Week6 IC2 Example

Pick the variable for your group by changing the y¡-Y5 statement (Y1 for group 1, Y2 for group 2, etc.)

```
load("week6_IC1_problem1.Rdata")
str(Lvl)

##  Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...

y<-Y5
N<-length(y)
```

Peform a classical or frequentist ANOVA

This is also known as an ”ordinary least squares” ANOVA.

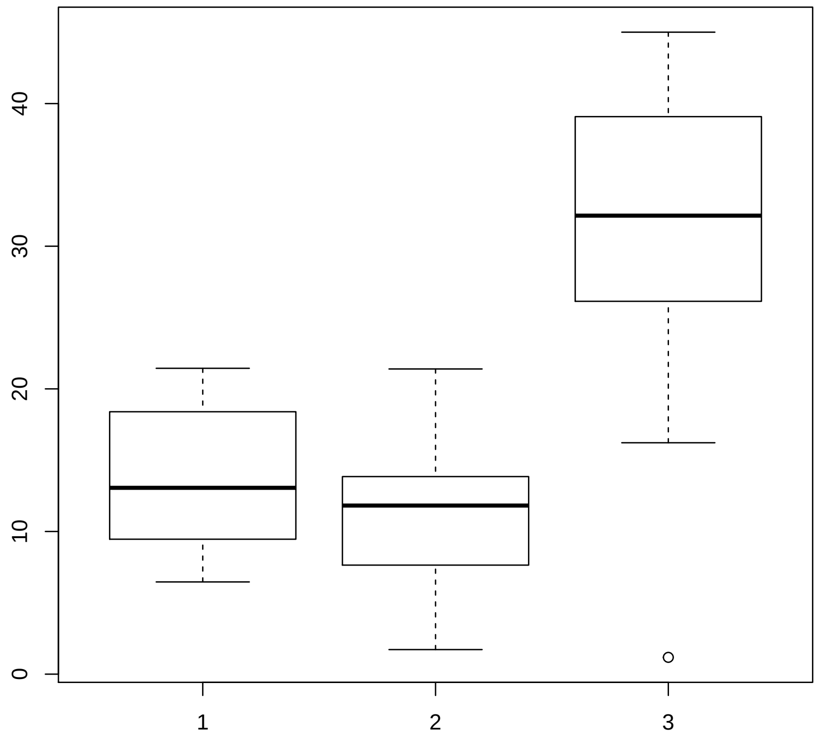
```
aov1<-aov(y~Lvl)            #run the ANOVA with factor Lvl
summary(aov1)

##             Df Sum Sq Mean Sq F value   Pr(>F)    
## Lvl          2   6248  3124.1   58.21 1.85e-15 ***
## Residuals   68   3650    53.7                     
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

tapply(Y5, Lvl, mean)       #compute means by level

##        1        2        3 
## 13.44741 11.46931 31.92787

boxplot(Y5~Lvl)             #boxplots of data by level
```



The coefficients represent the estimates of alpha1, (alpha2-alpha1), and (alpha3-alpha1), respectively:

```
aov1$coefficients

## (Intercept)        Lvl2        Lvl3
##   13.447406   -1.978098   18.480467
```

perform a Bayesian ANOVA

```
L<-nlevels(Lvl)                 #number of levels for the factor
level<-as.integer(Lvl)          #convert factor to an integer
                                #variable Lvl contains the single factor levels
```

Call STAN with model file specifying separate standard deviations by level

```r
library(rstan)                                  #make sure rstan is available

## Loading required package:  ggplot2
## rstan (Version 2.9.0-3, packaged:  2016-02-11 15:54:41 UTC, GitRev:
05c3d0058b6a)
## For execution on a local, multicore CPU with excess RAM we recommend
calling
## rstan_options(auto_write = TRUE)
## options(mc.cores = parallel::detectCores())

rstan_options(auto_write = TRUE)                #use multiple cores
options(mc.cores = parallel::detectCores())     #if we have them
stanfit<-stan("week6_IC1_problem1.stan")         #call STAN using defaults
print(stanfit)

## Inference for Stan model: week6_IC1_problem1.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##              mean se_mean   sd    2.5%     25%     50%     75%   97.5%
## alpha[1]    13.30    0.02 1.00   11.36   12.61   13.31   14.00   15.23
## alpha[2]    11.34    0.03 1.21    8.85   10.57   11.34   12.13   13.70
## alpha[3]    30.70    0.04 2.00   26.65   29.42   30.74   32.03   34.45
## sigma[1]     4.92    0.01 0.73    3.74    4.41    4.85    5.35    6.59
## sigma[2]     5.67    0.02 0.90    4.21    5.04    5.56    6.19    7.76
## sigma[3]    10.32    0.03 1.56    7.81    9.23   10.10   11.21   13.93
## d12          1.97    0.03 1.59   -1.20    0.89    1.96    3.02    5.09
## d13        -17.39    0.04 2.24  -21.76  -18.89  -17.44  -15.95  -12.89
## d23        -19.36    0.05 2.38  -24.02  -20.99  -19.38  -17.79  -14.68
## lp__      -174.15    0.04 1.76 -178.46 -175.08 -173.80 -172.84 -171.78
##           n_eff Rhat
## alpha[1]   2773    1
## alpha[2]   2230    1
## alpha[3]   2616    1
## sigma[1]   2574    1
## sigma[2]   2180    1
## sigma[3]   2798    1
## d12        2137    1
## d13        2710    1
## d23        2698    1
## lp__       1712    1
##
## Samples were drawn using NUTS(diag_e) at Thu Feb 25 07:23:14 2016.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

launch shinystan

```
library(shinystan)

## Loading required package:  shiny
##
## This is shinystan version 2.1.0

launch_shinystan(stanfit)

##
## Loading...
## Note:  for large models ShinyStan may take a few moments to launch.
##
## Listening on http://127.0.0.1:3031
```