2004-09-27

STATS 3N03/3J04 6-1

SAVING YOUR WORK

save.image( )

SAVES CURRENT WORKSPACE

DEFAULT FILE NAME* .RData

USE ls( ) TO LIST OBJECTS IN WORKSPACE.

savehistory( )

SAVES COMMANDS UP TO THIS ONE.

DEFAULT FILE NAME* .Rhistory

USE ↑ AND ↓ TO SCROLL THROUG EARLIER COMMANDS,

USE history( ) TO DISPLAY COMMAND LIST.

* IF YOU USE DEFAULT NAMES, FILES WILL LOAD AUTOMATICALLY AT LAUNCH, OTHERWISE LOAD FROM MENU.

When a variable can be numerical or categorical

```
> bricks$temp
 [1] 100 100 100 100 100 100 100 125 125 125 125 150 150 150 150 150 175 175
[19] 175 175 175 175

> as.factor(bricks$temp)
 [1] 100 100 100 100 100 100 100 125 125 125 125 150 150 150 150 150 175 175
[19] 175 175 175 175
Levels: 100 125 150 175

> as.numeric(as.factor(bricks$temp))
 [1] 1 1 1 1 1 1 1 2 2 2 2 3 3 3 3 3 4 4 4 4 4 4

> as.character(as.factor(bricks$temp))
 [1] "100" "100" "100" "100" "100" "100" "100" "125" "125" "125" "125" "150"
[13] "150" "150" "150" "150" "175" "175" "175" "175" "175" "175"

> as.numeric(as.character(as.factor(bricks$temp)))
 [1] 100 100 100 100 100 100 100 125 125 125 125 150 150 150 150 150 175 175
[19] 175 175 175 175
```

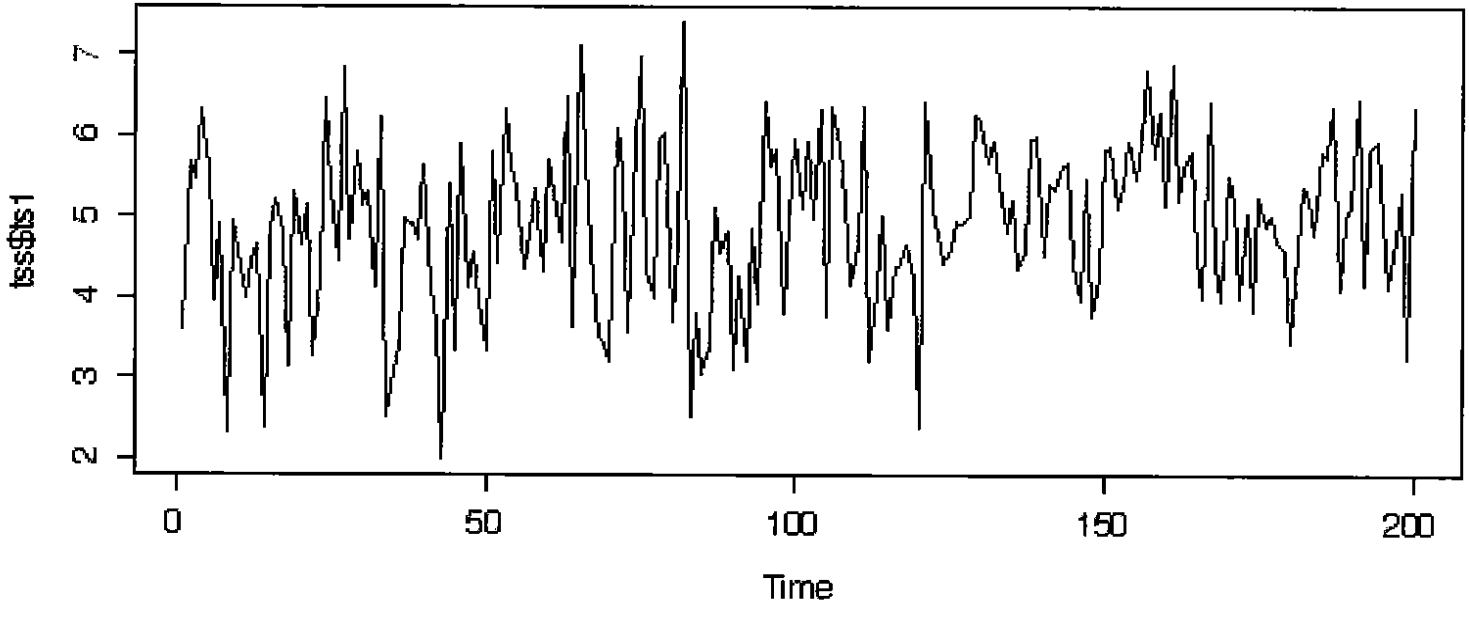
To add a new column to the data frame

```
> bricks$tempf <- as.factor(bricks$temp)
> bricks$tempf
 [1] 100 100 100 100 100 100 100 125 125 125 125 150 150 150 150 150 175 175
[19] 175 175 175 175
Levels: 100 125 150 175
```

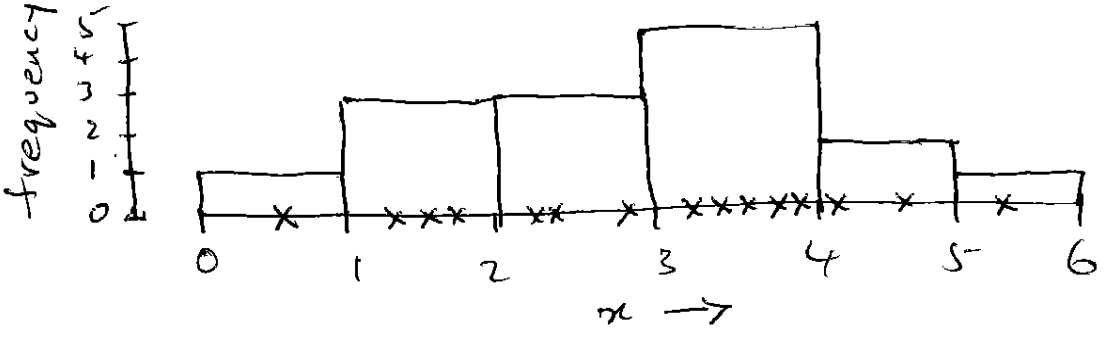
Time series plots

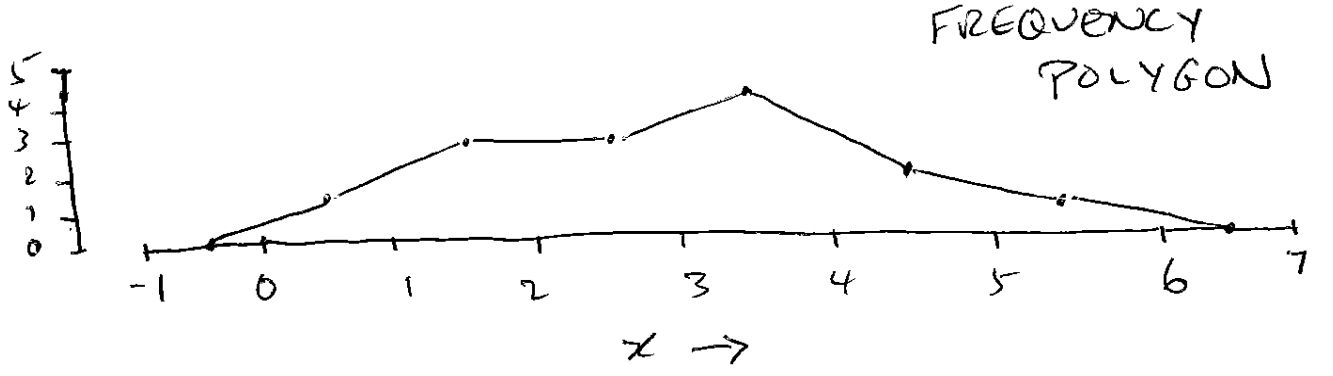
A time series object has values and times (or dates); ts.plot ()or plot.ts() can display the times (or dates) on the X-axis. Because tss$ts1 is a simple vector, these three commands will give identical plots:

```
> tss<-data.frame(ts1 = 5+rnorm(200))
> plot(tss$ts1, type="l", xlab="Time")
> plot.ts(tss$ts1)
> ts.plot(tss$ts1)
```



HISTOGRAMS



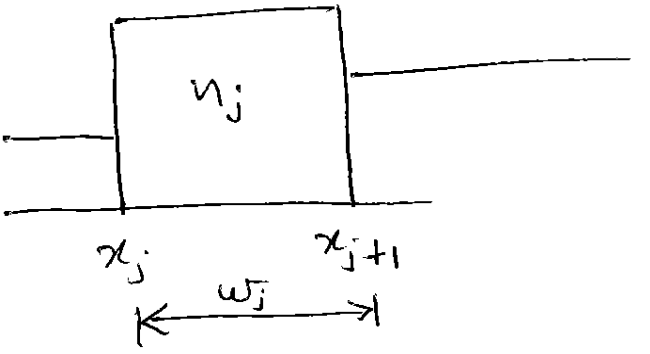


- DATA ARE GROUPED INTO BINS OR CELLS.

- GRAPH SHOWS WHERE THE POINTS ARE MOST DENSE ALONG THE X-AXIS.

- Y-AXIS CAN BE FREQUENCY OR DENSITY

- MUST USE DENSITY WHEN THE BINS ARE UNEQUAL WIDTH.



n = TOTAL SAMPLE SIZE

n_j = FREQUENCY (COUNT) IN BIN j

w_j = WIDTH OF BIN j

$= x_{j+1} - x_j$

$$f_j = \frac{n_j}{n\, w_j}$$ = PROBABILITY DENSITY IN BIN j

hist (x, breaks = , right = T, include.lowest = T, prob = T, ...)

x VECTOR OF DATA

breaks VECTOR OF CUT POINTS (BREAK POINTS) OR THE SUGGESTED NUMBER OF BINS.

NOTE: IF $n > 100$ AND DISTRIBUTION HAS A SIMPLE SHAPE, 25 BINS IS ENOUGH.

CONTROL OF DATA ON THE CUT POINTS:

right = T ( ]
F [ )

include.lowest = T

⇒ FIRST INTERVAL [ ]
IF right = T

OR

LAST INTERVAL [ ]
IF right = F

FREQUENCY OR DENSITY?

prob = F WILL SHOW FREQUENCIES IF BINS ARE OF EQUAL WIDTH.

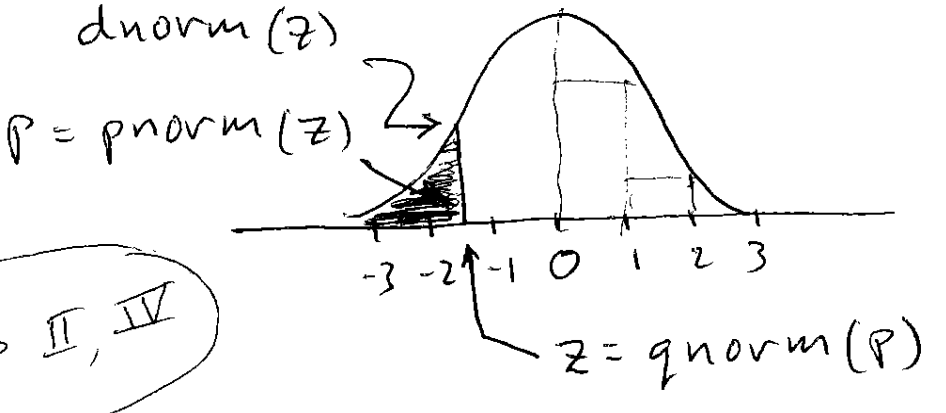
STANDARD NORMAL PROBABILITY DENSITY FUNCTION

$$f(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2}$$

$$-\infty < z < \infty$$

R FUNCTIONS

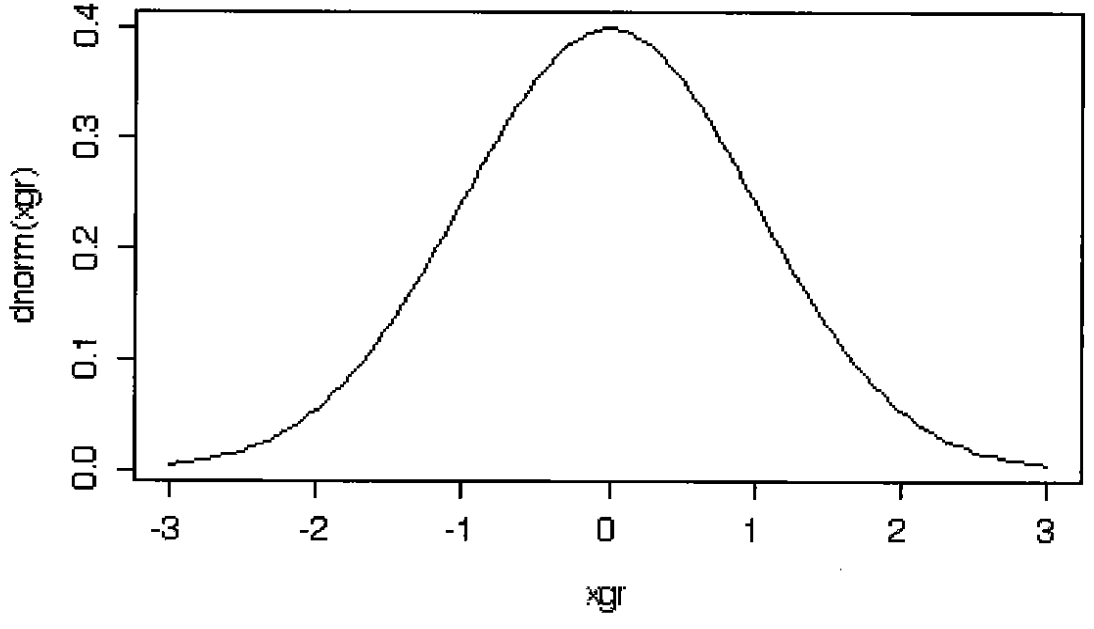
dnorm	DENSITY
pnorm	PROBABILITY INTEGRAL
qnorm	QUANTILE
rnorm	GENERATE RANDOM DATA



TABLES II, IV

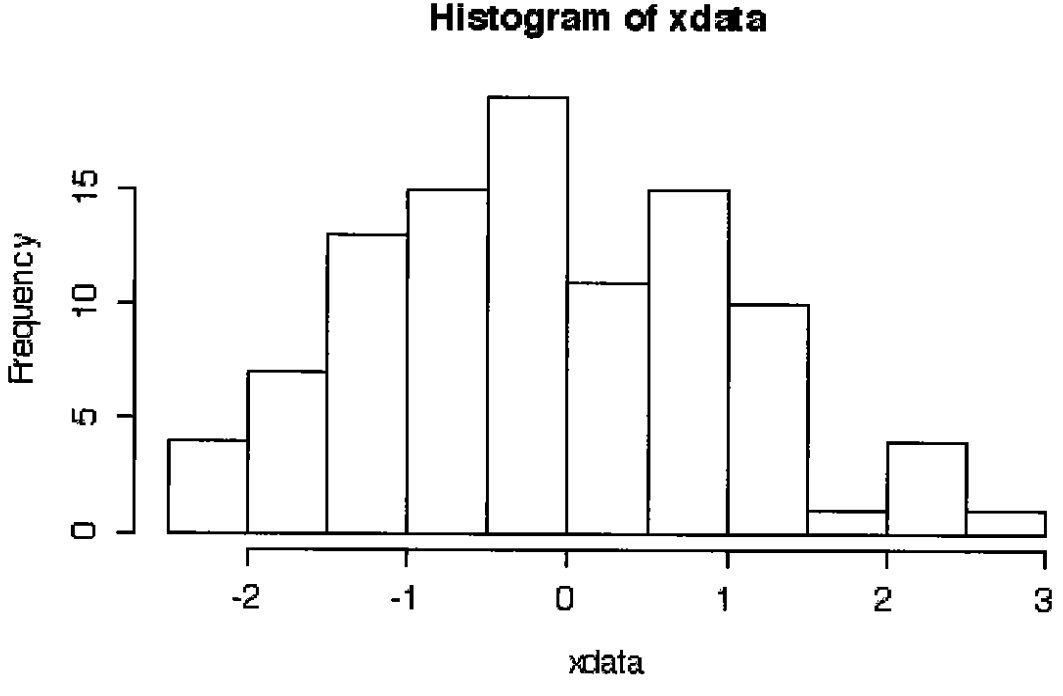
```
> xgr <- seq(-3,3,length=100)
> plot(xgr, dnorm(xgr), type="l")
```

PICK A CONVENIENT GRID, WITH ENOUGH POINTS TO GIVE A SMOOTH GRAPH.



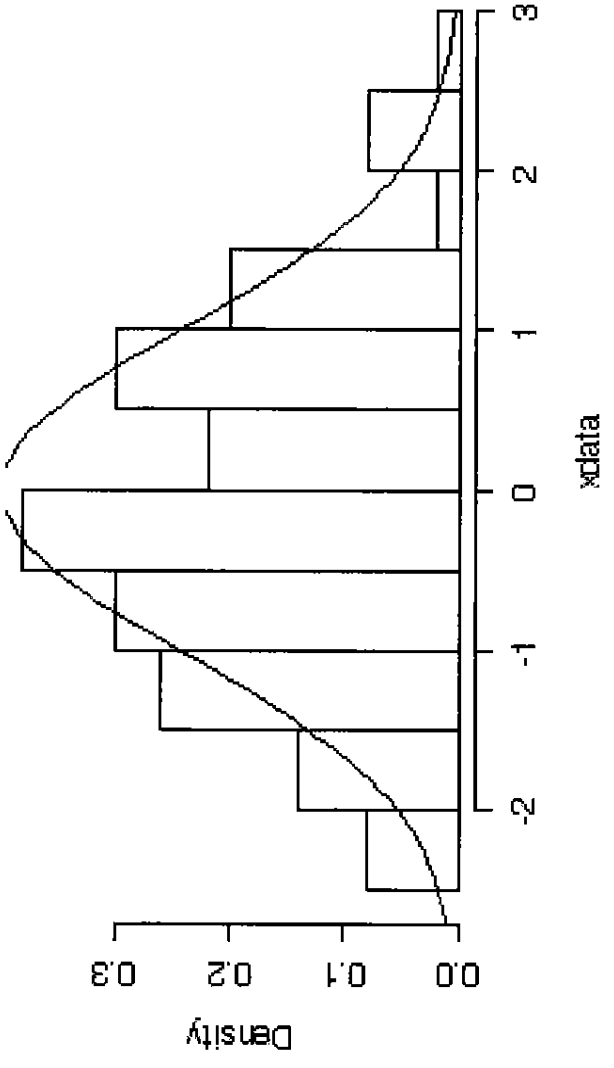

```
> xdata <- rnorm(100)
> xdata
  [1] -0.75203043 -0.79045236 -0.20772653  2.03461456 -1.30998712 -0.88450615
  [7]  0.21034194 -1.72374767 -0.06451537 -1.32353644  0.91994418 -0.64101966
 [13] -0.05880250  0.60361971 -1.11570316  1.14319485 -1.00975503  0.07268766
 [19] -1.20338179 -0.39337011  0.86866265 -1.13767155  0.76775870  0.52060887
 [25] -1.66377305 -0.91080300 -2.05919628 -1.53208173 -1.73094883 -2.06410713
 [31]  0.39114088  2.42131279  0.74106121 -1.37310054 -0.87437870 -2.16465679
 [37]  0.33589807 -0.03488092 -0.08532791  1.36557262  1.35164198 -1.31995704
 [43] -0.73572075 -0.47462189 -0.66989003 -0.58195175 -0.44863549  0.74301470
 [49] -0.35803123  1.33729931  2.07277832  0.43954190 -0.07222533  0.87171936
 [55] -0.34731896 -1.83057364  0.71035244 -1.37556595  0.01255697  2.11044307
 [61]  0.03587672 -1.76838407 -0.24261444 -0.72253323  1.73716406 -0.77580525
 [67]  0.63221643 -2.31356267  1.27878553 -0.05873865 -1.07931220  1.22013957
```

```
 [73] -0.31696936 -0.88024621  0.50927732 -1.48609038 -0.43143449 -0.77302837
 [79] -0.33892352 -1.04653953  1.40629958  0.60244898  0.26198451  2.98266305
 [85]  1.16303448 -1.18025171  0.33900330  0.16614656  0.87368904 -0.25203893
 [91]  1.14653402  1.00850091 -0.04049534 -0.35601063 -0.62632898 -0.87675328
 [97]  0.70897181  0.90342715 -1.53925529  0.18322698
> mean(xdata)
[1] -0.1322411
> var(xdata)
[1] 1.236172
> sqrt(var(xdata))
[1] 1.111833
> hist(xdata)
```



6-10

```
> hist(xdata, prob=T, col="yellow")
> lines(xgr, dnorm(xgr), col="red")
```



```
> hist(xdata, prob=T, col="yellow", ylim=c(0,.4))
> lines(xgr, dnorm(xgr), col="red")
```