QuantLib, R and Rcpp

Dr. Dirk Eddelbuettel
dirk.eddelbuettel@R-Project.org
edd@debian.org
@eddelbuettel

QuantLib User Meeting 2014
IKB Deutsche Industriebank AG
Düsseldorf, Germany
4 December 2014

Outline

1. QuantLib

QuantLib: *The* Open Source Quant Library

Accomplishments well know to all of us

In a nutshell:

- Fifteen years of blood, sweat and tears
- 700k lines of code, examples and unit tests[1]
- Ten of thousands of (svn and now git) commits
- Hundreds of modules, pricers, classes, functions ...
- Mostly Fernando & Luigi, plus a small core team[2]

[1] Well, Ohloh says so...

[2] My unscientific guess..

Outline

2 R

R: “Programming with Data”

In another nutshell:

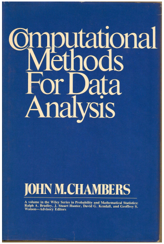
- *A language and an environment* [3]
- *Has forever altered the way people analyze, visualize and manipulate data* [4]
- *A vibrant community and ecosystem*: CRAN + BioConductor provide > 6k packages that “just work”
- Reliably cross-platform + cross-operating system
- *The lingua franca of (applied) statistical research*

[3] R FAQ, Question 2.1

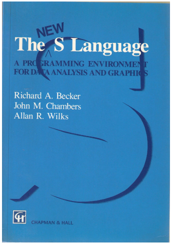
[4] 1999 ACM citation for John Chambers

R: History by the Books

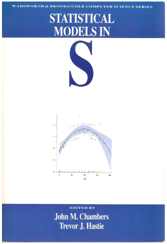
John Chambers, with a few co-authors



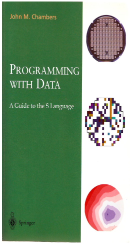
Chambers, *Computational Methods for Data Analysis*. Wiley, 1977.



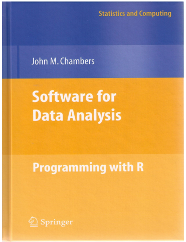
Becker, Chambers, and Wilks. *The New S Language*. Chapman & Hall, 1988.



Chambers and Hastie. *Statistical Models in S*. Chapman & Hall, 1992.



Chambers. *Programming with Data*. Springer, 1998.



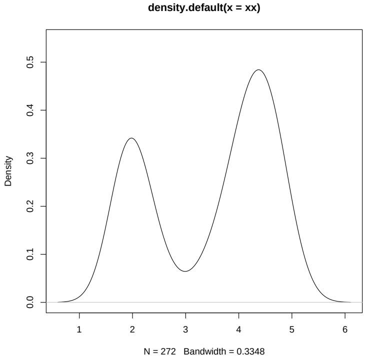
Chambers. *Software for Data Analysis: Programming with R*. Springer, 2008

Thanks to John Chambers for sending me high-resolution scans of the covers of his books.

What makes R so useful?

Succinct and expressive

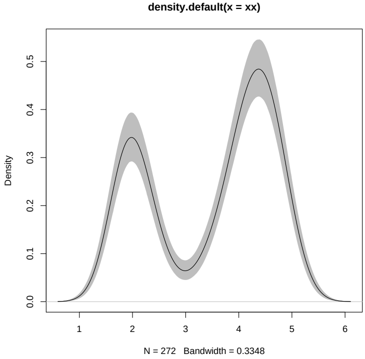
```
> xx <- faithful[,"eruptions"]
> fit <- density(xx)
> plot(fit)
```



What makes R so useful?

Succinct and expressive

```
> xx <- faithful[,"eruptions"]
> fit1 <- density(xx)
> fit2 <- replicate(10000, {
+
x <- sample(xx,replace=TRUE);
+
density(x, from=min(fit1$x),
+           to=max(fit1$x))$y
+ })
> fit3 <- apply(fit2, 1,
+   quantile,c(0.025,0.975))
> plot(fit1, ylim=range(fit3))
>
polygon(c(fit1$x,rev(fit1$x)),
+   c(fit3[1,], rev(fit3[2,])),
+   col='grey', border=F)
> lines(fit1)
```



The example was posted by Greg Snow on r-help a few years ago.

What makes R so useful?

Interactive

R enables us to

- work interactively
- explore and visualize data
- access, retrieve and/or generate data
- summarize and report into pdf, html, ...
- *dynamic documents* in markdown, Shiny dashboards and more

making it a preferred environment for many data analysts.

What makes R so useful?

Extensible

R has always been extensible via

C	via a bare-bones interface described in *Writing R Extensions*
Fortran	which is also used internally by R
Java	via **rJava** by S Urbanek
C++	but essentially at the bare-bones level of C

So 'in theory' this always worked – yet tedious 'in practice'.

Outline

3 RQuantLib

Obvious Idea: Take R, add QuantLib

Best of both worlds:

- Obvious appeal: R rocks for *Programming with Data*
- Obvious appeal: QuantLib rocks for Finance (pricing, risk management, ...)
- So see if we can join them together
- The first steps were very manual.[5]

[5] The very impressive parallel work by Joe Wang on the QL R / SWIG bindings is of course also acknowledged.

RQuantLib 0.1.0 as of 25 Feb 2002

Pre-Rcpp with heavy dose of help from Doug Bates

```
// simple helper function to insert "labelled" element into list
static inline
void insertListElement(SEXP &list, SEXP &names,
                       const int pos, const double value,
                       const char *label) {
  SEXP vec = PROTECT(allocVector(REALSXP, 1));
  REAL(vec)[0] = value;
  SET_VECTOR_ELT(list, pos, vec);
  SET_STRING_ELT(names, pos, mkChar(label));
  UNPROTECT(1);

}
```

RQuantLib 0.1.0 as of 25 Feb 2002

Pre-Rcpp with heavy dose of help from Doug Bates

```
// get the list element named str, or return NULL
// courtesy of the R Exts manual, and the nls package
static inline SEXP getListElement(SEXP list, char *str) {
  SEXP elmt = R_NilValue,
       names = getAttrib(list, R_NamesSymbol);
  int i;

  for (i = 0; i < length(list); i++)
    if(strcmp(CHAR(STRING_ELT(names, i)), str) == 0) {
      elmt = VECTOR_ELT(list, i);
      break;
    }
  return elmt;
}
```

Underlying C API for R somewhat bare

Everything evolves around `.Call` and `SEXP`

At the C++ level:

```
SEXP foo(SEXP a, SEXP b, SEXP C, ...)
```

and at the R level:

```
> res <- .Call("foo", a, b, c, ...,
+                PACKAGE="mypkg")
```

From RQuantLib (RQL) to Rcpp and back

- RQL 0.1.13 (2002 - Aug 2005): Two macros
- RQL 0.2.0 (Oct 2005): First minimal Rcpp (by Dominick): very different from what we use today
- RQL 0.2.9 (Aug 2008): Last with embedded old Rcpp
- Rcpp 0.6.0 (Nov 2008): New start following the Rcpp/RcppTemplate withdrawal
- RQL 0.2.10 (Dec 2008): Uses external Rcpp
- RQL 0.3.0 (Sep 2009): With Khanh's GSoC contributions
- Rcpp 0.7.0 (Dec 2009) Romain joined, leading to rapid Rcpp changes over next few years
- Rcpp 0.10.0 (Dec 2012): JJ added Attributes
- Rcpp 0.11.0 (Feb 2013): Easier build, no more linking
- RQL 0.4.0 (Dec 2014): Now with proper use of modern Rcpp

Fast-forward to today

Invoking (some) QuantLib functions for use from R can be as simple as this:

```
> s <- "QuantLib::Date calDemo(QuantLib::Date d,int dt) {
    return QuantLib::Argentina().advance(d,dt,QuantLib::Days);}"
> Rcpp::cppFunction(s, depends="RQuantLib")
> calDemo(Sys.Date(), 3)

## [1] "2014-12-03"
```

Fast-forward to today

Or written in a short C++ file ...

```
#include <RQuantLib.h>

// [[Rcpp::depends(RQuantLib)]]

// [[Rcpp::export]]
QuantLib::Date calDemo(QuantLib::Date day, int delta) {
    // or any other calendar
    QuantLib::Calendar cal = QuantLib::Argentina();
    QuantLib::Date newDate =
        cal.advance(day, delta, QuantLib::Days);
    return newDate;
}
```

Fast-forward to today

... which is sourced:

```
> Rcpp::sourceCpp("code/calDemo.cpp")
> calDemo(Sys.Date(), 3)

## [1] "2014-12-03"
```

Fast-forward to today

Actual Code Example from Package [slightly compacted, flat namespaces]

```
#include <rquantlib.h>
// [[Rcpp::interfaces(r, cpp)]]
// [[Rcpp::export]]
List europeanOptionEngine(std::string type, double underlying, double strike,
      double dividendYield, double riskFreeRate, double maturity, double volatility)
{
  int length           = int(maturity*360 + 0.5); // FIXME: this could be better
  Option::Type optionType = getOptionType(type);
  Date today = Date::todaysDate();
  Settings::instance().evaluationDate() = today;
  DayCounter dc = Actual360();
  shared_ptr<SimpleQuote> spot(new SimpleQuote(underlying));
  shared_ptr<SimpleQuote> vol(new SimpleQuote(volatility));
  shared_ptr<BlackVolTermStructure> volTS = flatVol(today,vol,dc);
  shared_ptr<SimpleQuote> qRate(new SimpleQuote(dividendYield));
  shared_ptr<YieldTermStructure> qTS = flatRate(today,qRate,dc);
  shared_ptr<SimpleQuote> rRate(new SimpleQuote(riskFreeRate));
  shared_ptr<YieldTermStructure> rTS = flatRate(today, rRate, dc);
  Date exDate = today + length;
  shared_ptr<Exercise> exercise(new EuropeanExercise(exDate));
  shared_ptr<StrikedTypePayoff> payoff(new PlainVanillaPayoff(optionType, strike));
  shared_ptr<VanillaOption> opt = makeOption(payoff,exercise,spot,qTS,rTS,volTS);
  return List::create(Named("value") = opt->NPV(), Named("delta") = opt->delta(),
                      Named("gamma") = opt->gamma(), Named("vega") = opt->vega(),
                      Named("theta") = opt->theta(), Named("rho") = opt->rho(),
                      Named("divRho") = opt->dividendRho());

}
```

Outline

4 Rcpp
- How
- Example: Recursion
- Example: VAR(1)
- Growth

How do we use Rcpp?

Rcpp Attributes: evalCpp, cppFunction, sourceCpp

```
> ## evaluate a C++ expression, retrieve result
> evalCpp("2 + 2")

## [1] 4

> ## create ad-hoc R function 'square'
> cppFunction('int square(int x) { return x*x;}')
> square(7L)

## [1] 49

> ## or source an entire file (including R code)
> #sourceCpp("code/squareWithRCall.cpp")
```

When do we use Rcpp?

Easy speedup: An Introductory Example

Consider a function defined as

$$f(n) \quad \text{such that} \quad \begin{cases} n & \text{when} \quad n < 2 \\ f(n-1) + f(n-2) & \text{when} \quad n \geq 2 \end{cases}$$

When do we use Rcpp?

Easy speedup: Simple R Implementation

```
> fibR <- function(n) {
+     if (n < 2) return(n)
+     return(fibR(n-1) + fibR(n-2))
+ }
> ## Using it on first 11 arguments
> sapply(0:10, fibR)

##  [1]  0  1  1  2  3  5  8 13 21 34 55
```

When do we use Rcpp?

Easy speedup: Timing R Implementation

```
> benchmark(fibR(10),fibR(15),fibR(20))[,1:4]

##         test replications elapsed relative
## 1 fibR(10)          100   0.017    1.000
## 2 fibR(15)          100   0.201   11.824
## 3 fibR(20)          100   2.132  125.412
```

When do we use Rcpp?

Easy speedup: C++ Implementation

```
> cppFunction("
  int fibCpp(int n) {
    if (n < 2) return(n);
    return(fibCpp(n-1) + fibCpp(n-2));
}")
> ## Using it on first 11 arguments
> sapply(0:10, fibCpp)

##  [1]  0  1  1  2  3  5  8 13 21 34 55
```

When do we use Rcpp?

Easy speedup: Putting it all together

```
> fibR <- function(n) {
+     if (n<2) return(n)
+     return(fibR(n-1) + fibR(n-2))
+ }
> cppFunction('int fibCpp(int n) {
   if (n<2) return n;
   return fibCpp(n-2) + fibCpp(n-1);
}')
> benchmark(fibR(25), fibCpp(25), order="relative")[,1:4]

##          test replications elapsed relative
## 2 fibCpp(25)          100   0.058      1.0
## 1   fibR(25)          100  24.157    416.5
```

When would we use Rcpp?

Easy speed gain: VAR(1) Simulation

Let's consider a simple possible VAR(1) system of k variables.

For $k = 2$:

$$X_t = X_{t-1}B + E_t$$

where X_t is a row vector of length 2, B is a 2 by 2 matrix and E_t is a row of the error matrix of 2 columns.

When do we use Rcpp?

Easy speedup:: VAR(1) Simulation

In R code, given both the coefficient and error matrices (revealing *k* and *n*):

```
> rSim <- function(B,E) {
+     n <- nrow(E); k <- ncol(E)
+     X <- matrix(0, n, k)
+     for (r in 2:n) {
+         X[r,] = X[r-1, ] %*% B + E[r, ]
+     }
+     return(X)
+ }
```

When do we use Rcpp?

Easy speed gain: VAR(1) Simulation

```
> cppFunction('
arma::mat cppSim(const arma::mat& B, const arma::mat& E)
    int n = E.n_rows; int k = E.n_cols;
    arma::mat X = arma::zeros<arma::mat>(n,k);
    for (int r=1; r < n; r++) {
        X.row(r) = X.row(r-1) * B + E.row(r);
    }
    return X;
}', depends="RcppArmadillo")
```

When do we use Rcpp?

Easy speed gain: VAR(1) Simulation

```
#include <RcppArmadillo.h>

// [[Rcpp::depends(RcppArmadillo)]]

// [[Rcpp::export]]
arma::mat cppSim(const arma::mat& B,
                 const arma::mat& E) {
    int n = E.n_rows; int k = E.n_cols;
    arma::mat X = arma::zeros<arma::mat>(n,k);
    for (int r=1; r < n; r++) {
        X.row(r) = X.row(r-1) * B + E.row(r);
    }
    return X;
}
```

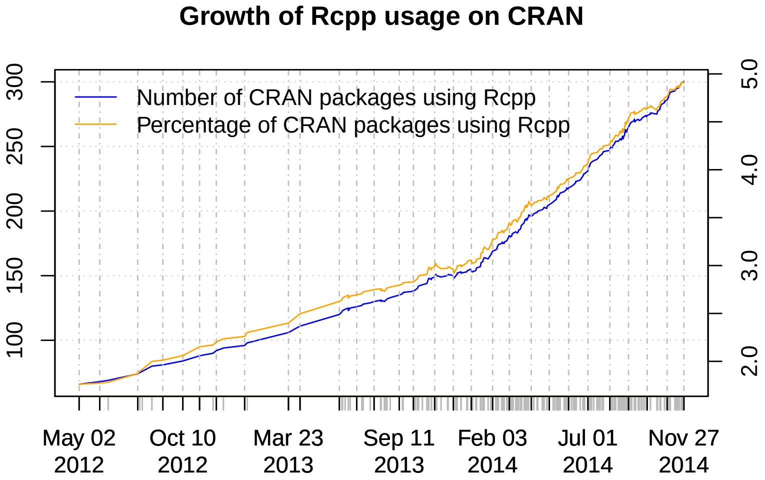
When do we use Rcpp?

Easy speed gain: VAR(1) Simulation

```
> a <- matrix(c(0.5,0.1,0.1,0.5),nrow=2)
> e <- matrix(rnorm(10000),ncol=2)
> all.equal(cppSim(a,e), rSim(a,e))

## [1] TRUE

> benchmark(cppSim(a,e), rSim(a,e),
+           order="relative")[,1:4]

##           test replications elapsed relative
## 1 cppSim(a, e)          100   0.024    1.000
## 2   rSim(a, e)          100   2.300   95.833
```

Rcpp on CRAN

Used by 304 packages, or just under 5 per cent



Outline

5 R and QuantLib
- General
- Shiny
- RMarkdown

Basics

- Basic just work: we convert standard C++ types, including STL containers seamlessly
- Custom converters can be added easily as shown for `QuantLib::Date`
- Should work out a proper R presentation of things like *curves* and *surfaces*
- R gives us a wealth of things for data creation, analysis and reporting
- Two recent R developments for reporting / communicating results highlighted in the next two sections.

Shiny for Dynamic Documents

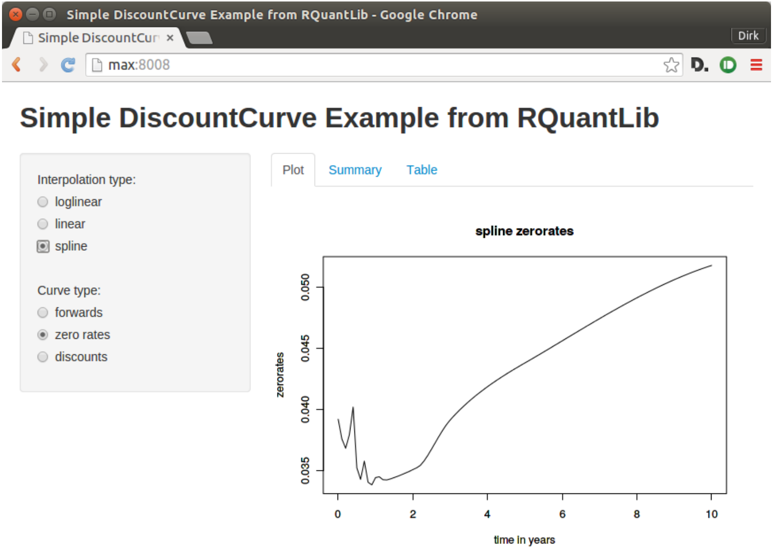
After a decade of GUI attempts, web frameworks, ...

Shiny just works:

- Minimal coding:
- One file `ui.R` to declare the user interface
- One file `server.R` to declare the backend
- Well documented, many examples and add-ons.

Shiny for Dynamic Documents

Quick Demo



Simple DiscountCurve Example from RQuantLib - Google Chrome

max:8008

Simple DiscountCurve Example from RQuantLib

Interpolation type:

- ○ loglinear
- ○ linear
- ● spline

Curve type:

- ○ forwards
- ● zero rates
- ○ discounts

Shiny for Dynamic Documents

ui.R for Demo

```
library(shiny)

shinyUI(fluidPage(
    ## Application title
    titlePanel("Simple DiscountCurve Example from RQuantLib"),
    ## Sidebar with controls to select parameters
    sidebarLayout(
        sidebarPanel(
            radioButtons("interpolation", "Interpolation type:",
                         c("loglinear" = "loglinear",
                           "linear" = "linear",
                           "spline" = "spline")),
            br(),
            radioButtons("curve", "Curve type:",
                         c("forwards" = "forwards",
                           "zero rates" = "zerorates",
                           "discounts" = "discounts"))
        ),
        ## Show a tabset that includes a plot, summary, and table view
        mainPanel(
            tabsetPanel(type = "tabs",
                        tabPanel("Plot", plotOutput("plot")),
                        tabPanel("Summary", verbatimTextOutput("summary")),
                        tabPanel("Table", tableOutput("table"))
                        )
        )
    )
))
```

Shiny for Dynamic Documents

server.R for Demo – slightly shortened / edited

```r
library(shiny); library(RQuantLib)
shinyServer(function(input, output) {
    params <- list(tradeDate=as.Date('2004-09-20'), ...)
    setEvaluationDate(as.Date("2004-09-20"))
    tsQuotes <- list(d1w = 0.0382, d1m = 0.0372, ...,  s15y = 0.055175)
    times <- seq(0,10,.1)
    data <- reactive({      ## Reactive expression to generate the requested curves.
        params$interpHow <- input$interpolation
        curve <- DiscountCurve(params, tsQuotes, times)
    })
    output$plot <- renderPlot({      ## Generate a plot of the data
        interp <- input$interpolation
        crv <- input$curve
        dat <- data()
        plot(dat[["times"]], dat[[crv]],
             type='l', main=paste(interp, crv), ylab=crv, xlab="time in years")
    })
    output$summary <- renderPrint({       ## Generate a summary of the data
        dat <- data()
        cat("Return Object Structure\n")
        str(dat)
        cat("\n\nSummary of first four elements\n")
        summary(data.frame(dat[1:4]))
    })
    output$table <- renderTable({     ## Generate an HTML table view of the data
        data.frame(x=data()[1:4])
    })
})
```

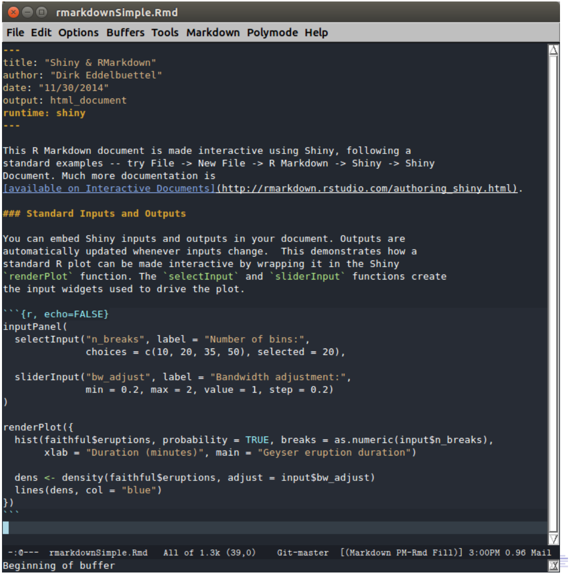
RMarkdown Overview

Easier (Informal) Publishing

RMarkdown extends basic Markdown in multiple ways:.

- Markdown can be mixed freely with R code expressions
- By relying on pandoc as the engine, conversion to html, latex/pdf, and even Word “just works”
- It complements the standard R + LaTeXapproach (used in these slides)
- The newest variant extends this for Dynamic Documents

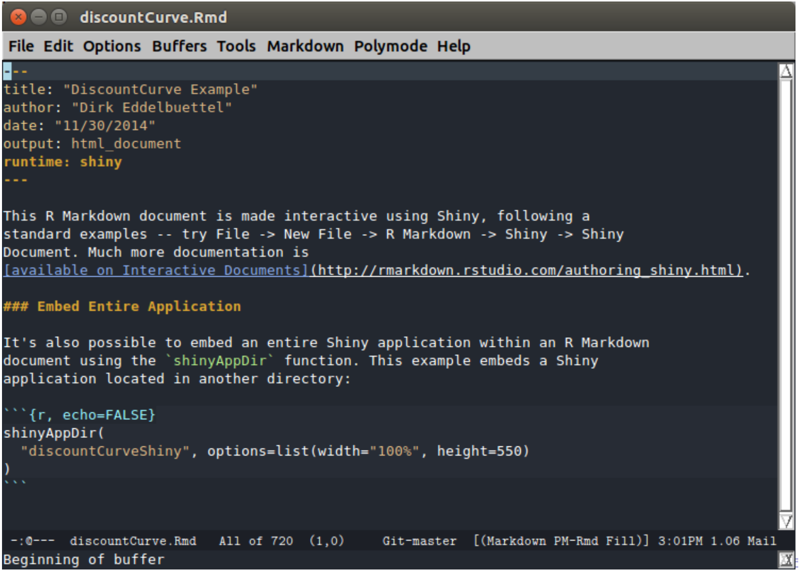
RMarkdown Demo



~~~
---
title: "Shiny & RMarkdown"
author: "Dirk Eddelbuettel"
date: "11/30/2014"
output: html_document
runtime: shiny
---

This R Markdown document is made interactive using Shiny, following a
standard examples -- try File -> New File -> R Markdown -> Shiny -> Shiny
Document. Much more documentation is
[available on Interactive Documents](http://rmarkdown.rstudio.com/authoring_shiny.html).

### Standard Inputs and Outputs

You can embed Shiny inputs and outputs in your document. Outputs are
automatically updated whenever inputs change.  This demonstrates how a
standard R plot can be made interactive by wrapping it in the Shiny
`renderPlot` function. The `selectInput` and `sliderInput` functions create
the input widgets used to drive the plot.

```{r, echo=FALSE}
inputPanel(
  selectInput("n_breaks", label = "Number of bins:",
              choices = c(10, 20, 35, 50), selected = 20),

  sliderInput("bw_adjust", label = "Bandwidth adjustment:",
              min = 0.2, max = 2, value = 1, step = 0.2)
)

renderPlot({
  hist(faithful$eruptions, probability = TRUE, breaks = as.numeric(input$n_breaks),
       xlab = "Duration (minutes)", main = "Geyser eruption duration")

  dens <- density(faithful$eruptions, adjust = input$bw_adjust)
  lines(dens, col = "blue")
})
~~~

```
 -:@---  rmarkdownSimple.Rmd   All of 1.3k (39,0)    Git-master  [(Markdown PM-Rmd Fill)] 3:00PM 0.96 Mail
Beginning of buffer
```

RMarkdown and Shiny Demo



~~~
---
title: "DiscountCurve Example"
author: "Dirk Eddelbuettel"
date: "11/30/2014"
output: html_document
runtime: shiny
---

This R Markdown document is made interactive using Shiny, following a
standard examples -- try File -> New File -> R Markdown -> Shiny -> Shiny
Document. Much more documentation is
[available on Interactive Documents](http://rmarkdown.rstudio.com/authoring_shiny.html).

### Embed Entire Application

It's also possible to embed an entire Shiny application within an R Markdown
document using the `shinyAppDir` function. This example embeds a Shiny
application located in another directory:

```{r, echo=FALSE}
shinyAppDir(
  "discountCurveShiny", options=list(width="100%", height=550)
)
```
~~~

```
 -:@---  discountCurve.Rmd   All of 720  (1,0)     Git-master  [(Markdown PM-Rmd Fill)] 3:01PM 1.06 Mail
Beginning of buffer
```

Outline

6 Issues

Open Issues / Challenges

- Statefulness etc is an issue for R interface: So far just simple calls and returns; one singleton.
- Eventually need a way to hang on to objects and revisit them.
- QL issue of 'time unit is a single day' is limiting
- Overall balance of featuritis and ease of use (still no simple Black/Scholes)

Outline

7 Successes

Achievements

- RQuantLib spawned Rcpp.
- RQuantLib, while incomplete, has a number of users.
- Automatic Windows builds from CRAN help a lot.
- (Mistyifies me as I see this a *development framework* rather than an appliance ...)
- The Future is so bright...

Postscriptum

Slides are available at my presentations page[6].

Example code is in the samplecode[7] github repo in the directory `quantlib-2014-12`.

Rcpp repository[8]

RQuantLib repository[9]

Rcpp Gallery[10]

[6] http://dirk.eddelbuettel.com/presentations/

[7] https://github.com/eddelbuettel/samplecode/

[8] https://github.com/RcppCore/Rcpp

[9] https://github.com/eddelbuettel/rquantlib

[10] http://gallery.rcpp.org