example

2024-05-26

Histogram

hist 函数的基本语法是：hist(v, main, xlab, xlim, ylim, breaks, col, border)

参数的含义如下：

v：这是一个向量，包含了我们要为其创建直方图的数值。

main：这是图表的标题。xlab：这是 x 轴的标签。

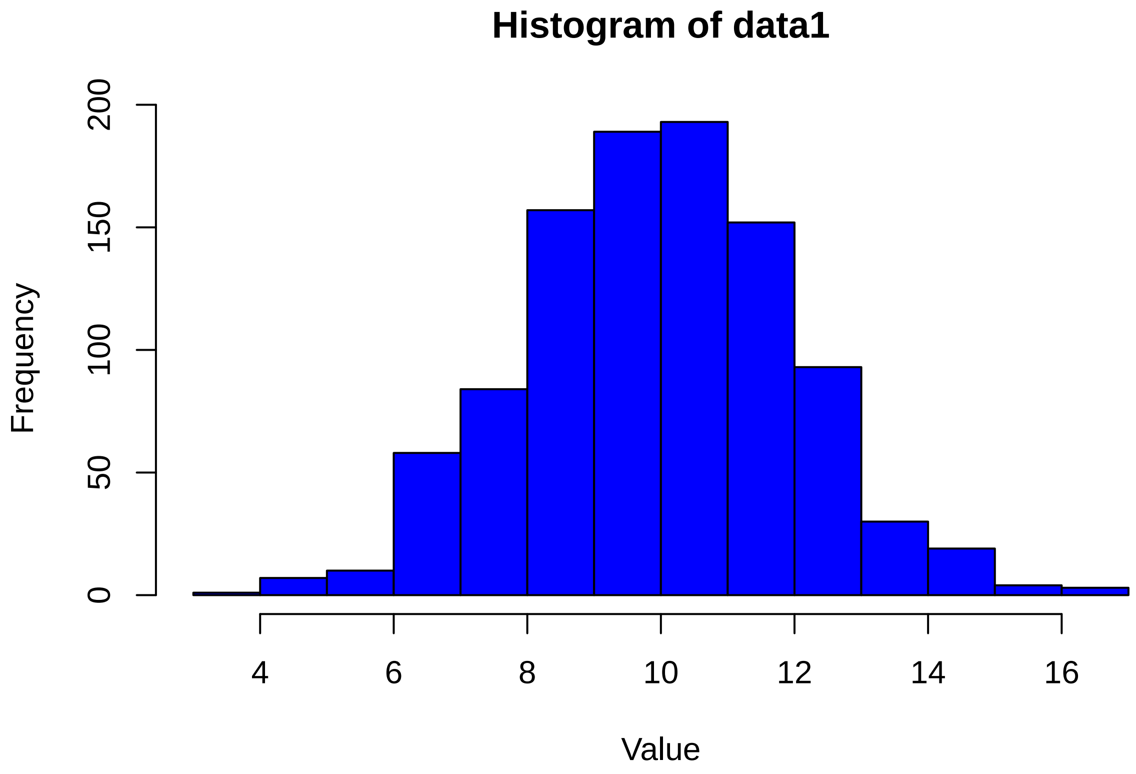
xlim：这是 x 轴的限制，是一个包含两个数字的向量，分别表示 x 轴的最小值和最大值。

ylim：这是 y 轴的限制，是一个包含两个数字的向量，分别表示 y 轴的最小值和最大值。

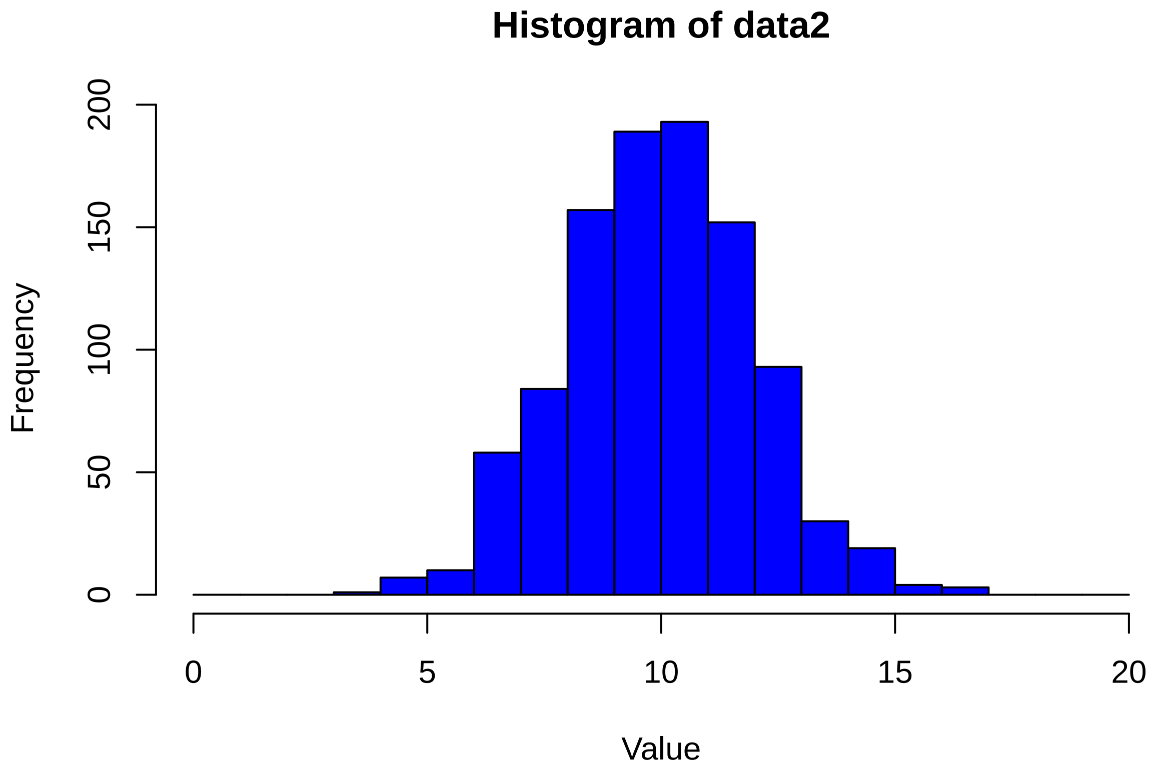
breaks：这是直方图的分箱规则。可以是一个数字（表示分箱的数量），也可以是一个向量（表示分箱的边界）。

col：这是直方图柱子的颜色。border：这是直方图柱子边界的颜色。

```r
data = rnorm(1000, 10, 2)
hist(data, breaks = 10, col = "blue", main = "Histogram of data1", xlab = "Value",
     ylab = "Frequency")
```



```
hist(data, breaks = c(0:20), col = "blue", main = "Histogram of data2", xlab = "Value",
     ylab = "Frequency")
```



tidyverse

tidyr

gather() 用于将宽格式的数据框转换为长格式的。

```r
library(tidyr)
data_wide <- data.frame(
  country = c("USA", "Canada", "Mexico"),
  year_2010 = c(300, 100, 200),
  year_2011 = c(320, 110, 210)
)
print(data_wide)
```

```
##   country year_2010 year_2011
## 1     USA       300       320
## 2  Canada       100       110
## 3  Mexico       200       210
```

```
data_long <- gather(data_wide, key = "year", value = "value", year_2010:year_2011)
print(data_long)
```

```
##   country      year value
## 1     USA year_2010   300
## 2  Canada year_2010   100
## 3  Mexico year_2010   200
## 4     USA year_2011   320
## 5  Canada year_2011   110
## 6  Mexico year_2011   210
```

spread() 将长格式数据转换为宽格式数据。

```
data_long <- data.frame(
  country = rep(c("USA", "Canada", "Mexico"), each = 2),
  year = rep(c("2010", "2011"), times = 3),
  value = c(300, 320, 100, 110, 200, 210)
)
print(data_long)
```

```
##   country year value
## 1     USA 2010   300
## 2     USA 2011   320
## 3  Canada 2010   100
## 4  Canada 2011   110
## 5  Mexico 2010   200
## 6  Mexico 2011   210
```

```
data_wide <- spread(data_long, key = "year", value = "value")
print(data_wide)
```

```
##   country 2010 2011
## 1  Canada  100  110
## 2  Mexico  200  210
## 3     USA  300  320
```

drop_na() 移除包含 NA 值的行。

```
data <- data.frame(
  id = 1:4,
  value = c(10, NA, 30, NA)
)
print(data)
```

```
##   id value
## 1  1    10
## 2  2    NA
## 3  3    30
## 4  4    NA
```

```
data_clean <- drop_na(data)
print(data_clean)
```

```
##   id value
## 1  1    10
## 2  3    30
```

fill() 用上一个非 NA 值填充 NA 值。

```
data <- data.frame(
  id = 1:5,
  value = c(10, NA, NA, 20, NA)
)
print(data)
```

```
##   id value
## 1  1    10
## 2  2    NA
## 3  3    NA
## 4  4    20
## 5  5    NA
```

```
data_filled <- fill(data, value)
print(data_filled)
```

```
##   id value
## 1  1    10
## 2  2    10
## 3  3    10
## 4  4    20
## 5  5    20
```

dplyr

filter() 用于根据特定列的条件筛选数据框中的行

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
data <- data.frame(
  id = 1:5,
  value = c(10, 20, 30, 40, 50),
  name = c("a","b","c","d","e"),
  sort = c(1, 1, 3, 2, 4)
)
print(data)
```

```
##   id value name sort
## 1  1    10    a    1
## 2  2    20    b    1
## 3  3    30    c    3
## 4  4    40    d    2
## 5  5    50    e    4
```

```r
filtered_data <- filter(data, value > 20 & name=="e")
print(filtered_data)
```

```
##   id value name sort
## 1  5    50    e    4
```

select() 用于选择数据框中的特定列。

```
selected_data <- select(data, id)
print(selected_data)
```

```
##   id
## 1  1
## 2  2
## 3  3
## 4  4
## 5  5
```

arrange() 用于根据一个或多个列对数据框进行排序。如果你指定了多个列名，那么 `arrange` 函数会按照你指定的顺序对这些列进行排序。默认升序，desc() 降序

```
arranged_data <- arrange(data, desc(sort))
print(arranged_data)
```

```
##   id value name sort
## 1  5    50    e    4
## 2  3    30    c    3
## 3  4    40    d    2
## 4  1    10    a    1
## 5  2    20    b    1
```

```
arranged_data <- arrange(data, desc(sort), desc(id))
print(arranged_data)
```

```
##   id value name sort
## 1  5    50    e    4
## 2  3    30    c    3
## 3  4    40    d    2
## 4  2    20    b    1
## 5  1    10    a    1
```

mutate() 用于在数据框中添加新列或修改现有列

```r
mutated_data <- mutate(data, value2 = value * 2)
print(mutated_data)
```

```
##   id value name sort value2
## 1  1    10    a    1     20
## 2  2    20    b    1     40
## 3  3    30    c    3     60
## 4  4    40    d    2     80
## 5  5    50    e    4    100
```

summary() 用于对数据框中的数据进行汇总

```r
summarised_data <- summary(data)
print(summarised_data)
```

```
##        id         value          name                 sort    
##   Min.   :1   Min.   :10   Length:5           Min.   :1.0  
##   1st Qu.:2   1st Qu.:20   Class :character   1st Qu.:1.0  
##   Median :3   Median :30   Mode  :character   Median :2.0  
##   Mean   :3   Mean   :30                      Mean   :2.2  
##   3rd Qu.:4   3rd Qu.:40                      3rd Qu.:3.0  
##   Max.   :5   Max.   :50                      Max.   :4.0  
```

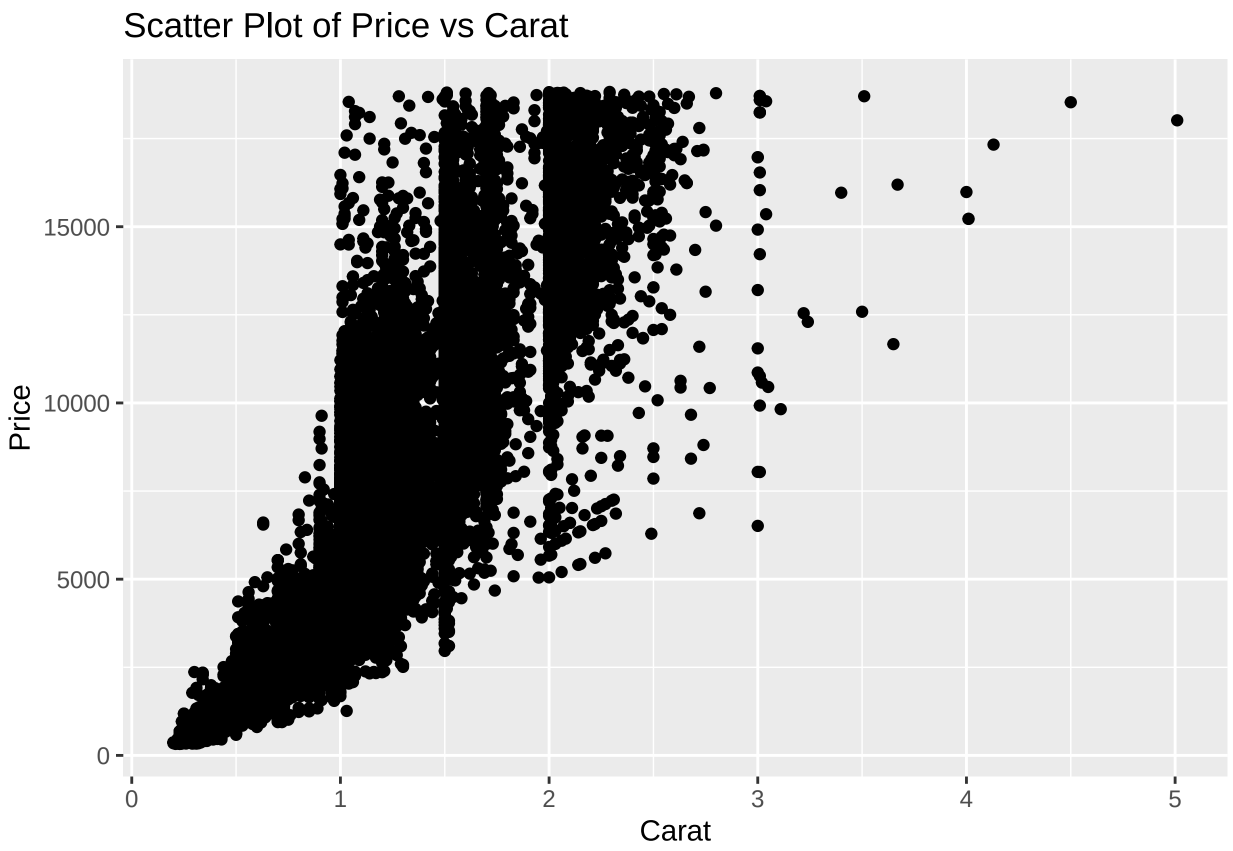
ggplot2

Scatter Plot with geom_point()

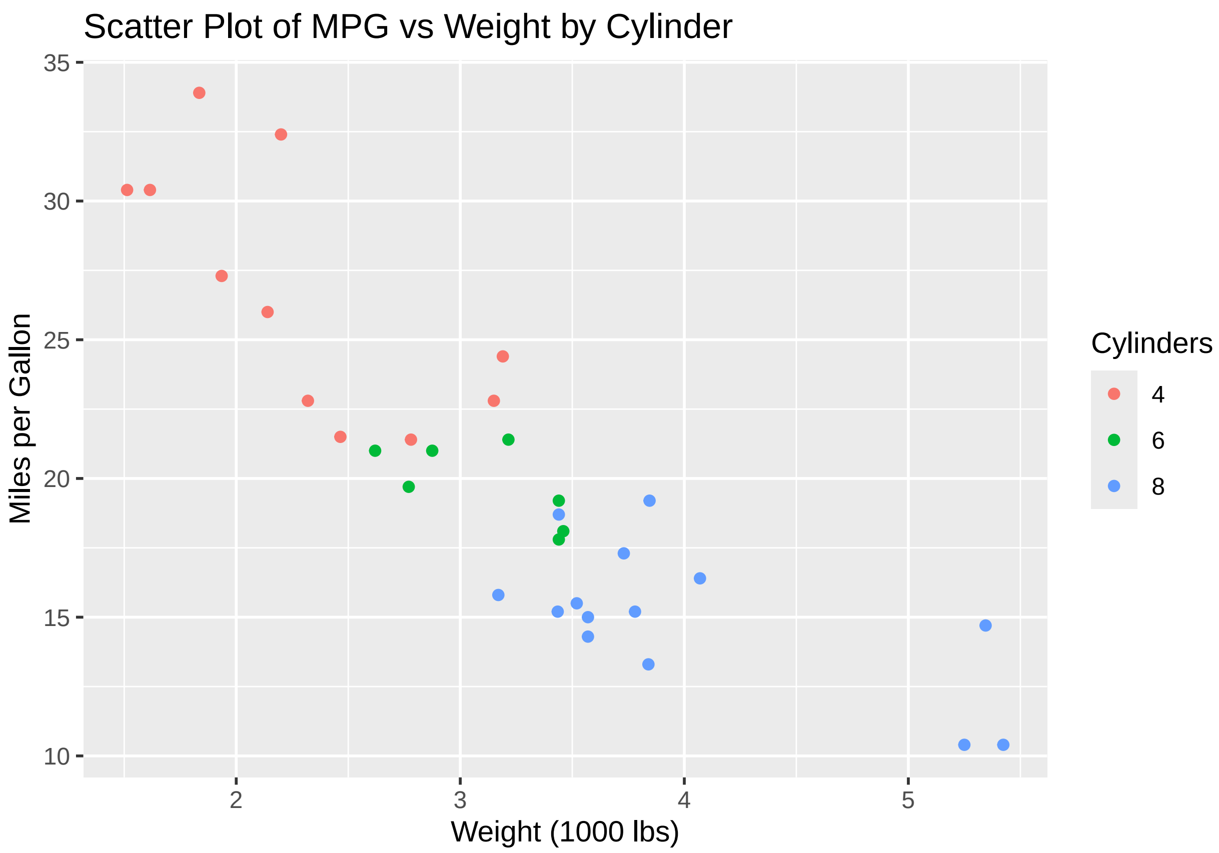
```r
library(ggplot2)
head(diamonds)
```

```
## # A tibble: 6 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
## 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
## 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
## 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
## 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
## 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

```
# Scatter plot of 'price' vs 'carat' in the 'diamonds' dataset
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point() +
  labs(title = "Scatter Plot of Price vs Carat",
       x = "Carat",
       y = "Price")
```

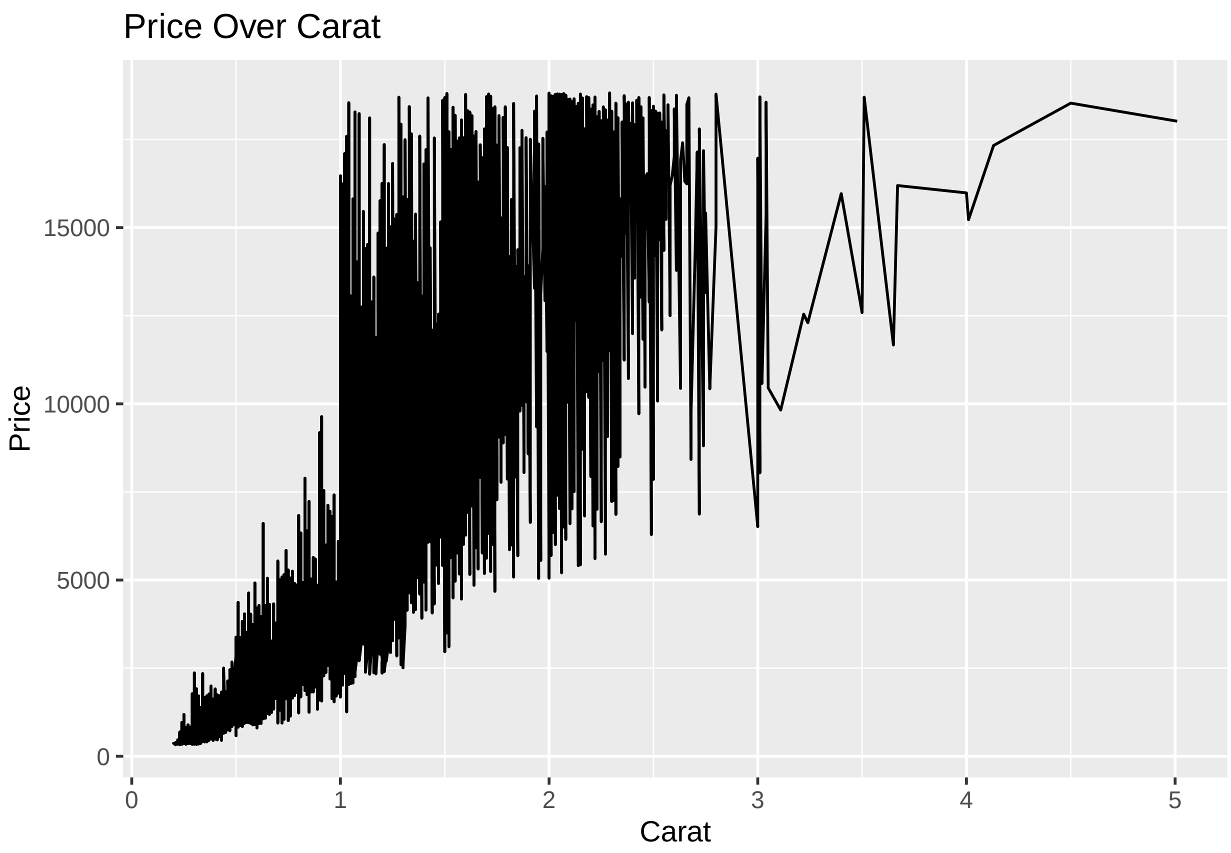


```
# Scatter plot with color aesthetic
ggplot(mtcars, aes(x = wt, y = mpg, color = as.factor(cyl))) +
  geom_point() +
  labs(title = "Scatter Plot of MPG vs Weight by Cylinder",
       x = "Weight (1000 lbs)",
       y = "Miles per Gallon",
       color = "Cylinders")
```

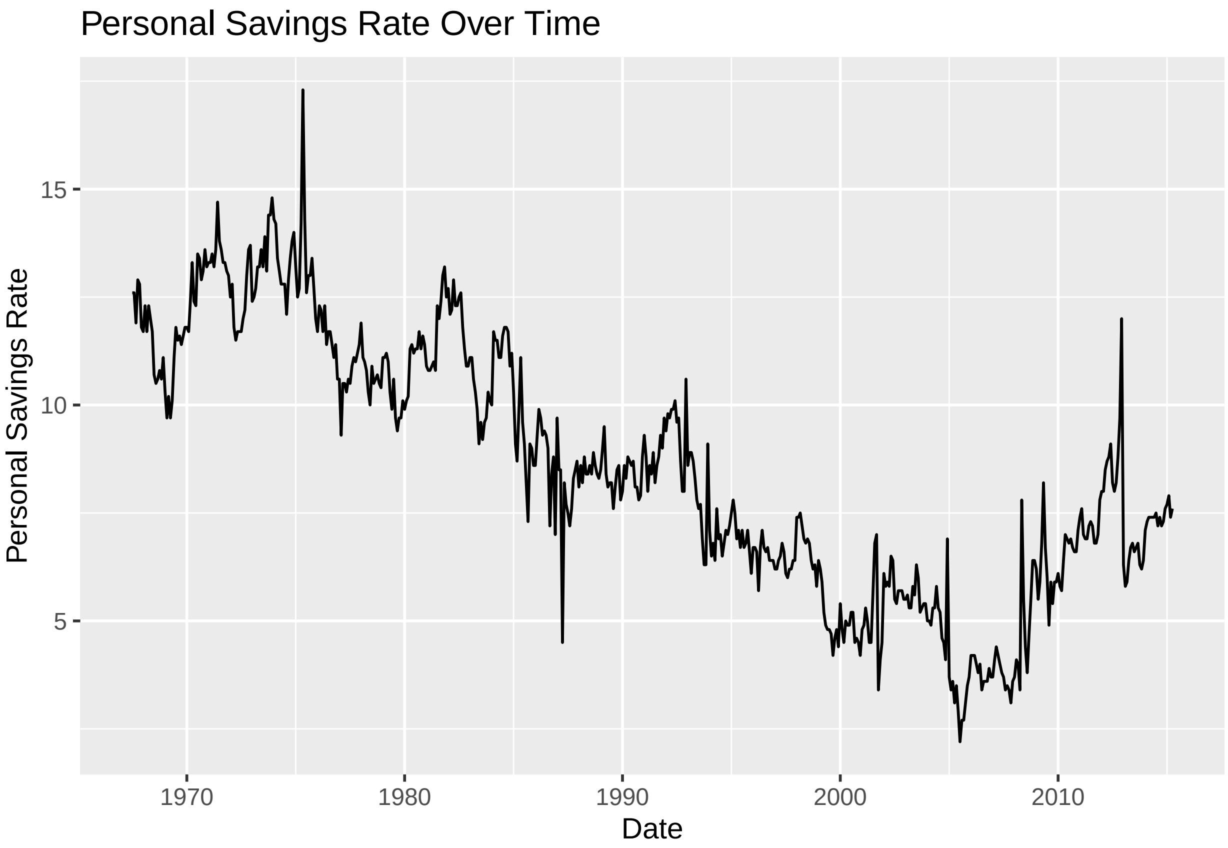


Line Plot with geom_line()

```r
# Line plot of 'price' over 'carat' in the 'diamonds' dataset
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_line() +
  labs(title = "Price Over Carat",
       x = "Carat",
       y = "Price")
```

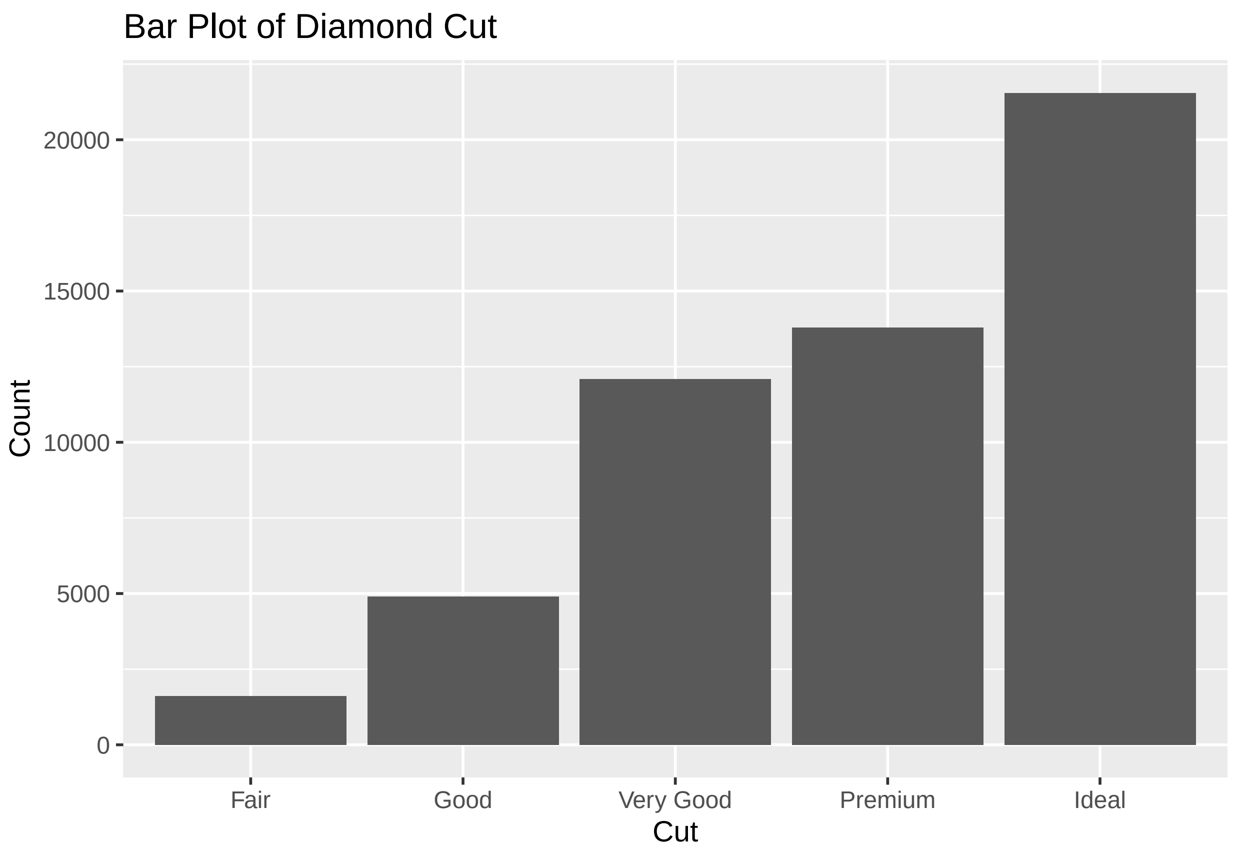


```r
# Line plot of 'psavert' over time in the 'economics' dataset
ggplot(economics, aes(x = date, y = psavert)) +
  geom_line() +
  labs(title = "Personal Savings Rate Over Time",
       x = "Date",
       y = "Personal Savings Rate")
```

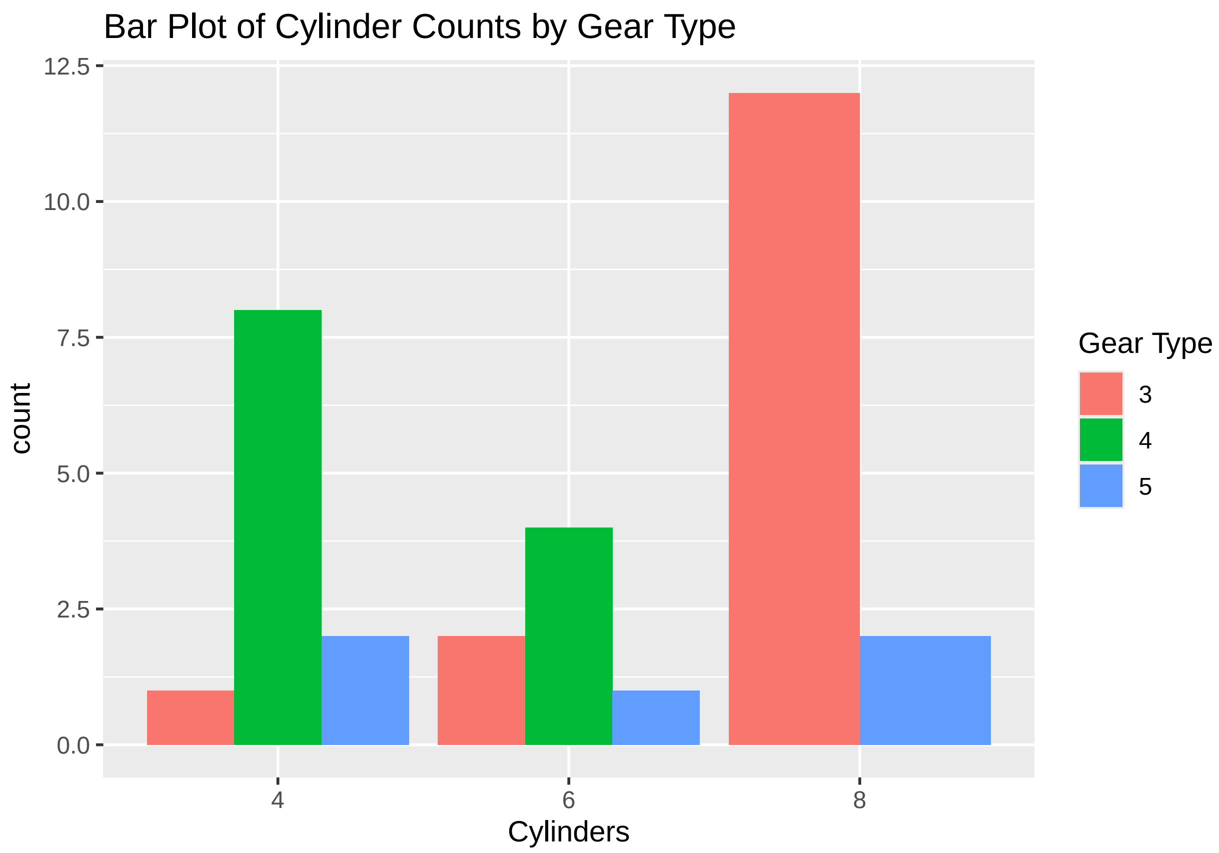


Bar Plot with geom_bar()

```
# Bar plot of the 'cut' distribution in the 'diamonds' dataset
ggplot(diamonds, aes(x = cut)) +
  geom_bar() +
  labs(title = "Bar Plot of Diamond Cut",
       x = "Cut",
       y = "Count")
```

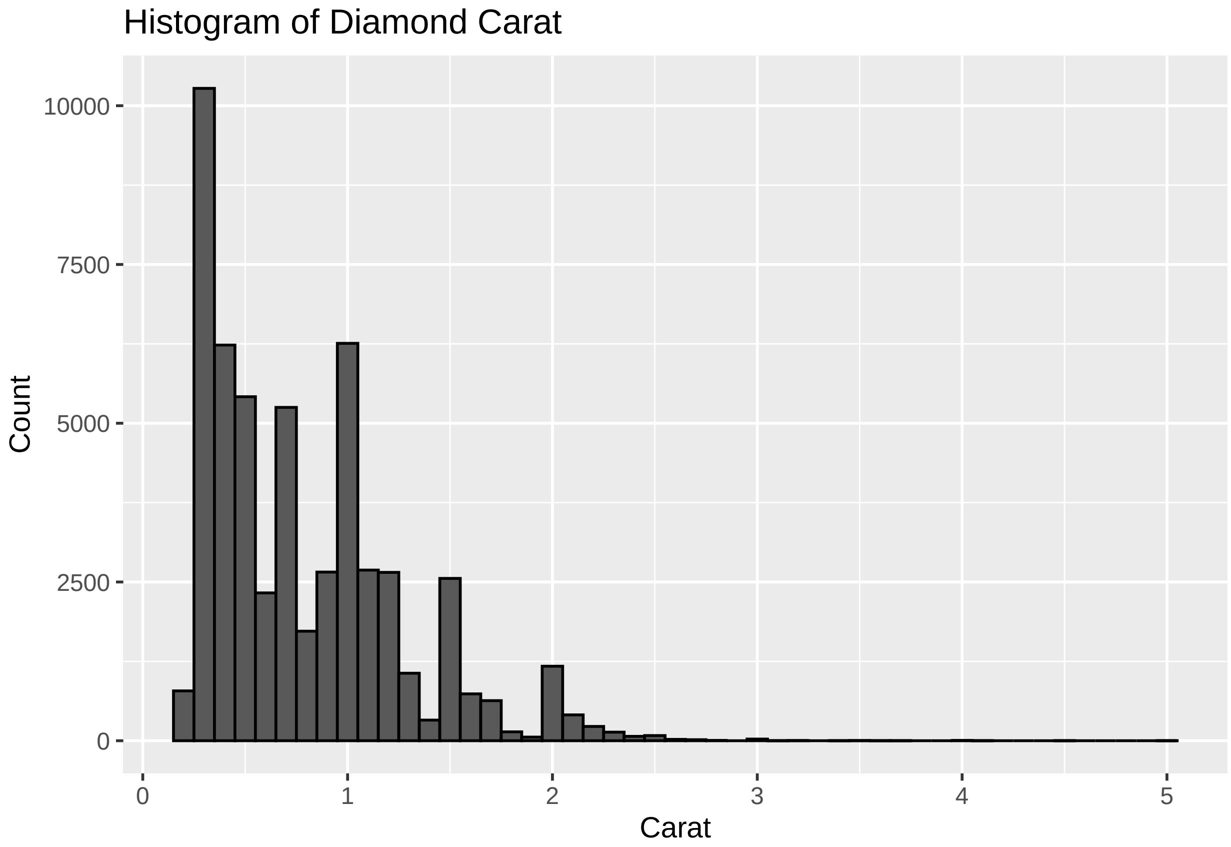


```r
# Bar plot with fill aesthetic
ggplot(mtcars, aes(x = as.factor(cyl), fill = as.factor(gear))) +
  geom_bar(position = "dodge") +
  labs(title = "Bar Plot of Cylinder Counts by Gear Type",
       x = "Cylinders",
       fill = "Gear Type")
```



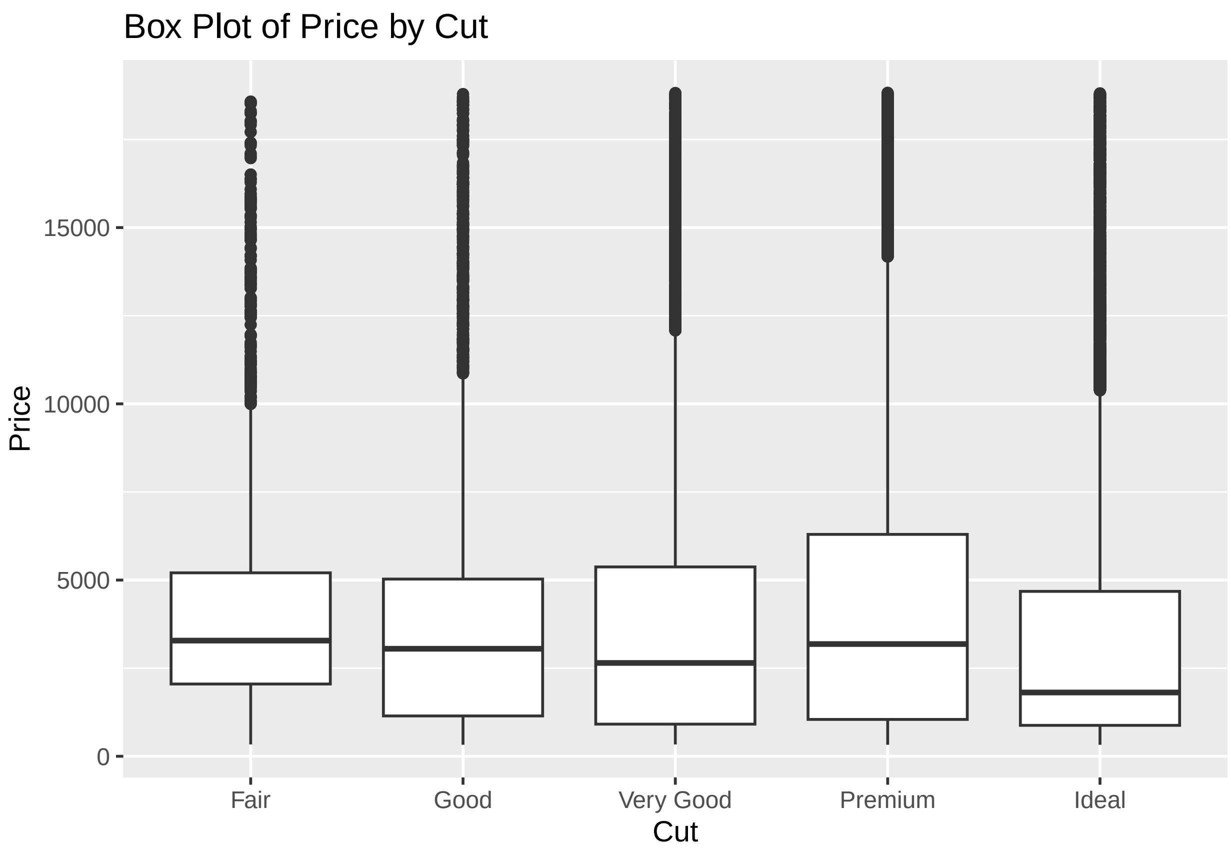
Histogram with geom_histogram()

```
# Histogram of 'carat' in the 'diamonds' dataset
ggplot(diamonds, aes(x = carat)) +
  geom_histogram(binwidth = 0.1, color = "black") +
  labs(title = "Histogram of Diamond Carat",
       x = "Carat",
       y = "Count")
```



Box Plot with geom_boxplot()

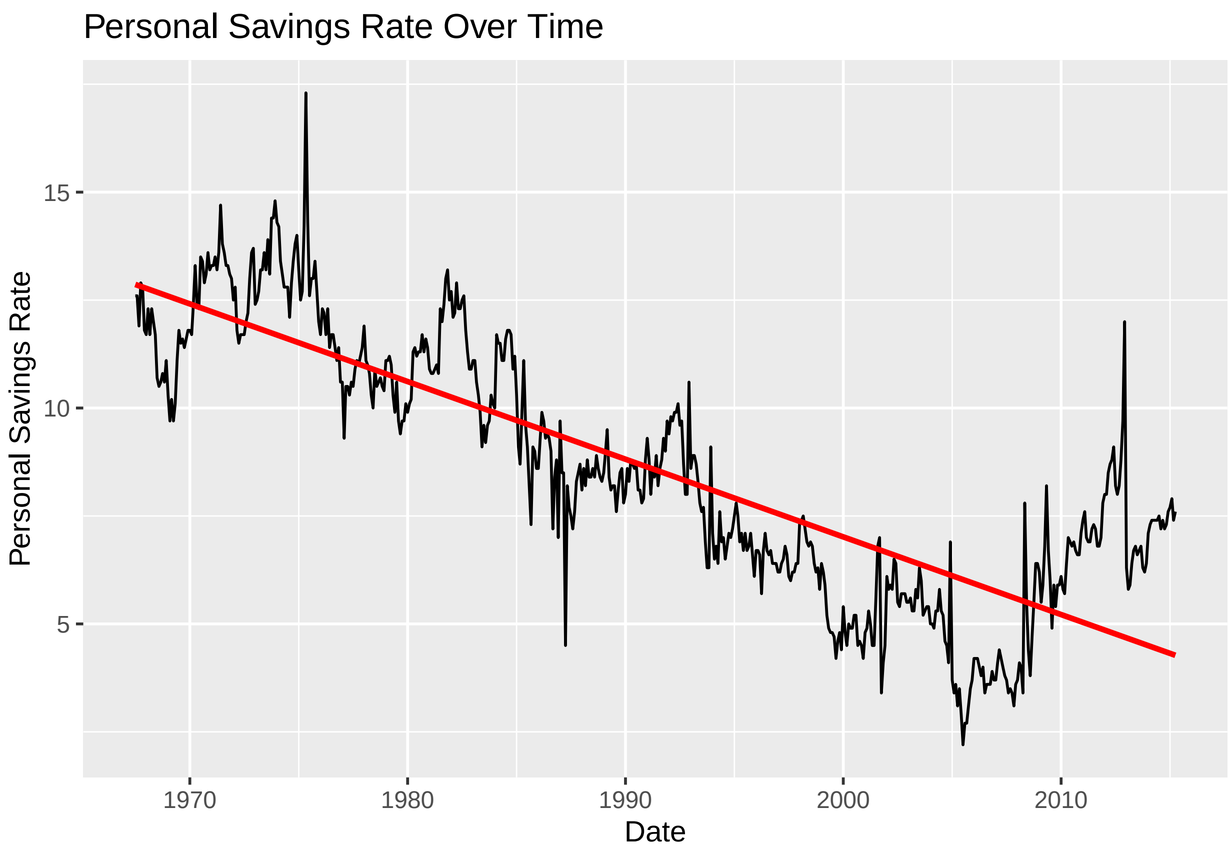
```r
# Box plot of 'price' by 'cut' in the 'diamonds' dataset
ggplot(diamonds, aes(x = cut, y = price))+
  geom_boxplot() +
  labs(title = "Box Plot of Price by Cut",
       x = "Cut",
       y = "Price")
```



Smoothing Line with geom_smooth()

```r
# Line plot of 'psavert' over time in the 'economics' dataset
ggplot(economics, aes(x = date, y = psavert)) +
  geom_line() +
  geom_smooth(method = "lm", color = "red", se = FALSE) +
  labs(title = "Personal Savings Rate Over Time",
       x = "Date",
       y = "Personal Savings Rate")
```

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



Test

单样本 t 检验（**One-Sample t-Test**）

```
# 生成样本数据
set.seed(123)
sample_data <- rnorm(100, mean = 50, sd = 10)

# 验证假设：样本正态性（Shapiro-Wilk 检验）
shapiro.test(sample_data)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  sample_data
## W = 0.99388, p-value = 0.9349
```

```r
# 单样本 t 检验
t.test(sample_data, mu = 50)
```

```
##
##  One Sample t-test
##
## data:  sample_data
## t = 0.99041, df = 99, p-value = 0.3244
## alternative hypothesis: true mean is not equal to 50
## 95 percent confidence interval:
##  49.09283 52.71528
## sample estimates:
## mean of x
##  50.90406
```

独立样本 t 检验（Two-Sample t-Test）

```r
# 生成样本数据
set.seed(123)
sample_data1 <- rnorm(50, mean = 50, sd = 10)
sample_data2 <- rnorm(50, mean = 52, sd = 10)

# 验证假设：样本正态性（Shapiro-Wilk 检验）
shapiro.test(sample_data1)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  sample_data1
## W = 0.98928, p-value = 0.9279
```

```r
shapiro.test(sample_data2)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  sample_data2
## W = 0.99073, p-value = 0.9618
```

```r
# 验证假设：样本方差相等（F 检验）
var.test(sample_data1, sample_data2)
```

```
##
##  F test to compare two variances
##
## data:  sample_data1 and sample_data2
## F = 1.0456, num df = 49, denom df = 49, p-value = 0.8766
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.5933642 1.8425790
## sample estimates:
## ratio of variances
##            1.04562
```

```r
# 独立样本 t 检验
t.test(sample_data1, sample_data2, var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  sample_data1 and sample_data2
## t = -1.7036, df = 98, p-value = 0.09162
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -6.7544625  0.5143682
## sample estimates:
## mean of x mean of y
##  50.34404  53.46408
```

配对样本 t 检验（Paired-Sample t-Test）

```r
# 生成样本数据
set.seed(123)
before <- rnorm(30, mean = 50, sd = 10)
after <- before + rnorm(30, mean = 2, sd = 5)
```

```
# 计算配对差值
differences <- after - before

# 验证假设：差值正态性（Shapiro-Wilk 检验）
shapiro.test(differences)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  differences
## W = 0.98662, p-value = 0.9614
```

```
# 配对样本 t 检验
t.test(before, after, paired = TRUE)
```

```
##
##  Paired t-test
##
## data:  before and after
## t = -3.7931, df = 29, p-value = 0.0006996
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -4.450901 -1.332482
## sample estimates:
## mean difference
##       -2.891692
```

非参数检验（Non-Parametric Tests）

Wilcoxon 符号秩检验（Wilcoxon Signed-Rank Test）

Mann-Whitney U 检验（Mann-Whitney U Test）是两种常用的非参数检验，适用于不满足正态性假设的数据。

```
# Wilcoxon 符号秩检验（配对数据或单样本）
wilcox.test(before, after, paired = TRUE)
```

```
##
##  Wilcoxon signed rank exact test
```

```
##
## data:  before and after
## V = 68, p-value = 0.0003801
## alternative hypothesis: true location shift is not equal to 0
```

```r
wilcox.test(sample_data, mu = 50)
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  sample_data
## V = 2763, p-value = 0.4142
## alternative hypothesis: true location is not equal to 50
```

```r
# Mann-Whitney U 检验（非配对数据）
wilcox.test(sample_data1, sample_data2, paired = FALSE)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  sample_data1 and sample_data2
## W = 995, p-value = 0.07935
## alternative hypothesis: true location shift is not equal to 0
```