Today

- Organizing your repository (naming files)
- Working some homework problems
  - coding demonstration
  - pair programming, questions?
- Indexing vectors
- Naming variables
- Repetition structures:
  - counter control using for

Naming files

- Good advice here:
  - https://datacarpentry.org/rr-organization1/01-file-naming/index.html
- Three principles
  - Machine readable
  - Human readable
  - Plays well with default ordering

Worked homework

- Tue 3.Q2 (class grades)
- Tue 3.Q2 fixed (pair programming)
- Thu Q1 (complete the algorithm)
- Thu Q2 (pair programming)

Indexing vectors

MyVec

Element 1	6.115
Element 2	7.726
Element 3	8.352
etc	6.289
	1.087
	7.344
	2.911
	3.209
	5.290
	4.445
	2.505
	4.541
	5.568
	6.873
	5.208
	3.631

MyVec[3]

Each element is a slot in the computer's memory (RAM).

See also week 1 Thu lecture

PS many languages use offset indexing (i.e. 0 is first element)

Naming variables

- See class style guide
- short
- descriptive
- underline separator

R: `for` repetition structure

Most programming languages have a specialized structure for counter-controlled repetition (usually called "for")

```
for ( i in starti:endi ) {
    expression
}
```

R: `for` repetition structure

Example

```
for ( i in 1:10 ) {
    j <- i * 2
    print(j)
}
```

What does this do?

The 4 components of counter control using `while` or `for`

Counter

Counter initialized to 1

```
i <- 1
while ( i <= n ) {
    expression
    i <- i + 1
}
```

Number of loops

Counter incremented by 1

Counter

Counter initialized to 1

```
for ( i in 1:n ) {
    expression
}
```

Number of loops

Counter increments by 1

R: `for` repetition structure

Correct

```
for ( i in 1:n ) {
    expression
}
```

Incorrect

```
i <- 1
for ( i in 1:n ) {
    expression
    i <- i + 1
}
```

```
# Finds the number (y) that is the bth power of x

# Initialize parameters
x <- 3.2     #Any real number
b <- 2       #Any integer > 0

# Initialize working variables
y <- 1
counter <- 1

# Processing phase
while ( counter <= b ) {
    y <- y * x
    counter <- counter + 1
}

# Termination phase
y
```

This code uses a while structure to do counter controlled repetition. Modify it to use a for counter-control structure instead.

Increment variation

```
for ( i in seq(0, n, 2) ) {
    expression
}
```

Counter initialized to 0

Counter incremented by 2

```
i <- 0
while ( i <= n ) {
    expression
    i <- i + 2
}
```

R: `for` is vector controlled

R's for structure is actually vector controlled repetition, a special case of counter controlled repetition

seq is an expression that evaluates to a vector

```
for ( var in seq ) {
    expression
}
```

var will in turn be assigned the value of each element in the vector

Any vector will do!

R: `for` is vector controlled

Example

```
a <- c(0.51,0.57,0.09,1.02,1.10)
for ( number in a ) {
    print(number * 2)
}
```

What does this do?