What (Almost) No One Else Will Tell You About R

Norm Matloff
University of California at Davis

Berkeley R Beginners Group
May 20, 2014

URL for these slides (repeated on final slide):
`http://heather.cs.ucdavis.edu/BerkeleyRGroup.pdf`

Norm Matloff
University of California at Davis

Topics to Be Covered

Topics to Be Covered

- Words to the wise beginner:

Topics to Be Covered

- Words to the wise beginner:
  - Proving the “R has a steep learning curve” naysayers wrong.

Topics to Be Covered

- Words to the wise beginner:
  - Proving the “R has a steep learning curve” naysayers wrong.
  - R is “documentationally challenged”—how to cope.

Topics to Be Covered

- Words to the wise beginner:
  - Proving the “R has a steep learning curve” naysayers wrong.
  - R is “documentationally challenged”—how to cope.
- Issues with R object types (*classes*):

Topics to Be Covered

- Words to the wise beginner:
    - Proving the "R has a steep learning curve" naysayers wrong.
    - R is "documentationally challenged"—how to cope.
- Issues with R object types (*classes*):
    - (Quite a bit here.)
    - Why it matters, even to beginners.

Topics to Be Covered

- Words to the wise beginner:
    - Proving the “R has a steep learning curve” naysayers wrong.
    - R is “documentationally challenged”—how to cope.
- Issues with R object types (*classes*):
    - (Quite a bit here.)
    - Why it matters, even to beginners.
    - Why so many different class types?

Topics to Be Covered

- Words to the wise beginner:
    - Proving the "R has a steep learning curve" naysayers wrong.
    - R is "documentationally challenged"—how to cope.
- Issues with R object types (*classes*):
    - (Quite a bit here.)
    - Why it matters, even to beginners.
    - Why so many different class types?
    - R classes and further documentational challenge.

Topics to Be Covered

- Words to the wise beginner:
    - Proving the “R has a steep learning curve” naysayers wrong.
    - R is “documentationally challenged”—how to cope.
- Issues with R object types (*classes*):
    - (Quite a bit here.)
    - Why it matters, even to beginners.
    - Why so many different class types?
    - R classes and further documentational challenge.
    - Using R classes to YOUR advantage.

Topics to Be Covered

- Words to the wise beginner:
    - Proving the "R has a steep learning curve" naysayers wrong.
    - R is "documentationally challenged"—how to cope.
- Issues with R object types (*classes*):
    - (Quite a bit here.)
    - Why it matters, even to beginners.
    - Why so many different class types?
    - R classes and further documentational challenge.
    - Using R classes to YOUR advantage.
- Debugging R!

Topics to Be Covered

- Words to the wise beginner:
    - Proving the “R has a steep learning curve” naysayers wrong.
    - R is “documentationally challenged”—how to cope.
- Issues with R object types (*classes*):
    - (Quite a bit here.)
    - Why it matters, even to beginners.
    - Why so many different class types?
    - R classes and further documentational challenge.
    - Using R classes to YOUR advantage.
- Debugging R!
- R vs. other languages

Norm Matloff
University of California at Davis

Words to the Wise R Beginner

Words to the Wise R Beginner

- R does NOT have a "steep learning curve."

Words to the Wise R Beginner

- R does NOT have a "steep learning curve." (Rumor started by SAS?)

Words to the Wise R Beginner

- R does NOT have a "steep learning curve." (Rumor started by SAS?)
- Best way to learn (or to teach others):

Words to the Wise R Beginner

- R does NOT have a "steep learning curve." (Rumor started by SAS?)
- Best way to learn (or to teach others):
  - Start with short R code snippets, from a Web tutorial.

Words to the Wise R Beginner

- R does NOT have a "steep learning curve." (Rumor started by SAS?)
- Best way to learn (or to teach others):
    - Start with short R code snippets, from a Web tutorial.
    - Then just do small modifications of those snippets, experimenting with them until you get bolder and try writing some longer ones.

Words to the Wise R Beginner

- R does NOT have a “steep learning curve.” (Rumor started by SAS?)
- Best way to learn (or to teach others):
    - Start with short R code snippets, from a Web tutorial.
    - Then just do small modifications of those snippets, experimenting with them until you get bolder and try writing some longer ones.
    - Matloff’s Motto:

Words to the Wise R Beginner

- R does NOT have a "steep learning curve." (Rumor started by SAS?)
- Best way to learn (or to teach others):
  - Start with short R code snippets, from a Web tutorial.
  - Then just do small modifications of those snippets, experimenting with them until you get bolder and try writing some longer ones.
  - Matloff's Motto:

    When in doubt,
    Try it out!

Words to the Wise R Beginner

- R does NOT have a "steep learning curve." (Rumor started by SAS?)
- Best way to learn (or to teach others):
  - Start with short R code snippets, from a Web tutorial.
  - Then just do small modifications of those snippets, experimenting with them until you get bolder and try writing some longer ones.
  - Matloff's Motto:

    When in doubt,
    Try it out!

    Example:

Words to the Wise R Beginner

- R does NOT have a “steep learning curve.” (Rumor started by SAS?)
- Best way to learn (or to teach others):
    - Start with short R code snippets, from a Web tutorial.
    - Then just do small modifications of those snippets, experimenting with them until you get bolder and try writing some longer ones.
    - Matloff’s Motto:

      When in doubt,
      Try it out!

      Example:
      Not sure if need parentheses to compute 2^{10}? Try it out!

Words to the Wise R Beginner

- R does NOT have a “steep learning curve.” (Rumor started by SAS?)
- Best way to learn (or to teach others):
  - Start with short R code snippets, from a Web tutorial.
  - Then just do small modifications of those snippets, experimenting with them until you get bolder and try writing some longer ones.
  - Matloff’s Motto:

    When in doubt,
    Try it out!

    Example:
    Not sure if need parentheses to compute 2^{10}? Try it out!

```
> 2^10
[1] 1024  # OK!
```

Words to the Wise R Beginner

- R does NOT have a “steep learning curve.” (Rumor started by SAS?)
- Best way to learn (or to teach others):
  - Start with short R code snippets, from a Web tutorial.
  - Then just do small modifications of those snippets, experimenting with them until you get bolder and try writing some longer ones.
  - Matloff’s Motto:

    When in doubt,
    Try it out!

    Example:
    Not sure if need parentheses to compute 2^{10}? Try it out!

    ```
    > 2^10
    [1] 1024   # OK!
    ```

    Experiment!

Words to Wise (cont'd.)

Words to Wise (cont'd.)

Getting help.

Words to Wise (cont'd.)

Getting help.

- Google is getting pretty good with queries like "R lm"

Words to Wise (cont'd.)

Getting help.

- Google is getting pretty good with queries like "R lm" (though possibly due to spying on you!).

Words to Wise (cont'd.)

Getting help.

- Google is getting pretty good with queries like "R lm" (though possibly due to spying on you!).
- Use Google's **site:** qualifier, e.g. for the **r-help** archives, say

```
fast apply site:tolstoy.newcastle.edu.au/R
```

Words to Wise (cont'd.)

Getting help.

- Google is getting pretty good with queries like "R lm" (though possibly due to spying on you!).
- Use Google's **site:** qualifier, e.g. for the **r-help** archives, say

```
fast apply site:tolstoy.newcastle.edu.au/R
```

- Include "CRAN" even if you aren't looking for a package.

Words to Wise (cont'd.)

Getting help.

- Google is getting pretty good with queries like "R lm" (though possibly due to spying on you!).
- Use Google's **site:** qualifier, e.g. for the **r-help** archives, say

  ```
  fast apply site:tolstoy.newcastle.edu.au/R
  ```

- Include "CRAN" even if you aren't looking for a package.
- Having multiple sources—many different Web tutorials or books helps a lot.

Words to Wise (cont'd.)

Getting help.

- Google is getting pretty good with queries like "R lm" (though possibly due to spying on you!).
- Use Google's **site:** qualifier, e.g. for the **r-help** archives, say

  ```
  fast apply site:tolstoy.newcastle.edu.au/R
  ```

- Include "CRAN" even if you aren't looking for a package.
- Having multiple sources—many different Web tutorials or books helps a lot. What is not covered well in one source may be done clearly in another.

Words to Wise (cont'd.)

Getting help.

- Google is getting pretty good with queries like "R lm" (though possibly due to spying on you!).
- Use Google's **site:** qualifier, e.g. for the **r-help** archives, say

```
fast apply site:tolstoy.newcastle.edu.au/R
```

- Include "CRAN" even if you aren't looking for a package.
- Having multiple sources—many different Web tutorials or books helps a lot. What is not covered well in one source may be done clearly in another.
- **r-help** There is (should be) no such thing as a dumb question—

Words to Wise (cont'd.)

Getting help.

- Google is getting pretty good with queries like "R lm" (though possibly due to spying on you!).
- Use Google's **site:** qualifier, e.g. for the **r-help** archives, say

```
fast apply site:tolstoy.newcastle.edu.au/R
```

- Include "CRAN" even if you aren't looking for a package.
- Having multiple sources—many different Web tutorials or books helps a lot. What is not covered well in one source may be done clearly in another.
- **r-help** There is (should be) no such thing as a dumb question—but do try your best to find the answer on your own first.

Words to Wise (cont'd.)

Getting help.

- Google is getting pretty good with queries like "R lm" (though possibly due to spying on you!).
- Use Google's **site:** qualifier, e.g. for the **r-help** archives, say

```
fast apply site:tolstoy.newcastle.edu.au/R
```

- Include "CRAN" even if you aren't looking for a package.
- Having multiple sources—many different Web tutorials or books helps a lot. What is not covered well in one source may be done clearly in another.
- **r-help** There is (should be) no such thing as a dumb question—but do try your best to find the answer on your own first. Then Ripley WILL help, believe it or not.

Words to Wise (cont'd.)

Getting help.

- Google is getting pretty good with queries like "R lm" (though possibly due to spying on you!).
- Use Google's **site:** qualifier, e.g. for the **r-help** archives, say

```
fast apply site:tolstoy.newcastle.edu.au/R
```

- Include "CRAN" even if you aren't looking for a package.
- Having multiple sources—many different Web tutorials or books helps a lot. What is not covered well in one source may be done clearly in another.
- **r-help** There is (should be) no such thing as a dumb question—but do try your best to find the answer on your own first. Then Ripley WILL help, believe it or not.
- Stack Overflow for more advanced questions.

Words to Wise (cont'd.)

Words to Wise (cont'd.)

Books I like:

- *R for Dummies*, by Meys and de Vries
- *R in Action*, by Kabacoff
- *R in a Nutshell*, by Adler

R's "Object Oriented" Ethos

R's "Object Oriented" Ethos

- This is important, even for beginners.

R's "Object Oriented" Ethos

- This is important, even for beginners.
- *Object-oriented programming* (OOP) is a style:

R's "Object Oriented" Ethos

- This is important, even for beginners.
- *Object-oriented programming* (OOP) is a style:
  - Put data and functions in the same object, for clairy (*encapsulation*).

R's "Object Oriented" Ethos

- This is important, even for beginners.
- *Object-oriented programming* (OOP) is a style:
  - Put data and functions in the same object, for clairy (*encapsulation*).
  - Have the same function, e.g. **print()**, **plot()** in R, capable of working on multiple object types, for simplicity, code reuse etc. (*polymorphism*).

R's "Object Oriented" Ethos

- This is important, even for beginners.
- *Object-oriented programming* (OOP) is a style:
    - Put data and functions in the same object, for clairy (*encapsulation*).
    - Have the same function, e.g. **print()**, **plot()** in R, capable of working on multiple object types, for simplicity, code reuse etc. (*polymorphism*).
    - Grant data access on a "need to know" (and need to modify) basis, for safety (don't accidentally write to the wrong variable) (*data hiding*).

R's "Object Oriented" Ethos

- This is important, even for beginners.
- *Object-oriented programming* (OOP) is a style:
  - Put data and functions in the same object, for clairy (*encapsulation*).
  - Have the same function, e.g. **print()**, **plot()** in R, capable of working on multiple object types, for simplicity, code reuse etc. (*polymorphism*).
  - Grant data access on a "need to know" (and need to modify) basis, for safety (don't accidentally write to the wrong variable) (*data hiding*).
  - Etc.

R's "Object Oriented" Ethos

- This is important, even for beginners.
- *Object-oriented programming* (OOP) is a style:
    - Put data and functions in the same object, for clairy (*encapsulation*).
    - Have the same function, e.g. **print()**, **plot()** in R, capable of working on multiple object types, for simplicity, code reuse etc. (*polymorphism*).
    - Grant data access on a "need to know" (and need to modify) basis, for safety (don't accidentally write to the wrong variable) (*data hiding*).
    - Etc.
- The Cult of OOP

R's "Object Oriented" Ethos

- This is important, even for beginners.
- *Object-oriented programming* (OOP) is a style:
  - Put data and functions in the same object, for clairy (*encapsulation*).
  - Have the same function, e.g. **print()**, **plot()** in R, capable of working on multiple object types, for simplicity, code reuse etc. (*polymorphism*).
  - Grant data access on a "need to know" (and need to modify) basis, for safety (don't accidentally write to the wrong variable) (*data hiding*).
  - Etc.
- The Cult of OOP
  - "OOP makes things easier to program, easier to maintain!"

R's "Object Oriented" Ethos

- This is important, even for beginners.
- *Object-oriented programming* (OOP) is a style:
  - Put data and functions in the same object, for clairy (*encapsulation*).
  - Have the same function, e.g. **print()**, **plot()** in R, capable of working on multiple object types, for simplicity, code reuse etc. (*polymorphism*).
  - Grant data access on a "need to know" (and need to modify) basis, for safety (don't accidentally write to the wrong variable) (*data hiding*).
  - Etc.
- The Cult of OOP
  - "OOP makes things easier to program, easier to maintain!"
  - "There is something wrong with you if you don't use OOP!"

R's "Object Oriented" Ethos

- This is important, even for beginners.
- *Object-oriented programming* (OOP) is a style:
    - Put data and functions in the same object, for clairy (*encapsulation*).
    - Have the same function, e.g. **print()**, **plot()** in R, capable of working on multiple object types, for simplicity, code reuse etc. (*polymorphism*).
    - Grant data access on a "need to know" (and need to modify) basis, for safety (don't accidentally write to the wrong variable) (*data hiding*).
    - Etc.
- The Cult of OOP
    - "OOP makes things easier to program, easier to maintain!"
    - "There is something wrong with you if you don't use OOP!"
    - OOP C++/Java has mostly replaced C in industry and academia.

OOP (cont'd.)

OOP (cont'd.)

- Is the hoopla justified?

OOP (cont'd.)

- Is the hoopla justified?
- Maybe in some senses.

OOP (cont'd.)

- Is the hoopla justified?
- Maybe in some senses.
- But OO code is <u>more</u> complex, <u>harder</u> to write/read, and <u>harder</u> to maintain.

OOP (cont'd.)

- Is the hoopla justified?
- Maybe in some senses.
- But OO code is <u>more</u> complex, <u>harder</u> to write/read, and <u>harder</u> to maintain.
- Let's call it a necessary evil. :-)

OOP (cont'd.)

- Is the hoopla justified?
- Maybe in some senses.
- But OO code is <u>more</u> complex, <u>harder</u> to write/read, and <u>harder</u> to maintain.
- Let's call it a necessary evil. :-)
- And most important, OO <u>is</u> big in R, in fact more and more so.

OOP and R

OOP and R

- Even many very mundane R functions use OOP.

OOP and R

- Even many very mundane R functions use OOP.
- E.g. **lm()** returns its results in an (S3) object

OOP and R

- Even many very mundane R functions use OOP.
- E.g. **lm()** returns its results in an (S3) object —indeed, objects within an object.

Norm Matloff
University of California at Davis

OOP and R

- Even many very mundane R functions use OOP.
- E.g. **lm()** returns its results in an (S3) object —indeed, objects within an object.
- So yes, beginners need to know something about OOP.

History of OOP in R

History of OOP in R

- Earlier R used “S3” object classes, from the S language.

History of OOP in R

- Earlier R used “S3” object classes, from the S language.
    - OOP mainly in the sense of polymorphism.

History of OOP in R

- Earlier R used "S3" object classes, from the S language.
  - OOP mainly in the sense of polymorphism.
  - Example:

```
> x <- rnorm(100)
> y <- x + rnorm(100)
> plot(x)  # plots a vector (against ‘‘time’’)
> plot(x,y)  # plots a scatter diagram
```

History of OOP in R

- Earlier R used “S3” object classes, from the S language.
  - OOP mainly in the sense of polymorphism.
  - Example:

```
> x <- rnorm(100)
> y <- x + rnorm(100)
> plot(x)  # plots a vector (against ‘‘time’’)
> plot(x,y)  # plots a scatter diagram
```

  R takes different actions, depending on whether we feed it a vector or a pair of vectors.

History of OOP in R

- Earlier R used "S3" object classes, from the S language.
  - OOP mainly in the sense of polymorphism.
  - Example:

    ```
    > x <- rnorm(100)
    > y <- x + rnorm(100)
    > plot(x)  # plots a vector (against ``time'')
    > plot(x,y)  # plots a scatter diagram
    ```

    R takes different actions, depending on whether we feed it a vector or a pair of vectors.
  - S3 is fairly simple.

History of OOP in R

- Earlier R used "S3" object classes, from the S language.
    - OOP mainly in the sense of polymorphism.
    - Example:

      ```
      > x <- rnorm(100)
      > y <- x + rnorm(100)
      > plot(x)  # plots a vector (against ''time'')
      > plot(x,y)  # plots a scatter diagram
      ```

      R takes different actions, depending on whether we feed it a vector or a pair of vectors.
    - S3 is fairly simple.
- Later S4, adding data hiding and the like.

History of OOP in R

- Earlier R used "S3" object classes, from the S language.
    - OOP mainly in the sense of polymorphism.
    - Example:

      ```
      > x <- rnorm(100)
      > y <- x + rnorm(100)
      > plot(x)  # plots a vector (against ''time'')
      > plot(x,y)  # plots a scatter diagram
      ```

      R takes different actions, depending on whether we feed it a vector or a pair of vectors.
    - S3 is fairly simple.
- Later S4, adding data hiding and the like.
- Recently, *reference classes* were added, for encapsulation (and to achieve certain performance advantages).

Norm Matloff
University of California at Davis

R's OOP from the Standpoint of Beginners

R's OOP from the Standpoint of Beginners

- I prefer S3, nice and simple. Beginners should know at least S3 object syntax ($ sign).

R's OOP from the Standpoint of Beginners

- I prefer S3, nice and simple. Beginners should know at least S3 object syntax ($ sign).
- Many R packages now use S4. Beginners need to know at least S4 object syntax (@ sign).

R's OOP from the Standpoint of Beginners

- I prefer S3, nice and simple. Beginners should know at least S3 object syntax ($ sign).
- Many R packages now use S4. Beginners need to know at least S4 object syntax (@ sign).
- Reference classes not common yet.

Case Study: Data Frames, lm()

Case Study: Data Frames, lm()

```
> library(freqparcoord)
> data(mlb)
> head(mlb)
             Name Team        Position Height Weight
1   Adam_Donachie  BAL         Catcher     74    180 2
2       Paul_Bako  BAL         Catcher     74    215 3
3 Ramon_Hernandez  BAL         Catcher     72    210 3
4    Kevin_Millar  BAL   First_Baseman     72    210 3
5     Chris_Gomez  BAL   First_Baseman     73    188 3
6   Brian_Roberts  BAL Second_Baseman     69    176 2
> class(mlb)
[1] "data.frame"
```

Case Study, cont'd.

Case Study, cont'd.

```
> class(mlb)
[1] "data.frame"
> str(mlb)
'data.frame':   1015 obs. of  7 variables:
 $ Name       : Factor w/ 1013 levels "A.J._Burnett",
199 134 717 703 66 22 ...
 $ Team       : Factor w/ 30 levels "ANA","ARZ","ATL"
4 4 4 ...
...
> names(mlb)
[1] "Name"        "Team"        "Position"    "Height
[6] "Age"         "PosCategory"
```

Case Study, cont'd.

Case Study, cont'd.

- Yes, data frames are objects!

Case Study, cont'd.

- Yes, data frames are objects! Object type is **"data.frame"**.

Case Study, cont'd.

- Yes, data frames are objects! Object type is **"data.frame"**.
- What is IN my object?

Case Study, cont'd.

- Yes, data frames are objects! Object type is **"data.frame"**.
- What is IN my object?
    - Usually the documentation WON'T TELL YOU.

Case Study, cont'd.

- Yes, data frames are objects! Object type is **"data.frame"**.
- What is IN my object?
    - Usually the documentation WON'T TELL YOU.
    - Use **str()** to view the innards of a class.

Case Study, cont'd.

- Yes, data frames are objects! Object type is **"data.frame"**.
- What is IN my object?
    - Usually the documentation WON'T TELL YOU.
    - Use **str()** to view the innards of a class.
    - Use **names()** to view the names of the components of a class.

Case Study, cont'd.

- Yes, data frames are objects! Object type is **"data.frame"**.
- What is IN my object?
    - Usually the documentation WON'T TELL YOU.
    - Use **str()** to view the innards of a class.
    - Use **names()** to view the names of the components of a class.
    - An S3 object is just an R list, with a class type tacked on.

Case Study, cont'd.

- Yes, data frames are objects! Object type is **"data.frame"**.
- What is IN my object?
  - Usually the documentation WON'T TELL YOU.
  - Use **str()** to view the innards of a class.
  - Use **names()** to view the names of the components of a class.
  - An S3 object is just an R list, with a class type tacked on.
  - The components of an S3 object are delineated by '$', e.g.

```
> head(mlb$Weight)
[1] 180 215 210 210 188 176
```

Case Study, cont'd.

Case Study, cont'd.

Try **lm()**:

Case Study, cont'd.

Try **lm()**:

```
> lmout <- lm(Weight ~ Height+Age, data=mlb)
> class(lmout)
[1] "lm"
> lmout

Call:
lm(formula = Weight ~ Height + Age, data = mlb)

Coefficients:
(Intercept)       Height          Age
  -187.6382       4.9236       0.9115
```

Case Study, cont'd.

Case Study, cont'd.

What do we see?

Case Study, cont'd.

What do we see?

- The **lm()** function output is an object!

Case Study, cont'd.

What do we see?

- The **lm()** function output is an object! Of class type **"lm"**.

Case Study, cont'd.

What do we see?

- The **lm()** function output is an object! Of class type **"lm"**.
- In R's interactive mode, typing the name of an object will call **print()** on that object, e.g.

Case Study, cont'd.

What do we see?

- The **lm()** function output is an object! Of class type **"lm"**.
- In R's interactive mode, typing the name of an object will call **print()** on that object, e.g.

```
> z <- 8
> z
[1] 8
> print(z)
[1] 8
```

Norm Matloff
University of California at Davis

Case Study, cont'd.

Case Study, cont'd.

- So, our typing

```
> lmout
```

Norm Matloff
University of California at Davis

Case Study, cont'd.

- So, our typing

  ```
  > lmout
  ```

  was the same as typing

Case Study, cont'd.

- So, our typing

  ```
  > lmout
  ```

  was the same as typing

  ```
  > print(lmout)
  ```

Case Study, cont'd.

- So, our typing

  ```
  > lmout
  ```

  was the same as typing

  ```
  > print(lmout)
  ```

- But that is the same as typing

Case Study, cont'd.

- So, our typing

  ```
  > lmout
  ```

  was the same as typing

  ```
  > print(lmout)
  ```

- But that is the same as typing

  ```
  > print.lm()
  ```

Case Study, cont'd.

- So, our typing

  ```
  > lmout
  ```

  was the same as typing

  ```
  > print(lmout)
  ```

- But that is the same as typing

  ```
  > print.lm()
  ```

  This is how R's polymorphism works: R *dispatches* our call to **print()** to a print function specific to **lm()**.

Case Study, cont'd.

Case Study, cont'd.

- Similarly, **summary(lmout)** will trigger **summary.lm(lmout)**.

Case Study, cont'd.

- Similarly, **summary(lmout)** will trigger **summary.lm(lmout)**.(Including, by the way, computation of adjusted R^2.

Case Study, cont'd.

- Similarly, **summary(lmout)** will trigger **summary.lm(lmout)**.(Including, by the way, computation of adjusted R^2.)
- The call **plot(lmout)** will be dispatched to **plot.lm(lmout)**, which will actually display a series of regression diagnostic plots.

Case Study, cont'd.

- Similarly, **summary(lmout)** will trigger **summary.lm(lmout)**.(Including, by the way, computation of adjusted R^2.)
- The call **plot(lmout)** will be dispatched to **plot.lm(lmout)**, which will actually display a series of regression diagnostic plots.
- This is really nice! If you are working with a new R function, you can learn about it by calling **summary()** on its return value—without knowing anything about the class type, etc.

Case Study, cont'd.

- Similarly, **summary(lmout)** will trigger **summary.lm(lmout)**.(Including, by the way, computation of adjusted R^2.)
- The call **plot(lmout)** will be dispatched to **plot.lm(lmout)**, which will actually display a series of regression diagnostic plots.
- This is really nice! If you are working with a new R function, you can learn about it by calling **summary()** on its return value—without knowing anything about the class type, etc.
- So, maybe OOP is good after all!

Case Study, cont'd.

- Similarly, **summary(lmout)** will trigger **summary.lm(lmout)**.(Including, by the way, computation of adjusted R^2.)
- The call **plot(lmout)** will be dispatched to **plot.lm(lmout)**, which will actually display a series of regression diagnostic plots.
- This is really nice! If you are working with a new R function, you can learn about it by calling **summary()** on its return value—without knowing anything about the class type, etc.
- So, maybe OOP is good after all! (At least for S3.)

Case Study, cont'd.)

Case Study, cont'd.)

```
> str(lmout)
List of 13
 $ coefficients : Named num [1:3] -187.638 4.924 0.91
  ..- attr(*, "names")= chr [1:3] "(Intercept)" "Heig
 $ residuals    : Named num [1:1015] -17.66 6.67 15.0
  ..- attr(*, "names")= chr [1:1015] "1" "2" "3" "4"
 $ effects      : Named num [1:1015] -6414.8 -352.5 -
...
  ..- attr(*, "names")= chr [1:1015] "(Intercept)" "H
 $ rank         : int 3
 $ fitted.values: Named num [1:1015] 198 208 195 199
  ..- attr(*, "names")= chr [1:1015] "1" "2" "3" "4"
 $ assign       : int [1:3] 0 1 2
 $ qr           :List of 5
  ..$ qr   : num [1:1015, 1:3] -31.8591 0.0314 0.0314
  .. ..- attr(*, "dimnames")=List of 2
...
```

Case Study, cont'd.)

Case Study, cont'd.)

- Output of **str(lmout)** is pretty complex!

Case Study, cont'd.)

- Output of **str(lmout)** is pretty complex!
- As mentioned, objects within objects!

Case Study, cont'd.)

- Output of **str(lmout)** is pretty complex!
- As mentioned, objects within objects!
- But still, inspecting of the innards of an object in this way will often make up for R's poor documentation.

Turning OOP to Your Advantage

Turning OOP to Your Advantage

- From one of my blog posts, `http://matloff.wordpress.com/2014/04/01/adding-annotation-to-r-objects/`

Turning OOP to Your Advantage

- From one of my blog posts, `http://matloff.wordpress.com/2014/04/01/adding-annotation-to-r-objects/`
- I showed how you can add your own personal information to an S3 object, e.g.:

Turning OOP to Your Advantage

- From one of my blog posts, `http://matloff.wordpress.com/2014/04/01/adding-annotation-to-r-objects/`
- I showed how you can add your own personal information to an S3 object, e.g.:

```
> lmout$daterun <- date()
> lmout$daterun
[1] "Tue May 20 13:44:32 2014"
```

Turning OOP to Your Advantage

- From one of my blog posts, `http://matloff.wordpress.com/2014/04/01/adding-annotation-to-r-objects/`
- I showed how you can add your own personal information to an S3 object, e.g.:

```
> lmout$daterun <- date()
> lmout$daterun
[1] "Tue May 20 13:44:32 2014"
```

- So now if I save this output, e.g. via

```
> save(lmout,file="MLBregress")
```

Turning OOP to Your Advantage

- From one of my blog posts, `http://matloff.wordpress.com/2014/04/01/adding-annotation-to-r-objects/`
- I showed how you can add your own personal information to an S3 object, e.g.:

```
> lmout$daterun <- date()
> lmout$daterun
[1] "Tue May 20 13:44:32 2014"
```

- So now if I save this output, e.g. via

```
> save(lmout,file="MLBregress")
```

I'll have a record of when I ran the analysis.

Turning OOP to Your Advantage

- From one of my blog posts, `http://matloff.wordpress.com/2014/04/01/adding-annotation-to-r-objects/`
- I showed how you can add your own personal information to an S3 object, e.g.:

```
> lmout$daterun <- date()
> lmout$daterun
[1] "Tue May 20 13:44:32 2014"
```

- So now if I save this output, e.g. via

```
> save(lmout,file="MLBregress")
```

  I'll have a record of when I ran the analysis. I can add various other annotations, whatever I want.

Turning OOP to Your Advantage

- From one of my blog posts, `http://matloff.wordpress.com/2014/04/01/adding-annotation-to-r-objects/`
- I showed how you can add your own personal information to an S3 object, e.g.:

```
> lmout$daterun <- date()
> lmout$daterun
[1] "Tue May 20 13:44:32 2014"
```

- So now if I save this output, e.g. via

```
> save(lmout,file="MLBregress")
```

  I'll have a record of when I ran the analysis. I can add various other annotations, whatever I want.
- However, I can NOT do this with S4.

Turning OOP to Your Advantage

- From one of my blog posts, `http://matloff.wordpress.com/2014/04/01/adding-annotation-to-r-objects/`
- I showed how you can add your own personal information to an S3 object, e.g.:

  ```
  > lmout$daterun <- date()
  > lmout$daterun
  [1] "Tue May 20 13:44:32 2014"
  ```

- So now if I save this output, e.g. via

  ```
  > save(lmout,file="MLBregress")
  ```

  I'll have a record of when I ran the analysis. I can add various other annotations, whatever I want.
- However, I can NOT do this with S4. Actually, that is the POINT of S4—to disallow, say, accidentally misspelling a variable name and unintentionally creating a new variable.

Turning OOP to Your Advantage

- From one of my blog posts, `http://matloff.wordpress.com/2014/04/01/adding-annotation-to-r-objects/`
- I showed how you can add your own personal information to an S3 object, e.g.:

```
> lmout$daterun <- date()
> lmout$daterun
[1] "Tue May 20 13:44:32 2014"
```

- So now if I save this output, e.g. via

```
> save(lmout,file="MLBregress")
```

  I'll have a record of when I ran the analysis. I can add various other annotations, whatever I want.
- However, I can NOT do this with S4. Actually, that is the POINT of S4—to disallow, say, accidentally misspelling a variable name and unintentionally creating a new variable.
- So, which is better, S3 or S4?

Turning OOP to Your Advantage

- From one of my blog posts, `http://matloff.wordpress.com/2014/04/01/adding-annotation-to-r-objects/`
- I showed how you can add your own personal information to an S3 object, e.g.:

```
> lmout$daterun <- date()
> lmout$daterun
[1] "Tue May 20 13:44:32 2014"
```

- So now if I save this output, e.g. via

```
> save(lmout,file="MLBregress")
```

  I'll have a record of when I ran the analysis. I can add various other annotations, whatever I want.
- However, I can NOT do this with S4. Actually, that is the POINT of S4—to disallow, say, accidentally misspelling a variable name and unintentionally creating a new variable.
- So, which is better, S3 or S4? You decide.

Debugging

Debugging

- Matloff's Principle of Confirmation:

Debugging

- Matloff's Principle of Confirmation:
  - Execute your code step-by-step, <u>confirming</u> at each step that "What you THINK is true, IS true."

Debugging

- Matloff's Principle of Confirmation:
    - Execute your code step-by-step, confirming at each step that "What you THINK is true, IS true." E.g. you think **x** is 3 but it turns out to be 8.

Debugging

- Matloff's Principle of Confirmation:
  - Execute your code step-by-step, confirming at each step that "What you THINK is true, IS true." E.g. you think **x** is 3 but it turns out to be 8.
  - Eventually something will FAIL to confirm—usually a clue as to the location of the bug.

Debugging

- Matloff's Principle of Confirmation:
  - Execute your code step-by-step, confirming at each step that "What you THINK is true, IS true." E.g. you think **x** is 3 but it turns out to be 8.
  - Eventually something will FAIL to confirm—usually a clue as to the location of the bug.
- But HOW can you step through your code?

Debugging

- Matloff's Principle of Confirmation:
    - Execute your code step-by-step, confirming at each step that "What you THINK is true, IS true." E.g. you think **x** is 3 but it turns out to be 8.
    - Eventually something will FAIL to confirm—usually a clue as to the location of the bug.
- But HOW can you step through your code?
    - R's built-in functions, especially **debug()** and **setBreakpoint()**.

Debugging

- Matloff's Principle of Confirmation:
    - Execute your code step-by-step, confirming at each step that "What you THINK is true, IS true." E.g. you think **x** is 3 but it turns out to be 8.
    - Eventually something will FAIL to confirm—usually a clue as to the location of the bug.
- But HOW can you step through your code?
    - R's built-in functions, especially **debug()** and **setBreakpoint()**.
    - RStudio's debugging tool.

Debugging

- Matloff's Principle of Confirmation:
  - Execute your code step-by-step, confirming at each step that "What you THINK is true, IS true." E.g. you think **x** is 3 but it turns out to be 8.
  - Eventually something will FAIL to confirm—usually a clue as to the location of the bug.
- But HOW can you step through your code?
  - R's built-in functions, especially **debug()** and **setBreakpoint()**.
  - RStudio's debugging tool.
  - The **ess-tracebug** tool is nice, if you have the patience for Emacs.

Debugging

- Matloff's Principle of Confirmation:
    - Execute your code step-by-step, confirming at each step that "What you THINK is true, IS true." E.g. you think **x** is 3 but it turns out to be 8.
    - Eventually something will FAIL to confirm—usually a clue as to the location of the bug.
- But HOW can you step through your code?
    - R's built-in functions, especially **debug()** and **setBreakpoint()**.
    - RStudio's debugging tool.
    - The **ess-tracebug** tool is nice, if you have the patience for Emacs.
    - My own debugging tool, **debugR**, at `http://heather.cs.ucdavis.edu/debugR.html`.

Debugging

- Matloff's Principle of Confirmation:
    - Execute your code step-by-step, confirming at each step that "What you THINK is true, IS true." E.g. you think **x** is 3 but it turns out to be 8.
    - Eventually something will FAIL to confirm—usually a clue as to the location of the bug.
- But HOW can you step through your code?
    - R's built-in functions, especially **debug()** and **setBreakpoint()**.
    - RStudio's debugging tool.
    - The **ess-tracebug** tool is nice, if you have the patience for Emacs.
    - My own debugging tool, **debugR**, at `http://heather.cs.ucdavis.edu/debugR.html`.
- Lots more to say, no time now.

R vs. the World

R vs. the World

Lots of pretenders to the stat software throne these days,

R vs. the World

Lots of pretenders to the stat software throne these days, notably Python and Julia.

R vs. the World

Lots of pretenders to the stat software throne these days, notably Python and Julia.

Some thoughts:

R vs. the World

Lots of pretenders to the stat software throne these days, notably Python and Julia.
Some thoughts:

- I'm a big fan of Python.

R vs. the World

Lots of pretenders to the stat software throne these days, notably Python and Julia.
Some thoughts:

- I'm a big fan of Python. It is clean and elegant,

R vs. the World

Lots of pretenders to the stat software throne these days, notably Python and Julia.
Some thoughts:

- I'm a big fan of Python. It is clean and elegant, and (for OOP cultists) it has a strong class structure.

R vs. the World

Lots of pretenders to the stat software throne these days, notably Python and Julia.
Some thoughts:

- I'm a big fan of Python. It is clean and elegant, and (for OOP cultists) it has a strong class structure.
- But R is created BY statisticians, FOR statisticians.

R vs. the World

Lots of pretenders to the stat software throne these days, notably Python and Julia.
Some thoughts:

- I'm a big fan of Python. It is clean and elegant, and (for OOP cultists) it has a strong class structure.
- But R is created BY statisticians, FOR statisticians. It is Statistically Correct.

R vs. the World

Lots of pretenders to the stat software throne these days, notably Python and Julia.
Some thoughts:

- I'm a big fan of Python. It is clean and elegant, and (for OOP cultists) it has a strong class structure.
- But R is created BY statisticians, FOR statisticians. It is Statistically Correct.
- Simply coding up a regression algorithm, say, by reading formulas ih a book, will generally NOT produce good, useful software.

R vs. the World

Lots of pretenders to the stat software throne these days, notably Python and Julia.
Some thoughts:

- I'm a big fan of Python. It is clean and elegant, and (for OOP cultists) it has a strong class structure.
- But R is created BY statisticians, FOR statisticians. It is Statistically Correct.
- Simply coding up a regression algorithm, say, by reading formulas ih a book, will generally NOT produce good, useful software. An Argentinian chef making Japanese sushii may have the right ingredients, but may still fall far short in various ways.

R vs. the World

Lots of pretenders to the stat software throne these days, notably Python and Julia.
Some thoughts:

- I'm a big fan of Python. It is clean and elegant, and (for OOP cultists) it has a strong class structure.
- But R is created BY statisticians, FOR statisticians. It is Statistically Correct.
- Simply coding up a regression algorithm, say, by reading formulas ih a book, will generally NOT produce good, useful software. An Argentinian chef making Japanese sushii may have the right ingredients, but may still fall far short in various ways.
- R's vast CRAN code repository is not going to be duplicated in Python or Julia anytime soon.

R vs. the World

Lots of pretenders to the stat software throne these days, notably Python and Julia.
Some thoughts:

- I'm a big fan of Python. It is clean and elegant, and (for OOP cultists) it has a strong class structure.
- But R is created BY statisticians, FOR statisticians. It is Statistically Correct.
- Simply coding up a regression algorithm, say, by reading formulas ih a book, will generally NOT produce good, useful software. An Argentinian chef making Japanese sushii may have the right ingredients, but may still fall far short in various ways.
- R's vast CRAN code repository is not going to be duplicated in Python or Julia anytime soon. Probably never, because stat code is not the prime mission of those languages.

R vs. the World

Norm Matloff
University of California at Davis

R vs. the World

- A speaker at BARUG last week said, “R isn’t a full language.”

R vs. the World

- A speaker at BARUG last week said, “R isn’t a full language.” Says who?!

R vs. the World

- A speaker at BARUG last week said, "R isn't a full language." Says who?!
- Yes, of course, there are always some things that language X does better than language Y.

R vs. the World

- A speaker at BARUG last week said, "R isn't a full language." Says who?!
- Yes, of course, there are always some things that language X does better than language Y.
- But overall R is just as powerful as Python or Julia, often more so.

R vs. the World

- A speaker at BARUG last week said, “R isn’t a full language.” Says who?!
- Yes, of course, there are always some things that language X does better than language Y.
- But overall R is just as powerful as Python or Julia, often more so.
- I currently use R for all the non-stat stuff that I used to use Python for.

R vs. the World: Serial Execution Speed

[1] There is also the fact that R checks for NAs, a vital but slow feature.

R vs. the World: Serial Execution Speed

The big advantange of Julia over R is allegedly execution speed.

[1]There is also the fact that R checks for NAs, a vital but slow feature.

R vs. the World: Serial Execution Speed

The big advantange of Julia over R is allegedly execution speed. This may be true for some apps, but there is also a lot of hype. Consider:

[1]There is also the fact that R checks for NAs, a vital but slow feature.

R vs. the World: Serial Execution Speed

The big advantange of Julia over R is allegedly execution speed. This may be true for some apps, but there is also a lot of hype. Consider:

- Many of the R-vs.-Julia speed comparisons have been unfair, as they don't take advantage of R's performance features (next slide).[1]

[1] There is also the fact that R checks for NAs, a vital but slow feature.

R vs. the World: Serial Execution Speed

The big advantange of Julia over R is allegedly execution speed. This may be true for some apps, but there is also a lot of hype. Consider:

- Many of the R-vs.-Julia speed comparisons have been unfair, as they don't take advantage of R's performance features (next slide).[1]
- R packages such as **data.table** and **dplyr** are quite fast for data frame manipulations, probably the bulk of high volume work that R users do.

[1]There is also the fact that R checks for NAs, a vital but slow feature.

R vs. the World: Serial Execution Speed

The big advantange of Julia over R is allegedly execution speed. This may be true for some apps, but there is also a lot of hype. Consider:

- Many of the R-vs.-Julia speed comparisons have been unfair, as they don't take advantage of R's performance features (next slide).[1]
- R packages such as **data.table** and **dplyr** are quite fast for data frame manipulations, probably the bulk of high volume work that R users do.
- For those few apps in which serial speed is really needed, **Rcpp** makes it easy to implement the core in C/C++.

[1]There is also the fact that R checks for NAs, a vital but slow feature.

R vs. the World: Serial Execution Speed

The big advantange of Julia over R is allegedly execution speed. This may be true for some apps, but there is also a lot of hype. Consider:

- Many of the R-vs.-Julia speed comparisons have been unfair, as they don't take advantage of R's performance features (next slide).[1]
- R packages such as **data.table** and **dplyr** are quite fast for data frame manipulations, probably the bulk of high volume work that R users do.
- For those few apps in which serial speed is really needed, **Rcpp** makes it easy to implement the core in C/C++.
- For true speed, one needs parallelism, which R does much better than Python and Julia (upcoming slide).

[1]There is also the fact that R checks for NAs, a vital but slow feature.

Brief Case Study: R vs. Julia

Brief Case Study: R vs. Julia

- At a BARUG meeting last year, a speaker presented an example of R-vs.-Julia in speed comparison:

Brief Case Study: R vs. Julia

- At a BARUG meeting last year, a speaker presented an example of R-vs.-Julia in speed comparison: Simulation of random walk.

Brief Case Study: R vs. Julia

- At a BARUG meeting last year, a speaker presented an example of R-vs.-Julia in speed comparison: Simulation of random walk.
- Outcome: Julia won, by a mile.

Brief Case Study: R vs. Julia

- At a BARUG meeting last year, a speaker presented an example of R-vs.-Julia in speed comparison: Simulation of random walk.
- Outcome: Julia won, by a mile.
- But...random walk is a SUM

Brief Case Study: R vs. Julia

- At a BARUG meeting last year, a speaker presented an example of R-vs.-Julia in speed comparison: Simulation of random walk.
- Outcome: Julia won, by a mile.
- But...random walk is a SUM (each step is an increment).

Brief Case Study: R vs. Julia

- At a BARUG meeting last year, a speaker presented an example of R-vs.-Julia in speed comparison: Simulation of random walk.
- Outcome: Julia won, by a mile.
- But...random walk is a SUM (each step is an increment).
- So I pointed out that one could vectorize the speaker's simulation by using R's **cumsum()**.

Brief Case Study: R vs. Julia

- At a BARUG meeting last year, a speaker presented an example of R-vs.-Julia in speed comparison: Simulation of random walk.
- Outcome: Julia won, by a mile.
- But...random walk is a SUM (each step is an increment).
- So I pointed out that one could vectorize the speaker's simulation by using R's **cumsum()**.
- Now it was R by a mile—over 1000X faster than Julia for 1000000 time steps.

Brief Case Study: R vs. Julia

- At a BARUG meeting last year, a speaker presented an example of R-vs.-Julia in speed comparison: Simulation of random walk.
- Outcome: Julia won, by a mile.
- But...random walk is a SUM (each step is an increment).
- So I pointed out that one could vectorize the speaker's simulation by using R's **cumsum()**.
- Now it was R by a mile—over 1000X faster than Julia for 1000000 time steps.
- Julia did outperform R fof 100000000 steps.

Brief Case Study: R vs. Julia

- At a BARUG meeting last year, a speaker presented an example of R-vs.-Julia in speed comparison: Simulation of random walk.
- Outcome: Julia won, by a mile.
- But...random walk is a SUM (each step is an increment).
- So I pointed out that one could vectorize the speaker's simulation by using R's **cumsum()**.
- Now it was R by a mile—over 1000X faster than Julia for 1000000 time steps.
- Julia did outperform R fof 100000000 steps. But it's clear that the Julia-faster-than-R claim was generally unwarranted.

R vs. the World: Parallel Computing

[2] There are alternatives in Python, but none satisfactory, in my view.

R vs. the World: Parallel Computing

- The really huge compute-bound apps need parallel computation.

[2] There are alternatives in Python, but none satisfactory, in my view.

R vs. the World: Parallel Computing

- The really huge compute-bound apps need parallel computation.
- For those, R is (at present) much better suited than are Python or Julia.

[2] There are alternatives in Python, but none satisfactory, in my view.

R vs. the World: Parallel Computing

- The really huge compute-bound apps need parallel computation.
- For those, R is (at present) much better suited than are Python or Julia.
- Multicore apps need *threaded* programming.

[2]There are alternatives in Python, but none satisfactory, in my view.

R vs. the World: Parallel Computing

- The really huge compute-bound apps need parallel computation.
- For those, R is (at present) much better suited than are Python or Julia.
- Multicore apps need *threaded* programming.
- Python does NOT have true-parallel threading.[2]

[2] There are alternatives in Python, but none satisfactory, in my view.

R vs. the World: Parallel Computing

- The really huge compute-bound apps need parallel computation.
- For those, R is (at present) much better suited than are Python or Julia.
- Multicore apps need *threaded* programming.
- Python does NOT have true-parallel threading.[2] Nor does Julia.

[2]There are alternatives in Python, but none satisfactory, in my view.

R vs. the World: Parallel Computing

- The really huge compute-bound apps need parallel computation.
- For those, R is (at present) much better suited than are Python or Julia.
- Multicore apps need *threaded* programming.
- Python does NOT have true-parallel threading.[2] Nor does Julia.
- Base R's **parallel** package enables true-parallel coding.

[2]There are alternatives in Python, but none satisfactory, in my view.

R vs. the World: Parallel Computing

- The really huge compute-bound apps need parallel computation.
- For those, R is (at present) much better suited than are Python or Julia.
- Multicore apps need *threaded* programming.
- Python does NOT have true-parallel threading.[2] Nor does Julia.
- Base R's **parallel** package enables true-parallel coding.
- My **Rdsm** package enables true-parallel coding in the threading style.

[2]There are alternatives in Python, but none satisfactory, in my view.

Conclusions

Conclusions

- R learning curve is NOT steep, if you go about it in the proper way.

Conclusions

- R learning curve is NOT steep, if you go about it in the proper way.
- OOP is overhyped, but it IS part of R, and even beginners need to know a bit about it.

Conclusions

- R learning curve is NOT steep, if you go about it in the proper way.
- OOP is overhyped, but it IS part of R, and even beginners need to know a bit about it.
- Take THAT, Python and Julia!

Conclusions

- R learning curve is NOT steep, if you go about it in the proper way.
- OOP is overhyped, but it IS part of R, and even beginners need to know a bit about it.
- Take THAT, Python and Julia!

Location of these slides:

Conclusions

- R learning curve is NOT steep, if you go about it in the proper way.
- OOP is overhyped, but it IS part of R, and even beginners need to know a bit about it.
- Take THAT, Python and Julia!

Location of these slides:
`http://heather.cs.ucdavis.edu/BerkeleyRGroup.pdf`