inspired by the corresponding tidyverse code, written by Hadley Wickham.

**See Also**

`library()`, `utils::install.packages()`

**Examples**

```
## Not run:
SciViews::R

## End(Not run)
```

---

timing	<i>Timing of R expressions</i>
--------	--------------------------------

---

**Description**

Similar to `system.time()` but returns a more convenient 'difftime' object with the overall timing (details are stored in the `details` attribute).

**Usage**

```
timing(expr, gc.first = TRUE)
```

**Arguments**

<code>expr</code>	Valid R expression to be timed. If missing, <code>proc.time()</code> is used instead and the function returns the time the currently running R process has already taken.
<code>gc.first</code>	Logical - should a garbage collection be performed immediately before the timing? Default is TRUE.

**See Also**

`system.time()`, `proc.time()`

**Examples**

```
test <- timing(Sys.sleep(0.5))
test
attr(test, "details")
```

---

vectorplot	<i>Plot vectors inside a unit circle (PCA loadings or correlations plots).</i>
------------	--

---

**Description**

Plots vectors with  $0 < \text{norms} < 1$  inside a circle. These plots are mainly designed to represent variables in principal components space for PCAs.

**Usage**

```
vectorplot(x, ...)
```

## Default S3 method:

```
vectorplot(
  x,
  y,
  col = par("col"),
  circle.col = "gray",
  ar.length = 0.1,
  pos = NULL,
  cex = par("cex"),
  labels = NULL,
  ...
)
```

## S3 method for class 'loadings'

```
vectorplot(
  x,
  choices = 1L:2L,
  col = par("col"),
  circle.col = "gray",
  ar.length = 0.1,
  pos = NULL,
  cex = par("cex"),
  labels = rownames(x),
  main = deparse(substitute(x)),
  ...
)
```

## S3 method for class 'Correlation'

```
vectorplot(
  x,
```

```

choices = 1L:2L,
col = par("col"),
circle.col = "gray",
ar.length = 0.1,
pos = NULL,
cex = par("cex"),
labels = rownames(x),
main = deparse(substitute(x)),
...
)

```

### Arguments

<code>x</code>	An object that has a <code>vectorplot()</code> method, like 'loadings' or 'correlation', or a numeric vector with $0 < \text{values} < 1$ .
<code>...</code>	Further arguments passed to plot functions.
<code>y</code>	A numeric vector with $0 < \text{values} < 1$ of same length as 'x'.
<code>col</code>	Color of the arrows and labels.
<code>circle.col</code>	The color for the circle around the vector plot.
<code>ar.length</code>	The length of the arrows.
<code>pos</code>	The position of text relative to arrows. If NULL, a suitable position is calculated according to the direction where the arrows are pointing.
<code>cex</code>	The factor of expansion for labels in the graph.
<code>labels</code>	The labels to draw near the arrows.
<code>choices</code>	A vector of two integers indicating the axes to plot.
<code>main</code>	The title of the plot.

### Value

The object 'x' is returned invisibly. These functions are called for their side-effect of drawing a vector plot.

### See Also

`pcomp()`, `loadings()`, `Correlation()`

### Examples

```

# Create a PCA and plot loadings and correlations
iris.pca <- pcomp(iris[, -5])
vectorplot(loadings(iris.pca))
vectorplot(Correlation(iris.pca))
# Note: on screen devices, change aspect ratio of the graph by resizing
# the window to reveal cropped labels...

```



# Index

- \* **Vector and circular plot**
    - vectorplot, 31
  - \* **aplot**
    - panels, 15
    - panels.diag, 19
    - vectorplot, 31
  - \* **color palettes**
    - colors, 8
  - \* **color**
    - colors, 8
  - \* **correlation matrix and plot**
    - correlation, 9
  - \* **datasets**
    - ln, 13
    - nr, 14
  - \* **distribution**
    - correlation, 9
  - \* **logarithm and exponential**
    - ln, 13
  - \* **math**
    - ln, 13
  - \* **models**
    - pcomp, 23
  - \* **packages loading**
    - SciViews\_R, 29
  - \* **panel plots**
    - panels, 15
    - panels.diag, 19
  - \* **principal component analysis and biplot**
    - pcomp, 23
  - \* **utilities**
    - SciViews\_R, 29
- ai\_ask, 4  
ai\_ask(), 5  
ai\_explain\_code(ai\_ask), 4  
ai\_explain\_error(ai\_ask), 4  
ai\_explain\_function(ai\_ask), 4  
ai\_explain\_term(ai\_ask), 4  
as.Correlation(correlation), 9  
as.correlation(correlation), 9  
biplot(), 27  
biplot.pcomp(pcomp), 23  
boxplot(), 22  
chatbot1, 7  
chatbot2, 7  
chatbot3, 7  
cm.colors(), 8  
colorRampPalette(), 8  
colors, 8  
COLS(nr), 14  
coplot(), 15, 18  
cor(), 18  
cor.test(), 18  
Correlation(correlation), 9  
correlation, 9  
Correlation(), 27, 32  
correlation(), 3  
correlation.pcomp(pcomp), 23  
cov(), 12  
cov.mcd(), 26  
cov.mve(), 26  
cov.wt(), 12, 26  
cov2cor(), 12  
cwm.colors(colors), 8  
cwm\_colors(colors), 8  
data.io::read(), 29  
density(), 21, 22  
E(ln), 13  
ellipse(), 18  
enum, 12  
enum(), 3  
hist(), 21, 22  
hsv(), 8  
is.Correlation(correlation), 9

is.correlation(correlation), 9  
 lb(ln), 13  
 lg(ln), 13  
 lg1p(ln), 13  
 library(), 30  
 lines.Correlation(correlation), 9  
 lines.pcomp(pcomp), 23  
 lm(), 17, 18  
 ln, 13  
 ln(), 3  
 ln1p(ln), 13  
 loadings(), 27, 32  
 log(), 14  
 lowess(), 18  
  
 model.frame(), 10, 25  
  
 na.fail(), 11, 25  
 na.omit(), 11, 25  
 nc(nr), 14  
 nr, 14  
 nr(), 3  
 nrow(), 15  
  
 options(), 25  
  
 pairs(), 15, 18, 19, 22  
 pairs.pcomp(pcomp), 23  
 panel.boxplot(pannels.diag), 19  
 panel.cor(pannels), 15  
 panel.density(pannels.diag), 19  
 panel.ellipse(pannels), 15  
 panel.hist(pannels.diag), 19  
 panel.qqnorm(pannels.diag), 19  
 panel.reg(pannels), 15  
 panel.smooth(), 18  
 panel\_boxplot(pannels.diag), 19  
 panel\_boxplot(), 3, 22  
 panel\_cor(pannels), 15  
 panel\_cor(), 12  
 panel\_density(pannels.diag), 19  
 panel\_density(), 22  
 panel\_ellipse(pannels), 15  
 panel\_hist(pannels.diag), 19  
 panel\_hist(), 22  
 panel\_qqnorm(pannels.diag), 19  
 panel\_qqnorm(), 22  
 panel\_reg(pannels), 15  
  
 panel\_reg(), 3  
 panel\_smooth(pannels), 15  
 pannels, 15  
 pannels.diag, 19  
 pcomp, 23  
 pcomp(), 3, 32  
 plot.Correlation(correlation), 9  
 plot.pcomp(pcomp), 23  
 plotcorr(), 11, 12  
 points.pcomp(pcomp), 23  
 prcomp(), 25, 27  
 predict.pcomp(pcomp), 23  
 princomp(), 25, 27  
 print(), 29  
 print.Correlation(correlation), 9  
 print.pcomp(pcomp), 23  
 print.SciViews\_conflicts  
     (SciViews\_packages), 28  
 print.SciViews\_R(SciViews\_R), 29  
 print.summary.Correlation  
     (correlation), 9  
 print.summary.pcomp(pcomp), 23  
 proc.time(), 30  
  
 qqnorm(), 22  
  
 R(SciViews\_R), 29  
 R(), 3  
 rlm(), 17  
 ROWS(nr), 14  
 rwb.colors(colors), 8  
 rwb\_colors(colors), 8  
 rwb\_colors(), 3  
 rwg.colors(colors), 8  
 rwg\_colors(colors), 8  
 ryg.colors(colors), 8  
 ryg\_colors(colors), 8  
  
 scale(), 26  
 SciViews(SciViews-package), 3  
 SciViews-package, 3  
 SciViews\_conflicts(SciViews\_packages),  
     28  
 SciViews\_conflicts(), 29  
 SciViews\_packages, 28  
 SciViews\_packages(), 29  
 SciViews\_packages\_topics  
     (SciViews\_packages), 28  
 SciViews\_R, 29

`scores (pcomp)`, [23](#)  
`screeplot.pcomp (pcomp)`, [23](#)  
`seq_along()`, [13](#)  
`summary.Correlation (correlation)`, [9](#)  
`summary.pcomp (pcomp)`, [23](#)  
`symnum()`, [11](#), [12](#)  
`system.time()`, [30](#)  
  
`text.pcomp (pcomp)`, [23](#)  
`tidyverse::tidyverse_conflicts()`, [28](#)  
`tidyverse::tidyverse_packages()`, [28](#)  
`timing`, [30](#)  
`timing()`, [3](#)  
  
`utils::install.packages()`, [30](#)  
  
`vectorplot`, [31](#)  
`vectorplot()`, [27](#), [32](#)