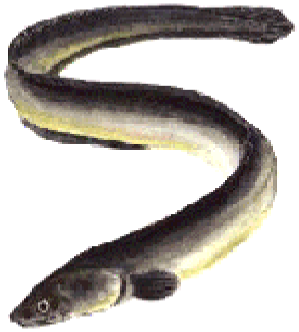


S-Plus workshop

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

(Minor) S-Plus limitations

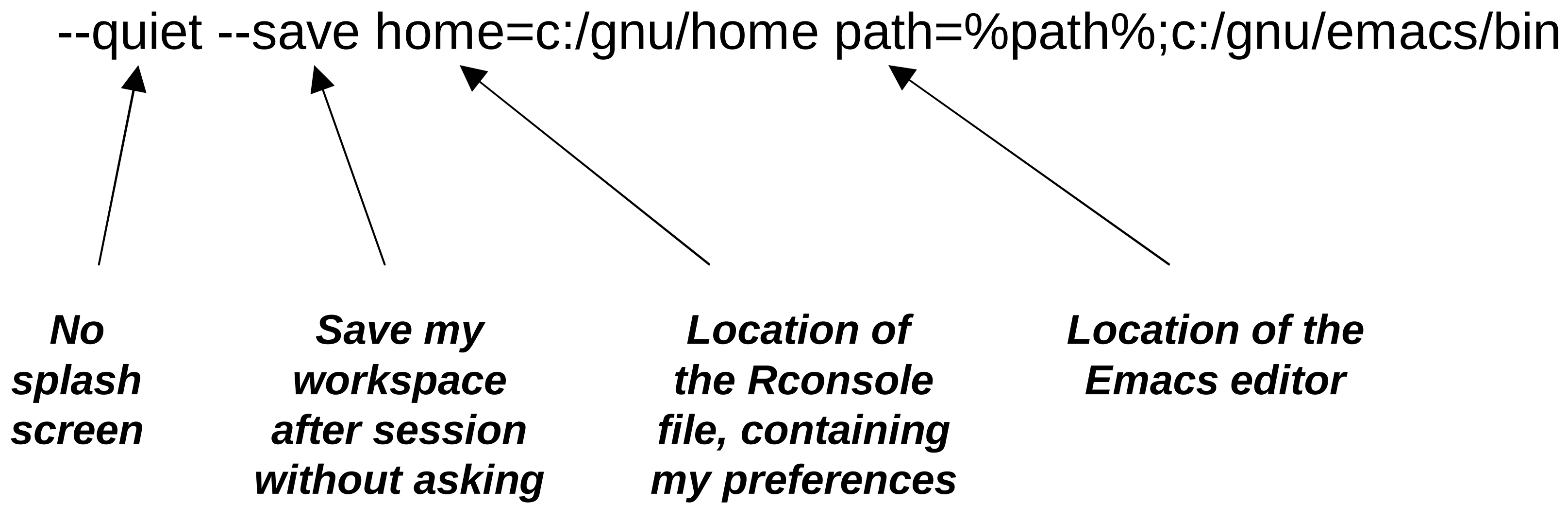
```
plot(mammals)                          # no plot.data.frame
boxplot(VitC~Date,data=cabbages)       # no boxplot.formula
ls.diag(cabbages.ancova)               # no ls.diag support for lm

coplot(VitC~HeadWt|Date*Cult, data=cabbages, panel=panel.smooth)
# crashed until we changed span from the default
```

If you think you can get your work done without the GUI, you’re probably better off switching to R now. This will become more as your programming needs increase.

R shortcut

Append this to your R shortcut, behind the .../bin/rgui.exe



```
--quiet --save home=c:/gnu/home path=%path%;c:/gnu/emacs/bin
```

No splash screen

Save my workspace after session without asking

Location of the Rconsole file, containing my preferences

Location of the Emacs editor

Today: Graphics

1 Traditional plot types

univariate, 2D, 3D, multivariate, object oriented

2 Trellis plots

univariate, 2D, 3D, multivariate

3 Graphical devices

on-the-fly, vector file, bitmap file, PDF file

4 Detail control

multipanel, fonts, colors, graphical parameters

Fetch data sets

```
library(MASS)
#R: data(cabbages, painters)
#S: cabbages <- cabbages
#S: painters <- painters
```

Traditional plots

Univariate

```
v <- cabbages$VitC
d <- cabbages$Date
hist(v)
barplot(table(v))
boxplot(v)
boxplot(split(v,d))
qqnorm(v)
```

2D scatter

```
x <- cabbages$HeadWt

y <- cabbages$VitC

plot(x,y)

plot(x, y, xlim=c(0,5), ylim=c(0,100), xaxs="i", yaxs="i",
     bty="L")

x <- rpois(100,1)

x

y <- rpois(100,1)

cbind(x,y)

plot(x,y)

sunflowerplot(x,y)
```

3D scatter

```
x <- runif(1000, min=-2, max=2)
y <- runif(1000, min=-2, max=2)
z <- cos(x)+sin(y) + rnorm(1000,s=0.1)
#R: install.packages("scatterplot3d")
#R: library(scatterplot3d)
#R: scatterplot3d(x,y,z, cex.symbols=0.5)
#S: brush(data.frame(x,y,z))
```

3D model surface

```
#R: library(modreg)

model <- loess(z~x+y)

xcoords <- seq(-2, 2, length=20)

ycoords <- seq(-2, 2, length=20)

grid <- expand.grid(x=xcoords, y=ycoords)

zvector <- predict(model, grid)

zmatrix <- matrix(zvector, nrow=length(xcoords))
```

3D model surface

```
contour(xcoords, ycoords, zmatrix)

contour(xcoords, ycoords, zmatrix, xlab="x", ylab="y",
        main="Loess model fit of cos(x)+sin(y) simulated data")

persp(xcoords, ycoords, zmatrix)

#R: persp(xcoords, ycoords, zmatrix, theta=45, phi=30,
#R:       expand=0.75, shade=0.5, ticktype="detailed")
```

Multivariate

```
interaction.plot(cabbages$Cult, cabbages$Date, cabbages$VitC)

painters  # columns 1-4 are numeric, column 5 is a factor

#S: faces(as.matrix(painters[,1:4], labels=row.names(painters))

stars(painters[,1:4], draw.segments=T, key.loc=c(16,1))

stars(painters[,1:4], full=F, key.loc=c(16,1))

pairs(painters)

parcoord(painters[,1:4])

my.thermos <- cbind(width=0.1, height=1, temp=(1:5)/5)

my.boxes <- cbind(w=0.5, h=4:8, up=abs(rnorm(5)),
                  dn=abs(rnorm(5)), md=0.5)

symbols(rnorm(5), rnorm(5), thermometers=my.thermos)

symbols(rnorm(5), rnorm(5), boxplots=my.boxes, bg=8)
```

Object oriented

```
methods(plot)
```

Add points, lines

```
plot(1:10, 1:10)

points(5,2, pch=8)

lines(c(5,5), c(2,5), col=2)

segments(5,2, 5,5)

abline(h=8, lty=2)
```

Add lines, text

```
qqnorm(cabbages$VitC)

qqline(cabbages$VitC)

text(0, 50, "bang", srt=20)

title(main="\n\nof sorts")

mtext("007", side=1, line=1)

identify(qqnorm(cabbages$VitC,plot=F), labels=rep("ouch",60))
```

#R: Add math notation

```
plot(rnorm(100))

my.expression <- expression(paste("Random draws from the  ",
  frac(1,sigma*sqrt(2*pi))," ",e^{frac(-(x - mu)^2,
  2 * sigma^2)}, " distribution"))

title(main=my.expression)

?plotmath
```

Add grid, polygon, legend

```
plot(-5:5, -5:5)

#R: grid()

#S: grid.render(grids=list(x=-5:5,y=-5:5))

polygon(-3:3, tan(-3:3), col=3)

#R1: legend(3, -3, c("One","Two"), pch=c(1,24), bg="white",
            pt.bg=c(0,3))

#R2: legend(3, -3, c("One","Two"), pch=c(1,17), bg="white",
            col=c(1,3))

#R2: legend(3, -3, c("One","Two"), pch=c(1,2))

#S:  legend(3, -3, c("One","Two"), marks=c(1,2), cex=2, bg=-1)
     # almost
```

Trellis plots

Univariate

```
v <- cabbages$VitC
d <- cabbages$Date
bwplot(~v)
bwplot(d~v)
dotplot(~v)
dotplot(d~v)
stripplot(~v)
stripplot(d~v)
```

Univariate

```
histogram(~v)

histogram(d~v)

densityplot(~v)

densityplot(~v|d)

qqmath(~v)

qqmath(~v|d)
```

2D scatter

```
x <- cabbages$HeadWt
y <- cabbages$VitC
xyplot(y~x)
xyplot(y~x|d)
xyplot(y~x, groups=d, panel=panel.superpose)
coplot(y~x|d)
```

3D scatter

```
x <- runif(1000, min=-2, max=2)
y <- runif(1000, min=-2, max=2)
z <- cos(x) + sin(y) + rnorm(1000, s=0.1)
cloud(z~x+y)
d <- c(rep("Many",990), rep("Few",10))
data.frame(x,y,z,d)
cloud(z~x+y|d, cex=0.5)
```

3D model surface

```
contourplot(zvector~grid$x+grid$y)

levelplot(zvector~grid$x+grid$y)


wireframe(zvector~grid$x+grid$y)

zvector

ok <- !is.na(zvector)

wireframe(zvector[ok]~grid$x[ok]+grid$y[ok])

wireframe(zvector[ok]~grid$x[ok]+grid$y[ok], drape=T,
          scales=list(arrows=FALSE), xlab="X", ylab="Y", zlab="Z")


# Examples from ?contourplot, ?levelplot, and ?wireframe
```

Multivariate

```
splom(~painters)
splom(~painters|painters$School, pscales=0)
splom(~painters[,1:4]|painters$School, pscales=0)
parallel(~painters[,1:4])
parallel(~painters[,1:4]|painters$School)
```

Roll your own

```
my.panel <- function(x, y, ...)
{
  panel.grid()
  panel.xyplot(x, y, ...)
  panel.lmline(x, y, ...)
  #R: ltext(mean(x), 40, mean(x))
  #R: ltext(1.1, mean(y), mean(y))
  #S: text(mean(x), 40, mean(x))
  #S: text(1.1, mean(y), mean(y))
}


xyplot(VitC~HeadWt|Date*Cult, data=cabbages, panel=my.panel,
       pch=16, lwd=2)
```

Pros and cons of Trellis

Extremely useful for exploring multivariate data

Considerable programming is required to change parts of the plot

I recommend learning both traditional and trellis graphics

Graphical devices

On-the-fly devices

```
#S: graphsheet()             # default device in S-Plus
#S: graphsheet(pages=T)      # cycle through plots with Ctrl-PgUp and Ctrl-PgDn
#R: windows()                # default device in R
#R: windows(record=T)        # cycle through plots with PgUp and PgDn
trellis.device()             # default trellis device
trellis.device(color=F)      # black and white trellis plots
```

Export to vector file (quality)

Vector file format retains smooth edges when imported into documents

```
postscript()          # global standard, not supported in MS Office 97 and older
#R: win.metafile()    # MS Office 97 vector file format in R
#S: wmf.graph()       # MS Office 97 vector file format in S-Plus
```

Export to bitmap file (editable)

Bitmap file format creates rough edges, but can be edited in graphics software

```
#R: png()
# compact file size, supported by MS Office, browsers, etc.

#R: bmp()
# large file size, but editable in MS Paint

#R: jpeg()
# unsharpens edges, only recommended if PNG file is too large

#S: graphsheet(file="GIF")
# similar to PNG, file="BMP" and file="JPG" also work
```

Export to PDF file (distribute)

I prefer distilling my own PDFs from postscript files, but this could be used to automate reports

```
#R: pdf()

#S: pdf.graph()
```

Trellis export

```
trellis.device(device="postscript")   # or any other device
```

Device management

```
dev.list()       # List open devices
dev.cur()        # Return name and number of current device
dev.set(which)   # Switch to device
dev.off()        # Turn off current device (write file if export device)
```

Detail control

Create a plot from scratch

```
plot(0, axes=F, type="n", xlab="", ylab="", xlim=c(-5,5),
ylim=c(-5,5))

points(rnorm(5), rnorm(5), pch=15, cex=1.5)

points(0, 0, cex=20)

axis(1)

axis(2)

axis(2, at=0, labels=0, tck=0.01)

axis(4, at=c(-2,2), labels=c(7,3), las=1, tck=-0.01)
```

Create a plot from scratch

```
box()
title(main="From scratch")
title(xlab="X label")
#R: title(ylab=list("Y label", cex=0.75, font=3, col=8))
#S: title(ylab="Y label", cex=0.75, font=3, col=8)
```

Multipanel layout - One size

```
par(mfrow=c(3,4))

for(i in 1:12) plot(rpois(100,i), rpois(100,i))
```

Multipanel layout - Different sizes

```
fig1 <- function()
{
  par(fig=c(0.1,0.6, 0.4,0.9))
  plot(1)
  par(fig=c(0.7,0.9, 0.5,0.9), new=T)
  plot(2)
  par(fig=c(0.1,0.9, 0.1,0.3), new=T)
  plot(3)
}
```

Fonts

Default device:

R supports styles and math expressions
S-Plus supports fonts (incl. symbols)

Postscript/PDF:

R supports fonts, styles, and math expressions
S-Plus supports fonts and styles

*** Create a temporary folder c:/spit on your computer

R fonts

```
plot(0:5, 0:5, type="n")
types <- c("Plain","Bold","Italic","Bold italic")
text(rep(2.5,4), 1:4, types, font=1:4, cex=3)


spitR <- function(fontnames=c("Courier","Helvetica","Times"))
{
  for(f in 1:length(fontnames))
  {
    filename <- paste("c:/spit/R", f, ".pdf", sep="")
    pdf(filename, family=fontnames[f], 11, 8.5)
    plot(0:5, 0:5, type="n", xlab=expression(tan(pi)))
    styles <- c("plain","bold","italic ","bold italic")
    for(s in 1:4)
      text(2.5, 5-s, paste(fontnames[f], types[s]), font=s, cex=4)
    dev.off()
  }
}
spitR()
```

S-Plus fonts

```
plot(0:5, 0:5, type="n")

winfonts <- data.frame(code=c(1:3,8),
  row.names=c("Arial","Times New Roman","Courier New","Symbol"))

for(i in 1:4)

  text(2.5, i, row.names(winfonts)[i], font=winfonts$code[i], cex=3)



psfonts <- data.frame(plain=c(1,2,3,13),
                      italic=c(4,7,10,13),
                      bold=c(5,8,11,13),
                      bold.italic=c(6,9,12,13),
                      row.names=c("Helvetica","Courier",
                                  "Times","Symbol"))
```

S-Plus fonts

```
spitS <- function(ftable)
{
  for(f in 1:nrow(ftable))
  {
    filename <- paste("c:/spit/S", f, ".pdf", sep="")
    pdf.graph(filename, T, 11, 8.5)
    plot(0:5, 0:5, type="n", xlab="tan( )", font=f)
    mtext("p", side=1, line=3, at=2.57, font=13)
    for(s in 1:4)
    {
      this.fontname <- paste(row.names(ftable)[f],names(ftable)[s])
      text(2.5, 5-s, this.fontname, font=ftable[f,s], cex=4)
    }
    dev.off()
  }
}

spitS(psfonts)
```

Colors

```
wow <- function(col)
{
  opar <- par(fig=c(0,1,0.1,1))
  x <- rep(1, length(col))
  barplot(x, axes=F, border=F, space=F, col=col,
          names=as.character(col), las=2, cex.names=0.8)
  par(opar)
}
```

R colors

```
wow(1:20)

colors()

wow(colors()[runif(20,1,657)])

rgb(red=1, green=0, blue=1)

wow(rgb(seq(0,1,length=20), 0, 0))

hsv(h=0.6, s=0.9, v=0.7)

wow(hsv(seq(0,1,length=500), 1, 1))

wow(hsv(seq(0.7,.95,length=5), 1, 1))

wow(terrain.colors(500))

wow(terrain.colors(5))
```

S-Plus colors

```
wow(1:20)
graphsheet(color.table="0,0,255|255,0,0")
wow(1:20)
```

Graphical parameters

Set parameters with par(mypar=x), or as a function argument like plot(x, y, mypar=x)

Store old parameters with old.values <- par(mypar=new.value)

Get parameters with par()$mypar

```
?par   # Important source of information about graphics
```

Graphical parameters

Plot details

```
axes, bty, las, mgp, xaxs, yaxs, xlim, ylim, tck  # format axes
xlab, ylab, main                                  # specify labels
type                                              # specify type
```

Element details

```
cex, col, font, srt  # format text
col, lty, lwd        # format line
cex, col, pch        # format plot character
```