Example 8.2.1 Daniel/Cross

D. Zeitler

October 15, 2015

```
#' Setup the libraries we'll need - install them if needed
library(RcmdrMisc)
```

```
## Loading required package: car
```

```
## Loading required package: sandwich
```

```
library(multcomp)
```

```
## Loading required package: mvtnorm
```

```
## Loading required package: survival
```

```
## Loading required package: TH.data
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'TH.data'
```

```
## The following object is masked from 'package:MASS':
##
##     geyser
```

```
library(abind)
library(MASS)
library(mosaic)
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following object is masked from 'package:car':
##
##     recode
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Loading required package: mosaicData
```

```
## Loading required package: Matrix
```

```
##
## The 'mosaic' package masks several functions from core packages in order to add additional features.
## The original behavior of these functions should not be affected by this.

##
## Attaching package: 'mosaic'

## The following object is masked from 'package:Matrix':
##
##     mean

## The following objects are masked from 'package:dplyr':
##
##     count, do, tally

## The following objects are masked from 'package:car':
##
##     deltaMethod, logit

## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cov, D, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var

## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum
```

```
library(tidyr)
```

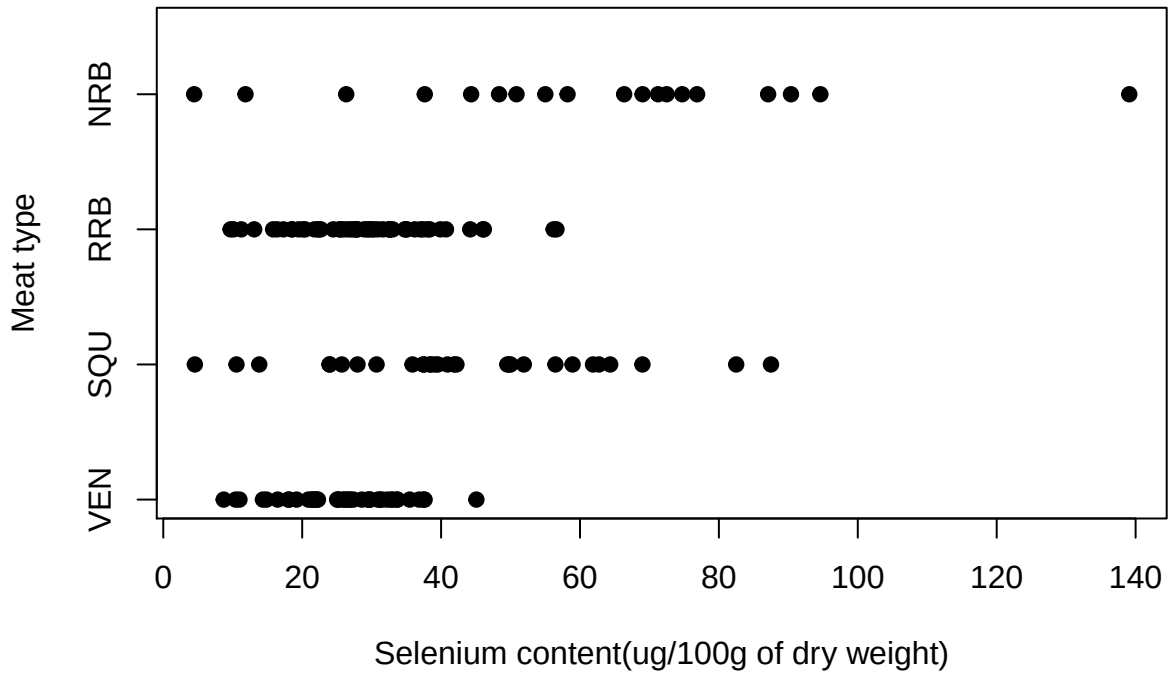
```
##
## Attaching package: 'tidyr'

## The following object is masked from 'package:Matrix':
##
##     expand
```

First let's read in the data and take a quick look at it.

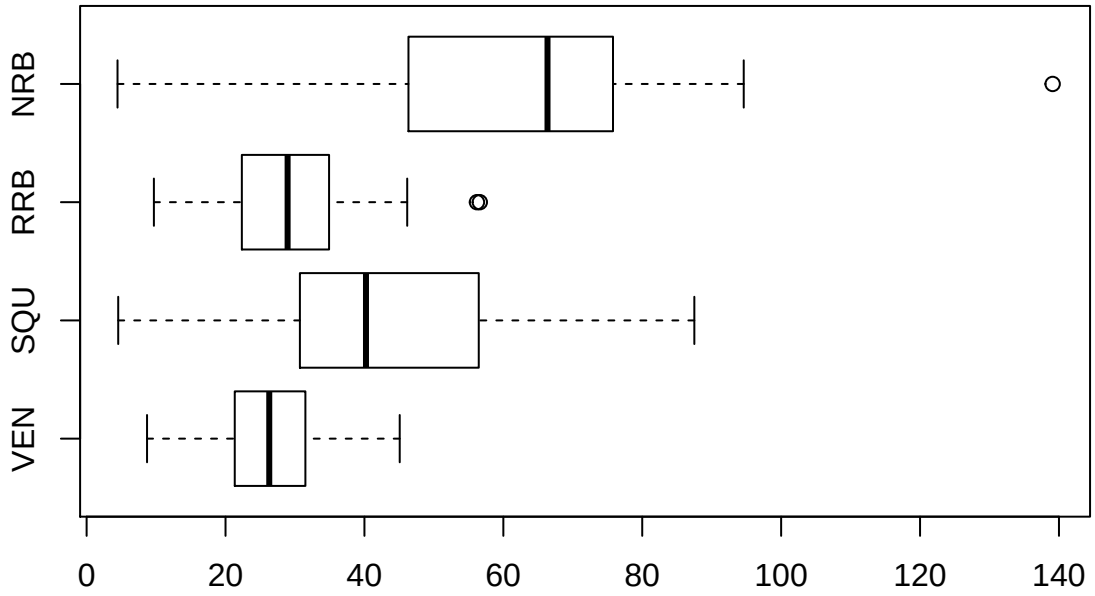
```
game.meats <- read.csv('EXA_C08_S02_01.csv')
head(game.meats)
```

```
##     VEN   SQU   RRB   NRB
## 1 26.72 37.42 11.23 44.33
## 2 28.58 56.46 29.63 76.86
## 3 29.71 51.91 20.42  4.45
## 4 26.95 62.73 10.12 55.01
## 5 10.97  4.55 39.91 58.21
## 6 21.97 39.17 32.66 74.72
```

```
stripchart(game.meats,method = "stack",pch=19,xlab='Selenium content(ug/100g of dry weight)',ylab='Meat
```




```
boxplot(game.meats,horizontal = T)
```



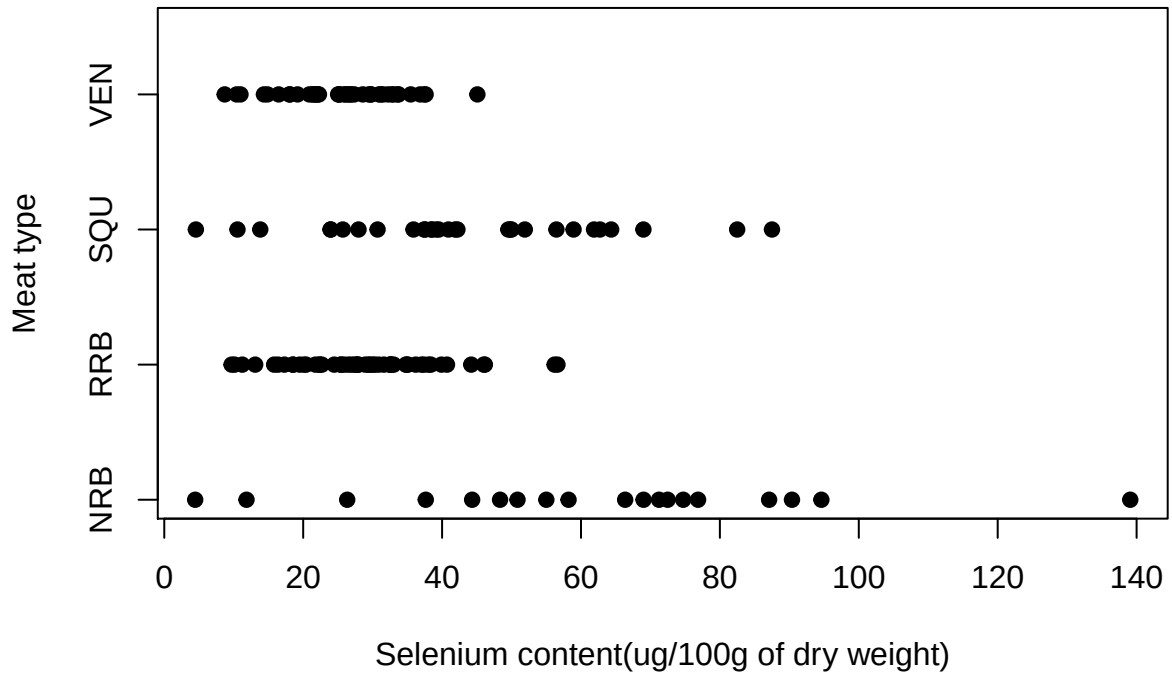
This data is not in our usual structure, let's restructure it.

```
game <- game.meats %>%
  gather(Game,Selenium,VEN:NRB) %>%
  filter(!is.na(Selenium)) %>%
  mutate(Game=factor(Game))
head(game)
```

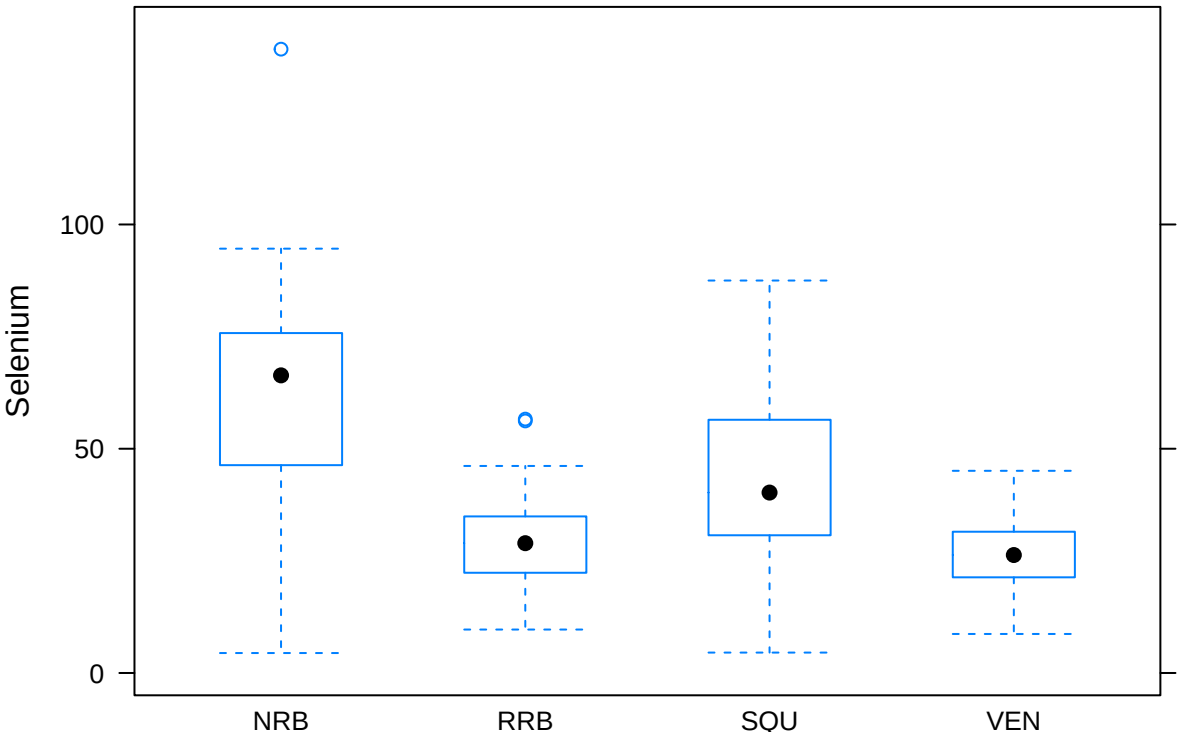
```
##   Game Selenium
## 1  VEN    26.72
## 2  VEN    28.58
## 3  VEN    29.71
```

```
## 4  VEN    26.95
## 5  VEN    10.97
## 6  VEN    21.97
```

```
stripchart(Selenium~Game,game,method='stack',pch=19,xlab='Selenium content(ug/100g of dry weight)',ylab=
```




```
bwplot(Selenium~Game,data=game)
```



Now run a levene's test.

```
favstats(Selenium~Game, data=game)
```

```
##   Game  min      Q1 median      Q3    max     mean        sd  n missing
## 1  NRB 4.45 46.3400 66.360 75.7900 139.09 62.04632 31.149825 19       0
## 2  RRB 9.69 22.3500 28.940 34.9100  56.61 29.08302 10.376722 53       0
## 3  SQU 4.55 32.0025 40.225 55.3225  87.50 43.24567 19.508738 30       0
## 4  VEN 8.70 21.3700 26.300 31.4250  45.08 25.87548  8.032421 42       0
```

```
leveneTest(Selenium~Game,data=game,center=mean)
```

```
## Levene's Test for Homogeneity of Variance (center = mean)
##        Df F value    Pr(>F)
## group   3   15.24 1.233e-08 ***
##       140
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The test rejected H_0, so we really shouldn't use ANOVA, but it's relatively robust so go ahead.

```
gameAnova <- aov(Selenium~Game,data=game)
summary(gameAnova)
```

```
##              Df Sum Sq Mean Sq F value   Pr(>F)
## Game          3  21262    7087      27 7.68e-14 ***
## Residuals   140  36747     262
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Look at the pairwise confidence intervals.

```
numSummary(game$Selenium , groups=game$Game,
           statistics=c("mean", "sd"))
```

```
##         mean        sd data:n
## NRB 62.04632 31.149825     19
## RRB 29.08302 10.376722     53
## SQU 43.24567 19.508738     30
## VEN 25.87548  8.032421     42
```

```
.Pairs <- glht(gameAnova, linfct = mcp(Game = "Tukey"))
summary(.Pairs) # pairwise tests
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Tukey Contrasts
##
##
## Fit: aov(formula = Selenium ~ Game, data = game)
##
## Linear Hypotheses:
##                Estimate Std. Error t value Pr(>|t|)
## RRB - NRB == 0  -32.963      4.332  -7.609  < 0.001 ***
## SQU - NRB == 0  -18.801      4.750  -3.958  < 0.001 ***
## VEN - NRB == 0  -36.171      4.479  -8.075  < 0.001 ***
## SQU - RRB == 0   14.163      3.702   3.826  0.00107 **
## VEN - RRB == 0   -3.208      3.347  -0.958  0.77036
## VEN - SQU == 0  -17.370      3.873  -4.485  < 0.001 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

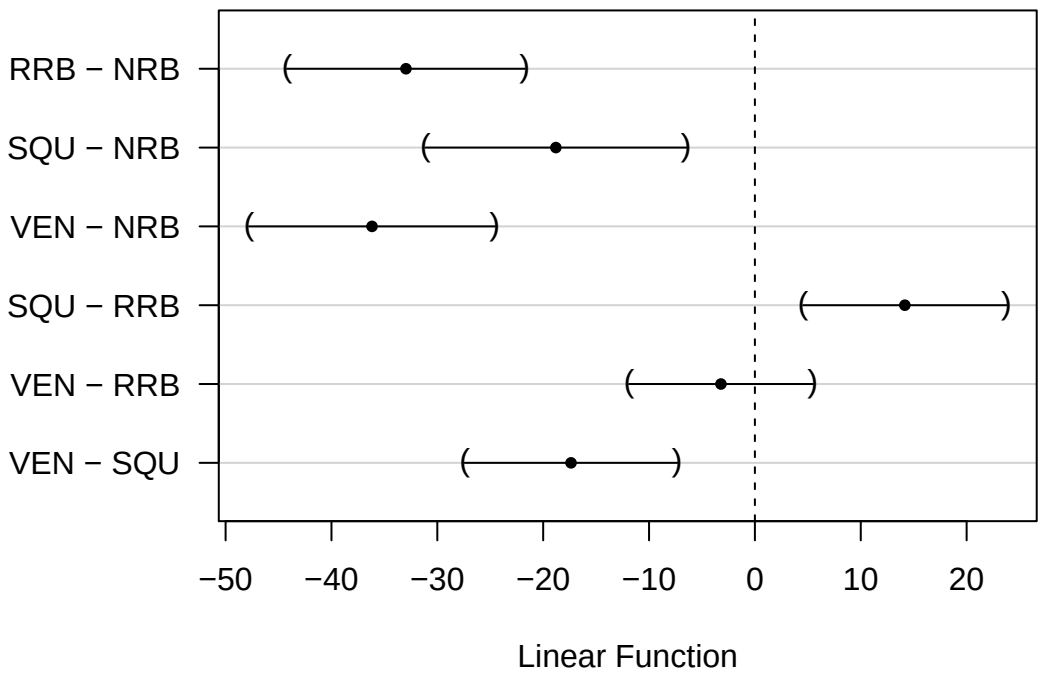
```
confint(.Pairs) # confidence intervals
```

```
##
##   Simultaneous Confidence Intervals
##
## Multiple Comparisons of Means: Tukey Contrasts
##
##
## Fit: aov(formula = Selenium ~ Game, data = game)
##
## Quantile = 2.594
## 95% family-wise confidence level
##
##
## Linear Hypotheses:
##                 Estimate lwr      upr
## RRB - NRB == 0 -32.9633 -44.2006 -21.7259
## SQU - NRB == 0 -18.8006 -31.1224  -6.4789
## VEN - NRB == 0 -36.1708 -47.7900 -24.5517
## SQU - RRB == 0  14.1626   4.5608  23.7645
## VEN - RRB == 0  -3.2075 -11.8894   5.4743
## VEN - SQU == 0 -17.3702 -27.4162  -7.3242
```

```
cld(.Pairs) # compact letter display
```

```
## NRB RRB SQU VEN
## "c" "a" "b" "a"
```

```
old.oma <- par(oma=c(0,5,0,0))
plot(confint(.Pairs))
```




```
par(old.oma)
remove(.Pairs)
```

Now let's do it with non-parametrics

```
tapply(game$Selenium, game$Game, median, na.rm=TRUE)
```

```
##    NRB    RRB    SQU    VEN
## 66.360 28.940 40.225 26.300
```

```
tapply(game$Selenium, game$Game, IQR, na.rm=TRUE)
```

```
##    NRB    RRB    SQU    VEN
## 29.450 12.560 23.320 10.055
```

```
kruskal.test(Selenium ~ Game, data=game)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  Selenium by Game
## Kruskal-Wallis chi-squared = 40.229, df = 3, p-value = 9.53e-09
```

pairwise

```
with(game,pairwise.wilcox.test(Selenium,Game,p.adjust.method='none'))
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test
##
## data:  Selenium and Game
##
```

```
##     NRB     RRB     SQU
## RRB 8.7e-06 -       -
## SQU 0.01167 0.00014 -
## VEN 8.4e-07 0.13492 3.9e-06
##
## P value adjustment method: none
```

```
with(game,pairwise.wilcox.test(Selenium,Game,p.adjust.method='bonferroni'))
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test
##
## data:  Selenium and Game
##
##     NRB     RRB     SQU
## RRB 5.2e-05 -       -
## SQU 0.07003 0.00085 -
## VEN 5.0e-06 0.80952 2.3e-05
##
## P value adjustment method: bonferroni
```