# Data Science for Everyone – Functions

Dr. Ab Mosca (they/them)

Slides based off slides courtesy of Jordan Crouser (https://jcrouser.github.io/)

## Plan for Today

- Built in functions
- User defined functions

## Repetition

- Suppose you're a data scientist given weekly datasets to analyze
- The datasets are similar; each is a `tbl` with three variables
- They come un-tidy, so first you must make them tidy
- For three weeks use the following code:

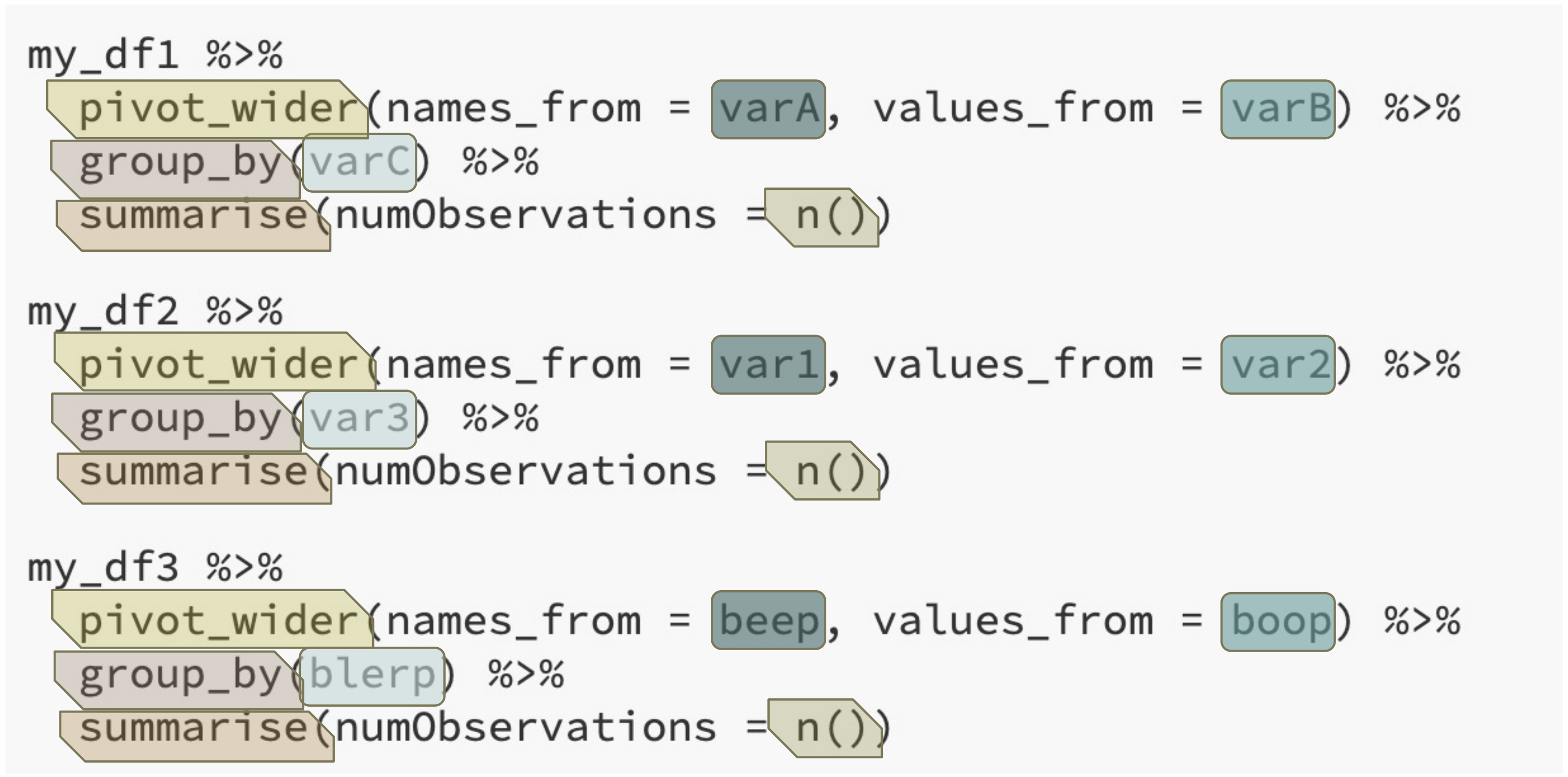
```r
my_df1 %>%
  pivot_wider(names_from = varA, values_from = varB) %>%
  group_by(varC) %>%
  summarise(numObservations = n())

my_df2 %>%
  pivot_wider(names_from = var1, values_from = var2) %>%
  group_by(var3) %>%
  summarise(numObservations = n())

my_df3 %>%
  pivot_wider(names_from = beep, values_from = boop) %>%
  group_by(blerp) %>%
  summarise(numObservations = n())
```

## Repetition

```
my_df1 %>%
  pivot_wider(names_from = varA, values_from = varB) %>%
  group_by(varC) %>%
  summarise(numObservations = n())

my_df2 %>%
  pivot_wider(names_from = var1, values_from = var2) %>%
  group_by(var3) %>%
  summarise(numObservations = n())

my_df3 %>%
  pivot_wider(names_from = beep, values_from = boop) %>%
  group_by(blerp) %>%
  summarise(numObservations = n())
```

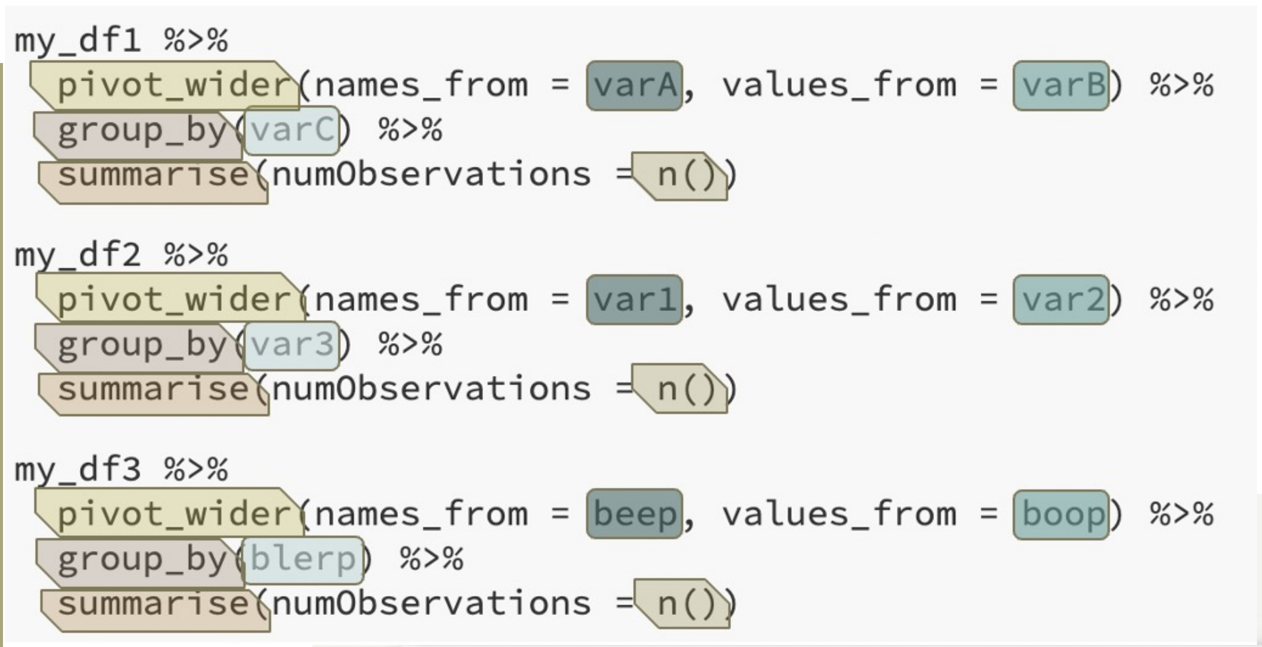
Work with the person next to you to find similarities in each code chunk above. Is there a common set of steps you take for each dataset?

## Repetition

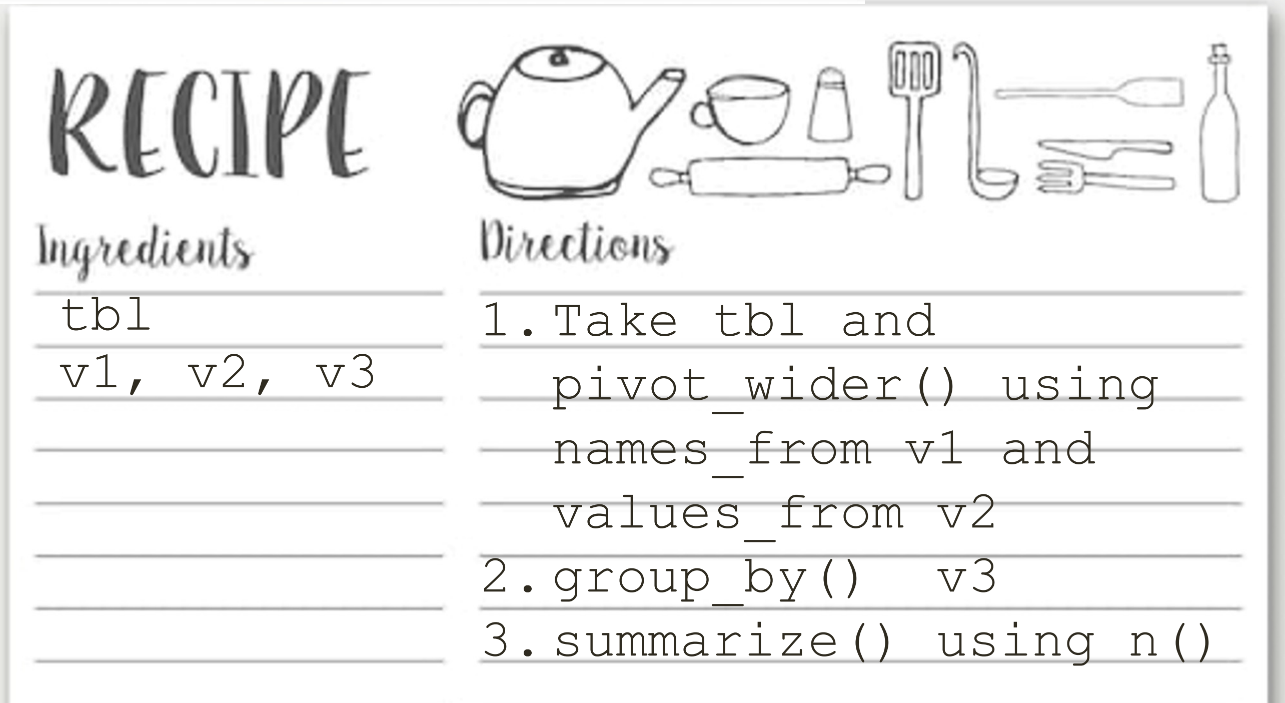


```
my_df1 %>%
  pivot_wider(names_from = varA, values_from = varB) %>%
  group_by(varC) %>%
  summarise(numObservations = n())

my_df2 %>%
  pivot_wider(names_from = var1, values_from = var2) %>%
  group_by(var3) %>%
  summarise(numObservations = n())

my_df3 %>%
  pivot_wider(names_from = beep, values_from = boop) %>%
  group_by(blerp) %>%
  summarise(numObservations = n())
```

## Repetition



```
my_df1 %>%
  pivot_wider(names_from = varA, values_from = varB) %>%
  group_by(varC) %>%
  summarise(numObservations = n())

my_df2 %>%
  pivot_wider(names_from = var1, values_from = var2) %>%
  group_by(var3) %>%
  summarise(numObservations = n())

my_df3 %>%
  pivot_wider(names_from = beep, values_from = boop) %>%
  group_by(blerp) %>%
  summarise(numObservations = n())
```



RECIPE

Ingredients

tbl

v1, v2, v3

Directions

1. Take tbl and pivot_wider() using names_from v1 and values_from v2
2. group_by() v3
3. summarize() using n()

## Functions

- Format:
  - `function_name(argument1, argument2, …)`
- Inputs
  - Arguments
  - Things like: `tbl, string, int`, etc.
- Output
  - Things like: `tbl, string, int`, etc.
- We've been using built in functions! Ex.
  - `ncol(babynames)`
    - Input = `tbl (babynames)`
    - Output = `int` (number of columns in `babynames`)

## Functions

- Format:
  - `function_name(argument1, argument2, …)`
- Inputs
  - Arguments
  - Things like: `tbl`, `string`, `int`, etc.
- Output
  - Things like: `tbl`, `string`, `int`, etc.
- We've been using built in functions! Ex.
  - `ncol(babynames)`
    - Input = `tbl (babynames)`
    - Output = `int` (number of columns in `babynames`)

What other built in functions have we used? What is the input and output?

## User-defined Functions

- We can (temporarily) add functions to R by defining them ourselves
    - We call these user-defined functions

- This is ***very*** useful for repetitive tasks

## User-defined Functions

### Defining your own functions

```
name_of_function <- function(data, var = "value") {
    ...
    ...
    <valid R code>
    ...
    ...
    return(x)
}
```

User-defined Functions

## Defining your own functions

```
name_of_function <- function(data, var = "value") {
    ...
    ...
    <valid R code>
    ...
    ...
    return(x)
}
```

What you want the function to be called

## User-defined Functions

### Defining your own functions

```
name_of_function <- function(data, var = "value") {
    ...
    ...
    <valid R code>
    ...
    ...
    return(x)
}
```

`function` is the key word that tells R "I'm creating a function"

`<-` assigns the function definition to the variable `name_of_function`

## User-defined Functions

## Defining your own functions

```
name_of_function <- function(data, var = "value") {
    ...
    ...
    <valid R code>
    ...
    ...
    return(x)
}
```

Input / arguments

### Inputs

- *arguments*: data, var
- data is required
- var is optional -- has a default value of "value"

User-defined Functions

## Defining your own functions

```
name_of_function <- function(data, var = "value") {
   ...
   ...
   <valid R code>
   ...
   ...
   return(x)
}
```

Body

Body

- Defines what the function should do
- Everything between { and }

User-defined Functions

## Defining your own functions

```
name_of_function <- function(data, var = "value") {
    ...
    ...
    <valid R code>
    ...
    ...
    return(x)
}
```

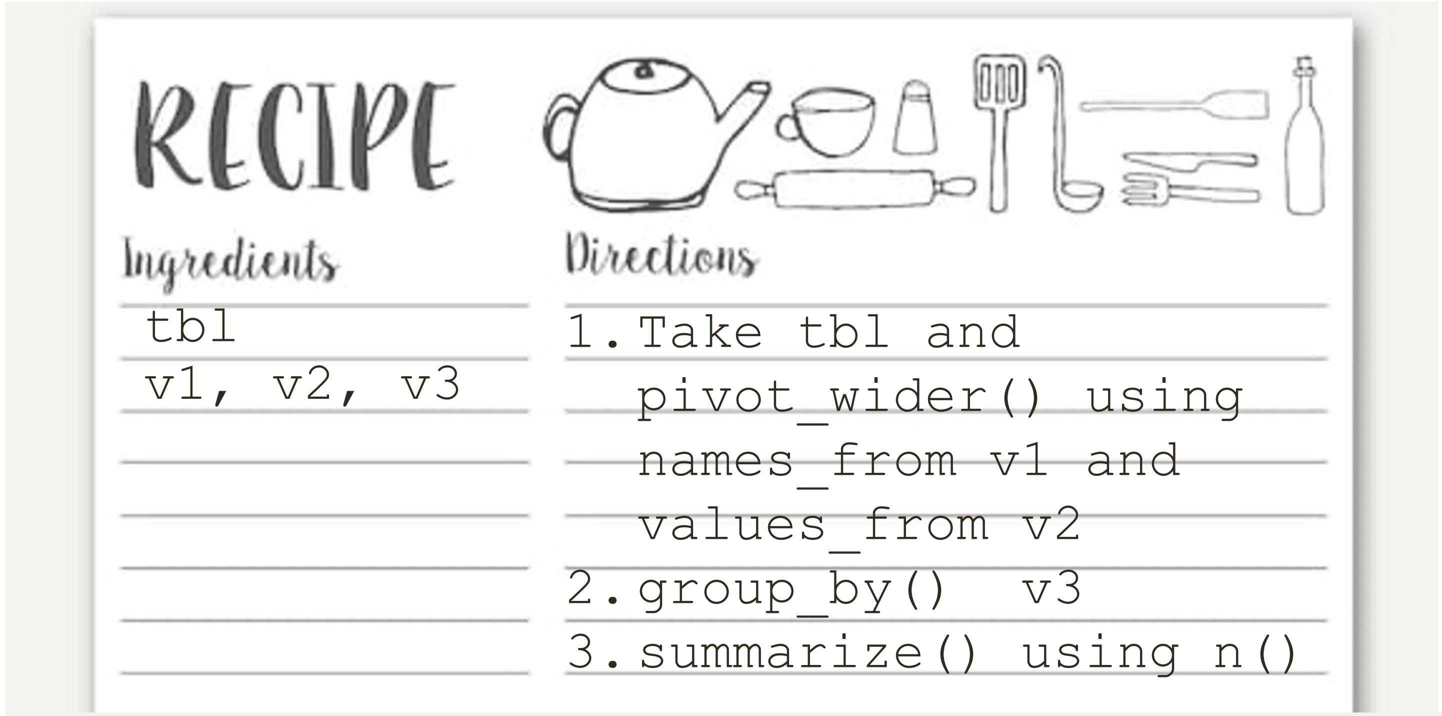
Output

## Output

- the *return* value
- by default output of last line in function body
- here, explicitly the object x

## User-defined Functions

Work with the person next to you to turn our recipe into a function.


RECIPE

Ingredients

tbl
v1, v2, v3

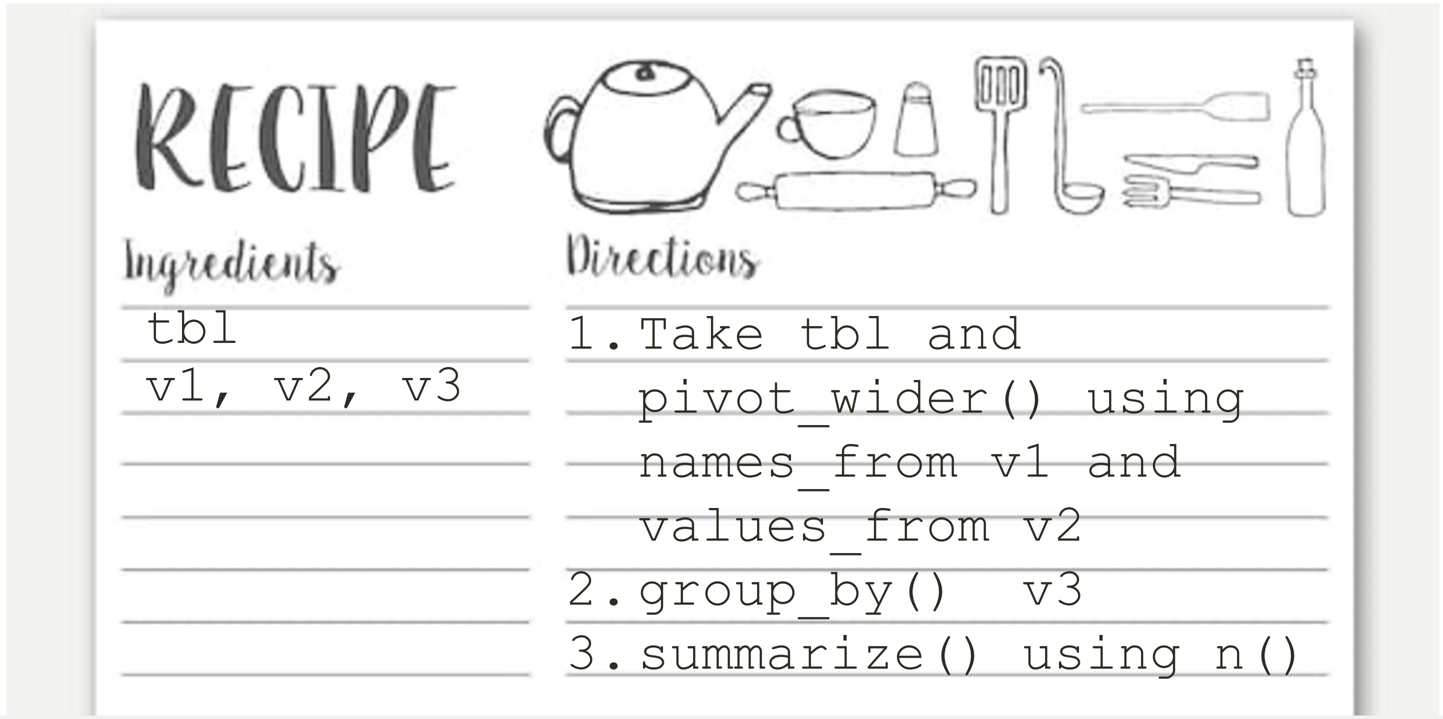
Directions

1. Take tbl and pivot_wider() using names_from v1 and values_from v2
2. group_by() v3
3. summarize() using n()

```
name_of_function <- function(data, var = "value") {
   ...
   <valid R code>
   ...
   return(x)
}
```

# User-defined Functions

Work with the person next to you to turn our recipe into a function.



RECIPE

Ingredients

tbl

v1, v2, v3

Directions

1. Take tbl and pivot_wider() using names_from v1 and values_from v2
2. group_by() v3
3. summarize() using n()

```
clean_data <- function(data, v1, v2, v3) {
   data %>%
      pivot_wider(names_from = v1, values_from = v2) %>%
      group_by(v3) %>%
      summarize(numObservations = n())
}
```

User-defined Functions

## Defining your own functions

- To use a function you defined, you "call" it with the appropriate arguments
- Ex. Let's call `clean_data()` to make `my_df1` from earlier tidy

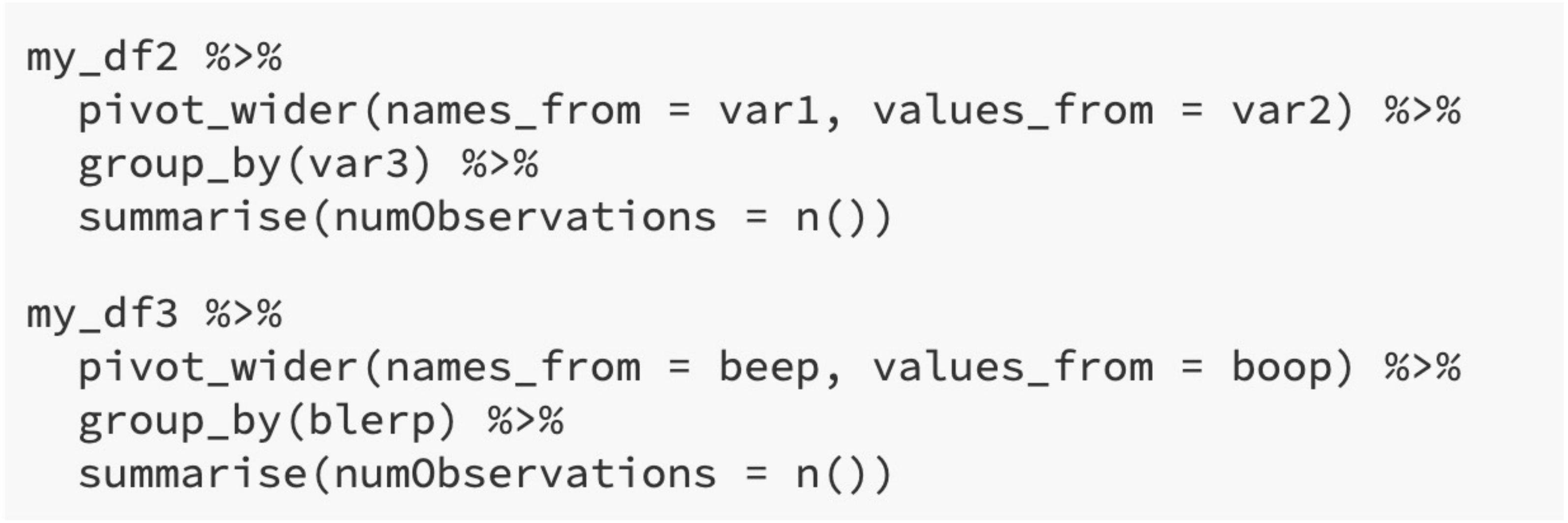
```
my_df1 %>%
  pivot_wider(names_from = varA, values_from = varB) %>%
  group_by(varC) %>%
  summarise(numObservations = n())
```

```
# Define clean_data
clean_data <- function(data, v1, v2, v3) {
    data %>%
        pivot_wider(names_from = v1, values_from = v2) %>%
        group_by(v3) %>%
        summarize(numObservations = n())
}
# Call clean_data
my_tidy_df1 <- clean_data(my_df1, varA, varB, varC)
```

## User-defined Functions

```
# Define clean_data
clean_data <- function(data, v1, v2, v3) {
    data %>%
        pivot_wider(names_from = v1, values_from = v2) %>%
        group_by(v3) %>%
        summarize(numObservations = n())
}
```

Work with the person next to you call `clean_data` for `my_df2` and `my_df3`.



```
my_df2 %>%
  pivot_wider(names_from = var1, values_from = var2) %>%
  group_by(var3) %>%
  summarise(numObservations = n())

my_df3 %>%
  pivot_wider(names_from = beep, values_from = boop) %>%
  group_by(blerp) %>%
  summarise(numObservations = n())
```

## User-defined Functions

```
# Define clean_data
clean_data <- function(data, v1, v2, v3) {
    data %>%
        pivot_wider(names_from = v1, values_from = v2) %>%
        group_by(v3) %>%
        summarize(numObservations = n())
}
```

Work with the person next to you call `clean_data` for `my_df2` and `my_df3`.

```
my_df2 %>%
  pivot_wider(names_from = var1, values_from = var2) %>%
  group_by(var3) %>%
  summarise(numObservations = n())

my_df3 %>%
  pivot_wider(names_from = beep, values_from = boop) %>%
  group_by(blerp) %>%
  summarise(numObservations = n())
```

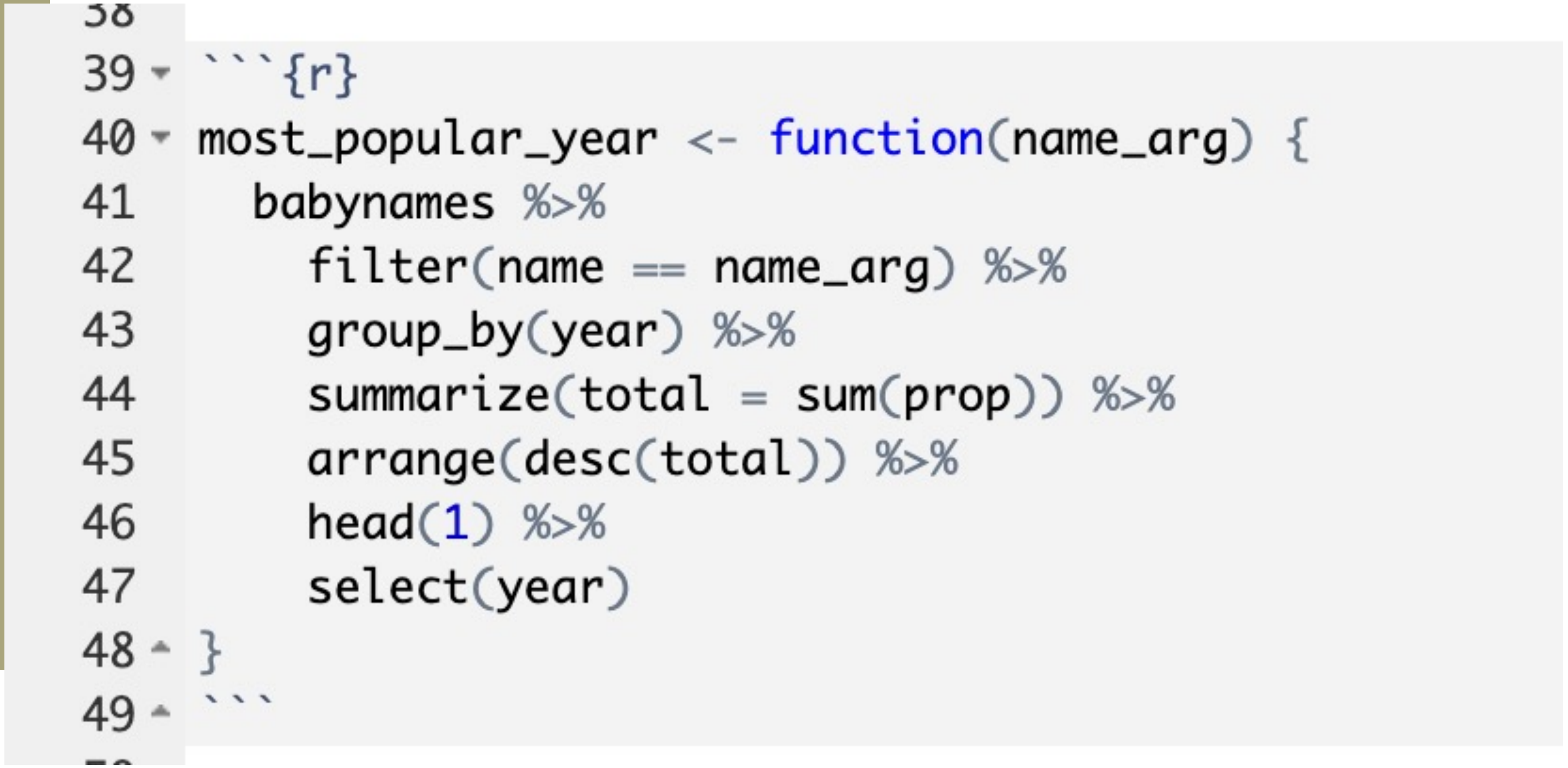
```
my_tidy_df2 <- clean_data(my_df2,
                          var1,
                          var2,
                          var3)

my_tidy_df3 <- clean_data(my_df3,
                          beep,
                          boop,
                          blerp)
```

## Practice

- Write a function that uses the babynames dataset to find the year a baby name was most popular. The function should work for any name.
    - What are the inputs?
    - What will be returned?
    - What steps does the function need to take?

## Practice

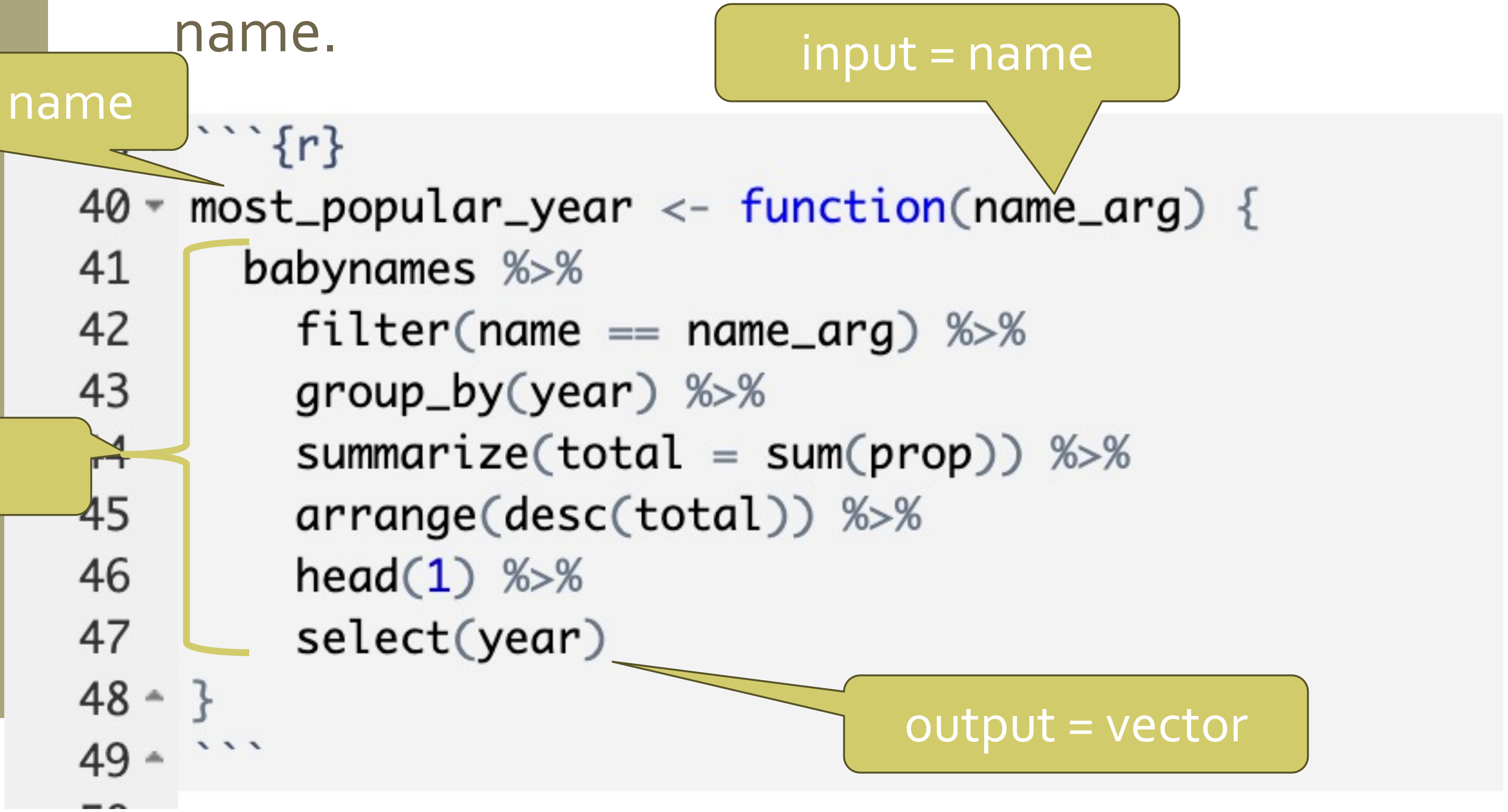
- Write a function that uses the babynames dataset to find the year a baby name was most popular. The function should work for any name.



~~~
```{r}
most_popular_year <- function(name_arg) {
  babynames %>%
    filter(name == name_arg) %>%
    group_by(year) %>%
    summarize(total = sum(prop)) %>%
    arrange(desc(total)) %>%
    head(1) %>%
    select(year)
}
```
~~~

## Practice

- Write a function that uses the babynames dataset to find the year a baby name was most popular. The function should work for any name.



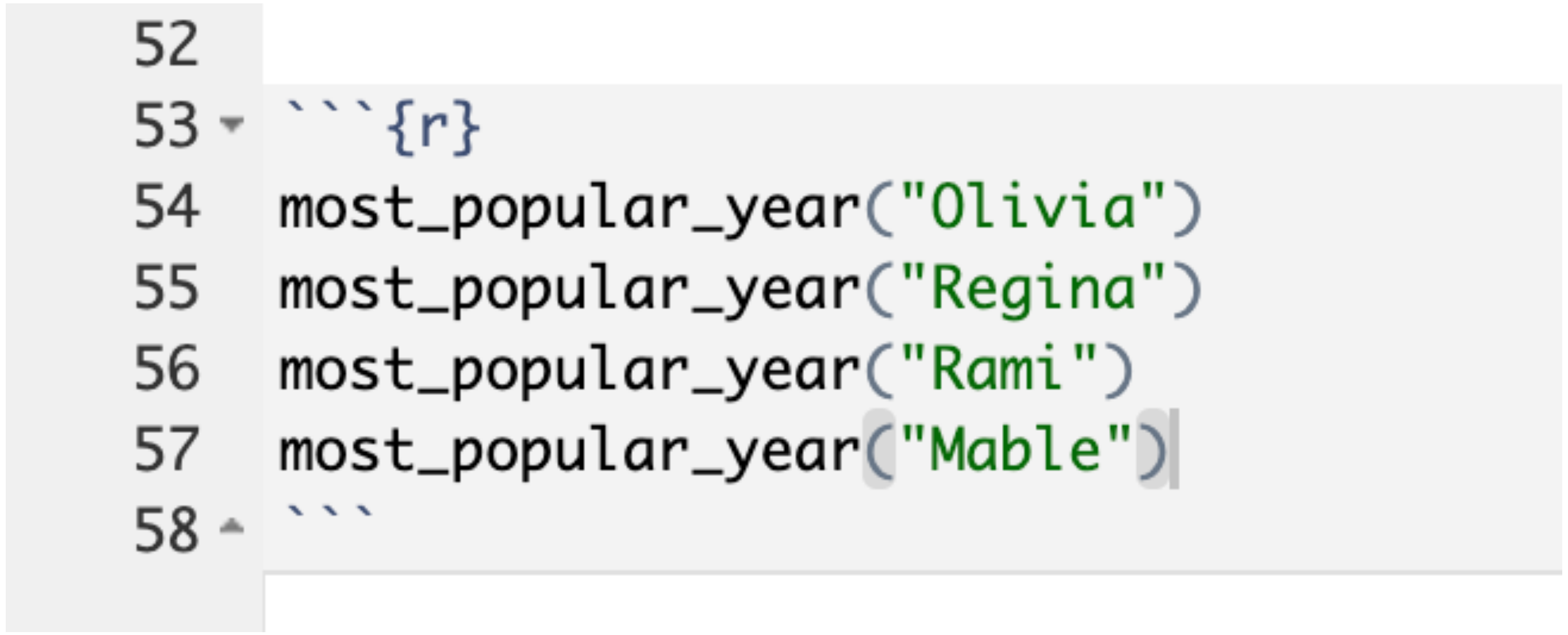
~~~
```{r}
most_popular_year <- function(name_arg) {
  babynames %>%
    filter(name == name_arg) %>%
    group_by(year) %>%
    summarize(total = sum(prop)) %>%
    arrange(desc(total)) %>%
    head(1) %>%
    select(year)
}
```
~~~

## Practice

- Use the function you wrote to find the most popular years for:
  - Olivia
  - Regina
  - Rami
  - Mable

## Practice

- Use the function you wrote to find the most popular years for:
  - Olivia - 2014
  - Regina – 1964
  - Rami - 2005
  - Mable - 1905



~~~
```{r}
most_popular_year("Olivia")
most_popular_year("Regina")
most_popular_year("Rami")
most_popular_year("Mable")
```
~~~