Session 3: Reading & Writing to files, Manipulating R objects

Foundations of Quantitative Ecology (EEOB 8896.11)

Paul J. Hurtado
(hurtado.10@mbi.osu.edu)

Mathematical Biosciences Institute (MBI)
The Ohio State University

Last compile: September 4, 2013

Interactive session

Open R, and a new script, to explore R syntax by example.

Interactive session

Open R, and a new script, to explore R syntax by example.

```
c(1,2,3,2+2) ## a vector (semicolon optional, but good form)

## [1] 1 2 3 4

c(1:4+2, 0, (1:4)+2, 0, 1:(4+2)) ## When in doubt, use parentheses!

##  [1] 3 4 5 6 0 3 4 5 6 0 1 2 3 4 5 6

seq(1,5,length=9) ## see ?seq for by=0.5 vs length=9 arguments.

## [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0

## Generate data for y=5+5*x, x=1:10, observation error sd=10
set.seed(9413) ## Ensure we all have the same "random" data
xydata = data.frame(x=1:10, # ?data.frame for details.
                    y=rnorm(10, mean=5+5*(1:10), sd=10))
```

Inside R Objects

Vectors, lists, data.frames, ...

```
x =  xydata$x   ## See course website for "=" vs "<-"!
y <- xydata$y;  ## Call variable alone to display value
## Alternatively, we could have used attach(xydata)
##
y[1]        # first element

## [1] 17

y[c(2,4,5)] # second, fourth and fifth elements

## [1] 15.43 22.94 38.25

x>4         # a vector of logical values

##  [1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE

y[x>4]      # subset all the "TRUE" elements

## [1] 38.25 20.49 32.43 37.59 42.89 57.03
```

Inside R Objects

```
xydata[2,]   # As a matrix: row2; all columns

##   x     y
## 2 2 15.43

head(xydata,2) # See ?tail

##   x     y
## 1 1 17.00
## 2 2 15.43

names(xydata)

## [1] "x" "y"

dim(xydata)

## [1] 10  2

c(nrow(xydata),ncol(xydata))

## [1] 10  2
```

Inside R Objects

Basic types: logical, integer, real, complex, string/character, raw
Functions like `plot()` query `class`, `type` to determine plotting routine.

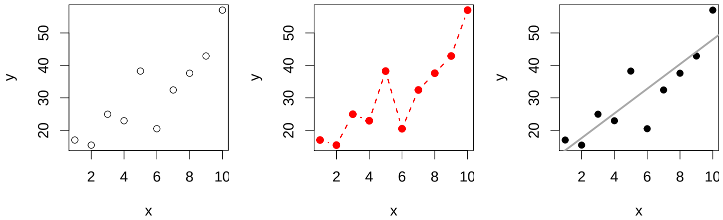
```
## What kind of object?
class(xydata)

## [1] "data.frame"

typeof(xydata)

## [1] "list"

mode(xydata)

## [1] "list"
```

```
## What kind of object?
class(x)

## [1] "integer"

typeof(x)

## [1] "integer"

mode(x)

## [1] "numeric"
```

See ?plot, ?mode, `?data.frame`, `?list`, `?unlist`, `?matrix`, `?as`.
Also: `http://stackoverflow.com/questions/6258004/`,
`http://cran.r-project.org/doc/manuals/r-release/R-lang.html#Objects`

Base Graphics

```
plot(x, y)  ## See ?par for plot arguments like...
plot(x, y, type = "b", lty = 2, lwd = 2, pch = 19, col = "red")
plot(xydata, pch = 19)
fit = lm(y ~ x, data = xydata)  ## linear regression
abline(fit, lty = 1, lwd = 3, col = "darkgray")  ## See ?lm, ?abline
```



Ex 1: What does `summary(fit)` do? `plot(fit)`?

Ex 2: Run `demo(graphics)`

Ex 3: See CRAN Task View on graphics packages.

Ex 4: Install `lattice`. Load it, then demo(lattice). See `ggplot2`.

Reading & Writing Files

Writing Files:

```
# WRITING files: formatting matters!
write.table(xydata, "xydata-table.csv", sep = ",",
            col.names = TRUE, row.names = FALSE)
write.csv(xydata,"xydata.csv")
# Alternatively, we can save the whole R object:
save(xydata, file="xydata.Rdata") ## See ?load to read *.Rdata
dir(all.files=TRUE)  ## view directory contents
```

Reading & Writing Files

Writing Files:

```
# WRITING files: formatting matters!
write.table(xydata, "xydata-table.csv", sep = ",",
            col.names = TRUE, row.names = FALSE)
write.csv(xydata,"xydata.csv")
# Alternatively, we can save the whole R object:
save(xydata, file="xydata.Rdata") ## See ?load to read *.Rdata
dir(all.files=TRUE)  ## view directory contents
```

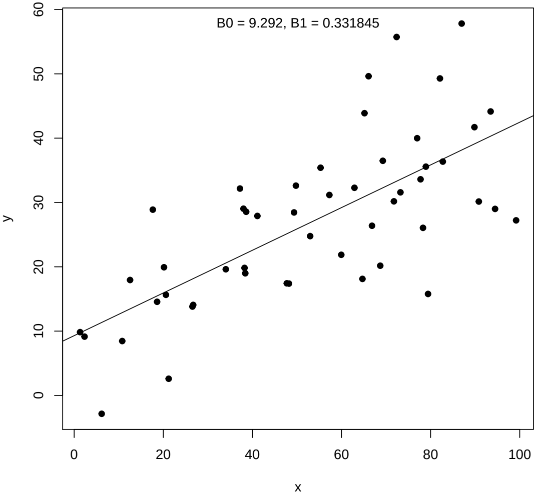
Reading files:

```
rm(xydata) # remove xydata from the workspace
# READING files: formatting REALLY matters!
xydatatable = read.csv("xydata-table.csv")
xydatacsv  = read.csv("xydata.csv")
# R objects load directly into memory
load("xydata.Rdata") ## See ?load for details.
## View contents by typing object names, or...
edit(xydatacsv)
```

Exercise: Modify the `write.csv()` call above to exclude the row names from the first column of `xydata.csv`.

LinSimScript.R

```r
##  Script to simulate simple linear model output,
##  recover model parameters by fitting simulated data,
# Simulation model
SimFunc <- function(x=sort(runif(50, 0, 100)), B0=10, B1=0.3, sd=10, seed=NULL)
    set.seed(seed) # Nothing happens if seed=NULL
    data.frame(x = x, y = rnorm(length(x), B0 + B1*x, sd)) ## Same as
   #data.frame(x = x, y = B0 + B1*x + rnorm(length(x), 0, sd))
}
# Generate data and plot it
xy=SimFunc(seed=1)
# Fit the model ...
fit = lm(y~x,data=xy)
# ... and plot.
plot(xy, pch=20, cex=1.4)
abline(fit)
text(mean(range(xy$x)), max(xy$y),
     paste("B0 = ",signif(fit$coefficients[1],4),
         ", B1 = ",signif(fit$coefficients[2]),sep=''))
summary(fit)
```


B0 = 9.292, B1 = 0.331845
y
0
10
20
30
40
50
60
x
0
20
40
60
80
100

Exercises:

1. Rewrite the script to use various non-normal errors. How good is the fit?
2. Write a `for` loop to save 1000 parameter estimates from 1000 simulated data sets. Plot the results as histograms.