Basics of R

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

National Workshop on Econometrics for Research in Economics (EFRIE 2022), 25–29 April 2022
Department of Economics, Central University of Haryana, Mahendergarh
28 April 2022 · **Lecture 1 of 2**

Plan of the session

1. Getting started: R, RStudio, help
2. Vectors and data frames
3. Writing functions: growth, NPV, MLE
4. Control flow, the apply family, simulation
5. Descriptive statistics and graphs

Data: US GNP, cross-country savings, US expenditure and the DAX index, all shipped with R.

By the end you can compute growth rates and index numbers with vectors; read data files and handle missing-value codes; write your own functions; use loops and `apply()`; simulate a sampling distribution; and summarise and plot economic data.

PART 1

Getting started

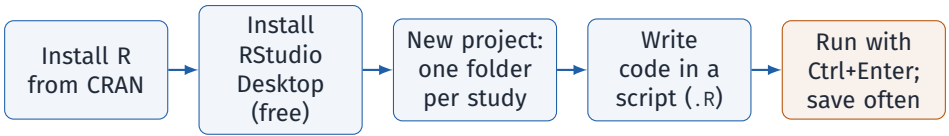
Why R for economic research?

- Free, open-source software for statistics, econometrics and graphics, on every platform
- About 19,000 add-on packages on CRAN (April 2022); the *Econometrics* Task View lists those for regression, panel data, time series and more
- Every step is code, so an analysis can be checked, rerun and shared
- Publication-quality graphs

A short history R grew out of S (John Chambers, Bell Labs, 1970s). Ross Ihaka and Robert Gentleman began R in Auckland in the early 1990s; version 1.0.0 appeared on 29 February 2000.

Limitations Data sit in memory (RAM), and there is a learning curve: no menus, just code.

From installation to a first script



RStudio pane	What it is for
Source	scripts: the permanent record of the analysis
Console	where code runs and results appear
Environment	objects currently in memory
Files, Plots, Packages, Help	project files, graphs, packages, documentation

`https://cran.r-project.org` and `https://posit.co/download/rstudio-desktop/`. Keep data in a `data/` sub-folder; use relative paths.

Packages used in this session

- Base R: vectors, data frames, functions, graphics and the data sets `longley`, `LifeCycleSavings`, `USPersonalExpenditure`, `EuStockMarkets`
- **moments**: `skewness()`, `kurtosis()`
- Install a package once; load it with `library()` in each session

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

# colours used in the graphs
col_main <- "#1F5FA8"
col_alt  <- "#B45309"
```

R as a financial calculator

```
10000 * (1 + 0.08)^5      # Rs 10,000 at 8% for 5 yrs
(259.4 / 234.3) - 1        # growth of a series
log(259.4) - log(234.3)    # log-difference
log(2) / log(1.07)         # years to double at 7%
70 / 7                     # the rule of 70
r <- 0.08                  # store a value
500 / (1 + r)^3            # present value of 500

## [1] 14693
## [1] 0.1071
## [1] 0.1018
## [1] 10.24
## [1] 10
## [1] 396.9
```

Compound growth:
$A = P(1 + r)^t$
Doubling time: $t = \ln 2 / \ln(1 + r)$
$\approx 70/(100r)$
Present value: $PV = F/(1 + r)^t$

- <- assigns a value to a name; R is case sensitive
- # starts a comment
- The log-difference is close to the growth rate when growth is small

Getting help

```
?lm                          # help page of lm()
help("read.csv")
??"unit root"               # search all help pages
help.search("time series")
example(diff)                # run the examples
?LifeCycleSavings          # data sets have pages too
```

```
args(diff)

## function (x, ...)
## NULL
```

- A help page has *Description*, *Usage*, *Arguments*, *Value* and *Examples*; the examples can be run as they are
- Built-in data sets are documented with their source: read the page before using the data
- CRAN Task Views (*Econometrics*, *TimeSeries*) point to packages by topic
- In RStudio, **F1** on a function name opens its help

PART 2

Vectors and data

Sequences for time indices

```r
2015:2022
seq(1950, 1970, by = 5)              # census-type years
seq(0, 1, length.out = 5)              # a grid
q <- rep(c("Q1", "Q2", "Q3", "Q4"), times = 2)
yr <- rep(2021:2022, each = 4)
paste(yr, q)                           # quarter labels

## [1] 2015 2016 2017 2018 2019 2020 2021 2022
## [1] 1950 1955 1960 1965 1970
## [1] 0.00 0.25 0.50 0.75 1.00
## [1] "2021 Q1" "2021 Q2" "2021 Q3" "2021 Q4" "2022 Q1"
## [6] "2022 Q2" "2022 Q3" "2022 Q4"
```

- `from:to` counts in steps of 1
- `seq()` takes a step (`by`) or a length (`length.out`)
- `rep()` repeats a vector (`times`) or each element (`each`)
- `paste()` joins text element by element

Use Year and quarter labels, plotting grids and group codes are all sequences.

A data vector: US GNP, 1947–1962

```
gnp <- longley$GNP                  # built-in data
names(gnp) <- longley$Year
length(gnp)
head(gnp, 6)
summary(gnp)

## [1] 16
##  1947  1948  1949  1950  1951  1952
## 234.3 259.4 258.1 284.6 329.0 347.0
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     234     318     381     388     454     555
```

Data `longley` (in base R) holds annual US macro data, 1947–1962: GNP, the GNP deflator (1954 = 100), employment and population. GNP is at current prices, in billions of dollars.

- `$` picks one column of a data frame
- `names()` labels each value with its year

Vectorised arithmetic: growth, real values, index numbers

```r
growth <- 100 * diff(log(gnp))        # % per year
round(growth, 1)
real <- 100 * gnp / longley$GNP.deflator
index <- 100 * real / real[1]         # 1947 = 100
round(index[c(1, 6, 11, 16)])

## 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958
## 10.2 -0.5  9.8 14.5  5.3  5.2 -0.6  9.0  5.3  5.5  0.4
## 1959 1960 1961 1962
##  8.2  4.0  3.1  6.8
## 1947 1952 1957 1962
##  100  125  145  168
```

$$g_t = 100\,(\ln Y_t - \ln Y_{t-1})$$
$$R_t = 100\,Y_t / P_t \quad (P_t\text{: deflator})$$
$$I_t = 100\,R_t / R_{1947}$$

No loops needed Each line works on all 16 years at once; `diff()` returns one value fewer. Real GNP reached 168% of its 1947 level by 1962.

Extracting, sorting and finding values

```
gnp["1955"]                    # by name
gnp[c(1, 16)]                  # by position
rgrowth <- 100 * diff(log(real))
rgrowth[rgrowth < 0]           # by condition
which.max(rgrowth)

##  1955
## 397.5
##  1947  1962
## 234.3 554.9
##    1949    1954    1958
## -0.1907 -1.6291 -1.7893
## 1950
##    3
```

```
round(sort(rgrowth,
           decreasing = TRUE)[1:3], 1)
order(rgrowth)[1:3]       # positions

## 1950 1955 1951
##  8.3  7.8  7.3
## [1] 11  7  2
```

Reading it Real GNP fell in 1949, 1954, 1958: the post-war US recessions. The fastest growth came in 1950.

Building a data frame

```
macro <- data.frame(year = longley$Year,
                    gnp = gnp, real = real,
                    unemp = longley$Unemployed)
rownames(macro) <- NULL
head(macro, 4)
macro$urate_proxy <- macro$unemp / longley$Population
```

```
##   year   gnp  real unemp
## 1 1947 234.3 282.3 235.6
## 2 1948 259.4 293.1 232.5
## 3 1949 258.1 292.6 368.2
## 4 1950 284.6 318.0 335.1
```

- A data frame is a table: columns are variables (of any type), rows are observations
- `$` adds a new column computed from others

Careful The last column is only a proxy: `Population` counts people aged 14 and over, not the labour force.

Reading CSV files

```
data1 <- read.csv("data/data1.csv")
data5 <- read.table("data/data5.csv",
                    header = TRUE, sep = ",")
dim(data1); dim(data5)
data1
```

```
## [1] 5 4
## [1] 4 4
##    A  B C  D
## 1 10 12 1 14
## 2 25 24 2 36
## 3 36 36 5 69
## 4 14 85 9 58
## 5 10 12 9 60
```

- `read.csv()` is `read.table()` with `sep = ","` and `header = TRUE` preset
- The first row supplies the variable names
- Here `data5` holds the first four rows of `data1`

```
# Excel files: package readxl
library(readxl)
d <- read_excel("data/file.xlsx", sheet = 1)
```

Missing-value codes in official statistics

```
# illustrative file; \n starts a new line
txt <- "r,inc,lit\nA,145.3,..\nB,n.a.,71.2\nC,149.3,-"
d <- read.csv(text = txt)
sapply(d, class)                    # text, not numbers!
d <- read.csv(text = txt,
              na.strings = c("..", "n.a.", "-"))
sapply(d, class)

##           r         inc         lit
## "character" "character" "character"
##           r         inc         lit
## "character"   "numeric"   "numeric"
```

- Statistical tables often mark gaps with `..`, `n.a.` or `-`
- One such entry turns a whole numeric column into text
- `na.strings` declares these codes as `NA`

Always check `str()` or `sapply(d, class)` right after reading: numeric variables must be `numeric` or `integer`.

A cross-country data frame: `LifeCycleSavings`

```
lcs <- LifeCycleSavings
head(lcs, 3)
lcs["India", ]
summary(lcs$sr)

##              sr pop15 pop75  dpi ddpi
## Australia 11.43 29.35  2.87 2330 2.87
## Austria   12.07 23.32  4.41 1508 3.93
## Belgium   13.17 23.80  4.43 2108 3.82
##       sr pop15 pop75   dpi ddpi
## India  9 41.31  0.96 88.94 1.54
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.60    6.97   10.51    9.67   12.62   21.10
```

Data 50 countries, averages for 1960–1970 (see `?LifeCycleSavings`): `sr` savings as % of disposable income, `pop15` and `pop75` % of population under 15 and over 75, `dpi` real per-capita disposable income, `ddpi` its % growth rate.

Life-cycle hypothesis: more young or old people, less saving. We look at this at the end.

Subsetting and ordering a data frame

```
fast <- subset(lcs, ddpi > 7,
               select = c(sr, dpi, ddpi))
rownames(fast)
top <- lcs[order(lcs$sr, decreasing = TRUE), ]
head(top[, c("sr", "dpi")], 3)

## [1] "Japan"       "Malta"       "Netherlands"
## [4] "Portugal"    "Jamaica"     "Libya"
##            sr    dpi
## Japan   21.10 1257.3
## Zambia  18.56  138.3
## Denmark 16.85 2496.5
```

- `data[rows, columns]`: a condition picks rows, names pick columns
- `subset()` does the same with less typing
- `order()` gives the row order that sorts by a column

Reading it 6 countries grew faster than 7% a year; Japan, with the highest savings ratio (21.1%), is among them.

PART 3

Writing functions

Anatomy of a function: compound annual growth

```r
cagr <- function(start, end, years) {
  rate <- (end / start)^(1 / years) - 1
  return(100 * rate)                  # in %
}
cagr(100, 200, 10)                    # check
# [[ ]] extracts one value, dropping its name
cagr(gnp[["1947"]], gnp[["1962"]], 15)
cagr(real[["1947"]], real[["1962"]], 15)

## [1] 7.177
## [1] 5.917
## [1] 3.526
```

$$\text{CAGR} = 100\left[(Y_T/Y_0)^{1/T} - 1\right]$$

- `function()` takes arguments, runs a body and returns a value
- Test on a known case: doubling in 10 years is about 7.2% a year

Reading it Nominal GNP grew 5.9% a year, real GNP 3.5%: the gap is inflation.

Standard error of a mean, with missing values

```
sr <- lcs$sr
sr[c(5, 20)] <- NA          # pretend two are missing
mean(sr)                     # NA spreads
se_mean <- function(x) {
  n <- sum(!is.na(x))        # usable observations
  c(n = n, mean = mean(x, na.rm = TRUE),
    se = sd(x, na.rm = TRUE) / sqrt(n))
}
round(se_mean(sr), 3)

## [1] NA
##       n    mean      se
## 48.000  9.618  0.657
```

$\mathrm{SE}(\bar{x}) = s/\sqrt{n}$, with n the number of non-missing values

- `na.rm = TRUE` drops `NA`s inside `mean()` and `sd()`
- `length()` would count the `NA`s and give the wrong n
- `c()` with names returns several results at once

Maximum likelihood: are DAX returns normal?

```
dax <- EuStockMarkets[, "DAX"]      # daily index
ret <- 100 * diff(log(dax))         # % log-returns
normal_mle <- function(x) {
  n <- length(x)
  c(mu = mean(x),
    sigma = sqrt(var(x) * (n - 1) / n))
}
round(normal_mle(ret), 4)
length(ret)

##     mu  sigma
## 0.0652 1.0298
## [1] 1859
```

MLEs for $N(\mu, \sigma^2)$:
$\hat{\mu} = \bar{x}, \quad \hat{\sigma}^2 = \frac{1}{n}\sum(x_i - \bar{x})^2$

Reading it Over 1859 trading days (1991–1998) the average daily return was 0.065%, with $\hat{\sigma} = 1.03$%. Whether a normal model fits is a separate question: see the Q-Q plot later.

A loop inside a function: net present value

```r
npv <- function(cf, rate) {
  pv <- 0
  for (t in seq_along(cf))       # one flow per pass
    pv <- pv + cf[t] / (1 + rate)^(t - 1)
  pv
}
cf <- c(-1000, 300, 400, 500)   # outlay, inflows
npv(cf, rate = 0.08)
uniroot(function(r) npv(cf, r), c(0, 1))$root

## [1] 17.63
## [1] 0.08895
```

$$NPV(r) = \sum_{t=0}^{T} \frac{CF_t}{(1+r)^t}$$

IRR solves $NPV(r) = 0$

Reading it At 8% the project is worth 17.63 today. `uniroot()`, which finds the zero of a function, gives the IRR: 8.9%.

PART 4

Control flow and apply

`if`, `else` and `ifelse()`

```
g <- lcs["India", "ddpi"]
if (g > 4) {
  "fast growth"
} else {
  "slow growth"
}
speed <- ifelse(lcs$ddpi > 4, "fast", "slow")
table(speed)

## [1] "slow growth"
## speed
## fast slow
##   15   35
```

Control structures

`if`/`else`: act on one condition
`ifelse()`: a condition for every element
`for`: repeat over a vector
`while`: repeat while a condition holds
`break`/`next`: leave / skip a pass

`if` needs one `TRUE`/`FALSE`; `ifelse()` works on a whole column.

`for` loops: growth year by year

```
n <- length(real)
g_loop <- numeric(n - 1)        # empty result
for (t in 2:n) {
  g_loop[t - 1] <- 100 * (real[t] / real[t - 1] - 1)
}
round(head(g_loop, 5), 2)
g_vec <- 100 * (real[-1] / real[-n] - 1)
all.equal(g_loop, unname(g_vec))

## [1]  3.85 -0.19  8.68  7.54  3.44
## [1] TRUE
```

- Create the result vector first, then fill it inside the loop
- `real[-1]` drops the first year, `real[-n]` the last: dividing them gives all ratios at once
- `all.equal()` confirms both routes agree

Simple growth vs log growth
$100(Y_t/Y_{t-1} - 1)$ is slightly larger than $100\,\Delta \ln Y_t$; the two are close when growth is small.

`while`: how long until income doubles?

```
income <- 100
years <- 0
while (income < 200) {
  income <- income * 1.07      # 7% growth
  years <- years + 1
}
years
round(income, 1)
log(2) / log(1.07)             # exact answer

## [1] 11
## [1] 210.5
## [1] 10.24
```

- `while` repeats as long as its condition is `TRUE`; use it when the number of passes is not known in advance
- Something inside the loop must change the condition, or it never stops

Reading it At 7% a year, income first reaches double its starting level after 11 whole years; the continuous answer is 10.24 years, close to the rule of 70.

`apply()`: US personal expenditure

```
pe <- USPersonalExpenditure     # $ billion, 5 x 5
rownames(pe) <- c("Food", "Household", "Medical",
                  "Personal", "Education")
apply(pe, 2, sum)               # total by year
round(100 * prop.table(pe, 2), 1)   # % shares

##   1940   1945   1950   1955   1960
##  37.61  68.71 102.56 129.70 163.14
##           1940 1945 1950 1955 1960
## Food      59.0 64.8 58.1 56.4 53.2
## Household 27.9 22.6 28.3 28.1 28.3
## Medical    9.4  8.4  9.5 10.8 12.9
## Personal   2.8  2.9  2.4  2.6  3.3
## Education  0.9  1.4  1.8  2.0  2.2
```

- `apply(X, 1, f)`: `f` on each row; `apply(X, 2, f)`: on each column
- `prop.table()` divides by the column totals

Reading it Food's share fell from 59% (1940) to 53.2% (1960) as spending grew (Engel's law); medical care and education gained share.

`lapply()` and `sapply()`

```
series <- list(real = real,
               employed = longley$Employed)
lapply(series, range)
sapply(series, function(v) cagr(v[[1]], v[[16]], 15))

## $real
## [1] 282.3 474.7
##
## $employed
## [1] 60.17 70.55
##
##     real employed
##    3.526    1.050
```

The apply family

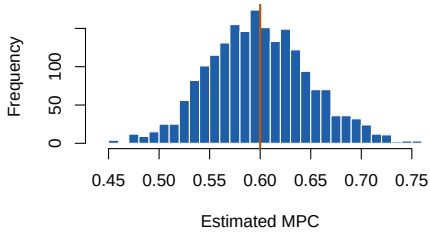
`apply()`: matrix rows or columns
`lapply()`: list elements; gives a list
`sapply()`: the same, simplified
`tapply()`: within groups

Functions can be written in place. Employment grew far more slowly than real GNP: output per worker rose.

Simulation: the sampling distribution of an OLS slope

```r
set.seed(2022)
income <- runif(50, min = 10, max = 50)   # fixed X
one_slope <- function() {
  cons <- 5 + 0.6 * income + rnorm(50, sd = 4)
  coef(lm(cons ~ income))[["income"]]
}
slopes <- replicate(2000, one_slope())
round(c(mean = mean(slopes), sd = sd(slopes)), 4)
4 / sqrt(sum((income - mean(income))^2))

##   mean     sd
## 0.5988 0.0498
## [1] 0.05054
```



A consumption function $C = 5 + 0.6\,Y + u$, $u \sim N(0, 4^2)$, estimated on 2000 simulated samples of 50 households.

What the simulation shows

$$\hat{\beta}_1 \sim N\left(\beta_1, \frac{\sigma^2}{\sum_i (x_i - \bar{x})^2}\right)$$ under the classical assumptions

- Unbiased: the slopes average 0.5988, against the true MPC of 0.6
- Precision: their SD, 0.0498, matches the formula's 0.0505
- Shape: the histogram is bell-shaped around 0.6

Why it matters The standard errors, t-tests and confidence intervals that `lm()` reports describe exactly this distribution. Lecture 2 uses them.

Ingredients A function for one replication, `replicate()` to repeat it, and a visible `set.seed()` so anyone can reproduce the numbers.

PART 5

Statistics and graphs

Location: mean, median, trimmed mean

```
dpi <- lcs$dpi
mean(dpi)
median(dpi)
mean(dpi, trim = 0.1)      # 10% off each end
rownames(lcs)[which.max(dpi)]

## [1] 1107
## [1] 695.7
## [1] 980.9
## [1] "United States"
```

Reading it Mean income (1107) is well above the median (696): a few rich countries pull the mean up. For skewed variables such as income, report the median.

Trimmed mean `trim` is a fraction between 0 and 0.5 of observations dropped from *each* end, not a count.

The mode: a function of our own

```
stat_mode <- function(x) {
  tab <- table(x)
  names(tab)[tab == max(tab)]     # all modes
}
table(state.region)               # US states
stat_mode(state.region)
hh <- c(4, 5, 4, 6, 3, 4, 5, 7, 5, 2)
stat_mode(hh)                     # two modes

## state.region
##     Northeast         South North Central          West
##             9            16            12            13
## [1] "South"
## [1] "4" "5"
```

- Base R has no function for the statistical mode (`mode()` reports the storage type)
- The mode suits categorical data: the most common region, occupation or crop
- Returning *all* maxima reveals ties: `hh` (illustrative household sizes) is bimodal

Spread and shape

```
cv <- function(x) 100 * sd(x) / mean(x)
round(sapply(lcs, cv), 1)            # CV in %
round(sapply(lcs, skewness), 2)
round(c(skew = skewness(ret),
        kurt = kurtosis(ret)), 2)    # DAX returns

##    sr pop15 pop75   dpi  ddpi
##  46.3  26.1  56.3  89.5  76.4
##    sr pop15 pop75   dpi  ddpi
## -0.01  0.00  0.31  0.98  2.21
##  skew  kurt
## -0.55  9.28
```

$\text{CV} = 100\, s/\bar{x}$

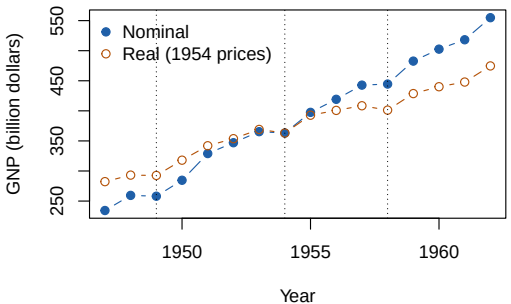
skewness $= m_3/m_2^{3/2}$

kurtosis $= m_4/m_2^2$ (3 for a normal)

Reading it Income (`dpi`) varies most (CV 90%) and is right-skewed. DAX returns have kurtosis 9.3: fat tails.

A time-series plot with annotations

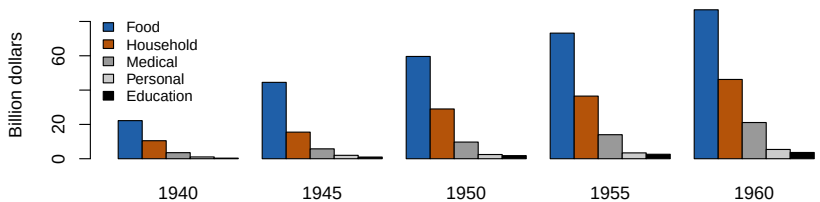
```
yr <- longley$Year
par(mar = c(4, 4, 1, 1))
plot(yr, gnp, type = "b", pch = 19,
     col = col_main, xlab = "Year",
     ylab = "GNP (billion dollars)")
lines(yr, real, type = "b", pch = 1,
      col = col_alt)
abline(v = c(1949, 1954, 1958), lty = 3)
legend("topleft", bty = "n",
       c("Nominal", "Real (1954 prices)"),
       col = c(col_main, col_alt),
       pch = c(19, 1))
```



`plot()` draws; `lines()`, `abline()`, `legend()` add to it. Dotted lines: recession years, when real GNP dips.

A grouped bar chart

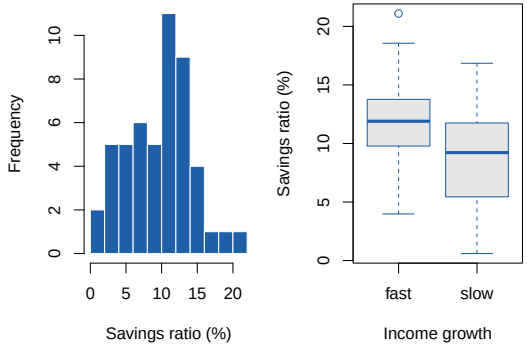
```
par(mar = c(3, 4, 1, 1))
barplot(pe, beside = TRUE, col = c(col_main, col_alt, "grey60", "grey80", "black"),
        ylab = "Billion dollars", legend.text = rownames(pe),
        args.legend = list(x = "topleft", bty = "n", cex = 0.8))
```



`beside = TRUE` puts categories side by side (default: stacked); all grew in current dollars.

Distribution of a variable: histogram and box plot

```
par(mfrow = c(1, 2), mar = c(4, 4, 1, 1))
hist(lcs$sr, breaks = 10, main = "",
     col = col_main, border = "white",
     xlab = "Savings ratio (%)")
boxplot(sr ~ speed, data = lcs, col = "grey90",
        border = col_main,
        xlab = "Income growth",
        ylab = "Savings ratio (%)")
```

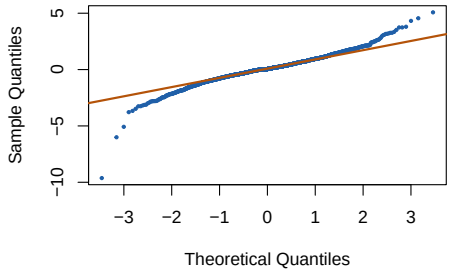


Reading it Savings ratios are fairly symmetric. Fast-growing countries (speed, from `ifelse()`) tend to save more: medians 11.9% against 9.2%.

Normal Q-Q plot: fat tails in stock returns

```r
par(mar = c(4, 4, 1, 1))
qqnorm(ret, main = "", pch = 19,
       cex = 0.4, col = col_main)
qqline(ret, col = col_alt, lwd = 2)
```

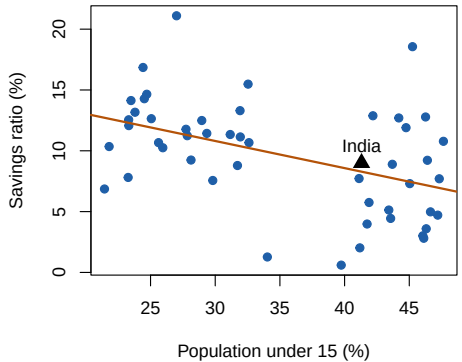
- Points follow the line if the data are normal
- Here both tails bend away: extreme days are far more common than a normal model allows



Reading it Shapiro–Wilk rejects normality (p <0.0001). A normal model understates the risk of extreme days.

Scatter plot: savings and the young population

```r
par(mar = c(4, 4, 1, 1))
plot(sr ~ pop15, lcs, pch = 19, col = col_main,
     xlab = "Population under 15 (%)",
     ylab = "Savings ratio (%)")
abline(lm(sr ~ pop15, lcs), col = col_alt, lwd = 2)
ind <- lcs["India", ]
points(ind$pop15, ind$sr, pch = 17, cex = 1.6)
text(ind$pop15, ind$sr, "India", pos = 3)
```



Reading it $r = -0.46$: countries with more children save less, as the life-cycle hypothesis predicts. How much less? Next lecture.

Key take-aways

1. **Work in a project with scripts** and relative paths: results can be rerun.
2. **Vectorise**: growth rates, real values and index numbers are one line each.
3. **Check data on arrival**: `str()`, `summary()`, and `na.strings` for codes such as `..` or `n.a.`
4. **Write functions** for repeated calculations (CAGR, NPV); test on a known case.
5. **Loops and the apply family** automate repetition; `replicate()` turns a function into a simulation study.
6. **Summaries and graphs together**: skewed incomes, fat-tailed returns and recessions are seen before they are tested.

Next session: regression analysis with `lm()`: estimation, inference and diagnostics.

References

- Dalgaard, P. (2008). *Introductory Statistics with R*, 2nd Ed., Springer, New York.
- Fox, J., and Weisberg, S. (2019). *An R Companion to Applied Regression*, 3rd Ed., Sage, Thousand Oaks.
- Jones, O., Harden, S., and Crawley, M. J. (2022). *The R Book*, 3rd Ed., Wiley, Hoboken.
- Wickham, H., Çetinkaya-Rundel, M., and Grolemund, G. (2023). *R for Data Science*, 2nd Ed., O'Reilly, Sebastopol. Free online: `https://r4ds.hadley.nz`
- R Core Team (2022). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna. `https://www.R-project.org`
- CRAN: `https://cran.r-project.org`; RStudio Desktop: `https://posit.co/download/rstudio-desktop/`

Thank you

Questions and discussion