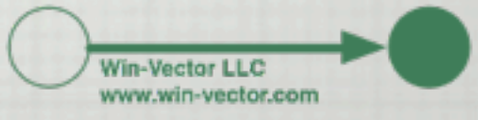
Seven Tips for Surviving R

John Mount
Win Vector LLC

Bay Area R Users Meetup
http://www.meetup.com/R-Users/calendar/11202051/
October 13, 2009
http://www.win-vector.com/dfiles/SurviveR.pdf
based on:
http://www.win-vector.com/blog/2009/09/survive-r/



A special thank you to Michael E. Driscoll (for the invitation), Microsoft (our hosts) and David M Smith, Director of Community for REvolution Computing (the main act).

Who am I?

- John Mount: a principal consultant at Win-Vector LLC (http://www.win-vector.com/)
  - We think of ourselves as “analytics shock troops.”
  - We listen, analyze and produce reports, algorithms and code.
  - Always looking for introductions.
- I was not originally a “statistics guy.”
  - I had the luxury of learning statistics after I had already learned linear algebra, computer science, algorithms, combinatorics, time series analysis, spectral analysis, probability and measure theory.
  - Given my background I felt statistics was pretty neat and has some really deep ideas and techniques.
  - If you have really found the technique that fits the problem, you can usually explain it without any of the fifty cent words listed above.
- I am not “an R guy.”
  - I can’t tell you “the right way” to do things in R.
  - I can tell you how to brute your way through.
  - I do like R.

I will give you my view of a few work style issues.

Required

This is where you should spend the money you did not spend to acquire R. My 3 core R books (still waiting for "Data Manipulation with R"). Take a breath and read an appropriate chapter (the solution will come, but perhaps not quickly- and write it down!). Lattice is based on the the 1968 William Cleveland book The Elements of Graphing Data. ggplot2 is based on the ideas from Leland Wilkison's Grammar of Graphics.

1: Write Down EVERYTHING

- When you do figure out how to do something in R it will be concise, powerful and completely un-mnemonic and impossible to find again through the help system.
- Use at least the following system:
  - Use the R gui's built in history pane
  - Copy and paste to/from an external editor
  - Maintain a task stack on paper
  - Edit down to a successful task and save in an appropriately named file in a big R directory.
- Literal masterpiece of writing things down: http://learnr.wordpress.com/ .

I will try not to read too many slides to you, but his one I will pretty much be the exception.

LearnR Example: Every figure in Sarkar's "Lattice" book redone in ggplot2

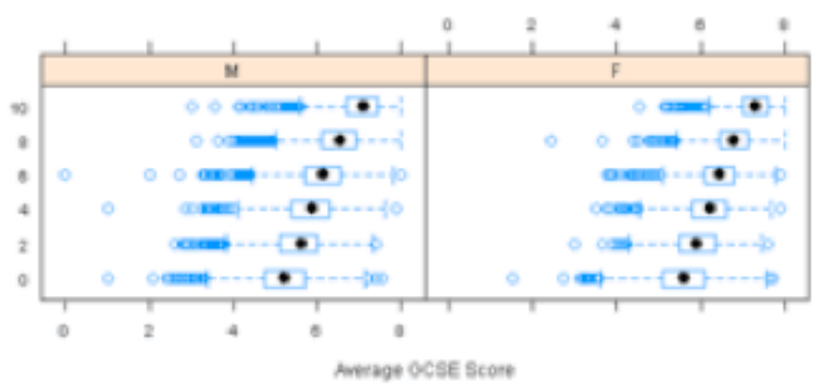
Figure 3.11

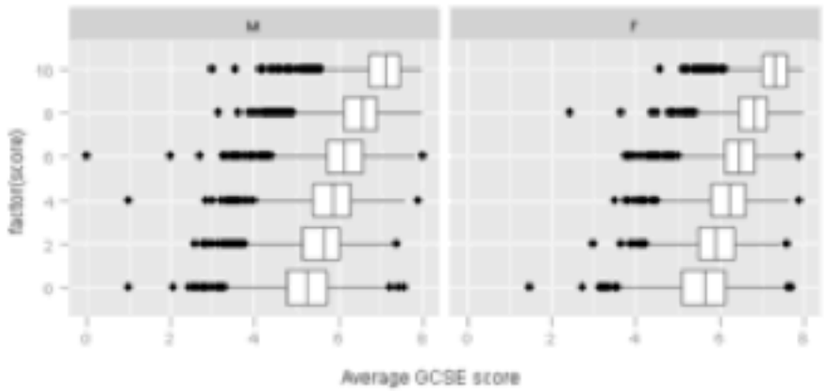
lattice

```
> pl <- bwplot(factor(score) ~ gcsescore | gender, data = Chem97,
+     xlab = "Average GCSE Score")
> print(pl)
```

ggplot2

```
> pg <- ggplot(Chem97, aes(factor(score), gcsescore)) +
+     geom_boxplot() + coord_flip() + ylab("Average GCSE score") +
+     facet_wrap(~gender)
> print(pg)
```





Win-Vector LLC
www.win-vector.com

Worked each example once in Lattice and once in ggplot2- a Rosetta stone for advanced graphics packages. Also notice the algebraic notation of ggplot2- you can use this to build up a graph and swap out the data. This is essential as you are always on your penultimate (next to last) analysis. Make everything scriptable and repeatable.

2: Find Some Way to search for R Answers

- On Google use “+R” (means search exactly as, not just force present)
- Use “site:”
  - http://groups.google.com/group/help-R
  - r-project.org
  - stackoverflow.com
  - http://stackoverflow.com/questions/102056/how-to-search-for-r-materials
  - www.rseek.org
  - http://search.r-project.org/
  - http://h4dev.com/entries?search=r+search+engine

It is very unfortunate R is a letter of the alphabet. Also, I am not a huge fan of any R-based search systems I have used. However, help() is very very useful for finding arguments of methods you already know about. Quick apology-in my earlier draft Univ. of Minnesota database I mention on slide 6 is not publicly available. Gene Tani offered this correction and his database of resources (which seems to be the most comprehensive, listing 13 sources): http://h4dev.com/entries?search=r+search+engine .

3: Learn to Examine Structures

- unclass()
- str()
- dput()
- names()
- dimnames()

R is object oriented, so in principle just about any call can be hijacked to be a method of the class of the object you are trying to examine (like print() and summary()). But deep down most complicated objects have an underlying list structure.

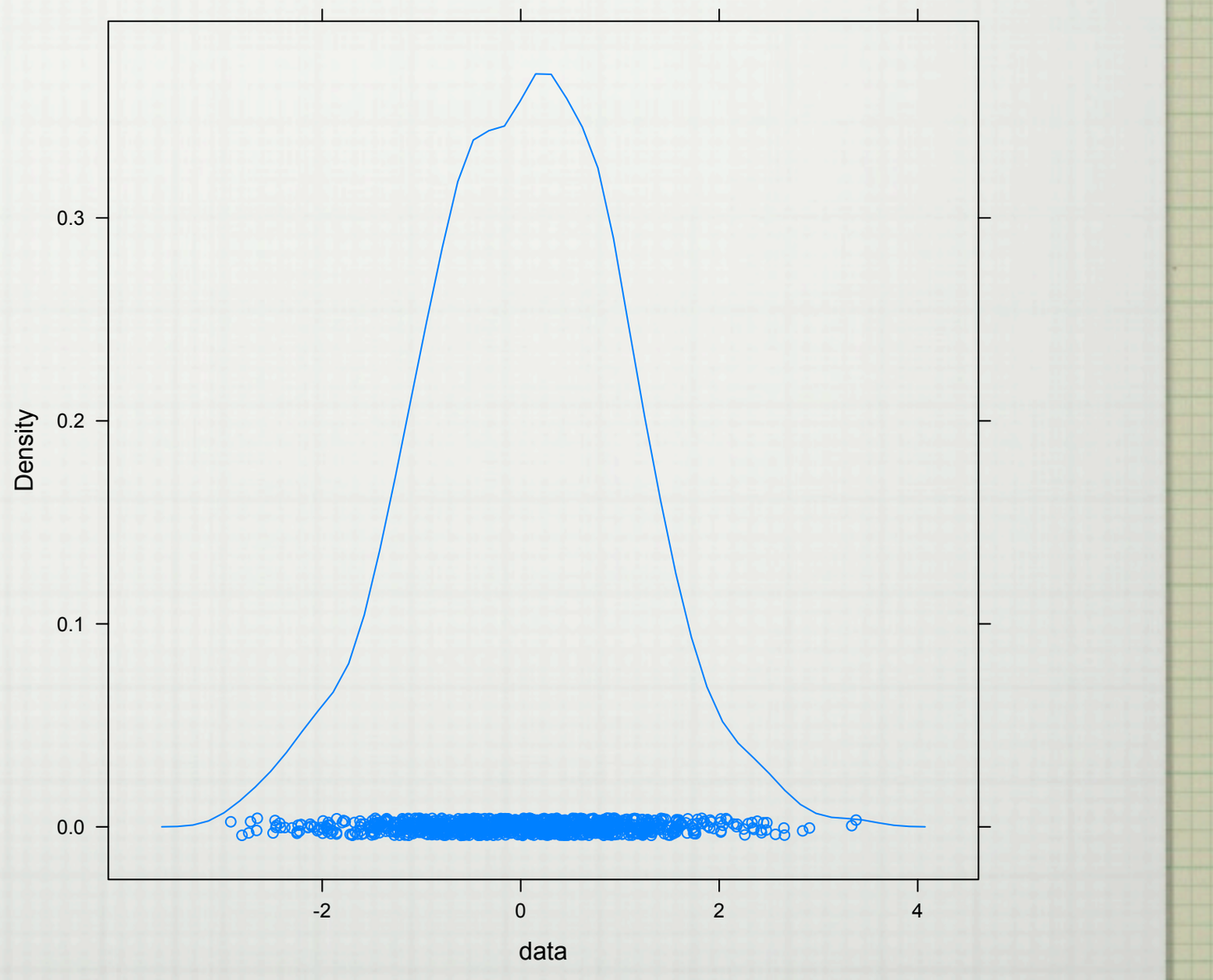
3: Learn to Examine Structures ...

```
> counts <- c(18,17,15,20,10,20,25,13,12)
> outcome <- gl(3,1,9)
> treatment <- gl(3,3)
>
>
> glm.D93 <- glm(counts ~ outcome + treatment, family=poisson())
> names(glm.D93)
 [1] "coefficients"      "residuals"         "fitted.values"     "effects"
 [5] "R"                 "rank"              "qr"                "family"
 [9] "linear.predictors" "deviance"          "aic"               "null.deviance"
[13] "iter"              "weights"           "prior.weights"     "df.residual"
[17] "df.null"           "y"                 "converged"         "boundary"
[21] "model"             "call"              "formula"           "terms"
[25] "data"              "offset"            "control"           "method"
[29] "contrasts"         "xlevels"
> print(glm.D93)

Call:  glm(formula = counts ~ outcome + treatment, family = poisson())

Coefficients:
(Intercept)     outcome2     outcome3   treatment2   treatment3
  3.045e+00   -4.543e-01   -2.930e-01    1.338e-15    1.421e-15

Degrees of Freedom: 8 Total (i.e. Null);  4 Residual
Null Deviance:      10.58
Residual Deviance: 5.129    AIC: 56.76
>
```


R is object oriented, so in principle just about any call can be hijacked to be a method of the class of the object you are trying to examine (like print() and summary()). But deep down most complicated objects have an underlying list structure.

4: Learn to Examine Methods

```
> library(lattice)
> data <- rnorm(1000)
> densityplot(data)
>
```



Win-Vector LLC
www.win-vector.com

Ever wondered how the graphing package magically knows the names of the variables you passed it? If you put your data in a variable named data then call densityplot(data) lattice knows to put the name “data” on the legend.

4: Learn to Examine Methods ...

```
> library(lattice)
> methods(densityplot)
[1] densityplot.formula* densityplot.numeric*

   Non-visible functions are asterisked
> getS3method('densityplot','numeric')
function (x, data = NULL, xlab = deparse(substitute(x)), ...)
{
    ocall <- sys.call(sys.parent())
    ocall[[1]] <- quote(densityplot)
    ccall <- match.call()
    if (!is.null(ccall$data))
        warning("explicit 'data' specification ignored")
    ccall$data <- environment()
    ccall$xlab <- xlab
    ccall$x <- ~x
    ccall[[1]] <- quote(lattice::densityplot)
    ans <- eval.parent(ccall)
    ans$call <- ocall
    ans
}
<environment: namespace:lattice>
>
```


You can examine the code and find out (answer: it is the deparse(substitute())-foo and the explicit re-execution of the call in the caller's environment). This is an example of "standard non-standard evaluation order" (praised by the graphics guys, not liked by others) and it really makes it hard to answer "what calling convention does R use?" Closest description of R calling semantics is "call by name with copy on write used to simulate call by value" (similar to Matlab).

5: Learn to Stomp Out Attributes

- attributes(x) <- c()
- names(x) <- c()
- dimnames(x) <- c()

Often (after calling summary() or MASS's fitdistr) you have something that has a class() of numeric but has an infections label (it prints its own custom label and every time you create some number that is a consequence of it, it also gets the infection). Turns out this is not related to class() but to attributes(). In R c() (the empty list) is also a synonym for null.

5: Learn to Stomp Out Attributes ...

```
> library(MASS)
> data <- rnorm(100)
> fit <- fitdistr(data,'Normal')
> fit
      mean          sd
  0.1566438   1.0669717
 (0.1066972) (0.0754463)
> names(fit)
[1] "estimate" "sd"       "n"        "loglik"
> fit$estimate[1]
     mean
0.1566438
> fit$estimate[2]
       sd
1.066972
> x <- fit$estimate[1] + 2*fit$estimate[2]
> x
    mean
2.290587
> attributes(x) <- c()
> x
[1] 2.290587
>
```

Often (after calling summary() or MASS's fitdistr) you have something that has a class() of numeric but has an infections label (it prints its own custom label and every time you create some number that is a consequence of it, it also gets the infection). Turns out this is not related to class() but to attributes(). In R c() (the empty list) is also a synonym for null.

6: Swallow Your Pride

- You may not find “the right way” or you may want to pre-process your data (either outside of R or in R).
  - Don’t be afraid to do your string processing before you get into R.
  - Don’t be afraid to create new data frames.
- Example
  - Problem: want mutable map (like string keys in lists, but able to pass the key around as a variable and able to alter the contents).
  - Solution: R evaluation environments.
    - map <- new.env(hash=TRUE,parent=emptyenv())
    - assign('dog',7,map)
    - ls(map)
    - x <- ‘dog’
    - get(x,map)
  - Do I feel good about that? Not really.

Thanks to Edward Ratzer for warning you had best set the parent environment to empty!

7: Find and Rely on “The One Liners”

- Why did I need a map?
  - To try and compute an empirical distribution.
  - Would unique(sort(keys)) have worked instead of insisting on a mutable solution?
  - Would it not be better to let R compute the whole thing with ecdf()?
- Look for complete solutions:
  - read.table()
  - table()
  - write.table()
  - anova()
  - RJDBC

Remember why you are using R.

Thank You