

R Cheat Sheet

☐ Getting started ☐ Core language ☐ Vectors & data structures ☐ Functional programming ☐ Object-oriented programming
☐ Conditions & debugging ☐ Metaprogramming ☐ Data import & export ☐ Statistics & graphics ☐ tidyverse & data.table
☐ Packages & tooling ☐ Performance & interop

R 4.6.1 · checked 2026-10-22 · R 4.6.2 scheduled 2026-10-29
Irurueta Docs · Programming Languages · R Reference

TOOL MAP, HELP, STARTUP

R console · Rscript · f.R a b
(commandArgs(TRUE)) · R CMD build pkg · R CMD
check --as-cran pkg-1.0.tar.gz · R CMD
INSTALL pkg · IDEs: RStudio, Positron.

```
?mean; help("["); ??regression
help(package = "stats"); example(lm)
vignette("grid"); str(x); args(paste)
```

Startup: Renviron.site → .Renviron
(KEY=value, Sys.getenv()) → Rprofile.site →
.Rprofile (R code). R —vanilla skips them.

SYNTAX

```
x <- 5; 5 -> y; z = 6 # = top level
f <- function() { n <- n + 1 # enclosing
  'my var' <- 1; '+'(1, 2) # backticks
  r"(C:\temp\"q")" # raw string 4.0
  c(1, 4, 9) |> sqrt() |> sum() # 6
  books |> lm(pages ~ year, data = _)
  \(x) x^2 # function(x) x^2, 4.1
  # comments run to end of line
```

Reserved: if else repeat while function for
in next break TRUE FALSE NULL Inf NaN NA
NA_integer_ NA_real_ NA_character_
NA_complex_1

Precedence (high to low, ?Syntax): :: \$ @ [[[
^ unary - : %any% |> * / + - comparisons ! &
&& | || -> <= = ?

TYPES AND SPECIAL VALUES

typeof	literals	typeof	literals
logical	TRUE NA	complex	1i 2+3i
integer	1L 0x10L	character	"a" 'a'
double	1.5 1e3 Inf	raw	as.raw(1)

Coercion logical < integer < double < complex <
character: c(1, "a") → "1" "a". typeof() ·
class() · mode() · str() · is.*() tests, as.*()
converts: as.numeric("x") → NA, NAs
introduced by coercion.

NA (NA_integer_, NA_real_, NA_character_)
propagates · NULL absent, length 0 · NaN = 0/0 ·
Inf = 1/0 · is.na(), na.rm = TRUE.

VECTORS

```
c(1, 2); 1:5; seq(0, 1, by = 0.25)
seq(2, 10, length.out = 5); rev(x)
seq_len(0) # integer(0); 1:0 is 1 0
rep(c("a", "b"), times = 2, each = 1)
1:6 + 1:2 # recycled: 2 4 6 6 8
2 %in% 1:3; 5 %notin% 1:3 # 4.6.0
NULL %||% "default" # 4.4.0
identical(0.1 + 0.2, 0.3) # FALSE
isTRUE(all.equal(0.1 + 0.2, 0.3))
```

1:3 + 1:2 warns longer object length is not a
multiple of shorter object length. Arithmetic is
vectorised. Float equality: R FAQ 7.31.

SUBSETTING GRID

atomic v	list l	matrix m	data frame df
[v[2:3]	l[l]	m[,] drops	df[i, j]
[[v[[2]]	l[["a"]]	m[[1, 2]]	df[["year"]]
\$ error	l\$a partial	error	df\$year

@ S4 slot, S7 property: obj@title

```
v[c(1, 3)]; v[-1] # positive; negative
v[v > 1]; v["a"] # logical; by name
m[cbind(i, j)]; v[l] # matrix; all
m[, 1, drop = FALSE] # also for df
df[which(df$year > 2015), ] # no NA
v[v < 0] <- 0 # sub-assignment
l["b"] <- list(NULL) # keep a NULL
l$b <- NULL # drop it (FAQ 7.1)
```

DATA STRUCTURES

```
f <- factor(c("lo", "hi", "lo"),
  levels = c("lo", "hi"))
levels(f); table(f); as.integer(f)
yr <- factor(c("1974", "2019"))
as.numeric(as.character(yr)) # FAQ 7.10
m <- matrix(1:6, nrow = 2); dim(m)
t(m); m %*% t(m); rbind(m, 7:9)
solve(A); solve(A, b) # A x = b
apply(m, 1, sum); array(1:24, 2:4)
l <- list(a = 1:3, b = "x"); l[["a"]]
df <- data.frame(id = 1:2, ok = TRUE)
str(books); head(books, 3); nrow(books)
subset(books, year > 2018, title:year)
merge(loans, books, by = "isbn")
aggregate(pages ~ genre, books, mean)
split(books$pages, books$genre)
sort.by(books, ~year) # 4.4.0
attr(x, "units") <- "cm"; attributes(x)
```

CONTROL FLOW

```
if (n > 0) "pos" else "not pos"
ifelse(x > 0, "pos", "neg") # vector
switch(type, csv = read.csv(f),
  stop("unknown type: ", type))
for (i in seq_along(x)) if (i > 1) next
while (k < 100) k <- k * 2
repeat { i <- i + 1; if (i > 2) break }
```

if (c(TRUE, FALSE)) → the condition has length
> 1 (4.2) · if (NA) → missing value where
TRUE/FALSE needed · && || need length 1 (4.3).

APPLY FAMILY

```
lapply(X, f) # list
sapply(X, f) # simplified: unsafe
vapply(X, f, numeric(1)) # typed
Map(f, a, b) # = mapply()
apply(m, 1, f) # 1 rows, 2 columns
tapply(x, grp, f) # per group
Reduce("+", 1:4, accumulate = TRUE)
Filter(is.numeric, l)
do.call(rbind, dfs) # list as arguments
```

FUNCTIONS

```
area <- function(w, h = w) w * h
area(3); area(h = 2, w = 3)
f <- function(x, ...) missing(x)
fmt <- function(type = c("csv", "rds"))
  match.arg(type) # "csv" by default
h <- function(x) {
  on.exit(cat("bye\n"), add = TRUE)
  if (length(x) == 0L) return(NA)
  invisible(max(x)) # no autoprint
}
counter <- function() { i <- 0
  function() { i <- i + 1; i } }
`second<-` <- function(x, value) {
  x[2] <- value; x } # second(v) <- 9
`%>%` <- function(a, b) paste(a, b)
```

Matching: exact name → partial → position; after
... only full names. ...names() (4.1).

ENVIRONMENTS AND LAZINESS

```
environment(); globalenv(); emptyenv()
parent.frame() # the caller's env
parent.env(e) # the enclosing env
search() # attached packages
e <- new.env(); assign("k", 1, e)
get("k", envir = e); local({ a <- 2 })
lazy <- function(x) 10
lazy(stop("never")) # 10: promise
make <- function(n) {
  force(n); function(x) x^n }
```

Lookup: own env → defining env → ... → global →
search() → base → empty. Arguments are lazy
promises.

STRINGS, REGEX, DATES

```
paste("a", "b", sep = "-")
paste(c("x", "y"), collapse = ", ")
sprintf("%-6s%5.1f%03d", "A", pi, 7L)
format(1234567, big.mark = ",")
nchar(s); substr(s, 2, 4); toupper(s)
grep("The", s); grepv("Guin", s) #4.5
sub("-", "", s); gsub("[^0-9]", "", s)
regmatches(s, gregexpr("[0-9]{4}", s))
strsplit("a,b", ",")[1]; trimws(s)
d <- as.Date("2026-06-30"); Sys.Date()
format(d, "%d %b %Y") # "30 Jun 2026"
d + 21L # "2026-07-21"
seq(d, by = "month", length.out = 3)
difftime(d2, d, units = "days")
as.POSIXct("2026-06-30 14:05",
  tz = "UTC")
```

METAPROGRAMMING

```
e <- quote(pages * 2) # a call
eval(e, list(pages = 10)) # 20
n <- 3; `bquote`x + .(n)) # x + 3
sb <- \(x) substitute(x); sb(a + b)
deparse(quote(x + y)) # "x + y"
eval(parse(text = "1 + 2"))
g <- function(x, y) match.call()
do.call("sum", list(1, 2, 3))
fo <- y ~ x + log(z); all.vars(fo)
update(fo, . ~ . - x)
```

OOP SIDE BY SIDE

```
Define; create
S3 b <- structure(list(title = t), class =
  "book")
S4 setClass("Book", slots = c(title =
  "character")); new("Book", title = t)
RC Shelf <- setRefClass("Shelf", fields =
  list(n = "integer")); Shelf$new(n = 0L)
R6 Counter <- R6Class("Counter", public =
  list(n = 0)); Counter$new()
S7 Book <- new_class("Book", properties =
  list(title = class_character());
  Book(title = t)
```

```
info <- \(x, ...) UseMethod("info") #S3
info.book <- \(x, ...) cat(x$title)
info.ebook <- \(x, ...) NextMethod()
print.book <- function(x, ...) {
  cat("<book>", x$title, "\n")
  invisible(x) }
setGeneric("blurb", function(x) # S4
  standardGeneric("blurb"))
setMethod("blurb", "Book",
  function(x) cat(x$title))
setValidity("Book", \(object)
  if (nzchar(object@title)) TRUE
  else "no")
say <- new_generic("say", "x") # S7
method(say, Book) <- \(x) x$title
```

RC and R6 are references (.self, self\$n); to
copy: RC \$copy(), R6 \$clone(). S7: CRAN 0.2.2.

CONDITIONS AND DEBUGGING

```
stop("bad: ", x, call. = FALSE)
warning("low"); message("progress")
tryCatch(log(-1),
  warning = function(w) NA_real_,
  error = \(e) conditionMessage(e),
  finally = cat("done\n"))
withCallingHandlers(f(),
  warning = \(w) {
    message(conditionMessage(w))
    invokeRestart("muffleWarning") })
stop(errorCondition("overdue",
  class = "overdue_error"))
tryCatch(f(), overdue_error = \(e) NA)
try(log("a"), silent = TRUE)
stopifnot("pages > 0" = all(pages > 0))
traceback() · browser() (n s c f Q) ·
debug(f, debugonce(f) · options(error =
  recover) · options(warn = 2)).
```

PERFORMANCE

```
system.time(f()); bench::mark(a(), b())
Rprof("p.out"); f(); Rprof(NULL)
sqrt(x) # vectorised, no loop
out <- numeric(n) # preallocate
parallel::mclapply(X, f, mc.cores = 2)
cl <- parallel::makeCluster(2)
parallel::parLapply(cl, X, f)
Rcpp::cppFunction(
  "int one() { return 1; }")
```

IMPORT AND EXPORT

```
books <- read.csv("books.csv")
write.csv(x, "o.csv", row.names = FALSE)
readLines(p); writeLines(x, p)
saveRDS(x, "b.rds"); readRDS("b.rds")
save(x, file = "a.rda"); load("a.rda")
file.path("data", "b.csv")
list.files("data", pattern = "csv$")
con <- file(p, "r"); readLines(con, 1)
close(con) # or on.exit(close(con))
j <- jsonlite::toJSON(df)
jsonlite::fromJSON(j)
library(DBI)
con <- dbConnect(RSQLite::SQLite())
dbWriteTable(con, "books", books)
dbGetQuery(con, "SELECT * FROM books
  WHERE year > ?", params = list(2018))
dbDisconnect(con)
request("https://api.example.org") |>
  req_url_query(q = "R") |> # http2
  req_perform() |> resp_body_json()
```

STATISTICS AND GRAPHICS

```
dnorm(0); pnorm(1.96); qnorm(0.975)
set.seed(42); rnorm(3, 10, 2); runif(2)
summary(x); table(g); quantile(x)
t.test(mpg ~ am, data = mtcars)
fit <- lm(mpg ~ wt, data = mtcars)
summary(fit); coef(fit); confint(fit)
predict(fit, data.frame(wt = 3))
png("p.png", 800, 600) # pdf(), svg()
par(mfrow = c(1, 2)); plot(x, y)
hist(x); boxplot(mpg ~ cyl, mtcars)
dev.off()
ggplot(mtcars, aes(wt, mpg)) +
  geom_point() + theme_minimal()
```

TIDYVERSE AND DATA.TABLE

```
books |> filter(year > 2000) |>
  mutate(long = pages > 300) |>
  group_by(genre) |>
  summarise(n = n(), avg = mean(pages))
pivot_longer(q, c(q1, q2))
map_dbl(l, mean)
left_join(loans, books, by = "isbn")
dt <- as.data.table(books)
dt[year > 2000, .(n = .N), by = genre]
dt[, long := pages > 300]
```

PACKAGES AND TOOLING

```
install.packages("dplyr")
library(dplyr)
dplyr::filter(df, x > 1)
requireNamespace("jsonlite")
use("tools", "file_ext") # R 4.4.0+
renv::init(); renv::snapshot()
renv::restore() # rebuild the library
usethis::create_package("shelfr")
devtools::document(); devtools::check()
#' @param x Text. @return export
test_that("ISBN is cleaned", {
  expect_equal(nchar("isbn"), 4L)
})
```

Changed 4.0 stringsAsFactors = FALSE · raw strings · matrix is also
since "array" · reference counting · new palette() · list2DF()
older 4.4 %||% · sort_by() · is.atomic(NULL) FALSE · use() ·
tutorials Tailcall() (experimental)

4.1 native pipe |> · \(x) lambda · ...names()

4.5 grepv() · penguins · tools::sha256sum() · is.R() ·
defunct · C23

4.2 if on length > 1 is an error · placeholder _ · UTF-8 on
Windows · Sys.setLanguage()

4.6 %notin% · messageCondition() · bitstring() ·
substr(i2 = NULL) · C++20

4.3 && || on length > 1 is an error · \$a chain (experimental)

Habits vapply not supply · seq_along · TRUE not T · no
attach(), setwd()