September 5, 2019

1 Installation instructions

The following instructions go over the complete installation procedure for compiling the auger-analysis program.

1.1 Folder structure

The folder structure of the auger-analysis program is as follows. The base non-compiled version of the program has:

- `configure`: Script for configuring the compilation and running the installation.
- `doc`: Documentation for the program. Installation and usage instructions.
- `include`: Folder containing all header files.
- `input`: Folder containing input files that are needed by some program options.
- `layout`: Folder containing program GUI layout instructions.
- `set_environment.sh`: Script for setting up environmental variables.
- `setup`: Folder containing additional setup files needed in the configuration.
- `src`: Folder containing all source code files.

The fully compiled version of the program also has:

- `bin`: Folder containing all executables.
- `lib`: Folder containing all created libraries.
- `results`: Default folder for saving analysis results. When cleaning the installation, this folder remains unchanged.
- `start.sh`: Script for running the GUI or batch version of the program.

1.2 Installation prerequisites

For this software to work correctly, ROOT [1], wxWidgets [2] and their prerequisites need to be installed on the system. The following packages are essential on a clean installation of Linux, before either ROOT or wxWidgets can be installed. General purpose prerequisites are:

```
# on Debian related systems
sudo apt-get update
sudo apt-get install binutils git patch build-essential cmake gcc g++ gfortran
    ↪ python doxygen python-dev wget
```

```
# on CentOS related systems, and Fedora 21 or below
sudo yum update
sudo yum install epel-release
sudo yum repolist
sudo yum install binutils git patch cmake gcc gcc-c++ gcc-gfortran python
    ↪ doxygen python-devel wget redhat-lsb-core bzip2
# on Fedora 22 or above related systems
sudo dnf update
sudo dnf repolist
sudo dnf install binutils git patch cmake gcc gcc-c++ gcc-gfortran python
    ↪ doxygen python-devel wget redhat-lsb-core bzip2
```

ROOT software prerequisites are:

```
# on Debian related systems
sudo apt-get install libx11-dev libxpm-dev libxft-dev libxext-dev libgsl0-dev
    ↪ libfftw3-dev libmysqlclient-dev libxml2-dev
# on CentOS related systems, and Fedora 21 or below
sudo yum install libX11-devel libXpm-devel libXft-devel libXext-devel gsl-
    ↪ devel fftw-devel mysql-devel libxml2-devel
# on Fedora 22 or above related systems
sudo dnf install libX11-devel libXpm-devel libXft-devel libXext-devel gsl-
    ↪ devel fftw-devel mysql-devel libxml2-devel
```

wxWidgets software prerequisites are:

```
# on Debian related systems
sudo apt-get install libgtk2.0-dev libc++1
# on CentOS related systems, and Fedora 21 or below
sudo yum install gtk2-devel
# on Fedora 22 or above related systems
sudo dnf install gtk2-devel
```

Newest versions of Linux distributions renamed the `mysql` client. In Debian releases, if installation of `libmysqlclient-dev` fails, use `default-libmysqlclient-dev` instead.

1.3 Installing ROOT

ROOT is a data analysis framework able to handle python or C/C++ code integration. The complete instructions are available on the software page at [1]. We assume here, that `$ROOTSOURCE` is the location of the downloaded source files (for example `/opt/source`) and `$ROOTINSTALL` is the location of the ROOT installation (for example `/opt`).

1.3.1 ROOT 5

Note that the ADST part of the Pierre Auger Observatory software Offline [3] is currently limited to ROOT 5 and will not work correctly for ROOT 6. ROOT version 5.XX.YY is installed as (installation of version 5.34.36 or higher is suggested):

```
cd $ROOTSOURCE
wget -nc https://root.cern.ch/download/root_v5.XX.YY.source.tar.gz
tar -zxf root_v5.XX.YY.source.tar.gz
```

```
mv ./root $ROOTINSTALL/root-5.XX.YY
cd $ROOTINSTALL/root-5.XX.YY
./configure --enable-mysql --enable-python --enable-minuit2 --enable-roofit --
    ↪ enable-tmva --enable-xml --enable-builtin-freetype
make
```

In order to setup ROOT's environmental variables once installed, run the following command:

```
source $ROOTINSTALL/root-5.XX.YY/bin/thisroot.sh
```

The following command should now return the correct version of ROOT:

```
root-config --version
```

Important: ROOT 5 will not compile correctly with g++ compilers 6.3.0 or above (see Chapter 2).

1.3.2 ROOT 6

ROOT 6 can still be used for the portion of the software that handles the MVA analysis and plotting. For this purpose, rewriting of ADST files has been disabled for this version of ROOT, but all other auger-analysis program options are enabled.
ROOT version 6.XX.YY is installed as (installation of version 6.12.00 or higher is suggested to get the most out of TMVA):

```
cd $ROOTSOURCE
wget -nc https://root.cern.ch/download/root_v6.XX.YY.source.tar.gz
tar -zxf root_v6.XX.YY.source.tar.gz
cd $ROOTSOURCE/root-6.XX.YY
mkdir ./builddir
cd ./builddir
cmake ../ -DCMAKE_INSTALL_PREFIX=$ROOTINSTALL/root-6.XX.YY -Dmysql=ON -Dpython
    ↪ =ON -Dminuit2=ON -Droofit=ON -Dtmva=ON -Dxml=ON -Dbuiltin-freetype=ON
cmake --build . --target VDT
cmake --build .
cmake --build . --target install
```

In order to setup ROOT's environmental variables once installed, run the following command:

```
source $ROOTINSTALL/root-6.XX.YY/bin/thisroot.sh
```

The following command should now return the correct version of ROOT:

```
root-config --version
```

Important: ROOT 6 requires `cmake` version 3.4.3 or above, which is not supplied with older Linux distributions (see Chapter 2).

1.4 Installing wxWidgets

wxWidgets is a cross-platform GUI library used for the graphical user interface of the auger-analysis program. The complete instructions are available on the

software page at [2]. We assume here, that `$WXSOURCE` is the location of the downloaded source files (for example `/opt/source`) and `$WXINSTALL` is the location of its installation (for example `/opt`).
wxWidgets version X.Y.Z is installed as (installation of version 3.0.3 or higher is suggested):

```
cd $WXSOURCE
wget -nc https://github.com/wxWidgets/wxWidgets/releases/download/vX.Y.Z/
    ↪ wxWidgets-X.Y.Z.tar.bz2
tar -jxf wxWidgets-X.Y.Z.tar.bz2
cd $WXSOURCE/wxWidgets-X.Y.Z
mkdir ./gtk-build
cd ./gtk-build
../configure --enable-unicode --enable-debug --prefix=$WXINSTALL/wxWidgets-X.Y
    ↪ .Z
make
make install
```

For wxWidgets to be correctly recognized, add the following environmental variables (preferrably in the `~/.bashrc` script):

```
export WXLIN=$WXINSTALL/wxWidgets-X.Y.Z
export PATH=$WXLIN/bin:$PATH
export LD_LIBRARY_PATH=$WXLIN/lib:$LD_LIBRARY_PATH
```

The following command should now return the correct version of wxWidgets:

```
wx-config --version
```

1.5 Installing ADST reader

The ADST reader is part of the Offline software developed by the Pierre Auger Observatory [3]. A handful of versions have been included with this software and are automatically installed, when needed. See section 1.7 for installation of the auger-analysis program.
If a new version of the ADST reader is needed, add its tar-ball into the `setup` directory and name it `ADST_vXrYpZ.tar.gz`, where `XrYpZ` denotes the version. The configure script will then automatically recognize that there is an additional version of the ADST reader available.
Important: The current version of ADST compiles with ROOT 6, but it does not work correctly, so any ADST file reading options have been disabled for this version of ROOT. See Chapter 2 for a possible solution.

1.6 Installing external TMVA

TMVA is the multivariate analysis package for ROOT [4]. New versions have been implemented directly into ROOT 6, while ROOT 5 can use an external version of TMVA. For this purpose, TMVA 4.2.0 has been included with this software and is automatically installed, when using ROOT 5. See section 1.7 for installation of the auger-analysis program.

1.7 Installation of the auger-analysis program

Once the above prerequesites and installations have been satisfied, it is now possible to install the auger-analysis program. The complete program is compiled and installed through a single call of the configure script:

```
./configure vXrYpZ
```

Here, the program is installed using ADST version `XrYpZ`, but other options can simply be checked by running `./configure` or `./configure help`. The program can then be run in batch mode for just rewriting ADST files:

```
./start.sh rewrite vXrYpZ [input ADST files]
```

or in graphical mode, with access to all options of the program:

```
./start.sh vXrYpZ
```

The program itself has been created in order to need minimal setup before installation. Currently, the program includes 24 different extensive air shower observables, which are listed in the `input/observables.txt` file of the auger-analysis program. In order to include additional observables, the user needs to add them to this file and implement them in the source files. Similarly, the `input/mva_options.txt` file of the auger-analysis program lists the different possible MVA methods and their options. The user can add more to the list or change the existing options. For instructions on how to perform both tasks, please see usage instructions in the second documentation file. Note that both the observables and MVA methods are read into the program when starting it with the `start.sh` script.

2 Linux distribution solutions

The following instructions go over different Linux distributions and possible solutions, if the software is not compiling or working correctly.

2.1 Comparison of linux distributions

Tab. 2.1 shows a comparison of Linux distributions that are based on Debian and their versions. This makes it easier to determine the distribution version that should be used. Note that there might be some differences in package options for different distributions.

Table 2.1: Comparison of Linux distribution versions based on Debian.

Debian	Ubuntu	Mint	elementary OS
8 (jessie)	14.04 (trusty)	17 (Qiana, Rebecca, Rafaela, Rosa)	0.3 (Freya)
9 (stretch)	16.04 (xenial)	18 (Sarah, Serena, Sonya, Sylvia)	0.4 (Loki)
10 (buster)	18.04 (bionic)	19 (Tara, Tessa)	5.0 (Juno)

In order to correctly install external software ROOT, wxWidgets and ADST, a correct version of `g++` and `cmake` compilers is needed. Tab. 2.2 lists the default compilers supplied to clean versions of different Linux distributions.
Important: ROOT 5 needs `g++` compiler versions below 6.3.0, ROOT 6 needs `cmake` versions 3.4.3 or above, and the complete installation needs `g++` compiler versions 4.8 or above (for c++11 extensions).

Table 2.2: Comparison of default `g++` compiler and `cmake` versions in different Linux distributions.

Distribution	`g++` version	`cmake` version
Debian 8 (jessie)	4.9.2	3.0.2
Debian 9 (stretch)	6.3.0	3.7.2
Ubuntu 14.04 (trusty)	4.8.2	2.8.12.2
Ubuntu 16.04 (xenial)	5.3.1	3.5.1
Ubuntu 18.04 (bionic)	7.3.0	3.10.2
CentOS 7	4.8.5	2.8.12.2
Fedora 23	5.3.1	3.4.3
Fedora 25	6.4.1	3.9.1
Fedora 27	7.3.1	3.11.2
Fedora 29	8.3.1	3.12.1

2.2 Software installation tests

Installation of the complete software package has been performed on a range of different clean Linux distributions listed in Tab. 2.3. The installed software versions for these test installations were ROOT 5.34.36, ROOT 6.14.00, wxWidgets 3.0.4, ADST v3r3p4 and external TMVA 4.2.0.

2.3 Solution to the ROOT version problem

In order to enable the complete auger-analysis software, ROOT 5 needs to be installed on your current system. However, this coupled with the need for c++11 extensions during compilation, the version of the `g++` compiler needs to be between 4.8 and 6.3 (ROOT 5 will already not compile with `g++` 6.3.0). The possible solutions are the use of an operating system environment container or

Table 2.3: Testing of installation on different clean Linux distributions. Green color marks that compilation and installation were successful, red color marks that either of them was unsuccessful, and blue color marks correct compilation and installation that results in errors. The external TMVA 4.2.0 is only installed, when using ROOT 5 and skipped for ROOT 6.

Distribution	ROOT 5	ROOT 6	wxWidgets	ADST	Ext. TMVA	auger-analysis
Debian 8 (jessie)						
Debian 9 (stretch)						
Ubuntu 14.04 (trusty)						
Ubuntu 16.04 (xenial)						
Ubuntu 18.04 (bionic)						
CentOS 7						
Fedora 23						
Fedora 25						
Fedora 27						
Fedora 29						

the installation of an older g++ compiler. The former is simple to set up, because it creates a contained virtual environment for the selected Linux distribution. However, it does need root access for creating the container. The latter can also be installed in a custom directory, but needs care when switching between currently installed versions and can produce some version mismatch with other libraries. As such, the environment container installation is preferred.

2.3.1 Singularity environment container (preferred)

Singularity [5] is an operating system environment container used in many applications, where precise versions of libraries and software is needed over multiple computers. It creates a clean version of a Linux distribution that can be used like any normal installation of Linux. The detailed documentation can be found in [6].
Singularity has the following dependencies that need to be installed on the host system:

```
# on Debian related systems
sudo apt-get install build-essential libssl-dev uuid-dev libgpgme11-dev
    ↪ squashfs-tools libseccomp-dev pkg-config
# on CentOS related systems, and Fedora 21 or below
sudo yum install openssl-devel libuuid-devel squashfs-tools libseccomp-devel
# on Fedora 22 or above related systems
sudo dnf install openssl-devel libuuid-devel squashfs-tools libseccomp-devel
```

The additional packages `debootstrap` and `yum` need to be installed for creating Ubuntu/Debian and CentOS/Fedora containers, respectively. The `debootstrap` package is installed with:

```
# on Debian related systems
sudo apt-get install debootstrap
# on CentOS related systems, and Fedora 21 or below
```

```
sudo yum install debootstrap
# on Fedora 22 or above related systems
sudo dnf install debootstrap
```

The yum package is installed with (not needed for systems based on yum):

```
# on Debian related systems
sudo apt-get install yum
# on Fedora 22 or above related systems
sudo dnf install yum
```

Once the dependencies are installed, the programming language go needs to be installed. Download the Linux archive version from [7] that should appear as goX.YY.linux-amd64.tar.gz, where X.YY is the version. We assume $GOINSTALL is the location of the downloaded source files (for example /opt/golang) and $SINGULARINSTALL is the install location of singularity (for example /opt/singularity). Install singularity with the following commands:

```
# Prepare installation of go
cd $GOINSTALL
tar -zxf goX.YY.linux-amd64.tar.gz
mkdir $GOINSTALL/gopath
# Set environment variables
export GOROOT=${GOINSTALL}/go
export GOPATH=${GOINSTALL}/gopath
export PATH=${GOROOT}/bin:$PATH
# Download singularity and its dependencies
go get -u github.com/golang/dep/cmd/dep
go get -d github.com/sylabs/singularity
cd $GOPATH/src/github.com/sylabs/singularity
git fetch
# Install singularity
./mconfig --prefix=$SINGULARINSTALL
cd builddir
make
sudo make install
cd /usr/bin
sudo ln -s $SINGULARINSTALL/bin/singularity singularity
```

The command sudo make install is important, because without root access, the containers will not work correctly. Once installed, container definitions are prepared in .def files and are used when creating containers. We supply a number of container definition files in Appendix A for clean installations of Linux distributions, where ROOT 5 works correctly.
It is possible to create the container to be read-only or writable:

```
# Create a read-only container
singularity build container.sif container.def
# Create a writable container
sudo singularity build --sandbox container/ container.def
```

Converting a writable container into a read-only container is done with:

```
sudo singularity build container.sif container/
```

Using the container, you can open its shell as read-only or as writable:

```
# Opening a read-only container
singularity shell container.sif
# Opening a writable container
sudo singularity shell --writable container/
```

For more options when using the container, see singularity documentation [6]. The most important options is for mounting disks, which is done with `-B /diskname`.
Important: Even if the container is opened in read-only mode, you are still able to write to any mounted disks and inside the home folder. Think of it as having normal user access without root access.

2.3.2 Installing older compiler versions

An older compiler version can be installed alongside the general system compiler. Choose the compiler version you wish to have and download the other g++ infrastructure software. We assume `$GCCSOURCE` is the location of the downloaded source files (for example `/opt/source`) and `$GCCINSTALL` is the install location (for example `/opt/gcc-X.Y.Z`).
Download and install the GMP infrastructure version X.Y.Z:

```
cd $GCCSOURCE
wget ftp://gcc.gnu.org/pub/gcc/infrastructure/gmp-X.Y.Z.tar.gz
tar -zxf gmp-X.Y.Z.tar.gz
cd $GCCSOURCE/gmp-X.Y.Z
./configure --prefix=$GCCINSTALL/infrastructure/gmp-X.Y.Z
make
make install
export LD_LIBRARY_PATH=$GCCINSTALL/infrastructure/gmp-X.Y.Z/lib:
    ↪ $LD_LIBRARY_PATH
```

Download and install the MPFR infrastructure version X.Y.Z:

```
cd $GCCSOURCE
wget ftp://gcc.gnu.org/pub/gcc/infrastructure/mpfr-X.Y.Z.tar.gz
tar -zxf mpfr-X.Y.Z.tar.gz
cd $GCCSOURCE/mpfr-X.Y.Z
./configure --prefix=$GCCINSTALL/infrastructure/mpfr-X.Y.Z --with-gmp=
    ↪ $GCCINSTALL/infrastructure/gmp-X.Y.Z
make
make install
export LD_LIBRARY_PATH=$GCCINSTALL/infrastructure/mpfr-X.Y.Z/lib:
    ↪ $LD_LIBRARY_PATH
```

Download and install the MPC infrastructure version X.Y.Z:

```
cd $GCCSOURCE
wget ftp://gcc.gnu.org/pub/gcc/infrastructure/mpc-X.Y.Z.tar.gz
tar -zxf mpfr-X.Y.Z.tar.gz
cd $GCCSOURCE/mpc-X.Y.Z
./configure --prefix=$GCCINSTALL/infrastructure/mpc-X.Y.Z --with-gmp=
    ↪ $GCCINSTALL/infrastructure/gmp-X.Y.Z --with-mpfr=$GCCINSTALL/
    ↪ infrastructure/mpfr-X.Y.Z
make
make install
```

```
export LD_LIBRARY_PATH=$GCCINSTALL/infrastructure/mpc-X.Y.Z/lib:
    ↪ $LD_LIBRARY_PATH
```

Then lastly, install the g++ `compiler` version X.Y.Z:

```
cd $GCCSOURCE
wget ftp://gcc.gnu.org/pub/gcc/releases/gcc-X.Y.Z/gcc-X.Y.Z.tar.gz
tar -zxf gcc-X.Y.Z.tar.gz
cd $GCCSOURCE/gcc-X.Y.Z
./configure --prefix=$GCCINSTALL/gcc-X.Y.Z --with-gmp=$GCCINSTALL/
    ↪ infrastructure/gmp-X.Y.Z --with-mpfr=$GCCINSTALL/infrastructure/mpfr-X.Y
    ↪ .Z --with-mpc=$GCCINSTALL/infrastructure/mpc-X.Y.Z
make
make install
```

In order to use this version of the g++ compiler, the following has to be added to the ~/`.bashrc` file or sourced when needed:

```
CC=$GCCINSTALL/gcc-X.Y.Z/bin/gcc
CXX=$GCCINSTALL/gcc-X.Y.Z/bin/g++
PATH=$GCCINSTALL/gcc-X.Y.Z/bin:$PATH
CLIB=$GCCINSTALL/gcc-X.Y.Z/lib64:$GCCINSTALL/infrastructure/gmp-X.Y.Z/lib:
    ↪ $GCCINSTALL/infrastructure/mpfr-X.Y.Z/lib:$GCCINSTALL/infrastructure/mpc
    ↪ -X.Y.Z/lib
if [ -z "$LD_LIBRARY_PATH" ]; then
  DELIM=
else
  DELIM=:
fi
LD_LIBRARY_PATH=${CLIB}${LD_LIBRARY_PATH:+:}$LD_LIBRARY_PATH
export CC CXX PATH LD_LIBRARY_PATH
```

The following command should now return the correct version of the compiler:

```
g++ --version
```

A Singularity container definitions

This appendix gives container definition files for clean installations of different Linux distributions, where ROOT 5 works correctly. We divide them into Debian/Ubuntu and CentOS/Fedora distributions. Typically, a container definition includes the bootstrap option for creating the container, the OS version and the mirror URL, where the distribution is located. The `%post` section runs while creating the container in order to preinstall some packages/software. Preinstalled packages can also be added with the include option. All of the testing installation container definition files are located in the `setup/singularity` folder of the auger-analysis software.

A.1 Debian/Ubuntu distributions

Debian and Ubuntu distributions must always have the `Bootstrap`, `OSVersion` and `MirrorURL` keywords. `Bootstrap` is always set to `debootstrap` and the `MirrorURL` is set to a mirror site, with different distribution versions. You can

find different mirror sites at [8] for Debian and at [9] for Ubuntu. In case the `MirrorURL` in the below examples does not work, choose a different one from the two mirror sites. `OSVersion` defines the distribution version, which can be set to `trusty` (14.04), `xenial` (16.04), `bionic` (18.04),... for Ubuntu, and `wheezy` (7), `jessie` (8), `stretch` (9),... for Debian. The additional script in the `%post` section adds the correct repository and installs locale information in order to support keyboards other than English. Note that the example sets locale support for US and Slovene keyboards.

Container definition file for Ubuntu 14.04 (trusty):

```
BootStrap: debootstrap
OSVersion: trusty
MirrorURL: http://us.archive.ubuntu.com/ubuntu/

%post
  apt-get -y --force-yes install software-properties-common
  add-apt-repository "deb http://us.archive.ubuntu.com/ubuntu/ trusty universe
      ↪ main"
  apt-get update
  apt-get -y --force-yes install locales

  locale-gen en_US.UTF-8
  locale-gen sl_SI.UTF-8
  localedef -i en_US -f UTF-8 en_US.UTF-8
  localedef -i sl_SI -f UTF-8 sl_SI.UTF-8
```

Container definition file for Ubuntu 16.04 (xenial):

```
BootStrap: debootstrap
OSVersion: xenial
MirrorURL: http://us.archive.ubuntu.com/ubuntu/

%post
  apt-get -y --force-yes install software-properties-common
  add-apt-repository "deb http://us.archive.ubuntu.com/ubuntu/ xenial universe
      ↪ main"
  apt-get update
  apt-get -y --force-yes install locales

  locale-gen en_US.UTF-8
  locale-gen sl_SI.UTF-8
  localedef -i en_US -f UTF-8 en_US.UTF-8
  localedef -i sl_SI -f UTF-8 sl_SI.UTF-8
```

Container definition file for Debian 8 (jessie):

```
BootStrap: debootstrap
OSVersion: jessie
MirrorURL: http://deb.debian.org/debian/

%post
  apt-get -y --force-yes install software-properties-common
  add-apt-repository "deb http://deb.debian.org/debian/ jessie main contrib non-
      ↪ free"
  apt-get update
  apt-get -y --force-yes install locales
```

```
  locale-gen en_US.UTF-8
  locale-gen sl_SI.UTF-8
  localedef -i en_US -f UTF-8 en_US.UTF-8
  localedef -i sl_SI -f UTF-8 sl_SI.UTF-8
```

A.2 CentOS/Fedora distributions

CentOS and Fedora distributions must always have the `Bootstrap` and `MirrorURL` keywords. `Bootstrap` is always set to `yum` and the `MirrorURL` is set to a mirror site, with different distribution versions. You can find different mirror sites at [10] for CentOS and at [11] for Fedora. In case the `MirrorURL` in the below examples does not work, choose a different one from the two mirror sites. `OSVersion` defines the optional distribution version, if the variable `%{OSVERSION}` is then used, while setting the `MirrorURL`. The additional script in the `%post` section adds the correct repository.

Container definition file for CentOS 7:

```
BootStrap: yum
OSVersion: 7
MirrorURL: http://mirror.centos.org/centos-%{OSVERSION}/%{OSVERSION}/os/x86_64/
Include: yum

%post
  yum -y update
  yum -y install epel-release
  yum -y repolist
```

Container definition file for Fedora 23:

```
BootStrap: yum
OSVersion: 23
MirrorURL: https://archives.fedoraproject.org/pub/archive/fedora/linux/releases/
    ↪ ${OSVERSION}/Workstation/x86_64/os/
Include: dnf

%post
  dnf -y update
  dnf -y repolist
```