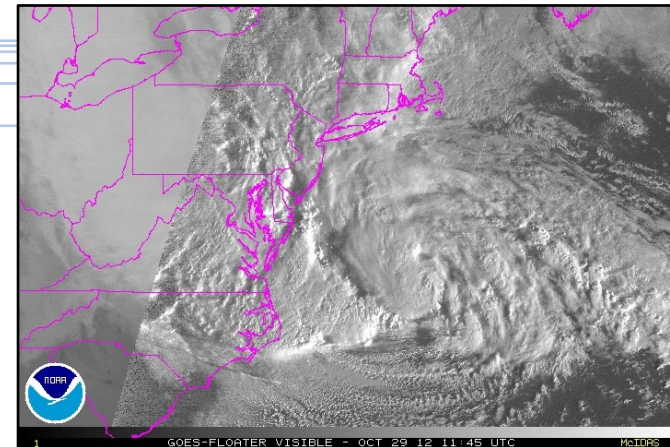
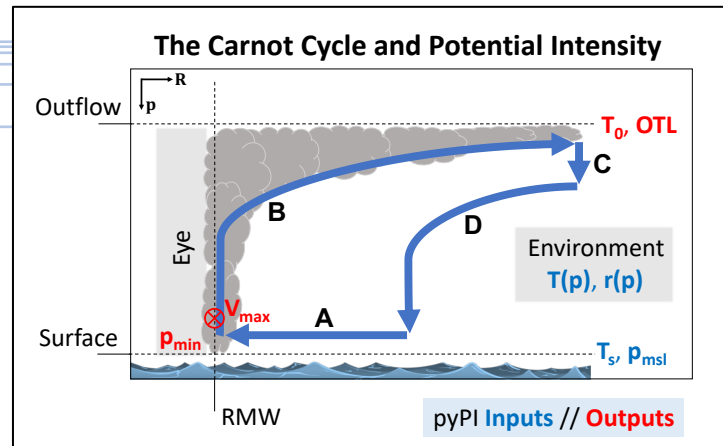


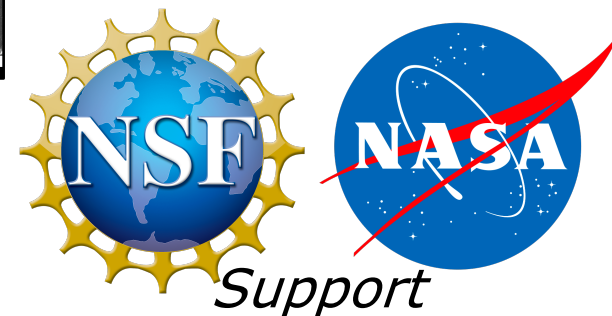
tcpyPI: Tropical Cyclone Potential Intensity Calculations in Python

Daniel Gilford, PhD ✉ dgilford@climatecentral.org 🐦 [@danielgilford](https://twitter.com/danielgilford)

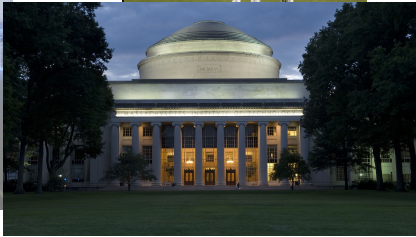
Contributors+Collaborators: Kerry Emanuel, Daniel Rothenberg, Allison Wing, Susan Solomon, Dan Chavas, Jonathan Lin, Raphael Rousseau-Rizzi, and maybe you?!?



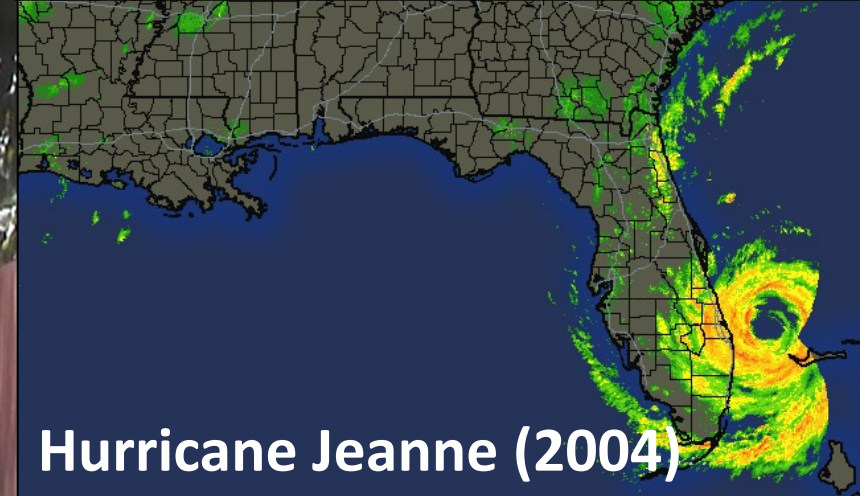
Institute of Earth, Ocean, and
Atmospheric Sciences



Who am I? And why do I care about hurricanes?

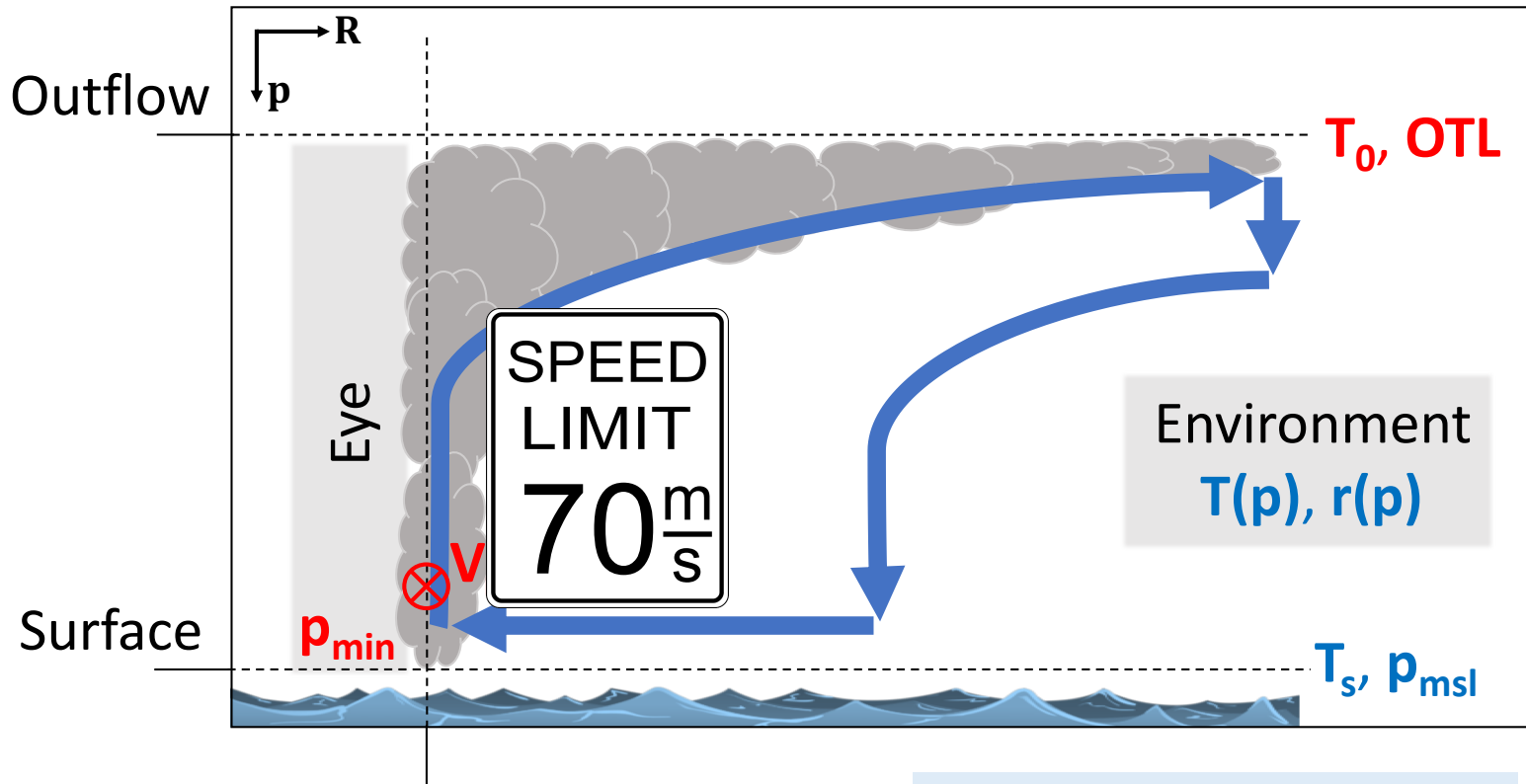


- I call Florida home
- Meteorologist, Climate Scientist
- CoCoRaHS observer
(FL-SM-38)
- Relatively new to open science and big data science



Potential Intensity (PI) is a TC's maximum speed given environmental conditions

The Carnot Cycle and Potential Intensity



Gilford 2021 (GMD), Emanuel (2006)

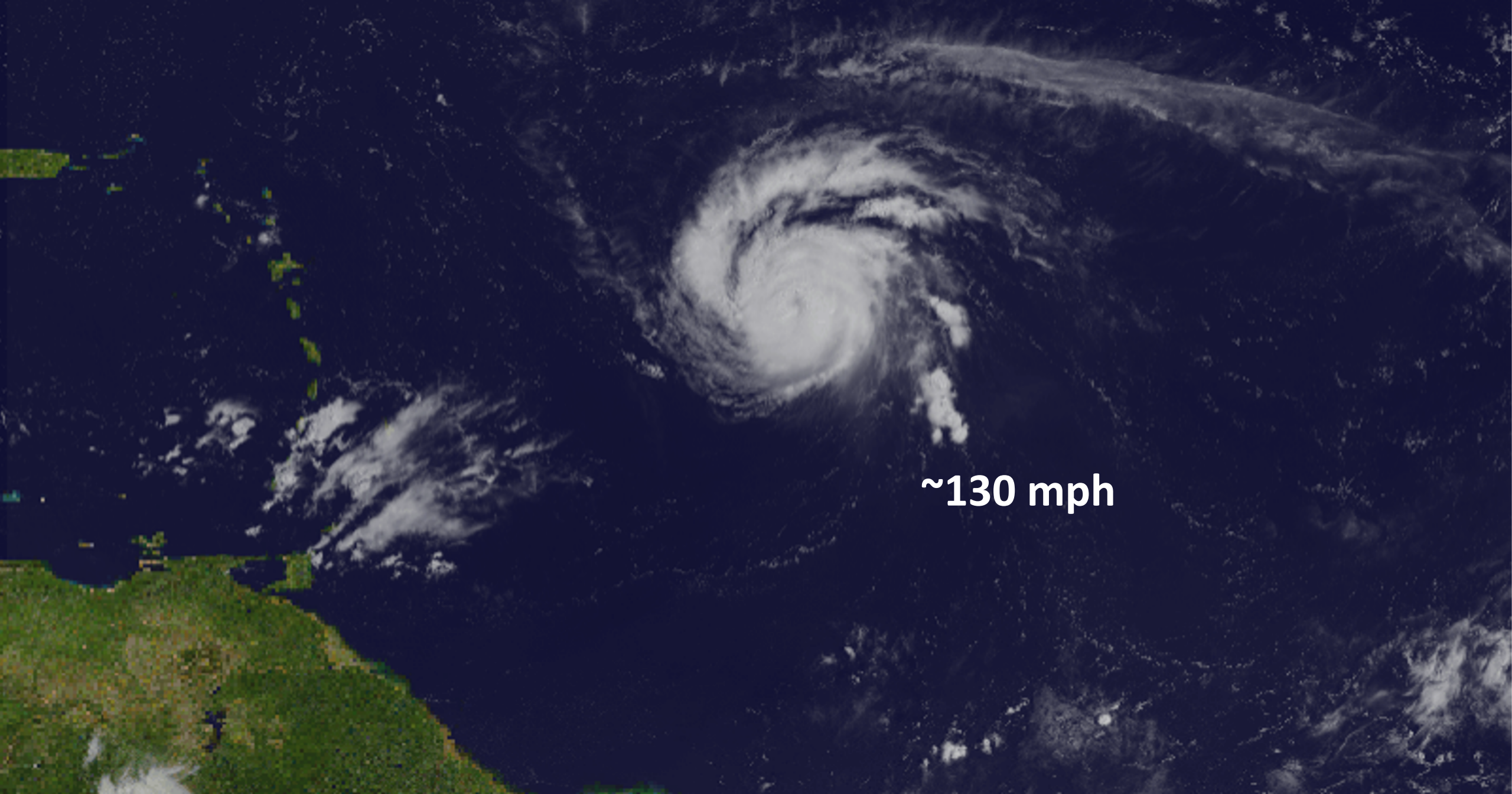
pyPI **Inputs** // **Outputs**

Thermodynamic
Disequilibrium

$$PI^2 = c * \underbrace{\frac{T_s - T_0}{T_0}}_{\text{Thermodynamic Efficiency}} * \underbrace{E}_{\text{Thermodynamic Disequilibrium}}$$

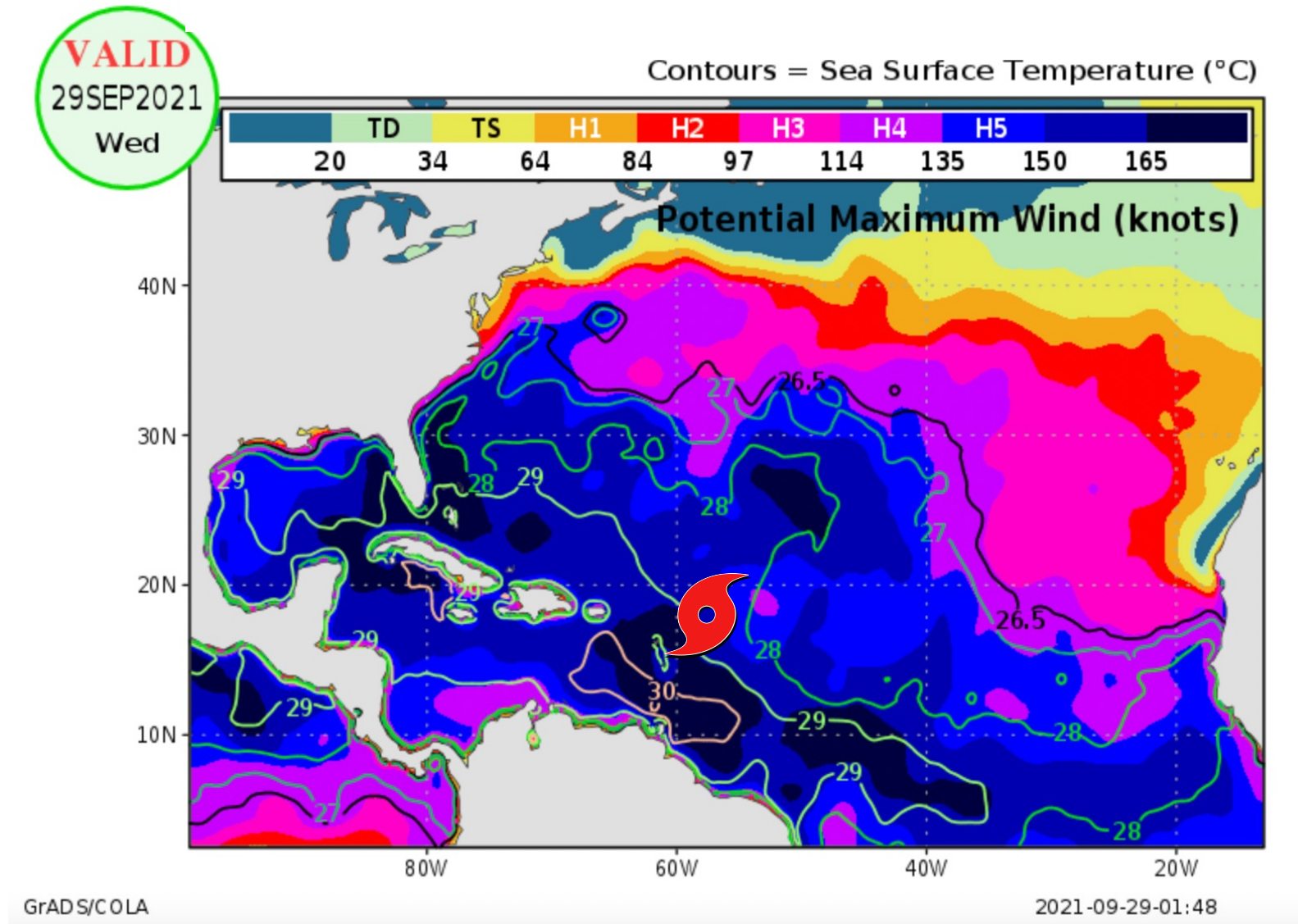
What causes PI to increase?

- Warmer SSTs (T_s)
- Cooler outflow temperatures (T_0) at the outflow temperature level (**OTL**)
- Increases in thermodynamic disequilibrium (largely through T_s)



~130 mph

A “realtime” example of PI calculations: Hurricane Sam



Why is tcpyPI needed?

pyPI Goals:

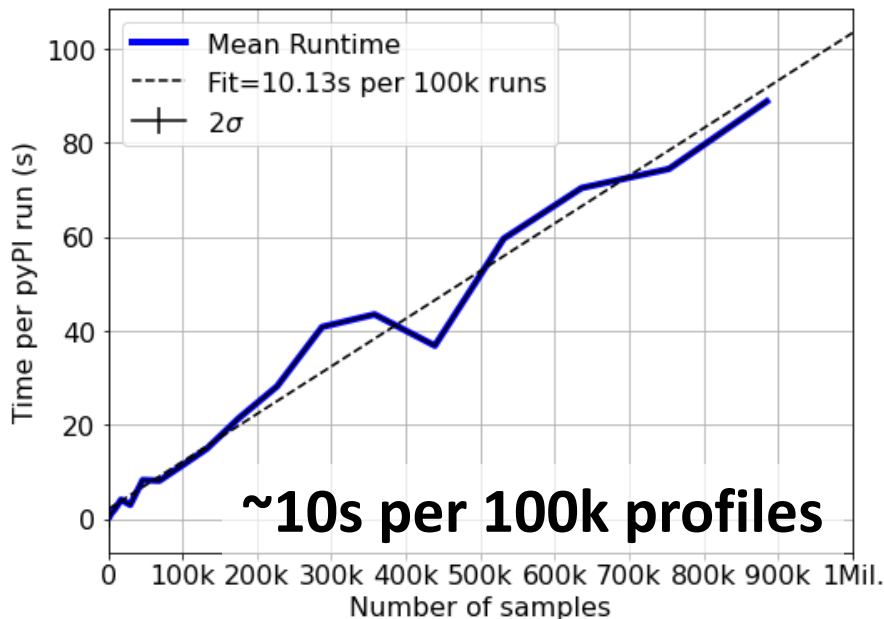
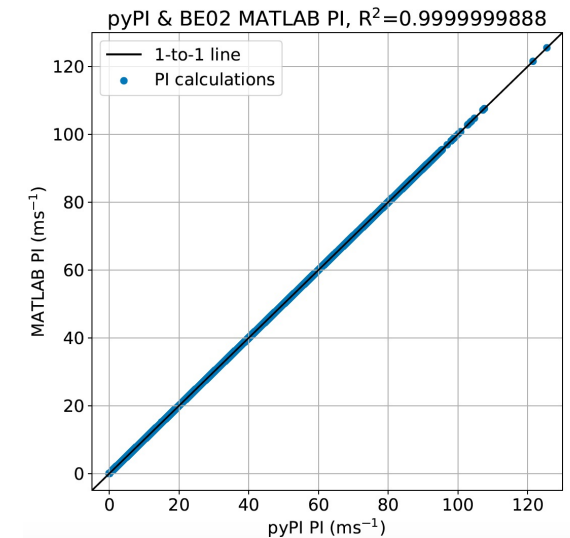
- Supply a freely available and validated Python PI calculator
- Document the PI algorithm and its Python implementation
- Demonstrate & encourage use of PI theory in TC analyses

The [*tcpyPI Git repository*](#) includes the algorithm, utility files, sample run code, and example analyses in Jupyter notebooks

tcpyPI is accurate, but maybe not fast enough

Validation: pyPI code improves on the Bister and Emanuel (2002) formulation by fixing minor errors and offering consistent handling of missing data

→ **tcpyPI is sufficiently accurate for TC research applications**

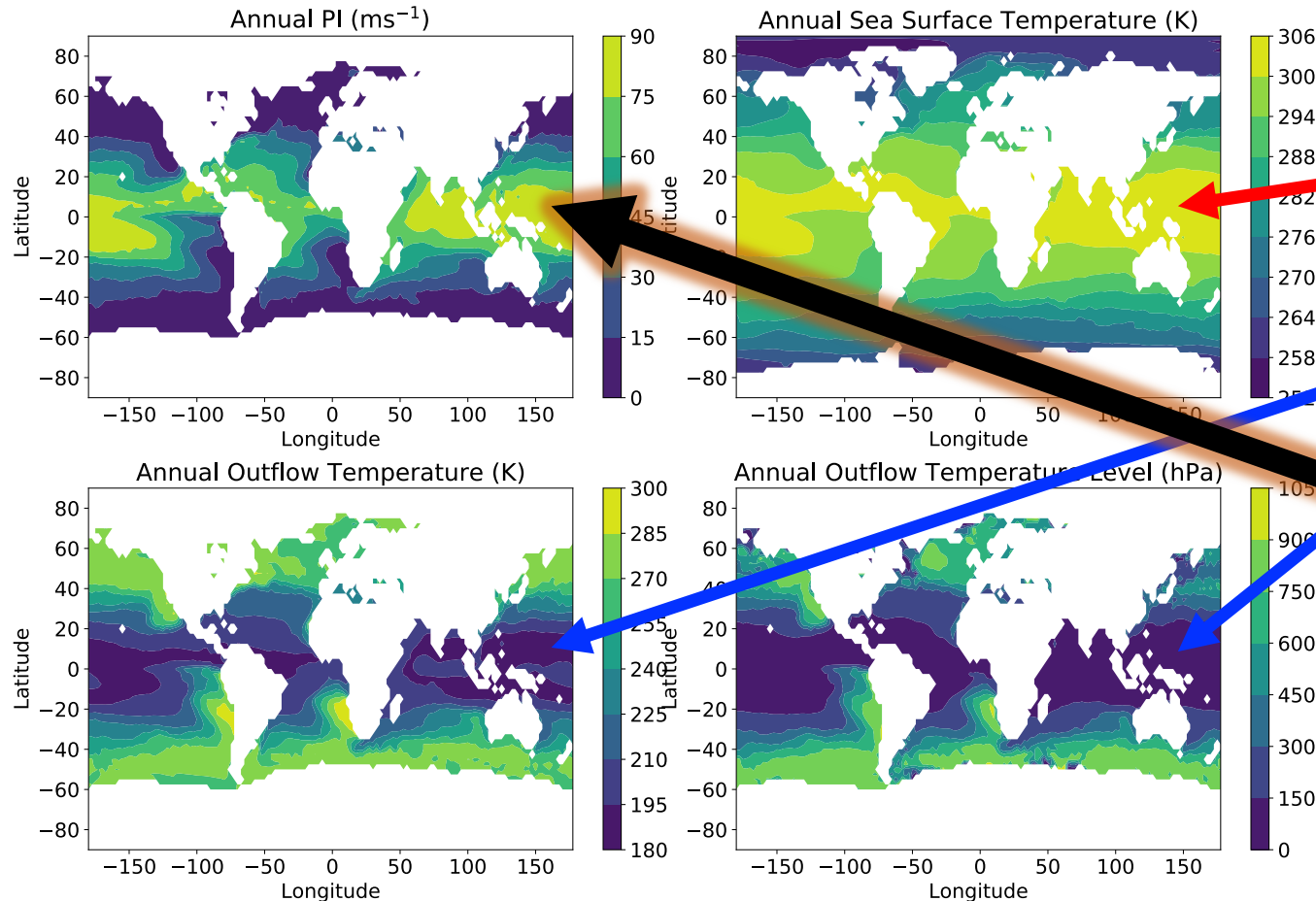


pyPI uses **numba** to optimize PI computations; climatological PI may be computed on gridded data using the **xarray** package

Average run time for one year of monthly $0.25^\circ \times 0.25^\circ$ ERA5 data is **~21 mins.**

tcpyPI in Action: Climatological Potential Intensities

2004 Annual Mean Outputs via 2.5° x 2.5° MERRA2 Inputs:



Warm SSTs

Cold/High Outflow

**Fast Speed
Limits**

For more e.g., see Gilford et al. (2017, 2019)

Flow and Contents of tcpyPI + the Git repository

User interface through Python function:

$(V_{MAX}, P_{MIN}, IFL, T_O, LNB) = \text{pi}(SST, MSL, P, T, R)$

Outputs

Environmental Conditions

Potential Intensity Algorithm (**pi**)

Algorithm 1 pyPI's iterative procedure to calculate tropical cyclone potential intensity.

```
input:  $T_s, p_{msl}, p, T(p), r(p)$ 
1: Check that inputs are appropriate
2: Calculate  $CAPE_{env}$ 
repeat
3: Calculate  $CAPE|_{RMW}$ 
4: Calculate  $CAPE^*, OTL, T_0$ 
5: Estimate  $p_m$  at this  $i^{th}$  iteration
until convergence. Objective:  $\Delta_{i,i-1} p_m < 0.5$  hPa
6: Calculate final  $p_{min}$ 
7: Calculate  $V_{max}$ 
return  $V_{max}, p_{min}, OTL, T_0$ , algorithm flag
```

CAPE Algorithm (**cape**)

Algorithm 2 pyPI's procedure to calculate convective available potential energy.

```
input:  $T_j=0, r_j=0, p_j=0, T(p), r(p), p$ 
1: Calculate parcel  $s$ 
2: Find  $p_{LCL}$ 
for each  $j^{th}$  pressure level do
  if  $p_j \geq p_{LCL}$  then
3: Calculate  $(T_p - T_{p,env})$  lifting the parcel along a dry adiabat
  else if  $p_j < p_{LCL}$  then
    repeat
4: Solve for  $T_j$  at this  $k^{th}$  iteration, conserving  $s$ 
    until convergence. Objective:  $\Delta_{k,k-1} T < 0.001$  K
5: Calculate final  $r_j$  dependent on the ascent_flag
6: Calculate  $(T_p - T_{p,env})$  lifting the parcel along a moist adiabat
    end if
  end for
7: Find LNB,  $T_{LNB}$ 
8: Calculate CAPE
return  $CAPE, T_{LNB}, LNB$ , algorithm flag
```

Additional Research Tool Functions:

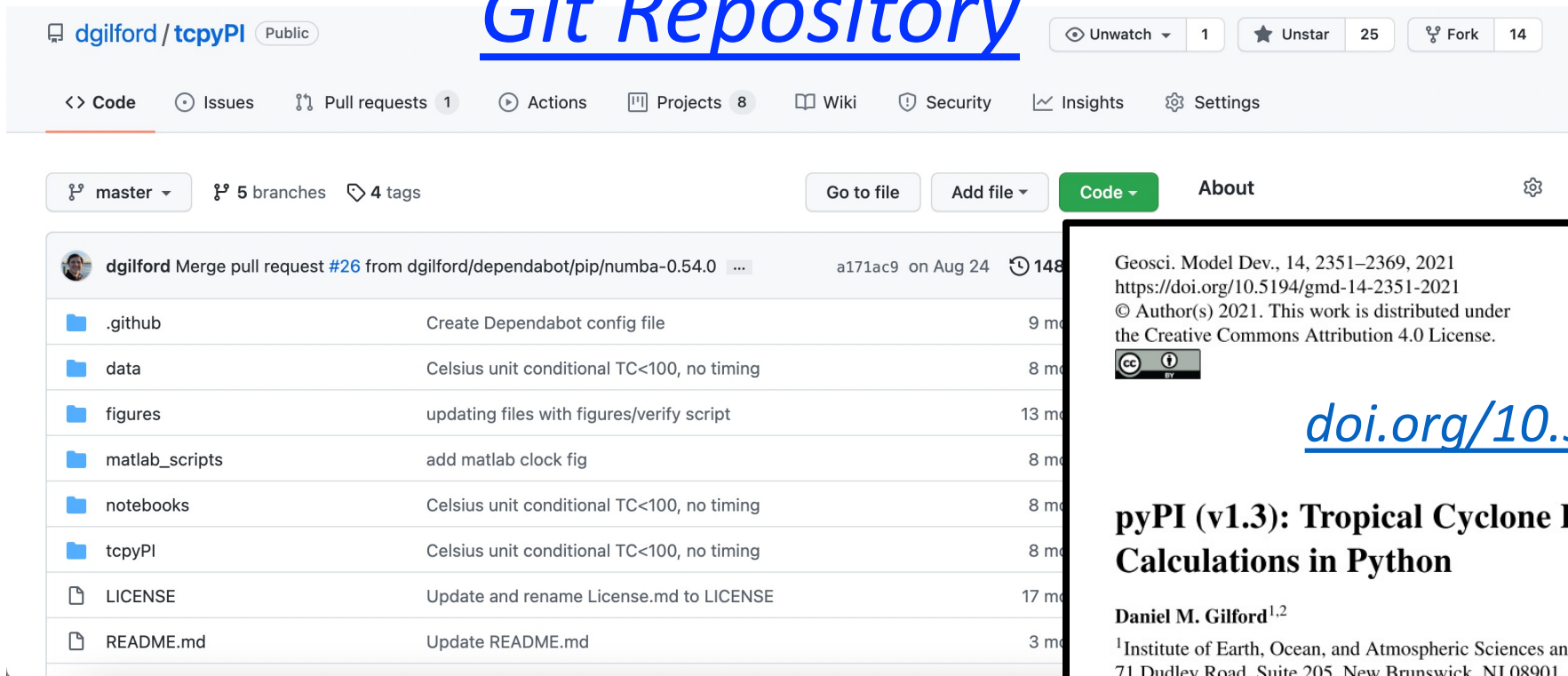
- Efficiency Calculations
- PI Decomposition
- Thermodynamic+profile calculations

Helpful Notebooks:

- Sample PI calculations
- Code interfacing w/ xarray
- Sensitivity Tests
- Runtime Estimation

tcpyPI Documentation and Publication

Git Repository



dgilford / tcpyPI Public

Unwatch 1 Unstar 25 Fork 14

<> Code Issues Pull requests 1 Actions Projects 8 Wiki Security Insights Settings

master 5 branches 4 tags

Go to file Add file Code About

File	Description	Last Commit
.github	Create Dependabot config file	9 mo
data	Celsius unit conditional TC<100, no timing	8 mo
figures	updating files with figures/verify script	13 mo
matlab_scripts	add matlab clock fig	8 mo
notebooks	Celsius unit conditional TC<100, no timing	8 mo
tcpyPI	Celsius unit conditional TC<100, no timing	8 mo
LICENSE	Update and rename License.md to LICENSE	17 mo
README.md	Update README.md	3 mo

Geosci. Model Dev., 14, 2351–2369, 2021
https://doi.org/10.5194/gmd-14-2351-2021
© Author(s) 2021. This work is distributed under
the Creative Commons Attribution 4.0 License.



Geoscientific
Model Development
Open Access
EGU

doi.org/10.5194/gmd-14-2351-2021

pyPI (v1.3): Tropical Cyclone Potential Intensity Calculations in Python

Daniel M. Gilford^{1,2}

¹Institute of Earth, Ocean, and Atmospheric Sciences and Department of Earth and Planetary Sciences, Rutgers University,
71 Dudley Road, Suite 205, New Brunswick, NJ 08901, USA

²Climate Central, Princeton, NJ, USA

Correspondence: Daniel M. Gilford (daniel.gilford@rutgers.edu)

Received: 18 August 2020 – Discussion started: 20 October 2020

Revised: 5 March 2021 – Accepted: 21 March 2021 – Published: 3 May 2021

Possible Scientific + Research Improvements to tcpyPI

- Include additional potential intensity measures/algorithms in the same package


Enable users to choose and/or compare PI calculations for the same environment

- Add user options on physical assumptions, e.g.:

LCL-pressure calculation, representation of outflow temperature, disequilibrium

- Include additional tropical cyclone indices, such as the genesis potential index (Camargo et al., 2007) and ventilation index (Tang and Emanuel, 2012)

Possible Code + Functionality Improvements to tcpyPI

- Incorporate [pint](#)  for unit support
- Develop an xarray compatibility layer:

Wrap tcpyPI functions using `xarray.apply_ufunc()`. This would enable direct input/output of xarray DataArrays within tcpyPI
- New ways to reduce runtimes? e.g. Potential improvements in the handling of missing data, or iteration convergence criteria

→ Some change is likely necessary to enable operationalization

I welcome feedback, and thanks for listening!!

Daniel Gilford, PhD

 dgilford@climatecentral.org  @danielgilford



Support From:

