Sage Installation Guide

Release 4.0.1

The Sage Development Team

June 06, 2009

CONTENTS

This is a brief explanation of the installation of Sage. Once Sage is installed, you can easily upgrade to a more recent version using `sage -upgrade` (or type `sage -h` for more options regarding the installation of Sage packages.)

CHAPTER

ONE

INTRODUCTION

You can install Sage either from a pre-built binary tarball or from source. The binary method is fastest, and has the fewest prerequisites. The source method gives you access to possibly a slightly more up-to-date version of Sage, and ensures that you can modify any of the Sage source code and recompile. Also, installing from source should be simpler than you're used to with most software, since much testing is done to make sure Sage and all its components can be successfully compiled with no user interaction on a range of computers. Thus it's probably easier for you to build all of Sage from source than it would be for you to build some of the packages that come with Sage (e.g., Singular).

The Sage distribution includes most programs on which Sage depends – see a partial list below. These programs are all released under a GPL-compatible license (see the COPYING.txt file in the Sage home directory for more details).

Here is a list of some of the software included with Sage:

- atlas: The ATLAS (Automatically Tuned Linear Algebra Software) project
- bzip2: bzip2 compression library
- clisp: Common lisp interpreter
- cython: the Cython programming language: a language, based on Pyrex, for easily writing C extensions for Python
- eclib: John Cremona's programs for enumerating and computing with elliptic curves defined over the rational numbers
- ecm: elliptic curve method for integer factorization
- flint: fast library for number theory
- fortran: the Fortran programming language
- GAP: A System for Computational Discrete Algebra
- genus2reduction: Reduction information about genus 2 curves
- gfan: Computation of Groebner fans and toric varieties
- ghmm: the hidden Markov model library
- givaro: a C++ library for arithmetic and algebraic computations
- gmp-mpir: MPIR is an open source multiprecision integer library derived from GMP (the GNU multiprecision library)
- gsl: GNU Scientific Library is a numerical library for C and C++ programmers

- ipython: An enhanced Python shell designed for efficient interactive work, a library to build customized interactive environments using Python as the basic language, and a system for interactive distributed and parallel computing
- jmol: a Java molecular viewer for three-dimensional chemical structures
- jsmath: include mathematics in HTML
- lapack: a library of Fortran 77 subroutines for solving the most commonly occurring problems in numerical linear algebra.
- lcalc: Rubinstein's L-functions calculator
- libfplll: contains different implementations of the floating-point LLL reduction algorithm, offering different speed/guarantees ratios
- libm4ri: Library for matrix multiplication, reduction and inversion over GF(2)
- linbox: C++ template library for exact, high-performance linear algebra computation
- matplotlib: a Python 2-D plotting library
- maxima: symbolic algebra and calculus
- mercurial: a Source Control Management system designed for handling of very large distributed projects
- mpfi: a C library for arithmetic by multi-precision intervals, based on MPFR and GMP
- mpfr: a C library for multiple-precision floating-point computations with correct rounding
- networkx: a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks
- NTL: number theory C++ library
- numpy: numerical linear algebra and other numerical computing capabilities for Python
- palp: a package for analyzing lattice polytopes
- pari: PARI number theory library
- pexpect: Python expect (for remote control of other systems)
- polybori: provide high-level data types for Boolean polynomials and monomials, exponent vectors, as well as for the underlying polynomial rings and subsets of the power set of the Boolean variables
- pynac: a modified version of GiNaC (a C++ library for symbolic mathematical calculations) that replaces the dependency on CLN by Python
- Python: The Python programming language
- R: a language and environment for statistical computing and graphics
- readline: GNU Readline line editor library
- scipy: scientific tools for Python
- singular: Polynomial computations in algebraic geometry, etc.
- symmetrica: routines for computing in the representation theory of classical and symmetric groups, and related areas
- sympow: Symmetric power L-functions and modular degrees

- sympy: a Python library for symbolic mathematics
- tachyon: Tachyon(tm) parallel/multiprocessor ray tracing software
- termcap: Display terminal library
- Twisted: Networking framework
- zlib: zlib compression library
- zn_poly: C library for polynomial arithmetic in Z/nZ[x]
- ZODB: Zope Object Database

CHAPTER

TWO

PRE-BUILT BINARY INSTALL

2.1 Linux and OS X

Installation from a pre-built binary tarball should in the long run be the easiest and fastest way to install Sage. This is not necessarily the case right now. Note that Sage is itself a programming environment, so building it from source guarantees you maximum flexibility in the long run. Nonetheless, we provide pre-built binaries.

Assumptions: You have a computer with at least 550 megabytes free disk space and the operating system is Linux (32-bit or 64-bit) or OS X.

Highly Recommended: It is highly recommended that you have LaTeX installed.

Download the latest tarball from http://www.sagemath.org/download.html . For example, it might be called `sage-x.y.z-x86_64-Linux.tgz`. Unpack it on your computer in a directory which you have permissions:

```
tar zxvf sage-x.y.z-x86_64-Linux.tgz
```

Change into the directory just created, e.g., `sage-x.y.z-x86_64-Linux` and type `./sage` to run Sage. You can move the directory `sage-x.y.z-x86_64-Linux` anywhere, and still run `sage` from it. You can also copy `sage` and put it anywhere, e.g., `/usr/local/bin/`, but you'll have likely have to edit the `ROOT="....."` line at the top.

We currently distribute .dmg files for OSX. But we would like to make Sage more of a native application. Work for that is ongoing, but help is always welcome.

2.2 Microsoft Windows

The best way to install Sage on Windows is to get the free VMware player and use the VMware Sage appliance, which is available at http://www.sagemath.org/bin/microsoft_windows/ . Be sure to read README.txt in that directory.

CHAPTER

THREE

INSTALL FROM SOURCE CODE

More familiarity with computers may be required to build Sage from source. If you do have all the tools, the process should be completely painless (but it will take your computer a while, though you don't have to watch), and has the major advantage that you have the latest version of Sage, and you can change absolutely any part of Sage or the programs on which it depends and recompile.

As of this writing, Sage is known to work on Linux (32 bit x86, 64 bit x86-64, IA64, or 32 bit PPC) and OS X (10.4 or 10.5, PPC or x86, 32 bit only). (See http://wiki.sagemath.org/SupportedPlatforms for the latest information.)

> **Solaris? FreeBSD? OS X 10.5 in 64 bit mode?**: Complete compilation of Sage is currently not supported on Solaris or *BSD. It is possible to compile most of Sage on Solaris machines and to fill in the extra parts using standard packages; please email sage-devel if you desperately need to run Sage on Solaris. We do plan to fully support Solaris - it's a very important platform. Work is ongoing.
>
> **Sage on FreeBSD**: The binaries can be run with the help of Linux emulation. We are working on a fully native port and the number of issues that need to be fixed are relatively small compared to the other ports.
>
> We hope to support OS X 10.5 in 64 bit mode in our next release.

Assumptions: You have a computer with about 850 megabytes of free disk space running Linux (32-bit or 64-bit) or OS X 10.4 with XCode. In particular, under Linux the following standard command-line development tools must be installed on your computer (under OS X they all come with XCode):

```
gcc  (with C++ support)
make
m4
perl
ranlib
tar
ssh-keygen -- needed to run the notebook in secure mode.
latex -- highly recommended, though not strictly required
```

To check if you have `m4` installed, for example, type `which m4` at a command line. If it gives an error (or returns nothing), then it is not installed. It is highly recommended that you have LaTeX installed, but not required. If you don't have `ssh-keygen` on your local system, then you cannot run the notebook in secure mode. To run it in insecure mode, run the command `notebook(secure=False)` instead of `notebook()`.

In OS X, make sure you have XCode version at least 2.4, i.e., `gcc -v` should output build at least 5363. If you don't, go to http://developer.apple.com/ sign up, and download the free Xcode package. Only OS X

$$\geq 10.4$$

is supported. This will give you all of the above commands.

On a Debian-based system (e.g., Ubuntu), ranlib is in the binutils package. On a newly installed Ubuntu system (this was tested on Ubuntu 7.04), you can install the above commands as follows:

```
sudo apt-get install gcc-4.2-base      # or the latest version available
sudo apt-get install make
sudo apt-get install m4
sudo apt-get install bison
sudo apt-get install flex
sudo apt-get install tar
sudo apt-get install perl
sudo apt-get install binutils
sudo apt-get install libstdc++6-dev
sudo apt-get install g++
sudo apt-get install openssh-client
```

The LaTeX package and a pdf previewer are optional but they can be installed using

```
sudo apt-get install tex-common
sudo apt-get install tetex-base
sudo apt-get install kpdf
```

(You must have the GNU version of `make` installed. For example, Sage won't build on a FreeBSD install that doesn't have the optional GNU version of `make` installed as well (and named `make`).)

Although some of Sage is written in Python, you do not need Python pre-installed on your computer, since the Sage installation includes everything you need. When the installation program is run, it will check that you have each of the above-listed prerequisites, and inform you of any that are missing.

- If you want to use Tcl/Tk libraries in Sage, do the following preferable before compilation. Sage's Python will automatically recognize your system's install of Tcl/Tk if it exists. You need to install the Tcl/Tk development libraries though, not just the Tck/Tk base.

  On Ubuntu, this is the command:

  ```
  sudo apt-get install tk8.5-dev    # or the latest version available
  ```

  Now you can install Sage and Sage's Python will automatically recognize your system's install of Tcl/Tk. If you forgot and installed Sage first anyway, all is not lost. Just issue the command:

  ```
  sage -f python-2.5.2.p8    # or the latest version available
  ```

  after installing the Tcl/Tk development libraries as above. If

  ```
  sage: import _tkinter
  sage: import Tkinter
  ```

  does not raise an ImportError then it worked.

- Sage is currently being developed using GCC Version 4.3.x, and is likely to compile fine with GCC Version 3.4.x and newer (it does not work with older gcc releases). If you are interested in working on support for Intel or Sun's CC compiler, please email sage-devel.

- One reason `perl` is required is that both the NTL and PARI configuration scripts are written in perl.

After extracting the Sage tarball, the subdirectory source contains the source distributions for everything on which Sage depends. We emphasize that all of this software is included with Sage, so you do not have to worry about trying to download and install any one of these packages (such as GAP, for example) yourself.

On tests using various Linux computer systems, the known problems are:

- does not build with gcc 4.3.0 yet, but work is ongoing to fix that.
- Moving the build after compiling breaks the PARI Galois fields database, which appears to be hard-coded into the PARI binary. (Somebody help fix this!)

3.1 Steps to Install from Source

Installation from source is (potentially) very easy, because the distribution contains (essentially) everything on which Sage depends.

> Make sure there are no spaces in the directory name under which you build . Running from a directory with spaces in it is supported but discouraged. Building is not possible, since several of the components of do not build if there are spaces in the path.

1. Go to http://sagemath.org/download-source.html , select a mirror, and download the file sage-*.tar.

   This tarfile contains the source code for Sage and the source for all programs on which Sage depends. Download it into a subdirectory of your home directory into which you want to install Sage. Note that this file is not compressed; it's just a plain tarball (which happens to be full of compressed files).

2. Extract:

   ```
   tar xvf sage-x.y.z.tar
   ```

3. This creates a directory `sage-x.y.z`.

4. Change into that directory

   ```
   cd sage-x.y.z
   ```

   This is Sage's home directory.

5. Optional (but highly recommended): Read the {README.txt} file there.

6. Type

   ```
   make
   ```

   This compiles Sage and all dependencies. Note that you do not need to be logged in as root, since no files are changed outside of the `sage-x.y.z` directory. [1] This command does the usual steps for each of the packages, but puts all the results in the local build tree. This can take close to an hour on some machines. Depending on the architecture of your system (e.g., Celeron, Pentium Mobile, Pentium 4, etc.), it can take over three hours to build Sage from source. If the build is successful, you will not see the word ERROR in the last 3-4 lines of output.

1. To start Sage, change into the Sage home directory and type:

   ```
   ./sage
   ```

[1] There is one exception–the `.ipythonrc` directory is created in your `HOME` directory if it doesn't exist.

The directory where you built Sage is NOT hardcoded into any part of . You should be able to safely move or rename that directory. (It's a bug if this is not the case - unfortunately there is one bug which hasn't yet been fixed along these lines, namely the PARI install hard-codes the location of the "galois data" files. Fixes welcome!)

After you build Sage, you may optionally copy or move the entire build tree to `/usr/local`. You might also copy the `sage-*/sage` script to `/usr/local/bin/` and edit `ROOT="....."` at the top of that file.

You should see the Sage prompt, which will look something like this (starting the first time can take a few seconds):

```
$ sage
----------------------------------------------------------------------
| SAGE Version 3.1, Release Date: 2008-08-16                         |
| Type notebook() for the GUI, and license() for information.        |
----------------------------------------------------------------------
sage:
```

Just starting successfully tests that many of the components built correctly. If the above is not displayed (e.g., if you get a massive traceback), please report the problem, e.g., to http://groups.google.com/group/sage-support . Please include in your email the file `install.log`. It would also be helpful to include the type of operating system you have and the version number (and date) of the copy of Sage you are using. (There are no formal requirements for bug reports - just send them; we appreciate everything.)

After starts, try a command:

```
sage: 2 + 2
4
```

Try something more complicated, which uses the PARI C library:

```
sage: factor(2005)
5 * 401
```

Try something simple that uses the Gap, Singular, Maxima and PARI/GP interfaces:

```
sage: gap('2+2')
4
sage: gp('2+2')
4
sage: maxima('2+2')
4
sage: singular('2+2')
4
sage: pari('2+2')
4
```

(For those familiar with GAP: Sage automatically builds a GAP "workspace" during installation, so the response time from this GAP command is relatively fast. For those familiar with GP/PARI, the `gp` command creates an object in the GP interpreter, and the `pari` command creates an object directly in the PARI C-library.)

Try running Gap, Singular or GP from Sage:

```
sage: gap_console()
GAP4, Version: 4.4.6 of 02-Sep-2005, x86_64-unknown-linux-gnu-gcc
gap> 2+2;
4
[ctrl-d]


sage: gp_console()
...
[ctrl-d]
```

```
sage: singular_console()
                     SINGULAR                                /  Development
 A Computer Algebra System for Polynomial Computations   /   version 3-0-1
                                                       0<
     by: G.-M. Greuel, G. Pfister, H. Schoenemann        \   October 2005
FB Mathematik der Universitaet, D-67653 Kaiserslautern    \
// ** executing /usr/local/sage/sage-0.8.2/bin/LIB/.singularrc
[ctrl-d]
> Auf Wiedersehen.
sage:
```

2. Optional: Check the interfaces to any non-included software that you have available. Note that each interface calls its corresponding program by a particular name: Mathematica is invoked by calling `math`, Maple by calling `maple`, et cetera. The easiest way to change this name or perform other customizations is to create a redirection script in `$SAGE_ROOT/local/bin`. Sage inserts this directory at the front of your PATH, so your script may need to use an absolute path to avoid calling itself; also, your script should use `$*` to pass along all of its arguments. For example, a `maple` script might look like:

   ```
   #!/bin/sh

   /etc/maple10.2/maple.tty $*
   ```

3. Optional: Different possibilities to make using Sage a little easier:

   - Copy `$SAGE_ROOT>/sage` to a location in your `PATH`. If you do this, make sure you edit the line with the `....`'s at the top of the `sage` script.
   - For KDE users, create a bash script {sage} containing the lines

     ```
     #!/bin/bash
     konsole -T "sage" -e <SAGE_ROOT>/sage
     ```

     which you make executable (`chmod a+x sage`) and put it somewhere in your path. (Note that you have to change `$SAGE_ROOT`} above!) You can also make a KDE desktop icon with this as the command (under the Application tab of the Properties of the icon, which you get my right clicking the mouse on the icon).
   - For bash shell users, type `echo $PATH` and `cp sage <your-path-dir>` into one of these directories, or else add this `bin` directory to your `PATH` variable, e.g., if you use the bash shell, add the line

     ```
     PATH="<sage-home-dir>/bin":$PATH
     export PATH
     ```

     in your .bashrc file (if it exists; if not, make one). After doing this and logging out and in again, typing `sage` at a shell prompt should start Sage.
   - On Linux and OSX systems, you can make an alias to `$SAGE_ROOT/sage`. For example, put something similar to the following line in your `.bashrc` file:

     ```
     alias 'sage'='/home/username/sage-3.1.2/sage'
     ```

     Having done so, quit your terminal emulator and restart it again. Now typing `sage` within your terminal emulator should start Sage.

4. Optional: Test the install by typing `./sage -testall`. This runs most examples in the source code and makes sure that they run exactly as claimed. To test all examples, use `./sage -testall -optional -long`; this will run examples that take a long time, and those that depend on optional packages and software, e.g., Mathematica or Magma. Some (optional) examples will likely fail because they assume that a database is installed. Alternatively, from within `$SAGE_ROOT`, you can type `make test` to run all the standard test code. This can take from 30 minutes to an hour or longer.

5. Optional: The directory `spkg/build` contains intermediate code that is used to build sage. Type `make clean` to delete it and a few other directories (e.g., `spkg/archive` and `devel/old`). This is safe and will save you about 500MB disk space. You may wish to type this periodically.

6. Optional: Install optional Sage packages and databases. Type `sage -optional` to see a list or visit http://www.sagemath.org/packages/optional/, and `sage -i <package name>` to automatically download and install a given package.

7. Optional: Run the `install_scripts` command from within Sage to create gp, singular, gap, etc., scripts in your `PATH`. Type `install_scripts?` in Sage for details.

Have fun! Discover some amazing conjectures!

3.2 Installation in a Multiuser Environment

This section addresses the question of how a system administrator can install a single copy of Sage in a multi-user computer network.

3.2.1 System-wide install

This is a compilation of posts to the Sage support list (in particular those of Luis Finotti).

1. Unpack the current Sage tarball (we shall assume it is `sage-2.5.2.tar`) at, e.g., `/usr/local/` and compile it as root. Assuming you are in a root shell and the tarball is in your current directory, type:

   ```
   cp sage-2.5.2.tar /usr/local
   cd /usr/local
   tar xvf sage-2.5.2.tar
   cd sage-2.5.2/
   make
   ```

   ```
   (Comment: It's better to build in place.  It's a bug if anything goes
   wrong when relocating the entire tarball -- unfortunately there
   is one bug I haven't fixed along these lines, namely the
   PARI install hard-codes the location of the "galois data" files.
   (Fixes welcome!))
   ```

2. Make sure to modify the line with the `.....`"'s at the top of the {sage} script. In other words, edit `SAGE_ROOT="....."` to say `SAGE_ROOT="/usr/local/sage-2.5.2"`.

3. There are some initial files that have to be created during the first run of Sage. Try starting up Sage once as root (or, to be more thorough, try `make test` as root to run all the standard test code). You can stop the tests by pressing {ctrl-z} followed by typing `kill %1` (assuming you had no other jobs in the background of that shell).

4. Make a copy of the `sage` script in `/usr/local/bin`:

   ```
   cp /usr/local/sage-2.5.2/sage /usr/local/bin/
   ```

   You make a copy instead of a symlink, since upgrading (with `sage -upgrade}` overwrites `/usr/local/sage-2.5.2/sage`, hence deleting the `ROOT=...` part of that file.

   Make sure that all files in `/usr/local/sage-2.5.2` are readable by all:

```
chmod a+rX -R /usr/local/sage-2.5.2
```

3.3 Special Notes

- (Found by Dorian Raymer) Sage will not build if you have only bison++. You should uninstall bison++ and install bison.
- (Found by Peter Jipsen) If you get an error like

  ```
  ImportError: /home/jipsen/Desktop/sage-1.3.3.1/local/lib/libpari-gmp.so.2:
       cannot restore segment prot after reloc:
  Permission denied
  ```

  then your SELinux configuration is preventing from launching. To rectify this issue, you can either change the default security context for Sage (??) or disable SELinux altogether by setting the line `SELINUX=disabled` in your `/etc/sysconfig/selinux` file.

CHAPTER

FOUR

DESKTOP ICON

These instruction will help you make a KDE desktop icon which starts the Sage notebook. Instructions for a Gnome desktop icon are probably similar.

1. Create a `notebook.sage` file containing only the line

   ```
   notebook(openviewer=True)
   ```

2. In your Desktop subdirectory, create a file `Sage-notebook.desktop` containing the lines

   ```
   [Desktop Entry]
   Comment=
   Comment[de]=
   Encoding=UTF-8
   Exec=/usr/local/bin/sage /home/martin/notebook.sage
   GenericName=
   GenericName[de]=
   Icon=
   MimeType=
   Name=Sage
   Name[de]=Sage
   Path=$HOME
   StartupNotify=true
   Terminal=false
   TerminalOptions=
   Type=Application
   X-DCOP-ServiceType=
   X-KDE-SubstituteUID=false
   X-KDE-Username=
   ```

   You will have to edit the `Exec=` line to point to your `sage` script and your `notebook.sage` file.

3. Right click on the Sage notebook desktop icon and click on `Properties`, then `Application`, then `Advanced Options`, then `Run in Terminal`. If you want to title the xwindow terminal, add in the terminal option box `-T "sage notebook"`.

To quit the Sage notebook, first enter `Ctrl-c` in the xwindow terminal running Sage, then enter `Ctrl-d` to quit Sage in the terminal, and finally close the browser (or browser tab) which was displaying the Sage notebook server.

For a picture for your icon, check out the Sage art at http://wiki.sagemath.org/art.

CHAPTER

FIVE

THE DOCUMENTATION

You do not need to install the documentation separately: it is included with Sage. This includes the tutorial, the manual, the developer's guide, and this installation guide. The tutorial includes a guided tour of Sage, and is a good starting point. The reference manual describes what is available in Sage along with examples of how to use it.

To build the HTML and PDF versions of the documentation, use the command `sage -docbuild {document} {format}`. For example, `sage -docbuild reference html` builds the HTML version of the reference manual; type `sage -docbuild` to see a list of all of the options.

CHAPTER

SIX

INDICES AND TABLES

- *Index*
- *Module Index*
- *Search Page*