Lab 3. Specialized Chart Types

PUBH 6199: Visualizing Data with R, Summer 2025

Xindi (Cindy) Hu, ScD

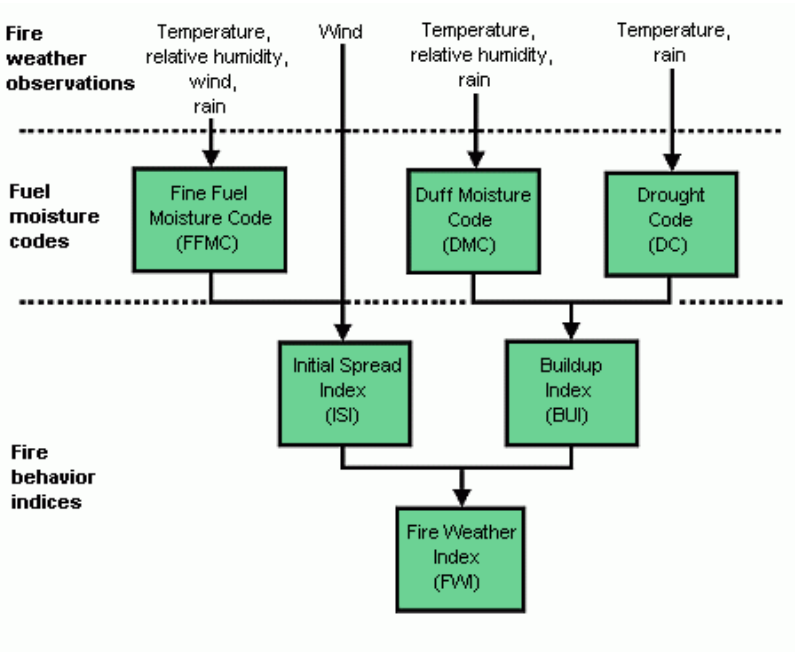
2025-06-05

Outline for today

- **Introducing the datasets**
- Review faceted plots
- Introduce radar charts, Sankey diagrams, and parallel coordinates plot
- Dos and Don’ts when designing visualizations

Forest Fires Dataset

- **Source**: UCI Machine Learning Repository https://archive.ics.uci.edu/ml/datasets/forest+fires
- **Context**: Collected from the Montesinho Natural Park in Portugal, this dataset captures weather conditions and fire activity over 200+ days.
- **Variables**:
  - `month`, `day`: Temporal context
  - `temp`, `RH`, `wind`, `rain`: Daily weather
  - `FFMC`, `DMC`, `DC`, `ISI`: Fire weather indices
  - `area`: Burned area in hectares



Canadian Forest Fire Weather Index (FWI) System

Forest Fires Dataset

```
1 library(tidyverse)
2 forest <- read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/forest-fires/forestfires.csv")
3 glimpse(forest)
```

```
Rows: 517
Columns: 13
$ X     <dbl> 7, 7, 7, 8, 8, 8, 8, 8, 8, 7, 7, 7, 6, 6, 6, 6, 5, 8, 6, 6, 6, 5…
$ Y     <dbl> 5, 4, 4, 6, 6, 6, 6, 6, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4…
$ month <chr> "mar", "oct", "oct", "mar", "mar", "aug", "aug", "aug", "sep", "…
$ day   <chr> "fri", "tue", "sat", "fri", "sun", "sun", "mon", "mon", "tue", "…
$ FFMC  <dbl> 86.2, 90.6, 90.6, 91.7, 89.3, 92.3, 92.3, 91.5, 91.0, 92.5, 92.5…
$ DMC   <dbl> 26.2, 35.4, 43.7, 33.3, 51.3, 85.3, 88.9, 145.4, 129.5, 88.0, 88…
$ DC    <dbl> 94.3, 669.1, 686.9, 77.5, 102.2, 488.0, 495.6, 608.2, 692.6, 698…
$ ISI   <dbl> 5.1, 6.7, 6.7, 9.0, 9.6, 14.7, 8.5, 10.7, 7.0, 7.1, 7.1, 22.6, 0…
$ temp  <dbl> 8.2, 18.0, 14.6, 8.3, 11.4, 22.2, 24.1, 8.0, 13.1, 22.8, 17.8, 1…
$ RH    <dbl> 51, 33, 33, 97, 99, 29, 27, 86, 63, 40, 51, 38, 72, 42, 21, 44, …
$ wind  <dbl> 6.7, 0.9, 1.3, 4.0, 1.8, 5.4, 3.1, 2.2, 5.4, 4.0, 7.2, 4.0, 6.7,…
$ rain  <dbl> 0.0, 0.0, 0.0, 0.2, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,…
$ area  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
```


Adult Income Dataset

- **Source**: UCI Machine Learning Repository
  https://archive.ics.uci.edu/ml/datasets/adult
- **Context**: Extracted from 1994 U.S. Census data to predict whether an individual's income exceeds $50K/year.
- **Variables**:
  - Demographics: `age`, `sex`, `race`, `education`, `marital-status`
  - Employment: `workclass`, `occupation`, `hours-per-week`
  - Target: `income` (>50K or ≤50K)
- **Use in Class**:
  - *Sankey diagram*: Show flows like `education → occupation → income`
  - *Parallel coordinates*: Visualize patterns across age, hours, and income class

Adult Income Dataset

```r
adult <- read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data",
                  col_names = c("age", "workclass", "fnlwgt", "education", "education_num",
                                "marital_status", "occupation", "relationship", "race", "sex",
                                "capital_gain", "capital_loss", "hours_per_week", "native_country", "income"))
glimpse(adult)
```

```
Rows: 32,561
Columns: 15
$ age            <dbl> 39, 50, 38, 53, 28, 37, 49, 52, 31, 42, 37, 30, 23, 32,…
$ workclass      <chr> "State-gov", "Self-emp-not-inc", "Private", "Private", …
$ fnlwgt         <dbl> 77516, 83311, 215646, 234721, 338409, 284582, 160187, 2…
$ education      <chr> "Bachelors", "Bachelors", "HS-grad", "11th", "Bachelors…
$ education_num  <dbl> 13, 13, 9, 7, 13, 14, 5, 9, 14, 13, 10, 13, 13, 12, 11,…
$ marital_status <chr> "Never-married", "Married-civ-spouse", "Divorced", "Mar…
$ occupation     <chr> "Adm-clerical", "Exec-managerial", "Handlers-cleaners",…
$ relationship   <chr> "Not-in-family", "Husband", "Not-in-family", "Husband",…
$ race           <chr> "White", "White", "White", "Black", "Black", "White", "…
$ sex            <chr> "Male", "Male", "Male", "Male", "Female", "Female", "Fe…
$ capital_gain   <dbl> 2174, 0, 0, 0, 0, 0, 0, 0, 14084, 5178, 0, 0, 0, 0, 0, …
$ capital_loss   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
$ hours_per_week <dbl> 40, 13, 40, 40, 40, 40, 16, 45, 50, 40, 80, 40, 30, 50,…
$ native_country <chr> "United-States", "United-States", "United-States", "Uni…
$ income         <chr> "<=50K", "<=50K", "<=50K", "<=50K", "<=50K", "<=50K", "…
```

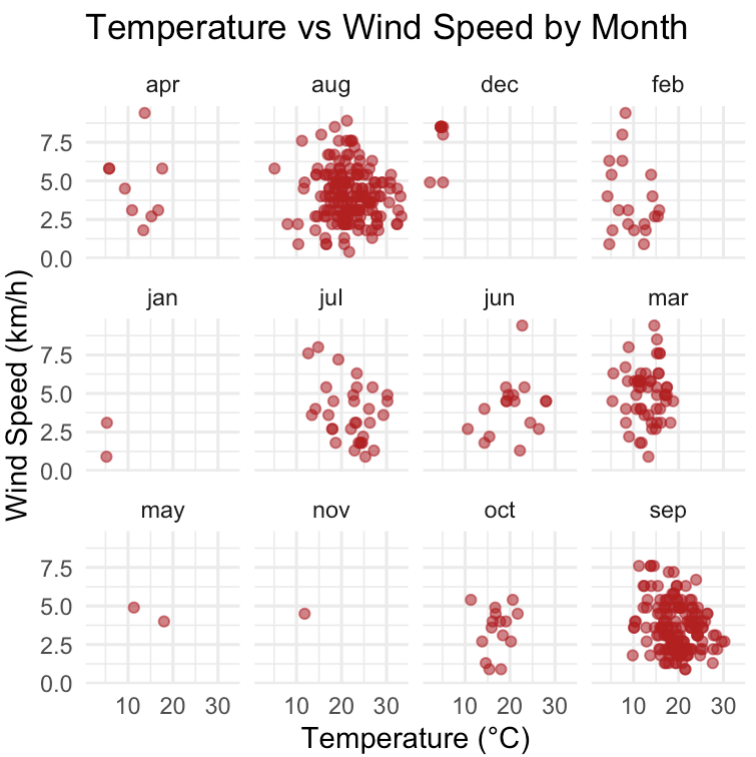

Outline for today

- Introducing the datasets
- **Review faceted plots**
- Introduce radar charts, Sankey diagrams, and parallel coordinates plot
- Dos and Don’ts when designing visualizations

Review: Faceted Plots

> In previous lecture/labs, we learned about faceted plots, which allow us to create multiple subplots based on a categorical variable. This is particularly useful for comparing distributions or trends across different groups.

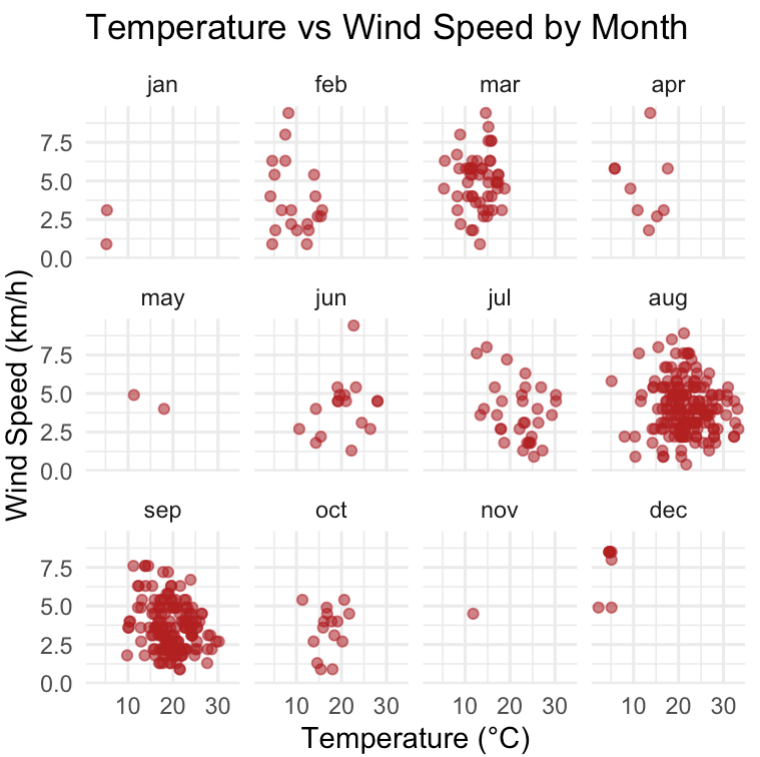
```
# Faceted scatter plot: Temp vs Wind, faceted by mor
ggplot(forest, aes(x = temp, y = wind)) +
  geom_point(alpha = 0.6, color = "firebrick") +
  facet_wrap(~ month, ncol = 4) +
  labs(title = "Temperature vs Wind Speed by Month",
       x = "Temperature (°C)",
       y = "Wind Speed (km/h)") +
  theme_minimal()
```



Review: Faceted Plots

> Optional: convert month to ordered factor for nicer facet layout

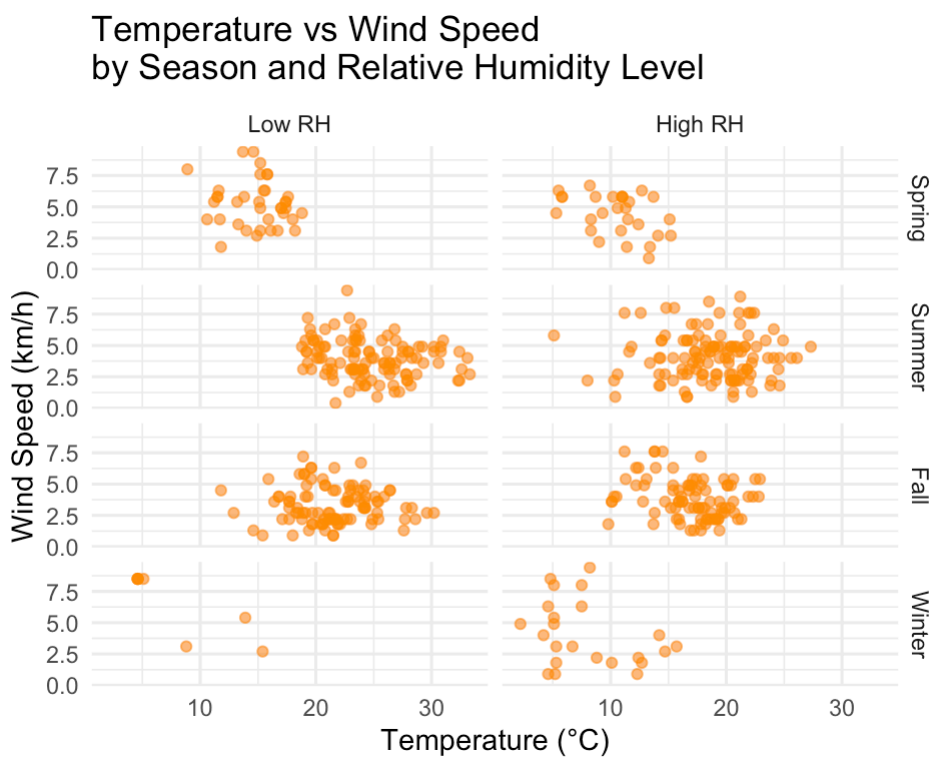
```
forest <- forest %>%
  mutate(month = factor(month, levels = c("jan", "fe
                                          "jul", "au

ggplot(forest, aes(x = temp, y = wind)) +
  geom_point(alpha = 0.6, color = "firebrick") +
  facet_wrap(~ month, ncol = 4) +
  labs(title = "Temperature vs Wind Speed by Month",
       x = "Temperature (°C)",
       y = "Wind Speed (km/h)") +
  theme_minimal()
```



Facet plots by combining two variables

Does dry air (low RH) change the relationship between wind and temperature across seasons? We use `facet_grid()` to create a grid of plots, where each row represents a season and each column represents a relative humidity level.

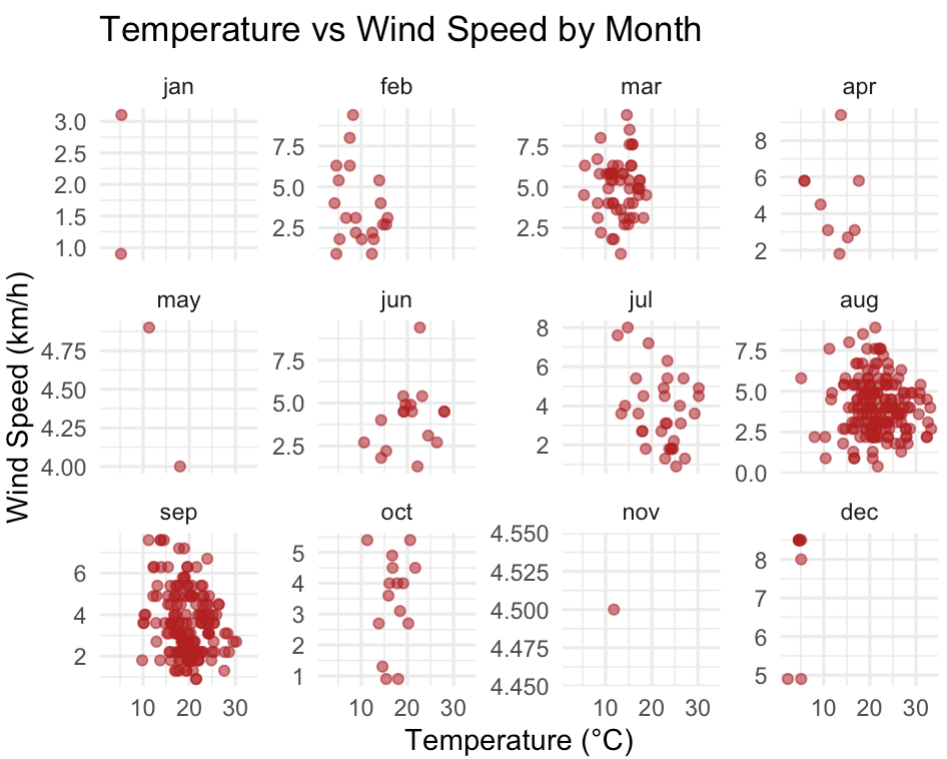
```r
forest <- forest %>%
  mutate(
    month = tolower(month),
    season = case_when(
      month %in% c("dec", "jan", "feb") ~ "Winter",
      month %in% c("mar", "apr", "may") ~ "Spring",
      month %in% c("jun", "jul", "aug") ~ "Summer",
      month %in% c("sep", "oct", "nov") ~ "Fall",
      TRUE ~ "Unknown"
    ),
    RH_level = if_else(RH >= median(RH, na.rm = TRUE), "High RH",
    season = factor(season, levels = c("Spring", "Summer", "Fall"
    RH_level = factor(RH_level, levels = c("Low RH", "High RH"))
  )

# Faceted scatter plot: temp vs wind by season and RH level
ggplot(forest, aes(x = temp, y = wind)) +
  geom_point(alpha = 0.6, color = "darkorange") +
  facet_grid(rows = vars(season), cols = vars(RH_level)) +
  labs(
    title = "Temperature vs Wind Speed\nby Season and Relative Hu
    x = "Temperature (°C)",
    y = "Wind Speed (km/h)"
  ) +
  theme_minimal(base_size = 11)
```



Use fixed vs free scales

Using fixed scales can help in comparing values across different facets, while free scales allow each facet to have its own scale, which can be useful when the data varies widely between groups.

```
forest <- forest %>%
  mutate(month = factor(month, levels = c("jan", "fe
                                          "jul", "au

ggplot(forest, aes(x = temp, y = wind)) +
  geom_point(alpha = 0.6, color = "firebrick") +
  facet_wrap(~ month, ncol = 4, scales = "free_y") +
  labs(title = "Temperature vs Wind Speed by Month",
       x = "Temperature (°C)",
       y = "Wind Speed (km/h)") +
  theme_minimal()
```



🚲 Your turn, get on the bike!

~ Head over to lab3 notebook! ~

Outline for today

- Introducing the datasets
- Review faceted plots
- **Introduce radar charts, Sankey diagrams, and parallel coordinates plot**
- Dos and Don’ts when designing visualizations

What is a Radar Chart?

- A radar chart (or spider chart) displays multivariate data on axes starting from the same point.
- Good for comparing profiles across a few categories or groups.
- Useful when variables are on the **same scale** or **normalized**.

Why Use a Radar Chart to explore the `forest` data?

- Compare **weather profiles** for different fire conditions.
- Visualize **relative intensity** of multiple variables at once (e.g., `temp`, `wind`, `RH`).
- Spot patterns across fire size categories.

Step 1: Load and prepare the data

> Forest Fires Dataset This dataset contains 517 observations, each representing a single day of recorded weather conditions and fire activity in the Montesinho Natural Park in Portugal. Each observation includes meteorological variables such as temperature, relative humidity (RH), wind speed, and rainfall, as well as fire weather indices (e.g., FFMC, DMC) and the area burned in hectares. The data spans multiple months and is commonly used to study environmental drivers of wildfire risk.

```
Rows: 517
Columns: 14
$ X          <dbl> 7, 7, 7, 8, 8, 8, 8, 8, 8, 7, 7, 7, 6, 6, 6, 6, 5, 8, 6, 6,…
$ Y          <dbl> 5, 4, 4, 6, 6, 6, 6, 6, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 4,…
$ month      <chr> "mar", "oct", "oct", "mar", "mar", "aug", "aug", "aug", "se…
$ day        <chr> "fri", "tue", "sat", "fri", "sun", "sun", "mon", "mon", "tu…
$ FFMC       <dbl> 86.2, 90.6, 90.6, 91.7, 89.3, 92.3, 92.3, 91.5, 91.0, 92.5,…
$ DMC        <dbl> 26.2, 35.4, 43.7, 33.3, 51.3, 85.3, 88.9, 145.4, 129.5, 88.…
$ DC         <dbl> 94.3, 669.1, 686.9, 77.5, 102.2, 488.0, 495.6, 608.2, 692.6…
$ ISI        <dbl> 5.1, 6.7, 6.7, 9.0, 9.6, 14.7, 8.5, 10.7, 7.0, 7.1, 7.1, 22…
$ temp       <dbl> 8.2, 18.0, 14.6, 8.3, 11.4, 22.2, 24.1, 8.0, 13.1, 22.8, 17…
$ RH         <dbl> 51, 33, 33, 97, 99, 29, 27, 86, 63, 40, 51, 38, 72, 42, 21,…
$ wind       <dbl> 6.7, 0.9, 1.3, 4.0, 1.8, 5.4, 3.1, 2.2, 5.4, 4.0, 7.2, 4.0,…
$ rain       <dbl> 0.0, 0.0, 0.0, 0.2, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,…
$ area       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
$ area_group <chr> "No fire", "No fire", "No fire", "No fire", "No fire", "No
```

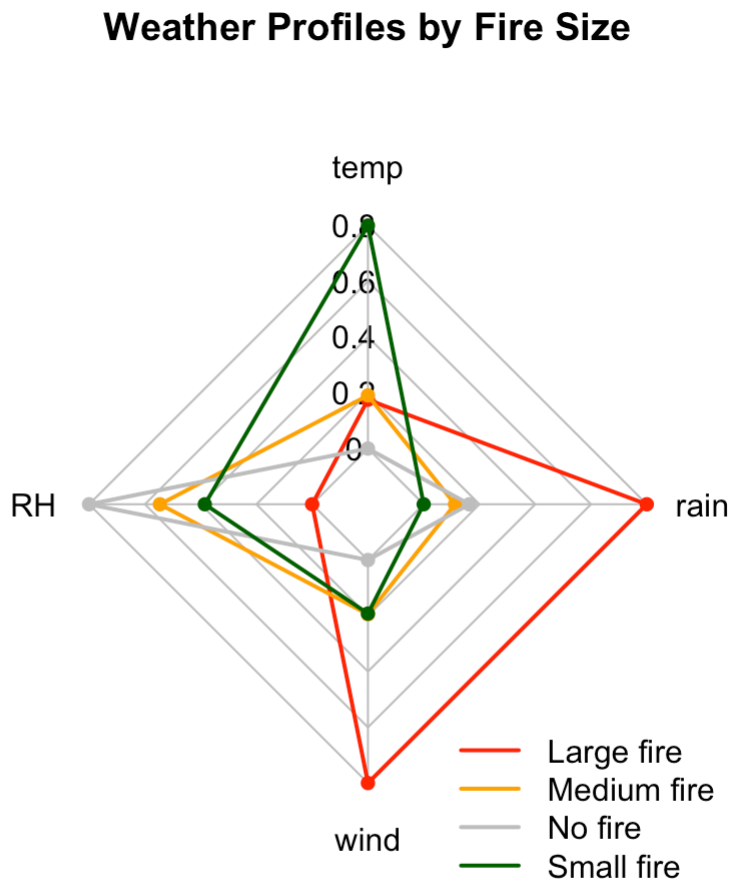

Step 2: Summarize and normalize variables

```r
library(scales)
library(fmsb)
# Select and normalize variables
radar_data <- forest %>%
    group_by(area_group) %>%
    summarise(
        temp = mean(temp),
        RH = mean(RH),
        wind = mean(wind),
        rain = mean(rain)
    ) %>%
    # normalize to [0, 1]
    mutate(across(where(is.numeric), rescale))

# Convert to fmsb format
# fmsb expects first two rows to be max and min
radar_df <- radar_data %>%
    tibble::column_to_rownames("area_group") %>%
    as.data.frame()

# fmsb expects first two rows: max then min
max_row <- rep(1, ncol(radar_df))
min_row <- rep(0, ncol(radar_df))
radar_df <- rbind(max = max_row, min = min_row, rada

```

	temp	RH	wind	rain
max	1.0000000	1.0000000	1.0000000	1.0000000
min	0.0000000	0.0000000	0.0000000	0.0000000
Large fire	0.2189738	0.0000000	1.0000000	1.0000000
Medium fire	0.2379999	0.6826801	0.2434875	0.1404139
No fire	0.0000000	1.0000000	0.0000000	0.2043269
Small fire	1.0000000	0.4812876	0.2399755	0.0000000

Step 3: Plot the Radar Chart

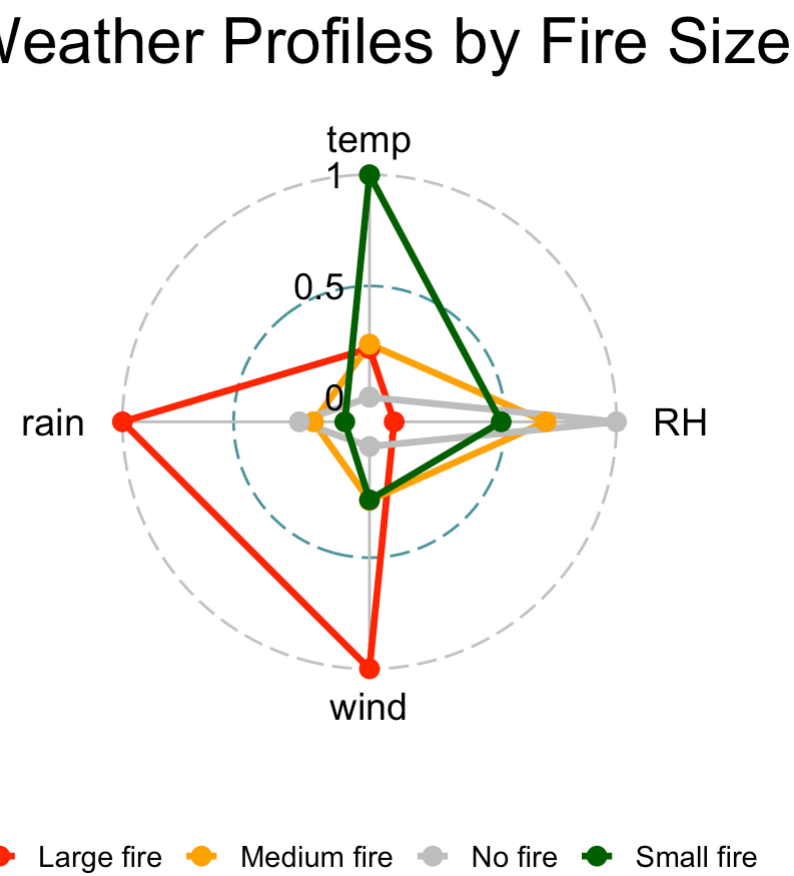
```r
radarchart(radar_df,
           axistype = 1,
           pcol = c("red", "orange", "grey", "darkgr
           plwd = 2,
           plty = 1,
           cglcol = "grey",
           cglty = 1,
           axislabcol = "black",
           caxislabels = seq(0, 1, 0.2),
           title = "Weather Profiles by Fire Size")

legend("bottomright",
       legend = rownames(radar_df[-c(1,2),]),
       col = c("red", "orange", "grey", "darkgreen")
       lty = 1, lwd = 2, bty = "n")
```



Make radar chart with {ggradar}

Once you have `radar_data` prepared, you can use the `ggradar` package to create a more polished radar chart.

```
remotes::install_github("ricardo-bion/ggradar")
library(ggradar)
ggradar(radar_data,
        grid.min = 0, grid.mid = 0.5, grid.max = 1,
        group.line.width = 1.2,
        group.colours = c('red', 'orange', 'grey', '
        group.point.size = 3,
        font.radar = "Arial",
        values.radar = c("0", "0.5", "1"),
        background.circle.colour = "transparent",
        legend.position = "bottom") +
    ggtitle("Weather Profiles by Fire Size") +
    theme(
        legend.text = element_text(size = 11),
        legend.key.width = unit(0.5, "cm")
    )
```



What is a Sankey diagram?

- A **Sankey diagram** visualizes flows between categories.
- The width of each flow is proportional to the count or magnitude.
- Great for showing step-wise processes, transitions, or categorical relationships.

Why Use a Sankey Diagram to explore the `adult` data?

- Visualize **income distribution** across multiple categorical variables.
- Visualize **distributional flows** like:
  - Education → Occupation → Income
  - Gender → Workclass → Income

Step 1: Load and Prepare the Data

> Adult Income Dataset This dataset contains 32,561 observations, each representing a single adult individual from the 1994 U.S. Census database. Each observation includes demographic and employment-related variables such as age, sex, education, occupation, and work hours, as well as an income label indicating whether the person earns more or less than $50,000 per year. The dataset is commonly used to study social and economic patterns, and to explore factors associated with income inequality.

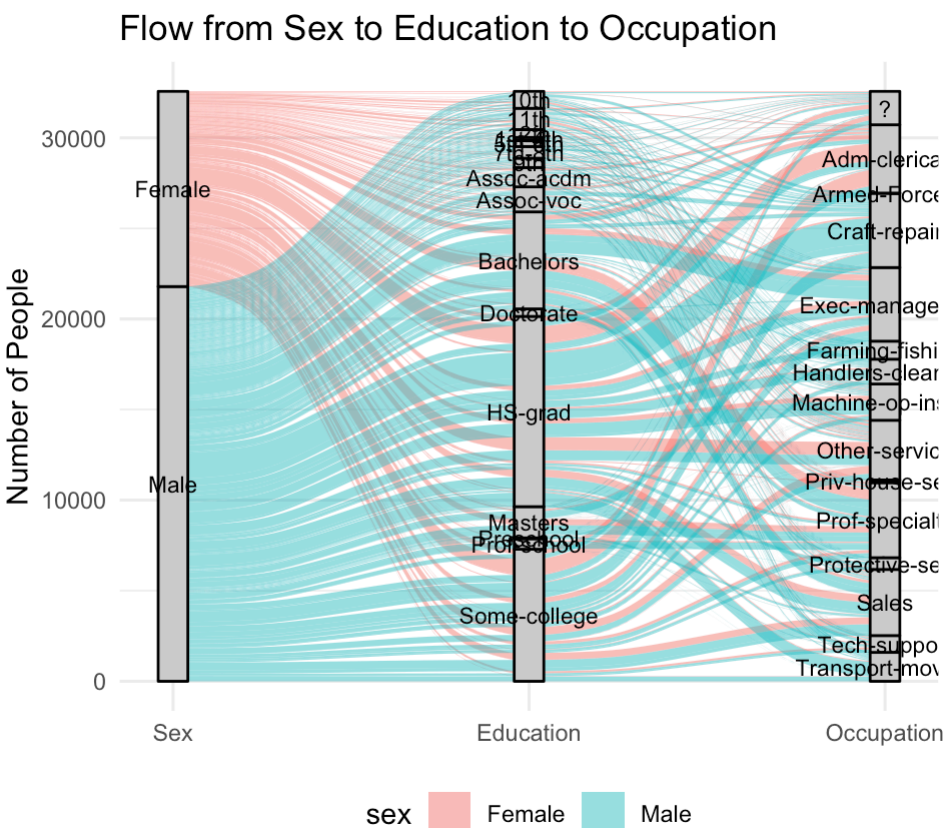
```r
# Keep only relevant columns
adult_sankey <- adult %>%
  select(sex, education, occupation, income) %>%
  filter(complete.cases(.))

flow_data <- adult_sankey %>%
  count(sex, education, occupation, income) %>%
  ungroup()

glimpse(flow_data)
```

```
Rows: 623
Columns: 5
$ sex        <chr> "Female", "Female", "Female", "Female", "Female", "Female",…
$ education  <chr> "10th", "10th", "10th", "10th", "10th", "10th", "10th", "10…
$ occupation <chr> "?", "Adm-clerical", "Craft-repair", "Craft-repair", "Exec-…
```

Step 2: Summarize and plot with {ggalluvial}

> 3-axis Sankey diagrams can quickly become cluttered when categorical variables have too many levels

```
library(ggalluvial)
library(ggplot2)

# Basic 3-axis Sankey
ggplot(flow_data,
       aes(axis1 = sex, axis2 = education, axis3 = c
    geom_alluvium(aes(fill = sex), width = 1/12) +
    geom_stratum(width = 1/12, fill = "grey80", colo
    geom_text(stat = "stratum", aes(label = after_st
    scale_x_discrete(limits = c("Sex", "Education",
    labs(title = "Flow from Sex to Education to Occu
         y = "Number of People") +
    theme_minimal()+
    theme(legend.position = "bottom")
```



Step 3: regroup categorical data

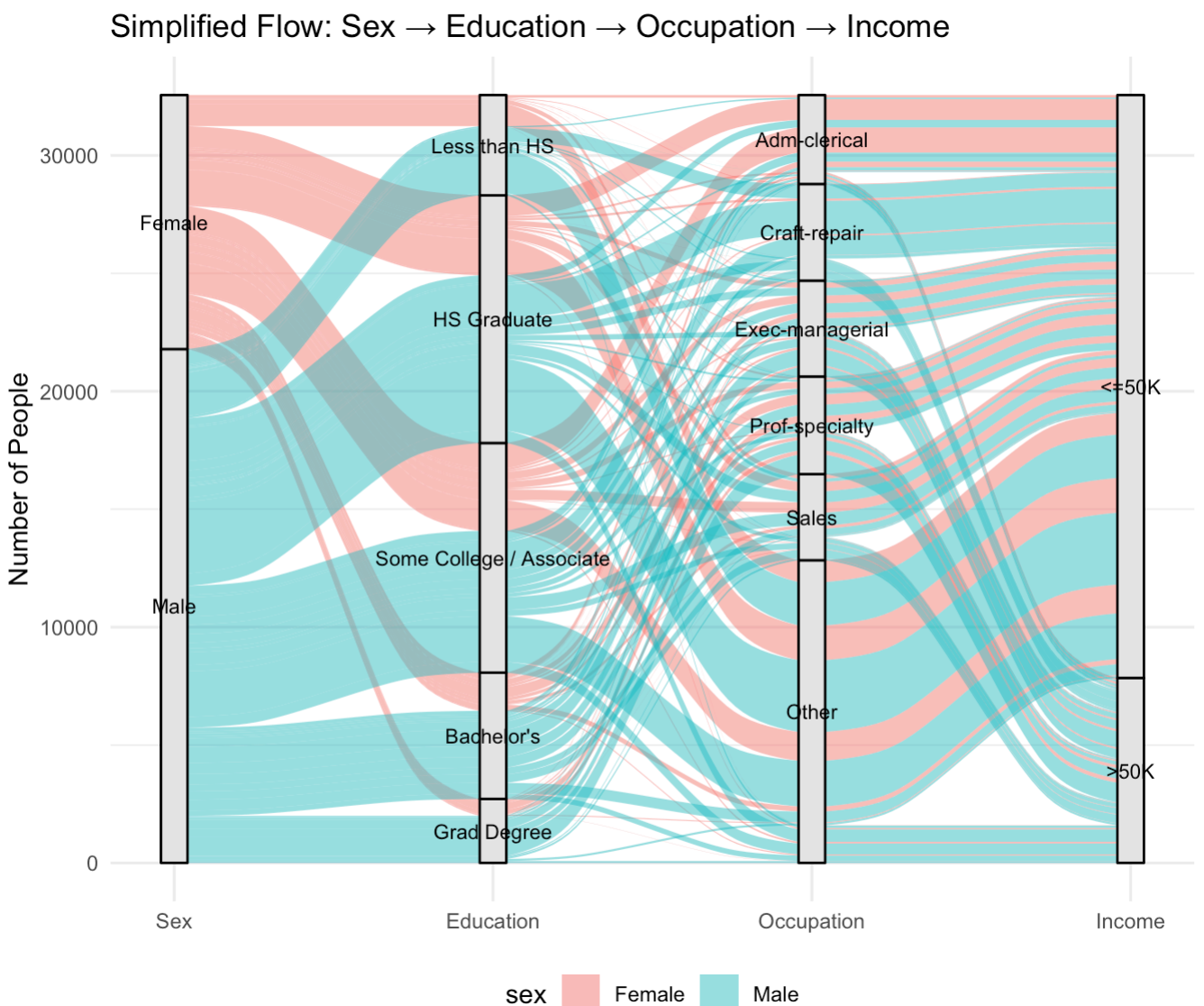
> Using tools like `forcats::fct_lump()` and binning to reduce the number of flows and make the diagram more readable.

```
adult_sankey <- adult_sankey %>%
    # keep the five most common occupations
    mutate(occupation = fct_lump(occupation, n = 5)) %>%
    # group education levels
    mutate(education_group = case_when(
    education %in% c("Preschool", "1st-4th", "5th-6th", "7th-8th", "9th", "10th", "11th", "12th") ~ "Less th
    education == "HS-grad" ~ "HS Graduate",
    education %in% c("Some-college", "Assoc-acdm", "Assoc-voc") ~ "Some College / Associate",
    education %in% c("Bachelors") ~ "Bachelor's",
    education %in% c("Masters", "Doctorate", "Prof-school") ~ "Grad Degree",
    TRUE ~ "Other"
  )) %>%
   mutate(education_group = factor(education_group, levels = c("Less than HS", "HS Graduate", "Some College /
flow_data <- adult_sankey %>%
   count(sex, education_group, occupation, income)
glimpse(flow_data)
```

```
Rows: 120
Columns: 5
$ sex             <chr> "Female", "Female", "Female", "Female", "Female", "Fem…
$ education_group <fct> Less than HS, Less than HS, Less than HS, Less than HS…
$ occupation      <fct> Adm-clerical, Adm-clerical, Craft-repair, Craft-repair…
$ income          <chr> "<=50K", ">50K", "<=50K", ">50K", "<=50K", ">50K", "<=…
$ n               <int> 108, 1, 29, 5, 25, 3, 20, 1, 191, 1, 925, 12, 883, 85,…
```

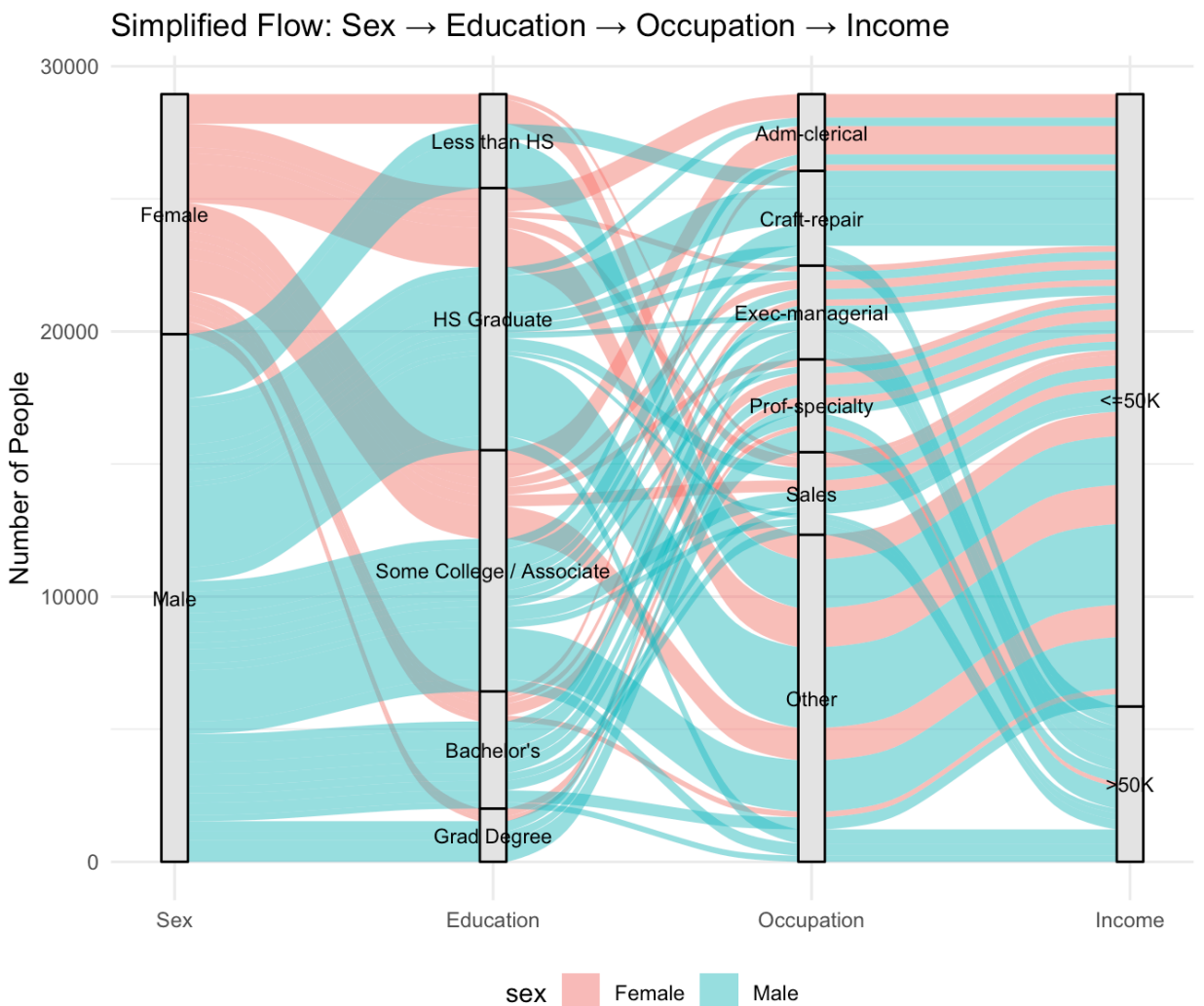

Step 4: replot Sankey diagram

```
# Plot Sankey with 4 axes
ggplot(flow_data,
       aes(axis1 = sex, axis2
    geom_alluvium(aes(fill =
    geom_stratum(width = 1/12
    geom_text(stat = "stratum
    scale_x_discrete(limits =
    labs(title = "Simplified
         y = "Number of Peopl
    theme_minimal() +
    theme(legend.position = "
```



Step 5: filter out weak links

```
flow_data_filtered <- flow_data %>%
  filter(n >= 0.005 * sum(n))  # Keep flows >= 0.5% of total
```



What is a Parallel Coordinates Plot (PCP)?

- A visualization for **multivariate data**, especially numeric.
- Each line represents:
  - an **individual** (for raw data), or
  - a **group profile** (for summarized counts).
- Useful for spotting clusters, trends, and differences between groups.

Variation 1: Individual-level PCP

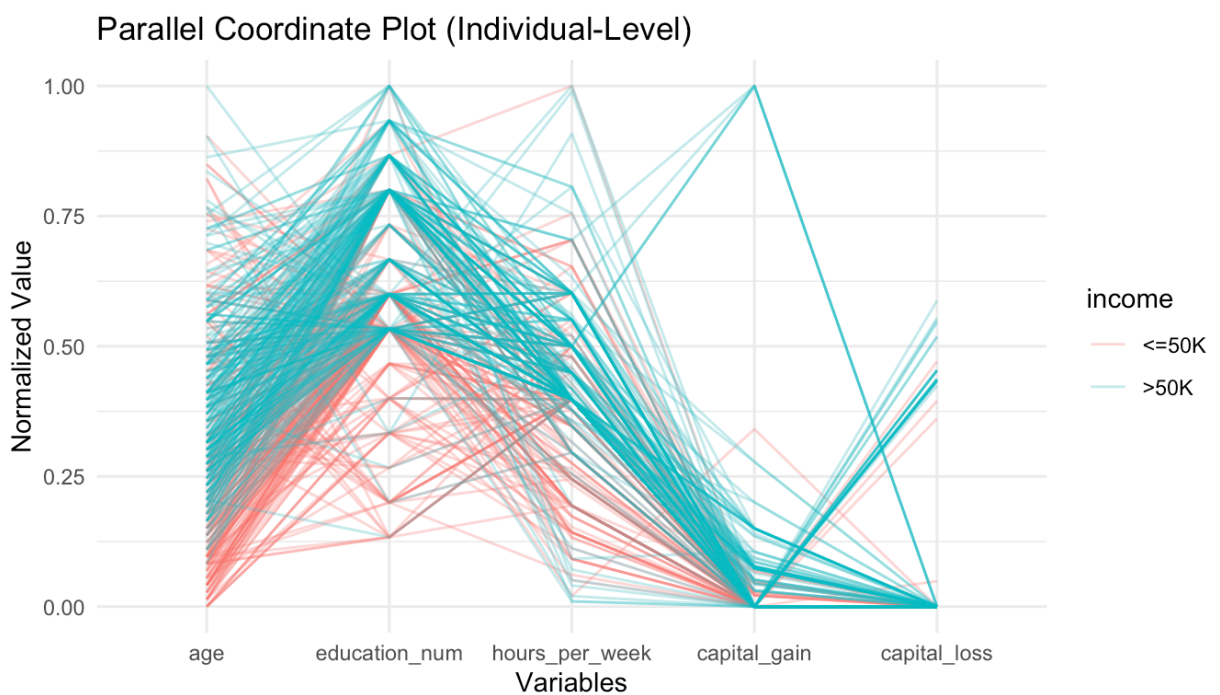
> Scaling variable is a crucial step to build a proper parallel coordinates chart. It transforms the raw data to a new scale that is common with other variables, and thus allow to compare them.

```r
library(GGally)
# Prepare numeric individual-level data
adult_pc <- adult %>%
  select(age, education_num, hours_per_week, capital_gain, capital_loss, income) %>%
  filter(complete.cases(.)) %>%
  mutate(across(where(is.numeric), rescale))  # normalize for comparability

# Sample for readability
set.seed(123)
adult_sample <- adult_pc %>%
  group_by(income) %>%
  sample_n(250)
glimpse(adult_sample)
```

```
Rows: 500
Columns: 6
Groups: income [2]
$ age            <dbl> 0.61643836, 0.38356164, 0.90410959, 0.24657534, 0.47945…
$ education_num  <dbl> 0.5333333, 0.6000000, 0.5333333, 0.6666667, 0.6000000, …
$ hours_per_week <dbl> 0.3979592, 0.3979592, 0.1938776, 0.2959184, 0.3979592, …
$ capital_gain   <dbl> 0.00000000, 0.00000000, 0.00000000, 0.00000000, 0.00000…
$ capital_loss   <dbl> 0.0000000, 0.0000000, 0.0000000, 0.0000000, 0.0000000, …
$ income         <chr> "<=50K", "<=50K", "<=50K", "<=50K", "<=50K", "<=50K", "…
```


Variation 1: Individual-level PCP

```r
# Individual-level parallel coordinate plot
GGally::ggparcoord(
  data = adult_sample,
  columns = 1:5,
  groupColumn = 6,
  scale = "globalminmax",
  showPoints = FALSE,
  alphaLines = 0.3,
  title = "Parallel Coordinate Plot (Individual-Level)") +
  theme_minimal() +
  labs(x = "Variables", y = "Normalized Value")
```



Variation 2: Group-level PCP

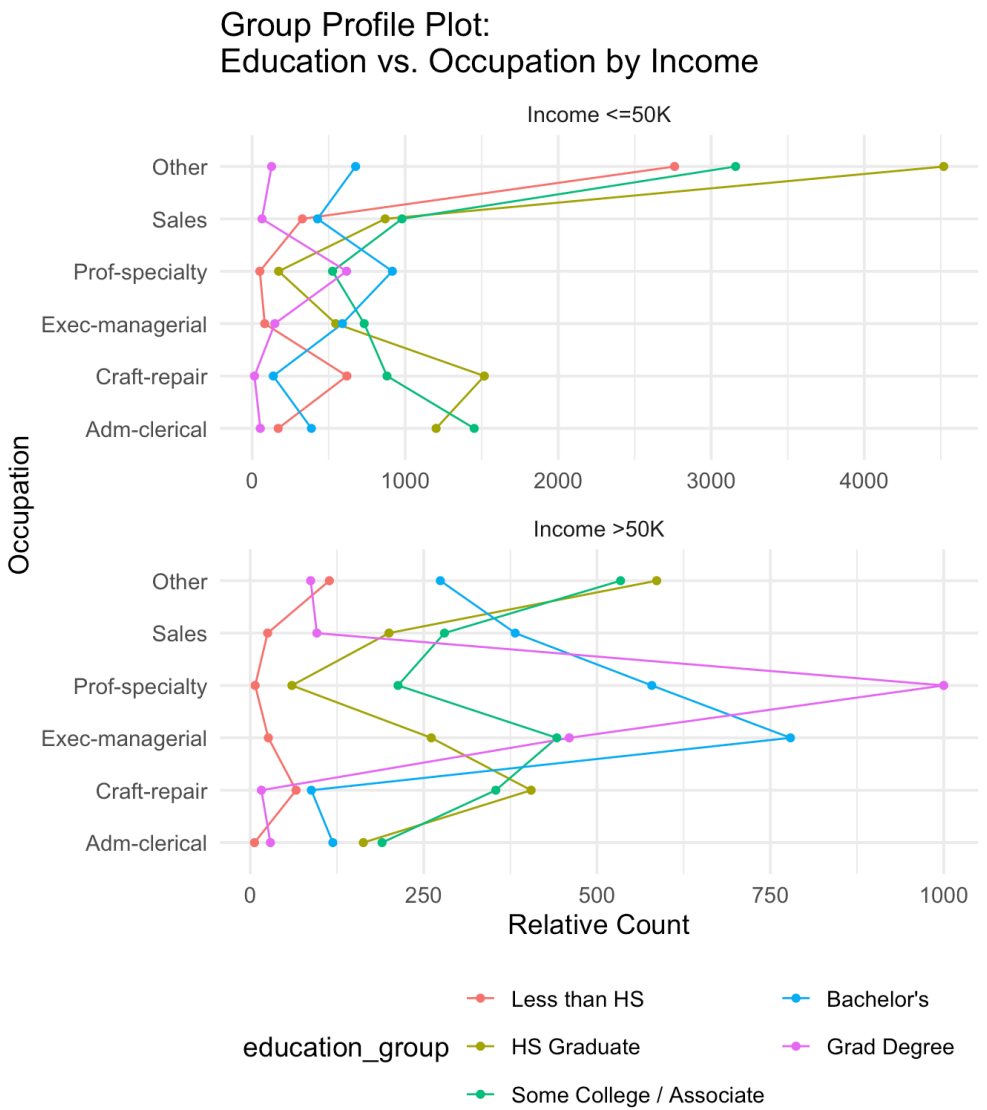
> In this variation, we summarize the data to create group profiles, which allows us to visualize the average characteristics of each income group across multiple variables.

```
# Count and reshape for parallel coordinates
edu_occ_income <- adult_sankey %>%
  count(income, education_group, occupation) %>%
  pivot_wider(names_from = occupation, values_from = n, values_fill = 0)%>%
  mutate(income = factor(income, labels = c("<=50K", ">50K")))
glimpse(edu_occ_income)
```

```
Rows: 10
Columns: 8
$ income            <fct> <=50K, <=50K, <=50K, <=50K, <=50K, >50K, >50K, >50K,…
$ education_group   <fct> Less than HS, HS Graduate, Some College / Associate,…
$ `Adm-clerical`    <int> 170, 1202, 1451, 387, 53, 6, 163, 190, 119, 29
$ `Craft-repair`    <int> 619, 1517, 881, 138, 15, 66, 405, 354, 88, 16
$ `Exec-managerial` <int> 82, 546, 732, 590, 148, 26, 261, 442, 779, 460
$ `Prof-specialty`  <int> 50, 173, 525, 916, 617, 7, 60, 213, 579, 1000
$ Sales             <int> 328, 869, 979, 427, 64, 25, 200, 280, 382, 96
$ Other             <int> 2760, 4519, 3159, 676, 127, 114, 586, 534, 274, 87
```


Variation 2: Group-level PCP

```
# Group-level profile parallel coordinate plot
income_labels <- c(`1` = "Income <=50K", `2` = "Inco
GGally::ggparcoord(
    data = edu_occ_income,
    columns = 3:ncol(edu_occ_income), # occupation c
    groupColumn = 2, #education group
    scale = "globalminmax",
    showPoints = TRUE,
    title = "Group Profile Plot:\nEducation vs. Occu
    theme_minimal(base_size = 14) +
    facet_wrap(. ~ income, ncol = 1, labeller = as_l
    labs(x = "Occupation", y = "Relative Count") +
    coord_flip() +
    theme(legend.position = "bottom")+
    guides(color = guide_legend(nrow = 3))
```



Comparing two variations

Individual-level parallel coordinates plot:

- Each line = one person
- Color by group variable, `income`
- Use this to explore variable interactions and clusters

Group-level parallel coordinates plot:

- Each line = one education group
- Use this to compare how different `education` groups are distributed across `occupations`
- `income` in represented by facets

Discussion:

When is it better to use individual-level data vs. group-level profiles in a parallel coordinate plot?

🛠 Your turn to make some graphs!

~ Head over to lab3 notebook! ~

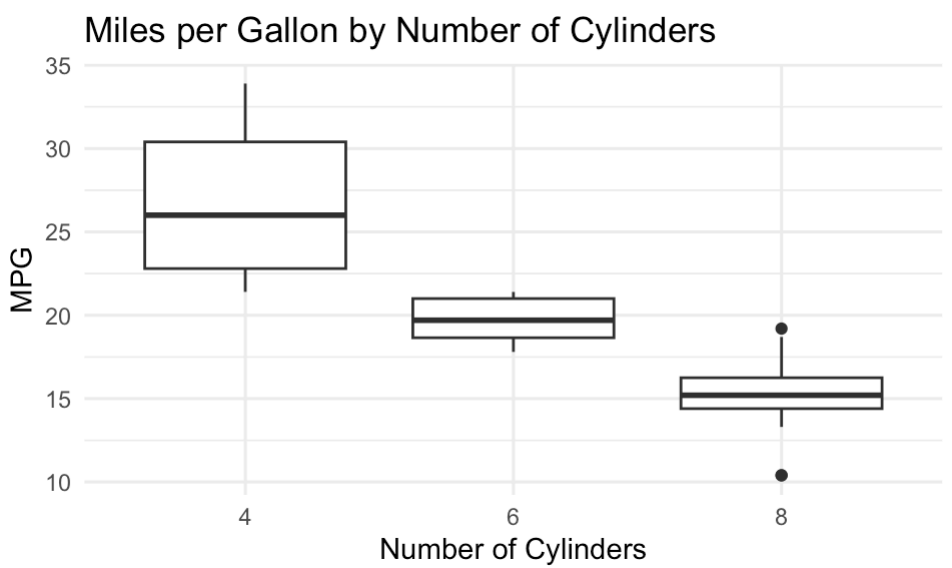
Outline for today

- Introducing the datasets
- Review faceted plots
- Introduce radar charts, Sankey diagrams, and parallel coordinates plot
- **Dos and Don’ts when designing visualizations**

DO: Clear, informative titles and labels

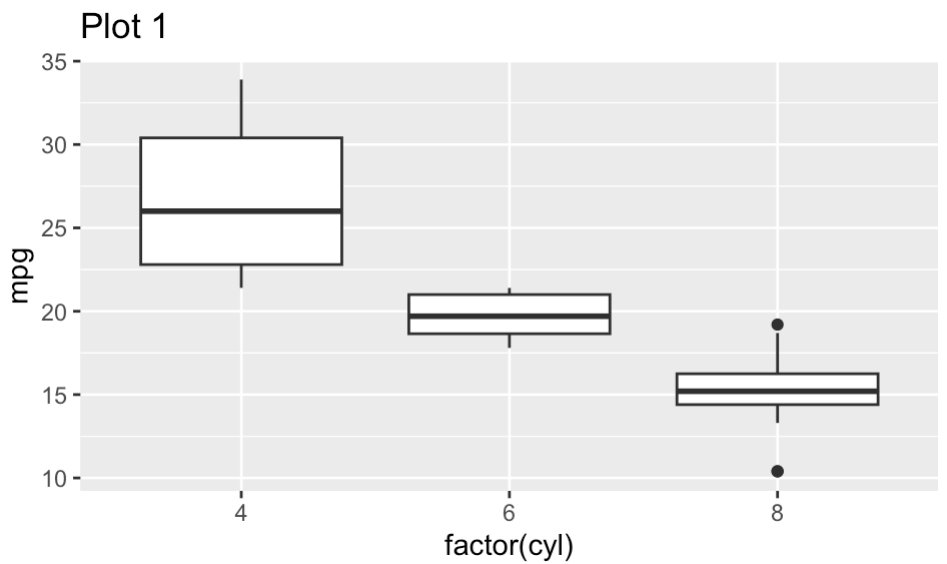
✅ DO

```
ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
  geom_boxplot() +
  labs(
    title = "Miles per Gallon by Number of Cylinders",
    x = "Number of Cylinders",
    y = "MPG"
  ) +
  theme_minimal()
```



❌ Don't

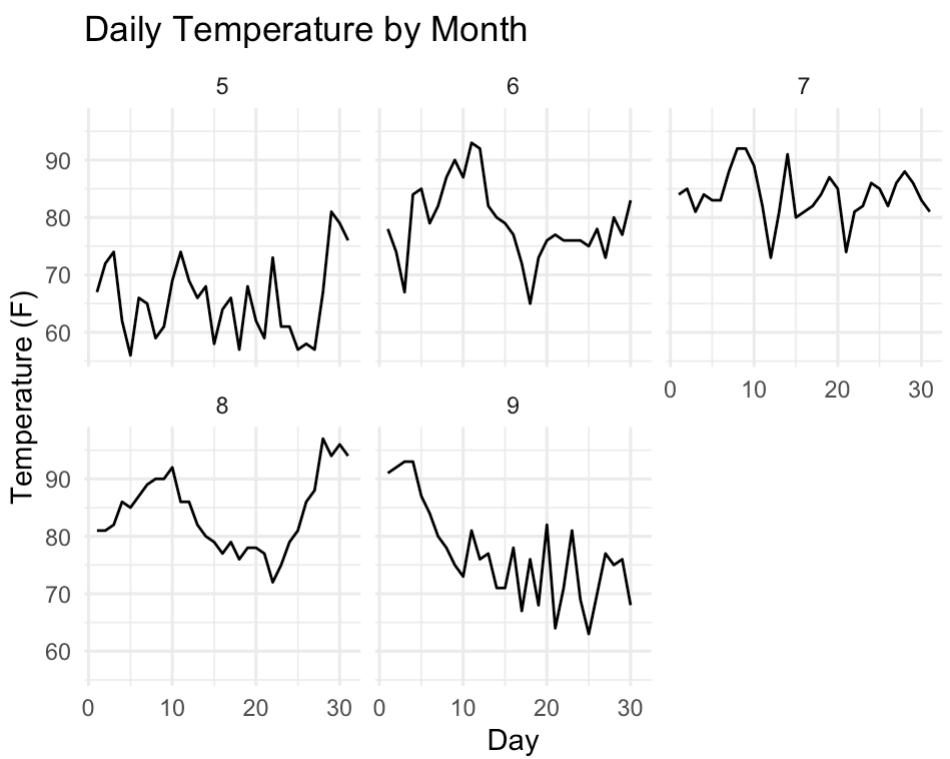
```
ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
  geom_boxplot() +
  ggtitle("Plot 1") +  # Uninformative
  theme_gray()         # Cluttered default theme
```



DO: Choose the right chart type

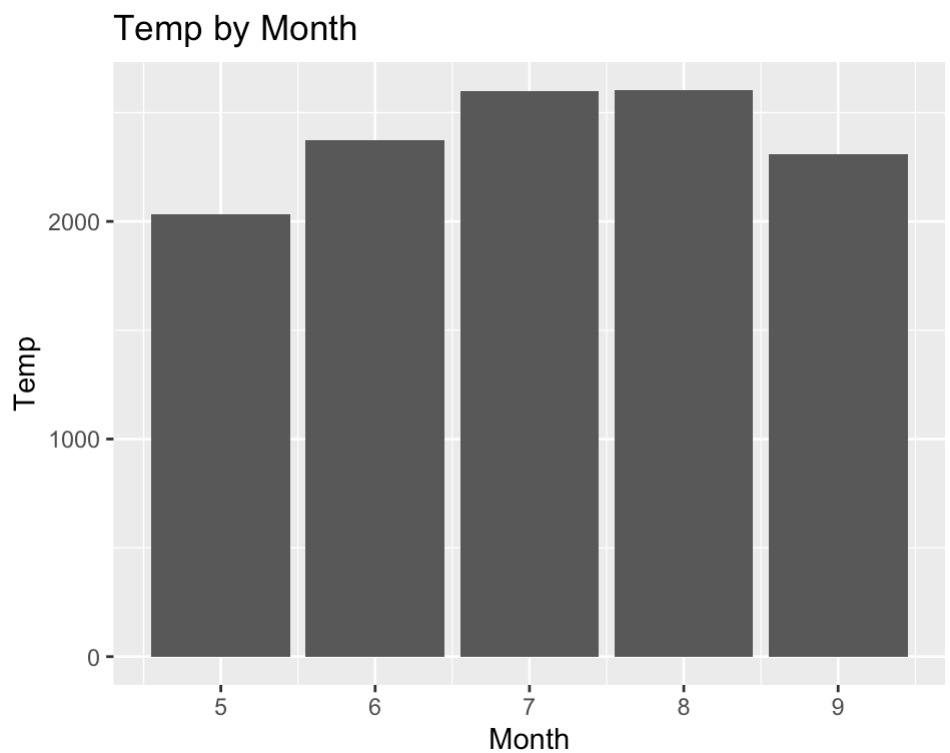
✅ DO

```
ggplot(airquality, aes(x = Day, y = Temp)) +
  geom_line() +
  facet_wrap(~ Month) +
  labs(title = "Daily Temperature by Month", x = "Da
  theme_minimal()
```



❌ Don't

```
ggplot(airquality, aes(x = Month, y = Temp)) +
  geom_bar(stat = "identity") +
  ggtitle("Temp by Month")  # Misleading chart type
```

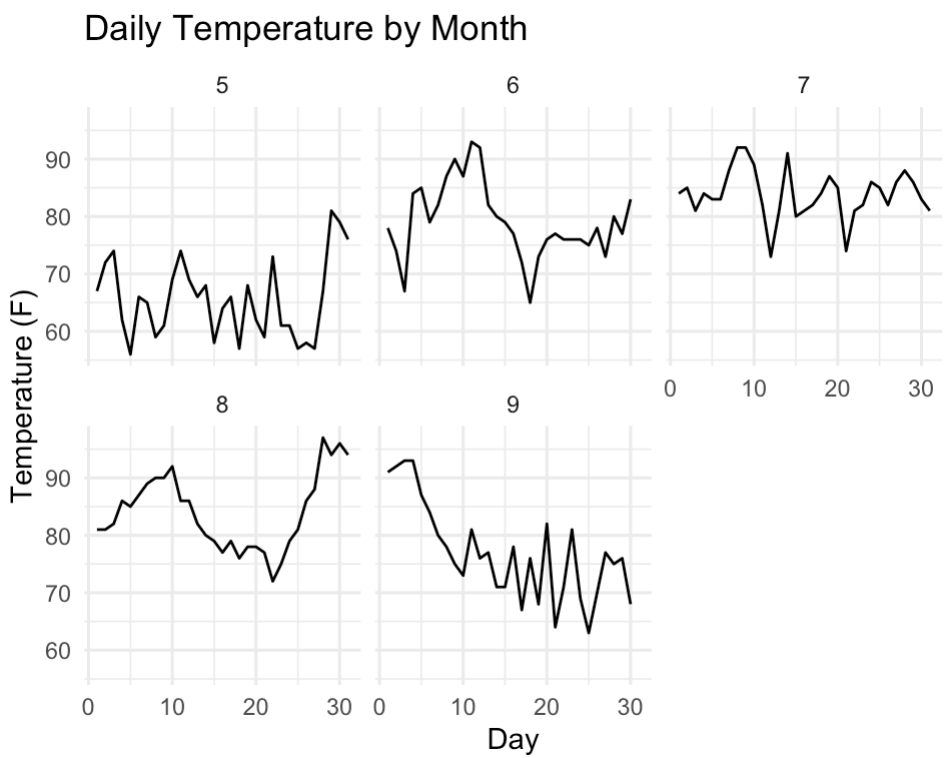


What is this chart showing? Average temperature or total temperature by month?

DO: Choose the right chart type

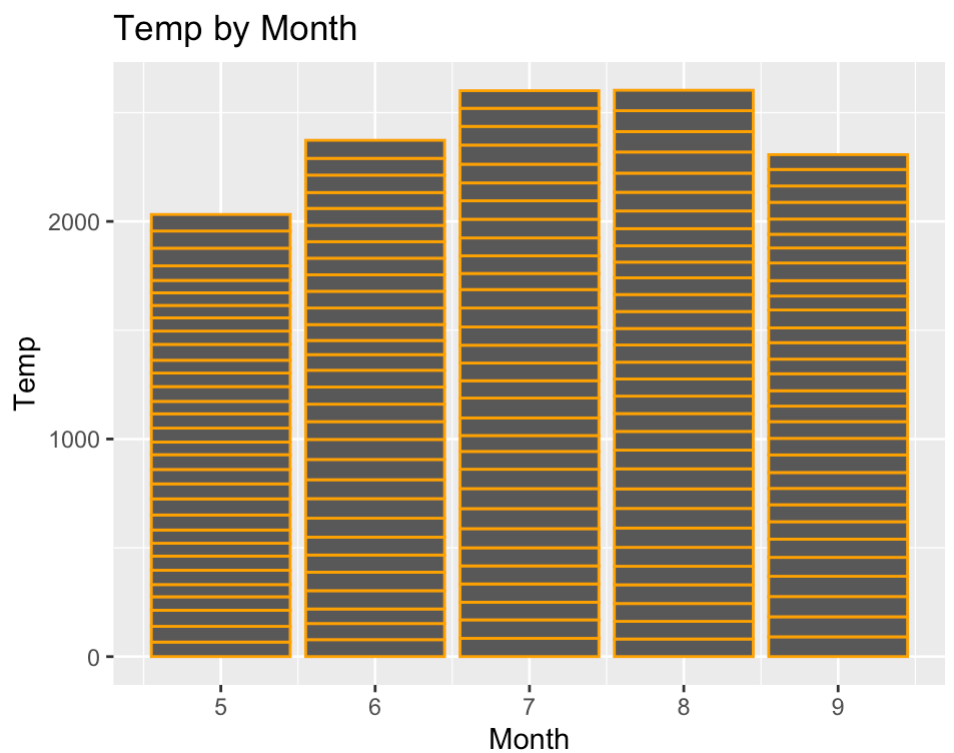
✅ DO

```
ggplot(airquality, aes(x = Day, y = Temp)) +
  geom_line() +
  facet_wrap(~ Month) +
  labs(title = "Daily Temperature by Month", x = "Da
  theme_minimal()
```



❌ Don't

```
ggplot(airquality, aes(x = Month, y = Temp)) +
  geom_bar(stat = "identity", color = 'orange') +
  ggtitle("Temp by Month")  # Misleading chart type
```

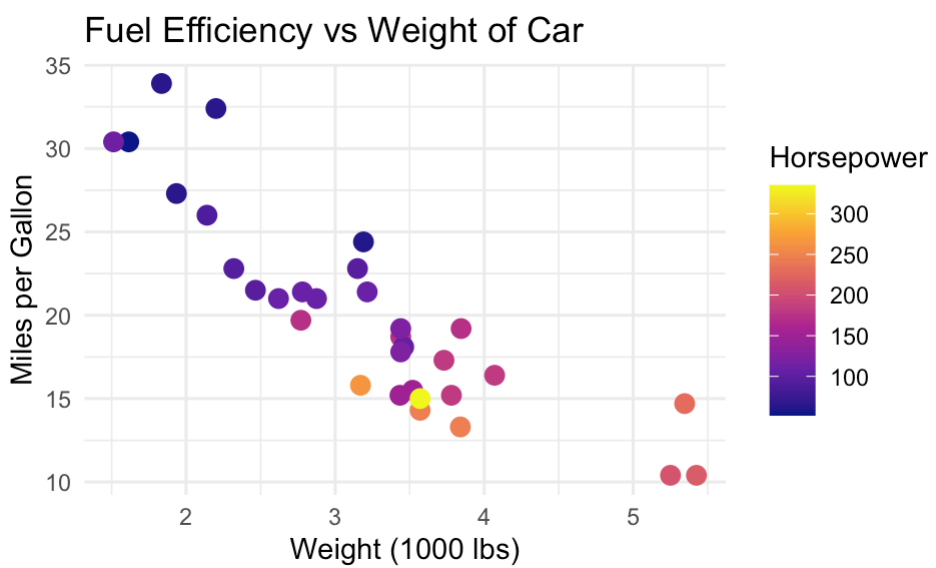


It stacks daily temperature on top of each other, which makes no sense

DO: Use accessible colors

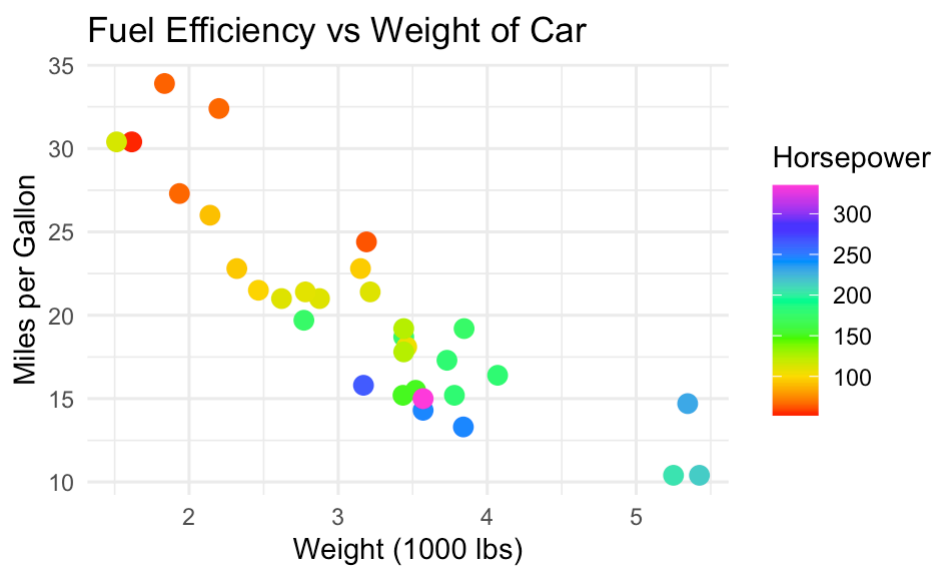
✅ DO use `viridis` scale

```
ggplot(mtcars, aes(x = wt, y = mpg, color = hp)) +
  geom_point(size = 3) +
  scale_color_viridis_c(option = "plasma") +
  labs(
    title = "Fuel Efficiency vs Weight of Car",
    x = "Weight (1000 lbs)",
    y = "Miles per Gallon",
    color = "Horsepower"
  ) +
  theme_minimal()
```



❌ Don't use rainbow colors

```
ggplot(mtcars, aes(x = wt, y = mpg, color = hp)) +
  geom_point(size = 3) +
  scale_color_gradientn(colors = rainbow(7)) +
  labs(
    title = "Fuel Efficiency vs Weight of Car",
    x = "Weight (1000 lbs)",
    y = "Miles per Gallon",
    color = "Horsepower"
  ) +
  theme_minimal()
```

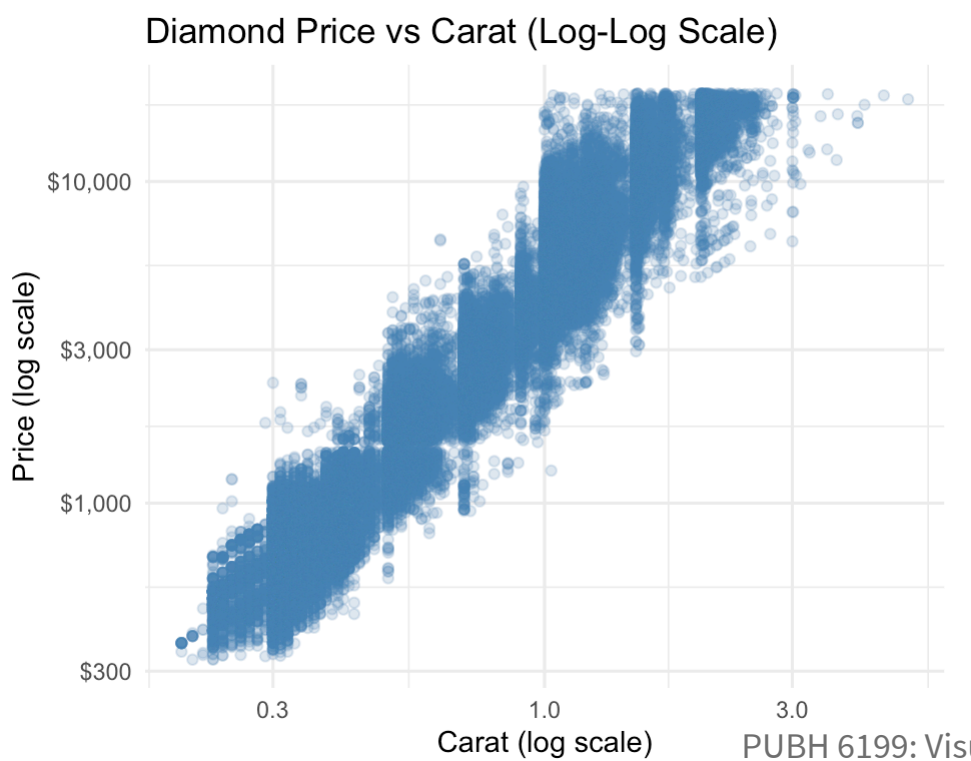


DO: Order categorical axes intentionally

DO: Use appropriate scales

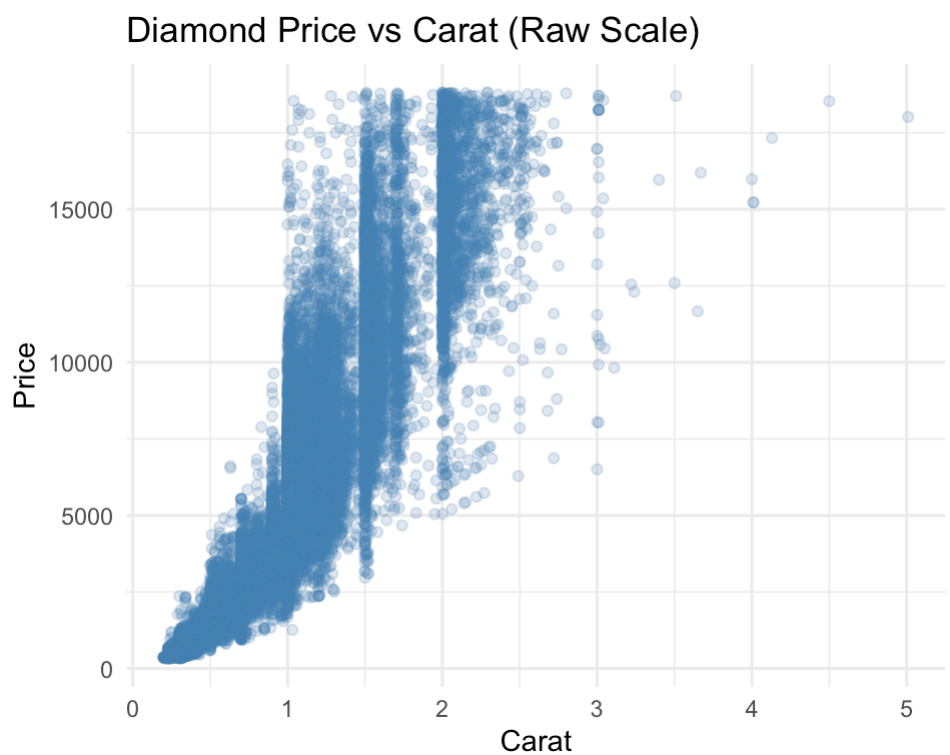
✅ DO log-transform right skewed variables

```
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point(alpha = 0.2, color = "steelblue") +
  scale_x_log10() +
  scale_y_log10(labels = scales::dollar_format(accu
  labs(
    title = "Diamond Price vs Carat (Log-Log Scale)"
    x = "Carat (log scale)",
    y = "Price (log scale)") +
  theme_minimal()
```



❌ Don't mindlessly use raw scale

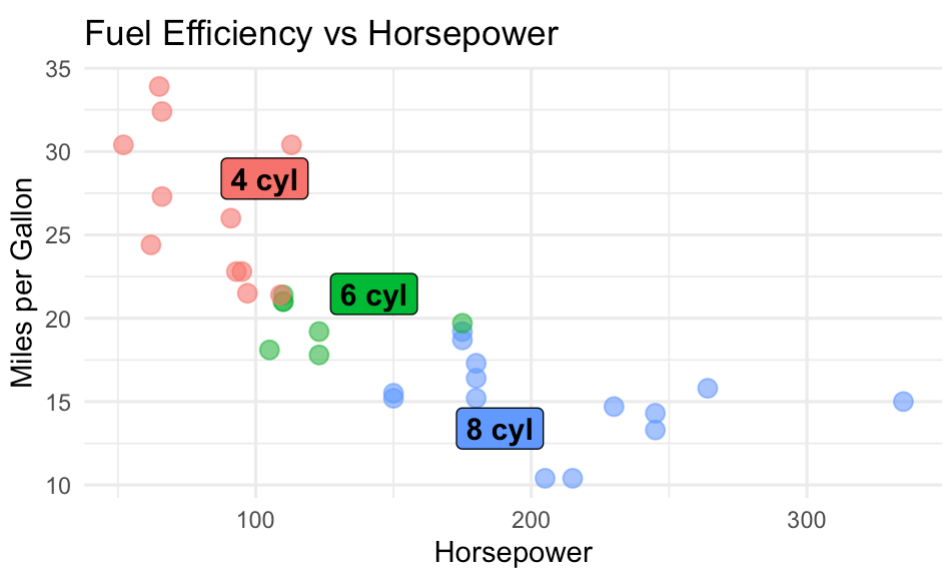
```
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point(alpha = 0.2, color = "steelblue") +
  labs(
    title = "Diamond Price vs Carat (Raw Scale)",
    x = "Carat",
    y = "Price"
  ) +
  theme_minimal()
```



DO: Add helpful annotations

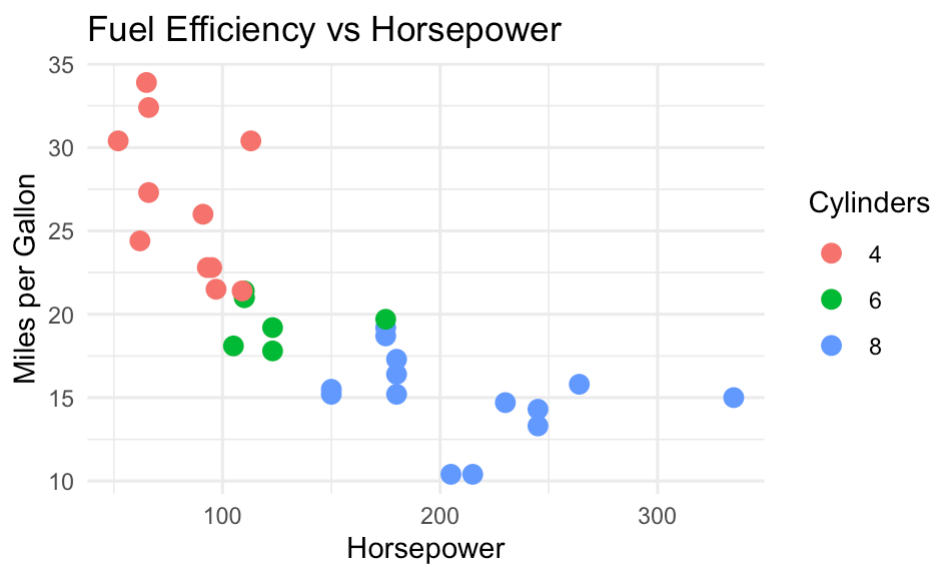
✅ DO use direct labels

```
label_df <- mtcars %>%
  group_by(cyl) %>%
  summarise(hp = mean(hp), mpg = mean(mpg))
ggplot(mtcars, aes(x = hp, y = mpg, color = factor(c
  geom_point(size = 3, alpha = 0.6) +
  ggrepel::geom_label_repel(
    data = label_df,
    aes(label = paste0(cyl, " cyl"), fill = factor(c
    show.legend = FALSE, size = 4, fontface = "bold"
  labs(title = "Fuel Efficiency vs Horsepower", x =
    guides(color = "none") +
  theme_minimal()
```



❌ Don't only rely on legends

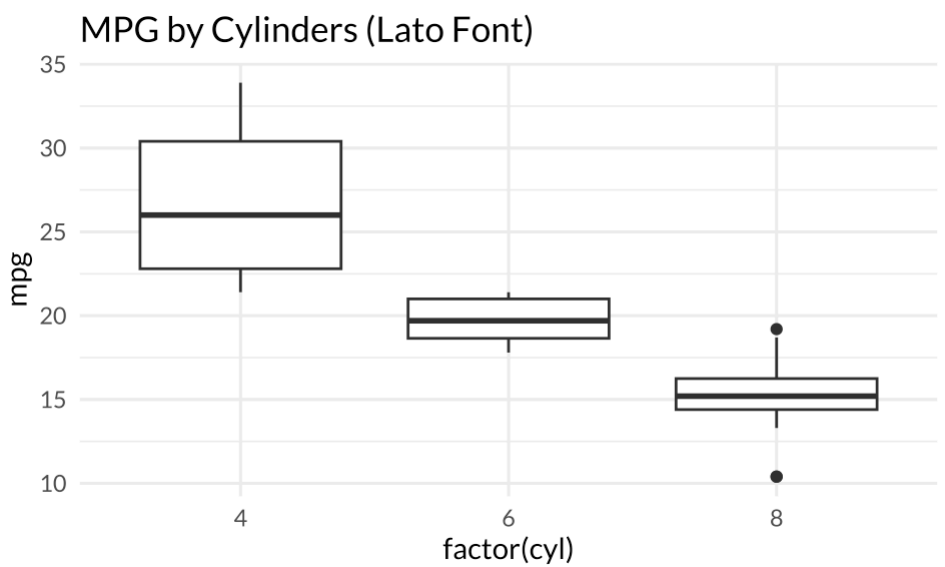
```
ggplot(mtcars, aes(x = hp, y = mpg, color = factor(c
  geom_point(size = 3) +
  labs(
    title = "Fuel Efficiency vs Horsepower",
    x = "Horsepower",
    y = "Miles per Gallon",
    color = "Cylinders"
  ) +
  theme_minimal()
```



DO: Use clear typeface and fonts

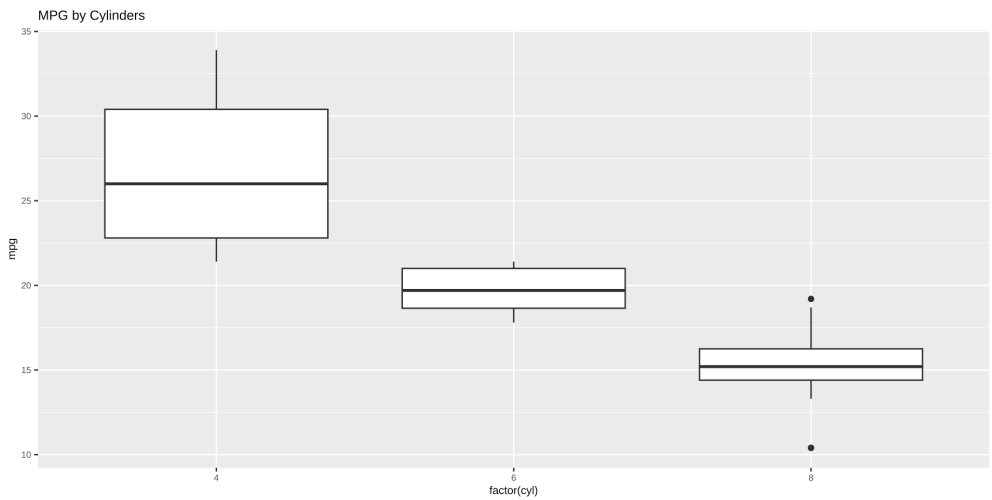
✅ DO

```
library(showtext)
font_add_google("Lato", "lato")
showtext_auto()
ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
  geom_boxplot() +
  theme_minimal(base_family = "lato") +
  labs(title = "MPG by Cylinders (Lato Font)")
```



❌ Don't have tiny fonts

```
ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
  geom_boxplot() +
  theme(text = element_text(size = 8)) +
  labs(title = "MPG by Cylinders")
```

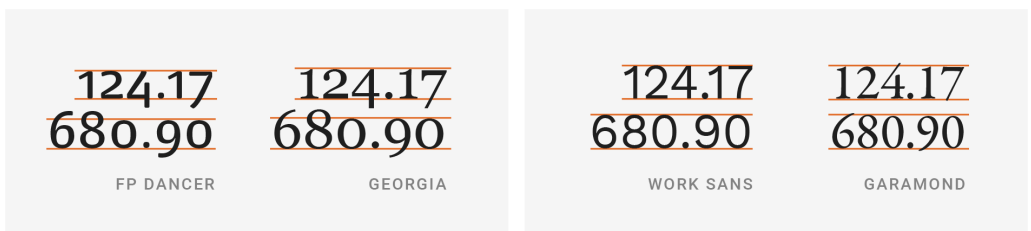


More on typeface

Source: Typography for a better user experience, by Suvo Ray

Some general rules

- Use **sans-serif** fonts for digital displays (e.g., Arial, Helvetica, Lato).
- **Serif** fonts are typically only used for visualization headlines (e.g., Times New Roman, Georgia).
- Avoid using too many typefaces (just 1-3)
- Use a typeface with lining figures for numerals



NOT IDEAL: OLDSTYLE BETTER: LINING

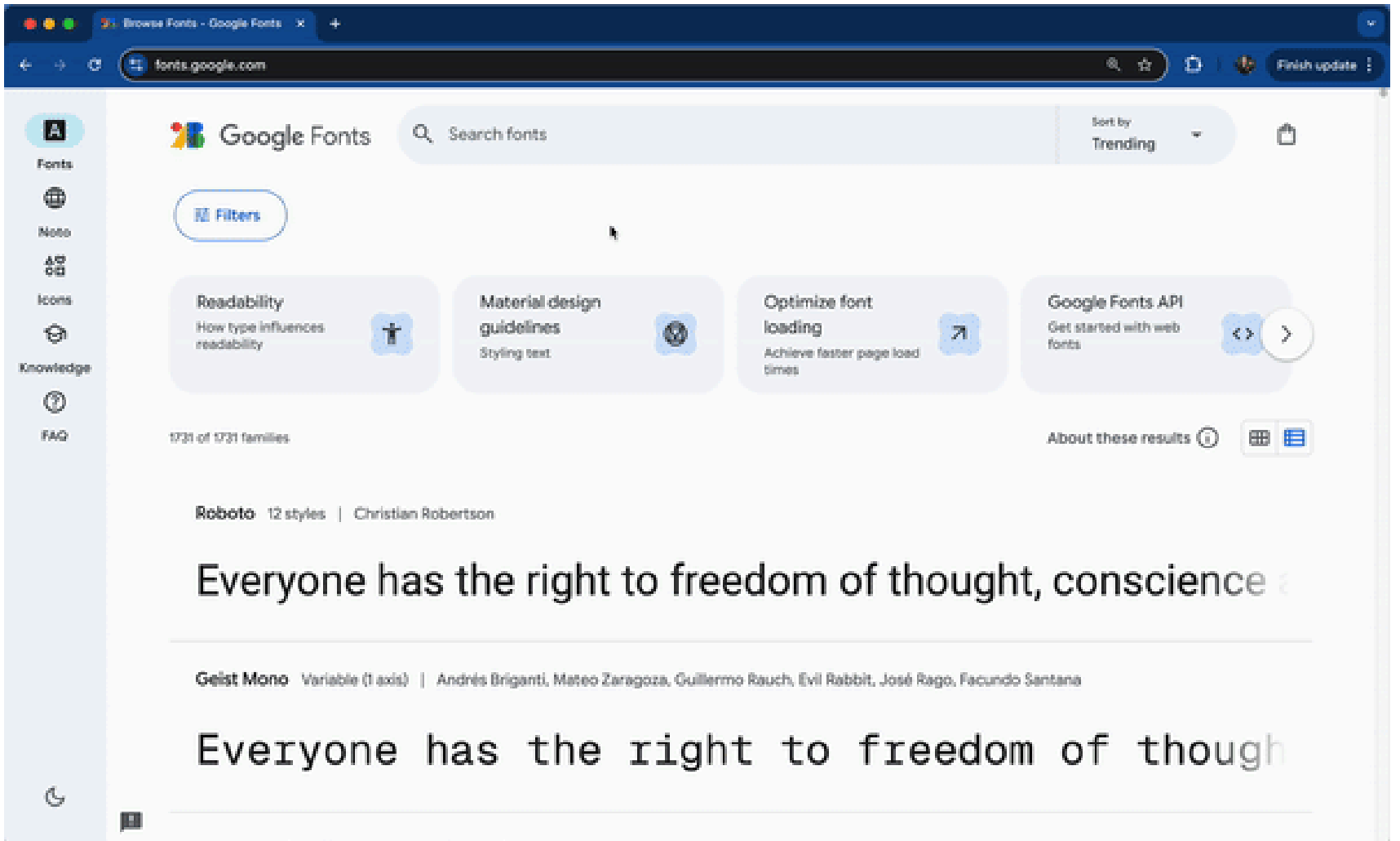
- Use a monospaced typeface for numerals

NOT IDEAL: PROPORTIONAL BETTER: TABULAR

Create hierarchy with font size, weight, and style

Source: The UX Designer's Guide to Typography by Chaosamran_Studio

Pick a typeface from Google Fonts



https://fonts.google.com/

Source: EDS 240

End-of-Class Survey

Fill out the end-of-class survey

~ This is the end of Lab 3 ~

10:00