Data Analysis using R

Dr. Kapil Kumar

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

9 February 2019

One-day Workshop on Data Analysis Using R (Tratrix Club)
Department of Mathematics & Statistics, Manipal University Jaipur
Invited lecture

Plan of the session

1. Getting started with R and RStudio
2. Vectors and data objects
3. Graphics in base R
4. Categorical data
5. Comparing groups: t-tests and ANOVA
6. Correlation and regression

By the end you can install R and RStudio, create and summarise vectors, read a CSV file, draw basic plots, and run and interpret a chi-square test, t-tests, a one-way ANOVA and a simple linear regression.

Outline

1. Getting started
2. Vectors and data
3. Graphics
4. Categorical data
5. Comparing groups
6. Correlation and regression

Installing R and RStudio

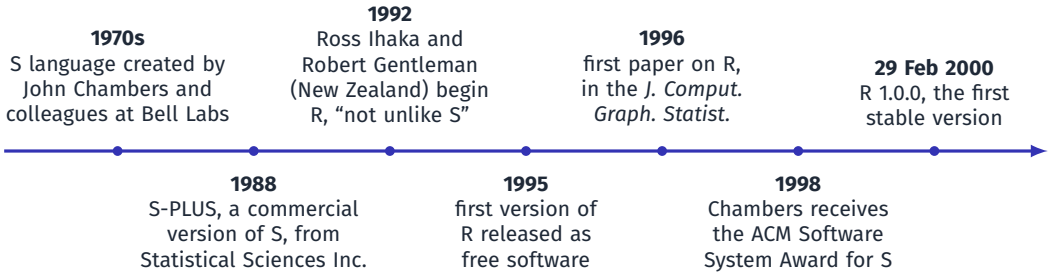
1. Install R first, from CRAN:
   `https://cran.r-project.org`
2. Then install the free RStudio Desktop:
   `https://posit.co/download/rstudio-desktop/`
3. Open **RStudio** (not R) for daily work; it finds R automatically.

Both are free and run on Windows, macOS and Linux.

Engine and dashboard R is the engine that does the computing. RStudio is the dashboard around it: an editor, a console, a plot window and a file browser in one place.

Then Install extra packages once with `install.packages()`; load them in every session with `library()`.

Where R came from



The name R is named partly after the first names of its two authors (Ross and Robert) and partly as a play on the name S.

Why R?

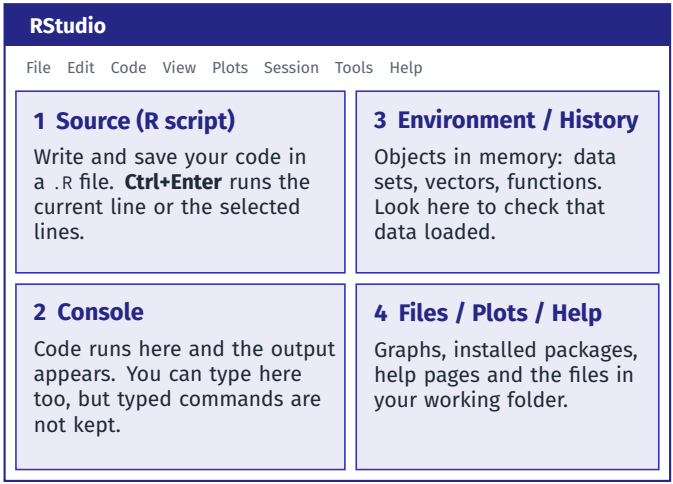
- Free and open-source software
- A full programming language: conditionals, loops, user-defined (even recursive) functions, input and output
- Excellent graphics for data exploration and publication
- The Comprehensive R Archive Network (CRAN) stores identical, up-to-date copies of R, its packages and documentation on servers worldwide
- On 7 February 2019 CRAN listed **13,689 packages**; 38 *Task Views* group them by topic

More than a statistics package
R is best thought of as an *environment* in which statistical techniques are implemented. Anyone can add new methods as a package.

Packages:
`https://cran.r-project.org/web/packages/`

Task Views:
`https://cran.r-project.org/web/views/`

The four panes of RStudio



- Write in the script; use the console only for quick checks
- Start comments with #
- Keep the script and its data in one folder (an RStudio *project*)
- *Import Dataset* reads CSV, Excel, SPSS, SAS and Stata files and shows the code it used

Packages used in this session

- Base R already contains everything for today: `read.csv()`, `table()`, `chisq.test()`, `t.test()`, `aov()`, `lm()`, `plot()`
- **`arm`**: `display()` prints a compact summary of a fitted regression
- Install a package once (internet needed); load it in each new session

```
# install once
# install.packages("arm")
library(arm)
```

> **Tip** `library()` without arguments lists the packages installed on your computer.

Outline

1. Getting started
2. Vectors and data
3. Graphics
4. Categorical data
5. Comparing groups
6. Correlation and regression

R as a calculator

Operator	Meaning
+ -	add, subtract
* /	multiply, divide
^	power
%/%	integer division
%%	remainder (modulo)
( )	group: evaluated first

As in school algebra, powers come before * and /, which come before + and -.

```
2 + 2; 5 - 3; 5 / 2; 5^2

## [1] 4
## [1] 2
## [1] 2.5
## [1] 25

3 + 2 * 5; (3 + 2) * 5   # * before +; brackets first

## [1] 13
## [1] 25

7 %/% 2; 7 %% 2  # integer part, remainder

## [1] 3
## [1] 1
```

A data vector and its summary statistics

```
x <- c(2, 4, 3, 5, 7, 9, 15)  # c() combines
length(x); sum(x); mean(x); median(x)

## [1] 7
## [1] 45
## [1] 6.429
## [1] 5

var(x); sd(x); range(x)

## [1] 19.95
## [1] 4.467
## [1]  2 15

summary(x)    # quantile(x) gives the quartiles

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00    3.50    5.00    6.43    8.00   15.00
```

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

- `<-` assigns: read it as “gets”
- `var()` and `sd()` use the divisor $n-1$
- Mean 6.43 $>$ median 5: the value 15 pulls the mean up (right skew)

Vector arithmetic works element by element

- A function applied to a vector acts on *every* element: no loop needed
- `log()` is the natural log; `log10()`, `log2()` or the `base` argument give others
- `gamma()` is the gamma function: $\Gamma(n) = (n-1)!$
- `round(v, 2)` keeps two decimals; `sqrt()`, `exp()`, `sin()`, `cos()` work the same way

```
1 + x; 2 * x; x^2

## [1]  3  5  4  6  8 10 16
## [1]  4  8  6 10 14 18 30
## [1]   4  16   9  25  49  81 225

round(log(x), 2)               # natural log

## [1] 0.69 1.39 1.10 1.61 1.95 2.20 2.71

round(log(x, base = 10), 2)    # same as log10(x)

## [1] 0.30 0.60 0.48 0.70 0.85 0.95 1.18

gamma(x)

## [1]           1           6           2          24
## [5]         720       40320 87178291200
```

Sorting, ordering and extracting elements

```
sort(x); rev(sort(x))   # up, then down

## [1]  2  3  4  5  7  9 15
## [1] 15  9  7  5  4  3  2

order(x); x[order(x)]   # positions; sorted

## [1] 1 3 2 4 5 6 7
## [1]  2  3  4  5  7  9 15
```

sort or order? `sort()` returns the values; `order()` returns their positions, which can be used to reorder other columns too.

```
x[2]; x[c(2, 4)]        # pick elements

## [1] 4
## [1] 4 5

x[-2]; x[-c(1, 3)]      # drop elements

## [1]  2  3  5  7  9 15
## [1]  4  5  7  9 15

x[x > 5]; which(x > 5)  # by a condition

## [1]  7  9 15
## [1] 5 6 7
```

`[ ]` extracts; a minus sign drops; a condition keeps the `TRUE` ones.

Sequences and combining vectors

```
3:6; seq(from = 3, to = 10)

## [1] 3 4 5 6
## [1]  3  4  5  6  7  8  9 10

seq(from = 3, to = 10, by = 0.5)

##  [1]  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5
##  [9]  7.0  7.5  8.0  8.5  9.0  9.5 10.0

seq(from = 3, to = 5, length.out = 5)

## [1] 3.0 3.5 4.0 4.5 5.0

rep(c(2, 3), times = c(5, 4)); rep(1:3, 3)

## [1] 2 2 2 2 2 3 3 3 3
## [1] 1 2 3 1 2 3 1 2 3
```

```
x1 <- c(1, 2, 3, 4, 5, 6, 7)
y1 <- c(2, 1, 4, 3, 2, 5, 6)
c(x1, y1)              # one long vector

##  [1] 1 2 3 4 5 6 7 2 1 4 3 2 5 6

rbind(x1, y1)          # as rows

##    [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## x1    1    2    3    4    5    6    7
## y1    2    1    4    3    2    5    6

dim(cbind(x1, y1))     # as columns: 7 x 2

## [1] 7 2
```

Naming objects and naming a vector

- A name starts with a letter (or a dot not followed by a digit) and may contain letters, digits, . and _
- R is case sensitive: `x` and `X` are different objects
- Avoid names of existing functions such as `c`, `t` or `mean`
- The elements of a vector can carry names too

```
names(x) <- c("A", "B", "C", "D",
              "E", "F", "G")
x; x["D"]

##  A  B  C  D  E  F  G
##  2  4  3  5  7  9 15
## D
## 5
```

Data objects vectors, matrices, data frames, lists and factors.

Lists and factors

A **list** collects objects of different types and lengths.

```
data5 <- list(x = c(2, 5, 4),
              Name = c("X", "Y", "Z"),
              Median = 4)
str(data5)

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

data5$Name; data5[["Median"]]

## [1] "X" "Y" "Z"
## [1] 4
```

A **factor** stores a categorical variable with its levels.

```
group <- gl(2, 5, labels = c("batchA",
                             "batchB"))
group

##  [1] batchA batchA batchA batchA batchA
##  [6] batchB batchB batchB batchB batchB
## Levels: batchA batchB

levels(group); table(group)

## [1] "batchA" "batchB"
## group
## batchA batchB
##      5      5
```

Getting help in R

```r
help(mean)              # or ?mean
??correlation           # search all packages
help.search("correlation")
help(package = "MASS")
example(mean)           # run the examples
?swiss                  # describe a data set
```

- `?` opens the help page of a function you know by name
- `??` searches every installed package for a word
- `data()` lists the data sets that come with R
- `str()` shows the structure of any object

```r
str(sleep)   # a built-in data frame: 20 rows, 3 columns

## 'data.frame': 20 obs. of  3 variables:
##  $ extra: num  0.7 -1.6 -0.2 -1.2 -0.1 3.4 3.7 0.8 0 2 ...
##  $ group: Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ ID   : Factor w/ 10 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...
```

Importing and exporting a CSV file

- Enter data in Excel, one row per case and variable names in the first row; save as *CSV (comma delimited)*
- `read.csv()` reads it into a data frame
- `write.csv()` saves a data frame
- Use relative paths such as `data/...`; avoid hard-coded paths like `D:/...`

```
data1 <- read.csv("data/data1.csv")
data1

##    A  B C  D
## 1 10 12 1 14
## 2 25 24 2 36
## 3 36 36 5 69
## 4 14 85 9 58

write.csv(data1, "data/data1_copy.csv",
          row.names = FALSE)
copy1 <- read.csv("data/data1_copy.csv")
identical(data1, copy1)   # same data back?

## [1] TRUE
```

`data1.csv`: four illustrative variables A–D, four rows.

`apply()`: summaries by row or by column

```r
apply(data1, 1, mean)     # row means
```

```
## [1]  9.25 21.75 36.50 41.50
```

```r
apply(data1, 2, mean)     # column means
```

```
##     A     B     C     D
## 21.25 39.25  4.25 44.25
```

```r
apply(data1, 2, sd)
```

```
##      A      B      C      D
## 11.701 32.035  3.594 24.391
```

```r
apply(data1, 2, median)
```

```
##    A    B    C    D
## 19.5 30.0  3.5 47.0
```

> **`apply(X, MARGIN, FUN)`** `MARGIN = 1` works along rows, `MARGIN = 2` along columns; `FUN` is any function that takes a vector.

Shortcuts for the most common cases: `rowMeans()`, `colMeans()`, `rowSums()`, `colSums()`.

Writing your own function: CV and standard error

$$\text{CV} = 100\,\frac{s}{\bar{x}}\,\%, \qquad \text{SE}(\bar{x}) = \frac{s}{\sqrt{n}}$$

- CV compares spread across variables measured on different scales (positive data only)
- SE measures how precisely $\bar{x}$ estimates the population mean

```
cv <- function(v) 100 * sd(v) / mean(v)
se <- function(v) sd(v) / sqrt(length(v))
round(apply(data1, 2, cv), 1)

##    A    B    C    D
## 55.1 81.6 84.6 55.1

round(apply(data1, 2, se), 2)

##     A     B     C     D
##  5.85 16.02  1.80 12.20
```

Reading it C has the smallest SE (1.8) but the largest CV (84.6%): small values, yet large *relative* spread.

Outline

1. Getting started
2. Vectors and data
3. Graphics
4. Categorical data
5. Comparing groups
6. Correlation and regression

Graphics in base R

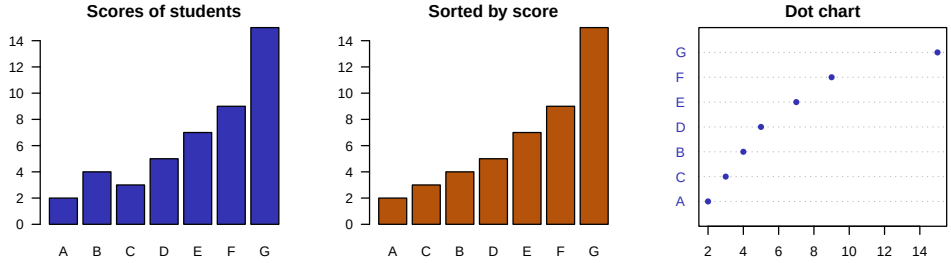
High level start a new plot	**Low level** add to the current plot
`plot()`, `hist()`	`points()`, `lines()`
`boxplot()`, `barplot()`	`abline()`, `text()`
`dotchart()`, `qqnorm()`	`grid()`, `legend()`
`pie()`, `curve()`	`qqline()`, `title()`

`par(mfrow = c(r, c))` puts $r \times c$ plots on one page; `par(mar = ...)` sets the margins.

```
length(colors())   # named colours

## [1] 657

colors()[c(26, 254, 552)]

## [1] "blue"  "green" "red"

# colours for today (hex codes)
col_main <- "#3333B3"   # blue
col_alt  <- "#B45309"   # amber
```

Bar plots and dot charts of named scores

```r
par(mfrow = c(1, 3), mar = c(4, 4, 2, 1))
barplot(x, las = 1, col = col_main, main = "Scores of students")
barplot(sort(x), las = 1, col = col_alt, main = "Sorted by score")
dotchart(sort(x), pch = 16, col = col_main, main = "Dot chart")
```



`las = 1` keeps axis labels horizontal. A dot chart shows the same information with less ink and stays readable with many categories.

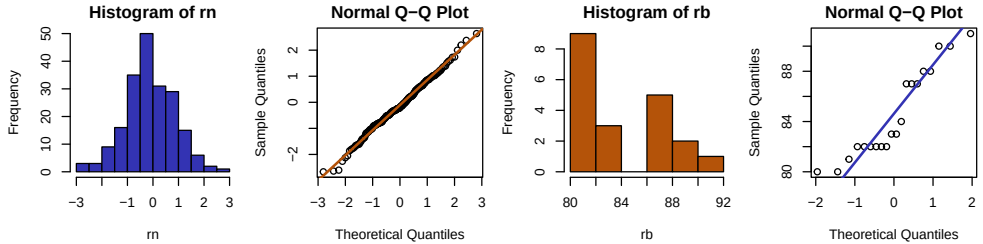
Building a plot layer by layer

```
x2 <- c(1, 2, 3, 4, 5, 6, 7)
y2 <- c(2, 1, 4, 3, 2, 5, 6)
par(mar = c(4, 4, 2, 1))
plot(x2, y2, type = "b", pch = 5, lty = 4,
     col = col_main, main = "My plot",
     xlab = "x axis", ylab = "y axis")
grid(col = "grey80")
text(2.5, 5, "this is my example plot")
abline(h = 2, v = 4, col = c(col_alt,
       "darkgreen"), lty = c(1, 4), lwd = 2)
```

`type`: `"p"` points, `"l"` lines, `"b"` both; `pch`: symbol; `lty`: line type; `lwd`: line width.

Is a sample normal? Histogram and normal Q–Q plot

```r
set.seed(2019)
rn <- rnorm(200); rb <- rbinom(20, 100, 0.85)  # N(0, 1) and Binomial(100, 0.85) samples
par(mfrow = c(1, 4), mar = c(4, 4, 2, 1))
hist(rn, col = col_main); qqnorm(rn); qqline(rn, col = col_alt, lwd = 2)
hist(rb, col = col_alt);  qqnorm(rb); qqline(rb, col = col_main, lwd = 2)
```



Points close to the line suggest normality. The binomial counts are whole numbers, so tied values form horizontal steps; with only 20 values the histogram is rough.

Outline

1. Getting started
2. Vectors and data
3. Graphics
4. Categorical data
5. Comparing groups
6. Correlation and regression

The sugar data

```
data4 <- read.table("data/data4.txt",
                    header = TRUE)
str(data4)

## 'data.frame': 20 obs. of  3 variables:
##  $ Sugar : chr  "no" "no" "yes" "yes" ...
##  $ Age   : chr  "old" "young" "old" "old" ...
##  $ Gender: chr  "female" "female" "male" "male" ...

head(data4, 3)

##   Sugar   Age Gender
## 1    no   old female
## 2    no young female
## 3   yes   old   male
```

- Hypothetical data on 20 patients
- `Sugar`: diabetic (yes/no); `Age`: old/young; `Gender`
- `read.table()` reads text files whose columns are separated by spaces; `header = TRUE` takes names from the first row

Question Is having sugar associated with age group, or with gender?

Frequency tables

```
table(data4$Sugar)

##
##  no yes
##  13   7

with(data4, table(Sugar, Gender))

##      Gender
## Sugar female male
##   no       6    7
##   yes      5    2
```

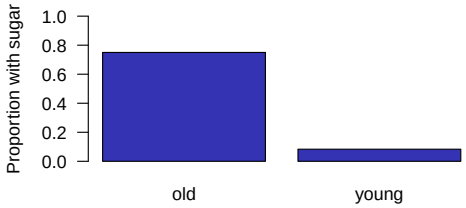
`with(data4, ...)` lets us write `Sugar` instead of `data4$Sugar`.

```
with(data4, table(Sugar, Age))

##      Age
## Sugar old young
##   no    2    11
##   yes   6     1

ftable(with(data4, table(Gender, Age, Sugar)))

##              Sugar no yes
## Gender Age
## female old          1   4
##        young        5   1
## male   old          1   2
##        young        6   0
```

Row proportions tell the story

```
tab <- with(data4, table(Age, Sugar))
addmargins(tab)                # with totals

##        Sugar
## Age     no yes Sum
##   old    2   6   8
##   young 11   1  12
##   Sum   13   7  20

round(prop.table(tab, 1), 2)  # 1 = rows, 2 = cols

##        Sugar
## Age       no  yes
##   old   0.25 0.75
##   young 0.92 0.08
```

```
p_yes <- prop.table(tab, 1)[, "yes"]
par(mar = c(4, 4, 1, 1))
barplot(p_yes, col = col_main, las = 1,
        ylim = c(0, 1),
        ylab = "Proportion with sugar")
```



75% of the old patients have sugar, against 8% of the young ones.

Chi-square test of independence, and a small-sample check

H_0: sugar status and age group are independent

$$X^2 = \sum \frac{(O - E)^2}{E}$$

E = row total $\times$ column total / n; df $= (r-1)(c-1)$

Rule of thumb The χ^2 approximation needs expected counts $E \geq 5$. Two are smaller here (R warns that the approximation may be incorrect), so use Fisher's exact test.

```
ct <- chisq.test(tab)
round(ct$expected, 1)      # expected counts

##         Sugar
## Age       no yes
##   old    5.2 2.8
##   young  7.8 4.2

round(ct$p.value, 4)       # Yates-corrected

## [1] 0.0098

ft <- fisher.test(tab)     # exact test
round(ft$p.value, 4)

## [1] 0.0044
```

Reading the categorical analysis

```
tab_g <- with(data4, table(Gender, Sugar))
ft_g <- fisher.test(tab_g)
```

- **Age:** Fisher's exact test gives $p = 0.0044$, well below 0.05. Sugar is associated with age group: 6 of 8 old patients against 1 of 12 young ones.
- **Gender:** the same test gives $p = 0.374$: no evidence of an association in these data.
- The chi-square test agrees for age ($p = 0.0098$), but its p-value is only approximate with so few patients.

What a p-value says The probability, if H_0 were true, of a table at least as far from independence as the one observed. It is not the probability that H_0 is true.

Caution Association is not causation, and 20 hypothetical patients cannot represent a population.

Outline

1. Getting started
2. Vectors and data
3. Graphics
4. Categorical data
5. Comparing groups
6. Correlation and regression

Two independent batches: the question

Scores of five students in each of two batches of the same course. Do the two batches have the same mean score?

$$H_0 : \mu_A = \mu_B \quad \text{vs} \quad H_1 : \mu_A \neq \mu_B$$

$$\text{Welch: } t = \frac{\bar{x}_A - \bar{x}_B}{\sqrt{s_A^2/n_A + s_B^2/n_B}}$$

Welch's test does not assume equal variances; its degrees of freedom are estimated from the data, so they need not be a whole number.

```
A1 <- c(20, 20.5, 13, 16, 11)  # batch A
B1 <- c(40, 42, 38, 56, 57)    # batch B
c(mean(A1), mean(B1))

## [1] 16.1 46.6

c(sd(A1), sd(B1))

## [1] 4.189 9.154
```

First look Batch B has the higher mean and the larger spread. Is the gap larger than chance variation between students would produce?

Welch's two-sample t-test in R

```
t.test(A1, B1)        # Welch's test is the default

##
##  Welch Two Sample t-test
##
## data:  A1 and B1
## t = -6.8, df = 5.6, p-value = 0.0007
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -41.71 -19.29
## sample estimates:
## mean of x mean of y
##      16.1      46.6
```

The output gives the statistic, its degrees of freedom and p-value, a 95% confidence interval for $\mu_A - \mu_B$ and the two sample means.

The pooled t-test, and what `t.test()` returns

Assuming $\sigma_A^2 = \sigma_B^2$:

$$s_p^2 = \frac{(n_A - 1)s_A^2 + (n_B - 1)s_B^2}{n_A + n_B - 2}$$

$$t = \frac{\bar{x}_A - \bar{x}_B}{s_p\sqrt{1/n_A + 1/n_B}} \sim t_{n_A+n_B-2}$$

The result is a list: `names(fit.t)` lists its ten parts, and `$` picks one out.

```
fit.t <- t.test(A1, B1, var.equal = TRUE)
fit.t$statistic; fit.t$parameter

##      t
## -6.774
## df
##  8

fit.t$p.value

## [1] 0.0001415

fit.t$conf.int

## [1] -40.88 -20.12
## attr(,"conf.level")
## [1] 0.95
```

Reading the two-sample tests

- Mean scores: batch A 16.1, batch B 46.6; difference -30.5 points
- Welch: $t = -6.77$, $p = 0.0007$; 95% CI for $\mu_A - \mu_B$: -41.7 to -19.3
- Pooled: $t = -6.77$ on 8 df, $p = 0.0001$, CI -40.9 to -20.1
- Both reject H_0: batch B scored higher on average

Which one? Welch is R's default and is safe whether or not the variances are equal (here $s_A = 4.2$, $s_B = 9.2$).

The interval We are 95% confident that batch B's mean exceeds batch A's by between 19 and 42 points; the interval excludes 0.

Paired t-test: if the same five students sat both tests

$$d_i = x_{Ai} - x_{Bi}, \quad H_0 : \mu_d = 0$$

$$t = \frac{\bar{d}}{s_d/\sqrt{n}} \sim t_{n-1}$$

Reading it Mean difference -30.5 points, $t = -5.81$ on 4 df, $p = 0.0044$. Pair only when the design pairs the data.

```
t.test(A1, B1, paired = TRUE)

##
##  Paired t-test
##
## data:  A1 and B1
## t = -5.8, df = 4, p-value = 0.004
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -45.09 -15.91
## sample estimates:
## mean difference
##           -30.5
```

The formula interface and a data frame

```
score <- c(A1, B1)
group <- gl(2, 5, labels = c("batchA",
                             "batchB"))
data6 <- data.frame(score, group)
head(data6, 3)

##   score  group
## 1  20.0 batchA
## 2  20.5 batchA
## 3  13.0 batchA

t.test(score ~ group, data = data6)$p.value

## [1] 0.0006718
```

- A **data frame** is a list of equal-length columns: one row per student
- `score ~ group` reads "score explained by group"
- Same Welch test, same p-value as before

> **Why bother?** The same formula style is used by `aov()`, `lm()`, `boxplot()` and many more.

Three batches: one-way analysis of variance

$$y_{ij} = \mu + \tau_i + \varepsilon_{ij}, \quad \varepsilon_{ij} \sim N(0, \sigma^2)$$
$$H_0 : \tau_A = \tau_B = \tau_C = 0$$
$$F = \frac{\mathsf{MS}_{\text{between}}}{\mathsf{MS}_{\text{within}}} \sim F_{k-1,\, n-k}$$

Comparing three means with three separate t-tests would inflate the chance of a false positive; ANOVA tests them together.

```
C1 <- c(10, 15, 12, 10, 3)     # batch C
data7 <- data.frame(
  score = c(A1, B1, C1),
  group = gl(3, 5, labels = c("batchA",
                    "batchB", "batchC")))
M2 <- aov(score ~ group, data = data7)
print(summary(M2), signif.stars = FALSE)

##             Df Sum Sq Mean Sq F value    Pr(>F)
## group        2   3845    1923    47.7 0.0000019
## Residuals   12    483      40
```

Group means and Tukey's multiple comparisons

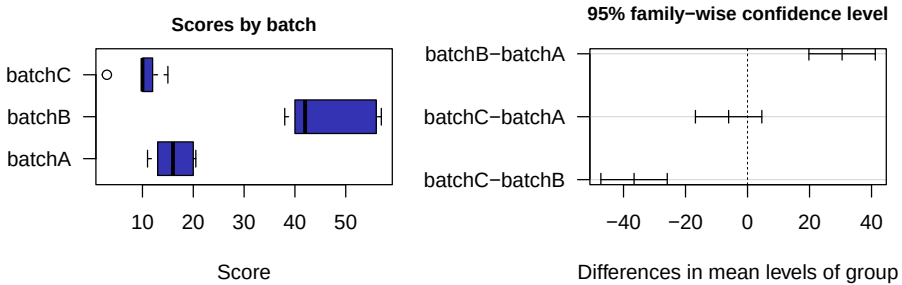
```
model.tables(M2, type = "means")

## Tables of means
## Grand mean
##
## 24.23
##
##  group
## group
## batchA batchB batchC
##   16.1   46.6   10.0
```

Tukey's HSD compares every pair of means while keeping the family-wise error rate at 5%.

```
TukeyHSD(M2, which = "group")

##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = score ~ group, data = data7)
##
## $group
##                diff    lwr     upr  p adj
## batchB-batchA  30.5  19.79  41.209 0.0000
## batchC-batchA  -6.1 -16.81   4.609 0.3167
## batchC-batchB -36.6 -47.31 -25.891 0.0000
```

Seeing the ANOVA

```
par(mfrow = c(1, 2), mar = c(4, 7, 2, 1), cex.main = 0.9)
boxplot(score ~ group, data = data7, col = col_main, las = 1, horizontal = TRUE,
        ylab = "", xlab = "Score", main = "Scores by batch")
plot(TukeyHSD(M2, which = "group"), las = 1)
```



Right: 95% family-wise intervals for each difference. An interval that crosses 0 (C–A) means those two batches cannot be distinguished.

Reading the ANOVA

- $F = 47.7$ on 2 and 12 df, $p < 0.001$: the three batch means are not all equal
- Tukey: B is higher than A by 30.5 and higher than C by 36.6 points (both adjusted $p < 0.001$)
- A and C differ by 6.1 points, adjusted $p = 0.32$: no evidence of a difference

In one sentence Batch B outperformed both other batches; batches A and C performed alike.

Assumptions Independent observations, roughly normal errors and similar spreads in the groups. With five students per batch, look at the box plots before trusting the test.

Outline

1. Getting started
2. Vectors and data
3. Graphics
4. Categorical data
5. Comparing groups
6. Correlation and regression

Correlation between marks

```
data8 <- read.table("data/data8.txt",
                    header = TRUE)
data8

##   batchA batchB batchC
## 1   20.0     40     10
## 2   20.5     42     15
## 3   13.0     56     12
## 4   16.0     38     10
## 5   11.0     57      3

round(cor(data8), 3)

##        batchA batchB batchC
## batchA  1.000 -0.830  0.696
## batchB -0.830  1.000 -0.476
## batchC  0.696 -0.476  1.000
```

Pearson's correlation coefficient

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

$$-1 \leq r \leq 1$$

Pairs needed Correlation needs two measurements on the *same* units. Read each row as one student's marks in three papers (illustrative data).

Testing a correlation

```
ct8 <- cor.test(data8$batchA, data8$batchB)
ct8$estimate; ct8$p.value

##     cor
## -0.8299
## [1] 0.08206

ct8$conf.int[1:2]     # 95% CI for rho

## [1] -0.9884  0.1956

cor(data8$batchA, data8$batchB,
    method = "spearman")

## [1] -0.6
```

$$H_0 : \rho = 0, \quad t = \frac{r\sqrt{n-2}}{\sqrt{1-r^2}} \sim t_{n-2}$$

Reading it $r = -0.83$ looks strong, yet $p = 0.082$ and the 95% CI for ρ runs from -0.99 to 0.2. With $n = 5$ we cannot rule out zero correlation. Spearman's rank correlation is -0.6.

Fitting a simple linear regression

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2)$$

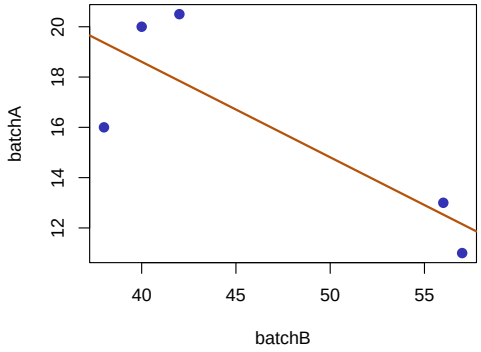
Least squares: choose $\hat{\beta}_0, \hat{\beta}_1$ to minimise $\sum (y_i - \hat{y}_i)^2$

`lm()` fits the model; `summary()` gives the full table and `display()` from **arm** a compact one.

```
M1 <- lm(batchA ~ batchB, data = data8)
display(M1, detail = TRUE)

## lm(formula = batchA ~ batchB, data = data8)
##             coef.est coef.se t value Pr(>|t|)
## (Intercept) 33.80     6.98    4.85    0.02
## batchB      -0.38     0.15   -2.58    0.08
## ---
## n = 5, k = 2
## residual sd = 2.70, R-Squared = 0.69
```

Reading the regression

```
par(mar = c(4, 4, 1, 1))
plot(batchA ~ batchB, data = data8, pch = 16,
     col = col_main, cex = 1.3)
abline(M1, col = col_alt, lwd = 2)
```



- Fitted line: $\widehat{\text{A}} = 33.8 - 0.38 \times \text{B}$
- Each extra mark in paper B goes with 0.38 fewer marks in paper A, on average
- $R^2 = 0.69$, but the slope has $p = 0.082$ and $r = -0.83$ is not significant at 5%

Honest conclusion Five points and 3 residual df cannot establish a relationship. Use this example to learn the commands, not to draw conclusions.

Key take-aways

1. **Work in a script**, inside an RStudio project, with relative paths.
2. **Vectors are the building blocks**: functions act on every element at once.
3. **Look at the data first**: tables, proportions and plots before tests.
4. **Check a test's conditions**: small expected counts call for Fisher's exact test.
5. **Let the design choose the test**: independent groups or paired measurements.
6. **Report effect sizes with intervals**, not only p-values, and remember that tiny samples give weak evidence.

References

- Dalgaard, P. (2008). *Introductory Statistics with R*, 2nd Ed., Springer, New York.
- Gelman, A., and Hill, J. (2007). *Data Analysis Using Regression and Multilevel/Hierarchical Models*, Cambridge University Press, Cambridge.
- Ihaka, R., and Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3), 299–314.
- Jones, O., Harden, S., and Crawley, M. J. (2022). *The R Book*, 3rd Ed., Wiley, Hoboken.
- Verzani, J. (2014). *Using R for Introductory Statistics*, 2nd Ed., CRC Press, Boca Raton.
- 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