# Capacity Building in Seasonal Hydrological Forecasting

## Modeling and Machine Learning

**AGRHYMET, Climate Regional Center for West-Africa and Sahel**

**@Arsène KIEMA**

2025-10-06

# Pedagogical Objectives

**Learning outcomes**

By the end of this module, participants will be able to:

- Understand the theoretical foundations of four key models: PCR, Ridge, Lasso, and Random Forest.
- Explain how these models handle collinearity, regularization, and overfitting.
- Train and evaluate each model in R using hydrological data.
- Interpret model coefficients, variable importance, and performance metrics.

# Practical Implementation in R

We now apply these models to a simple rainfall–runoff dataset.

## Setting Up the Environment

```r
library(tidyverse)
library(tidymodels)
library(ranger)
library(rsample)
library(lubridate)
library(hydroGOF)
```

# Practical Implementation in R

## Load a sample hydro-climatic dataset

```r
data <- read.csv("data/hydro_features.csv") |>
  mutate(Date = as.Date(Date)) |>
  arrange(Date) |>
  drop_na()
```

# Practical Implementation in R

## Define rolling-origin resamples

We define training and testing periods that move forward in time.

```r
n_initial <- ceiling(nrow(data) * 0.7)
n_assess  <-floor(nrow(data) * 0.2)
n_skip    <- 1

ro <- rolling_origin(
  data,
  initial    = n_initial,
  assess     = n_assess,
  skip       = n_skip,
  cumulative = TRUE
)
```

# Practical Implementation in R

**Recipe design to avoid leakage**

All preprocessing (centering, scaling, PCA, etc.) is done **inside** the recipe. `fit_resamples()` and `tune_grid()` will re-estimate these steps **within each split**.

# Principal Component Regression (PCR)

PCR reduces correlated predictors into principal components before regression. This helps when rainfall, PET, and temperature are strongly related.

```
rec_pcr <- recipe(Qobs ~ Rain + Temp + PET, data = data) |>
  step_zv(all_predictors()) |>
  step_normalize(all_predictors()) |>
  step_pca(all_predictors(), num_comp = tune())

mod_pcr <- linear_reg() |> set_engine("lm")

wf_pcr <- workflow() |> add_recipe(rec_pcr) |> add_model(mod_pcr)

grid_pcr <- tibble(num_comp = 1:3)
```

# Principal Component Regression (PCR)

## Train PCR

Train and evaluate the PCR model across all time splits:

```r
set.seed(123)
res_pcr <- tune_grid(
  wf_pcr,
  resamples = ro,
  grid = grid_pcr,
  metrics = metric_set(yardstick::rmse, yardstick::rsq),
  control = control_grid(save_pred = TRUE)
)
show_best(res_pcr, metric = "rmse")
```

```
# A tibble: 3 x 7
  num_comp .metric .estimator  mean     n std_err .config
     <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
```

# Principal Component Regression (PCR)

## Train PCR

```
# A tibble: 3 x 7
  num_comp .metric .estimator  mean     n std_err .config
     <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1        3 rmse    standard    549.     3    20.1 pre3_mod0_post0
2        2 rmse    standard    610.     3    43.2 pre2_mod0_post0
3        1 rmse    standard    611.     3    36.9 pre1_mod0_post0
```

# Ridge Regression (L2 Regularization)

Ridge regression reduces overfitting by shrinking large coefficients.

```
mod_ridge <- linear_reg(penalty = tune(), mixture = 0) |> set_engine(

rec_ridge <- recipe(Qobs ~ Rain + Temp + PET, data = data) |>
  step_normalize(all_predictors())

wf_ridge <- workflow() |> add_model(mod_ridge) |> add_recipe(rec_ridg
```

# Ridge Regression (L2 Regularization)

## Train Ridge

```r
set.seed(123)
res_ridge <- tune_grid(
  wf_ridge,
  resamples = ro,
  grid = 20,
  metrics = metric_set(yardstick::rmse, yardstick::rsq),
  control = control_grid(save_pred = TRUE)
)

show_best(res_ridge, metric = "rmse")
```

# Ridge Regression (L2 Regularization)

## Train Ridge

```
# A tibble: 5 x 7
   penalty .metric .estimator  mean     n std_err .config
     <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1 1.21e-10 rmse    standard    549.     3    20.7 pre0_mod01_post0
2 3.71e-10 rmse    standard    549.     3    20.7 pre0_mod02_post0
3 1.00e- 9 rmse    standard    549.     3    20.7 pre0_mod03_post0
4 3.23e- 9 rmse    standard    549.     3    20.7 pre0_mod04_post0
5 1.07e- 8 rmse    standard    549.     3    20.7 pre0_mod05_post0
```

# Lasso Regression (L1 Regularization)

Lasso penalizes coefficients and sets some of them to zero (variable selection).

```
mod_lasso <- linear_reg(penalty = tune(), mixture = 1) |> set_engine(

rec_lasso <- recipe(Qobs ~ Rain + Temp + PET, data = data) |>
  step_normalize(all_predictors())

wf_lasso <- workflow() |> add_model(mod_lasso) |> add_recipe(rec_lass
```

# Lasso Regression (L1 Regularization)

## Train Lasso

```r
set.seed(123)
res_lasso <- tune_grid(
  wf_lasso,
  resamples = ro,
  grid = 20,
  metrics = metric_set(yardstick::rmse, yardstick::rsq),
  control = control_grid(save_pred = TRUE)
)

show_best(res_lasso, metric = "rmse")
```

# Lasso Regression (L1 Regularization)

## Train Lasso

```
# A tibble: 5 x 7
   penalty .metric .estimator  mean     n std_err .config
     <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1 1.21e-10 rmse    standard    549.     3    20.2 pre0_mod01_post0
2 3.71e-10 rmse    standard    549.     3    20.2 pre0_mod02_post0
3 1.00e- 9 rmse    standard    549.     3    20.2 pre0_mod03_post0
4 3.23e- 9 rmse    standard    549.     3    20.2 pre0_mod04_post0
5 1.07e- 8 rmse    standard    549.     3    20.2 pre0_mod05_post0
```

# Random Forest

Random Forest is an ensemble of trees, ideal for nonlinear rainfall–runoff relationships.

```
mod_rf <- rand_forest(
  mtry = tune(),
  min_n = tune(),
  trees = 500
) |>
  set_engine("ranger", importance = "permutation") |>
  set_mode("regression")

rec_rf <- recipe(Qobs ~ Rain + Temp + PET, data = data)

wf_rf <- workflow() |> add_model(mod_rf) |> add_recipe(rec_rf)
```

# Random Forest

## Training

```r
set.seed(123)
res_rf <- tune_grid(
  wf_rf,
  resamples = ro,
  grid = grid_rf,
  metrics = metric_set(yardstick::rmse, yardstick::rsq),
  control = control_grid(save_pred = TRUE)
)

show_best(res_rf, metric = "rmse")
```

# Random Forest

## Training

```
# A tibble: 5 x 8
   mtry min_n .metric .estimator  mean     n std_err .config
  <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1     3    15 rmse    standard    572.     3    57.6 pre0_mod14_pos
2     2    11 rmse    standard    572.     3    54.4 pre0_mod08_pos
3     2    15 rmse    standard    574.     3    55.7 pre0_mod09_pos
4     1     2 rmse    standard    575.     3    53.7 pre0_mod01_pos
5     3    20 rmse    standard    575.     3    53.3 pre0_mod15_pos
```

# Comparing Model Performances

We aggregate results from all models to identify the best one.

```
cv_tbl <- bind_rows(
  mutate(collect_metrics(res_pcr), model = "PCR"),
  mutate(collect_metrics(res_ridge), model = "Ridge"),
  mutate(collect_metrics(res_lasso), model = "Lasso"),
  mutate(collect_metrics(res_rf), model = "RandomForest")
)
```

# Comparing Model Performances

We aggregate results from all models to identify the best one.

```
# A tibble: 6 x 11
  num_comp .metric .estimator    mean     n std_err .config model
     <int> <chr>   <chr>        <dbl> <int>   <dbl> <chr>   <chr>
1        1 rmse    standard   6.11e+2     3 3.69e+1 pre1_m~ PCR
2        1 rsq     standard   5.17e-3     3 4.45e-3 pre1_m~ PCR
3        2 rmse    standard   6.10e+2     3 4.32e+1 pre2_m~ PCR
4        2 rsq     standard   5.47e-3     3 5.00e-3 pre2_m~ PCR
5        3 rmse    standard   5.49e+2     3 2.01e+1 pre3_m~ PCR
6        3 rsq     standard   4.89e-2     3 2.58e-2 pre3_m~ PCR
# i 1 more variable: min_n <int>
```

# THANK YOU FOR YOUR ATTENTATION