Earth Diagnostics Documentation

Release 3.0.0b48

BSC-CNS Earth Sciences Department

Apr 03, 2017

CONTENTS

1 Tutorial **1**
- 1.1 Installation . . . 1
- 1.2 Creating a config file . . . 1
- 1.3 Prepare the run script . . . 2

2 Configuration file options **3**
- 2.1 DIAGNOSTICS . . . 3
  - 2.1.1 Mandatory configurations . . . 3
  - 2.1.2 Optional configurations . . . 3
- 2.2 EXPERIMENT . . . 4
- 2.3 CMOR . . . 4
  - 2.3.1 Cmorization options . . . 4
  - 2.3.2 Metadata options . . . 5
- 2.4 THREDDS . . . 6
- 2.5 ALIAS . . . 6

3 Diagnostic list **7**
- 3.1 General . . . 7
  - 3.1.1 att . . . 7
  - 3.1.2 dailymean . . . 7
  - 3.1.3 monmean . . . 8
  - 3.1.4 relink . . . 8
  - 3.1.5 relinkall . . . 8
  - 3.1.6 rewrite: . . . 9
  - 3.1.7 scale . . . 9
  - 3.1.8 yearlymean . . . 9
- 3.2 Ocean . . . 10
  - 3.2.1 areamoc . . . 10
  - 3.2.2 averagesection . . . 10
  - 3.2.3 convectionsites . . . 10
  - 3.2.4 cutsection . . . 11
  - 3.2.5 gyres . . . 11
  - 3.2.6 heatcontent . . . 11
  - 3.2.7 heatcontentlayer . . . 11
  - 3.2.8 interpolate . . . 12
  - 3.2.9 interpolateCDO . . . 12
  - 3.2.10 maxmoc . . . 12
  - 3.2.11 mixedlayerheatcontent . . . 13
  - 3.2.12 mixedlayersaltcontent . . . 13
  - 3.2.13 moc . . . 13
  - 3.2.14 mxl . . . 13

3.2.15 psi . . . 14
3.2.16 regmean . . . 14
3.2.17 rotate . . . 14
3.2.18 siasiesiv . . . 15
3.2.19 verticalmean . . . 15
3.2.20 verticalmeanmeters . . . 15
3.3 Statistics . . . 15
3.3.1 climpercent . . . 15
3.3.2 monpercent . . . 16

4 Tips and tricks **17**
4.1 Working with ORCA1 . . . 17
4.2 Configuring core usage . . . 17
4.3 NEMO files . . . 17

5 What to do if you have an error **19**

6 Developer's guide **21**
6.1 Developing a diagnostic . . . 21

7 Frequently Asked Questions **23**

8 Module documentation **25**
8.1 earthdiagnostics . . . 25
8.1.1 earthdiagnostics.box . . . 25
8.1.2 earthdiagnostics.cdftools . . . 26
8.1.3 earthdiagnostics.cmorizer . . . 26
8.1.4 earthdiagnostics.cmormanager . . . 26
8.1.5 earthdiagnostics.config . . . 29
8.1.6 earthdiagnostics.constants . . . 30
8.1.7 earthdiagnostics.datamanager . . . 34
8.1.8 earthdiagnostics.diagnostic . . . 37
8.1.9 earthdiagnostics.earthdiags . . . 38
8.1.10 earthdiagnostics.parser . . . 39
8.1.11 earthdiagnostics.utils . . . 39
8.1.12 earthdiagnostics.variable . . . 42
8.2 earthdiagnostics.general . . . 42
8.2.1 earthdiagnostics.general.attribute . . . 42
8.2.2 earthdiagnostics.general.dailymean . . . 43
8.2.3 earthdiagnostics.general.monthlymean . . . 44
8.2.4 earthdiagnostics.general.relink . . . 44
8.2.5 earthdiagnostics.general.relinkall . . . 45
8.2.6 earthdiagnostics.general.rewrite . . . 46
8.2.7 earthdiagnostics.general.scale . . . 46
8.2.8 earthdiagnostics.general.yearlymean . . . 47
8.3 earthdiagnostics.ocean . . . 48
8.3.1 earthdiagnostics.ocean.areamoc . . . 48
8.3.2 earthdiagnostics.ocean.averagesection . . . 48
8.3.3 earthdiagnostics.ocean.convectionsites . . . 49
8.3.4 earthdiagnostics.ocean.cutsection . . . 50
8.3.5 earthdiagnostics.ocean.gyres . . . 51
8.3.6 earthdiagnostics.ocean.heatcontent . . . 51
8.3.7 earthdiagnostics.ocean.heatcontentlayer . . . 52
8.3.8 earthdiagnostics.ocean.interpolate . . . 53
8.3.9 earthdiagnostics.ocean.interpolatecdo . . . 54

8.3.10 earthdiagnostics.ocean.maxmoc . . . 54
8.3.11 earthdiagnostics.ocean.mixedlayerheatcontent . . . 55
8.3.12 earthdiagnostics.ocean.mixedlayersaltcontent . . . 56
8.3.13 earthdiagnostics.ocean.moc . . . 57
8.3.14 earthdiagnostics.ocean.mxl . . . 57
8.3.15 earthdiagnostics.ocean.psi . . . 58
8.3.16 earthdiagnostics.ocean.rotation . . . 58
8.3.17 earthdiagnostics.ocean.siasiesiv . . . 59
8.3.18 earthdiagnostics.ocean.verticalmean . . . 60
8.3.19 earthdiagnostics.ocean.verticalmeanmeters . . . 60
8.4 earthdiagnostics.statistics . . . 61
8.4.1 earthdiagnostics.statistics.climatologicalpercentile . . . 61
8.4.2 earthdiagnostics.statistics.monthlypercentile . . . 62

Python Module Index **63**

Index **65**

CHAPTER

ONE

TUTORIAL

So, you are planning to use the Earth Diagnostics? You don't know how to use them? This is the place to go. From now on this tutorial will guide you through all the process from installation to running.

Hint: If you have any problem with this tutorial, please report it to <javier.vegas@bsc.es> so it can be corrected. A lof of people will benefit from it.

1.1 Installation

If you have access to the BSC-ES machines, you don't need to install it. Just use the available module:

In case that you need a custom installation for development or can not use the BSC-ES machines, install it from BSC-ES GitLab repository:

```
pip install git+https://earth.bsc.es/gitlab/es/ocean_diagnostics.git
```

You will also need

- CDO version 1.7.2 (other versions could work, but this is the one we use)
- NCO version 4.5.4 or newer
- Python 2.7 or newer (but no 3.x) with bscearth.utils, CDO and NCO packages, among others.
- Access to CDFTOOLS_3.0 executables for BSC-ES. The source code is available on Github (https://github.com/jvegasbsc/CDFTOOLS) and it can be compiled with CMake

1.2 Creating a config file

Go to the folder where you installed the EarthDiagnostics. You will see a folder called earthdiagnostics, and, inside it, a diags.conf file that can be used as a model for your config file. Create a copy of it wherever it suites you.

Now open your brand new copy with your preferred text editor. The file contains commentaries explaining each one of its options, so read it carefully and edit whatever you need. Don't worry about DIAGS option, we will talk about it next.

After this, you need to choose the diagnostics you want to run. For a simple test, it's recommended to use the monmean diagnostic to compute monthly means from daily data. We recommend it because it can be used with any variable, the user has to provide parameters but they are quite intuitive and it's relatively fast to compute. If your experiment does not have daily data, you can use any other diagnostic. Check next section for a list of available diagnostics and choose whichever suits you better. From now on, we will assume that you are going to run the monmean diagnostic.

Hint: For old Ocean Diagnostics users: you can use most of the old names as aliases to launch one or multiple diagnostics. Check the ALIAS section on the diags.conf to see which ones are available.

First, choose a variable that has daily data. Then replace the DIAGS option with the next one where $VARIABLE represents the variable's name and $DOMAIN its domain (atmos, ocean, seaice, landice...)

```
DIAGS = monmean,$VARIABLE,$DOMAIN
```

1.3 Prepare the run script

Once you have configured your experiment you can execute any diagnostic with the provided launch_diags.sh script. Create a copy and change the variables PATH_TO_CONF_FILE and PATH_TO_DIAGNOSTICS so they point to your conf file and installation folder.

Now, execute the script (or submit it to bsceslogin01, it has the correct header) and... that's it! You will find your results directly on the storage and a folder for the temp files in the scratch named after the EXPID.

CHAPTER

TWO

CONFIGURATION FILE OPTIONS

This section contains the list and explanation about all the options that are available on the configuration file. Use it as a reference while preparing your configuration file. Each subsection will refer to the matching section from the config file. Those subsections explanation may be divided itself for the shake of clarity but this further divisions have nothing to do with the config file syntax itself.

2.1 DIAGNOSTICS

This section contains the general configuration for the diagnostics. The explanation has been divided in two subsections: the first one will cover all the mandatory options that you must specify in every configuration, while the second will cover all the optional configurations.

2.1.1 Mandatory configurations

- **SCRATCH_DIR:** Temporary folder for the calculations. Final results will never be stored here.
- **DATA_DIR:** ':' separated list of folders to look for data in. It will look for file in the path $DATA_FOLDER/$EXPID and $DATA_FOLDER/$DATA_TYPE/$MODEL/$EXPID
- **CON_FILES:** Folder containing mask and mesh files for the dataset.
- **FREQUENCY:** Default data frequency to be used by the diagnostics. Some diagnostics can override this configuration or even ignore it completely.
- **DIAGS:** List of diagnostic to run, in the order you want them to run

2.1.2 Optional configurations

- **SCRATCH_MASKS** Common scratch folder for the ocean masks. This is useful to avoid replicating them for each run at the fat nodes. By default is '/scratch/Earth/ocean_masks'
- **RESTORE_MESHES** By default, Earth Diagnostics only copies the mask files if they are not present in the scratch folder. If this option is set to true, Earth Diagnostics will copy them regardless of existence. Default is False.
- **DATA_ADAPTOR** This is used to choose the mechanism for storing and retrieving data. Options are CMOR (for our own experiments) or THREDDS (for anything else). Default value is CMOR
- **DATA_TYPE** Type of the dataset to use. It can be exp, obs or recon. Default is exp.
- **DATA_CONVENTION** Convention to use for file paths and names and variable naming among other things. Can be SPECS, PRIMAVERA or CMIP6. Default is SPECS.

- **CDFTOOLS_PATH** Path to the folder containing CDFTOOLS executables. By default is empty, so CDFTOOLS binaries must be added to the system path.
- **MAX_CORES** Maximum number of cores to use. By default the diagnostics will use all cores available to them. It is not necessary when launching through a scheduler, as Earthdiagnostics can detect how many cores the scheduler has allocated to it.

2.2 EXPERIMENT

This sections contains options related to the experiment's definition or configuration.

- **MODEL** Name of the model used for the experiment.
- **MODEL_VERSION** Model version. Used to get the correct mask and mesh files
- **ATMOS_TIMESTEP** Time between outputs from the atmosphere. This is not the model simulation timestep! Default is 6.
- **OCEAN_TIMESTEP** Time between outputs from the ocean. This is not the model simulation timestep! Default is 6.
- **ATMOS_GRID** Atmospheric grid definition. Will be used as a default target for interpolation diagnostics.
- **INSTITUTE** Institute that made the experiment, observation or reconstruction
- **EXPID** Unique identifier for the experiment
- **NAME** Experiment's name. By default it is the EXPID.
- **STARTDATES** Startdates to run as a space separated list
- **MEMBER** Members to run as a space separated list. You can just provide the number or also add the prefix
- **MEMBER_DIGITS** Number of minimum digits to compose the member name. By default it is 1. For example, for member 1 member name will be fc1 if MEMBER_DIGITS is 1 or fc01 if MEMBER_DIGITS is 2
- **MEMBER_PREFIX** Prefix to use for the member names. By default is 'fc'
- **MEMBER_COUNT_START** Number corresponding to the first member. For example, if your first member is 'fc1', it should be 1. If it is 'fc0', it should be 0. By default is 0
- **CHUNK_SIZE** Length of the chunks in months
- **CHUNKS** Number of chunks to run
- **CALENDAR** Calendar to use for date calculation. All calendars supported by Autosubmit are available. Default is 'standard'

2.3 CMOR

In this section, you can control how will work the cmorization process. All options belonging to this section are optional.

2.3.1 Cmorization options

This options control when and which varibales will be cmorized.

- **FORCE** If True, launches the cmorization, regardless of existence of the extracted files or the package containing the online-cmorized ones. If False, only the non-present chunks will be cmorized. Default value is False
- **FORCE_UNTAR** Unpacks the online-cmorized files regardless of exstience of extracted files. If 'FORCE is True, this parameter has no effect. If False, only the non-present chunks will be unpacked. Default value is False.
- **FILTER_FILES** Only cmorize original files containing any of the given strings. This is a space separated list. Default is the empty string.
- **OCEAN_FILES** Boolean flag to activate or no NEMO files cmorization. Default is True.
- **ATMOSPHERE_FILES** Boolean flag to activate or no IFS files cmorization. Default is True.
- **USE_GRIB** Boolean flag to activate or no GRIB files cmorization for the atmosphere. If activated and no GRIB files are present, it will cmorize using the MMA files instead (as if it was set to False). Default is True.
- **CHUNKS** Space separated list of chunks to be cmorized. If not provided, all chunks are cmorized
- **VARIABLE_LIST** Space separated list of variables to cmorize. Variables must be specified as domain:var_name. If no one is specified, all the variables will be cmorized

Grib variables extraction

These three options ares used to configure the variables to be CMORized from the grib atmospheric files. They must be specified using the IFS code in a list separated by comma.

You can also specify the levels to extract using one of the the following syntaxes:

- VARIABLE_CODE
- VARIABLE_CODE:LEVEL,
- VARIABLE_CODE:LEVEL_1-LEVEL_2-...-LEVEL_N
- VARIABLE_CODE:MIN_LEVEL:MAX_LEVEL:STEP

Some examples to clarify it further: * Variable with code 129 at level 30000: 129:30000 * Variable with code 129 at levels 30000, 40000 and 60000: 129:30000-40000-60000 * Variable with code 129 at levels between 30000 and 600000 with 10000 intervals:

129:30000:60000:10000 equivalent to 129:30000-40000-50000-60000

- **ATMOS_HOURLY_VARS** Configuration of variables to be extracted in an hourly basis
- **ATMOS_DAILY_VARS** Configuration of variables to be extracted in a daily basis
- **ATMOS_MONTHLY_VARS** Configuration of variables to be extracted in a monthly basis

2.3.2 Metadata options

All the options in this subsection will serve just to add the given values to the homonymous attributes in the cmorized files.

- **ASSOCIATED_EXPERIMENT** Default value is 'to be filled'
- **ASSOCIATED_MODEL** Default value is 'to be filled'
- **INITIALIZATION_DESCRIPTION** Default value is 'to be filled'
- **INITIALIZATION_METHOD** Default value is '1'

- **PHYSICS_DESCRIPTION** Default value is 'to be filled'
- **PHYSICS_VERSION** Default value is '1'
- **SOURCE** Default value is 'to be filled'

2.4 THREDDS

For now, there is only one option for the THREDDS server configuration.

- **SERVER_URL** THREDDS server URL

2.5 ALIAS

This config file section is different from all the others because it does not contain a set of configurations. Instead, in this section the user can define a set of aliases to be able to launch its most used configurations with ease. To do this, the user must add an option with named after the desired alias and assign to it the configuration or configurations to launch when this ALIAS is invoked. See the next example:

```
ALIAS_NAME = diag,opt1,opt2 diag,opt1new,opt2
```

In this case, the user has defined a new alias 'ALIAS' that can be used launch two times the diagnostic 'diag', the first with the options 'opt1' and 'opt2' and the second replacing 'opt1' with 'opt1new'.

In this example, configuring the DIAGS as

```
DIAGS = ALIAS_NAME
```

will be identical to

```
DIAGS = diag,opt1,opt2 diag,opt1new,opt2
```

coding=utf-8

CHAPTER

THREE

DIAGNOSTIC LIST

In this section you have a list of the available diagnostics, with a small description of each one and a link to the full documentation. To see what options are available for each diagnostic, see generate_jobs documentation.

Remember that diagnostics are specified separated by spaces while options are given separated by commas:

```
DIAGS = diag1 diag2,option1,option2 diag3
```

3.1 General

The diagnostics from this section are of general use and can be used with any variable you may have. Most of them are meant to help you to solve usual issues that you may have with the data: incorrect metadata, scaled up or down variables, links missing. This section also contains the diagnostic used to calculate the monthly means.

3.1.1 att

Writes a global attributte to all the netCDF files for a given variable. See *Attribute*

Options:

1. **Variable:** Variable name
2. **Domain:** Variable domain
3. **Attributte name:** Attributte to write
4. **Attribute value:** Atrribute's new value. Replace ',' with '&;' and ' ' with '&.' to avoid parsing errors when processing the diags
5. **Grid = '':** Variable grid. Only required in case that you want to use interpolated data.

3.1.2 dailymean

Calculates the daily mean for a given variable. See *DailyMean*

> **Warning:** This diagnostic does not use the frequency configuration from the config file. You must specify the original frequency when calling it.

Options:

1. **Variable:** Variable name
2. **Domain:** Variable domain
3. **Original frequency:** Original frequency to use
4. **Grid = '':** Variable grid. Only required in case that you want to use interpolated data.

3.1.3 monmean

Calculates the monthly mean for a given variable. See *MonthlyMean*

> **Warning:** This diagnostic does not use the frequency configuration from the config file. You must specify the original frequency when calling it. Otherwise, it will always try to use daily data.

Options:

1. **Variable:** Variable name
2. **Domain:** Variable domain
3. **Original frequency = daily:** Original frequency to use
4. **Grid = '':** Variable grid. Only required in case that you want to use interpolated data.

3.1.4 relink

Regenerates the links created in the monthly_mean, daily_mean, etc folders for a given varible. See *Relink*

Options:

1. **Variable:** Variable name
2. **Domain:** Variable domain
3. **Move old =** True: If True, any data founded in the target directory will be moved to another folder (called FOLDER_NAME_old) instead of deleted.
4. **Grid = '':** Variable grid. Only required in case that you want to use interpolated data.

3.1.5 relinkall

Regenerates the links created in the monthly_mean, daily_mean, etc folders for all variables See *RelinkAll*

Options:

This diagnostic has no options

3.1.6 rewrite:

Just rewrites the CMOR output of a given variable. Useful to correct metadata or variable units. See *Rewrite*

Options:

1. **Variable:** Variable name
2. **Domain:** Variable domain
3. **Grid = '':** Variable grid. Only required in case that you want to use interpolated data.

3.1.7 scale

Scales a given variable using a given scale factor and offset (NEW_VALUE = OLD_VALUE * scale + offset). Useful to correct errors on the data.

See *Scale*

Options:

1. **Variable:** Variable name
2. **Domain:** Variable domain
3. **Scale value:** Scale factor for the variable
4. **Offset value:** Value to add to the original value after scaling
5. **Grid = '':** Variable grid. Only required in case that you want to use interpolated data.
6. **Min limit = NaN:** If there is any value below this threshold, scale will not be applied
7. **Max limit = NaN:** If there is any value above this threshold, scale will not be applied
8. **Frequencies = [*Default_frequency*]:** List of frequencies ('-' separated) to apply the scale on. Default is the frequency defined globally for all the diagnostics

3.1.8 yearlymean

Calculates the daily mean for a given variable. See *YearlyMean*

Warning: This diagnostic does not use the frequency configuration from the config file. You must specify the original frequency when calling it.

Options:

1. **Variable:** Variable name
2. **Domain:** Variable domain
3. **Original frequency:** Original frequency to use
4. **Grid = '':** Variable grid. Only required in case that you want to use interpolated data.

3.2 Ocean

The diagnostics from this section are meant to be used with NEMO variables. Some of them will compute new variables while others just calculate means or sections for variables in the ORCA grid. The interpolation diagnostics are also included here as they are usually used with variables in the ORCA grid.

3.2.1 areamoc

Compute an Atlantic MOC index by averaging the meridional overturning in a latitude band between 1km and 2km or any other index averaging the meridional overturning in a given basin and a given domain. See *AreaMoc*

> **Warning:** The MOC for the given basin must be calculated previously. Usually, it will suffice to call the 'moc' diagnostic earlier in the DIAGS list.

Options:

1. **Min latitude:** Minimum latitude to compute
2. **Max latitude:** Maximum latitude to compute
3. **Min depth:** Minimum depth (in levels)
4. **Max depth:** Maximum depth (in levels)
5. **Basin = 'Global':** Basin to calculate the diagnostic on.

3.2.2 averagesection

Compute an average of a given zone. The variable MUST be in a regular grid See *AverageSection*

Options:

1. **Variable:** Variable to average
2. **Min longitude:** Minimum longitude to compute
3. **Max longitude:** Maximum longitude to compute
4. **Min latitude:** Minimum latitude to compute
5. **Max latitude:** Maximum latitude to compute
6. **Domain = ocean:** Variable domain

3.2.3 convectionsites

Compute the intensity of convection in the four main convection sites. See *ConvectionSites*

Options:

This diagnostic has no options

3.2.4 cutsection

Cuts a meridional or zonal section. See *CutSection*

Options:

1. **Variable:** Variable to cut the section on
2. **Zonal:** If True, calculates a zonal section. If False, it will be a meridional one
3. **Value:** Reference value for the section
4. **Domain = ocean:** Variable's domain

3.2.5 gyres

Compute the intensity of the subtropical and subpolar gyres. See *Gyres*

Options:

This diagnostic has no options

3.2.6 heatcontent

Compute the total and mean ocean heat content. See *HeatContent*

Options:

1. **Basin** Basin to calculate the heat content one
2. **Mixed layer:** If 1, reduces the compuation to the mixed layer. If -1, excludes the mixed layer from the computations. If 0, no effect.
3. **Min depth:** Minimum depth for the calculation in levels. If 0, whole depth is used
4. **Max depth:** Maximum depth for the calculation in levels

3.2.7 heatcontentlayer

Point-wise Ocean Heat Content in a specified ocean thickness. See *HeatContentLayer*

Options:

3. **Min depth:** Minimum depth for the calculation in meteres
4. **Max depth:** Maximum depth for the calculation in meters
5. **Basin = 'Global':** Basin to calculate the heat content on.

options_available = (DiagnosticIntOption('min_depth'), DiagnosticIntOption('max_depth'), DiagnosticBasinOption('basin', Basins.Global))

3.2.8 interpolate

3-dimensional conservative interpolation to the regular atmospheric grid. It can also be used for 2D (i,j) variables. See *Interpolate*

Warning: This interpolation requires the pre-generated weights that can be found in '/es-nas/autosubmit/con_files/weights'. Make sure that they are available for your configuration.

Options:

1. **Target grid:** New grid for the data
2. **Variable:** Variable to interpolate
3. **Domain = ocean:** Variable's domain
4. **Invert latitude:** If True, inverts the latitude in the output file.
5. **Original grid = '':** Source grid to choose. By default this is the original data, but sometimes you will want to use another (for example, the 'rotated' one produced by the rotation diagnostic)

3.2.9 interpolateCDO

Bilinear interpolation to a given grid using CDO. See *InterpolateCDO*

Warning: This interpolation is non-conservative, so treat its output with care. It has the advantage that does not require the pre-generated weights so it can be used when the 'interp' diagnostic is not available.

Options:

1. **Variable:** variable to interpolate
2. **Target grid:** Variable domain
3. **Domain = ocean:** Variable's domain
4. **Mask oceans = True:** If True, replaces the values in the ocean by NaN. You must only set it to false if, for some reason, you are interpolating an atmospheric or land variable that is stored in the NEMO grid (yes, this can happen, i.e. with tas).
5. **Original grid = '':** Source grid to choose. By default this is the original data, but sometimes you will want to use another (for example, the 'rotated' one produced by the rotation diagnostic)

3.2.10 maxmoc

Compute an Atlantic MOC index by finding the maximum of the annual mean meridional overturning in a latitude / depth region. Output from this diagnostic will be always in yearly frequency. See *MaxMoc*

Warning: The MOC for the given basin must be calculated previously. Usually, it will suffice to call the 'moc' diagnostic earlier in the DIAGS list.

Warning: This diagnostic can only be computed for full years. It will discard incomplete years and only compute the index in those with the full 12 months available.

Options:

1. **Min latitude:** Minimum latitude to compute
2. **Max latitude:** Maximum latitude to compute
3. **Min depth:** Minimum depth (in levels)
4. **Max depth:** Maximum depth (in levels)
5. **Basin = 'Global':** Basin to calculate the diagnostic on.

3.2.11 mixedlayerheatcontent

Compute mixed layer heat content. See *MixedLayerHeatContent*

Options:

This diagnostic has no options

3.2.12 mixedlayersaltcontent

Compute mixed layer salt content. See *MixedLayerSaltContent*

Options:

This diagnostic has no options

3.2.13 moc

Compute the MOC for oceanic basins. Required for 'areamoc' and 'maxmoc' See *Moc*

Options:

This diagnostic has no options

3.2.14 mxl

Compute the mixed layer depth. See *Mxl*

Options:

This diagnostic has no options

3.2.15 psi

Compute the barotropic stream function. See *Psi*

Options:

This diagnostic has no options

3.2.16 regmean

Computes the mean value of the field (3D, weighted). For 3D fields, a horizontal mean for each level is also given. If a spatial window is specified, the mean value is computed only in this window. See `RegionMean`

Options:

1. **Domain:** Variable domain
2. **Variable:** Variable to average
3. **Grid_point:** NEMO grid point used to store the variable: T, U, V ...
4. **Basin = Global:** Basin to compute
5. **Save 3d = True:** If True, it also stores the average per level
6. **Min depth:** Minimum depth to compute in levels. If -1, average from the surface
7. **Max depth:** Maximum depth to compute in levels. If -1, average to the bottom
8. **Variance = False:** If True, it also stores the variance
9. **Original grid = '':** Source grid to choose. By default this is the original data, but sometimes you will want to use another (for example, the 'rotated' one produced by the rotation diagnostic)

3.2.17 rotate

Rotates the given variables See *Rotation*

Options:

1. **Variable u:** Variable's u component
2. **Variable v:** Variable's u component
3. **Domain = ocean:** Variable domain:
4. **Executable = /home/Earth/jvegas/pyCharm/cfutools/interpolation/rotateUVorca:** Path to the executable that will compute the rotation

Warning: This default executable has been compiled for ORCA1 experiments. For other resolutions you must use other executables compiled ad-hoc for them

3.2.18 siasiesiv

Compute the sea ice extent , area and volume in both hemispheres or a specified region. See *Siasiesiv*

Options:

1. **Basin = 'Global':** Basin to restrict the computation to.

3.2.19 verticalmean

Chooses vertical level in ocean, or vertically averages between 2 or more ocean levels. See *VerticalMean*

Options:

1. **Variable:** Variable to average
2. **Min depth = -1:** Minimum level to compute. If -1, average from the surface
3. **Max depth:** Maximum level to compute. If -1, average to the bottom

3.2.20 verticalmeanmeters

Averages vertically any given variable. See *VerticalMeanMeters*

Options:

1. **Variable:** Variable to average
2. **Min depth = -1:** Minimum depth to compute in meters. If -1, average from the surface
3. **Max depth:** Maximum depth to compute in meters. If -1, average to the bottom

3.3 Statistics

3.3.1 climpercent

Calculates the specified climatological percentile of a given variable. See *ClimatologicalPercentile*

Options:

1. **Domain:** Variable's domain
2. **Variable:** Variable to compute diagnostic on
3. **Leadtimes:** Leadtimes to compute
4. **Bins:** Number of bins to use to discretize the variable

3.3.2 monpercent

Calculates the specified monthly percentile of a given variable. See *MonthlyPercentile*

Options:

1. **Domain:** Variable's domain
2. **Variable:** Variable to compute diagnostic on
3. **Percentiles:** List of requested percentiles ('-' separated)

CHAPTER

FOUR

TIPS AND TRICKS

4.1 Working with ORCA1

If you plan to run diagnostics for ORCA1 resolution, be aware that your workstation will be more than capable to run them. At this resolution, memory and CPU consumption is low enough to allow you keep using the machine while running, specially if you reserve a pair of cores for other uses.

4.2 Configuring core usage

By default, the Earth Diagnostics creates a thread for each available core for the execution. If you are using a queueing system, the diagnostics will always use the number of cores that you reserved. If you are running outside a queueing system, the diagnostics will try to use all the cores on the machine. To avoid this, add the MAX_CORES parameter to the DIAGNOSTICS section inside the diags.conf file that you are using.

4.3 NEMO files

Unlike the bash version of the ocean diagnostics, this program keeps the NEMO files in the scratch folder so you can launch different configurations for the same experiment with reduced start time. You will need to remove the experiment's folder in the scratch directory at the end of the experiment to avoid wasting resources. To do this, just use

```
earthdiags -f PATH_TO_CONF --clean
```

If you plan to run the earthdiagnostics only once, you can add this line after the execution

CHAPTER

FIVE

WHAT TO DO IF YOU HAVE AN ERROR

Sometimes, the diagnostics may crash and you will not know why. This section will give you a procedure to follow before reporting the issue. This procedure is intended to solve some common problems or, at least, to help you in creating good issue reports. Remember: a good issue report reduces the time required to solve it!

Hint: Please, read carefully the error message. Most times the error message will point you to the problem's source and sometimes even give you a hint of how to solve it by yourself. And if this it not the case or if you find it obscure, even if it was helpful, please contact the developers so it can be improved in further versions

Try this simple steps BEFORE reporting an issue

- Clean scratch folder
- Update to the latest compatible tag: maybe your issue is already solved in it
- If you get the error for the first chunk of a given diagnostic, change the number of chunks to 1
- Call the diags with the -lc DEBUG -log log.txt options

Now, you have two options: if everything is fine, the error was probably due to some corrupted files or some unstable machine state. Nevertheless, try running the diagnostic with -lc DEBUG -log log.txt for all the chunks. If everything it's fine that's all.

If you experienced the same problem again, go to the GitLab portal and look into the open issues (https://earth.bsc.es/gitlab/es/ocean_diagnostics/issues). If you find your issue or a very similar one, use it to report your problems. If you can not find an open one that suites your problem, create a new one and explain what is happening to you. In any case, it will be very useful if you can attach your diags.conf and log.txt files and specify the machine you were using.

After that, it's just a matter of waiting for the developers to do their work and answering the questions that they may have. Please, be patient.

Caution: Of course, there is a third option: you keep experiencing an error that appears randomly on some executions but you are not able to reproduce it in a consistent manner. Report it and attach as much logs and configuration files as you have, along with the date and time of the errors.

CHAPTER

SIX

DEVELOPER'S GUIDE

The tool provides a set of useful diagnostics, but a lot more can be required at anytime. If you miss something and are able to develop it, you are more than welcome to collaborate. Even if you can not develop it, please let us know what do you want.

The first step is to go to the GitLab page for the project (https://earth.bsc.es/gitlab/es/ocean_diagnostics/) and open a new issue. Be sure that the title is self-explicative and give a detailed description of what you want. Please, be very explicit about what you want to avoid misunderstandings.

Hint: If reading your description, you think that you are taking the developers as stupids, you are doing it perfectly.

Don't forget to add the relevant tags. At this stage you will have to choose between 'enhancement', if you are proposing an improvement on a currently available feature, or 'new feature' in any the other case.

Now, if you are thinking on developing it yourself, please refer to the BSC-ES Git strategy (wiki_link_when_available) If you have any doubts, or just want help to start the development, contact javier.vegas@bsc.es.

6.1 Developing a diagnostic

For new diagnostics development, we have some advice to give:

- Do not worry about performance at first, just create a version that works. Developers can help you to optimize it later.
- There is nothing wrong with doing some common preparations in the generate_jobs of the diagnostic.
- Parallelization is achieved by running multiple diagnostics at a time. You don't need to implement it at diagnostic level
- Use the smallest time frame for your diagnostic: if you can work at chunk level, do not ask for full year data.
- Prefer NCO over CDO, you will have less problems when versions change.
- Ask for help as soon as you get stuck.
- Use always the methods in Utils instead of writing your own code.
- Use meaningful variable names. If you are using short names just to write less, please switch to an editor with autocompletion!
- Do not modify the mesh and mask files, another diagnostic can be using them at the same time.

CHAPTER

SEVEN

FREQUENTLY ASKED QUESTIONS

Here will be the answers to the most usual questions. For the moment, there is nothing to see here...

CHAPTER

EIGHT

MODULE DOCUMENTATION

8.1 earthdiagnostics

8.1.1 earthdiagnostics.box

class earthdiagnostics.box.**Box**(*depth_in_meters=False*)

Bases: object

Represents a box in the 3D space. Also allows easy conversion from the coordinate values to significant string representations

depth_in_meters = None

If True, treats the depth as if it is given in meters. If False, as it is given in levels :rtype: bool

get_depth_str()

Gets a string representation of depth. For depth expressed in meters, it adds th character 'm' to the end If min_depth is different from max_depth, it concatenates the two values :return: string representation for depth :rtype: str

get_lat_str()

Gets a string representation of the latitude in the format XX{N/S}. If min_lat is different from max_lat, it concatenates the two values :return: string representation for latitude :rtype: str

get_lon_str()

Gets a string representation of the longitude in the format XX{E/W}. If min_lon is different from max_lon, it concatenates the two values :return: string representation for longitude :rtype: str

max_depth = None

Maximum depth :rtype: float

max_lat

Maximum latitude :rtype: float

max_lon

Maximum longitude :rtype: float

min_depth = None

Minimum depth :rtype: float

min_lat

Minimum latitude :rtype: float

min_lon

Minimum longitude :rtype: float

8.1.2 earthdiagnostics.cdftools

class earthdiagnostics.cdftools.**CDFTools**(*path=''*)
Bases: object

Class to run CDFTools executables

Parameters **path** (*str*) – path to CDFTOOLS binaries

run(*command*, *input*, *output=None*, *options=None*, *log_level=20*, *input_option=None*)
Runs one of the CDFTools

Parameters

- **command** (*str | iterable*) – executable to run
- **input** (*str*) – input file
- **output** – output file. Not all tools support this parameter
- **options** (*str | list[str] | Tuple[str]*) – options for the tool.
- **log_level** (*int*) – log level at which the output of the cdftool command will be added

8.1.3 earthdiagnostics.cmorizer

class earthdiagnostics.cmorizer.**Cmorizer**(*data_manager*, *startdate*, *member*)
Bases: object

Class to manage CMORization

Parameters

- **data_manager** (CMORManager) – experiment's data manager
- **startdate** (*str*) – startdate to cmorize
- **member** (*int*) – member to cmorize

cmorize_atmos()
CMORizes atmospheric data, from grib or MMA files :return:

cmorize_ocean()
CMORizes ocean files from MMO files :return:

extract_variable(*file_path*, *handler*, *frequency*, *variable*)
Extracts a variable from a file and creates the CMOR file

Parameters

- **file_path** (*str*) – path to the file
- **handler** (*netCDF4.Dataset*) – netCDF4 handler for the file
- **frequency** (*Frequency*) – variable's frequency
- **variable** (*str*) – variable's name

8.1.4 earthdiagnostics.cmormanager

class earthdiagnostics.cmormanager.**CMORManager**(*config*)
Bases: *earthdiagnostics.datamanager.DataManager*

Data manager class for CMORized experiments

get_file(*domain*, *var*, *startdate*, *member*, *chunk*, *grid=None*, *box=None*, *frequency=None*, *vartype=1*)

Copies a given file from the CMOR repository to the scratch folder and returns the path to the scratch's copy

Parameters

- **domain** (*Domain*) – CMOR domain
- **var** (*str*) – variable name
- **startdate** (*str*) – file's startdate
- **member** (*int*) – file's member
- **chunk** (*int*) – file's chunk
- **grid** (*str|NoneType*) – file's grid (only needed if it is not the original)
- **box** (Box) – file's box (only needed to retrieve sections or averages)
- **frequency** (*Frequency|NoneType*) – file's frequency (only needed if it is different from the default)
- **vartype** (*VariableType*) – Variable type (mean, statistic)

Returns path to the copy created on the scratch folder

Return type str

get_file_path(*startdate*, *member*, *domain*, *var*, *cmor_var*, *chunk*, *frequency*, *grid=None*, *year=None*, *date_str=None*)

Returns the path to a concrete file :param startdate: file's startdate :type startdate: str :param member: file's member :type member: int :param domain: file's domain :type domain: Domain :param var: file's var :type var: var :param chunk: file's chunk :type chunk: int :param frequency: file's frequency :type frequency: Frequency :param grid: file's grid :type grid: str|NoneType :param year: file's year :type year: int|str|NoneType :param date_str: date string to add directly. Overrides year or chunk configurations :type date_str: str|NoneType :return: path to the file :rtype: str|NoneType

get_year(*domain*, *var*, *startdate*, *member*, *year*, *grid=None*, *box=None*)

Ge a file containing all the data for one year for one variable :param domain: variable's domain :type domain: str :param var: variable's name :type var: str :param startdate: startdate to retrieve :type startdate: str :param member: member to retrieve :type member: int :param year: year to retrieve :type year: int :param grid: variable's grid :type grid: str :param box: variable's box :type box: Box :return:

link_file(*domain*, *var*, *cmor_var*, *startdate*, *member*, *chunk=None*, *grid=None*, *frequency=None*, *year=None*, *date_str=None*, *move_old=False*, *vartype=1*)

Creates the link of a given file from the CMOR repository.

Parameters

- **move_old** –
- **date_str** –
- **year** (*int*) – if frequency is yearly, this parameter is used to give the corresponding year
- **domain** (*Domain*) – CMOR domain
- **var** (*str*) – variable name
- **startdate** (*str*) – file's startdate
- **member** (*int*) – file's member
- **chunk** (*int*) – file's chunk

- **grid** (*str*) – file's grid (only needed if it is not the original)
- **frequency** (*Frequency*) – file's frequency (only needed if it is different from the default)
- **vartype** (*VariableType*) – Variable type (mean, statistic)

Returns path to the copy created on the scratch folder

Return type str

prepare()

Prepares the data to be used by the diagnostic.

If CMOR data is not created, it show a warning and closes. In the future, an automatic cmorization procedure will be launched

If CMOR data is available but packed, the procedure will unpack it.

Returns

send_file(*filetosend*, *domain*, *var*, *startdate*, *member*, *chunk=None*, *grid=None*, *region=None*, *box=None*, *rename_var=None*, *frequency=None*, *year=None*, *date_str=None*, *move_old=False*, *diagnostic=None*, *cmorized=False*, *vartype=1*)

Copies a given file to the CMOR repository. It also automatically converts to netCDF 4 if needed and can merge with already existing ones as needed

Parameters

- **move_old** (*bool*) – if true, moves files following older conventions that may be found on the links folder
- **date_str** – exact date_str to use in the cmorized file
- **year** (*int*) – if frequency is yearly, this parameter is used to give the corresponding year
- **rename_var** (*str*) – if exists, the given variable will be renamed to the one given by var
- **filetosend** (*str*) – path to the file to send to the CMOR repository
- **region** (*str*) – specifies the region represented by the file. If it is defined, the data will be appended to the CMOR repository as a new region in the file or will overwrite if region was already present
- **domain** (*Domain*) – CMOR domain
- **var** (*str*) – variable name
- **startdate** (*str*) – file's startdate
- **member** (*int*) – file's member
- **chunk** (*int*) – file's chunk
- **grid** (*str*) – file's grid (only needed if it is not the original)
- **box** (Box) – file's box (only needed to retrieve sections or averages)
- **frequency** (*Frequency*) – file's frequency (only needed if it is different from the default)
- **diagnostic** (Diagnostic) – diagnostic used to generate the file
- **cmorized** (*bool*) – flag to indicate if file was generated in cmorization process
- **vartype** (*VariableType*) – Variable type (mean, statistic)

Type str

8.1.5 earthdiagnostics.config

class earthdiagnostics.config.**Config**(*path*)
Bases: object

Class to read and manage the configuration

Parameters **path** (*str*) – path to the conf file

cdftools_path = None
Path to CDFTOOLS executables

con_files = None
Mask and meshes folder path

data_adaptor = None
Scratch folder path

data_dir = None
Root data folder path

data_type = None
Data type (experiment, observation or reconstruction)

experiment = None
Configuration related to the experiment

Return type *ExperimentConfig*

frequency = None
Default data frequency to be used by the diagnostics

get_commands()
Returns the list of commands after replacing the alias :return: full list of commands :rtype: list(str)

max_cores = None
Maximum number of cores to use

restore_meshes = None
If True, forces the tool to copy all the mesh and mask files for the model, regardless of existence

scratch_dir = None
Scratch folder path

scratch_masks = None
Common scratch folder for masks

class earthdiagnostics.config.**ExperimentConfig**(*parser*)
Bases: object

Encapsulates all chunk related tasks

Parameters **parser** (*Parser*) – parser for the config file

get_chunk_list()
Return a list with all the chunks :return: List containing tuples of startdate, member and chunk :rtype: tuple[str, int, int]

get_full_years(*startdate*)
Returns the list of full years that are in the given startdate :param startdate: startdate to use :type startdate: str :return: list of full years :rtype: list[int]

get_member_list()
Return a list with all the members :return: List containing tuples of startdate and member :rtype: tuple[str, int, int]

get_member_str(*member*)
Returns the member name for a given member number. :param member: member's number :type member: int :return: member's name :rtype: str

get_year_chunks(*startdate*, *year*)
Get the list of chunks containing timesteps from the given year :param startdate: startdate to use :type startdate: str :param year: reference year :type year: int :return: list of chunks containing data from the given year :rtype: list[int]

8.1.6 earthdiagnostics.constants

Contains the enumeration-like classes used by the diagnostics

class earthdiagnostics.constants.**Basin**(*shortname*, *fullname*, *box=None*)
Bases: object

Class representing a given basin

Parameters

- **shortname** (*str*) – sfull basin's name
- **fullname** (*str*) – full basin's name
- **box** (Box) – box defining the basin

box = None
Box representing the basin

fullname
Basin's full name :rtype: str

shortname
Basin's short name :rtype: str

class earthdiagnostics.constants.**Basins**
Bases: object

Predefined basins

Antarctic = <earthdiagnostics.constants.Basin object>
Antarctic Ocean

AntarcticAtlantic = <earthdiagnostics.constants.Basin object>
Antarctic Ocean Atlantic Sector

AntarcticIndian = <earthdiagnostics.constants.Basin object>
Antarctic Ocean Indian Sector

Arctic = <earthdiagnostics.constants.Basin object>
Arctic Ocean

ArcticMarginalSeas = <earthdiagnostics.constants.Basin object>
Arctic Ocean

ArcticNorthAtlantic = <earthdiagnostics.constants.Basin object>
Arctic Ocean North Atlantic

Atlantic = <earthdiagnostics.constants.Basin object>
Atlantic ocean

Baffin = <earthdiagnostics.constants.Basin object>
Baffin

Baffin_Bay = <earthdiagnostics.constants.Basin object>
Baffin_Bay

Baltic_Sea = <earthdiagnostics.constants.Basin object>
Baltic_Sea

BarKara = <earthdiagnostics.constants.Basin object>
BarKara

Barents_Sea = <earthdiagnostics.constants.Basin object>
Barents_Sea

Beaufort_Chukchi_Sea = <earthdiagnostics.constants.Basin object>
Beaufort_Chukchi_Sea

Beaufort_Sea = <earthdiagnostics.constants.Basin object>
Beaufort_Sea

Bellingshausen_Sea = <earthdiagnostics.constants.Basin object>
Bellingshausen_Sea

Bering = <earthdiagnostics.constants.Basin object>
Bering

Bering_Strait = <earthdiagnostics.constants.Basin object>
Bering_Strait

CanArch = <earthdiagnostics.constants.Basin object>
CanArch

Canadian_Waters = <earthdiagnostics.constants.Basin object>
Canadian_Waters

Caspian_Sea = <earthdiagnostics.constants.Basin object>
Caspian_Sea

Central_Arctic = <earthdiagnostics.constants.Basin object>
Central_Arctic

Chukchi_Sea = <earthdiagnostics.constants.Basin object>
Chukchi_Sea

East_Siberian_Sea = <earthdiagnostics.constants.Basin object>
East_Siberian_Sea

Eastern_Central_Arctic = <earthdiagnostics.constants.Basin object>
Eastern_Central_Arctic

Fram_Strait = <earthdiagnostics.constants.Basin object>
Fram_Strait

Global = <earthdiagnostics.constants.Basin object>
Global ocean

Global_Ocean = <earthdiagnostics.constants.Basin object>
Global_Ocean

Greenland_Sea = <earthdiagnostics.constants.Basin object>
Greenland_Sea

Grnland = <earthdiagnostics.constants.Basin object>
Grnland

Hudson = <earthdiagnostics.constants.Basin object>
Hudson

Icelandic_Sea = <earthdiagnostics.constants.Basin object>
Icelandic_Sea

Indian = <earthdiagnostics.constants.Basin object>
Indian Ocean

IndoPacific = <earthdiagnostics.constants.Basin object>
Indo Pacific Ocean

Irminger_Sea = <earthdiagnostics.constants.Basin object>
Irminger_Sea

Kara_Gate_Strait = <earthdiagnostics.constants.Basin object>
Kara_Gate_Strait

Kara_Sea = <earthdiagnostics.constants.Basin object>
Kara_Sea

Labrador_Sea = <earthdiagnostics.constants.Basin object>
Labrador_Sea

Laptev_East_Siberian_Chukchi_Seas = <earthdiagnostics.constants.Basin object>
Laptev_East_Siberian_Chukchi_Seas

Laptev_East_Siberian_Seas = <earthdiagnostics.constants.Basin object>
Laptev_East_Siberian_Seas

Laptev_Sea = <earthdiagnostics.constants.Basin object>
Laptev_Sea

Lincoln_Sea = <earthdiagnostics.constants.Basin object>
Lincoln_Sea

Mediterranean_Sea = <earthdiagnostics.constants.Basin object>
Mediterranean_Sea

Nares_Strait = <earthdiagnostics.constants.Basin object>
Nares_Strait

Nordic_Barents_Seas = <earthdiagnostics.constants.Basin object>
Nordic_Barents_Seas

Nordic_Seas = <earthdiagnostics.constants.Basin object>
Nordic_Seas

NorthAtlantic = <earthdiagnostics.constants.Basin object>
North Atlantic Ocean

NorthPacific = <earthdiagnostics.constants.Basin object>
North Pacific Ocean

NorthWest_Passage = <earthdiagnostics.constants.Basin object>
NorthWest_Passage

North_Atlantic_Arctic = <earthdiagnostics.constants.Basin object>
North_Atlantic_Arctic

North_Hemisphere_Ocean = <earthdiagnostics.constants.Basin object>
North_Hemisphere_Ocean

Norwegian_Sea = <earthdiagnostics.constants.Basin object>
Norwegian_Sea

Okhotsk = <earthdiagnostics.constants.Basin object>
Okhotsk

OpenOcean = <earthdiagnostics.constants.Basin object>
OpenOcean

Pacific = <earthdiagnostics.constants.Basin object>
Pacific Ocean

Ross_Sea = <earthdiagnostics.constants.Basin object>
Ross_Sea

Serreze_Arctic = <earthdiagnostics.constants.Basin object>
Serreze_Arctic

Southern_Hemisphere = <earthdiagnostics.constants.Basin object>
Southern_Hemisphere

StLawr = <earthdiagnostics.constants.Basin object>
StLawr

Subpolar_Gyre = <earthdiagnostics.constants.Basin object>
Subpolar_Gyre

TotalArc = <earthdiagnostics.constants.Basin object>
TotalArc

TropicalAtlantic = <earthdiagnostics.constants.Basin object>
Tropical Atlantic Ocean

TropicalIndian = <earthdiagnostics.constants.Basin object>
Tropical Indian Ocean

TropicalPacific = <earthdiagnostics.constants.Basin object>
Tropical Pacific Ocean

Vilkitsky_Strait = <earthdiagnostics.constants.Basin object>
Vilkitsky_Strait

Weddell_Sea = <earthdiagnostics.constants.Basin object>
Weddell_Sea

Western_Central_Arctic = <earthdiagnostics.constants.Basin object>
Western_Central_Arctic

classmethod parse(*basin*)
Return the basin matching the given name. If the parameter basin is a Basin instance, directly returns the same instance. This bahaviour is intended to facilitate the development of methods that can either accept a name or a Basin instance to characterize the basin.

Parameters **basin** (*str | Basin*) – basin name or basin instance

Returns basin instance corresponding to the basin name

Return type *Basin*

class earthdiagnostics.constants.**Models**
Bases: object

Predefined models

ECEARTH_2_3_O1L42 = 'Ec2.3_O1L42'
EC-Earth 2.3 ORCA1 L42

ECEARTH_3_0_O1L46 = 'Ec3.0_O1L46'
EC-Earth 3 ORCA1 L46

ECEARTH_3_0_O25L46 = 'Ec3.0_O25L46'
EC-Earth 3 ORCA0.25 L46

ECEARTH_3_0_O25L75 = 'Ec3.0_O25L75'
EC-Earth 3 ORCA0.25 L75

ECEARTH_3_1_O25L75 = 'Ec3.1_O25L75'
EC-Earth 3.1 ORCA0.25 L75

ECEARTH_3_2_O1L75 = 'Ec3.2_O1L75'
EC-Earth 3.2 ORCA1 L75

GLORYS2_V1_O25L75 = 'glorys2v1_O25L75'
GLORYS2v1 ORCA0.25 L75

NEMOVAR_O1L42 = 'nemovar_O1L42'
NEMOVAR ORCA1 L42

NEMO_3_2_O1L42 = 'N3.2_O1L42'
NEMO 3.2 ORCA1 L42

NEMO_3_3_O1L46 = 'N3.3_O1L46'
NEMO 3.3 ORCA1 L46

NEMO_3_6_O1L46 = 'N3.6_O1L75'
NEMO 3.6 ORCA1 L75

8.1.7 earthdiagnostics.datamanager

class earthdiagnostics.datamanager.**DataManager**(*config*)
Bases: object

Class to manage the data repositories

Parameters **config** (Config) –

file_exists(*domain*, *var*, *startdate*, *member*, *chunk*, *grid=None*, *box=None*, *frequency=None*, *vartype=1*)
Checks if a given file exists

Parameters

- **domain** (*Domain*) – CMOR domain
- **var** (*str*) – variable name
- **startdate** (*str*) – file's startdate
- **member** (*int*) – file's member
- **chunk** (*int*) – file's chunk
- **grid** (*str*) – file's grid (only needed if it is not the original)

- **box** (*Box*) – file's box (only needed to retrieve sections or averages)
- **frequency** (*Frequency*) – file's frequency (only needed if it is different from the default)
- **vartype** (*VariableType*) – Variable type (mean, statistic)

Returns path to the copy created on the scratch folder

Return type str

get_file(*domain*, *var*, *startdate*, *member*, *chunk*, *grid=None*, *box=None*, *frequency=None*, *vartype=1*)

Copies a given file from the CMOR repository to the scratch folder and returns the path to the scratch's copy

Parameters

- **domain** (*Domain*) – CMOR domain
- **var** (*str*) – variable name
- **startdate** (*str*) – file's startdate
- **member** (*int*) – file's member
- **chunk** (*int*) – file's chunk
- **grid** (*str*) – file's grid (only needed if it is not the original)
- **box** (*Box*) – file's box (only needed to retrieve sections or averages)
- **frequency** (*Frequency*) – file's frequency (only needed if it is different from the default)
- **vartype** (*VariableType*) – Variable type (mean, statistic)

Returns path to the copy created on the scratch folder

Return type str

get_year(*domain*, *var*, *startdate*, *member*, *year*, *grid=None*, *box=None*)

Ge a file containing all the data for one year for one variable :param domain: variable's domain :type domain: Domain :param var: variable's name :type var: str :param startdate: startdate to retrieve :type startdate: str :param member: member to retrieve :type member: int :param year: year to retrieve :type year: int :param grid: variable's grid :type grid: str :param box: variable's box :type box: Box :return:

link_file(*domain*, *var*, *cmor_var*, *startdate*, *member*, *chunk=None*, *grid=None*, *frequency=None*, *year=None*, *date_str=None*, *move_old=False*, *vartype=1*)

Creates the link of a given file from the CMOR repository.

Parameters

- **cmor_var** –
- **move_old** –
- **date_str** –
- **year** (*int*) – if frequency is yearly, this parameter is used to give the corresponding year
- **domain** (*Domain*) – CMOR domain
- **var** (*str*) – variable name
- **startdate** (*str*) – file's startdate
- **member** (*int*) – file's member

- **chunk** (*int*) – file's chunk
- **grid** (*str*) – file's grid (only needed if it is not the original)
- **frequency** (*str*) – file's frequency (only needed if it is different from the default)
- **vartype** (*VariableType*) – Variable type (mean, statistic)

Returns path to the copy created on the scratch folder

Return type str

prepare()
Prepares the data to be used by the diagnostic. :return:

send_file(*filetosend*, *domain*, *var*, *startdate*, *member*, *chunk=None*, *grid=None*, *region=None*, *box=None*, *rename_var=None*, *frequency=None*, *year=None*, *date_str=None*, *move_old=False*, *diagnostic=None*, *cmorized=False*, *vartype=1*)
Copies a given file to the CMOR repository. It also automatically converts to netCDF 4 if needed and can merge with already existing ones as needed

Parameters

- **move_old** (*bool*) – if true, moves files following older conventions that may be found on the links folder
- **date_str** – exact date_str to use in the cmorized file
- **year** (*int*) – if frequency is yearly, this parameter is used to give the corresponding year
- **rename_var** (*str*) – if exists, the given variable will be renamed to the one given by var
- **filetosend** (*str*) – path to the file to send to the CMOR repository
- **region** (*str*) – specifies the region represented by the file. If it is defined, the data will be appended to the CMOR repository as a new region in the file or will overwrite if region was already present
- **domain** (*Domain*) – CMOR domain
- **var** (*str*) – variable name
- **startdate** (*str*) – file's startdate
- **member** (*int*) – file's member
- **chunk** (*int*) – file's chunk
- **grid** (*str*) – file's grid (only needed if it is not the original)
- **box** (Box) – file's box (only needed to retrieve sections or averages)
- **frequency** (*Frequency*) – file's frequency (only needed if it is different from the default)
- **diagnostic** (Diagnostic) – diagnostic used to generate the file
- **cmorized** (*bool*) – flag to indicate if file was generated in cmorization process
- **vartype** (*VariableType*) – Variable type (mean, statistic)

Type str

class earthdiagnostics.datamanager.**NetCDFFile**(*remote_file*, *local_file*, *domain*, *var*, *cmor_var*, *data_convention*, *region*)

Bases: object

Class to manage netCDF file and pr

Parameters

- **remote_file** (*str*) –
- **local_file** (*str*) –
- **domain** (*Domain*) –
- **var** (*str*) –
- **cmor_var** (Variable) –

class earthdiagnostics.datamanager.**UnitConversion**(*source*, *destiny*, *factor*, *offset*)
Bases: object

Class to manage unit conversions

classmethod add_conversion(*conversion*)
Adds a conversion to the dictionary

Parameters **conversion** (UnitConversion) – conversion to add

classmethod get_conversion_factor_offset(*input_units*, *output_units*)
Gets the conversion factor and offset for two units . The conversion has to be done in the following way: converted = original * factor + offset

Parameters

- **input_units** (*str*) – original units
- **output_units** (*str*) – destiny units

Returns factor and offset

Return type [float, float]

classmethod load_conversions()
Load conversions from the configuration file

8.1.8 earthdiagnostics.diagnostic

class earthdiagnostics.diagnostic.**Diagnostic**(*data_manager*)
Bases: object

Base class for the diagnostics. Provides a common interface for them and also has a mechanism that allows diagnostic retrieval by name.

Parameters **data_manager** (DataManager) – data manager that will be used to store and retrieve the necessary data

alias = None
Alias to call the diagnostic. Must be overridden at the derived clases

compute()
Calculates the diagnostic and stores the output

Must be implemented by derived classes

classmethod generate_jobs(*diags*, *options*)
Generate the instances of the diagnostics that will be run by the manager

Must be implemented by derived classes.

Parameters

- **diags** (*Diags*) – diagnostics manager
- **options** (*list[str]*) – list of strings containing the options passed to the diagnostic

Returns

static get_diagnostic(*name*)
Return the class for a diagnostic given its name

Parameters **name** (*str*) – diagnostic alias

Returns the selected Diagnostic class, None if name can not be found

Return type *Diagnostic*

static register()
Register a new diagnostic using the given alias. It must be call using the derived class. :param cls: diagnostic class to register :type cls: Diagnostic

send_file(*filetosend*, *domain*, *var*, *startdate*, *member*, *chunk=None*, *grid=None*, *region=None*, *box=None*, *rename_var=None*, *frequency=None*, *year=None*, *date_str=None*, *move_old=False*, *vartype=1*)

Parameters

- **filetosend** –
- **domain** (*ModelingRealm*) –
- **var** –
- **startdate** –
- **member** –
- **chunk** –
- **grid** –
- **region** –
- **box** –
- **rename_var** –
- **frequency** (*Frequency*) –
- **year** –
- **date_str** –
- **move_old** –
- **vartype** (*VariableType*) – Variable type (mean, statistic)

Returns

8.1.9 earthdiagnostics.earthdiags

class earthdiagnostics.earthdiags.**EarthDiags**(*config_file*)
Bases: object

Launcher class for the diagnostics

Parameters **config_file** (*str*) – path to the configuration file

static parse_args()
Entry point for the Earth Diagnostics. For more detailed documentation, use -h option

run()
Run the diagnostics

8.1.10 earthdiagnostics.parser

8.1.11 earthdiagnostics.utils

class earthdiagnostics.utils.**TempFile**
Bases: object

Class to manage temporal files

autoclean = True
If True, new temporary files are added to the list for future cleaning

static clean()
Removes all temporary files created with Tempfile until now

files = []
List of files to clean automatically

static get(*filename=None*, *clean=None*, *suffix='.nc'*)
Gets a new temporal filename, storing it for automated cleaning

Parameters

- **suffix** –
- **filename** (*str*) – if it is not none, the function will use this filename instead of a random one
- **clean** (*bool*) – if true, stores filename for cleaning

Returns path to the temporal file

Return type str

prefix = 'temp'
Prefix for temporary filenames

scratch_folder = ''
Scratch folder to create temporary files on it

class earthdiagnostics.utils.**Utils**
Bases: object

Container class for miscellaneous utility methods

exception ExecutionError
Bases: exceptions.Exception

Exception to raise when a command execution fails

exception Utils.**UnzipException**
Bases: exceptions.Exception

Excpetion raised when unzip fails

static Utils.**available_cpu_count**()
Number of available virtual or physical CPUs on this systemx

Utils.**cdo = <cdo.Cdo object>**
An instance of Cdo class ready to be used

static Utils.**concat_variables**(*source*, *destiny*, *remove_source=False*)
Add variables from a nc file to another :param source: path to source file :type source: str :param destiny: path to destiny file :type destiny: str :param remove_source: if True, removes source file :type remove_source: bool

static Utils.**convert2netcdf4**(*filetoconvert*)
Checks if a file is in netCDF4 format and converts to netCDF4 if not

Parameters **filetoconvert** (*str*) – file to convert

static Utils.**copy_dimension**(*source*, *destiny*, *dimension*, *must_exist=True*, *new_names=None*)
Copies the given dimension from source to destiny, including dimension variables if present

Parameters

- **new_names** (*dict*) – dictionary containing variables to rename and new name as key-value pairs
- **source** (*netCDF4.Dataset*) – origin file
- **destiny** (*netCDF4.Dataset*) – destiny file
- **dimension** (*str*) – variable to copy
- **must_exist** (*bool*) – if false, does not raise an error uf variable does not exist

Returns

static Utils.**copy_file**(*source*, *destiny*)
Copies a file from source to destiny, creating dirs if necessary

Parameters

- **source** (*str*) – path to source
- **destiny** (*str*) – path to destiny

static Utils.**copy_variable**(*source*, *destiny*, *variable*, *must_exist=True*, *add_dimensions=False*, *new_names=None*)
Copies the given variable from source to destiny

Parameters

- **add_dimensions** (*bool*) – if it's true, dimensions required by the variable will be automatically added to the file. It will also add the dimension variable
- **source** (*netCDF4.Dataset*) – origin file
- **destiny** (*netCDF4.Dataset*) – destiny file
- **variable** (*str*) – variable to copy
- **must_exist** (*bool*) – if false, does not raise an error uf variable does not exist
- **new_names** (*dict*) – dictionary containing variables to rename and new name as key-value pairs

Returns

static Utils.**create_folder_tree**(*path*)
Createas a fodle path will and parent directories if needed. :param path: folder's path :type path: str

static Utils.**execute_shell_command**(*command*, *log_level=10*)
Executes a sheel command :param command: command to execute

Log.info('Detailed time for diagnostic class') :param log_level: log level to use for command output :type log_level: int :return: command output :rtype: list

static Utils.**get_datetime_from_netcdf**(*handler*, *time_variable='time'*)
Gets a datetime array from a netCDF file

Parameters

- **handler** (*netCDF4.Dataset*) – file to read
- **time_variable** (*str*) – variable to read, by default 'time'

Returns Datetime numpy array created from the values stored at the netCDF file

Return type np.array

static Utils.**get_file_hash**(*filepath*)
Returns the MD5 hash for the given filepath :param filepath: path to the file to compute hash on :type filepath:str :return: file's MD5 hash :rtype: str

static Utils.**get_mask**(*basin*)
Returns a numpy array containing the mask for the given basin

Parameters **basin** (Basin) – basin to retrieve

Returns mask

Return type numpy.array

static Utils.**move_file**(*source*, *destiny*)
Moves a file from source to destiny, creating dirs if necessary

Parameters

- **source** (*str*) – path to source
- **destiny** (*str*) – path to destiny

Utils.**nco = <nco.nco.Nco object>**
An instance of Nco class ready to be used

static Utils.**openCdf**(*filepath*, *mode='a'*)
Opens a netCDF file and returns a handler to it

Parameters

- **filepath** (*str*) – path to the file
- **mode** (*str*) – mode to open the file. By default, a (append)

Returns handler to the file

Return type netCDF4.Dataset

static Utils.**remove_file**(*path*)
Removes a file, checking before if its exists

Parameters **path** (*str*) – path to file

static Utils.**rename_variable**(*filepath*, *old_name*, *new_name*, *must_exist=True*, *rename_dimension=False*)
Rename multiple variables from a NetCDF file :param filepath: path to file :type filepath: str :param old_name: variable's name to change :type old_name: str :param new_name: new name :type new_name: str :param must_exist: if True, the function will raise an exception if the variable name does not exist :type

must_exist: bool :param rename_dimension: if True, also rename dimensions with the same name :type rename_dimension: bool

static Utils.**rename_variables**(*filepath*, *dic_names*, *must_exist=True*, *rename_dimension=False*)
Rename multiple variables from a NetCDF file :param filepath: path to file :type filepath: str :param dic_names: dictionary containing old names as keys and new names as values :type dic_names: dict :param must_exist: if True, the function will raise an exception if the variable name does not exist :type must_exist: bool :param rename_dimension: if True, also rename dimensions with the same name :type rename_dimension: bool

static Utils.**setminmax**(*filename*, *variable_list*)
Sets the valid_max and valid_min values to the current max and min values on the file :param filename: path to file :type filename: str :param variable_list: list of variables in which valid_min and valid_max will be set :type variable_list: str | list

static Utils.**untar**(*files*, *destiny_path*)
Untar files to a given destiny :param files: files to unzip :type files: list[Any] | Tuple[Any] :param destiny_path: path to destination folder :type destiny_path: str

static Utils.**unzip**(*files*, *force=False*)
Unzip a list of files :param files: files to unzip :type files: list | str :param force: if True, it will overwrite unzipped files :type force: bool

8.1.12 earthdiagnostics.variable

class earthdiagnostics.variable.**Variable**
Bases: object

Class to characterize a CMOR variable. It also contains the static method to make the match between thje original name and the standard name. Requires data _convetion to be available in cmor_tables to work.

class earthdiagnostics.variable.**VariableAlias**(*alias*)
Bases: object

Class to characterize a CMOR variable. It also contains the static method to make the match between thje original name and the standard name. Requires data _convetion to be available in cmor_tables to work.

8.2 earthdiagnostics.general

8.2.1 earthdiagnostics.general.attribute

class earthdiagnostics.general.attribute.**Attribute**(*data_manager*, *startdate*, *member*, *chunk*, *domain*, *variable*, *grid*, *attributte_name*, *attributte_value*)
Bases: *earthdiagnostics.diagnostic.Diagnostic*

Rewrites files without doing any calculations. Can be useful to convert units or to correct wrong metadata

Original author Javier Vegas-Regidor<javier.vegas@bsc.es>

Created July 2016

Parameters

- **data_manager** (DataManager) – data management object
- **startdate** (*str*) – startdate

- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number
- **variable** (*str*) – variable's name
- **domain** (*ModelingRealm*) – variable's domain

alias = 'att'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – variable, domain, grid

Returns

8.2.2 earthdiagnostics.general.dailymean

class earthdiagnostics.general.dailymean.**DailyMean**(*data_manager*, *startdate*, *member*, *chunk*, *domain*, *variable*, *frequency*, *grid*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Calculates daily mean for a given variable

Original author Javier Vegas-Regidor<javier.vegas@bsc.es>

Created July 2016

Parameters

- **data_manager** (DataManager) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number
- **variable** (*str*) – variable's name
- **domain** (*ModelingRealm*) – variable's domain
- **frequency** (*str*) – original frequency
- **grid** (*str*) – original data grid

alias = 'daymean'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – variable, domain, frequency=day, grid=''

Returns

8.2.3 earthdiagnostics.general.monthlymean

class earthdiagnostics.general.monthlymean.**MonthlyMean**(*data_manager*, *startdate*, *member*, *chunk*, *domain*, *variable*, *frequency*, *grid*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Calculates monthly mean for a given variable

Original author Javier Vegas-Regidor<javier.vegas@bsc.es>

Created July 2016

Parameters

- **data_manager** (DataManager) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number
- **variable** (*str*) – variable's name
- **domain** (*ModelingRealm*) – variable's domain
- **frequency** (*str*) – original frequency
- **grid** (*str*) – original data grid

alias = 'monmean'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – variable, domain, frequency=day, grid=''

Returns

8.2.4 earthdiagnostics.general.relink

class earthdiagnostics.general.relink.**Relink**(*data_manager*, *startdate*, *member*, *chunk*, *domain*, *variable*, *move_old*, *grid*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Recreates the links for the variable specified

Original author Javier Vegas-Regidor<javier.vegas@bsc.es>

Created September 2016

Parameters

- **data_manager** (DataManager) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number
- **variable** (*str*) – variable's name
- **domain** (*ModelingRealm*) – variable's domain
- **move_old** (*bool*) – if true, looks for files following the old convention and moves to avoid collisions

alias = 'relink'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – variable, domain, move_old=False

Returns

8.2.5 earthdiagnostics.general.relinkall

class earthdiagnostics.general.relinkall.**RelinkAll**(*data_manager*, *startdate*)
Bases: *earthdiagnostics.diagnostic.Diagnostic*

Recreates the links for the variable specified

Original author Javier Vegas-Regidor<javier.vegas@bsc.es>

Created September 2016

Parameters

- **data_manager** (DataManager) – data management object
- **startdate** (*str*) – startdate

alias = 'relinkall'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – variable, domain, move_old=False

Returns

8.2.6 earthdiagnostics.general.rewrite

class earthdiagnostics.general.rewrite.**Rewrite**(*data_manager*, *startdate*, *member*, *chunk*, *domain*, *variable*, *grid*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Rewrites files without doing any calculations. Can be useful to convert units or to correct wrong metadata

Original author Javier Vegas-Regidor<javier.vegas@bsc.es>

Created July 2016

Parameters

- **data_manager** (*DataManager*) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number
- **variable** (*str*) – variable's name
- **domain** (*ModelingRealm*) – variable's domain

alias = 'rewrite'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – variable, domain, grid

Returns

8.2.7 earthdiagnostics.general.scale

class earthdiagnostics.general.scale.**Scale**(*data_manager*, *startdate*, *member*, *chunk*, *value*, *offset*, *domain*, *variable*, *grid*, *min_limit*, *max_limit*, *frequency*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Scales a variable by the given value also adding at offset Can be useful to correct units or other known errors (think of a tas file declaring K as units but with the data stored as Celsius)

Original author Javier Vegas-Regidor<javier.vegas@bsc.es>

Created July 2016

Parameters

- **data_manager** (*DataManager*) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int :*) – chunk's number

- **variable** (*str*) – variable's name
- **domain** (*ModelingRealm*) – variable's domain

alias = 'scale'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – variable, domain, grid

Returns

8.2.8 earthdiagnostics.general.yearlymean

class earthdiagnostics.general.yearlymean.**YearlyMean**(*data_manager*, *startdate*, *member*, *chunk*, *domain*, *variable*, *frequency*, *grid*)
Bases: *earthdiagnostics.diagnostic.Diagnostic*

Calculates yearly mean for a given variable

Original author Javier Vegas-Regidor<javier.vegas@bsc.es>

Created July 2016

Parameters

- **data_manager** (DataManager) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number
- **variable** (*str*) – variable's name
- **domain** (*ModelingRealm*) – variable's domain
- **frequency** (*str*) – original frequency
- **grid** (*str*) – original data grid

alias = 'yearmean'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – variable, domain, frequency=day, grid=''

Returns

8.3 earthdiagnostics.ocean

8.3.1 earthdiagnostics.ocean.areamoc

class earthdiagnostics.ocean.areamoc.**AreaMoc**(*data_manager*, *startdate*, *member*, *chunk*, *basin*, *box*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Compute an Atlantic MOC index by averaging the meridional overturning in a latitude band between 1km and 2km or any other index averaging the meridional overturning in a given basin and a given domain

Original author Virginie Guemas <virginie.guemas@bsc.es>

Contributor Javier Vegas-Regidor<javier.vegas@bsc.es>

Created March 2012

Last modified June 2016

Parameters

- **data_manager** (DataManager) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number
- **basin** (Basin) – basin to compute
- **box** (Box) – box to compute

alias = 'mocarea'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – minimum latitude, maximum latitude, minimum depth, maximum depth, basin=Global

Returns

8.3.2 earthdiagnostics.ocean.averagesection

class earthdiagnostics.ocean.averagesection.**AverageSection**(*data_manager*, *startdate*, *member*, *chunk*, *domain*, *variable*, *box*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Compute an average of a given zone. The variable MUST be in a regular grid

Original author Virginie Guemas <virginie.guemas@bsc.es>

Contributor Javier Vegas-Regidor<javier.vegas@bsc.es>

Created March 2012

Last modified June 2016

Parameters

- **data_manager** (*DataManager*) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number
- **variable** (*str*) – variable's name
- **domain** (*ModelingRealm*) – variable's domain
- **box** (*Box*) – box to use for the average

alias = 'avgsection'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – variable, minimum longitude, maximum longitude, minimum latitude, maximum latitude, domain=ocean

Returns

8.3.3 earthdiagnostics.ocean.convectionsites

class earthdiagnostics.ocean.convectionsites.**ConvectionSites**(*data_manager*, *startdate*, *member*, *chunk*, *model_version*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Compute the intensity of convection in the four main convection sites

Original author Virginie Guemas <virginie.guemas@bsc.es>

Contributor Javier Vegas-Regidor<javier.vegas@bsc.es>

Created October 2013

Last modified June 2016

Parameters

- **data_manager** (*DataManager*) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number

- **chunk** (*int*) – chunk's number
- **model_version** (*str*) – model version

alias = 'convection'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – None

Returns

8.3.4 earthdiagnostics.ocean.cutsection

class earthdiagnostics.ocean.cutsection.**CutSection**(*data_manager*, *startdate*, *member*, *chunk*, *domain*, *variable*, *zonal*, *value*)
Bases: *earthdiagnostics.diagnostic.Diagnostic*

Cuts a meridional or zonal section

Original author Virginie Guemas <virginie.guemas@bsc.es>

Contributor Javier Vegas-Regidor<javier.vegas@bsc.es>

Created September 2012

Last modified June 2016

Parameters

- **data_manager** (DataManager) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number
- **variable** (*str*) – variable's name
- **domain** (*Domain*) – variable's domain
- **zonal** (*bool*) – specifies if section is zonal or meridional
- **value** (*int*) – value of the section's coordinate

alias = 'cutsection'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – variable, zonal, value, domain=ocean

Returns

8.3.5 earthdiagnostics.ocean.gyres

class earthdiagnostics.ocean.gyres.**Gyres**(*data_manager*, *startdate*, *member*, *chunk*, *model_version*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Compute the intensity of the subtropical and subpolar gyres

Original author Virginie Guemas <virginie.guemas@bsc.es>

Contributor Javier Vegas-Regidor<javier.vegas@bsc.es>

Created October 2013

Last modified June 2016

Parameters

- **data_manager** (DataManager) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number
- **model_version** (*str*) – model version

alias = 'gyres'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – None

Returns

8.3.6 earthdiagnostics.ocean.heatcontent

class earthdiagnostics.ocean.heatcontent.**HeatContent**(*data_manager*, *startdate*, *member*, *chunk*, *basin*, *mixed_layer*, *box*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Compute the total ocean heat content

Original author Virginie Guemas <virginie.guemas@bsc.es>

Contributor Javier Vegas-Regidor<javier.vegas@bsc.es>

Created May 2012

Last modified June 2016

Parameters

- **data_manager** (DataManager) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number
- **mixed_layer** (*int*) – If 1, restricts calculation to the mixed layer, if -1 exclude it. If 0, no effect
- **box** (Box) – box to use for the average

alias = 'ohc'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – basin, mixed layer option (1 to only compute at the mixed layer, -1 to exclude it, 0 to ignore), minimum depth, maximum depth

Returns

8.3.7 earthdiagnostics.ocean.heatcontentlayer

class earthdiagnostics.ocean.heatcontentlayer.**HeatContentLayer**(*data_manager*, *startdate*, *member*, *chunk*, *box*, *weight*, *min_level*, *max_level*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Point-wise Ocean Heat Content in a specified ocean thickness (J/m-2)

Original author Isabel Andreu Burillo

Contributor Virginie Guemas <virginie.guemas@bsc.es>

Contributor Eleftheria Exarchou <eleftheria.exarchou@bsc.es>

Contributor Javier Vegas-Regidor<javier.vegas@bsc.es>

Created June 2012

Last modified June 2016

Parameters

- **data_manager** (DataManager) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number

- **chunk** (*int*) – chunk's number
- **box** (Box) – box to use for the calculations

alias = 'ohclayer'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – minimum depth, maximum depth, basin=Global

8.3.8 earthdiagnostics.ocean.interpolate

class earthdiagnostics.ocean.interpolate.**Interpolate**(*data_manager*, *startdate*, *member*, *chunk*, *domain*, *variable*, *target_grid*, *model_version*, *invert_lat*, *original_grid*)
Bases: *earthdiagnostics.diagnostic.Diagnostic*

3-dimensional conservative interpolation to the regular atmospheric grid. It can also be used for 2D (i,j) variables

Original author Virginie Guemas <virginie.guemas@bsc.es>

Contributor Javier Vegas-Regidor<javier.vegas@bsc.es>

Created November 2012

Last modified June 2016

Parameters

- **data_manager** (DataManager) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number
- **variable** (*str*) – variable's name
- **domain** (*Domain*) – variable's domain
- **model_version** (*str*) – model version

alias = 'interp'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class

- **options** (*list[str]*) – target_grid, variable, domain=ocean

Returns

8.3.9 earthdiagnostics.ocean.interpolatecdo

class earthdiagnostics.ocean.interpolatecdo.**InterpolateCDO**(*data_manager*, *startdate*, *member*, *chunk*, *domain*, *variable*, *target_grid*, *model_version*, *mask_oceans*, *original_grid*, *weights*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

3-dimensional conservative interpolation to the regular atmospheric grid. It can also be used for 2D (i,j) variables

Original author Javier Vegas-Regidor<javier.vegas@bsc.es>

Created October 2016

Parameters

- **data_manager** (*DataManager*) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number
- **variable** (*str*) – variable's name
- **domain** (*ModelingRealm*) – variable's domain
- **model_version** (*str*) – model version

alias = 'interpcdo'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – target_grid, variable, domain=ocean

Returns

8.3.10 earthdiagnostics.ocean.maxmoc

class earthdiagnostics.ocean.maxmoc.**MaxMoc**(*data_manager*, *startdate*, *member*, *year*, *basin*, *box*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Compute an Atlantic MOC index by finding the maximum of the annual mean meridional overturning in a latitude / depth region

Original author Virginie Guemas <virginie.guemas@bsc.es>

Contributor Javier Vegas-Regidor<javier.vegas@bsc.es>

Created March 2012

Last modified June 2016

Parameters

- **data_manager** (*DataManager*) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **year** (*int*) – year to compute
- **basin** (*Basin*) – basin to compute
- **box** (*Box*) – box to compute

alias = 'mocmax'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each complete year to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – minimum latitude, maximum latitude, minimum depth, maximum depth, basin=global

Returns

8.3.11 earthdiagnostics.ocean.mixedlayerheatcontent

class earthdiagnostics.ocean.mixedlayerheatcontent.**MixedLayerHeatContent**(*data_manager*, *startdate*, *member*, *chunk*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Compute mixed layer heat content

Original author Virginie Guemas <virginie.guemas@bsc.es>

Contributor Javier Vegas-Regidor<javier.vegas@bsc.es>

Created February 2012

Last modified June 2016

Parameters

- **data_manager** (*DataManager*) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number

- **chunk** (*int*) – chunk's number

alias = 'mlotsthc'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – None

Returns

8.3.12 earthdiagnostics.ocean.mixedlayersaltcontent

class earthdiagnostics.ocean.mixedlayersaltcontent.**MixedLayerSaltContent**(*data_manager*, *startdate*, *member*, *chunk*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Compute mixed layer salt content

Original author Virginie Guemas <virginie.guemas@bsc.es>

Contributor Javier Vegas-Regidor<javier.vegas@bsc.es>

Created February 2012

Last modified June 2016

Parameters

- **data_manager** (*DataManager*) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number

alias = 'mlotstsc'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – None

Returns

8.3.13 earthdiagnostics.ocean.moc

class earthdiagnostics.ocean.moc.**Moc**(*data_manager*, *startdate*, *member*, *chunk*)
Bases: *earthdiagnostics.diagnostic.Diagnostic*

Compute the MOC for oceanic basins

Original author Virginie Guemas <virginie.guemas@bsc.es>

Contributor Javier Vegas-Regidor<javier.vegas@bsc.es>

Created March 2012

Last modified June 2016

Parameters

- **data_manager** (DataManager) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number

alias = 'moc'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – None

Returns

8.3.14 earthdiagnostics.ocean.mxl

class earthdiagnostics.ocean.mxl.**Mxl**(*data_manager*, *startdate*, *member*, *chunk*)
Bases: *earthdiagnostics.diagnostic.Diagnostic*

Compute the mixed layer depth

Parameters

- **data_manager** (DataManager) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number

alias = 'mxl'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod **generate_jobs**(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – None

Returns

8.3.15 earthdiagnostics.ocean.psi

class earthdiagnostics.ocean.psi.**Psi**(*data_manager*, *startdate*, *member*, *chunk*)
Bases: *earthdiagnostics.diagnostic.Diagnostic*

Compute the barotropic stream function

Original author Virginie Guemas <virginie.guemas@bsc.es>

Contributor Javier Vegas-Regidor<javier.vegas@bsc.es>

Created March 2012

Last modified June 2016

Parameters

- **data_manager** (DataManager) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number

alias = 'psi'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod **generate_jobs**(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – None

Returns

8.3.16 earthdiagnostics.ocean.rotation

class earthdiagnostics.ocean.rotation.**Rotation**(*data_manager*, *startdate*, *member*, *chunk*, *domain*, *variableu*, *variablev*, *executable*)
Bases: *earthdiagnostics.diagnostic.Diagnostic*

Cuts a meridional or zonal section

Original author Virginie Guemas <virginie.guemas@bsc.es>

Contributor Javier Vegas-Regidor<javier.vegas@bsc.es>

Created September 2012

Last modified June 2016

Parameters

- **data_manager** (*DataManager*) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number
- **variable** (*str*) – variable's name
- **domain** (*Domain*) – variable's domain

alias = 'rotate'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – variable, zonal, value, domain=ocean

Returns

8.3.17 earthdiagnostics.ocean.siasiesiv

class earthdiagnostics.ocean.siasiesiv.**Siasiesiv**(*data_manager*, *startdate*, *member*, *chunk*, *basin*, *mask*)
Bases: *earthdiagnostics.diagnostic.Diagnostic*

Compute the sea ice extent , area and volume in both hemispheres or a specified region.

Original author Virginie Guemas <virginie.guemas@bsc.es>

Contributor Neven Fuckar <neven.fuckar@bsc.es>

Contributor Ruben Cruz <ruben.cruzgarcia@bsc.es>

Contributor Javier Vegas-Regidor <javier.vegas@bsc.es>

Created April 2012

Last modified June 2016

alias = 'siasiesiv'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class

- **options** (*list[str]*) – basin

Returns

8.3.18 earthdiagnostics.ocean.verticalmean

class earthdiagnostics.ocean.verticalmean.**VerticalMean**(*data_manager*, *startdate*, *member*, *chunk*, *variable*, *box*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Chooses vertical level in ocean, or vertically averages between 2 or more ocean levels

Original author Virginie Guemas <virginie.guemas@bsc.es>

Contributor Eleftheria Exarchou <eleftheria.exarchou@bsc.es>

Contributor Javier Vegas-Regidor <javier.vegas@bsc.es>

Created February 2012

Last modified June 2016

Parameters

- **data_manager** (DataManager) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number
- **variable** (*str*) – variable to average
- **box** (Box) – box used to restrict the vertical mean

alias = 'vertmean'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – variable, minimum depth (level), maximum depth (level)

Returns

8.3.19 earthdiagnostics.ocean.verticalmeanmeters

class earthdiagnostics.ocean.verticalmeanmeters.**VerticalMeanMeters**(*data_manager*, *startdate*, *member*, *chunk*, *domain*, *variable*, *box*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Averages vertically any given variable

Original author Virginie Guemas <virginie.guemas@bsc.es>

Contributor Javier Vegas-Regidor<javier.vegas@bsc.es>

Created February 2012

Last modified June 2016

Parameters

- **data_manager** (*DataManager*) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number
- **variable** (*str*) – variable to average
- **box** (*Box*) – box used to restrict the vertical mean

alias = 'vertmeanmeters'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – variable, minimum depth (meters), maximum depth (meters)

Returns

8.4 earthdiagnostics.statistics

8.4.1 earthdiagnostics.statistics.climatologicalpercentile

class earthdiagnostics.statistics.climatologicalpercentile.**ClimatologicalPercentile**(*data_manager*, *domain*, *variable*, *leadtimes*, *num_bins*, *experiment_config*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Calculates the climatological percentiles for the given leadtimes

Parameters

- **data_manager** (DataManager) – data management object
- **variable** (*str*) – variable to average
- **experiment_config** (ExperimentConfig) –

alias = 'climpercent'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – domain, variable, percentil number, maximum depth (level)

Returns

8.4.2 earthdiagnostics.statistics.monthlypercentile

class earthdiagnostics.statistics.monthlypercentile.**MonthlyPercentile**(*data_manager*, *startdate*, *member*, *chunk*, *domain*, *variable*, *percentiles*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Calculates the montlhy percentiles

Parameters

- **data_manager** (DataManager) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number
- **variable** (*str*) – variable to average

alias = 'monpercent'
Diagnostic alias for the configuration file

compute()
Runs the diagnostic

classmethod generate_jobs(*diags*, *options*)
Creates a job for each chunk to compute the diagnostic

Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – domain, variable, percentil number, maximum depth (level)

Returns

e

earthdiagnostics.box, 25
earthdiagnostics.cdftools, 26
earthdiagnostics.cmorizer, 26
earthdiagnostics.cmormanager, 26
earthdiagnostics.config, 29
earthdiagnostics.constants, 30
earthdiagnostics.datamanager, 34
earthdiagnostics.diagnostic, 37
earthdiagnostics.earthdiags, 38
earthdiagnostics.general.attribute, 42
earthdiagnostics.general.dailymean, 43
earthdiagnostics.general.monthlymean, 44
earthdiagnostics.general.relink, 44
earthdiagnostics.general.relinkall, 45
earthdiagnostics.general.rewrite, 46
earthdiagnostics.general.scale, 46
earthdiagnostics.general.yearlymean, 47
earthdiagnostics.ocean.areamoc, 48
earthdiagnostics.ocean.averagesection, 48
earthdiagnostics.ocean.convectionsites, 49
earthdiagnostics.ocean.cutsection, 50
earthdiagnostics.ocean.gyres, 51
earthdiagnostics.ocean.heatcontent, 51
earthdiagnostics.ocean.heatcontentlayer, 52
earthdiagnostics.ocean.interpolate, 53
earthdiagnostics.ocean.interpolatecdo, 54
earthdiagnostics.ocean.maxmoc, 54
earthdiagnostics.ocean.mixedlayerheatcontent, 55
earthdiagnostics.ocean.mixedlayersaltcontent, 56
earthdiagnostics.ocean.moc, 57
earthdiagnostics.ocean.mxl, 57
earthdiagnostics.ocean.psi, 58
earthdiagnostics.ocean.rotation, 58
earthdiagnostics.ocean.siasiesiv, 59
earthdiagnostics.ocean.verticalmean, 60
earthdiagnostics.ocean.verticalmeanmeters, 60
earthdiagnostics.statistics.climatologicalpercentil 61
earthdiagnostics.statistics.monthlypercentile, 62
earthdiagnostics.utils, 39
earthdiagnostics.variable, 42

A

add_conversion() (earthdiagnostics.datamanager.UnitConversion class method), 37
alias (earthdiagnostics.diagnostic.Diagnostic attribute), 37
alias (earthdiagnostics.general.attribute.Attribute attribute), 43
alias (earthdiagnostics.general.dailymean.DailyMean attribute), 43
alias (earthdiagnostics.general.monthlymean.MonthlyMean attribute), 44
alias (earthdiagnostics.general.relink.Relink attribute), 45
alias (earthdiagnostics.general.relinkall.RelinkAll attribute), 45
alias (earthdiagnostics.general.rewrite.Rewrite attribute), 46
alias (earthdiagnostics.general.scale.Scale attribute), 47
alias (earthdiagnostics.general.yearlymean.YearlyMean attribute), 47
alias (earthdiagnostics.ocean.areamoc.AreaMoc attribute), 48
alias (earthdiagnostics.ocean.averagesection.AverageSection attribute), 49
alias (earthdiagnostics.ocean.convectionsites.ConvectionSites attribute), 50
alias (earthdiagnostics.ocean.cutsection.CutSection attribute), 50
alias (earthdiagnostics.ocean.gyres.Gyres attribute), 51
alias (earthdiagnostics.ocean.heatcontent.HeatContent attribute), 52
alias (earthdiagnostics.ocean.heatcontentlayer.HeatContentLayer attribute), 53
alias (earthdiagnostics.ocean.interpolate.Interpolate attribute), 53
alias (earthdiagnostics.ocean.interpolatecdo.InterpolateCDO attribute), 54
alias (earthdiagnostics.ocean.maxmoc.MaxMoc attribute), 55
alias (earthdiagnostics.ocean.mixedlayerheatcontent.MixedLayerHeatContent attribute), 56
alias (earthdiagnostics.ocean.mixedlayersaltcontent.MixedLayerSaltContent attribute), 56
alias (earthdiagnostics.ocean.moc.Moc attribute), 57
alias (earthdiagnostics.ocean.mxl.Mxl attribute), 57
alias (earthdiagnostics.ocean.psi.Psi attribute), 58
alias (earthdiagnostics.ocean.rotation.Rotation attribute), 59
alias (earthdiagnostics.ocean.siasiesiv.Siasiesiv attribute), 59
alias (earthdiagnostics.ocean.verticalmean.VerticalMean attribute), 60
alias (earthdiagnostics.ocean.verticalmeanmeters.VerticalMeanMeters attribute), 61
alias (earthdiagnostics.statistics.climatologicalpercentile.ClimatologicalPerc attribute), 62
alias (earthdiagnostics.statistics.monthlypercentile.MonthlyPercentile attribute), 62
Antarctic (earthdiagnostics.constants.Basins attribute), 30
AntarcticAtlantic (earthdiagnostics.constants.Basins attribute), 30
AntarcticIndian (earthdiagnostics.constants.Basins attribute), 30
Arctic (earthdiagnostics.constants.Basins attribute), 30
ArcticMarginalSeas (earthdiagnostics.constants.Basins attribute), 30
ArcticNorthAtlantic (earthdiagnostics.constants.Basins attribute), 30
AreaMoc (class in earthdiagnostics.ocean.areamoc), 48
Atlantic (earthdiagnostics.constants.Basins attribute), 30
Attribute (class in earthdiagnostics.general.attribute), 42
autoclean (earthdiagnostics.utils.TempFile attribute), 39
available_cpu_count() (earthdiagnostics.utils.Utils static method), 39
AverageSection (class in earthdiagnostics.ocean.averagesection), 48

B

Baffin (earthdiagnostics.constants.Basins attribute), 31
Baffin_Bay (earthdiagnostics.constants.Basins attribute), 31
Baltic_Sea (earthdiagnostics.constants.Basins attribute), 31

Barents_Sea (earthdiagnostics.constants.Basins attribute), 31
BarKara (earthdiagnostics.constants.Basins attribute), 31
Basin (class in earthdiagnostics.constants), 30
Basins (class in earthdiagnostics.constants), 30
Beaufort_Chukchi_Sea (earthdiagnostics.constants.Basins attribute), 31
Beaufort_Sea (earthdiagnostics.constants.Basins attribute), 31
Bellingshausen_Sea (earthdiagnostics.constants.Basins attribute), 31
Bering (earthdiagnostics.constants.Basins attribute), 31
Bering_Strait (earthdiagnostics.constants.Basins attribute), 31
Box (class in earthdiagnostics.box), 25
box (earthdiagnostics.constants.Basin attribute), 30

C

Canadian_Waters (earthdiagnostics.constants.Basins attribute), 31
CanArch (earthdiagnostics.constants.Basins attribute), 31
Caspian_Sea (earthdiagnostics.constants.Basins attribute), 31
CDFTools (class in earthdiagnostics.cdftools), 26
cdftools_path (earthdiagnostics.config.Config attribute), 29
cdo (earthdiagnostics.utils.Utils attribute), 39
Central_Arctic (earthdiagnostics.constants.Basins attribute), 31
Chukchi_Sea (earthdiagnostics.constants.Basins attribute), 31
clean() (earthdiagnostics.utils.TempFile static method), 39
ClimatologicalPercentile (class in earthdiagnostics.statistics.climatologicalpercentile), 61
cmorize_atmos() (earthdiagnostics.cmorizer.Cmorizer method), 26
cmorize_ocean() (earthdiagnostics.cmorizer.Cmorizer method), 26
Cmorizer (class in earthdiagnostics.cmorizer), 26
CMORManager (class in earthdiagnostics.cmormanager), 26
compute() (earthdiagnostics.diagnostic.Diagnostic method), 37
compute() (earthdiagnostics.general.attribute.Attribute method), 43
compute() (earthdiagnostics.general.dailymean.DailyMean method), 43
compute() (earthdiagnostics.general.monthlymean.MonthlyMean method), 44
compute() (earthdiagnostics.general.relink.Relink method), 45
compute() (earthdiagnostics.general.relinkall.RelinkAll method), 45
compute() (earthdiagnostics.general.rewrite.Rewrite method), 46
compute() (earthdiagnostics.general.scale.Scale method), 47
compute() (earthdiagnostics.general.yearlymean.YearlyMean method), 47
compute() (earthdiagnostics.ocean.areamoc.AreaMoc method), 48
compute() (earthdiagnostics.ocean.averagesection.AverageSection method), 49
compute() (earthdiagnostics.ocean.convectionsites.ConvectionSites method), 50
compute() (earthdiagnostics.ocean.cutsection.CutSection method), 50
compute() (earthdiagnostics.ocean.gyres.Gyres method), 51
compute() (earthdiagnostics.ocean.heatcontent.HeatContent method), 52
compute() (earthdiagnostics.ocean.heatcontentlayer.HeatContentLayer method), 53
compute() (earthdiagnostics.ocean.interpolate.Interpolate method), 53
compute() (earthdiagnostics.ocean.interpolatecdo.InterpolateCDO method), 54
compute() (earthdiagnostics.ocean.maxmoc.MaxMoc method), 55
compute() (earthdiagnostics.ocean.mixedlayerheatcontent.MixedLayerHeatContent method), 56
compute() (earthdiagnostics.ocean.mixedlayersaltcontent.MixedLayerSaltContent method), 56
compute() (earthdiagnostics.ocean.moc.Moc method), 57
compute() (earthdiagnostics.ocean.mxl.Mxl method), 57
compute() (earthdiagnostics.ocean.psi.Psi method), 58
compute() (earthdiagnostics.ocean.rotation.Rotation method), 59
compute() (earthdiagnostics.ocean.siasiesiv.Siasiesiv method), 59
compute() (earthdiagnostics.ocean.verticalmean.VerticalMean method), 60
compute() (earthdiagnostics.ocean.verticalmeanmeters.VerticalMeanMeters method), 61

compute() (earthdiagnostics.statistics.climatologicalpercentile.ClimatologicalPercentile method), 62
compute() (earthdiagnostics.statistics.monthlypercentile.MonthlyPercentile method), 62
con_files (earthdiagnostics.config.Config attribute), 29
concat_variables() (earthdiagnostics.utils.Utils static method), 40
Config (class in earthdiagnostics.config), 29
ConvectionSites (class in earthdiagnostics.ocean.convectionsites), 49
convert2netcdf4() (earthdiagnostics.utils.Utils static method), 40
copy_dimension() (earthdiagnostics.utils.Utils static method), 40
copy_file() (earthdiagnostics.utils.Utils static method), 40
copy_variable() (earthdiagnostics.utils.Utils static method), 40
create_folder_tree() (earthdiagnostics.utils.Utils static method), 40
CutSection (class in earthdiagnostics.ocean.cutsection), 50

D

DailyMean (class in earthdiagnostics.general.dailymean), 43
data_adaptor (earthdiagnostics.config.Config attribute), 29
data_dir (earthdiagnostics.config.Config attribute), 29
data_type (earthdiagnostics.config.Config attribute), 29
DataManager (class in earthdiagnostics.datamanager), 34
depth_in_meters (earthdiagnostics.box.Box attribute), 25
Diagnostic (class in earthdiagnostics.diagnostic), 37

E

earthdiagnostics.box (module), 25
earthdiagnostics.cdftools (module), 26
earthdiagnostics.cmorizer (module), 26
earthdiagnostics.cmormanager (module), 26
earthdiagnostics.config (module), 29
earthdiagnostics.constants (module), 30
earthdiagnostics.datamanager (module), 34
earthdiagnostics.diagnostic (module), 37
earthdiagnostics.earthdiags (module), 38
earthdiagnostics.general.attribute (module), 42
earthdiagnostics.general.dailymean (module), 43
earthdiagnostics.general.monthlymean (module), 44
earthdiagnostics.general.relink (module), 44
earthdiagnostics.general.relinkall (module), 45
earthdiagnostics.general.rewrite (module), 46
earthdiagnostics.general.scale (module), 46
earthdiagnostics.general.yearlymean (module), 47
earthdiagnostics.ocean.areamoc (module), 48
earthdiagnostics.ocean.averagesection (module), 48
earthdiagnostics.ocean.convectionsites (module), 49
earthdiagnostics.ocean.cutsection (module), 50
earthdiagnostics.ocean.gyres (module), 51
earthdiagnostics.ocean.heatcontent (module), 51
earthdiagnostics.ocean.heatcontentlayer (module), 52
earthdiagnostics.ocean.interpolate (module), 53
earthdiagnostics.ocean.interpolatecdo (module), 54
earthdiagnostics.ocean.maxmoc (module), 54
earthdiagnostics.ocean.mixedlayerheatcontent (module), 55
earthdiagnostics.ocean.mixedlayersaltcontent (module), 56
earthdiagnostics.ocean.moc (module), 57
earthdiagnostics.ocean.mxl (module), 57
earthdiagnostics.ocean.psi (module), 58
earthdiagnostics.ocean.rotation (module), 58
earthdiagnostics.ocean.siasiesiv (module), 59
earthdiagnostics.ocean.verticalmean (module), 60
earthdiagnostics.ocean.verticalmeanmeters (module), 60
earthdiagnostics.statistics.climatologicalpercentile (module), 61
earthdiagnostics.statistics.monthlypercentile (module), 62
earthdiagnostics.utils (module), 39
earthdiagnostics.variable (module), 42
EarthDiags (class in earthdiagnostics.earthdiags), 38
East_Siberian_Sea (earthdiagnostics.constants.Basins attribute), 31
Eastern_Central_Arctic (earthdiagnostics.constants.Basins attribute), 31
ECEARTH_2_3_O1L42 (earthdiagnostics.constants.Models attribute), 34
ECEARTH_3_0_O1L46 (earthdiagnostics.constants.Models attribute), 34
ECEARTH_3_0_O25L46 (earthdiagnostics.constants.Models attribute), 34
ECEARTH_3_0_O25L75 (earthdiagnostics.constants.Models attribute), 34
ECEARTH_3_1_O25L75 (earthdiagnostics.constants.Models attribute), 34
ECEARTH_3_2_O1L75 (earthdiagnostics.constants.Models attribute), 34
execute_shell_command() (earthdiagnostics.utils.Utils static method), 40
experiment (earthdiagnostics.config.Config attribute), 29
ExperimentConfig (class in earthdiagnostics.config), 29
extract_variable() (earthdiagnostics.cmorizer.Cmorizer method), 26

F

file_exists() (earthdiagnostics.datamanager.DataManager method), 34
files (earthdiagnostics.utils.TempFile attribute), 39

Fram_Strait (earthdiagnostics.constants.Basins attribute), 31
frequency (earthdiagnostics.config.Config attribute), 29
fullname (earthdiagnostics.constants.Basin attribute), 30

G

generate_jobs() (earthdiagnostics.diagnostic.Diagnostic class method), 37
generate_jobs() (earthdiagnostics.general.attribute.Attribute class method), 43
generate_jobs() (earthdiagnostics.general.dailymean.DailyMean class method), 43
generate_jobs() (earthdiagnostics.general.monthlymean.MonthlyMean class method), 44
generate_jobs() (earthdiagnostics.general.relink.Relink class method), 45
generate_jobs() (earthdiagnostics.general.relinkall.RelinkAll class method), 45
generate_jobs() (earthdiagnostics.general.rewrite.Rewrite class method), 46
generate_jobs() (earthdiagnostics.general.scale.Scale class method), 47
generate_jobs() (earthdiagnostics.general.yearlymean.YearlyMean class method), 47
generate_jobs() (earthdiagnostics.ocean.areamoc.AreaMoc class method), 48
generate_jobs() (earthdiagnostics.ocean.averagesection.AverageSection class method), 49
generate_jobs() (earthdiagnostics.ocean.convectionsites.ConvectionSites class method), 50
generate_jobs() (earthdiagnostics.ocean.cutsection.CutSection class method), 50
generate_jobs() (earthdiagnostics.ocean.gyres.Gyres class method), 51
generate_jobs() (earthdiagnostics.ocean.heatcontent.HeatContent class method), 52
generate_jobs() (earthdiagnostics.ocean.heatcontentlayer.HeatContentLayer class method), 53
generate_jobs() (earthdiagnostics.ocean.interpolate.Interpolate class method), 53
generate_jobs() (earthdiagnostics.ocean.interpolatecdo.InterpolateCDO class method), 54
generate_jobs() (earthdiagnostics.ocean.maxmoc.MaxMoc class method), 55
generate_jobs() (earthdiagnostics.ocean.mixedlayerheatcontent.MixedLayerHeatContent class method), 56
generate_jobs() (earthdiagnostics.ocean.mixedlayersaltcontent.MixedLayerSaltContent class method), 56
generate_jobs() (earthdiagnostics.ocean.moc.Moc class method), 57
generate_jobs() (earthdiagnostics.ocean.mxl.Mxl class method), 57
generate_jobs() (earthdiagnostics.ocean.psi.Psi class method), 58
generate_jobs() (earthdiagnostics.ocean.rotation.Rotation class method), 59
generate_jobs() (earthdiagnostics.ocean.siasiesiv.Siasiesiv class method), 59
generate_jobs() (earthdiagnostics.ocean.verticalmean.VerticalMean class method), 60
generate_jobs() (earthdiagnostics.ocean.verticalmeanmeters.VerticalMeanMeters class method), 61
generate_jobs() (earthdiagnostics.statistics.climatologicalpercentile.ClimatologicalPercentile class method), 62
generate_jobs() (earthdiagnostics.statistics.monthlypercentile.MonthlyPercentile class method), 62
get() (earthdiagnostics.utils.TempFile static method), 39
get_chunk_list() (earthdiagnostics.config.ExperimentConfig method), 29
get_commands() (earthdiagnostics.config.Config method), 29
get_conversion_factor_offset() (earthdiagnostics.datamanager.UnitConversion class method), 37
get_datetime_from_netcdf() (earthdiagnostics.utils.Utils static method), 41
get_depth_str() (earthdiagnostics.box.Box method), 25
get_diagnostic() (earthdiagnostics.diagnostic.Diagnostic static method), 38
get_file() (earthdiagnostics.cmormanager.CMORManager method), 26
get_file() (earthdiagnostics.datamanager.DataManager method), 35
get_file_hash() (earthdiagnostics.utils.Utils static method), 41

get_file_path() (earthdiagnostics.cmormanager.CMORManager method), 27
get_full_years() (earthdiagnostics.config.ExperimentConfig method), 29
get_lat_str() (earthdiagnostics.box.Box method), 25
get_lon_str() (earthdiagnostics.box.Box method), 25
get_mask() (earthdiagnostics.utils.Utils static method), 41
get_member_list() (earthdiagnostics.config.ExperimentConfig method), 29
get_member_str() (earthdiagnostics.config.ExperimentConfig method), 30
get_year() (earthdiagnostics.cmormanager.CMORManager method), 27
get_year() (earthdiagnostics.datamanager.DataManager method), 35
get_year_chunks() (earthdiagnostics.config.ExperimentConfig method), 30
Global (earthdiagnostics.constants.Basins attribute), 31
Global_Ocean (earthdiagnostics.constants.Basins attribute), 31
GLORYS2_V1_O25L75 (earthdiagnostics.constants.Models attribute), 34
Greenland_Sea (earthdiagnostics.constants.Basins attribute), 31
Grnland (earthdiagnostics.constants.Basins attribute), 32
Gyres (class in earthdiagnostics.ocean.gyres), 51

H

HeatContent (class in earthdiagnostics.ocean.heatcontent), 51
HeatContentLayer (class in earthdiagnostics.ocean.heatcontentlayer), 52
Hudson (earthdiagnostics.constants.Basins attribute), 32

I

Icelandic_Sea (earthdiagnostics.constants.Basins attribute), 32
Indian (earthdiagnostics.constants.Basins attribute), 32
IndoPacific (earthdiagnostics.constants.Basins attribute), 32
Interpolate (class in earthdiagnostics.ocean.interpolate), 53
InterpolateCDO (class in earthdiagnostics.ocean.interpolatecdo), 54
Irminger_Sea (earthdiagnostics.constants.Basins attribute), 32

K

Kara_Gate_Strait (earthdiagnostics.constants.Basins attribute), 32
Kara_Sea (earthdiagnostics.constants.Basins attribute), 32

L

Labrador_Sea (earthdiagnostics.constants.Basins attribute), 32
Laptev_East_Siberian_Chukchi_Seas (earthdiagnostics.constants.Basins attribute), 32
Laptev_East_Siberian_Seas (earthdiagnostics.constants.Basins attribute), 32
Laptev_Sea (earthdiagnostics.constants.Basins attribute), 32
Lincoln_Sea (earthdiagnostics.constants.Basins attribute), 32
link_file() (earthdiagnostics.cmormanager.CMORManager method), 27
link_file() (earthdiagnostics.datamanager.DataManager method), 35
load_conversions() (earthdiagnostics.datamanager.UnitConversion class method), 37

M

max_cores (earthdiagnostics.config.Config attribute), 29
max_depth (earthdiagnostics.box.Box attribute), 25
max_lat (earthdiagnostics.box.Box attribute), 25
max_lon (earthdiagnostics.box.Box attribute), 25
MaxMoc (class in earthdiagnostics.ocean.maxmoc), 54
Mediterranean_Sea (earthdiagnostics.constants.Basins attribute), 32
min_depth (earthdiagnostics.box.Box attribute), 25
min_lat (earthdiagnostics.box.Box attribute), 25
min_lon (earthdiagnostics.box.Box attribute), 25
MixedLayerHeatContent (class in earthdiagnostics.ocean.mixedlayerheatcontent), 55
MixedLayerSaltContent (class in earthdiagnostics.ocean.mixedlayersaltcontent), 56
Moc (class in earthdiagnostics.ocean.moc), 57
Models (class in earthdiagnostics.constants), 33
MonthlyMean (class in earthdiagnostics.general.monthlymean), 44
MonthlyPercentile (class in earthdiagnostics.statistics.monthlypercentile), 62
move_file() (earthdiagnostics.utils.Utils static method), 41
Mxl (class in earthdiagnostics.ocean.mxl), 57

N

Nares_Strait (earthdiagnostics.constants.Basins attribute), 32
nco (earthdiagnostics.utils.Utils attribute), 41
NEMO_3_2_O1L42 (earthdiagnostics.constants.Models attribute), 34
NEMO_3_3_O1L46 (earthdiagnostics.constants.Models attribute), 34

NEMO_3_6_O1L46 (earthdiagnostics.constants.Models attribute), 34
NEMOVAR_O1L42 (earthdiagnostics.constants.Models attribute), 34
NetCDFFile (class in earthdiagnostics.datamanager), 36
Nordic_Barents_Seas (earthdiagnostics.constants.Basins attribute), 32
Nordic_Seas (earthdiagnostics.constants.Basins attribute), 32
North_Atlantic_Arctic (earthdiagnostics.constants.Basins attribute), 32
North_Hemisphere_Ocean (earthdiagnostics.constants.Basins attribute), 33
NorthAtlantic (earthdiagnostics.constants.Basins attribute), 32
NorthPacific (earthdiagnostics.constants.Basins attribute), 32
NorthWest_Passage (earthdiagnostics.constants.Basins attribute), 32
Norwegian_Sea (earthdiagnostics.constants.Basins attribute), 33

O

Okhotsk (earthdiagnostics.constants.Basins attribute), 33
openCdf() (earthdiagnostics.utils.Utils static method), 41
OpenOcean (earthdiagnostics.constants.Basins attribute), 33

P

Pacific (earthdiagnostics.constants.Basins attribute), 33
parse() (earthdiagnostics.constants.Basins class method), 33
parse_args() (earthdiagnostics.earthdiags.EarthDiags static method), 38
prefix (earthdiagnostics.utils.TempFile attribute), 39
prepare() (earthdiagnostics.cmormanager.CMORManager method), 28
prepare() (earthdiagnostics.datamanager.DataManager method), 36
Psi (class in earthdiagnostics.ocean.psi), 58

R

register() (earthdiagnostics.diagnostic.Diagnostic static method), 38
Relink (class in earthdiagnostics.general.relink), 44
RelinkAll (class in earthdiagnostics.general.relinkall), 45
remove_file() (earthdiagnostics.utils.Utils static method), 41
rename_variable() (earthdiagnostics.utils.Utils static method), 41
rename_variables() (earthdiagnostics.utils.Utils static method), 42
restore_meshes (earthdiagnostics.config.Config attribute), 29
Rewrite (class in earthdiagnostics.general.rewrite), 46
Ross_Sea (earthdiagnostics.constants.Basins attribute), 33
Rotation (class in earthdiagnostics.ocean.rotation), 58
run() (earthdiagnostics.cdftools.CDFTools method), 26
run() (earthdiagnostics.earthdiags.EarthDiags method), 39

S

Scale (class in earthdiagnostics.general.scale), 46
scratch_dir (earthdiagnostics.config.Config attribute), 29
scratch_folder (earthdiagnostics.utils.TempFile attribute), 39
scratch_masks (earthdiagnostics.config.Config attribute), 29
send_file() (earthdiagnostics.cmormanager.CMORManager method), 28
send_file() (earthdiagnostics.datamanager.DataManager method), 36
send_file() (earthdiagnostics.diagnostic.Diagnostic method), 38
Serreze_Arctic (earthdiagnostics.constants.Basins attribute), 33
setminmax() (earthdiagnostics.utils.Utils static method), 42
shortname (earthdiagnostics.constants.Basin attribute), 30
Siasiesiv (class in earthdiagnostics.ocean.siasiesiv), 59
Southern_Hemisphere (earthdiagnostics.constants.Basins attribute), 33
StLawr (earthdiagnostics.constants.Basins attribute), 33
Subpolar_Gyre (earthdiagnostics.constants.Basins attribute), 33

T

TempFile (class in earthdiagnostics.utils), 39
TotalArc (earthdiagnostics.constants.Basins attribute), 33
TropicalAtlantic (earthdiagnostics.constants.Basins attribute), 33
TropicalIndian (earthdiagnostics.constants.Basins attribute), 33
TropicalPacific (earthdiagnostics.constants.Basins attribute), 33

U

UnitConversion (class in earthdiagnostics.datamanager), 37
untar() (earthdiagnostics.utils.Utils static method), 42
unzip() (earthdiagnostics.utils.Utils static method), 42
Utils (class in earthdiagnostics.utils), 39
Utils.ExecutionError, 39
Utils.UnzipException, 39

V

Variable (class in earthdiagnostics.variable), 42
VariableAlias (class in earthdiagnostics.variable), 42
VerticalMean (class in earthdiagnostics.ocean.verticalmean), 60
VerticalMeanMeters (class in earthdiagnostics.ocean.verticalmeanmeters), 60
Vilkitsky_Strait (earthdiagnostics.constants.Basins attribute), 33

W

Weddell_Sea (earthdiagnostics.constants.Basins attribute), 33
Western_Central_Arctic (earthdiagnostics.constants.Basins attribute), 33

Y

YearlyMean (class in earthdiagnostics.general.yearlymean), 47