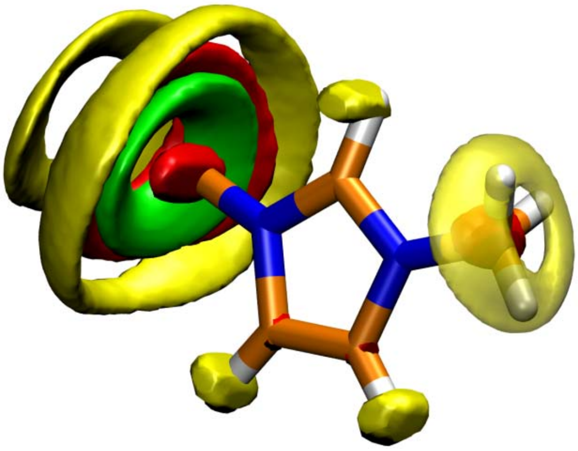
TRAVIS

Trajectory Analyzer and Visualizer



Quick Start Guide

(c) Martin Brehm, Kirchner Group,
University of Leipzig, 2009-2011

http://www.uni-leipzig.de/~travis

Table of Contents

1. Introduction

TRAVIS („*TRajectory Analyzer and VISualizer*“) is a program package for analyzing and visualizing Monte Carlo and molecular dynamics trajectories. TRAVIS is open-source freeware and licensed under the terms of the GNU General Public License v3 (see section *License*).

This Quick Start Guide briefly explains how to install TRAVIS and how to perform some basic analyses. The full manual will be available in some weeks. However, as the operation of TRAVIS is interactive (it asks questions that the user has to answer), it should be self-explanatory to some degree. Feel free to try out advanced features which are not covered in this short guide.

A good overview on the features included in TRAVIS can be gained by reading our article, which has shortly been published:

M. Brehm and B. Kirchner: "TRAVIS - A free Analyzer and Visualizer for Monte Carlo and Molecular Dynamics Trajectories"
J. Chem. Inf. Model., **2011**, DOI 10.1021/ci200217w

Please cite this article in every scientific work which uses results obtained from TRAVIS.

If you are not allowed to access the full text version of this article, you can write an email to the TRAVIS developers (see `http://www.uni-leipzig.de/~travis`) in order to receive a personal copy.

2. Compiling and Installing TRAVIS

This section explains how to get a working version of TRAVIS. Except for some features, TRAVIS is platform-independent and should run on almost every machine that offers a working C++ compiler.

Please note that some useful features like self-debugging (if TRAVIS crashes, it displays the source code line where the crash appeared and the full call stack – there is no need to run TRAVIS in a debugger) are only available when running in an Linux environment. **We highly recommend to operate TRAVIS under Linux.** Around 90% of the testing has been performed on machines running 64 Bit Linux.

2.1. On Linux/UNIX machines

As a first step, you should download the source code of the current TRAVIS version from `http://www.uni-leipzig.de/~travis`. Extract the contents of the archive file to a new and empty directory.

Open the file named "Makefile" with a standard text editor of your choice. Within the first lines, you can see the entry "`CXX = g++`". Replace "`g++`" with the C++ compiler that you want to use. Make sure that the compiler is working. In principle, TRAVIS can be compiled with every C++ compiler. However, most of the testing has been performed with a version compiled by the free compiler g++ (from the GNU Compiler Collection) on 64 Bit Linux.

As TRAVIS does not require any external library (expect the C/C++ standard libraries, which are included in your compiler), there is no additional configuration necessary.

To compile TRAVIS, just type "`make`" in the base directory of the TRAVIS source.

If everything worked, there should now be an executable file with the name “travis” located in the “exe” subdirectory. This is your TRAVIS executable. There is no installation procedure needed; feel free to copy this file to a location of your choice (e.g. to “`/usr/bin`” or maybe a bin directory in the home of a specific user). Make sure that this location is included in the “`PATH`” environment variable; this will enable you to execute TRAVIS by just typing “`travis`”.

Try out if your version of TRAVIS is working by just starting it (type “`travis`”) with no further parameters. If everything is Ok, there should be no error message and a short list of command line parameters should appear. If there are problems, see the section “Troubleshooting”.

2.2. On Microsoft Windows machines

For Windows users, the normal way of obtaining a program is to download a pre-compiled version, which already contains the executable (“`.exe` file”). If you do not have a C++ compiler running on your windows machine, this is the preferred way for TRAVIS also. You can download a pre-compiled Windows executable of TRAVIS from `http://www.uni-leipzig.de/~travis`. This file should be functional on all versions of Windows, starting from Windows 2000.

As TRAVIS is a text-mode application and has no graphical user interface, you have to use the Windows console to operate TRAVIS. Start TRAVIS from the console and specify the command line arguments that you require, as described in the next section.

If you decide to compile TRAVIS on Windows, you first should create an empty Windows Console application in the C++ development environment of your choice. The next step is to download the TRAVIS source code, extract the files from the archive and copy all files from the “src” subdirectory to the project directory of your just created empty application. Depending on your development environment, maybe you now have to add the copied files to your application.

Now open the file “config.h”. You should find the following two lines:

```
//#define TARGET_WINDOWS
#define TARGET_LINUX
```

To tell the compiler that you want to compile a Windows version of TRAVIS, change these lines to the following:

```
#define TARGET_WINDOWS
//#define TARGET_LINUX
```

Now you should be able to compile the application. As already mentioned in the Linux part, there are no external libraries necessary to build TRAVIS.

3. Using TRAVIS / General Hints

3.1. Running TRAVIS / File Formats

As TRAVIS is a text-mode program and does not include a graphical user interface, it should generally be run from the console. Most parameters are interactively obtained from the user.

We recommend to perform only one kind of analysis per TRAVIS run. In principle, you can select several analyses, but there are thousands of possible combinations so that it is impossible to check if all conbinations are working fine.

When running TRAVIS, you should specify the name of the trajectory file that you want to analyze. This is done with the "`-p`" command line option ("`p`" like "`positions`"). The input trajectory has to be in `XYZ` or `PDB` format, and has to bear the corresponding file extension.

For example, in a Linux environment, the command to run TRAVIS could look like this:

```
      travis –p ../trajectories/trajec1.xyz
```

As you can see, the trajectory to analyze does not have to be located in the current working directory. All results of the analysis are written to the current working directory. **TRAVIS never overwrites any file**. Imagine TRAVIS wants to write a file named "`output.txt`". If this file is already existent, it is renamed to "`#1#output.txt`". If this file is also existing, the name "`#2#output.txt`" is chosen instead, and so on. After running TRAVIS, the most recent file will always be called "`output.txt`". The file that was most recent before this run will bear the highest number. A file bearing the "`##`" label will never be renamed again by TRAVIS.

When running, TRAVIS always writes a log file called "`travis.log`". This file contains all the information that is written to the screen. Therefore, there is no need to manually redirect the TRAVIS screen output to a file, which is anyhow not recommendet, as the screen output contains the control sequences for text colors, that will not display correctly in an editor.

TRAVIS creates several types of output files. A list of them and some hints how to handle them follows here:

- **CSV**: Plain text format containing columns of data that are separated by a "`;`". Can be opened with most plotting tools, like e.g. Microsoft Excel, OpenOffice Calc and xmgrace. To open these files with xmgrace, use the command line "`xmgrace -block file.csv -bxy 1:2`", depending on which columns you want to plot.

- **AGR**: Text file containing style information for xmgrace. Can only be opened with xmgrace.

- **CUBE**: Gaussian Cube file. This is a volumetric data file, containing data on a three-dimensional grid. Cube files are in text format. They can be opened with many visualisation programs, like e.g. VMD, which we recommend.

- **PLT**: gOpenMol PLT file. Like CUBE files, PLT files contain data on three-dimensional grid. PLT files are binary files and therefore much smaller than the corresponding CUBE file. They also can be visualized with VMD.

- **NB**: Mathematica Notebook file. Contains data on a two-dimensional grid. Is a text file, but can only be used with Wolfram Mathematica, which is great for creating these kinds of plots.

- **DAT**: Plain text file, format is depending on the analysis you performed.

3.2. Answering the Questions

As already mentioned, TRAVIS is an interactive program and asks questions to the user. We hope that most of the questions are somewhat self-explanatory.

Most of the questions have a default answer, which is shown in square brackets after the question. A general rule is: If you do not fully understand a question or are not sure which answer to choose, **the default value will be a good choice**. Some questions really go into complicated details, and values other than the default are only rarely needed there.

There is no possibility to go back one or more steps, if an incorrect answer has been given. This is not possible for technical details. After each question, TRAVIS initializes some internal data structures, creates objects, and so on. To unroll these steps in a reliable way would make it necessary to track all changes and to maintain an undo list, which is more work than this feature would be worth in our opinion.

However, if you passed an incorrect answer to TRAVIS, your preceding input is not lost. While answering the questions, TRAVIS writes an input file (called “`input.txt`”), containing all your input, together with short descriptions of the questions. If you mistyped something, you should abort TRAVIS (by pressing CTRL+C) and open the `input.txt` in a text editor. Locate the incorrect input and delete all lines from this point on up to the end. Be sure not to leave some empty lines, as they also will be regarded as input.

Now you can start TRAVIS again, but with specifying the modified input file as input source. This is done with the “`-i`” command line parameter:

```
      travis –p ../trajectories/trajec1.xyz -i input.txt
```

When the input file reaches its end and there are still questions to answer, TRAVIS switches back to keyboard input and you can continue.

3.3. Scripting and Batch operation

Sometimes you will have to perform many analyses of the same type on different trajectories, or many analyses that differ only a little on the same trajectory. In these cases, it would be a very inefficient way to operate TRAVIS manually.

The idea behind our scripting support is quite easy: As already mentioned, TRAVIS writes an input file if the input comes from the keyboard. To prepare many analyses, run TRAVIS once and answer all the questions how you need to. When the main analysis starts, cancel it by pressing CTRL+C. Now take the `input.txt` that has been written, make copies of it for each analysis you want to perform, and change the answers in that file according to the different analyses. Create subdirectories for all analyses, copy the input files to the corresponding subdirectories and write a small batch script which runs TRAVIS successive in each directory. If your subdirectories all have names starting with "`dir`", this could, e.g., look like the following:

```
for i in dir*
do
  cd $i
  travis -p /home/user1/trajectories/traj1.xyz -i input.txt
  cd ..
done
```

Please do never run two instances of TRAVIS in the same directory at the same time. As they both will attempt to write to the "`travis.log`" file, at least one of the instances will crash. It is a good practice to have a separate directory for each analysis.

4. Troubleshooting

In general, if you face problems with TRAVIS and made sure that these problems are not related to your machine and your operating system, you may contact the developers (see `http://www.uni-leipzig.de/~travis`). However, please note that a reply to your question might take quite a long time, as the developers have many projects to run they are paid for, and supporting TRAVIS users is officially none of them, but done voluntarily.

When running in a Linux environment, TRAVIS has the self-debugging feature: If TRAVIS crashes, it displays the source code file name and line number where the crash appeared and the full decoded calling stack – there is normally no need to run TRAVIS inside a debugger. If you have problems with program crashes, please send the `travis.log` file, which also contains the self-debugging information, to the developers.

Some general hints to avoid problems:

- If you have problems with the color of the TRAVIS screen output, you should switch your color scheme to "bright text on dark background". If you do not want this or the problems persist, you can specify the command line option "`-nocolor`" when running TRAVIS. Please note that this will make the screen output much less clear, as all information will be printed in the same color now.

- Make sure that TRAVIS has the rights to write files in the current working directory. If this is not the case, TRAVIS will not be able to run.

There is NO support on how to modify the TRAVIS source code. However, suggestions for changes and new features are welcome.

5. License

TRAVIS is licensed under the terms of the GNU General Public License v3.

In short terms, this means that you may freely download and use TRAVIS. You are allowed to change, copy and distribute any version of TRAVIS, as long as you do this also under the terms of the GNU GPL and do not remove the names of the original authors and the “credits” section. It is not allowed to sell copies of TRAVIS or own programs that use any piece of the TRAVIS source code. If you make any changes to TRAVIS, you have to distribute the modified source code; it is not allowed to just distribute the binary file.

You can find details on “`http://www.gnu.org/copyleft/gpl.html`”. The full text of the license can also be found on the TRAVIS homepage (`http://www.uni-leipzig.de/~travis`) and is included in the TRAVIS downloads.

6. Credits

Persons that contributed to TRAVIS:

Martin Brehm	Main Developer
Barbara Kirchner	Group Leader

Marc Bruessel	Testing and Debugging
Philipp di Dio	Atom Parameters and Math Support
Dorothea Golze	Testing and Debugging
Daniela Kerlé	Name Finding
Miriam Kohagen	Testing and Debugging
Fred Malberg	Testing and Debugging
Jens Thar	Fruitful Discussion and Scientific Input
Henry Weber	Testing, Debugging and creative Ideas