Composable Ocean Modeling with ClimateModels.jl, MITgcm.jl, and Climatology.jl

JuliaEO25, Terceira, Azores

Gaël FORGET, 2025/01/09

Outline

- ClimateModels.jl
- MITgcm.jl
- Climatology.jl
- What's next?

ClimateModels.jl

an interface to all the best models — whatever the language!

Workflows That Run Models

- Random Walk model (Julia) ➪ code link
- ShallowWaters.jl model (Julia) ➪ code link
- Oceananigans.jl model (Julia) ➪ code link
- Hector global climate model (C++) ➪ code link
- FaIR global climate model (Python) ➪ code link
- SPEEDY atmosphere model (Fortran90) ➪ code link
- MITgcm general circulation model (Fortran) ➪ code link

Workflows That Replay Models

- IPCC report 2021 (NetCDF, CSV) ➪ code link
- CMIP6 model output (Zarr) ➪ code link
- ECMWF IFS 1km (NetCDF) ➪ code link
- ECCO version 4 (NetCDF) ➪ code link
- Pathway Simulations (binary, jld2) ➪ code link

JOSS
The Journal of Open Source Software

MITgcm.jl: a Julia Interface to the MITgcm

Gaël Forget [1]

1 Massachusetts Institute of Technology, Cambridge, MA, USA

DOI: 10.21105/joss.06710

Software

- Review
- Repository
- Archive

Editor: Anjali Sandip

Reviewers:

- @simone-silvestri
- @swilliamson7

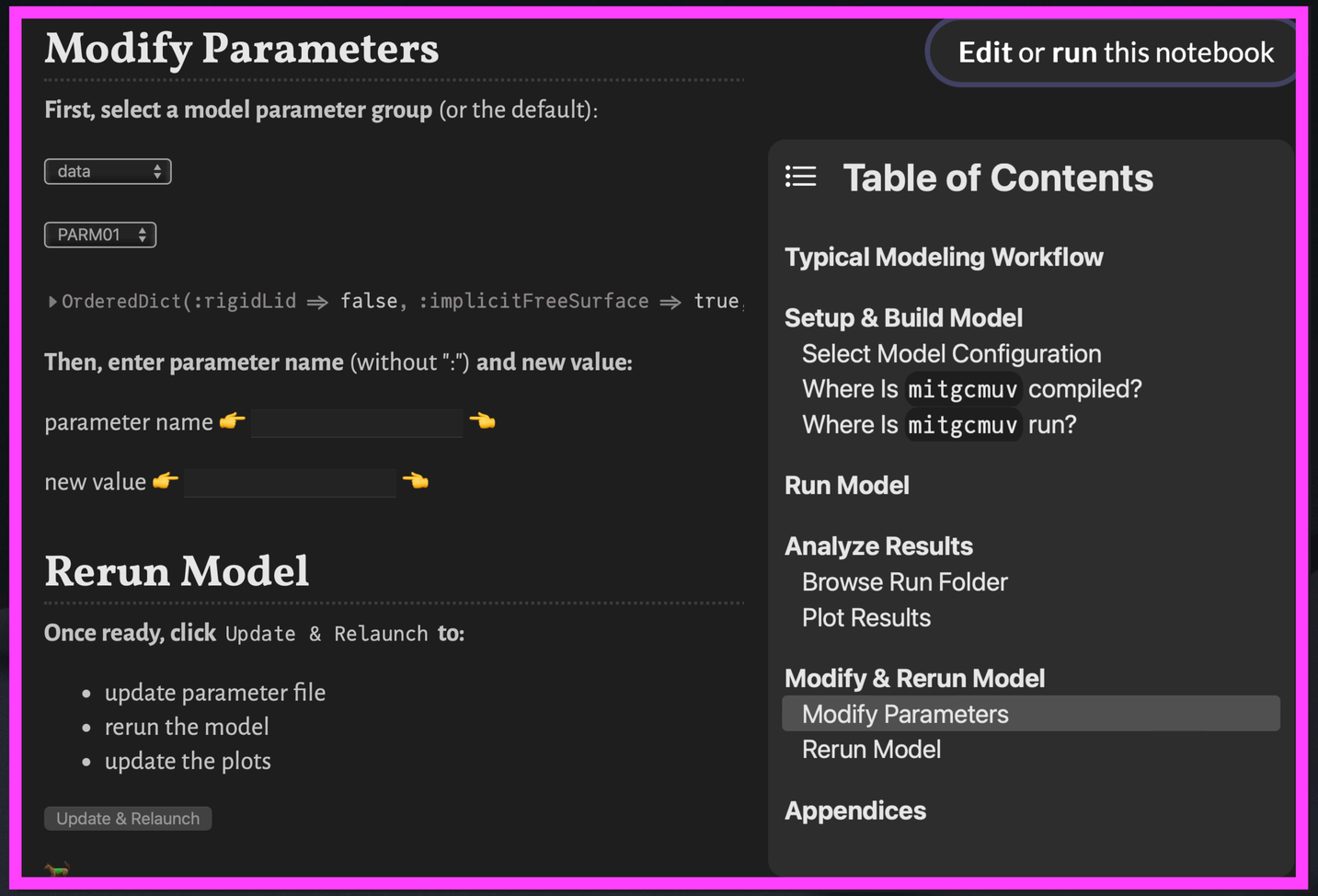
Submitted: 03 April 2024
Published: 22 October 2024

Summary

General circulation models are used to study climate, ocean and atmosphere dynamics, biogechemistry, ecology, and more. The `MITgcm`, written in `Fortran`, is one of the most widely-used models of this class. We present `MITgcm.jl`, a two-way interface to `MITgcm` written in Julia that can be used to not only analyze model results but also drive model simulations. With `MITgcm.jl` users can setup, build, and launch `MITgcm` without having to know shell scripting or having to edit text files manually. `MITgcm.jl` provides support in `Julia` for the various input and output formats used in `MITgcm`. It implements the `ClimateModels.jl` interface, and opens up a whole new way of using `MITgcm` interactively from Jupyter and Pluto notebooks. `MITgcm.jl` in turn brings full-featured, reliable ocean modeling to Julia.

Interactivity Provided by Julia

e.g. Pluto reactive notebook examples



e.g. Julia REPL

```
using MITgcm
params=read_toml(:ECCO4)
MC=MITgcm_config(inputs=params)
run(MC)
```

A Simple Interface

Modular and Convenient

Often one may prefer to break things down though. Let's start with defining the model:

```
MC=ModelConfig(model=ClimateModels.RandomWalker)
```

The sequence of calls within `ModelRun` can then be expanded as shown below. In practice, `setup` typically handles files and software, `build` may compile a chosen model configuration, and `launch` takes care of the main computation.

```
setup(MC)
build(MC)
launch(MC)
```

or just

run(MC)

```
readdir(MC)
```

```
2-element Vector{String}:
 "RandomWalker.csv"
 "log"
```

a Modeling Sequence Example

With the workflow getting recorded via git

```
MC=ModelConfig(f,(NS=100,filename="run01.csv"))
run(MC)

MC.inputs[:NS]=200
MC.inputs[:filename]="run02.csv"
put!(MC)
launch(MC)

log(MC)
```

```
9-element Vector{String}:
 "8ba17c4 initial setup"
 "60bd047 initial tracked_parameters.toml"
 "8d989e9 add Project.toml to log"
 "6eb6027 add Manifest.toml to log"
 "7246c12 task started [3c48089b-0293-4bce-b2d3-7a4a5549dc94]"
 "aa7d237 task ended   [3c48089b-0293-4bce-b2d3-7a4a5549dc94]"
 "91989a4 task started [a04dfe58-c58b-403b-837d-a497da2783da]"
 "0e4076d modify tracked_parameters.toml"
 "16f6efe (HEAD -> main) task ended   [a04dfe58-c58b-403b-837d-a497da2783da]"
```

IPCC model output (a replay example)

Climate Model Archive (Zarr or Netcdf files)

Edit or run this notebook

Choose Model Configuration

"Done retrieving lists of institution_id"

institution_id : IPSL

source_id : IPSL-CM6A-LR

Via `parameters` we now record that we want to access temperature (`tas`) from a model run by `IPS` provided as part of CMIP6 (Coupled Model Intercomparison Project Phase 6).

- `institution_id` = IPSL
- `source_id` = IPSL-CM6A-LR
- `variable_id` = tas

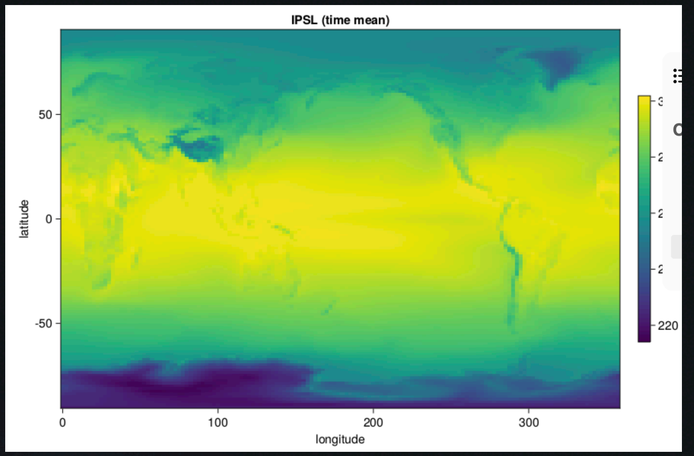
The `cmip_main` function, defined below, calls `demo.cmip_averages` and then writes the output to files:

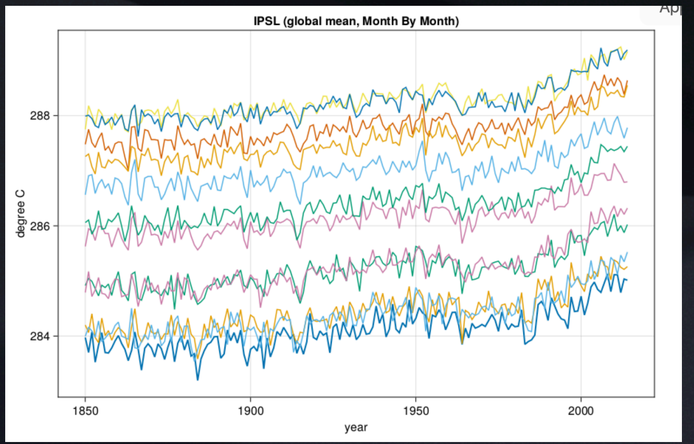
- Global averages in a `CSV` file
- Maps + meta-data in a `NetCDF` file
- Meta-data in a `TOML` file

```
MC =    ID            = 61b81984-bb8f-4f2a-9cc5-02b1560269d5
        model         = CMIP6_averages
        configuration = cmip_main
        run folder    = /tmp/61b81984-bb8f-4f2a-9cc5-02b1560269d5
        log subfolder = /tmp/61b81984-bb8f-4f2a-9cc5-02b1560269d5/log
```

```
MC=ModelConfig(model="CMIP6_averages",configuration=cmip_main,inputs=parameters)
```

Table of Contents





Hector (an emulator example)

Simple, Fast, Global Climate Model (C++)

Modify Parameters & Rerun

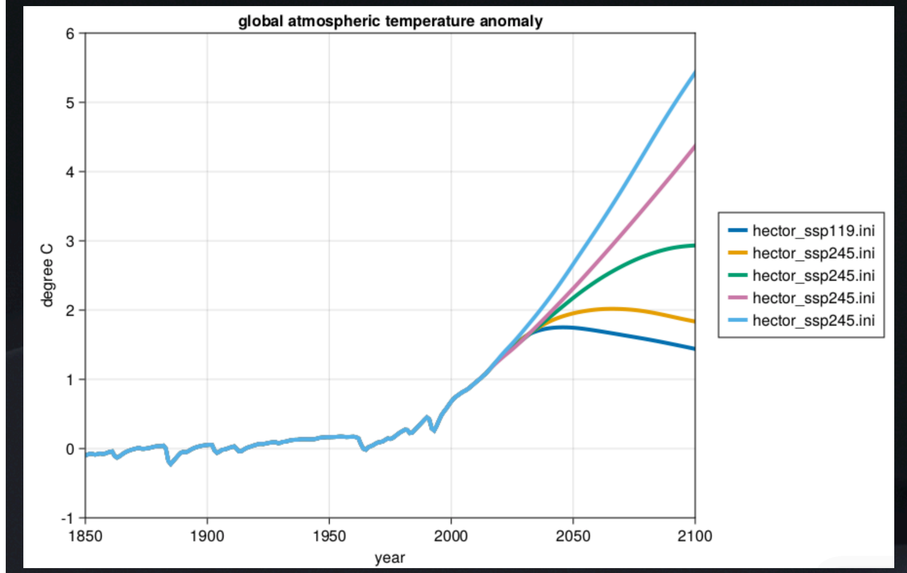
Let's consider the same sbuset of model parameters as done in HectorUI.

Here, we start from hector_ssp245.ini but with a `Equilibrium climate sensitivity` of 3.5 instead of 3.

Parameter name	Value	unit
CO2 fertilization factor	0.36	unitless
Aerosol forcing scaling factor	1.0	unitless
Ocean heat diffusivity	2.3	cm2/s
Equilibrium climate sensitivity	3.5	degC
Preindustrial CO2 conc,	276.0896713615024	ppmv CO2
Temp. sensitivity factor (Q10)	2.0	unitless
Volcanic forcing scaling factor	1.0	unitless

Update & Rerun Model

"/tmp/0f7618b3-3ee4-4e3f-9025-26e7d01a7668/hector/inst/input/custom_parameters.nml"



https://jgcri.github.io/hector/index.html

Outline

- ClimateModels.jl
- MITgcm.jl
- Climatology.jl
- What's next?

Running ECCO4 & larger setups on HPC

```
using MITgcm
params=read_toml(:ECCO4)
MC=MITgcm_config(inputs=params)
run(MC)
```

OR

```
using MITgcm

params=read_toml(:ECCO4)
folder=joinpath(pwd(),"tmp1")

MC=MITgcm_config(inputs=params,folder=folder)

#providing executable (optional)
#push!(MC.inputs[:setup][:main],(:exe => joinpath(pwd(),"mitgcmuv")))

#providing input folder (optional)
#push!(MC.inputs[:setup][:main],(:input_folder => joinpath(pwd(),"input_folder")))

#modifying run time options (optional)
#MC.inputs[:pkg][:PACKAGES][:useECCO]=false

setup(MC)

#modifying build options (optional)
#MC.inputs[:setup][:build][:options]=MITgcm.build_options_pleiades

build(MC)

launch(MC)
```

Running ECCO4 & larger setups on HPC

A bit mode detail

- List of readily supported configurations :
  - https://gaelforget.github.io/MITgcm.jl/dev/functionalities_configurations/
- Run Time Parameters in one TOML file :
  - https://github.com/gaelforget/MITgcm.jl/blob/master/examples/configurations/OCCA2.toml
- Submitting jobs via a queuing system :
  - https://github.com/gaelforget/MITgcm.jl/blob/master/src/ShellScripting.jl

ClimateModels.jl + MITgcm.jl + Drifters.jl + ...

Composing more complex workflows with several models

Setup Model

The most standard MITgcm configurations (*verification experiments*) are all readily available via `MITgcmTools.jl`'s `MITgcm_config` function.

The `setup` function links input files to the `run/` subfolder. The model executable `mitcmuv` is normally found in the `build/` subfolder of the selected experiment and will get linked to `run/`.

Note

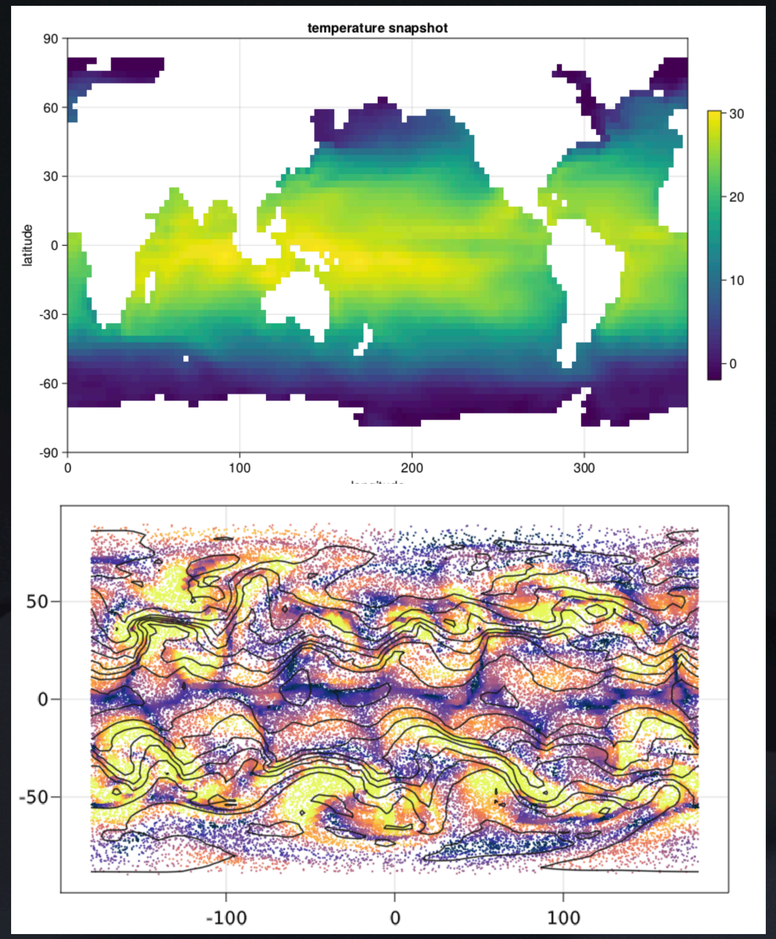
If `mitcmuv` is not found at this stage then it is assumed that the chosen model configuration still needs to be compiled (once, via the `build` function). This might take a lot longer than a normal model run due to the one-time cost of compiling the model.

```
Configuration:

  ID            = 02587801-3024-4df5-bc2d-c85e9a815b45
  model         = MITgcm
  configuration = tutorial_global_oce_biogeo
  run folder    = /tmp/02587801-3024-4df5-bc2d-c85e9a815b45
  log subfolder = /tmp/02587801-3024-4df5-bc2d-c85e9a815b45/log
  task(s)       = MITgcm_launch

With paramater groups:

Any[:main, :diagnostics, :dic, :gchem, :gmredi, :pkg, :ptracers, :eedata]
```



MITgcm is a differentiable Ocean Model (+ Seaice, BGC, Simple Atmospheres,...)

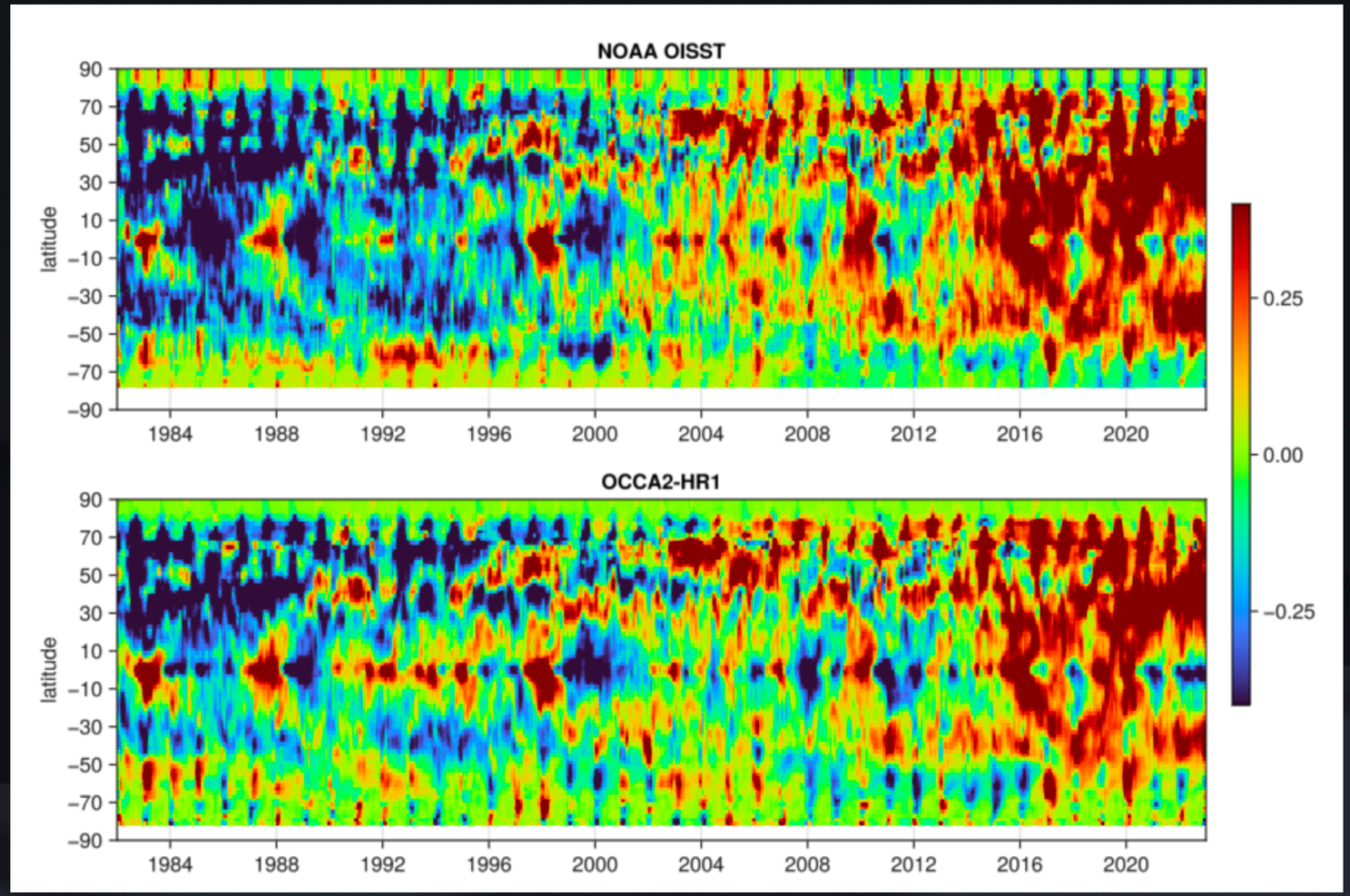
Outline

- ClimateModels.jl
- MITgcm.jl
- Climatology.jl
- What's next?

Energy Imbalance in the Sunlit Ocean

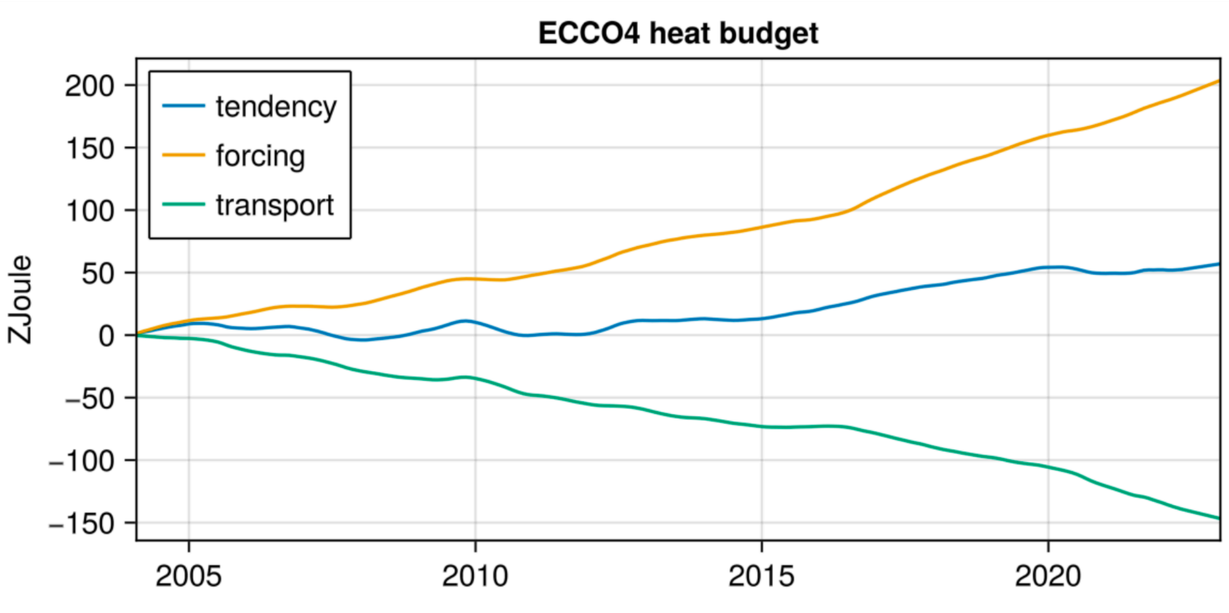
Sunlit Ocean Layer ~ 0-200m layer

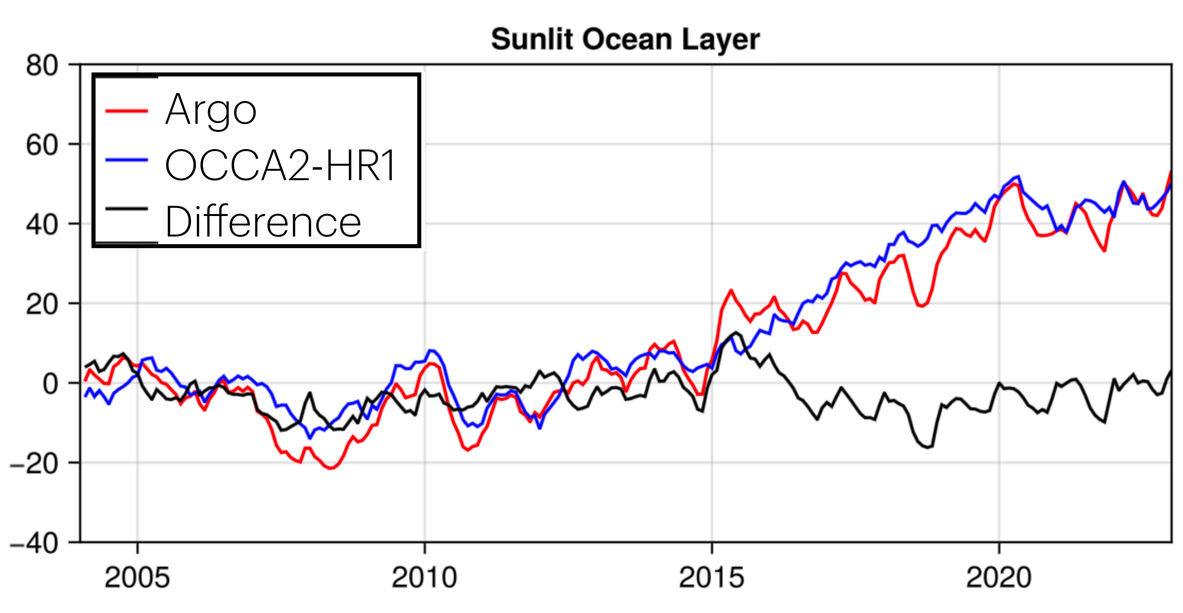
- Recent Acceleration in SST
- EEI accumulation in SOL
- Marine Heatwaves …



Energy Imbalance in the Sunlit Ocean

Global Heat Budget (0-200m)





A large Fraction of EEI goes through the SOL and gets stored below 200m

But recently EEI accumulating in the SOL more and more, leading to greater risk

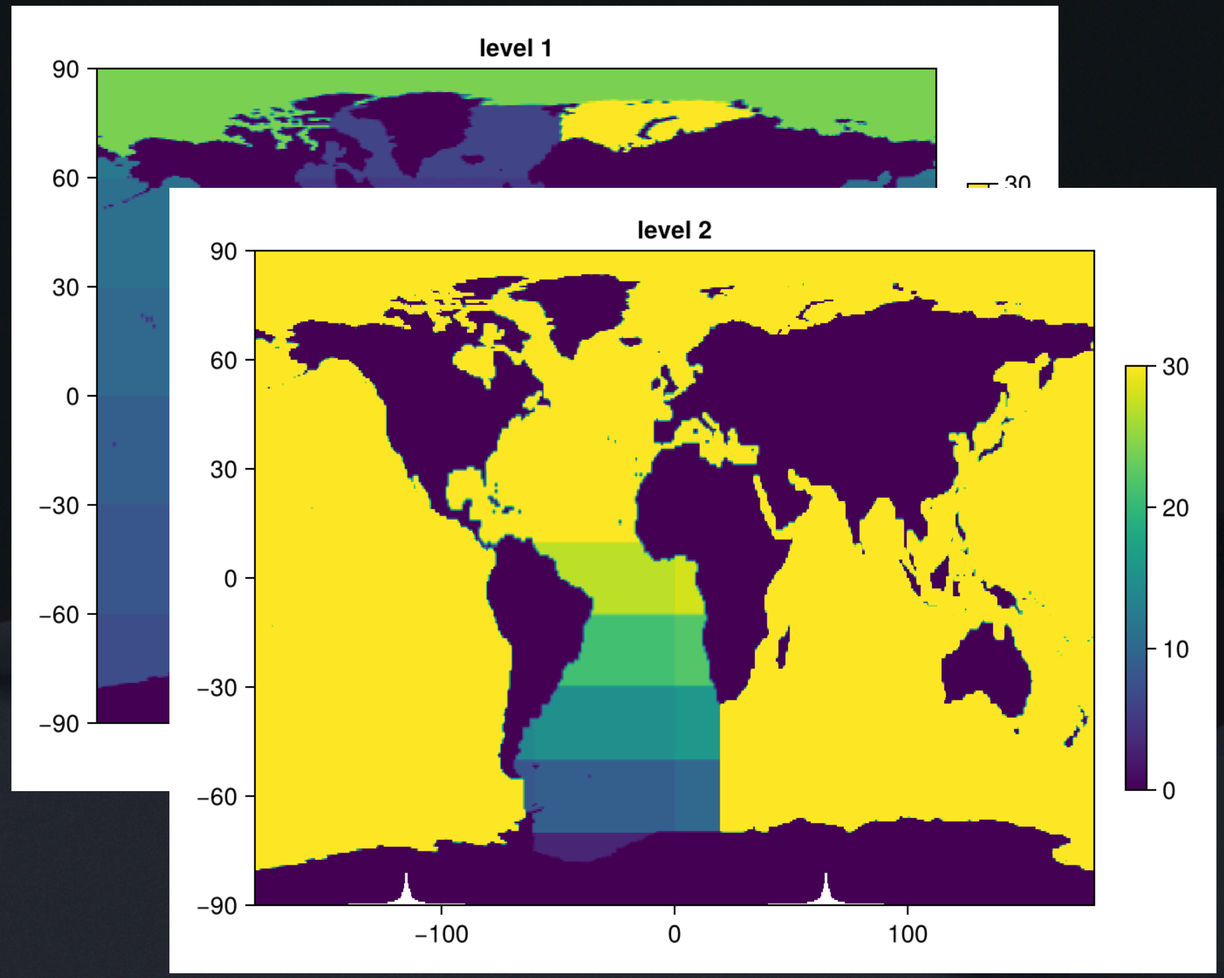
Using and Creating Gridded Datasets

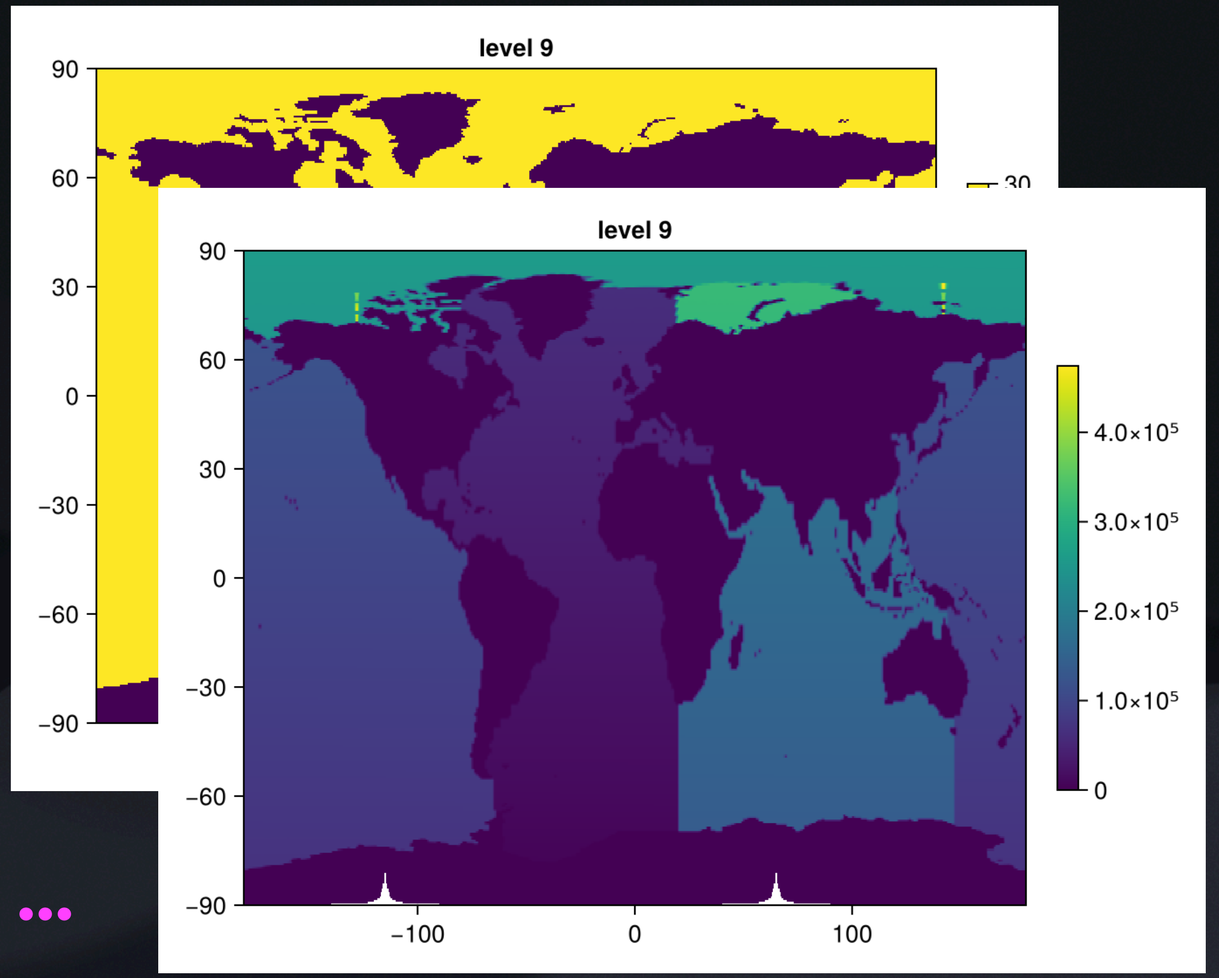
OCCA2 workflow — MITgcm.jl + OceanRobots.jl + Climatology.jl + ...

- Get ERA5 forcing over 1960-present
- Compute climatological mean difference between ECCO4 and ERA5 forcings
- Rerun ECCO model over 1960-present with this forcing —> OCCA2HR2
- Compute clim. difference between ECCO4 & OCCA2HR2 over 1992-2011 —> OCCA2HR2′
- Compute clim. difference between Argo & OCCA2HR2′ over 2004-present —> OCCA2HR2″
- Compute time variable diff. between Argo & OCCA2HR2″ over 2004-present —> OCCA2

OCCA2 workflow — the geostats part

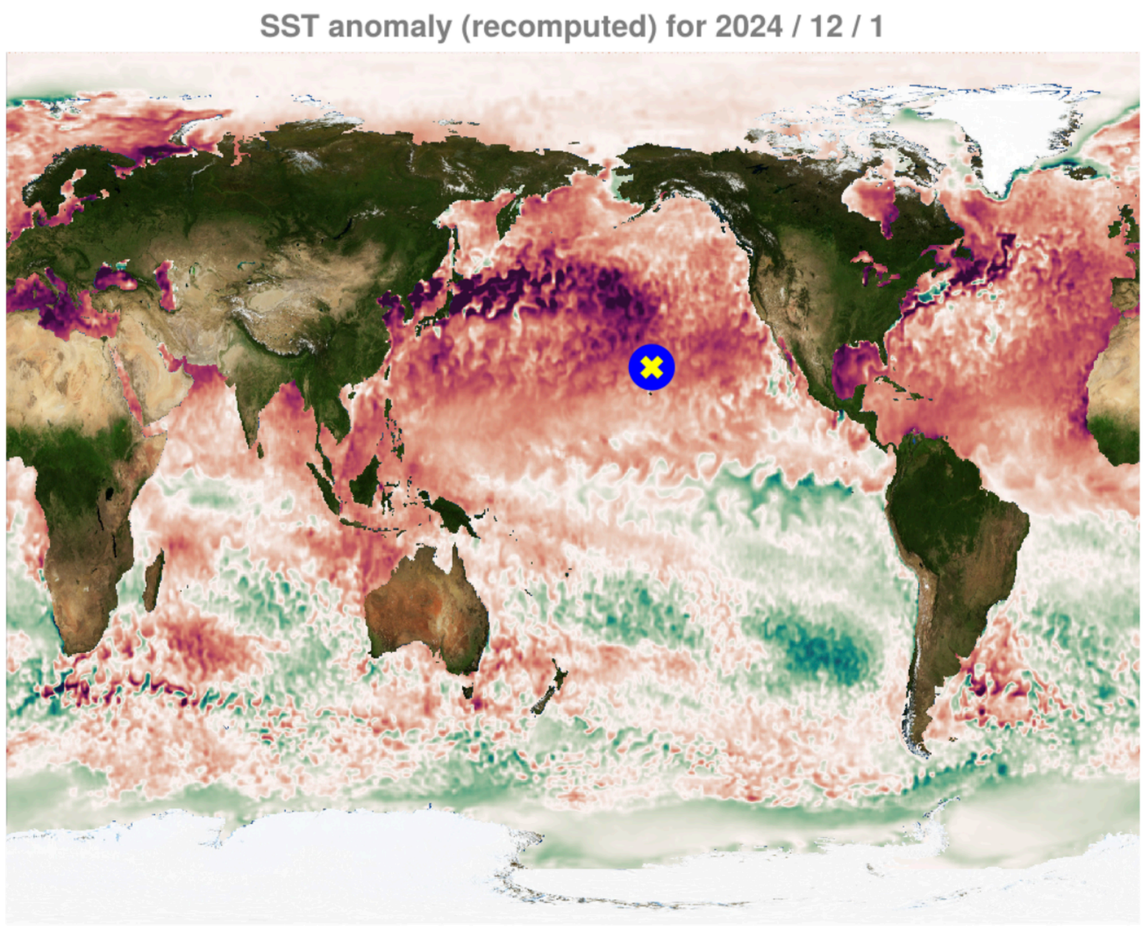
Multigrid sample statistics based on Argo profile collections

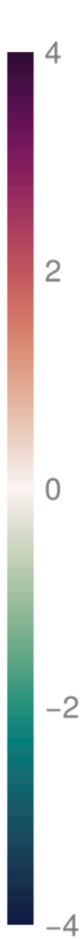


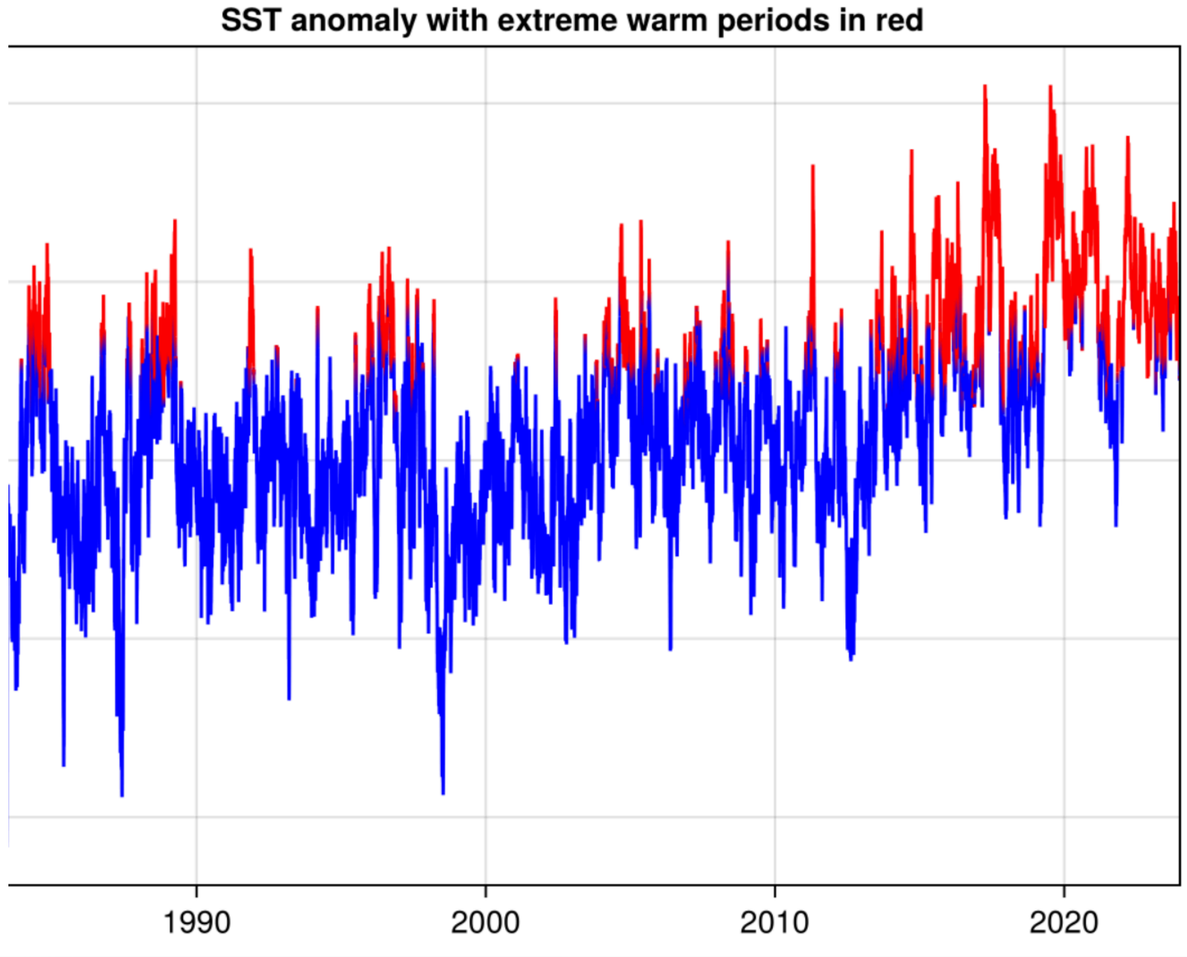


Marine Heat Waves are “increasing”

Tracking them and providing a heat budget for them







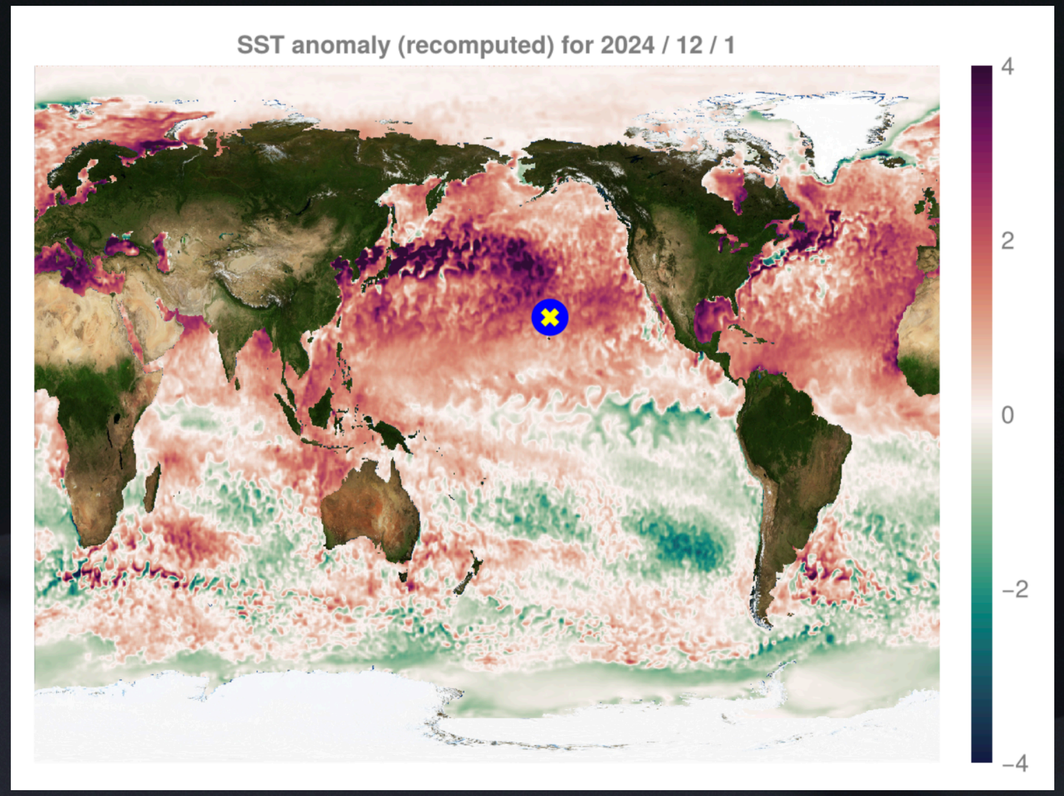
Using and Creating Gridded Datasets

MHW workflow — Climatology.jl + GeometryOps.jl + ...

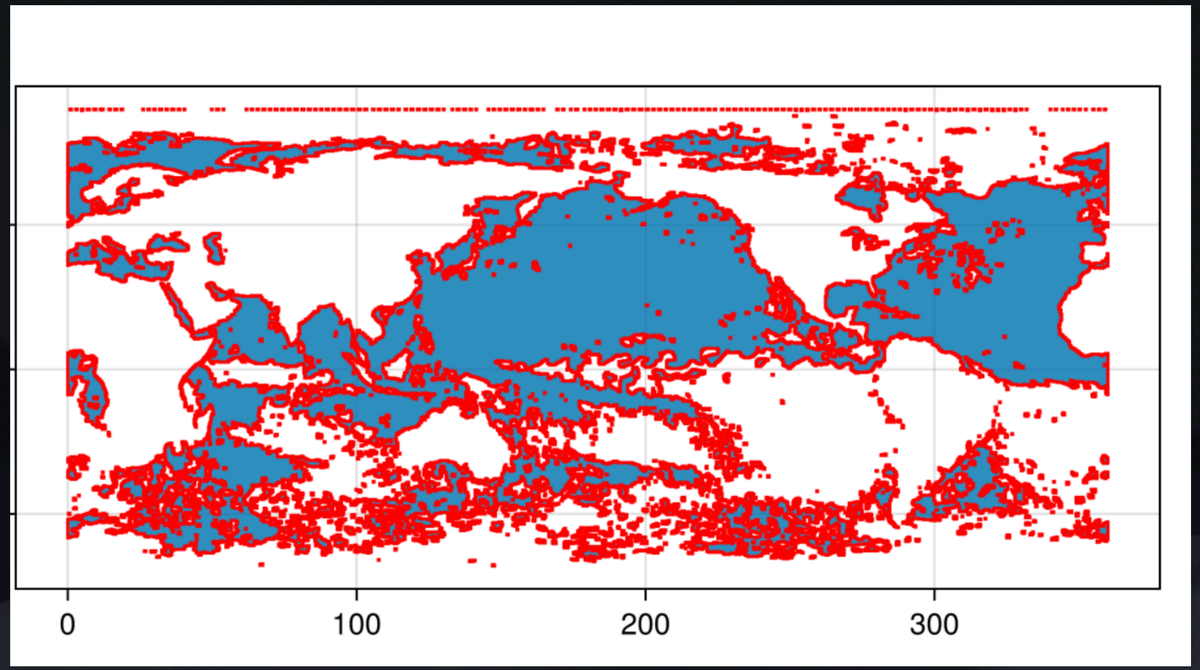
- Download sea surface temperature data collection
- Identify MHW from time series analysis for each location
- Convert MHW mask (raster) into polygon for every time step
- Calculate intersection of polygons between consecutive dates
- Aggregate series of overlapping polygons over duration of MHW event
- Calculate heat budget of MHW event using ECCO4 / OCCA2

MHW workflow — the event tracking part

Polygonize, intersect, … on daily SST map collections



…



…

Outline

- ClimateModels.jl
- MITgcm.jl
- Climatology.jl
- What's next?

What's next?

- More data : Dataverse.jl , STAC.jl , CDSAPI.jl , DataDeps.jl , ODIS ...
- More models : ClimaOcean.jl, SpeedyWeather.jl , ...
- Build model emulators : FAiR, Darwin3, ...
- Leverage established model interfaces : FMI, BMI, ...
- Deeper integration MeshArrays.jl & JuliaGeo ecosystem ...