ESMValTool recipes

ESMValTool workshop, 30-31 May 2023

Structure of an ESMValTool recipe

A recipe includes four sections:

- documentation
- datasets
- preprocessors
- diagnostics

See overview of recipes from ESMValTool documentation.

1. Documentation

The esmvaltool/config-references.yml file contains the list of ESMValTool diagnostic and recipe authors, references and projects.

The documentation section includes:

- title
- description
- authors
- maintainer
- references
- projects

Example:
recipe_ocean_amoc.yml

```yaml
# ESMValTool
# recipe_ocean_amoc.yml
---
documentation:
  title: Atlantic Meridional Overturning Circulation Recipe

  description: |
    Recipe to produce time series figures of the derived variable, the
    Atlantic meriodinal overturning circulation (AMOC).
    This recipe also produces transect figures of the stream functions for
    the years 2001-2004.

  authors:
    - demora_lee

  maintainer:
    - demora_lee

  references:
    - demora2018gmd

  projects:
    - ukesm
```

2. Datasets

Dataset dictionaries define standardized data specifications via key-value pairs:

- dataset name
- project
- experiment key
- ensemble member
- sub-experiment id
- time range
- model grid (CMIP6 models)

Example: recipe_esacci_lst.yml

```yaml
# Recipe to call ESA CCI LST diagnostic.
---
documentation:
  title: ESA CCI LST diagnostic
  description: |
    Please add description here
  authors:
    - king_robert

  maintainer:
    - king_robert

  references:
    - esacci_lst

  projects:
    - cmug

datasets:
  - {dataset: CESM2, project: CMIP6, exp: historical, ensemble: r(2:3)i1p1f1,
     start_year: 2004, end_year: 2005, grid: gn}
  - {dataset: UKESM1-0-LL, project: CMIP6, exp: historical,
     ensemble: r(1:2)i1p1f2, start_year: 2004, end_year: 2005, grid: gn}
  - {dataset: ESACCI-LST, project: OBS, type: sat, tier: 2,
     start_year: 2004, end_year: 2005, version: '1.00'}
```

2. Datasets – continued

Datasets can also be specified inside a diagnostics section using the keyword

additional_datasets

Example: recipe_correlation.yml

```yaml
diagnostics:
  analyses:
    description: |
      Pearsons r correlation coefficient with respect to a reference dataset.
      Note that a mean over the time coordinate is taken before computing the
      correlation, because there is no preprocessor function to regrid the
      time coordinate.
    themes:
      - phys
    realms:
      - atmos
    variables:
      ta:
        preprocessor: preprocess_3d_data
        reference_dataset: ERA-Interim
        start_year: 2000
        end_year: 2002
        project: CMIP5
        mip: Amon
        exp: historical
        ensemble: r1i1p1
        additional_datasets:
          # One or more datasets can be added here
          - {dataset: bcc-csm1-1}
          # The reference dataset is required
          - {dataset: ERA-Interim, project: OBS6, tier: 3, type: reanaly, version: 1}
```

3. Preprocessors

https://docs.esmvaltool.org/projects/ESMValCore/en/latest/recipe/preprocessor.html

- ESMValCore provides preprocessing procedures that are common for many types of analysis. Not all preprocessing steps are required.
- Preprocessing follows a default order of procedures. The order can be changed by the user by setting the `custom_order` flag in the recipe.

[1] Variable derivation	[6] Area masking	[11] Time manipulation	[16] Detrend
[2] CMORization and dataset-specific fixes	[7] Mask by values	[12] Area manipulation	[17] Rolling window statistics
[3] Supplementary variables	[8] Horizontal regridding	[13] Volume manipulation	[18] Unit conversion
[4] Vertical interpolation	[9] Ensemble statistics	[14] Cycles	[19] Bias
[5] land-sea weighting	[10] Multi-model statistics	[15] Trend	[20] Clip data

(1) Variable derivation

- Derive variables which are not in the CMIP standard data request using standard variables as input.
- Requires a name definition and corresponding CMOR table.

```
# recipe_ocean_amoc.yml
diag_timeseries_amoc:
    description: atlantic_meridional_overturning_circulation
    variables:
        amoc:
            mip: Omon
            derive: true
            force_derivation: false
```

(2) CMORization and dataset-specific fixes

Data checking

- Data preprocessed by ESMValCore is automatically checked against its CMOR definition.
  - Requested coordinates are present and comply with their definition.
  - Correctness of variable names, units and other metadata.
  - Compliance with the valid minimum and maximum values allowed if defined.

Dataset specific fixes

Some datasets have specific fixes, applied in three steps:
(1) `fix_file`; (2) `fix_metadata`; (3) `fix_data`

(2) CMORization and dataset-specific fixes

Example: Fix chlorophyll data in the ESACCI_OC dataset.

```
# esacci_oc.py
def _fix_data(cube, var):
    """Specific data fixes for different variables."""
    logger.info("Fixing data ...")
    with constant_metadata(cube):
        if var == 'chl':
            cube *= 1.e-06
    return cube
```

(3) Supplementary variables

Supplementary variables are added automatically in preprocessor functions when needed.

If automatic selection does not give desired result, supplementary variables may be added explicitly.

Preprocessor	**Variable short name**	**Variable standard name**
`area_statistics`	areacella, areacello	cell_area
`mask_landsea`	sftlf, sftof	land_area_fraction, sea_area_fraction
`mask_landseaice`	sftgif	land_ice_area_fraction
`volume_statistics`	volcello	ocean_volume
`weighting_landsea_fractio`	sftlf, sftof	land_area_fraction, sea_area_fraction

(4) Vertical interpolation

extract_levels:

levels: {numeric levels, named levels, from dataset}

scheme: {linear, linear_extrapolate, nearest, nearest_extrapolate}

coordinate: override default z-axis coordinate

```
preprocessors:
    preproc_select_levels_from_list:
        extract_levels:
            levels: [100000., 50000., 3000., 1000.]
            scheme: linear
```

(5) Land/sea fraction weighting

This function multiplies the given input field by a fraction in the range [0,1] to account for the fact that not all grid points are completely land- or sea-covered.

weighting_landsea_fraction:
    area_type: {land, sea}
    exclude: [<named dataset>,
        'reference_dataset',
        'alternative_dataset']

Example: recipe_wenzel16nat.yml

```yaml
highlat_gpp:
  custom_order: true
  weighting_landsea_fraction:
    area_type: land
  extract_region: &extract_region
    start_longitude: 0.
    end_longitude: 360.
    start_latitude: 60.
    end_latitude: 90.
  area_statistics:
    operator: sum
  annual_statistics:
    operator: mean
```

(6) Area masking

Where possible, the masking is realized using the standard mask files provided together with the model data as part of the CMIP data request (ancillary variable). In the absence of these files, the Natural Earth masks are used.

mask_landsea:
    mask_out: {land, sea}

mask_landseaice:
    mask_out: {landsea, ice}

mask_glaciated:
    mask_out: glaciated

Example: recipe_collins13ipcc.yml

```
preproc_map_land:
  mask_landsea:
    mask_out: sea
  mask_landseaice:
    mask_out: ice
  regrid:
    target_grid: 1x1
    scheme: linear
```

(7) Mask by values

- mask_fillvalues: combine missing values masks from individual models into a multi-model missing values mask
- mask_multimodel: create a common mask for multiple datasets with common coordinates
- mask_above_threshold; mask_below_threshold; mask_inside_range; mask_outside range

```
missing_values_preprocessor:
    mask_fillvalues:
        threshold_fraction: 0.95
        min_value: 19.0
        time_window: 10.0
```

(8) Horizontal regridding

The use of the horizontal regridding functionality is flexible depending on what type of reference grid and what interpolation scheme is preferred.

regird:

target_grid: {reference dataset, regular grid, regional target}

scheme: {linear, linear_extrapolate, nearest, area_weighted,
unstrictured_nearest}

(8) Horizontal regridding

Regular grid specification

```
regrid_preprocessor:
    regrid:
        target_grid: 1.0x1.0
        Scheme: nearest
```

Reference dataset grid

```
regrid_preprocessor:
    regrid:
        target_grid: ERA-Interim
        Scheme: linear
```

(9) Ensemble statistics

Compute ensemble statistics for models with many ensemble members.

Example: [recipe_preprocessor_test.yml]

```yaml
# Calculate ensemble means, then multi-model mean
preprocessor_6:
  regrid:
    target_grid: 3x3
    scheme: linear
  ensemble_statistics:
    statistics: [mean]
    exclude: [GFDL-ESM2G]
  multi_model_statistics:
    span: overlap
    statistics: [mean]
    keep_input_datasets: false
    exclude: [GFDL-ESM2G]
```

```
example_preprocessor:
    ensemple_statistics:
        statistics: [mean, median]
```

(10) Multi-model statistics

multi_model_statistics:

span: { overlap, full } - use overlapping times only, or full datasets

statistics: { mean, median, max, min, std_dev, pXX.YY } - pXX.YY=percentiles

groupby: { list of dataset keys , tag }

exclude: { list of datasets }

Example:
[recipe_ocean_example.yml]

```
preprocessors:
  # ------------------------------------------------------
  # Time series preprocessors
  # ------------------------------------------------------
  prep_timeseries_1: # For 2D fields
    custom_order: true
    area_statistics:
      operator: mean
    multi_model_statistics:
      span: overlap
      statistics: [mean ]
```

(11) Time manipulation

- extract_time; extract_season; extract_month:
  Extract a time range, season or month from a cube.
- hourly_statistics; daily_statistics; monthly_statistics; seasonal_statistics; annual_statistics; decadal_statistics; climate_statistics:
  Compute statistics for fixed period of time (climate=full time range)
- resample_time; resample_hours: Resample data
- anomalies: Compute (standardized) anomalies
- regrid_time: Aligns the time axis of each dataset to have common time points and calendars.
- timeseries_filter: Allows application of a filter to the time-series data.

(12) Area manipulation

- extract_coordinate_points: Extract a point with arbitrary coordinates given an interpolation scheme.
- extract_region: Extract a region from a cube based on lat/lon corners.
- extract_named_regions: Extract a specific region from in the region coordinate.
- extract_shape: Extract a region defined by a shapefile.
- extract_point: Extract a single point (with interpolation)
- extract_location: Extract a single point by its location (with interpolation)
- zonal_statistics: Compute zonal statistics.
- meridional_statistics: Compute meridional statistics.
- area_statistics: Compute area statistics.

(13) Volume manipulation

- axis_statistics: Perform operations along a given axis.
- extract_volume: Extract a specific depth range from a cube.
- volume_statistics: Calculate the volume-weighted average.
- depth_integration: Integrate over the depth dimension.
- extract_transect: Extract data along a line of constant latitude or longitude.
- extract_trajectory: Extract data along a specified trajectory.

(14) Cycles

Extract the peak-to-peak amplitude (maximum value minus minimum value) of a field aggregated over specified coordinates.

amplitude:

coords: {year, month, day_of_year, …}

Example: recipe_wenzel16nat.yml

- Note re-use of extract_point parameters.

```yaml
highlat_co2:
  custom_order: true
  extract_point: &extract_point
    latitude: 71.323
    longitude: 203.389
    scheme: nearest
  annual_statistics:
    operator: mean

highlat_amp:
  custom_order: true
  extract_point: *extract_point
  amplitude:
    coords: year
```

(15) Trend

Calculate the linear trend, and/or standard error of the linear trend, of a dataset (defined as slope of an ordinary linear regression) along a specified coordinate.

- linear_trend:
  coordinate: {time}
- linear_trend_stderr:
  coordinate: {time}

(16) Detrend

ESMValCore supports detrending along any dimension using the preprocessor function ‘detrend’.

detrend:
    dimension: {time}
    method: {linear, constant}

Example: recipe_climwip_brunner20esd.yml

```yaml
detrended_std:
  custom_order: true
  <<: *general
  annual_statistics:
    operator: mean
  detrend:
    dimension: time
    method: linear
  climate_statistics:
    operator: std_dev
```

(17) Rolling window statistics

rolling_window_statistics:

coordinate: { time }

operator: { mean, median, std_dev, min, max, sum }

window_length: { size of rolling window }

Example from documentation:

Calculate two-day rolling precipitation sum for daily precipitation data.

```yaml
preprocessors:
  preproc_rolling_window:
    coordinate: time
    operator: sum
    window_length: 2
```

(18) Unit conversion

Different datasets might have different units, for example when comparing CMIP5 and CMIP6 variables where the units have changed or in case of observational datasets that are delivered in different units.

Example: recipe_esacci_oc.yml

```yaml
preprocessors:

  prep_chl:
    custom_order: true
    extract_levels:
      levels: 0.
      scheme: nearest_extrapolate
    climate_statistics:
      operator: mean
    regrid:
      target_grid: 2x2
      scheme: linear
    convert_units:
      units: mg m-3
    mask_above_threshold:
      threshold: 1.5
```

(19) Bias

Calculates biases with respect to a given reference dataset. All input datasets need to have identical dimensional coordinates.

For dataset, include reference_for_bias: true

bias:

bias_type: { absolute, relative }
denominator_mask_threshold: { float, default = 1e-3 }
keep_reference_dataset: { True, [False] }
exclude: { list of datasets }

```
datasets:
   - {dataset: CanESM2, project: CMIP6, ensemble: r1i1p1f1,
grid: gn}
   - {dataset: CESM2,   project: CMIP6, ensemble: r1i1p1f1,
grid: gn}
   - {dataset: ERA-Interim, project: OBS6, tier: 3, type:
reanaly, version: 1, reference_for_bias: true}

preprocessors:
   preproc_bias:
      bias:
         bias_type: relative
         denominator_mask_threshold: 1e-8
         keep_reference_dataset: true
         exclude: [CanESM2]
```

(20) Clip data

Clip data values to a certain minimum, maximum or range.

clip:
    minimum: {numerical value, null}
    maximum: {numerical value, null}

```
example_preprocessor:
    clip:
        minimum: 0
        maximum: null
```

4. Diagnostics

The diagnostics section includes one or more diagnostics. Each diagnostic section will include:

- The variable(s) to preprocess, including the preprocessor to be applied to each variable
- The diagnostic script(s) to be run
- A description of the diagnostic and lists of themes and realms that it applies to
- An optional additional_datasets section
- An optional title and description, used to generate the title and description in the index.html output file.

Preproc + diagnostic

Example from
recipe_ocean_multimap.yml

A diagnostic does not need to invoke additional scripts;
- scripts: null

See e.g.
recipe_concatenate_exps.yml

```yaml
# ------------------------------------------------
# Preprocessors
# ------------------------------------------------
preprocessors:
  # ------------------------------------------------
  # map preprocessors
  # ------------------------------------------------
  # For a 2D global surface map
  prep_surface_map_2D:
    climate_statistics:
      operator: mean
    regrid:
      target_grid: 1x1
      scheme: linear

# ------------------------------------------------
# Diagnostics
# ------------------------------------------------
diagnostics:

  # ------------------------------------------------
  # Map diagnostics vs OBS
  # ------------------------------------------------
  diag_surface_multimap:
    description: Global Ocean Surface maps vs OBS
    variables:
      fgco2:
        preprocessor: prep_surface_map_2D
        mip: Omon
        maps_range: [-0.12, 0.12]
        diff_range: [-0.09, 0.09]
        layout_rowcol: [4, 4]
    additional_datasets:
      - {dataset: Landschuetzer2016,  project: OBS,  type: clim,  version: v2016,
start_year: 1995,  end_year: 2014,  tier: 2}
    scripts:
      Global_Ocean_multi_vs_obs:
        script: ocean/diagnostic_maps_multimodel.py
        observational_dataset: {dataset: Landschuetzer2016, project: OBS}
        <<: *write_opts
```

Revise standard recipes

Revising standard recipes can usually be done fairly easily for changes that involve model datasets or preprocessor configurations. Replacing obsdata or changing diagnostic scripts may require some work.

- Updating datasets from CMIP5 to CMIP6 requires including model grid specification.
- Some variables may not be available for all CMIP models.
- Diagnostic scripts do not follow a standardized format.

See e.g. noresmvaltool tested recipes for recipes using NorESM model datasets.