THE UNIVERSITY OF MINNESOTA

Statistics 5021 May 1, 2003

MacAnova Notes on the Analaysis of Variance

This illustrates the use of MacAnova in many of the examples in Chapter 12,

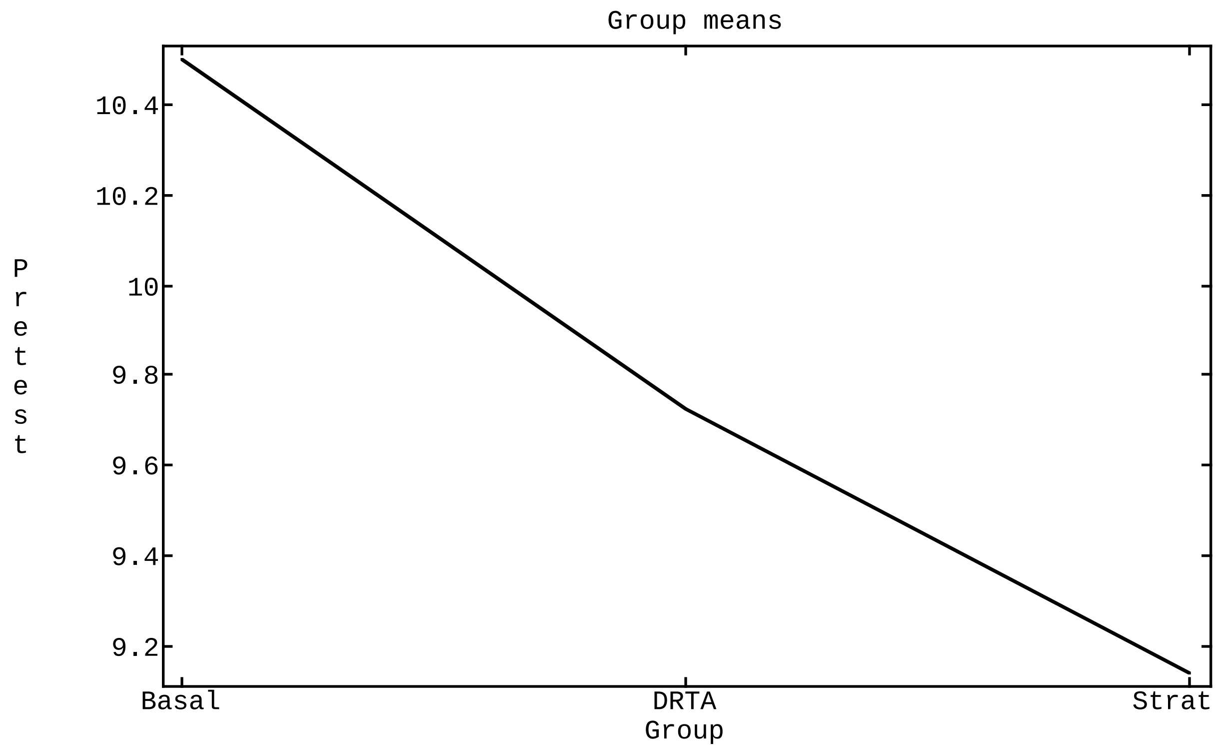
Example 12.6

```
Cmd> readdata("") # read ta12_001.txt
# File ta12_001.txt from publisher's web site
# Data for Table 12.1, p. 757 of IPS4
# Col. 1: id = subject number (1-66)
# Col. 2: group, (factor 1=Basal, 2=DRTA, 3=Strat)
# Col. 3: score = pretest reading score
Read from file "TP1:Stat5021:Data:Ch12:ta12_001.txt"
Column 1 saved as REAL vector id
Column 2 saved as factor group
Column 3 saved as REAL vector score

Cmd> hconcat(id,group,score)[run(6),] # first 10 cases
(1,1)          1          1          4
(2,1)          2          1          6
(3,1)          3          1          9
(4,1)          4          1         12
(5,1)          5          1         16
(6,1)          6          1         15
```

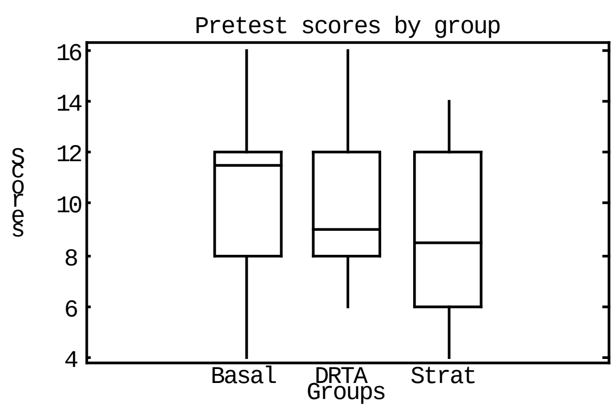
Note that `readdata()` automatically made `group` a factor. Here's a way to check the match of group numbers and factor levels.

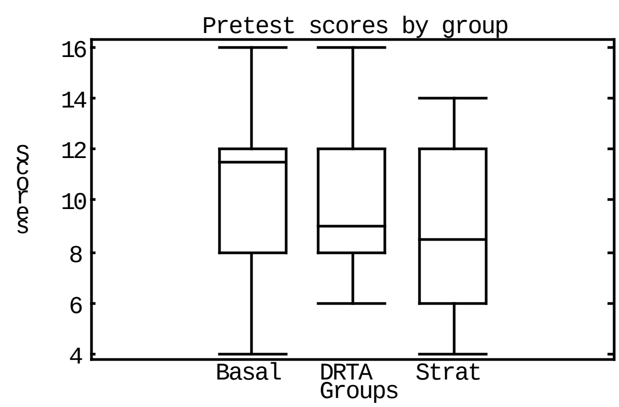
```
Cmd> group[unique(group,index:T)] # match of values and groups
       Basal        DRTA       Strat
           1           2           3

Cmd> groupnames <- unique(getlabels(group));groupnames
(1) "Basal"
(2) "DRTA"
(3) "Strat"

Cmd> tabs(score,group,mean:T,n:T,stddev:T) # compare with Fig. 12.7
component: mean
(1)        10.5      9.7273      9.1364
component: count
(1)          22          22          22
component: stddev
(1)      2.9721      2.6936      3.3423
```

```
Cmd> lineplot(1,tabs(score,group,mean:T),xlab:"Group",ylab:"Pretest",\
xticks:run(3),xticklab:groupnames,title:"Group means")
```




```
Cmd> vboxplot(split(score,group),xticklabs:groupnames,\
   xlab:"Groups",ylab:"Scores",title:"Pretest scores by group")

Cmd> vboxplot(split(score,group),xticklabs:groupnames,xlab:"Groups",\
   ylab:"Scores",title:"Pretest scores by group",boxtype:2)
```

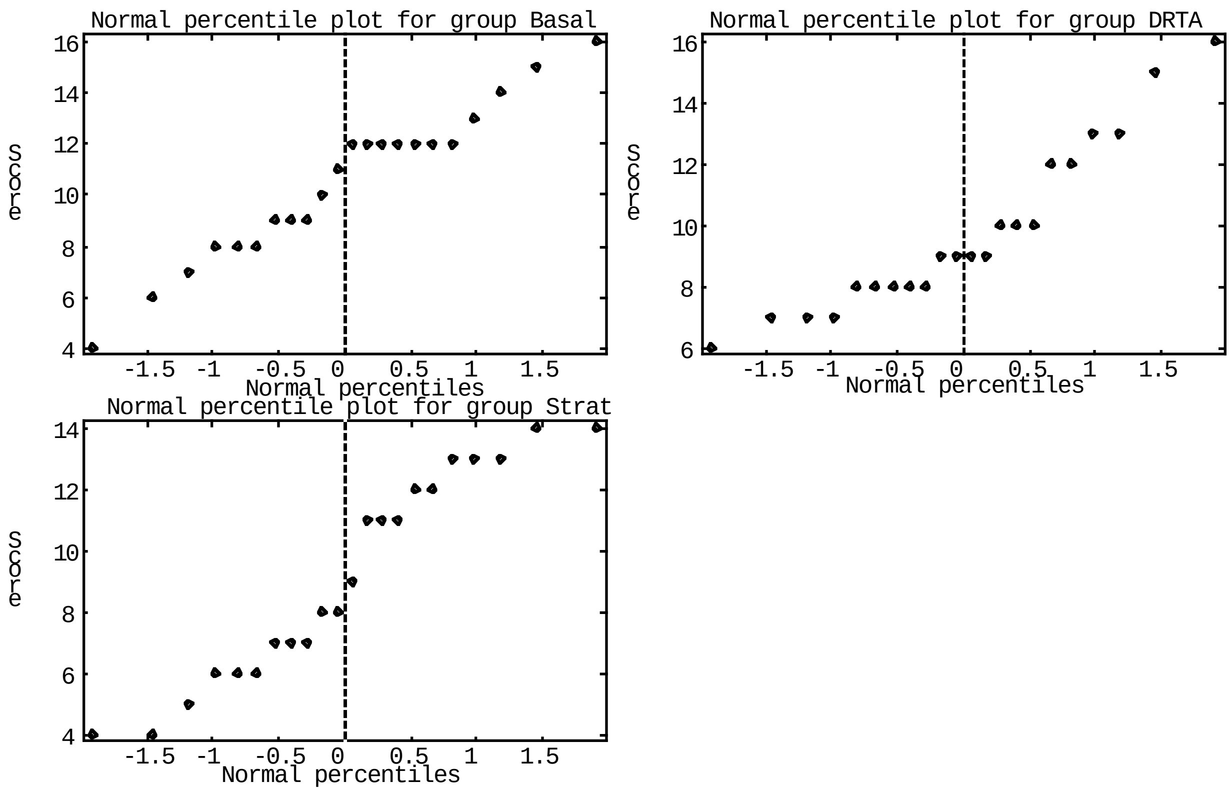




The left one is the standard MacAnova style. The right one, created with `boxtype:2` is similar to the style in Moore and McCabe.

Make normal percentile plots of each group separately

```
Cmd> tempy <- score[group == 1] # data for group 1

Cmd> plot(rankits(tempy),tempy,symbols:"\1",\
   xlab:"Normal percentiles",ylab:"Score",\
   title:"Normal percentile plot for group Basal")

Cmd> tempy <- score[group == 2]

Cmd> plot(rankits(tempy),tempy,symbols:"\1",\
   xlab:"Normal percentiles",ylab:"Score",\
   title:"Normal percentile plot for group DRTA")

Cmd> tempy <- score[group == 3]

Cmd> plot(rankits(tempy),tempy,symbols:"\1",\
   xlab:"Normal percentiles",ylab:"Score",\
   title:"Normal percentile plot for group Strat")
```



These all look adequately straight for small sample normal percentile plots.

```
Cmd> anova("score=group",fstat:T) # compare with Fig 12.8
Model used is score=group
                DF          SS          MS           F     P-value
CONSTANT         1        6323        6323   695.85803           0
group            2      20.576      10.288     1.13221     0.32879
ERROR1          63      572.45      9.0866

Cmd> DF # same as DF column
    CONSTANT       group      ERROR1
           1           2          63

Cmd> SS # same as SS column
    CONSTANT       group      ERROR1
        6323      20.576      572.45

Cmd> MS <- SS/DF; MS # same as MS column
    CONSTANT       group      ERROR1
        6323      10.288      9.0866

Cmd> fstats <- MS[run(2)]/MS[3]; fstats # same as F column
    CONSTANT       group
      695.86      1.1322

Cmd> pvalue <- 1 - cumF(fstats,DF[run(2)],DF[3]); pvalue
(1)            0     0.32879
```

Example 12.9

Create modified response by adding 1 to the Basal group and subtracting 1 from the Strat group.

```
Cmd> scorex <- score

Cmd> scorex[group==1] <- score[group==1] + 1

Cmd> scorex[group==3] <- score[group==3] - 1

Cmd> tabs(scorex,group,mean:T,stddev:T,n:T)
component: mean
(1)         11.5      9.7273      8.1364
component: count
(1)           22          22          22
component: stddev
(1)       2.9721      2.6936      3.3423

Cmd> anova("scorex=group",fstat:T)
Model used is scorex=group
                DF          SS          MS           F     P-value
CONSTANT         1        6323        6323   695.85803  9.3958e-36
group            2      124.58      62.288     6.85493    0.002025
ERROR1          63      572.45      9.0866
```

Although the text doesn't supply the post test data analyzed in several examples including 12.10, I have constructed artificial data with the same means and standard deviations. It is in file `eg12_010.txt` posted Thursday to the web site.

```
Cmd> readdata("")
# File eg12_010.txt (not from publisher's web site)
# Data similar to that used in Example 12.10. p. 765  of IPS4
# It has the same group means and the same standard deviations
# Col. 1: id = subject number (1-66)
# Col. 2: group, (factor 1=Basal, 2=DRTA, 3=Strat)
# Col. 3: score = pretest reading score
Read from file "TP1:Stat5021:Stat5021S03:Data:Ch12:eg12_010.txt"
Column 1 saved as REAL vector id
Column 2 saved as factor group
Column 3 saved as REAL vector posttest

Cmd> anova("posttest=group",fstat:T) # Compare with Figure 12.14
Model used is posttest=group
         DF          SS          MS           F     P-value
CONSTANT  1  1.2786e+05  1.2786e+05  3207.18687           0
group     2       357.3      178.65     4.48108    0.015151
ERROR1   63      2511.7      39.868
```

Example 12.14

The `contrast()` command computes statistics related to contrasts.

```
Cmd> info <- contrast(group,vector(-1,.5,.5)); info
component: estimate Value  of  contrast  -xbar_1+(xbar_2+xbar_3)/2
(1)      4.4545
component: ss
(1)      291.03
component: se       Standard  error
(1)      1.6487

Cmd> tstat <- info$estimate/info$se; tstat
(1)      2.7018

Cmd> 1 - cumstu(tstat,DF[3]) # 1-tail P-value
(1)   0.0044249

Cmd> t_025 <- invstu(1 - .025,DF[3]); t_025 # critical value
(1)      1.9983

Cmd> info$estimate + vector(-1,1)*t_025*info$se # 95% C.I.
(1)      1.1598      7.7492
```

Example12.15

```
Cmd> info <- contrast(group,vector(0,1,-1)); info
component: estimate        xbar_2  -  xbar_3
(1)      2.4545
component: ss
(1)      66.273
component: se              standard  error
(1)      1.9038

Cmd> tstat <- info$estimate/info$se; tstat
(1)      1.2893
```

```
Cmd> twotailt(tstat,DF[3]) # 2-tail P-value
(1)     0.20201

Cmd> info$estimate + vector(-1,1)*t_025*info$se # 95% C.I.
(1)     -1.3498      6.2589
```

Example 12.16

```
Cmd> mse <- SS[3]/DF[3];mse # Mean square error
      ERROR1
      39.995

Cmd> means <- tabs(posttest,group,mean:T); means # sample means
(1)      41.045      46.727      44.273

Cmd> n <- tabs(posttest,group,n:T); n # sample sizes
(1)          22          22          22

Cmd> stderr_12 <- sqrt(mse*(1/n[1]+1/n[2])); stderr_12
(1)      1.9038
```

Since $n_1 = n_2 = n_3 = 22$, all pairwise standard errors are the same.

```
Cmd> stderr_13 <-  stderr_23 <-  stderr_12

Cmd> t_12 <- (means[1] - means[2])/stderr_12; t_12
(1)     -2.9845

Cmd> t_13 <- (means[1] - means[3])/stderr_13; t_13
(1)     -1.6952

Cmd> t_23 <- (means[2] - means[3])/stderr_23; t_23
(1)      1.2893
```

Alternatively, you can compute these using `contrast()`.

```
Cmd> info12 <- contrast(group,vector(1,-1,0))

Cmd> info13 <- contrast(group,vector(1,0,-1))

Cmd> info23 <- contrast(group,vector(0,1,-1))

Cmd> info12$estimate/info12$se # t-statistic group 1 vs 2
(1)     -2.9845

Cmd> info13$estimate/info13$se # t-statistic group 1 vs 3
(1)     -1.6952

Cmd> info23$estimate/info23$se # t-statistic group 2 vs 3
(1)      1.2893
```

You get the critical value for Bonferroni comparisons the same way you get it for an ordinary t-test, except you divide the intended α value by the number of comparisons, 3 in this example.

```
Cmd> bonf_025 <- invstu(1 - .025/3,DF[3]); bonf_025
(1)      2.4596

Cmd> msd_12 <- bonf_025*stderr_12; msd_12
(1)      4.6899

Cmd> msd <- msd_12 # all are the same

Cmd> vector(info12$estimate,info13$estimate,info23$estimate)
```

```
(1)     -5.6818     -3.2273      2.4545

Cmd> vector(t_12,t_13,t_23)
(1)     -2.9845     -1.6952      1.2893
```

The only significant difference (larger than MSD) is between groups 1 and 2.

You get the same result by multiplying P-values by the number of comparisons.

```
Cmd> 3*twotailt(vector(t_12,t_13,t_23),DF[3])
(1)    0.012117     0.28493     0.60602
```

Example 12.17

`pairwise()` does them all together, producing a diagram more or less equivalent to Figure 12.16.

```
Cmd> pairwise("group",.05)
   |      1    -2.97
   | |    3    0.258
     |    2     2.71
```

Groups connected by a vertical line are not significantly different at the 5% signficance level. Because there is no vertical line connecting them, groups are 1 (Basal) and 2 (DRTA) are significantly different as the Bonferronized t-tests have shown.

Example 12.18

```
Cmd> means[1] - means[2] + vector(-1,1)*bonf_025*stderr_12
(1)     -10.372    -0.99187

Cmd> info12$estimate + vector(-1,1)*bonf_025*info12$se # same thing
(1)     -10.364    -0.99932

Cmd> info13$estimate + vector(-1,1)*bonf_025*info13$se
(1)     -7.9098      1.4552

Cmd> info23$estimate + vector(-1,1)*bonf_025*info23$se
(1)      -2.228       7.137
```

These are simultaneous confidence intervals. In repeated sampling, all three intervals will contain the true differences $\mu_i - \mu_j$ with probability at least 0.95.

Example 12.20

```
Cmd> mu <- vector(41,47,44); sigma <- 7 # p. 779

Cmd> n <- 10

Cmd> lambda <- n*sum((mu - sum(mu)/3)^2)/sigma^2; lambda
(1)      3.6735

Cmd> df_error <- 3*(n-1); df_error
(1)          27

Cmd> df_treat <- 3 - 1; df_treat
(1)           2

Cmd> f_05 <- invF(1 - .05,df_treat,df_error); f_05
(1)      3.3541         5% critical value for F-test

Cmd> 1 - cumF(f_05,2,df_error,lambda) # P(F > f_05 | H_a)
(1)     0.34893         Power

Cmd> noncen <- sum((mu - sum(mu)/3)^2)/sigma^2  # lambda/n
```

```
Cmd> power(noncen,3,n,.05) # black box way to find power
(1)     0.34893

Cmd> power(noncen,3,.05,n) # either order of args should work
(1)     0.34893
```

To find a sample size that has specified power, you can use `samplesize()`:

```
Cmd> amplesize(noncen,3,.05,.9)
(1)          36

Cmd> power(noncen,3,.05,36) # power ≥ .90 for n = 36
(1)     0.90461

Cmd> power(noncen,3,.05,35) # power < .90 for n = 35
(1)     0.89581
```