ESMValTool-tutorial

November 29, 2013

1 Introduction

The ESMValtool is a software package used to compare and visualize climate data sets. The tool reads climate data from CMIP5 compliant netCDF files and generates output in postscript, pdf, png or netCDF. This document is a revised version of the ESMValTool tutorial given at the EMBRACE General Assembly in Exeter, June 2013.

For an overview of the tool, see the slides from the presentation given at the GA[5] or the '*doc/overview.pdf*' document. The tool is directly built upon the Chemistry-Climate-eValuation Tool[4] (CCMVal Tool) and the CCMVal reference can provide some background for the tool.

Outline: The remainder of this document is organized as follows: section 2 lists the ESMValTool tutorial prerequisites, section 3 gives an overview of the files and folders of the ESMValTool, section 4 details how to run the tool, section 5 goes through the simplified MyDiag example, Appendix A.1 lists the configuration files and scripts that are edited by the user in the tutorial '*Try out*'-blocks.

2 ESMVal prerequisites

The ESMValTool requirements are,

- compulsory: Python 2.x installation (versions 3.x will not work)

- compulsory: NCAR Command Language (NCL)[6] version 6.1 or higher. Google "install ncl" to find the NCAR page describing how to download and install NCL locally

- compulsory: CMIP5-style[8] netCDF data sets available on the file system. In practice, "CMIP5-style" means that some core netCDF attributes are required to comply with the CF-metadata conventions[7] and, to work work out of the box, the netCDF input file names should follow the CMIP5-naming convention[10]. Note that the tool can be configured to accept non-CMIP5 filenames and to rewrite non-CF compliant input files on the fly, this is described in the '*doc/reference.pdf*'-file. The '*tutorial.tar.gz*'[3] contains a minimal set of netCDF sufficient for running the examples mentioned in this document.

  Alternatively, for the better part of the exercises in this document it is enough to download the following two files from ESGF[1]

```
ta_Amon_MPI-ESM-LR_historical_r1i1p1_199001-199912.nc
ta_Amon_MPI-ESM-LR_historical_r1i1p1_200001-200512.nc
```

- optional: The R scripting language[9] version 2.14 or later (older version may work). R is not needed for the exercises described in this document, only if the R-based plot scripts are to be used. These plot scripts does in turn use the following R-packages,

  - ncdf4
  - fields
  - maps
  - MASS
  - chron

3 ESMValTool folder structure

In a fresh check out from the trunk[2] the following are the most important folders to be aware of,

```
ESMValTool - Earth System Model Validation Tool

For further help, check the doc/-folder for pdfs
and references therein

Important files and folders:
----------------------------

main.py              - Main script

doc/overview.pdf     - Start here if you're new to ESMValTool
doc/*                - Additional documentation for the tool

interface_scripts/   - General Python/NCL routines providing
                       core functionality to the tool
r_code/              - Additional R routines

nml/                 - Implemented namelists for the tool
nml/cfg_*/*          - Diagnostic configuration files

diag_scripts/        - Implemented diagnostics

variable_defs/       - Declaration of all variables and definition
                       of compound variables

reformat_scripts/    - NCL scripts for reformatting input data files

interface_data/      - Folder for temporary files used to pass in-
                       formation from main.py to the diagnostic
                       scripts
```

In addition to the above folders, the '*tutorial.tar.gz*'-tarball[3] includes a `Data/`-folder with the CMIP5 data set used to run the example in this document.

```
ta_Amon_MPI-ESM-LR_historical_r1i1p1_199001-199912.nc
ta_Amon_MPI-ESM-LR_historical_r1i1p1_200001-200512.nc
```

To set up the tutorial, issue,

```
$ svn co --username <USERNAME> <TRUNK-URL> <TARGET-FOLDER>
$ cp tutorial.tar.gz <TARGET-FOLDER>
$ cd  <TARGET-FOLDER>
$ tar xf tutorial.tar.gz
```

Alternatively, if the '*tutorial.tar.gz*' isn't available, downloading the following two files from ESGF[1] is sufficient for most exercises in this document,

```
ta_Amon_MPI-ESM-LR_historical_r1i1p1_199001-199912.nc
ta_Amon_MPI-ESM-LR_historical_r1i1p1_200001-200512.nc
```

Place these files in a local `Data/`-folder.

4 Running the ESMValTool

To run a session of ESMVal, issue

```
$ ./main.py nml/namelist.xml
```

where `namelist.xml` is any of the namelists in the `nml/`-folder. The Python main script will read the namelist configuration file, e.g., `nml/namelist.xml`, which contains sections for **(a)** global settings, **(b)** the models that should be included in this session, **(c)** a section listing the diagnostics to plot. These concepts are further explained in in the following sections.

5 The MyDiag template

For the purpose of the ESMValTool tutorial a stripped-down and simplified set of namelist/diagnostics have been prepared. These files have been designed to let the user play around with the basic features of the tool without requiring too much knowledge in neither Python, NCL or the tool design. To be easily recognizable the configuration files in the MyDiag template have been consistently named:

- nml/namelist_MyDiag.xml, see section 5.2
- variable_defs/MyVar.ncl, see section 5.4
- diag_scripts/MyDiag.ncl, see section 5.5

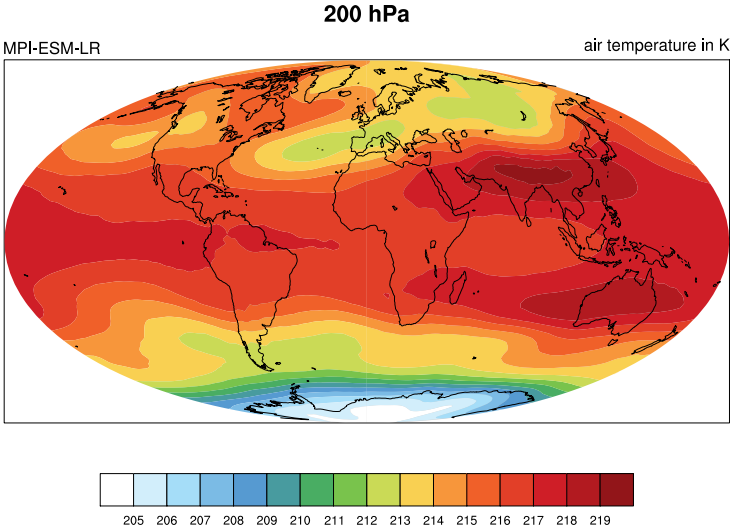

Figure 1: The MyDiag example run "out-of-the-box"

5.1 Running the MyDiag example

To run the MyDiag example, make sure the `Data/`-folder has been unpacked from the '*tutorial.tar.gz*'-tarball and is available in the same folder as the `main.py`-script[1]. Next, issue

```
$ ./main.py nml/namelist_MyDiag.xml
```

This will create a new folder specified by the `namelist_MyDiag.xml<plot_dir>`-tag containing a postscript figure '`MyDiag_MyVar.ps`', see figure 1. The following sections describes the configuration files/diagnostic scripts that generates this figure.

5.2 The MyDiag namelist

The main configuration file for the ESMValTool is the namelist, the important sections from the the MyDiag namelist are given below,

[1] Alternatively, use the manual download approach described at the end of section 3

```
<GLOBAL>
    <write_plots type="boolean">           True     </write_plots>
    <write_netcdf type="boolean">          True     </write_netcdf>
    <force_processing type="boolean">     False     </force_processing>
    <wrk_dir type="path">                  work/    </wrk_dir>
    <plot_dir type="path">                plots/    </plot_dir>
    <climo_dir type="path">               climo/    </climo_dir>
    <write_plot_vars type="boolean">       True     </write_plot_vars>
    <max_data_filesize type="integer">      100     </max_data_filesize>
    <max_data_blocksize type="integer">     500     </max_data_blocksize>
    <verbosity  type="integer">               2     </verbosity>
    <exit_on_warning type="boolean">       True     </exit_on_warning>
    <output_file_type>                       ps     </output_file_type>
</GLOBAL>

<MODELS>
    <model>  CMIP5  MPI-ESM-LR   Amon   historical  r1i1p1
                                          1999 2004   Data/  </model>
</MODELS>

<DIAGNOSTICS>
    <diag>
        <description> Textual description of diag </description>
        <variable_def_dir>   ./variable_defs/   </variable_def_dir>
        <variable>             MyVar            </variable>
        <field_type>           T3M              </field_type>

        <diag_script_cfg_dir>./nml/cfg_MyDiag/  </diag_script_cfg_dir>
        <diag_script cfg="cfg_MyDiag.ncl">
                                  MyDiag.ncl       </diag_script>
    </diag>
</DIAGNOSTICS>
```

The three main sections (tags) listed above are,

- <GLOBAL>: global settings such as <plot_dir> - the path to the output figures and <output_file_type> - the format of the output figures
- <MODELS>: defines which (gridded) data sets should be used for the diagnostics. In the MyDiag example there is just a single model defined. The first entry of the <model>-tag, 'CMIP5', indicates that the data set file names should adhere to the CMIP5 filename syntax[8]. The final <model>-tag entry defines the full path to the folder with the input files

and the second to the seventh entries are used to match which files to include.

- <DIAGNOSTICS>: lists which diagnostics, delimited by the <diag>-tags, that should be produced for data sets selected in the <MODELS>-tag. In this case there is a single diagnostic defined by the diagnostic script 'MyDiag.ncl'

Try out: Update some of the global settings, for example setting the 'output_file_type' to png, and rerun the namelist.

Try out: If you've downloaded additional air temperature (ta) CMIP5-data sets from ESGF[1], try rerunning the namelist specifying one of those data set instead of the MPI-ESM-LR.

5.3 The MyDiag diagnostics

The <diag>-tags of the nml/namelist_Mydiag.xml-file specifies which diagnostic to plot. The <diag>-tag given above is repeated here for convenience,

```
<diag>
    <description> Textual description of diag </description>
    <variable_def_dir>   ./variable_defs/   </variable_def_dir>
    <variable>             MyVar            </variable>
    <field_type>           T3M              </field_type>
    <diag_script_cfg_dir>./nml/cfg_MyDiag/  </diag_script_cfg_dir>

    <diag_script cfg="cfg_MyDiag.ncl">
                               MyDiag.ncl       </diag_script>
</diag>
<diag>
```

The <diag> tag encloses the following tags,

- <description>: a textual summary of the current <diag>-tag
- <variable_defs>: folder where the variable scripts are located
- <variable>: a variable script - the script defines the variable to plot and if/how it should be transformed prior to plotting. See section 5.4 for details

- <field_type>: a string describing the required type of the input field. Here, T3M indicates that the data is a time series (T), 3-dimensional (3), monthly means (M).
- <diag_script_cfg_dir>: folder where the diagnostic specific configuration file is located
- <diag_script>: the diagnostic script to execute. The script should be located in the local diag_scripts/-folder. Prior to executing the script the settings in the file defined by the cfg-attribute is loaded

Try out: Try updating the diag_script configuration file, 'cfg_MyDiag.ncl' (located in the <diag_script_cfg_dir>), and rerun the namelist. The comments in 'cfg_MyDiag.ncl' suggests how to change some values in the configuration file. Additional options for, e.g. projections, can be found in the NCL documentation at the NCAR homepage[6].

5.4 The MyDiag variables

The variable tag in the MyDiag diagnostic configuration file (section 5.3) is a script that defines the variable to plot and if/how it should be transformed prior to plotting. Variable scripts are located in the folder specified by the <variable_defs>-tag. In the MyDiag-example the variable is called 'MyVar' and is defined in the script 'variable_defs/MyVar.ncl':

```
;
;  Requires: ta:*3*
;
variable_info = True
variable_info@derived = True
variable_info@long_name = "air temperature"
variable_info@units = "K"
variable_info@MyDiag_title = "200 hPa"

load "interface_script/add_data_var.ncl"
undef("calculate")
function calculate(index [1] : integer,
                   variable [1] : string,
                   field_number [1] : string)
;;                 return_val [1] : logical
;; Arguments:
;;    index    - index to current infile defined in a
```

```
;;               temporary file in the folder 'interface_data'
;;    variable - logical with releveant variable as string attribute
;;    field_number  - string with field number classification
;; Return value:
;;    data_new - logical
local tmp, dum, dimension
begin
    data_new = True
    tmp = read_data(index, "ta", "*3*")
    dimension = 1  ; Level

    ; See ./interface_scripts/extract_data.ncl
    dum = extract_data(index, tmp, dimension, 200., 200.)
    dum@long_name = variable_info@long_name

    ; See ./interface_scripts/add_data_var.ncl
    add_data_var(index, data_new, dum, variable)
    return(data_new)
end
```

The important lines from the above script, slightly rearranged, are,

1. `variable_info = True`
   The '`variable_info`' is a global variable used to control the logic of the tool

2. `varable_info@derived = True`
   `derived = True` indicates that the function `calculate`, defined in the variable script, should be executed in order to transform/modify the read netCDF before it is sent to the diagnostic script

3. `; Requires:  ta:*3*`
   This lists the dependencies needed for the `calculate`-function. The general syntax is

   ```
   ;  Requires: variable1:field_type1,variable2:field_type2,...
   ```

   where the '`variable?`' refers to other variables and the '`field_type`' is the field_type described in section 5.3. Note that on the '`Requires:`'-line the field type allows for wildcards and a '`*`' can represent any character. In this particular case '`*3*`' could match '`T3M`', '`C3M`' (Climatology), '`T3D`' (Daily), etc.. If no transfomation is to be performed the syntax '`; Requires:none`' should be used.

4. `function calculate(index [1] : integer,`
   The function to transform the requested variable. In the above example a 2D layer is extracted from a 3D data set.

5. `tmp = read_data(index, "ta", "*3*")`
   Reads data from the netCDF files matched in the MyDiag namelist, section 5.2. The field to read, 3-dimensional air temperature, is encoded in the `read_data`-arguments

6. `dum = extract_data(index, tmp, dimension, 200., 200.)`
   The variable '`dimension`' is set to match the levels-dimension of the 3D-data set at hand. '`200., 200.`' denotes a range along this dimension, i.e., this line will extract the 200 hPa level.

To conclude, the MyDiag variable script reads air temperature from 3-dimensional data sets, extracts the 200hPa level and sends it to the diagnostic script described in section 5.5.

Try out: Try extracting the 850 hPa level instead of the 200hPa, note that '`200`' is hardcoded not only in the '`extract_data`'-call but also in the title definition, '`MyDiag_title`'. Because of intermediated file caching of the variable script output it is necessary to remove the files in the directory specified by the namelist tag <`climo_dir`>.

```
rm -f <CLIMO_DIR>-FOLDER
```

before rerunning the the updated variable script.

Try out: Try plotting a different 3D-variable, e.g., `ua` if these have been downloaded separately from ESGF[1]. Proceed by by updating the '`ta`' on the `Requires:`-line and in the `read_data(...)`-call + removing the intermediate file if present. **NB:** if explicit contour lines were activated in the '*Try out*' block in section 5.3 these contour lines will have to be updated to a valid `ua`-range or inactivated.

Try out: Try using a different variable script, e.g., '`variable_defs/pr.ncl`' instead of the default '`variable_defs/MyVar.ncl`'. This requires updates to the `<variable>` and `<field_type>`-tag in the MyDiag diagnostics (section 5.3). Because of the way the `MyDiag`-script is implemented it is also necessary explictly add the following `variable_info`-attributes,

```
var_att_info@MyDiag_title = "title"
var_att_info@long_name = "long_name"
var_att_info@units = "units"
```

to the '`variable_defs/pr.ncl`' file[2] **NB:** if explicit contour lines were activated in the '*Try out*' block in section 5.3 these contour lines will have to be updated to a valid precip range or inactivated.

Try out: Try using the '`variable_defs/pr_mmday.ncl`'. This requires similar changes as the ones above.

5.5 The MyDiag diagnostic script

The `<diag_script>` tag in the MyDiag diagnostic configuration file (section 5.3) specifies which script that should be used. The script should be located in the local `diag_scripts/`-folder. Prior to executing the diagnostic script the selected data sets are sent through, and possibly modified by, the variable script (section 5.4). Settings for the <`diag_script`> can be defined in the configuration file defined by its `cfg`-attribute.

The MyDiag diagnostic script takes a single model and creates global contour plot averaged over the years specified in the namelist (section 5.2). The script is a minimalist example of a diagnostic script where many feautres usually available in diagnostic scripts are removed.

Try out: Try adding the add years used for averaging to the main title of the diagnostic script. The years are accessible in the diagnostic script from the '`models`'-variable. Use the below lines to copy the start/end years to a local variable.

```
start_date = models@start_year(0)
end_date = models@end_year(0)
```

Concatenate the '`start_year`'/'`end_year`' strings to the main title by,

```
res@tiMainString      = MyParam + " (" \
                                + start_year + "-" \
                                + end_year   + ")"
```

Backslash ('\') indicates a continuation line.

Try out: Try adding a loop over models so that multiple models can be defined in the namelist (section 5.2) and plotted into separate figures. What is needed to do is

- **(1)** add models to the namelist, `namelist_MyDiag.xml`

[2] '`pr`'-data sets needs to be downloaded separately from ESGF[1]

- **(2)** update the `plot_script/MyDiag.ncl`-script along the lines,

```
do imod = 0, dimsizes(models@name) - 1   ; The loop
    ...
    ;; Model aware output figure name
    outfile = plot_dir + "/" + diag_script + "/MyDiag_"\
                       + variable + models@name(imod)\
                       + "."      + file_type
    ;; Open output figure file
    wks = gsn_open_wks(file_type, outfile)
    ...
    ;; Plot!
    map = gsn_csm_contour_map(wks, data1, res)
    ...
    ;; Delete variables that may be re-used with new size
    delete(A0)
    delete(data1)
end do  ; End of loop
```

- **(3)** download additional data sets from the ESGF data nodes[1] to get a range of models to loop over.

Examples of similar loops can be found in the existing diagnostic scripts in the diag_scripts/-folder. **NB-1:** NCL cannot reuse variables if they are re-assigned with a different size in a later loop iteration; these variables has to be deleted with '`delete(variable)`'. **NB-2:** To compare the result of different data sets it is useful to activate the explicit contour lines mentioned in the '*Try out*'-block in section 5.3, otherwise each plot will use different color ranges.

A Appendix

A.1 User editable files

Table 1 in this section summarizes the files edited by the user in the ESMValTool tutorial.

`nml/namelist_*.xml`	global settings, models, diagnostics
`variable_defs/<VAR>.xml`	variable transformation (if any), variable specific settings
`nml/cfg_*/*`	diagnostic script settings
`diag_scripts/*`	diagnostic script

Table 1: Reference table for files updated by the user in the ESMValTool tutorial

References

[1] ESGF. ESGF Portal. http://pcmdi9.llnl.gov/esgf-web-fe.

[2] ESMValTool. ESMValTool subversion repository. https://svn.dlr.de/ESM-Diagnostic/source/trunk.

[3] Martin Evaldsson. ESMValTool Tutorial tarball, 2013. http://www.embrace-project.eu/index.php/financial-administration/project-information/meeting-details/2nd-ga-2013.

[4] A. Gettelman, V. Eyring, C. Fischer, H. Shiona, I. Cionni, M. Neish, O. Morgenstern, S. W. Wood, and Z. Li. A community diagnostic tool for chemistry climate model validation. *Geosci. Model Dev. Discuss.*, 5 (2):1229–1261, 2012. http://www.geosci-model-dev-discuss.net/5/1229/2012/gmdd-5-1229-2012.html.

[5] Klaus-Dirk Gottschaldt. Introduction to the ESMValTool, 2013. http://www.embrace-project.eu/index.php/financial-administration/project-information/meeting-details/2nd-ga-2013.

[6] NCAR. CISL's NCAR Command Language (NCL). http://www.ncl.ucar.edu/.

[7] PCMDI-CF. CF Metadata Conventions. http://cf-pcmdi.llnl.gov/documents/cf-conventions/latest-cf-conventions-document-1.

[8] PCMDI-CMIP5. CMIP5 Overview. http://cmip-pcmdi.llnl.gov/cmip5/.

[9] R. R Language Definition. http://cran.r-project.org/doc/manuals/r-release/R-lang.html.

[10] Karl E. Taylor and Charles Doutriaux. CMIP5 model output requirements: File contents and format, data structure and metadata. Technical report, Program for Climate Model Diagnosis and Intercomparison (PCMDI), 2010. http://cmip-pcmdi.llnl.gov/cmip5/docs/CMIP5_output_metadata_requirements.pdf.