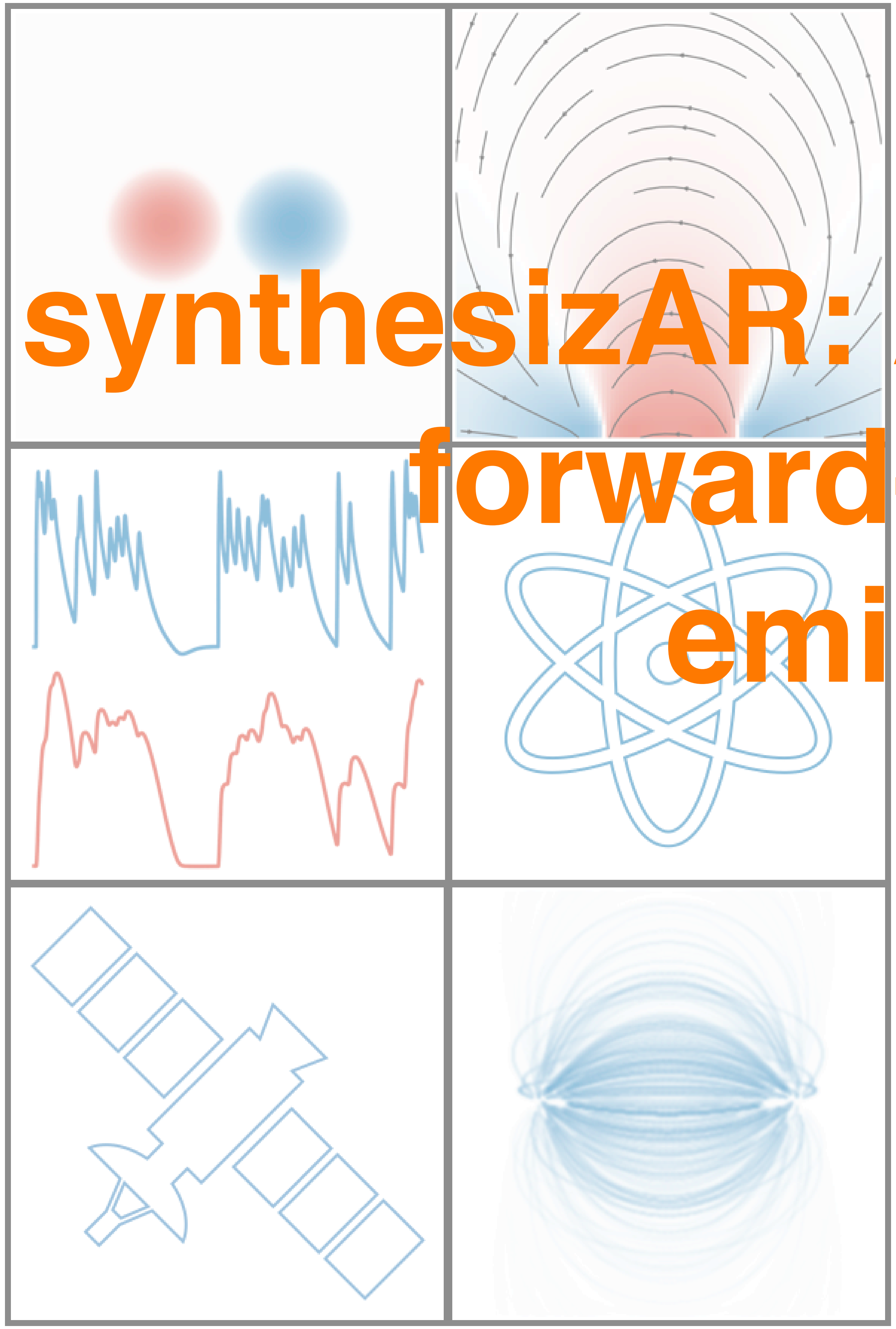


synthesizAR: A Python framework for forward-modeling optically-thin emission from field-aligned hydrodynamic models

Coronal Heating ISFM Meeting / NASA GSFC / 3 April 2024

Will Barnes
AU/NASA GSFC

wtbarnes.github.io/klimchuk-isfm-2024-talk

sunpy powered

Modeling Optically-thin Emission

- Modeling optically-thin intensity, I, requires computing the following line-of-sight integral:

$$I = \int \mathrm{d}h\, \varepsilon(n, T),$$

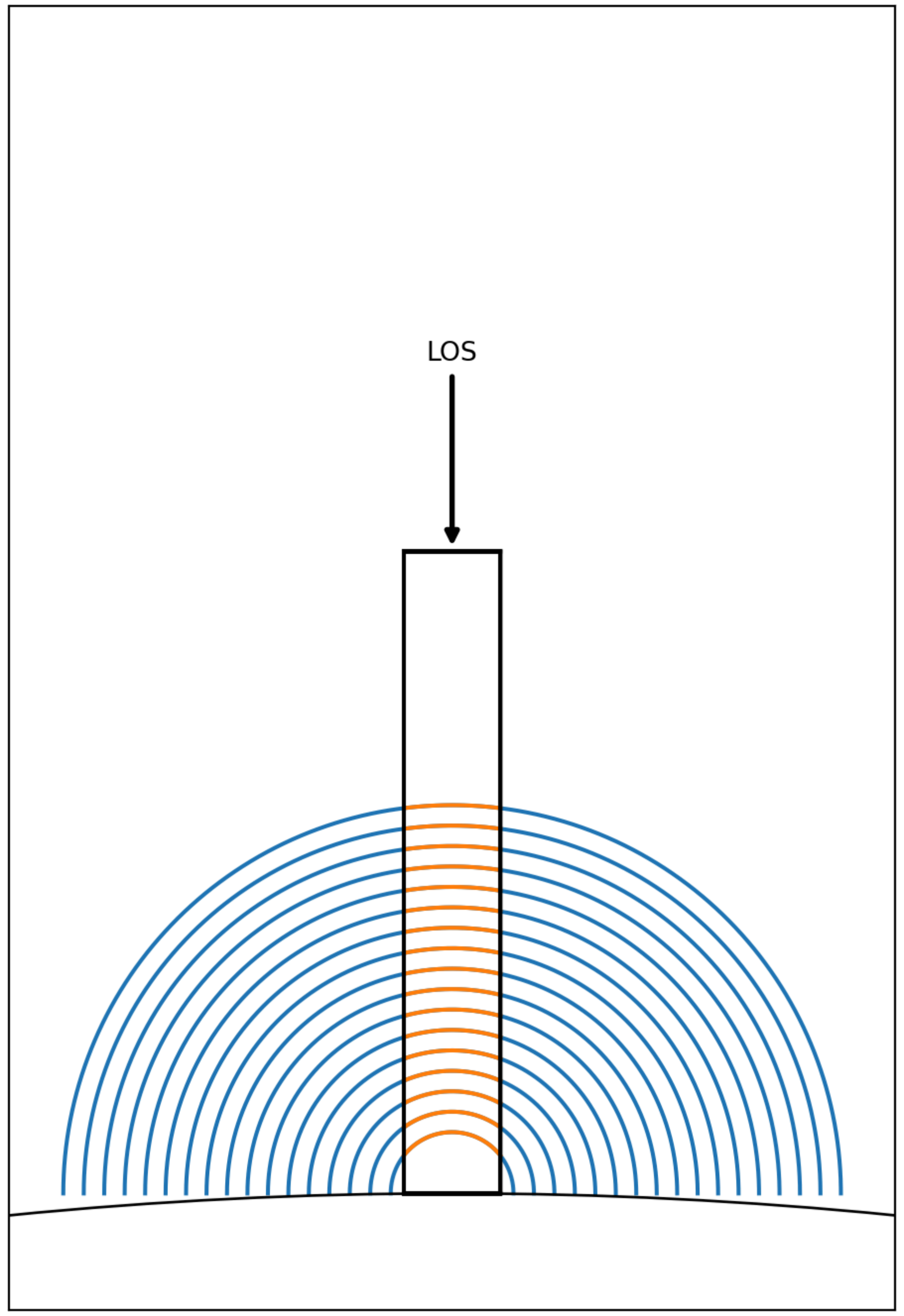
- where:
  - h is a vertical coordinate along the line of sight (LOS)
  - ε is the emissivity
  - $n \equiv n(h), T \equiv T(h)$ are the temperature and density along the LOS
- This requires knowing four things:
  1. What structures are emitting–**geometry**
  2. T, n of emitting structures as a function of space and time–**loop model**
  3. How the plasma is emitting as a function of n, T–**emission model**
  4. How is the emission being observed–**instrument**

Modeling Optically-thin Emission

$$I = \int \mathrm{d}h\,\varepsilon(n,T)$$

$$\int \mathrm{d}A\,I = \int \mathrm{d}A \int \mathrm{d}h\,\varepsilon(n,T) = \int \mathrm{d}V\,\varepsilon(n,T)$$

$$I \approx \frac{1}{A_{\mathrm{pix}}} \int \mathrm{d}V\,\varepsilon(n,T)$$



- Emission confined to discrete loops intersecting LOS
- Loop segments have cross-section A_s and length δ_s

$$\int \mathrm{d}V\,\varepsilon(n,T) = \sum_s \int \mathrm{d}V_s \varepsilon_s(n,T) \approx \sum_s A_s \delta_s \varepsilon_s(n,T)$$

$$I \approx \frac{1}{A_{\mathrm{pix}}} \sum_s A_s \delta_s \varepsilon_s(n,T)$$

wtbarnes.github.io/klimchuk-isfm-2024-talk

sunpy powered

The Emissivity kernel

- Compute per loop segment using field-aligned model–$T \equiv T(s,t), n \equiv n(s,t)$
- For a narrowband imager (e.g. AIA),

$$\varepsilon = K(T)n^2 \quad [\mathrm{DN\,pix^{-1}\,s^{-1}\,cm^{-1}}]$$

- For a spectral line intensity with a transition at λ_{ij},

$$\begin{aligned}\varepsilon &= \frac{1}{4\pi}A_{ij}\frac{hc}{\lambda_{ij}}n_{ij} \quad [\mathrm{erg\,cm^{-3}\,s^{-1}\,sr^{-1}}] \\ &= \frac{1}{4\pi}A_{ij}\frac{hc}{\lambda_{ij}}\frac{n_{ij}}{n_{X^+}}\frac{n_{X^+}}{n_X}\frac{n_X}{n_H}\frac{n_H}{n_e}n_e = \frac{1}{4\pi}G_{ij}(n,T)n_e^2\end{aligned}$$

- For the emission measure distribution on the interval $T_a \leq T < T_b$,

$$\varepsilon = n^2 H(T - T_a)H(T_b - T) \quad [\mathrm{cm^{-6}}]$$

The synthesizAR Package

- synthesizAR = synthesis of Active Region emission (pronounced like "synthesizer")
- Combine field-aligned models to produce spatially-resolved, time-dependent forward model
- Strengths:
  - **Modular**–geometry, field-aligned model, instrument all configurable
  - **Modern**–pure Python, leverages scientific Python and "PyAstro" software stack
  - **Efficient**–emissivity from each strand computed in parallel
  - **Dynamic**–forward-modeled emission is *time-dependent*
- Limitations:
  - All emission is assumed to be **optically-thin**
  - All emission is assumed to be **thermal** (i.e. no transport effects)
  - All emission confined to **discrete field lines** (i.e. not volume filling)

More details in Barnes et al. (2019)

github.com/wtbarnes/synthesizAR / synthesizAR.readthedocs.io

sunpy powered

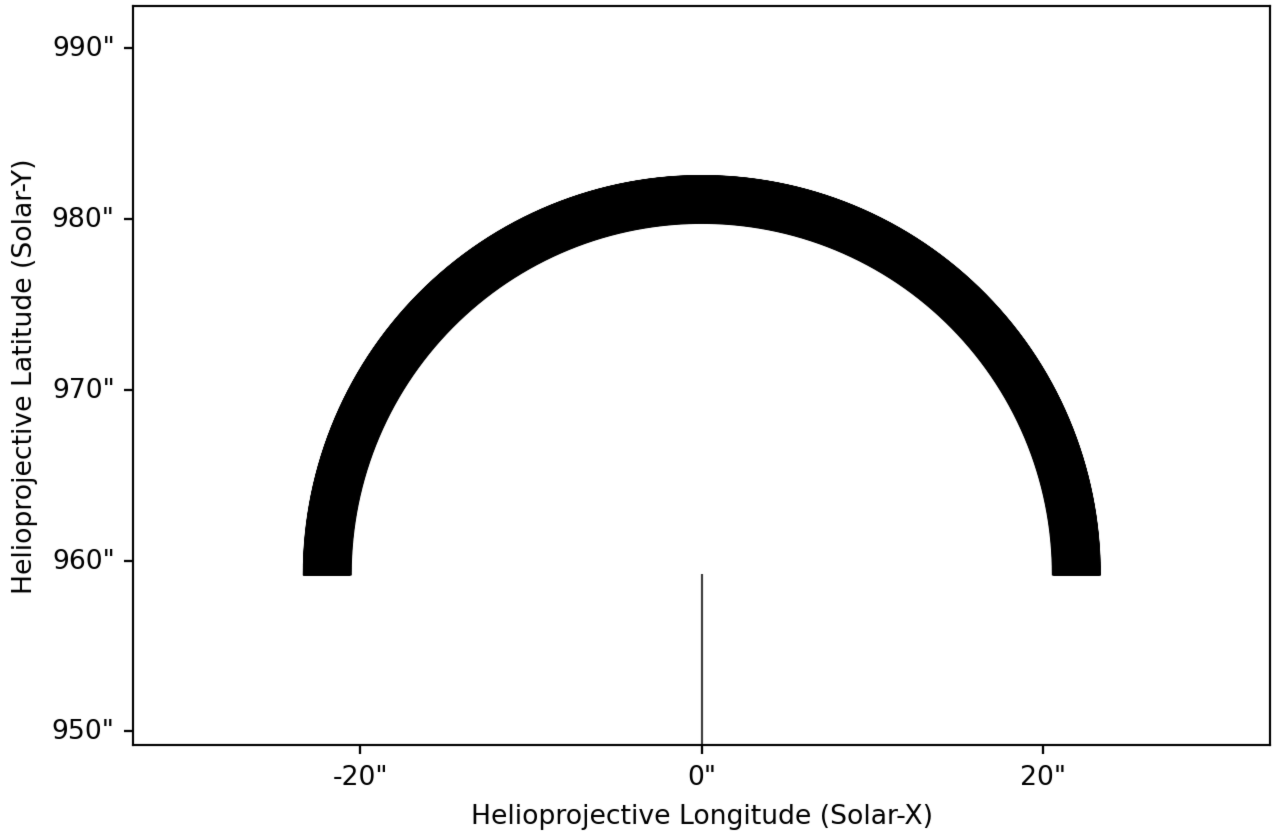
Toy Loop Model

```python
import synthesizAR
from synthesizAR.models import semi_circular_bundle

obstime = astropy.time.Time.now()
pos = SkyCoord(lon=0*u.deg, lat=0*u.deg, radius=1*u.AU, obstime=obstime, frame='heliographic_stonyhurst')
bundle_coords = semi_circular_bundle(50 * u.Mm, 1*u.Mm, 500, observer=pos)
print(bundle_coords[0][:2])
```

```
<SkyCoord (Heliocentric: obstime=2024-04-03 15:02:43.614007, observer=<HeliographicStonyhurst Coordinate (obstime=2024-04-03
15:02:43.614007, rsun=695700.0 km): (lon, lat, radius) in (deg, deg, AU)
    (0., 0., 1.)>): (x, y, z) in Mm
    [(-15.31694891, -0.13318253, 695.7      ),
     (-15.31687318, -0.13318253, 695.7481677)]>
```

```python
strands = [synthesizAR.Loop(f'strand{i}', c) for i, c in enumerate(bundle_coords)]
bundle = synthesizAR.Skeleton(strands)
side_on_view = SkyCoord(lon=0*u.deg, lat=-90*u.deg, radius=1*u.AU, frame=pos.frame)
bundle.peek(observer=side_on_view)
```



wtbarnes.github.io/klimchuk-isfm-2024-talk

sunpy powered

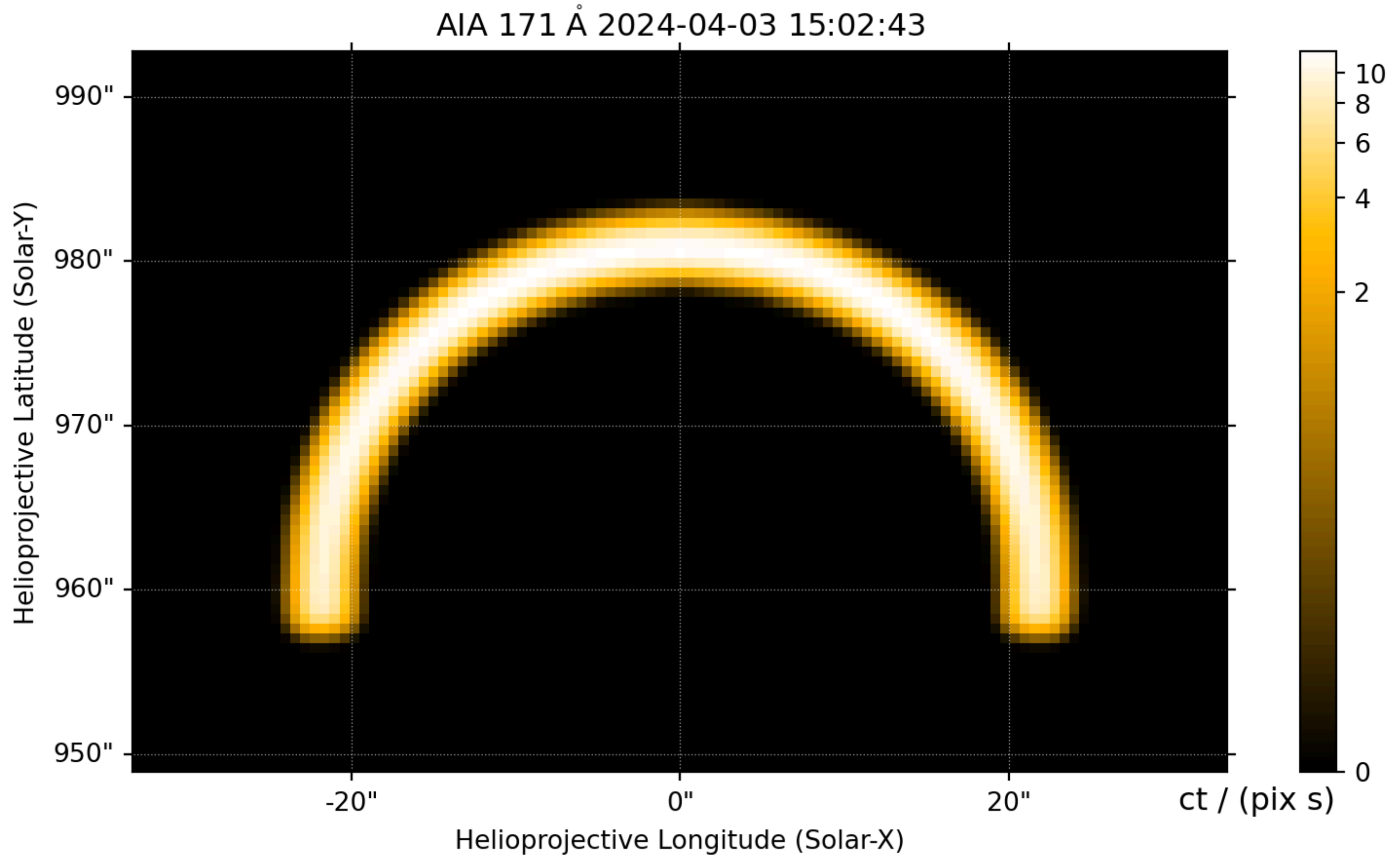
Toy Loop Model

```
from synthesizAR.instruments import InstrumentSDOAIA
from synthesizAR.interfaces import RTVInterface

rtv = RTVInterface(heating_rate=1e-3*u.Unit('erg cm-3 s-1'))
bundle.load_loop_simulations(rtv)
print(bundle.loops[0].electron_temperature[0,:5])
print(bundle.loops[0].density[0,:5])
```

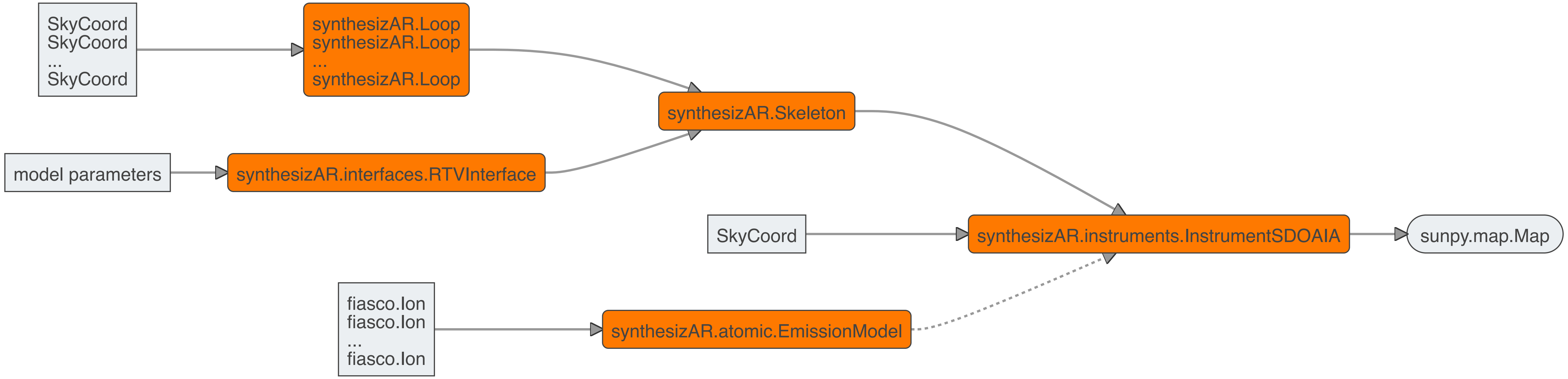
```
[2636831.60572369 2636831.60572369 2636831.60572369 2636831.60572369
 2636831.60572369] K
[1.71094708e+09 1.71094708e+09 1.71094708e+09 1.71094708e+09
 1.71094708e+09] 1 / cm3
```

```
aia = InstrumentSDOAIA([0, 1]*u.s, side_on_view, pad_fov=(10, 10)*u.arcsec)
maps = aia.observe(bundle, channels=aia.channels[2:3])
maps['171'][0].peek()
```



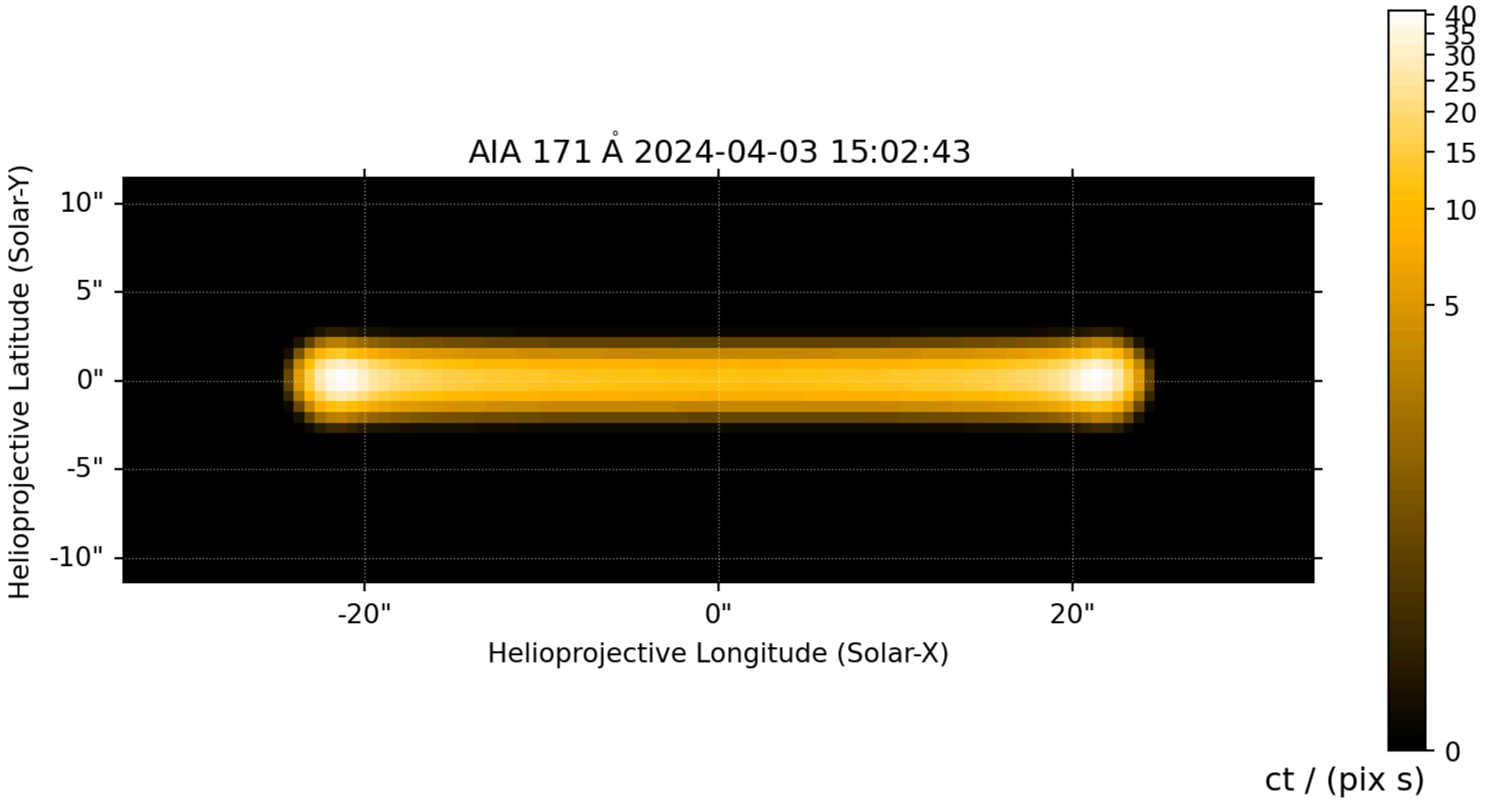
Toy Loop Model: Summary of Steps

1. What structures are emitting–`synthesizAR.Skeleton`, `synthesizAR.Loop`
2. T, n of these structures as a function of s, t–`synthesizAR.interfaces.RTVInterface`
3. How plasma emits as a function of T, n–`synthesizAR.atomic.EmissionModel` (optional)
4. How is the emission being observed–`synthesizar.instruments.InstrumentSDOAIA`

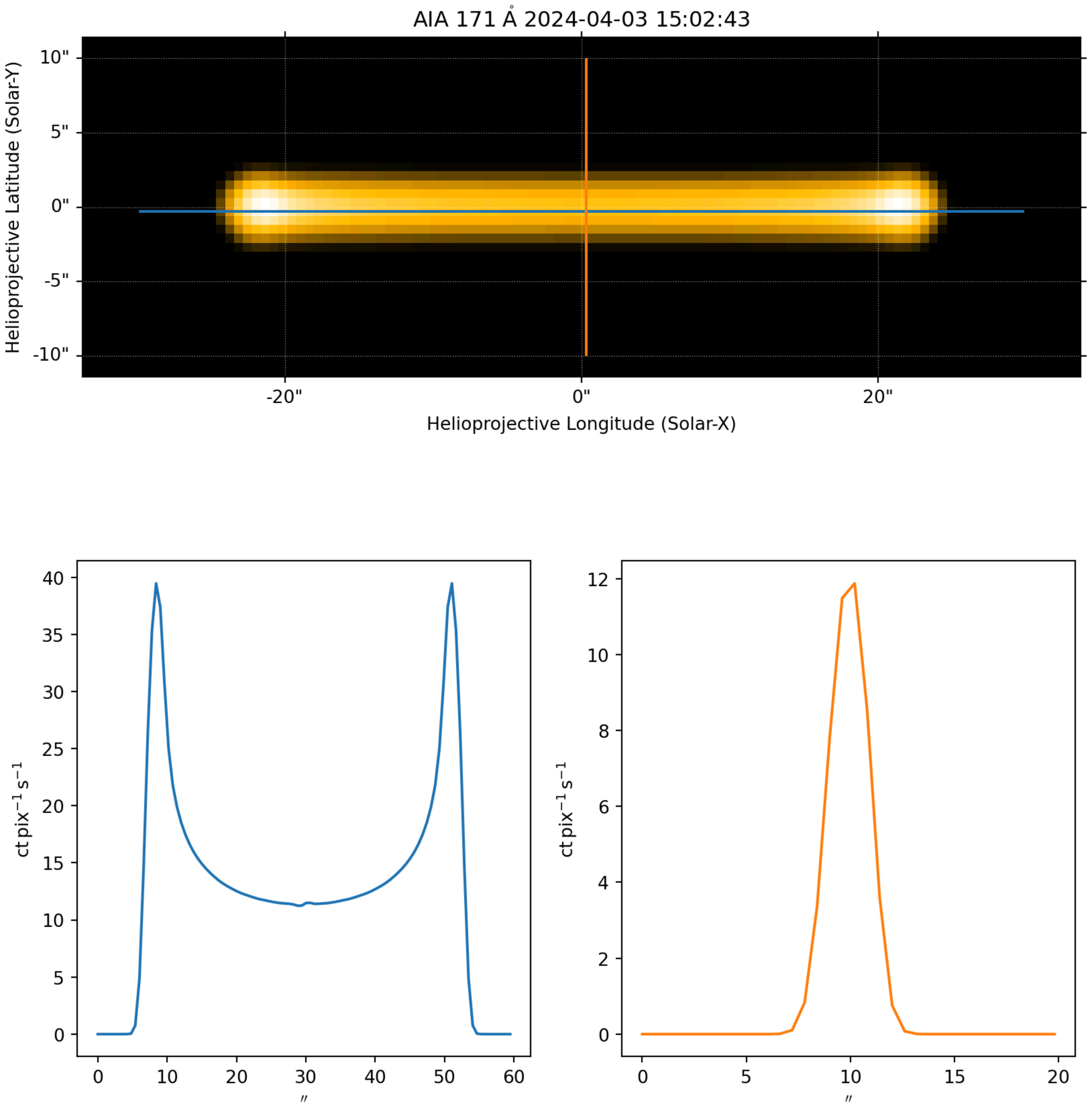


Toy Loop Model: Different Observer

```
top_down_view = SkyCoord(lon=0*u.deg, lat=0*u.deg, radius=1*u.AU, frame=pos.frame)
aia = InstrumentSDOAIA([0, 1]*u.s, top_down_view, pad_fov=(10, 10)*u.arcsec)
maps = aia.observe(bundle, channels=aia.channels[2:3])
maps['171'][0].peek()
```



Toy Loop Model: Different Observer



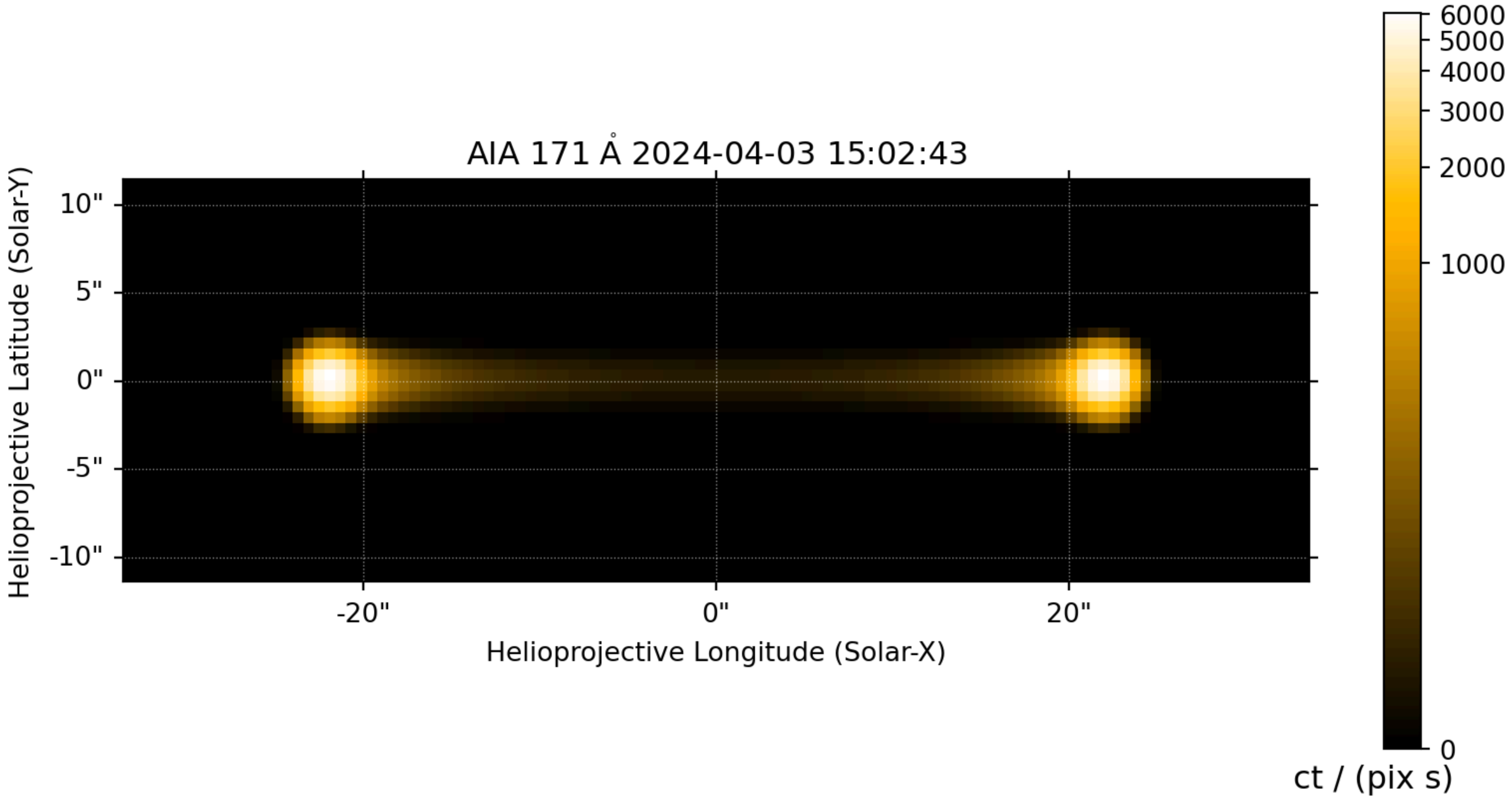
wtbarnes.github.io/klimchuk-isfm-2024-talk

sunpy powered

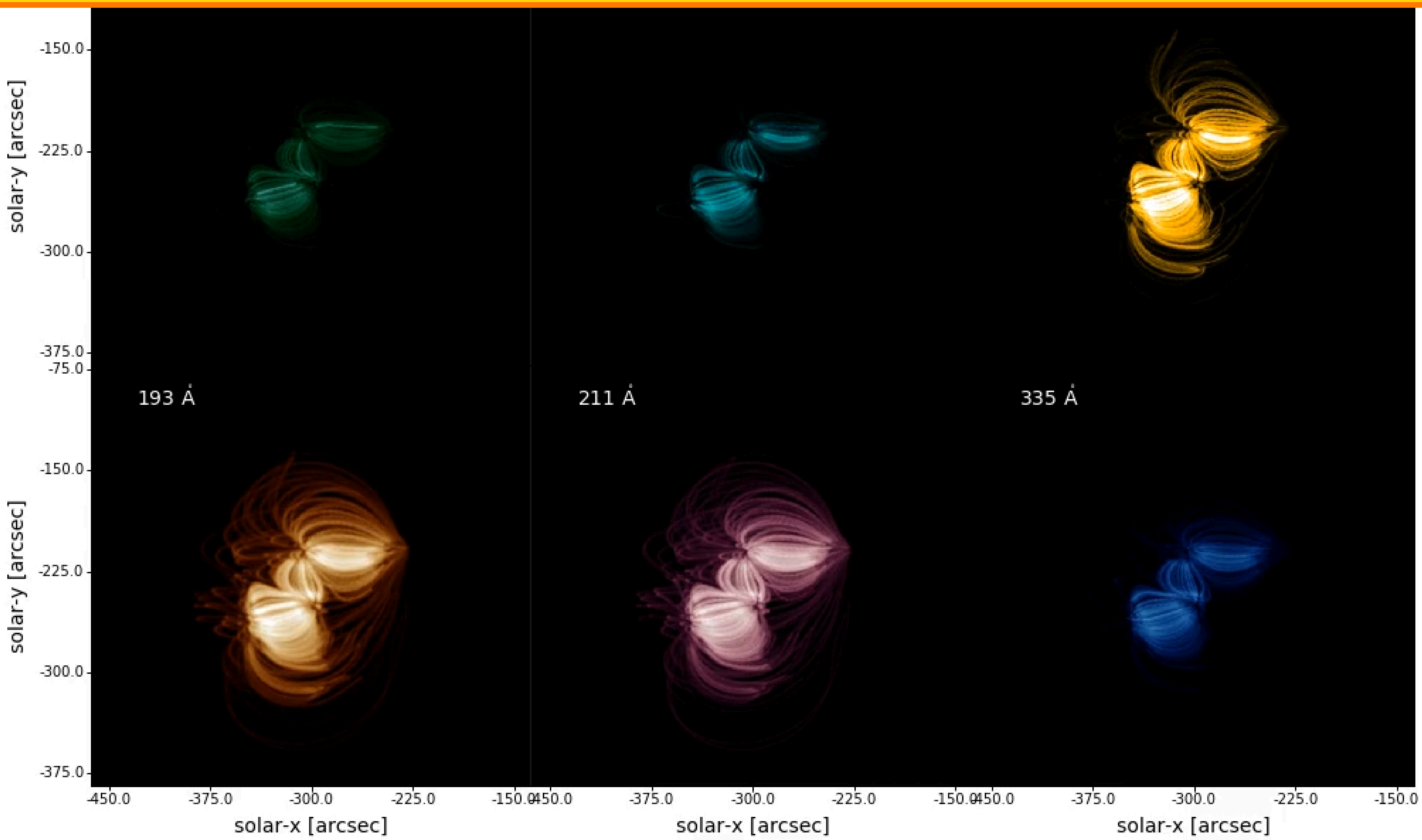
Toy Loop Model: Using a Different Loop Model

```
from synthesizAR.interfaces import MartensInterface

martens = MartensInterface(1e-3*u.Unit('erg cm-3 s-1'))
bundle.load_loop_simulations(martens)
maps = aia.observe(bundle, channels=aia.channels[2:3])
maps['171'][0].peek()
```



Application: Active Region Heating Diagnostics



See Barnes et al. (2019),Barnes et al. (2021)

wtbarnes.github.io/klimchuk-isfm-2024-talk

sunpy powered

Application: Active Region Heating Diagnostics

- Emission measure slope, a,

$$\mathrm{EM}(T) \sim n^2\tau_{rad} \sim T^{1-\alpha}n \sim T^{1-\alpha+1/\ell}$$
$$\mathrm{EM}(T) \propto T^a$$

  - For uninterrupted radiative and enthalpy-driven cooling, $2 \lesssim a \lesssim 2.5$
  - As heating frequency increases, a increases with more isothermal EM

- Time lag τ_{AB} between AIA channels A and B

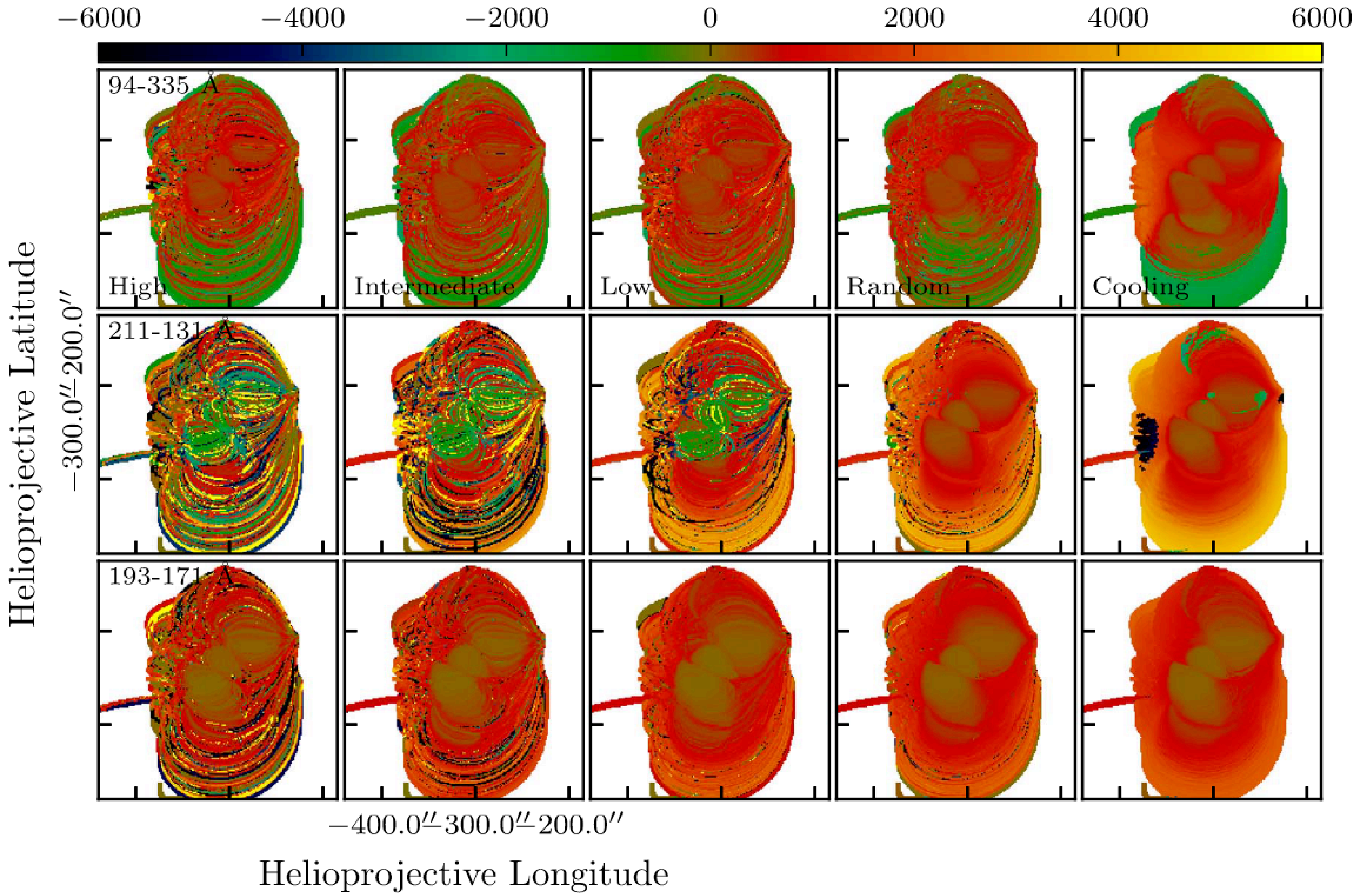
$$\mathcal{C}_{AB}(\tau) = \mathcal{I}_A(t) \star \mathcal{I}_B(t) = \mathscr{F}^{-1}\left\{\mathscr{F}\left\{\mathcal{I}_A(-t)\right\}\mathscr{F}\left\{\mathcal{I}_B(t)\right\}\right\}$$
$$\tau_{AB} = \operatorname{argmax}_\tau \mathcal{C}_{AB}(\tau)$$

  - Proxy for the cooling time between characteristic temperatures of A and B
  - $\tau_{AB} > 0$ indicative of cooling, $\tau_{AB} < 0$ suggests heating

Bradshaw & Cargill (2010), Cargill (2014), Viall & Klimchuk (2012), Barnes et al. (2019)

sunpy powered

Application: Active Region Heating Diagnostics

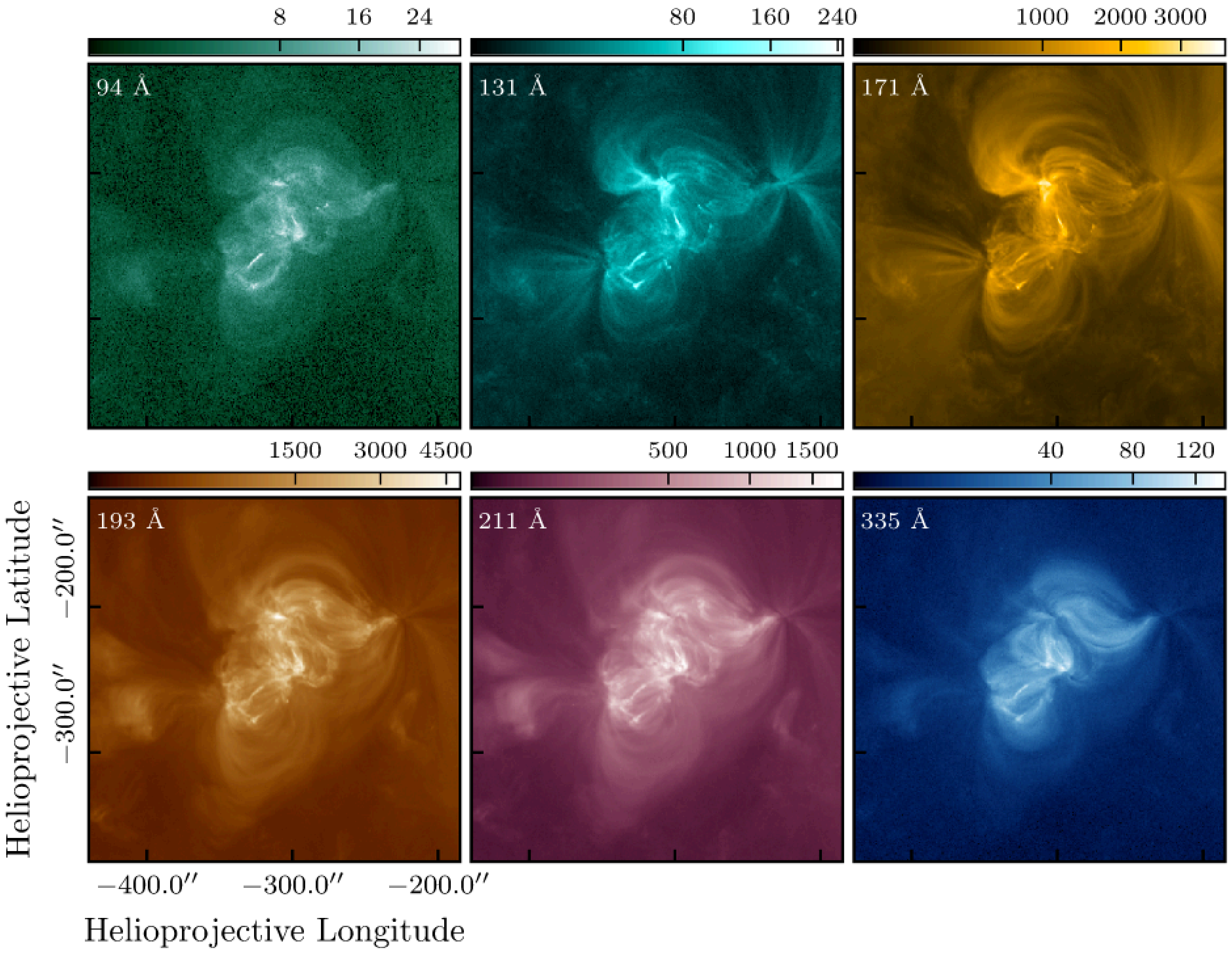


See Barnes et al. (2019)

wtbarnes.github.io/klimchuk-isfm-2024-talk

sunpy powered

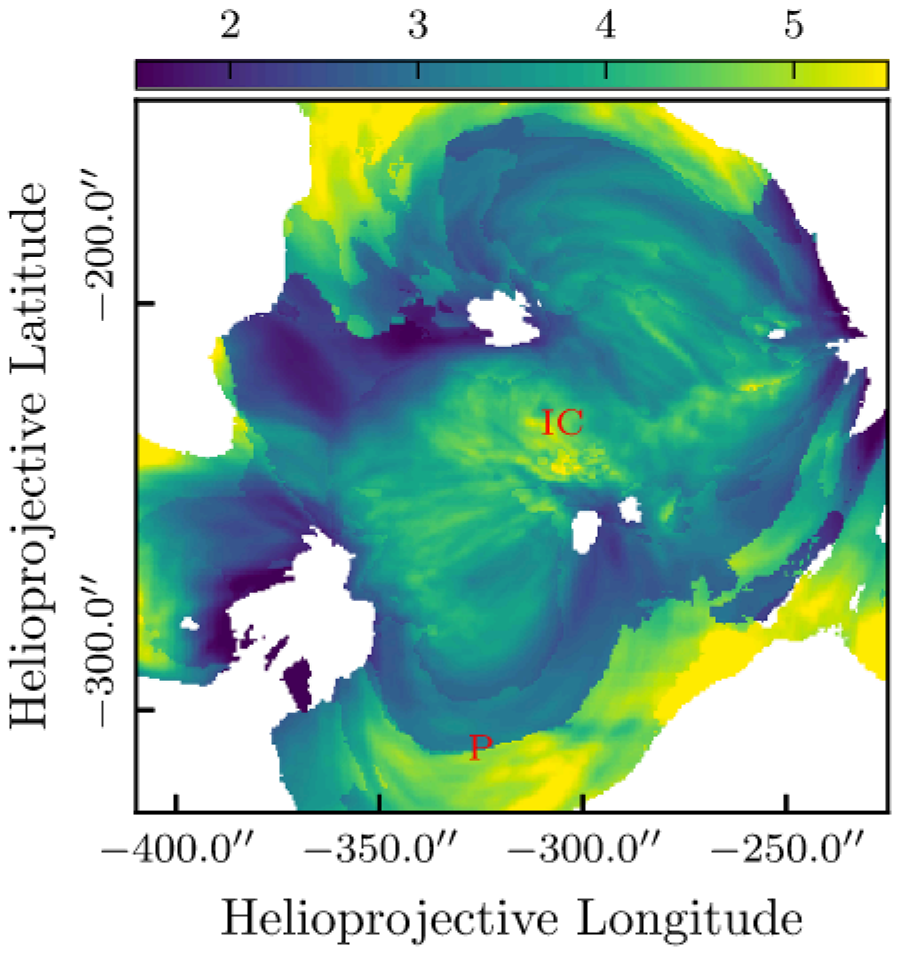
Application: Active Region Heating Diagnostics

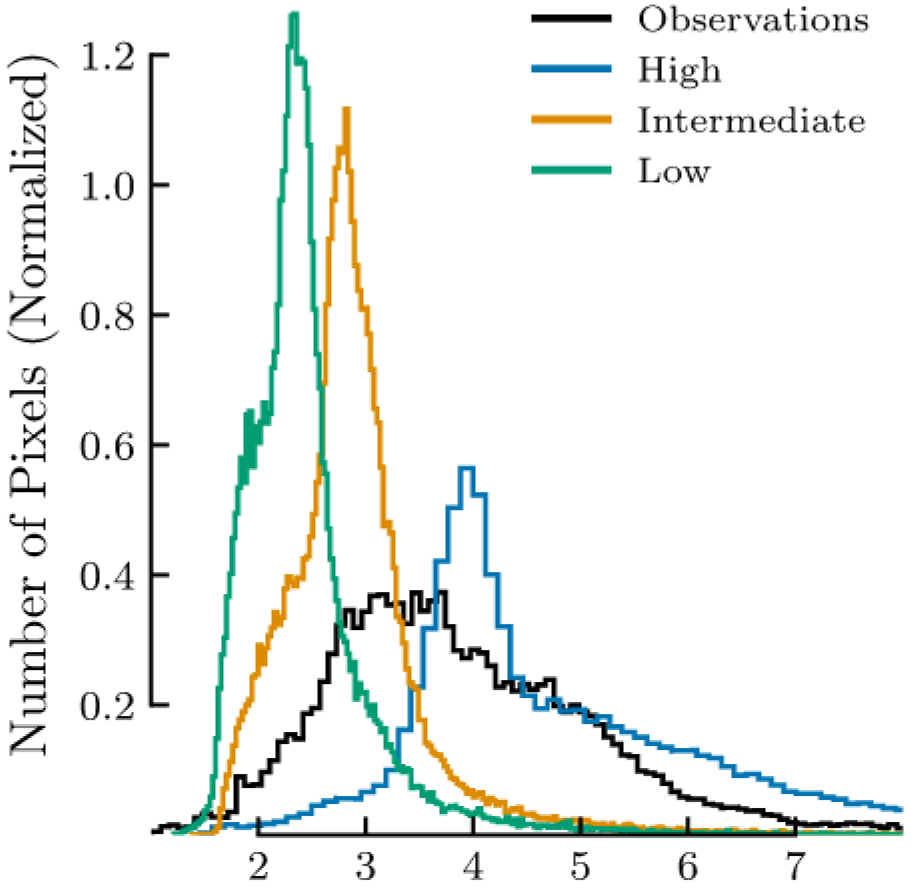


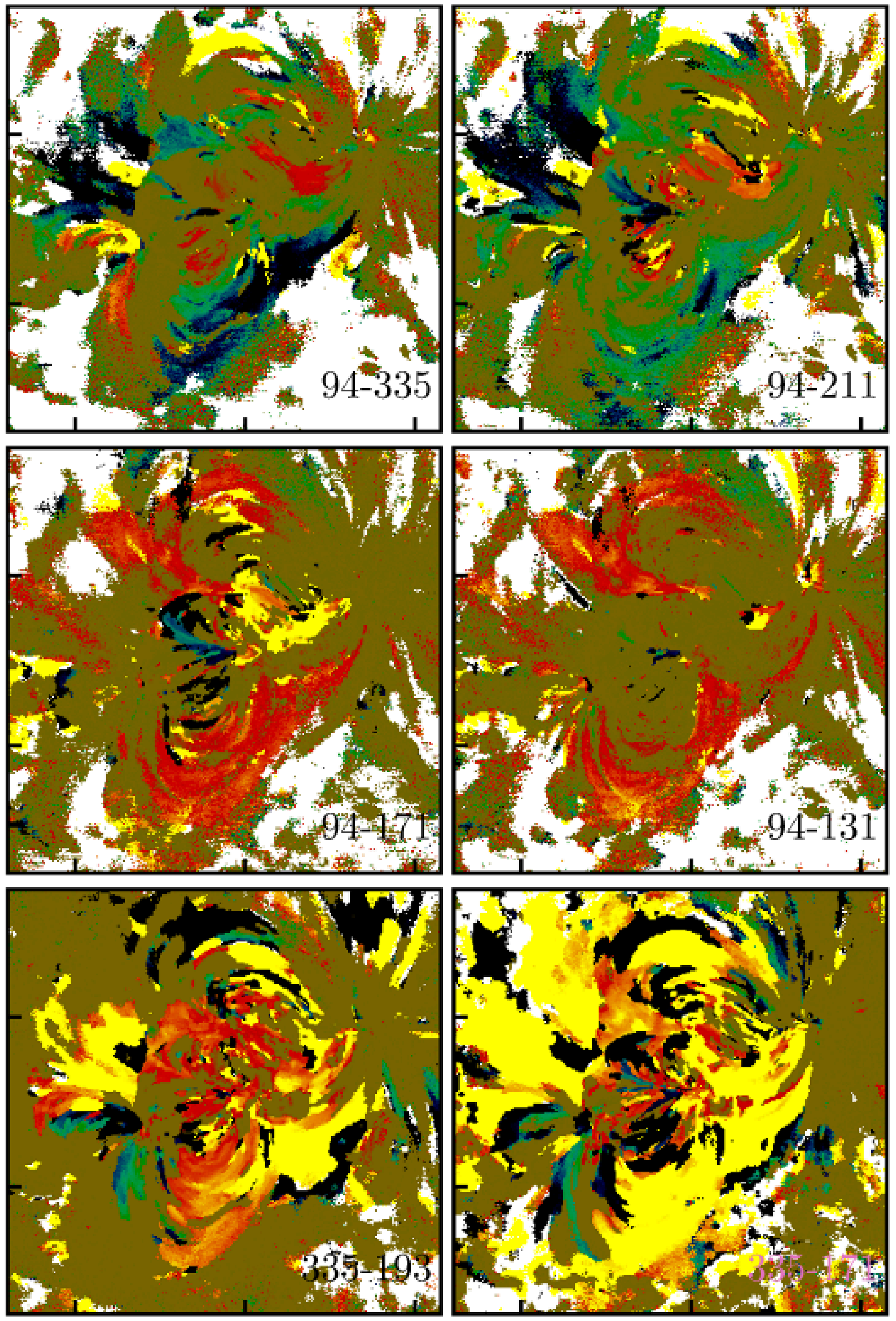
See Barnes et al. (2021)

sunpy powered

Application: Active Region Heating Diagnostics





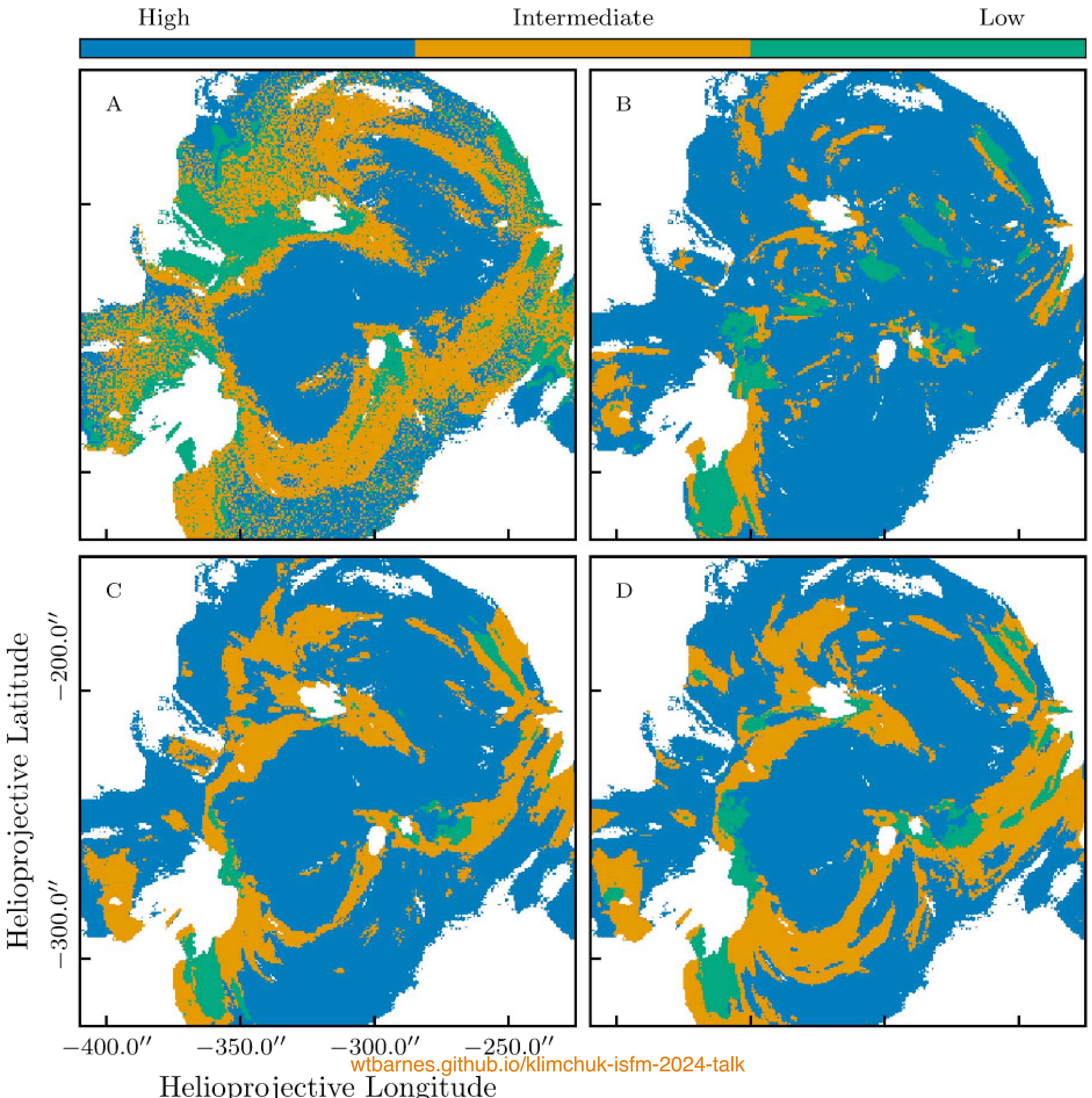


See Barnes et al. (2021)

wtbarnes.github.io/klimchuk-isfm-2024-talk

sunpy powered

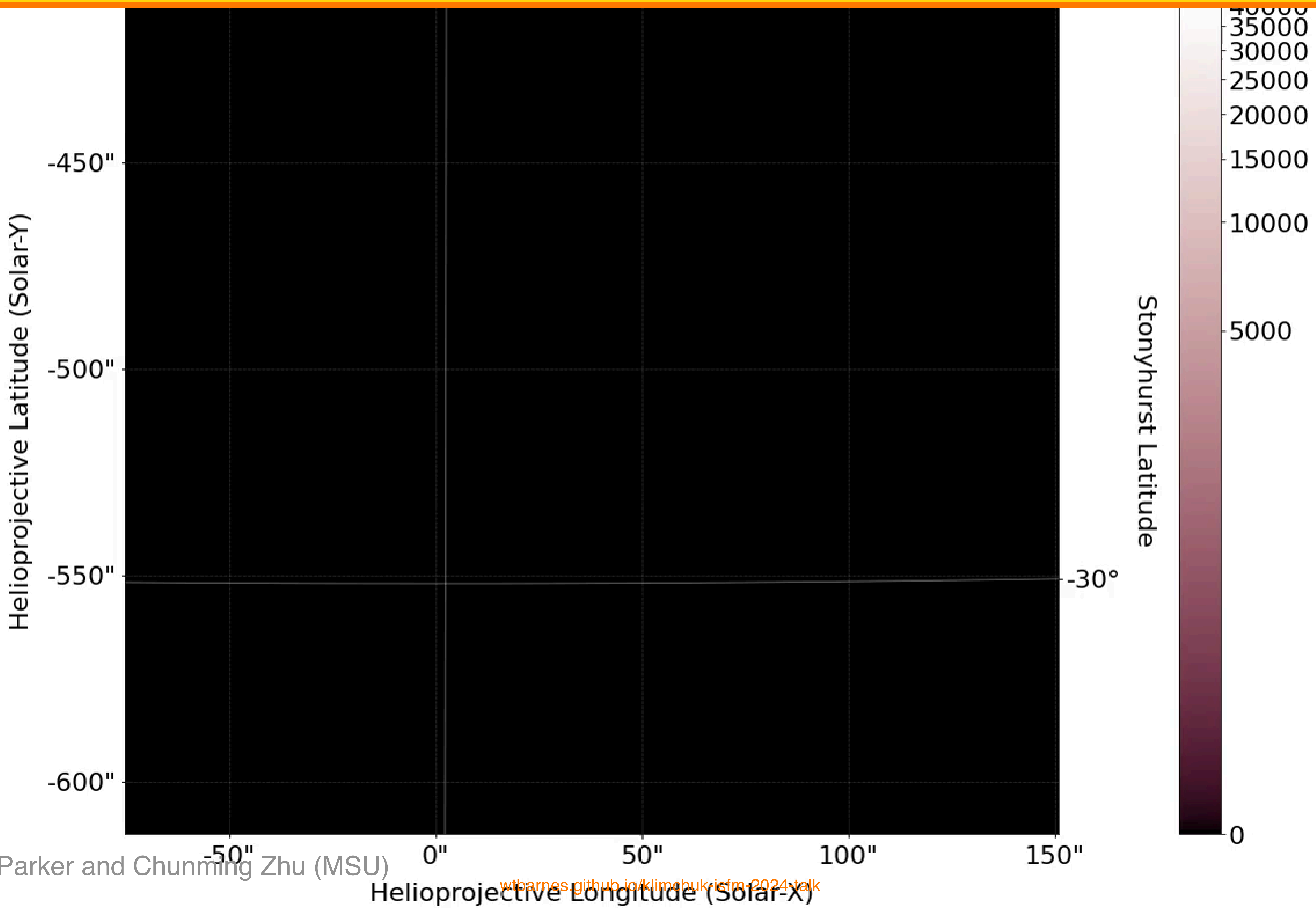
Application: Active Region Heating Diagnostics



See Barnes et al. (2021)

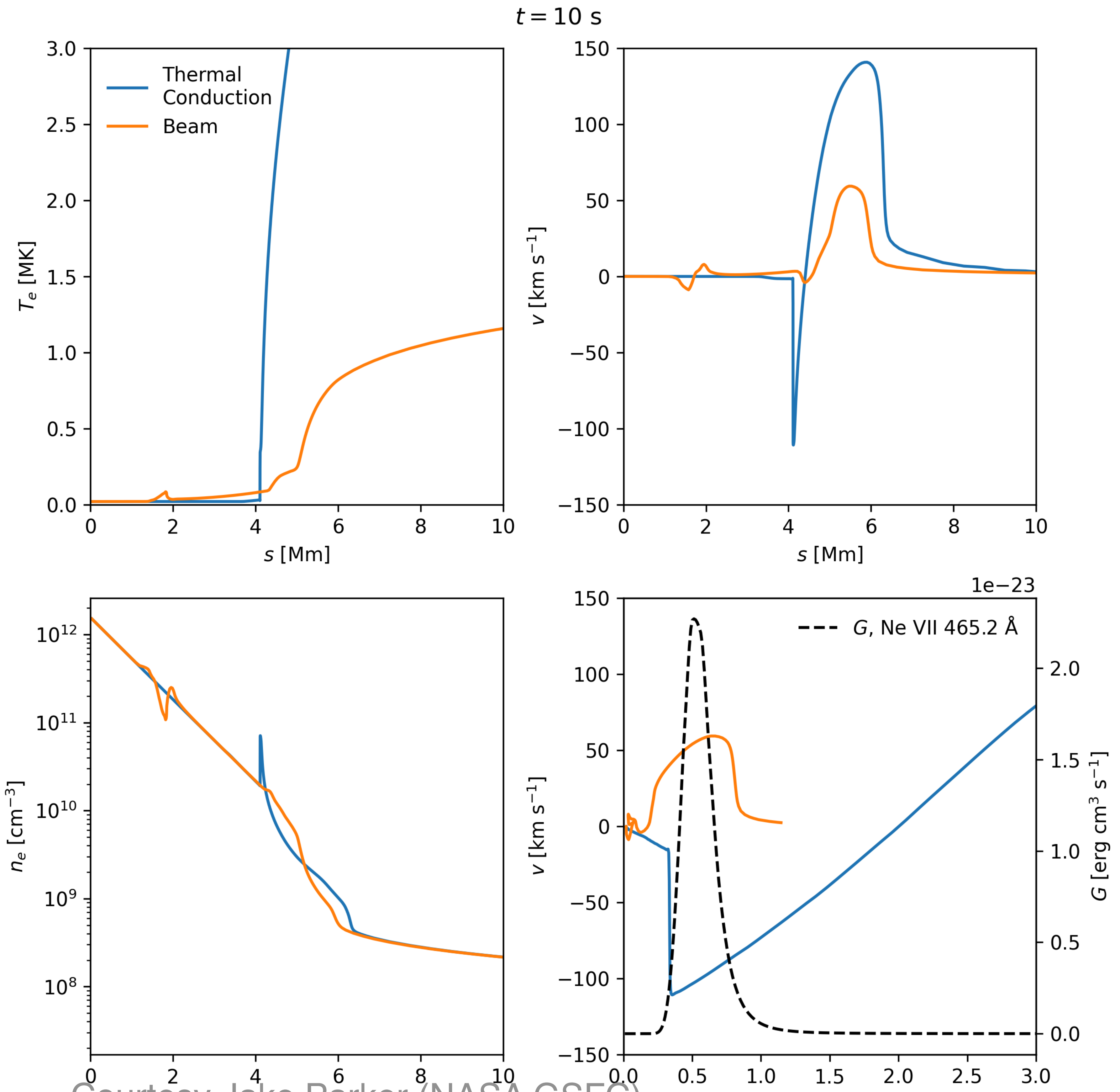
sunpy powered

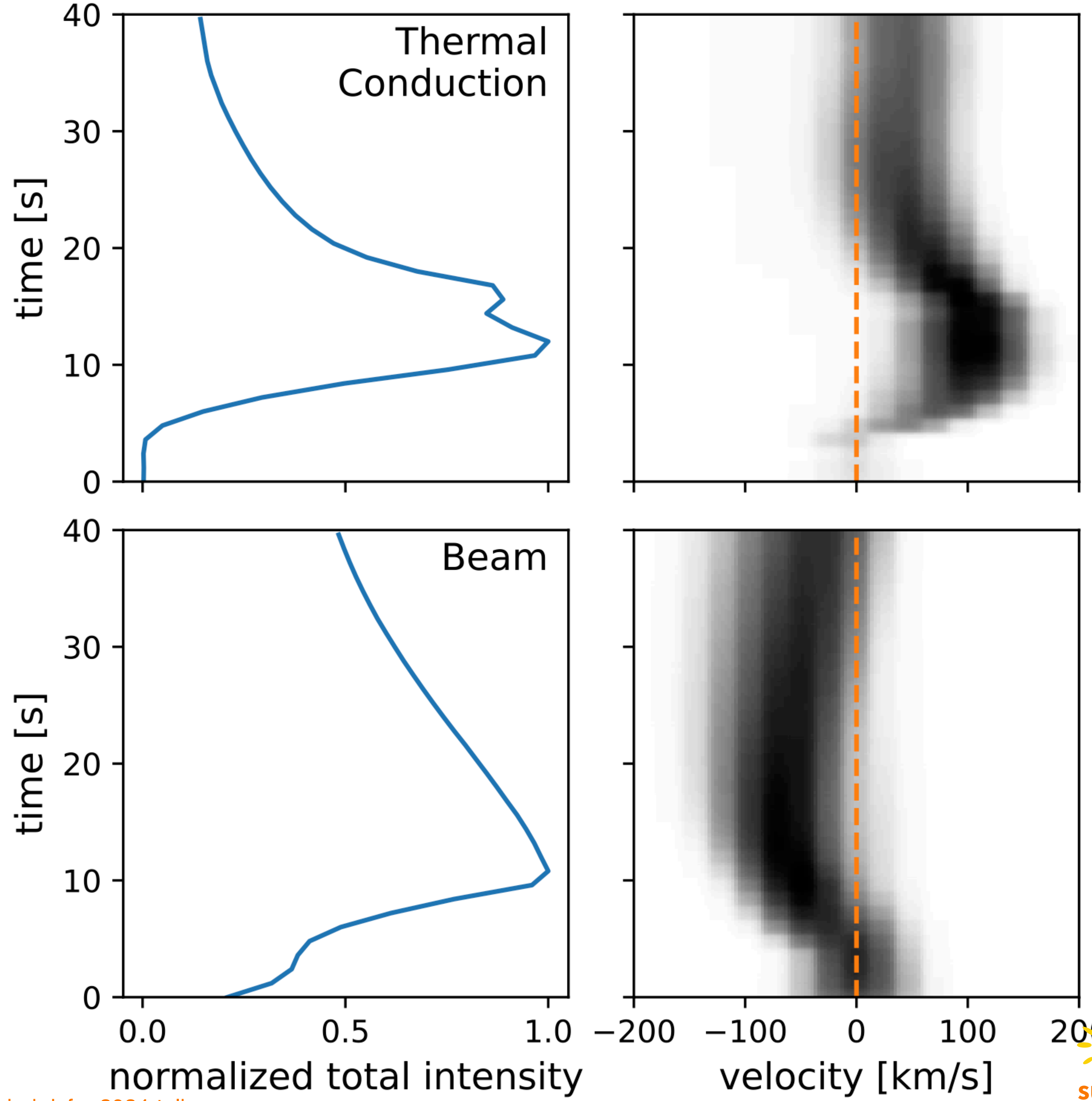
Application: EUV Flare Emission



Courtesy Brock Parker and Chunming Zhu (MSU)

Application: Observing Flows in the TR with ESIS-II

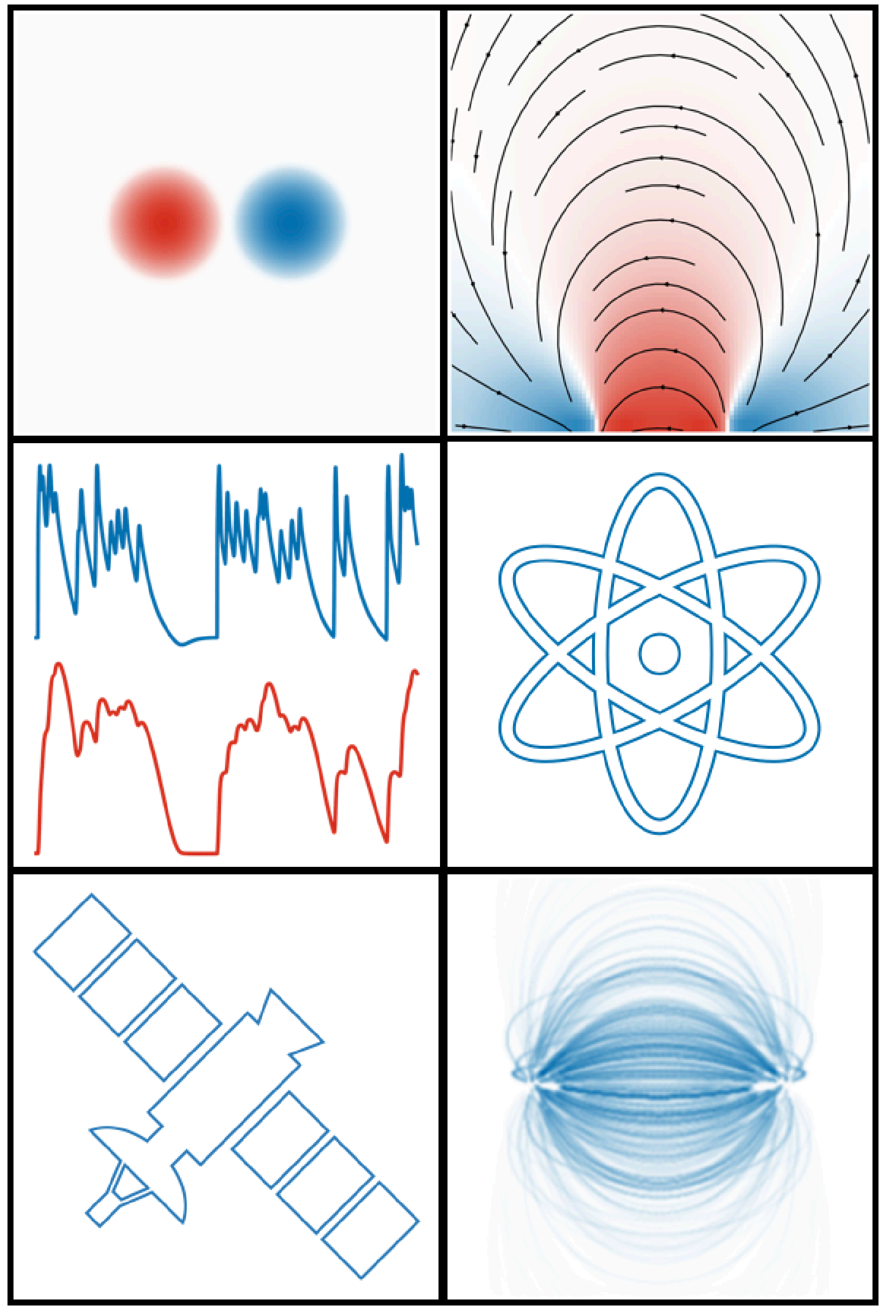




Courtesy Jake Parker (NASA GSFC)

wtbarnes.github.io/klimchuk-isfm-2024-talk

Summary



- synthesizAR–pure *Python* package for modeling time-dependent emission from field-aligned models
- Models *thermal*, *optically-thin* emission confined to *discrete* loop structures
- Prioritizes modularity and flexibility–geometry, loop models, instrument
- Capabilities:
  - Works with *any* field-aligned loop model
  - Incorporate detailed atomic physics in emission modeling
  - Time-dependent, spatially-resolved emission
  - High-resolution, spectrally-resolved diagnostics

References

Barnes, W. T., Bradshaw, S. J., & Viall, N. M. 2019, ApJ, 880, 56

Barnes, W. T., Bradshaw, S. J., & Viall, N. M. 2021, ApJ, 919, 132

Bradshaw, S. J., & Cargill, P. J. 2010, ApJ, 717, 163

Cargill, P. J. 2014, ApJ, 784, 49

Viall, N. M., & Klimchuk, J. A. 2012, ApJ, 753, 35

sunpy powered