R programming basics: introduction to the tidyverse

Ecological Systems Modeling

Jan 22-26, 2024

Active participation (optional)

- Open RStudio in Jupyter Hub
- In the Files/Plots/etc. pane, navigate to: `$HOME/Labs/Intro_R_part3/`
- Click on `Intro_to_R_part3_tidyverse.rmd`
- File should open in the Source pane
- Run the code chunks, add to chunks, or type code in Console

Learning objectives

- Wrangle data using `dplyr`
- Re-structure data using `tidyr`
- Create plots using `ggplot2`

Tidyverse

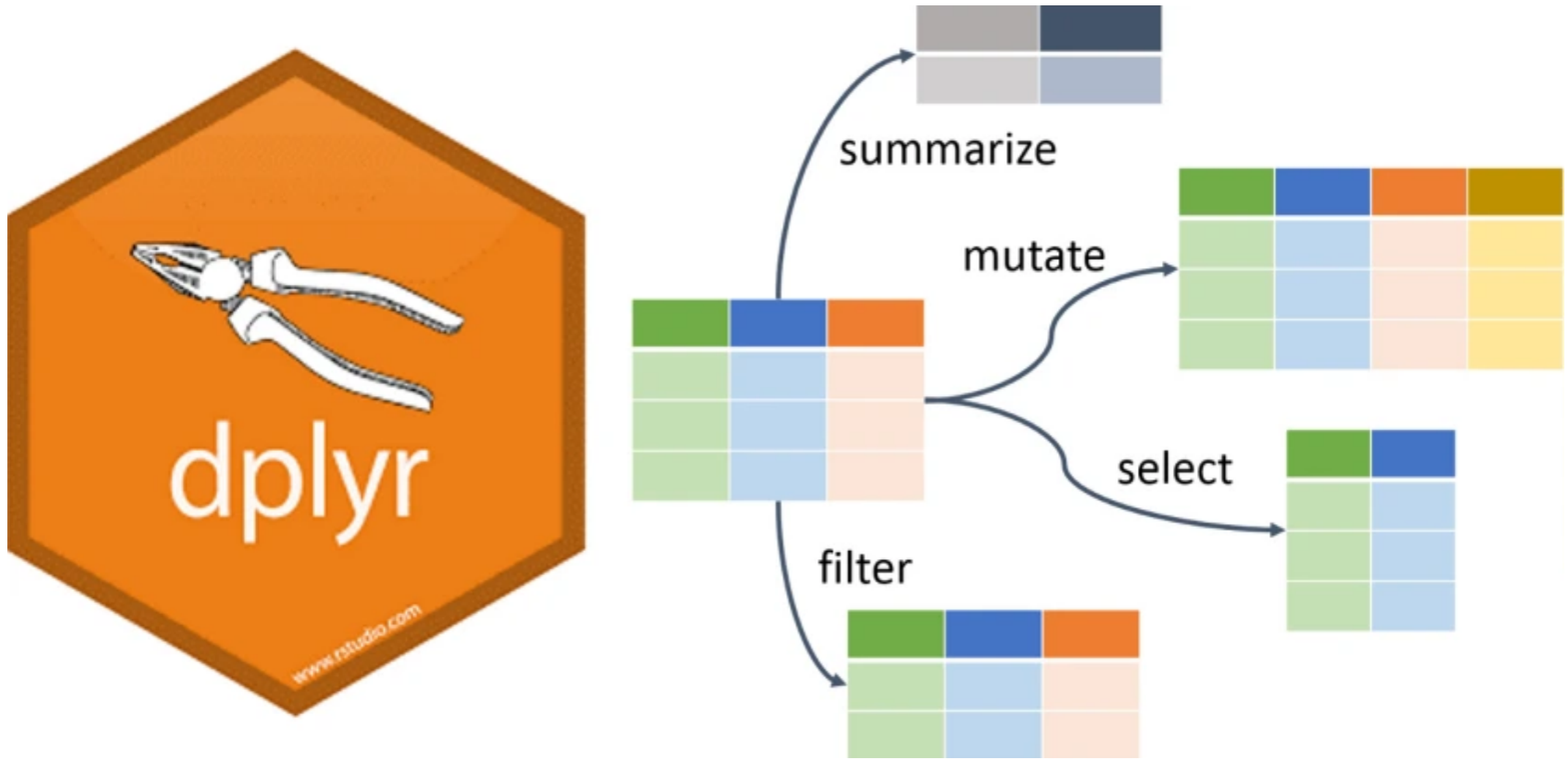
- Collection (“metapackage”) of R packages designed for data science
- Data analysis pipeline to clean, process, model, and visualize data
- Write simpler, more readable code

Source: education.RStudio.com

Essential packages: `dplyr` and `tidyr`

You probably spend a lot of time preparing, organizing, formatting, and cleaning data

- `dplyr` is for data cleaning and manipulation

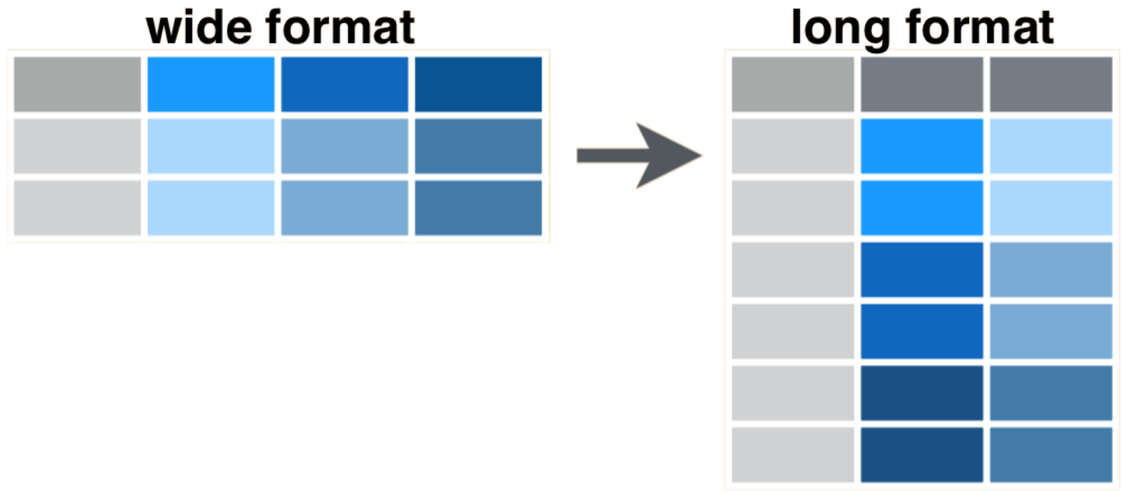


Source: Towards Data Science

Essential packages: `dplyr` and `tidyr`

You probably spend a lot of time preparing, organizing, formatting, and cleaning data

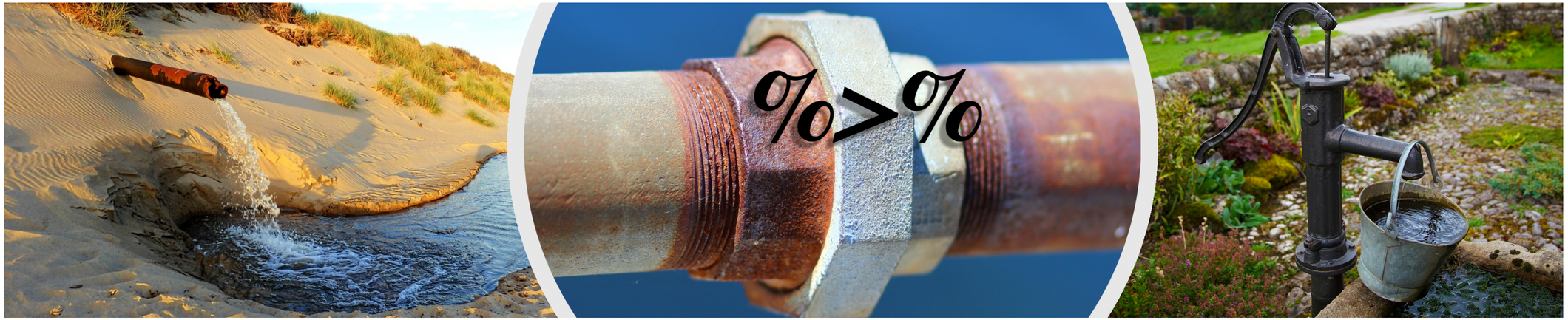
- `tidyr` is useful for converting between different data formats (e.g., wide to long)



Source: R for Excel Users

The pipe operator %>%

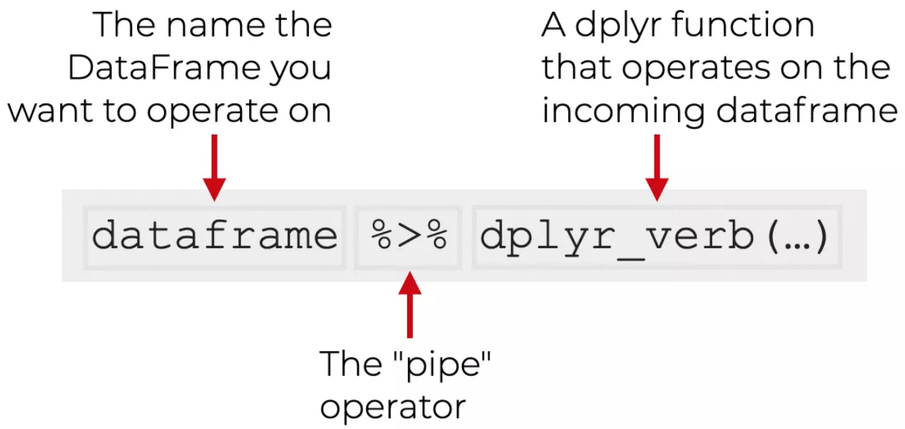
- The tidyverse pipe operator %>% performs a sequences of actions
- Imagining the data stream as a flow of water in pipes



Source: static-bcrf.biochem.wisc.edu

The pipe operator %>%

- Stream of data can be modified by each successive function (piped)
- “Verbs” in `dplyr` are `arrange()`, `mutate()`, `select()`, `filter()`, and `summarize()`



WHEN YOU USE MULTI-STEP DPLYR PIPES, THE DPLYR VERBS CAN BE ON DIFFERENT LINES

```
dataframe %>%
   verb1(…) %>%
   verb2(…)
```

Source: sharpsightlabs.com

Base R vs. tidyverse step structure

Some fake data

```
df <- data.frame(w = c(1:4), x = c(5:8), y = c(9:12))
```

Base R steps

Object (e.g., a data frame) is modified multiple times

```
# Base R step structure
df_modified <- df[,1:2] # Select cols 1 and 2
df_modified <- subset(df_modified, x >= 3) # Subset x values >= 3
df_modified$z <- rowSums(df_modified) # Add cols x and y
df_modified
```

```
  w x  z
1 1 5  6
2 2 6  8
3 3 7 10
4 4 8 12
```

Base R vs. tidyverse step structure

Some fake data

```r
df <- data.frame(w = c(1:4), x = c(5:8), y = c(9:12))
```

Tidyverse steps

Use pipeline to modify an object

```r
df_modified <- df %>%
  select(w, x) %>% # Select by column name
  filter(x >= 3) %>% # Subset x values >= 3
  rowwise() %>% # Row-wise operations
  mutate(z = sum(c(w, x))) # Sum rows
df_modified
```

```
# A tibble: 4 × 3
# Rowwise:
      w     x     z
  <int> <int> <int>
1     1     5     6
2     2     6     8
3     3     7    10
4     4     8    12
```

Syntax comparison

Base R

- Not always intuitive
- Sometimes inconsistent

```r
# Base R step structure
df_modified <- df[,1:2] # Select cols 1
df_modified <- subset(df_modified, x >=
df_modified$z <- rowSums(df_modified) #
df_modified
```

```
  w x  z
1 1 5  6
2 2 6  8
3 3 7 10
4 4 8 12
```

Tidyverse

- Easier to learn and more consistent
- Functions have descriptive names

```r
df_modified <- df %>%
  select(w, x) %>% # Select by column n
  filter(x >= 3) %>% # Subset x values
  rowwise() %>% # Row-wise operations
  mutate(z = sum(c(w, x))) # Sum rows
df_modified
```

```
# A tibble: 4 × 3
# Rowwise:
      w     x     z
  <int> <int> <int>
1     1     5     6
2     2     6     8
3     3     7    10
4     4     8    12
```

dplyr: wrangle data

- Use readr to import CRVO21 weather data (from Lab 2) as table

```
# File location
weath_fl <- here("Intro_R_part3", "files", "CRVO21.txt")

# Pipeline to import file, convert to data frame, rename cols, and filter
weath_dat <- weath_fl %>%
  read_table(skip = 1, col_names = FALSE,
             col_types = cols(X7 = col_skip())) %>%
  data.frame()  %>%
  rename(c("month" = 1, "day" = 2, "tmax" = 3, "tmin" = 4, "prec" = 5, "dd50" = 6)) %
  filter(month %in% c(6,7,8))
```

`dplyr`: wrangle data

- Use `readr` to import `CRVO21` weather data (from Lab 2) as table
- Convert to data frame

```
# File location
weath_fl <- here("Intro_R_part3", "files", "CRVO21.txt")

# Pipeline to import file, convert to data frame, rename cols, and filter
weath_dat <- weath_fl %>%
  read_table(skip = 1, col_names = FALSE,
             col_types = cols(X7 = col_skip())) %>%
  data.frame()  %>%
  rename(c("month" = 1, "day" = 2, "tmax" = 3, "tmin" = 4, "prec" = 5, "dd50" = 6)) %
  filter(month %in% c(6,7,8))
```

`dplyr`: wrangle data

- Use `readr` to import `CRVO21` weather data (from Lab 2) as table
- Convert to data frame
- Rename columns and filter data for summer months

```
# File location
weath_fl <- here("Intro_R_part3", "files", "CRVO21.txt")

# Pipeline to import file, convert to data frame, rename cols, and filter
weath_dat_summer <- weath_fl %>%
  read_table(skip = 1, col_names = FALSE,
             col_types = cols(X7 = col_skip())) %>%
  data.frame() %>%
  rename(c("month" = 1, "day" = 2, "tmax" = 3, "tmin" = 4, "prec" = 5, "dd50" = 6)) %
  filter(month %in% c(6,7,8))
```

`dplyr`: create new columns (variables)

- e.g., convert numeric months to month abbreviations
  - `mutate()` creates new columns (may inc. conditional statements)

```
# Create factor variable for month
weath_dat_summer <- weath_dat_summer %>%
  # Create column with month abbreviations
  mutate(month_char = ifelse(month == 6, "Jun", ifelse(month == 7, "Jul", "Aug"))) %>
  # Put new column data in correct order using `factor()`
  mutate(month_fact = factor(month_char, levels = c("Jun", "Jul", "Aug")))
```

`dplyr`: create new columns (variables)

- e.g., convert numeric months to month abbreviations
  - `mutate()` creates new columns (may inc. conditional statements)
  - month abbreviations put in correct order using `factor()`

```
# Create factor variable for month
weath_dat_summer <- weath_dat_summer %>%
  # Create column with month abbreviations
  mutate(month_char = ifelse(month == 6, "Jun", ifelse(month == 7, "Jul", "Aug"))) %>
  # Put new column data in correct order using `factor()`
  mutate(month_fact = factor(month_char, levels = c("Jun", "Jul", "Aug")))
```

Comparison of original vs. "wrangled" data

```r
# View structure of original data
str(weath_dat)
```

```
'data.frame':   92 obs. of  6 variables:
 $ month: num  6 6 6 6 6 6 6 6 6 6 ...
 $ day  : num  1 2 3 4 5 6 7 8 9 10 ...
 $ tmax : num  93 89.3 82.8 77.3 65.6 63.5 61.9 65.4 68.6 67.5 ...
 $ tmin : num  57.6 55.4 50.6 48.8 45 38.1 47.8 43.3 36.2 41.1 ...
 $ prec : chr  "0.00" "0.00" "0.00" "0.00" ...
 $ dd50 : num  25.3 22.4 16.7 13.1 5.9 3.6 5 5.4 5.3 5.8 ...
```

```r
# View structure of subsetted data
str(weath_dat_summer)
```

```
'data.frame':   92 obs. of  8 variables:
 $ month      : num  6 6 6 6 6 6 6 6 6 6 ...
 $ day        : num  1 2 3 4 5 6 7 8 9 10 ...
 $ tmax       : num  93 89.3 82.8 77.3 65.6 63.5 61.9 65.4 68.6 67.5 ...
 $ tmin       : num  57.6 55.4 50.6 48.8 45 38.1 47.8 43.3 36.2 41.1 ...
 $ prec       : chr  "0.00" "0.00" "0.00" "0.00" ...
 $ dd50       : num  25.3 22.4 16.7 13.1 5.9 3.6 5 5.4 5.3 5.8 ...
 $ month_char: chr  "Jun" "Jun" "Jun" "Jun" ...
 $ month_fact: Factor w/ 3 levels "Jun","Jul","Aug": 1 1 1 1 1 1 1 1 1 1 ...
```

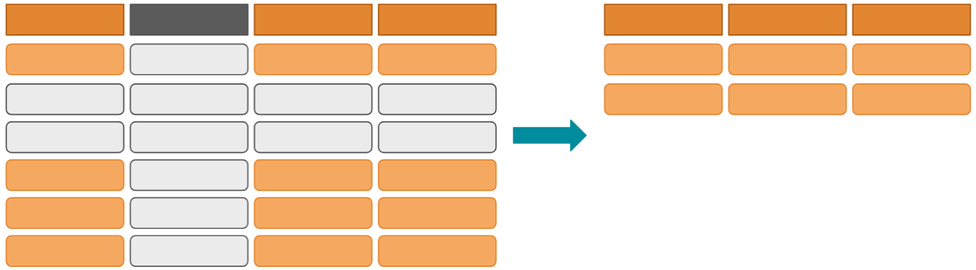
dplyr: calculate summary statistics

Summarize tmax and tmin for summer months

```
1  # Calculate average temperature in summ
2  weath_dat_mean <- weath_dat_summer %>%
3    group_by(month) %>%
4    summarize(
5      across(c(tmax,tmin), mean, na.rm =
6
7  # Look at results
8  weath_dat_mean
```

```
# A tibble: 3 × 3
  month  tmax  tmin
  <dbl> <dbl> <dbl>
1     6  80.5  52.0
2     7  86.2  53.9
3     8  84    55.0
```

Customize with **group_by()** and **summarize()**



Source: learn.r-journalism.com

`tidyr`: change data structure

- e.g., use `pivot_wider()` to convert data from long to wide format
- The `select()` function in `dplyr` removes unwanted data

```
# Show Tmax data in wide format
tmax_summer_wide <- weath_dat_summer %>%
  select(-tmin, -prec, -dd50) %>%
  pivot_wider(names_from = day, values_from = tmax)
```

`tidyr`: comparison of data sets

Long format

```
# Data before
dim(weath_dat_summer)
```

```
[1] 92  8
```

```
# Col names
names(weath_dat_summer)
```

```
[1] "month"      "day"        "tmax"       "tmin"       "prec"
[6] "dd50"       "month_char" "month_fact"
```

Wide format

```
# Data  after
dim(tmax_summer_wide)
```

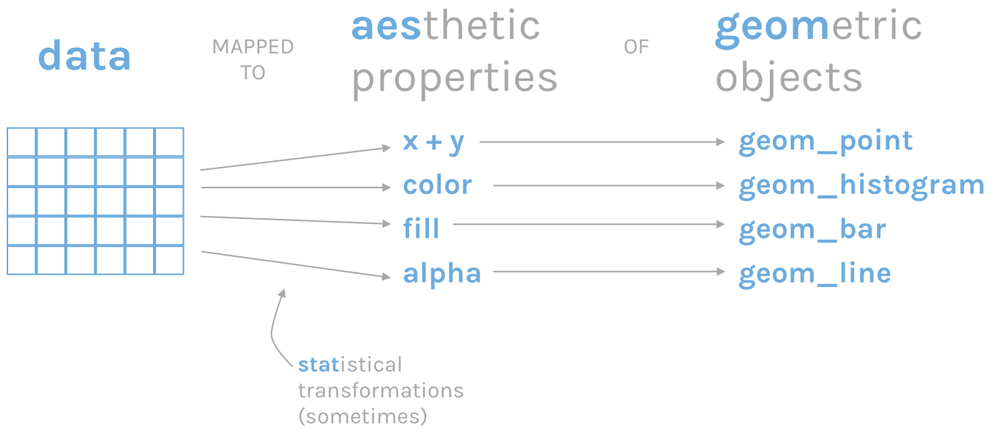
```
[1]  3 34
```

```
# Col names
names(tmax_summer_wide)
```

```
 [1] "month"      "month_char" "month_fact" "1"          "2"
 [6] "3"          "4"          "5"          "6"          "7"
[11] "8"          "9"          "10"         "11"         "12"
[16] "13"         "14"         "15"         "16"         "17"
[21] "18"         "19"         "20"         "21"         "22"
[26] "23"         "24"         "25"         "26"         "27"
[31] "28"         "29"         "30"         "31"
```

`ggplot2`: another essential tidyverse package

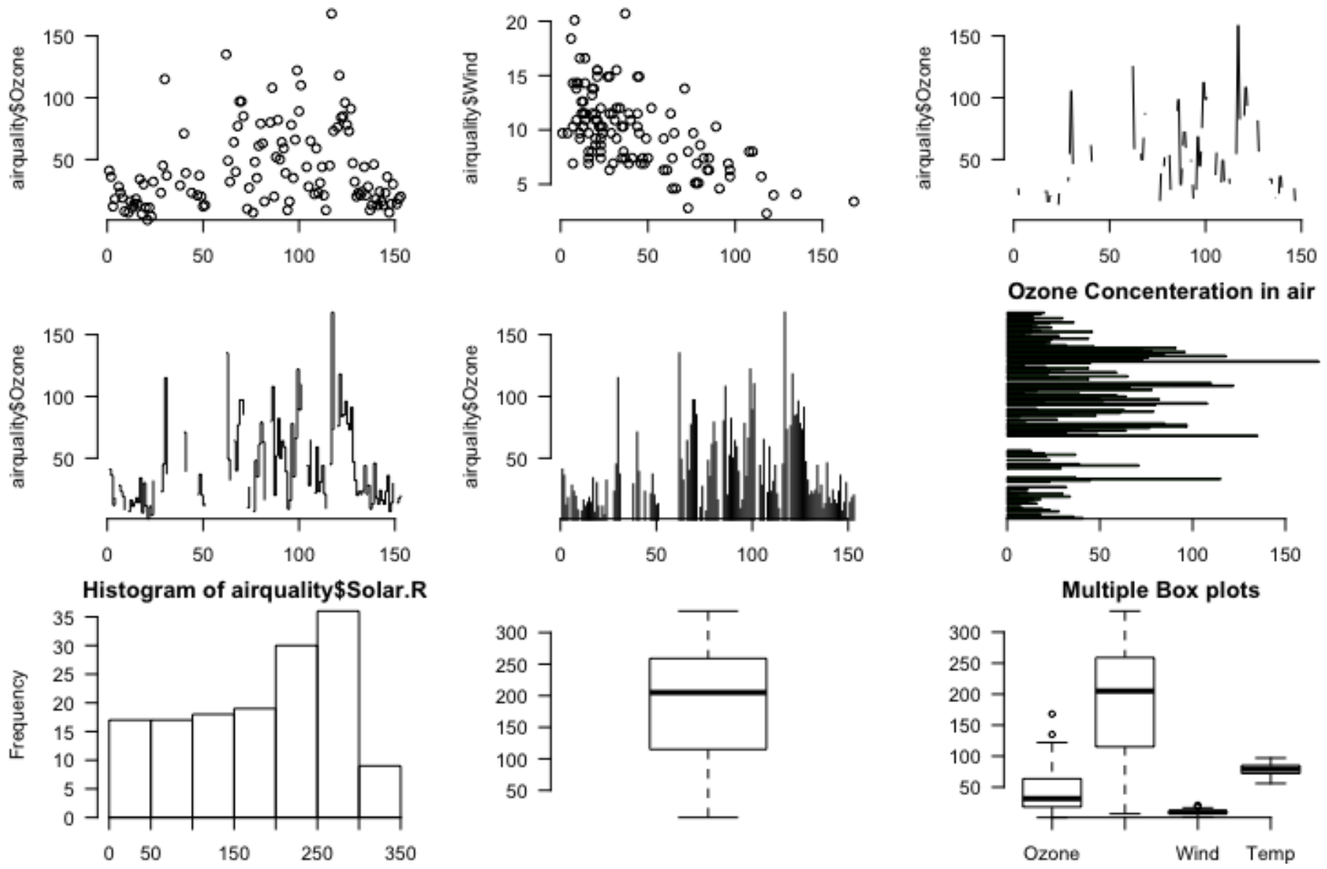
- Uses layered approach to create graphics (The Grammar of Graphics)
- You provide the data, tell `ggplot2` how to map variables to aesthetics, what graphical elements to use, and it takes care of the details



Source: R for the rest of us

Base R plots vs. tidyverse (`ggplot2`) plots

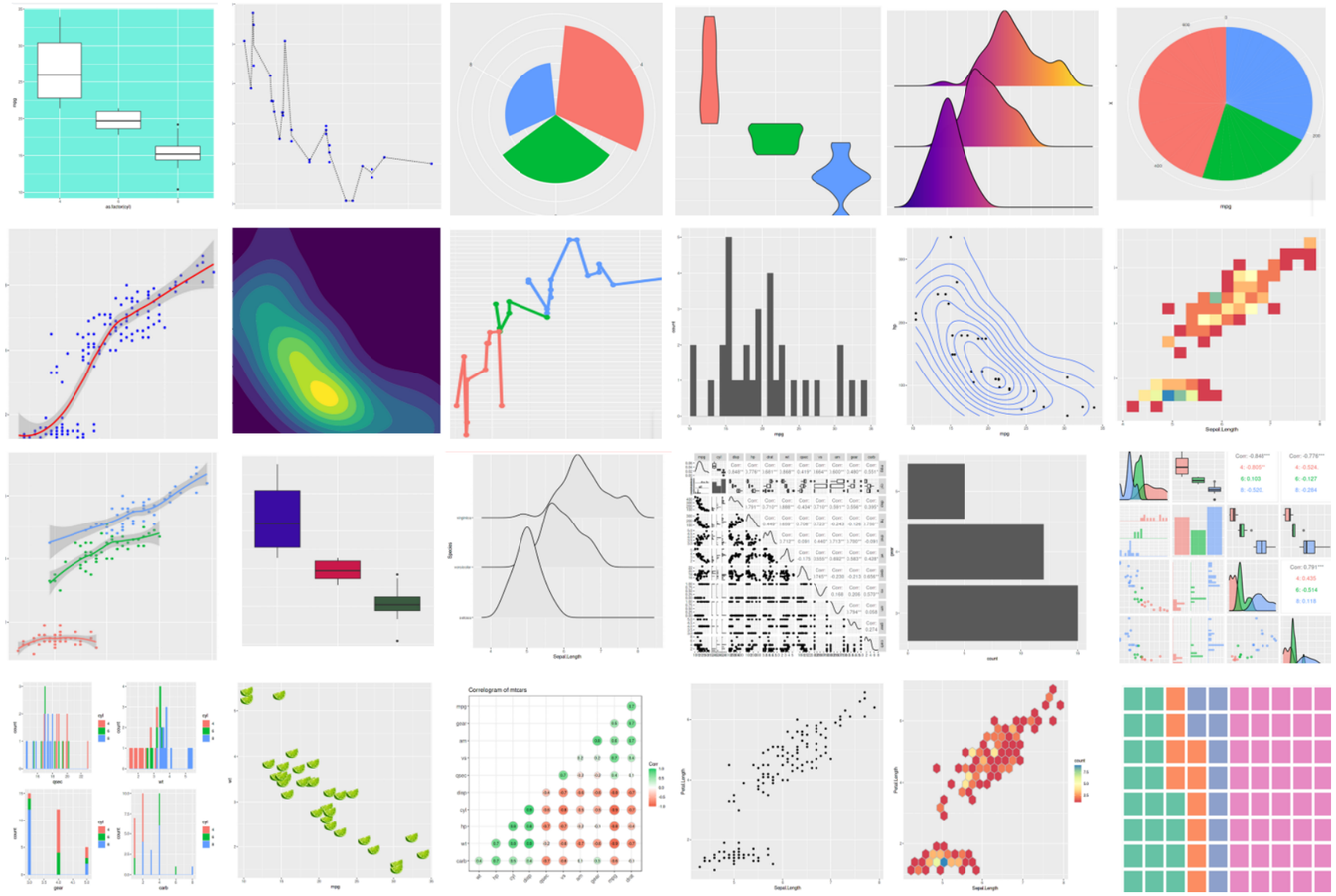
Plots made with base R functions



Source: sharpsightlabs.com

Base R plots vs. tidyverse (`ggplot2`) plots

Plots made with `ggplot2`



Source: www.icertglobal.com

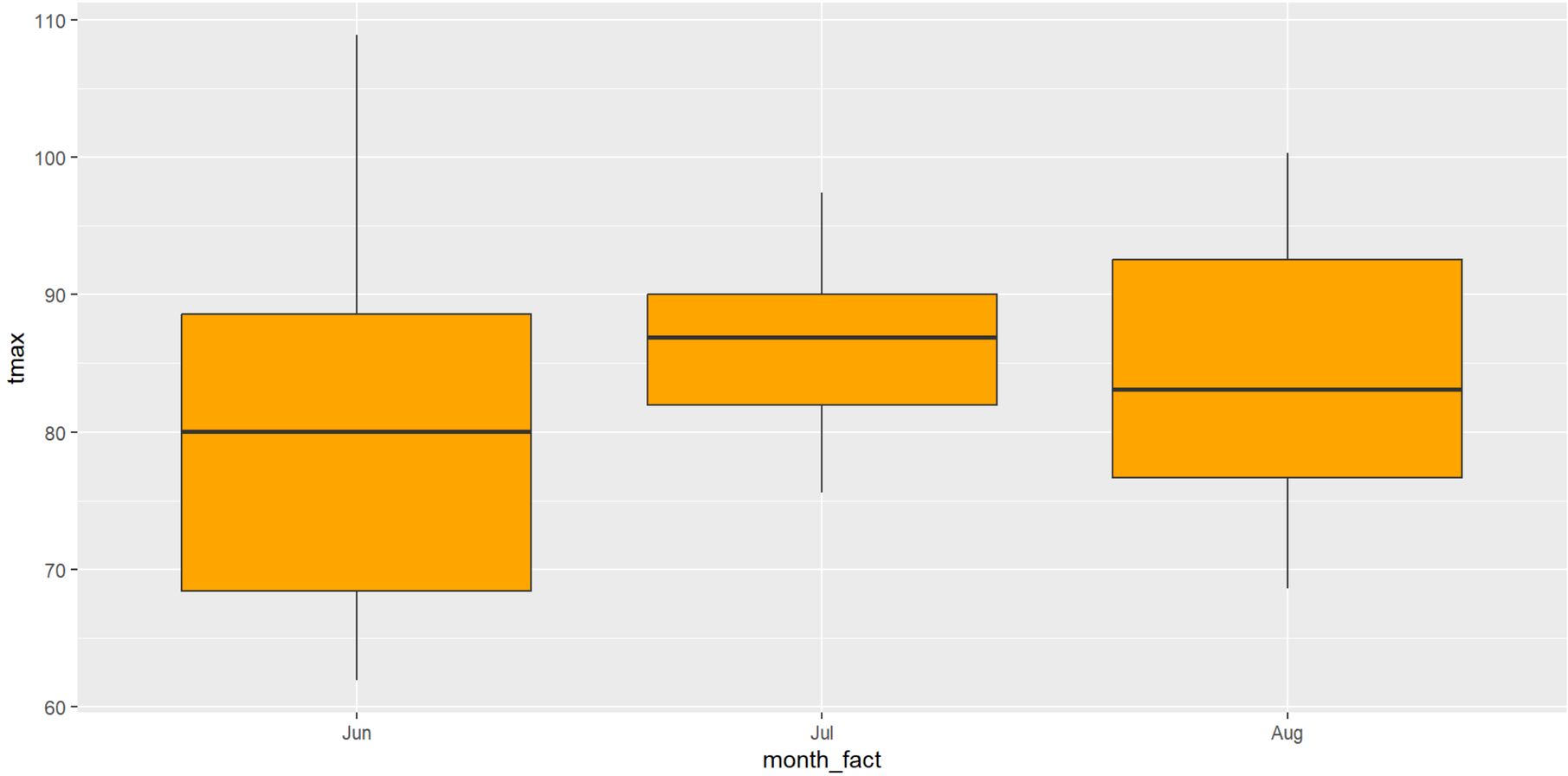
Example bar plot: mean *Tmax*

- Visualize distribution of max temps for summer months
- Box plot created using `ggplot2`
  - Plot using the `ggplot()` function
  - The `+` operator adds layers
  - Aesthetics (x, y, etc.) defined with `aes()`

```
# Create a box plot
# Use month abbreviations for x-axis
# Orange fill for boxes
boxplot_tmax <- ggplot(data = weath_dat_summer,
                       aes(x = month_fact, y = tmax, group = month)) +
  geom_boxplot(fill = "orange")
```

Default box plot

This plot doesn't look that great…



Add layers and custom themes to improve the plot

```
# Make a prettier ggplot
boxplot_tmax <- boxplot_tmax +
  theme_bw() + # Black and white theme
  ggtitle("Maximum temperatures at CRVO in 2021") +
  xlab("Month") + # x-axis label
  ylab("Tmax (deg F)") + # y-axis label
  theme(axis.text = element_text(size = 14), # Increase font sizes
        axis.title = element_text(size = 18),
        title = element_text(size = 18))
```

- Use black and white theme

Add layers and custom themes to improve the plot

```
# Make a prettier ggplot
boxplot_tmax <- boxplot_tmax +
  theme_bw() + # Black and white theme
  ggtitle("Maximum temperatures at CRVO in 2021") +
  xlab("Month") + # x-axis label
  ylab("Tmax (deg F)") + # y-axis label
  theme(axis.text = element_text(size = 14), # Increase font sizes
        axis.title = element_text(size = 18),
        title = element_text(size = 18))
```

- Use black and white theme
- Add custom labels

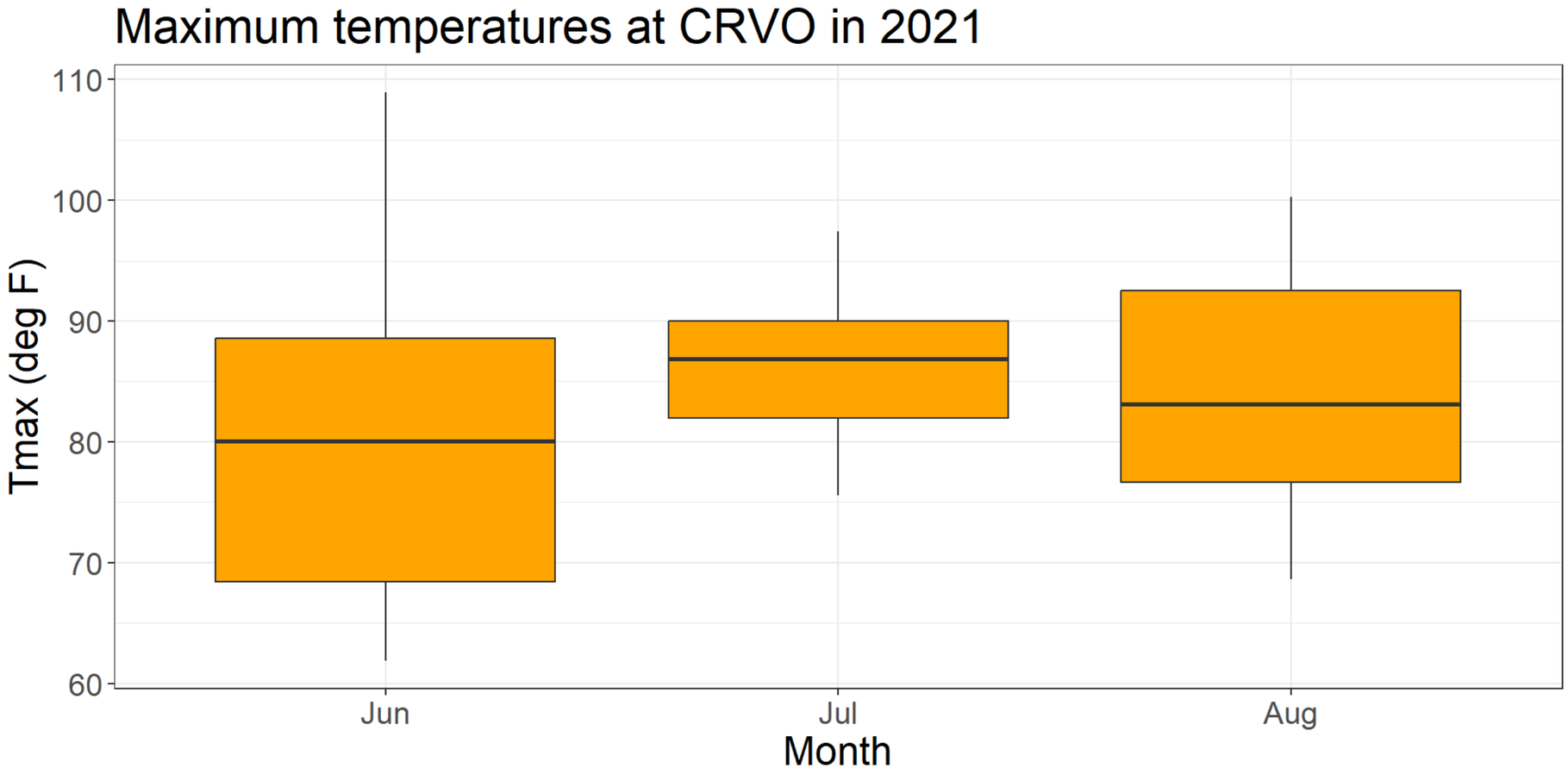
Add layers and custom themes to improve the plot

```
# Make a prettier ggplot
boxplot_tmax <- boxplot_tmax +
  theme_bw() + # Black and white theme
  ggtitle("Maximum temperatures at CRVO in 2021") +
  xlab("Month") + # x-axis label
  ylab("Tmax (deg F)") + # y-axis label
  theme(axis.text = element_text(size = 14), # Increase font sizes
        axis.title = element_text(size = 18),
        title = element_text(size = 18))
```

- Use black and white theme
- Add custom labels
- Increase font sizes

Improved box plot

This plot looks better!



Summary

- There's many ways to do things in R, but I prefer the tidyverse approach
  - more readable code
  - offers order and consistency
  - functions to support entire end-to-end workflow
  - beautiful graphics

Source: reddit.com/r/Rlanguage