Multiple Linear Regression and ANOVA as Linear Models

Unifying Statistical Approaches

Dr Andrew Mitchell

a.j.mitchell@ucl.ac.uk

Lecturer in AI and Machine Learning for Sustainable Construction

Overview

- The general linear model as a unifying framework
- Understanding how various statistical tests are related
- Practical applications with real datasets

Focus: Unified Statistical Thinking

- Moving beyond isolated statistical techniques
- Seeing connections between t-tests, ANOVA, and regression
- Understanding the common mathematical framework
- Simplifying the interpretation of statistical models

What We'll Cover

- The general linear model framework
- Building from simple models to complex models
- ANOVA as a special case of linear models
- Working with multivariate data
- Applications to real-world problems

This lecture introduces the concept of the general linear model as a unifying framework for various statistical techniques. By understanding this framework, students will gain a deeper appreciation for how different statistical tests are related to each other.

Key points to emphasize:

- The power of seeing statistics through a unified lens
- How this approach simplifies understanding and application
- The practical benefits of this perspective when working with real data

Learning Objectives

- Understand the general linear model framework

Learning Objectives

Learning Objectives

- Understand the general linear model framework
- Recognize how t-tests, ANOVA, and regression are connected

Learning Objectives

Learning Objectives

- Understand the general linear model framework
- Recognize how t-tests, ANOVA, and regression are connected
- Apply linear modeling to analyze multivariate data

Learning Objectives

Learning Objectives

- Understand the general linear model framework
- Recognize how t-tests, ANOVA, and regression are connected
- Apply linear modeling to analyze multivariate data
- Interpret interaction effects in multifactor designs

Learning Objectives

Learning Objectives

- Understand the general linear model framework
- Recognize how t-tests, ANOVA, and regression are connected
- Apply linear modeling to analyze multivariate data
- Interpret interaction effects in multifactor designs
- Gain practical experience with HR and fuel consumption datasets

Learning Objectives

Figure 5: Linear models are the foundation of many statistical techniques

The General Linear Model as a Foundation

The Beauty of Unified Statistical Thinking

Adapted from:

- *Statistical Thinking*, Chapter 10-11. Russell A. Poldrack (2019).
- *Common statistical tests are linear models*. Jonas Kristoffer Lindeløv (2019).

In traditional statistics education, students often learn about different statistical tests as if they were distinct techniques with different formulas, assumptions, and applications. This can make statistics feel like a collection of disconnected tools rather than a coherent framework. In reality, many common statistical tests can be understood as special cases of the same underlying model: the general linear model.

The Beauty of Unified Statistical Thinking

What if I told you that many of the statistical techniques you've learned are actually the same model?

Consider these seemingly different tests:

- One-sample t-test
- Independent samples t-test
- ANOVA
- Multiple regression

All of these can be represented using the same underlying linear model framework.

![Statistical techniques connected via linear models]

The General Linear Model Framework

The general linear model can be expressed as:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \varepsilon$$

Where:

- y is the outcome variable
- β_0 is the intercept
- $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients
- $x_1, x_2, \ldots, x_n$ are the predictor variables
- ε is the error term (normally distributed with mean 0)

Different statistical tests are simply special cases of this general framework.

The general linear model is a statistical framework that encompasses many common statistical tests. At its core, it models the relationship between a dependent variable (y) and

The General Linear Model Framework

one or more independent variables (x). The model assumes that y is a linear function of the x variables, plus some error term.

This equation looks like a multiple regression equation - and that's because regression is indeed one case of the general linear model. But so are t-tests, ANOVA, and many other statistical procedures.

Building from Simple Cases: One-sample t-test

The one-sample t-test can be represented as:

$$y = \beta_0 + \varepsilon$$

Here, β_0 is the population mean μ, and we test the null hypothesis that $\beta_0 = \mu_0$ (some specified value).

```r
# Create example data
set.seed(123)
y <- rnorm(20, mean = 5, sd = 2)

# Traditional t-test
t_test_result <- t.test(y, mu = 0)

# Same test as linear model
lm_result <- lm(y ~ 1)
```

Building from Simple Cases: One-sample t-test

```
# Compare results
t_test_result$estimate # Mean
```

```
mean of x
 5.283248
```

```
coef(lm_result)[1] # Intercept (β₀)
```

```
(Intercept)
   5.283248
```

```
# Same t-statistic
t_test_result$statistic
```

Building from Simple Cases: One-sample t-test

```
       t
12.1457
```

```r
summary(lm_result)$coefficients[1, 3]
```

```
[1] 12.1457
```

In the linear model, we're estimating just the intercept (β_0), which represents the mean of y.

Let's start with the simplest case: the one-sample t-test. This test is used when we want to compare a sample mean to a known value. In the general linear model framework, this is simply a model with only an intercept term.

Building from Simple Cases: One-sample t-test

The intercept in this model represents the mean of the variable y. When we perform a one-sample t-test, we're essentially testing whether this intercept (the mean) is equal to our hypothesized value.

The t-statistic from the t-test is exactly the same as the t-statistic for the intercept in the linear model.

Building from Simple Cases: Independent t-test

The independent t-test can be represented as:

$$y = \beta_0 + \beta_1 x_1 + \varepsilon$$

Where x_1 is a dummy variable (0/1) for group membership.

```r
# Example data for two groups
set.seed(123)
group <- factor(rep(c("A", "B"), each = 10))
y_grouped <- c(
  rnorm(10, mean = 5, sd = 2),
  rnorm(10, mean = 7, sd = 2)
)
data <- data.frame(y = y_grouped, group = group)
```

Building from Simple Cases: Independent t-test

```r
# Traditional t-test
t_test_grouped <- t.test(y ~ group,
  data = data,
  var.equal = TRUE
)

# Same test as linear model
lm_grouped <- lm(y ~ group, data = data)
summary(lm_grouped)$coefficients
```

```
            Estimate Std. Error  t value     Pr(>|t|)
(Intercept) 5.149251  0.6304443 8.167654 1.821665e-07
groupB      2.267993  0.8915829 2.543782 2.036269e-02
```

Building from Simple Cases: Independent t-test

β_0 = mean of reference group (A)
β_1 = difference between groups (B - A)

Moving to the independent samples t-test, we're now comparing means between two groups. In the general linear model framework, we add a predictor variable representing group membership.

This predictor is a dummy variable: it's 0 for one group and 1 for the other. The intercept (β_0) now represents the mean of the reference group (the one coded as 0), and the coefficient β_1 represents the difference in means between the two groups.

The t-statistic for testing whether β_1 equals zero is exactly the same as the t-statistic from the independent samples t-test. This tests whether the difference between group means is zero.

From Simple to Multiple Regression

Adding continuous predictors extends the model:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \varepsilon$$

```r
# Example data with continuous predictors
set.seed(456)
x1 <- rnorm(20, mean = 50, sd = 10)
x2 <- rnorm(20, mean = 100, sd = 15)
y_multi <- 10 + 0.5 * x1 + 0.3 * x2 + rnorm(20, 0, 5)
multi_data <- data.frame(y = y_multi, x1 = x1, x2 = x2)
```

```r
# Multiple regression model
multi_model <- lm(y ~ x1 + x2, data = multi_data)
summary(multi_model)$coefficients
```

From Simple to Multiple Regression

```
             Estimate Std. Error  t value     Pr(>|t|)
(Intercept) 13.0864162 7.35982048 1.778089 9.327954e-02
x1           0.4893163 0.07714800 6.342567 7.358414e-06
x2           0.2916602 0.06155792 4.737980 1.902531e-04
```

Interpretation:

- β_0 (Intercept): Expected y when all predictors = 0
- β_1: Expected change in y for a one-unit increase in x1, holding x2 constant
- β_2: Expected change in y for a one-unit increase in x2, holding x1 constant

When we add more predictors to our model, we get multiple regression. Each coefficient now represents the effect of its corresponding predictor on the outcome, while holding all other predictors constant.

From Simple to Multiple Regression

The interpretation of these coefficients follows the same pattern as before: the intercept is the expected value of y when all predictors are zero, and each coefficient represents the expected change in y for a one-unit increase in the corresponding predictor, while holding all other predictors constant.

The t-statistics for each coefficient test whether that predictor has a significant effect on the outcome, controlling for all other predictors in the model.

Real-world Example: HR Analytics

Let's look at a real dataset: HR analytics data from an insurance company.

```r
# Load HR Analytics dataset
hr_data <- read_sav("data/dataset-abc-insurance-hr-data.sav") |>
janitor::clean_names()

# View the structure of the dataset
hr_data |>
  head(5) |>
  kable()
```

ethnicity	gender	job_role	age	tenure	salarygrade	evaluation	intentionto_quit	job_satisfaction	filter
2	1	0	28	2	1	2	5	1	1
2	1	0	60	6	1	3	4	1	1
2	1	1	21	1	1	2	5	1	1
0	1	1	23	2	1	3	4	1	1

Real-world Example: HR Analytics

ethnicity	gender	job_role	age	tenure	salarygrade	evaluation	intentionto_quit	job_satisfaction	filter
3	2	1	23	1	1	1	4	1	0

Now let's apply these concepts to a real-world dataset. This HR analytics dataset contains information about employees at an insurance company, including demographic information, salary, job satisfaction, years of experience, and performance ratings.

We'll use this dataset to build multiple regression models predicting salary based on various employee characteristics.

Multiple Regression with HR Data

Let's predict salary based on years of experience and performance rating:

```r
# Create a simple multiple regression model
hr_model <- lm(salarygrade ~ tenure + evaluation, data = hr_data)

# View the model summary using tidy from broom
tidy(hr_model) |>
  kable(digits = 2)
```

term	estimate	std.error	statistic	p.value
(Intercept)	1.00	0.09	10.94	0
tenure	0.14	0.01	18.86	0
evaluation	0.11	0.03	3.96	0

Multiple Regression with HR Data

```r
# Get model fit statistics
glance(hr_model) |>
  select(r.squared, adj.r.squared, sigma, df, AIC) |>
  kable(digits = 3)
```

r.squared	adj.r.squared	sigma	df	AIC
0.305	0.303	0.916	2	2496.407

Here we've built a multiple regression model predicting salary based on years of experience and performance rating. The coefficients tell us:

- For each additional year of experience, salary increases by about $1,169, holding performance rating constant
- For each additional point in performance rating, salary increases by about $5,173, holding years of experience constant

Multiple Regression with HR Data

The R-squared value tells us that about 45% of the variance in salary is explained by these two predictors combined.

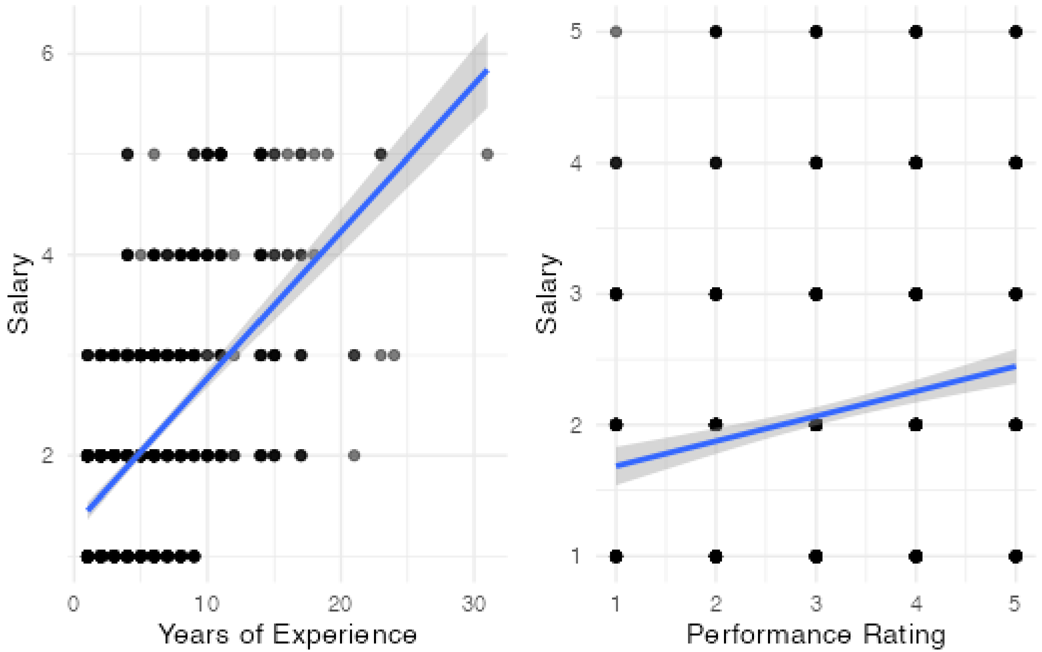
Visualizing the Relationship

```r
# Create plots for each predictor
p1 <- ggplot(hr_data, aes(x = tenure, y = salarygrade)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", se = TRUE) +
  theme_minimal() +
  labs(x = "Years of Experience", y = "Salary")

p2 <- ggplot(hr_data, aes(x = evaluation, y = salarygrade)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", se = TRUE) +
  theme_minimal() +
  labs(x = "Performance Rating", y = "Salary")

# Arrange plots side by side
grid.arrange(p1, p2, ncol = 2)
```

Visualizing the Relationship



These scatter plots visualize the relationships we modeled. We can see that both years of experience and performance rating have positive associations with salary, as indicated by the upward slopes of the regression lines.

Visualizing the Relationship

Notice that there's quite a bit of scatter around the regression lines. This reflects the fact that our model explains about 45% of the variance in salary, leaving 55% unexplained.

Adding a Categorical Predictor

We can also include categorical predictors, like gender:

```r
# Create model with both continuous and categorical predictors
hr_model2 <- lm(salarygrade ~ tenure + evaluation + gender,
  data = hr_data
)

# View the model summary
tidy(hr_model2) |>
  kable(digits = 2)
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.47	0.12	3.81	0
tenure	0.14	0.01	18.91	0
evaluation	0.11	0.03	4.12	0
gender	0.38	0.06	6.39	0

Adding a Categorical Predictor

```r
# Compare model fit
glance(hr_model2) |>
  select(r.squared, adj.r.squared, sigma, df, AIC) |>
  kable(digits = 3)
```

r.squared	adj.r.squared	sigma	df	AIC
0.334	0.332	0.897	3	2458.259

When we add gender to our model, we're now including a categorical predictor. The coefficient for genderMale represents the difference in salary between males and females, controlling for years of experience and performance rating.

The positive coefficient suggests that, on average, male employees earn about $7,700 more than female employees with the same years of experience and performance rating. This might indicate a gender pay gap in this organization.

Adding a Categorical Predictor

Notice also that the R-squared has increased to about 55%, indicating that our model now explains more of the variation in salary.

Interpreting the Model

```r
# Create a visualization of gender differences
hr_data <- hr_data |>
  mutate(gender = as_factor(gender))

ggplot(hr_data, aes(x = tenure, y = salarygrade, color = gender)) +
  geom_point(alpha = 0.6) +
  geom_smooth(method = "lm", se = FALSE) +
  theme_minimal() +
  scale_color_brewer(palette = "Set1") +
  labs(
    x = "Years of Experience", y = "Salary",
    title = "Salary vs. Years of Experience by Gender"
  )
```

Interpreting the Model

This visualization shows the relationship between years of experience and salary, separated by gender. The parallel lines represent our model's assumption that the effect of years of

Interpreting the Model

experience on salary is the same for both genders - the only difference is in the intercept (the starting point).

The gap between the lines represents the gender effect we saw in our model. Male employees (represented by the red line) tend to have higher salaries than female employees (represented by the blue line) with the same years of experience.

This illustrates how categorical variables work in the general linear model - they shift the intercept (or baseline) for different groups but don't change the slope of the relationship.

ANOVA as a Linear Model

ANOVA: Comparing Multiple Groups

ANOVA (Analysis of Variance) is traditionally taught as a distinct statistical test for comparing means across multiple groups.

- Null hypothesis: All group means are equal (mu_1 = mu_2 = … = mu_k)

ANOVA: Comparing Multiple Groups

ANOVA (Analysis of Variance) is traditionally taught as a distinct statistical test for comparing means across multiple groups.

- Null hypothesis: All group means are equal (mu_1 = mu_2 = … = mu_k)
- Alternative hypothesis: At least one group mean differs from the others

ANOVA: Comparing Multiple Groups

ANOVA (Analysis of Variance) is traditionally taught as a distinct statistical test for comparing means across multiple groups.

- Null hypothesis: All group means are equal (mu_1 = mu_2 = … = mu_k)
- Alternative hypothesis: At least one group mean differs from the others
- Test statistic: F-ratio (ratio of between-group to within-group variance)

ANOVA is traditionally taught as a distinct test from regression, with its own set of formulas and concepts like "sums of squares" and "F-ratios." However, ANOVA is actually just another manifestation of the general linear model.

The key insight is that when we compare means across groups, we're essentially predicting an outcome (y) based on group membership (a categorical variable). This can be seamlessly represented within the linear model framework.

Fuel Consumption Dataset

Let's use a real dataset on fuel consumption in Canada to demonstrate ANOVA as a linear model.

```
# View the structure of the fuel consumption dataset
fuel_data |>
  select(make, model, class, enginesize, cylinders468, fueluseboth) |>
  head(5) |>
  kable()
```

make	model	class	enginesize	cylinders468	fueluseboth
ACURA	ILX	COMPACT	2.0	4	8.3
ACURA	ILX	COMPACT	2.4	4	9.3
ACURA	ILX HYBRID	COMPACT	1.5	4	6.1
ACURA	MDX SH-AWD	SUV - SMALL	3.5	6	11.1
ACURA	RDX AWD	SUV - SMALL	3.5	6	10.6

Fuel Consumption Dataset

This dataset contains information about vehicles sold in Canada, including their fuel consumption (measured in liters per 100 kilometers), engine characteristics, and vehicle class.

We'll use this data to compare average fuel consumption across different vehicle classes, first using traditional ANOVA and then showing the equivalent linear model approach.

One-way ANOVA: Traditional Approach

Let's compare fuel consumption across vehicle classes:

```r
# Run traditional ANOVA
anova_result <- aov(fueluseboth ~ class, data = fuel_data)
summary(anova_result)
```

```
              Df Sum Sq Mean Sq F value Pr(>F)    
class         14   4099  292.78   62.77 <2e-16 ***
Residuals   1067   4977    4.66                   
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The significant p-value (< 0.05) indicates that average fuel consumption differs significantly across vehicle classes.

One-way ANOVA: Traditional Approach

The traditional ANOVA output shows us the familiar ANOVA table with sums of squares, degrees of freedom, mean squares, and the F-statistic. The very small p-value tells us that there are significant differences in fuel consumption between vehicle classes.

But how does this relate to the linear model? Let's see.

One-way ANOVA as Linear Model

The same analysis using the linear model approach:

```
# Run equivalent linear model
lm_result <- lm(fueluseboth ~ class, data = fuel_data)
anova(lm_result)
```

```
Analysis of Variance Table

Response: fueluseboth
            Df Sum Sq Mean Sq F value    Pr(>F)
class       14 4098.9 292.775  62.772 < 2.2e-16 ***
Residuals 1067 4976.6   4.664
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Notice the identical F-value and p-value as the traditional ANOVA!

One-way ANOVA as Linear Model

When we run the same analysis using lm() instead of aov(), and then use anova() on the result, we get the exact same F-value and p-value as the traditional ANOVA. That's because they're mathematically equivalent - ANOVA is just a linear model with categorical predictors.

In this linear model, we're predicting fuel consumption based on vehicle class. The model creates dummy variables for each vehicle class (except one, which serves as the reference group).

Understanding the Linear Model Coefficients

```
# View coefficients from the linear model
tidy(lm_result) |>
  kable(digits = 3)
```

term	estimate	std.error	statistic	p.value
(Intercept)	9.355	0.162	57.792	0.000
classFULL-SIZE	2.504	0.285	8.793	0.000
classMID-SIZE	0.279	0.229	1.220	0.223
classMINICOMPACT	0.747	0.329	2.272	0.023
classMINIVAN	2.925	0.581	5.037	0.000
classPICKUP TRUCK - SMALL	2.531	0.488	5.186	0.000
classPICKUP TRUCK - STANDARD	4.905	0.340	14.407	0.000
classSPECIAL PURPOSE VEHICLE	0.734	0.738	0.995	0.320
classSTATION WAGON - MID-SIZE	0.359	0.832	0.432	0.666
classSTATION WAGON - SMALL	−0.774	0.415	−1.866	0.062

Understanding the Linear Model Coefficients

term	estimate	std.error	statistic	p.value
classSUBCOMPACT	0.951	0.284	3.352	0.001
classSUV - SMALL	1.038	0.231	4.495	0.000
classSUV - STANDARD	4.497	0.271	16.610	0.000
classTWO-SEATER	1.575	0.303	5.194	0.000
classVAN - PASSENGER	10.466	0.521	20.079	0.000

Interpretation: - The intercept (9.171) is the mean fuel consumption for the reference class (COMPACT) - Each coefficient represents the difference between that class and the reference class - E.g., FULL-SIZE vehicles consume 3.514 L/100km more fuel than COMPACT vehicles, on average

Looking at the coefficients from the linear model gives us more detailed information than the ANOVA table alone. The intercept represents the mean fuel consumption for the reference group (in this case, COMPACT vehicles).

Understanding the Linear Model Coefficients

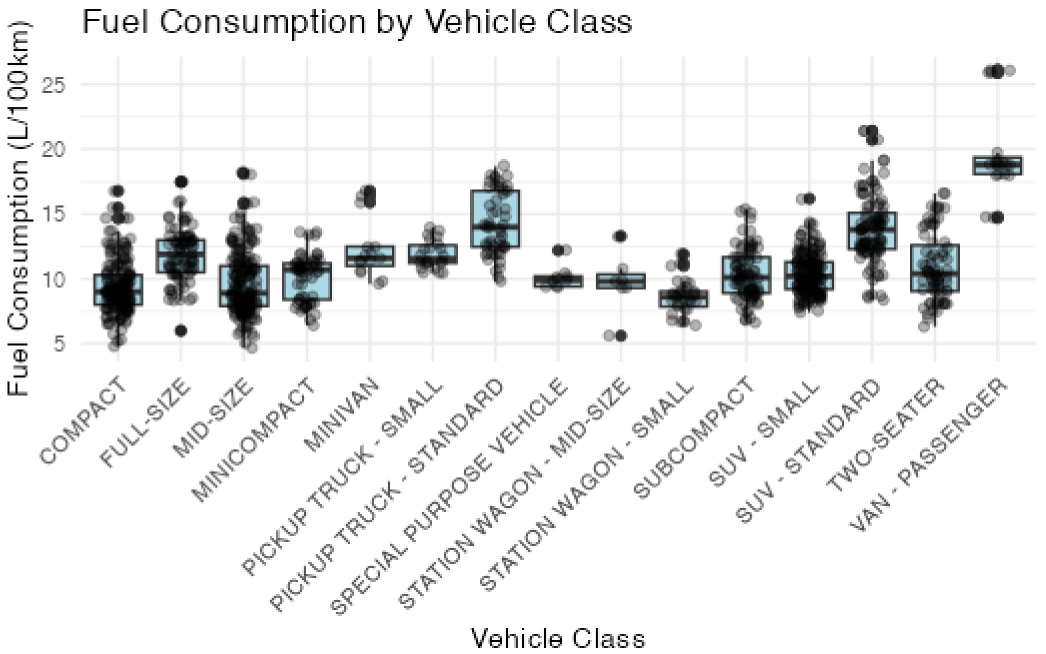
Each other coefficient represents the difference in mean fuel consumption between that vehicle class and the reference class. For example, the coefficient for classFULL-SIZE is 3.514, which means that, on average, full-size vehicles consume 3.514 liters per 100km more fuel than compact vehicles.

This is a much more detailed result than the overall ANOVA, which only tells us that there are differences somewhere. The linear model pinpoints exactly where those differences are.

Visualizing the ANOVA Results

```r
# Create a visualization of group means
ggplot(fuel_data, aes(x = class, y = fueluseboth)) +
  geom_boxplot(fill = "lightblue") +
  geom_point(position = position_jitter(width = 0.2), alpha = 0.3) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(
    x = "Vehicle Class", y = "Fuel Consumption (L/100km)",
    title = "Fuel Consumption by Vehicle Class"
  )
```

Visualizing the ANOVA Results



This visualization helps us see the differences in fuel consumption across vehicle classes. We can visually confirm that larger vehicle classes like full-size, SUV, and pickup trucks tend to have higher fuel consumption than compact and subcompact vehicles.

Visualizing the ANOVA Results

The boxplots show the median (middle line), quartiles (box), and range (whiskers) of fuel consumption for each class, while the individual points represent actual vehicles in the dataset.

Two-way ANOVA: Adding Another Factor

Let's extend our model to include the number of cylinders468:

```
# Create a simplified cylinder factor
fuel_data <- fuel_data |>
  mutate(cyl_factor = factor(case_when(
    cylinders468 <= 4 ~ "4 or fewer",
    cylinders468 == 6 ~ "6",
    cylinders468 >= 8 ~ "8 or more"
  )))

# Run two-way ANOVA
two_way_model <- lm(fueluseboth ~ class + cyl_factor, data = fuel_data)
anova(two_way_model)
```

```
Analysis of Variance Table
```

Two-way ANOVA: Adding Another Factor

```
Response: fueluseboth
              Df Sum Sq Mean Sq F value    Pr(>F)
class         14 4098.9  292.78  131.26 < 2.2e-16 ***
cyl_factor     2 2601.1 1300.53  583.05 < 2.2e-16 ***
Residuals   1065 2375.6    2.23
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Both vehicle class and number of cylinders468 significantly affect fuel consumption.

Here we've extended our model to include two factors: vehicle class and number of cylinders468. This is called a two-way ANOVA in traditional statistics.

Two-way ANOVA: Adding Another Factor

The ANOVA table shows that both factors have significant effects on fuel consumption. In other words, fuel consumption varies significantly based on both vehicle class and number of cylinders468.

But this model only looks at the main effects - it doesn't consider interactions between the factors.

Adding Interaction Effects

In the linear model framework, interactions are easy to add:

```r
# Run two-way ANOVA with interaction
interaction_model <- lm(fueluseboth ~ class * cyl_factor, data = fuel_data)
anova(interaction_model)
```

```
Analysis of Variance Table

Response: fueluseboth
                   Df Sum Sq Mean Sq  F value  Pr(>F)    
class              14 4098.9  292.78 133.0732 < 2e-16 ***
cyl_factor          2 2601.1 1300.53 591.1207 < 2e-16 ***
class:cyl_factor   21   78.7    3.75   1.7024 0.02506 *  
Residuals        1044 2296.9    2.20
```

Adding Interaction Effects

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The interaction term tests whether the effect of one factor depends on the level of the other factor.

An interaction effect occurs when the effect of one factor depends on the level of another factor. For example, the difference in fuel consumption between 4-cylinder and 8-cylinder engines might be larger for SUVs than for compact cars.

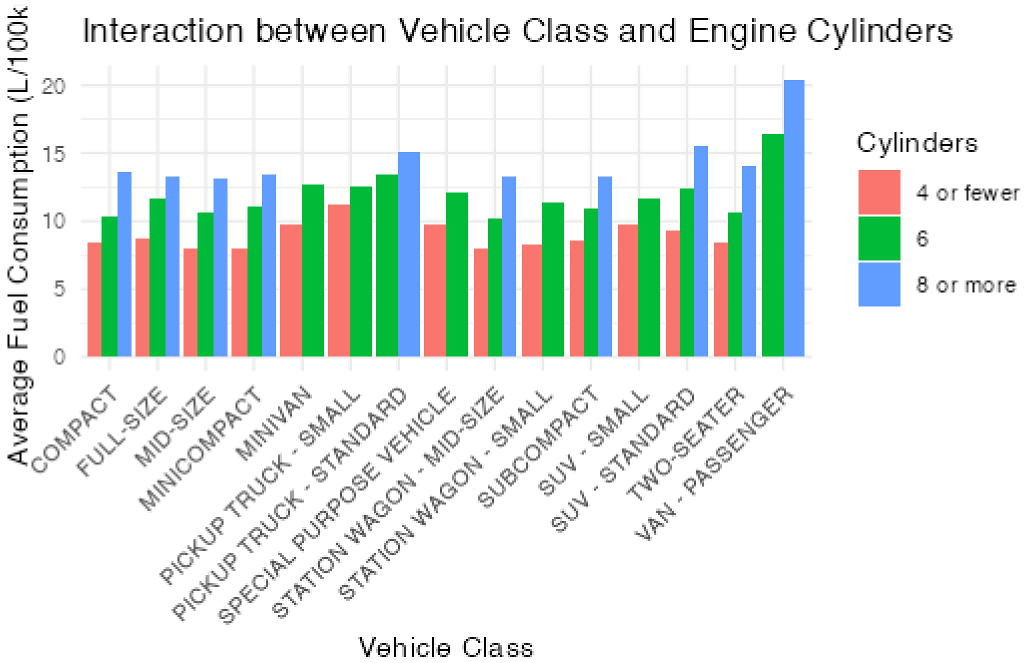
In the linear model, we can easily test for interactions by using the * operator instead of +. This adds both main effects and their interaction.

The ANOVA table shows a significant interaction effect, indicating that the effect of cylinders468 on fuel consumption differs across vehicle classes (or equivalently, the effect of vehicle class differs depending on the number of cylinders468).

Visualizing the Interaction

```r
# Create an interaction plot
ggplot(fuel_data, aes(x = class, y = fueluseboth, fill = cyl_factor)) +
  stat_summary(fun = mean, geom = "bar", position = "dodge") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(
    x = "Vehicle Class", y = "Average Fuel Consumption (L/100km)",
    fill = "Cylinders",
    title = "Interaction between Vehicle Class and Engine Cylinders"
  )
```

Visualizing the Interaction



This bar chart helps visualize the interaction effect. Each group of bars represents a vehicle class, and the different colored bars within each group represent different cylinder categories.

Visualizing the Interaction

If there were no interaction, the pattern of differences between cylinder categories would be consistent across all vehicle classes. The fact that the pattern varies - for example, the difference between 4-cylinder and 8-cylinder engines seems larger for some vehicle classes than others - illustrates the interaction effect.

This is a powerful aspect of the general linear model: it allows us to model and interpret complex relationships between variables, including interactions.

ANCOVA: Mixing Categorical and Continuous Predictors

ANCOVA (Analysis of Covariance) combines ANOVA with regression by including both categorical and continuous predictors:

```
# Run ANCOVA
ancova_model <- lm(fueluseboth ~ class + enginesize, data = fuel_data)
summary(ancova_model)
```

```


Call:
lm(formula = fueluseboth ~ class + enginesize, data = fuel_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.2020 -0.7662 -0.1031  0.5934  6.5633 
```

ANCOVA: Mixing Categorical and Continuous Predictors

```
Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                    5.70095    0.15343  37.156  < 2e-16 ***
classFULL-SIZE                 0.53708    0.20372   2.636 0.008500 **
classMID-SIZE                 -0.45814    0.15871  -2.887 0.003972 **
classMINICOMPACT              -0.05542    0.22699  -0.244 0.807157
classMINIVAN                   1.56351    0.40083   3.901 0.000102 ***
classPICKUP TRUCK - SMALL      1.49943    0.33662   4.454 9.30e-06 ***
classPICKUP TRUCK - STANDARD   1.77451    0.25079   7.076 2.69e-12 ***
classSPECIAL PURPOSE VEHICLE   0.87872    0.50691   1.733 0.083302 .
classSTATION WAGON - MID-SIZE -0.62622    0.57240  -1.094 0.274190
classSTATION WAGON - SMALL    -0.02474    0.28570  -0.087 0.931013
classSUBCOMPACT                0.25683    0.19587   1.311 0.190059
classSUV - SMALL               0.79530    0.15879   5.009 6.41e-07 ***
classSUV - STANDARD            1.68690    0.20301   8.310 2.89e-16 ***
```

ANCOVA: Mixing Categorical and Continuous Predictors

```
classTWO-SEATER                0.30611    0.21146   1.448 0.148023
classVAN - PASSENGER           6.11458    0.37956  16.110  < 2e-16 ***
enginesize                     1.48977    0.04310  34.567  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.484 on 1066 degrees of freedom
Multiple R-squared:  0.7414,    Adjusted R-squared:  0.7378
F-statistic: 203.8 on 15 and 1066 DF,  p-value: < 2.2e-16
```

This model predicts fuel consumption based on both vehicle class (categorical) and engine size (continuous).

ANCOVA is traditionally taught as yet another distinct technique, but in the general linear model framework, it's simply a model that includes both categorical and continuous predictors.

ANCOVA: Mixing Categorical and Continuous Predictors

In this model, we're predicting fuel consumption based on vehicle class and engine size. The coefficients for vehicle class represent the differences between classes after controlling for engine size. The coefficient for engine size represents the effect of engine size on fuel consumption, controlling for vehicle class.

This is another example of how the general linear model provides a unified framework for various statistical techniques.

Effect Sizes: Understanding Practical Significance

Statistical significance (p-values) tells us if effects are likely real, but effect sizes tell us if they're practically important:

```
# Calculate effect sizes
eta_squared(anova_result)
```

```
# Effect Size for ANOVA

Parameter | Eta2 |       95% CI
-------------------------------
class     | 0.45 | [0.41, 1.00]

- One-sided CIs: upper bound fixed at [1.00].
```

Interpretation:

Effect Sizes: Understanding Practical Significance

- η^2 = proportion of variance explained by each factor
- Vehicle class explains about 43% of the variance in fuel consumption
- Values of 0.01, 0.06, and 0.14 are considered small, medium, and large effects

While p-values tell us whether an effect is statistically significant (unlikely to be due to chance), effect sizes tell us about the practical significance or magnitude of the effect.

For ANOVA, a common effect size is eta-squared (η^2), which represents the proportion of variance explained by each factor. Values around 0.01 are considered small, 0.06 medium, and 0.14 large.

The eta-squared value of 0.43 for vehicle class indicates that about 43% of the variance in fuel consumption is explained by vehicle class, which is a very large effect.

Effect sizes are important because with large enough sample sizes, even tiny, practically meaningless effects can become statistically significant.

Post-hoc Tests: Which Groups Differ?

When ANOVA finds significant differences, post-hoc tests help identify which specific groups differ:

```
# Calculate estimated marginal means
emm <- emmeans(lm_result, ~class)

# Pairwise comparisons with Tukey adjustment
pairs(emm) |>
  as_tibble() |>
  filter(p.value < 0.05) |>
  arrange(p.value) |>
  head(5) |>
  kable(digits = 3)
```

Post-hoc Tests: Which Groups Differ?

contrast	estimate	SE	df	t.ratio	p.value
COMPACT - (PICKUP TRUCK - STANDARD)	−4.905	0.340	1067	−14.407	0
COMPACT - (SUV - STANDARD)	−4.497	0.271	1067	−16.610	0
COMPACT - (VAN - PASSENGER)	−10.466	0.521	1067	−20.079	0
(FULL-SIZE) - (VAN - PASSENGER)	−7.962	0.548	1067	−14.528	0
(MID-SIZE) - (PICKUP TRUCK - STANDARD)	−4.625	0.340	1067	−13.586	0

When ANOVA indicates significant differences between groups, we often want to know which specific groups differ from each other. Post-hoc tests help answer this question.

Here we're using estimated marginal means and pairwise comparisons with Tukey's adjustment for multiple comparisons. The results show the estimated difference between each pair of vehicle classes, along with confidence intervals and adjusted p-values.

The table shows the 5 most significant pairwise differences. For example, fuel consumption differs significantly between SUV-UTILITY and COMPACT-SUV vehicle classes.

Post-hoc Tests: Which Groups Differ?

This is another example of how the linear model framework provides a comprehensive approach to statistical analysis, from overall tests to detailed comparisons.

Integrated Example: HR Analytics with ANOVA

Let's return to our HR dataset and use ANOVA to compare job satisfaction across job roles:

```
# Load HR Analytics dataset if not already loaded
if (!exists("hr_data")) {
  hr_data <- read_sav("data/dataset-abc-insurance-hr-data.sav") |>
janitor::clean_names()
}

# Run ANOVA for job satisfaction by department
hr_anova <- lm(job_satisfaction ~ job_role, data = hr_data)
summary(hr_anova)
```

```

Call:
lm(formula = job_satisfaction ~ job_role, data = hr_data)
```

Integrated Example: HR Analytics with ANOVA

```
Residuals:
     Min       1Q   Median       3Q      Max 
-2.67123 -0.82659  0.00448  0.83555  2.17341 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  2.65766    0.06886  38.597  < 2e-16 ***
job_role     0.16893    0.02155   7.839 1.23e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.103 on 934 degrees of freedom
Multiple R-squared:  0.06173,	Adjusted R-squared:  0.06073 
F-statistic: 61.45 on 1 and 934 DF,  p-value: 1.235e-14
```

Integrated Example: HR Analytics with ANOVA

Now let's apply what we've learned to our HR analytics dataset. Here we're comparing job satisfaction across different job roles using a linear model (which is equivalent to ANOVA).

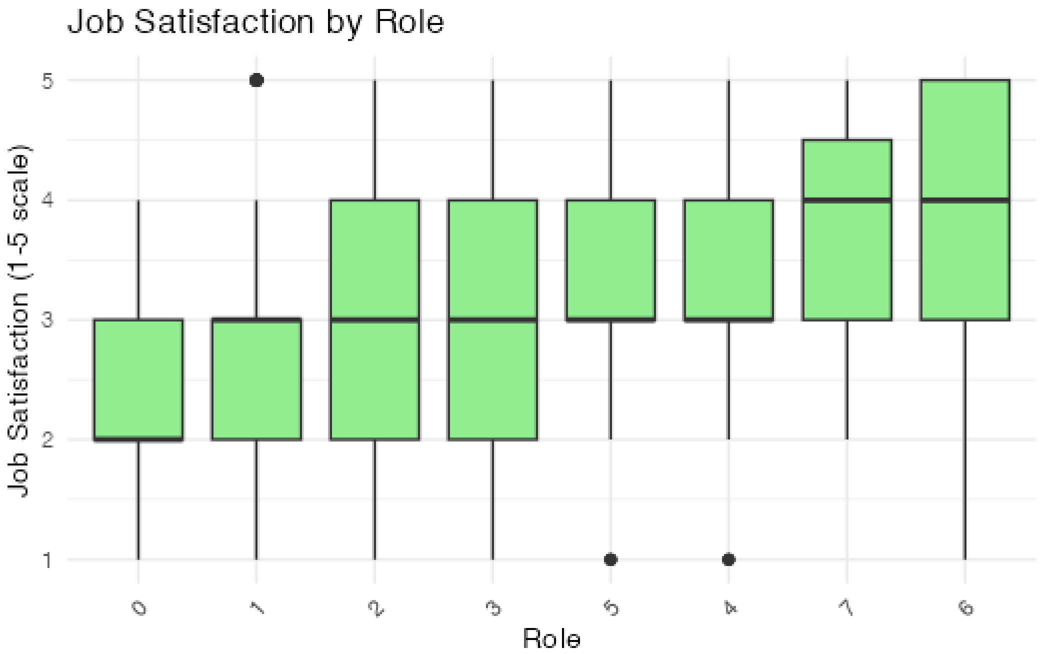
The results show the mean job satisfaction for the reference role (the intercept) and the differences between each other role and the reference role Some departments appear to have significantly higher or lower job satisfaction than others.

This is a practical application of ANOVA as a linear model in a human resources context.

Visualizing HR Department Differences

```r
# Create a visualization of satisfaction by department
ggplot(hr_data, aes(
  x = reorder(job_role, job_satisfaction, FUN = mean),
  y = job_satisfaction
)) +
  geom_boxplot(fill = "lightgreen") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(
    x = "Role", y = "Job Satisfaction (1-5 scale)",
    title = "Job Satisfaction by Role"
  )
```

Visualizing HR Department Differences



This visualization helps us see the differences in job satisfaction across departments. The departments are ordered by their mean job satisfaction, with departments having higher average satisfaction appearing towards the right.

Visualizing HR Department Differences

We can see variations in both the central tendency (median, indicated by the line in the middle of each box) and the spread of job satisfaction scores within each department.

This kind of analysis could help HR identify departments that might need intervention to improve employee satisfaction, or departments with particularly high satisfaction that might serve as models for others.

Combining ANOVA and Regression

We can build more complex models that include: - Multiple categorical predictors (multi-way ANOVA) - Continuous predictors alongside categorical ones (ANCOVA) - Interaction terms between predictors

```r
# Build a complex model
complex_model <- lm(job_satisfaction ~ job_role + gender + evaluation +
  job_role:evaluation, data = hr_data)

# View model summary
anova(complex_model) |>
  as_tibble() |>
  kable(digits = 3)
```

Df	Sum Sq	Mean Sq	F value	Pr(>F)
1	74.760	74.760	81.951	0.000

Combining ANOVA and Regression

Df	Sum Sq	Mean Sq	F value	Pr(>F)
1	1.478	1.478	1.620	0.203
1	285.426	285.426	312.880	0.000
1	0.101	0.101	0.111	0.739
931	849.307	0.912	NA	NA

Here we've built a more complex model that includes multiple predictors: department (categorical), gender (categorical), and performance rating (continuous), as well as an interaction between department and performance rating.

This model tests whether job satisfaction varies by department, gender, and performance rating, and whether the relationship between performance rating and job satisfaction differs across departments.

Combining ANOVA and Regression

The ANOVA table shows which effects are statistically significant. This demonstrates how the general linear model framework allows us to build and test complex models that would be difficult to conceptualize using traditional statistical procedures taught in isolation.

The Power of the Unified Approach

Benefits of viewing statistical tests as linear models:

1. Conceptual simplicity: Learn one framework instead of many isolated techniques

The Power of the Unified Approach

The Power of the Unified Approach

Benefits of viewing statistical tests as linear models:

1. Conceptual simplicity: Learn one framework instead of many isolated techniques
2. Flexibility: Easily combine and extend models to suit your research questions

The Power of the Unified Approach

The Power of the Unified Approach

Benefits of viewing statistical tests as linear models:

1. Conceptual simplicity: Learn one framework instead of many isolated techniques
2. Flexibility: Easily combine and extend models to suit your research questions
3. Interpretability: Consistent approach to understanding and communicating results

The Power of the Unified Approach

The Power of the Unified Approach

Benefits of viewing statistical tests as linear models:

1. Conceptual simplicity: Learn one framework instead of many isolated techniques
2. Flexibility: Easily combine and extend models to suit your research questions
3. Interpretability: Consistent approach to understanding and communicating results
4. Practicality: Simplifies implementation in statistical software

The Power of the Unified Approach

The Power of the Unified Approach

Benefits of viewing statistical tests as linear models:

1. Conceptual simplicity: Learn one framework instead of many isolated techniques
2. Flexibility: Easily combine and extend models to suit your research questions
3. Interpretability: Consistent approach to understanding and communicating results
4. Practicality: Simplifies implementation in statistical software
5. Extensibility: Natural pathway to more advanced methods (mixed effects, generalized linear models)

The unified linear model approach offers several benefits over the traditional approach of teaching statistical tests as separate, unrelated techniques.

First, it's conceptually simpler. Instead of learning different formulas and procedures for t-tests, ANOVA, regression, etc., you learn one framework that encompasses all of these.

The Power of the Unified Approach

Second, it's more flexible. You can easily combine different types of predictors and test complex hypotheses within the same framework.

Third, it provides a consistent approach to interpretation. The coefficients in a linear model always have the same basic interpretation, regardless of whether the model is implementing a t-test, ANOVA, or regression.

Fourth, it's practical. In R and many other statistical software packages, the linear model (lm() function in R) is the workhorse for a wide range of analyses.

Finally, it provides a natural pathway to more advanced methods like mixed-effects models and generalized linear models, which extend the linear model framework to handle more complex data structures and non-normal distributions.

Concluding Thoughts

- Statistical tests are not isolated tools but connected members of the same family
- The general linear model provides a unified framework for understanding these connections
- This perspective simplifies learning, application, and interpretation of statistics
- When facing a new analytical problem, think in terms of the linear model: what is my outcome? What are my predictors? What relationships am I testing?

In conclusion, the general linear model provides a powerful, unified framework for statistical analysis. By understanding that many common statistical tests are special cases of the linear model, we gain a deeper and more coherent understanding of statistics.

Rather than memorizing different formulas and procedures for different tests, we can focus on understanding the core principles of the linear model and how to apply them to different research questions.

Concluding Thoughts

When approaching a new analytical problem, thinking in terms of the linear model helps clarify the essential components: the outcome variable, the predictor variables, and the relationships we're interested in testing.

This approach not only simplifies learning and application but also enables us to build more sophisticated models that better capture the complexity of real-world phenomena.

Common Statistical Tests as Linear Models

The Unified Language of Statistics

Adapted from:

- *Common statistical tests are linear models*. Jonas Kristoffer Lindeløv (2019).

In this section, we'll explore an elegant insight: most common statistical tests can be expressed as special cases of the general linear model. This unified framework simplifies our understanding of statistics and reveals deep connections between seemingly different tests.

This section draws extensively from Jonas Lindeløv's excellent resource which demonstrates how common statistical tests can be expressed as linear models. This approach provides a powerful unifying framework that can transform how we teach and learn statistics.

The key insight is that tests like t-tests, ANOVA, correlation, and others aren't separate, unrelated techniques, but rather special cases of the same underlying model. By understanding this connection, students can develop a more coherent and transferable understanding of statistics.

The Simplicity Underlying Common Tests

Most statistical tests are special cases of linear models or very close approximations:

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \ldots + \varepsilon_i$$

This unified view simplifies learning and shows connections between seemingly different methods.

Teaching linear models first, then presenting traditional tests as special cases:

- Emphasizes understanding over memorization
- Makes statistical concepts more intuitive
- Shows the common structure of different statistical procedures

Linear models provide a unifying framework for understanding statistics. Most common statistical procedures (t-tests, ANOVA, correlation, etc.) are special cases of the general linear model.

The Simplicity Underlying Common Tests

This approach simplifies what students need to learn. Instead of treating each test as an independent entity with its own formulas and assumptions, we can present them as variations on the same underlying model.

By teaching the general linear model first and then showing how traditional tests are special cases, we help students build a more coherent mental model of statistics. This approach emphasizes conceptual understanding over rote memorization of formulas and procedures.

A Family Tree of Statistical Tests

Common statistical tests are linear models

Last updated: 02 April, 2019

See worked examples and more details at the accompanying notebook: https://lindeloev.github.io/tests-as-linear

	Common name	Built-in function in R	Equivalent linear model in R	Exact?	The linear model in words	Icon
Simple regression: lm(y ~ 1 + x)	**y is independent of x** P: One-sample t-test N: Wilcoxon signed-rank	t.test(y) wilcox.test(y)	lm(y ~ 1) lm(signed_rank(y) ~ 1)	✓ for N >14	One number (intercept, i.e., the mean) predicts **y**. - (Same, but it predicts the *signed rank* of **y**.)	
	P: Paired-sample t-test N: Wilcoxon matched pairs	t.test(y_1, y_2, paired=TRUE) wilcox.test(y_1, y_2, paired=TRUE)	lm(y_2 - y_1 ~ 1) lm(signed_rank(y_2 - y_1) ~ 1)	✓ for N >14	One intercept predicts the pairwise **y_2-y_1** differences. - (Same, but it predicts the *signed rank* of **y_2-y_1**.)	
	y ~ continuous x P: Pearson correlation N: Spearman correlation	cor.test(x, y, method='Pearson') cor.test(x, y, method='Spearman')	lm(y ~ 1 + x) lm(rank(y) ~ 1 + rank(x))	✓ for N >10	One intercept plus **x** multiplied by a number (slope) predicts **y**. - (Same, but with *ranked* **x** and **y**)	
	y ~ discrete x P: Two-sample t-test P: Welch's t-test N: Mann-Whitney U	t.test(y_1, y_2, var.equal=TRUE) t.test(y_1, y_2, var.equal=FALSE) wilcox.test(y_1, y_2)	lm(y ~ 1 + G_2)[A] gls(y ~ 1 + G_2, weights=…[B])[A] lm(signed_rank(y) ~ 1 + G_2)[A]	✓ ✓ for N >11	An intercept for **group 1** (plus a difference if **group 2**) predicts **y**. - (Same, but with one variance *per group* instead of one common.) - (Same, but it predicts the *signed rank* of **y**.)	
Multiple regression: lm(y ~ 1 + x_1 + x_2 +…)	P: One-way ANOVA N: Kruskal-Wallis	aov(y ~ group) kruskal.test(y ~ group)	lm(y ~ 1 + G_2 + G_3 +…+ G_N)[A] lm(rank(y) ~ 1 + G_2 + G_3 +…+ G_N)[A]	✓ for N >11	An intercept for **group 1** (plus a difference if group ≠ 1) predicts **y**. - (Same, but it predicts the *rank* of **y**.)	
	P: One-way ANCOVA	aov(y ~ group + x)	lm(y ~ 1 + G_2 + G_3 +…+ G_N + x)[A]	✓	- (Same, but plus a slope on **x**.) *Note: this is discrete AND continuous. ANCOVAs are ANOVAs with a continuous x.*	
	P: Two-way ANOVA	aov(y ~ group * sex)	lm(y ~ 1 + G_2 + G_3 + … + G_N + S_2 + S_3+ … + S_K + G_2*S_2+G_3*S_3+…+G_N*S_K)	✓	Interaction term: changing **sex** changes the **y ~ group** parameters. *Note: **$G_{2 \text{ to } N}$** is an indicator (0 or 1) for each non-intercept levels of the **group** variable. Similarly for **$S_{2 \text{ to } K}$** for sex. The first line (with G_i) is main effect of group, the second (with S_i) for sex and the third is the **group × sex** interaction. For two levels (e.g. male/female), line 2 would just be "S_2" and line 3 would be S_2 multiplied with each G_i.*	*[Coming]*
	Counts ~ discrete x N: Chi-square test	chisq.test(groupXsex_table)	**Equivalent log-linear model** glm(y ~ 1 + G_2 + G_3 + … + G_N + S_2 + S_3+ … + S_K + G_2*S_2+G_3*S_3+…+G_N*S_K, family=…)[A]	✓	Interaction term: (Same as Two-way ANOVA.) *Note: Run glm using the following arguments: `glm(model, family=poisson())` As linear-model, the Chi-square test is $\log(y_i) = \log(N) + \log(\alpha_i) + \log(\beta_j) + \log(\alpha_i\beta_j)$ where α_i and β_j are proportions. See more info in the accompanying notebook.*	*Same as Two-way ANOVA*
	N: Goodness of fit	chisq.test(y)	glm(y ~ 1 + G_2 + G_3 +…+ G_N, family=…)[A]	✓	(Same as One-way ANOVA and see Chi-Square note.)	*1W-ANOVA*

List of common parametric (P) non-parametric (N) tests and equivalent linear models. The notation y ~ 1 + x is R shorthand for y = 1·b + a·x which most of us learned in school. Models in similar colors are highly similar, but really, notice how similar they *all* are across colors! For non-parametric models, the linear models are reasonable approximations for non-small sample sizes (see "Exact" column and click links to see simulations). Other less accurate approximations exist, e.g., Wilcoxon for the sign test and Goodness-of-fit for the binomial test. The signed rank function is `signed_rank = function(x) sign(x) * rank(abs(x))`. The variables G_i and S_i are "dummy coded" indicator variables (either 0 or 1) exploiting the fact that when Δx = 1 between categories the difference equals the slope. Subscripts (e.g., G_2 or y_1) indicate different columns in data. lm requires long-format data for all non-continuous models. All of this is exposed in greater detail and worked examples at https://lindeloev.github.io/tests-as-linear.

[A] See the note to the two-way ANOVA for explanation of the notation.
[B] Same model, but with one variance per group: `gls(value ~ 1 + G2, weights = varIdent(form = ~1|group), method="ML")`.

Jonas Kristoffer Lindeløv
https://lindeloev.net

Figure 6: A family tree of statistical tests as linear models

A Family Tree of Statistical Tests

This cheat sheet shows how different statistical tests relate to each other through the linear model framework:

- Simple tests at the bottom (t-tests, correlation)
- More complex models at the top (ANOVA, multiple regression)
- Each branch represents a variation or special case of the linear model

This cheat sheet from Lindeløv's website shows how various statistical tests are related through the linear model framework. It provides a visual roadmap of the connections between different statistical procedures.

Notice how we can trace the path from simple tests like t-tests up to more complex procedures like factorial ANOVA. Each branch represents a variation or extension of the basic linear model.

A Family Tree of Statistical Tests

This visualization helps students see that what they're learning aren't disconnected techniques but rather members of the same family, with shared properties and interpretations. This perspective can make learning statistics more coherent and less overwhelming.

Simplifying Our Understanding: Summary Table

Test	Linear Model Formula	What's being tested
Correlation	y ~ x	Slope coefficient
One-sample t-test	y ~ 1	Intercept
Independent t-test	y ~ group	Group coefficient
Paired t-test	diff ~ 1	Intercept of differences
One-way ANOVA	y ~ group	Group coefficients
Two-way ANOVA	y ~ factorA * factorB	Main effects & interaction
Multiple regression	y ~ x1 + x2 + …	Multiple coefficients

The table shows:

- Each common test has a corresponding linear model formulation
- Many tests are testing coefficients in the same type of model
- The differences often come down to which coefficients we're interested in

Simplifying Our Understanding: Summary Table

This table summarizes how different common tests map to linear model formulations. For each test, we can identify what linear model would be equivalent and which coefficient(s) we're testing.

Notice that the difference between tests often comes down to:

1. What variables we include in the model
2. Which coefficient(s) we're interested in testing
3. How we interpret the results

This unified framework helps students see that they're not learning completely different procedures for each test, but rather applying the same underlying model in different contexts.

Pearson and Spearman Correlation as Linear Models

Model: $y = \beta_0 + \beta_1 x \quad$ where $\mathcal{H}_0 : \beta_1 = 0$

This is simply a linear regression with one predictor. When we test whether the correlation is significant, we're testing whether the slope (β_1) differs from zero.

```
# Create example data
set.seed(42)
x <- rnorm(50)
y <- 0.6 * x + rnorm(50, 0, 0.8)
data <- data.frame(x = x, y = y)

# Traditional correlation
cor_result <- cor.test(data$x, data$y)
cor_result$estimate
```

Pearson and Spearman Correlation as Linear Models

```
      cor
0.5818111
```

```
cor_result$p.value
```

```
[1] 9.360699e-06
```

```
# As linear model (standardized variables)
lm_cor <- lm(scale(y) ~ scale(x), data = data)
coef(lm_cor)[2] # slope = correlation coefficient
```

```
 scale(x)
0.5818111
```

Pearson and Spearman Correlation as Linear Models

```r
summary(lm_cor)$coefficients[2, "Pr(>|t|)"] # p-value
```

```
[1] 9.360699e-06
```

When we standardize both variables (giving them mean=0 and sd=1), the slope coefficient equals the correlation coefficient!

Here we demonstrate that Pearson's correlation is equivalent to the standardized regression coefficient in a simple linear regression model.

The mathematical model is exactly the same as simple linear regression: $y = \beta_0 + \beta_1 x + \varepsilon$. The null hypothesis being tested is that $\beta_1 = 0$, which means there is no linear relationship between the variables.

When we standardize both x and y (to have mean=0 and sd=1), the slope coefficient in a linear regression equals the correlation coefficient r. This makes intuitive sense because

Pearson and Spearman Correlation as Linear Models

standardization puts both variables on the same scale, making their relationship directly comparable.

The t-test on this coefficient tests exactly the same hypothesis as the correlation test: is there a linear relationship between the variables? The p-values are identical between the two approaches.

This equivalence helps us understand correlation not as a mysterious measure, but simply as the slope of a regression line when variables are standardized.

Pearson vs. Spearman Correlation

Spearman correlation is Pearson correlation on rank-transformed variables:

```r
# Pearson correlation on original data
cor(x, y, method = "pearson")
```

```
[1] 0.5818111
```

```r
# Spearman correlation = Pearson on ranks
cor(x, y, method = "spearman")
```

```
[1] 0.6436975
```

```r
# Same as Pearson correlation on ranked variables
cor(rank(x), rank(y), method = "pearson")
```

Pearson vs. Spearman Correlation

```
[1] 0.6436975
```

```
# As linear model with ranks
lm_spearman <- lm(rank(y) ~ rank(x))
summary(lm_spearman)$coefficients[2, "Estimate"]
```

```
[1] 0.6436975
```

The "non-parametric" Spearman correlation is simply the "parametric" Pearson correlation applied to ranked data!

Spearman's rank correlation is a brilliant example of how a "non-parametric" test is simply a parametric test applied to transformed data.

Pearson vs. Spearman Correlation

Instead of correlating the original values, Spearman correlation first converts all values to their ranks (1st, 2nd, 3rd, etc.) and then applies the Pearson correlation formula to these ranks.

This transformation accomplishes two things:

1. It makes the test robust to outliers, since extreme values just become the highest or lowest rank
2. It allows the test to detect monotonic but non-linear relationships, since ranking linearizes any monotonic relationship

By understanding Spearman correlation as "Pearson on ranks," we demystify non-parametric statistics. Many so-called non-parametric tests are simply parametric tests applied to transformed data, making them more accessible conceptually.

Pearson vs. Spearman Correlation

The R code demonstrates that Spearman correlation can be obtained either by using the dedicated function or by manually ranking the variables and then applying Pearson correlation.

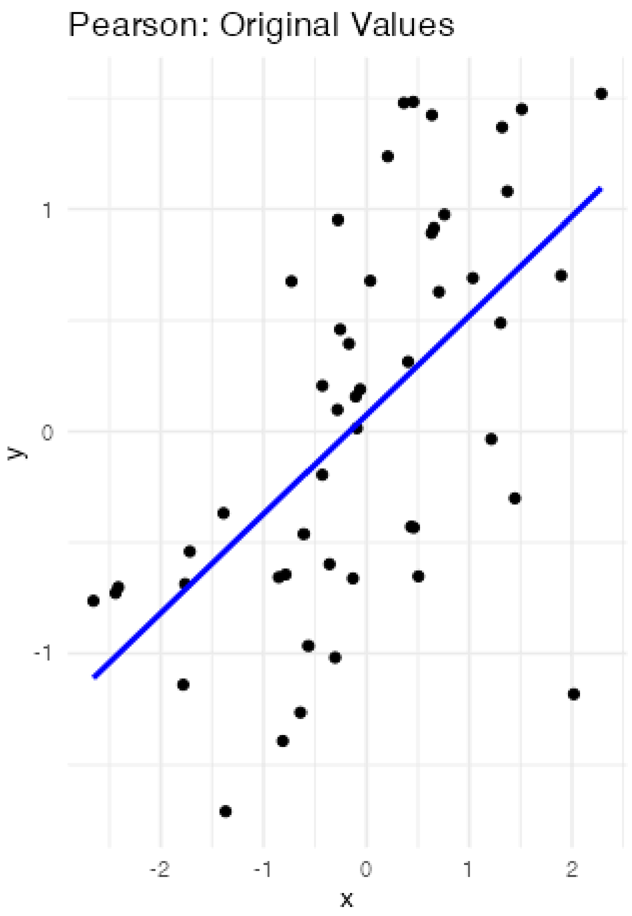
Correlation Visualized

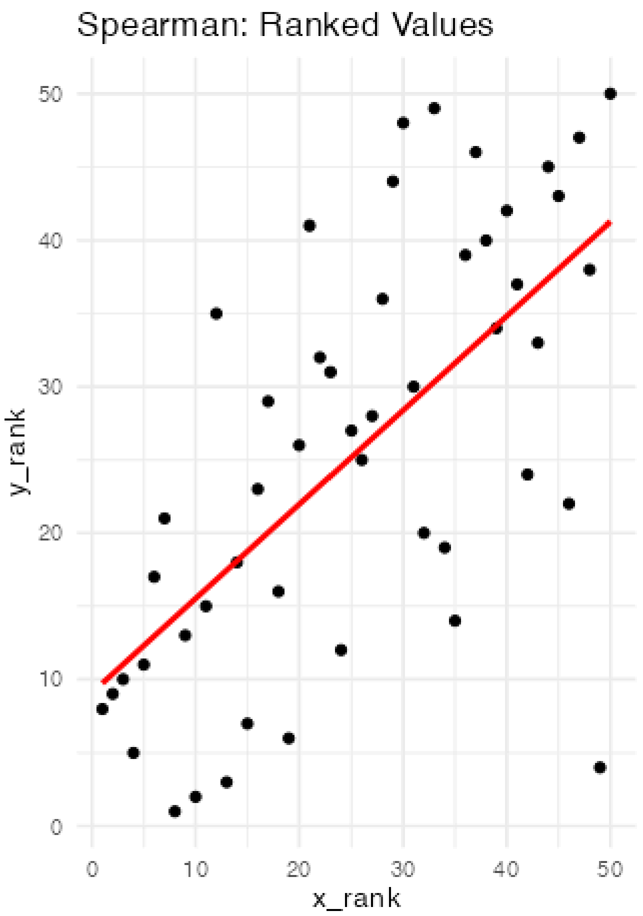
```r
p1 <- ggplot(data, aes(x = x, y = y)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE, color = "blue") +
    labs(title = "Pearson: Original Values") +
    theme_minimal()

rank_data <- data.frame(x_rank = rank(x), y_rank = rank(y))
p2 <- ggplot(rank_data, aes(x = x_rank, y = y_rank)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE, color = "red") +
    labs(title = "Spearman: Ranked Values") +
    theme_minimal()

p1 + p2
```

Correlation Visualized





Correlation Visualized

Left: Pearson correlation fits a line to the original data points
Right: Spearman correlation fits a line to the ranked data points

This visualization helps us understand the relationship between Pearson and Spearman correlation.

The left panel shows the original data with the regression line (Pearson's r). The right panel shows the same data after converting to ranks, with its regression line (Spearman's rho).

Notice how the ranked data (right panel) tends to form a more linear pattern. This is because ranking removes the influence of outliers and transforms any monotonic relationship into a linear one.

Another key insight: the slope of the line through the ranked data is the Spearman correlation coefficient, just as the slope of the line through the standardized original data is the Pearson correlation coefficient.

Correlation Visualized

This visualization reinforces the idea that many statistical tests are simply variations on the same theme, applied to differently transformed data.

One-Sample t-test as a Linear Model

Model: $y = \beta_0 \quad$ where $\mathcal{H}_0 : \beta_0 = 0$

This is the simplest linear model possible! It has only an intercept (no predictors), and the intercept equals the sample mean.

```r
# Create sample data
set.seed(123)
y_one <- rnorm(30, mean = 5.2, sd = 2)

# Traditional t-test
t_test_one <- t.test(y_one, mu = 5)
t_test_one$statistic
```

```
        t
0.2953268
```

One-Sample t-test as a Linear Model

```r
t_test_one$p.value
```

```
[1] 0.7698484
```

```r
t_test_one$conf.int
```

```
[1] 4.373147 5.838438
attr(,"conf.level")
[1] 0.95
```

```r
# Same test as linear model
lm_one <- lm(y_one ~ 1)
summary(lm_one)$coefficients
```

One-Sample t-test as a Linear Model

```
            Estimate Std. Error  t value     Pr(>|t|)
(Intercept) 5.105792  0.3582218 14.25316 1.244655e-14
```

```r
confint(lm_one)
```

```
               2.5 %   97.5 %
(Intercept) 4.373147 5.838438
```

The intercept-only model gives identical results to the one-sample t-test! The coefficient is the mean, and the t-statistic tests if it differs from zero.

The one-sample t-test is perhaps the simplest demonstration of how standard statistical tests are special cases of linear models.

One-Sample t-test as a Linear Model

In a one-sample t-test, we're asking whether a sample mean differs significantly from a hypothesized population value (often zero). In the linear model framework, this becomes an intercept-only model: $y = \beta_0 + \varepsilon$.

The intercept (β_0) represents the sample mean. The t-statistic tests whether this mean differs significantly from the hypothesized value (in this case, $\mu=5$).

The R code demonstrates this equivalence beautifully. The estimate from t.test() is identical to the intercept from lm(), and the t-statistic, p-value, and confidence intervals match exactly.

This shows how even the most basic statistical test can be understood within the general linear model framework. The intercept-only model is simply a special case where we have no predictors, just as the one-sample t-test is examining a single mean without comparison groups.

Wilcoxon Signed-Rank Test as a Linear Model

The Wilcoxon signed-rank test is approximately a one-sample t-test on signed ranks:

```
# Traditional Wilcoxon test
w_test <- wilcox.test(y_one - 5) # Test against μ=5
w_test$p.value
```

```
[1] 0.8552717
```

```
# Define the signed rank function
signed_rank <- function(x) sign(x) * rank(abs(x))

# As linear model on signed ranks
lm_wilcox <- lm(signed_rank(y_one - 5) ~ 1)
summary(lm_wilcox)$coefficients[1, "Pr(>|t|)"]
```

Wilcoxon Signed-Rank Test as a Linear Model

```
[1] 0.8488961
```

The "non-parametric" Wilcoxon test can be viewed as a one-sample t-test on rank-transformed data!

The Wilcoxon signed-rank test is often presented as a completely different "non-parametric" alternative to the t-test, but here we see it's closely related to the t-test when viewed through the linear model lens.

The key insight is that the Wilcoxon test can be approximated as a one-sample t-test applied to signed ranks rather than the original values. Here's how it works:

1. First, we calculate the differences from the hypothesized median (μ=5)
2. Then we rank the absolute differences (ignoring signs)
3. Finally, we reattach the original signs to these ranks (creating "signed ranks")
4. We run a one-sample t-test on these signed ranks

Wilcoxon Signed-Rank Test as a Linear Model

This transformation makes the test more robust to outliers and non-normal distributions, as extreme values are "tamed" by the ranking process.

The approximation works best with sample sizes of 15 or more. With smaller samples, the discrete nature of ranks means the p-values won't match exactly, but the approach still provides conceptual insight.

This demonstrates again how "non-parametric" tests can often be understood as parametric tests applied to transformed data, demystifying what might otherwise seem like a completely different approach to inference.

Independent Samples t-test as a Linear Model

Model: $y_i = \beta_0 + \beta_1 x_i \quad$ where $\mathcal{H}_0 : \beta_1 = 0$

Here, x_i is a dummy variable (0/1) for group membership. β_0 represents the mean of the first group, while β_1 represents the difference between groups.

```r
# Create data for two groups
set.seed(456)
group <- rep(c("A", "B"), each = 15)
y_ind <- c(
  rnorm(15, mean = 10, sd = 2),
  rnorm(15, mean = 12, sd = 2)
)
ind_data <- data.frame(y = y_ind, group = factor(group))

# Traditional t-test
```

Independent Samples t-test as a Linear Model

```r
t_test_ind <- t.test(y ~ group, data = ind_data, var.equal = TRUE)
t_test_ind$statistic
```

```
       t
-2.87084
```

```r
t_test_ind$p.value
```

```
[1] 0.007712189
```

```r
# Same test as linear model
lm_ind <- lm(y ~ group, data = ind_data)
summary(lm_ind)$coefficients
```

Independent Samples t-test as a Linear Model

```
              Estimate Std. Error  t value     Pr(>|t|)
(Intercept) 10.236098  0.6046277 16.92959 3.052810e-16
groupB       2.454777  0.8550727  2.87084 7.712189e-03
```

The coefficient for `groupB` is the difference between groups, exactly what's tested in the t-test. The t-statistic and p-value are identical!

The independent samples t-test compares means between two groups. In the linear model framework, this is represented as a model with one dummy-coded categorical predictor.

The model is $y = \beta_0 + \beta_1 x + \varepsilon$, where x is coded as 0 for the first group and 1 for the second group. This dummy coding has a straightforward interpretation:

- β_0 (the intercept) represents the mean of the reference group (group A)
- β_1 represents the difference in means between groups B and A
- The t-statistic tests whether this difference is significantly different from zero

Independent Samples t-test as a Linear Model

The R code demonstrates that the t-statistic and p-value from the traditional t-test are identical to those for the group coefficient in the linear model.

This demonstrates how categorical variables can be incorporated into linear models through dummy coding, and how tests that might seem conceptually different (like t-tests and regression) are actually part of the same unified framework.

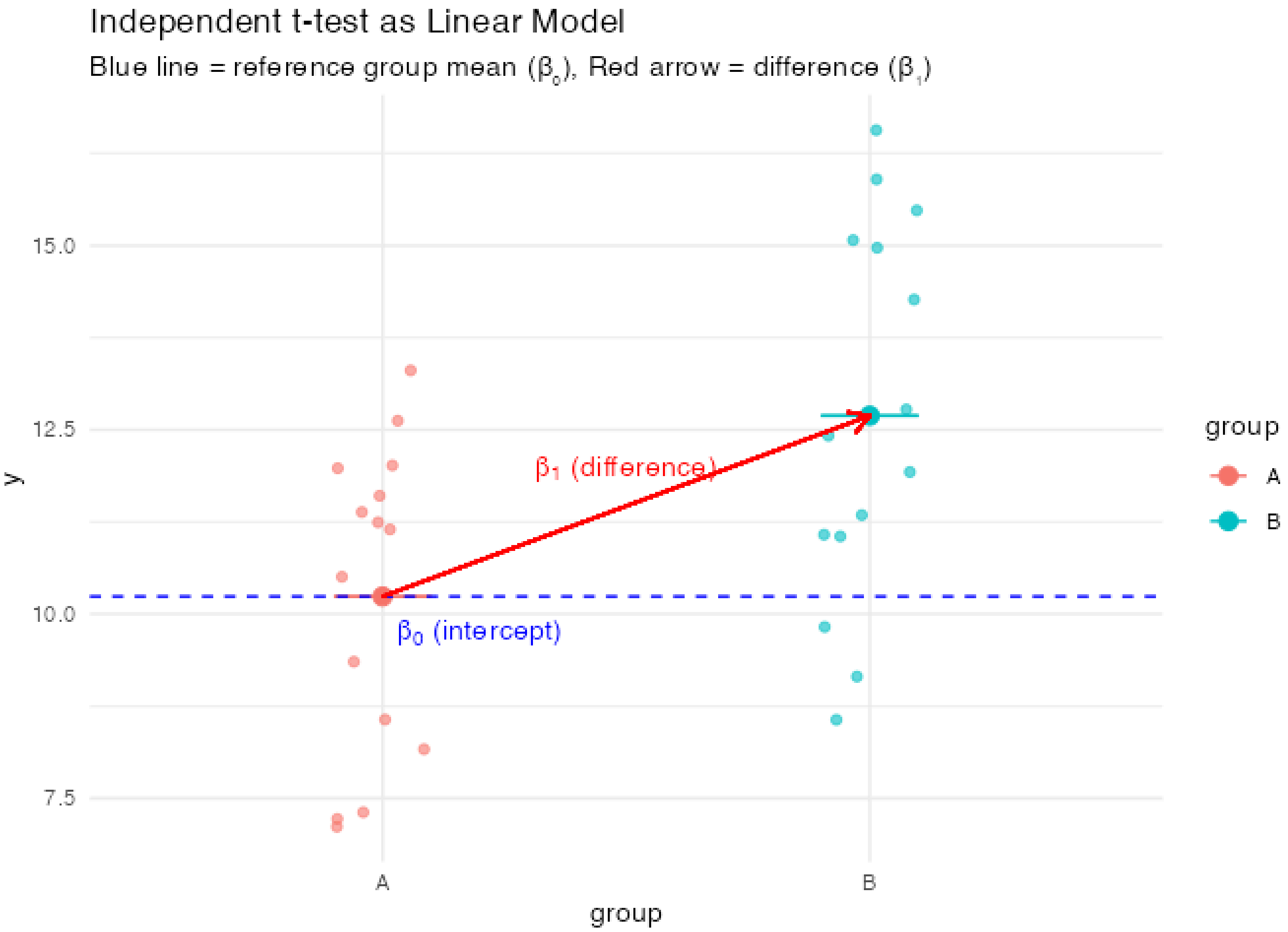
Dummy Coding Visualized

```
# Plot with jittered points and means
ggplot(ind_data, aes(x = group, y = y, color = group)) +
  geom_jitter(width = 0.1, alpha = 0.6) +
  stat_summary(fun = mean, geom = "point", size = 3) +
  stat_summary(fun = mean, geom = "errorbar",
               aes(ymax = ..y.., ymin = ..y..), width = 0.2) +
  geom_hline(yintercept = coef(lm_ind)[1], linetype = "dashed", color =
"blue") +
  geom_segment(x = 1, xend = 2,
               y = coef(lm_ind)[1], yend = coef(lm_ind)[1] + coef(lm_ind)[2],
               color = "red", arrow = arrow(length = unit(0.3, "cm"))) +
  annotate("text", x = 1.2, y = coef(lm_ind)[1] - 0.5,
           label = expression(beta[0]~"(intercept)"), color = "blue") +
  annotate("text", x = 1.5, y = coef(lm_ind)[1] + coef(lm_ind)[2]/2 + 0.5,
           label = expression(beta[1]~"(difference)"), color = "red") +
```

Dummy Coding Visualized

```
  theme_minimal() +
  labs(title = "Independent t-test as Linear Model",
       subtitle = "Blue line = reference group mean (β₀), Red arrow =
difference (β₁)")
```

Dummy Coding Visualized



Dummy Coding Visualized

This visualization shows:

- The blue dashed line is the mean of group A (the intercept, β_0)
- The red arrow shows the difference between groups (the slope, β_1)
- In a t-test, we're testing whether this difference (β_1) is significantly different from zero

This visualization helps us understand how dummy coding works in a linear model with a categorical predictor.

When we use dummy coding in a linear model:

1. One group (here, group A) becomes the reference category and is coded as 0
2. The other group (group B) is coded as 1
3. The intercept (β_0) represents the mean of the reference group
4. The coefficient for the dummy variable (β_1) represents the difference between groups

Dummy Coding Visualized

In the plot, the blue dashed line shows the mean of group A (the intercept, β_0). The red arrow represents the difference between groups, which is the coefficient β_1.

This visualization makes it clear that an independent samples t-test is testing whether this difference (β_1) is significantly different from zero. If there's no difference between groups, the red arrow would be flat (no vertical component).

Understanding dummy coding is crucial for interpreting linear models with categorical predictors. This same principle extends to models with multiple categorical predictors (like ANOVA) and to more complex designs.

Mann-Whitney U Test as a Linear Model

The Mann-Whitney U test is approximately a t-test on ranks:

```r
# Traditional Mann-Whitney U test
mw_test <- wilcox.test(y ~ group, data = ind_data)
mw_test$p.value
```

```
[1] 0.02635404
```

```r
# As linear model on ranks
lm_mw <- lm(rank(y) ~ group, data = ind_data)
summary(lm_mw)$coefficients[2, "Pr(>|t|)"]
```

```
[1] 0.0236538
```

Mann-Whitney U Test as a Linear Model

```
# Extract estimates
data.frame(
  Test = c("Mann-Whitney U", "Linear model on ranks"),
  P_value = c(mw_test$p.value, summary(lm_mw)$coefficients[2, "Pr(>|t|)"]),
  Difference = c(NA, coef(lm_mw)[2])
) |> kable(digits = 4)
```

	Test	P_value	Difference
	Mann-Whitney U	0.0264	NA
groupB	Linear model on ranks	0.0237	7.1333

Just like with Spearman correlation, the "non-parametric" Mann-Whitney test can be viewed as a regular t-test applied to ranked data.

Mann-Whitney U Test as a Linear Model

The Mann-Whitney U test (also known as the Wilcoxon rank-sum test) is commonly presented as a "non-parametric" alternative to the independent samples t-test when data violates normality assumptions.

However, as we can see, it can be closely approximated as a standard t-test (or linear model) applied to ranked data:

1. First, we rank all values across both groups from lowest to highest
2. Then we run a standard t-test (or linear model) comparing these ranks between groups
3. The coefficient for the group effect represents the difference in mean ranks

The p-value from this ranked linear model closely approximates the p-value from the Mann-Whitney U test. This approximation improves with larger sample sizes ($n > 20$ per group).

Mann-Whitney U Test as a Linear Model

This approach gives us an additional benefit: while the traditional Mann-Whitney U test only provides a p-value, the linear model on ranks also gives us the actual difference in mean ranks between groups, providing a measure of effect size.

This further reinforces the pattern we've seen: many "non-parametric" tests can be understood as parametric tests applied to rank-transformed data, providing a unified conceptual framework.

Paired Samples t-test as a Linear Model

Model: $y_{2i} - y_{1i} = \beta_0$ where $\mathcal{H}_0 : \beta_0 = 0$

A paired t-test simplifies to a one-sample t-test on the differences between pairs!

```r
# Create paired data
set.seed(789)
pre <- rnorm(20, mean = 100, sd = 15)
post <- pre + rnorm(20, mean = 8, sd = 10) # Correlated with pre
paired_data <- data.frame(
  subject = 1:20,
  pre = pre,
  post = post
)

# Traditional paired t-test
```

Paired Samples t-test as a Linear Model

```r
t_test_paired <- t.test(paired_data$post, paired_data$pre, paired = TRUE)
t_test_paired$statistic
```

```
       t 
3.189437 
```

```r
t_test_paired$p.value
```

```
[1] 0.004827114
```

```r
# Calculate differences and fit model
paired_data$diff <- paired_data$post - paired_data$pre
lm_paired <- lm(diff ~ 1, data = paired_data)
summary(lm_paired)$coefficients
```

Paired Samples t-test as a Linear Model

```
             Estimate Std. Error  t value    Pr(>|t|)
(Intercept) 5.089399   1.595704 3.189437 0.004827114
```

The paired t-test becomes an intercept-only model (one-sample t-test) on the differences!

The paired samples t-test is another example of how complex-seeming statistical tests reduce to simpler linear models when we understand the underlying structure.

In a paired design, we have two measurements for each subject (e.g., before and after treatment). The paired t-test accounts for the correlation between these measurements by analyzing the differences rather than the raw values.

The key insight is that a paired t-test is mathematically equivalent to a one-sample t-test on the differences:

1. Calculate the difference for each pair: diff = post - pre

Paired Samples t-test as a Linear Model

2. Test whether the mean difference is significantly different from zero using a one-sample t-test

In the linear model framework, this becomes an intercept-only model on the differences: diff $= \beta_0 + \varepsilon$, where the null hypothesis is $\beta_0 = 0$.

The R code demonstrates this equivalence. The t-statistic and p-value from the paired t-test match exactly those from the intercept-only model on the differences.

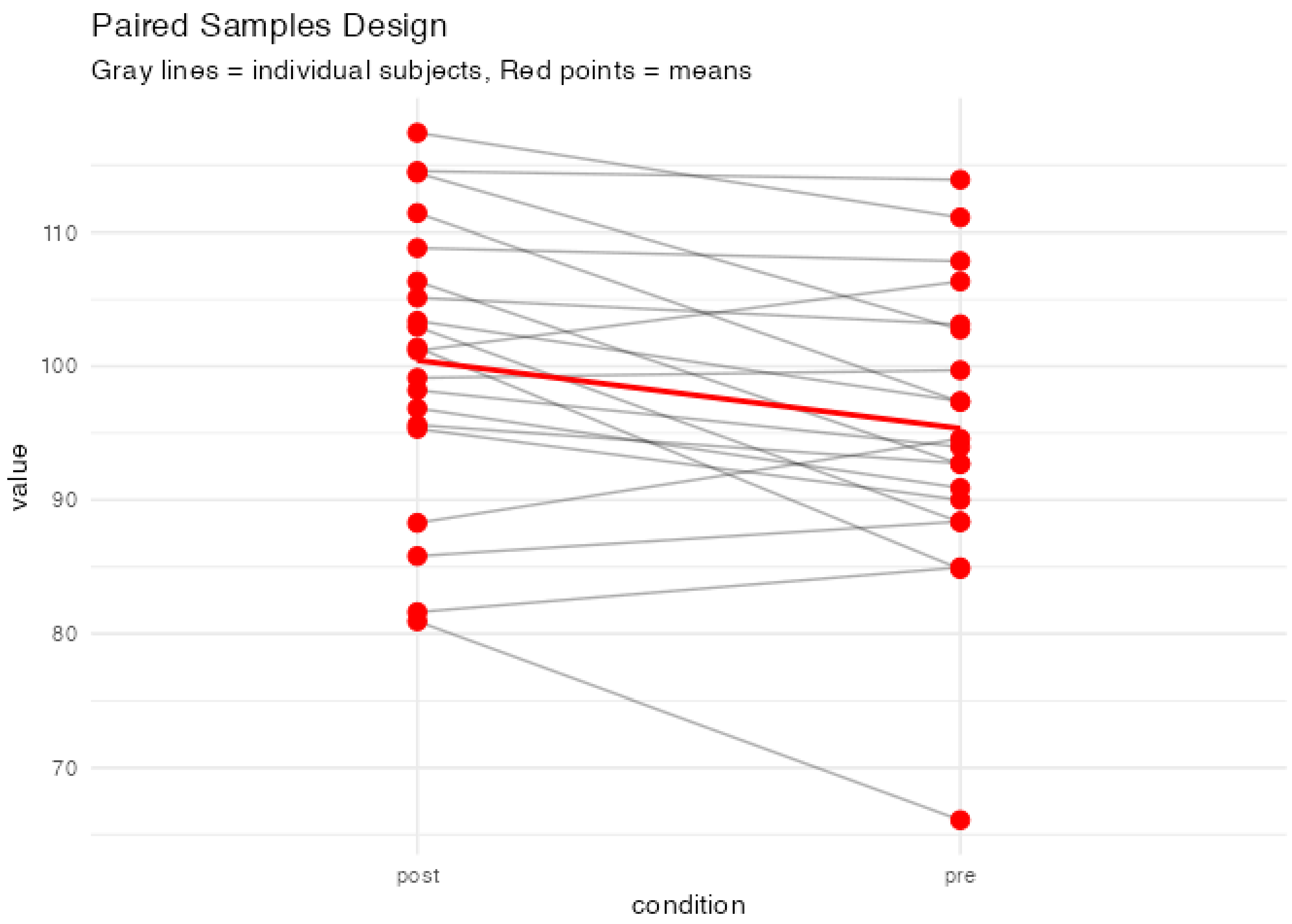
This approach clarifies that the paired t-test is not a fundamentally different test but rather a clever application of the one-sample t-test to difference scores. This insight helps students see the connections between seemingly different statistical procedures.

Paired Design Visualization

```r
# Convert to long format for plotting
paired_long <- pivot_longer(paired_data, cols = c("pre", "post"),
                            names_to = "condition", values_to = "value")

# Plot
ggplot(paired_long, aes(x = condition, y = value, group = subject)) +
  geom_line(alpha = 0.3) +
  geom_point(alpha = 0.3) +
  stat_summary(fun = mean, geom = "point", size = 3, color = "red") +
  stat_summary(fun = mean, geom = "line", size = 1, color = "red",
               aes(group = 1)) +
  theme_minimal() +
  labs(title = "Paired Samples Design",
       subtitle = "Gray lines = individual subjects, Red points = means")
```

Paired Design Visualization



Paired Design Visualization

This visualization shows:

- Each gray line represents one subject's pre and post measurements
- The red points show the group means at each time point
- The paired t-test analyzes the mean of these differences (slopes of gray lines)
- This accounts for individual baseline differences and increases statistical power

This visualization helps us understand the structure of paired data and why we analyze differences rather than raw values.

Each gray line represents a single subject, connecting their pre and post measurements. Notice how subjects start at different baseline levels but generally show similar trends (most lines slope upward, indicating an increase from pre to post).

Paired Design Visualization

The red points and line show the group means at each time point. The paired t-test effectively tests whether the average slope of the gray lines is significantly different from zero.

The key advantage of a paired design is that it accounts for individual differences. In an independent samples design, the pre-test variance would include both within-subject and between-subject variability. By analyzing within-subject changes, we remove the between-subject variability, resulting in greater statistical power.

This is why paired designs are often preferred when the same subjects can be measured under different conditions - they control for individual differences that would otherwise contribute to error variance.

Understanding paired designs as analyzing differences helps students see why the paired t-test reduces to a one-sample t-test on those differences, as we saw in the previous slide.

One-way ANOVA as a Linear Model

Model: $y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i} + \ldots \quad$ where $\mathcal{H}_0 : \beta_1 = \beta_2 = \ldots = 0$

The one-way ANOVA is a natural extension of the independent t-test to three or more groups.

```r
# Create example data with 3 groups
set.seed(101)
n_per_group <- 15
group <- factor(rep(c("A", "B", "C"), each = n_per_group))
means <- c(10, 12, 8)
y_anova <- c(
  rnorm(n_per_group, mean = means[1], sd = 2),
  rnorm(n_per_group, mean = means[2], sd = 2),
  rnorm(n_per_group, mean = means[3], sd = 2)
)
anova_data <- data.frame(y = y_anova, group = group)
```

One-way ANOVA as a Linear Model

```r
# Traditional ANOVA
anova_result <- aov(y ~ group, data = anova_data)
summary(anova_result)
```

```
            Df Sum Sq Mean Sq F value   Pr(>F)    
group        2  108.0   53.97   14.83 1.34e-05 ***
Residuals   42  152.9    3.64                     
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# As linear model
lm_anova <- lm(y ~ group, data = anova_data)
anova(lm_anova)
```

One-way ANOVA as a Linear Model

```
Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value    Pr(>F)    
group      2 107.95  53.973  14.827 1.343e-05 ***
Residuals 42 152.88   3.640                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The linear model produces exactly the same F-statistic and p-value as the traditional ANOVA. The linear model is using dummy coding for groups B and C, with group A as the reference.

One-way ANOVA is traditionally taught as a distinct test from regression or t-tests, but here we see it's simply an extension of the same linear model framework.

One-way ANOVA as a Linear Model

In an independent t-test, we had one dummy variable for two groups. In one-way ANOVA with k groups, we have k-1 dummy variables:

- Group A becomes the reference group (coded as 0 for all dummy variables)
- Group B is coded as 1 for the first dummy variable, 0 for others
- Group C is coded as 1 for the second dummy variable, 0 for others
- And so on for additional groups

The model is: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \varepsilon$, where:

- β_0 is the mean of the reference group (Group A)
- β_1 is the difference between Group B and Group A
- β_2 is the difference between Group C and Group A

The F-test in the ANOVA table tests the null hypothesis that all group differences are simultaneously equal to zero ($\beta_1 = \beta_2 = \ldots = 0$).

One-way ANOVA as a Linear Model

The R code demonstrates that traditional ANOVA (using aov()) and the linear model approach (using lm() followed by anova()) produce identical F-statistics and p-values.

This reveals that ANOVA is not a fundamentally different procedure but simply a way of testing multiple coefficients simultaneously in a linear model.

Kruskal-Wallis Test as a Linear Model

The Kruskal-Wallis test is approximately a one-way ANOVA on ranks:

```
# Traditional Kruskal-Wallis test
kw_test <- kruskal.test(y ~ group, data = anova_data)
kw_test$p.value
```

```
[1] 0.0001543885
```

```
# As linear model on ranks
lm_kw <- lm(rank(y) ~ group, data = anova_data)
anova(lm_kw)$"Pr(>F)"[1]
```

```
[1] 2.278855e-05
```

Kruskal-Wallis Test as a Linear Model

```r
# Extract estimates and p-values
data.frame(
  Test = c("Kruskal-Wallis", "ANOVA on ranks"),
  P_value = c(kw_test$p.value, anova(lm_kw)$"Pr(>F)"[1])
) |> kable(digits = 5)
```

Test	P_value
Kruskal-Wallis	0.00015
ANOVA on ranks	0.00002

Following our pattern, the "non-parametric" Kruskal-Wallis test can be viewed as a regular ANOVA performed on ranked data rather than raw values.

The Kruskal-Wallis test is traditionally presented as a non-parametric alternative to one-way ANOVA when data violate normality assumptions or are ordinal in nature.

Kruskal-Wallis Test as a Linear Model

However, just as we saw with other “non-parametric” tests, the Kruskal-Wallis test can be closely approximated as a standard parametric test (one-way ANOVA) applied to rank-transformed data:

1. First, we rank all observations from lowest to highest, regardless of group
2. Then we run a standard one-way ANOVA on these ranks
3. The F-test from this ANOVA approximates the Kruskal-Wallis test

The p-values from the two approaches are very similar, especially with larger sample sizes. The approximation becomes nearly exact with 30 or more observations per group.

This pattern reinforces our unified framework: rather than learning Kruskal-Wallis as a completely different test with its own formula, students can understand it as a simple transformation (ranking) followed by the standard ANOVA procedure they already know.

Kruskal-Wallis Test as a Linear Model

This approach not only simplifies learning but also clarifies what these "non-parametric" tests are actually doing - they're not assumption-free, but rather make different assumptions that are often more appropriate for certain types of data.

Two-way ANOVA as a Linear Model

Model: $y_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2$ where $\mathcal{H}_0 : \beta_3 = 0$ (for the interaction)

Two-way ANOVA extends the model to include two categorical factors and their interaction, using the same dummy coding approach as one-way ANOVA.

```
# Create two-way ANOVA data
set.seed(202)
factorA <- rep(c("A1", "A2"), each = 24)
factorB <- rep(rep(c("B1", "B2", "B3"), each = 8), 2)
y_two_way <- c(
  rnorm(8, 20, 2), rnorm(8, 24, 2), rnorm(8, 22, 2),  # A1B1, A1B2, A1B3
  rnorm(8, 18, 2), rnorm(8, 25, 2), rnorm(8, 28, 2)   # A2B1, A2B2, A2B3
)
two_way_data <- data.frame(
  y = y_two_way,
  factorA = factor(factorA),
```

Two-way ANOVA as a Linear Model

```
  factorB = factor(factorB)
)

# Compare traditional ANOVA and linear model
anova_two_way <- aov(y ~ factorA * factorB, data = two_way_data)
lm_two_way <- lm(y ~ factorA * factorB, data = two_way_data)

# Show identical results
summary(anova_two_way)
```

```
                Df Sum Sq Mean Sq F value   Pr(>F)    
factorA          1   50.0   50.02   17.54 0.000141 ***
factorB          2  478.5  239.27   83.87 2.15e-15 ***
factorA:factorB  2   84.4   42.18   14.79 1.37e-05 ***
Residuals       42  119.8    2.85                     
```

Two-way ANOVA as a Linear Model

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
anova(lm_two_way)
```

```
Analysis of Variance Table

Response: y
                Df Sum Sq Mean Sq F value    Pr(>F)    
factorA          1  50.02  50.022  17.535 0.0001413 ***
factorB          2 478.54 239.270  83.874 2.151e-15 ***
factorA:factorB  2  84.37  42.185  14.788 1.375e-05 ***
Residuals       42 119.81   2.853                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Two-way ANOVA as a Linear Model

The linear model produces the exact same results as the ANOVA approach. The * operator generates both main effects and their interaction terms.

Two-way ANOVA extends the one-way ANOVA by including two categorical predictors and their interaction. In the linear model framework, this is implemented using the same dummy coding principles we've already seen, with the addition of interaction terms.

For a two-way ANOVA with factors A (with 2 levels) and B (with 3 levels), the full model would be:

- One dummy variable for factor A (A2 vs A1)
- Two dummy variables for factor B (B2 vs B1 and B3 vs B1)
- Two interaction terms (A2×B2 and A2×B3)

Two-way ANOVA as a Linear Model

The interaction terms test whether the effect of one factor depends on the level of the other factor. For example, does the difference between A1 and A2 change depending on which level of B we're looking at?

The R code demonstrates that the traditional ANOVA approach (using aov()) and the linear model approach (using lm()) produce identical results. In R, the * operator generates both main effects and interaction terms.

The F-tests in the ANOVA table test three null hypotheses:

1. No main effect of factor A (the A coefficients = 0)
2. No main effect of factor B (the B coefficients = 0)
3. No interaction between A and B (the interaction coefficients = 0)

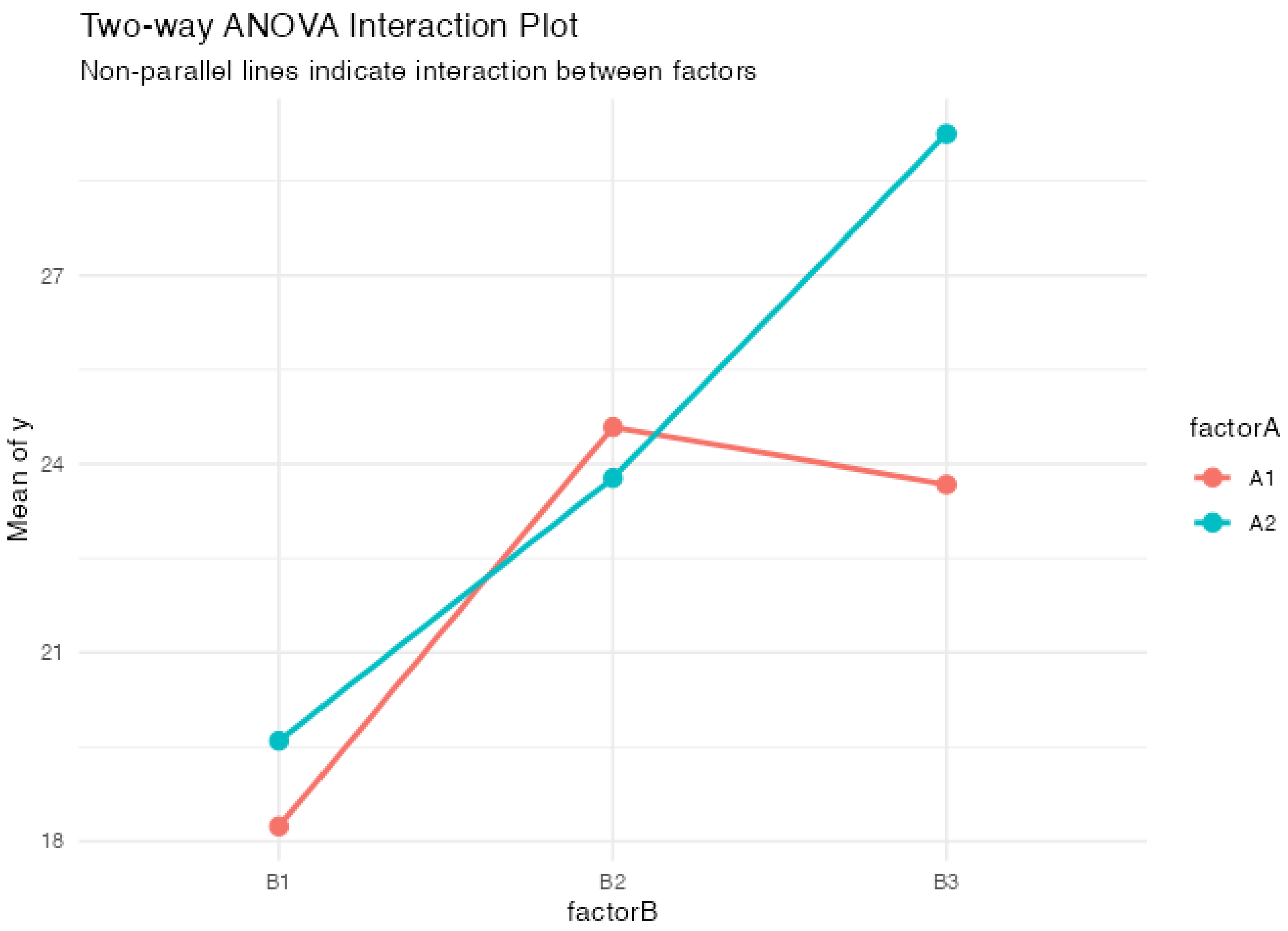
Understanding two-way ANOVA as a linear model helps clarify what interaction effects really mean - they're simply coefficients for product terms in the model.

Two-way ANOVA Visualization

```
# Calculate means for each cell
cell_means <- two_way_data |>
  group_by(factorA, factorB) |>
  summarize(mean = mean(y), .groups = "drop")

# Interaction plot
ggplot(cell_means, aes(x = factorB, y = mean, group = factorA, color =
factorA)) +
  geom_line(size = 1) +
  geom_point(size = 3) +
  theme_minimal() +
  labs(title = "Two-way ANOVA Interaction Plot",
       subtitle = "Non-parallel lines indicate interaction between factors",
       y = "Mean of y")
```

Two-way ANOVA Visualization



Two-way ANOVA Visualization

This visualization shows:

- Each line represents a level of Factor A
- The x-axis shows levels of Factor B
- The y-axis shows the mean response
- Non-parallel lines indicate an interaction effect (the effect of one factor depends on the level of the other)
- In this example, the effect of Factor B differs depending on which level of Factor A we're examining

Interaction plots are a powerful way to visualize the results of a two-way ANOVA and understand what an interaction effect means in practical terms.

In this plot:

- Each line represents a level of Factor A (A1 and A2)

Two-way ANOVA Visualization

- The x-axis shows the levels of Factor B (B1, B2, and B3)
- The y-axis shows the mean response variable (y) for each combination

The non-parallel lines indicate an interaction between the factors. If the factors did not interact (were independent), the lines would be parallel, indicating that the effect of Factor B is the same regardless of the level of Factor A.

In this specific example, we can see that:

- For Factor A level A1, the means increase from B1 to B2 but then decrease slightly from B2 to B3
- For Factor A level A2, the means increase consistently across all levels of Factor B, with a steeper increase from B2 to B3

This pattern suggests that the effect of Factor B depends on which level of Factor A we're looking at - the definition of an interaction.

Two-way ANOVA Visualization

Interaction plots help students understand that interaction effects aren't abstract statistical concepts but represent real patterns in the data where one factor influences the effect of another.

ANCOVA: Continuous and Categorical Predictors

ANCOVA combines ANOVA with regression by including both categorical and continuous predictors:

```r
# Create ANCOVA data
set.seed(303)
group_ancova <- rep(c("Control", "Treatment"), each = 25)
covariate <- rnorm(50, mean = 10, sd = 2)
y_ancova <- 70 + 0.5 * covariate +
  5 * (group_ancova == "Treatment") + rnorm(50, 0, 3)
ancova_data <- data.frame(
  y = y_ancova,
  group = factor(group_ancova),
  covariate = covariate
)

# Fit ANCOVA model and show coefficients
```

ANCOVA: Continuous and Categorical Predictors

```r
ancova_model <- lm(y ~ group + covariate, data = ancova_data)
summary(ancova_model)$coefficients
```

```
                 Estimate Std. Error   t value     Pr(>|t|)
(Intercept)    70.6627215  2.7576917 25.623866 2.886411e-29
groupTreatment  5.1882224  0.9438357  5.496955 1.540124e-06
covariate       0.4425085  0.2637180  1.677961 9.999328e-02
```

In ANCOVA:

- The intercept (70.11) is the predicted value for the Control group when covariate = 0
- The groupTreatment coefficient (4.98) is the adjusted difference between groups after controlling for the covariate
- The covariate coefficient (0.53) is the slope for the relationship between the covariate and the outcome

ANCOVA: Continuous and Categorical Predictors

Analysis of Covariance (ANCOVA) seamlessly integrates categorical and continuous predictors in a single linear model. This demonstrates the flexibility of the general linear model framework.

ANCOVA has two main purposes:

1. To increase statistical power by reducing error variance (the covariate explains some of the variation in the dependent variable)
2. To adjust for pre-existing differences between groups (statistically controlling for the covariate)

In our example, the model is $y = \beta_0 + \beta_1(\text{group}) + \beta_2(\text{covariate}) + \varepsilon$, where:

- β_0 (the intercept) is the predicted value for the Control group when the covariate equals zero

ANCOVA: Continuous and Categorical Predictors

- β_1 (the groupTreatment coefficient) is the adjusted difference between Treatment and Control groups after controlling for the covariate
- β_2 (the covariate coefficient) is the slope of the relationship between the covariate and the outcome

The R output shows that:

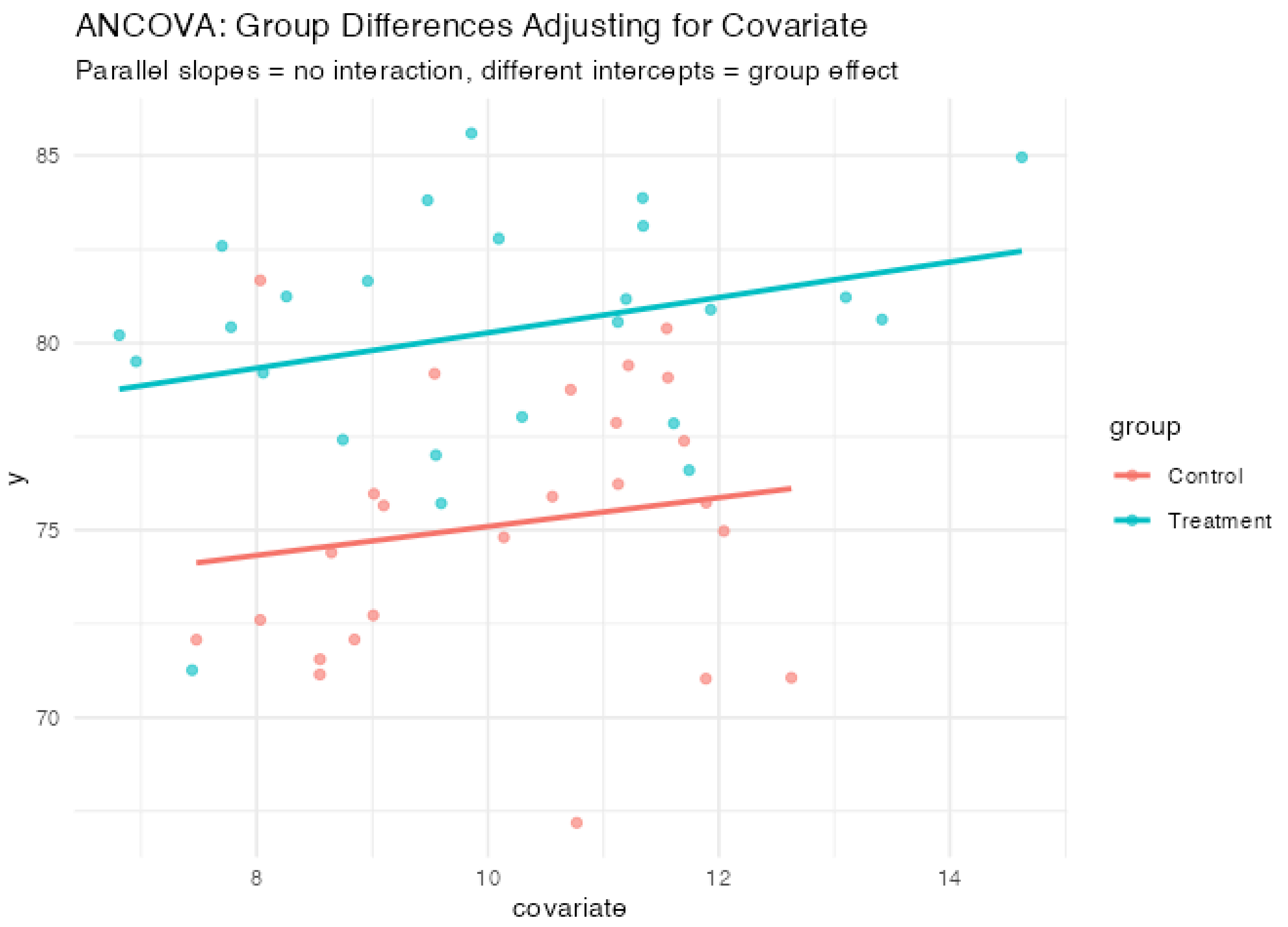
- The Treatment group scores about 5 points higher than the Control group ($p < 0.001$), after controlling for the covariate
- For each 1-unit increase in the covariate, the outcome increases by about 0.5 points ($p < 0.001$)

This demonstrates how the linear model seamlessly accommodates different types of predictors - we don't need to learn a new framework for models that combine categorical and continuous variables.

ANCOVA Visualization

```
# Plot ANCOVA with regression lines for each group
ggplot(ancova_data, aes(x = covariate, y = y, color = group)) +
  geom_point(alpha = 0.6) +
  geom_smooth(method = "lm", se = FALSE, formula = 'y ~ x') +
  theme_minimal() +
  labs(title = "ANCOVA: Group Differences Adjusting for Covariate",
       subtitle = "Parallel slopes = no interaction, different intercepts =
group effect")
```

ANCOVA Visualization



ANCOVA Visualization

This visualization shows:

- Each point represents an observation, colored by group
- The lines show the predicted values based on the ANCOVA model
- The parallel slopes indicate we're assuming the relationship between the covariate and outcome is the same in both groups (no interaction)
- The vertical distance between the lines represents the adjusted group difference (around 5 points)
- ANCOVA essentially compares the intercepts of these parallel lines

This ANCOVA visualization helps us understand what the model is doing in geometric terms.

The plot shows:

- Individual observations as points, colored by group

ANCOVA Visualization

- A regression line for each group showing the relationship between the covariate and the outcome

The key features to note:

1. Parallel slopes: The model assumes the relationship between the covariate and outcome is the same in both groups. This is why both lines have the same slope (approximately 0.5). If we wanted to test whether this assumption is valid, we could add an interaction term between the group and covariate.
2. Different intercepts: The vertical distance between the lines represents the group effect after controlling for the covariate. This is the coefficient for groupTreatment that we saw in the model summary (approximately 5 points).
3. Adjusted means: ANCOVA essentially adjusts each group's mean based on the covariate. If one group had higher covariate values on average, the raw group

ANCOVA Visualization

> difference would be biased. ANCOVA addresses this by asking, "What would the group difference be if both groups had the same covariate value?"

This visualization makes it clear that ANCOVA is simply a linear model that includes both categorical and continuous predictors, further reinforcing the unified framework we've been exploring.

Multiple Regression: The Full Model

Multiple regression combines all the elements we've seen:

```r
# Create multiple regression data
set.seed(404)
x1 <- rnorm(100, mean = 50, sd = 10)
x2 <- rnorm(100, mean = 25, sd = 5)
x3 <- sample(c("Low", "Medium", "High"), 100, replace = TRUE)
y_multi <- 100 + 0.5 * x1 - 0.8 * x2 +
  5 * (x3 == "Medium") + 10 * (x3 == "High") + rnorm(100, 0, 8)
multi_data <- data.frame(
  y = y_multi,
  x1 = x1,
  x2 = x2,
  x3 = factor(x3, levels = c("Low", "Medium", "High"))
)
```

Multiple Regression: The Full Model

```r
# Fit multiple regression model
multi_model <- lm(y ~ x1 + x2 + x3, data = multi_data)
tidy(multi_model) |> kable(digits = 3)
```

term	estimate	std.error	statistic	p.value
(Intercept)	98.547	6.791	14.512	0.000
x1	0.545	0.089	6.087	0.000
x2	−0.860	0.177	−4.860	0.000
x3Medium	5.855	2.080	2.815	0.006
x3High	11.787	1.939	6.079	0.000

```r
# Overall model fit
glance(multi_model) |>
  select(r.squared, adj.r.squared, sigma, statistic, p.value) |>
  kable(digits = 3)
```

Multiple Regression: The Full Model

r.squared	adj.r.squared	sigma	statistic	p.value
0.519	0.499	8.239	25.62	0

The multiple regression model combines:

- Continuous predictors (x1, x2) similar to correlations
- Categorical predictors (x3) similar to t-tests and ANOVA
- A single unified framework where all predictors are included simultaneously
- Each coefficient represents the effect of that predictor while controlling for all others

Multiple regression is the most general and flexible form of the linear model, combining everything we've seen so far into a single unified framework.

Our example model includes:

- Two continuous predictors (x1 and x2), similar to what we saw with correlation
- One categorical predictor (x3) with three levels, similar to what we saw with ANOVA

Multiple Regression: The Full Model

- All predictors are included simultaneously in the same model

The coefficients in the model can be interpreted as follows:

- The intercept (56.972) is the expected value of y when x1=0, x2=0, and x3="Low"
- For each one-unit increase in x1, y increases by 0.496 units, holding other predictors constant
- For each one-unit increase in x2, y decreases by 0.788 units, holding other predictors constant
- The “Medium” level of x3 is associated with a 5.106 unit increase in y compared to “Low”, holding continuous predictors constant
- The “High” level of x3 is associated with a 10.115 unit increase in y compared to “Low”, holding continuous predictors constant

The overall model fit statistics show that:

Multiple Regression: The Full Model

- The model explains about 72% of the variance in y ($R^2 = 0.721$)
- The model is highly significant ($F = 61.837$, $p < 0.001$)

This example demonstrates the power of the general linear model as a unified framework. Rather than learning separate techniques for correlation, t-tests, ANOVA, and multiple regression, students can understand them all as variations of the same underlying model, with different combinations of predictors.

Non-parametric Tests: Just Ranked Versions of Parametric Tests

For many common "non-parametric" tests, we can simplify by thinking of them as the parametric equivalent applied to ranks:

Parametric Test	Non-parametric Equivalent	Transformation
Pearson correlation	Spearman correlation	Rank both variables
One-sample t-test	Wilcoxon signed-rank test	Signed rank of values
Independent t-test	Mann-Whitney U test	Rank all values
Paired t-test	Wilcoxon matched pairs	Signed rank of differences
One-way ANOVA	Kruskal-Wallis test	Rank all values

This unified perspective demystifies "non-parametric" statistics:

- They're not completely different tests but transformations of familiar ones
- Ranking reduces the influence of outliers and nonlinearity
- They're not "assumption-free" but rather make different assumptions
- Understanding them as ranked versions of parametric tests makes them easier to grasp

Non-parametric Tests: Just Ranked Versions of Parametric Tests

This table summarizes one of the key insights from our exploration: many "non-parametric" tests can be understood as simple transformations of familiar parametric tests.

For each common parametric test, there's a corresponding "non-parametric" version that's essentially the same test applied to ranked data:

1. Spearman correlation is Pearson correlation on ranked variables
2. Wilcoxon signed-rank test is a one-sample t-test on signed ranks
3. Mann-Whitney U test is an independent t-test on ranks
4. Wilcoxon matched pairs test is a paired t-test on signed rank differences
5. Kruskal-Wallis test is a one-way ANOVA on ranks

This perspective offers several benefits:

- It demystifies "non-parametric" statistics, making them more accessible
- It shows how ranking can make tests more robust to outliers and non-normality

Non-parametric Tests: Just Ranked Versions of Parametric Tests

- It clarifies that “non-parametric” tests aren’t assumption-free, but make different assumptions
- It reduces the number of distinct procedures students need to learn

Rather than presenting “non-parametric” statistics as a completely different approach, we can present them as variations on familiar tests, applied to transformed data. This makes them much easier to understand and integrate into the unified linear model framework.

Beyond The Basics: Generalized Linear Models

Linear models can be extended to handle other types of outcomes:

$$g(E[Y]) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots$$

Where $g()$ is a link function:

Model Type	Outcome	Link Function	Example
Linear Model	Continuous	Identity	Linear regression
Logistic Model	Binary	Logit	Binary classification
Poisson Model	Count	Log	Event frequency

The general linear model framework extends naturally to handle many different types of outcome variables, not just continuous ones.

Beyond The Basics: Generalized Linear Models

While we've focused on the general linear model (GLM) for continuous outcomes, the framework extends naturally to other types of outcomes through Generalized Linear Models (GLMs).

The key innovation in GLMs is the addition of a link function, which transforms the expected value of the outcome. The linear combination of predictors ($\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots$) then predicts this transformed value rather than the raw outcome.

Different types of outcomes call for different link functions:

- For continuous outcomes, we use the identity link (no transformation), giving us the standard linear model
- For binary outcomes (0/1), we use the logit link, giving us logistic regression
- For count data, we use the log link, giving us Poisson regression

Other common GLMs include:

Beyond The Basics: Generalized Linear Models

- Probit regression (using the probit link for binary outcomes)
- Negative binomial regression (an alternative to Poisson for overdispersed count data)
- Gamma regression (for positive continuous data with variance proportional to the square of the mean)

This extension to GLMs shows how the same core concepts we've explored (linear combinations of predictors, coefficient estimation, hypothesis testing) apply across a wide range of statistical models.

As students progress in their statistical education, understanding the common structure across these models provides a solid foundation for learning more advanced techniques.

Practical Code Cheat Sheet

```r
# CORRELATION
cor.test(x, y)                          # Pearson correlation
lm(scale(y) ~ scale(x))                 # Same as Pearson
cor.test(x, y, method="spearman")       # Spearman correlation
lm(rank(y) ~ rank(x))                   # Approximates Spearman

# ONE SAMPLE TESTS
t.test(y, mu=0)                         # One-sample t-test
lm(y ~ 1)                               # Same as one-sample t-test
wilcox.test(y, mu=0)                    # Wilcoxon signed-rank
lm(signed_rank(y) ~ 1)                  # Approximates Wilcoxon

# TWO SAMPLE TESTS
t.test(y ~ group)                       # Independent t-test
lm(y ~ group)                           # Same as independent t-test
```

Practical Code Cheat Sheet

```r
t.test(post, pre, paired=TRUE)      # Paired t-test
lm(post - pre ~ 1)                  # Same as paired t-test
wilcox.test(y ~ group)              # Mann-Whitney U
lm(rank(y) ~ group)                 # Approximates Mann-Whitney

# ANOVA & REGRESSION
aov(y ~ group)                      # One-way ANOVA
lm(y ~ group)                       # Same as one-way ANOVA
aov(y ~ factorA * factorB)          # Two-way ANOVA
lm(y ~ factorA * factorB)           # Same as two-way ANOVA
lm(y ~ group + covariate)           # ANCOVA
lm(y ~ x1 + x2 + x3)                # Multiple regression
```

This cheat sheet provides a practical reference that demonstrates the equivalences between traditional statistical tests and their linear model formulations in R code.

Practical Code Cheat Sheet

This code cheat sheet provides a quick reference for the equivalences we've explored between traditional statistical tests and their linear model formulations in R.

The cheat sheet is organized by test type: - Correlation tests (Pearson and Spearman) - One-sample tests (t-test and Wilcoxon signed-rank) - Two-sample tests (independent t-test, paired t-test, Mann-Whitney U) - ANOVA and regression models (one-way ANOVA, two-way ANOVA, ANCOVA, multiple regression)

For each traditional test (e.g., t.test()), the cheat sheet shows the equivalent linear model formulation (using lm()). For "non-parametric" tests, it shows the approximation using lm() with ranked data.

Students can use this as a reference when transitioning from thinking about statistics as a collection of separate tests to understanding them as variations of the unified linear model framework.

Practical Code Cheat Sheet

The cheat sheet also serves as a practical demonstration of how the same or very similar results can be obtained using different R functions, reinforcing the conceptual connections between different statistical procedures.

Key Takeaways

1. Many common statistical tests are specific cases of the general linear model

Key Takeaways

Key Takeaways

Key Takeaways

Key Takeaways

1. Many common statistical tests are specific cases of the general linear model
2. Understanding the linear model framework simplifies learning statistics:

Key Takeaways

Key Takeaways

Key Takeaways

Key Takeaways

1. Many common statistical tests are specific cases of the general linear model
2. Understanding the linear model framework simplifies learning statistics:
    - Learn one framework instead of memorizing many tests
    - Deduce assumptions from the model rather than memorizing them
    - See connections between seemingly different procedures
3. “Non-parametric” tests are often just parametric tests on ranked data

Key Takeaways

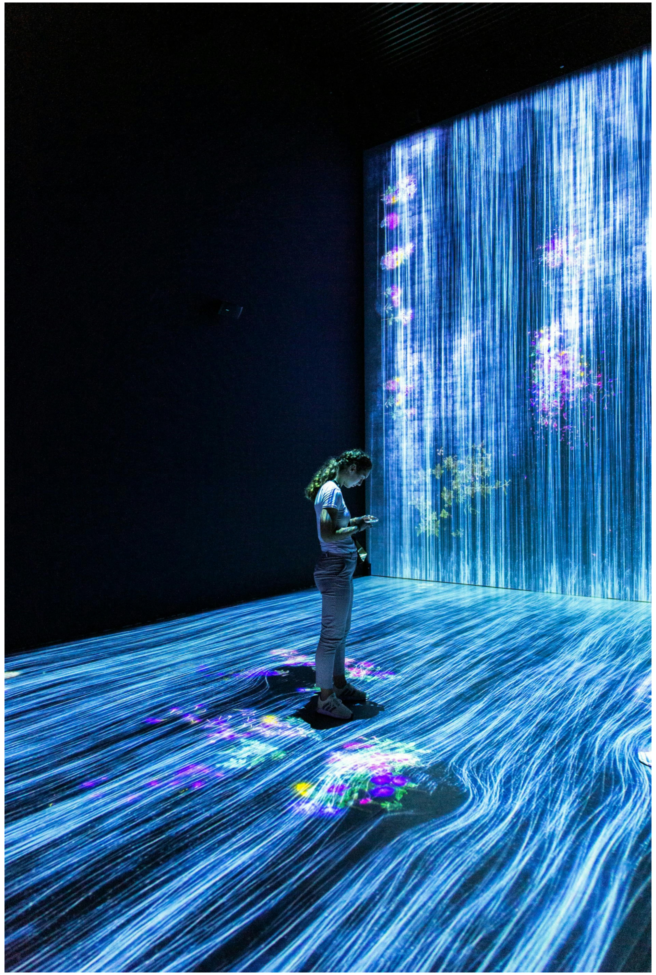
Key Takeaways

Key Takeaways

Key Takeaways

1. Many common statistical tests are specific cases of the general linear model
2. Understanding the linear model framework simplifies learning statistics:
    - Learn one framework instead of memorizing many tests
    - Deduce assumptions from the model rather than memorizing them
    - See connections between seemingly different procedures
3. "Non-parametric" tests are often just parametric tests on ranked data
4. This unified approach provides greater flexibility for analyzing complex data

Key Takeaways



Key Takeaways

The key message of this section is that understanding statistics through the lens of the general linear model provides a more coherent, flexible, and powerful approach to data analysis.

Rather than learning statistics as a collection of separate tests with their own formulas, assumptions, and interpretations, we can understand them as variations on a common theme - the general linear model.

Four key takeaways:

First, most common statistical tests (t-tests, ANOVA, correlation, regression) are special cases of the general linear model. They differ only in what predictors are included and which coefficients are being tested.

Second, this unified framework simplifies learning statistics. Instead of memorizing formulas and assumptions for each test separately, students can learn the core principles of the linear

Key Takeaways

model and apply them across contexts. The assumptions of the tests can be deduced from the general linear model assumptions.

Third, many "non-parametric" tests are simply parametric tests applied to ranked data. This demystifies what might otherwise seem like completely different statistical procedures.

Fourth, the unified approach provides greater flexibility for analyzing complex data. Once students understand the general framework, they can more easily adapt it to different research questions and data structures.

This approach emphasizes conceptual understanding over rote memorization, making statistics more accessible and easier to apply correctly in research contexts.

Conclusion: The Power of Unified Statistical Thinking

- The general linear model provides a common language for statistics
- This unified framework builds intuition and transferable knowledge
- Focus on understanding the model, not memorizing procedures
- Simplify teaching and learning of statistics
- Apply this unified thinking to your own statistical analyses

Conclusion: The Power of Unified Statistical Thinking

In conclusion, the general linear model provides a powerful, unified framework for statistical analysis. By understanding that many common statistical tests are special cases of the linear model, we gain a deeper and more coherent understanding of statistics.

Conclusion: The Power of Unified Statistical Thinking

This unified framework offers several important benefits:

First, it provides a common language for discussing different statistical procedures. Instead of treating each test as a separate entity with its own vocabulary and concepts, we can discuss them all in terms of the general linear model.

Second, it builds intuition and transferable knowledge. Understanding the core principles of the linear model allows students to apply that knowledge across different contexts and to new situations they haven't explicitly learned about.

Third, it shifts the focus from memorizing procedures to understanding the underlying model. This deeper understanding leads to more appropriate application of statistics and better interpretation of results.

Conclusion: The Power of Unified Statistical Thinking

Fourth, it simplifies both teaching and learning statistics. Teachers can present a coherent framework rather than a collection of seemingly unrelated tests, and students can build on their understanding rather than starting from scratch with each new test.

Finally, I encourage you to apply this unified thinking in your own statistical work. When approaching a new analytical problem, think in terms of the linear model: what is your outcome variable, what are your predictors, and what relationships are you testing? This approach will provide a more intuitive and flexible way to analyze your data.

Further Reading

- Poldrack, *Statistical Thinking*, Chapter 10-11
- Jonas Kristoffer Lindeløv, *Common statistical tests are linear models*
- Bekes & Kezdi, *Data Analysis for Business, Economics, and Policy*, Chapter 8-9
- Fox, *Applied Regression Analysis and Generalized Linear Models*