R tip o' the Day

-Introduction to Data Science-

Lecturer: Darren Homrighausen, PhD

R Tip of the Day: Vectorization

R is a vectorized, object oriented language.

Vectorization: Generalize operations on scalars (real numbers) to apply transparently to vectors, matrices, and higher dimensional arrays

This means, that usually doing something to a vector mimics the analogous operation on a single number. Example:

```
> a = b = 1
> a+b
[1] 2
> a = 1:9
> b = 2:10
> a + b
[1]  3  5  7  9 11 13 15 17 19
> 10*a
[1] 10 20 30 40 50 60 70 80 90
```

R Tip of the Day: Vectorization

R will also cycle shorter vectors when multiplied by longer ones

R will guess at what you mean, but this may be wrong. watch out!

```
> c = c(0,1)
> a*c
[1] 0 2 0 4 0 6 0 8 0
Warning message:
In a*c: longer object length is not a multiple of shorter object
> a = 1:10
> a*c
 [1]  0  2  0  4  0  6  0  8  0 10
```

R Tip of the Day: Numerical maximization

Suppose we have a function f which is a function of a variable x and we want to find

$$x_* = \arg\max_x f(x).$$

How can we do this in R? Suppose x_* is known to be between 0 and 1.

```
f = function(x){
  #some stuff#
  return(f.x)
}
xGrid  = (0:100)/100 #Gives us a grid for the function
fOut   = f(xGrid)
x_star = xGrid[which.max(fOut)]
```

(it's worth pointing out that there are many, more advanced ways of numerical optimization: L-BFGS, gradient descent, genetic algorithms, ...)

R Tip of the Day

Remember: R is an object oriented language.

This means that it stores everything as an object, inside an object, inside an object, ... Practically, this means that when you write

```
a = 10
```

there is now an object in R that has name 'a' and value 10.

But, it can go much deeper. Objects can have objects as well.

When this happens, we can access object2 inside the object1 by typing object1$object2

```
a = data.frame('cows' = 10, 'chickens' = 34)
> a$cow
[1] 10
> a$chi
[1] 34
```

R Tip of the Day

Even deeper, functions themselves are objects

```
> a = 10
> b = a
> b
[1] 10
> sum
function (..., na.rm = FALSE)  .Primitive("sum")
> sum = a + b
> sum
[1] 20
> sum(c(a,b))
[1] 20
```

Good coding practice

R Tip of the Day: Good Coding Practice

It helps other people (and yourself) know what you are accomplishing with a certain program.

(our future self is more of a stranger than we'd like to believe)

This isn't meant to be exhaustive, just a couple of tips:

- Use descriptive object names (even if they are long).

R Tip of the Day: Good Coding Practice

It helps other people (and yourself) know what you are accomplishing with a certain program.

(our future self is more of a stranger than we'd like to believe)

This isn't meant to be exhaustive, just a couple of tips:

- Use descriptive object names (even if they are long).
- Give a good introductory summary (metadata).

R Tip of the Day: Good Coding Practice

It helps other people (and yourself) know what you are accomplishing with a certain program.

(our future self is more of a stranger than we'd like to believe)

This isn't meant to be exhaustive, just a couple of tips:

- Use descriptive object names (even if they are long).
- Give a good introductory summary (metadata).
- Limit comments in the body of your code.

R Tip of the Day: Good Coding Practice

It helps other people (and yourself) know what you are accomplishing with a certain program.

(our future self is more of a stranger than we'd like to believe)

This isn't meant to be exhaustive, just a couple of tips:

- Use descriptive object names (even if they are long).
- Give a good introductory summary (metadata).
- Limit comments in the body of your code.
- Keep each file/function doing only one specific task.

R Tip of the Day: Good Coding Practice

It helps other people (and yourself) know what you are accomplishing with a certain program.

(our future self is more of a stranger than we'd like to believe)

This isn't meant to be exhaustive, just a couple of tips:

- Use descriptive object names (even if they are long).
- Give a good introductory summary (metadata).
- Limit comments in the body of your code.
- Keep each file/function doing only one specific task.
- Use consistent spacing both before objects and on each line.

R Tip of the Day: Good Coding Practice

It helps other people (and yourself) know what you are accomplishing with a certain program.

(our future self is more of a stranger than we'd like to believe)

This isn't meant to be exhaustive, just a couple of tips:

- Use descriptive object names (even if they are long).
- Give a good introductory summary (metadata).
- Limit comments in the body of your code.
- Keep each file/function doing only one specific task.
- Use consistent spacing both before objects and on each line.
- Keep each line doing only one major thing.

R Tip of the Day: Good Coding Practice

It helps other people (and yourself) know what you are accomplishing with a certain program.

(our future self is more of a stranger than we'd like to believe)

This isn't meant to be exhaustive, just a couple of tips:

- Use descriptive object names (even if they are long).
- Give a good introductory summary (metadata).
- Limit comments in the body of your code.
- Keep each file/function doing only one specific task.
- Use consistent spacing both before objects and on each line.
- Keep each line doing only one major thing.
- **Do not optimize for speed or storage before necessary**.

R Tip of the Day: Good Coding Practice

Here is an example. Suppose we want to write our own 'scale' function. Then we might do the following: Create a file called columnScale.r with the following

```
#### columnScale(X)
####      Arguments:
####                * X is an n by p matrix which we are
####                  normalizing to have column sample
####                  standard deviation equal to 1.
columnScale = function(X){
  Xstd    = apply(X,2,sd)
  Xscaled = t(t(X)/Xstd)
  return( Xscaled )
}
```

R Tip of the Day: Good Coding Practice

Then in another file, called columnCenter.r

```
#### columnCenter(X)
####      Arguments:
####                * X is an n by p matrix which we are
####                   normalizing to have column sample
####                   mean equal to zero.
columnCenter = function(X){
  XcenteringMatrix = apply(X,2,mean)
  Xcentered        = t( t(X) - XcenteringMatrix)
  return( Xcentered )
}
```

R Tip of the Day: Good Coding Practice

Putting it all together, in another file we would have:

```
#### scaleNew(X,center=TRUE,scale=TRUE)
####      Arguments:
####                * X is an n by p matrix which we are
####                   normalizing to have either (or both)
####                   column sample mean equal to zero
####                   or column sample sd equal to one.
scaleNew = function(X,center=TRUE,scale=TRUE){
  source(columnCenter.r)
  source(columnScale.r)
  if(center) X = columnCenter(X)
  if(scale)  X = columnScale(X)
  return( X )
}
```

(As an aside, this code wouldn't allow someone to scale before centering (I'm not sure why you'd want to))