

Medical Statistics and Data Analysis

Lecture 3: Introduction to R - Programming Fundamentals and Visualization

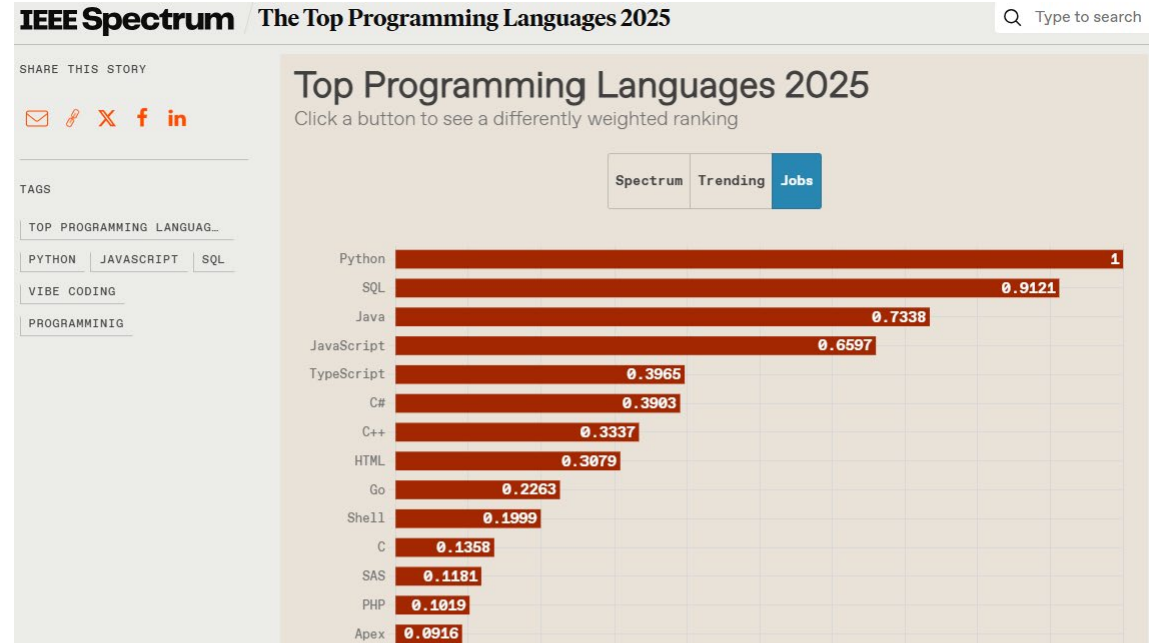
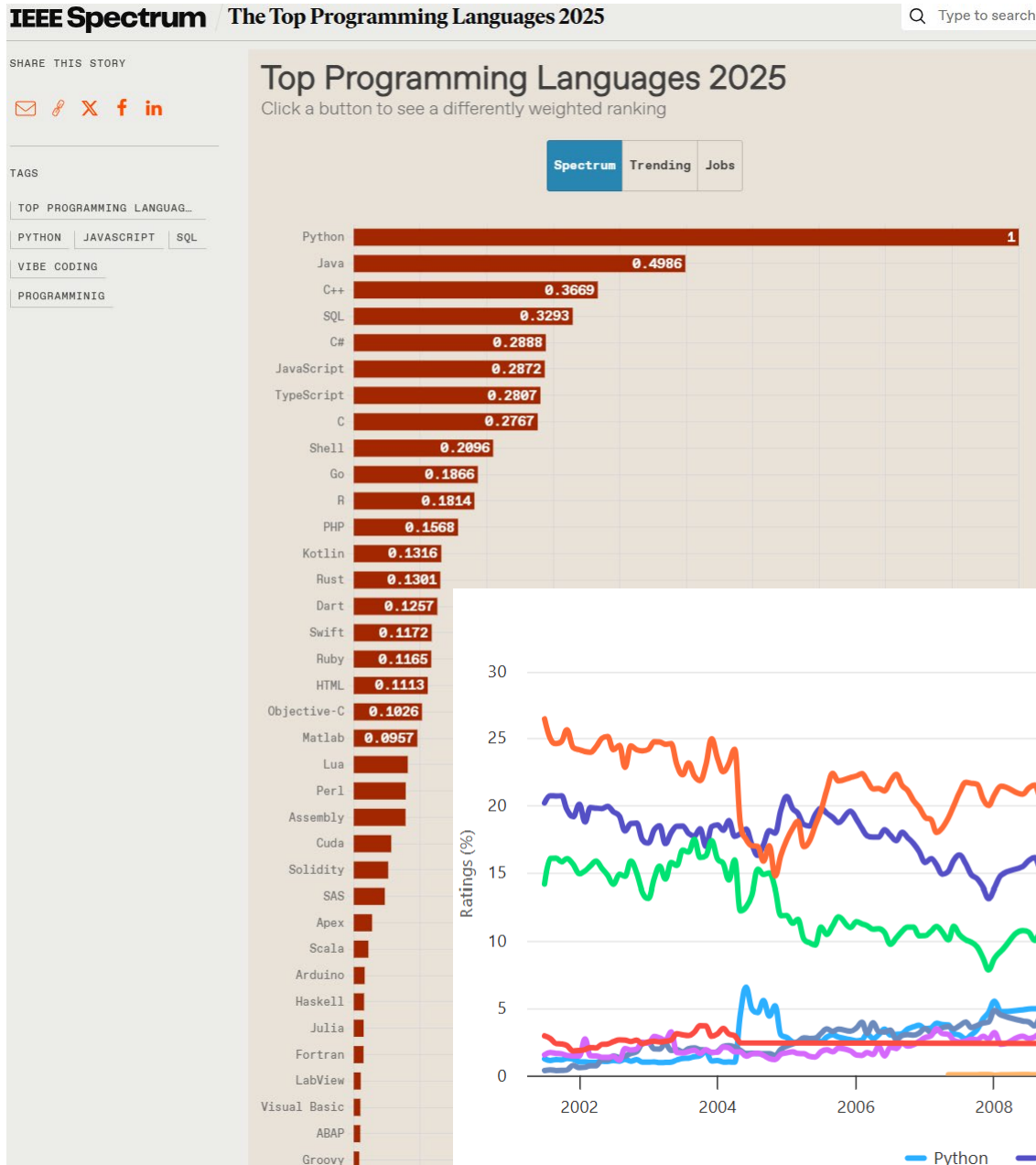
Zhiyu Wan, PhD (wanzhy@shanghaitech.edu.cn)

Assistant Professor of Biomedical Engineering

ShanghaiTech University

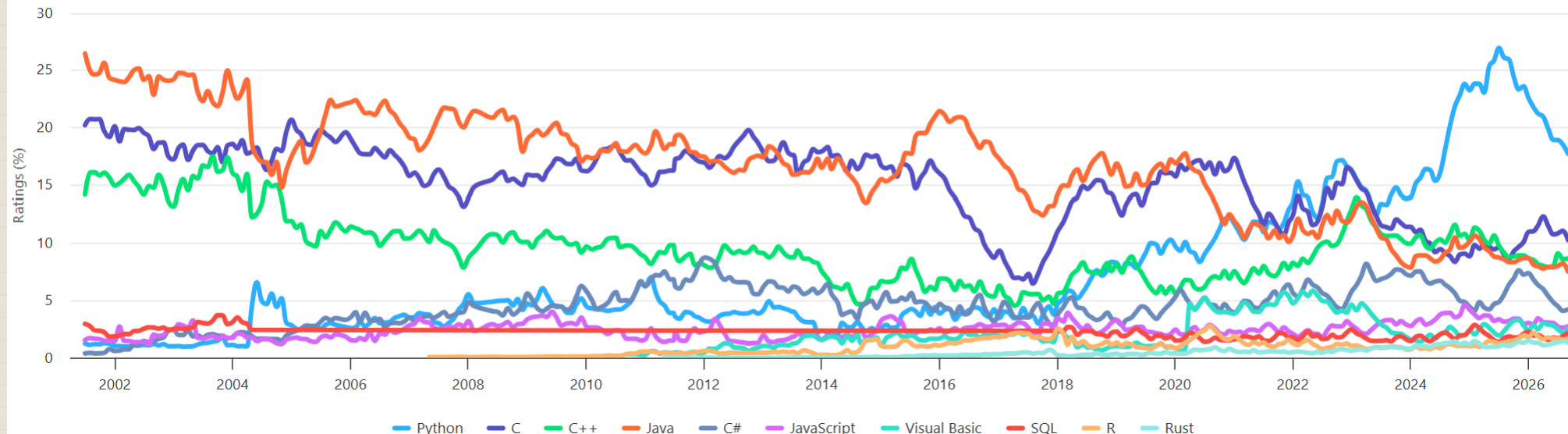
September 23, 2026

Top Programming Languages 2025-2026

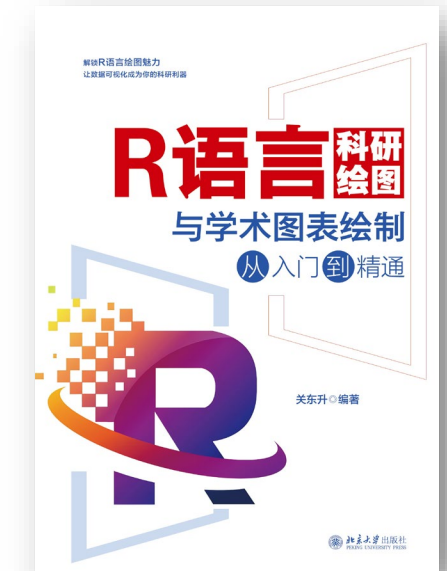
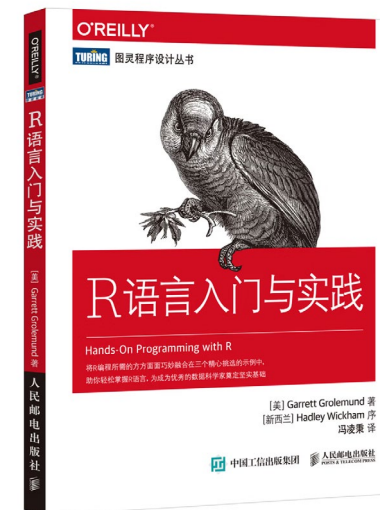
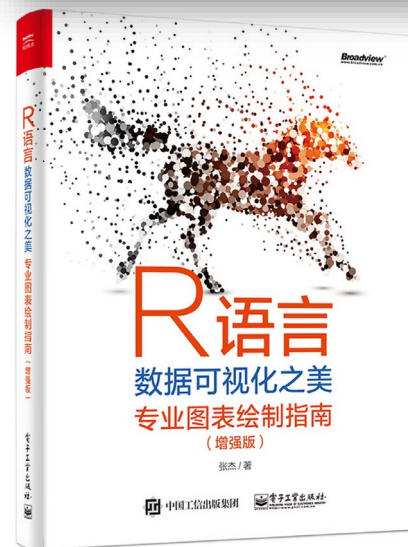
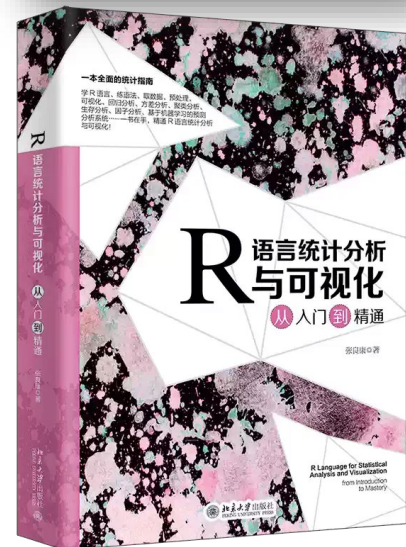
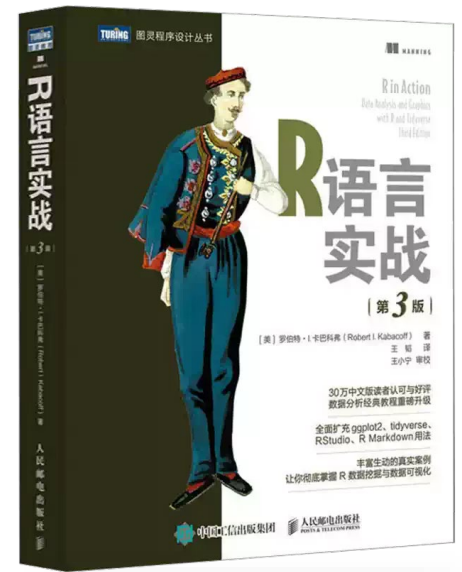
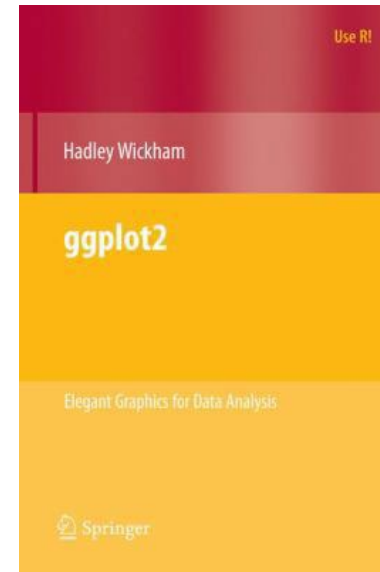
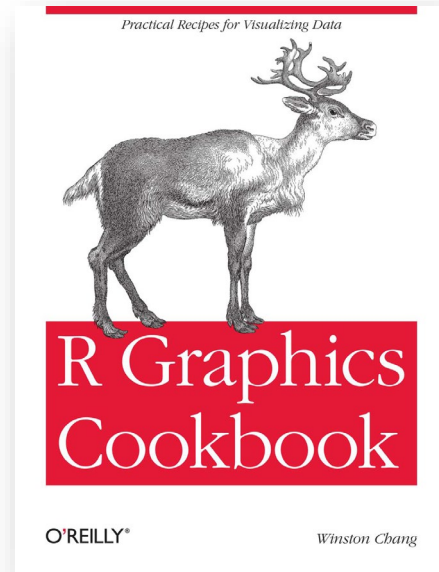
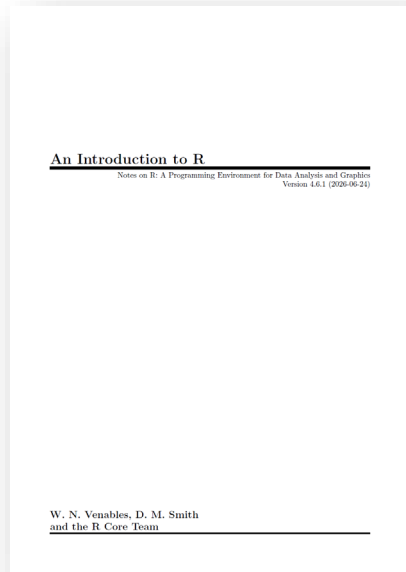


TIOBE Programming Community Index

Source: www.tiobe.com



Textbooks and Recommended Readings



Learning Goals of This Lecture

- Understand the R environment and core syntax
- Work with common data structures and scientific files
- Use functions, control flow, and reproducible scripts
- Choose appropriate univariate, bivariate, and multivariate graphics
- Transform and visualize a real dataset with dplyr and ggplot2

Course Roadmap

- R environment, syntax, and data types
- Control flow, data structures, and functions
- Scientific data import, cleaning, and summaries
- Univariate, bivariate, and multivariate visualization
- ToothGrowth practice with dplyr and ggplot2

Part I: R Foundations

Environment, syntax, data types, and operators

R and RStudio

R

- R is the language and statistical engine
- Runs commands and stores objects
- Provides statistical and graphics functions

RStudio

- RStudio is an integrated development environment
- Source editor and console
- Environment, files, plots, packages, and help

Good habit: write important commands in a script, then run them in the console.

R History, Strengths, and Help

- R grew from the S language and supports statistical computing
- Open-source packages extend analysis and visualization
- Vectorized operations make many tasks concise
- Use `?function`, `help()`, `example()`, and package documentation
- Search error messages only after reading the complete message

Your First R Commands

```
# A comment starts with #  
2 + 3  
  
message <- "Hello, R!"  
print(message)  
  
sqrt(25)  
?mean
```

Objects, Names, and Comments

```
age <- 21
weight_kg <- 63.5
smoker <- FALSE
patient_id <- "P001"

class(weight_kg)
ls()
# R is case-sensitive
```

Atomic Data Types

Numeric and logical

- numeric: 72.5
- integer: 72L
- complex: 2 + 3i
- logical: TRUE or FALSE

Text and conversion

- character: "Control"
- raw: low-level bytes
- class(x) and typeof(x)
- as.numeric(), as.character()

Operators and Missing Values

```
x <- c(4, 7, NA, 10)
```

```
x + 2
```

```
x >= 7
```

```
(x >= 7) & !is.na(x)
```

```
mean(x, na.rm = TRUE)
```

```
# Never test missingness with x == NA
```

Part II: Control Flow and Data Structures

Decisions, loops, vectors, matrices, data frames, and factors

Decision Statements

```
risk_score <- 8

if (risk_score >= 10) {
  category <- "High"
} else if (risk_score >= 5) {
  category <- "Moderate"
} else {
  category <- "Low"
}

category
```

Loops and Vectorization

```
values <- c(4, 7, 10)
result <- numeric(length(values))

for (i in seq_along(values)) {
  result[i] <- values[i]^2
}

# Vectorized equivalent
values^2
```

Vectors

```
bp <- c(118, 125, 132, 121, 140)
length(bp)
bp[1]
bp[c(1, 3, 5)]
bp[bp >= 130]
mean(bp)
bp + 5
```

A vector contains values of one atomic type. Many R operations are vectorized.

Lists, Matrices, and Arrays

Lists

- List: elements may have different types
- Useful for model outputs and nested results
- Access named elements with \$ or [[]]

Matrices and arrays

- Matrix: two dimensions, one atomic type
- Array: more than two dimensions
- Useful for images, repeated measures, and simulations

Data Frames

```
patients <- data.frame(  
  id = c("P01", "P02", "P03", "P04"),  
  age = c(45, 67, 52, 71),  
  group = c("Control", "Treatment", "Treatment", "Control"),  
  sbp = c(118, 142, 135, 128)  
)  
  
str(patients)  
head(patients)
```

Rows, Columns, and Conditions

```
patients[1, ]  
patients[, "sbp"]  
patients[c("age", "sbp")]  
patients[patients$age >= 65, ]  
  
subset(patients, group == "Treatment",  
       select = c(id, age, sbp))
```

Factors and Strings

```
patients$group <- factor(  
  patients$group,  
  levels = c("Control", "Treatment")  
)
```

```
levels(patients$group)  
table(patients$group)  
tolower(patients$id)  
paste("Patient", patients$id)
```

Importing a CSV file

```
patients <- read.csv("patients.csv")

head(patients)
dim(patients)
names(patients)
str(patients)
summary(patients)

# Reproducible paths are better than setwd()
# Consider an RStudio Project for each analysis.
```

Descriptive statistics

Numeric variables

```
mean(x, na.rm = TRUE)
median(x, na.rm = TRUE)
sd(x, na.rm = TRUE)
quantile(x, na.rm = TRUE)
range(x, na.rm = TRUE)
```

Categorical variables

```
table(x, useNA = "ifany")
prop.table(table(x))
```

Group summaries:

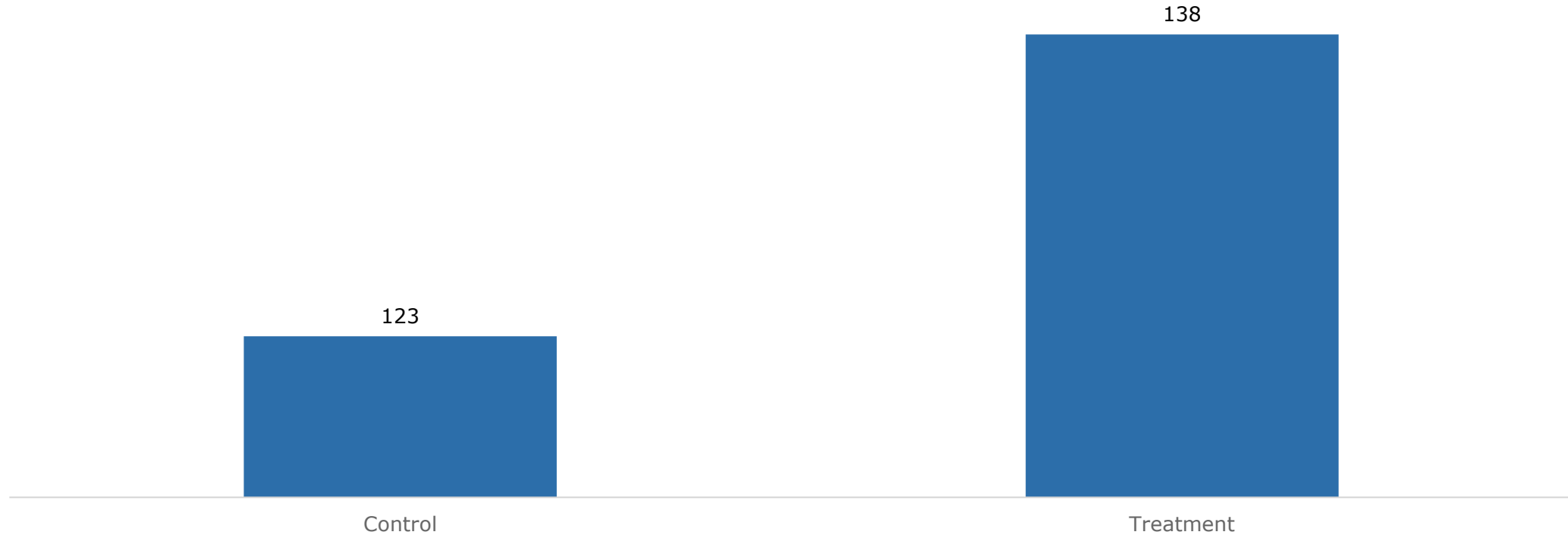
```
tapply(sbp, group, mean, na.rm = TRUE)
```

Blood pressure by study group

```
group_mean <- c(Control = 123, Treatment = 138)

barplot(
  group_mean,
  ylab = "Mean systolic BP (mmHg)",
  col = "steelblue"
)
```

Blood pressure by study group (Figure)



```
barplot(c(Control = 123, Treatment = 138), ylab = "Mean systolic BP (mmHg)")
```

Base R graphics

```
hist(clean$sbp,  
      main = "Distribution of systolic blood pressure",  
      xlab = "Systolic BP (mmHg)",  
      col = "steelblue")  
  
boxplot(sbp ~ group, data = clean,  
        ylab = "Systolic BP (mmHg)",  
        col = c("grey80", "skyblue"))
```

Part III: Functions and Scientific Data

Reusable code, data interfaces, cleaning, and statistical summaries

Writing a Function

```
summarize_bp <- function(x, remove_na = TRUE) {  
  c(  
    mean = mean(x, na.rm = remove_na),  
    median = median(x, na.rm = remove_na),  
    sd = sd(x, na.rm = remove_na)  
  )  
}  
  
summarize_bp(patients$sbp)
```

Arguments, Scope, and Recursion

Arguments

- Arguments may be positional or named
- Default values reduce repetition
- Named arguments improve readability

Scope and recursion

- Local variables exist inside a function
- Global assignment can create hidden dependencies
- Recursion is useful for naturally recursive problems but rarely needed in basic data analysis

Projects and Working Directories

- Use one RStudio Project for each analysis
- Store raw data, scripts, outputs, and figures in separate folders
- Prefer relative paths within the project
- Use `getwd()` to inspect the current directory
- Avoid changing directories repeatedly with `setwd()`

Reading and Writing Files

```
patients <- read.csv("data/patients.csv")
write.csv(patients, "output/patients_clean.csv", row.names = FALSE)

# Common extensions
# readxl::read_excel("data.xlsx")
# writexl::write_xlsx(patients, "output.xlsx")
# jsonlite::fromJSON("data.json")
```

Missing Values and Outliers

```
colSums(is.na(patients))  
complete <- patients[complete.cases(patients), ]  
  
q <- quantile(patients$sbp, c(0.25, 0.75), na.rm = TRUE)  
iqr <- q[2] - q[1]  
lower <- q[1] - 1.5 * iqr  
upper <- q[2] + 1.5 * iqr  
flag <- patients$sbp < lower | patients$sbp > upper
```

Descriptive and Correlation Analysis

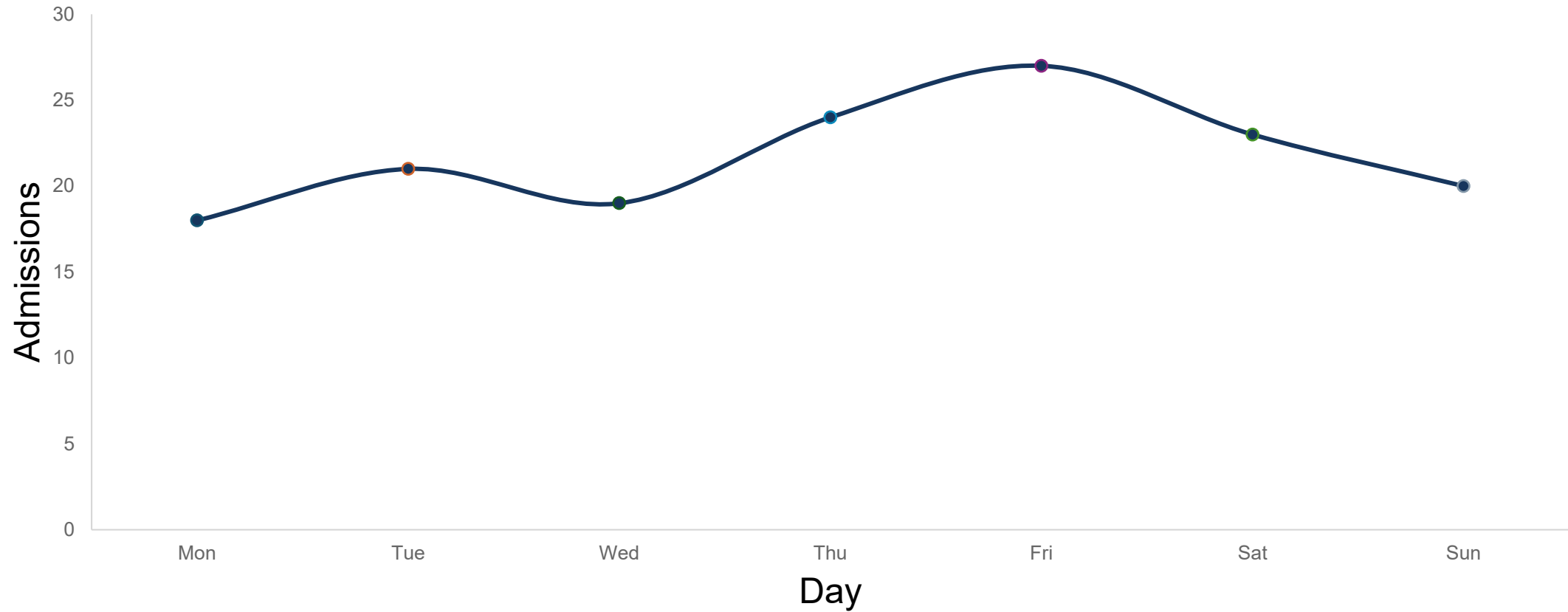
Descriptive analysis

- Mean, median, standard deviation
- Quantiles and range
- Counts and proportions
- Summaries by clinically relevant groups

Correlation analysis

- Scatterplots before correlation coefficients
- Pearson for linear association under suitable assumptions
- Spearman for monotonic association
- Association does not establish causation

Time Series Basics



```
plot()  
lines()
```

Part IV: Scientific Data Visualization

From graphical grammar to univariate, bivariate, and multivariate figures

A Scientific Figure Has a Job

- Show a distribution, comparison, relationship, composition, or change over time
- Choose a chart that matches variable types
- Label axes with units and define abbreviations
- Show uncertainty when estimating group effects
- Use color to encode information, not decoration
- Write a caption that allows the figure to stand alone

The ggplot2 Grammar

Foundation

- Data: the source table
- Aesthetics: x, y, color, fill, size
- Geometry: points, lines, boxes, bars

Additional layers

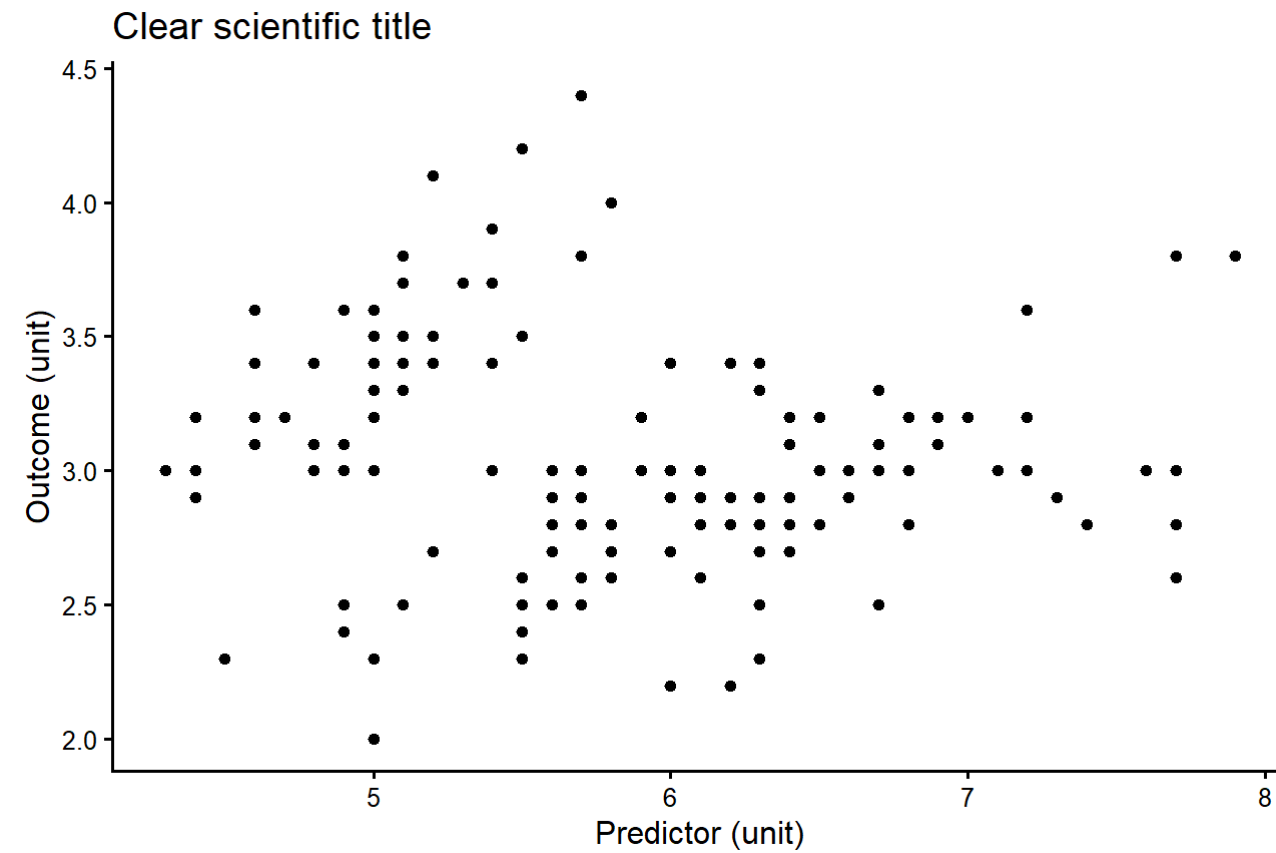
- Statistics: counts, smoothers, summaries
- Scales: axes, colors, labels
- Coordinates, facets, and themes

A Reusable ggplot Pattern

```
ggplot(data, aes(x = variable_x, y = variable_y)) +  
  geom_point() +  
  labs(  
    title = "Clear scientific title",  
    x = "Predictor (unit)",  
    y = "Outcome (unit)"  
  ) +  
  theme_classic(base_size = 12)
```

Themes, Labels, and Export

```
p <- p +  
  scale_color_viridis_d() +  
  theme_classic(base_size = 12) +  
  theme(legend.position = "right")  
  
ggsave(  
  "figures/toothgrowth.png", p,  
  width = 7, height = 5, dpi = 300  
)
```



Common Built-in Datasets in R

The datasets package provides more than 100 datasets for learning, demonstration, and testing

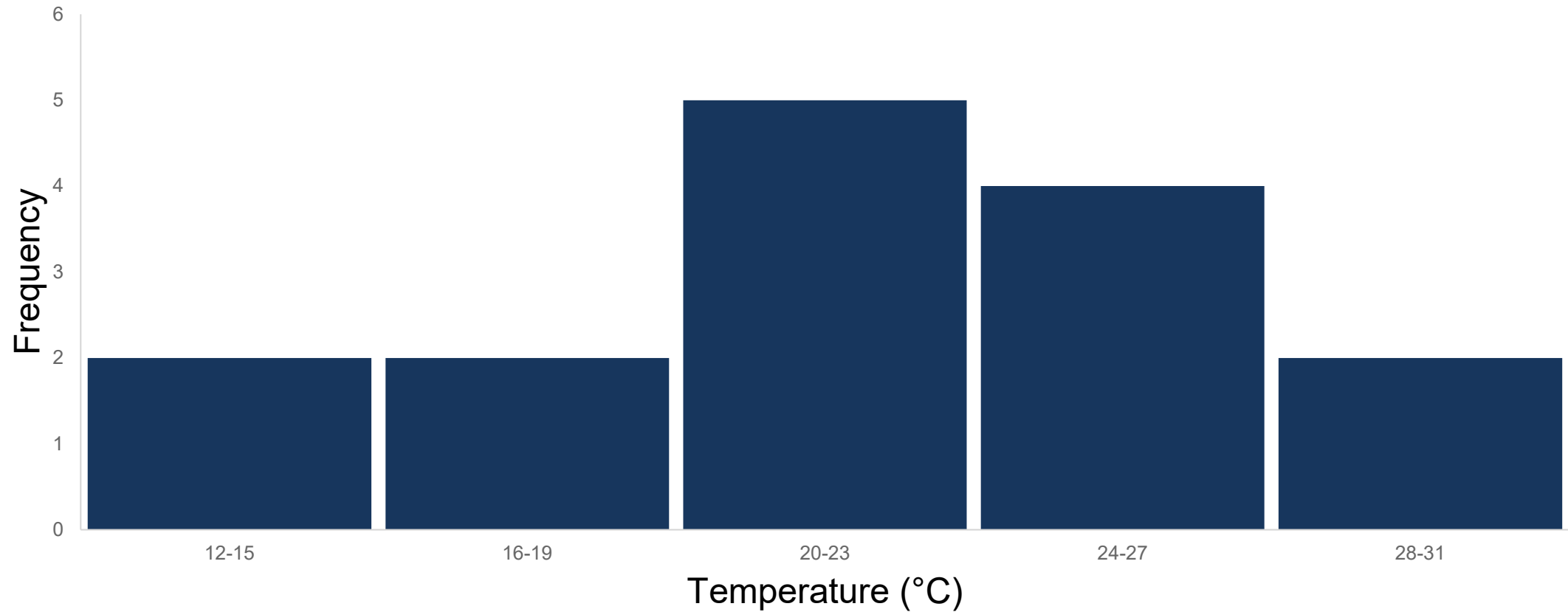
Dataset	Content	Typical applications
iris	Sepal and petal measurements with species labels	Classification, clustering, and introductory visualization
mtcars	Fuel economy, horsepower, weight, and design variables	Regression analysis and multivariable plots
airquality	Daily New York air quality and weather measurements	Time series and handling missing values (NA)
faithful	Old Faithful eruption duration and waiting time	Distribution patterns and bivariate relationships
ToothGrowth	Vitamin C dose, supplement type, and tooth growth	Group comparisons and analysis of variance
USArrests	State-level violent-crime arrest rates and urbanization	Clustering and principal component analysis

Common Built-in Datasets in ggplot2

Six datasets cover common lessons in statistical graphics and data preparation

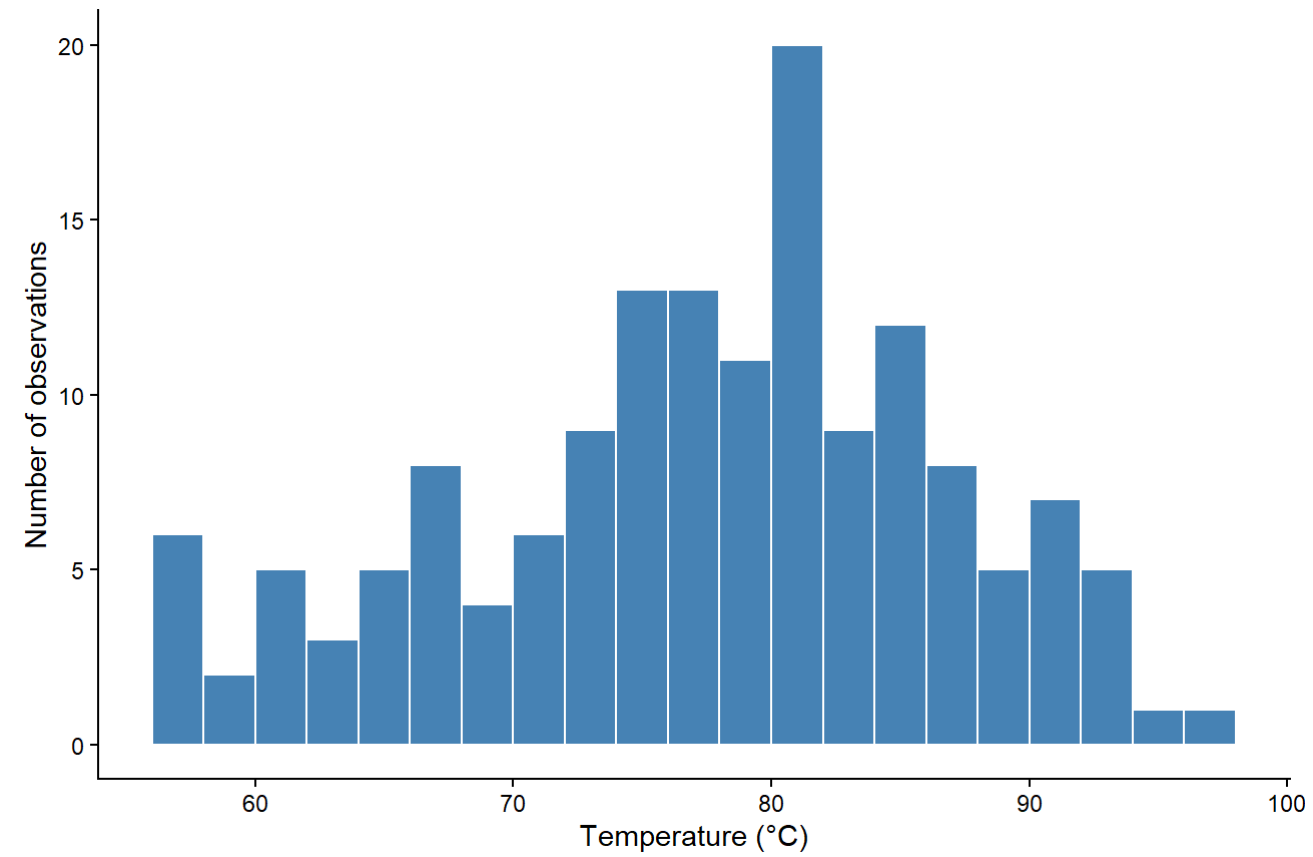
Dataset	Dimensions	Typical content	Typical applications
mpg	234 × 11	Vehicle fuel economy	Scatter plots, grouped box plots, faceting
diamonds	53,940 × 10	Diamond prices and attributes	Histograms, bar charts, alpha, geom_smooth()
economics	574 × 6	US economic time series	Line plots and time-series visualization
msleep	83 × 11	Mammalian sleep and body traits	Bubble plots and missing-value cleaning
midwest	437 × 28	County demographics in the US Midwest	Multivariable scatter plots and density plots
txhousing	8,602 × 9	Monthly Texas housing sales	Faceted line plots across cities

Histogram: One Numeric Variable

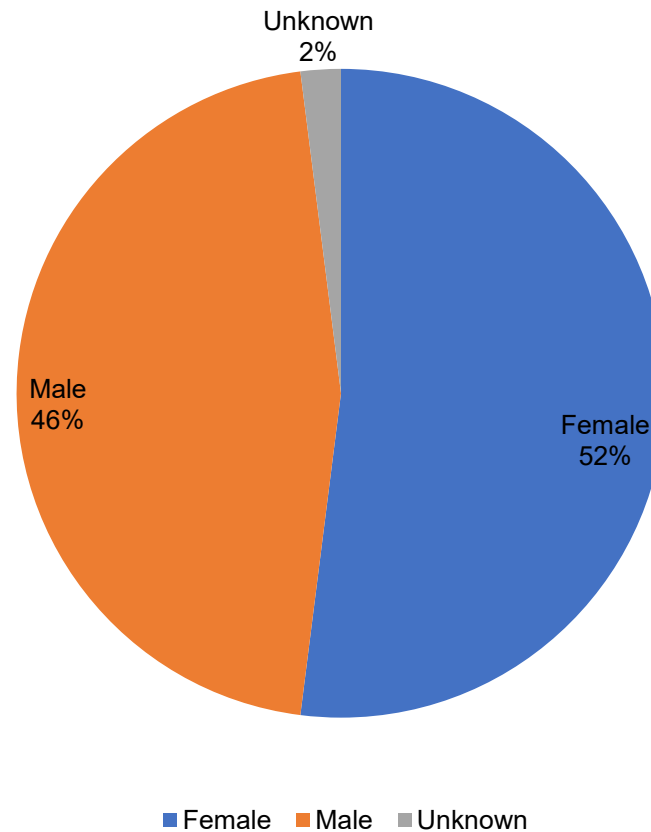


Histogram in ggplot2

```
ggplot(airquality, aes(x = Temp)) +  
  geom_histogram(  
    binwidth = 2,  
    boundary = 0,  
    color = "white",  
    fill = "steelblue"  
  ) +  
  labs(  
    x = "Temperature (°C)",  
    y = "Number of observations"  
  ) +  
  theme_classic()
```



Pie Charts and Composition



```
geom_bar() + coord_polar(theta = "y")
```

Boxplots, Density Plots, and Violin Plots

Boxplots

- Boxplot: median, quartiles, and flagged outliers
- Compact for comparing several groups
- Can hide multimodality and sample size

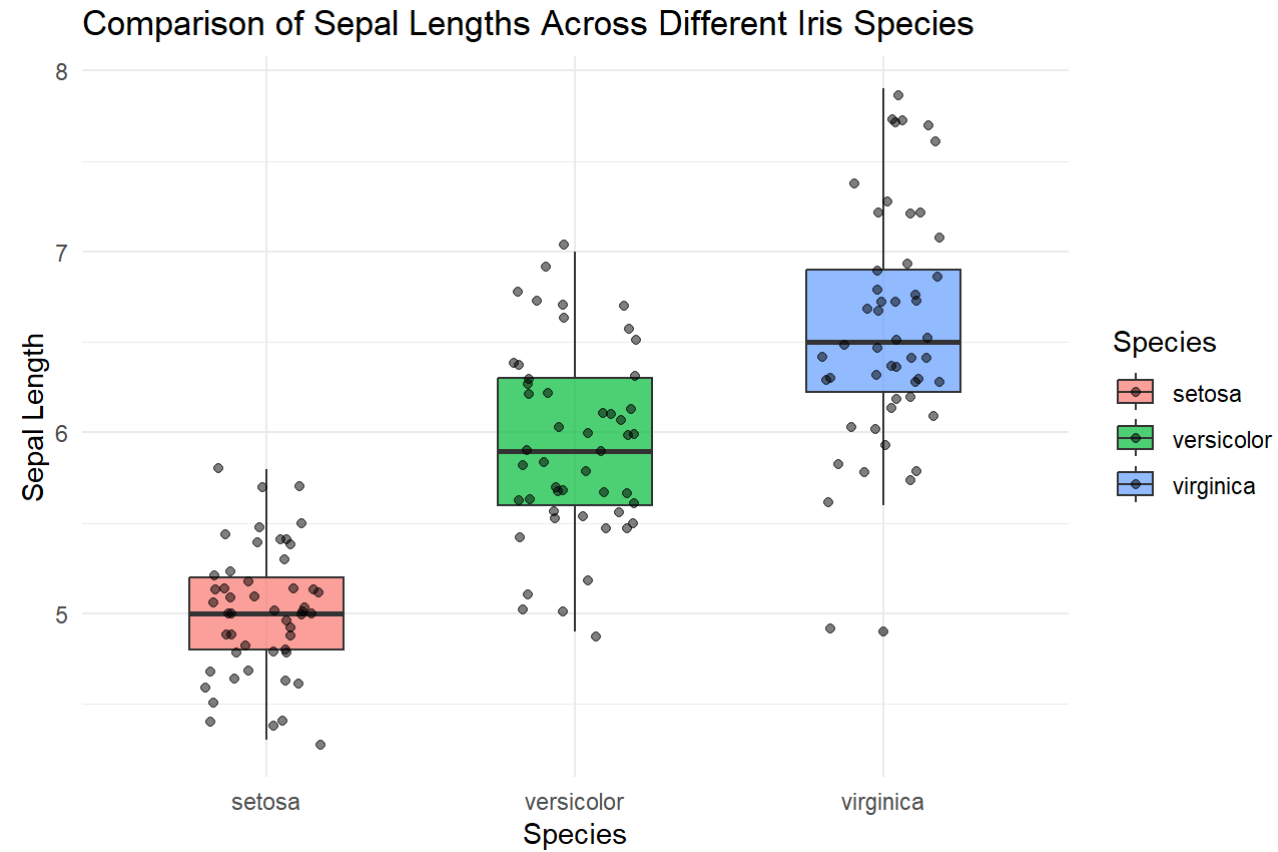
Density and violin plots

- Density and violin plots show distribution shape
- Bandwidth controls smoothness
- Add raw points or sample-size labels when possible

Boxplots in ggplot2

```
ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +  
  geom_boxplot(alpha = 0.7, width = 0.5) +           # boxplot  
  geom_jitter(width = 0.2, size = 1.5, alpha = 0.5) + # add jittered points  
  theme_minimal() +                                 # use a minimal theme  
  labs(title = "Comparison of Sepal Lengths Across Different Iris Species",  
        x = "Species",  
        y = "Sepal Length")
```

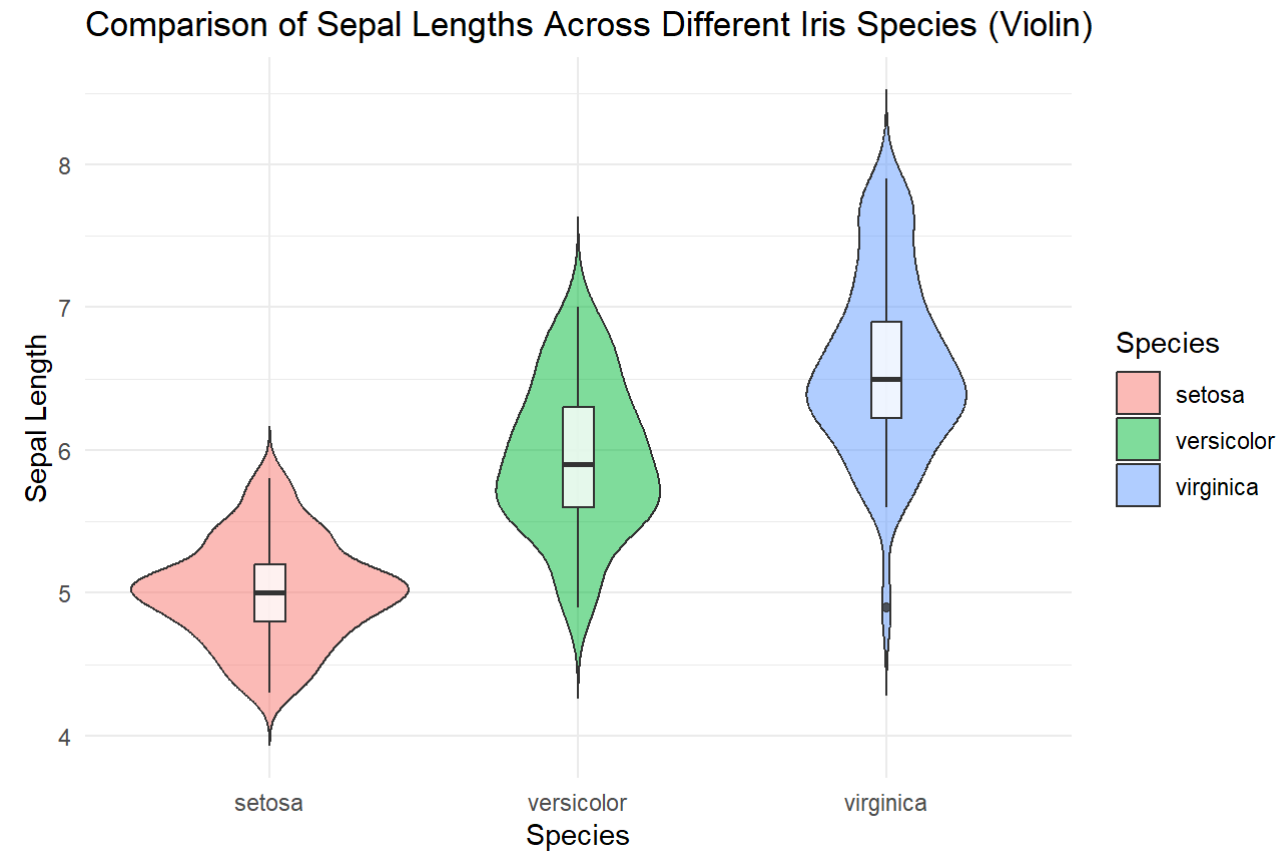
Boxplots



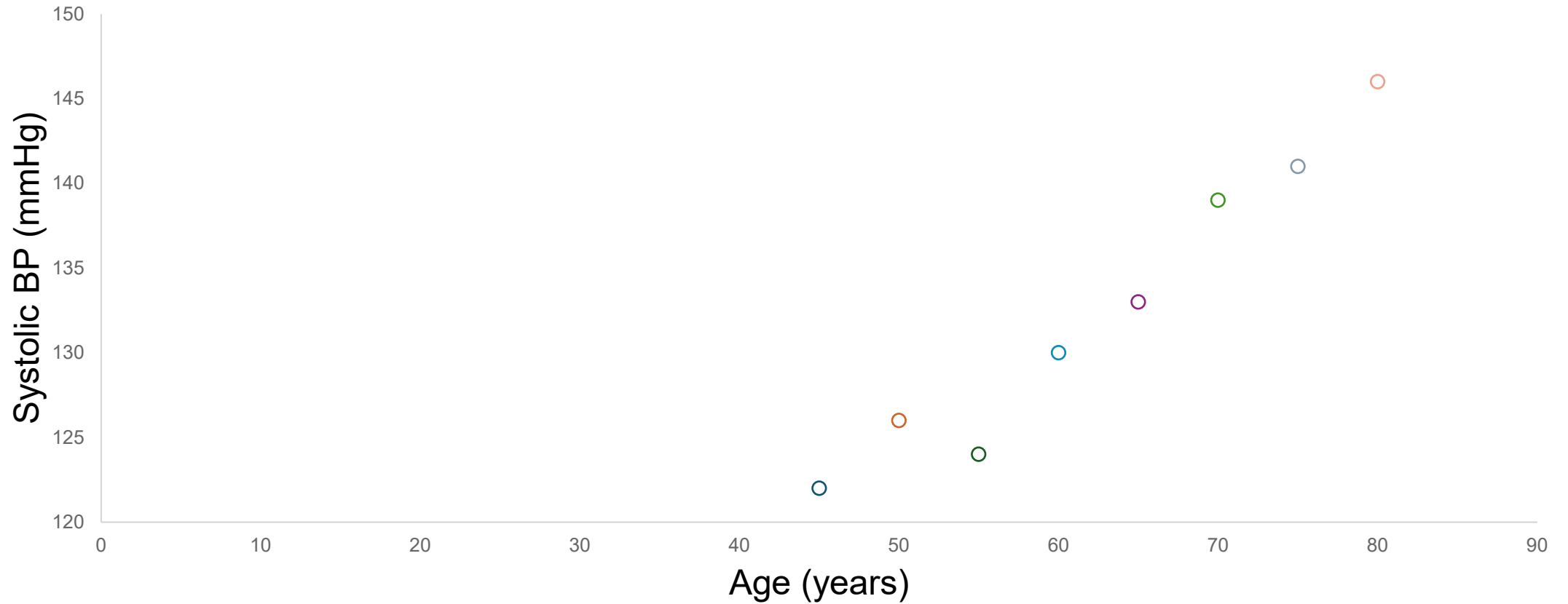
Violin Plots in ggplot2

```
ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +  
  geom_violin(alpha = 0.5, trim = FALSE) +      # Violin plot, trim=False keeps tails  
  geom_boxplot(width = 0.1, fill = "white", outlier.size = 1.5, alpha = 0.8) +  
  theme_minimal() +                             # use a minimal theme  
  labs(title = "Comparison of Sepal Lengths Across Different Iris Species (Violin)",  
        x = "Species",  
        y = "Sepal Length")
```

Violin Plots



Scatterplots

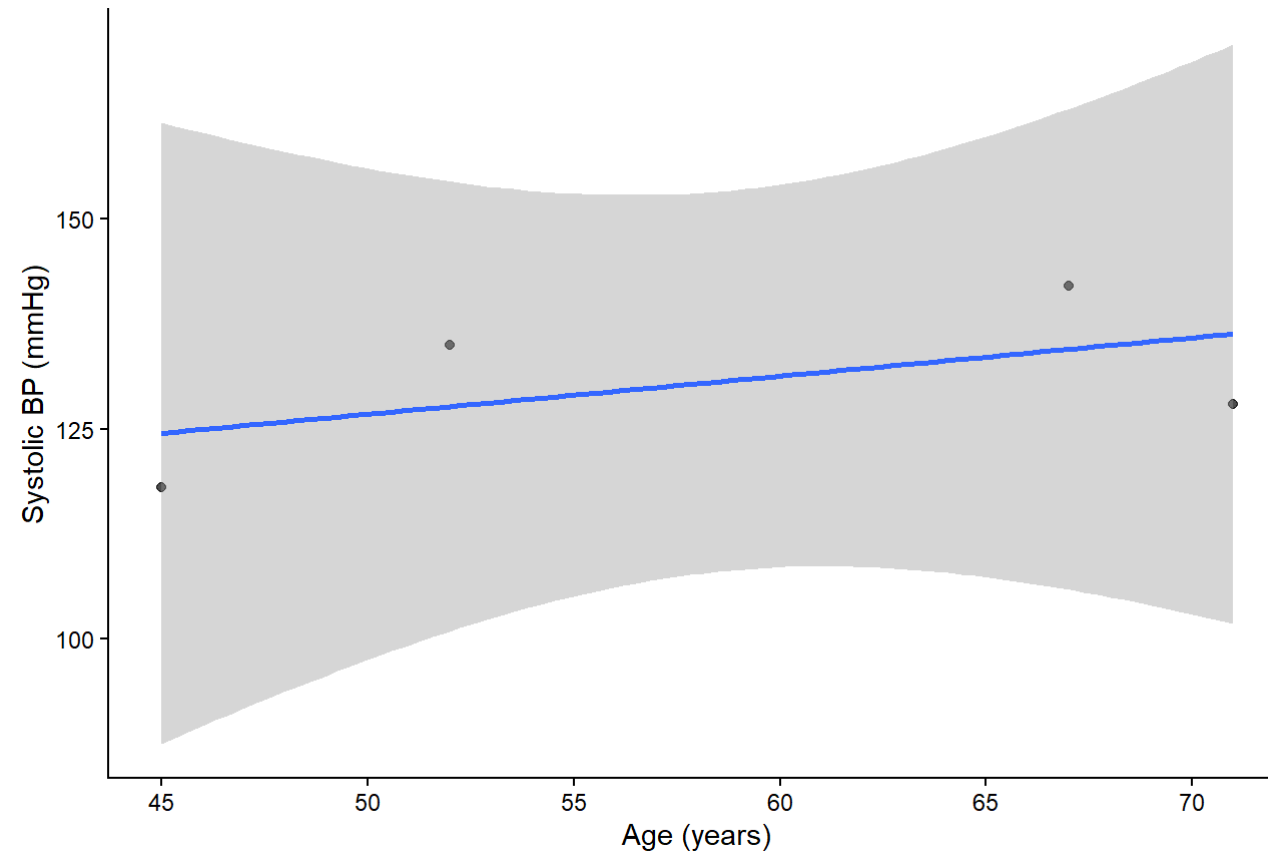


`geom_point()`

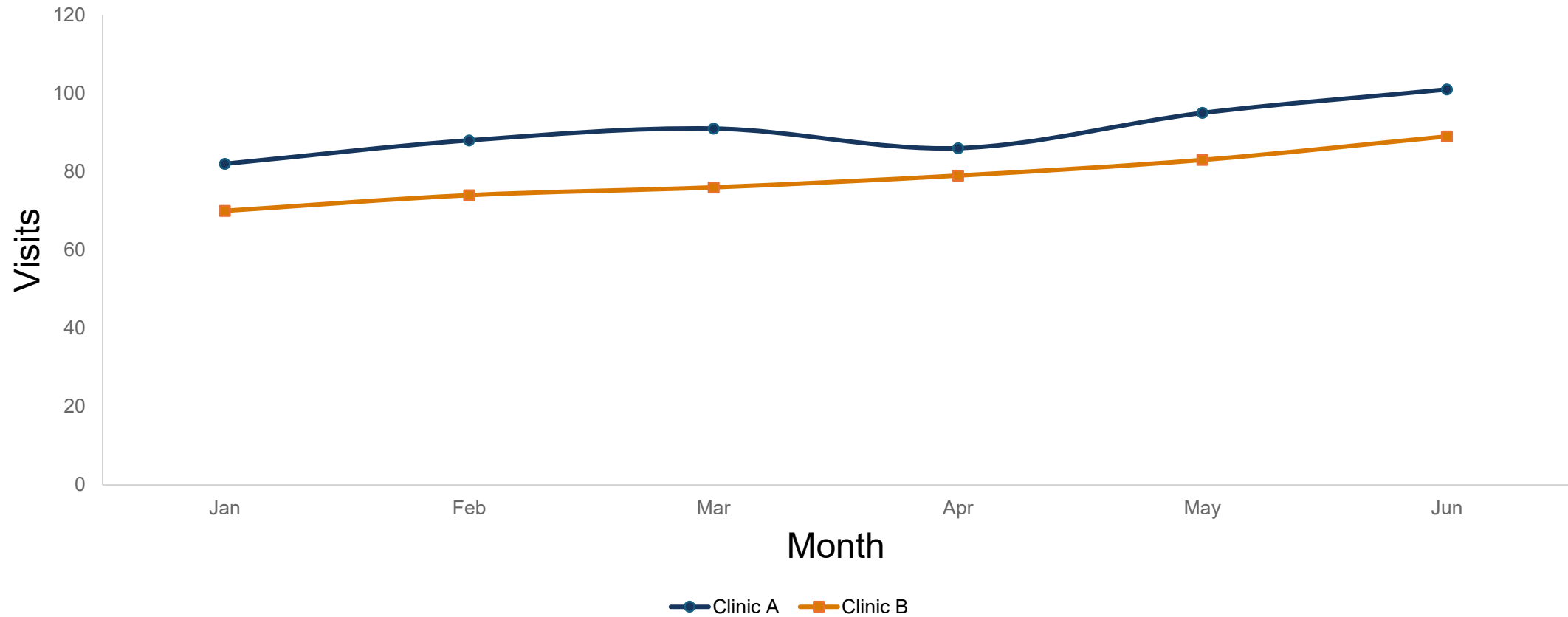
Regression Lines Need Context

```
ggplot(patients, aes(x = age, y = sbp)) +  
  geom_point(alpha = 0.7) +  
  geom_smooth(method = "lm", se = TRUE) +  
  labs(  
    x = "Age (years)",  
    y = "Systolic BP (mmHg)"  
  ) +  
  theme_classic()
```

Regression Lines

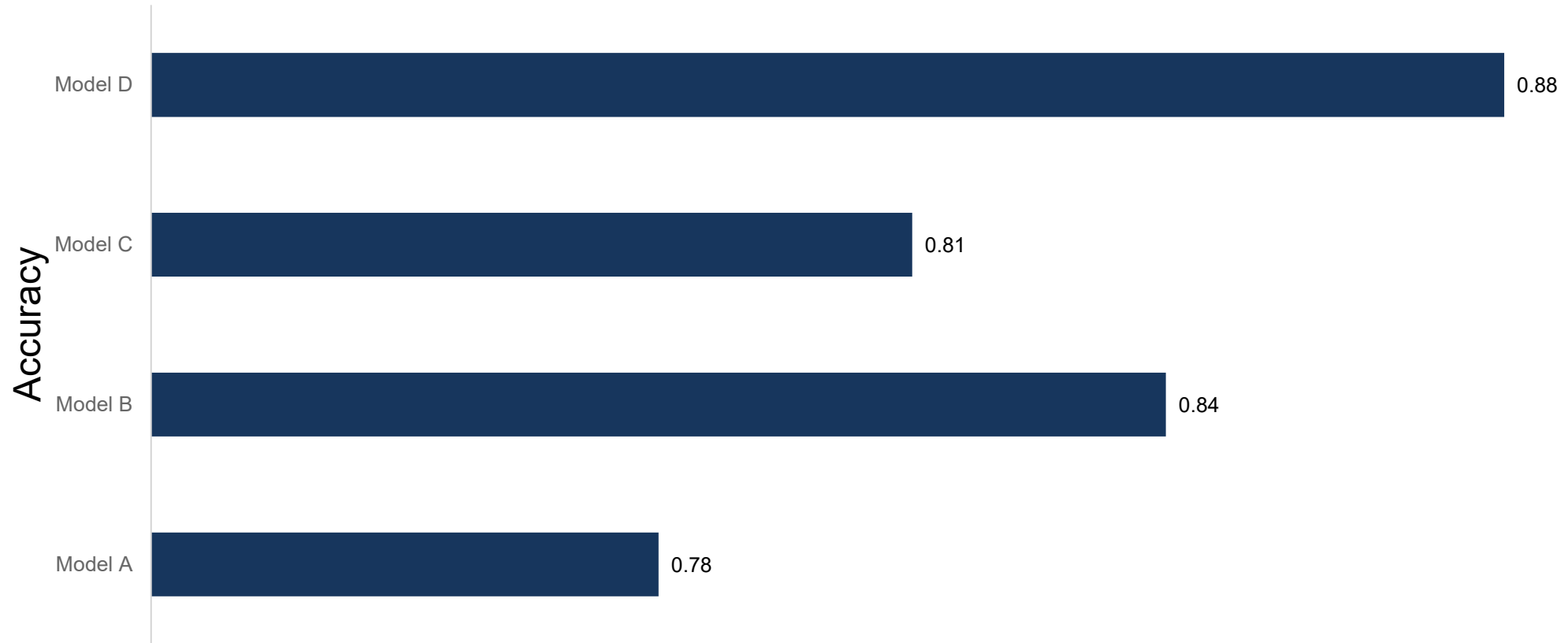


Line and Area Charts



```
geom_line()
```

Column and Bar Charts



geom_bar()

Heatmaps and Density Comparisons

Heatmaps

- Heatmap: values encoded by color in a matrix
- Useful for correlations, expression data, or repeated measurements
- Use a perceptually ordered and colorblind-safe scale

Density comparisons

- Overlaid density curves compare distributions
- Limit the number of groups
- Direct labels can be clearer than a distant legend

Heatmaps in ggplot2

```
# Set the random seed for reproducibility
set.seed(110)

# Generate a 10 × 10 matrix
data <- matrix(
  rnorm(100, mean = 0, sd = 5),
  nrow = 10,
  ncol = 10
)

# Set row and column names
colnames(data) <- paste0("col", 1:10)
rownames(data) <- paste0("row", 1:10)

# Convert the matrix to long-format data
data_df <- as.data.frame(as.table(data))
colnames(data_df) <- c("Row", "Column", "Value")

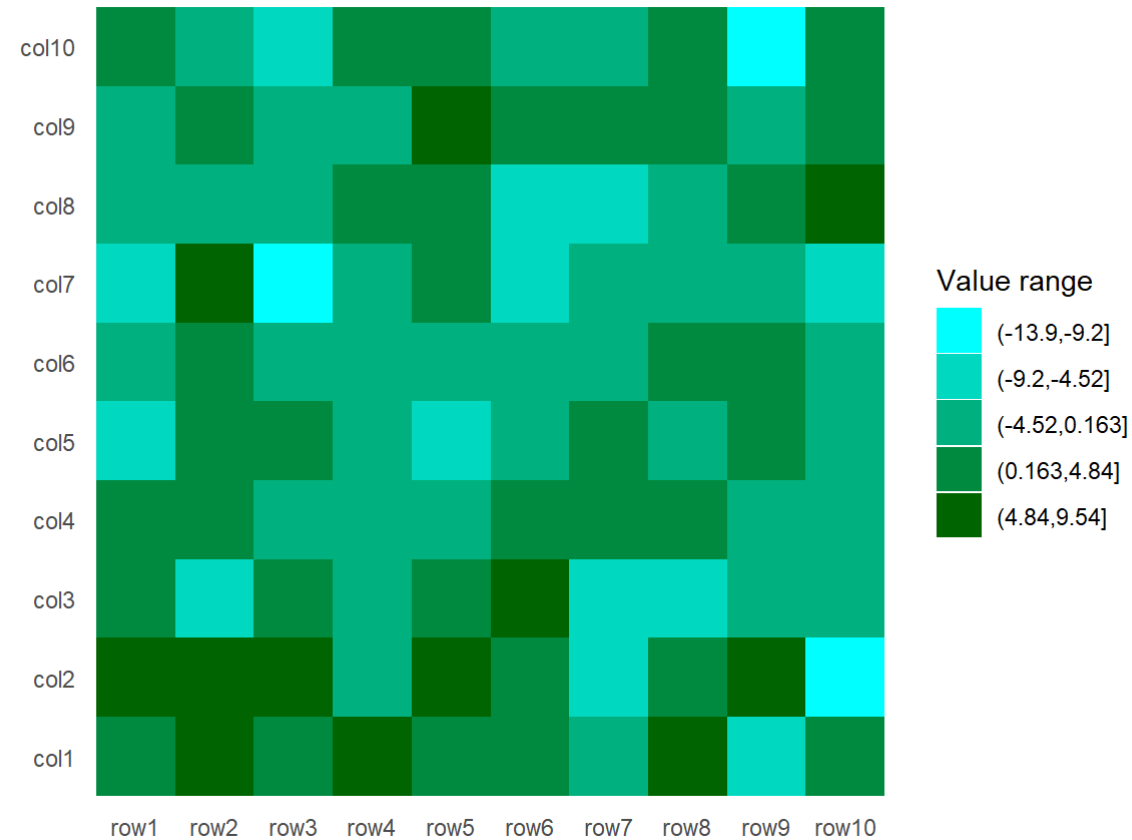
# Create a custom color palette
my_colors <- colorRampPalette(
  c("cyan", "darkgreen")
)
```

Heatmaps in ggplot2 (cont.)

```
# Divide the values into five intervals
data_df$Value_group <- cut(
  data_df$Value,
  breaks = 5
)

# Create the heatmap
ggplot( data_df, aes( x = Row, y = Column, fill = Value_group ) ) +
  geom_tile() +
  scale_fill_manual( values = my_colors(5), name = "Value range" ) +
  labs( x = NULL, y = NULL ) + theme_minimal() +
  theme( panel.grid = element_blank() ) +
  coord_fixed(ratio = 1)
```

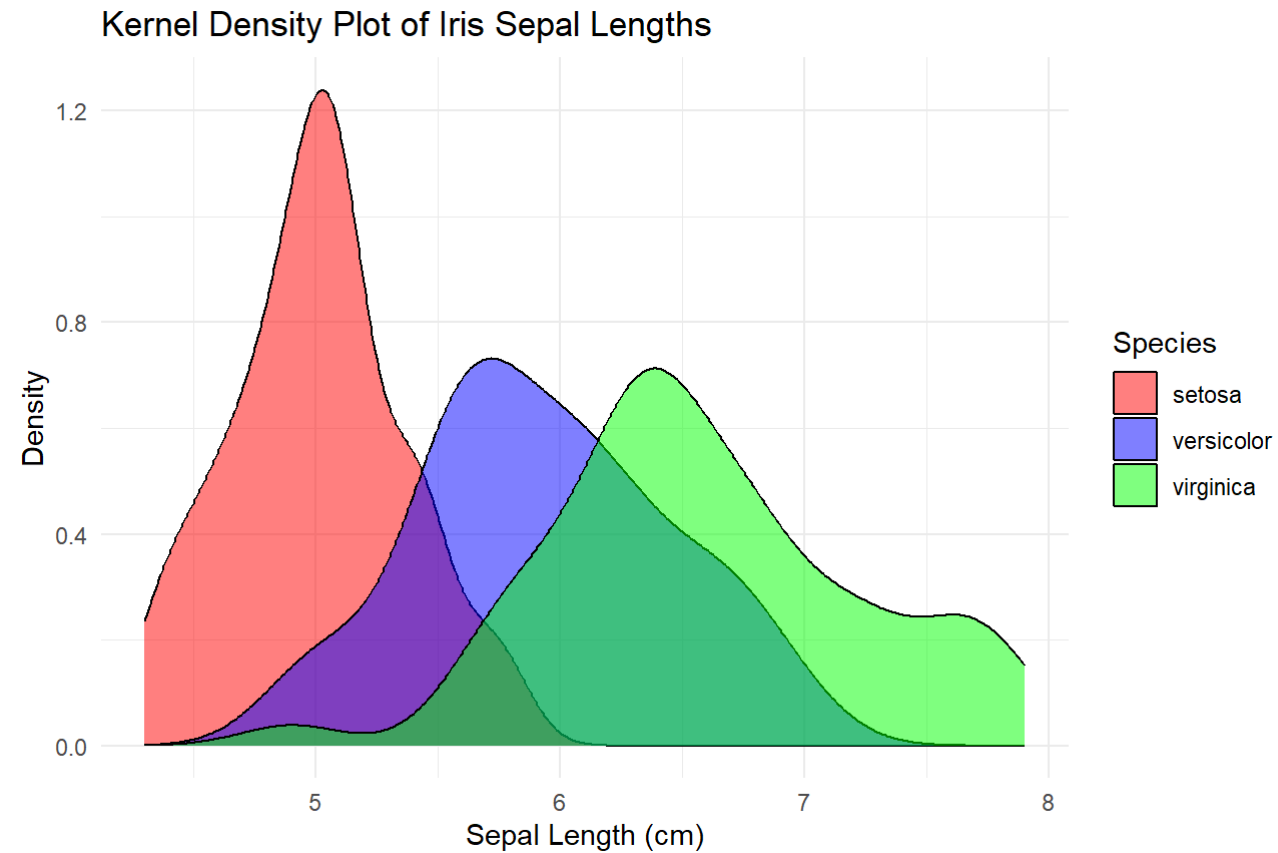
Heatmaps



Kernal Density Plot in ggplot2

```
ggplot( data = iris, aes(x = Sepal.Length, fill = Species) ) +  
  geom_density(alpha = 0.5) +  
  labs( title = "Kernel Density Plot of Iris Sepal Lengths",  
        x = "Sepal Length (cm)",  
        y = "Density" ) +  
  scale_fill_manual( values = c( "setosa" = "red", "versicolor" = "blue",  
    "virginica" = "green" ) ) +  
  theme_minimal()
```

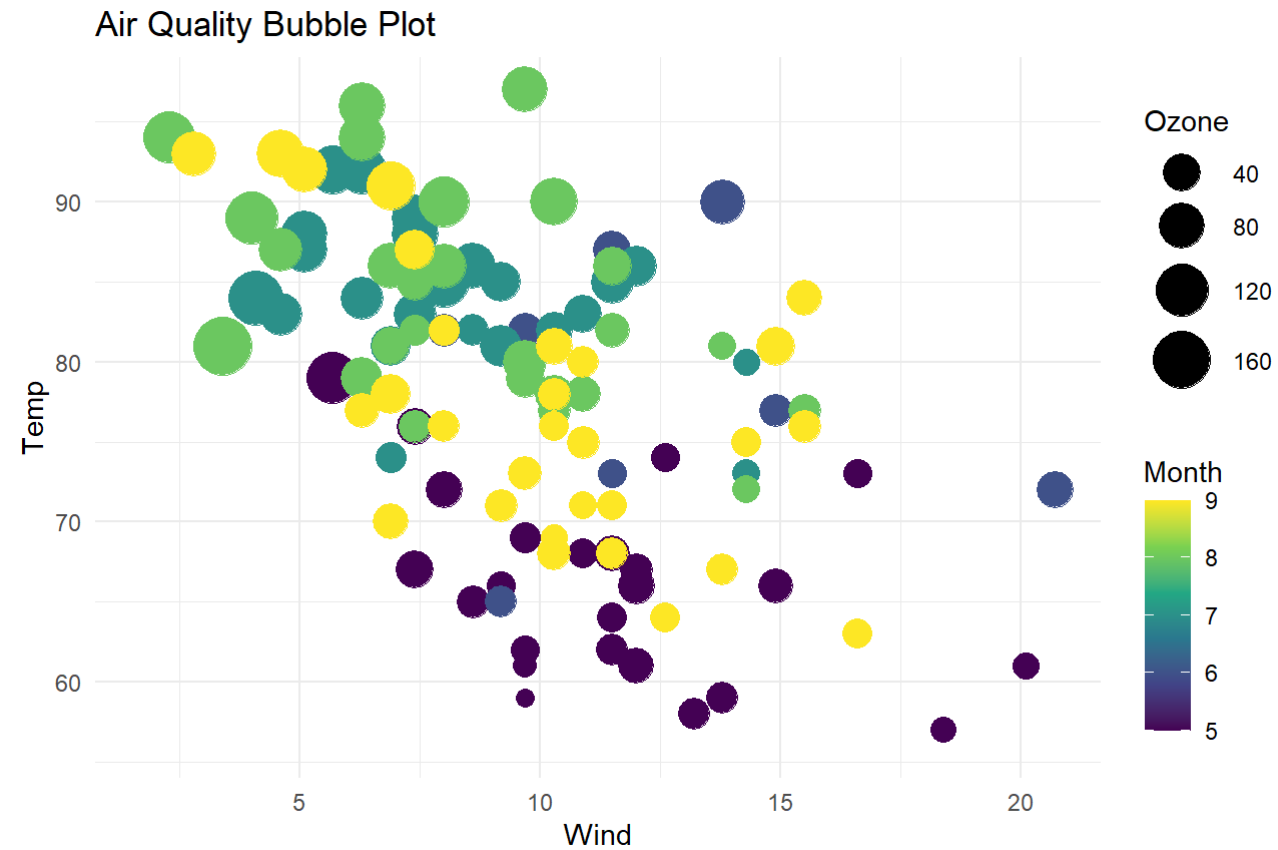
Kernal Density Plot



Bubble Plot in ggplot2

```
ggplot(airquality, aes(x = Wind, y = Temp,  
size = Ozone, color = Month)) +  
  geom_point(na.rm = TRUE) +  
  scale_size_continuous(range = c(3,10)) +  
  scale_color_viridis_c() +  
  labs( title = "Air Quality Bubble Plot",  
        x = "Wind", y = "Temp", size = "Ozone") +  
  theme_minimal()
```

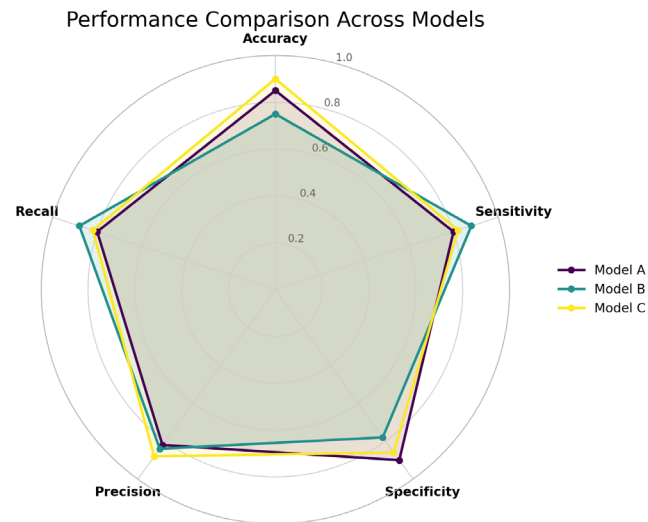
Bubble Plots



Radar and Network Graphics

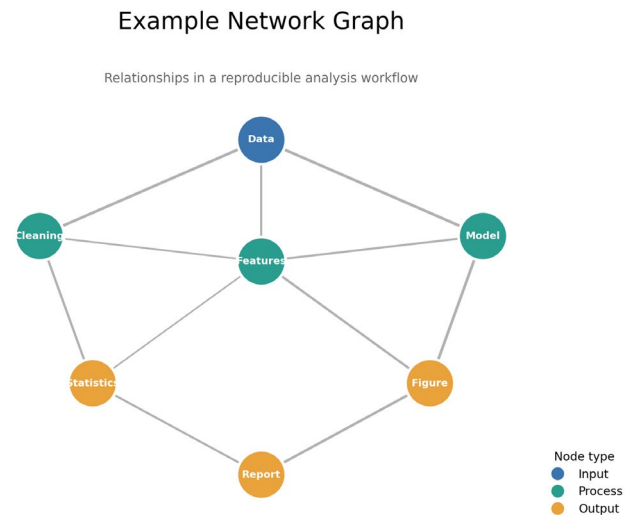
Radar charts

- Radar charts compare profiles across several standardized dimensions
- Axes must share meaningful scales
- Limit the number of profiles and variables

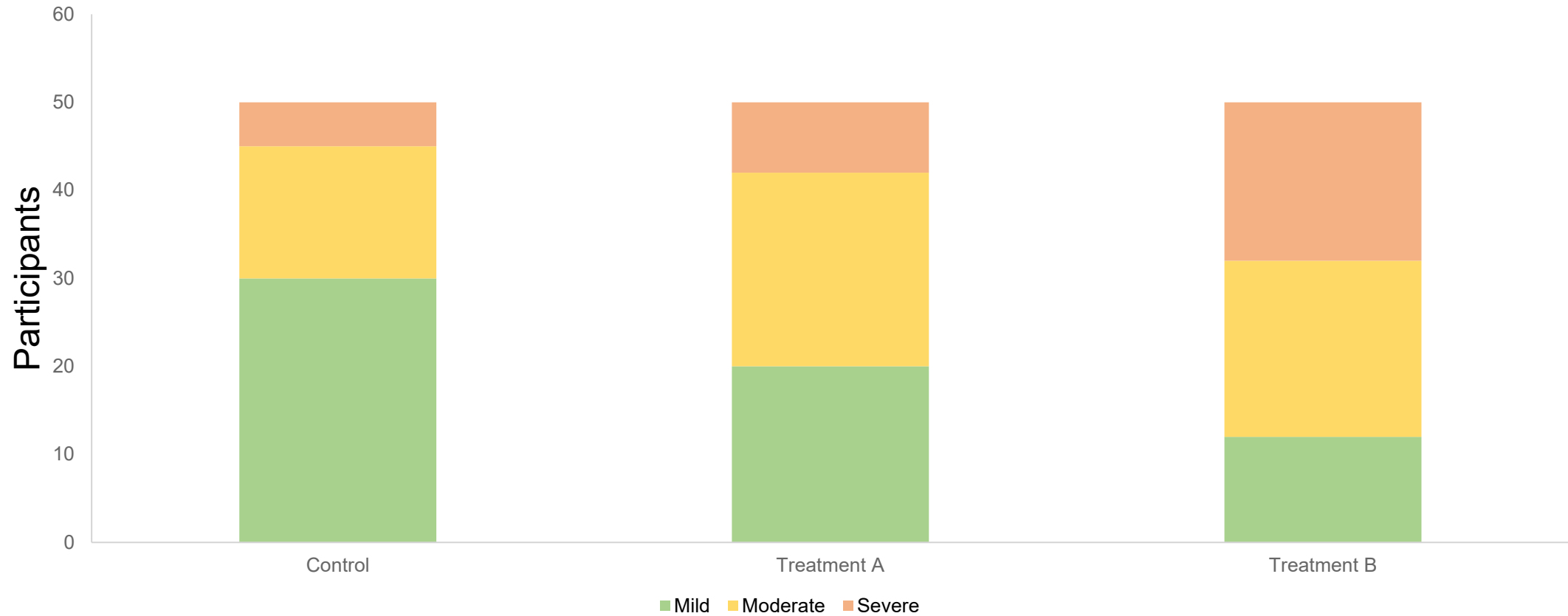


Network graphs

- Network graphs show nodes and relationships
- Layout affects perception
- Define edge meaning, direction, and weight



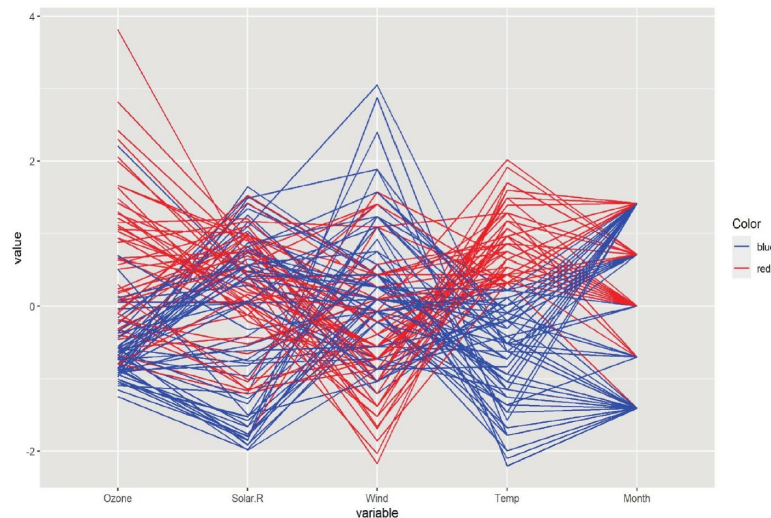
Stacked Columns



Parallel Coordinates and Matrix Plots

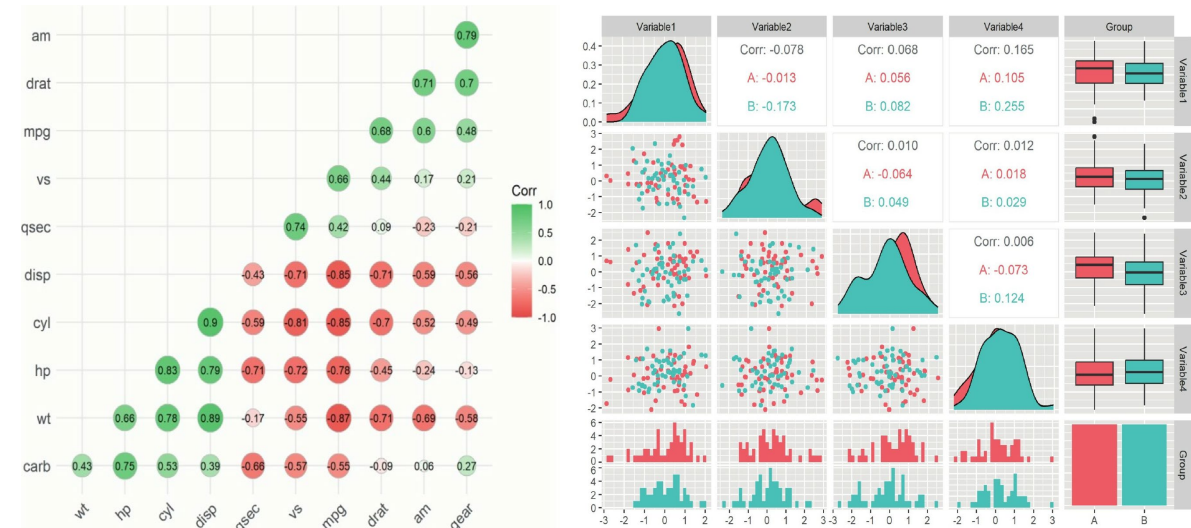
Parallel coordinates

- Parallel coordinates display multivariable profiles
- Standardize variables with different units
- Use transparency or clustering for many observations



Matrix plots

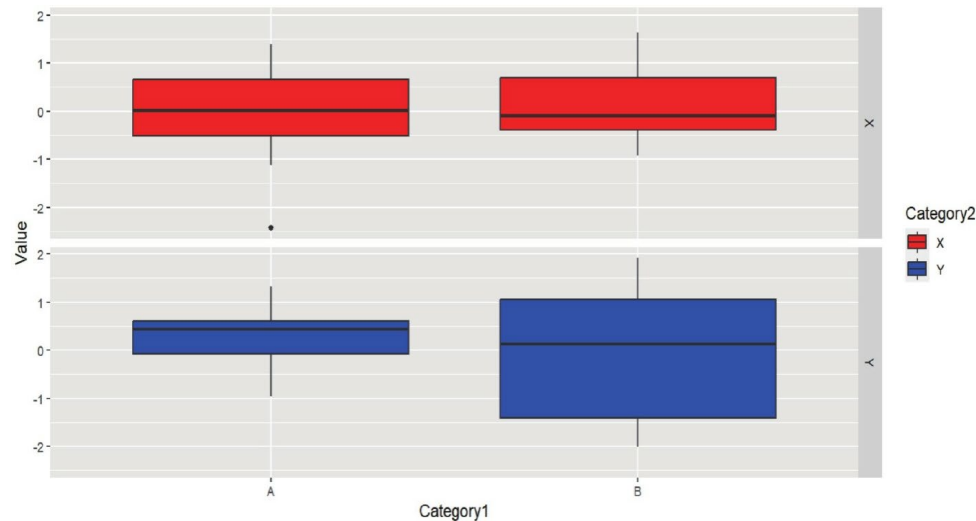
- Correlation matrices summarize pairwise association
- Scatterplot matrices show form and outliers
- Order variables to reveal structure



Faceting and Ternary Plots

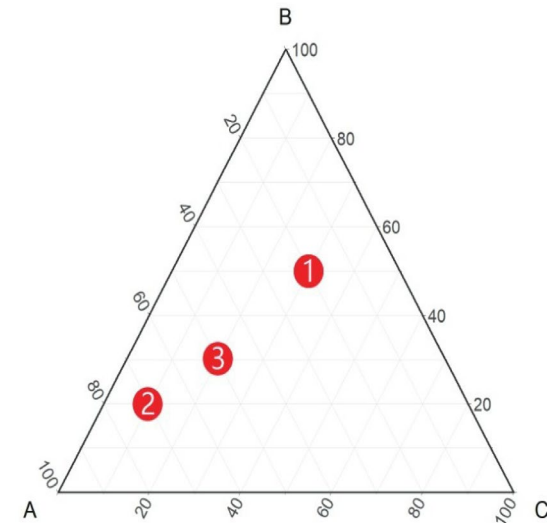
Faceting

- Faceting repeats the same plot across subgroups
- Keep scales comparable when direct comparison matters
- Use facet labels that carry clinical meaning



Ternary plots

- Ternary plots represent three parts that sum to a constant
- Common in composition data
- Every point reflects a trade-off among the three components



Part V: Integrated Practice

ToothGrowth, dplyr, ggplot2, and reproducible interpretation

The ToothGrowth Dataset

Study context

- 60 guinea pigs
- Outcome: tooth length
- Supplement: orange juice or vitamin C
- Dose: 0.5, 1, or 2 mg/day

Variables

- Each row represents one animal
- len: tooth length
- supp: OJ or VC
- dose: daily dose

Packages and First Inspection

```
library(dplyr)
library(ggplot2)

tg <- ToothGrowth
glimpse(tg)
head(tg)
dim(tg)
summary(tg)
count(tg, supp, dose)
```

The dplyr Grammar

- `select()` keeps or reorders columns
- `filter()` keeps rows that satisfy conditions
- `mutate()` creates or changes variables
- `arrange()` sorts rows
- `summarise()` reduces observations to summary values
- `group_by()` repeats a summary within groups

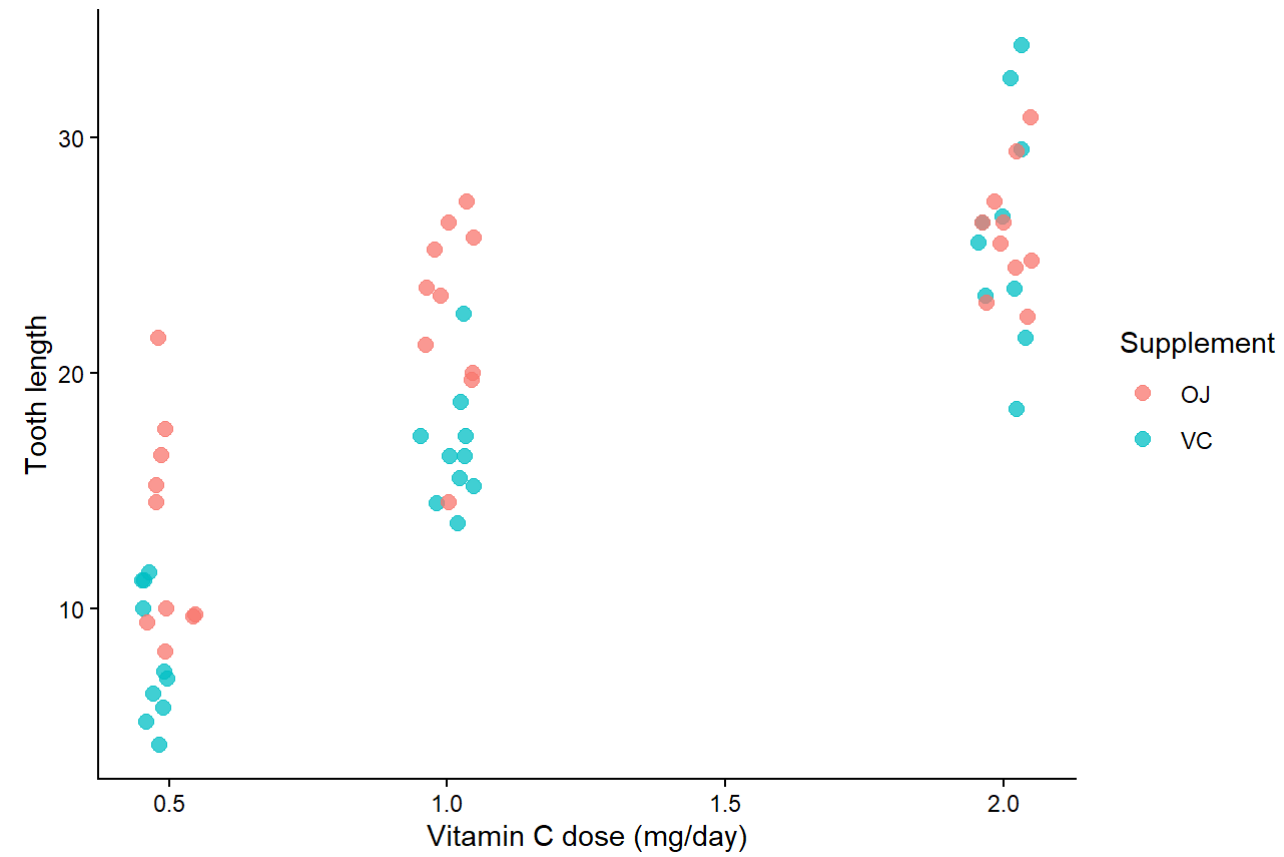
A dplyr Pipeline

```
summary_tg <- tg |>
  group_by(supp, dose) |>
  summarise(
    n = n(),
    mean_len = mean(len),
    sd_len = sd(len),
    .groups = "drop"
  ) |>
  arrange(supp, dose)
```

```
summary_tg
```

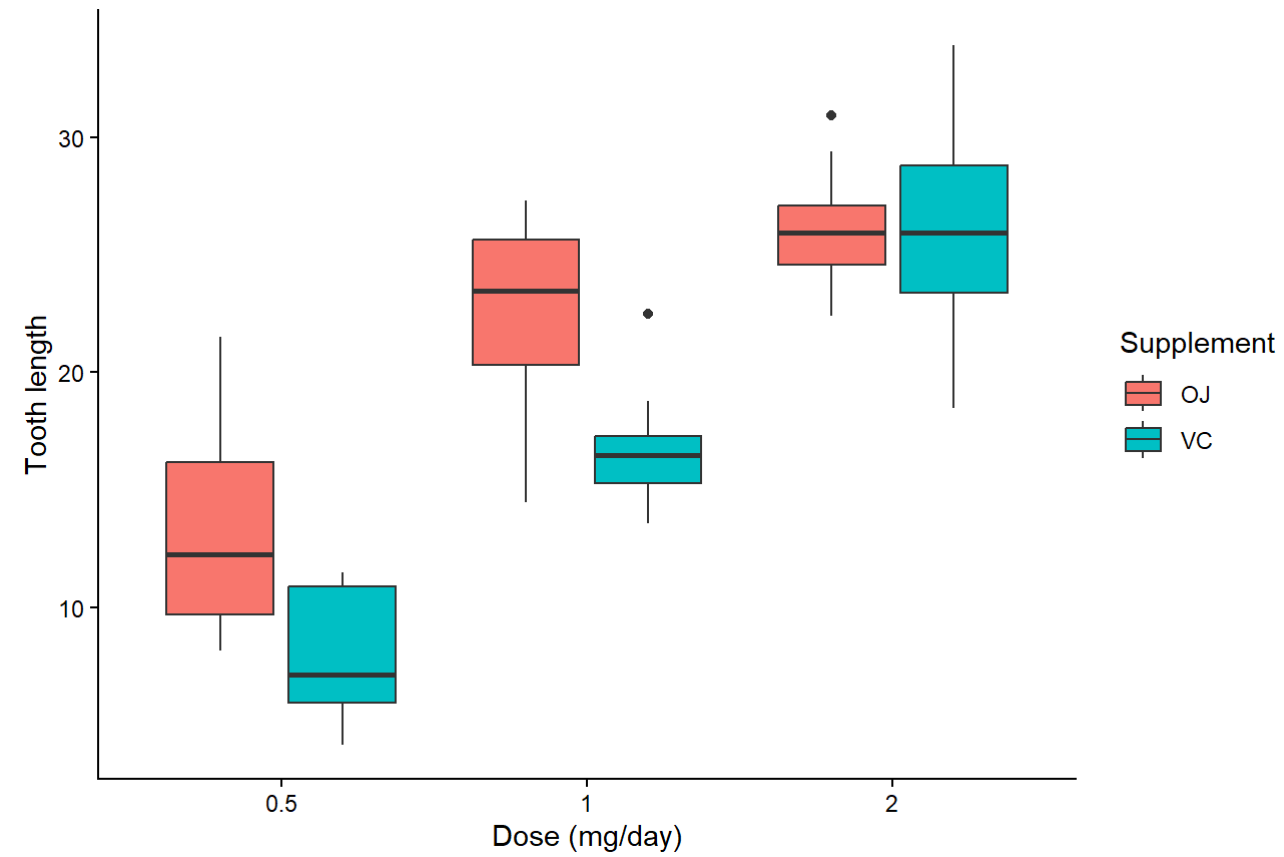
A First ggplot

```
ggplot(tg, aes(x = dose, y = len, color = supp)) +  
  geom_point(  
    position = position_jitter(width = 0.05),  
    alpha = 0.75, size = 2.5  
  ) +  
  labs(  
    x = "Vitamin C dose (mg/day)",  
    y = "Tooth length",  
    color = "Supplement"  
  ) +  
  theme_classic()
```

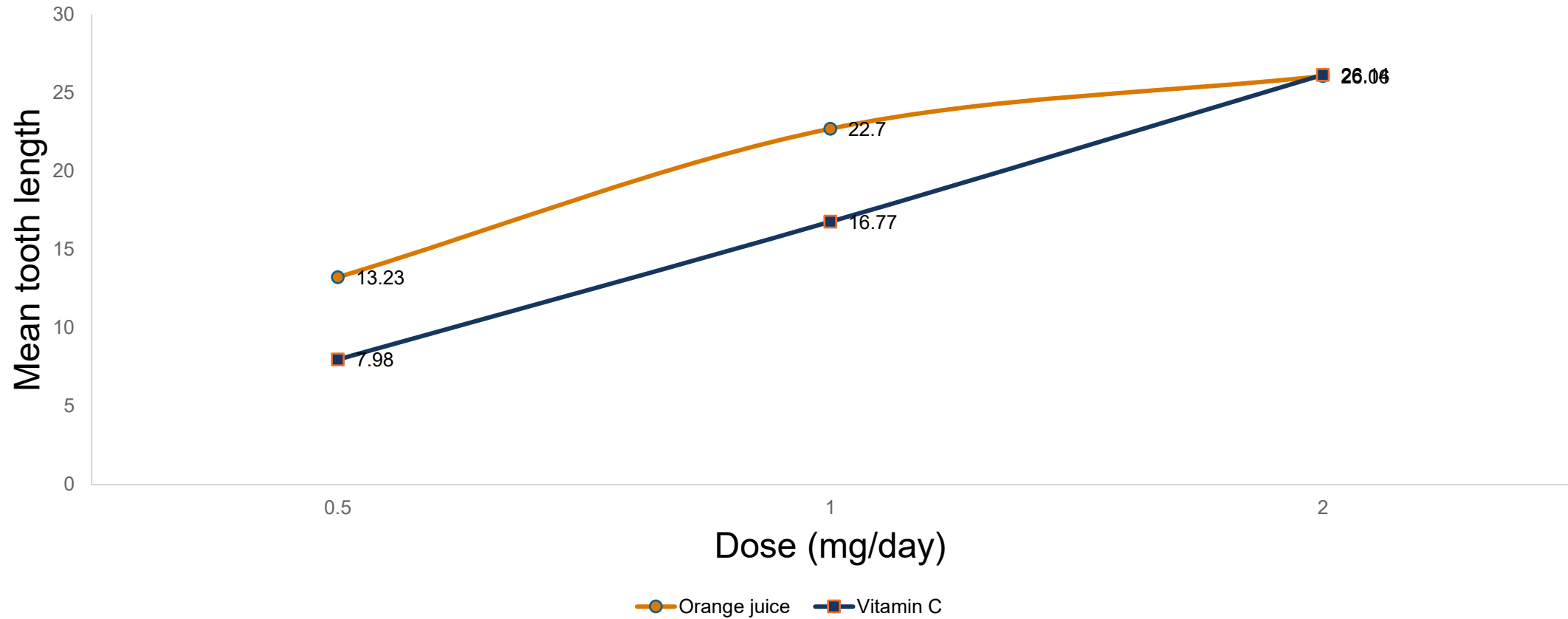


Grouped Distributions

```
ggplot(tg, aes(
  x = factor(dose), y = len, fill = supp
)) +
  geom_boxplot(
    position = position_dodge(width = 0.8),
    width = 0.7
  ) +
  labs(
    x = "Dose (mg/day)",
    y = "Tooth length", fill = "Supplement"
  ) +
  theme_classic()
```



Mean Tooth Length by Dose



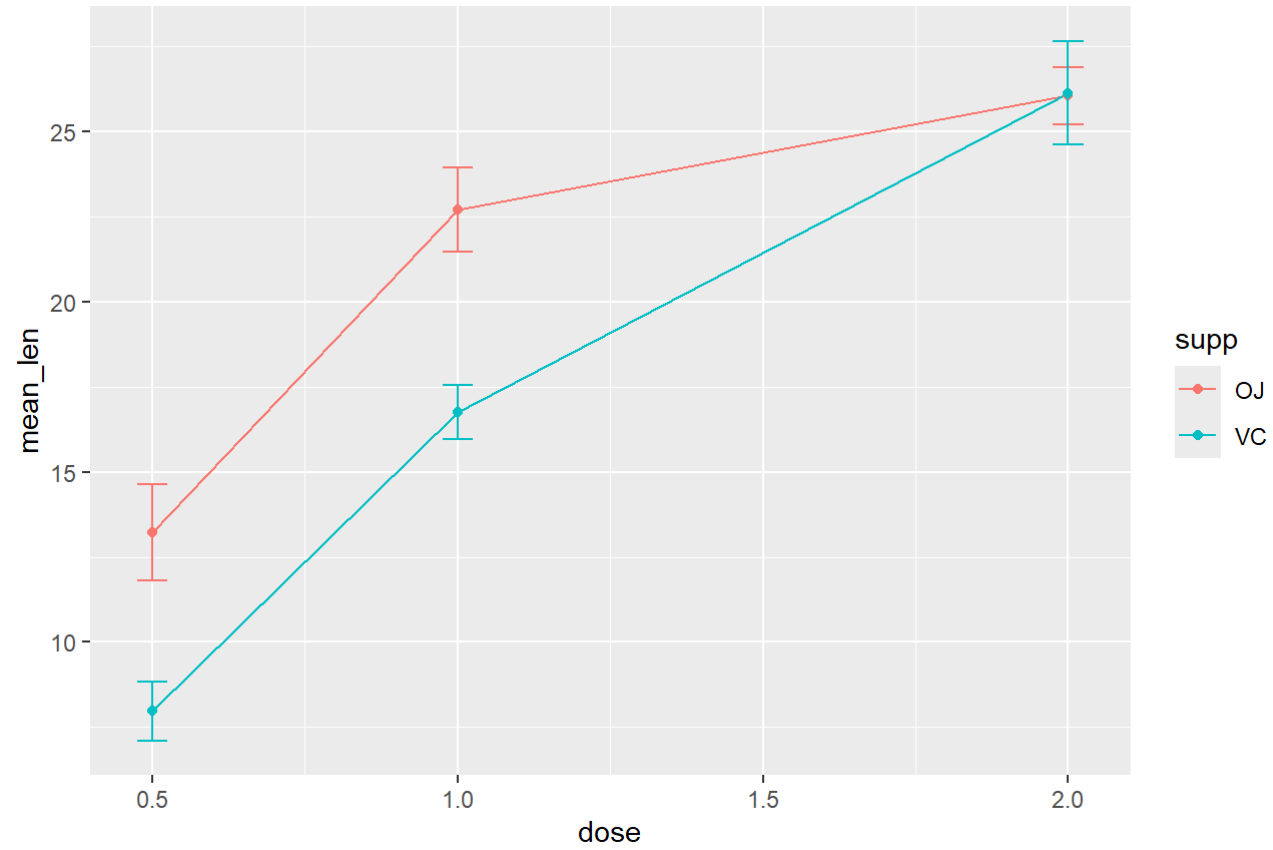
Integrated Practice

- Inspect ToothGrowth and verify the sample size
- Calculate mean and standard deviation by supplement and dose
- Create a grouped distribution plot
- Create a mean profile plot with uncertainty
- Write two descriptive sentences
- Save the final figure with ggsave()

Practice Answer

```
summary_tg <- tg |>
  group_by(supp, dose) |>
  summarise(
    n = n(),
    mean_len = mean(len),
    sd_len = sd(len),
    se_len = sd_len / sqrt(n),
    .groups = "drop"
  )

ggplot(summary_tg, aes(dose, mean_len, color = supp)) +
  geom_line() + geom_point() +
  geom_errorbar(aes(ymin = mean_len - se_len,
                    ymax = mean_len + se_len), width = 0.05)
```



Common Errors and Quick Checks

- object not found: spelling, capitalization, or execution order
- unexpected symbol: quotes, commas, or parentheses
- subscript out of bounds: requested position does not exist
- NA result: inspect missing values and `na.rm`
- package not found: install once, then load each session
- plot looks wrong: inspect mappings, grouping, scales, and data types

A Reproducible Analysis Script

```
# 1. Purpose and research question  
# 2. Load packages  
# 3. Import data  
# 4. Inspect and clean data  
# 5. Transform and summarize  
# 6. Create and save figures  
# 7. Record session information  
  
sessionInfo()
```

Key Takeaways

- R scripts make data analysis transparent and reproducible
- Data structures and types determine how functions behave
- Cleaning decisions require documentation and sensitivity checks
- The scientific question and variable types determine the graphic
- Show data, variation, uncertainty, labels, and units
- Use dplyr and ggplot2 to connect transformation with visualization

Questions

Practice with a small real dataset and adapt the workflow to your own research