github.com/
JuliaDynamics

DrWatson

The perfect sidekick to your scientific inquiries

presented by
George Datseris

latest version:
v0.3.0

Motivation

- I am tired of typing `savename = “w=2_f=2.5_x=5.dat”` can't I do it automatically?
- I moved my folders and my load commands don't work anymore!
- Do I have to produce a dataframe of my simulations AGAIN?!
- Yeah you've sent me your project but none of the scripts work...

Sounds familiar...? **DrWatson** tries to eradicate such bad thoughts and bedtime nightmares!

Functionality

DrWatson is a **scientific project assistant** software

Project Setup
- Well-designed project structure
- Robust navigation of project
- Works no matter where it is on disk
- Fully reproducible project

Naming Simulations
- Deterministic scheme for creating names given *any* parameter collection
- Fully customizable but also with sensible defaults

Saving Tools
- Automatically add Git commit and source file information to saved data
- Safe-saving over existing files
- Produce or load existing files flexibly

Running & Listing Simulations
- Preparing multiple simulation runs
- Automatically creating a parameter table (dataframe) of completed runs
- Support for serial & parallel clusters

Principles

non-invasive
no strict rules to follow, no command line, no changing the way you work

simple
easy to understand and use, function based (just call a function)

reproducible
reproducibility of any result (versioning: code and dependencies)

consistent
functionality identical across all projects

modular
flexible design, use only what you need in *your* project

incremental
you can add more, completely new, simulations/data

Navigating a reproducible project

- Call the function `initialize_project(folder, name)`
- Always find the e.g. data folder by calling `datadir()`, no matter where the calling script is located
- Load source: `include(srcdir()*"unitcells.jl")`
- Automatically add (current) Git commit ID to some data to save: `tagsave(datadir()*“exponents.bson”, data)`
- Send your colleague the `folder` and they have to do:
  `Pkg.activate(“path/to/folder”)`
  `Pkg.instantiate()`
  It is now **guaranteed** that all code that run for you, runs for them!

Getting a name out of a container

- Define a (named) parameter container like so:
  `c = (T = 100, dt = 0.1, N = 25, mode = “bi”)`
- Then calling `savename(c)` gives
  `“N=25_T=100_dt=0.1_mode=bi”`
- Doing instead `savename(datadir(), c, “dat”)` gives
  `“path/to/data/N=25_T=100_dt=0.1_mode=bi.dat”`
- This works for **any** named container, including **any Julia struct**
- Sensibly excludes non-fitting fields (e.g. vector-valued fields) but can be customized to full extent

Producing a dataframe from saved files

- You have run some simulations with parameters a = 1 or 2, b = “x” or “y” and c = 0.3. You save them in directory “data”
- `df1 = collect_results!("results.bson", "data/")`

4×3 DataFrame

Row	a Int64?	b String?	c Float64?
1	1	x	0.3
2	1	y	0.3
3	2	x	0.3
4	2	y	0.3

- You now run a new simulation with a=4, b=“z”, a **new parameter** d=0.5 or 0.6 and the parameter c **does not exist** anymore!
- `df2 = collect_results!("results.bson", "data/")`

6×4 DataFrame

Row	a Any	b Any	c Any	d Float64?
1	1	x	0.3	missing
2	1	y	0.3	missing
3	2	x	0.3	missing
4	2	y	0.3	missing
5	4	z	missing	0.5
6	4	z	missing	0.6

wanted | beta testers | more contributors | feature requests

https://github.com/JuliaDynamics/DrWatson.jl

scan me!