Multivariate Statistics Week 01: R, RStudio, and Linear Algebra

Noah Silbert

August 25, 2017

R & RStudio

R is a statistical programming language, and RStudio is (among other things) a development environment that makes using R very easy. R is useful in large part because it is free and has numerous free extensions (packages) that users have developed.

In RStudio, you can write scripts, execute code, see what variables and functions you have defined, look at your command history, see plots, and various other things.

R Basics

Variable types

- numeric, logical, character

Data types

- scalar, vector, matrix, array
- factor
- list, data frame

R Basics: Variable assignment

Assign a value to a variable with = or <-. Expressions that do not assign value(s) to a variable print the output to the R console:

```
# assign scalar, numeric values to variables x, y
x = 5
y <- 10
x + y
```

```
## [1] 15
```

You can think of the variables `x` and `y` as containers with (in this case) 5 and 10 inside.

R Basics: Vectors

A vector is a sequence of elements (typically numbers).

You can create a vector with `c()` ("combine"), `seq()`, `rep()`, or `N:M` (where N is the first value in the vector, and M is the last).

```
v0 = c(1,3,5,3,1)
v1 = seq(from=5,to=15,by=5)
v2 = rep(x=2,times=3)
v3 = 3:7
```

Note that `c()`, `seq()`, and `rep()` are functions. Functions (usually) take input arguments and (usually) return outputs of some sort. You can specify the input arguments by name or position. Specifying by name takes up more space, but it makes your code easier to read and understand.

R Basics: Matrices

A matrix is a 2D array of elements (typically numbers).

You can create a matrix with the function `matrix()`. When creating a matrix, you can (but do not have to) give it input arguments for `data`, `nrow`, `ncol`, `byrow`, and `dimnames`.

```
m1 = matrix(data=1:12,nrow=3,byrow=T)
m1
```

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

R Basics: More Matrices

`byrow` tells R to fill in the values row by row (if `True`) or column by column (if `False`).

```
m1 = matrix(data=1:12,nrow=3,byrow=T)
m1
```

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

```
m2 = matrix(data=1:12,nrow=3,byrow=F)
m2
```

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

R Basics: Variable equality, Boolean variables

Check for equality of variables with ==. The result of == is either `True` or `False`.

```
# x = 5, y = 10
x == y
```

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

```
x == y-5
```

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

R Basics: Matrix equality

Comparing two matrices with == checks for equality element-by-element, returning a matrix of Booleans.

```
A = matrix(1:9,nrow=3,byrow=T)
B = matrix(1:9,nrow=3)
A == B
```

```
##       [,1]  [,2]  [,3]
## [1,]  TRUE FALSE FALSE
## [2,] FALSE  TRUE FALSE
## [3,] FALSE FALSE  TRUE
```

R Basics: Matrix transposition

The transpose of a matrix $\mathbf{A}$ is denoted $\mathbf{A}^T$ or $\mathbf{A}'$, such that $a_{ij}^T = a_{ji}$.

Transpose a matrix in R with the function `t()`.

```
C = matrix(1:4,nrow=2,byrow=T)
D = matrix(1:4,nrow=2,byrow=F)
C == D
```

```
##       [,1]  [,2]
## [1,]  TRUE FALSE
## [2,] FALSE  TRUE
```

```
C == t(D)
```

```
##      [,1] [,2]
## [1,] TRUE TRUE
## [2,] TRUE TRUE
```

R Basics: Matrix symmetry

A matrix **M** is symmetric if $m_{ij} = m_{ji}^T$ for all i, j.

```
M = matrix(c(10,2,3,2,20,4,3,4,30),nrow=3)
M
```

```
##      [,1] [,2] [,3]
## [1,]   10    2    3
## [2,]    2   20    4
## [3,]    3    4   30
```

```
M == t(M)
```

```
##      [,1] [,2] [,3]
## [1,] TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE
## [3,] TRUE TRUE TRUE
```

R Basics: Diagonal matrices

Create a diagonal matrix or extract the diagonal from a matrix with `diag()`.

```
# create a matrix of zeros with 1, 2, 3 on the diagonal
diag(1:3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    2    0
## [3,]    0    0    3
```

```
# extract the diagonal elements of M
diag(M)
```

```
## [1] 10 20 30
```

R Basics: Matrix addition

The sum of two matrices **A** and **B** is the matrix **C** with elements equal to the sums of the elements of the original matrices, i.e., $c_{ij} = a_{ij} + b_{ij}$.

```
A = matrix(1:6,nrow=3,byrow=T)
B = matrix(1,3,2)
C = A + B
C
```

```
##      [,1] [,2]
## [1,]    2    3
## [2,]    4    5
## [3,]    6    7
```

R Basics: Matrix multiplication

The number of columns of the left matrix must be equal to the number of rows in the right matrix. So, e.g., you can multiply a $m \times n$ matrix $\mathbf{A}$ and a $n \times p$ matrix $\mathbf{B}$ to get $\mathbf{AB}$, but you $\mathbf{BA}$ is not legal unless $m = p$.

If $\mathbf{C} = \mathbf{AB}$, then $\mathbf{C}$ is a $m \times p$ matrix with

$$\begin{aligned} c_{ij} &= a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj} \\ &= \sum_{k=1}^{n} a_{ik}b_{kj} \end{aligned}$$

Mathematically, a length n vector is just a $n \times 1$ or $1 \times n$ matrix. Unless otherwise specified, the default assumption is that a vector is a column vector (i.e., $n \times 1$).

R Basics: Matrix multiplication

Multiply matrices (and/or vectors) in R with %*%. Note that, in R, vectors are *not* $n \times 1$ or $1 \times n$ matrices. They are different kinds of data containers. This is annoying (to me, anyway), but we just have to deal with it.

```
v = rep(1,3)
X = matrix(1:12,nrow=3,byrow=F)
v %*% X
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    6   15   24   33
```

R Basics: I, the identity matrix

The identity matrix **I** is the diagonal matrix with ones on the diagonal. If you multiply a matrix **M** by **I**, you just get **M** back.

```
# M = matrix(c(10,2,3,2,20,4,3,4,30),nrow=3)
I = diag(1,3,3)
I%*%M
```

```
##      [,1] [,2] [,3]
## [1,]   10    2    3
## [2,]    2   20    4
## [3,]    3    4   30
```

The identity matrix is the linear algebra equivalent of multiplying by 1. It can be very useful for factoring expressions, e.g., $\mathbf{A} - \mathbf{AB} = \mathbf{A}(\mathbf{I} - \mathbf{B})$.

R Basics: J, the matrix of ones

The matrix of ones is often called **j** or **J** (depending on if it's a vector or a matrix). It's useful for doing things like calculating means of matrix columns or rows (among other things).

```
N = matrix(1:12,nrow=3)
J = matrix(1,nrow=1,ncol=3)
(1/3)*J%*%N
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2    5    8   11
```

R Basics: Matrix inverse

If a matrix $\mathbf{A}$ is square and of full rank (i.e., all rows and/or columns are linearly independent), then there is an inverse matrix $\mathbf{A}^{-1}$ such that $\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$.

```
Mi = solve(M)
M%*%Mi
```

```
##                  [,1]          [,2] [,3]
## [1,]  1.000000e+00 -3.469447e-18    0
## [2,] -6.938894e-18  1.000000e+00    0
## [3,] -1.110223e-16  5.551115e-17    1
```

```
round(M%*%Mi) # rounding to save space
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```

R Basics: Trace and determinant

The trace of a matrix is the sum of its diagonal elements. The determinant is not very intuitive to define in general, but it shows up a lot in multivariate statistics. It's easy to calculate in R.

```
# M = matrix(c(10,2,3,2,20,4,3,4,30),nrow=3)
sum(diag(M))
```

```
## [1] 60
```

```
det(M)
```

```
## [1] 5588
```

It's jumping ahead by a few weeks, but the determinant of a covariance matrix is 'generalized variance'.

R Basics: Scalar x vector or matrix multiplication

If you multiply a vector or matrix by a scalar, the scalar is multiplied by each element of the vector or matrix. That is, with scalar a and vector $\mathbf{v}$, $a\mathbf{v} = \mathbf{w}$ such that $w_i = av_i$.

```
a = 10
v = 1:5
# note the use of *, rather than %*%
a*v
```

```
## [1] 10 20 30 40 50
```

R Basics: Linear combinations, linear independence

A linear combination of vectors is the sum of a set of vectors multiplied by scalars. For example, $\mathbf{v}$ here is the linear combination of three other vectors:

$$\mathbf{v} = a\mathbf{w} + b\mathbf{x} + c\mathbf{y}$$

If $\mathbf{d} = [a, b, c]^T$ and $\mathbf{Z} = [\mathbf{w} \quad \mathbf{x} \quad \mathbf{y}]$, then $\mathbf{v} = \mathbf{Z}\mathbf{d}$.

A set of vectors are linearly independent if none of them can be expressed as a linear combination of the others.

R Basics: Random number generation

Generating random numbers can be very useful for simulating data and exploring how various analysis techniques work. R has a number of random number generation functions. Note the use of `rpois` to generate Poisson random variates and `rnorm` to generate normal random variates.

```
N = 100
x = rpois(N,lambda=20)
X = cbind(1,x)
b = c(-17,7.5)
e = rnorm(N,mean=0,sd=25)
y = X %*% b + e
df = data.frame(x=x,y=y)
```

Data frames are useful data containers. We'll come back to them soon, and we'll use them quite a bit.

R Basics: Intalling and using packages

You can install a pacakges (e.g., `ggplot2`) by calling `install.packages(ggplot2)`. Once it is installed, you can invoke it for use by calling `library(ggplot2)`.

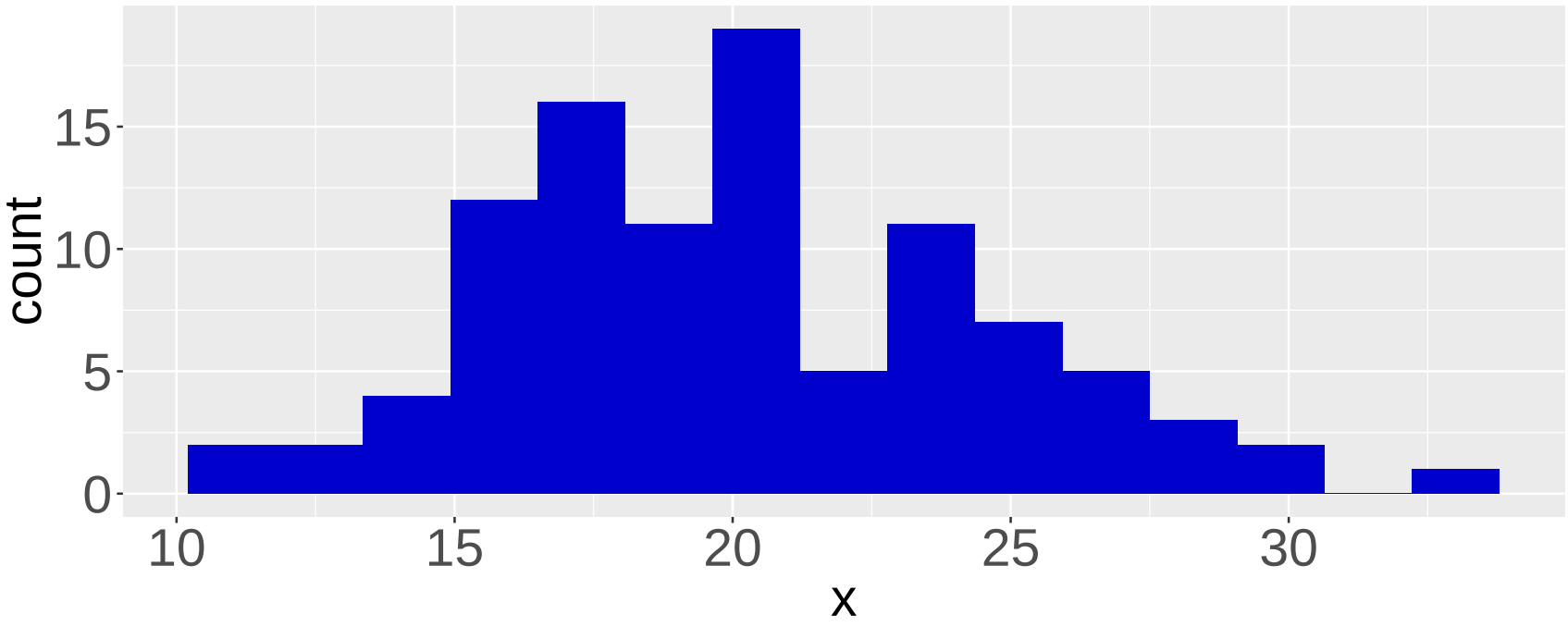
```
library(ggplot2) # already installed on my computer
```

`ggplot2` is a graphics package that is... mostly good, but frustrating at times. The basic graphical capabilities of R (Base Graphics) are fine, but it can take a lot of code to get a figure to look the way you want it to. `ggplot2` can be useful for making reasonably nice looking figures quickly, but it can take some getting used to.

Anyway, all the packages available on the comprehensive R archive network are available for free to install and use.

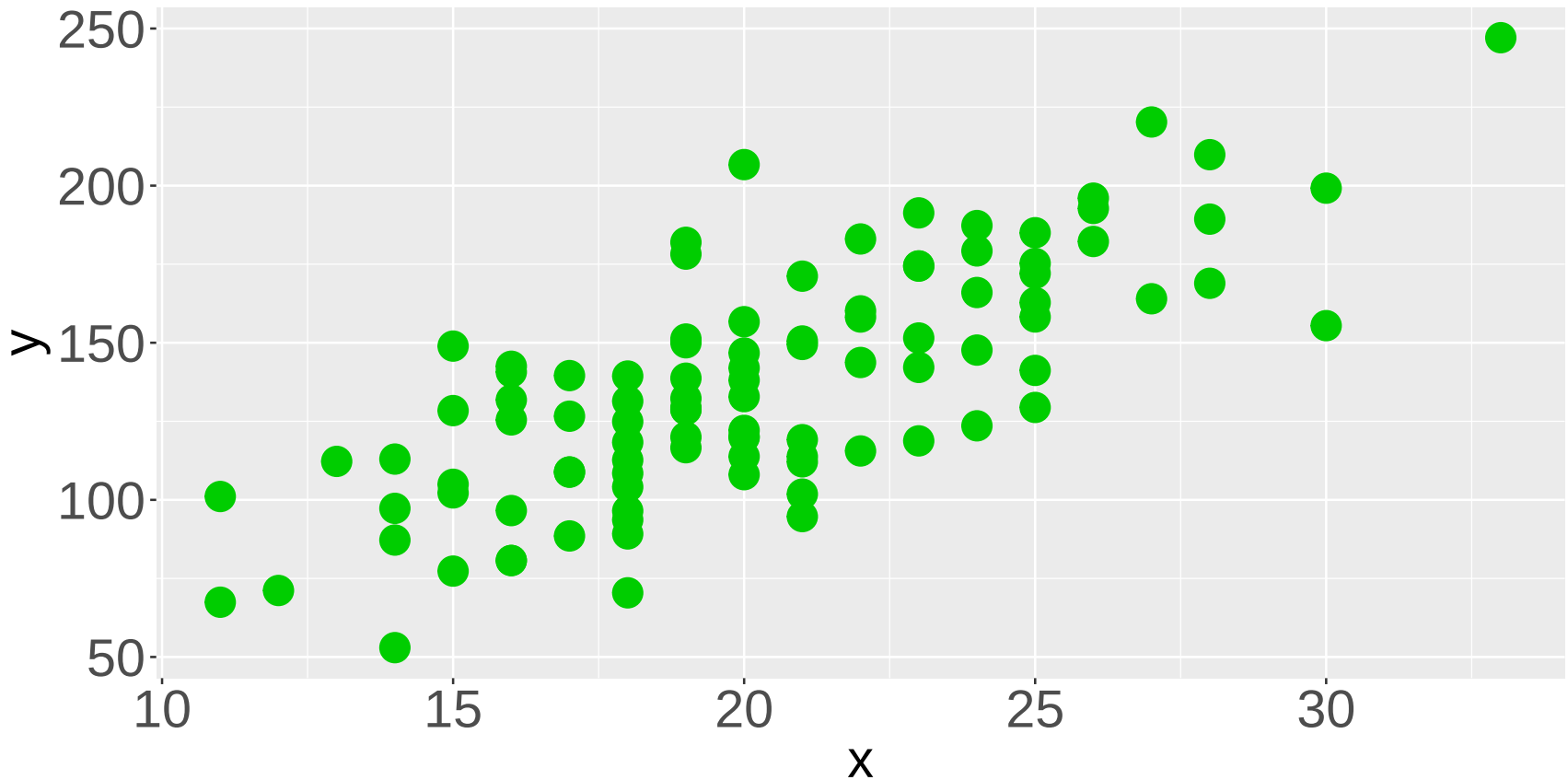
R Basics: Visualizing data with ggplot

```
ggplot(data=df, mapping=aes(x=x,fill=I("blue3"))) +
  geom_histogram(bins=15) +
  theme(legend.position = "none",
        axis.text=element_text(size=24),
        axis.title=element_text(size=24))
```



R Basics: Visualizing data with ggplot

```
ggplot(df, aes(x=x,y=y,color=I("green3"))) +
  geom_point(size=6) +
  theme(legend.position = "none",
        axis.text=element_text(size=24),
        axis.title=element_text(size=24))
```



R Basics: Practice makes perfect

Create a new script (.R file), and do the following:

1. Create a variable `N` equal to 500.
2. Create a vector `x` containing `N` random uniform variates using the function `runif`.
3. Create a vector of ones of length `N`.
4. Use `cbind()` to put the ones and the random uniform variates into the columns of a matrix `X`.
5. Create a vector `b` of length two with elements 7 and -4.
6. Create a vector `e` of length `N` of random normal variates with mean 0 and SD 3.
7. Create a vector `y` that is the linear combination of **Xb** and `e`.
8. Make a scatter plot of `x` and `y`.