Linear Regression

2025-12-05

Table of contents

1 Introduction

In this lab, we will focus on the practical aspects of implementing linear regression models in R. The lab complements the respective lecture, but extends it by going from simple linear regression to multiple regression. We'll also explore how to handle non-linear relationships through data transformation practically and discuss the important concept of omitted variable bias, something that was not part of the lecture itself.

1.1 Learning Objectives

By the end of this lab, you will be able to:

- Implement simple and multiple linear regression in R
- Calculate and interpret R^2 manually and using `R` functions
- Understand and demonstrate omitted variable bias
- Perform data transformations to linearize relationships
- Create effective visualizations using `geom_smooth()`
- Conduct basic model diagnostics

1.2 Prerequisites

- Basic understanding of linear regression concepts from lecture
- Familiarity with R and RStudio, as developed in previous labs
- Understanding of basic statistical concepts (mean, variance, correlation), as provided by the statistics recap tutorials

2 Setup and Data

2.1 Loading Required Packages

In this lab we will use the following libraries:

```r
# Load required packages explicitly
library(ggplot2)     # For data visualization
library(dplyr)       # For data manipulation
library(readr)       # For reading CSV files
library(moderndive)  # For regression tables
library(broom)       # For model summaries
library(here)        # For file paths
library(kableExtra)  # For nice HTML tables, optional for you
```

2.2 Loading the Datasets

For this lab, we'll work with the following business data sets that demonstrate different aspects of regression analysis. They are available via the lab webpage.

```r
# Load the three datasets using here package
marketing_data <- read_csv("marketing_roi.csv")
pricing_data <- read_csv("pricing_strategy.csv")
hr_data <- read_csv("hr_salaries.csv")
firm_growth_data <- read_csv("firm_growth.csv")
marketing_efficiency_data <- read_csv("marketing_efficiency.csv")
```

3 Simple Linear Regression

Let's start with the fundamentals of simple linear regression using the marketing dataset. It is always a good idea to first inspect the data set you are using:

```r
glimpse(marketing_data)
```

```
Rows: 36
Columns: 3
$ ad_spend        <dbl> 10751.550, 20766.103, 13179.538, 22660.348, 23809.346,~
$ website_traffic <dbl> 11003.952, 16667.299, 9407.983, 18692.329, 17995.468, ~
$ sales_revenue   <dbl> 46252.00, 82831.26, 44909.07, 83920.10, 84140.37, 3299~
```

3.0.1 Basic Implementation

To fit a linear regression model we use the function `lm()` (which stands for "linear model"). Here we conduct a regression with sales revenue as the dependent, and ad spending as the independent variable:

$$SALES = \beta_0 + \beta_1 EXP_{Ads} + \epsilon$$

To this end we specify the LHS and RHS of the regression equation, separated by a `~`, through the argument `formula`. The variable names must be the same as in the data set we use, and which we specify through the argument `data`:

```r
# Simple regression: Sales Revenue ~ Ad Spend
model_simple <- lm(formula = sales_revenue ~ ad_spend, data = marketing_data)
model_simple
```

```

Call:
lm(formula = sales_revenue ~ ad_spend, data = marketing_data)

Coefficients:
(Intercept)     ad_spend
  14969.808        2.945
```

You can get additional information by using the function `summary()` on the resulting object, or the function `get_regression_table` from the package `moderndive`:

```
# Display results using moderndive for clean output
get_regression_table(model_simple) %>%
  kable(caption = "Simple Linear Regression: Sales Revenue ~ Ad Spend") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Table 1: Simple Linear Regression: Sales Revenue Ad Spend

term	estimate	std_error	statistic	p_value	lower_ci	upper_ci
intercept	14969.808	2351.591	6.366	0	10190.801	19748.815
ad_spend	2.945	0.133	22.139	0	2.674	3.215

3.0.2 Interpreting the basic regression model

We estimated two main parameters of interest, the intercept β_0 and the slope β_1:

```
# Extract key values for interpretation
intercept <- round(coef(model_simple)[1], 0)
slope <- round(coef(model_simple)[2], 3)
```

In our fitted model we have:

- $\hat{\beta}_0 = 1.497 \times 10^4$
- $\hat{\beta}_1 = 2.945$

The means that:

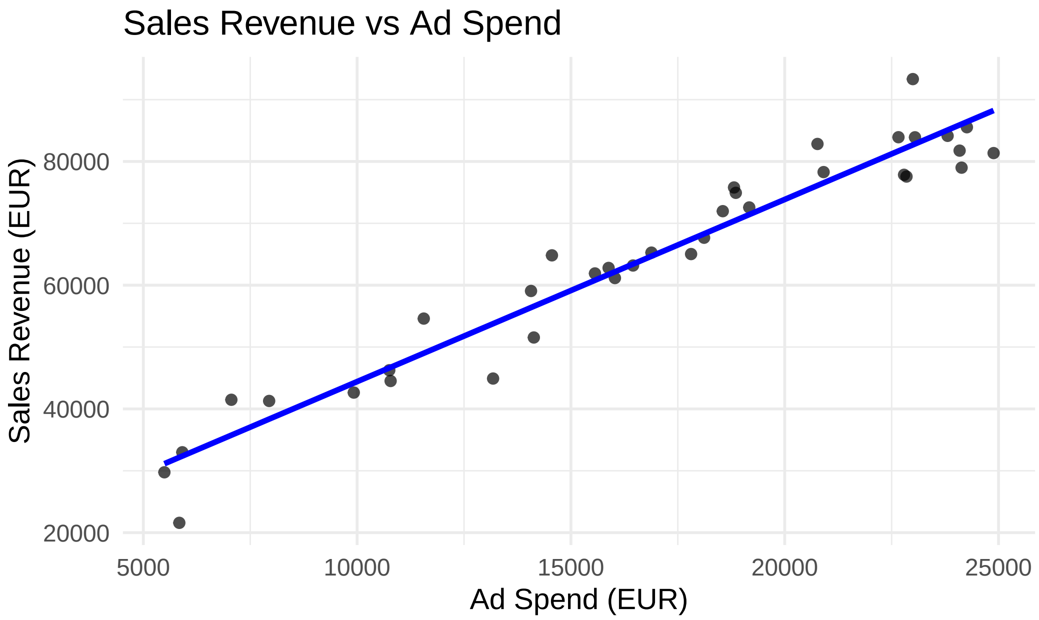
- **Base Revenue**: Even without advertising, we can expect a baseline revenue of 14,970 EUR (although intercepts must be interpreted with great care)
- **Return on Investment**: Every EUR spent on advertising is associated with on average approximately 2.94 EUR in additional sales revenue

For more details, see the accompanying lecture.

3.0.3 Visualizing the Regression

One of the most useful features of `ggplot2` is the `geom_smooth()` function, which can automatically fit and display regression lines:

```
# Create scatter plot with regression line
ggplot(marketing_data, aes(x = ad_spend, y = sales_revenue)) +
  geom_point(alpha = 0.7) +
  geom_smooth(method = "lm", se = FALSE, color = "blue", linewidth = 1) +
  labs(
    title = "Sales Revenue vs Ad Spend",
    x = "Ad Spend (EUR)",
    y = "Sales Revenue (EUR)",
    caption = "Blue line shows linear regression fit"
  ) +
  theme_minimal()
```



Blue line shows linear regression fit

Understanding geom_smooth()

- `method = "lm"` fits a linear model
- `se = FALSE` removes the confidence interval bands
- `se = TRUE` (default) shows 95% confidence intervals
- You can also use `method = "loess"` for flexible, non-parametric fits

4 Multiple Linear Regression

Real-world relationships often involve multiple variables. Let's extend our model to include website traffic.

4.0.1 Adding a Second Predictor

Conceptually, we are now estimating the following model:

$$SALES = \beta_0 + \beta_1 EXP_{Ads} + \beta_2 TRAFFIC + \epsilon$$

To do this in `R`, we proceed exactly as before and just add the new independent variable to the `formula`:

```
# Multiple regression: Sales ~ Ad Spend + Website Traffic
model_multiple <- lm(sales_revenue ~ ad_spend + website_traffic,
                     data = marketing_data)

get_regression_table(model_multiple) %>%
  kable(caption = "Multiple Linear Regression: Sales Revenue ~ Ad Spend + Website Traffic
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Table 2: Multiple Linear Regression: Sales Revenue Ad Spend + Website Traffic

term	estimate	std_error	statistic	p_value	lower_ci	upper_ci
intercept	12965.099	2398.245	5.406	0.000	8085.833	17844.365
ad_spend	2.216	0.349	6.341	0.000	1.505	2.927
website_traffic	0.985	0.440	2.236	0.032	0.089	1.881

4.1 Interpreting Both Predictors

We now estimated *three* main parameters of interest, the intercept β_0 and two slope parameters β_1, and β_2:

```
# Extract coefficients for interpretation
intercept_mult <- round(coef(model_multiple)[1], 0)
ad_coef_mult <- round(coef(model_multiple)[2], 3)
traffic_coef <- round(coef(model_multiple)[3], 3)
```

In our fitted model we have:

- $\hat{\beta}_0 = 1.2965 \times 10^4$
- $\hat{\beta}_1 = 2.216$
- $\hat{\beta}_2 = 0.985$

In the multiple regression framework the interpretation of the coefficients changes slightly, as we now estimate *ceteris paribus* effects, sometimes also called *direct effects*, in contrast to the *total association* that was our focus in the simple regression framework.

More precisely, the estimates in the multiple regression framework control for changes in the other variables. The means that these are the changes we can expect in the dependent variable, if the independent variable changed by one unit, and all other variables stayed the same.

More precisely:

- $\hat{\beta}_1 = 2.216$ means that for every increase of 1 EUR in ad spending, there is an associated increase of revenue of, on average and ceteris paribus, 2.216 EUR.

- $\hat{\beta}_2 = 0.985$ means that every increase of website traffic by 1 person is an associated with an increase of revenue of, on average and ceteris paribus, 0.985 EUR.

Note that the *indirect effect* of ad spending is different to the direct effect estimated! More precisely, the direct effect of advertising (2.216 EUR) is smaller than the total effect (2.945 EUR) because some of advertising's impact works through increased website traffic, and the multiple regression framework helps us to identify these different channels.

We also say that the simple regression model was *confounding* the effects of ad spend and website traffic!

4.2 Omitted Variable Bias

This brings is to one of the most important concepts in regression analysis: *omitted variable bias*. It occurs when both of the following conditions are met:

- There is a variable that correlates with our dependent variable **and…**
- also correlates with one of our independent variables.

4.2.1 The Theory

When we omit a relevant variable from our regression, the coefficients of included variables become **biased**. The bias depends on:

1. How strongly the omitted variable affects the outcome
2. How correlated the omitted variable is with included variables

4.2.2 Example 1: Marketing Data - Coefficient Size Change

In our previous example, $\hat{\beta}_1$ for the simple model was 2.94 and for the multiple model it was 2.22, meaning that we have a total bias of 0.73.

Since website traffic is positively correlated with ad spend AND positively affects sales, the simple model **overestimates** the effect of ad spend.

4.2.3 Example 2: When Coefficients Change Sign Completely

Sometimes omitted variable bias is so severe that it completely changes the **direction** of the relationship. Let's look at the first example of the lecture, the analysis of beer consumption:

```r
beer_data <- DataScienceExercises::beer %>%
  dplyr::mutate(income = income/1000)

model_1 <- lm(
  formula = consumption ~ income,
  data=beer_data)
```

	Simple	Model 2	Model 3	Model 4
(Intercept)	96.439***	57.160***	67.440**	82.159***
	(7.521)	(9.468)	(19.995)	(17.962)
income	−1.237***	2.580**	2.776**	1.995*
	(0.229)	(0.769)	(0.847)	(0.776)
price		−27.653***	−25.968***	−23.743***
		(5.438)	(6.212)	(5.429)
price_liquor			−2.611	−4.077
			(4.457)	(3.890)
price_other				12.924**
				(4.164)
Num.Obs.	30	30	30	30
R2	0.511	0.750	0.754	0.822

+ p <0.1, * p <0.05, ** p <0.01, *** p <0.001

```r
model_2 <- lm(
  formula = consumption ~ income + price,
  data=beer_data)
model_3 <- lm(
  formula = consumption ~ income + price + price_liquor,
  data=beer_data)
model_4 <- lm(
  formula = consumption ~ income + price  + price_liquor + price_other,
  data=beer_data)
```

```r
modelsummary::modelsummary(
  models = list(
    "Simple"=model_1,
    "Model 2"=model_2,
    "Model 3"=model_3,
    "Model 4"=model_4),
  gof_map = c("nobs", "r.squared"),
  stars = TRUE)
```

This example shows how omitted variable bias can lead to completely **wrong business conclusions**:

- **Wrong conclusion** (simple model): "Higher-income customers drink less beer - target low-income segments"
- **Correct conclusion** (multiple model): "Higher-income customers drink more beer, but they're price-sensitive - consider premium pricing strategies"

The simple regression result could lead to disastrous policy decisions!

5 Model Diagnostics

Good regression analysis doesn't stop at fitting the model. We need to check our assumptions and test how good the model explains the dependent variable.

5.1 The explanatory power: R^2

R^2 measures the proportion of variation in the dependent variable explained by our model.

It is defined as the ratio between the total variation ('Total Sum of Squares') and the explained variation. Since the residuals are our measure for *unexplained* variation (see the lecture), we can compute R^2 by the following formula:

$$R^2 = 1 - \frac{RSS}{TSS}$$

Let's calculate it manually to understand what it means.

```r
# Get the actual and fitted values
y_actual <- marketing_data$sales_revenue
y_fitted <- predict(model_simple)
y_mean <- mean(y_actual)

# Calculate the components
TSS <- sum((y_actual - y_mean)^2)      # Total Sum of Squares
RSS <- sum((y_actual - y_fitted)^2)    # Residual Sum of Squares

# Calculate R²
r_squared_manual <- 1 - (RSS / TSS)

cat("Manual R² calculation:", round(r_squared_manual, 4), "\n")
```

```
Manual R² calculation: 0.9351
```

```r
cat("R's built-in R²:", round(summary(model_simple)$r.squared, 4), "\n")
```

```
R's built-in R²: 0.9351
```

But in practice there is no need to compute it manually, as it is stored in the `summary` of every regression object:

```r
# Compare R² between models
r2_simple <- summary(model_simple)$r.squared
r2_multiple <- summary(model_multiple)$r.squared

cat("Simple model R²:", round(r2_simple, 4), "\n")
```

```
Simple model R²: 0.9351
```

```
cat("Multiple model R²:", round(r2_multiple, 4), "\n")
```

```
Multiple model R²: 0.9437
```

```
cat("Improvement:", round(r2_multiple - r2_simple, 4), "\n")
```

```
Improvement: 0.0085
```

In our case:

- The R^2 of the simple model is 0.9351. This means that this model explains 0.94 % of the variation in the dependent variable.
- The R^2 of the multiple model is 0.9437. This means that this model explains 0.94 % of the variation in the dependent variable.

But be careful: a higher R^2 does not necessarily imply better prediction or inference capability of the model!

5.2 Key Diagnostic Plots

The most important assumptions of linear regression are about the nature of the error term ϵ. But as discussed previously, the error term is located on the population level, meaning it will always remain unobservable. Therefore, we usually inspect its sample equivalent, the residuals, to test the most important assumptions.

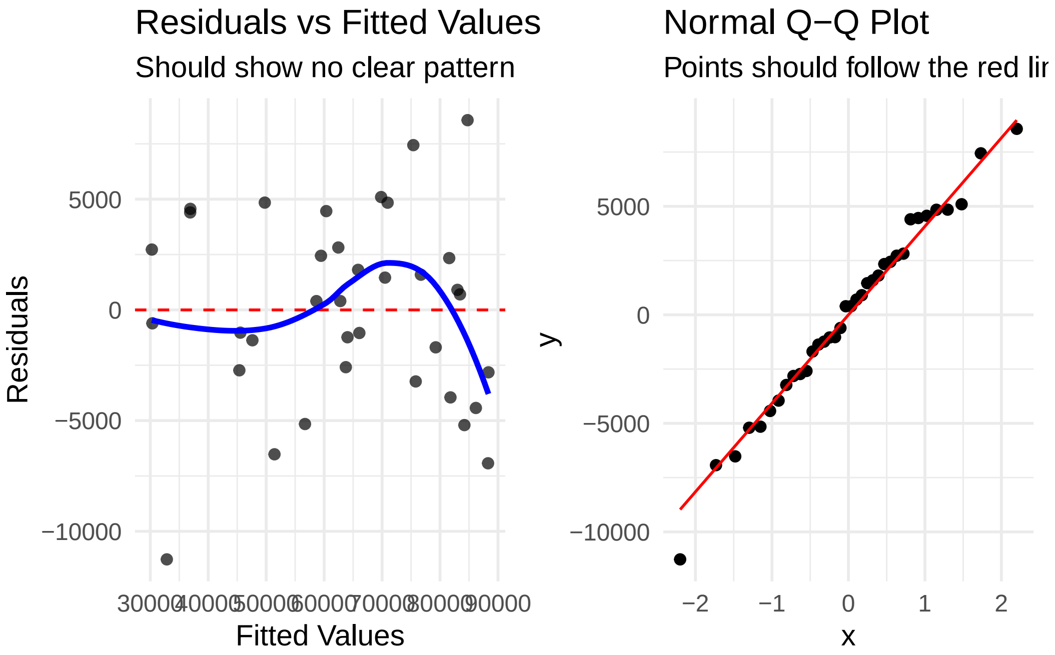
The two most important assumptions are:

1. The error term is uncorrelated with the dependent variable.
2. The error follows a normal distribution with mean zero.

To test assumption 1, we can look at the correlation between residuals and fitted values. If our model is good, we should see no structure.

To test assumption 2, we can look at a so called QQ-plot. QQ stands for quantile-quantile. Such a plot plots the quantiles of the data (here: residuals) against the theoretical quantiles if the data was following a normal distribution. If the residuals are normally distributed, they should follow a straight line.

```
# Create augmented data with fitted values and residuals
marketing_augmented <- augment(model_multiple)

# Tukey-Anscombe Plot (Residuals vs Fitted)
p1 <- ggplot(marketing_augmented, aes(x = .fitted, y = .resid)) +
  geom_point(alpha = 0.7) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  geom_smooth(se = FALSE, color = "blue") +
  labs(
    title = "Residuals vs Fitted Values",
    x = "Fitted Values",
    y = "Residuals",
    subtitle = "Should show no clear pattern"
  ) +
  theme_minimal()

# Normal Q-Q Plot
p2 <- ggplot(marketing_augmented, aes(sample = .resid)) +
  stat_qq() +
  stat_qq_line(color = "red") +
  labs(
    title = "Normal Q-Q Plot",
    subtitle = "Points should follow the red line"
  ) +
  theme_minimal()

gridExtra::grid.arrange(p1, p2, ncol = 2)
```



In this example we see that both assumptions seem to be satisfied, although some small structure in the residuals persist. In the end, there is no clear-cut rule when there is no structure, but it is usually difficult to do an even better job as in this example.

6 Data Transformation for Non-linear Relationships

Not all relationships are linear! But remember that for linear regression, we only need to assume *linearity in parameters*! This means that sometimes we can transform our data to make initially non-linear relationships linear and still use linear regression effectively.

6.1 When to Transform Data

Common scenarios requiring transformation:

- **Exponential relationships**: Often seen with salary/experience, population growth
- **U-shaped relationships**: Common in economics (e.g., effort vs performance)
- **Diminishing returns**: Sales response to advertising often follows this pattern

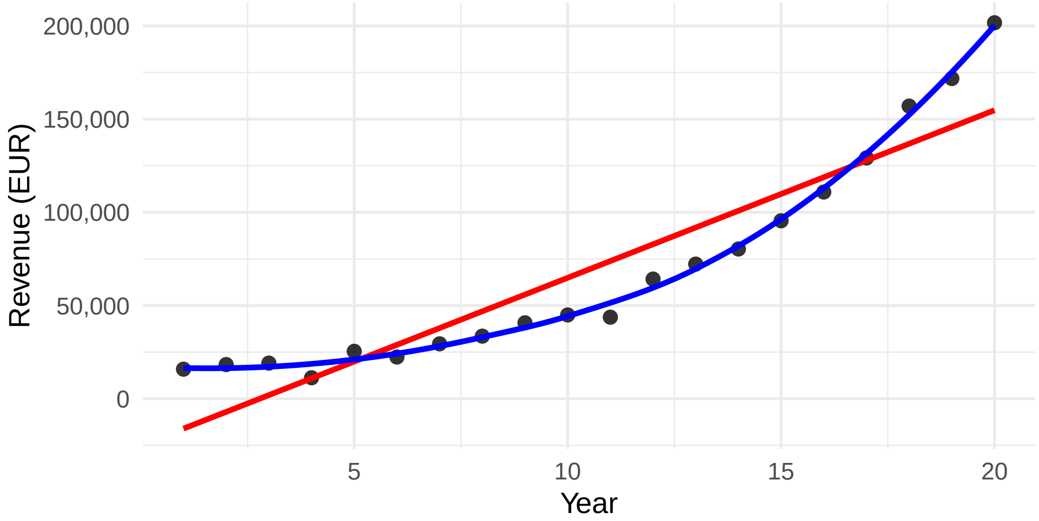
In practice, always visualize your data first and experiment with various transformation strategies - this is the best way to determine whether data transformation can help you.

Also, the diagnostic plots discussed above can also provide hints on necessary transformations: if the Tukey-Anscombe plot shows structure, think about transforming the data!

6.2 Log Transformation Example

Let's examine a **company's revenue growth over time** - a classic example of exponential growth that appears linear after log transformation:

```
# First, visualize the raw exponential relationship
ggplot(firm_growth_data, aes(x = year, y = revenue)) +
  geom_point(size = 2, alpha = 0.8) +
  geom_smooth(method = "lm", se = FALSE, color = "red", linewidth = 1) +
  geom_smooth(method = "loess", se = FALSE, color = "blue", linewidth = 1) +
  labs(
    title = "Company Revenue Growth Over Time (Raw Data)",
    subtitle = "Red = Linear fit (clearly inadequate), Blue = Flexible fit",
    x = "Year",
    y = "Revenue (EUR)",
    caption = "Notice how the linear fit completely misses the exponential pattern"
  ) +
  scale_y_continuous(labels = scales::comma) +
  theme_minimal()
```



Company Revenue Growth Over Time (Raw Data)

Red = Linear fit (clearly inadequate), Blue = Flexible fit

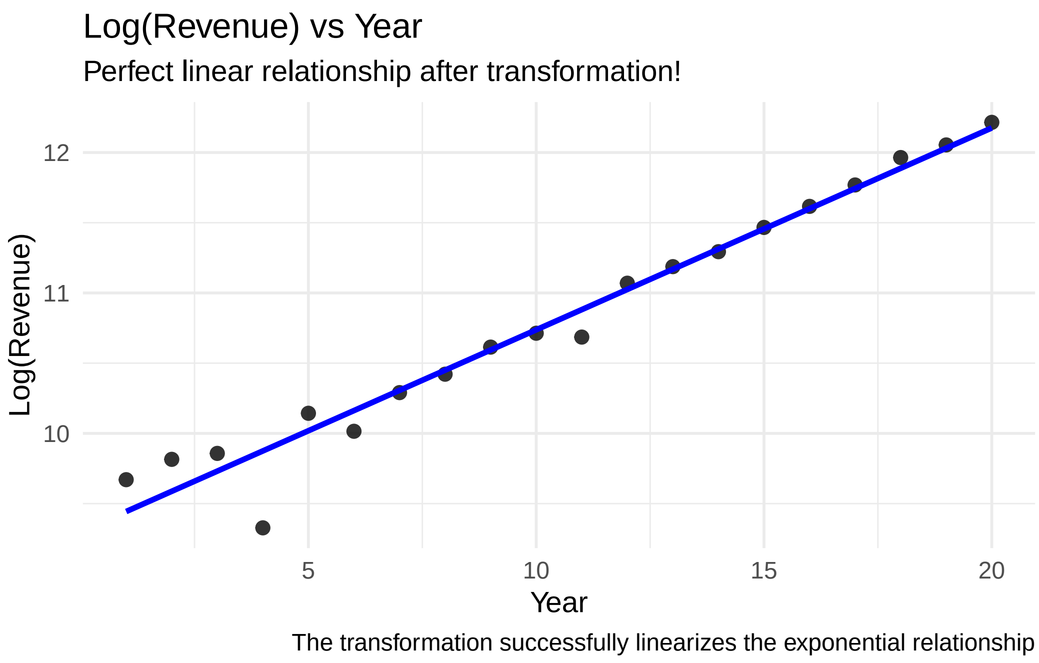
Notice how the linear fit completely misses the exponential pattern

The exponential pattern is very clear! The red linear line completely fails to capture the relationship. Let's try a log transformation:

```
# Create log-transformed variable
firm_growth_data_log <- firm_growth_data %>%
  mutate(log_revenue = log(revenue))

# Fit both models
model_linear_exp <- lm(revenue ~ year, data = firm_growth_data)
model_log_exp <- lm(log_revenue ~ year, data = firm_growth_data_log)

# Visualize the log-transformed relationship
ggplot(firm_growth_data_log, aes(x = year, y = log_revenue)) +
  geom_point(size = 2, alpha = 0.8) +
  geom_smooth(method = "lm", se = FALSE, color = "blue", linewidth = 1) +
  labs(
    title = "Log(Revenue) vs Year",
    subtitle = "Perfect linear relationship after transformation!",
    x = "Year",
    y = "Log(Revenue)",
    caption = "The transformation successfully linearizes the exponential relationship"
  ) +
  theme_minimal()
```

	Lin-Lin	Log-lin
(Intercept)	−25 037.426*	9.300***
	(10 107.611)	(0.078)
year	8991.437***	0.144***
	(843.767)	(0.007)
Num.Obs.	20	20
R2	0.863	0.964

+ p < 0.1, * p < 0.05, ** p < 0.01, *** p < 0.001


Log(Revenue) vs Year

Perfect linear relationship after transformation!

The transformation successfully linearizes the exponential relationship

```
modelsummary::modelsummary(
  models = list("Lin-Lin"=model_linear_exp, "Log-lin"=model_log_exp),
  gof_map = c("nobs", "r.squared"),
  stars = TRUE, output = "kableExtra")
```

6.2.1 Interpreting Log Models

When you transform data, you need to interpret your results accordingly. For instance, when the dependent variable is log-transformed, coefficients represent **percentage changes**:

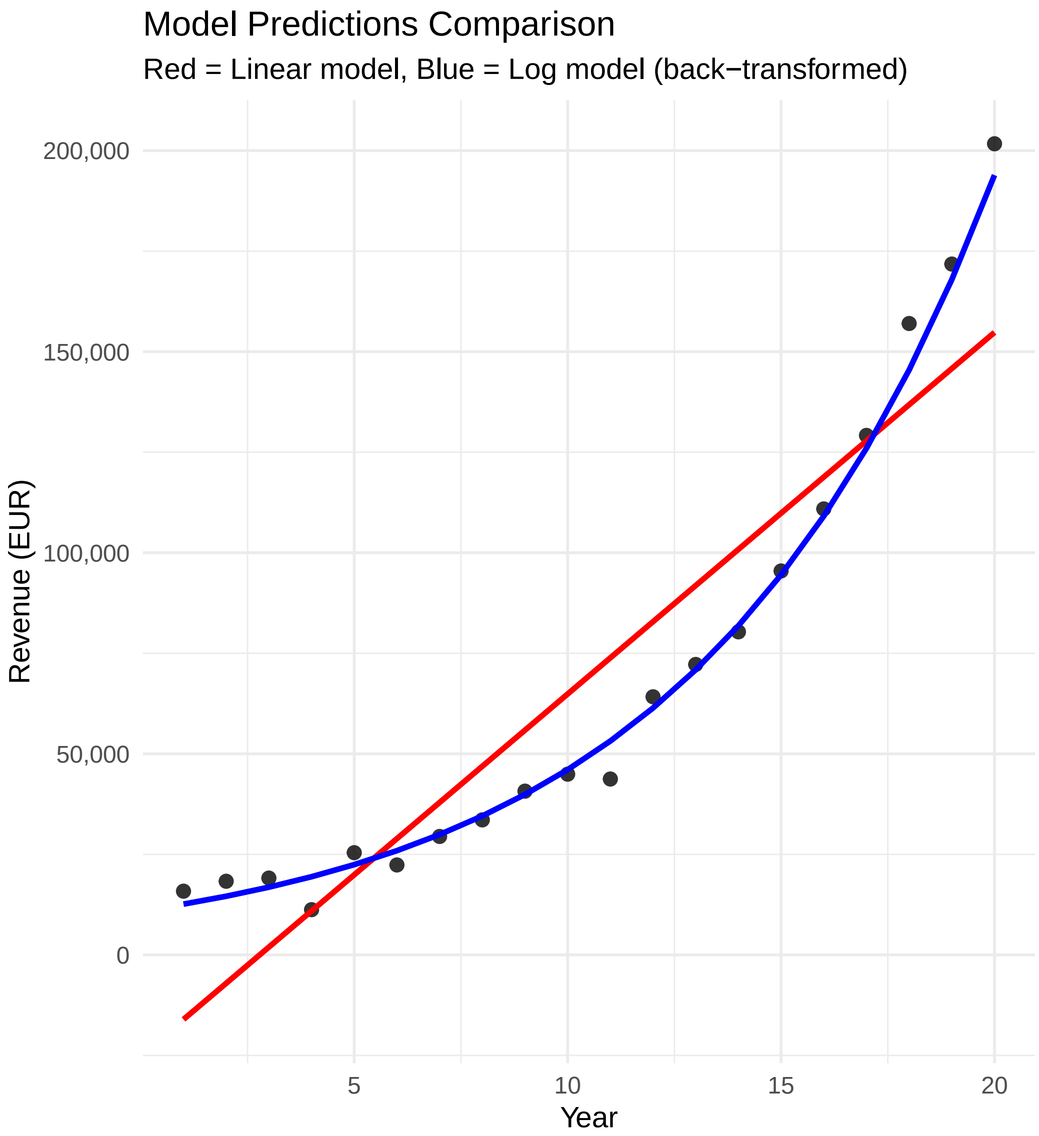
> Every year is, on average, associated with a
> 14.4 % increase in revenue.

❗ Why Log Transformation Works for Exponential Data

Exponential relationships have the form: $Y = A \cdot e^{Bx}$
Taking the natural log: $\ln(Y) = \ln(A) + Bx$
This transforms the exponential curve into a straight line where: - The slope B represents the **growth rate** - $e^B - 1$ gives the percentage change per unit increase in x

The residual analysis discussed below can reveal how much better to log model performs in this context. But also by plotting the model on the raw data already makes this very clear:

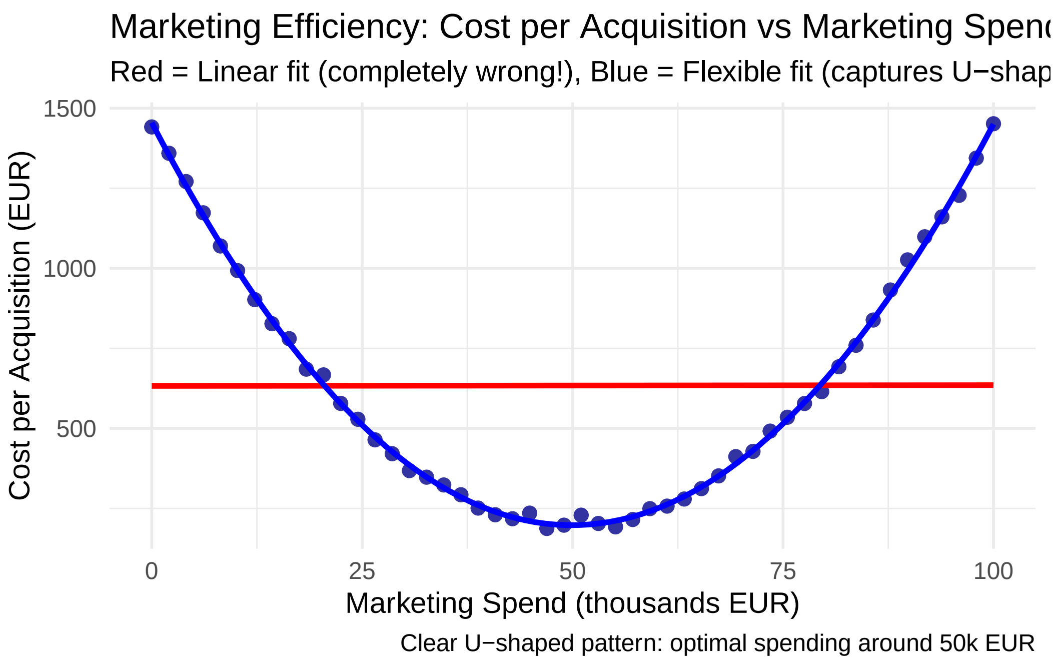
```r
# Show predictions on original scale
exp_predictions <- firm_growth_data %>%
  mutate(
    linear_pred = predict(model_linear_exp),
    log_pred = exp(predict(model_log_exp, newdata = firm_growth_data_log))
  )

p3 <- ggplot(exp_predictions, aes(x = year)) +
  geom_point(aes(y = revenue), size = 2, alpha = 0.8) +
  geom_line(aes(y = linear_pred), color = "red", linewidth = 1) +
  geom_line(aes(y = log_pred), color = "blue", linewidth = 1) +
  labs(title = "Model Predictions Comparison",
       subtitle = "Red = Linear model, Blue = Log model (back-transformed)",
       x = "Year", y = "Revenue (EUR)") +
  scale_y_continuous(labels = scales::comma) +
  theme_minimal()
p3
```



6.3 Quadratic Relationships

Sometimes relationships are U-shaped or inverted-U shaped. A perfect example is **marketing efficiency**: too little spending is inefficient (high fixed costs), and too much spending leads to diminishing returns. Let's explore this dramatic pattern:

```
# First, visualize the clear U-shaped relationship
ggplot(marketing_efficiency_data, aes(x = marketing_spend, y = cost_per_acquisition)) +
  geom_point(size = 2, alpha = 0.8, color = "darkblue") +
  geom_smooth(method = "lm", se = FALSE, color = "red", linewidth = 1) +
  geom_smooth(method = "loess", se = FALSE, color = "blue", linewidth = 1) +
  labs(
    title = "Marketing Efficiency: Cost per Acquisition vs Marketing Spend",
    subtitle = "Red = Linear fit (completely wrong!), Blue = Flexible fit (captures U-sha
    x = "Marketing Spend (thousands EUR)",
    y = "Cost per Acquisition (EUR)",
    caption = "Clear U-shaped pattern: optimal spending around 50k EUR"
  ) +
  theme_minimal()
```



The U-shaped pattern is unmistakable! Now let's compare linear vs quadratic models:

```
# Create squared term for quadratic model
marketing_efficiency_data <- marketing_efficiency_data %>%
  mutate(marketing_spend_squared = marketing_spend^2)

# Fit both models
marketing_linear <- lm(cost_per_acquisition ~ marketing_spend, data = marketing_efficienc
marketing_quadratic <- lm(cost_per_acquisition ~ marketing_spend + marketing_spend_square
                          data = marketing_efficiency_data)

# Show regression results
cat("=== LINEAR MODEL (completely inadequate) ===\n")
```

```
=== LINEAR MODEL (completely inadequate) ===
```

```
get_regression_table(marketing_linear) %>%
  kable(caption = "Linear Model: Cost per Acquisition ~ Marketing Spend") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Table 3: Linear Model: Cost per Acquisition Marketing Spend

term	estimate	std_error	statistic	p_value	lower_ci	upper_ci
intercept	632.844	110.713	5.716	0.000	410.242	855.447
marketing_spend	0.021	1.908	0.011	0.991	-3.815	3.857

```r
cat("\n=== QUADRATIC MODEL (captures the U-shape) ===\n")
```

```
=== QUADRATIC MODEL (captures the U-shape) ===
```

```r
get_regression_table(marketing_quadratic) %>%
  kable(caption = "Quadratic Model: Cost per Acquisition ~ Marketing Spend + Marketing Sp
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Table 4: Quadratic Model: Cost per Acquisition Marketing Spend + Marketing Spend^2

term	estimate	std_error	statistic	p_value	lower_ci	upper_ci
intercept	1452.080	5.857	247.914	0	1440.297	1463.863
marketing_spend	-50.157	0.271	-185.159	0	-50.702	-49.612
marketing_spend_squared	0.502	0.003	191.548	0	0.497	0.507

```r
# Compare R² - dramatic difference!
r2_linear_quad <- summary(marketing_linear)$r.squared
r2_quadratic <- summary(marketing_quadratic)$r.squared

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

```
=== MODEL COMPARISON ===
```

```r
cat("\nLinear model R²:", round(r2_linear_quad, 3))
```

```
Linear model R²: 0
```

```r
cat("\nQuadratic model R²:", round(r2_quadratic, 3))
```

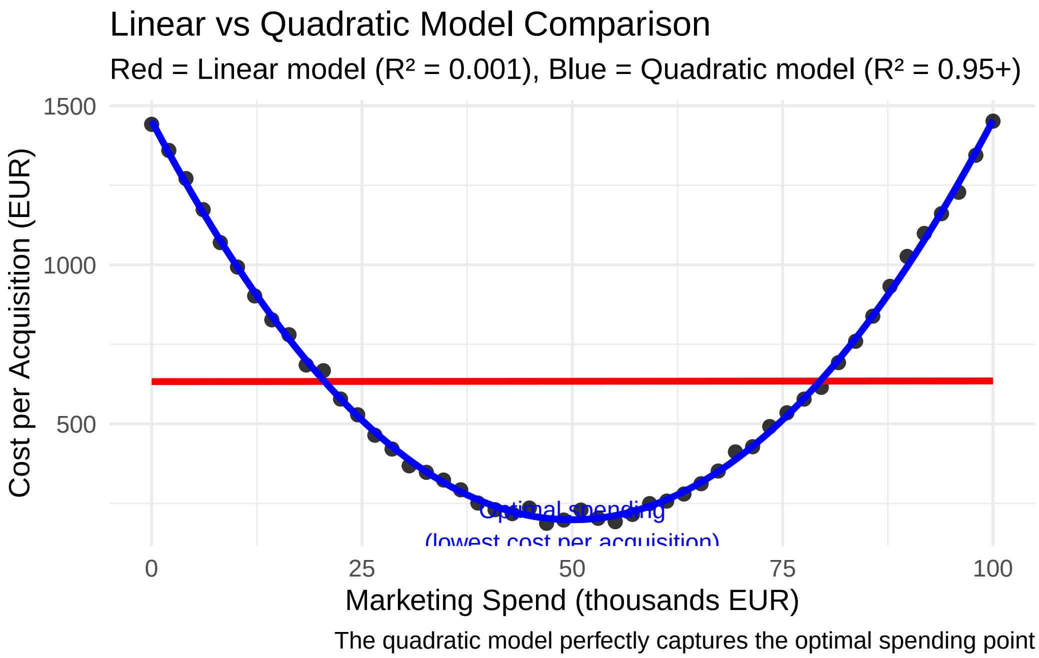
```
Quadratic model R²: 0.999
```

```r
cat("\nImprovement:", round(r2_quadratic - r2_linear_quad, 3))
```

```
Improvement: 0.999
```

Now let's visualize both model fits to see the dramatic difference:

```r
# Create predictions for smooth curves
marketing_range <- seq(0, 100, length.out = 100)
pred_data <- tibble(
  marketing_spend = marketing_range,
  marketing_spend_squared = marketing_range^2
)

# Get predictions
pred_data$linear_pred <- predict(marketing_linear, newdata = pred_data)
pred_data$quadratic_pred <- predict(marketing_quadratic, newdata = pred_data)

# Create comprehensive comparison plot
ggplot(marketing_efficiency_data, aes(x = marketing_spend, y = cost_per_acquisition)) +
  geom_point(size = 2, alpha = 0.8, color = "black") +
  geom_line(data = pred_data, aes(y = linear_pred),
            color = "red", linewidth = 1.2) +
  geom_line(data = pred_data, aes(y = quadratic_pred),
            color = "blue", linewidth = 1.2) +
  labs(
    title = "Linear vs Quadratic Model Comparison",
    subtitle = "Red = Linear model (R² = 0.001), Blue = Quadratic model (R² = 0.95+)",
    x = "Marketing Spend (thousands EUR)",
    y = "Cost per Acquisition (EUR)",
    caption = "The quadratic model perfectly captures the optimal spending point"
  ) +
  annotate("text", x = 50, y = 180,
           label = "Optimal spending\n(lowest cost per acquisition)",
           hjust = 0.5, color = "blue", size = 3) +
  theme_minimal()
```



6.3.1 Understanding Quadratic Interpretation

With quadratic terms, interpretation becomes more nuanced because the effect of the variable **changes** depending on its current level:

```r
# Extract coefficients for interpretation
linear_coef <- coef(marketing_quadratic)["marketing_spend"]
quadratic_coef <- coef(marketing_quadratic)["marketing_spend_squared"]

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

```
=== COEFFICIENT INTERPRETATION ===
```

```r
cat("Linear term coefficient:", round(linear_coef, 3), "\n")
```

```
Linear term coefficient: -50.157
```

```r
cat("Quadratic term coefficient:", round(quadratic_coef, 4), "\n\n")
```

```
Quadratic term coefficient: 0.5018
```

```r
# Find the optimal point (minimum of the parabola)
optimal_spend <- -linear_coef / (2 * quadratic_coef)
cat("Optimal marketing spend:", round(optimal_spend, 1), "thousand EUR\n")
```

```
Optimal marketing spend: 50 thousand EUR
```

```
# Calculate minimum cost per acquisition
optimal_cost <- predict(marketing_quadratic,
                       newdata = tibble(marketing_spend = optimal_spend,
                                       marketing_spend_squared = optimal_spend^2))
cat("Minimum cost per acquisition:", round(optimal_cost, 0), "EUR\n")
```

```
Minimum cost per acquisition: 199 EUR
```

6.3.2 Marginal Effects in Quadratic Models

The **marginal effect** (slope) at any point is: Linear coefficient + 2 × Quadratic coefficient × X

```
# Calculate marginal effects at different spending levels
spending_levels <- c(20, 40, 50, 60, 80)

marginal_effects <- linear_coef + 2 * quadratic_coef * spending_levels

marginal_table <- tibble(
  `Spending Level (k EUR)` = spending_levels,
  `Marginal Effect` = round(marginal_effects, 3),
  `Interpretation` = case_when(
    marginal_effects < -0.1 ~ "Strong efficiency gains from more spending",
    marginal_effects > 0.1 ~ "Diminishing returns - reduce spending",
    TRUE ~ "Near optimal - small changes have little effect"
  )
)

marginal_table %>%
  kable(caption = "Marginal Effects at Different Spending Levels") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Table 5: Marginal Effects at Different Spending Levels

Spending Level (k EUR)	Marginal Effect	Interpretation
20	-30.086	Strong efficiency gains from more spending
40	-10.015	Strong efficiency gains from more spending
50	0.021	Near optimal - small changes have little effect
60	10.057	Diminishing returns - reduce spending
80	30.128	Diminishing returns - reduce spending

Tip

Understanding Quadratic Models
The quadratic model equation: Cost = + ×Spend + ×Spend^2
Key insights:

1. **Linear term** (= -50.157): The initial direction of the relationship
2. **Quadratic term** (= 0.5018): How the slope changes
   - Positive → U-shaped (costs increase at extremes)
   - Negative → Inverted-U (optimal point in middle)
3. **Optimal point**: Where marginal effect = 0
4. **Business implication**: There's "sweet spot" for efficiency

6.3.3 Practical Business Application

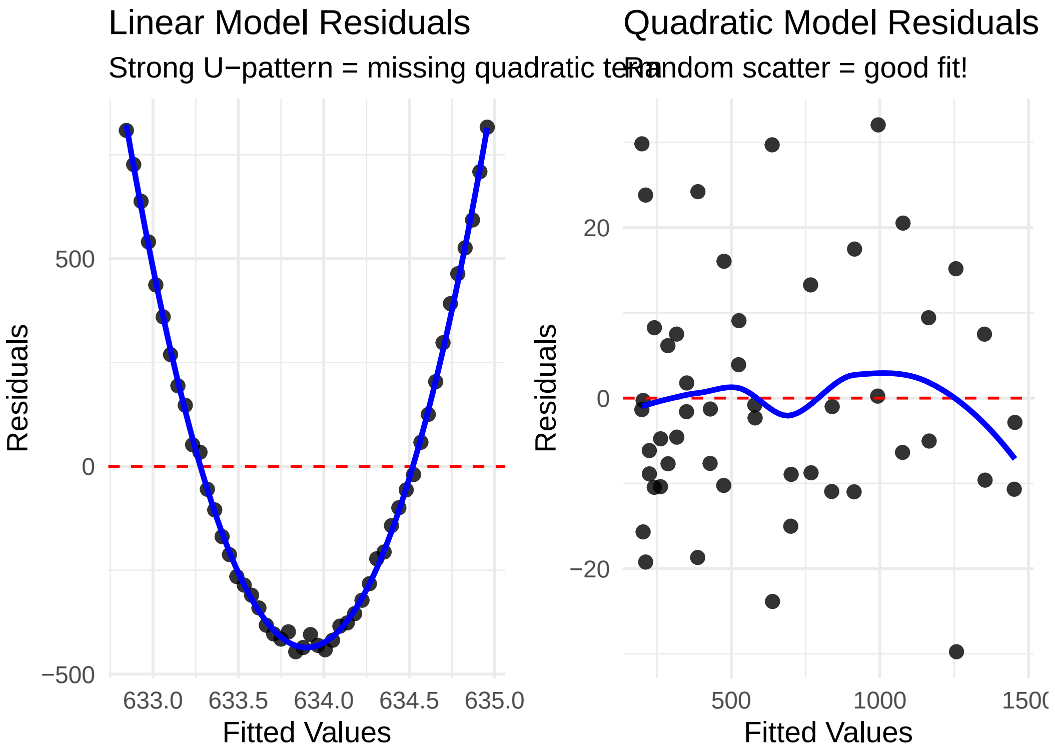
```r
# Create a strategy table based on current spending
current_scenarios <- tibble(
  `Current Spending` = c("10k EUR", "30k EUR", "50k EUR", "70k EUR", "90k EUR"),
  `Predicted Cost` = round(predict(marketing_quadratic,
                                   newdata = tibble(marketing_spend = c(10, 30, 50, 70, 9
                                                    marketing_spend_squared = c(10, 30, 50,
  `Recommendation` = c(
    "Increase spending significantly",
    "Increase spending moderately",
    "Optimal level - maintain",
    "Reduce spending moderately",
    "Reduce spending significantly"
  ),
  `Rationale` = c(
    "Far from optimum, high potential gains",
    "Below optimum, efficiency improvements available",
    "At optimal point for efficiency",
    "Above optimum, diminishing returns setting in",
    "Far above optimum, wasting resources"
  )
)

current_scenarios %>%
  kable(caption = "Strategic Recommendations Based on Quadratic Model") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed")) %>%
  row_spec(3, bold = TRUE, color = "green") # Highlight optimal
```

Table 6: Strategic Recommendations Based on Quadratic Model

Current Spending	Predicted Cost	Recommendation	Rationale
10k EUR	1001	Increase spending significantly	Far from optimum, high potential
30k EUR	399	Increase spending moderately	Below optimum, efficiency improv
50k EUR	**199**	**Optimal level - maintain**	**At optimal point for efficiency**
70k EUR	400	Reduce spending moderately	Above optimum, diminishing retur
90k EUR	1002	Reduce spending significantly	Far above optimum, wasting resou

6.3.4 Residual Analysis Comparison

```r
# Compare residual patterns
library(gridExtra)

# Linear model residuals - should show clear pattern
p1 <- marketing_efficiency_data %>%
  mutate(fitted = fitted(marketing_linear), residuals = residuals(marketing_linear)) %>%
  ggplot(aes(x = fitted, y = residuals)) +
  geom_point(size = 2, alpha = 0.8) +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  geom_smooth(se = FALSE, color = "blue") +
  labs(title = "Linear Model Residuals",
       subtitle = "Strong U-pattern = missing quadratic term",
       x = "Fitted Values", y = "Residuals") +
  theme_minimal()

# Quadratic model residuals - should be random
p2 <- marketing_efficiency_data %>%
  mutate(fitted = fitted(marketing_quadratic), residuals = residuals(marketing_quadratic)
  ggplot(aes(x = fitted, y = residuals)) +
  geom_point(size = 2, alpha = 0.8) +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  geom_smooth(se = FALSE, color = "blue") +
  labs(title = "Quadratic Model Residuals",
       subtitle = "Random scatter = good fit!",
       x = "Fitted Values", y = "Residuals") +
  theme_minimal()

gridExtra::grid.arrange(p1, p2, ncol = 2)
```



ℹ Business Interpretation

The quadratic model reveals important business insights:

1. **Optimal spending**: Around 50k EUR gives the lowest cost per acquisition
2. **Efficiency curve**: Both under-spending and over-spending are inefficient
3. **Marginal effects**: The impact of additional spending depends on current spending level
4. **Strategic implications**: There's a "sweet spot" for marketing efficiency

ℹ Your Turn

1. How would you interpret the linear and quadratic coefficients in business terms?
2. At what spending levels is marketing particularly inefficient?
3. What would happen if we used the linear model for business decisions?

7 Practical Exercises

7.1 Exercise 1: Complete HR Analysis

Analyze the relationship between experience and salary, controlling for education level:

```
# Your turn: Create dummy variables for education and run multiple regression
hr_with_dummies <- hr_data %>%
  mutate(
    master = ifelse(education == "Master", 1, 0),
    phd = ifelse(education == "PhD", 1, 0)
    # Bachelor's degree is the reference category
  )

# Fit the multiple regression model
hr_multiple <- lm(salary ~ experience + master + phd, data = hr_with_dummies)
get_regression_table(hr_multiple) %>%
  kable(caption = "Multiple Regression: Salary ~ Experience + Education") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Table 7: Multiple Regression: Salary Experience + Education

term	estimate	std_error	statistic	p_value	lower_ci	upper_ci
intercept	29729.90	829.309	35.849	0	28093.228	31366.568
experience	1985.06	50.810	39.068	0	1884.784	2085.336
master	10217.58	798.832	12.791	0	8641.056	11794.102
phd	25514.67	1061.832	24.029	0	23419.112	27610.238

Interpretation Challenge

1. How do you interpret the coefficient for `experience` now?
2. What’s the salary premium for having a Master’s degree vs Bachelor’s?
3. How much omitted variable bias was there in the simple experience-only model?

8 Key Takeaways

8.1 Main Learning Points

1. **Multiple regression** controls for confounding variables and reduces omitted variable bias
2. R^2 measures explained variance but isn’t the only criterion for model quality
3. **Data transformation** can help linearize non-linear relationships
4. **geom_smooth(method = “lm”)** provides easy and effective visualization of regression fits
5. **Residual analysis** is crucial for checking model assumptions
6. **Business context** should always guide model interpretation
7. **Omitted variable bias** can completely reverse coefficient signs and lead to wrong business decisions
8. **Quadratic models** capture non-linear relationships and reveal optimal points

8.2 Common Pitfalls to Avoid

- Don't interpret correlation as causation
- Don't ignore potential confounding variables
- Don't rely solely on R^2 for model selection
- Don't forget to check model assumptions through residual analysis
- Don't use linear models for clearly non-linear relationships without considering transformations
- Don't make business decisions based on biased models