
SMEFT19 Documentation

Release 2.0

Jorge Alda

Nov 08, 2020

CONTENTS:

1	SMEFTglob	1
2	ellipse	3
3	obsuncert	5
4	comparepulls	7
5	plots	9
6	scenarios	13
7	ml	17
8	Indices and tables	19
	Python Module Index	21
	Index	23

SMEFTGLOB

Common functions used to calculate likelihood values and pulls of the fits.

`SMEFT19.SMEFTglob.likelihood_fits(x, wfun)`

Calculates the log-likelihood of a NP hypothesis for several classes of observables.

Arguments

- `x`: Point in parameter space to be evaluated.
- `wfun`: Function that takes a point in parameter space and returns a dictionary of Wilson coefficients.

Returns

- A dictionary of log-likelihoods, for each of the classes of observables defined by *smelli*.

`SMEFT19.SMEFTglob.likelihood_global(x, wfun)`

Calculates the global log-likelihood of a NP hypothesis.

Arguments

- `x`: Point in parameter space to be evaluated.
- `wfun`: Function that takes a point in parameter space and returns a dictionary of Wilson coefficients.

Returns

- The global log-likelihood.

`SMEFT19.SMEFTglob.loadobslist()`

Loads from a *.yaml* file a list of all observables available, ordered by their pull in the SM. If the file does not exist, this functions creates it.

Returns

- A list with all observables available.

`SMEFT19.SMEFTglob.prediction(x, obs, wfun)`

Interfaces *flavio* to compute the NP prediction of a given observable.

Arguments

- `x`: Point in parameter space to be evaluated.
- `obs`: observable, as defined by *flavio*, whose prediction will be computed. If the observable does not depend on any parameter, `obs` is a string. If the observable depends on numerical parameters (such as `q2`), `obs` is a list containing a string and one or more floats.
- `wfun`: Function that takes a point in parameter space and returns a dictionary of Wilson coefficients.

Returns

- The prediction of the observable.

`SMEFT19.SMEFTglob.pull_obs(x, obs, wfun)`

Calculates the pull, in sigmas, of the prediction of a given observable in NP with respect to its experimental value.

Arguments

- `x`: Point in parameter space to be evaluated.
- `obs`: observable, as defined by *flavio*, whose prediction will be computed. If the observable does not depend on any parameter, `obs` is a string. If the observable depends on numerical parameters (such as `q2`), `obs` is a list containing a string and one or more floats.
- `wfun`: Function that takes a point in parameter space and returns a dictionary of Wilson coefficients.

Returns

- The pull of the observable.

ELLIPSE

Assuming that the likelihood of the fit follows a gaussian distribution (Central Limit Theorem), and therefore the log-likelihood is characterized by a quadratic form around the minimum, this script finds this quadratic form, and parametrizes (ellipsoidal) sections of constant likelihood.

`SMEFT19.ellipse.load(filename)`

Loads an ellipse saved in a *.yaml* file to a python dictionary.

Arguments

- filename: Path to the *.yaml* file where the shape of the ellipse has been saved by the *save* method.

Returns

A python dictionary containing:

- bf: *np.array* with the point in parameter space with the best fit.
- v: *np.matrix* containing the orientation of the axes of the ellipsoid.
- d: *np.array* containing the principal axes of the ellipsoid.
- L: Log-likelihood at the best fit point.
- [name: Name of the fit.]
- [fit: Scenario used in the fit.]

`SMEFT19.ellipse.minimum(fit, x0)`

Finds the minimum of the fit function and approximates its neighbourhood by an ellipsoid.

Arguments

- fit: function that takes one point in parameter space and returns its negative log-likelihood.
Example: `-SMEFTglob.likelihood_global(x, scenarios.scVI)`.
- x0: list or *np.array* containing an initial guess.

Returns

- bf: *np.array* with the point in parameter space with the best fit.
- v: Unitary matrix containing the axes of the ellipse.
- d: diagonal matrix containing the inverse of the squares of the semiaxes.
- Lmin: Log-likelihood at the best fit point.

`SMEFT19.ellipse.notablepoints` (*fin, fout, fit*)

Finds the extrema of the ellipse, the intersection with the coordinate axis and the closest and furthest point from the origin.

Arguments

- *fin*: Path to *.yaml* file containing the information about the ellipse.
- *fout*: Path to *.tex* file to save a table with the coordinates of the notable points.
- *fit*: Function used in the minimization.

`SMEFT19.ellipse.parametrize` (*x, bf, v, d, nsigmas=1*)

Maps points on the unit hypersphere to points on the ellipsoid of constant likelihood.

Arguments

- *x*: *np.array* containing a point in the surface of the unit *n*-hypersphere.
- *bf*: *np.array* with the point in parameter space with the best fit.
- *v*: *np.matrix* containing the orientation of the axes of the ellipsoid.
- *d*: *np.array* containing the principal axes of the ellipsoid.
- [*nsigmas*: significance of the isoprobability hypersurface with respect to the best fit.]

Returns

- *xe*: Projection of the point *xe* in the ellipsoid of equal probability

`SMEFT19.ellipse.save` (*bf, v, d, L, filename, name=None, fit=None*)

Saves the results of the minimization in a *.yaml* file.

Arguments

- *bf*: *np.array* with the point in parameter space with the best fit.
- *v*: *np.matrix* containing the orientation of the axes of the ellipsoid.
- *d*: *np.array* containing the principal axes of the ellipsoid
- *filename*: Path to the *.yaml* file where the shape of the ellipse will be saved.
- *L*: Log-likelihood at the best fit point.
- [*name*: Descriptive name of the fit.]
- [*fit*: scenario used to fit the data.]

OBSUNCERT

Module used to compute the uncertainty of some observables using a MonteCarlo analysis.

`SMEFT19.obsuncert.calculate` (*wfun*, *minx*, *maxx*, *fout*, *bf*, *name*, *num*=50, *cores*=1)

Computes the central value and uncertainty of a selection of observables, using a MonteCarlo analysis. The observables are $\$R_{\{K^{(*)}\}}\$$ and $\$R_{\{D^{(*)}\}}\$$, and can be modified by editing the variable `obsuncert.obslist`.

Arguments

- *wfun*: Function that takes a point in parameter space and returns a dictionary of Wilson coefficients.
- *minx*: Minimum of the search region. If the fit is multidimensional, *minx* is a list containing the minimum of the search region in each direction.
- *maxx*: Maximum of the search region. If the fit is multidimensional, *maxx* is a list containing the maximum of the search region in each direction.
- *fout*: Path to the *.yaml* file where the statistical values will be saved.
- *bf*: Coordinates of the best fit point.
- *name*: Name of the fit.
- [*num*: Number of MonteCarlo points used to compute the uncertainty. Default=50.]
- [*cores*: number of cores used to parallel-compute the uncertainty. Default=1 (no parallelization).]

COMPAREPULLS

This module contains several functions used to compare between different NP scenarios and the Standard Model.

`SMEFT19.comparepulls.compare` (*wfun, fin, fout*)

Lists the comparison between the pull of each observable in the NP hypothesis and the SM.

Arguments

- *wfun*: Function that takes a point in parameter space and returns a dictionary of Wilson coefficients.
- *fin*: Path to the file *.yaml* where the ellipsoid is saved.
- *fout*: Path to the *.tex* file where the comparison table will be written. The observables are ordered by their SM pull, and are shaded in green if the NP improves this pull and in red otherwise.

`SMEFT19.comparepulls.notablepulls` (*wfun, fin, fout*)

Determines the observables whose pull changes the most between the best fit and the notable points of the ellipsoid.

Arguments

- *wfun*: Function that takes a point in parameter space and returns a dictionary of Wilson coefficients.
- *fin*: Path to the file *.yaml* where the ellipsoid is saved.
- *fout*: Path to the *.tex* file where the comparison table will be written.

`SMEFT19.comparepulls.pointpull` (*x, wfun, bf, printlevel=1, numres=5*)

Determines the observable whose pull changes the most between two NP hypothesis.

Arguments

- *x*: Point in space parameter of the tested NP hypothesis.
- *wfun*: Function that takes a point in parameter space and returns a dictionary of Wilson coefficients.
- *bf*: Point in space parameter of the reference NP hypothesis (e.g. the best fit).
- [*printlevel*: 0 for silent mode, 1 for verbose mode.]
- [*numres*: Number of observables displayed. Default=5.]

Returns

- A multi-line string. Each line contains the id number of the observable, its name and the squared difference of the pulls.

SMEFT19.comparepulls.**pullevolution**(*obscode*, *wfun*, *fin*, *direction*)

Calculates the variation of the pull along a line connecting two opposite notable points of the ellipsoid.

Arguments

- *obscode*: ID-Number of the observable, as returned by comparepulls.pointpull
- *wfun*: Function that takes a point in parameter space and returns a dictionary of Wilson coefficients.
- *fin*: Path to the file .yaml where the ellipsoid is saved.
- *direction*: string with the following format:
 - ‘wc’ + str(i): for the i-th Wilson coefficient.
 - ‘ax’ + str(i): for the i-th principal axis of the ellipsoid.
 - ‘sm’: for the direction joining the bf and sm points.

PLOTS

This module contains functions to plot the results of the fits.

`SMEFT19.plots.compare_plot` (*wfun, fin, fout*)

Plots the pull of each observable in the SM and in the NP hypothesis.

Arguments

- *wfun*: Function that takes a point in parameter space and returns a dictionary of Wilson coefficients.
- *fin*: Path to the *.yaml* file where the ellipsoid is saved.
- *fout*: Path to the files where the plots will be saved. Two files are created, one *.pdf* and one *.pgf* (to use in TeX). Extensions are added automatically.

`SMEFT19.plots.error_plot` (*flist, plotype, fout*)

Plots the uncertainty intervals for several observables in NP scenarios, SM and experimental values.

Arguments

- *flist*: List of paths to files created by *obsuncert.calculate*.
- *plotype*: Selects the observables to be plotted:
 - ‘RK’: Plots RK in the [1.1,6.0] bin and RK* in the [0.045,1.1] and [1.1,6] bins.
 - ‘RD’: Plots RD, and RD* using only muons or muons+electrons.
- *fout*: Path to the files where the plots will be saved. Two files are created, one *.pdf* and one *.pgf* (to use in TeX). Extensions are added automatically.

`SMEFT19.plots.evolution_plot` (*obscodes, wfun, fin, direction, fout*)

Plots the variation of the pull of several observables along a line connecting two opposite notable points of the ellipsoid.

Arguments

- *obscodes*: List of ID-Numbers of the observables, as returned by *comparepulls.pointpull*
- *wfun*: Function that takes a point in parameter space and returns a dictionary of Wilson coefficients.
- *fin*: Path to the *.yaml* file where the ellipsoid is saved.
- *direction*: string with the following format:
 - ‘ax’ + str(i): for the i-th principal axis of the ellipsoid.
 - ‘sm’: for the direction joining the bf and sm points.

- `fout`: Path to the files where the plots will be saved. Two files are created, one *.pdf* and one *.pgf* (to use in TeX). Extensions are added automatically.

`SMEFT19.plots.hatch_contour` (*x*, *y*, *z*, *levels*, *interpolation_factor*=1, *interpolation_order*=2, *col*=0, *label*=None, *hatched*=True, *contour_args*={}, *contourf_args*={})

Plots coloured and hatched confidence contours (or bands) given numerical input arrays. Based on the *flavio* function

Arguments

- *x*, *y*: 2D arrays containing *x* and *y* values as returned by `numpy.meshgrid`
- *z*: value of the function to plot. 2D array in the same shape as *x* and *y*. The lowest value of the function should be 0 (i.e. the best fit point).
- *levels*: list of function values where to draw the contours. They should be positive and in ascending order.
- [*interpolation_factor*:: in between the points on the grid, the function can be interpolated to get smoother contours. This parameter sets the number of subdivisions (default: 1, i.e. no interpolation). It should be larger than 1.]
- [*col*: number between 0 and 9 to choose the color of the plot from a predefined palette.]
- [*label*: label that will be added to a legend created with `matplotlib.pyplot.legend()`.]
- [*contour_args*: dictionary of additional options that will be passed to `matplotlib.pyplot.contour()` (that draws the contour lines).]
- [*contourf_args*: dictionary of additional options that will be passed to `matplotlib.pyplot.contourf()` (that paints the contour filling).]

`SMEFT19.plots.likelihood_plot` (*wfun*, *xmin*, *xmax*, *ymin*, *ymax*, *fits*, *axlabels*, *fout*=None, *locleg*=0, *n_sigma*=(1, 2), *steps*=55, *hatched*=False, *threads*=1, *bf*=None)

Plots a contour plot of the log-likelihood of the fit.

Arguments

- *wfun*: Function that takes a point in parameter space and returns a dictionary of Wilson coefficients.
- *xmin*: Minimum value of the *x* coordinate.
- *xmax*: Maximum value of the *x* coordinate.
- *ymin*: Minimum value of the *y* coordinate.
- *ymax*: Maximum value of the *y* coordinate.
- *fits*: List containing the fits, as defined by *smelli*, displayed in the plot. Valid values are:
 - ‘RK’: All LFUV $b \rightarrow s l l$ decays.
 - ‘RD’: All LFUV $b \rightarrow c l \nu$ decays.
 - ‘EW’: All electroweak precision tests.
 - ‘LFV’: All observables violating lepton flavour.
 - ‘ZLFV’: All observables violating flavour in Z decays.
 - ‘global’: All observables implemented by *smelli*.
- *axlabels*: List containing two strings to label the *x* and *y* axes.
- [*fout*: Path to the files where the plots will be saved. Two files are created, one *.pdf* and one *.pgf* (to use in TeX). Extensions are added automatically.]

- [locleg: Position of the legend of the plot, using *matplotlib*'s syntax. Default=0 (best position).]
- [numsigma: List containing the significance (in sigmas) of each contour. Default = (1,2).]
- [steps: number of function calls calculated *in each* direction, that is, the total number of function calls is steps^2 . Default=50.]
- [hatched: Boolean that determines whether the area between contours will be hatched (True) or filled (False). Default: False.]
- [threads: number of parallel threads used to compute the log-likelihood. Default: 1 (no parallelization).]
- [bf: Coordinates of the best fit point(s). It can be *None* (no point marked), a list containing two floats (one point marked) or a list of lists of points (several points marked). Default: *None*.]

SCENARIOS

This module contains all the NP hypothesis that we have considered and some auxiliar functions to implement them.

`SMEFT19.scenarios.idemp(a, b)`

Creates an idempotent hermitic 3x3 matrix using to parameters.

Arguments

- a, b: Parameter of the matrix.

Returns

- A *np.matrix*.

`SMEFT19.scenarios.massrotation(x)`

NP affects only the third generation in the interaction basis and then is rotated to the mass basis. Couplings to the first generation not negligible. C1 and C3 can be different.

Arguments -x : Coordinates in the parameter space of the fit. $x = [C1, C3, \alpha_l, \beta_l, \alpha_q, \beta_q]$.

Returns

- A dictionary containing the SMEFT Wilson Coefficients of the fit.

`SMEFT19.scenarios.rot2lqU1(x, M=1.5)`

Coupling of the U(1) leptoquarks obtained from the Wilson Coefficients.

Arguments

- x: Coordinates in the parameter space of the fit. It assumes scenario BI if $\text{len}(x)==3$ or scenario BII if $\text{len}(x)==5$.
- [M: Mass of the leptoquark, in TeV. Default=1.5.]

Returns

- A *np.matrix* containing the couplings.

`SMEFT19.scenarios.rotBI(x)`

Scenario BI: NP affects only the third generation in the interaction basis and then is rotated to the mass basis. Couplings to the first generation negligible.

Arguments -x : Coordinates in the parameter space of the fit. $x = [C, \beta_l, \beta_q]$.

Returns

- A dictionary containing the SMEFT Wilson Coefficients of the fit.

`SMEFT19.scenarios.rotBII(x)`

Scenario BII: NP affects only the third generation in the interaction basis and then is rotated to the mass basis. Couplings to the first generation not negligible.

Arguments -x : Coordinates in the parameter space of the fit. $x = [C, \alpha_l, \beta_l, \alpha_q, \beta_q]$.

Returns

- A dictionary containing the SMEFT Wilson Coefficients of the fit.

`SMEFT19.scenarios.scI(x)`

Scenario I: NP only affects to electrons.

Arguments

- x: Coordinates in the parameter space of the fit.

Returns

- A dictionary containing the SMEFT Wilson Coefficients of the fit.

`SMEFT19.scenarios.scII(x)`

Scenario II: NP only affects to muons.

Arguments

- x: Coordinates in the parameter space of the fit.

Returns

- A dictionary containing the SMEFT Wilson Coefficients of the fit.

`SMEFT19.scenarios.scIII(x)`

Scenario III: NP only affects to taus.

Arguments

- x: Coordinates in the parameter space of the fit.

Returns

- A dictionary containing the SMEFT Wilson Coefficients of the fit.

`SMEFT19.scenarios.scIV(x)`

Scenario IV: NP only affects to electrons and muons.

Arguments

- x: Coordinates in the parameter space of the fit.

Returns

- A dictionary containing the SMEFT Wilson Coefficients of the fit.

`SMEFT19.scenarios.scIX(x)`

Scenario IX: NP affects to electrons and taus equally, and to muons by an opposite ammount.

Arguments

- x: Coordinates in the parameter space of the fit.

Returns

- A dictionary containing the SMEFT Wilson Coefficients of the fit.

`SMEFT19.scenarios.scV(x)`

Scenario V: NP only affects to electrons and taus.

Arguments

- x: Coordinates in the parameter space of the fit.

Returns

- A dictionary containing the SMEFT Wilson Coefficients of the fit.

`SMEFT19.scenarios.scVI(x)`

Scenario VI: NP only affects to muons and taus.

Arguments

- x: Coordinates in the parameter space of the fit.

Returns

- A dictionary containing the SMEFT Wilson Coefficients of the fit.

`SMEFT19.scenarios.scVII(x)`

Scenario VII: NP affects to electrons, muons and taus.

Arguments

- x: Coordinates in the parameter space of the fit.

Returns

- A dictionary containing the SMEFT Wilson Coefficients of the fit.

`SMEFT19.scenarios.scVIII(x)`

Scenario VIII: NP affects to electrons, muons and taus equally.

Arguments

- x: Coordinates in the parameter space of the fit.

Returns

- A dictionary containing the SMEFT Wilson Coefficients of the fit.

This module contains the functions needed to train a Machine Learning-based Montecarlo scan, and to assess its performance.

`SMEFT19.ml.SHAP_bf` (*fmodel*)

Computes the SHAP values of the best fit point

Arguments

- *fmodel*: Path to the file where the model was saved.

`SMEFT19.ml.SHAP_param` (*fmodel*, *points*, *param*)

Creates an scatter plot displaying how the SHAP values change as functions of each parameter of the fit.

Arguments

- *fmodel*: Path to the file where the model was saved.
- *points*: Pandas Dataframe containing the dataset.
- *param*: Fit parameter. 0 = C, 1 = al, 2 = bl, 3 = aq, 4 = bq.

`SMEFT19.ml.SHAP_summary` (*fmodel*, *points*)

Creates a summary plot of the average SHAP values on a dataset.

Arguments

- *fmodel*: Path to the file where the model was saved.
- *points*: Pandas Dataframe containing the dataset.

`SMEFT19.ml.hist` (*ML*, *vpoints*)

Plots an histogram for the predicted and actual likelihoods, and compares them to the chi-square distribution

Arguments

- *ML*: The Machine Learning scan module.
- *vpoints*: Path to the file containing the points in the validation dataset.

`SMEFT19.ml.load_model` (*fmodel*, *vpoints*)

Loads a XGBoost model previously saved

Arguments

- *fmodel*: Path to the file where the model was saved.

Returns

- Machine Learning scan.

`SMEFT19.ml.regr` (*ML*, *vpoints*)

Plots the predicted likelihood vs the actual likelihood and computes their regression coefficient

Arguments

- ML: The Machine Learning scan module.
- vpoints: Path to the file containing the points in the validation dataset.

Returns

- A tuple containing the Pearson r coefficient and the p-value of the regression

`SMEFT19.ml.train(fMC, fval, fmodel)`

Trains the Machine Learning algorithm with the previously computed Metropolis points

Arguments

- fMC: Path to the file containing the Montecarlo pre-computed points.
- fval: Path to the file where the validation points will be saved.
- fmodel: Path to the file where the XGBoost model will be saved.

Returns

- The Machine Learning scan module, already trained and ready to be used

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

S

SMEFT19.comparepulls, [5](#)
SMEFT19.ellipse, [2](#)
SMEFT19.ml, [15](#)
SMEFT19.obsuncert, [4](#)
SMEFT19.plots, [8](#)
SMEFT19.scenarios, [11](#)
SMEFT19.SMEFTglob, [1](#)

C

`calculate()` (in module *SMEFT19.obsuncert*), 5
`compare()` (in module *SMEFT19.comparepulls*), 7
`compare_plot()` (in module *SMEFT19.plots*), 9

E

`error_plot()` (in module *SMEFT19.plots*), 9
`evolution_plot()` (in module *SMEFT19.plots*), 9

H

`hatch_contour()` (in module *SMEFT19.plots*), 10
`hist()` (in module *SMEFT19.ml*), 17

I

`idemp()` (in module *SMEFT19.scenarios*), 13

L

`likelihood_fits()` (in module *SMEFT19.SMEFTglob*), 1
`likelihood_global()` (in module *SMEFT19.SMEFTglob*), 1
`likelihood_plot()` (in module *SMEFT19.plots*), 10
`load()` (in module *SMEFT19.ellipse*), 3
`load_model()` (in module *SMEFT19.ml*), 17
`loadobslist()` (in module *SMEFT19.SMEFTglob*), 1

M

`massrotation()` (in module *SMEFT19.scenarios*), 13
`minimum()` (in module *SMEFT19.ellipse*), 3

N

`notablepoints()` (in module *SMEFT19.ellipse*), 3
`notablepulls()` (in module *SMEFT19.comparepulls*), 7

P

`parametrize()` (in module *SMEFT19.ellipse*), 4
`pointpull()` (in module *SMEFT19.comparepulls*), 7
`prediction()` (in module *SMEFT19.SMEFTglob*), 1
`pull_obs()` (in module *SMEFT19.SMEFTglob*), 2

`pullevolution()` (in module *SMEFT19.comparepulls*), 7

R

`regr()` (in module *SMEFT19.ml*), 17
`rot2lqU1()` (in module *SMEFT19.scenarios*), 13
`rotBI()` (in module *SMEFT19.scenarios*), 13
`rotBII()` (in module *SMEFT19.scenarios*), 13

S

`save()` (in module *SMEFT19.ellipse*), 4
`scI()` (in module *SMEFT19.scenarios*), 14
`scII()` (in module *SMEFT19.scenarios*), 14
`scIII()` (in module *SMEFT19.scenarios*), 14
`scIV()` (in module *SMEFT19.scenarios*), 14
`scIX()` (in module *SMEFT19.scenarios*), 14
`scV()` (in module *SMEFT19.scenarios*), 14
`scVI()` (in module *SMEFT19.scenarios*), 15
`scVII()` (in module *SMEFT19.scenarios*), 15
`scVIII()` (in module *SMEFT19.scenarios*), 15
`SHAP_bf()` (in module *SMEFT19.ml*), 17
`SHAP_param()` (in module *SMEFT19.ml*), 17
`SHAP_summary()` (in module *SMEFT19.ml*), 17
SMEFT19.comparepulls (module), 5
SMEFT19.ellipse (module), 2
SMEFT19.ml (module), 15
SMEFT19.obsuncert (module), 4
SMEFT19.plots (module), 8
SMEFT19.scenarios (module), 11
SMEFT19.SMEFTglob (module), 1

T

`train()` (in module *SMEFT19.ml*), 18