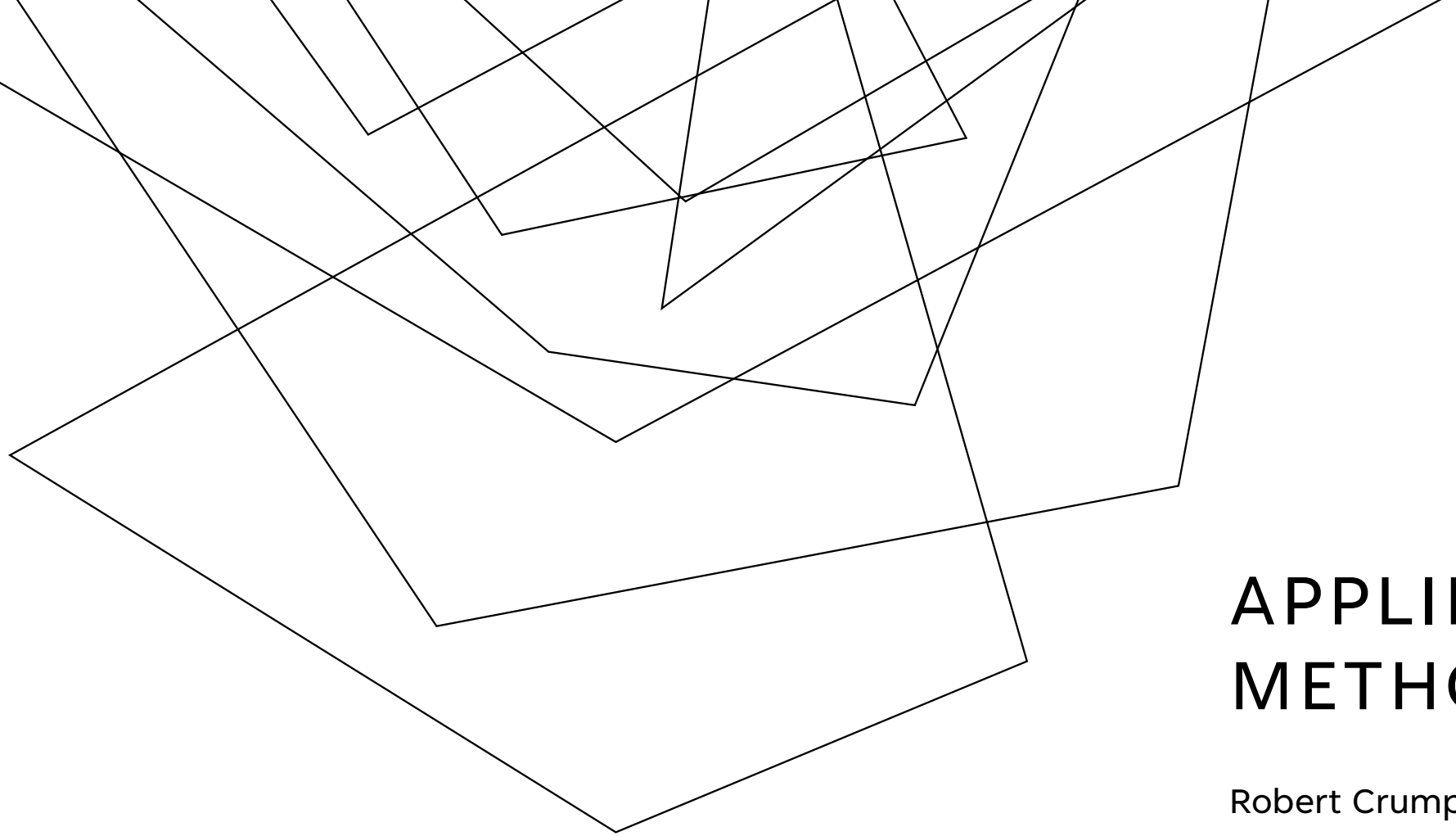


APPLIED RESEARCH METHODS

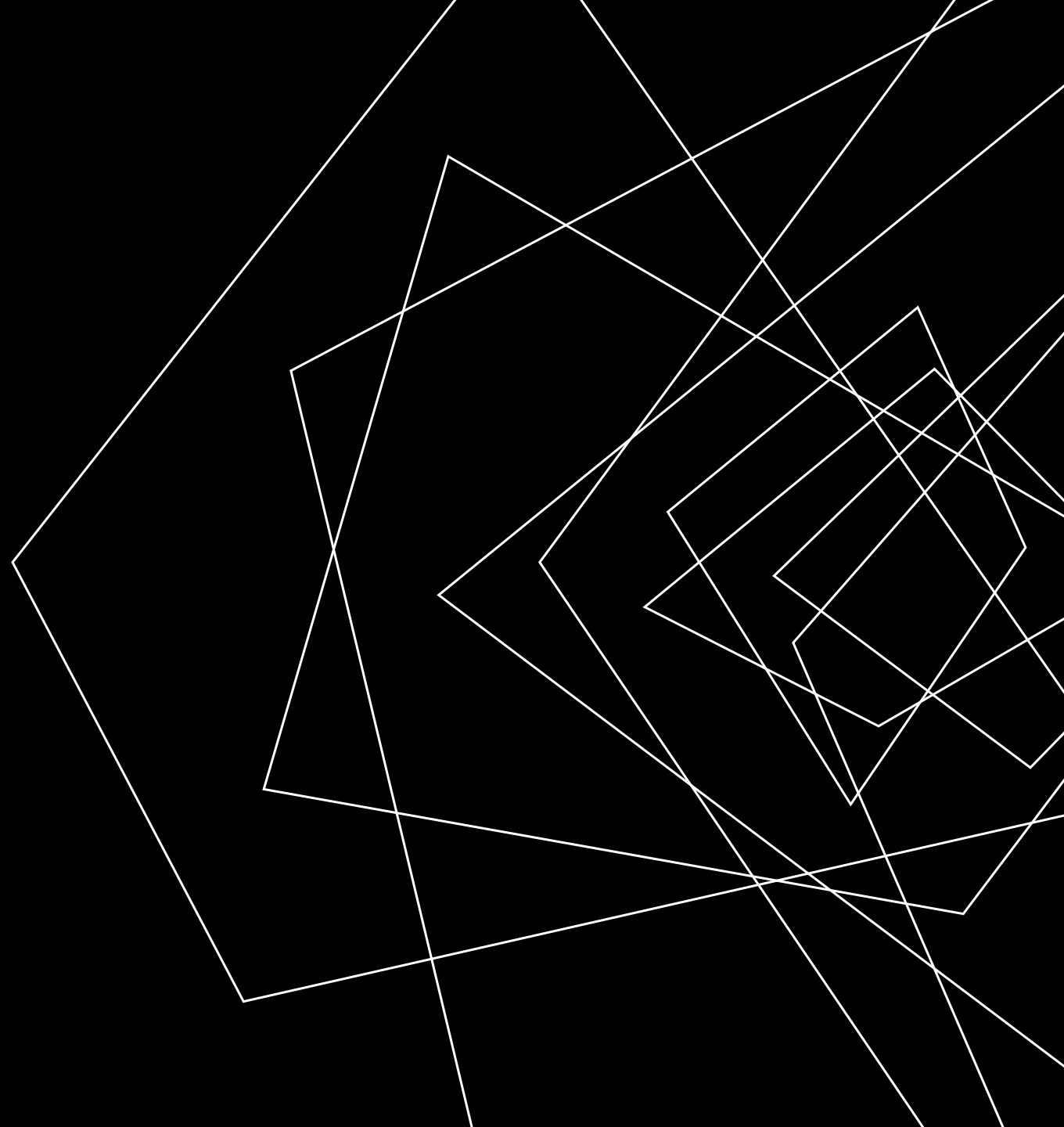
Robert Crump || 11/09/2023

Introduction to Econometrics, ECO 33150

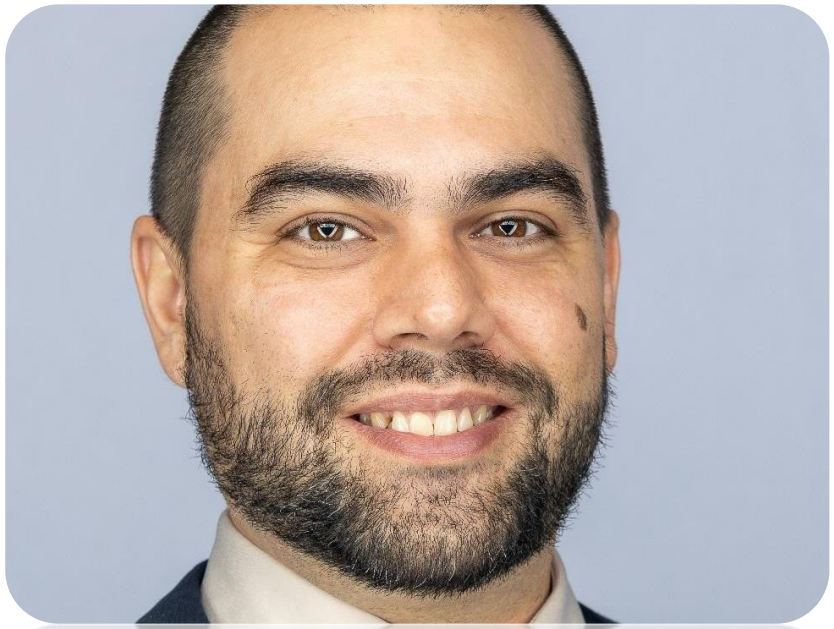
The City University of New York

AGENDA

- Introduction
- Big Picture
- Tradecraft
- Advanced material



ROBERT CRUMP



- Consultant & Data Analyst
- University of Chicago, Master of Public Policy
- Urban land use and transportation focus

GOALS FOR TODAY

- Provide encouragement to keep coding
- Illustrate strategic framework for data analysis
- Showcase some coding techniques

OUTLINE

1) Big Picture

- Pursue your interests
- Use your resources
- Think in Development Stages
- Remain flexible

2) Tradecraft

- EDA cycle
- Tools of the trade
- Useful techniques

3) Advanced Material

- User-defined functions
- GIS
- Simulations

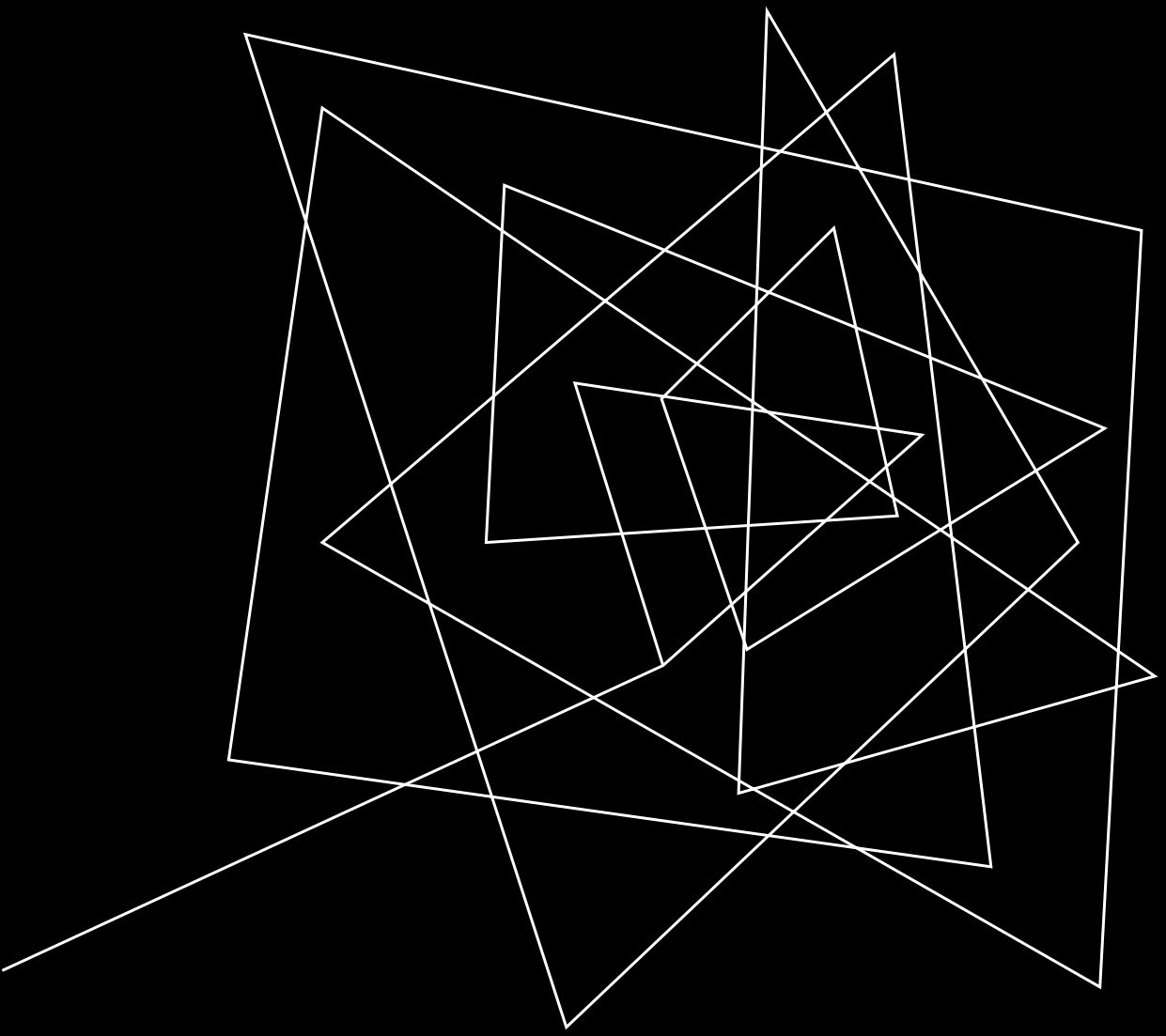
KEY THEMES

What does it mean to pursue a research question?

Creativity comes from immersion

Continuous learning is not a straight path

Skill acquisition requires discomfort



BIG PICTURE

- Pursue your interests
- Use your resources
- Think in development stages
- Remain flexible

ABOVE ALL PURSUE YOUR INTERESTS

Continue learning with projects that keep you engaged.

- Maintain resilience – you will hit roadblocks
- Motivation for quality work
- Organize new information as it comes to you

THE INTERNET IS VAST, DIVE IN!

Your success depends on using resources effectively.

- Package Cheatsheets (dplyr cheatsheet)
- R-Graph Gallery (r-graph-gallery.com)
- R for Data Science (r4ds.hadley.nz)
- Big Book of R (bigbookofr.com)

THINK IN DEVELOPMENT STAGES

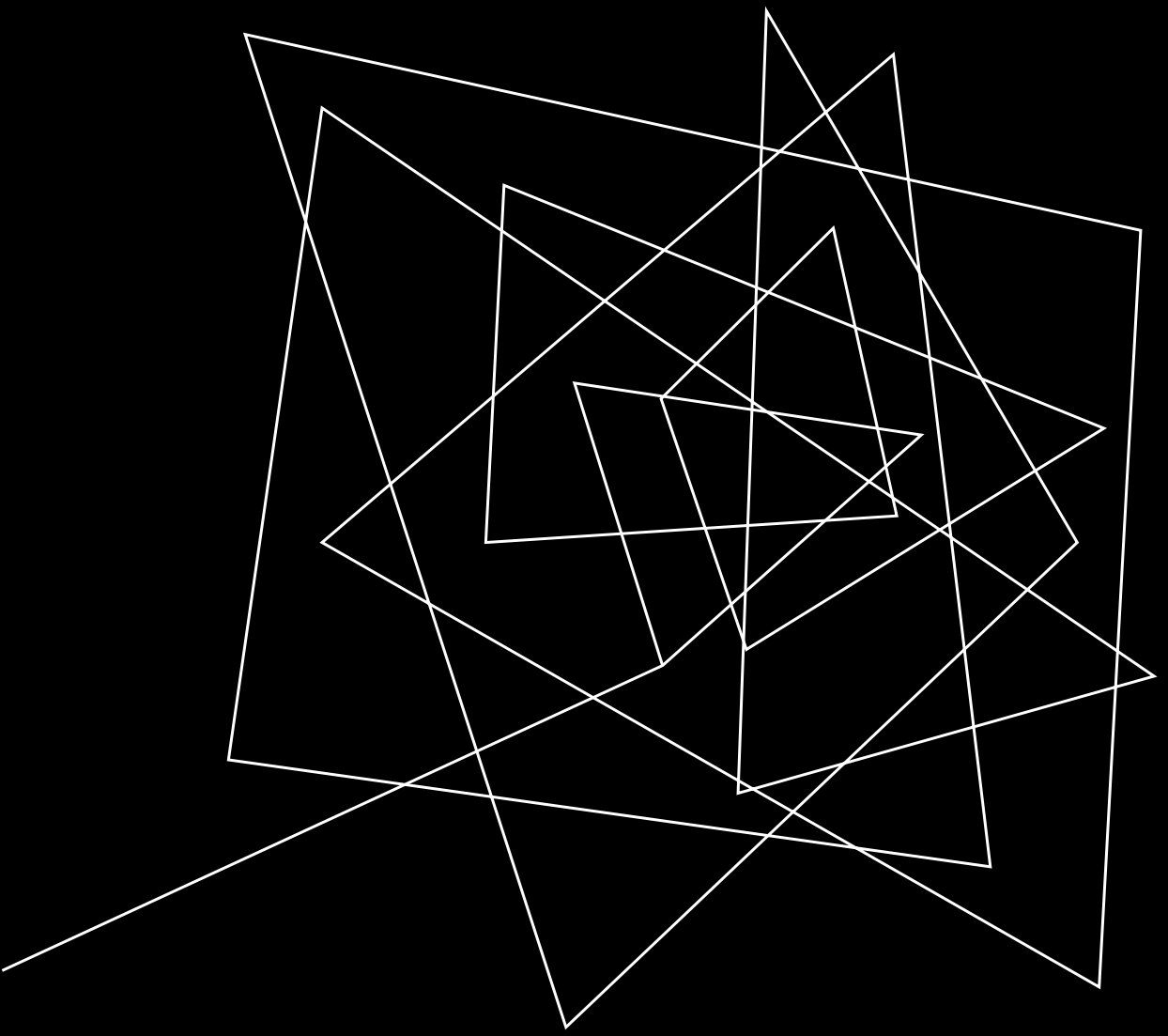
Don’t try to write a whole script.

- Break projects and scripts into smaller pieces
- Write code that is interpretable to you
- Talk to a duck 
- Treat each stage as a step towards concept mastery

REMAIN FLEXIBLE

Research is about discovery, not getting the right answer.

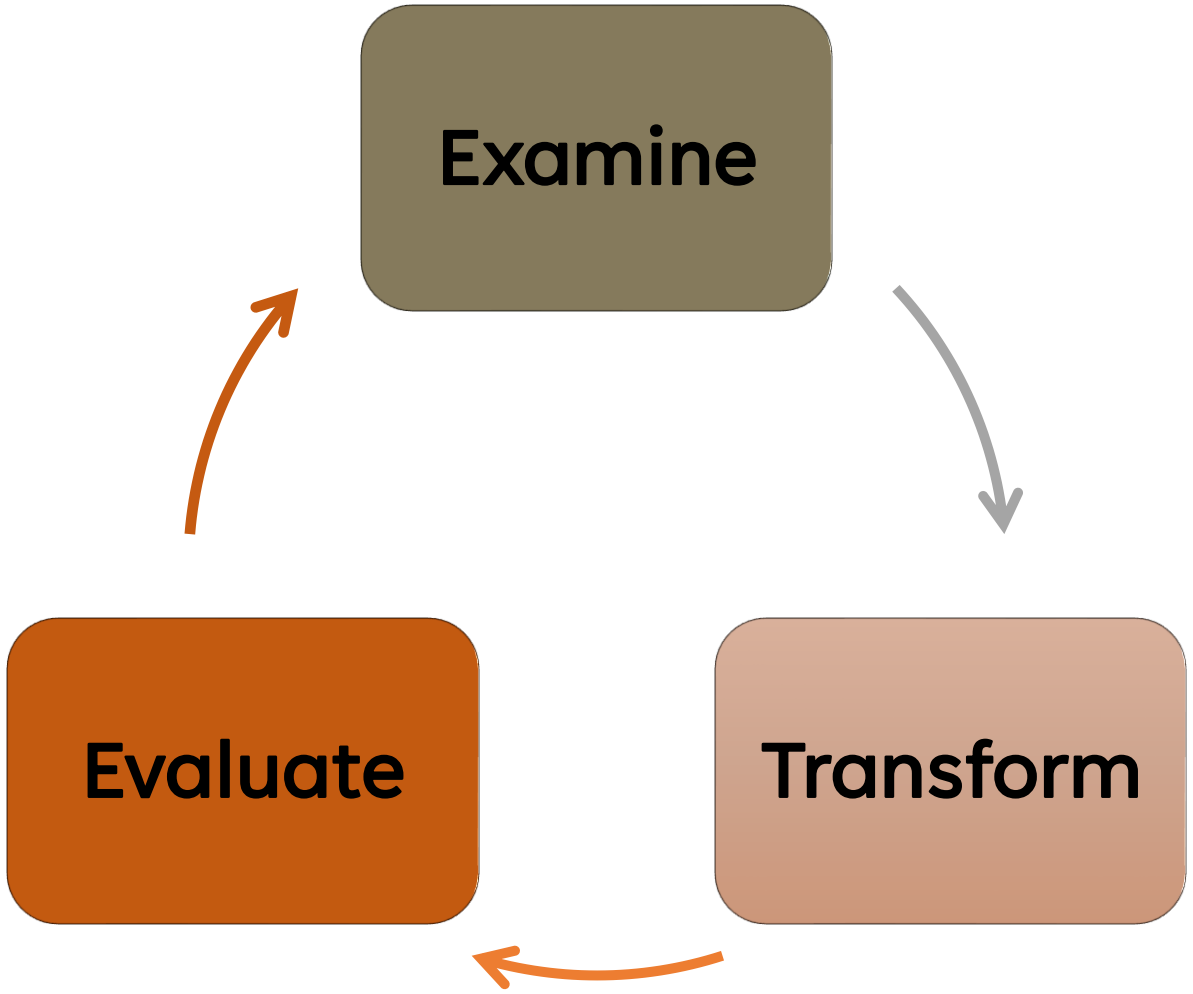
- Development stages are incremental and nonlinear
- Be open to new information
- Take notes, save code snippets
- Get comfortable with hitting dead ends



TRADECRAFT

- EDA cycle
- Tools of the trade
- Useful techniques

EXPLORATORY DATA ANALYSIS (EDA)



EDA IS A TOOL FOR SELF-TEACHING

- Find anomalies, outliers, and errors
- Practice code syntax
- Develop questions about your data
- Propose answers, then attempt to validate your proposals

SETUP

```
library(tidyverse)
library(palmerpenguins)
library(ggplot2)

data("penguins",
     package = "palmerpenguins")

penguins <- palmerpenguins::penguins
```

```
view(penguins)
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
1	Adelie	Torgersen	39.1	18.7	181	3750	male	2007
2	Adelie	Torgersen	39.5	17.4	186	3800	female	2007
3	Adelie	Torgersen	40.3	18.0	195	3250	female	2007
4	Adelie	Torgersen	*NA*	*NA*	*NA*	*NA*	*NA*	2007
5	Adelie	Torgersen	36.7	19.3	193	3450	female	2007
6	Adelie	Torgersen	39.3	20.6	190	3650	male	2007
7	Adelie	Torgersen	38.9	17.8	181	3625	female	2007
8	Adelie	Torgersen	39.2	19.6	195	4675	male	2007
9	Adelie	Torgersen	34.1	18.1	193	3475	*NA*	2007
10	Adelie	Torgersen	42.0	20.2	190	4250	*NA*	2007
11	Adelie	Torgersen	37.8	17.1	186	3300	*NA*	2007
12	Adelie	Torgersen	37.8	17.3	180	3700	*NA*	2007
13	Adelie	Torgersen	41.1	17.6	182	3200	female	2007
14	Adelie	Torgersen	38.6	21.2	191	3800	male	2007
15	Adelie	Torgersen	34.6	21.1	198	4400	male	2007
16	Adelie	Torgersen	36.6	17.8	185	3700	female	2007
17	Adelie	Torgersen	38.7	19.0	195	3450	female	2007

```
glimpse(penguins)
```

```
## Rows: 344
## Columns: 8
## $ species           <fct> Adelie, Adelie, Adelie~
## $ island            <fct> Torgersen, Torgersen, ~
## $ bill_length_mm    <dbl> 39.1, 39.5, 40.3, NA, ~
## $ bill_depth_mm     <dbl> 18.7, 17.4, 18.0, NA, ~
## $ flipper_length_mm <int> 181, 186, 195, NA, 193~
## $ body_mass_g       <int> 3750, 3800, 3250, NA, ~
## $ sex               <fct> male, female, female, ~
## $ year              <int> 2007, 2007, 2007, 2007~
```

```r
class(penguins)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```r
class(penguins$species)
```

```
## [1] "factor"
```

TASK: MANIPULATE DATA

- Narrow focus to variables of interest
- Convert variable class
- Quantify categorical variable
- Quantify numeric variable

```r
penguins_mass <- penguins |>
  select(species, body_mass_g)

view(penguins_mass)
```

	species	body_mass_g
1	Adelie	3750
2	Adelie	3800
3	Adelie	3250
4	Adelie	*NA*
5	Adelie	3450

```r
penguins_mass <- penguins |>
  select(species, body_mass_g) |>
  mutate(species = as.character(species))

class(penguins_mass$species)
```

```
## [1] "character"
```

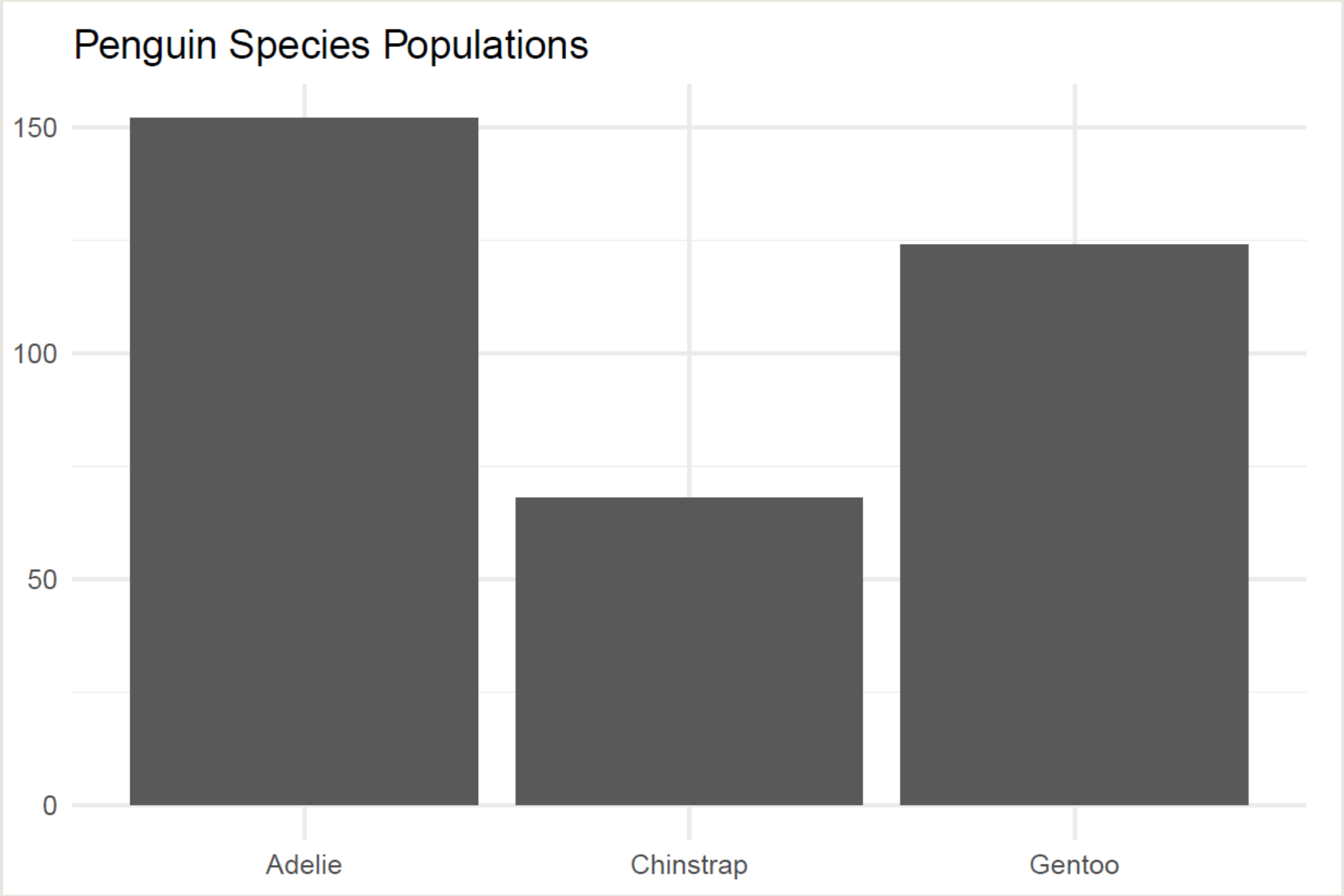
```
glimpse(penguins_mass)
```

```
## Rows: 344
## Columns: 2
## $ species     <chr> "Adelie", "Adelie", "Adelie"~
## $ body_mass_g <int> 3750, 3800, 3250, NA, 3450, ~
```

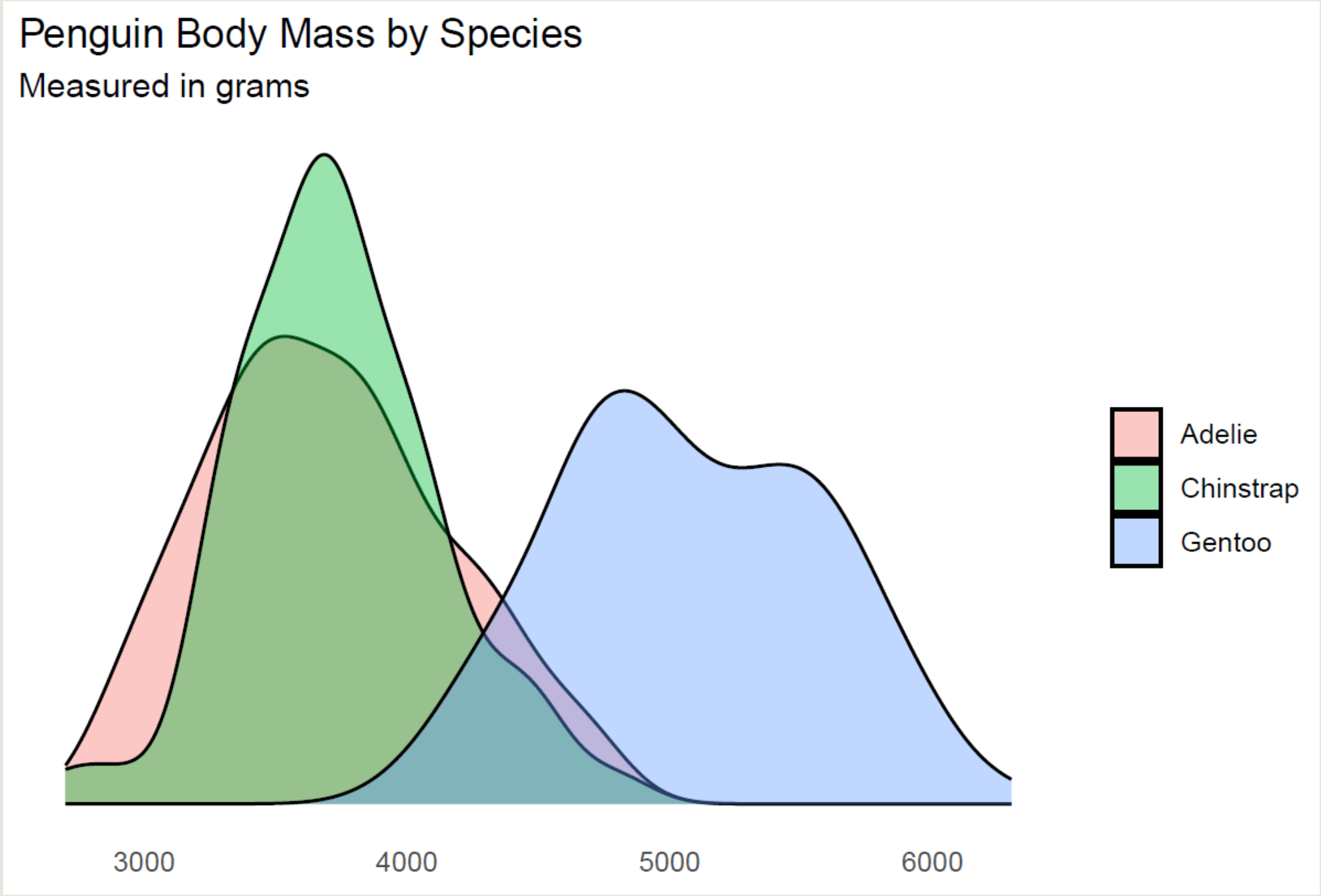
```
count(penguins_mass, species)
```

```
## # A tibble: 3 x 2
##   species       n
##   <chr>     <int>
## 1 Adelie      152
## 2 Chinstrap    68
## 3 Gentoo      124
```

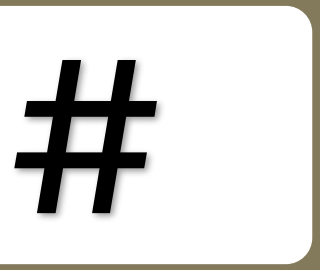
```r
penguins_mass |>
  ggplot(aes(x = species)) +
  geom_bar() +
  labs(title = "Penguin Species Populations",
       x = NULL,
       y = NULL) +
  theme_minimal()
```


Penguin Species Populations
150
100
50
0
Adelie
Chinstrap
Gentoo

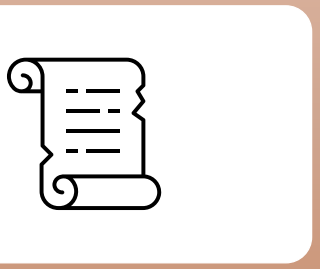
```r
penguins_mass |>
  ggplot(aes(x = body_mass_g, group = species,
             fill = species)) +
  geom_density(alpha = 0.4) +
  labs(title = "Penguin Body Mass by Species",
       subtitle = "Measured in grams",
       x = NULL, y = NULL, fill = NULL) +
  theme_minimal() +
  theme(axis.text.y = element_blank(),
        panel.grid = element_blank())
```


Penguin Body Mass by Species
Measured in grams
3000
4000
5000
6000
Adelie
Chinstrap
Gentoo

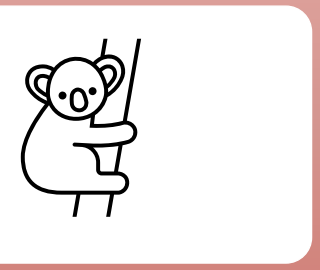
TOOLS OF THE TRADE



Comments



Documentation



Names

WRITE CODE THAT IS INTERPRETABLE TO YOU

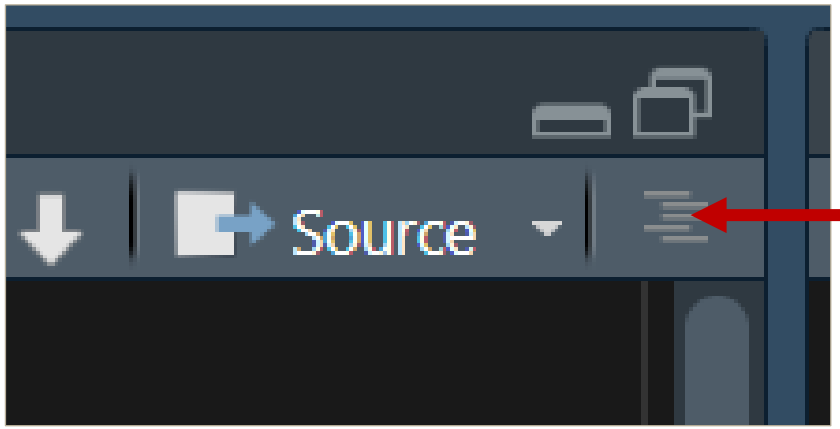
- It’s okay to comment every line; 
- Use active voice and transitive verbs
- Keep scratch paper open to test ideas; EDA cycle
- There is no limit on the number of drafts you can write
- … or data objects you can create

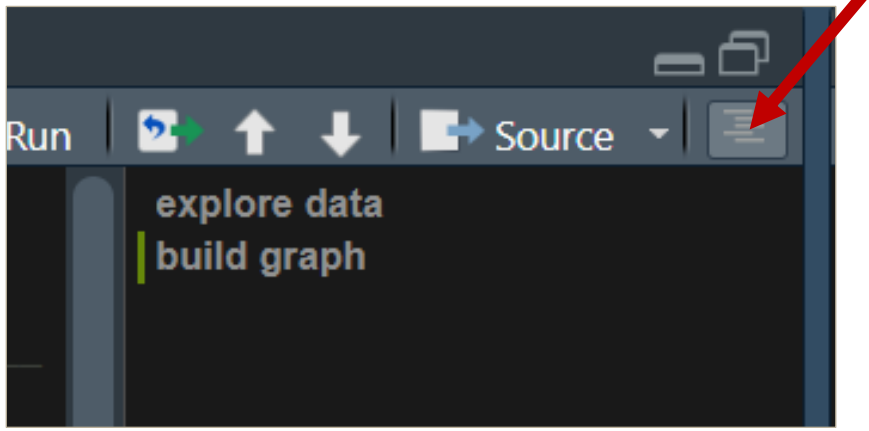
WRITE CODE THAT IS INTERPRETABLE TO YOU

- Commenting greatly enhances efficiency
- `scratch_paper.R` or `test_field.R`
- Debug, run small test changes, clean workspace
- `Ctrl + Shift + r` creates a section break

```r
# explore data ----------------------------------

# narrow focus to two variables
penguins_mass <- penguins |>

  # select species and body mass variables
  select(species, body_mass_g) |>

  # convert species variable to character
  mutate(species = as.character(species))

# check species class
class(penguins_mass$species)
```

```r
# build graph ---------------------------------

penguins_mass |>

  # configure chart
  ggplot(aes(x = body_mass_g,
             group = species, fill = species)) +
  geom_density(alpha = 0.4) +

  # add labels
  labs(title = "Penguin Body Mass by Species",
       subtitle = "Measured in grams") +

  # adjust visual elements
  theme_minimal() +
  theme(axis.text.y = element_blank(),
        panel.grid = element_blank())
```





RSTUDIO OUTLINE BUTTON

**top-right of Source Pane*

HANDLING CATEGORICAL VARIABLES

```
unique(penguins$species)
```

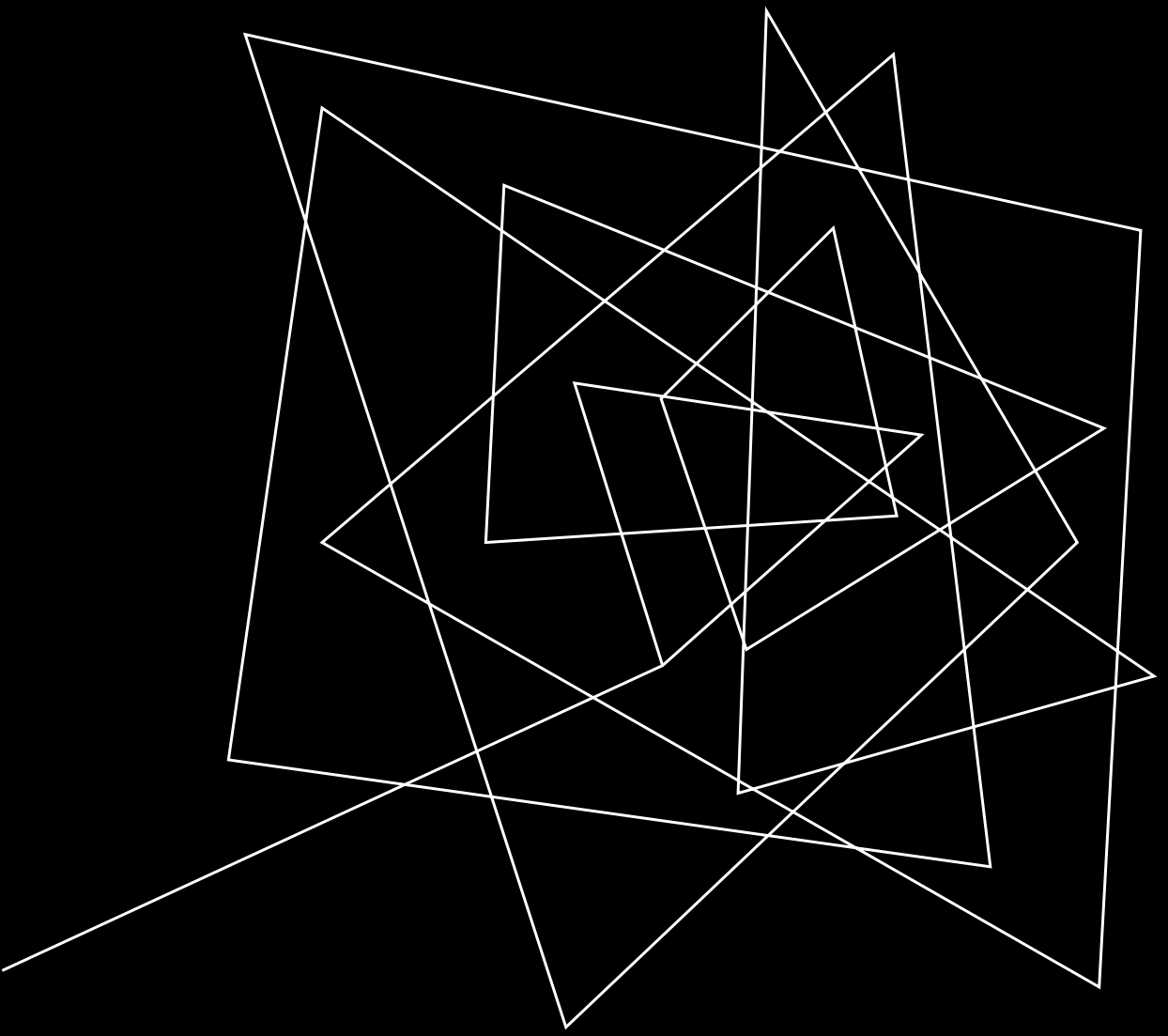
```
## [1] Adelie    Gentoo    Chinstrap
## Levels: Adelie Chinstrap Gentoo
```

```
penguins_species <- levels(factor(penguins$species))

penguins_species
```

```
## [1] "Adelie"    "Chinstrap" "Gentoo"
```

```r
penguins_mass <- penguins |>
  mutate(
    mass_groups = if_else(
      body_mass_g < 3000, "small",
      if_else(
        body_mass_g <= 4000 & body_mass_g > 3000,
        "medium", "large"
        )
      )
    )


?dplyr::if_else
```

```r
penguins_mass <- penguins |>
  mutate(
    mass_groups = case_when(
      body_mass_g < 3000 ~ "small",
      body_mass_g <= 4000 &
        body_mass_g > 3000 ~ "medium",
      body_mass_g > 4000 ~ "large",
      TRUE ~ "other" #capture NA values
      )
    )


?dplyr::case_when
```



ADVANCED MATERIALS

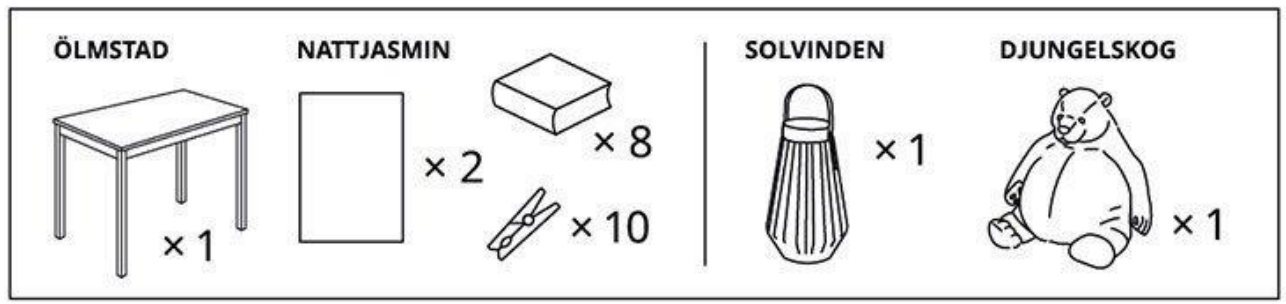
- User-defined functions
- GIS
- Simulations

USER-DEFINED FUNCTIONS ARE POWERFUL TOOLS

You already understand functions.

- User-defined functions are just more explicit and customizable
- Instruction manual for building furniture
- ... or like writing a recipe, etc
- Flexible tool for building new data structures

HÖUSE



Make sure that the structure is safe. Do not leave children unattended.
The suggested examples are not ofcial IKEA user guides for IKEA products.
If you can't And the products referred to in the instructions, use similar ones.

WHEN TO WRITE A FUNCTION

- You've copied and pasted a block of code more than twice*
- You want to clarify and / or standardize a process
- You want to automate a multi-step process
- You have a unique goal and / or unusually shaped data
- Keeps code D.R.Y.

*https://r4ds.hadley.nz/functions

FUNCTION STRUCTURE

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

- `name` : Function name, stored as an object in R environment
- `arguments` : Variable / vector the function iterates over
- `body` : Instructions applied to `arguments`

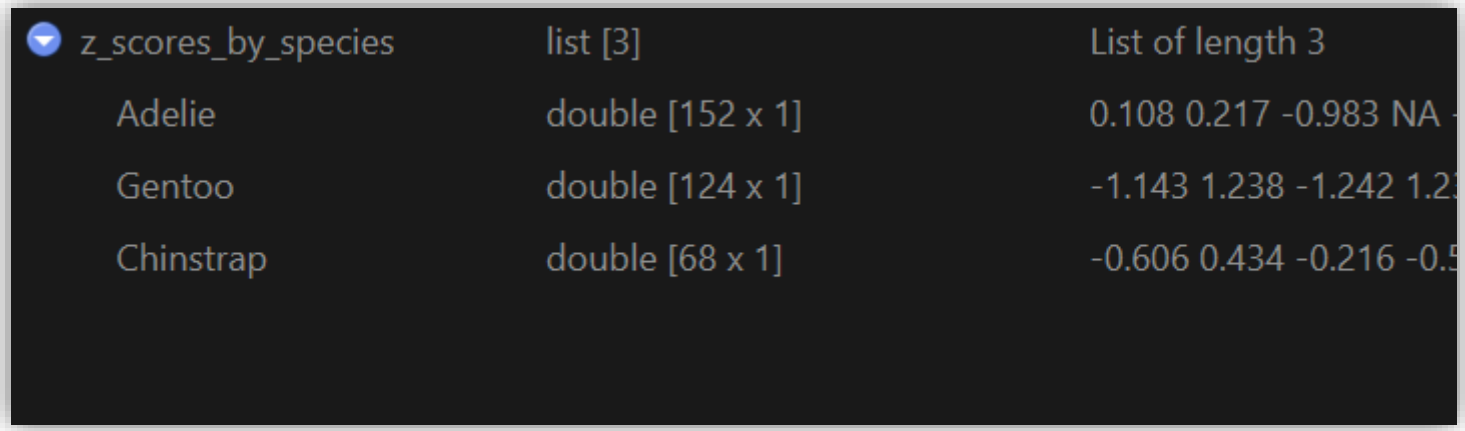
TASK: CALCULATE Z-SCORE OF ANIMAL MASS

- Accept standard `data.frame` input
- Calculate z-score by species
- Store results in standard `list` output
- Use ChatGPT !

```r
calculate_z_scores_by_species <- function(df) {

  # List to store z-scores for each species
  z_scores_by_species <- list()

  # Get unique species in the dataset
  unique_species <- unique(df$species)

  # Calculate z-scores for each species
  for (species in unique_species) {
    subset_data <- df[df$species == species, ]
    z_scores <- scale(subset_data$body_mass_g)
    z_scores_by_species[[species]] <- z_scores
  }
  return(z_scores_by_species)
}
```

```
calculate_z_scores_by_species(penguins)
```

```
## $Adelie
##                [,1]
##    [1,]  0.10759
##    [2,]  0.21663
##    [3,] -0.98276
##    [4,]       NA
##    [5,] -0.54662
##    [6,] -0.11048
##    [7,] -0.16500
##    [8,]  2.12475
##    [9,] -0.49210
##   [10,]  1.19795
```



CUSTOM FUNCTIONS ARE EXTENSIBLE

```
calculate_z_scores_by_species(bears)

calculate_z_scores_by_species(fish)

calculate_z_scores_by_species(krakens)
```

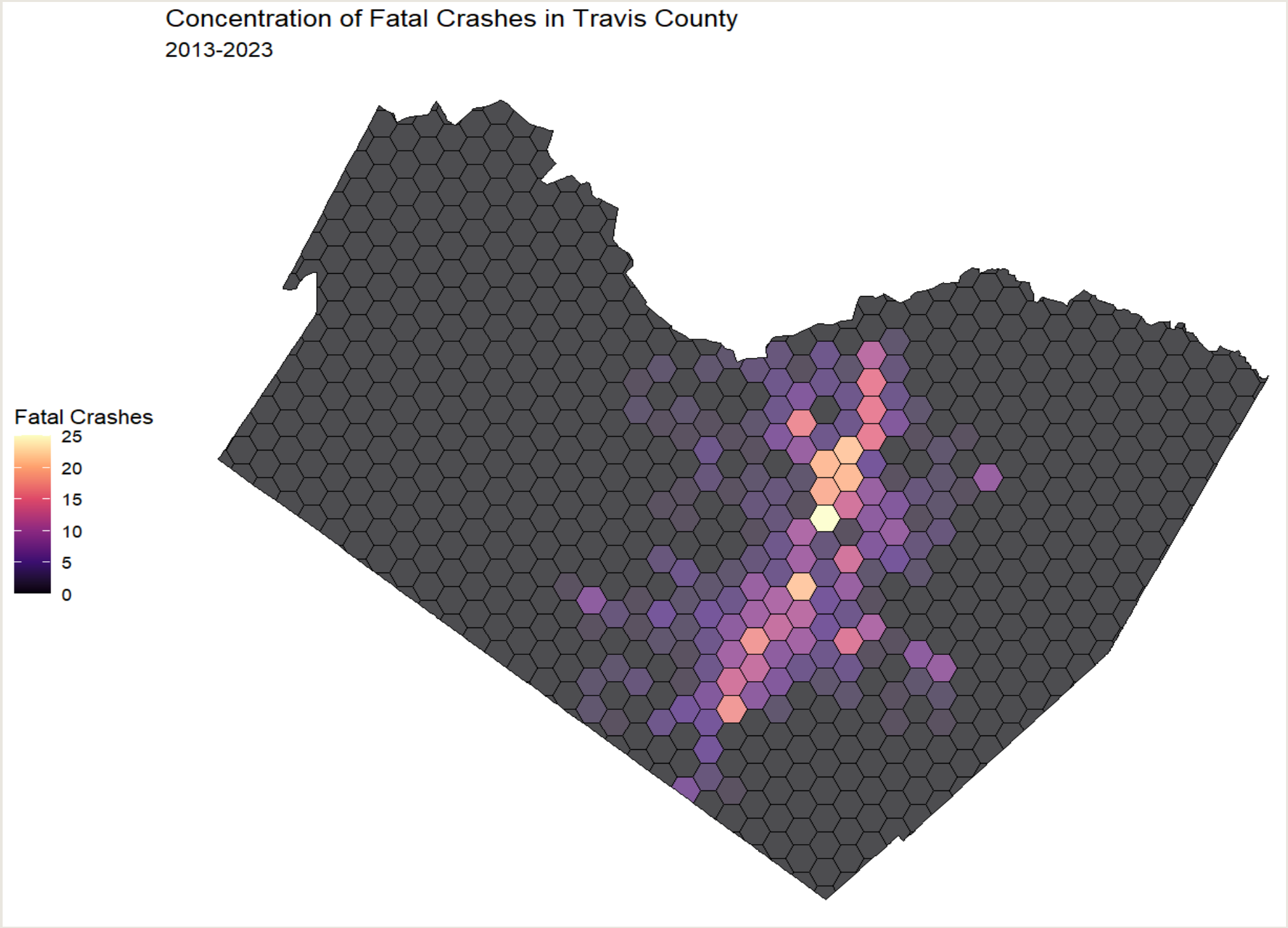
provided you have the same variables in each data set

GEOGRAPHIC INFORMATION SYSTEMS (GIS)

- Geographic information conveys the spatial nature of data
- Maps are data models
- Spatial elements provide important context to policy makers

GIS IN R

- `sf` package will do most of everything you will ever need
- `ggplot::geom_sf`
- Well-integrated into `tidyverse`
- https://r-spatial.github.io/sf/


Concentration of Fatal Crashes in Travis County
2013-2023
Fatal Crashes
25
20
15
10
5
0

MY RESEARCH QUESTION

- What impact does Interstate 35 have on fatal vehicle collisions in the Austin metro region?

- Null Hypothesis: “I-35 ***does not*** have a statistically significant impact on fatal vehicle collisions in the Austin metro region.”

- Alternative Hypothesis: “I-35 ***does have*** a statistically significant impact on fatal vehicle collisions in the Austin metro region.”

- Empirical statement: “A driver in Austin is ***y*** times more likely to be involved in a fatal vehicle collision within ***x*** proximity to I-35.”

DEVELOPMENT STAGE 1

- Geographic EDA – build a heat map
- Validate measurements, observe patterns, plan Stage 2
- Clean and organize data
- Organize project folder directory
- Start building documentation

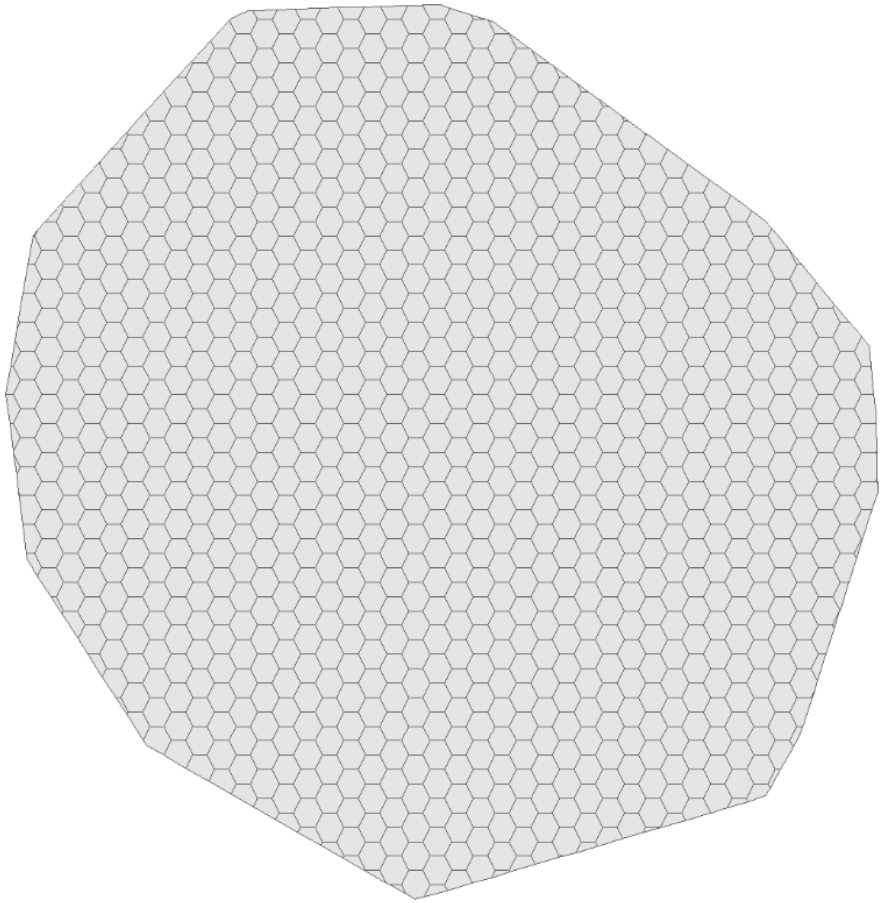
PROBLEMS WITH MY RESEARCH QUESTION

- Complex geographic environment reduced to 2 dimensions
- Traffic patterns are agglomerations of many individual decisions and movements
- Subject behavior (driving) is not accounted for
- Aggregated measures are not sufficient for statistical measures

DEVELOPMENT STAGE 2

- Resolve issues identified in Stage 1
- Build custom function to simulate subject behavior
  - *Drivers moving through space*
- Change study area base map
- Learn to love the struggle

CHANGE STUDY AREA BASE MAP



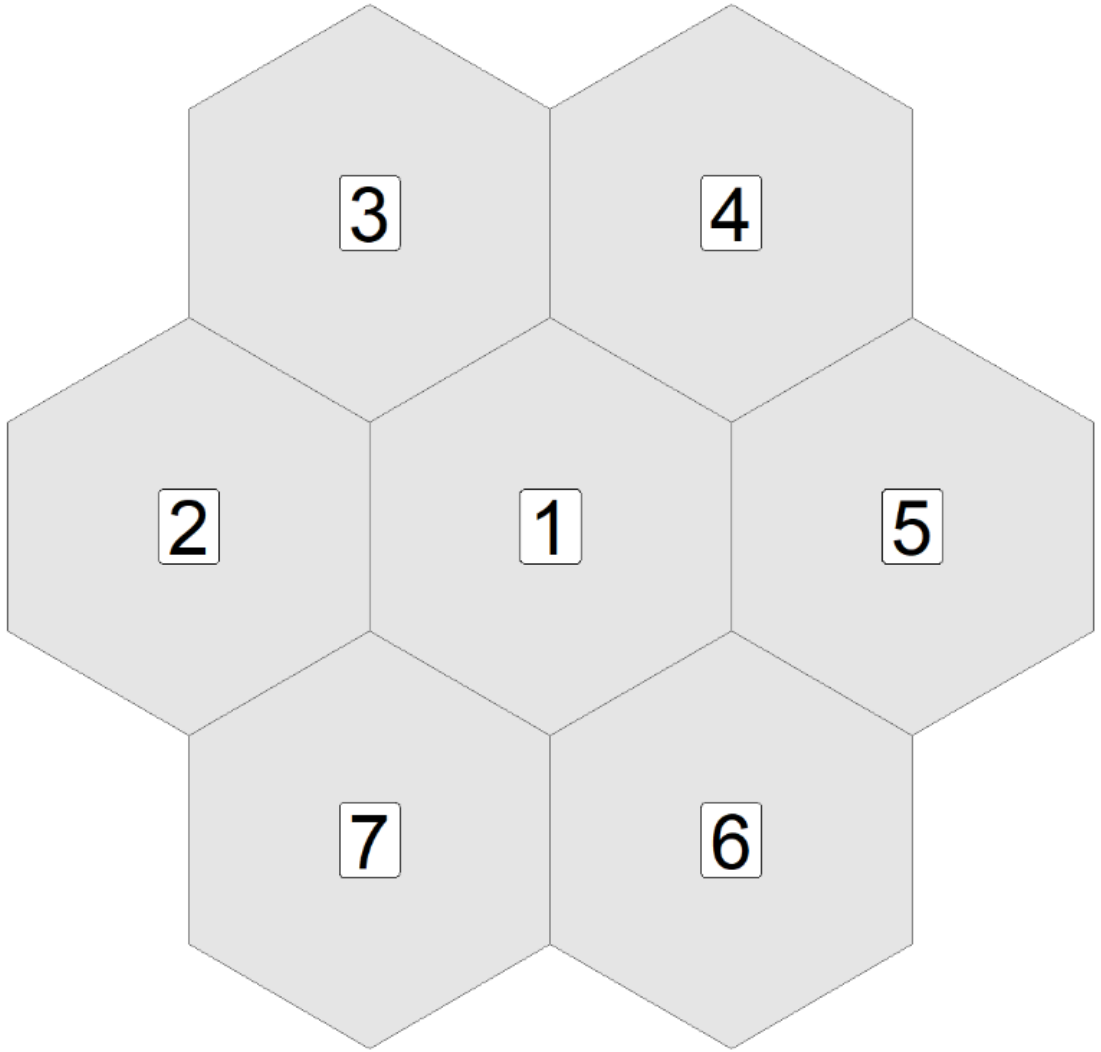
CHANGE STUDY AREA BASE MAP

```
atx_city_boundary <- st_read("atx.shp")

atx_hull <- atx_city_boundary |>

  #merge multi-line object into hull
  st_combine() |>
  st_convex_hull() |>

  #generate sf object
  st_as_sf()
```

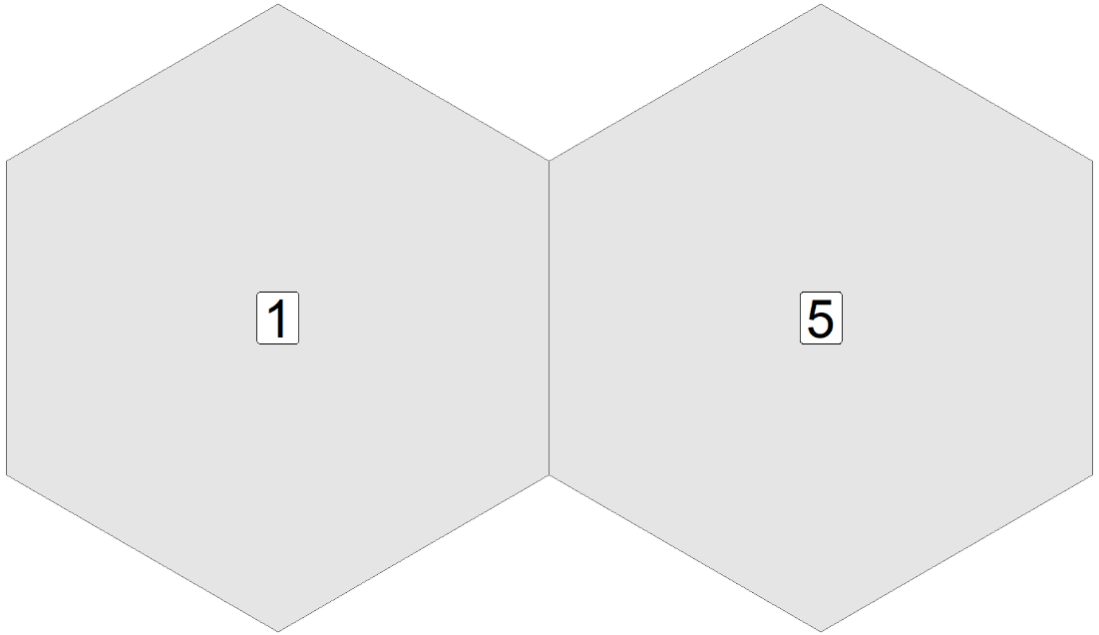
```
atx_hex <- atx_hull |>

  # convert polygon to grid
  st_make_grid(n = 35, square = F,
               flat_topped = T) |>
  st_intersection(atx_hull) |>
  st_as_sf() |>

  # ad unique ids to hexes
  mutate(hex_id = row_number())
```

SIMULATE MOVEMENT

1. Find adjacent hexagons



2. Sample from list

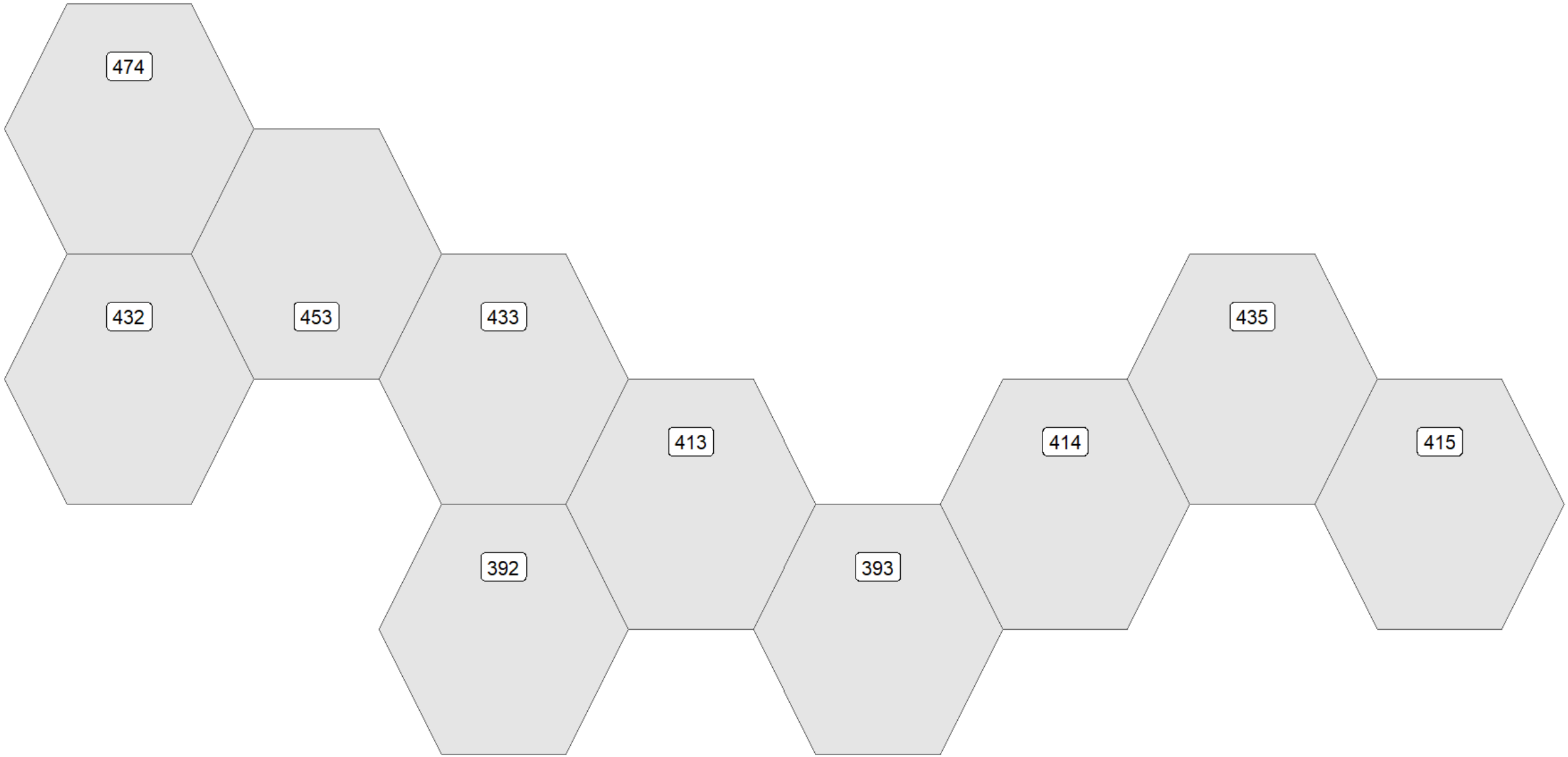


TASK: CREATE SEQUENTIAL SAMPLING FUNCTION

- Start with random point on hex grid
- Identify adjacent hexes (“neighbors”)
- Sample from list of neighbors
- Append chosen neighbors in sequence to form path

```r
# build list of all hex neighbors by self-referencing
adjaceny_matrix = sf::st_touches(atx_hex, atx_hex)

# pull 10 random hexes
start_pts = as.integer(sample(atx_hex$hex_id, 10))

# begin single path
path_1 = sample(start_pts, 1)
```

```r
# run loop to take and append steps to path
for(i in 1:10){

  # pull neighbors list from adjaceny matrix
  neighbors = adjaceny_matrix[[tail(path[i])]]

  # exclude previous steps, prevent backtracking
  neighbors = neighbors[!neighbors %in% path]

  # choose step
  nth.step = sample(neighbors, 1)

  # take and append step to path
  path = c(path, nth.step)
}
```


474
432
453
433
413
392
393
414
435
415

TASK: RUN SINGLE PATH FUNCTION MANY TIMES

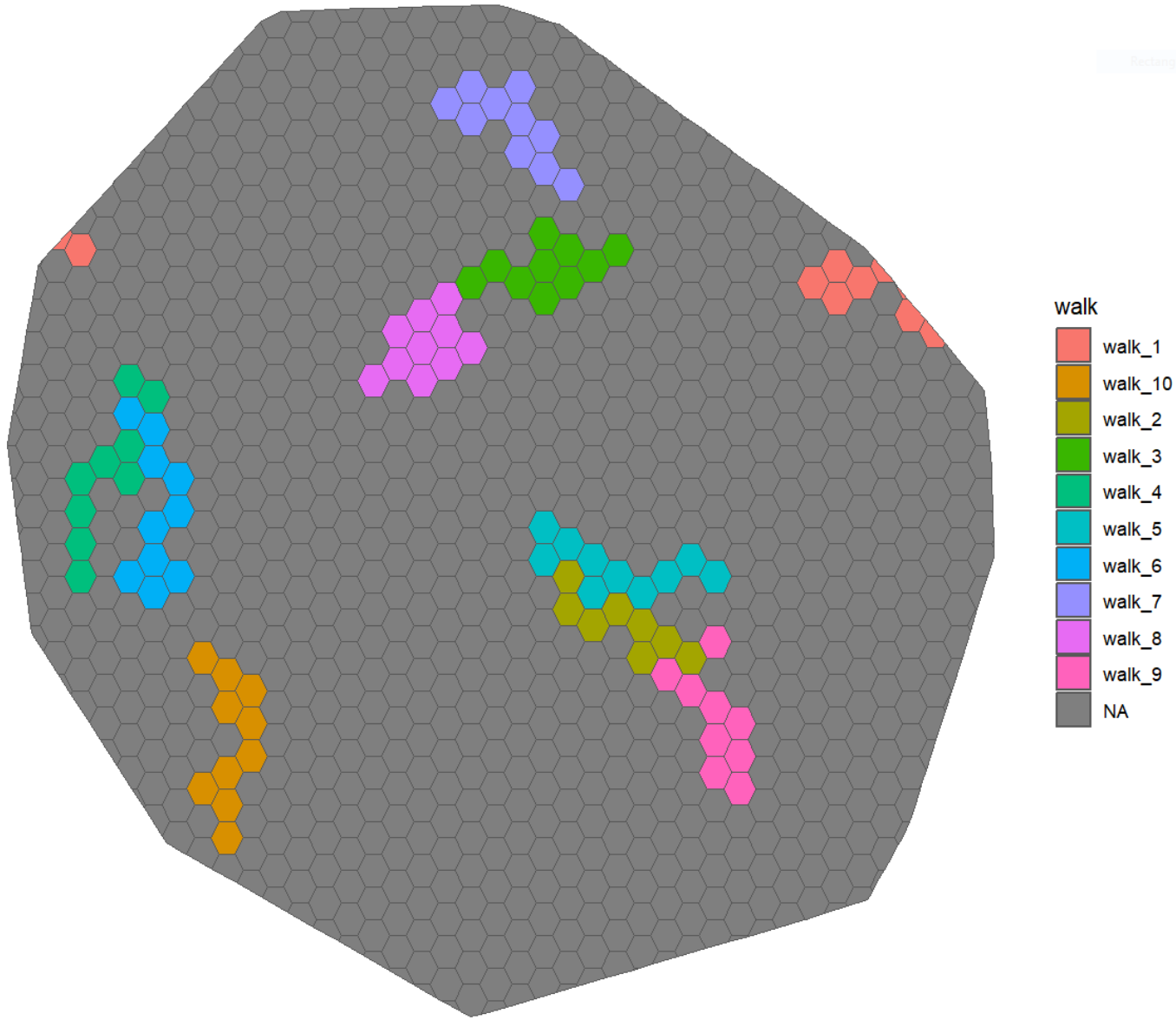
- Sample group of random point on hex grid
- Run sequence from each start point
- Collect results into a list

```r
multi_path <- function(df, n.paths, n.steps){

  ## create objects ##

  # generate unique ids for each polygon
  df$id = 1:nrow(df)

  # produce adjacency matrix with `sf` package
  adjaceny_matrix = sf::st_touches(df, df)

  # pull sample of ids into integer vector
  start_pts = as.integer(sample(df$id, n.paths))
```

```r
## define internal looping function ##
single_path <- function(x){

  # choose starting polygon
  path = sample(start_pts, 1)

  # loop for n.steps
  for(i in 1:n.steps){

    # find potential steps
    neighbors = adjaceny_matrix[[tail(path[i], 1)]]
    neighbors = neighbors[!neighbors %in% path]

    # choose forward step
    nth.step = sample(neighbors, 1)

    # append step to path
    path = c(path, nth.step)
  }
```

```r
    # remove final step to match user input
    # start_pts[i] counts as step 1
    # might be a better way to do this
    path = tail(path, -1)

    return(path)
  }

  ## compile loops into list
  paths_list <- lapply(start_pts, single_path)

  # assign names to list
  names(paths_list) = paste0(
    "walk_", as.character(1:length(paths_list))
  )

  return(paths_list)
}
```

```
multi_path(atx_hex, 10, 10)
```



DEVELOPMENT STAGE 3

- Debug and stress test `multi_path`
- Read and study machine learning documentation
- Evaluate and choose machine learning model
- Define empirical statement mathematically

ACKNOWLEDGEMENTS

Dr. Peter Ganong - voices.uchicago.edu/ganong

Dr. Finoa Burlig - fionaburlig.com

Dr. Hadley Wickham - hadley.nz

Dr. Patt Schloss - schlosslab.org

THANK YOU

rwrcrump.co