

Chapitre 1

Installation et découverte

1.1 Installation

Si vous êtes un utilisateur de Windows ou de macOS, le plus simple est de se rendre sur le site officiel de Julia <https://julialang.org/> dans la section des téléchargements, et de télécharger la dernière version stable (1.3.1 à l'heure où j'écris ces lignes).

Si vous êtes utilisateur d'une distribution Linux utilisant un gestionnaire de sources ou de paquets (par exemple Ubuntu, Debian, Gentoo), vous pouvez installer Julia à l'aide de ce gestionnaire. Notez toutefois que les versions des distributions ont souvent du retard. Je déconseille fortement d'utiliser une version de Julia inférieure à 1.0. Ainsi, les utilisateurs d'une version d'Ubuntu inférieure à 18.10 (Cosmic) doivent se tourner vers l'installation manuelle. L'installation manuelle se fait toujours avec un téléchargement sur le site de Julia.

Les binaires génériques Linux et FreeBSD ne nécessitent aucune étape d'installation particulière, mais vous devrez vous assurer que votre système peut trouver l'exécutable julia.

En supposant que le fichier `julia-1.3.1-linux-x86_64.tar.gz` a été téléchargé dans le dossier Téléchargements de votre répertoire utilisateur, vous pouvez taper dans un terminal

```
sudo tar -zxvf ~/Téléchargements/julia-1.3.1-linux-x86_64.tar.gz \  
--strip-components=1 -C /usr/local/
```

Si vous êtes familier avec l'application Snap, une alternative est d'installer Julia par ce biais <https://snapcraft.io/julia>.

1.3 Invite de commande

```
julia> log(ans)
5.0

julia> M=[1 1;1 0]
2x2 Array{Int64,2}:
 1  1
 1  0

julia> M^6
2x2 Array{Int64,2}:
 13  8
  8  5

julia> [M M; M zero(M)]
4x4 Array{Int64,2}:
 1  1  1  1
 1  0  1  0
 1  1  0  0
 1  0  0  0

julia> f(x)=x^2+1
f (generic function with 1 method)

julia> f(6)
37
```

On quitte l'interpréteur avec la commande `exit()` (bien noter les parenthèses).

À l'invite `julia>`, frapper la touche « ? » donne accès à une aide intégrée. La complétion automatique avec TAB est encore disponible.

```
help?> fo
foldl  foldr   for    foreach
help?> for
for    foreach
help?> for
search: for foreach foldr floor mapfoldr factorial EOFError
OverflowError

for

for loops repeatedly evaluate a block of statements while
iterating over a sequence of values.

Examples
-----
```

```
julia> for i in [1, 4, 0]
        println(i)
      end

1
4
0
```

Une astuce utile sous Linux : l'historique des sessions Julia est enregistré dans

`~/.julia/logs/repl_history.jl` (`~/.julia_history.jl` pour les anciennes versions).

1.4 IDE Juno

Juno est un environnement de développement intégré qui simplifie grandement l'écriture et la mise au point de programmes Julia.

C'est en réalité un mode de fonctionnement particulier pour l'IDE Atom, qu'il faut commencer par installer.

On le télécharge sur le site <https://atom.io/>. L'installation sous Windows ou Mac OS X n'amène pas de commentaire particulier.

1.4.1 Installation d'Atom sous Ubuntu

Dans un terminal, taper sur la ligne de commande

```
sudo apt-get install git libgconf-2-4
```

pour installer des dépendances dont atom aura besoin.

Télécharger ensuite sur le site <https://atom.io/> le paquet `atom-amd64.deb`, puis l'installer par

```
sudo dpkg -i Téléchargements/atom-amd64.deb
```

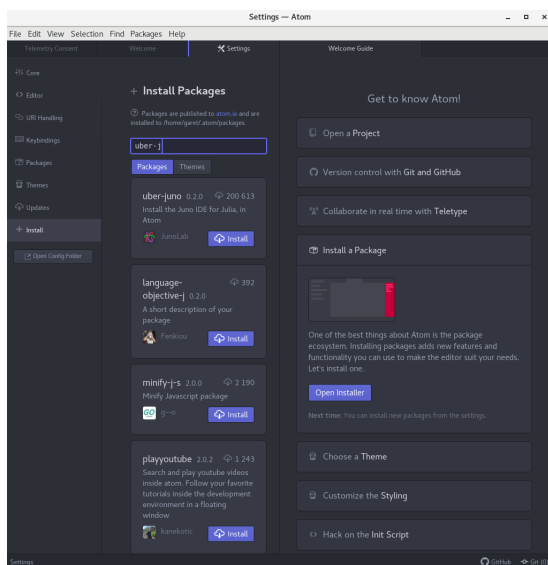
en supposant qu'il a été sauvegardé dans le répertoire Téléchargements.

1.4.2 Installation de Juno

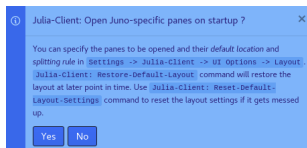
— Méthode 1 (méthode générale, tous systèmes)

Lancer atom. Cliquer sur Install a Package, puis Open Installer. Dans la barre de recherche, chercher uber-juno, puis cliquer Install.

1.4 IDE Juno



Un symbole d'interdiction apparait, puis l'installation commence. C'est un peu long. Ensuite, Juno propose d'être le mode par défaut au lancement d'Atom.



Cliquer sur Yes. Ensuite, il demande si vous voulez envoyer des statistiques aux développeurs. Faites votre choix. Appuyez enfin entrée dans la fenêtre Julia pour lancer la session et terminer l'installation (pré-compilation du module Atom pour Julia). Cette étape peut être un peu longue, particulièrement sous Windows. Ca y est ! L'installation est terminée.

— Méthode 2 (pour Linux) Taper dans un terminal






```
apm install uber-juno
atom
```

Au démarrage d'atom, l'installation de Juno commence automatiquement. La suite est comme dans la première méthode.

Par défaut, le thème installé est sombre (Atom Dark). Si vous souhaitez le modifier, il suffit de faire Edit/Preferences/Themes, puis de choisir votre thème préféré.

1.4.3 Utilisation

Dans la barre de gauche, des icônes permettent d'effectuer un certain nombre d'actions.

- Les icônes  et  permettent classiquement d'ouvrir ou de fermer un fichier.
- L'icône  permet de charger et d'exécuter de la fenêtre de code tandis que  fait la même chose pour un bloc de code sélectionné.
- L'icône  fait apparaître le panneau de l'outil de débogage, qui sera bien utile pour la mise au point de projets complexes.

Note : sur un écran de petite taille, il est possible que la barre d'icônes ne s'affiche pas complètement. Dans ce cas, positionner la souris, faire un clic droit, puis Position/Top.

1.5 Fichiers de programmes

Les fichiers de programmes Julia sont de simples fichiers textes, dont le nom se termine traditionnellement par l'extension « .jl ».

Pour charger et exécuter le programme `prg.jl`, il suffit de saisir dans l'interpréteur

```
include("prg.jl")
```

Le programme peut être lancé depuis un terminal (Bash par exemple) comme suit :

```
julia prg.jl
```

Dans ce cas, le terminal lance Julia, qui se charge, puis charge le programme, l'exécute (si il y a quelque chose à exécuter), puis rend la main.

On peut rendre la procédure transparente : si on ajoute au début du programme

```
#!/usr/local/bin/julia
```

puis qu'on le rend exécutable par

```
chmod +x prg.jl
```

1.6 Paquets

alors, taper en ligne de commande `./prg.jl` exécute le programme.

On peut également passer des paramètres au programme. Ainsi, si on a dans un fichier nommé `cml.jl` la ligne

```
println(ARGS)
```

alors en tapant en ligne de commande

```
julia cml.jl param1 param2
```

on obtient en retour

```
["param1", "param2"]
```

Ainsi, les paramètres sont passés dans un tableau de chaînes de caractères, que l'on peut analyser.

Ce mécanisme fait ainsi de Julia un langage de script.

Si on ne veut pas réinventer la roue pour l'analyse de la ligne de commande, on peut utiliser la bibliothèque **ArgParse**.

1.6 Paquets

1.6.1 Philosophie générale

Comme la plupart des langages modernes, Julia a vocation à être utilisé avec des bibliothèques de programmes qui vont compléter les possibilités offertes par le langage de base.

La philosophie générale est que le programme Julia installé sur votre ordinateur va, à votre demande, aller chercher une bibliothèque dans une archive web, puis en installer une copie dans votre répertoire personnel.

Parmi ces bibliothèques, certaines sont écrites en pur langage Julia, tandis que d'autres réalisent une interface entre Julia et une bibliothèque binaire ou écrite dans un autre langage interprété (Python ou Java, par exemple). Dans le deuxième cas, ces bibliothèques externes devront être installées sur le système avant l'installation de la bibliothèque d'interface.

Les bibliothèques de programmes de Julia sont appelées des paquets (packages en anglais). La collection des paquets installés sur l'ordinateur se fait à l'aide d'un gestionnaire de paquets appelé **Pkg**.

1.6.2 Les commandes

La manière la plus simple d'accéder au gestionnaire de paquets est de taper Alt Gr +] depuis la ligne de commande REPL. On fait touche retour arrière (Backspace) ou Control-C pour quitter le gestionnaire de paquets.

Voici quelques commandes du gestionnaire parmi les plus utiles :

- **add** Nom_de_Paquet installe le paquet Nom_de_Paquet et ses dépendances ;
- **rm** Nom_de_Paquet supprime le paquet Nom_de_Paquet ;
- **update** Nom_de_Paquet met à jour le paquet Nom_de_Paquet si il y a une version plus récente disponible ;
- **pin** Nom_de_Paquet épingle un paquet, c'est à dire qu'il est marqué comme ne devant pas être mis à jour ;
- **free** Nom_de_Paquet enlève l'épingle sur un paquet ;
- **update** met à jour tous les paquets du système qui n'ont pas été épinglés ;
- **gc** (*garbage collector*) fait le ménage en supprimant les paquets inutiles.

On peut utiliser les commandes de gestion de paquets sans entrer dans le mode du gestionnaire de paquets. Pour cela, il faut lancer la commande

```
using Pkg
```

pour charger² le paquet **Pkg**, puis appeler la commande voulue par `Pkg.nom_de_la_commande` suivi des arguments nécessaires.

1.6.3 Installation d'un paquet

Par exemple, pour installer le paquet `Plots`, on a deux méthodes :

- Ancienne Méthode :

```
using Pkg
Pkg.add("Plots")
```

- Méthode moderne : Alt Gr +] pour accéder au gestionnaire de paquets, puis **add** `Plots`, puis touche retour arrière (Backspace) ou Control-C pour quitter le gestionnaire de paquets.

². Il n'y a pas besoin d'installer le paquet **Pkg** ; c'est un paquet de la *Standard Library* (STDL) qui est déjà installée.

1.7 Autres éditeurs

1.6.4 Utilisation et résolution des problèmes

Un paquet qui a été installé peut être utilisé à l'aide de la commande

```
using Nom_du_paquet
```

Au premier lancement du paquet après son installation ou sa mise à jour, il n'est pas rare qu'un paquet amorce une phase de précompilation qui peut prendre un peu de temps.

Il peut arriver que le démarrage ou l'installation échoue car le système n'est pas dans un état approprié. Les messages d'erreur déplorent parfois l'absence de bibliothèques externes, pas installées ou pas accessibles. Mais les erreurs sont parfois dues à des incompatibilités de versions de différents paquets, le dépôt des bibliothèques étant en perpétuelle évolution.³ Dans ce cas, des combinaisons des commandes **update**, **rm**, **bg** et **add** peuvent suffire à régler le problème.

Le comportement par défaut de Julia est d'installer la version la plus récente du paquet demandé. Il peut arriver qu'une mise à jour casse la compatibilité de certains codes. Dans ce cas, on peut demander l'installation d'une version précise du paquet avec **add** `Nom_de_Paquet@x.y.z`, où `x.y.z` est la version souhaitée pour le paquet. Notons également le mot clef **preview**, qui, précédant le nom d'une commande, permet d'en voir les effets sans modifier le système.

On est très loin d'avoir présenté toutes les possibilités du gestionnaire de paquets, qui permet entre autres de travailler avec différents systèmes de paquets, et ainsi d'essayer des paquets en version expérimentale sans mettre en péril son installation. Pour plus de renseignement, on pourra se reporter à la documentation <https://julialang.github.io/Pkg.jl/v1/>.

1.7 Autres éditeurs

De mon point de vue, le couple Atom/Juno est aujourd'hui le meilleur environnement de développement pour Julia. D'autres solutions méritent toutefois d'être évoquées.

3. Les programmes contenus dans ce livre ont tous été testés avec Julia 1.3.1, avec les versions des paquets les plus récentes à la date de l'écriture, à savoir : AbstractAlgebra v0.8.0, ArgParse v0.6.2, Calculus v0.5.1, DataStructures v0.17.6, DifferentialEquations v6.9.0, LightGraphs v1.3.0, Plots v0.28.4, PyPlot v2.8.2, SpecialFunctions v0.9.0, SymPy v1.0.10.

1.7.1 Julia pour Jupyter Notebook

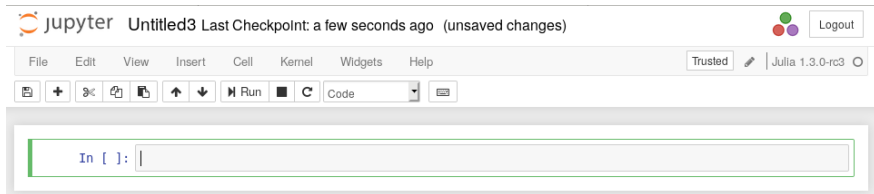
Jupyter Notebook est un éditeur de feuilles de calcul qui se lance dans un navigateur, permettant de présenter sur une même feuille lignes de calculs et résultats des commandes (y compris des graphiques). Pour installer Jupyter Notebook sous Ubuntu, il suffit d'installer le paquet jupyter

```
sudo apt-get install jupyter
```

Jupyter étant installé, il faut lancer julia et installer le paquet **Julia**, suivant la procédure décrite à la section précédente. L'installation est maintenant terminée. Pour lancer Jupyter Notebook, il faudra alors saisir dans un terminal

```
jupyter notebook
```

Jupyter s'ouvre alors dans un navigateur web. On peut alors créer une nouvelle feuille de calcul avec le menu « New » (choisir Julia parmi les langages proposés) ou ouvrir une feuille existante (un fichier d'extension ipynb). La feuille de calcul ressemble à ceci :



Les commandes saisies dans une cellule sont exécutées lorsque l'on clique sur Run ou que l'on tape Ctrl+Entrée. La croix permet de créer une nouvelle cellule.

1.7.2 Les autres

De nombreux éditeurs ont des modules permettant la prise en charge du langage Julia. On peut les retrouver à l'adresse

<https://github.com/JuliaEditorSupport>.

Cela inclut bien sûr les grandes vedettes : Emacs et Vi.