Configuring runxtb

Setting up xtb (and crest) for the use with this repository

Obtain the latest version of xtb from its GitHub page. There you will also find the crest binary. You may also want to check the manual of it on read the docs. At the time of writing the following files should be downloaded: `xtb-200316.tar.xz` and `crest-200219.tgz` corresponding to xtb v6.2.3 and crest v.2.9.

Pick a directory, where you would like to install the packages. For myself I chose to install it as a separate user, e.g. software, and I picked the root directory as:

`/home/software/chemsoft/xtb`

Change to this directory and unpack the downloaded xtb archive:

`tar xf /path/to/xtb-*.tar.xz`

This will create a new subdirectory. In the avove mentioned case this will be:

`/home/software/chemsoft/xtb/xtb_6.2.3`

For the installation of `runxtb` this will also be `xtb_install_root`. There are plenty of other ways to do this, but if you are wondering about these, you probably do not need this guide.

For convenience I'd like to use the `runxxtb` script to also be useable with crest. I found that bundeling the crest executable into the repository works best. Therefore, change to the `bin` directory of the newly created directory; for the example this would be:

`/home/software/chemsoft/xtb/xtb_6.2.3/bin`

Simply unpack the crest archive here, as in the past it usually only contained the crest executable:

`tar xf /path/to/crest-*.tgz`

Make sure that `xtb` and `crest` are marked as executable.

This already concludes the basic set-up, which I recommend for using the `runxtb` wrapper.

Using enso with this wrapper

The latest version of enso can also be found on GitHub. In the past, it was possible to use it in conjunction with this script in a similar fashion as with crest, described above. However, I have not yet had the time to repeat the process and write a guide on it. I'll certainly will come around to it, but for the meantime, I just skip it.

There is an older (probably outdated) guide here, which may serve as a starting point, if you would like to try it out yourself.

Installing and configuring runxtb

Get the latest release of the repository from GitHub. I personally install it alongside the original binary, i.e. `/home/software/chemsoft/xtb`. Unpack the contents, and if necessary rename the directory. I just let GitHub do the packing, since there is no compiling to be done, you can just work with the source code. If I remember correctly, GitHub simply uses the tag name for the created archive file. So depending on your taste and organisation of directories, you might want to move this. Let's assume we have extracted the content to the following directory:

`/home/software/chemsoft/xtb/runxtb.bash_0.4.0`

You can of course also clone this repository.

Change to the new directory. If you have a previous version of the script, it might be possible to import your settings. This is dependent on where your installed them. The probably easiest way is to copy the configuration file into this directory as `.runxtbrc`, as this is the first file that will be applied if found. The configuration files of version 0.3.x should be fine, if they are older, they cannot be imported.

Note that if you have an earlier version of the script installed, and created symbolic links for this script, you should remove them. For example:

```
rm -vi ~/bin/runxtb ~/bin/crest.prepare
```

Now you can simply run the configure script, which will prompt for the values with a short description. If you are using a fresh installation, it will recover the default settings from the supplied `runxtb.rc`. At the end, you will be prompted for a location for the configuration file. Keep in mind that the wrapper script will first look for a file `.runxtbrc`, then for a file `runxtb.rc`, in directories of the following order: `scriptpath`, `/home/$USER`, `/home/$USER/.config`, and `$PWD`. If a `.runxtbrc` is found, it will skip `runxtb.rc`. The location the script has predefined is `$PWD/runxtb.rc`, hence overwriting the provided example file, if it is launched from the installation directory of runxtb. As an alternative option, it will suggest the (safer) location `'scriptpath'/.runxtbrc`, which can be chosen by entering auto. After that, the configuration script will ask about creating symbolic links in `~/bin`. If you choose to do that, you can basically access the script from anywhere in your file system.

The configuration is now complete.

Examples for testing

The following are some suggestions how you might want to test your installation, of course you can also dive right into the real work. If you are not sure, or using it the first time, these are the basic steps for working with this script:

0. Create a directory for testing, e.g. `~/test_xtb`.
1. Generate a structure, with your preferred editor, or from a SMILES, or from a crystal structure, etc.. For example: butan-2-one with Open Babel:

   ```
   obabel -:'CC(=O)CC' --gen3d -oxyz -Ostart.xyz
   ```

2. Run a calculation with xtb. This can be an optimisation, or just a single point. If you want to perform a conformational analysis later, you should choose to optimise the structure with xtb at the same settings as you later intend to use with crest. This is recommended as crest uses this structure for sanity checks. Note that the double dashes `--` devide the options for `runxtb` from the options send to `xtb` (or `crest`). For example:

   ```
   runxtb -- <XYZ> --opt --gbsa <SOLVENT>
   ```

   Check the output. You should immediately see whether this worked or not.

3. The conformational check, or conformer-rotamer-ensemble (CRE), can be done with crest. The little script `crest.prepare` simply takes a `<XYZ>` (if found `xtbopt.xyz`) from the current directory, converts it to a `coord` (via Open Babel), and puts it into a directory called `crest`. (Run `crest.prepare -h` for possible options.) On default settings, for example:

   ```
   crest.prepare
   cd crest
   ```

   Now you are ready to run the calculation. If you have followed the instrructions above, this can be done with

   ```
   runxtb -C crest -- -nmr -gbsa <SOLVENT> -chrg <INT> -uhf <INT>
   ```

   The output of this step will write to `crest.runxtb.out`. This might take a few minutes, or longer depending on the molecule you chose. For larger molecules the submit funktion (`-s` or `-S`) may come in handy. Check the output. If everythin went well, you will have a file `crest_best.xyz` with the lowest lying conformation.

   Now testing is complete.

(Martin; 2020-03-29; wrapper version 0.4.0.rc)