Programmation Julia

Résumé de cours – 30 heures

Yaé Ulrich Gaba

Préface

Ce document est la version française abrégée du cours *Julia Programming*. La version anglaise complète – dix chapitres, code exécutable, exercices et projets – se trouve dans le fichier `../en/notes.pdf` du présent dépôt. La traduction française intégrale est prévue mais pas encore disponible.

Le cours couvre Julia depuis les fondamentaux du langage jusqu'au calcul scientifique et à l'apprentissage automatique : types, dispatch multiple, tableaux et algèbre linéaire, manipulation de données avec DataFrames, visualisation, optimisation de performance, calcul scientifique (équations différentielles via `DifferentialEquations.jl`), et apprentissage automatique avec `MLJ.jl` et `Flux.jl`.

Public visé : étudiants de licence ou de master en mathématiques, physique, ou science des données ayant une expérience préalable d'un langage comme Python, R, MATLAB ou C++. Prérequis mathématiques : algèbre linéaire et calcul différentiel au niveau licence.

Pourquoi Julia ?

Julia a été conçu pour résoudre le *problème des deux langages* : les chercheurs prototypent en langage de haut niveau (Python, MATLAB, R) puis réécrivent le code critique en termes de performance en C ou Fortran. Julia élimine cette étape : le même code, écrit dans un style proche de Python, atteint des vitesses comparables à C grâce à la compilation JIT via LLVM.

Trois caractéristiques définissent Julia :

- **Compilation JIT.** Les fonctions sont compilées en code machine natif lors de leur premier appel. Les appels suivants sont rapides.
- **Dispatch multiple.** Les fonctions sont spécialisées pour chaque combinaison de types d'arguments, ce qui remplace la programmation orientée objet par un modèle plus proche de la pensée mathématique.
- **Langage conçu pour le calcul scientifique.** Syntaxe mathématique naturelle, Unicode pour les variables (, ,), bibliothèques numériques et scientifiques de première classe.

Aperçu des chapitres

1. **Prise en main de Julia.** Installation, JuliaUp, REPL, gestionnaire de paquets `Pkg`, VS Code, Pluto, et premier programme exécutable.
2. **Types, variables et structures de contrôle.** Système de types, hiérarchie des nombres, structures composites, conditionnelles et boucles.
3. **Fonctions et dispatch multiple.** Définition de fonctions, méthodes, dispatch sur les types d'arguments, et son rôle central dans l'écosystème Julia.
4. **Tableaux et algèbre linéaire.** Tableaux multidimensionnels, indexation, opérations vectorisées, bibliothèque `LinearAlgebra` standard.
5. **DataFrames et manipulation de données.** `DataFrames.jl`, chargement de fichiers CSV, filtres, jointures, opérations `group-by`.
6. **Visualisation.** `Plots.jl` comme couche d'abstraction, `Makie.jl` pour les rendus de qualité publication, choix d'un back-end selon le contexte.
7. **Performance et optimisation.** Annotations de types, pièges classiques (variables globales, instabilité de type), profilage, benchmarking avec `BenchmarkTools.jl`.
8. **Calcul scientifique.** Équations différentielles ordinaires et stochastiques via `DifferentialEquati` optimisation, calculs symboliques avec `Symbolics.jl`.
9. **Apprentissage automatique avec Julia.** Écosystème `MLJ.jl` (modèles classiques), `Flux.jl` (apprentissage profond), différentiation automatique via `Zygote.jl`.
10. **Projets capstone.** Trois projets intégrés : pricing d'options avec Monte-Carlo, classification d'images MNIST, modélisation épidémiologique de la propagation d'une maladie infectieuse.

Pour aller plus loin

- **Version anglaise complète** : `../en/notes.pdf` (environ 200 pages avec code et exercices).
- **Documentation officielle Julia** : `https://docs.julialang.org`.
- **Forum communautaire** : `https://discourse.julialang.org`.
- **Cours gratuit du MIT** : *Introduction to Computational Thinking*, `https://computationalthinki` `mit.edu`.

Contact

Pour toute question, suggestion ou demande concernant la traduction complète, écrire à : `gabayae2@gmail.com`.