Multivariate Statistics Week 02: Describing and displaying multivariate data

Noah Silbert

September 1, 2017

Univariate expected values and means

The expected value of a random variable X is denoted $\mathbb{E}[X]$. For a continuous random variable:

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x f(x) dx$$

For a discrete random variable:

$$\mathbb{E}[X] = \sum_{i=-\infty}^{\infty} x_i p(x_i)$$

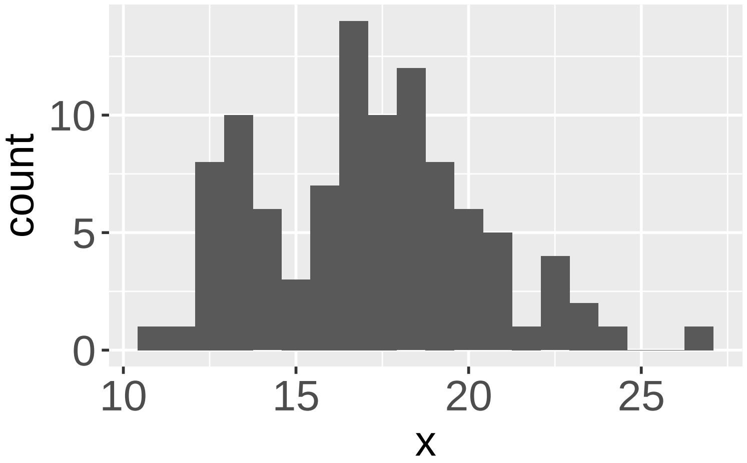
With a sample of a univariate random variable $x_i, i \in 1, 2, \ldots, N$, we approximate $p(x)$ with $\frac{1}{N}$, and calculate the (sample) mean as $\bar{x} = \frac{x_1 + x_2 + \cdots + x_n}{N} = \frac{1}{N} \sum_{i=1}^{N} x_i$.

Means in R

We can use R to generate a sample of a normally distributed random variable, then use linear algebra and the **j** vector to calculate this easily (we could also just use the function `mean()`):

```
N = 100
x = matrix(rnorm(N,17,3.2),nrow=N,ncol=1)
j = matrix(1,nrow=1,ncol=N)
x.bar = (1/N)*j%*%x
```

Visualization



```
x.bar
```

```
##           [,1]
## [1,] 17.12362
```

Multivariate means

With multivariate data, each observation is a vector rather than a single scalar. Hence, instead of a single expected value, we get a vector of expected values: one for each dimension of the multivariate random variable.

Multivariate data is typically organized with N observations in the rows and P variables in columns. So, for example, if we measure the height and weight ($P = 2$) of 100 people, we would have a 100×2 matrix.

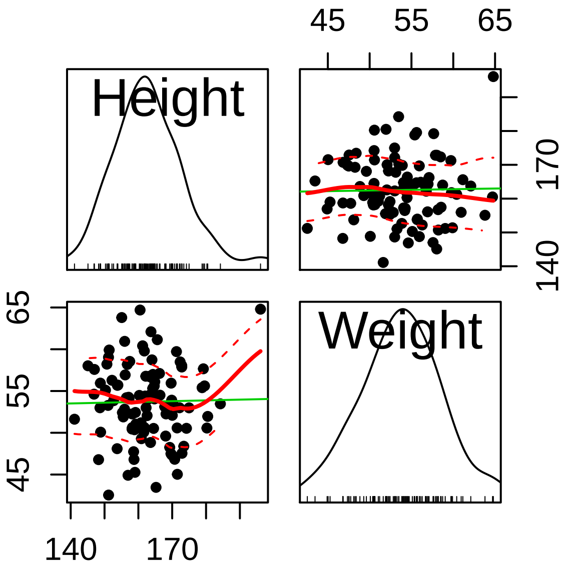
Multivariate data in R

```r
X = cbind(rnorm(N,162,10),rnorm(N,54,5))
colnames(X) = list("Height","Weight")
```

There are a few packages and functions available for making a matrix of scatterplots. The best one I've found so far is the `scatterplotMatrix()` function from the `car` (Companion to Applied Regression) package. We can install this package and invoke it like this (without the # before the first line):

```r
#install.packages("car")
library(car)
```

```r
scatterplotMatrix(X,cex=.75,pch=19,cex.axis=1.25)
```



Multivariate means in R

In this case, the data are two-dimensional, so we will have a length 2 mean vector (one mean for height, one for weight). We can use the **j** matrix to calculate this, too (`N` and `j` were defined above; we could also use `colMeans()`):

```
X.bar = (1/N)*j%*%X
X.bar
```

```
##          Height   Weight
## [1,] 162.5496 53.71652
```

Variance

The variance of a univariate random variable is given by:

$$\begin{aligned}\mathbb{E}[(x-\mu)^2] &= \int_{-\infty}^{\infty}(x-\mu)^2 f(x)dx && \text{continuous}\\ &= \sum_{i=-\infty}^{\infty}(x_i-\mu)^2 p(x) && \text{discrete}\end{aligned}$$

That is, the variance is the expected value of the squared deviation of the data from the mean. The standard deviation is the square root of the variance.

For a sample, we can estimate the variance as:

$$\mathrm{Var}(x) = \frac{1}{N-1}\sum_{i=1}^{N}(x_i-\bar{x})^2$$

Variance formulas, bias, MLE

Note that there are two formulas for the variance, one with the fraction $\frac{1}{N}$, and one with the fraction $\frac{1}{N-1}$.

For normally distributed data, the former is the maximum likelihood estimate, but it's biased, whereas the latter is unbiased, but it is not the maximum likelihood estimate.

The bias of a statistical estimator θ (e.g., a mean, variance, etc. . .) is $\mathbb{E}[\hat{\theta} - \theta]$.

We will return to the statistical concept of maximum likelihood estimation later.

Covariance

With multivariate data, we have variances for each variable, but we also have *covariances* between variables. The covariance for two random variables X and Y is the expected value of the cross-product of their deviations:

$$\mathrm{Cov}(X, Y) = \sigma_{xy} = \mathbb{E}[(X - \mathbb{E}(X))(Y - \mathbb{E}(Y))]$$

Note that this just reduces to the definition of variance when $X = Y$:

$$\begin{aligned}\mathrm{Cov}(X, X) = \sigma_{xx} &= \mathbb{E}[(X - \mathbb{E}(X))(X - \mathbb{E}(X))] \\ &= \mathbb{E}[(X - \mathbb{E}(X))^2]\end{aligned}$$

Sample covariance

A useful linear algebra fact: $(\mathbf{ABC})^T = \mathbf{C}^T\mathbf{B}^T\mathbf{A}^T$

A useful linear algebra definition: A matrix $\mathbf{A}$ is *idempotent* if $\mathbf{AA} = \mathbf{A}$.

The sample covariance matrix for a data matrix X can be expressed in linear algebra notation as follows ($\mathbf{I}$ and $\mathbf{J}$ are $N \times N$):

$$\begin{aligned}
\Sigma &= \frac{1}{N-1}\left(\mathbf{X} - \frac{1}{N}\mathbf{JX}\right)^T\left(\mathbf{X} - \frac{1}{N}\mathbf{JX}\right) \\
&= \frac{1}{N-1}\mathbf{X}^T\left(\mathbf{I} - \frac{1}{N}\mathbf{J}\right)^T\left(\mathbf{I} - \frac{1}{N}\mathbf{J}\right)\mathbf{X} \\
&= \frac{1}{N-1}\mathbf{X}^T\left(\mathbf{I} - \frac{1}{N}\mathbf{J}\right)\mathbf{X}
\end{aligned}$$

Variance and covariance in R

We can use the formula above to calculate a covariance matrix:

```
J = matrix(1,N,N)
I = diag(1,N,N)
S = (1/(N-1))*t(X)%*%(I-(1/N)*J)%*%X
S
```

```
##             Height     Weight
## Height 88.0559814  0.8044789
## Weight  0.8044789 20.9168399
```

Covariance and correlation

The correlation between two variables x and y is defined as

$$\rho_{xy} = \frac{\sigma_{xy}}{\sqrt{\sigma_{xx}\sigma_{yy}}}$$

We can calculate the correlation matrix using linear algebra by by pre- and post-multiplying the covariance matrix by a diagonal matrix containing the reciprocals of the standard deviations:

```
W = diag(1/sqrt(diag(S)),2,2)
```

The inner call to `diag()` extracts the diagonal values from `S`, the square roots of these are then put into the denominators of fractions (with numerator 1), after which the outer call to `diag()` puts these reciprocal variances into the diagonal of a new matrix (i.e., $\mathbf{W}_{ij} = \frac{1}{\sqrt{\sigma_{ij}}}$ if $i = j$, 0 otherwise).

Calculating the correlation matrix

If we pre- and post-multiply **S** by **W**, we get the correlation matrix:

```
R = W%*%S%*%W
R
```

```
##              [,1]       [,2]
## [1,] 1.00000000 0.01874508
## [2,] 0.01874508 1.00000000
```

We could also use the functions `cov()` and `cor()` to get covariance and correlation matrices, respectively, but it's important to understand the linear algebra of this stuff for many of the topics we'll cover later on.

for loops

We could also use `for` loops to calculate the correlation matrix. We typically wouldn't do so, since we know how to use linear algebra and basic R functions, but `for` loops are very useful, so we'll introduce them here.

A `for` loop in R looks like this:

```r
for(i in 1:5){
  # code that does something interesting at step i
}
```

We can write a `for` loop within a `for` loop (within a `for` loop, etc...) as necessary:

```r
for(i in 1:5){
  for(j in 1:8){
    # code that does something at step i, j
  }
}
```

Calculate correlation with a for loop

```
nrc = dim(S) # get dimensions of S
nr = nrc[1] # number of rows
nc = nrc[2] # number of columns
M = matrix(0,nr,nc) # initialize a matrix of zeros
for(ri in 1:nr){ # loop through rows
  for(ci in 1:nc){ # loop through columns
    M[ri,ci] = S[ri,ci]/sqrt(S[ri,ri]*S[ci,ci])
  }
}
M
```

```
##             [,1]       [,2]
## [1,] 1.00000000 0.01874508
## [2,] 0.01874508 1.00000000
```

Some points about R

A few things in the code on the previous slide are worth noting.

First, instead of just using the number 2, we extracted the dimensions of `S`. Using (properties of) defined variables makes your code more robust. With code like this, we could calculate a new, larger `S`, and the code above would still work. If we had hard-coded 2 rows and 2 columns, it wouldn't.

Second, the index variables `ri` and `ci` iterate through vectors (`1:nr` and `1:nc`). `ri` and `ci` are assigned the values in the vectors one by one. We need to be careful to define the vectors appropriately and not to overwrite `ri` or `ci` with the code inside the loop(s).

Finally, note that we indexed the rows and columns of our matrices `M` and `S` using square brackets and `ri` and `ci` (e.g., the `ri`th row and `ci`th column of `M` is given by `M[ri,ci]`). You can get a whole row or column with `M[ri,]` and `M[,ci]`, respectively.

Plotting multivariate data in R

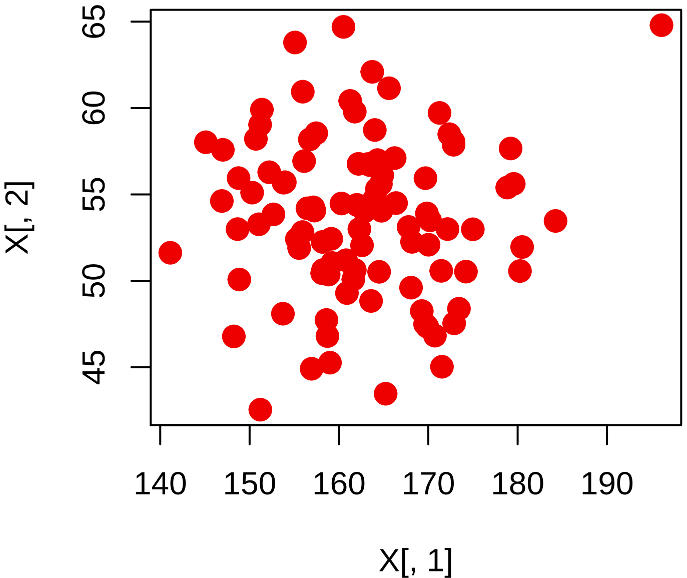
There are a number of ways to visualize multivariate data, but the most useful, straightforward way to do so is, in my opinion, with scatterplots. We saw a matrix of scatterplots above, but we'll take a few steps back and cover some more visualization issues here.

We can make a simple bivariate scatterplot in R using the `plot()` function. This function has two required arguments: two data vectors. The first will be on the `x` axis, the second on the `y` axis.

You can also specify a number of other properties, such as the type of symbol to be plotted (using the argument `pch`), the type of line (if you're using lines, using `lty`), the title and axis labels, and lots of other things. We'll cover some of these over the course of the semester.

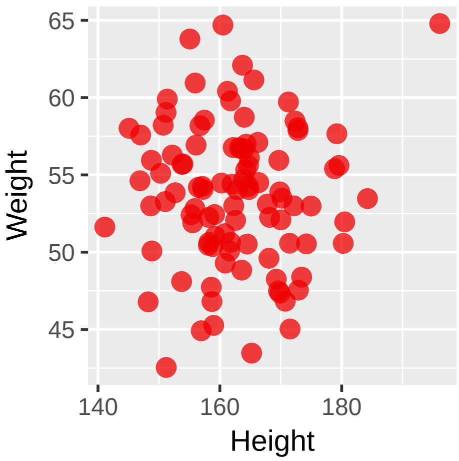
Example with base graphics

```r
plot(X[,1],X[,2],pch=19,col="red2",cex=1.5)
```



Example with ggplot2

```r
X.df = data.frame(X)
ggplot(X.df,aes(x=Height,y=Weight,color=I("red2"))) +
  geom_point(size=3,alpha=.75)
```

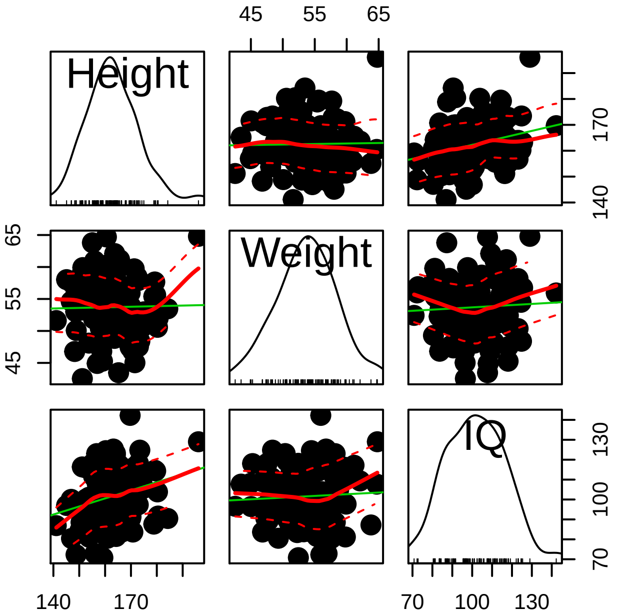


Plotting multivariate data in R (2)

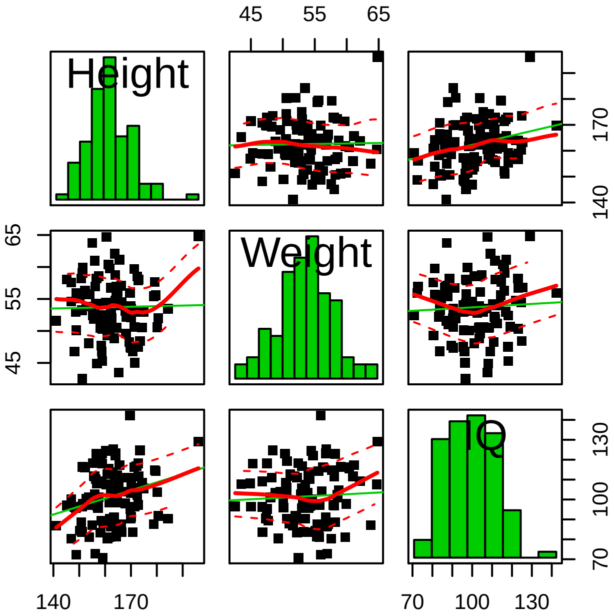
If you have more than two variables, a single bivariate scatterplot isn't enough. In addition, it is useful to also visualize the distribution of each variable on its own. This is what we saw above with the `scatterplotMatrix()` function above. We'll explore some of its features again here.

```r
Y = cbind(X,rnorm(N,100,15))
colnames(Y) = list("Height","Weight","IQ")
```

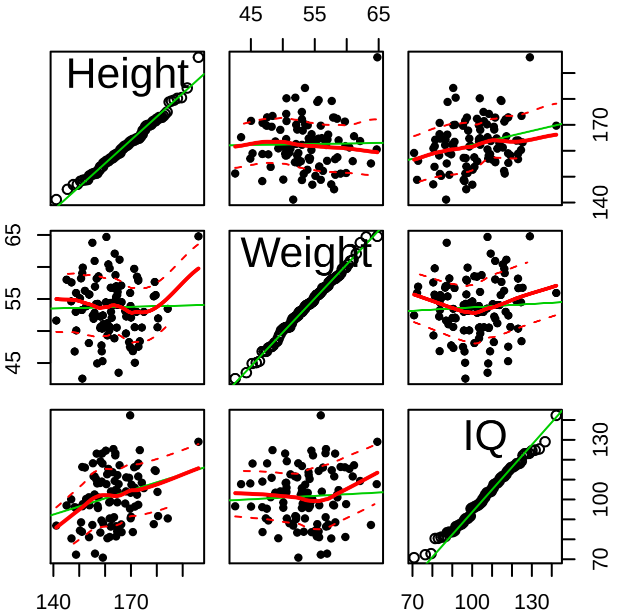
```
scatterplotMatrix(Y,pch=19,cex=2)
```



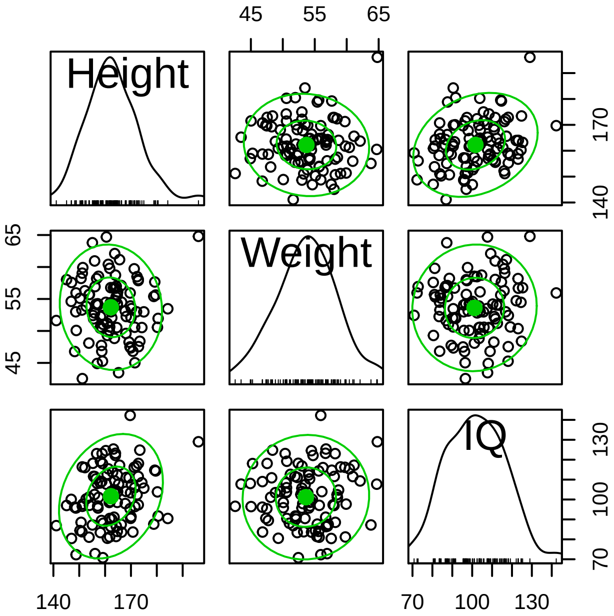
```
scatterplotMatrix(Y,pch=15,diagonal="histogram")
```



```
scatterplotMatrix(Y,pch=20,diagonal="qqplot")
```



```
scatterplotMatrix(Y,ellipse=T,smoother=F,reg.line=F)
```



A few other things in R & RStudio

If you need help with a function in RStudio, just type a `?` before it (e.g., `?scatterplotMatrix`). Be warned, though, that R help files can be remarkably unhelpful.

Given that, there are some useful R resources online. Quick-R gives good (though sometimes slightly outdated) information on a lot of basic R. You can also often find a lot of good questions and answers on StackOverflow (which also has a lot of questions and answers about other programming languages).

If you want to save a figure to a file, there are a number of functions that will do that. For example, you can call `png()`, `pdf()`, `jpeg()`, `bmp()`, or `tiff()` prior to your figure-generation code, then call `dev.off()` to turn off the graphics device your code created.

Saving a figure to a file

```
png('scatterplot_matrix.png',width=750,heigh=750)
scatterplotMatrix(Y,ellipse=T,pch=17,cex=2,
                  col=c("blue3","red3","orange3"),
                  reg.line=F,cex.axis=2)
dev.off()
```

```
## pdf
##   2
```