plotXVG

User & Reference Manual

Version 1.1b2

Måns Rosenbaum and David van der Spoel

January 23, 2026

Contents

List of Figures

Preface

This manual is a work in progress, describing the current functions and utilities of plotXVG version 1.1b2 and how to install and use this tool.

Command lines will be highlighted `like this` but can also be presented

```
in this manner.
```

Enhancements, reports of issues and overall suggestions for improvements are very welcome, either by e-mail to mans.rosenbaum@icm.uu.se or on the GitHub at https://github.com/AlexandriaChemistry/plotXVG/issues.

We kindly request that you cite the paper about plotXVG if you use the tool for a publication of your own.

plotXVG can be downloaded as a free and open source software from https://github.com/AlexandriaChemistry/plotXVG.

About plotXVG

plotXVG is a command line tool based on the Matplotlib library as well as dependencies from the NumPy package meant to in part replace the Grace software to alleviate quick generations of plots from GROMACS generated xvg files, but supports csv files as well. It currently consists of 740 lines of Python code with an addition of testing of a test script.

Below follows the User Manual, an installation guide and practical usage of plotXVG including descriptions of all flags.

We hope this tool suits your needs!

User Manual

1 Installation of plotXVG

plotXVG is written in python and requires only a few dependencies to run.

1.1 Prerequisites

The following software packages are required for plotXVG to work:

1. Matplotlib version 3.7.0 or higher.
2. NumPy version 1.24.1 or higher.
3. Python, version 3.7 or higher.
4. Pip package manager, preferably of newest release.
5. (Optional) XQuartz or similar for opening external windows while being on a remote server.

1.2 Running the Installation

Installation of plotXVG can be done in two ways, The first one is by cloning the git repository to your local computer by running

```
git clone https://github.com/AlexandriaChemistry/plotXVG
```

and then go in to that directory and, while being in your local environment, run the installation with pip:

```
python -m pip install .
```

Now you can use plotXVG! Another option is to simply type

```
python -m pip install plotxvg
```

and plotXVG will be fetched from PyPi and installed, simple as that. The benefit with cloning the repository from GitHub is the example runs are available to test the script. It is not a requirement.

2 Using plotXVG

plotXVG can be called both from the command line and as an integrated API for in-script analysis. CLI operated plotXVG can be useful for quick plotting of new output data files and opens an external window operated by matplotlib which lets the user interact with the plot. API operated plotXVG is suitable for analysis in, for example, a Jupyter notebook and can alleviate plotting of multiple files along with simpler programmatic control over styles, labels and more. Both are easily callable but with some small differences.

2.1 CLI

The general command for using plotXVG in the command line is:

```
plotxvg -f file(s) [-flags]
```

As apparent, there is only one required argument, namely an input file in .xvg or .csv format. This will produce a scatter plot from your input data. An external window will automatically be opened by matplotlib showing you an interactive plot from which you can save it by the supported buttons in the interactive window. Alternatively, plotXVG has a `-save` flag which saves the plot after the layout of the plot parts has been regulated. To clarify, the interactive matplotlib window might not show the actual finalized image. If no external window is wished to be opened, `-noshow` can be added to the command.

```
plotxvg -f file(s) -save output.pdf -noshow
```

In order to create for example a line plot, one additional argument `-ls` (short for linestyle) needs to be added along with a line style in order to tell plotXVG that lines rather than markers are desired.

```
plotxvg -f file(s) -ls solid
```

will create a line plot with a solid line. Supported line style are solid, dashed, dotdashed and dotted lines. The user can also by all means combine lines with markers by adding both flags:

```
plotxvg -f file(s) -ls dashed -mk +
```

which results in a dashed line with crosses on each data point.

2.2 API

To use plotXVG in a python script or a notebook you need first to import plotXVG as a library:

```
import plotxvg
```

Then simply call on the `plot()` function:

```
plotxvg.plot("file")
```

To add multiple files, put them in a list:

```
plotxvg.plot(["file1", "file2])
```

Adding flags is done by naming the flag and assert the chosen values as strings in a list:

```
plotxvg.plot(['file1.xvg','file2.xvg'],
             linestyle=['solid','dashed'],
             panels=True,
             save='plot.pdf')
```

Note that boolean flags do not need to be strings.

2.3 Creating Heatmaps

Creating heatmaps in order to study for example free energy landscapes is also supported by plotXVG. To create a heatmap from your data, use the heatmap mode by adding the `-heatmap` flag:

```
    plotxvg -f file -heatmap
```

In the API, simply set the `heatmap` parameter to `True`:

```
plotxvg.plot("file.xvg", heatmap=True)
```

The input file should contain three columns: x-coordinates, y-coordinates, and values to be mapped to the color scale. This is particularly useful for visualizing free energy landscapes and other 2D scalar field data. You can also create plots with contour lines instead using the flag `-contour`. Additional flags to expand the use of heatmap plotting are what colormap to prefer with `cmap` (default is *viridis*) and the number of levels in the contour parsed with the flag `-levels`. Default number of levels is 15. See example plots in Figures 20 and 21.

3 Additional Flags

Below all flags are given and a small description of each flag. This description can also be called by typing `plotxvg -h` in the command line.

```
-h, --help
                show this help message and exit

-f [FILENAME ...], --filename [FILENAME ...]
                Filename(s) to read and plot

-follow, --follow     Continuously update the plot by
                re-reading the input file(s)

-debug, --debug
                Turn on printing of debugging messages

-font FONTNAME, --fontname FONTNAME
                Font for all text.

-alfs AXISLABELFONTSIZE,
--axislabelfontsize AXISLABELFONTSIZE
                Axis label font size, default 26

-tfs TITLEFONTSIZE, --titlefontsize TITLEFONTSIZE
                Title font size, set to zero for no title,
                default 30

-lfs LEGENDFONTSIZE, --legendfontsize LEGENDFONTSIZE
               Legend font size, set to zero for no legend,
                default 26

-tickfs TICKFONTSIZE, --tickfontsize TICKFONTSIZE
                Tick font size, default 24

-ls LINESTYLE [LINESTYLE ...],
--linestyle LINESTYLE [LINESTYLE ...]
                What kind of line style:
```

```
                solid, dashed, dashdot, dotted

-mk MARKER [MARKER ...], --marker MARKER [MARKER ...]
                Use markers for data sets: o, +, x, <, >...

-mksize MARKERSIZE, --markersize MARKERSIZE
             Size of filled markers for data sets, default 10

-mkwidth MARKEREDGEWIDTH, --markeredgewidth MARKEREDGEWIDTH
            Size of character markers (e.g. +) for data sets,
                default 2

-colors [COLORS ...], --colors [COLORS ...]
             Colors for the plots. Colors defined by the user
            will be applied to the datasets in order. If there
            are more datasets than color inputs, default colors
                will be used.

-save SAVE, --save SAVE
               Save plot. Please specify saving location and
                 preferred filetype (.pdf, .png...)

-sqfig, --squarefig
                Make the figure square

-eqax, --equalaxes
               Make the plot square with equally large axes

-bar, --bar
                Make a bar graph

-noshow, --noshow
                Do not show the figure

-res, --residual
               Subtract x from y for all data sets - useful
                for correlation plots
```

```
-fl, --filelabel
                Add the filename to the labels in the plot
                (may yield long labels)

-logy, --logy
                Use a log scale on the Y-axis

-xmin XMIN, --xmin XMIN
           Minimum value of X-axis. Default = defined by data.

-xmax XMAX, --xmax XMAX
           Maximum value of X-axis. Default = defined by data.

-ymin YMIN, --ymin YMIN
           Minimum value of Y-axis. Default = defined by data.

-ymax YMAX, --ymax YMAX
           Maximum value of Y-axis. Default = defined by data.

-xframe XFRAME, --xframe XFRAME
                Width of the plot 100 pixels, default 16

-yframe YFRAME, --yframe YFRAME
                Height of the plot 100 pixels, default 9

-panels [top,side], --panels [top,side]
           Generate different panels to plot in, one file per
                panel. Add 'side' for side-by side panels.

-sfx SUBFIGUREX, --subfigureX SUBFIGUREX
            X position of subfigure label when using panels.
                Default -0.15

-sfy SUBFIGUREY, --subfigureY SUBFIGUREY
            Y position of subfigure label when using panels.
                Default 1.0

-ign [IGNORE ...], --ignore [IGNORE ...]
```

```
            legends of the series to ignore. Please specify
               the whole legend label.

-title TITLE [TITLE ...], --title TITLE [TITLE ...]
              User-defined title(s). This flag overwrites
             pre-defined titles. If the user wants to use
            user-defined titles and pre-defined titles on
           different panels, use None as placeholders for
            predefnied title panels. If the user wants to
          remove a specific panel title, put an empty string ''.

-notitles, --notitles
             Remove all titles (Both user-defined and title
                   pre-defined by data.)

-dslegends [DATASETLEGENDS ...],
--datasetlegends [DATASETLEGENDS ...]
           Set user-defined legends. If legends are already
            defined in the input file they are combined for
               each dataset
          i.e. '<user-defined legend> <pre-defined legend>'.

-sharelabel, --sharelabel
            Show only the x-labels on the last row of plots
             and the y-labels on the first column of plots
               (useful if all subplots share the same
               x- and y-labels)

-legend_x LEGEND_X, --legend_x LEGEND_X
           Put the legend box horizontally on this position,
               default 0.02

-legend_y LEGEND_Y, --legend_y LEGEND_Y
            Put the legend box vertically on this position,
               default 0.98

-stats, --stats
               Print RMSD and R2 values of datasets
```

```
                (x-axis is reference data and y-axis holds
                the predicted values)

-csvx CSVX, --csvx CSVX
          Choose what x column you would like from a csv file.
               Choose only one column. Default is column 1

-csvy CSVY [CSVY ...], --csvy CSVY [CSVY ...]
          Choose what y column(s) you would like from a csv file.
               Default is column 3

-v, --version        show program's version number and exit

-heatmap, --heatmap   2D heatmap plot

-contour, --contour   Contour 2D plot

-cmap CMAP, --cmap CMAP
                        Matplotlib colormap

-levels LEVELS, --levels LEVELS
                        Number of contour levels
```

4 Example Plots

This section aims to give an easier understanding of how one can use plotXVG, using examples given by the test.py script found in the plotXVG GitHub repository. For each plot is also the exact plotXVG command for easy reproducibility. In order to run these successfully, you need to be in the tests directory. We hope this can further show the scope of plotXVG's abilities.

Command for Fig. 1:

```
plotxvg -f gmx_files/rmsd_calpha.xvg -save \
  plotxvg_tests-output/00default.pdf -noshow
```

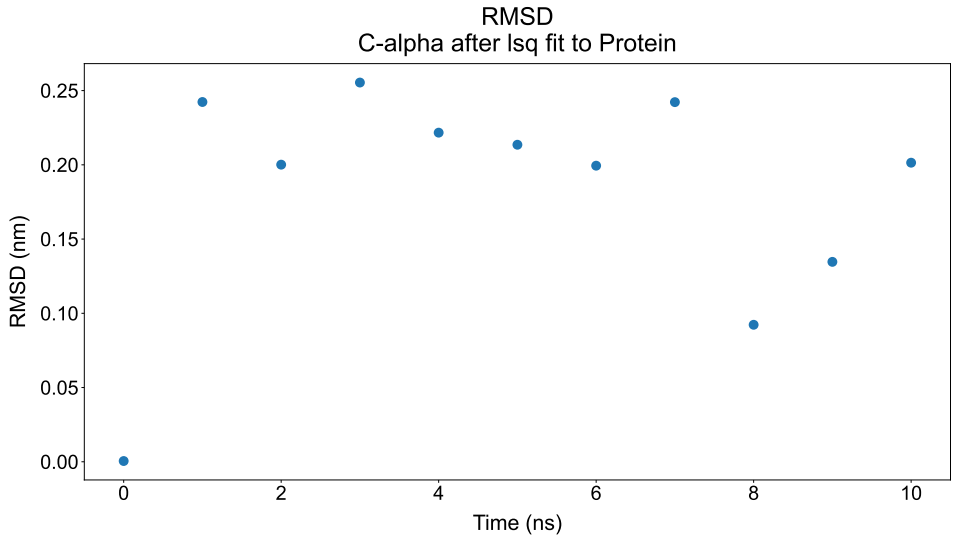

Figure 1: This plot is created without flags. Default setting is to make a scatterplot.

Command for Fig. 2:

```
plotxvg -f gmx_files/potential_energy.xvg -ls solid -save \
  plotxvg_tests-output/01only_lines.pdf -noshow
```

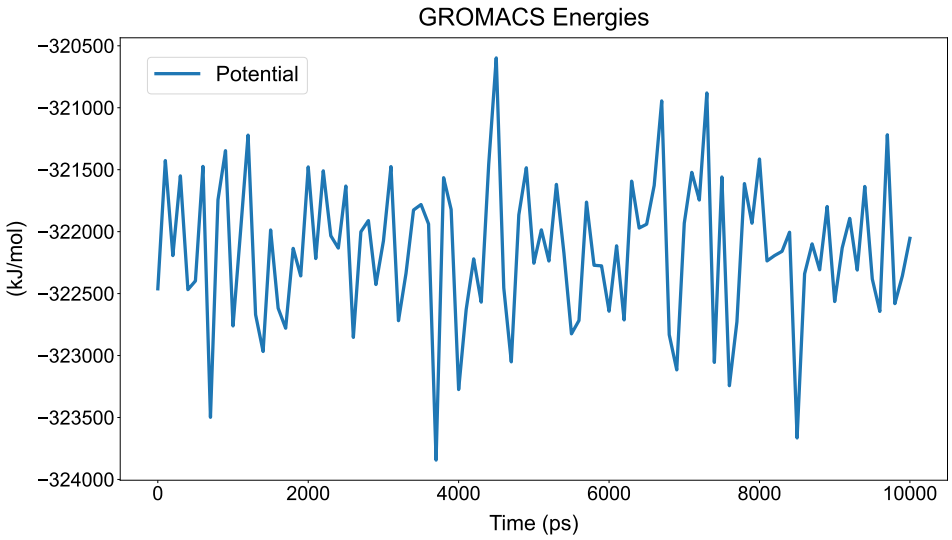

Figure 2: Using lines

Command for Fig. 3:

```
plotxvg -f other_files/ammonium#chloride.xvg  -save \
  plotxvg_tests-output/02markers_5datasets.pdf -noshow
```

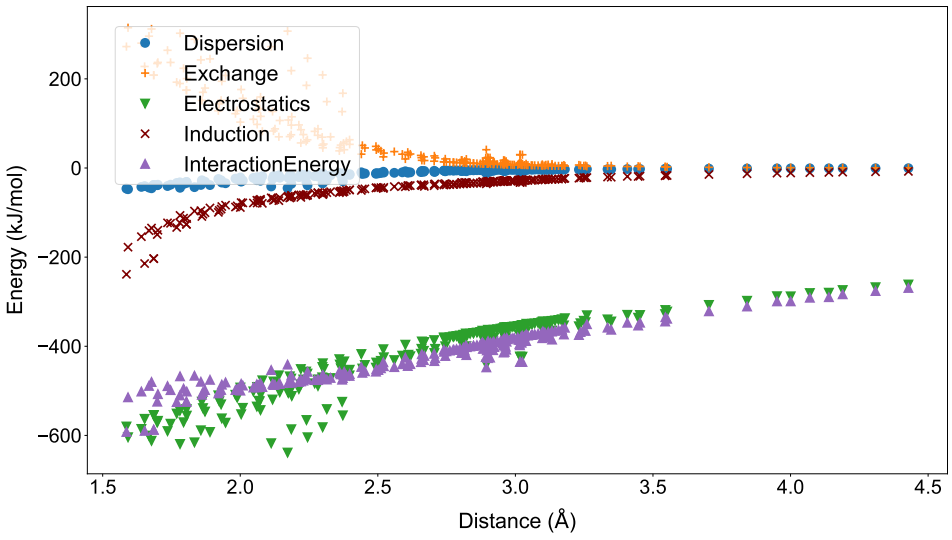

Figure 3: One file containing five datasets, without any flags added (will thus be plotted using markers).

Command for Fig. 4:

```
plotxvg -f gmx_files/gyrate.xvg -ls dotted solid dashed \
  dashdot -save plotxvg_tests-output/03lines_4datasets.pdf \
   -noshow
```

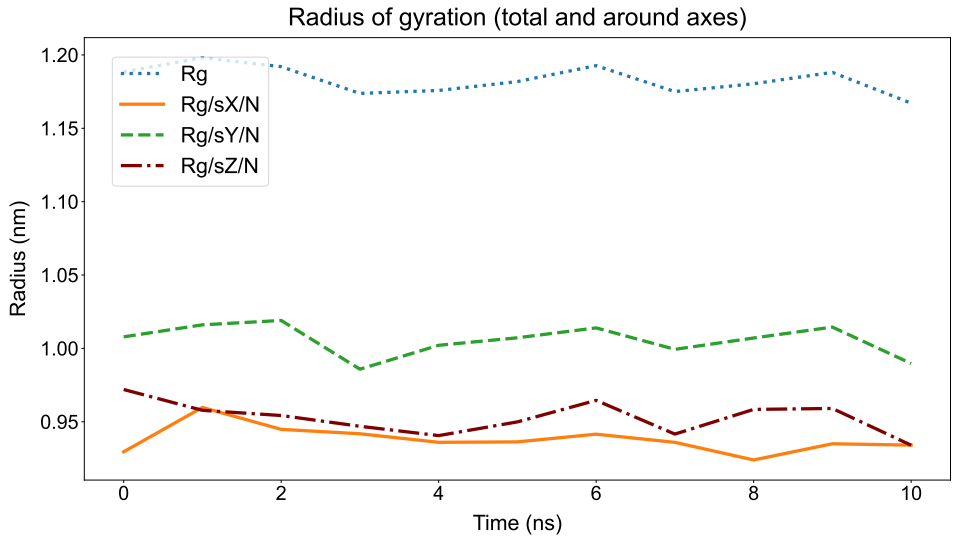

Figure 4: User-defined lines

Command for Fig. 5:

```
plotxvg -f other_files/ammonium#chloride.xvg -ls solid \
  dashed solid None None -mk None None x + . -save \
  plotxvg_tests-output/04mk_and_ls.pdf -noshow
```

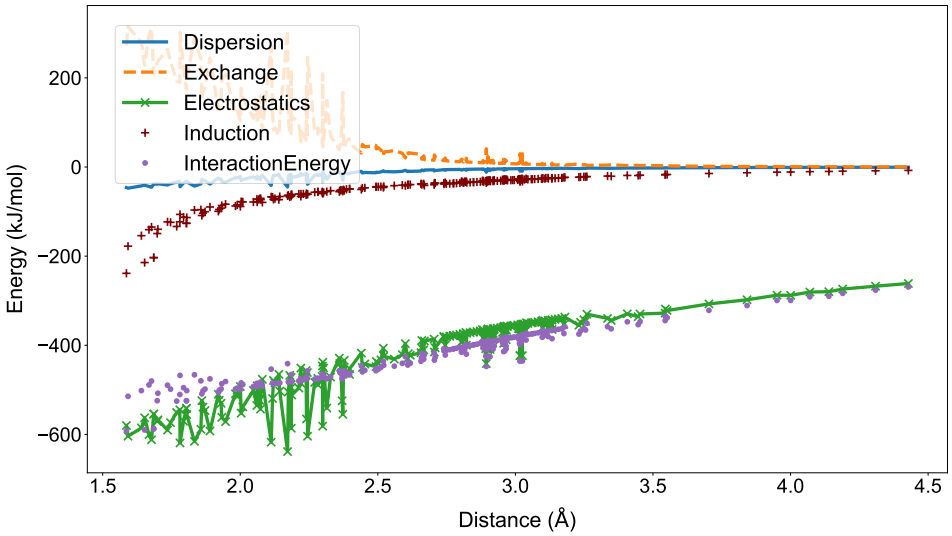

Figure 5: Both markers and linetyles combined in the same plot. Note how markers and lines can be used separately and combined

Command for Fig. 6:

```
plotxvg -f other_files/ammonium#chloride.xvg -legend_x 0.68 \
  -save plotxvg_tests-output/05move_legendbox.pdf -noshow
```

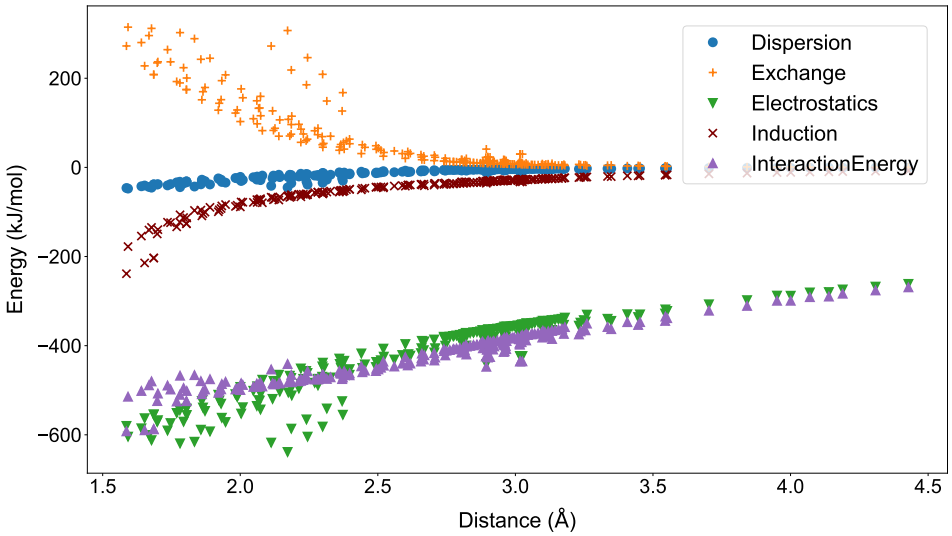

Figure 6: Moving the legendbox to the right. Moving the box up and down can be done similarly with -legend_y.

Command for Fig. 7:

```
plotxvg -f gmx_files/rmsd_calpha.xvg \
 gmx_files/rmsd_sidechain.xvg -panels -sharelabel -notitles \
  -ls solid  -save plotxvg_tests-output/06two_panels.pdf \
  -noshow
```

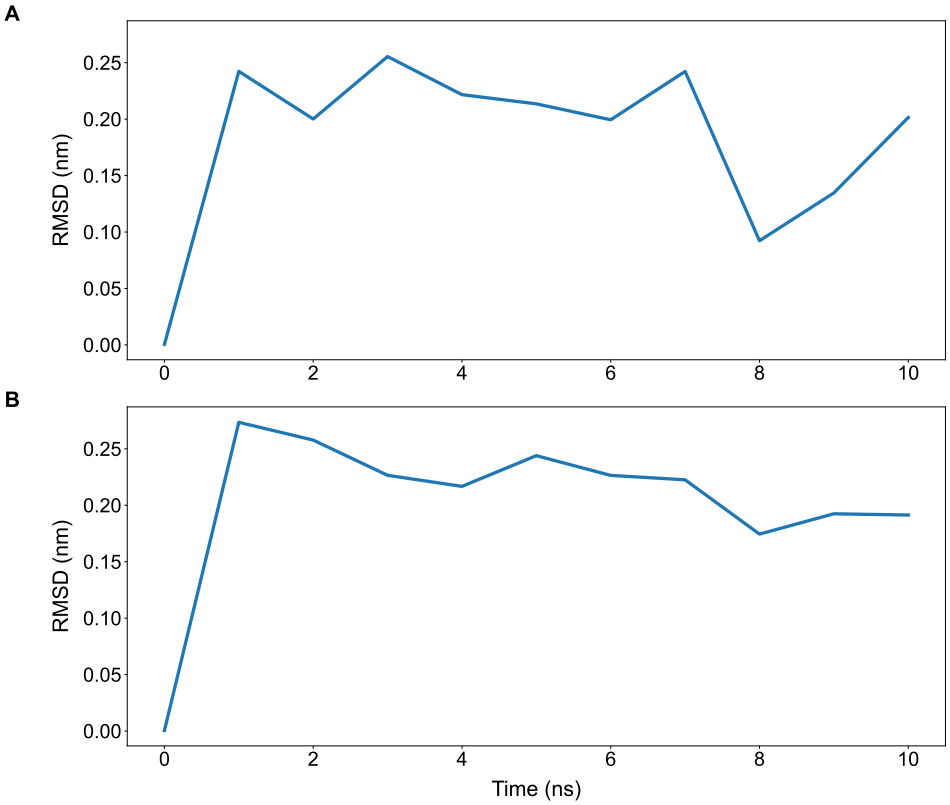

Figure 7: Using the panels flag, along with -notitles. Also note -sharelabel, which removes axis labels except for the first column and the last row. Suitable if all subplots shares the same axis labels.

Command for Fig. 8:

```
plotxvg -f gmx_files/rmsd_calpha.xvg \
  gmx_files/temp_press.xvg gmx_files/gyrate.xvg \
 gmx_files/2dproj_PC1_PC2.xvg -ls solid solid solid solid \
  dotted dashdot dashed None -mk None None None o + x ^ + \
  -panels -tfs 40 -alfs 35 -mksize 20 -mkwidth 4 -save \
  plotxvg_tests-output/07mult_panels.pdf -noshow
```

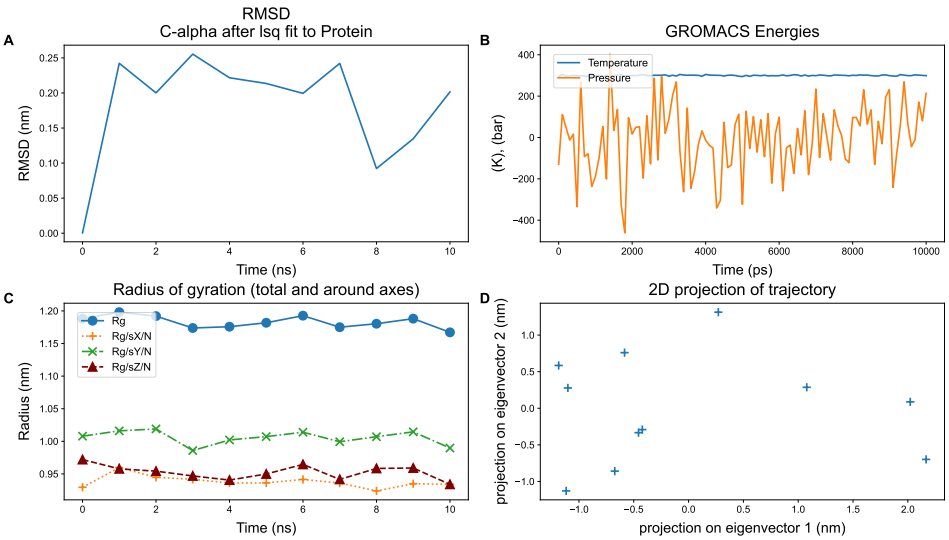

Figure 8: Panels that shows differently expressed data, such as two with lines and two with markers. Font- line- or marker-sizing are dynamic based on the number of subplot columns, but specified at will, by for example adding -mksize 20 -mkwidth 4 in a subplot of two columns.

Command for Fig. 9:

```
plotxvg -f gmx_files/intra_energies.xvg -colors green purple \
  red -save plotxvg_tests-output/08colors.pdf -noshow
```

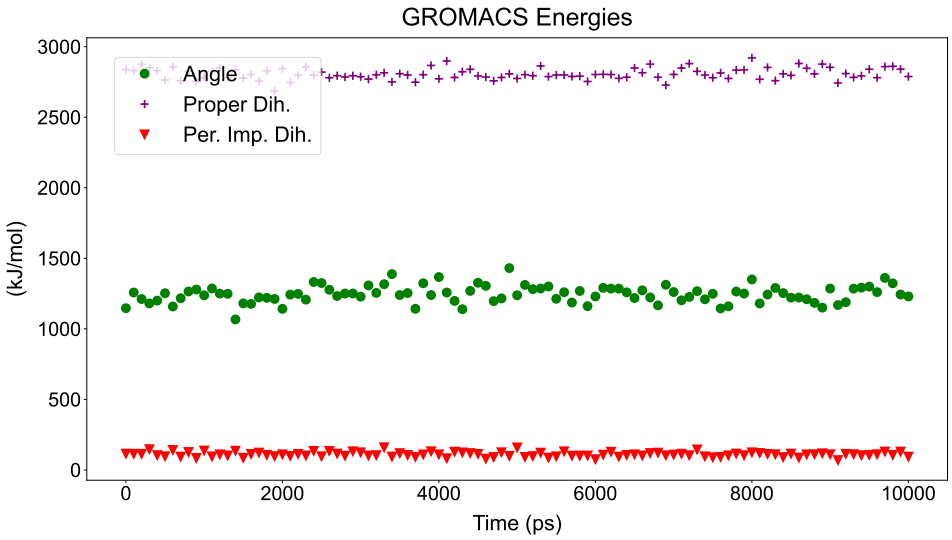

Figure 9: A custom choice of colors. Colors defined by the user will be applied to the datasets in order. If there are more datasets than color inputs, default colors will be used.

Command for Fig. 10:

```
plotxvg -f act_files/COULOMB-PC-elec.xvg \
  act_files/COULOMB-PC+GS-elec.xvg -dslegends 'PC-elec' \
  'PC+GS-elec' -eqax -sharelabel -panels side  -save \
  plotxvg_tests-output/09equalaxes.pdf -noshow
```

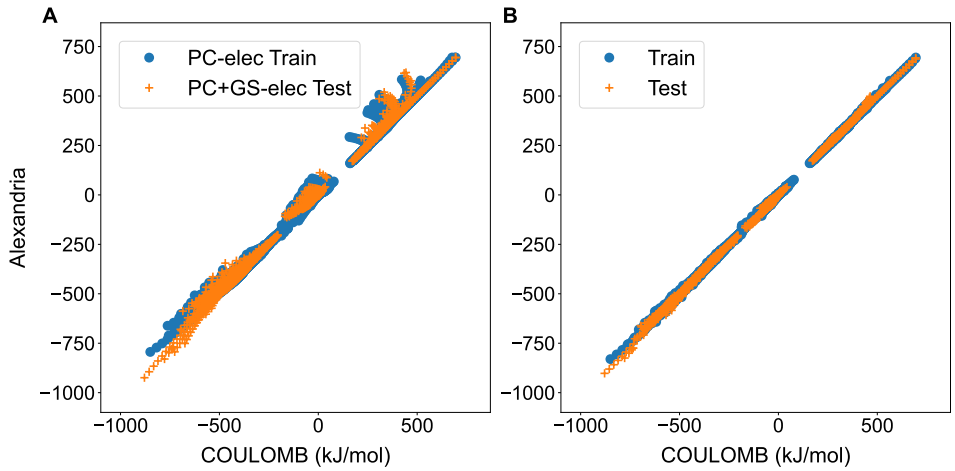

Figure 10: Demonstrates the equal axes flag. This flag makes the plot square with equally large axes, suitable for correlation plots. Also note the possibility to plot panels side-by-side by adding side to the panels flag.

Command for Fig. 11:

```
plotxvg -f gmx_files/rmsd_calpha.xvg \
  gmx_files/temp_press.xvg gmx_files/gyrate.xvg \
 gmx_files/2dproj_PC1_PC2.xvg -ls solid solid solid solid \
  dotted dashdot dashed None -mk None None None o + x ^ + \
  -panels -mksize 20 -mkwidth 4 -sqfig -save \
  plotxvg_tests-output/10squarefig.pdf -noshow
```

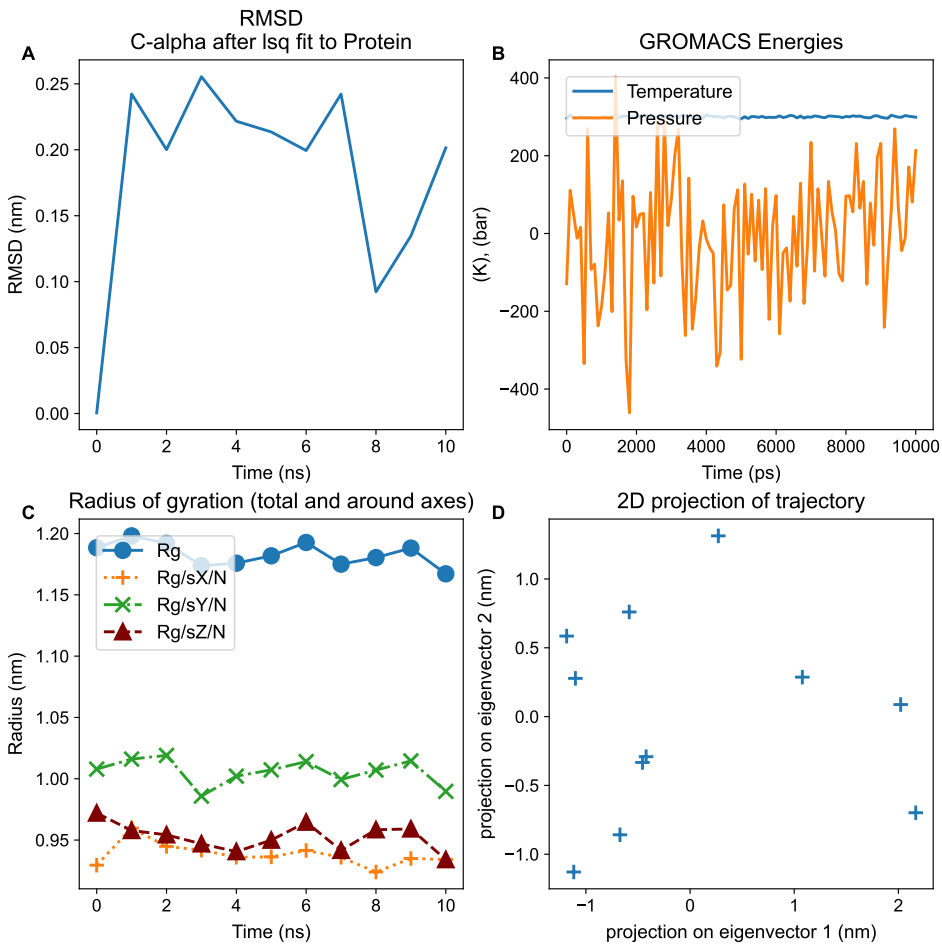

Figure 11: Demonstrates the square figure flag. This flag simply makes the saved figure square.

Command for Fig. 12:

```
plotxvg -f act_files/COULOMB-PC-elec.xvg \
  act_files/COULOMB-PC-allelec.xvg \
  act_files/COULOMB-PC+GS-elec.xvg \
  act_files/COULOMB-PC+GS-allelec.xvg -dslegends PC-elec \
 PC-allelec PC+GS-elec PC+GS-allelec -alfs 38 -lfs 19 -panels \
  -sharelabel -sqfig -stats -save \
  plotxvg_tests-output/11stats.pdf -noshow
```

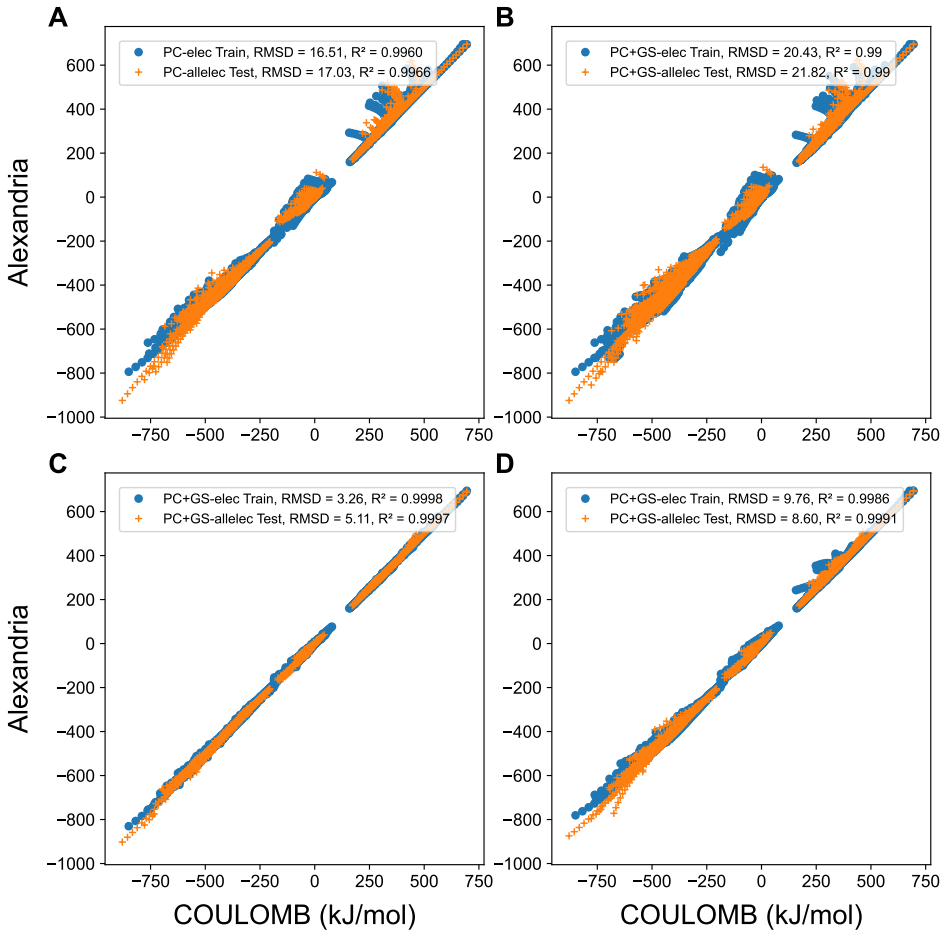

Figure 12: Shows statistics (RMSD, R²). If R² is close to 1 more digits are added. Caution for strange plotting behaviours if the legend is too long or the font is large!

Command for Fig. 13:

```
plotxvg -f act_files/COULOMB-PC-elec.xvg \
  act_files/COULOMB-PC-allelec.xvg \
  act_files/COULOMB-PC+GS-elec.xvg \
  act_files/COULOMB-PC+GS-allelec.xvg -dslegends PC-elec \
 PC-allelec PC+GS-elec PC+GS-allelec -alfs 38 -lfs 19 -panels \
  -sharelabel -sqfig -stats -res -save \
  plotxvg_tests-output/12res.pdf -noshow
```

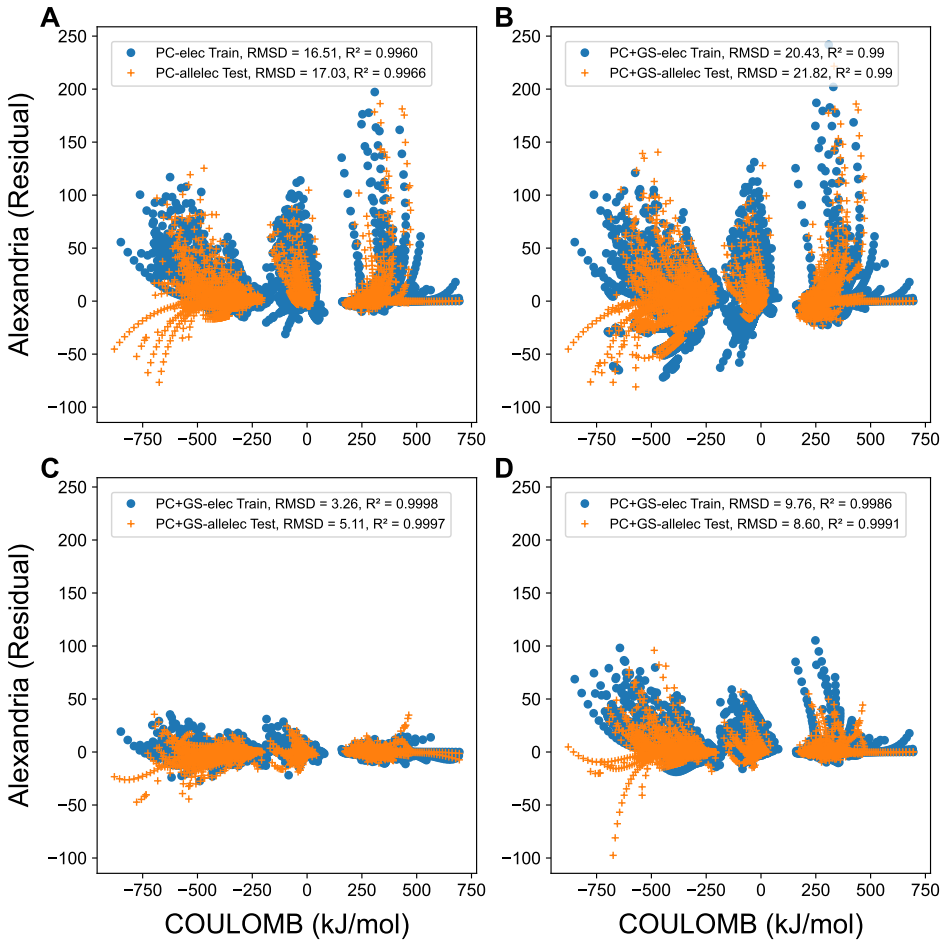

Figure 13: Plots the residual of the data, meaning x is substracted from y for all data sets. Statistics are based on original train and test values and will not be affected by the residual flag.

Command for Fig. 14:

```
plotxvg -f gmx_files/rmsf_residues.xvg -bar -save \
  plotxvg_tests-output/13bar.pdf -noshow
```

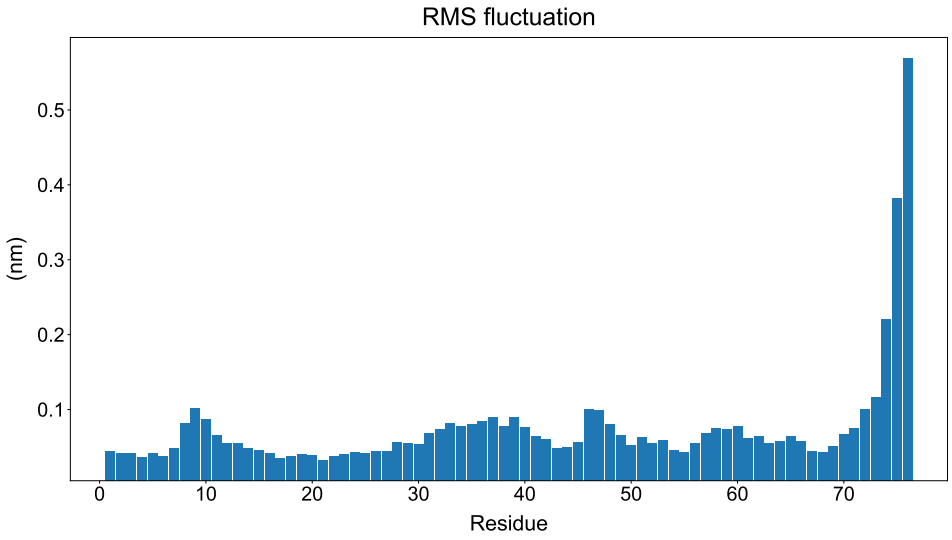

Figure 14: Histogram with one dataset

Command for Fig. 15:

```
plotxvg -f other_files/rmsf_res_66-76.xvg \
  other_files/rmsf_res_66-76x1.2.xvg \
  other_files/rmsf_res_66-76x0.8.xvg -bar -save \
  plotxvg_tests-output/14threebars.pdf -noshow
```

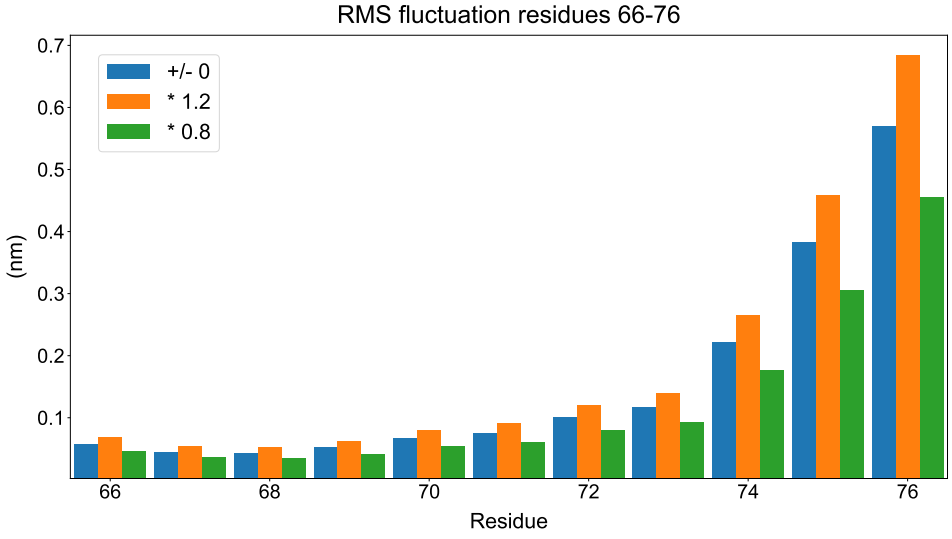

Figure 15: Histogram with three datasets

Command for Fig. 16:

```
plotxvg -f gmx_files/resarea.xvg -ls dotted -mk o -save \
  plotxvg_tests-output/15std_plot.pdf -noshow
```

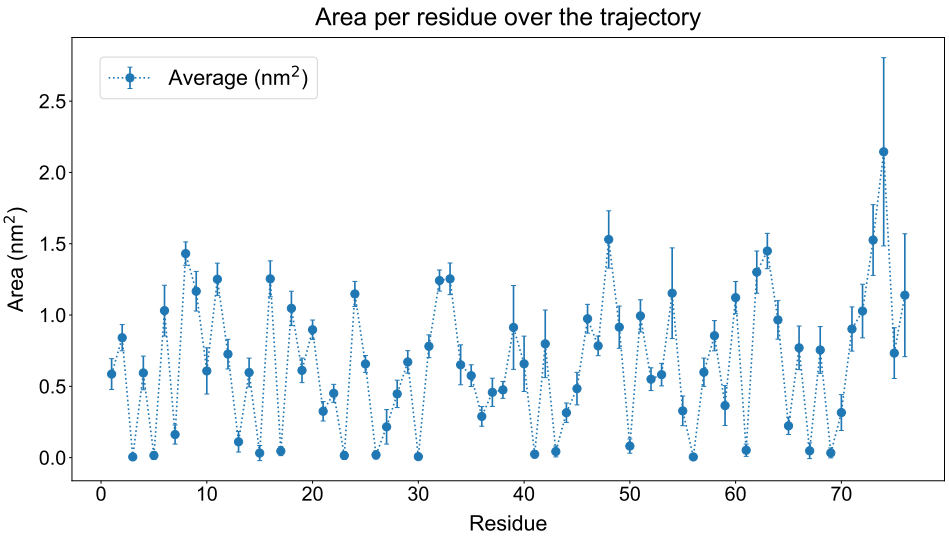

Figure 16: A xvg file containing standard deviations is automatically incorporated as errorbars.

Command for Fig. 17:

```
plotxvg -f gmx_files/2dproj_PC1_PC2.xvg -font Tahoma -save \
  plotxvg_tests-output/16font.pdf -noshow
```

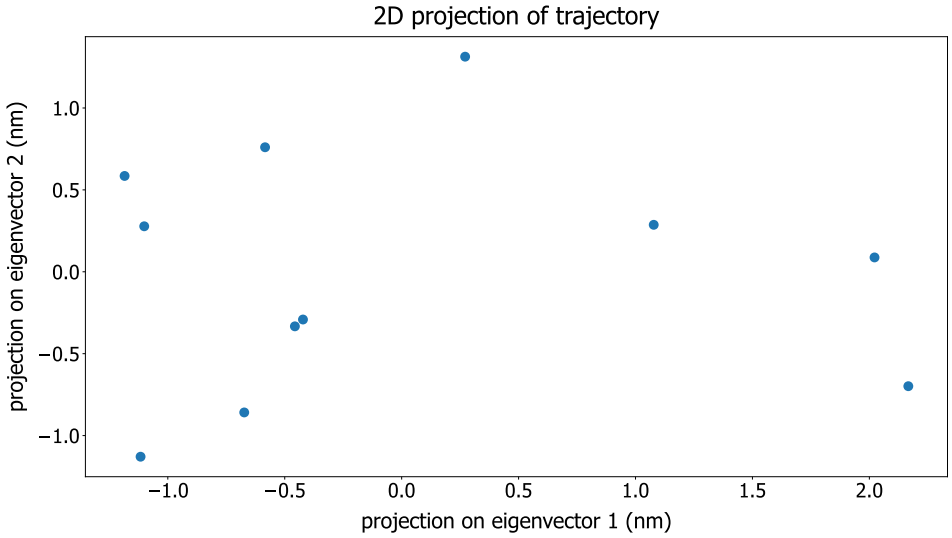

Figure 17: Change the font for all texts. Here it is changed to 'Tahoma'.

Command for Fig. 18:

```
plotxvg -f gmx_files/eigenval.xvg gmx_files/gyrate.xvg \
 gmx_files/potential_energy.xvg gmx_files/2dproj_PC1_PC2.xvg \
  gmx_files/resarea.xvg gmx_files/rmsd_backbone.xvg \
  gmx_files/rmsd_calpha.xvg gmx_files/rmsd_sidechain.xvg \
  gmx_files/rmsf_residues.xvg gmx_files/sasa_total.xvg \
 gmx_files/temp_press.xvg gmx_files/intra_energies.xvg -mk x \
 None None None None None '*' None None None None None o None \
  None None o x ^ -ls None solid dashed dashdot solid solid \
 None solid dashed solid solid solid None solid solid solid \
  solid solid solid -panels -save \
  plotxvg_tests-output/17Alot_of_panels.pdf -noshow
```

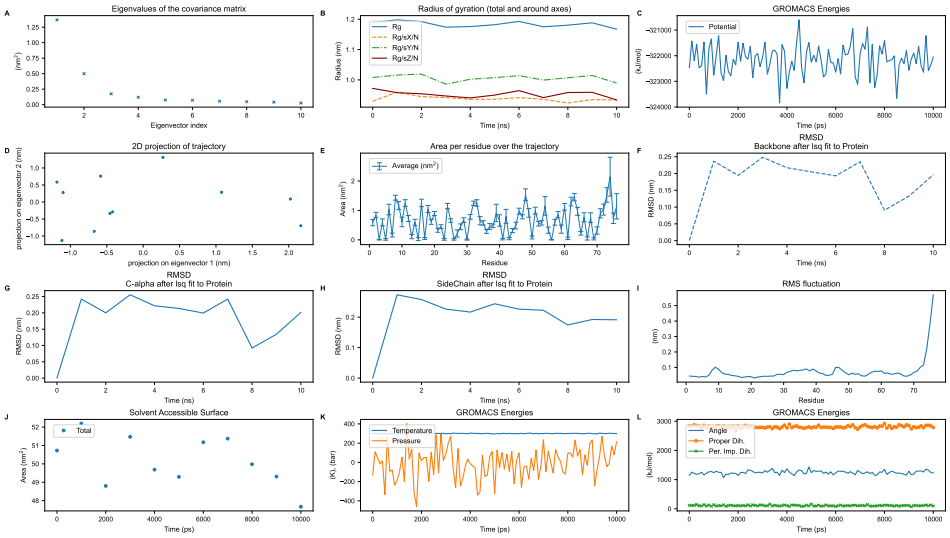

Figure 18: This demonstrates the dynamics of the program showing that even twelve files can be plotted simultaneously.

Command for Fig. 19:

```
plotxvg -f other_files/openmm.csv -csvx 2 -csvy 7 -ls solid \
  -save plotxvg_tests-output/18openMMfile.pdf -noshow
```

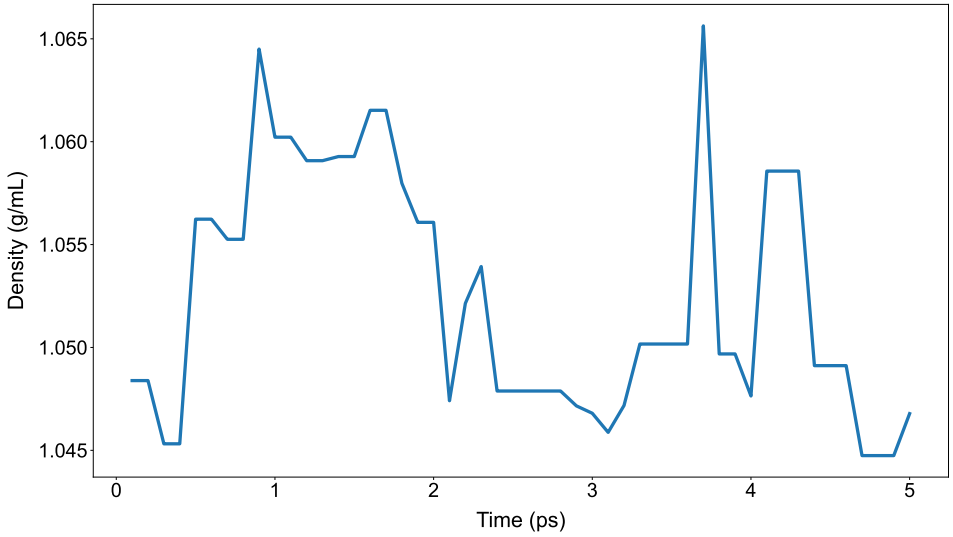

Figure 19: This shows the support for OpenMM csv files. -csvx takes one argument while -csvy can take multiple.

Command for Fig. 20:

```
plotxvg -f other_files/gaussian_hill_2d.xvg -heatmap -save \
  plotxvg_tests-output/19heatmap.pdf -noshow
```

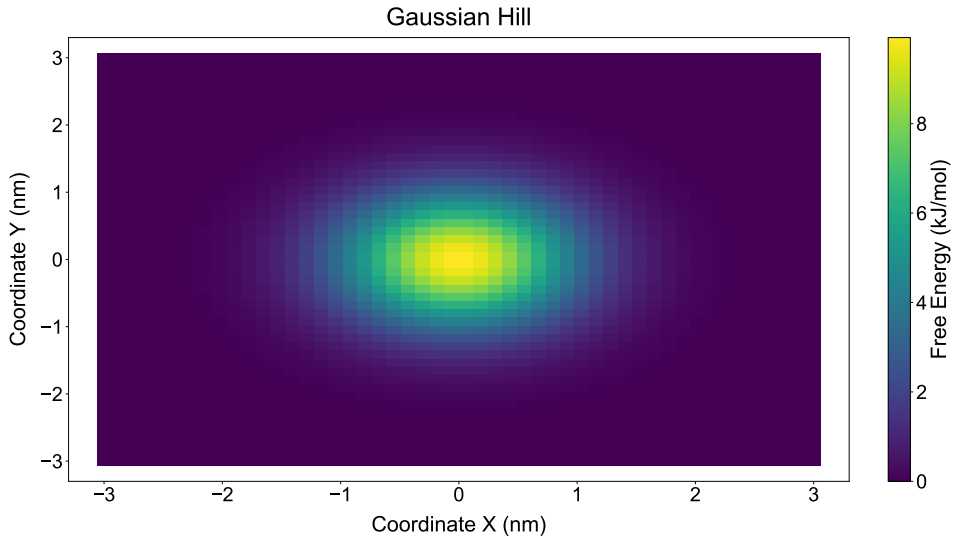

Figure 20: Heatmap using matplotlib's pcolormesh

Command for Fig. 21:

```
plotxvg -f other_files/gaussian_hill_2d.xvg -contour -save \
  plotxvg_tests-output/20contour.pdf -noshow
```

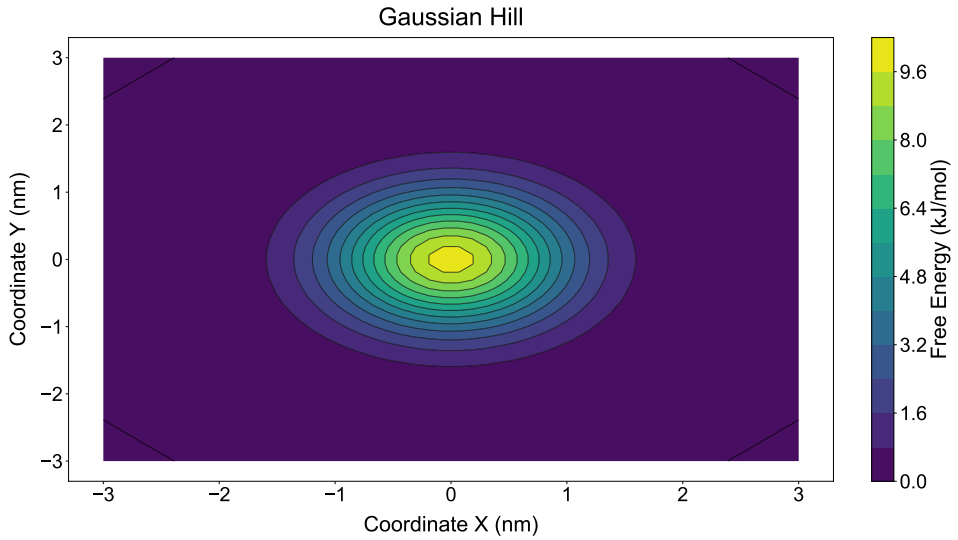

Figure 21: contour using matplotlib's contourf and contour

Command for Fig. 22:

```
plotxvg -f gmx_files/rmsd_calpha.xvg \
  gmx_files/temp_press.xvg gmx_files/gyrate.xvg \
  gmx_files/2dproj_PC1_PC2.xvg -ls solid solid solid solid \
  dotted dashdot dashed None -mk None None None o + x ^ + \
  -panels -tfs 45 -alfs 40 -mksize 20 -mkwidth 4 -save \
  plotxvg_tests-output/21fig1_article.pdf -noshow
```

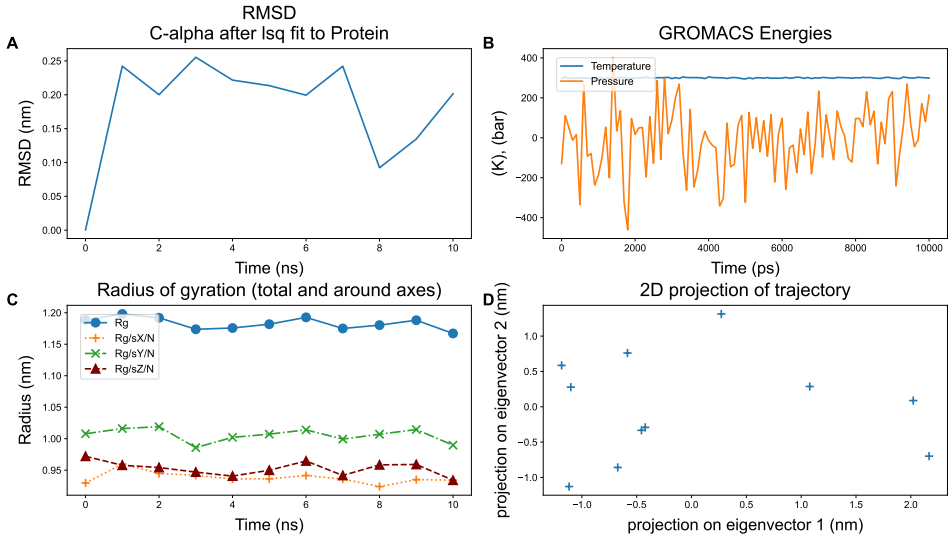

Figure 22: The exact run for reproducing Fig.1 in the article.

Command for Fig. 23:

```
plotxvg -f act_files/COULOMB-PC-elec.xvg \
  act_files/COULOMB-PC+GS-elec.xvg -dslegends 'PC-elec' \
 'PC+GS-elec' -lfs 18 -eqax -sharelabel -stats -panels side \
  -save plotxvg_tests-output/22fig2_article.pdf -noshow
```

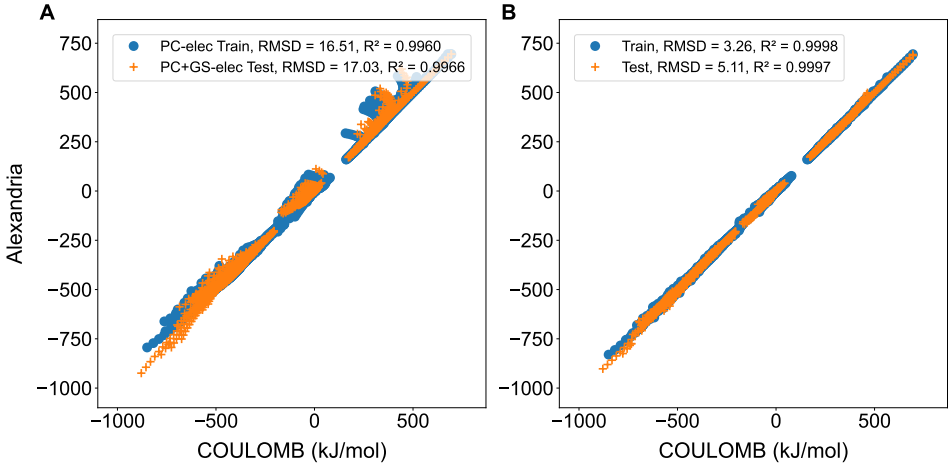

Figure 23: The exact run for reproducing Fig.2 in the article.

Command for Fig. 24:

```
plotxvg -f other_files/openmm.csv -csvx 2 -csvy 7 -alfs 38 \
  -ls solid -save plotxvg_tests-output/23fig3_article.pdf \
  -noshow
```

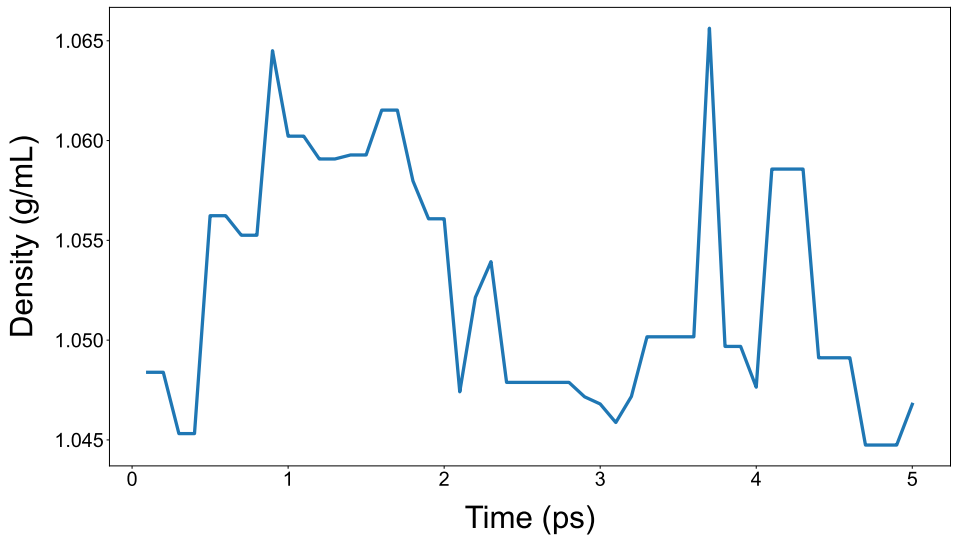

Figure 24: The exact run for reproducing Fig.3 in the article.