# Graphics in R
## The Big Picture

Gordon Erlebacher

# Where are we?

- Vectors
- Vector element extraction
    - logical expressions
- Data Frames and data extraction
    - rows versus columns
- factors
- Functions
- Logic

# Where are we going?

- How to visualize data
    - easier to understand

- Next two lesssons

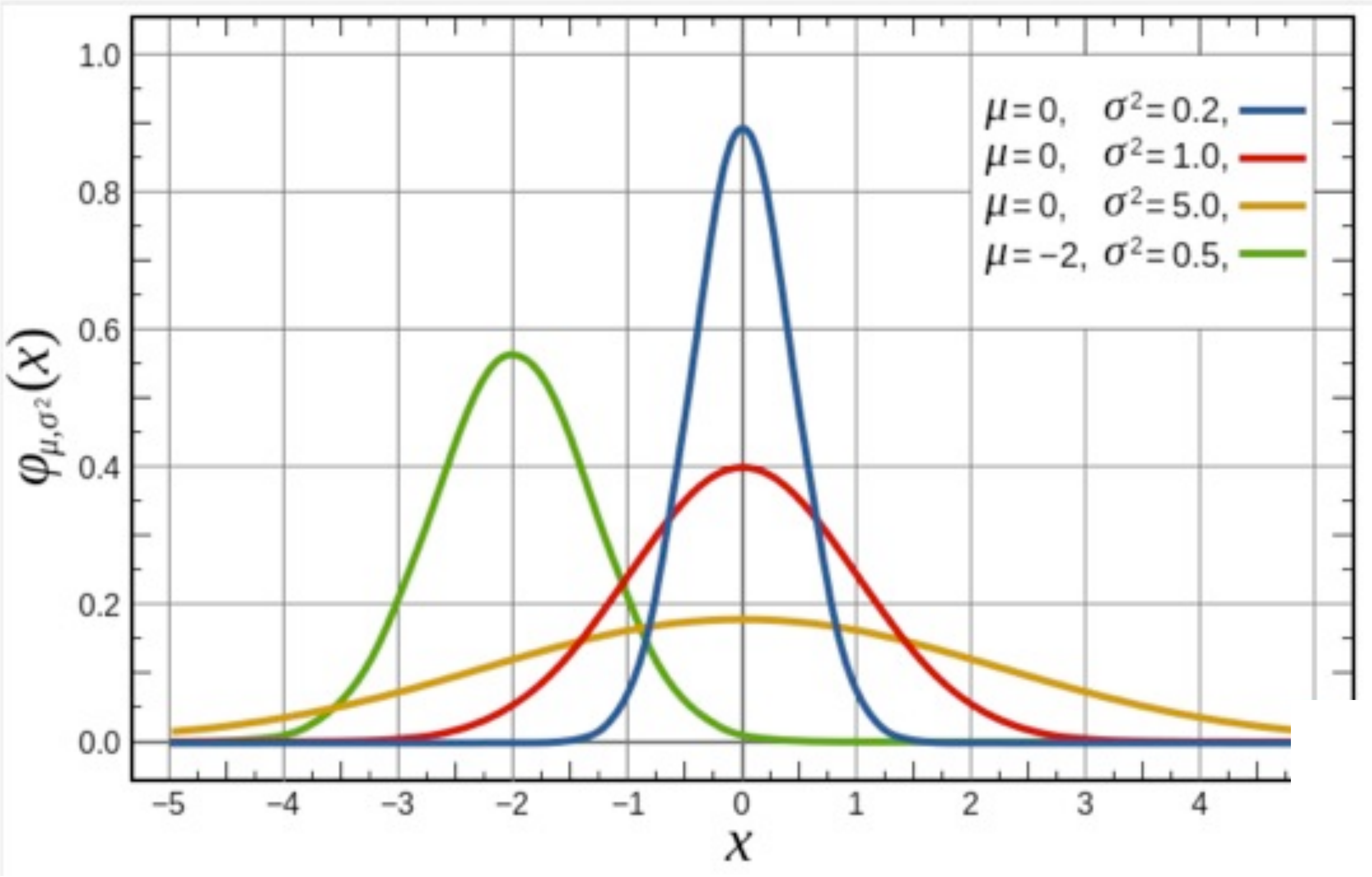
# Functions used

- vector, data.frame, seq, mean, var, sd
- factor, source
- class, str, summary
- : , ? , < , > , <=, ==, +, -, *, etc.

# New functions

- hist()
- plot()
- par()
- print(), cat() : printing data to the screen

# The Normal Distribution

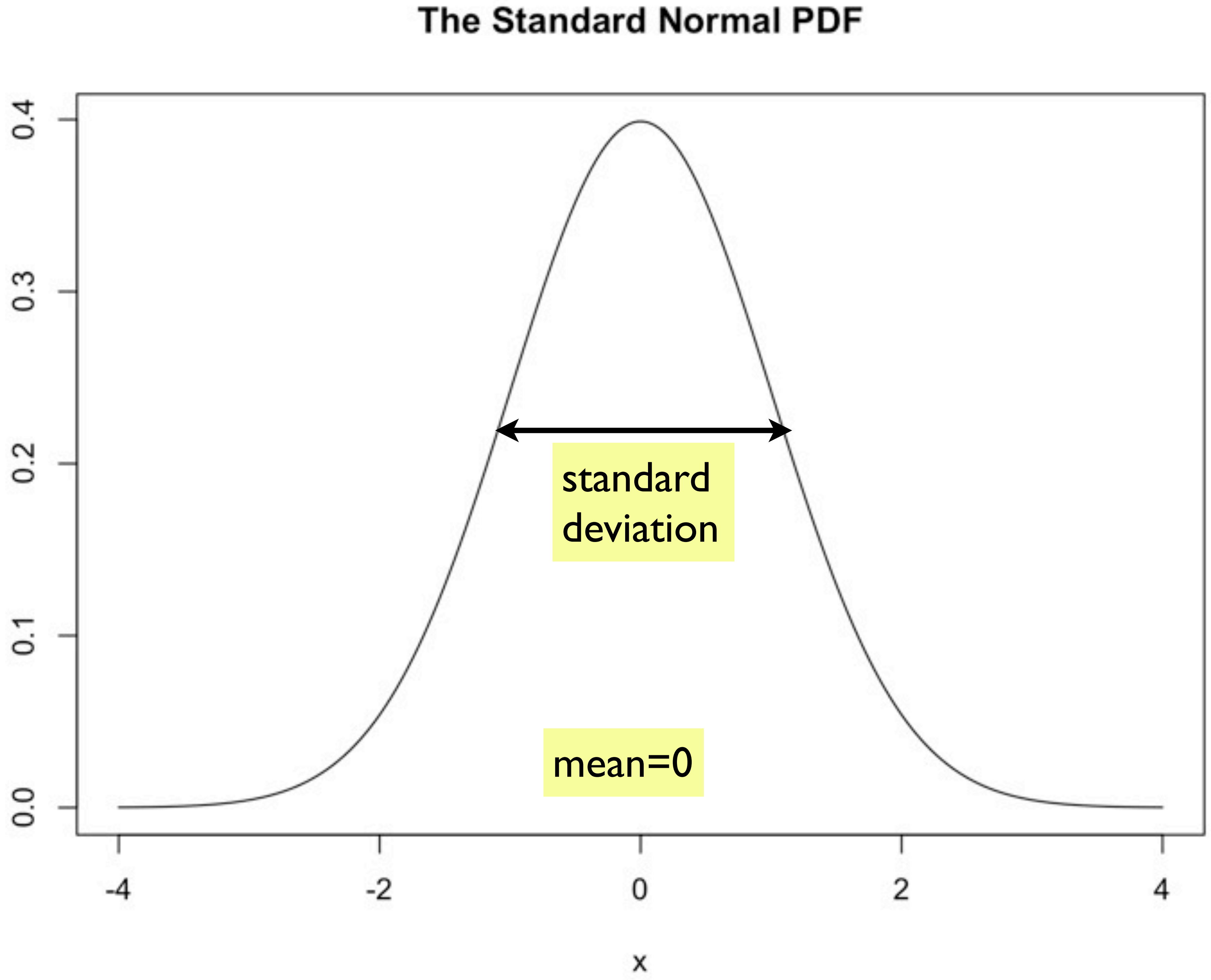


WIKIPEDIA

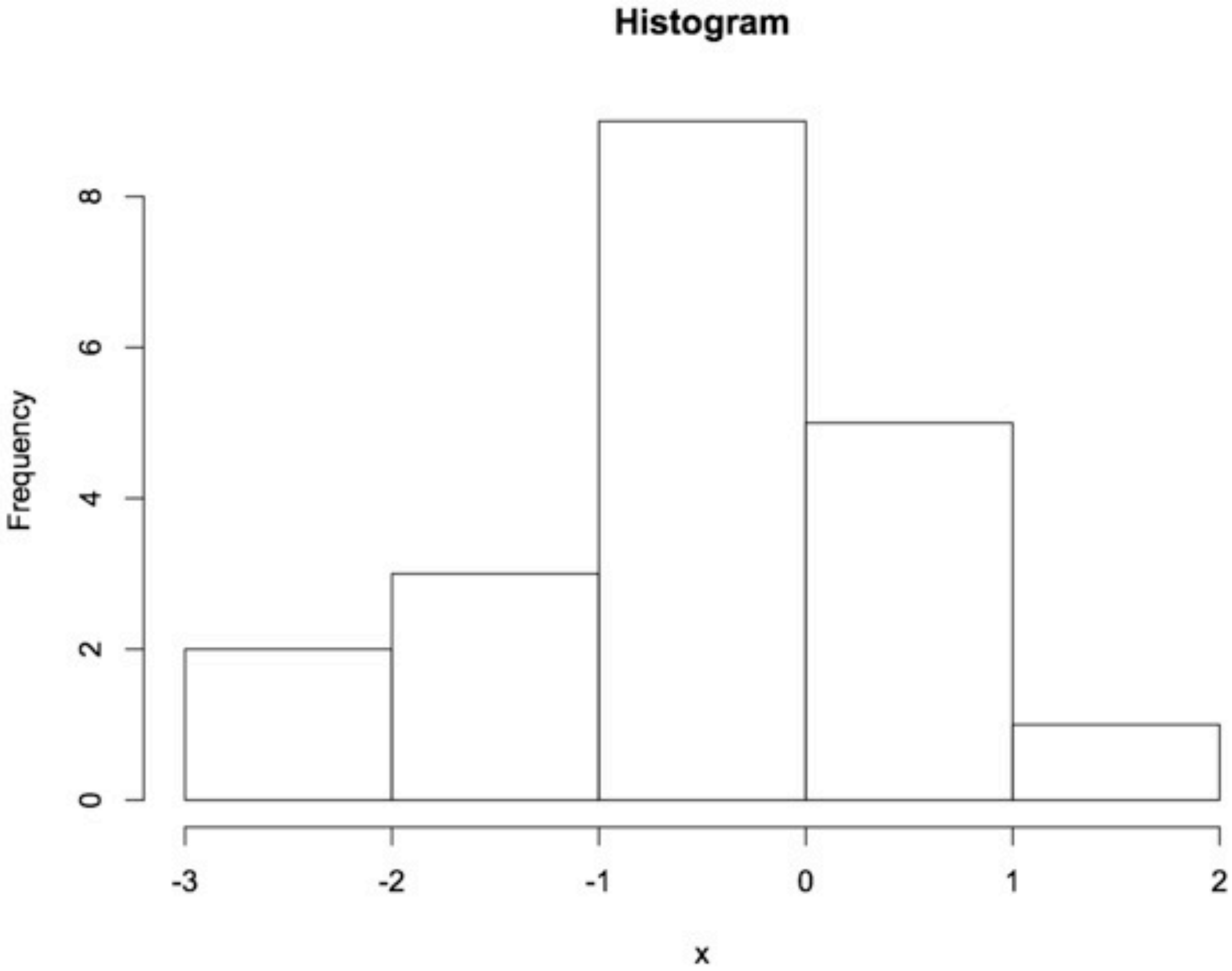
# Central Limit Theorem in probability theory

**Central limit theorem** - sum of many independent and identically distributed (i.i.d.) random variables, will tend to be distributed according to one of a small set of probability distributions. When the variance and mean of the i.i.d. variables are finite, the distribution of the sum is the normal distribution.

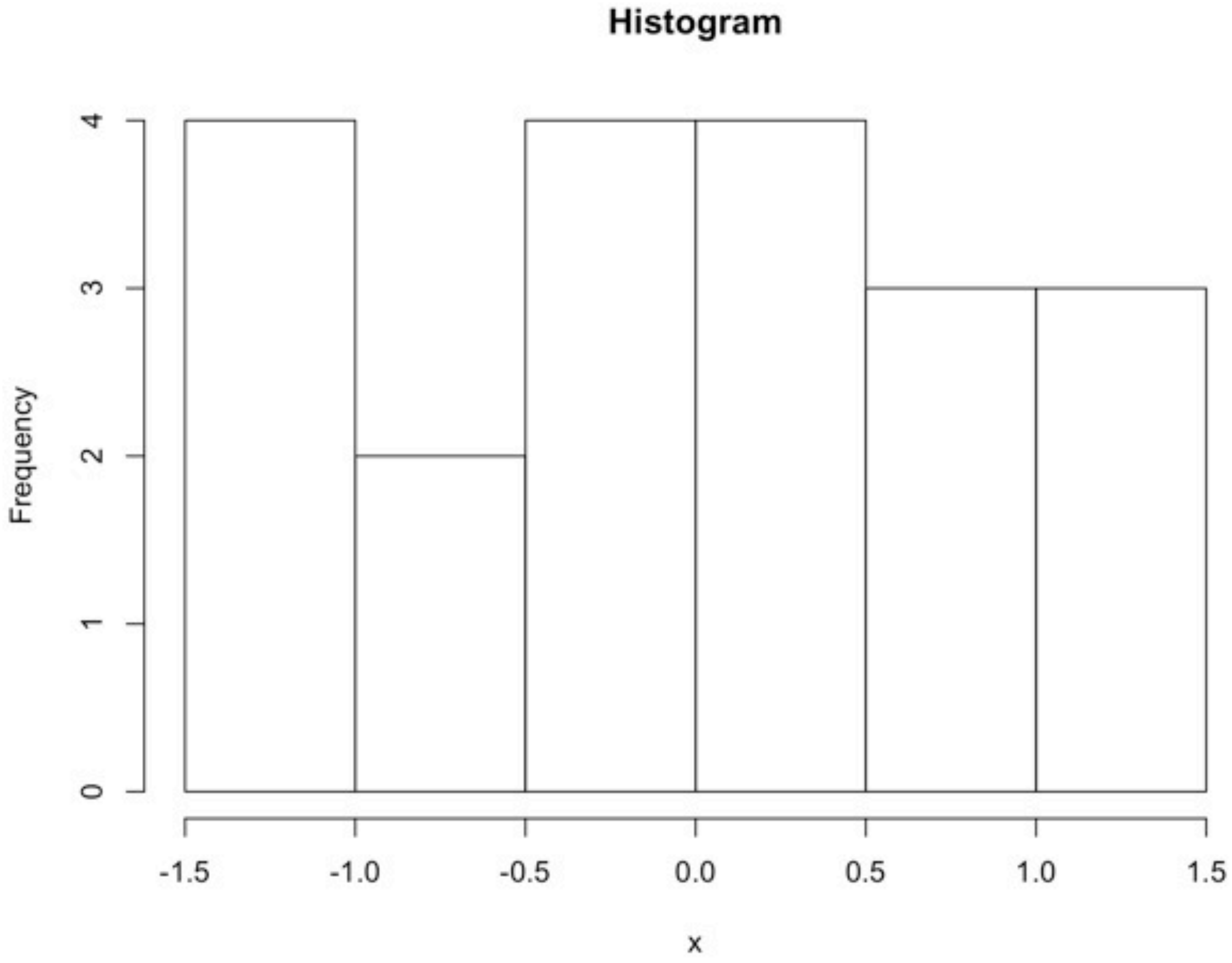
WIKIPEDIA


The Standard Normal PDF
0.4
0.3
0.2
0.1
0.0
-4
-2
0
2
4
x
standard deviation
mean=0

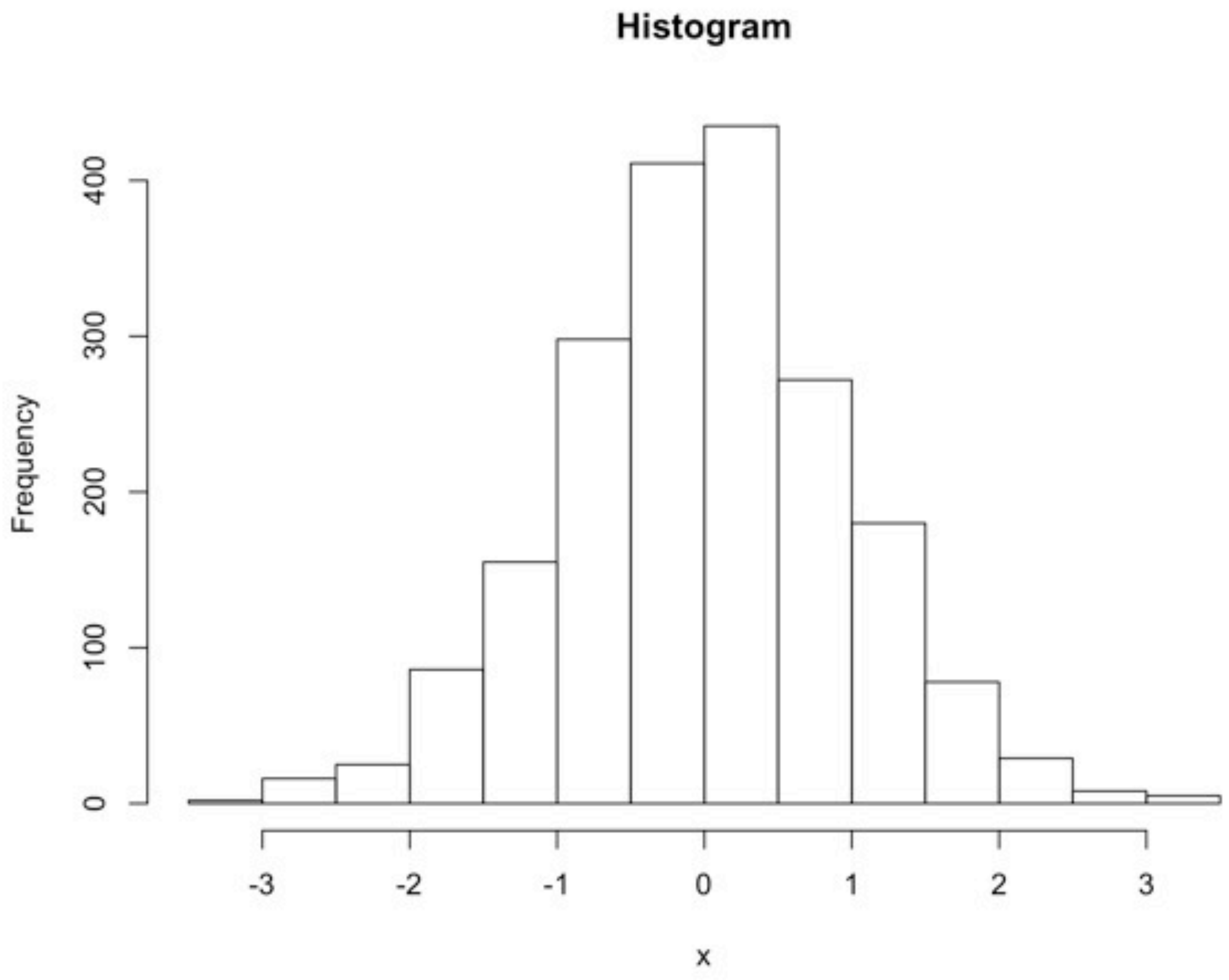
# Small Samples (n=20)



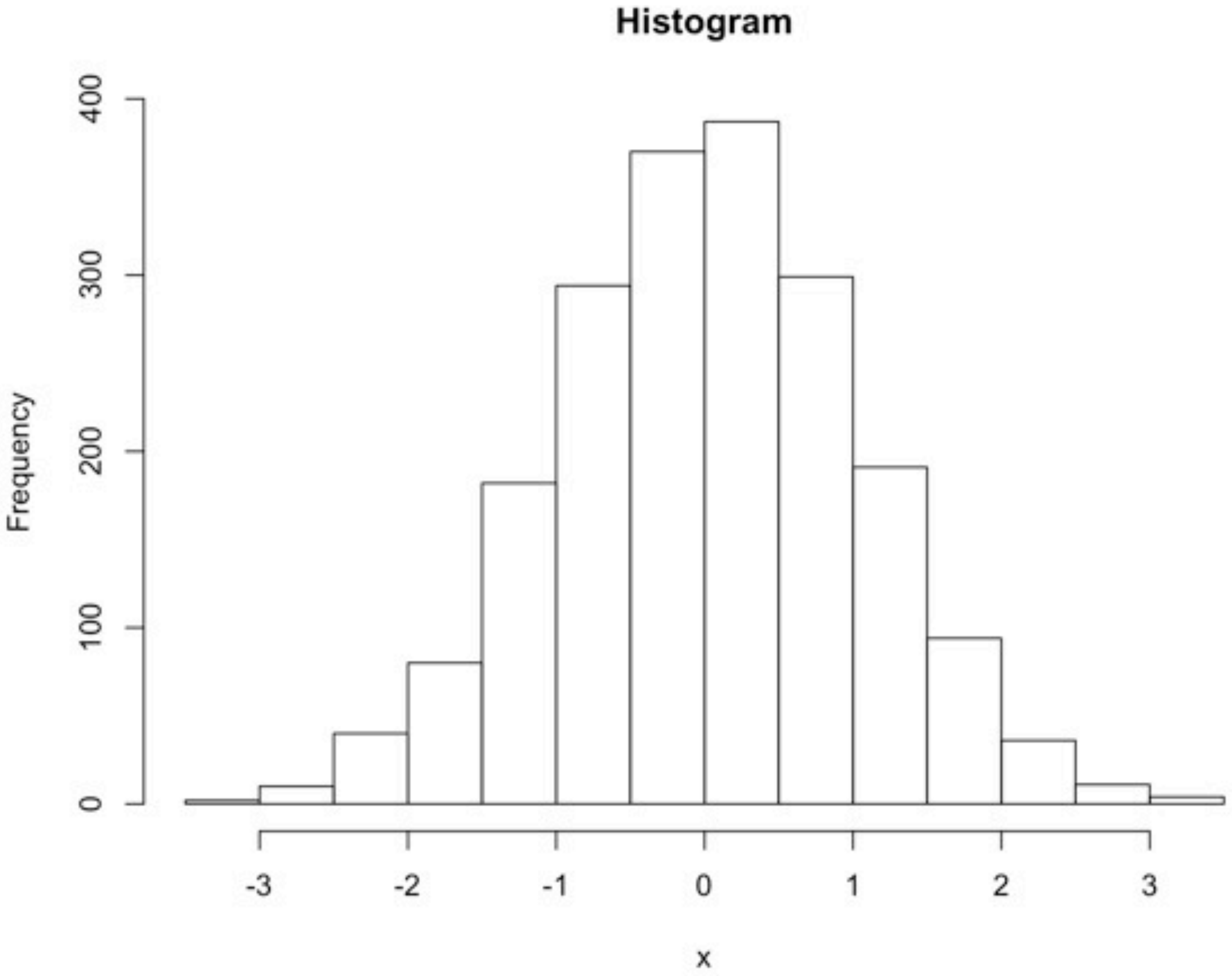
# Small Samples (n=20)



# Larger Samples



# Larger Samples



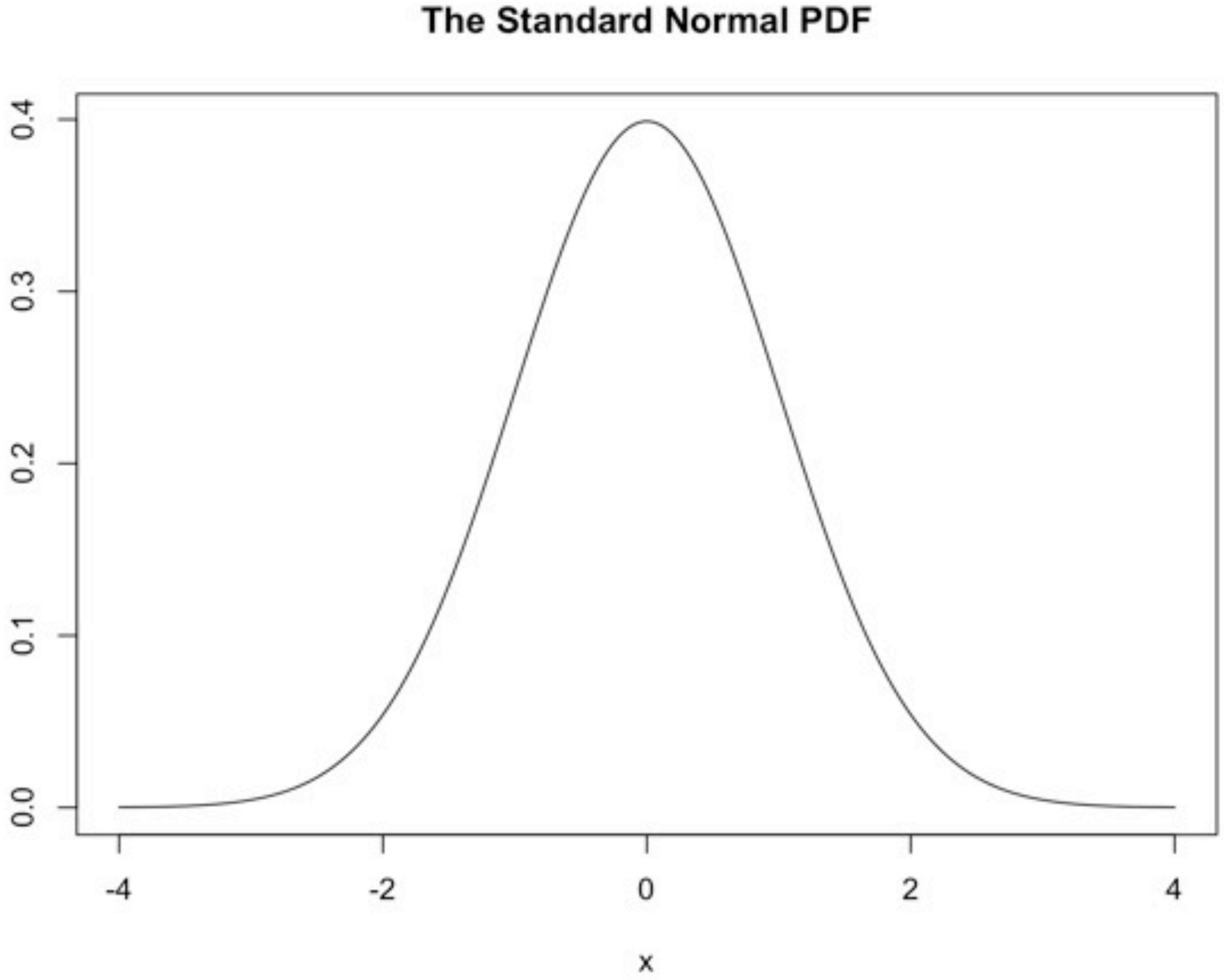
# How did I generate the data and how did I plot it?

1. Generate the data to plot

v = rnorm(n) generate “n” samples from a normal distribution and stores it in a vector named “v”

2. plot the results using commands (functions) such as plot() and hist()

# rnorm(n)



# hist()

hist {graphics}

## Histograms

## Description

The generic function `hist` computes a histogram of the given data values. If `plot=TRUE`, the resulting object of class `"histogram"` is plotted by plot.histogram, before it is returned.

## Usage

```
hist(x, ...)
```

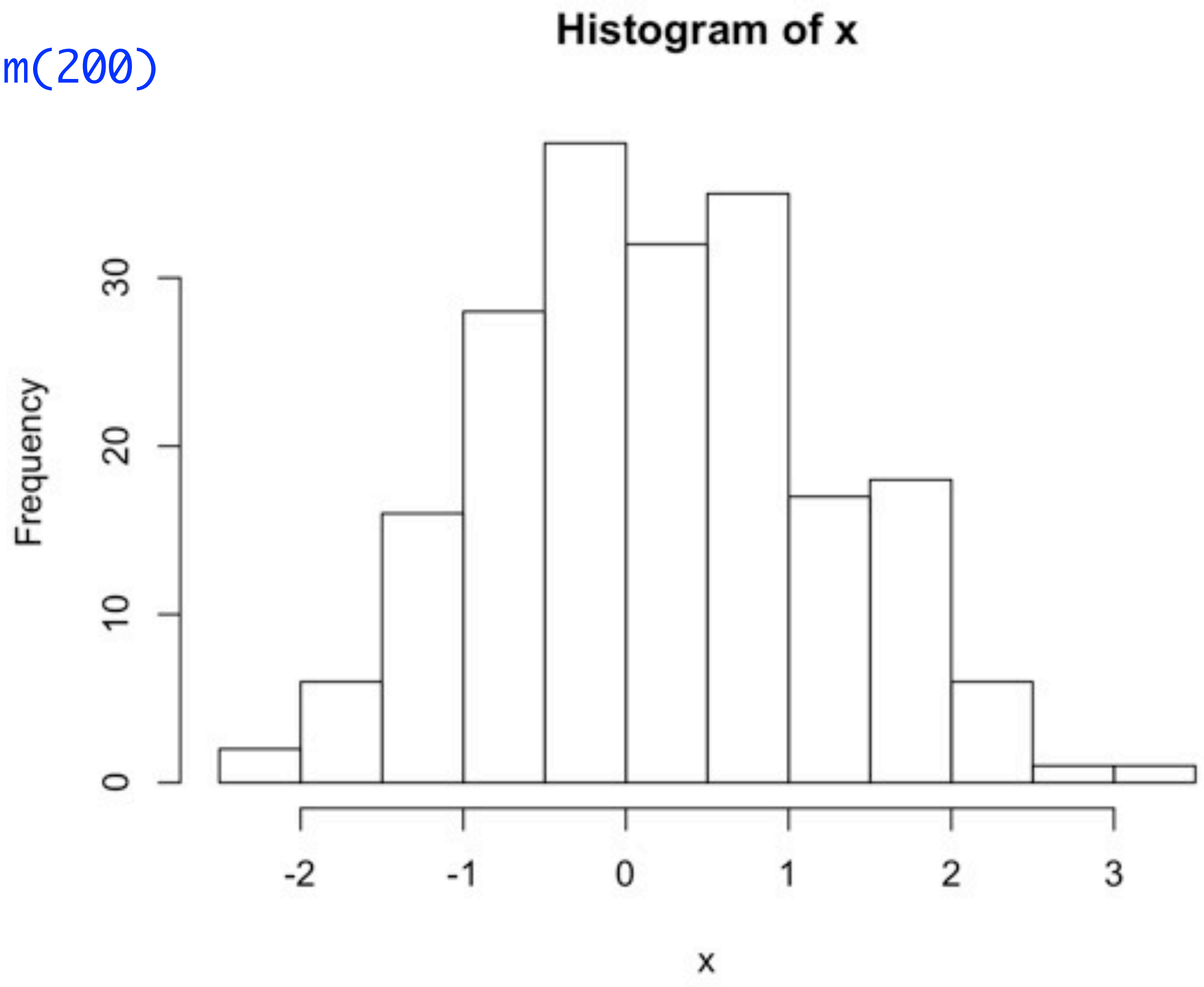
# Large vs. Small

Try it...

```
> hist(rnorm(20))
...

> hist(rnorm(200))
...

> hist(rnorm(2000))
...
```

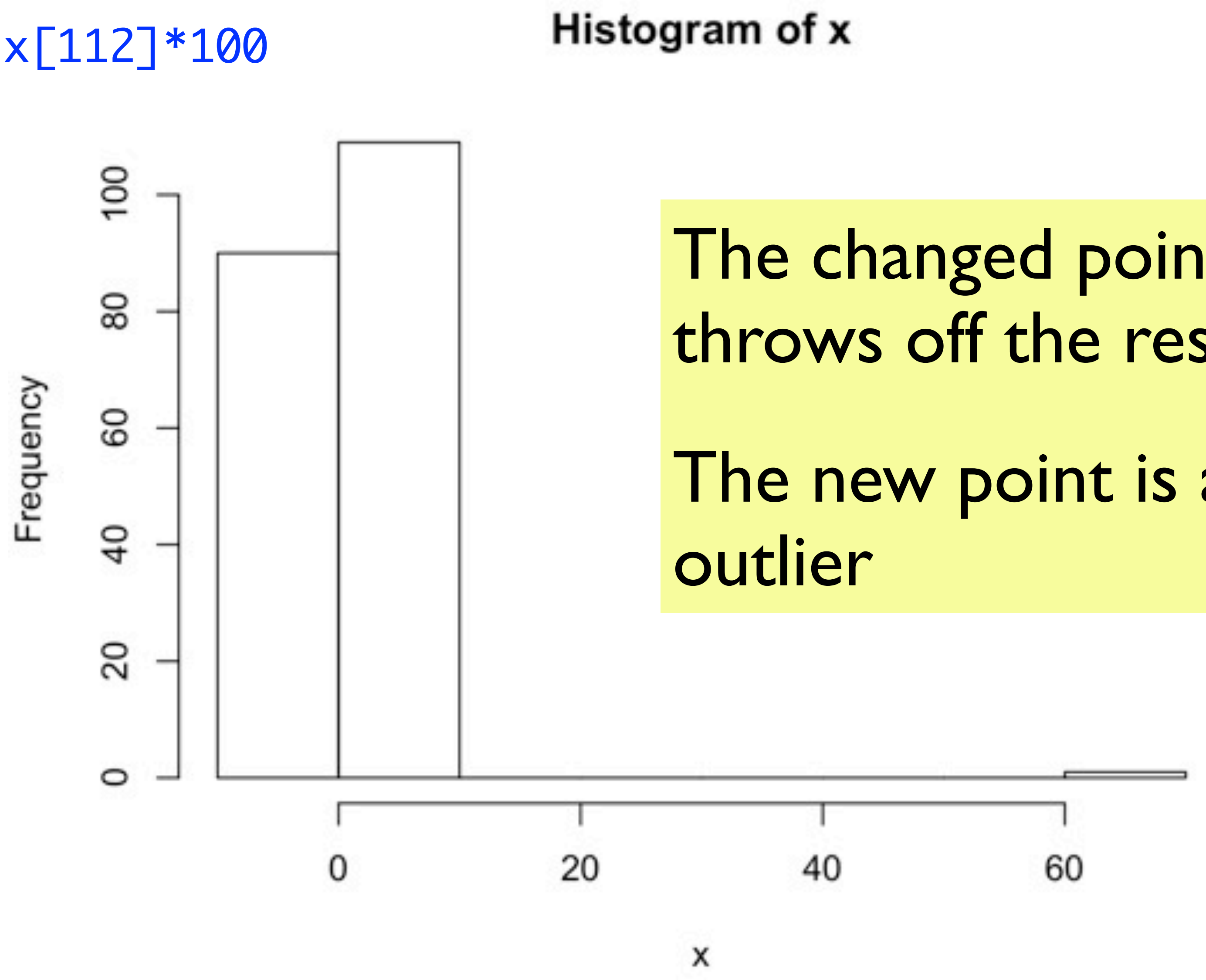
# Data Checking

```
> x = rnorm(200)
> hist(x)
```



# Data Checking

```
> x[112] = x[112]*100
> hist(x)
```



The changed point throws off the results!

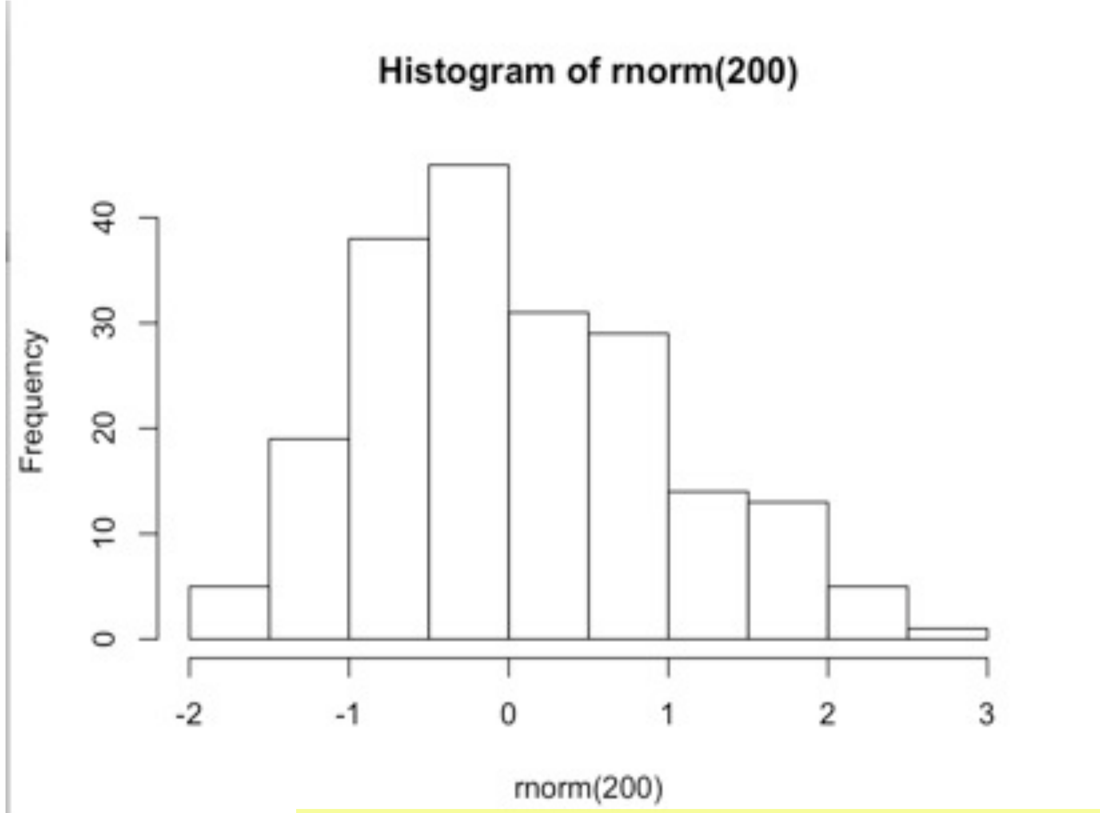
The new point is an outlier

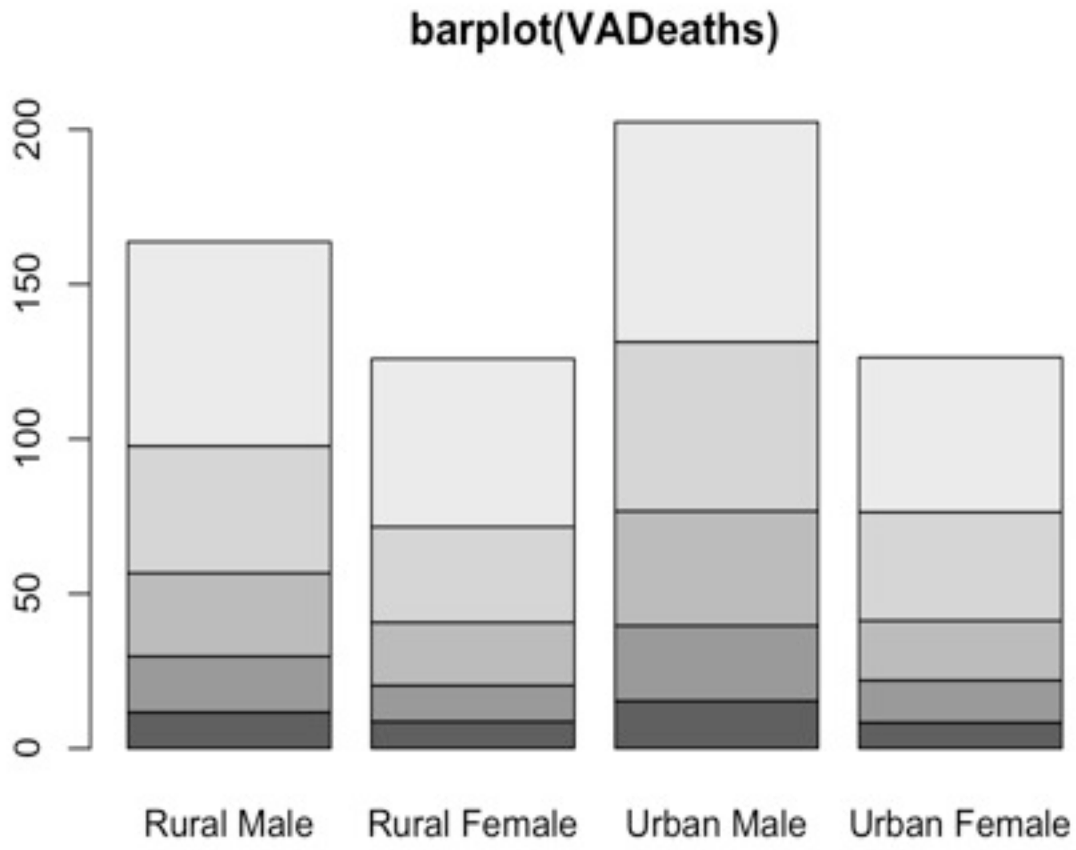
# hist() vs. barplot()

**histogram** - a graphical representation showing a visual impression of the distribution of data. It is an estimate of the probability distribution of a continuous variable.

**bar chart/graph** (barplot in R) - a chart with rectangular bars with lengths proportional to the values that they represent. ... Bar charts provide a visual presentation of categorical data.

# hist() vs. barplot()





shading indicates age groups

```
> head(VADeaths)
      Rural Male Rural Female Urban Male Urban Female
50-54       11.7          8.7       15.4          8.4
55-59       18.1         11.7       24.3         13.6
60-64       26.9         20.3       37.0         19.3
65-69       41.0         30.9       54.6         35.1
70-74       66.0         54.3       71.1         50.0
```

# class of VADeaths

```
> data(VADeaths)
> class(VADeaths)

matrix
```

barplot(VADeaths)  works
barplot(as.data.frame(VADeaths)) gives an **error**

# ?barplot

barplot(height, width = 1, space = NULL,  ..........

height: either a **vector or matrix** of values describing the bars that make up the plot.

Thus, if height is a data frame, barplot will not work properly.

For now, a matrix is a special version of a data frame where all columns are of the same type!

# ?hist

```
hist(x, ...)

## Default S3 method:
hist(x, breaks = "Sturges",
     freq = NULL, probability = !freq,
     include.lowest = TRUE, right = TRUE,
     density = NULL, angle = 45, col = NULL, border = NULL,
     main = paste("Histogram of" , xname),
     xlim = range(breaks), ylim = NULL,
     xlab = xname, ylab,
     axes = TRUE, plot = TRUE, labels = FALSE,
     nclass = NULL, warn.unused = TRUE, ...)
```

**breaks**: one of:

- a vector giving the breakpoints between histogram cells,
- a single number giving the number of cells for the histogram

... there are two other possibilities ...

# Arguments to hist()

Title
Subtitle
Axis Labels
Axis Limits
Frequencies vs. Proportions
Colors

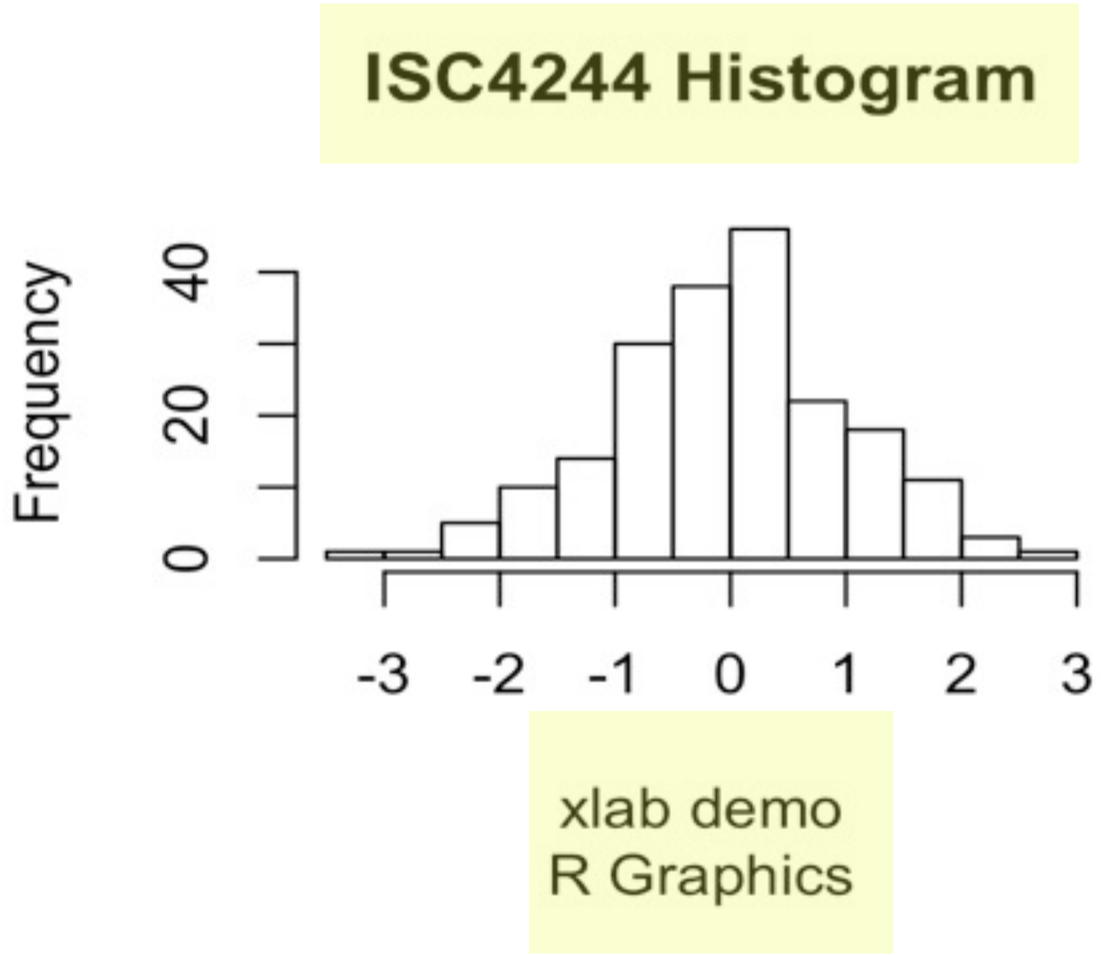
# Title & Subtitle

main=
sub=

```
> hist(rnorm(200))
> hist(rnorm(200), main="ISC4244 Histogram")
> hist(rnorm(200), main="ISC4244 Histogram", sub="R Graphics")
```

# Axis Labels

```
> hist(rnorm(200), main="ISC4244 Histogram", sub="R Graphics", xlab="xlab demo")
```



# Typeface vs. Font

In typography, a **typeface** is a set of characters that share common design features.

A **font** is traditionally defined as a quantity of sorts composing a complete character set of a single size and style of a particular typeface.

Here, the typeface is Arial, the fonts are Arial 32 pt, Arial 32pt bold, Arial 44pt.

What is a “point”?

WIKIPEDIA

# Typeface vs Font

## family

The name of a font family for drawing text. ... The default value is `""` which means that the default device fonts will be used (and what those are should be listed on the help page for the device). Standard values are `"serif"`, `"sans"` and `"mono"`...

## font

An integer that specifies which font to use for text. If possible, device drivers arrange so that 1 corresponds to plain text (the default), 2 to bold face, 3 to italic and 4 to bold italic. ...

to get above information: **?par** and scroll down

hist() arguments: font, font.axis, font.lab, font.main, font.sub

# Typeface vs Font

font: affects axis numbers
font.main : main title
font.axis : axis numbers (same as font)
font.lab: labels
font.sub: subtitle of x label

```
> hist(rnorm(200), main="ISC4244 Histogram",family="mono")
> hist(rnorm(200), main="ISC4244 Histogram",family="mono",font=2)
> hist(rnorm(200), main="ISC4244 Histogram",family="mono",font=3)
```

typing the same long list of arguments over and over again gets tedious. Create your own function (advanced)

# User function

Create a file with the name “myfunctions.r”
and store the following

```
histo = function(x, title="")
{
        hist(x, main=title, family='mono', font=3)
}
```

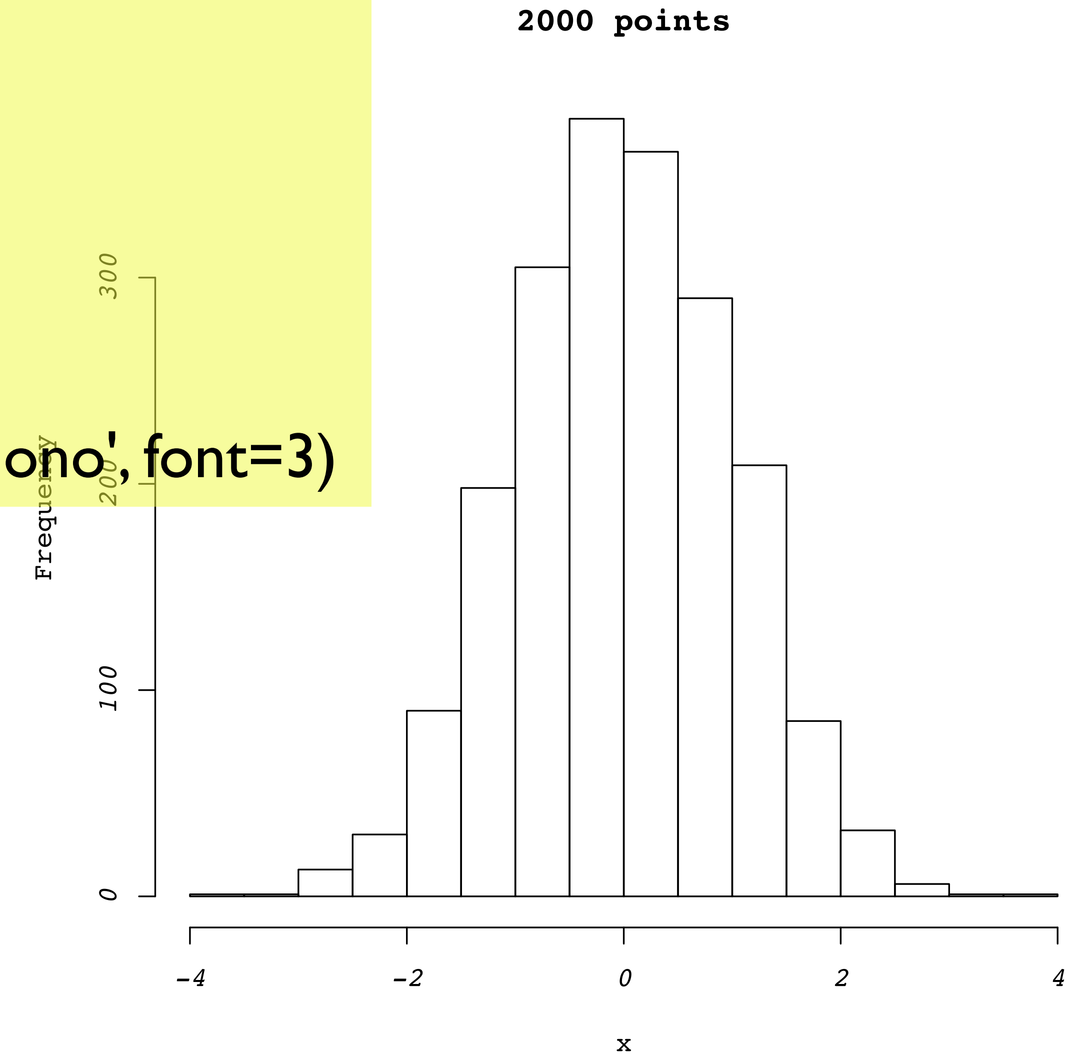
The file “myfunctions.r” is a script that contains a function, which is used like any other function

# Try it!

```
> histo(rnorm(2000), title="2000 points")  # my own function
```

equivalent to:

```
hist(rnorm(2000), main=”2000 points”, family='mono', font=3)
```



# Axis Limits

xlim, ylim: vector with two entries:
minimum value and maximum value

xlim=c(begin, end)

```
> hist(rnorm(200), xlim=c(-5,5))
> hist(rnorm(200), xlim=c(5,-5))
> hist(rnorm(200), xlim=c(0,4))
```

# Colors

`col`
A specification for the default plotting color. See section ‘Color Specification’.

Some functions such as lines and text accept a vector of values which are recycled and may be interpreted slightly differently.

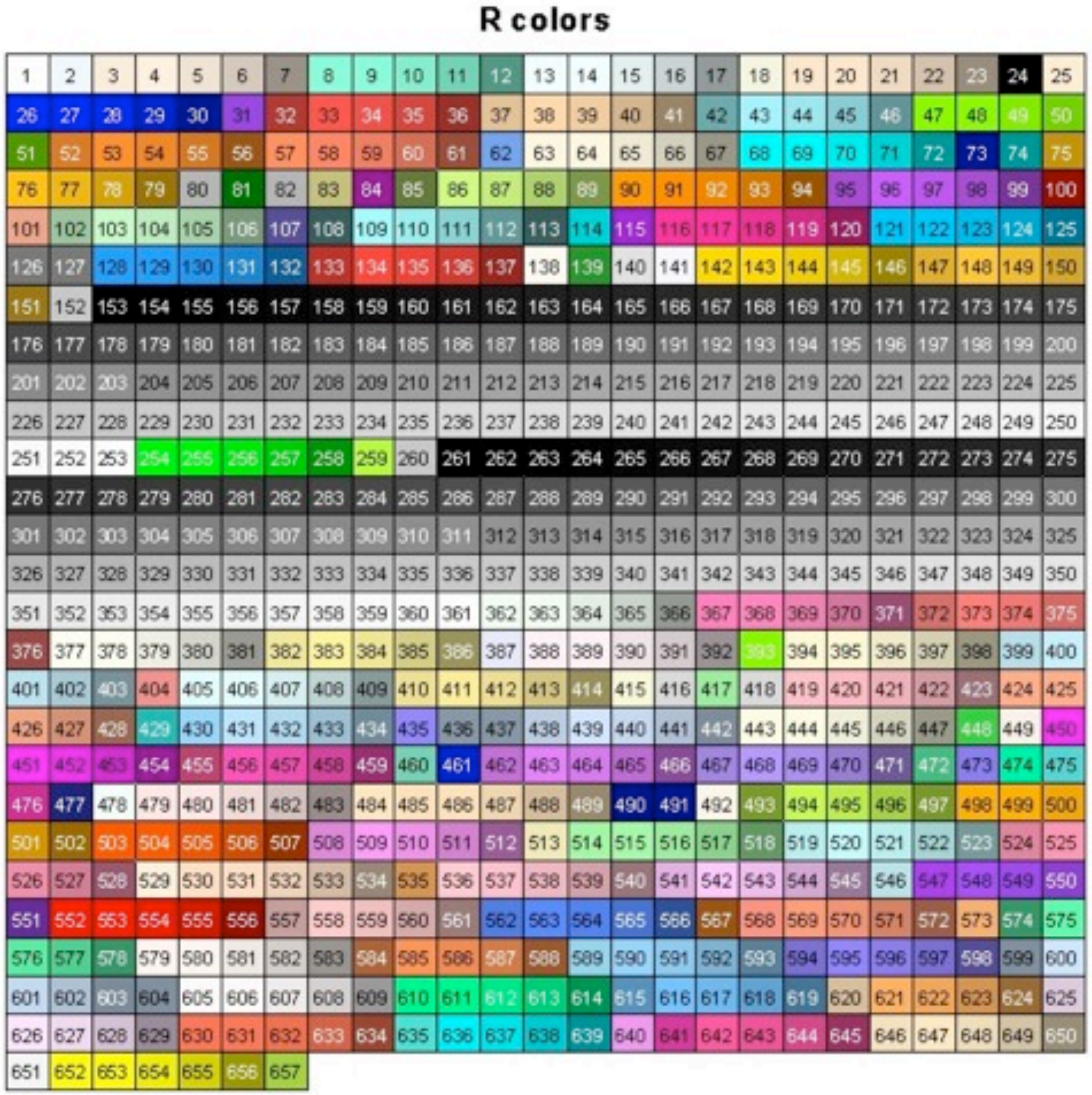
`col.axis, col.lab, col.main, col.sub`
Defaults to `"black"`.

E.g., “black”, “white”, “red”, “green”, “blue”

# Colors

```
> hist(rnorm(200), col="red")

> hist(rnorm(200), col="green")

> hist(rnorm(200), col="papayawhip")
```

http://research.stowers-institute.org/efg/R/Color/Chart/
http://research.stowers-institute.org/efg/R/Color/Chart/ColorChart.pdf



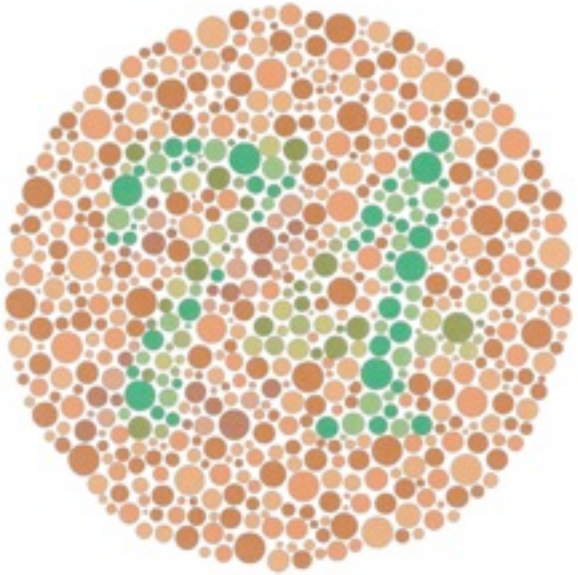
R colors

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 |
| 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 |
| 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 |
| 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 |
| 126 | 127 | 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 | 144 | 145 | 146 | 147 | 148 | 149 | 150 |
| 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 | 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 |
| 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 | 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 200 |
| 201 | 202 | 203 | 204 | 205 | 206 | 207 | 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | 217 | 218 | 219 | 220 | 221 | 222 | 223 | 224 | 225 |
| 226 | 227 | 228 | 229 | 230 | 231 | 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 | 240 | 241 | 242 | 243 | 244 | 245 | 246 | 247 | 248 | 249 | 250 |
| 251 | 252 | 253 | 254 | 255 | 256 | 257 | 258 | 259 | 260 | 261 | 262 | 263 | 264 | 265 | 266 | 267 | 268 | 269 | 270 | 271 | 272 | 273 | 274 | 275 |
| 276 | 277 | 278 | 279 | 280 | 281 | 282 | 283 | 284 | 285 | 286 | 287 | 288 | 289 | 290 | 291 | 292 | 293 | 294 | 295 | 296 | 297 | 298 | 299 | 300 |
| 301 | 302 | 303 | 304 | 305 | 306 | 307 | 308 | 309 | 310 | 311 | 312 | 313 | 314 | 315 | 316 | 317 | 318 | 319 | 320 | 321 | 322 | 323 | 324 | 325 |
| 326 | 327 | 328 | 329 | 330 | 331 | 332 | 333 | 334 | 335 | 336 | 337 | 338 | 339 | 340 | 341 | 342 | 343 | 344 | 345 | 346 | 347 | 348 | 349 | 350 |
| 351 | 352 | 353 | 354 | 355 | 356 | 357 | 358 | 359 | 360 | 361 | 362 | 363 | 364 | 365 | 366 | 367 | 368 | 369 | 370 | 371 | 372 | 373 | 374 | 375 |
| 376 | 377 | 378 | 379 | 380 | 381 | 382 | 383 | 384 | 385 | 386 | 387 | 388 | 389 | 390 | 391 | 392 | 393 | 394 | 395 | 396 | 397 | 398 | 399 | 400 |
| 401 | 402 | 403 | 404 | 405 | 406 | 407 | 408 | 409 | 410 | 411 | 412 | 413 | 414 | 415 | 416 | 417 | 418 | 419 | 420 | 421 | 422 | 423 | 424 | 425 |
| 426 | 427 | 428 | 429 | 430 | 431 | 432 | 433 | 434 | 435 | 436 | 437 | 438 | 439 | 440 | 441 | 442 | 443 | 444 | 445 | 446 | 447 | 448 | 449 | 450 |
| 451 | 452 | 453 | 454 | 455 | 456 | 457 | 458 | 459 | 460 | 461 | 462 | 463 | 464 | 465 | 466 | 467 | 468 | 469 | 470 | 471 | 472 | 473 | 474 | 475 |
| 476 | 477 | 478 | 479 | 480 | 481 | 482 | 483 | 484 | 485 | 486 | 487 | 488 | 489 | 490 | 491 | 492 | 493 | 494 | 495 | 496 | 497 | 498 | 499 | 500 |
| 501 | 502 | 503 | 504 | 505 | 506 | 507 | 508 | 509 | 510 | 511 | 512 | 513 | 514 | 515 | 516 | 517 | 518 | 519 | 520 | 521 | 522 | 523 | 524 | 525 |
| 526 | 527 | 528 | 529 | 530 | 531 | 532 | 533 | 534 | 535 | 536 | 537 | 538 | 539 | 540 | 541 | 542 | 543 | 544 | 545 | 546 | 547 | 548 | 549 | 550 |
| 551 | 552 | 553 | 554 | 555 | 556 | 557 | 558 | 559 | 560 | 561 | 562 | 563 | 564 | 565 | 566 | 567 | 568 | 569 | 570 | 571 | 572 | 573 | 574 | 575 |
| 576 | 577 | 578 | 579 | 580 | 581 | 582 | 583 | 584 | 585 | 586 | 587 | 588 | 589 | 590 | 591 | 592 | 593 | 594 | 595 | 596 | 597 | 598 | 599 | 600 |
| 601 | 602 | 603 | 604 | 605 | 606 | 607 | 608 | 609 | 610 | 611 | 612 | 613 | 614 | 615 | 616 | 617 | 618 | 619 | 620 | 621 | 622 | 623 | 624 | 625 |
| 626 | 627 | 628 | 629 | 630 | 631 | 632 | 633 | 634 | 635 | 636 | 637 | 638 | 639 | 640 | 641 | 642 | 643 | 644 | 645 | 646 | 647 | 648 | 649 | 650 |
| 651 | 652 | 653 | 654 | 655 | 656 | 657 | | | | | | | | | | | | | | | | | | |

# Color “Blindness”

8% Males  0.5% Females

Red-Green (most common – traffic lights?)



0.05 × 0.05 = 0.0025, or just 0.25%

Blue-Yellow (not sex-linked)

WIKIPEDIA

# plot()

## Description

Generic function for plotting of **R** objects. For more details about the graphical parameter arguments, see par.

For simple scatter plots, plot.default will be used. However, there are `plot` methods for many **R** objects, including functions, data.frames, density objects, etc. Use `methods(plot)` and the documentation for these.

## Usage

```
plot(x, y, ...)
```

# plot()

## data(attitude);  ?attitude

### The Chatterjee–Price Attitude Data

### Description

From a survey of the clerical employees of a large financial organization, the data are aggregated from the questionnaires of the approximately 35 employees for each of 30 (randomly selected) departments. The numbers give the percent proportion of favorable responses to seven questions in each department.

```
> colnames(attitude)
[1] "rating"     "complaints" "privileges" "learning"   "raises"
[6] "critical"   "advance"
> plot(attitude$rating)
> plot(attitude$complaints,attitude$rating)
> plot(attitude[,1:2])     # Different plot???
```

# plot()

main=
sub=
col=
font=
family=
xlim=
xlab=

# ylab= ylim=

```
> plot(attitude$complaints,attitude$rating)
> plot(attitude$complaints,attitude$rating,ylab="Department Rating")
> plot(attitude$complaints,attitude$rating,ylim=c(0,100))
```

# Next Time

More Graphics and Scripting

# Additional Slides

# data.frame

**head(...)** : write out first few lines

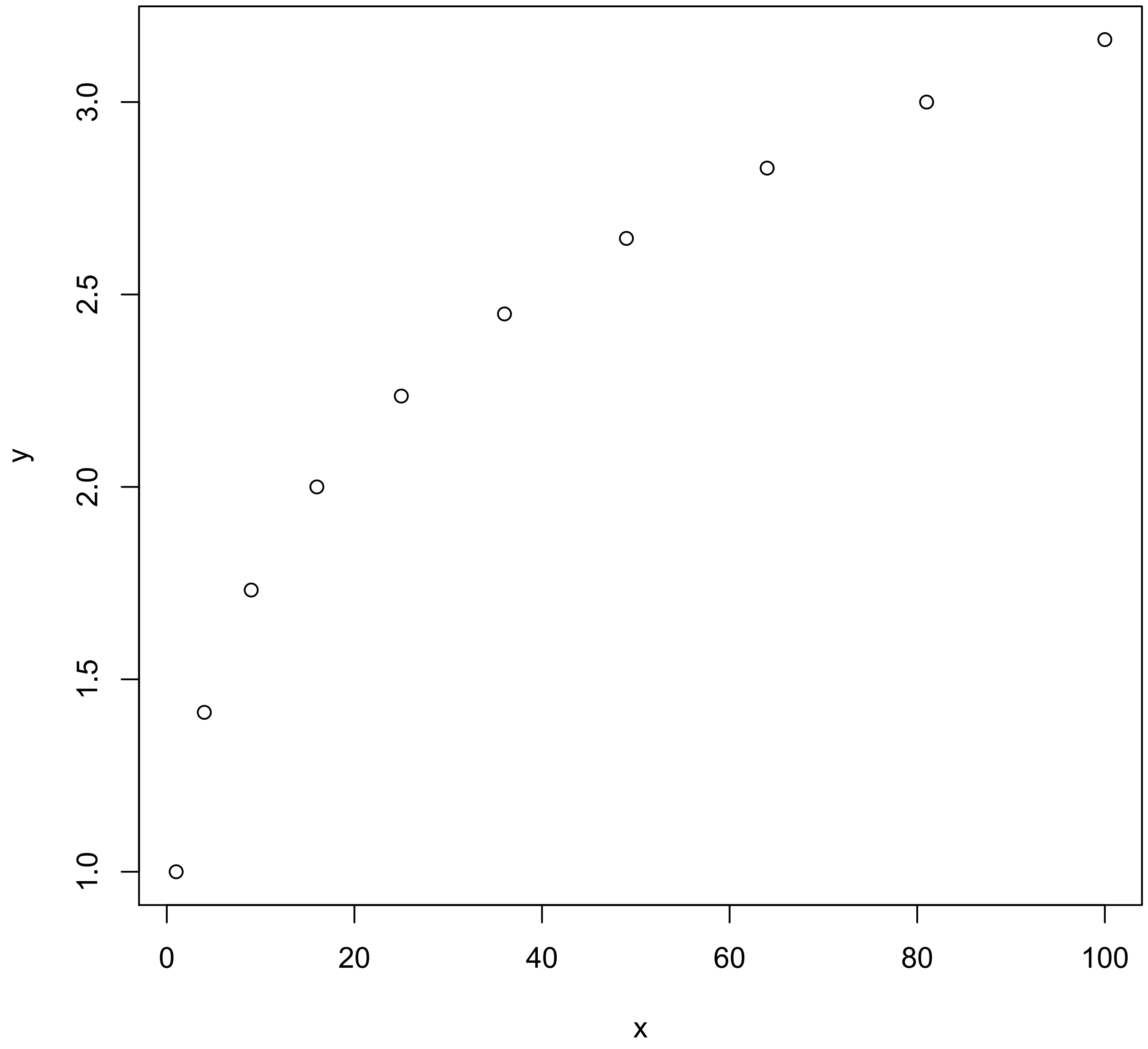
```
> head(quakes)

     lat   long depth mag stations
1 -20.42 181.62   562 4.8       41
2 -20.62 181.03   650 4.2       15
3 -26.00 184.10    42 5.4       43
4 -17.97 181.66   626 4.1       19
5 -20.42 181.96   649 4.0       11
6 -19.68 184.31   195 4.0       12
>
> class(quakes)
[1] "data.frame"
```

# plot(x,y)

x and y are two vectors
vector: a collection of numbers

```
> x = c(10,14,-23)
> y = c(-14,2,22)
> plot(x,y)
> x = seq(1,10)^2
> x
 [1]   1   4   9  16  25  36  49  64  81 100
> y = seq(1,10)^(1/2)
> y
 [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490
2.645751 2.828427
 [9] 3.000000 3.162278
> plot(x,y)
>
```

# Plot x versus y



> plot(x,y)

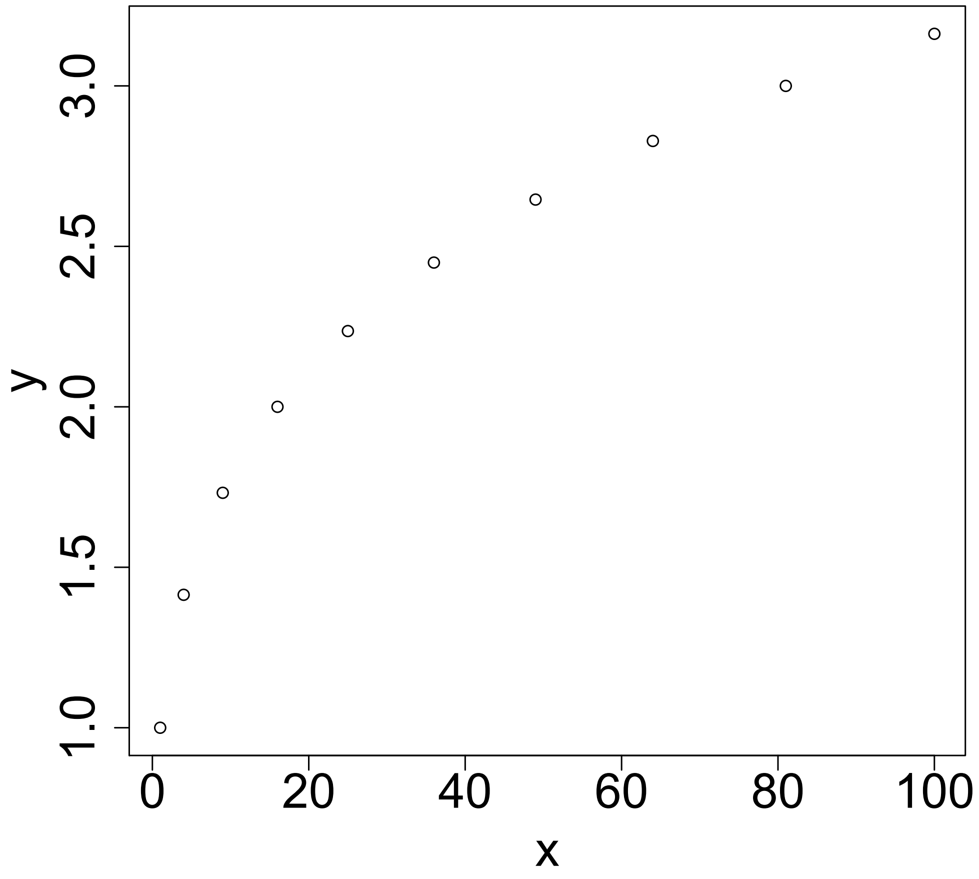
Plots a set of points
This is a scattergram

Tick marks and axis labels
are automatically generated

# How can I beautify this plot?

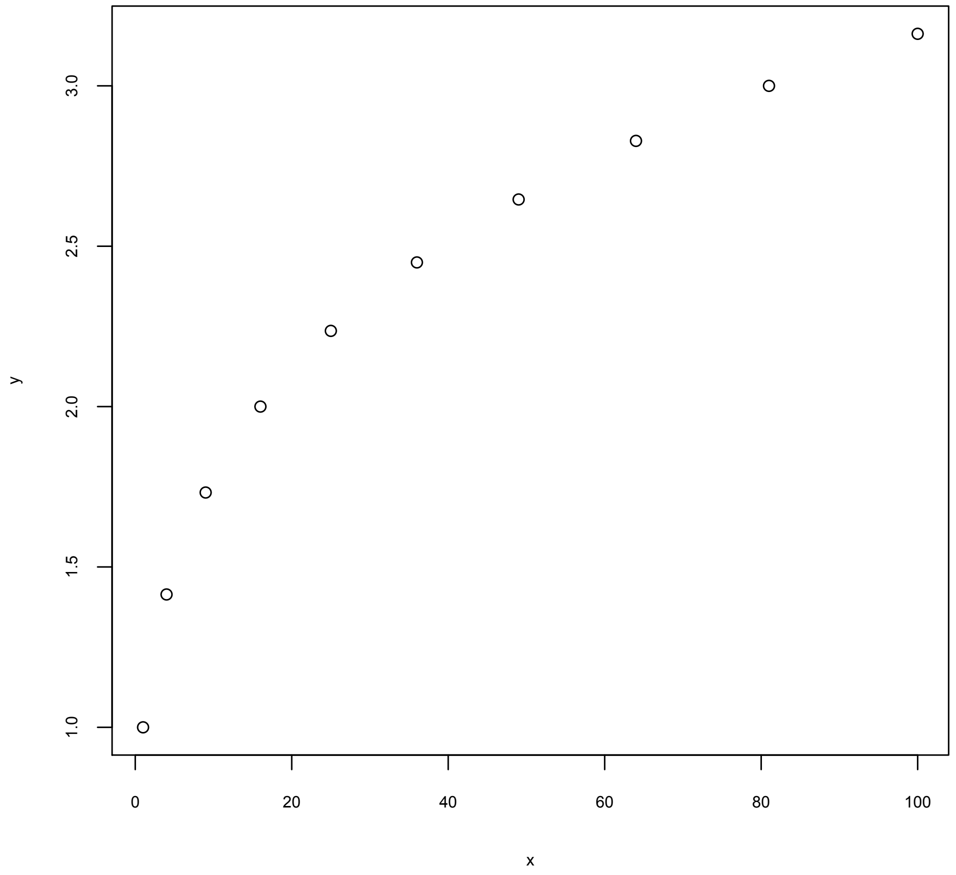
- Plot enhancements
    - Increase font size
    - Add plot title
    - Add better x and y labels
    - Multiple plots per page
- We do these one at a time

# Font size



```
> par(ps=24)
> plot(x,y)
```

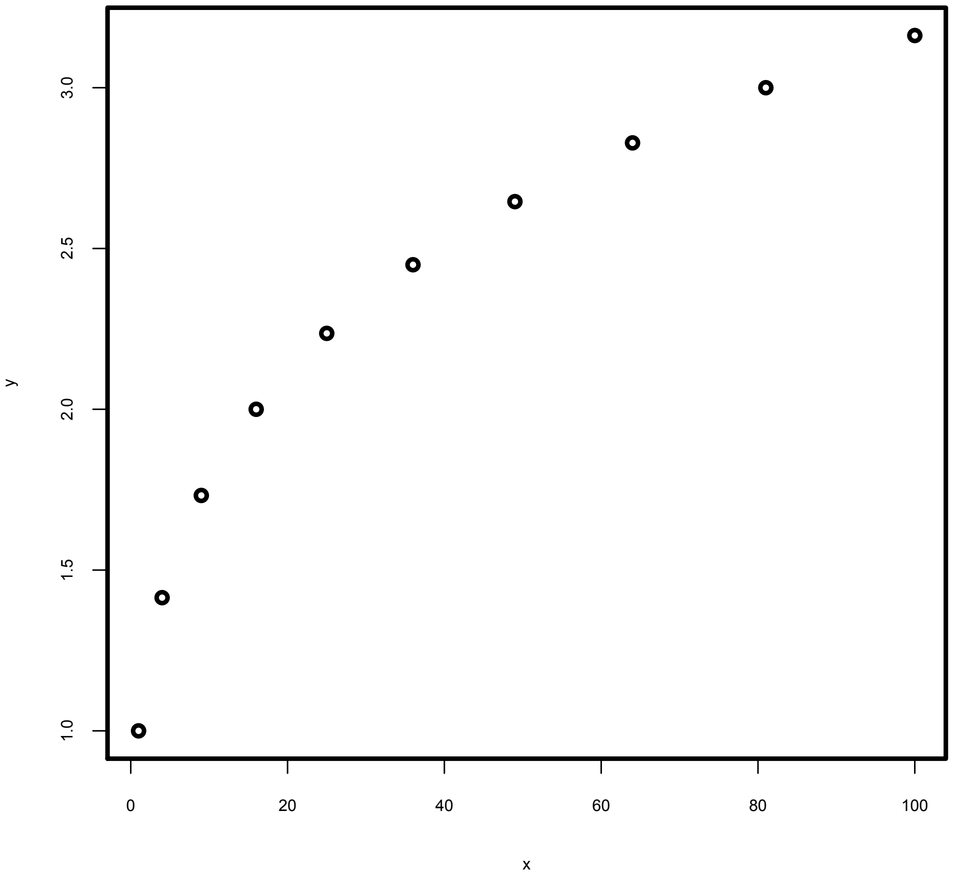
24 point font



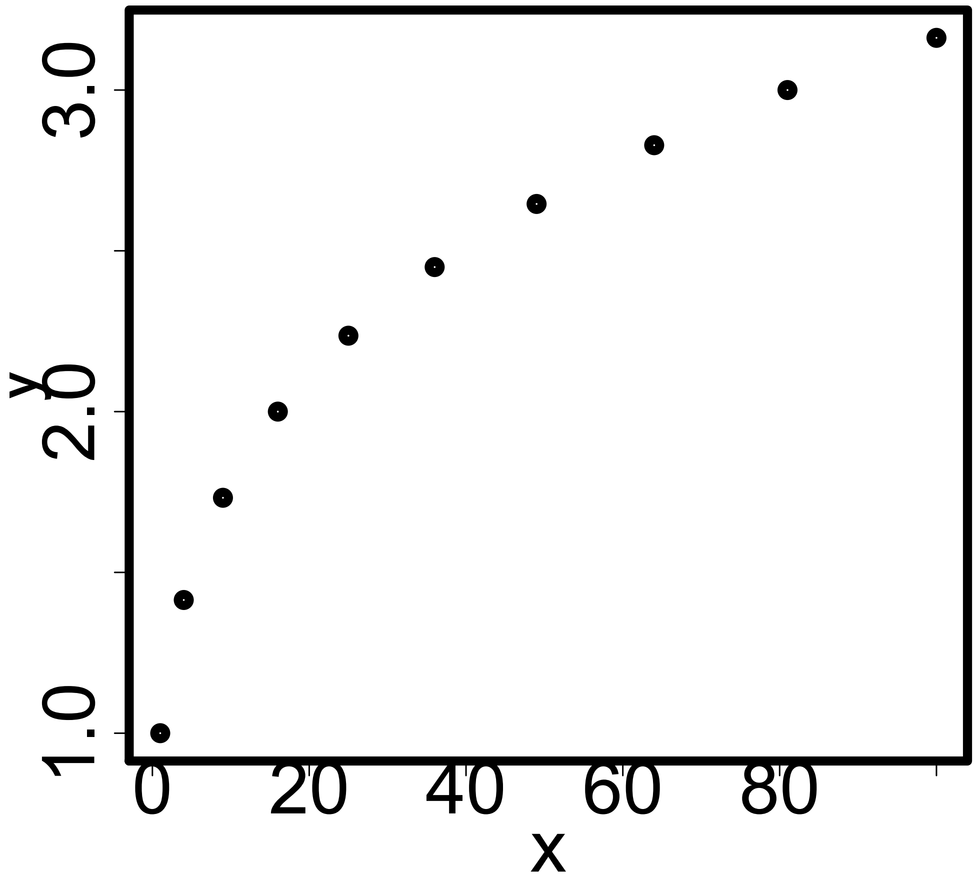
```
> par(ps=8)
> plot(x,y)
```

8 point font

# Line Width



```
> par(lwd=3)
> plot(x,y)
```



```
> par(lwd=6,ps=24)
> plot(x,y)
```

# Some Comments

- In a given R session, the effects of the command par() remain in force
- Many options can be set with par() (see next slide)

```
> par()

$xlog
[1] FALSE

$ylog
[1] FALSE

$adj
[1] 0.5

$ann
[1] TRUE

$ask
[1] FALSE

$bg
[1] "transparent"

$bty
[1] "o"

$cex
[1] 1

$cex.axis
[1] 1

$cex.lab
[1] 1

$cex.main
[1] 1.2

$cex.sub
[1] 1

$cin
[1] 0.15 0.20

$col
[1] "black"

$col.axis
[1] "black"

$col.lab
[1] "black"

$col.main
[1] "black"

$col.sub
[1] "black"

$cra
[1] 10.8 14.4

$crt
[1] 0

$csi
[1] 0.2

$cxy
[1] 2.78437500 0.09051395

$din
[1] 7 7

$err
[1] 0

$family
[1] ""

$fg
[1] "black"

$fig
[1] 0 1 0 1

$fin
[1] 7 7

$font
[1] 1

$font.axis
[1] 1

$font.lab
[1] 1

$font.main
[1] 2

$font.sub
[1] 1

$lab
[1] 5 5 7

$las
[1] 0

$lend
[1] "round"

$lheight
[1] 1

$ljoin
[1] "round"

$lmitre
[1] 10

$lty
[1] "solid"

$lwd
[1] 6

$mai
[1] 1.02 0.82 0.82 0.42

$mar
[1] 5.1 4.1 4.1 2.1

$mex
[1] 1

$mfcol
[1] 1 1

$mfg
[1] 1 1 1 1

$mfrow
[1] 1 1

$mgp
[1] 3 1 0

$mkh
[1] 0.001

$new
[1] FALSE

$oma
[1] 0 0 0 0

$omd
[1] 0 1 0 1

$omi
[1] 0 0 0 0

$pch
[1] 1

$pin
[1] 5.76 5.16

$plt
[1] 0.1171429 0.9400000
0.1457143 0.8828571

$ps
[1] 36

$pty
[1] "m"

$smo
[1] 1

$srt
[1] 0

$tck
[1] NA

$tcl
[1] -0.5

$usr
[1]  -2.9600000 103.9600000
0.9135089   3.2487688

$xaxp
[1]   0 100   5

$xaxs
[1] "r"

$xaxt
[1] "s"

$xpd
[1] FALSE

$yaxp
[1] 1 3 4

$yaxs
[1] "r"

$yaxt
[1] "s"

>
```

**80 options!!**

# Quakes dataset

- One of the datasets in R is the quakes dataset
- This dataset is a data.frame with five columns:
    - (see next slide)

# ?quakes

Locations of Earthquakes off Fiji

Description:

The data set give the locations of 1000 seismic events of MB > 4.0. The events occurred in a cube near Fiji since 1964.

Format:

A data frame with 1000 observations on 5 variables.

| | | | |
|---|---|---|---|
| [,1] | lat | numeric | Latitude of event |
| [,2] | long | numeric | Longitude |
| [,3] | depth | numeric | Depth (km) |
| [,4] | mag | numeric | Richter Magnitude |
| [,5] | stations | numeric | Number of stations reporting |

# Where are the quakes?

- Plot Longitude versus Latitude
- Each point will thus denote a location

# What to plot

- We need two vectors
- Let us plot latitude versus longitude to get quake location

# Digression on data.frames

```
> head(quakes,4)
     lat   long depth mag stations
1 -20.42 181.62   562 4.8       41
2 -20.62 181.03   650 4.2       15
3 -26.00 184.10    42 5.4       43
4 -17.97 181.66   626 4.1       19
```

```
> str(quakes)
'data.frame':   1000 obs. of  5 variables:
 $ lat     : num  -20.4 -20.6 -26 -18 -20.4 ...
 $ long    : num  182 181 184 182 182 ...
 $ depth   : int  562 650 42 626 649 195 82 194 211 622 ...
 $ mag     : num  4.8 4.2 5.4 4.1 4 4 4.8 4.4 4.7 4.3 ...
 $ stations: int  41 15 43 19 11 12 43 15 35 19 ...
```

# Function : head(...)

**head** package:utils R Documentation

Return the First or Last Part of an Object

Description:

> Returns the first or last parts of a vector, matrix, table, data frame or function. Since ‘head()’ and ‘tail()’ are generic functions, they may also have been extended to other classes.

head(x, n = 6L, ...)

Returns first 6 lines by default

# Function : str()

str                    package:utils                    R Documentation

Compactly Display the Structure of an Arbitrary R Object

Description:

Compactly display the internal *str*ucture of an R object, a diagnostic function and an alternative to 'summary' (and to some extent, 'dput'). Ideally, only one line for each 'basic' structure is displayed. It is especially well suited to compactly display the (abbreviated) contents of (possibly nested) lists. The idea is to give reasonable output for *any* R object. It calls 'args' for (non-primitive) function objects.

```
> str(quakes)
'data.frame':   1000 obs. of  5 variables:
 $ lat     : num  -20.4 -20.6 -26 -18 -20.4 ...
 $ long    : num  182 181 184 182 182 ...
 $ depth   : int  562 650 42 626 649 195 82 194 211 622 ...
 $ mag     : num  4.8 4.2 5.4 4.1 4 4 4.8 4.4 4.7 4.3 ...
 $ stations: int  41 15 43 19 11 12 43 15 35 19 ...
```

# print only beginning of data.frame

```
> head(quakes,4)
     lat   long depth mag stations
1 -20.42 181.62   562 4.8       41
2 -20.62 181.03   650 4.2       15
3 -26.00 184.10    42 5.4       43
4 -17.97 181.66   626 4.1       19
```

```
> head(quakes[1],4)
     lat
1 -20.42
2 -20.62
3 -26.00
4 -17.97
```

There is a similar command called **tail(...)**
Care to guess what this function does?

# str(...)
# Information about an object

```
> str(quakes)
'data.frame':   1000 obs. of  5 variables:
 $ lat     : num  -20.4 -20.6 -26 -18 -20.4 ...
 $ long    : num  182 181 184 182 182 ...
 $ depth   : int  562 650 42 626 649 195 82 194 211 622 ...
 $ mag     : num  4.8 4.2 5.4 4.1 4 4 4.8 4.4 4.7 4.3 ...
 $ stations: int  41 15 43 19 11 12 43 15 35 19 ...

> str(quakes[1])
'data.frame':   1000 obs. of  1 variable:
 $ lat: num  -20.4 -20.6 -26 -18 -20.4 ...
```

# Plot: first attempt

```
> head(quakes[1],4)
     lat
1 -20.42
2 -20.62
3 -26.00
4 -17.97
```

```
> head(quakes[2],4)
    long
1 181.62
2 181.03
3 184.10
4 181.66
```

```
> plot(quakes[1],quakes[2])
Error in stripchart.default(x1, ...) : invalid plotting method
```

Why is there a problem?

# ?plot

Usage:

plot(x, y, ...)

Arguments:

**x: the coordinates of points in the plot.** Alternatively, a single plotting structure, function or _any R object with a 'plot' method_ can be provided.

**y: the y coordinates of points in the plot, _optional_ if 'x' is an appropriate structure.**

...: Arguments to be passed to methods, such as graphical parameters (see 'par'). Many methods will accept the following arguments:

# Diagnostic

- plot() expects as argument, either
  - two vectors for the first two arguments
  - a more complex structure for the first argument, and no second argument

# data.frame access

- We need to find a way to extract the first column of quakes as a collection of numbers
- Easiest way: use the **$** access notation

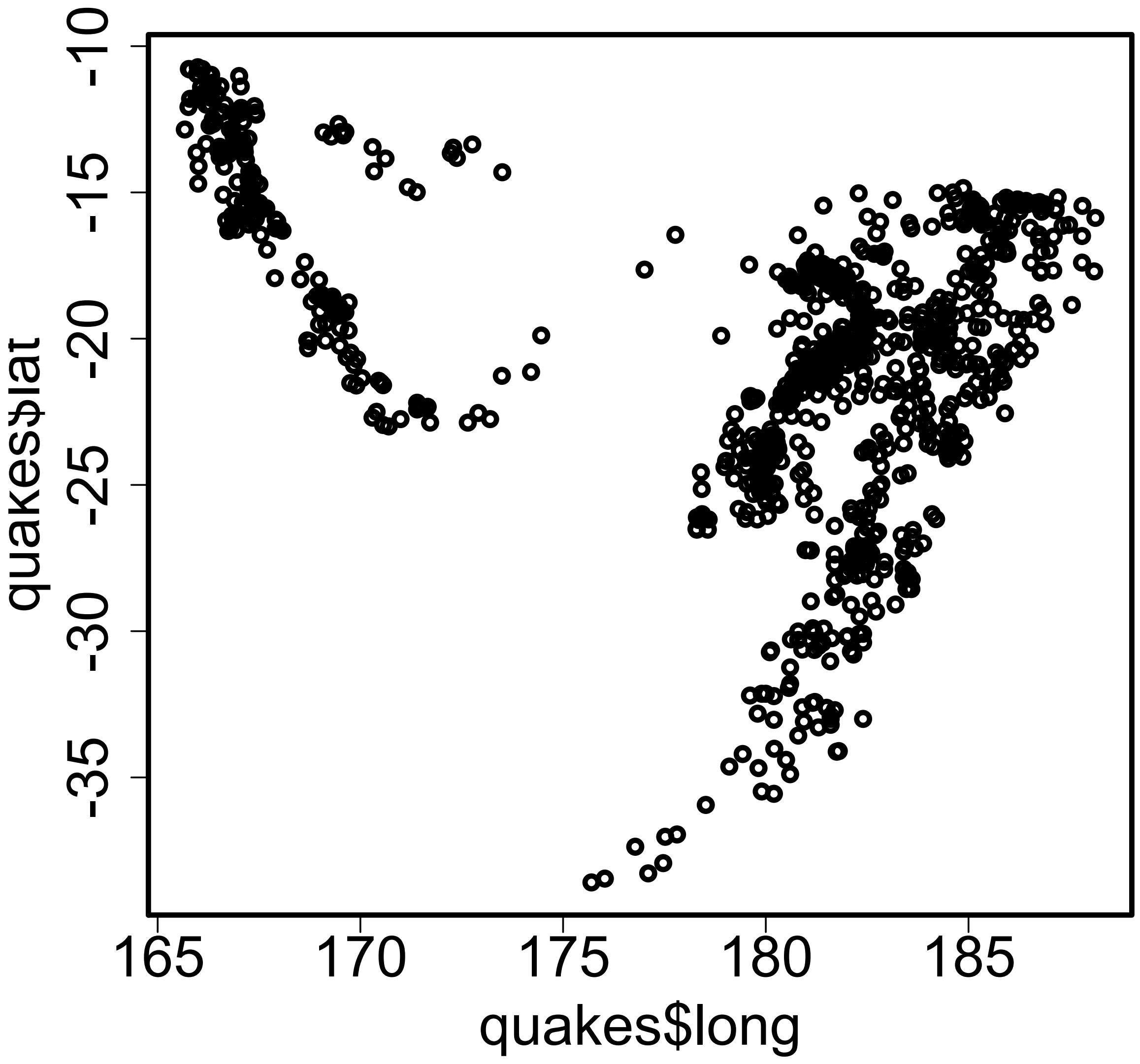
```
> names(quakes)
[1] "lat"     "long"    "depth"   "mag"     "stations"
```

```
> head(quakes$lat,4)
[1] -20.42 -20.62 -26.00 -17.97
```

quakes$lat is a **vector**

```
> class(quakes$lat)
[1] "numeric"
> class(c(1,2,3))
[1] "numeric"
```

# Plot: Second Attempt



```
> par(ps=24,lwd=3)
> plot(quakes$long, quakes$lat)
```

# Alternative data.frame Access

```
> plot(quakes[[2]], quakes[[1]])
> plot(quakes[,"long"], quakes[,"lat"])
> plot(quakes[,2], quakes[,1])
> plot(quakes$long, quakes$lat)
```

```
> with(quakes, plot(long, lat))
```

```
> attach(quakes)
> plot(long, lat)
> detach(quakes)
```

# with(...)

- At its most basic, with(...) is a way to avoid excessive typing
- First argument to with():
  - a data.frame
- Second argument to with():
  - any R expression
  - columns of the data.frame are accessed by name. Instead of **quakes$long**, **long is sufficient**

```
> plot(quakes$long, quakes$lat)
```

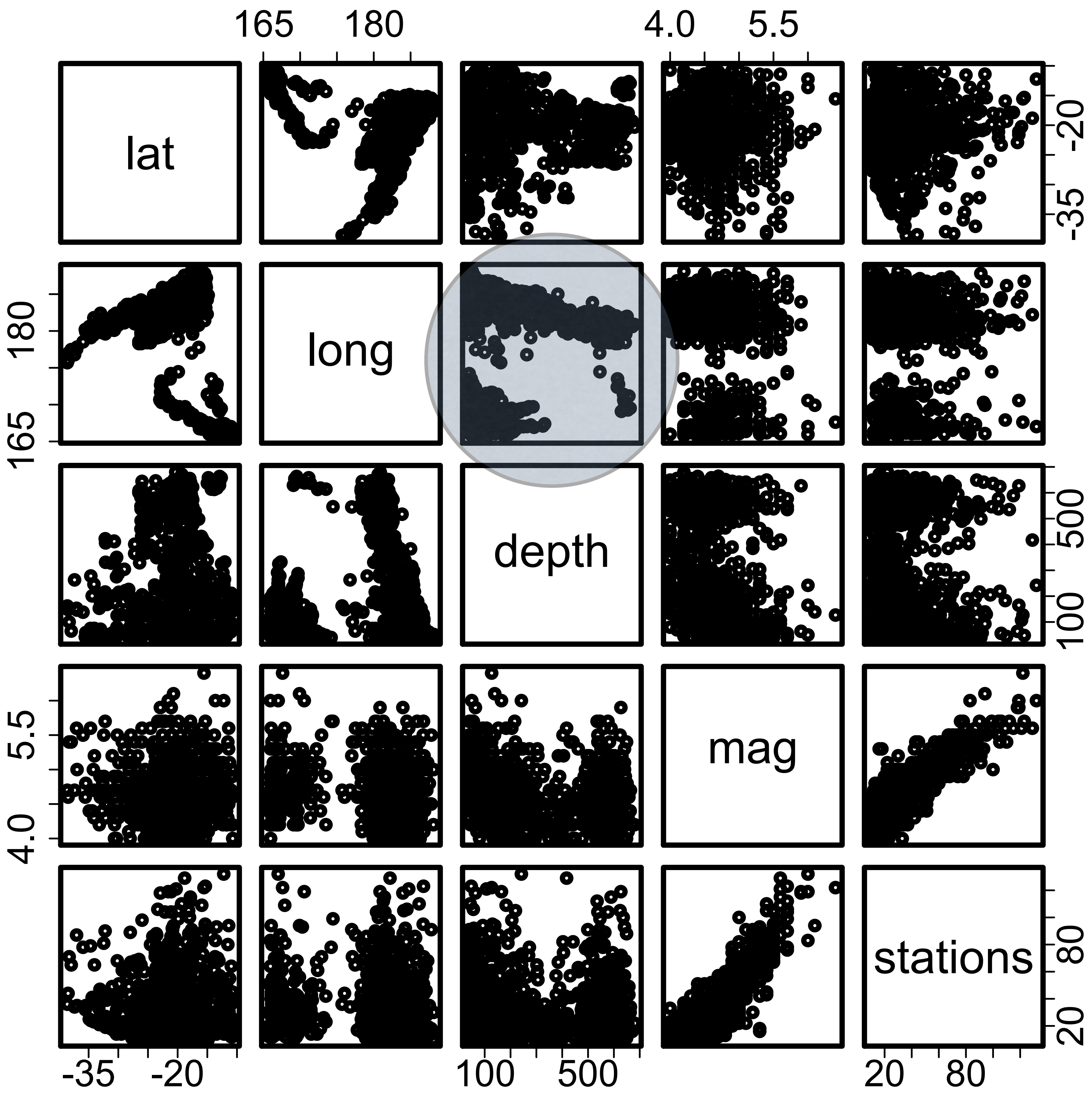
```
> with(quakes, plot(long, lat))
```

# Scattergrams

- The quake data.frame has 5 columns
- I have plotted longitude versus latitude
- Now, I'd like to plot every column against every other column
- Solution

```
plot(quakes)
```

Every column is plotted against every other column

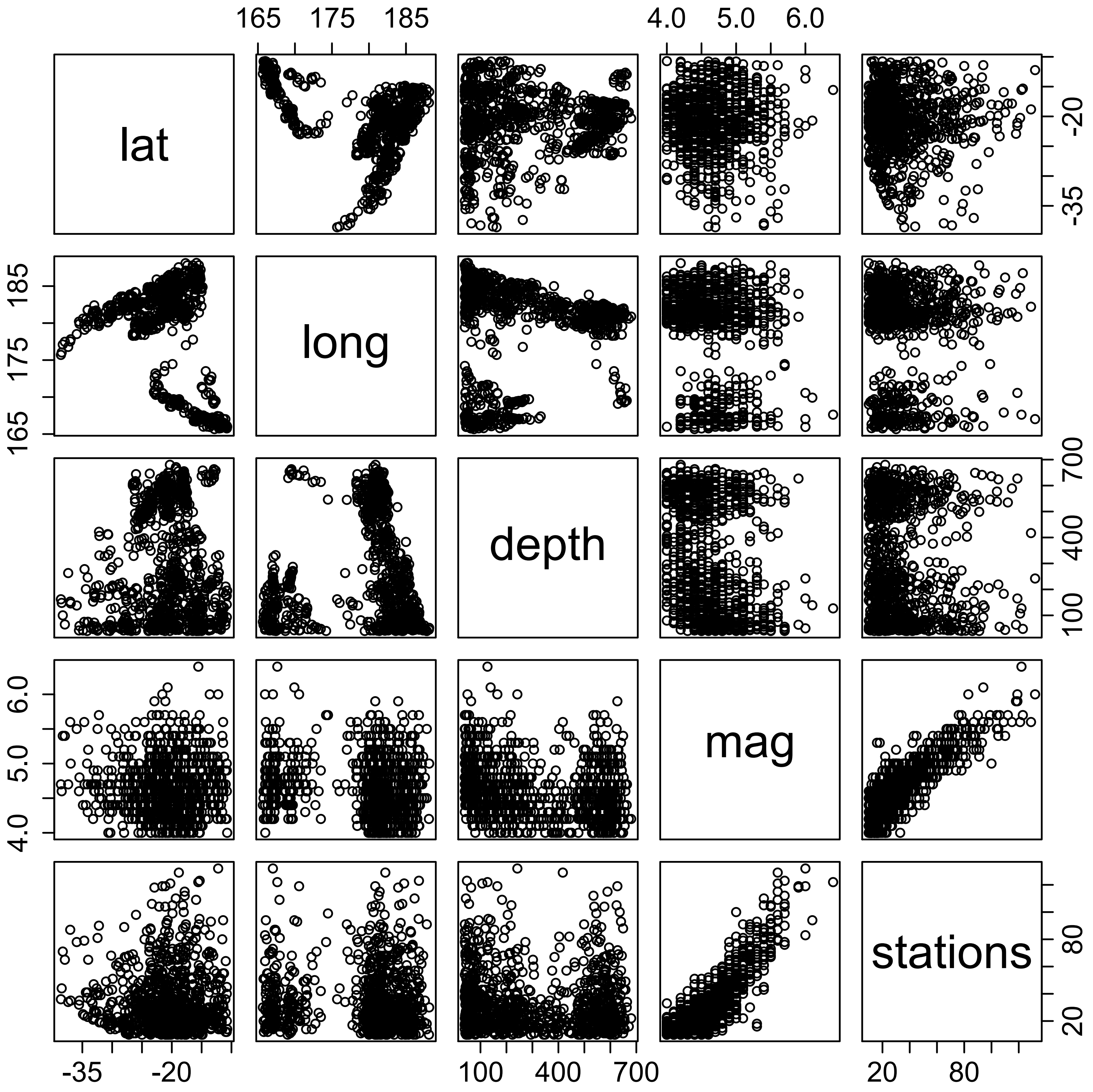


plot(quakes)

Fonts are slightly too large

Lines are somewhat too thick

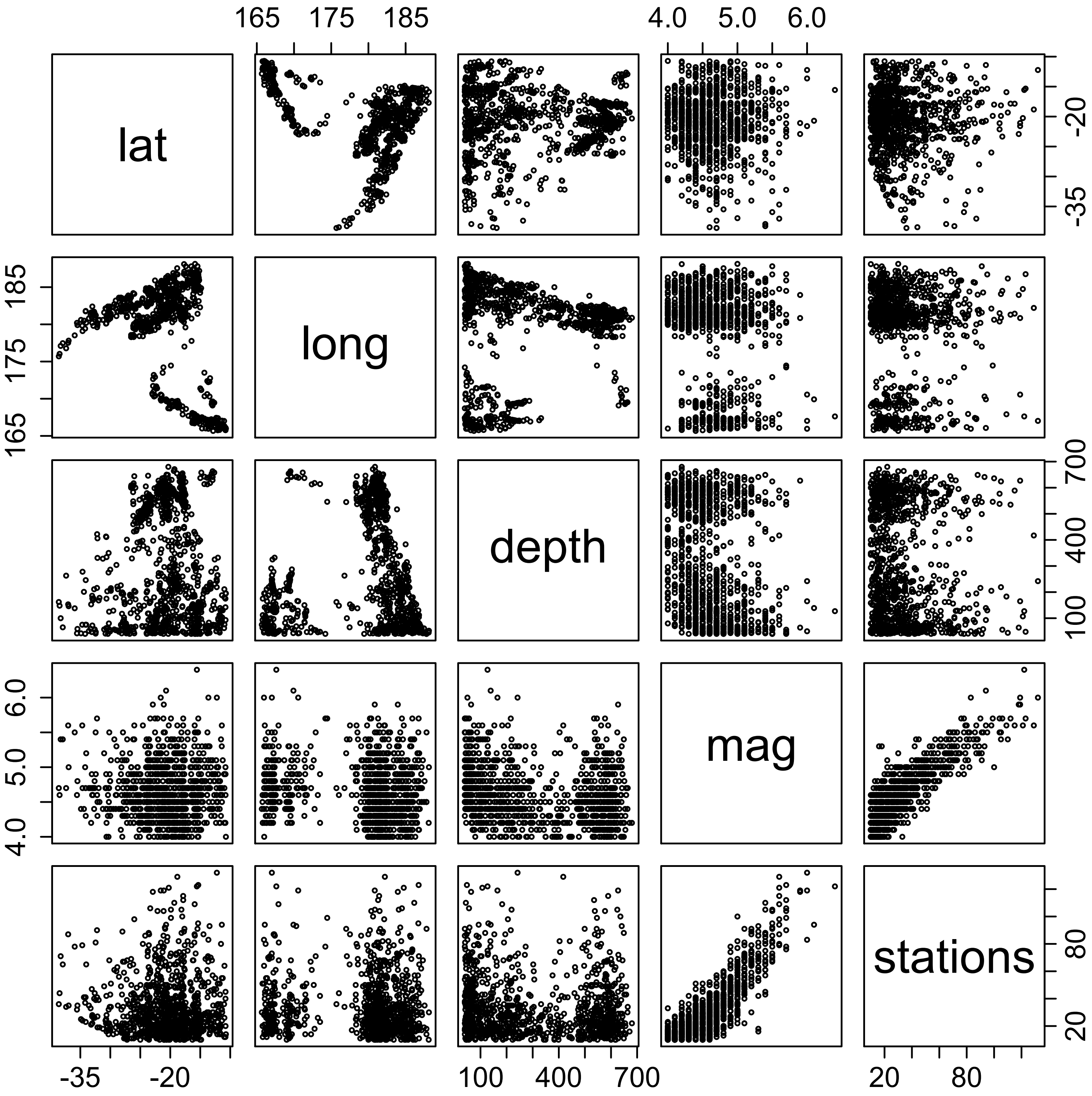
Plot of **long** (vertical axis) versus **depth** (horizontal axis)



```
> par(ps=18,lwd=1)
> plot(quakes)
```

plot is improved
Next problem:
circles are too large

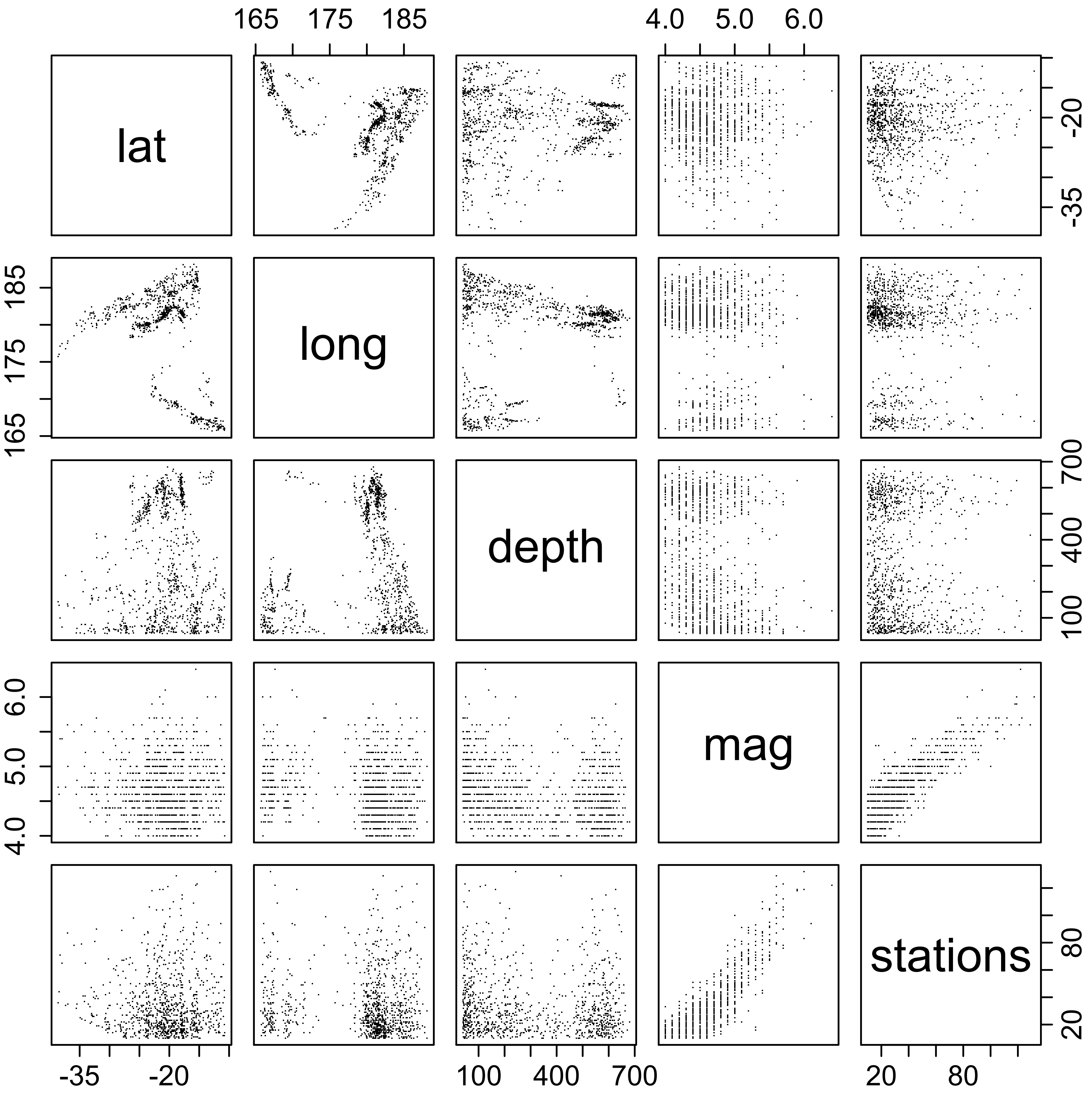
Look among the 80 options of par()



> plot(quakes,**cex=.5**)

**cex** stands for **C**haracter **Ex**pansion

The circle diameter decreased by 50%



```
> par(pch=".")
> plot(quakes)
```

I changed the circles to dots

But what if I want smaller circles?

# Plot formats

- Save images to files in on of several formats:
- png : no compression
  - the image can be reconstructed with 100 percent accuracy from the information in the saved file
- jpeg : compressed format
  - the image is compressed. The information in the file can only be used to reconstruct the original image to within some approximation
- Next slides: examples

# jpeg format

```
> d=quakes$depth
> l=quakes$lat
> jpeg()
> plot(l,d)
> dev.off()
```

The image is plotted to the file: **Rplot001.jpeg**

Instead of plotting to the screen, plot to a file.

```
> d=quakes$depth
> l=quakes$lat
> jpeg(“quake.jpg”)
> plot(l,d)
> dev.off()
```

The image will become stored in the file: **quake.jpg**

**The next use of plot() draws to the screen**

# ?jpeg

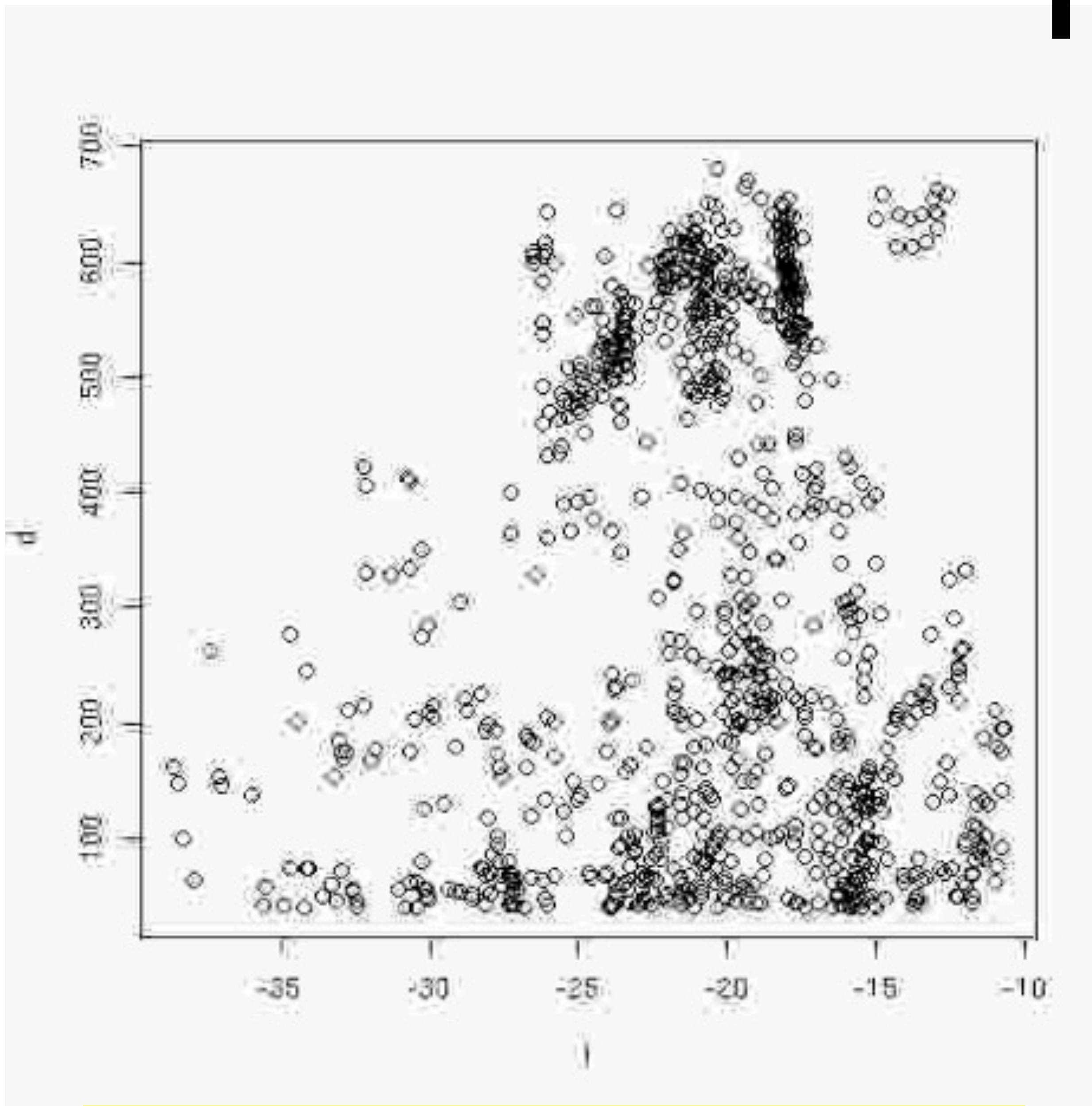
Description:

Graphics devices for JPEG, PNG or TIFF format bitmap files.

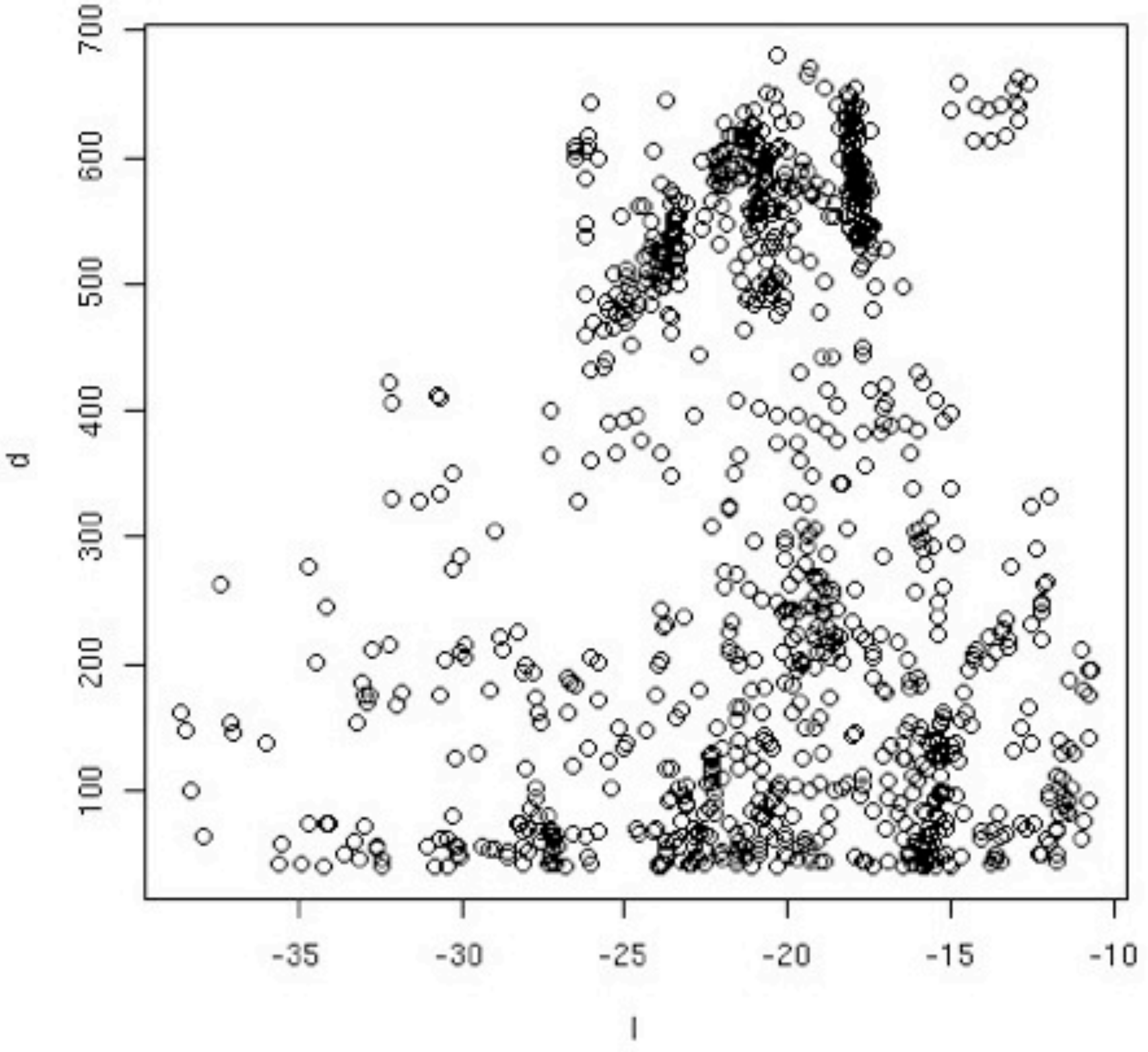
```
jpeg(filename = "Rplot%03d.jpeg",
     width = 480, height = 480, units = "px",
     pointsize = 12, quality = 75, bg = "white", res = NA, ...,
     type = c("cairo", "Xlib", "quartz"), antialias)
```

**Default quality: 75 percent**

# Effect of quality (jpg)

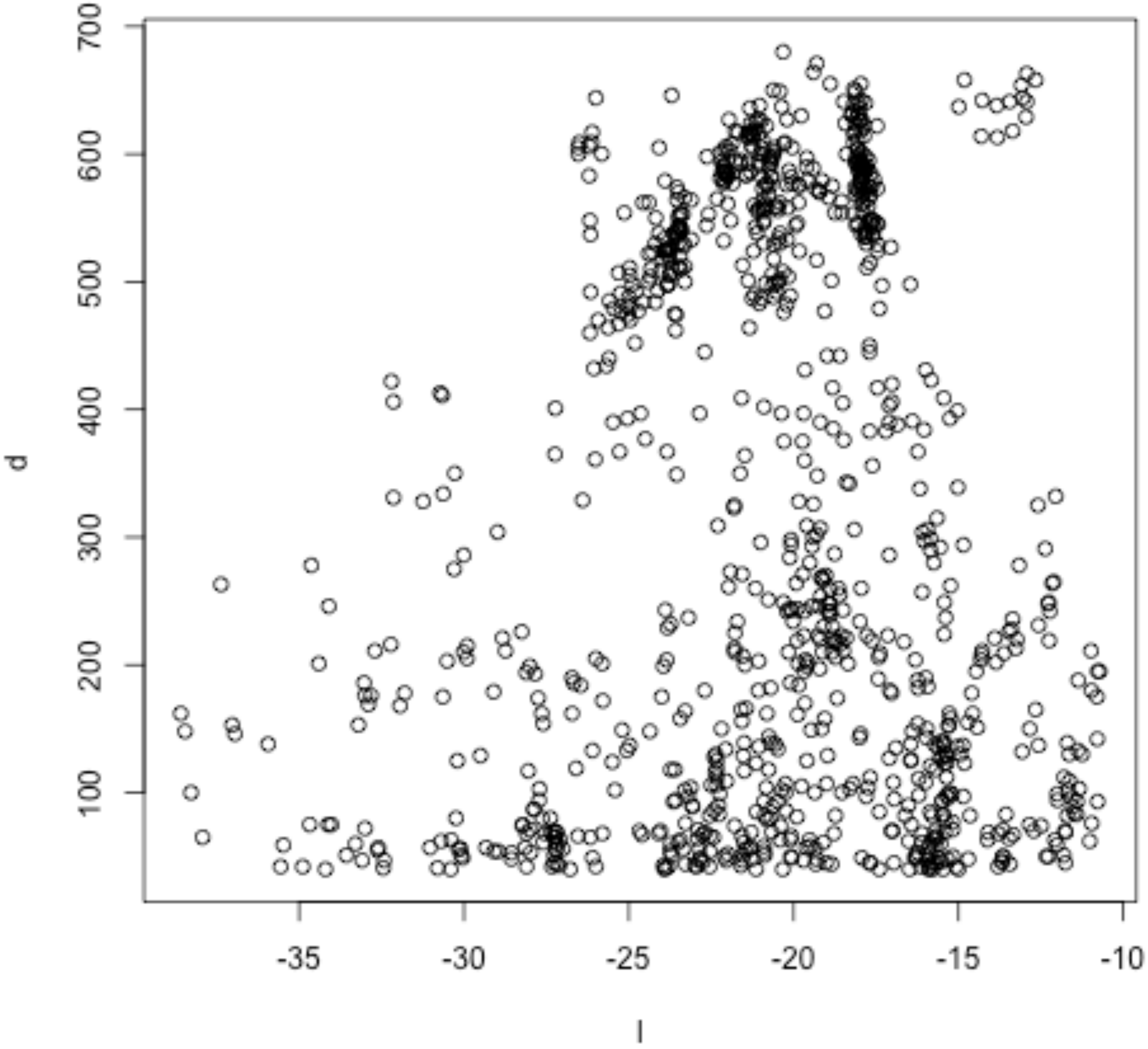


```
> jpeg("low_quality.jpg", quality=5)
> plot(l,d)
> dev.off()
```

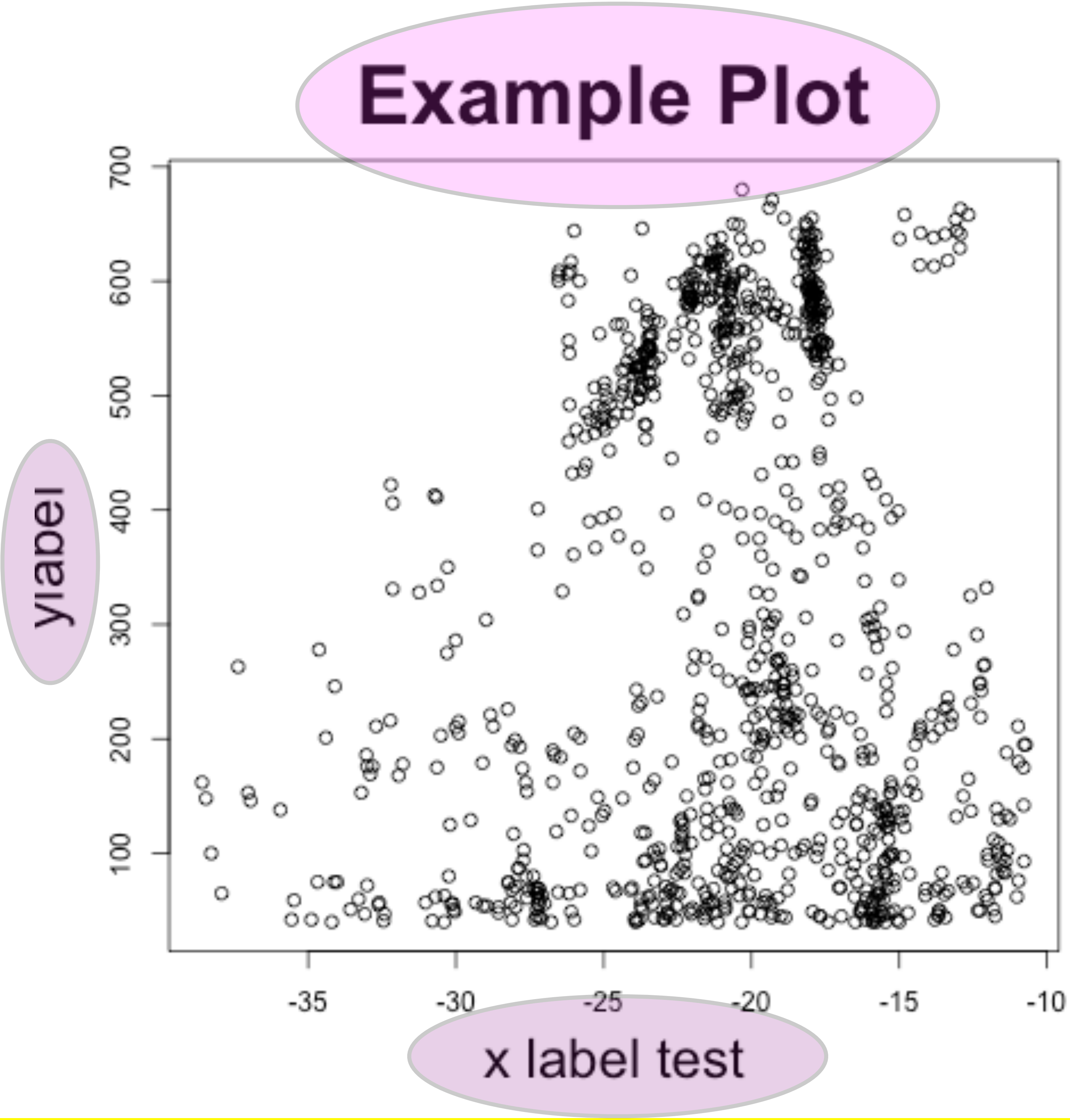


```
> jpeg("high_quality.jpg", quality=90)
> plot(l,d)
> dev.off()
```

# png file



```
> par(ps=18)
> png("high_quality.png")
> plot(l,d)
> dev.off()
```



```
> par(ps=18)
> png("high_quality.png")
> dev.off()
```

```
> png("high_quality.png")
> plot(l,d,main="Example Plot",xlab="x label test",
+ ylab="ylabel",cex.lab=2,cex.main=3)
> dev.off()
```

# More plotting

- We will examine our data through different types of plots throughout the course
- The principles always remain the same
- There are additional packages (lattice, ggplot2, ...) to do more sophisticated plots, or to simplify usage
    - we might look at some of these later in the course

# Next lesson

- We will discuss vectors, lists and data.frames more in depth
- These three structures are the backbone of R
- This will take one or two lessons depending on the questions you have and how many examples we work with

# First Lab

- Practice using R commands
- Do some plotting
- Do simple linear regression using lm()
- ?lm to find out more
- ?plot