Matrix Algebra in R

Paul E. Johnson[1][2]

[1] Department of Political Science

[2] Center for Research Methods and Data Analysis, University of Kansas

2018

KU

Outline

KU

Outline

Key Terms

Vector: a column of numbers

Matrix: columns of equal length side-by-side, all elements of same type (numeric, etc)

- This presentation reviews the R (R Core Team, 2017) way of working with vectors and matrices
- Along the way, we try to become tolerant of jargon like "inner product" "conform", "transpose", "symmetry", "identity matrix", "inverse", and orthogonal.

Matrix in stats: Regression!

- Regression

$$Y = X\beta + \varepsilon$$

- $X\beta$ is the "linear predictor", the inputs converted to a "regression line"

dep. var

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

Slopes

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}$$

indep var

$$X = \begin{bmatrix} 1 & x1_1 \dots & xp_1 \\ 1 & x1_2 & xp_2 \\ \vdots & \vdots & \vdots \\ 1 & x1_N \dots & xp_N \end{bmatrix}$$

error

$$\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_N \end{bmatrix}$$

predicted values

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_N \end{bmatrix} = X\hat{\beta}$$

Outline

KU

Terminology: vector

- In math, **"vector"** means **column** vector.

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

- If anybody says "vector", it is assumed they mean a column.
- This has N elements.

Creating Vectors in R: easy as falling off a log

- Many R functions create vectors.

```
x1a <- vector(mode = "double", length = 10)
x1b <- double(length = 10)
x1c <- numeric(length = 10)
## numeric is older terminology
x1a
```

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

```
identical(x1a, x1b, x1c)
```

```
[1] TRUE
```

- On screen, these things look like rows. But they are to be thought of as rows.
- Use `length()` to ask a vector how many pieces of information it holds

Creating Vectors in R: easy as falling off a log ...

```
length(x1a)
```

```
[1] 10
```

- `c()` is "concatenate", `seq()` creates "sequences"

```
x2a <- c(10, 9, 8, 7, 6, 5, 4, 3, 2, 1)
x2b <- seq(10, 1, by = -1)
identical(x2a, x2b)
```

```
[1] TRUE
```

- R refers to vectors of numbers or letters as "atomic" vectors. These hold values which are not further reducible into other structures. It does not have "attributes".

Add and Subtract Vectors

- In Math, Addition and Subtraction allowed if vectors that are EXACTLY the same size

$$\begin{bmatrix} 4 \\ 2 \\ 1 \\ 0 \\ 3 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \\ 5 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 6 \\ 0 \\ 4 \end{bmatrix} \quad or \quad \begin{bmatrix} 4 \\ 2 \\ 1 \\ 0 \\ 3 \end{bmatrix} - \begin{bmatrix} 1 \\ 2 \\ 5 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \\ -4 \\ 0 \\ 2 \end{bmatrix}$$

- Conformability of vectors: same size!
- Addition is term-by-term.

Vector addition in R

- R is similar

```
x <- c(4, 2, 1, 0, 3)
y <- c(1, 2, 5, 0, 1)
x + y
```

```
[1] 5 4 6 0 4
```

```
y - x
```

```
[1] -3  0  4  0 -2
```

- R differs: "recycling"!

```
x <- c(1,2,3,4,5,6)
y <- c(1,2)
x + y
```

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

R allows coercion of vector types

- An integer can be promoted to floating point values

```
x <- c(1L, 2L, 3L)
class(x)
```

```
[1] "integer"
```

```
## same as as.numeric()
y <- as.double(x)
y
```

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

```
class(y)
```

```
[1] "numeric"
```

R allows coercion of vector types ...

- `as.integer()` has effect of "rounding down"

```
x <- c(1.1, 2.2, 3.9)
is.integer(x)
```

```
[1] FALSE
```

```
y <- as.integer(x)
y
```

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

Transpose= turn sideways

- **The superscript T means transpose**, the column becomes a row:

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}^T = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix} \tag{1}$$

  - Often the symbol $'$ is also common, $x^T = x'$, esp. in older books.
- In R, transpose is a function `t()`.

Transpose= turn sideways ...

- On the screen, however, they both look like rows, but indexes differ slightly

```
x
```

```
[1] 1.1 2.2 3.9
```

```
t(x)
```

```
     [,1] [,2] [,3]
[1,]  1.1  2.2  3.9
```

Multiply 2 Vectors: "inner product"

- The inner product is defined as the sum of the products, $x^T \cdot y$, as follows

$$\begin{bmatrix} a & b & c & d & e \end{bmatrix} \cdot \begin{bmatrix} f \\ g \\ h \\ i \\ j \end{bmatrix} = af + bg + ch + di + ej \tag{2}$$

- The result is a single number (a "**scalar**")

Inner product of 2 vectors

$$\begin{bmatrix} a & b & c & d & e \end{bmatrix} \cdot \begin{bmatrix} f \\ g \\ h \\ i \\ j \end{bmatrix} = af + bg + ch + di + ej$$

- In math, this is defined ONLY IF the row and column vectors have EXACTLY the same number of elements.
  - Conformability

Inner product of 2 vectors ...

- Sometimes called a "dot product", but it is not necessary to write the dot
- Example

$$\begin{bmatrix} 3 & 1 & 6 & 2 \end{bmatrix} \cdot \begin{bmatrix} 1/3 \\ 1 \\ 1/6 \\ 1/2 \end{bmatrix} = 1 + 1 + 1 + 1 = 4$$

- Now you tell me. What is:

$$\begin{bmatrix} 1 & 12 & 21 \end{bmatrix} \begin{bmatrix} 0.1 \\ 0.5 \\ 1/3 \end{bmatrix} ?$$

Back to the R side

- `%*%` is the R operator calculates inner-product

```
x <- c(1, 12, 21)
y <- c(0.1, 0.5, 1/3)
t(x) %*% y
```

```
     [,1]
[1,] 13.1
```

R does check conformability for multiplication!

```
x <- c(1, 12, 21, 19, 18)
y <- c(0.1, 0.5, 1/3)
t(x) %*% y
```

```
Error in x %*% y : non-conformable arguments
```

Application: Sum of Squares

- Calculate the sum of "squares" as $x^T x$

$$\begin{bmatrix} a & b & c & d & e \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} = a^2 + b^2 + c^2 + d^2 + e^2$$

- Sum of squared residuals in regression:

$$\sum_i^N (y_i - \hat{y}_i)^2$$

$$= (y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 + (y_3 - \hat{y}_3)^2 \dots$$

$$= (y - \hat{y})^T (y - \hat{y})$$

Application: Sum of Squares ...

$$= (y_1 - \hat{y}_1, y_2 - \hat{y}_2, y_3 - \hat{y}_3 \ldots, y_N - \hat{y}_N) \begin{bmatrix} y_1 - \hat{y}_1 \\ y_2 - \hat{y}_2 \\ \vdots \\ y_N - \hat{y}_N \end{bmatrix}$$

Back on the R side of the story

- While in the math book, an inner product is not defined unless the first vector is transposed, R does not care.
- Observe. You can transpose if you want to. But it is not necessary.

```
x <- c(1,2,3)
y <- c(4,5,6)
x %*% y
```

```
     [,1]
[1,]   32
```

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

```
     [,1]
[1,]   32
```

What if you forget the percent signs in the %*% Symbol?

- In a math book, it would be very rare to see vector multiplication that is not an inner product.
- However, in stats, we do sometimes want an element-by-element multiplication

```
x <- c(1,2,3)
y <- c(4,5,6)
x * y
```

```
[1]  4 10 18
```

- R recycles x and issues a warning:

```
x <- 1:3
y <- 1:10
x*y
```

What if you forget the percent signs in the %*% Symbol?

...

```
 [1]  1  4  9  4 10 18  7 16 27 10
```

```
Warning message:
In x * y : longer object length is not a multiple
of shorter object length
```

What if you forget the percent signs in the %*% Symbol? ...

- See what it did? It manufactured a 10 element `x` for us, *as if*

```
(x <- c(1:3, 1:3, 1:3, 1))
```

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

```
(y <- 1:10)
```

```
[1]  1  2  3  4  5  6  7  8  9 10
```

```
x*y
```

```
[1]  1  4  9  4 10 18  7 16 27 10
```

Outline

KU

Many ways to create matrices

1. Use the matrix function to manufacture a 4×6 matrix

```
X <- matrix(1:24, nrow = 4, ncol = 6, byrow = FALSE,
    dimnames = list(NULL, letters[1:6]))
X
```

```
     a b  c  d  e  f
[1,] 1 5  9 13 17 21
[2,] 2 6 10 14 18 22
[3,] 3 7 11 15 19 23
[4,] 4 8 12 16 20 24
```

`byrow = FALSE` is the default, not needed to explicitly state that.

Not necessary to supply values, could have put `NA` or `0` instead.

I insert `dimnames` just to prove I can. That seems difficult for beginners. NULL row names, and a vector of column names

2. Combine columns to form a matrix (`cbind` = column bind)

Many ways to create matrices ...

```
x1 <- 1:4; x2 <- 5:8; x3 <- 9:12;
x4 <- 13:16; x5 <- 17:20; x6 <- 21:24
cbind(x1, x2, x3, x4, x5, x6)
```

```
     x1 x2 x3 x4 x5 x6
[1,]  1  5  9 13 17 21
[2,]  2  6 10 14 18 22
[3,]  3  7 11 15 19 23
[4,]  4  8 12 16 20 24
```

Difference between vectors and matrices

- In mathematics, one might say a vector is a one column matrix
- R would differentiate those ideas.
- One hint of the difference in R is that a vector does not answer to the `dim()` function (or nrow()), but a matrix does:

```
dim(x1a)
```

```
NULL
```

```
dim(X)
```

```
[1] 4 6
```

A vector answers `length()`.

- Observe that if we create a one column selection from an R matrix, R "demotes" that thing to a vector.

Difference between vectors and matrices ...

```
## Take 4th column from X
X4 <- X[ , 4]
is.matrix(X)
```

```
[1] TRUE
```

```
is.matrix(X4)
```

```
[1] FALSE
```

```
is.vector(X4)
```

```
[1] TRUE
```

- We can ask R to *not demote* X4 to become a vector by inserting a third argument

Difference between vectors and matrices ...

```
X4mat <- X[ , 4, drop = FALSE]
is.matrix(X4mat)
```

```
[1] TRUE
```

```
is.vector(X4mat)
```

```
[1] FALSE
```

- If you have a vector, however, some special functions exist that will treat it like a matrix. For example:

```
NROW(x1a)
```

```
[1] 10
```

```
NCOL(x1a)
```

Difference between vectors and matrices ...

```
[1] 1
```

These capital-letter versions of `nrow` and `ncol` can be convenient in functions where we don't know for sure if in put might be a vector or a matrix.

Multiply a matrix times a vector

- I'll create a matrix that is (2×5) (2 rows, 5 columns).
- multiply "on the right" by a vector (5×1)

$$\begin{bmatrix} a & b & c & d & e \\ r & s & t & u & v \end{bmatrix} \cdot \begin{bmatrix} f \\ g \\ h \\ i \\ j \end{bmatrix} = \begin{bmatrix} af + bg + ch + di + ej \\ rf + sg + th + ui + vj \end{bmatrix}$$

- Idea: treat matrix as two rows, calculate inner product for each one.
- $[2 \times 5] \cdot [5 \times 1]$ yields a $[2 \times 1]$ result
- Matrices must **conform**. Number of columns of first matrix must equal number of rows in 2nd one.

Multiply a matrix times a vector ...

- Example: $X\hat{\beta}$ is predicted values in regression

$$\begin{bmatrix} 1 & x1_1 & x2_1 \\ 1 & x1_2 & x2_2 \\ 1 & \dots & \dots \\ 1 & x1_N & x2_N \end{bmatrix} \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \end{bmatrix} = \begin{bmatrix} \hat{\beta}_0 + \hat{\beta}_1 x1_1 + \hat{\beta}_2 x2_1 \\ \hat{\beta}_0 + \hat{\beta}_1 x1_2 + \hat{\beta}_2 x2_2 \\ \dots \\ \hat{\beta}_0 + \hat{\beta}_1 x1_N + \hat{\beta}_2 x2_N \end{bmatrix}$$

Multiply a matrix times a matrix

$$\begin{bmatrix} a & b & c & d & e \\ r & s & t & u & v \end{bmatrix} \cdot \begin{bmatrix} f & k \\ g & l \\ h & m \\ i & n \\ j & o \end{bmatrix}$$

$$= \begin{bmatrix} af + bg + ch + di + ej & ak + bl + cm + dn + eo \\ rf + sg + th + ui + vj & rk + sl + tm + un + vo \end{bmatrix}$$

- Break into sequences of vector multiplications, row 1 · column 1, row2 · column 1, row 1 · column 2, row 2 · column 2.
- $[2 \times 5] \cdot [5 \times 2]$ yields a $[2 \times 2]$ result

R has matrix multiplication also: %*%

```
X1 <- matrix(1:12, nrow = 2)
X1
```

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

```
X2 <- matrix(13:24, ncol = 2)
X2
```

```
     [,1] [,2]
[1,]   13   19
[2,]   14   20
[3,]   15   21
[4,]   16   22
[5,]   17   23
[6,]   18   24
```

```
X1 %*% X2
```

R has matrix multiplication also: %*% ...

```
     [,1] [,2]
[1,]  593  809
[2,]  686  938
```

Transpose a Matrix

- X^T means "X turned on its side"

$$X^T = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ x1_1 & x1_2 & x1_3 & & x1_N \\ x2_1 & x2_2 & x2_3 & & x2_N \end{bmatrix}$$

- Example, predictors in a regression:

$$X = \begin{bmatrix} 1 & 3 & 33 \\ 1 & 2 & 62 \\ 1 & 5 & 65 \\ 1 & 1 & 45 \\ 1 & 5 & 66 \end{bmatrix} \quad X \text{ is } 5x3$$

$$X^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 3 & 2 & 5 & 1 & 5 \\ 33 & 62 & 65 & 45 & 66 \end{bmatrix} \quad X^T \text{ is } 3\text{x}5$$

X^TX is an important matrix in statistics

- And the product X^TX is

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 3 & 2 & 5 & 1 & 5 \\ 33 & 62 & 65 & 45 & 66 \end{bmatrix} \begin{bmatrix} 1 & 3 & 33 \\ 1 & 2 & 62 \\ 1 & 5 & 65 \\ 1 & 1 & 45 \\ 1 & 5 & 66 \end{bmatrix} = \begin{bmatrix} 5 & 16 & 271 \\ 16 & 64 & 923 \\ 271 & 923 & 15539 \end{bmatrix}$$

- Sum of squares (diagonal) and cross products (off-diagonals)
- Used to calculate correlations, regression coefficients
- X is $N \times p$, X^T is $(p \times N)$, so X^TX is $(p \times p)$, much smaller than either X or X^T
  - In the pencil days of stats, the matrix X^TX was especially heavily emphasized
  - In computer era, it has less emphasis because of "rounding error".

R has 2 ways to get this done

- In R, do not run t(X) %*% X
- Instead, use the optimized function

```
crossprod(X)
```

```
    a   b   c    d    e    f
a  30  70 110  150  190  230
b  70 174 278  382  486  590
c 110 278 446  614  782  950
d 150 382 614  846 1078 1310
e 190 486 782 1078 1374 1670
f 230 590 950 1310 1670 2030
```

R has 2 ways to get this done ...

Why is `crossprod` better? (more efficient! faster!)

1. The result is "symmetric", same above and below. Hence, computer should only need to calculate an upper triangle and copy the answer to the other triangle.
2. Creating a new transposed matrix `t(X)` unnecessarily requires a copy of the columns of `X` into the rows of `t(X)`. Computer can find values in `X` (whereas humans need to see `t(X)` explicitly).

See also `tcrossprod()` and functions in the `Matrix` package.

Accelerated matrix algebra libraries

- Many C, C++, and Fortran libraries exist, competing to be the fastest, most accurate calculation routines
- They adhere to a common, internationally accepted interface (generally referred to as BLAS)
- Over time, R has relied on LINPACK, and now LAPACK for fast calculations
- On the horizon, some are narrower stats & modeling matrix libraries, like Armadillo and Eigen, are in the spotlight through packages like RcppArmadillo and RcppEigen

KU

Outline

KU

Got Simulations?

- The MASS package for R (Venables and Ripley, 2002) introduced a simulator for Multivariate Normal draws. Allows us to generate "correlated columns"
- The theoretical model is represented as

$$MVN(\mu, \Sigma)$$

where μ is a vector of means and Σ is the covariance matrix .

$$MVN\left(\begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_p \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & \sigma_{12} & & \sigma_{1p} \\ \sigma_{12} & \sigma_2^2 & & \sigma_{2p} \\ & & \ddots & \\ \sigma_{1p} & \sigma_{2p} & & \sigma_p^2 \end{bmatrix}\right)$$

- If the variables all have 0 means and are uncorrelated, of course, MVN is the same as drawing 3 separate uncorrelated "standard normal" columns

Got Simulations? ...

```
library(MASS)
mu <- c(0, 0, 0)
Sigma <- diag(c(1, 1, 1))
Sigma
```

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

```
mvrnorm(5, mu, Sigma)
```

```
            [,1]       [,2]       [,3]
[1,] -0.1162478 -1.8179560  0.5855288
[2,]  1.8173120  0.6300986  0.7094660
[3,]  0.3706279 -0.2761841 -0.1093033
[4,]  0.5202165 -0.2841597 -0.4534972
[5,] -0.7505320 -0.9193220  0.6058875
```

Got Simulations? ...

- But we might have correlated values, for example

```
mu <- rep(0, 3)
Sigma <- matrix(c(1, .3, -.1, .3, 1, .2, -.1,
    .2, 1), nrow = 3)
Sigma
```

```
     [,1] [,2] [,3]
[1,]  1.0  0.3 -0.1
[2,]  0.3  1.0  0.2
[3,] -0.1  0.2  1.0
```

```
mvrnorm(5, mu = mu, Sigma = Sigma)
```

```
            [,1]       [,2]       [,3]
[1,] -0.1796238 -1.5100143  1.0723613
[2,] -0.3950370  1.1846379  1.3880722
[3,]  0.8681192 -0.1118966 -0.2419762
[4,]  0.3432726 -1.4629586 -1.5008948
[5,]  0.5923470 -0.3757720 -1.5577252
```

Got Simulations? ...

- The Challenge: On a theoretical level, how do we conceptualize the desired covariance matrix? What do we write in?

I understand mean and variance

- The expected value is the center point of a distribution (AKA mean).
- Variance is "dispersion" or "diversity".
- Suppose x is Normally distributed $(x \sim N(\mu, \sigma_x^2))$
  - Expected Value: $E[x] = \mu$
  - Variance: $V[x] = \sigma_x^2$, the standard deviation is σ_x.
- Next, we expand this to apply to several variables

Variance-Covariance Matrix

- Think of X as 5 predictor columns, $x1$, ..., $x5$ for N rows of observations

$$X = \begin{bmatrix} x1_1 & x2_1 & x3_1 & x4_1 & x5_1 \\ x1_2 & x2_2 & x3_2 & x4_2 & x5_2 \\ x1_3 & x2_3 & x3_3 & x4_3 & x5_3 \\ & & \vdots & & \\ x1_N & x2_N & x3_N & x4_N & x5_N \end{bmatrix} \tag{3}$$

$$Var(X) = \Sigma = \begin{bmatrix} \sigma^2_{x1} & \sigma_{x2,x1} & \sigma_{x3,x1} & \sigma_{x4,x1} & \sigma_{x5,x1} \\ \sigma_{x2,x1} & \sigma^2_{x1} & \sigma^2_{x3,x2} & \sigma_{x4,x2} & \sigma_{x5,x2} \\ \sigma_{x3,x1} & \sigma_{x3,x2} & \sigma^2_{x3} & \sigma_{x4,x3} & \sigma_{x5,x3} \\ \sigma_{x4,x1} & \sigma_{x4,x2} & \sigma_{x4,x3} & \sigma^2_{x4} & \sigma_{x5,x4} \\ \sigma_{x5,x1} & \sigma_{x5,x2} & \sigma_{x5,x3} & & \sigma^2_{x5} \end{bmatrix} \tag{4}$$

Variance-Covariance Matrix ...

- The diagonals are variances, which range from $[0, \infty)$, but the off-diagonals are scale-free numbers called "*covariances*", that range from $(-\infty, \infty)$.
- If covariance is positive, two variables "go together". But how big should it be? I can't conceptualize that.
- It is easier for me to conceptualize
  1. The standard deviations of the columns: $[\sigma_{x1}, \sigma_{x2}, \ldots, \sigma_{x5}]$
  2. The Pearson correlation matrix among the columns

You know correlations, right?

- Pearson product moment correlation is the ratio of covariance to the product of the standard deviations (example with variables $x1$ and $x3$):

$$\rho_{x1,x3} = \frac{\sigma_{x1,x3}}{\sigma_{x1} \cdot \sigma_{x3}}$$

- Correlation ranges from $(-1, 1)$
  - 0 indicates 2 variables are not related.
- Rearrange to create another way to calculate Covariance

$$\sigma_{x1,x3} = \sigma_{x1} \cdot \sigma_{x3} \cdot \rho_{x1,x3} \tag{5}$$

Cov and Corr matrices

- A Correlation matrix

$$\rho = \begin{bmatrix} 1 & \rho_{12} & \rho_{13} & \cdots & \rho_{1p} \\ \rho_{21} & 1 & \rho_{23} & & \rho_{2p} \\ \rho_{31} & \ddots & 1 & & \rho_{3p} \\ \vdots & & \rho_{11} & \ddots & \\ \rho_{p1} & \rho_{11} & \rho_{11} & & 1 \end{bmatrix}$$

- Its Symmetric! Elements bounded between -1 and $+1$
- Example

$$\rho = \begin{bmatrix} 1 & .8 & 0 & \ldots & 0 \\ .8 & 1 & 0 & & 0 \\ 0 & \ddots & 1 & & 0 \\ \vdots & & 0 & \ddots & \\ 0 & 0 & 0 & & 1 \end{bmatrix}$$

Cov and Corr matrices ...

- Restriction: the intercorrelations among several variables must make sense. Suppose
  - $x1$ is very tightly correlated with $x2$, and
  - $x2$ is tightly correlated with $x3$, then
  - its not conceptually meaningful to suppose $x1$ is negatively related to $x3$
- The restriction is that ρ is "positive definite", meaning $y^T \rho\, y > 0$ for any vector y. Roughly speaking, a vector cannot be negatively correlated with itself.

Variance-Covariance Matrix as Re-scaled Correlation

$$Variance = Std.Deviation \times Correlation \times Std.Deviation$$

$$\Sigma = \begin{bmatrix} \sigma_{x1} & 0 & 0 & 0 & 0 \\ 0 & \sigma_{x2} & 0 & 0 & 0 \\ 0 & 0 & \sigma_{x3} & 0 & 0 \\ 0 & 0 & 0 & \sigma_{x4} & 0 \\ 0 & 0 & 0 & 0 & \sigma_{x5} \end{bmatrix} \times \begin{bmatrix} 1 & \rho_{12} & \rho_{13} & \cdots & \rho_{1p} \\ \rho_{21} & 1 & \rho_{23} & & \rho_{2p} \\ \rho_{31} & \ddots & 1 & & \rho_{3p} \\ \vdots & 11 & \rho_{11} & \ddots & \\ \rho_{p1} & \rho_{11} & \rho_{11} & & 1 \end{bmatrix} \times \begin{bmatrix} \sigma_{x1} & 0 & 0 & 0 & 0 \\ 0 & \sigma_{x2} & 0 & 0 & 0 \\ 0 & 0 & \sigma_{x3} & 0 & 0 \\ 0 & 0 & 0 & \sigma_{x4} & 0 \\ 0 & 0 & 0 & 0 & \sigma_{x5} \end{bmatrix} \quad (6)$$

- Inspect an individual piece

Variance-Covariance Matrix as Re-scaled Correlation ...

- Σ_{11} should be the variance of $x1$

$$\sigma_{x1,x1} = \sigma_{x1} \cdot \sigma_{x1} = \sigma_{x1}^2$$

- Σ_{13} is a "cross" term, that weights the two standard deviations by their correlations

$$\sigma_{x1,x3} = \rho_{13}\sigma_{x1}\sigma_{x2} \tag{7}$$

R has tools to get that done

- An example correlation matrix: everything is equally strongly correlated with everything else:

```
Rho
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]  1.0  0.5  0.5  0.5  0.5
[2,]  0.5  1.0  0.5  0.5  0.5
[3,]  0.5  0.5  1.0  0.5  0.5
[4,]  0.5  0.5  0.5  1.0  0.5
[5,]  0.5  0.5  0.5  0.5  1.0
```

- Is the diagonal full of 1's?

```
Rho.d <- diag(Rho)
Rho.d
```

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

R has tools to get that done ...

```
all.equal(Rho.d, rep(1, times = 5))
```

```
[1] TRUE
```

- Is it symmetric?

```
isSymmetric(Rho)
```

```
[1] TRUE
```

- Are all values in $[-1, 1]$?

```
## Seems like should be more direct way, but...
z <- as.vector(Rho)
z
```

R has tools to get that done ...

```
 [1] 1.0 0.5 0.5 0.5 0.5 0.5 1.0 0.5 0.5 0.5 0.5 0.5 1.0 0.5 0.5 0.5
     0.5 0.5 1.0 0.5 0.5 0.5 0.5 0.5
[25] 1.0
```

```
## single | for vector compare
any(z > 1 | z < -1)
```

```
[1] FALSE
```

- How to check if it is positive definite? In the `MASS::mvrnorm` function, Venables and Ripley show one way.
  - In "exact math" a matrix is positive definite if all of its eigenvalues are positive
  - Computers don't do exact math, however
  - V&R's solution is to require that the estimated eigenvalues must be positive, or nearly so. The variable "tol" is tolerance, 10^{-6}, a practical standard

R has tools to get that done ...

```
eS <- eigen(Sigma, symmetric = TRUE)
ev <- eS$values
if (!all(ev >= -tol * abs(ev[1L])))
          stop("'Sigma' is not positive definite")
```

- This allows the possibility that the smallest eigenvalue, `ev[1L]`, might be negative, but it must not be too far below 0.
- I found that so useful I put same calculation into a function in `rockchalk` called "`checkPosDef`".

What's all that good for?

- In 30 years of teaching, I wrote 2 good lectures, one of which is: http://pj.freefaculty.org/guides/stat/Regression/Multicollinearity/Multicollinearity-1-lecture.pdf
- Get the highlights:

```
library(rockchalk)
example(mcGraph3)
```

The Regression Book says . . .

- Regression book says

$$y = X\beta + \varepsilon$$

- The "first order condition" for optimizing values of β is the "normal equation":

$$(X^T X)\beta = X^T y \tag{8}$$

- Which the book will say is solved by finding an inverse matrix $(X^T X)^{-1}$ that we multiply on the left to get $\hat{\beta}$ by itself

$$\cancel{(X^T X)^{-1}}\cancel{(X^T X)}\hat{\beta} = (X^T X)^{-1} X^T y$$

$$\hat{\beta} = (X^T X)^{-1} X^T y \tag{9}$$

The Regression Book says . . . ...

- While correct on a theoretical level, this amounts to poor computational numeric linear algebra. Regression estimates are not calculated in that way.
- Now I'll explain all of the inter-related terms.

Identity Matrix

The matrix equivalent of the number 1 is I, the Identity Matrix

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- I times $anything$ gives back $anything$
- $anything$ times I gives back $anything$

$$I \times y = y$$

$$X \times I = X$$

Inverse Matrix

- The sum of squares and cross products is a square matrix $(X^T X)$.
- If we could find an inverse matrix $(X^T X)^{-1}$, then

$$(X^T X)^{-1}(X^T X) = I$$

- The matrix $(X^T X)$ is "invertible" under some "regularity" conditions (lets worry about that another time).
- Hence, in exact math, the normal equation $(X^T X)\beta = X^T y$ can be converted to the solution

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

R can calculate the inverse of a matrix

- Virtually every stats teacher I know has used R matrix calculations to show we can reproduce the estimates from a regression function. Here's a sketch

```
mod1 <- lm(y ~ x1 + x2 + x3, data =
    any_data_you_have)
summary(mod1)
X <- model.matrix(mod1)
XTX <- t(X) %*% X
XTXinv <- solve(XTX)
BetaHat <- XTXinv %*% t(X) %*% any_data_you_have$y
```

- I ran example(lm), which created an outcome variable weight and a regression object lm.D9.
  - Then:

R can calculate the inverse of a matrix ...

```
X <- model.matrix(lm.D9)
XTX <- t(X) %*% X
XTXinv <- solve(XTX)
Beta <- XTXinv %*% t(X) %*% weight
```

- Which appears to be the same as the fitted model:

```
> Beta
              [,1]
(Intercept)  5.032
groupTrt    -0.371
> coef(lm.D9)
(Intercept)    groupTrt
      5.032      -0.371
```

- But, if we dial up the number of displayed digits, the numbers are not the same:

R can calculate the inverse of a matrix ...

```
options.orig <- options()
options(digits = 22)
> coef(lm.D9)
              (Intercept)                    groupTrt
 5.0320000000000000284217 -0.3709999999999997188915
> Beta
                                  [,1]
(Intercept)  5.0320000000000000916600
groupTrt    -0.3710000000000000995648
options(options.orig)
```

- Why these differ in the decimal places, or how they come to differ, is the big news in the next few slides.

Now the tragic news

- No respectable software today would explicitly form X^TX for the purpose of calculating regression estimates. Digital rounding error, inherent in floating point calculations, is damaging and unnecessary
- No respectable software calculates $(X^TX)^{-1}$. Doing so compounds on the rounding error inherent in X^TX
- There are many ways to calculate inverses, some are more numerically stable, some faster. But all are better than $(X^TX)^{-1}$

How do they actually do it?

- "Use the Source, Luke" (Kenobe, 1977)
- First, type "lm" with no parentheses

```
lm
```

- Scan through there, look for "`lm.fit(x, y, ...)`".
- Check the code for "lm.fit". There's no $(X^T X)^{-1}$, no hint of t(x) %*% x
- Instead, the magic bullet is

```
z <- .Call(C_Cdqrls, x, y, tol, FALSE)
```

- That's a call to a C function which calculates the "QR" decomposition of x

The QR decomposition of the predictor matrix

- The QR decomposition: A matrix X can be reproduced as the product of 2 parts,
  1. An orthogonal matrix Q
  2. An upper right triangular R (with rows of 0's padding the bottom so that it is length N).
- Suppose X is $N \times p$ (regression predictors). Reproduce X as

$$X = Q \begin{bmatrix} R \\ 0 \end{bmatrix}$$

- The matrix Q is $N \times N$, which means it is huge, but it has a very interesting property: the correlation between each column and any of the other columns is 0. I mean to say, the columns are orthogonal to each other. For two columns $Q_{.j}$ and $Q_{.k}$,

$$Q_{.j}^T Q_{.k} = 0$$

The QR decomposition of the predictor matrix ...

- The Q matrix is also scaled so the inner product of any column with itself is 1.

$$Q_{.j}^T Q_{.j} = 1$$

- This implies: $Q^T Q = I$, $QQ^T = I$.
  - Literally, $Q^{-1} = Q^T$.
  - The inverse of an orthogonal matrix is very easy to calculate, in other words.
- The requirement that Q is huge, $N \times N$, would ordinarily discourage us because memory storage would be very expensive. However, it turns out we only need the first p columns from Q.

KU

The QR decomposition of the predictor matrix ...

- The bottom part of the stack, $\begin{bmatrix} R \\ 0 \end{bmatrix}$, is $N - p$ rows of 0's:

$$\begin{bmatrix} R \\ 0 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} & r_{1p} \\ 0 & r_{22} & r_{23} & r_{24} & r_{2p} \\ 0 & 0 & r_{33} & r_{34} & r_{3p} \\ 0 & 0 & 0 & \ddots & r_{(N-1)p} \\ 0 & 0 & 0 & 0 & r_{NN} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{10}$$

The QR decomposition of the predictor matrix ...

- The bottom rows of R are all zeros, meaning that the columns on the right side of Q don't matter.

$$X = \begin{bmatrix} q_{11} & & q_{1N} \\ & \ddots & \\ q_{21} & [N \times N] & \\ & \ddots & \\ q_{N1} & & q_{NN} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{1p} \\ 0 & r_{22} & r_{23} & r_{2p} \\ 0 & 0 & r_{33} & r_{3p} \\ & & \ddots & \\ 0 & 0 & & r_{(p-1)p} \\ 0 & 0 & 0 & r_{pp} \\ 0 & 0 & [N-p] & 0 \\ 0 & 0 & [rows] & 0 \\ 0 & 0 & [of\ 0's] & 0 \end{bmatrix} \tag{11}$$

  The last $m - n$ columns of Q get zeroed out by the 0's on the bottom of R.

- The original matrix X can be reproduced if we just use the p columns on the left side of Q and the triangular matrix R

The QR decomposition of the predictor matrix ...

$$X = \begin{bmatrix} q_{11} & q_{12} & & q_{1p} \\ q_{21} & \ddots & & \\ & [N \times p] & & \\ & & & \\ q_{N1} & & & q_{Np} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} & r_{1p} \\ 0 & r_{22} & r_{23} & r_{24} & r_{2p} \\ 0 & 0 & r_{33} & r_{34} & r_{3p} \\ 0 & 0 & 0 & \ddots & r_{(p-1)p} \\ 0 & 0 & 0 & 0 & r_{pp} \end{bmatrix} \tag{12}$$

- This more "petite" version (Q_f) is the one that R saves in memory

$$X = Q_f R \tag{13}$$

The QR decomposition of the predictor matrix ...

- In regression analysis, we symbolically derive

$$\hat{\beta} = (X^T X)^{-1} X^T y \tag{14}$$

A very accurate, reasonably fast way to calculate that is with QR. Replace X by the petite $Q_f R$.

$$\hat{\beta} = ((Q_f R)^T (Q_f R))^{-1} (Q_f R)^T y \tag{15}$$

If we use the rules for inverses and transposes mentioned above, we can algebraically reduce that:z $>$ 1 || z $<$ -1

$$\begin{aligned} \hat{\beta} &= (R^T Q_f^T Q_f R))^{-1} (Q_f R)^T y && (16) \\ & (R^T R)^{-1} R^T Q_f^T y && (17) \\ & R^{-1} R^{T^{-1}} R^T Q_f^T y && (18) \\ & R^{-1} Q_f^T y && (19) \end{aligned}$$

The QR decomposition of the predictor matrix ...

- What's the big idea there?
  - We need the QR decomposition of X to calculate regression estimates
  - We do not need $(X^T X)^{-1}$.z > 1 || z $<$ -1
- The only regression book in which I have found this written out clearly is Simon Wood's *Generalized Additive Models* (2006).
- I started more notes on this http://pj.freefaculty.org/guides/stat/Math/Matrix-Decompositions

KU

Outline

KU

The High Points

- If we are doing statistics, we are doing math
  - with vectors and matrices
- There are some basic chores that all methodologists should be able to handle which will require some comfort with matrices
  - Creating covariance and correlation matrices
  - Drawing random samples
- This lecture introduced only a small slice of matrix algebra in order to illustrate 2 main points
  - R has code to do calculations that math books describe, but
  - in a digital computer, matrix algebra does not work exactly as planned in a math book that presumes exact calculations of floating point numbers
- If we study the way the R team has implemented numerical calculations, we can push forward our study of matrix algebra by focusing on the tools that are immediately relevant (the QR decomposition, for example)

KU

That intriguing comment in prcomp

- In the base R distribution, there are 2 functions for principal components analysis, princomp and prcomp.
  - princomp is the older one
  - prcomp is the newer one
- Care to guess why there are two?
  - In princomp, "Details" explains

```
The calculation is done using 'eigen' on the
   correlation or covariance matrix, as
   determined by 'cor'.  This is done for
   compatibility with the S-PLUS result. A
   preferred method of      calculation is to
   use 'svd' on 'x', as is done in 'prcomp'
```

- The SVD (Singular Value Decomposition) of a matrix is
  - more accurate, but also more expensive to calculate

That intriguing comment in prcomp ...

- The traditional approach is to calculate the eigenvalue-decomposition on a square crossproducts matrix, $X^T X$, rather than X itself.
- Because SVD can apply to X, without forming $X^T X$, it is more accurate.

Online Free Resources

- Højsgaard, Soren, "Linear algebra in R". This is my favorite. A beautifully done essay that covers many details. I can't find this in Hojsgaard's page today, but I find plenty of other people have it available if you search in Google.
- Farnsworth, Grant V, "Econometrics in R".
- Bates, Douglas, (June 2004) "Least Squares Calculations in R: Timing Different Approaches", Rnews, 4(1): 17
- Quick R, "Matrix Algebra"

References

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Session

```
sessionInfo()
```

```
R version 3.4.4 (2018-03-15)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 18.04 LTS

Matrix products: default
BLAS: /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
     LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=en_US.UTF-8     LC_MONETARY=en_US.UTF-8
     LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8
    LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  base

other attached packages:
[1] rockchalk_1.8.111 MASS_7.3-49
```

Session ...

```
loaded via a namespace (and not attached):
 [1] Rcpp_0.12.15         lattice_0.20-35     grid_3.4.4
     MatrixModels_0.4-1   nlme_3.1-137
 [6] SparseM_1.77         minqa_1.2.4         nloptr_1.0.4        car_2.1-6
                Matrix_1.2-14
[11] splines_3.4.4        lme4_1.1-17         tools_3.4.4
     pbkrtest_0.4-7       parallel_3.4.4
[16] compiler_3.4.4       mgcv_1.8-23         nnet_7.3-12
     quantreg_5.35        methods_3.4.4
```