Applied Statistics and Working with R

Techniques of Note

Lucas Liona

Table of contents

Introduction 2
- R as a Statistical Environment 2

Data Structures and Manipulation 2
- Working with Vectors 2
- Data Frames 3
- Using Factors for Categorical Data 4
- Tidyverse Style Data Manipulation 5

Statistical Methods 6
- Descriptive Statistics 6
- Hypothesis Testing 7
  - t-tests 7
  - ANOVA 7
- Correlation and Regression 8
  - Correlation Analysis 8
  - Linear Regression 8

Data Visualization 10
- Basic Graphics with ggplot2 10
- Visualizing Distributions 11

Case Studies 13
- Case Study 1: Die Tossing Experiment 13
- Case Study 2: The Central Limit Theorem 16
- Case Study 3: DEA Budget and Drug Deaths Analysis 19

Best Practices 23
- Code Style and Organization 23

Reproducible Research 23
Data Visualization Principles 24

Conclusion **24**

Introduction

This document provides a summary of key techniques and concepts for applied statistics using R. The examples demonstrate how to implement statistical methods, visualize data, and interpret results in a clear, reproducible manner. Following the principles of tidy data and the tidyverse philosophy, we'll show how to leverage R's strengths in data analysis and statistics.

R as a Statistical Environment

R is an open-source programming language specifically designed for statistical computing and graphics. As noted in Advanced R, R has several key advantages:

- It's free, open-source, and available on every major platform
- It contains a diverse set of statistical tools
- It has powerful visualization capabilities
- It has a large, active community of users and developers
- It's designed specifically for data analysis and statistics

Throughout this document, we'll apply these advantages to various statistical problems.

Data Structures and Manipulation

Working with Vectors

Vectors are the fundamental data structure in R, forming the building blocks for more complex structures. Let's create a few examples:

```r
# Creating numeric vectors
x <- c(4, 7, 9, 12, 15)
y <- seq(1, 10, by = 0.5)

# Character vectors
names <- c("Alice", "Bob", "Charlie", "David")

# Logical vectors
```

```r
is_large <- x > 10

# Accessing elements
x[3]  # Third element
```

```
[1] 9
```

```r
x[x > 10]  # Elements greater than 10
```

```
[1] 12 15
```

```r
names[is.element(names, c("Alice", "David"))]  # Elements in a specified set
```

```
[1] "Alice" "David"
```

Data Frames

Data frames are the most common way to store data in R. They're tabular (like a spreadsheet) with rows representing observations and columns representing variables.

```r
# Creating a data frame
student_data <- data.frame(
  id = 1:5,
  name = c("Alice", "Bob", "Charlie", "David", "Emma"),
  score = c(85, 92, 78, 90, 88),
  pass = c(TRUE, TRUE, TRUE, TRUE, TRUE)
)

# Examining the data
head(student_data)
```

```
  id    name score pass
1  1   Alice    85 TRUE
2  2     Bob    92 TRUE
3  3 Charlie    78 TRUE
4  4   David    90 TRUE
5  5    Emma    88 TRUE
```

```r
str(student_data)
```

```
'data.frame':	5 obs. of  4 variables:
 $ id   : int  1 2 3 4 5
 $ name : chr  "Alice" "Bob" "Charlie" "David" ...
 $ score: num  85 92 78 90 88
 $ pass : logi  TRUE TRUE TRUE TRUE TRUE
```

```r
summary(student_data)
```

```
       id        name               score         pass        
 Min.   :1   Length:5           Min.   :78.0   Mode:logical  
 1st Qu.:2   Class :character   1st Qu.:85.0   TRUE:5        
 Median :3   Mode  :character   Median :88.0                 
 Mean   :3                      Mean   :86.6                 
 3rd Qu.:4                      3rd Qu.:90.0                 
 Max.   :5                      Max.   :92.0                 
```

Using Factors for Categorical Data

Factors are a special data type in R used to represent categorical variables:

```r
# Create a factor with explicit levels
gender <- factor(c("Male", "Female", "Female", "Male", "Non-binary"),
                 levels = c("Male", "Female", "Non-binary"))

# Summarize the factor
summary(gender)
```

```
      Male     Female Non-binary 
         2          2          1 
```

```r
# Create an ordered factor
education <- factor(
  c("High School", "Bachelor's", "Master's", "Bachelor's", "PhD"),
  levels = c("High School", "Bachelor's", "Master's", "PhD"),
  ordered = TRUE
)

# Compare levels
education[1] < education[5]  # Is High School < PhD?
```

```
[1] TRUE
```

Tidyverse Style Data Manipulation

Following the tidyverse style guide, we can create cleaner, more readable code:

```r
# Creating example data
set.seed(123)
survey_data <- data.frame(
  id = 1:50,
  age = sample(18:70, 50, replace = TRUE),
  income = runif(50, 20000, 100000),
  education = sample(c("High School", "Bachelor's", "Master's", "PhD"),
                     50, replace = TRUE),
  satisfaction = sample(1:10, 50, replace = TRUE)
)

# Using dplyr for data transformation
library(dplyr)

survey_summary <- survey_data %>%
  group_by(education) %>%
  summarize(
    count = n(),
    avg_age = mean(age),
    avg_income = mean(income),
    avg_satisfaction = mean(satisfaction)
  ) %>%
  arrange(desc(avg_income))

kable(survey_summary, digits = 1)
```

education	count	avg_age	avg_income	avg_satisfaction
Master's	15	50.5	64512.4	6.5
High School	6	45.8	62289.5	6.8
PhD	11	38.5	61808.9	7.2
Bachelor's	18	44.1	59753.7	5.1

Statistical Methods

Descriptive Statistics

Descriptive statistics summarize the main features of a dataset. Here's how to calculate key statistics in R:

```r
# Generate example data
set.seed(456)
exam_scores <- rnorm(100, mean = 75, sd = 8)

# Basic summary statistics
mean_score <- mean(exam_scores)
median_score <- median(exam_scores)
sd_score <- sd(exam_scores)
range_score <- range(exam_scores)
quantiles <- quantile(exam_scores, probs = c(0.25, 0.5, 0.75))

# Display results in a table
stats_df <- data.frame(
  Statistic = c("Mean", "Median", "Standard Deviation", "Minimum", "Maximum",
                "25th Percentile", "50th Percentile", "75th Percentile"),
  Value = c(mean_score, median_score, sd_score, range_score[1], range_score[2],
            quantiles[1], quantiles[2], quantiles[3])
)

kable(stats_df, digits = 2)
```

Statistic	Value
Mean	75.96
Median	75.63
Standard Deviation	8.01
Minimum	56.96
Maximum	93.24
25th Percentile	70.74
50th Percentile	75.63
75th Percentile	81.43

Hypothesis Testing

Hypothesis testing is used to determine if there's enough evidence to support a particular claim.

t-tests

```r
# Generate data for two groups
set.seed(789)
group_a <- rnorm(30, mean = 65, sd = 10)
group_b <- rnorm(30, mean = 70, sd = 10)

# Perform a two-sample t-test
t_test_result <- t.test(group_a, group_b)

# Display results
t_test_result
```

```

    Welch Two Sample t-test

data:  group_a and group_b
t = -3.728, df = 50.08, p-value = 0.0004919
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -13.524855  -4.054116
sample estimates:
mean of x mean of y
 62.19052  70.98000
```

ANOVA

```r
# Create data for ANOVA
groups <- rep(c("A", "B", "C"), each = 20)
values <- c(
  rnorm(20, mean = 10, sd = 2),
  rnorm(20, mean = 12, sd = 2),
  rnorm(20, mean = 11, sd = 2)
)
```

```r
anova_data <- data.frame(group = groups, value = values)

# Perform ANOVA
anova_result <- aov(value ~ group, data = anova_data)

# Display results
summary(anova_result)
```

```
            Df Sum Sq Mean Sq F value Pr(>F)
group        2   24.5  12.252   2.765 0.0714 .
Residuals   57  252.6   4.431
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Correlation and Regression

Correlation Analysis

```r
# Generate correlated data
set.seed(101)
x_var <- rnorm(50)
y_var <- 2 * x_var + rnorm(50, sd = 0.5)
z_var <- rnorm(50)  # Uncorrelated with others

# Calculate correlation coefficients
cor_matrix <- cor(cbind(x_var, y_var, z_var))

# Display correlation matrix
round(cor_matrix, 3)
```

```
      x_var y_var z_var
x_var 1.000 0.971 0.062
y_var 0.971 1.000 0.023
z_var 0.062 0.023 1.000
```

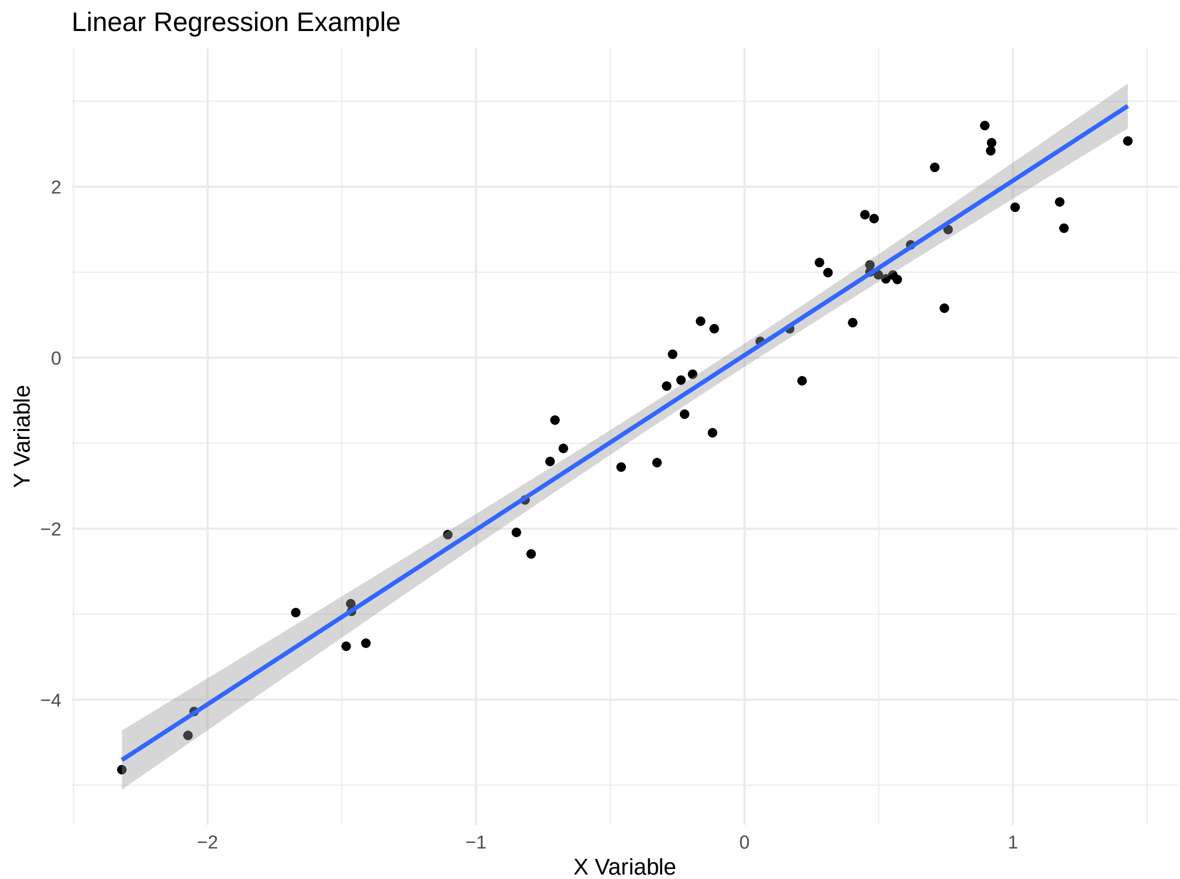
Linear Regression

```r
# Create data frame for regression
reg_data <- data.frame(x = x_var, y = y_var)

# Fit linear model
lm_model <- lm(y ~ x, data = reg_data)

# Summarize model
summary(lm_model)
```

```

Call:
lm(formula = y ~ x, data = reg_data)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.97062 -0.33570  0.00548  0.32046  0.85862 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.02998    0.06733   0.445    0.658    
x            2.04187    0.07232  28.236   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4719 on 48 degrees of freedom
Multiple R-squared:  0.9432,	Adjusted R-squared:  0.942 
F-statistic: 797.3 on 1 and 48 DF,  p-value: < 2.2e-16
```

```r
# Plot regression line
ggplot(reg_data, aes(x = x, y = y)) +
  geom_point() +
  geom_smooth(method = "lm", se = TRUE) +
  labs(title = "Linear Regression Example",
       x = "X Variable",
       y = "Y Variable") +
  theme_minimal()
```

```
`geom_smooth()` using formula = 'y ~ x'
```



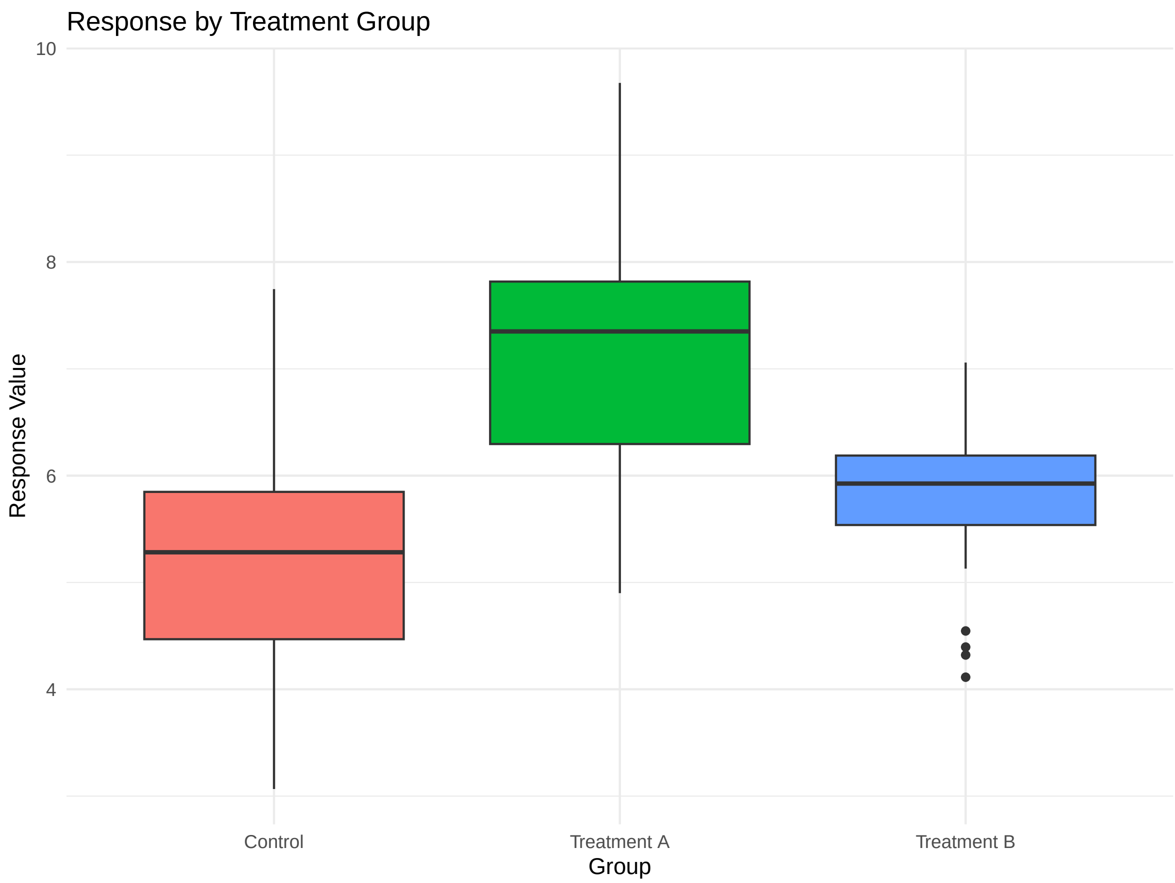
Data Visualization

Basic Graphics with ggplot2

Following the principles of the Grammar of Graphics:

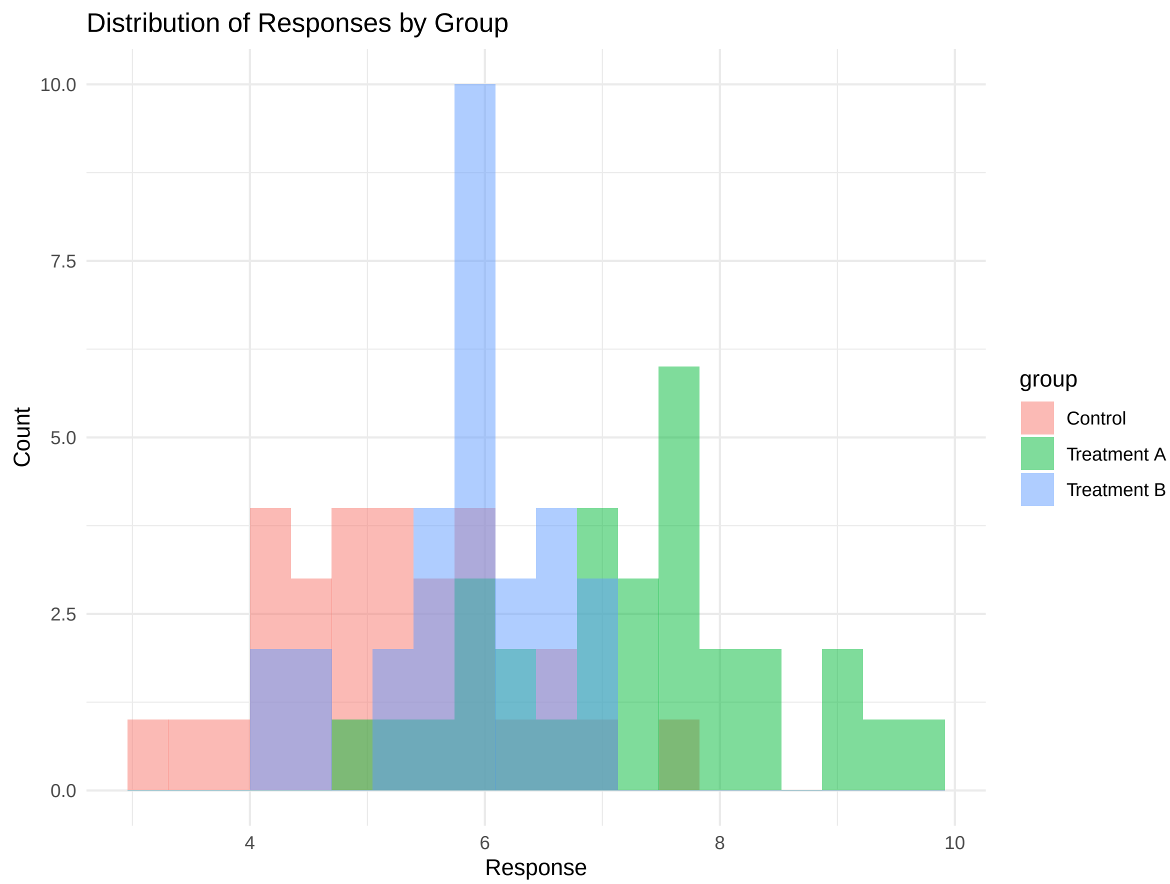
```
# Create example data
set.seed(202)
viz_data <- data.frame(
  group = rep(c("Control", "Treatment A", "Treatment B"), each = 30),
  response = c(
    rnorm(30, mean = 5, sd = 1),
    rnorm(30, mean = 7, sd = 1.2),
    rnorm(30, mean = 6, sd = 0.8)
  )
```

```
)

# Create a boxplot
ggplot(viz_data, aes(x = group, y = response, fill = group)) +
  geom_boxplot() +
  labs(title = "Response by Treatment Group",
       x = "Group",
       y = "Response Value") +
  theme_minimal() +
  theme(legend.position = "none")
```

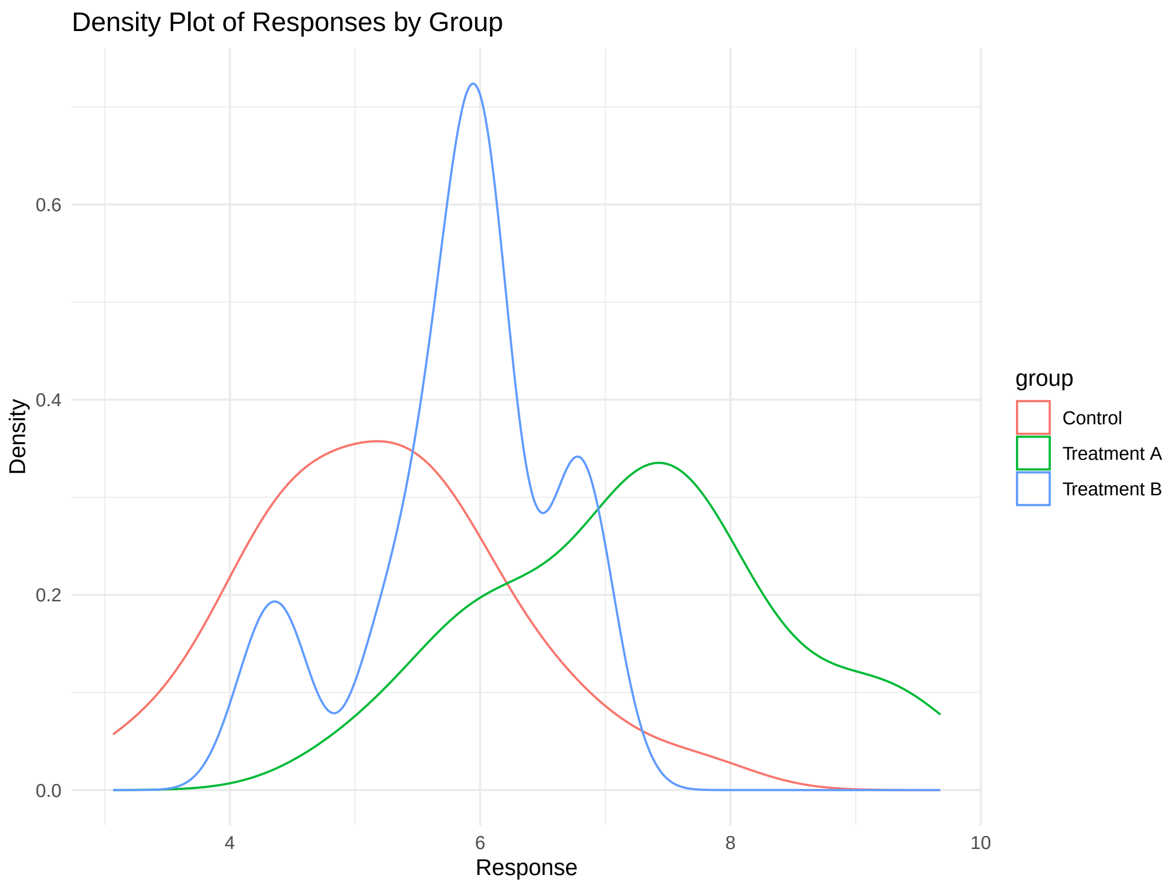


Visualizing Distributions

```r
# Visualize distributions with histograms and density plots
ggplot(viz_data, aes(x = response, fill = group)) +
  geom_histogram(position = "identity", alpha = 0.5, bins = 20) +
  labs(title = "Distribution of Responses by Group",
       x = "Response",
       y = "Count") +
  theme_minimal()
```




```r
ggplot(viz_data, aes(x = response, color = group)) +
  geom_density() +
  labs(title = "Density Plot of Responses by Group",
       x = "Response",
       y = "Density") +
  theme_minimal()
```



Case Studies

Case Study 1: Die Tossing Experiment

This example illustrates the sampling distribution of means and medians, building on the provided example.

```
# Simulate die tossing experiment
set.seed(123)
n_experiments <- 1000
n_tosses <- 3
die_outcomes <- 1:6

# Store results
medians <- numeric(n_experiments)
```

```r
means <- numeric(n_experiments)

# Run simulations
for (i in 1:n_experiments) {
  tosses <- sample(die_outcomes, n_tosses, replace = TRUE)
  medians[i] <- median(tosses)
  means[i] <- mean(tosses)
}

# Create data frame for plotting
results_df <- data.frame(
  Statistic = rep(c("Mean", "Median"), each = n_experiments),
  Value = c(means, medians)
)

# Plot distributions
ggplot(results_df, aes(x = Value, fill = Statistic)) +
  geom_histogram(position = "identity", alpha = 0.5, bins = 30) +
  labs(title = "Sampling Distributions from Die Tossing",
       x = "Value",
       y = "Frequency") +
  theme_minimal() +
  facet_wrap(~ Statistic, ncol = 1)
```


```r
# Calculate summary statistics
means_summary <- c(mean = mean(means), sd = sd(means))
medians_summary <- c(mean = mean(medians), sd = sd(medians))

# Display results
summary_df <- data.frame(
  Statistic = c("Mean", "Median"),
  Expected_Value = c(3.5, 3.5),
  Observed_Mean = c(means_summary["mean"], medians_summary["mean"]),
  Observed_SD = c(means_summary["sd"], medians_summary["sd"])
)

kable(summary_df, digits = 3)
```

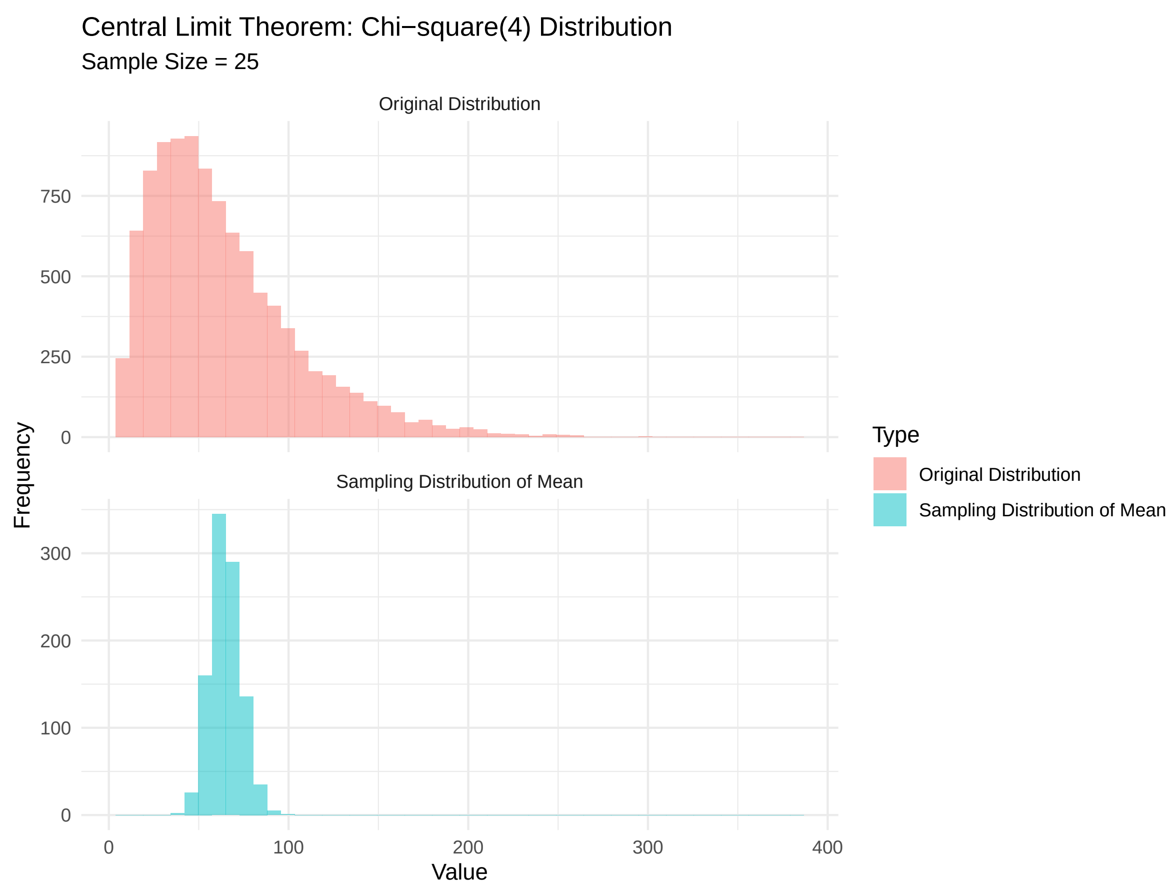
Statistic	Expected_Value	Observed_Mean	Observed_SD
Mean	3.5	3.493	0.993
Median	3.5	3.508	1.364

Case Study 2: The Central Limit Theorem

This case demonstrates the central limit theorem through simulation with different underlying distributions.

```
# Function to generate samples and plot distributions
simulate_clt <- function(dist_func, dist_name, n_samples = 1000, sample_size = 25) {
  # Generate a large number of observations from the distribution
  original_data <- dist_func(10000)

  # Generate samples and calculate means
  sample_means <- numeric(n_samples)
  for (i in 1:n_samples) {
    sample_data <- dist_func(sample_size)
    sample_means[i] <- mean(sample_data)
  }

  # Create a combined data frame for plotting
  plot_data <- data.frame(
    Type = c(rep("Original Distribution", length(original_data)),
             rep("Sampling Distribution of Mean", length(sample_means))),
    Value = c(original_data, sample_means)
  )

  # Create the plot
  p <- ggplot(plot_data, aes(x = Value, fill = Type)) +
    geom_histogram(alpha = 0.5, position = "identity", bins = 50) +
    labs(title = paste("Central Limit Theorem:", dist_name),
         subtitle = paste("Sample Size =", sample_size),
         x = "Value",
         y = "Frequency") +
    theme_minimal() +
    facet_wrap(~ Type, scales = "free_y", ncol = 1)

  # Calculate summary statistics
  stats <- data.frame(
    Distribution = c("Original", "Sample Means"),
```

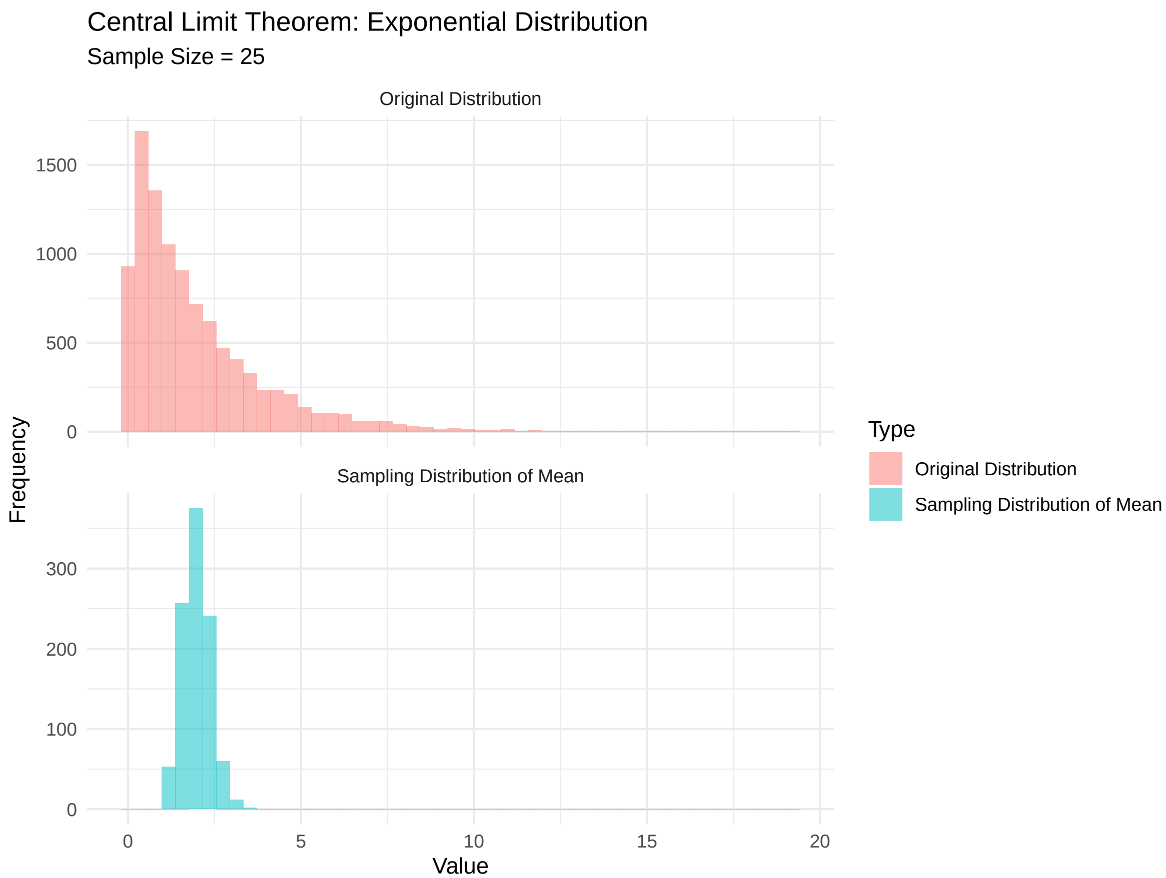
```
    Mean = c(mean(original_data), mean(sample_means)),
    SD = c(sd(original_data), sd(sample_means)),
    Theoretical_SE = c(NA, sd(original_data) / sqrt(sample_size))
  )

  list(plot = p, stats = stats)
}

# Chi-square distribution (4 df)
chi_sq_4 <- function(n) rchisq(n, df = 4) * 15 + 4.5
chi_sq_results <- simulate_clt(chi_sq_4, "Chi-square(4) Distribution")

# Display results
chi_sq_results$plot
```



```r
kable(chi_sq_results$stats, digits = 3)
```

Distribution	Mean	SD	Theoretical_SE
Original	63.867	41.746	NA
Sample Means	64.885	8.464	8.349

```r
# Exponential distribution
exp_dist <- function(n) rexp(n, rate = 0.5)
exp_results <- simulate_clt(exp_dist, "Exponential Distribution")

# Display results
exp_results$plot
```



```r
kable(exp_results$stats, digits = 3)
```

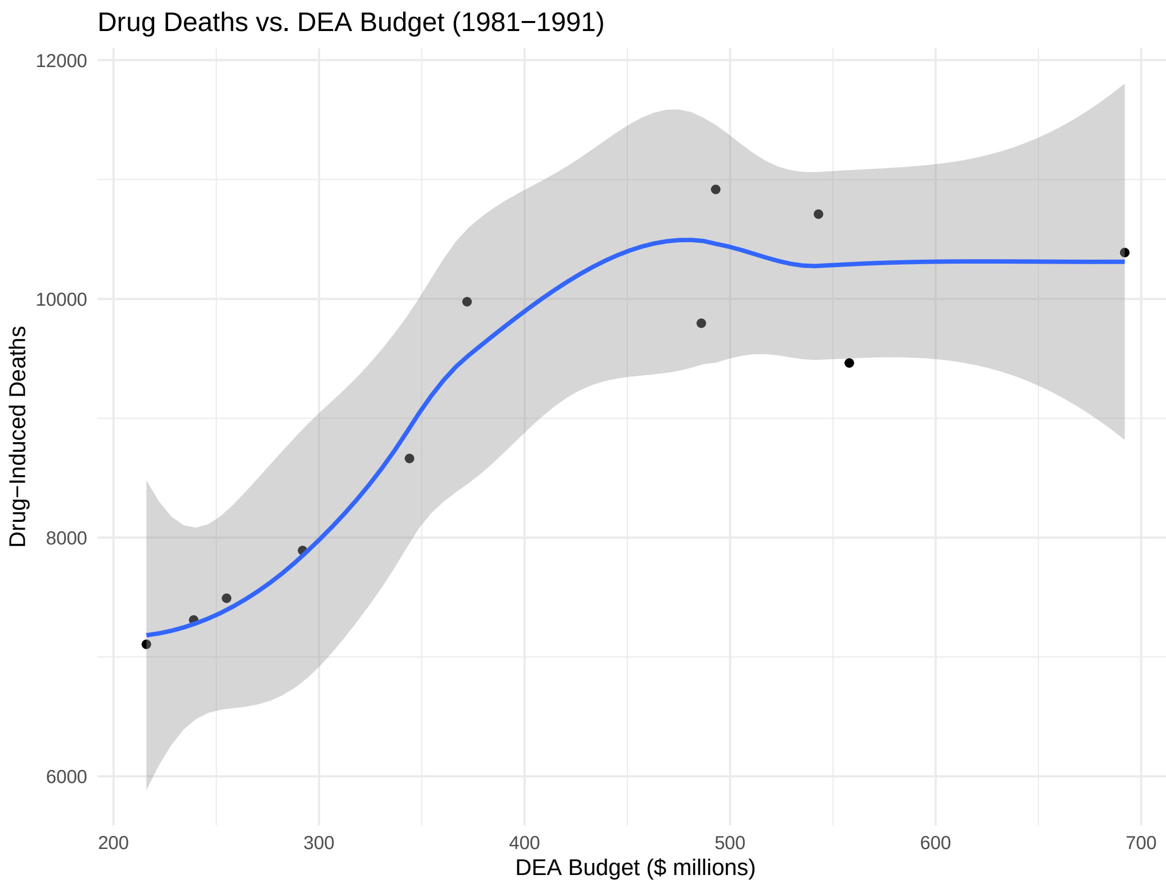
Distribution	Mean	SD	Theoretical_SE
Original	1.988	2.031	NA
Sample Means	1.978	0.392	0.406

Case Study 3: DEA Budget and Drug Deaths Analysis

This example explores relationships between variables over time and examines potential lurking variables.

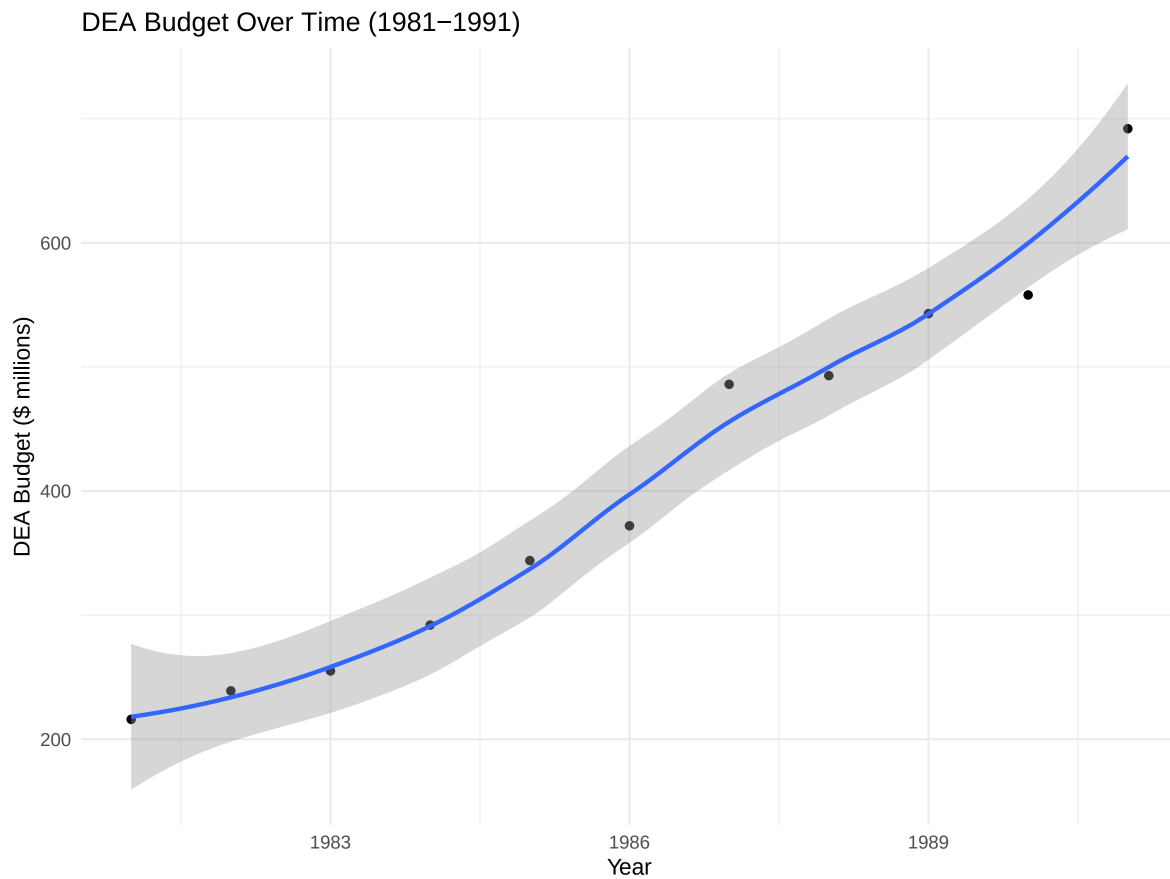
```r
# Create the data
year <- c(1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991)
budget <- c(216, 239, 255, 292, 344, 372, 486, 493, 543, 558, 692)
deaths <- c(7106, 7310, 7492, 7892, 8663, 9976, 9796, 10917, 10710, 9463, 10388)

dea_data <- data.frame(year, budget, deaths)

# Look at relationships
# 1. Deaths vs. Budget
ggplot(dea_data, aes(x = budget, y = deaths)) +
  geom_point() +
  geom_smooth(method = "loess", se = TRUE) +
  labs(title = "Drug Deaths vs. DEA Budget (1981-1991)",
       x = "DEA Budget ($ millions)",
       y = "Drug-Induced Deaths") +
  theme_minimal()
```

```
`geom_smooth()` using formula = 'y ~ x'
```



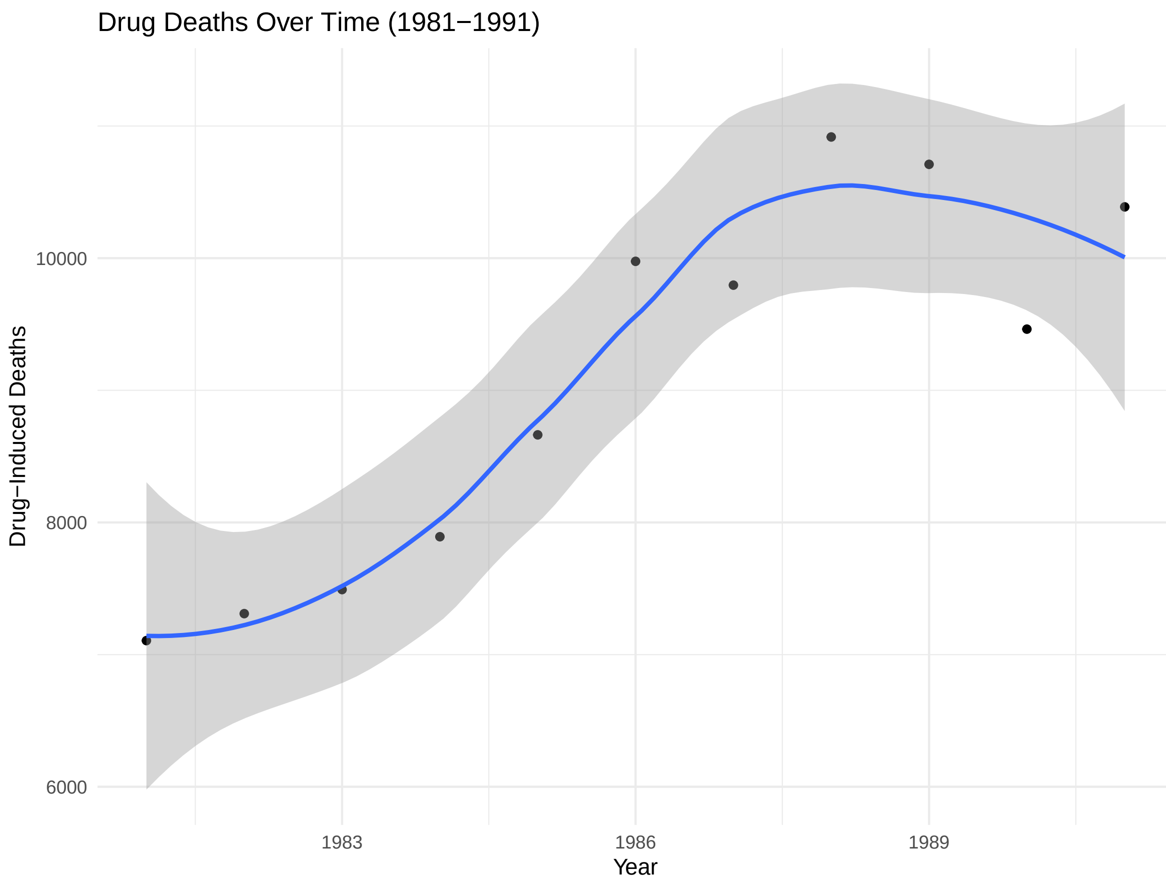

```
# 2. Budget vs. Year
ggplot(dea_data, aes(x = year, y = budget)) +
  geom_point() +
  geom_smooth(method = "loess", se = TRUE) +
  labs(title = "DEA Budget Over Time (1981-1991)",
       x = "Year",
       y = "DEA Budget ($ millions)") +
  theme_minimal()
```

```
`geom_smooth()` using formula = 'y ~ x'
```




```
# 3. Deaths vs. Year
ggplot(dea_data, aes(x = year, y = deaths)) +
  geom_point() +
  geom_smooth(method = "loess", se = TRUE) +
  labs(title = "Drug Deaths Over Time (1981-1991)",
       x = "Year",
       y = "Drug-Induced Deaths") +
  theme_minimal()
```

```
`geom_smooth()` using formula = 'y ~ x'
```




```r
# Correlation matrix
cor_matrix <- cor(dea_data)
round(cor_matrix, 3)
```

```
        year budget deaths
year   1.000  0.981  0.885
budget 0.981  1.000  0.862
deaths 0.885  0.862  1.000
```

```r
# Test different models for Budget vs Year
linear_model <- lm(budget ~ year, data = dea_data)
quadratic_model <- lm(budget ~ year + I(year^2), data = dea_data)
exp_model <- lm(log(budget) ~ year, data = dea_data)

# Compare models with AIC
model_comparison <- data.frame(
```

```r
  Model = c("Linear", "Quadratic", "Exponential"),
  AIC = c(AIC(linear_model), AIC(quadratic_model), AIC(exp_model))
)
kable(model_comparison, digits = 2)
```

Model	AIC
Linear	111.29
Quadratic	108.27
Exponential	-28.17

Best Practices

Code Style and Organization

Following the tidyverse style guide, code should be organized with:

1. Clear, descriptive variable names
2. Consistent spacing and indentation
3. Pipes (`%>%`) for readable data operations
4. Comments to explain complex operations
5. Breaking complex operations into steps

Reproducible Research

For reproducible research:

1. Set a seed for random operations
2. Document all data transformations
3. Use relative file paths
4. Include session information

```r
# Display session information
sessionInfo()
```

```
R version 4.1.2 (2021-11-01)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 22.04.3 LTS

Matrix products: default
```

```
BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.10.0
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] ggplot2_3.5.1 dplyr_1.1.4   knitr_1.50

loaded via a namespace (and not attached):
 [1] rstudioapi_0.17.1 magrittr_2.0.3    splines_4.1.2     tidyselect_1.2.1
 [5] munsell_0.5.1     lattice_0.20-45   colorspace_2.1-1  R6_2.6.1
 [9] rlang_1.1.5       fastmap_1.2.0     tools_4.1.2       grid_4.1.2
[13] nlme_3.1-155      gtable_0.3.6      mgcv_1.8-39       xfun_0.52
[17] cli_3.6.4         withr_3.0.2       htmltools_0.5.8.1 yaml_2.3.10
[21] digest_0.6.37     tibble_3.2.1      lifecycle_1.0.4   Matrix_1.4-0
[25] farver_2.1.2      vctrs_0.6.5       glue_1.8.0        evaluate_1.0.3
[29] rmarkdown_2.29    labeling_0.4.3    compiler_4.1.2    pillar_1.10.1
[33] generics_0.1.3    scales_1.3.0      jsonlite_2.0.0    pkgconfig_2.0.3
```

Data Visualization Principles

1. Choose appropriate chart types for your data
2. Use meaningful labels and titles
3. Consider accessibility (colorblind-friendly palettes)
4. Keep visualizations clean and focused
5. Use facets to compare across categories

Conclusion

R provides a powerful environment for statistical analysis with a rich ecosystem of packages. By following best practices for coding style, data visualization, and statistical methodology, we can produce analyses that are robust, reproducible, and easy to understand.

This document has demonstrated various techniques for data manipulation, statistical analysis, and visualization using R. The case studies show how these techniques can be applied to real-world problems, from understanding sampling distributions to analyzing trends over time.