Introduction to R

- Open-source implementation of the S language
  - Free; can see source code; UNIX/PC/Mac
  - Scaled down vs S-PLUS (little garbage; not all functions are available yet)
  - Some improved control over graphics
    `xlab=expression(hat(theta))`
- Static memory model
  - All data/functions in **`.RData`**
  - All read into RAM:
    `?Memory`
    `~/.Renviron`
  - Issue re multiple BATCH processes
  - **`save.image()`**
- Run from within **emacs**
  - ESS line in **`~/.emacs`**
  - **`M-x R`** [enter starting directory]
  - Cut-and-paste from another file
- Miscellaneous
  - **`q()`**
  - **`help.start()`** **`help()`** **`?`**
  - **`library(ctest)`**
  - **`nice +19 R BATCH ifile ofile &`**

Data modes

- numeric — is.numeric(), as.numeric()
- character
- logical
- factor
- ordered
- complex
- NULL

Data structures

- vector — all elements must have same mode
- matrix — "
- array — "
- list — ordered set; possibly different modes
- data.frame — list of vectors each of same length (has characteristics of list and matrix)

Missing values

NA — is.na()

Assignment

```
a <- 1
```

Generating data

Vectors: **c() : seq() rep()**

```
a <- c(1, 3, -5.6, 21.89, 0.56)
b <- 1:20
d <- -5:-30
e <- seq(5, 10.6, length=50)
f <- seq(5, -38, by=-3)
g <- rep(2, 50)
h <- rep(c(1,2), 50)
i <- rep(c(-1,3), c(30,40))
j <- runif(50,2,20)
k <- rnorm(100,10,3)
rpois, rgamma, rbinom, …
```

Matrices: **matrix()**

```
A <- matrix(rnorm(100,10,3),ncol=5)
B <-matrix(c(1,3,5,10),ncol=2,byrow=T)
```

Lists: **list() vector("list",n)**

```
x <- list(c(1,5,10,1:20),letters[1:8])
y <- vector("list",5)
```

Data.frame **data.frame() read.table()**

```
a <- read.table("mydata.csv", sep=",",
                    header=TRUE)
```

Operators

```
+   -   *   /   ^
```

`%/%` (modulo) `%%` (remainder)

```
==    <     >     <=    >=    !=
```

```
&&    ||    &     |     !
```

Math functions

```
sqrt  log   exp   log2  log10
abs   sin   cos   tan   asin
sinh  asinh   gamma   lgamma
beta  choose  lchoose
```

Vectorized calculations

```
x <- 1:10
y <- 11:20
x + y; sqrt(x)
x^(-log(y)) + 1

x <- rep(c(TRUE, FALSE), 2)
y <- rep(c(TRUE,FALSE), c(2,2))
!x
x & y
x && y
```

Matrix operators/functions

```
%*%       t()       solve()  chol()
diag()    eigen()   outer()
qr()      svd()     kronecker()
```

Functions

```
mean(x)
mean(x,na.rm=T)
sd(x)
var(x)
runif(50, 2, 20)
rnorm(100, sd=3)
```

Subsetting

Vectors

```
x <- rnorm(26, 0, 3)
y <- rnorm(26, 5, 3)
x[c(1,5,10)]
x[-(1:12)]
x[x < 1]
x[x <= 1 & y > 3 & is.na(x)]

names(x) <- letters
x[c("a","b","f")]
```

Subsetting

Matrices

```
A <- matrix(rnorm(100),ncol=5)
A[1:5,]
A[,c(2,5)]
A[,-(3:4)]
A[A[,1]<5,]
A[14:19,c(1,3)]
dim(A); length(A)
dimnames(A)<- list(as.character(1:20),
          paste("col",1:5,sep=""))
A[,"col3"]
```

Lists

```
v <- list(c(1,5,20),c("aardvark",
          "antelope"),
           matrix(runif(100),ncol=4))
v[[1]]
v[1:2]
names(v) <- c("number","animal","mat")
v$number
v$mat[1:5,]
```

Data.frames

Use either the matrix or list version

Programming

Conditional statements

```
if( … ) {
}
else {
}
```

Loops

```
for(i in 1:ncol(x)) {
}
while(any(abs(a) > tol)) {
}
repeat {
   if(…) break;
}
```

Vectorized calculations $\leftrightarrow$ **loop avoidance**

```
apply(mat, 2, quantile)
t(apply(mat, 1, quantile))
lapply(li, function(a,b)
                  mean(a[a>b]), val)
sapply(li, function(a,b)
                  mean(a[a>b]), val)
a <- 1:100
b <- sample(1:3,100,repl=T)
tapply(a,b,sd)
sapply(split(a,b),sd)
```

Graphics

- **plot()**
- **lines()**
- **points()**
- **boxplot()**
- **barplot()**
- **hist()**
- **pairs()**
- **legend()**

Special control

- **par()**
  - **las**
  - **xpd**
  - **cex**
  - **mfrow**
  - **mfcol**
  - **lty**
  - **lwd**
  - **new**
  - **usr**
  - **xaxt**
  - **yaxt**
  - **ask**

Miscellaneous

```
ls()
objects()
attach("/home1/…/.RData")
search()
objects(2)
objects(2,pattern="tab")
objects(4,pattern="^na[.]")

.First <- function() {
   attach("…/.RData")
   options(object.size=100000)
   library(ctest,mva,modreg)
}

source("myfile.R")
unix.time(a <- rnorm(100000))

x <- matrix(rnorm(100),ncol=5)
nrow(x)  ncol(x)   row(x)  col(x)

sort()   order()   rev()
pgamma() dgamma()  qgamma()  rgamma()
cat()    grep()    nchar()   paste()
substring()        abbreviate()

unlist()       sample()
```

Examples

$k_{ij} \sim \text{Poisson}(\lambda_i)$ for $j = 1, \ldots, n_i$

$x_{ij} \mid k_{ij} \sim \text{normal}(\alpha + \beta\, k_{ij}, \sigma^2)$

```
sim.np <-
function(para=c(0:3 + 0.1,   10,5,2),
         n=c(30,30,30,33))
{
  if(length(para) != length(n) + 3)
    stop("length(para) != length(n)+3")

  n.grp <- length(n)
  lambda <- para[1:n.grp]
  absig <- para[-(1:n.grp)]

  k <- rpois(sum(n), rep(lambda,n))
  x <- rnorm(sum(n), absig[1] +
                k*absig[2],  absig[3])

  list(k=k, x=x)
}
```

Examples

```
# Fisher's (exact) test
fisher  <- function(tab,n.sim=1000)
{
  bot0 <- sum(lgamma(tab+1)) # observed

  bot <- 1:n.sim
  a <- list(rep(row(tab),tab),
            rep(col(tab),tab))
  for(i in 1:n.sim) {
      a <- lapply(a,sample)
      tab2 <- table(a)
      bot[i] <- sum(lgamma(tab2+1))
  }
  mean(bot0 < bot)
}

x <- matrix(c(2,1,3,4,
              4,1,1,6,
              0,1,0,7), ncol=4, byrow=T)

fisher(x)
fisher(x,n.sim=500)
```

Statistical models

```
lm()     aov()       glm()
nls()    [ library(nls) ]
tree()   [ library(tree) ]

mydata <- data.frame(x1=rnorm(100),
                     x2=rnorm(100))
mydata$y <- 0.5*mydata$x1 -
            mydata$x2+rnorm(100,0,0.2)
mydata$y[sample(1:100,10)] <- NA

o1 <- lm(y ~ x1 + x2, data=mydata)
o2 <- lm(y ~ -1 + x1 + x2,data=mydata)
summary(o2)
plot(o1$fitted,o1$resid)

na.action = na.fail na.omit na.include

mydata$y2 <-rbinom(100,5,
   exp(mydata$x1)/(1+exp(mydata$x1)))
mydata$y2 <- mydata$y2 / 5
mydata$n <- rep(5,100)
o3 <-glm(y2~x1+x2,
   family=binomial(link=logit),
   data=mydata,weights=n)
```