# Higher-Performance R via C++

## Part 5: Rcpp Packaging and Debugging

Dirk Eddelbuettel

UZH/ETH Zürich R Courses

June 24-25, 2015

# Overview

# Overview: R Packaging

# R Packages

- This is an important topic in R programming
- Organising code in packages maybe *the* single most helpful step
- Core topic in R Programming / Advanced R courses
- Penn 2014 workshop had 90 minutes on this

# R Packages

- `package.skeleton()` helpful as it creates a stanza
- `package.skeleton()` not helpful as it creates a stanza that does not pass `R CMD check` cleanly
- I wrote pkgKitten to provide `kitten()` which creates *packages that purr*
- Rcpp (and RcppArmadillo, RcppEigen) all have their own versions of `package.skeleton()`
- They can use `kitten()` if pkgKitten is installed
- Alternative: `devtools::create()` if you don't mind Hadleyverse dependency
- Also: RStudio File -> New Project -> New Directory -> R Package; and toggle 'R Package' to 'R Package w/ Rcpp'

# Case Studies

# Case Studies: RcppAnnoy

# RcppAnnoy

- Uses only one C++ header (one plus header for Windows)

```
edd@don:~/git/rcppannoy$ tree inst/include/
inst/include/
   annoylib.h
   mman.h

0 directories, 2 files
edd@don:~/git/rcppannoy$ tree src/
src/
   annoy.cpp
   Makevars

0 directories, 2 files
edd@don:~/git/rcppannoy$
```

# RcppAnnoy: `src/Makevars`

- One include indirection to the header file

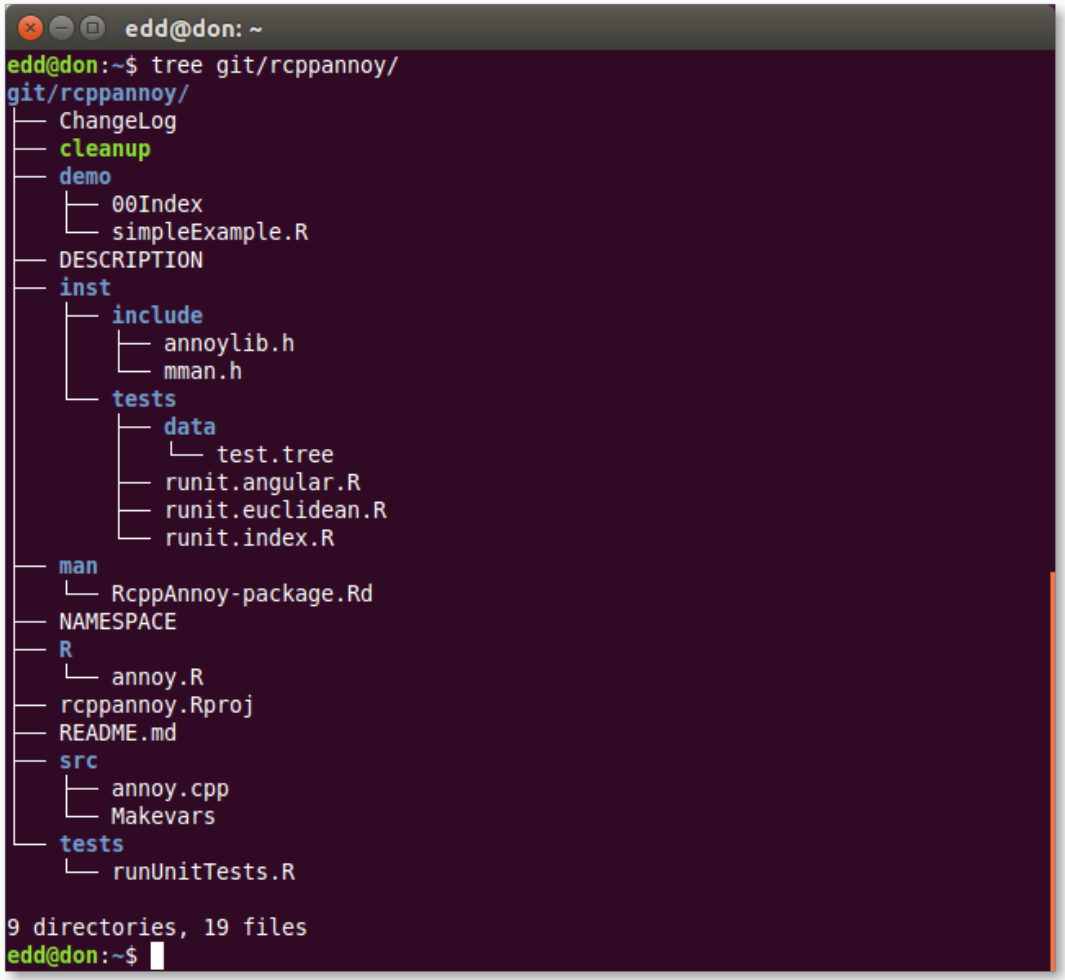
```
## We want C++11 as it gets us 'long long' as well
CXX_STD = CXX11

PKG_CPPFLAGS = -I../inst/include/
```

# RcppAnnoy: `src/annoy.cpp`

- Implemented as Rcpp Modules (not discussed today)
- Wraps around templated C++ class for either
  - Angular distance, or
  - Euclidian distance
- Package interesting as upstream C++ core used with Python by upstream

# RcppAnnoy




Plus a few additional files for tests and documentation.

# Case Studies: RcppCNPy

# RcppCNPy

- Uses one C++ header and one C++ source file from CNPy

```
edd@don:~/git/rcppcnpy$ tree src/
src/
   cnpy.cpp            # from CNPy
   cnpy.h              # from CNPy
   cnpyMod.cpp         # our wrapper
   Makevars            # add -lz (from R) and C++11
   Makevars.win        # ditto

0 directories, 5 files
edd@don:~/git/rcppcnpy$
```

# RcppCNPy

- For this package no other customization is needed
- Simply add the two source files
- Code integration done via Rcpp Modules (which we won't cover today)
- Here we just need one linker flag (supplied by R)

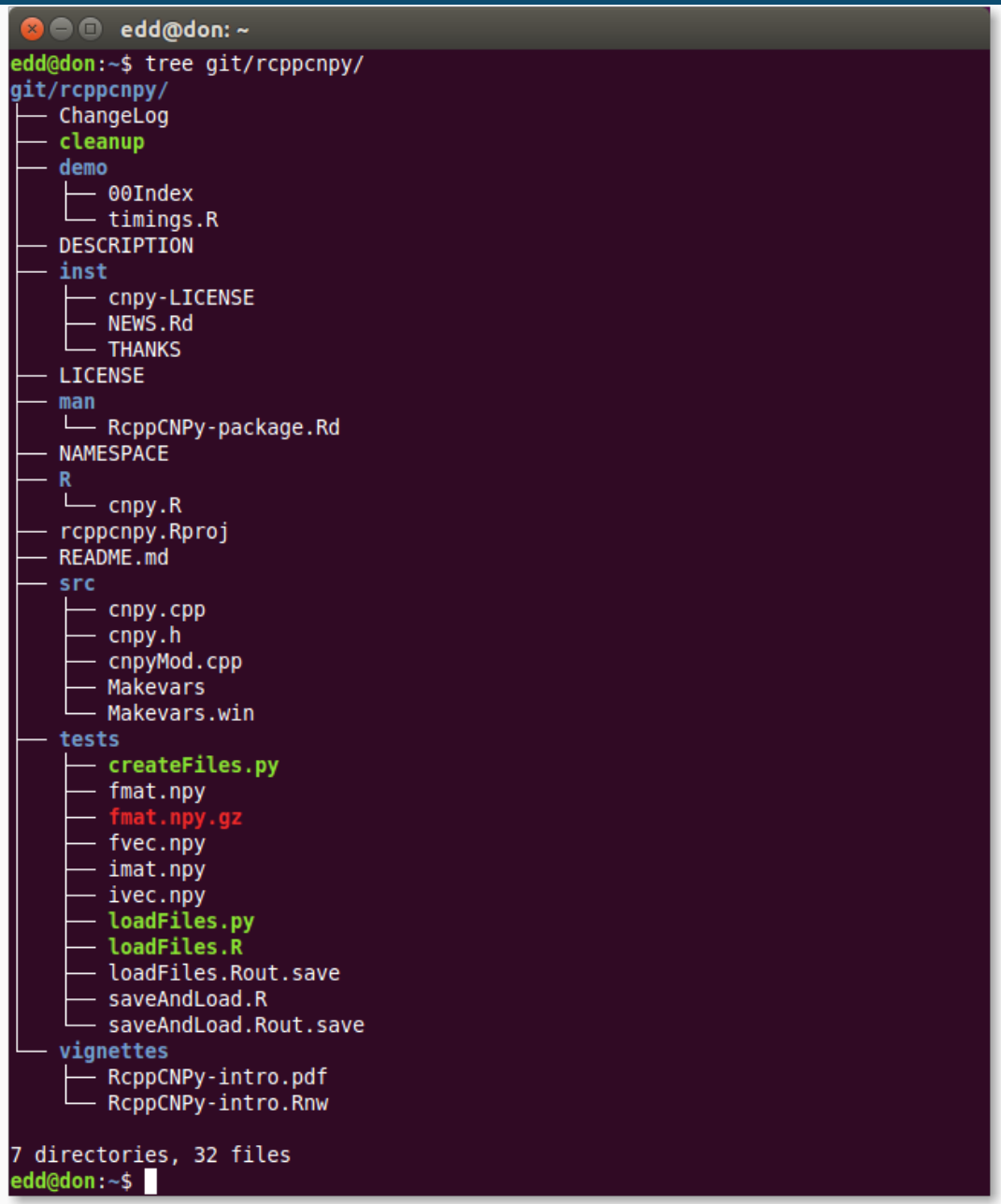
# RcppCNPy: `src/Makevars`

One linker flag (and a compiler option for `long long`)

```
## We need the compression library
PKG_LIBS = -lz

## We want C++11 as it gets us 'long long' as well
CXX_STD = CXX11
```

# RcppCNPy




More test files, more documentation files make this look more "busy" – but still a simple package.

# Case Studies: RcppAPT

# RcppAPT

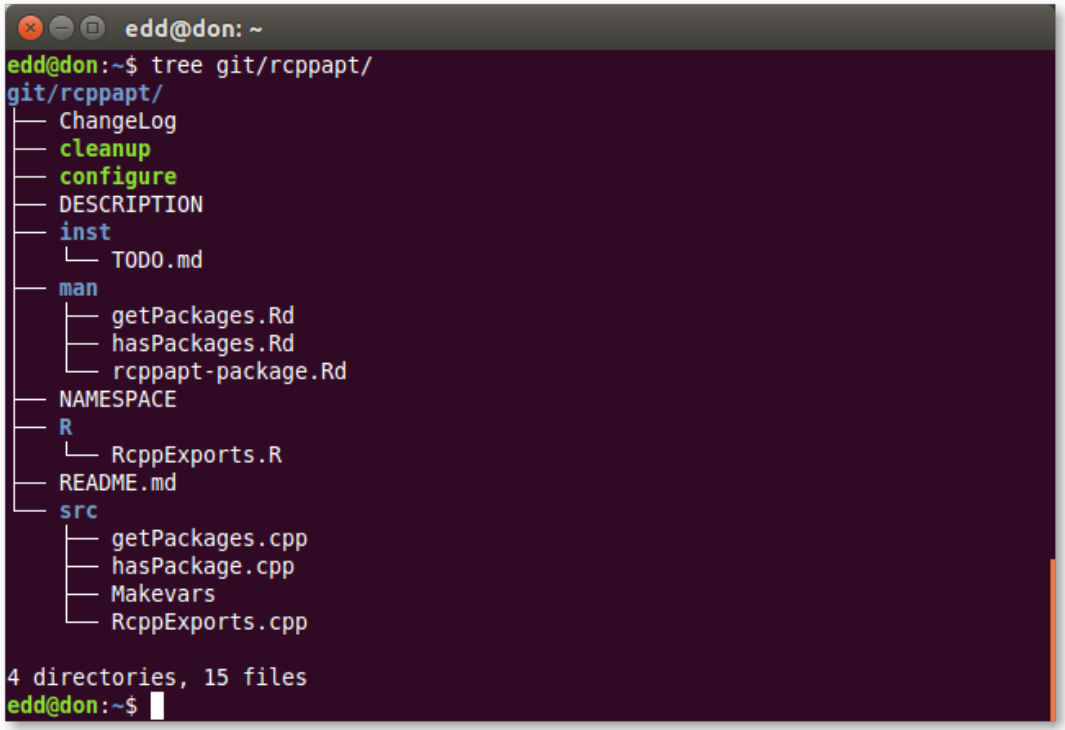
- A somewhat experimental package which only builds on Ubuntu or Debian
- Interface a system library we can assume to be present on those systems – but not on OS X, Windows or even other Linux systems

# RcppAPT: `src/Makevars`

- Very simple

```
PKG_LIBS = -lapt-pkg
```

# RcppAPT




Very simple: two functions wrapping code from system library.

# Case Studies: RcppFaddeeva

# RcppFaddeeva

- Very recent package by Baptiste Auguie with some help from me
- Wrapper around some complex-valued error functions by Steven Johnson
- Upstream ships a single header and a single C++ file –> just place in `src/`
- Usage pretty easy: loop over elements of argument vector and call respective function to build return vector

# RcppFaddeeva

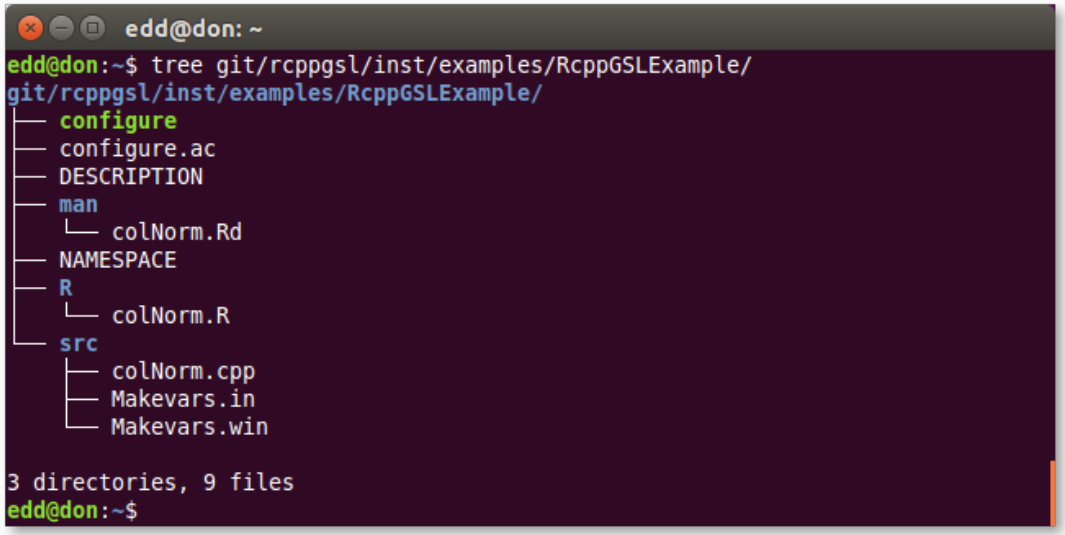
```
//' @title Faddeeva family of error functions of the complex variable
//' @description the Faddeeva function
//' @param z complex vector
//' @param relerr double, requested error
//' @return complex vector
//' @describeIn wrap compute w(z) = exp(-z^2) erfc(-iz)
//' @family wrapper
//' @examples
//' Faddeeva_w(1:10 + 1i)
//' @export
// [[Rcpp::export]]
std::vector< std::complex<double> > Faddeeva_w(const std::vector< std::complex<
 relerr=0) {
  int N = z.size();
  std::vector< std::complex<double> > result(N);
  for(int i=0; i<N; i++) {
    result[i] = Faddeeva::w(z[i], relerr);
  }
  return result;
}
```

# Case Studies: RcppGSLExample

# RcppGSLExample

- This package is included in the RcppGSL package and part of the test environment
- It implements the same column norm example we looked at earlier.

# RcppGSLExample




Simple package against library which we test for (`configure`) and set environment variable for (`src/Makevars.win`)

# R-only to R and Rcpp

- No full example here
- Easy to do manually:
  - Add `LinkingTo: Rcpp` to `DESCRIPTION`
  - Also add `Imports: Rcpp` to `DESCRIPTION`
  - Add `importFrom(Rcpp, "evalCpp")` to `NAMESPACE`
- Add some C++ code in `src/`
- Remember to run `compileAttributes()` each time you add (or change!) a C++ intrface

# Debugging

# Debugging: Using gdb

# Using gdb

- Debugging is unfortunately platform-specific
- When the compiler is `g++`, the debugger is `gdb`
- When the compiler is `clang++`, the debugger is `lldb`.
- I use `g++` more often (under Linux) so we'll focus on `gdb`. However, `lldb` is very similar.

# Using gdb

```
#include <Rcpp.h>

// [[Rcpp::export]]
bool divbyzero(int x) {
    int res = x / 0L;
    Rcpp::Rcout << "res is now " << res << std::endl;
    return true;
}
```

# Using gdb

```
R> sourceCpp("debugEx.cpp")
debugEx.cpp: In function ‘bool divbyzero(int)’:
debugEx.cpp:6:17: warning: division by zero [-Wdiv-by-zero]
     int res = x / 0L;
                 ^
R> divbyzero(10L)
res is now
Process R floating point exception (core dumped) at Wed Jun 24 20:19:12 2015
```

# Using gdb

Start R with `-d gdb` switch, then type `run` to launch R.

```
R> Rcpp::sourceCpp("debugEx.cpp")
[Thread 0xb5cecb40 (LWP 25544) exited]
[Thread 0xb64edb40 (LWP 25543) exited]
[Thread 0xb44ebb40 (LWP 25545) exited]
debugEx.cpp: In function ‘bool divbyzero(int)’:
debugEx.cpp:6:17: warning: division by zero [-Wdiv-by-zero]
     int res = x / 0L;
                 ^

R> divbyzero(10L)
res is now
Program received signal SIGFPE, Arithmetic exception.
0xb0b94a5f in divbyzero (x=10) at debugEx.cpp:6
6       int res = x / 0L;
(gdb)
```

Now at line of floating point exception.

# Using gdb

- Worth learning more about gdb
- Some tutorials:
  - SO post of mine
  - Similar SO post for OS X
  - Seth Falcon (of R Core) video
  - BioConductor HOWTO on C debugging

# Debugging: ASAN Errors

# ASAN Errors

- CRAN is now using recent g++ / clang++ features for
  - ASAN ("Address Sanitizer")
  - UBSAN ("Undefined Behaviour Sanitizer")
- These allow us to "instrument" R with compiler-dependent run-time diagnostics
- Problem: Needs R sources, recent compilers, knowledge of building R from source
- Solution: Docker! (but we'd need more time than we have today to properly introduce Docker)
- sanitizers package triggers 'true positives' validating toolchain setups so that errors can be replicated & fixed.
- For more, see eg my sanitizers page and my worked UBSAN example

# ASAN Errors

```
#include <R.h>
#include <Rdefines.h>

extern "C" {
    // https://code.google.com/p/address-sanitizer/wiki/ExampleHeapOutOfBounds
    SEXP heapAddressSanitize(SEXP xs) {
        int *array = new int[100];
        int x, y;
        SEXP res;
        int *pres;
        array[0] = 0;
        x = INTEGER_VALUE(xs);
        y = array[x + 100];          // BOOM
        delete [] array;
        PROTECT(res = NEW_INTEGER(1));  // Allocating storage space
        pres = INTEGER_POINTER(res);    // pointer to SEXP object
        pres[0] = y;
        UNPROTECT(1);
        return res;
    }
}
```

# ASAN Errors

```
edd@max:~/git$ docker run --rm -ti -v $(pwd):/mnt  rocker/r-devel-san RD CMD check /mnt/sanitizers_0.1.0.1
* using log directory '//sanitizers.Rcheck'
* using R Under development (unstable) (2015-06-17 r68530)
[...]
* checking tests ...
  Running 'testHeapAddressSanitize.R'
 ERROR
Running the tests in 'tests/testHeapAddressSanitize.R' failed.
Last 13 lines of output:
    Freed heap region:       fd
    Stack left redzone:      f1
    Stack mid redzone:       f2
    Stack right redzone:     f3
    Stack partial redzone:   f4
    Stack after return:      f5
    Stack use after scope:   f8
    Global redzone:          f9
    Global init order:       f6
    Poisoned by user:        f7
    Contiguous container OOB:fc
    ASan internal:           fe
  ==266==ABORTING
* checking PDF version of manual ... OK
* DONE

Status: 1 ERROR
See
  '//sanitizers.Rcheck/00check.log'
for details.
```

# Debugging: UBSAN Errors

# UBSAN Errors

- For UBSAN we use a different Docker image
- It includes a wrapper script `check.r` which makes deployment very easy.

# UBSAN Errors

```
#include <R.h>
#include <Rdefines.h>

extern "C" {
    // with thanks to Greg Jefferis (https://github.com/eddelbuettel/docker-debi
    // call with a sufficiently large x such as 31
    SEXP intOverflow(SEXP xs) {
        int x, y;
        SEXP res;
        int *pres;

        x = INTEGER_VALUE(xs);
        y = (1 << x)  - 1;          // BOOM -- (signed) int overflow

        PROTECT(res = NEW_INTEGER(1));  // Allocating storage space
        pres = INTEGER_POINTER(res);    // pointer to SEXP object
        pres[0] = y;
        UNPROTECT(1);
        return res;
    }
}
```

# UBSAN Errors

```
edd@max:~/git$ docker run --rm -ti -v $(pwd):/mnt  rocker/r-devel-ubsan-clang check.r -s /mnt sanitizers_0
* using log directory ‘/mnt/sanitizers.Rcheck’
[...]
* checking for unstated dependencies in ‘tests’ ... OK
* checking tests ...
  Running ‘testHeapAddressSanitize.R’
  Running ‘testIntOverflowSanitize.R’
 ERROR
Running the tests in ‘tests/testIntOverflowSanitize.R’ failed.
Last 13 lines of output:

  R is a collaborative project with many contributors.
  Type 'contributors()' for more information and
  'citation()' on how to cite R or R packages in publications.

  Type 'demo()' for some demos, 'help()' for on-line help, or
  'help.start()' for an HTML browser interface to help.
  Type 'q()' to quit R.

  >
  > library(sanitizers)
  > intOverflowSanitize(31)
  int_overflow.cpp:17:23: runtime error: signed integer overflow: -2147483648 - 1 cannot be represented in
* checking PDF version of manual ... OK
* DONE

Status: 1 ERROR
See
  ‘/mnt/sanitizers.Rcheck/00check.log’
for details.
```