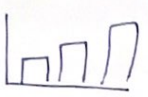



- `barplot()` → 
- `pie()` → 
- $y \sim x$ means
y depending on x
- | | |
|---|---|
| <code>cex.names</code> size of names label
<code>cex.axis</code> size of vertical axis label
<code>main</code> Title
<code>xlab</code> x-label
<code>ylab</code> y-label
<code>xlim</code> range of x
<code>ylim</code> range of y
<code>pch</code> (for dot plot, sets character) | <code>beside = TRUE</code> :
column plotted side by side.

<code>legend = TRUE</code> :
adds legend |
|---|---|
- `runif(length, min = -, max = -)` ← random noise
 - `sample(possible outcome, size = -, replace = -)` ← simulation, TRUE if duplicates okay, FALSE for unique
 - Factor**: Efficient because levels are internally coded as integers, labels are unique. Shows all possible values
 - Data frame**: 2D array-like structure in which each column contains values of 1 variable & each row 1 set of values from each column.
 - `str(data)`: get summary of dimensions & data.
 - `$` operator is used to access data frame columns. `with(data, column)` works in the same way.
 - `subset(data, condition)` → extracts the data that meets the condition.
 - `aggregate(measurements ~ factor, data = my data, FUN = myfun)`: first term is formula, FUN specifies function

Days Since:

```
> library(chron)
> dateEvent <- "YY-MM-DD"
> numOfDays <- Chron(dates = dateEvent, format = "%y-%m-%d")
> currentDate <- "YY-MM-DD"
> numOfDayCurrent <- Chron(dates = currentDate, format = c("%y-%m-%d"))
> as.numeric(numOfDayCurrent)
> numOfDayCurrent - numOfDay
```

Binary representation of fraction:

multiply by 2, take the whole number out
take remaining fraction, multiply by 2, take whole number
repeat

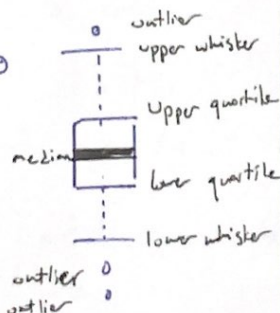
Exact storage of numbers:

If fraction is in form of $\frac{n}{2^m}$, $m < 1000$, then it can be stored exactly.

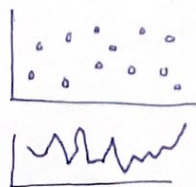
`hist(data)` →



`boxplot(vector)` →



`plot(data)`



`type = "p"`

`type = "l"`

Barplot, dotplot, pie chart display individual values

Histograms, boxplots, and QQ plots plot display distribution

Scatterplots display pairs of values.

for loop:

for (i in values) { command }

If statement:

if (condition) { command when TRUE } else { command when FALSE }

Setting graphical parameters: use with `par()`

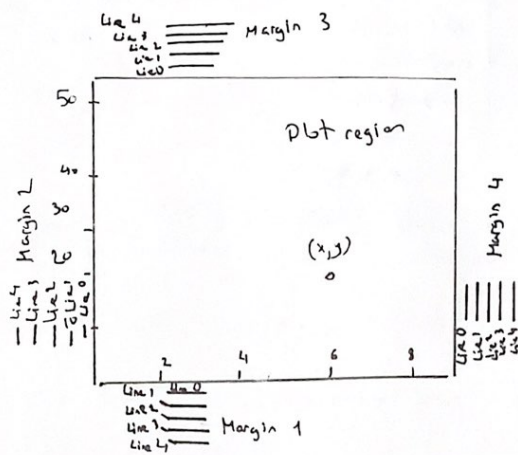
`mfig = c(m, n)`, draws m rows and n columns of plots per page

`mfig = c(i, j)`, tells r to draw figure in row i column j next.

`cex = value`, expands characters in plot region by value.

`mar = c(s1, s2, s3, s4)`, sets margins to given num of lines on each side

`oma = c(s1, s2, s3, s4)`, sets outer margins



Functions to ADD components to plot region of graphs

- `points(x, y, ...)` adds point
- `lines(x, y, ...)` adds line segment
- `text(x, y, labels, ...)` adds text
- `abline(a, b, ...)` adds line $y = a + bx$
- `abline(h = y, ...)` adds horizontal line
- `abline(v = x, ...)` adds vertical line

`title(main, sub, xlab, ylab, ...)` adds title, subtitle, x-label, y-label
`mtext(text, side, line, ...)` draws text in margins
`axis(side, at, labels, ...)` axis to the plot
`box(...)` adds a box around the plot

} outside plotting region

Functions are made up of:

- A header that includes the word function and an argument list (which might be empty)
- A body which includes a set of statements enclosed by curly braces `{}`.
- Can be called within body of other functions, all functions produce a single output.
- If there's no `return()`, value of last statement is returned.

Lattice:

• `dotplot(myFactor ~ myMeasurement | optionalFactor, data = myData)` ← dotplot



• `bwplot(myFactor ~ myMeasurement | optionalFactor, data = myData)` ← boxplot



• `xyplot(y ~ x | optionalFactor, data = myData)` ← Scatterplot



- We can condition on a variable by adding the vertical slash and the additional factor to condition on.
- `lm()` function can be used to fit predictive models where there are several predictor variables.
- The PRESS statistic can be used to decide which model is preferred; smaller the better.
- Predictions and prediction intervals can be computed for fitted models using `predict` function

`rpart(formula + anyextra variable, data = myData)`

To plot that use:

`plot()` and `text()`

`seq(start, end, step)`

`rep(m, x)` repeats for $m \times x$ times

`Summary(x)` computes several summary statistics

`length(x)` number of elements in x

`min(x)` min value of x

`max(x)` max value of x

`pmin(x, y)` pairwise minime of corresponding x and y

`pmax(x, y)` pairwise maxime of x and y

`range(x)` difference between max and min of x

`IQR(x)` difference between 1st and 3rd quartiles

`sd(x)` standard deviation of x

`var(x)` variance of the data

`diff(x)` successive differences of the values in x

`sort(x)` arranges the elements of x in ascending order