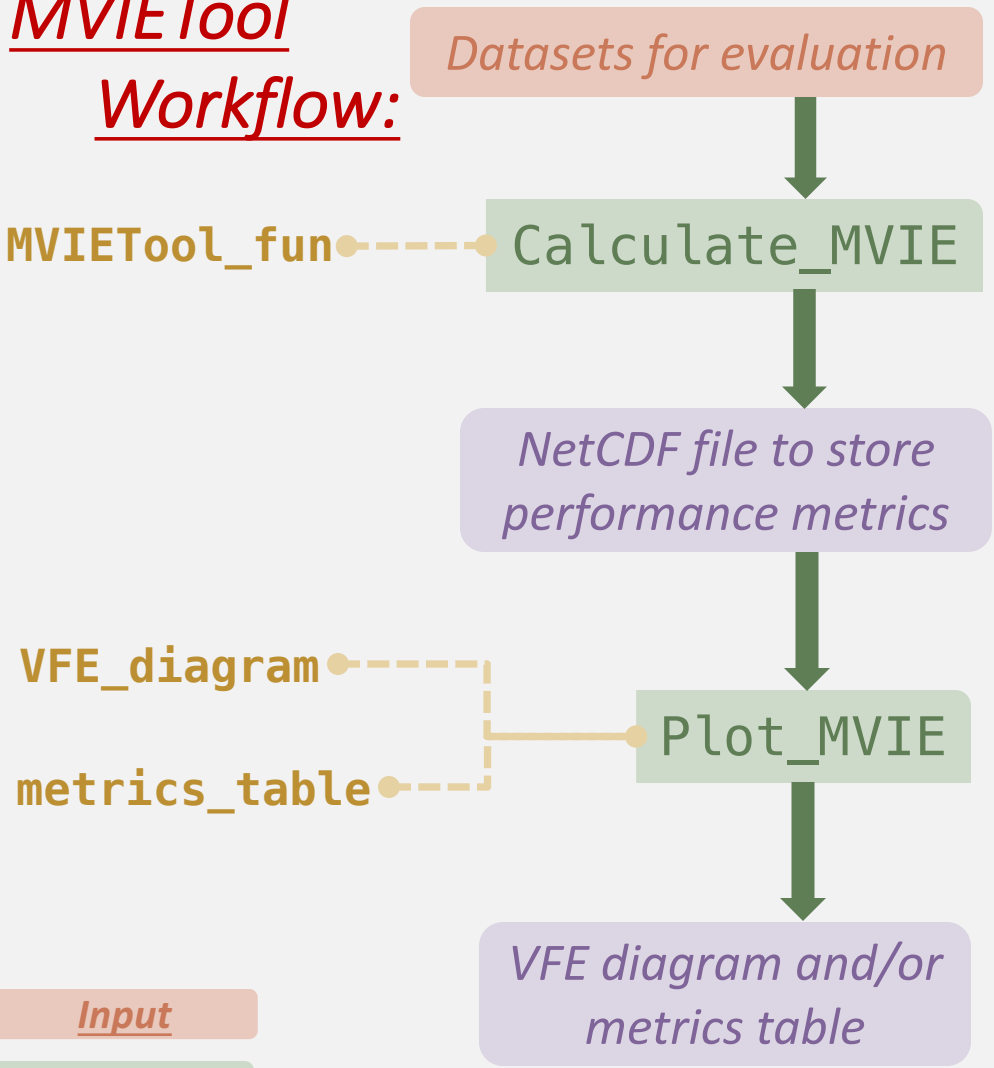
MVIETool v1.0 User Guide

NCL version & Python3 version

Meng-Zhuo Zhang 2021.02.19

MVIETool Workflow:



Input

Main script

Fun script

output

What does user need to do?

Prepare datasets

Set arguments in `Calculate_MVIE`

Run `Calculate_MVIE`

Set arguments in `Plot_MVIE`

Run `Plot_MVIE`

1. How to preprate data?

2. How to set arguments for plotting?

✓ Discription of arguments in main scripts can be found in **readme.namelist**

Package is available at https://github.com/Mengzhuo-Zhang/MVIETool *from the GitHub repository*

Any questions contact us:

zhangmengzhuo1996@163.com

1. How to preprate data?

Input Datasets in MVIETool:

- All evaluated variables of one model/observation/reanalysis dataset are stored in **the individual NetCDF file**.
- All NectCDF files are put **in the same directory**.
- Variables stored in the datafile need to be **on the same grid**.

How to regrid data?

NCL ····· Refer to: ***http://www.ncl.ucar.edu/Applications/regrid.shtml***

Python3 ····· With **Xarray**, refer to: ***http://xarray.pydata.org/en/stable/interpolation.html***

CDO ·····

Refer to: ***https://www.ch.cam.ac.uk/computing/software/climate-data-operators-cdo***

e.g: To remap all fields bilinear to a new grid written in grid file use:

```
cdo remapbil,grid_file input_file output_file
```

To interpolate 3D variables on height levels to a new set of height levels use:

```
cdo intlevel,10,50,100,500,1000 input_file output_file
```

All variables should be arrays with fixed format

Var_Coords=False: *Arrays with the same dimension size (shape)*
– **the entire array are used for evaluation**

Var_Coords=True: *Arrays with the Dimension names and Coordinate variables*
– **can evaluate sub part of the array by setting relative arguments**

Var_Coords	Dimension size (shape)	Num of dimentions	Coordinate variables	Area weighting
False	All variables' arrays need to be in the same dimension size (shape).	No limit to the number of dimensions and no need for dimension names.	No need for coordinate variables.	Cannot add area weighting.
True	All variables' arrays can have different dimension size (shape), but selected part of arrays should have the same dimension size (shape).	The number of dimensions is up to 4, and dimension names should be for latitude, longitude, time and level coordinates, which are same for all varibales.	Coordinate variable should be assigned to each dimension. Coordinate variables in different datafiles can be different, but their dimension names and their values in selected part of evaluated variable arrays should be same.	Can add area weighting when evaluated variable with latitude dimension and coordinate.

Note Please install ***numpy***, ***xarray***, ***Nio*** and ***Ngl*** under Python-version.

Ref
Dimension: *http://www.ncl.ucar.edu/Document/Language/named.shtml*
Coordinate variable: *http://www.ncl.ucar.edu/Document/Language/cv.shtml*

How to define coordinate variables ?

Coordinate	Content	Coordinate variable creation in datasets	
Geo	Latitude and/or longitude coordinate variables assigned to evaluated variable. MVIETool can read sub part of evaluated variable by **Range_geo**.	***1-D arrays of latitude and longitude which are monotonic***	

		e.g.: NCL
		`latitude = fspan(-90.,90., 181)`
		`longitude = fspan(0.,359., 360)`
			e.g.: Python3
			`import numpy as np`
			`lat = np.linespace(-90.,90., 181)`
			`lon = np.linespace(0.,359., 360)`

Time	Time coordinate variable assigned to evaluated variable.
	MVIETool can read sub part of evaluated variable by **Range_time**.
		[1] 1-D array with the mixed Julian/Gregorian date with a 'calendar' attrbution using cd_inv_calendar
		e.g.:
		`Ntime = 12`
		`YYYY = new(Ntime,integer)`
		`YYYY = 2000`
		`MM = ispan(1, 12, 1)`
		`A_0 = new(12,integer)`
		`A_0 = 0`
		`time = cd_inv_calendar(YYYY,MM,A_0,A_0,A_0\`
		`,A_0,"hours since 1900-01-01 00:00:00",0)`
		[2] 1-D array of numeric values with time step
		e.g.: `time = ispan(1, 12, 1)`
			[1] Time series with DateTimeIndex
			e.g.:
			`import pandas as pd`
			`Ntime= 12`
			`time = pd.Series(np.linespace(1,\`
			`  Ntime,Ntime),index=pd.date_range(\`
			`  "2000-01-01",periods=Ntime,\`
			`  freq="M"))`
			[2] Time series
			e.g.:
			`import pandas as pd`
			`Ntime= 12`
			`time = np.linespace(1,Ntime,Ntime)`

Level	Level coordinate variable assigned to evaluated variable. MVIETool can read specific level of evaluated variable as an individual variable by **VarLev**. If some of evaluated variables with level coordinate and the others without, the level value in **VarLev** for variables without level dimension can set to be arbitrary value that would be omitted by the tool.	***1-D array of levels***	

Ref

cd_inv_calendar: http://www.ncl.ucar.edu/Document/Functions/Built-in/cd_inv_calendar.shtml

DataTime Index: https://pandas.pydata.org/docs/user_guide/timeseries.html

How to save variables with dimention name and coordinate variables?

Assume that a variable named 'psl' ' has been preprocessed into [Nlat]X[Nlon]X[Ntime] array and waits to be saved with dimension names (i.e.,"lat", "lon", "time") and coordinate variables (i.e.,latitude, longitude, time). Coordinate variables have been defined in examples of " How to define coordinate variables?". Example code is given as follows:

NCL `filedimdef,filevardef,filevarattdef`

```
f = addfile(filename, "c")
;Define dimension names
filedimdef(f,(/"time","lat","lon"/),(/Nlat,Nlon,Ntime/),(/False,False,False/))
;Save coordinate variables
filevardef(f,"lat",Typ_lat,"lat")
f->lat = (/latitude/)
filevardef(f,"lon",Typ_lon,"lon")
f->lon = (/longitude/)
filevardef(f,"time",Typ_time,"time")
Att = True ;when setting time using Type-[1], its units must be set
Att@units = "hours since 1900-01-000:00:00"
filevarattdef(f_new, "time", Att)
f->time = (/time/)
;Save model/obs data
filevardef(f,"psl",Typ_var,(/"time","lat","lon"/))
f->psl = (/psl/)
```

Python3 `file.create_dimension,file.variables`

```
import Nio
f = Nio.open_file(filename,"c")
#Define dimension names
f_new.create_dimension('lat',Nlat)
f_new.create_dimension('lon',Nlat)
f_new.create_dimension('time',Ntime)
#Save coordinate variables
f_new.create_variable('lat','f',('lat',))
f_new.variables['lat'].assign_value(np.float32(lat))
f_new.create_variable('lon','f',('lon',))
f_new.variables['lon'].assign_value(np.float32(lon))
f_new.create_variable('time','d',('time',))
f_new.variables['time'].assign_value(time)
#Save model/obs data
f_new.create_variable('psl','f',("time","lat","lon"))
f_new.variables['psl'][:] = np.float32(psl)
```

Arguments in MVIETool for example:

NCL version==========

```
Var_Coords = True

isCoords_geo = True
Coords_geo = (/"lat","lon"/)
Range_geo = (/"lat|0:90","lon|0:360"/)
Isarea_wgt = "lat"

isCoords_time = True
Coords_time = "time"
Range_time = (/1,10/)
Range_time = (/"200001","200010"/)
```

array of two int/float in time coordinate variable

array of two strs in the format : "YYYYMM", "YYYYMMDD", or "YYYYMMDDHH", where YYYY is the year, MM is the month, DD is the day, and HH is the hour time coordinate variable in input datafiles must with 'clendar' attribution

Python3 version ==========

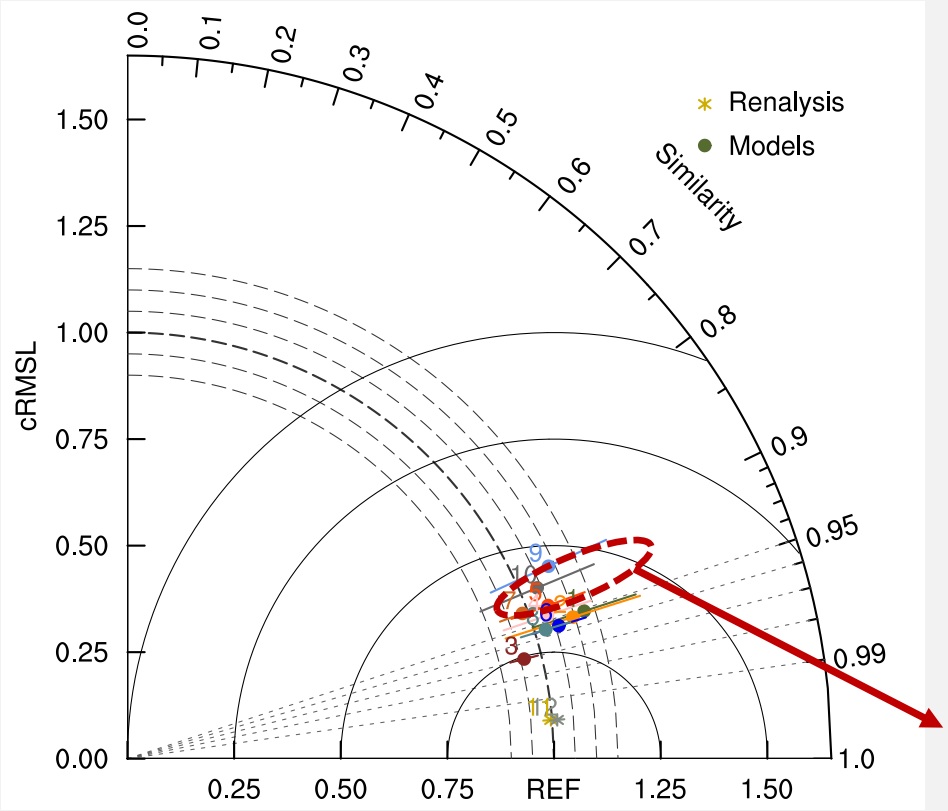
```
Var_Coords = True

isCoords_geo = True
Coords_geo = ["lat","lon"]
Range_geo = {"lat":[0,90],"lon":[0:360]}
Isarea_wgt = "lat"

isCoords_time = True
Coords_time = "time"
Range_time = [1,10]
Range_time = ["2000-01-01 ","2000-10-01"]
```

List/tuple of two int/float values in time coordinate variable

List/tuple of two DateTimeIndex (str), and time coordinate variable in input datasets must be defined with DateTimeIndex

2. How to set arguments for plotting?

VFE diagram



VFE diagram *provides a concise evaluation of how vector field matches to another in terms of RMSL, VSC, RMSVD.*

Azimuthal position ---> **VSC**
Radial distance from the origin ---> **RMSL**
Distance between the point and point (1,0) ---> **RMSVD**

RMSL and VSC are needed to create VFE diagram.
VFE diagram can be used for individual scalar variables, individual vector variables, or multivariable fields by setting statistic-name in `Stats_metrics1`.

e.g.:

```
Stats_metrics1 =  (/"cRMSL","cVSC"/)
plot_std_l     =  "SD_std"
```

```
Stats_metrics1 =  ["cRMSL","cVSC"]
plot_std_l     =  "SD_std"
```

	Stats_metrics1	NOTE
Individual scalar variable	SD, CORR	Need to assign which variable to plot with `ivar` and if not, plot statistics of first variable
	rms, uCORR	
Individual vector variable / Multivariable field	cRMSL, cVSC	Can add lines for each of points, which can represent SD_std/rms_std, with `plot_std_l`
	RMSL, VSC	

VFE diagram: opt_VFE

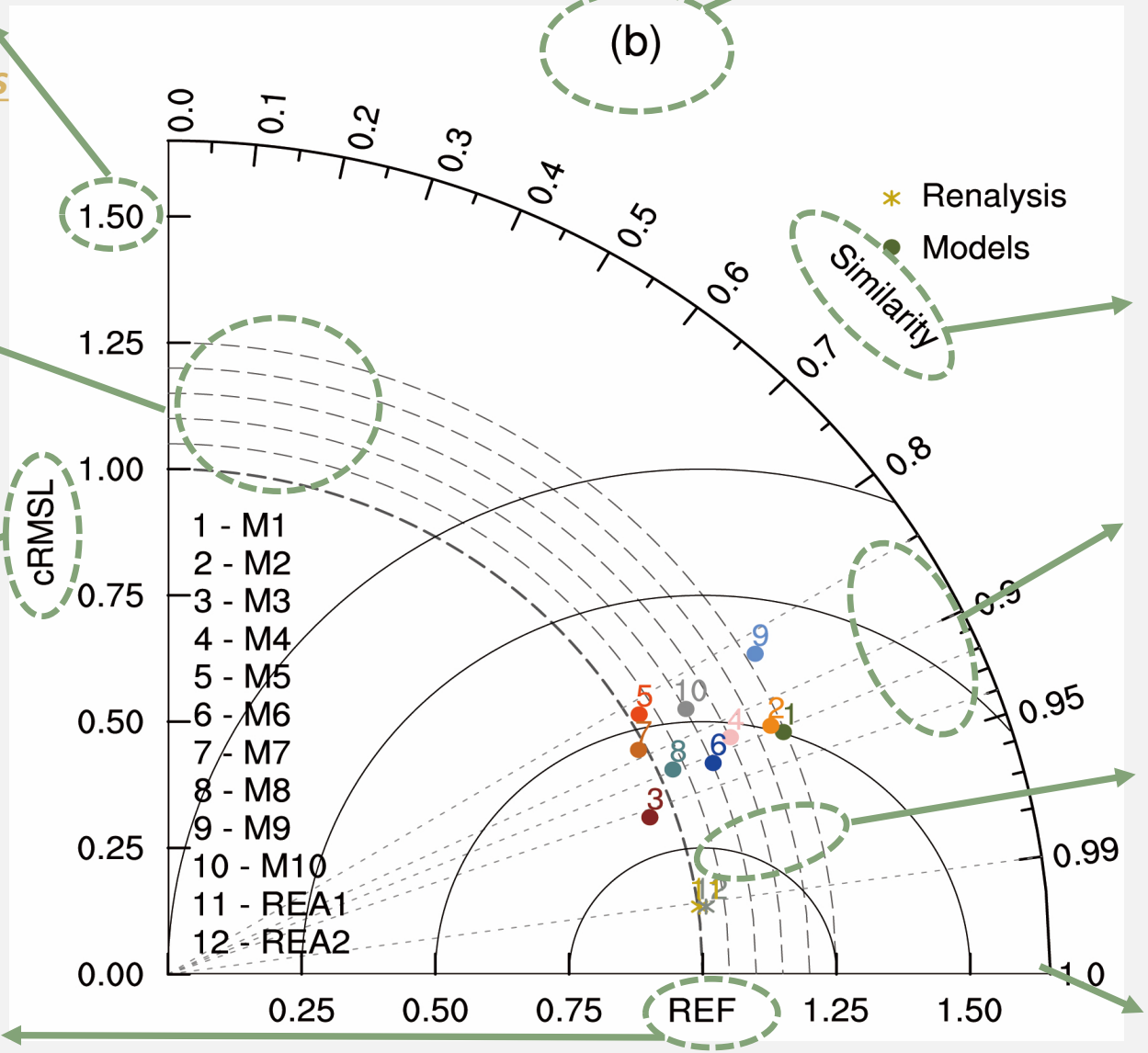
tiMainString
tiMainFontHeightF
diagram title string and its font height

xyFontHeightF
font height for tickmark labels

stnArc
plot specified standard arc

tiYAxisString
title string for y-axis

OneX
string marked at (1,0)



CorLabel
title string for arc-axis

ccRays
plot specified radial lines

cirArc
plot specified circumferential arc

xyMax
max value of X-axis and Y-axis

VFE diagram: opt_VFE

Ref

Marker Index http://www.ncl.ucar.edu/Document/Graphics/Images/markers.png

Named colors http://www.ncl.ucar.edu/Document/Graphics/named_colors.shtml
http://www.pyngl.ucar.edu/Graphics/rgb.txt

NhlTColorIndex http://www.ncl.ucar.edu/Document/HLUs/Classes/Workstation.shtml#NhlTColorIndex

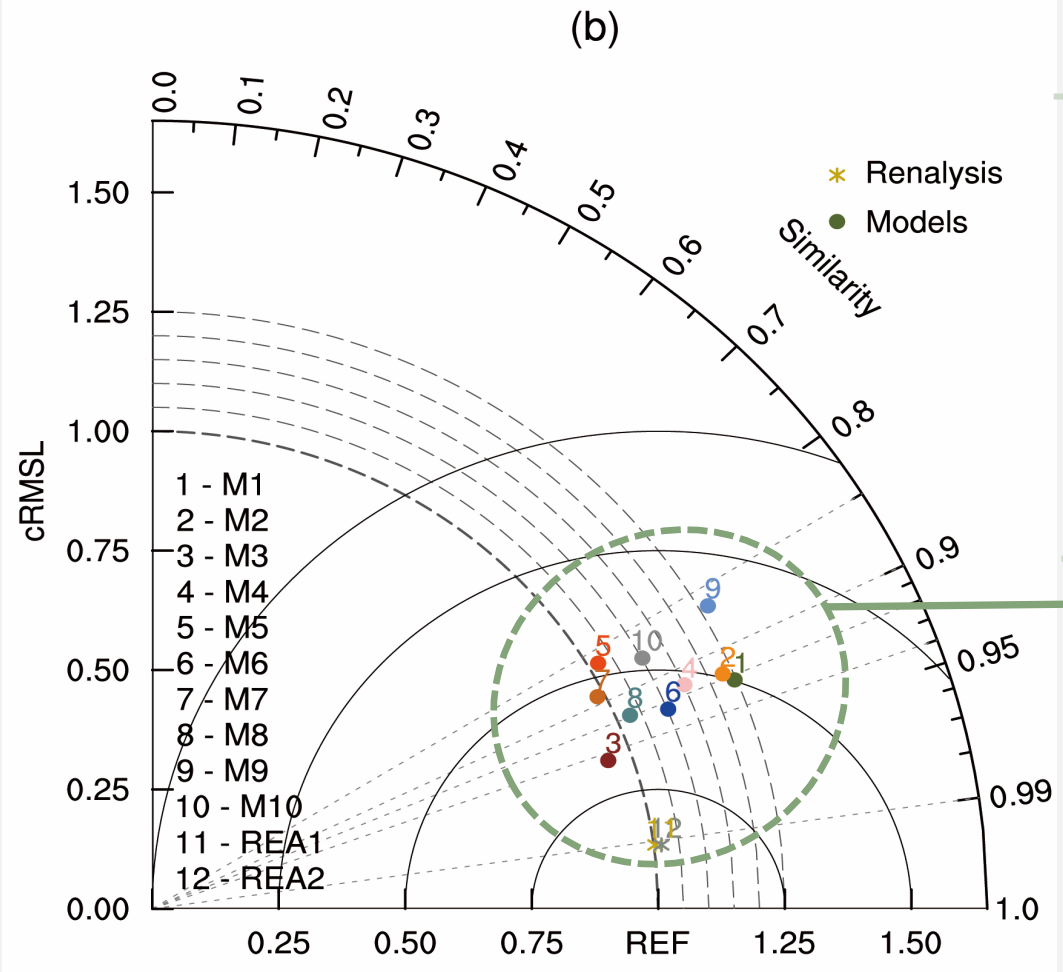
MarkersOn

Valid when set 'True'

whether to plot markers for models and/or obs

if False, only use count numbers to represent models and/or obs

- **Markers** *marker types for models or/and obs points*
  All models (obs) points use the same Marker Index.
- **gsMarkerThicknessF** *thickness for markers*
- **gsMarkerSizeF** *size for markers*



Colors

colors for model or/and obs points

Three ways to set N colors:

I. 1-D string array / list(tuple) consisting of str using named color
II. 1-D int array / list(tuple) consisting of int using NhlTColorIndex
III. N x 3 (N x 4) RGB (RGBA) array / N-len list(tuple) consisting of 3(4)-len list(tuple) representing RGB (RGBA)

Colors are assigned to points of cases (model+obs) in order. If num of colors is less than num of cases, the tool uses one color to represent all models (obs) points.

CountTxOn

whether to add count-text for model/obs points

e.g., 1–12 in left diagram

when MarkersOn=False, tool will make CountTxOn=True

Valid when set 'True'

- **CountFontHeight** *font height for count of points*
- **CountTxXOffset / CountTxYOffset** *distance between point and count number in X/Y-axis*

when MarkersOn=False, tool will set to 0

VFE diagram: opt_VFE

plot_caselabels

whether to plot case labels

If set to **True**, list all the names of cases with their count number in the diagram.

Valid when set 'True'

caseLabelsXLoc
caseLabelsYLoc

X (Y) start location of case labels

caseLabelsYinterval

Y-interval between two cases in caselabels

caseLabelsFontHeightF

font height for caselabels

plot_DatatypeLegendon *whether to plot legend for markers*

Valid when set 'True'

DatatypeLegend

content of legend for data sets

e.g. **(/"Models","Reanalysis"/)** / **["Models","Reanalysis"]**

DatatypeLegendWidth / DatatypeLegendHeight

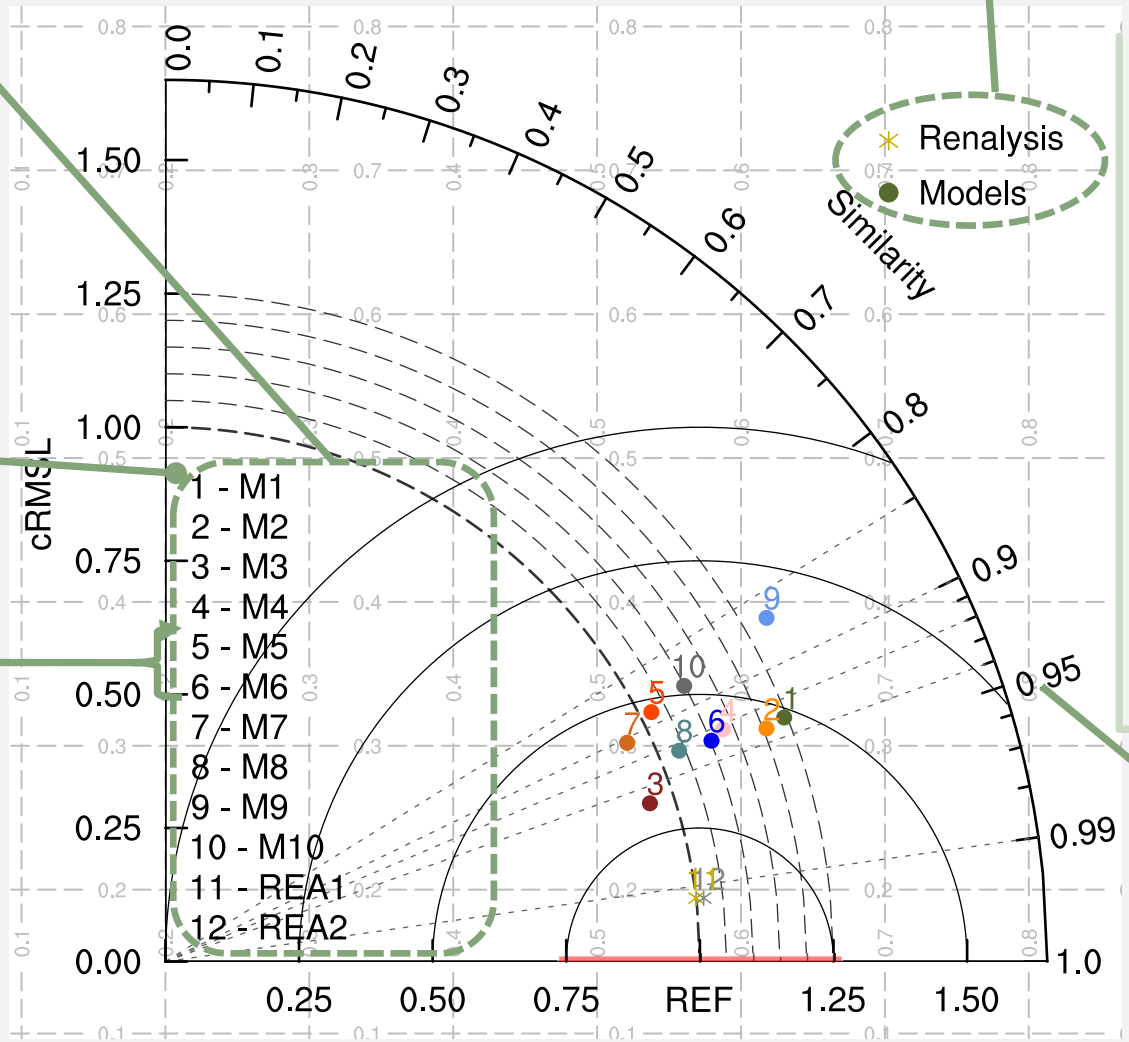
Width / Height of data type legend

DatatypeLegendXL / DatatypeLegendYL

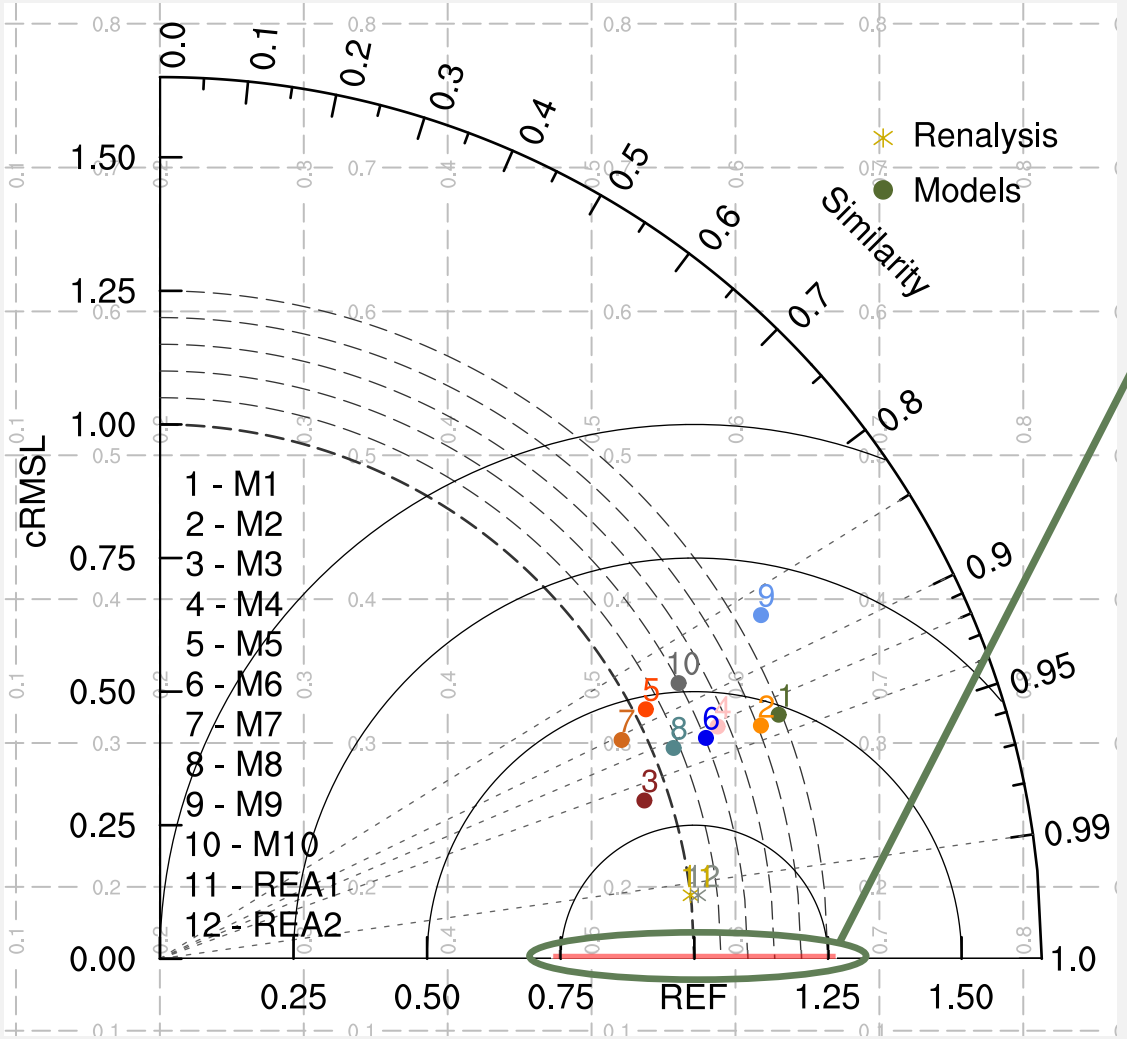
X /Y strat location for legend

draw_grid

whether to draw grid on workstation in order to get location



VFE diagram: opt_VFE



obsUncer

a value measuring observational uncertainty when there is not only one observations and/or reanalysis in VFE diagram

If this argument is set, MVIETool use this value as the length to plot a red horizonal bar centered at (1,0) on X-axis in VFE diagram, which can represent observational uncertainty.

In the example (Sec 5.2) of paper '*An improved multivariable integrated evaluation method and NCL code for multimodel intercomparison (MVIETool version 1.0)*' (Zhang et al., 2021), **the area-weighted mean of standard deviations** (M_{SD}) derived from multiple observations is used as the estimation of the observational uncertainty **in the evaluation of climatological mean spatial field**.

The data file and scripts for generating M_{SD} in the example is available at *https://github.com/Mengzhuo-Zhang/MVIETool* from the GitHub repository.

2. How to set arguments for plotting?

Metrics table

metrics table *is the table to show various statistics of model evaluation for numerous models. The color filled in each grid cell represents the value of statistics.*

level and **colormap** is to help select filling colors for statistics. **level** is a monotonic 1-D array of numerics values and **colormap** should match wih **level**, indicating number of colors must be greater than len of **level** plus 1. If **level** uses default, **colormap** would use default automatically whether **colormap** set or not.

Due to the ranges vary from one statistics to another and different standards of better perforamce in statistics, statistics are divided into **4 groups**, having individual **level** and **colormap**.

Statistic Names	Range of Statitstics	Standard of better performance	Arguments
ME, VME, MEVM, MEVD	numeric values	closer to 0	`MElevels`, `MEcmap`
RMSD, RMSVD, rms_std, cRMSD, cRMSVD, SD_std	positive values	closer to 0	`RMSDlevels`, `RMSDcmap`
rms, RMSL, SD, cRMSL	positive values	closer to 1	`RMSlevels`, `RMScmap`
uCORR, VSC, CORR, cVSC, uMISS, cMISS	0-1	closer to 1	`CORRlevels`, `CORRcmap`

Metrics table

	NCL version	Python version
level	level should be set to 1-D monotonic array of numeric values, e.g.: `(/0.1,0.2,0.3,0.4,0.5/)`	level should be set to list or tuple consisting of int/float and monotonic with increasing index, e.g.: `[0.1,0.2,0.3,0.4,0.5]`
colormap	**[1]** Set to a string using colorfile names, and tool will get colors from this colorfile equalspacedly according to the level. e.g.: `"cmocean_curl"` **[2]** Set to 1-D array of NhlTColorIndex. e.g.: `(/1,2,3,4,5/)` **[3]** Set to 1-D array of named-color strings. e.g.: `(/"red","blue","green"/)` **[4]** Set to N x 3 (N x 4) RGB (RGBA) array of numeric values, where N is the number of colors. e.g.: `(/(/0.5,0.5,0.4/),(/0.2,0.4,0.8/),(/0.9,0.1,0.2/)/)` **[5]** Set to 1-D array of named-color string and number-string for transition between two colors, which has format as (/'color$_1$', 'N$_1$', 'color$_2$','N$_2$',...'color$_{K-1}$','N$_{K-1}$','color$_K$'/), where N$_{k-1}$ is the number of transition colors between color$_{k-2}$ to color$_{k-1}$, and it provides (N$_1$+N$_2$+...+N$_{K-1}$)-(K-2) colors in all. e.g.: `(/"white","3","red","5","blue","3","green"/)`	**[1]** Set to list/tuple consisting of named-color strings, e.g.: `["red","blue","green"]` **[2]** Set to list/tuple consisting of NhlTColorIndex (int), e.g.: `[1,2,3,4,5]` **[3]** Set to string using colormap name, and the tool will get colors from setted color table equalspacedly according to the level, e.g.: `"cmocean_curl"` **[4]** Set to list or tuple consisting of RGB/RGBA-list/tuple, e.g.: `[[0.5,0.5,0.5],[0.1,0.1,0.1],[0.4,0.5,0.3]]`

Ref

Names colors *http://www.ncl.ucar.edu/Document/Graphics/named_colors.shtml*
http://www.pyngl.ucar.edu/Graphics/rgb.txt

Color-file name *http://www.ncl.ucar.edu/Document/Graphics/color_table_gallery.shtml*
http://www.pyngl.ucar.edu/Graphics/color_table_gallery.shtml

NhlTColorIndex *http://www.ncl.ucar.edu/Document/HLUs/Classes/Workstation.shtml#NhlTColorIndex*

*An example for **level** and **colormap***

*To distinguish **levels** and **colormaps** for different types of statistics, **black thick lines** are plotted between statistics with different colormaps.*

METRICS		M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	REA1	REA2
ME	SLP	0.106	0.068	0.025	0.041	0.395	0.711	-0.397	0.014	0.364	-0.059	0.004	-0.004
	Q600	0.034	0.043	0.008	-0.010	-0.094	0.013	0.074	0.093	0.059	-0.011	-0.002	0.002
	SST	0.371	0.296	-0.080	0.059	0.296	0.038	-0.301	0.018	0.019	0.321	-0.061	0.061
	T850	-0.026	0.026	-0.022	-0.077	-0.139	-0.151	-0.037	0.114	0.005	-0.037	0.009	-0.009
	uv850	0.072	0.046	0.038	0.025	0.067							
	uv200	0.097	0.055	0.013	0.175	0.010							
VME		0.125	0.095	0.035	0.091	0.118	0.089	0.111	0.110	0.096	0.121	0.020	0.020
cRMSD	SLP	0.563	0.758	0.414	0.499	0.598	0.451	0.406	0.546	0.595	0.486	0.072	0.072
	Q600	0.137	0.114	0.126	0.147	0.215	0.134	0.151	0.136	0.163	0.216	0.063	0.063
	SST	0.468	0.445	0.258	0.316	0.410	0.387	0.297	0.352	0.395	0.626	0.100	0.100
	T850	0.097	0.103	0.092	0.135	0.126	0.123	0.132	0.102	0.152	0.146	0.037	0.037
	uv850	0.483	0.454	0.311	0.437	0.476	0.414	0.432	0.396	0.655	0.517	0.119	0.119
	uv200	0.274	0.256	0.256	0.317	0.328							
cRMSVD		0.352	0.331	0.242	0.326	0.360	0.310	0.342	0.303	0.452	0.400	0.086	0.086
SD_std		0.129	0.167	0.029	0.106	0.092	0.069	0.056	0.065	0.152	0.156	0.017	0.017
SD	SLP	1.275	1.400	1.010	1.171	0.964	1.182	0.999	1.106	1.210	1.196	1.023	0.981
	Q600	1.048	1.030	0.970	1.024	1.136	0.977	0.979	0.963	0.907	0.944	0.975	1.028
	SST	1.292	1.276	0.982	0.960	1.223	1.126	0.870	1.139	1.019	1.352	1.017	0.993
	T850	0.998	0.946	0.989	1.072	1.035	1.052	1.010	0.970	0.890	0.901	1.012	0.990
	uv850	1.245	1.206	0.954	1.130	1.022							
	uv200	1.005	0.977	0.920	0.856	0.977							
cRMSL		1.127	1.094	0.958	1.036	1.050	1.059	0.984	1.028	1.085	1.043	0.997	1.010
CORR	SLP	0.906	0.852	0.915	0.906	0.815	0.928	0.917	0.870	0.872	0.917	0.998	0.998
	Q600	0.992	0.994	0.992	0.990	0.988	0.991	0.989	0.991	0.990	0.977	0.998	0.998
	SST	0.948	0.952	0.966	0.949	0.952	0.941	0.959	0.954	0.924	0.901	0.995	0.995
	T850	0.995	0.996	0.996	0.994	0.993	0.994	0.991	0.995	0.994	0.994	0.999	0.999
	uv850	0.930	0.932	0.951	0.923	0.890	0.927	0.904	0.925	0.868	0.883	0.993	0.993
	uv200	0.963	0.967	0.968	0.953	0.945							
cVSC		0.952	0.954	0.970	0.949	0.940							
cMISS		0.960	0.960	0.980	0.963	0.957	0.968	0.959	0.969	0.934	0.943	0.997	0.998
uMISS		0.979	0.982	0.991	0.983	0.978	0.984	0.978	0.985	0.967	0.972	0.999	0.999

Colormap Method [1]

Colormap Method [3]

Colormap Method [2]

Colormap Method [4]

```
opt_metrics@MElevs = (/-0.3,-0.25,-0.2,-0.15,-0.1,-0.075,-0.05,-0.025,-0.01,0.,0.01,0.025,0.05,0.075,\
   0.1,0.15,0.2,0.25,0.3,0.4,0.5,0.6,0.7/)
opt_metrics@MEcmap = "MPL_BuPu"

opt_metrics@RMSDlevs = (/0.05,0.1,0.15,0.2,0.25,0.3,0.35,0.4,0.45,0.5,0.6,0.7/)
opt_metrics@RMSDcmap = (/"white","lightgoldenrodyellow","lightgoldenrod1","gold1","darkgoldenrod1",\
   "lightsalmon","salmon","orange","tomato","lightseagreen","lightslateblue","lightskyblue",\
      "lightslateblue"/)

opt_metrics@RMSlevs = (/0.85,0.9,0.95,0.99,0.995,0.999,1.0,1.001,1.005,1.01,1.05,1.1,1.15,1.25,1.3,1.35/)
opt_metrics@RMScmap = (/2132898229,2135069383,2136645837,2138619862,2141904102,2145121791,2146499839,\
   2147479797,2147475680,2147269546,2147258994,2147183178,2146450475,2144081950,2141589268,\
      2138573327,2136932352/)

opt_metrics@CORRlevs = (/0.85,0.9,0.92,0.94,0.96,0.98,0.99,0.995,0.998/)
opt_metrics@CORRcmap = (/(/0.493,0.568,0.266/),(/0.578,0.613,0.315/),(/0.659,0.654,0.368/),\
  (/0.704,0.663,0.425/),(/0.750,0.677,0.487/),(/0.795,0.696,0.553/),(/0.840,0.723,0.623/),\
     (/0.886,0.759,0.697/),(/0.931,0.805,0.775/),(/0.977,0.865,0.857/)/)
```

NCL

```
opt_metrics.MElevs = [-0.3,-0.25,-0.2,-0.15,-0.1,-0.075,-0.05,-0.025,-0.01,0.,0.01,0.025,0.05,0.075,\
   0.1,0.15,0.2,0.25,0.3,0.4,0.5,0.6,0.7]
opt_metrics.MEcmap = "MPL_BuPu"

opt_metrics.RMSDlevs = [0.05,0.1,0.15,0.2,0.25,0.3,0.35,0.4,0.45,0.5,0.6,0.7]
opt_metrics.RMSDcmap = ["white","lightgoldenrodyellow","lightgoldenrod1","gold1", "darkgoldenrod1",\
   "lightsalmon","salmon","orange","tomato","lightseagreen", "lightslateblue","lightskyblue",\
      "lightslateblue"]

opt_metrics.RMSlevs = [0.85,0.9,0.95,0.99,0.995,0.999,1.0,1.001,1.005,1.01,1.05,1.1,1.15,1.25,1.3,1.35]
opt_metrics.RMScmap = [2132898229,2135069383,2136645837,2138619862,2141904102,2145121791, 2146499839,\
   2147479797,2147475680,2147269546,2147258994,2147183178,2146450475,2144081950, 2141589268,\
      2138573327,2136932352]

opt_metrics.CORRlevs = [0.85,0.9,0.92,0.94,0.96,0.98,0.99,0.995,0.998]
opt_metrics.CORRcmap = [[0.493,0.568,0.266],[0.578,0.613,0.315],[0.659,0.654,0.368],[0.704,0.663,0.425]\
   ,[0.750,0.677,0.487],[0.795,0.696,0.553],[0.840,0.723,0.623],[0.886,0.759,0.697],\
      [0.931,0.805,0.775],[0.977,0.865,0.857]]
```

Python3

➢ ***Colormap default**: better model performance with lighter color*

Metrics table: opt_metrics

Design of metrics table: **CaseLocation**

CaseLocation = "Top"

caselabels on the top and statstic label in the left

when the number of **statistics** shown is relative larger than that of cases

caseWidth

caseHeight

zoom

varWidth

	METRICS	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	REA1	REA2
SD	SLP	1.269	1.396	1.024	1.166	0.965	1.172	1.004	1.100	1.199	1.171	1.025	0.984
	SST	1.041	1.031	0.992	1.020	1.135	0.979	0.979	0.969	0.909	0.962	0.972	1.032
	Q600	1.278	1.270	0.972	0.951	1.223	1.120	0.868	1.134	1.016	1.332	1.014	0.997
	T850	0.997	0.946	0.982	1.081	1.034	1.049	1.011	0.971	0.891	0.907	1.011	0.991
	uv850	1.247	1.212	0.952	1.137	1.022	1.104	0.965	1.037	1.295	1.094	1.001	1.015
	uv200	1.000	0.976	0.916	0.851	0.977	1.009	1.047	1.051	0.985	0.984	0.986	1.016
cRMSL		1.125	1.096	0.959	1.040	1.050	1.058	0.986	1.026	1.085	1.041	0.997	1.012

CaseLocation = "Left"

caselabels at the left and statstic label on the top

when the number of **cases** shown is relative larger than that of statistics

titleWidth

titleHeight

caseHeight

zoom

METRICS	cRMSD						cRMSVD
	SLP	SST	Q600	T850	uv850	uv200	
M1	0.566	0.144	0.463	0.098	0.490	0.269	0.353
M2	0.751	0.115	0.442	0.104	0.464	0.252	0.335
M3	0.405	0.134	0.272	0.097	0.313	0.254	0.244
M4	0.495	0.154	0.321	0.202	0.447	0.315	0.336
M5	0.599	0.216	0.409	0.127	0.476	0.328	0.360
M6	0.443	0.138	0.387	0.125	0.420	0.265	0.312
M7	0.401	0.153	0.299	0.136	0.445	0.405	0.349
M8	0.531	0.138	0.349	0.108	0.400	0.296	0.303
M9	0.582	0.165	0.394	0.157	0.658	0.344	0.452
M10	0.477	0.217	0.619	0.157	0.527	0.294	0.403
REA1	0.097	0.069	0.105	0.043	0.127	0.048	0.091
REA2	0.097	0.069	0.105	0.043	0.127	0.048	0.091

Metrics table: opt_metrics

varTHeight
font height of variable names

statsTHeight
font height of statistic names

tableTitle
title for metrics table

titleTHeight
font height of table title

TitleBackgroundcolor
background color for table title

METRICS	cRMSD						cRMSVD
	SLP	SST	Q600	T850	uv850	uv200	
M1	0.566	0.144	0.463	0.098	0.490	0.269	0.353
M2	0.751	0.115	0.442	0.104	0.464	0.252	0.335
M3	0.405	0.134	0.272	0.097	0.313	0.254	0.244
M4	0.495	0.154	0.321	0.202	0.447	0.315	0.336
M5	0.599	0.216	0.409	0.127	0.476	0.328	0.360
M6	0.443	0.138	0.387	0.125	0.420	0.265	0.312
M7	0.401	0.153	0.299	0.136	0.445	0.405	0.349
M8	0.531	0.138	0.349	0.108	0.400	0.296	0.303
M9	0.582	0.165	0.394	0.157	0.658	0.344	0.452
M10	0.477	0.217	0.619	0.157	0.527	0.294	0.403
REA1	0.097	0.069	0.105	0.043	0.127	0.048	0.091
REA2	0.097	0.069	0.105	0.043	0.127	0.048	0.091

caseTHeight
font height of caselabels

VarBackgroundcolor
background color for caselabels / statistics labels/ var labels

metricsTHeight
font height of statistic value in each square

Users can decide which statistics are shown in the metric table by editing the **stats_metrics2**. It is an **1-D array** or **list/tuple** of string with statistics names. Specially, statistics for summary, i.e., cMISS, uMISS, can be enclosed with red thick box for emphasis, with **highlightsummary** for **opt_metrics**. e.g:

METRICS	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	REA1	REA2
cVSC	0.951	0.953	0.970	0.946	0.940	0.956	0.938	0.955	0.909	0.923	0.996	0.996
cMISS	0.960	0.960	0.979	0.961	0.957	0.968	0.958	0.969	0.934	0.943	0.997	0.997
uMISS	0.979	0.982	0.990	0.982	0.978	0.984	0.978	0.985	0.967	0.972	0.999	0.999

NCL

```
stats_metrics2          = (/"cVSC","cMISS","uMISS"/)
opt_metrics@showTextOn          = True
opt_metrics@decialMetricsN      = "4.3"
opt_metrics@highlightsummary    = True
```

Python3

```
stats_metrics2          = ["cVSC","cMISS","uMISS"]
opt_metrics.showTextOn          = True
opt_metrics.decialMetricsN      = "4.3"
opt_metrics.highlightsummary    = True
```

If there are too many cases and statistics needing to be shown in the table, statistic values can be omitted in table with **showTextOn**. In this case, each grid cell is only filled with color and then the colorbar will be drawn by the tool. Besides, the format of values marked in the colorbars also can be set with **decialBarN** of **opt_metrics**. **decialMetricsN**/ **decialBarN** is a string like "A.B", where A(B) indicates showing number of digits (after the decimal point) for the value. **colorbarFHeight** of **opt_metrics** is used to set font height of colorbar markers.

If we choose to show statistics, the format of statistics shown can be set with **decialMetricsN** for **opt_metrics** as well.

When **ME** shown in the table, if the datafile has **MEVD**, **MEVD** of 2-D individual variables can be chosen to plot near its **ME** with **plotMEVD** of **opt_metrics**. **MEVD** dividing 180 is used to get filling color in MEcmap.

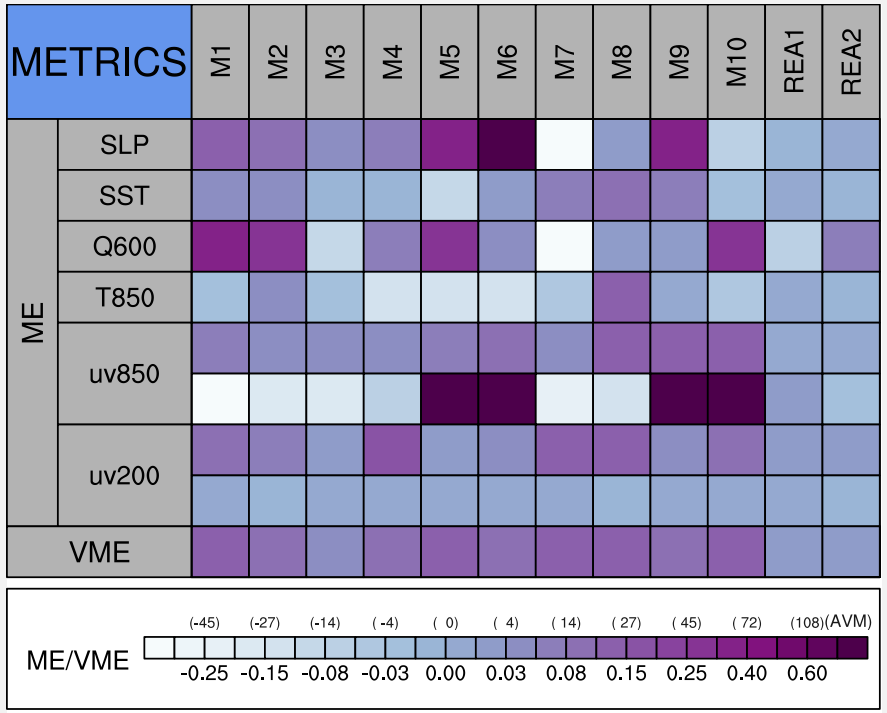
e.g:

NCL

```
opt_metrics@showTextOn    = False
opt_metrics@decialBarN    = "3.2 "
opt_metrics@plotMEVD      = True
```

Python3

```
opt_metrics.showTextOn    = False
opt_metrics.decialBarN    = "3.2 "
opt_metrics.plotMEVD      = True
```



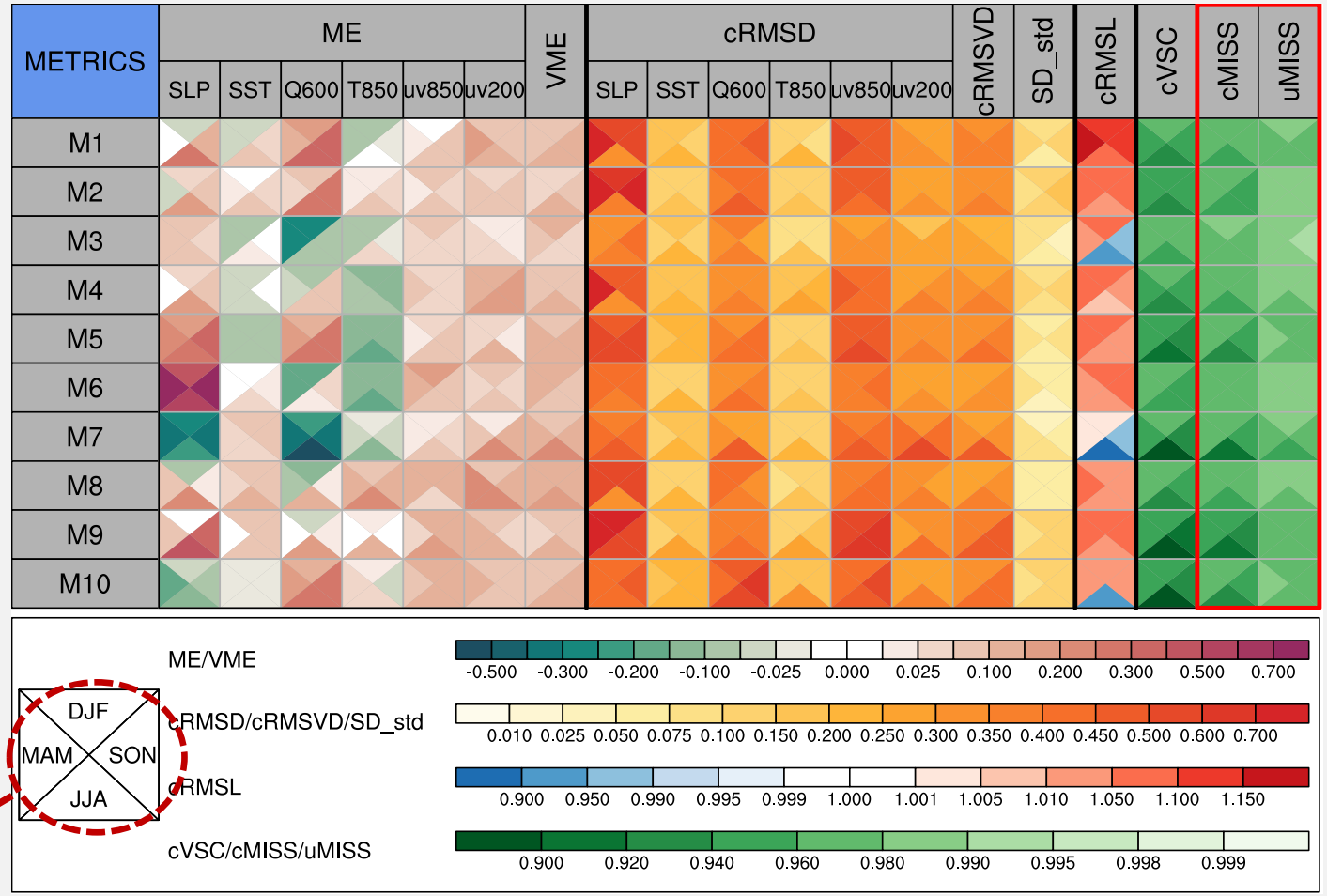
Metrics table: opt_metrics

Commonly, statistics from different evaluation objects, e.g., different seasons, different periods, can be shown in a table at the same time, by setting **MVIE_filenames2** with some datafile names. Note that the number of datafiles is up to 4.

each square divided into 4 triangles

In this case, each grid cell of table is divided into some parts and only filled with colors without values. **stats_metrics2** should be the common statistics of all datafiles.

Box legend is provided with **box_legend** for **opt_metrics** and it is set to **an array of strings** / **list or tuple consisting of str** that matches with the number of datafiles. Its default is not showing box legend. The height of box legend can be set with **box_lgheight** of **opt_metrics** (ratio of its height to width matches with grid cell in the table) and font height can be set with **box_lgFontHeightF** of **opt_metrics**.



Box legend

*Acronyms of statistics mentioned in **User Guide** and description*

Acronyms	Description	Acronyms	Description
SD	Standard deviation	*cRMSL*	Centered root mean square length
CORR	Correlation coefficient	*cVSC*	Centered vector similarity coefficient
cRMSD	Centered root mean square difference	*cRMSVD*	Centered root mean square vector difference
rms	Root mean square	*SD_std*	Standard deviation of SD values
uCORR	Uncentered correlation coefficient	*RMSL*	Root mean square length
RMSD	Root mean square difference	*VSC*	Vector similarity coefficient
ME	Mean error	*RMSVD*	Root mean square vector difference
cMISS	Centered multivariable integrated skill score	*rms_std*	Standard deviation of rms values
		VME	Vector mean error
uMISS	Uncentered multivariable integrated skill score	*MEVM*	Mean error of vector magnitude
		MEVD	Mean error of vector direction

NOTE: *Input statistics* should be an array with dimension size (shape) as [Ncase] x [Nvar],
Input statistics should be an array with dimension size (shape) as [Ncase]