BZAN 535: Comprehensive Exam Concepts Guide with R Code Examples

==

TABLE OF CONTENTS

==

==

PART 1: LINEAR REGRESSION CONCEPTS

==

--- 1.1 BASIC LINEAR REGRESSION WITH SIGNIFICANCE TESTING ---

CONCEPT:
Linear regression models the relationship between a dependent variable (Y)
and independent variables (X):

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \varepsilon$$

Where:
- β_0 = intercept (Y when all X = 0)
- β_j = slope (effect of X_j on Y)
- ε = error term

SIGNIFICANCE TESTING:
For each coefficient, we test:
H_0: $\beta_j = 0$ (no effect)
H_1: $\beta_j \neq 0$ (has an effect)

Using t-statistic: t = coefficient / standard_error
And p-value: P(|t| > |t_observed|)

Decision: If p-value < 0.05, reject H_0 (significant at 5% level)

WHEN TO USE:
- Predicting continuous outcomes
- Testing if variables significantly affect outcome
- Estimating magnitude of relationships
- Hypothesis testing

R CODE EXAMPLE:

```
# Load data
data(mtcars)
head(mtcars)

# Fit basic regression model
model1 <- lm(mpg ~ wt + hp, data = mtcars)

# View results
summary(model1)

# OUTPUT EXPLANATION:
# Call: lm(formula = mpg ~ wt + hp, data = mtcars)
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept) 37.22727    1.59879  23.285  < 2e-16 ***
# wt          -3.87783    0.63273  -6.129  1.12e-06 ***
# hp          -0.03177    0.00903  -3.519  0.00145 **
#
# R-squared: 0.8268
# Adjusted R-squared: 0.8148
# F-statistic: 69.21 on 2 and 29 DF, p-value: 9.109e-12
```

INTERPRETATION:

1. REGRESSION EQUATION:
   mpg = 37.23 - 3.88*wt - 0.032*hp

This means:
- Baseline (at wt=0, hp=0): 37.23 mpg
- Each 1000 lb weight increase: -3.88 mpg
- Each 1 hp increase: -0.032 mpg

2. SIGNIFICANCE TEST FOR WEIGHT:
- Coefficient: -3.87783
- Std Error: 0.63273
- t-statistic: -6.129
- p-value: 1.12e-06 (very small!)

Conclusion: Weight is HIGHLY SIGNIFICANT ($p < 0.001$)
Weight definitely affects mpg

3. SIGNIFICANCE TEST FOR HP:
- Coefficient: -0.03177
- Std Error: 0.00903
- t-statistic: -3.519
- p-value: 0.00145

Conclusion: HP is SIGNIFICANT ($p < 0.01$)
Horsepower significantly affects mpg

EXTRACT AND TEST COEFFICIENTS IN R:

```
# Get coefficients
coefs <- coef(model1)
print(coefs)

# Get standard errors
se <- sqrt(diag(vcov(model1)))
print(se)

# Calculate t-statistics
t_stats <- coefs / se
print(t_stats)

# Calculate p-values (two-tailed test)
n <- nrow(mtcars)
p_vals <- 2 * (1 - pt(abs(t_stats), df = n - 3))
print(p_vals)
```

```
# Create results table
results <- data.frame(
  Coefficient = coefs,
  Std_Error = se,
  t_stat = t_stats,
  p_value = p_vals,
  Significant_at_5pct = p_vals < 0.05
)
print(results)
```

==========

--- 1.2 MODEL FIT: R^2 INTERPRETATION ---

CONCEPT:
R^2 (R-squared) measures what proportion of variance in Y is explained by X:

R^2 = SS_explained / SS_total = 1 - (SS_residual / SS_total)

Range: 0 to 1 (or 0% to 100%)

INTERPRETATION GUIDELINES:
- R^2 = 0.9+ : Excellent fit (explains 90%+ of variation)
- R^2 = 0.7-0.9: Good fit (explains 70-90%)
- R^2 = 0.5-0.7: Moderate fit (explains 50-70%)
- R^2 = 0.3-0.5: Weak fit (explains 30-50%)
- R^2 < 0.3 : Poor fit (explains <30%)

ADJUSTED R^2:
When adding variables, R^2 always increases (even if variables are useless).
Adjusted R^2 penalizes for adding variables:

Adjusted R^2 = 1 - [(1-R^2)(n-1)/(n-p-1)]

Use this to compare models with different numbers of variables.

WHEN TO USE:
- Assessing overall model quality
- Comparing different models
- Communicating fit to stakeholders

R CODE EXAMPLE:

```
# Compare different model specifications
model_simple <- lm(mpg ~ wt, data = mtcars)
model_medium <- lm(mpg ~ wt + hp, data = mtcars)
model_complex <- lm(mpg ~ wt + hp + cyl + disp + drat, data = mtcars)

# Extract R² and Adjusted R²
get_fit_metrics <- function(mod) {
  r2 <- summary(mod)$r.squared
  adj_r2 <- summary(mod)$adj.r.squared
  n_vars <- length(coef(mod)) - 1
  return(data.frame(
    R_squared = round(r2, 4),
    Adj_R_squared = round(adj_r2, 4),
    Num_Variables = n_vars,
    Pct_Explained = paste0(round(r2*100, 1), "%")
  ))
}

# Create comparison table
comparison <- rbind(
  Simple = get_fit_metrics(model_simple),
  Medium = get_fit_metrics(model_medium),
  Complex = get_fit_metrics(model_complex)
)
print(comparison)

# OUTPUT:
#         R_squared Adj_R_squared Num_Variables Pct_Explained
# Simple   0.7528        0.7446             1         75.3%
# Medium   0.8268        0.8148             2         82.7%
# Complex  0.8486        0.8161             5         84.9%
```

INTERPRETATION:

1. Simple model (wt only):
   $R^2 = 0.7528 \rightarrow$ explains 75.3% of mpg variation
   Good fit with just one variable!

2. Medium model (wt + hp):
   $R^2 = 0.8268 \rightarrow$ explains 82.7% of mpg variation
   Adding hp improved fit by 7.4 percentage points

3. Complex model (wt + hp + cyl + disp + drat):
   $R^2 = 0.8486 \rightarrow$ explains 84.9% of mpg variation
   Adding 3 more variables improved fit by only 2.1 points

   BUT: Adjusted R^2 actually DECREASED from 0.8148 to 0.8161
   This suggests the extra variables aren't worth the complexity!

==========

--- 1.3 QUADRATIC (NONLINEAR) TERMS ---

CONCEPT:
Linear relationships assume the effect of X is constant at all levels.
Quadratic terms allow the effect to change with X:

  Model 1 (Linear): $Y = \beta_0 + \beta_1 X + \varepsilon$
  Model 2 (Quadratic): $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \varepsilon$

If β_2 is negative: Inverted-U shape (concave)
  $\rightarrow$ Y increases with X initially, then decreases
  $\rightarrow$ Peak at $X^* = -\beta_1/(2\beta_2)$

If β_2 is positive: U-shape (convex)
  $\rightarrow$ Y decreases with X initially, then increases
  $\rightarrow$ Minimum at $X^* = -\beta_1/(2\beta_2)$

WHEN TO USE:
- Diminishing returns (e.g., study hours vs test score)
- Inverted-U relationships (e.g., age vs productivity)
- Non-monotonic effects (goes up then down)
- When visual inspection suggests curves

R CODE EXAMPLE:

```
# Create example data with quadratic relationship
set.seed(123)
x <- seq(1, 10, length.out = 100)
y <- 50 + 10*x - 0.8*(x^2) + rnorm(100, sd = 5)

# Fit linear model
```

```
linear_model <- lm(y ~ x)

# Fit quadratic model
quadratic_model <- lm(y ~ x + I(x^2))

# Compare models
summary(linear_model)
summary(quadratic_model)

# Extract quadratic coefficient
coef_quad <- coef(quadratic_model)["I(x^2)"]
coef_lin <- coef(quadratic_model)["x"]

# If β₂ < 0: inverted-U (find peak)
if (coef_quad < 0) {
  peak_x <- -coef_lin / (2 * coef_quad)
  cat("Peak occurs at X =", round(peak_x, 2), "\n")
}
```

TEST FOR SIGNIFICANCE OF QUADRATIC TERM:

```
# Is the quadratic term significant?
summary(quadratic_model)

# OUTPUT shows Pr(>|t|) for I(x^2)
# If p < 0.05: Quadratic term is significant!
```

INTERPRETATION:

If quadratic term significant:
- Simple linear model too restrictive
- Effect of X changes across X values
- Must report marginal effects at different X levels

If quadratic term NOT significant:
- Linear model is sufficient
- Effect of X is constant

==========

--- 1.4 INTERACTION EFFECTS ---

CONCEPT:

Interaction means the effect of one variable depends on another:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 (X_1 \times X_2) + \varepsilon$$

β_1 = effect of X_1 when $X_2 = 0$

β_3 = how the effect of X_1 changes when X_2 increases by 1

Example: Effect of training on earnings might differ by gender

- For males: Effect = β_1
- For females: Effect = $\beta_1 + \beta_3$

WHEN TO USE:

- When you expect group differences in relationships
- Testing if effects are heterogeneous (different for different groups)
- Examining policy effects by demographic
- Testing for differential treatment effects

R CODE EXAMPLE:

```
# Create example data
set.seed(123)
n <- 200
df <- data.frame(
  income = rnorm(n, 50000, 20000),
  female = rbinom(n, 1, 0.5),
  spending = NA
)

# Generate spending with interaction
# Effect of income is DIFFERENT for males vs females
df$spending <- 500 +
            0.008*df$income +          # Main effect of income
            -200*df$female +           # Main effect of female
            0.005*df$income*df$female +  # INTERACTION
            rnorm(n, 0, 1000)          # Random noise

# Fit model WITH interaction
model_interact <- lm(spending ~ income + female + income:female, data = df)

summary(model_interact)
```

EXTRACTING AND INTERPRETING COEFFICIENTS:

```
coefs <- coef(model_interact)

cat("=== REGRESSION EQUATION ===\n")
cat(sprintf("Spending = %.2f + %.6f*Income + %.2f*Female + %.8f*Income×Female\n\n",
            coefs[1], coefs[2], coefs[3], coefs[4]))

cat("=== EFFECT OF INCOME BY GENDER ===\n")

income_for_males <- coefs[2]
income_for_females <- coefs[2] + coefs[4]

cat(sprintf("For MALES (Female=0):\n")
cat(sprintf("  Effect of income = %.6f\n", income_for_males))
cat(sprintf("  → Each $1,000 increase in income → $%.2f more spending\n\n",
            income_for_males*1000))

cat(sprintf("For FEMALES (Female=1):\n")
cat(sprintf("  Effect of income = %.6f\n", income_for_females))
cat(sprintf("  → Each $1,000 increase in income → $%.2f more spending\n\n",
            income_for_females*1000))

cat(sprintf("DIFFERENCE:\n")
cat(sprintf("  Females respond $%.2f MORE per $1,000 income increase\n",
            (income_for_females - income_for_males)*1000))
```

KEY PRINCIPLE:

When you have an interaction $X_1 \times X_2$:
- NEVER report β_1 alone (it only applies when $X_2=0$)
- ALWAYS report effects CONDITIONALLY on X_2
- Say: "For $X_2=0$, effect is...; for $X_2=1$, effect is..."

==========

--- 1.5 LOGISTIC REGRESSION FOR BINARY OUTCOMES ---

CONCEPT:
When Y is binary (0/1), use logistic regression:

$P(Y=1|X) = 1 / (1 + e^{(-[\beta_0 + \beta_1 X_1 + ... + \beta_p X_p])})$

This is called the "logistic function" or "sigmoid"
- Output bounded between 0 and 1 (valid probabilities)
- Nonlinear relationship between X and probability

Use glm() function with family=binomial(link="logit")

WHEN TO USE:
- Predicting binary outcomes (purchase/no purchase, approve/deny)
- Estimating probability of events
- Classification problems

R CODE EXAMPLE:

```
# Create binary outcome data
set.seed(123)
n <- 200
df <- data.frame(
  age = rnorm(n, 45, 15),
  income = rnorm(n, 75000, 25000),
  female = rbinom(n, 1, 0.5),
  purchase = NA
)

# Generate binary outcome based on logistic model
df$purchase <- rbinom(n, 1,
                      plogis(-2 + 0.03*df$age + 0.00002*df$income - 0.5*df$female))

# Fit logistic regression
logit_model <- glm(purchase ~ age + income + female,
                   family = binomial(link = "logit"),
                   data = df)

summary(logit_model)
```

INTERPRETING COEFFICIENTS:

```
coefs <- coef(logit_model)
```

```
cat("=== LOGIT COEFFICIENT INTERPRETATION ===\n\n")

cat("Age coefficient:", round(coefs[2], 6), "\n")
if (coefs[2] > 0) {
  cat("  → POSITIVE: Older people MORE likely to purchase\n\n")
} else {
  cat("  → NEGATIVE: Older people LESS likely to purchase\n\n")
}

cat("Income coefficient:", round(coefs[3], 8), "\n")
if (coefs[3] > 0) {
  cat("  → POSITIVE: Higher income MORE likely to purchase\n\n")
} else {
  cat("  → NEGATIVE: Higher income LESS likely to purchase\n\n")
}

cat("Female coefficient:", round(coefs[4], 6), "\n")
if (coefs[4] > 0) {
  cat("  → POSITIVE: Females MORE likely to purchase than males\n\n")
} else {
  cat("  → NEGATIVE: Females LESS likely to purchase than males\n\n")
}
```

==========

--- 1.6 ODDS RATIOS AND PROBABILITY INTERPRETATION ---

CONCEPT:
In logistic regression, interpret coefficients as odds ratios:

  Odds Ratio = exp(coefficient)

Interpretation:
  - OR = 1: No effect (multiplies odds by 1)
  - OR = 1.5: Increase in predictor multiplies odds by 1.5 (50% increase)
  - OR = 0.7: Increase multiplies odds by 0.7 (30% decrease)

IMPORTANT: Odds ratio is constant, but probability change varies!

R CODE EXAMPLE:

```
# Convert logit coefficients to odds ratios
```

```
or <- exp(coef(logit_model))
print(or)

# Create interpretation
female_or <- or[4]

cat("=== ODDS RATIO INTERPRETATION ===\n\n")
cat(sprintf("Female Odds Ratio: %.4f\n", female_or))

if (female_or > 1) {
  pct_increase <- (female_or - 1) * 100
  cat(sprintf("Being female multiplies the odds of purchase by %.4f\n", female_or))
  cat(sprintf("This is equivalent to a %.1f%% INCREASE in odds\n", pct_increase))
} else {
  pct_decrease <- (1 - female_or) * 100
  cat(sprintf("Being female multiplies the odds of purchase by %.4f\n", female_or))
  cat(sprintf("This is equivalent to a %.1f%% DECREASE in odds\n", pct_decrease))
}
```

PREDICTED PROBABILITIES:

```
# Create two customer profiles
customer_A <- data.frame(age = 30, income = 40000, female = 0)
customer_B <- data.frame(age = 60, income = 100000, female = 1)

# Predict probability
pred_A <- predict(logit_model, newdata = customer_A, type = "response")
pred_B <- predict(logit_model, newdata = customer_B, type = "response")

cat("\n=== PREDICTED PROBABILITIES ===\n\n")
cat(sprintf("Customer A (Age 30, Income $40k, Male):\n")
cat(sprintf("  Predicted probability = %.4f (%.2f%%)\n\n", pred_A, pred_A*100))

cat(sprintf("Customer B (Age 60, Income $100k, Female):\n")
cat(sprintf("  Predicted probability = %.4f (%.2f%%)\n\n", pred_B, pred_B*100))

cat(sprintf("Difference: %.4f (%.2f percentage points)\n",
            pred_B - pred_A, (pred_B - pred_A)*100))
```

KEY INSIGHT:

Same odds ratio can produce different probability changes!

Example with OR=1.5:
- If baseline prob = 0.1: New prob ≈ 0.13 → ΔProb = 0.03 (3 ppts)
- If baseline prob = 0.5: New prob ≈ 0.60 → ΔProb = 0.10 (10 ppts)
- If baseline prob = 0.9: New prob ≈ 0.93 → ΔProb = 0.03 (3 ppts)

WHY? The logistic curve is S-shaped:
- Flat at extremes (0, 1)
- Steep in middle (0.5)
→ Probability changes most in the middle!

==========

PART 2: CAUSAL INFERENCE CONCEPTS

==========

--- 2.1 BIAS FROM OMITTING CONTROLS (SELECTION BIAS) ---

CONCEPT:
When treatment (D) is not randomly assigned, comparing outcomes between treated and untreated groups gives BIASED estimates.

The bias comes from:
1. People who self-select into treatment differ from those who don't
2. These differences affect outcomes independent of treatment
3. We can't separate treatment effect from selection advantage

EXAMPLE:
Question: Does job training increase earnings?

Simple comparison:
Earnings[trained] - Earnings[untrained] = $5,000

Is this the training effect? NO!

Why? Ambitious people are more likely to:
- Choose training
- Earn more anyway (even without training)

So the $5,000 includes:
- True training effect: $2,000
- Selection advantage (ambition): $3,000

WHEN TO USE:
- Understanding bias
- Justifying why we need controls
- Recognizing when causal claims are invalid

R CODE EXAMPLE:

```
# Generate data with selection bias
set.seed(123)
n <- 500

# Ability affects both training choice and earnings
ability <- rnorm(n, 0, 1)

# Training choice depends on ability (selection!)
training <- rbinom(n, 1, plogis(ability))

# Earnings depend on both ability AND training
earnings <- 30000 + 3000*training + 10000*ability + rnorm(n, 0, 2000)

df <- data.frame(earnings, training, ability)

# NAIVE COMPARISON (Biased!)
naive_estimate <- mean(earnings[training == 1]) -
                  mean(earnings[training == 0])

cat("=== NAIVE COMPARISON (BIASED) ===\n")
cat(sprintf("Average earnings trained: $%.0f\n", mean(earnings[training == 1])))
cat(sprintf("Average earnings untrained: $%.0f\n", mean(earnings[training == 0])))
cat(sprintf("Naive estimate of training effect: $%.0f\n\n", naive_estimate))
cat("⚠ This is BIASED! Includes selection advantage.\n\n")

# SIMPLE OLS (Same bias!)
naive_model <- lm(earnings ~ training, data = df)
cat("Simple OLS coefficient on training:",
    round(coef(naive_model)[2], 0), "\n")
cat("⚠ Still biased because doesn't control for ability\n\n")

# WITH CONTROL FOR ABILITY (Unbiased!)
controlled_model <- lm(earnings ~ training + ability, data = df)
cat("OLS with ability control coefficient on training:",
    round(coef(controlled_model)[2], 0), "\n")
cat("✓ This is approximately unbiased!\n")
```

```
cat("✓ Much closer to true effect of $3,000\n")
```

==========

--- 2.2 ADDING CONTROLS TO REDUCE CONFOUNDING ---

CONCEPT:
Controls remove confounding bias by holding other variables constant.

Model: $Y = \beta_0 + \tau D + \beta_1 X_1 + ... + \beta_p X_p + \varepsilon$

This compares treated vs untreated WITHIN each level of X:
- For $X_1 = x$, compare outcomes of treated vs untreated
- For $X_1 = x'$, compare outcomes of treated vs untreated
- Average these comparisons across X values

This removes selection bias IF all confounders are in X.

WHEN TO USE:
- Observational data with selection bias
- Estimating treatment effects when randomization unavailable
- Removing confounding from measured variables

R CODE EXAMPLE:

```
# Compare simple vs controlled estimates
# Using the data from Section 2.1

cat("=== MODEL COMPARISON ===\n\n")

# Model 1: No controls (BIASED)
model1 <- lm(earnings ~ training, data = df)
estimate1 <- coef(model1)[2]

# Model 2: With ability control (UNBIASED)
model2 <- lm(earnings ~ training + ability, data = df)
estimate2 <- coef(model2)[2]

comparison <- data.frame(
  Model = c("No Controls", "With Ability Control"),
  Estimate = c(estimate1, estimate2),
```

```
  Bias = c(estimate1 - 3000, estimate2 - 3000),
  True_Effect = 3000
)
print(comparison)

cat("\nInterpretation:\n")
cat("- No controls:", round(estimate1, 0),
    "← Includes selection bias\n")
cat("- With ability:", round(estimate2, 0),
    "← Much closer to truth!\n")


HOW ADDING CONTROLS CHANGES IDENTIFYING VARIATION:

cat("\n=== HOW CONTROLS CHANGE THE COMPARISON ===\n\n")

cat("WITHOUT CONTROLS:\n")
cat("  Compare: All trained people vs all untrained people\n")
cat("  Problem: Groups differ in ability (confounded)\n")
cat("  Trained group: Average ability =",
    round(mean(ability[training==1]), 2), "\n")
cat("  Untrained group: Average ability =",
    round(mean(ability[training==0]), 2), "\n")
cat("  → Selection bias!\n\n")

cat("WITH CONTROLS:\n")
cat("  Compare: Trained vs untreated WITH SAME ABILITY\n")
cat("  → Look at both groups at ability = -1\n")
cat("  → Look at both groups at ability = 0\n")
cat("  → Look at both groups at ability = 1\n")
cat("  → Average these comparisons\n")
cat("  → Result: Remove ability confounding!\n")


================================================================================

--- 2.3 NONLINEAR EFFECTS AND MARGINAL EFFECTS ---

CONCEPT:
Marginal effect = how much Y changes when X increases by 1 unit

For linear model:    ∂Y/∂X = β (constant)

For nonlinear model: ∂Y/∂X = β₁ + 2·β₂·X (depends on X!)
```

Quadratic example:

$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \varepsilon$

$\partial Y/\partial X = \beta_1 + 2 \cdot \beta_2 \cdot X$

The marginal effect DEPENDS on X level.

WHEN TO USE:

- Nonlinear relationships
- Communicating effects at specific X values
- Economics: diminishing returns, productivity curves

R CODE EXAMPLE:

```
# Create data with nonlinear relationship
set.seed(123)
x <- seq(1, 50, length.out = 100)
y <- 50 + 2*x - 0.02*(x^2) + rnorm(100, sd = 5)

df <- data.frame(x, y)

# Fit quadratic model
nlmodel <- lm(y ~ x + I(x^2), data = df)
coefs <- coef(nlmodel)

cat("=== NONLINEAR MODEL ===\n")
cat(sprintf("Y = %.2f + %.4f·X + %.6f·X²\n\n",
            coefs[1], coefs[2], coefs[3]))

# Marginal effect at different X values
cat("=== MARGINAL EFFECTS AT DIFFERENT X LEVELS ===\n\n")
cat("Marginal Effect Formula: ∂Y/∂X = ",
    round(coefs[2], 4), " + 2·",
    round(coefs[3], 6), "·X\n\n")

x_levels <- c(10, 25, 40)
for (x_val in x_levels) {
  me <- coefs[2] + 2*coefs[3]*x_val
  cat(sprintf("At X = %d: Marginal effect = %.6f\n", x_val, me))
}

cat("\nInterpretation:\n")
```

```
cat("- At low X: Marginal effect is POSITIVE and LARGE\n")
cat("- At mid X: Marginal effect gets SMALLER\n")
cat("- At high X: Marginal effect becomes NEGATIVE\n")
cat("⟹ Inverted-U relationship: Y peaks at some X level\n")

# Calculate peak
peak_x <- -coefs[2] / (2*coefs[3])
cat(sprintf("\nPeak Y occurs at X ≈ %.2f\n", peak_x)
```

==========

--- 2.4 HETEROGENEOUS TREATMENT EFFECTS (INTERACTIONS) ---

CONCEPT:
Treatment effect might differ by group:

$$Y = \beta_0 + \beta_1 D + \beta_2 \text{Female} + \beta_3 (D \times \text{Female}) + \varepsilon$$

β_1 = treatment effect for males (Female=0)
β_3 = ADDITIONAL effect for females

Total effect for females = $\beta_1 + \beta_3$

WHEN TO USE:
- Testing if treatment works differently for different groups
- Policy evaluation: Does program help everyone equally?
- Examining heterogeneous impacts

R CODE EXAMPLE:

```
# Generate data with heterogeneous treatment effects
set.seed(123)
n <- 400
df <- data.frame(
  treatment = rbinom(n, 1, 0.5),
  female = rbinom(n, 1, 0.5),
  age = rnorm(n, 40, 15)
)

# Outcome depends on treatment, but effect differs by gender!
df$outcome <- 100 +
```

```
            10*df$treatment +             # Main treatment effect
            -5*df$female +                # Female penalty
            5*df$treatment*df$female +    # INTERACTION: females benefit more!
            0.5*df$age +
            rnorm(n, 0, 10)

# Fit model WITH interaction
het_model <- lm(outcome ~ treatment + female + treatment:female + age, data = df)

summary(het_model)

coefs <- coef(het_model)

cat("=== HETEROGENEOUS TREATMENT EFFECTS ===\n\n")

te_males <- coefs[2]
te_females <- coefs[2] + coefs[4]

cat(sprintf("Treatment Effect for MALES: %.2f\n", te_males))
cat(sprintf("Treatment Effect for FEMALES: %.2f\n", te_females))
cat(sprintf("\nFemales benefit an additional: %.2f\n", te_females - te_males))

cat("\nInterpretation:\n")
cat("- Treatment increases male outcomes by", round(te_males, 1), "\n")
cat("- Treatment increases female outcomes by", round(te_females, 1), "\n")
cat("- Females benefit MORE from treatment!\n")
```

==========

--- 2.5 DOUBLE MACHINE LEARNING (DML) ---

CONCEPT:
DML estimates treatment effects when:

1. Many confounders (10+)
2. Unknown nonlinear relationships
3. Need to control complex confounding

Four steps:

1. Fit ML model to predict Y from X → $\hat{m}(X)$
2. Fit ML model to predict D from X → $\hat{g}(X)$
3. Compute residuals: $\tilde{Y} = Y - \hat{m}(X)$, $\tilde{D} = D - \hat{g}(X)$
4. Regress residuals: $\tilde{Y} = \tau \cdot \tilde{D} + \text{error}$

WHEN TO USE:
- Many confounders (difficult to specify manually)
- Nonlinear confounding
- High-dimensional covariates

R CODE EXAMPLE:

```
# Example: Training effect on earnings with many confounders
set.seed(123)
n <- 1000

# Create many confounders
df <- data.frame(
  age = rnorm(n, 40, 15),
  education = rnorm(n, 13, 2),
  experience = rnorm(n, 15, 10),
  iq = rnorm(n, 100, 15),
  motivation = rnorm(n, 0, 1),
  x6 = rnorm(n, 0, 1),
  x7 = rnorm(n, 0, 1),
  x8 = rnorm(n, 0, 1),
  x9 = rnorm(n, 0, 1),
  x10 = rnorm(n, 0, 1)
)

# Training selection depends on confounders (complex nonlinear)
df$training <- rbinom(n, 1,
                      plogis(-1 + 0.02*df$age + 0.1*df$education +
                          0.001*df$iq + 0.5*df$motivation))

# Outcome depends on confounders AND training
df$earnings <- 25000 +
               5000*df$training +          # True training effect
               500*df$age +
               1000*df$education +
               200*df$experience +
               50*df$iq +
               2000*df$motivation +
               rnorm(n, 0, 3000)

# APPROACH 1: Simple OLS (BIASED)
simple_model <- lm(earnings ~ training, data = df)
```

```
simple_est <- coef(simple_model)[2]

# APPROACH 2: OLS with some controls (BETTER but maybe incomplete)
ols_model <- lm(earnings ~ training + age + education + experience + iq + motivation,
                data = df)
ols_est <- coef(ols_model)[2]

# APPROACH 3: DML (MOST FLEXIBLE)
# Note: Real DML requires ML packages like causalml or doubleml
# Here's a simplified version using built-in methods

# Step 1: Predict earnings from X (no training)
X_cols <- c("age", "education", "experience", "iq", "motivation",
            "x6", "x7", "x8", "x9", "x10")
outcome_model <- lm(earnings ~ .,
                    data = df[, c(X_cols, "earnings")])
pred_earnings <- predict(outcome_model)

# Step 2: Predict training from X
treatment_model <- glm(training ~ .,
                       family = binomial,
                       data = df[, c(X_cols, "training")])
pred_prob <- predict(treatment_model, type = "response")

# Step 3: Compute residuals
resid_earnings <- df$earnings - pred_earnings
resid_training <- df$training - pred_prob

# Step 4: Estimate effect
dml_model <- lm(resid_earnings ~ resid_training)
dml_est <- coef(dml_model)[2]

# COMPARE ESTIMATES
cat("=== COMPARISON OF ESTIMATES ===\n\n")
cat(sprintf("Simple OLS (BIASED):          $%.0f\n", simple_est))
cat(sprintf("OLS with controls (BETTER):    $%.0f\n", ols_est))
cat(sprintf("DML (MOST FLEXIBLE):           $%.0f\n", dml_est))
cat(sprintf("True effect:                   $5,000\n")



INTERPRETATION:

Simple OLS overestimates because:
  - Ambitious people select into training
```

- Ambitious people earn more anyway
- This inflates the training coefficient

DML does better by:
- Using flexible ML to capture complex confounding
- Controlling for nonlinear relationships automatically
- Not requiring us to manually specify functional form

KEY ADVANTAGES OF DML:

1. Handles high dimensions
   - Simple OLS struggles with 10+ confounders
   - DML uses one flexible model for all

2. Discovers functional form
   - Don't need to guess which X^2 terms to include
   - Don't need to specify interactions
   - ML finds patterns automatically

3. Orthogonalization
   - Removes confounding in clean way
   - Robust to moderate overfitting
   - Valid causal inference under unconfoundedness

ASSUMPTIONS FOR CAUSALITY:

1. Unconfoundedness: All confounders in X
   - If unmeasured confounders exist, still biased!

2. Overlap: $0 < P(D=1|X) < 1$
   - Every unit has positive probability of treatment

3. Consistency: $Y = Y_d$ when unit gets D
   - No multiple versions of treatment

4. No interference: Treatment of one unit doesn't affect others
   - Training one person doesn't affect others' wages

==========

SUMMARY TABLE: WHEN TO USE EACH METHOD

==========

METHOD	WHEN TO USE	KEY ASSUMPTION
Simple OLS	Randomized experiment	Treatment is random
Linear w/ controls	Few confounders (3-5)	Unconfoundedness
Nonlinear terms	Known nonlinear pattern	Unconfoundedness
Interactions	Expecting group diffs	Unconfoundedness
Logistic	Binary outcome	Standard assumptions
Odds ratios	Logistic regression	Unconfoundedness
DML	Many confounders (10+)	Unconfoundedness

KEY FORMULAS REFERENCE

1. Linear Regression Coefficient:
   $\beta_j = Cov(X_j, Y) / Var(X_j)$

2. t-statistic:
   t = coefficient / standard_error

3. p-value (two-tailed):
   p = 2 × P(|t| > |t_observed|)

4. R-squared:
   $R^2 = 1 - (SS\_residual / SS\_total)$

5. Marginal effect (quadratic):
   $\partial Y/\partial X = \beta_1 + 2\beta_2 X$

6. Interaction interpretation:
   Effect of X_1 when $X_2 = x$: $\beta_1 + \beta_3 \cdot x$

7. Odds ratio:
   OR = exp(logit coefficient)

8. Predicted probability (logistic):
   $P(Y=1|X) = 1 / (1 + e^{(-X\beta)})$

9. DML residuals:
   $\tilde{Y} = Y - \hat{m}(X)$
   $\tilde{D} = D - \hat{g}(X)$

10. DML treatment effect:
$\hat{\tau} = \text{Cov}(\tilde{D}, \tilde{Y}) / \text{Var}(\tilde{D})$

==========

EXAM TIPS

==========

1. SIGNIFICANCE TESTING:
- Always check p-value (< 0.05 = significant)
- Report both coefficient AND p-value
- Interpret direction and magnitude

2. R^2 INTERPRETATION:
- Compare models: higher R^2 = better fit
- $R^2 < 0.3$ = weak fit
- $R^2 > 0.7$ = good fit
- Use adjusted R^2 when comparing different numbers of variables

3. NONLINEAR TERMS:
- Test if quadratic term is significant
- If significant: effect depends on X level
- Calculate and report marginal effects at specific X values

4. INTERACTIONS:
- ALWAYS interpret conditionally
- Must report BOTH β_1 and β_3
- Show effect at each level of interacting variable

5. LOGISTIC REGRESSION:
- Interpret via odds ratios (exp(β))
- Remember: same OR → different probability changes
- Always predict probabilities for specific scenarios

6. CAUSALITY INFERENCE:
- Unconfoundedness is critical (can't be tested!)
- Adding controls reduces but doesn't eliminate bias
- DML more flexible but still needs unconfoundedness

7. DML CONTEXT:
- Use when: Many confounders + complex relationships
- Know the 4 steps: outcome model, treatment model, residuals, OLS
- Remember: Still assumes unconfoundedness!