Programming Languages for Data Analysis

Douglas Bates
Dept. of Statistics
U. of Wisconsin-Madison

July 14, 2017

For the impatient

For the impatient

- The final part of this talk will be about the Julia programming language

For the impatient

- The final part of this talk will be about the Julia programming language
- For an overview of Julia for econometrics, see quantecon.org

For the impatient

- The final part of this talk will be about the Julia programming language
- For an overview of Julia for econometrics, see quantecon.org
- The lectures tab provides a Python version and a Julia version

For the impatient

- The final part of this talk will be about the Julia programming language
- For an overview of Julia for econometrics, see quantecon.org
- The lectures tab provides a Python version and a Julia version
- The About tab is a very good discussion of why these two languages were chosen.

More Julia resources

More Julia resources

- The original developers formed a company, Julia Computing, to provide consulting services and oversee the development of the language.

More Julia resources

- The original developers formed a company, Julia Computing, to provide consulting services and oversee the development of the language.
- The case studies section is of particular interest

More Julia resources

- The original developers formed a company, Julia Computing, to provide consulting services and oversee the development of the language.
- The case studies section is of particular interest
- Registered Julia packages are at pkg.julialang.org

More Julia resources

- The original developers formed a company, Julia Computing, to provide consulting services and oversee the development of the language.
- The case studies section is of particular interest
- Registered Julia packages are at pkg.julialang.org
- Note that Julia packages are git repositories, usually housed on github.com

More Julia resources

- The original developers formed a company, Julia Computing, to provide consulting services and oversee the development of the language.
- The case studies section is of particular interest
- Registered Julia packages are at pkg.julialang.org
- Note that Julia packages are git repositories, usually housed on github.com
- See also Julia Package Ecosystem Pulse

More Julia resources

- The original developers formed a company, Julia Computing, to provide consulting services and oversee the development of the language.
- The case studies section is of particular interest
- Registered Julia packages are at pkg.julialang.org
- Note that Julia packages are git repositories, usually housed on github.com
- See also Julia Package Ecosystem Pulse
- Documentation for the language itself is at docs.julialang.org

Language shapes the way we think - Benjamin Lee Whorf

Language shapes the way we think - Benjamin Lee Whorf

- I have been doing data analysis programming for a long time

Language shapes the way we think - Benjamin Lee Whorf

- I have been doing data analysis programming for a long time
  - took my only computing course in 1967 - didn't like it

Language shapes the way we think - Benjamin Lee Whorf

- I have been doing data analysis programming for a long time
  - took my only computing course in 1967 - didn't like it
  - in the 70's wrote Fortran code that I hope no one ever discovers

Language shapes the way we think - Benjamin Lee Whorf

- I have been doing data analysis programming for a long time
  - took my only computing course in 1967 - didn't like it
  - in the 70's wrote Fortran code that I hope no one ever discovers
  - late 70's read Kernighan and his co-authors, saw programming differently

Language shapes the way we think - Benjamin Lee Whorf

- I have been doing data analysis programming for a long time
  - took my only computing course in 1967 - didn't like it
  - in the 70's wrote Fortran code that I hope no one ever discovers
  - late 70's read Kernighan and his co-authors, saw programming differently
  - was able to use sophisticated languages, almost by accident, in thesis research

Language shapes the way we think - Benjamin Lee Whorf

- I have been doing data analysis programming for a long time
  - took my only computing course in 1967 - didn't like it
  - in the 70's wrote Fortran code that I hope no one ever discovers
  - late 70's read Kernighan and his co-authors, saw programming differently
  - was able to use sophisticated languages, almost by accident, in thesis research
  - some SPSS and SAS use, Minitab for teaching

Language shapes the way we think - Benjamin Lee Whorf

- I have been doing data analysis programming for a long time
  - took my only computing course in 1967 - didn't like it
  - in the 70's wrote Fortran code that I hope no one ever discovers
  - late 70's read Kernighan and his co-authors, saw programming differently
  - was able to use sophisticated languages, almost by accident, in thesis research
  - some SPSS and SAS use, Minitab for teaching
  - heard about the work on S by John Chambers and co. at Bell Labs

What was different about S?

What was different about S?

- an interactive language (REPL - read-eval-print-loop)

What was different about S?

- an interactive language (REPL - read-eval-print-loop)
- functional language (in the sense of defining and calling functions)

What was different about S?

- an interactive language (REPL - read-eval-print-loop)
- functional language (in the sense of defining and calling functions)
- heterogeneous, self-describing, recursive, extensible data structures

What was different about S?

- an interactive language (REPL - read-eval-print-loop)
- functional language (in the sense of defining and calling functions)
- heterogeneous, self-describing, recursive, extensible data structures
  - contrast with flat-files structures in SPSS, SAS; vectors (columns) in Minitab

What was different about S?

- an interactive language (REPL - read-eval-print-loop)
- functional language (in the sense of defining and calling functions)
- heterogeneous, self-describing, recursive, extensible data structures
  - contrast with flat-files structures in SPSS, SAS; vectors (columns) in Minitab
- random-access memory based, not a filter

What was different about S?

- an interactive language (REPL - read-eval-print-loop)
- functional language (in the sense of defining and calling functions)
- heterogeneous, self-describing, recursive, extensible data structures
  - contrast with flat-files structures in SPSS, SAS; vectors (columns) in Minitab
- random-access memory based, not a filter
  - required a "second megabyte of memory"

What was different about S?

- an interactive language (REPL - read-eval-print-loop)
- functional language (in the sense of defining and calling functions)
- heterogeneous, self-describing, recursive, extensible data structures
  - contrast with flat-files structures in SPSS, SAS; vectors (columns) in Minitab
- random-access memory based, not a filter
  - required a "second megabyte of memory"
- explicit interfaces to compiled code

What was different about S?

- an interactive language (REPL - read-eval-print-loop)
- functional language (in the sense of defining and calling functions)
- heterogeneous, self-describing, recursive, extensible data structures
  - contrast with flat-files structures in SPSS, SAS; vectors (columns) in Minitab
- random-access memory based, not a filter
  - required a "second megabyte of memory"
- explicit interfaces to compiled code
  - original versions were more of a wrapper around numerical and graphics code bases

What was the same with S?

What was the same with S?

- explicitly designed for data analysis and graphics

What was the same with S?

- explicitly designed for data analysis and graphics
- provision for missing data was built in at a very low level

What was the same with S?

- explicitly designed for data analysis and graphics
- provision for missing data was built in at a very low level
- internally, data structures were always vectors, possibly with "attributes"

What was the same with S?

- explicitly designed for data analysis and graphics
- provision for missing data was built in at a very low level
- internally, data structures were always vectors, possibly with "attributes"
- "lists" are actually vectors of pointers, not linked lists as in Lisp

What was the same with S?

- explicitly designed for data analysis and graphics
- provision for missing data was built in at a very low level
- internally, data structures were always vectors, possibly with "attributes"
- "lists" are actually vectors of pointers, not linked lists as in Lisp
- "semi-proprietary" software

What was the same with S?

- explicitly designed for data analysis and graphics
- provision for missing data was built in at a very low level
- internally, data structures were always vectors, possibly with "attributes"
- "lists" are actually vectors of pointers, not linked lists as in Lisp
- "semi-proprietary" software
  - AT&T couldn't market S (or the Unix operating system)
  - some universities became "beta-test sites"
  - U. of Washington Stats. Dept. spun off "StatSci" and marketed S-PLUS

The 90's and the birth of R

The 90's and the birth of R

- Open-Source software gained traction (perl, python, emacs, GNU/Linux, MySQL, PostgreSQL) in some areas

The 90's and the birth of R

- Open-Source software gained traction (perl, python, emacs, GNU/Linux, MySQL, PostgreSQL) in some areas
  - Servers were often characterized as LAMP (Linux, Apache, MySQL, Perl/Python)
  - First releases of both Python and Linux were in 1991

The 90's and the birth of R

- Open-Source software gained traction (perl, python, emacs, GNU/Linux, MySQL, PostgreSQL) in some areas
  - Servers were often characterized as LAMP (Linux, Apache, MySQL, Perl/Python)
  - First releases of both Python and Linux were in 1991
- mid-90's Ross Ihaka and Robert Gentleman started work on a language "not-unlike S"

The 90's and the birth of R

- Open-Source software gained traction (perl, python, emacs, GNU/Linux, MySQL, PostgreSQL) in some areas
  - Servers were often characterized as LAMP (Linux, Apache, MySQL, Perl/Python)
  - First releases of both Python and Linux were in 1991
- mid-90's Ross Ihaka and Robert Gentleman started work on a language "not-unlike S"
  - S-PLUS had been ported to DOS/Windows but not to Macintosh
  - essentially a "clean room" reimplementation of the S language
  - Martin Mächler contributed so many patches they gave him an account
  - Martin encouraged release under the GPL
  - others joined the party, 1997 R-Core was formed.

The 90's and CRAN

The 90's and CRAN

- Kurt Hornik and Fritz Leisch, then at TU-Wien, created the "Comprehensive R Archive Network" (**CRAN**)

The 90's and CRAN

- Kurt Hornik and Fritz Leisch, then at TU-Wien, created the "Comprehensive R Archive Network" (**CRAN**)
  - patterned after CTAN (TeX) and CPAN (Perl) archive networks

The 90's and CRAN

- Kurt Hornik and Fritz Leisch, then at TU-Wien, created the "Comprehensive R Archive Network" (**CRAN**)
  - patterned after CTAN (TeX) and CPAN (Perl) archive networks
- Encouraged users to become developers and researchers to provide a reference implementation of their methods

The 90's and CRAN

- Kurt Hornik and Fritz Leisch, then at TU-Wien, created the "Comprehensive R Archive Network" (**CRAN**)
  - patterned after CTAN (TeX) and CPAN (Perl) archive networks
- Encouraged users to become developers and researchers to provide a reference implementation of their methods
- Promoted the development of package standards, testing etc.

The 90's and CRAN

- Kurt Hornik and Fritz Leisch, then at TU-Wien, created the "Comprehensive R Archive Network" (**CRAN**)
  - patterned after CTAN (TeX) and CPAN (Perl) archive networks
- Encouraged users to become developers and researchers to provide a reference implementation of their methods
- Promoted the development of package standards, testing etc.
- One important facility added later (Uwe Ligges) was "Win-builder" to create binary Windows packages

The 90's and CRAN

- Kurt Hornik and Fritz Leisch, then at TU-Wien, created the "Comprehensive R Archive Network" (**CRAN**)
  - patterned after CTAN (TeX) and CPAN (Perl) archive networks
- Encouraged users to become developers and researchers to provide a reference implementation of their methods
- Promoted the development of package standards, testing etc.
- One important facility added later (Uwe Ligges) was "Win-builder" to create binary Windows packages
  - most developers were on Linux, most users on Windows

The aughts - R fourishes

The aughts - R fourishes

- Initially R was considered by many to be a "poor man's S-PLUS"

The aughts - R fourishes

- Initially R was considered by many to be a "poor man's S-PLUS"
  - that is "Freeware" couldn't possibly be as good as commercial software

The aughts - R fourishes

- Initially R was considered by many to be a "poor man's S-PLUS"
  - that is "Freeware" couldn't possibly be as good as commercial software
- Eventually the open-source model was recognized as producing high-quality code

The aughts - R fourishes

- Initially R was considered by many to be a "poor man's S-PLUS"
  - that is "Freeware" couldn't possibly be as good as commercial software
- Eventually the open-source model was recognized as producing high-quality code
  - Eric S. Raymond, "Given enough eyeballs, all bugs are shallow."

The aughts - R fourishes

- Initially R was considered by many to be a "poor man's S-PLUS"
  - that is "Freeware" couldn't possibly be as good as commercial software
- Eventually the open-source model was recognized as producing high-quality code
  - Eric S. Raymond, "Given enough eyeballs, all bugs are shallow."
- R-1.0.0 was released on February 29, 2000 - the day that didn't exist

The aughts - R fourishes

- Initially R was considered by many to be a "poor man's S-PLUS"
  - that is "Freeware" couldn't possibly be as good as commercial software
- Eventually the open-source model was recognized as producing high-quality code
  - Eric S. Raymond, "Given enough eyeballs, all bugs are shallow."
- R-1.0.0 was released on February 29, 2000 - the day that didn't exist
- CRAN went from 10's to 100's to 1000's of packages

The aughts - R fourishes

- Initially R was considered by many to be a "poor man's S-PLUS"
  - that is "Freeware" couldn't possibly be as good as commercial software
- Eventually the open-source model was recognized as producing high-quality code
  - Eric S. Raymond, "Given enough eyeballs, all bugs are shallow."
- R-1.0.0 was released on February 29, 2000 - the day that didn't exist
- CRAN went from 10's to 100's to 1000's of packages
- useR! conferences started, *R Newsletter* later to become *R Journal* founded

The aughts - R fourishes

- Initially R was considered by many to be a "poor man's S-PLUS"
  - that is "Freeware" couldn't possibly be as good as commercial software
- Eventually the open-source model was recognized as producing high-quality code
  - Eric S. Raymond, "Given enough eyeballs, all bugs are shallow."
- R-1.0.0 was released on February 29, 2000 - the day that didn't exist
- CRAN went from 10's to 100's to 1000's of packages
- useR! conferences started, *R Newsletter* later to become *R Journal* founded
- papers, books, online resources, became available

The aughts - other Open-Source languages enter the mix

The aughts - other Open-Source languages enter the mix

- S and R were designed for data analysis and graphics, not general purpose programming

The aughts - other Open-Source languages enter the mix

- S and R were designed for data analysis and graphics, not general purpose programming
  - as Ross said, "we thought maybe a couple of hundred people total would use it"

The aughts - other Open-Source languages enter the mix

- S and R were designed for data analysis and graphics, not general purpose programming
  - as Ross said, "we thought maybe a couple of hundred people total would use it"
- every language is a trade-off between pre-defined, high-level operations and low-level building blocks

The aughts - other Open-Source languages enter the mix

- S and R were designed for data analysis and graphics, not general purpose programming
  - as Ross said, "we thought maybe a couple of hundred people total would use it"
- every language is a trade-off between pre-defined, high-level operations and low-level building blocks
  - R is oriented to high-level tools: vectors, matrices, data frames are okay; scalars, loops, low-level logic not so much
  - the number of internal data types is surprisingly small (32-bit integers, 64-bit floats and character strings)

The aughts - other Open-Source languages enter the mix

- S and R were designed for data analysis and graphics, not general purpose programming
  - as Ross said, "we thought maybe a couple of hundred people total would use it"
- every language is a trade-off between pre-defined, high-level operations and low-level building blocks
  - R is oriented to high-level tools: vectors, matrices, data frames are okay; scalars, loops, low-level logic not so much
  - the number of internal data types is surprisingly small (32-bit integers, 64-bit floats and character strings)
- languages like Python provided more flexibility but without the data science specific structures

The aughts - other Open-Source languages enter the mix

- S and R were designed for data analysis and graphics, not general purpose programming
  - as Ross said, "we thought maybe a couple of hundred people total would use it"
- every language is a trade-off between pre-defined, high-level operations and low-level building blocks
  - R is oriented to high-level tools: vectors, matrices, data frames are okay; scalars, loops, low-level logic not so much
  - the number of internal data types is surprisingly small (32-bit integers, 64-bit floats and character strings)
- languages like Python provided more flexibility but without the data science specific structures
  - required add-ons like numpy, pandas, etc.

The teens and enhancements like RStudio and Rcpp

The teens and enhancements like RStudio and Rcpp

- RStudio - definitive IDE for R

The teens and enhancements like RStudio and Rcpp

- RStudio - definitive IDE for R
  - Introduced in 2011
  - Freely available for personal use - important for teaching
  - J.J. Allaire brought Hadley Wickham on as Chief Scientist. More hires followed.
  - RStudio is now much more than an IDE - the most creative group in the R community
  - Fast moving group in contrast to more conservative R core
- Rcpp - "seamless" integration of R and C++

The teens and enhancements like RStudio and Rcpp

- RStudio - definitive IDE for R
  - Introduced in 2011
  - Freely available for personal use - important for teaching
  - J.J. Allaire brought Hadley Wickham on as Chief Scientist. More hires followed.
  - RStudio is now much more than an IDE - the most creative group in the R community
  - Fast moving group in contrast to more conservative R core
- Rcpp - "seamless" integration of R and C++
  - Integrate ease of use of R with speed of a compiled language relatively painlessly
  - But - C++
  - There is a fundamental mismatch in the languages (static vs dynamic)

My adventures fitting mixed-effects models - nlme

My adventures fitting mixed-effects models - nlme

- In the mid-90's I was dragged into research on models with fixed- and random-effects, called *mixed-effects models* in Statistics

My adventures fitting mixed-effects models - nlme

- In the mid-90's I was dragged into research on models with fixed- and random-effects, called *mixed-effects models* in Statistics
- José Pinheiro and I wrote a package for S called *nlme* to fit linear and nonlinear mixed-effects models

My adventures fitting mixed-effects models - nlme

- In the mid-90's I was dragged into research on models with fixed- and random-effects, called *mixed-effects models* in Statistics
- José Pinheiro and I wrote a package for S called *nlme* to fit linear and nonlinear mixed-effects models
  - One of the first packages ported to R.
  - Also a Springer book published in 2000.
  - About 14,000 lines of R code and 3,000 lines of C code
  - Used S3 classes and methods.
  - Compiled code called through `.C` and `.Call`

My adventures fitting mixed-effects models - nlme

- In the mid-90's I was dragged into research on models with fixed- and random-effects, called *mixed-effects models* in Statistics
- José Pinheiro and I wrote a package for S called *nlme* to fit linear and nonlinear mixed-effects models
  - One of the first packages ported to R.
  - Also a Springer book published in 2000.
  - About 14,000 lines of R code and 3,000 lines of C code
  - Used S3 classes and methods.
  - Compiled code called through `.C` and `.Call`
- As soon as it was released I started thinking there were better ways to do this

My adventures fitting mixed-effects models - lme4

My adventures fitting mixed-effects models - lme4

- In the aughts Martin Mächler and I started work on mixed-effects models using S4 classes and methods

My adventures fitting mixed-effects models - lme4

- In the aughts Martin Mächler and I started work on mixed-effects models using S4 classes and methods
  - Incorporated sparse matrix methods for which we needed to access more linear algebra software
  - Initially the linear algebra was part of **lme4**, later split to the **Matrix** package
  - Later **Rcpp**'ized **Eigen** (C++ matrix package) as **RcppEigen**
  - Needed better optimization software for difficult case
  - Ended up using S3, S4, reference classes, ...
  - In the teens, Ben Bolker became the primary maintainer.
  - Now about 6,000 lines of R code, 2700 lines of C/C++ code
  - Fits both linear mixed models and generalized linear mixed models
  - Faster and more reliable than `nlme` although coverage of two packages not identical
  - Can fit models with crossed random effects, such as for Subject/Item data

Why not stop at lme4?

Why not stop at lme4?

- Code is unwieldy and opaque - difficult to maintain and/or extend

Why not stop at lme4?

- Code is unwieldy and opaque - difficult to maintain and/or extend
- Still problems with

Why not stop at lme4?

- Code is unwieldy and opaque - difficult to maintain and/or extend
- Still problems with
  - optimizers
  - linear algebra
  - using large objects in iterative algorithms
  - the static/dynamic barrier in general

Why not stop at lme4?

- Code is unwieldy and opaque - difficult to maintain and/or extend
- Still problems with
  - optimizers
  - linear algebra
  - using large objects in iterative algorithms
  - the static/dynamic barrier in general
- Many of these issues are the result of S/R initial design decisions
  - Lazy evaluation
  - Functional semantics

The Julia language

The Julia language

- Julia was developed by very skilled programmers explicitly for quantitative work

The Julia language

- Julia was developed by very skilled programmers explicitly for quantitative work
- First public release in 2011 so relatively immature. Still working on important infrastructure (e.g. DataFrames)

The Julia language

- Julia was developed by very skilled programmers explicitly for quantitative work
- First public release in 2011 so relatively immature. Still working on important infrastructure (e.g. DataFrames)
- Developers has extensive experience with many languages (Python, R, Matlab, ...), but wanted more
- Functions/methods, classes (a.k.a, types) as in R but much more rigorous
- Flexibility and speed can be achieved with Just-In-Time compilation (JIT)

MixedModels package

- Similar in coverage to the lme4 package
- About 2700 lines of Julia code
- Algorithmic enhancements relative to lme4
- Usually about 10x faster than lme4