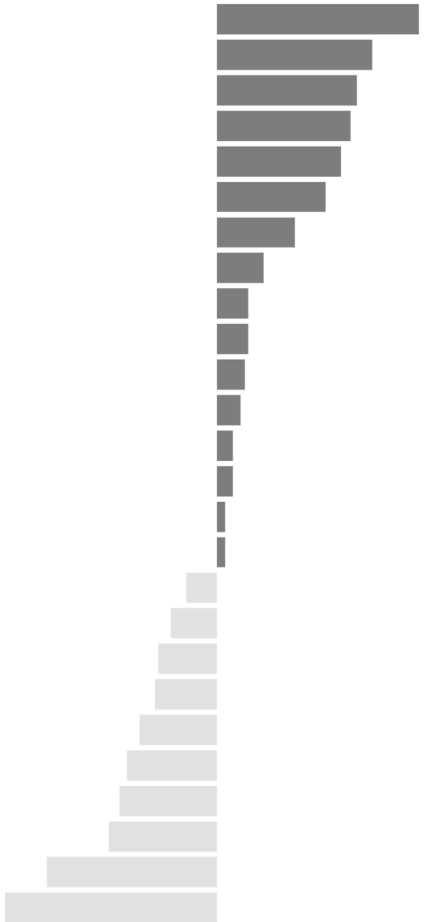


R for Lunch:

Custom functions and ggplot2

McCall Pitcher
Data Visualization Specialist
Center for Data and Visualization Sciences (CDVS)
Duke University Libraries

February 27, 2025

Center for Data and Visualization Sciences (CDVS)

http://library.duke.edu/data
askdata@duke.edu

Free workshops & consulting on:

- Data Sources
- Data Science
- Mapping and GIS
- Data Management
- Data Visualization

Overview

- Quick review of **{dplyr}** and **{ggplot2}**
- Custom functions in R
- Exercises

Quick review

Aggregating with {dplyr}

Start with this:

```
congress
```

chamber	party	n_members
House	Republican	218
House	Democrat	215
House	Other	2
Senate	Republican	53
Senate	Democrat	45
Senate	Other	2

Aggregating with {dplyr}

Start with this:

```
congress
```

chamber	party	n_members
House	Republican	218
House	Democrat	215
House	Other	2
Senate	Republican	53
Senate	Democrat	45
Senate	Other	2

Aggregate to this:

```
congress_agg
```

chamber	total
House	435
Senate	100

Aggregating with {dplyr}

Start with this:

```
congress
```

chamber	party	n_members
House	Republican	218
House	Democrat	215
House	Other	2
Senate	Republican	53
Senate	Democrat	45
Senate	Other	2

Aggregate to this:

```
congress_agg
```

chamber	total
House	435
Senate	100

How we got there:

```
congress_agg <- congress |>
  group_by(chamber) |>
  summarize(total = sum(n_members))
```

Visualizing with {ggplot2}

Basic syntax:

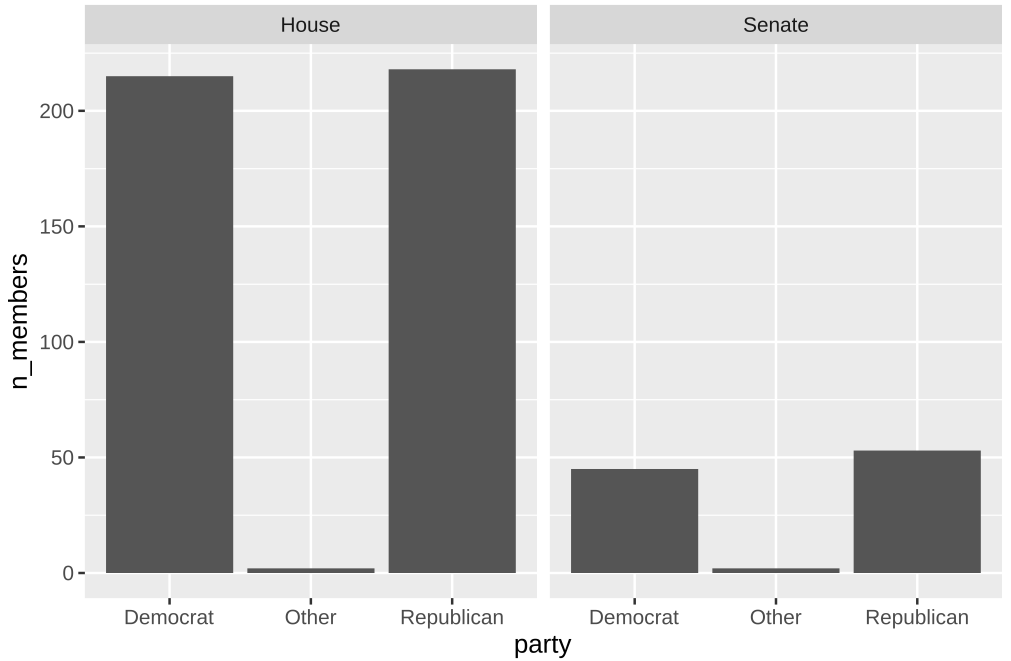
```
ggplot(your data, aes(your mapping variables)) +
  geom_ () +
  ...
```

Visualizing with {ggplot2}

Basic syntax:

```
ggplot(your data, aes(your mapping variables)) +
  geom_ () +
  ...
```

```
ggplot(congress, aes(party, n_members)) +
  geom_col() +
  facet_wrap(vars(chamber))
```

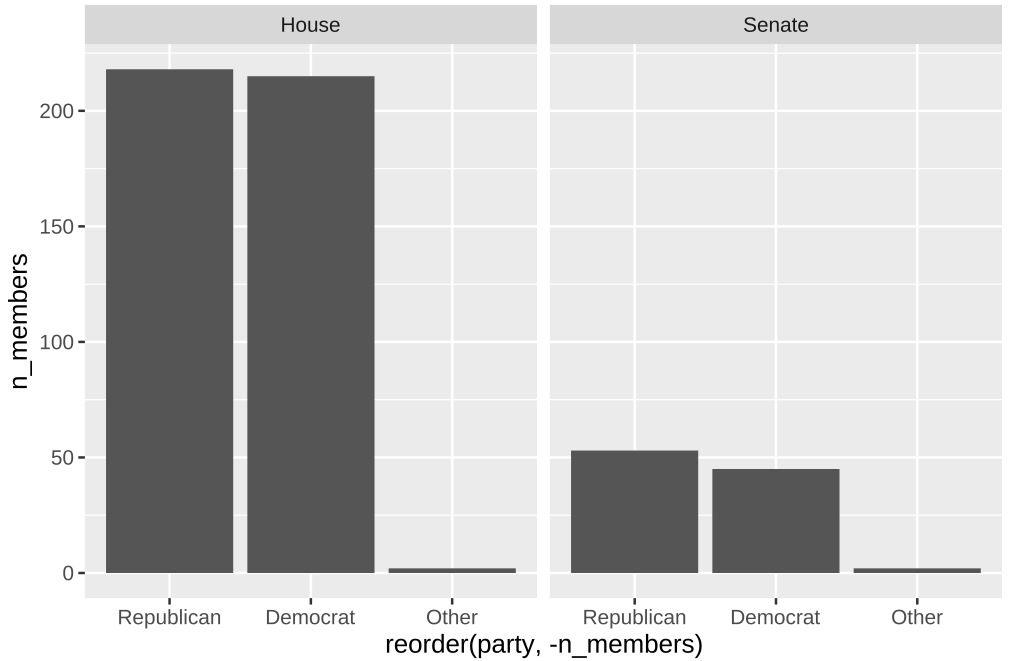


Visualizing with {ggplot2}

Basic syntax:

```
ggplot(your data, aes(your mapping variables)) +
  geom_ () +
  ...
```

```
ggplot(congress,
       aes(reorder(party,-n_members),
           n_members)) +
  geom_col() +
  facet_wrap(vars(chamber))
```

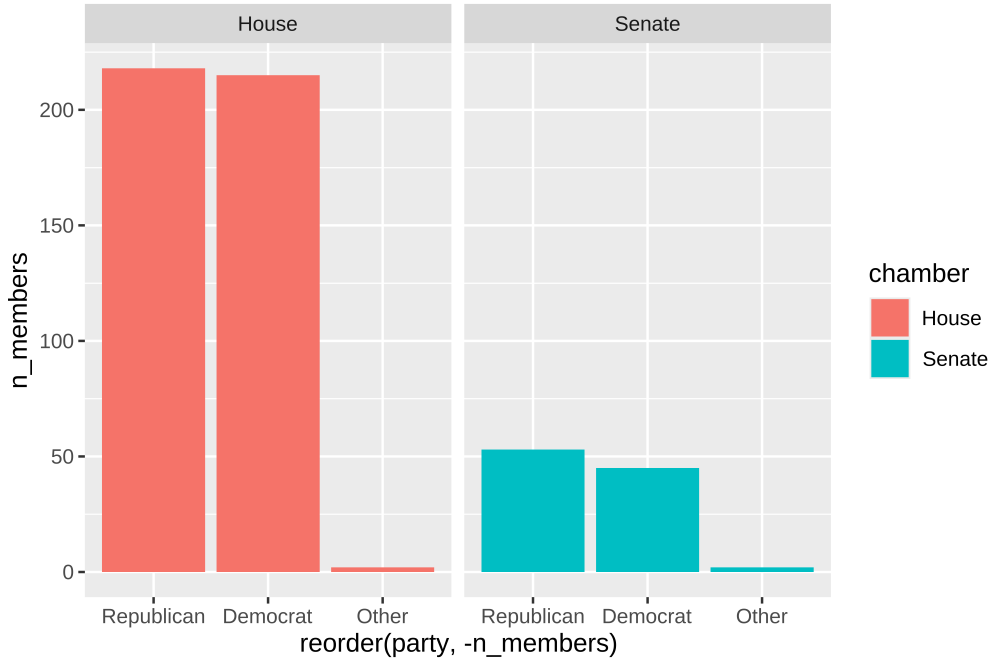


Visualizing with {ggplot2}

Basic syntax:

```
ggplot(your data, aes(your mapping variables)) +
  geom_ () +
  ...
```

```
ggplot(congress,
       aes(reorder(party,-n_members),
           n_members,
           fill = chamber)) +
  geom_col() +
  facet_wrap(vars(chamber))
```

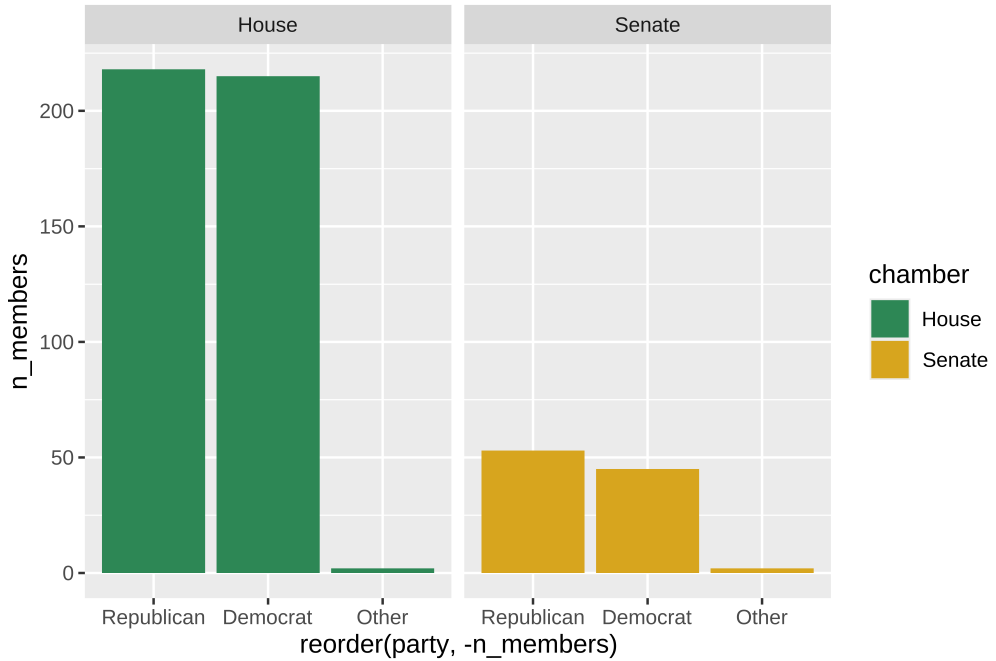


Visualizing with {ggplot2}

Basic syntax:

```
ggplot(your data, aes(your mapping variables)) +
  geom_ () +
  ...
```

```
ggplot(congress,
       aes(reorder(party,-n_members),
           n_members,
           fill = chamber)) +
  geom_col() +
  facet_wrap(vars(chamber)) +
  scale_fill_manual(values = c("seagreen",
                               "goldenrod"))
```

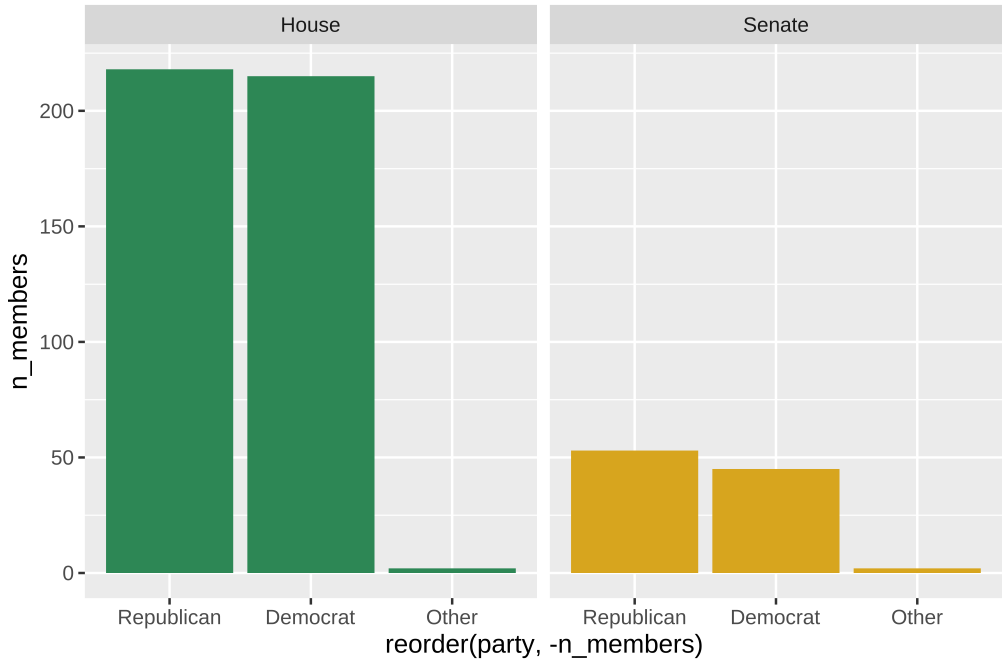


Visualizing with {ggplot2}

Basic syntax:

```
ggplot(your data, aes(your mapping variables)) +
  geom_ () +
  ...
```

```
ggplot(congress,
       aes(reorder(party,-n_members),
           n_members,
           fill = chamber)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(vars(chamber)) +
  scale_fill_manual(values = c("seagreen",
                               "goldenrod"))
```



Custom functions in R

What is a custom function?

- "Shell" set of operations that performs a task when you plug in inputs

What is a custom function?

- "Shell" set of operations that performs a task when you plug in inputs
- Purpose is to decrease repetition

What is a custom function?

- "Shell" set of operations that performs a task when you plug in inputs
- Purpose is to decrease repetition
- Can be very useful (and fun!) for visualization

What is a custom function?

- "Shell" set of operations that performs a task when you plug in inputs
- Purpose is to decrease repetition
- Can be very useful (and fun!) for visualization
- See *R for Data Science (2e)* Chapter 25 for more on Functions: https://r4ds.hadley.nz/functions

Defining a function

```
name <- function(arguments) {
  body
}
```

Defining a function

```
name <- function(arguments) {
  body
}
```

name: what you call the function

function(): tell R that this is a user-written function

arguments: inputs to your function

body: what you want the function to do (inside the curly brackets)

Good practices

- As you're starting out, avoid beginning with a function
  - Write one instance first, then convert into a function

Good practices

- As you're starting out, avoid beginning with a function
  - Write one instance first, then convert into a function

- Names should be clear and short

- Generally:
  - Names should be *verbs*
  - Arguments should be *nouns*

Simple function

- McCall doesn't know how to add 9 to things

Simple function

- McCall doesn't know how to add 9 to things
- Start with one instance/task:

```
8 + 9
```

```
## [1] 17
```

Simple function

Build the function:

```
add_9 <- function(x) {
  x + 9
}
```

Simple function

Build the function:

```
add_9 <- function(x) {
  x + 9
}
```

Call the function:

```
add_9(x = 3)
```

```
## [1] 12
```

```
add_9(x = 7)
```

```
## [1] 16
```

Simple function

Build the function:

```
add_9 <- function(x) {
  x + 9
}
```

Call the function:

```
add_9(x = 3)
```

```
## [1] 12
```

```
add_9(x = 7)
```

```
## [1] 16
```

```
# this also works
add_9(12)
```

```
## [1] 21
```

Iteration

Same function:

```
add_9 <- function(x) {
  x + 9
}
```

Call the function:

```
# vector argument, get a vector back
add_9(c(3,7,12))
```

```
## [1] 12 16 21
```

Iteration

Same function:

```
add_9 <- function(x) {
  x + 9
}
```

Call the function:

```
# vector argument, get a vector back
add_9(c(3,7,12))
```

```
## [1] 12 16 21
```

```
# apply over vector, get a list back
map(c(3,7,12), add_9)
```

```
## [[1]]
## [1] 12
##
## [[2]]
## [1] 16
##
## [[3]]
## [1] 21
```

Functions return the last called argument

This returns nothing:

```
# write the function
divide <- function(x, y) {
  object <- x / y
}

# call the function
divide(100, 20)
```

Functions return the last called argument

This returns nothing:

```
# write the function
divide <- function(x, y) {
  object <- x / y
}

# call the function
divide(100, 20)
```

This returns output:

```
# write the function
divide <- function(x, y){
  object <- x / y
  object
}

# call the function
divide(100, 20)
```

```
## [1] 5
```

Embracing

When using a variable name as a function argument with tidyverse, you must **{{ embrace }}**

This returns an error:

```
scatter <- function(var1, var2) {
  ggplot(mtcars,
         aes(var1, var2)) +
    geom_point()
}

scatter(mpg, hp)
```

```
## Error in `geom_point()`:
## ! Problem while computing aesthetics.
## ℹ Error occurred in the 1st layer.
## Caused by error:
## ! object 'hp' not found
```

Embracing

When using a variable name as a function argument with tidyverse, you must **{{ embrace }}**

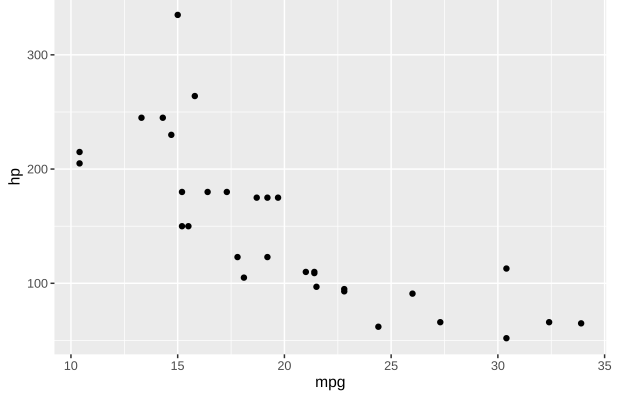
This returns an error:

```
scatter <- function(var1, var2) {
  ggplot(mtcars,
         aes(var1, var2)) +
    geom_point()
}

scatter(mpg, hp)
```

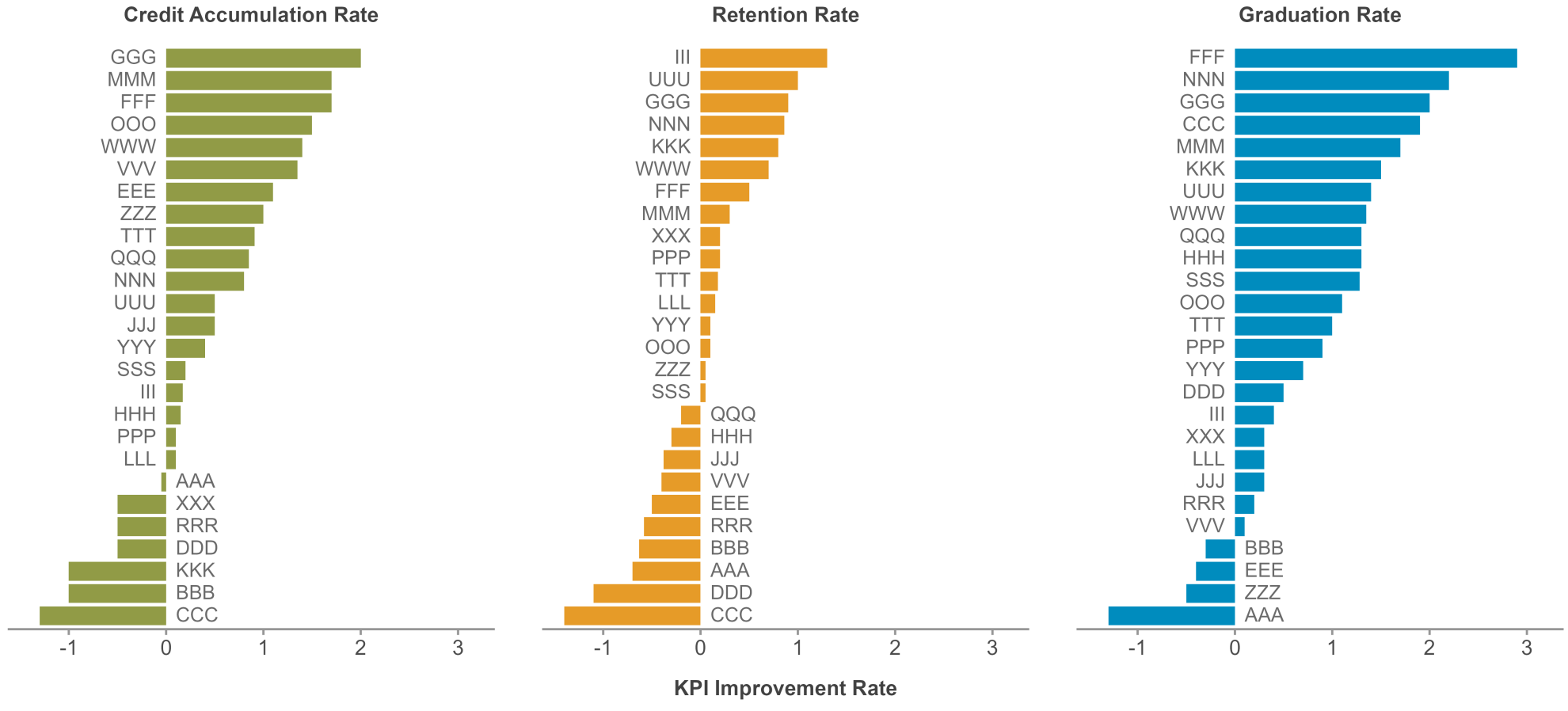
```
## Error in `geom_point()`:
## ! Problem while computing aesthetics.
## ℹ Error occurred in the 1st layer.
## Caused by error:
## ! object 'hp' not found
```

This runs properly:

```
scatter <- function(var1, var2) {
  ggplot(mtcars,
         aes({{ var1 }}, {{ var2 }})) +
    geom_point()
}

scatter(mpg, hp)
```

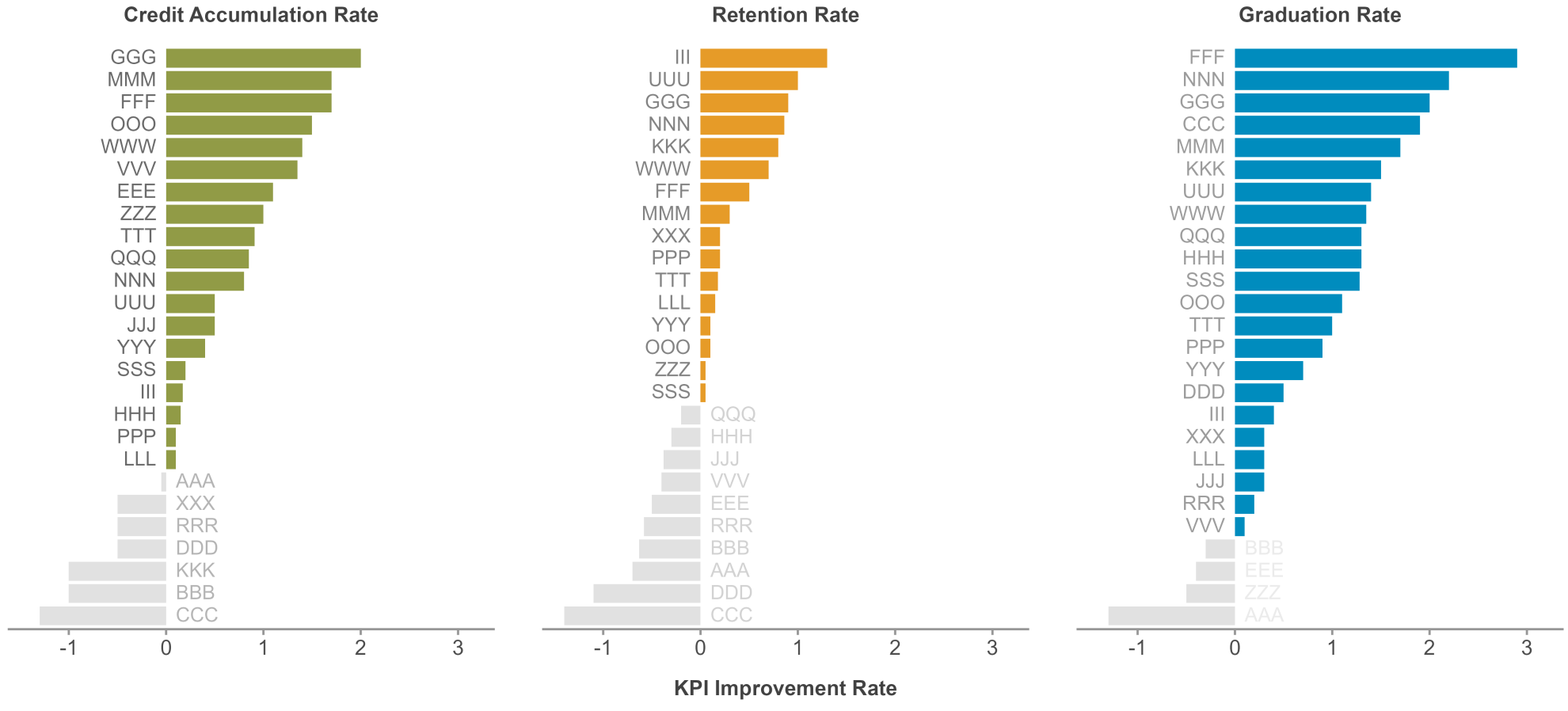


We will use functions to highlight



Note: Improvement rate is based on the slope of a regression line (percentage points per year) fitted to the annual KPI trend values.

We will use functions to highlight

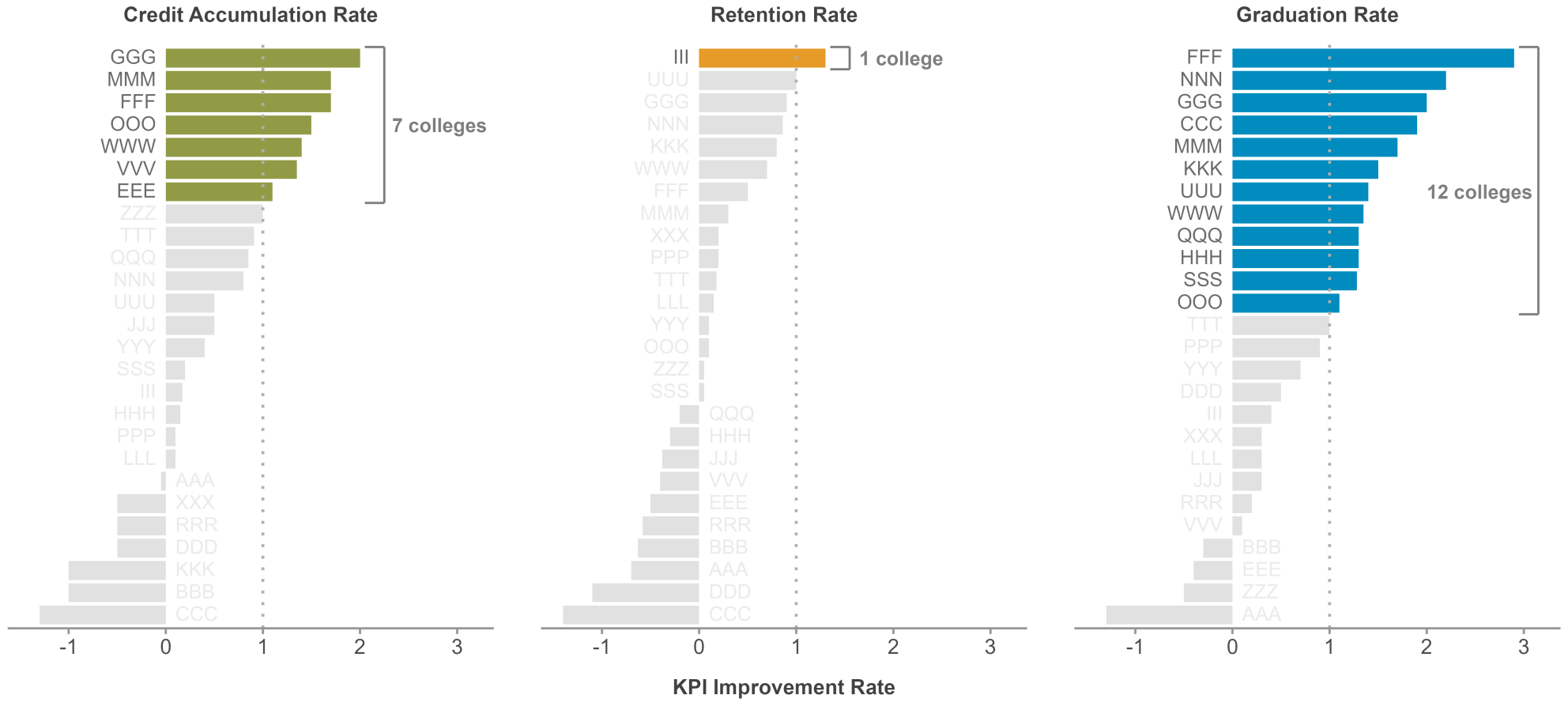


Note: Improvement rate is based on the slope of a regression line (percentage points per year) fitted to the annual KPI trend values.

We will use functions to highlight

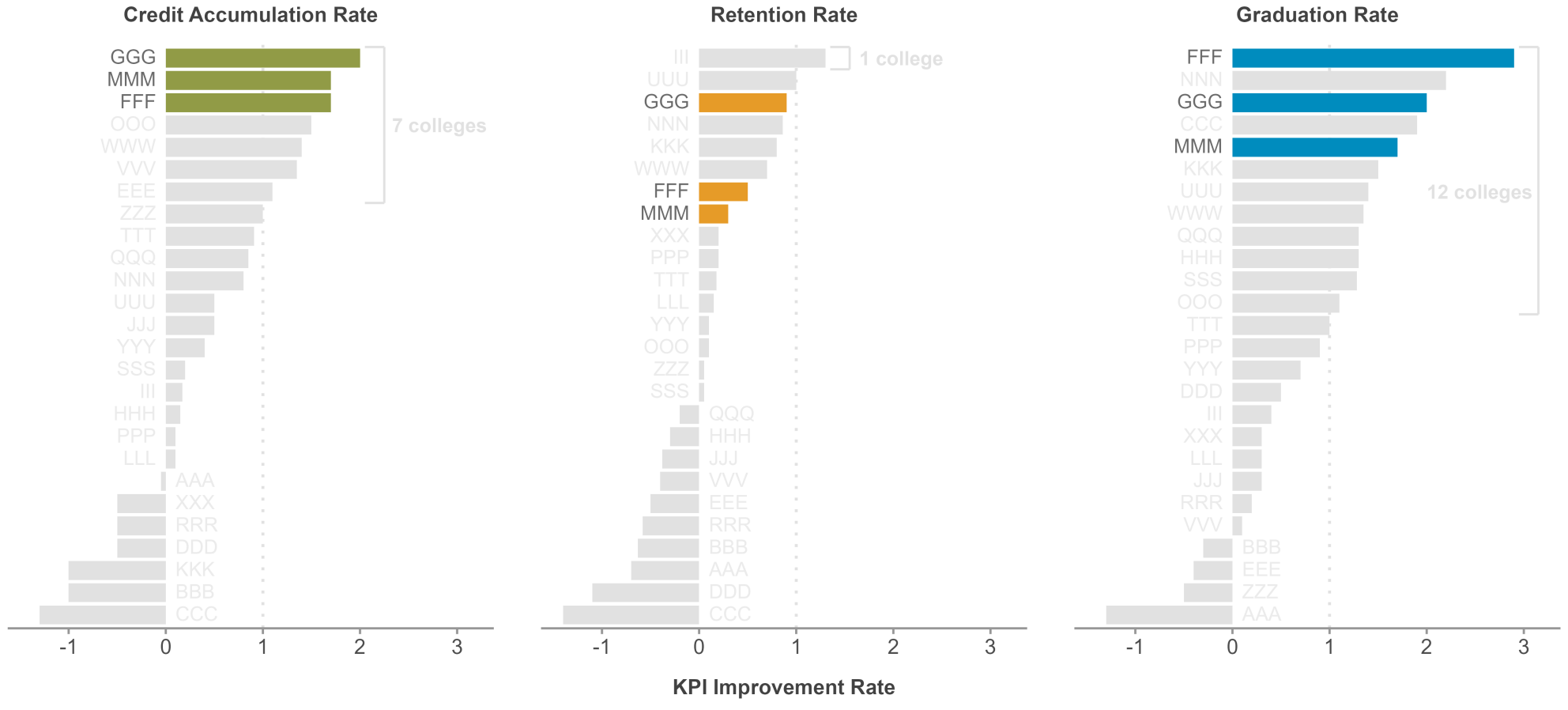
Note: Improvement rate is based on the slope of a regression line (percentage points per year) fitted to the annual KPI trend values.

We will use functions to highlight



Note: Improvement rate is based on the slope of a regression line (percentage points per year) fitted to the annual KPI trend values.

We will use functions to highlight



Note: Improvement rate is based on the slope of a regression line (percentage points per year) fitted to the annual KPI trend values.

https://bit.ly/functions_workshop