Photo by Daniel Zacatenco (https://unsplash.com/@danielzacatenco) on Unsplash (https://unsplash.com/)

Data storage: HDF5

3 minute read

In this lesson we will learn how to store different kind of data on disk. For this purpose we will use `JLD.jl` (https://github.com/JuliaIO/JLD.jl), a Julia dialect of HDF5, which is a file format designed to store and organise large amounts of data.

Operating on `.jld` files

Installing `JLD.jl`

First of all we need to install `JLD`, to do it type the following code:

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

Exporting data

`JLD` can save to disk almost any form of data, including **variables, dictionaries** and even **concrete types**. In order to save some data, we first need to create a dictionary containing a string identifier for each element (the key in the dictionary) and the data. Then we can export that dictionary with the `save` function. For example, we can do it in this way:

```
1  using JLD
2
3  x = collect(-3:0.1:3)
4  y = collect(-3:0.1:3)
5
6  xx = reshape([xi for xi in x for yj in y], length(y), length(x))
7  yy = reshape([yj for xi in x for yj in y], length(y), length(x))
8
9  z = sin.(xx .+ yy.^2)
10
11 data_dict = Dict("x" => x, "y" => y, "z" => z)
12
13 save("data_dict.jld", data_dict)
```

At line 3-4 we define `x` and `y` (remember that `collect` transforms a range into an array), at line 6-7 we create a grid of `x` and `y` to compute all the possible combinations of `x` and `y`. At line 9 we compute z and at line 11 we create a dictionary containing the variables that we want to store: `x`, `y` and `z`. At line 13 we export `data_dict` through the `save` function to a file called `data_dict.jld`.

Reading data

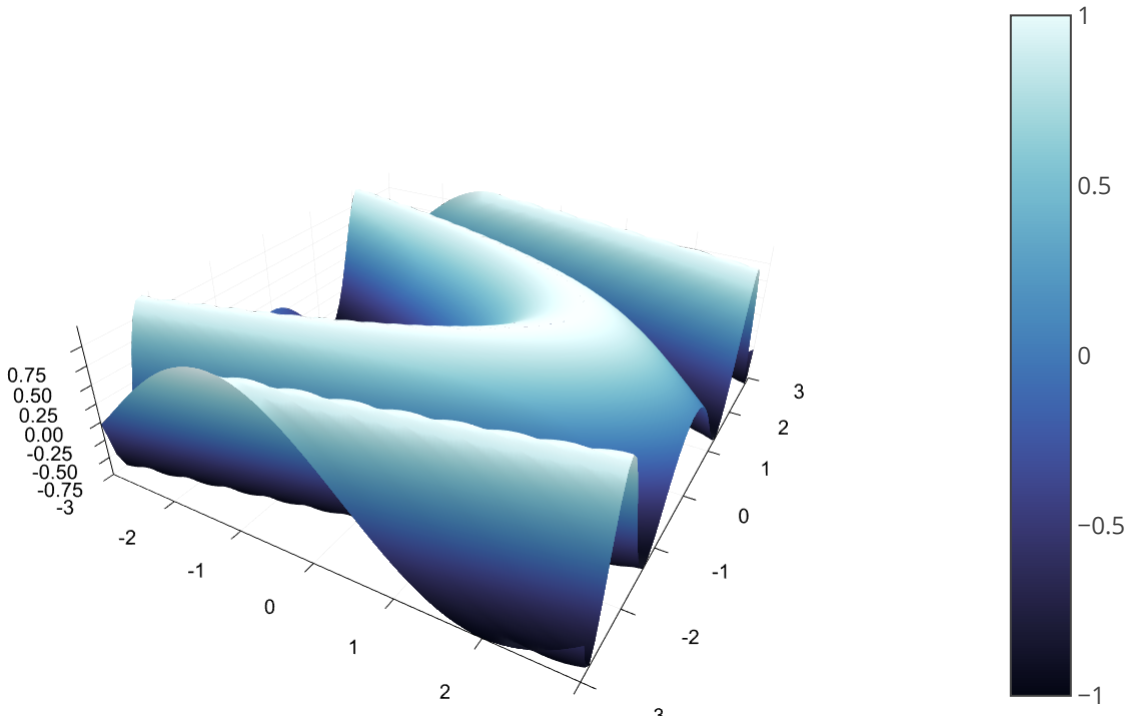
In order to demonstrate that the data is actually read from disk, please **restart the REPL**.

It is possible to read a `.jld` file through the `load` function:

```
1  using JLD
2  data_dict2 = load("data_dict.jld")
```

We can now inspect the content of `data_dict2` and perform some operations with the loaded data, for example we can plot it:

```
1  x2 = data_dict2["x"]
2  y2 = data_dict2["y"]
3  z2 = data_dict2["z"]
4
5  using Plots
6  plotly()
7
8  plot(x2, y2, z2, st = :surface, color = :ice)
```



Structures

It is also possible to **store structures** in `.jld` archives, which is done in the following way:

```
using JLD
struct Person
    height::Float64
    weight::Float64
end

bob = Person(1.84, 74)

dict_new = Dict("bob" => bob)
save("bob.jld", dict_new)
```

The file is loaded in the same way as before with one exception: the `Person` structure should be defined before loading the archive. Before running the following code please restart the REPL.

```
using JLD
struct Person
    height::Float64
    weight::Float64
end
bob2 = load("bob.jld")

>>>bob2["bob"]
Person(1.84, 74.0)
```

If we restart the REPL and we omit redefining `Person`, we get the following output:

```
using JLD

>>>bob3 = load("bob.jld")
Warning: type Person not present in workspace; reconstructing

>>>bob3["bob"]
JLD.var"##Person#402"(1.84, 74.0)

>>>bob3["bob"].height
1.84
```

As you can see at line 4, we were able to import the file but we didn't get a `Person` structure (line 7), as `Person` was not known at the time of the import. Nonetheless we can retrieve the data stored inside `bob`, as shown at line 9.

> At the time of writing, it is not possible to store data with units of measurement inside `.jld` files.

Conclusions

In this lesson we have learned how it is possible to store and retrieve data using `JLD.jl`. Moreover, in the case of structures, we have seen that it is better to define the desired structure before importing the data.

If you liked this lesson and you would like to receive further updates on what is being published on this website, I encourage you to subscribe to the **newsletter** (https://techytok.com/newsletter/)! If you have any **question** or **suggestion**, please post them in the **discussion below**!

Thank you for reading this lesson and see you soon on TechyTok!

Tags: Julia

Categories: Lessons Tutorial

Updated: December 27, 2019

LEAVE A COMMENT

ALSO ON **TECHYTOK**

Parallel computing

7 months ago • 4 comments

From zero to Julia Lesson 15. Parallel computing

Control Flow

8 months ago • 5 comments

From zero to Julia Lesson 4. Control Flow

Variable Scope

7 months ago • 4 comments

From zero to Julia Lesson 6. Variable scope

Multiprocessing Julia: writing a m

9 months ago • 3 comm

A tutorial on how to \ module in Julia whic multiprocessing

/