UNIVERSITY OF COPENHAGEN DEPARTMENT OF BIOSTATISTICS

Faculty of Health Sciences

What we wish people knew more about when working with R

Peter Dalgaard

Dept. of Biostatistics

University of Copenhagen

Background

- R has entered the mainstream, and a great many research projects in statistics now involve R programming or the writing of R packages
- Young researchers will typically need to be taught about relatively advanced aspects of R
- Consider planning, say, an advanced course on R programming
- Much will be pretty straightforward
- Not necessarily easy, but you know that you need to take the students from A to B along a path with certain twist and turns and stumbling stones

Background

- R has entered the mainstream, and a great many research projects in statistics now involve R programming or the writing of R packages
- Young researchers will typically need to be taught about relatively advanced aspects of R
- Consider planning, say, an advanced course on R programming
- Much will be pretty straightforward
- Not necessarily easy, but you know that you need to take the students from A to B along a path with certain twist and turns and stumbling stones

Background

- R has entered the mainstream, and a great many research projects in statistics now involve R programming or the writing of R packages
- Young researchers will typically need to be taught about relatively advanced aspects of R
- Consider planning, say, an advanced course on R programming
- Much will be pretty straightforward
- Not necessarily easy, but you know that you need to take the students from A to B along a path with certain twist and turns and stumbling stones

Background

- R has entered the mainstream, and a great many research projects in statistics now involve R programming or the writing of R packages
- Young researchers will typically need to be taught about relatively advanced aspects of R
- Consider planning, say, an advanced course on R programming
- Much will be pretty straightforward
- Not necessarily easy, but you know that you need to take the students from A to B along a path with certain twist and turns and stumbling stones

Background

- R has entered the mainstream, and a great many research projects in statistics now involve R programming or the writing of R packages
- Young researchers will typically need to be taught about relatively advanced aspects of R
- Consider planning, say, an advanced course on R programming
- Much will be pretty straightforward
- Not necessarily easy, but you know that you need to take the students from A to B along a path with certain twist and turns and stumbling stones

The blank stare

- At some points, however, you find yourself facing a wall of ignorance
- There are things students just don't know the first thing about
- Say, you want to show how to speed up a slow piece of R code
- So you explain that they should rewrite parts of the code in C, compile it, and link it dynamically
  - What is C?
  - What is a compiler?
  - What is linking?

The blank stare

- At some points, however, you find yourself facing a wall of ignorance
- There are things students just don’t know the first thing about
- Say, you want to show how to speed up a slow piece of R code
- So you explain that they should rewrite parts of the code in C, compile it, and link it dynamically
    - What is C?
    - What is a compiler?
    - What is linking?

The blank stare

- At some points, however, you find yourself facing a wall of ignorance
- There are things students just don't know the first thing about
- Say, you want to show how to speed up a slow piece of R code
- So you explain that they should rewrite parts of the code in C, compile it, and link it dynamically
  - What is C?
  - What is a compiler?
  - What is linking?

The blank stare

- At some points, however, you find yourself facing a wall of ignorance
- There are things students just don't know the first thing about
- Say, you want to show how to speed up a slow piece of R code
- So you explain that they should rewrite parts of the code in C, compile it, and link it dynamically
  - What is C?
  - What is a compiler?
  - What is linking?

The blank stare

- At some points, however, you find yourself facing a wall of ignorance
- There are things students just don't know the first thing about
- Say, you want to show how to speed up a slow piece of R code
- So you explain that they should rewrite parts of the code in C, compile it, and link it dynamically
  - What is C?
  - What is a compiler?
  - What is linking?

The blank stare

- At some points, however, you find yourself facing a wall of ignorance
- There are things students just don't know the first thing about
- Say, you want to show how to speed up a slow piece of R code
- So you explain that they should rewrite parts of the code in C, compile it, and link it dynamically
  - What is C?
  - What is a compiler?
  - What is linking?

The blank stare

- At some points, however, you find yourself facing a wall of ignorance
- There are things students just don't know the first thing about
- Say, you want to show how to speed up a slow piece of R code
- So you explain that they should rewrite parts of the code in C, compile it, and link it dynamically
  - What is C?
  - What is a compiler?
  - What is linking?

Generic problem

- In order to explain Z, I must first tell them about Y, but that won't make sense to them because they never heard of X, etc.
- This is getting worse! A generic trend in computing is that more and more functionality gets hidden away.
- In some senses, this may be a good trend, making computers accessible by more people
- However, from a scientific point of view, it makes it harder to understand what is going on inside a computer
- (Car analogy: Making cars simpler and safer to operate does not make better car engineers)

Generic problem

- In order to explain Z, I must first tell them about Y, but that won't make sense to them because they never heard of X, etc.
- This is getting worse! A generic trend in computing is that more and more functionality gets hidden away.
- In some senses, this may be a good trend, making computers accessible by more people
- However, from a scientific point of view, it makes it harder to understand what is going on inside a computer
- (Car analogy: Making cars simpler and safer to operate does not make better car engineers)

Generic problem

- In order to explain Z, I must first tell them about Y, but that won't make sense to them because they never heard of X, etc.
- This is getting worse! A generic trend in computing is that more and more functionality gets hidden away.
- In some senses, this may be a good trend, making computers accessible by more people
- However, from a scientific point of view, it makes it harder to understand what is going on inside a computer
- (Car analogy: Making cars simpler and safer to operate does not make better car engineers)

Generic problem

- In order to explain Z, I must first tell them about Y, but that won't make sense to them because they never heard of X, etc.
- This is getting worse! A generic trend in computing is that more and more functionality gets hidden away.
- In some senses, this may be a good trend, making computers accessible by more people
- However, from a scientific point of view, it makes it harder to understand what is going on inside a computer
- (Car analogy: Making cars simpler and safer to operate does not make better car engineers)

Generic problem

- In order to explain Z, I must first tell them about Y, but that won't make sense to them because they never heard of X, etc.
- This is getting worse! A generic trend in computing is that more and more functionality gets hidden away.
- In some senses, this may be a good trend, making computers accessible by more people
- However, from a scientific point of view, it makes it harder to understand what is going on inside a computer
- (Car analogy: Making cars simpler and safer to operate does not make better car engineers)

How do we know what we know?

- Is education deteriorating?
- Not really. If we look back, people who were into statistical computing were often not formally educated.
- Some people had switched from Computer Science to Statistics
- Others came out of the "Commodore 64" generation (typically teenagers from the 80s and 90s)
- At about the time R took off, there was the IT explosion and the whole Unix/Linux/Open Source culture around the turn of the millennium
- We are now moving from a relatively tight-knit subculture to a position in the mainstream, and this requires new thinking

How do we know what we know?

- Is education deteriorating?
- Not really. If we look back, people who were into statistical computing were often not formally educated.
- Some people had switched from Computer Science to Statistics
- Others came out of the "Commodore 64" generation (typically teenagers from the 80s and 90s)
- At about the time R took off, there was the IT explosion and the whole Unix/Linux/Open Source culture around the turn of the millennium
- We are now moving from a relatively tight-knit subculture to a position in the mainstream, and this requires new thinking

How do we know what we know?

- Is education deteriorating?
- Not really. If we look back, people who were into statistical computing were often not formally educated.
- Some people had switched from Computer Science to Statistics
- Others came out of the "Commodore 64" generation (typically teenagers from the 80s and 90s)
- At about the time R took off, there was the IT explosion and the whole Unix/Linux/Open Source culture around the turn of the millennium
- We are now moving from a relatively tight-knit subculture to a position in the mainstream, and this requires new thinking

How do we know what we know?

- Is education deteriorating?
- Not really. If we look back, people who were into statistical computing were often not formally educated.
- Some people had switched from Computer Science to Statistics
- Others came out of the "Commodore 64" generation (typically teenagers from the 80s and 90s)
- At about the time R took off, there was the IT explosion and the whole Unix/Linux/Open Source culture around the turn of the millennium
- We are now moving from a relatively tight-knit subculture to a position in the mainstream, and this requires new thinking

How do we know what we know?

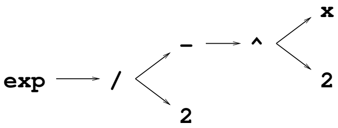
- Is education deteriorating?
- Not really. If we look back, people who were into statistical computing were often not formally educated.
- Some people had switched from Computer Science to Statistics
- Others came out of the "Commodore 64" generation (typically teenagers from the 80s and 90s)
- At about the time R took off, there was the IT explosion and the whole Unix/Linux/Open Source culture around the turn of the millennium
- We are now moving from a relatively tight-knit subculture to a position in the mainstream, and this requires new thinking

How do we know what we know?

- Is education deteriorating?
- Not really. If we look back, people who were into statistical computing were often not formally educated.
- Some people had switched from Computer Science to Statistics
- Others came out of the "Commodore 64" generation (typically teenagers from the 80s and 90s)
- At about the time R took off, there was the IT explosion and the whole Unix/Linux/Open Source culture around the turn of the millennium
- We are now moving from a relatively tight-knit subculture to a position in the mainstream, and this requires new thinking

Example: Parse trees

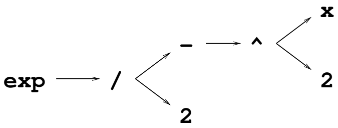
```
exp(-x^2/2)
```



- In math, people know operator precedence intuitively
- However, they may not always realize that there is a well-defined process (parsing) leading from one representation to the other
- Or, that this in R is represented as an object which forms the basis of the later evaluation

Example: Parse trees

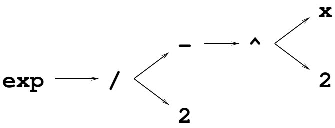
```
exp(-x^2/2)
```



- In math, people know operator precedence intuitively
- However, they may not always realize that there is a well-defined process (parsing) leading from one representation to the other
- Or, that this in R is represented as an object which forms the basis of the later evaluation

Example: Parse trees

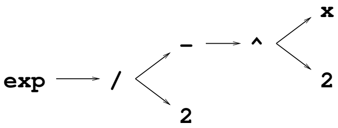
```
exp(-x^2/2)
```



- In math, people know operator precedence intuitively
- However, they may not always realize that there is a well-defined process (parsing) leading from one representation to the other
- Or, that this in R is represented as an object which forms the basis of the later evaluation

Example: Parse trees

```
exp(-x^2/2)
```



- In math, people know operator precedence intuitively
- However, they may not always realize that there is a well-defined process (parsing) leading from one representation to the other
- Or, that this in R is represented as an object which forms the basis of the later evaluation

How did I know about parsing?

- Mixture of many sources
- Back pages of “Pascal User Manual and Report”: recursive descent parser
- PL/0 parser in Wirth: “Algorithms + Data Stuctures = Programs”. This was not actually in the curriculum, but I rubbed shoulders with 3rd yr CS students
- Exposure to Genstat, BMDP (ca. 1980)
- Aho & Ullman’s “Dragon book” taught me about LALR(1) grammars
- HP-UX series 300 computer on a project with som eye doctors. This contained YACC – “Yet Another Compiler-Compiler”

How did I know about parsing?

- Mixture of many sources
- Back pages of “Pascal User Manual and Report”: recursive descent parser
- PL/0 parser in Wirth: “Algorithms + Data Stuctures = Programs”. This was not actually in the curriculum, but I rubbed shoulders with 3rd yr CS students
- Exposure to Genstat, BMDP (ca. 1980)
- Aho & Ullman’s “Dragon book” taught me about LALR(1) grammars
- HP-UX series 300 computer on a project with som eye doctors. This contained YACC – “Yet Another Compiler-Compiler”

How did I know about parsing?

- Mixture of many sources
- Back pages of "Pascal User Manual and Report": recursive descent parser
- PL/0 parser in Wirth: "Algorithms + Data Stuctures = Programs". This was not actually in the curriculum, but I rubbed shoulders with 3rd yr CS students
- Exposure to Genstat, BMDP (ca. 1980)
- Aho & Ullman's "Dragon book" taught me about LALR(1) grammars
- HP-UX series 300 computer on a project with som eye doctors. This contained YACC – "Yet Another Compiler-Compiler"

How did I know about parsing?

- Mixture of many sources
- Back pages of “Pascal User Manual and Report”: recursive descent parser
- PL/0 parser in Wirth: “Algorithms + Data Stuctures = Programs”. This was not actually in the curriculum, but I rubbed shoulders with 3rd yr CS students
- Exposure to Genstat, BMDP (ca. 1980)
- Aho & Ullman’s “Dragon book” taught me about LALR(1) grammars
- HP-UX series 300 computer on a project with som eye doctors. This contained YACC – “Yet Another Compiler-Compiler”

How did I know about parsing?

- Mixture of many sources
- Back pages of “Pascal User Manual and Report”: recursive descent parser
- PL/0 parser in Wirth: “Algorithms + Data Stuctures = Programs”. This was not actually in the curriculum, but I rubbed shoulders with 3rd yr CS students
- Exposure to Genstat, BMDP (ca. 1980)
- Aho & Ullman’s “Dragon book” taught me about LALR(1) grammars
- HP-UX series 300 computer on a project with som eye doctors. This contained YACC – “Yet Another Compiler-Compiler”

How did I know about parsing?

- Mixture of many sources
- Back pages of “Pascal User Manual and Report”: recursive descent parser
- PL/0 parser in Wirth: “Algorithms + Data Stuctures = Programs”. This was not actually in the curriculum, but I rubbed shoulders with 3rd yr CS students
- Exposure to Genstat, BMDP (ca. 1980)
- Aho & Ullman’s “Dragon book” taught me about LALR(1) grammars
- HP-UX series 300 computer on a project with som eye doctors. This contained YACC – “Yet Another Compiler-Compiler”

A catalogue of ignorance

- Parsing
- Interfacing to C
- Floating point issues
- Computational linear algebra
- Finer points in computer languages
- Obvious pitfall: Trying to explain in a 40 minute talk what I claim requires a significant chunk of a largish course
- Pitfall no. 2: The grumpy old man. . .
- Pitfall no. 3: Displaying my own ignorance

A catalogue of ignorance

- Parsing
- Interfacing to C
- Floating point issues
- Computational linear algebra
- Finer points in computer languages
- Obvious pitfall: Trying to explain in a 40 minute talk what I claim requires a significant chunk of a largish course
- Pitfall no. 2: The grumpy old man. . .
- Pitfall no. 3: Displaying my own ignorance

A catalogue of ignorance

- Parsing
- Interfacing to C
- Floating point issues
- Computational linear algebra
- Finer points in computer languages
- Obvious pitfall: Trying to explain in a 40 minute talk what I claim requires a significant chunk of a largish course
- Pitfall no. 2: The grumpy old man. . .
- Pitfall no. 3: Displaying my own ignorance

A catalogue of ignorance

- Parsing
- Interfacing to C
- Floating point issues
- Computational linear algebra
- Finer points in computer languages
- Obvious pitfall: Trying to explain in a 40 minute talk what I claim requires a significant chunk of a largish course
- Pitfall no. 2: The grumpy old man. . .
- Pitfall no. 3: Displaying my own ignorance

A catalogue of ignorance

- Parsing
- Interfacing to C
- Floating point issues
- Computational linear algebra
- Finer points in computer languages
- Obvious pitfall: Trying to explain in a 40 minute talk what I claim requires a significant chunk of a largish course
- Pitfall no. 2: The grumpy old man...
- Pitfall no. 3: Displaying my own ignorance

A catalogue of ignorance

- Parsing
- Interfacing to C
- Floating point issues
- Computational linear algebra
- Finer points in computer languages
- Obvious pitfall: Trying to explain in a 40 minute talk what I claim requires a significant chunk of a largish course
- Pitfall no. 2: The grumpy old man. . .
- Pitfall no. 3: Displaying my own ignorance

A catalogue of ignorance

- Parsing
- Interfacing to C
- Floating point issues
- Computational linear algebra
- Finer points in computer languages
- Obvious pitfall: Trying to explain in a 40 minute talk what I claim requires a significant chunk of a largish course
- Pitfall no. 2: The grumpy old man. . .
- Pitfall no. 3: Displaying my own ignorance

A catalogue of ignorance

- Parsing
- Interfacing to C
- Floating point issues
- Computational linear algebra
- Finer points in computer languages
- Obvious pitfall: Trying to explain in a 40 minute talk what I claim requires a significant chunk of a largish course
- Pitfall no. 2: The grumpy old man...
- Pitfall no. 3: Displaying my own ignorance

Parsing

- Internal structure of expressions, code
- Needed in plotmath, model formulas
- Names and syntactical names
- Tokenizer, lexical analysis, (regular expressions)
- Properties of computer syntax: One-step lookahead, R's newline anomaly

Parsing

- Internal structure of expressions, code
- Needed in plotmath, model formulas
- Names and syntactical names
- Tokenizer, lexical analysis, (regular expressions)
- Properties of computer syntax: One-step lookahead, R's newline anomaly

Parsing

- Internal structure of expressions, code
- Needed in plotmath, model formulas
- Names and syntactical names
- Tokenizer, lexical analysis, (regular expressions)
- Properties of computer syntax: One-step lookahead, R's newline anomaly

Parsing

- Internal structure of expressions, code
- Needed in plotmath, model formulas
- Names and syntactical names
- Tokenizer, lexical analysis, (regular expressions)
- Properties of computer syntax: One-step lookahead, R's newline anomaly

Parsing

- Internal structure of expressions, code
- Needed in plotmath, model formulas
- Names and syntactical names
- Tokenizer, lexical analysis, (regular expressions)
- Properties of computer syntax: One-step lookahead, R's newline anomaly

Floating-point issues

- Limits of accuracy, decimals not representable in binary
- (FAQ 7.31...)
- Deeper issue: knowledge of bit-level storage and hardware
- IEEE standards
- FP exceptions
- Loss of fine control caused by optimizers reordering code

Floating-point issues

- Limits of accuracy, decimals not representable in binary
- (FAQ 7.31...)
- Deeper issue: knowledge of bit-level storage and hardware
- IEEE standards
- FP exceptions
- Loss of fine control caused by optimizers reordering code

Floating-point issues

- Limits of accuracy, decimals not representable in binary
- (FAQ 7.31...)
- Deeper issue: knowledge of bit-level storage and hardware
- IEEE standards
- FP exceptions
- Loss of fine control caused by optimizers reordering code

Floating-point issues

- Limits of accuracy, decimals not representable in binary
- (FAQ 7.31...)
- Deeper issue: knowledge of bit-level storage and hardware
- IEEE standards
- FP exceptions
- Loss of fine control caused by optimizers reordering code

Floating-point issues

- Limits of accuracy, decimals not representable in binary
- (FAQ 7.31...)
- Deeper issue: knowledge of bit-level storage and hardware
- IEEE standards
- FP exceptions
- Loss of fine control caused by optimizers reordering code

Floating-point issues

- Limits of accuracy, decimals not representable in binary
- (FAQ 7.31...)
- Deeper issue: knowledge of bit-level storage and hardware
- IEEE standards
- FP exceptions
- Loss of fine control caused by optimizers reordering code

C, Fortran

- Structure of compiled languages
- Modular programs, linking,.libraries
- The C preprocessor
- Calling conventions

C, Fortran

- Structure of compiled languages
- Modular programs, linking,.libraries
- The C preprocessor
- Calling conventions

C, Fortran

- Structure of compiled languages
- Modular programs, linking,.libraries
- The C preprocessor
- Calling conventions

C, Fortran

- Structure of compiled languages
- Modular programs, linking,.libraries
- The C preprocessor
- Calling conventions

Interfaces to C and Fortran

- Access macros
- Some level of knowledge about the evaluator and internal storage of code
- Classical LISP implementation CAR/CDR/CONS
- Garbage collection and PROTECT
- The “tree” of objects that do not need protection

Interfaces to C and Fortran

- Access macros
- Some level of knowledge about the evaluator and internal storage of code
- Classical LISP implementation CAR/CDR/CONS
- Garbage collection and PROTECT
- The "tree" of objects that do not need protection

Interfaces to C and Fortran

- Access macros
- Some level of knowledge about the evaluator and internal storage of code
- Classical LISP implementation CAR/CDR/CONS
- Garbage collection and PROTECT
- The “tree” of objects that do not need protection

Interfaces to C and Fortran

- Access macros
- Some level of knowledge about the evaluator and internal storage of code
- Classical LISP implementation CAR/CDR/CONS
- Garbage collection and PROTECT
- The “tree” of objects that do not need protection

Interfaces to C and Fortran

- Access macros
- Some level of knowledge about the evaluator and internal storage of code
- Classical LISP implementation CAR/CDR/CONS
- Garbage collection and PROTECT
- The “tree” of objects that do not need protection

Algorithms and numerics

- Error sensitivity, e.g. SVD vs $(X'X)^{-1}$
- Computational complexity
- Memory consumption
- BLAS issues, CPU architecture

Algorithms and numerics

- Error sensitivity, e.g. SVD vs $(X'X)^{-1}$
- Computational complexity
- Memory consumption
- BLAS issues, CPU architecture

Algorithms and numerics

- Error sensitivity, e.g. SVD vs $(X'X)^{-1}$
- Computational complexity
- Memory consumption
- BLAS issues, CPU architecture

Algorithms and numerics

- Error sensitivity, e.g. SVD vs $(X'X)^{-1}$
- Computational complexity
- Memory consumption
- BLAS issues, CPU architecture

Markup languages

- Need it for Rd format files
- HTML, LaTeX, XML
- General idea that text is a computable quantity
- . . . and that higher-level structure is beneficial

Markup languages

- Need it for Rd format files
- HTML, LaTeX, XML
- General idea that text is a computable quantity
- . . . and that higher-level structure is beneficial

Markup languages

- Need it for Rd format files
- HTML, LaTeX, XML
- General idea that text is a computable quantity
- . . . and that higher-level structure is beneficial

Markup languages

- Need it for Rd format files
- HTML, LaTeX, XML
- General idea that text is a computable quantity
- . . . and that higher-level structure is beneficial

Programming language taxonomy

- (“Lots of quaintly named little languages”)
- Compiled vs. interpreted languages
- Late and early binding
- OOP concepts
- Lazy evaluation
- A better theoretical overview should help explaining why R sometimes behaves “strangely”

Programming language taxonomy

- ("Lots of quaintly named little languages")
- Compiled vs. interpreted languages
- Late and early binding
- OOP concepts
- Lazy evaluation
- A better theoretical overview should help explaining why R sometimes behaves "strangely"

Programming language taxonomy

- (“Lots of quaintly named little languages”)
- Compiled vs. interpreted languages
- Late and early binding
- OOP concepts
- Lazy evaluation
- A better theoretical overview should help explaining why R sometimes behaves “strangely”

Programming language taxonomy

- (“Lots of quaintly named little languages”)
- Compiled vs. interpreted languages
- Late and early binding
- OOP concepts
- Lazy evaluation
- A better theoretical overview should help explaining why R sometimes behaves “strangely”

Programming language taxonomy

- (“Lots of quaintly named little languages”)
- Compiled vs. interpreted languages
- Late and early binding
- OOP concepts
- Lazy evaluation
- A better theoretical overview should help explaining why R sometimes behaves “strangely”

Programming language taxonomy

- (“Lots of quaintly named little languages”)
- Compiled vs. interpreted languages
- Late and early binding
- OOP concepts
- Lazy evaluation
- A better theoretical overview should help explaining why R sometimes behaves “strangely”

R behaving badly

```
x <- 8
ll <- BinomialLikelihood(x, 20)
x <- 2
curve(ll)
x <- 15
curve(ll)
```

With an unfortunate coding of `BinomialLikelihood`, this gives the curve for BinomialLikelihood(2, 20) twice!

R behaving badly

```
x <- 8
ll <- BinomialLikelihood(x, 20)
x <- 2
curve(ll)
x <- 15
curve(ll)
```

With an unfortunate coding of `BinomialLikelihood`, this gives the curve for BinomialLikelihood(2, 20) twice!

Toolchains

- A group of problems relates to lack of knowledge about basic programs in the OS (or in Rtools)
- Compiler, linker, libraries
- (And how to install them when they are not there)
- Makefiles
- Scripts (Perl, shell)

Toolchains

- A group of problems relates to lack of knowledge about basic programs in the OS (or in Rtools)
- Compiler, linker, libraries
- (And how to install them when they are not there)
- Makefiles
- Scripts (Perl, shell)

Toolchains

- A group of problems relates to lack of knowledge about basic programs in the OS (or in Rtools)
- Compiler, linker, libraries
- (And how to install them when they are not there)
- Makefiles
- Scripts (Perl, shell)

Toolchains

- A group of problems relates to lack of knowledge about basic programs in the OS (or in Rtools)
- Compiler, linker, libraries
- (And how to install them when they are not there)
- Makefiles
- Scripts (Perl, shell)

Toolchains

- A group of problems relates to lack of knowledge about basic programs in the OS (or in Rtools)
- Compiler, linker, libraries
- (And how to install them when they are not there)
- Makefiles
- Scripts (Perl, shell)

So what to do about it?

- We cannot reasonably stuff a major part of theoretical computer science into a stat/maths curriculum
- Project-based studying lets students satisfy their own needs, but it has the same issue as teaching: The sudden need for a large amount of knowledge in s short time
- It may well be the case that we need to rethink topics as part of a somewhat longer story, e.g. text processing, then lexical analysis, then parsing, then CAR et al.
- However, some topics, e.g. C programming, are quite clearly delineated and there is probably no way around teaching them as an independent (sub-)course

So what to do about it?

- We cannot reasonably stuff a major part of theoretical computer science into a stat/maths curriculum
- Project-based studying lets students satisfy their own needs, but it has the same issue as teaching: The sudden need for a large amount of knowledge in s short time
- It may well be the case that we need to rethink topics as part of a somewhat longer story, e.g. text processing, then lexical analysis, then parsing, then CAR et al.
- However, some topics, e.g. C programming, are quite clearly delineated and there is probably no way around teaching them as an independent (sub-)course

So what to do about it?

- We cannot reasonably stuff a major part of theoretical computer science into a stat/maths curriculum
- Project-based studying lets students satisfy their own needs, but it has the same issue as teaching: The sudden need for a large amount of knowledge in s short time
- It may well be the case that we need to rethink topics as part of a somewhat longer story, e.g. text processing, then lexical analysis, then parsing, then CAR et al.
- However, some topics, e.g. C programming, are quite clearly delineated and there is probably no way around teaching them as an independent (sub-)course

So what to do about it?

- We cannot reasonably stuff a major part of theoretical computer science into a stat/maths curriculum
- Project-based studying lets students satisfy their own needs, but it has the same issue as teaching: The sudden need for a large amount of knowledge in s short time
- It may well be the case that we need to rethink topics as part of a somewhat longer story, e.g. text processing, then lexical analysis, then parsing, then CAR et al.
- However, some topics, e.g. C programming, are quite clearly delineated and there is probably no way around teaching them as an independent (sub-)course

Summary

- R came out of a “historical coincidence” where a number of people turned out to have both similar and complementary abilities, in areas that were not actually being taught in any systematic fashion
- The challenge at this point in time is to formalize and systematize these abilities in a way that can be taught at a general level
- Doing so is essential for the continued development of R and statistical computing in general

Summary

- R came out of a “historical coincidence” where a number of people turned out to have both similar and complementary abilities, in areas that were not actually being taught in any systematic fashion
- The challenge at this point in time is to formalize and systematize these abilities in a way that can be taught at a general level
- Doing so is essential for the continued development of R and statistical computing in general

Summary

- R came out of a "historical coincidence" where a number of people turned out to have both similar and complementary abilities, in areas that were not actually being taught in any systematic fashion
- The challenge at this point in time is to formalize and systematize these abilities in a way that can be taught at a general level
- Doing so is essential for the continued development of R and statistical computing in general