Data Analysis and Documentation with R

Athar Ali Khan: Email atharkhan1962@gmail.com

December 9, 2019

Contents

1 Introduction to R Language

R is a language as well as a software. It was introduced by Ross Ihaka and Robert Gentleman in 1996.

1.1 Basics of R

R works like a calculator also. Let us create first chunk.

```
x=5
y=2
z=x+y
z
```

```
## [1] 7
```

1.2 Creating a vector—c()

The function `c()` is used to create a vector. We can do operations on a vector also. We get arithmetic transformation of data vectors also.

```
x=c(2,5,4,3,4)
y=c(3,6,5,6,10)
log(x)
```

```
## [1] 0.6931472 1.6094379 1.3862944 1.0986123 1.3862944
```

```
log(x,base=10)
```

```
## [1] 0.3010300 0.6989700 0.6020600 0.4771213 0.6020600
```

```
log(x,base=2)
```

```
## [1] 1.000000 2.321928 2.000000 1.584963 2.000000
```

```
exp(x) #antilog
```

```
## [1]   7.389056 148.413159  54.598150  20.085537  54.598150
```

```
exp(log(x))# report orginal numbers
```

```
## [1] 2 5 4 3 4
```

1.3 Getting summary features of a data vector

We get summary features of a data vectors, like mean, sd, median and quartiles.

```
mean(x)
```

```
## [1] 3.6
```

```
mean(y)
```

```
## [1] 6
```

```
sd(x)
```

```
## [1] 1.140175
```

```
sd(y)
```

```
## [1] 2.54951
```

```
median(x)
```

```
## [1] 4
```

```
quantile(x)
```

```
##   0%  25%  50%  75% 100% 
##    2    3    4    4    5
```

```
quantile(x,prob=c(.25,.5,.75)) #quartiles
```

```
## 25% 50% 75% 
##   3   4   4
```

```
min(x)
```

```
## [1] 2
```

```
max(x)
```

```
## [1] 5
```

```
#get length of the vector
length(x)
```

```
## [1] 5
```

1.4 Defining a new function—`function()`

In this subsection we define functions for computing standard error of mean `sem()` and coefficient of variation `cv()`. The function dumped using the function `dump()` and it can be sourced using the function `source()`.

```
#Define a function for standard error of mean
sem=function(x) sd(x)/sqrt(length(x))
#dump it as a text file
dump("sem",file="sem.txt")
```

```r
#source it
source("sem.txt")
# Define a function for coefficient of variation
cv=function(x) sd(x)/mean(x)*100
dump("cv",file="cv.txt")
source("cv.txt")
# let us assume height of 6 students in cm is
height=c(168,165,162,170,178,169)
dump("height",file="height.dat")
source("height.dat")
#get their sem and cv
sem(height)
```

```
## [1] 2.216103
```

```r
cv(height)
```

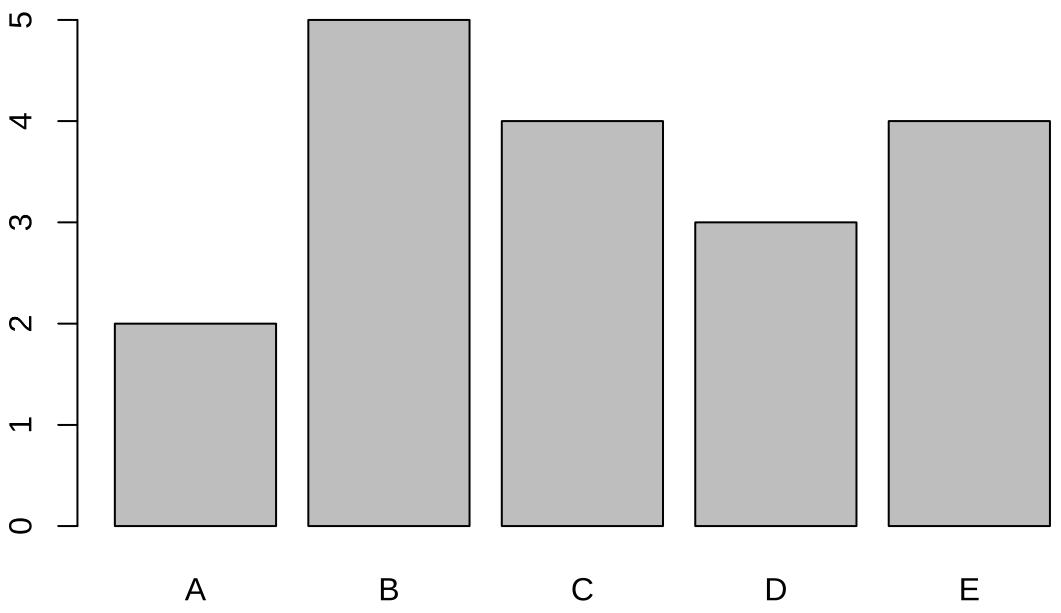
```
## [1] 3.218372
```

1.5 Naming a vector and get barplot—`barplot()`

```r
x=c(2,5,4,3,4)
names(x)=c("A","B","C","D","E")
x
```

```
## A B C D E
## 2 5 4 3 4
```

```r
barplot(x)
```



```
#save it as png file
png("FigBarplot.png")
barplot(x)
dev.off()
```

```
## pdf
##   2
```

2 Factor vector or categorical vector—`factor()`

2.1 Concept of a factor vector

A factor vector is a kind of categorical vector in `R`. It is termed as qualitative variable also. For example, gender is a factor variable, its labels are male and female. Income is also a factor variable and its labels are low, median and high. It is created by using the function `factor`. It may be noted that factor variables are treated **differently** in `R` for analysis and graphics.

```
#create a vector
male=c(0,0,1,0,1,1)
male
```

```
## [1] 0 0 1 0 1 1
```

```
#Change it into a facor vector
Fmale=factor(male,labels=c("female","male"))
Fmale
```

```
## [1] female female male   female male   male
## Levels: female male
```

```
# Create a vector of rating scale
rating=c(3,1,2,2,3,1,1)
rating
```

```
## [1] 3 1 2 2 3 1 1
```

```
Frating=factor(rating,labels=c("good","okay","bad"))
Frating
```

```
## [1] bad  good okay okay bad  good good
## Levels: good okay bad
```

3 Study relation between variables—`cov(x,y)` and `cor(x,y)`

We can study relation between x and y

```
cov(x,y) # covariance between  x and y
```

```
## [1] 1.5
```

```
cor(x,y) # correlation between x and y
```

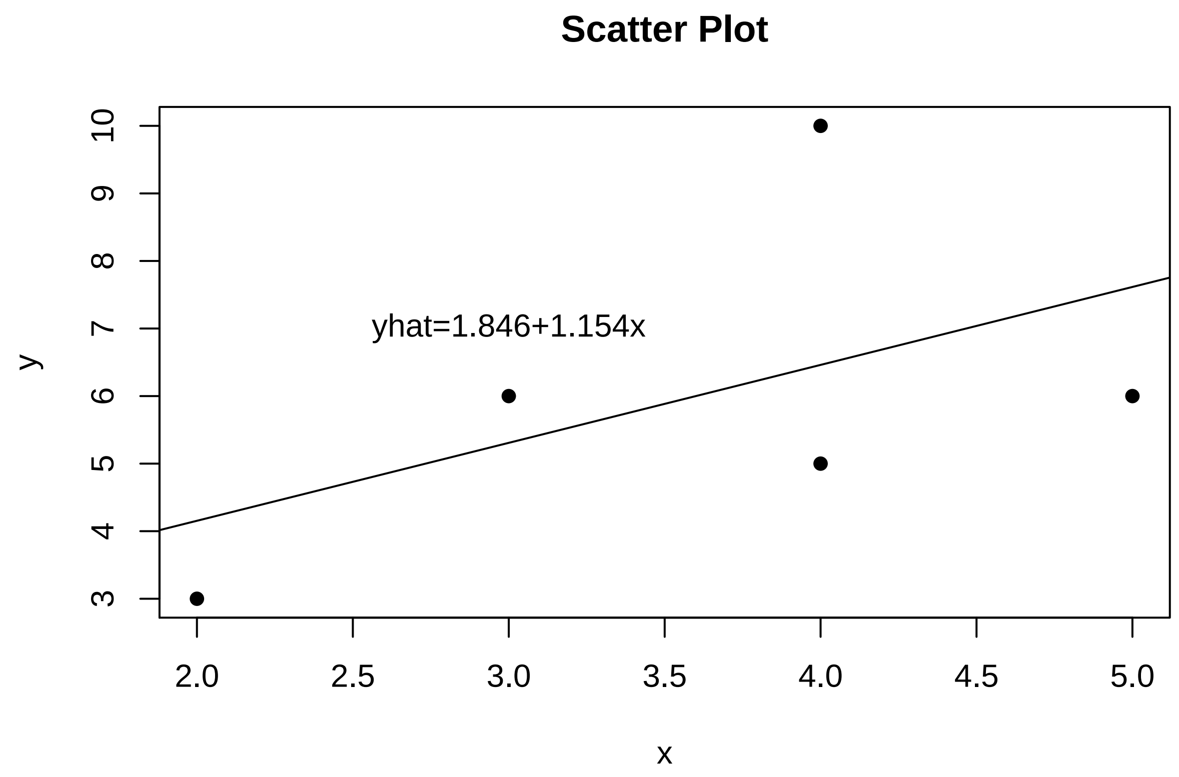
```
## [1] 0.5160157
```

3.1 Get Scatter plot and fitted line—`plot()`

```
x=c(2,5,4,3,4)
y=c(3,6,5,6,10)
#Fit a line
M1=lm(y~x)
coef(M1)
```

```
## (Intercept)           x
##    1.846154    1.153846
```

```
#get aplot
plot(x,y,pch=16,main="Scatter Plot")
# add a fitted line
abline(M1)
text(3,7,"yhat=1.846+1.154x")
```

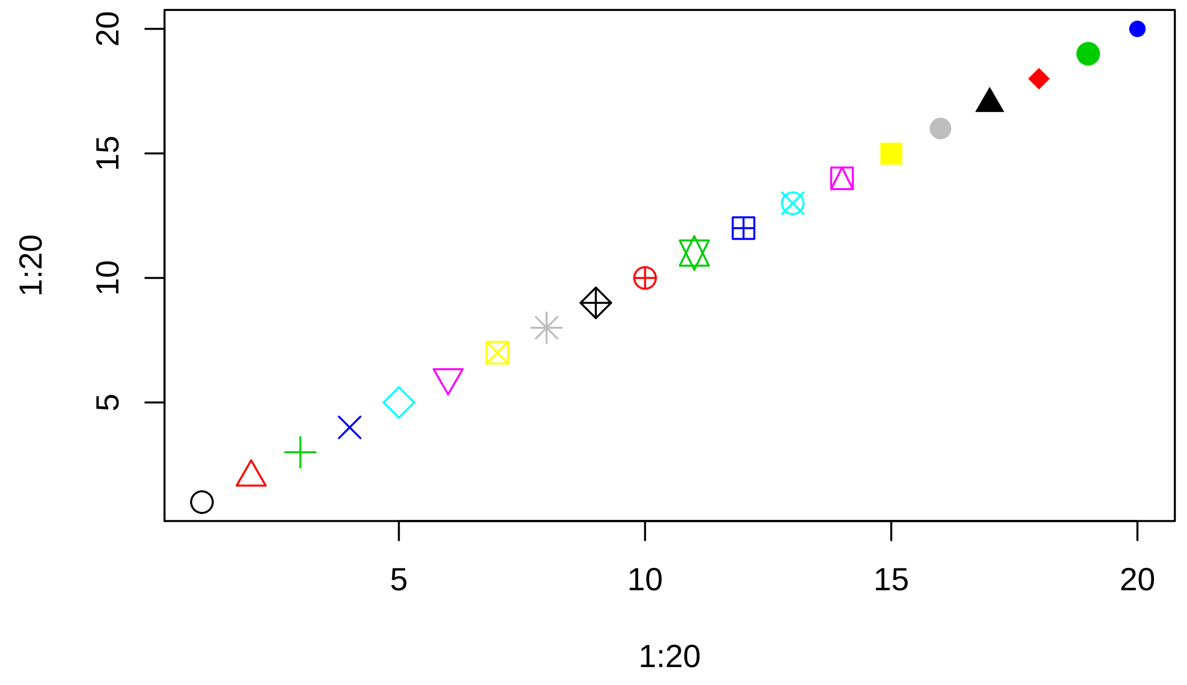


```r
# Save it as png file
png("Fig1.png")
plot(x,y,pch=16,main="Scatter Plot")
# add a fitted line
abline(M1)
text(3,7,"yhat=1.846+1.154x")
dev.off()
```

```
## pdf
##   2
```

3.2 Get plotting characters and colour options

```r
plot(x=1:20,y=1:20,pch=1:20,cex=1.5,col=1:20)
```



```
## save the plot as a pdf file
pdf("FigPlottingCharacters.pdf")
plot(x=1:20,y=1:20,pch=1:20,cex=1.5,col=1:20)
dev.off()
```

```
## pdf
##   2
```

4 Creation of a matrix—matrix() and cbind(), rbind()

The function `cbind()` is meant for column wise binding whereas `rbind()` is meant for row wise binding.

```
x1=c(2,4)
x2=c(5,4)
cmat1=cbind(x1,x2) # columnwise binding
cmat1
```

```
##      x1 x2
## [1,]  2  5
## [2,]  4  4
```

```
rmat1=rbind(x1,x2) # rowwise binding
rmat1
```

```
##    [,1] [,2]
## x1    2    4
## x2    5    4
```

4.1 Operations on matrices—%*%,*,t(), det(), ‘solve()

```
cmat1%*%rmat1 # matrix multiplication
```

```
##      [,1] [,2]
## [1,]   29   28
## [2,]   28   32
```

```
cmat1*rmat1  # element wise multiplication
```

```
##      x1 x2
## [1,]  4 20
## [2,] 20 16
```

```
t(cmat1) # transpose
```

```
##    [,1] [,2]
## x1    2    4
## x2    5    4
```

```
det(cmat1) #determinant
```

```
## [1] -12
```

```
solve(rmat1) # inverse
```

```
##              x1         x2
## [1,] -0.3333333  0.3333333
## [2,]  0.4166667 -0.1666667
```

4.2 Creation of listed data—list()

Listed data combines different data types. For example, it can combine a numeric vector, a character string and a matrix together in one object.

```
x=c(2,4,5)
x2=c("Radha", "Salman")
mat3=matrix(c(1,2,5,10),ncol=2)
LD1=list(x=x,x2=x2,mat3=mat3)
LD1
```

```
## $x
## [1] 2 4 5
##
## $x2
## [1] "Radha"  "Salman"
##
## $mat3
##      [,1] [,2]
## [1,]    1    5
## [2,]    2   10
```

```
LD1$mat3
```

```
##      [,1] [,2]
## [1,]    1    5
## [2,]    2   10
```

```
det(LD1$mat3)
```

```
## [1] 0
```

5 Creation of data frame—`data.frame()` and `read.table()`

Data frame is like a data matrix. It is a kind of generalization of a matrix. In matrix all the elements must be of same type, here in a data frame a numeric column can be combined together with a column of logical or binary, with a column of character string.

```
height=c(168,165,170,162,168) #numeric
weight=c(60,55,68,57,58) #numeric
female=c(0,1,0,0,1) #binary
name=c("Ram","Nazia","Asjad","Ajmal","Radhika") #Character string

#Combnie all the above in a single object, whci is a data frame
DF1=data.frame(height,weight,female,name)
print(DF1)
```

```
##   height weight female    name
## 1    168     60      0     Ram
## 2    165     55      1   Nazia
## 3    170     68      0   Asjad
## 4    162     57      0   Ajmal
## 5    168     58      1 Radhika
```

```
## Save it using write.table()
write.table(DF1, file="DF1.txt")

# Create a data frame from text file DF1.txt using read.table()
DF1=read.table("DF1.txt",header=TRUE)
DF1
```

```
##   height weight female    name
## 1    168     60      0     Ram
## 2    165     55      1   Nazia
## 3    170     68      0   Asjad
## 4    162     57      0   Ajmal
## 5    168     58      1 Radhika
```

The function `read.table()` can be used to read data from a website also, just give the full path of the file, like `https//.../DF1`.

5.1 Having a nice output—`kable()` of package `knitr`

To have a nice output one can use the function `kable()` of `kintr` package.

```
require(knitr)
kable(DF1,caption="Data Frame with kable")
```

Table 1: Data Frame with kable

height	weight	female	name
168	60	0	Ram

height	weight	female	name
165	55	1	Nazia
170	68	0	Asjad
162	57	0	Ajmal
168	58	1	Radhika

5.2 Creation of a data frame from Excel

We can create data into `Excel` and call into `R` as a data frame. Note that data in `Excel` should be entered with first row as name of the variables. After entering data it should be saved as *comma delimited* that is as a `.csv` file in same working folder, and then it could be called by `R` using the function `read.csv`.

```
Weightheight=read.csv("Weightheight.csv")
Weightheight
```

```
##   Hieght Weight Gender
## 1    168     55   male
## 2    170     60 female
## 3    180     68 female
## 4    165     62   male
## 5    168     63   male
```

```
names(Weightheight)=c("Height", "Weight", "Gender")
WH=Weightheight
WH
```

```
##   Height Weight Gender
## 1    168     55   male
## 2    170     60 female
## 3    180     68 female
## 4    165     62   male
## 5    168     63   male
```

```
write.csv(WH, file="WH.csv",row.names=FALSE)
```

6 Graphics with `R`

There are three main graphics in `R`

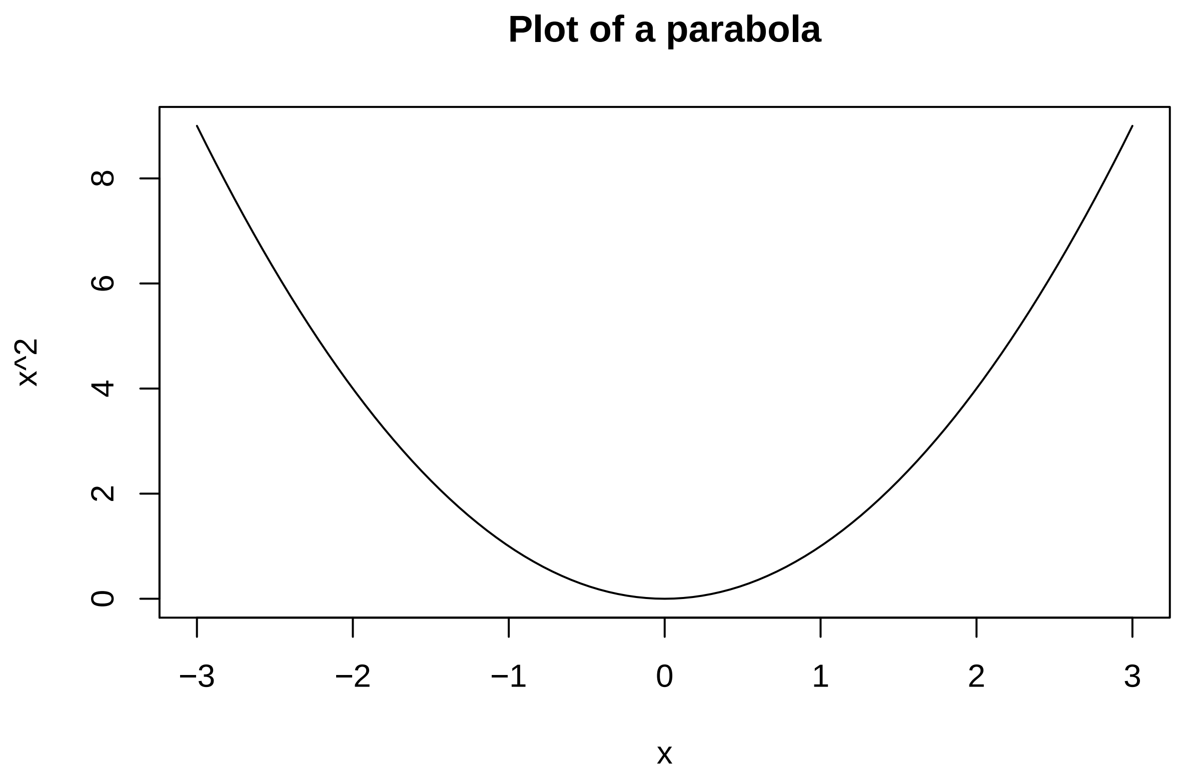
- Base graphics due to Robert Gentleman
- `lattice` package due to Deepayan Sarkar
- `ggplot2` due Wickhem.

We shall discuss only the base graphics in this document.

6.1 The function `curve`

It is a general purpose plotting function. It requires only expression or function to be plotted along with its range on x-axis. A simple example is plotting of a parabola.

```
curve(x^2, from=-3,to=3, main="Plot of a parabola")
```



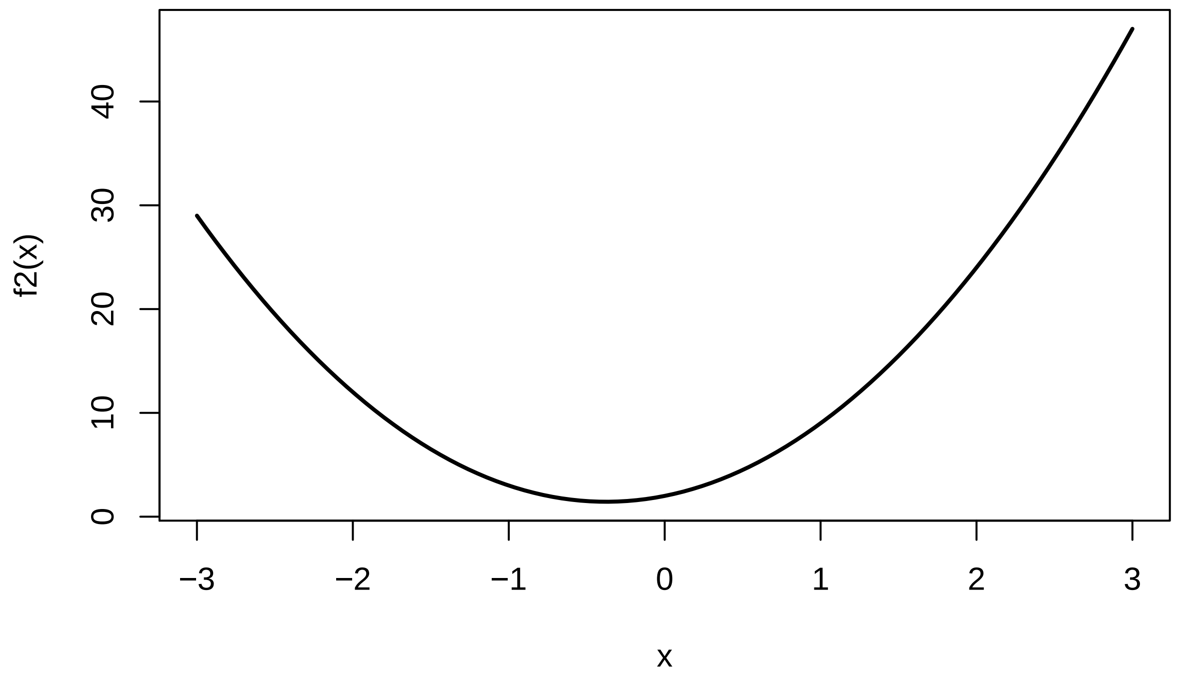
```r
#to save it as png file
png("FigParabola.png")
curve(x^2, from=-3,to=3, main="Plot of a parabola")
dev.off()
```

```
## pdf
##   2
```

6.2 Define a new function and plot it

Suppose you want to plot a function
$$f2(x) = 2 + 3x + 4x^2$$
You can do that by defining the function in `R` which can be plotted using the function `curve` as:

```r
# Define the function
f2=function(x) 2+3*x+4*x^2
#plot it in the range of -3 to to 3
curve(f2(x), from=-3,to=3, lwd=2)
```



6.3 Students' t test—`t.test()`

Students' t test is used for comparing two means. We want to compare the two groups of lazy and sporty patients in terms of their average weights(in kg). Data and its analysis is reported in the following chunk.

```r
wt_lazy=c(76,101,66,72,88,82,79,73,76,85,75,64,76,81,86)
wt_sporty=c(64,65,56,62,59,76,66,82,91,57,92,80,82,67,54)
length(wt_lazy)
```

```
## [1] 15
```

```r
length(wt_sporty)
```

```
## [1] 15
```

```r
# Use t.test() to compare means
t.test(wt_lazy, wt_sporty, var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  wt_lazy and wt_sporty
## t = 2.0969, df = 28, p-value = 0.04516
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   0.1956937 16.7376396
## sample estimates:
```

```
## mean of x mean of y
##  78.66667  70.20000
```

```r
# Same result can be obtained by organizing data in a data frame
weight_ls=data.frame(wtl=wt_lazy, wts=wt_sporty)
head(weight_ls)
```

```
##   wtl wts
## 1  76  64
## 2 101  65
## 3  66  56
## 4  72  62
## 5  88  59
## 6  82  76
```

```r
# Use t test for this data frame
with(weight_ls,t.test(wtl,wts, var.equal = TRUE))
```

```
##
##  Two Sample t-test
##
## data:  wtl and wts
## t = 2.0969, df = 28, p-value = 0.04516
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   0.1956937 16.7376396
## sample estimates:
## mean of x mean of y
##  78.66667  70.20000
```

6.4 Students' t test for stacked data

We stack the data using the command `stack` so that one column represents weight and other column represents grouping factor women.

```r
wtstack=stack(weight_ls)
head(wtstack)
```

```
##   values ind
## 1     76 wtl
## 2    101 wtl
## 3     66 wtl
## 4     72 wtl
## 5     88 wtl
## 6     82 wtl
```

```r
names(wtstack)=c("weight","women")
head(wtstack)
```

```
##   weight women
## 1     76   wtl
## 2    101   wtl
## 3     66   wtl
## 4     72   wtl
## 5     88   wtl
## 6     82   wtl
```

```r
tail(wtstack)
```

```
##    weight women
## 25     57   wts
## 26     92   wts
## 27     80   wts
## 28     82   wts
## 29     67   wts
## 30     54   wts
```

```r
str(wtstack)
```

```
## 'data.frame':    30 obs. of  2 variables:
##  $ weight: num  76 101 66 72 88 82 79 73 76 85 ...
##  $ women : Factor w/ 2 levels "wtl","wts": 1 1 1 1 1 1 1 1 1 1 ...
```

```r
# Save the data
write.table(wtstack, file="wtstack.txt")
# Now use t test
M1=t.test(weight~women,var.equal=TRUE, data=wtstack)
M1
```

```
##
##  Two Sample t-test
##
## data:  weight by women
## t = 2.0969, df = 28, p-value = 0.04516
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   0.1956937 16.7376396
## sample estimates:
## mean in group wtl mean in group wts
##          78.66667          70.20000
```

```r
names(M1)
```

```
## [1] "statistic"   "parameter"   "p.value"     "conf.int"    "estimate"
## [6] "null.value"  "alternative" "method"      "data.name"
```

```r
M1$p.value
```

```
## [1] 0.04516067
```

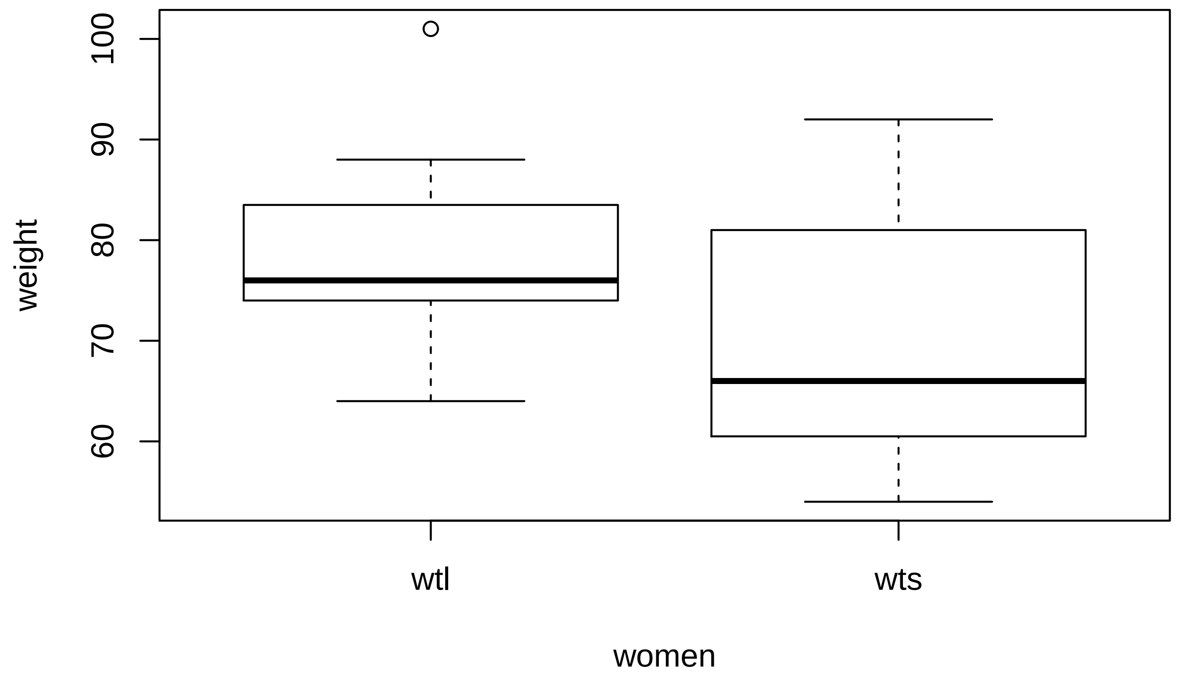
```r
M1$conf.int
```

```
## [1]  0.1956937 16.7376396
## attr(,"conf.level")
## [1] 0.95
```

6.5 Graphic output of Students’t analysis

Box plot is the best summary of this kind of data.

```r
wtstack=read.table("wtstack.txt", header=TRUE)
plot(weight~women, data=wtstack)
```

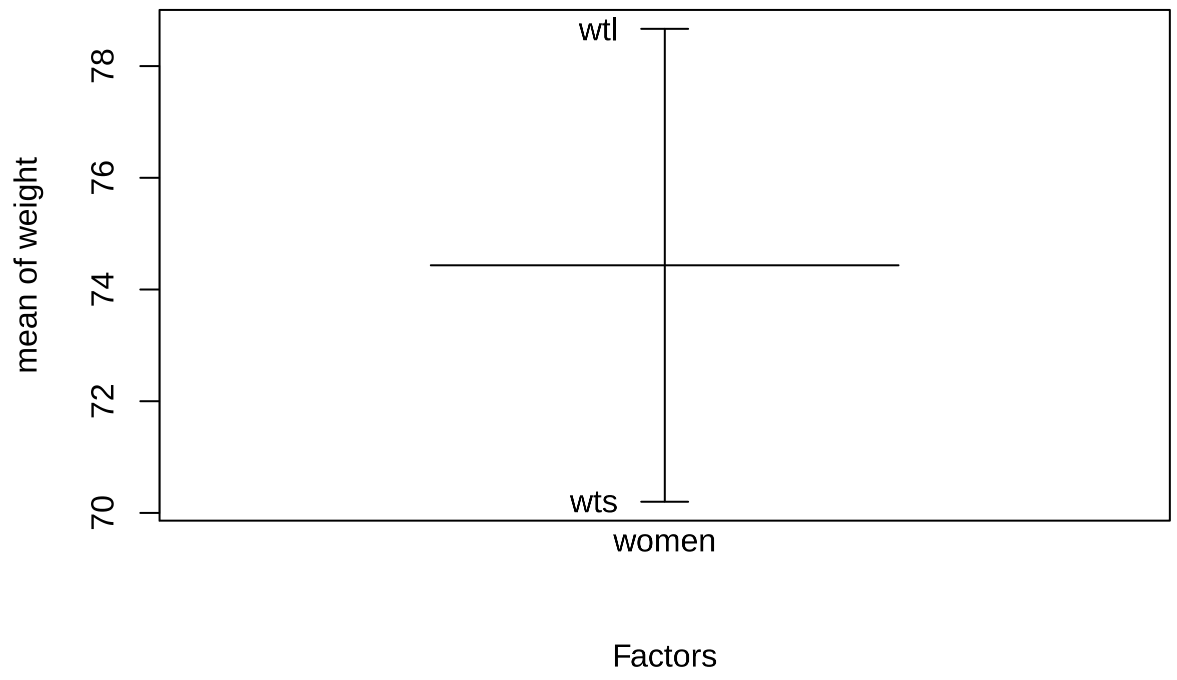


From this plot it is evident that the weight of sporty women is less than that lazy. This difference is statistically significant also. It is evident that both the weights are positively skewed.

6.6 Another plot `plot.design`

In `plot.design` the main argument is the data frame.

```
plot.design(wtstack)
```



In the above plot the big horizontal line represents overall mean, whereas small horizontal lines represent weight of sporty `wts` and lazy `wtl` women. From this plot it is evident that sporty has much less weight than the lazy.

7 Analysis of variance

To compare more than two groups t test can not be used and one has to go for analysis of variance technique.

7.1 Calcium trial data

In a calcium trial data 4 levels of concentrations of calcium were used as A=1, B=5, C=10, D=20; root length(cm) was the response. Create data and analyze it into `R` using analysis of variance technique.

```
Calcium=expand.grid(Replicate=paste0("R",1:5), Concentration=c("A","B","C","D"))
head(Calcium)
```

```
##   Replicate Concentration
## 1        R1             A
## 2        R2             A
## 3        R3             A
## 4        R4             A
## 5        R5             A
## 6        R1             B
```

```
# Add a column of Length in the data frame Calcium using $ operator
Calcium$Length=c(58,52,74,58,79,
                 80,68,72,74,85,
                 49,70,72,74,71,
                 47,49,45,48,38)
head(Calcium)
```

```
##   Replicate Concentration Length
## 1        R1             A     58
## 2        R2             A     52
## 3        R3             A     74
## 4        R4             A     58
## 5        R5             A     79
## 6        R1             B     80
```

```
# save it to a text file "Calcium.txt"
write.table(Calcium, file="Calcium.txt")
```

7.2 Analysis of variance—aov()

```
Calcium=read.table("Calcium.txt", header=TRUE)
names(Calcium)
```

```
## [1] "Replicate"     "Concentration" "Length"
```

```
# Fit a model
calaov=aov(Length~Concentration, data=Calcium)
summary(calaov)
```

```
##               Df Sum Sq Mean Sq F value   Pr(>F)
## Concentration  3   2463   821.0   10.75 0.000409 ***
## Residuals     16   1222    76.4
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From above summary of results it is evident that there is significant difference among the four group of concentrations of Calcium as `p value`, `Pr(>F)=0.000409` which is much less than `0.05`. Now we shall have to look into the pairwise comparisons using `TukeyHSD` command.

7.3 Tukeys Honest Significant Difference Analysis—TukeyHSD()

`TukeyHSD` requires first of its argument, a fitted object with `aov` and second argument `which` which decides for which factor `TukyHSD` will be implemented.

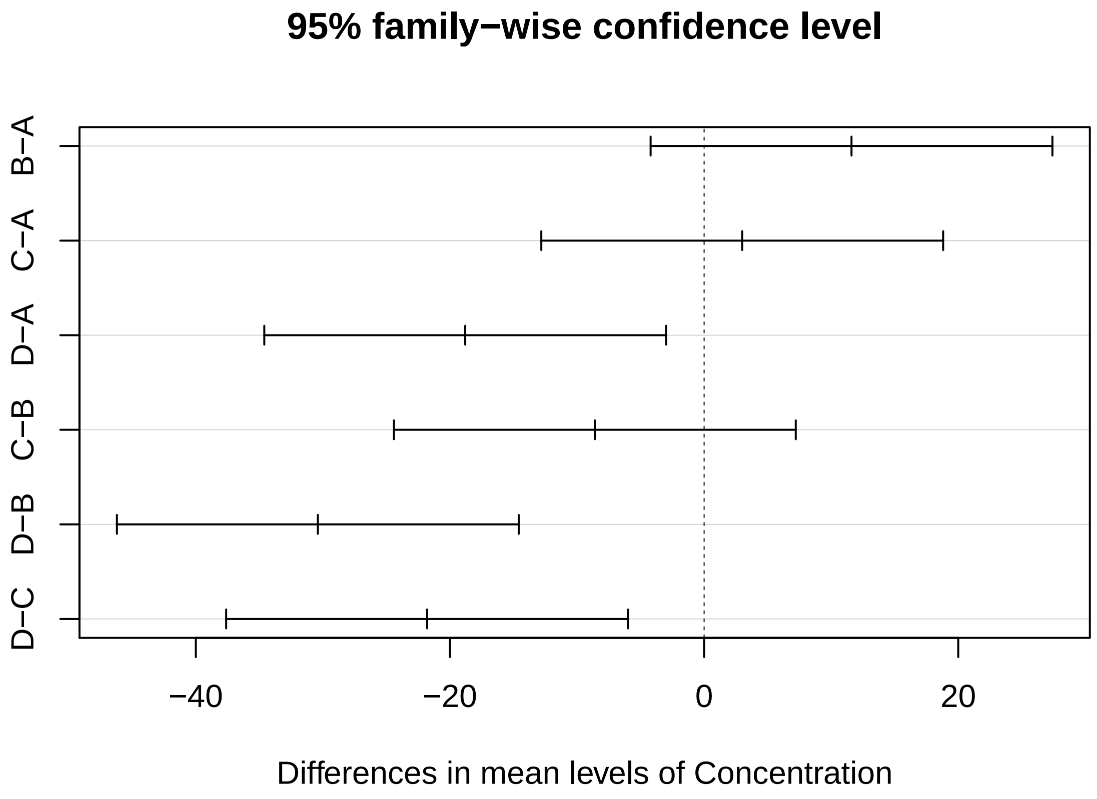
```
out1=TukeyHSD(calaov,which="Concentration")
out1
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Length ~ Concentration, data = Calcium)
##
## $Concentration
##      diff        lwr       upr     p adj
## B-A  11.6  -4.210856  27.410856 0.1954073
## C-A   3.0 -12.810856  18.810856 0.9471343
## D-A -18.8 -34.610856  -2.989144 0.0172308
## C-B  -8.6 -24.410856   7.210856 0.4294993
## D-B -30.4 -46.210856 -14.589144 0.0002558
## D-C -21.8 -37.610856  -5.989144 0.0057328
```

From the `p value` it is evident that `D-A`, `D-B` and `D-C` are statistically significant contrasts. It is also to be noted that corresponding intervals of these contrasts do not contain zero. On the other hand other contrasts are not significant as their corresponding intervals contain zero and `p values` are greater than `0.05`. These facts can be nicely depicted in a graph.

7.4 Plot of `TukeyHSD`

```r
plot(out1)
```



From above plot it is clear that the vertical dashed line at zero is not crossing the intervals of significant contrasts.

8 Bayesian Modeling with rstanarm package

Bayesian statistics is an approach to statistics which formally seeks use of prior information along with the data. These two sources of information are combined together to reach final inference.

```r
require(rstanarm)
# Fit model for wtstack data
```

```
names(wtstack)
M1=stan_glm(weight~women,data=wtstack)
```

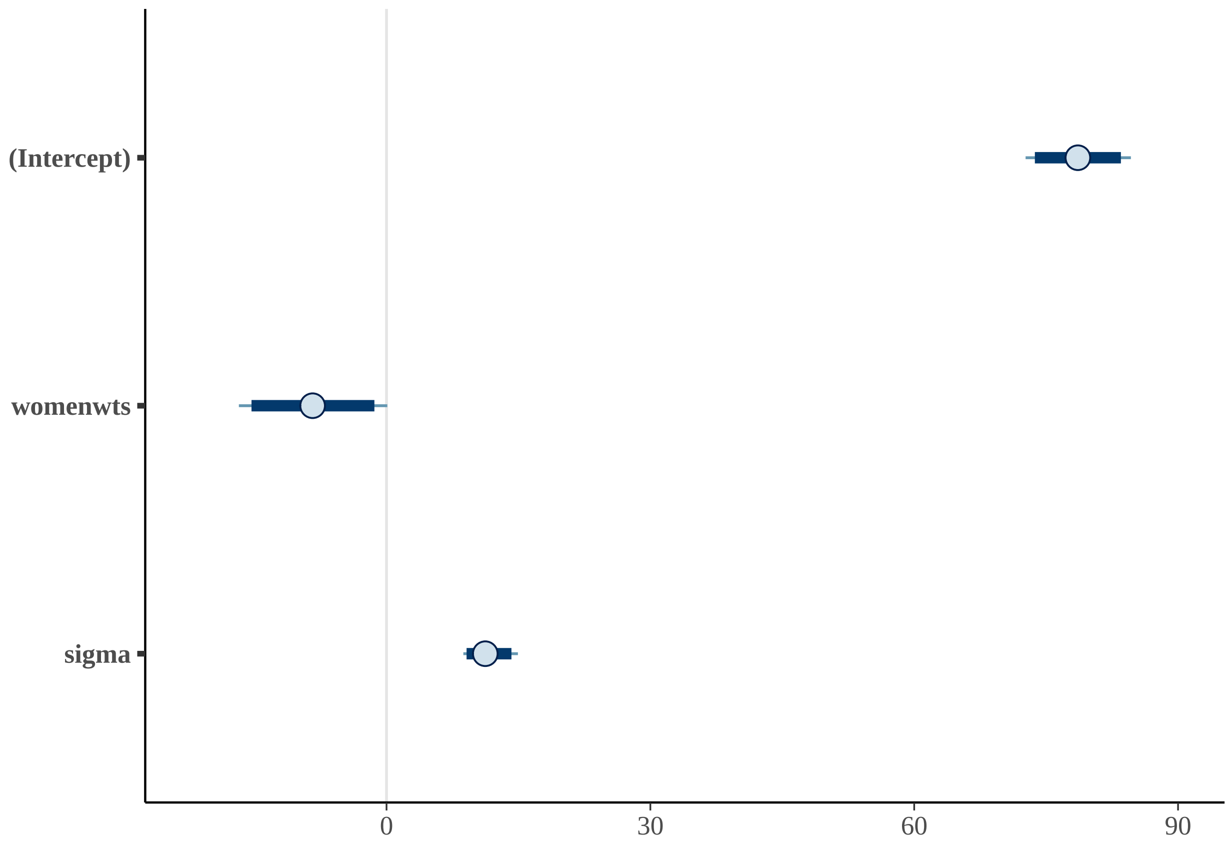
8.1 Find out the results of Bayesian analysis

```
summary(M1)
```

```
##
## Model Info:
##
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      weight ~ women
##  algorithm:    sampling
##  priors:       see help('prior_summary')
##  sample:       4000 (posterior sample size)
##  observations: 30
##  predictors:   2
##
## Estimates:
##                 mean   sd     2.5%   25%    50%    75%    97.5%
## (Intercept)     78.6    3.0   72.7   76.7   78.6   80.5   84.6
## womenwts        -8.4    4.2  -16.8  -11.1   -8.4   -5.6    0.1
## sigma           11.4    1.6    8.7   10.3   11.2   12.3   14.9
## mean_PPD        74.5    3.0   68.4   72.5   74.4   76.5   80.4
## log-posterior -123.1    1.3 -126.4 -123.7 -122.7 -122.1 -121.6
##
## Diagnostics:
##               mcse Rhat n_eff
## (Intercept)   0.1  1.0  3466
## womenwts      0.1  1.0  3361
## sigma         0.0  1.0  3073
## mean_PPD      0.0  1.0  3703
## log-posterior 0.0  1.0  1440
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

8.2 Get a graphic output of it

```
plot(M1,prob=0.95)
```



8.3 Intervals poterior and prective interval—`posterior_interval()`, `predictive_interval`

```
#get the 95% credible interval
posterior_interval(M1,prob=.95)
```

```
##                    2.5%       97.5%
## (Intercept)  72.66393 84.63847613
## womenwts    -16.78716  0.08741156
## sigma         8.73564 14.94098753
```

```
#get predictive interval
predictive_interval(M1,prob=0.95)
```

```
##        2.5%     97.5%
## 1  54.64417 101.78508
## 2  56.15388 101.93248
## 3  55.10358 102.29936
## 4  54.92582 101.50942
## 5  55.00436 102.88626
## 6  55.96165 100.96083
## 7  55.48841 102.10101
## 8  56.53586 102.75270
## 9  55.61600 102.77523
## 10 56.46201 102.20410
## 11 54.48485 102.80010
## 12 56.51881 102.91256
## 13 55.77346 101.62957
## 14 54.66249 100.64067
## 15 55.19031 103.04531
## 16 46.35403  92.90952
## 17 46.68349  93.40784
## 18 47.34728  93.68747
## 19 46.56180  93.95477
## 20 45.95735  93.91393
## 21 47.57239  94.69070
## 22 46.49641  93.41323
## 23 46.74412  92.74656
## 24 46.42565  93.92345
## 25 46.40840  94.33978
## 26 46.35281  94.24984
## 27 47.67113  94.03447
## 28 47.04534  93.66997
## 29 45.57553  93.91675
## 30 46.70664  93.18969
```

8.4 Bayesian analysis of `Calcium` data—one way ANOVA

```
Calcium=read.table("Calcium.txt",header=TRUE)
str(Calcium)
```

```
## 'data.frame':    20 obs. of  3 variables:
##  $ Replicate    : Factor w/ 5 levels "R1","R2","R3",..: 1 2 3 4 5 1 2 3 4 5 ...
##  $ Concentration: Factor w/ 4 levels "A","B","C","D": 1 1 1 1 1 2 2 2 2 2 ...
##  $ Length       : int  58 52 74 58 79 80 68 72 74 85 ...
```

8.4.1 Fit it with—stan_glm()

```
Mcalc=stan_glm(Length~Concentration,data=Calcium)
```

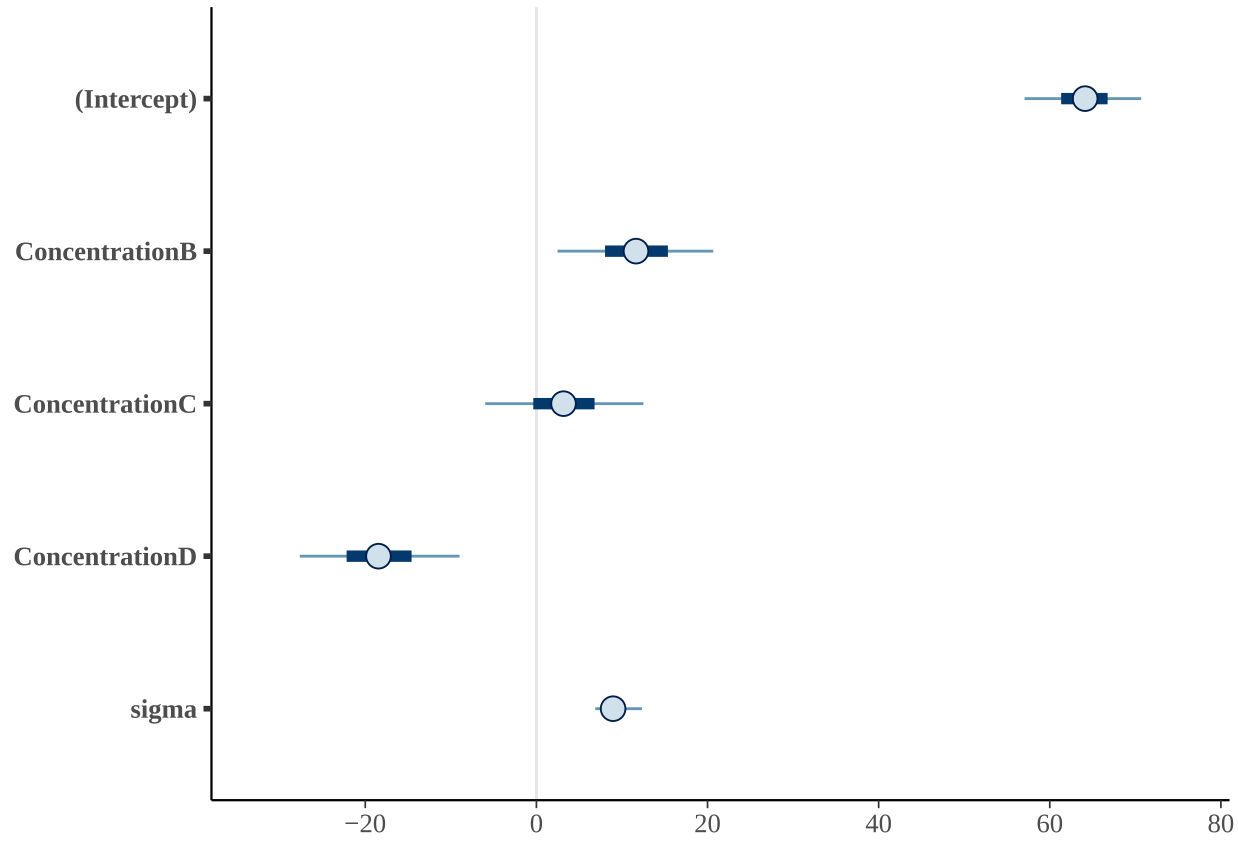
8.4.2 Print summary of results

```
summary(Mcalc)
```

```
##
## Model Info:
##
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      Length ~ Concentration
##  algorithm:    sampling
##  priors:       see help('prior_summary')
##  sample:       4000 (posterior sample size)
##  observations: 20
##  predictors:   4
##
## Estimates:
##                  mean   sd   2.5%   25%   50%   75%   97.5%
## (Intercept)     64.0    4.2  55.7   61.3  64.1  66.8  72.0
## ConcentrationB  11.6    5.6   0.4    8.0  11.6  15.4  22.7
## ConcentrationC   3.2    5.6  -7.9   -0.4   3.2   6.8  14.6
## ConcentrationD -18.4    5.7 -29.4  -22.2 -18.5 -14.6  -7.1
## sigma            9.2    1.7   6.6    8.0   9.0  10.1  13.0
## mean_PPD        63.2    3.0  57.3   61.3  63.2  65.2  68.9
## log-posterior  -82.3    1.7 -86.5  -83.3 -81.9 -81.1 -80.0
##
## Diagnostics:
##                mcse Rhat n_eff
## (Intercept)    0.1  1.0  1845
## ConcentrationB 0.1  1.0  2359
## ConcentrationC 0.1  1.0  2287
## ConcentrationD 0.1  1.0  2201
## sigma          0.0  1.0  2539
## mean_PPD       0.1  1.0  3380
## log-posterior  0.0  1.0  1212
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

8.5 Get graphic output

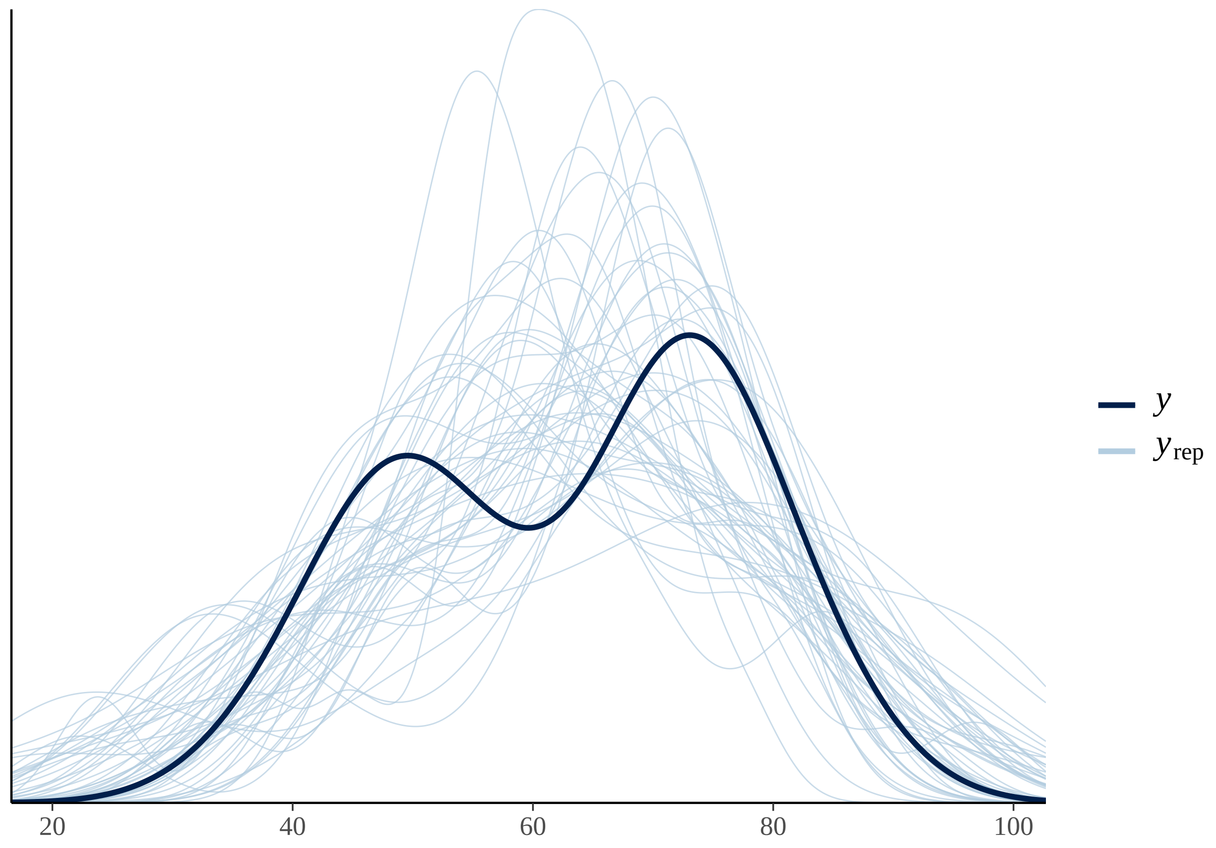
```r
plot(Mcalc)
```



It is evident from this plot that `D-A` and `B-A` are significant, whereas `D-C` is not significant.

8.6 Posterior predictive check—`pp_check()`

```
pp_check(Mcalc)
```



From this output it is evident that there is close agreement between observed and predicted y.

8.7 Bayesian LOOIC—loo package

Leave one out criteria is a criteria to check the predictive ability of the fitted model. It is meant for comparing model on the basis of loo criteria. However, here we are fitting it for a single model. For comparing two or more models, lower value of **looic** indicates better fit of the model.

```
require(loo)
loo(Mcalc)
```

```
## Warning: Found 1 observation(s) with a pareto_k > 0.7. We recommend calling 'loo' again with argumen
```

```
##
## Computed from 4000 by 20 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -75.1 3.4
## p_loo         4.5 1.5
## looic       150.2 6.9
## ------
## Monte Carlo SE of elpd_loo is NA.
##
```

```
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     18    90.0%   541
##  (0.5, 0.7]   (ok)        1     5.0%   670
##    (0.7, 1]   (bad)       1     5.0%   96
##    (1, Inf)   (very bad)  0     0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
```

9 Regression Analysis—lm()

Analysis of `usair` data which is discussed by Wang et al(2018): Bayesian Regression Modeling with INLA. CRC Press. In this data `SO2`content in air is the response and there are 5 predictors

```
require(brinla)
data(usair)
names(usair)
```

```
## [1] "SO2"     "negtemp" "manuf"   "pop"     "wind"    "precip"  "days"
```

```
M1usair=lm(SO2~.,data=usair)
summary(M1usair)$coef
```

```
##                Estimate  Std. Error   t value     Pr(>|t|)
## (Intercept) 111.72848064 47.31810073  2.361221 0.0240867374
## negtemp       1.26794109  0.62117952  2.041183 0.0490557189
## manuf         0.06491817  0.01574825  4.122245 0.0002277862
## pop          -0.03927674  0.01513274 -2.595482 0.0138461970
## wind         -3.18136579  1.81501910 -1.752800 0.0886503978
## precip        0.51235896  0.36275507  1.412410 0.1669175999
## days         -0.05205019  0.16201386 -0.321270 0.7499724652
```

9.1 Stepwise Regression

It may be noted that except `wind`, `precip` and `days` all the predictors are significant. We can use stepwise regression to see which variables are to be dropped

```
library(MASS)
M1step=stepAIC(M1usair,trace=FALSE)
M1step$anova
```

```
## Stepwise Model Path
## Analysis of Deviance Table
##
## Initial Model:
## SO2 ~ negtemp + manuf + pop + wind + precip + days
##
## Final Model:
## SO2 ~ negtemp + manuf + pop + wind + precip
##
##
##     Step Df Deviance Resid. Df Resid. Dev      AIC
## 1                           34   7283.266 226.3703
## 2 - days  1 22.10994        35   7305.376 224.4945
```

Thus only `days` is dropped from the model.

9.2 Factorial experiment with 2 factors in RBD

The data `piarelief` discussed by Wang et al(2018): **Bayesian Regression Modeling with INLA**. CRC Press. is a data in which response is the pain relief score `Relief` and `PainLevel` is the blocking factor, the two treatment factors each with two levels are: `Codeine` and `Accupuncture`. Details are available with INLA book mentioned above.

```
#create factor variables
painrelief=read.table("painrelief.txt",header=TRUE)
painrelief$PainLevel=as.factor(painrelief$PainLevel) #Blocking factor
painrelief$Codeine=as.factor(painrelief$Codeine)
painrelief$Acupuncture=as.factor(painrelief$Acupuncture)
```

9.3 Fit the factorial model

```
M2pain=lm(Relief~PainLevel+Codeine*Acupuncture,data=painrelief)
summary(M2pain)
```

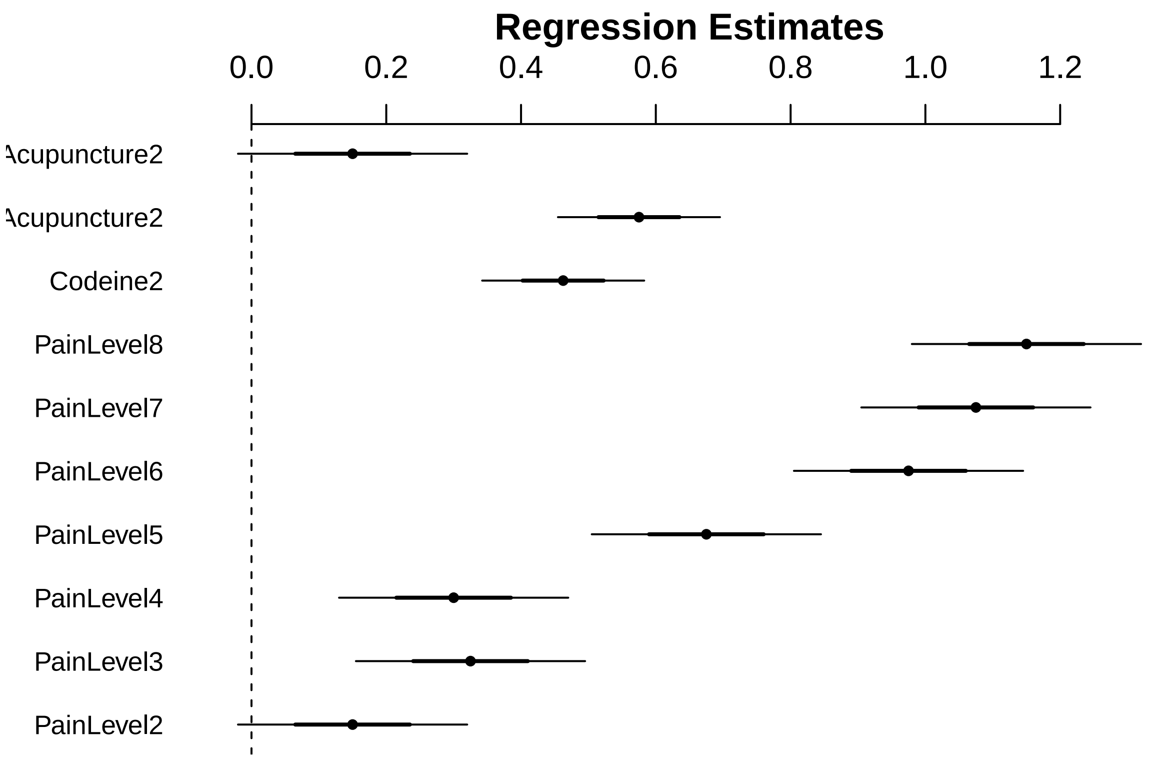
```
##
## Call:
## lm(formula = Relief ~ PainLevel + Codeine * Acupuncture, data = painrelief)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.18125 -0.06250  0.00000  0.04688  0.24375
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)             0.01875    0.07051   0.266 0.792903
## PainLevel2              0.15000    0.08504   1.764 0.092304 .
## PainLevel3              0.32500    0.08504   3.822 0.000994 ***
## PainLevel4              0.30000    0.08504   3.528 0.001998 **
## PainLevel5              0.67500    0.08504   7.937 9.34e-08 ***
## PainLevel6              0.97500    0.08504  11.465 1.68e-10 ***
## PainLevel7              1.07500    0.08504  12.641 2.77e-11 ***
## PainLevel8              1.15000    0.08504  13.523 7.80e-12 ***
## Codeine2                0.46250    0.06013   7.691 1.54e-07 ***
## Acupuncture2            0.57500    0.06013   9.562 4.22e-09 ***
## Codeine2:Acupuncture2   0.15000    0.08504   1.764 0.092304 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1203 on 21 degrees of freedom
## Multiple R-squared:  0.9739, Adjusted R-squared:  0.9615
## F-statistic: 78.37 on 10 and 21 DF,  p-value: 2.521e-14
```

It is evident that blocking is effective. Moreover, two main effects are significant whereas interaction effect is not, as corresponding `p-values` are less than 0.05.

9.4 Graphic summary of above object—`coefplot()` of arm

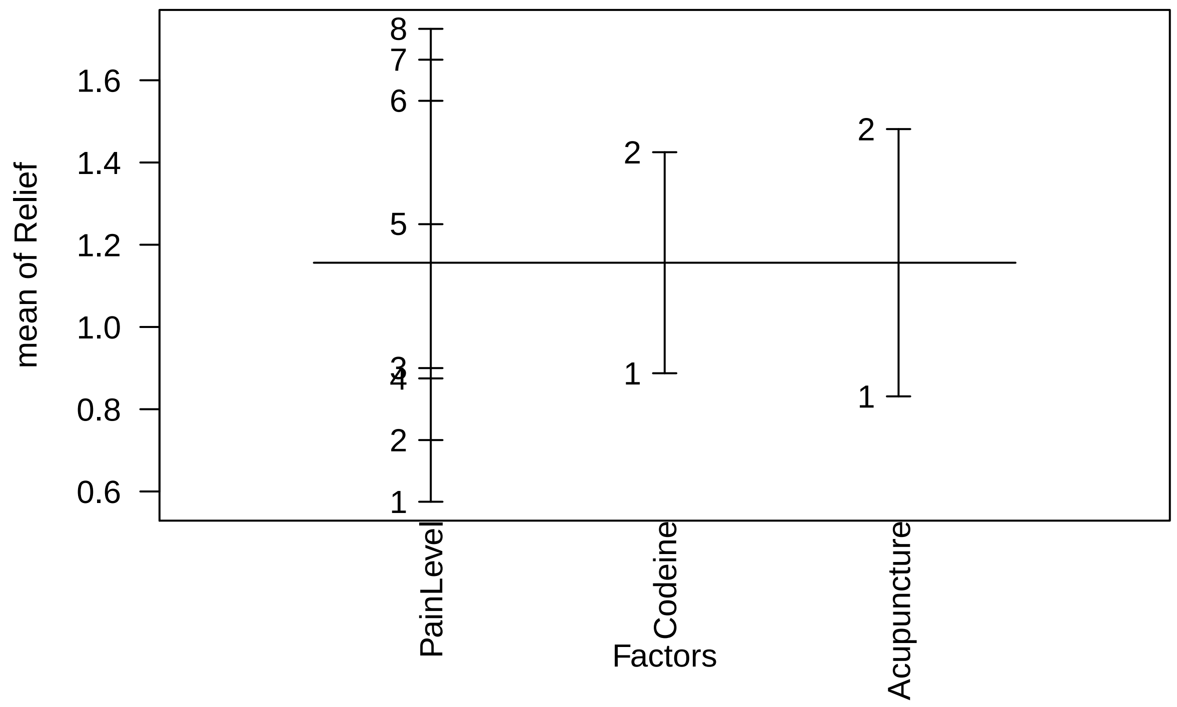
We can have a nice graphic summary of above fitted object using `coefplot()` of **arm** package. This plot provides statistical inference in a graphic output.

```r
require(arm)
coefplot(M2pain)
```



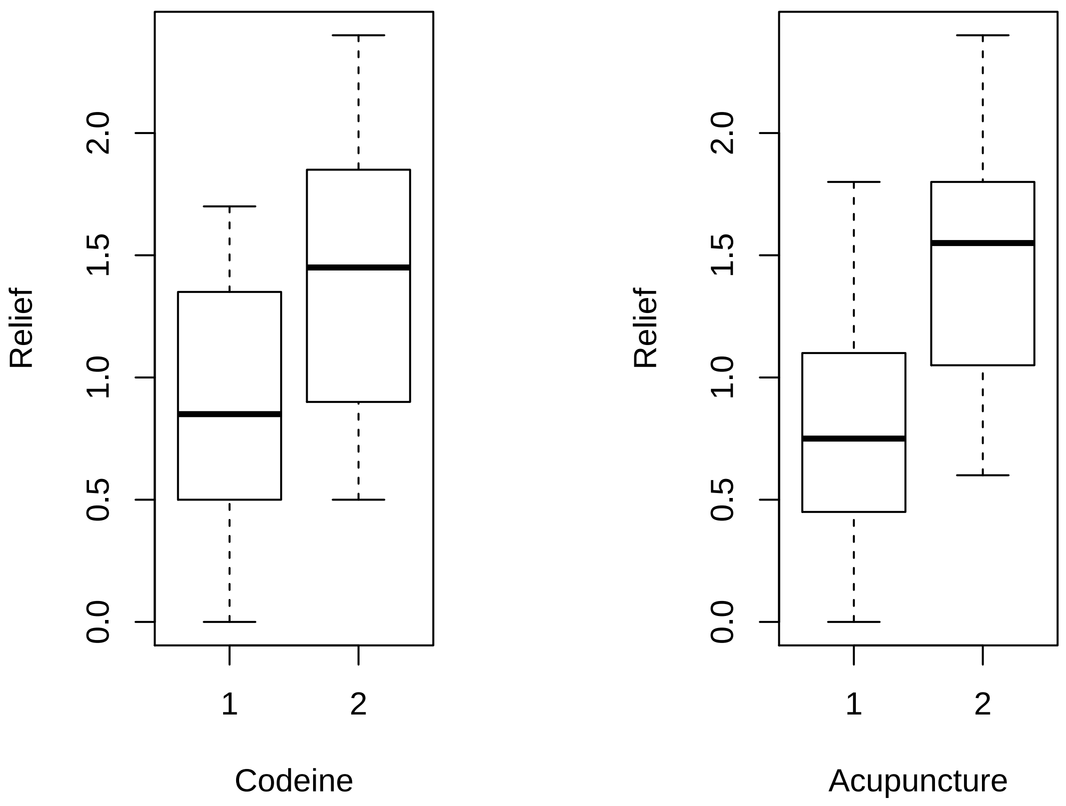
9.5 Look into graphics—`plot.design` and `boxplot`

```r
plot.design(painrelief,las=2)
```



This plot of means is self explanatory. Long horizontal line represents overall mean, whereas small horizontal lines represent means at different levels of the factors. We can have corresponding boxplots to look into variability

```r
opar=par(mfrow=c(1,2), mar=c(4,8,2,1))
plot(Relief~Codeine,data=painrelief)
plot(Relief~Acupuncture,data=painrelief)
```



```
par(opar)
```

10 Ridge Regression for multicollinearity—`lm.ridge()` in MASS

When predictors are highly correlated the problem of multicollinearity arises in which `se(beta)` gets inflated or unstable. We take data `frencheconomy` in which `IMPORT` is the response and predictors are domestic production (`DOPROD`), stock formation (`STOCK`), and domestic consumption (`CONSUM`). Let us see the data

```
data(frencheconomy, package="brinla")
head(frencheconomy,n=4)
```

```
##   YEAR IMPORT DOPROD STOCK CONSUM
## 1   49   15.9  149.3   4.2  108.1
## 2   50   16.4  161.2   4.1  114.8
## 3   51   19.0  171.5   3.1  123.2
## 4   52   19.1  175.5   3.1  126.9
```

```
#get the correlation matrix among predictors third to fifth columns
cor(frencheconomy[,c(3,4,5)])
```

```
##           DOPROD     STOCK    CONSUM
## DOPROD 1.0000000 0.2154456 0.9989330
## STOCK  0.2154456 1.0000000 0.2136902
## CONSUM 0.9989330 0.2136902 1.0000000
```

See the high correlation between `DOPROD` and `CONSUM`. This will cause the problem of multicollinearity. First we scale the data

```
fe.scaled=cbind(frencheconomy[,1:2],scale(frencheconomy[,c(-1,-2)]))
head(fe.scaled)
```

```
##   YEAR IMPORT     DOPROD      STOCK     CONSUM
## 1   49   15.9 -1.3888726  0.2998892 -1.4255958
## 2   50   16.4 -1.2015204  0.2424636 -1.2644647
## 3   51   19.0 -1.0393585 -0.3317923 -1.0624496
## 4   52   19.1 -0.9763830 -0.3317923 -0.9734668
## 5   53   18.8 -0.8929404 -1.4803040 -0.8484099
## 6   54   20.4 -0.7370760 -0.8486226 -0.7137331
```

We fit this data now

```
library(MASS)
ridge2=lm.ridge(IMPORT~DOPROD+STOCK+CONSUM,data=fe.scaled, lambda=seq(0,1,length=100))
ridge2.final=lm.ridge(IMPORT~DOPROD+STOCK+CONSUM,data=fe.scaled,lambda=ridge2$kHKB)
ridge2.final
```

```
##                 DOPROD      STOCK     CONSUM
## 30.0777778  4.9559793  0.7176308  7.1669364
```

10.1 Regression with auroregressive errors

This is case when errors are correlated, example is time series data. In this situation $\sigma^2 I$ is replaced by Σ which leads to generalized least . The function `gls` in the package **nlme** is used to model such data. **First order autoregressive process**, `AR(1)`:

$$e_t = \rho e_{t-1} + \eta_t$$

. Note that $\rho = cor(e_{t-1}, e_t)$ is the error autocorrelation at lag 1 and $\eta_t \sim N(0, \sigma_\eta^2)$.

```
require(brinla)
data(nzunemploy,package="brinla")
nzunemploy$centeredadult=with(nzunemploy,adult-mean(adult)) #centering
require(nlme)
```

```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:lme4':
##
##     lmList
```

```
nzunemploy.gls<-gls(youth~centeredadult*policy,correlation=corAR1(form=~1),data=nzunemploy)
summary(nzunemploy.gls)
```

```
## Generalized least squares fit by REML
##   Model: youth ~ centeredadult * policy
##   Data: nzunemploy
##        AIC      BIC    logLik
##   353.0064 368.5162 -170.5032
##
## Correlation Structure: AR(1)
##  Formula: ~1
##  Parameter estimate(s):
##       Phi
## 0.5012431
##
## Coefficients:
##                               Value Std.Error  t-value p-value
## (Intercept)               16.328637 0.2733468 59.73598       0
## centeredadult              1.522192 0.1274453 11.94389       0
## policyEqual                9.082626 0.8613543 10.54459       0
## centeredadult:policyEqual  2.545011 0.5771780  4.40940       0
##
##  Correlation:
##                           (Intr) cntrdd plcyEq
## centeredadult             -0.020
## policyEqual               -0.318  0.007
## centeredadult:policyEqual -0.067 -0.155  0.583
##
## Standardized residuals:
##         Min          Q1         Med          Q3         Max
## -2.89233359 -0.55460580 -0.02419759  0.55449166  2.29571080
##
## Residual standard error: 1.5052
## Degrees of freedom: 102 total; 98 residual
```

11 Generalized Linear Models

Generalized linear model is an extension of linear model in which error distribution gets extended from Normal to an exponential family of sufficient statistic. This family includes Normal, Gamma, Poisson, Bernoulli, Binomial, Negative Binomial etc. Thus, continous as well as discrete resposes are taken into account. Consequently a rich family of distributions can be fitted for discrete as well as continous data. We shall begin with binary response data.

11.1 Low birth weight data

To illustrate logistic regression model, we are going to analyze a data set that contains information on 189 births to women seen in the obstetric clinic, where data were collected as a part of a larger study at Baystate Medical Center in Springfield. The response variable `LOW` is a binary outcome indicating birth rate less than 2500 grams, which has been of concern to physician for years. The predictors are `AGE`, `LWT`, `RACE`, `SMOKE`, `HT`, `UI`, and `FTV`. The details of the data are available with the data object `lowbwt` with **brinla** package.

```r
require(brinla)
data(lowbwt,package="brinla")
head(lowbwt)
```

```
##   LOW AGE LWT RACE SMOKE HT UI FTV
## 1   1  28 120    3     1  0  1   0
## 2   1  29 130    1     0  0  1   2
## 3   1  34 187    2     1  1  0   0
## 4   1  25 105    3     0  1  0   0
## 5   1  25  85    3     0  0  1   0
## 6   1  27 150    3     0  0  0   0
```

```r
str(lowbwt)
```

```
## 'data.frame':    189 obs. of  8 variables:
##  $ LOW  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ AGE  : int  28 29 34 25 25 27 23 24 24 21 ...
##  $ LWT  : int  120 130 187 105 85 150 97 128 132 165 ...
##  $ RACE : Factor w/ 3 levels "1","2","3": 3 1 2 3 3 3 3 2 3 1 ...
##  $ SMOKE: Factor w/ 2 levels "0","1": 2 1 2 1 1 1 1 1 1 2 ...
##  $ HT   : Factor w/ 2 levels "0","1": 1 1 2 2 1 1 1 1 2 2 ...
##  $ UI   : Factor w/ 2 levels "0","1": 2 2 1 1 2 1 2 1 1 1 ...
##  $ FTV  : int  0 2 0 0 0 0 1 1 0 1 ...
```

```r
lowbwt$RACE=factor(lowbwt$RACE,labels = c("white","black","other"))
lowbwt$SMOKE=factor(lowbwt$SMOKE,labels=c("no","yes"))
lowbwt$HT=factor(lowbwt$HT,labels=c("no","yes"))
lowbwt$UI=factor(lowbwt$UI,labels=c("no","yes"))
str(lowbwt)
```

```
## 'data.frame':    189 obs. of  8 variables:
##  $ LOW  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ AGE  : int  28 29 34 25 25 27 23 24 24 21 ...
##  $ LWT  : int  120 130 187 105 85 150 97 128 132 165 ...
##  $ RACE : Factor w/ 3 levels "white","black",..: 3 1 2 3 3 3 3 2 3 1 ...
##  $ SMOKE: Factor w/ 2 levels "no","yes": 2 1 2 1 1 1 1 1 1 2 ...
##  $ HT   : Factor w/ 2 levels "no","yes": 1 1 2 2 1 1 1 1 2 2 ...
##  $ UI   : Factor w/ 2 levels "no","yes": 2 2 1 1 2 1 2 1 1 1 ...
##  $ FTV  : int  0 2 0 0 0 0 1 1 0 1 ...
```

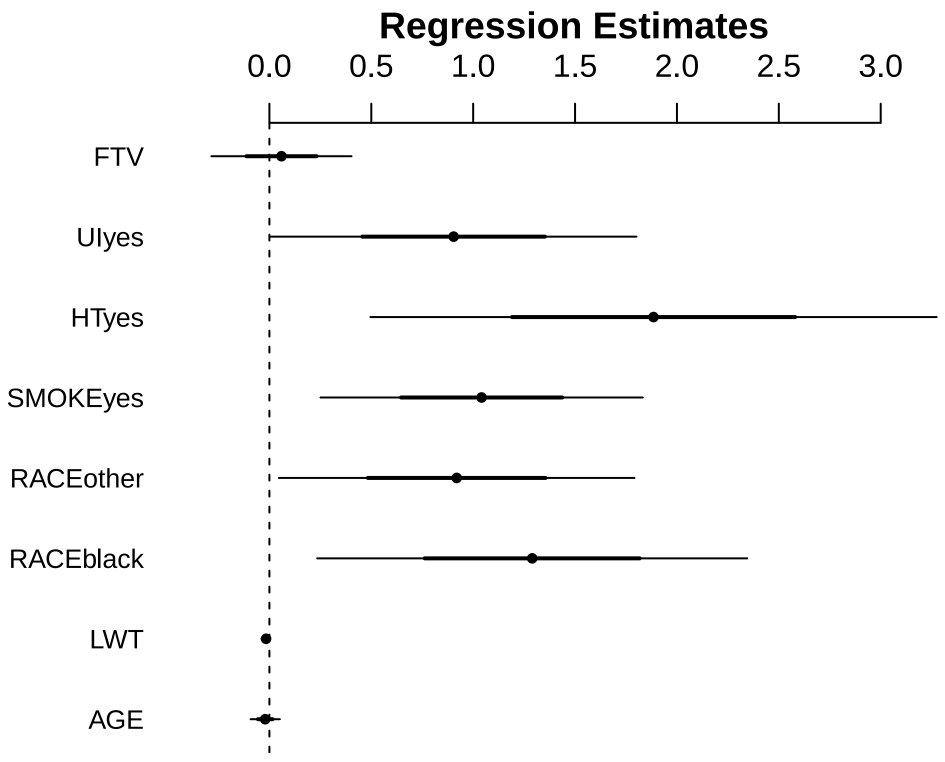
11.2 Fitting Logistic regression model—`glm()`

```r
M1low=glm(LOW~AGE+LWT+RACE+SMOKE+HT+UI+FTV,data=lowbwt,family=binomial())
round(coef(summary(M1low)),3)
```

```
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)    0.455      1.185   0.384    0.701
## AGE           -0.021      0.036  -0.570    0.568
## LWT           -0.017      0.007  -2.409    0.016
## RACEblack      1.290      0.528   2.445    0.015
## RACEother      0.919      0.436   2.106    0.035
## SMOKEyes       1.042      0.395   2.634    0.008
## HTyes          1.885      0.695   2.713    0.007
## UIyes          0.904      0.449   2.015    0.044
## FTV            0.059      0.172   0.344    0.731
```

It is evident from above output that except `Age` and `FTV` remaining all the regressor are significant. Thus, the odds ratio for `LWT` is $exp(-0.017) = 0.98$. It is interpreted as we expect to see $1.74\%(= 1 - 0.98)$ decreases in the odds of having a low birth weight baby for a one-unit increase in mother's weight, assuming all other predictors are fixed. This fact can be represented graphically using `coefplot()` of **arm** package

```r
require(arm)
opar=par(mar=c(4,10,4,2))# set margins of the figure
coefplot(M1low)
```



```r
par(opar)# return to the original settings
```

11.3 Count reaponses

We shall cover modeling of count data using Poisson regression, and Negative Binomial regression.

11.3.1 Poisson regression model

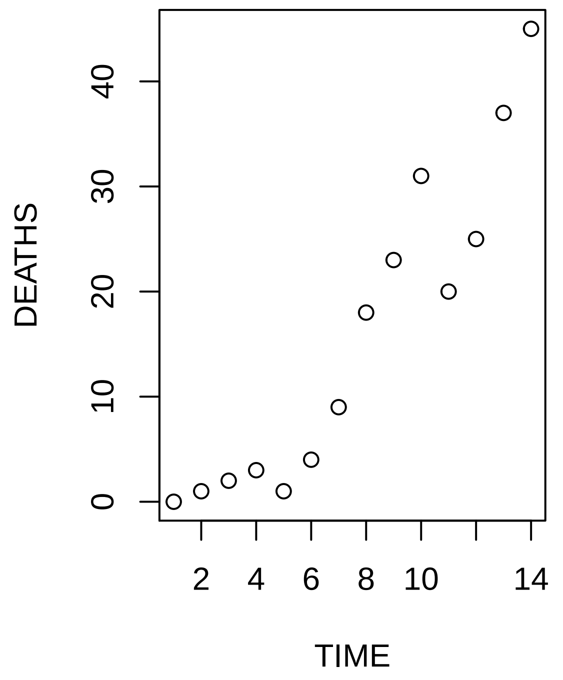
`AIDS` data available with **brinla** package provides a relationship between number of deaths `DEATH` with `TIME` time measured in multiple of three months after January 1983(continous).

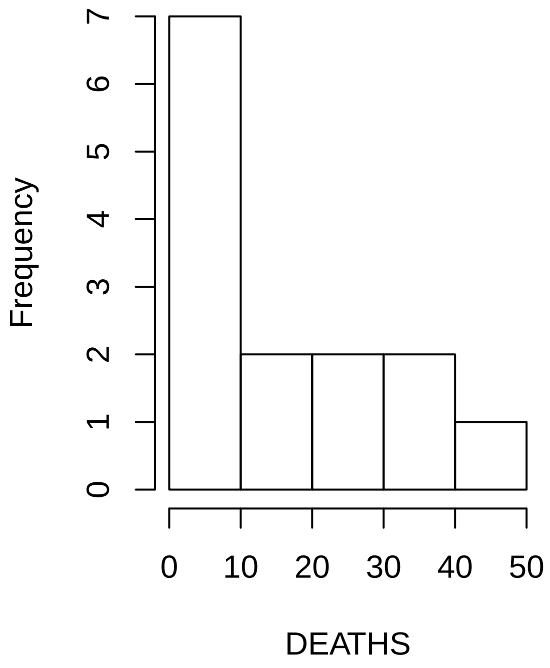
```r
require(brinla)
data(AIDS,package="brinla")
str(AIDS)
```

```
## 'data.frame':    14 obs. of  2 variables:
##  $ TIME  : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ DEATHS: num  0 1 2 3 1 4 9 18 23 31 ...
```

To understand the data we make a histogram of `DEATHS` and we shall make a plot of `DEATHS` Vs `TIME`. From these plots it is evident that the frequency distribution of `DEATHS` is highly skewed(not Normal). Moreover, there is a non linear relationship between `DEATHS` and `TIME`.

```r
opar=par(mfrow=c(1,2))
plot(DEATHS~TIME, data=AIDS,main="")
hist(AIDS$DEATHS,xlab = "DEATHS",main="")
```





```r
par(opar)
```

Now we fit the model using `glm()` with `family=poisson()`

```r
M1aids=glm(DEATHS~log(TIME),data=AIDS,family=poisson())
summary(M1aids)
```

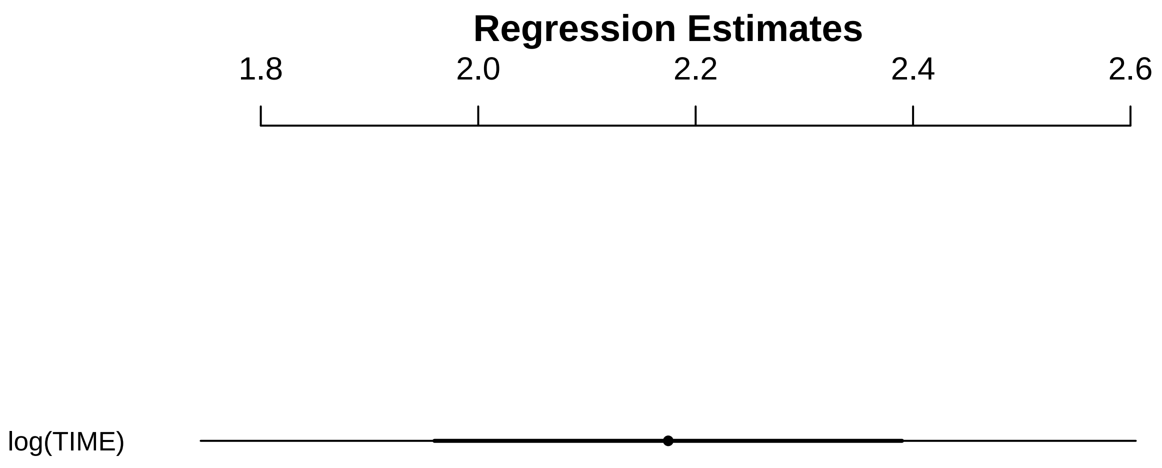
```
##
## Call:
## glm(formula = DEATHS ~ log(TIME), family = poisson(), data = AIDS)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.08992  -1.07141  -0.04657   0.38956   1.94311
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.9442     0.5116   -3.80 0.000145 ***
## log(TIME)     2.1748     0.2150   10.11  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 207.272  on 13  degrees of freedom
## Residual deviance:  17.092  on 12  degrees of freedom
## AIC: 74.019
```

```
##
## Number of Fisher Scoring iterations: 4
```

Thus, estimated equation is

$$\hat{\mu} = exp(-1.9442 + 2.1748 \times log(TIME))$$

```
require(arm)
coefplot(M1aids)
```



Since, vertical dashed line at 0 is absent, hence regression coefficient of `log(TIME)` is significant.

11.4 Negative Binomial regression model

It is used when over dispersion is found in the data. We are going to discuss `crab` data discussed by Agresti(2012) and same data is discussed by Wang et al(2018). The response variable in this data is number of satellites (`SATELLITES`) for each female crab, and there are four predictors.

```
require(brinla)
data(crab, package="brinla")
str(crab)
```

```
## 'data.frame':    173 obs. of  5 variables:
##  $ COLOR     : Factor w/ 4 levels "1","2","3","4": 2 3 3 4 2 1 4 2 2 2 ...
##  $ SPINE     : Factor w/ 3 levels "1","2","3": 3 3 3 2 3 2 3 3 1 3 ...
##  $ WEIGHT    : num  28.3 26 25.6 21 29 25 26.2 24.9 25.7 27.5 ...
##  $ WIDTH     : num  3.05 2.6 2.15 1.85 3 2.3 1.3 2.1 2 3.15 ...
##  $ SATELLITES: int  8 4 0 0 1 3 0 0 8 6 ...
```

11.4.1 Fitting of the model

Negative Binomial cab be fitted with the function `glm.nb` available with **MASS** package.

```
require(MASS)
M1crab=glm.nb(SATELLITES~COLOR+SPINE+WIDTH,data=crab)
round(coef(summary(M1crab)),3)
```

```
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -0.321      0.564  -0.570    0.569
## COLOR2        -0.321      0.373  -0.861    0.389
## COLOR3        -0.595      0.416  -1.432    0.152
## COLOR4        -0.579      0.464  -1.247    0.213
## SPINE2        -0.241      0.393  -0.613    0.540
## SPINE3         0.042      0.248   0.171    0.864
## WIDTH          0.693      0.166   4.183    0.000
```