- Start a new RStudio Project (.Rproj) → keeps everything relative to project root.
- {here} builds portable paths:
  library(here)
  read.csv(here("data", "raw", "file.csv"))
- Avoid hardcoding like “C:/Users/…/file.csv”.
- Jupyter Notebook:
  - Code cells, markdown cells, outputs.
  - Metadata (kernelspec, language).
- R Markdown:
  - YAML header (title/author/output).
  - Text (markdown).
  - Code chunks (R, Python, etc.) with output.
- Plain text formatting embedded with code.
- Common syntax:
  - Heading, ## Subheading
  - bold, italic, inline code
  - Lists: - item, 1. numbered
  - Links: text
- Used in both Jupyter (Markdown cells) & R Markdown (text sections).
- R Markdown (chunks):

  ```{r chunk-name, echo=TRUE}
  summary(cars)
  ```

- Jupyter (cells):
  - Run independently; no chunk options, but outputs inline.
- echo=FALSE → hide code, show results.
- eval=FALSE → show code, don’t run.
- include=FALSE → run but hide code+output.
- warning=FALSE, message=FALSE → suppress messages.
- fig.width=, fig.height= → control figure size.
- Write one sentence per line (not word-wrap at paragraph width).
- Makes version control diffs easier to read: This is the first sentence. This is the second sentence.
- Appears at top of .Rmd file, between —.
- Controls metadata + output type:

  title: “My Analysis”
  author: “Your Name”
  date: “r Sys.Date()”
  output: html_document —

- Quarto = evolution of R Markdown; works with R, Python, Julia.
- Supports .qmd files instead of .Rmd.
- Unified engine (Pandoc) → render to HTML, PDF, Word, slides, blogs, books.
- More flexible metadata & cross-referencing than R Markdown.
- Example render command: quarto render report.qmd –to html
- Reports: reproducible .qmd / .Rmd documents.
- Slides: presentations with reveal.js (Quarto/Jupyter).
- Blogs: Quarto blog sites (Markdown posts, themes).
- Books: long-form documents (Quarto Book or Jupyter Book).
- Websites: project documentation with Quarto.
- Jupyter Notebook → slides: jupyter nbconvert notebook.ipynb –to slides –reveal-prefix “https://revealjs.com”
- Quarto slides (reveal.js):
  title: “My Talk” format: revealjs — ## Slide 1 Some text — ## Slide 2 A plot
- Render project → _site/ folder.
- Push repo to GitHub → Settings → Pages → choose branch + folder.
- Quarto command for deployment: quarto publish gh-pages
- Computational environment = software, libraries, dependencies used to run a project.
- Ensures reproducibility: • Same package versions across machines. • Prevents “works on my machine” errors. • Captured in lockfiles (environment.yml, renv.lock).
- Tools: Conda, renv, Docker.
- Python = language & interpreter.
- Conda = package + environment manager (handles Python + system libraries).
- Anaconda = full distribution (~1,500+ preinstalled packages + Conda).
- Miniconda = lightweight installer with only Conda.
- pip = Python’s built-in package manager (pip install numpy).
- Create environment: conda create -n myenv python=3.10 numpy pandas
- Activate/deactivate: conda activate myenv conda deactivate
- Export environment for reproducibility: conda env export > environment.yml conda env create -f environment.yml
- Initialize project-local environment: renv::init()
- Install packages as usual (install.packages(“dplyr”)) → renv tracks versions.
- Save environment state: renv::snapshot()
- Restore on another machine: renv::restore()
- Lockfile (renv.lock) ensures exact reproducibility.
- Regex = text patterns for search & match.
- Functions: • R: grepl(), gsub(), stringr::str_detect() • Python: re.search(), re.findall(), re.sub()
- Example (Python): import re re.findall(r”“,”abc123xyz”) # [‘123’]
- Special chars: . = any character digit, = word char, = whitespace - = 1+, * = 0+, ? = 0/1
- Ranges: [a-z], [A-Z], [0-9], [aeiou]
- Anchors: ^ = start of string $ = end of string
- Example (R): stringr::str_detect(“abc123”, “[1]+$”) # TRUE
- Search (Python): re.search(r”dog”, “the dog runs”) # match
- Extract (R): stringr::str_extract(“price: $45”, “”) # “45”
- Replace/clean (Python): re.sub(r”+“,” “,”too many spaces”) # “too many spaces”
- Match filenames by pattern (Python): import glob files = glob.glob(“data/*.csv”)
- Regex filtering filenames (Python): import os, re [f for f in os.listdir(“data”) if re.search(r”202[0-9]-.*.csv”, f)]
- R example: list.files(“data”, pattern = “^202[0-9].*.csv$“)

[1] a-z