Data Science for Everyone – Functions Pt2

Dr. Ab Mosca (they/them)

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

Plan for Today

- Recap user defined functions
- Scope
- Default values

User-defined Functions

Defining your own functions

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

User-defined Functions

- Scope
- **Global environment**
  - The general space in which you're working
- **Global variable**
  - Variable declared in your script outside of the body of a function
  - **Global variables exist everywhere**

User-defined Functions

- Scope
- **Global environment**
  - The general space in which you're working
- **Global variable**
  - Variable declared in your script outside of the body of a function
  - **Global variables exist everywhere**
- Ex.

```
# global variable
global_var <- mtcars

top_cars <- function() {
   # function body has access to global_var
   local_var <- head(global_var)
   return(local_var)
}
# Can print global_var outside of the function too
global_var
```

User-defined Functions

- Scope
- **Local environment**
  - Body of a function
- **Local variable**
  - Variable declared within the body of a function
  - **Local variables exist only within the body** of the function in which they are declared

User-defined Functions

- Scope
- **Local environment**
  - Body of a function
- **Local variable**
  - Variable declared within the body of a function
  - **Local variables exist only within the body** of the function in which they are declared
- Ex.

```
# global variable
global_var <- mtcars

top_cars <- function() {
    # declare local_var
    local_var <- head(global_var)
    return(local_var)
}
# Cannot print local_var outside of the function
local_var # causes error
```

```
## Error in eval(expr, envir, enclos): object 'local_var' not found
```

User-defined Functions

Default Values

- When defining a function, you can set default values for the arguments
- This can make functions easier to use
- Any default value can be overwritten

User-defined Functions

Default Values

```
my_car_info <- function(mod = "civic", n = 3) {
    mpg %>%
        filter(model == mod) %>%
        select(-manufacturer, -class) %>%
        head(n)
}
my_car_info()
```

```
## # A tibble: 3 x 9
##   model displ  year   cyl trans      drv     cty   hwy fl
##   <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr>
## 1 civic   1.6  1999     4 manual(m5) f        28    33 r
## 2 civic   1.6  1999     4 auto(l4)   f        24    32 r
## 3 civic   1.6  1999     4 manual(m5) f        25    32 r
```

User-defined Functions

Overriding Default Values

```
my_car_info <- function(mod = "civic", n = 3) {
   mpg %>%
      filter(model == mod) %>%
      select(-manufacturer, -class) %>%
      head(n)
}
my_car_info()
```

```
## # A tibble: 3 x 9
##   model displ  year   cyl trans      drv     cty   hwy fl
##   <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr>
## 1 civic   1.6  1999     4 manual(m5) f        28    33 r
## 2 civic   1.6  1999     4 auto(l4)   f        24    32 r
## 3 civic   1.6  1999     4 manual(m5) f        25    32 r
```

```
my_car_info(mod = “jetta”, n = 2)
```

```
## # A tibble: 2 x 9
##   model displ  year   cyl trans      drv     cty   hwy fl
##   <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr>
## 1 jetta   1.9  1999     4 manual(m5) f        33    44 d
## 2 jetta   2    1999     4 manual(m5) f        21    29 r
```

User-defined Functions

Naming Arguments

- Optional

User-defined Functions

Naming Arguments

- Optional

```
my_car_info(mod = “jetta”, n = 2)
```

```
## # A tibble: 2 x 9
##   model displ  year   cyl trans      drv     cty   hwy fl
##   <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr>
## 1 jetta   1.9  1999     4 manual(m5) f        33    44 d
## 2 jetta   2    1999     4 manual(m5) f        21    29 r
```

```
my_car_info( “jetta”, 2)
```

```
## # A tibble: 2 x 9
##   model displ  year   cyl trans      drv     cty   hwy fl
##   <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr>
## 1 jetta   1.9  1999     4 manual(m5) f        33    44 d
## 2 jetta   2    1999     4 manual(m5) f        21    29 r
```

User-defined Functions

Naming Arguments

- Optional
- But order matters if arguments are unnamed

User-defined Functions

Naming Arguments

- Optional
- But order matters if arguments are unnamed

```
my_car_info(2, “jetta”)
```

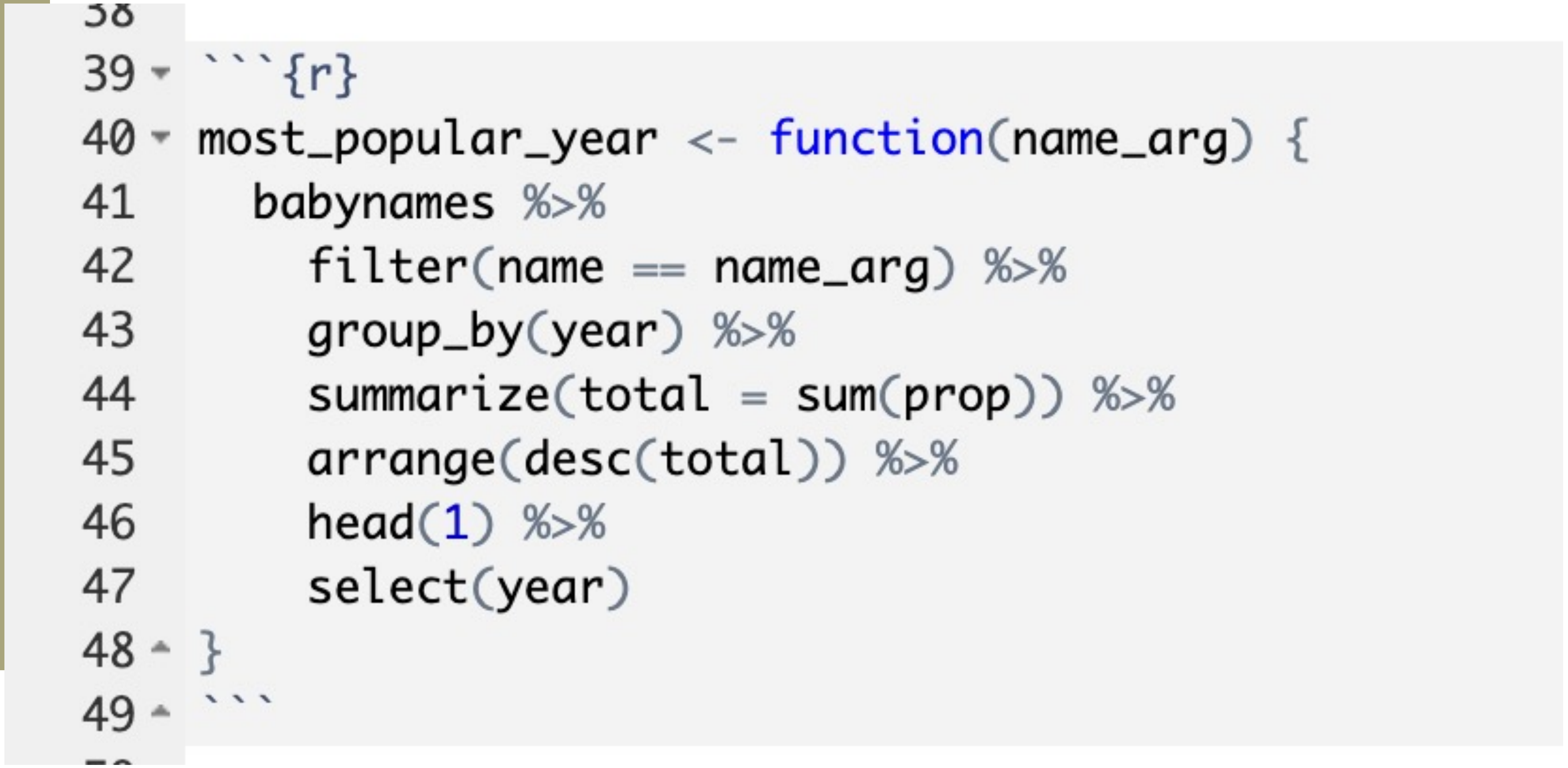
```
## # A tibble: 0 x 9
## # … with 9 variables: model <chr>, displ <dbl>, year <int>, cyl <int>,
## #   trans <chr>, drv <chr>, cty <int>, hwy <int>, fl <chr>
```

```
my_car_info(n = 2, mod = “jetta”)
```

```
## # A tibble: 2 x 9
##   model displ  year   cyl trans      drv     cty   hwy fl
##   <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr>
## 1 jetta   1.9  1999     4 manual(m5) f        33    44 d
## 2 jetta   2    1999     4 manual(m5) f        21    29 r
```

Practice

- Remember our most_popular_year function? Re-write this function to:
    - Take data as an argument, with the default being babynames
- Then, call your function for the name “Scout” for M and F babies separately



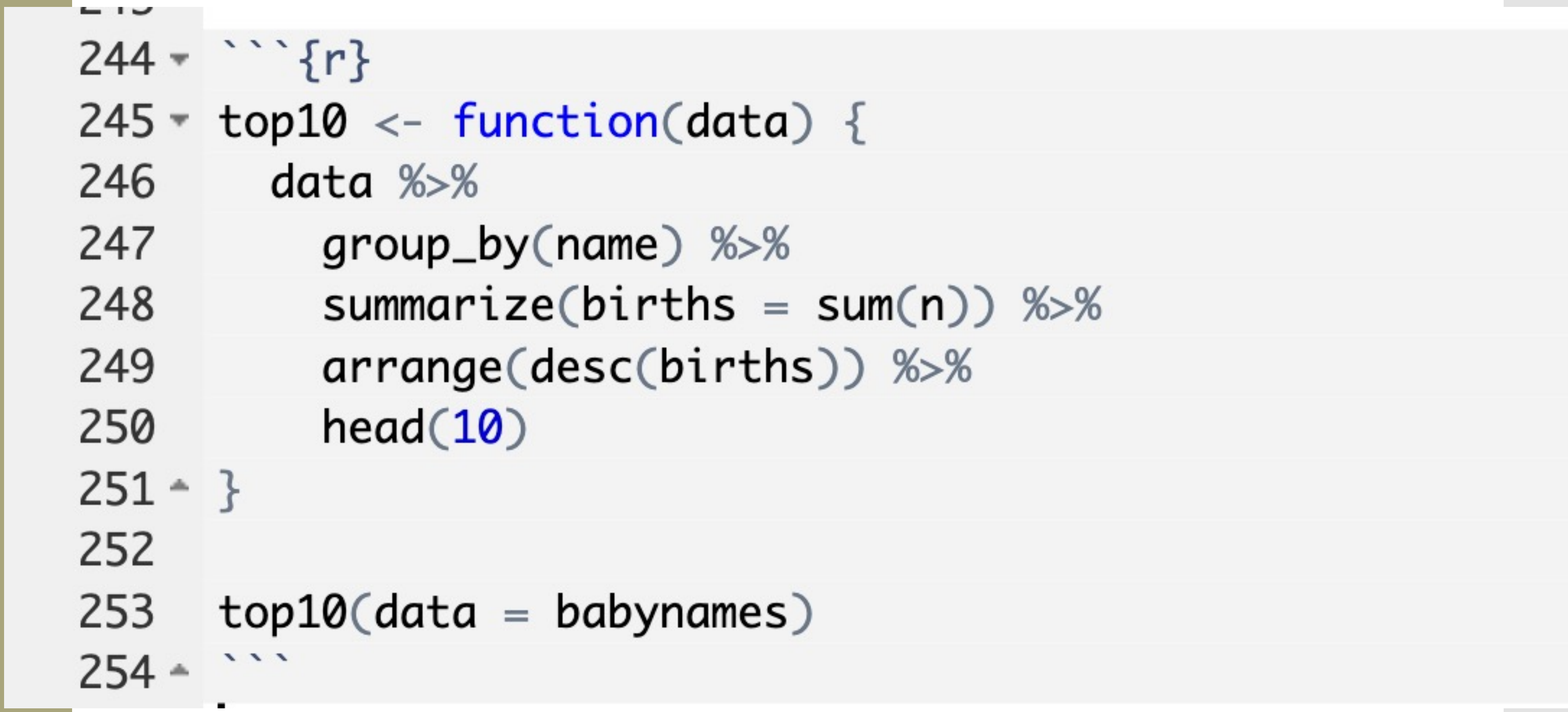
~~~
```{r}
most_popular_year <- function(name_arg) {
  babynames %>%
    filter(name == name_arg) %>%
    group_by(year) %>%
    summarize(total = sum(prop)) %>%
    arrange(desc(total)) %>%
    head(1) %>%
    select(year)
}
```
~~~

Practice

- Write a function that will computer the 10 most popular baby names for a given dataset

Practice

- Write a function that will computer the 10 most popular baby names for a given dataset

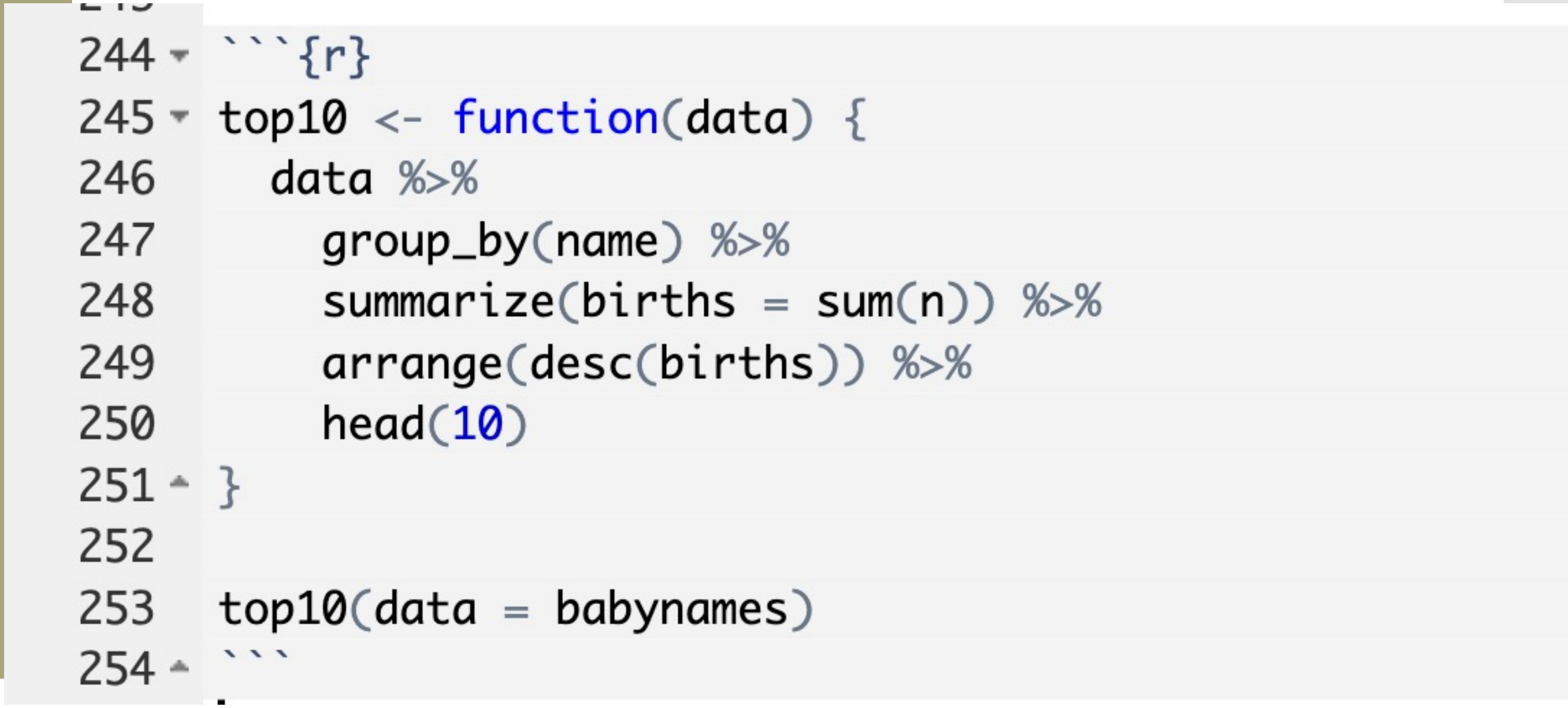


~~~
```{r}
top10 <- function(data) {
  data %>%
    group_by(name) %>%
    summarize(births = sum(n)) %>%
    arrange(desc(births)) %>%
    head(10)
}

top10(data = babynames)
```
~~~

Practice

- Write a function that will computer the 10 most popular baby names for a given dataset
- Call this function for each of the **most recent three decades** in the babynames dataset



~~~
```{r}
top10 <- function(data) {
  data %>%
    group_by(name) %>%
    summarize(births = sum(n)) %>%
    arrange(desc(births)) %>%
    head(10)
}

top10(data = babynames)
```
~~~