Basics in R

Master FBE · CRBE · Université de Toulouse

From installation to publication-ready figures. This guide covers core R syntax, data manipulation with **dplyr/tidyr**, visualisation with **ggplot2**, control flow, functions, and practical tips for reproducible research.

1. Installation and setup

R from cran.r-project.org · **RStudio** from posit.co

Panel	Location	Role
Source	Top-left	Write scripts — run with Ctrl+Enter
Console	Bottom-left	Interactive code and output
Environment	Top-right	All objects in memory
Files/Plots/Help	Bottom-right	Browse, visualise, get help

```
R.version.string
install.packages("ggplot2")
library(ggplot2)
?mean
```

RStudio Projects (recommended): File → New Project sets the working directory automatically and keeps your files organised.

2. Objects and data types

Type	Example	class()	Coerce with
Numeric	3.14	"numeric"	as.numeric()
Character	"setosa"	"character"	as.character()
Logical	TRUE	"logical"	as.logical()
Integer	5L	"integer"	as.integer()
Factor	factor("A")	"factor"	as.factor()

```
x <- 3.14; name <- "setosa"; flag <- TRUE
class(x); is.numeric(x)
as.character(3.14)     # "3.14"
as.logical(0)          # FALSE

# Special values
NA; NaN; Inf; NULL
is.na(NA)              # TRUE
is.finite(Inf)         # FALSE

# Factors (important for modelling)
```

```
f <- factor(c("A","B","A","C"))
levels(f)
table(f)               # count per level
```

3. Vectors

```
v <- c(3,1,4,1,5,9,2,6)
seq(0,1,by=0.25)         # 0.00 0.25 0.50 0.75 1.00
rep(c("A","B"),times=3)  # A B A B A B

# Vectorised arithmetic
v*2; v^2; log(v); sqrt(v)
v + c(10,20)              # recycling

# Summary
mean(v); sd(v); var(v)
range(v); cumsum(v); diff(v)

# Logical operations
v > 3                     # T F T F T T F T
which(v > 3)              # 1 3 5 6 8
v[v > 3]                  # 3 4 5 9 6
any(v > 8); all(v > 0)

# Named vectors
traits <- c(BL=82, BD=15, HL=18)
traits["BL"]              # 82
traits[c("BL","HD")]
```

4. Data frames

```
str(iris); head(iris,3); dim(iris)

# Access
iris$Sepal.Length
iris[,"Sepal.Length"]
iris[1:3,c("Species","Sepal.Length")]

# Filter
iris[iris$Species=="setosa",]
subset(iris, Sepal.Length>7 & Species=="virginica")

# Add/modify
iris$ratio  <- iris$Sepal.Length / iris$Petal.Length
iris$class  <- ifelse(iris$Sepal.Length>6,"large","small")
iris$class2 <- cut(iris$Sepal.Length,breaks=c(0,5,6,Inf),
                   labels=c("small","medium","large"))

# Remove column
iris$ratio <- NULL

# Sort
iris[order(iris$Sepal.Length, decreasing=TRUE),]
```

5. Control flow

```
# if / else
x <- mean(iris$Sepal.Length)
if (x > 6) {
  cat("Large sepals\n")
} else if (x > 5) {
  cat("Medium sepals\n")
} else {
  cat("Small sepals\n")
}

# Vectorised: ifelse(condition, yes, no)
iris$size <- ifelse(iris$Sepal.Length > 6, "large", "small")

# for loop
for (sp in levels(iris$Species)) {
  sub <- iris[iris$Species==sp, "Sepal.Length"]
  cat(sp, ": mean =", round(mean(sub),2), "\n")
}

# while loop
i <- 1
while (iris$Sepal.Length[i] < 7) i <- i+1
cat("First row with SL>=7:", i, "\n")

# next (skip) and break (exit)
for (i in 1:10) {
  if (i %% 2 == 0) next   # skip even numbers
  if (i > 7)       break  # stop at 7
```

```
  cat(i,"")
}
```

6. Writing functions

```
# Basic function
square <- function(x) x^2
square(4)    # 16

# With default arguments and multiple outputs
my_stats <- function(x, na.rm=TRUE, digits=2) {
  list(
    n      = sum(!is.na(x)),
    mean   = round(mean(x, na.rm=na.rm), digits),
    sd     = round(sd(x, na.rm=na.rm), digits),
    cv_pct = round(sd(x,na.rm=na.rm)/mean(x,na.rm=na.rm)*100, 1)
  )
}
my_stats(iris$Sepal.Length)

# Anonymous function (R 4.1+ shorthand)
sapply(iris[,1:4], \(x) round(mean(x),2))

# Error handling
safe_log <- function(x) {
  tryCatch(
    log(x),
    warning = function(w) { cat("Warning:", conditionMessage(w)); NA },
    error   = function(e) { cat("Error:",   conditionMessage(e)); NA }
  )
}
safe_log(-1)   # returns NA with warning message
```

7. The apply family

Avoid explicit for loops for simple operations — the apply family is cleaner and often faster.

Function	Input	Output	When to use
apply()	matrix/array	vector/list	Rows or columns of a matrix
lapply()	list/vector	list	Apply to each element, keep list
sapply()	list/vector	vector/matrix	Like lapply, simplify if possible
vapply()	list/vector	typed vector	Like sapply, specify return type
tapply()	vector+groups	array	Apply by group (like group_by)
mapply()	multiple lists	list	Apply with multiple arguments

```
m <- iris[,1:4]

# apply: over rows (1) or columns (2)
apply(m, 2, mean)            # column means
apply(m, 1, max)             # row maximum
```

```
# sapply: named vector result
sapply(m, function(x) round(sd(x)/mean(x)*100, 1))  # CV per trait

# lapply: returns list
results <- lapply(levels(iris$Species), function(sp) {
  sub <- iris[iris$Species==sp, 1:4]
  colMeans(sub)
})
names(results) <- levels(iris$Species)

# tapply: mean per group per trait
tapply(iris$Sepal.Length, iris$Species, mean)
```

8. String manipulation

```
# Base R
species <- c("Gobio gobio","Cottus perifretum","Leuciscus cephalus")

paste("sp", 1:3, sep="_")         # "sp_1" "sp_2" "sp_3"
paste0("sp", 1:3)                  # "sp1" "sp2" "sp3"

toupper("gobio gobio")             # "GOBIO GOBIO"
nchar(species)                     # 10 17 17

gsub(" ", "_", species)            # replace spaces
sub("^(\\w+) .*", "\\1", species)  # extract genus

grepl("Gobio", species)            # FALSE for non-matches
grep("us$", species, value=TRUE)   # ends with "us"

# Extract genus
strsplit(species, " ")[[1]][1]     # "Gobio"
sapply(strsplit(species," "), `[`, 1)  # all genera

# stringr (tidyverse — cleaner syntax)
library(stringr)
str_to_upper(species)
str_detect(species, "us$")         # same as grepl
str_extract(species, "^\\w+")      # genus
str_replace_all(species, " ", "_") # gsub equivalent
```

9. Import and export

```
# CSV
df <- read.csv("traits.csv")
df <- read.csv("traits.csv", sep=";", dec=",")
df <- read.csv("traits.csv", na.strings=c("NA","","N/A"))

# Tab-separated
df <- read.table("traits.txt", header=TRUE, sep="\t")

# Excel
library(readxl)
df <- read_excel("traits.xlsx", sheet="Sheet1")
df <- read_excel("traits.xlsx", skip=2)   # skip 2 header rows

# Large files (fast)
library(data.table)
df <- fread("large_file.csv")              # 10x faster than read.csv

# Write
write.csv(df, "output.csv", row.names=FALSE)
write.table(df, "output.txt", sep="\t", row.names=FALSE, quote=FALSE)

# R binary (preserves all R attributes)
saveRDS(df, "traits.rds")         # single object
df2 <- readRDS("traits.rds")

save(df, model, "workspace.RData") # multiple objects
```

```
load("workspace.RData")
```

10. Data manipulation with dplyr

```
library(dplyr)

# The 8 core verbs
iris |>
  filter(Sepal.Length > 5.0) |>           # keep rows
  select(Species, starts_with("Sepal")) |> # keep columns
  rename(sp = Species) |>                  # rename
  mutate(
    ratio = Sepal.Length/Sepal.Width,      # add column
    size  = case_when(                     # multiple conditions
      Sepal.Length > 6.5 ~ "large",
      Sepal.Length > 5.5 ~ "medium",
      TRUE               ~ "small"
    )
  ) |>
  group_by(sp, size) |>
  summarise(
    n        = n(),
    mean_SL  = mean(Sepal.Length),
    sd_SL    = sd(Sepal.Length),
    .groups  = "drop"
  ) |>
  arrange(desc(mean_SL))
```

Joining datasets:

```
sp_info <- data.frame(
  Species = levels(iris$Species),
  habitat = c("meadow","wetland","forest"),
  native  = c(TRUE, TRUE, FALSE)
)

left_join(iris, sp_info, by="Species")  # all iris rows kept
inner_join(iris,sp_info,by="Species")   # only matching rows
anti_join(iris, sp_info,by="Species")   # rows NOT in sp_info
```

Window functions:

```
iris |>
  group_by(Species) |>
  mutate(
    rank_SL   = rank(Sepal.Length),
    cumsum_SL = cumsum(Sepal.Length),
    z_SL      = (Sepal.Length - mean(Sepal.Length)) / sd(Sepal.Length)
  )
```

11. Reshaping with tidyr

```
library(tidyr)

# Wide → Long (one row per measurement)
iris_long <- iris |>
  pivot_longer(cols=-Species, names_to="trait", values_to="value")
head(iris_long,4)
```

```
# Long → Wide
iris_long |>
  group_by(Species,trait) |>
  summarise(mean=mean(value),.groups="drop") |>
  pivot_wider(names_from=trait, values_from=mean)

# Separate one column into multiple
df <- data.frame(name=c("Gobio gobio","Cottus perifretum"))
tidyr::separate(df, name, into=c("genus","species"), sep=" ")

# Unite multiple columns into one
df2 <- data.frame(genus=c("Gobio","Cottus"),species=c("gobio","perifretum"))
tidyr::unite(df2, "binomial", genus,species, sep=" ")

# Handle NAs
drop_na(df)                           # remove rows with any NA
replace_na(df, list(SL=0, BL=0))      # fill NAs with 0
fill(df, SL, .direction="down")       # fill down from last non-NA
```

12. Visualisation with ggplot2

Layer	Code	Role
Canvas	ggplot(data, aes(...))	Data + aesthetic mapping
Points	geom_point()	Scatter plot
Lines	geom_line() / geom_smooth()	Trends and curves
Bars	geom_bar() / geom_col()	Counts or values
Distributions	geom_histogram() / geom_density()	Distribution of one variable
Boxes	geom_boxplot() / geom_violin()	Group comparisons
Facets	facet_wrap() / facet_grid()	Small multiples
Labels	labs() / annotate()	Text on the plot
Theme	theme_bw() / theme()	Overall look

```
library(ggplot2)

# Scatter + regression
ggplot(iris,aes(Sepal.Length,Petal.Length,colour=Species)) +
  geom_point(size=2.5,alpha=0.7) +
  geom_smooth(method="lm",se=FALSE,linewidth=0.8) +
  scale_colour_manual(values=c("#2980b9","#27ae60","#e67e22")) +
  labs(title="Functional trait space",
       x="Sepal length (cm)",y="Petal length (cm)") +
  theme_bw()+theme(panel.grid.minor=element_blank())

# Violin + boxplot + points
ggplot(iris,aes(Species,Sepal.Length,fill=Species)) +
  geom_violin(alpha=0.4,trim=FALSE) +
  geom_boxplot(width=0.15,alpha=0.8,outlier.shape=NA) +
  geom_jitter(width=0.08,alpha=0.5,size=1.2) +
  scale_fill_manual(values=c("#2980b9","#27ae60","#e67e22")) +
  theme_bw()+theme(legend.position="none")

# Faceted — all traits at once
library(tidyr)
iris_long <- pivot_longer(iris,-Species,names_to="trait",values_to="value")
ggplot(iris_long,aes(Species,value,fill=Species)) +
  geom_boxplot(alpha=0.7) +
  facet_wrap(~trait,scales="free_y",nrow=2) +
  scale_fill_manual(values=c("#2980b9","#27ae60","#e67e22")) +
  theme_bw()+theme(legend.position="none",
                   axis.text.x=element_text(angle=30,hjust=1))

# Correlation matrix heatmap
library(reshape2)
cor_mat <- round(cor(iris[,1:4]),2)
cor_melt <- melt(cor_mat)
ggplot(cor_melt,aes(Var1,Var2,fill=value,label=value)) +
  geom_tile(colour="white") +
  geom_text(size=3) +
  scale_fill_gradient2(low="#c0392b",mid="white",high="#2980b9",midpoint=0) +
```

```
  theme_minimal()+theme(axis.text.x=element_text(angle=45,hjust=1))

# Publication-ready: save correctly
ggsave("figures/traits.pdf",width=8,height=5,device=cairo_pdf)
ggsave("figures/traits.png",dpi=300,width=8,height=5,bg="white")
```

13. Reproducibility tips

Practice	Why	How
set.seed()	Reproducible random numbers	set.seed(42) before sample/permutations
renv	Track package versions	renv::init(); renv::snapshot()
sessionInfo()	Record R + package versions	Call at end of script
RMarkdown / Quarto	Code + text in one document	knitr::knit() / quarto::quarto_render()
here package	Portable file paths	here::here(data, traits.csv)

```
# Always start a reproducible script with:
set.seed(2025)

# Record session info at the end:
sessionInfo()
# or the more readable:
library(sessioninfo)
session_info()
```

References: R for Data Science (r4ds.hadley.nz) · ggplot2: Elegant Graphics for Data Analysis (ggplot2-book.org) · Advanced R (adv-r.hadley.nz)