Wab 3

Simulations

generating random numbers

rnorm	generates random Normal variates
dnorm	evaluates the Normal probability density
pnorm	evaluates cumulative distribut. function
rpois	random Poisson variates

Common prefixes:

d for density
r for generating
p cumulative distribution
q for quantile

rnorm (n, mean =0, sd = 1)

set.seed(1) - to insure ~~per~~ reproductivity

Poisson data

```
rpois(10, 1), rpois(10, 2)   rpois(10, 20)

ppois(2, 2)
    0.6767  # Pr(x <= 2)

ppois(4, 2)      # Pr(x <= 4)
```

Linear models

$y = \beta_0 + \beta_1 x + \varepsilon$ | $\beta_0 = 0.5, \ \beta_1 = 2$

$\varepsilon \sim N(0, 2^2)$

$x \sim N(0, 1^2)$

```
set.seed(20)

x <- rnorm(100)
e <- rnorm(100, 0, 2)
y <- 0.5 + 2 * x + e

plot(x, y)
```

rbinom (100, 1, 0.5)

↙ ↓

1 and 0 5 ...

Generalized Linear Model

$Y \sim \text{Poisson}(\mu)$

$\log \mu = \beta_0 + \beta_1 X$

$\beta_0 = 0.5 \quad \beta_1 = 0.3$

```
set.seed(1)
x = rnorm(100)
log.mu = 0.5 + 0,3 * x
y <- rpois(100, exp(log.mu))
    (discrete, nat.)
```

Random sampling

sample (1:10, 4)
sample (letters, 5)
sample (1:10) ← permutation

sample (1:10, replace = TRUE)

Plotting

base packages (library(package.name))

graphics: plot, hist, boxplot,

lattice: xyplot, bwplot, levelplot, based on grid

~~grServices~~
grDevices: X11, PDF, PostScript, PNG, etc

Making a plot

1. What device? (x11, windows, pdf, etc)

2. For viewing temporary or for using?

3. Is the amount large?

4. Do I need to resize?

5. ~~Grid~~ base or ~~lattice~~?

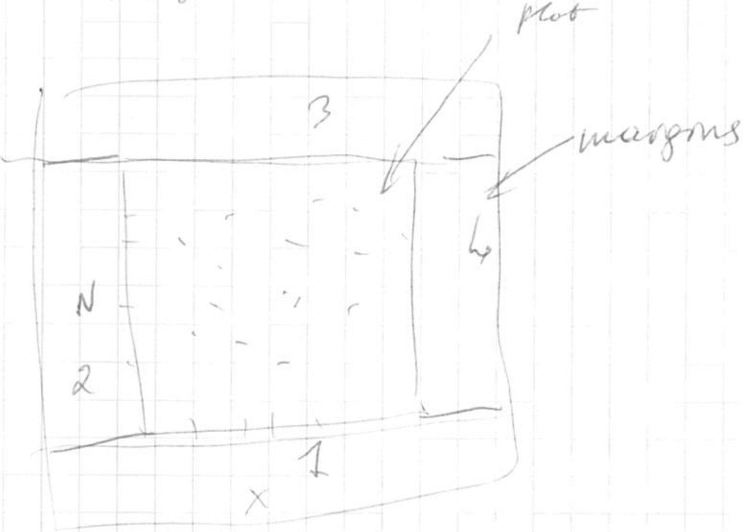
par - global graphical parameters.

?par - help

Copying:

dev.copy - copy to another device

dev.copy2pdf



```
X > rnorm (100)
y>rnorm (100)

par(mar = c(2,2,2,2)
plot(x,y)
```

Changes margins

↙ solid circles

plot(x, y, pch - 20)

2 ← Δ

built in

example (points) — Demos!

title("Scatterplot") adds a title

text(-2, -2, "Label")

legend ("topleft", legent = "Data",
pch=20)

adding a lines:

fit <- lm(y ~ x)

abline (fit)

abline (fit, lwd=3) , color="blue"

↑

thick

plot (x, y, xlab = "Weight", ylab = "Height",

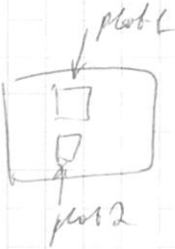
main = "Scatterplot", pch = 20)

legend ("topright", legend = "Data", pch = 20)

Several plots:

par(mfrow = c(2, 1))

plot (x, y, pch = 20) }
plot (x, z, pch = 19)



par(mfcol = c(2, 2))

↑
order of appearance

Adding to a plot

x <- rnorm(100)

y <- x + rnorm(100)

g <- gl(2, 50) <- groups $\left(\begin{smallmatrix} 50 \times 1 \\ 50 \times 2 \end{smallmatrix}\right)$

↑
label = c("Male", "Female")

```
plot(x, y, type = "n")
```

↑ don't put the data!

```
points (x[g == "Male"],
        y [g = "Male"], col = "green")

points ( x [g == "Female"],
         y [g == "Feemale ], col = "blue")
```

Lattice

xyplot

bwplot stripplot

histogram

dotplot

Generally take a formula

$$\underbrace{y \sim x}_{\text{variables}} \mid \underbrace{f * g}_{\text{conditional variables}}$$

Lattice functions return an object

```
x ← rnorm(100)
y ← x + rnorm (100, sd = 0.5 )
f ← gl (2, 50, labels = c("Group 1",
                          "Group 2"))
```

```
xyplot ( y ~ x | f )
           ↑
```

y vs x conditioned on f

demo

```
data( environment )
xyplot(ozone ~ radiation | data =
                          environment)
```

temp.cut = equal.~~cut~~ count (vec, 4)

function

equal.count splits into given number of intervals

```
xyplot(ozone ~ radiation | temp.cut,
         data = e)
```

Mathematical Annotations

?plotmath

```
plot(0,0, main=expression(theta==0),
     ylab=expression(hat(gamma)==0),
     xlab=expression(sum(x[i]*y[i], i==1, n)))
```

$\bar{X} = 15$
(mean)

```
xlab= substitute(bar(X)==k,
        list(k=mean(x)))
```

↗ replaces ~~X~~ k in ~~bar~~ the expression
onto value

Programming Assignments

in

el %in b = true if el in b

```
c(1,2,4,5) %in% 2:4
        ↓
    F, T, T, F
```

Sorting

sort(read) - returns a new sorted vector

order(read) - sorts and return the order (ids)

read[order(read)]

||

sort(read)

↖ returns the correct order of indices

order(read, prog)

↑

on two variables

order(prog, -read)

↑

reverse order

order(prog, na.last=F)

↑

NAs are last or first

order(prog, na.last=NA)

↑

don't include NAs

Ordering by medians

death - values
states - a factor with states

need: to order by median

medians = tapply(death, states, median)
↗
results with medians

order.by.medians = order(medians)

↖ sorted by medians order

levels(states)
returns a list of all factors
⇓
# levels(states)[order.by.medians]
# returns a sorted list of states ordered by median

ordered.states = levels(state) [order.by.medians]

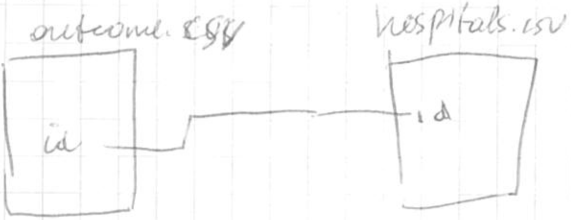
states.by.medians = ordered(states, ordered.states)

↓
builds an ordered factor
based on factor states
and the given order

⇓
so it can be used as:

boxplot(death ~ states.by.median)

Merging data



```
merge(outcome, hospitals, by = "id")
```

Errors

```
a = function() {
    if(...) {
        stop("reason")
    }
}
```

R: working under proxy

```
Sys.setenv(http_proxy = "http://username:
        password@proxy:8080")
```