Some Thoughts on Languages for Statistical Computing and Graphics

Luke Tierney

Department of Statistics & Actuarial Science
University of Iowa

July 28, 2019

Graphics to Language

- New ideas in 1980s for exploring data with dynamic graphics:
  - Point cloud rotation;
  - Linked brushing in scatterplot matrices.
- Intriguing, but used specialized/expensive hardware.
- Apple Macintosh provided new opportunities.
  - First approach: create some stand-alone applications
- Challenges
  - Need to get data into the applications.
  - How to experiment without having to rebuild the application.
- Natural approach:
  - Embed graphics in a language.
  - Allow customization using the language.

Graphics to Language

- New ideas in 1980s for exploring data with dynamic graphics:
  - Point cloud rotation;
  - Linked brushing in scatterplot matrices.
- Intriguing, but used specialized/expensive hardware.
- Apple Macintosh provided new opportunities.
  - First approach: create some stand-alone applications
- Challenges
  - Need to get data into the applications.
  - How to experiment without having to rebuild the application.
- Natural approach:
  - Embed graphics in a language.
  - Allow customization using the language.

Graphics to Language

- New ideas in 1980s for exploring data with dynamic graphics:
  - Point cloud rotation;
  - Linked brushing in scatterplot matrices.
- Intriguing, but used specialized/expensive hardware.
- Apple Macintosh provided new opportunities.
  - First approach: create some stand-alone applications
- Challenges
  - Need to get data into the applications.
  - How to experiment without having to rebuild the application.
- Natural approach:
  - Embed graphics in a language.
  - Allow customization using the language.

Graphics to Language

- New ideas in 1980s for exploring data with dynamic graphics:
  - Point cloud rotation;
  - Linked brushing in scatterplot matrices.
- Intriguing, but used specialized/expensive hardware.
- Apple Macintosh provided new opportunities.
  - First approach: create some stand-alone applications
- Challenges
  - Need to get data into the applications.
  - How to experiment without having to rebuild the application.
- Natural approach:
  - Embed graphics in a language.
  - Allow customization using the language.

Graphics to Language

- New ideas in 1980s for exploring data with dynamic graphics:
  - Point cloud rotation;
  - Linked brushing in scatterplot matrices.
- Intriguing, but used specialized/expensive hardware.
- Apple Macintosh provided new opportunities.
  - First approach: create some stand-alone applications
- Challenges
  - Need to get data into the applications.
  - How to experiment without having to rebuild the application.
- Natural approach:
  - Embed graphics in a language.
  - Allow customization using the language.

Graphics to Language

- New ideas in 1980s for exploring data with dynamic graphics:
  - Point cloud rotation;
  - Linked brushing in scatterplot matrices.
- Intriguing, but used specialized/expensive hardware.
- Apple Macintosh provided new opportunities.
  - First approach: create some stand-alone applications
- Challenges
  - Need to get data into the applications.
  - How to experiment without having to rebuild the application.
- Natural approach:
  - Embed graphics in a language.
  - Allow customization using the language.

Graphics to Language

- New ideas in 1980s for exploring data with dynamic graphics:
  - Point cloud rotation;
  - Linked brushing in scatterplot matrices.
- Intriguing, but used specialized/expensive hardware.
- Apple Macintosh provided new opportunities.
  - First approach: create some stand-alone applications
- Challenges
  - Need to get data into the applications.
  - How to experiment without having to rebuild the application.
- Natural approach:
  - Embed graphics in a language.
  - Allow customization using the language.

Graphics to Language

- New ideas in 1980s for exploring data with dynamic graphics:
  - Point cloud rotation;
  - Linked brushing in scatterplot matrices.
- Intriguing, but used specialized/expensive hardware.
- Apple Macintosh provided new opportunities.
  - First approach: create some stand-alone applications
- Challenges
  - Need to get data into the applications.
  - How to experiment without having to rebuild the application.
- Natural approach:
  - Embed graphics in a language.
  - Allow customization using the language.

Graphics to Language

- New ideas in 1980s for exploring data with dynamic graphics:
  - Point cloud rotation;
  - Linked brushing in scatterplot matrices.
- Intriguing, but used specialized/expensive hardware.
- Apple Macintosh provided new opportunities.
  - First approach: create some stand-alone applications
- Challenges
  - Need to get data into the applications.
  - How to experiment without having to rebuild the application.
- Natural approach:
  - Embed graphics in a language.
  - Allow customization using the language.

Graphics to Language

- New ideas in 1980s for exploring data with dynamic graphics:
    - Point cloud rotation;
    - Linked brushing in scatterplot matrices.
- Intriguing, but used specialized/expensive hardware.
- Apple Macintosh provided new opportunities.
    - First approach: create some stand-alone applications
- Challenges
    - Need to get data into the applications.
    - How to experiment without having to rebuild the application.
- Natural approach:
    - Embed graphics in a language.
    - Allow customization using the language.

Graphics to Language

- New ideas in 1980s for exploring data with dynamic graphics:
    - Point cloud rotation;
    - Linked brushing in scatterplot matrices.
- Intriguing, but used specialized/expensive hardware.
- Apple Macintosh provided new opportunities.
    - First approach: create some stand-alone applications
- Challenges
    - Need to get data into the applications.
    - How to experiment without having to rebuild the application.
- Natural approach:
    - Embed graphics in a language.
    - Allow customization using the language.

Graphics to Language

- New ideas in 1980s for exploring data with dynamic graphics:
  - Point cloud rotation;
  - Linked brushing in scatterplot matrices.
- Intriguing, but used specialized/expensive hardware.
- Apple Macintosh provided new opportunities.
  - First approach: create some stand-alone applications
- Challenges
  - Need to get data into the applications.
  - How to experiment without having to rebuild the application.
- Natural approach:
  - Embed graphics in a language.
  - Allow customization using the language.

Graphics to Language

- Build on an existing general-purpose language or design a new one?
- Some possible general purpose language choices at the time:
  - APL
  - Lisp
- Lisp seemed a good choice
  - Available C implementations for modification.
  - Used by a number of other researchers.
- XLISP was used as the basis for Lisp-Stat.
  - XLISP provided a subset of Common Lisp.
  - More Common Lisp features were added to Lisp-Stat.

Graphics to Language

- Build on an existing general-purpose language or design a new one?
- Some possible general purpose language choices at the time:
  - APL
  - Lisp
- Lisp seemed a good choice
  - Available C implementations for modification.
  - Used by a number of other researchers.
- XLISP was used as the basis for Lisp-Stat.
  - XLISP provided a subset of Common Lisp.
  - More Common Lisp features were added to Lisp-Stat.

Graphics to Language

- Build on an existing general-purpose language or design a new one?
- Some possible general purpose language choices at the time:
  - APL
  - Lisp
- Lisp seemed a good choice
  - Available C implementations for modification.
  - Used by a number of other researchers.
- XLISP was used as the basis for Lisp-Stat.
  - XLISP provided a subset of Common Lisp.
  - More Common Lisp features were added to Lisp-Stat.

Graphics to Language

- Build on an existing general-purpose language or design a new one?
- Some possible general purpose language choices at the time:
  - APL
  - Lisp
- Lisp seemed a good choice
  - Available C implementations for modification.
  - Used by a number of other researchers.
- XLISP was used as the basis for Lisp-Stat.
  - XLISP provided a subset of Common Lisp.
  - More Common Lisp features were added to Lisp-Stat.

Graphics to Language

- Build on an existing general-purpose language or design a new one?
- Some possible general purpose language choices at the time:
    - APL
    - Lisp
- Lisp seemed a good choice
    - Available C implementations for modification.
    - Used by a number of other researchers.
- XLISP was used as the basis for Lisp-Stat.
    - XLISP provided a subset of Common Lisp.
    - More Common Lisp features were added to Lisp-Stat.

Graphics to Language

- Build on an existing general-purpose language or design a new one?
- Some possible general purpose language choices at the time:
  - APL
  - Lisp
- Lisp seemed a good choice
  - Available C implementations for modification.
  - Used by a number of other researchers.
- XLISP was used as the basis for Lisp-Stat.
  - XLISP provided a subset of Common Lisp.
  - More Common Lisp features were added to Lisp-Stat.

Graphics to Language

- Build on an existing general-purpose language or design a new one?
- Some possible general purpose language choices at the time:
  - APL
  - Lisp
- Lisp seemed a good choice
  - Available C implementations for modification.
  - Used by a number of other researchers.
- XLISP was used as the basis for Lisp-Stat.
  - XLISP provided a subset of Common Lisp.
  - More Common Lisp features were added to Lisp-Stat.

Graphics to Language

- Build on an existing general-purpose language or design a new one?
- Some possible general purpose language choices at the time:
  - APL
  - Lisp
- Lisp seemed a good choice
  - Available C implementations for modification.
  - Used by a number of other researchers.
- XLISP was used as the basis for Lisp-Stat.
  - XLISP provided a subset of Common Lisp.
  - More Common Lisp features were added to Lisp-Stat.

Graphics to Language

- Build on an existing general-purpose language or design a new one?
- Some possible general purpose language choices at the time:
  - APL
  - Lisp
- Lisp seemed a good choice
  - Available C implementations for modification.
  - Used by a number of other researchers.
- XLISP was used as the basis for Lisp-Stat.
  - XLISP provided a subset of Common Lisp.
  - More Common Lisp features were added to Lisp-Stat.

Graphics to Language

- Build on an existing general-purpose language or design a new one?
- Some possible general purpose language choices at the time:
    - APL
    - Lisp
- Lisp seemed a good choice
    - Available C implementations for modification.
    - Used by a number of other researchers.
- XLISP was used as the basis for Lisp-Stat.
    - XLISP provided a subset of Common Lisp.
    - More Common Lisp features were added to Lisp-Stat.

Useful Language Features

- Valuable features:
  - Vector-oriented operations.
  - First class functions.
  - Good error and warning handling.
  - A module system.
  - Framework for interfacing to lower-level languages.
  - Compiler to reduce the need for lower level languages.
- Supporting interactive graphics:
  - Object oriented programming framework.
  - Custom event handlers.
- Features not supported by Lisp-Stat
  - Missing value propagation.
  - Pass by value/immutable data objects.

Useful Language Features

- Valuable features:
  - Vector-oriented operations.
  - First class functions.
  - Good error and warning handling.
  - A module system.
  - Framework for interfacing to lower-level languages.
  - Compiler to reduce the need for lower level languages.
- Supporting interactive graphics:
  - Object oriented programming framework.
  - Custom event handlers.
- Features not supported by Lisp-Stat
  - Missing value propagation.
  - Pass by value/immutable data objects.

Useful Language Features

- Valuable features:
  - Vector-oriented operations.
  - First class functions.
  - Good error and warning handling.
  - A module system.
  - Framework for interfacing to lower-level languages.
  - Compiler to reduce the need for lower level languages.
- Supporting interactive graphics:
  - Object oriented programming framework.
  - Custom event handlers.
- Features not supported by Lisp-Stat
  - Missing value propagation.
  - Pass by value/immutable data objects.

Useful Language Features

- Valuable features:
  - Vector-oriented operations.
  - First class functions.
  - Good error and warning handling.
  - A module system.
  - Framework for interfacing to lower-level languages.
  - Compiler to reduce the need for lower level languages.
- Supporting interactive graphics:
  - Object oriented programming framework.
  - Custom event handlers.
- Features not supported by Lisp-Stat
  - Missing value propagation.
  - Pass by value/immutable data objects.

Useful Language Features

- Valuable features:
    - Vector-oriented operations.
    - First class functions.
    - Good error and warning handling.
    - A module system.
    - Framework for interfacing to lower-level languages.
    - Compiler to reduce the need for lower level languages.
- Supporting interactive graphics:
    - Object oriented programming framework.
    - Custom event handlers.
- Features not supported by Lisp-Stat
    - Missing value propagation.
    - Pass by value/immutable data objects.

Useful Language Features

- Valuable features:
  - Vector-oriented operations.
  - First class functions.
  - Good error and warning handling.
  - A module system.
  - Framework for interfacing to lower-level languages.
  - Compiler to reduce the need for lower level languages.
- Supporting interactive graphics:
  - Object oriented programming framework.
  - Custom event handlers.
- Features not supported by Lisp-Stat
  - Missing value propagation.
  - Pass by value/immutable data objects.

Useful Language Features

- Valuable features:
  - Vector-oriented operations.
  - First class functions.
  - Good error and warning handling.
  - A module system.
  - Framework for interfacing to lower-level languages.
  - Compiler to reduce the need for lower level languages.
- Supporting interactive graphics:
  - Object oriented programming framework.
  - Custom event handlers.
- Features not supported by Lisp-Stat
  - Missing value propagation.
  - Pass by value/immutable data objects.

Useful Language Features

- Valuable features:
  - Vector-oriented operations.
  - First class functions.
  - Good error and warning handling.
  - A module system.
  - Framework for interfacing to lower-level languages.
  - Compiler to reduce the need for lower level languages.
- Supporting interactive graphics:
  - Object oriented programming framework.
  - Custom event handlers.
- Features not supported by Lisp-Stat
  - Missing value propagation.
  - Pass by value/immutable data objects.

Useful Language Features

- Valuable features:
    - Vector-oriented operations.
    - First class functions.
    - Good error and warning handling.
    - A module system.
    - Framework for interfacing to lower-level languages.
    - Compiler to reduce the need for lower level languages.
- Supporting interactive graphics:
    - Object oriented programming framework.
    - Custom event handlers.
- Features not supported by Lisp-Stat
    - Missing value propagation.
    - Pass by value/immutable data objects.

Useful Language Features

- Valuable features:
  - Vector-oriented operations.
  - First class functions.
  - Good error and warning handling.
  - A module system.
  - Framework for interfacing to lower-level languages.
  - Compiler to reduce the need for lower level languages.
- Supporting interactive graphics:
  - Object oriented programming framework.
  - Custom event handlers.
- Features not supported by Lisp-Stat
  - Missing value propagation.
  - Pass by value/immutable data objects.

Useful Language Features

- Valuable features:
  - Vector-oriented operations.
  - First class functions.
  - Good error and warning handling.
  - A module system.
  - Framework for interfacing to lower-level languages.
  - Compiler to reduce the need for lower level languages.
- Supporting interactive graphics:
  - Object oriented programming framework.
  - Custom event handlers.
- Features not supported by Lisp-Stat
  - Missing value propagation.
  - Pass by value/immutable data objects.

Useful Language Features

- Valuable features:
  - Vector-oriented operations.
  - First class functions.
  - Good error and warning handling.
  - A module system.
  - Framework for interfacing to lower-level languages.
  - Compiler to reduce the need for lower level languages.
- Supporting interactive graphics:
  - Object oriented programming framework.
  - Custom event handlers.
- Features not supported by Lisp-Stat
  - Missing value propagation.
  - Pass by value/immutable data objects.

Useful Language Features

- Valuable features:
    - Vector-oriented operations.
    - First class functions.
    - Good error and warning handling.
    - A module system.
    - Framework for interfacing to lower-level languages.
    - Compiler to reduce the need for lower level languages.
- Supporting interactive graphics:
    - Object oriented programming framework.
    - Custom event handlers.
- Features not supported by Lisp-Stat
    - Missing value propagation.
    - Pass by value/immutable data objects.

Lisp(-Stat) Features in R

- R now incorporates Many features originally explored in Lisp-Stat.
    - Condition system for handling errors and warnings.
    - Name spaces.
    - Memory management framework.
    - Byte code compilation.
- One feature that is not supported:
    - Programmable dynamic graphics.

Lisp(-Stat) Features in R

- R now incorporates Many features originally explored in Lisp-Stat.
  - Condition system for handling errors and warnings.
  - Name spaces.
  - Memory management framework.
  - Byte code compilation.
- One feature that is not supported:
  - Programmable dynamic graphics.

Lisp(-Stat) Features in R

- R now incorporates Many features originally explored in Lisp-Stat.
  - Condition system for handling errors and warnings.
  - Name spaces.
  - Memory management framework.
  - Byte code compilation.
- One feature that is not supported:
  - Programmable dynamic graphics.

Lisp(-Stat) Features in R

- R now incorporates Many features originally explored in Lisp-Stat.
  - Condition system for handling errors and warnings.
  - Name spaces.
  - Memory management framework.
  - Byte code compilation.
- One feature that is not supported:
  - Programmable dynamic graphics.

Lisp(-Stat) Features in R

- R now incorporates Many features originally explored in Lisp-Stat.
  - Condition system for handling errors and warnings.
  - Name spaces.
  - Memory management framework.
  - Byte code compilation.
- One feature that is not supported:
  - Programmable dynamic graphics.

Lisp(-Stat) Features in R

- R now incorporates Many features originally explored in Lisp-Stat.
  - Condition system for handling errors and warnings.
  - Name spaces.
  - Memory management framework.
  - Byte code compilation.
- One feature that is not supported:
  - Programmable dynamic graphics.

Lisp(-Stat) Features in R

- R now incorporates Many features originally explored in Lisp-Stat.
  - Condition system for handling errors and warnings.
  - Name spaces.
  - Memory management framework.
  - Byte code compilation.
- One feature that is not supported:
  - Programmable dynamic graphics.

Language Challenges

- For simple tasks, graphical interfaces are often easier to learn.
- With a language-based system:
  - To do anything at all you need to know some of the language.
  - But each thing you do teaches you more about the language.
- The right abstraction level helps.

Language Challenges

- For simple tasks, graphical interfaces are often easier to learn.
- With a language-based system:
  - To do anything at all you need to know some of the language.
  - But each thing you do teaches you more about the language.
- The right abstraction level helps.

Language Challenges

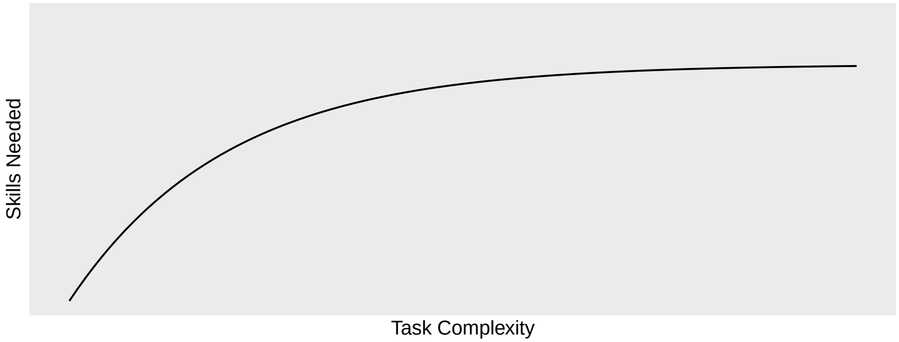
- For simple tasks, graphical interfaces are often easier to learn.
- With a language-based system:
  - To do anything at all you need to know some of the language.
  - But each thing you do teaches you more about the language.
- The right abstraction level helps.

Language Challenges

- For simple tasks, graphical interfaces are often easier to learn.
- With a language-based system:
    - To do anything at all you need to know some of the language.
    - But each thing you do teaches you more about the language.
- The right abstraction level helps.

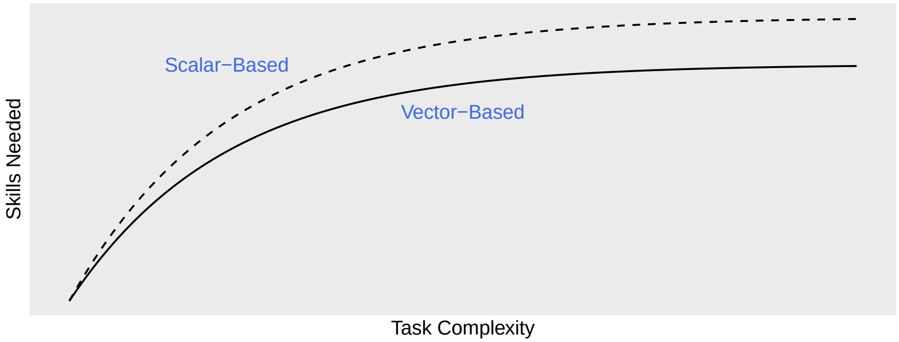
Language Challenges

- For simple tasks, graphical interfaces are often easier to learn.
- With a language-based system:
  - To do anything at all you need to know some of the language.
  - But each thing you do teaches you more about the language.
- The right abstraction level helps.



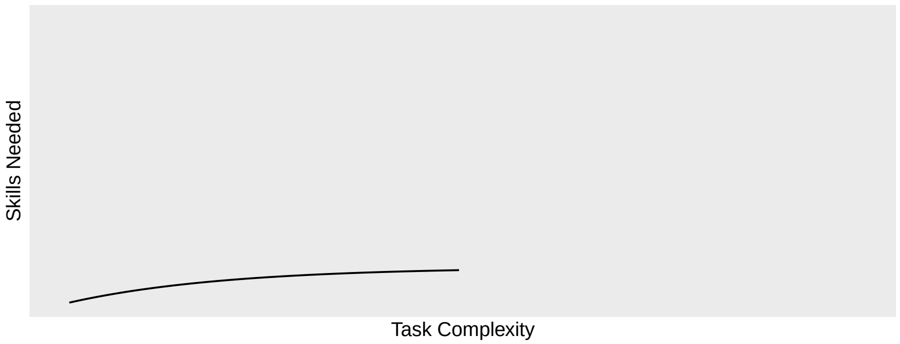
Language Challenges

- For simple tasks, graphical interfaces are often easier to learn.
- With a language-based system:
    - To do anything at all you need to know some of the language.
    - But each thing you do teaches you more about the language.
- The right abstraction level helps.



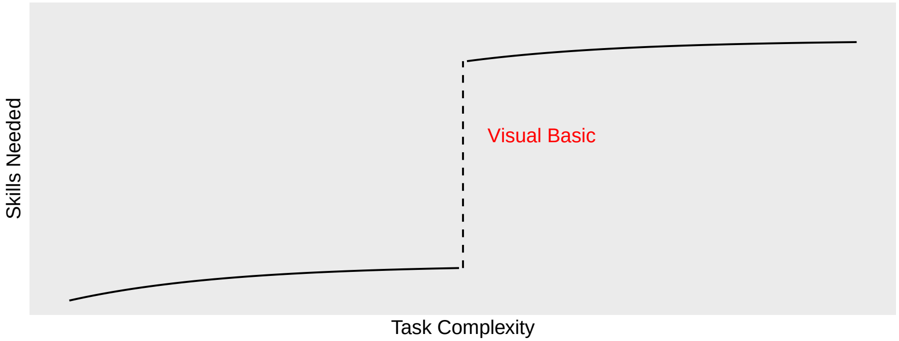
Language Challenges

- A lot can be done quite easily with a system like Excel.
- But at some point a transition to VB is needed to go further.
- Graphical interfaces built on a language system like R can try to mitigate this transition.



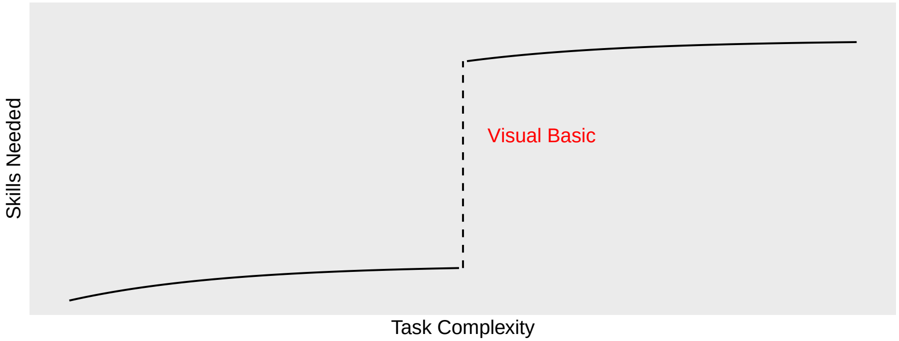
Language Challenges

- A lot can be done quite easily with a system like Excel.
- But at some point a transition to VB is needed to go further.
- Graphical interfaces built on a language system like R can try to mitigate this transition.



Language Challenges

- A lot can be done quite easily with a system like Excel.
- But at some point a transition to VB is needed to go further.
- Graphical interfaces built on a language system like R can try to mitigate this transition.



Language Challenges

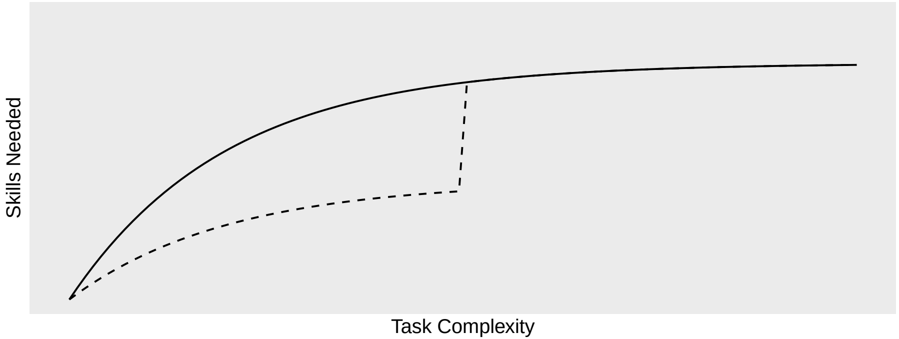
- R shares with Lisp the ability to build new sub-languages on top of the basic language.
- This can be very useful for handling problems that share the targeted structure.
- It does risk run the risk of forcing a steep transition.
- Good design may be able to reduce the magnitude, or defer the point of transition.

Language Challenges

- R shares with Lisp the ability to build new sub-languages on top of the basic language.
- This can be very useful for handling problems that share the targeted structure.
- It does risk run the risk of forcing a steep transition.
- Good design may be able to reduce the magnitude, or defer the point of transition.

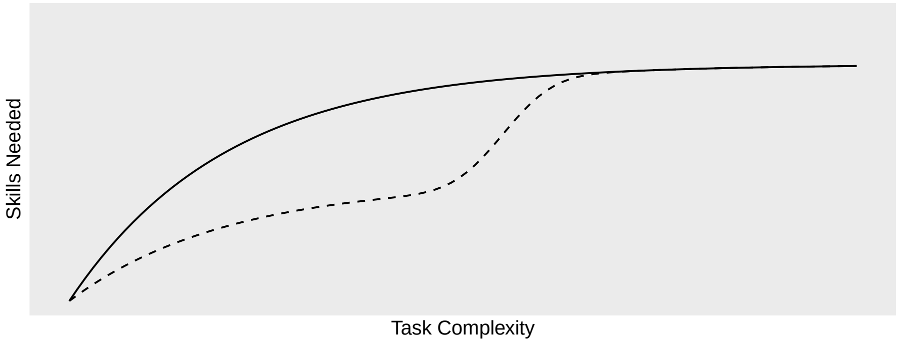
Language Challenges

- R shares with Lisp the ability to build new sub-languages on top of the basic language.
- This can be very useful for handling problems that share the targeted structure.
- It does risk run the risk of forcing a steep transition.
- Good design may be able to reduce the magnitude, or defer the point of transition.



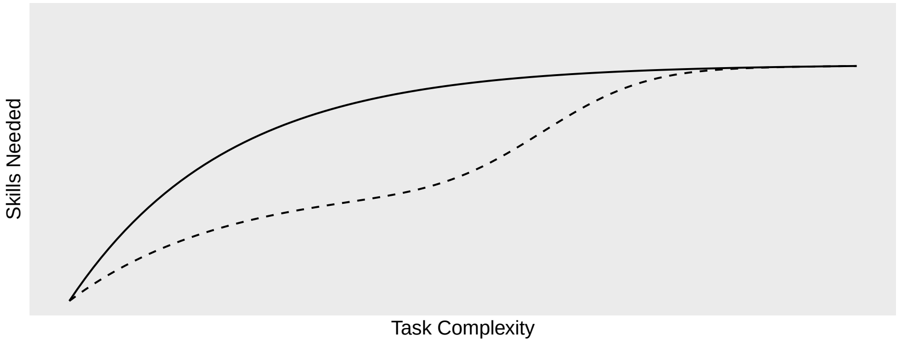
Language Challenges

- R shares with Lisp the ability to build new sub-languages on top of the basic language.
- This can be very useful for handling problems that share the targeted structure.
- It does risk run the risk of forcing a steep transition.
- Good design may be able to reduce the magnitude, or defer the point of transition.



Language Challenges

- R shares with Lisp the ability to build new sub-languages on top of the basic language.
- This can be very useful for handling problems that share the targeted structure.
- It does risk run the risk of forcing a steep transition.
- Good design may be able to reduce the magnitude, or defer the point of transition.



Some Directions for R

- With the growth and success of R come challenges:
  - Changing language features in hard, but sometimes possible.
  - Changing implementation features is hard, but sometimes possible.
- A language change that may happen:

  stringsAsFactors = FALSE
- Some implementation changes being explored:
  - More use of and extensions to the ALTREP framework.
  - Switch to reference counting for reducing copying.
  - Improved compiled code performance.

Some Directions for R

- With the growth and success of R come challenges:
    - Changing language features in hard, but sometimes possible.
    - Changing implementation features is hard, but sometimes possible.
- A language change that may happen:

  stringsAsFactors = FALSE
- Some implementation changes being explored:
    - More use of and extensions to the ALTREP framework.
    - Switch to reference counting for reducing copying.
    - Improved compiled code performance.

Some Directions for R

- With the growth and success of R come challenges:
  - Changing language features in hard, but sometimes possible.
  - Changing implementation features is hard, but sometimes possible.
- A language change that may happen:

  stringsAsFactors = FALSE
- Some implementation changes being explored:
  - More use of and extensions to the ALTREP framework.
  - Switch to reference counting for reducing copying.
  - Improved compiled code performance.

Some Directions for R

- With the growth and success of R come challenges:
  - Changing language features in hard, but sometimes possible.
  - Changing implementation features is hard, but sometimes possible.
- A language change that may happen:

  stringsAsFactors = FALSE
- Some implementation changes being explored:
  - More use of and extensions to the ALTREP framework.
  - Switch to reference counting for reducing copying.
  - Improved compiled code performance.

Some Directions for R

- With the growth and success of R come challenges:
  - Changing language features in hard, but sometimes possible.
  - Changing implementation features is hard, but sometimes possible.
- A language change that may happen:

  stringsAsFactors = FALSE
- Some implementation changes being explored:
  - More use of and extensions to the ALTREP framework.
  - Switch to reference counting for reducing copying.
  - Improved compiled code performance.

Some Directions for R

- With the growth and success of R come challenges:
  - Changing language features in hard, but sometimes possible.
  - Changing implementation features is hard, but sometimes possible.
- A language change that may happen:

  stringsAsFactors = FALSE
- Some implementation changes being explored:
  - More use of and extensions to the ALTREP framework.
  - Switch to reference counting for reducing copying.
  - Improved compiled code performance.

Some Directions for R

- With the growth and success of R come challenges:
  - Changing language features in hard, but sometimes possible.
  - Changing implementation features is hard, but sometimes possible.
- A language change that may happen:

  stringsAsFactors = FALSE
- Some implementation changes being explored:
  - More use of and extensions to the ALTREP framework.
  - Switch to reference counting for reducing copying.
  - Improved compiled code performance.

Some Directions for R

- With the growth and success of R come challenges:
  - Changing language features in hard, but sometimes possible.
  - Changing implementation features is hard, but sometimes possible.
- A language change that may happen:

  stringsAsFactors = FALSE
- Some implementation changes being explored:
  - More use of and extensions to the ALTREP framework.
  - Switch to reference counting for reducing copying.
  - Improved compiled code performance.

Some Directions for R

- With the growth and success of R come challenges:
  - Changing language features in hard, but sometimes possible.
  - Changing implementation features is hard, but sometimes possible.
- A language change that may happen:

  stringsAsFactors = FALSE
- Some implementation changes being explored:
  - More use of and extensions to the ALTREP framework.
  - Switch to reference counting for reducing copying.
  - Improved compiled code performance.

Thank You