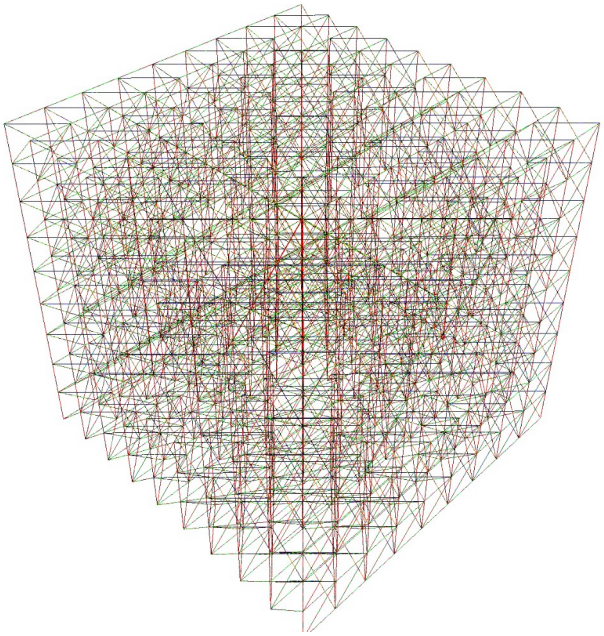
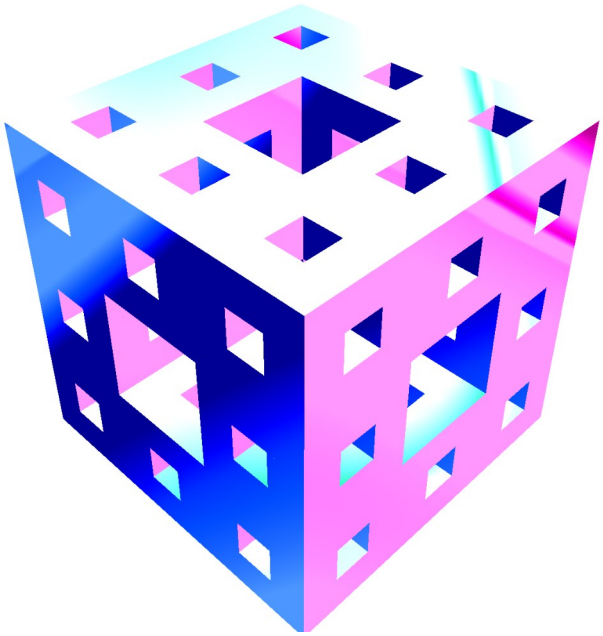
Julia et DB2 for i





VERSION 1.3

Table des matières

Versions :
1.0 publiée le 14/12/2022
1.1 publiée le 15/12/2022 : ajout de livres Packt Publishing dans le chapitre « Bibliographie »
1.2 publiée le 18/12/2022 : ajout du chapitre « Ressources internet complémentaires »
1.3 publiée le 18/12/2022 : ajout du chapitre « Dataviz en accéléré »

Auteur : Grégory Jarrige

Mise à jour : 14-12-2022
Couverture : « Menger Sponge » de niveau 2, générée avec Ogl.js
Repo : https://github.com/gregja/juliaworks

Introduction

J'ai découvert l'existence du langage Julia dans le courant de l'année 2022. Projet récent et Open source, développé par une équipe de chercheurs du MIT, Julia est annoncé comme étant, en résumé : "... aussi souple d'écriture que du Python, et presque aussi performant que du C...".

Cela m'a intrigué, d'autant que j'ai été confronté à plusieurs reprises à des problématiques de volumétrie - en lien avec des bases de données DB2 for i - pour lesquelles mes langages de prédilection (NodeJS et PHP) n'étaient pas vraiment adaptés. J'ai eu envie de tester Julia, et de m'assurer qu'il était possible de le faire dialoguer avec une base de données DB2 for i.

Dans ce dossier, j'explique comment installer le langage Julia sur Windows 10, et comment le connecter à une base de données DB2 for i.

L'installation de Julia se fait en plusieurs étapes :
- installation de Julia sur le PC
- configuration d'une variable d'environnement
- configuration d'un IDE pour pouvoir développer en Julia
- connexion de Julia à DB2 for i (via un driver ODBC)

C'est ce que nous allons voir dans la suite de ce dossier.

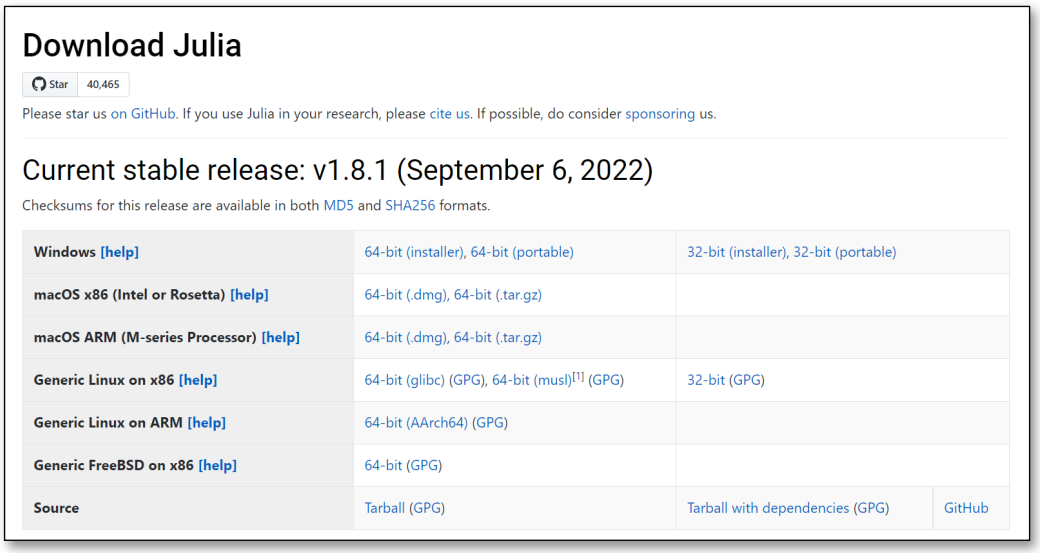
Nous verrons aussi quelques exemples de scripts Julia dialoguant avec DB2 for i, plus quelques exemples de scripts de visualisation de données qui pourront servir de base pour des représentations plus complexes.

1 - Télécharger et installer Julia

Installation

Lien de téléchargement :
[Download Julia (julialang.org)]

Plusieurs choix sont possibles, selon la plateforme cible.



Download Julia

Star 40,465

Please star us on GitHub. If you use Julia in your research, please cite us. If possible, do consider sponsoring us.

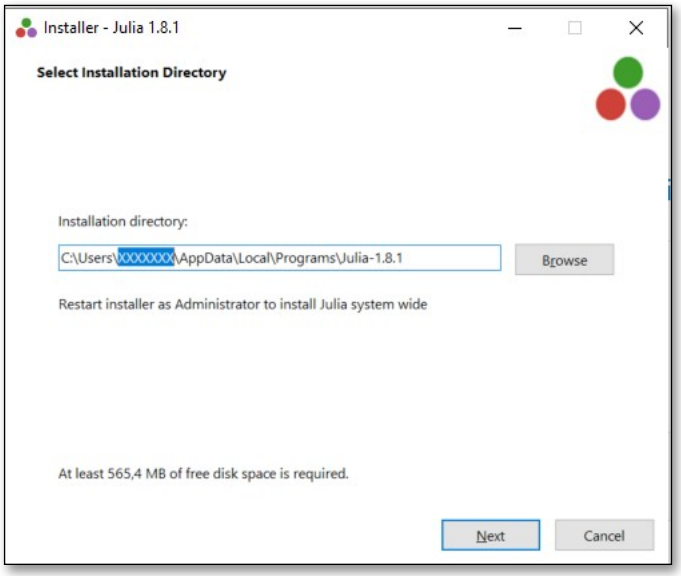
Current stable release: v1.8.1 (September 6, 2022)

Checksums for this release are available in both MD5 and SHA256 formats.

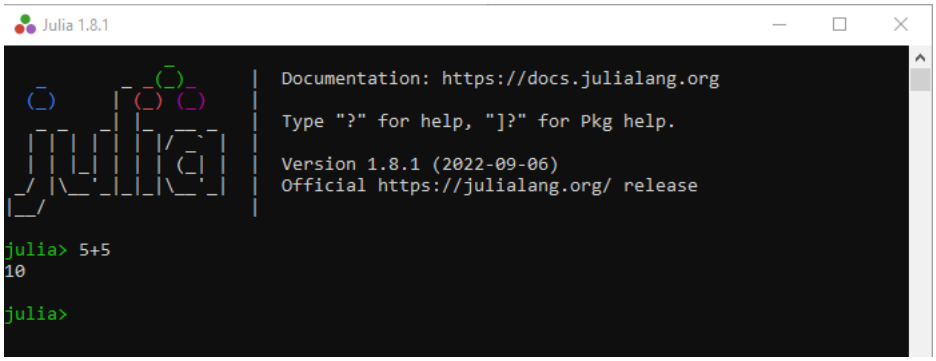
Windows [help]	64-bit (installer), 64-bit (portable)	32-bit (installer), 32-bit (portable)	
macOS x86 (Intel or Rosetta) [help]	64-bit (.dmg), 64-bit (.tar.gz)		
macOS ARM (M-series Processor) [help]	64-bit (.dmg), 64-bit (.tar.gz)		
Generic Linux on x86 [help]	64-bit (glibc) (GPG), 64-bit (musl)[1] (GPG)	32-bit (GPG)	
Generic Linux on ARM [help]	64-bit (AArch64) (GPG)		
Generic FreeBSD on x86 [help]	64-bit (GPG)		
Source	Tarball (GPG)	Tarball with dependencies (GPG)	GitHub

On notera que pour Windows, il existe une version « 64-bit Portable », pouvant être installée sans nécessiter de droits administrateurs.
Si on dispose des droits administrateurs, utiliser de préférence la version « 64-bit (installer) ».

Une fois l'installeur téléchargé, lancez-le et suivez les instructions.
L'installation sous Windows est très simple, on peut personnaliser le chemin d'installation si nécessaire :

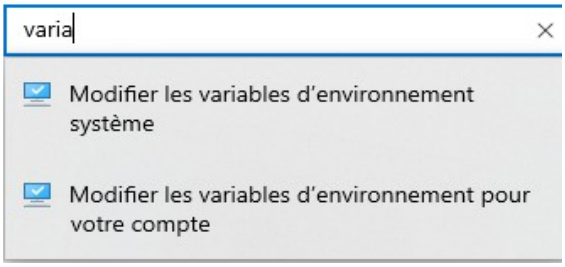


Une fois l'installation terminée, on peut immédiatement tester Julia :



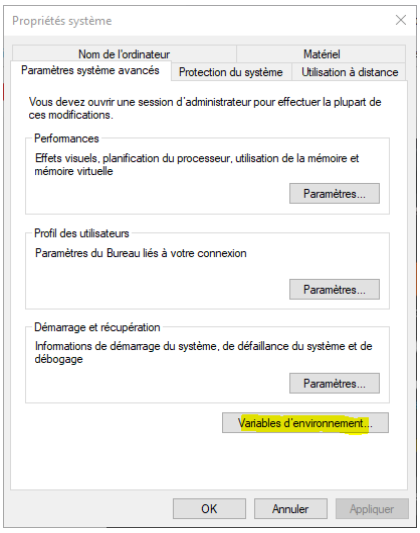
Variable d'environnement

Vous devez maintenant modifier les variables d'environnement de votre système. Recherchez une option qui s'intitule « modifier les variables d'environnement système » :



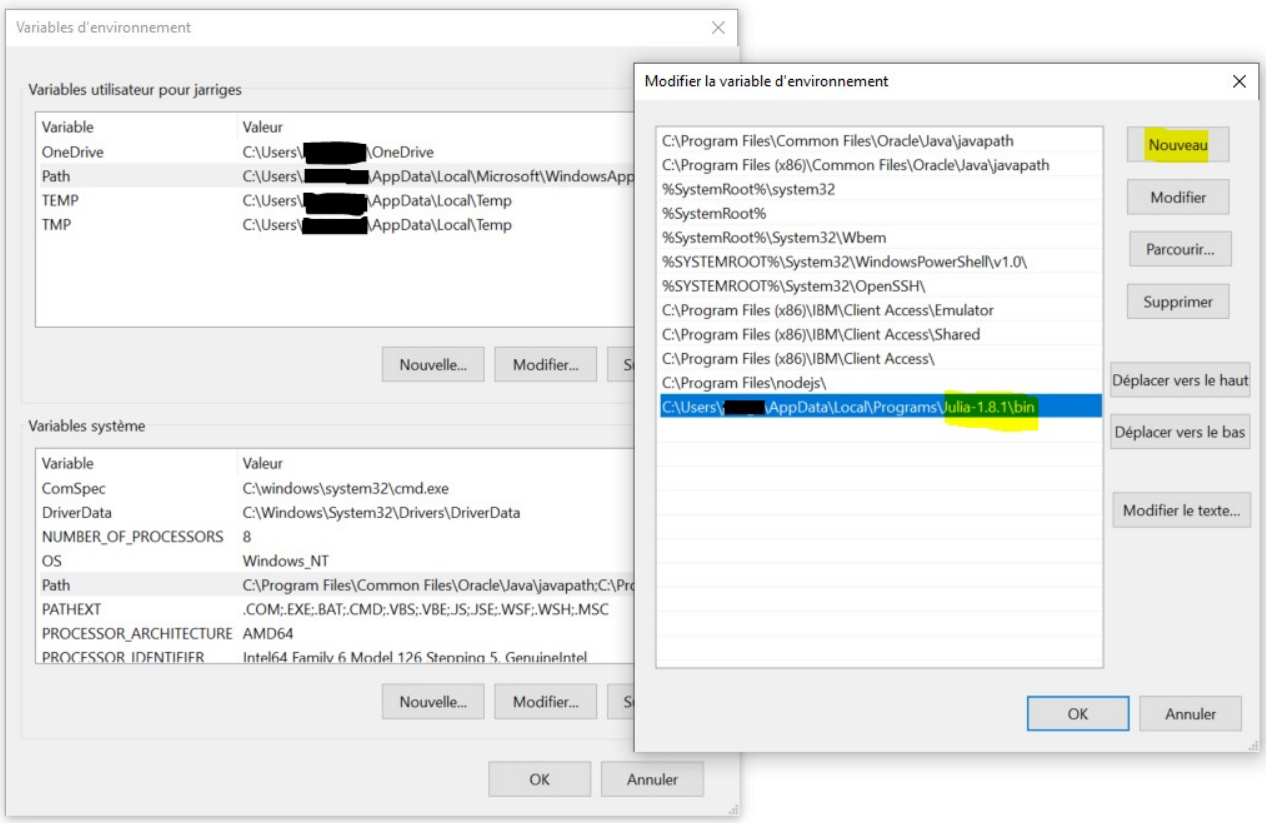
(il y a plusieurs moyens d'arriver à cette option, le plus rapide étant par la barre de recherche de Windows)

Dans la fenêtre ci-dessous, cliquez sur le bouton « variables d'environnement » :



Sélectionnez l'option « Path », puis cliquez sur le bouton « Modifier ».

Puis ajoutez une nouvelle variable d'environnement contenant le chemin d'accès vers le répertoire « bin » dans lequel se trouve l'exécutable de Julia. Puis validez en cliquant sur « OK » :



2 - Installer et configurer un IDE

Pour le choix d'un IDE, plusieurs possibilités s'offrent à vous pour développer du code Julia sur Windows, parmi lesquels Visual Studio Code (VSC), Juno, Jupyter Notebooks, Weave... Je me suis focalisé sur VSC.

Site officiel de VSC :
https://code.visualstudio.com/

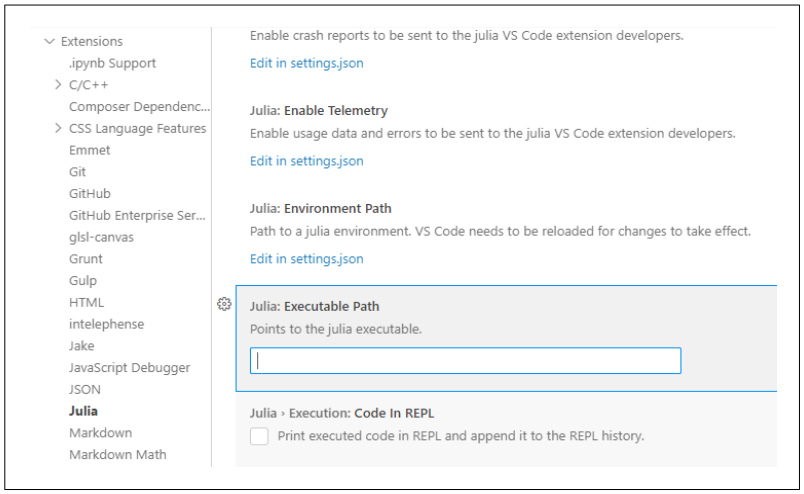
VSC est un très bon IDE pour l'écriture de programmes Julia, à condition de lui ajouter l'extension adéquate, qui s'appelle tout simplement « Julia ».

Pour ajouter l'extension « Julia », allez dans le menu :

```
Fichier → Préférences → Extensions      (ou Ctrl+Maj+X)
```

… puis installez et activez l'extension :

Allez ensuite dans les paramètres de VSC, pour indiquer à l'extension Julia où se trouve l'exécutable Julia.exe que vous avez installé au chapitre précédent :



Attention : j'ai constaté que l'extension Julia de VSC n'appréciait pas les chemins d'accès « à la windows » (c'est-à-dire avec les doubles antislashs), aussi je vous recommande de saisir un chemin d'accès « à la Linux », même si vous êtes sous Windows. Sur mon PC de développement perso, cela donne ceci :

```
C:/Users/grego/AppData/Local/Programs/Julia-1.8.3/bin/julia.exe
```

Une fois le chemin d'accès saisi, relancez VSC.

Pour tester Julia dans VSC, créez un petit script ayant l'extension « .jl ». Appelez-le par exemple « test_string.jl », et saisissez-y le code suivant :

```
# creating 3 strings
s1 = "I"
s2 = "Love"
s3 = "Julia"

# concatenating the strings
s = "$s1 $s2 $s3"

print(s)
```

Ouvrez le terminal de VSC et saisissez la commande suivante :

```
>julia test_string.jl
I Love Julia
```

3 - configurer le proxy et installer le package ODBC

Si vous n'avez pas de contrainte de proxy, sautez la partie configuration, et allez tout de suite à la fin de cette page, pour installer le package ODBC.
Par contre, si vous utilisez Julia derrière un proxy, vous devez le configurer dans Julia pour pouvoir télécharger les packages nécessaires à vos développements. Pour cela, localisez le fichier "startup.jl", qui devrait se trouver dans le sous répertoire "../etc/julia/.." de votre instance Julia, et éditez-le avec votre éditeur préféré. Ajoutez-y les 2 lignes commençant par ENV ci-dessous, en personnalisant l'url du proxy avec vos propres identifiants, adresse IP et port :

```
# This file should contain site-specific commands to be executed on Julia startup;
# Users may store their own personal commands in `~/.julia/config/startup.jl`.
ENV["HTTP_PROXY"] = "http://USER:PASSWORD@adresseIP:port"
ENV["HTTPS_PROXY"] = "http://USER:PASSWORD@adresseIP:port"
```

Sauvegardez, refermez le fichier de configuration, et relancez Julia.

Vous pouvez vérifier la bonne prise en compte des paramètres proxy en saisissant la commande suivante :

```
julia> println(ENV["HTTP_PROXY"]); println(ENV["HTTPS_PROXY"])

http://UUUUUUUU:PPPPPPPP@xxx.xxx.xxx.xxx:zzzz
http://UUUUUUUU:PPPPPPPP@xxx.xxx.xxx.xxx:zzzz
```

Dans les exemples suivants, nous allons utiliser le package ODBC de Julia, alors installons-le tout de suite via la commande ci-dessous :

```
julia> import Pkg; Pkg.add("ODBC")
```

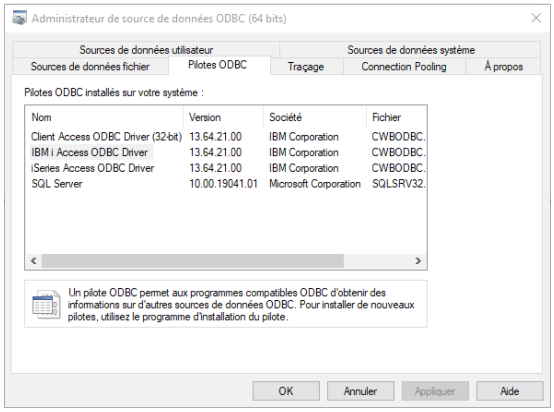
Si la configuration du proxy est correcte, vous devez voir une longue liste de packages s'installer, comme dans l'extrait suivant :

```
  Installed Preferences ──────────────── v1.3.0
  Installed Tables ───────────────────── v1.10.0
  Installed Libiconv_jll ─────────────── v1.16.1+1
  Installed DataAPI ──────────────────── v1.12.0
  Installed DecFP_jll ────────────────── v2.0.3+1
  Installed DataValueInterfaces ──────── v1.0.0
  Installed IteratorInterfaceExtensions ─ v1.0.0
```

…

4 - Configurer une connexion ODBC

Nous allons maintenant connecter Julia à la base de donnée DB2 d'une plateforme IBM i. Pour ce faire, nous avons besoin du driver suivant : {IBM i Access ODBC Driver}
Ce driver est généralement fourni d'office avec le logiciel ACS d'IBM, mais il peut aussi être installé de manière indépendante.
Vous pouvez vérifier s'il est installé sur votre machine en saisissant « ODBC » dans la barre de recherche de Windows, et en sélectionnant l'option « Sources de données ODBC ». Une fenêtre apparaît, dans laquelle vous devez sélectionner l'onglet « Pilotes ODBC ».



Si le driver est absent et que vous devez l'installer vous-même, reportez-vous à la documentation suivante :
https://ibmi-oss-docs.readthedocs.io/en/latest/odbc/installation.html

Si le driver est bien présent, assurez-vous que vous disposez d'un profil utilisateur et d'un mot de passe valide, pour vous connecter à l'un des serveurs IBM i de votre organisation.

Mais avant de tester la connexion, vous pouvez dans un premier temps vérifier si Julia « voit » bien les drivers ODBC disponibles, via le script suivant :

```
using ODBC
df = ODBC.drivers()
println(df)
```

Sauvegardez et lancez ce script dans le terminal VSC avec la commande « julia ». Vous devriez obtenir un affichage proche de celui ci-dessous :

```
#  Dict(
#    "iSeries Access ODBC Driver" => "SQLLevel=2\0APILevel=2\0ConnectFunctions=YYY\0CPTimeout=60\0DriverODBCVer=03.51\0",
#    "IBM i Access ODBC Driver" => "DriverODBCVer=03.51\0CPTimeout=60\0ConnectFunctions=YYY\0APILevel=2\0SQLLevel=2\0",
#    "SQL Server" => "APILevel=2\0ConnectFunctions=YYY\0CPTimeout=60\0DriverODBCVer=03.50\0FileUsage=0\0SQLLevel=1\0UsageCount=1\0",
#    "Client Access ODBC Driver (32-bit)" => "SQLLevel=2\0DriverODBCVer=03.51\0CPTimeout=60\0APILevel=2\0ConnectFunctions=YYY\0"
#  )
```

Créez maintenant un petit script « configdb.jl » contenant les variables suivantes :

```
dbq = "MABIBL"           # votre database par défaut
userid = "XXXXXXXXX"     # votre profil utilisateur
userpw = "YYYYYYYYY"     # le mot de passe associé à votre profil
system = "MyIBMiServer" # le nom du système IBM i cible (ou son adresse IP)
```

Quand vous aurez besoin de récupérer ces variables dans un autre script, vous utiliserez l'instruction suivante :

```
include("configdb.jl");
```

Voici un petit script qui exécute une requête SQL sur une base DB2 for i, de manière à récupérer la date et l'heure système. Je l'ai appelé « test_odbc01.jl » mais vous pouvez lui donner un autre nom:

```
using ODBC
using DataFrames

include("configdb.jl");
conn = ODBC.Connection("DRIVER={IBM i Access ODBC Driver};" *
        "SYSTEM=$system;NAM=1;DBQ=$dbq;CCSID=1208;UID=$userid;PWD=$userpw")

sql = "SELECT current_timestamp as tstamp FROM SYSIBM.SYSDUMMY1"
df = DBInterface.execute(conn, sql) |> DataFrame

println(df)

## 1×1 DataFrame
##   Row │ TSTAMP
##       │ DateTime
##  ─────┼──────────────────────────
##     1 │ 2022-12-14T15:58:52.362
```

5 - Script Julia de test, avec table SQL

Dans l'exemple qui suit, nous allons utiliser le package DataFrames. C'est un package développé en Julia qui simplifie grandement la manipulation de datasets en provenances de bases de données. Nous allons utiliser de nouveau l'ordre « import Pkg », comme dans l'exemple suivant :

```
julia> import Pkg; Pkg.add("DataFrames")
    Updating registry at `C:\Users\[redacted].julia\registries\General.toml`
   Resolving package versions...
   Installed Formatting ──────── v0.4.2
   Installed SnoopPrecompile ──── v1.0.1
   Installed LaTeXStrings ─────── v1.3.0
   Installed Crayons ──────────── v4.1.1
```

L'installation du package DataFrames va déclencher l'installation d'une douzaine de dépendances. La liste ci-dessus n'en est qu'un extrait.

Voici un exemple de script Julia, qui effectue une requête SQL sur une table des codes pays, table dont le code source vous est fourni en annexe :

```
using ODBC
using DataFrames

include("configdb.jl");
conn = ODBC.Connection("DRIVER={IBM i Access ODBC Driver};" *
        "SYSTEM=$system;NAM=1;DBQ=$dbq;CCSID=1208;UID=$userid;PWD=$userpw")

# sélection des pays dont le nom contient FR
sql = "SELECT ID, CODISO3, CODISO2, COUNTRYNAM FROM MYLIBRARY.COUNTRIES WHERE COUNTRYNAM LIKE ?"
df = DBInterface.execute(conn, sql, ["%FR%"]) |> DataFrame

# force la colonne ID en type INTEGER (Int64) dans le dataset de sortie
df.ID = Int.(df.ID)

# exemple pour forcer une colonne en type Float64 (pas utilisé ici)
#  df.ID = Float64.(df.ID)

println(df)

println("Total des IDs => ", sum(df.ID));
println("ID moyen => ", sum(df.ID)/length(df.ID));
```

Je suis d'accord avec vous, faire des cumuls ou des moyennes sur une colonne ID (identifiant), cela ne présente pas d'intérêt, mais c'était juste pour vous montrer le principe.

Résultat produit par le script à l'exécution:

```
7×4 DataFrame
 Row │ ID     CODISO3  CODISO2  COUNTRYNAM
     │ Int64  String   String   String
─────┼──────────────────────────────────────────────────────
   1 │   248  ZAF      ZA       AFRIQUE DU SUD
   2 │   290  CAF      CF       CENTRAFRICAINE, REPUBLIQUE
   3 │   324  FRA      FR       FRANCE
   4 │   342  GUF      GF       GUYANE FRANCAISE
   5 │   428  PYF      PF       POLYNESIE FRANCAISE
   6 │   441  MAF      MF       SAINT-MARTIN (PARTIE FRANCAISE)
   7 │   472  ATF      TF       TERRES AUSTRALES FRANCAISES

Total des IDs => 2545
ID moyen => 363.57142857142856
```

Si tout a bien fonctionné, vous pouvez vous amuser à tester différentes techniques fournies par le package DataFrames, telles que :

```
println(size(df))           # (7, 4)
println(nrow(df))           # 7
println(ncol(df))           # 4
println(names(df))          # ["ID", "CODISO3", "CODISO2", "COUNTRYNAM"]
println(propertynames(df))  # [:ID, :CODISO3, :CODISO2, :COUNTRYNAM]

println(describe(df))
# 4×7 DataFrame
#  Row │ variable     mean     min             median  max                          nmissing  eltype
#      │ Symbol       Union…   Any             Union…  Any                          Int64     DataType
# ─────┼──────────────────────────────────────────────────────────────────────────────────────────────
#    1 │ ID           363.571  248             342.0   472                                 0  Int64
#    2 │ CODISO3               ATF                     ZAF                                  0  String
#    3 │ CODISO2               CF                      ZA                                   0  String
#    4 │ COUNTRYNAM           AFRIQUE DU SUD          TERRES AUSTRALES FRANCAISES         0  String

# Récupération des valeurs de la colonne CODISO3 sous forme d'un tableau
println(df."CODISO3")       # ["ZAF", "CAF", "FRA", "GUF", "PYF", "MAF", "ATF"]
# autre technique équivalente
println(df[:, "CODISO3"])   # ["ZAF", "CAF", "FRA", "GUF", "PYF", "MAF", "ATF"]

# Renommage des colonnes du dataset
rename!(df, [:a, :b, :c, :d])

println(df)
# 7×4 DataFrame
#  Row │ a      b       c       d
#      │ Int64  String  String  String
# ─────┼──────────────────────────────────────────────────
#    1 │   248  ZAF     ZA      AFRIQUE DU SUD
#    2 │   290  CAF     CF      CENTRAFRICAINE, REPUBLIQUE
# etc.
```

Nous n'avons fait qu'effleurer l'utilisation du package Dataframes. Voici quelques ressources qui vous permettront d'aller plus loin :

- Introduction · DataFrames.jl (juliadata.org)

- https://github.com/bkamins/Julia-DataFrames-Tutorial/

- https://juliaacademy.com/p/introduction-to-dataframes-jl1

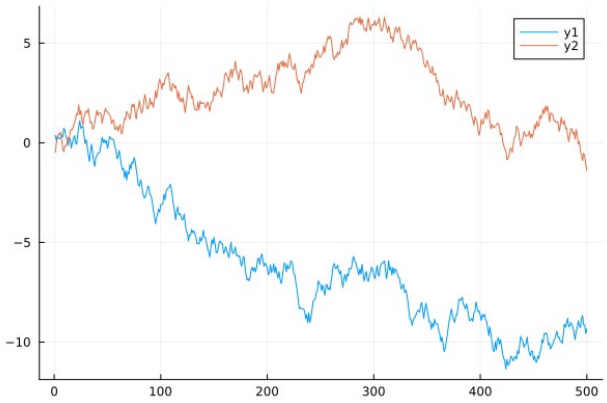
- le livre "Julia for Data Analysis", de Bogumil Kaminski (voir le chapitre "bibliographie").

6 - Dataviz en accéléré

Julia fournit une batterie d'outils permettant d'être immédiatement productif en matière de visualisation de données.
Voici quelques exemples de graphiques générés (en SVG et PNG) avec le package Plot :

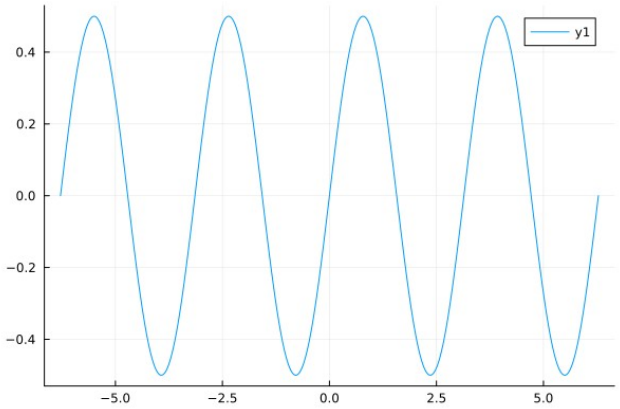
- Exemple 1

```
# Source : https://community.ibm.com/community/user/powerdeveloper/blogs/swati-karmarkar/2022/09/07/julia-on-ibm-power

import Pkg

Pkg.add("Plots")

using Plots

# plot some data

plot([cumsum(rand(500) .- 0.5), cumsum(rand(500) .- 0.5)])

# save the current figure

savefig("plots.png")

savefig("plots.svg")
```



- exemple 2 :

```
# Source : https://grapeup.com/blog/getting-started-with-the-julia-language/

import Pkg

Pkg.add("Plots")

using Plots

f(x) = sin(x)cos(x)

# plot some data

plot(f, -2pi, 2pi)

# save the current figure

savefig("plots2.png")

savefig("plots2.svg")
```



Quelques alternatives à Plots pour le tracé de graphismes :

- https://www.juliapackages.com/p/gnuplot
- https://github.com/JuliaGraphics/Luxor.jl
- https://github.com/JuliaAnimators/Javis.jl

Bibliographie

Quelques livres pour bien débuter avec Julia, pour la plupart disponibles en version électronique (PDF ou Epub).

- Beginning Julia Programming, *For Engineers and Scientists*

de Sandeep Nagar, Springer 2017

- Julia Quick Syntax Reference, *A Pocket Guide for Data Science Programming*

de Antonello Lobianco, Springer 2019

- Introduction to the Tools of Scientific Computing

de Einar Smith, Springer 2022

- ***Julia as a Second Language***

de Erik Engheim, Ed Manning 2022

- Julia for Data Analysis

de Bogumil Kaminski, Ed Manning 2022

- Hands-On Design Patterns and Best Practices with Julia

de Tom Kwong, PacktPub 2020

- Interactive Visualization and Plotting with Julia

de Diego Javier Zea, PacktPub 08/2022

- Web Development with Julia and Genie

de Ivo Balbaert, PacktPub 11/2022

- Julia Programming Language - From Zero to Expert

de Dr. Mohammad Nauman, PacktPub 2021

- Hands-On Computer Vision with Julia

de Dmitrijs Cudihins, PacktPub 2018

- Julia 1.0 High Performance

de Avik Sengupta, PacktPub 2019

- Julia 1.0 Programming Cookbook

de Przemys aw Szufel, PacktPub 2018

- ***Julia Data Science***

Livre open source consultable en ligne et téléchargeable en PDF, de Jose Storopoli, Rik Huijzer et Lazaro Alonso

Welcome - Julia Data Science

- ThinkJulia

Livre consultable en ligne, de Ben Lauwens, pour apprendre à programmer en Julia (livre aussi disponible chez O'Reilly)

https://benlauwens.github.io/ThinkJulia.jl/latest/book.html

- ***Julia 1.x Documentation***

Documentation du site officiel de Julia, consultable en ligne et téléchargeable en PDF (plus de 1500 pages)

Julia Documentation · The Julia Language

Ressources internet complémentaires

- Chaîne youtube dédiée :
https://www.youtube.com/@TheJuliaLanguage

- Cours du MIT
https://computationalthinking.mit.edu/Fall20/

https://ocw.mit.edu/courses/18-s190-introduction-to-computational-thinking-with-julia-with-applications-to-modeling-the-covid-19-pandemic-spring-2020/

- Tutoriaux officiels
https://julialang.org/learning/

- Julia Academy
https://juliaacademy.com/courses

- Julia Database interfaces
https://juliadatabases.org/

- Doc du connecteur ODBC de Julia
https://odbc.juliadatabases.org/dev/

- Introduction à Julia
https://grapeup.com/blog/getting-started-with-the-julia-language/

- Tuto Julia chez Tutorialspoint.com
https://www.tutorialspoint.com/julia/

- Tuto Julia sur Developpez.com
https://julia.developpez.com/tutoriels/decouverte-julia/

- Julia package for working with persistent data sets
https://juliadb.juliadata.org/latest/

- Blog du framework Genie (framework Julia pour le développement de webapps)
https://genieframework.com/blog/

- Communautés de dev Julia
http://gitter.im/JuliaLang/julia
https://discourse.julialang.org/

- Panoply, middleware propriétaire pour interfacer Julia avec DB2 (pas testé)
https://panoply.io/integrations/ibm-db/julia/

- Projet de portable de Julia sur IBM Power
https://julialang.org/downloads/#support_tiers_for_the_latest_stable_release_of_julia

- Installation de Yum sur IBM i
https://www.ibm.com/support/pages/getting-started-open-source-package-management-ibm-i-acs

Annexe

Table SQL des codes pays adaptée à DB2, conçue pour fonctionner avec l'exemple du chapitre 5.

```
-- List of ISO 3166 country codes adapted for DB2
-- Source : https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes?adlt=strict

CREATE OR REPLACE TABLE MYLIBRARY.COUNTRIES (
  ID INTEGER NOT NULL
    GENERATED ALWAYS AS IDENTITY
    ( START WITH 1 , INCREMENT BY 1 , CACHE 10 )
    UNIQUE,
  CODISO3 CHAR(3) NOT NULL DEFAULT '',
  CODISO2 CHAR(2) NOT NULL DEFAULT '',
  COUNTRYNAM VARCHAR(50) NOT NULL DEFAULT ''
);

LABEL ON TABLE MYLIBRARY.COUNTRIES IS 'List of ISO 3166 country codes';

LABEL ON COLUMN MYLIBRARY.COUNTRIES (
  ID IS 'ID du pays' ,
  CODISO3 IS 'Code ISO sur 3c' ,
  CODISO2 IS 'Code ISO sur 2c' ,
  COUNTRYNAM IS 'Nom du pays'
);

CREATE INDEX MYLIBRARY.COUNTRIES01 ON MYLIBRARY.COUNTRIES (CODISO3) ;
CREATE INDEX MYLIBRARY.COUNTRIES02 ON MYLIBRARY.COUNTRIES (CODISO2) ;

---------- DATAS ----------

INSERT INTO MYLIBRARY.COUNTRIES (CODISO3, CODISO2, COUNTRYNAM)
VALUES
 ('AFG ', 'AF ', 'AFGHANISTAN'),
 ('ZAF ', 'ZA ', 'AFRIQUE DU SUD'),
 ('ALA ', 'AX ', 'ALAND, ILES'),
 ('ALB ', 'AL ', 'ALBANIE'),
 ('DZA ', 'DZ ', 'ALGERIE '),
 ('DEU ', 'DE ', 'ALLEMAGNE'),
 ('AND ', 'AD ', 'ANDORRE'),
 ('AGO ', 'AO ', 'ANGOLA'),
 ('AIA ', 'AI ', 'ANGUILLA'),
 ('ATA ', 'AQ ', 'ANTARCTIQUE'),
 ('ATG ', 'AG ', 'ANTIGUA-ET-BARBUDA'),
 ('ANT ', 'AN ', 'ANTILLES NEERLANDAISES '),
 ('SAU ', 'SA ', 'ARABIE SAOUDITE'),
 ('ARG ', 'AR ', 'ARGENTINE'),
 ('ARM ', 'AM ', 'ARMENIE'),
 ('ABW ', 'AW ', 'ARUBA'),
 ('AUS ', 'AU ', 'AUSTRALIE'),
 ('AUT ', 'AT ', 'AUTRICHE'),
 ('AZE ', 'AZ ', 'AZERBAIDJAN'),
 ('BHS ', 'BS ', 'BAHAMAS'),
 ('BHR ', 'BH ', 'BAHREIN'),
 ('BGD ', 'BD ', 'BANGLADESH'),
…
```

(le code source complet de la table DB2 est fourni dans le repo Github indiqué page 2 du présent dossier).