What is your major mal-function?

PROGRAMMING WITH DPLYR

Dr. Chester Ismay

Educator, Data Scientist, and R/Python Consultant

Recall Uruguay's CPI

```
imf_data %>%
  select(iso, country, year,
         consumer_price_index) %>%
  filter(country == "Uruguay",
         year > 2010)
```

```
# A tibble: 9 × 4
  iso   country  year consumer_price_index
  <chr> <chr>   <int>                <dbl>
1 URY   Uruguay  2011                 105.
2 URY   Uruguay  2012                 114.
3 URY   Uruguay  2013                 123.
4 URY   Uruguay  2014                 134.
5 URY   Uruguay  2015                 146.
6 URY   Uruguay  2016                 160.
7 URY   Uruguay  2017                 170.
8 URY   Uruguay  2018                 183.
9 URY   Uruguay  2019                 197.
```

Another country's CPI

```r
imf_data %>%
  select(iso, country, year,
         consumer_price_index) %>%
  filter(country == "Belize",
         year > 2010)
```

```
# A tibble: 9 × 4
  iso   country  year consumer_price_index
  <chr> <chr>   <int>                <dbl>
1 BLZ   Belize   2011                 130.
2 BLZ   Belize   2012                 132.
3 BLZ   Belize   2013                 133.
4 BLZ   Belize   2014                 134.
5 BLZ   Belize   2015                 133.
6 BLZ   Belize   2016                 134.
7 BLZ   Belize   2017                 136.
8 BLZ   Belize   2018                 136.
9 BLZ   Belize   2019                 136.
```

Creating a function instead

```r
cpi_by_country <- function(country_name) {
  imf_data %>%
    select(iso, country, year,
           consumer_price_index) %>%
    filter(country == country_name,
           year > 2010)
}
```

Use the function

```
cpi_by_country <- function(country_name) {
  imf_data %>%
    select(iso, country, year,
           consumer_price_index) %>%
    filter(country == country_name,
           year > 2010)
}
```

```
cpi_by_country(country_name = "Samoa")
```

```
# A tibble: 9 × 4
  iso   country  year consumer_price_index
  <chr> <chr>   <int>                <dbl>
1 WSM   Samoa    2011                 102.
2 WSM   Samoa    2012                 109.
3 WSM   Samoa    2013                 108.
4 WSM   Samoa    2014                 107.
5 WSM   Samoa    2015                 109.
6 WSM   Samoa    2016                 109.
7 WSM   Samoa    2017                 111.
8 WSM   Samoa    2018                 115.
9 WSM   Samoa    2019                 117.
```

Joining IMF and World Bank data

```r
joined <- imf_data %>%
    inner_join(world_bank_data,
               by = c("iso", "year")) %>%
  relocate(continent, region, .after = year)
```

Results of the join

```
joined
```

```
# A tibble: 299 × 22
   iso   country.x  year continent region                    gdp_in_billions...
   <chr> <chr>     <dbl> <fct>     <fct>                                <dbl>
 1 ALB   Albania    2011 Europe    Southern Europe                      12.9
 2 ALB   Albania    2012 Europe    Southern Europe                      12.3
 3 AGO   Angola     2014 Africa    Middle Africa                       146.
# ... with 296 more rows, and 16 more variables:
#   usd_conversion_rate <dbl>, total_investment_as_perc_gdp <dbl>,
#   consumer_price_index <dbl>, imports_perc_change <dbl>,
#   exports_perc_change <dbl>, population_in_millions <dbl>,
#   gov_revenue_as_perc_gdp <dbl>, gov_net_debt_as_perc_gdp <dbl>,
#   country.y <chr>, infant_mortality_rate <dbl>, fertility_rate <dbl>,
#   perc_electric_access <dbl>, perc_college_complete <dbl>, ...
```

Mean Government Revenue as GDP % by Continent

```r
joined %>%
  group_by(continent) %>%
  summarize(mean_gov_revenue = mean(
    gov_revenue_as_perc_gdp)
  )
```

```
# A tibble: 5 × 2
  continent mean_gov_revenue
  <fct>                <dbl>
1 Africa                18.2
2 Americas              25.6
3 Asia                  26.7
4 Europe                41.5
5 Oceania               35.6
```

A function for grouping?

```r
grouped_mean_gov_revenue <- function(group_col) {
  joined %>%
      group_by(group_col) %>%
      summarize(mean_gov_revenue = mean(gov_revenue_as_perc_gdp))
}
grouped_mean_gov_revenue(group_col = year)
```

```
Error: Must group by variables found in `.data`.
* Column `group_col` is not found.
```

The curly-curly {{ }} operator

```r
library(rlang)
grouped_mean_gov_revenue <- function(group_col) {
  joined %>%
    group_by({{ group_col }}) %>%
    summarize(mean_gov_revenue = mean(gov_revenue_as_perc_gdp))
}
grouped_mean_gov_revenue(group_col = year)
```

```
# A tibble: 17 × 2
    year mean_gov_revenue
   <dbl>            <dbl>
 1  2000             39.4
 2  2001             30.8
...
16  2015             34.7
17  2016             32.3
```

Let's practice!

PROGRAMMING WITH DPLYR

Bang-bang!!

PROGRAMMING WITH DPLYR

Dr. Chester Ismay

Educator, Data Scientist, and R/Python Consultant

Unpacking curly-curly

```r
grouped_mean_gov_revenue <- function(group_col) {
  joined %>%
    group_by({{ group_col }}) %>%
    summarize(mean_gov_revenue = mean(gov_revenue_as_perc_gdp))
}
grouped_mean_gov_revenue(group_col = continent)
```

`{{ }}`

- Forces function argument
- Defuses function argument

Bang-bang-enquo forces and defuses

`!!enquo()` is the same as `{{ }}`

```
grouped_mean_gov_revenue <- function(group_col) {
  joined %>%
    group_by(!!enquo(group_col)) %>%
    summarize(
      mean_gov_revenue = mean(
        gov_revenue_as_perc_gdp
      )
   )
}
grouped_mean_gov_revenue(group_col = continent)
```

```
# A tibble: 17 × 2
    year mean_gov_revenue
   <dbl>            <dbl>
 1  2000             39.4
 2  2001             30.8
...
16  2015             34.7
17  2016             32.3
```

Multiple function arguments

```
grouped_mean_gov_revenue <- function(group_col) {
  joined %>%
    group_by(!!enquo(group_col)) %>%
    summarize(mean_gov_revenue = mean(gov_revenue_as_perc_gdp))
}
```

```
grouped_mean_for_column <- function(group_col, col_to_mean) {
  joined %>%
    group_by(!!enquo(group_col)) %>%
    summarize(mean(!!enquo(col_to_mean)))
}
```

Calling grouped_mean_for_column()

```
grouped_mean_for_column(group_col = continent,
                        col_to_mean = perc_cvd_crd_70)
```

```
# A tibble: 5 × 2
  continent `mean(perc_cvd_crd_70)`
  <fct>                       <dbl>
1 Africa                      23.8 
2 Americas                    14.7 
3 Asia                        19.8 
4 Europe                      16.3 
5 Oceania                      9.91
```

Let's practice!

PROGRAMMING WITH DPLYR

Rlang-ing in your rocking chair

PROGRAMMING WITH DPLYR

Dr. Chester Ismay

Educator, Data Scientist, and R/Python Consultant

Strange summarize column name

```
grouped_mean_by_column <- function(group_col, col_to_mean) {
  joined %>%
    group_by(!!enquo(group_col)) %>%
    summarize(mean(!!enquo(col_to_mean)))
}
grouped_mean_by_column(group_col = year, col_to_mean = perc_cvd_crd_70)
```

```
# A tibble: 17 × 2
    year `mean(perc_cvd_crd_70)`
   <dbl>                   <dbl>
 1  2000                    15.8
 2  2001                    23.3
...
17  2016                    15.8
```

What we'd like to be

```
# A tibble: 17 × 2
    year mean_of_perc_cvd_crd_70
   <dbl>                   <dbl>
 1  2000                    15.8
 2  2001                    23.3
 3  2002                    15  
 4  2003                    14.8
 5  2004                    21.1
...
15  2014                    16.7
16  2015                    16.0
17  2016                    15.8
```

as_name() and the walrus operator :=

`as_name()`

- Converts unquoted, defused column name to be a string

`:=`

- Allows for a variable in R to be on the left-hand side in `mutate()`

Wrapping up the function

```r
grouped_mean_by_column <- function(.data, group_col, col_to_mean) {
  name_of_col_to_mean <- as_name(enquo(col_to_mean))
  new_col_name <- paste0("mean_of_", name_of_col_to_mean)
  .data %>%
    group_by(!!enquo(group_col)) %>%
    summarize(!!new_col_name := mean(!!enquo(col_to_mean)))
}
```

Applying the wrapped up function

```
grouped_mean_by_column(.data = joined,
                       group_col = continent,
                       col_to_mean = perc_rural_pop)
```

```
# A tibble: 5 × 2
  continent mean_of_perc_rural_pop
  <fct>                      <dbl>
1 Africa                      65.1
2 Americas                    28.1
3 Asia                        31.9
4 Europe                      30.1
5 Oceania                     14.1
```

Cleaning things up a bit

```r
grouped_mean_by_column <- function(.data, group_col, col_to_mean) {
  name_of_col_to_mean <- as_name(enquo(col_to_mean))
  new_col_name <- paste0("mean_of_", name_of_col_to_mean)
  .data %>%
    group_by({{ group_col }}) %>%
    summarize(!!new_col_name := mean({{ col_to_mean }}))
}
joined %>%
  grouped_mean_by_column(group_col = continent,
                         col_to_mean = perc_rural_pop)
```

Let's practice!

PROGRAMMING WITH DPLYR

A great ggplot twist

PROGRAMMING WITH DPLYR

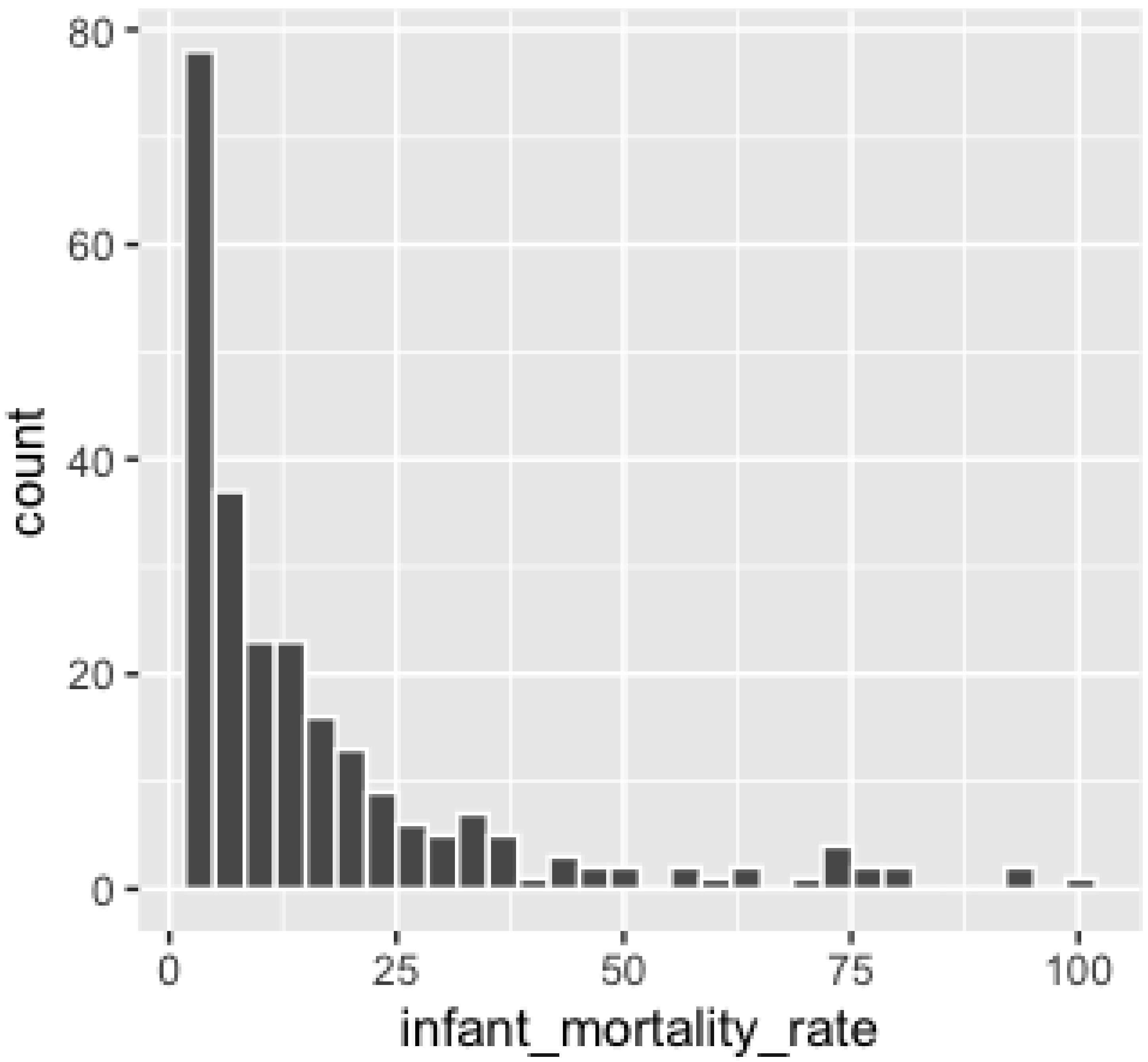
Dr. Chester Ismay

Educator, Data Scientist, and R/Python Consultant

datacamp

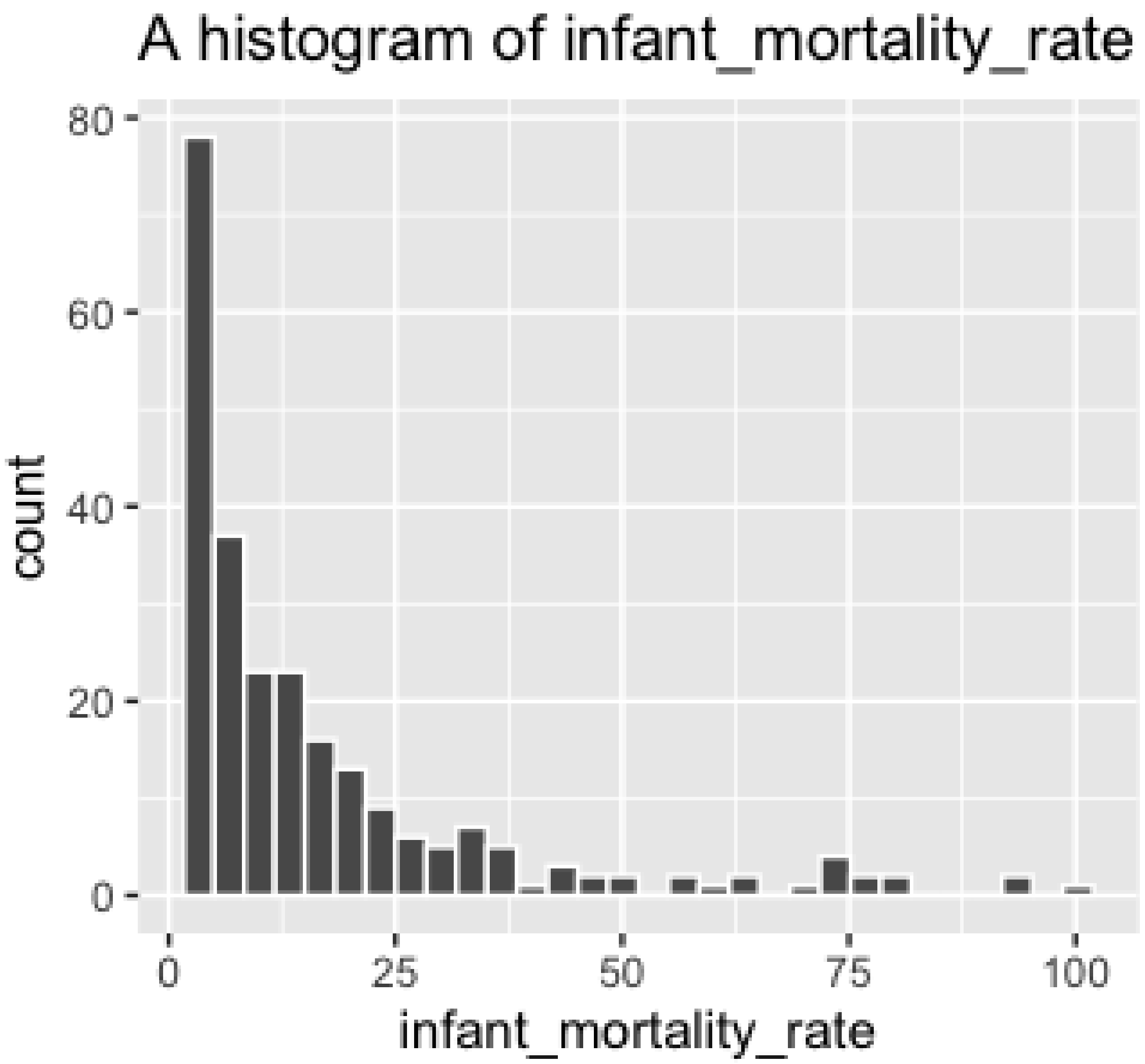
ggplot2 basics

```
library(ggplot2)
ggplot(
    world_bank_data,
    aes(x = infant_mortality_rate)
) +
  geom_histogram(color = "white")
```



Adding a title

```
ggplot(
    world_bank_data,
    aes(x = infant_mortality_rate)
) +
  geom_histogram(color = "white") +
  ggtitle("A histogram of infant_mortality_rate")
```



A histogram of infant_mortality_rate

Wrapping into a function

```r
# Previous plot code
ggplot(world_bank_data, aes(x = infant_mortality_rate)) +
  geom_histogram(color = "white") +
  ggtitle("A histogram of infant_mortality_rate")
```

```r
# Define a function
my_histogram <- function(
    df, x_var) {
  ggplot(df, aes(x = x_var)) +
    geom_histogram(color = "white") +
    ggtitle(paste("A histogram of", x_var))
}
```

Working on our function

```r
# First attempt at a function
my_histogram <- function(df, x_var) {
  ggplot(df, aes(x = x_var)) +
    geom_histogram(color = "white") +
    ggtitle(paste("A histogram of", x_var))
}
# Call the function
my_histogram(df = world_bank_data,
             x_var = infant_mortality_rate)
```

```
Error in paste("A histogram of", x_var) :
 object 'infant_mortality_rate' not found
```

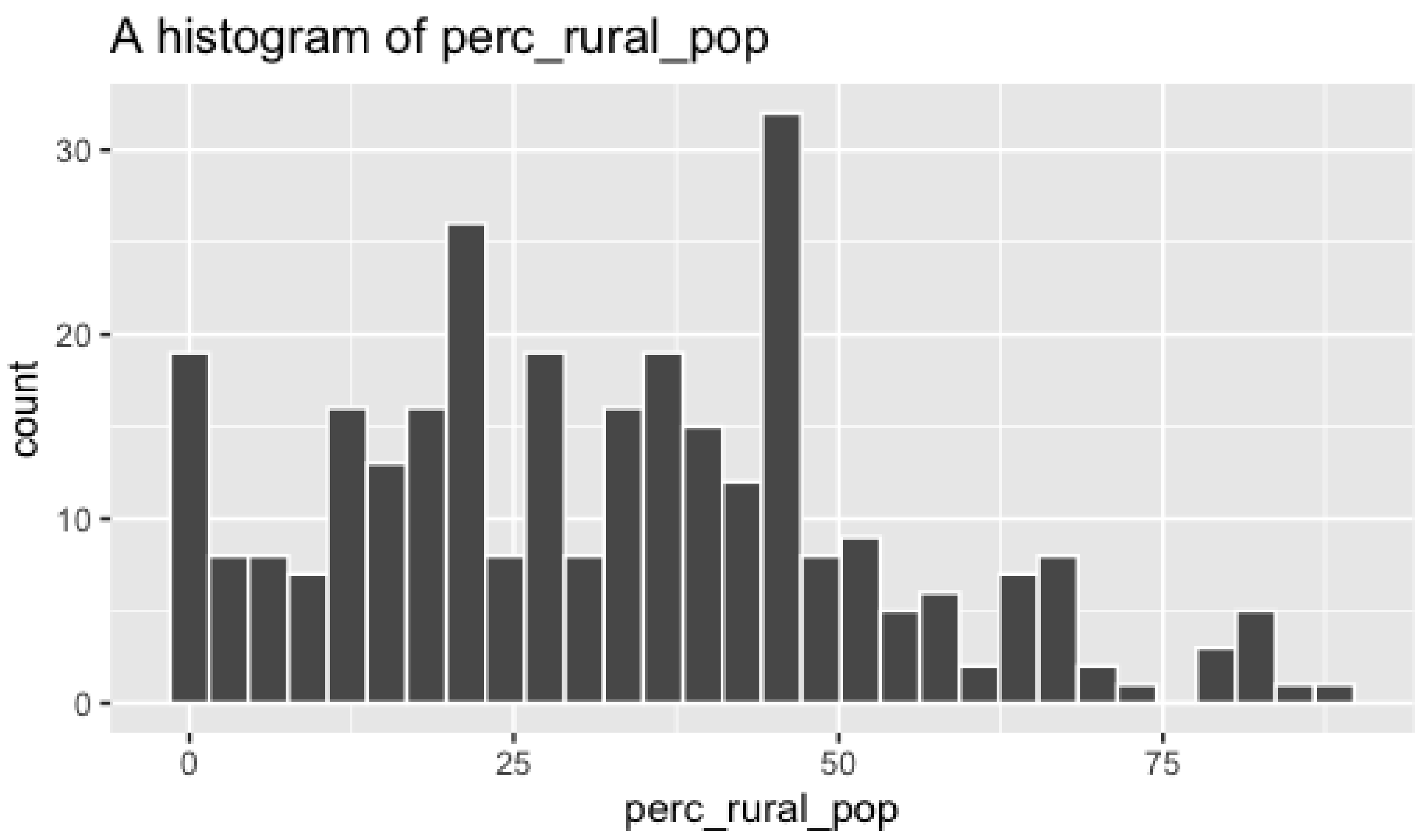
Adding in rlang

```r
# First attempt at a function
my_histogram <- function(df, x_var) {
  ggplot(df, aes(x = x_var)) +
    geom_histogram(color = "white") +
    ggtitle(paste("A histogram of", x_var))
}
```

```r
# Making needed tweaks to stop error
my_histogram <- function(df, x_var) {
  ggplot(df, aes(x = {{ x_var }})) +
    geom_histogram(color = "white") +
    ggtitle(paste("A histogram of",
                  as_label(enquo(x_var)))
}
```

Using our function

```r
# Defined function with rlang operators
my_histogram <- function(df, x_var) {
  ggplot(df, aes(x = {{ x_var }})) +
    geom_histogram(color = "white") +
    ggtitle(paste(
      "A histogram of",
      as_label(enquo(x_var))
    ))
}
# Call the function with the pipe!
world_bank_data %>%
  my_histogram(x_var = perc_rural_pop)
```



Let's practice!

PROGRAMMING WITH DPLYR

Congratulations!

PROGRAMMING WITH DPLYR

Dr. Chester Ismay

Educator, Data Scientist, and R/Python Consultant

Chapter 1 - select() and its helper functions

- Pipelines with the main `dplyr` verbs of `select()` , `filter()` , `summarize()` , `group_by()` , and `mutate()`
- `starts_with()` , `ends_with()` , and `contains()` helper functions
- `matches()` to look for regular expression patterns

Chapter 2 - Column transformations

- Move columns around with `select()` , `everything()` , and `relocate()`
- Use `across()` to do the same transformation on many columns
- Choose rows that match certain column criteria with `if_any()` and `if_all()`
- Calculate across each row with `rowwise()` and `c_across()`

Chapter 3 - Joins and set theory

- `dplyr` joins
  - Review of `inner_join()` , `left_join()` , and `anti_join()`
- Set theory clauses
  - `intersect()`
  - `union()` and `union_all()`
  - `setequal()` and `setdiff()`

Chapter 4 - Getting your feet wet with rlang

- Writing functions using curly-curly `{{ }}`
- Using the forcing "bang-bang" operator `!!` and the defusing operator `enquo()`
- Creating variable names using the walrus operator `:=`, `as_name()`, and `!!`
- Wrapping `ggplot2` plotting code in a function with `as_label()`

Other relevant courses/tracks

Courses

- Visualization Best Practices in R
- Introduction to Data Visualization with ggplot2
- Intermediate Data Visualization with ggplot2

Tracks

- Tidyverse Fundamentals with R
- Intermediate Tidyverse Toolbox
- R Programmer

Woohoo for you!

PROGRAMMING WITH DPLYR