

---

# **Earth Diagnostics Documentation**

***Release 3.0.0b20***

**BSC-CNS Earth Sciences Department**

Nov 08, 2016



## CONTENTS

<b>1</b>	<b>Tutorial</b>	<b>1</b>
1.1	Installation . . . . .	1
1.2	Creating a config file . . . . .	1
1.3	Prepare the run script . . . . .	2
<b>2</b>	<b>Diagnostic list</b>	<b>3</b>
2.1	General . . . . .	3
2.2	Ocean . . . . .	3
2.3	Statistics . . . . .	4
<b>3</b>	<b>Tips and tricks</b>	<b>5</b>
3.1	Working with ORCA1 . . . . .	5
3.2	Configuring core usage . . . . .	5
3.3	NEMO files . . . . .	5
<b>4</b>	<b>What to do if you have an error</b>	<b>7</b>
<b>5</b>	<b>Developer’s guide</b>	<b>9</b>
5.1	Developing a diagnostic . . . . .	9
<b>6</b>	<b>Frequently Asked Questions</b>	<b>11</b>
<b>7</b>	<b>Module documentation</b>	<b>13</b>
7.1	earthdiagnostics . . . . .	13
7.1.1	earthdiagnostics.box . . . . .	13
7.1.2	earthdiagnostics.cdftools . . . . .	14
7.1.3	earthdiagnostics.cmorizer . . . . .	14
7.1.4	earthdiagnostics.cmormanager . . . . .	14
7.1.5	earthdiagnostics.config . . . . .	17
7.1.6	earthdiagnostics.constants . . . . .	18
7.1.7	earthdiagnostics.datamanager . . . . .	22
7.1.8	earthdiagnostics.diagnostic . . . . .	25
7.1.9	earthdiagnostics.earthdiags . . . . .	26
7.1.10	earthdiagnostics.parser . . . . .	26
7.1.11	earthdiagnostics.utils . . . . .	29
7.1.12	earthdiagnostics.variable . . . . .	33
7.2	earthdiagnostics.general . . . . .	33
7.2.1	earthdiagnostics.general.monthlymean . . . . .	33
7.2.2	earthdiagnostics.ocean.relink . . . . .	34
7.2.3	earthdiagnostics.ocean.rewrite . . . . .	35
7.2.4	earthdiagnostics.ocean.scale . . . . .	35

7.3	earthdiagnostics.ocean . . . . .	36
7.3.1	earthdiagnostics.ocean.areamoc . . . . .	36
7.3.2	earthdiagnostics.ocean.averagesection . . . . .	37
7.3.3	earthdiagnostics.ocean.convectionsites . . . . .	37
7.3.4	earthdiagnostics.ocean.cutsection . . . . .	38
7.3.5	earthdiagnostics.ocean.gyres . . . . .	39
7.3.6	earthdiagnostics.ocean.heatcontent . . . . .	40
7.3.7	earthdiagnostics.ocean.heatcontentlayer . . . . .	40
7.3.8	earthdiagnostics.ocean.interpolate . . . . .	41
7.3.9	earthdiagnostics.ocean.interpolatecdo . . . . .	42
7.3.10	earthdiagnostics.ocean.maxmoc . . . . .	43
7.3.11	earthdiagnostics.ocean.mixedlayerheatcontent . . . . .	43
7.3.12	earthdiagnostics.ocean.mixedlayersaltcontent . . . . .	44
7.3.13	earthdiagnostics.ocean.moc . . . . .	45
7.3.14	earthdiagnostics.ocean.psi . . . . .	45
7.3.15	earthdiagnostics.ocean.siasiesiv . . . . .	46
7.3.16	earthdiagnostics.ocean.verticalmean . . . . .	47
7.3.17	earthdiagnostics.ocean.verticalmeanmeters . . . . .	47
7.4	earthdiagnostics.statistics . . . . .	48
7.4.1	earthdiagnostics.statistics.climatologicalpercentile . . . . .	48

<b>Python Module Index</b>	<b>51</b>
----------------------------	-----------

<b>Index</b>	<b>53</b>
--------------	-----------

## 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](mailto:javier.vegas@bsc.es)> so it can be corrected. A lot of people will benefit from it.

---

### 1.1 Installation

For now, you only have one option: download the diagnostics directly from BSC-ES's Gitlab:

```
git clone https://earth.bsc.es/gitlab/es/ocean_diagnostics.git
```

You will also need

- CDO version 1.6.9 (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 Autosubmit, CDO and NCO packages, among others. A virtual environment with all requisites fulfilled is available at `/shared/earth/ClimatePrediction/EarthDiagnostics`
- Access to CDFTOOLS\_3.0 executables for BSC-ES. At this point, those are located at `/shared/earth/ClimatePrediction/CDFTOOLS_CMOR/bin`.

### 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 suits 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 bsclogin01, 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.

## 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
```

### 2.1 General

- **monmean:** Calculates the monthly mean of the given variable. See [MonthlyMean](#)
- **relink:** Regenerates the links created in the `monthly_mean`, `daily_mean`, folders. See [Relink](#)
- **rewrite:** Just rewrites the CMOR output of a given variable. Useful to correct metadata or variable units. See [Rewrite](#)
- **scale:** Scales a given variable using a given scale factor and offset. Useful to correct erros on the data. See [Scale](#)

### 2.2 Ocean

- **areamoc:** Compute an Atlantic MOC index. See [AreaMoc](#)
- **averagesection:** Compute an average of a given zone. The variable MUST be in a regular grid See [AverageSection](#)
- **convectionsites:** Compute the intensity of convection in the four main convection sites. See [ConvectionSites](#)
- **cutsection:** Cuts a meridional or zonal section. See [CutSection](#)
- **gyres:** Compute the intensity of the subtropical and subpolar gyres. See [Gyres](#)
- **heatcontent:** Compute the total ocean heat content. See [HeatContent](#)
- **heatcontentlayer:** Point-wise Ocean Heat Content in a specified ocean thickness. See [HeatContentLayer](#)
- **interpolate:** 3-dimensional conservative interpolation to the regular atmospheric grid. It can also be used for 2D (i,j) variables. See [Interpolate](#)
- **interpolateCDO:** Bilinear interpolation to a given grid using CDO. See [InterpolateCDO](#)
- **maxmoc:** Compute an Atlantic MOC index by finding the maximum of the annual mean meridional overturning in a latitude / depth region See [MaxMoc](#)

- **mixedlayerheatcontent:** Compute mixed layer heat content. See [MixedLayerHeatContent](#)
- **mixedlayersaltcontent:** Compute mixed layer salt content. See [MixedLayerSaltContent](#)
- **moc:** Compute the MOC for oceanic basins. See [Moc](#)
- **psi:** Compute the barotropic stream function. See [Psi](#)
- **siasiesiv:** Compute the sea ice extent , area and volume in both hemispheres or a specified region. See [Siasiesiv](#)
- **verticalmean:** Chooses vertical level in ocean, or vertically averages between 2 or more ocean levels. See [VerticalMean](#)
- **verticalmeanmeters:** Averages vertically any given variable. See [VerticalMeanMeters](#)

## 2.3 Statistics

- **climpercent:** Calculates the specified climatological percentile of a given variable. See [ClimatologicalPercentile](#)
- **monpercent:** Calculates the specified monthly percentile of a given variable. See [MonthlyPercentile](#)



## TIPS AND TRICKS

### 3.1 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.

### 3.2 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.

### 3.3 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.



## 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 is 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](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.



## 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/](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](mailto:javier.vegas@bsc.es).

### 5.1 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.



## **FREQUENTLY ASKED QUESTIONS**

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





## MODULE DOCUMENTATION

### 7.1 earthdiagnostics

#### 7.1.1 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 the 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

### 7.1.2 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)

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

### 7.1.3 earthdiagnostics.cmorizer

**class** earthdiagnostics.cmorizer.Cmorizer (*data\_manager*, *startdate*, *member*)

Bases: `object`

Class to manage CMORization

**Parameters**

- **data\_manager** (*DataManager*) – 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** (*netCDF\$.Dataset*) – netCDF4 handler for the file
- **frequency** (*str*) – variable's frequency
- **variable** (*str*) – variable's name

### 7.1.4 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*, *var-type=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** (*str*) – file's frequency (only needed if it is different from the default)

**Returns** path to the copy created on the scratch folder

#### Return type *str*

**get\_file\_path** (*startdate*, *member*, *domain*, *var*, *chunk*, *frequency*, *box=None*, *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: str :param chunk: file's chunk :type chunk: int :param frequency: file's frequency :type frequency: str :param box: file's box :type box: Box :param grid: file's grid :type grid: str :param year: file's year :type year: int :param date\_str: date string to add directly. Overrides year or chunk configurations :type date\_str: str :return: path to the file :rtype: str

**get\_year** (*domain*, *var*, *startdate*, *member*, *year*, *grid=None*, *box=None*)

Get 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*, *startdate*, *member*, *chunk=None*, *grid=None*, *box=None*, *frequency=None*, *year=None*, *date\_str=None*, *move\_old=False*)

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)
- **box** (*Box*) – file's box (only needed to retrieve sections or averages)

- **frequency** (*str*) – file’s frequency (only needed if it is different from the default)

**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** (*str*) – 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

**Type** *str*

### 7.1.5 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

**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

**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]

## 7.1.6 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

**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 behaviour 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
```

### 7.1.7 earthdiagnostics.datamanager

**class** earthdiagnostics.datamanager.**DataManager** (*config*)

Bases: `object`

Class to manage the data repositories

**Parameters** *config* (`Config`) –

**get\_file** (*domain*, *var*, *startdate*, *member*, *chunk*, *grid=None*, *box=None*, *frequency=None*, *var-type=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** (*str*) – file's frequency (only needed if it is different from the default)

**Returns** path to the copy created on the scratch folder

**Return type** `str`

**get\_year** (*domain, var, startdate, member, year, grid=None, box=None*)

Get 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, startdate, member, chunk=None, grid=None, box=None, frequency=None, year=None, date\_str=None, move\_old=False*)

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)
- **box** (*Box*) – file's box (only needed to retrieve sections or averages)
- **frequency** (*str*) – file's frequency (only needed if it is different from the default)

**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** (*str*) – 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

Type *str*

**class** earthdiagnostics.datamanager.**NetCDFFile** (*remote\_file*, *local\_file*, *domain*, *var*,  
*cmor\_var*)

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

## 7.1.8 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 classes

**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** (`Domain`) –
- **var** –
- **startdate** –
- **member** –
- **chunk** –
- **grid** –
- **region** –
- **box** –

- **rename\_var** –
- **frequency** –
- **year** –
- **date\_str** –
- **move\_old** –

**Returns**

### 7.1.9 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

### 7.1.10 earthdiagnostics.parser

**class** earthdiagnostics.parser.**Parser** (*defaults=None*, *dict\_type=<class 'collections.OrderedDict'>*, *allow\_no\_value=False*)

Bases: `ConfigParser.SafeConfigParser`

Class to manage the config file. It add options to manage default values and to convert strings to the desired types (int, bool, list ...)

**add\_section** (*section*)

Create a new section in the configuration.

Raise DuplicateSectionError if a section by the specified name already exists. Raise ValueError if name is DEFAULT or any of it's case-insensitive variants.

**check\_exists** (*section, option*)

Checks if an option exists

**Parameters**

- **section** (*str*) – section that contains the option
- **option** (*str*) – option to check

**Returns** True if option exists, False otherwise

**Return type** `bool`

**check\_is\_boolean** (*section, option, must\_exist*)

Checks if an option is a boolean value

**Parameters**

- **section** (*str*) – section that contains the option
- **option** (*str*) – option to check

- **must\_exist** (*bool*) – if True, option must exist

**Returns** True if option value is boolean, False otherwise

**Return type** *bool*

**check\_is\_choice** (*section, option, must\_exist, choices*)

Checks if an option is a valid choice in given self

**Parameters**

- **section** (*str*) – section that contains the option
- **option** (*str*) – option to check
- **must\_exist** (*bool*) – if True, option must exist
- **choices** (*list*) – valid choices

**Returns** True if option value is a valid choice, False otherwise

**Return type** *bool*

**check\_is\_int** (*section, option, must\_exist*)

Checks if an option is an integer value

**Parameters**

- **section** (*str*) – section that contains the option
- **option** (*str*) – option to check
- **must\_exist** (*bool*) – if True, option must exist

**Returns** True if option value is integer, False otherwise

**Return type** *bool*

**check\_regex** (*section, option, must\_exist, regex*)

Checks if an option complies with a regular expression

**Parameters**

- **section** (*str*) – section that contains the option
- **option** (*str*) – option to check
- **must\_exist** (*bool*) – if True, option must exist
- **regex** (*str*) – regular expression to check

**Returns** True if option complies with regex, False otherwise

**Return type** *bool*

**get** (*section, option, raw=False, vars=None*)

Get an option value for a given section.

If ‘vars’ is provided, it must be a dictionary. The option is looked up in ‘vars’ (if provided), ‘section’, and in ‘defaults’ in that order.

All % interpolations are expanded in the return values, unless the optional argument ‘raw’ is true. Values for interpolation keys are looked up in the same manner as the option.

The section DEFAULT is special.

**get\_bool\_option** (*section, option, default=True*)

Gets a boolean option

**Parameters**

- **section** (*str*) – section that contains the option
- **option** (*str*) – option to get
- **default** (*bool*) – value to be returned if option is not present

**Returns** option value

**Return type** *bool*

**get\_float\_option** (*section, option, default=0.0*)

Gets a float option

**Parameters**

- **section** (*str*) – section that contains the option
- **option** (*str*) – option to get
- **default** (*float*) – value to be returned if option is not present

**Returns** option value

**Return type** *float*

**get\_int\_option** (*section, option, default=0*)

Gets an integer option

**Parameters**

- **section** (*str*) – section that contains the option
- **option** (*str*) – option to get
- **default** (*int*) – value to be returned if option is not present

**Returns** option value

**Return type** *int*

**get\_list\_option** (*section, option, default=[], separator=' '*)

Gets a list option

**Parameters**

- **section** (*str*) – section that contains the option
- **option** (*str*) – option to get
- **default** (*object*) – value to be returned if option is not present
- **separator** (*str*) – separator used to split the list

**Returns** option value

**Return type** *list*

**get\_option** (*section, option, default=''*)

Gets an option

**Parameters**

- **section** (*str*) – section that contains the option
- **option** (*str*) – option to get
- **default** (*object*) – value to be returned if option is not present



**Returns** option value

**Return type** `str`

**has\_option** (*section*, *option*)

Check for the existence of a given option in a given section.

**has\_section** (*section*)

Indicate whether the named section is present in the configuration.

The DEFAULT section is not acknowledged.

**items** (*section*, *raw=False*, *vars=None*)

Return a list of tuples with (name, value) for each option in the section.

All % interpolations are expanded in the return values, based on the defaults passed into the constructor, unless the optional argument 'raw' is true. Additional substitutions may be provided using the 'vars' argument, which must be a dictionary whose contents overrides any pre-existing defaults.

The section DEFAULT is special.

**options** (*section*)

Return a list of option names for the given section name.

**read** (*filenames*)

Read and parse a filename or a list of filenames.

Files that cannot be opened are silently ignored; this is designed so that you can specify a list of potential configuration file locations (e.g. current directory, user's home directory, systemwide directory), and all existing configuration files in the list will be read. A single filename may also be given.

Return list of successfully read files.

**readfp** (*fp*, *filename=None*)

Like read() but the argument must be a file-like object.

The 'fp' argument must have a 'readline' method. Optional second argument is the 'filename', which if not given, is taken from fp.name. If fp has no 'name' attribute, '<???' is used.

**remove\_option** (*section*, *option*)

Remove an option.

**remove\_section** (*section*)

Remove a file section.

**sections** ()

Return a list of section names, excluding [DEFAULT]

**set** (*section*, *option*, *value=None*)

Set an option. Extend ConfigParser.set: check for string values.

**write** (*fp*)

Write an .ini-format representation of the configuration state.

## 7.1.11 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.Uutils`

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*

Exception 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 if 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 if 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)`

Creates a folder path with all parent directories if needed. :param path: folder's path :type path: str

**static** `Utils.execute_shell_command(command, log_level=10)`

Executes a shell 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.expand_path(path)`

Expands character ~ and system variables on the given path :param path: path to expand :type path: str :return: path after the expansion

**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 :param force: if True, it will overwrite unzipped files :type force: bool

## 7.1.12 earthdiagnostics.variable

**class** `earthdiagnostics.variable.Variable(line)`

Bases: `object`

Class to characterize a CMOR variable. It also contains the static method to make the match between the original name and the standard name. Requires `cmor_table.csv` to work.

**classmethod** `get_variable(original_name, silent=False)`

Returns the cmor variable instance given a variable name

### Parameters

- **original\_name** (*str*) – original variable's name
- **silent** (*bool*) – if True, omits log warning when variable is not found

**Returns** CMOR variable

**Return type** *Variable*

**classmethod** `load_variables()`

Loads the `cmor_table.csv` and creates the variables dictionary

## 7.2 earthdiagnostics.general

### 7.2.1 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](mailto: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** (*Domain*) – 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

## 7.2.2 earthdiagnostics.ocean.relink

**class** earthdiagnostics.general.relink.**Relink** (*data\_manager, startdate, member, chunk, domain, variable, move\_old*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Recreates the links for the variable specified

**Original author** Javier Vegas-Regidor<[javier.vegas@bsc.es](mailto: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** (*Domain*) – 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

### 7.2.3 earthdiagnostics.ocean.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](mailto: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** (*Domain*) – 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

### 7.2.4 earthdiagnostics.ocean.scale

**class** earthdiagnostics.general.scale.**Scale**(*data\_manager, startdate, member, chunk, value, offset, domain, variable, grid, min\_limit, max\_limit*)

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](mailto: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** (*Domain*) – 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

## 7.3 earthdiagnostics.ocean

### 7.3.1 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**

### 7.3.2 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** (*Domain*) – variable’s domain
- **box** (*Box*) – box to use for the average

**alias** = ‘avgsection’

Dagnostic 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**

### 7.3.3 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’

Dagnostic 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**

### 7.3.4 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** (*str*) – 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

### 7.3.5 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

### 7.3.6 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’

Dagnostic 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

### 7.3.7 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](mailto: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

### 7.3.8 earthdiagnostics.ocean.interpolate

```
class earthdiagnostics.ocean.interpolate.Interpolate(data_manager, startdate, member, chunk, domain, variable, target_grid, model_version, invert_lat)
```

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](mailto:virginie.guemas@bsc.es)>

**Contributor** Javier Vegas-Regidor<[javier.vegas@bsc.es](mailto: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** (*str*) – variable’s domain
- **model\_version** (*str*) – model version

**alias = 'interp'**

Dagnostic 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**

### 7.3.9 earthdiagnostics.ocean.interpolatecdo

**class** earthdiagnostics.ocean.interpolatecdo.**InterpolateCDO** (*data\_manager, startdate, member, chunk, domain, variable, target\_grid, model\_version*)

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](mailto: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** (*Domain*) – variable's domain
- **model\_version** (*str*) – model version

**alias = 'interpcdo'**

Dagnostic 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**

### 7.3.10 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

### 7.3.11 earthdiagnostics.ocean.mixedlayerheatcontent

**class** earthdiagnostics.ocean.mixedlayerheatcontent.**MixedLayerHeatContent** (*data\_manager*,  
*start-date*,  
*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**

### 7.3.12 earthdiagnostics.ocean.mixedlayersaltcontent

```
class earthdiagnostics.ocean.mixedlayersaltcontent.MixedLayerSaltContent (data_manager,
                                                                              start-
                                                                              date,
                                                                              mem-
                                                                              ber,
                                                                              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**

### 7.3.13 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**

### 7.3.14 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**

### 7.3.15 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**

### 7.3.16 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

### 7.3.17 earthdiagnostics.ocean.verticalmeanmeters

**class** earthdiagnostics.ocean.verticalmeanmeters.**VerticalMeanMeters** (*data\_manager, startdate, member, chunk, 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

## 7.4 earthdiagnostics.statistics

### 7.4.1 earthdiagnostics.statistics.climatologicalpercentile

**class** earthdiagnostics.statistics.climatologicalpercentile.**ClimatologicalPercentile** (*data\_manager, domain, main, variable, lead-times, 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**

earthdiagnostics.statistics.monthlypercentile \_\_\_\_\_ \_ .. automodule:: earthdiagnostics.statistics.monthlypercentile

**show-inheritance**

**members**



**e**

- earthdiagnostics.box, 13
- earthdiagnostics.cdftools, 14
- earthdiagnostics.cmorizer, 14
- earthdiagnostics.cmormanager, 14
- earthdiagnostics.config, 17
- earthdiagnostics.constants, 18
- earthdiagnostics.datamanager, 22
- earthdiagnostics.diagnostic, 25
- earthdiagnostics.earthdiags, 26
- earthdiagnostics.general.monthlymean, 33
- earthdiagnostics.general.relink, 34
- earthdiagnostics.general.rewrite, 35
- earthdiagnostics.general.scale, 35
- earthdiagnostics.ocean.areamoc, 36
- earthdiagnostics.ocean.averagesection, 37
- earthdiagnostics.ocean.convectionsites, 37
- earthdiagnostics.ocean.cutsection, 38
- earthdiagnostics.ocean.gyres, 39
- earthdiagnostics.ocean.heatcontent, 40
- earthdiagnostics.ocean.heatcontentlayer, 40
- earthdiagnostics.ocean.interpolate, 41
- earthdiagnostics.ocean.interpolatecd, 42
- earthdiagnostics.ocean.maxmoc, 43
- earthdiagnostics.ocean.mixedlayerheatcontent, 43
- earthdiagnostics.ocean.mixedlayersaltcontent, 44
- earthdiagnostics.ocean.moc, 45
- earthdiagnostics.ocean.psi, 45
- earthdiagnostics.ocean.siasiesiv, 46
- earthdiagnostics.ocean.verticalmean, 47
- earthdiagnostics.ocean.verticalmeanmeters, 47
- earthdiagnostics.parser, 26
- earthdiagnostics.statistics.climatologicalpercentile, 48
- earthdiagnostics.utils, 29
- earthdiagnostics.variable, 33





# A

add\_conversion() (earthdiagnostics.datamanager.UnitConversion class method), 24  
 add\_section() (earthdiagnostics.parser.Parser method), 26  
 alias (earthdiagnostics.diagnostic.Diagnostic attribute), 25  
 alias (earthdiagnostics.general.monthlymean.MonthlyMean attribute), 34  
 alias (earthdiagnostics.general.relink.Relink attribute), 34  
 alias (earthdiagnostics.general.rewrite.Rewrite attribute), 35  
 alias (earthdiagnostics.general.scale.Scale attribute), 36  
 alias (earthdiagnostics.ocean.areamoc.AreaMoc attribute), 36  
 alias (earthdiagnostics.ocean.averagesection.AverageSection attribute), 37  
 alias (earthdiagnostics.ocean.convectionsites.ConvectionSites attribute), 38  
 alias (earthdiagnostics.ocean.cutsection.CutSection attribute), 39  
 alias (earthdiagnostics.ocean.gyres.Gyres attribute), 39  
 alias (earthdiagnostics.ocean.heatcontent.HeatContent attribute), 40  
 alias (earthdiagnostics.ocean.heatcontentlayer.HeatContentLayer attribute), 41  
 alias (earthdiagnostics.ocean.interpolate.Interpolate attribute), 41  
 alias (earthdiagnostics.ocean.interpolatecdo.InterpolateCDO attribute), 42  
 alias (earthdiagnostics.ocean.maxmoc.MaxMoc attribute), 43  
 alias (earthdiagnostics.ocean.mixedlayerheatcontent.MixedLayerHeatContent attribute), 44  
 alias (earthdiagnostics.ocean.mixedlayersaltcontent.MixedLayerSaltContent attribute), 44  
 alias (earthdiagnostics.ocean.moc.Moc attribute), 45  
 alias (earthdiagnostics.ocean.psi.Psi attribute), 46  
 alias (earthdiagnostics.ocean.siasiesiv.Siasiesiv attribute), 46  
 alias (earthdiagnostics.ocean.verticalmean.VerticalMean attribute), 47  
 alias (earthdiagnostics.ocean.verticalmeanmeters.VerticalMeanMeters attribute), 48  
 alias (earthdiagnostics.statistics.climatologicalpercentile.ClimatologicalPercentile attribute), 48  
 Antarctic (earthdiagnostics.constants.Basins attribute), 18  
 AntarcticAtlantic (earthdiagnostics.constants.Basins attribute), 18  
 AntarcticIndian (earthdiagnostics.constants.Basins attribute), 18  
 Arctic (earthdiagnostics.constants.Basins attribute), 18  
 ArcticMarginalSeas (earthdiagnostics.constants.Basins attribute), 18  
 ArcticNorthAtlantic (earthdiagnostics.constants.Basins attribute), 18  
 AreaMoc (class in earthdiagnostics.ocean.areamoc), 36  
 Atlantic (earthdiagnostics.constants.Basins attribute), 18  
 autoclean (earthdiagnostics.utils.TempFile attribute), 29  
 available\_cpu\_count() (earthdiagnostics.utils.Utils static method), 30  
 AverageSection (class in earthdiagnostics.ocean.averagesection), 37

# B

Baffin (earthdiagnostics.constants.Basins attribute), 18  
 Baffin\_Bay (earthdiagnostics.constants.Basins attribute), 18  
 Baltic\_Sea (earthdiagnostics.constants.Basins attribute), 18  
 Barents\_Sea (earthdiagnostics.constants.Basins attribute), 19  
 BarKara (earthdiagnostics.constants.Basins attribute), 19  
 Basin (class in earthdiagnostics.constants), 18  
 BasinHeadCenter (earthdiagnostics.constants), 18  
 Beaufort\_Chukchi\_Sea (earthdiagnostics.constants.Basins attribute), 19  
 Beaufort\_Sea (earthdiagnostics.constants.Basins attribute), 19  
 Bellingshausen\_Sea (earthdiagnostics.constants.Basins attribute), 19  
 Bering (earthdiagnostics.constants.Basins attribute), 19  
 Bering\_Strait (earthdiagnostics.constants.Basins attribute), 19

Box (class in earthdiagnostics.box), 13

box (earthdiagnostics.constants.Basin attribute), 18

## C

Canadian\_Waters (earthdiagnostics.constants.Basins attribute), 19

CanArch (earthdiagnostics.constants.Basins attribute), 19

Caspian\_Sea (earthdiagnostics.constants.Basins attribute), 19

CDFTools (class in earthdiagnostics.cdfutils), 14

cdfutils\_path (earthdiagnostics.config.Config attribute), 17

cdo (earthdiagnostics.utils.Utils attribute), 30

Central\_Arctic (earthdiagnostics.constants.Basins attribute), 19

check\_exists() (earthdiagnostics.parser.Parser method), 26

check\_is\_boolean() (earthdiagnostics.parser.Parser method), 26

check\_is\_choice() (earthdiagnostics.parser.Parser method), 27

check\_is\_int() (earthdiagnostics.parser.Parser method), 27

check\_regex() (earthdiagnostics.parser.Parser method), 27

Chukchi\_Sea (earthdiagnostics.constants.Basins attribute), 19

clean() (earthdiagnostics.utils.TempFile static method), 29

ClimatologicalPercentile (class in earthdiagnostics.statistics.climatologicalpercentile), 48

cmorize\_atmos() (earthdiagnostics.cmorizer.Cmorizer method), 14

cmorize\_ocean() (earthdiagnostics.cmorizer.Cmorizer method), 14

Cmorizer (class in earthdiagnostics.cmorizer), 14

CMORManager (class in earthdiagnostics.cmormanager), 14

compute() (earthdiagnostics.diagnostic.Diagnostic method), 25

compute() (earthdiagnostics.general.monthlymean.MonthlyMean method), 34

compute() (earthdiagnostics.general.relink.Relink method), 34

compute() (earthdiagnostics.general.rewrite.Rewrite method), 35

compute() (earthdiagnostics.general.scale.Scale method), 36

compute() (earthdiagnostics.ocean.areamoc.AreaMoc method), 36

compute() (earthdiagnostics.ocean.averagesection.AverageSection method), 37

compute() (earthdiagnostics.ocean.convectionsites.ConvectionSites method), 38

compute() (earthdiagnostics.ocean.cutsection.CutSection method), 39

compute() (earthdiagnostics.ocean.gyres.Gyres method), 39

compute() (earthdiagnostics.ocean.heatcontent.HeatContent method), 40

compute() (earthdiagnostics.ocean.heatcontentlayer.HeatContentLayer method), 41

compute() (earthdiagnostics.ocean.interpolate.Interpolate method), 42

compute() (earthdiagnostics.ocean.interpolatecdo.InterpolateCDO method), 42

compute() (earthdiagnostics.ocean.maxmoc.MaxMoc method), 43

compute() (earthdiagnostics.ocean.mixedlayerheatcontent.MixedLayerHeatContent method), 44

compute() (earthdiagnostics.ocean.mixedlayersaltcontent.MixedLayerSaltContent method), 44

compute() (earthdiagnostics.ocean.moc.Moc method), 45

compute() (earthdiagnostics.ocean.psi.Psi method), 46

compute() (earthdiagnostics.ocean.siasiesiv.Siasiesiv method), 46

compute() (earthdiagnostics.ocean.verticalmean.VerticalMean method), 47

compute() (earthdiagnostics.ocean.verticalmeanmeters.VerticalMeanMeters method), 48

compute() (earthdiagnostics.statistics.climatologicalpercentile.ClimatologicalPercentile method), 48

con\_files (earthdiagnostics.config.Config attribute), 17

concat\_variables() (earthdiagnostics.utils.Utils static method), 30

Config (class in earthdiagnostics.config), 17

ConvectionSites (class in earthdiagnostics.ocean.convectionsites), 37

convert2netcdf4() (earthdiagnostics.utils.Utils static method), 30

copy\_dimension() (earthdiagnostics.utils.Utils static method), 30

copy\_file() (earthdiagnostics.utils.Utils static method), 31

copy\_variable() (earthdiagnostics.utils.Utils static method), 31

create\_folder\_tree() (earthdiagnostics.utils.Utils static method), 31

CutSection (class in earthdiagnostics.ocean.cutsection), 38

## D

data\_adaptor (earthdiagnostics.config.Config attribute), 17

data\_dir (earthdiagnostics.config.Config attribute), 17

DataManager (class in earthdiagnostics.datamanager), 22

depth\_in\_meters (earthdiagnostics.box.Box attribute), 13

Diagnostic (class in earthdiagnostics.diagnostic), 25

## E

earthdiagnostics.box (module), 13

earthdiagnostics.cdfutils (module), 14

earthdiagnostics.cmorizer (module), 14

earthdiagnostics.cmormanager (module), 14

earthdiagnostics.config (module), 17

earthdiagnostics.constants (module), 18

earthdiagnostics.datamanager (module), 22

earthdiagnostics.diagnostic (module), 25

earthdiagnostics.earthdiags (module), 26

earthdiagnostics.general.monthlymean (module), 33

earthdiagnostics.general.relink (module), 34

earthdiagnostics.general.rewrite (module), 35

earthdiagnostics.general.scale (module), 35

earthdiagnostics.ocean.areamoc (module), 36

earthdiagnostics.ocean.averagesection (module), 37

earthdiagnostics.ocean.convectionsites (module), 37

earthdiagnostics.ocean.cutsection (module), 38

earthdiagnostics.ocean.gyres (module), 39

earthdiagnostics.ocean.heatcontent (module), 40

earthdiagnostics.ocean.heatcontentlayer (module), 40

earthdiagnostics.ocean.interpolate (module), 41

earthdiagnostics.ocean.interpolatecd (module), 42

earthdiagnostics.ocean.maxmoc (module), 43  
earthdiagnostics.ocean.mixedlayerheatcontent (module), 43

earthdiagnostics.ocean.mixedlayersaltcontent (module), 44

earthdiagnostics.ocean.moc (module), 45

earthdiagnostics.ocean.psi (module), 45

earthdiagnostics.ocean.siasiesiv (module), 46

earthdiagnostics.ocean.verticalmean (module), 47

earthdiagnostics.ocean.verticalmeanmeters (module), 47

earthdiagnostics.parser (module), 26

earthdiagnostics.statistics.climatologicalpercentile (module), 48

earthdiagnostics.utils (module), 29

earthdiagnostics.variable (module), 33

EarthDiags (class in earthdiagnostics.earthdiags), 26

East\_Siberian\_Sea (earthdiagnostics.constants.Basins attribute), 19

Eastern\_Central\_Arctic (earthdiagnostics.constants.Basins attribute), 19

ECEARTH\_2\_3\_O1L42 (earthdiagnostics.constants.Models attribute), 21

ECEARTH\_3\_0\_O1L46 (earthdiagnostics.constants.Models attribute), 21

ECEARTH\_3\_0\_O25L46 (earthdiagnostics.constants.Models attribute), 21

ECEARTH\_3\_0\_O25L75 (earthdiagnostics.constants.Models attribute), 22

ECEARTH\_3\_1\_O25L75 (earthdiagnostics.constants.Models attribute), 22

ECEARTH\_3\_2\_O1L75 (earthdiagnostics.constants.Models attribute), 22

execute\_shell\_command() (earthdiagnostics.utils.Utils static method), 31

expand\_path() (earthdiagnostics.utils.Utils static method), 31

experiment (earthdiagnostics.config.Config attribute), 17

ExperimentConfig (class in earthdiagnostics.config), 17

extract\_variable() (earthdiagnostics.cmorizer.Cmorizer method), 14

## F

files (earthdiagnostics.utils.TempFile attribute), 30

Fram\_Strait (earthdiagnostics.constants.Basins attribute), 19

frequency (earthdiagnostics.config.Config attribute), 17

fullname (earthdiagnostics.constants.Basin attribute), 18

## G

generate\_jobs() (earthdiagnostics.diagnostic.Diagnostic class method), 25

generate\_jobs() (earthdiagnostics.general.monthlymean.MonthlyMean class method), 34

generate\_jobs() (earthdiagnostics.general.relink.Relink class method), 34

generate\_jobs() (earthdiagnostics.general.rewrite.Rewrite class method), 35

generate\_jobs() (earthdiagnostics.general.scale.Scale class method), 36

generate\_jobs() (earthdiagnostics.ocean.areamoc.AreaMoc class method), 36

generate\_jobs() (earthdiagnostics.ocean.averagesection.AverageSection class method), 37

generate\_jobs() (earthdiagnostics.ocean.convectionsites.ConvectionSites class method), 38

generate\_jobs() (earthdiagnostics.ocean.cutsection.CutSection class method), 39

generate\_jobs() (earthdiagnostics.ocean.gyres.Gyres class method), 39

generate\_jobs() (earthdiagnostics.ocean.heatcontent.HeatContent class method), 40  
 generate\_jobs() (earthdiagnostics.ocean.heatcontentlayer.HeatContentLayer class method), 41  
 generate\_jobs() (earthdiagnostics.ocean.interpolate.Interpolate class method), 42  
 generate\_jobs() (earthdiagnostics.ocean.interpolatecdo.InterpolateCDO class method), 42  
 generate\_jobs() (earthdiagnostics.ocean.maxmoc.MaxMoc class method), 43  
 generate\_jobs() (earthdiagnostics.ocean.mixedlayerheatcontent.MixedLayerHeatContent class method), 44  
 generate\_jobs() (earthdiagnostics.ocean.mixedlayersaltcontent.MixedLayerSaltContent class method), 44  
 generate\_jobs() (earthdiagnostics.ocean.moc.Moc class method), 45  
 generate\_jobs() (earthdiagnostics.ocean.psi.Psi class method), 46  
 generate\_jobs() (earthdiagnostics.ocean.siasiesiv.Siasiesiv class method), 46  
 generate\_jobs() (earthdiagnostics.ocean.verticalmean.VerticalMean class method), 47  
 generate\_jobs() (earthdiagnostics.ocean.verticalmeanmeters.VerticalMeanMeters class method), 48  
 generate\_jobs() (earthdiagnostics.statistics.climatologicalpercentile.ClimatologicalPercentile class method), 49  
 get() (earthdiagnostics.parser.Parser method), 27  
 get() (earthdiagnostics.utils.TempFile static method), 30  
 get\_bool\_option() (earthdiagnostics.parser.Parser method), 27  
 get\_chunk\_list() (earthdiagnostics.config.ExperimentConfig method), 17  
 get\_commands() (earthdiagnostics.config.Config method), 17  
 get\_conversion\_factor\_offset() (earthdiagnostics.datamanager.UnitConversion class method), 24  
 get\_datetime\_from\_netcdf() (earthdiagnostics.utils.Utils static method), 31  
 get\_depth\_str() (earthdiagnostics.box.Box method), 13  
 get\_diagnostic() (earthdiagnostics.diagnostic.Diagnostic static method), 25  
 get\_file() (earthdiagnostics.cmormanager.CMORManager method), 14  
 get\_file() (earthdiagnostics.datamanager.DataManager method), 22  
 get\_file\_hash() (earthdiagnostics.utils.Utils static method), 31  
 get\_file\_path() (earthdiagnostics.cmormanager.CMORManager method), 15  
 get\_float\_option() (earthdiagnostics.parser.Parser method), 28  
 get\_full\_years() (earthdiagnostics.config.ExperimentConfig method), 17  
 get\_int\_option() (earthdiagnostics.parser.Parser method), 28  
 get\_int\_str() (earthdiagnostics.box.Box method), 13  
 get\_list\_option() (earthdiagnostics.parser.Parser method), 28  
 get\_long\_str() (earthdiagnostics.box.Box method), 13  
 get\_mask() (earthdiagnostics.utils.Utils static method), 32  
 get\_member\_list() (earthdiagnostics.config.ExperimentConfig method), 17  
 get\_member\_str() (earthdiagnostics.config.ExperimentConfig method), 17  
 get\_option() (earthdiagnostics.parser.Parser method), 28  
 get\_variable() (earthdiagnostics.variable.Variable class method), 33  
 get\_year() (earthdiagnostics.cmormanager.CMORManager method), 15  
 get\_year() (earthdiagnostics.datamanager.DataManager method), 22  
 get\_year\_chunks() (earthdiagnostics.config.ExperimentConfig method), 17  
 Global\_Ocean (earthdiagnostics.constants.Basins attribute), 19  
 GLORYS2\_V1\_O25L75 (earthdiagnostics.constants.Models attribute), 22  
 Greenland\_Sea (earthdiagnostics.constants.Basins attribute), 19  
 Grnland (earthdiagnostics.constants.Basins attribute), 19  
 Gyres (class in earthdiagnostics.ocean.gyres), 39  
**H**  
 has\_option() (earthdiagnostics.parser.Parser method), 29  
 has\_section() (earthdiagnostics.parser.Parser method), 29  
 HeatContent (class in earthdiagnostics.ocean.heatcontent), 40  
 HeatContentLayer (class in earthdiagnostics.ocean.heatcontentlayer), 40  
 Hudson (earthdiagnostics.constants.Basins attribute), 19

## I

Icelandic\_Sea (earthdiagnostics.constants.Basins attribute), 19

Indian (earthdiagnostics.constants.Basins attribute), 20

IndoPacific (earthdiagnostics.constants.Basins attribute), 20

Interpolate (class in earthdiagnostics.ocean.interpolate), 41

InterpolateCDO (class in earthdiagnostics.ocean.interpolatecdo), 42

items() (earthdiagnostics.parser.Parser method), 29

## K

Kara\_Gate\_Strait (earthdiagnostics.constants.Basins attribute), 20

Kara\_Sea (earthdiagnostics.constants.Basins attribute), 20

## L

Labrador\_Sea (earthdiagnostics.constants.Basins attribute), 20

Laptev\_East\_Siberian\_Chukchi\_Seas (earthdiagnostics.constants.Basins attribute), 20

Laptev\_East\_Siberian\_Seas (earthdiagnostics.constants.Basins attribute), 20

Laptev\_Sea (earthdiagnostics.constants.Basins attribute), 20

Lincoln\_Sea (earthdiagnostics.constants.Basins attribute), 20

link\_file() (earthdiagnostics.cmormanager.CMORManager method), 15

link\_file() (earthdiagnostics.datamanager.DataManager method), 23

load\_conversions() (earthdiagnostics.datamanager.UnitConversion class method), 24

load\_variables() (earthdiagnostics.variable.Variable class method), 33

## M

max\_cores (earthdiagnostics.config.Config attribute), 17

max\_depth (earthdiagnostics.box.Box attribute), 13

max\_lat (earthdiagnostics.box.Box attribute), 13

max\_lon (earthdiagnostics.box.Box attribute), 13

MaxMoc (class in earthdiagnostics.ocean.maxmoc), 43

Mediterranean\_Sea (earthdiagnostics.constants.Basins attribute), 20

min\_depth (earthdiagnostics.box.Box attribute), 13

min\_lat (earthdiagnostics.box.Box attribute), 13

min\_lon (earthdiagnostics.box.Box attribute), 13

MixedLayerHeatContent (class in earthdiagnostics.ocean.mixedlayerheatcontent), 43

MixedLayerSaltContent (class in earthdiagnostics.ocean.mixedlayersaltcontent), 44

Moc (class in earthdiagnostics.ocean.moc), 45

Models (class in earthdiagnostics.constants), 21

MonthlyMean (class in earthdiagnostics.general.monthlymean), 33

move\_file() (earthdiagnostics.utils.Utils static method), 32

## N

Nares\_Strait (earthdiagnostics.constants.Basins attribute), 20

nco (earthdiagnostics.utils.Utils attribute), 32

NEMO\_3\_2\_O1L42 (earthdiagnostics.constants.Models attribute), 22

NEMO\_3\_3\_O1L46 (earthdiagnostics.constants.Models attribute), 22

NEMO\_3\_6\_O1L46 (earthdiagnostics.constants.Models attribute), 22

NEMOVAR\_O1L42 (earthdiagnostics.constants.Models attribute), 22

NetCDFFile (class in earthdiagnostics.datamanager), 24

Nordic\_Barents\_Seas (earthdiagnostics.constants.Basins attribute), 20

Nordic\_Seas (earthdiagnostics.constants.Basins attribute), 20

North\_Atlantic\_Arctic (earthdiagnostics.constants.Basins attribute), 20

North\_Hemisphere\_Ocean (earthdiagnostics.constants.Basins attribute), 20

NorthAtlantic (earthdiagnostics.constants.Basins attribute), 20

NorthPacific (earthdiagnostics.constants.Basins attribute), 20

NorthWest\_Passage (earthdiagnostics.constants.Basins attribute), 20

Norwegian\_Sea (earthdiagnostics.constants.Basins attribute), 20

## O

Okhotsk (earthdiagnostics.constants.Basins attribute), 20

openCdf() (earthdiagnostics.utils.Utils static method), 32

OpenOcean (earthdiagnostics.constants.Basins attribute), 20

options() (earthdiagnostics.parser.Parser method), 29

## P

Pacific (earthdiagnostics.constants.Basins attribute), 21

parse() (earthdiagnostics.constants.Basins class method), 21

parse\_args() (earthdiagnostics.earthdiags.EarthDiags static method), 26

Parser (class in earthdiagnostics.parser), 26

prefix (earthdiagnostics.utils.TempFile attribute), 30



prepare() (earthdiagnostics.cmormanager.CMORManager method), 16

prepare() (earthdiagnostics.datamanager.DataManager method), 23

Psi (class in earthdiagnostics.ocean.psi), 45

## R

read() (earthdiagnostics.parser.Parser method), 29

readfp() (earthdiagnostics.parser.Parser method), 29

register() (earthdiagnostics.diagnostic.Diagnostic static method), 25

Relink (class in earthdiagnostics.general.relink), 34

remove\_file() (earthdiagnostics.utils.Utils static method), 32

remove\_option() (earthdiagnostics.parser.Parser method), 29

remove\_section() (earthdiagnostics.parser.Parser method), 29

rename\_variable() (earthdiagnostics.utils.Utils static method), 32

rename\_variables() (earthdiagnostics.utils.Utils static method), 32

restore\_meshes (earthdiagnostics.config.Config attribute), 17

Rewrite (class in earthdiagnostics.general.rewrite), 35

Ross\_Sea (earthdiagnostics.constants.Basins attribute), 21

run() (earthdiagnostics.cdftools.CDFTools method), 14

run() (earthdiagnostics.earthdiags.EarthDiags method), 26

## S

Scale (class in earthdiagnostics.general.scale), 35

scratch\_dir (earthdiagnostics.config.Config attribute), 17

scratch\_folder (earthdiagnostics.utils.TempFile attribute), 30

sections() (earthdiagnostics.parser.Parser method), 29

send\_file() (earthdiagnostics.cmormanager.CMORManager method), 16

send\_file() (earthdiagnostics.datamanager.DataManager method), 23

send\_file() (earthdiagnostics.diagnostic.Diagnostic method), 25

Serreze\_Arctic (earthdiagnostics.constants.Basins attribute), 21

set() (earthdiagnostics.parser.Parser method), 29

setminmax() (earthdiagnostics.utils.Utils static method), 32

shortname (earthdiagnostics.constants.Basin attribute), 18

Siasiesiv (class in earthdiagnostics.ocean.siasiesiv), 46

Southern\_Hemisphere (earthdiagnostics.constants.Basins attribute), 21

StLawr (earthdiagnostics.constants.Basins attribute), 21

Subpolar\_Gyre (earthdiagnostics.constants.Basins attribute), 21

## T

TempFile (class in earthdiagnostics.utils), 29

TotalArc (earthdiagnostics.constants.Basins attribute), 21

TropicalAtlantic (earthdiagnostics.constants.Basins attribute), 21

TropicalIndian (earthdiagnostics.constants.Basins attribute), 21

TropicalPacific (earthdiagnostics.constants.Basins attribute), 21

## U

UnitConversion (class in earthdiagnostics.datamanager), 24

untar() (earthdiagnostics.utils.Utils static method), 32

unzip() (earthdiagnostics.utils.Utils static method), 33

Utils (class in earthdiagnostics.utils), 30

Utils.ExecutionError, 30

Utils.UnzipException, 30

## V

Variable (class in earthdiagnostics.variable), 33

VerticalMean (class in earthdiagnostics.ocean.verticalmean), 47

VerticalMeanMeters (class in earthdiagnostics.ocean.verticalmeanmeters), 47

Vilkitsky\_Strait (earthdiagnostics.constants.Basins attribute), 21

## W

Weddell\_Sea (earthdiagnostics.constants.Basins attribute), 21

Western\_Central\_Arctic (earthdiagnostics.constants.Basins attribute), 21

write() (earthdiagnostics.parser.Parser method), 29