Programming: Intro to For Loop

Stat 133 with Gaston Sanchez

Introduction to `for` Loops

Big favor

In order to learn about loops, I'm going to ask you to **forget about vectorization**.

Instead, let's describe how to perform those type of operations "manually", step by step.

Vectorization Reminder

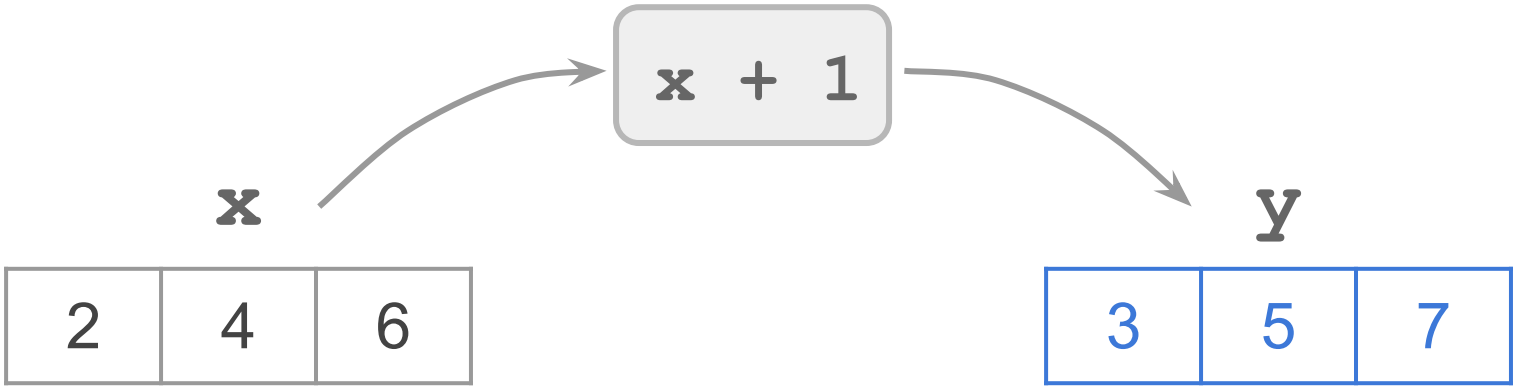
A **vectorized** computation is any computation that when applied to a vector operates on all of its elements

```
c(1, 2, 3) + c(3, 2, 1)
```

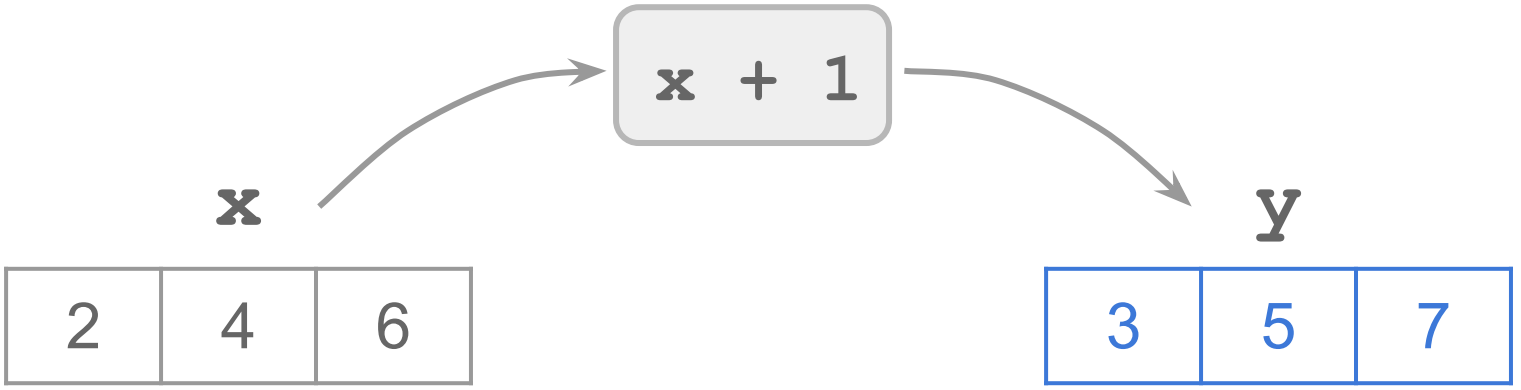
```
c(1, 2, 3) * 3
```

```
abs(c(-1, 2, 0))
```

Motivation example



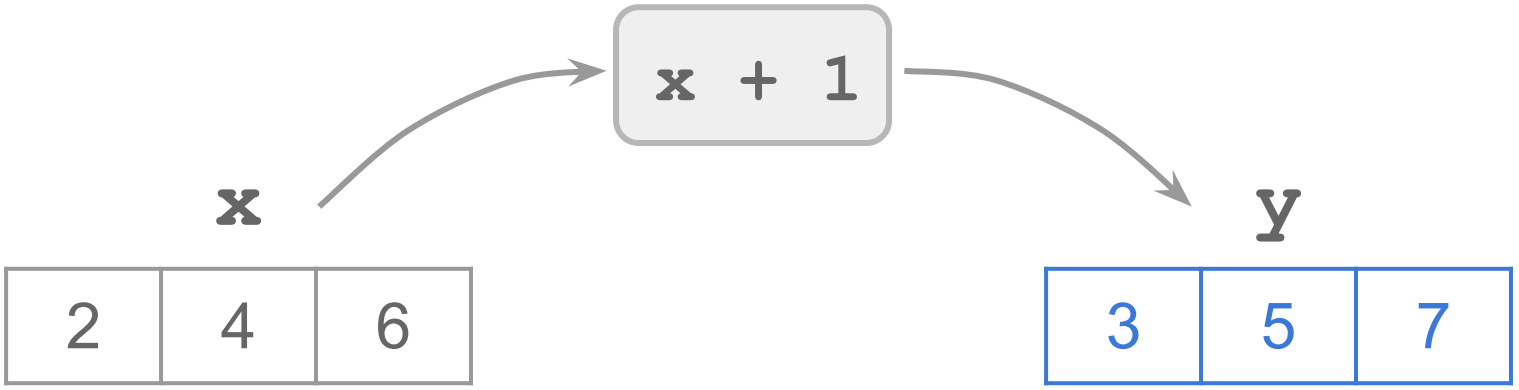
How to get ***x+1****, step-by-step, without using vectorization?*




```
# initialize empty y
y <- rep(0, 3)

y[1] <- x[1] + 1
y[2] <- x[2] + 1
y[3] <- x[3] + 1
```

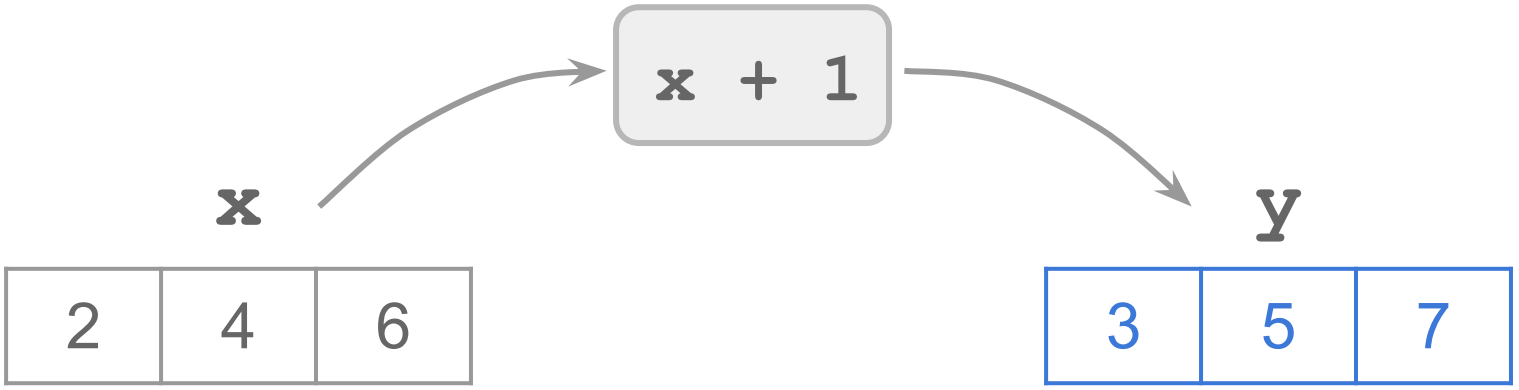
Let's use a `for` loop




```
# initialize empty y
y <- rep(0, 3)

y[1] <- x[1] + 1
y[2] <- x[2] + 1
y[3] <- x[3] + 1
```

What do all of these commands have in common?



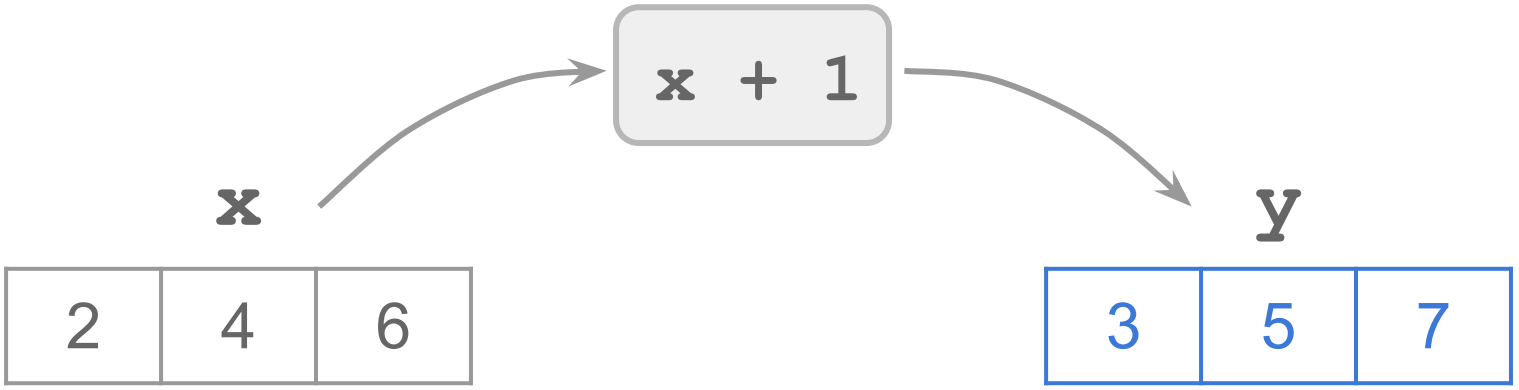

```
# initialize empty y
y <- rep(0, 3)

y[1] <- x[1] + 1
y[2] <- x[2] + 1
y[3] <- x[3] + 1
```

They all have the same format:

```
y[pos] <- x[pos] + 1
```

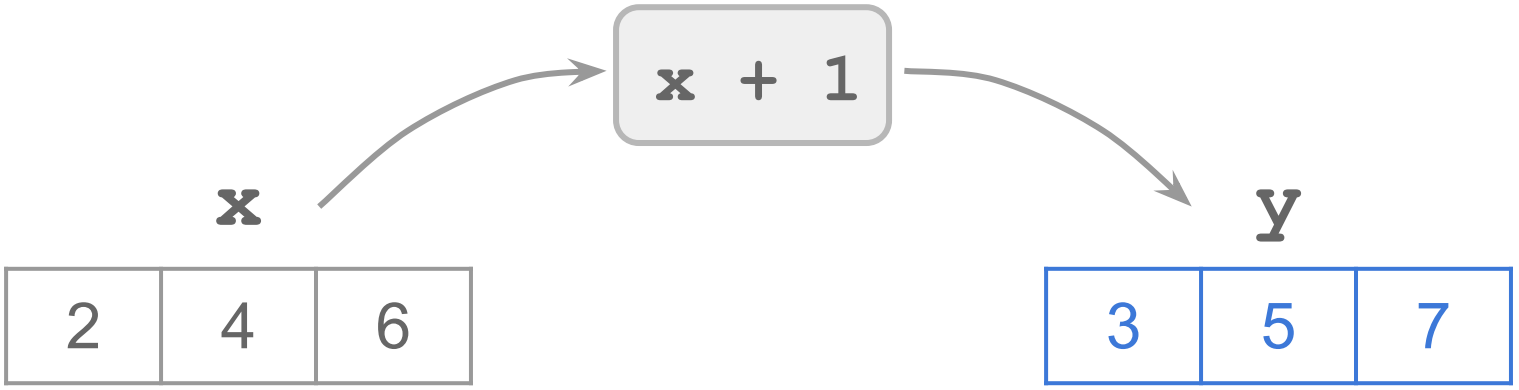
(`pos` *indicates position)*




```
# initialize empty y
y <- rep(0, 3)
```

Step 1

```
y[pos] <- x[pos] + 1
```

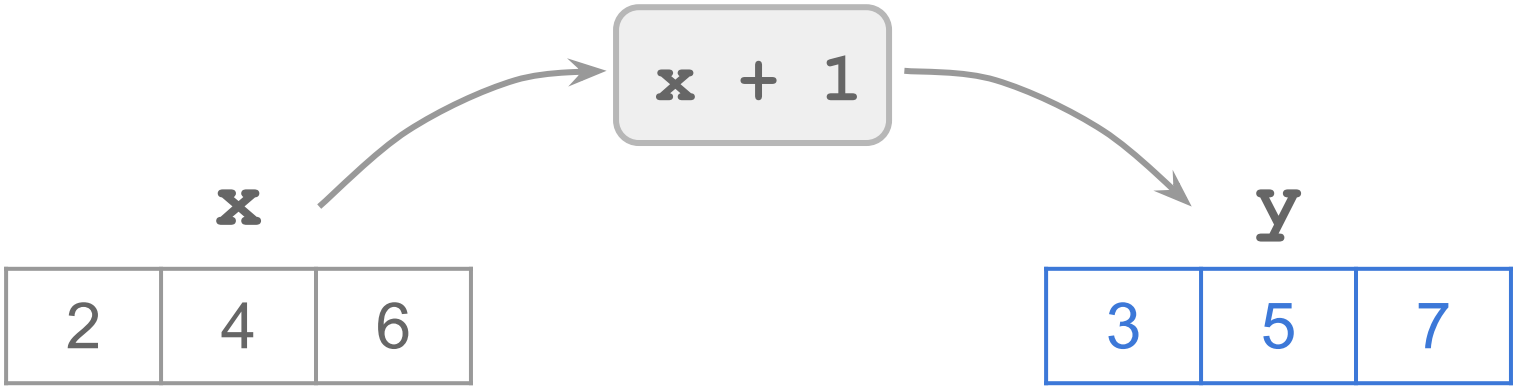



```
# initialize empty y
y <- rep(0, 3)

for (                ) {
   y[pos] <- x[pos] + 1
}
```

Step 1

Step 2



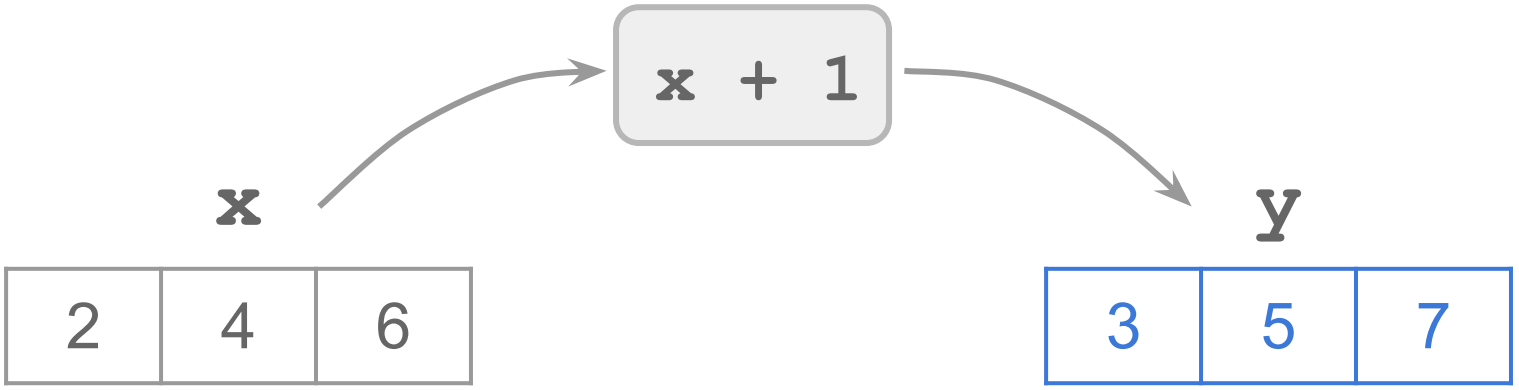

```
# initialize empty y
y <- rep(0, 3)

for (pos in     ) {
   y[pos] <- x[pos] + 1
}
```

Step 1

Step 2

Step 3




```
# initialize empty y
y <- rep(0, 3)

for (pos in 1:3) {
   y[pos] <- x[pos] + 1
}
```

Step 1

Step 2

Step 3

Step 4

Anatomy of a `for` loop

```
x <- c(2, 4, 6)
y <- rep(0, 3)

for (pos in 1:3) {
  y[pos] <- x[pos] + 1
}
```

```
x <- c(2, 4, 6)
y <- rep(0, 3)
```

for statement

```
for (pos in 1:3) {
  y[pos] <- x[pos] + 1
}
```

```
x <- c(2, 4, 6)
y <- rep(0, 3)
```

Iterator: auxiliary variable

```
for (pos in 1:3) {
  y[pos] <- x[pos] + 1
}
```

```
x <- c(2, 4, 6)
y <- rep(0, 3)
```

"in" keyword

```
for (pos in 1:3) {
  y[pos] <- x[pos] + 1
}
```

```
x <- c(2, 4, 6)
y <- rep(0, 3)
```

Vector of "times"

```
for (pos in 1:3) {
  y[pos] <- x[pos] + 1
}
```

```
x <- c(2, 4, 6)
y <- rep(0, 3)

for (pos in 1:3) {
  y[pos] <- x[pos] + 1
}
```

Code wrapped within braces
(i.e. R compound expression)

```r
x <- c(2, 4, 6)
y <- rep(0, 3)

for (pos in 1:3) {
  y[pos] <- x[pos] + 1
}
```

When to use for loops?

The characteristic situation for when to use a for-loop is when you want to perform the same operation(s) a fixed, and known, number of times.

(i.e. you know how many times you have to repeat a process)

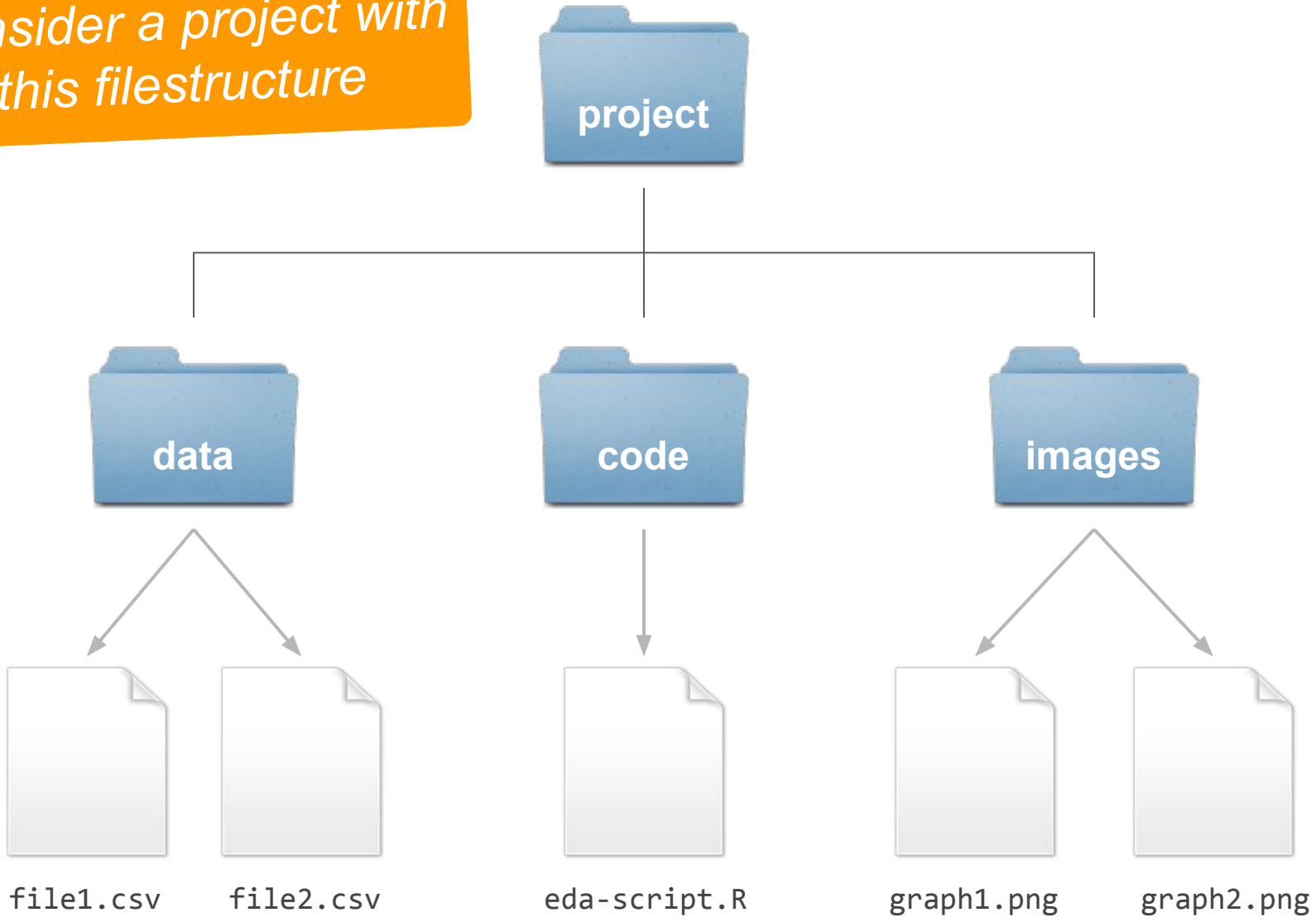
More about Loops

About

In this slides I want to discuss a couple of examples that involve using loops.

The idea is to go through less basic (and more interesting) cases for working with loops.

Consider a project with this filestructure



Filestructure

```
project/
   data/
      file1.csv
      file2.csv
   code/
      eda-script.R
   images/
      graph1.png
      graph2.png
```

Hypothetical Project

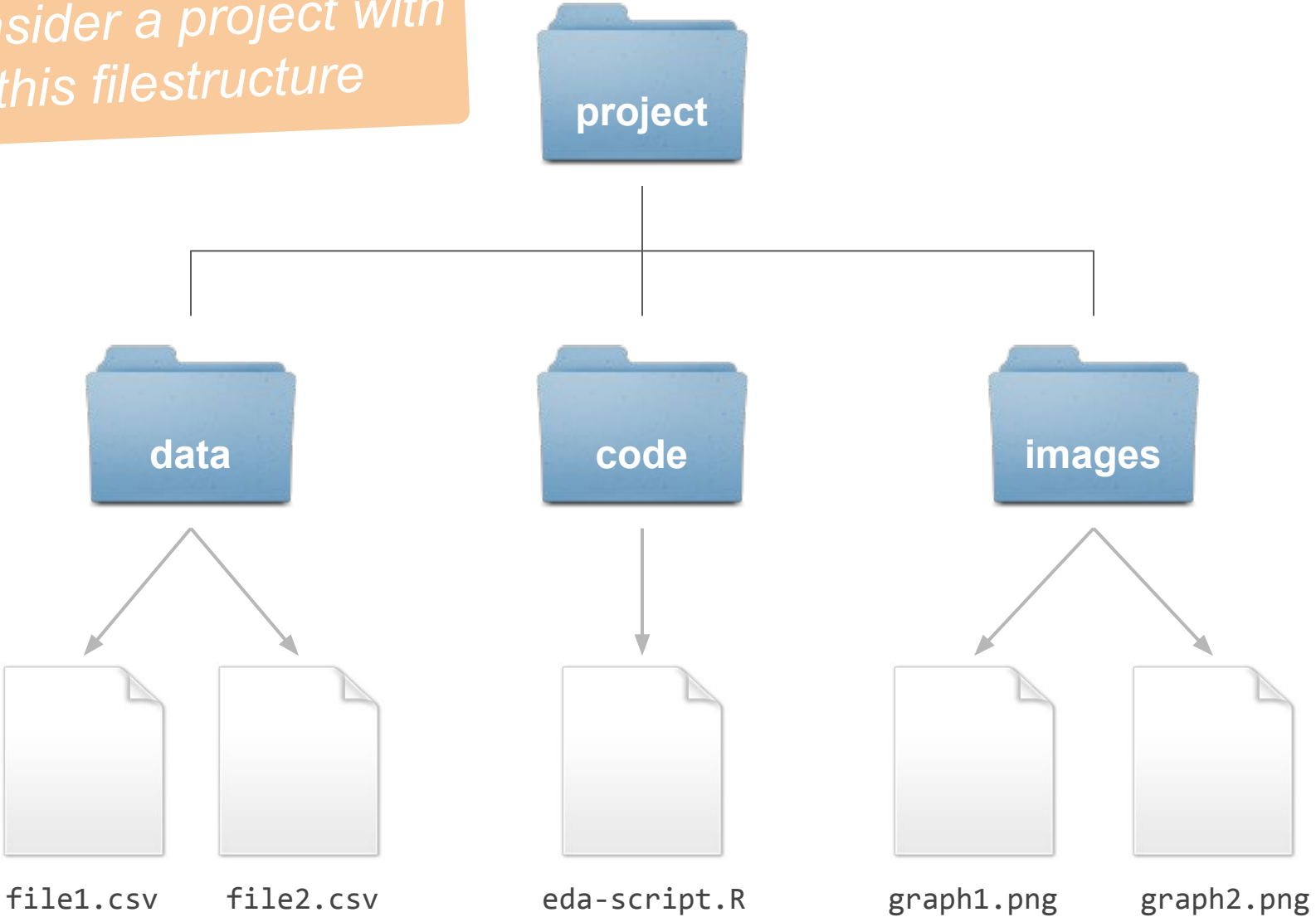
Say you have a couple of CSV data files, which are supposed to be your “raw” data files.

As part of the *Data Preparation* stage, you will have to do a little bit of exploratory data analysis (eda), e.g. creating scatterplots for each data file.

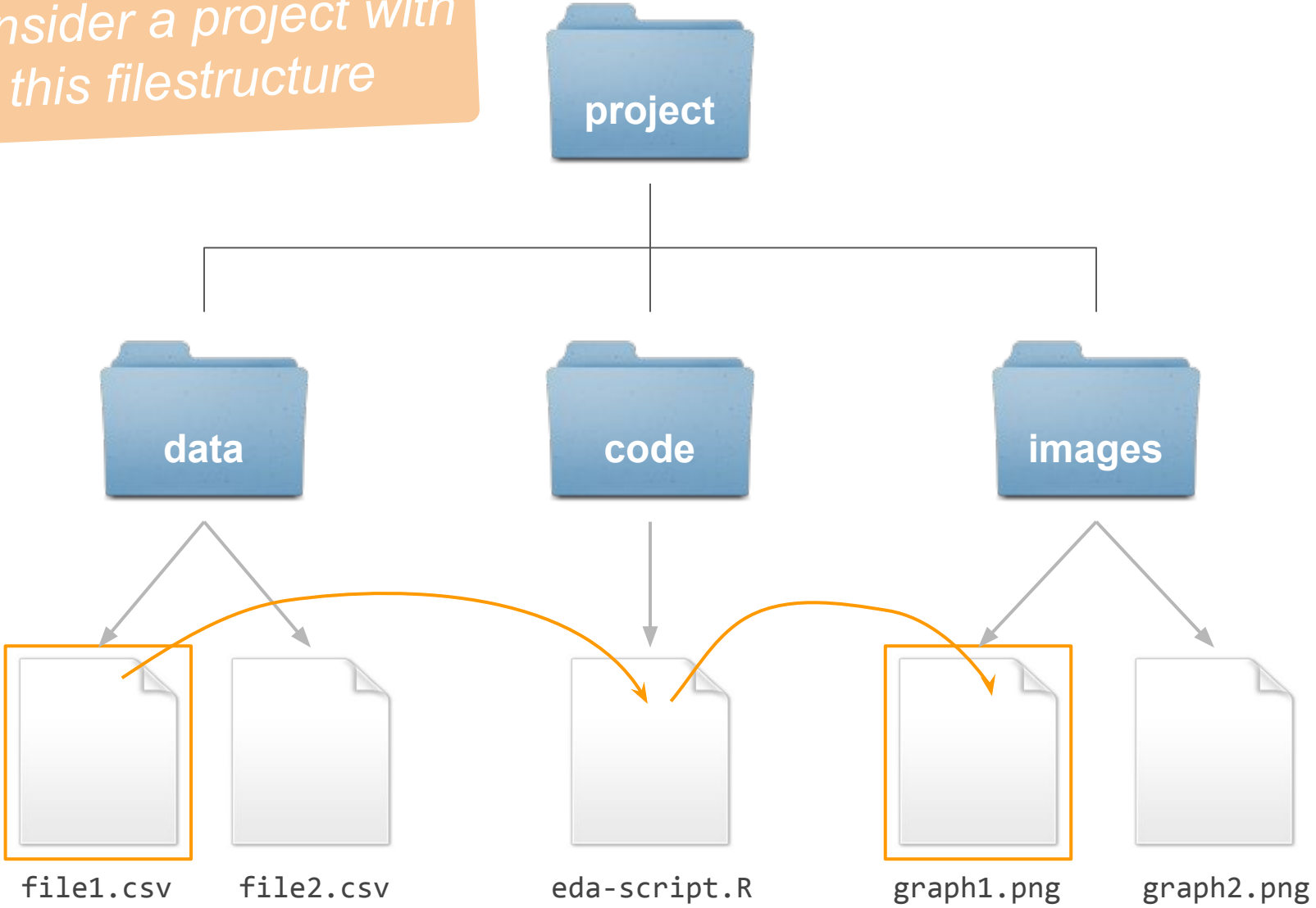
Also, suppose you will use the **`eda-script.R`** file to write the code for EDA.

This obviously involves importing (reading in) the CSV files, and making the graphs.

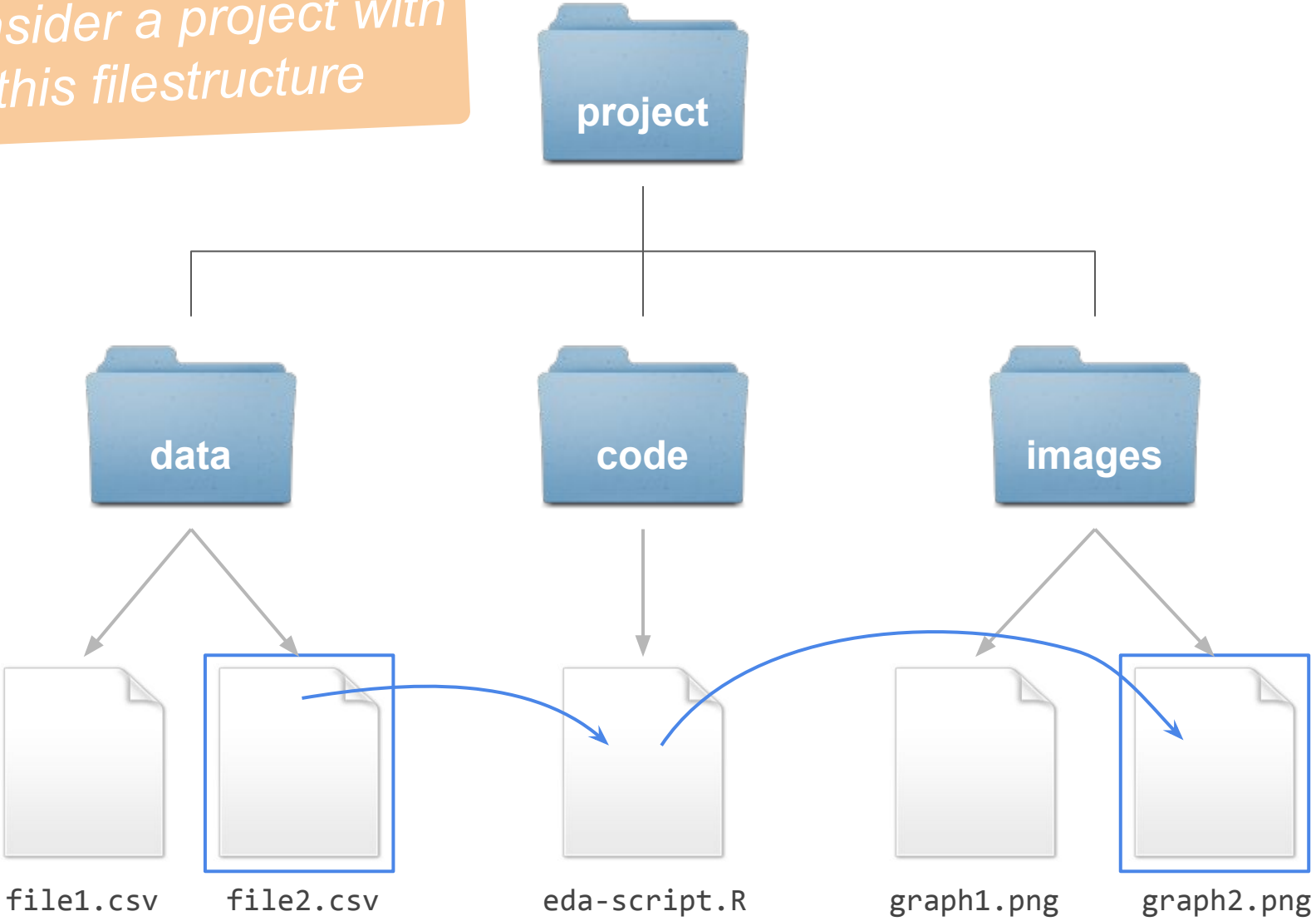
Consider a project with this filestructure



Consider a project with this filestructure



Consider a project with this filestructure

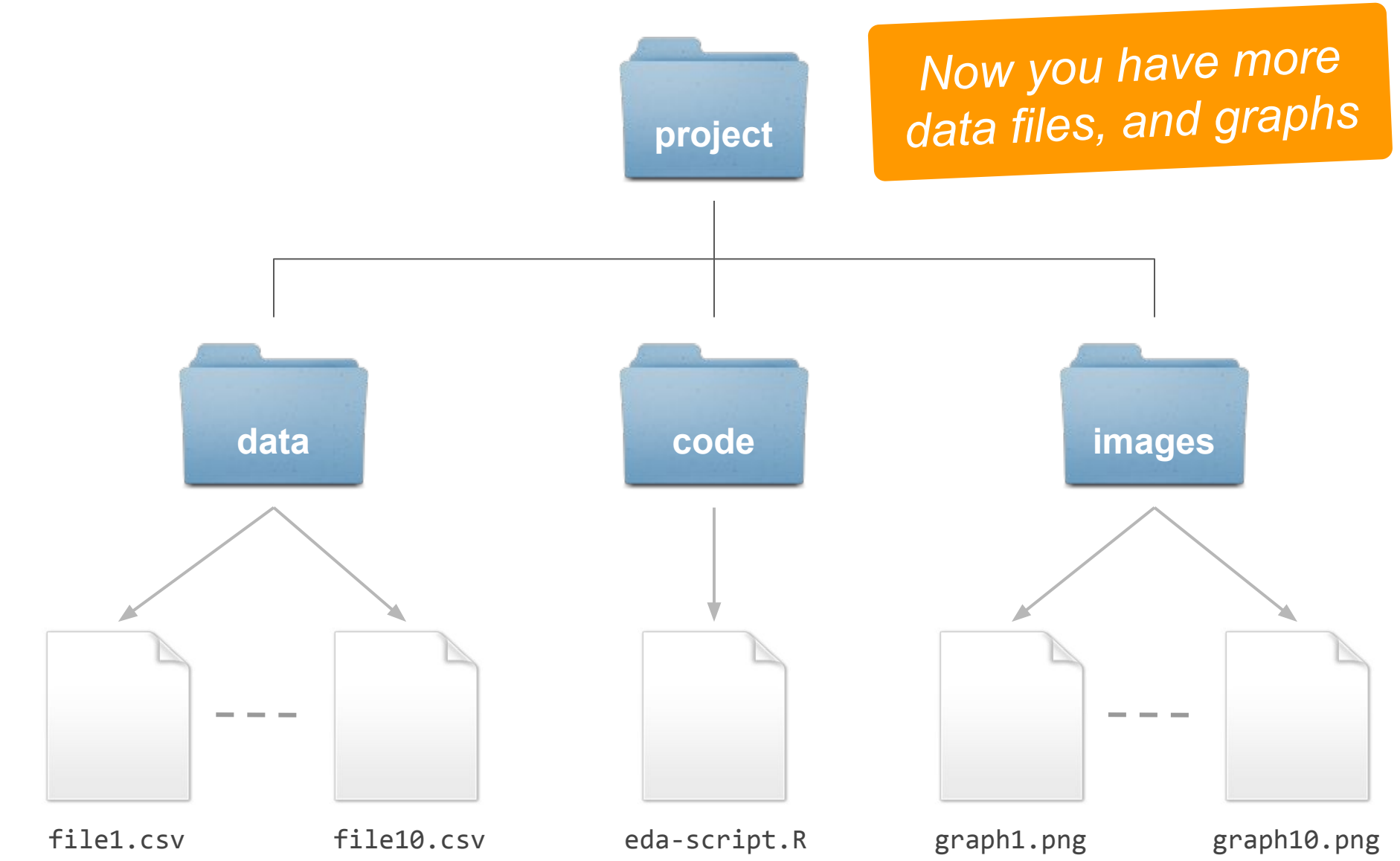


```
# importing raw data files
raw1 <- read.csv("../data/file1.csv")
raw2 <- read.csv("../data/file2.csv")

# scatterplots
graph1 <- ggplot(data = raw1) +
   geom_point(aes(x = A, y = B)) +
   labs(title = "scatter 1")
ggsave("../images/graph1.png", graph1)

graph2 <- ggplot(data = raw2) +
   geom_point(aes(x = A, y = B)) +
   labs(title = "scatter 2")
ggsave("../images/graph2.png", graph2)
```

What if you had to do the same tasks for 5, or 10, or 100 data files?


project
Now you have more data files, and graphs
data
code
images
file1.csv
file10.csv
eda-script.R
graph1.png
graph10.png

Filestructure

```
project/
   data/
      file1.csv
      ...
      file10.csv
   code/
      eda-script.R
   images/
      graph1.png
      ...
      graph10.png
```

```
# importing raw data files
raw1 <- read.csv("../data/file1.csv")
raw2 <- read.csv("../data/file2.csv")
raw3 <- read.csv("../data/file3.csv")
raw4 <- read.csv("../data/file4.csv")
raw5 <- read.csv("../data/file5.csv")
raw6 <- read.csv("../data/file6.csv")
raw7 <- read.csv("../data/file7.csv")
raw8 <- read.csv("../data/file8.csv")
raw9 <- read.csv("../data/file9.csv")
raw10 <- read.csv("../data/file10.csv")
```

Too much repetition

Imagine if you had 100 (or more) files.
This is labor intensive, time consuming,
error prone, boring … DON'T do this!

Let's use a `for` loop

```
# importing raw data files
raw1 <- read.csv("../data/file1.csv")
raw2 <- read.csv("../data/file2.csv")
raw3 <- read.csv("../data/file3.csv")
raw4 <- read.csv("../data/file4.csv")
raw5 <- read.csv("../data/file5.csv")
raw6 <- read.csv("../data/file6.csv")
raw7 <- read.csv("../data/file7.csv")
raw8 <- read.csv("../data/file8.csv")
raw9 <- read.csv("../data/file9.csv")
raw10 <- read.csv("../data/file10.csv")
```

What is it common in all these commands?

```r
# creating file names
paste0("../data/file", 1, ".csv")

paste0("../data/file", 2, ".csv")

paste0("../data/file", 3, ".csv")

...

paste0("../data/file", 10, ".csv")
```

eda-script.R

```
for (num in 1:10) {
   # importing file
   filepath <- paste0("../data/file",
                      num,
                      ".csv")
   dat <- read.csv(filepath)

   # scatterplot
   ...
}
```

Code in next slide

eda-script.R

```
for (num in 1:10) {
   # importing file
   ...

   # scatterplot
   scat <- paste0("scatter ", num)
   graph <- ggplot(data = dat) +
      geom_point(aes(x = A, y = B)) +
      labs(title = scat)



}
```

Code in previous slide

eda-script.R

```
for (num in 1:10) {
   # importing file
   ...
```

Code in previous slide

```
   # scatterplot
   scat <- paste0("scatter ", num)
   graph <- ggplot(data = dat) +
      geom_point(aes(x = A, y = B)) +
      labs(title = scat)

   gfile <- paste0("../images/graph",
                   num, ".png")
   ggsave(gfile, graph)
}
```

Putting it all together

eda-script.R

```
for (num in 1:10) {
    # importing file
    filepath <- paste0("../data/file", num, ".csv")
    dat <- read.csv(filepath)

    # scatterplot
    scat <- paste0("scatter ", num)
    graph <- ggplot(data = dat) +
        geom_point(aes(x = A, y = B)) +
        labs(title = scat)

    gfile <- paste0("../images/graph", num, ".png")
    ggsave(gfile, graph)
}
```