



# Intro to Modular w/ R Functions

October 01, 2024

# Intro to Modular w/ R Functions

## Today's Co-organizer:

Jade Young

[jade.young@cuanschultz.edu](mailto:jade.young@cuanschultz.edu)

## Facilitators:

Yichi (Ella) Chen & Kewalin Samart

## Presenter:

Zhixin Lun

[zhixin.lun@cuanschultz.edu](mailto:zhixin.lun@cuanschultz.edu)

## Volunteer signup



## Today's sponsor:

JRavi Lab, DBMI



# Who We R!?



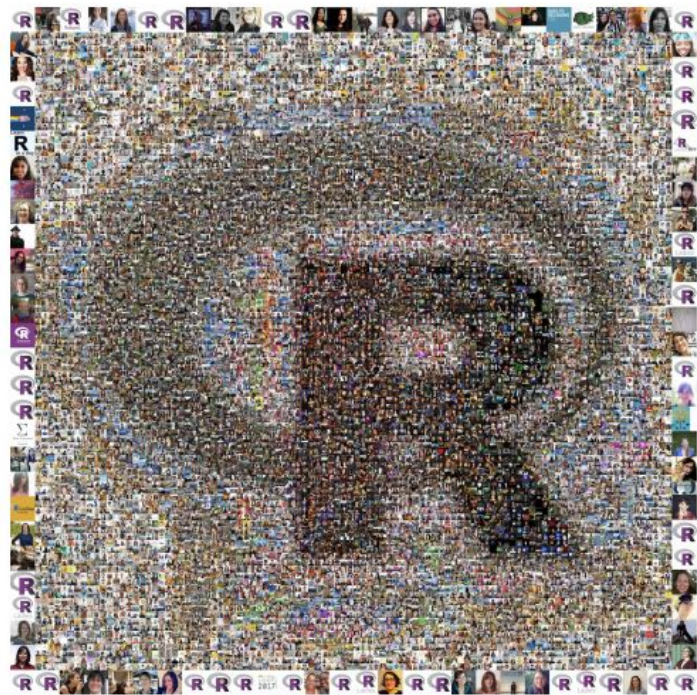
R-Ladies Kick-off Event on Jul 25, 2024




RLA Co-organizer team: Jade, Keenan, Janani, Kewalin, Stacey (left to right)




Worldwide organization that promotes **gender diversity** in the **#rstats** community via meetups and mentorship in a **friendly** and **safe** environment







Part of **R-Ladies - 240 groups**

**R-Ladies Aurora**

 Aurora, CO, USA

 69 members · Public group

 Organized by **R-Ladies Global** and **4 others**

# New Members?

1. Join us, RSVP for events on **Meetup**

<https://www.meetup.com/rladies-aurora/>

2. Join our discussion forum on **Discord**

<https://bit.ly/RLA-discord>

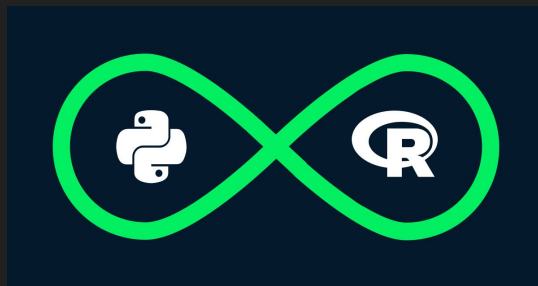




# Upcoming Workshops

## Fall' 24 R-Ladies Aurora Series:

- **Intro to R Functions**
- DataViz 2.0
- R Package Development
- Presenting to Non-Scientific Audiences
- Python in R

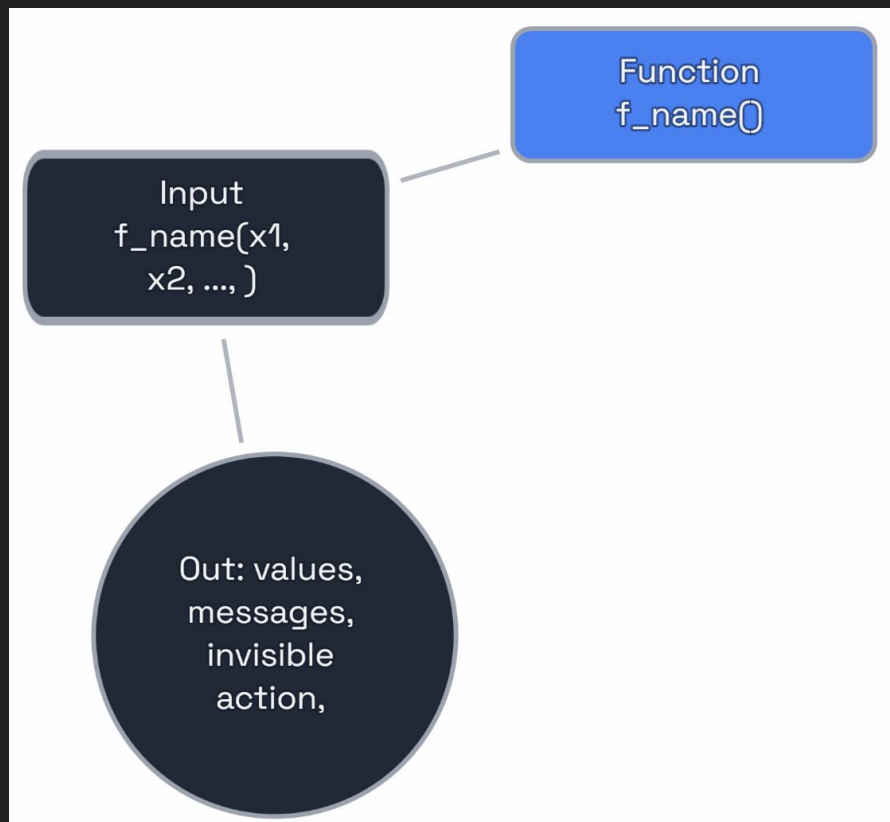


# Today! Oct 01, 2024

## TOPICS

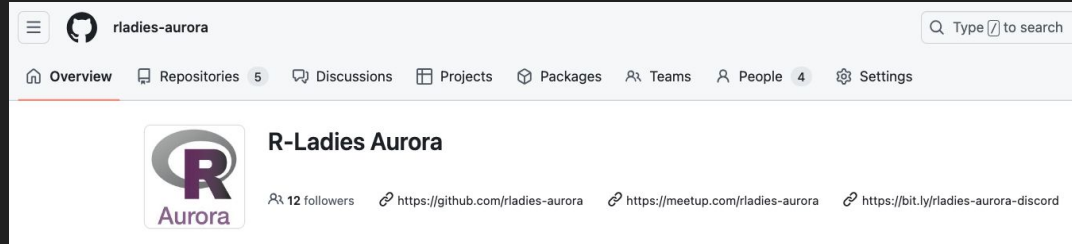
- Intro to R functions

Ping us on Discord with any Questions



# Presentation Materials

<https://github.com/rladies-aurora>



The screenshot shows the GitHub profile page for the organization 'rladies-aurora'. At the top, there is a navigation bar with the GitHub logo, the organization name 'rladies-aurora', and a search bar. Below the navigation bar, there is a horizontal menu with tabs for 'Overview', 'Repositories' (5), 'Discussions', 'Projects', 'Packages', 'Teams', 'People' (4), and 'Settings'. The 'Overview' tab is currently selected. The main content area displays the organization's profile picture, which is a stylized 'R' with 'Aurora' written below it. To the right of the profile picture, the organization's name 'R-Ladies Aurora' is displayed. Below the name, there are links for '12 followers', the GitHub repository URL 'https://github.com/rladies-aurora', the Meetup URL 'https://meetup.com/rladies-aurora', and the Discord URL 'https://bit.ly/rladies-aurora-discord'.

rladies-aurora


Search Type to search

Overview Repositories 5 Discussions Projects Packages Teams People 4 Settings

R-Ladies Aurora

12 followers <https://github.com/rladies-aurora> <https://meetup.com/rladies-aurora> <https://bit.ly/rladies-aurora-discord>

# Intro | Meet & Greet

- Pizza + Sign-up 
- Familiar w/ R and ggplot? → *Shuffle*
- Introduce yourself to your neighbor
- Who you are | Name, affiliation
- Do you have the same version of R (4.0+), RStudio & Tidyverse?
  - NO? Installation time!
    - <https://rstudio-education.github.io/hopr/starting.html>
- Need help? **Pink sticky up! All set: Blue!**





# R Functions

- You should consider writing a function whenever you've copied and pasted a block of code **more than twice**.  
-- Book by Wickham & Grolemund
- Book by Wickham & Grolemund, R4DS
  - Chapter 19 on Functions  
<https://r4ds.had.co.nz/functions.html>



# R Functions

Convert the temperature into different units:

$$C = (F - 32) \times \frac{5}{9}$$

```
C_vec[1] = (F_vec[1] - 32)*5/9
```

```
C_vec[2] = (F_vec[2] - 32)*5/9
```

```
C_vec[3] = (F_vec[3] - 32)*5/9
```

```
C_vec[4] = (F_vec[4] - 32)*5/9
```



# Function Structure

```
function_name <- function(argument1 , argument2, ...) {  
  # function body  
    Line 1  
    Line 2  
    ...  
}
```

Call a function:

```
function_name(argument1 = x, argument2 = y, ...)
```



Demo time!



# Function Structure

```
f_to_c <- function(f_value){  
  c_value = (f_value - 32)*5/9  
  return(c_value)  
}
```

Both **function** and **return** are keywords in R, you also notice the color in R Studio



# R Functions

Convert the temperature in different units:

$$C = (F - 32) \times \frac{5}{9}$$

```
C_vec = sapply(F_vec, FUN=f_to_c) ✓
```

**sapply**: apply the function on every element in the vector F\_vec.



What's *wrong* with this function?

```
current_temp_F = 100  
current_temp_C = 38
```

```
f_to_c_mtk <- function(f_value) {  
  current_temp_C = (f_value - 32)*5/9}
```

Attempting to change a global variable  
inside a function!!!

```
f_to_c_mtk(85)  
f_to_c_mtk(65)  
f_to_c_mtk(45)
```





# Local Variables and Global Variables

- Beginners are easy to mess up their difference:
  - **Global variables** are for the assignment done in the main environment or outside the chunk of function.
  - **Local variables** are for the assignment done inside the chunk of function.
- Don't attempt to change a global variable inside a function.
- **Correct way: return a value from the function and assign it to global variables.**
- Observe the Environment window in the RStudio all the time for new assignment.



## Correct way

```
current_temp_F = 100  
current_temp_C = 38
```

```
f_to_c_corr <- function(f_value) {  
  c_value = (f_value - 32)*5/9  
  return(c_value)  
}
```



```
current_temp_C = f_to_c_corr(85)  
current_temp_C = f_to_c_corr(65)  
current_temp_C = f_to_c_corr(45)
```



# From Easy to Moderate Function

```
f_to_c_human <- function(f_value) {  
  if (f_value >= 91.8 & f_value <= 100.8) {  
    c_value = (f_value - 32)*5/9  
    return(c_value)  
  }  
  else {  
    message("Too low or too high!!")  
  }  
}
```



# From Easy to Moderate Function

```
f_to_c_human_msg <- function(f_value) {  
  if (f_value >= 91.8 & f_value <= 100.8) {  
    c_value = (f_value - 32)*5/9  
    return(c_value)  
  }  
  else if (f_value < 91.8) {  
    message("Too low!!")  
  }  
  else {  
    message("Too high!!")  
  }  
}
```



# Error Control

- Beginners usually don't pay enough attention on the error control.
- A lot of programming bugs are due to inappropriate error control.
- Error control can handle the incorrect or unexpected input.
- The function `f_to_c_human_msg()` is an example with error control.
- Think about what is your expected result when encounter unexpected inputs.
  - Ignore the error and keep the function running?
  - Stop the function?



# Stop the function

```
f_to_c_human_stop <- function(f_value) {  
  if (f_value >= 91.8 & f_value <= 100.8) {  
    c_value = (f_value - 32)*5/9  
    return(c_value)  
  }  
  else if (f_value < 91.8) {  
    stop("Too low!!") # stop the function  
  }  
  else {  
    stop("Too high!!") # stop the function  
  }  
}
```



## Compare the difference

```
C_vec = sapply(F_vec, FUN = f_to_c_human_msg)  
C_vec = sapply(F_vec, FUN = f_to_c_human_stop)
```

message: Keep the function running and store NULL values for invalid input in C\_vec.

stop: stop the function and reset the vector C\_vec.





# You Don't Need to Reinvent the Wheel!

- Try to explore if there are some existing functions or packages in R.
- Create your own functions ONLY if you can't find any available functions.
- Don't limit the function output as values, numbers, messages. It can be figures!
- Let's try to run a function using **ggplot2** functions.



# Good Coding Practices

- Function names: verbs, never start with numbers. Consider using underscores to separate words, e.g., `impute_missing()`, `calc_avg()`.
- Argument names: nouns
- Output: explicitly return values or messages using `return()`, `message()`



# Useful Materials

- R for Data Science <https://r4ds.had.co.nz/>
- Advanced R <https://adv-r.hadley.nz/>
- Best Coding Practices for R  
<https://bookdown.org/content/d1e53ac9-28ce-472f-bc2c-f499f18264a3/>
- Intro to functions in R, R-Ladies East Lansing  
<https://github.com/rladies-eastlansing/2021-rfunctions>



# Thank you!

R-Ladies Aurora

