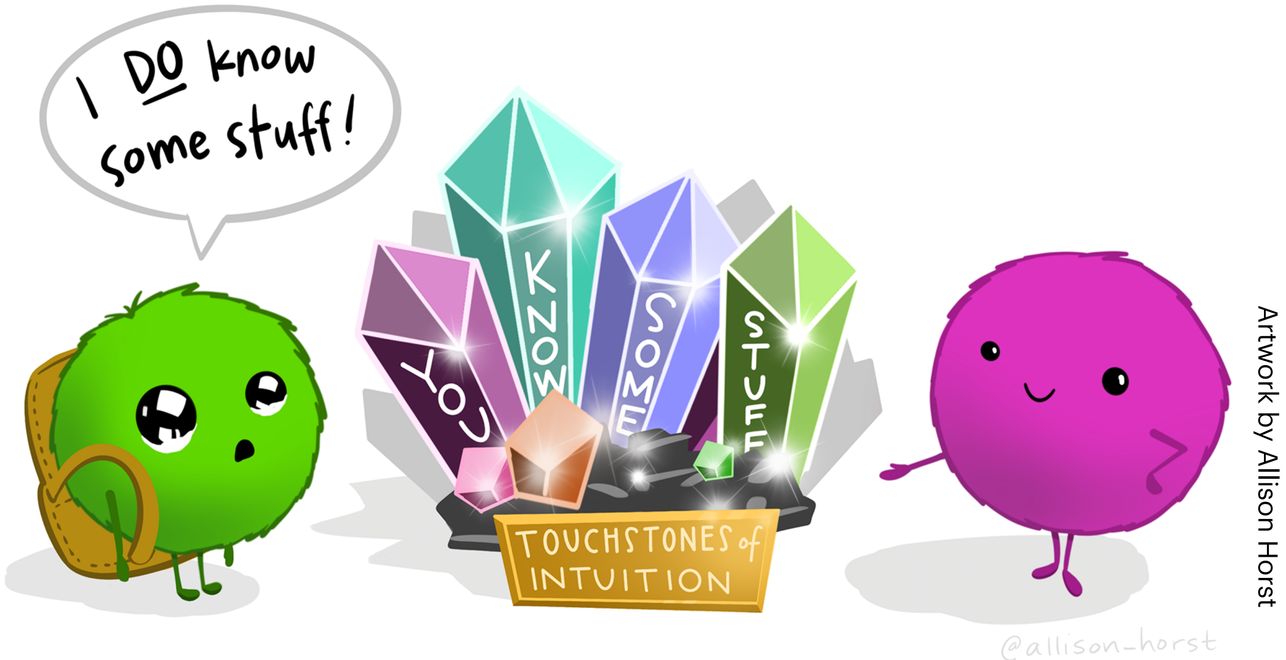
Iterations, ~~conditionals~~, and functions

Week 10 (12/03/25)



Artwork by Allison Horst

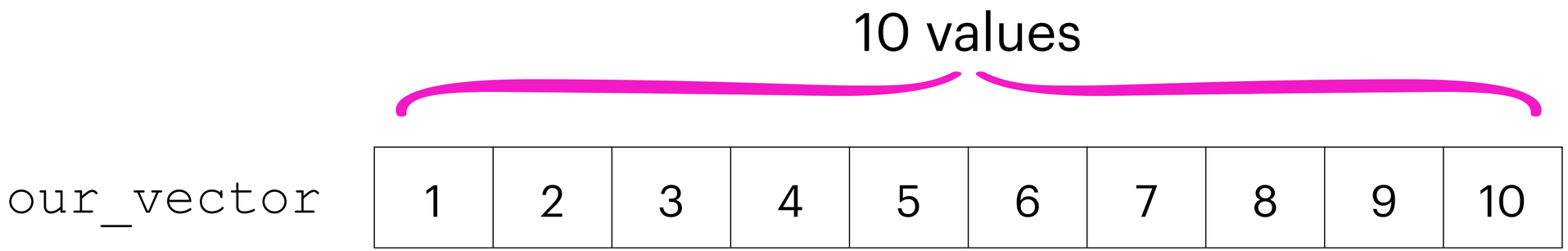
Stepfanie M. Aguillon

Outline of today's class

- Introduction to selecting elements
- Creating a `for` loop with correct syntax
- Creating a function with correct syntax

Selecting elements in R

Selecting elements in a vector

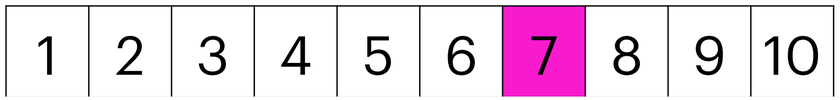



```
our_vector <- 1:10
```

Selecting vector elements with [ ]

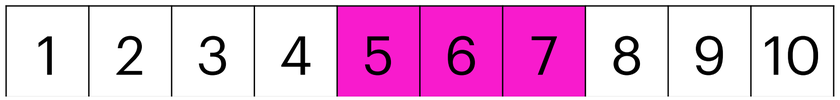
```
vector_name[index]
```

our_vector




```
our_vector[7]
```

our_vector




```
our_vector[5:7]
```

Selecting elements in a matrix

3 columns

`our_matrix`

1	5	9
2	6	10
3	7	11
4	8	12

4 rows

```
our_matrix <- matrix(1:12, nrow = 4)
```

Selecting matrix elements with [ ]

```
matrix_name[row, column]
```

our_matrix

1	5	9
2	6	10
3	**7**	11
4	8	12

```
our_matrix[3, 2]
```

our_matrix

1	**5**	9
2	**6**	10
3	**7**	11
4	**8**	12

```
our_matrix[ , 2]
```

our_matrix

1	5	9
2	**6**	**10**
3	7	11
4	8	12

```
our_matrix[2, ]
```

Selecting matrix elements with [ ]

```
matrix_name[row, column]
```

our_matrix

1	5	9
2	6	10
3	7	11
4	8	12

```
our_matrix[ , 2:3]
```

our_matrix

1	5	9
2	6	10
3	7	11
4	8	12

```
our_matrix[3:4, ]
```

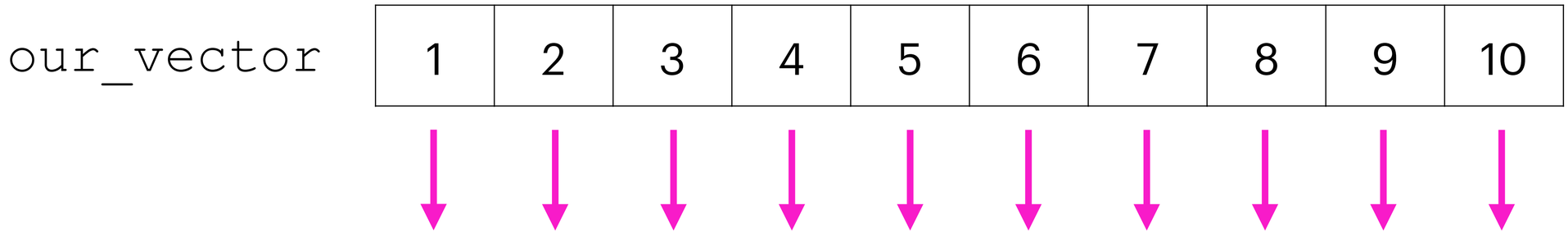
our_matrix

1	5	9
2	6	10
3	7	11
4	8	12

```
our_matrix[3:4, 2:3]
```

The basics of `for` loops

Why use a `for` loop?



multiply by 2

```
our_vector[1]*2
    our_vector[2]*2
        our_vector[3]*2
            our_vector[4]*2
                our_vector[5]*2
```

```
our_vector[6]*2
    our_vector[7]*2
        our_vector[8]*2
            our_vector[9]*2
                our_vector[10]*2
```

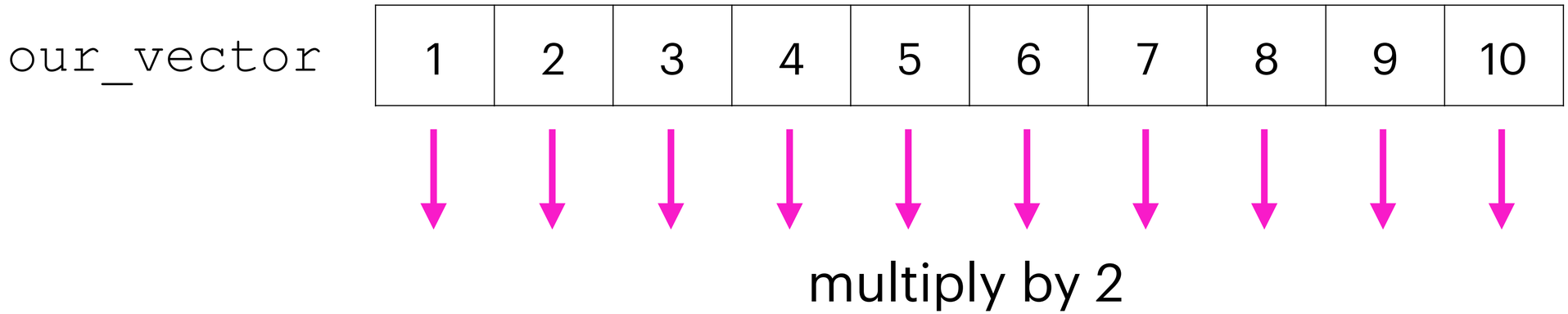
Basic syntax of a `for` loop

(parentheses surround what you're iterating over)

```
for (an_element in a_vector){
  do_a_thing
}
```

{ curly brackets surround what you're doing to the elements }

Basic syntax of a `for` loop




```
our_vector[i]*2
```

```
for (i in 1:length(our_vector)){
  print(our_vector[i]*2)
}
```

Using a `for` loop in a matrix

`our_matrix`

1	5	9
2	6	10
3	7	11
4	8	12

multiply column 3 values by 2

and write new values in `our_matrix`

```
for (an_element in a_vector){
  do_a_thing
}
```

Take `for` loops step-by-step!

1. **What is the "thing" you're trying to do?** What action do you want taken on each element?

2. **What elements are you iterating over?** How can you represent this as a vector within your loop? Can you do this in a repeatable way?

3. **How do you want the output to be saved?** In a new vector? Displayed in the console? Included in the original dataset?

4. **Test on a smaller problem**. Are there ways to break your bigger goal down into smaller sub-steps?

Using a `for` loop in a matrix

`our_matrix`

1	5	9
2	6	10
3	7	11
4	8	12

multiply column 3 values by 2

and write new values in `our_matrix`

```
for (an_element in a_vector){
  do_a_thing
}
```

1. What is the “thing” you’re trying to do?
2. What elements are you iterating over?
3. How do you want the output to be saved?

Using a `for` loop in a matrix

`our_matrix`

1	5	9
2	6	10
3	7	11
4	8	12

multiply column 3 values by 2

and write new values in `our_matrix`

```
for (i in 1:nrow(our_matrix)){
  our_matrix[i,3] <- our_matrix[i,3]*2
}
```

The basics of functions

What is a function?

"Functions allow you to automate common tasks in a more powerful and general way than copy-and-pasting."

R for Data Science

What do you need to *write* a function?

1. **Function Name:** What to call your function? Try for something descriptive!

2. **Arguments:** The input that varies with the use of the function.

3. **Body:** The code that is run on the arguments when the function is called.

R for Data Science

General syntax for functions

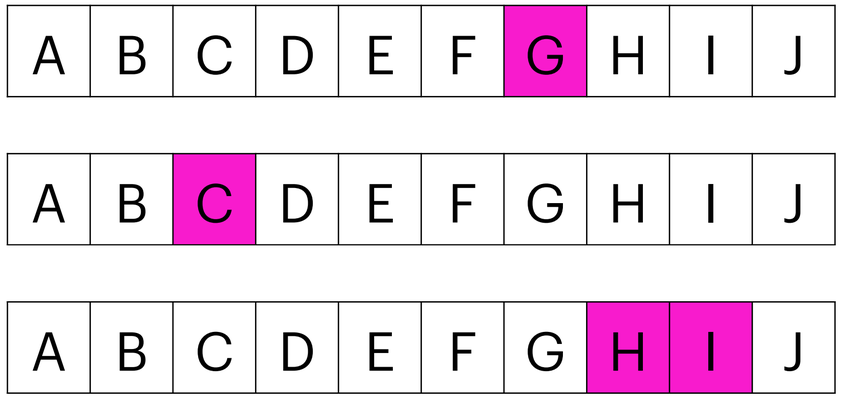
1 function name

2 (arguments)

```
function_name <- function(input_arguments) {
  output_value <- do_a_thing_with_input_arguments
  return(output_value)
}
```

3 {body}

Let's write our first function!

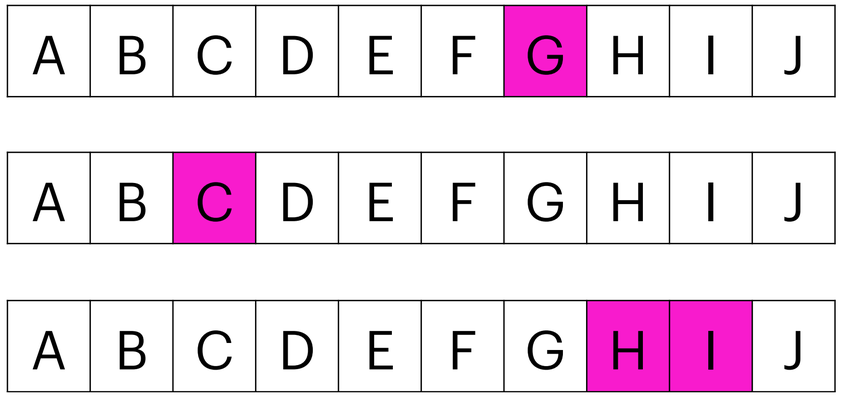



```
vector[7]
vector[3]
vector[8:9]
```

Let's make a function!

1. **Function Name** — `vector_value`
2. **(Input Arguments)** — `a vector, index`
3. **{ Body }** — `similar code to above`

Let's write our first function!




```
vector[7]
vector[3]
vector[8:9]
```

Let's make a function!

1 (2)

```
vector_value <- function(a_vector, index) {
   x <- a_vector[index]
   return(x)
}
```

{ 3 }

notice that the arguments match those included in the body!

Using the function!

Running this saves the function to your *environment*

```
vector_value <- function(a_vector, index) {
  x <- a_vector[index]
  return(x)
}
```

Let’s use the function on some vectors!

```
vector_value(our_vector, 7)
```

Default values of an argument

```
vector_value <- function(a_vector, index = 1) {
  x <- a_vector[index]
  return(x)
}
```

Use an equal sign to set a default value for an argument

```
vector_value(our_vector)
```

Integrating functions with ggplot2

What do you think this function does? And how do we use it?

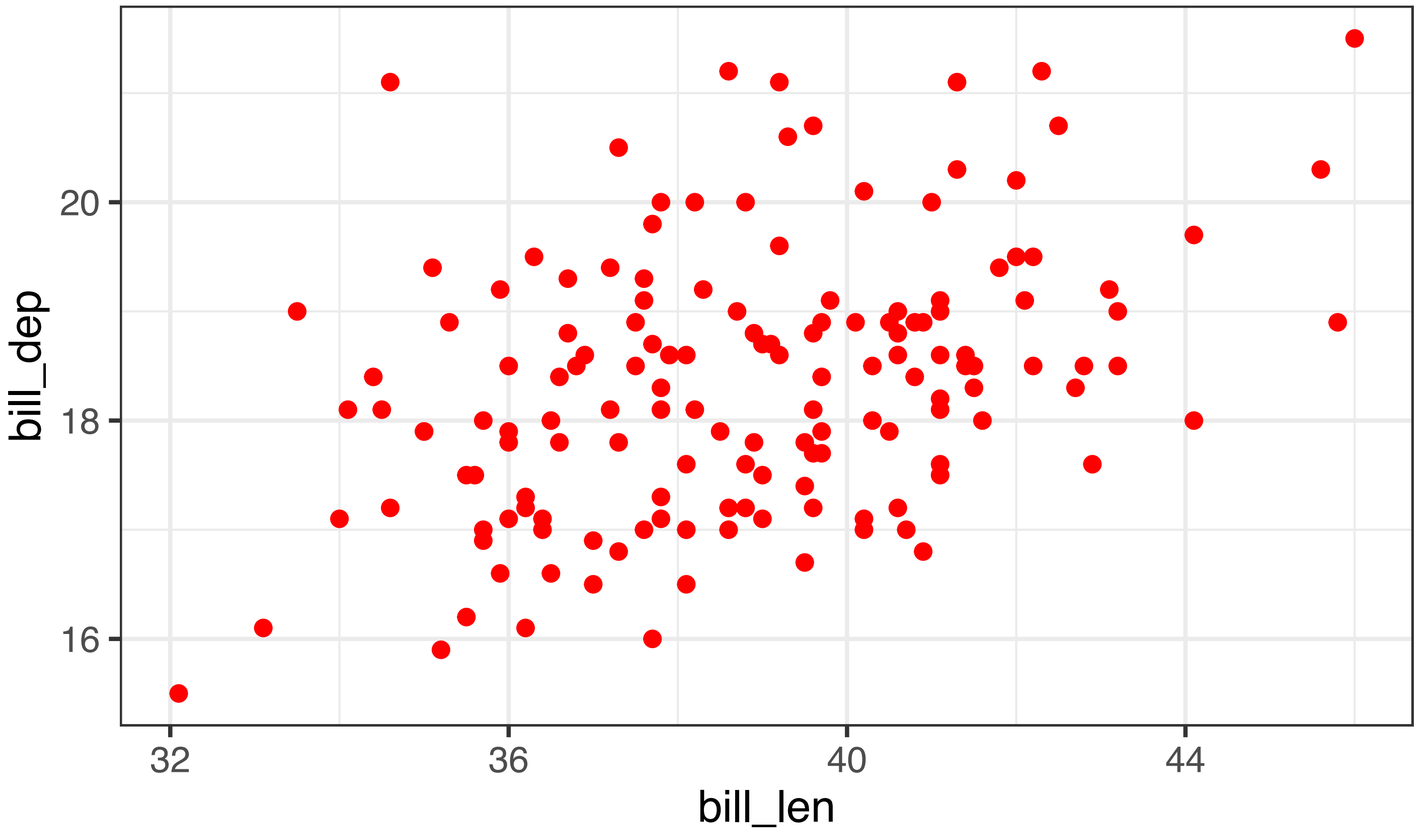
```
penguin_plot <- function(df, species_name, color_name = "red") {
   df %>%
      filter(species == species_name) %>%
      ggplot(mapping = aes(x = bill_len, y = bill_dep)) +
      geom_point(color = color_name) +
      theme_bw()
}
```

```
penguin_plot(penguins, "Adelie")
```

```
penguin_plot(penguins, "Gentoo", "blue")
```

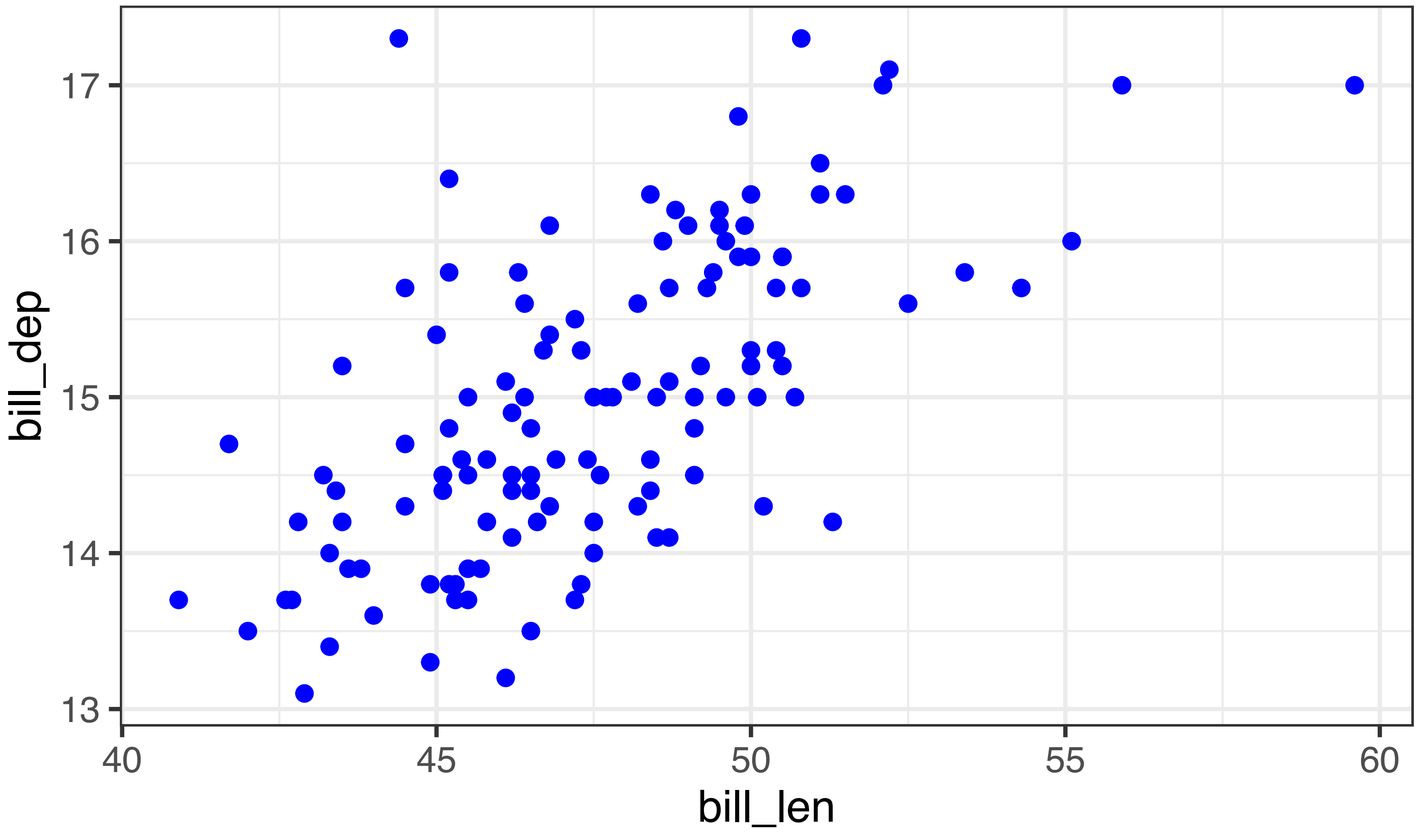
Integrating functions with ggplot2

```
penguin_plot(penguins, "Adelie")
```



Integrating functions with ggplot2

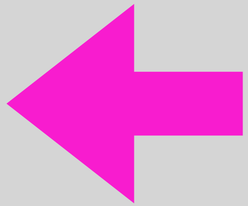
```
penguin_plot(penguins, "Gentoo", "blue")
```



Create your own `ggplot` theme!

Create your custom theme with a function:

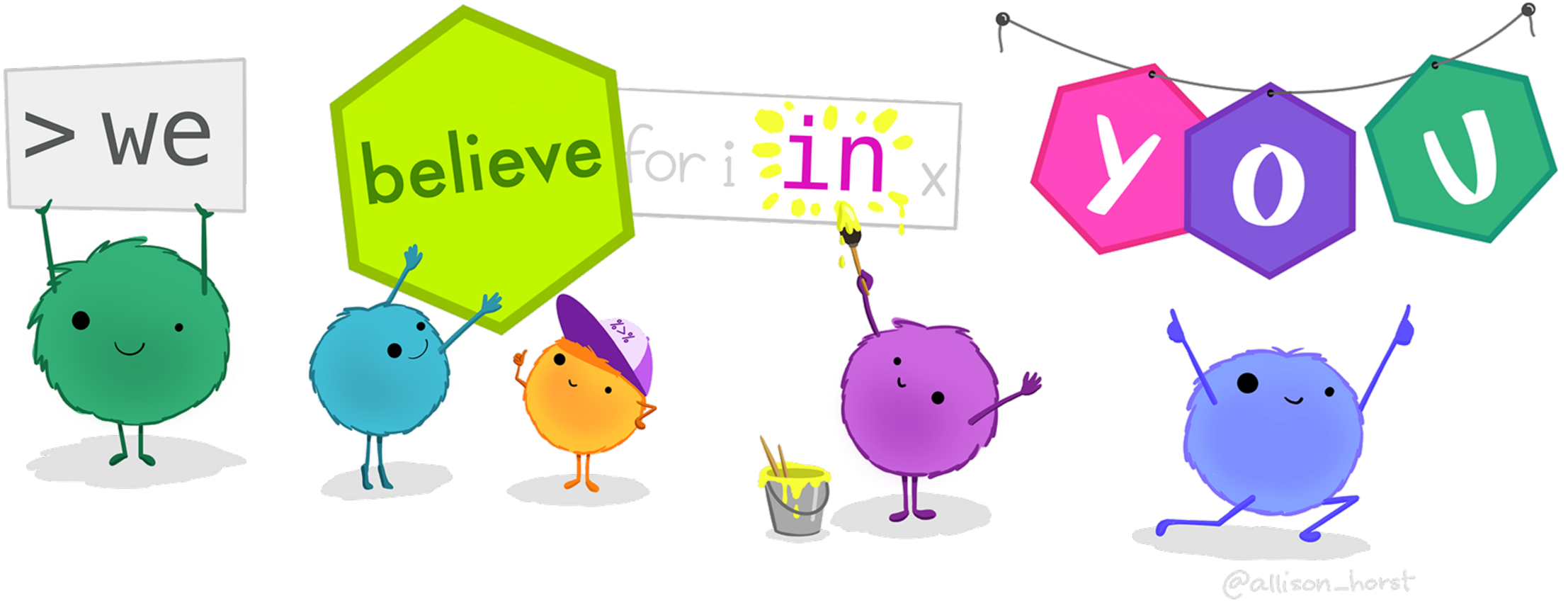
```
my_theme <- function() {
      theme(
         # theme elements go here...
      )
}
```



add any theme elements you'd like!

Call your theme at the end of your `ggplot`:

```
ggplot(<DATA>, aes(x = <VARIABLE>, y = <VARIABLE>)) +
   geom_point() +
   my_theme()
```



Artwork by Allison Horst

Further reading

- Wickham & Grolemund 2017, *R for Data Science*, Chapters 25-27
  - Chapter 25 = Functions
  - Chapter 26 = Iteration
  - Chapter 27 = A field guide to base R
- Wickham 2019, *Advanced R*, Chapter 4