1 Installation

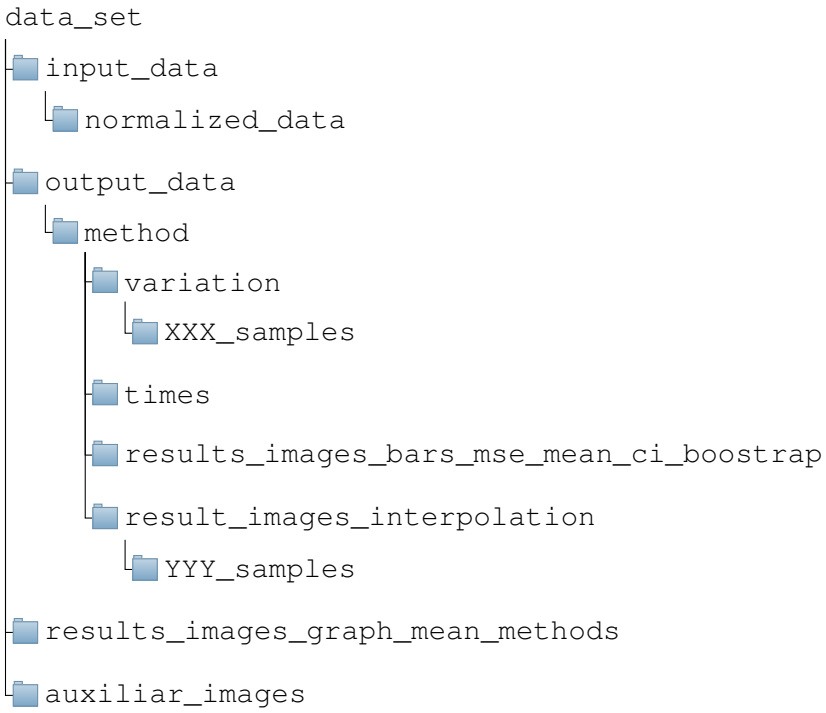
Here you can find the Python programs and the data necessary to obtain all the results of the article **Evaluation of interpolation methods for generating maps in Cultural Heritage chemical applications**.

The installation process is very simple: download the files and run the different programs.

There are two folders, man and transfiguration that correspond to the two datasets, **The Man** and **The Transfiguration**, and a program do_all.py that allows to create all the data, graphs and maps files. Another possibility is to execute the programs individually.

2 Folder hierarchy

For each dataset, there is this folder hierarchy:

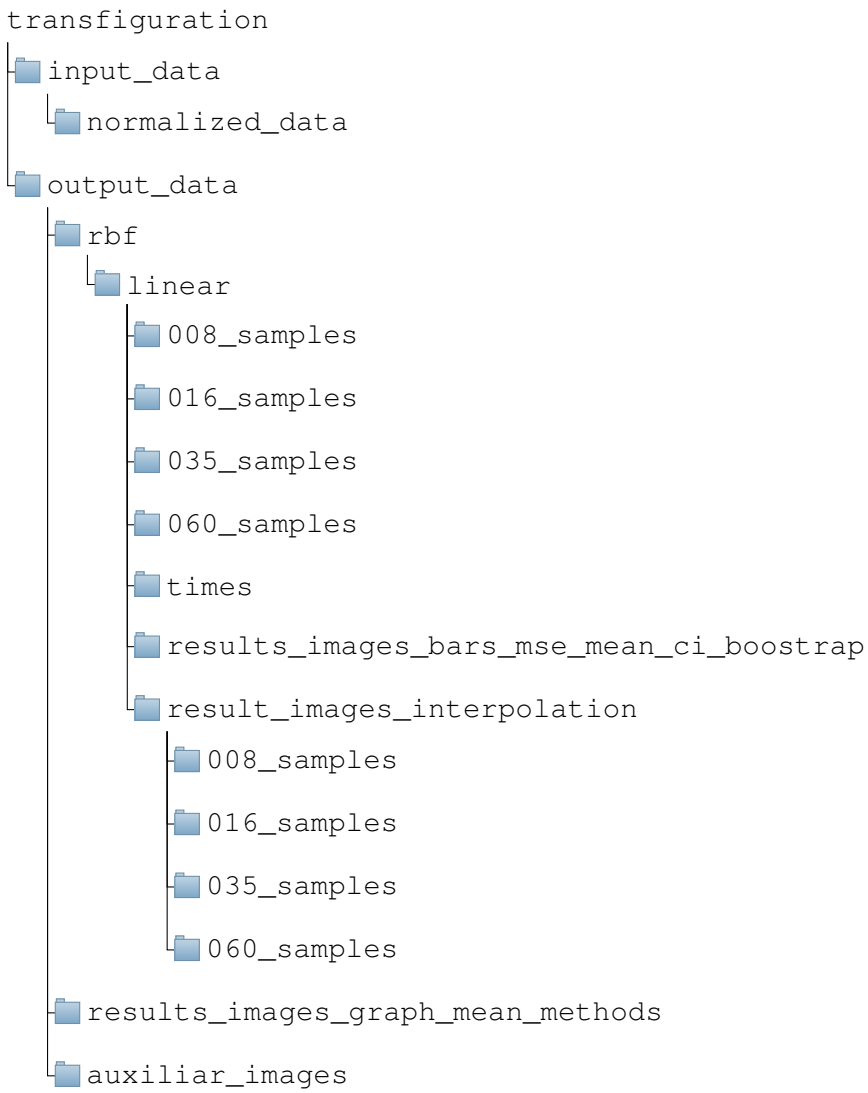


The content of each folder is as follows:

- input_data: contains the input data.
  - normalized_data: the data of the elements
- ouput_data: contains the output files created by the programs.
  - method (e.g. rbf): contains the folders for each method.
    - variation (e.g. linear): contains the folders for each variation of a method.

        - XXX_samples (e.g. 082_samples): a set of folders called XXX_samples, where XXX is the number of used points. Each one contains the MSE data for XXX num of points.
        - times: contains information about the time used to do the computations
        - results_images_bars_mse_mean_ci_boostrap: contains the bar graphs for each element
        - result_images_interpolation: a set of folders called YYY_samples (e.g. 165_samples),where YYY is the number of used points. Each one contains the interpolation images for XXX num of points.
    - results_images_graph_mean_methods: contains the images with the comparison of several methods for each element.
    - auxiliar_images: contains auxiliar images (e.g. the measurement positions).

For **The Transfiguration** the hierarchy could be as follows (only 4 XXX_samples folders):



3 Programs

Entering in each dataset folder we can find the following programs:

- Programs to create the MSE data
  - barnes_create_mse.py: program to create the MSE data for the Barnes method.
  - cressman_create_mse.py: program to create the MSE data for the Cressman method.
  - kriging_create_mse.py: program to create the MSE data for the kriging method in its different possibilities.
  - mhd_create_mse.py: program to create the MSE data for the MHD method.
  - radial_basis_function_create_mse.py: program to create the MSE data for the RBF method in its different possibilities.
- Program to create the bars graphs
  create_bars_graphs_mse_mean_ci_bootstrap_methods.py: this program generates the bar charts showing the MSE with the CI that are calculated with the bootstrap method. Each graph shows only one method and one element.
- Program to create the line graphs
  create_graphs_mse_methods.py: this program generates the graphs showing the MSE of different methods for one element.
- Programs to create the maps
  - barnes_create_interpolation_image.py: program to create the interpolation images for the Barnes method.
  - cressman_create_interpolation_image.py: program to create the nterpolation images for the Cressman method.
  - kriging_create_interpolation_image.py: program to create the nterpolation images for the kriging method in its different possibilities.
  - mhd_create_interpolation_image.py: program to create the nterpolation images for the MHD method.
  - radial_basis_function_create_interpolation_image.py: program to create the nterpolation images for the RBF method in its different possibilities.
- Program to create the auxiliary images
  draw_images.py: program to create auxiliar images:
  - The positions where the measurements are placed.
  - The set of positions for creating the interpolation function.
  - The set of positions for computing the MSE.

For the execution of a program it is enough to put its name without parameters. For example, python3 barnes_create_mse.py or python3 create_graphs_mse_methods.py.

It is important that the execution of the programs to follow an order for obtaining correct results:

1. The MSE data is created using the programs that identify the different methods: barnes _create_mse.py, cressman_create_mse.py, kriging_create_mse.py, mhd_create_mse.py and radial _basis _function_create_mse.py.

2. The error bar graphs are created using create_bars_graphs_mse_mean_ci_bootstrap_methods.py.
3. The line graphs showing the comparison for different elements are created using create_graphs _mse_methods.py

Once all the statistic information is created, the maps can be produced with the corresponding programs: barnes_create_interpolation_image.py, cressman_create_interpolation_image.py, kriging _create_interpolation_image.py, mhd_create_interpolation_image.py and rbf_create _interpolation _image.py.

3.1 Running all

do_all.py is a program that facilitates the task of obtaining all data files and images, automatically running all programs in the correct order. It has two parameters that control the results: data_set and mode.

data_set controls what dataset to use: **The Man**, using man, or **The Transfiguration**, using transfiguration. mode controls if we want to produce the statistical data, using stats, or we want to produce the maps, using maps.

For example, if we would want to obtain the statistical data of **The Man**, the following command would be used: pyhon3 do_all.py man stats, if we would want to obtain the maps of **The Transfiguration**, the following command would be used: pyhon3 do_all.py transfiguration maps. do_all.py will execute all the programs with their current values.

3.2 Configuration

- Programs to create the MSE data
  The programs to obtain the MSE values can be configured by changing certain parameters. The most important are the elements and the percentages of points for which the calculations are performed. By default the computations are performed for 5 elements but it can be changed by commenting the current option and uncommenting the option with all the elements, or you can create a custom list. The Save variable allows you to indicate to the program whether or not to save the graphs. The value False indicates that you do not want to save them.

```
# For saving the results as images
Save=False

# elements
#Elements=["As","Ba","Ca","Cd","Co","Cr","Cu","Fe","Hg","K","Mn","Ni","Pb","Sb","←
    Se","Sn","Ti","Zn"]
Elements=["Cu","Fe","Hg","Pb","Ti"]
```

  For the points used, just modify the proposed list of percentages.

```
# percentages
Vec_percentages=[5,10,20,30,40,50,60,70,80,90]
```

- Program to create the bars graphs
  The program create_bars_graphs_mse_mean_ci_bootstrap_methods.py can be easily modified to obtain more or less results. To do this, the Version variable must be modified: if it has the value FULL it will calculate all the variations of the methods for the list of elements. If the

value is SMALL only the most important variant will be calculated. By default it has the value SMALL. It is important to note that the list of elements is reduced even for the FULL case. In case you want to make the calculation with all the elements, you must comment the current list and uncomment the list with all the elements. You can also modify the type of output with the variable File_type: You can obtain PDF files if it has the value .pdf or PNG files by setting .png.

```
# Version: FULL, SMALL
Version='SMALL'

# For saving the results as images
Save=True

File_type='.pdf'
```

- Program to create the line graphs
  The program create_graphs_mse_methods.py can be easily modified to obtain more or less results in the same way as discussed in the previous case.

- Programs to create the maps
  The programs to obtain the interpolation images can be configured by changing certain parameters.

  The Elements list contains the names of the elements that will be computed. The current list has only one element, Pb. It possible to comment it and uncomment the list with all the elements, or you can create a custom list.

```
# elements to be computed
# Elements=["As","Ba","Ca","Cd","Co","Cr","Cu","Fe","Hg","K","Mn","Ni","Pb","Sb","↩
    Se","Sn","Ti","Zn"]
Elements=["Pb"]
```

  For some methods, kriging and RBF, there are variations. They can be changed using their corresponding lists or commenting the current one and uncommenting the list with all the variations.

```
# type of functions
# RBF_functions=["multiquadric","inverse","gaussian","linear","cubic","quintic","↩
    thin_plate"]
RBF_functions=["linear"]
```

  The color map can be changed to be one of predefined in Python, or to use the new one we have created for color blind people. The current selection is the color blind one.

```
# selected color map
# Map='rainbow'
Map=Blindcmp
# Map='summer'
```

  The map depends on the number of points that are used to create the interpolation function. It is possible to control the number of points by means of a list of percentages. The current list has only one value: the total number of points that corresponds to 100%. By commenting the current list and uncommenting the previous line, it is possible to define the list with the corresponding number of points.

```
# percentages
Vec_percentages=[5,10,20,30,40,50,60,70,80,90]
```

```
# Number of test points
# Vec_num_points=[int(round(float(x)*float(Num_points)/100.0)) for x in ↩
    Vec_percentages]
Vec_num_points=[Num_points]
```

- Program to create the auxiliary images
  The program draw_images.py permits to create auxiliary images. It can be changed in a similar way as discussed in the previous case. The new control variables are Input_image and Draw_mode.

  Input_image controls what image is used as the background. These files must be in input_data folder. The possible cases are:

  - ORIGINAL: To use the original image. The file must be called vis_visible.png, or to change the code.
  - ORIGINAL_UNSATURATED: To use an unsaturated version of the orginal image. The unsaturated version can be created from the original image with a program like Photoshop or Gimp. The file must be called vis_visible_small_desaturated.png, or to change the code.
  - MAP: To use a map. It can be any of the interpolation images. The file must be called interpolation.png, or to change the code.

  Input_image controls the type of output. The possible cases are:

  - ORIGINAL: A copy of original image is produced. Only changes the size
  - POSITIONS: The measurement positions are drawn over the background image as red marks.
  - SELECTED_POSITIONS: The selected positions are drawn over the background image as black marks.
  - UNSELECTED_POSITIONS_WITH_ERROR: The unselected positions are drawn over the background image as square marks with the color that corresponds to the MSE. A scale with the color mapping is added.

```
# Input image: 'ORIGINAL','ORIGINAL_UNSATURATED','MAP'
Input_image='MAP'

# draw mode: "ORIGINAL","POSITIONS","SELECTED_POSITIONS","↩
    UNSELECTED_POSITIONS_WITH_ERROR"
Draw_mode="UNSELECTED_POSITIONS_WITH_ERROR"
```