Big Data and Open Science

René van Westen

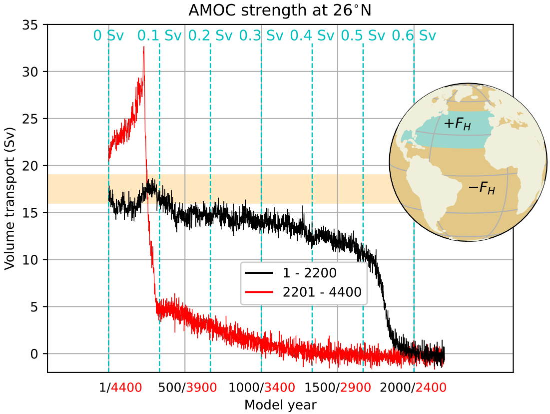
Institute for Marine and Atmospheric Research Utrecht

Utrecht University

20 February 2024

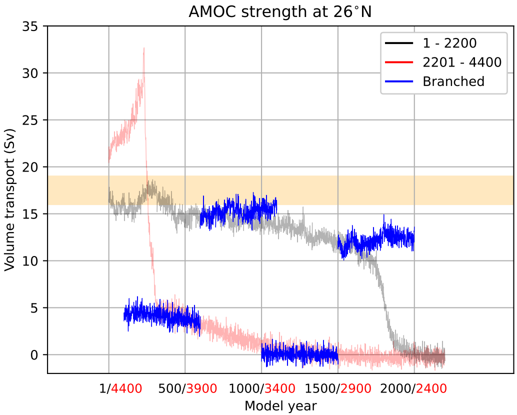
The CESM Hosing Simulation

- Hysteresis experiment
  - 4,400 model years



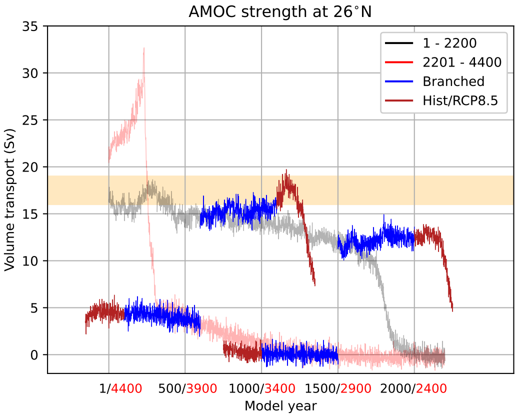
The CESM Hosing Simulation

- Hysteresis experiment
  - 4,400 model years
- Branched simulations
  - 4 $\times$ 600 model years



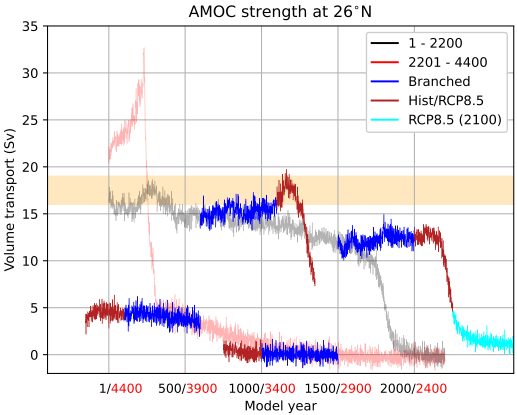
The CESM Hosing Simulation

- Hysteresis experiment
  - 4,400 model years
- Branched simulations
  - 4 × 600 model years
- Historical forcing and climate change
  - 4 × 250 model years



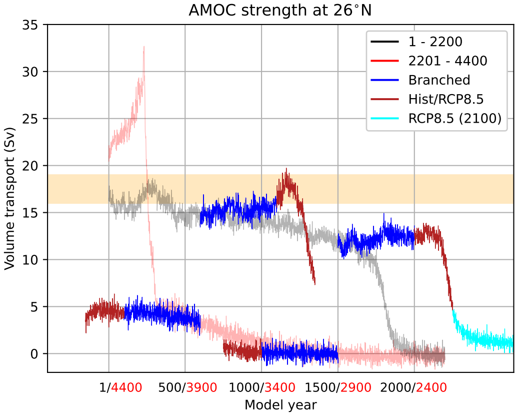
The CESM Hosing Simulation

- Hysteresis experiment
  - 4,400 model years
- Branched simulations
  - 4 × 600 model years
- Historical forcing and climate change
  - 4 × 250 model years
- Constant 2100 RCP8.5 conditions
  - 1 × 400 model years



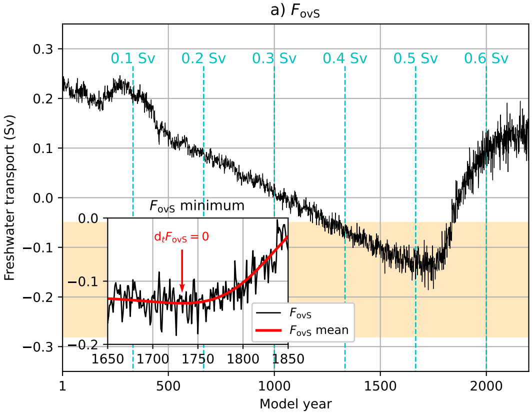
The CESM Hosing Simulation

- Hysteresis experiment
  - 4,400 model years
- Branched simulations
  - 4 × 600 model years
- Historical forcing and climate change
  - 4 × 250 model years
- Constant 2100 RCP8.5 conditions
  - 1 × 400 model years
- 8,200 models years in total
  - 1 monthly ocean file → 1.1 Gb



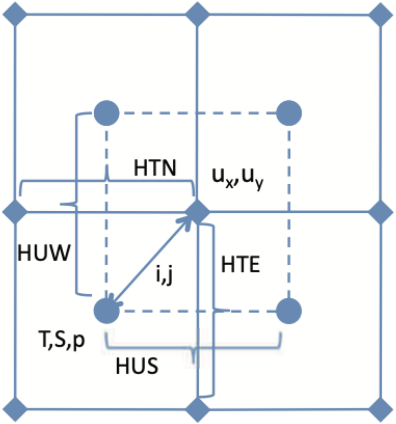
Analysis on the CESM output I

- $$F_{\mathrm{ov}}(y) = -\frac{1}{S_0}\int_{-H}^{0}\left[\int_{x_W}^{x_E} v^* \mathrm{d}x\right][\langle S\rangle - S_0]\,\mathrm{d}z$$

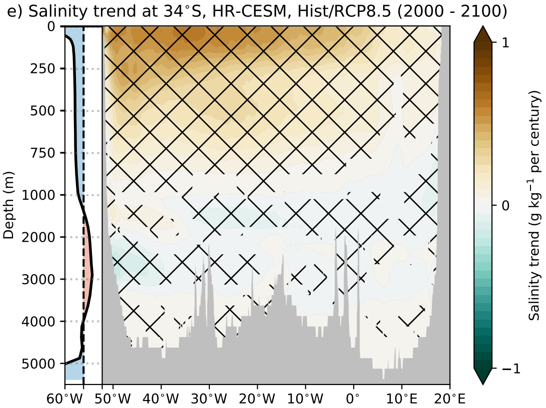


Analysis on the CESM output I

- $$F_{\mathrm{ov}}(y) = -\frac{1}{S_0}\int_{-H}^{0}\left[\int_{x_W}^{x_E} v^* \mathrm{d}x\right]\left[\langle S\rangle - S_0\right]\mathrm{d}z$$

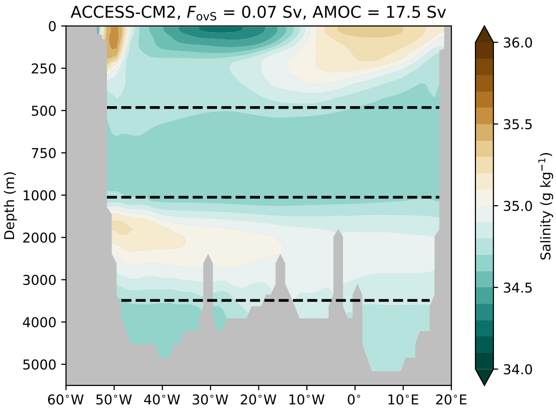


- $$F_{\text{ov}}(y) = -\frac{1}{S_0}\int_{-H}^{0}\left[\int_{x_W}^{x_E} v^* \mathrm{d}x\right]\left[\langle S\rangle - S_0\right]\mathrm{d}z$$
- Statistical tests



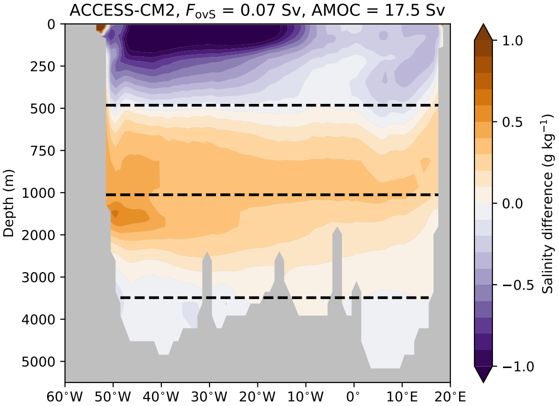
Analysis on the CESM output I

- $$F_{\text{ov}}(y) = -\frac{1}{S_0}\int_{-H}^{0}\left[\int_{x_W}^{x_E} v^* \mathrm{d}x\right][\langle S\rangle - S_0]\,\mathrm{d}z$$
- Statistical tests
- Interpolation onto different grid



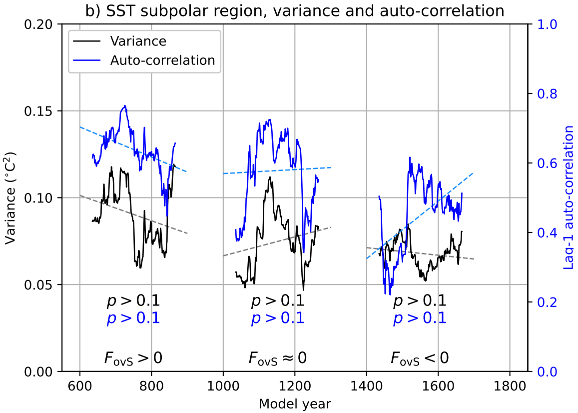
Analysis on the CESM output I

- $$F_{\mathrm{ov}}(y) = -\frac{1}{S_0} \int_{-H}^{0} \left[\int_{x_W}^{x_E} v^* \mathrm{d}x \right] [\langle S \rangle - S_0] \, \mathrm{d}z$$
- Statistical tests
- Interpolation onto different grid



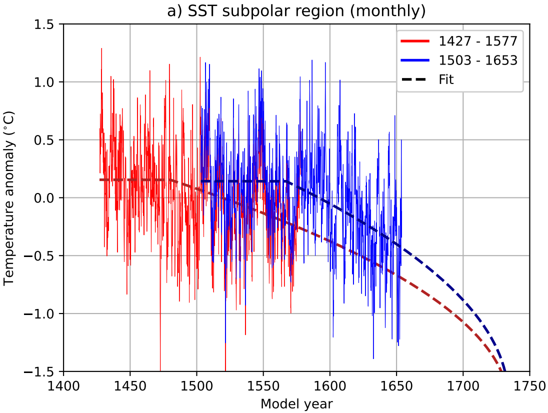
Analysis on the CESM output II

- Boers (2021)
  - Early Warning Indicators



Analysis on the CESM output II

- Boers (2021)
  - Early Warning Indicators
- Ditlevsen & Ditlevsen (2023)
  - Estimating the AMOC tipping point



FAIR Principles

- **F**indability
- **A**ccessibility
- **I**nteroperability
- **R**euse

FAIR Principles

- **F**indability
  - Data are registered with a globally unique identifier
- **A**ccessibility
  - Data are retrievable and permanently stored
- **I**nteroperability
  - The data use a formal and accessible language
- **R**euse
  - The data are richly described with metadata

My (personal) problems with Boers (2021)

Code availability

All Python code used for the analysis is available from the author upon request (boers@pik-potsdam.de) or on GitHub at https://github.com/niklasboers/AMOC_EWS.

My (personal) problems with Boers (2021)

niklasboers Add files via upload		be10717 · 3 years ago 1 Commits
EWS_functions.py	Add files via upload	3 years ago
model_data_comparison.py	Add files via upload	3 years ago
salinity.py	Add files via upload	3 years ago
shiftgrid.py	Add files via upload	3 years ago
ssts.py	Add files via upload	3 years ago

My (personal) problems with Boers (2021)

```python

weights = np.cos(np.radians(lat))
weights = weights / np.sum(weights)
print(weights)
weights = np.tile(weights, (lon.shape[0], 1)).T
sstay = sstay * weights
ssty = ssty * weights


# gl_mean_ay = np.nanmean(sstay.reshape(sstay.shape[0], sstay.shape[1] * sstay.shape[2])[:, gloidx
#
# gl_mean = np.nanmean(ssty.reshape(ssty.shape[0], ssty.shape[1] * ssty.shape[2])[:, gloidx], axis
# nh_mean = np.nanmean(ssty.reshape(ssty.shape[0], ssty.shape[1] * ssty.shape[2])[:, nhidx], axis
#
# amoc1 = np.nanmean(ssty.reshape(ssty.shape[0], ssty.shape[1] * ssty.shape[2])[:, sgi], axis = 1)

gl_mean_ay = np.nansum(sstay.reshape(sstay.shape[0], sstay.shape[1] * sstay.shape[2])[:, gloidx],

gl_mean = np.nansum(ssty.reshape(ssty.shape[0], ssty.shape[1] * ssty.shape[2])[:, gloidx], axis =
nh_mean = np.nansum(ssty.reshape(ssty.shape[0], ssty.shape[1] * ssty.shape[2])[:, nhidx], axis = 1

amoc1 = np.nansum(ssty.reshape(ssty.shape[0], ssty.shape[1] * ssty.shape[2])[:, sgi], axis = 1) /


amoc2 = amoc1 - gl_mean
amoc2 = (amoc2 - np.mean(amoc2))
np.savetxt('data/amoc_idx_niklas.txt', amoc2)

```

My (personal) problems with Boers (2021)

```python
var_sst_trend = np.zeros((la, lo))
ar1_sst_trend = np.zeros((la, lo))
lambda_sst_trend = np.zeros((la, lo))

# count = 0
# for i in range(la):
#     for j in range(lo):
#         if np.sum(np.isnan(ssty[:, i, j])) == 0:
#             ts_temp = ssty[:, i, j] - runmean(ssty[:, i, j], sm_w)
#
#             sst_var_spatial[:, i, j] = runstd(ts_temp, rmw)**2
#             p1, p0 = np.polyfit(time[bound : - bound], sst_var_spatial[:, i, j][bound : - bound]
#             var_sst_trend[i, j] = p1
#
#             sst_ar1_spatial[:, i, j] = runac(ts_temp, rmw)
#             p1, p0 = np.polyfit(time[bound : - bound], sst_ar1_spatial[:, i, j][bound : - bound]
#             ar1_sst_trend[i, j] = p1
#
#             # sst_lambda_spatial[:, i, j] = run_fit_a(ts_temp, rmw)
#             # p1, p0 = np.polyfit(time[bound : - bound], sst_lambda_spatial[:, i, j][bound : - b
#             # lambda_sst_trend[i, j] = p1
#             count += 1
#             print(count)
#
# np.save('data/am_sst_global_runstd.npy', sst_var_spatial)
# np.save('data/am_sst_global_runac.npy', sst_ar1_spatial)
# # np.save('data/am_sst_global_runlambda.npy', sst_lambda_spatial)
```

My (personal) problems with Boers (2021)

```python
def fourrier_surrogates(ts, ns):
    ts_fourier  = np.fft.rfft(ts)
    random_phases = np.exp(np.random.uniform(0, 2 * np.pi, (ns, ts.shape[0] // 2 + 1)) * 1.0j)
    ts_fourier_new = ts_fourier * random_phases
    new_ts = np.real(np.fft.irfft(ts_fourier_new))
    return new_ts

def kendall_tau_test(ts, ns, tau, mode1 = 'fourier', mode2 = 'linear'):
    tlen = ts.shape[0]

    if mode1 == 'fourier':
        tsf = ts - ts.mean()
        nts = fourrier_surrogates(tsf, ns)
    elif mode1 == 'shuffle':
        nts = shuffle_surrogates(ts, ns)
    stat = np.zeros(ns)
    tlen = nts.shape[1]
    if mode2 == 'linear':
        for i in range(ns):
            stat[i] = st.linregress(np.arange(tlen), nts[i])[0]
    elif mode2 == 'kt':
        for i in range(ns):
            stat[i] = st.kendalltau(np.arange(tlen), nts[i])[0]
    p = 1 - st.percentileofscore(stat, tau) / 100.
    return p
```

FAIR Principles score Boers (2021)

- **F**indability: 8/10
- **A**ccessibility: 4/10
- **I**nteroperability: 2/10
- **R**euse: 1/10

FAIR Principles score Boers (2021)

- **F**indability: 8/10
- **A**ccessibility: 4/10
- **I**nteroperability: 2/10
- **R**euse: 1/10

- **Overall score**: 4/10

Some minor problems with Ditlevsen & Ditlevsen (2023)

Code availability

Computer code (Matlab and R) can be found in the following repository: https://doi.org/10.17894/ucph.b8f99b67-d4e6-4a2e-b518-00bddeed323b.

Some minor problems with Ditlevsen & Ditlevsen (2023)

Name	Date	Size
AMOCdata.txt	2023-07-09 14:26:01	102002
AMOCestimation.html	2023-07-09 19:33:42	1387093
AMOCestimation.Rmd	2023-07-09 19:33:42	22614
EstimMatrix_1000repetitions.xlsx	2023-07-09 14:26:01	106797
SimulatedTraces.Rdata	2023-07-09 14:26:01	14111716
Supplementary_Information.pdf	2023-07-09 14:26:01	181285

Some minor problems with Ditlevsen & Ditlevsen (2023)

Estimation of tipping time from AMOC fingerprint

The following code chunk loads the AMOC fingerprint data and estimates parameters. Estimates are reported.

Code

```
##      t0  alpha0     mu0  sigma2     tau       a       m lambda0      tc
## 1924.00    3.06    0.25    0.30  132.52    0.87   -1.51   -2.69 2056.52
```

Estimation of tipping time from AMOC fingerprint

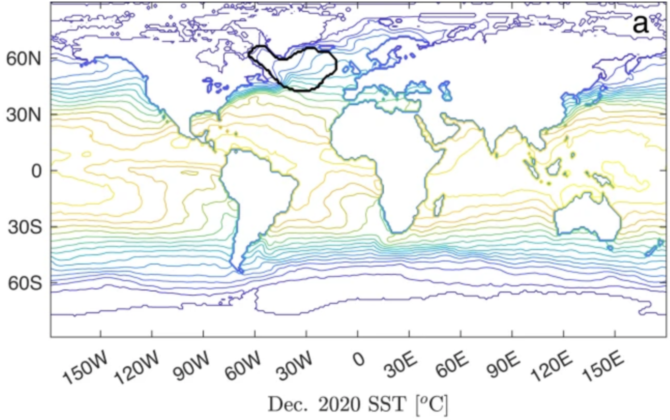
The following code chunk loads the AMOC fingerprint data and estimates parameters. Estimates are reported.

Hide

```
##############################
### Estimation on AMOC data ###
##############################

## Read data
AMOC.data = read.table("AMOCdata.txt", header = TRUE)
## time: calendar time in years
## AMOC0: SST (sea surface temperature) in subpolar gyre, subtracted the monthly mean
## AMOC1: AMOC0 subtracted the global mean SST
## AMOC2: AMOC0 subtracted two times the global mean SST (Arctic amplification)
## GM: global mean SST

## Adding fingerprint with 3 times subtracted global warming for robustness analysis
AMOC.data$AMOC3 = AMOC.data$AMOC0 - 3 * AMOC.data$GM
```

Some minor problems with Ditlevsen & Ditlevsen (2023)

Fig. 1: The Atlantic meridional overturning circulation (AMOC) fingerprint, sea surface temperature (SST) and global mean (GM).



FAIR Principles score Ditlevsen & Ditlevsen (2023)

- **F**indability: 10/10
- **A**ccessibility: 7/10
- **I**nteroperability: 7/10
- **R**euse: 6/10

FAIR Principles score Ditlevsen & Ditlevsen (2023)

- **F**indability: 10/10
- **A**ccessibility: 7/10
- **I**nteroperability: 7/10
- **R**euse: 6/10

- **Overall score**: 7.5/10

Frustrations with van Westen et al. (2024)

Software and model output

The (processed) model output and analysis scripts are provided at: https://doi.org/10.5281/zenodo.10461549. The reanalysis and assimilation products can be accessed through GLORYS12V1 (https://doi.org/10.48670/moi-00021), SODA3.15.2 (http://soda.umd.edu), ORAS5 (https://doi.org/10.24381/cds.67e8eeb7), ORA-20C (https://icdc.cen.uni-hamburg.de/thredds/catalog/ftpthredds/EASYInit/ora20c/opa0/catalog.html), and ECCO-V4r4 (https://ecco-group.org/products-ECCO-V4r4.htm).

Frustrations with van Westen et al. (2024)

RenevanWesten Update README.md 7c14f52 · last month 15 Commits

Data	Add files via upload	5 months ago
Program	Add files via upload	3 months ago
README.md	Update README.md	last month

README

SA-AMOC-Collapse

Physics-Based Early Warning Signal shows AMOC is on Tipping Course, Science Advances (January 2024)

René M. van Westen, Michael Kliphuis and Henk A. Dijkstra

These directories contain Python (v3) scripts for plotting/analysing model output.

Python scripts can be found in the directory 'Program'. Model output can be found in the directory 'Data'.

The processed model output are stored as NETCDF files and using the relevant scripts one can regenerate all the figures. We provided parts of the original model output (native grid) and is only converted to yearly-averaged data (due to storage limitations). Some scripts (e.g., FOV_index_34S.py and AMOC_transport.py) use the original model output, the generated time series are already available in the relevant directories.

Frustrations with van Westen et al. (2024)

SA-AMOC-Collapse / Program / CESM / **Ocean** /

Add file ···

RenevanWesten Add files via upload 748d861 · 3 months ago History

Name	Last commit message	Last commit date
..		
AMOC_structure_plot.py	Add files via upload	5 months ago
AMOC_tipping_point.py	Add files via upload	3 months ago
AMOC_transport.py	Add files via upload	5 months ago
AMOC_transport_plot.py	Add files via upload	3 months ago
DSL_plot.py	Add files via upload	5 months ago

Frustrations with van Westen et al. (2024)

```python
def ReadinData(filename, depth_min_index, depth_max_index):

        fh = netcdf.Dataset(filename, 'r')

        #First get the u-grid
        year     = fh.variables['year'][:]                                                  #Model year
        lon      = fh.variables['ULONG'][:]                                                 #Longitude
        lat      = fh.variables['ULAT'][:]                                              #Latitude
        depth    = fh.variables['z_t'][depth_min_index:depth_max_index]      #Depth (m)
        layer    = fh.variables['dz'][depth_min_index:depth_max_index]              #Layer thickness (m)
        grid_x   = fh.variables['DXU'][:]                                         #Zonal grid cell length (m)
        v_vel    = fh.variables['VVEL'][:, depth_min_index:depth_max_index, 0] #Meridional velocity (m/s)

        fh.close()

        return year, lon, lat, depth, layer, grid_x, v_vel

#-----------------------------------------------------------------------------------------
#--------------------------------MAIN SCRIPT STARTS HERE----------------------------------
#-----------------------------------------------------------------------------------------

depth_min       = 0
depth_max       = 1000

#-----------------------------------------------------------------------------------------

files = glob.glob(directory+'Data/AMOC_section_26N/CESM_year_*.nc')
files.sort()
```

Frustrations with van Westen et al. (2024)

```
for depth_i in range(len(depth)):
	#Determine the total length of the section, based on non-masked elements
	layer_field[depth_i]	= layer[depth_i]
	layer_field[depth_i]	= ma.masked_array(layer_field[depth_i], mask = v_vel_all[0, depth_i].mask)

	#Determine where the layer needs to be adjusted, partial depth cells
	depth_diff	= np.sum(layer_field, axis = 0) - depth_u

	if depth_i == len(depth) - 1:
		#Last layer, get the depth difference with respect to top and depth max boundary
		depth_diff	= layer_field[depth_i] - (depth_max -  depth_top[depth_i])

	#If the depth difference is negative (i.e. bottom is not reached), set to zero
	depth_diff	= ma.masked_where(depth_diff < 0, depth_diff)
	depth_diff	= depth_diff.filled(fill_value = 0.0)

	#Subtract the difference of the current layer with the difference
	layer_field[depth_i]	= layer_field[depth_i] - depth_diff
```

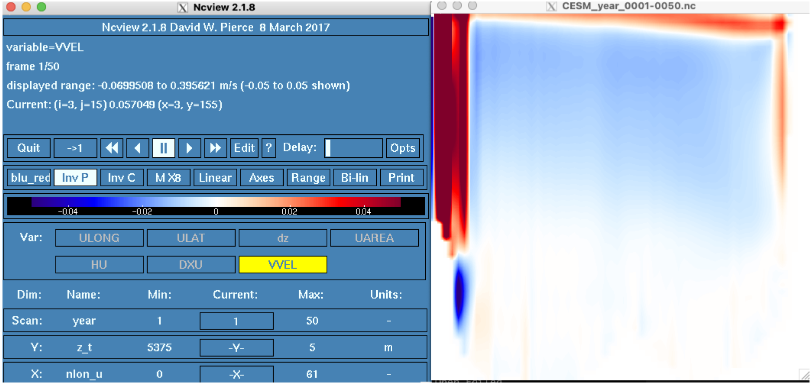
Frustrations with van Westen et al. (2024)

SA-AMOC-Collapse / Data / CESM / Data / AMOC_section_26N /

RenevanWesten Add files via upload

Name	Last commit message
..	
CESM_year_0001-0050.nc	Add files via upload
CESM_year_0051-0100.nc	Add files via upload
CESM_year_0101-0150.nc	Add files via upload
CESM_year_0151-0200.nc	Add files via upload
CESM_year_0201-0250.nc	Add files via upload

Frustrations with van Westen et al. (2024)



Frustrations with van Westen et al. (2024)

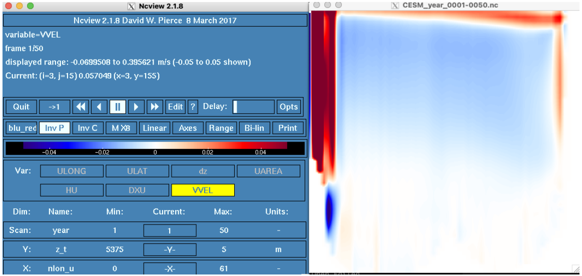
```python

def Kendall_tau_test(data, num_surr, time_data = False):
        """Conducts the Kendall-Fourier test (adapted from Boers 2021)"""

        #Remove masked elements (if present)
        mask_index  = np.where(data.mask == False)[0]
        data        = data[mask_index]

        if type(time_data) == type(False):
                #No time array is provided, use dummy variable
                trend, base = stats.linregress(np.arange(len(data)),data)[0], stats.linregress(np.arange(len(data)),data)[1]
        else:
                #Time array is provided, use these to fit the trends
                time_data       = time_data[mask_index]
                trend, base     = stats.linregress(time_data,data)[0], stats.linregress(time_data,data)[1]

        #Make time mean zero and create surrogate data
        data_0      = data - np.mean(data)
        data_surr   = Fourrier_surrogates(data_0, num_surr)

        #Determine the linear regression of the surrogate time series
        stat_surr   = np.zeros(num_surr)

        for surr_i in range(num_surr):
                #Determine the linear regression
                stat_surr[surr_i] = stats.linregress(np.arange(len(data)), data_surr[surr_i])[0]

        #Determine the significant level w.r.t. trend
        p = 1 - stats.percentileofscore(stat_surr, trend) / 100.

        return p, trend, base
```

My personal FAIR protocol

- Structure your scripts and data
  - (Sub)folders are great

My personal FAIR protocol

- Structure your scripts and data
  - (Sub)folders are great
- Upload semi-raw model output
  - Helps to understand data processing



My personal FAIR protocol

- Structure your scripts and data
  - (Sub)folders are great
- Upload semi-raw model output
  - Helps to understand data processing
- Remove personal comments

```
#-----------------------------------------------------------------------------------------
time_year               = ma.masked_all(int(len(time)/12))
transport_all           = ma.masked_all(len(time_year))

fh = netcdf.Dataset(directory+'Ocean/AMOC_transport_depth_'+str(depth_min)+'-'+str(depth_max)+'m.nc', 'r')

time_AMOC               = fh.variables['time'][:]
transport_AMOC          = fh.variables['Transport'][:]  #AMOC strength (Sv)

fh.close()

time_year[:len(time_AMOC)]      = time_AMOC
transport_all[:len(time_AMOC)]  = transport_AMOC

#for year_i in range(int(np.min(time)), int(np.min(time))+len(time_year)):
for year_i in range(int(np.max(time_AMOC)), int(np.min(time))+len(time_year)):
```

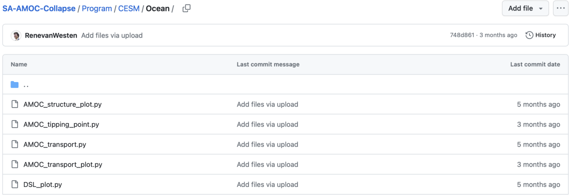
My personal FAIR protocol

- Structure your scripts and data
  - (Sub)folders are great
- Upload semi-raw model output
  - Helps to understand data processing
- Remove personal comments
- Add comments!
  - Use names for iterable objects

```python
for depth_i in range(len(depth)):
	#Determine the total length of the section, based on non-masked elements
	layer_field[depth_i]	= layer[depth_i]
	layer_field[depth_i]	= ma.masked_array(layer_field[depth_i], mask = v_vel_all[0, depth_i].mask)

	#Determine where the layer needs to be adjusted, partial depth cells
	depth_diff	= np.sum(layer_field, axis = 0) - depth_u

	if depth_i == len(depth) - 1:
		#Last layer, get the depth difference with respect to top and depth max boundary
		depth_diff	= layer_field[depth_i] - (depth_max -  depth_top[depth_i])

	#If the depth difference is negative (i.e. bottom is not reached), set to zero
	depth_diff	= ma.masked_where(depth_diff < 0, depth_diff)
	depth_diff	= depth_diff.filled(fill_value = 0.0)

	#Subtract the difference of the current layer with the difference
	layer_field[depth_i]	= layer_field[depth_i] - depth_diff
```

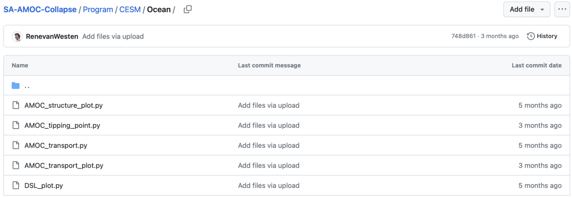
My personal FAIR protocol

- Structure your scripts and data
  - (Sub)folders are great
- Upload semi-raw model output
  - Helps to understand data processing
- Remove personal comments
- Add comments!
  - Use names for iterable objects
- Upload all data and plotting scripts
  - For large files, this can be a problem



My personal FAIR protocol

- Structure your scripts and data
  - (Sub)folders are great
- Upload semi-raw model output
  - Helps to understand data processing
- Remove personal comments
- Add comments!
  - Use names for iterable objects
- Upload all data and plotting scripts
  - For large files, this can be a problem
- Run and check your scripts
  - One work day for each publication


SA-AMOC-Collapse / Program / CESM / Ocean /

Add file

RenevanWesten Add files via upload 748d861 · 3 months ago History

Name	Last commit message	Last commit date
..		
AMOC_structure_plot.py	Add files via upload	5 months ago
AMOC_tipping_point.py	Add files via upload	3 months ago
AMOC_transport.py	Add files via upload	5 months ago
AMOC_transport_plot.py	Add files via upload	3 months ago
DSL_plot.py	Add files via upload	5 months ago