RLadies webinar: R for marine ecologists: wrangling Earth System Model outputss

Jessica Bolin & Nerea Ochoa Lezama

2026-02-18

Table of contents

Preface

Workshop date: February 18th, 5:30-7pm

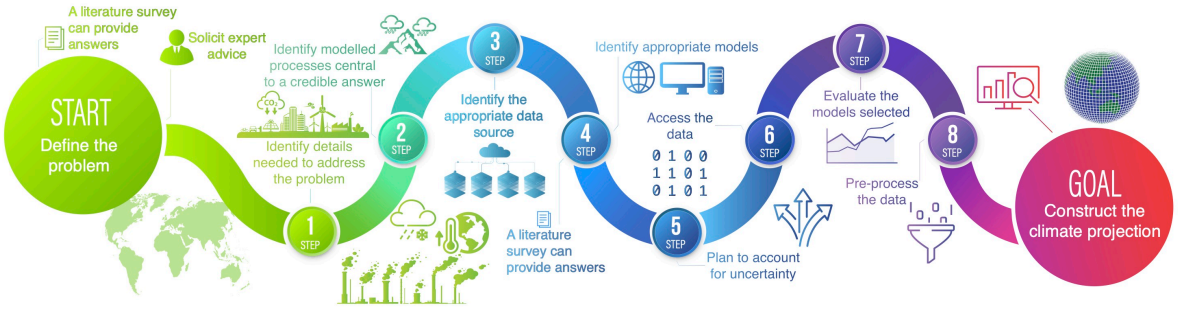
Instructors: Jessica Bolin, Nerea Lezama Ochoa

> **!** Important
>
> This is a condensed version of the full-day workshop. To access the full workshop notes and materials, click here.

`R` is undoubtedly the programming language of choice for many marine ecologists. However, navigating the complexities of downloading, processing and wrangling gridded oceanographic data within `R` can represent a steep learning curve. As climate-related projects continue to receive funding and the impacts of climate variability and change on our oceans become increasingly evident, **the ability to efficiently interact with, and use, Earth System Model (ESM) outputs for our research is more critical than ever**.

The aim of this workshop is to train researchers in skills needed to be proficient in creating projections of physical variables from ESM outputs, with the goal of using them for ecological applications. **Specifically, we'll be creating projections of sea-surface temperature across two time periods out to 2100, across two climate scenarios, for the California Current region.**

Our workshop is designed around Step 8 (pre-process the data) in the workflow of best practices for using ESMs for marine ecologists, as defined in Figure 1 within (Schoeman et al. 2023).



Instructors

Today's instructors are **Jessica Bolin** and **Nerea Lezama-Ochoa**

Jessie is a Postdoc in Provost Lab, working on modeling and mapping climate refugia for red abalone across California. Jessie has a background in quantitative marine science and obtained her PhD from the University of the Sunshine Coast in 2024, and has broad research interests spanning climate adaptation, fisheries forecasting, marine disease ecology and sustainable seafood. Jessie has been using R since 2016, and has worked with ESMs since 2021. `jabbolin@ucdavis.edu`

Nerea is an Associate Project Scientist at the University of California Santa Cruz and works collaboratively with the NOAA's Southwest Fisheries Science Center. Nerea has previously worked on the conservation and management of species taken incidentally in tropical tuna purse seine fisheries (bycatch), and now works with climate projections for highly migratory species and marine spatial planning. Nerea's main research interests are ecological modeling, spatial ecology, oceanography and ecosystem-based management approaches for the conservation of marine top predators.

If you have questions about the climate projections themselves, feel free to contact Mer Pozo Buil (mpozobui@ucsc.edu) who is Associate Project Scientist at the University of California Santa Cruz, who works closely with ESMs, specifically in the California Current region.

Workshop materials

The workshop data are ~`2 GB`, so ensure you have enough storage space on your machine before downloading.

If you're a Github user, you can fork and clone the workshop repository here. Then, you will then need to download the workshop data here, which needs to be copied into the root directory. We've added the `/__data` folder to the `.gitignore` file within the root directory, since we'll be working with large files (and Github breaks when pushing/pulling large files).

We have inserted the expected time duration for running processor-intensive chunks of code in **blockquotes** throughout the eBook. This is based on Jessie's machine (MacBook Pro 2023 M3 Max 64 GB Memory). Your machine may be faster or slower, depending on its specifications.

Expected outcomes

By the end of the course, participants should be able to:

- Download, inspect, and wrangle netCDF file formats within R
- Use `CDO` to speed up netCDF file manipulation
- Download ESM output from ESGF and understand terminology re. variables/naming conventions
- Remap and bias-correct ESM outputs from the CMIP6 suite across multiple climate scenarios and time periods
- Make publication-quality maps

Acknowledgements

We acknowledge funding from the UC Davis Coastal and Marine Science Institute's internal grant scheme (2024-2025), and thank Bodega Marine Laboratory for providing space and equipment to run the pilot workshop in June 2025. We also thank Mer Pozo Buil, Mikaela Provost and Mary Fisher for co-organizing the pilot workshop.

We are very thankful for Barb Muhling (NOAA SWFSC) and Alice Pidd (UniSC) who reviewed workshop materials. We also thank Isaac Brito-Morales for developing the following `R` workshop materials here and here on analysing netCDFs and climate data, which our workshop took inspiration from. We also thank Dave Schoeman who developed and shared the first iteration of some code in this workshop, and who introduced Jessie to the world of climate models in the first place!

Acronyms

We will be using **a lot** of acronyms during today's workshop..!

- `ACCESS-CM2` = Australian Community Climate and Earth System Simulator Coupled Model Version 2
- `CDO` = Climate Data Operators
- CMIP6 = Coupled Model Intercomparison Project Version 6
- ESGF = Earth System Grid Federation
- ESM = Earth System Model
- GCM = Global Coupled Model
- IPCC = Intergovernmental Panel on Climate Change
- `IPSL-CM6A-LR` = Institut Pierre-Simon Laplace Climate Model for Phase 6 of CMIP - Low Resolution
- netCDF = Network Common Data Form (file format = `.nc`)
- OISST = Optimum Interpolation Sea Surface Temperature
- `omon` = monthly time step
- `wget` = World Wide Web Get
- SSPs = Shared socio-economic pathways
- SST = Sea-surface temperature
- `tos` = sea-surface temperature (temperature ocean surface)

Prerequisites

Prerequisites

Our course is not intended for absolute beginners in R. While participants don't need to be highly proficient, a basic understanding of R and programming syntax is required. For example, participants should know how to create an R project and open a script, read in data using commands, make basic plots using `base` R, and write for loops. A basic understanding of the shell/command line is advantageous, but not required. Further, previous experience using the `terra` package and working with `netCDF` files is a plus, but again, not required.

In addition to having R and RStudio already installed, you will need to install (i) `CDO`, (ii) `NCO`, (iii) `wget`, (iv) `Panoply` and (v) a suite of R packages.

> **!** Important
>
> We **strongly** recommend using a macOS laptop for the workshop. We've found that Windows users can run into issues with installing and using `CDO`, which is required for the workshop (see below).
> To see if you may have software issues during our workshop, install the prerequisites below, download this folder here, and run the `runme.R` script. If you don't run into any errors, great! If you do, you will need to do some troubleshooting before the workshop.

CDO

You will need **CDO** (Climate Data Operators) installed prior to the workshop. CDO comprises a suite of over 600 operators for standard (and speedy) processing of climate and forecast model outputs. More information on CDO found here.

MacOS

The simplest way to install CDO is via Homebrew. If you haven't already installed Homebrew, do so here. Then, open terminal and run the following code:

```
brew install cdo
```

More information available on the CDO MacOS website.

Windows

CDO is meant for use on POSIX-compatible operating systems (e.g., like Linux and MacOS), so downloading on Windows requires some extra steps.

Recent versions of Windows (>=10) includes an Ubuntu embedded Linux, offering the opportunity to install CDO via Ubuntu's native package manager. First, install the Ubuntu app from the Microsoft Store application. Then open the Ubuntu terminal and type:

```
sudo apt-get upgrade
sudo apt-get install cdo #write your password, if prompted
```

More information available on the CDO Windows website.

NCO

Similar to CDO, we will install NCO (netCDF operators), which are a suite of operators that take netCDF file formats and facilitate file manipulation.

Mac

```
brew install nco
```

Windows

Installing `NCO` on windows can be tricky. Here is a discussion forum.

`wget`

`wget` is a bash command used for downloading files from the internet. We will use this to download our ESMs via `wget` scripts from the ESGF website.

macOS

Run the following in terminal:

```
brew install wget
```

Windows

Check out the link here. You may need to do some extra fiddling - see here or check out Youtube for further help.

Panoply

Panoply is a data viewer for netCDF, HDF and GRIB data arrays, administered by NASA Goddard Space Flight Center. **Panoply requires that your computer has a compatible Java 11 (or later version) JRE or JDK installed.**

Download Panoply here.

R Packages

Ensure you have the following R packages installed, listed below.

```
#Install pacman
if (!require("pacman")) install.packages("pacman")

# Install packages, if not already installed
pacman::p_load(tidyverse, # Working with 'tidy' data
               furrr, parallelly, purrr, # Run functions in parallel
               tictoc, beepr, # Understand code execution time
```

```
                ncdf4, raster, terra, # For working with netCDFs
                tmap, # Map visualization
                plotrix, # Taylor Diagrams
                RCurl, xml2, rvest, # Downloading files
                zoo)
```

Changing your path

Lastly, once you have downloaded the repository, you will need to open `__scripts/helpers.R`, and change your `pth` object to suit your machine (see Line 7). I've set mine as `"/Users/admin/Documents/GitHub/esmRworkshop_rladies26"`; change yours to reflect where this folder is located on your machine.

❗ Below are the dependencies and package versions the instructors will use for the workshop.

1 What is an ESM?

They use mathematical laws and equations

> **Model jargon**
>
> You may have heard the terms climate model, Earth System model (ESM), general circulation model (GCM) and coupled atmosphere-ocean general circulation models (AOGCMs). These are not all interchangeable! Historically, a 'climate model' referred to AOGCMs or GCMs, which do not simulate biogeochemical processes. ESMs, on the other hand, evolved from GCMs to simulate ocean chemistry and marine plankton due to their importance in carbon cycling. As such, ESMs simulate the 3D evolution of the ocean and its biogeochemical processes.

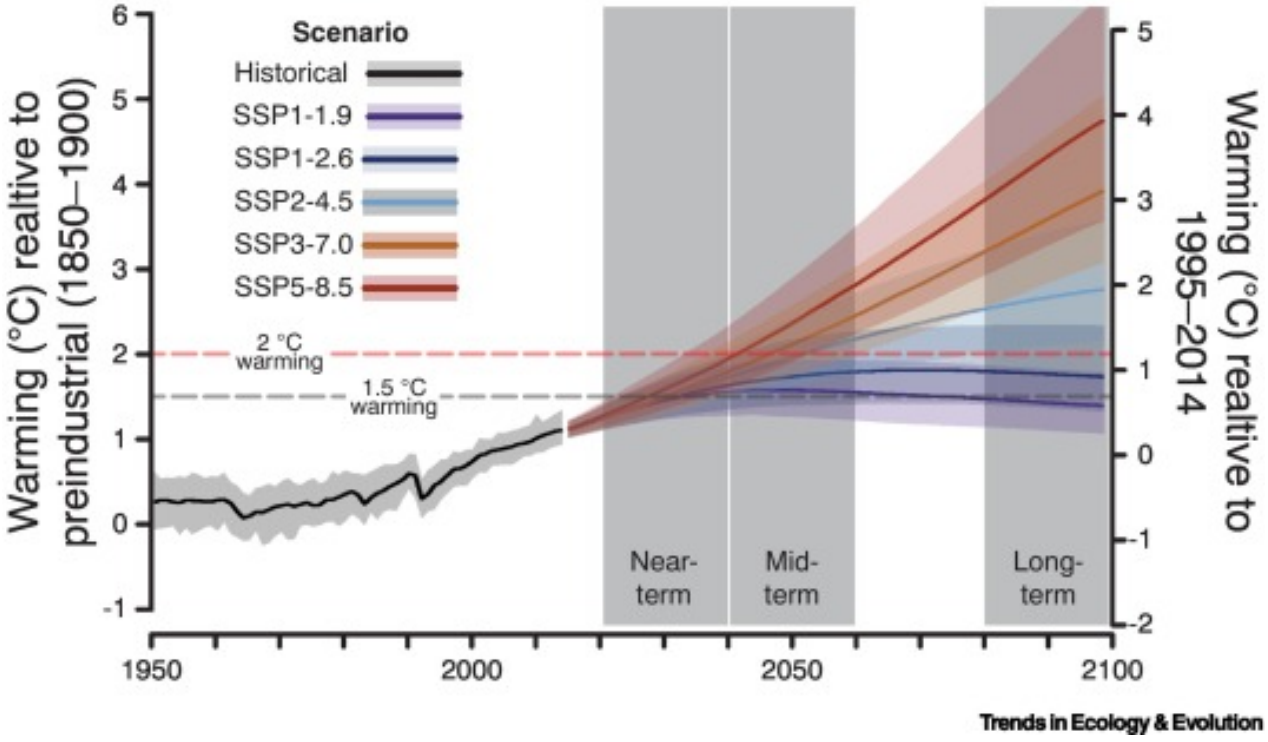
~100 distinct climate models

49 different modeling groups

1.1 Experiments

here

1.2 Climate scenarios (SSPs)



1.3 ESMs we're using today!

2 Download ESM output from ESGF

2.1 Downloading CMIP6 Earth System Models

2.2 Download `wget` shell scripts

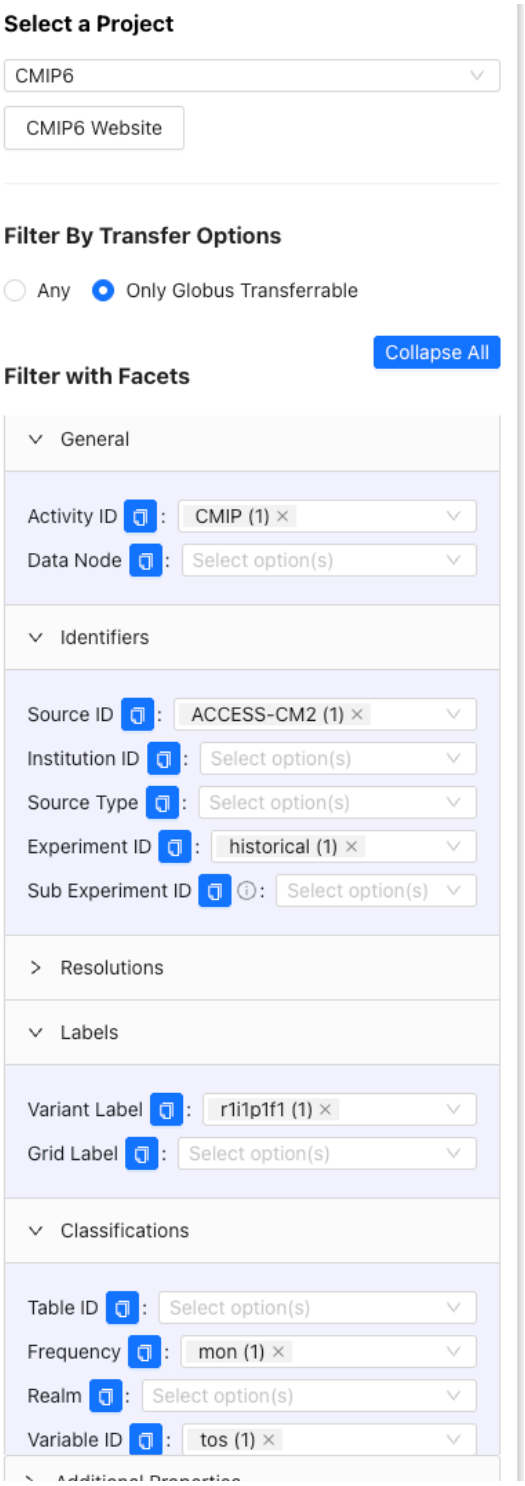
2.2.1 Navigate to the Metagrid node

- here
- nodes

- ORNL metagrid

2.2.2 Filtering ESM components

-
-
-
-


Select a Project
CMIP6
CMIP6 Website
Filter By Transfer Options
Any
Only Globus Transferrable
Collapse All
Filter with Facets
General
Activity ID : CMIP (1)
Data Node : Select option(s)
Identifiers
Source ID : ACCESS-CM2 (1)
Institution ID : Select option(s)
Source Type : Select option(s)
Experiment ID : historical (1)
Sub Experiment ID : Select option(s)
Resolutions
Labels
Variant Label : r1i1p1f1 (1)
Grid Label : Select option(s)
Classifications
Table ID : Select option(s)
Frequency : mon (1)
Realm : Select option(s)
Variable ID : tos (1)

2.2.3 Download `wget` shell script

2.2.4 Rinse and repeat

2.3 Download `.nc` files with `wget` scripts

> **Warning: storage**
>
> In total, we are about to download eight `.nc` files that comprise 2.65 GB of data. These are just for two models and two scenarios + historical, at a monthly time-step. Storing ESM outputs can take a lot of space - for example, if you're working with daily data across multiple models and climate scenarios, you'll almost certainly need an external hard drive or other storage solution.

2.3.1 Define dependencies

```
source(paste0(getwd(), "/__scripts/helpers.R"))
```

2.3.2 Prepare for parallel processing

```
parallelly::availableCores() # Output says I have 16 cores on my machine
w <- 14  # Leave two cores for other background processes
```

2.3.3 Function to download files

```
wget_files <- function(script) {
  setwd(paste0(pth, cmip_pth)) #set directory to where data will be stored
  system(paste0("bash ", script, " -s")) #run wget on the shell script
  setwd(pth) #set directory back to home directory
}
```

```
files <- list.files(paste0(pth, wget_pth), #full path where wget scripts are stored
                    pattern = "wget", #only files with wget in the name
                    full.names = TRUE #list the full path
                    )
files
```

2.3.4 Run function in parallel

```
plan(multisession, workers = w) # Change to multi-threaded processing
tic(); future_walk(files, wget_files); toc() #Run the function in parallel
plan(sequential) # Return to single threaded processing (i.e., sequential/normal)
```

> **Download speeds**
>
> Download speeds depend on your Wifi connection, the capabilities of your local machine, and the performance of the server/node you're attempting to connect to. You can check whether ESGF servers are online here.

```
list.files(paste0(pth, cmip_pth))
```

```
system(paste0("rm ", pth, cmip_pth, "/",
              "tos_Omon_ACCESS-CM2_ssp585_r1i1p1f1_gn_210101-230012.nc"))
system(paste0("rm ", pth, cmip_pth, "/",
              "tos_Omon_IPSL-CM6A-LR_ssp585_r1i1p1f1_gn_210101-230012.nc"))
```

Beware of `rm`

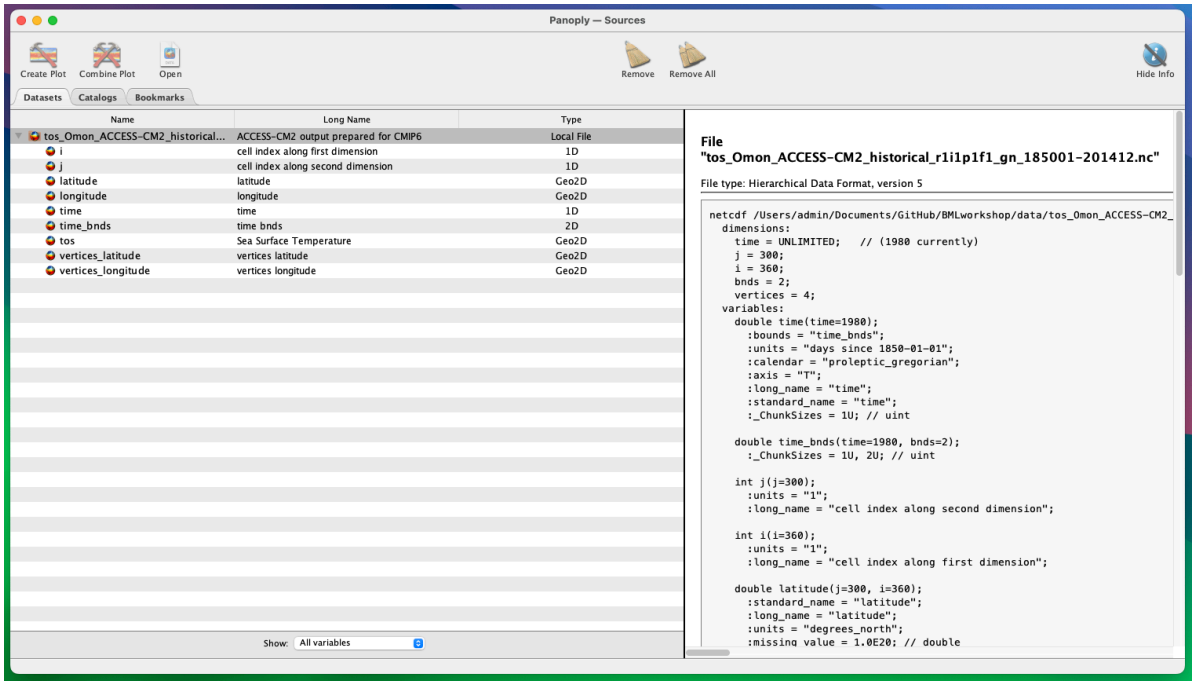
`rm` is a handy function for removing files, but there is no way of retrieving them if you've accidentally removed the wrong file. So, use this function wisely and make sure you're 100% confident you're removing the correct file!

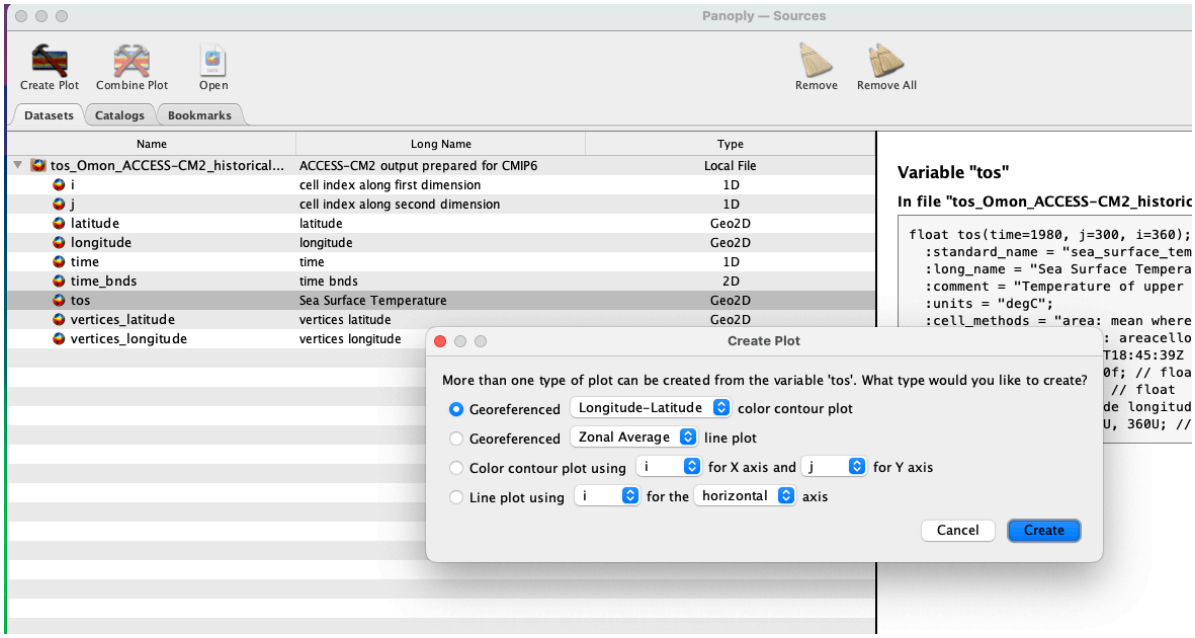
3 Remapping

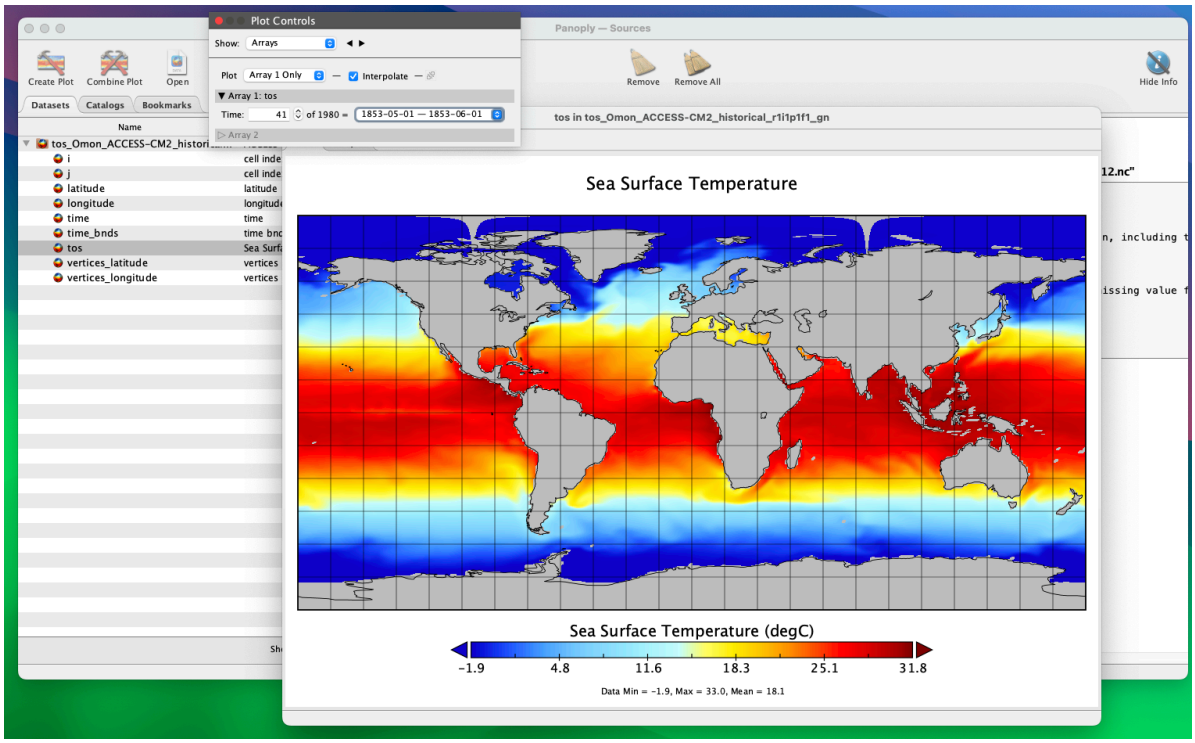
> **!** Important
>
> Note that interpolation, regridding and remapping mean the same thing (for the purposes of our workshop).

3.1 Inspect an ESM

3.1.1 Panoply

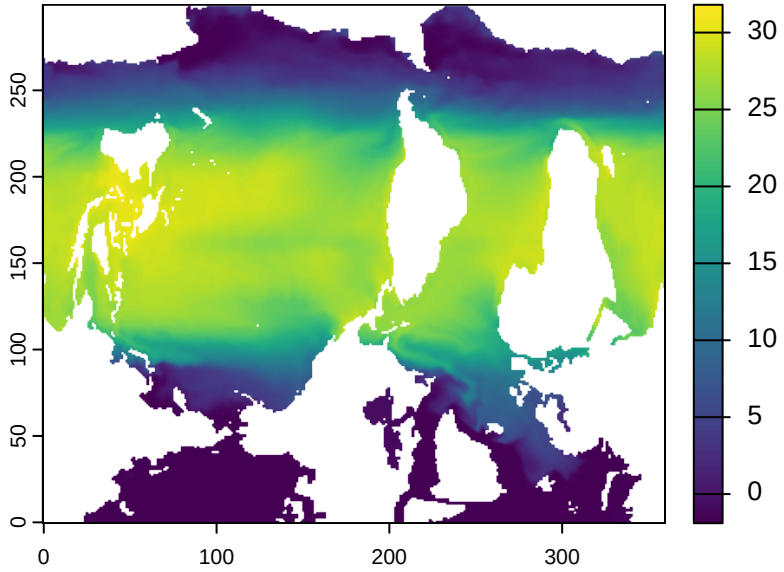

Panoply — Sources
Create Plot
Combine Plot
Open
Remove
Remove All
Hide Info
Datasets
Catalogs
Bookmarks
Name
Long Name
Type
tos_Omon_ACCESS-CM2_historical...
ACCESS-CM2 output prepared for CMIP6
Local File
i
cell index along first dimension
1D
j
cell index along second dimension
1D
latitude
latitude
Geo2D
longitude
longitude
Geo2D
time
time
1D
time_bnds
time bnds
2D
tos
Sea Surface Temperature
Geo2D
vertices_latitude
vertices latitude
Geo2D
vertices_longitude
vertices longitude
Geo2D
Show:
All variables
File
"tos_Omon_ACCESS-CM2_historical_r1i1p1f1_gn_185001-201412.nc"
File type: Hierarchical Data Format, version 5
netcdf /Users/admin/Documents/GitHub/BMLworkshop/data/tos_Omon_ACCESS-CM2_
dimensions:
time = UNLIMITED; // (1980 currently)
j = 300;
i = 360;
bnds = 2;
vertices = 4;
variables:
double time(time=1980);
:bounds = "time_bnds";
:units = "days since 1850-01-01";
:calendar = "proleptic_gregorian";
:axis = "T";
:long_name = "time";
:standard_name = "time";
:_ChunkSizes = 1U; // uint
double time_bnds(time=1980, bnds=2);
:_ChunkSizes = 1U, 2U; // uint
int j(j=300);
:units = "1";
:long_name = "cell index along second dimension";
int i(i=360);
:units = "1";
:long_name = "cell index along first dimension";
double latitude(j=300, i=360);
:standard_name = "latitude";
:long_name = "latitude";
:units = "degrees_north";
:missing value = 1.0E20; // double


Panoply — Sources
Create Plot
Combine Plot
Open
Remove
Remove All
Datasets
Catalogs
Bookmarks
Name
Long Name
Type
tos_Omon_ACCESS-CM2_historical...
ACCESS-CM2 output prepared for CMIP6
Local File
i
cell index along first dimension
1D
j
cell index along second dimension
1D
latitude
latitude
Geo2D
longitude
longitude
Geo2D
time
time
1D
time_bnds
time bnds
2D
tos
Sea Surface Temperature
Geo2D
vertices_latitude
vertices latitude
Geo2D
vertices_longitude
vertices longitude
Variable "tos"
In file "tos_Omon_ACCESS-CM2_historic
float tos(time=1980, j=300, i=360);
:standard_name = "sea_surface_tem
:long_name = "Sea Surface Tempera
:comment = "Temperature of upper
:units = "degC";
:cell_methods = "area: mean where
Create Plot
More than one type of plot can be created from the variable 'tos'. What type would you like to create?
Georeferenced
Longitude-Latitude
color contour plot
Georeferenced
Zonal Average
line plot
Color contour plot using
i
for X axis and
j
for Y axis
Line plot using
i
for the
horizontal
axis
Cancel
Create



3.1.2 R

```
rr <- rast(paste0(pth, cmip_pth, "/tos_Omon_ACCESS-CM2_historical_r1i1p1f1_gn_185001-201412.n
rr <- rr[[1]] #the first layer
plot(rr)
```



3.2 Remapping

> **Caution**
>
> You can also use **first order conservative remapping** if bilinear interpolation fails (which may occur if the input grid is 'unstructured' as opposed to 'spherical').

```
rr
```

❗ We can also check more detailed metadata contained in the ESM file using `ncdf4::nc_open()`

This gives you the same information after reading a file into `Panoply`.

```
nc <- nc_open(paste0(pth, cmip_pth, "/tos_Omon_ACCESS-CM2_historical_r1i1p1f1_gn_185001-201
nc
```

```
File /Users/admin/Documents/GitHub/esmrworkshop_rladies26/__data/cmip6_raw/tos_Omon_ACCESS-

     6 variables (excluding dimension variables):
        double time_bnds[bnds,time]   (Chunking: [2,1])  (Compression: level 1)
        double latitude[i,j]   (Chunking: [360,300])  (Compression: level 1)
            standard_name: latitude
            long_name: latitude
            units: degrees_north
            missing_value: 1e+20
            _FillValue: 1e+20
            bounds: vertices_latitude
        double longitude[i,j]   (Chunking: [360,300])  (Compression: level 1)
            standard_name: longitude
            long_name: longitude
            units: degrees_east
            missing_value: 1e+20
            _FillValue: 1e+20
            bounds: vertices_longitude
        double vertices_latitude[vertices,i,j]   (Chunking: [2,360,300])  (Compression: lev
            units: degrees_north
            missing_value: 1e+20
            _FillValue: 1e+20
        double vertices_longitude[vertices,i,j]   (Chunking: [2,360,300])  (Compression: le
            units: degrees_east
            missing_value: 1e+20
```

```
        _FillValue: 1e+20
    float tos[i,j,time]   (Chunking: [360,300,1])  (Compression: level 1)
        standard_name: sea_surface_temperature
        long_name: Sea Surface Temperature
        comment: Temperature of upper boundary of the liquid ocean, including temperatu
        units: degC
        cell_methods: area: mean where sea time: mean
        cell_measures: area: areacello
        history: 2019-11-08T18:45:39Z altered by CMOR: replaced missing value flag (-1e
        missing_value: 1.00000002004088e+20
        _FillValue: 1.00000002004088e+20
        coordinates: latitude longitude

 5 dimensions:
    time  Size:1980   *** is unlimited ***
        bounds: time_bnds
        units: days since 1850-01-01
        calendar: proleptic_gregorian
        axis: T
        long_name: time
        standard_name: time
    j  Size:300
        units: 1
        long_name: cell index along second dimension
    i  Size:360
        units: 1
        long_name: cell index along first dimension
    bnds  Size:2 (no dimvar)
    vertices  Size:4 (no dimvar)

47 global attributes:
    Conventions: CF-1.7 CMIP-6.2
    activity_id: CMIP
    branch_method: standard
    branch_time_in_child: 0
    branch_time_in_parent: 0
    creation_date: 2019-11-08T18:45:44Z
    data_specs_version: 01.00.30
    experiment: all-forcing simulation of the recent past
    experiment_id: historical
    external_variables: areacello
```

```
        forcing_index: 1
        frequency: mon
        further_info_url: https://furtherinfo.es-doc.org/CMIP6.CSIRO-ARCCSS.ACCESS-CM2.hist
        grid: native atmosphere N96 grid (144x192 latxlon)
        grid_label: gn
        history: 2019-11-08T18:45:44Z ; CMOR rewrote data to be consistent with CMIP6, CF-1
        initialization_index: 1
        institution: CSIRO (Commonwealth Scientific and Industrial Research Organisation, A
        institution_id: CSIRO-ARCCSS
        mip_era: CMIP6
        nominal_resolution: 250 km
        notes: Exp: CM2-historical; Local ID: bj594; Variable: tos (['sst'])
        parent_activity_id: CMIP
        parent_experiment_id: piControl
        parent_mip_era: CMIP6
        parent_source_id: ACCESS-CM2
        parent_time_units: days since 0950-01-01
        parent_variant_label: r1i1p1f1
        physics_index: 1
        product: model-output
        realization_index: 1
        realm: ocean
        run_variant: forcing: GHG, Oz, SA, Sl, Vl, BC, OC, (GHG = CO2, N2O, CH4, CFC11, CFC
        source: ACCESS-CM2 (2019):
aerosol: UKCA-GLOMAP-mode
atmos: MetUM-HadGEM3-GA7.1 (N96; 192 x 144 longitude/latitude; 85 levels; top level 85 km)
atmosChem: none
land: CABLE2.5
landIce: none
ocean: ACCESS-OM2 (GFDL-MOM5, tripolar primarily 1deg; 360 x 300 longitude/latitude; 50 lev
ocnBgchem: none
seaIce: CICE5.1.2 (same grid as ocean)
        source_id: ACCESS-CM2
        source_type: AOGCM
        sub_experiment: none
        sub_experiment_id: none
        table_id: Omon
        table_info: Creation Date:(30 April 2019) MD5:e14f55f257cceafb2523e41244962371
        title: ACCESS-CM2 output prepared for CMIP6
        variable_id: tos
        variant_label: r1i1p1f1
```

```
version: v20191108
cmor_version: 3.4.0
tracking_id: hdl:21.14100/0bcaaa74-aedb-4d45-a5e5-cb3ab467f2b5
license: CMIP6 model data produced by CSIRO is licensed under a Creative Commons At
```

3.2.1 A (brief) intro to CDO

Common CDO operators

`cdo -yearmean` calculates the annual mean of a monthly data input netCDF file
`cdo -yearmin` calculates the annual min of a monthly data input netCDF file
`cdo -yearmax` calculates the annual max of a monthly data input netCDF file
`cdo -ensmean` calculates the ensemble mean of several netCDF files. If your input files are different models, this function will estimate a mean of all those models
`cdo -vertmean` calculates the vertical mean for netCDF with olevel (i.e., depth)
`cdo -mergetime` merge all the netCDF files in your directory

3.2.2 Remapping with CDO and R

```
remap_n_crop_temp <- function(nc_file,
                              cell_res = 0.25,
                              infold = paste0(pth, cmip_pth),
                              outfold = paste0(pth, cmip_pth_proc),
                              xmin = -126, xmax = -115, ymin = 32, ymax = 43) {

  system(paste0("cdo -L -sellonlatbox,", xmin, ",", xmax, ",", ymin, ",", ymax,
                " -remapbil,r", 360*(1/cell_res), "x", 180*(1/cell_res),
                " -select,name=tos ", infold, "/", nc_file, " ", outfold, "/", nc_file))

}

fileys <- list.files(paste0(pth, cmip_pth)) #list file names of downloaded ESMs

w <- 14 #number of workers
plan(multisession, workers = w) # Change to multi-threaded processing
tic(); future_walk(fileys, remap_n_crop_temp); toc() #Run the function in parallel (takes 8 s
plan(sequential) # Return to single threaded processing
```

3.2.3 Function explainer

-
-
-
-

-

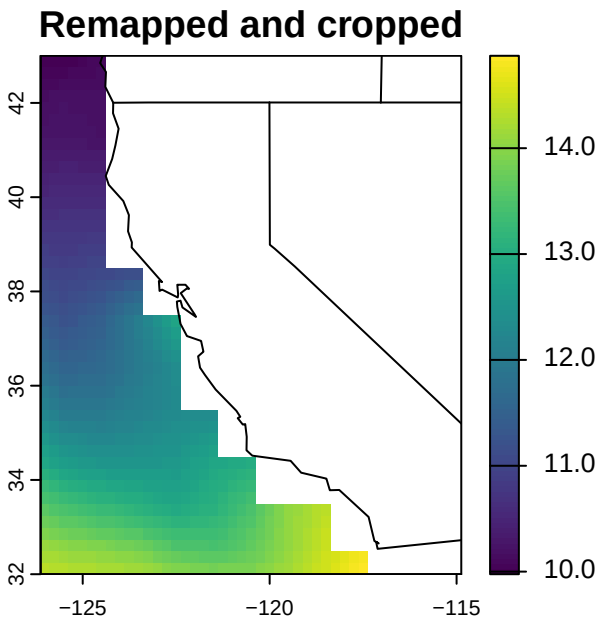
-
-
-
-

-
-
-
-

> **!** Important
>
> `CDO` requires syntax (i..e, spaces and commas) to be precise. If your code doesn't run, first check that you have them in the right places.

3.3 Check that it worked

```
# Read in one of the processed output files
rr <- rast(paste0(pth, cmip_pth_proc, "/tos_Omon_ACCESS-CM2_historical_r1i1p1f1_gn_185001-20
rr <- rr[[1]] #first layer
plot(rr, main = "Remapped and cropped")
maps::map("state", add = T) #add US state boundaries
```




```
rr #display metadata
```

4 OISST

Caution

We will not run this script during the webinar due to time constraints. Instead, the script and all required information are provided below. Preprocessed (i.e., regridded) OISST outputs are available in `__data/oisst_processed`.

OISST V2.1

Oceanography terminology: 'observations'

Whether data are referred to as **observations** depends on who you to talk to. For the purposes of our workshop, and as marine ecologists, we refer to `OISST` as our 'observed' data set, because it represents 'real-life' SST. However, sea-going physical oceanographers might not refer to `OISST` as observations, because the product represents a series of observations obtained from different platforms (i.e., ARGO, moorings, satellites, ship-borne sensors etc.) that have been 'modeled' or 'interpolated' into a gridded, cloud-free product. Technically, OISST represents a modeled product *based* on observations. Rather, they would refer to observations as the raw, unprocessed data coming from sensors themselves. Just something to keep in mind when talking to people from different disciplines.

4.1 Download OISST

here

```
# Set destination folder
oFold <- paste0(pth, "/__data/oisst_raw/")
# Set base URL
url <- "https://www.ncei.noaa.gov/data/sea-surface-temperature-optimum-interpolation/v2.1/ac

# Get the links that appear there as yyyymm dates
pg <- read_html(url) # Read the HTML
sFld <- html_attr(html_nodes(pg, "a"), "href") %>%  # Extract the links
  grep("\\d\\d\\d\\d\\d\\d", ., value = TRUE) # Just the folders with 6 digits (i.e., one pe

# Check what files might already be in the folder
f <- dir(oFold, pattern = ".nc")

# Extract identifiers from the file names - works at level of month
ncs <- gsub("oisst-avhrr-v02r01.", "", f) %>%
  gsub("\\d\\d\\.nc", "", .) %>%
  unique() %>%
  paste0(., "/")

# Exclude files (months) that are already downlaoded
sFld <- sFld[!sFld %in% ncs]

download_oisst <- function(yrstrt, yrend) {

  # Subset sFld to only include time period of interest
  ind1 <- grep(yrstrt, sFld) %>% min
  ind2 <- grep(yrend, sFld) %>% max
  sFld <- sFld[ind1:ind2]
  length(sFld) #downloading 372 files for 30 yrs

  for(i in sFld) {
    urli <- paste0(url, "/", i)
    pgi <- read_html(urli) # Read the HTML
    sFldi <- html_attr(html_nodes(pgi, "a"), "href") %>%  # Extract the links
```

```
      grep(".nc", ., value = TRUE) # Just the netCDFs
    for(j in sFldi) {
      download.file(paste0(urli, "/", j), paste0(oFold, j))
    }
  }
}

tic(); download_oisst(yrstrt = 1995, yrend = 2014); toc()
```

4.2 Preprocess OISST

```
oisst_mr <- function(yr,
                     infile = paste0(pth, oisst_pth),
                     outfile = paste0(pth, oisst_pth_proc),
                     xmin = -126, xmax = -116, ymin = 32, ymax = 43,
                     cell_res = 0.25) {


  # Combine all daily files for X year into one file
  merged_1yr <- paste0("cdo -mergetime ",
                 infile, "/", "oisst-avhrr-v02r01.", yr, "*.nc ",
                 outfile, "/", "oisst-avhrr-merged_", yr, ".nc")
  system(merged_1yr) # takes a few seconds


  # Calculate monthly means for X year
  mthmeans <- paste0("cdo -L monmean ",
                   outfile, "/oisst-avhrr-merged_", yr, ".nc ",
                   outfile, "/mean_", yr, ".nc")
  system(mthmeans)

  # Select SST, crop and remap
  oisst_regrid <- paste0("cdo -L -select,name=,sst ",
                   "-sellonlatbox,", xmin, ",", xmax, ",", ymin, ",", ymax,
```

```
                    " -remapbil,r", 360*(1/cell_res), "x", 180*(1/cell_res),
                    " ", outfile, "/mean_", yr, ".nc",
                    " ", outfile, "/mean_remap_", yr, ".nc")
  system(oisst_regrid)

  # Remove temporary files
  system(paste0("rm ", outfile, "/", "oisst-avhrr-merged_", yr, ".nc"))
  system(paste0("rm ", outfile, "/mean_", yr, ".nc"))

}

# Vector of years to process
years <- 1995:2014  # Modify as needed

plan(multisession, workers = 14)  # Change workers to suit your machine
tic(); future_map(years, ~ oisst_mr(.x)); toc()
plan(sequential) # Return to sequential processing
```

```
list.files(paste0(pth, "/__data/oisst_processed"))
```

! Leap years

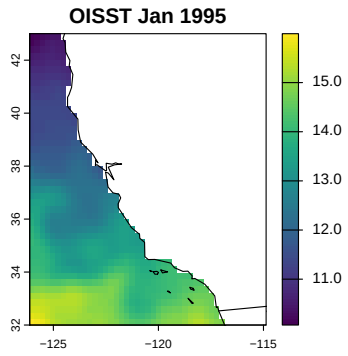
Here, we've created monthly means of OISST from the daily SST fields, to match the temporal resolution of our monthly ESM projections. However, if working with with daily data, you might want to remove February 29th to get around any calendar issues when merging files for bias correction (particularly if working with daily ESM data). This code removes the 29th day of February, and sets the calendar to Gregorian (i.e., 365 days) for `infile`. We skipped this today to save time, as it can take a while to process (merging

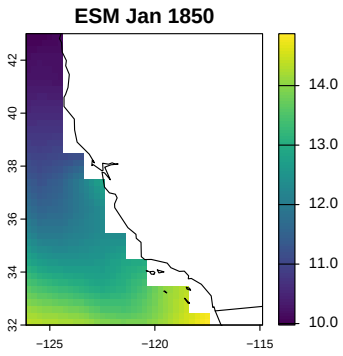
20 years' worth of daily files is processor-intensive, and results in a large file).

```
cdo_code <- paste0("cdo -L -setcalendar,365_day -delete,month=2,day=29 ", infile, " ", outf
system(cdo_code)
```

4.3 Visualise

```
oisst <- terra::rast(paste0(pth, oisst_pth_proc, "/mean_remap_1995.nc"))[[1]]
esm <- terra::rast(paste0(pth, cmip_pth_proc, "/tos_Omon_ACCESS-CM2_historical_r1i1p1f1_gn_18
terra::plot(oisst, main = "OISST Jan 1995")
maps::map("world", add = T)
terra::plot(esm, main = "ESM Jan 1850")
maps::map("world", add = T)
```





5 Bias correction & downscaling

> **Caution**
>
> We will not run this script during the webinar due to time constraints. Instead, the script and all required information are provided below. Bias-corrected outputs are available in `__data/bias_correct`.

> **Important**
>
> Note - a lot of physical oceanographers caution against interpolating or bias-correcting ESMs to a (much) higher resolution, as results can often be misleading. Today, we'll show you how to downscale an ESM from 1 ° to 0.25 ° and perform bias-correction; but always best to chat to an oceanographer first, when working with your own study system.

OISST V2.1

! Defining climatologies

We often use 30 years to define a climatology in marine ecology, but you can shorten or lengthen your climatology based on what makes sense for your question and study system. We used 20 years today to save time.

5.1 Merge daily OISST files

```
# List all processed OISST files
day_ncs <- dir(paste0(pth, oisst_pth_proc), full.names = TRUE) %>%
  paste0(., collapse = " ")

# CDO code
cdo_code <- paste0("cdo -s -L ", #Deploy CDO silently and in low memory mode
                   "-f nc4 ",  # Output file should be netCDF format
                   "-z zip ", # Compress the output file using zip
                   "-mergetime ", # Merge multiple input netCDF files
                   day_ncs, # The names of the input files
                   " ",
                   paste0(pth, bc_pth, "/_1_OISST_baseline_combined.nc")) # output file

# Run
tic(); system(cdo_code); toc()
```

```
rr <- terra::rast(paste0(pth, bc_pth, "/_1_OISST_baseline_combined.nc"))
rr
```

5.2 Delete leap days

> **Note**
>
> Here, we've already created monthly means of OISST from the daily data. However, if you're working with daily data, you might want to remove February 29th to get around any calendar issues when merging files. This code removes the 29th day of February, and sets the calendar to Gregorian (i.e., 365 days).

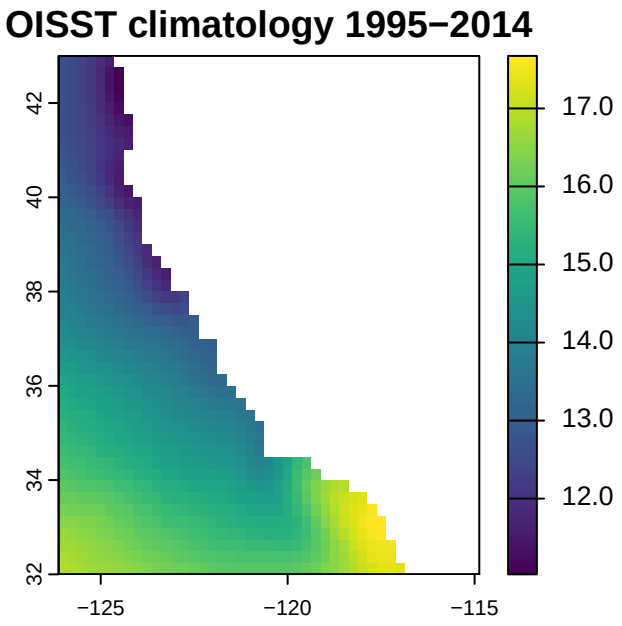
```
cdo_code <- paste0("cdo -L -setcalendar,365_day -delete,month=2,day=29 ", infile, " ", outf
system(cdo_code)
```

5.3 Create OISST climatology (1995-2014)

```
input_file <- paste0(pth, bc_pth, "/_1_OISST_baseline_combined.nc")
output_file <- paste0(pth, bc_pth, "/_2_OISST_climatology.nc")

# Calculate the mean in each grid cell (across all years in the time period)
cdo_code <- paste0("cdo timmean ", input_file, " ", output_file )
system(cdo_code)
```

```
r <- terra::rast(paste0(pth, bc_pth, "/_2_OISST_climatology.nc"))
r #ignore 'time' field
```

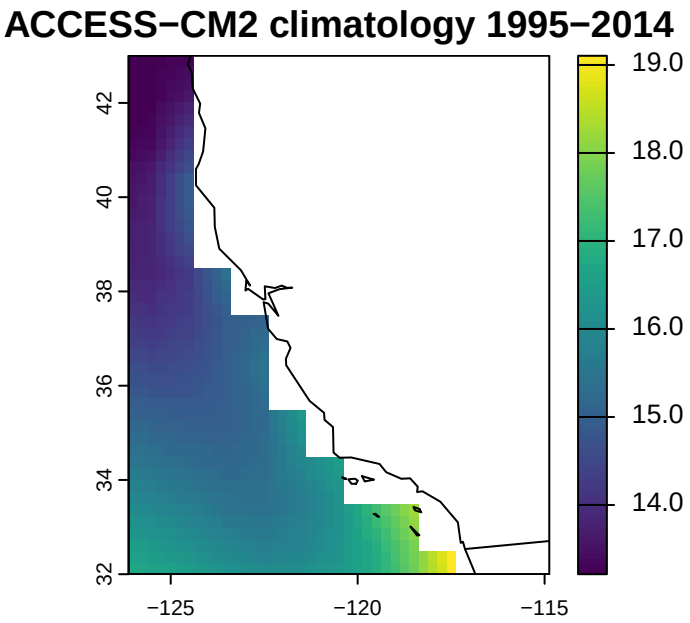
```
plot(r, main = "OISST climatology 1995-2014") #smoothed over
```



5.4 Create ESM historical climatology (1995-2014)

```
histclim <- function(model) {

### HISTORICAL
# This creates a climatology/baseline 1993-2014 file using the ESM's historical run
# Create baseline file 1993-2014 and mean
filey <- list.files(paste0(pth, cmip_pth_proc),
                    pattern = "historical", full.names = T)
filey <- filey[grep(model, filey)]
rr <- rast(filey)
#time(rr) <- as.Date(time(rr))
rr <- rr[[time(rr) > as.Date("1995-01-01")]]
mean_rr <- mean(rr)
terra::writeCDF(mean_rr,
                paste0(pth, bc_esm_pth, "/_1_climatology/tos_mo_", model,
                       "_1995-2014_clim_historical.nc"),
                overwrite = T)

}


histclim("ACCESS-CM2")
histclim("IPSL-CM6A-LR")
```

```
clim <- terra::rast(paste0(pth, bc_esm_pth, "/_1_climatology/tos_mo_ACCESS-CM2_1995-2014_cli
plot(clim, main = "ACCESS-CM2 climatology 1995-2014"); maps::map("world", add = T)
```

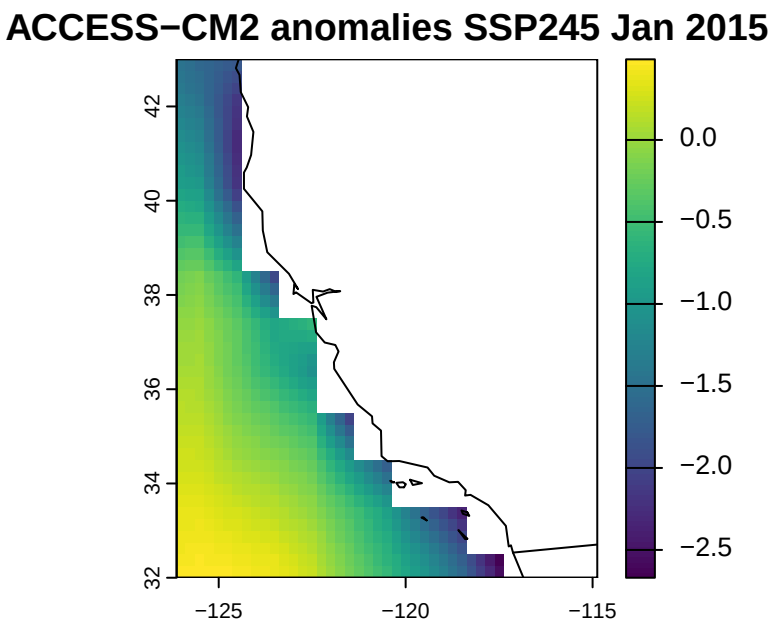


5.5 Create anomalies for ESM projections (2015-2100)

```
anom_esm <- function(model) {

  mean_rr <- rast(paste0(pth, bc_esm_pth, "/_1_climatology/tos_mo_", model,
                   "_1995-2014_clim_historical.nc"))

  ### SSPS
  # Then substract mean climatology from 2015-2100 for both EMSs, creating anomalies
  filey <- list.files(paste0(pth, cmip_pth_proc),
                      pattern = "ssp", full.names = T)
  filey <- filey[grep(model, filey)]

  for (i in 1:length(filey)) {

    #isolate SSP from string
    ssp_string <- strsplit(filey[i], model, "_")[[1]][2]
    ssp <- strsplit(ssp_string, "_r1i1p1f1")[[1]][1]

    #import raster and calc anomalies
```

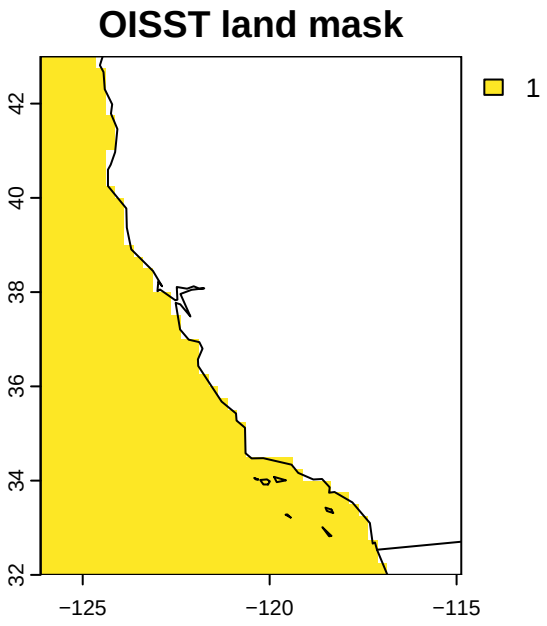
```
    scen <- rast(filey[i])
    anom <- scen - mean_rr
    writeCDF(anom, paste0(pth, bc_esm_pth, "/_2_anomalies/tos_mo_",
                          model, "_2015-2100_anom", ssp, ".nc"),
             overwrite = T)
  }
}

# Run function
tic(); anom_esm("ACCESS-CM2"); toc() #Jessie: 0.689 seconds
anom_esm("IPSL-CM6A-LR")
```

```
rr <- terra::rast(paste0(pth, bc_pth, bc_pth_anom, "/tos_mo_ACCESS-CM2_2015-2100_anom_ssp245
plot(rr[[1]], main = "ACCESS-CM2 anomalies SSP245 Jan 2015"); maps::map("world", add = T)
```



5.6 Create a land mask with OISST

```
infile <- paste0(pth, bc_pth, "/_2_OISST_climatology.nc")
outfile <- paste0(pth, bc_pth, "/_3_OISST_mask.nc")

# make a mask saying "NA:land, 1:ocean"
system(paste0("cdo -expr,'sst = ((sst>-2)) ? 1.0 : sst/0.0' ", infile, " ", outfile))

#Plotting the mask results in...
r <- rast(outfile)
plot(r, main = "OISST land mask"); values(r) %>% range(na.rm=T); maps::map("world", add = T)
```



5.7 Interpolate ESM anomalies to OISST grid

```
input_folder <- paste0(pth, bc_pth, "/", bc_pth_anom)
output_folder <- paste0(pth, bc_pth, "/", bc_pth_anom_rm)
msk <- paste0(pth, bc_pth, "/_3_OISST_mask.nc")

# Function to do the re-gridding ------------------------------------------

do_regrid <- function(f) {

  output_file <- basename(f) %>% # Get input file name
    gsub(".nc", "_remapped.nc", .) %>% # Add "_remapped" to end of file name
    paste0(output_folder, "/", .)

  # Step 1 - set grid via nco
  cdo_code <- paste0('ncatted -O -a units,longitude,c,c,"degrees_east" -a units,latitude,c,c
                     f, " ", paste0(output_folder, "/tmp1.nc"))
  system(cdo_code)

  # Get variable name
  rr <- rast(paste0(output_folder, "/tmp1.nc"))
  vary <- varnames(rr)

  # Step 2 - remove lat/lon attributes
  cdo_code2 <- paste0('ncatted -O -a grid_mapping,', vary, ',d,, ',
                      paste0(output_folder, "/tmp1.nc"), ' ',
                      paste0(output_folder, "/tmp2.nc"))
  system(cdo_code2)

  # Step 3 - remap ESM anomaly with OISST mask using bilinear interpolation
  # Set missing vals to nearest neighbour
  cdo_code3 <- paste0("cdo -s -L -f nc4 -z zip ", # Zip the file up
                      "-setmisstonn ", # Set missing values to nearest neighbor value
```

```
                      "-remapbil,", msk, " ", # Remap with OISST mask sing bilinear interpola
                      paste0(output_folder, "/tmp2.nc"), " ", #input file
                      paste0(output_folder, "/tmp3.nc"))  # output file
  system(cdo_code3)

  # Step 4 - remove mask
  cdo_code <- paste0("cdo -s -L -f nc4 -z zip ", # Zip the file up
                     "-mul ", msk, " ", # Multiply the result of the lines, below by the mask
                     paste0(output_folder, "/tmp3.nc"),
                     " ", output_file) # Mask the remapped anomaly file,  and save as output_
  system(cdo_code)

  # Step 5 -remove temporary files
  system(paste0("rm ", output_folder, "/tmp1.nc", " ",
                output_folder, "/tmp2.nc ",
                output_folder, "/tmp3.nc"))
}

# Run function ------------------------------------------------------------

files <- list.files(input_folder, pattern = "anom", full.names = TRUE) # Files to process
 plan(multisession, workers = 14) # Setting up to run in parallel, change workers to suit you
 tic(); furrr::future_walk(files, do_regrid); toc()
 plan(sequential) # Go back to sequential processing

 # If having trouble with using future_walk(), you can try the for loop version.
 # However, this takes longer, since it iterates over each file one at a time!
 # for (f in files) {
 #   do_regrid(f)
 #   print(f)
 # }
```

❗ There's a lot going on in this function. Let's break it down...

5.7.1 Create output_file name

Takes the input file (e.g., `tos_mo_ACCESS-CM2_2015-2100_anom_ssp245.nc`), and adds `_remapped` to the end of the file name.

```
output_file <- basename(f) %>% # Get input file name
    gsub(".nc", "_remapped.nc", .) %>% # Add "_remapped" to end of file name
    paste0(output_folder, "/", .)

output_file
#[1] "/Users/admin/Documents/GitHub/BMLworkshop/__data/bias_correct/esm/_3_anomalies_remapp
```

5.7.2 Set grid via NCO

This takes the file created above, and uses the `ncatted` command from `NCO` to modify the input file's `longitude` and `latitude` variables' attribute units, setting them to `degrees_east` and `degrees_north`, respectively. The output is saved as `tmp1.nc`. We then get the variable name from the file, which will be the original name of the file, e.g., `tos_mo_ACCESS-CM2_2015-2100_anom_ssp245`.

```
  cdo_code <- paste0('ncatted -O -a units,longitude,c,c,"degrees_east" -a units,latitude,c,
                     f, " ", paste0(output_folder, "/tmp1.nc"))
  system(cdo_code)

  rr <- rast(paste0(output_folder, "/tmp1.nc"))
  vary <- varnames(rr)
```

5.7.3 Remove lon/lat attributes

This code removes the `grid_mapping` attribute from the variable(s) specified in `vary`, and save the output as `tmp2.nc`.

```
  cdo_code2 <- paste0('ncatted -O -a grid_mapping,', vary, ',d,, ',
                      paste0(output_folder, "/tmp1.nc"), ' ',
                      paste0(output_folder, "/tmp2.nc"))
  system(cdo_code2)
```

5.7.4 Remapping

Here, we remap the ESM anomalies with the OISST mask `msk` using bilinear interpolation, and set missing values to nearest neighbor. The output is saved as `tmp3.nc`.

```
  cdo_code3 <- paste0("cdo -s -L -f nc4 -z zip ", # Zip the file up
                      "-setmisstonn ", # Set missing values to nearest neighbor value
                      "-remapbil,", msk, " ", # Remap with OISST mask sing bilinear interpo
                      paste0(output_folder, "/tmp2.nc"), " ", #input file
                      paste0(output_folder, "/tmp3.nc"))  # output file
  system(cdo_code3)
```

5.7.5 Remove the OISST mask

This code uses the OISST mask to turn land values back to `NA`.

```
  cdo_code <- paste0("cdo -s -L -f nc4 -z zip ", # Zip the file up
                     "-mul ", msk, " ", # Multiply contents of file with the mask to make l
                     paste0(output_folder, "/tmp3.nc"),
                     " ", output_file) # Mask the remapped anomaly file,  and save as outpu
  system(cdo_code)
```

5.7.6 Remove temporary files

We use the `rm` command (i.e., a base LINUX function) to remove the three temporary files created previously.

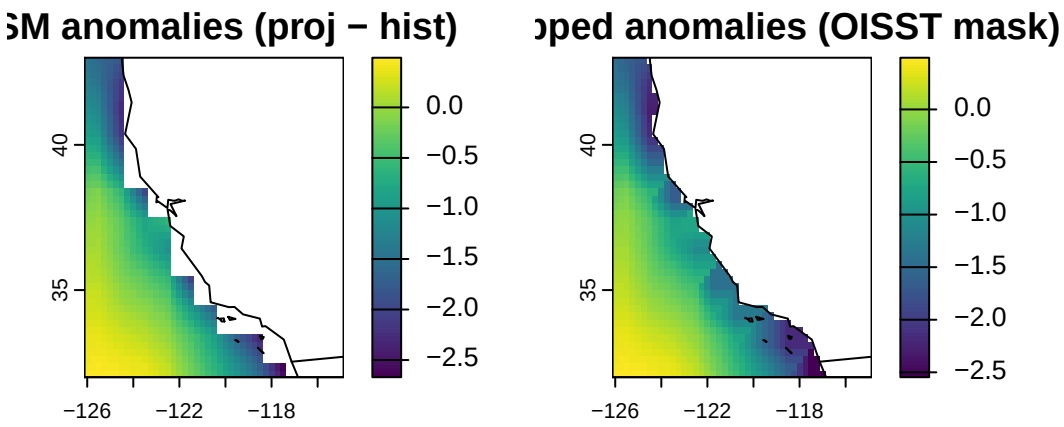
```
  system(paste0("rm ", output_folder, "/tmp1.nc", " ",
                output_folder, "/tmp2.nc ",
                output_folder, "/tmp3.nc"))
```

5.7.7 Run the function

Here, we use `list.files()` to list all files in our `input_folder` with the pattern `anom` in the file name. We then set up our session to run in parallel, where I (Jessie) have set to 14 workers (you will need to change this to suit your machine). We then run the `do_regrid()` function in parallel using `future_walk`. Once the function has finished running, we set our session back to normal (i.e., sequential) processing.

```
files <- list.files(input_folder, pattern = "anom", full.names = TRUE) # Files to process
 plan(multisession, workers = 14) # Setting up to run in parallel, change workers to suit y
 tic(); furrr::future_walk(files, do_regrid); toc() #Jessie: 5.7 seconds
 plan(sequential) # Go back to sequential processing
```

```
par(mfrow=c(1,2))
rast(paste0(pth, bc_pth, bc_pth_anom, "/tos_mo_ACCESS-CM2_2015-2100_anom_ssp245.nc"))[[1]] %
  plot(main = "ESM anomalies (proj - hist)")
maps::map("world", add =T)
rast(paste0(pth, bc_pth, bc_pth_anom_rm, "/tos_mo_ACCESS-CM2_2015-2100_anom_ssp245_remapped
  plot(main = "Remapped anomalies (OISST mask)")
maps::map("world", add =T)
```



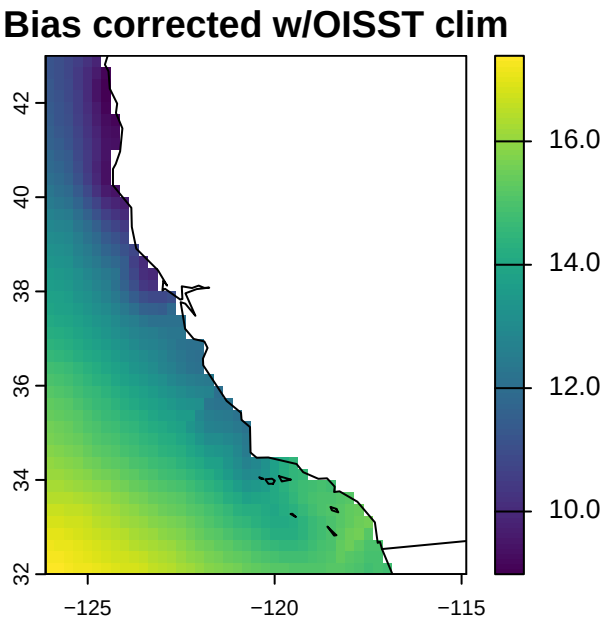
Troubleshooting CDO gridding errors

If you ever receive errors when using CDO relating to gridding, projections or remapping, it's a good idea to check the grid structure of the file using `griddes`, and ensure that the files you are manipulating/trying to create have the same grid structure as your input file.

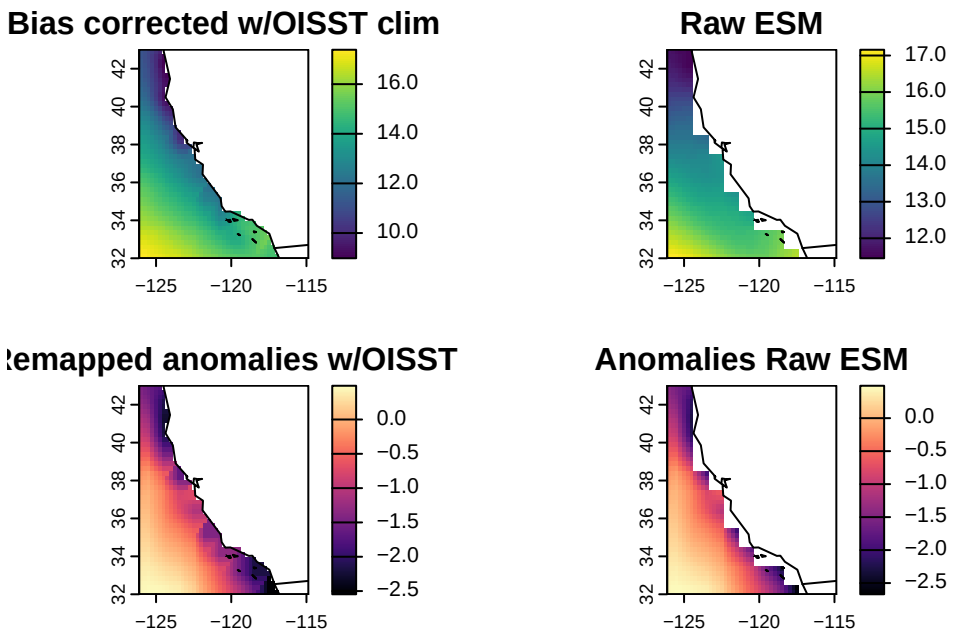
```
system(paste0("cdo griddes ", inputfile))
```

5.8 Bias correction

```
input_folder <- paste0(pth, bc_esm_pth)
obs <- paste0(pth, bc_pth, "/_2_OISST_climatology.nc")

# Function ----------------------------------------------------------------

do_add_clim <- function(f) {

  output_file <- basename(f) %>%
    gsub("_anom_", "_bc_", .) %>% # Replace anom code in file name with a code for bias corre
    paste0(input_folder, "/_4_bias_corrected/", .) # Include the path

  cdo_code <- paste0("cdo -s -L -f nc4 -z zip ", # Zip the file up
                     "-add ", f, " ", # To the remapped regridded anomalies, add...
                     obs, " ", output_file) # The observed climatology (map of means)
  system(cdo_code)
}

#  Run function -----------------------------------------------------------

files <- list.files(paste0(input_folder, "/_3_anomalies_remapped"),
                    pattern = "remapped", full.names = TRUE) # The files we want to process
tic(); walk(files, do_add_clim); toc()
```




```
par(mfrow=c(2,2))

input_folder <- paste0(pth, "/__data/bias_correct/esm")

rr <- rast(paste0(input_folder, "/_4_bias_corrected/tos_mo_ACCESS-CM2_2015-2100_bc_ssp245_re
plot(rr, main = "Bias corrected w/OISST clim")
maps::map("world", add = T)

tt <- rast(paste0(pth, cmip_pth_proc, "/tos_Omon_ACCESS-CM2_ssp245_r1i1p1f1_gn_201501-210012
plot(tt, main = "Raw ESM")
maps::map("world", add = T)

rr <- rast(paste0(pth, bc_esm_pth,  "/_3_anomalies_remapped/tos_mo_ACCESS-CM2_2015-2100_anom
plot(rr, col = viridis::magma(255), main = "Remapped anomalies w/OISST")
maps::map("world", add = T)

rr <- rast(paste0(pth, bc_esm_pth, "/_2_anomalies/tos_mo_ACCESS-CM2_2015-2100_anom_ssp245.nc
plot(rr, col = viridis::magma(255), main = "Anomalies Raw ESM")
maps::map("world", add = T)
```



5.9 Rinse and repeat

```
list.files(paste0(pth, "/__data/bias_correct/esm/_4_bias_corrected"),
           pattern = "1995-2014")
```

6 Evaluate accuracy of historical ESM projections

6.1 Brief explainer

-
-
-

6.2 Create dataframes of baseline climatologies

```r
source(paste0(pth, "/__scripts/helpers.R"))
```

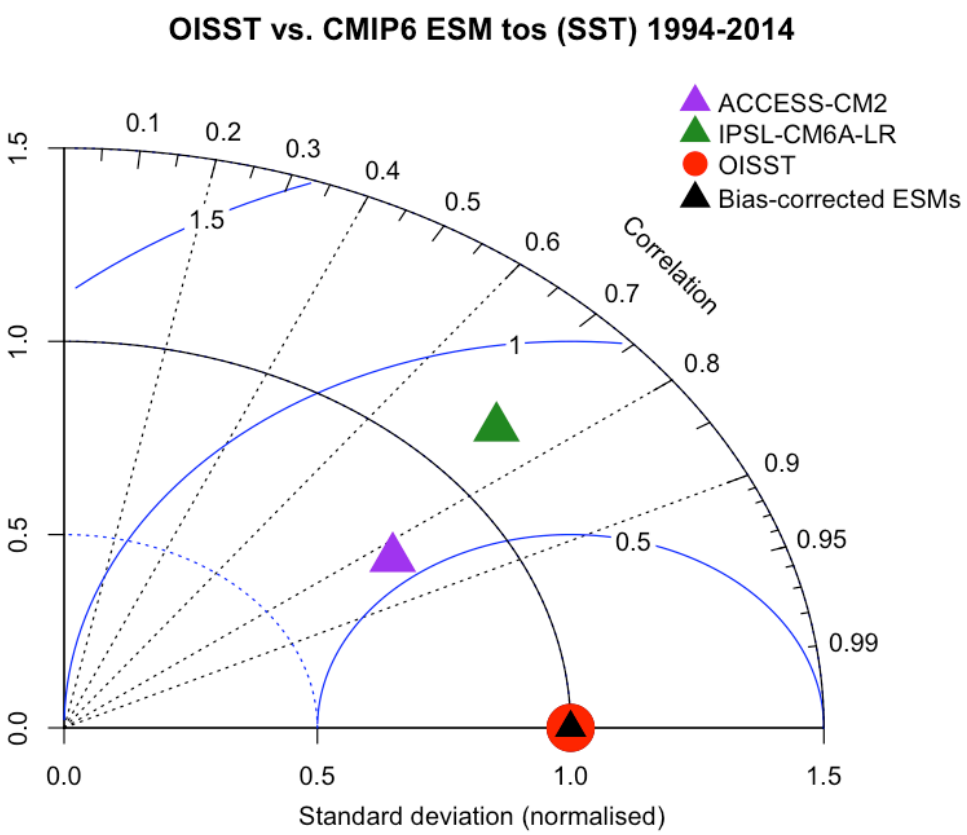
```r
library(terra)
library(tidyverse)
library(plotrix)

oisst <- rast(paste0(pth, bc_pth, "/_2_OISST_climatology.nc"))
access_bc <- rast(paste0(pth, bc_pth, bc_pth_bc, "/tos_mo_ACCESS-CM2_1995-2014_bc_historical_
ipsl_bc <- rast(paste0(pth, bc_pth, bc_pth_bc, "/tos_mo_IPSL-CM6A-LR_1995-2014_bc_historical_
access_raw <- rast(paste0(pth, cmip_pth_proc, "/tos_Omon_ACCESS-CM2_historical_r1i1p1f1_gn_18
access_raw <- access_raw[[time(access_raw) > "1994-01-01"]]
ipsl_raw <- rast(paste0(pth, cmip_pth_proc, "/tos_Omon_IPSL-CM6A-LR_historical_r1i1p1f1_gn_18
ipsl_raw <- ipsl_raw[[time(ipsl_raw) > "1994-01-01"]]


# Convert rasters to dataframes ------------------------------------------

# Calculate mean of each field (i.e., baseline/climatology)
oisst_df <- oisst %>% mean %>% as.data.frame(xy = T)
access_df <- access_raw %>% mean %>% as.data.frame(xy = T)
access_bc_df <- access_bc %>% mean %>% as.data.frame(xy = T)
ipsl_df <- ipsl_raw %>% mean %>% as.data.frame(xy = T)
ipsl_bc_df <- ipsl_bc %>% mean %>% as.data.frame(xy = T)


# Merge and fix names ----------------------------------------------------

alldata <- inner_join(oisst_df, access_df, by = c("x","y"))
names(alldata) <- c("x", "y", "oisst_mean", "access_mean")
alldata <- inner_join(alldata, access_bc_df, by = c("x","y"))
names(alldata) <- c("x", "y", "oisst_mean", "access_mean", "access_bc_mean")

alldata2 <- inner_join(oisst_df, ipsl_df, by = c("x","y"))
names(alldata2) <- c("x", "y", "oisst_mean", "ipsl_mean")
alldata2 <- inner_join(alldata2, ipsl_bc_df, by = c("x","y"))
names(alldata2) <- c("x", "y", "oisst_mean", "ipsl_mean", "ipsl_bc_mean")

alldata <- inner_join(alldata, alldata2)

alldata <- alldata %>% #ensemble mean
  mutate(ens_bc_mean = (access_bc_mean + ipsl_bc_mean) / 2)
```

```
alldata %>% head
```

6.3 Construct Taylor Diagram

```
taylor.diagram(alldata$oisst_mean,
               alldata$oisst_mean,
               ref.sd = T, #display arc of ref. std. dev. (i.e., 1)
               normalize=TRUE, #normalize models so ref has SD of 1
               sd.arcs=TRUE,  #display arcs along SD axes
               pcex = 4,
               pch = 19,
               col = "red",
               xlab = "Standard deviation (normalised)",
               pos.cor = T, #show correlation (y-axis) from 0-1
```

```
                gamma.col = "blue", #RMSE arcs
                main="OISST vs. CMIP6 ESM tos (SST) 1995-2014")

# Add ESM points
taylor.diagram(alldata$oisst_mean,
                alldata$access_mean,
                add=TRUE, normalize=TRUE,
                pcex=3, pch=17, col= "purple")

taylor.diagram(alldata$oisst_mean,
                alldata$ipsl_mean,
                add=TRUE, normalize=TRUE,
                pcex=3, pch=17, col= "forestgreen")

# Add bias-corrected point
taylor.diagram(alldata$oisst_mean,
                alldata$access_bc_mean,
                add=TRUE, normalize=TRUE,
                pcex=2, pch=17, col= "black")


# Legend ------------------------------------------------------------------

legend(1.2, 1.7, cex=1, pt.cex=2, pch=17,
       legend=c("ACCESS-CM2", "IPSL-CM6A-LR"),
       col=c("purple", "forestgreen"),
       bty = "n")

legend(1.2, 1.54, cex=1, pt.cex=2, pch=19,
       legend=c("OISST"),
       col= 'red',
       bty = "n")
legend(1.2, 1.45, cex=1, pt.cex=2, pch=17,
       legend=c("Bias-corrected ESMs"),
       col= 'black',
       bty = "n")
```


OISST vs. CMIP6 ESM tos (SST) 1994-2014
ACCESS-CM2
IPSL-CM6A-LR
OISST
Bias-corrected ESMs
Correlation
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0.95
0.99
1.5
1
0.5
0.0
0.5
1.0
1.5
0.0
0.5
1.0
1.5
Standard deviation (normalised)

7 Making projections

7.1 Time series

7.1.1 Create yearly averages of temperature

```
models = c("ACCESS-CM2|IPSL-CM6A-LR")
models2 = c("ACCESS-CM2", "IPSL-CM6A-LR")

indir_proj <- paste0(pth, "/__data/bias_correct/esm/_4_bias_corrected")
outdir <- "__data/timeseries"

# Projections ------------------------------------------------------------

timeseries_ssp <- function(ssp) {

  allfiles_proj <- list.files(indir_proj, pattern = models, full.names = T)
  allfiles_proj2 <- allfiles_proj[grep(ssp, allfiles_proj)]
  rr <- rast(allfiles_proj2)
  rr <- rr[[time(rr) < "2100-12-31"]] #Only data until 2100
  dateys <- time(rr) %>% unique
```

```r
years <- lubridate::year(dateys) %>% unique #get years

emplist <- list()
emplist_allmodels <- list()

### Ensemble mean
for (i in 1:length(years)) {

  alldates <- dateys[grep(years[i], dateys)]
  rasty <- rr[[time(rr) == alldates]]
  meanrast <- mean(rasty) # Average across the year
  meanval <- values(meanrast) %>% mean(na.rm=T) #Get average temp for California
  forlist <- data.frame(date = years[i], value = meanval)
  emplist[[i]] <- forlist
  print(paste0("ens_", years[i]))
}

### Individual models
for (j in 1:length(models2)) {
  emplist_indmodel <- list()

  for (h in 1:(length(years)-1)) {
    modelrast <- rast(allfiles_proj2[grep(models2[j], allfiles_proj2)])

    dateys <- time(modelrast) %>% unique
    alldates <- dateys[grep(years[h], dateys)]
    modelrast <- modelrast[[time(modelrast) == alldates]]
    meanrast <- mean(modelrast)
    meanval <- values(meanrast) %>% mean(na.rm=T)
    forlist <- data.frame(model = models2[j], date = years[h], value = meanval)
    emplist_indmodel[[h]] <- forlist
  }

  toadd <- do.call(rbind, emplist_indmodel)
  emplist_allmodels[[j]] <- toadd
  print(models2[j])
}

saveRDS(emplist,
        paste0("__data/timeseries/sst_year_proj_ens_",
               ssp, ".RDS"))
saveRDS(emplist_allmodels,
```

```
          paste0("__data/timeseries/sst_year_proj_ind_",
                 ssp, ".RDS"))
}

ssps <- c("ssp245", "ssp585")
tic(); future_walk(ssps, timeseries_ssp); toc() #26 seconds for both
```

```
# Ensemble mean
ens <- readRDS("__data/timeseries/sst_year_proj_ens_ssp245.RDS")
ens <- do.call (rbind, ens)
head(ens)
```

```
# For each ESM...
ind <- readRDS("__data/timeseries/sst_year_proj_ind_ssp245.RDS")
ind <- do.call (rbind, ind)
head(ind)
```

> **Caution**
>
> Today, we're using the ensemble mean. But in your own work, you may want to consider using the ensemble median instead. The median may be less impacted by outliers/extremes in the data, especially if you're using ESMs that are known to be 'too-hot' in your region of interest.

7.1.2 Bind everything and apply 5-yr smooth

```
smooth_esms <- function(ssp, window_size) {

# Ensemble mean
timeseries_proj_ens <- readRDS(paste0("__data/timeseries/sst_year_proj_ens_", ssp, ".RDS"))

  ens <- do.call(rbind, timeseries_proj_ens)
  zoo_data <- zoo(ens$value, order.by = ens$date)
  smoothed_esm <- rollapply(zoo_data, width = window_size,
                            FUN = mean,
                            align = "center",
                            fill = "extend")
  smooth_esm <- data.frame(date = time(smoothed_esm),
                             values = coredata(smoothed_esm))
  assign(paste0("smooth_esm_", ssp),
         smooth_esm,
         envir = globalenv())

# Individual models
  timeseries_proj <- readRDS(paste0("__data/timeseries/sst_year_proj_ind_", ssp, ".RDS"))
  timeseries_allmodels <- do.call(rbind, timeseries_proj)
  allmodels <- timeseries_allmodels
  emplist <- list()
  ens2 <- allmodels

  for (i in 1:length(unique(allmodels$model))) {

    ens <- subset(ens2, model == unique(ens2$model)[i])
    zoo_data <- zoo(ens$value, order.by = ens$date)
```

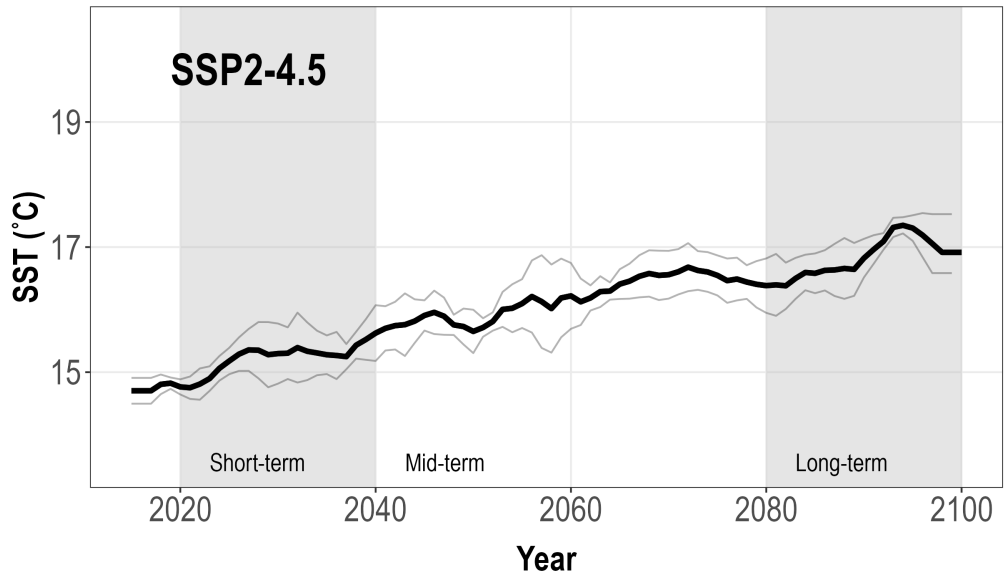
```
    smoothed_esm <- rollapply(zoo_data, width = window_size,
                              FUN = mean, align = "center", fill = "extend")
    smooth_11_esm <- data.frame(date = time(smoothed_esm),
                                values = coredata(smoothed_esm))
    smooth_11_esm$model <- unique(ens$model)
    emplist[[i]] <- smooth_11_esm
  }

  ssp_smoothed_ind <- do.call(rbind, emplist)
  assign(paste0("ssp_smoothed_ind_", ssp),
         ssp_smoothed_ind,
         envir = globalenv())

}

tic(); smooth_esms("ssp245", window_size = 5); toc()
smooth_esms("ssp585", window_size = 5)
```

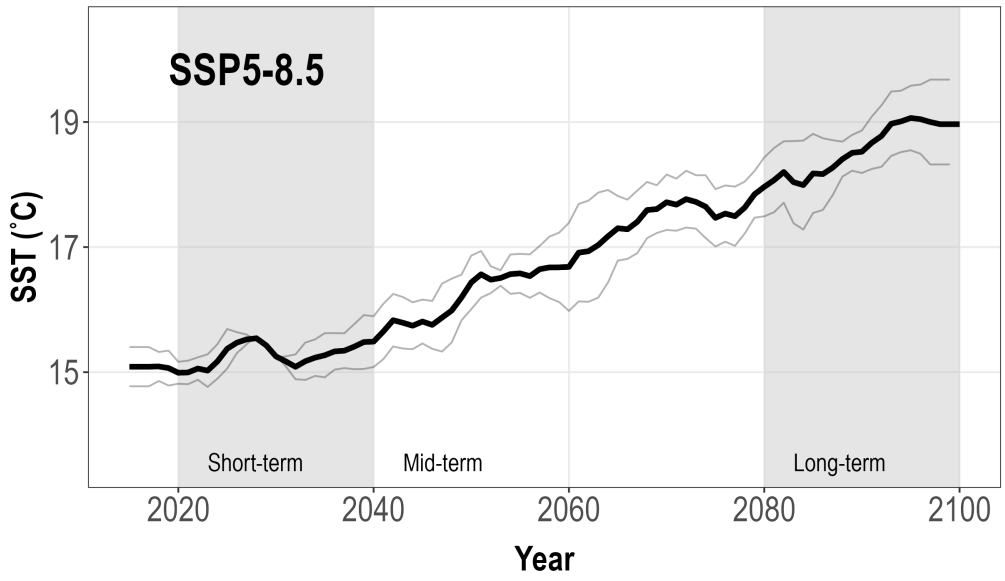
7.1.3 Plot!

```
plot_ts <- function(ssp, sspletter) {

  smooth_esm <- get(paste0("smooth_esm_", ssp)) #relies on this being in global env
  ssp_smoothed_ind <- get(paste0("ssp_smoothed_ind_", ssp))

  # Make a kick-ass plot
 p1 <- ggplot() +
    geom_line(smooth_esm,
              mapping = aes(x = date, y = values),
              lwd = 1.5) +
  geom_rect(data = data.frame(),
            mapping = aes(xmin = 2020, xmax = 2040, ymin = -Inf, ymax = Inf),
             fill = "grey",
            alpha = 0.4) +
```

```
  geom_rect(data = data.frame(),
            mapping = aes(xmin = 2080, xmax = 2100, ymin = -Inf, ymax = Inf),
            fill = "grey",
            alpha = 0.4) +
  geom_line(smooth_esm,
            mapping = aes(x = date, y = values),
            lwd = 1.5) +
geom_line(subset(ssp_smoothed_ind, model == "ACCESS-CM2"),
             mapping = aes(x = date, y = values),
          col = "black",
          alpha = 0.3) +
   geom_line(subset(ssp_smoothed_ind, model == "IPSL-CM6A-LR"),
             mapping = aes(x = date, y = values),
             col = "black",
             alpha = 0.3) +
   theme_bw() +
   scale_x_continuous(name = "Year",
                      n.breaks = 6) +
   scale_y_continuous(name = "SST (°C)",
                      limits = c(13.5, 20.5)) +
   theme(panel.grid.minor = element_blank(),
         plot.margin=unit(c(1,0.1,.1,0.1),"cm"),
         axis.title = element_text(size = 20,
                                   family = "Arial Narrow",
                                   face = "bold"),
         axis.text = element_text(size = 20,
                                  family = "Arial Narrow"),
         axis.title.x = element_text(margin = margin(t = 10, r = -20))) +
  annotate("text", x = 2019, y = 20.1,
           label = sspletter,
           size = 9,
           fontface = "bold",
           family = "Arial Narrow",
           hjust = 0,
           vjust = 1) +
  annotate("text", x = 2083, y = 13.7,
           label = "Long-term", size = 5,
           family = "Arial Narrow",
           hjust = 0,
           vjust = 1) +
  annotate("text", x = 2023, y = 13.7,
           label = "Short-term", size = 5,
```

```
             family = "Arial Narrow",
             hjust = 0,
             vjust = 1)

     ggsave(p1,
            filename = paste0(outdir, "/",
                              ssp, "_SST_timeseries_1995-2100_11yrsmooth.png"),
            width = 8, height = 5)

}

# Run function
tic(); plot_ts("ssp245", sspletter = "SSP2-4.5"); toc() #Jessie: 0.197 seconds
plot_ts("ssp585", sspletter = "SSP5-8.5")
```





7.2 Projections

7.2.1 Individual ESM projections

```
models = c("ACCESS-CM2", "IPSL-CM6A-LR")
ssps = c("ssp245", "ssp585")
term = c("near", "long")
outdir = paste0(pth, "/__data/projections")
indir_proj <- paste0(pth, bc_pth, bc_pth_bc)

termdf <- data.frame(timeperiod = c("near", "long"),
                     st = c("2020-01-01", "2080-01-01"),
                     fin = c("2040-01-01", "2100-01-01"))

tic(); for (k in term) {
  for (i in ssps) {
```

```
    for (j in models) {

      allfiles_proj <- list.files(indir_proj, pattern = j, full.names = T)
      allfiles_proj <- allfiles_proj[grep(i, allfiles_proj)]
      allfiles_proj <- allfiles_proj[grep("2100", allfiles_proj)]
      rr <- rast(allfiles_proj)

      # subset to time period
      tp <- subset(termdf, timeperiod == k)
      rr <- rr[[time(rr) > tp[,"st"] & time(rr) < tp[,"fin"] ]]

      # Mean and SD
      proj_u <- mean(rr)
      proj_sd <- stdev(rr)
      # write to outdir
      filename_u <- paste0(outdir, "/ind/mean_", j, "_", i, "_", k, "_", "proj.nc" )
      filename_sd <- paste0(outdir, "/ind/sd_", j, "_", i, "_", k, "_", "proj.nc" )
      terra::writeCDF(proj_u, filename_u, overwrite = T)
      terra::writeCDF(proj_sd, filename_sd, overwrite = T)
    }
  }
}; toc() #Jessie: 1.8 seconds
```

```
list.files(paste0(pth, "/__data/projections/ind"))
```

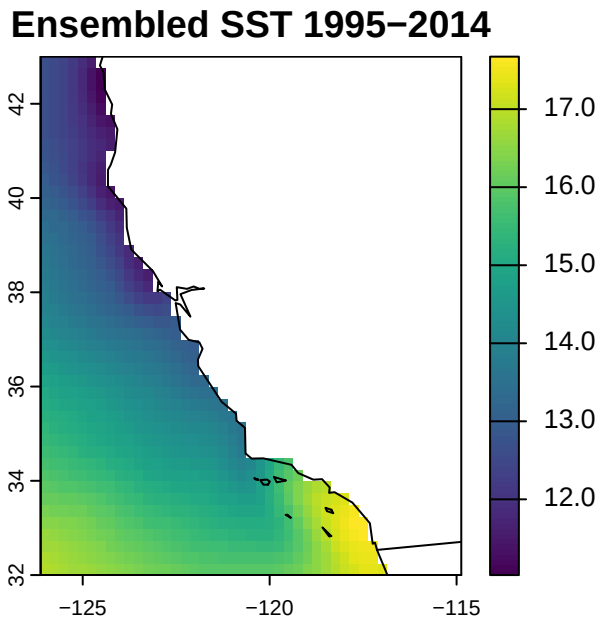
7.2.2 Ensembled projections

```
tic(); for (k in term) {
  for (i in ssps) {


      allfiles_proj <- list.files(paste0(pth, "/__data/projections/ind"),
                                  pattern = i, full.names = T)
      allfiles_proj <- allfiles_proj[grep(k, allfiles_proj)]
      allfiles_proj <- allfiles_proj[grep("mean", allfiles_proj)]
      rr <- rast(allfiles_proj)

      # Mean and SD
      proj_u <- mean(rr)
      proj_sd <- stdev(rr)

      # write to outdir
      filename_u <- paste0(outdir, "/ens/mean_ens_", i, "_", k, "_", "proj.nc" )
      filename_sd <- paste0(outdir, "/ens/sd_ens_", i, "_", k, "_", "proj.nc" )
      terra::writeCDF(proj_u, filename_u, overwrite = T)
      terra::writeCDF(proj_sd, filename_sd, overwrite = T)

  }
}; toc() #Jessie: 0.3 seconds
```

```
list.files(paste0(pth, "/__data/projections/ens"))
```

7.2.3 Delta difference

```
#Ensemble average of 1995-2014 for both models
bc_pth <- paste0(pth, "/__data/bias_correct/esm/_4_bias_corrected/")
r1 <- rast(paste0(bc_pth, "tos_mo_ACCESS-CM2_1995-2014_bc_historical_remapped.nc"))
r2 <- rast(paste0(bc_pth, "tos_mo_IPSL-CM6A-LR_1995-2014_bc_historical_remapped.nc"))
rr <- c(r1, r2)
mean_hist <- mean(rr)
plot(mean_hist, main = "Ensembled SST 1995-2014"); maps::map("world", add = T)
```

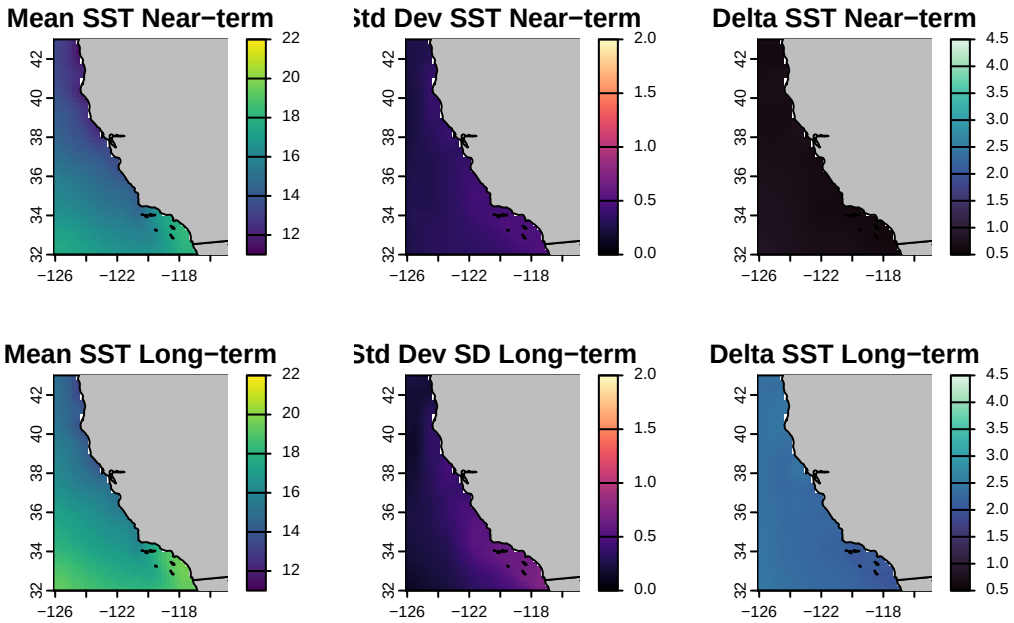



```
ssps = c("ssp245", "ssp585")

tic(); for (i in ssps) {
```

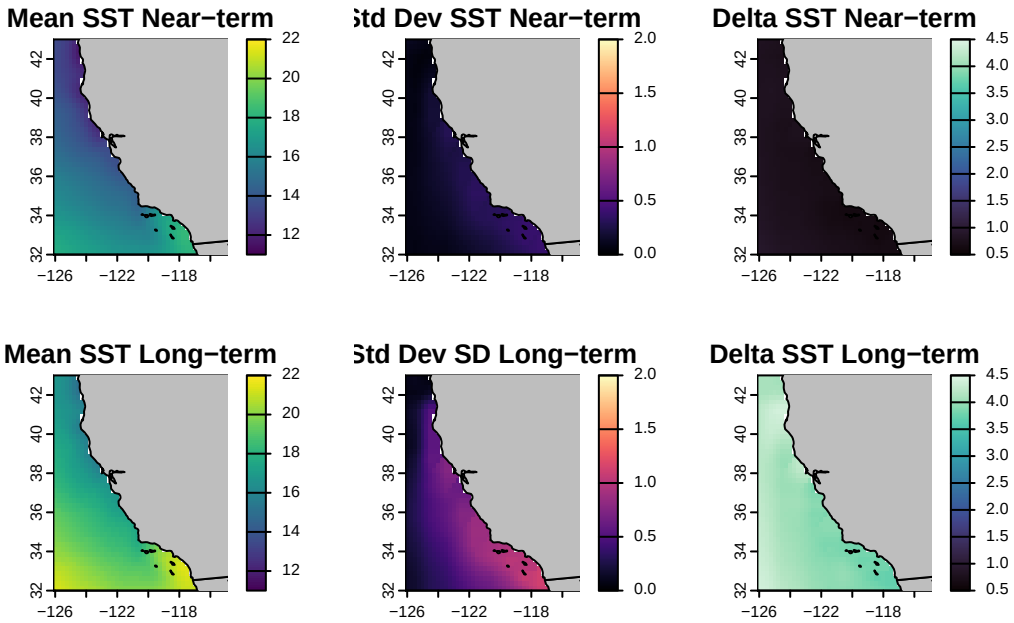
```
  near_mean <- rast(paste0(pth, "/__data/projections/ens/mean_ens_", i, "_near_proj.nc"))
  rr <- near_mean - mean_hist
  writeCDF(rr, paste0(outdir, "/delta/delta_mean_ens_near_", i, ".nc"),
           overwrite = T)

  long_mean <- rast(paste0(pth, "/__data/projections/ens/mean_ens_", i, "_long_proj.nc"))
  rr <- long_mean - mean_hist
  writeCDF(rr, paste0(outdir, "/delta/delta_mean_ens_long_", i, ".nc"),
           overwrite = T)

}; toc()
```

```
list.files(paste0(pth, "/__data/projections/delta"))
```

7.2.4 Plot: SSP2-4.5



7.2.5 Plot: SSP5-8.5



7.3 Uncertainty

8 Handy resources

- CDO reference card
- `hotrstuff`

- `climate4R`
- explainer explainer
- news article

References

https://doi.org/10.1071/ES19040

https://doi.org/10.1029/2019MS002010

https://doi.org/10.1111/gcb.16371

https://doi.org/10.1093/icesjms/fsv250

https://doi.org/10.1093/icesjms/fsab100

https://doi.org/10.1093/icesjms/fsaa103

https://doi.org/10.1016/j.tree.2023.04.005

https://doi.org/10.1016/j.pocean.2010.09.001

https://doi.org/10.1029/2000JD900719

https://doi.org/10.1038/s41467-019-09519-w