# kallisto

*Release 1.0*

**Nov 11, 2020**

# Quickstart

You found a bug? No problem, just open an issue at github.

CHAPTER 1

# Setup and Installation

This guide deal with the general setup and local installation of the `kallisto` program.

**Contents**



## 1.1 Getting the Program

*kallisto* runs on *python3*

### 1.1.1 Setup virtual environment

Python development setup. Install the *pyenv* python version manager:

```
> curl https://pyenv.run | bash
```

and add this to the *~/.bashrc* and source it:

```
> export PATH="~/.pyenv/bin:$PATH"
> eval "$(pyenv init -)"
> eval "$(pyenv virtualenv-init -)"
```

Install the latest python versions

```
> pyenv install 3.8.2
> pyenv install 3.7.7
> pyenv local 3.8.2 3.7.7
```

You could also take *conda* to build a new virtual environment,

```
> conda create --name kallisto python=3.8
```

however, problems could occur while running the test suite due to some incompatibilities between *poetry* and *conda*, which may at the time of reading already been solved.

### 1.1.2 Install *kallisto*

Clone the repository

```
> git clone git@github.com:f3rmion/kallisto.git
```

Install a python package manager, where we choose to go with poetry

```
> curl -sSL https://raw.githubusercontent.com/python-poetry/poetry/master/get-poetry.
↪py | python
> source ~/.poetry/env
```

or alternatively via *pip*

```
> pip install --user poetry
```

Now, if you haven't already done so, change into the cloned *kallisto* directory and download the dependencies via *poetry*:

```
> cd kallisto
> poetry install
```

Finally install the test automation environment nox via *pip'*

```
> pip install --user --upgrade nox
```

Run *nox* to test the setup (this may fail when you are using *conda*).

## 1.2 Getting Help from `kallisto`

Beside this manual you can check the in-program help by

```
> kallisto --help
```

### 1.2.1 The Verbose Mode

If you think some information is missing in your calculation you can switch to the verbose mode by using `--verbose` in the command line arguments.

CHAPTER 2

# Commandline Usage

The *kallisto* program is intended to be applied via the commandline. We use *click* to build *kallisto*, which enables us to define subcommands in a clear and structured way.

**Contents**



## 2.1 Subcommands

The most basic properties in `kallisto` are obtained by sub-commands: coordination numbers (cns), electronegativity equilibration atomic partial charges (eeq), atomic static polarizabilities (alp), and charge dependent atomic van der Waals radii (vdw).

**Coordination number**

**command** `cns`

**input** `--inp <str>` (optional)

**default** `coord` (*Turbomole*)

**description** input file in *xyz* or *Turbomole* format (string)

**cntype** `--cntype {exp, cov, erf}` (optional)

**default** `cov`

**description** choose between different damping functions (string)

**output** standard output (or to file with name `<output>`)

```
> kallisto cns --inp <str> --cntype <str> <output>
```

**Electronegativity equilibration atomic partial charges**

**command** `eeq`

**input** `--inp <str>` (optional)

**default** `coord` (*Turbomole*)

**description** input file in *xyz* or *Turbomole* format (string)

**charge** `--chrg <int>` (optional)

**default** `0`

**description** absolute charge of the molecule (integer)

**output** standard output (or to file with name `<output>`)

```
> kallisto eeq --inp <str> --chrg <int> <output>
```

**Static atomic polarizabilities**

**command** `alp`

**input** `--inp <str>` (optional)

**default** `coord` (*Turbomole*)

**description** input file in *xyz* or *Turbomole* format (string)

**charge** `--chrg <int>` (optional)

**default** `0`

**description** absolute charge of the molecule (integer)

**output** standard output (or to file with name `<output>`)

```
> kallisto alp --inp <str> --chrg <int> <output>
```

**Charge dependent van der Waals radii**

**command** `vdw`

**input** `--inp <str>` (optional)

**default** `coord` (*Turbomole*)

**description** input file in *xyz* or *Turbomole* format (string)

**input** `--vdwtype <str>` (optional)

**default** `rahm`

**description**

reference atomic van der Waals radii
rahm (DOI: 10.1002/chem.201700610)
truhlar (DOI: 10.1021/jp8111556)

**charge** `--chrg <int>` (optional)

**default** `0`

**description** absolute charge of the molecule (integer)

**charge** `--angstrom` (optional)

**default** radii in Bohr

**description** print out van der Waals radii in Angstrom instead of Bohr.

**output** standard output (or to file with name `<output>`)

```
> kallisto vdw --inp <str> --chrg <int> <output>
```

**Write out connectivity of underlying structure**

**command** `bonds`

**input** `--inp <str>` (optional)

**default** `coord` (*Turbomole*)

**description** input file in *xyz* or *Turbomole* format (string)

**input** `--partner <int>` (optional)

**description** write out partner for atom (indexing starts with `0` for the first atom)

**input** `--constrain` (optional)

**default** False

**description** write out `constrain.inp` file in `xtb` format. Constrains all bonds in structure.

**output** standard output (or to file with name `<output>`)

```
> kallisto bonds --inp <str> --partner <str> --constrain <output>
```

**Sort underlying structure according to a breadth first search (BFS) algorithm with respect to connectivity**

**command** `sort`

**input** `--inp <str>` (optional)

**default** `coord` (*Turbomole*)

**description** input file in *xyz* or *Turbomole* format (string)

**input** `--start <int>` (optional)

**default** `0`

**description** define the start of the BFS sorting.

**output** standard output (or to file with name `<output>`)

```
> kallisto sort --inp <str> --start <int> <output>
```

CHAPTER 3

# Need Help

If you’re heading trouble please contact hello at eikecaldeweyher.de

CHAPTER 4

# License