# Guide to the **SEXPtools** Package

Drew Schmidt

January 14, 2014

## Contents

## Acknowledgement

We thank Christian Heckendorf and Wei-Chen Chen for helping with the package's configuration issues, especially in regard to the building of the static library, as well as in helping with portability of the package.

# 1 Introduction

## 1.1 Purpose

This package is intended to serve somewhat the same purpose as the very (deservedly!) popular package **Rcpp**.

## 1.2 License

The **SEXPtools** package is licensed under the very permissive 2-clause BSD license, commonly referred to as the FreeBSD license.

## 1.3 Installation

# 2 Package Use

# 3 Q&A

This section is a set of frequently asked questions (FAQ), with frequency uniformly equal to zero.

## 3.1 Why make this?

Probably my biggest motivator was fun; I just wanted to make something like this. Another, more pragmatic reason is that part of my workload (for very non-standard reasons not worth getting into) prevents me from using **Rcpp**. This leaves me stuck with the native C interface for R. And I don't like R's native C interface. This is my attempt to make that interface (slightly) more friendly.

## 3.2 Why the strange name?

Every R object (underneath, in the C interface) is an `SEXP` (short for S-expression) object, which is a struct pointer. This is explained in the R Internals manual. This package is a collection of tools for more easily managing SEXP objects.

## 3.3 How does this differ from Rcpp?

Each of these packages makes an attempt at solving a serious problem with utilizing compiled code from R: the native interface for C code in R sucks. There are huge differences between the two packages, however. In short, **Rcpp** is *much* a much more comprehensive solution. If you are new to using compiled code with R, frankly this package probably is not for you; you would likely be much better served by **Rcpp**. However, if for some combination of reasons you either cannot or prefer to not use **Rcpp**, then this package may be of interest to you.

Beyond the scope and ease of use of each project (where **Rcpp** handily wins), there are some other critical differences between the projects. A few of note are:

1. **SEXPtools** is more permissively licensed than **Rcpp** (BSD rather than GPL)
2. **SEXPtools** is pure C while **Rcpp** is C++.

These things may not matter in the least to you. If that's the case, then you may well be better served by **Rcpp**.

## 3.4 Why would I want to use it?

You may well not. But it is an option available to you.

## 3.5 How would I use SEXPtools in a package?