Hands-on: Accelerating k-NN with Rcpp

Overview

In this 30 min session you will:

1. Download and inspect the C++ implementation for k-NN.
2. Compile and load the Rcpp code.
3. Run benchmarks comparing pure-R vs Rcpp predictions.
4. Analyze how sample size and dimensionality affect performance.

Setup

Download the C++ and helper scripts into your working directory:

```
curl -O https://raw.githubusercontent.com/mmadoliat/WSoRT/refs/heads/main/src/knn_pred.cpp
curl -O https://raw.githubusercontent.com/mmadoliat/WSoRT/refs/heads/main/runthis.R
```

Open the files in your editor to review the code:

- **knn_pred.cpp** contains the `knn_pred_cpp()` function (Rcpp).
- **runthis.R** sources both R and C++ implementations and runs `microbenchmark()`.

---

1. Compile the C++ code

In an R console or RStudio, run:

```r
Rcpp::sourceCpp("knn_pred.cpp")
```

If successful, you should see `knn_pred_cpp` available:

```r
ls("package:base") # confirm knn_pred_cpp is loaded
# [1] "knn_pred_cpp"
```

---

2. Inspect the runner script

Open **runthis.R**, which contains:

```r
source("R/knn_s3_formula.R")              # loads knn_s3 and predict()
Rcpp::sourceCpp("src/knn_pred.cpp")       # loads Rcpp function

# Simulate data and benchmark
data <- simulate_knn_data(n = 1000, p = 5, m = 200, k = 10)
mb <- microbenchmark::microbenchmark(
  Rcpp = knn_pred_cpp(data$train_x, data$train_y, data$test_x, data$k),
  R    = knn_pred_R(data$train_x, data$train_y, data$test_x, data$k),
  times = 20
)
print(mb)
```

Try running this script:

```r
source("runthis.R")
```

---

3. Vary parameters

Modify **runthis.R** or re-run interactively to examine different settings:

- Increase **n** (training size) from 1000 to 5000 or 10000.
- Increase **p** (dimensions) from 5 to 20 or 50.
- Observe how the Rcpp version scales relative to pure-R.

Focus on how the Rcpp implementation stays much faster as complexity grows.

---

4. Discussion

- Where does Rcpp help most?
- Are there settings where pure R is sufficient?
- How might you further optimize (e.g., using STL partial_sort)?

Next steps

- Try integrating this into your `knn_s3` class and Shiny app.
- Explore parallel Rcpp implementations (OpenMP).
- Consider other statistical routines with nested loops.