

# Data Science using R Programming

## An Introduction to R

Dr. Kapil Kumar

Assistant Professor, Department of Statistics  
Central University of Haryana, Mahendergarh, India  
kapilstats@gmail.com

2–3 November 2018

Workshop on Data Science using R-Programming  
Department of Statistics, Hindu College, University of Delhi  
**Technical session**

# Plan of the session

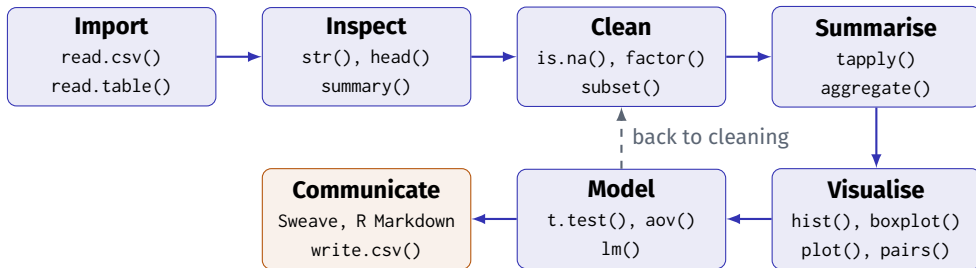
- 1 Data science and R
- 2 Import and inspect
- 3 Clean and summarise
- 4 Visualise
- 5 Model
- 6 Communicate

One pass through a complete analysis,  
with base R and five small data sets.

**By the end you can** import a data file, check and clean it, summarise it by groups, draw informative graphs, fit and interpret  $t$ -tests, an ANOVA and a regression, and save your results so that others can reproduce them.

- 1 Data science and R
- 2 Import and inspect
- 3 Clean and summarise
- 4 Visualise
- 5 Model
- 6 Communicate

# The data-science workflow



Real analyses loop: a model often reveals a data problem that sends you back to cleaning. Every step lives in one script, so the whole loop can be re-run in seconds.

# Why R for data science?

- An integrated suite for data manipulation, calculation and graphical display
- A real programming language: functional, vectorised and extensible
- Free and open source, on every platform, with thousands of packages on CRAN
- Graphics of publication quality
- Created by Ross Ihaka and Robert Gentleman (University of Auckland); now developed by the R Core Team

**Reproducibility** A script records every step from the raw file to the final figure. Anyone, including you in six months, can re-run it and get the same answer.

**RStudio** A free development environment for R with an editor, console, plots and help in one window. Install R first, then RStudio.

```
workshop/  
|  
| analysis.R  
| report.Rnw  
| data/  
| | D1.csv, DF3.txt  
| | airquality_clean.csv  
| figure/
```

- One folder per project, opened as an RStudio *project*: it becomes the working directory
- Read files with relative paths, "data/D1.csv", so the folder works on any computer
- Treat raw data as read-only: every correction is made in code, never by hand in Excel
- Save derived files under new names

# Packages, data and help

```
# install once (needs internet)
# install.packages("arm")
library(arm) # for display()
```

```
?read.csv # a function
help(package = "datasets") # data sets
example(boxplot) # examples
```

| Data set    | Used for                      |
|-------------|-------------------------------|
| D1.csv      | import, summaries, two groups |
| DF3.txt     | text files, reshaping         |
| C02         | checking a design, bar charts |
| airquality  | missing values, regression    |
| sleep       | paired $t$ -test              |
| PlantGrowth | one-way ANOVA                 |

The two files are small illustrative data sets; the others come with R.

- 1 Data science and R
- 2 Import and inspect
- 3 Clean and summarise
- 4 Visualise
- 5 Model
- 6 Communicate

# R objects you will meet

| Object     | Holds                                             | Created with                                              |
|------------|---------------------------------------------------|-----------------------------------------------------------|
| vector     | values of one type: numbers, text or TRUE/FALSE   | <code>c()</code> , <code>seq()</code> , <code>1:10</code> |
| factor     | categories, such as treatment groups              | <code>factor()</code> , <code>gl()</code>                 |
| list       | named components of any type and length           | <code>list()</code> ; every test result                   |
| data frame | a data set: rows are units, columns are variables | <code>data.frame()</code> , <code>read.csv()</code>       |
| function   | reusable code                                     | <code>function(x) { ... }</code>                          |

**Two rules** Everything in R is an object, and everything that happens is a function call. Use `class()` and `str()` to ask any object what it is.

# Vectors: computing on a whole column at once

```
x <- c(2, 4, 3, 5, 7, 9)           # scores of 6 students
round((x - mean(x)) / sd(x), 2)    # z-scores, no loop
## [1] -1.15 -0.38 -0.77  0.00  0.77  1.53

x > mean(x)                        # a logical vector
## [1] FALSE FALSE FALSE FALSE  TRUE  TRUE

sum(x > 4); mean(x > 4)            # count, proportion
## [1] 3
## [1] 0.5
```

**Vectorised** One expression acts on every element. It is shorter, clearer and faster than a loop.

In arithmetic TRUE counts as 1 and FALSE as 0, so `sum()` of a condition counts the cases and `mean()` gives their proportion.

# Lists and data frames

```
LD1 <- list(x = c(2, 5, 4), Name = c("X", "Y", "Z"),
           Median = 4)
str(LD1)

## List of 3
## $ x      : num [1:3] 2 5 4
## $ Name   : chr [1:3] "X" "Y" "Z"
## $ Median: num 4

scores <- data.frame(student = LETTERS[1:6], score = x)
scores[scores$score > 4, ] # rows meeting a condition

##   student score
## 4      D      5
## 5      E      7
## 6      F      9
```

**A data frame is a list** of equal-length columns, printed as a table: `is.list(scores)` is TRUE. So `scores$score` extracts a column just as `LD1$Name` extracts a component.

`scores[rows, columns]`: a condition in the row slot filters, an empty slot means “all”.

# Importing files

```
D1 <- read.csv("data/D1.csv")
head(D1, 3)

##    obese lean
## 1   9.21 7.53
## 2  11.51 7.48
## 3  12.79 8.08

DF3 <- read.table("data/DF3.txt", header = TRUE)
DF3

##    batchA batchB batchC
## A    20.0     40     10
## B    20.5     42     15
## C    13.0     38     12
## D    16.0     56     10
## E    11.0     57      3
```

| File       | Read with                                    |
|------------|----------------------------------------------|
| .csv       | <code>read.csv()</code>                      |
| .txt, .dat | <code>read.table(header = TRUE)</code>       |
| Excel      | save as CSV, or <code>readxl</code>          |
| R objects  | <code>readRDS()</code> , <code>load()</code> |

D1: energy expenditure of four obese and four lean women. DF3: marks of five students A–E in three courses; the first column of the file becomes the row names.

# A first look at a data set

```
dim(CO2); names(CO2)
```

```
## [1] 84 5  
## [1] "Plant"      "Type"      "Treatment" "conc"  
## [5] "uptake"
```

```
head(CO2, 3)
```

```
## Grouped Data: uptake ~ conc | Plant  
##   Plant   Type Treatment conc uptake  
## 1   Qn1 Quebec nonchilled   95   16.0  
## 2   Qn1 Quebec nonchilled  175   30.4  
## 3   Qn1 Quebec nonchilled  250   34.8
```

```
summary(CO2$uptake)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.   
##      7.7   17.9   28.3   27.2   37.1   45.5
```

- CO2: carbon dioxide uptake of 12 grass plants, six from Quebec and six from Mississippi
- Half of each were chilled overnight; uptake was measured at seven ambient CO<sub>2</sub> concentrations
- `help(CO2)` describes the variables and their units

**First questions** How many rows and columns? What is each variable? Do the ranges look sensible?

# Checking the design

```
table(CO2$Type, CO2$Treatment)

##
##              nonchilled chilled
##   Quebec              21      21
##   Mississippi          21      21

unique(CO2$conc)

## [1]   95  175  250  350  500  675 1000

anyNA(CO2); anyDuplicated(CO2)

## [1] FALSE
## [1] 0
```

**Reading it** A balanced design: 21 measurements in each Type  $\times$  Treatment cell, that is 3 plants  $\times$  7 concentrations. Concentration takes 7 fixed values. No missing values and no duplicated rows.

Checks like these take a minute and prevent the most common analysis errors.

- 1 Data science and R
- 2 Import and inspect
- 3 Clean and summarise**
- 4 Visualise
- 5 Model
- 6 Communicate

# Missing values

```
aq <- airquality
dim(aq)

## [1] 153 6

colSums(is.na(aq))      # NAs per column

##      Ozone Solar.R      Wind      Temp      Month      Day
##       37       7         0         0         0         0

mean(aq$Ozone)

## [1] NA

mean(aq$Ozone, na.rm = TRUE)

## [1] 42.13
```

- `airquality`: daily ozone (ppb), solar radiation, wind (mph) and maximum temperature (°F) in New York, May to September 1973
- NA means “not available”; any arithmetic with NA gives NA
- `na.rm = TRUE` drops missing values first

# Recoding and creating variables

```
aq$Month <- factor(aq$Month, levels = 5:9, labels = month.abb[5:9]) # numbers to labels
aq$TempC <- round((aq$Temp - 32) * 5 / 9, 1) # Fahrenheit to Celsius
head(aq, 3)
```

```
##   Ozone Solar.R Wind Temp Month Day TempC
## 1    41     190  7.4   67   May  1  19.4
## 2    36     118  8.0   72   May  2  22.2
## 3    12     149 12.6   74   May  3  23.3
```

**Factors** A month coded 5–9 is a category, not a quantity. As a factor it gets readable labels and is treated as groups by tables, graphs and models.

**Keep the original** New variables are added as new columns; the raw file on disk is never changed.

# Where are the gaps?

```
table(aq$Month, missing = is.na(aq$Ozone))

##      missing
##      FALSE TRUE
##   May      26   5
##   Jun       9  21
##   Jul      26   5
##   Aug      26   5
##   Sep      29   1

aqc <- na.omit(aq)      # complete cases only
dim(aqc)

## [1] 111   7
```

**June is mostly missing** Ozone is missing on 21 of 30 June days, so any June summary rests on only 9 days.

**Complete cases** `na.omit()` keeps 111 of 153 days. Report how many rows were dropped, and ask whether the gaps could be related to the values (for example, a sensor failing on hot days).

# Selecting rows and reshaping

```
subset(aqc, TempC > 34,  
       select = c(Month, Day, Ozone, TempC))
```

```
##      Month Day Ozone TempC  
## 120    Aug  28    76  36.1  
## 121    Aug  29   118  34.4  
## 122    Aug  30    84  35.6  
## 123    Aug  31    85  34.4
```

The hottest days: `subset()` filters rows by a condition and keeps the named columns.

```
long <- stack(DF3)      # wide to long  
names(long) <- c("mark", "course")  
dim(long); head(long, 3)
```

```
## [1] 15  2  
##   mark course  
## 1 20.0 batchA  
## 2 20.5 batchA  
## 3 13.0 batchA
```

DF3 is *wide* (one column per course).  
Models and grouped graphs need it  
*long*: one value column and one group column.

# Grouped summaries

```
tapply(aq$Ozone, aq$Month, mean, na.rm = TRUE)
```

```
##   May   Jun   Jul   Aug   Sep  
## 23.62 29.44 59.12 59.96 31.45
```

```
aggregate(Ozone ~ Month, data = aq, FUN = median)
```

```
##   Month Ozone  
## 1   May    18  
## 2   Jun    23  
## 3   Jul    60  
## 4   Aug    52  
## 5   Sep    23
```

```
with(CO2, tapply(uptake,  
                 list(Type, Treatment), mean))
```

```
##                nonchilled chilled  
## Quebec                35.33    31.75  
## Mississippi            25.95    15.81
```

**Split, apply, combine** `tapply()`  
and `aggregate()` split a variable  
by groups, apply a function to  
each piece and combine the  
results.

# A summary function of your own

```
describe <- function(v) {  
  c(n = length(v), mean = mean(v),  
    sd = sd(v), cv = 100 * sd(v) / mean(v),  
    se = sd(v) / sqrt(length(v)))  
}  
  
round(sapply(D1, describe), 2)  
  
##      obese lean  
## n      4.00 4.00  
## mean 10.57 7.80  
## sd    1.90 0.34  
## cv    17.97 4.38  
## se     0.95 0.17
```

```
oz <- split(aqc$Ozone, aqc$Month) # by month  
round(sapply(oz, describe), 1)  
  
##      May  Jun  Jul  Aug  Sep  
## n      24.0  9.0 26.0 23.0 29.0  
## mean  24.1 29.4 59.1 60.0 31.4  
## sd     22.9 18.2 31.6 41.8 24.1  
## cv     94.9 61.8 53.5 69.6 76.8  
## se      4.7  6.1  6.2  8.7  4.5
```

## One function, many groups

`split()` cuts a variable into a list by group; `sapply()` applies `describe` to each piece and returns a table.

- **Energy expenditure:** mean 10.57 for the obese and 7.80 for the lean women. The obese group is far more variable (CV 18% against 4%).
- **Ozone:** July and August average about 60 ppb, roughly twice the level of May, June and September.
- Ozone's CV is above 50% in every month, and overall its mean (42.1 ppb) exceeds its median (31.5 ppb): the distribution is right-skewed, a hint for the modelling step.

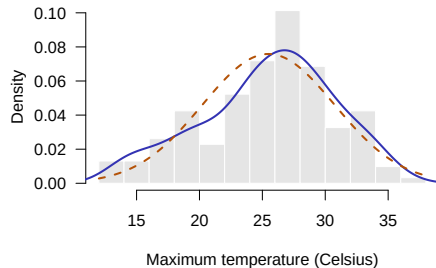
**Sample sizes matter** June's mean rests on 9 complete days and each group in D1 has only four women. A summary table should always carry its  $n$ .

- 1 Data science and R
- 2 Import and inspect
- 3 Clean and summarise
- 4 Visualise**
- 5 Model
- 6 Communicate

# One variable: histogram, density and a normal curve

```
col_main <- "#3333B3"; col_alt <- "#B45309"
par(mar = c(4, 4, 1, 1), las = 1)
hist(aq$TempC, freq = FALSE, breaks = 12,
     main = NULL, col = "grey90", border = "white",
     xlab = "Maximum temperature (Celsius)")
lines(density(aq$TempC), col = col_main, lwd = 2)
curve(dnorm(x, mean(aq$TempC), sd(aq$TempC)),
      add = TRUE, col = col_alt, lwd = 2, lty = 2)
```

freq = FALSE puts the histogram on the density scale, so that curves can be laid over it. The first line fixes the two plot colours used from now on.

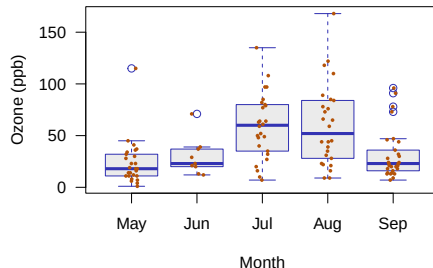


**Reading it** From 13.3 to 36.1°C. The kernel density (solid) is close to the normal curve (dashed), with a slightly longer left tail.

# Comparing groups: box plots with the data

```
par(mar = c(4, 4, 1, 1), las = 1)
boxplot(Ozone ~ Month, data = aq, col = "grey92",
        border = col_main, ylab = "Ozone (ppb)")
stripchart(Ozone ~ Month, data = aq, add = TRUE,
           vertical = TRUE, method = "jitter",
           pch = 16, cex = 0.5, col = col_alt)
```

**Show the points** The June box looks as solid as the others, but the dots reveal that it summarises only nine days.

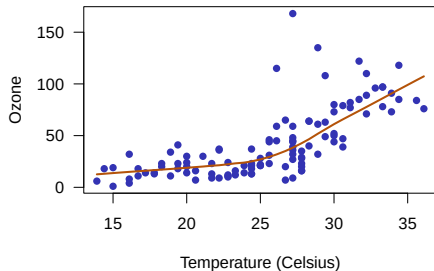


**Reading it** July and August have higher medians and a much larger spread; unusually high days occur in every month.

## Two variables: a scatter plot with a smoother

```
par(mar = c(4, 4, 1, 1), las = 1)
plot(Ozone ~ TempC, data = aqc, pch = 16,
     col = col_main, xlab = "Temperature (Celsius)")
lines(lowess(aqc$TempC, aqc$Ozone),
      col = col_alt, lwd = 2)
```

`lowess()` fits a smooth curve without assuming any shape; it is the quickest check of whether a straight line is reasonable.

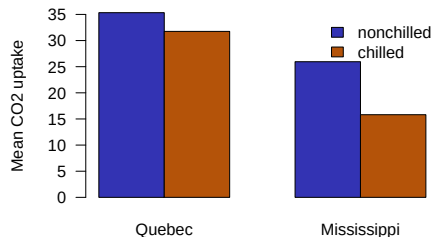


**Reading it** Ozone rises with temperature, bending upwards above about 27°C, and the scatter grows with the level: model  $\log(\text{Ozone})$ .

## Categories: a grouped bar chart

```
m <- with(CO2, tapply(uptake, list(Treatment, Type),
                           mean))
par(mar = c(4, 4, 1, 1), las = 1)
barplot(m, beside = TRUE, col = c(col_main, col_alt),
        ylab = "Mean CO2 uptake", legend.text = TRUE,
        args.legend = list(x = "topright", bty = "n"))
```

When the effect of one factor depends on the level of another, the factors *interact*. A graph shows it at a glance.



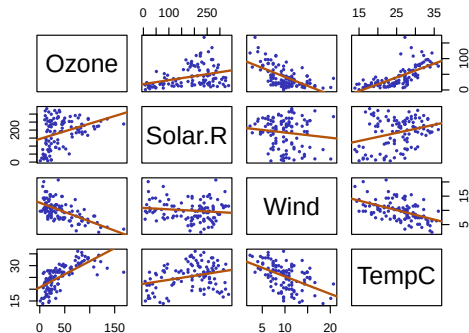
**Reading it** Chilling lowers mean uptake a little in Quebec plants (35.3 to 31.8) but sharply in Mississippi plants (26.0 to 15.8).

# Many variables: a scatterplot matrix with your own panel

```
pan1 <- function(x, y) {  
  points(x, y, pch = 16, cex = 0.6,  
         col = col_main)  
  abline(lm(y ~ x), col = col_alt, lwd = 2)  
}  
vars <- c("Ozone", "Solar.R", "Wind",  
         "TempC")
```

`pairs()` calls `pan1` once for every pair of variables. Writing the panel function yourself puts a fitted line in each one.

```
pairs(aqc[, vars], panel = pan1)
```



- 1 Data science and R
- 2 Import and inspect
- 3 Clean and summarise
- 4 Visualise
- 5 Model**
- 6 Communicate

## Two independent groups: obese and lean women

```
t.test(D1$obese, D1$lean)          # Welch two-sample t-test

##
##  Welch Two Sample t-test
##
## data:  D1$obese and D1$lean
## t = 2.9, df = 3.2, p-value = 0.06
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.194  5.744
## sample estimates:
## mean of x mean of y
##    10.57    7.80
```

The data are stored wide (one column per group), so the two columns are passed directly. Welch's test does not assume equal variances, which matters here: the two SDs differ more than fivefold.

## Reading the comparison, and a second opinion

```
wilcox.test(D1$obese, D1$lean)$p.value  
  
## [1] 0.02857  
  
2 / choose(8, 4)      # smallest possible p  
  
## [1] 0.02857
```

**Welch  $t$ -test** Mean difference 2.77;  
95% CI from  $-0.19$  to  $5.74$ ,  $p = 0.059$ .

**Wilcoxon rank-sum test** Every  
obese value exceeds every lean  
value, which gives the smallest  $p$   
possible with  $4 + 4$  observations,  
 $0.029$ .

**What to conclude** Two reasonable tests fall on either side of 5%. With four women per group the evidence is suggestive, not conclusive: the honest next step is a larger study.

# Paired data: the sleep experiment

```
drug1 <- sleep$extra[sleep$group == 1] # extra hours of sleep
drug2 <- sleep$extra[sleep$group == 2] # same 10 patients
t.test(drug2, drug1, paired = TRUE)

##
## Paired t-test
##
## data: drug2 and drug1
## t = 4.1, df = 9, p-value = 0.003
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  0.7001 2.4599
## sample estimates:
## mean difference
##           1.58
```

```
# ignoring the pairing:
t.test(drug2, drug1)$p.value

## [1] 0.07939
```

**Reading it** Each patient took both drugs, and the two responses correlate ( $r = 0.8$ ). Differencing removes this patient-to-patient variation:  $p$  falls from 0.08 to 0.003.

# More than two groups: one-way ANOVA

```
fit_pg <- aov(weight ~ group,  
              data = PlantGrowth)  
print(summary(fit_pg),  
      signif.stars = FALSE)  
  
##              Df Sum Sq Mean Sq F value Pr(>F)  
## group         2   3.77    1.883    4.85  0.016  
## Residuals    27  10.49    0.389
```

PlantGrowth: dried weight of plants under a control and two treatments, 10 plants each.

```
TukeyHSD(fit_pg)$group  
  
##              diff      lwr      upr    p adj  
## trt1-ctrl    -0.371 -1.0622  0.3202  0.39087  
## trt2-ctrl     0.494 -0.1972  1.1852  0.19800  
## trt2-trt1     0.865  0.1738  1.5562  0.01201
```

```
bartlett.test(weight ~ group,  
              data = PlantGrowth)$p.value  
  
## [1] 0.2371  
  
shapiro.test(residuals(fit_pg))$p.value  
  
## [1] 0.4379
```

## Reading the ANOVA

- $F = 4.85$  on 2 and 27 df,  $p = 0.016$ : the three mean weights are not all equal
- Tukey's comparisons: only treatment 2 versus treatment 1 differs (by 0.87, adjusted  $p = 0.012$ )
- Neither treatment differs significantly from the control
- Checks: Bartlett  $p = 0.24$  (equal variances) and Shapiro-Wilk  $p = 0.44$  (normal residuals): no evidence against the assumptions

**For the report** “Treatment 2 gave heavier plants than treatment 1; neither treatment clearly beat the control.” A significant  $F$  does not mean every pair differs.

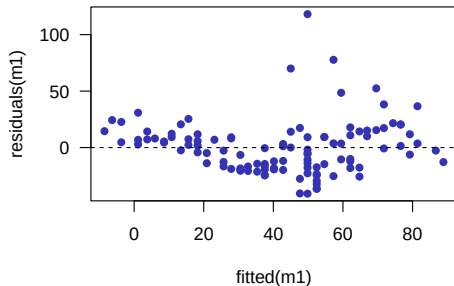
# Regression: ozone and temperature

```
m1 <- lm(Ozone ~ TempC, data = aqc)
display(m1)

## lm(formula = Ozone ~ TempC, data = aqc)
##               coef.est coef.se
## (Intercept) -69.48    11.20
## TempC        4.39     0.43
## ---
## n = 111, k = 2
## residual sd = 23.94, R-Squared = 0.49
```

Each extra degree Celsius goes with about 4.4 ppb more ozone;  $R^2 = 0.49$ . But the residuals (right) curve and fan out: the straight line is not adequate on this scale.

```
par(mar = c(4, 4, 1, 1), las = 1)
plot(fitted(m1), residuals(m1), pch = 16,
     col = col_main)
abline(h = 0, lty = 2)
```



# A better model on the log scale

```
m2 <- lm(log(Ozone) ~ TempC, data = aqc)
round(exp(coef(m2)), 3)           # multiplicative effects

## (Intercept)      TempC
##      1.377      1.129

m3 <- lm(log(Ozone) ~ TempC + Wind + Solar.R, data = aqc)
display(m3, digits = 3)

## lm(formula = log(Ozone) ~ TempC + Wind + Solar.R, data = aqc)
##               coef.est coef.se
## (Intercept)  1.317    0.376
## TempC        0.088    0.011
## Wind        -0.062    0.016
## Solar.R      0.003    0.001
## ---
## n = 111, k = 4
## residual sd = 0.509, R-Squared = 0.66
```

## Reading the models

- On the log scale a coefficient  $b$  means: one unit more of  $x$  multiplies the response by  $e^b$
- Temperature alone: each extra degree Celsius multiplies ozone by 1.129, about 13% more;  $R^2 = 0.55$
- Holding wind and sunshine fixed:  $+1^\circ\text{C}$  gives +9% ozone, +1 mph of wind gives -6%, and 100 more langleys of solar radiation give +29%;  $R^2 = 0.66$

**Keep in mind** These are associations in observational data, not causes. Before reporting, check the residuals of the final model, just as we did for the straight line.

- 1 Data science and R
- 2 Import and inspect
- 3 Clean and summarise
- 4 Visualise
- 5 Model
- 6 Communicate**

- A report that *runs its own code*: text, R code and output in one file
- Change the data, re-run, and every number, table and figure updates
- **Sweave / knitr** (.Rnw):  $\text{\LaTeX}$  + R  $\rightarrow$  PDF. The workshop handout and these slides are made this way
- **R Markdown** (.Rmd): Markdown + R  $\rightarrow$  HTML, Word or PDF

## An .Rnw code chunk

```
<<ozone-summary, echo=TRUE>>=  
summary(aq$Ozone)  
@
```

## A number inside a sentence

Mean ozone was  
`\Sexpr{round(mean(aqc$Ozone), 1)}` ppb.  
becomes: Mean ozone was 42.1 ppb.

# Exporting results

```
write.csv(aqc, "data/airquality_clean.csv",
          row.names = FALSE)
check <- read.csv("data/airquality_clean.csv")
dim(check)           # round trip is complete

## [1] 111    7

R.version.string      # record the versions

## [1] "R version 4.3.3 (2024-02-29)"

packageVersion("arm")

## [1] '1.13.1'
```

```
png("figure/ozone_by_month.png",
    width = 1600, height = 1000, res = 200)
boxplot(Ozone ~ Month, data = aq)
dev.off()
```

Save figures with `png()` or `pdf()` and `dev.off()`; record the R and package versions with the results (`sessionInfo()` lists them all).

# The workflow, function by function

| Step        | Functions used today                                                                                                                                        |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Import      | <code>read.csv()</code> , <code>read.table()</code>                                                                                                         |
| Inspect     | <code>dim()</code> , <code>names()</code> , <code>head()</code> , <code>str()</code> , <code>summary()</code> , <code>table()</code> , <code>anyNA()</code> |
| Clean       | <code>is.na()</code> , <code>na.omit()</code> , <code>factor()</code> , <code>subset()</code> , <code>stack()</code>                                        |
| Summarise   | <code>tapply()</code> , <code>aggregate()</code> , <code>sapply()</code> , <code>split()</code> , <b>your own</b> <code>describe()</code>                   |
| Visualise   | <code>hist()</code> , <code>density()</code> , <code>boxplot()</code> , <code>plot()</code> , <code>barplot()</code> , <code>pairs()</code>                 |
| Model       | <code>t.test()</code> , <code>wilcox.test()</code> , <code>aov()</code> , <code>TukeyHSD()</code> , <code>lm()</code> , <code>display()</code>              |
| Communicate | Sweave / R Markdown, <code>write.csv()</code> , <code>png()</code> , <code>sessionInfo()</code>                                                             |

**Where next** The same workflow can be written with the **dplyr**, **tidyr** and **ggplot2** packages; *R for Data Science* (Wickham et al., free online) teaches it step by step.

## Key take-aways

- 1 **Data science is a loop:** import, inspect, clean, summarise, visualise, model, communicate, and back again.
- 2 **Inspect before you analyse:** dimensions, types, ranges, design, duplicates and missing values.
- 3 **Missing is information:** count it, locate it, and report what you dropped.
- 4 **Graphs find what tables hide:** skewness, interactions, curvature and thin groups.
- 5 **The design chooses the test:** independent groups, pairs, or several groups.
- 6 **Transform when the data ask for it,** and interpret effects on the new scale.
- 7 **Make it reproducible:** one script or report, relative paths, recorded versions.

- Wickham, H., Çetinkaya-Rundel, M., and Golemund, G. (2023). *R for Data Science*, 2nd Ed., O'Reilly, Sebastopol. Free online: <https://r4ds.hadley.nz>
- Dalgaard, P. (2008). *Introductory Statistics with R*, 2nd Ed., Springer, New York.
- Jones, O., Harden, S., and Crawley, M. J. (2022). *The R Book*, 3rd Ed., Wiley, Hoboken.
- Tukey, J. W. (1977). *Exploratory Data Analysis*, Addison-Wesley, Reading.
- Xie, Y., Allaire, J. J., and Golemund, G. (2018). *R Markdown: The Definitive Guide*, CRC Press, Boca Raton.
- Gelman, A., and Hill, J. (2007). *Data Analysis Using Regression and Multilevel/Hierarchical Models*, Cambridge University Press, Cambridge.
- R Core Team. *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna. <https://www.R-project.org>

# Thank you

Questions and discussion