Capacity Building in Seasonal Hydrological Forecasting

Introduction to R and Programming Basics

AGRHYMET, Climate Regional Center for West-Africa and Sahel

@Arsène KIEMA

2025-10-08

Pedagogical Objectives

Learning outcomes

By the end of this session, you will be able to:

- Run R interactively (console, scripts, notebooks).
- Manipulate basic R objects: vectors, matrices, data frames, lists.
- Use simple functions and control structures (conditions, loops).
- Perform basic data manipulation with `dplyr`.
- Apply knowledge in a mini-project using hydrological flow data.

Syntax Rules in R

- **Case sensitive**: `Flow` `flow`
- **Assignment operators**:
- <- (recommended) or =
  - Example: `x <- 10`
- **Comments**: use # to add a comment
- **Functions**: always end with `()` even if no argument (`mean(x)`)
- **Vectors**: created with `c()`
- **Indexing**: `[ ]` to extract elements from vectors/data frames

Syntax Rules in R

Common pitfalls

- Forgetting quotes around text (`"River"` vs `River`).
- Mixing commas `,` and semicolons `;` (R only uses commas).
- Using reserved keywords (`if`, `else`, `TRUE`, `FALSE`) as variable names.

Naming Conventions

To keep code clear and readable, follow these guidelines:

- Use **lowercase_with_underscores** for variables.
  Example: `mean_flow`, `daily_precip`
- Use **UpperCamelCase** or verbs for functions.
  Example: `CalculateDischarge()`
- Names must start with a letter (not a number).
- Avoid spaces or special characters (#, %, -).
- Keep names **short but descriptive**:
  - Good: `q_obs` (observed discharge)
  - Bad: `qqqq123`

Naming Conventions

! Best practice

Consistent naming makes scripts easier to read and share.
Adopt one style (snake_case is common in R) and stick to it across your projects.

Variables and Assignment

In R, we use <- or = to assign values to variables:

```r
# Variable assignment
river_length <- 1250  # kilometers
watershed_area <- 2500  # square kilometers
mean_annual_discharge <- 45.6  # m³/s

# Display variables
river_length
```

```
[1] 1250
```

```r
watershed_area
```

```
[1] 2500
```

```r
mean_annual_discharge
```

Basic Data Types

R supports several fundamental data types:

- **Numeric**: real numbers (e.g. `3.14`, `-10`, `1000`)
- **Integer**: whole numbers (e.g. `1L`, `25L`)
- **Character**: text strings (e.g. `"River"`, `"Hydrology"`)
- **Logical**: Boolean values (`TRUE`, `FALSE`)
- **Date/Time**: calendar dates (`as.Date("2020-01-01")`)

Basic Data Types

```r
# Numeric (decimal numbers)
precipitation <- 125.6
cat("Precipitation:", precipitation, "mm\n")
```

```
Precipitation: 125.6 mm
```

```r
cat("Type:", typeof(precipitation), "\n\n")
```

```
Type: double
```

```r
# Integer (whole numbers)
number_stations <- 15L
cat("Number of stations:", number_stations, "\n")
```

```
Number of stations: 15
```

```r
cat("Type:", typeof(number_stations), "\n\n")
```

Basic Data Types

```r
# Character (text)
station_name <- "Station_Hydro_01"
cat("Station name:", station_name, "\n")
```

```
Station name: Station_Hydro_01
```

```r
cat("Type:", typeof(station_name), "\n\n")
```

```
Type: character
```

```r
# Logical (TRUE/FALSE)
is_operational <- TRUE
cat("Is operational:", is_operational, "\n")
```

```
Is operational: TRUE
```

```r
cat("Type:", typeof(is_operational), "\n")
```

Data Types in R

Note

Unlike some languages, R automatically converts between types if needed. But you can explicitly check/convert types: - `class(x)` $\rightarrow$ data type of `x` - `as.numeric()`, `as.character()`, `as.logical()`, etc.

Basic Arithmetic Operations

```r
# Hydrological calculations
daily_precipitation <- 25  # mm
catchment_area <- 150  # km²

# Convert precipitation to volume
precipitation_volume <- daily_precipitation * catchment_area * 1000
cat("Daily precipitation volume:", precipitation_volume, "m³\n")
```

```
Daily precipitation volume: 3750000 m³
```

Basic Arithmetic Operations

```r
# More calculations
annual_precipitation <- 1200  # mm
effective_runoff <- 0.35  # runoff coefficient
annual_runoff <- annual_precipitation * effective_runoff
cat("Annual runoff depth:", annual_runoff, "mm\n")
```

```
Annual runoff depth: 420 mm
```

```r
# Unit conversions
flow_rate <- 15.6  # m³/s
daily_volume <- flow_rate * 24 * 3600  # seconds in a day
cat("Daily water volume:", daily_volume, "m³\n")
```

```
Daily water volume: 1347840 m³
```

Data Structures in R

R provides several object types. The most common are:

- **Vector**: sequence of values of the same type.
- **Matrix**: 2D array of numbers.
- **Data frame**: tabular data, similar to Excel.
- **List**: heterogeneous collection (can contain data frames, vectors, etc.).

Data Structures in R

Vector

Vectors are the fundamental data structure in R:

```r
# Vector
flows <- c(12, 25, 33, 8, 54)
flows
```

```
[1] 12 25 33  8 54
```

Basic R Objects

```r
# Matrix
mat <- matrix(1:6, nrow = 2, ncol = 3)
mat
```

```
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Basic R Objects

Data Frames

Data frames are like Excel spreadsheets in R:

```r
# Create a hydrological data frame
hydrological_data <- data.frame(
  station = c("STA_North", "STA_South", "STA_East", "STA_West"),
  discharge_m3s = c(125.6, 89.3, 210.4, 156.8),
  catchment_area_km2 = c(250, 180, 320, 275),
  altitude_m = c(450, 280, 620, 380)
)
```

Basic R Objects

Data Frames

Data frames are like Excel spreadsheets in R:

```r
# Access specific columns
cat("\nDischarge values:\n")
```

```

Discharge values:
```

```r
hydrological_data$discharge_m3s
```

```
[1] 125.6  89.3 210.4 156.8
```

```r
cat("\nCatchment areas:\n")
```

```

Catchment areas:
```

```r
hydrological_data$catchment_area_km2
```

Basic R Objects

```r
# List
my_list <- list(station = "A", records = hydrological_data, area_km2
my_list
```

```
$station
[1] "A"

$records
    station discharge_m3s catchment_area_km2 altitude_m
1 STA_North         125.6                250        450
2 STA_South          89.3                180        280
3  STA_East         210.4                320        620
4  STA_West         156.8                275        380

$area_km2
```

Exercise 1: Basic Calculations

Calculate the following hydrological parameters:

1. A river has a mean discharge of 45 m^3/s. Calculate the annual volume in cubic meters.

2 .A catchment of 300 km^2 receives 800 mm of rainfall annually. Calculate the total volume of rainfall.

3. Convert a flow rate of 25 m^3/s to liters per second.

Exercise 1: Solution

```
# Solution space - try your code here!

# 1. Annual volume calculation
mean_discharge <- 45  # m³/s
seconds_per_year <- 365 * 24 * 3600
annual_volume <- mean_discharge * seconds_per_year
cat("Annual volume:", annual_volume, "m³\n")
```

Exercise 1: Solution

```
# 2. Rainfall volume calculation
catchment_area <- 300  # km²
annual_rainfall <- 800  # mm
rainfall_volume <- (annual_rainfall / 1000) * catchment_area * 1e6  #
cat("Annual rainfall volume:", rainfall_volume, "m³\n")
```

Exercise 1: Solution

```
# 3. Unit conversion
flow_rate <- 25  # m³/s
flow_liters <- flow_rate * 1000  # 1 m³ = 1000 liters
cat("Flow rate:", flow_liters, "L/s\n")
```

Control Structures

Conditions and loops allow automation in R.

```r
# Condition
Q <- 120
if (Q > 100) {
  print("High flow")
} else {
  print("Normal flow")
}
```

```
[1] "High flow"
```

Control Structures

Conditions and loops allow automation in R.

```
# Vectorized condition
State <- ifelse(hydrological_data$discharge_m3s > 130, "Above", "Belo
State
```

```
[1] "Below" "Below" "Above" "Above"
```

Control Structures

Conditions and loops allow automation in R.

```
# For loop
for (q in flows) {
  print(q * 2)
}
```

```
[1] 24
[1] 50
[1] 66
[1] 16
[1] 108
```

Functions

Functions make code reusable.

```r
# Example: specific discharge
specific_discharge <- function(Q, area_km2) {
  return(Q / area_km2)
}
specific_discharge(150, 300)  # 0.5 m³/s/km²
```

```
[1] 0.5
```

Data Manipulation with dplyr

dplyr is part of the **tidyverse**.
Key functions:

- filter() → filter rows
- select() → select columns
- mutate() → create new columns
- group_by() + summarise() → aggregation

```
library(dplyr)

df |>
  mutate(Class = ifelse(Qobs > 30, "Flood", "Normal")) |>
  group_by(Class) |>
  summarise(
    Mean = mean(Qobs),
    Max = max(Qobs)
```

Mini-project: Flow Data Analysis

Dataset

You have a CSV file `flows_station.csv` with two columns: - `Date` (YYYY-MM-DD) - `Qobs` (observed flow, m^3/s)

Tasks

1. Import the file.
2. Compute: annual mean, annual maximum, annual minimum.
3. Compute : Q25, Q75
4. Categorize each day:
   - Above if $Q > Q75$
   - Normal if $Q25 < Q$ Q75
   - Below if Q   Q25
5. Save results as `outputs/flows_categorized.csv`.

THANK YOU FOR YOUR ATTENTATION

CILSS