Some experiences in working with Madagascar: installation & development

Tengfei Wang, Peng Zou

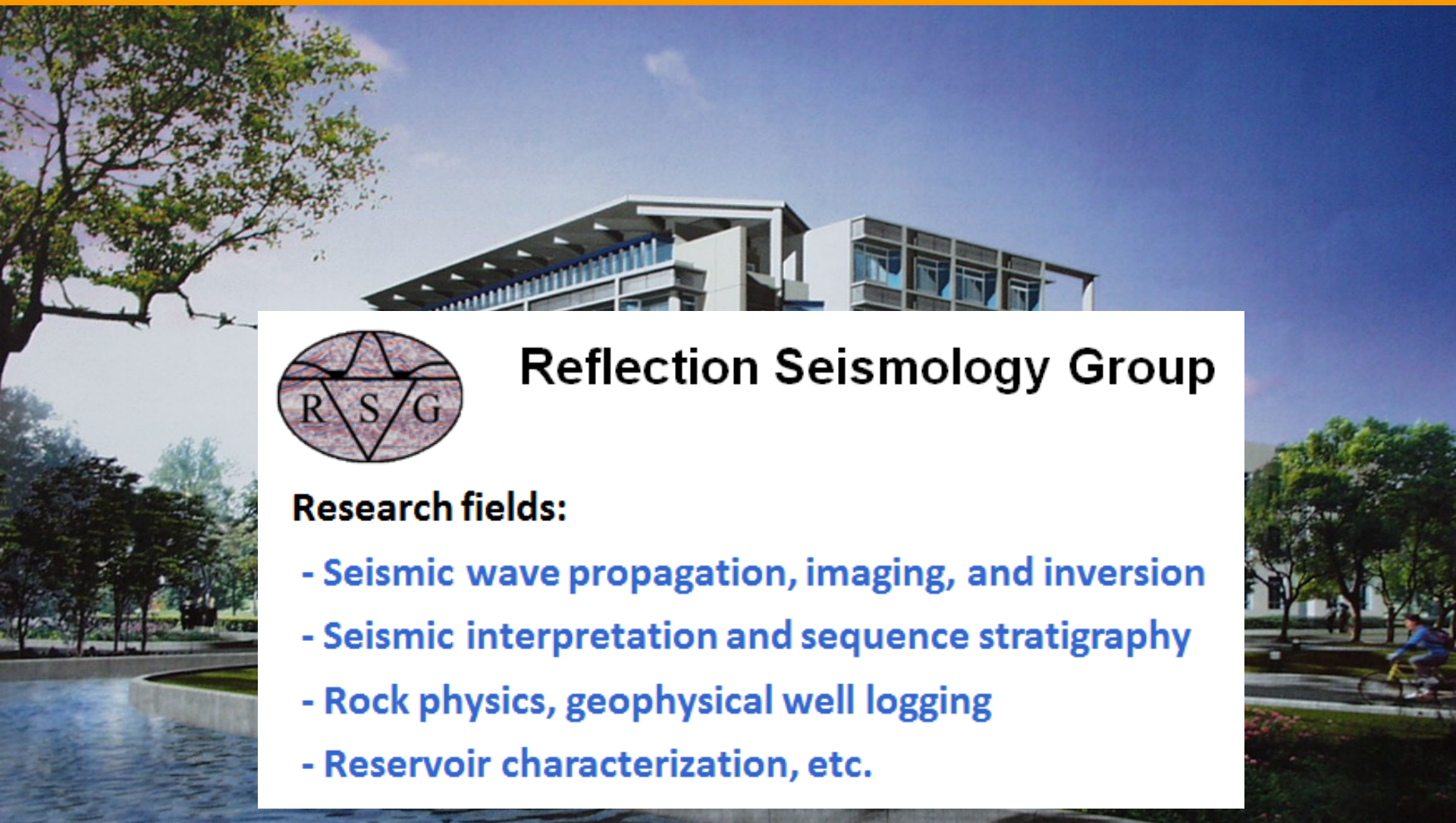
Tongji university

Tongji University, Shanghai, China

Map data @ Google

同济大学海洋与地球科学学院

School of Ocean and Earth Science, Tongji University



Reflection Seismology Group

Research fields:

- Seismic wave propagation, imaging, and inversion
- Seismic interpretation and sequence stratigraphy
- Rock physics, geophysical well logging
- Reservoir characterization, etc.

Outline

- Reproducible research in Madagascar
- How to conduct a successful installation
- Programming & writing
- Some suggestions

Software for Reproducible Research

Madagascar: an open-source software package for multidimensional data analysis and reproducible computational experiments. Learn more at the Madagascar home page.

SEGTeX: a LaTeX package for geophysical publications. Learn more at http://www.reproducibility.org/wiki/SEGTeX .

For more information, please contact **Sergey Fomel**.
Telephone 512-475-9573. E-mail sergey.fomel@beg.utexas.edu.

ICES The Institute for Computational Engineering and Sciences

Main
Staff
Research
Publications
Software

Members' Area

Madagascar

Madagascar: A very powerful open-source software package for multidimensional data analysis and reproducible computational experiments.

It provides:

- ✓ A convenient and powerful environment
- ✓ A convenient technology transfer tool
- ✓ Intellectual property from the contributors
- ✓ A active community

for researchers (including students) in geophysics and related fields.

Why choose Madagascar

- The attached codes contain the up-to-date work.
- Programming in Madagascar is easy (standing on the shoulders of previous authors).
- Writing paper/abstract with LaTeX in Madagascar is amazing, strongly recommended !

Outline

- Reproducible research in Madagascar
- How to conduct a successful installation
- Programming & writing
- Some suggestions

Regular installation

Step 1:

./configure --prefix=/directory/madagascar/installed

Step 2:

Make & make install

Step 3

Set the environment varibles

source $RSFROOT/share/madagascar/etc/env.sh

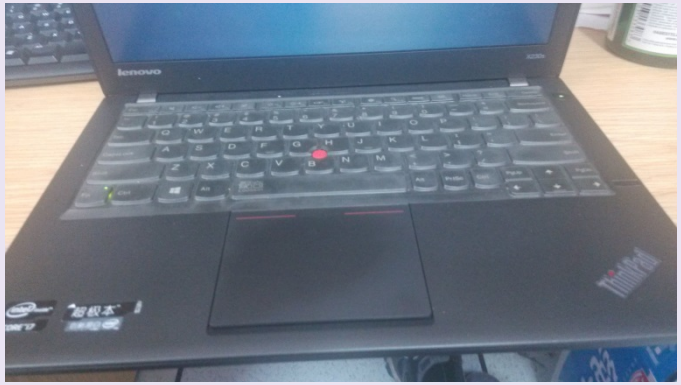
However …

After many times of installation and update, I find it's always not so easy !

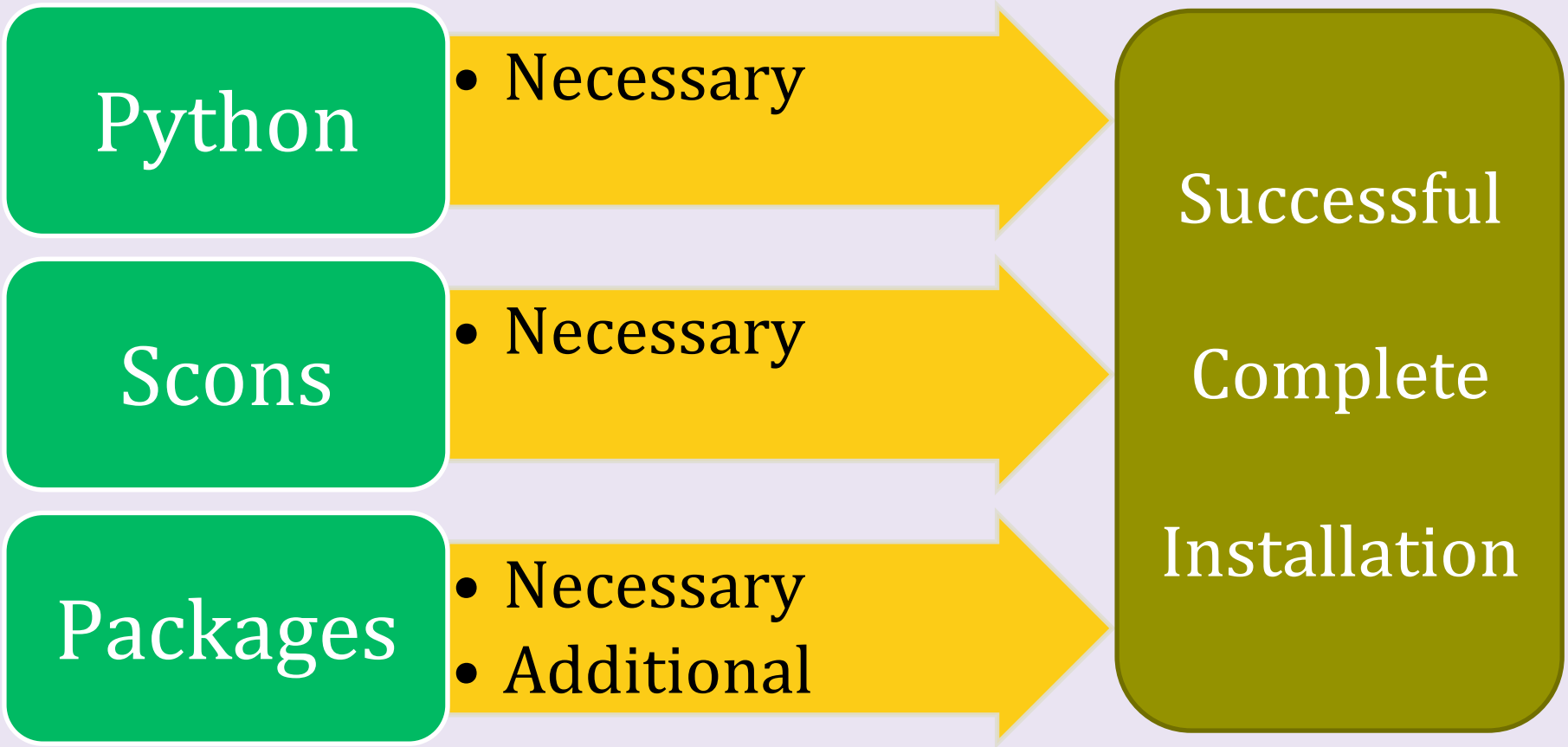
CentOS (7.1)

CentOS (6.5)

redhat (5.5)



Ingredients of a successful installation



Python and Scons

Very difficult to fix if there are problems.
Luckily, these problems are rare.

```
/bin:/home/wtf/bin)
[wtf@node01 madagascar-1.7]$ ./configure
checking for Python ... /usr/bin/python
checking Python version ... 2.6.6
checking for RSFROOT ... no

I am setting RSFROOT to /usr

checking for SCons ... no
error: could not create '/usr/lib/scons-2.3.4': Permission denied
Installing SCons ... failed

Automatic SCons installation failed.

Please install SCons manually!
```

Set RSFROOT in the .bashrc is ok

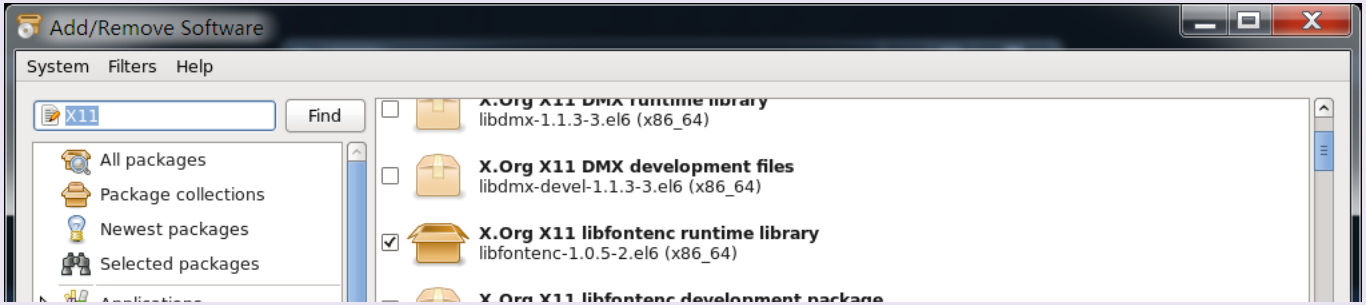
Packages are very important

```
checking complex support ... yes
checking for X11 headers ... /usr/include
checking for X11 libraries ... /usr/lib
checking for OpenGL ... yes
checking for sfpen ... (cached) xtpen
checking for ppm ... no
checking for tiff ... yes
checking for GD (PNG) ... no

  gdpen will not be built.
checking for plplot ... no
checking for ffmpeg ... no
checking for cairo (PNG) ... yes
checking for cairo (SVG) ... yes
checking for cairo (PDF) ... yes
checking for jpeg ... yes
checking for BLAS ... no
checking for LAPACK ... no
checking for SWIG ... (cached) /usr/bin/swig
checking for numpy ... (cached) yes
checking API options ... (cached) []
checking for C++ compiler ... (cached) g++
checking if g++ works ... yes
checking if g++ accepts '-Wall -pedantic' ... yes
```

Ways to install the packages

- If internet is ok, then it is easy.



- Otherwise, cumbersome
  - You can download the *.rpm that is needed, e.g. libX11-1.6.0-2.2.el6.x86_64.rpm
  - The rpm must be suitable for your system version !

How *configure* works

- configure → search for the PATH of a certain package or support and create the config.py
  - config.py

```
File  Edit  View  Search  Terminal  Help
ENV = {'MPICH_CXX': 'g++', 'HOME': '/home/mada', 'PATH': '/usr/local/bin:/opt/bin:/bin:
/usr/bin', 'MPICH_CC': 'gcc', 'LD_LIBRARY_PATH': '', 'MPICH_CLINKER': 'gcc'}
RSFROOT = '/home/mada/RSF'
AR = 'ar'
JPEG = 'jpeg'
OPENGL = None
OPENGLFLAGS = None
```

- Log of configure
  - config.log

```
scons: Configure: checking for BLAS ...
.sconf_temp/conftest_19.c <-
  |
  |    #ifdef __APPLE__
  |    #include <Accelerate/Accelerate.h>
  |    #else
  |    #ifdef HAVE_MKL
  |    #include <mkl.h>
  |    #else
  |    #include <cblas.h>
  |    #endif
  |    #endif
  |    int main(int argc,char* argv[]) {
  |    float d, x[]={1.,2.,3.}, y[]={3.,2.,1.};
  |    d = cblas_sdot(3,x,1,y,1);
  |    return 0;
  |    }
  |
gcc -o .sconf_temp/conftest_19.o -c -O2 -x c -std=gnu99 -Wall -pedantic .sconf_temp/conftest_19.c
gcc -o .sconf_temp/sfconftest_19 .sconf_temp/conftest_19.o -lm
.sconf_temp/conftest_19.o: In function `main':
conftest_19.c:(.text+0x43): undefined reference to `cblas_sdot'
collect2: ld returned 1 exit status
.sconf_temp/conftest_20.c <-
```

How *configure* works

- **framework/configure.py**
  - Python script to check the packages

```
def check_all(context):

    # FDNSI = Failure Does Not Stop
 Installation
    identify_platform(context)
    cc  (context)
    ar  (context)
    libs(context)
    c99 (context) # FDNSI
    x11 (context) # FDNSI
    opengl(context) # FDNSI
    sfpen(context) # FDNSI
    ppm (context) # FDNSI
    tiff (context) # FDNSI
    gd  (context) # FDNSI
    plplot  (context) # FDNSI
    ffmpeg  (context) # FDNSI
    cairo(context) # FDNSI
    jpeg(context) # FDNSI
    mkl(context)  # FDNSI
    blas(context) # FDNSI
    lapack(context) # FDNSI
    swig(context)
                    91,1          4%
```

X11 example

In **framework/configure.py**

```
# The two lists below only used in the x1
1 check
xinc = [
    '/opt/X11/include',
    '/usr/X11/include',
    '/usr/X11R6/include',
    '/usr/X11R5/include',
    '/usr/X11R4/include',
    '/usr/include/X11',
    '/usr/include/X11R6',
    '/usr/include/X11R5',
    '/usr/include/X11R4',
    '/usr/local/X11/include',
    '/usr/local/X11R6/include',
    '/usr/local/X11R5/include',
    '/usr/local/X11R4/include',
    '/usr/local/include/X11',
    '/usr/local/include/X11R6',
                          383,8         16%
```

```
xlib = [
    '/opt/X11/lib',
    '/usr/X11/lib64',
    '/usr/X11/lib',
    '/usr/X11R6/lib64',
    '/usr/X11R6/lib',
    '/usr/X11R5/lib',
    '/usr/X11R4/lib',
    '/usr/lib/X11',
    '/usr/lib/X11R6',
    '/usr/lib/X11R5',
    '/usr/lib/X11R4',
    '/usr/local/X11/lib',
    '/usr/local/X11R6/lib',
    '/usr/local/X11R5/lib',
    '/usr/local/X11R4/lib',
    '/usr/local/lib/X11',
    '/usr/local/lib/X11R6',
    '/usr/local/lib/X11R5',
    '/usr/local/lib/X11R4',
    '/usr/X386/lib',
                          434,26
```

```
def x11(context):
    text = '''
    #include <X11/Intrinsic.h>
    #include <X11/Xaw/Label.h>
    int main(int argc,char* argv[]) {
    return 0;
    }\n'''

    context.Message("checking for X11 headers
```

```
    context.Message("checking for X
11 libraries ... ")

    LIB = path_get(context,'XLIBPAT
H')
    oldlibpath = path_get(context,'
LIBPATH')
    oldlibs = path_get(context,'LIB
S')
    XLIBS = path_get(context,'XLIBS
')
```

In **config.log**

```
scons: Configure: checking for X11
headers ...
.sconf_temp/conftest_4.c <-
  |
  |    #include <X11/Intrinsic.h>
  |    #include <X11/Xaw/Label.h>
  |    int main(int argc,char* argv
[]) {
  |    return 0;
  |    }
  |
gcc -o .sconf_temp/conftest_4.o -c
-O2 -x c -std=gnu99 -Wall -pedantic
 -I/usr/include .sconf_temp/conftes
t_4.c
scons: Configure: /usr/include
```

In **config.py**

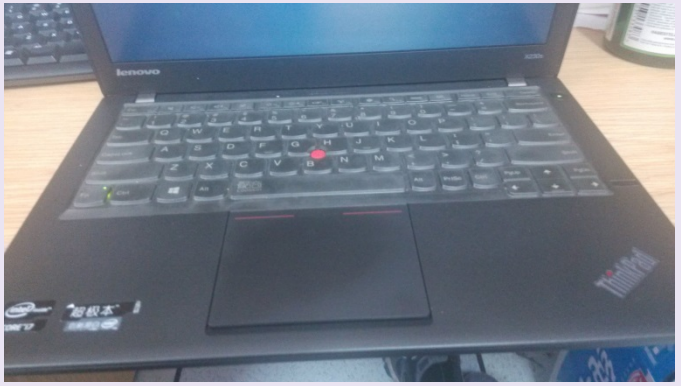
```
XLIBPATH = ['/usr/lib']
XLIBS = ['Xaw', 'Xt', 'X11']
XINC = ['/usr/include']
```

An attempt recently

Latest released CentOS version

CentOS (7.1)

Everything is ok, except the BLAS and LAPACK.
And they are important math libraries for me.



```
  gdpen will not be built.
checking for plplot ... no
checking for ffmpeg ... no
checking for cairo (PNG) ... yes
checking for cairo (SVG) ... yes
checking for cairo (PDF) ... yes
checking for jpeg ... yes
checking for BLAS ... no
checking for LAPACK ... no
```

We use the ATLAS package for the BLAS and LAPACK.

Check the installed package

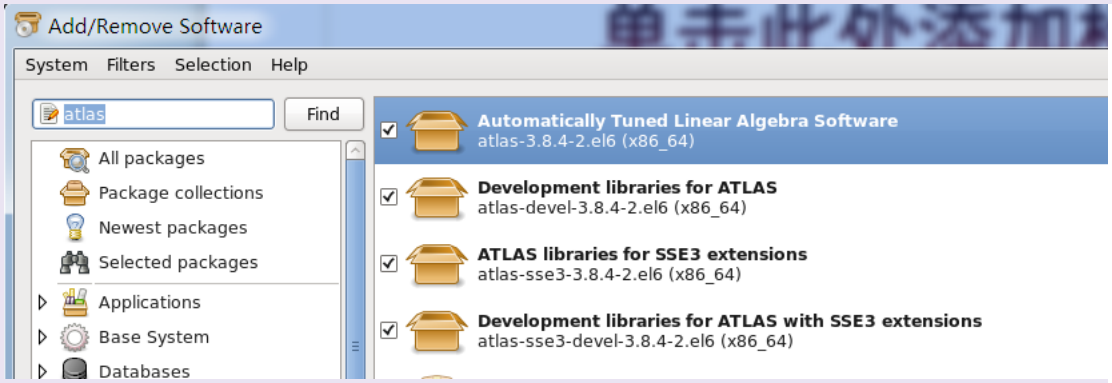
In **config.log**

```
gcc -o .sconf_temp/sfconftest_20 .sconf_temp/conftest_20.o -L/usr/lib64/atlas -l
m -lf77blas -lcblas -latlas
/bin/ld: cannot find -lf77blas
/bin/ld: cannot find -lcblas
collect2: error: ld returned 1 exit status
scons: Configure: no
```

For blas

```
gcc -o .sconf_temp/sfconftest_23 .sconf_temp/conftest_23.o -L/usr/lib64/atlas -l
m -lf77blas -lcblas -latlas
/bin/ld: cannot find -lf77blas
/bin/ld: cannot find -lcblas
collect2: error: ld returned 1 exit status
scons: Configure: no
```

For lapack



Files

- /etc/ld.so.conf.d/atlas-x86_64.conf
- /usr/lib64/atlas/
- /usr/lib64/atlas/libsatlas.so.3
- /usr/lib64/atlas/libsatlas.so.3.10
- /usr/lib64/atlas/libtatlas.so.3
- /usr/lib64/atlas/libtatlas.so.3.10
- /usr/share/doc/atlas-3.10.1/
- /usr/share/doc/atlas-3.10.1/README.dist

In this new version of ATLAS package, there are no libf77blas and libcblas at all !! And the name for libatlas is not the same as previous one !!

Find the bug and fix it !

The original version

```
else:
    # some systems require cblas and atlas
    for atlas_dir in filter(os.path.isdir,
                            ['/usr/lib64/atlas/',
                             '/usr/lib/atlas/']):
        context.env['LIBPATH'].append(atlas_dir)
    LIBS.pop()
    LIBS.append('f77blas')
    LIBS.append('cblas')
    LIBS.append('atlas')
    res = context.TryLink(text,'.c')
    if res:
```

```
else:
    LIBS.pop()
    LIBS.pop()
    mylibs = ['f77blas','cblas','atlas']
    LIBS.extend(mylibs)
    res = context.TryLink(text,'.c')
    if res:
        context.Result(res)
        context.env['LAPACK'] = mylibs
    else:
        context.Result(context_failure)
        context.env['LAPACK'] = None
        need_pkg('lapack', fatal=False)
        LIBS.pop()
        LIBS.pop()
        LIBS.pop()
```

The fixed version

```
else:
    # some systems require cblas and atlas
    for atlas_dir in filter(os.path.isdir,
                            ['/usr/lib64/atlas/',
                             '/usr/lib/atlas/']):
        context.env['LIBPATH'].append(atlas_dir)
    LIBS.pop()
    LIBS.append('f77blas')
    LIBS.append('cblas')
    LIBS.append('satlas')
    res = context.TryLink(text,'.c')
```

```
else:
    # some systems require cblas and atlas
    LIBS.pop()
    LIBS.pop()
    mylibs = ['f77blas','cblas','atlas']
    mylibs = ['satlas']
    LIBS.extend(mylibs)
    res = context.TryLink(text,'.c')
    if res:
        context.Result(res)
        context.env['LAPACK'] = mylibs
    else:
```

Outline

- Reproducible research in Madagascar
- How to conduct a successful installation
- Programming & writing
- Some suggestions

Programming in C or C++

- No especially restricts
- Start with the codes in the directory of ***user/*** and choose a simple one.
- Tip: add comments with /*<…>*/ in your subfunctions, otherwise it will fail when compiling.

Attention: please respect others' work if you want use them. Add proper citation and acknowledgement !

When using new libraries

- Put the libraries in the default include path and *export C_INCLUDE_PATH*

- Compiling & linking
  - Add the libraries into the SConstruct

Tip: when testing your program, you don't need make install. Just add the full-path before the executable file in the workflow, e.g. `/home/mada/RSFSRC/user/wangtf/sfvti2delr`

When need to change flags

- My way: modify the config.py directly, and compile only in your own source code directory

```
CFLAGS = '-O2 -x c -std=gnu99 -Wall -pedantic -
fopenmp'
CPPPATH = []
LIBPATH = ['/usr/lib64/atlas/']
LIBS = ['m', 'f77blas', 'cblas', 'atlas',
'lapack', 'cblas', 'gomp', 'fftw3f']
LINKFLAGS = ['-pthread', '-fopenmp']
XLIBPATH = ['/usr/lib']
XLIBS = ['Xaw', 'Xt', 'X11']
XINC = ['/usr/include']
API = ['c++']
CXX = 'g++'
CXXFLAGS = '-O2 -Wall -pedantic -fopenmp'
```

- Caution: do not make & make install in the RSFSRC directory after you modify the config.py

Awesome typesetting tool -- LaTeX

- Tailor-made tool for researchers
- More efficient than Word if there is template

Outline

- Reproducible research in Madagascar
- How to conduct a successful installation
- Programming & writing
- Some suggestions

Some suggestions ?

- It is challenging to let the information which are originally shown on the screen to save in a log file
  - ✓ Installing : make (error info)
  - ✓ Programming: some intermediate output for check
  - ✓ Run a program in the background
- How to debug the code easily

Some suggestions ?

- About the manual of this software
  - Thanks to the developers, there are many useful manuals
  - But a single and integrated manual may be more helpful, so that we can learn to solve some problems according to the outline of the manual
- About the templates for publication
  - Not easy to use the templates for “Geophysical Journal International”
  - More templates, e.g., “Geophysical Prospecting”

Thanks for your attention !