# Basic Tutorial

* 1. Prepare for the Model System Environment
     1. Download Julia Language and Learn about the Basic Syntax

Julia language is a fast, dynamic, reproducible, composable, general, and open-source language. Due to its fast performance, Julia may have a great performance on modeling the integrated assessment models, even not less than the commercial software.

Julia could be downloaded from its official website <https://julialang.org/downloads/>, a current stable release version is welcomed according to the operation system of users. For Chinese users, an stable and faster download from Tsinghua University open source software mirror station is recommended <https://mirrors.tuna.tsinghua.edu.cn/julia-releases/bin/>.

Learning about the basic syntax of Julia is helpful to develop the model system individually, therefore, users are encouraged to learn from Julia documentation or other souces. <https://docs.julialang.org/en/v1/> (English) and <https://docs.juliacn.com/latest/> (Chinese) are the sources of documentation.

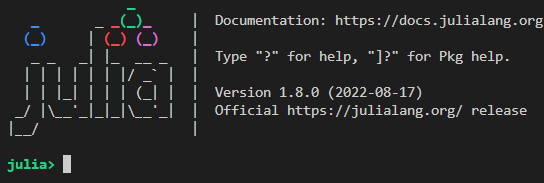
### Download Visual Studio Code as the IDE and Code Editor

VS Code is an open-source, powerful code editor and IDE, which is compitable to various languages, therefore, helpful to develop the GRID model. A stable build version could be downloaded from <https://code.visualstudio.com/>. After installing, the Julia extensions in VS code is needed to make the Julia works in the VS Code. VS code has some integrated functions including code editor, Julia running, debugger and terminal. Git is another useful tool for the joint development of GRID model.

* + 1. Install the dependent packages for modeling

Before developing the model system, the dependent packages need installing by using the Terminal in VS code. Firstly, user should test whether Julia is in the environment variable by entering Julia in the terminal. If the version of Julia is printed in the Terminal, then all is okay for the next step; else the Julia should be added in the environment variable.

Package installation in Julia is conducted by using Pkg in the Terminal, shown in below. Currently, the necessary packages are: JuMP, DataFrames, CSV, PathSolver, and Complementarity. DataFrames and CSV are required to read data from data system, JuMP is required as the modeling language in Julia, PathSolver and Complementarity are required to solve the GRID model by using PATH solver.



using Pkg

Pkg.add("Pkg\_Name")

* + 1. A Quick start

A quick start can be done using CGE1.jl, which is a simple computable general equilibrium model with two sectors and no intermediate inputs. The code includes three parts: preparing for parameters, generating CGE model, and solving the model by calling the solve\_cge() function. If the model is solved normally, the user may be ready for the next modeling tutorial.

* 1. Learning the Julia-based CGE Modeling Step by Step
     1. How to learn CGE models

A good textbook for CGE modeling is useful and highly recommended to learn about the theory, model structure and traditional modeling approach of CGE models. Needs a recommendation for English users. For Chinese users, the Basic principles and programming of computable general equilibrium models, written by Zhang, Xin, is a comprehensive introduction to the theory and modeling approach, which is also very easy to understand. If the users are quite interested in CGE modeling and wish to work deeply, the textbooks could be read as necessary.

The operational research and numeric analysis are also some basic requirements for CGE modelers. It may be better to learn about some non-linear algorithms. By combining the theory of general equilibrium and the though of operational research, users could have more ability to update and use the GRID model.

* + 1. Learning about the JuMP language

JuMP is a domain-specific modeling language for mathematical optimization embedded in Julia. It currently supports a number of open-source and commercial solvers for a variety of problem classes, including linear, mixed-integer, second-order conic, semidefinite, and nonlinear programming. Please read the section of Introduction, Tutorials and Manuals to understand how a model could be formulated and solved in JuMP. GRID model is a nonlinear model, and thus please give more attention to the part of nonlinear modeling. PathSolver.jl and Complementarity.jl are the secondary developmented packages for solving mixed complementarity problem (MCP) by using PATH solver, which can be found in <https://github.com/chkwon/PATHSolver.jl> and

<https://github.com/chkwon/Complementarity.jl>. The introduction to the two packages can be found in README.md.

A CGE model is a large system of simultaneous non-linear equations, which is generally modeled in three software, including General Equilibrium Modelling Package (GEMPACK), General Algebraic Modeling System (GAMS) and Mathematical Programming System for General Equilibrium analysis (GAM/MPSGE). Although the three frameworks are effective for CGE modeling, they are commercial and thus hinders the widespread application in all regions and organizations. Therefore, an open-source modeling framework is attractive and important to build a CGE modeling community, and track the frontier modeling techniques around the world. Two mainstream open-source modeling language is Python-based, open-source optimization modeling (Pyomo) and Julia for Mathematical Programming (JuMP). JuMP would be a better choice for open-source CGE modeling because of the time efficiency in large-scale model, and a wrapper call to the PATH solver using packages PATHSolver.jland Complementarity.jl.

By applying JuMP and Complementarity.jl, CGE model could be described as a mixed complementarity problem (MCP). Taking the carbon emission cap as an example, if carbon emissions is smaller than the emission cap, the carbon price would be zero; otherwise, if the carbon emissions is larger than the cap, the carbon price must be a positive value to control the emissions equal to the cap. We would have . For most equations about price and non-zero amount, given that if the model could be solved normally, the equations must be met. Therefore, we have no need to set the bounds of most variables, in this situation, the equations can be also represented as MCP problem.

* + 1. Practice with the CGE Cases

The tutorial provides lots of CGE cases with an increasing complexity one by one. It is thus helpful to practice with those CGE cases at first, then a comprehensive model could be easier to understand, which could be found in GitHub. CGE7.jl and CGE13.jl are dropped because those two have only little improvement. More cases will be added in the GitHub https://github.com/KennethAnn/Julia-based-CGE-Tutorial.git. If the model of users is developed in the form of our model, please add reference, the link will be added in the tutorial after the methodology article is published.

CGE1: A simple model with two sectors and no intermediate input

CGE2: A Simple CGE Model with three sectors and intermediate input

CGE3: A Simple CGE Model with three sectors and intermediate input, as well as Linear Expenditure System

CGE4: A Simple CGE Model with three sectors and intermediate input, as well as Linear Expenditure System, add the price base

CGE5: A Simple CGE Model with government

CGE6: A Simple CGE Model with one household and one government

CGE8: A Large CGE Model with 42 sectors, 2 households and one government

CGE9: A Large CGE Model with 42 sectors, 2 households, 1 government and investment

CGE10: A open-economy CGE model with 42 sectors, 2 households, 1 government, 1 investment and 1 foreigner

CGE11: A energy-economy CGE model, considering the energy-KL bundle

CGE12: A energy-economy-climate CGE model, adding the carbon accounting and carbon pricing module

CGE14: A recursive-dynamic energy-economy-climate CGE model, adding the carbon accounting and carbon pricing module, using efficiency variable

* + 1. Prepare for Data

Lots of data are required for CGE modeling. The data needed is at least including:

1. **Social Accounting Matrix**

Social Accounting Matrix is generated by combining the input-output table and capital flow table. It is therefore necessary to find the methodology and treat the data of the interested region. Both single and multi-regional SAM table could be supported by the CGE model. The above CGE cases are single-regional, but if the multi-regional SAM table is finished, the code for multi-regional CGE model could be provided.

1. **Energy balance table**

Energy balance table is required to calibrate the energy demand and energy structure in the economic system. Through combining the energy balance table and social accounting matrix, an energy-economic CGE model can be obtained as CGE14.

1. **Prediction data, especially GDP, population.**