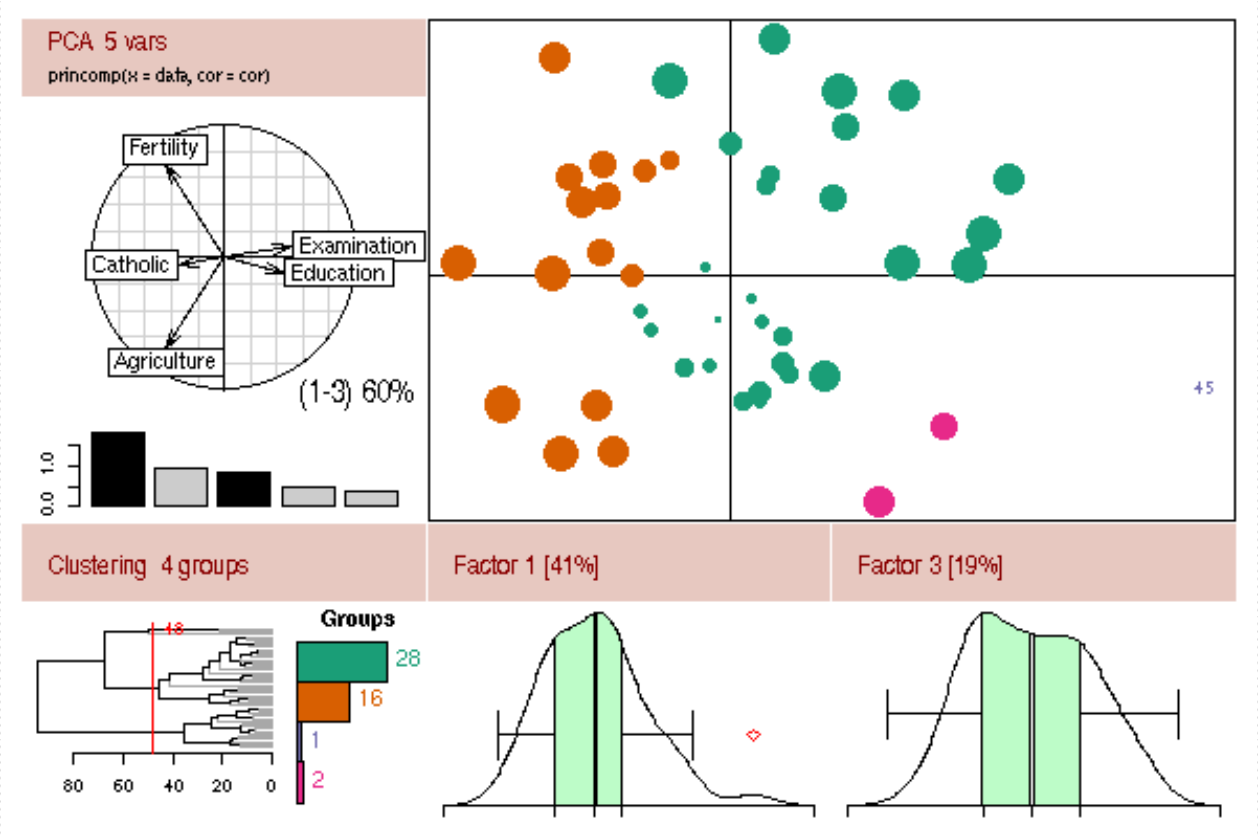
Statistics and Graphics in R



Plot Options

- **Multiple plots in a single graphic window**

  par(mfrow=c(2,3)) #allows 6 plots to appear on a page (2 rows of 3 plots each)

- **Adjusting graphical parameters**

  Types for plots and lines: type="l" (lines); type="b" (both); type="o" (overstruck);
  type="h" (high density)

  The line types using lty argument within plot() command;

  Colors and characters using col and pch argument

- **Putting text to the plot; controlling the text size**

  y <- as.data.frame(matrix(runif(36), ncol=3, dimnames=list(month.abb, LETTERS[1:3])))

  plot(y[,1], y[,2])

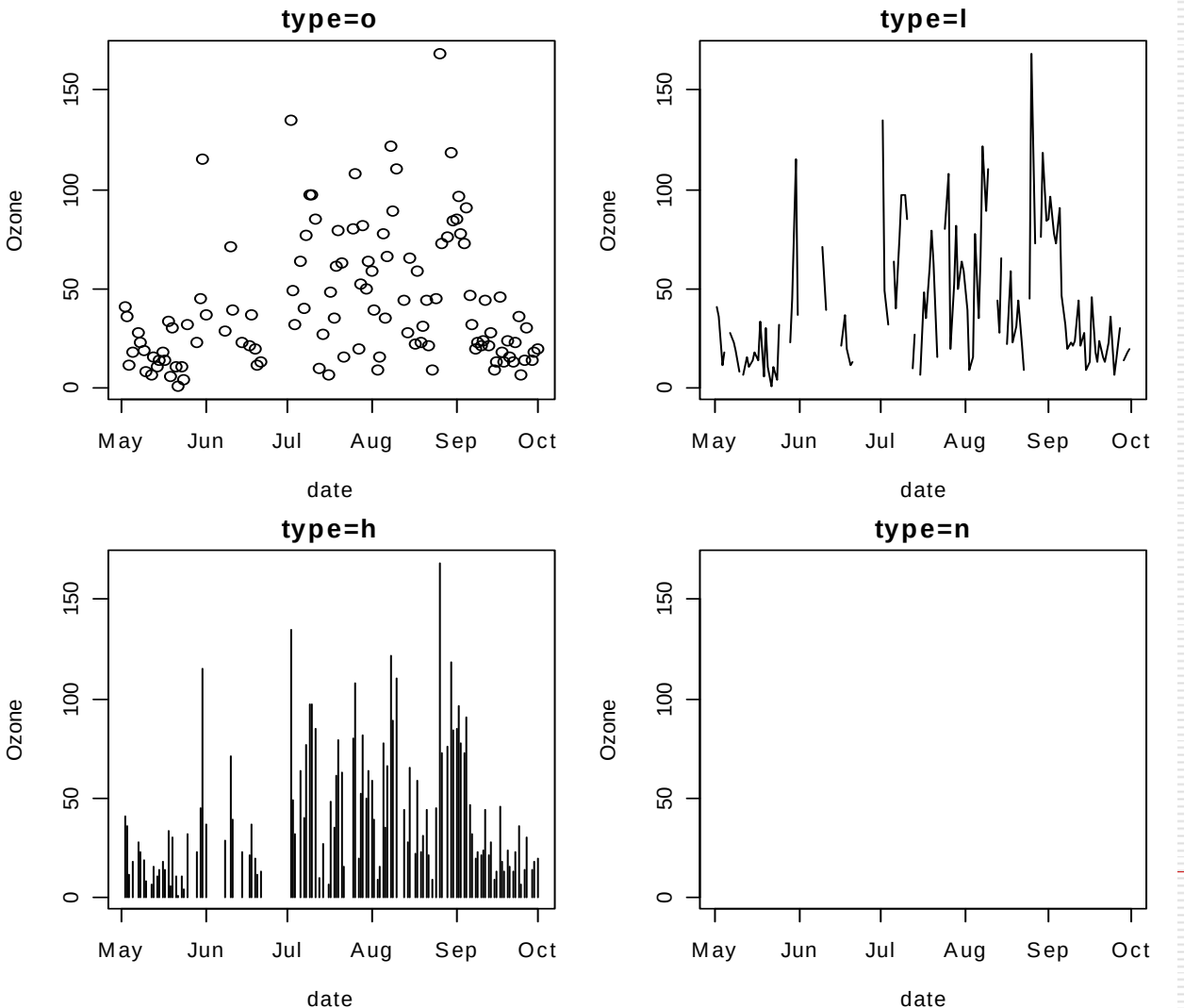
  plot(y[,1], y[,2], type="n", main="Plot of Labels"); text(y[,1], y[,2], rownames(y))

  plot(y[,1], y[,2], pch=20, col="red", main="Plot of Symbols and Labels");

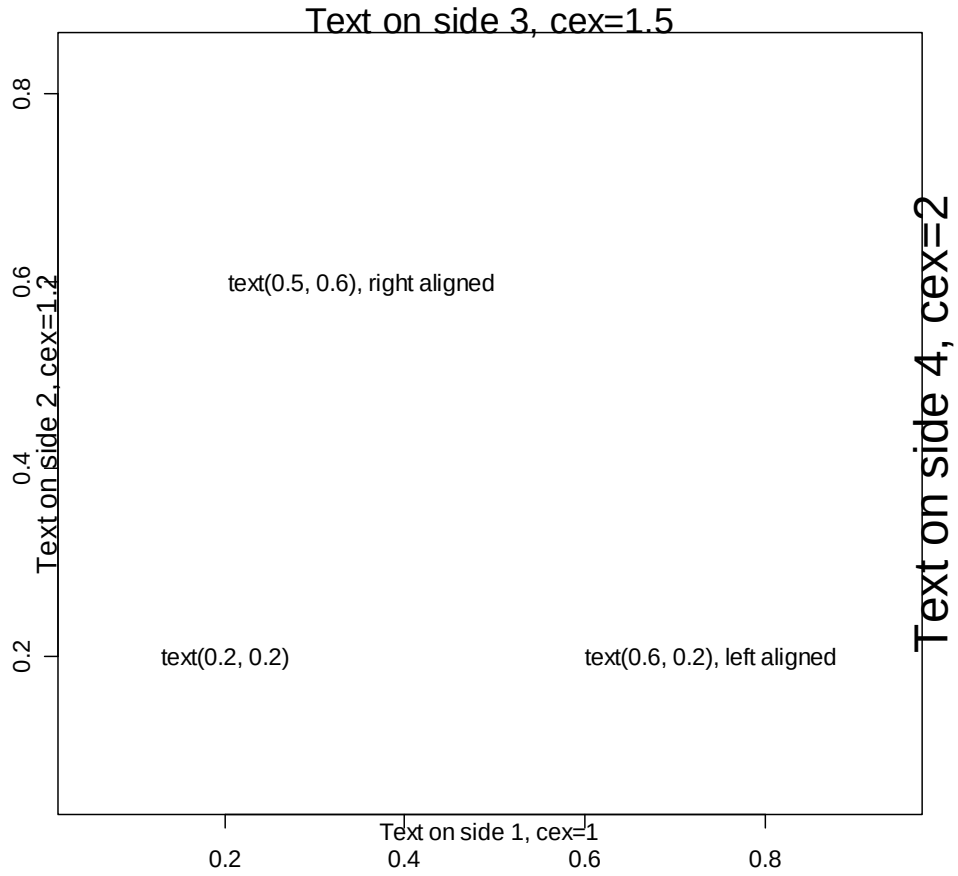
  text(y[,1]+0.03, y[,2], rownames(y)) # Plots both, symbols plus their labels.

Demo

```
data(airquality)
names(airquality)
airquality$date<-with(airquality, ISOdate(1973,Month,Day))

opar <- par(mfrow = c(2, 2),mar = c(4.1, 4.1, 2.1, 1.1))
plot(Ozone~date, data=airquality, main="type=o")
plot(Ozone~date, data=airquality,type="l", main="type=l")
plot(Ozone~date, data=airquality,type="h", main="type=h")
plot(Ozone~date, data=airquality,type="n", main="type=n")
par(opar)
```

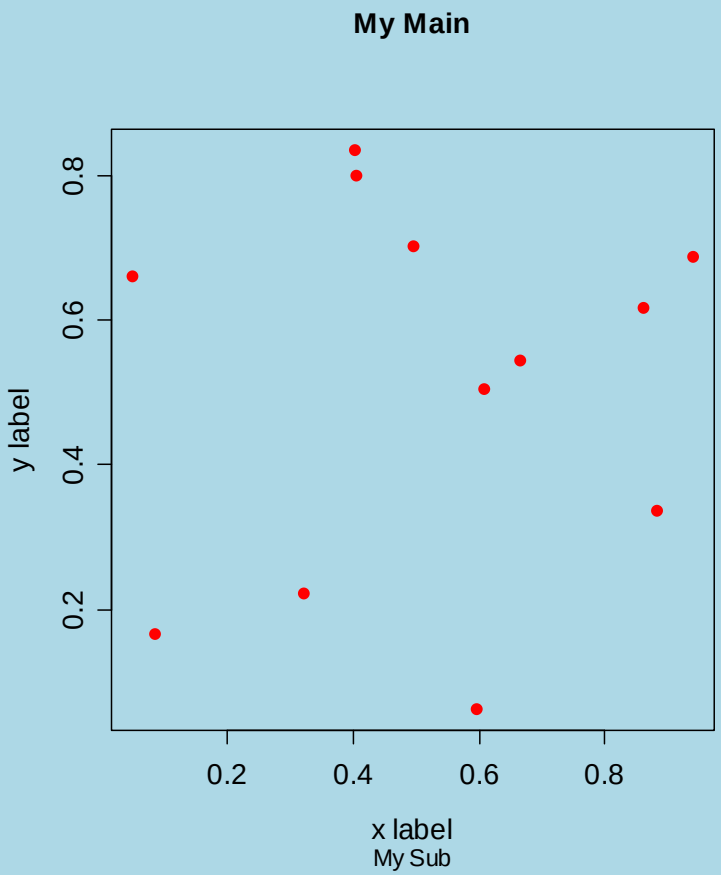

type=o
type=l
type=h
type=n
Ozone
0
50
100
150
May
Jun
Jul
Aug
Sep
Oct
date

Plot Options




```
plot(y[,1:2], xlab=" ", ylab=" ",
     type="n")
mtext("Text on side 1, cex=1",
     side=1,cex=1)
mtext("Text on side 2, cex=1.2",
     side=2,cex=1.2)
mtext("Text on side 3, cex=1.5",
     side=3,cex=1.5)
mtext("Text on side 4, cex=2",
     side=4,cex=2)
text(0.2, 0.2, "text(0.2, 0.2)")
text(0.6, 0.2, adj=0, "text(0.6,
     0.2), left aligned")
text(0.5, 0.6, adj=1, "text(0.5,
     0.6), right aligned")
```

Plot Options




```
op <- par(mar=c(8,8,8,8), bg="lightblue");
plot(y[,1], y[,2], type="p", col="red",
      cex.lab=1.2, cex.axis=1.2,
      cex.main=1.2, cex.sub=1, lwd=4,
      pch=20, xlab="x label", ylab="y
      label", main="My Main", sub="My
      Sub");
par(op)
```

Barplot

- y <- as.data.frame(matrix(runif(36), ncol=3, dimnames=list(month.abb, LETTERS[1:3])))

  barplot(as.matrix(y[1:4,]), ylim=c(0,max(y[1:4,])+0.1), beside=T)

  text(x=seq(1.5, 13, by=1)+sort(rep(c(0,1,2), 4)),
  y=as.vector(as.matrix(y[1:4,]))+0.02,labels=round(as.vector(as.matrix(y[1:4,])),2))
- ysub <- as.matrix(y[1:4,]); myN <- length(ysub[,1]);

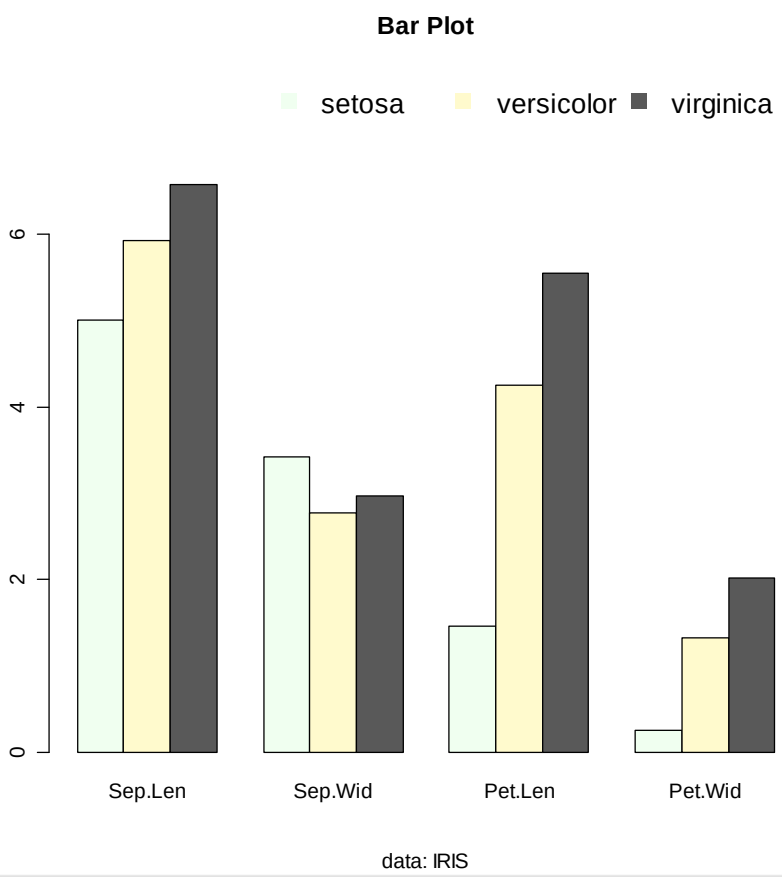
  mycol1 <- gray(1:(myN+1)/(myN+1))[-(myN+1)]; mycol2 <- sample(colors(),myN);

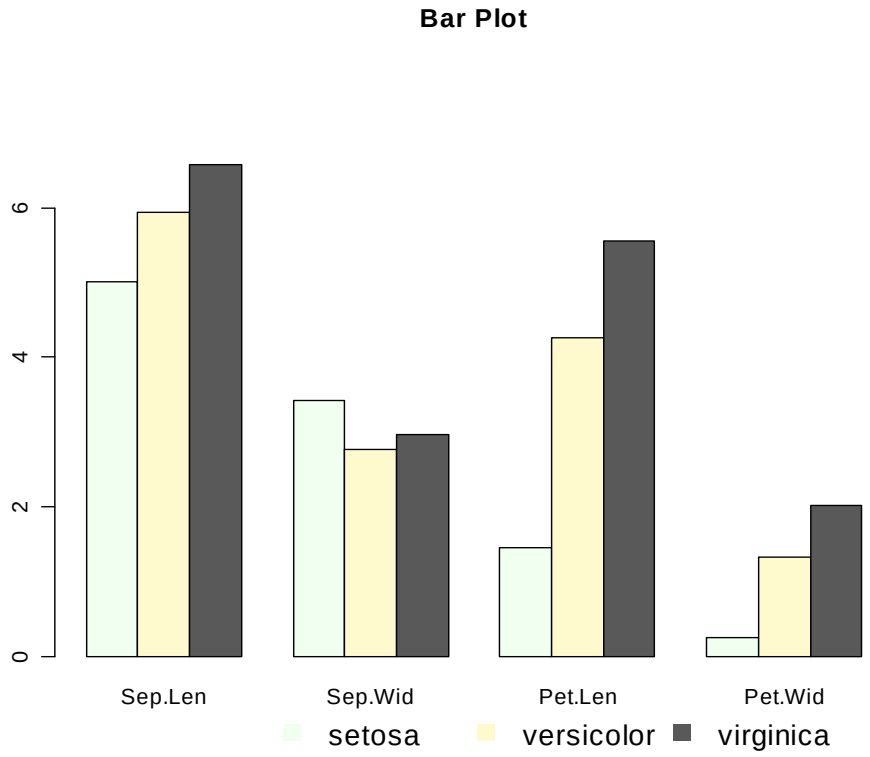
  barplot(ysub, beside=T, ylim=c(0,max(ysub)*1.2), col=mycol2, main="Bar Plot",
  sub="data: example");

  legend("topright", legend=row.names(ysub), cex=1.3, bty="n", pch=15, pt.cex=1.8,
  col=mycol2, ncol=myN)
- par(mar=c(10.1, 4.1, 4.1, 2.1)); par(xpd=TRUE);

  barplot(ysub, beside=T, ylim=c(0,max(ysub)*1.2), col=mycol2, main="Bar Plot");
  legend(x=4.5, y=-0.6, legend=row.names(ysub), cex=1.3, bty="n", pch=15, pt.cex=1.8,
  col=mycol2, ncol=myN)

Barplot


Bar Plot
setosa
versicolor
virginica
0
2
4
6
Sep.Len
Sep.Wid
Pet.Len
Pet.Wid
data: IRIS


Bar Plot
0
2
4
6
Sep.Len
Sep.Wid
Pet.Len
Pet.Wid
setosa
versicolor
virginica

Barplot with Confidence Interval

- stdev=data.frame(Sep.Len=tapply(iris[,1], iris$Species,sd), Sep.Wid =tapply(iris[,2], iris$Species,sd), Pet.Len=tapply(iris[,3], iris$Species,sd), Pet.Wid=tapply(iris[,4], iris$Species,sd))

  bar <- barplot(ysub, beside=T, ylim=c(0,max(ysub)*1.2), col=mycol2, main="Bar Plot")

  arrows(as.vector(bar), as.vector(ysub), as.vector(bar), as.vector(ysub)+ as.vector(as.matrix(stdev)), length=0.15, angle = 90);

  arrows(as.vector(bar), as.vector(ysub), as.vector(bar), as.vector(ysub)- as.vector(as.matrix(stdev)), length=0.15, angle = 90)

  legend("topright", legend=row.names(ysub), cex=1.3, bty="n", pch=15, pt.cex=1.8, col=mycol2, ncol=myN)

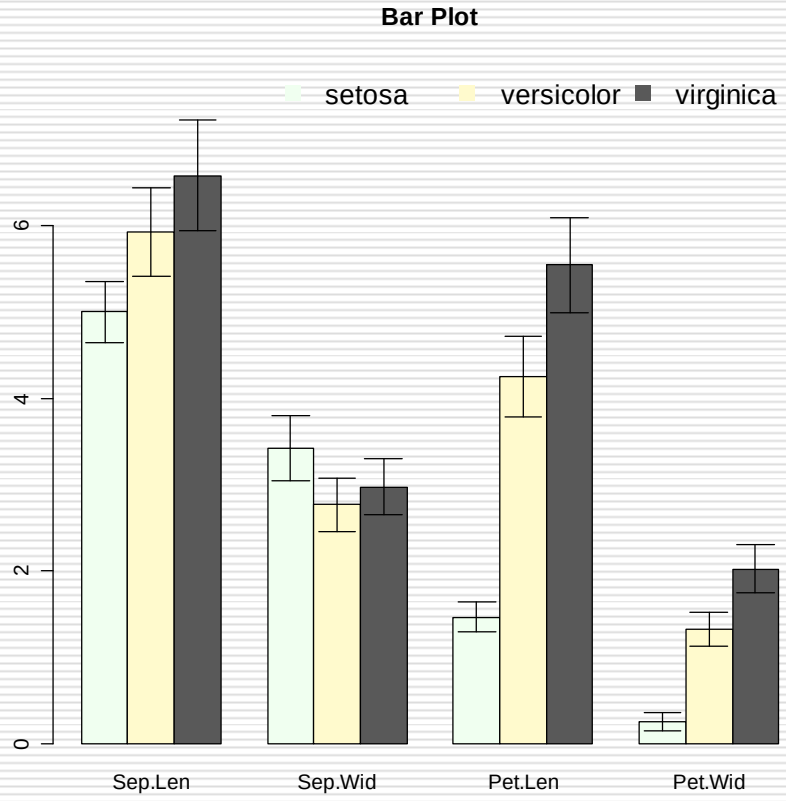
- require(gplots)

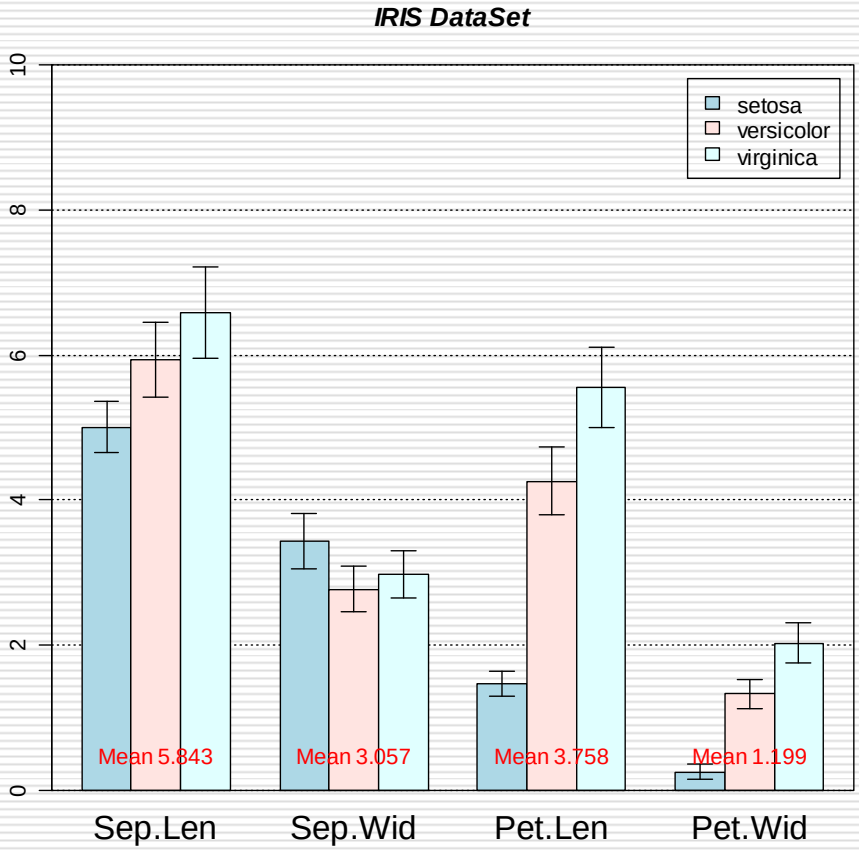
  mybarcol <- “gray20”; ci.l <- as.matrix(ysub-stdev); ci.u <- as.matrix(ysub+stdev)

  mp <- barplot2(ysub, beside = TRUE,col = c("lightblue", "mistyrose","lightcyan"), legend = rownames(ysub), ylim = c(0, 10), main = "IRIS DataSet", font.main = 4, col.sub = mybarcol, cex.names = 1.5, plot.ci = TRUE, ci.l = ci.l, ci.u = ci.u,plot.grid = TRUE)

  mtext(side = 1, at = colMeans(mp), line = -2, text = paste("Mean", formatC(colMeans(ysub))), col = "red")

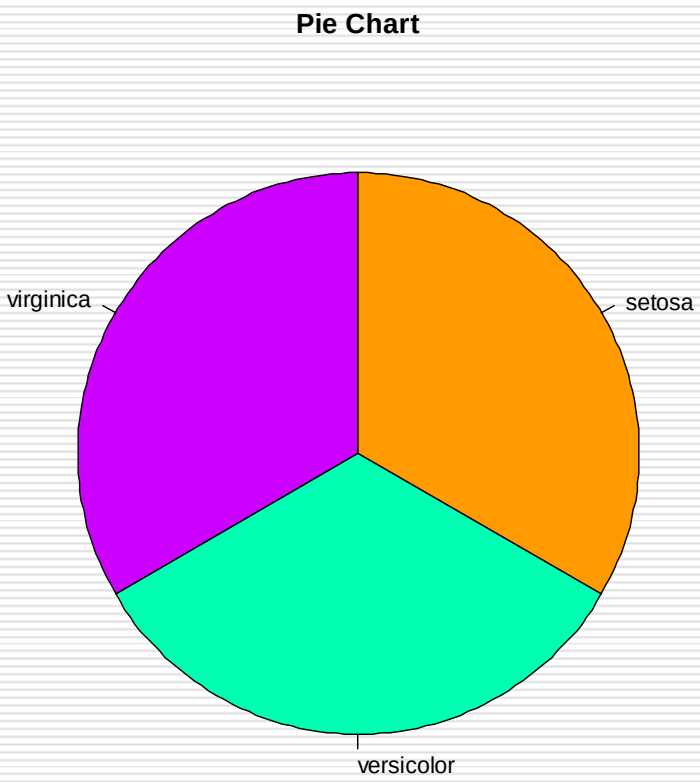
  box()

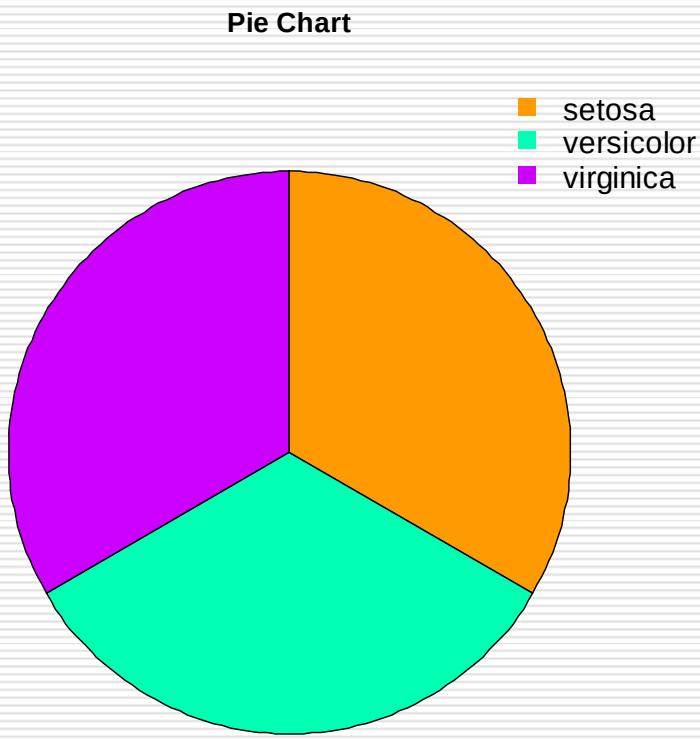

Bar Plot
setosa
versicolor
virginica
0
2
4
6
Sep.Len
Sep.Wid
Pet.Len
Pet.Wid


IRIS DataSet
setosa
versicolor
virginica
0
2
4
6
8
10
Mean 5.843
Mean 3.057
Mean 3.758
Mean 1.199
Sep.Len
Sep.Wid
Pet.Len
Pet.Wid

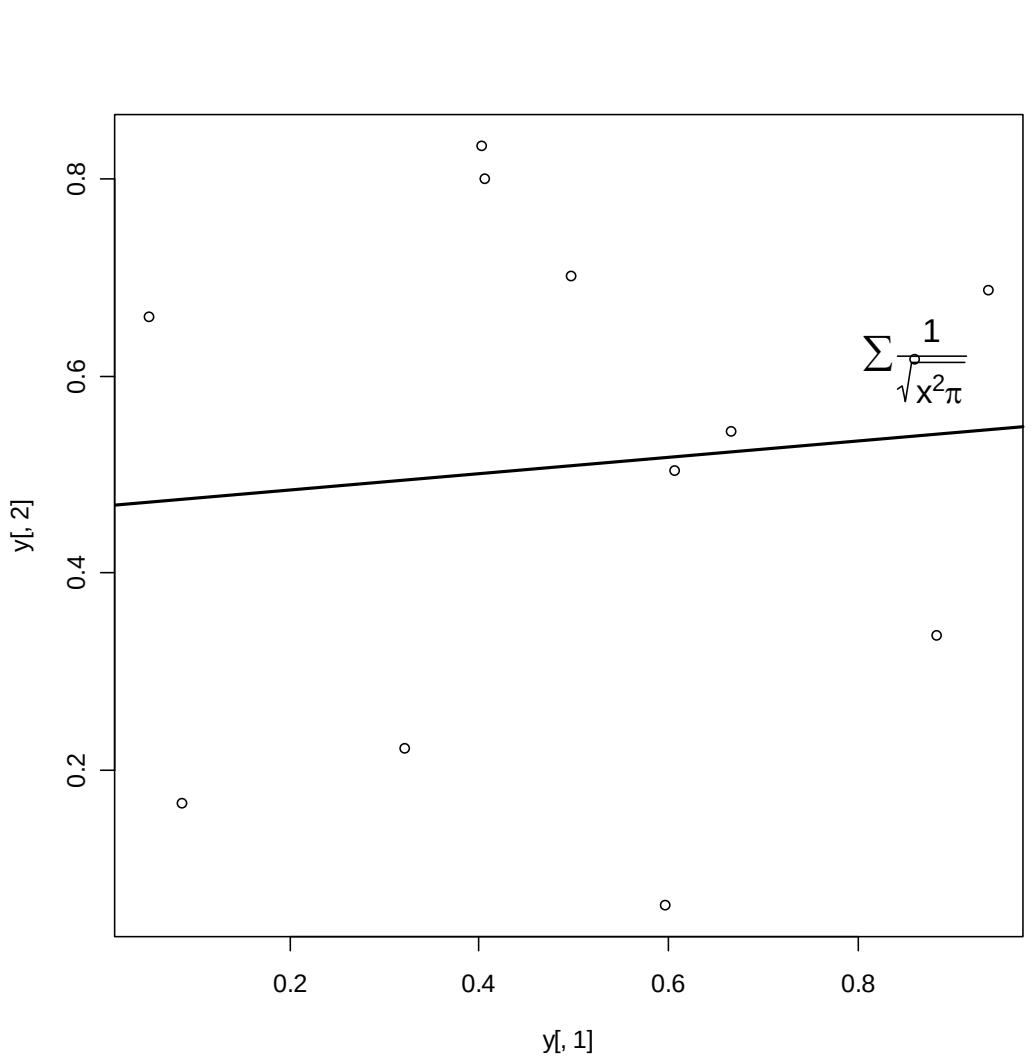
Pie Chart

- y <- table(iris$Species)
- pie(y, col=rainbow(length(y), start=0.1, end=0.8), main="Pie Chart", clockwise=T) # Plots a simple pie chart.
- pie(y, col=rainbow(length(y), start=0.1, end=0.8), labels=NA, main="Pie Chart", clockwise=T); legend("topright", legend=row.names(y), cex=1.3, bty="n", pch=15, pt.cex=1.8, col=rainbow(length(y), start=0.1, end=0.8), ncol=1)


Pie Chart
virginica
setosa
versicolor


Pie Chart
setosa
versicolor
virginica

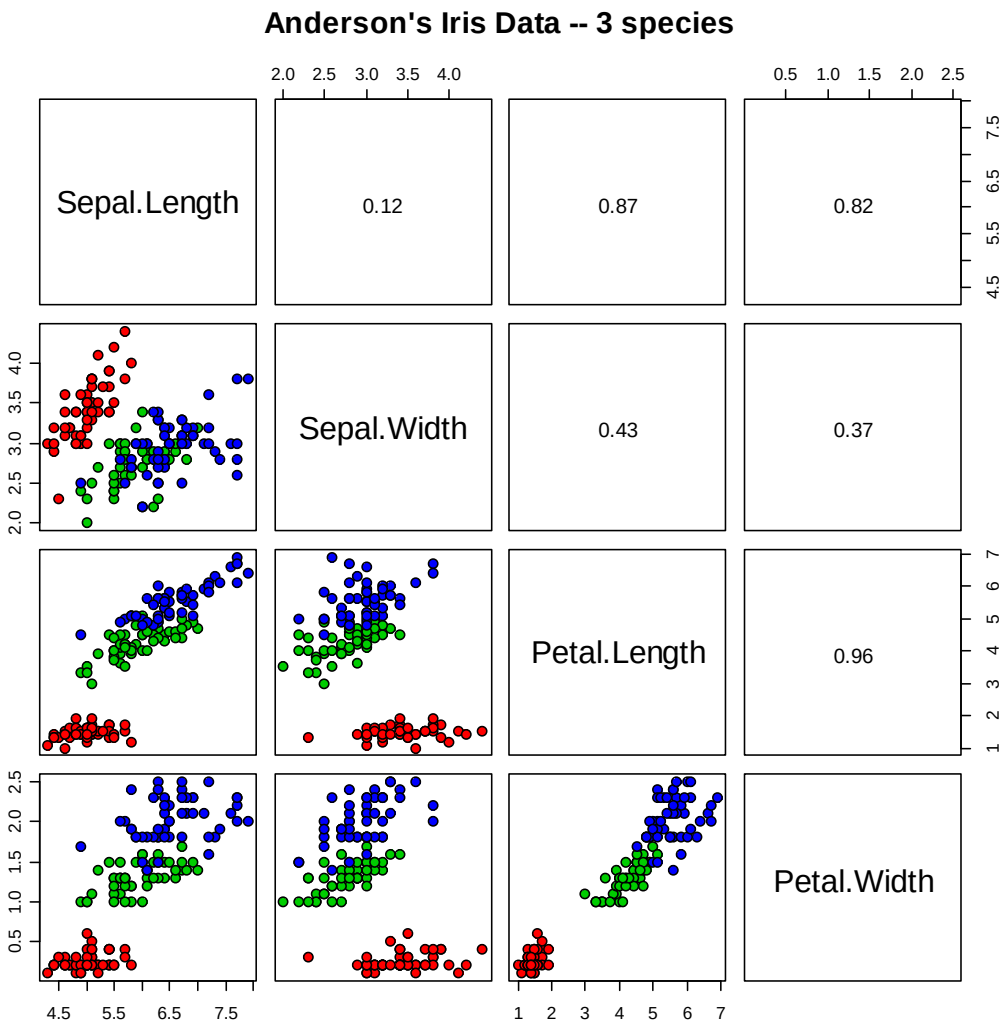
Scatter Plots



- **Adds a regression line and a mathematical formula to the plot**

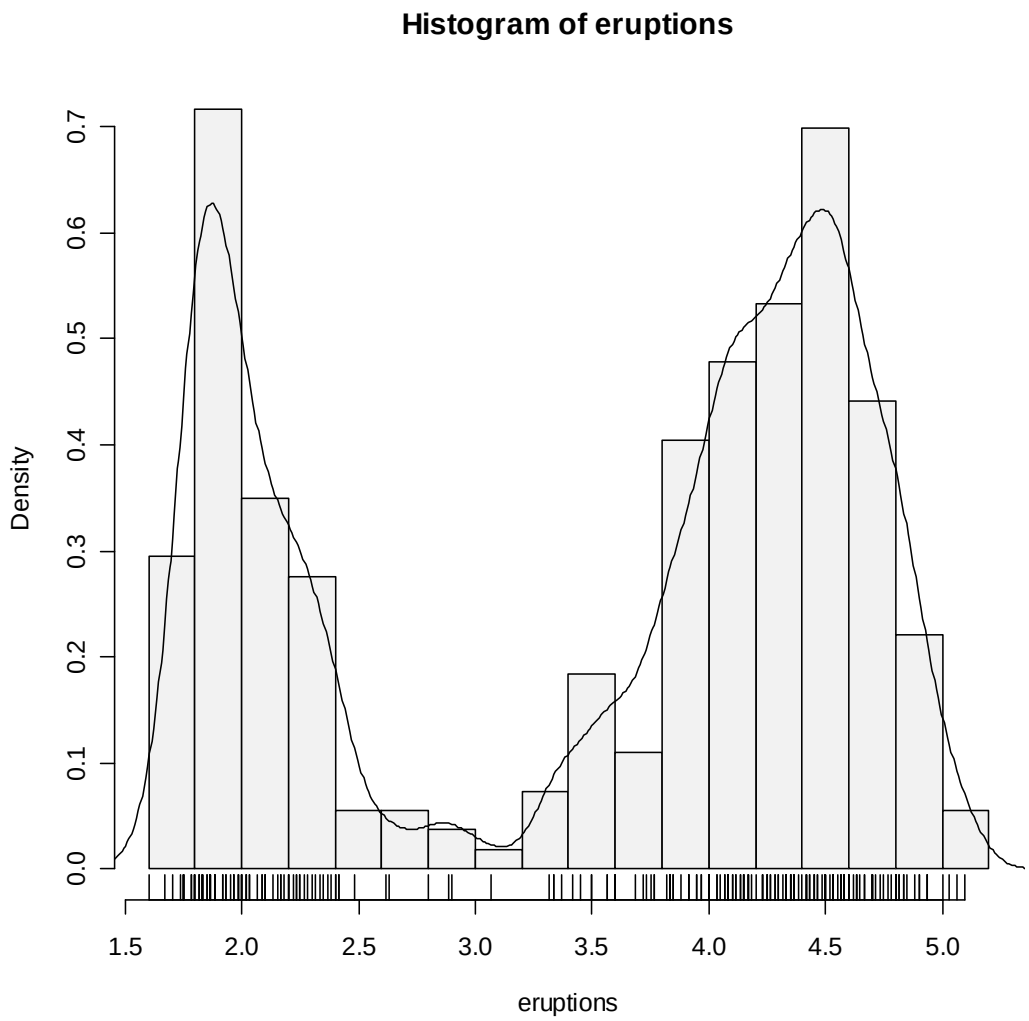
```
myline <- lm(y[,2]~y[,1], data=y[,1:2])
plot(y[,1], y[,2]); text(y[1,1], y[1,2],
      expression(sum(frac(1,sqrt(x^2*pi
      )))), cex=1.3)
abline(myline, lwd=2)
```

Scatterplot Matrices



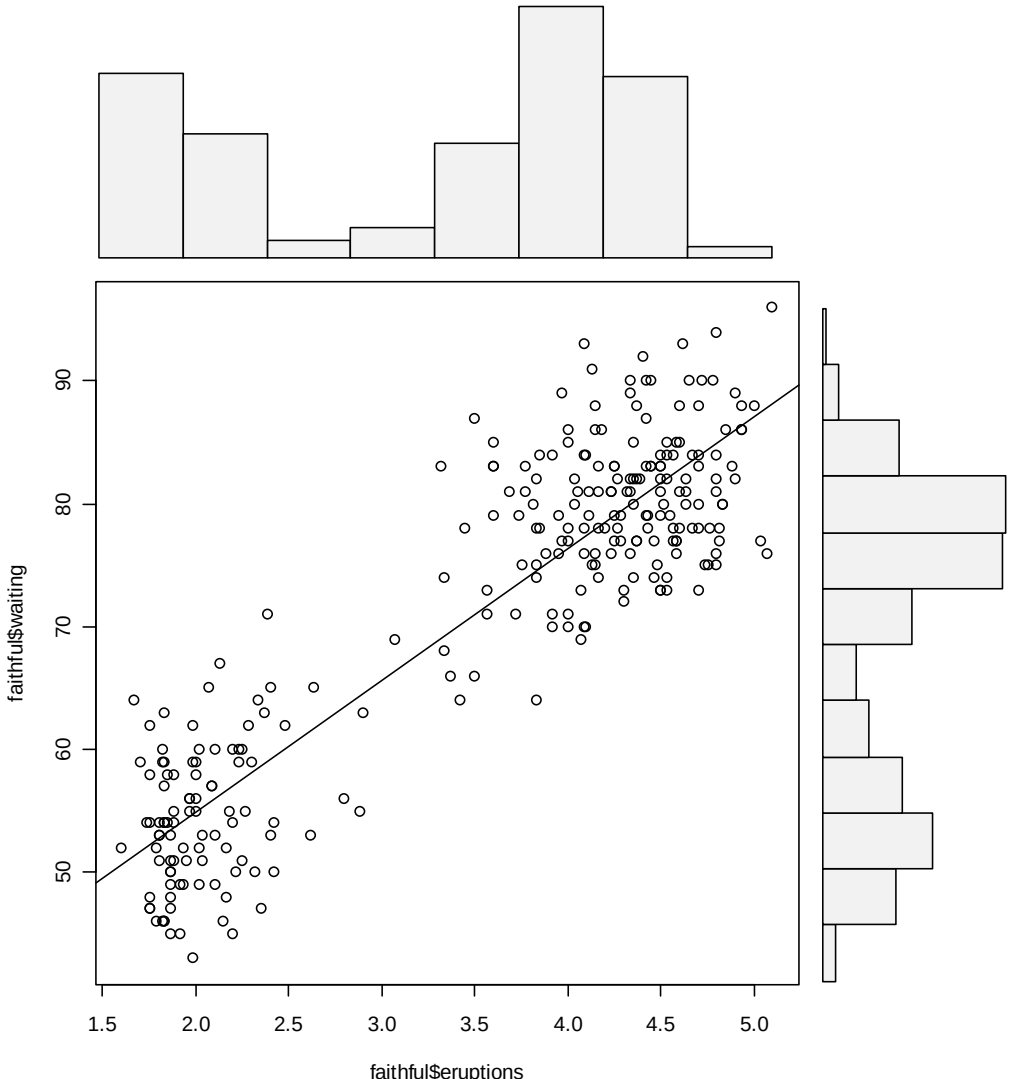

```
panel.cor <- function(x, y, digits=2, prefix="",
        cex.cor){
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <- abs(cor(x, y))
    txt <- format(c(r, 0.123456789),
        digits=digits)[1]
    txt <- paste(prefix, txt, sep="")
    if(missing(cex.cor)) cex <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt, cex = 1.3)
}

xy.panel <- function(x,y){
  points(x,y,pch = 21, bg = c("red", "green3",
        "blue")[unclass(iris$Species)],cex=1.3)
}

pairs(iris[1:4], main = "Anderson's Iris Data -- 3
        species",lower.panel=xy.panel,
        upper.panel=panel.cor)
```

Histogram




```
hist(eruptions, seq(1.6, 5.2, 0.2),
    prob=T, col = gray(0.95))
lines(density(eruptions, bw=0.1))
rug(eruptions, side=1)
```

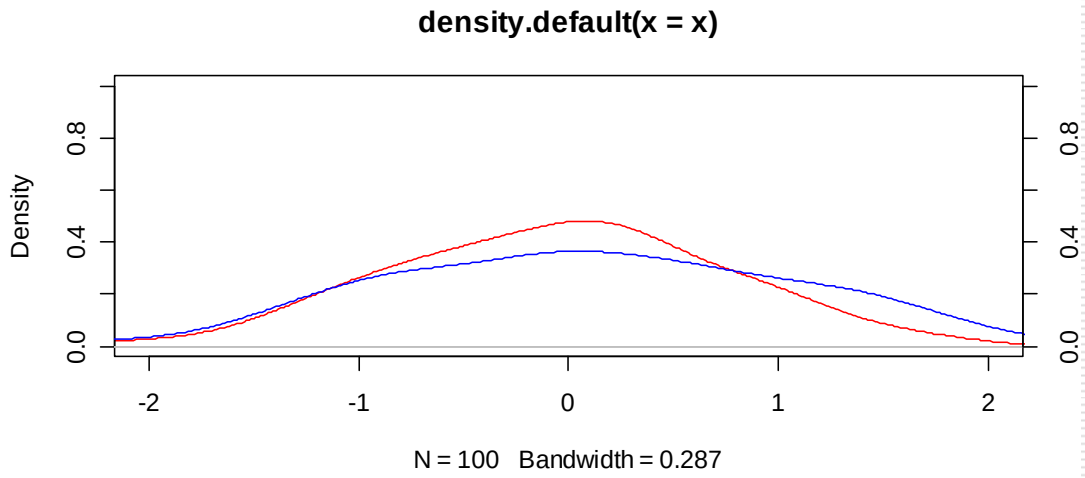
Bivariate Histogram

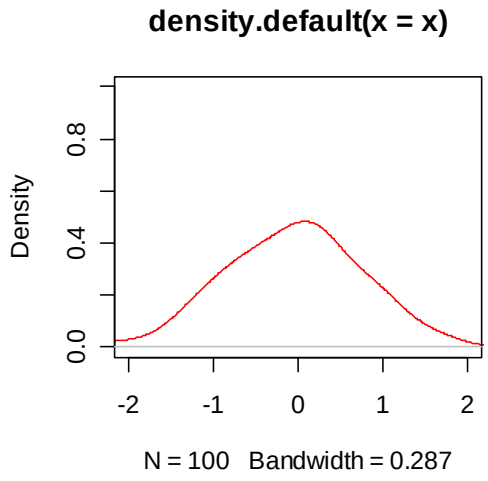


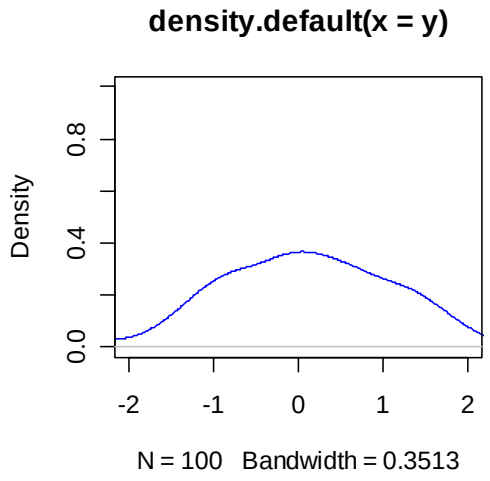

```
library(UsingR)

scatter.with.hist(faithful$eruptions,faithf
      ul$waiting)
```

Density Plots

Overlayed Plots on One Graphic





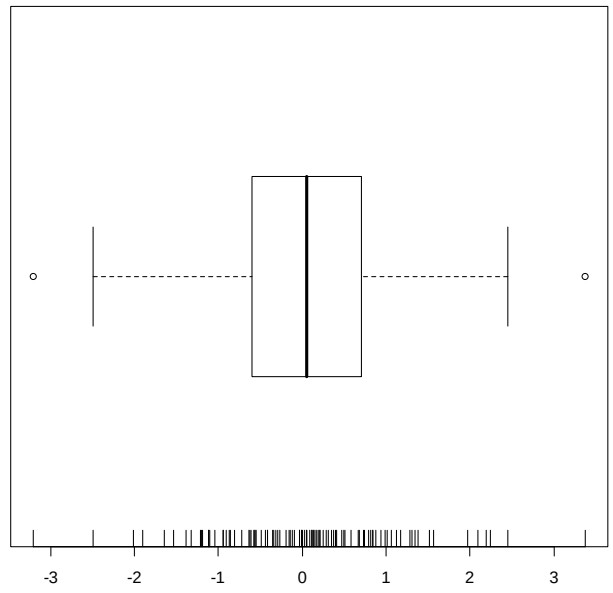


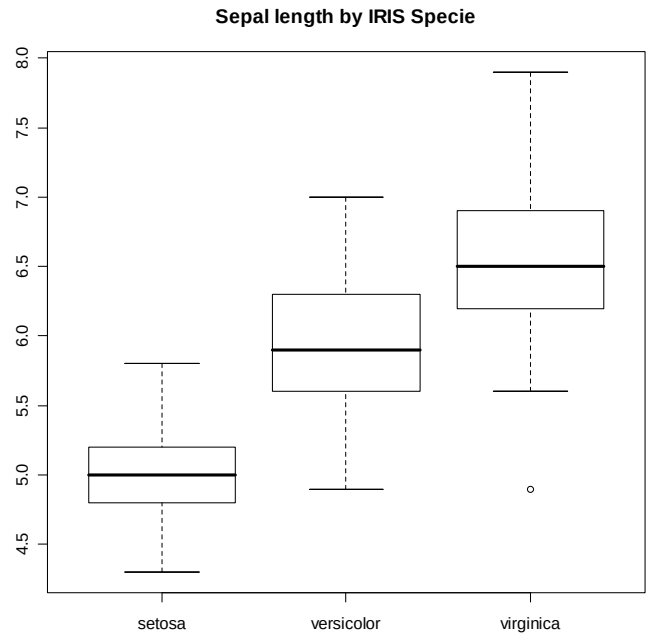

```
x=rnorm(100);y=rnorm(100)
split.screen(c(2,1));
split.screen(c(1,2),screen=2)
screen(3)
plot(density(x), xlim=c(-2,2), ylim=c(0,1),
      col="red");
screen(4)
plot(density(y), xlim=c(-2,2), ylim=c(0,1),
      col="blue");
screen(1)
plot(density(x), xlim=c(-2,2), ylim=c(0,1),
      col="red");
screen(1, new=FALSE);
plot(density(y), xlim=c(-2,2), ylim=c(0,1),
      col="blue", xaxt="n", yaxt="n",
      ylab="", xlab="", main="",bty="n");
axis(4)
close.screen(all = TRUE)
```

Boxplots

- y=rnorm(100)
  boxplot(y) # usual vertical boxplot
  boxplot(y, horizontal=T) # horizontal boxplot ; rug(y, side=1)

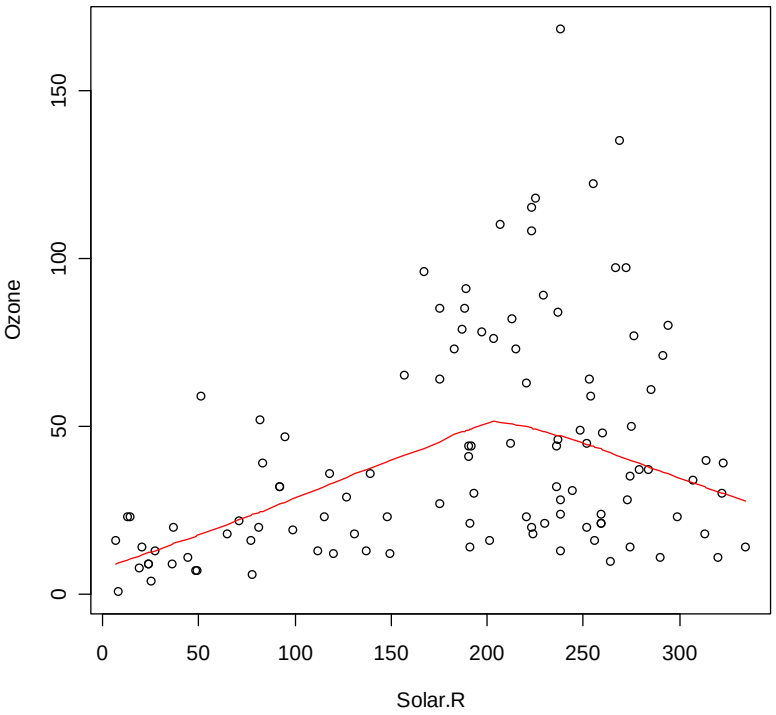
- boxplot(Sepal.Length ~Species,data=iris)
  title("Sepal length by IRIS Species")

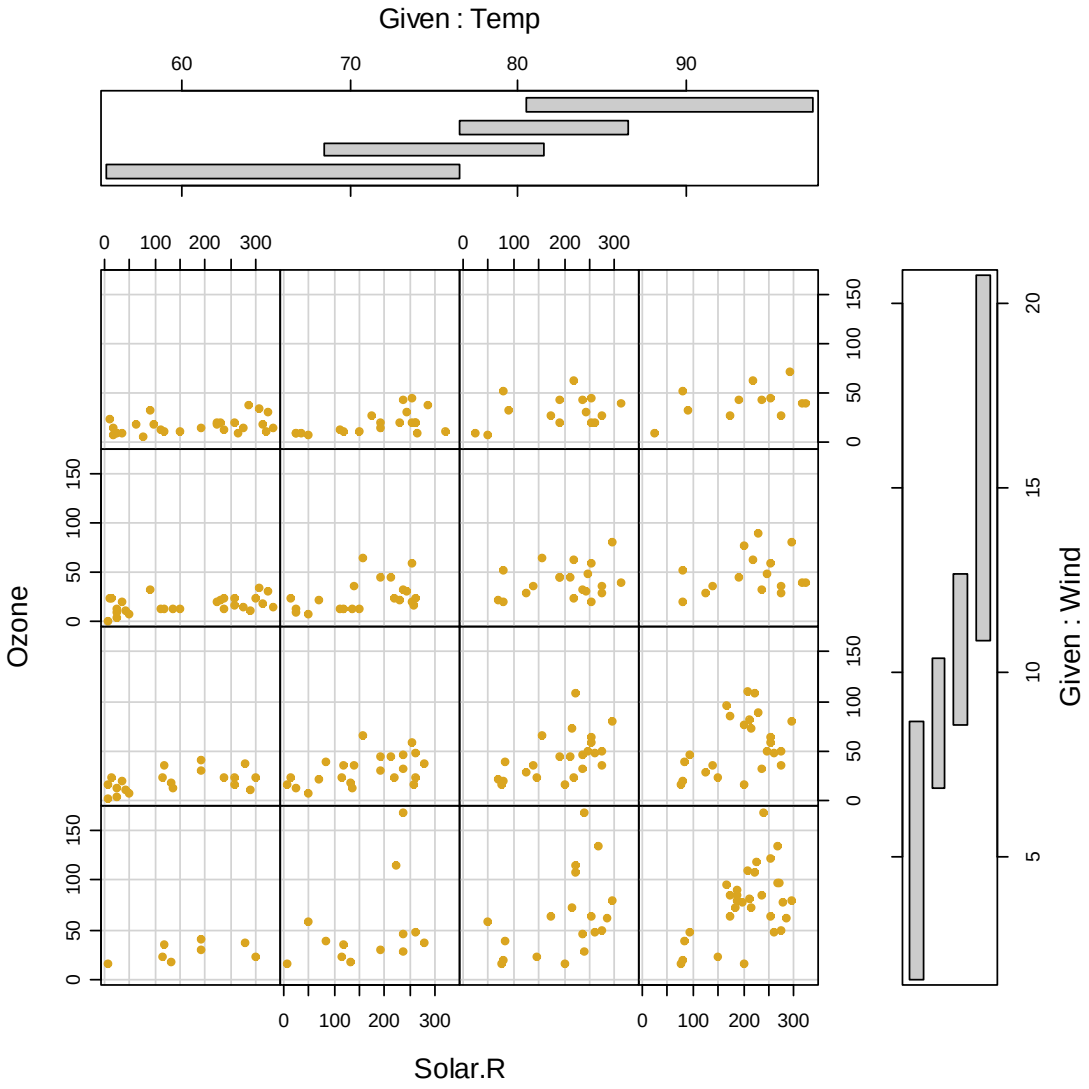




Conditioning Plot

```
data(airquality)
plot(Ozone~Solar.R, data=airquality)
lines(lowess(Ozone~Solar.R,data=airquality),lty=1,col="red")


coplot(Ozone ~ Solar.R | Temp * Wind, number = c(4, 4), data
      = airquality, pch = 21, col = "goldenrod", bg = "goldenrod")
```


Ozone
Solar.R
0
50
100
150
200
250
300

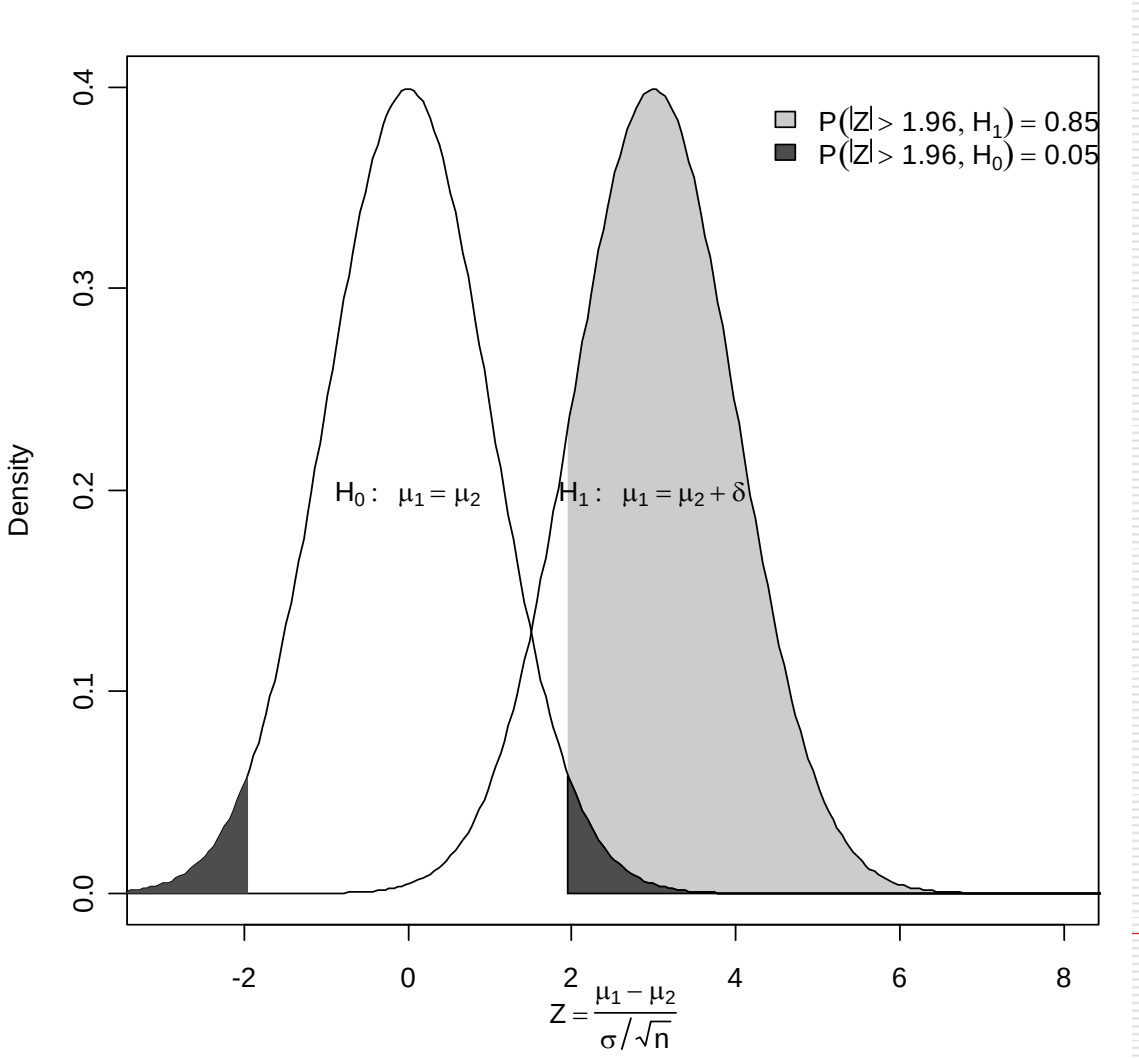

Given : Temp
60
70
80
90
Ozone
Solar.R
Given : Wind
0
50
100
150
200
250
300
5
10
15
20

Mathematical Annotation on Plots

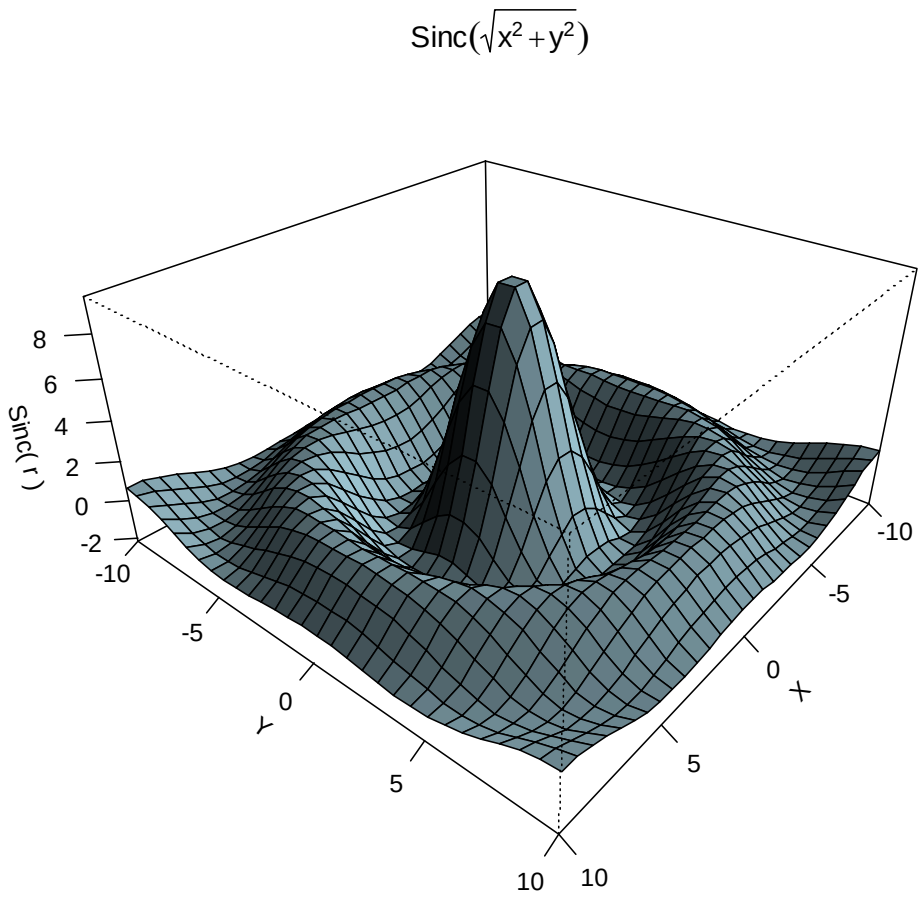
- Syntax is like Latex
- Command: **quote(), expression()**
- Special conventions: subscript([]), superscript(^), alpha, beta, sum, int etc.
- See **help(plotmath)** for details.

Demo

```
x<-seq(-10,10,length=400)
y1<-dnorm(x)
y2<-dnorm(x,m=3)
par(mar=c(5,4,2,1))
plot(x,y2,xlim=c(-3,8),type="n",xlab=quote(Z==frac(mu[1]-mu[2], sigma/sqrt(n))),
      ylab="Density")
polygon(c(1.96,1.96,x[240:400],10),c(0,dnorm(1.96,m=3),y2[240:400],0),col="grey80",
      lty=0)
lines(x,y2); lines(x,y1)
polygon(c(-1.96,-1.96,x[161:1],-10),c(0,dnorm(-1.96,m=0), y1[161:1],0), col="grey30",
      lty=0)
polygon(c(1.96,1.96,x[240:400],10),c(0,dnorm(1.96,m=0),y1[240:400],0),col="grey30")
legend(4.2,.4,fill=c("grey80","grey30"),legend=expression(P(abs(Z)>1.96,H[1])==0.85,
      P(abs(Z)>1.96,H[0])==0.05),bty="n")
text(0,.2,quote(H[0]:~~mu[1]==mu[2]))
text(3,.2,quote(H[1]:~~mu[1]==mu[2]+delta))
```

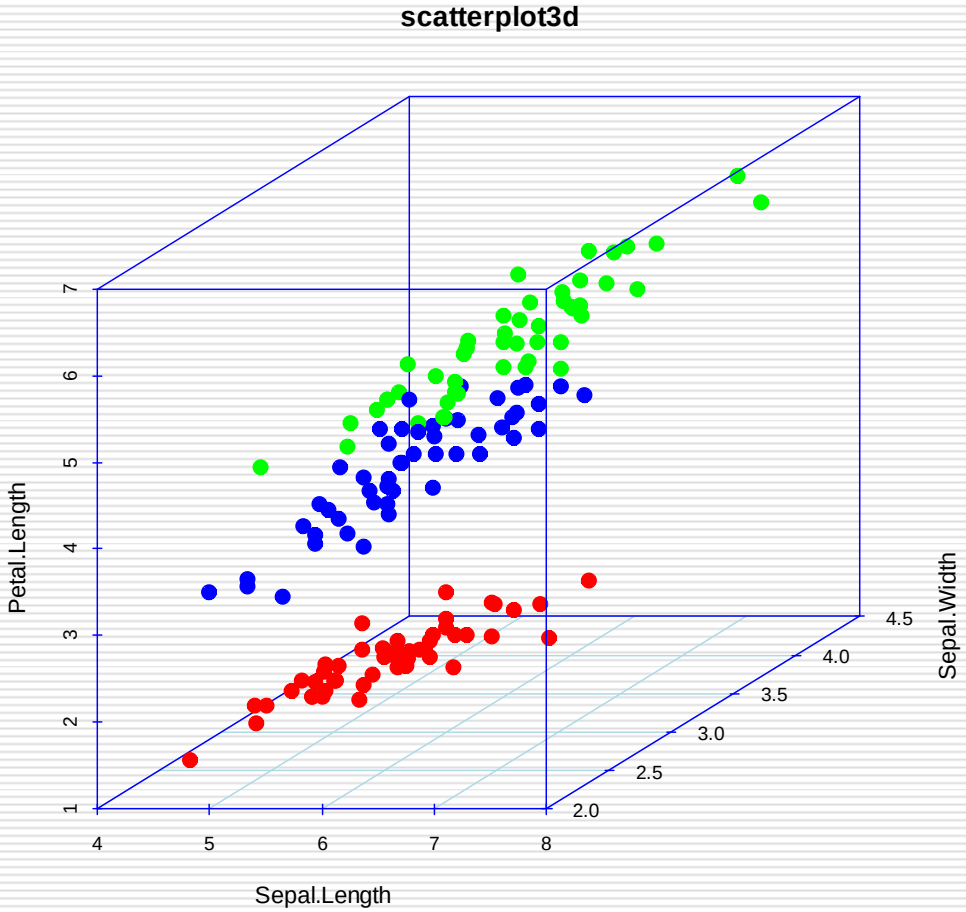

Density
0.0
0.1
0.2
0.3
0.4
-2
0
2
4
6
8
$P(|Z| > 1.96, H_1) = 0.85$
$P(|Z| > 1.96, H_0) = 0.05$
$H_0: \mu_1 = \mu_2$
$H_1: \mu_1 = \mu_2 + \delta$
$Z = \frac{\mu_1 - \mu_2}{\sigma / \sqrt{n}}$

Perspective Plots



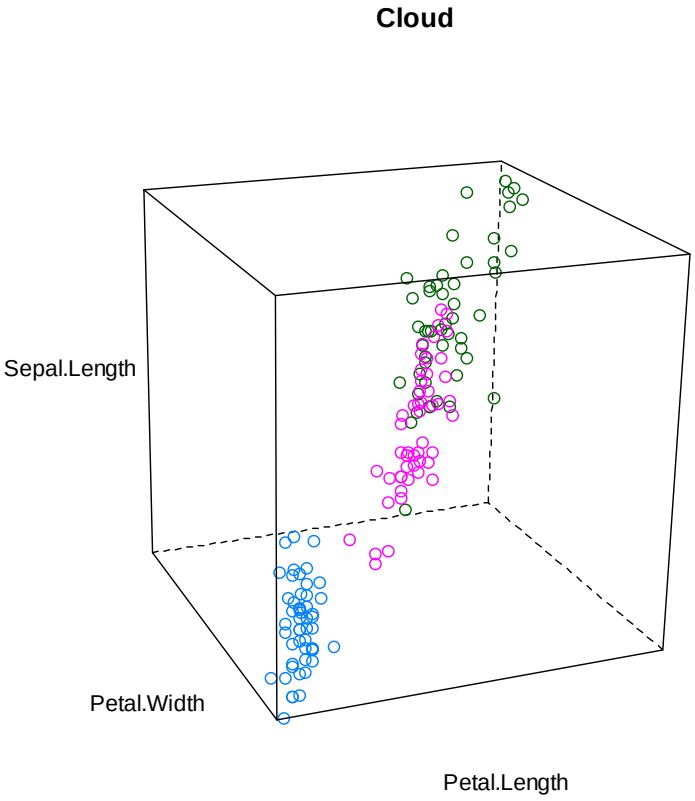

```
persp(x, y, z, theta = 130, phi = 30,
      expand = 0.5, col = "lightblue",
      ltheta = -120, shade = 0.75,
      ticktype = "detailed", xlab = "X",
      ylab = "Y", zlab = "Sinc( r )"
)

title(expression(z=Sinc(sqrt(x^2+y^2
      ))))
```

3-dimensional Scatterplots

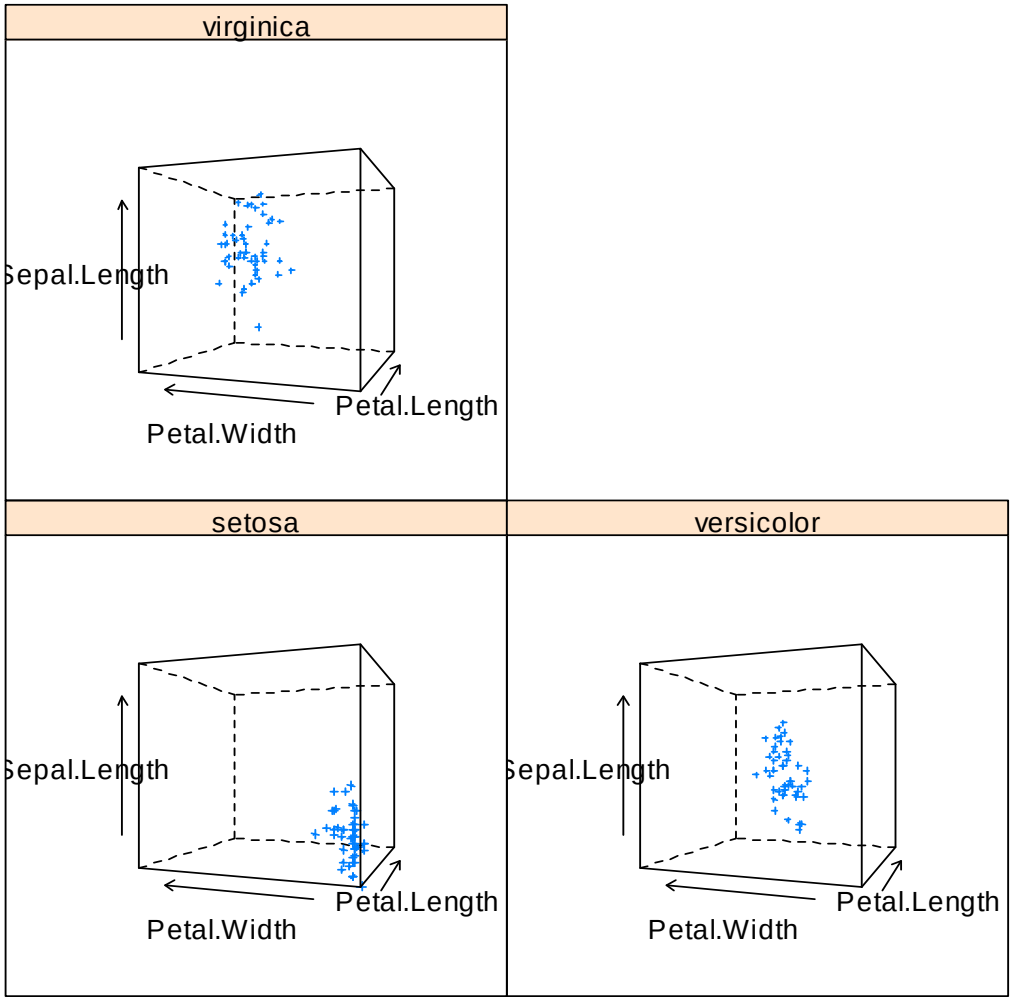



```
library(scatterplot3d)
scatterplot3d(iris[,1:3],color=c("red","bl
    ue","green")[iris$Species],
    col.axis="blue", col.grid
    ="lightblue", main
    ="scatterplot3d",
    pch=20,cex.symbols=2)
```

3-dimensional Plot



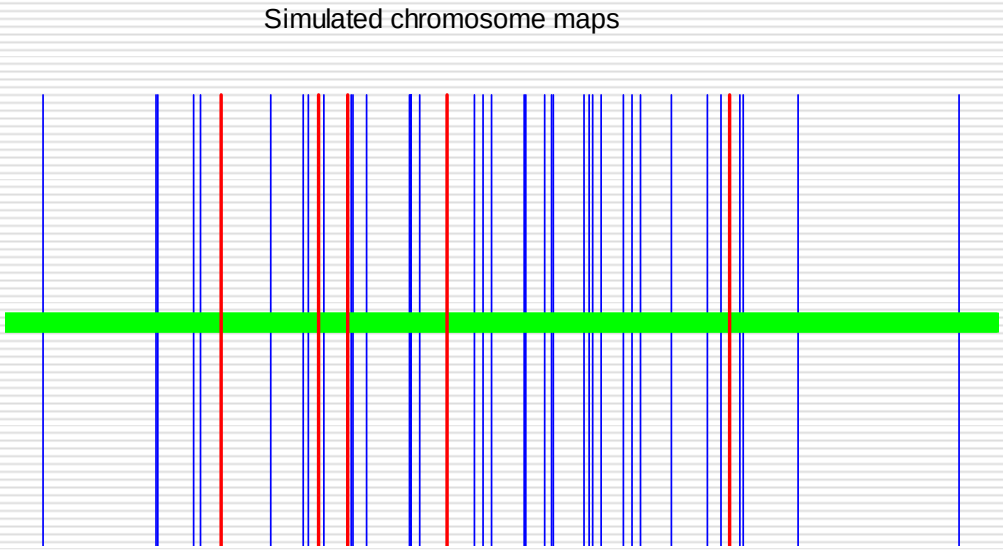

```
library(lattice)
par.set <-list(axis.line = list(col = "transparent"),
clip = list(panel = FALSE))

cloud(Sepal.Length ~ Petal.Length* Petal.Width,
data = iris, cex = 1.0, groups = Species, main =
"Cloud", screen = list(z = 20, x = -70, y = 3),
par.settings = par.set)

cloud(Sepal.Length ~ Petal.Length * Petal.Width
| Species, data = iris, screen = list(x = -90, y = 70),
distance = .4, zoom = .6)
```


virginica
Sepal.Length
Petal.Length
Petal.Width
setosa
Sepal.Length
Petal.Length
Petal.Width
versicolor
Sepal.Length
Petal.Length
Petal.Width

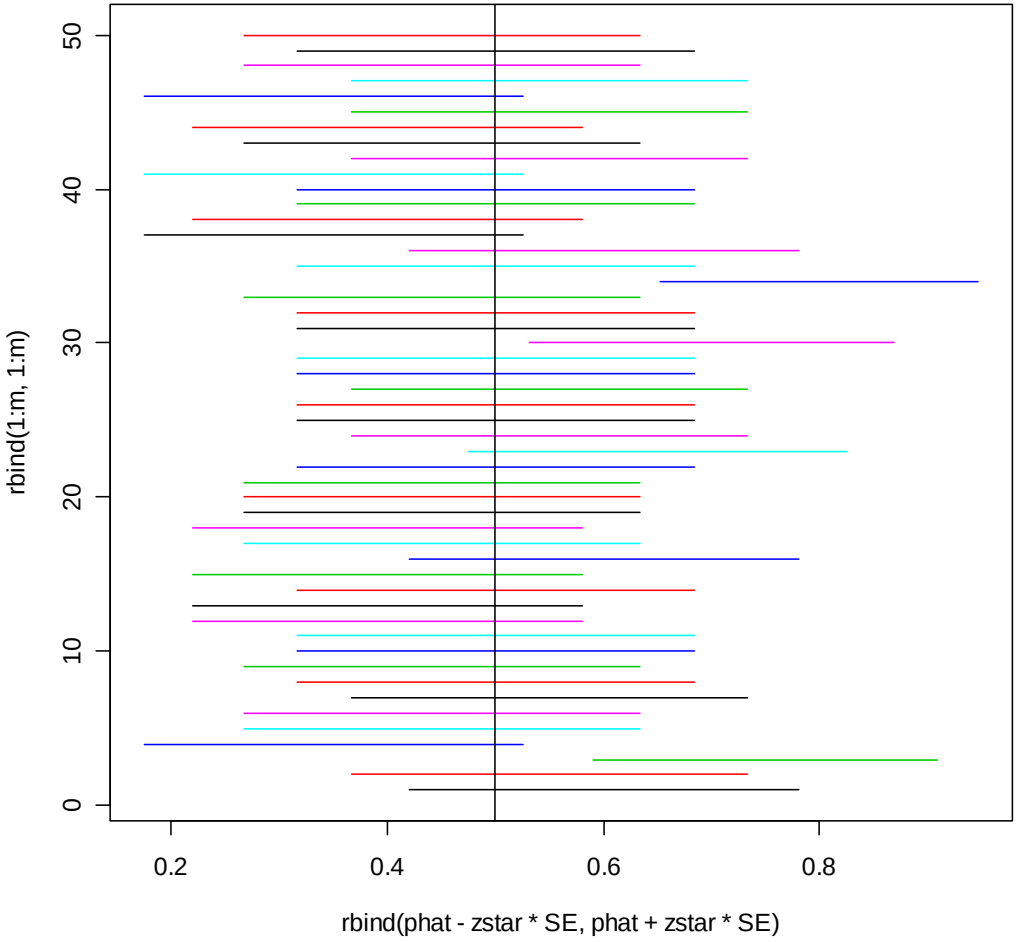
Some Extra Insights

Feature Maps for DNA sequence

```
plot(x <- rnorm(40,2e+07,sd=1e+07), y <- rep(1,times=40), type="h", col="blue", xaxt="n",
      yaxt="n", bty="n",xlab="",ylab="");
abline(h=0.78, col="green", lwd=12);
lines(a <- rnorm(5,2e+07,sd=1e+07), b <- rep(1,times=5), type="h", col="red", lwd=2)
text(locator(1), "Simulated chromosome maps")
```



Matrix Plots



Confidence Interval

```
m = 50; n=20; p = .5; # toss 20 coins 50 times
phat = rbinom(m,n,p)/n # divide by n for proportions
SE = sqrt(phat*(1-phat)/n) # compute SE
alpha = 0.10;zstar = qnorm(1-alpha/2)
matplot(rbind(phat - zstar*SE, phat + zstar*SE),rbind(1:m,1:m),type="l",lty=1)
abline(v=p)
```

Saving Graphics to Files

- **jpeg("test.jpeg"); plot(1:10, 1:10); dev.off()** # After the 'jpeg("test.jpeg")' command all graphs are redirected to the file "test.jpeg" in JPEG format. To export images with the highest quality, the default setting "quality = 75" needs to be changed to 100%. The actual image data are not written to the file until the 'dev.off()' command is executed!
- **pdf("test.pdf"); plot(1:10, 1:10); dev.off()** # Same as above, but for pdf format. The pdf format provides often the best image quality, since it scales to any size without pixelation.
- **png("test.png"); plot(1:10, 1:10); dev.off()** # Same as above, but for png format.
- **postscript("test.ps"); plot(1:10, 1:10); dev.off()** # Same as above, but for PostScript format.

Some Statistical Tests

- Continuous data for testing group effects: **t.test, wilcox.test, oneway.test, kruskal.test, var.test, ks.test**
- Categorical data: **prop.test, chisq.test, fisher.test**
- Correlation test: **cor.test**

Demo

- library(ISwR)
  data(intake); attach(intake)
  t.test(pre, post, paired=TRUE); var.test(pre, post)
  detach()

- data(caesarean) # loads a table
  caesar.shoe; chisq.test(caesar.shoe); fisher.test(caesar.shoe)
  x <- caesar.shoe[1,]
  n <- margin.table(caesar.shoe,2); n
  prop.trend.test(x,n)

Statistical models in R

The operator ~ is used to define a model formula in R. The form, for an ordinary linear model, is

response ~ op_1 term_1 op_2 term_2 op_3 term_3 ...

where

response is a vector or matrix, (or expression evaluating to a vector or matrix) defining the response variable(s).

op_i : an operator, either + or -, implying the inclusion or exclusion of a term in the model, (the first is optional).

term_i : is either a vector or matrix expression, or 1, a factor, or a formula expression consisting of factors, vectors or matrices connected by formula operators.

In all cases each term defines a collection of columns either to be added to or removed from the model matrix. A 1 stands for an intercept column and is by default included in the model matrix unless explicitly removed.

Model Formulas in R

- Suppose y, x, x0, x1, x2, . . . are numeric variables, X is a matrix and A, B, C, . . . are factors.
- y ~ x - 1 #Simple linear regression of y on x through the origin (that is, without an intercept term).
- log(y) ~ x1 + x2 # Multiple regression of the transformed variable, log(y), on x1 and x2 (with an implicit intercept term).
- y ~ poly(x,2) ; y ~ 1 + x + I(x^2)

  #Polynomial regression of y on x of degree 2. The first form uses orthogonal polynomials, and the second uses explicit powers, as basis.
- y ~ X + poly(x,2)

  #Multiple regression y with model matrix consisting of the matrix X as well as polynomial terms in x to degree 2.

Model Formulas in R

- y ~ A #Single classification analysis of variance model of y, with classes determined by A.

- y ~ A + x

  #Single classification analysis of covariance model of y, with classes determined by A, and with covariate x.

- y ~ A*B ; y ~ A + B + A:B ;

  y ~ B %in% A ; y ~ A/B

  #Two factor non-additive model of y on A and B. The first two specify the same crossed classification and the second two specify the same nested classification. In abstract terms all four specify the same model subspace.

Model Formulas in R

- y ~ (A + B + C)^2 ; y ~ A*B*C - A:B:C

  #Three factor experiment but with a model containing main effects and two factor interactions only. Both formulae specify the same model.

- y ~ A * x ; y ~ A/x ; y ~ A/(1 + x) - 1

  #Separate simple linear regression models of y on x within the levels of A, with different codings. The last form produces explicit estimates of as many different intercepts and slopes as there are levels in A.

- y ~ A*B + Error(C)

  #An experiment with two risk factors, A and B, and error strata determined by factor C. For example a split plot experiment, with whole plots (and hence also subplots), determined by factor C.

Linear Model

```
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2,10,20, labels=c("Ctl","Trt"))
weight <- c(ctl, trt)
anova(fm1 <- lm(weight ~ group)) ;
summary(fm2 <- lm(weight ~ group - 1))# omitting intercept
summary(resid(fm1) - resid(fm2))
```

Generic functions for extracting model information

- The value of lm() is a fitted model object; technically a list of results of class "lm". Generic functions that orient themselves to objects of class "lm“ include,

  add1 deviance formula predict step

  alias drop1 kappa print summary

  anova effects labels proj vcov

  coef family plot residuals

- ***anova(object_1, object_2)***

  Compare two or more models and produce an analysis of variance table.

- ***coef(object)***

  Extract the regression coefficient (matrix).

- ***deviance(object)***

  Residual sum of squares, weighted if appropriate.

Generic functions for extracting model information (Cont'd)

- ***formula(object)***

  Extract the model formula.
- ***plot(object)***

  Produce four plots, showing residuals, fitted values and some diagnostics.
- ***predict(object, newdata=data.frame)***

  The data frame supplied must have variables specified with the same labels as the original. The value is a vector or matrix of predicted values corresponding to the determining variable values in data.frame.
- ***residuals(object)***

  Extract the (matrix of) residuals, weighted as appropriate.
- ***step(object)***

  Select a suitable model by adding or dropping terms and preserving hierarchies. The model with the smallest value of AIC (Akaike's An Information Criterion) discovered in the stepwise search is returned.
- ***summary(object)***

  Print a comprehensive summary of the results of the regression analysis.
- ***vcov(object)***

  Returns the variance-covariance matrix of the main parameters of a fitted model object.

Demo

```
data(airquality)
aq <- transform(airquality, Month=factor(Month))
fit.aq <- lm(log(Ozone) ~ Solar.R + Wind +Temp + Month, data=aq)
fit.aq2 <- update(fit.aq, ~ . - Month);  summary(fit.aq)
opar <- par(mfrow = c(2, 2), oma = c(0, 0, 1.1, 0),mar = c(4.1, 4.1, 2.1, 1.1))
plot(fit.aq)
par(opar)
drop1(fit.aq, test="F"); anova(fit.aq, fit.aq2)
```

Demo

```
plot(cars, xlab = "Speed (mph)", ylab = "Stopping distance (ft)", las = 1,
xlim = c(0, 25))
d <- seq(0, 25, len = 200)
 for(degree in 1:4) {
    fm <- lm(dist ~ poly(speed, degree), data = cars)
    assign(paste("cars", degree, sep="."), fm)
    lines(d, predict(fm, data.frame(speed=d)), col = degree)
}
anova(cars.1, cars.2, cars.3, cars.4)
```

Generalized linear models

- There is a response, y, of interest and stimulus variables x_1, x_2, ..., whose values influence the distribution of the response. The stimulus variables influence the distribution of y through a single linear function, only. This linear function is called the linear predictor, and is usually written

  eta = beta_1 x_1 + beta_2 x_2 + ... + beta_p x_p,

  hence x_i has no influence on the distribution of y if and only if beta_i is zero.

- The distribution of y is of the form

  f_Y(y; mu, phi) = exp((A/phi) * (y lambda(mu) - gamma(lambda(mu))) + tau(y, phi)),

  where phi is a scale parameter (possibly known), and is constant for all observations, A represents a prior weight, assumed known but possibly varying with the observations, and mu is the mean of y. So it is assumed that the distribution of y is determined by its mean and possibly a scale parameter as well.

- The mean, mu, is a smooth invertible function of the linear predictor:

  mu = m(eta), eta = m^{-1}(mu) = ell(mu)

  and this inverse function, ell(), is called the link function.

Families and Link Functions on GLM

Family name	Link functions
binomial	logit, probit, log, cloglog
gaussian	identity, log, inverse
Gamma	identity, inverse, log
inverse.gaussian	1/mu^2, identity, inverse, log
poisson	identity, log, sqrt
quasi	logit, probit, cloglog, identity, inverse, log, 1/mu^2, sqrt

GLM Function

- The R function to fit a generalized linear model is glm() which uses the form

  > fitted.model <- glm(formula, family=family.generator, data=data.frame)

- To fit a binomial model using glm() there are three possibilities for the response:

  If the response is a vector it is assumed to hold binary data, and so must be a 0/1 vector.

  If the response is a two-column matrix it is assumed that the first column holds the number of successes for the trial and the second holds the number of failures.

  If the response is a factor, its first level is taken as failure (0) and all other levels as `success' (1).

- kalythos <- data.frame(x = c(20,35,45,55,70), n = rep(50,5), y = c(6,17,26,37,44))

  kalythos$Ymat <- cbind(kalythos$y, kalythos$n - kalythos$y)

  fmp <- glm(Ymat ~ x, family = binomial(link=probit), data = kalythos)

  fml <- glm(Ymat ~ x, family = binomial, data = kalythos) #logit link is the default

  summary(fmp); summary(fml)

Demo

```
no.yes <- c("No","Yes")
smoking <- gl(2, 1, 8, no.yes); obesity <- gl(2, 2, 8, no.yes)
snoring <- gl(2, 4, 8, no.yes)
n.tot <- c(60,17,8,2,187,85,51,23); n.hyp <- c(5,2,1,0,35,13,15,8)
data.frame(smoking,obesity,snoring,n.tot,n.hyp)
hyp.tbl <- cbind(n.hyp,n.tot-n.hyp)
glm.hyp <- glm(hyp.tbl~smoking+obesity+snoring,
      family=binomial("logit"))
summary(glm.hyp); drop1(glm.hyp, test="Chisq")
library(MASS)
confint(glm.hyp)
```

Survival Analysis

- The survival package in R employs the same underlying code as in Splus, written by Terry Therneau at the Mayo Clinic.
- A central concept is **Surv** objects, involving right-censored survival time, and possibly also left-truncation (delayed entry), interval censoring, parametric survival model and more.
- **survfit** for Kaplan-Meier curves
- **survdiff** for logrank test
- **coxph** for fitting the proportional hazards model, using an interface very similar to that of glm

Survival Objects

- Surv(time,status) for simple right-censoring
- Surv(entry,exit,status) for left truncation (late entry) and for time-dependent covariates.
- Surv(entry,exit,status,origin=zero) for when time=0 is different for different people
- Surv(a,b,type="interval2") Interval censoring

Demo

- library(survival); data(ovarian) ## a Mayo ovarian cancer study
  km<- survfit(Surv(futime,fustat)~rx,data=ovarian)
  km; summary(km); plot(km); plot(km[1])

- model1<-coxph(Surv(futime,fustat)~rx+ecog.ps+age,data=ovarian)
  plot(survfit(model1),conf.int=F) ##survival at mean covariates
  summary(model1)
  survdiff(Surv(futime, fustat) ~ rx,data=ovarian)

- leukemia.surv <- survfit(Surv(time, status) ~ x, data = aml)
  plot(leukemia.surv, lty = 2:3,col=1:2,lwd=2)
  legend(100, .9, c("Maintenance", "No Maintenance"), lty = 2:3, col=1:2, lwd=2)
  title("Kaplan-Meier Curves\nfor AML Maintenance Study")

Other Statistics Model in R

- Mixed models. The recommended nlme package provides functions ***lme()*** and ***nlme()*** for linear and non-linear mixed-effects models.
- Local approximating regressions. The ***loess()*** function fits a nonparametric regression by using a locally weighted regression.

  Function loess is in the standard package stats, together with code for projection pursuit regression ***ppr().***
- Robust regression. There are several functions available for fitting regression models in a way resistant to the influence of extreme outliers in the data. Function ***lqs*** in the recommended package MASS provides state-of-art algorithms for highly-resistant fits. Less resistant but statistically more efficient methods are available in packages, for example function rlm in package MASS.
- Additive models. This technique aims to construct a regression function from smooth additive functions of the determining variables, usually one for each determining variable. Functions ***avas*** and ***ace*** in package acepack and functions ***bruto*** and ***mars*** in package mda provide some examples of these techniques in user-contributed packages to R. An extension is Generalized Additive Models, implemented in user-contributed packages gam and mgcv.
- Tree-based models. Tree-based models seek to bifurcate the data, recursively, at critical points of the determining variables in order to partition the data ultimately into groups that are as homogeneous as possible within, and as heterogeneous as possible between. The tree model can be implemented in packages rpart and tree.