Statistical Methods for Research

Longhai Li

2025-10-31

Table of contents

1 Introduction to Statistical Methods for Research

Welcome

This book contains lecture notes for **STAT 845: Statistical Methods for Research** at the **University of Saskatchewan**.

Table of Contents

2 A Quick Introduction to using R for Data Analysis

2.1 Basic R Objects and Operations

```
## create a vector
x <- 1:10
x <- seq (30,3, by = -2)
a <- c(66.32, 69.87, 70.12, 90.37, 50.08, 61.20, 65.00, 57.65)
d <- a [1]
a [1] <- 85.34

mean (a)
```

```
[1] 68.70375
```

```
ma <- mean (a)
## read a vector of numbers from a file
x <- scan("numbers.txt")
x2 <- scan("number2.txt")

## one can also read number withoug saving to a file
y <- scan(text = "7  8  9 10 11 12 13 13 14 17 17 45")

## create a matrix
A <- matrix (0, 4, 2)

A <- matrix (1:8, 4,2)

A
```

```
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
```

```
D <- matrix (a, 4, 2, byrow=T)

D <- matrix(1:8, 2, 4)
D
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
```

```
## create another matrix with all entry 0
B <- matrix (1:5000, 100, 50)

## assign a number to B
B[2,4] <- 45
B[1,]
```

```
 [1]    1  101  201  301  401  501  601  701  801  901 1001 1101 1201 1301 1401
[16] 1501 1601 1701 1801 1901 2001 2101 2201 2301 2401 2501 2601 2701 2801 2901
[31] 3001 3101 3201 3301 3401 3501 3601 3701 3801 3901 4001 4101 4201 4301 4401
[46] 4501 4601 4701 4801 4901
```

```
B[,1]
```

```
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
 [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
 [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
 [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
 [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
 [91]  91  92  93  94  95  96  97  98  99 100
```

```
B[1,] <- 1:50


## create a list
E <- list (newa = a, newA = A)
## list the names of components
names (E)
```

```
[1] "newa" "newA"
```

```
## to look at the component of E
E$newA
```

```
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
```

```
E$newa <- 10:17

## create a dataframe
scores <- c (30, 45, 50)
names <- c("Peter", "John", "Alice")
stat245_scores <- data.frame (names, scores)
stat245_scores
```

```
  names scores
1 Peter     30
2  John     45
3 Alice     50
```

```
stat245_scores$names
```

```
[1] "Peter" "John"  "Alice"
```

```
stat245_scores$scores [1] <- 40
stat245_scores
```

```
  names scores
1 Peter     40
2  John     45
3 Alice     50
```

```
stat245_scores$perc <- stat245_scores$scores/50 * 100
stat245_scores
```

```
  names scores perc
1 Peter     40   80
2  John     45   90
3 Alice     50  100
```

```r
stat245_scores$adj <- stat245_scores$perc + 10
stat245_scores
```

```
  names scores perc adj
1 Peter     40   80  90
2  John     45   90 100
3 Alice     50  100 110
```

```r
################################################################################
```

2.2 Import a dataset into R environment and Simple Operation

```r
################################################################################

## import myagpop.csv into an R data frame called 'myagpop'
agpop <- read.csv("agpop.csv")

## Now, we can use the data:

## preview agpop
head (agpop)
```

```
                 county state acres92 acres87 acres82 farms92 farms87 farms82
1 ALEUTIAN ISLANDS AREA    AK  683533  726596  764514      26      27      28
2        ANCHORAGE AREA    AK   47146   59297  256709     217     245     223
3        FAIRBANKS AREA    AK  141338  154913  204568     168     175     170
4           JUNEAU AREA    AK     210     214     127       8       8      12
5  KENAI PENINSULA AREA    AK   50810   85712   98035      93     119     137
6        AUTAUGA COUNTY    AL  107259  116050  145044     322     388     453
  largef92 largef87 largef82 smallf92 smallf87 smallf82 region
1       14       16       20        6        4        1      W
2        9       10       11       41       52       38      W
3       25       28       21       12       18       25      W
4        0        0        0        5        4        8      W
5        9       18       17       12       18       19      W
6       25       32       32        8       19       17      S
```

```r
## look at the variable name
colnames (agpop)
```

```
 [1] "county"   "state"    "acres92"  "acres87"  "acres82"  "farms92"
 [7] "farms87"  "farms82"  "largef92" "largef87" "largef82" "smallf92"
[13] "smallf87" "smallf82" "region"
```

```
## find number of cols
ncol (agpop)
```

```
[1] 15
```

```
## find number of rows
nrow (agpop)
```

```
[1] 3078
```

```
## access a certain row
agpop [2, ]
```

```
          county state acres92 acres87 acres82 farms92 farms87 farms82 largef92
2 ANCHORAGE AREA    AK   47146   59297  256709     217     245     223        9
  largef87 largef82 smallf92 smallf87 smallf82 region
2       10       11       41       52       38      W
```

```
## access a certain column
agpop [1:20, "acres92"] ## equivalent to
```

```
 [1] 683533  47146 141338    210  50810 107259 167832 177189  48022 137426
[11] 144799  96427  73841 109555 121504  99466  67950  61426  68478  47200
```

```
agpop$acres92[1:20]
```

```
 [1] 683533  47146 141338    210  50810 107259 167832 177189  48022 137426
[11] 144799  96427  73841 109555 121504  99466  67950  61426  68478  47200
```

```
agpop$largef92[1:20]
```

```
 [1] 14  9 25  0  9 25 24 40  6  9 29 18  4 22 24  8  9 13  4  5
```

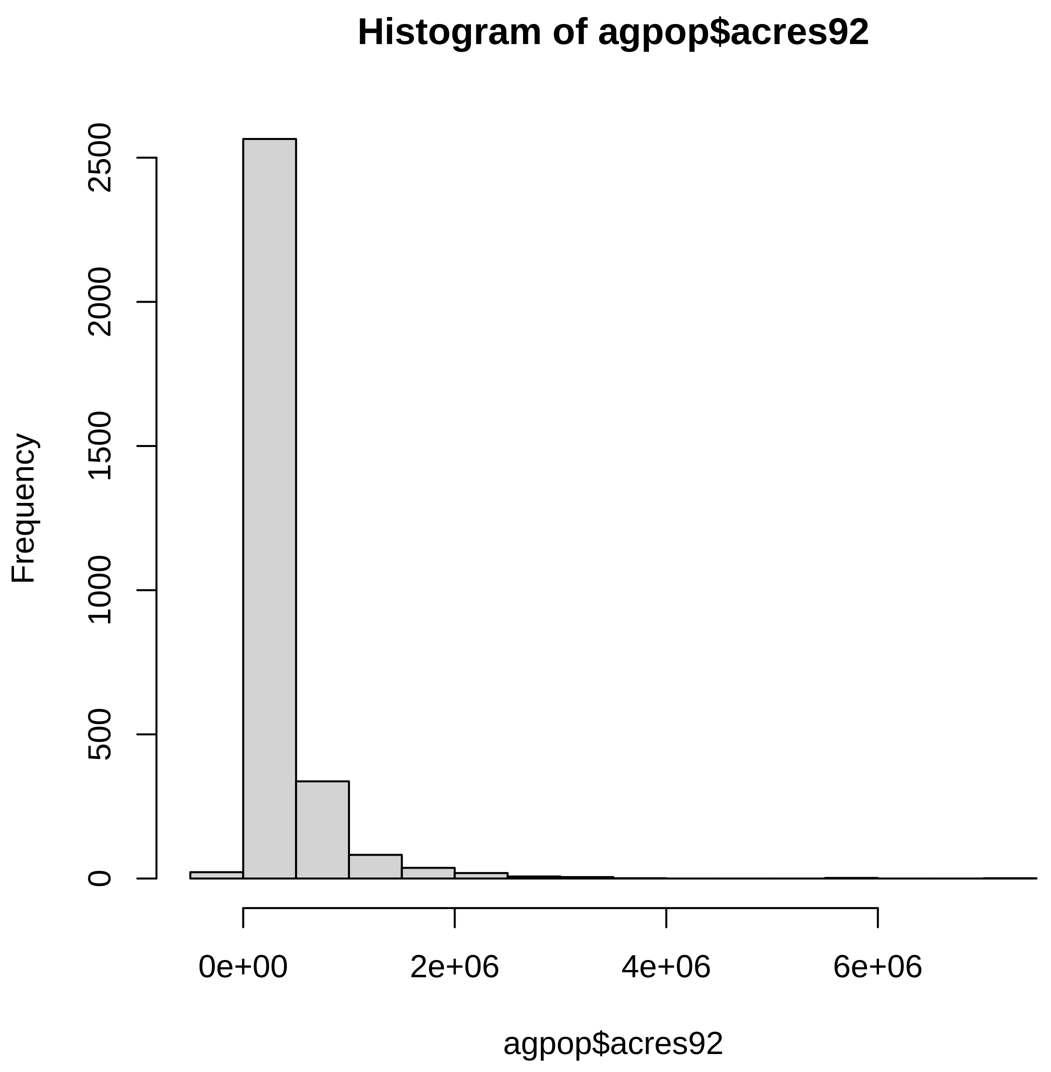
```
## find mean of acres92
mean (agpop $acres92)
```

```
[1] 306677
```

```
## find sd of acres92
sd (agpop $acres92)
```

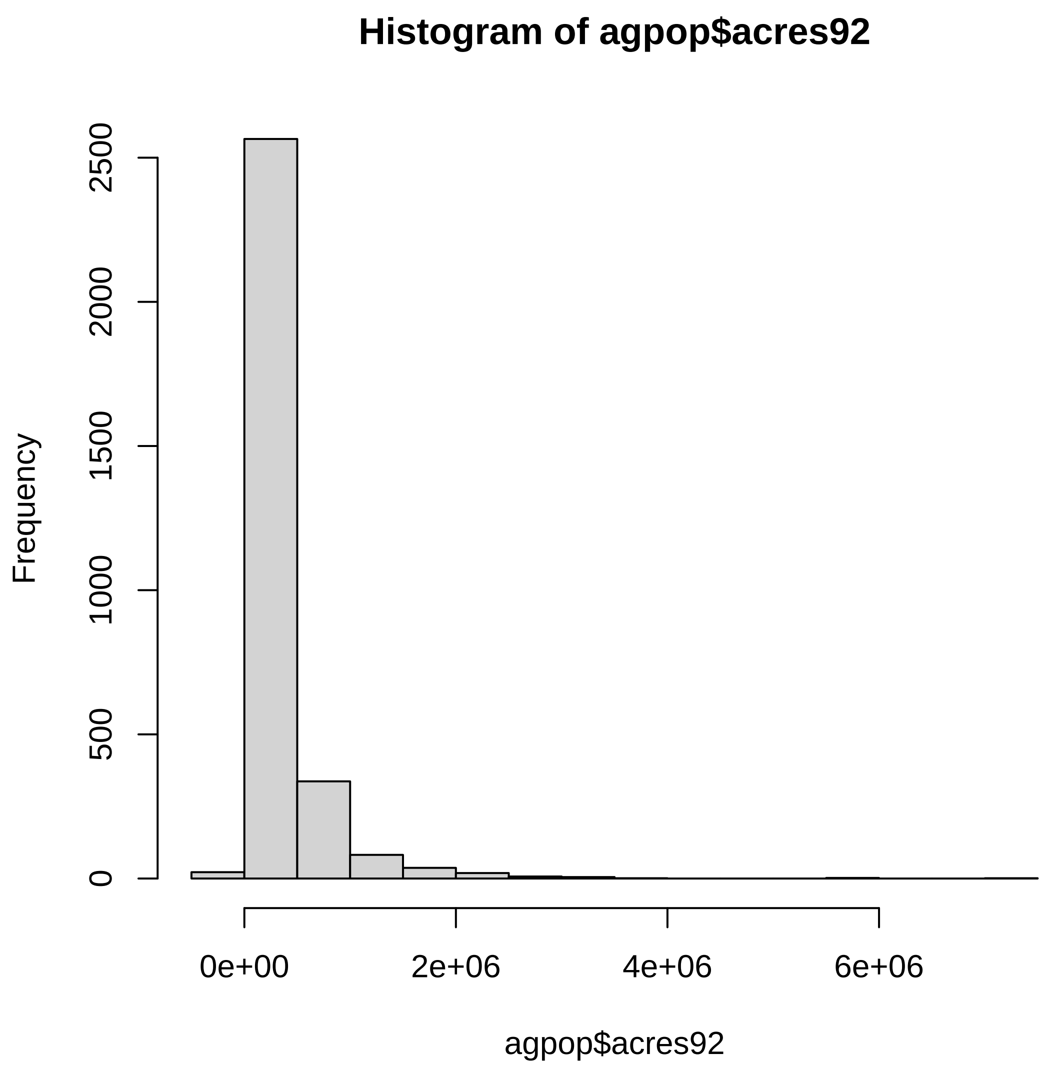
```
[1] 424686.7
```

```
agpop_AK  <- agpop [agpop$state == "AK", ]

agpop_AK <- subset (agpop, state == "AK")

agpop_W <- subset (agpop, region == "W")

agpop_largefarm <- subset (agpop, largef92 > 10)


hist (agpop$acres92)
```

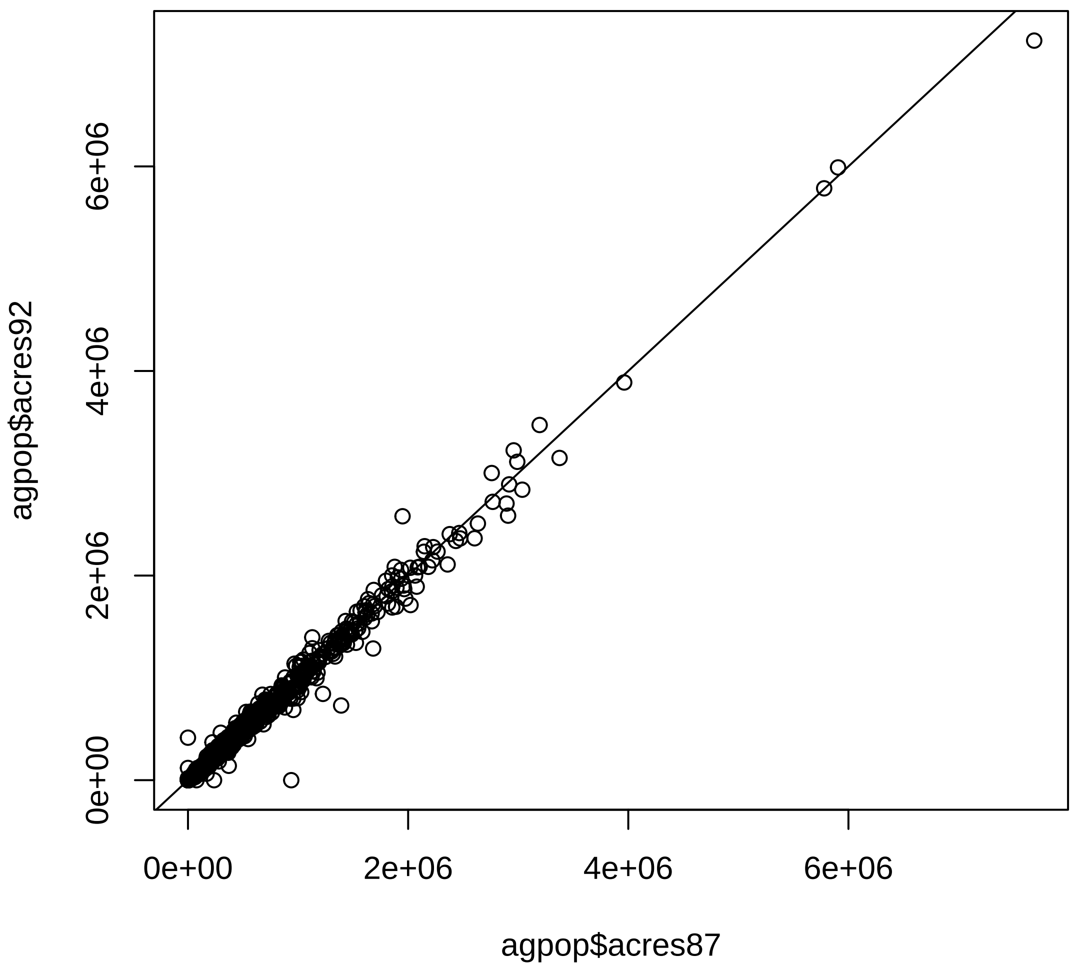


Produce Plots

```r
#pdf ("hist_acres92.pdf") ## use this command and dev.off to save the output to a file
hist (agpop$acres92)
```



```
#dev.off()

#jpeg ("agpop_acres_87v92.jpg")

plot (agpop$acres87, agpop$acres92)
abline (a = 0, b = 1)
```



```
#dev.off()## this is used to close the jpeg file
```

2.3 Create your own function

```
### data is a matrix or data.frame
means_col <- function (data)
{
    n <- ncol (data)
    cmeans <- rep (NA, n)
    for (j in 1:n)
    {
        cmeans[j] <- mean (data[,j])

    }
    cmeans
}

### apply function
means_col (agpop[, 3:13])
```

```
 [1] 306676.97141 313016.37817 320193.69298    625.50357    678.28428
 [6]    728.06238     56.17674     54.86160     52.62248     54.09227
[11]     59.53769
```

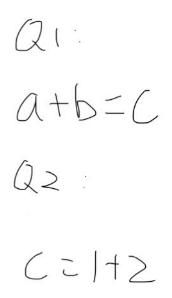
```
### R built-in function
colMeans (agpop[, 3:13])
```

```
     acres92      acres87      acres82      farms92      farms87      farms82
306676.97141 313016.37817 320193.69298    625.50357    678.28428    728.06238
    largef92     largef87     largef82     smallf92     smallf87
    56.17674     54.86160     52.62248     54.09227     59.53769
```

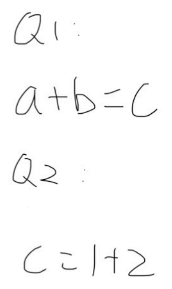
2.4 Include Images Saved in An External File

Using the following R code to include your images saved in an external file.

```
knitr::include_graphics("handwriting.png")
```


Q1:
a+b=c
Q2:
c=1+2

You can hide the above R code by setting “echo=FALSE” for the r chunk. For example, I will include the image once again as follows:



Q1:

$a+b=c$

Q2:

$c=1+2$

Figure 2.1: This is a figure inserted from the file called “handwriting.png”

3 Simple Linear Regression

A Simulation Illustration with R

```r
require("knitr")
knitr::opts_chunk$set(
  comment = "#",
  fig.width = 8,
  fig.height = 6,
  cache = TRUE
)
set.seed(47)

options(sim_rebuild=FALSE)
```

3.1 Overview of Simple Linear Regression

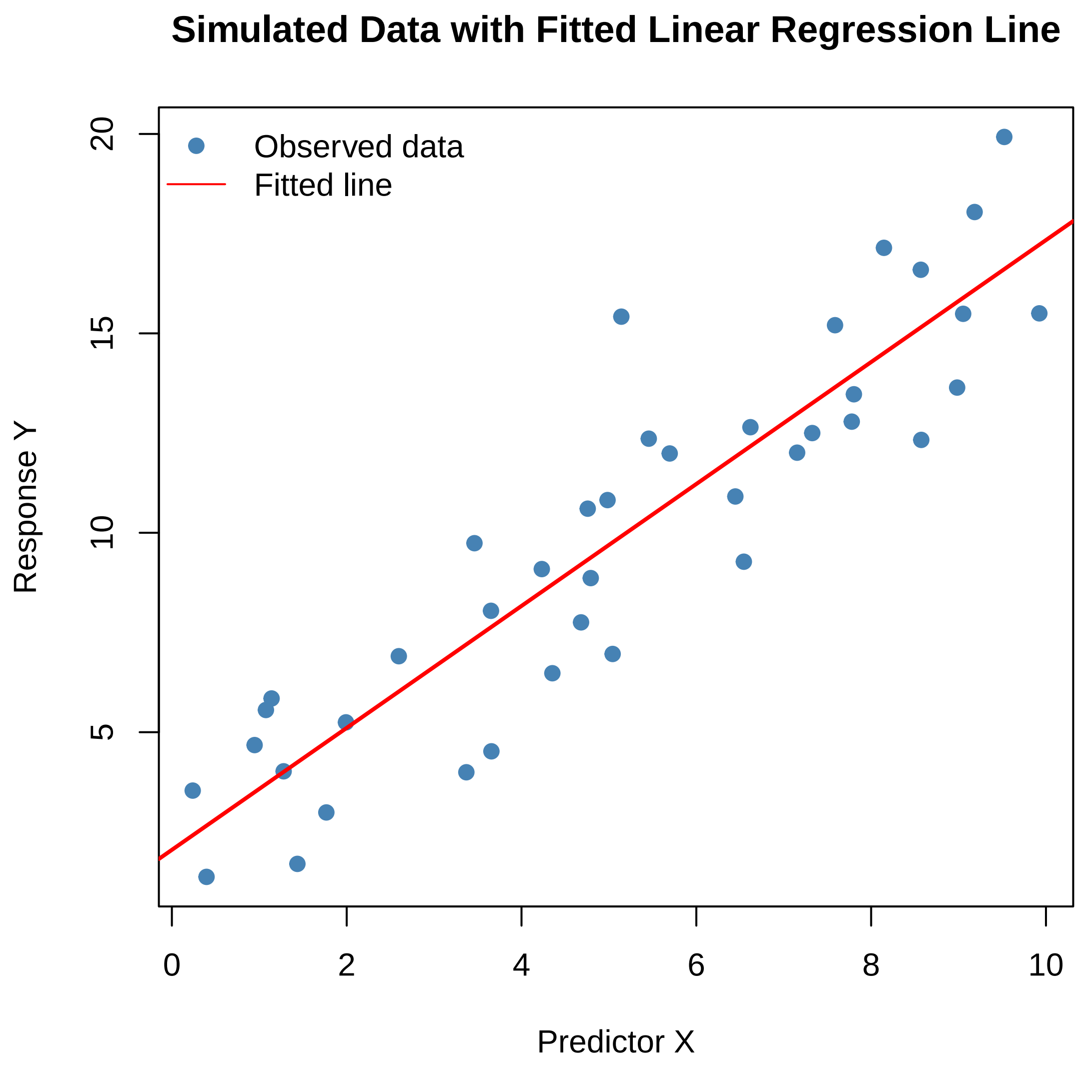
To make the simple linear regression model concrete, let's first visualize a simulated dataset that follows

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i, \qquad \varepsilon_i \sim \mathcal{N}(0, \sigma^2).$$

Here, β_0 is the intercept, β_1 is the slope, and ε_i represents random noise.

```r
set.seed(2025)
n <- 40
beta0 <- 2; beta1 <- 1.5; sigma <- 2
x <- runif(n, 0, 10)
y <- beta0 + beta1 * x + rnorm(n, 0, sigma)
dat <- data.frame(x, y)

fit <- lm(y ~ x, data = dat)

plot(x, y, pch = 19, col = "steelblue",
     xlab = "Predictor X", ylab = "Response Y",
     main = "Simulated Data with Fitted Linear Regression Line")
```

```r
abline(fit, col = "red", lwd = 2)
legend("topleft", legend = c("Observed data", "Fitted line"),
       pch = c(19, NA), lty = c(NA, 1), col = c("steelblue", "red"), bty = "n")
```



The scatterplot shows data points scattered around a line — the red line is the fitted regression model.

3.1.1 Least Squares Estimation

Goal: Find $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize

$$\text{SSE} = \sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2.$$

Solutions:

$$\widehat{\beta}_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} = \frac{S_{xy}}{S_{xx}}, \qquad \widehat{\beta}_0 = \bar{y} - \widehat{\beta}_1 \, \bar{x}.$$

Here

$$S_{xy} = \sum_i (x_i - \bar{x})(y_i - \bar{y}), \qquad S_{xx} = \sum_i (x_i - \bar{x})^2.$$

Shortcut (computational) formulas:

$$S_{xy} = \sum_i x_i y_i - n\,\bar{x}\,\bar{y}, \qquad S_{xx} = \sum_i x_i^2 - n\,\bar{x}^2.$$

Interpretation:
- $\widehat{\beta}_1$ measures the estimated change in Y for each unit increase in X.
- $\widehat{\beta}_0$ represents the fitted value of Y when $X = 0$.

3.1.2 Residual and Sum of Squares Definitions

Let $\widehat{y}_i = \widehat{\beta}_0 + \widehat{\beta}_1 x_i$ and $e_i = y_i - \widehat{y}_i$.

Symbol	Definition	Computing Formula (in terms of S_{xx}, S_{xy}, etc.)
SST	Total Sum of Squares	$\sum_i (y_i - \bar{y})^2 = S_{yy} = \sum_i y_i^2 - n\,\bar{y}^2$
SSR	Regression Sum of Squares	$\sum_i (\widehat{y}_i - \bar{y})^2 = \widehat{\beta}_1^2 S_{xx} = \frac{S_{xy}^2}{S_{xx}}$
SSE	Error (Residual) Sum of Squares	$\sum_i (y_i - \widehat{y}_i)^2 = S_{yy} - \frac{S_{xy}^2}{S_{xx}}$

Identity:

$$\text{SST} = \text{SSR} + \text{SSE}.$$

Here,

$$S_{xx} = \sum_i (x_i - \bar{x})^2 = \sum_i x_i^2 - n\bar{x}^2, \qquad S_{yy} = \sum_i (y_i - \bar{y})^2 = \sum_i y_i^2 - n\bar{y}^2, \qquad S_{xy} = \sum_i (x_i - \bar{x})(y_i - \bar{y}) = \sum_i x_i y_i - n$$

3.1.3 Coefficient of Determination (R^2)

Measures the proportion of total variation in Y explained by X:

$$R^2 = \frac{\text{SSR}}{\text{SST}} = 1 - \frac{\text{SSE}}{\text{SST}}.$$

Interpretation:

- $R^2 = 1$ means perfect linear fit;
- $R^2 = 0$ means the model explains none of the variation.

3.1.4 F-test for Overall Significance

Tests whether X is linearly related to Y.

Hypotheses:

$$H_0 : \beta_1 = 0 \quad \text{vs.} \quad H_A : \beta_1 \neq 0.$$

Test Statistic:

$$F = \frac{\text{MSR}}{\text{MSE}} = \frac{\text{SSR}/1}{\text{SSE}/(n-2)} \sim F_{1,n-2} \quad (H_0).$$

p-value approach for observe F^{obs}:

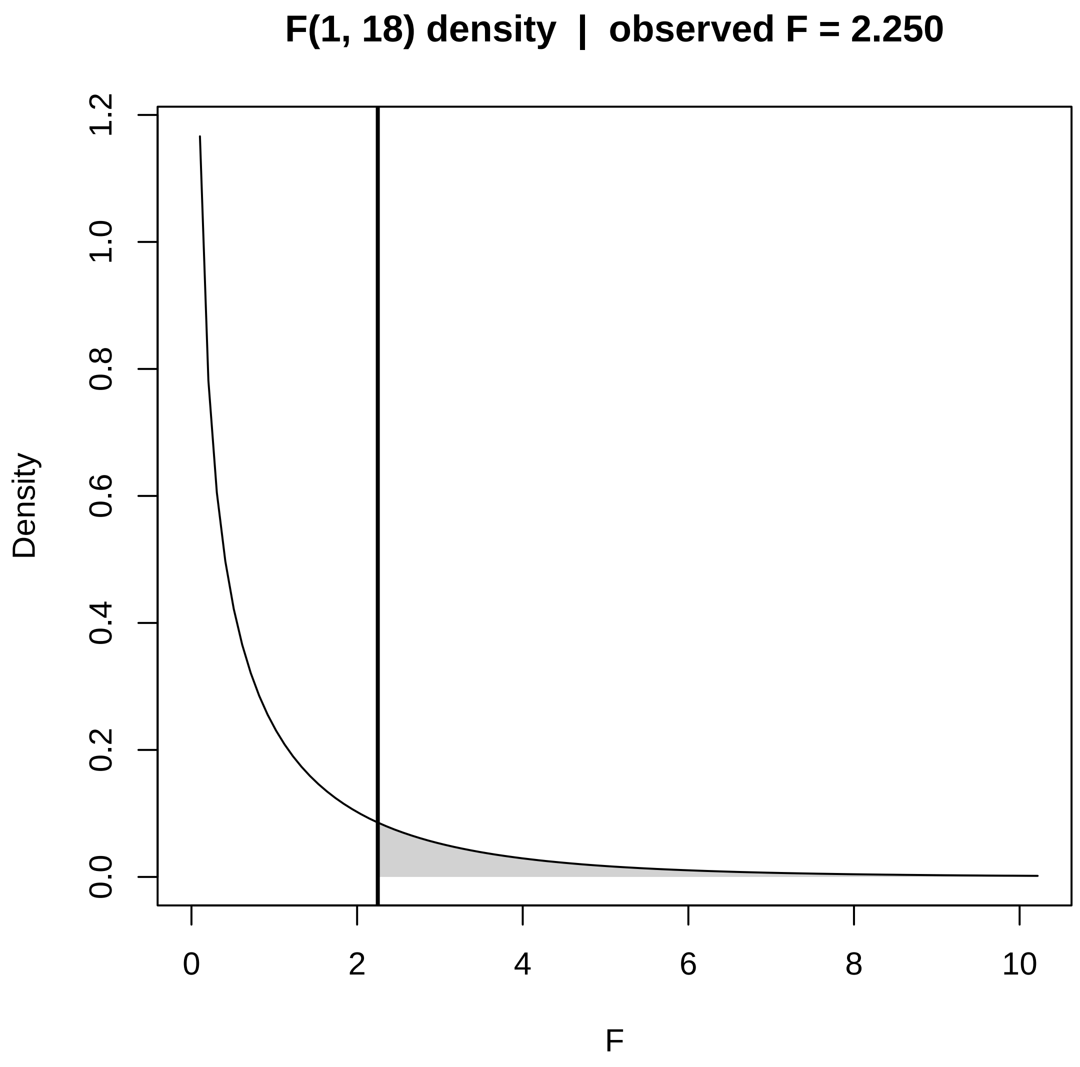
Given the observed statistic F^{obs} with $(1,\, n-2)$ df,

$$p-\text{value} \;=\; \Pr\big(F_{1,\,n-2} \geq F^{\text{obs}}\big) \;=\; \text{pf}\big(F^{\text{obs}},\, 1,\, n-2,\, \text{lower.tail} = \text{FALSE}\big).$$

```r
## -- Inputs (provide these from your analysis context) -------------------------
## n   <- ...    # sample size
## SSR <- ...    # regression sum of squares
## SSE <- ...    # error sum of squares
n   <- 20
SSR <- 5
SSE <- 40





df1  <- 1
df2  <- n - 2
Fobs <- (SSR/df1) / (SSE/df2)         # observed F
pval <- pf(Fobs, df1 = df1, df2 = df2, lower.tail = FALSE)
pval
```

```
[1] 0.1509505
```

```
## -- Plot F density and shade the p-value tail (with proper annotations) -------
xmax <- max(qf(0.995, df1, df2), Fobs * 1.2)  # extra space for labels
peak <- max(df(seq(0, xmax, length.out = 500), df1, df2))

## Density curve
curve(df(x, df1, df2), from = 0, to = xmax,
      xlab = "F", ylab = "Density",
      main = sprintf("F(%d, %d) density  |  observed F = %.3f", df1, df2, Fobs))

## Shade right tail (p-value region)
xs <- seq(Fobs, xmax, length.out = 300)
ys <- df(xs, df1, df2)
polygon(c(Fobs, xs, xmax), c(0, ys, 0),
        col = rgb(0, 0, 0, 0.18), border = NA)

## Vertical line at Fobs (optional visual aid)
abline(v = Fobs, lwd = 2)

## ---- Annotation for F^obs pointing to the x-axis value (Fobs, 0) -------------
x_txt_F <- Fobs + 0.06 * xmax
y_txt_F <- 0.45 * peak
arrows(x0 = x_txt_F, y0 = y_txt_F, x1 = Fobs, y1 = 0,
       length = 0.08, lwd = 1.5)
text(x_txt_F, y_txt_F,
     labels = bquote(F^{obs} == .(format(Fobs, digits = 3))),
     pos = 4)

## ---- Annotation for p-value pointing into the shaded tail --------------------
x_tip_p <- (Fobs + xmax) / 1.7
y_tip_p <- df(x_tip_p, df1, df2)
x_txt_p <- Fobs + 0.08 * xmax
y_txt_p <- 0.80 * peak
arrows(x0 = x_txt_p, y0 = y_txt_p, x1 = x_tip_p, y1 = y_tip_p,
       length = 0.08, lwd = 1.5)
text(x_txt_p, y_txt_p,
     labels = bquote(p == .(format(pval, digits = 4, scientific = TRUE))),
     pos = 4)
```



3.1.5 t-test for the Slope β_1

Equivalent to the F-test in simple regression since $t^2 = F$.

Formula:

$$t = \frac{\hat{\beta}_1}{\mathrm{SE}(\hat{\beta}_1)}, \qquad \mathrm{SE}(\hat{\beta}_1) = \sqrt{\frac{\hat{\sigma}^2}{\sum_i (x_i - \bar{x})^2}}, \qquad \hat{\sigma}^2 = \frac{\mathrm{SSE}}{n-2}.$$

Distribution:

$$t \sim t_{n-2} \quad (H_0 : \beta_1 = 0).$$

3.1.6 Prediction for a New Case x_0

Predicted mean response:

$$\hat{y}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0.$$

95% Confidence interval for mean response:

$$\hat{y}(x_0) \pm t_{1-\alpha/2,,n-2}, \hat{\sigma}, \sqrt{\frac{1}{n} + \frac{(x_0 - \bar{x})^2}{\sum_i (x_i - \bar{x})^2}}.$$

95% Prediction interval for a new observation:

$$\hat{y}(x_0) \pm t_{1-\alpha/2,,n-2}, \hat{\sigma}, \sqrt{1 + \frac{1}{n} + \frac{(x_0 - \bar{x})^2}{\sum_i (x_i - \bar{x})^2}}.$$

Summary Cheat Sheet

Concept	Key Formula
Model	$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$
LS Estimates	$\hat{\beta}_1 = S_{xy}/S_{xx}$, $\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$
Decomposition	$\text{SST} = \text{SSR} + \text{SSE}$
R^2	$R^2 = 1 - \text{SSE}/\text{SST}$
F-test	$F = (\text{SSR}/1)/(\text{SSE}/(n-2))$
t-test	$t = \hat{\beta}_1 / \text{SE}(\hat{\beta}_1)$
Prediction	$\hat{y}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0$

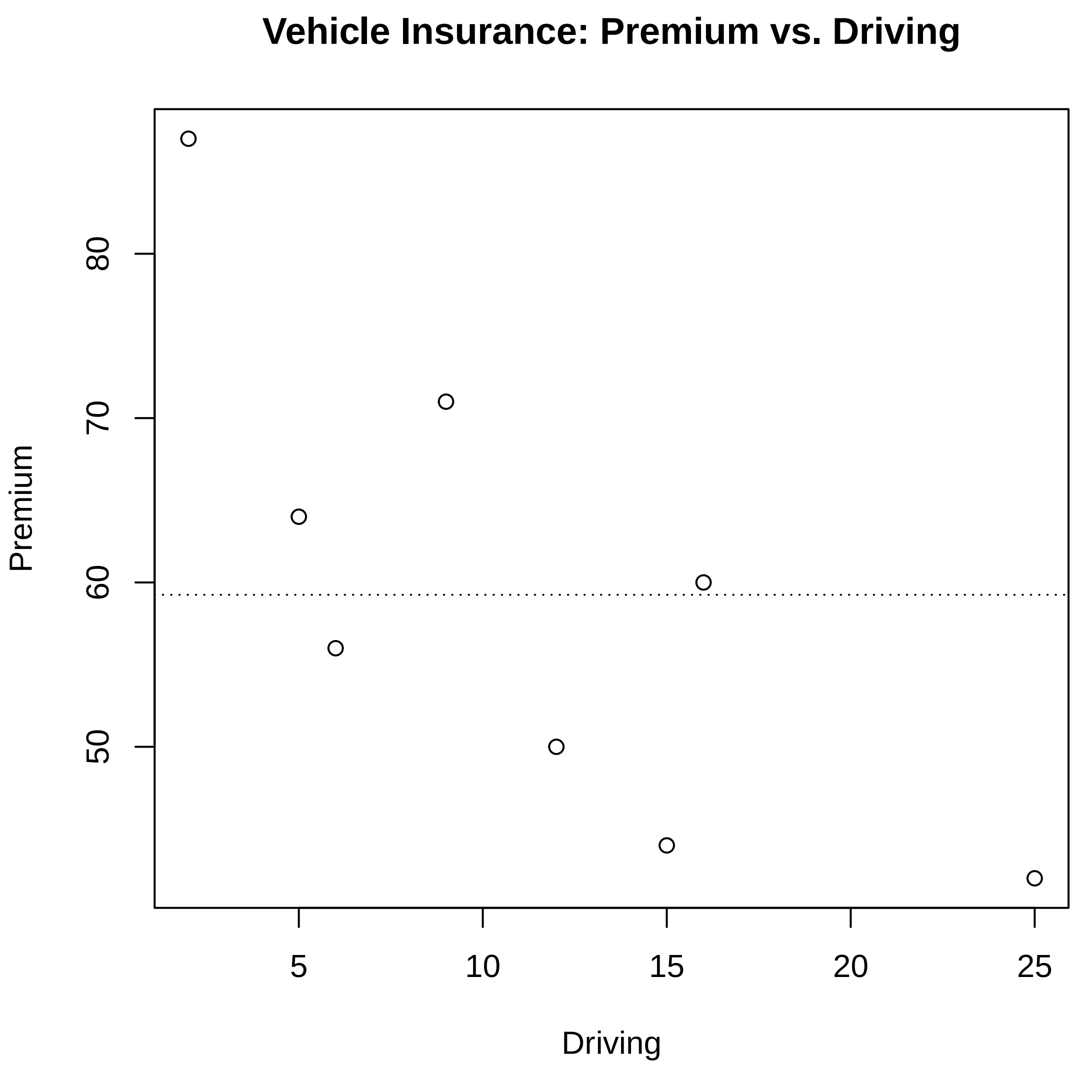
3.2 Example 1: Vehicle Insurance Premium (warm-up)

We examine premiums y vs. driving amount x. The scatterplot hints at a **downward** trend.

3.2.1 Input data

```r
issu <- data.frame(
  driving = c(5, 2, 12, 9, 15, 6, 25, 16),
  premium = c(64, 87, 50, 71, 44, 56, 42, 60)
)
```

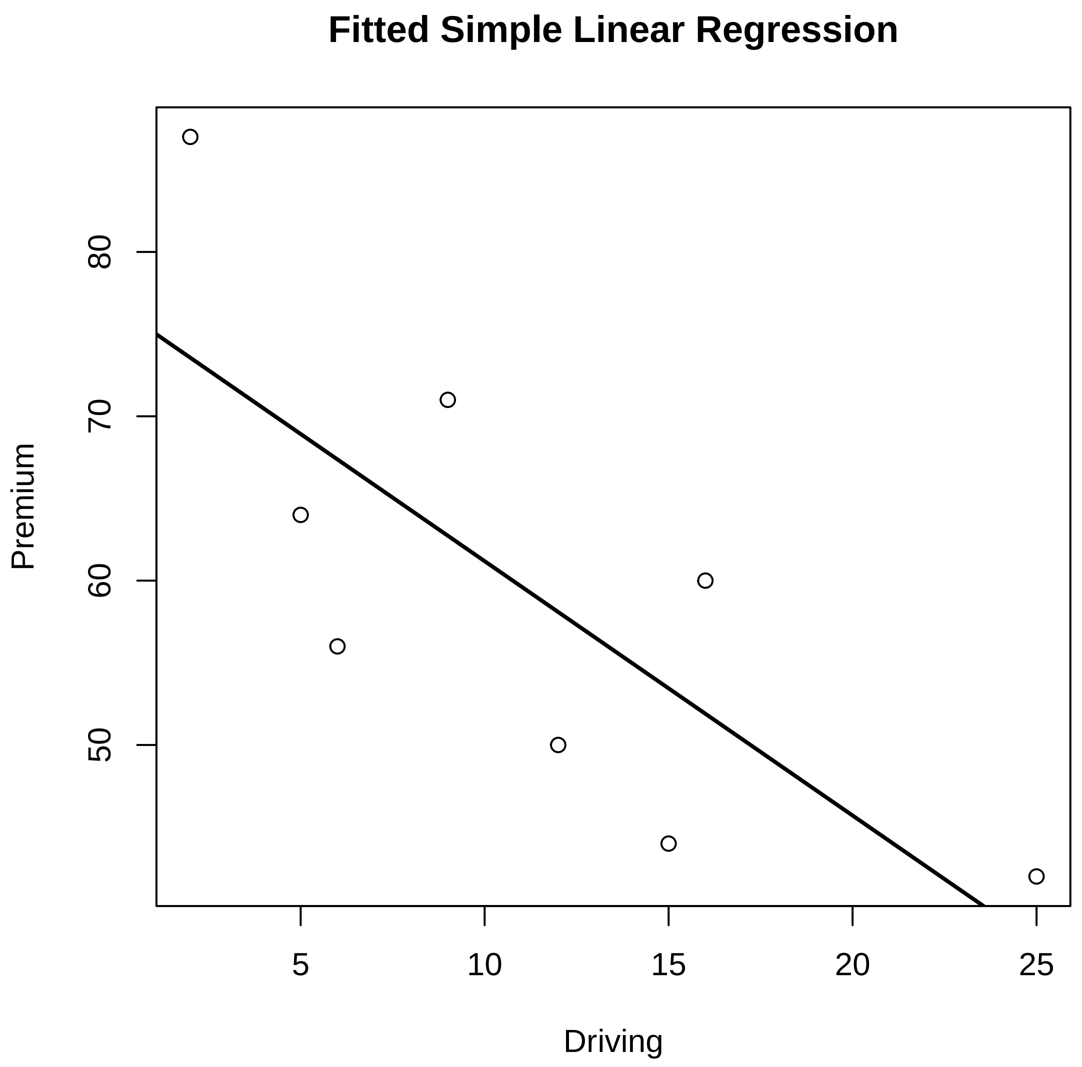
```r
y <- issu$premium
x <- issu$driving
xbar <- mean(x); ybar <- mean(y); n <- length(y)

plot(x, y, xlab = "Driving", ylab = "Premium",
     main = "Vehicle Insurance: Premium vs. Driving")
abline(h = ybar, lty = 3)
```



Narrative. The horizontal line at $\bar{y}$ represents the intercept-only model. Any fitted line that tilts away from this must earn its keep by reducing residual variation enough to offset the loss of one degree of freedom.

3.2.2 Estimating regression coefficients

```r
fit.issu <- lm(y ~ x)
plot(x, y, xlab = "Driving", ylab = "Premium",
     main = "Fitted Simple Linear Regression")
abline(fit.issu, lwd = 2)
```



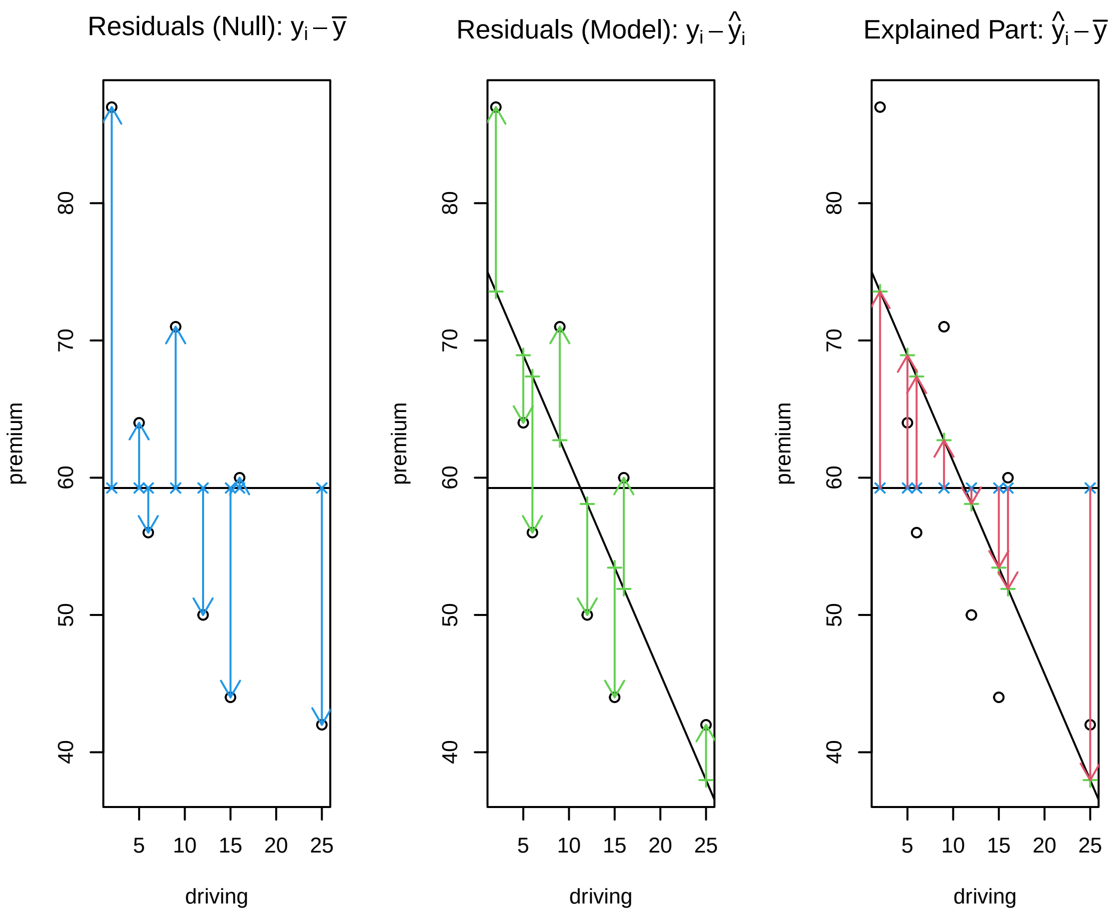
The slope estimate $\hat{\beta}_1$ captures the **marginal change in premium per unit of driving** (units of y per unit of x). Inference on β_1 tells us whether the pattern rises above noise.

3.2.3 Residuals and fitted values (geometry picture)

Let $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ and $\tilde{y}_i = \bar{y}$. Residuals are $e_i = y_i - \hat{y}_i$ (model) and $y_i - \bar{y}$ (null). Visualizing all three clarifies the ANOVA identity.

```
beta0 <- coef(fit.issu)[1]
beta1 <- coef(fit.issu)[2]
fitted1 <- beta0 + beta1 * x
fitted0 <- rep(ybar, n)
residual1 <- y - fitted1
residual0 <- y - fitted0

data.frame(y, fitted0, residual0, fitted1, residual1,
           diff.fitted = fitted1 - fitted0)
```

```
   y fitted0 residual0  fitted1  residual1 diff.fitted
1 64   59.25      4.75 68.92243  -4.922425    9.672425
2 87   59.25     27.75 73.56519  13.434811   14.315189
3 50   59.25     -9.25 58.08931  -8.089309   -1.160691
4 71   59.25     11.75 62.73207   8.267927    3.482073
5 44   59.25    -15.25 53.44654  -9.446545   -5.803455
6 56   59.25     -3.25 67.37484 -11.374837    8.124837
7 42   59.25    -17.25 37.97066   4.029335  -21.279335
8 60   59.25      0.75 51.89896   8.101043   -7.351043
```



3.2.4 SST, SSR, SSE and their meanings

- $\mathrm{SST} = \sum (y_i - \bar{y})^2$ quantifies **total** variability around the mean.
- $\mathrm{SSR} = \sum (\hat{y}_i - \bar{y})^2$ is the part **explained by** x.
- $\mathrm{SSE} = \sum (y_i - \hat{y}_i)^2$ is the **leftover** (unexplained) variability.

```
SST <- sum((y - fitted0)^2); SST
```

```
[1] 1557.5
```

```
SSE <- sum((y - fitted1)^2); SSE
```

```
[1] 639.0065
```

```
SSR <- SST - SSE; SSR
```

```
[1] 918.4935
```

Direct check: SSR $= \sum(\hat{y}_i - \bar{y})^2$.

```
sum((fitted1 - fitted0)^2)
```

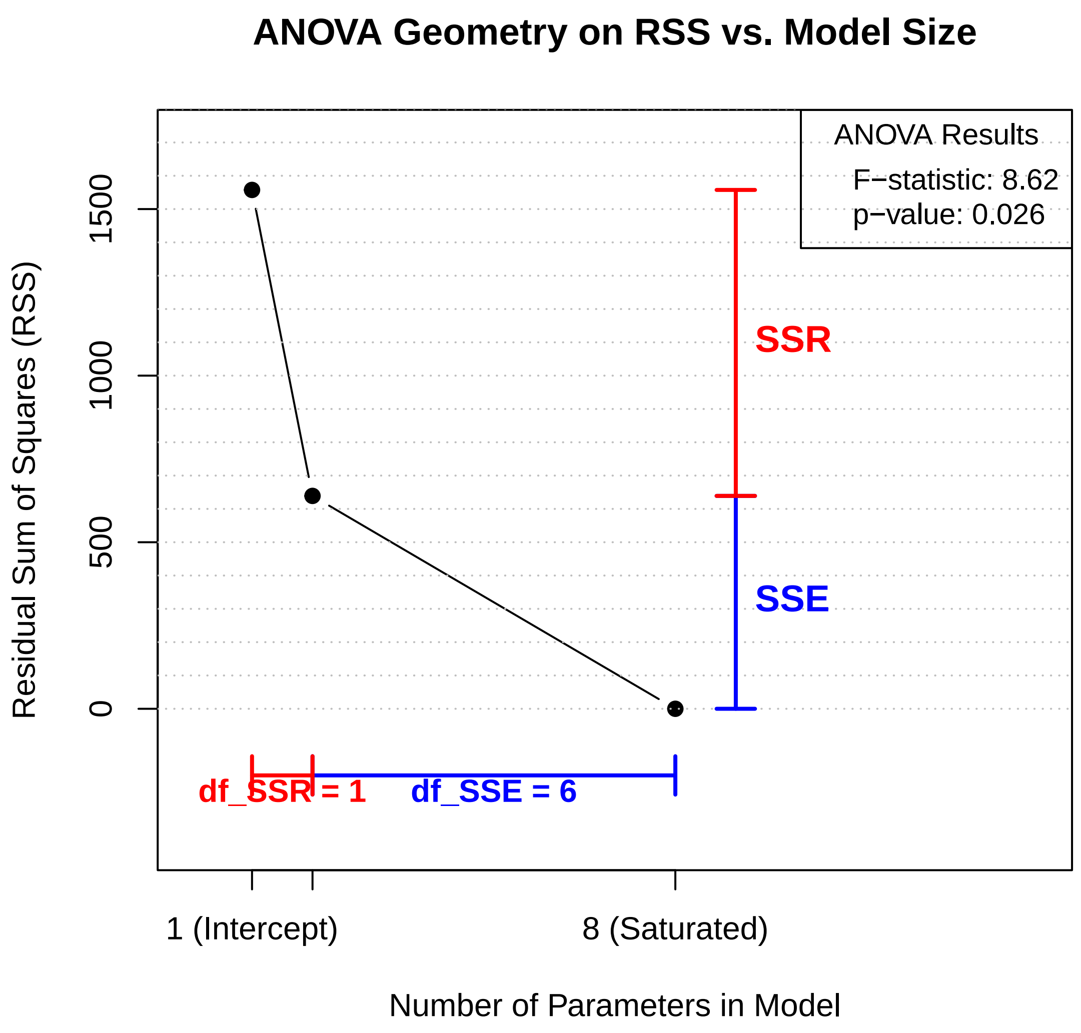
```
[1] 918.4935
```

3.2.5 Visual ANOVA on an RSS plot

We place the **residual sum of squares** against model dimension to show the trade-off between fit and df.

```
## Recompute cleanly
SST <- sum((y - mean(y))^2)
SSE <- sum(resid(fit.issu)^2)
SSR <- SST - SSE
df_SSR <- 1
df_SSE <- n - 2

par(mar = c(6, 4, 4, 2) + 0.1)
plot(c(1, 2, n), c(SST, SSE, 0), type = "b", pch = 19,
     xlab = "Number of Parameters in Model",
     ylab = "Residual Sum of Squares (RSS)",
     main = "ANOVA Geometry on RSS vs. Model Size",
     xlim = c(0, 14), ylim = c(-400, SST * 1.1), xaxt = "n")
axis(1, at = c(1, 2, n), labels = c("1 (Intercept)", "2 (+Slope)", paste(n, "(Saturated)")))
abline(h = seq(0, 2000, by = 100), lty = 3, col = "grey")

par(xpd = TRUE)
arrows(9, 0, 9, SSE, col = "blue", code = 3, angle = 90, length = 0.1, lwd = 2)
text(9, SSE/2, "SSE", col = "blue", pos = 4, font = 2, cex = 1.2)

arrows(9, SSE, 9, SST, col = "red", code = 3, angle = 90, length = 0.1, lwd = 2)
text(9, (SST + SSE)/2, "SSR", col = "red", pos = 4, font = 2, cex = 1.2)

arrows(2, -200, n, -200, col = "blue", code = 3, angle = 90, length = 0.1, lwd = 2)
text((2 + n)/2, -250, paste("df_SSE =", df_SSE), col = "blue", font = 2)

arrows(1, -200, 2, -200, col = "red", code = 3, angle = 90, length = 0.1, lwd = 2)
text(1.5, -250, paste("df_SSR =", df_SSR), col = "red", font = 2)
par(xpd = FALSE)

f_value <- (SSR/df_SSR) / (SSE/df_SSE)
```

```
p_value <- pf(f_value, df1 = df_SSR, df2 = df_SSE, lower.tail = FALSE)
legend("topright",
       legend = c(sprintf("F-statistic: %.2f", f_value),
                  sprintf("p-value: %.3f", p_value)),
       title = "ANOVA Results", bty = "o", cex = 0.9)
```



3.2.6 R^2, F and a compact ANOVA table

```
R2 <- SSR / SST; R2
```

```
[1] 0.5897229
```

```r
f  <- (SSR/1) / (SSE/(n-2)); f
```

```
[1] 8.624264
```

```r
pvf <- pf(f, df1 = 1, df2 = n-2, lower.tail = FALSE); pvf
```

```
[1] 0.0260588
```

```r
Ftable <- data.frame(
  Source = c("Regression", "Error"),
  df     = c(1, n - 2),
  SS     = c(SSR, SSE),
  MS     = c(SSR/1, SSE/(n-2)),
  F      = c(f, NA),
  pvalue = c(pvf, NA),
  R2part = c(SSR, SSE) / SST
)
Ftable
```

```
      Source df       SS       MS        F    pvalue    R2part
1 Regression  1 918.4935 918.4935 8.624264 0.0260588 0.5897229
2      Error  6 639.0065 106.5011       NA        NA 0.4102771
```

A call to anova() reproduces the same test:

```r
anova(fit.issu)
```

```
Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value  Pr(>F)
x          1 918.49  918.49  8.6243 0.02606 *
Residuals  6 639.01  106.50
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.2.7 Sampling distributions via animation

Under $H_0 : \beta_1 = 0$, F follows $F_{1,n-2}$. Under H_A, the distribution shifts right (noncentral F).

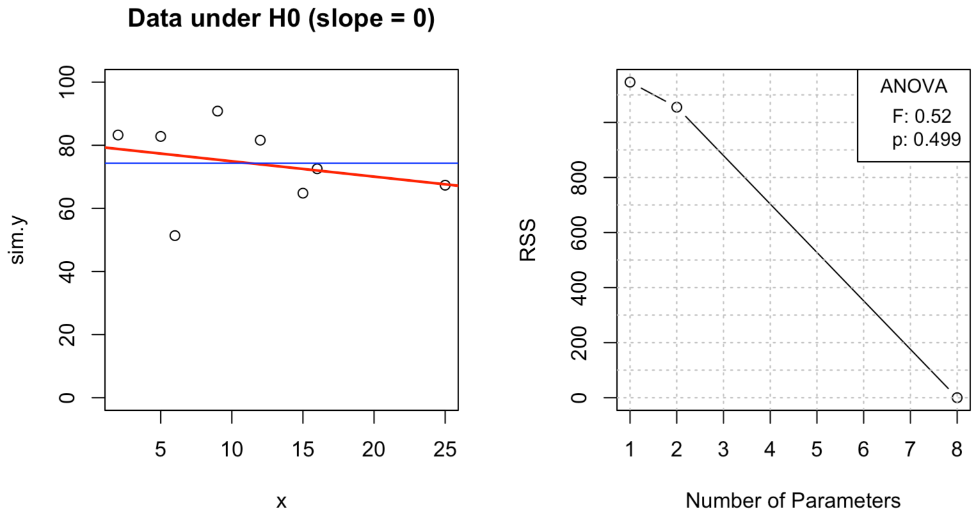


Figure 3.1: Simulation under H0: animated GIF (HTML) and static PNG (PDF).

3.2.7.1 Null world (H_0 true)

3.2.7.2 Alternative world (H_1 true)

3.3 Example 2: Oxygen Purity Data

We model oxygen purity y as a function of hydrocarbon level x and report both **mean response** and **prediction** uncertainty.

3.3.1 Data

```
x <- c(0.99, 1.02, 1.15, 1.29, 1.46, 1.36, 0.87, 1.23, 1.55, 1.40, 1.19,
       1.15, 0.98, 1.01, 1.11, 1.20, 1.26, 1.32, 1.43, 0.95)
y <- c(90.01, 89.05, 91.43, 93.74, 96.73, 94.45, 87.59, 91.77, 99.42, 93.65,
       93.54, 92.52, 90.56, 89.54, 89.85, 90.39, 93.25, 93.41, 94.98, 87.33)
n <- length(x); n
```

```
[1] 20
```

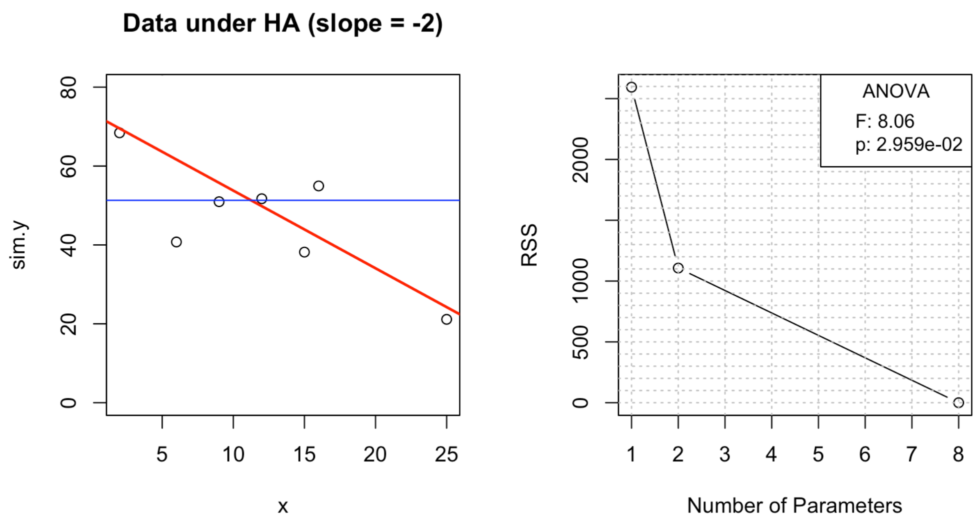


Figure 3.2: Simulation under HA (slope = -2): animated GIF for HTML, static PNG for PDF.

```r
purity.data <- data.frame(x = x, y = y)
head(purity.data)
```

```
     x     y
1 0.99 90.01
2 1.02 89.05
3 1.15 91.43
4 1.29 93.74
5 1.46 96.73
6 1.36 94.45
```

3.3.2 Fit and quick summary

```r
fit <- lm(y ~ x, data = purity.data)
summary(fit)
```

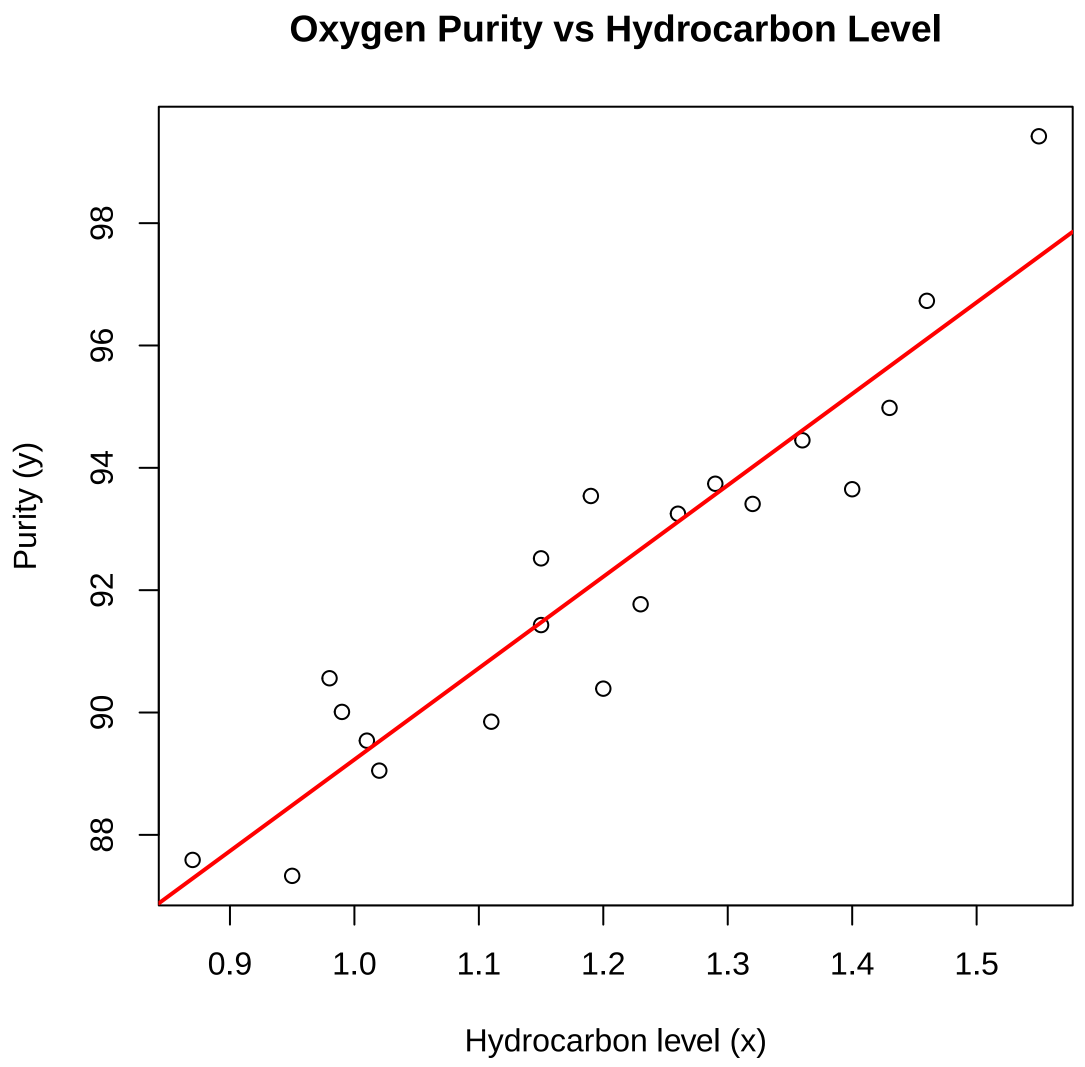
```

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

Residuals:
     Min       1Q   Median       3Q      Max
```

```
-1.83029 -0.73334  0.04497  0.69969  1.96809

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   74.283      1.593   46.62  < 2e-16 ***
x             14.947      1.317   11.35 1.23e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.087 on 18 degrees of freedom
Multiple R-squared:  0.8774,    Adjusted R-squared:  0.8706
F-statistic: 128.9 on 1 and 18 DF,  p-value: 1.227e-09
```

Interpretation. The slope's sign gives the direction of association; its t test (or F with 1 df) assesses evidence for a trend. Look at $\widehat{\sigma}$ for noise scale and R^2 for variance explained.

3.3.3 Scatter with fitted line

```r
plot(purity.data$x, purity.data$y,
     xlab = "Hydrocarbon level (x)", ylab = "Purity (y)",
     main = "Oxygen Purity vs Hydrocarbon Level")
abline(fit, col = "red", lwd = 2)
```



3.3.4 Coefficient CIs and ANOVA

```
confint(fit, level = 0.95)
```

```
                 2.5 %   97.5 %
(Intercept) 70.93555 77.63108
x           12.18107 17.71389
```

```
anova(fit)
```

```
Analysis of Variance Table
```

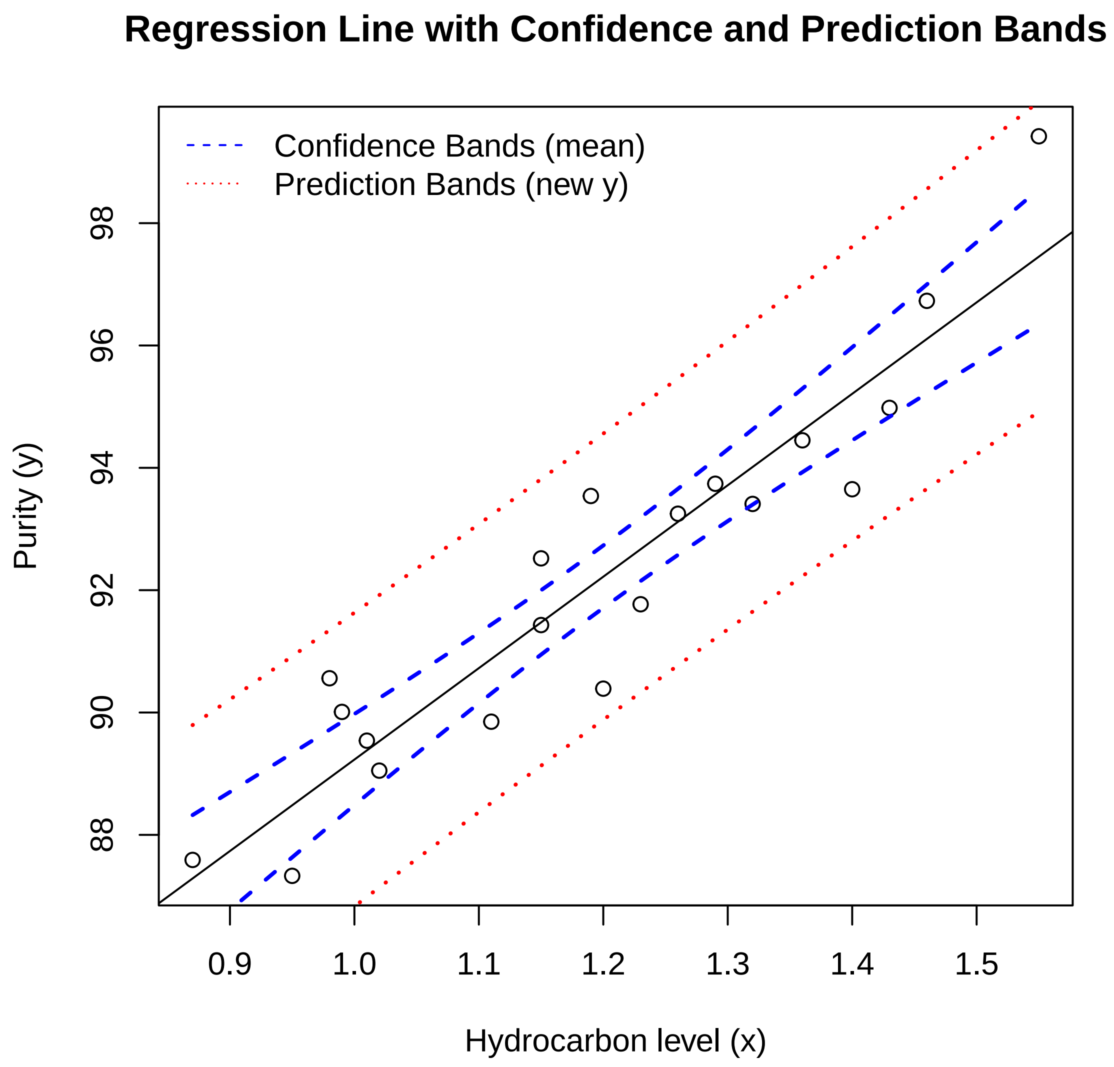
```
Response: y
          Df Sum Sq Mean Sq F value    Pr(>F)
x          1 152.13 152.127  128.86 1.227e-09 ***
Residuals 18  21.25   1.181
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.3.5 Mean-response and prediction bands

The **mean-response CI** narrows near $\bar{x}$ and widens at the extremes; the **prediction band** is wider by the irreducible noise term.

```r
x0 <- seq(min(purity.data$x), max(purity.data$x), length = 50)
newdata <- data.frame(x = x0)

est.mean <- predict(fit, newdata = newdata, interval = "confidence", level = 0.95)
pred.new <- predict(fit, newdata = newdata, interval = "prediction", level = 0.95)
```

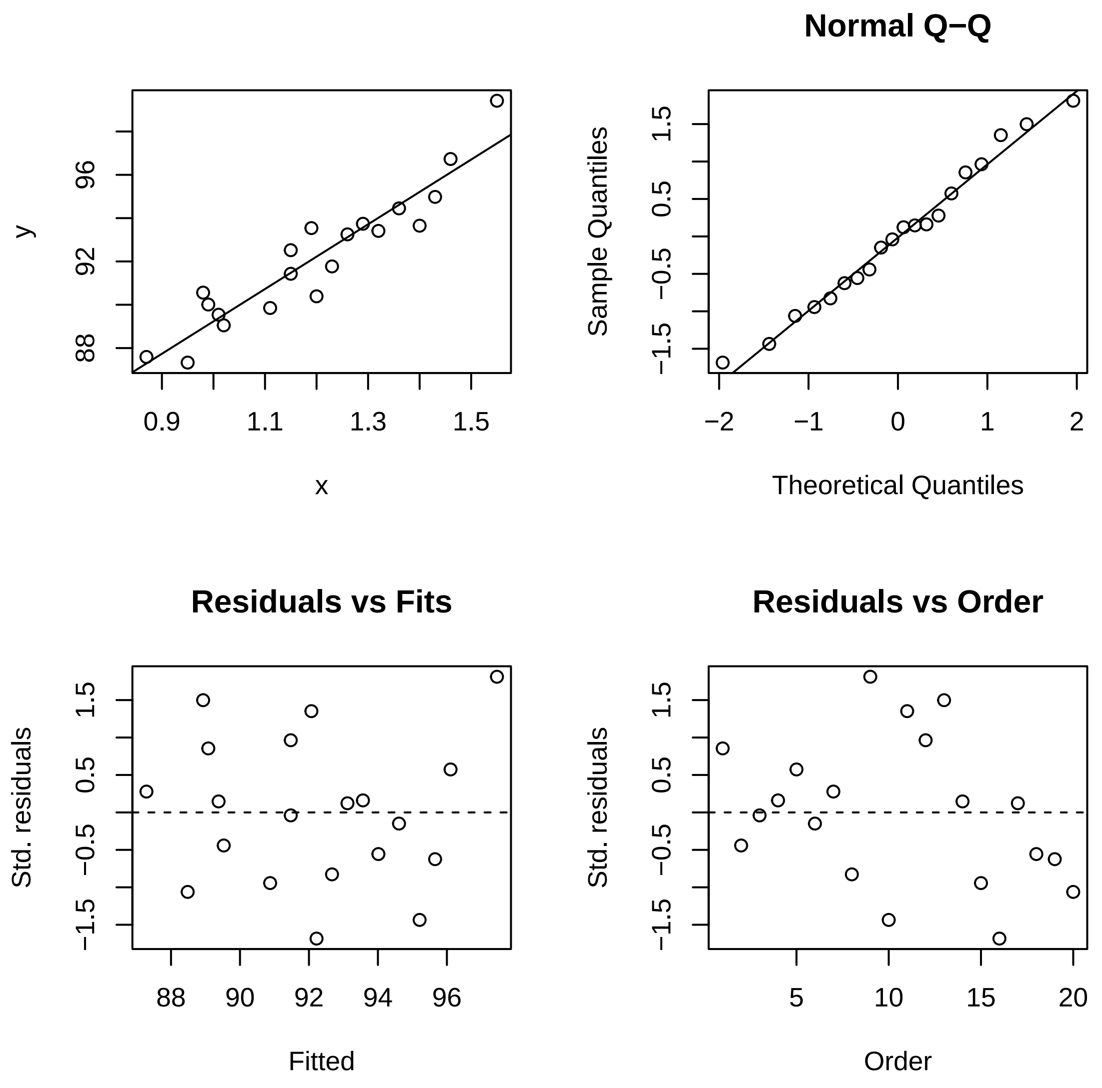
```r
plot(purity.data$x, purity.data$y,
     xlab = "Hydrocarbon level (x)", ylab = "Purity (y)",
     main = "Regression Line with Confidence and Prediction Bands")
abline(fit)
matlines(x0, est.mean[, 2:3], col = "blue", lty = 2, lwd = 2)
matlines(x0, pred.new[, 2:3], col = "red",  lty = 3, lwd = 2)
legend("topleft", c("Confidence Bands (mean)", "Prediction Bands (new y)"),
       col = c("blue", "red"), lty = 2:3, bty = "n")
```



3.3.6 Residual diagnostics (assumptions check)

We look for **no pattern** in residuals vs. fits and **approximate straightness** in the Q–Q plot.

```
pred <- fitted.values(fit)
e <- resid(fit)
d <- e / summary(fit)$sigma

par(mfrow = c(2,2))
plot(purity.data$x, purity.data$y, xlab = "x", ylab = "y"); abline(fit)
qqnorm(d, main = "Normal Q-Q"); qqline(d)
plot(pred, d, xlab = "Fitted", ylab = "Std. residuals", main = "Residuals vs Fits"); abline(h
plot(1:n, d, xlab = "Order", ylab = "Std. residuals", main = "Residuals vs Order"); abline(h =
```



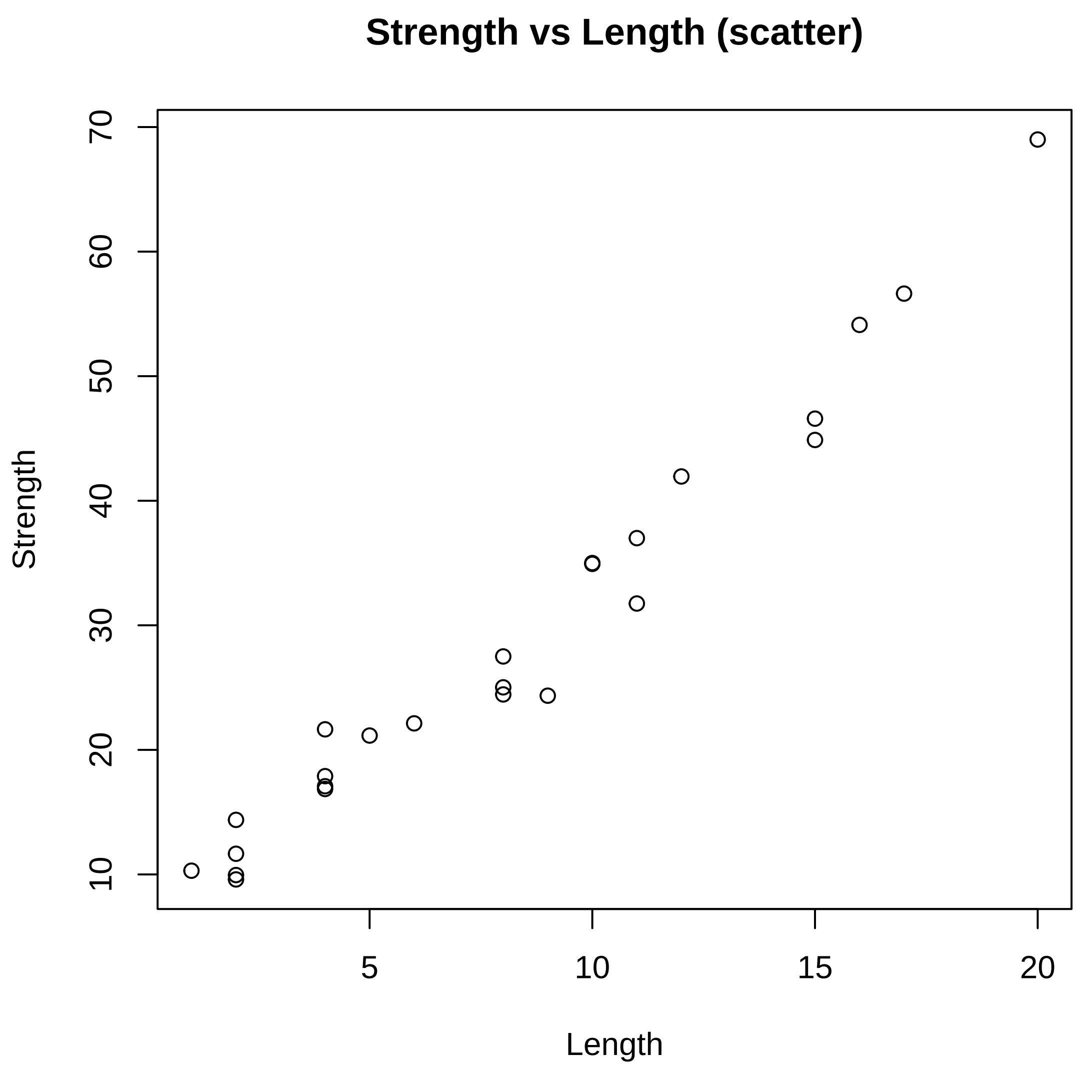
```r
par(mfrow = c(1,1))
```

3.4 Correlation analysis (for comparison, not causation)

Correlation summarizes linear association without fitting a line or making model assumptions.

3.4.1 Data and scatter

```
strength <- c(9.95,24.45,31.75,35.00,25.02,16.86,14.38,9.60,24.35,
              27.50,17.08,37.00,41.95,11.66,21.65,17.89,69.00,10.30,
              34.93,46.59,44.88,54.12,56.63,22.13,21.15)
length <- c(2,8,11,10,8,4,2,2,9,8,4,11,12,2,4,4,20,1,10,
            15,15,16,17,6,5)
plot(length, strength, xlab = "Length", ylab = "Strength",
     main = "Strength vs Length (scatter)")
```



3.4.2 Pearson correlation and test

```
cor(strength, length)
```

```
[1] 0.9818118
```

```r
cor.test(strength, length)
```

```

	Pearson's product-moment correlation

data:  strength and length
t = 24.801, df = 23, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.9585414 0.9920735
sample estimates:
      cor 
0.9818118 
```

Note. A large $|r|$ and small p indicate linear association; regression further quantifies the slope and supports prediction, with diagnostics to check assumptions.

3.5 What to report (checklist)

- Estimated line $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$ with units.
- t/F test for slope, p-value and CI for β_1.
- R^2 and $\hat{\sigma}$ (RMSE) for fit quality.
- Mean-response and prediction intervals at substantively relevant x_0.
- Residual diagnostics and any remedies (transformations, robust methods) if needed.

4 Multiple Linear Regression

4.1 An Example: Wire Bond Strength Dataset

4.1.1 Loading Data and Visualization

Note: You must change the file paths in the `read.csv()` functions below to match the location of the files on your computer (for example `C:\\Users\\<YourUsername>\\Documents` on Windows).

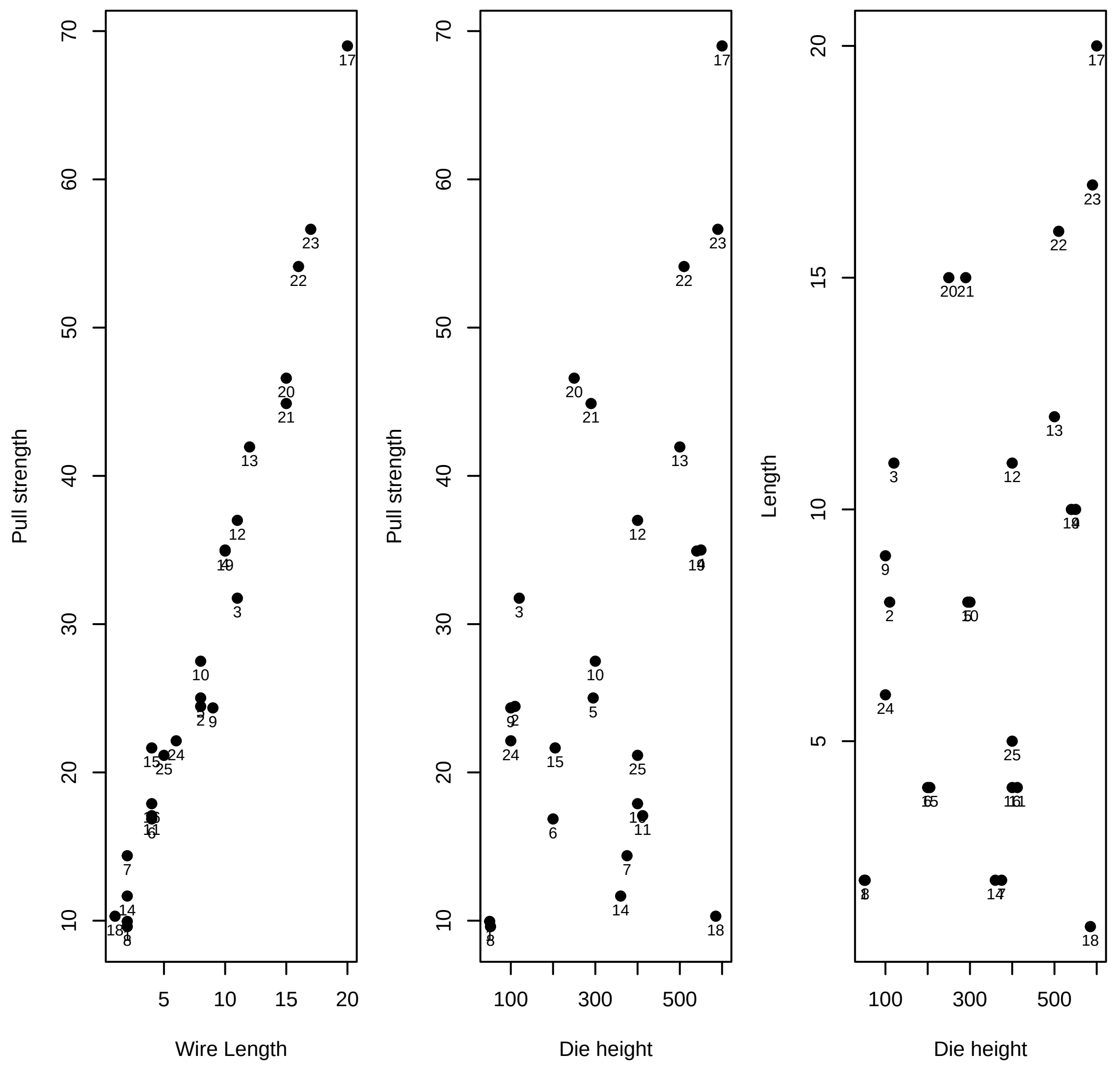
```
## Read data. Change the path as necessary.
## Example: bond.data <- read.csv("wire-bond.csv")
bond.data <- read.csv("wire-bond.csv")

## This will now be automatically rendered as a paged table
bond.data
```

```
   strength length height
1      9.95      2     50
2     24.45      8    110
3     31.75     11    120
4     35.00     10    550
5     25.02      8    295
6     16.86      4    200
7     14.38      2    375
8      9.60      2     52
9     24.35      9    100
10    27.50      8    300
11    17.08      4    412
12    37.00     11    400
13    41.95     12    500
14    11.66      2    360
15    21.65      4    205
16    17.89      4    400
17    69.00     20    600
18    10.30      1    585
19    34.93     10    540
20    46.59     15    250
21    44.88     15    290
```

```
22    54.12     16    510
23    56.63     17    590
24    22.13      6    100
25    21.15      5    400
```

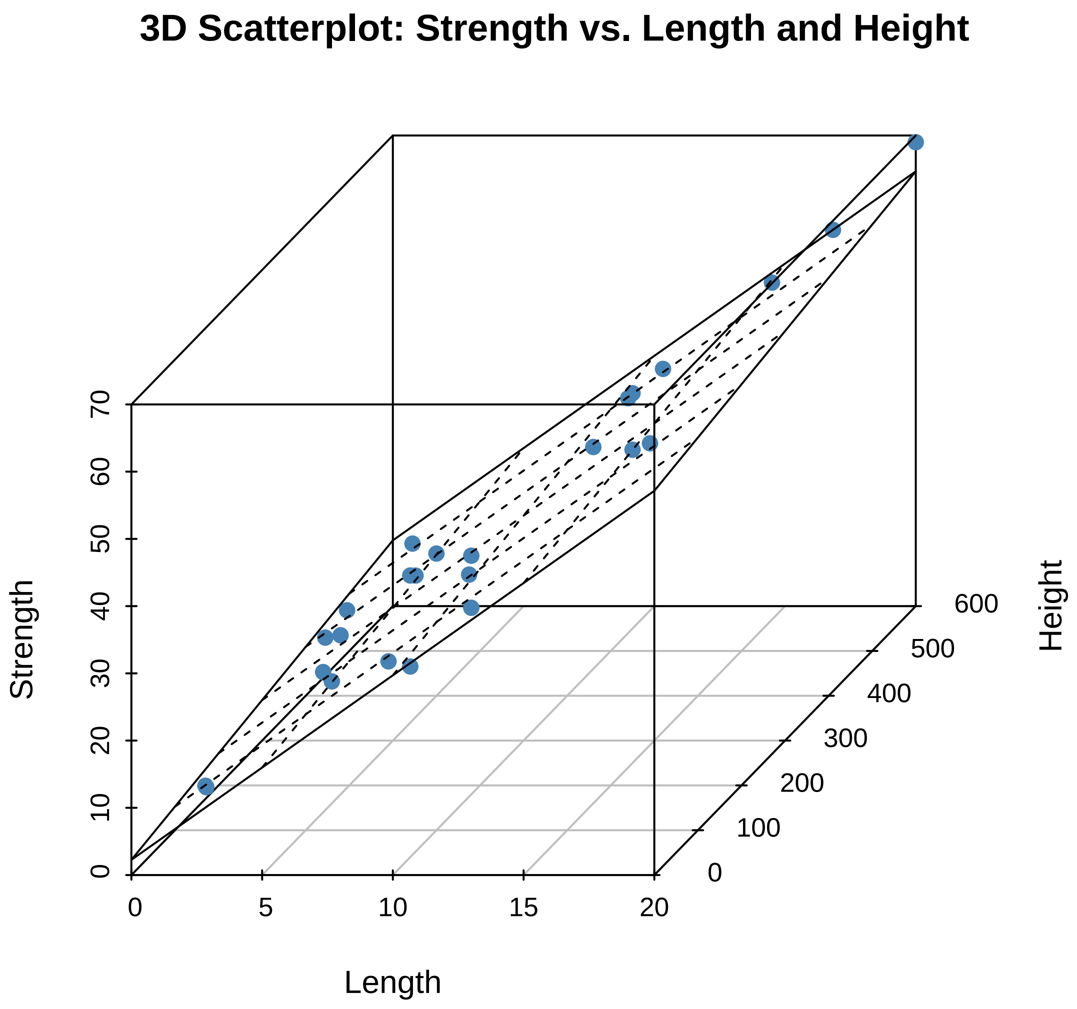
2D Visualization

```
par(mfrow = c(1, 3), mar = c(5, 4, 2, 1))

## 1) length vs strength
i1 <- which(!is.na(bond.data$length) & !is.na(bond.data$strength))
plot(bond.data$length[i1], bond.data$strength[i1],
     xlab = "Wire Length", ylab = "Pull strength", pch = 19)
text(bond.data$length[i1], bond.data$strength[i1],
     labels = i1, pos = 1, offset = 0.4, cex = 0.75)

## 2) height vs strength
i2 <- which(!is.na(bond.data$height) & !is.na(bond.data$strength))
plot(bond.data$height[i2], bond.data$strength[i2],
     xlab = "Die height", ylab = "Pull strength", pch = 19)
text(bond.data$height[i2], bond.data$strength[i2],
     labels = i2, pos = 1, offset = 0.4, cex = 0.75)

## 3) height vs length
i3 <- which(!is.na(bond.data$height) & !is.na(bond.data$length))
plot(bond.data$height[i3], bond.data$length[i3],
     xlab = "Die height", ylab = "Length", pch = 19)
text(bond.data$height[i3], bond.data$length[i3],
     labels = i3, pos = 1, offset = 0.4, cex = 0.75)
```



3D Visualize

```r
library(scatterplot3d)

par(mfrow = c(1,1))
s3d <- with(bond.data, scatterplot3d(
  x = length,
  y = height,
  z = strength,
  pch = 19,
  color = "steelblue",
  main = "3D Scatterplot: Strength vs. Length and Height",
  xlab = "Length",
  ylab = "Height",
  zlab = "Strength",
  angle = 60
))

fit <- lm(strength ~ length + height, data = bond.data)
s3d$plane3d(fit, lty.box = "solid")
```



4.1.2 Model Fitting and Summary

We fit a multiple linear regression model with `strength` as the response variable and `length` and `height` as predictors.

```
fit <- lm(strength ~ length + height, data = bond.data)
summary(fit)
```

```
Call:
lm(formula = strength ~ length + height, data = bond.data)

Residuals:
   Min     1Q Median     3Q    Max
-3.865 -1.542 -0.362  1.196  5.841

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.263791   1.060066   2.136 0.044099 *
length      2.744270   0.093524  29.343  < 2e-16 ***
height      0.012528   0.002798   4.477 0.000188 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.288 on 22 degrees of freedom
Multiple R-squared:  0.9811,    Adjusted R-squared:  0.9794
F-statistic: 572.2 on 2 and 22 DF,  p-value: < 2.2e-16
```

The summary provides the ANOVA F-test for overall significance, R^2, adjusted R^2, and t-tests for individual coefficients.

4.1.3 Confidence Intervals and Model Components

```r
## Confidence intervals
confint(fit)
```

```
                  2.5 %     97.5 %
(Intercept) 0.065348613 4.46223426
length      2.550313061 2.93822623
height      0.006724246 0.01833138
```

```r
## Fitted values and residuals
pred <- fitted.values(fit)
e <- resid(fit)
data.frame(y = bond.data$strength, y.hat = pred, e = e)
```

```
       y     y.hat           e
1   9.95  8.378721  1.57127871
2  24.45 25.596008 -1.14600783
3  31.75 33.954095 -2.20409488
```

```
4  35.00 36.596784 -1.59678413
5  25.02 27.913653 -2.89365294
6  16.86 15.746432  1.11356772
7  14.38 12.450260  1.92974001
8   9.60  8.403777  1.19622309
9  24.35 28.214999 -3.86499936
10 27.50 27.976292 -0.47629200
11 17.08 18.402328 -1.32232830
12 37.00 37.461882 -0.46188206
13 41.95 41.458933  0.49106715
14 11.66 12.262343 -0.60234282
15 21.65 15.809071  5.84092866
16 17.89 18.251995 -0.36199456
17 69.00 64.665871  4.33412887
18 10.30 12.336831 -2.03683074
19 34.93 36.471506 -1.54150602
20 46.59 46.559789  0.03021107
21 44.88 47.060901 -2.18090138
22 54.12 52.561290  1.55871047
23 56.63 56.307784  0.32221591
24 22.13 19.982190  2.14780957
25 21.15 20.996264  0.15373580
```

```
## Covariance matrix and standard errors
cov.mat <- vcov(fit)
cov.mat
```

```
             (Intercept)        length        height
(Intercept)  1.123740429 -3.921612e-02 -1.781991e-03
length      -0.039216122  8.746709e-03 -9.903775e-05
height      -0.001781991 -9.903775e-05  7.831149e-06
```

```
data.frame(std.error = sqrt(diag(cov.mat)))
```

```
              std.error
(Intercept) 1.060066238
length      0.093523844
height      0.002798419
```

4.2 RSS-based Inference: F-test, and adjusted R^2

The General Linear Model

The general linear model is:

$$y = X\beta + \epsilon$$

- y: $n \times 1$ vector of responses
- X: $n \times p$ design matrix (first column often ones)
- β: $p \times 1$ parameter vector, where $p = k + 1$
- ϵ: $n \times 1$ error vector

4.2.1 RSS-Based Quantities

4.2.1.1 RSS-Based Quantities

Source	Sum of Squares	R^2	df	Mean Squares	F	SS_{adj}	$\hat{\sigma}^2$	R^2_{adj}
$x^\top\beta$	$\mathrm{SSR} = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2$	$\frac{\mathrm{SSR}}{\mathrm{SST}}$	k	$\mathrm{MSR} = \frac{\mathrm{SSR}}{k}$	$\frac{\mathrm{MSR}}{\mathrm{MSE}}$	$\mathrm{SSR}_{\text{adj}}$	$\hat{\sigma}^2_{x^\top\beta} = \frac{\mathrm{SSR}_{\text{adj}}}{n-1}$	$\frac{\mathrm{SSR}_{\text{adj}}}{\mathrm{SST}} = 1 - \frac{\mathrm{MSE}}{\mathrm{MST}}$
ϵ	$\mathrm{SSE} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$	—	$n-p$	$\mathrm{MSE} = \frac{\mathrm{SSE}}{n-p}$	—	SSE	$\hat{\sigma}^2_\epsilon = \mathrm{MSE}$	—
y	$\mathrm{SST} = \sum_{i=1}^{n}(y_i - \bar{y})^2$	—	$n-1$	$\mathrm{MST} = \frac{\mathrm{SST}}{n-1}$	—	SST	$\hat{\sigma}^2_y = \mathrm{MST}$	—

Interpretation of the $\hat{\sigma}^2$ Column

The $\hat{\sigma}^2$ column highlights how each sum of squares corresponds to an estimated variance.
This view makes the adjusted coefficient of determination clear:

$$R^2_{\text{adj}} = 1 - \frac{\hat{\sigma}^2_\epsilon}{\hat{\sigma}^2_y} = \frac{\hat{\sigma}^2_{x^\top\beta}}{\hat{\sigma}^2_y}.$$

Hence, the adjusted R^2 simply expresses the **proportion of total estimated variance** attributable to the fitted model $X\beta$ rather than the residual noise ϵ.

4.2.2 Remarks

4.2.2.1 Fundamental Identities

$$\begin{aligned} \text{SST} &= \text{SSR} + \text{SSE}, \\ \text{MST} &= \text{MSE} + \frac{\text{SSR}_{\text{adj}}}{n-1}. \end{aligned}$$

where

$$\text{SSR}_{\text{adj}} = (n-1)MST - (n-p+k)\text{MSE} = \text{SST} - \text{SSE} - k\,\text{MSE} = \text{SSR} - k\,\text{MSE}.$$

4.2.2.2 Difference of $\hat{\sigma}^2$ and Mean Squares

The quantity $\hat{\sigma}^2$ represents the **estimated variance** associated with each component of the model. MSE and MST are the estimated variances of the ϵ and y itself. However, the MSR, although called **Mean Square for Regression (MSR)** is *NOT* an estimate of the variance or sample variance of $x^\top \beta$. The name of "mean" here is used to indicate a different thing. Its name "Mean Square" reflects that it is also an estimate estimate of noise variance σ^2 under $H_0\colon\ \beta = 0$:

$$E[\text{MSR} \mid H_0] = \sigma^2, \qquad E[\text{MSR} \mid H_1] > \sigma^2.$$

Hence the F-statistic

$$F = \frac{\text{MSR}}{\text{MSE}}$$

is approximately equal to 1 subject to the variability as characterized with F-distribution with degree freedoms of k and $n - p$. This test is to test whether any regression coefficients are not equal to 0.

4.2.2.3 $\hat{\sigma}^2_{x^\top \beta} = \frac{\text{SSR}_{\text{adj}}}{n-1}$

$\hat{\sigma}^2_{x^\top \beta}$ is an unbiased estimator of the variance of linear signal when x is a regarded as a random variable. This can be seen from the following equations:

$$E[\text{SSR}] = k\,\sigma^2 + \beta^\top X^\top (I - J/n)\, X\, \beta, \qquad E[\text{MSE}] = \sigma^2.$$

Hence,

$$\begin{aligned} E[\text{SSR}_{\text{adj}}] &= E[\text{SSR}] - k\,E[\text{MSE}] \\ &= \beta^\top X^\top (I - J/n)\, X\, \beta \\ &= \sum_{i=1}^{n} (\mu_i - \bar{\mu})^2, \end{aligned}$$

where

$$\begin{aligned} \mu_i &= x_i^\top \beta \\ \bar{\mu} &= \tfrac{1}{n} \sum_{i=1}^{n} \mu_i \end{aligned}$$

For fixed X, $\text{SSR}_{\text{adj}}/(n-1)$ equals the **sample variance** of the true means $\{\mu_i\}$ over the observed design points. If the rows of X are independently sampled with covariance matrix Σ_X (the random-X model), then

$$\mathbb{E}_X \left[\frac{\text{SSR}_{\text{adj}}}{n-1} \right] = \beta^\top \Sigma_X \beta = \text{Var}(x^\top \beta),$$

4.2.2.4 Connection to Rao-Blackwell Formula

The decomposition of $\hat{\sigma}^2$ is consistent with the **Rao–Blackwell formula** for total variance:

$$\text{Var}(y) = \text{Var}\big(E[y \mid x]\big) + E\big(\text{Var}[y \mid x]\big).$$

Here,

- $\text{Var}\big(E[y \mid x]\big)$ corresponds to the **explained variation** due to the regression component $x^\top \beta$, and
- $E\big(\text{Var}[y \mid x]\big)$ corresponds to the **residual variation** due to ϵ.

4.2.3 A Simulation Study to Understand the Distributions of RSS

Data Generating Model

For $n = 30$ and $p_{max} = 20$, simulate with either $H_0 : \beta = \mathbf{0}$ or H_1 where only $\beta_1 \neq 0$; $\epsilon_i \sim N(0,1)$.

Sequence of Fitted Models

Model Name	# of Predictors (k)	# of Parameters (p)	R Formula
Model 0	0	1	`y ~ 1`
Model 1	2	3	`y ~ x_1 + x_2`
…	…	…	…
Final Model	20	21	`y ~ x_1 + ... + x_20`

4.2.3.1 When H_0 is true

rss-h0.mp4

4.2.3.2 When H_1 is true

rss-h1.mp4

4.2.4 Example: Modelling Children Weight with Height and Age

```r
## Data: Weight, height and age of children
wgt <- c(64, 71, 53, 67, 55, 58, 77, 57, 56, 51, 76, 68)
hgt <- c(57, 59, 49, 62, 51, 50, 55, 48, 42, 42, 61, 57)
age <- c(8, 10, 6, 11, 8, 7, 10, 9, 10, 6, 12, 9)
child.data <- data.frame(wgt, hgt, age)
```

4.2.4.1 Problem 1: Height then Age

```r
fit_hgt_age <- lm(wgt ~ hgt + age, data = child.data)
summary(fit_hgt_age)
```

```

Call:
lm(formula = wgt ~ hgt + age, data = child.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-6.8708 -1.7004  0.3454  1.4642 10.2336 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   6.5530    10.9448   0.599   0.5641
```

```
hgt           0.7220     0.2608   2.768   0.0218 *
age           2.0501     0.9372   2.187   0.0565 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.66 on 9 degrees of freedom
Multiple R-squared:   0.78, Adjusted R-squared:  0.7311
F-statistic: 15.95 on 2 and 9 DF,  p-value: 0.001099
```

```
fit_hgt <- lm(wgt ~ hgt, data = child.data)
summary(fit_hgt)
```

```

Call:
lm(formula = wgt ~ hgt, data = child.data)

Residuals:
    Min      1Q  Median      3Q     Max
-5.8736 -3.8973 -0.4402  2.2624 11.8375

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   6.1898    12.8487   0.482  0.64035
hgt           1.0722     0.2417   4.436  0.00126 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.471 on 10 degrees of freedom
Multiple R-squared:  0.663, Adjusted R-squared:  0.6293
F-statistic: 19.67 on 1 and 10 DF,  p-value: 0.001263
```

```
anova(fit_hgt, fit_hgt_age)
```

```
Analysis of Variance Table

Model 1: wgt ~ hgt
Model 2: wgt ~ hgt + age
  Res.Df    RSS Df Sum of Sq      F  Pr(>F)
1     10 299.33
2      9 195.43  1     103.9 4.7849 0.05649 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
anova(fit_hgt_age)
```

```
Analysis of Variance Table

Response: wgt
          Df Sum Sq Mean Sq F value    Pr(>F)    
hgt        1 588.92  588.92 27.1216 0.0005582 ***
age        1 103.90  103.90  4.7849 0.0564853 .  
Residuals  9 195.43   21.71                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.2.4.2 Problem 2: Age then Height

```r
fit_age <- lm(wgt ~ age, data = child.data)
summary(fit_age)
```

```

Call:
lm(formula = wgt ~ age, data = child.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-11.000  -3.911   1.143   4.071  10.000 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)   
(Intercept)  30.5714     8.6137   3.549  0.00528 **
age           3.6429     0.9551   3.814  0.00341 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.015 on 10 degrees of freedom
Multiple R-squared:  0.5926,    Adjusted R-squared:  0.5519 
F-statistic: 14.55 on 1 and 10 DF,  p-value: 0.003407
```

```r
fit_age_hgt <- lm(wgt ~ age + hgt, data = child.data)
summary(fit_age_hgt)
```

```

Call:
```

```
lm(formula = wgt ~ age + hgt, data = child.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-6.8708 -1.7004  0.3454  1.4642 10.2336 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)  
(Intercept)   6.5530    10.9448   0.599   0.5641  
age           2.0501     0.9372   2.187   0.0565 .
hgt           0.7220     0.2608   2.768   0.0218 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.66 on 9 degrees of freedom
Multiple R-squared:   0.78, Adjusted R-squared:  0.7311 
F-statistic: 15.95 on 2 and 9 DF,  p-value: 0.001099
```

```
anova(fit_age, fit_age_hgt)
```

```
Analysis of Variance Table

Model 1: wgt ~ age
Model 2: wgt ~ age + hgt
  Res.Df    RSS Df Sum of Sq      F  Pr(>F)  
1     10 361.86                              
2      9 195.43  1    166.43 7.6646 0.02181 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(fit_age_hgt)
```

```
Analysis of Variance Table

Response: wgt
          Df Sum Sq Mean Sq F value    Pr(>F)    
age        1 526.39  526.39 24.2419 0.0008205 ***
hgt        1 166.43  166.43  7.6646 0.0218070 *  
Residuals  9 195.43   21.71                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.2.5 Example: Wire bond strength

```r
fit_len_hgt <-  lm(strength ~ length + height, data = bond.data)
fit_hgt_len <-  lm(strength ~ height+length, data = bond.data)
anova(fit_len_hgt)
```

```
Analysis of Variance Table

Response: strength
          Df Sum Sq Mean Sq  F value    Pr(>F)
length     1 5885.9  5885.9 1124.293 < 2.2e-16 ***
height     1  104.9   104.9   20.041 0.0001883 ***
Residuals 22  115.2     5.2
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
anova(fit_hgt_len)
```

```
Analysis of Variance Table

Response: strength
          Df Sum Sq Mean Sq F value    Pr(>F)
height     1 1483.2  1483.2  283.32 4.731e-14 ***
length     1 4507.5  4507.5  861.01 < 2.2e-16 ***
Residuals 22  115.2     5.2
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
summary(fit_hgt_len)
```

```


Call:
lm(formula = strength ~ height + length, data = bond.data)

Residuals:
   Min     1Q Median     3Q    Max
-3.865 -1.542 -0.362  1.196  5.841

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.263791   1.060066   2.136 0.044099 *
```

```
height      0.012528   0.002798   4.477 0.000188 ***
length      2.744270   0.093524  29.343  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.288 on 22 degrees of freedom
Multiple R-squared:  0.9811,    Adjusted R-squared:  0.9794
F-statistic: 572.2 on 2 and 22 DF,  p-value: < 2.2e-16
```

```r
summary(fit_len_hgt)
```

```

Call:
lm(formula = strength ~ length + height, data = bond.data)

Residuals:
   Min     1Q Median     3Q    Max
-3.865 -1.542 -0.362  1.196  5.841

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.263791   1.060066   2.136 0.044099 *
length      2.744270   0.093524  29.343  < 2e-16 ***
height      0.012528   0.002798   4.477 0.000188 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.288 on 22 degrees of freedom
Multiple R-squared:  0.9811,    Adjusted R-squared:  0.9794
F-statistic: 572.2 on 2 and 22 DF,  p-value: < 2.2e-16
```

4.2.6 Relationship between t-test and partial F-test

- A t-test for a single coefficient is a special case of the partial F-test; the relationship is $F = t^2$ for 1 df in the numerator.
- The p-value from t-test (output of *summary(lm())*) is the same as anova test for: $H_0 : \beta_j = 0$ vs H_1: all covaraites have non-zero effects.

4.3 Predictions for Mean Response and a Future Observation

4.3.1 Confidence Interval for Mean Response

```r
predict(fit, newdata = data.frame(length = 8, height = 275),
        interval = "confidence", level = 0.95)
```

```
      fit      lwr      upr
1 27.6631 26.66324 28.66296
```

4.3.2 Prediction Interval for a New Observation

```r
predict(fit, newdata = data.frame(length = 8, height = 275),
        interval = "prediction", level = 0.95)
```

```
      fit      lwr      upr
1 27.6631 22.81378 32.51241
```

4.4 Model Diagnostics

4.4.1 Residual Calculations

```r
residuals_df <- data.frame(
  hat_values = hatvalues(fit),
  ordinary_resid = resid(fit),
  standardized_resid = resid(fit) / sigma(fit),
  studentized_internal = rstandard(fit),
  studentized_external = rstudent(fit)
)
residuals_df
```

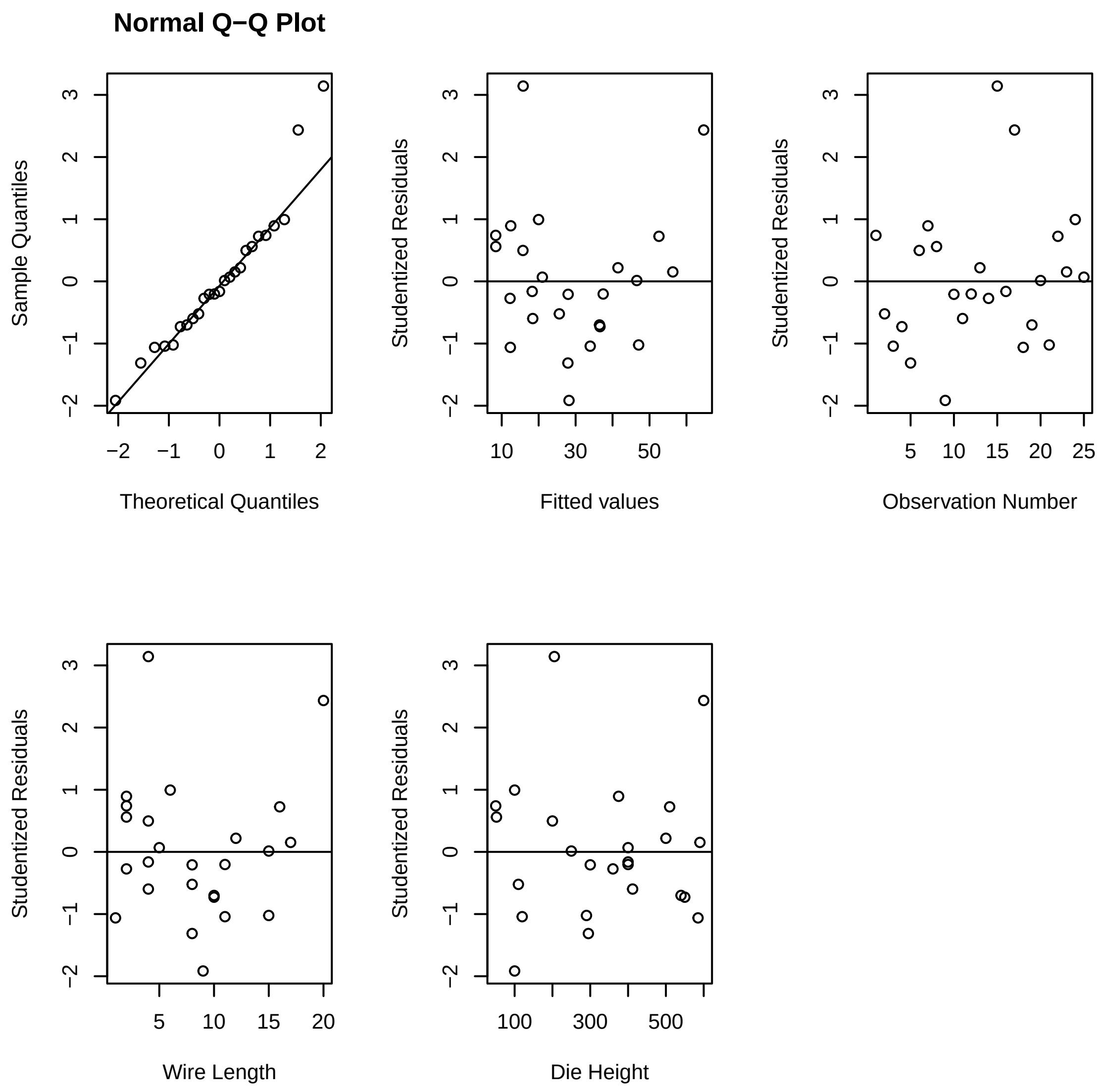
```
   hat_values ordinary_resid standardized_resid studentized_internal
1  0.15728923     1.57127871         0.68673363           0.74808172
2  0.11164598    -1.14600783        -0.50086730          -0.53140990
3  0.14191905    -2.20409488        -0.96330846          -1.03992315
4  0.10188923    -1.59678413        -0.69788088          -0.73640435
5  0.04178381    -2.89365294        -1.26468257          -1.29196212
```

```
6  0.07486842     1.11356772          0.48668921            0.50599936
7  0.11806106     1.92974001          0.84340057            0.89807919
8  0.15608149     1.19622309          0.52281407            0.56911105
9  0.12797685    -3.86499936         -1.68921340           -1.80892479
10 0.04131672    -0.47629200         -0.20816532           -0.21260369
11 0.09253979    -1.32232830         -0.57792886           -0.60668127
12 0.05256700    -0.46188206         -0.20186740           -0.20739197
13 0.08202675     0.49106715          0.21462286            0.22400668
14 0.11291577    -0.60234282         -0.26325633           -0.27950939
15 0.07373697     5.84092866          2.55280118            2.65246601
16 0.08794942    -0.36199456         -0.15821117           -0.16566382
17 0.25934228     4.33412887          1.89424832            2.20104100
18 0.29287870    -2.03683074         -0.89020500           -1.05862725
19 0.09617553    -1.54150602         -0.67372136           -0.70866056
20 0.14726101     0.03021107          0.01320387            0.01429859
21 0.12963943    -2.18090138         -0.95317165           -1.02169558
22 0.13580052     1.55871047          0.68124063            0.73281364
23 0.18237610     0.32221591          0.14082575            0.15574183
24 0.10908869     2.14780957          0.93870874            0.99452024
25 0.07287021     0.15373580          0.06719084            0.06978142
   studentized_external
1            0.74035927
2           -0.52255660
3           -1.04194550
4           -0.72850799
5           -1.31305171
6            0.49726770
7            0.89397096
8            0.56016499
9           -1.91552083
10          -0.20792931
11          -0.59775404
12          -0.20282206
13           0.21910643
14          -0.27356920
15           3.14216850
16          -0.16195600
17           2.43521394
18          -1.06168251
19          -0.70040768
20           0.01396991
21          -1.02276424
22           0.72486668
23           0.15224503
24           0.99426154
```

```
25          0.06818458
```

4.4.2 Residual Plots

```
n <- nrow(bond.data)
r <- rstudent(fit)
y.hat <- fitted.values(fit)

par(mfrow = c(2, 3))
qqnorm(r, main = "Normal Q-Q Plot"); qqline(r)
plot(y.hat, r, xlab = "Fitted values", ylab = "Studentized Residuals"); abline(h = 0)
plot(1:n, r, xlab = "Observation Number", ylab = "Studentized Residuals"); abline(h = 0)
plot(bond.data$length, r, xlab = "Wire Length", ylab = "Studentized Residuals"); abline(h = 0)
plot(bond.data$height, r, xlab = "Die Height", ylab = "Studentized Residuals"); abline(h = 0)
```



4.5 Influential Observations

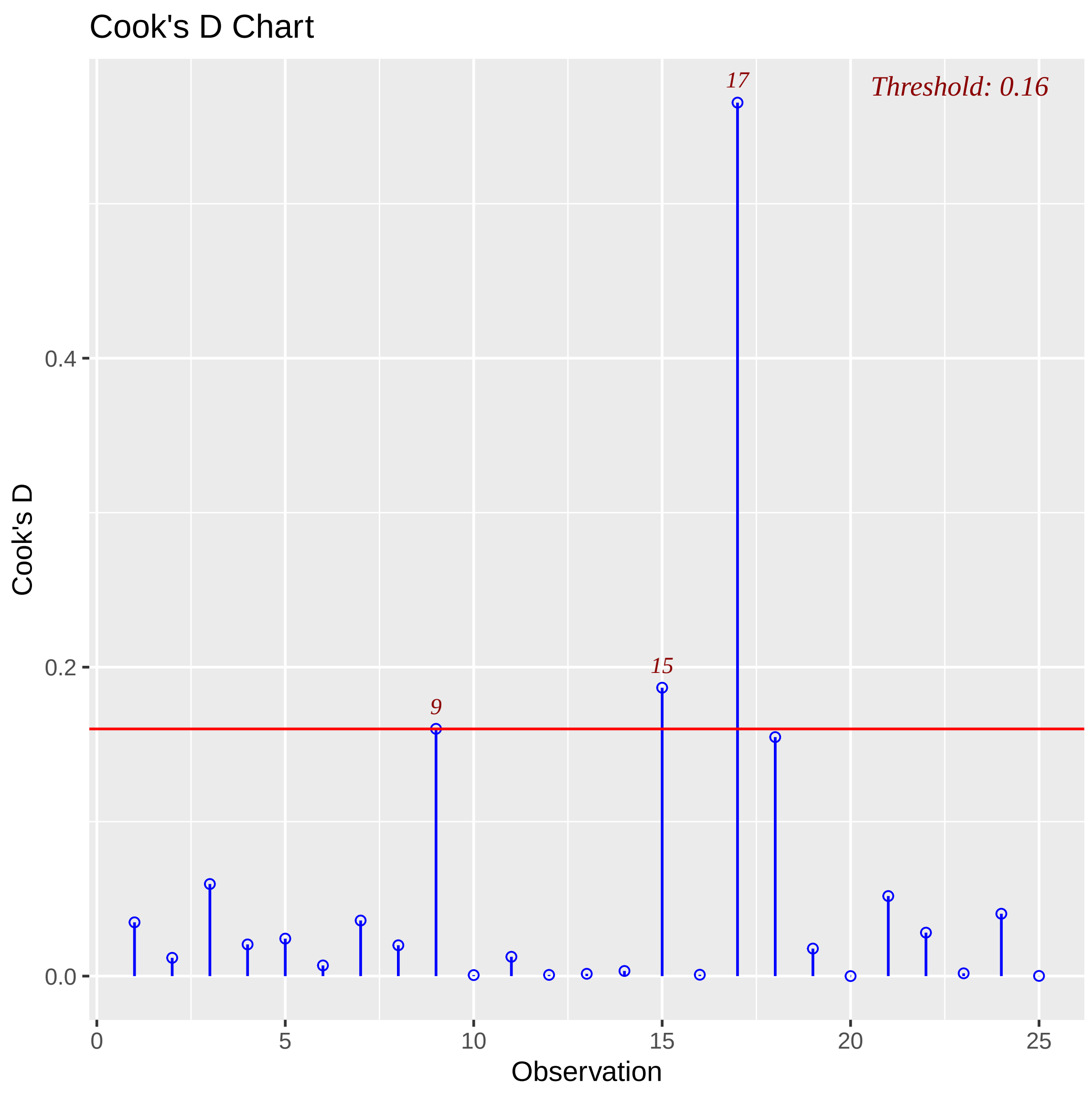
```r
influence_df <- data.frame(dffits = dffits(fit),
                           cook.D = cooks.distance(fit),
                           dfbetas(fit))
influence_df
```

```
        dffits       cook.D  X.Intercept.       length       height
1   0.319854702 3.481748e-02  0.3179493921 -0.100534181 -0.200085326
2  -0.185251548 1.183028e-02 -0.1403477437 -0.051464370  0.148315370
3  -0.423741713 5.962023e-02 -0.2219151046 -0.237135616  0.339340552
```

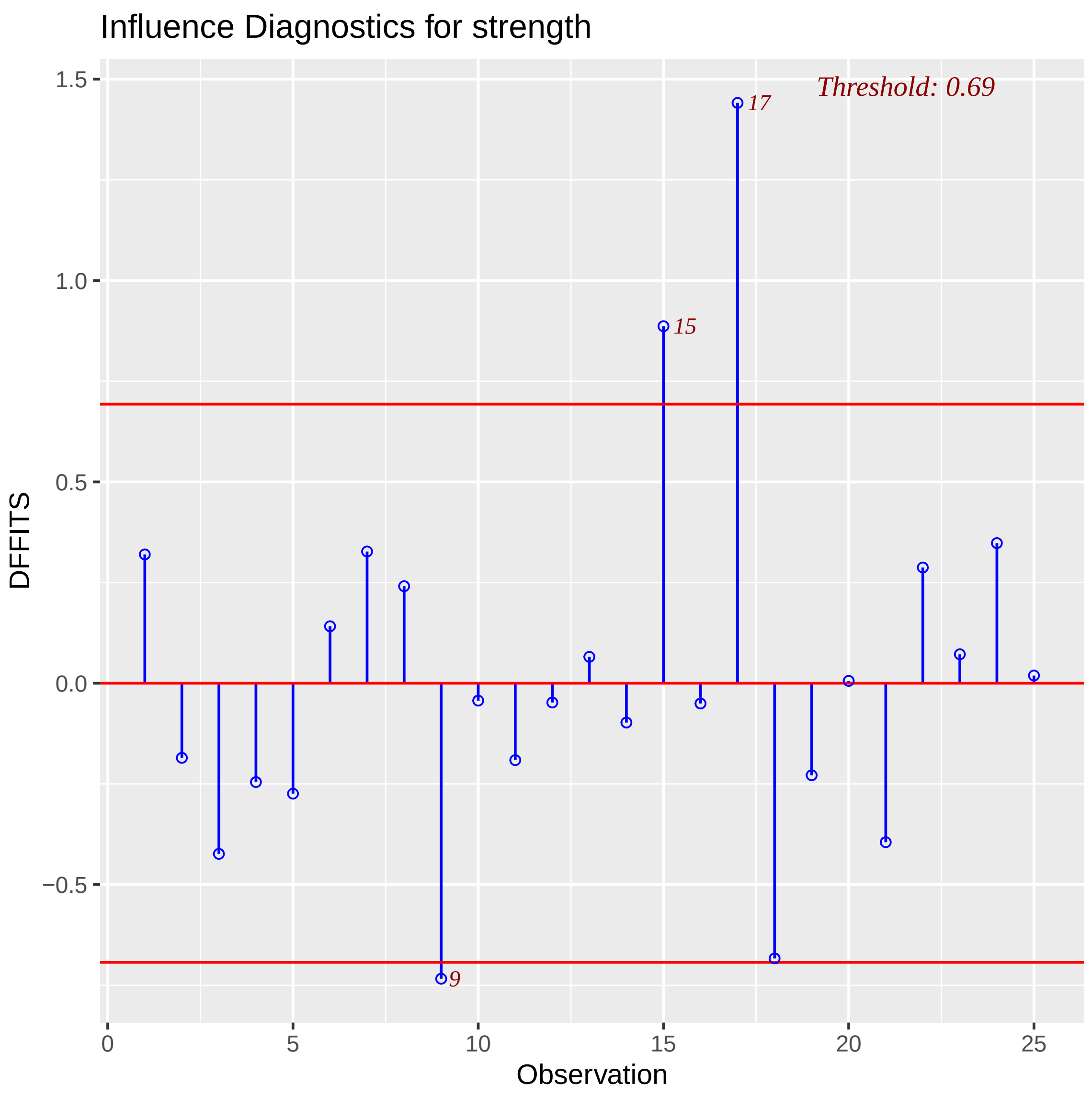
```
4  -0.245376811 2.050736e-02  0.0787635526  0.022343842 -0.184260891
5  -0.274191565 2.426179e-02 -0.1572410603 -0.009662357  0.055328303
6   0.141461363 6.906752e-03  0.1301249135 -0.058073567 -0.049408295
7   0.327082639 3.598953e-02  0.1479853099 -0.261848970  0.142220906
8   0.240902557 1.996750e-02  0.2394962591 -0.076575387 -0.149787102
9  -0.733818383 1.600749e-01 -0.5011686139 -0.283749099  0.605559181
10 -0.043165927 6.493384e-04 -0.0241138520 -0.001038287  0.007460760
11 -0.190885522 1.251125e-02 -0.0602003847  0.132053762 -0.102733745
12 -0.047774650 7.954763e-04  0.0017554145 -0.016926531 -0.008495572
13  0.065496465 1.494604e-03 -0.0224255162  0.017340091  0.033756253
14 -0.097602753 3.314830e-03 -0.0483552961  0.077880727 -0.038066705
15  0.886553487 1.866936e-01  0.8097463528 -0.374290156 -0.292048214
16 -0.050292599 8.821617e-04 -0.0177647675  0.034746758 -0.025275682
17  1.441003392 5.654455e-01 -0.8513738015  1.008880052  0.413618783
18 -0.683268805 1.547244e-01 -0.0218935465  0.521608456 -0.532432956
19 -0.228476293 1.781295e-02  0.0700729860  0.018004228 -0.167581999
20  0.005805362 1.176892e-05  0.0005613509  0.004752581 -0.003094588
21 -0.394724862 5.182743e-02 -0.0084618169 -0.324109965  0.170622396
22  0.287343813 2.812893e-02 -0.1326208213  0.183002776  0.076391058
23  0.071903545 1.803448e-03 -0.0412553376  0.040164093  0.030365108
24  0.347915085 4.036930e-02  0.3084561584  0.016541769 -0.262089402
25  0.019115733 1.275757e-04  0.0062730782 -0.011614674  0.009459029
```

4.5.1 Plotting with the `olsrr` Package

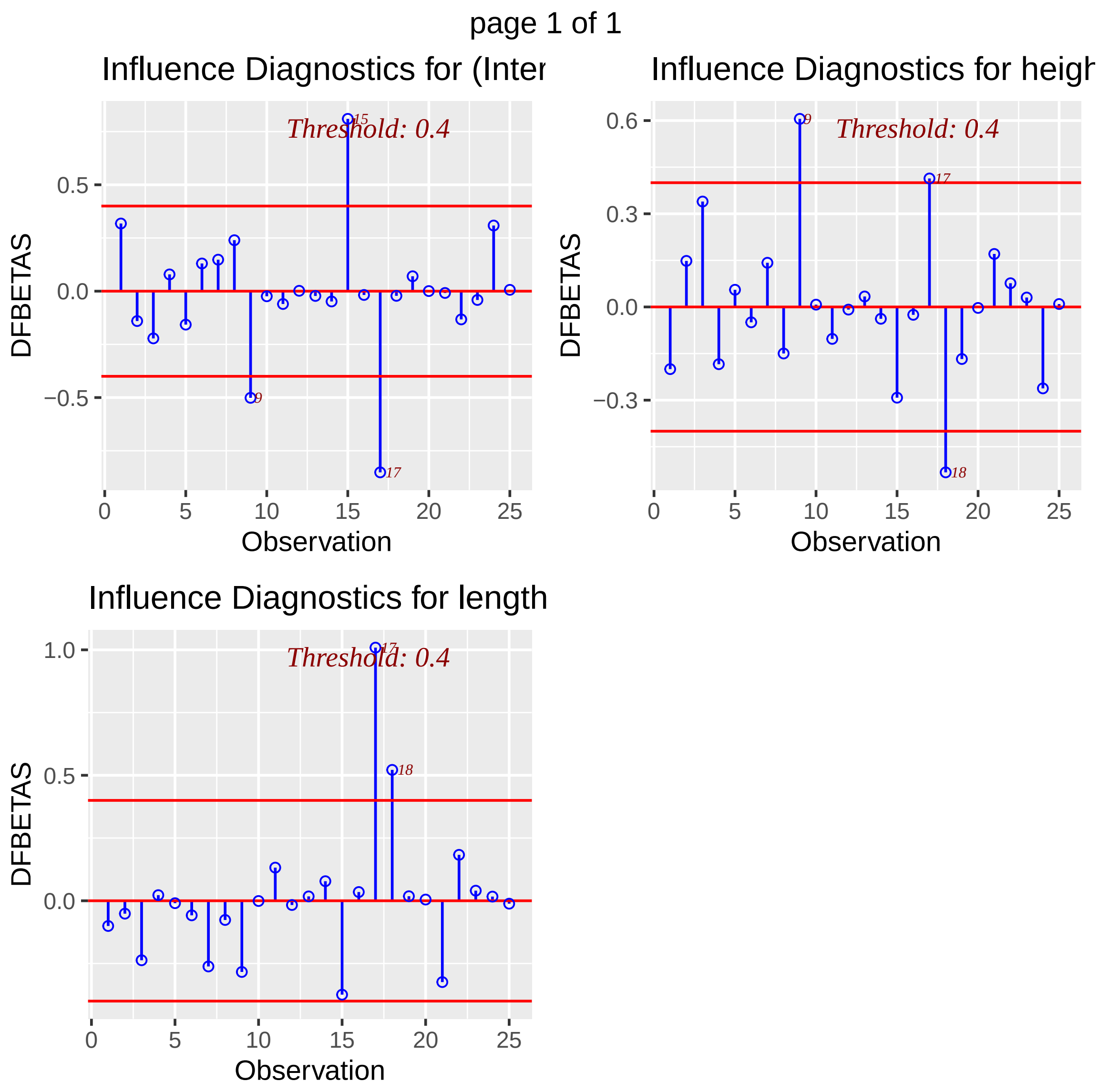
```
## install.packages("olsrr") # Run once if needed
library(olsrr)

ols_plot_cooksd_chart(fit)
```



```r
ols_plot_dffits(fit)
```



```r
ols_plot_dfbetas(fit)
```



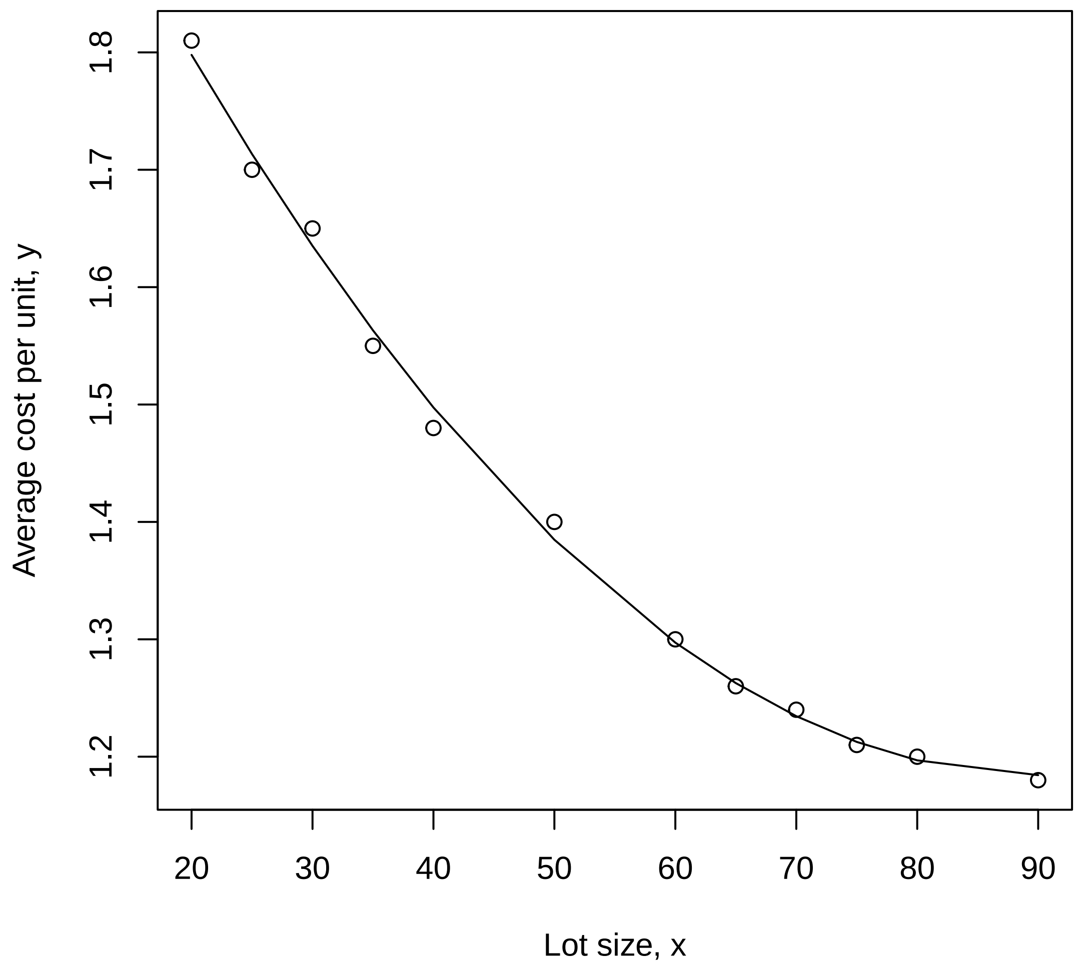
4.6 Polynomial Regression

```r
y <- c(1.81, 1.70, 1.65, 1.55, 1.48, 1.40, 1.30, 1.26, 1.24, 1.21, 1.20, 1.18)
x <- c(20, 25, 30, 35, 40, 50, 60, 65, 70, 75, 80, 90)
fit_poly <- lm(y ~ x + I(x^2))
summary(fit_poly)
```

```
Call:
lm(formula = y ~ x + I(x^2))
```

```
Residuals:
       Min         1Q     Median         3Q        Max 
-0.0174763 -0.0065087  0.0001297  0.0071482  0.0151887 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  2.198e+00  2.255e-02   97.48 6.38e-15 ***
x           -2.252e-02  9.424e-04  -23.90 1.88e-09 ***
I(x^2)       1.251e-04  8.658e-06   14.45 1.56e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.01219 on 9 degrees of freedom
Multiple R-squared:  0.9975,	Adjusted R-squared:  0.9969 
F-statistic:  1767 on 2 and 9 DF,  p-value: 2.096e-12
```

```r
plot(x, y, xlab = "Lot size, x", ylab = "Average cost per unit, y")
lines(x, predict(fit_poly, newdata = data.frame(x = x)), type = "l")
```



```r
fit1 <- lm(y ~ x)
anova(fit1, fit_poly)
```

```
Analysis of Variance Table

Model 1: y ~ x
Model 2: y ~ x + I(x^2)
  Res.Df      RSS Df Sum of Sq      F    Pr(>F)
1     10 0.032340
2      9 0.001337  1  0.031002 208.67 1.564e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.7 Handling Categorical Variables with Dummy Variables

Investigate the common observation that males tend to have higher blood pressure than females of similar age.

```r
## Note: Update this path to your local file location
sbpdata <- read.csv("sbpdata.csv")
sbpdata
```

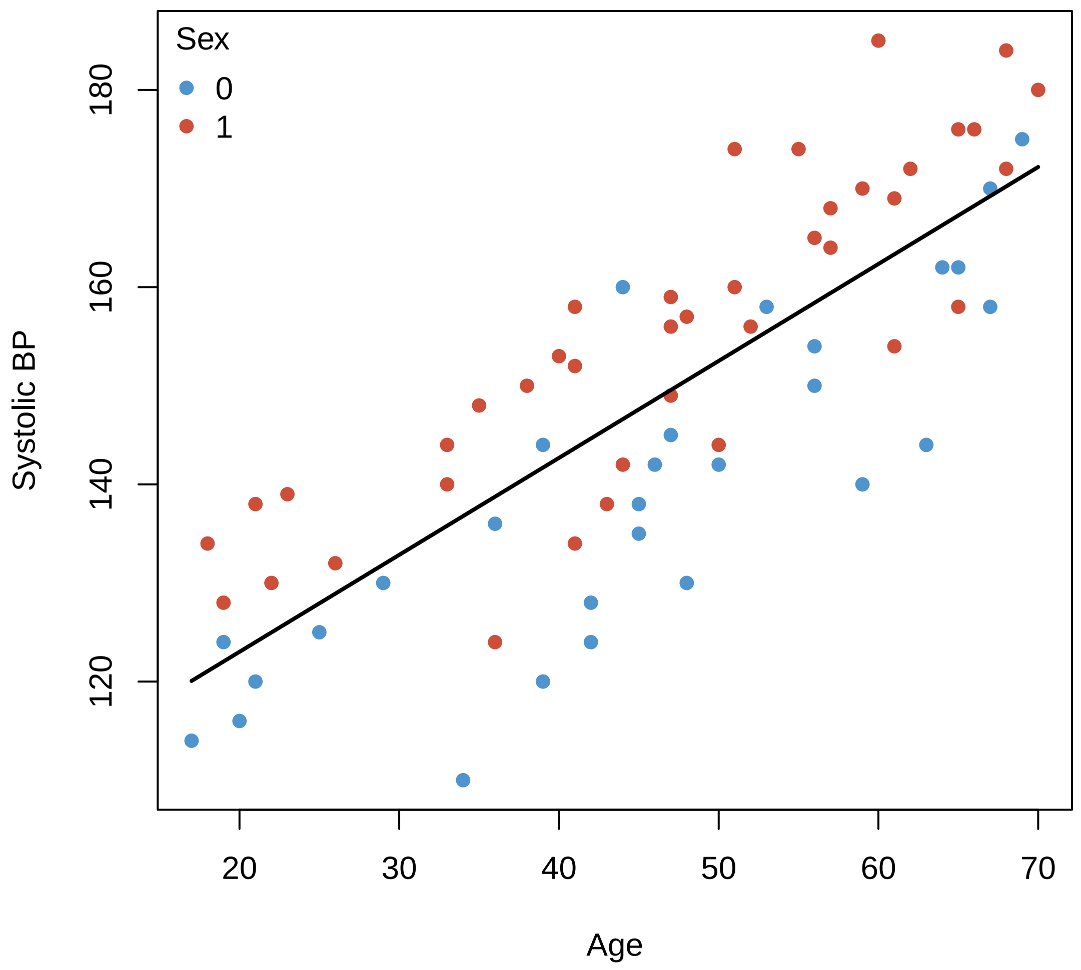
```
   sex sbp age
1    0 144  39
2    0 138  45
3    0 145  47
4    0 162  65
5    0 142  46
6    0 170  67
7    0 124  42
8    0 158  67
9    0 154  56
10   0 162  64
11   0 150  56
12   0 140  59
13   0 110  34
14   0 128  42
15   0 130  48
16   0 135  45
17   0 114  17
18   0 116  20
19   0 124  19
20   0 136  36
21   0 142  50
22   0 120  39
23   0 120  21
24   0 160  44
25   0 158  53
26   0 144  63
27   0 130  29
28   0 125  25
29   0 175  69
30   1 158  41
31   1 185  60
32   1 152  41
33   1 159  47
34   1 176  66
35   1 156  47
36   1 184  68
37   1 138  43
38   1 172  68
39   1 168  57
```

```
40   1 176  65
41   1 164  57
42   1 154  61
43   1 124  36
44   1 142  44
45   1 144  50
46   1 149  47
47   1 128  19
48   1 130  22
49   1 138  21
50   1 150  38
51   1 156  52
52   1 134  41
53   1 134  18
54   1 174  51
55   1 174  55
56   1 158  65
57   1 144  33
58   1 139  23
59   1 180  70
60   1 165  56
61   1 172  62
62   1 160  51
63   1 157  48
64   1 170  59
65   1 153  40
66   1 148  35
67   1 140  33
68   1 132  26
69   1 169  61
```

4.7.1 Four Models Involving "sex"

4.7.1.1 Coincidence Model (Age Only)

```r
## Ensure sex is a factor (labels will appear in the legend)
sbpdata$sex <- as.factor(sbpdata$sex)

## Fit (you already have this)
fit.age <- lm(sbp ~ age, data = sbpdata)

## Generate predictions over the observed age range
new_age <- seq(min(sbpdata$age, na.rm = TRUE),
               max(sbpdata$age, na.rm = TRUE),
               length.out = 200)
pred <- predict(fit.age, newdata = data.frame(age = new_age))

## Simple palette for the sex levels (works for 1-3 levels; expand if needed)
lev  <- levels(sbpdata$sex)
cols <- setNames(c("steelblue3", "tomato3", "darkorchid3")[seq_along(lev)], lev)

## Scatter plot with colored points by sex
plot(sbp ~ age, data = sbpdata,
     col = cols[sbpdata$sex], pch = 16,
     xlab = "Age", ylab = "Systolic BP")

## Add predicted line
lines(new_age, pred, lwd = 2)

## Legend
legend("topleft", legend = lev, col = cols[lev], pch = 16, bty = "n", title = "Sex")
```



```
data.frame(model.matrix(fit.age))
```

```
   X.Intercept. age
1             1  39
2             1  45
3             1  47
4             1  65
5             1  46
6             1  67
7             1  42
8             1  67
9             1  56
10            1  64
11            1  56
12            1  59
13            1  34
14            1  42
```

15	1	48
16	1	45
17	1	17
18	1	20
19	1	19
20	1	36
21	1	50
22	1	39
23	1	21
24	1	44
25	1	53
26	1	63
27	1	29
28	1	25
29	1	69
30	1	41
31	1	60
32	1	41
33	1	47
34	1	66
35	1	47
36	1	68
37	1	43
38	1	68
39	1	57
40	1	65
41	1	57
42	1	61
43	1	36
44	1	44
45	1	50
46	1	47
47	1	19
48	1	22
49	1	21
50	1	38
51	1	52
52	1	41
53	1	18
54	1	51
55	1	55
56	1	65
57	1	33
58	1	23
59	1	70

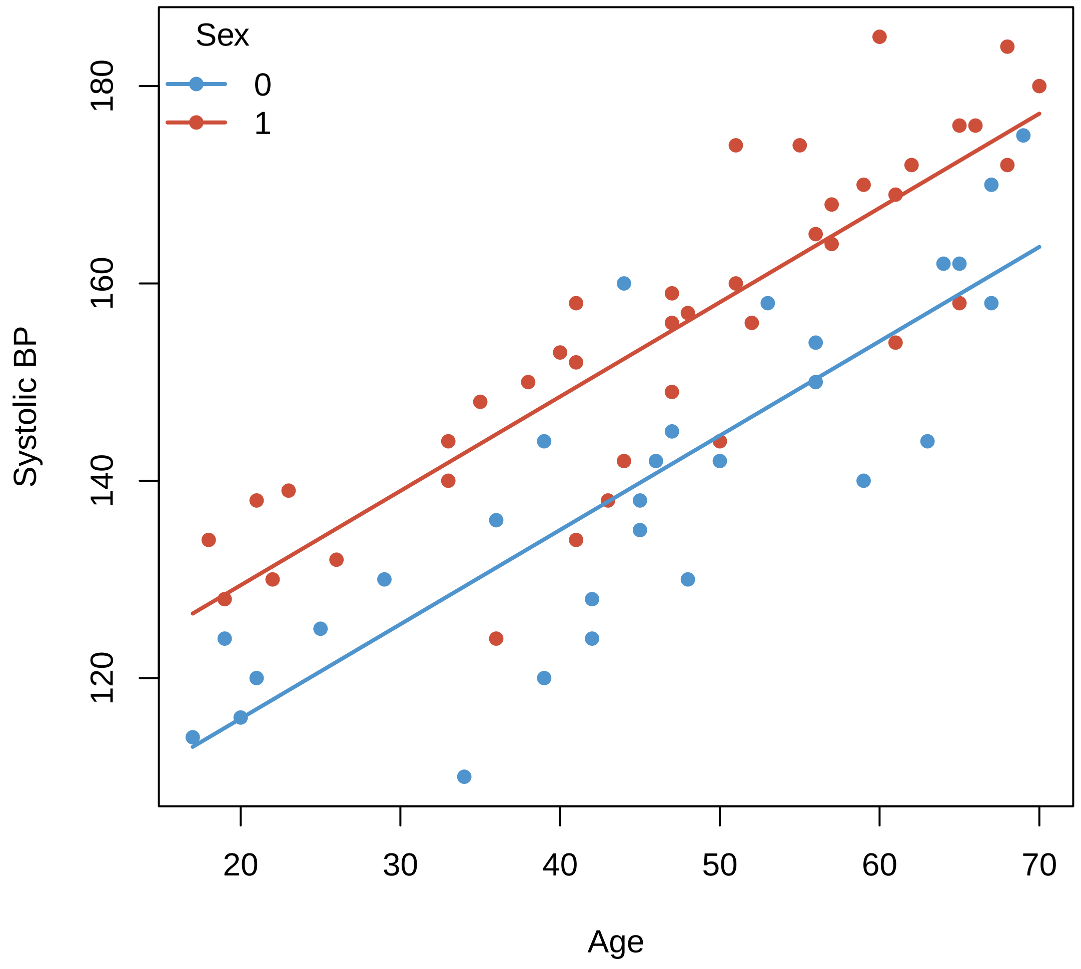
```
60             1  56
61             1  62
62             1  51
63             1  48
64             1  59
65             1  40
66             1  35
67             1  33
68             1  26
69             1  61
```

```
print(anova(fit.age))
```

```
Analysis of Variance Table

Response: sbp
          Df  Sum Sq Mean Sq F value    Pr(>F)
age        1 14951.3 14951.3  121.27 < 2.2e-16 ***
Residuals 67  8260.5   123.3
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.7.1.2 Additive Effect Model (Age + Sex)

```
## Parallelism: H0: beta3=0 (Sex has additive effect)
fit.agePLUSsex <- lm(sbp ~ age + sex, data = sbpdata)

## Ensure sex is a factor for labeling/colors
sbpdata$sex <- factor(sbpdata$sex)

## Fit (additive: parallelism)
fit.agePLUSsex <- lm(sbp ~ age + sex, data = sbpdata)

## X-range and palette
ages <- seq(min(sbpdata$age, na.rm = TRUE),
            max(sbpdata$age, na.rm = TRUE),
            length.out = 200)
lev  <- levels(sbpdata$sex)
cols <- setNames(c("steelblue3", "tomato3", "darkorchid3")[seq_along(lev)], lev)

## Scatter with colored points by sex
plot(sbp ~ age, data = sbpdata,
     col = cols[sbpdata$sex], pch = 16,
     xlab = "Age", ylab = "Systolic BP")

## Parallel fitted lines: one per sex (same slope, different intercepts)
for (sx in lev) {
  nd <- data.frame(age = ages, sex = factor(sx, levels = lev))
  yhat <- predict(fit.agePLUSsex, newdata = nd)
  lines(ages, yhat, col = cols[sx], lwd = 2)
}

## Legend
legend("topleft", legend = lev, col = cols[lev], pch = 16, lwd = 2, bty = "n", title = "Sex")
```



```
data.frame(model.matrix(fit.agePLUSsex))
```

```
  X.Intercept. age sex1
```

```
1               1  39     0
2               1  45     0
3               1  47     0
4               1  65     0
5               1  46     0
6               1  67     0
7               1  42     0
8               1  67     0
9               1  56     0
10              1  64     0
11              1  56     0
12              1  59     0
13              1  34     0
14              1  42     0
15              1  48     0
16              1  45     0
17              1  17     0
18              1  20     0
19              1  19     0
20              1  36     0
21              1  50     0
22              1  39     0
23              1  21     0
24              1  44     0
25              1  53     0
26              1  63     0
27              1  29     0
28              1  25     0
29              1  69     0
30              1  41     1
31              1  60     1
32              1  41     1
33              1  47     1
34              1  66     1
35              1  47     1
36              1  68     1
37              1  43     1
38              1  68     1
39              1  57     1
40              1  65     1
41              1  57     1
42              1  61     1
43              1  36     1
44              1  44     1
45              1  50     1
```

```
46             1  47    1
47             1  19    1
48             1  22    1
49             1  21    1
50             1  38    1
51             1  52    1
52             1  41    1
53             1  18    1
54             1  51    1
55             1  55    1
56             1  65    1
57             1  33    1
58             1  23    1
59             1  70    1
60             1  56    1
61             1  62    1
62             1  51    1
63             1  48    1
64             1  59    1
65             1  40    1
66             1  35    1
67             1  33    1
68             1  26    1
69             1  61    1
```

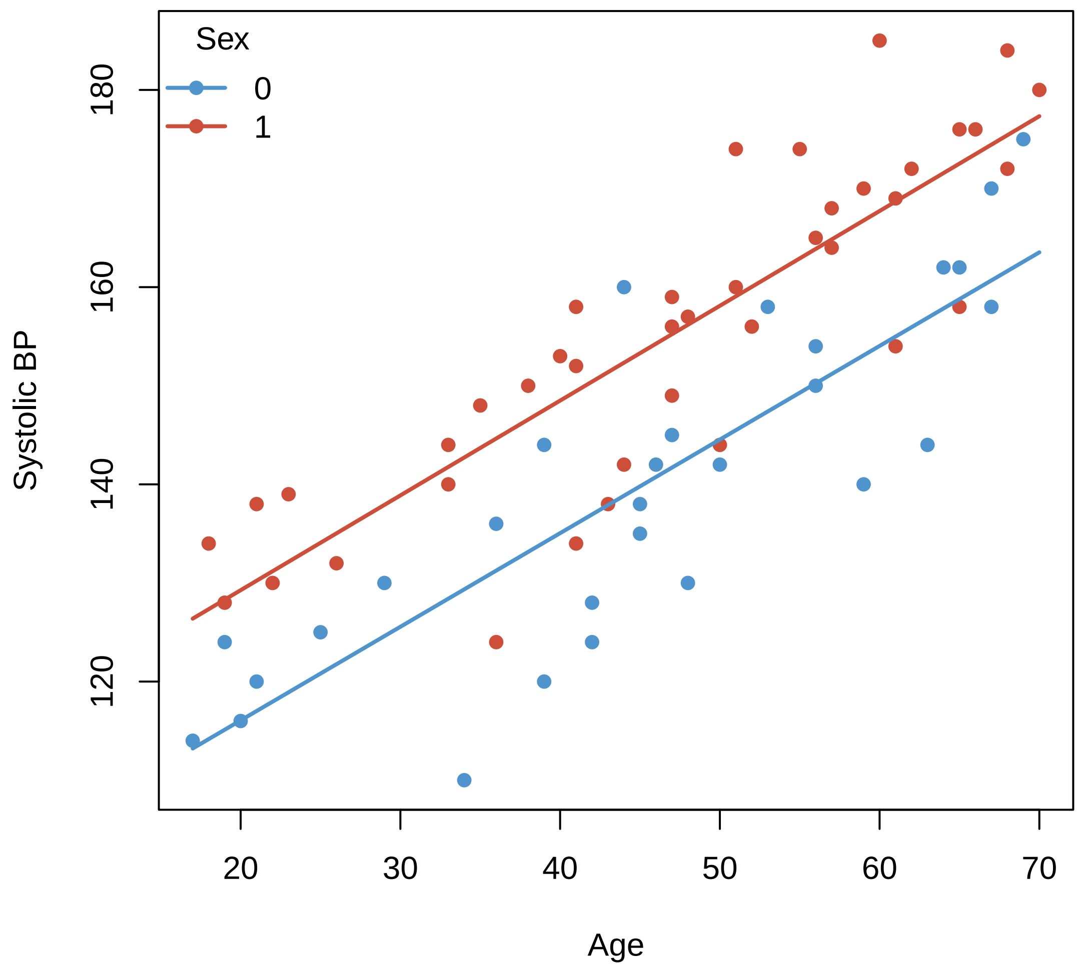
```
print(anova(fit.age, fit.agePLUSsex))
```

```
Analysis of Variance Table

Model 1: sbp ~ age
Model 2: sbp ~ age + sex
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1     67 8260.5
2     66 5202.0  1    3058.5 38.805 3.701e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.7.1.3 Varying Intercept and Varying Slope Model (Age + Sex + Age:Sex)

```
## Make sure sex is a factor (for colors/legend)
sbpdata$sex <- factor(sbpdata$sex)
```

```r
## Fit (interaction: different slopes by sex)
fit.age.TIMES.sex <- lm(sbp ~ age + sex + age:sex, data = sbpdata)

## Age grid and palette
ages <- seq(min(sbpdata$age, na.rm = TRUE),
            max(sbpdata$age, na.rm = TRUE),
            length.out = 200)
lev  <- levels(sbpdata$sex)
cols <- setNames(c("steelblue3", "tomato3", "darkorchid3")[seq_along(lev)], lev)

## Scatter: color points by sex
plot(sbp ~ age, data = sbpdata,
     col = cols[sbpdata$sex], pch = 16,
     xlab = "Age", ylab = "Systolic BP")

## Fitted lines: one per sex (different slopes allowed)
for (sx in lev) {
  nd <- data.frame(age = ages, sex = factor(sx, levels = lev))
  yhat <- predict(fit.age.TIMES.sex, newdata = nd)
  lines(ages, yhat, col = cols[sx], lwd = 2)
}

## Legend
legend("topleft", legend = lev, col = cols[lev], pch = 16, lwd = 2, bty = "n", title = "Sex")
```



Model Matrix and ANOVA

```r
data.frame(model.matrix(fit.age.TIMES.sex))
```

```
   X.Intercept. age sex1 age.sex1
1             1  39    0        0
2             1  45    0        0
3             1  47    0        0
4             1  65    0        0
5             1  46    0        0
6             1  67    0        0
7             1  42    0        0
8             1  67    0        0
9             1  56    0        0
10            1  64    0        0
11            1  56    0        0
12            1  59    0        0
```

```
13            1  34    0         0
14            1  42    0         0
15            1  48    0         0
16            1  45    0         0
17            1  17    0         0
18            1  20    0         0
19            1  19    0         0
20            1  36    0         0
21            1  50    0         0
22            1  39    0         0
23            1  21    0         0
24            1  44    0         0
25            1  53    0         0
26            1  63    0         0
27            1  29    0         0
28            1  25    0         0
29            1  69    0         0
30            1  41    1        41
31            1  60    1        60
32            1  41    1        41
33            1  47    1        47
34            1  66    1        66
35            1  47    1        47
36            1  68    1        68
37            1  43    1        43
38            1  68    1        68
39            1  57    1        57
40            1  65    1        65
41            1  57    1        57
42            1  61    1        61
43            1  36    1        36
44            1  44    1        44
45            1  50    1        50
46            1  47    1        47
47            1  19    1        19
48            1  22    1        22
49            1  21    1        21
50            1  38    1        38
51            1  52    1        52
52            1  41    1        41
53            1  18    1        18
54            1  51    1        51
55            1  55    1        55
56            1  65    1        65
57            1  33    1        33
```

```
58             1  23    1       23
59             1  70    1       70
60             1  56    1       56
61             1  62    1       62
62             1  51    1       51
63             1  48    1       48
64             1  59    1       59
65             1  40    1       40
66             1  35    1       35
67             1  33    1       33
68             1  26    1       26
69             1  61    1       61
```

```
summary(fit.age.TIMES.sex)
```

```
Call:
lm(formula = sbp ~ age + sex + age:sex, data = sbpdata)

Residuals:
    Min      1Q  Median      3Q     Max 
-20.647  -3.410   1.254   4.314  21.153 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 97.07708    5.17046  18.775  < 2e-16 ***
age          0.94932    0.10864   8.738 1.43e-12 ***
sex1        12.96144    7.01172   1.849   0.0691 .  
age:sex1     0.01203    0.14519   0.083   0.9342    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.946 on 65 degrees of freedom
Multiple R-squared:  0.7759,    Adjusted R-squared:  0.7656 
F-statistic: 75.02 on 3 and 65 DF,  p-value: < 2.2e-16
```

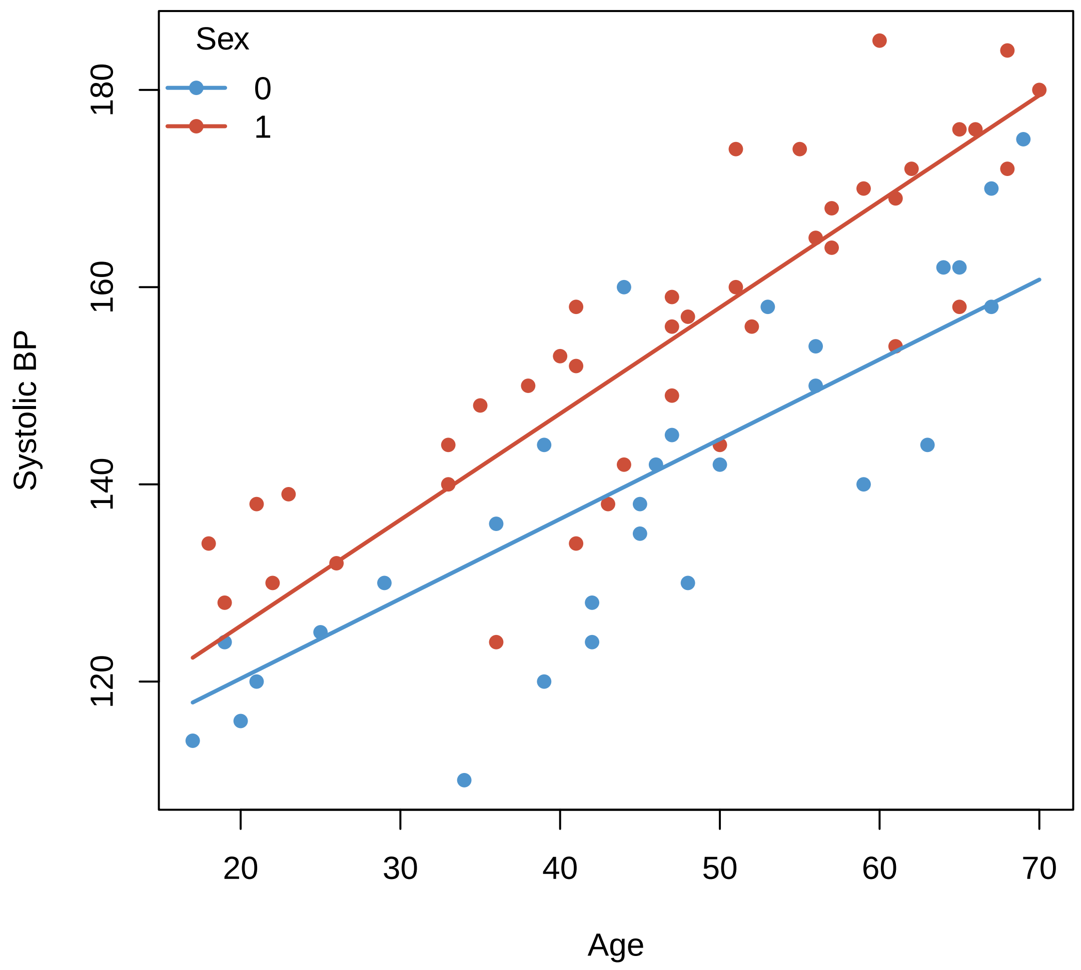
```
print(anova(fit.age,fit.agePLUSsex,fit.age.TIMES.sex))
```

```
Analysis of Variance Table

Model 1: sbp ~ age
Model 2: sbp ~ age + sex
Model 3: sbp ~ age + sex + age:sex
```

```
  Res.Df    RSS Df Sum of Sq       F    Pr(>F)    
1     67 8260.5                                   
2     66 5202.0  1   3058.52 38.2210 4.692e-08 ***
3     65 5201.4  1      0.55  0.0069    0.9342    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.7.1.4 Varying Slope, Equal Intercept Model (Age + Age:Sex)

```r
## Make sure sex is a factor (for colors/legend)
sbpdata$sex <- factor(sbpdata$sex)

## Fit (interaction: different slopes by sex)
fit.equal.intercept <- lm(sbp ~ age + age:sex, data = sbpdata)


## Age grid and palette
ages <- seq(min(sbpdata$age, na.rm = TRUE),
            max(sbpdata$age, na.rm = TRUE),
            length.out = 200)
lev  <- levels(sbpdata$sex)
cols <- setNames(c("steelblue3", "tomato3", "darkorchid3")[seq_along(lev)], lev)

## Scatter: color points by sex
plot(sbp ~ age, data = sbpdata,
     col = cols[sbpdata$sex], pch = 16,
     xlab = "Age", ylab = "Systolic BP")

## Fitted lines: one per sex (different slopes allowed)
for (sx in lev) {
  nd <- data.frame(age = ages, sex = factor(sx, levels = lev))
  yhat <- predict(fit.equal.intercept, newdata = nd)
  lines(ages, yhat, col = cols[sx], lwd = 2)
}

## Legend
legend("topleft", legend = lev, col = cols[lev], pch = 16, lwd = 2, bty = "n", title = "Sex")
```



4.7.2 Orders of Terms Matters in ANOVA and Warnings in Interpreting t-test Tables

```
fit.int <- lm(sbp ~ 1, data = sbpdata)
fit.sex <- lm(sbp ~ sex, data = sbpdata)

print(anova(fit.int,fit.age,fit.agePLUSsex, fit.age.TIMES.sex))
```

```
Analysis of Variance Table

Model 1: sbp ~ 1
Model 2: sbp ~ age
Model 3: sbp ~ age + sex
Model 4: sbp ~ age + sex + age:sex
  Res.Df     RSS Df Sum of Sq        F    Pr(>F)
1     68 23211.8
```

```
2     67  8260.5  1   14951.3 186.8390 < 2.2e-16 ***
3     66  5202.0  1    3058.5  38.2210 4.692e-08 ***
4     65  5201.4  1       0.5   0.0069    0.9342
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
print(anova(fit.int,fit.age,fit.equal.intercept, fit.age.TIMES.sex))
```

```
Analysis of Variance Table

Model 1: sbp ~ 1
Model 2: sbp ~ age
Model 3: sbp ~ age + age:sex
Model 4: sbp ~ age + sex + age:sex
  Res.Df     RSS Df Sum of Sq        F    Pr(>F)
1     68 23211.8
2     67  8260.5  1   14951.3 186.8390 < 2.2e-16 ***
3     66  5474.9  1    2785.6  34.8107 1.437e-07 ***
4     65  5201.4  1     273.4   3.4171   0.06907 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
print(anova(fit.int,fit.sex,fit.agePLUSsex, fit.age.TIMES.sex))
```

```
Analysis of Variance Table

Model 1: sbp ~ 1
Model 2: sbp ~ sex
Model 3: sbp ~ age + sex
Model 4: sbp ~ age + sex + age:sex
  Res.Df     RSS Df Sum of Sq        F    Pr(>F)
1     68 23211.8
2     67 19282.5  1    3929.2  49.1017 1.684e-09 ***
3     66  5202.0  1   14080.6 175.9583 < 2.2e-16 ***
4     65  5201.4  1       0.5   0.0069    0.9342
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(fit.age)
```

```
Call:
```

```
lm(formula = sbp ~ age, data = sbpdata)

Residuals:
    Min      1Q  Median      3Q     Max 
-26.782  -7.632   1.968   8.201  22.651 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 103.34905    4.33190   23.86   <2e-16 ***
age           0.98333    0.08929   11.01   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 11.1 on 67 degrees of freedom
Multiple R-squared:  0.6441,	Adjusted R-squared:  0.6388 
F-statistic: 121.3 on 1 and 67 DF,  p-value: < 2.2e-16
```

```
summary(fit.equal.intercept)
```

```
Call:
lm(formula = sbp ~ age + age:sex, data = sbpdata)

Residuals:
     Min       1Q   Median       3Q      Max 
-21.6338  -4.3067   0.9922   4.9819  20.2753 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 104.12501    3.55578  29.283  < 2e-16 ***
age           0.80908    0.07918  10.219 3.14e-15 ***
age:sex1      0.26705    0.04608   5.795 2.09e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.108 on 66 degrees of freedom
Multiple R-squared:  0.7641,	Adjusted R-squared:  0.757 
F-statistic: 106.9 on 2 and 66 DF,  p-value: < 2.2e-16
```

```
summary(fit.agePLUSsex)
```

```
Call:
```

```
lm(formula = sbp ~ age + sex, data = sbpdata)

Residuals:
    Min      1Q  Median      3Q     Max 
-20.705  -3.299   1.248   4.325  21.160 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 96.77353    3.62085  26.727  < 2e-16 ***
age          0.95606    0.07153  13.366  < 2e-16 ***
sex1        13.51345    2.16932   6.229  3.7e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.878 on 66 degrees of freedom
Multiple R-squared:  0.7759,    Adjusted R-squared:  0.7691 
F-statistic: 114.2 on 2 and 66 DF,  p-value: < 2.2e-16
```

```r
summary(fit.age.TIMES.sex)
```

```
Call:
lm(formula = sbp ~ age + sex + age:sex, data = sbpdata)

Residuals:
    Min      1Q  Median      3Q     Max 
-20.647  -3.410   1.254   4.314  21.153 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 97.07708    5.17046  18.775  < 2e-16 ***
age          0.94932    0.10864   8.738 1.43e-12 ***
sex1        12.96144    7.01172   1.849   0.0691 .  
age:sex1     0.01203    0.14519   0.083   0.9342    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.946 on 65 degrees of freedom
Multiple R-squared:  0.7759,    Adjusted R-squared:  0.7656 
F-statistic: 75.02 on 3 and 65 DF,  p-value: < 2.2e-16
```

4.8 Model Building

```r
library(olsrr)
## Note: Update this path to your local file location
wine <- read.csv("wine.csv")

model.wine <- lm(quality ~ ., data = wine)
```

4.8.1 All Possible Regression

```r
ols_step_best_subset(model.wine)
```

```
             Best Subsets Regression
-------------------------------------------------
Model Index    Predictors
-------------------------------------------------
     1         flavor
     2         flavor oakiness
     3         aroma flavor oakiness
     4         clarity aroma flavor oakiness
     5         clarity aroma body flavor oakiness
-------------------------------------------------

                                                  Subsets Regression Summary
----------------------------------------------------------------------------------------------------
                       Adj.        Pred
Model    R-Square    R-Square    R-Square     C(p)        AIC        SBIC        SBC         MSE
----------------------------------------------------------------------------------------------------
  1        0.6242      0.6137      0.5868    9.0436    130.0214    21.6859    134.9341    61.4
  2        0.6611      0.6417      0.6058    6.8132    128.0901    20.1242    134.6404    57.0
  3        0.7038      0.6776      0.6379    3.9278    124.9781    18.0702    133.1661    51.3
  4        0.7147      0.6801      0.6102    4.6747    125.5480    19.2854    135.3736    50.9
  5        0.7206      0.6769       0.587    6.0000    126.7552    21.0956    138.2183    51.5
----------------------------------------------------------------------------------------------------
AIC: Akaike Information Criteria
 SBIC: Sawa's Bayesian Information Criteria
 SBC: Schwarz Bayesian Criteria
 MSEP: Estimated error of prediction, assuming multivariate normality
 FPE: Final Prediction Error
 HSP: Hocking's Sp
 APC: Amemiya Prediction Criteria
```

4.8.2 Automated Stepwise Procedures

```r
## Backward Elimination (alpha_out = 0.1)
ols_step_backward_p(model.wine, p_val = 0.1)
```

```
                             Stepwise Summary
------------------------------------------------------------------------
Step    Variable        AIC        SBC        SBIC       R2        Adj. R2
------------------------------------------------------------------------
 0      Full Model    126.755    138.218    21.096    0.72060    0.67694
 1      body          125.548    135.374    19.285    0.71471    0.68013
 2      clarity       124.978    133.166    18.070    0.70377    0.67763
------------------------------------------------------------------------

Final Model Output
------------------

                         Model Summary
---------------------------------------------------------------
R                        0.839       RMSE                1.098
R-Squared                0.704       MSE                 1.207
Adj. R-Squared           0.678       Coef. Var           9.338
Pred R-Squared           0.638       AIC               124.978
MAE                      0.868       SBC               133.166
---------------------------------------------------------------
 RMSE: Root Mean Square Error
 MSE: Mean Square Error
 MAE: Mean Absolute Error
 AIC: Akaike Information Criteria
 SBC: Schwarz Bayesian Criteria

                             ANOVA
-------------------------------------------------------------------
               Sum of
              Squares        DF    Mean Square      F          Sig.
-------------------------------------------------------------------
Regression    108.935         3         36.312    26.925     0.0000
Residual       45.853        34          1.349
Total         154.788        37
-------------------------------------------------------------------

                                  Parameter Estimates
```

```
-----------------------------------------------------------------------------------------
      model       Beta    Std. Error    Std. Beta       t        Sig      lower     upper
-----------------------------------------------------------------------------------------
(Intercept)      6.467         1.333                 4.852     0.000      3.759     9.176
      aroma      0.580         0.262        0.307    2.213     0.034      0.047     1.113
     flavor      1.200         0.275        0.603    4.364     0.000      0.641     1.758
   oakiness     -0.602         0.264       -0.217   -2.278     0.029     -1.140    -0.065
-----------------------------------------------------------------------------------------
```

```r
## Forward Selection (alpha_in = 0.1)
ols_step_forward_p(model.wine, p_val = 0.1)
```

```
                             Stepwise Summary
-------------------------------------------------------------------------
Step    Variable        AIC        SBC       SBIC       R2       Adj. R2
-------------------------------------------------------------------------
 0      Base Model    165.209    168.484    55.141    0.00000    0.00000
 1      flavor        130.021    134.934    21.686    0.62417    0.61373
 2      oakiness      128.090    134.640    20.124    0.66111    0.64175
 3      aroma         124.978    133.166    18.070    0.70377    0.67763
-------------------------------------------------------------------------

Final Model Output
------------------

                         Model Summary
----------------------------------------------------------------
R                       0.839       RMSE                 1.098
R-Squared               0.704       MSE                  1.207
Adj. R-Squared          0.678       Coef. Var            9.338
Pred R-Squared          0.638       AIC                124.978
MAE                     0.868       SBC                133.166
----------------------------------------------------------------
 RMSE: Root Mean Square Error
 MSE: Mean Square Error
 MAE: Mean Absolute Error
 AIC: Akaike Information Criteria
 SBC: Schwarz Bayesian Criteria

                               ANOVA
-------------------------------------------------------------------
              Sum of
             Squares        DF     Mean Square      F         Sig.
```

```
-------------------------------------------------------------------
Regression     108.935          3          36.312     26.925     0.0000
Residual        45.853         34           1.349
Total          154.788         37
-------------------------------------------------------------------

                              Parameter Estimates
----------------------------------------------------------------------------------------
      model        Beta    Std. Error    Std. Beta       t        Sig       lower     upper
----------------------------------------------------------------------------------------
(Intercept)       6.467         1.333                 4.852     0.000     3.759     9.176
     flavor       1.200         0.275        0.603    4.364     0.000     0.641     1.758
   oakiness      -0.602         0.264       -0.217   -2.278     0.029    -1.140    -0.065
      aroma       0.580         0.262        0.307    2.213     0.034     0.047     1.113
----------------------------------------------------------------------------------------
```

```
## Stepwise Regression (alpha_in = 0.1, alpha_out = 0.1)
ols_step_both_p(model.wine, p_enter = 0.1, p_remove = 0.1)
```

```
                              Stepwise Summary
--------------------------------------------------------------------------
Step    Variable          AIC        SBC        SBIC       R2         Adj. R2
--------------------------------------------------------------------------
 0      Base Model      165.209    168.484    55.141    0.00000    0.00000
 1      flavor (+)      130.021    134.934    21.686    0.62417    0.61373
 2      oakiness (+)    128.090    134.640    20.124    0.66111    0.64175
 3      aroma (+)       124.978    133.166    18.070    0.70377    0.67763
--------------------------------------------------------------------------

Final Model Output
------------------

                         Model Summary
---------------------------------------------------------------
R                        0.839       RMSE                1.098
R-Squared                0.704       MSE                 1.207
Adj. R-Squared           0.678       Coef. Var           9.338
Pred R-Squared           0.638       AIC               124.978
MAE                      0.868       SBC               133.166
---------------------------------------------------------------
 RMSE: Root Mean Square Error
 MSE: Mean Square Error
 MAE: Mean Absolute Error
 AIC: Akaike Information Criteria
 SBC: Schwarz Bayesian Criteria

                               ANOVA
-------------------------------------------------------------------
              Sum of
             Squares        DF    Mean Square       F          Sig.
-------------------------------------------------------------------
Regression    108.935        3         36.312    26.925     0.0000
Residual       45.853       34          1.349
Total         154.788       37
-------------------------------------------------------------------

                                  Parameter Estimates
----------------------------------------------------------------------------------------
      model      Beta    Std. Error    Std. Beta      t        Sig      lower      upper
----------------------------------------------------------------------------------------
(Intercept)     6.467         1.333                 4.852    0.000      3.759      9.176
     flavor     1.200         0.275        0.603    4.364    0.000      0.641      1.758
   oakiness    -0.602         0.264       -0.217   -2.278    0.029     -1.140     -0.065
      aroma     0.580         0.262        0.307    2.213    0.034      0.047      1.113
----------------------------------------------------------------------------------------
```

4.9 Multicollinearity

4.9.1 A Simple Example

```r
y <- c(19, 20, 37, 39, 36, 38)
x1 <- c(4, 4, 7, 7, 7.1, 7.1)
x2 <- c(16, 16, 49, 49, 50.4, 50.4)
cor(data.frame(x1, x2))
```

```
          x1        x2
x1 1.0000000 0.9999713
x2 0.9999713 1.0000000
```

```r
fit_multi <- lm(y ~ x1 + x2)
summary(fit_multi)
```

```
Call:
```

```
lm(formula = y ~ x1 + x2)

Residuals:
   1    2    3    4    5    6 
-0.5  0.5 -1.0  1.0 -1.0  1.0 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -156.056    117.158  -1.332    0.275
x1            65.444     45.890   1.426    0.249
x2            -5.389      4.152  -1.298    0.285

Residual standard error: 1.225 on 3 degrees of freedom
Multiple R-squared:  0.9897,    Adjusted R-squared:  0.9829 
F-statistic: 144.3 on 2 and 3 DF,  p-value: 0.001043
```

```
fit1_multi <- lm(y ~ x1)
summary(fit1_multi)
```

```
Call:
lm(formula = y ~ x1)

Residuals:
      1       2       3       4       5       6 
-0.5260  0.4740 -0.1925  1.8075 -1.7814  0.2186 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  -4.0293     2.3332  -1.727    0.159    
x1            5.8888     0.3762  15.654 9.73e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.325 on 4 degrees of freedom
Multiple R-squared:  0.9839,    Adjusted R-squared:  0.9799 
F-statistic: 245.1 on 1 and 4 DF,  p-value: 9.725e-05
```

```
ols_vif_tol(fit_multi)
```

```
  Variables    Tolerance      VIF
1        x1 5.738191e-05 17427.09
2        x2 5.738191e-05 17427.09
```

4.9.2 VIFs in the Wine Quality Data

```
wine.x <- wine[, -ncol(wine)] # Assuming quality is the last column
cor(wine.x)
```

```
              clarity     aroma       body      flavor  oakiness
clarity    1.00000000 0.0619021 -0.3083783 -0.08515993 0.1832147
aroma      0.06190210 1.0000000  0.5489102  0.73656121 0.2016444
body      -0.30837826 0.5489102  1.0000000  0.64665917 0.1521059
flavor    -0.08515993 0.7365612  0.6466592  1.00000000 0.1797605
oakiness   0.18321471 0.2016444  0.1521059  0.17976051 1.0000000
```

```
## VIF using olsrr (data frame output)
ols_vif_tol(model.wine)
```

```
  Variables Tolerance      VIF
1   clarity 0.7896462 1.266390
2     aroma 0.4199665 2.381143
3      body 0.4862649 2.056492
4    flavor 0.3728175 2.682277
5  oakiness 0.9118005 1.096731
```

4.9.3 VIFs in the Children Height Data

```
## Data: Weight, height and age of children
wgt <- c(64, 71, 53, 67, 55, 58, 77, 57, 56, 51, 76, 68)
hgt <- c(57, 59, 49, 62, 51, 50, 55, 48, 42, 42, 61, 57)
age <- c(8, 10, 6, 11, 8, 7, 10, 9, 10, 6, 12, 9)

fit_age_hgt <- lm(wgt ~ hgt + age, data = child.data)
ols_vif_tol(fit_age_hgt)
```

```
  Variables Tolerance      VIF
1       hgt 0.6232021 1.604616
2       age 0.6232021 1.604616
```

5 Logistic Regression

5.1 Odds as a Function of Probability

For an event with probability p, the odds is

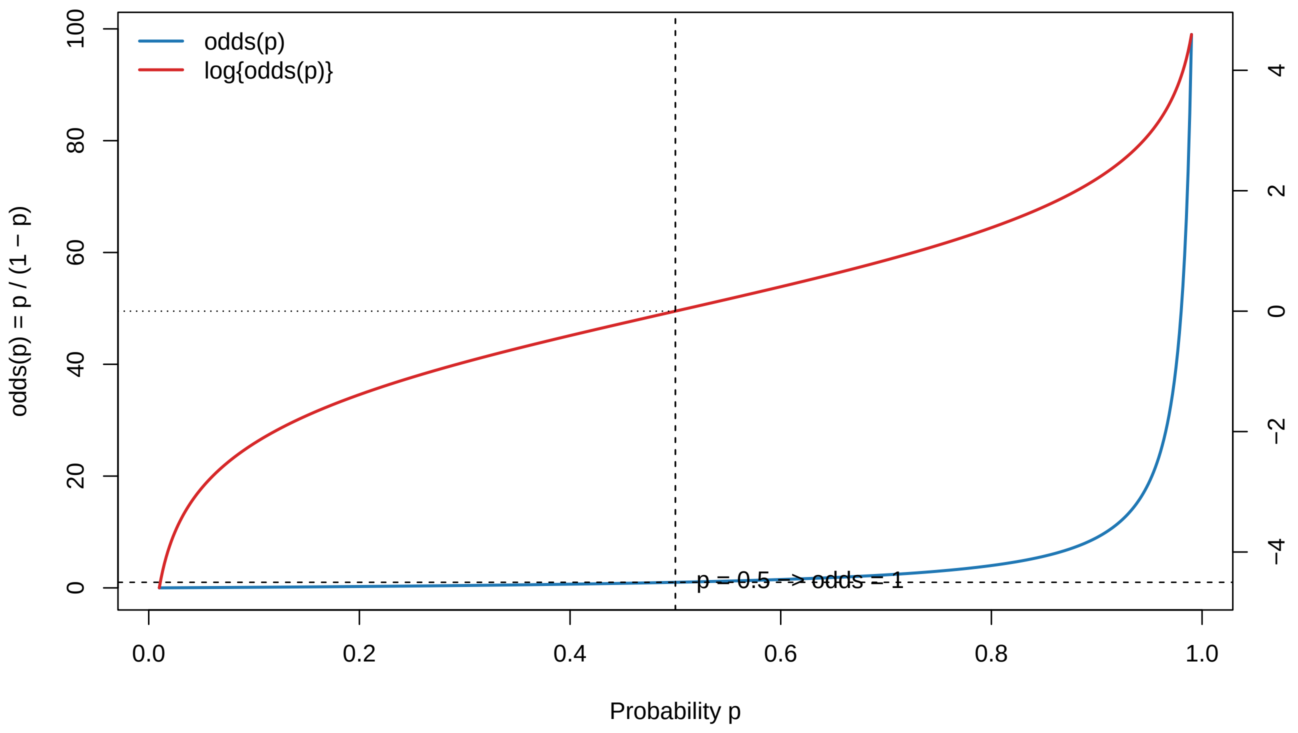
$$\text{odds}(p) = \frac{p}{1-p}$$

and the log-odds (logit) is

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$$

.

```r
## Plot odds(p) with a right-hand axis for log(odds(p)),
## using different line colors for the two curves.
## Defaults: p in [0.01, 0.99].
## Args:
##   p_min, p_max : endpoints for p-grid (0<p_min<p_max<1)
##   n            : number of grid points
##   annotate     : add reference lines/labels if TRUE
##   odds_col     : color for odds(p)
##   logit_col    : color for log(odds(p))
##   lwd1, lwd2   : line widths for the two curves


plot_odds <- function(p_min = 0.01, p_max = 0.99, n = 400,
                      annotate = TRUE,
                      odds_col = "steelblue",
                      logit_col = "firebrick",
                      lwd1 = 2, lwd2 = 2) {
  stopifnot(p_min > 0, p_max < 1, p_min < p_max, n >= 10)
  p <- seq(p_min, p_max, length.out = n)
  odds <- p / (1 - p)
  logit <- log(odds)

  ## Left y-axis: odds(p)
  plot(p, odds, type = "l", lwd = lwd1, col = odds_col,
       xlab = "Probability p",
       ylab = "odds(p) = p / (1 - p)")
  if (annotate) {
    abline(h = 1, v = 0.5, lty = 2)
    text(0.52, 1.05, "p = 0.5 → odds = 1", adj = 0)
  }

  ## Right y-axis: logit(p) = log(odds)
  op <- par(new = TRUE)
  on.exit(par(op), add = TRUE)
  plot(p, logit, type = "l", lwd = lwd2, col = logit_col,
       axes = FALSE, xlab = "", ylab = "")
  axis(4)
  mtext("log{odds(p)} = log{p/(1 - p)}", side = 4, line = 3)

  if (annotate) {
    abline(v = 0.5, lty = 2)
    # logit(0.5) = 0 reference (horizontal) on the right-axis scale
    usr <- par("usr")
    segments(x0 = usr[1], y0 = 0, x1 = 0.5, y1 = 0, lty = 3)
  }

  legend("topleft",
         legend = c("odds(p)", "log{odds(p)}"),
         col = c(odds_col, logit_col),
         lwd = c(lwd1, lwd2), bty = "n")

  invisible(list(p = p, odds = odds, logit = logit))
}

## Example usage:
## plot_odds()  # defaults: steelblue for odds, firebrick for log-odds (right axis)
plot_odds(odds_col = "#1f77b4", logit_col = "#d62728", n = 600)
```



Logistic regression models **log-odds** linearly in predictors, which both keeps fitted probabilities in $(0, 1)$ and turns multiplicative effects on odds into **additive** effects on the linear predictor.

5.2 A Simulated Data

We simulate data from a logistic model where the **logit** is a linear function of x:

$$\operatorname{logit} p(x) = \log\left(\frac{p(x)}{1 - p(x)}\right) = \beta_0 + \beta_1 x,$$

so that

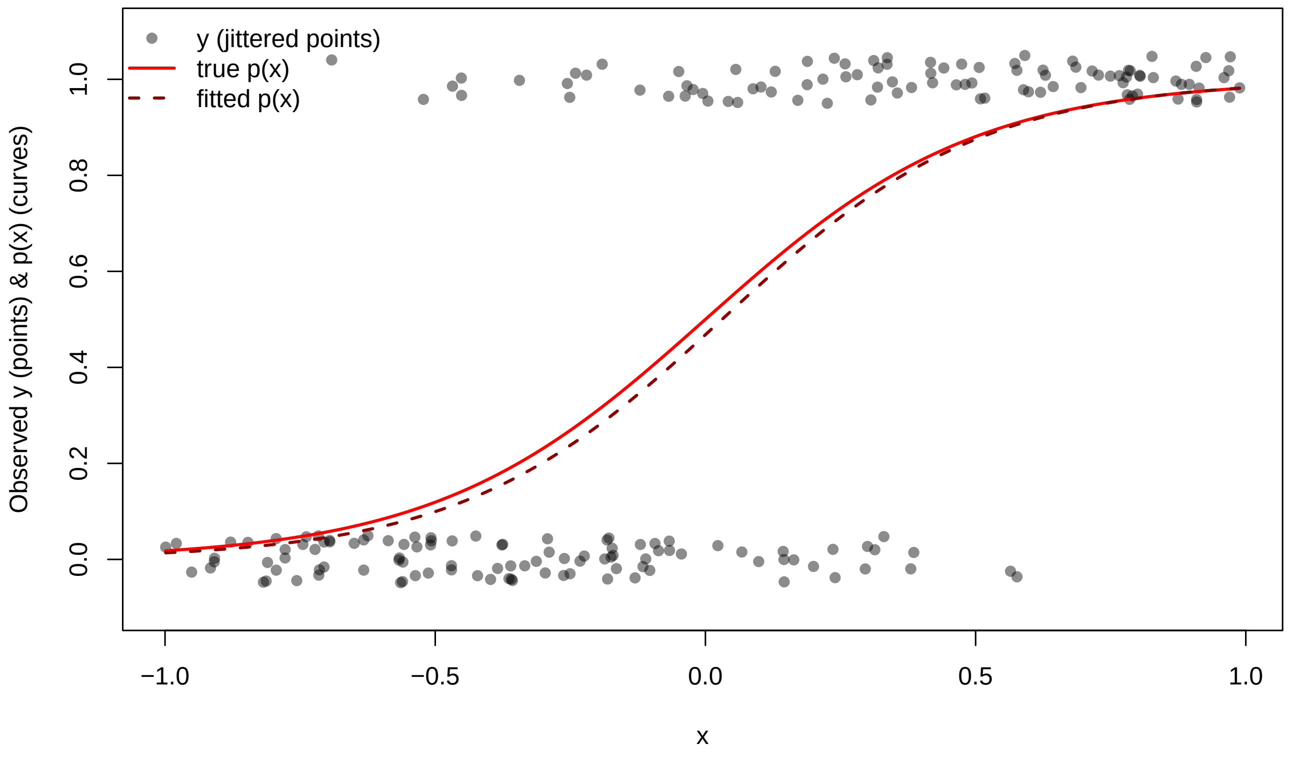
$$p(x) = \operatorname{logit}^{-1}(\beta_0 + \beta_1 x) = \frac{1}{1 + \exp -(\beta_0 + \beta_1 x)}.$$

We then display the observed y_i (binary outcomes) and the true probability curve $p(x)$ in red.

```r
set.seed(123)

## -- Truth (edit as desired) --
n     <- 200
beta0 <- 0
beta1 <-  4

## -- Simulate --
x   <- runif(n, -1, 1)             # predictor
eta <- beta0 + beta1 * x
p   <- plogis(eta)                 # true p(x)
y   <- rbinom(n, size = 1, prob = p) # outcomes

sim.data <- data.frame(x = x, y = y, p = p)
```

5.2.1 Fit a logistic model to the simulated data

```r
## -- Optional: fit a model to the simulated data --
sim.fit <- glm(y ~ x, data = sim.data, family = binomial())
p_fit <- predict(sim.fit, newdata = data.frame(x = x), type = "response")

## -- Plot: points for y_i (jittered), red line for true p(x) --
## Define jitter amount
jit <- 0.05
## jitter to separate 0/1 visually
yj <- jitter(sim.data$y, amount = jit)

plot(sim.data$x, yj,
     pch = 16, col = rgb(0, 0, 0, 0.45),
     xlab = "x",
     ylab = "Observed y (points) & p(x) (curves)",
     ylim = c(-0.1, 1.1))

## True probability curve (red)
xg <- seq(min(x), max(x), length.out = 500)
lines(xg, plogis(beta0 + beta1 * xg), col = "red", lwd = 2)

## Optional: add fitted probability curve (dashed dark red)
lines(xg, predict(sim.fit, newdata = data.frame(x = xg), type = "response"),
      col = "darkred", lwd = 2, lty = 2)

legend("topleft",
       legend = c("y (jittered points)", "true p(x)", "fitted p(x)"),
       pch    = c(16, NA, NA),
       lty    = c(NA, 1, 2),
       col    = c(rgb(0,0,0,0.45), "red", "darkred"),
       lwd    = c(NA, 2, 2),
       bty    = "n")
```



5.3 Example of Coronary Heart Disease Data

5.3.1 Load a dataset

This dataset is about a follow-up study to determine the development of coronary heart disease (CHD) over 9 years of follow-up of 609 white males from Evans County, Georgia.

Variable meanings (as provided):

- `chd`: 1 if a person has the disease, 0 otherwise.
- `smk`: 1 if smoker, 0 if not.
- `cat`: 1 if catecholamine level is high, 0 if low.
- `sbp`: systolic blood pressure (continuous).
- `age`: age in years (continuous).
- `chl`: cholesterol level (continuous).

- `ecg`: 1 if electrocardiogram is abnormal, 0 if normal.
- `hpt`: 1 if high blood pressure, 0 if normal.

```
## Adjust the path if needed. The default is your original V: drive path.
data_path <- "evans.dat"

## Read data (expects a header row)
CHD.data <- read.table(data_path, header = TRUE)

CHD.data
```

```
      id chd age cat chl dbp ecg sbp smk hpt
1     21   0  56   0 270  80   0 138   0   0
2     31   0  43   0 159  74   0 128   1   0
3     51   1  56   1 201 112   1 164   1   1
4     71   0  64   1 179 100   0 200   1   1
5     74   0  49   0 243  82   0 145   1   0
6     91   0  46   0 252  88   0 142   1   0
7    111   1  52   0 179  80   1 128   1   0
8    131   0  63   0 217  92   0 135   0   0
9    141   0  42   0 176  76   0 114   1   0
10   191   0  55   0 250 114   1 182   0   1
11   201   0  74   0 293 100   0 166   0   1
12   241   0  53   0 179  90   0 158   0   0
13   251   0  58   0 201  86   0 142   1   0
14   261   0  56   0 206  85   0 120   1   0
15   271   0  69   0 225  84   0 168   0   1
16   283   1  51   1 259 102   1 135   0   1
17   291   0  43   0 193  78   0 118   1   0
18   311   0  64   1 185 100   1 180   0   1
19   312   0  44   0 150 108   0 160   0   1
20   331   0  42   0 211  86   1 122   0   0
21   351   0  57   0 216  88   0 130   0   0
22   381   1  64   1 247  75   1 130   0   0
23   401   0  49   0 200  82   0 130   0   0
24   411   0  68   1 205  74   0 152   1   0
25   431   0  41   0 225  98   0 135   1   1
26   441   0  64   0 263  98   0 162   1   1
27   451   0  41   0 205  80   0 120   0   0
28   481   0  59   0 253  98   0 154   0   1
29   501   0  50   0 282  90   0 142   1   0
30   521   0  56   0 230  80   0 118   0   0
31   541   0  57   1 203 112   0 182   0   1
32   561   0  42   0 211  86   0 144   0   0
33   571   0  59   0 234  84   0 164   1   1
```

34	581	0	44	0	202	94	1	174	1	1
35	611	0	52	0	162	78	0	134	1	0
36	621	0	45	0	191	85	0	135	0	0
37	641	0	41	0	220	110	0	178	0	1
38	651	0	59	0	240	80	0	130	0	0
39	671	0	52	0	189	110	0	168	0	1
40	681	0	64	0	247	102	0	170	0	1
41	731	0	46	0	181	122	1	176	1	1
42	741	0	42	0	168	75	0	104	1	0
43	751	0	54	0	187	86	0	146	1	0
44	761	0	48	0	196	98	0	130	0	1
45	811	0	45	0	155	70	0	142	1	0
46	851	0	66	1	173	100	0	160	1	1
47	861	0	41	0	138	70	0	115	1	0
48	871	0	76	0	269	94	0	175	1	1
49	881	1	49	0	266	102	0	152	1	1
50	921	0	57	1	200	100	0	160	1	1
51	941	0	51	0	188	84	0	124	1	0
52	961	1	43	0	218	108	1	136	1	1
53	971	0	43	0	212	80	1	108	1	0
54	981	0	45	0	212	102	0	150	1	1
55	991	0	45	0	180	80	0	122	1	0
56	1061	1	46	1	166	76	1	162	0	1
57	1071	0	40	0	257	84	0	130	0	0
58	1081	0	48	0	243	82	1	154	1	0
59	1091	0	64	1	179	100	1	148	1	1
60	1111	0	70	0	167	64	0	112	1	0
61	1151	0	52	0	178	84	1	112	1	0
62	1171	0	55	0	178	94	0	152	0	0
63	1181	0	49	0	211	68	0	114	1	0
64	1191	1	56	0	171	85	0	125	1	0
65	1201	1	66	1	205	80	0	150	1	0
66	1221	0	48	0	229	130	0	195	1	1
67	1231	0	47	0	238	120	1	160	1	1
68	1471	0	54	1	195	112	0	174	1	1
69	1501	0	44	0	162	82	0	120	0	0
70	1561	0	51	0	240	84	1	126	1	0
71	1691	0	43	0	177	102	1	138	1	1
72	1701	0	68	0	252	88	1	112	1	0
73	1741	0	49	0	217	105	0	148	0	1
74	1751	0	55	0	263	84	0	114	0	0
75	1761	0	51	0	229	100	0	162	1	1
76	1791	0	50	0	245	96	0	144	0	1
77	1811	0	65	0	177	74	0	122	0	0
78	1821	0	42	0	203	78	0	134	1	0

```
79    1851   0  57   0 194  75   1 114   0   0
80    1881   0  42   0 288 110   0 142   0   1
81    1891   0  53   0 217  70   0 120   1   0
82    1901   0  57   1 163  94   0 184   0   1
83    1911   0  61   0 180  84   0 136   0   0
84    1951   0  53   0 209  98   0 142   1   1
85    1961   0  45   0 200  80   0 135   0   0
86    1971   0  44   0 194  80   0 120   1   0
87    2241   0  63   0 227  90   1 135   0   0
88    2252   0  42   0 158  92   0 135   1   0
89    2273   0  73   1 183 120   1 220   0   1
90    2281   0  47   0 253 110   0 140   1   1
91    2311   0  56   0 198  88   0 122   1   0
92    2371   1  41   0 228 132   0 162   1   1
93    2381   0  58   0 217  86   0 140   0   0
94    2391   0  55   0 163  70   0 110   1   0
95    2401   0  46   0 212 124   0 184   1   1
96    2461   0  57   0 144  95   0 130   0   1
97    2481   0  44   0 134  74   0 114   1   0
98    2501   0  52   1 183  96   0 158   1   1
99    2511   0  56   0 212 108   0 144   0   1
100   2531   0  64   0 214  82   0 128   1   0
101   2541   0  54   0 249  92   0 120   1   0
102   2571   0  52   0 180  78   1 104   1   0
103   2591   0  42   0 212  92   0 125   1   0
104   2611   0  46   0 167  82   0 120   1   0
105   2621   0  46   0 273  94   0 152   0   0
106   2631   0  42   0 210  96   0 134   1   1
107   2641   0  54   1 173 110   0 170   1   1
108   2671   0  43   0 256  72   0 114   1   0
109   2681   0  53   0 234  80   0 122   0   0
110   2691   1  40   0 221 100   0 140   1   1
111   2711   0  46   0 261  86   0 128   1   0
112   2731   0  43   0 299  80   0 116   0   0
113   2851   0  43   0 192  75   0 115   1   0
114   2861   0  47   0 185  80   1 146   1   0
115   2871   0  44   0 283  70   0 108   1   0
116   2881   0  49   0 176  92   0 134   1   0
117   2891   1  56   1 331 110   0 190   1   1
118   2901   1  56   0 203  82   0 120   1   0
119   2911   0  64   1 217  92   0 166   1   1
120   2921   0  54   0 164  72   0 122   1   0
121   2931   0  54   0 256  98   0 148   0   1
122   2991   0  51   0 184  98   0 170   0   1
123   3001   0  49   0 165  80   0 114   1   0
```

124	3011	0	47	0	189	92	0	145	0	0
125	3031	0	58	0	221	88	0	140	1	0
126	3061	0	70	1	126	66	1	164	1	1
127	3601	0	42	0	169	80	1	122	1	0
128	3611	0	59	0	266	92	0	138	0	0
129	3621	0	57	1	153	92	0	148	1	0
130	3651	0	76	1	211	114	1	228	1	1
131	3661	0	43	0	113	76	0	114	1	0
132	3701	0	46	0	200	85	0	145	1	0
133	3721	0	75	1	172	114	1	162	1	1
134	3751	0	42	0	131	84	0	130	0	0
135	3761	0	64	0	214	84	0	120	0	0
136	3771	0	63	1	236	94	1	190	0	1
137	3791	0	54	0	213	90	0	142	0	0
138	3811	0	66	0	226	90	0	166	0	1
139	3813	0	44	0	200	110	0	160	1	1
140	3841	0	72	0	188	78	0	130	0	0
141	3861	0	50	0	268	102	0	138	0	1
142	3871	0	59	1	195	114	1	208	0	1
143	3881	1	59	0	216	95	0	140	1	1
144	3891	0	53	0	182	92	0	130	1	0
145	3901	0	48	0	178	95	0	135	1	1
146	3911	0	40	0	191	76	0	152	1	0
147	3941	0	61	0	255	80	0	120	0	0
148	3951	0	42	0	225	80	0	126	1	0
149	4161	0	42	0	166	90	0	145	0	0
150	4191	0	49	0	278	84	0	126	1	0
151	4202	0	40	0	235	72	0	116	0	0
152	4221	0	51	0	251	86	0	128	1	0
153	4242	0	44	0	217	90	0	146	0	0
154	4261	0	44	0	181	94	0	144	1	0
155	4271	0	47	0	208	108	0	178	0	1
156	4291	0	51	0	182	112	0	182	0	1
157	4301	0	69	0	228	75	0	115	1	0
158	4321	0	58	1	170	88	1	152	1	0
159	4331	0	74	1	147	80	1	200	0	1
160	4341	0	48	0	190	78	0	114	1	0
161	4381	0	64	0	205	98	1	140	0	1
162	4401	0	53	0	216	78	0	124	1	0
163	4411	0	71	0	170	90	0	140	1	0
164	4421	0	47	0	127	74	0	110	1	0
165	4451	0	56	0	235	92	0	128	1	0
166	4461	0	40	0	200	72	0	118	0	0
167	4491	0	46	0	283	100	0	148	1	1
168	4531	0	68	1	157	94	0	162	0	1

```
169  4551   1  54   0 206  76   1 142   0   0
170  4581   0  54   0 197  88   0 125   1   0
171  4591   0  45   0 163  75   0 115   1   0
172  4601   0  66   0 176  60   1 124   0   0
173  4641   0  58   0 211  88   0 146   1   0
174  4681   0  49   0 161  75   0 115   0   0
175  4711   0  51   0 244  90   0 128   0   0
176  4731   0  44   0 172 100   0 138   0   1
177  4751   0  61   1 166  86   0 156   1   0
178  4771   0  48   0 184  76   0 116   1   0
179  4781   0  63   0 143  92   0 122   1   0
180  4791   0  54   0 196  84   0 138   1   0
181  4801   0  52   0 189  88   0 142   1   0
182  4811   0  45   0 227  98   1 140   1   1
183  4821   0  62   0 236  94   0 160   0   1
184  4831   0  41   0 240  86   0 144   0   0
185  4851   0  41   0 256  90   0 145   1   0
186  4861   0  61   0 200  84   0 148   1   0
187  4871   0  42   0 199 104   0 166   1   1
188  4901   0  42   0 161  88   0 124   0   0
189  4911   0  72   0 211  80   1 104   0   0
190  4951   0  43   0 180  64   0  92   0   0
191  4961   1  72   0 200  86   1 138   0   0
192  4971   0  51   0 206  80   0 132   1   0
193  4981   0  58   0 254  94   0 152   1   0
194  5011   0  41   0 215  90   0 142   1   0
195  5061   0  71   1 162  98   1 184   1   1
196  5071   1  63   0 145  96   0 162   1   1
197  5091   0  44   0 220  90   1 130   1   0
198  5101   0  45   0 298 108   0 170   1   1
199  5111   0  54   0 300  94   0 148   1   0
200  5131   1  52   1 306 108   0 178   1   1
201  5141   0  55   0 302 134   1 206   1   1
202  5181   1  41   0 158  80   0 140   1   0
203  5191   0  54   0 194 130   1 170   1   1
204  5211   0  64   1 229  94   1 156   1   0
205  5251   0  61   0 259  82   0 118   0   0
206  5281   0  40   0 214  94   0 130   0   0
207  5301   0  51   0 168 106   0 156   1   1
208  5361   0  51   0 265  90   0 158   1   0
209  5391   0  75   0 225  80   0 125   0   0
210  5421   1  40   0 219  80   0 115   1   0
211  5451   1  63   0 202 110   0 160   0   1
212  5461   0  42   1 217  94   1 138   0   0
213  5471   1  64   0 231  85   0 120   1   0
```

214	5521	1	50	0	215	114	0	170	1	1
215	5601	0	49	0	146	98	1	145	1	1
216	5621	0	48	0	198	75	0	120	1	0
217	5631	0	58	0	206	92	0	154	0	0
218	5641	0	46	0	227	98	0	168	1	1
219	5671	0	46	0	214	92	1	166	1	1
220	6341	0	42	0	225	100	1	162	1	1
221	6351	0	57	0	193	86	0	124	0	0
222	6371	0	50	0	186	102	0	160	0	1
223	6391	0	46	0	147	85	0	122	1	0
224	6411	0	45	0	205	100	0	166	0	1
225	6421	0	57	1	196	98	1	196	1	1
226	6441	0	46	0	195	96	0	138	0	1
227	6451	0	45	1	153	108	1	212	1	1
228	6461	0	58	1	172	96	1	168	0	1
229	6482	0	42	0	293	110	0	176	1	1
230	6491	0	53	0	274	106	0	158	1	1
231	6501	0	55	0	221	106	0	162	0	1
232	6511	0	53	0	197	70	0	112	0	0
233	6531	0	69	1	194	100	1	150	0	1
234	6551	0	58	0	204	74	0	122	1	0
235	6561	0	46	0	203	84	0	114	1	0
236	6591	0	62	0	293	90	1	142	1	0
237	6631	0	61	0	197	72	1	110	0	0
238	6641	0	49	0	195	82	0	138	1	0
239	6651	0	48	0	184	96	0	144	1	1
240	6661	1	55	0	209	85	0	130	1	0
241	6681	0	52	1	209	98	1	170	0	1
242	6691	0	61	0	214	100	0	158	0	1
243	6721	0	68	1	130	106	1	200	0	1
244	6731	0	55	0	196	70	0	125	0	0
245	6741	0	52	1	237	126	1	224	0	1
246	6751	0	43	0	185	85	1	140	1	0
247	6761	1	47	0	248	104	1	132	1	1
248	6781	0	57	0	252	106	0	166	0	1
249	6791	0	55	0	198	96	0	144	1	1
250	6801	0	71	0	176	62	0	138	0	0
251	6811	0	74	1	193	98	1	202	0	1
252	6821	1	65	0	185	105	0	156	0	1
253	6831	0	65	0	241	102	1	146	0	1
254	6871	0	44	0	231	70	0	108	0	0
255	6881	0	40	0	157	78	1	122	0	0
256	6891	0	45	0	152	106	1	148	1	1
257	6911	0	50	0	237	102	0	156	1	1
258	6921	0	64	1	175	110	1	142	0	1

```
259  6931   1  56   0 195  94   1 150   0   0
260  6941   0  62   1 151  88   0 165   0   1
261  6961   0  44   0 205  80   0 128   1   0
262  6981   0  73   0 190  75   0 115   0   0
263  7001   0  46   0 239 100   0 160   1   1
264  7021   0  51   0 232  80   0 120   0   0
265  7031   0  59   1 170 100   0 180   1   1
266  7051   1  67   1 319 104   0 182   0   1
267  7091   0  54   0 225  86   0 122   0   0
268  7101   0  49   0 252  90   0 128   1   0
269  7121   0  46   0 224  84   0 130   1   0
270  7131   0  42   1 229  90   1 145   0   0
271  8641   0  68   0 195  76   1 116   1   0
272  8651   0  43   0 230  85   1 135   1   0
273  8671   0  56   1 186  98   1 154   0   1
274  8682   0  68   1 192  94   0 154   1   0
275  8711   0  46   0 184  78   0 110   1   0
276  8721   1  64   1 233  94   0 140   1   0
277  8731   0  54   0 175  96   0 156   1   1
278  8751   0  48   0 188 106   0 148   1   1
279  8771   0  41   0 232  82   0 126   1   0
280  8811   0  65   1 178 106   1 194   0   1
281  8841   0  41   0 187 108   0 154   0   1
282  8851   1  42   0 207  86   1 128   1   0
283  8971   0  66   0  94  86   0 134   0   0
284  8981   0  44   0 211  90   0 142   1   0
285  9011   0  42   0 275 100   1 150   1   1
286  9021   0  51   0 165  85   0 130   1   0
287  9031   0  56   0 282  94   0 134   1   0
288  9051   1  64   1 239  94   0 162   1   1
289  9061   0  44   0 256 106   0 162   1   1
290  9071   1  55   0 175 108   0 160   1   1
291  9091   0  55   0 306  82   0 160   1   1
292  9101   1  67   0 188 102   1 168   0   1
293  9191   1  56   1 221  78   1 154   1   0
294  9201   1  63   1 213 156   1 256   0   1
295  9261   1  67   0 250 100   0 158   0   1
296  9471   0  48   0 268 120   0 172   1   1
297  9601   1  45   0 263  86   0 132   0   0
298  9631   0  49   0 150  98   1 120   1   1
299  9651   1  70   1 251 108   1 174   1   1
300  9671   0  45   0 180 102   0 156   1   1
301  9681   0  48   0 336 110   0 174   1   1
302  9711   1  42   0 210  70   0 124   1   0
303  9721   0  69   1 179 110   0 175   1   1
```

304	9731	0	44	0	177	75	0	120	0	0
305	9751	0	48	0	227	92	0	158	1	0
306	9791	0	46	0	195	72	0	120	1	0
307	9801	0	52	0	227	76	0	116	1	0
308	9811	0	73	0	250	84	0	154	0	0
309	9831	0	67	0	218	96	1	148	0	1
310	9841	0	63	0	229	100	0	168	1	1
311	9871	0	45	1	197	80	1	134	0	0
312	9881	0	46	0	190	86	0	122	1	0
313	9891	0	68	1	189	104	1	202	1	1
314	9901	0	49	0	185	80	0	120	1	0
315	9911	1	63	0	194	90	0	190	1	1
316	9931	0	59	0	192	66	0	134	0	0
317	9941	0	67	1	261	80	1	160	1	1
318	9951	0	49	0	174	78	1	108	0	0
319	9961	0	65	1	189	114	1	168	1	1
320	9981	0	44	0	248	100	0	145	1	1
321	10011	0	45	0	214	94	0	122	0	0
322	10041	0	47	0	275	76	0	114	1	0
323	10051	0	46	0	259	92	0	130	1	0
324	10071	0	52	0	230	68	0	100	0	0
325	10091	0	60	0	206	84	0	138	1	0
326	10121	0	45	0	275	95	0	125	1	1
327	10151	1	67	1	237	100	1	170	1	1
328	10181	0	60	0	289	80	1	118	0	0
329	10201	0	65	1	176	82	0	200	1	1
330	10221	0	72	1	232	80	1	210	1	1
331	10231	1	71	0	184	90	0	160	1	1
332	10241	0	55	0	283	108	1	178	1	1
333	10271	0	54	0	214	110	0	170	1	1
334	10401	0	52	1	161	76	0	162	1	1
335	10402	0	48	0	232	98	0	154	1	1
336	10921	0	66	0	228	72	0	120	1	0
337	10951	0	52	1	206	120	1	206	0	1
338	10971	0	64	0	218	80	0	110	1	0
339	11011	0	42	0	262	92	0	142	1	0
340	11081	0	52	0	227	66	0	98	0	0
341	11101	0	51	0	215	60	0	100	0	0
342	11141	0	54	0	146	70	0	115	0	0
343	11151	0	51	0	268	85	0	140	1	0
344	11161	0	60	0	211	94	0	166	0	1
345	11221	0	48	0	213	90	1	145	1	0
346	11281	0	73	0	249	108	0	206	1	1
347	11291	0	50	0	218	92	0	130	1	0
348	11321	0	45	0	221	92	0	128	0	0

```
349 11341   1  56   1 228  92   0 152   1   0
350 11351   1  46   0 240 104   0 142   1   1
351 11361   1  76   1 279  96   0 136   1   1
352 11391   0  52   0 186  70   0 118   0   0
353 11441   0  54   0 160 110   1 200   1   1
354 11461   0  53   1 222 104   1 154   0   1
355 11481   0  43   0 211  65   0 112   1   0
356 11491   0  46   0 195 132   1 230   1   1
357 11501   0  63   0 290  90   0 150   0   0
358 11511   0  44   0 220  95   0 138   0   1
359 11531   0  42   0 161  80   0 124   1   0
360 11553   0  74   1 212  98   0 164   1   1
361 11611   0  53   0 182  86   0 136   1   0
362 11651   0  56   1 223 110   1 208   1   1
363 11661   0  47   0 290  92   0 136   1   0
364 11711   0  43   0 249  90   1 162   1   1
365 11721   0  51   0 174  92   0 124   1   0
366 11731   0  63   1 204  92   1 190   1   1
367 11781   0  49   0 245  62   0 124   1   0
368 11791   0  57   1 216 114   0 174   1   1
369 11811   0  43   0 245 120   1 145   0   1
370 11831   0  58   0 151  98   0 138   1   1
371 11851   0  49   1 178 102   0 166   1   1
372 11891   0  47   0 227  88   0 132   1   0
373 11911   0  45   0 253 104   0 152   1   1
374 11941   1  65   1 222  88   1 162   0   1
375 11971   0  51   0 258  94   1 178   1   1
376 11981   0  49   0 182  84   1 124   1   0
377 11991   0  51   0 184  96   0 150   1   1
378 12051   1  67   0 357  90   0 129   0   0
379 12111   0  47   0 193  90   0 135   1   0
380 12121   0  50   0 198  82   1 136   1   0
381 12141   0  48   0 263  76   0 102   0   0
382 12151   0  48   0 254  74   0 124   0   0
383 12181   0  64   0 248  74   0 126   1   0
384 12221   0  43   0 197  84   0 122   1   0
385 12231   0  41   0 282  98   0 132   0   1
386 12241   0  48   0 238 106   0 144   1   1
387 12251   0  50   0 156  74   0 122   1   0
388 12271   0  46   0 234  70   0 120   1   0
389 12281   0  44   0 203  82   0 110   1   0
390 12291   1  65   0 200  90   0 160   1   1
391 12293   0  44   0 209  84   0 132   1   0
392 12311   0  40   0 245  94   0 142   0   0
393 12351   0  56   0 124  86   0 142   0   0
```

394	12371	0	56	1	199	86	1	154	1	0
395	12381	1	47	0	148	85	1	145	1	0
396	12391	0	48	0	246	92	0	122	1	0
397	12401	0	46	0	233	96	0	138	0	1
398	12431	0	48	0	265	100	1	142	1	1
399	12461	0	50	0	207	86	1	142	1	0
400	12471	0	69	0	227	72	1	108	1	0
401	12481	0	45	0	205	130	1	182	1	1
402	12641	0	57	0	189	102	1	128	1	1
403	12681	1	69	0	191	102	0	164	1	1
404	12741	0	45	0	171	91	0	145	1	0
405	12742	0	52	0	178	91	1	145	1	0
406	12751	0	63	0	229	94	0	148	1	0
407	12761	0	61	1	169	90	0	140	1	0
408	12801	0	48	0	238	88	0	134	1	0
409	12831	1	45	0	216	94	0	138	1	0
410	12861	0	66	1	178	110	0	198	0	1
411	12891	0	54	0	173	92	0	162	0	1
412	12901	0	45	0	173	64	0	120	1	0
413	12911	1	66	0	180	104	1	162	1	1
414	12921	0	53	0	168	110	0	154	1	1
415	12941	0	40	0	277	80	0	120	0	0
416	13021	0	55	0	181	78	0	132	1	0
417	13041	0	48	0	272	98	1	156	1	1
418	13051	0	49	0	307	88	0	130	0	0
419	13101	0	61	0	203	94	1	146	0	0
420	13111	0	41	0	212	90	0	120	1	0
421	13121	0	43	0	248	118	0	142	1	1
422	13131	0	47	0	208	110	0	160	1	1
423	13321	0	46	0	218	86	0	126	1	0
424	13351	0	63	1	163	76	0	175	1	1
425	13391	0	62	0	261	88	0	130	1	0
426	13421	0	72	0	224	100	1	190	0	1
427	13431	0	50	0	292	80	1	128	1	0
428	13451	0	46	0	202	100	1	172	1	1
429	13461	0	44	0	145	72	0	114	1	0
430	13471	0	46	1	183	88	1	162	1	1
431	13481	0	47	0	188	88	0	126	1	0
432	13511	0	51	1	209	106	0	180	1	1
433	13521	0	46	0	217	84	0	144	1	0
434	13531	0	47	0	180	78	0	126	1	0
435	13541	0	44	0	190	90	0	140	0	0
436	13551	0	55	0	211	80	0	115	1	0
437	13571	0	56	0	204	76	0	124	1	0
438	13591	0	54	0	185	98	0	170	0	1

```
439 13611   1  50   0 206  70   0 108   1   0
440 13641   0  59   0 265  96   0 150   1   1
441 13651   0  47   0 246  80   0 130   0   0
442 13661   0  65   1 171 102   0 166   0   1
443 13662   0  41   0 211  91   0 145   0   0
444 13671   0  47   0 139  96   1 192   1   1
445 13691   0  49   0 155  84   0 124   0   0
446 13721   0  50   0 229  90   0 134   0   0
447 13731   0  56   1 148 110   0 168   1   1
448 13751   0  50   0 198  86   0 134   1   0
449 13761   0  55   1 186 120   0 172   1   1
450 13771   0  52   0 211  70   0 112   1   0
451 13811   0  57   0 210  80   0 120   1   0
452 13841   1  47   1 212 122   1 220   1   1
453 13861   0  59   0 227  70   0 122   0   0
454 13901   0  47   0 232  90   0 142   0   0
455 13911   0  42   0 176  88   0 122   1   0
456 13931   0  56   0 166  86   0 126   0   0
457 13941   1  43   0 268  88   1 132   1   0
458 13951   0  55   0 178  80   0 104   1   0
459 13961   0  49   1 147 134   1 300   1   1
460 13971   0  71   1 164  94   0 174   0   1
461 14041   0  71   1 187 114   1 172   0   1
462 14071   0  69   1 165  96   1 140   0   1
463 14691   0  47   0 250  88   1 122   1   0
464 14701   0  44   0 199  70   0 120   1   0
465 14711   1  65   1 233 116   1 180   1   1
466 14731   0  65   0 182  74   0 124   0   0
467 14741   0  40   0 210  94   0 128   0   0
468 14751   0  47   0 235  86   0 128   1   0
469 14761   0  54   0 172  92   0 144   1   0
470 14771   0  64   1 198 100   0 178   0   1
471 14781   0  59   1 212  90   1 198   0   1
472 14801   0  72   0 285  90   1 150   1   0
473 14811   0  52   0 194  82   0 132   0   0
474 14861   0  56   0 237  70   0 106   0   0
475 14871   0  56   0 153  66   1  96   0   0
476 14881   0  54   0 219  92   0 152   1   0
477 14901   0  60   1 188 114   1 210   1   1
478 14911   0  63   0 276  95   1 145   0   1
479 14931   0  73   0 203  68   0 130   1   0
480 14941   1  49   0 228  98   0 140   1   1
481 14981   0  57   0 199  80   0 134   0   0
482 15001   0  53   1 162  90   0 158   1   0
483 15011   0  43   0 221  94   0 125   1   0
```

```
484 15021   0  68   0 150  78   0 145   0   0
485 15031   0  53   0 140  80   0 120   1   0
486 15041   0  50   0 162  65   1 110   0   0
487 15061   0  49   0 171  86   1 125   1   0
488 15091   0  62   0 206  94   0 144   0   0
489 15111   0  52   0 226  76   0 130   1   0
490 15121   0  57   1 113  94   1 146   1   0
491 15141   0  45   0 197  78   0 118   0   0
492 15161   0  50   0 180  88   0 132   1   0
493 15171   0  50   1 180 118   1 214   1   1
494 15191   0  53   0 196  86   0 144   1   0
495 15201   0  51   0 211  98   0 135   0   1
496 15211   0  58   1 164  96   1 155   1   1
497 15221   0  64   0 218  85   0 154   0   0
498 15251   1  49   0 191  76   0 132   1   0
499 15271   0  55   0 189  64   0 110   0   0
500 15311   0  50   0 156  82   0 114   1   0
501 15321   0  50   0 223  80   0 130   1   0
502 15361   0  57   0 165  76   0 132   1   0
503 15401   0  55   1 200  94   1 188   1   1
504 15421   0  48   1 162 135   1 250   1   1
505 15431   0  56   1 207 110   0 172   1   1
506 15441   0  72   0 262  84   0 172   1   1
507 15511   1  67   1 236 106   1 200   0   1
508 15541   0  70   1 192  90   1 162   0   1
509 15562   0  57   1 203 100   0 170   1   1
510 15611   0  67   1 200 160   1 224   0   1
511 15641   0  62   0 280  86   0 124   1   0
512 15651   0  72   1 229 140   1 270   1   1
513 15661   0  70   0 290  84   0 138   0   0
514 15671   0  65   0 222  88   0 146   0   0
515 15691   0  58   0 259 100   1 154   0   1
516 15711   0  64   0 205  80   0 140   0   0
517 15761   0  44   0 276  74   0 112   1   0
518 15791   0  55   0 171  68   0 110   0   0
519 15831   0  71   0 287  90   0 130   0   0
520 15851   1  72   0 174  78   1 192   1   1
521 15882   0  71   0 277 110   1 200   0   1
522 15891   0  54   0 192  85   0 130   0   0
523 15911   0  51   0 196  90   0 128   1   0
524 15921   0  68   0 203  74   0 138   1   0
525 15931   0  47   0 271  85   0 145   1   0
526 15941   0  49   1 169  85   1 145   0   0
527 15951   0  48   0 201  98   0 150   1   1
528 15981   0  74   0 244  94   0 164   0   1
```

529	15991	0	49	0	161	92	0	120	0	0
530	16321	1	53	0	192	106	0	164	1	1
531	16431	0	46	0	192	86	1	116	1	0
532	16441	0	59	0	230	84	0	158	1	0
533	16461	0	50	0	312	98	1	138	0	1
534	16481	1	69	1	230	100	1	170	0	1
535	16501	1	75	1	233	90	1	222	1	1
536	16531	0	42	0	207	72	0	106	1	0
537	16541	0	50	0	317	90	1	138	0	0
538	16571	0	44	0	213	84	0	118	1	0
539	16581	0	44	0	220	98	0	140	0	1
540	16591	0	42	0	225	95	0	140	0	1
541	16622	0	42	0	288	104	0	150	1	1
542	16691	0	44	0	168	94	1	134	1	0
543	16701	0	57	1	182	96	1	138	1	1
544	16711	1	68	1	242	84	0	128	1	0
545	16752	0	69	0	258	82	0	145	1	0
546	16761	0	74	1	172	100	0	190	1	1
547	16841	0	56	1	239	140	1	220	1	1
548	16871	1	58	1	209	94	1	140	1	0
549	16891	0	46	0	181	84	0	124	0	0
550	16911	0	60	1	199	100	0	162	0	1
551	16931	0	62	0	217	90	0	144	1	0
552	16971	0	74	0	200	78	0	118	1	0
553	17071	0	44	0	268	80	0	126	1	0
554	17111	0	54	0	202	86	0	134	0	0
555	17121	0	49	0	224	86	1	134	1	0
556	17131	0	46	0	302	102	0	160	1	1
557	17151	0	45	0	239	90	0	128	0	0
558	17161	0	57	0	205	88	0	140	0	0
559	17171	0	56	1	192	170	1	270	0	1
560	17181	0	42	0	282	114	0	170	1	1
561	17191	0	52	0	232	94	0	144	1	0
562	17211	0	49	0	229	92	0	162	1	1
563	17231	0	51	0	336	86	0	130	1	0
564	17251	0	40	0	146	84	0	125	0	0
565	17271	0	43	0	224	72	0	115	0	0
566	17291	0	45	0	228	76	1	136	1	0
567	17361	0	63	0	211	108	0	144	1	1
568	17401	0	52	0	212	76	0	118	0	0
569	17481	1	67	1	243	118	1	220	1	1
570	17961	0	72	1	208	94	1	174	1	1
571	17991	0	54	0	284	98	0	146	1	1
572	18061	0	52	0	190	88	1	130	1	0
573	18071	0	49	0	264	92	0	162	0	1

```
574 18101   0  42   0 288 108   0 146   0   1
575 18121   0  41   0 181  94   0 136   1   0
576 18131   1  56   1 283 100   0 188   1   1
577 18141   0  46   0 217  66   0 120   1   0
578 18151   1  52   0 250  80   0 132   1   0
579 18161   0  44   0 209  70   0 116   1   0
580 18171   1  43   0 189 106   0 154   1   1
581 18201   0  73   0 190  78   0 138   0   0
582 18401   0  54   0 223  82   0 122   1   0
583 18411   0  46   0 241  84   0 120   1   0
584 18421   0  44   0 214  96   1 142   1   1
585 18441   0  43   0 207  86   0 122   0   0
586 18481   0  46   0 186  86   0 130   0   0
587 18491   1  74   1 212  70   1 144   1   0
588 18511   0  54   1 211  94   1 152   0   0
589 18521   0  63   0 223  86   0 158   1   0
590 18551   0  58   1 206 108   1 192   0   1
591 18581   0  50   0 194  92   0 134   1   0
592 18631   0  71   0 193  82   0 115   1   0
593 18661   0  52   0 213  90   0 140   1   0
594 18681   0  63   0 318  82   1 126   0   0
595 18711   0  66   0 216 104   1 154   0   1
596 18731   0  60   0 211  66   0 128   1   0
597 18752   0  47   0 219  88   0 128   1   0
598 18771   0  57   0 322  98   0 144   1   1
599 18801   0  66   1 239 100   1 184   1   1
600 18841   0  66   1 195 104   0 158   1   1
601 18871   0  47   0 243  78   0 118   1   0
602 18921   0  60   0 223  92   1 122   1   0
603 18971   0  48   0 174 102   0 160   1   1
604 19003   0  61   1 163  86   1 144   1   0
605 19011   0  64   0 225  90   0 160   1   1
606 19061   1  46   0 252 122   0 158   1   1
607 19091   0  49   0 261 102   0 166   1   1
608 19121   0  51   0 184  88   0 118   1   0
609 19161   0  64   0 206  82   0 152   0   0
```

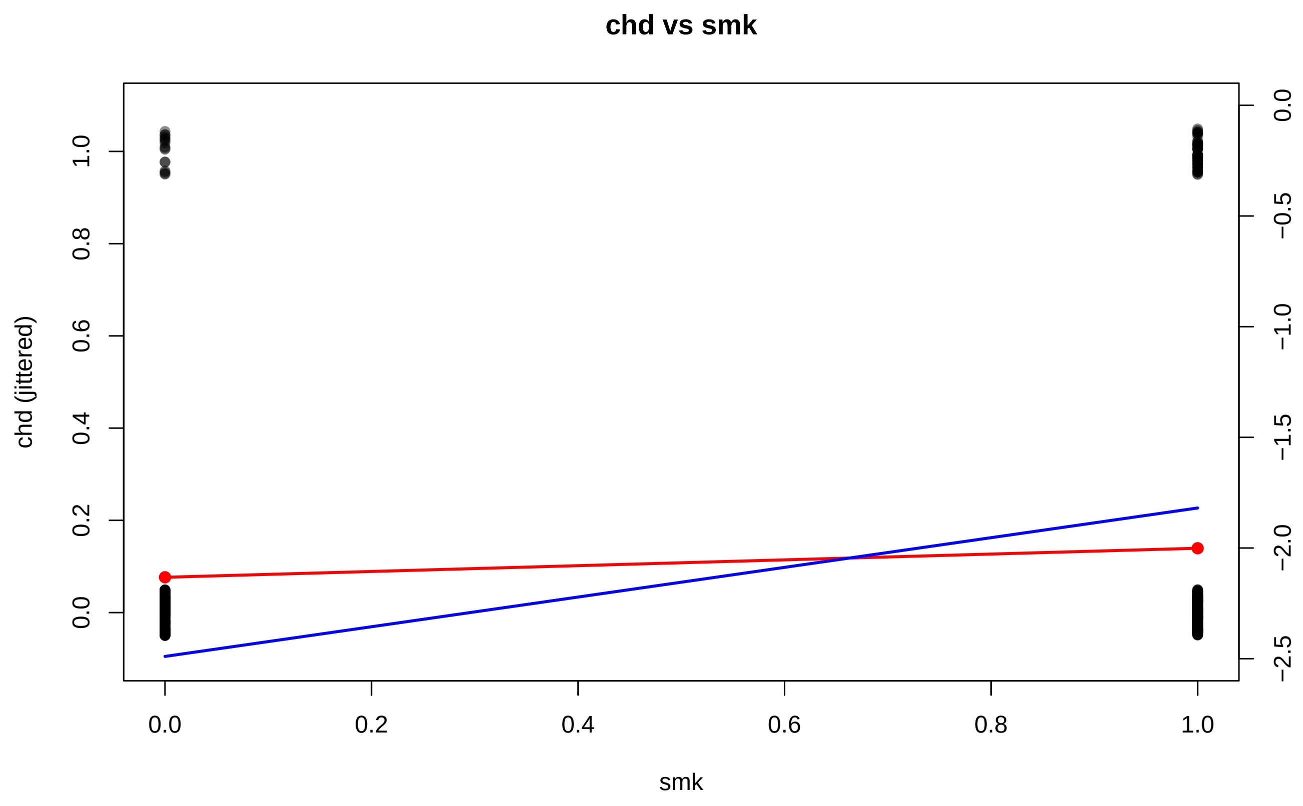
```r
colnames(CHD.data)
```

```
 [1] "id"  "chd" "age" "cat" "chl" "dbp" "ecg" "sbp" "smk" "hpt"
```

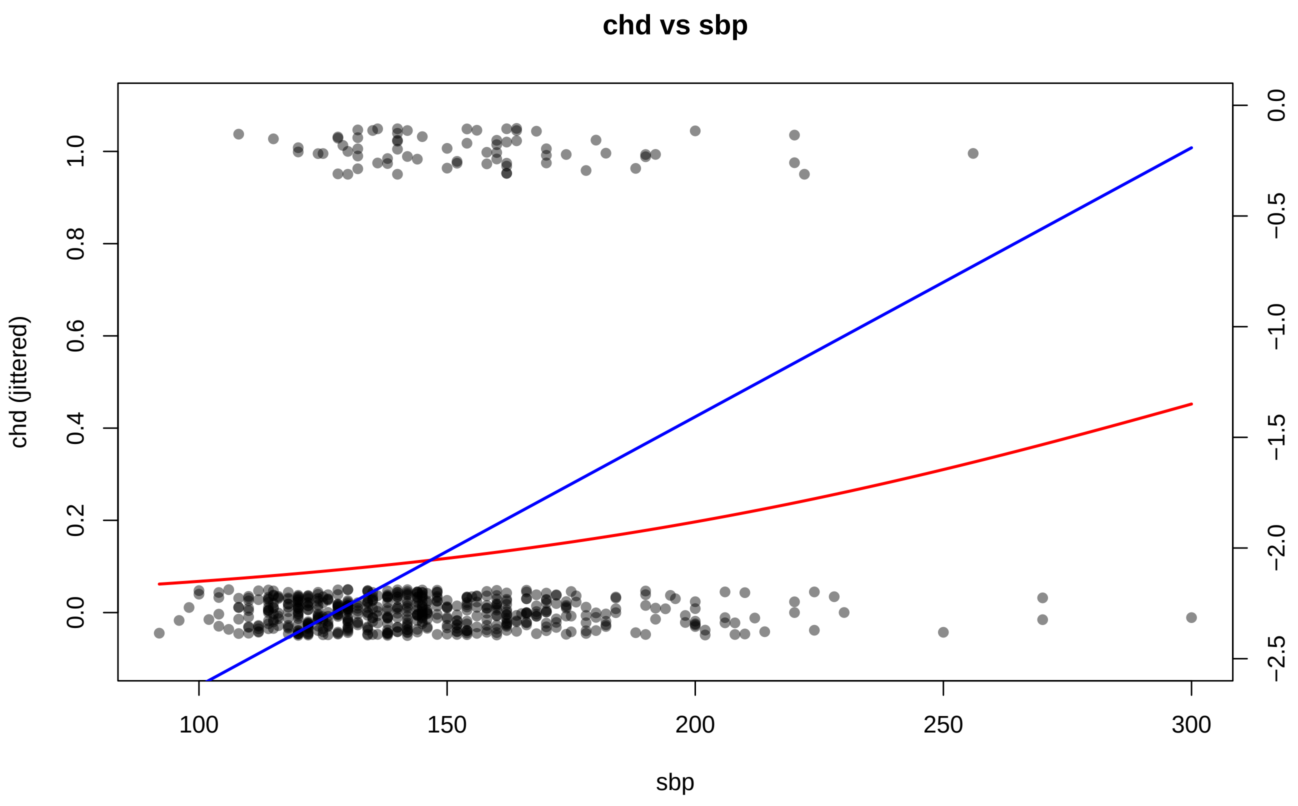
5.3.2 Fit Logistic Regression Model for a Single Variable

```r
vars <- c("smk", "sbp", "age", "chl")
#jit  <- 0.01  # global jitter amount for y

## par(mfrow = c(2, 2), mar = c(4, 4, 2, 4) + 0.1)  # extra right margin for axis(4)

for (v in vars) {
  ## Univariate logistic regression using ORIGINAL variable name in the formula
  fit <- glm(
    formula = reformulate(v, response = "chd"),
    data    = CHD.data,
    family  = binomial()
  )
  print(summary(fit))

  ## Base scatter of chd with small jitter (left axis: probability scale)
  plot(
    CHD.data[[v]],
    jitter(CHD.data$chd, amount = jit),
    pch  = 16, col = rgb(0, 0, 0, 0.45),
    xlab = v, ylab = "chd (jittered)",
    main = paste("chd vs", v),
    ylim = c(-0.1, 1.1)
  )

  ## Fitted  (x) in red (left axis)
  if (length(unique(CHD.data[[v]])) == 2) {
    # binary predictor
    xcat <- sort(unique(CHD.data[[v]]))
    nd   <- setNames(data.frame(xcat), v)
    pcat <- predict(fit, newdata = nd, type = "response")
    points(xcat, pcat, pch = 19, col = "red")
    lines(xcat, pcat, col = "red", lwd = 2)

    # Right-axis: logit{ (x)} with fixed y-limits
    logit_p <- log(pcat / (1 - pcat))
    par(new = TRUE)
    plot(
      xcat, logit_p, type = "l", lwd = 2, col = "blue",
      axes = FALSE, xlab = "", ylab = "",
      xlim = range(CHD.data[[v]]), ylim = c(-2.5, 0)
    )
    axis(4)
    mtext("logit(p(x))", side = 4, line = 3)
    par(new = FALSE)

  } else {
    # continuous predictor
    xg <- seq(min(CHD.data[[v]]), max(CHD.data[[v]]), length.out = 400)
    nd <- setNames(data.frame(xg), v)
    pg <- predict(fit, newdata = nd, type = "response")
    lines(xg, pg, col = "red", lwd = 2)

    # Right-axis: logit{ (x)} with fixed y-limits
    logit_pg <- log(pg / (1 - pg))
    par(new = TRUE)
    plot(
      xg, logit_pg, type = "l", lwd = 2, col = "blue",
      axes = FALSE, xlab = "", ylab = "",
      xlim = range(xg), ylim = c(-2.5, 0)
    )
    axis(4)
    mtext("logit(p(x))", side = 4, line = 3)
    par(new = FALSE)
  }
}
```

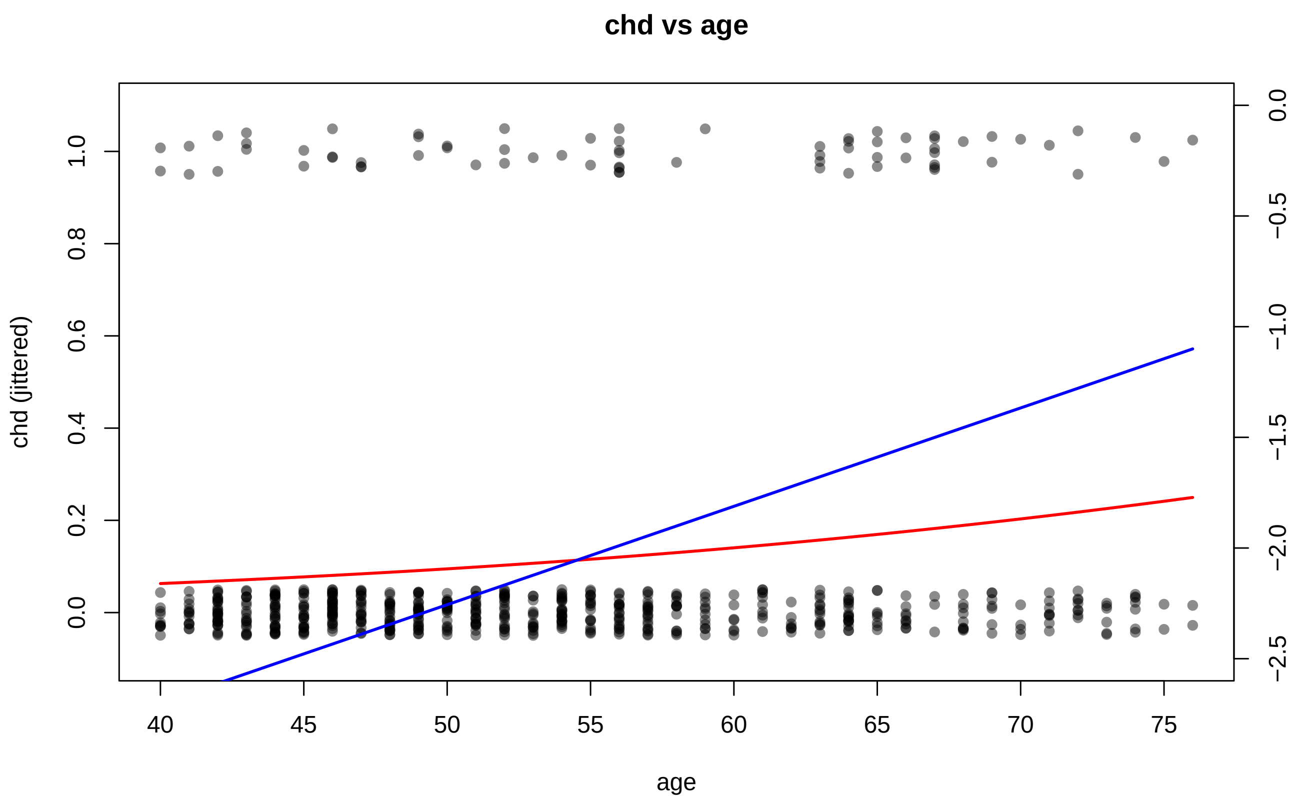
```
Call:
glm(formula = reformulate(v, response = "chd"), family = binomial(),
    data = CHD.data)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -2.4898     0.2524  -9.865   <2e-16 ***
smk           0.6706     0.2919   2.297   0.0216 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 438.56  on 608  degrees of freedom
Residual deviance: 432.81  on 607  degrees of freedom
AIC: 436.81

Number of Fisher Scoring iterations: 5
```



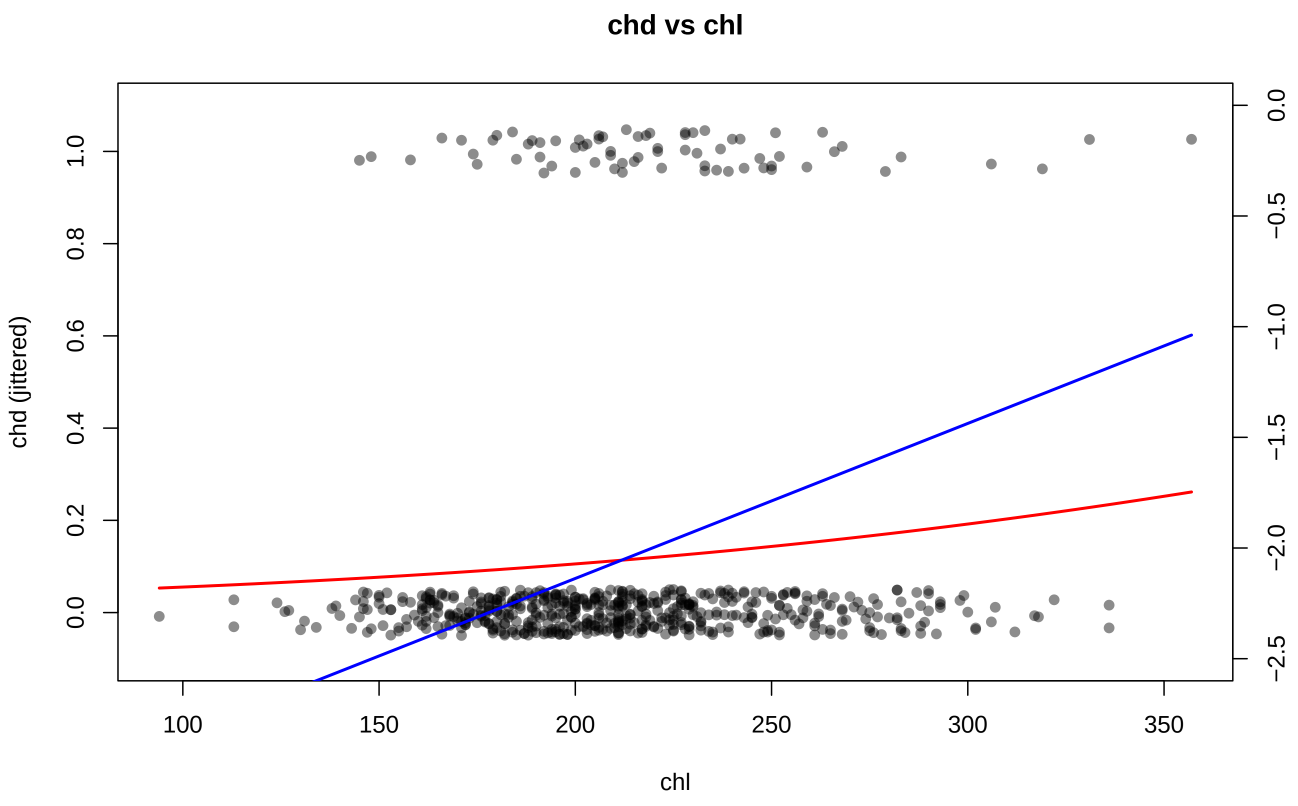
```
Call:
glm(formula = reformulate(v, response = "chd"), family = binomial(),
    data = CHD.data)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.837912   0.629805  -6.094  1.1e-09 ***
sbp          0.012154   0.004036   3.011   0.0026 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 438.56  on 608  degrees of freedom
Residual deviance: 430.06  on 607  degrees of freedom
AIC: 434.06

Number of Fisher Scoring iterations: 4
```



```
Call:
glm(formula = reformulate(v, response = "chd"), family = binomial(),
    data = CHD.data)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.47833    0.75610  -5.923 3.16e-09 ***
age          0.04445    0.01315   3.381 0.000723 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 438.56  on 608  degrees of freedom
Residual deviance: 427.22  on 607  degrees of freedom
AIC: 431.22

Number of Fisher Scoring iterations: 5
```



```
Call:
glm(formula = reformulate(v, response = "chd"), family = binomial(),
    data = CHD.data)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.538260   0.686879  -5.151 2.59e-07 ***
chl          0.007004   0.003064   2.286   0.0223 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 438.56  on 608  degrees of freedom
Residual deviance: 433.42  on 607  degrees of freedom
AIC: 437.42

Number of Fisher Scoring iterations: 4
```



5.3.3 Fit Logistic Regression Model with all variables

We fit a logistic regression with a logit link:

```
fit1_chd <- glm(
  chd ~ smk + cat + sbp + age + chl + ecg + hpt,
  data = CHD.data,
  family = binomial(link = "logit")
)
summary(fit1_chd)
```

```
Call:
glm(formula = chd ~ smk + cat + sbp + age + chl + ecg + hpt,
    family = binomial(link = "logit"), data = CHD.data)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.048892   1.345165  -4.497  6.9e-06 ***
smk          0.855951   0.306505   2.793  0.00523 **
cat          0.732763   0.376129   1.948  0.05139 .
```

```
sbp         -0.006995   0.006976  -1.003  0.31600
age          0.033956   0.015344   2.213  0.02690 *
chl          0.008970   0.003274   2.740  0.00615 **
ecg          0.417776   0.295553   1.414  0.15750
hpt          0.655498   0.359976   1.821  0.06861 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 438.56  on 608  degrees of freedom
Residual deviance: 399.35  on 601  degrees of freedom
AIC: 415.35

Number of Fisher Scoring iterations: 5
```

Notes for interpretation:

- Positive coefficients increase the log-odds of CHD; negative coefficients decrease it.
- For indicator variables (e.g., `smk`), `exp(beta)` is the adjusted odds ratio comparing the group with value 1 versus 0, holding others fixed.
- For continuous predictors (e.g., `sbp`, `age`), `exp(beta)` is the multiplicative change in the odds for a one□unit increase. For a *d*-unit increase, the OR is `exp(d * beta)`.

5.4 Inference for Coefficients: Confidence Intervals and Covariance Matrix

We extract profile□likelihood CIs and the covariance matrix to confirm standard errors.

```
ci_95 <- confint(fit1_chd, level = 0.95)     # profile-likelihood CI
vcov_mat <- vcov(fit1_chd)                    # covariance matrix of coefficients
se_vec   <- sqrt(diag(vcov_mat))          # standard errors

ci_95
```

```
                    2.5 %       97.5 %
(Intercept) -8.718003347 -3.427904298
smk          0.275699158  1.483333169
cat         -0.006873216  1.471885644
sbp         -0.021166144  0.006266328
age          0.003687290  0.064005215
chl          0.002533226  0.015404292
ecg         -0.171584621  0.990632546
```

```
hpt         -0.050184520  1.364993401
```

```
vcov_mat
```

```
             (Intercept)           smk           cat           sbp
(Intercept)  1.809468553 -1.014526e-01  0.1391440386 -4.908229e-03
smk         -0.101452600  9.394560e-02 -0.0032961000 -1.653230e-04
cat          0.139144039 -3.296100e-03  0.1414730484 -9.299960e-04
sbp         -0.004908229 -1.653230e-04 -0.0009299960  4.866901e-05
age         -0.011142995  7.738971e-04 -0.0017879998 -1.311191e-05
chl         -0.002111134  3.161443e-05  0.0003146354 -1.821907e-06
ecg          0.003442546  9.255483e-03 -0.0204455233 -2.982539e-04
hpt          0.139817180  6.954592e-03 -0.0044220690 -1.486400e-03
                     age           chl           ecg           hpt
(Intercept) -1.114300e-02 -2.111134e-03  3.442546e-03  1.398172e-01
smk          7.738971e-04  3.161443e-05  9.255483e-03  6.954592e-03
cat         -1.788000e-03  3.146354e-04 -2.044552e-02 -4.422069e-03
sbp         -1.311191e-05 -1.821907e-06 -2.982539e-04 -1.486400e-03
age          2.354442e-04 -1.480501e-06 -4.972374e-05  4.044434e-04
chl         -1.480501e-06  1.071766e-05  5.040548e-05 -6.046197e-05
ecg         -4.972374e-05  5.040548e-05  8.735130e-02  9.506863e-04
hpt          4.044434e-04 -6.046197e-05  9.506863e-04  1.295828e-01
```

```
se_vec  # should match the SE column in summary(fit1_chd)
```

```
(Intercept)         smk         cat         sbp         age         chl
1.345164879 0.306505459 0.376129032 0.006976318 0.015344190 0.003273784
        ecg         hpt
0.295552531 0.359976081
```

5.5 Inference for Odds Ratios

5.5.1 Interpretation of Odds Ratios in Logistic Regression

A multiple logistic regression model expresses the log-odds (logit) of an event as a linear function of predictors:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k.$$

Here,

- $p = \Pr(Y = 1 \mid x_1, x_2, \dots, x_k)$ is the probability of the event,
- β_0 is the intercept, and
- each β_j represents the **change in the log-odds** of the event per one-unit increase in x_j, *holding all other variables constant*.

Exponentiating both sides gives the model in odds form:

$$\frac{p}{1-p} = \exp(\beta_0) \times \exp(\beta_1 x_1) \times \exp(\beta_2 x_2) \times \cdots \times \exp(\beta_k x_k).$$

An R function for OR at two Profiles

The or_from_predict R function is a utility designed to calculate the Odds Ratio (OR) and its 95% confidence interval (CI) between two specific covariate profiles (new1 and new0) for a given logistic regression model (fit). The calculation is performed on the link (logit) scale. For a logistic model $\text{logit}(p) = \eta = \mathbf{X}\beta$, the log-Odds Ratio (logOR) is the difference between the linear predictors (η_1, η_0) for the two profiles:

$$\widehat{\text{logOR}} = \eta_1 - \eta_0 = (\mathbf{x}_1^T - \mathbf{x}_0^T)\beta = \mathbf{c}^T\beta \tag{5.1}$$

Here, $\mathbf{c} = \mathbf{x}_1 - \mathbf{x}_0$ is the linear contrast vector derived from the model matrices of the two profiles. The function estimates the variance of this contrast as $\text{Var}(\widehat{\text{logOR}}) = \mathbf{c}^T\mathbf{V}\mathbf{c}$, where $\mathbf{V}$ is the model's variance-covariance matrix (vcov(fit)). The standard error $SE = \sqrt{\mathbf{c}^T\mathbf{V}\mathbf{c}}$ is used to compute the $100(1-\alpha)\%$ confidence interval for the logOR:

$$\widehat{\text{logOR}} \pm z_{1-\alpha/2} \times SE.$$

These values (estimate and CI bounds) are then exponentiated to produce the final $\widehat{\text{OR}} = \exp(\widehat{\text{logOR}})$ and its 95% CI. The function also prints two helpful summaries to the console: a data frame showing only the variables that differ between the new0 and new1 profiles, and a 2x3 table presenting the estimates and CIs for both the OR and the logOR.

The R function to find ORs

```r
## Compute OR and 95% CI via predict() on the LINK scale
## OR = exp( eta(new1) - eta(new0) ), where eta(.) = logit{ (.)}
## Compute OR via predict() contrast on the LINK scale, also:
## (ii) print a 2-row data.frame of only variables that differ between new0 and new1
## (iii) print a 2x3 table (rows: OR, logOR; cols: Estimate, CI_low, CI_up)
or_from_predict <- function(fit, new1, new0, level = 0.95, digits = 4, tol = 1e-12) {
  stopifnot(is.data.frame(new1), is.data.frame(new0))

  ## --- REFACTORED SECTION START ---
  ## ---- (ii) Two-row data.frame with only changed variables ----

  ## Helper function to find differing variables between two profiles
  ## This is defined *inside* or_from_predict for encapsulation
  get_changed_vars <- function(d0, d1, tolerance) {
```

```
  common <- intersect(names(d0), names(d1))
  diffv  <- vapply(common, function(nm) {
    x0 <- d0[[nm]]; x1 <- d1[[nm]]
    if (is.numeric(x0) && is.numeric(x1)) {
      !isTRUE(all.equal(as.numeric(x0), as.numeric(x1), tolerance = tolerance))
    } else {
      !identical(x0, x1)
    }
  }, logical(1))

  keep <- common[diffv]
  if (length(keep) == 0L) {
    out <- data.frame(`_no_changes_` = "no differences")
    rownames(out) <- c("new0", "new1")
    return(out)
  }
  out <- rbind(d0[keep], d1[keep])
  rownames(out) <- c("new0", "new1")
  out
}

## Call the helper function
changes_df <- get_changed_vars(new0, new1, tol)
## --- REFACTORED SECTION END ---


## ---- Linear contrast for log-OR and its variance ----

## (i) Calculate logOR estimate using predict(type="link")
## eta(.) = logit{p(.)}
eta1 <- predict(fit, newdata = new1, type = "link")
eta0 <- predict(fit, newdata = new0, type = "link")
logOR_hat <- as.numeric(eta1 - eta0) # logOR = eta1 - eta0

## (ii) Calculate standard error using the contrast vector 'cvec'
X1 <- model.matrix(delete.response(terms(fit)), data = new1)
X0 <- model.matrix(delete.response(terms(fit)), data = new0)
cvec      <- as.numeric(X1 - X0)
V         <- vcov(fit)
se_logOR  <- sqrt(as.numeric(t(cvec) %*% V %*% cvec))

alpha  <- 1 - level
z      <- qnorm(1 - alpha / 2)
ci_log <- c(logOR_hat - z * se_logOR, logOR_hat + z * se_logOR)
```

```r
  ## ---- 2x3 table: rows OR and logOR; columns Estimate, CI_low, CI_up ----
  res_tab <- data.frame(
    Estimate = c(exp(logOR_hat),           logOR_hat),
    CI_low   = c(exp(ci_log[1L]),          ci_log[1L]),
    CI_up    = c(exp(ci_log[2L]),          ci_log[2L]),
    row.names = c("OR", "logOR")
  )

  ## ---- Print requested items ----
  cat("\nVariables that differ between new0 and new1:\n")
  print(changes_df)
  cat("\nOdds Ratio summary:\n")
  print(round(res_tab, digits = digits)) # Added rounding for neatness

  ## ---- Return (invisibly) ----
  invisible(list(
    OR         = exp(logOR_hat),
    CI_OR      = exp(ci_log),
    logOR      = logOR_hat,
    CI_logOR   = ci_log,
    se_logOR   = se_logOR,
    changes    = changes_df,
    table      = res_tab
  ))
}

## --- Example usage ---
## Suppose 'fit1_chd' is your fitted model and 'CHD.data' is your data
## base_prof <- as.data.frame(lapply(CHD.data, function(col) if (is.numeric(col)) mean(col) el
## new0 <- base_prof; new0$smk <- 0
## new1 <- base_prof; new1$smk <- 1
## or_from_predict(fit1_chd, new1 = new1, new0 = new0)


mean_profile <- function(data, vars_binary_as = c(0,1)) {
  ## Build a single-row data.frame of typical values:
  out <- lapply(data, function(col) {
    if (is.numeric(col)) {
      # If strictly 0/1, keep mean (works fine for GLM prediction),
      # or switch to mode if you prefer.
      if (all(col %in% c(0,1))) mean(col) else mean(col, na.rm = TRUE)
    } else {
      # Fallback to first level for factors/characters
      if (is.factor(col)) levels(col)[1] else unique(col)[1]
    }
  })
  as.data.frame(out)
}
```

5.5.2 Examples of Finding ORs and Their CIs for the CHD Dataset

5.5.2.1 OR Smoking (`smk`) (1 vs 0)

```
### 1) Smoking OR: smk = 1 vs 0 (other vars at their means)
## Example profiles at sample means (adjust as you like)
base_prof <- mean_profile(CHD.data)
new0 <- base_prof; new0$smk <- 0
new1 <- base_prof; new1$smk <- 1

res_smk <- or_from_predict(fit1_chd, new1 = new1, new0 = new0)
```

```
Variables that differ between new0 and new1:
     smk
new0   0
new1   1

Odds Ratio summary:
      Estimate CI_low  CI_up
OR      2.3536 1.2907 4.2917
logOR   0.8560 0.2552 1.4567
```

How to read this:

- `OR_smk > 1` suggests higher odds of CHD among smokers (adjusted for other variables). If the 95% CI excludes 1, the association is statistically significant at the 5% level.

5.5.2.2 OR for Systolic Blood Pressure (`sbp`): from 120 to 160

We compute the adjusted OR for a 40□unit increase in `sbp` (from 120 to 160):

```
### 2) SBP OR: 160 vs 120 (other vars at their means)
new0 <- base_prof; new0$sbp <- 120
new1 <- base_prof; new1$sbp <- 160

res_sbp <- or_from_predict(fit1_chd, new1 = new1, new0 = new0)
```

```
Variables that differ between new0 and new1:
     sbp
new0 120
new1 160

Odds Ratio summary:
      Estimate  CI_low  CI_up
OR      0.7559  0.4375 1.3062
logOR  -0.2798 -0.8267 0.2671
```

5.5.2.3 OR for Combined Effects of Two Variables: Smoking with an Age Difference

Suppose we compare two groups that differ in **smoking status** and **age**:

- **Group A:** `smk = 1`, `age = 50` (all other covariates equal)
- **Group B:** `smk = 0`, `age = 30`

The log□odds contrast is ($A = \beta_{smk} + (50 - 20)\beta_{age}$), so the OR is (exp$(A)$).

```r
new0 <- base_prof; new0$age <- 30; new0$smk <- 0
new1 <- base_prof; new1$age <- 50; new1$smk <- 1

res_ageAsmk <- or_from_predict(fit1_chd, new1 = new1, new0 = new0)
```

```
Variables that differ between new0 and new1:
     age smk
new0  30   0
new1  50   1

Odds Ratio summary:
      Estimate CI_low   CI_up
OR      4.6417 1.8546 11.6168
logOR   1.5351 0.6177  2.4525
```

5.6 Assessing Statistical Significance with Wilks' Theorem (Analogue of F-test for OLS)

In the context of logistic regression, Wilks' theorem provides the basis for the Likelihood Ratio Test (LRT) used to assess the significance of predictor variables. The theorem states that when comparing a full model (M_1) to a nested null model (M_0), the test statistic, Λ, asymptotically follows a chi-squared (χ^2) distribution under the null hypothesis (i.e., that the simpler model M_0 is correct).

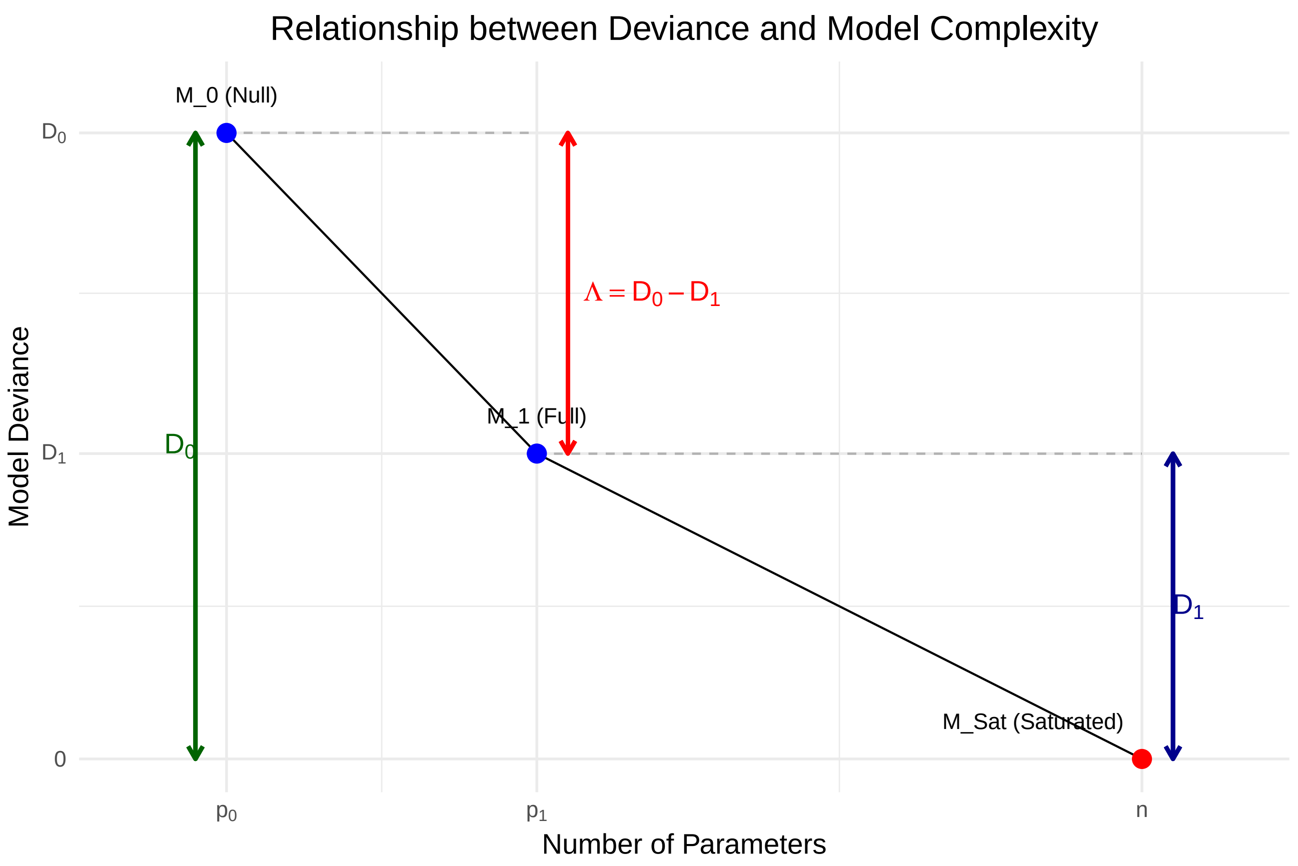
The statistic Λ is calculated as the difference in the maximized log-likelihoods:

$$\Lambda = -2(\log L_0 - \log L_1) \tag{5.2}$$

where log L_0 and log L_1 are the log-likelihoods of the null and full models, respectively. In logistic regression, this is equivalent to the difference in the deviances: $\Lambda = \text{Deviance}_0 - \text{Deviance}_1$. This test statistic Λ represents the reduction in deviance (a measure of badness-of-fit) achieved by adding the extra predictors to the model.

The following R code chunk generates a conceptual plot of this relationship:

```r
#| label: plot-lrt-concept
#| echo: false
#| fig-cap: "Conceptual plot of Deviance versus Number of Parameters, illustrating the Likelihood

library(ggplot2)

## 1. Create conceptual data for the plot
## These are just for illustration
n_obs <- 60 # Number of observations
p0 <- 1     # Parameters in null model (intercept)
p1 <- 21    # Parameters in full model (e.g., intercept + 7 predictors)
psat <- n_obs # Parameters in saturated model (1 per observation)

D0 <- 41 # Null deviance
D1 <- 20 # Full model deviance (residual deviance of M1)
D_sat <- 0 # Saturated model deviance

## Data frame for the three points
plot_data <- data.frame(
  model = c("M_0 (Null)", "M_1 (Full)", "M_Sat (Saturated)"),
  params = c(p0, p1, psat),
  deviance = c(D0, D1, D_sat),
  ## Add custom justification and nudges for labels
  hjust_val = c(0.5, 0.5, 1.1), # Right-align the last label
  nudge_x_val = c(0, 0, 0)
)

## 2. Create the ggplot
ggplot(plot_data, aes(x = params, y = deviance)) +
  ## Draw dashed guide lines for D0 and D1
  geom_segment(aes(x = p0, y = D0, xend = p1, yend = D0), linetype = "dashed", color = "grey70")
  geom_segment(aes(x = p1, y = D1, xend = psat, yend = D1), linetype = "dashed", color = "grey70"

  ## Connect the points with lines
  geom_line(color = "black", linetype = "solid", linewidth = 0.5) +
  ## Draw the main points
  geom_point(size = 4, aes(color = model)) +

  ## --- MODIFIED LABEL PLACEMENT ---
  ## Label the points using custom nudge/justification
  geom_text(
    aes(label = model, hjust = hjust_val, nudge_x = nudge_x_val),
    nudge_y = 2.5,  # Use a much smaller vertical nudge
    size = 4
  ) +

  ## --- ADDED BACK D0 and D1 ANNOTATIONS ---
  ## D0 (Null Deviance)
  geom_segment(
    aes(x = p0 - 2, y = D0, xend = p0 - 2, yend = D_sat), # Nudged left
    arrow = arrow(ends = "both", length = unit(0.1, "inches")),
    color = "darkgreen",
    linewidth = 1
  ) +
  annotate(
    "text",
    x = p0 - 3, y = D0 / 2, # Nudged left
    label = "D[0]", parse = TRUE,
    color = "darkgreen", hjust = 0.5, size = 5
  ) +

  ## D1 (Residual Deviance of Full Model)
  geom_segment(
    aes(x = psat + 2, y = D1, xend = psat + 2, yend = D_sat), # Nudged right
    arrow = arrow(ends = "both", length = unit(0.1, "inches")),
    color = "darkblue",
    linewidth = 1
  ) +
  annotate(
    "text",
    x = psat + 3, y = D1 / 2, # Nudged right
    label = "D[1]", parse = TRUE,
    color = "darkblue", hjust = 0.5, size = 5
  ) +

  ## LRT statistic Λ = D0 - D1
  geom_segment(
    aes(x = p1 + 2, y = D0, xend = p1 + 2, yend = D1), # Nudged right
    arrow = arrow(ends = "both", length = unit(0.1, "inches")),
    color = "red",
    linewidth = 1
  ) +
  annotate(
    "text",
    x = p1 + 3, y = D1 + (D0 - D1) / 2, # Nudged right
    label = "Lambda == D[0] - D[1]",
    parse = TRUE,
    color = "red", hjust = 0, size = 5
  ) +

  ## Customize axes to show the symbolic labels
  scale_x_continuous(
    breaks = c(p0, p1, psat),
    labels = c(expression(p[0]), expression(p[1]), expression(n)),
    expand = expansion(mult = 0.1) # Add some padding
  ) +
  scale_y_continuous(
    breaks = c(D_sat, D1, D0),
    labels = c(expression(0), expression(D[1]), expression(D[0]))
  ) +

  ## Labels and Title
  labs(
    title = "Relationship between Deviance and Model Complexity",
    x = "Number of Parameters",
    y = "Model Deviance"
  ) +

  ## Clean theme
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(hjust = 0.5),
    legend.position = "none" # Remove legend, as points are labeled
  ) +
  scale_color_manual(values = c("M_0 (Null)" = "blue", "M_1 (Full)" = "blue", "M_Sat (Saturated)"
```



As the diagram illustrates, the null model (M_0) has fewer parameters (p_0) and a higher deviance (D_0, or worse fit), while the full model (M_1) has more parameters (p_1) and a lower deviance (D_1). The Likelihood Ratio Test statistic D is the magnitude of this drop in deviance.

For assessing the overall significance of a regression model (`fit1_chd`), this involves comparing it to its corresponding intercept-only (null) model. The degrees of freedom for the χ^2 test is the difference in the number of parameters, $df = p_1 - p_0$, which equals the number of predictors in the full model.

Here is an R code chunk demonstrating how to compute this p-value directly from a `glm` fit object, assuming it is named `fit1_chd`.

```r
## Calculate the Likelihood Ratio Test statistic (D) and degrees of freedom (df)
## by comparing the model's deviance to the null (intercept-only) deviance,
## both of which are stored in the 'fit1_chd' object.
summary(fit1_chd)
```

```

Call:
glm(formula = chd ~ smk + cat + sbp + age + chl + ecg + hpt,
    family = binomial(link = "logit"), data = CHD.data)
```

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)    
(Intercept) -6.048892   1.345165  -4.497  6.9e-06 ***
smk          0.855951   0.306505   2.793  0.00523 ** 
cat          0.732763   0.376129   1.948  0.05139 .  
sbp         -0.006995   0.006976  -1.003  0.31600    
age          0.033956   0.015344   2.213  0.02690 *  
chl          0.008970   0.003274   2.740  0.00615 ** 
ecg          0.417776   0.295553   1.414  0.15750    
hpt          0.655498   0.359976   1.821  0.06861 .  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 438.56  on 608  degrees of freedom
Residual deviance: 399.35  on 601  degrees of freedom
AIC: 415.35

Number of Fisher Scoring iterations: 5
```

```
lrt_statistic <- fit1_chd$null.deviance - fit1_chd$deviance
lrt_df <- fit1_chd$df.null - fit1_chd$df.residual

## Compute the p-value from the chi-squared distribution
## We use lower.tail = FALSE to get P(ChiSq > D)
p_value <- pchisq(lrt_statistic, lrt_df, lower.tail = FALSE)

## Create and print the result in an ANOVA-like table
## Row 1: Null model
## Row 2: Full model (fit1_chd), showing the test against the null
lrt_table <- data.frame(
  "Resid. Df" = c(fit1_chd$df.null, fit1_chd$df.residual),
  "Resid. Dev" = c(round(fit1_chd$null.deviance, 4), round(fit1_chd$deviance, 4)),
  "Test Df" = c(NA, lrt_df),
  "Test Statistic (D)" = c(NA, round(lrt_statistic, 4)),
  "p-value" = c(NA, format.pval(p_value, digits = 4)),
  row.names = c("Null Model", "Full Model (fit1_chd)"),
  check.names = FALSE # Prevent R from changing 'p-value' to 'p.value'
)

cat("Likelihood Ratio Test for Model Significance:\n")
```

```
Likelihood Ratio Test for Model Significance:
```

```r
lrt_table
```

```
                        Resid. Df Resid. Dev Test Df Test Statistic (D)   p-value
Null Model                    608   438.5583      NA                 NA      <NA>
Full Model (fit1_chd)         601   399.3539       7            39.2044 1.787e-06
```

Using built-in anova() function

```r
fit0_chd <- glm (chd~1, data = CHD.data, family = binomial())
anova(fit0_chd, fit1_chd)
```

```
Analysis of Deviance Table

Model 1: chd ~ 1
Model 2: chd ~ smk + cat + sbp + age + chl + ecg + hpt
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)    
1       608     438.56                          
2       601     399.35  7   39.204 1.787e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
anova(fit1_chd, test="LRT")
```

```
Analysis of Deviance Table

Model: binomial, link: logit

Response: chd

Terms added sequentially (first to last)

     Df Deviance Resid. Df Resid. Dev  Pr(>Chi)    
NULL                   608     438.56              
smk   1   5.7453       607     432.81 0.0165324 *  
cat   1  14.3716       606     418.44 0.0001501 ***
sbp   1   0.7574       605     417.68 0.3841353    
age   1   5.2821       604     412.40 0.0215455 *  
chl   1   7.8619       603     404.54 0.0050489 ** 
ecg   1   1.8701       602     402.67 0.1714609    
hpt   1   3.3159       601     399.35 0.0686113 .  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

5.7 Assessing Predictive Effect-Size (Anologue to R^2_{adj})

While the LRT assesses overall model significance (in-sample fit), it's also crucial to evaluate how well the model predicts new, unseen data (out-of-sample performance). A common method is to split the data into a training set (e.g., 2/3 of the data) and a test set (e.g., 1/3). The model is fit using only the training data and then used to make predictions for the test data. We can then compare these predictions to the actual outcomes in the test set.

5.7.1 Understanding the Confusion Matrix and Metrics

To evaluate a model's predictive performance, we classify its probabilistic predictions using a threshold (typically 0.5) and compare them to the true outcomes in a **Confusion Matrix**:

	Predicted: 0	**Predicted: 1**
Actual: 0	True Negative (TN)	False Positive (FP)
Actual: 1	False Negative (FN)	True Positive (TP)

From this matrix, we derive several key performance metrics:

- **Misclassification Error Rate (ER):** The proportion of all predictions that were incorrect.

$$\text{Error Rate} = \frac{FP + FN}{TP + TN + FP + FN}$$

- **Precision (Positive Predictive Value):** Answers: "Of all the times the model predicted positive, how often was it correct?" This is crucial when the cost of a **False Positive** is high.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall (Sensitivity or True Positive Rate):** Answers: "Of all the actual positive cases, how many did the model find?" This is crucial when the cost of a **False Negative** is high.

$$\text{Recall (TPR)} = \frac{TP}{TP + FN}$$

- **ROC Curve and AUC:** An **ROC (Receiver Operating Characteristic) Curve** is a graph that shows a model's diagnostic ability across *all possible classification thresholds*. It plots the **True Positive Rate (Recall)** on the y-axis against the **False Positive Rate** (FPR = $\frac{FP}{FP+TN}$) on the x-axis.
  - **Interpretation:** The curve shows the trade-off between sensitivity (finding all the positives) and specificity (not mislabeling negatives). A random "no-skill" classifier is represented by a diagonal line from (0,0) to (1,1). A perfect classifier would hug the **top-left corner** (TPR = 1, FPR = 0).

  - **AUC (Area Under the Curve):** The AUC summarizes the entire curve into a single number from 0 to 1. An AUC of 0.5 corresponds to a random guess, while an AUC of 1.0 represents a perfect model.

- **Precision-Recall (PR) Curve:** A **PR Curve** plots **Precision** (y-axis) against **Recall** (x-axis) at all possible thresholds.
  - **Interpretation:** This curve shows the trade-off between how *reliable* a positive prediction is (Precision) and how *complete* the model is at finding all positives (Recall).
  - **When to Use:** The PR curve is particularly informative when the dataset is **imbalanced** (i.e., one class, like "fraud" or "disease," is much rarer than the other). Unlike the ROC curve, the PR curve's baseline (the "no-skill" line) is a horizontal line at the proportion of positive cases, which makes it easier to see if the model is performing significantly better than chance in a low-positive-rate scenario. A perfect classifier would hug the **top-right corner** (Precision = 1, Recall = 1).

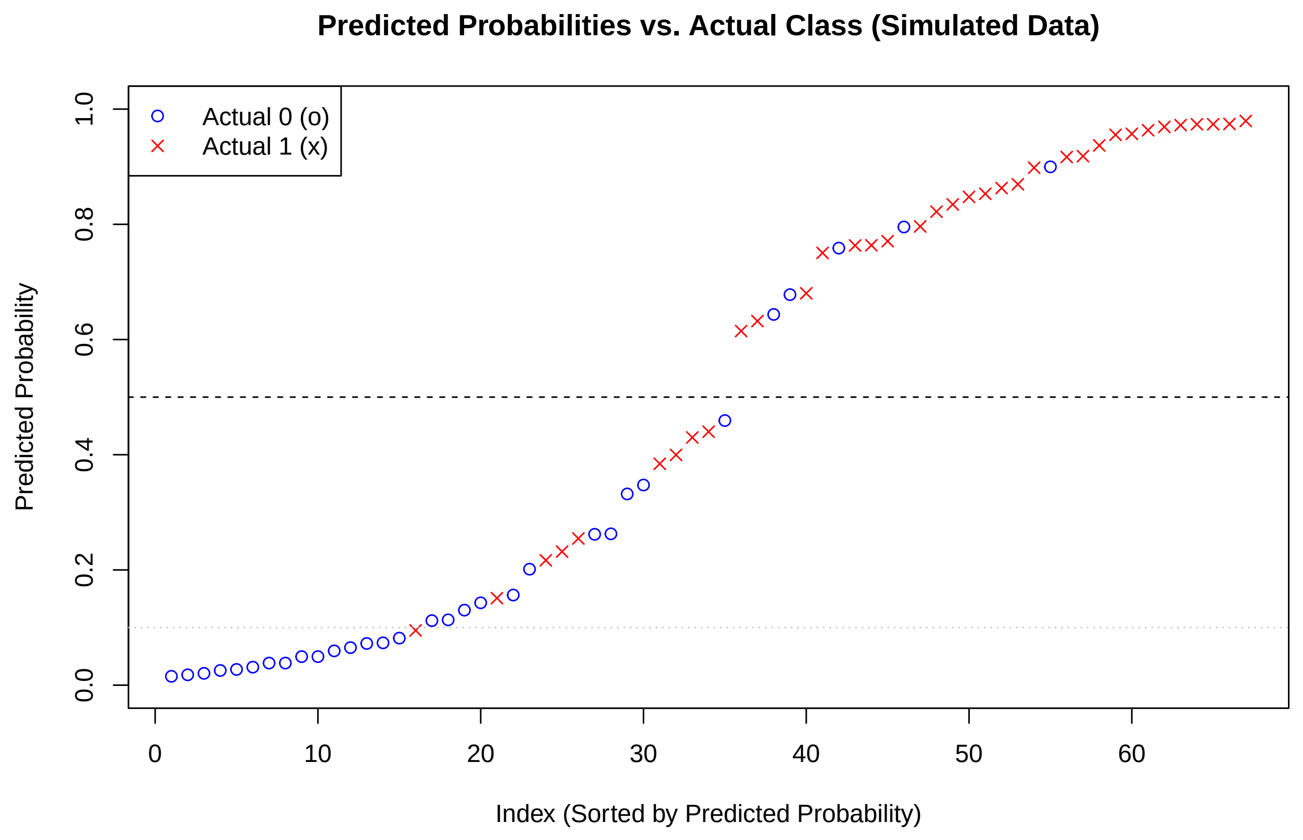
5.7.2 Illustration with the Simulated Dataset

This section applies the train/test split and model evaluation workflow to the `sim.data` created in the previous step.

```r
## Load the pROC library for AUC calculation
## install.packages("pROC") # Uncomment to install if needed
library(pROC)

## --- 1. Split the data ---
## We use 'sim.data' which has 200 rows
set.seed(123) # for reproducibility
n_sim <- nrow(sim.data)
train_size_sim <- floor(2/3 * n_sim)
train_indices_sim <- sample(1:n_sim, size = train_size_sim)
train_data_sim <- sim.data[train_indices_sim, ]
test_data_sim  <- sim.data[-train_indices_sim, ]

## --- 2. Refit the model on the training data ---
## We fit the model y ~ x on the training data
fit_train_sim <- glm(
  y ~ x,
  data = train_data_sim,
  family = binomial(link = "logit")
)

## --- 3. Make predictions on the test data ---
## Note: The true probabilities 'p' are also in test_data_sim
## We predict from the *fitted* model
pred_probs_sim <- predict(fit_train_sim, newdata = test_data_sim, type = "response")
```

Plotting the Predictive Probabilities with True Labels

```r
## --- 5. Plot sorted predicted probabilities ---

## Create a data frame for plotting
plot_data_sim <- data.frame(
  Prob = pred_probs_sim,
  Actual = as.factor(test_data_sim$y),
  TrueProb = test_data_sim$p # Include true probs for comparison
)

## Sort by predicted probability
plot_data_sim <- plot_data_sim[order(plot_data_sim$Prob), ]
plot_data_sim$Rank <- 1:nrow(plot_data_sim)

## Create the plot
plot(
  plot_data_sim$Rank,
  plot_data_sim$Prob,
  pch = ifelse(plot_data_sim$Actual == 0, 1, 4),
  col = ifelse(plot_data_sim$Actual == 0, "blue", "red"),
  xlab = "Index (Sorted by Predicted Probability)",
  ylab = "Predicted Probability",
  main = "Predicted Probabilities vs. Actual Class (Simulated Data)",
  ylim = c(0, 1)
)
abline(h = 0.5, lty = 2, col = "black")
abline(h = 0.1, lty = 3, col = "grey")

## Add the true probability curve (sorted by predicted prob)
## This shows how well the fitted model's predictions align with the true probs
#lines(plot_data_sim$Rank, plot_data_sim$TrueProb[order(plot_data_sim$Prob)], col = "darkgreen",


## Add a legend
legend(
  "topleft",
  legend = c("Actual 0 (o)", "Actual 1 (x)"),
  pch = c(1, 4),
  lty = c(NA, NA),
  lwd = c(NA, NA),
  col = c("blue", "red")
)
```



Confusion Matrix with threshold=0.5

```r
## --- 4. Assess accuracy ---

## 4a. Misclassification Error Rate (using 0.5 threshold)
threshold <- 0.5
pred_class_sim <- ifelse(pred_probs_sim > threshold, 1, 0)
conf_matrix_sim <- table(Actual = test_data_sim$y, Predicted = pred_class_sim)

## --- MODIFIED LINES START ---
cat("Confusion Matrix (Counts, threshold = 0.5):\n")
```

```
Confusion Matrix (Counts, threshold = 0.5):
```

```r
print(conf_matrix_sim)
```

```
      Predicted
Actual  0  1
     0 26  5
     1  9 27
```

```r
cat("\nRow Proportions (Given Actual, % Predicted -- Relates to TPR/FPR):\n")
```

```
Row Proportions (Given Actual, % Predicted -- Relates to TPR/FPR):
```

```r
## margin = 1 calculates proportions across rows
print(round(prop.table(conf_matrix_sim, margin = 1), 3))
```

```
      Predicted
Actual     0     1
     0 0.839 0.161
     1 0.250 0.750
```

```r
cat("\nColumn Proportions (Given Predicted, % Actual -- Relates to Precision):\n")
```

```
Column Proportions (Given Predicted, % Actual -- Relates to Precision):
```

```r
## margin = 2 calculates proportions across columns
print(round(prop.table(conf_matrix_sim, margin = 2), 3))
```

```
      Predicted
Actual     0     1
     0 0.743 0.156
     1 0.257 0.844
```

```r
## --- MODIFIED LINES END ---


## Check if matrix has 2x2 dimensions, otherwise metrics will fail
if (all(dim(conf_matrix_sim) == c(2, 2))) {
  TN <- conf_matrix_sim[1, 1]
  FP <- conf_matrix_sim[1, 2]
  FN <- conf_matrix_sim[2, 1]
  TP <- conf_matrix_sim[2, 2]

  ## Calculate metrics
  error_rate <- (FP + FN) / (TP + TN + FP + FN)
  TPR_Recall <- TP / (TP + FN) # True Positive Rate (Recall / Sensitivity)
  FPR <- FP / (FP + TN)      # False Positive Rate (1 - Specificity)
  Precision <- TP / (TP + FP)  # Positive Predictive Value
  cat(paste("\nMisclassification Error Rate:", round(error_rate, 4), "\n"))
  cat(paste("True Positive Rate (Recall):", round(TPR_Recall, 4), "\n"))
  cat(paste("False Positive Rate:", round(FPR, 4), "\n"))
  cat(paste("Precision:", round(Precision, 4), "\n"))
} else {
  cat("\nCannot calculate full metrics: model predicted only one class.\n")
}
```

```
Misclassification Error Rate: 0.209
True Positive Rate (Recall): 0.75
False Positive Rate: 0.1613
Precision: 0.8438
```

Confusion Matrix with threshold=0.1

```
threshold <- 0.1
pred_class_sim <- ifelse(pred_probs_sim > threshold, 1, 0)
conf_matrix_sim <- table(Actual = test_data_sim$y, Predicted = pred_class_sim)

## --- MODIFIED LINES START ---
cat("Confusion Matrix (Counts, threshold = 0.1):\n")
```

```
Confusion Matrix (Counts, threshold = 0.1):
```

```
print(conf_matrix_sim)
```

```
      Predicted
Actual  0  1
     0 15 16
     1  1 35
```

```
cat("\nRow Proportions (Given Actual, % Predicted -- Relates to TPR/FPR):\n")
```

```
Row Proportions (Given Actual, % Predicted -- Relates to TPR/FPR):
```

```
## margin = 1 calculates proportions across rows
print(round(prop.table(conf_matrix_sim, margin = 1), 3))
```

```
      Predicted
Actual     0     1
     0 0.484 0.516
     1 0.028 0.972
```

```
cat("\nColumn Proportions (Given Predicted, % Actual -- Relates to Precision):\n")
```

```
Column Proportions (Given Predicted, % Actual -- Relates to Precision):
```

```
## margin = 2 calculates proportions across columns
print(round(prop.table(conf_matrix_sim, margin = 2), 3))
```

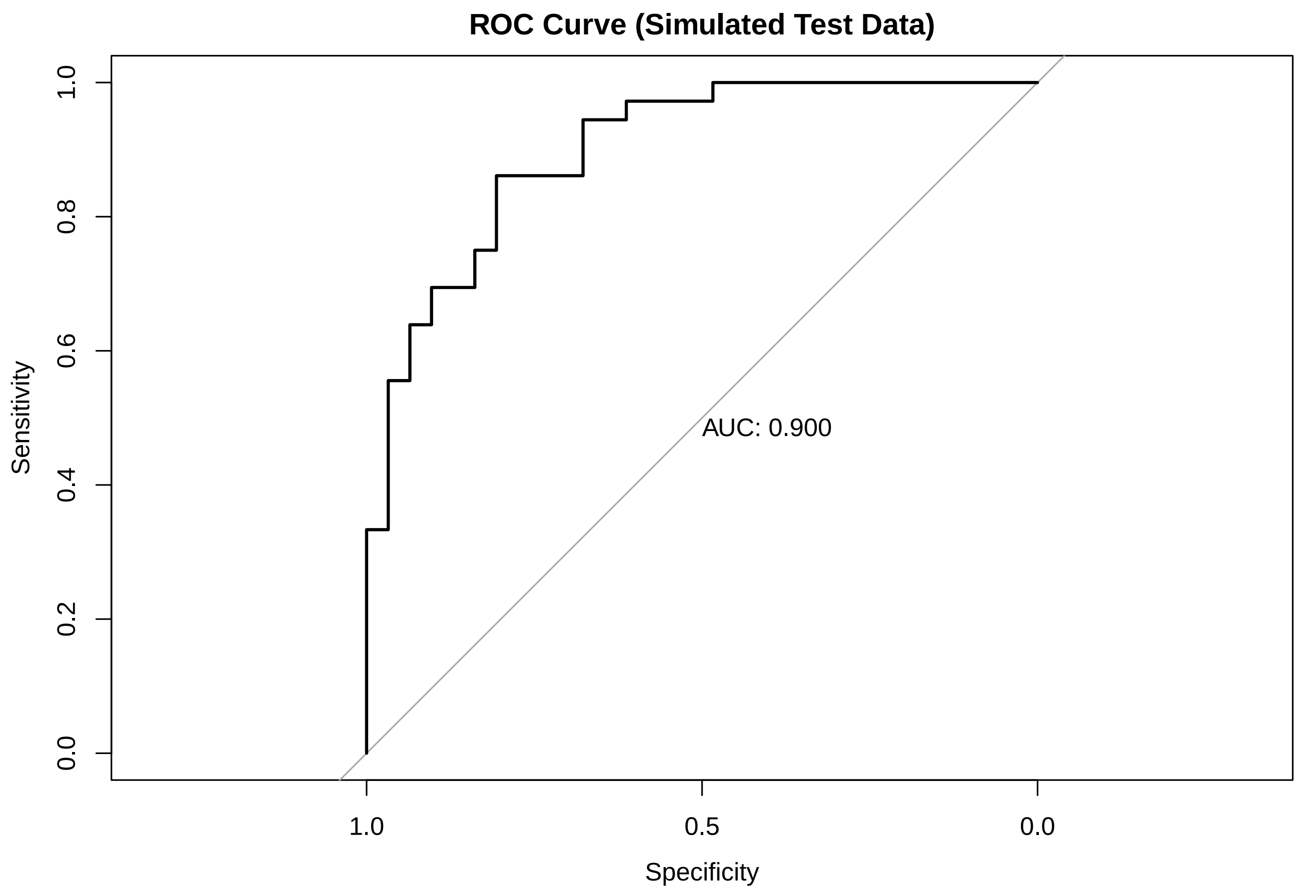
```
      Predicted
Actual     0     1
     0 0.938 0.314
     1 0.062 0.686
```

```
## --- MODIFIED LINES END ---


## Check if matrix has 2x2 dimensions
if (all(dim(conf_matrix_sim) == c(2, 2))) {
  TN <- conf_matrix_sim[1, 1]
  FP <- conf_matrix_sim[1, 2]
  FN <- conf_matrix_sim[2, 1]
  TP <- conf_matrix_sim[2, 2]

  ## Calculate metrics
  error_rate <- (FP + FN) / (TP + TN + FP + FN)
  TPR_Recall <- TP / (TP + FN) # True Positive Rate (Recall / Sensitivity)
  FPR <- FP / (FP + TN)      # False Positive Rate (1 - Specificity)
  Precision <- TP / (TP + FP)  # Positive Predictive Value

  cat(paste("\nMisclassification Error Rate:", round(error_rate, 4), "\n"))
  cat(paste("True Positive Rate (Recall):", round(TPR_Recall, 4), "\n"))
  cat(paste("False Positive Rate:", round(FPR, 4), "\n"))
  cat(paste("Precision:", round(Precision, 4), "\n"))
} else {
  cat("\nCannot calculate full metrics: model predicted only one class.\n")
}
```

```
Misclassification Error Rate: 0.2537
True Positive Rate (Recall): 0.9722
False Positive Rate: 0.5161
Precision: 0.6863
```

ROC curve and Area Under the ROC (AUC)

```r
## 4b. Area Under the Curve (AUC)
roc_curve_sim <- roc(test_data_sim$y, pred_probs_sim, quiet = TRUE)

## Plot the ROC curve
plot(roc_curve_sim, main = "ROC Curve (Simulated Test Data)", print.auc = TRUE)
```



```r
auc_value_sim <- auc(roc_curve_sim)
cat(paste("Area Under the Curve (AUC):", round(auc_value_sim, 4), "\n\n"))
```

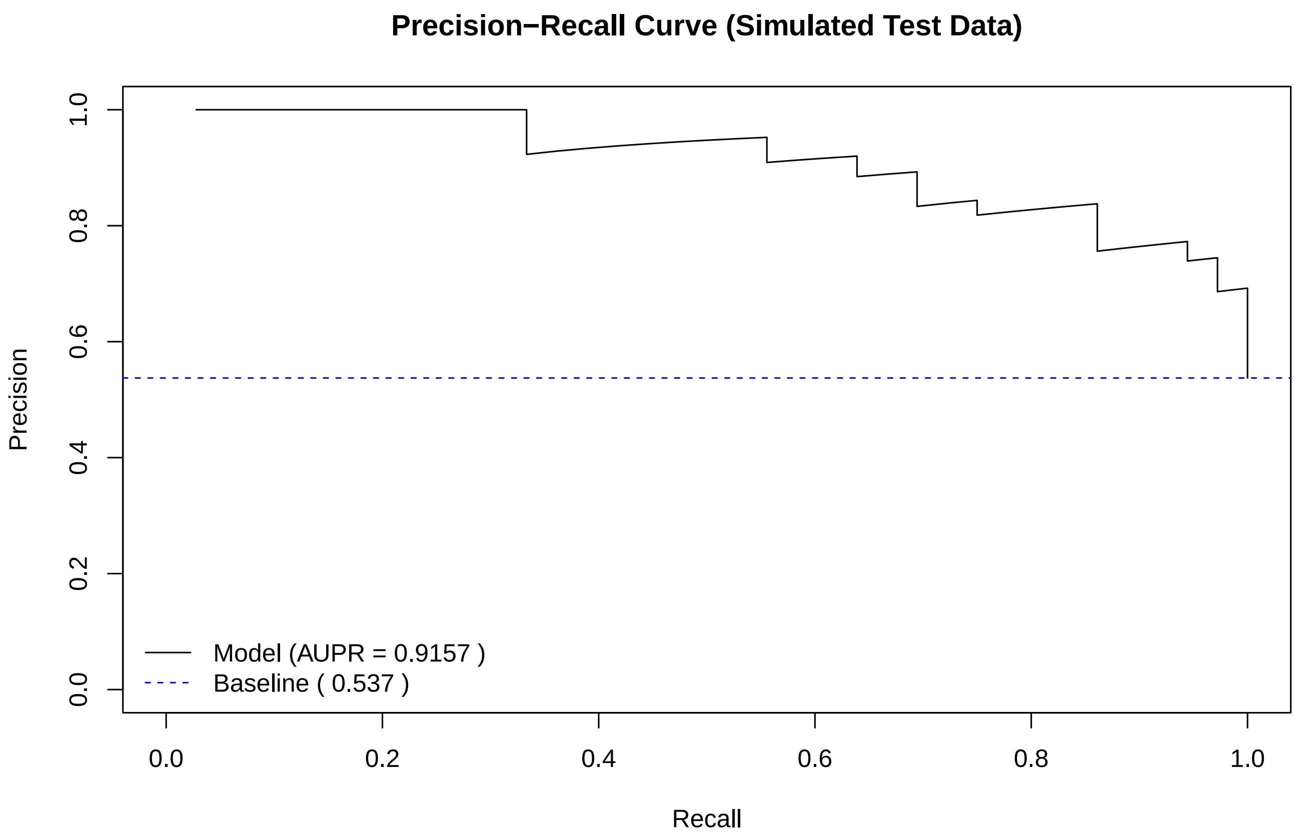
```
Area Under the Curve (AUC): 0.8996
```

PR curve and Area Under PR Curve (AUPR)

```r
## Load the ROCR library
## install.packages("ROCR") # Uncomment to install if needed
library(ROCR)

## --- 1. Create a 'prediction' object ---
## 'prediction' takes all predictions and all true labels
pred_obj <- prediction(pred_probs_sim, test_data_sim$y)

## --- 2. Create a 'performance' object for PR ---
## "prec" is for precision, "rec" is for recall
perf_pr <- performance(pred_obj, measure = "prec", x.measure = "rec")

## --- 3. Calculate Area Under the PR Curve (AUPR) ---
perf_auc <- performance(pred_obj, measure = "aucpr") # "aucpr" = Area Under PR Curve
aupr_value <- perf_auc@y.values[[1]]
cat(paste("Area Under PR Curve (AUPR):", round(aupr_value, 4), "\n"))
```

```
Area Under PR Curve (AUPR): 0.9157
```

```r
## --- 4. Plot the performance object ---
plot(perf_pr,
     main = "Precision-Recall Curve (Simulated Test Data)",
     xlim = c(0, 1),
     ylim = c(0, 1),
     col = "black")

## --- 5. Calculate and add the 'no-skill' baseline ---
baseline_precision_sim <- sum(test_data_sim$y == 1) / length(test_data_sim$y)
abline(h = baseline_precision_sim, col = "blue", lty = 2)

## --- 6. Add a legend with AUPR ---
legend("bottomleft",
       legend = c(
           paste("Model (AUPR =", round(aupr_value, 4), ")"),  # <-- MODIFIED LINE
           paste("Baseline (", round(baseline_precision_sim, 3), ")")
       ),
       col = c("black", "blue"),
       lty = c(1, 2),
       bty = "n") # bty="n" removes the box
```

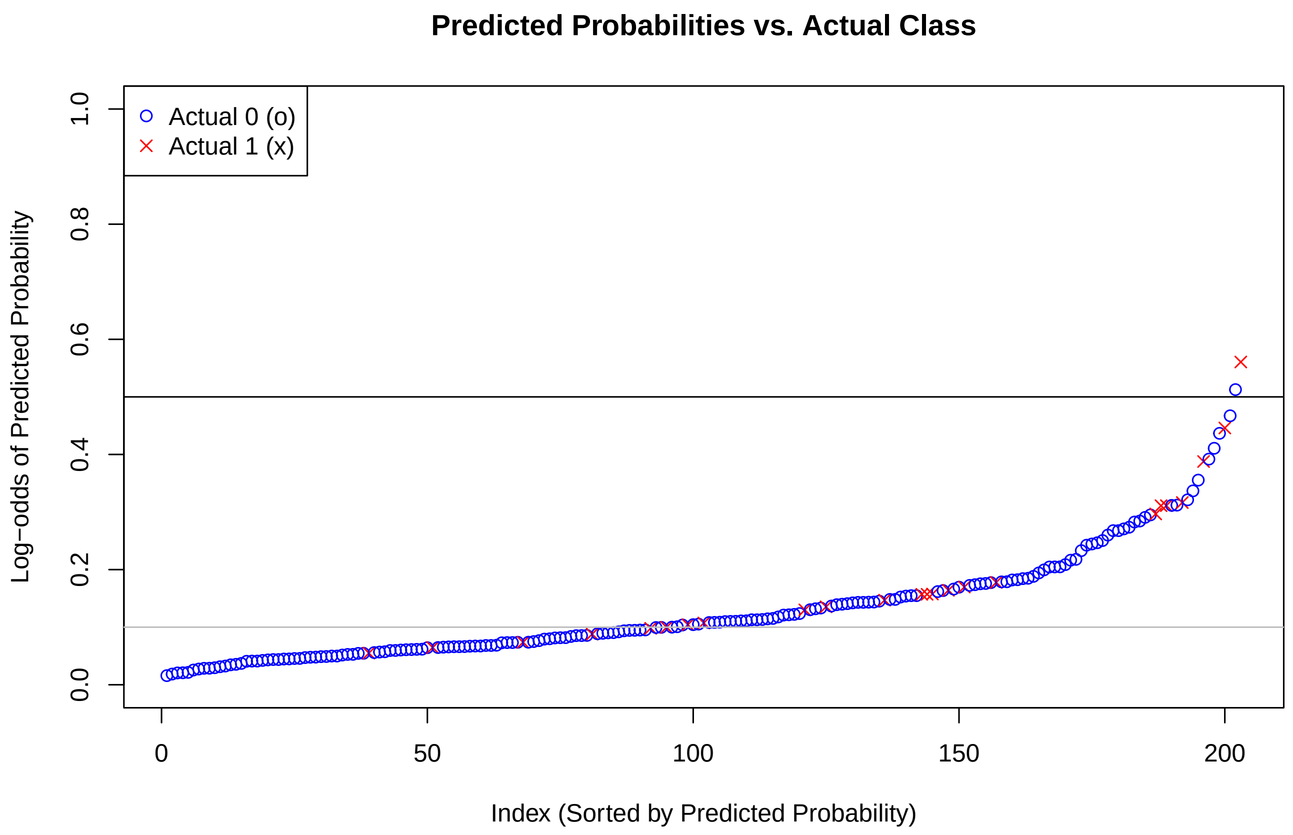


5.7.3 Application to the CHD Dataset

```r
## Load the pROC library for AUC calculation
## install.packages("pROC") # Uncomment to install if needed
library(pROC)

## --- 1. Split the data ---
set.seed(123) # for reproducibility
n <- nrow(CHD.data)
train_size <- floor(2/3 * n)
train_indices <- sample(1:n, size = train_size)
train_data <- CHD.data[train_indices, ]
test_data  <- CHD.data[-train_indices, ]

## --- 2. Refit the model on the training data ---
fit_train <- glm(
  chd ~ smk + cat + sbp + age + chl + ecg + hpt,
  data = train_data,
  family = binomial(link = "logit")
)

## --- 3. Make predictions on the test data ---
pred_probs <- predict(fit_train, newdata = test_data, type = "response")
```

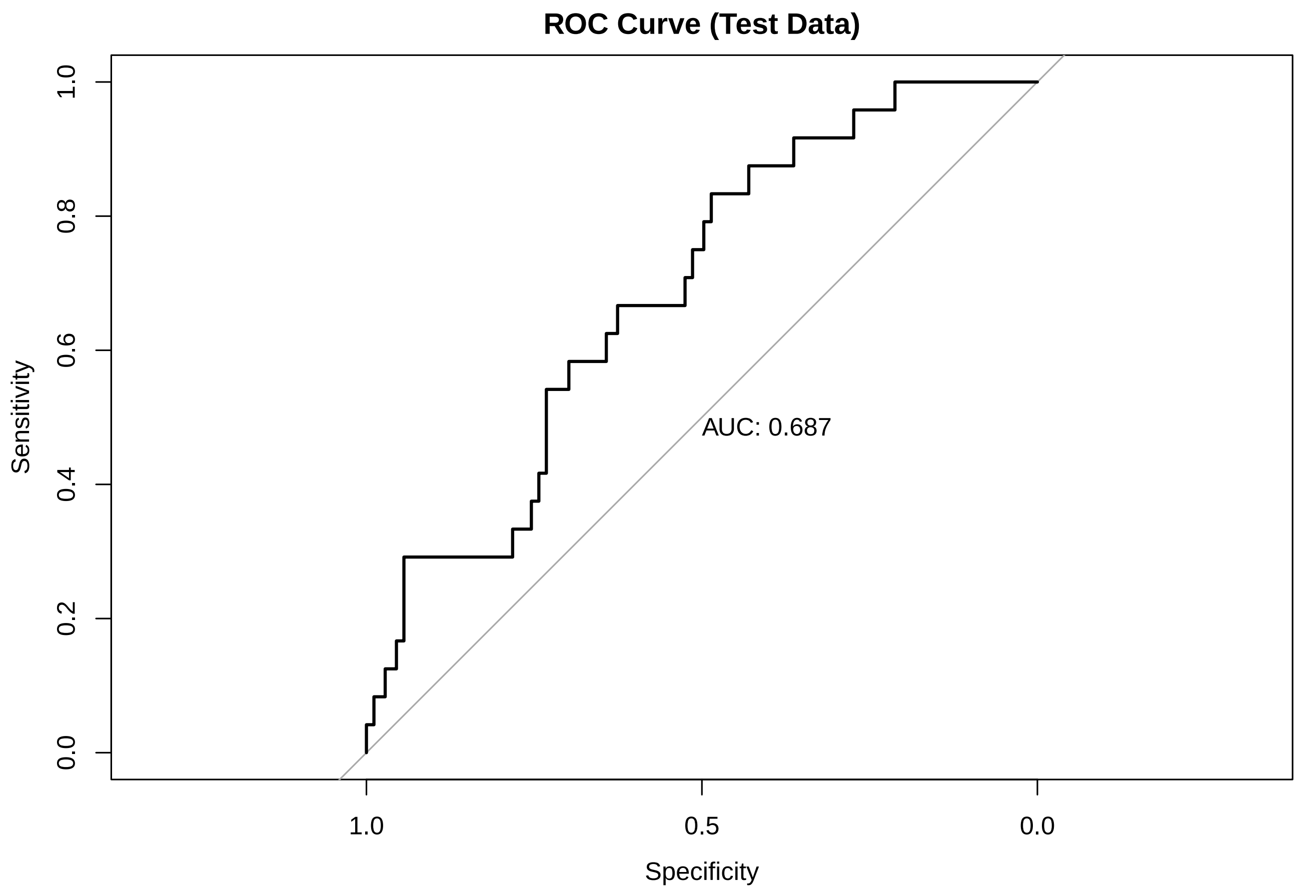
Plot Predictive Probabilities

```
## --- 5. Plot sorted predicted probabilities ---

## Create a data frame for plotting
plot_data <- data.frame(
  Prob = pred_probs,
  Actual = as.factor(test_data$chd)
)

## Sort by predicted probability
plot_data <- plot_data[order(plot_data$Prob), ]
plot_data$Rank <- 1:nrow(plot_data)

## Create the plot
## We use 'pch' (plot character) to set different symbols
## 'pch = 1' is 'o' (default)
## 'pch = 4' is 'x'
plot(
  plot_data$Rank,
  plot_data$Prob,
  pch = ifelse(plot_data$Actual == 0, 1, 4),
  col = ifelse(plot_data$Actual == 0, "blue", "red"),
  xlab = "Index (Sorted by Predicted Probability)",
  ylab = "Log-odds of Predicted Probability",
  main = "Predicted Probabilities vs. Actual Class",
  ylim = c(0,1)
)
abline(h=0.5)
abline(h=0.1, col="grey")

## Add a legend
legend(
  "topleft",
  legend = c("Actual 0 (o)", "Actual 1 (x)"),
  pch = c(1, 4),
  col = c("blue", "red")
)
```



ROC curve and Area Under the ROC (AUC)

```r
## 4b. Area Under the Curve (AUC)
roc_curve <- roc(test_data$chd, pred_probs, quiet = TRUE)

## Plot the ROC curve
plot(roc_curve, main = "ROC Curve (Test Data)", print.auc = TRUE)
```



```r
auc_value <- auc(roc_curve)
cat(paste("Area Under the Curve (AUC):", round(auc_value, 4), "\n\n"))
```

```
Area Under the Curve (AUC): 0.6872
```

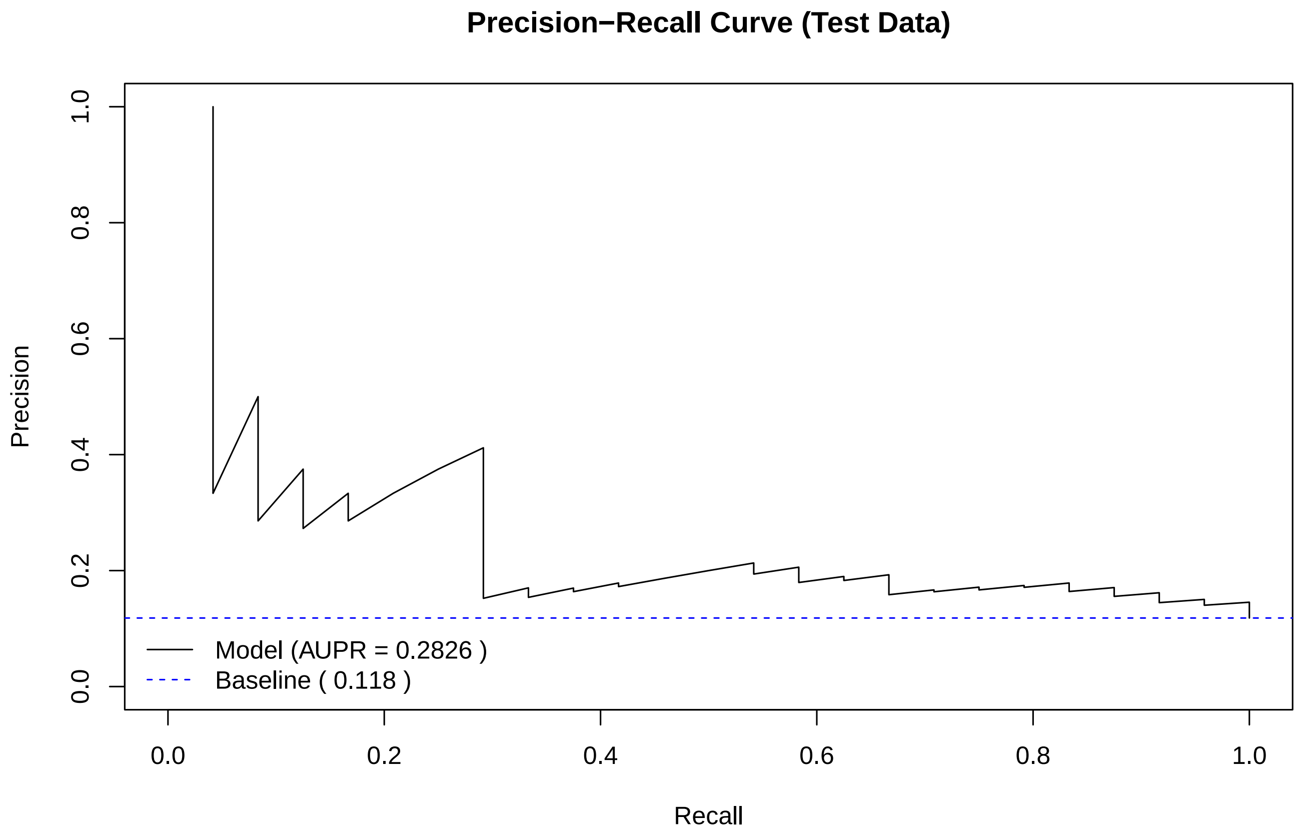
PR curve and Area Under PR Curve (AUPR)

```r
## Load the ROCR library
## install.packages("ROCR") # Uncomment to install if needed
library(ROCR)

## --- 1. Create a 'prediction' object ---
## 'prediction' takes all predictions and all true labels
## We use 'pred_probs' and 'test_data$chd' from the CHD data split
pred_obj <- prediction(pred_probs, test_data$chd)

## --- 2. Create a 'performance' object for PR ---
## "prec" is for precision, "rec" is for recall
```

```r
perf_pr <- performance(pred_obj, measure = "prec", x.measure = "rec")

## --- 3. Calculate Area Under the PR Curve (AUPR) ---
perf_auc <- performance(pred_obj, measure = "aucpr") # "aucpr" = Area Under PR Curve
aupr_value <- perf_auc@y.values[[1]]
cat(paste("Area Under PR Curve (AUPR):", round(aupr_value, 4), "\n"))
```

```
Area Under PR Curve (AUPR): 0.2826
```

```r
## --- 4. Plot the performance object ---
plot(perf_pr,
     main = "Precision-Recall Curve (Test Data)",
     xlim = c(0, 1),
     ylim = c(0, 1),
     col = "black")

## --- 5. Calculate and add the 'no-skill' baseline ---
baseline_precision <- sum(test_data$chd == 1) / length(test_data$chd)
abline(h = baseline_precision, col = "blue", lty = 2)

## --- 6. Add a legend with AUPR ---
legend("bottomleft",
       legend = c(
           paste("Model (AUPR =", round(aupr_value, 4), ")"),  # <-- MODIFIED LINE
           paste("Baseline (", round(baseline_precision, 3), ")")
       ),
       col = c("black", "blue"),
       lty = c(1, 2),
       bty = "n") # bty="n" removes the box
```


Precision−Recall Curve (Test Data)
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0
Model (AUPR = 0.2826)
Baseline (0.118)

6 One-factor Design

6.1 Completely Randomized Design

6.1.1 Plasma Etching Experiment

This section analyzes data from a **Completely Randomized Design (CRD)**. In a CRD, experimental units (in this case, the silicon wafers being etched) are assigned to treatments (the RF Power levels) completely at random. The primary goal is to determine if changing the RF Power level has a statistically significant effect on the mean etch rate.

6.1.1.1 Data and Visualization

We begin by loading the data into a single, tidy `data.frame`. The response variable, `rate`, contains all the etch rate observations. The predictor variable, `power`, is a **factor**, which is R's way of representing a categorical variable. This tells R to treat the different power levels as distinct groups.

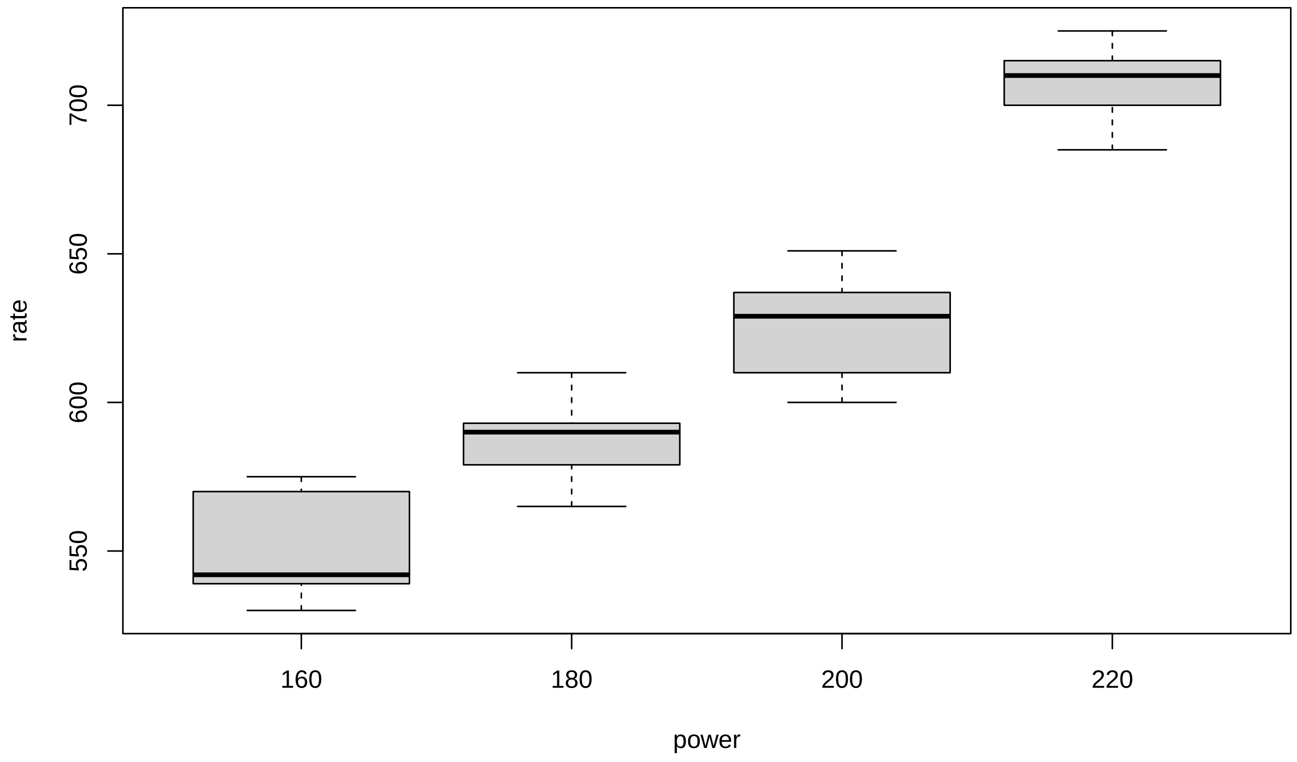
```r
## Define the data vectors
rate <- c(575, 542, 530, 539, 570, 565, 593, 590, 579, 610,
          600, 651, 610, 637, 629, 725, 700, 715, 685, 710)
power_levels <- c(160, 180, 200, 220)

## Create the data frame
etching_df <- data.frame(
  rate = rate,
  power = factor(rep(power_levels, each = 5))
)

## Display the first few rows
etching_df
```

```
  rate power
1  575   160
2  542   160
3  530   160
4  539   160
5  570   160
```

```
6    565    180
7    593    180
8    590    180
9    579    180
10   610    180
11   600    200
12   651    200
13   610    200
14   637    200
15   629    200
16   725    220
17   700    220
18   715    220
19   685    220
20   710    220
```

Grouped Boxplots

```
boxplot(rate~power, data=etching_df)
```



Using ggplot to visualize grouped data

```r
library(ggplot2)
library(dplyr) # Using dplyr for easier data manipulation

## Calculate group means and their start/end indices
mean_rates <- etching_df %>%
  mutate(obs_index = row_number()) %>%
  group_by(power) %>%
  summarise(
    mean_rate = mean(rate),
    x_start = min(obs_index) - 0.5,
    x_end = max(obs_index) + 0.5
  )

ggplot(etching_df, aes(x = 1:nrow(etching_df), y = rate, color = power)) +
  geom_point(size = 3, alpha = 0.7) + # Plot individual data points
  geom_segment(
    data = mean_rates,
    aes(x = x_start, xend = x_end, y = mean_rate, yend = mean_rate),
    linetype = "dashed",
    size = 1.2
  ) + # Add line segments for group means
  labs(
    title = "Etch Rate Observations by RF Power Level",
    x = "Observation Index",
    y = "Etch Rate",
    color = "RF Power (W)"
  ) +
  scale_color_brewer(palette = "Set1") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

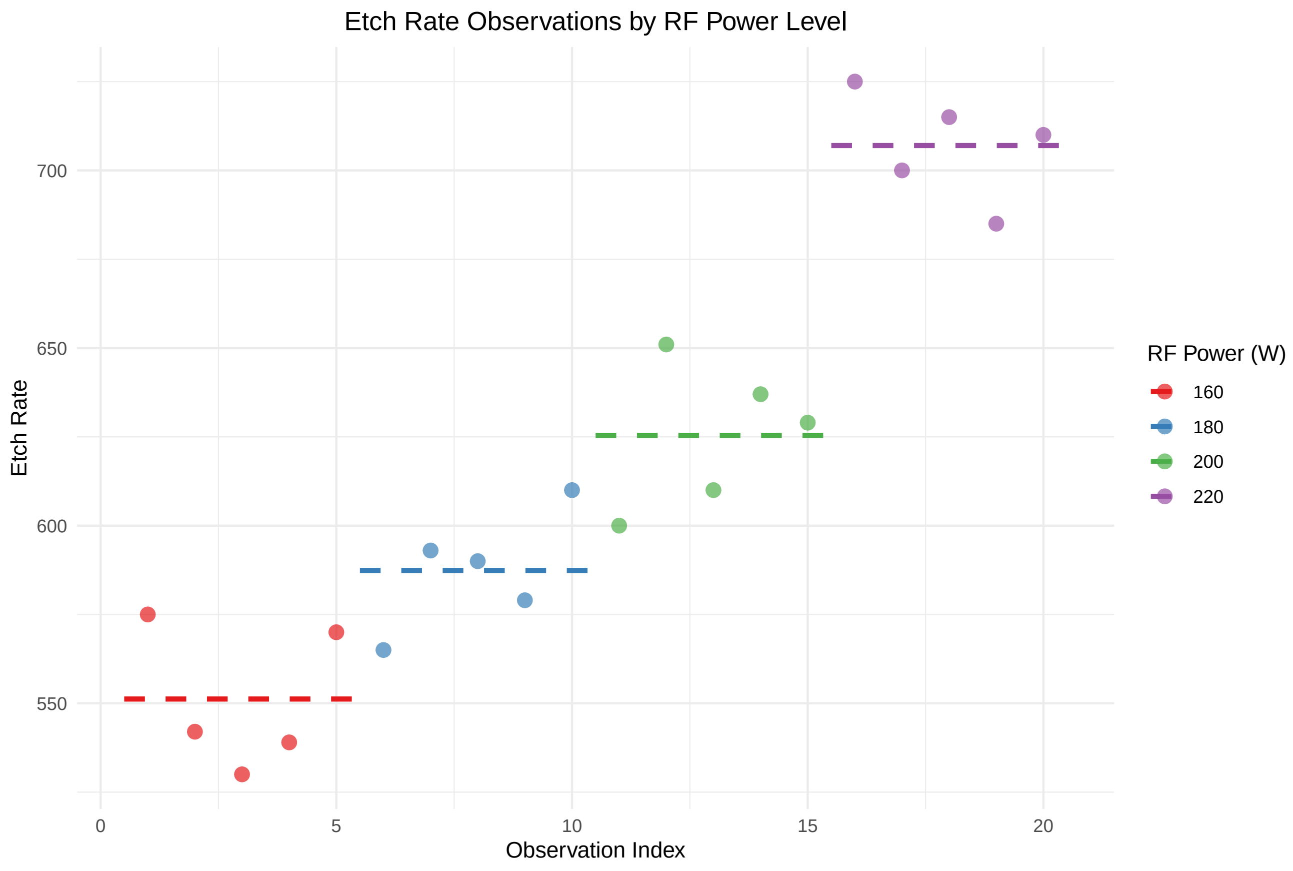


Figure 6.1: Index Plot of Etch Rate with Group-Specific Mean Lines

6.1.1.2 Model Fitting with Sum-to-Zero Constraint

We fit a linear model using the `lm()` function to perform an **Analysis of Variance (ANOVA)**. The model is specified as `rate ~ power`, and we now include the `data = etching_df` argument.

To get interpretable estimates for the treatment effects (τ_i), we use a **sum-to-zero constraint** (`contr.sum`), which forces the sum of the treatment effects to be zero ($\sum \tau_i = 0$).

```
fit <- lm(rate ~ power, data = etching_df, contrasts = list(power = contr.sum))
cat ("Model Matrix:\n")
```

```
Model Matrix:
```

```
model.matrix(fit)
```

```
   (Intercept) power1 power2 power3
1            1      1      0      0
```

```
2             1      1      0      0
3             1      1      0      0
4             1      1      0      0
5             1      1      0      0
6             1      0      1      0
7             1      0      1      0
8             1      0      1      0
9             1      0      1      0
10            1      0      1      0
11            1      0      0      1
12            1      0      0      1
13            1      0      0      1
14            1      0      0      1
15            1      0      0      1
16            1     -1     -1     -1
17            1     -1     -1     -1
18            1     -1     -1     -1
19            1     -1     -1     -1
20            1     -1     -1     -1
attr(,"assign")
[1] 0 1 1 1
attr(,"contrasts")
attr(,"contrasts")$power
    [,1] [,2] [,3]
160    1    0    0
180    0    1    0
200    0    0    1
220   -1   -1   -1
```

```
summary.fit <- summary(fit)
cat ("Summary of lm fitting results:\n")
```

```
Summary of lm fitting results:
```

```
summary.fit
```

```
Call:
lm(formula = rate ~ power, data = etching_df, contrasts = list(power = contr.sum))

Residuals:
   Min     1Q Median     3Q    Max 
 -25.4  -13.0    2.8   13.2   25.6 
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  617.750      4.085 151.234  < 2e-16 ***
power1       -66.550      7.075  -9.406 6.39e-08 ***
power2       -30.350      7.075  -4.290 0.000563 ***
power3         7.650      7.075   1.081 0.295602    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 18.27 on 16 degrees of freedom
Multiple R-squared:  0.9261,	Adjusted R-squared:  0.9122 
F-statistic:  66.8 on 3 and 16 DF,  p-value: 2.883e-09
```

6.1.1.3 Point Estimation of Parameters

The output of the model provides estimates for the overall mean ($\hat{\mu}$) and the treatment effects for the first k-1 levels ($\hat{\tau}_1, \hat{\tau}_2, \hat{\tau}_3$).

- $\hat{\mu}$ (the Intercept) is the estimate of the grand mean etch rate across all power levels.
- $\hat{\tau}_i$ is the estimated effect of the i-th power level, representing how much that level's mean deviates from the grand mean.

Using the sum-to-zero constraint, we can manually calculate the effect for the final level, $\hat{\tau}_4$.

```r
## Extract coefficients
est <- coef(fit)
tau4.hat <- -sum(est[-1])
taui.hat <- c(est[-1], tau4.hat)
print("Estimated Treatment Effects (tau_i):")
```

```
[1] "Estimated Treatment Effects (tau_i):"
```

```r
print(taui.hat)
```

```
power1 power2 power3        
-66.55 -30.35   7.65  89.25 
```

```r
## Estimates of treatment means (mu_i)
mu.hat <- est[1]
mui.hat <- mu.hat + taui.hat
print("Estimated Treatment Means (mu_i):")
```

```
[1] "Estimated Treatment Means (mu_i):"
```

```
print(mui.hat)
```

```
power1 power2 power3
 551.2  587.4  625.4  707.0
```

6.1.1.4 ANOVA Table

The ANOVA table partitions the total variation into variation **between** treatment groups (`power`) and variation **within** treatment groups (random error). The **p-value** (Pr(>F)) indicates if the treatment has a significant effect.

```
anova(fit)
```

```
Analysis of Variance Table

Response: rate
          Df Sum Sq Mean Sq F value    Pr(>F)
power      3  66871 22290.2  66.797 2.883e-09 ***
Residuals 16   5339   333.7
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

6.1.1.5 95% Confidence Intervals for Treatment Means

A **confidence interval** provides a range of plausible values for the true mean etch rate at each power level.

```
## Number of replicates
n <- 5
## Extract sqrt(MSE) and error df
sqrt.MSE <- summary.fit$sigma
DF <- fit$df.residual
## Find t-value
t.value <- qt(0.975, DF)
## Calculate CIs
CI.lower <- mui.hat - t.value * sqrt.MSE / sqrt(n)
CI.upper <- mui.hat + t.value * sqrt.MSE / sqrt(n)

## Display CIs
data.frame(Power_Level = power_levels, Mean = mui.hat, Lower_CI = CI.lower, Upper_CI = CI.upper)
```

```
       Power_Level  Mean Lower_CI Upper_CI
power1         160 551.2 533.8815 568.5185
power2         180 587.4 570.0815 604.7185
power3         200 625.4 608.0815 642.7185
               220 707.0 689.6815 724.3185
```

Alternatively, one can use a model without intercept

```
fit_nointercpt <- lm(rate ~ 0+power, data = etching_df)
summary(fit_nointercpt)
```

```
Call:
lm(formula = rate ~ 0 + power, data = etching_df)

Residuals:
   Min     1Q Median     3Q    Max 
 -25.4  -13.0    2.8   13.2   25.6 

Coefficients:
         Estimate Std. Error t value Pr(>|t|)    
power160  551.200      8.169   67.47   <2e-16 ***
power180  587.400      8.169   71.90   <2e-16 ***
power200  625.400      8.169   76.55   <2e-16 ***
power220  707.000      8.169   86.54   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 18.27 on 16 degrees of freedom
Multiple R-squared:  0.9993,    Adjusted R-squared:  0.9991 
F-statistic:  5768 on 4 and 16 DF,  p-value: < 2.2e-16
```

```
confint(fit_nointercpt)
```

```
            2.5 %   97.5 %
power160 533.8815 568.5185
power180 570.0815 604.7185
power200 608.0815 642.7185
power220 689.6815 724.3185
```

6.1.1.6 Comparison with Default "Treatment" Contrast

Fitting the model without specifying contrasts uses R's default ("treatment" contrast), which sets $\tau_1 = 0$. The fundamental results (ANOVA, treatment means) remain unchanged.

```r
fit1 <- lm(rate ~ power, data = etching_df)
summary(fit1)
```

```
Call:
lm(formula = rate ~ power, data = etching_df)

Residuals:
   Min     1Q Median     3Q    Max 
 -25.4  -13.0    2.8   13.2   25.6 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  551.200      8.169  67.471  < 2e-16 ***
power180      36.200     11.553   3.133  0.00642 ** 
power200      74.200     11.553   6.422 8.44e-06 ***
power220     155.800     11.553  13.485 3.73e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 18.27 on 16 degrees of freedom
Multiple R-squared:  0.9261,	Adjusted R-squared:  0.9122 
F-statistic:  66.8 on 3 and 16 DF,  p-value: 2.883e-09
```

6.1.1.7 Pairwise Comparisons

Since our ANOVA result was significant, we perform **post-hoc tests** to determine exactly which pairs of power levels have different means.

6.1.1.7.1 Tukey's HSD Test

Tukey's Honest Significant Difference (HSD) controls the **family-wise error rate**, adjusting p-values to account for multiple comparisons.

```r
fit.aov <- aov(rate ~ power, data = etching_df)
TukeyHSD(fit.aov)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = rate ~ power, data = etching_df)

$power
         diff        lwr       upr     p adj
180-160  36.2   3.145624  69.25438 0.0294279
200-160  74.2  41.145624 107.25438 0.0000455
220-160 155.8 122.745624 188.85438 0.0000000
200-180  38.0   4.945624  71.05438 0.0215995
220-180 119.6  86.545624 152.65438 0.0000001
220-200  81.6  48.545624 114.65438 0.0000146
```

6.1.1.7.2 Fisher's LSD Test

The **Fisher's Least Significant Difference (LSD)** test does not control the family-wise error rate but is more powerful.

```r
with(etching_df, pairwise.t.test(rate, power, p.adj = "none"))
```

```


    Pairwise comparisons using t tests with pooled SD

data:  rate and power

    160     180     200
180 0.0064  -       -
200 8.4e-06 0.0046  -
220 3.7e-10 1.7e-08 2.7e-06

P value adjustment method: none
```

6.1.1.8 Checking Model Assumptions

The validity of our ANOVA results depends on three key assumptions about the model's residuals. We use diagnostic plots to check them.

```r
r <- rstudent(fit)
fitted <- fitted.values(fit)
```

6.1.1.8.1 Normality of Residuals

A **Normal Q-Q plot** is used to check if the residuals are normally distributed. The points should fall closely along the straight diagonal line.

```r
qqnorm(r)
qqline(r)
```

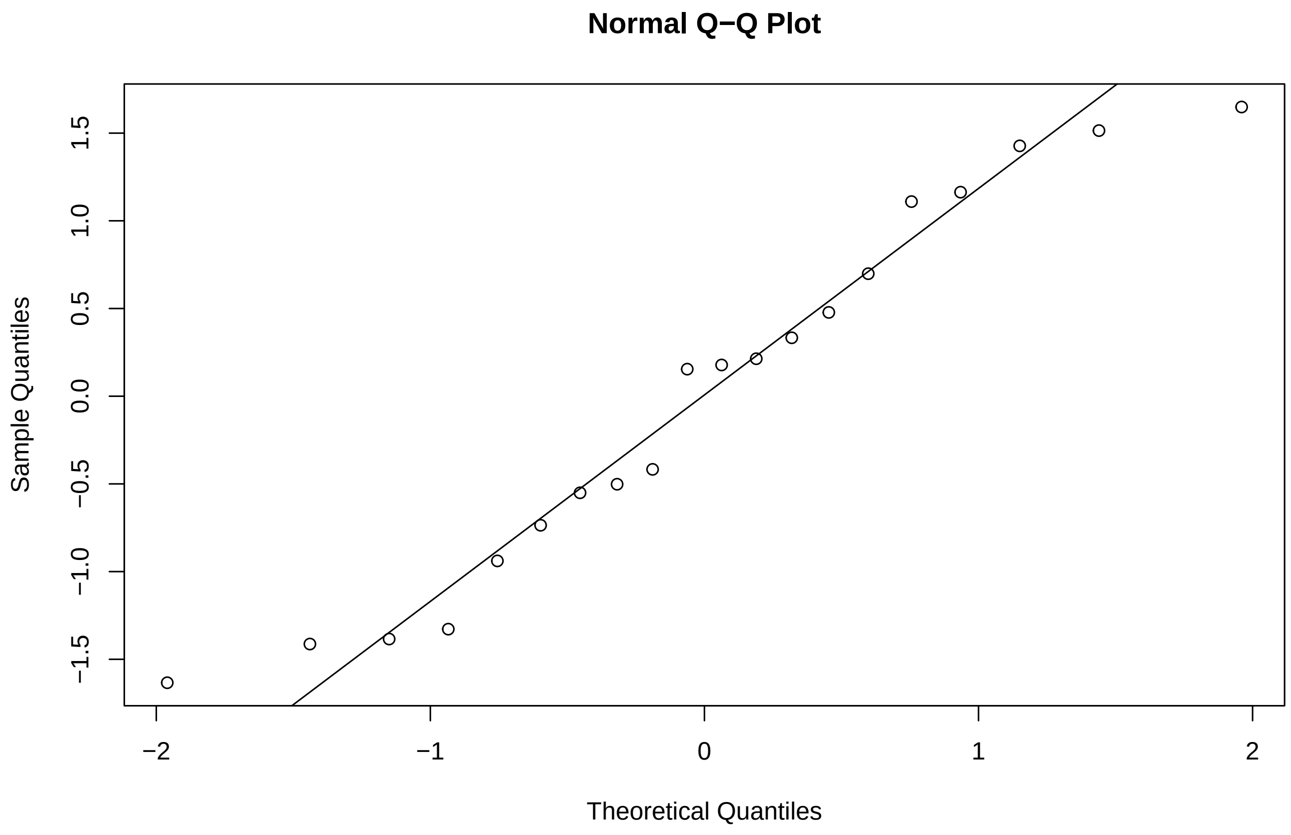


Figure 6.2: Normal Q-Q plot of standardized residuals.

6.1.1.8.2 Independence of Residuals

A plot of **residuals versus run order** helps check for independence. We look for random scatter around the zero line.

```r
plot(r, ylab = "Standardized residuals", xlab = "Run order",
     main = "Plot of residuals vs. run order")
abline(h = 0)
```

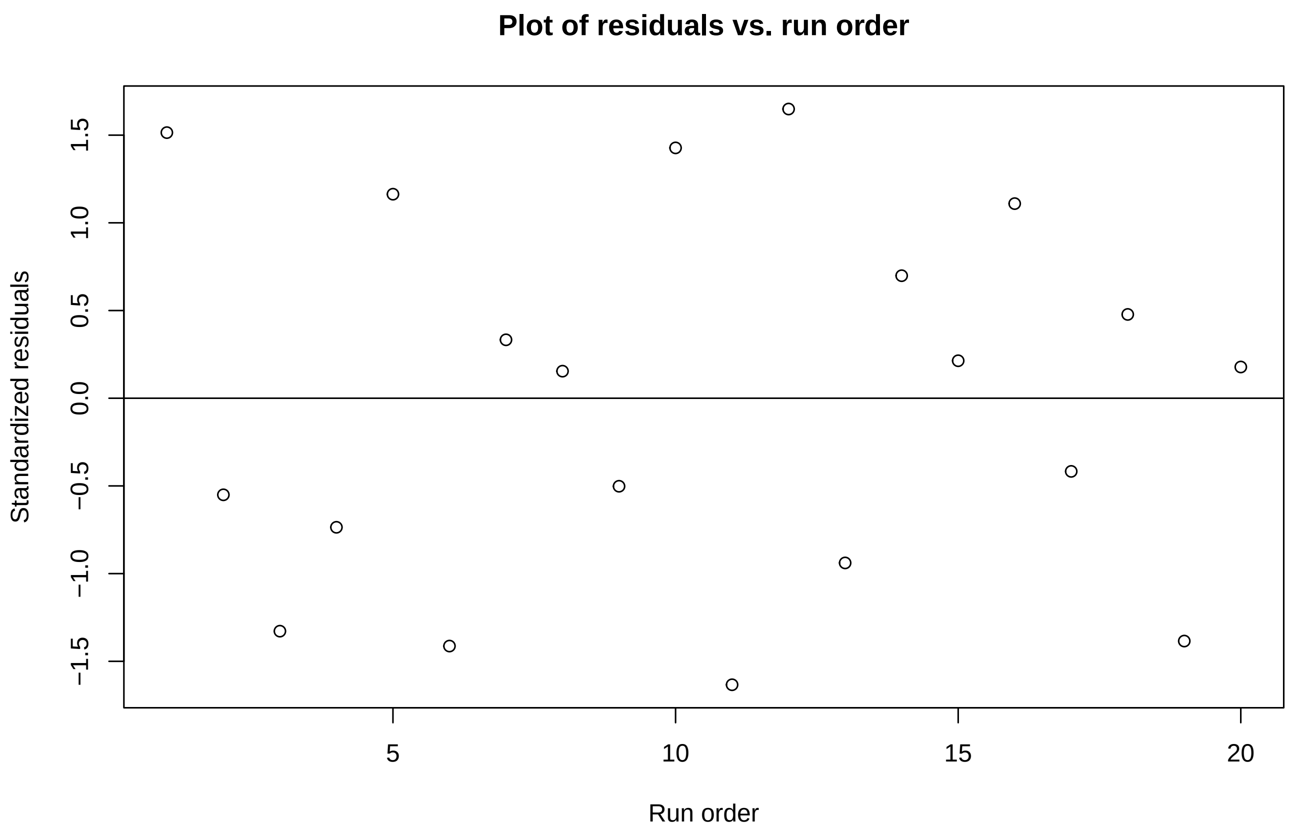


Figure 6.3: Standardized residuals vs. run order.

6.1.1.8.3 Constant Variance (Homoscedasticity)

A plot of **residuals versus fitted values** helps check for constant variance. The spread of residuals should be roughly constant across all fitted values.

```r
plot(fitted, r, ylab = "Standardized residuals",
     xlab = "Fitted values", main = "Plot of residuals vs. fitted values")
abline(h = 0)
```

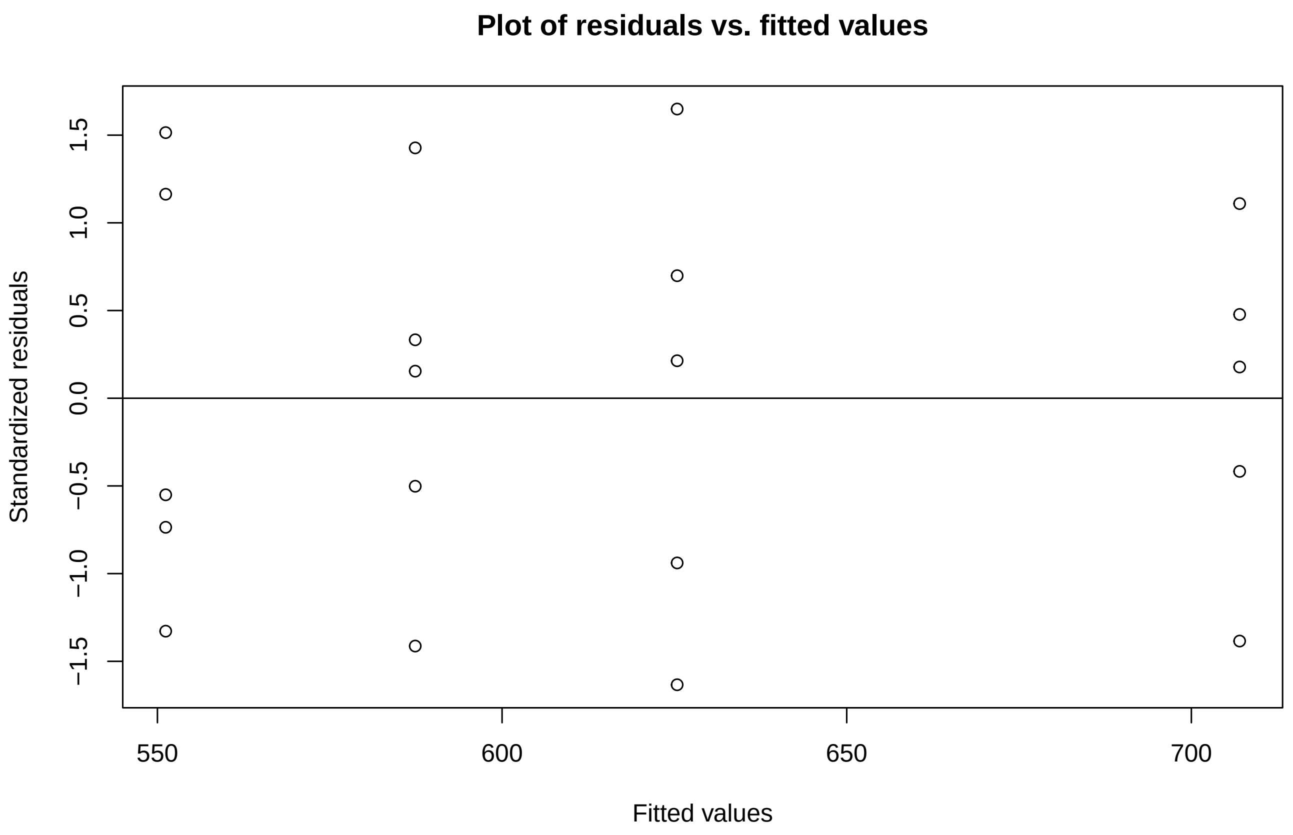


Figure 6.4: Standardized residuals vs. fitted values.

6.2 Unbalanced Designs with nequal Sample Sizes

The ANOVA framework also handles **unbalanced designs**. We again start by creating a data frame.

```r
## Create the data frame
bricks_df <- data.frame(
  density = c(21.8, 21.9, 21.7, 21.6, 21.7,
              21.7, 21.4, 21.5, 21.4,
              21.9, 21.8, 21.8, 21.6, 21.5,
              21.9, 21.7, 21.8, 21.4),
  temperature = factor(c(rep(100, 5), rep(125, 4), rep(150, 5), rep(175, 4)))
)

bricks_df
```

```
   density temperature
1     21.8         100
2     21.9         100
```

```
3     21.7          100
4     21.6          100
5     21.7          100
6     21.7          125
7     21.4          125
8     21.5          125
9     21.4          125
10    21.9          150
11    21.8          150
12    21.8          150
13    21.6          150
14    21.5          150
15    21.9          175
16    21.7          175
17    21.8          175
18    21.4          175
```

```r
## Fit the model and get the ANOVA table
fit2 <- lm(density ~ temperature, data = bricks_df)
summary(fit2)
```

```

Call:
lm(formula = density ~ temperature, data = bricks_df)

Residuals:
   Min     1Q Median     3Q    Max 
-0.300 -0.100  0.000  0.095  0.200 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)    21.74000    0.07171 303.150   <2e-16 ***
temperature125 -0.24000    0.10757  -2.231   0.0425 *  
temperature150 -0.02000    0.10142  -0.197   0.8465    
temperature175 -0.04000    0.10757  -0.372   0.7156    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1604 on 14 degrees of freedom
Multiple R-squared:  0.3025,	Adjusted R-squared:  0.153 
F-statistic: 2.024 on 3 and 14 DF,  p-value: 0.1569
```

```
anova(fit2)
```

```
Analysis of Variance Table

Response: density
            Df  Sum Sq  Mean Sq F value Pr(>F)
temperature  3 0.15611 0.052037  2.0237 0.1569
Residuals   14 0.36000 0.025714
```

In this case, the large p-value (0.133) indicates that there is no statistically significant evidence that firing temperature affects brick density.

6.3 Randomized Complete Block Design

6.3.1 Vascular Graft Experiment

This section analyzes a **Randomized Complete Block Design (RCBD)**, used to control for a known source of variability (here, “batches of resin,” treated as **blocks**).

6.3.1.1 Data and Visulization

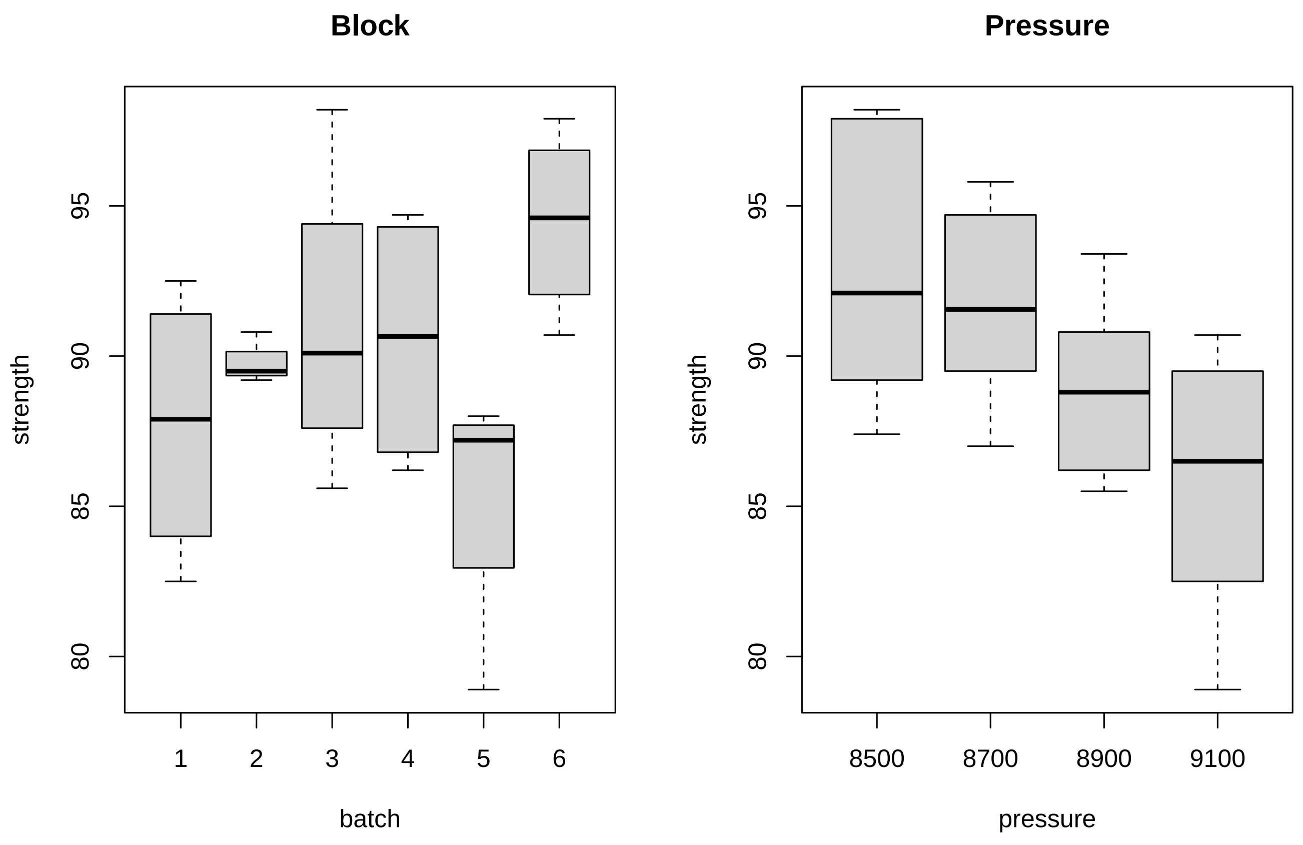
We structure the data in a `data.frame` to identify the response, treatment (`pressure`), and block (`batch`) for each observation.

```
## Define data vectors
strength <- c(90.3, 89.2, 98.2, 93.9, 87.4, 97.9,
              92.5, 89.5, 90.6, 94.7, 87.0, 95.8,
              85.5, 90.8, 89.6, 86.2, 88.0, 93.4,
              82.5, 89.5, 85.6, 87.4, 78.9, 90.7)
pressure_levels <- rep(c(8500, 8700, 8900, 9100), each = 6)
batch_levels <- rep(1:6, 4)

## Create the data frame
graft_df <- data.frame(
  strength = strength,
  pressure = factor(pressure_levels),
  batch = factor(batch_levels)
)


graft_df
```

```
   strength pressure batch
1      90.3     8500     1
2      89.2     8500     2
3      98.2     8500     3
4      93.9     8500     4
5      87.4     8500     5
6      97.9     8500     6
7      92.5     8700     1
8      89.5     8700     2
9      90.6     8700     3
10     94.7     8700     4
11     87.0     8700     5
12     95.8     8700     6
13     85.5     8900     1
14     90.8     8900     2
15     89.6     8900     3
16     86.2     8900     4
17     88.0     8900     5
18     93.4     8900     6
19     82.5     9100     1
20     89.5     9100     2
21     85.6     9100     3
22     87.4     9100     4
23     78.9     9100     5
24     90.7     9100     6
```

Visualize the Block and Treatment Effects

```r
par (mfrow = c(1,2))
#boxplot
plot(strength ~ batch, data=graft_df , main = "Block")
plot(strength ~ pressure, data=graft_df , main = "Pressure")
```



Interaction Plots

```r
ggplot(graft_df, aes(x = pressure, y = strength, group = batch, color = batch)) +
  stat_summary(fun = mean, geom = "line", size = 1) +
  stat_summary(fun = mean, geom = "point", size = 3) +
  labs(
    title = "Interaction Plot: Batch and Pressure",
    x = "Pressue",
    y = "Strength",
    color = "Batch"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

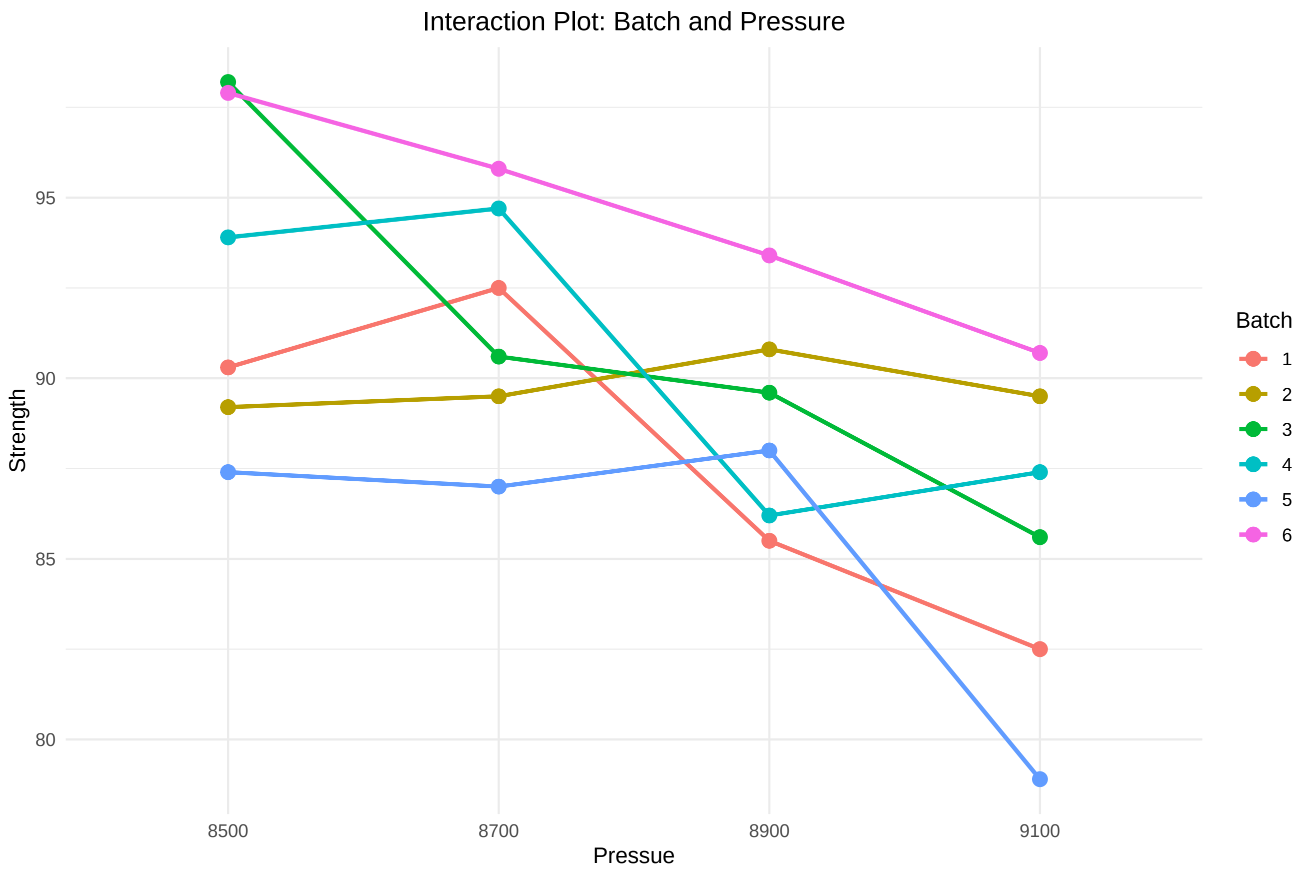


Figure 6.5: Interaction between Material Type and Temperature.

6.3.1.2 Model Fitting and ANOVA

The model `strength ~ pressure + batch` partitions the total variance into treatment, block, and error components. Our primary interest is in the significance of the `pressure` factor.

```r
rcbd.fit1 <- aov(strength ~ pressure + batch, data = graft_df)
anova(rcbd.fit1)
```

```
Analysis of Variance Table

Response: strength
          Df Sum Sq Mean Sq F value   Pr(>F)
pressure   3 178.17  59.390  8.1071 0.001916 **
batch      5 192.25  38.450  5.2487 0.005532 **
Residuals 15 109.89   7.326
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The small p-value for `pressure` (0.0019) provides strong evidence that extrusion pressure significantly affects graft strength after accounting for batch differences.

6.3.1.3 Model Adequacy Checks

The assumptions for an RCBD are the same as for a CRD. We perform the same diagnostic checks.

```r
rcbd.r1 <- rstudent(rcbd.fit1)
rcbd.fitted1 <- fitted.values(rcbd.fit1)

qqnorm(rcbd.r1, main = "Normal Q-Q Plot")
qqline(rcbd.r1)
plot(rcbd.fitted1, rcbd.r1, ylab = "Standardized residuals",
     xlab = "Fitted values", main = "Residuals vs. Fitted")
abline(h = 0)
plot(graft_df$pressure, rcbd.r1, ylab = "Standardized residuals",
     xlab = "Extrusion pressure", main = "Residuals vs. Treatment")
abline(h = 0)
plot(graft_df$batch, rcbd.r1, ylab = "Standardized residuals",
     xlab = "Batches of raw material", main = "Residuals vs. Block")
abline(h = 0)
```

6.3.1.4 Pairwise Comparisons

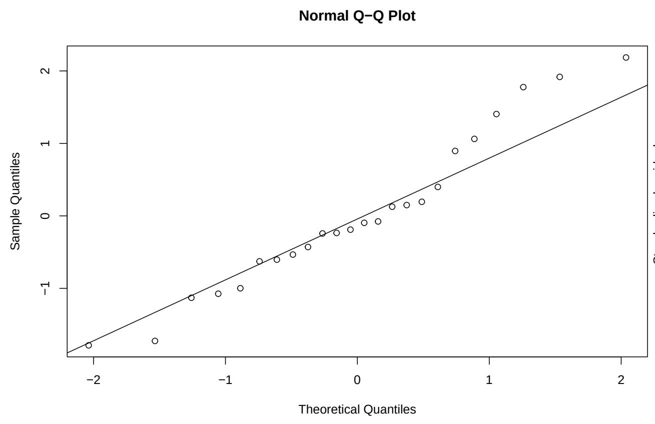
Again, since the treatment factor (`pressure`) is significant, we perform post-hoc tests.

6.3.1.4.1 Tukey's HSD Test

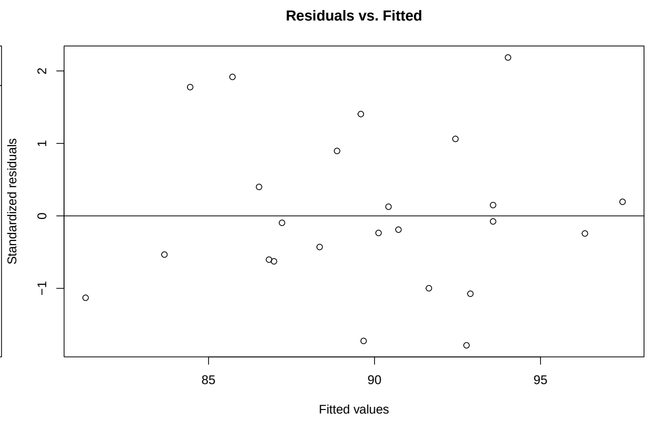
Tukey's HSD compares all pairs of treatment levels while controlling the family-wise error rate.

```r
TukeyHSD(rcbd.fit1, which = "pressure")
```

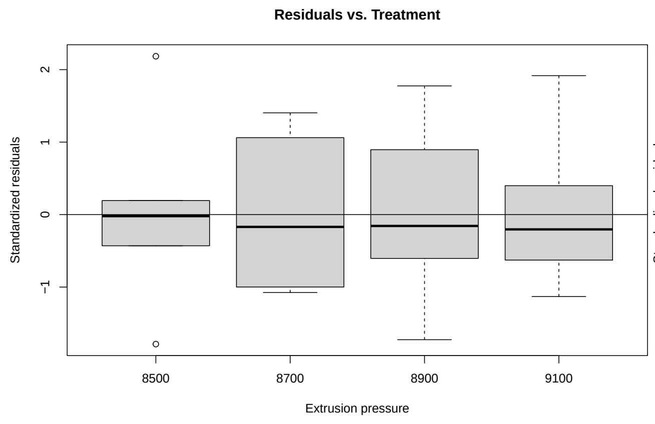
```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = strength ~ pressure + batch, data = graft_df)

$pressure
               diff        lwr       upr     p adj
8700-8500 -1.133333  -5.637161  3.370495 0.8854831
8900-8500 -3.900000  -8.403828  0.603828 0.1013084
9100-8500 -7.050000 -11.553828 -2.546172 0.0020883
```



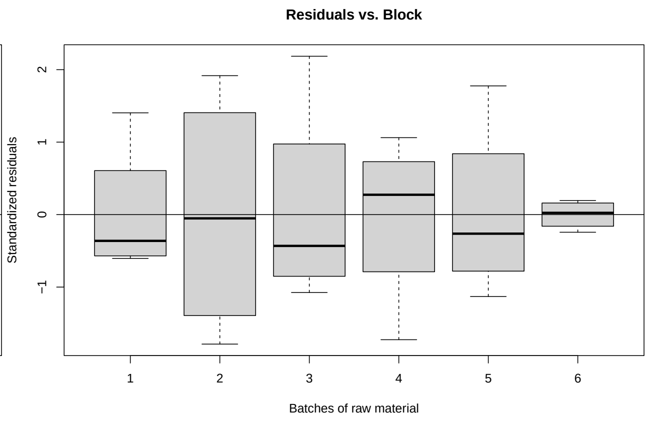
(a) Normal Q-Q Plot



(a) Residuals vs. Fitted Values



(a) Residuals vs. Treatment (Pressure)



(a) Residuals vs. Block (Batch)

```
8900-8700 -2.766667  -7.270495  1.737161 0.3245644
9100-8700 -5.916667 -10.420495 -1.412839 0.0086667
9100-8900 -3.150000  -7.653828  1.353828 0.2257674
```

6.3.1.4.2 Fisher's LSD Test

The `LSD.test()` function from the `agricolae` package correctly handles the error structure of an RCBD.

```
## install.packages("agricolae")
library(agricolae)

out <- LSD.test(rcbd.fit1, trt = "pressure", p.adj = "none", group = FALSE)
print(out$comparison)
```

```
            difference pvalue signif.        LCL       UCL
8500 - 8700   1.133333 0.4795         -2.1974047  4.464071
8500 - 8900   3.900000 0.0247       *  0.5692620  7.230738
8500 - 9100   7.050000 0.0004     ***  3.7192620 10.380738
8700 - 8900   2.766667 0.0970       . -0.5640714  6.097405
8700 - 9100   5.916667 0.0018      **  2.5859286  9.247405
8900 - 9100   3.150000 0.0621       . -0.1807380  6.480738
```

7 Two-Factor Factorial Design

7.1 Battery Design Experiment

This analysis explores data from a **two-factor factorial experiment** designed to assess the lifespan of a battery. The experiment investigates two factors: **material type** (with 3 levels) and **operating temperature** (with 3 levels: 15°C, 70°C, and 125°C). The primary goal is to understand not only how each factor individually affects battery life but, more importantly, whether the effect of temperature depends on the material type used. This combined effect is known as an **interaction**.

7.2 Data Setup and Preparation

First, we organize the raw data into a structured `data.frame`. This is a best practice in R that makes the data easier to manage and the code more readable. We create columns for the response variable `life` and the two factors, `material` and `temperature`, ensuring they are treated as categorical variables (factors) for the analysis.

```
## Response variable: battery life
life <- c(130,155,74,180,  34,40,80,75,   20,70,82,58,
          150,188,159,126, 136,122,106,115, 25,70,58,45,
          138,110,168,160, 174,120,150,139, 96,104,82,60)

## Create the data frame
battery_df <- data.frame(
  life = life,
  material = factor(rep(1:3, each = 12)),
  temperature = factor(rep(rep(c(15, 70, 125), each = 4), 3))
)

## Preview the data
battery_df
```

```
   life material temperature
1   130        1          15
2   155        1          15
3    74        1          15
```

```
4   180          1          15
5    34          1          70
6    40          1          70
7    80          1          70
8    75          1          70
9    20          1         125
10   70          1         125
11   82          1         125
12   58          1         125
13  150          2          15
14  188          2          15
15  159          2          15
16  126          2          15
17  136          2          70
18  122          2          70
19  106          2          70
20  115          2          70
21   25          2         125
22   70          2         125
23   58          2         125
24   45          2         125
25  138          3          15
26  110          3          15
27  168          3          15
28  160          3          15
29  174          3          70
30  120          3          70
31  150          3          70
32  139          3          70
33   96          3         125
34  104          3         125
35   82          3         125
36   60          3         125
```

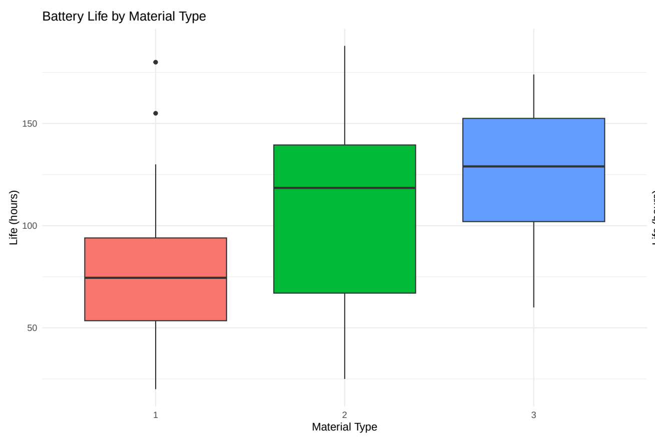
7.3 Exploratory Data Analysis and Visualization

Before fitting a formal model, we visualize the data to get an intuition for the relationships between the factors and the response.

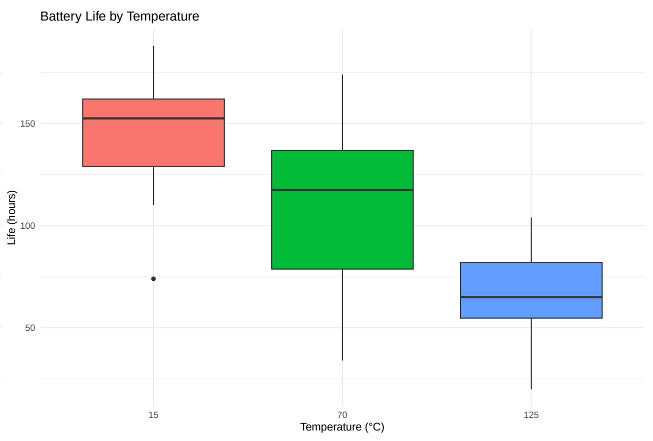
7.4 Boxplots of Main Effects

Boxplots are excellent for examining the distribution of battery life for each level of our factors independently. This gives us a preliminary look at the **main effects**—the individual impact of material type and temperature.

```r
library(ggplot2)

## Boxplot for Material Type
ggplot(battery_df, aes(x = material, y = life, fill = material)) +
  geom_boxplot() +
  labs(title = "Battery Life by Material Type", x = "Material Type", y = "Life (hours)") +
  theme_minimal() +
  theme(legend.position = "none")
## Boxplot for Temperature
ggplot(battery_df, aes(x = temperature, y = life, fill = temperature)) +
  geom_boxplot() +
  labs(title = "Battery Life by Temperature", x = "Temperature (°C)", y = "Life (hours)") +
  theme_minimal() +
  theme(legend.position = "none")
```



(a) Distribution of Battery Life by Material and Temperature.



(a) Distribution of Battery Life by Material and Temperature.

7.5 Interaction Plot

The most crucial plot for a factorial experiment is the **interaction plot**. It displays the mean battery life for each combination of material and temperature. If the lines are parallel, it suggests there is no interaction. If the lines are not parallel (i.e., they cross or diverge), it indicates that the effect of temperature on battery life is different for each material type, signaling a likely interaction.

```
ggplot(battery_df, aes(x = temperature, y = life, group = material, color = material)) +
  stat_summary(fun = mean, geom = "line", size = 1) +
  stat_summary(fun = mean, geom = "point", size = 3) +
  labs(
    title = "Interaction Plot: Material Type and Temperature",
    x = "Temperature (°C)",
    y = "Average Battery Life (hours)",
    color = "Material Type"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

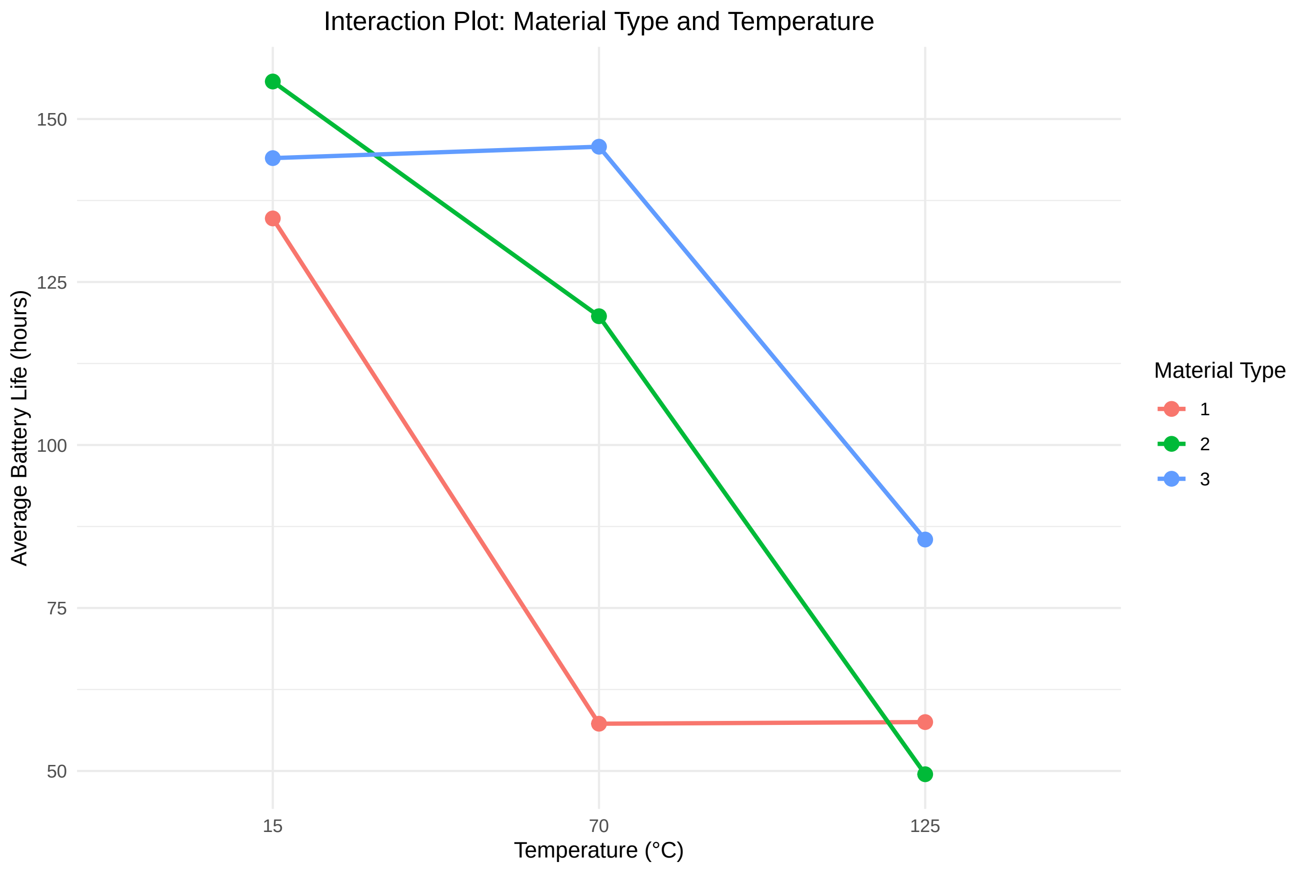


Figure 7.3: Interaction between Material Type and Temperature.

The non-parallel lines in the plot strongly suggest that a significant interaction effect is present. Specifically, the performance of Material 3 drops less dramatically with increasing temperature compared to Materials 1 and 2.

7.6 Model Fitting and Analysis of Variance (ANOVA)

We now fit a linear model to formally test the significance of the main effects and the interaction term. The model `life ~ material * temperature` is shorthand for `life ~ material + temperature + material:temperature`. We use a sum-to-zero contrast (`contr.sum`) for balanced interpretation of the effects. The **ANOVA table** will tell us if the variation caused by our factors is statistically significant compared to the random variation in the data.

```r
## Fit the full factorial model
battery_fit <- lm(life ~ material * temperature,
                  data = battery_df,
                  contrasts = list(material = contr.sum, temperature = contr.sum))

summary(battery_fit)
```

```

Call:
lm(formula = life ~ material * temperature, data = battery_df,
    contrasts = list(material = contr.sum, temperature = contr.sum))

Residuals:
    Min      1Q  Median      3Q     Max 
-60.750 -14.625   1.375  17.938  45.250 

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)    
(Intercept)             105.528      4.331  24.367  < 2e-16 ***
material1               -22.361      6.125  -3.651  0.00111 ** 
material2                 2.806      6.125   0.458  0.65057    
temperature1             39.306      6.125   6.418  7.1e-07 ***
temperature2              2.056      6.125   0.336  0.73975    
material1:temperature1   12.278      8.662   1.417  0.16778    
material2:temperature1    8.111      8.662   0.936  0.35735    
material1:temperature2  -27.972      8.662  -3.229  0.00325 ** 
material2:temperature2    9.361      8.662   1.081  0.28936    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 25.98 on 27 degrees of freedom
Multiple R-squared:  0.7652,	Adjusted R-squared:  0.6956 
F-statistic:    11 on 8 and 27 DF,  p-value: 9.426e-07
```

```r
## Generate the ANOVA table
anova(battery_fit)
```

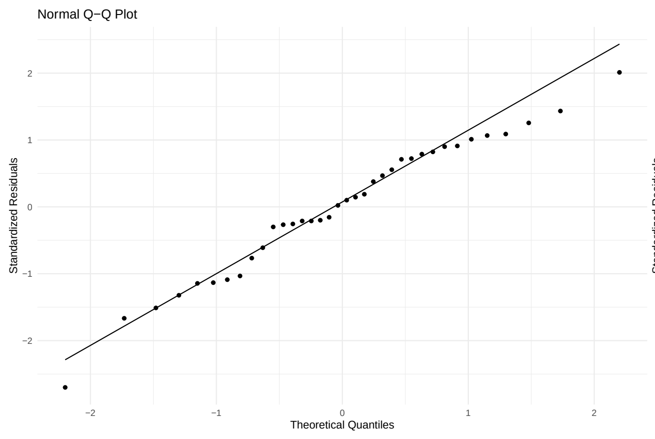
```
Analysis of Variance Table

Response: life
                     Df Sum Sq Mean Sq F value    Pr(>F)    
material              2  10684  5341.9  7.9114  0.001976 ** 
temperature           2  39119 19559.4 28.9677 1.909e-07 ***
material:temperature  4   9614  2403.4  3.5595  0.018611 *  
Residuals            27  18231   675.2                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The ANOVA table shows very small p-values (`Pr(>F)`) for `material`, `temperature`, and, most importantly, the `material:temperature` interaction. This confirms our visual inspection: all effects are statistically significant. **Because the interaction is significant, our interpretation should focus on the interaction itself rather than the main effects in isolation.**

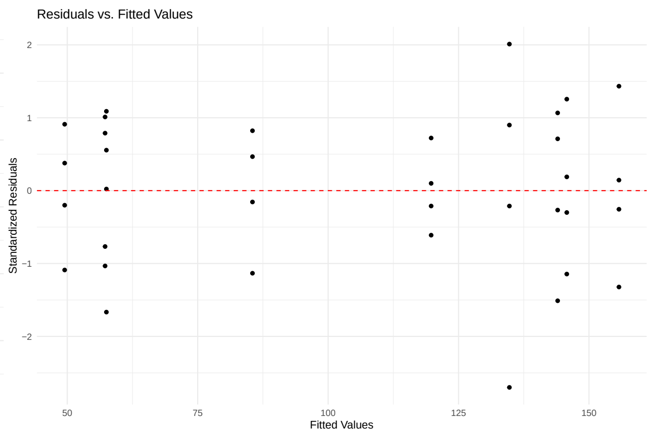
7.7 Model Adequacy Checks

The validity of our ANOVA results depends on the model's residuals meeting certain assumptions (normality, constant variance, independence). We check these with diagnostic plots.

```r
## Extract standardized residuals and fitted values
battery_fit_diag <- data.frame(
  residuals = rstandard(battery_fit),
  fitted = fitted.values(battery_fit)
)

## Normal Q-Q Plot
p1 <- ggplot(battery_fit_diag, aes(sample = residuals)) +
  stat_qq() +
  stat_qq_line() +
  labs(title = "Normal Q-Q Plot", x = "Theoretical Quantiles", y = "Standardized Residuals") +
  theme_minimal()

## Residuals vs. Fitted Plot
p2 <- ggplot(battery_fit_diag, aes(x = fitted, y = residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Residuals vs. Fitted Values", x = "Fitted Values", y = "Standardized Residuals
  theme_minimal()

p1
p2
```



(a) Diagnostic plots for the battery life model.



(a) Diagnostic plots for the battery life model.

The Normal Q-Q plot shows the points falling roughly along the line, suggesting the normality assumption is met. The Residuals vs. Fitted plot shows a random scatter of points around the zero line, indicating that the variance is reasonably constant. The model assumptions appear to be satisfied.

7.8 Post-Hoc Analysis: Pairwise Comparisons

Since the interaction is significant, we must compare the means of the nine specific treatment combinations (3 materials × 3 temperatures). Simply comparing the average effect of Material 1 vs. Material 2 would be misleading, as that difference depends on the temperature.

7.9 Tukey's HSD Test

Tukey's Honest Significant Difference (HSD) test is a post-hoc test that compares all possible pairs of means while controlling the family-wise error rate. We apply it to an `aov` model object. The output for the `material:temperature` interaction shows which specific combinations are significantly different from one another.

```
## Fit the model using aov() for Tukey's test
battery_aov <- aov(life ~ material * temperature, data = battery_df)
```

```r
## Perform Tukey's HSD test
TukeyHSD(battery_aov)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = life ~ material * temperature, data = battery_df)

$material
        diff        lwr      upr     p adj
2-1 25.16667 -1.135677 51.46901 0.0627571
3-1 41.91667 15.614323 68.21901 0.0014162
3-2 16.75000 -9.552344 43.05234 0.2717815

$temperature
            diff        lwr       upr     p adj
70-15  -37.25000  -63.55234 -10.94766 0.0043788
125-15 -80.66667 -106.96901 -54.36432 0.0000001
125-70 -43.41667  -69.71901 -17.11432 0.0009787

$`material:temperature`
               diff          lwr        upr     p adj
2:15-1:15     21.00  -40.823184  82.823184 0.9616404
3:15-1:15      9.25  -52.573184  71.073184 0.9998527
1:70-1:15    -77.50 -139.323184 -15.676816 0.0065212
2:70-1:15    -15.00  -76.823184  46.823184 0.9953182
3:70-1:15     11.00  -50.823184  72.823184 0.9994703
1:125-1:15   -77.25 -139.073184 -15.426816 0.0067471
2:125-1:15   -85.25 -147.073184 -23.426816 0.0022351
3:125-1:15   -49.25 -111.073184  12.573184 0.2016535
3:15-2:15    -11.75  -73.573184  50.073184 0.9991463
1:70-2:15    -98.50 -160.323184 -36.676816 0.0003449
2:70-2:15    -36.00  -97.823184  25.823184 0.5819453
3:70-2:15    -10.00  -71.823184  51.823184 0.9997369
1:125-2:15   -98.25 -160.073184 -36.426816 0.0003574
2:125-2:15  -106.25 -168.073184 -44.426816 0.0001152
3:125-2:15   -70.25 -132.073184  -8.426816 0.0172076
1:70-3:15    -86.75 -148.573184 -24.926816 0.0018119
2:70-3:15    -24.25  -86.073184  37.573184 0.9165175
3:70-3:15      1.75  -60.073184  63.573184 1.0000000
1:125-3:15   -86.50 -148.323184 -24.676816 0.0018765
2:125-3:15   -94.50 -156.323184 -32.676816 0.0006078
3:125-3:15   -58.50 -120.323184   3.323184 0.0742711
2:70-1:70     62.50    0.676816 124.323184 0.0460388
```

```
3:70-1:70     88.50   26.676816 150.323184 0.0014173
1:125-1:70     0.25  -61.573184  62.073184 1.0000000
2:125-1:70    -7.75  -69.573184  54.073184 0.9999614
3:125-1:70    28.25  -33.573184  90.073184 0.8281938
3:70-2:70     26.00  -35.823184  87.823184 0.8822881
1:125-2:70   -62.25 -124.073184  -0.426816 0.0474675
2:125-2:70   -70.25 -132.073184  -8.426816 0.0172076
3:125-2:70   -34.25  -96.073184  27.573184 0.6420441
1:125-3:70   -88.25 -150.073184 -26.426816 0.0014679
2:125-3:70   -96.25 -158.073184 -34.426816 0.0004744
3:125-3:70   -60.25 -122.073184   1.573184 0.0604247
2:125-1:125   -8.00  -69.823184  53.823184 0.9999508
3:125-1:125   28.00  -33.823184  89.823184 0.8347331
3:125-2:125   36.00  -25.823184  97.823184 0.5819453
```

7.10 Fisher's LSD Method

The **Fisher's Least Significant Difference (LSD)** method is another option for pairwise comparisons. To test the interaction means, we must specify both factors in the `trt` argument.

```r
library(agricolae)

## Perform LSD test on the interaction term
lsd_results <- LSD.test(battery_aov, trt = c("material", "temperature"),
                        p.adj = "none", group = FALSE)

## Print the comparison table
print(lsd_results$comparison)
```

```
              difference pvalue signif.         LCL        UCL
1:125 - 1:15      -77.25 0.0003     *** -114.950479 -39.549521
1:125 - 1:70        0.25 0.9892          -37.450479  37.950479
1:125 - 2:125       8.00 0.6667          -29.700479  45.700479
1:125 - 2:15      -98.25 0.0000     *** -135.950479 -60.549521
1:125 - 2:70      -62.25 0.0022      **  -99.950479 -24.549521
1:125 - 3:125     -28.00 0.1392          -65.700479   9.700479
1:125 - 3:15      -86.50 0.0001     *** -124.200479 -48.799521
1:125 - 3:70      -88.25 0.0001     *** -125.950479 -50.549521
1:15 - 1:70        77.50 0.0002     ***   39.799521 115.200479
1:15 - 2:125       85.25 0.0001     ***   47.549521 122.950479
1:15 - 2:15       -21.00 0.2631          -58.700479  16.700479
1:15 - 2:70        15.00 0.4214          -22.700479  52.700479
1:15 - 3:125       49.25 0.0124       *   11.549521  86.950479
```

```
1:15 - 3:15        -9.25 0.6187          -46.950479  28.450479
1:15 - 3:70       -11.00 0.5544          -48.700479  26.700479
1:70 - 2:125        7.75 0.6765          -29.950479  45.450479
1:70 - 2:15       -98.50 0.0000     *** -136.200479 -60.799521
1:70 - 2:70       -62.50 0.0021      ** -100.200479 -24.799521
1:70 - 3:125      -28.25 0.1358          -65.950479   9.450479
1:70 - 3:15       -86.75 0.0001     *** -124.450479 -49.049521
1:70 - 3:70       -88.50 0.0000     *** -126.200479 -50.799521
2:125 - 2:15     -106.25 0.0000     *** -143.950479 -68.549521
2:125 - 2:70      -70.25 0.0007     *** -107.950479 -32.549521
2:125 - 3:125     -36.00 0.0605       .  -73.700479   1.700479
2:125 - 3:15      -94.50 0.0000     *** -132.200479 -56.799521
2:125 - 3:70      -96.25 0.0000     *** -133.950479 -58.549521
2:15 - 2:70        36.00 0.0605       .   -1.700479  73.700479
2:15 - 3:125       70.25 0.0007     ***   32.549521 107.950479
2:15 - 3:15        11.75 0.5279          -25.950479  49.450479
2:15 - 3:70        10.00 0.5907          -27.700479  47.700479
2:70 - 3:125       34.25 0.0732       .   -3.450479  71.950479
2:70 - 3:15       -24.25 0.1980          -61.950479  13.450479
2:70 - 3:70       -26.00 0.1685          -63.700479  11.700479
3:125 - 3:15      -58.50 0.0036      **  -96.200479 -20.799521
3:125 - 3:70      -60.25 0.0029      **  -97.950479 -22.549521
3:15 - 3:70        -1.75 0.9248          -39.450479  35.950479
```

The results from both Tukey's HSD and Fisher's LSD provide detailed p-values for comparing pairs of treatment combinations, allowing us to make specific conclusions, such as "at 125°C, Material 3 has a significantly longer life than Materials 1 and 2."