Functions

Warmup activity

Work on the activity (handout) with a neighbor, then we will discuss as a class

Warmup

```r
grouped_max <- function(df, group_var, max_var) {
  df |>
    group_by(group_var) |>
    summarize(max(max_var, na.rm=T))
}

grouped_max(penguins, species, bill_depth_mm)
```

What is this code *trying* to do?

- function: trying to create groups according to one variable, and find max of another variable within each group

- trying to use this function on penguins data

want:

species	max bill depth
Adelie	...
chinstrap	...
Gentoo	...

Warmup

```r
grouped_max <- function(df, group_var, max_var) {
  df |>
    group_by(group_var) |>
    summarize(max(max_var, na.rm=T))
}

grouped_max(penguins, species, bill_depth_mm)
```

(ideal)
(not actually happening)

```
Error in `group_by()`:
! Must group by variables found in `.data`.
✖ Column `group_var` is not found.
```

What is causing the error?

R is trying:

```
penguins |>
    group_by(group_var) |>
    ...
```

problem: group_var is not a column in penguins data!

Warmup

```r
grouped_max <- function(df, group_var, max_var) {
  df |>
    group_by(group_var) |>
    summarize(max(max_var, na.rm=T))
}

grouped_max(penguins, species, bill_depth_mm)
```

```
Error in `group_by()`:
! Must group by variables found in `.data`.
✖ Column `group_var` is not found.
```

What should we change so the code runs correctly?

Embracing

```r
grouped_max <- function(df, group_var, max_var) {
  df |>
    group_by({{ group_var }}) |>
    summarize(max({{ max_var }}, na.rm=T))
}

grouped_max(penguins, species, bill_depth_mm)
```

don't look for a column literally called "group_var"
Instead, look for a column w/ given name

```
# A tibble: 3 × 2
  species   `max(bill_depth_mm, na.rm = T)`
  <fct>                                <dbl>
1 Adelie                                21.5
2 Chinstrap                             20.8
3 Gentoo                                17.3
```

Why do we need embracing?

```
penguins |>
  filter(species == "Adelie")
```

This code contains two different types of variables:

- `penguins` is an **env-variable** (environment variable)
- `species` is a **data-variable** (it makes sense only within the context of a data frame)

Env-variables

Env-variables are objects in the R environment that we can interact with directly. For example:

```r
head(penguins)
```

```
# A tibble: 6 × 8
  species island    bill_length_mm bill_depth_mm flipper_length_mm
body_mass_g
  <fct>   <fct>              <dbl>         <dbl>             <int>
<int>
1 Adelie  Torgersen           39.1          18.7               181
3750
2 Adelie  Torgersen           39.5          17.4               186
3800
3 Adelie  Torgersen           40.3          18                 195
3250
4 Adelie  Torgersen           NA            NA                  NA
NA
```

Data-variables

Data-variables only exist in the context of a data frame:

```
# R doesn't know what 'species' is:
species
```

```
Error: object 'species' not found
```

```
# R DOES understand species in the context of penguins:
penguins$species
```

(species column from penguins data)

```
  [1] Adelie    Adelie    Adelie    Adelie    Adelie    Adelie
Adelie
  [8] Adelie    Adelie    Adelie    Adelie    Adelie    Adelie
Adelie
 [15] Adelie    Adelie    Adelie    Adelie    Adelie    Adelie
Adelie
 [22] Adelie    Adelie    Adelie    Adelie    Adelie    Adelie
Adelie
 [29] Adelie    Adelie    Adelie    Adelie    Adelie    Adelie
Adelie
 [36] Adelie    Adelie    Adelie    Adelie    Adelie    Adelie
Adelie
```

Tidy evaluation

Many `tidyverse` functions are nice and allow us to reference *data-variables*:

```
penguins |>
  filter(species == "Adelie")
```

establishing context (variables are referring to penguins data)

Here `filter` **knows** to look for a column called `species` in the `penguins` data.

Tidy evaluation

Of course, you will get an error if you try to reference a data-variable that doesn't exist! E.g. if we mis-spell the name:

```
penguins |>
  filter(speices == "Adelie")
```

```
Error in `filter()`:
ℹ In argument: `speices == "Adelie"`.
Caused by error:
! object 'speices' not found
```

Tidy evaluation

Of course, you will get an error if you try to reference a data-variable that doesn't exist!

```
penguins |>
  group_by(group_var) |>
  summarize(max(max_var, na.rm=T))
```

```
Error in `group_by()`:
! Must group by variables found in `.data`.
✖ Column `group_var` is not found.
```

The problem: `group_var` and `max_var` are not columns in the `penguins` data!

Tidy evaluation

```r
grouped_max <- function(df, group_var, max_var) {
  df |>
    group_by(group_var) |>
    summarize(max(max_var, na.rm=T))
}

grouped_max(penguins, species, bill_depth_mm)
```

```
Error in `group_by()`:
! Must group by variables found in `.data`.
✖ Column `group_var` is not found.
```

What we **want** R to run:

```r
penguins |>
  group_by(species) |>
  summarize(max(bill_depth_mm, na.rm=T))
```

```
# A tibble: 3 × 2
  species   `max(bill_depth_mm, na.rm = T)`
  <fct>                                <dbl>
1 Adelie                                21.5
```

Tidy evaluation

```r
grouped_max <- function(df, group_var, max_var) {
  df |>
    group_by(group_var) |>
    summarize(max(max_var, na.rm=T))
}

grouped_max(penguins, species, bill_depth_mm)
```

```
Error in `group_by()`:
! Must group by variables found in `.data`.
✖ Column `group_var` is not found.
```

What R is **actually** running:

```r
penguins |>
  group_by(group_var) |>
  summarize(max(max_var, na.rm=T))
```

```
Error in `group_by()`:
! Must group by variables found in `.data`.
✖ Column `group_var` is not found.
```

The solution: embracing

```r
grouped_max <- function(df, group_var, max_var) {
  df |>
    group_by({{ group_var }}) |>
    summarize(max({{ max_var }}, na.rm=T))
}

grouped_max(penguins, species, bill_depth_mm)
```

"group_var" will refer to df context

```
# A tibble: 3 × 2
  species   `max(bill_depth_mm, na.rm = T)`
  <fct>                                <dbl>
1 Adelie                                21.5
2 Chinstrap                             20.8
3 Gentoo                                17.3
```

What R is running now:

```r
penguins |>
  group_by(species) |>
  summarize(max(bill_depth_mm, na.rm=T))
```

Another example

Suppose we want to fit a simple linear regression model:

```r
penguins |>
  lm(bill_length_mm ~ bill_depth_mm, data = _) |>
  coef()
```

```
(Intercept) bill_depth_mm
 55.0673698    -0.6498356
```

Another example

```r
penguins |>
  lm(bill_length_mm ~ bill_depth_mm, data = _) |>
  coef()
```

```
(Intercept) bill_depth_mm
 55.0673698    -0.6498356
```

```r
lm_coef <- function(df, x, y) {
  df |>
    lm({{ y }} ~ {{ x }}, data = _) |>
    coef()
}

lm_coef(penguins, bill_depth_mm, bill_length_mm)
```

Do you think this code will work?

Another example

```r
penguins |>
  lm(bill_length_mm ~ bill_depth_mm, data = _) |>
  coef()
```

```
(Intercept) bill_depth_mm
 55.0673698    -0.6498356
```

```r
lm_coef <- function(df, x, y) {
  df |>
    lm({{ y }} ~ {{ x }}, data = _) |>
    coef()
}

lm_coef(penguins, bill_depth_mm, bill_length_mm)
```

```
Error: object 'bill_length_mm' not found
```

Why does this code fail?

Another example

```r
penguins |>
  lm(bill_length_mm ~ bill_depth_mm, data = _) |>
  coef()
```

```
(Intercept) bill_depth_mm
 55.0673698    -0.6498356
```

```r
lm_coef <- function(df, x, y) {
  df |>
    lm({{ y }} ~ {{ x }}, data = _) |>
    coef()
}

lm_coef(penguins, bill_depth_mm, bill_length_mm)
```

```
Error: object 'bill_length_mm' not found
```

Problem: The `lm` function does not support tidy evaluation!

(To see if a function does support tidy evaluation, look for

data-masking

tidy-selection

(key words in documentation)

Fixing the issue

```r
penguins |>
  lm(bill_length_mm ~ bill_depth_mm, data = _) |>
  coef()
```

```
(Intercept) bill_depth_mm
 55.0673698    -0.6498356
```

```r
lm_coef <- function(df, x, y) {
  df |>
    lm({{ y }} ~ {{ x }}, data = _) |>
    coef()
}

lm_coef(penguins, bill_depth_mm, bill_length_mm)
```

```
Error: object 'bill_length_mm' not found
```

If `lm` doesn't support tidy evaluation, what could we do differently?

Fixing the issue

SLR slope: $\hat{\beta}_1 = \dfrac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n}(x_i - \overline{x})^2}$

← covariance between x & y

← variance of x

```r
penguins |>
  lm(bill_length_mm ~ bill_depth_mm, data = _) |>
  coef()
```

```
(Intercept) bill_depth_mm
 55.0673698    -0.6498356
```

```r
penguins |>
  summarize(slope = cov(bill_depth_mm, bill_length_mm,
                        use="complete.obs")/
              var(bill_depth_mm, na.rm=T))
```

```
[1] -0.6498356
```

Fixing the issue

```
penguins |>
  summarize(slope = cov(bill_depth_mm, bill_length_mm,
                        use="complete.obs")/
              var(bill_depth_mm, na.rm=T))
```

How would I turn this into a function?

explanatory variable

```
slr_slope <- function(df, x, y) {
      df |>
        summarize(slope = cov({{x}}, {{y}}, ...)/var({{x}}, ...)

}
```

response

Fixing the issue

```r
slr_slope <- function(df, x, y) {
  df |>
    summarize(slope = cov({{ x }}, {{ y }}, use="complete.obs")/
                var({{ x }}, na.rm=T))
}

slr_slope(penguins, bill_depth_mm, bill_length_mm)
```

```
# A tibble: 1 × 1
   slope
   <dbl>
1 -0.650
```

```r
slr_slope(penguins, flipper_length_mm, bill_length_mm)
```

```
# A tibble: 1 × 1
  slope
  <dbl>
1 0.255
```

Class activity

https://sta279-f25.github.io/class_activities/ca_10.html

- Work with a neighbor on the class activity
- At the end of class, submit your work as an HTML file on Canvas (one per group, list all your names)

For next time, read:

- Chapter 26.3 in *R for Data Science*