Overview of ESMValTool

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-files. This document gives a practical overview of the tool; describing its background, the project workflow, the control flow of the tool and basic operations such as configuring it to produce a certain plot/variable, how to add new plot scripts/variables. Specific issues are further explained in the ESMValTool '*doc/reference.pdf*'. For a more "hands-on" approach, see the '*doc/toy-diagnostics-tutorial.pdf*'. The tool is directly built upon the Chemistry-Climate-eValuation Tool[4] (CCMVal Tool). For an in-depth technical discussion on the purpose and possibilities with ESMValTool, please see the corresponding discussion for CCMVal[4].

Outline The remainder of this document is organized as follows. Section 2 lists the ESMValTool prerequisites, section 3 project workflow, section 4 describes the overall structure of the tool, section 5 details the configuration files used by the ESMValTool, section 6 works through a number of examples. Finally, section 7 describes the template that can be used as a toy diagnostic for becoming familiar with the ESMValTool structure.

2 ESMVal prerequisites

The ESMValTool requirements are,

- compulsory: Python 2.x installation (versions 3.x will not work)
- compulsory: NCAR Command Language (NCL)[5] version 6.1 or higher. Google "install ncl" to find the NCAR page describing how to download and install NCL locally
- compulsory: CMIP5-style[7] 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[6] and, to work work out of the box, the netCDF input file names should follow the CMIP5-naming convention[9]. 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.
- optional: The R scripting language[8] version 2.14 or later (older version may work). R is only needed if the R-based plot scripts are used for output. These plot scripts does in turn use the following R-packages,
  - ncdf4
  - fields
  - maps
  - MASS
  - chron

3 Project workflow

The evaluation tool development is managed through a number of services run on the German Air and Space agency's (DLR) infrastructure. This includes a;

- **wiki:** Documentation and overview of the implemented diagnostics, technical details on the development process[3]
- **issue tracker:** Detailed description and progress of ongoing work[1]
- **subversion repository:** Latest stable and development versions of the code[2]

Access to the above sites requires log in and is available only on request.

4 Overall structure of ESMValTool

Figure 1 gives an overview of the tool where it has been divided into a number of components,

1. Configuration files specifying the general settings and which variables and diagnostics to produce
2. Python scripts which will read the configuration files and launch the following scripts in sequence
3. A number of NCL (NCAR Command Language) scripts to read, check and reformat the input netCDF data consistently
4. A number of NCL scripts to, if requested, compute derived variables from existing data sets
5. A number of diagnostic scripts, in any langauge, that will read the data sets from points 3-4 and output a figure or a netCDF-file with statistics

To run a session of ESMVal, issue

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

The Python main script will read the namelist configuration file, in figure 1 called `namelist.xml`, which contains

(a) global settings

(b) models to include in this session

(c) diagnostics to include in this session

Next, the Python scripts will translate the configuration settings to a temporary text file in the diagnostic script target language. Some of the information is also written and to specific environment variables. Finally the sequence of scripts, points 3-5 in figure 1, are launched. The launched scripts uses the information in the temporary text files (and/or the environment variables) to read the specified netCDF files, write necessary intermediate netCDF files and generate the specified output.

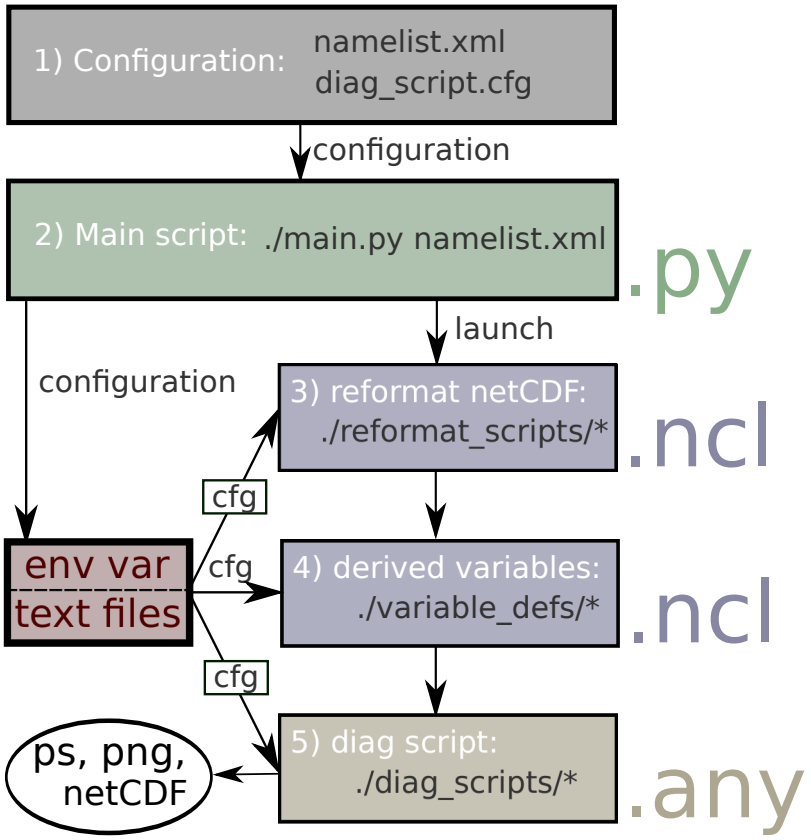

Figure 1: Overview of the ESMValTool package components. **1)** General and diagnostic script specific configuration files, xml and language dependent respectively. **2)** Main python script orchestrating the exectution. **3)** NCL script to check and standardize netCDF input files. **4)** If requested, compute derived variables. **5)** Diagnostic scripts in any language. Communication from the configuration files to the different stages of the tool is through environment variables (env var) and temporary text files.

5 Configuration files

A complete set of diagnostics is definted by up to three different configuration files,

- the namelist configuration file, e.g., `nml/namelist.xml` This file is changed as part of the ordinary ESMValTool workflow. See section 5.1 for more details.
- the plot script configuration file, e.g., `nml/cfg_MyDiag/cfg_MyDiag.ncl` This file is less frequently changed. See section 5.2 for more details.
- the variable definition file, e.g., `variable_defs/MyVar.ncl`. This file is only updated when setting up a new diagnostic. See section 5.3 for more details.

5.1 Namelist configuration file

A sample of a namelist configuration file is shown in table 1 on the following page. This file contains three sections,

- `<GLOBAL>` - Controls the general settings for the namelist, see section 5.1.1 for further details
- `<MODELS>` - Defines which models should be included and the path to their respective netCDF file. The syntax is explained in section 5.1.2.
- `<DIAGNOSTICS>` - Defines which diagnostics that should be created, see section 5.1.3.

Note the logical structure of the xml-configuration files, or indeed any xml-file, is described by an opening tag, e.g., `<GLOBAL>`, and a corresponding closing tag, `</GLOBAL>`.

5.1.1 GLOBAL

The global section configures general settings for the tool. Each tag in this section is parsed as a key, and the content between the opening/closing tag is the corresponding value to that key. The full key-value list is shown in table 2 on the next page. Some tags also define an attribute "type", which declares which type the value should have once read into Python. The default type is a string.

```
<!-- A comment begins like this and ends on
     the same or on a later line with -->
<GLOBAL>
  <plot_dir type="path">  plots/    </plot_dir>
  <output_file_type>          ps     </output_file_type>
  ....
</GLOBAL>
<MODELS>
  <model> CMIP5  EC-EARTH  Amon ... [path to netCDF files] </model>
  <model> CMIP5  GPCP      Amon ... [path to netCDF files] </model>
   ....
</MODELS>
<DIAGNOSTICS>
  <diag>
     ....
  </diag>
  ....
</DIAGNOSTICS>
```

Table 1: A sample of a namelist configuration file, `namelist.xml`, with its three sections, `GLOBAL`, `MODELS` and `DIAGNOSTICS`.

key/tag	value	description
write_plots	(True|False)	*Currently not used*
write_netcdf	(True|False)	*Currently not used*
force_processing	(True|False)	Force rewriting of certain intermediate netCDF files
wrk_dir	path	Specify output path for references/acknowledgements
plot_dir	wrk_dir/plots	Specify the output plot directory
max_data_filesize	Integer	Max data size to read into memory
max_data_blocksize	Integer	Max block data size to use when writing intermediate files
climo_dir	path	Specify the output directory for intermediate and climatology netCDF files
write_plot_vars	(True|False)	*Currently not used*
verbosity	Integer	Set verbosity level (0+, zero being minimal output)
exit_on_warning	(True|False)	Crash if there is a warning in a plot script
output_file_type	(PDF|PNG|PS|EPS)	Format of output figure files. Note that not all plot scripts supports all values
debuginfo	(True|False)	Write plot type defined debuginfo onto output figure

Table 2: The global settings available in the namelist configuration file.

```
CMIP5  EC-EARTH  Amon   historical  1  1998 2000      path_to_netCDF
CMIP5  ERAINT    DECADAL historical 0  1998 2000      path_to_netCDF
CMIP5_SMHI GPCP  OBS                1  1998 2000 mon  path_to_netCDF
```

Table 3: Example of the `MODELS` section of the namelist configuration file. The first entry on each line defines how that line should be parsed. Here, the general "CMIP5"-specifier and the institute specific "CMIP5"-specifier are used. Notice the the number of entries for different specifiers may differ.

5.1.2 `MODELS`

The `MODELS` section describes which model data sets and years should be used when plotting the listed diagnostics. This section consists of a number of lines as shown in table 3, The first entry on each model line is referred to as the "project specifier". The remaining entries are used to specify which data sets should be used for the plotting. The project specifier does, on a source code level, define exactly how each model line should be parsed. It is thus possible for the user to set up his/her own project specifier for data sets that do not conform to the default CMIP5-filename syntax[7]. For the CMIP5-project specifier, the entries on each model line are (here with an artificial line break to fit the line widht),

For the CMIP5_SMHI specifier, the entries are, (again with an artificial line break to fit the line widht)

Defining new project specifiers can be used not only to handle data sets with non-CMIP5 filenames but also to pass on additional information about each model to the tool. In this particular case the CMIP5_SMHI class is used

to manage the specific directory structure such that only the root_folder for the for CMIP5-data sets on the local filesystem needs to be specified. See the '*doc/reference.pdf*' or the code for additional examples how to work with project specifiers.

5.1.3 DIAGNOSTICS

The DIAGNOSTICS section of the namelist.xml-file lists the requested diagnostics. Each diagnostic is enclosed in an opening <diag> and closing </diag>-tag. A sample section is presented below,

```
<DIAGNOSTICS>

<diag>
   <description> A textual summary of the current diag-tag </description>
   <variable>                                pr_mmday    </variable>
   <field_type>                                  T2Ms    </field_type>

   <diag_script_cfg_dir>     ./nml/cfg_generic    </diag_script_cfg_dir>
   <diag_script cfg="cfg_generic_tsline.cfg">    tsline   </diag_script>
   <diag_script cfg="cfg_generic_monline.cfg">  monline   </diag_script>

   <model>  CMIP5  HadGEM2-ES    Amon ... [path to netCDF files] </model>
</diag>
...
</DIAGNOSTICS>
```

In the <DIAGNOSTICS>-tag any number of <diag> sections can be defined. Each <diag> section should in turn list,

- <**variable**> a variable script - the script defines the variable to plot and if/how it should be transformed prior to plotting. Variable scripts should be located in the local variable_defs/-folder and be written in NCL. The script is executed in step 4 of figure 1. Although the script itself has to be in NCL any meta data that is defined in the script is transcribed to the target diagnositc script language. Defining multiple <variable>-tags means that multiple variables are passed on to the diagnostic script.

- <**field_type**> a string describing the requried type of field. For example, T2Ms indicates the data is a time series (T), 2-dimensional (2),

monthly means (M) and at the surface (s). The type information is used to control the behaviour of the various diagnostic scripts. For a full reference of field types, see table 4 on page 22. If multiple <variable>-tags are defined a single (which is applied to all) or an equal number of <field_type>-tags has to be defined.

- <**diag_script**> the diagnostic script to execute. The script can be in any language currently interfaced to ESMValTool and should be located in the local diagnostic_scripts/-folder. Prior to executing the script the settings in the file defined by the cfg-attribute is loaded.

- <**diag_script_cfg_dir**> This tag defines the folder where the diagnostic script configuration file is located. See section 5.2 for details on the diagnosic script configuration file.

- <**model**> [optional] Additional data sets specific for this <diag>-section. Data sets defined here will be applied together with the models from the namelist.xml to the diagnosic scripts in the current <diag>-section but are then removed.

5.2 Diagnostic script configuration files

The diagnostic script configuration file for each diagnostic script is specified in the cfg-attribute (see section 5.1.3). The file is written in the plot script target language and is imported explicitly in the diagnostic script. Below follows an example for the "Standardized Precipitation Index"-diagnostic written in R,

```
begin.ref.year <- 1970
end.ref.year <- 2000
timescale <- 3  ## Valid values are 3, 6 and 12 months
seasons <- c("ann", "djf", "mam", "jja", "son")
spi_colorbar_max <- 0.75
my.colors <- colorRampPalette(c("brown","orange","white",
                                "lightblue","blue"))

## Specific settings for PNG output
png_width <- 1600
png_height <- 960
png_units <- "px"
```

```
png_pointsize <- 12
png_bg <- "white"
```

This configuration file should contain diagnostic script settings that the user might want to change (hence they should be moved out of the diagnostic script) but are too specific to merit a place in the namelist. In the actual plot script this configuration file is loaded/imported in the standard way for that language. For example, the above example is imported as,

```
source('data_interface/r.interface')
source(diag_script_cfg)
```

where `data_interface/r.interface` is the temporary R-file created on the fly with all configuration settings, including the `diag_script_cfg`-variable.

5.3 Variable defintion file

The variable definition file specifies transformations of the <`diag`>-tag variables (see section 5.1.3). It is located at,

```
variable_defs/<VARIABLE>.ncl
```

If no transformation should be applied, e.g., when the variable should be read/plotted directly from its netCDF source, this file will be almost empty,

```
;
; Requires: none
;
variable_info = True
variable_info@derived = False
```

For a derived variable this file will list the required variables (and fields), it will include an NCL function `calculate(...)` defining the transformation and, if necessary, meta data regarding the transformation,

```
;
; Requires: pr:T2*s
;
variable_info = True
variable_info@derived = True
variable_info@long_name="Precipitation Rate"
```

```
variable_info@units="mm/day"

undef("calculate")
function calculate(index [1] : integer, ...
begin
   ...
end
```

The 'Requires'-comment at the top lists all the variables needed to compute the derived variable. The `calculate(...)`-function is executed in step 4) of figure 1 and will rewrite the input data prior to launching the actual diagnostic script. Any meta data defined in the variable_defs-file, i.e., the attributes of `variable_info`, is rewritten to a temporary file in the target diagnostic script language and imported in the diagnostic script. See the practical exampel in section 6.2 and the '*doc/reference.pdf*' for further details.

6 Examples

This section explains the basic use of ESMValTool by working through the examples listed below. For a more detailed approach, see the '*doc/toy-diagnostic-tutorial.pdf*'.

6.1 A precipitation surface contour plot

The first example, figure 2, is a global surface contour plot of precipitation flux for a single model. The exact procedure to produce this plot is described step by step in sections 6.1.1 to 6.1.4.

6.1.1 Download netCDF data file

Start by downloading historical precipitation data for the global model EC-Earth from the CMIP5 archive to a local directory,

```
$ ls /nobackup/cmip5-data/*
pr_Amon_EC-EARTH_historical_r12i1p1_195001-201212.nc
```

6.1.2 Add/update a namelist configuration file

We start from a template namelist and fill out the <GLOBAL> and <MODELS> settings as shown below. See sections 5.1.1 and 5.1.2 respectively for more details on these sections, or check the '*doc/reference.pdf*'.

```
<namelist>
<namelist_summary>
    Textual description of the namelist
</namelist_summary>
<GLOBAL>
    <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>                    eps   </output_file_type>
    <debuginfo type="boolean">          False   </debuginfo>
    <write_plots type="boolean">         True   </write_plots>
    <write_netcdf type="boolean">        True   </write_netcdf>
    <force_processing type="boolean">   False   </force_processing>
</GLOBAL>

<MODELS>
     <model>  CMIP5  EC-EARTH    Amon historical
              r12i1p1  1998 2004 /local_disk/cmip5-data </model>
</MODELS>

<DIAGNOSTICS>
    <diag>
         <!-- See section below -->
    </diag>
</DIAGNOSTICS>
</namelist>
```

6.1.3 Add diagnostics

The <diag>-tag in the previous section should specify which variables/-diagnostics to combine and the "type" of the data set. For this example we'll use a global contour plot for precipitation,

```
<DIAGNOSTICS>
<diag>
    <description> Example used in doc/overview.pdf </description>
    <variable_def_dir>     ./variable_defs/       </variable_def_dir>
    <variable>               pr                   </variable>
    <field_type>             T2Ms                 </field_type>

    <diag_script_cfg_dir>  ./nml/overview_cfg/    </diag_script_cfg_dir>
    <diag_script cfg="precip.ncl"> surfconplot_simple.ncl  </diag_script>
</diag>
</DIAGNOSTICS>
```

The above snippet indicates that we would like to plot precipitation (variable `pr`) using the existing diagnostic script '`diag_scripts/surfconplot_simple.ncl`'. This is a simple script capable of producing absolute or difference surface contour plots. The input field should be `T2Ms`, i.e., a time series of 2-dimensional monthly means surface data. The tags <diag_script> and <variable> are further described below.

6.1.3.1 The <diag_script>-tag

Diagnostic scripts are external language scripts written to create a specific plot, sets of plots, or netCDF output. The python part of ESMValTool, section 2) in figure 1, is responsible for interfacing the information in the configuration files to the target diagnostic script language. From a user perspective this is completeley transparent. See the '*doc/reference.pdf*' or exising scripts for details on how to interface a new diagnostic or how to create an interface of an unsupported language.

Existing ESMValTool diagnostics are available in the diag_scripts/-folder and documented on the project wiki[3] (login required). For the current example we will use the diag_scripts '`surfconplot_simple`' which is defined in the script `diag_scripts/ surfconplot_simple.ncl`.

6.1.3.2 The <diag_script> cfg attribute

The diag_script tag takes an attribute **cfg** which points to a diag_script specific configuration file. The format of this file should allow it to easily be

imported to the diag script, and it should be located in the folder dictated by the earlier <diag_script_cfg_dir>-tag. In this example the configuration file, 'precip.ncl', is written in NCL, hence easily read directly by the diagnostic script. The cfg-file contains two settings,

```
diag_script_info = True
diag_script_info@cn_levels_mean = (/1.0625, 2.125, 3.1875, 4.25,\
                                    5.3125, 6.375, 7.4375, 8.5,\
                                    9.5625, 10.625, 11.6875,12.75,\
                                    13.8125, 14.875, 15.9375/) * 1e-5
diag_script_info@seasons = (/"ANN"/)
```

The properties set are which contour levels to use and the season to average over (ANN=Annual). Exactly which properties are set in the diagnostic script configuration file vs which are set directly in the diagnostic script itself is a design question for the diagnostics at hand. See section 5.2 or an exisintg diagnostic script configuration files for further examples.

6.1.3.3 The <variable>-tag

The <variable>-tag referres to a script located in the local sub-directory 'variable_defs/<VARIABLE> .ncl'. The script is always written in NCL and exectued in section 4 of figure 1. In this specific case we will use variable_defs/pr.ncl. The variable script should contain the logical variable 'variable_info' and the attributes of this variable will be passed on to the diagnostic script via the language dependent interface briefly mentioned in the<diag_script>-tag above. Our file looks like,

```
;
; Requires: none
;
variable_info = True
variable_info@derived = False
```

The only attribute used, derived = False, means the the input data (pr) should not be processed but sent directly to the plot script. The Requires-line is, although a comment, compulsory and further described in section 6.2.

6.1.4 Generate the plot

Running ESMValTool with,

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

will output the plot in figure 2. Notice that the max/mean values look awkward, apparently the `surfconplot_simple` script write these values with two decimal positions, regardless of the required precision. Also note that some properties, in this case the colormap, is set automatically by NCL since it is not explicitly defined.

If the diagnostic script, for any reason, fails in step 3 of figure 1 a temporary, incomplete, reformat file might have been created. When rerunning the namelist the tool will notice the presence of the (incomplete) file and automatically skip this step which will cause yet another error because of the errenoeous reformat file. Simply remove all the reformat files, located according to the namelist '<`climo_dir`>'-tag, and rerun again.

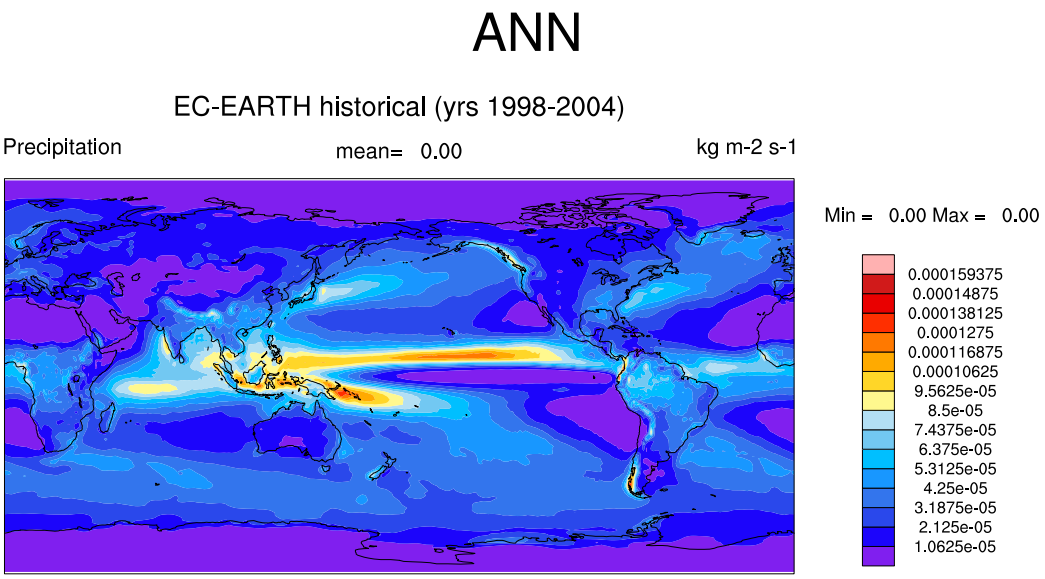

Figure 2: Precipitation flux average for EC-Earth 1997-2000 in $kg * m^2/s$

6.2 Working with derived variables

In addition to plotting variables directly ESMValTool provides functionality to easily define variables derived from existing ones. A very simple example is be to re-plot figure 2 using units mm/day instead of $kg * m^2/s$. To this end we define the new variable `variable_defs/pr_mmday.ncl` accordingly,

```
;
; Requires: pr:T2*s
;

variable_info = True
variable_info@derived = True
variable_info@long_name="Precipitation Rate"
variable_info@units="mm/day"

load "interface_scripts/add_data_var.ncl"
undef("calculate")
function calculate(index [1] : integer,
                   variable [1] : string,
                   field_number [1] : string)
begin
    result = read_data(index, "pr", "T2*s")
    T = extract_data(index, result, -1, 0, 0)
    T = T * 24 * 3600
    T@units = variable_info@units
    data_new=True
    add_data_var(index, data_new, T, variable)
    return(data_new)
end
```

The '`Requires: pr:T2Ms`'-row at the top declares this variable to be derived from the variable `pr_att.ncl` (example 2). Additionally, the `derived` attribute (`variable_info`) now set to True implies that the `calculate`-function should be used to transform the variable before sending it to the plot script. Because the variable is transformed the `long_name` and `units` are updated as well. Adding a new <`diag`>-tag with this variable to our namelist,

```
...
<DIAGNOSTICS>
<diag>
    <description> Example used in doc/overview.pdf </description>
```

```
    <variable_def_dir>     ./variable_defs/       </variable_def_dir>
    <variable>               pr                   </variable>
    <field_type>             T2Ms                 </field_type>
    <diag_script_cfg_dir>  ./nml/overview_cfg/    </diag_script_cfg_dir>

    <diag_script cfg="precip.ncl"> surfconplot_simple.ncl  </diag_script>
</diag>
<diag>
    <description> Another example from doc/overview.pdf </description>
    <variable_def_dir>     ./variable_defs/       </variable_def_dir>
    <variable>               pr_mmday             </variable>
    <field_type>             T2Ms                 </field_type>
    <diag_script_cfg_dir>  ./nml/overview_cfg/    </diag_script_cfg_dir>

    <diag_script cfg="precip_mmday.ncl">
                                surfconplot_simple.ncl  </diag_script>
</diag>
...
```

and re-running the main script produces figure 3 along with the earlier figure 2. Since the range will be different with the new units it is necessary to set the contour levels in `precip_mmday.ncl` to,

```
diag\_script_info = True
diag\_script_info@cn_levels_mean = (/1,  2,  3,  4,  5,\
                                    6,  7,  8,  9, 10,\
                                   11, 12, 13, 14, 15/)
diag\_script_info@seasons = (/"ANN"/)
```

6.3 Adding a new diagnostic

Diagnostics are arbitrary language scripts prepared to read the input data, as defined and processed by steps 1-4 in figure 1, and produce some sort of output, figure plots and/or netCDFs with statistics. Currently implemented diagnostic scripts are located in the directory '`diag_scripts/`' and documented on the project wiki[3].

To create a new diagnostic, start from an existing one as a template. If there are no template diagnostic scripts available in the target language it may be a currently unsupported language. Check the '*doc/references.pdf*' for some details on existing language interfaces which may serve as a guide for creating new interfaces.

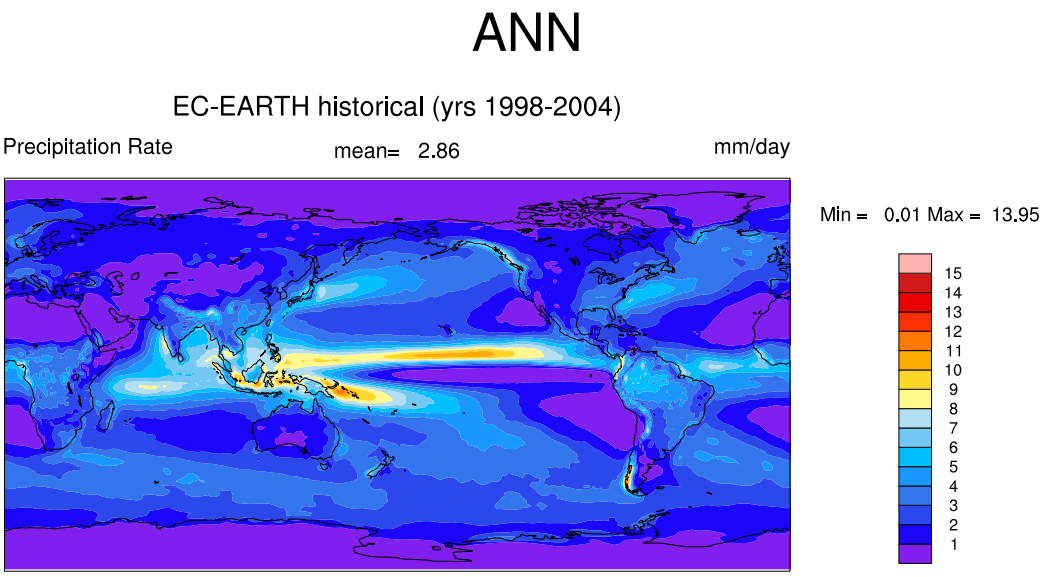

Figure 3: Precipitation flux average for EC-Earth 1998-2004 in mm/day

For NCL scripts, the `MyDiag` suite consisting of,

- `nml/namelist_MyDiag.xml`
- `diag_scripts/MyDiag.ncl`
- `variable_defs/MyVar.ncl`

provides a consistent set of templates that can be used for building new diagnostics. The MyDiag template is further described in section 7.

6.4 Working with observations

Gridded observational data sets in CMIP5 format can be added as just another model in the namelist configuration file, e.g., as is done for GPCP in table 1. The observational data will then be used in the listed diagnostics. To facilitate comparison, some diagnostics are written to pick a certain model as reference. E.g., if we supply the `surfconplot_simple`-diag_script with a reference it will plot the difference between the reference and the current

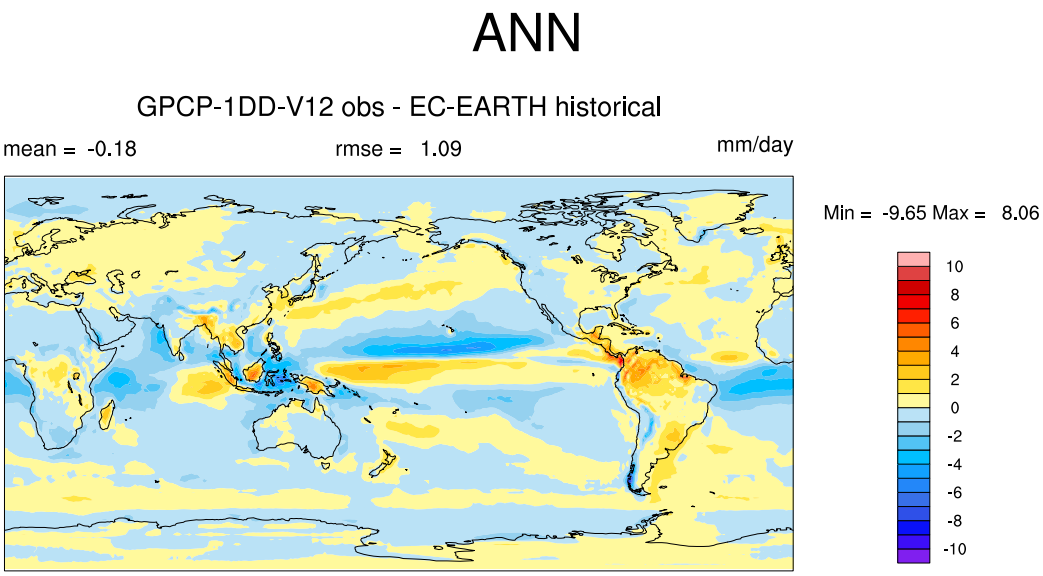

Figure 4: Precipitation flux average for EC-Earth subtracted from GPCP 1998-2004 in mm/day.

model. Technically this can be accomplished by adding GPCP as a model to our previous example (in the `namelist_overview.xml`-file) and designate it as a reference model in the `precip_mmday.cfg`-file by adding the lines,

```
diag_script_info@ref_model = "GPCP-1DD-V12"
diag_script_info@cn_levels_mean_diff = (/-10, -9, -8,...
```

In addition to the `ref_model`-attribute there is also a new contour levels specific for model differences, `cn_levels_mean_diff`. Note that these attributes are implemented in the `surfconplot`-script and may or may not work for other plot scripts. The result is figure 4.

Observational data not in gridded format can be added explicitly in the relevant plot script routines. See ?? for an example of this.

7 The MyDiag template

To get started with the ESMValTool a stripped-down and simplified set of namelist/diagnostic/variable scripts has been put into the repository. These scripts have been designed to let the user play around with the basic features of the tool without requiring too much knowledge in neither NCL nor the tool design. The template consists of the files,

- **nml/namelist_MyDiag.xml:** A namelist defining a single model and a single diagnostic. Running the namelist without modifications requires files matching

  `/local_disk/CMIP5/ta_Amon_MPI-ESM-LR_historical_r1i1p1*`

  to be present.

- **variable_defs/MyVar.ncl:** Specifies variable "`ta`" to be read and extract a 2D field at 200hPa

- **diag_script/MyDiag.ncl:** Creates a contour plot for a single model

Running the tool from the command line with,

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

will output a postscript contour plot of parameter `ta` on 200hPa in the folder `plots/MyDiag/`. See the pdf '*doc/toy-diagnostic-tutorial.pdf*' for a detailed explanation of the `MyDiag`-templates.

T2Ms	Monthly-mean 2-d atmosphere or land surface data (longitude, latitude, time:month)
T3M	Monthly-mean 3-d atmosphere data (longitude, latitude, pressure, time:month)
T2Mz	Monthly-mean zonal mean 2-d atmosphere or land surface data (longitude, pressure, time:month)
T1Ms	Monthly-mean 1-d atmosphere or land surface data on a certain pressure level (latitude, time:month)
T2Ds	Daily-mean 2-d atmosphere data (longitude, latitude, time:day)
T3D	Daily-mean 3-d atmosphere data (longitude, latitude, pressure, time:day)
T2Dz	Daily-mean zonal mean 2-d atmosphere data (latitude, pressure, time:month)
T2Is	Daily instantaneous 2-d atmosphere data for all years (longitude, latitude, time:day)
T3I	Daily-instantaneous 3-d atmosphere data for selected years (longitude, latitude, model levels, time:day)
T2Iz	Daily instantaneous zonal mean 2-d atmosphere data for all years (latitude, pressure, time:day)
T1Iz	Daily instantaneous 1-d field for all years (latitude-pressure, time:day)
T0I	Daily instantaneous 0-d field for all years (time:day)
T0As	Annual-mean 0-d atmosphere or land surface data on a certain pressure level (latitude, time:year)
F2Ms	2D Time independent land surface data Latitude-Longitude
TO2Ms	Monthly-mean 2-d ocean or sea ice data (longitude, latitude, time:month)

Table 4: The "types" of netCDF data supported by ESMValTool

References

[1] ESMValTool. ESMValTool issue tracker, . https://mantis.dlr.de/mantis/main_page.php.

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

[3] ESMValTool. ESMValTool wiki page, . https://teamsites-extranet.dlr.de/pa/ESMValTool/Wiki/Home.aspx.

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

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

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

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

[9] 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.