Stats 302/767 Lab1

Beatrix Jones

Week 1

Please use R markdown to prepare your assignments. R markdown will allow you to intersperse text, R code, and output, formating the latter two in a relatively friendly way. This lab has been prepared in R markdown; you can view the .Rmd file that produced the lab on Canvas. You will want to produce a word or pdf document to hand in (the later requires LaTex to be installed.)

For this lab, please download the data set possum.csv into a directory of your choice, and make that your working directory. To find your current working directory use the following command:

```
getwd()
```

To change your working directory use the following, replacing *pathname* with the path for your desired directory :

```
setwd("pathname")
```

Matrices in R

First, we will think about how R does matrix arithmetic. Frequently we will read in a data matrix, but if we create a matrix in R the default is to fill up the matrix by columns; there is an option to change this. We can also reverse rows and columns (take the *transpose*) with the function t().

```
x<-matrix(c(1:12), ncol=3)
x
```

```
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
## [3,]    3    7   11
## [4,]    4    8   12
```

```
y<-matrix(c(1:12), ncol=3, byrow=TRUE)
y
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
## [4,]   10   11   12
```

```
t(x)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
```

Note that * does element-wise multiplication; %*% does matrix multiplication. Remember that the number of columns in the first argument needs to equal the number of rows in the second argument. If this is not the case, you will get the error "non conformable arguments."

```
x*y
```

```
##      [,1] [,2] [,3]
## [1,]    1   10   27
## [2,]    8   30   60
## [3,]   21   56   99
## [4,]   40   88  144
```

```
t(x)%*%y
```

```
##      [,1] [,2] [,3]
## [1,]   70   80   90
## [2,]  158  184  210
## [3,]  246  288  330
```

If we want to create a diagonal matrix (for instance, an identity matrix), we can use the command diag in a couple of different ways. Try the following:

```
diag(1, 10)
diag(1:9)
```

The matrices we deal with most commonly in statistics are the data matrix and covariance/correlation matrices. If we are going to print one of these out and look at it, we should round; two decimal places is plenty for a correlation matrix.

```
read.csv('possum.csv')->possum
possum.cor<-cor(possum[,7:15])
round(possum.cor,2)
```

```
##          hdlngth skullw totlngth taill footlgth earconch   eye chest belly
## hdlngth     1.00   0.71     0.69  0.29       NA     0.12  0.35  0.63  0.56
## skullw      0.71   1.00     0.53  0.26       NA     0.00  0.32  0.63  0.45
## totlngth    0.69   0.53     1.00  0.57       NA     0.15  0.25  0.58  0.52
## taill       0.29   0.26     0.57  1.00       NA    -0.39  0.20  0.17  0.29
## footlgth      NA     NA       NA    NA        1       NA    NA    NA    NA
## earconch    0.12   0.00     0.15 -0.39       NA     1.00 -0.17  0.20  0.07
## eye         0.35   0.32     0.25  0.20       NA    -0.17  1.00  0.15  0.24
## chest       0.63   0.63     0.58  0.17       NA     0.20  0.15  1.00  0.61
## belly       0.56   0.45     0.52  0.29       NA     0.07  0.24  0.61  1.00
```

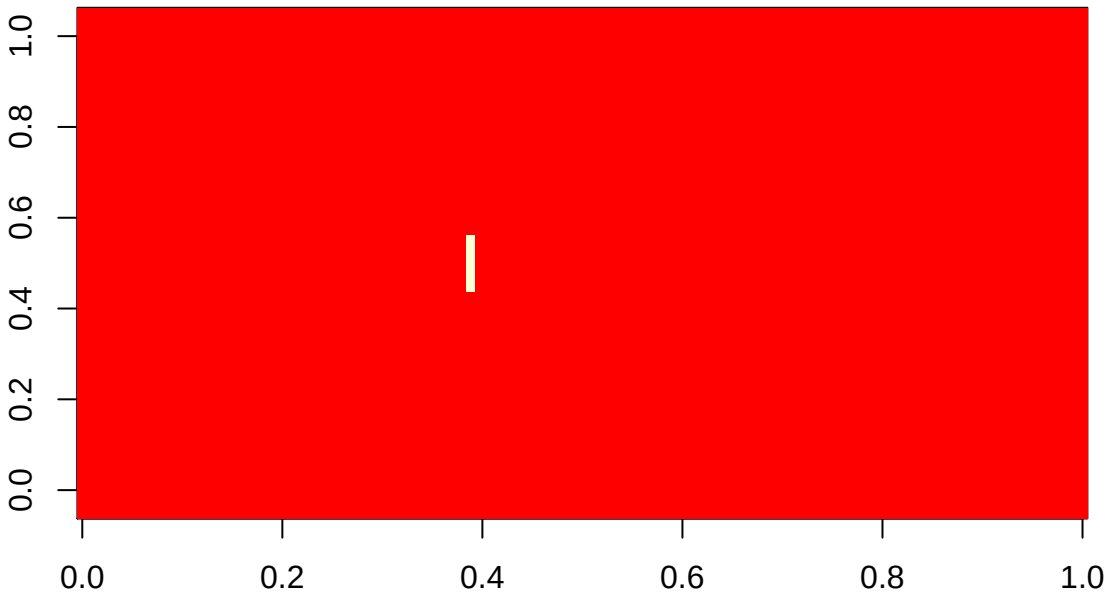
We see some NAs. This will be caused by a missing value in footlength. We can track it down with:

```
which(is.na(possum$footlgth))
```

```
## [1] 41
```

If this produces a long list of stuff, we can visualize how the missing values are structured with:

```
image(is.na(possum[,7:15]))
```



Note the x axis indexes the rows, with the first row on the far left, and the first column at the bottom.

Here, we see there is a single missing value. Other common patters are 1 variable with many missing values (a horizontal strip with lots of white), 1 observation with lots missing (a vertical strip with lots white), or all missing observations at the beginning or end (vertical strips at the far left or far right–usually due to blank rows in an input file).

Generally we need to think about whether missing values are "missing at random" before we remove them. If we are just looking at the correlation matrix, we can also specify options to deal with this. What is the difference between the following two commands?

```
round(cor(possum[,7:15], use='pairwise.complete.obs'),2)
round(cor(possum[,7:15], use='complete.obs'),2)
```

Note we can consider this via matrix subtraction (this is always done elementwise), looking at an extra decimal place to see any small changes.

```
round(cor(possum[,7:15], use='pairwise.complete.obs')-cor(possum[,7:15], use='complete.obs'),3)
```

```
##          hdlngth skullw totlngth  taill footlgth earconch    eye  chest
## hdlngth    0.000 -0.005    0.003  0.001        0    0.019 -0.013 -0.004
## skullw    -0.005  0.000   -0.003  0.000        0   -0.001 -0.001  0.000
## totlngth   0.003 -0.003    0.000  0.000        0    0.015 -0.010 -0.003
## taill      0.001  0.000    0.000  0.000        0    0.012 -0.003  0.000
## footlgth   0.000  0.000    0.000  0.000        0    0.000  0.000  0.000
## earconch   0.019 -0.001    0.015  0.012        0    0.000 -0.012 -0.004
## eye       -0.013 -0.001   -0.010 -0.003        0   -0.012  0.000  0.000
## chest     -0.004  0.000   -0.003  0.000        0   -0.004  0.000  0.000
## belly      0.003 -0.002    0.003  0.001        0    0.013 -0.007 -0.002
##            belly
## hdlngth    0.003
## skullw    -0.002
## totlngth   0.003
## taill      0.001
## footlgth   0.000
## earconch   0.013
## eye       -0.007
## chest     -0.002
```

```
## belly     0.000
```

If we want to invert a matrix, we use the command "solve." Remember not all matrices are invertible–they must be of full rank. Try solve(x) using the x matrix we created above and see what happens.

```
possum.cor<-cor(possum[,7:15], use='complete.obs')
ipossum<-solve(possum.cor)
ipossum
```

```
##             hdlngth      skullw    totlngth       taill     footlgth
## hdlngth   3.2111078 -1.20899690 -1.27627656  0.40148581 -0.240137272
## skullw   -1.2089969  2.41794648  0.07082957 -0.04537851 -0.186615214
## totlngth -1.2762766  0.07082957  3.69297816 -1.86450075 -0.857123295
## taill     0.4014858 -0.04537851 -1.86450075  2.38393405  0.086419475
## footlgth -0.2401373 -0.18661521 -0.85712330  0.08641948  4.017413572
## earconch  0.2167459  0.37176670 -0.35628012  1.05295396 -2.888366930
## eye      -0.3471049 -0.20784627 -0.09653001  0.03907748 -0.009926576
## chest    -0.2415155 -0.84191891 -0.55388277  0.31728880 -0.338225502
## belly    -0.4146487  0.15559639 -0.03865658 -0.22613098 -0.217022990
##              earconch          eye       chest       belly
## hdlngth   0.21674587 -0.347104877 -0.24151552 -0.41464872
## skullw    0.37176670 -0.207846274 -0.84191891  0.15559639
## totlngth -0.35628012 -0.096530010 -0.55388277 -0.03865658
## taill     1.05295396  0.039077482  0.31728880 -0.22613098
## footlgth -2.88836693 -0.009926576 -0.33822550 -0.21702299
## earconch  3.71914830  0.216116456  0.06189932  0.16981076
## eye       0.21611646  1.237525080  0.26256013 -0.14240632
## chest     0.06189932  0.262560128  2.53375467 -0.79545499
## belly     0.16981076 -0.142406318 -0.79545499  1.82244016
```

```
possum.cor%*%ipossum
```

```
##                hdlngth        skullw      totlngth         taill
## hdlngth   1.000000e+00 -4.163336e-17  9.714451e-17  2.775558e-17
## skullw   -1.942890e-16  1.000000e+00  1.526557e-16  1.387779e-17
## totlngth -2.498002e-16 -1.387779e-17  1.000000e+00 -6.938894e-17
## taill    -1.665335e-16  4.857226e-17  5.620504e-16  1.000000e+00
## footlgth -1.387779e-16  2.081668e-17  1.925543e-16  1.387779e-17
## earconch -2.428613e-17 -5.898060e-17  3.859760e-17  2.949030e-17
## eye      -2.775558e-16  6.938894e-17 -2.602085e-17  4.857226e-17
## chest    -1.110223e-16  1.526557e-16  3.851086e-16 -2.775558e-17
## belly    -1.665335e-16  2.220446e-16  3.191891e-16 -8.326673e-17
##               footlgth      earconch           eye         chest
## hdlngth   1.387779e-17  2.775558e-17  1.387779e-17  1.110223e-16
## skullw    1.249001e-16 -1.249001e-16  1.526557e-16  0.000000e+00
## totlngth  3.191891e-16 -2.775558e-17  4.163336e-17  5.551115e-17
## taill     1.110223e-16 -1.804112e-16  4.163336e-17 -2.775558e-17
## footlgth  1.000000e+00 -1.249001e-16 -3.469447e-17 -2.775558e-17
## earconch -3.382711e-16  1.000000e+00  5.204170e-18  2.775558e-17
## eye       1.249001e-16 -5.551115e-17  1.000000e+00  0.000000e+00
## chest     1.387779e-16 -1.665335e-16 -1.387779e-17  1.000000e+00
## belly    -5.551115e-17  0.000000e+00  5.551115e-17 -1.110223e-16
##                  belly
## hdlngth   0.000000e+00
## skullw    0.000000e+00
## totlngth  0.000000e+00
```

```
## taill     0.000000e+00
## footlgth  0.000000e+00
## earconch -1.387779e-17
## eye       5.551115e-17
## chest     0.000000e+00
## belly     1.000000e+00
```

We should get the identity... but there are a few numerical leftovers. Rounding (even with lots of decimal places) will get rid of this.

```
round(possum.cor%*%ipossum,10)
```

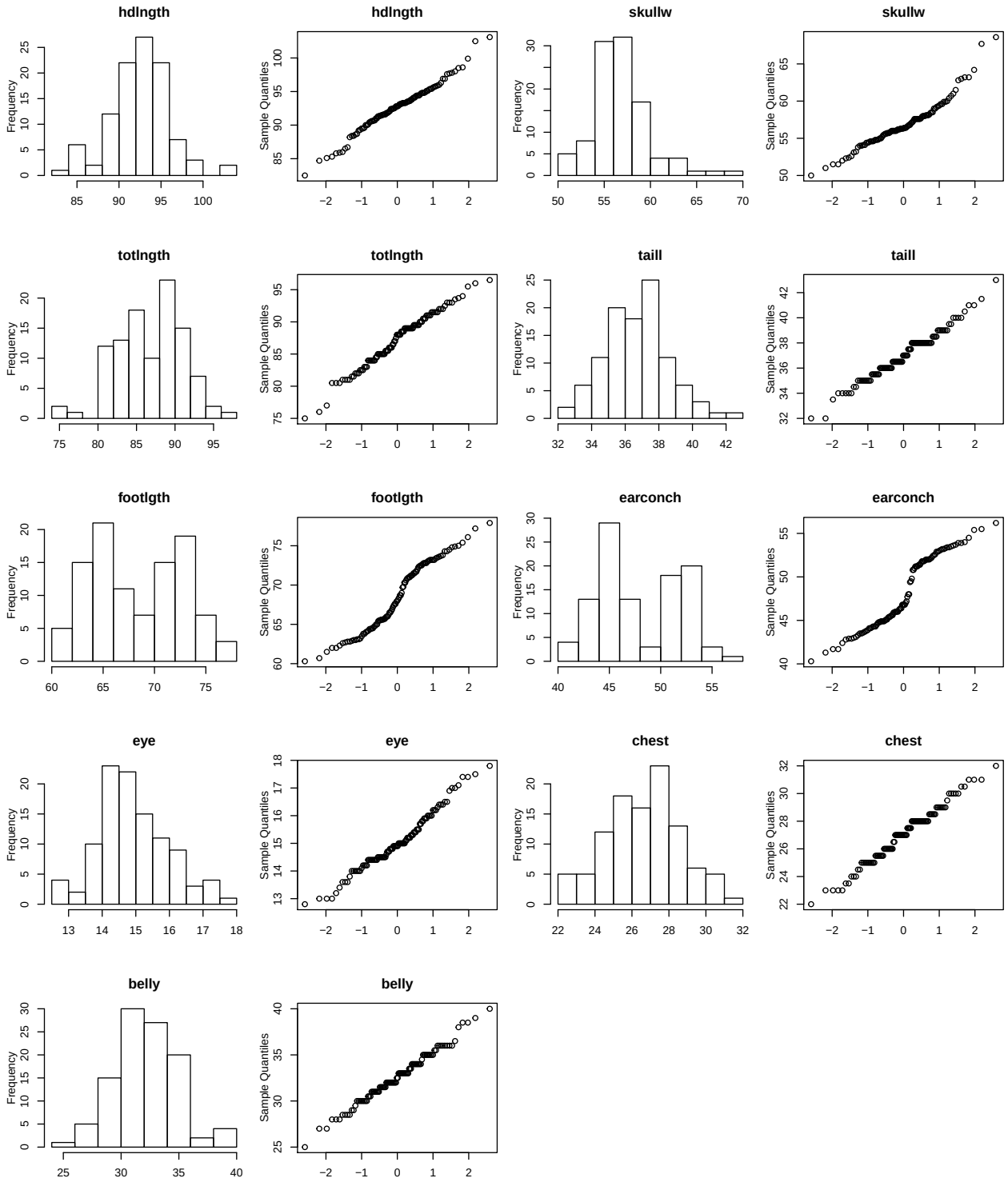
```
##          hdlngth skullw totlngth taill footlgth earconch eye chest belly
## hdlngth        1      0        0     0        0        0   0     0     0
## skullw         0      1        0     0        0        0   0     0     0
## totlngth       0      0        1     0        0        0   0     0     0
## taill          0      0        0     1        0        0   0     0     0
## footlgth       0      0        0     0        1        0   0     0     0
## earconch       0      0        0     0        0        1   0     0     0
## eye            0      0        0     0        0        0   1     0     0
## chest          0      0        0     0        0        0   0     1     0
## belly          0      0        0     0        0        0   0     0     1
```

For loops and summary statistics.

When dealing with multivariate datasets, for loops are your friend. For instance, consider the possum morphology measurements. Suppose we want histograms and qq plots of each variable (14 plots in all). Use "?par" to read about the arguments to the par command. These are setting up the multiplot array and compressing the margins a little bit to make the most of the space. (You'll notice the plots in this section are a little bit bigger. Have a peek at the .Rmd file to see how this is done.)

```
morph<-possum[,7:15]

par(mfrow=c(5,4), mar=c(4,4,3,0), mgp=c(2,1,0))
for(i in 1:9){
hist(morph[,i], main=names(morph)[i], xlab='')
  qqnorm(morph[,i], main=names(morph)[i], xlab='')
}
```



To complute summary statistics for each column, use the apply command. For instance, to compute the standard deviation for the numeric variables:

```
apply(morph,2,sd, na.rm=TRUE)
```

```
##  hdlngth   skullw totlngth    taill footlgth earconch      eye    chest
## 3.573349 3.113426 4.310549 1.959518 4.395306 4.109380 1.050374 2.045597
##    belly
```

```
## 2.761949
```

```
## or save in an object
possum.sd<-apply(morph,2,sd, na.rm=TRUE)
```

The second argument "2" indicates the function is to be applied to the columns; we can do the same for rows with "1". Note we can include arguments for our summary command–here, na.rm=TRUE to ignore the 41st observation when computing the sd of footlength. We can use any summary we want other than sd: mean, median, sum, and so on.

Visualizing the Correlation matrix.

In Monday's lecture, we did a visualization of the correlations of the Mutual fund data. Let's produce that for the possum data.

```
library(ggplot2)
library(reshape2)
PcorMelt<-melt(possum.cor)  # have a look---what has this done?
names(PcorMelt)<-c("Measurement1", "Measurement2", "Correlation")
ggplot(data=PcorMelt, aes(x=Measurement1, y=Measurement2, fill=Correlation)) + geom_tile()
```