

Bill Venables, S and R

Ross Ihaka
University of Auckland

Talk Outline

Talk Outline

- Bill's role in the R and S community.

Talk Outline

- Bill's role in the R and S community.
- Some of Bill's programming work (Examples).

Talk Outline

- Bill's role in the R and S community.
- Some of Bill's programming work (Examples).
 - Code from the `english` package.

Talk Outline

- Bill's role in the R and S community.
- Some of Bill's programming work (Examples).
 - Code from the `english` package.
 - Some code from the `polynom` package.

Talk Outline

- Bill's role in the R and S community.
- Some of Bill's programming work (Examples).
 - Code from the `english` package.
 - Some code from the `polynom` package.
 - The `SOAR` package.



Image credit: <http://thomasmonson.com>

Bill the Educator

Bill the Educator

- His original set of notes on S was an entry-point for many early adopters of S.

Bill the Educator

- His original set of notes on S was an entry-point for many early adopters of S.
- The MASS book (with Brian Ripley) has been a standard introduction to both S and R as well providing guidance on the use of statistics to students and practitioners.

Bill the Educator

- His original set of notes on S was an entry-point for many early adopters of S.
- The MASS book (with Brian Ripley) has been a standard introduction to both S and R as well providing guidance on the use of statistics to students and practitioners.
- The S Programming book (again with Ripley) provided a concise introduction to programming in S.

Bill the Educator

- His original set of notes on S was an entry-point for many early adopters of S.
- The MASS book (with Brian Ripley) has been a standard introduction to both S and R as well providing guidance on the use of statistics to students and practitioners.
- The S Programming book (again with Ripley) provided a concise introduction to programming in S.
- His code has often provided a prototype for how to build software depending on some of the “less familiar” aspects of S and R.

Bill the Educator

- His original set of notes on S was an entry-point for many early adopters of S.
- The MASS book (with Brian Ripley) has been a standard introduction to both S and R as well providing guidance on the use of statistics to students and practitioners.
- The S Programming book (again with Ripley) provided a concise introduction to programming in S.
- His code has often provided a prototype for how to build software depending on some of the “less familiar” aspects of S and R.
- Bill has spent a great deal of time explaining the finer points of R and S to a worldwide community of users.



Image credit: Fernandoh Rosa (Brazil 2005)

Bill the Scholar

Bill brings a serious tone to any discussion.

Bill the Scholar

Bill brings a serious tone to any discussion.

- *Exegesis* – a critical explanation or interpretation of a text.

Bill the Scholar

Bill brings a serious tone to any discussion.

- *Exegesis* – a critical explanation or interpretation of a text.
- *Infelicity* – a thing that is inappropriate, especially a remark or expression.

Bill the Programmer

Bill the Programmer

- Bill writes good software.

Bill the Programmer

- Bill writes good software.

“His negative binomial software was the first such software I found that actually worked for me.”

Bill the Programmer

- Bill writes good software.

“His negative binomial software was the first such software I found that actually worked for me.”

- His software is often designed to be read by others. This is a mark of good software.

Bill the Programmer

- Bill writes good software.

“His negative binomial software was the first such software I found that actually worked for me.”

- His software is often designed to be read by others. This is a mark of good software.
- I have often been surprised when I have begin poking about in an area only to find Bill's footprints already there.

Bill the Programmer

- Bill writes good software.

“His negative binomial software was the first such software I found that actually worked for me.”

- His software is often designed to be read by others. This is a mark of good software.
- I have often been surprised when I have begin poking about in an area only to find Bill's footprints already there.
- So let's take a look inside . . .

A Function for Printing Numbers as Words

This is a function extracted from the `english` package (with the object-oriented wrapping discarded).

A Function for Printing Numbers as Words

This is a function extracted from the `english` package (with the object-oriented wrapping discarded).

```
> words(123)
```

A Function for Printing Numbers as Words

This is a function extracted from the `english` package (with the object-oriented wrapping discarded).

```
> words(123)
[1] "one hundred and twenty three"
```

A Function for Printing Numbers as Words

This is a function extracted from the `english` package (with the object-oriented wrapping discarded).

```
> words(123)
[1] "one hundred and twenty three"

> words(1234)
```

A Function for Printing Numbers as Words

This is a function extracted from the `english` package (with the object-oriented wrapping discarded).

```
> words(123)
```

```
[1] "one hundred and twenty three"
```

```
> words(1234)
```

```
[1] "one thousand two hundred and thirty four"
```

A Function for Printing Numbers as Words

This is a function extracted from the `english` package (with the object-oriented wrapping discarded).

```
> words(123)
```

```
[1] "one hundred and twenty three"
```

```
> words(1234)
```

```
[1] "one thousand two hundred and thirty four"
```

```
> words(253)
```

A Function for Printing Numbers as Words

This is a function extracted from the `english` package (with the object-oriented wrapping discarded).

```
> words(123)
[1] "one hundred and twenty three"

> words(1234)
[1] "one thousand two hundred and thirty four"

> words(2^53)
[1] "nine quadrillion seven trillion
    one hundred and ninety nine billion
    two hundred and fifty four million
    seven hundred and forty thousand
    nine hundred and ninety two"
```

Useful Programming Strategies

- Decompose big problems into smaller ones, recursively if necessary.
- Solve the smaller problems, preferably in simple ways.
- Combine small solutions into bigger ones, simple combination methods are preferable to complex ones.
- Write code for people to read and understand rather than for machines to execute.

Dealing with Smaller Values

For numbers less than one thousand, the basic idea is to split the value into its individual digits.

123 → 1, 2 and 3

These are used to make up the component words of the value.

1 → “one” “hundred”

2 → “twenty”

3 → “three”

If the number is greater than 99 and either of the tens or one digits is non-zero then “and” must be inserted.

“one hundred and twenty three”

Dealing with Larger Values

Values of one thousand or more are split into groups of three digits

1,234,567 → 1, 234 and 567.

Each group of three is treated in the same way as smaller values, but has a suitable size suffix added.

1 → “one” “million”

234 → “two hundred and thirty four” “thousand”

567 → “five hundred and sixty seven” “”

Implementation

The following components are apparent from this analysis:

- Tables in which to look up the character string components of numbers and size suffixes.
- A function that handles “small” (less than 1000) values.
- A function that uses the “small value” function to encode larger values.

There is really only one thing of interest here; the function that can be applied to both big and small values.

The tables and “small value” function should be hidden away.

Hiding Complexity

R provides a way of hiding “private” functions and variables “inside” a function.

These are visible inside the function, but not outside.

```
fun =  
  local({  
    <private variable and function definitions>  
  
    <function definition>  
  })
```

This *closure* mechanism is inherited from Lisp and Scheme and provides a powerful way of building software components.

Code Structure

```
words = local({  
  
    helper =  
        local({  
            ⟨Tables for word component lookup⟩  
            function(x) {  
                ⟨Code for the small number helper function⟩  
            }  
        })  
  
    function(x) {  
        ⟨Code for the general case function⟩  
    }  
})
```

Alternative Code Structure

```
words = local({
```

```
    ⟨Tables for word component lookup⟩
```

```
    helper =
```

```
        function(x) {
```

```
            ⟨Code for the small number helper function⟩
```

```
        }
```

```
    function(x) {
```

```
        ⟨Code for the general case function⟩
```

```
    }
```

```
})
```

A Small Worry

The code contains the following definition.

```
trim =  
  function (text)  
    sub("^ *", "",  
        sub(" *$", "",  
            gsub("  +", " ", text)))
```

This function removes, leading, trailing and repeated spaces.

A Small Worry

The code contains the following definition.

```
trim =  
  function (text)  
    sub("^ *", "",  
        sub(" *$", "",  
            gsub("  +", " ", text)))
```

This function removes, leading, trailing and repeated spaces.

This is a classic cleanup *hack*.

Possible Enhancements

It is standard to place a hyphen between tens and ones digits (for values greater than twenty).

Numbers are sometimes written in a comma separated form.

1234567 → 1,234,567

It can be useful to use this kind of separation when writing numbers in words. (This is controversial and should perhaps be optional.)

One million, two hundred and thirty-four thousand,
five hundred and sixty-seven.

Restrictions

Care must be taken when dealing with integer values greater than or equal to 2^{53} .

Restrictions

Care must be taken when dealing with integer values greater than or equal to 2^{53} .

```
> as.character(2^53 - 1)
[1] "9007199254740991"
```

Restrictions

Care must be taken when dealing with integer values greater than or equal to 2^{53} .

```
> as.character(2^53 - 1)
[1] "9007199254740991"
```

```
> as.character(2^53)
[1] "9007199254740992"
```

Restrictions

Care must be taken when dealing with integer values greater than or equal to 2^{53} .

```
> as.character(2^53 - 1)
[1] "9007199254740991"
```

```
> as.character(2^53)
[1] "9007199254740992"
```

```
> as.character(2^53 + 1)
[1] "9007199254740992"
```

It might be useful to warn users that the number passed in may not be as accurate as they think they are.

Polynomial Manipulation

- A polynomial

$$a_0 + a_1x + a_2x^2 + \cdots + a_dx^d$$

can be represented as an R vector

$$c(a_0, a_1, \dots, a_d).$$

- Polynomial manipulation using operations such as addition, subtraction etc. can then be carried out using R.
- This is what Bill's `polynom` library does.
- The library is written as an example of how to write object-oriented code.

Polynomial Multiplication

Consider multiplying the polynomials

$$p = a_0 + a_1x + a_2x^2,$$
$$q = b_0 + b_1x + b_2x^2 + b_3x^3.$$

This can be done by multiplying out terms

$$a_i x^i \times b_j x^j = a_i b_j x^{i+j}$$

and combining like powered terms by addition.

$$\left(\sum_{k=i+j} a_i b_j \right) x^k$$

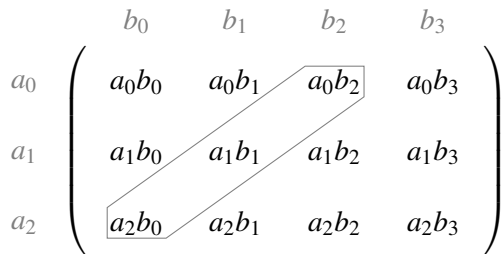
Polynomial Multiplication Computation

The individual polynomial coefficient products can be computed as the outer product of the coefficient vectors.

$$\begin{array}{c} a_0 \\ a_1 \\ a_2 \end{array} \begin{pmatrix} b_0 & b_1 & b_2 & b_3 \\ a_0b_0 & a_0b_1 & a_0b_2 & a_0b_3 \\ a_1b_0 & a_1b_1 & a_1b_2 & a_1b_3 \\ a_2b_0 & a_2b_1 & a_2b_2 & a_2b_3 \end{pmatrix}$$

Polynomial Coefficient Calculation

The coefficients of the product polynomial can then be calculated by by summing the appropriate elements of the outer product.

$$\begin{array}{c} a_0 \\ a_1 \\ a_2 \end{array} \begin{pmatrix} b_0 & b_1 & b_2 & b_3 \\ a_0b_0 & a_0b_1 & a_0b_2 & a_0b_3 \\ a_1b_0 & a_1b_1 & a_1b_2 & a_1b_3 \\ a_2b_0 & a_2b_1 & a_2b_2 & a_2b_3 \end{pmatrix}$$
The diagram shows a 3x4 matrix representing the outer product of vectors $[a_0, a_1, a_2]$ and $[b_0, b_1, b_2, b_3]$. The columns are labeled b_0, b_1, b_2, b_3 at the top. The rows are labeled a_0, a_1, a_2 on the left. The matrix elements are $a_i b_j$. A white parallelogram highlights the diagonal elements a_0b_2, a_1b_1, a_2b_0 , which are the coefficients of x^2 in the product polynomial.

The coefficients above can be summed to obtain the coefficient of x^2 in the polynomial.

R Code

Here is the R code that implements polynomial multiplication.

```
pmult =  
  function(a, b) {  
    m = outer(a, b)  
    tapply(m, row(m)+col(m), sum)  
  }
```

This code is simple, clear and very useful for teaching object-oriented programming.

R Code

Here is the R code that implements polynomial multiplication.

```
pmult =  
  function(a, b) {  
    m = outer(a, b)  
    tapply(m, row(m)+col(m), sum)  
  }
```

This code is simple, clear and very useful for teaching object-oriented programming.

However ...

R Code

Here is the R code that implements polynomial multiplication.

```
pmult =  
  function(a, b) {  
    m = outer(a, b)  
    tapply(m, row(m)+col(m), sum)  
  }
```

This code is simple, clear and very useful for teaching object-oriented programming.

However ...

There are other ways of multiplying polynomials.

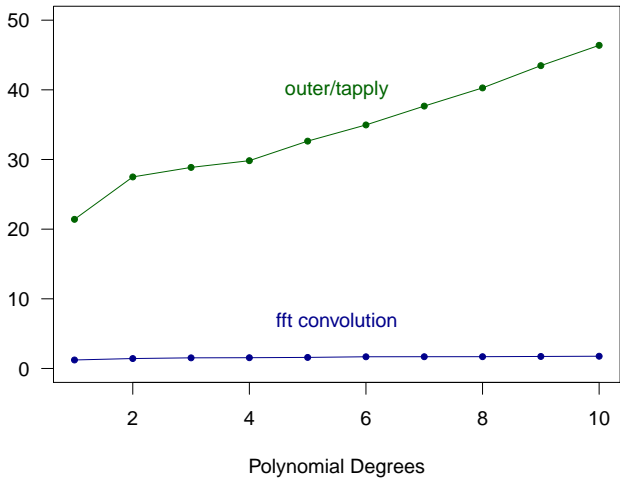
Polynomial Multiplication Via the FFT

The coefficients of the product polynomial can be obtained by convolution of the coefficients of the polynomials being multiplied.

This can be done efficiently using the FFT.

```
pmult =  
  function(a, b) {  
    na = length(a)  
    nb = length(b)  
    Re(fft(fft(c(a, rep(0, nb - 1))) *  
        fft(c(b, rep(0, na - 1))),  
        inverse = TRUE)) / (na + nb - 1)  
  }
```

Time Taken for 100,000 Multiplies (Seconds)



The SOAR Package

SOAR provides a way for you to have your data immediately available while still leaving memory free to get work done.

The SOAR Package

SOAR provides a way for you to have your data immediately available while still leaving memory free to get work done.

```
> objects()  
character(0)
```


The SOAR Package

SOAR provides a way for you to have your data immediately available while still leaving memory free to get work done.

```
> objects()  
character(0)  
> x = 1:10
```

The SOAR Package

SOAR provides a way for you to have your data immediately available while still leaving memory free to get work done.

```
> objects()
character(0)
> x = 1:10
> objects()
[1] "x"
```

The SOAR Package

SOAR provides a way for you to have your data immediately available while still leaving memory free to get work done.

```
> objects()  
character(0)  
> x = 1:10  
> objects()  
[1] "x"  
> Store(x)
```

The SOAR Package

SOAR provides a way for you to have your data immediately available while still leaving memory free to get work done.

```
> objects()
character(0)
> x = 1:10
> objects()
[1] "x"
> Store(x)
> objects()
character(0)
```

The SOAR Package

SOAR provides a way for you to have your data immediately available while still leaving memory free to get work done.

```
> objects()
character(0)
> x = 1:10
> objects()
[1] "x"
> Store(x)
> objects()
character(0)
> x
[1]  1  2  3  4  5  6  7  8  9 10
```

The Mechanism

- SOAR relies on the *delay/force* mechanism (inherited from Scheme) that R uses to implement lazy evaluation of function arguments.
- The underlying data structure is a “promise” which consists of a “think” (a function of no arguments that can be called to obtain a value) and a cached version of the value it returns.
- When a promise is referenced, the value in the cache is returned.
- If there is no value in the cache, the think is called to obtain a value for the cache.

How SOAR Works

- When the **Store** function is called, the data value is written to disk and a promise to fetch the value is created and given the same name.
- The original value is then removed.
- If the value is referenced, the “promise to load” is forced and the value becomes available.
- If the value remains unreferenced, the promise cache remains empty and memory is conserved.

A Warning

From the SOAR documentation:

While it is convenient to carry objects over from one session to another, particularly during the period where an analysis is being developed, it can be a mistake to rely on R data objects gradually severing the link with the primary sources of the data. We would encourage users to make and keep scripts which construct all important data sets and analyses from primary sources and to be able to re-construct the entire process from them.

Summary

- I have just touched briefly on Bill's ongoing contributions to the S and R communities.
- I have tried to show show some of the sophistication present in his programming work.

Summary

- I have just touched briefly on Bill's ongoing contributions to the S and R communities.
- I have tried to show show some of the sophistication present in his programming work.
- Long may both types of contribution continue!

Summary

- I have just touched briefly on Bill's ongoing contributions to the S and R communities.
- I have tried to show show some of the sophistication present in his programming work.
- Long may both types of contribution continue!
- Which just leaves ...

HAPPY
BIRTHDAY