STUDENT INSTALLATION AND LAB GUIDE

Getting Started with R and RStudio

Installation, basic coding, Excel import, and exploratory analysis with UrbanMart

1	2	3	4
INSTALL	SET UP	CODE	EXPLORE

Prepared by Prof Soudeep Deb • August 2026

Start here

This guide assumes no prior R experience. Complete the installation test, place the two course files together, and then work through the UrbanMart script one numbered section at a time.

> **The one-line mental model**
> R performs the analysis; RStudio is the workspace that makes R easier to write, run, organise, and inspect. Install R first, then RStudio.

What you will be able to do

- Install the latest stable R and the open-source RStudio Desktop.
- Create a clean RStudio Project and run code from a saved script.
- Create objects, vectors, data frames, and handle missing values.
- Install and load the packages used in the course.
- Read `UrbanMart_dataset.xlsx` and conduct exploratory analysis.

Three habits to learn immediately

Habit	Why it matters
Scripts are the record	Type important work in a saved `.R` file, not only in the Console.
Keep the two course files together	For this starter script, save the `.R` file and UrbanMart workbook in the same folder.
Inspect before analysing	Check names, types, missing values, ranges, and duplicates first.

1. Install R and RStudio

Before downloading

- Use only official R Project/CRAN and Posit download pages.
- Confirm whether your computer is Windows, macOS, or Linux and proceed accordingly.
- You may need administrator permission on an organisation-managed computer.
- Install R first. RStudio looks for an existing R installation when it starts.

> **Use the current stable releases**
> Download R and the open-source RStudio Desktop only from the official pages linked in Appendix B. Choose the current stable release shown there, not a daily or development build.

Windows

1. Open the official CRAN page for R on Windows and choose the current stable 64-bit installer.
2. Run the downloaded `.exe` file. For a first installation, accept the standard options and default location.
3. Open the Posit RStudio IDE download page and select the open-source Windows installer.
4. Run the RStudio installer, finish the setup wizard, and launch RStudio from the Start menu.

Compatibility note: current R for Windows uses 64-bit Windows and UCRT. Check the live RStudio support page for the operating systems supported by the latest desktop release.

macOS

1. Choose Apple menu > About This Mac. Note the macOS version and whether the chip is Apple M-series or Intel. Download the matching R `.pkg`: `arm64` for Apple silicon or `x86_64` for an Intel Mac from CRAN - R for macOS.
2. Open the `.pkg` and complete the installer. The standard installation is sufficient for this course.
3. Download the open-source RStudio `.dmg`, open it, and drag RStudio into Applications.
4. Launch RStudio from Applications. If macOS blocks the first launch, use System Settings > Privacy & Security to allow the signed application.

Learners on older Macs should use Posit's supported-version guidance or a browser-based R environment. XQuartz, Xcode, and a Fortran compiler are not needed for this beginner workflow.

Ubuntu Linux

If your organisation manages Linux centrally, ask IT to complete this section. CRAN publishes current repository commands on its official Ubuntu page.

```
sudo apt update -qq
sudo apt install --no-install-recommends software-properties-common dirmngr
wget -qO- https://cloud.r-project.org/bin/linux/ubuntu/marutter_pubkey.asc | \
  sudo tee -a /etc/apt/trusted.gpg.d/cran_ubuntu_key.asc
sudo add-apt-repository \
  "deb https://cloud.r-project.org/bin/linux/ubuntu $(lsb_release -cs)-cran40/"
sudo apt install --no-install-recommends r-base
```

Then download the current RStudio `.deb` for your supported Ubuntu or Debian release from Posit and install it with your system software manager. Do not copy an old installer filename from this guide.

Five-minute installation test

Open RStudio. Wait until the Console shows the > prompt. Then, enter the following lines one at a time.

```
R.version.string
1 + 1
getwd()
```

You should see an R version, the answer 2, and a folder path, with no red error message. If RStudio selected the wrong R version, open Tools > Global Options > General and change the R version.

2. Set up RStudio

Know the four panes

Pane	Use it for
Source	Write and save scripts. This is where most course work should live.
Console	Run commands and display results, warnings, and errors.
Environment	Show objects currently held in memory.
Output	Show files, plots, installed packages, help, and other results.

Create a project

1. Choose File > New Project > New Directory > New Project.
2. Name the folder, for example `Applied_Regression_R`, and choose a location you can find again.
3. Place `UrbanMart_dataset.xlsx` and `R_Basics_and_UrbanMart_EDA.R` inside the same folder.
4. Open the project by double-clicking its `.Rproj` file before each class.

```
Applied_Regression_R/
|-- Applied_Regression_R.Rproj
|-- UrbanMart_dataset.xlsx
|-- R_Basics_and_UrbanMart_EDA.R
`-- urbanmart_eda_outputs/    # created by the script
```

How this script finds the workbook

The supplied script uses `rstudioapi` to make the folder containing the active saved script the working directory. Open the `.R` file in RStudio, save it, and keep `UrbanMart_dataset.xlsx` in that same folder. This is not a hard-coded personal path.

Use a clean startup

In Tools > Global Options > General, clear "Restore `.RData` into workspace at startup" and choose "Never" for saving the workspace on exit. Your script and data, not an invisible saved workspace, should be able to recreate the analysis.

Run code deliberately

Action	Windows/Linux	macOS
Current line or selection	Ctrl + Enter	Cmd + Enter
Run the whole script	Source button	Source button
Stop a running command	Esc	Esc
Restart R	Ctrl + Shift + F10	Cmd + Shift + F10

For the first attempt, open the saved script in RStudio and run one numbered section at a time. Read every warning or error before moving on; the earliest error is usually the most useful.

3. Basic R coding

R is case-sensitive: `Sales`, `sales`, and `SALES` are three different names. Use clear lowercase names joined by underscores.

A compact syntax map

Idea	Example
Comment	`# explain why this step exists`
Assignment	`total_sales <- 2500`
Vector	`profit <- c(80, -25, 45, NA)`
Function	`mean(profit, na.rm = TRUE)`
Select a column	`urbanmart$profit`
Comparison	`urbanmart$profit > 0`
Pipe	`urbanmart |> dplyr::filter(profit > 0)`
Help	`?mean` or `help(mean)`

Objects, vectors, and missing values

```
course_sessions <- 5
hours_per_session <- 2
total_hours <- course_sessions * hours_per_session

profit <- c(120, -30, 85, 0, NA)
mean(profit, na.rm = TRUE)
sum(profit > 0, na.rm = TRUE)
```

`NA` means “missing”, not zero. Most summary functions return `NA` if a missing value is present; `na.rm = TRUE` removes missing observations from that calculation. Always decide whether that removal is sensible.

Data frames

```r
example_orders <- data.frame(
  order_id = c("A101", "A102", "A103"),
  sales = c(500, 750, 300),
  profit = c(80, -25, 45)
)

example_orders$profit
example_orders[example_orders$profit > 0, ]
```

Read code from the inside out

In `mean(profit, na.rm = TRUE)`, R first locates the object `profit`, then passes it to `mean()`, while the named argument tells `mean()` how to handle missing values.

4. Install and load libraries

In R, a library usually means the folder that stores packages. A package is a bundle of functions, documentation, and sometimes data.

Action	When	Example
Install	Usually once per computer or R library	`install.packages("readxl")`
Load	Once in every new R session	`library(readxl)`
Use explicitly	Any time; shows the package source	`readxl::read_excel(...)`

A reliable classroom pattern

```r
required_packages <- c(
  "readxl", "dplyr", "ggplot2",
  "janitor", "readr", "scales"
)

installed_names <- rownames(installed.packages())
missing_packages <- setdiff(required_packages, installed_names)

if (length(missing_packages) > 0) {
  install.packages(missing_packages, dependencies = TRUE)
}

invisible(lapply(required_packages, library, character.only = TRUE))
```

The UrbanMart script installs and loads `readxl`, `dplyr`, `ggplot2`, `janitor`, `readr`, and `scales`. Its working-folder line also uses `rstudioapi`. If R reports that `rstudioapi` is unavailable, run `install.packages("rstudioapi")` once, restart RStudio, and rerun the section.

If installation fails

- Check the internet connection and retry from a new R session.
- On a managed computer, a firewall or write restriction may require IT support.
- If R asks to install from source, choose the available binary for a beginner setup when offered.
- Copy the complete first error message; later messages often follow from it.

5. Read UrbanMart_dataset.xlsx

Keep the script and workbook together

The supplied script makes the saved script's folder the working directory and then looks for `UrbanMart_dataset.xlsx` there. Before importing, `file.exists(data_file)` must return `TRUE`.

```
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))

data_file <- "UrbanMart_dataset.xlsx"
file.exists(data_file)
```

Import and clean the column names

```
readxl::excel_sheets(data_file)

urbanmart_raw <- readxl::read_excel(
  path = data_file,
  sheet = 1,
  na = c("", "NA", "N/A")
)

urbanmart <- janitor::clean_names(urbanmart_raw)
```

`clean_names()` standardises names such as `Order.ID` or `Order ID` to `order_id`. The script then checks that the variables needed for the exercise are present before continuing.

Inspect before changing anything

```
dim(urbanmart)
names(urbanmart)
head(urbanmart, 6)
dplyr::glimpse(urbanmart)
summary(urbanmart)
```

Check	Question it answers
`dim()`	Did the expected number of rows and columns arrive?
`names()`	Are the expected variables present and spelled correctly?
`glimpse()`	Are dates, numbers, and text stored in sensible types?
`summary()`	Do ranges and missing values look plausible?

A repeated Order ID is not automatically a duplicate

UrbanMart is at the order-line level. One order can contain several products, so repeated `order_id` values may be valid. The script checks duplicate complete rows separately.

Course-file check. The supplied workbook should import as 9,994 order lines and 19 variables. If `dim(urbanmart)` differs, stop and confirm that you opened the correct workbook and first sheet.

6. Run a simple exploratory analysis

Exploratory analysis is a disciplined first look, not a hunt for a preferred conclusion. Move from data quality to overall performance, then to useful business groups and relationships.

The UrbanMart workflow

Stage	What the script does
1. Validate	Missing values, complete-row duplicates, numeric ranges, shipping time
2. Summarise	Order lines, distinct orders, sales, profit, margin, discount
3. Compare	Category, region, subcategory, and discount band
4. Visualise	Sales-profit relationship, category profit, discount margin
5. Save	CSV summaries and PNG charts in `urbanmart_eda_outputs`

Overall summary

```
overall_summary <- urbanmart |>
  dplyr::summarise(
    order_lines = dplyr::n(),
    distinct_orders = dplyr::n_distinct(order_id),
    total_sales = sum(sales, na.rm = TRUE),
    total_profit = sum(profit, na.rm = TRUE),
    overall_profit_margin = total_profit / total_sales,
    average_discount = mean(discount, na.rm = TRUE)
  )
```

A ratio of totals, `total_profit / total_sales`, answers a different question from the unweighted mean of line-level margins. For a business portfolio summary, the ratio of totals is a clearer starting point.

Grouped comparison

```
category_summary <- urbanmart |>
  dplyr::group_by(category) |>
  dplyr::summarise(
    order_lines = dplyr::n(),
    total_sales = sum(sales, na.rm = TRUE),
    total_profit = sum(profit, na.rm = TRUE),
    profit_margin = total_profit / total_sales,
    .groups = "drop"
  )
```

Visual question: how are discount and margin related?

```
ggplot2::ggplot(
  discount_summary,
  ggplot2::aes(x = discount_band, y = profit_margin)
) +
  ggplot2::geom_hline(yintercept = 0, colour = "#B33A3A") +
  ggplot2::geom_col(fill = "#D9A441") +
  ggplot2::scale_y_continuous(labels = scales::label_percent())
```

Questions to ask after running the script

1. Which category and region contribute the most total profit?
2. Are high-sales order lines always profitable?
3. At which discount bands does profit performance weaken?
4. Which subcategories deserve a closer commercial review?
5. What additional checks are needed before drawing causal conclusions?

> **Do not over-interpret the fitted line**
>
> The exploratory sales-profit chart includes a simple linear trend as a visual summary. It is not yet a validated causal or predictive model. Diagnostics and model design come next in the course.

7. Troubleshooting and good practice

Symptom	First response
Cannot open the connection / file does not exist	Run `getwd()` and `list.files()`. Confirm the saved script and the dataset are in that folder and that the filename matches exactly.
There is no package called …	Run `install.packages("package_name")` once, then `library(package_name)`.
Object … not found	Run the earlier lines that create the object; check spelling and capitalisation.
Could not find function …	Load the relevant package or use `package::function()`.
Unexpected symbol / + prompt	Look for an unclosed quote or bracket. Press Esc, correct the line, and rerun it.
Summary returns `NA`	Check missing values. Use `na.rm = TRUE` only when removing them is okay.
Dates become `NA`	Inspect the imported type and original format before selecting `as.Date()` format.
Permission denied	Use a folder you can write to or ask IT for access; avoid system folders.
There is no package called `rstudioapi`	Run `install.packages("rstudioapi")` once, restart RStudio, and rerun Section 3.
Cannot change working directory / path has length zero	Save and open `R_Basics_and_UrbanMart_EDA.R` in RStudio, then rerun the `setwd(...)` line.

A reproducible working rhythm

- Open the `.Rproj` file first, then open the saved course script from inside that project.
- Restart R and rerun the script from the top before submitting work.
- Keep original data read-only; create new objects for cleaned data.
- Save code and final outputs, not a cluttered `.RData` workspace.
- Record `sessionInfo()` when reproducibility matters.
- Never put passwords, API keys, or confidential data in a shared script.

Final checklist

- ☐ RStudio opens and `R.version.string` returns a version.
- ☐ The course RStudio Project opens from its `.Rproj` file.
- ☐ `UrbanMart_dataset.xlsx` is in the same folder as the saved course script.
- ☐ The supplied `.R` script opens in the Source pane.
- ☐ All six analysis packages load, and the `rstudioapi` working-folder line runs without an error.
- ☐ After a full run, CSV and PNG files appear in `urbanmart_eda_outputs`.

Appendix A. Using the UrbanMart starter script

The standalone `.R` file is the complete runnable version. The shorter code blocks in Sections 3–6 of this guide explain selected parts of it; run the full workflow from `R_Basics_and_UrbanMart_EDA.R` in RStudio.

Before running

Save and open `R_Basics_and_UrbanMart_EDA.R` in RStudio. Keep `UrbanMart_dataset.xlsx` in the same folder. In Section 3, continue only if `file.exists(data_file)` prints `TRUE`.

Script roadmap

Section	Purpose
1. Basic R coding	Calculator, objects, vectors, `NA`, logical tests, and a small data frame
2. Packages	Install missing packages and load all six required libraries
3. Locate and read	Find `UrbanMart_dataset.xlsx`, inspect sheets, and import the first sheet
4–5. Inspect and validate	Check dimensions, names, types, summary, and expected columns
6. Prepare variables	Convert dates and numbers; create shipping days, status, and margin
7. Data quality	Count missing values and duplicate complete rows; inspect ranges
8–10. Summarise	Overall, category, region, subcategory, and discount-band summaries
11. Visualise	Sales-profit, category-profit, and discount-margin charts
12. Save outputs	Write four CSV files and three 300-dpi PNG charts
13. Reflect	Five business and interpretation questions

First run

1. Open the RStudio Project, then open `R_Basics_and_UrbanMart_EDA.R` in the Source pane.
2. Confirm `UrbanMart_dataset.xlsx` is in the same folder as the saved R script.
3. Run Sections 1 and 2. The first package installation may take several minutes.
4. Run Section 3 and confirm `file.exists(data_file)` returns `TRUE`; then run Sections 4–5 and inspect the imported dimensions, names, and types.
5. Continue through the summaries and plots only after the validation checks pass.
6. Restart R and source the script from the top to confirm it is reproducible.

Outputs created by a successful run

Output	Contents
CSV summaries	`overall_summary.csv`, `category_summary.csv`, `region_summary.csv`, `discount_summary.csv`
PNG charts	`sales_vs_profit.png`, `category_profit.png`, `discount_margin.png`
Folder	`urbanmart_eda_outputs` beside the saved script and workbook

Before you finish

Restart R, source the complete script from the top, and confirm that the output folder contains four CSV summaries and three PNG charts. Read warnings before treating the run as complete.

Appendix B. Official links and references

Software and operating-system support change. Use these official live pages when installing or troubleshooting, and choose the current stable release shown on each page.

- The R Project for Statistical Computing - what R is and access to CRAN mirrors.
- CRAN - R for Windows - current stable Windows installer and platform notes.
- CRAN - R for macOS - current Apple silicon and Intel installers and macOS requirements.
- CRAN - Ubuntu packages for R - current supported Ubuntu releases and repository commands.
- Posit - RStudio IDE User Guide and downloads - current open-source RStudio Desktop installers and system support.
- Posit - Get Started with RStudio - RStudio panes, Projects, blank-slate settings, and package workflow.
- readxl - read_excel() reference - official arguments and behaviour for importing `.xls` and `.xlsx` files.

When you need help

Send the instructor the first complete error message, the section number, and the output of `getwd()`, `list.files()`, and `sessionInfo()`. Do not send passwords, confidential data, or screenshots that reveal personal folders unnecessarily.