S-Plus workshop

7-9 and 14-16 January

students.washington.edu/arnima/s

Syllabus

Tue 7	**Introduction** Import data, summarize, regression, plots, export graphs
Wed 8	**Basic statistics** Descriptive statistics, significance tests, linear models
Thu 9	**Linear models** Anova, LM, GLM, loess
Tue 14	**Graphics** Types, multipanel, export graphs
Wed 15	**Data manipulation** Data objects, describe, extract, sort, manipulate
Thu 16	**Programming** Functions, import/export, project management, packages

Today: Data manipulation

1 Data objects
storage mode, vector, matrix, data frame, list

2 Description, coercion
dim, length, names, is.*, as.*

3 Comparison, extraction, sorting
<, ==, >, &, |, match, x[i], x[i,j], x[[i]], x$a, sort

4 Element manipulation
rounding numbers, working with character strings

Data objects

Storage mode

```
x <- TRUE              # logical
storage.mode(x)

x <- 1                 # double
storage.mode(x)

x <- as.integer(1)     # integer
storage.mode(x)

x <- "ok"              # character
storage.mode(x)
```

Vectors and factors

```
v <- c(1,10)
n <- c("small","big")
names(v) <- n
v <- 1:10
seq(1, 10, length=100)
v <- seq(1, 10, by=3)
c(v, 1/v)
rep(v, 4)
rep(v, each=4)
x <- c("Mid", "South", "Mid", "North", "South", "Mid")
y <- factor(x)
z <- ordered(x, levels=c("South", "Mid", "North"))
```

Matrices and arrays

```
m <- matrix(c(v,1/v), ncol=2)
my.row.names <- c("i", "ii", "iii", "iv")
my.col.names <- c("simple", "inverse")
dimnames(m) <- list(my.row.names, my.col.names)
t(m)
m <- matrix(c(v,1/v), ncol=2,
            dimnames=list(my.row.names, my.col.names))
cbind(v, 1/v)
rbind(v, 1/v)
array(c(v,1/v), dim=c(4,2))
```

Data frames

```
d <- data.frame(simple=v, log=log(v), auto=5)
d <- data.frame(values=v, big=v>8, roman=c("i","ii","iii","iv"))
storage.mode(d$values)
storage.mode(d$big)
storage.mode(d$roman)
d <- data.frame(values=v, big=v>8, roman=I(c("i","ii","iii","iv")))
storage.mode(d$roman)
```

Lists

```
l <- list(values=v, big=v>8, roman=c("i","ii","iii","iv"))
storage.mode(l$roman)
```

Data frames vs. matrices/arrays

Data frame is the default choice for statistical analysis (mammals and cabbages are data frames)

It can store vectors of different modes, allows shorthand notation in lm(), xyplot(), etc.

I use matrices and arrays only inside computing-intensive functions, because they're fast

Describe objects

Storage mode of a vector

```
storage.mode(x)
storage.mode(v)
is.character(x)
is.numeric(x)
```

Object class

```
is.matrix(m)
is.data.frame(m)
is.data.frame(d)
is.matrix(d)
is.list(l)
is.list(d)
class(m)
class(d)
class(l)
```

Dimensions

```
length(v)
dim(d)
nrow(d)
ncol(d)
```

Get names

```
names(d)
row.names(d)
dimnames(d)
dimnames(m)
```

Set names

```
names(d) <- c("numbers", "large", "italian")
names(d)[1] <- "x"
```

Range and unique values

```
range(v)
range(m)
min(v)   # same as range(v)[1]
max(m)   # same as range(m)[2]
unique(x)
```

Size in memory

```
object.size(x)
x <- matrix(rep(pi,1e6), ncol=1e3)
object.size(x)
object.size(as.data.frame(x))
```

Object structure

```
n <- 1
attributes(n)
attributes(d)
attributes(x)
attributes(y)
attributes(z)
```

Coerce

Coerce elements between modes

```
as.character(z)
as.numeric(z)
as.character(10^(0:3))
```

Coerce objects between classes

```
unlist(l)
as.list(d)
as.data.frame(l)
as.data.frame(m)
as.matrix(d)
as.vector(m)
```

Compare

Simple comparison

```
x <- 4
y <- pi
z <- -4

x < y
x == abs(z)
y >= sqrt(x)
z != 4

(x < y) && (y > z)  # and
(x < y) || (y > z)  # and/or
```

Multiple elements

```
x <- rpois(20,3)
y <- 5.5*x - x^2 - 5.5
z <- data.frame(x,y)

x==1
y[x==1]
x > 3
y > 0

x>3 & y>0
x>3 & y>0
any(x>3 & y>0)
all(x>3 & y>0)
```

Single &| and double &&||

Use single &| to compare vector elements, usually inside []

```
# part of z where x<4 and y>1
z[x<4 & y>1, ]
```

Use double &&|| to test if something is true, usually in if()

```
if(is.numeric(x) || pi>3)
  print("Neither condition is true")
else
  print("One or both are true")
```

Find matches in a vector

```
x==3
match(3, x)    # first
which(x==3)    # all
```

Find identical values in two vectors

```
intersect(x, y)
```

Extract

From vector

```
n <- c("A","C","B")
n[1]
n[2:3]
n[c(TRUE, FALSE, TRUE)]
n[-1]
n[-c(1,3)]
row.names(m)[2]

nv <- c(one=10, two=20, three=30)
nv["20"]
```

From matrix

```
m[1,2]
m[2, ]
m[, 2]
m[-3,2]
m[2:3, 1:2]
m[2:3, 1]
m[,1][2:3]

m["ii", "simple"]
m["ii", ]
m[, "simple"]
```

From data frame

```
d$italian
d[,3]
d$numbers[2:3]
d[2:3,1]
d[,-3]

d[, "italian"]
d[3, "italian"]
```

From list

```
l$roman
l[[3]]
l["roman"]
```

Sort

Sort one vector

```
rev(n)
sort(n)
rev(sort(n))
order(n)
```

Sort two vectors

```
x <- rpois(20,3)
y <- 5.5*x - x^2 - 5.5
cbind(x,y)
plot(x,y)
sort(x)
cbind(sort(x), sort(y))  # wrong, they don't match
order(x)
x[order(x)]
y[order(x)]
cbind(x[order(x)], y[order(x)])
plot(cbind(x[order(x)], y[order(x)]), type="l")
```

Sort data frame

```
my.frame <- data.frame(abc=letters[1:20], x=x, y=y, z=1/(x*y))
my.frame[order(my.frame$y),]
```

Manipulate numbers and strings

Arni Magnusson

15 January 2003

Round numbers

```
round(rnorm(10), 2)
x <- seq(-5, 5, by=0.5)
round(x, 0)   # go to the even digit, IEEE standard
trunc(x)      # truncate at decimal, this is done by as.integer coercion
floor(x)      # go down
ceiling(x)    # go up
```

Formatting numbers

```
format(2^(1:10))
```

Pasting strings

```
a <- letters[1:10]
b <- 1:10
paste(a, b)
cbind(paste(a, b))
cbind(paste(a, b, sep=""))
cbind(paste(a, b, sep=" is number "))
cbind(paste("The letter", a, "is number", b))
p <- paste("The letter", a, "is number", b)
paste(a, b, sep="", collapse=",")
```

Substrings

```
nchar(p)
substring(p, 5, 12)
substring(p, 6, 8)
substring(p, 6, 8) <- "add"
substring(p, 5, 10)
substring(p, 5, 10) <- "character"
p
substring(p, 5, 10) <- "letter"
```

Replace substrings

```
#R: gsub("letter", "character", p)
strsplit(p, "is")
#R: chartr(old="s", new="z", p)
tolower(p)
toupper(p)
```

Find patterns in strings

```
grep("r 1", p)          # find elements that match pattern
p[grep("r 1", p)]       # find elements that match pattern
```

```
regexpr("er", p[1])
# locate first substring inside an element that matches pattern
regexpr("er", p)
# locate first substring inside elements that matches pattern
```

Subsetting painters data frame

```
library(MASS)
#R: data(cabbages, painters)
#S: cabbages <- cabbages
#S: painters <- painters
class(painters)
class(painters$School)
as.character(painters$School)
as.numeric(painters$School)
painters[,1:4]
painters[painters$School=="B",]   # factors behave more like strings...
painters[painters$School==2,]     # ...than numbers
painters[painters$School=="B", painters$Colour>=10]
```

Subsetting cabbage model input

```
lm(VitC~HeadWt, data=cabbages)
lm(VitC~HeadWt, data=cabbages, subset=(HeadWt>3 & Cult=="c39"))
```