Week 2
Tidyverse & Manipulating Data

Data Visualization for Social Scientists

Lecturer: Jeffrey Ziegler, PhD
Teaching Fellow: Shekhar Kedia

ASDS - Trinity College Dublin

Spring 2026

Road map for today

- Today:
  - Work with rectangular data using `gapminder` dataset
  - Manipulate data with core `dplyr` verbs
  - Summarize amounts and proportions and plot them with `ggplot2`
    - Practice histograms, density plots, ridge plots, boxplots, violins, and jittered points
    - Emphasize choices (bin width, bandwidth, outlier rules) that change how uncertainty appears
- By next week, please...
  - Problem set #2

Key dplyr verbs

- **filter()**: extract rows (cases) that meet logical conditions
- **select()**: choose columns (variables) from a data frame
- **arrange()**: sort rows by one or more variables
- **mutate()**: create or transform columns
- **group_by() + summarize()**: compute group-level summaries

FILTER(): SUBSETTING ROWS

- Pattern: `filter(data, condition)` returns rows where `condition` is TRUE
- Example: `filter(gapminder, country == "Denmark")` returns all Denmark rows
- Uses logical operators such as `==`, `<`, `>`, `<=`, `>=`

Logical tests

- `x < y`: less than
- `x > y`: greater than
- `x == y`: equal to (used for testing, not assignment)
- `x <= y`, `x >= y`: less/greater than or equal to

Ex: Studying African Farmer-Led Irrigation

- SAFI study looked at farming and irrigation methods in Tanzania and Mozambique
- Survey data was collected through interviews conducted between November 2016 and June 2017

Variable	Description
village	Village name
interview_date	Date of interview
no_membrs	How many members in the household?
years_liv	How many years have you been living in this village or neighboring village?
rooms	How many rooms in the main house are used for sleeping?
memb_assoc	Are you a member of an irrigation association?
affect_conflicts	Have you been affected by conflicts with other irrigators in the area?
liv_count	Number of livestock owned
no_meals	How many meals do people in your household normally eat in a day?

Ex: Filtering with filter()

```r
# Import dataset and load library
SAFI <- read.csv("https://raw.githubusercontent.com/ASDS-TCD/
    DataViz_2026/refs/heads/main/datasets/SAFI.csv")

# All households in village "God"
SAFI_god <- filter(SAFI, village == "God")

# Households with more than 6 members
large_households <- filter(SAFI, no_membrs > 6)

# Households in "God" or "Ruaca" with more than 4 members
god_ruaca_large <- filter(SAFI, village %in% c("God", "Ruaca") &
    no_membrs > 4)
```

Ex: Selecting with select()

```r
# Keep only identification and location variables
id_loc <- SAFI |>
  select(key_ID, village)

# Move village to the first position
village_first <- SAFI |>
  select(village, everything())

# Drop columns related to survey timing
no_timing <- SAFI |>
  select(-starts_with("interview_date"))
```

Ex: Arranging data with arrange()

```r
# Households ordered by size (ascending)
SAFI_by_size <- SAFI |>
  arrange(no_membrs)

# Households ordered by size (descending) within village
SAFI_by_village_size <- SAFI |>
  arrange(village, desc(no_membrs))
```

mutate(): creating new variables

- Pattern: `mutate(data, new_var = expression)` adds or modifies columns
- Examples: indicator for African countries, logged GDP per capita, classification of regions
- Often combined with `group_by()` and `summarize()` for richer summaries

Ex: Adding new columns with mutate()

```r
# Add a logical indicator for "large" households
SAFI <- SAFI |>
  mutate(large_hh = no_membrs >= 6)

# Years of living in village in decades
SAFI <- SAFI |>
  mutate(living_decades = years_liv / 10)
```

Ex: Combining multiple verbs with pipes (|>)

```r
# Categorize household size
SAFI <- SAFI |>
  mutate(
    hh_size_cat = case_when(
      no_membrs <= 3 ~ "small",
      no_membrs >= 4 & no_membrs <= 7 ~ "medium",
      no_membrs > 7 ~ "large"
      )
  )
```

```r
# Large households in Ruaca with selected columns
ruaca_large_small_df <- SAFI |>
  filter(village == "Ruaca", no_membrs > 6) |>
  mutate(
    large_hh = TRUE,
    members_per_room = no_membrs / rooms
  ) |>
  select(village , no_membrs, rooms, members_per_room, large_hh)
```

Two-step workflow for proportions

- Step 1: Summarize data with `dplyr`, often using `group_by()` and `summarize()`
- Step 2: Visualize summarized data with `ggplot2`
- Clarifies how proportions are computed before plotting them

Summarizing with group_by() and summarize()

- Workflow: `data |> group_by(group_vars) |> summarize(stats)`
- Typical statistics: mean, median, min, max, count, etc.
- Used to compute quantities like median house members by village

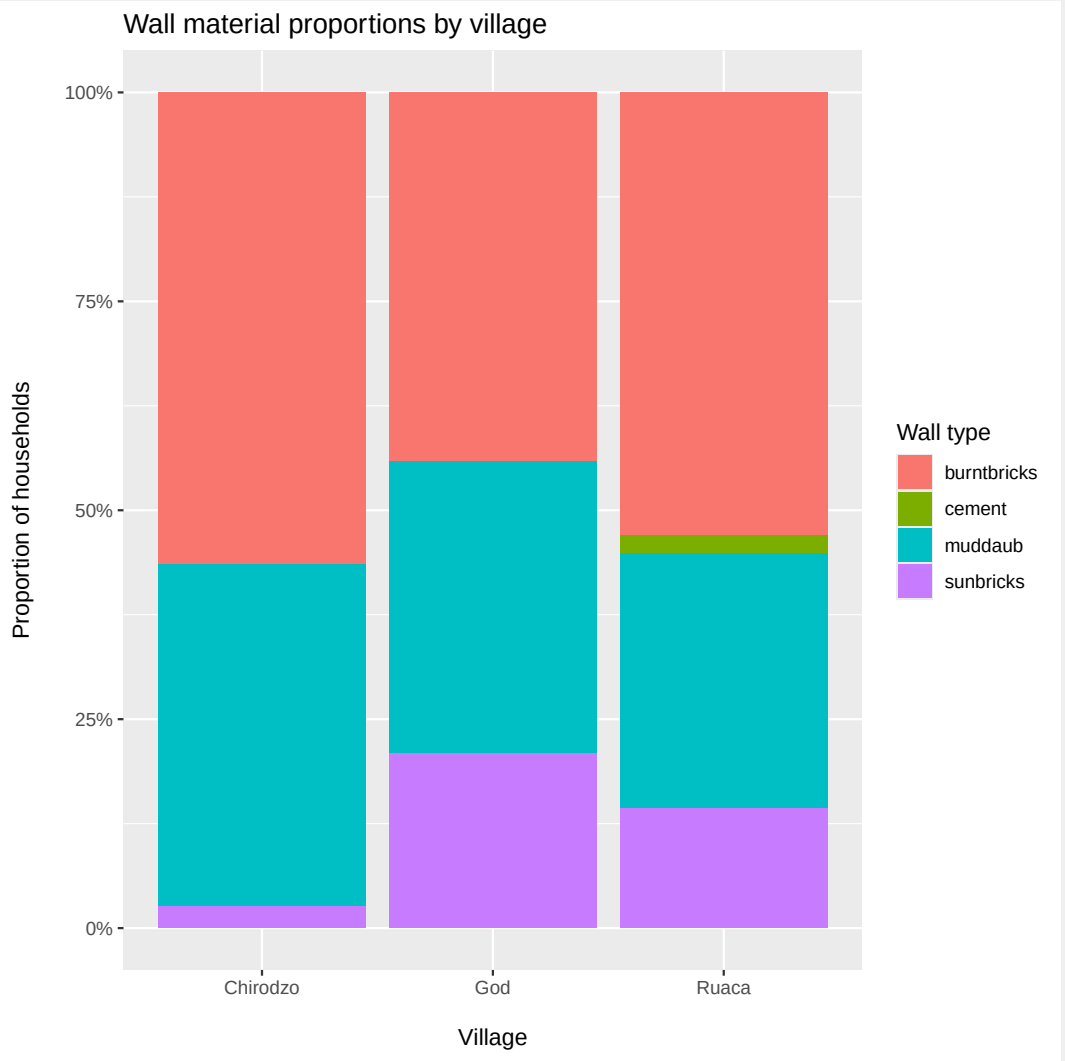
```r
# Mean household size by village
village_hh_summary <- SAFI |>
  group_by(village) |>
  summarize(
    mean_members = mean(no_membrs, na.rm = TRUE),
    median_members = median(no_membrs, na.rm = TRUE),
    n_households = n(),
    .groups = "drop"
  )
```

Ex: Proportions with group_by() |> summarize()

```r
# Mean years in village by roof type
wall_living_summary <- SAFI |>
  count(wall_type, village, name = "n") |>
  group_by(village) |>
  mutate(prop = n / sum(n))
```

Ex: Proportions with group_by() |> summarize()

```
ggplot(wall_living_summary,
       aes(x = village, y =
     prop, fill = wall_type)
      ) +
  geom_col(position = "fill")
      +
  scale_y_continuous(labels =
       scales::percent_format
      ()) +
  labs(
    x = "\nVillage",
    y = "Proportion of
      households\n",
    fill = "Wall type",
    title = "Wall material
      proportions by village"
  )
```



From summaries to graphics

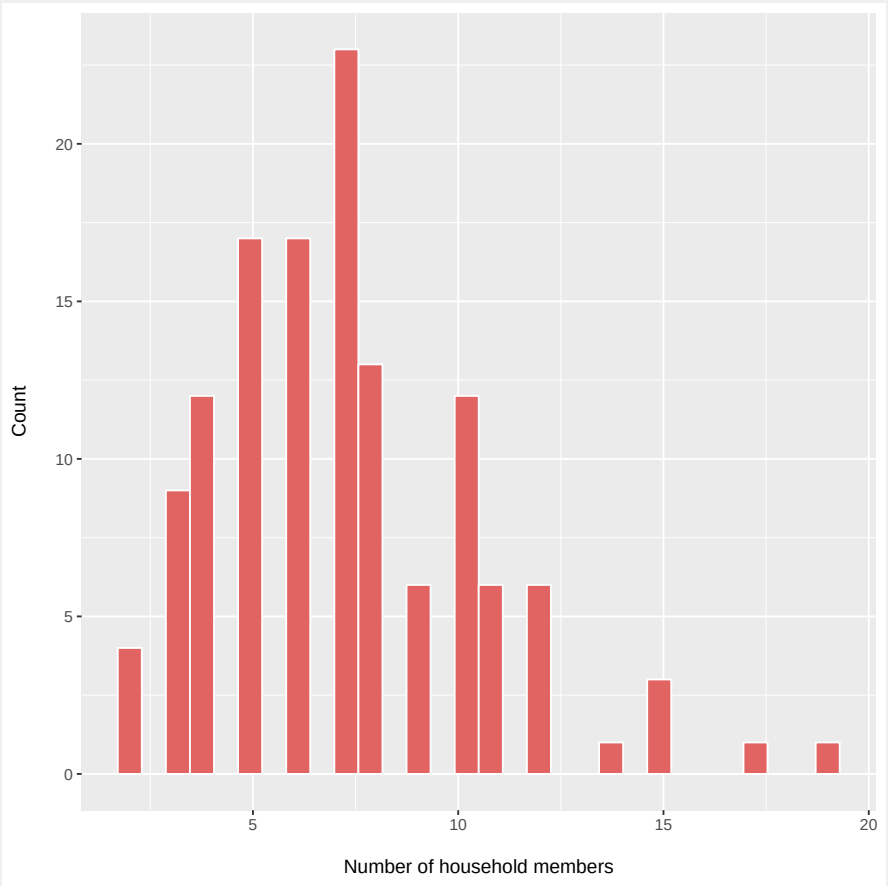
- Use summarized tables (e.g., counts or proportions by group) as ggplot inputs
- Choose appropriate geometries (bars, lines, etc.) to represent amounts and proportions
- Emphasize truthful, clear visual communication of aggregated data

Histograms and bin width

- Histograms show counts of observations within ranges (bins) of a variable such as `no_membrs`
- Bin width strongly affects the apparent shape of the distribution and must be chosen deliberately
- Options include `binwidth`, bar border `color`, fill color (e.g. #E16462), and `boundary` to align bins at whole numbers

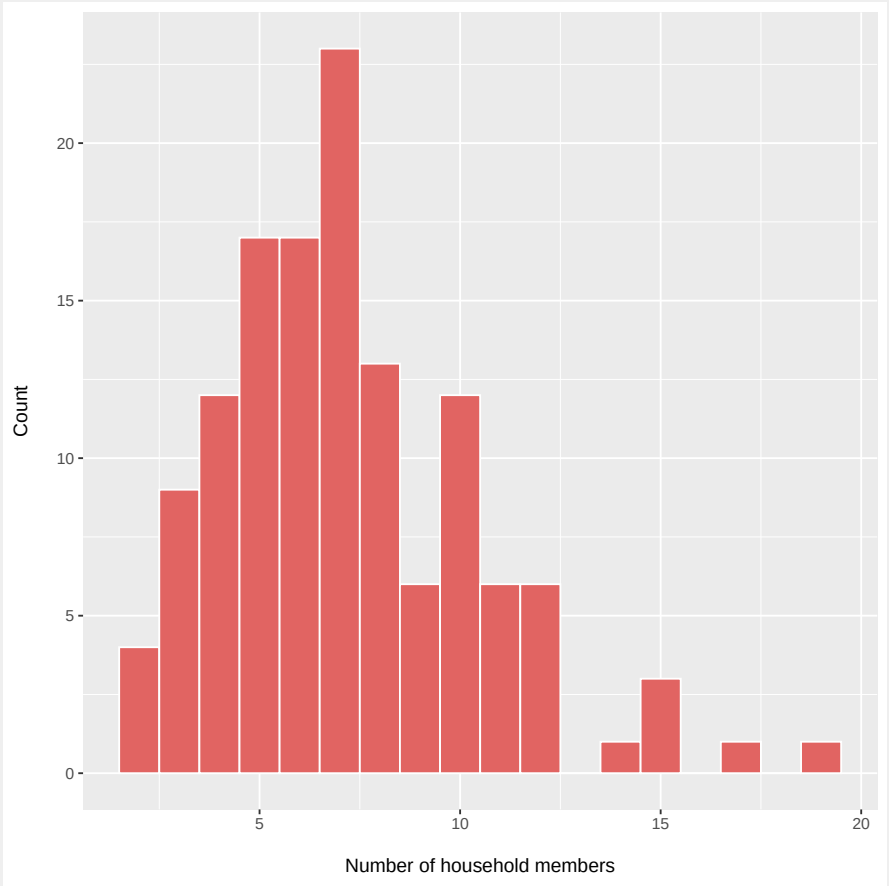
Ex: Histogram of household size

Default bin widths (=30)




```
ggplot(SAFI, aes(x = no_membrs)) +
  geom_histogram(color = "white", fill = "#
      E16462") +
  labs(x = "\nNumber of household members",
       y = "Count\n")
```

bin=1



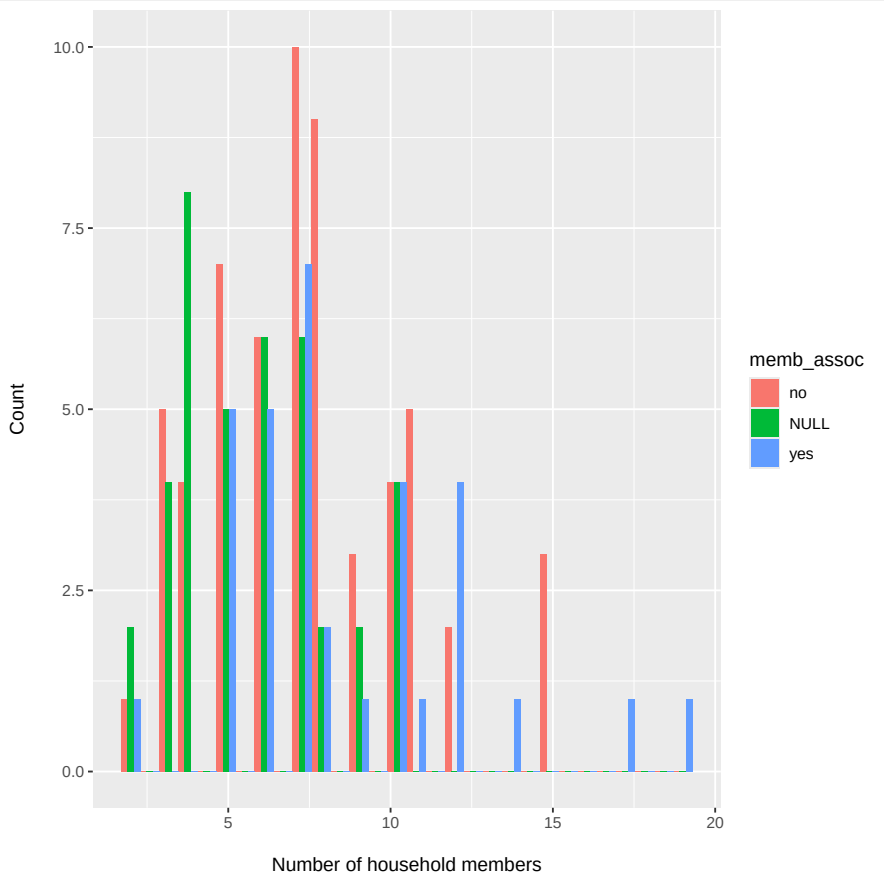

```
ggplot(SAFI, aes(x = no_membrs)) +
  geom_histogram(binwidth = 1, color = "
      white", fill = "#E16462") +
  labs(x = "\nNumber of household members",
       y = "Count\n")
```

Histograms with groups

- Additional variables like `memb_assoc` (member of irrigation association) can be mapped to `fill`
- Overlaid histograms with multiple fills can be hard to interpret when groups overlap
- Faceting (e.g. `facet_wrap(vars(village))`) separates groups into small multiples for clearer comparisons

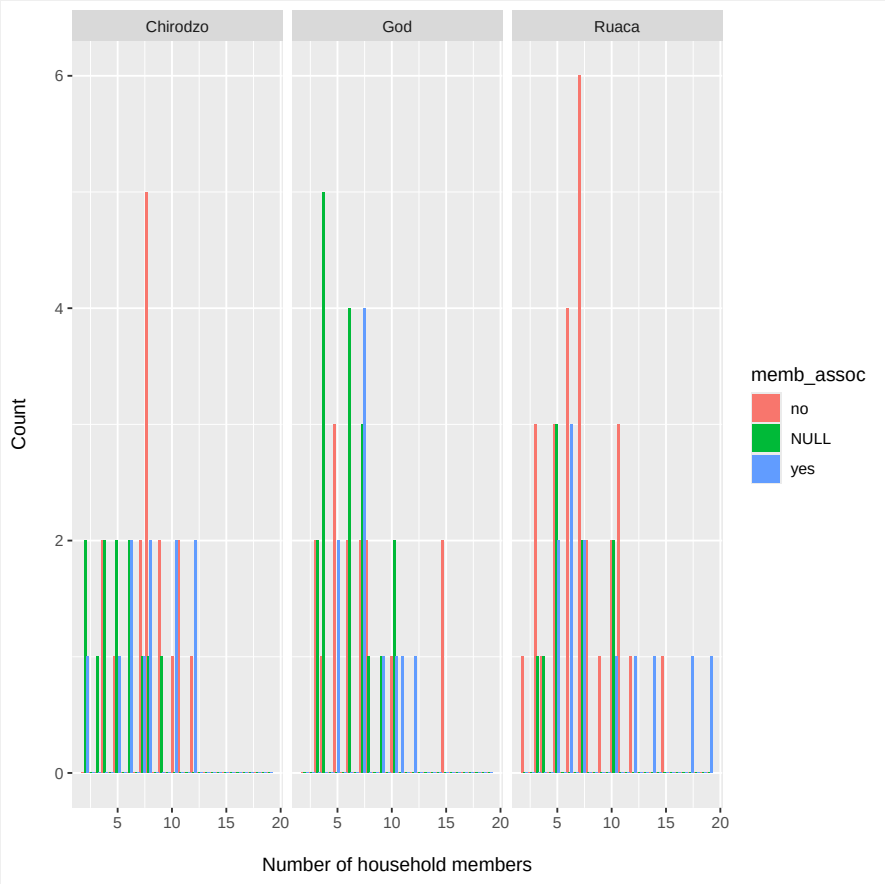
Ex: Histograms with groups

Member association




```
ggplot(SAFI, aes(x = no_membrs, fill = memb_
    assoc)) +
  geom_histogram(position="dodge") +
  labs(x = "\nNumber of household members",
       y = "Count\n")
```

Member association by village



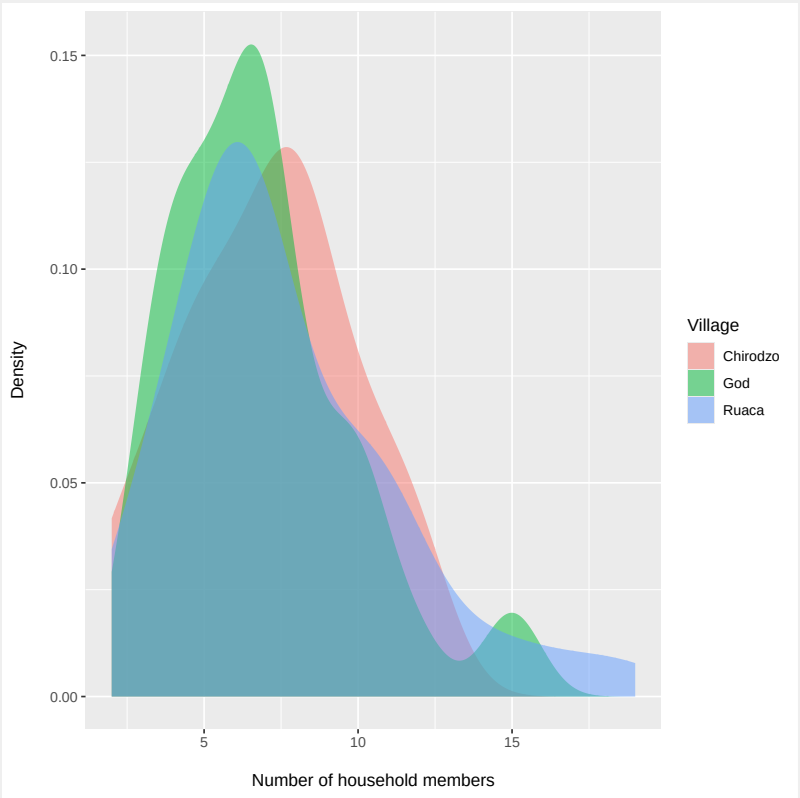

```
ggplot(SAFI, aes(x = no_membrs, fill = memb_
    assoc)) +
  geom_histogram(position="dodge") +
  facet_wrap(vars(village)) +
  labs(x = "\nNumber of household members",
       y = "Count\n")
```

Density and ridge plots

- Density plots provide a smoothed estimate of the distribution; bandwidth (`bw`) controls smoothing
- Aesthetics include `fill` (e.g. #E16462), border `color` (e.g. #9C3836), and line width
- Mapping `fill = drv` and using transparency (`alpha`) overlays group densities; `ggridges` can convert these to ridge plots with `geom_density_ridges()`

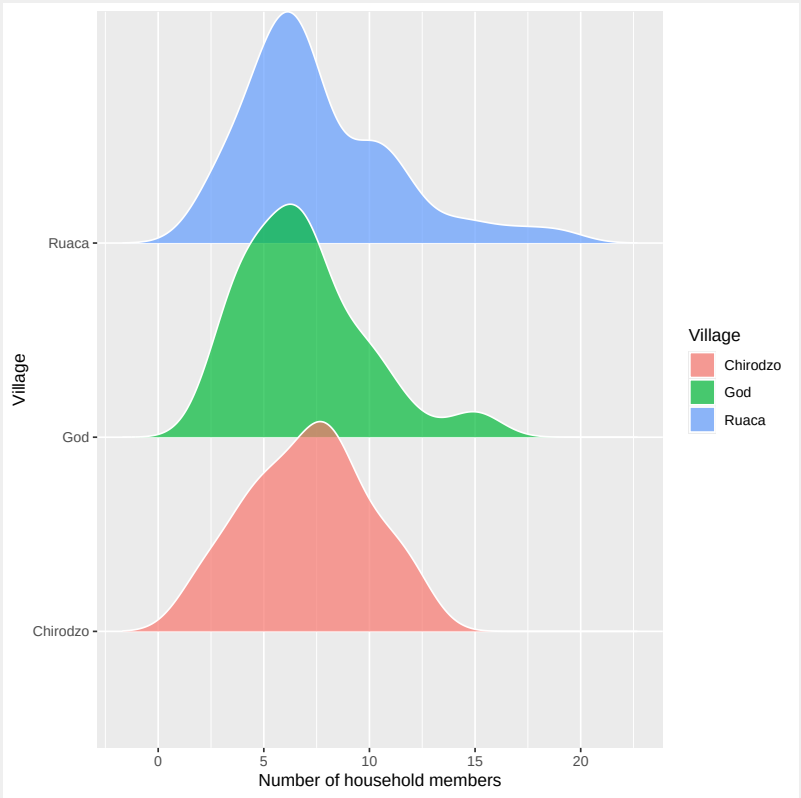
Ex: # of household members by village

Density plots




```
1 ggplot(SAFI, aes(x = no_membrs, fill =
      village)) +
2   geom_density(alpha = 0.5, color = NA,
      adjust = 1) +
3   labs(x = "\nNumber of household members",
4        y = "Density\n",
5        fill = "Village")
```

Ridge plots



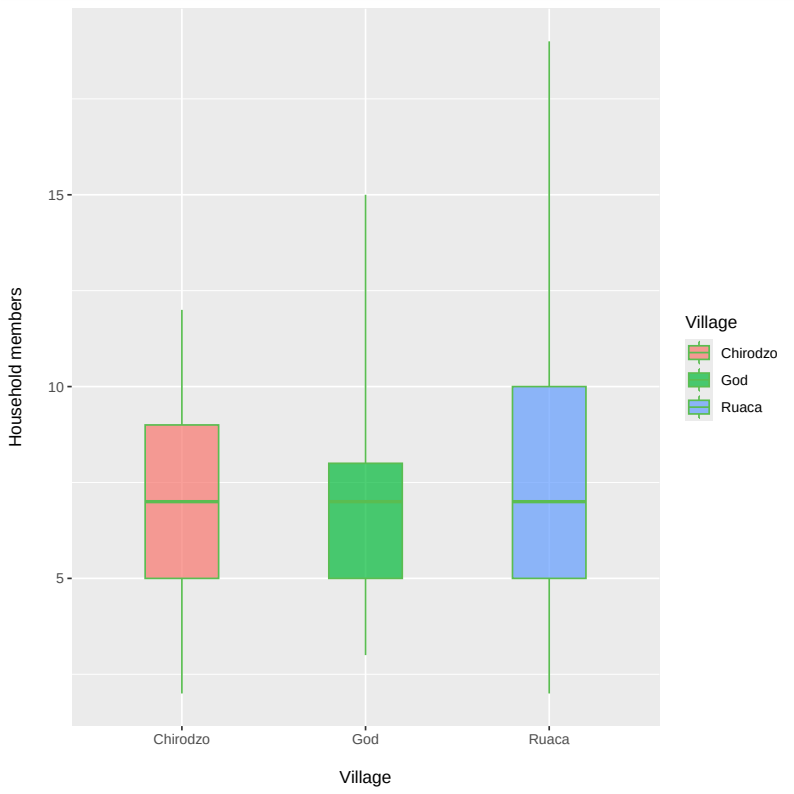

```
1 ggplot(SAFI, aes(x = no_membrs, y = village,
      fill = village)) +
2   geom_density_ridges(alpha = 0.7, color = "
      white", scale = 1.2) +
3   labs(x = "\nNumber of household members",
4        y = "Village\n",
5        fill = "Village")
```

Boxplots, violins, and dots

- Boxplots show quartiles, median, and outliers; `coef` controls the outlier rule (e.g. 5 $\times$ IQR)
- Visual settings include `fill` (e.g. #E6AD3C) and `color` (e.g. #5ABD51)
- Violin plots show mirrored density shapes; adding jittered points (e.g. width 0.1, size 0.5) reveals individual observations by category such as `village`

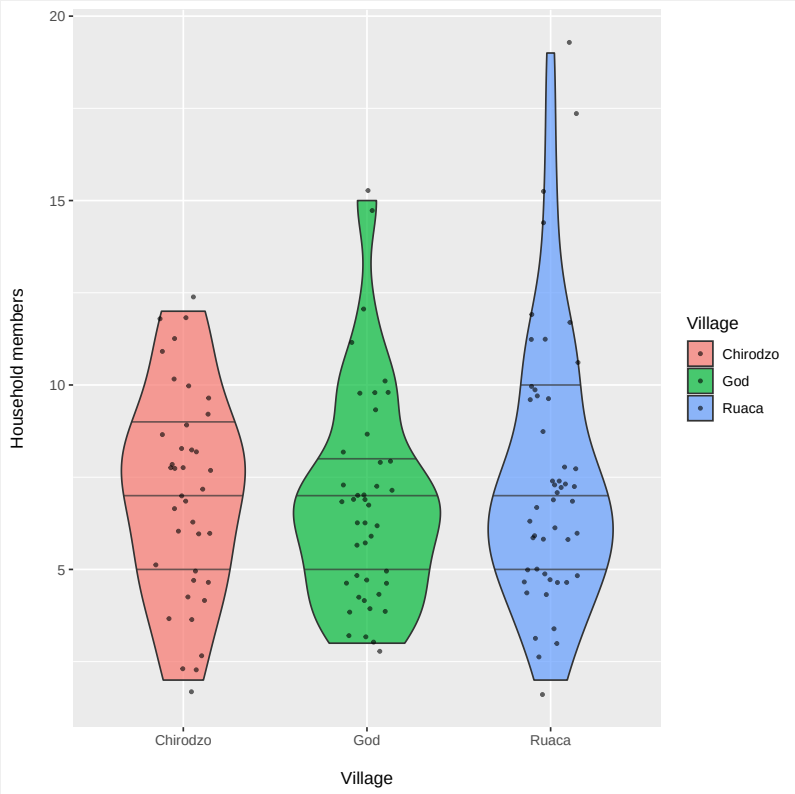
Ex: # of household members by village

Box plot




```
ggplot(SAFI, aes(x = village, y = no_membrs,
       fill = village)) +
  geom_boxplot(alpha = 0.7, color = "#5ABD51
      ", coef = 3, width = 0.4) +
  labs(fill = "Village", x = "\nVillage", y
      = "Household members\n")
```

Violin plot




```
ggplot(SAFI, aes(x = village, y = no_membrs,
       fill = village)) +
  geom_violin(alpha = 0.7, width = 0.8, draw
      _quantiles = c(0.25, 0.5, 0.75)) +
  geom_jitter(width = 0.15, alpha = 0.6,
      size = 0.8) +
  labs(fill = "Village", x = "\nVillage", y
      = "Household members\n")
```

Class business

- Read required (and suggested) online materials
- Fork `GitHub` repository
- Problem set #2 is up on GitHub