Essential Elements of the R Language

Glen Sargeant
Northern Prairie Wildlife Research Center
Jamestown, North Dakota
glen_sargeant@usgs.gov

Hosted by the
USGS Western Fisheries Research Center
Seattle, Washington

24-25 March, 2009

Objective

To provide a minimally sufficient, coordinated *overview* of the R language. We will revisit key concepts later (repeatedly) when we manipulate objects, implement statistical methods, and prepare graphics.

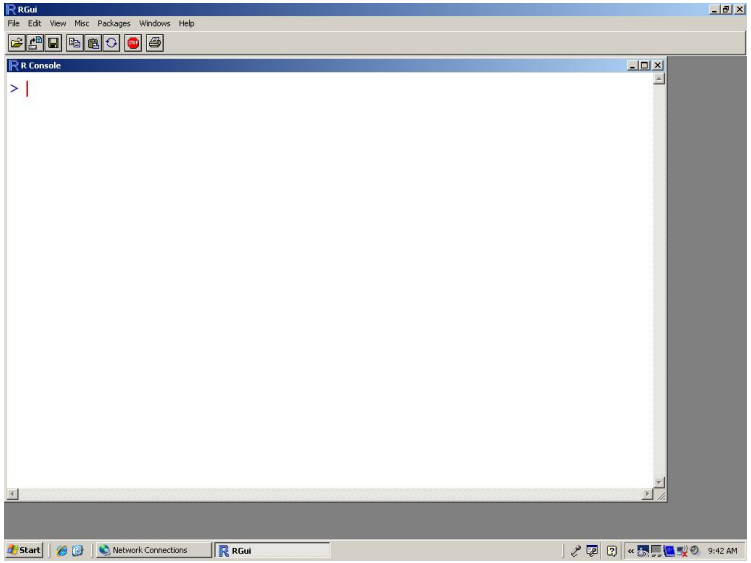
Key themes

- Using expressions to communicate with R
- Types, properties, and roles of objects
- Using functions and methods to implement analyses

Expressions

Users generally interact with R through the **R console**.

- Statements issued to the R console (from the command prompt, >) for evaluation are called **expressions**.
- Unless they are redirected, results are returned to the R console.



Types of expressions

Most expressions are of 3 types:

- Mathematical operations

```
> 2 + 2
[1] 4
```

- Logical operations

```
> (2 * 3) < (2 + 3)
[1] FALSE
```

- Function calls

```
> sqrt(2)
[1] 1.414214
```

Mathematical operations

- Addition and subtraction

```
> 2 + 3
[1] 5
> 2 - 3
[1] -1
```

- Exponentiation

```
> 2^3
[1] 8
```

- Multiplication and division

```
> 2 * 3
[1] 6
> 2/3
[1] 0.6666667
```

Logical operations

Logical operations are comparisons that return `TRUE` or `FALSE`.

```
> 2 + 3 < 3 * 2
```

```
[1] TRUE
```

Some require specialized syntax:

Comparison	**Operator**
"Equals"	==
"Not equals"	!=
"Greater than"	>
"Less than"	<
"Greater than or equal to"	>=
"Less than or equal to"	<=

Combining logical operations

"And" (&) and "or" (|) can be used to combine any number of logical operations.

- "And" returns TRUE if all conditions are met:

```
> (3 + 3 == 6) & (2 * 3 == 6)
[1] TRUE
```

- "Or" (|) returns TRUE if at least one condition is met:

```
> (3 + 3 == 5) | (2 * 3 == 6)
[1] TRUE
```

Combining logical operations

Components of compound operations must all be complete logical expressions!

- Why does R seem to think 3 is equal to 10 or 12?

```
> 3 + 3 == 10 | 12
[1] TRUE
```

Functions

- Needs of R users rarely are limited to simple calculations and comparisons. More complex operations require **functions**.
- Functions are built-in or home-made procedures that automate and simplify routine tasks.

Function calls

Functions are invoked with expressions that supply a **function call** with necessary **arguments**.

- Arguments follow the call in parentheses:

```
> sqrt(100)
[1] 10
```

- Some arguments are required. Others are optional. *The parentheses are required, even for functions that do not accept user-specified arguments.*

Expressions can be **nested**

- Expressions (including function calls) can operate on the output of other expressions:

```
> log(sqrt(100))
[1] 2.302585
```

- As in mathematics, parentheses control the order of execution (inner-most expressions are evaluated first).

Assignments

- Expressions return results: **assignments** preserve results so they can be reused.
- Two **assignment operators** (<- and =) are "almost always" equivalent (but using <- is a good idea).

```
> y <- 100
> y
[1] 100
```

```
> x = sqrt(y)
> x
[1] 10
```

Objects

- Assignments create **objects**.
- Objects are the building blocks of R.
- Virtually *everything* is an object.

A clear, coordinated understanding of objects is the key to efficient programming!

Types of objects reflect different purposes

Components of an analysis usually include **data**, **tools**, and **results**. In R, *all* of these are represented by **objects**.

- Data usually are represented by **vectors**, **matrices**, or **data frames**
- Tools for manipulating and analyzing data usually are encapsulated in **functions**
- Results often are output as **lists**

Vectors are the basic units of data organization

- Every vector is described by its **mode** (the most basic descriptor of elements) and **length** (number of elements).
- Elements must all be of the same mode (or missing). Missing values usually are represented by the characters `NA`.

Modes

Though ecological measurements take many forms, nearly all are represented by just 3 modes:

```
> mode(1)
[1] "numeric"
> mode("some text")
[1] "character"
```

```
> mode(TRUE)
[1] "logical"
```

Knowing this will help you understand R conventions for handling classification variables, dates, times, etc.

Attributes

- The mode and length of a vector are called **intrinsic attributes**.
- Additional attributes can be used to define and describe complex structures, including matrices, arrays, and data frames.

Matrices and arrays (as R sees them)

Matrices and arrays, for example, are vectors with an additional attribute (`dim`):

```
> x <- 1:4
> x
[1] 1 2 3 4
```

```
> dim(x) <- c(2, 2)
> x
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

As derivatives of vectors, matrices and arrays follow the same rule: elements must all be of the same mode.

Where do attributes come from?

The preceding example was somewhat unusual. Attributes usually are not set by "hand."

- Intrinsic attributes usually are inferred from input
- Functions that create objects often assign additional attributes

Data frames

Data typically consist of several measurements per sampling unit. The measures usually are not of a single mode.

- **Data frames**, like spreadsheets, combine vectors that are all of the same length, but possibly different modes.

```
  gender antler.points pregnant
1   male             4       NA
2   male             5       NA
3 female            NA     TRUE
4 female            NA    FALSE
```

Documenting analyses and storing results

- Analyses typically involve not only data, but also methods and results.
- In R, all of these are represented by objects.
- Objects associated with an analysis typically cannot be stored in a single vector, matrix, or data frame.

Why might this pose a problem?

Components of a regression analysis

Even something as simple as a regression analysis is likely to involve *many* components!

- Data
- A model
- Degrees of freedom
- Parameter estimates
- A model likelihood
- Hypothesis tests
- Components of variation, R^2
- Residuals, fitted values, other diagnostics

Tracking many associated objects separately is burdensome and likely to lead to errors!

Lists

All of the objects associated with an analysis can be combined in a **list**: essentially a vector with *no* restrictions on elements.

- Each element of a list is an object.
- Each element may be of *any* type.
- Elements may be of *different* types.

A single list may comprise expressions, functions, vectors of different lengths and modes, data frames, and even other lists!

Uses of lists

We will see that lists are exceptionally useful for at least 3 purposes:

- Applying functions repeatedly
- Storing complex input for analyses
- Documenting analyses and storing output

Complex ecological data

- Using just 3 modes to represent most ecological data (`character`, `numeric`, and `logical`) presents obvious limitations.
- Just as attributes can be used to define complex data structures (e.g., matrices and arrays), they can be used to specify complex interpretations of data.

Classes for data

Every object belongs to a **class**, which is either the mode or specified by an additional attribute. Classes tell R how objects should be interpreted. Note, for example, some of the many interpretations that can be attached to numbers:

Descriptor	Mode	Class(es)
Size, distance	`numeric`	`"numeric"`
Number	`numeric`	`"integer"`
Gender	`numeric`	`"factor"`
Age group	`numeric`	`"ordered" "factor"`
When	`numeric`	`"Dates"`
	`numeric`	`"chron" "dates" "times"`
	`numeric`	`"POSIXct"`

Functions revisited

Recall functions, which are "program" objects that automate and simplify routine tasks.

- Functions are invoked with a **function call** followed by **arguments** in parentheses.
- Inputs (arguments) control output (objects or graphics).

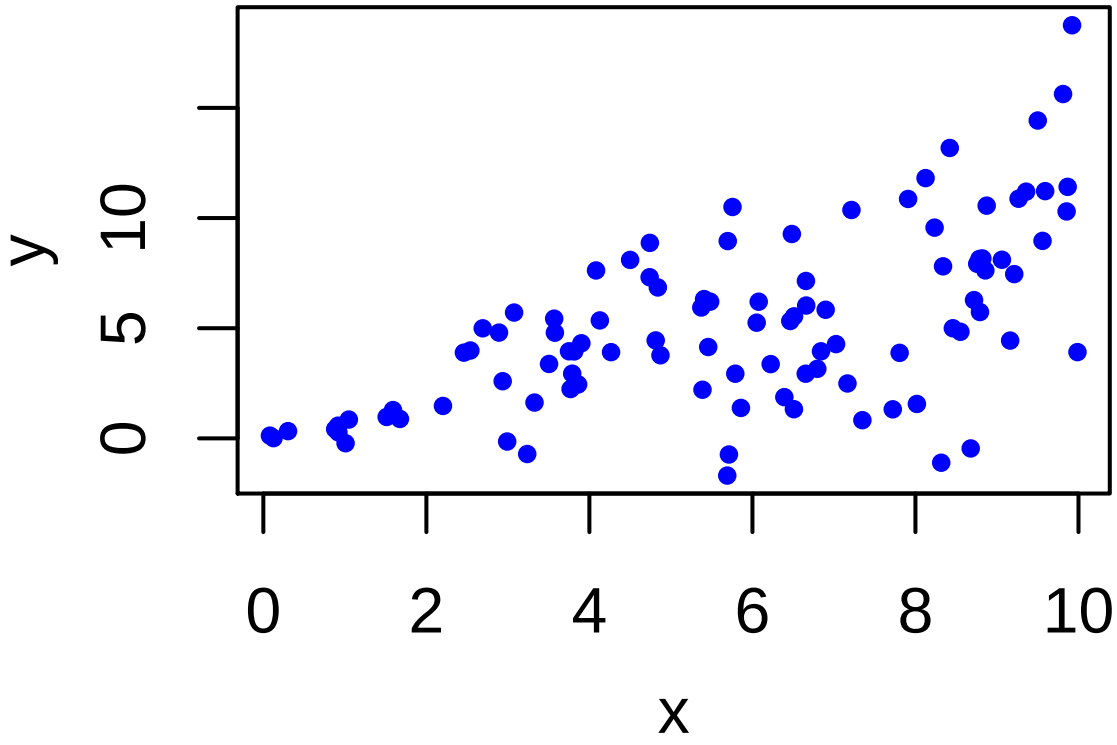
How do arguments control output?

- Arguments are objects
- As objects, arguments have modes, classes, and possibly other attributes
- Control may be exerted directly by arguments, or indirectly, by attributes of arguments

An example of direct control

Direct control is superficially similar to other software packages:

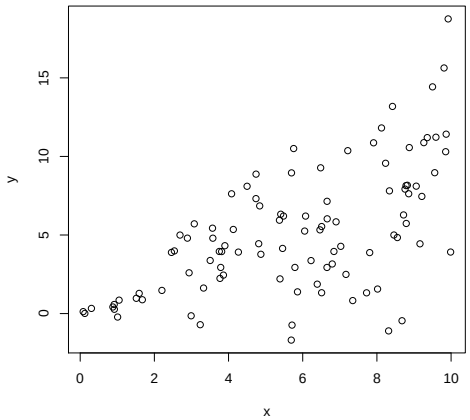
```
> plot(x, y, pch = 20, col = "blue")
```



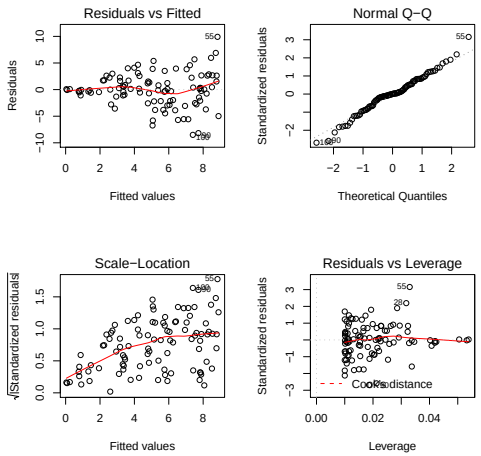
An example of indirect control

Indirect control is less obvious and accomplished with **methods** and **classes**:

```
> plot(X)
```




```
> plot(X)
```



Here, the same code produces a different result when the argument, `X`, is a data frame (left) than when it is a list that describes a linear model (right).

Methods and classes

- **Generic functions** include **methods** for different classes of input.
- Output produced by a generic function depends on the class of input.
- Methods and classes are *essential* to the power of R.
  - Many functions anticipate needs by producing results that are conditional on characteristics of input.
  - Careful coding of data is essential!

Methods for a given class: `methods()`

Some methods for the class lm:

```
> methods(class = "lm")

 [1] "add1.lm"            "alias.lm"
 [3] "anova.lm"           "case.names.lm"
 [5] "confint.lm"         "cooks.distance.lm"
 [7] "deviance.lm"        "dfbeta.lm"
 [9] "dfbetas.lm"         "drop1.lm"
[11] "dummy.coef.lm"      "effects.lm"
[13] "extractAIC.lm"      "family.lm"
[15] "formula.lm"         "hatvalues.lm"
[17] "HTML.lm"            "influence.lm"
[19] "kappa.lm"           "labels.lm"
```

Methods for a generic function: `methods()`

Some methods for plot():

```
> methods(plot)

 [1] "plot.acf"            "plot.agnes"
 [3] "plot.areg"           "plot.areg.boot"
 [5] "plot.aregImpute"     "plot.biVar"
 [7] "plot.curveRep"       "plot.data.frame"
 [9] "plot.Date"           "plot.decomposed.ts"
[11] "plot.default"        "plot.dendrogram"
[13] "plot.density"        "plot.diana"
[15] "plot.drawPlot"       "plot.ecdf"
[17] "plot.factor"         "plot.formula"
[19] "plot.gbayes"         "plot.hclust"
```

Expressions

- Statements submitted to the R console for evaluation are called **expressions**.
- Results *usually* are returned to the R console unless they are **assigned** to objects.
- Virtually every element of any analysis (including data, tools, and results) is an object.

Objects

- **Vectors** are the fundamental units of data organization.
- **Matrices** and **arrays** generalize vectors to 1 or more dimensions.
- **Data frames** generalize matrices to include multiple types of data.
- **Lists** combine any number of objects of any type.
- **Functions** are program objects that automate routine tasks.

Attributes

- **Modes** are the most basic descriptors of data.
- Elements of vectors, matrices, and arrays must all be of the same mode.
- Most ecological data are represented by just 3 modes (**numbers**, **character strings**, and **logical values**).
- **Classes** are attributes that convey additional information about objects and the way they should be interpreted.

Functions

- Functions are controlled, either directly or indirectly, by **arguments**.
- **Generic functions** provide **methods** for different classes of input.
- Methods produce results that are specific to a particular class of input.
- Coding data with appropriate classes is an essential component of R programming!

Acknowledgements

These workshop notes describe the use of software developed and documented by the R Development Core Team. The correct citation for R is as follows:

> *R Development Core Team (2008). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.*

The use of trade, product, or firm names does not imply endorsement by the U.S. Government.