Introduction to R output

2023-01-14

```
################  Introduction to R, STAT 472/572  #################

##R package can be downloaded from website for free http://www.r-project.org##

##use "help" to know the functions##

help(sqrt)
#r will tell you the usage of sqrt function such as explaination and examples

## or use help.search("***")

## Use a '#' comments out any text afterwards on a line

######### Random Number Generators #####################################
##Seed
##A pseudo random number generator is an algorithm based on a starting point called "seed".
##If you want to perform an exact replication of your program, you have to specify the seed
##using the function set.seed(). The argument of set.seed has to be an integer.
set.seed(1)
runif(10)
```

```
##  [1] 0.26550866 0.37212390 0.57285336 0.90820779 0.20168193 0.89838968
##  [7] 0.94467527 0.66079779 0.62911404 0.06178627
```

```
# got different numbers
runif(10)
```

```
##  [1] 0.2059746 0.1765568 0.6870228 0.3841037 0.7698414 0.4976992 0.7176185
##  [8] 0.9919061 0.3800352 0.7774452
```

```
# get back the numbers with seed 1
set.seed(1)
runif(10)
```

```
##  [1] 0.26550866 0.37212390 0.57285336 0.90820779 0.20168193 0.89838968
##  [7] 0.94467527 0.66079779 0.62911404 0.06178627
```

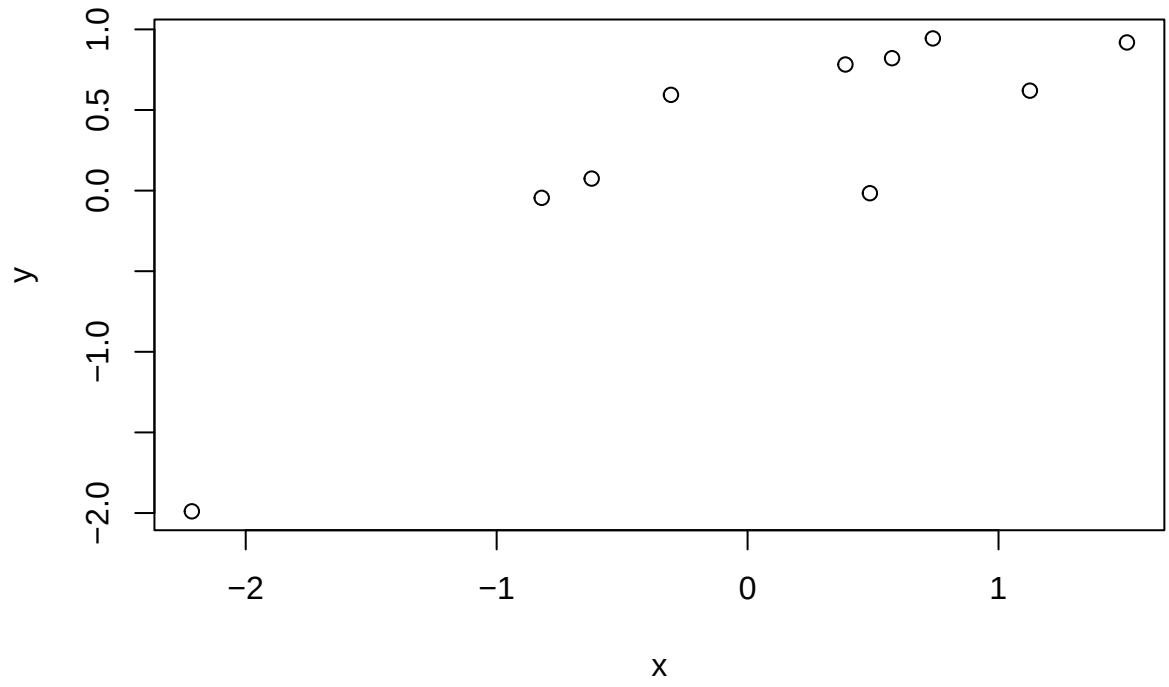
```
x <- rnorm(10,mean=0,sd=1) ## default generates N(0,1)
x
```

```
##  [1] -0.8204684  0.4874291  0.7383247  0.5757814 -0.3053884  1.5117812
##  [7]  0.3898432 -0.6212406 -2.2146999  1.1249309
```

```r
y <- rnorm(n=10)
y
```

```
##  [1] -0.04493361 -0.01619026  0.94383621  0.82122120  0.59390132  0.91897737
##  [7]  0.78213630  0.07456498 -1.98935170  0.61982575
```

```r
## plotting
plot(x,y)
```




```r
##Sampling in a vector
##Toss 10 coins

sample(0:1,10,replace=T)
```

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

```r
#Roll 10 dice

sample(1:6,10,replace=T)
```

```
##  [1] 3 2 6 6 2 5 2 6 6 6
```

```r
##play lottery (6 random numbers out of 49 without replacement)

sample(1:49,6,replace=F) ##replace=F is the default
```

```
## [1]  1 40  6 23 44 47
```

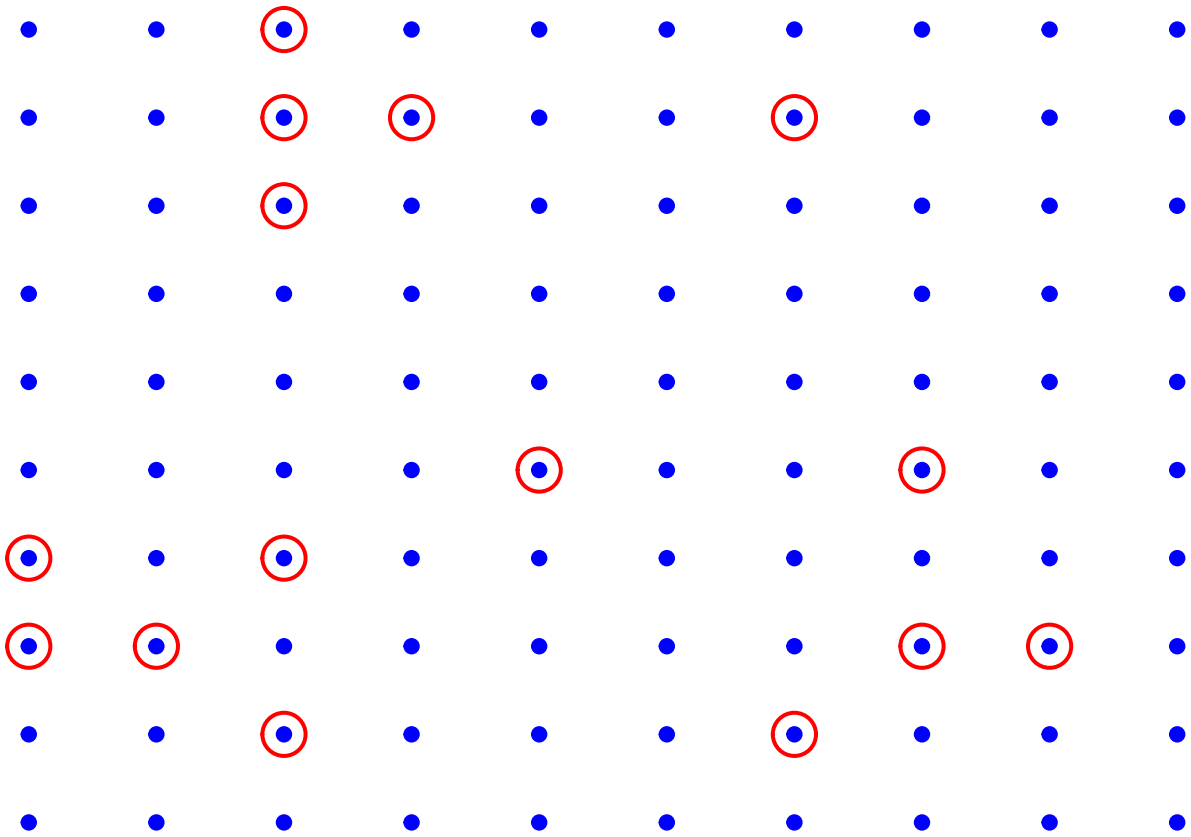
```r
##select a simple random sample

a<-c(1,3,5,7,9,11,13,15,17,19)
b<-sample(1:10,5)
b
```

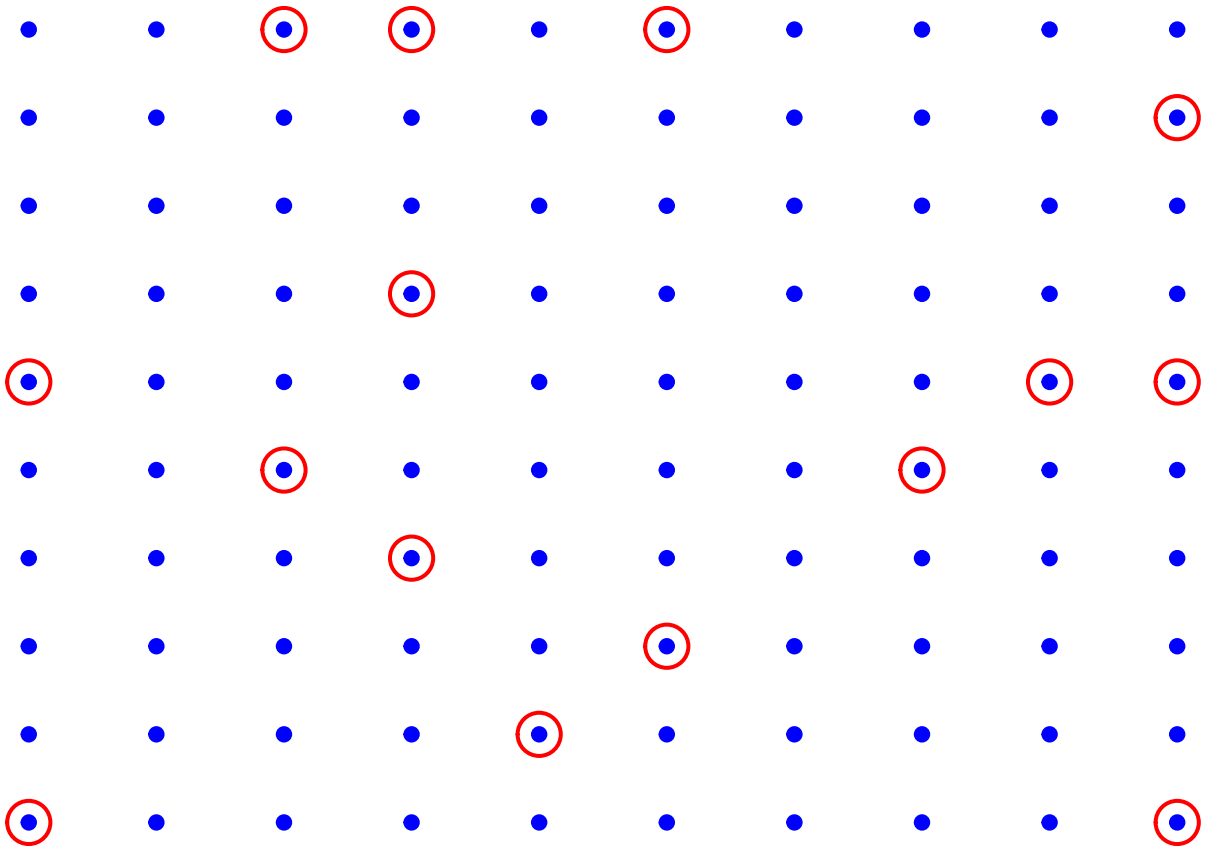
```
## [1] 8 7 3 1 4
```

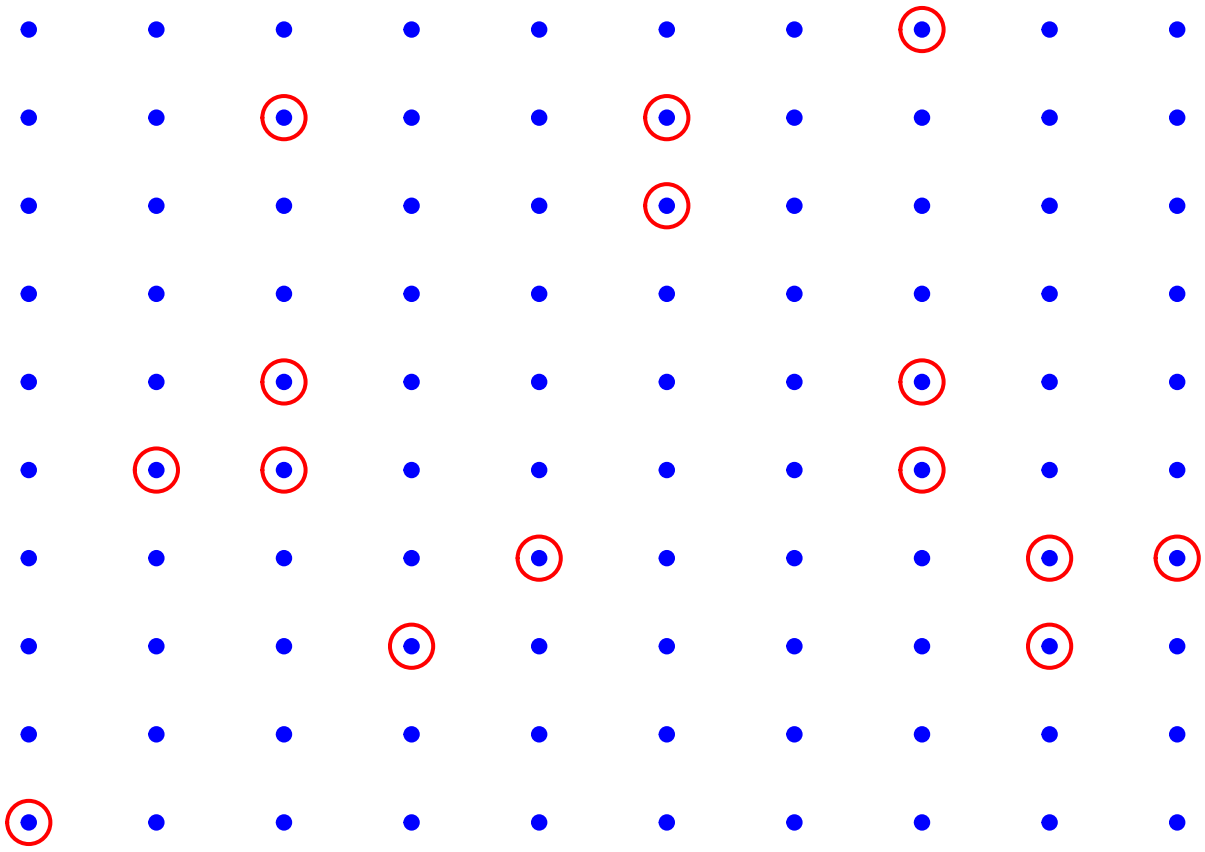
```r
a[b]
```

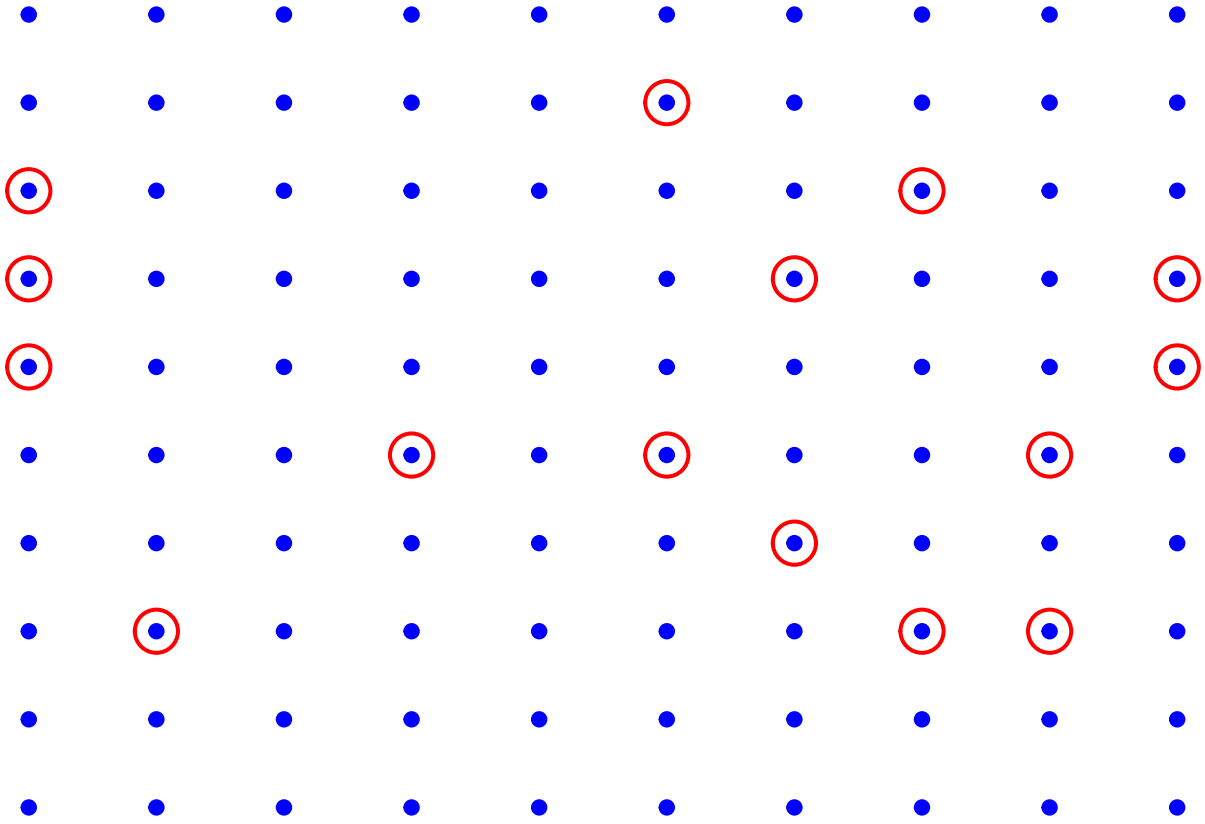
```
## [1] 15 13  5  1  7
```

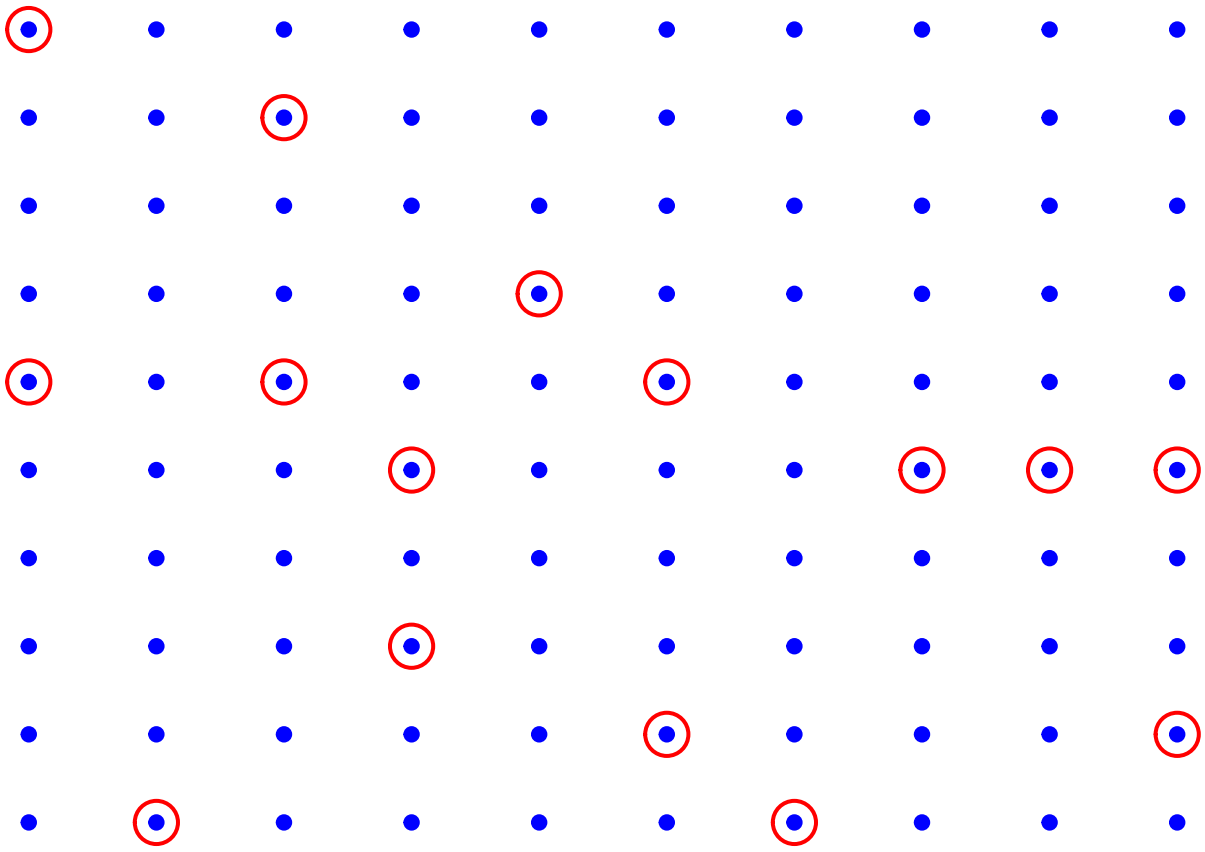
```r
#or
srs.sample<-a[sample(1:10, 5, replace=F)]

##illustrate simple random sampling (from Yihui XIE)
x = cbind(rep(1:10, 10), gl(10, 10))
par(mar = rep(0.1, 4))
for (i in 1:10) {
  plot(x, pch = 19, col = "blue", axes = F, ann = F)
  points(x[sample(100, 15), ], col = "red", cex = 3, lwd = 2)
  Sys.sleep(1)
}
```

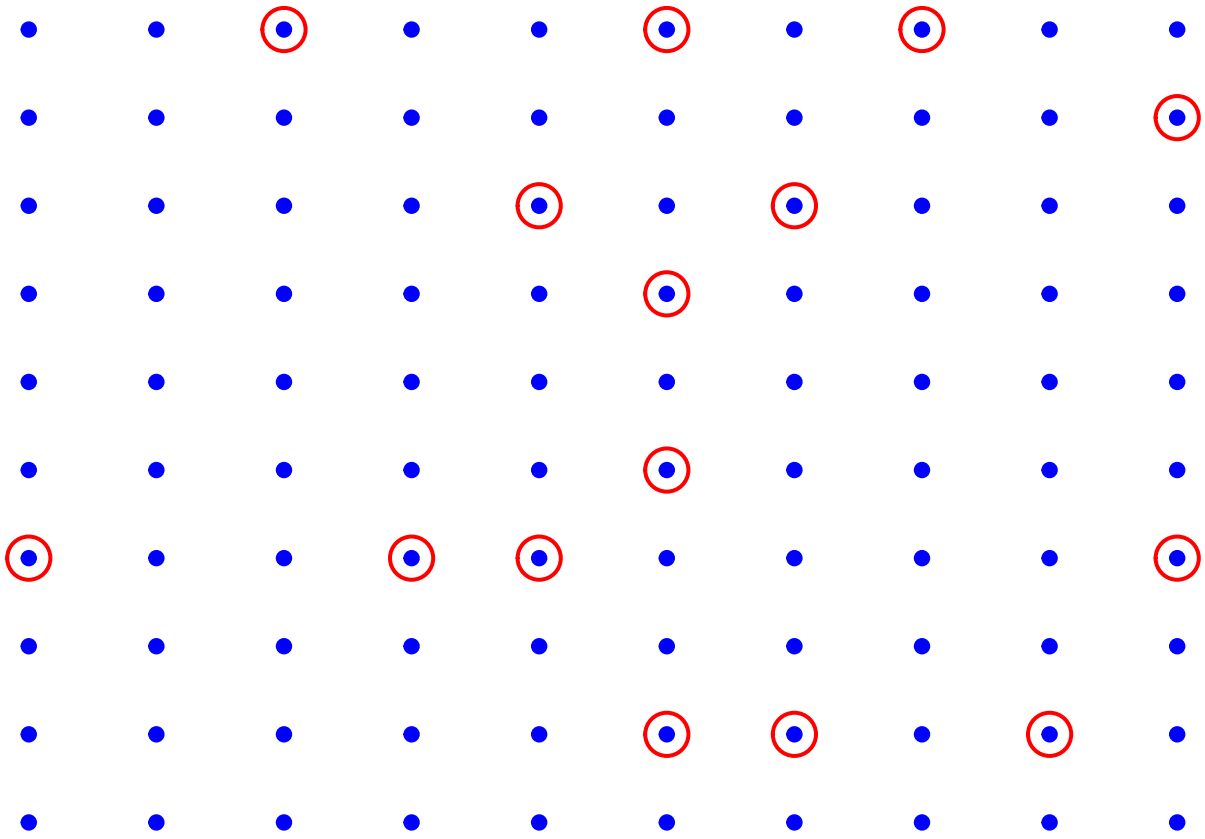


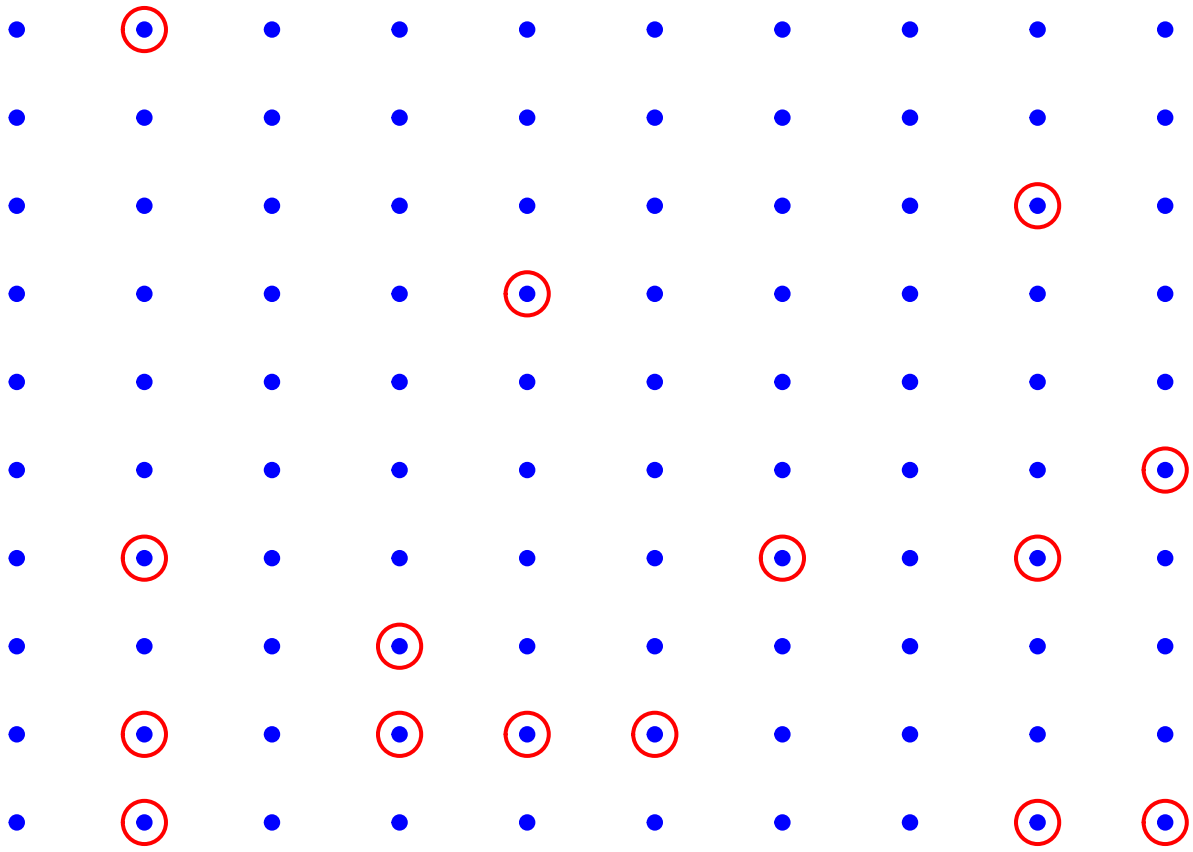


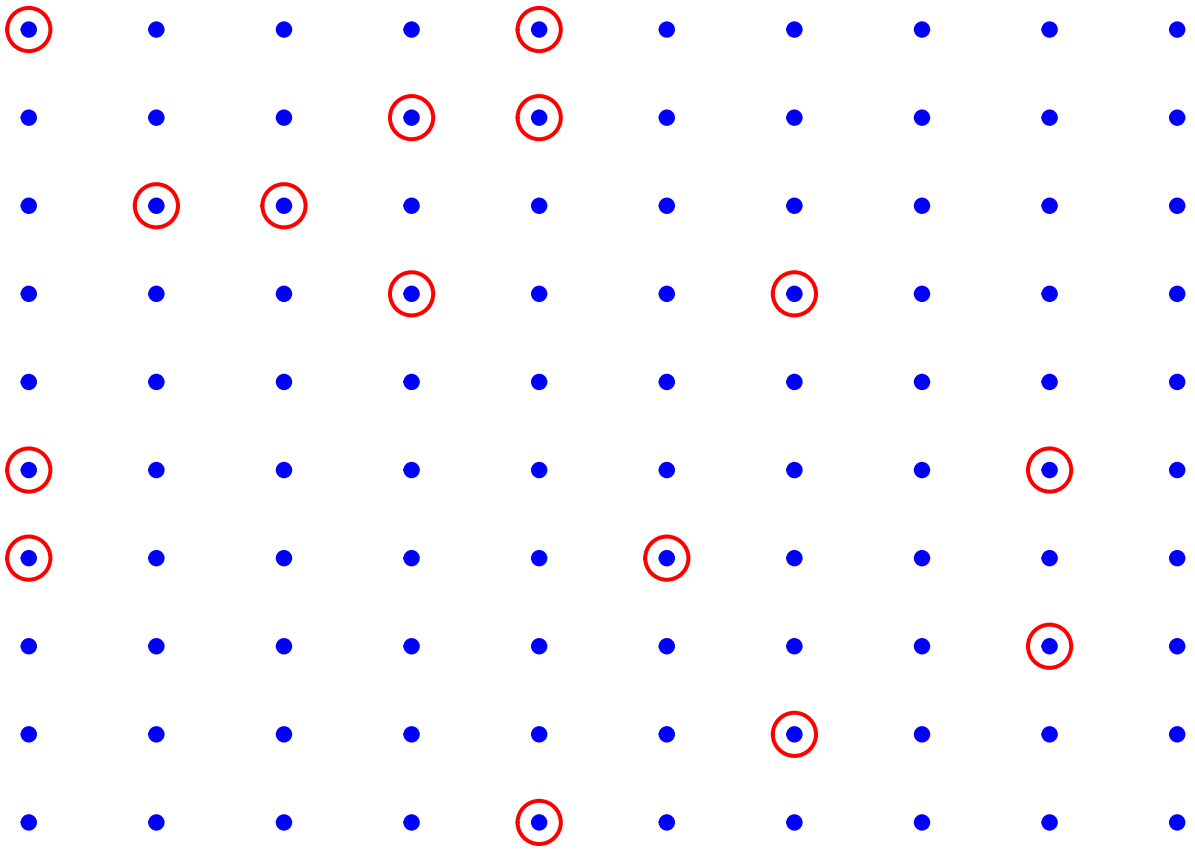


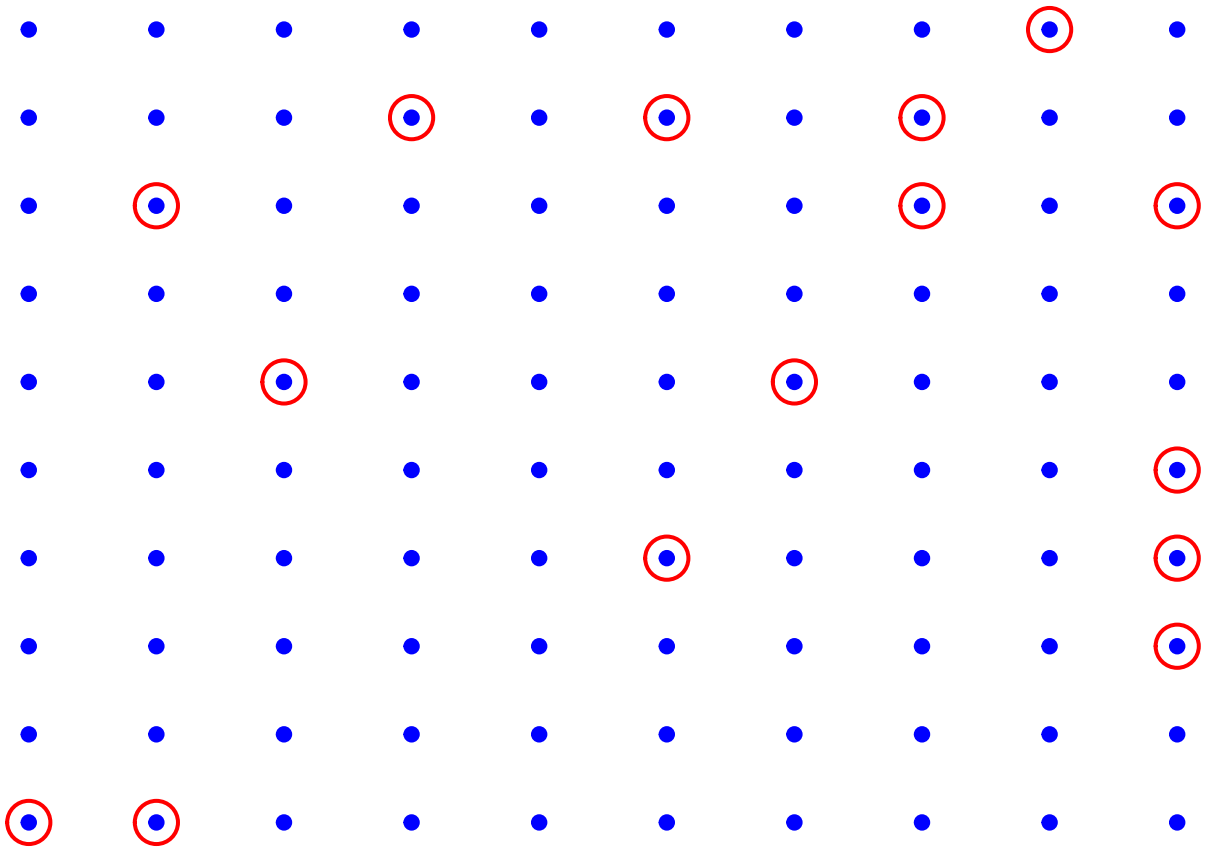


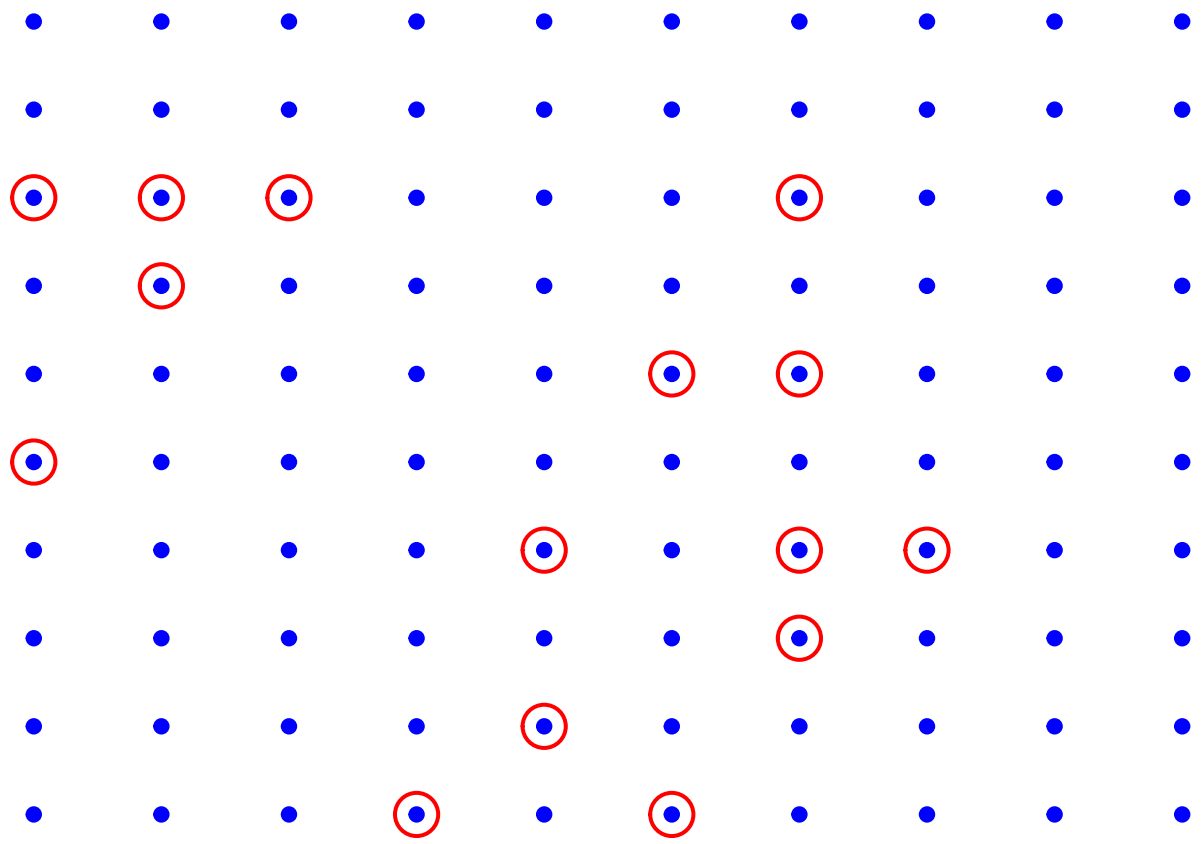












```
###### Matrix Algebra #########


######### Vectors ############

## Concatenate function 'c' to create a vector
u <- c(1,2,3,4)

## print u
print(u)
```

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

```
##just typing u also prints
u
```

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

```
u <- 1:4  ## makes the same vector as above
u
```

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

```r
v <- seq(0,1.5,by=.5)
v
```

```
## [1] 0.0 0.5 1.0 1.5
```

```r
v <- seq(0,1.5,length=4) ## same thing
v
```

```
## [1] 0.0 0.5 1.0 1.5
```

```r
# repeat u two times
w <- rep(u, times=2)
w
```

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

```r
## Scalar mult
2*u
```

```
## [1] 2 4 6 8
```

```r
## Vector Addition
u+v
```

```
## [1] 1.0 2.5 4.0 5.5
```

```r
## square every element of u
u^2
```

```
## [1]  1  4  9 16
```

```r
## multiply each element of u by the corresponding elements of v
u*v
```

```
## [1] 0 1 3 6
```

```r
## sum up every element of u
sum(u)
```

```
## [1] 10
```

```r
## subsetting
u
```

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

```r
u[1]        ## get the first element of u
```

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

```r
u[1:3]      ## get the first 3 elements of u
```

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

```r
u[c(2,4)]   ## get the 2nd and 4th elements of u
```

```
## [1] 2 4
```

```r
u[-4]       ## get all but the 4th element of u
```

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

```r
######### Matrices ##########

A <- rbind(u,v)  #row bind
A
```

```
##   [,1] [,2] [,3] [,4]
## u    1  2.0    3  4.0
## v    0  0.5    1  1.5
```

```r
B <- cbind(u,v)  #column bind
B
```

```
##      u   v
## [1,] 1 0.0
## [2,] 2 0.5
## [3,] 3 1.0
## [4,] 4 1.5
```

```r
## subsetting
A[1,1]         ## get the (1,1) element of A
```

```
## u
## 1
```

```r
A[1:2,1:3]     ## get the first two rows of A with the first three columns
```

```
##   [,1] [,2] [,3]
## u    1  2.0    3
## v    0  0.5    1
```

```r
A[,1:2]        ## get the 1st and 2nd columns of A (with all rows)
```

```
##   [,1] [,2]
## u    1  2.0
## v    0  0.5
```

```r
B[-3,]         ## get all but the 3rd row of B (with all columns)
```

```
##      u   v
## [1,] 1 0.0
## [2,] 2 0.5
## [3,] 4 1.5
```

```r
## transpose
t(A)
```

```
##      u   v
## [1,] 1 0.0
## [2,] 2 0.5
## [3,] 3 1.0
## [4,] 4 1.5
```

```r
## Matrix multiplication
C <- A%*%B  #A: 2*4, B: 4*2
C  #2*2
```

```
##    u    v
## u 30 10.0
## v 10  3.5
```

```r
# A%*%C  ## doesn't work b.c. A is (2x4) and C is (2x2)
C%*%A  ## does work
```

```
##   [,1]  [,2]  [,3]   [,4]
## u   30 65.00 100.0 135.00
## v   10 21.75  33.5  45.25
```

```r
## Element by element multiplication
C*C   ## is not the same as C%*%C
```

```
##     u      v
## u 900 100.00
## v 100  12.25
```

```r
C%*%C
```

```
##      u      v
## u 1000 335.00
## v  335 112.25
```

```r
#A*C   ## doesn't work, A and C must have the same dimensions

## Matrix Inverse
C.inv <- solve(C)
C.inv
```

```
##      u  v
## u  0.7 -2
## v -2.0  6
```

```r
C%*%C.inv
```

```
##              u v
## u 1.000000e+00 0
## v 8.881784e-16 1
```

```r
## Note: C^(-1) is not equal to the inverse matrix of C. it is the
# reciprocal of each element of C
C%*%C^(-1)    ## does not equal the identity
```

```
##           u        v
## u 2.0000000 5.857143
## v 0.6833333 2.000000
```

```r
####### Data Frames #########

## Data frames are basically the same as matrices

# Converting matrix B to a data frame
B
```

```
##      u   v
## [1,] 1 0.0
## [2,] 2 0.5
## [3,] 3 1.0
## [4,] 4 1.5
```

```r
B_df <- as.data.frame(B)

# Displaying the resulting data frame
print(B_df)
```

```
##   u   v
## 1 1 0.0
## 2 2 0.5
## 3 3 1.0
## 4 4 1.5
```

```r
mydata<-data.frame(x = c(1,4,3,2), y = c(2,4,2,4), z = c(3,7,2,3))
mydata
```

```
##   x y z
## 1 1 2 3
## 2 4 4 7
## 3 3 2 2
## 4 2 4 3
```

```r
## read data into a data frame from a webpage
ex.data <- read.table(file="https://math.unm.edu/~luyan/stat57918/seeds.txt",header=T)
ex.data
```

```
##    variety root  y  n
## 1        1    1 10 39
## 2        1    1 23 62
## 3        1    1 23 81
## 4        1    1 26 51
## 5        1    1 17 39
## 6        1    2  5  6
## 7        1    2 53 74
## 8        1    2 55 72
## 9        1    2 32 51
## 10       1    2 46 79
## 11       1    2 10 13
## 12       2    1  8 16
## 13       2    1 10 30
## 14       2    1  8 28
## 15       2    1 23 45
## 16       2    1  0  4
## 17       2    2  3 12
## 18       2    2 22 41
## 19       2    2 15 30
## 20       2    2 32 51
## 21       2    2  3  7
```

```r
## Read data from a local file
seeds <- read.csv("~/Documents/yan/teaching/stat579glmm/data/seeds.txt", sep="")
# header =True is the default
# read first 6 rows of the data
head(seeds)
```

```
##   variety root  y  n
## 1       1    1 10 39
## 2       1    1 23 62
## 3       1    1 23 81
## 4       1    1 26 51
## 5       1    1 17 39
## 6       1    2  5  6
```

```
## Import data with RStudio
# Use the ``Import Dataset'' dropdown from the ``Environment'' window (top right panel in RStudio).
# The import formats are grouped into 3 categories: text data, spreadsheet data, and statistical data.
# import data from webpage, Select ``From Text (readr)'', enter URL address, and click ``Import''.
# import data from local file
# Select ``From Text (base)'' or ``From Excel'', select a local file and click ``Import''

# read data by SDAResources
library(SDAResources)
agsrs[1:10,]
```

```
##               county state acres92 acres87 acres82 farms92 farms87 farms82
## 1      COFFEE COUNTY    AL  175209  179311  194509     760     842     944
## 2     COLBERT COUNTY    AL  138135  145104  161360     488     563     686
## 3       LAMAR COUNTY    AL   56102   59861   72334     299     362     447
## 4     MARENGO COUNTY    AL  199117  220526  231207     434     471     622
## 5      MARION COUNTY    AL   89228  105586  113618     566     658     748
## 6  TUSCALOOSA COUNTY    AL   96194  120542  134616     436     521     650
## 7    COLUMBIA COUNTY    AR   57253   66305   80909     320     411     477
## 8    FAULKNER COUNTY    AR  210692  223594  227593    1051    1103    1169
## 9  HOT SPRING COUNTY    AR   78498   80267   72175     419     526     512
## 10     MONROE COUNTY    AR  219444  234605  235409     278     306     369
##    largef92 largef87 largef82 smallf92 smallf87 smallf82 region
## 1        29       28       21       57       47       66      S
## 2        37       41       42       12       44       47      S
## 3         4        4        3       16       20       30      S
## 4        48       66       62       14       11       28      S
## 5         7        9        9       11       23       27      S
## 6        20       17       23       18       32       29      S
## 7         6        4        9       17       12       27      S
## 8        23       32       27       42       41       49      S
## 9         7        5        5       15       35       18      S
## 10       87       86       72        8        6        4      S
```

```
## Extract variables
ex.data2<-ex.data[,c(1,3)]  #extract variables variety and y
ex.data2
```

```
##    variety  y
## 1        1 10
## 2        1 23
## 3        1 23
## 4        1 26
## 5        1 17
## 6        1  5
## 7        1 53
## 8        1 55
## 9        1 32
## 10       1 46
## 11       1 10
## 12       2  8
## 13       2 10
## 14       2  8
```

```
## 15       2 23
## 16       2  0
## 17       2  3
## 18       2 22
## 19       2 15
## 20       2 32
## 21       2  3
```

```
ex.data2$y[ex.data2$variety=="1"]
```

```
##  [1] 10 23 23 26 17  5 53 55 32 46 10
```

```
ex.data2[ex.data2$y>10,]
```

```
##    variety  y
## 2        1 23
## 3        1 23
## 4        1 26
## 5        1 17
## 7        1 53
## 8        1 55
## 9        1 32
## 10       1 46
## 15       2 23
## 18       2 22
## 19       2 15
## 20       2 32
```

```
table(ex.data$variety)
```

```
##
##  1  2
## 11 10
```

```
## calculate mean according to factor level
tapply(ex.data$y,ex.data$variety,mean)
```

```
##        1        2
## 27.27273 12.40000
```

```
##fit anova model
myfit = aov(y~factor(variety)*factor(root),data=ex.data)
summary(myfit)
```

```
##                             Df Sum Sq Mean Sq F value Pr(>F)
## factor(variety)              1   1159  1158.7   5.741 0.0284 *
## factor(root)                 1    485   485.2   2.404 0.1394
## factor(variety):factor(root)  1     94    94.2   0.467 0.5036
## Residuals                   17   3431   201.8
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
stable<-summary(myfit)
library(xtable)
# latex code to derive anova table
print(xtable(stable), type="latex")
```

```
## % latex table generated in R 4.2.1 by xtable 1.8-4 package
## % Tue Jan 16 22:20:25 2024
## \begin{table}[ht]
## \centering
## \begin{tabular}{lrrrrr}
##   \hline
##  & Df & Sum Sq & Mean Sq & F value & Pr($>$F) \\
##   \hline
## factor(variety)              & 1 & 1158.66 & 1158.66 & 5.74 & 0.0284 \\
##   factor(root)                 & 1 & 485.24 & 485.24 & 2.40 & 0.1394 \\
##   factor(variety):factor(root) & 1 & 94.24 & 94.24 & 0.47 & 0.5036 \\
##   Residuals                    & 17 & 3431.10 & 201.83 &  &  \\
##    \hline
## \end{tabular}
## \end{table}
```

```r
######## Functions ##########

## You won't use these much in this class but
## they are very useful.

## Creating a function that adds two vectors together

my.add <- function(a,b){
  return(a + b)
}

my.add(u,v)
```

```
## [1] 1.0 2.5 4.0 5.5
```

```r
my.add2 <- function(a,b){  ## performs the same function as above
  n <- length(a)
  temp <- rep(0, n)
  for(i in 1:n){
    temp[i] <- a[i] + b[i]
  }
  return(temp)
}

my.add2(u,v)
```

```
## [1] 1.0 2.5 4.0 5.5
```