This section continues where we left off last week, introducing you to increasingly complex matrix manipulation in R and finishing with some puzzles in R. Soon we'll move on to real life data, I promise. But for now... matrices!

Matrix operations

We once again define A and B as before:

```
A <- matrix(1:6, ncol=2)
B <- matrix(1:6, ncol=3, byrow=TRUE)
```

As always, keeping track of your matrix dimensions is a Good Idea™. That's where the `dim()` command comes in handy:

```
dim(A)
dim(B)
```

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

Matrix muliplication in R is bound to `%*%`, whereas scalar multiplication is bound to `*`. Consider the product $\mathbf{BA}$:

```
B %*% A
```

```
     [,1] [,2]
[1,]   14   32
[2,]   32   77
```

The dimensions have to line up properly for matrix multiplication to be appropriately applied, otherwise R returns an error, as is the case with the product $\mathbf{BA}'$:

```
B %*% t(A)
```

```
Error in B %*% t(A) : non-conformable arguments
```

If scalar multiplication is applied to matrices of exactly the same dimensions, then the result is element-wise multiplication. This type of operation is sometimes called the Hadamard product, denoted $\mathbf{B} \circ \mathbf{A}'$:

```
B * t(A)
```

```
     [,1] [,2] [,3]
[1,]    1    4    9
[2,]   16   25   36
```

More common, if we want to scale all elements by a factor of two, say, we just multiply a matrix by a scalar; but note that `class(2)` must be not be `matrix` but rather `numeric` so as to avoid a non-conformable error:

```
A * 2
```

```
     [,1] [,2]
[1,]    2    8
[2,]    4   10
[3,]    6   12
```

```
A * matrix(2)
```

```
Error in A * matrix(2) : non-conformable arrays
```

Consider a more complicated operation, whereby each column of a matrix is multiplied element-wise by another, fixed column. Here, each column of a particular matrix is multiplied in-place by a fixed column of residuals. Let $\mathbf{e}$ be a vector defined as an increasing sequence of length three:

```
e <- matrix(1:3)
```

Note first that the default sequence in `R` is a column vector, and not a row vector. We would like to `apply` a function to each column of $\mathbf{A}$, specifically a function that multiplies each column in-place by $\mathbf{e}$. We must supply a 2 to ensure that the function is applied to the second dimension (columns) of $\mathbf{A}$:

```
apply(A, 2, function(x) {x * e})
```

```
     [,1] [,2]
[1,]    1    4
[2,]    4   10
[3,]    9   18
```

The function that is applied is anonymous, but it could also be bound to a variable – just as a matrix is bound to a variable:

```
whoop <- function(x) {x * e}
apply(A, 2, whoop)
```

```
     [,1] [,2]
[1,]    1    4
[2,]    4   10
[3,]    9   18
```

We will often need to define an identity matrix of dimension n, or $\mathbf{I}_n$. This is quick using `diag`:

```
I <- diag(5)
```

There are many ways to calculate the trace of $\mathbf{I}_5$. One method has been bundled into a function, called `tr()`, that is included in a package called `psych` which is not included in the base distribution of `R`. We will need to grab and call the library to have access to the function, installing it with the command `install.packages("psych")`. For this, you'll need an internet connection.

```
library(psych)
tr(I)
```

```
[1] 5
```

Linear algebra puzzles

1. Define vectors $\mathbf{x} = [1\ \ 2\ \ 3]'$, $\mathbf{y} = [2\ \ 3\ \ 4]'$, and $\mathbf{z} = [3\ \ 5\ \ 7]$. Define $\mathbf{W} = [\mathbf{x}\ \ \mathbf{y}\ \ \mathbf{z}]$. Calculate $\mathbf{W}^{-1}$. If you cannot take the inverse, explain why not and adjust $\mathbf{W}$ so that you *can* take the inverse. *Hint*: the `solve()` function will return the inverse of the supplied matrices.

2. Show, somehow, that $(\mathbf{X}')^{-1} = (\mathbf{X}^{-1})'$.

3. Generate a 3×3 matrix $\mathbf{X}$, where each element is drawn from a standard normal distribution. Let $\mathbf{A} = \mathbf{I}_3 - \frac{1}{3}\mathbf{B}$ be a demeaning matrix, with ß a 3×3 matrix of ones. First show that $\mathbf{A}$ is idempotent and symmetric. Next show that each row of the matrix $\mathbf{XA}$ is the deviation of each row in $\mathbf{X}$ from its mean. Finally, show that $(\mathbf{XA})(\mathbf{XA})' = \mathbf{XAX}'$, first through algebra and then `R` code.

4. Demonstrate from random matrices that $(\mathbf{XYZ})^{-1} = \mathbf{Z}^{-1}\mathbf{Y}^{-1}\mathbf{X}^{-1}$.

5. Let $\mathbf{X}$ and $\mathbf{Y}$ be square 20×20 matrices. Show that $tr(\mathbf{X} + \mathbf{Y}) = tr(\mathbf{X}) + tr(\mathbf{Y})$.

6. Generate a diagonal matrix $\mathbf{X}$, where each element on the diagnonal is drawn from $U[10, 20]$. Now generate a matrix $\mathbf{B}$ s.t. $\mathbf{X} = \mathbf{BB}'$. *Hint*: There is a method in `R` that makes this easy. Does the fact that you can generate $\mathbf{B}$ tell you anything about $\mathbf{X}$?

7. Demonstrate that for any scalar c and any square matrix $\mathbf{X}$ of dimension n that $\det(c\mathbf{X}) = c^n \det(\mathbf{X})$.

8. Demonstrate that for an $m \times m$ matrix $\mathbf{A}$ and a $p \times p$ matrix $\mathbf{B}$ that $\det(\mathbf{A} \otimes \mathbf{B}) = \det(\mathbf{A})^p \det(\mathbf{B})^m$. *Hint*: Note that $\otimes$ indicates the Kronecker product[1]. Google the appropriate `R` function.

[1] The Kronecker product is a useful mathemagical tool for econometricians, allowing us to more easily describe block-diagonal matricees for use in panel data settings. I wouldn't lose sleep over it, though.