第六课（第16-18课时）

回归分析和基于模拟的分析

- 线性回归分析
- 点估计和区间估计
- 置换检验（Permutation Test）
- 自助法（Bootstrap）

前情回顾

- **如何检验一个变量的一组取值是否符合某种分布**
  - 图形分析
  - 使用样本数字特征
  - 使用假设检验？
- **如何检验两个变量之间的关系**
  - 图形分析
  - 根据变量类型选择合适的分析方法
  - 如不相互独立，则进一步分析

基础统计方法

		自变量	
		连续型	类别型
因变量	连续型	相关分析 回归分析	t-检定 方差分析
	类别型	广义线性模型	卡方检定

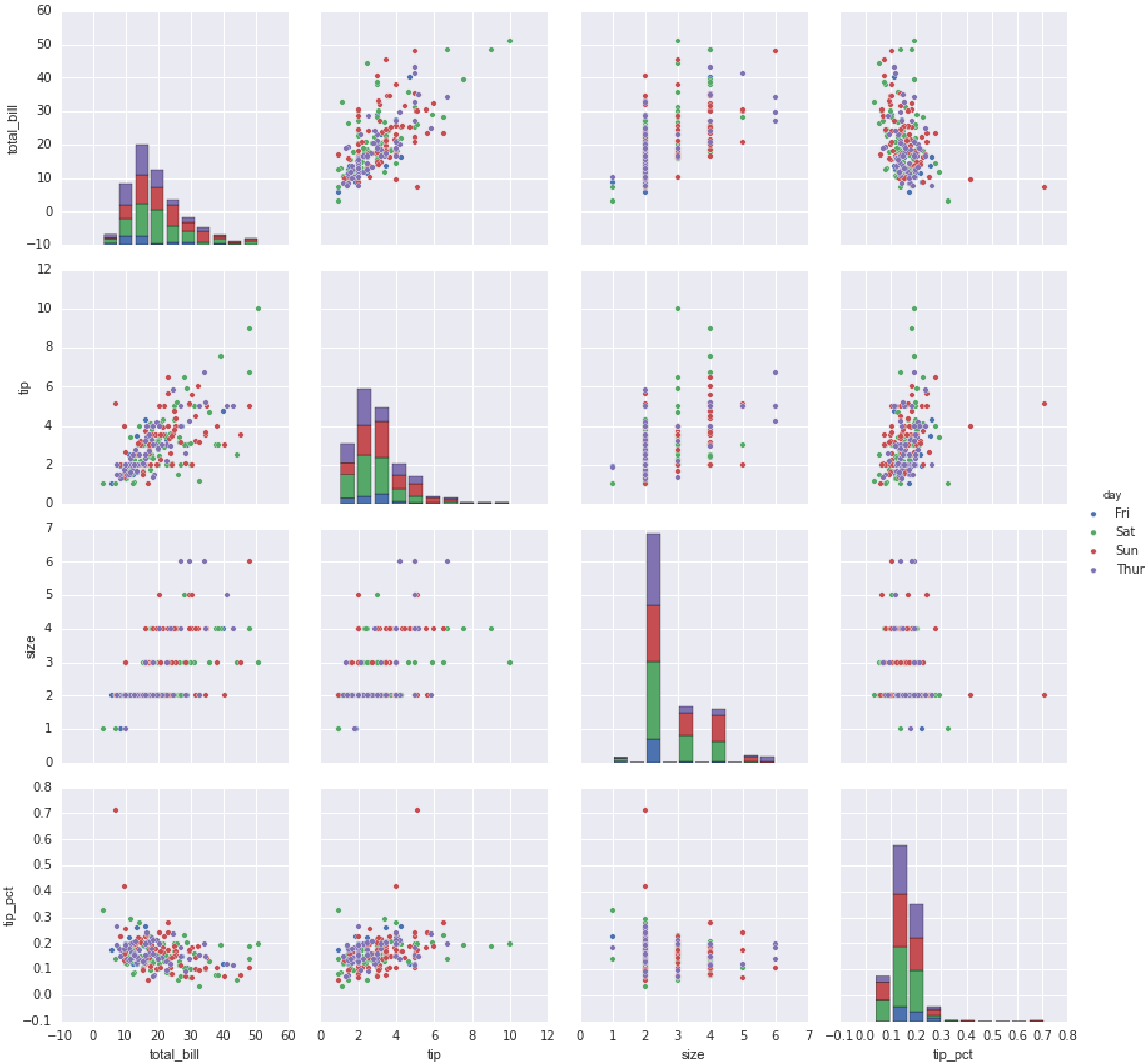
载入数据：tips.csv

小象学院 ChinaHadoop.cn

- **载入常用库**
  - import pandas as pd
  - import numpy as np
  - import matplotlib.pyplot as plt
- **载入模块**
  - from pandas import Series, DataFrame
  - from scipy import stats
- **数据：链接: http://pan.baidu.com/s/1bpKAd8V 密码: dw8g**
  - **tips.csv**
- **读入数据（假设文件在工作目录路径下）**
  - tips=pd.read_csv('tips.csv')
- **加工数据：**
  - tips['tip_pct']=tips['tip']/tips['total_bill']

可视化分析变量之间的关系

- import seaborn as sns
- sns.set()
- sns.pairplot(tips, hue="day")



数据分析和数据挖掘

任务描述

- 通过回归分析，确定变量之间的关系，即“模型”
- 理解线性回归的原理，输出的含义
- 掌握如何评价和选择回归模型
- 掌握基于重抽样（模拟）的分析方法：置换检验，和自助法

回归分析

- **回归分析：**
  - 用一个或多个预测变量（自变量、解释变量）来预测响应变量（因变量、效标变量、结果变量）的方法
  - **挑选**与响应变量相关的解释变量
  - **描述**两者关系
  - 通过解释变量**预测**响应变量
  - 接近现实世界
  - 交互式的，拟合一系列模型，选择“最佳”模型
  - 难题：
    - 问题的提出，有用和可测的响应变量，合适的数据

回归分析

小象学院 ChinaHadoop.cn

回归分析的类型

回归类型	用　途
简单线性	用一个量化的解释变量预测一个量化的响应变量
多项式	用一个量化的解释变量预测一个量化的响应变量，模型的关系是n阶多项式
多元线性	用两个或多个量化的解释变量预测一个量化的响应变量
多变量	用一个或多个解释变量预测多个响应变量
Logistic	用一个或多个解释变量预测一个类别型响应变量
泊松	用一个或多个解释变量预测一个代表频数的响应变量
Cox比例风险	用一个或多个解释变量预测一个事件（死亡、失败或旧病复发）发生的时间
时间序列	对误差项相关的时间序列数据建模
非线性	用一个或多个量化的解释变量预测一个量化的响应变量，不过模型是非线性的
非参数	用一个或多个量化的解释变量预测一个量化的响应变量，模型的形式源自数据形式，不事先设定
稳健	用一个或多个量化的解释变量预测一个量化的响应变量，能抵御强影响点的干扰

回归分析

小象学院 ChinaHadoop.cn

- **回归模型**
  - 对于机理尚不明确的变量建立模型

$$Y = f(X_1, X_2, ..., X_{p-1}) + \epsilon$$

未考虑因素

  - 利用观测数据确定f(Xi)和误差
- **线性回归**

$$Y = \beta_0 + \beta_1 X_1 + ... + \beta_{p-1} X_{p-1} + \epsilon$$

  - 一般模型也可以转化为线性回归模型

回归分析

小象学院 ChinaHadoop.cn

- **线性回归：**
  - 对Y, X1,...Xp-1进行n（>=p）次独立观测: (yi; xi1,...xi,p-1), i= 1, 2, ..., n

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_{p-1} X_{p-1} + \epsilon$$

  - 矩阵形式

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1,p-1} \\ 1 & x_{21} & \cdots & x_{2,p-1} \\ \vdots & & \vdots & \vdots \\ 1 & x_{n1} & \cdots & x_{n,p-1} \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{p-1} \end{bmatrix} \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

回归分析

- **线性回归：**

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon$$

设计矩阵
假设满秩

不可观测的随机误
假设服从n维标准正态分布

- **参数估计及其性质：**

$$\beta, \quad \sigma^2$$

回归分析

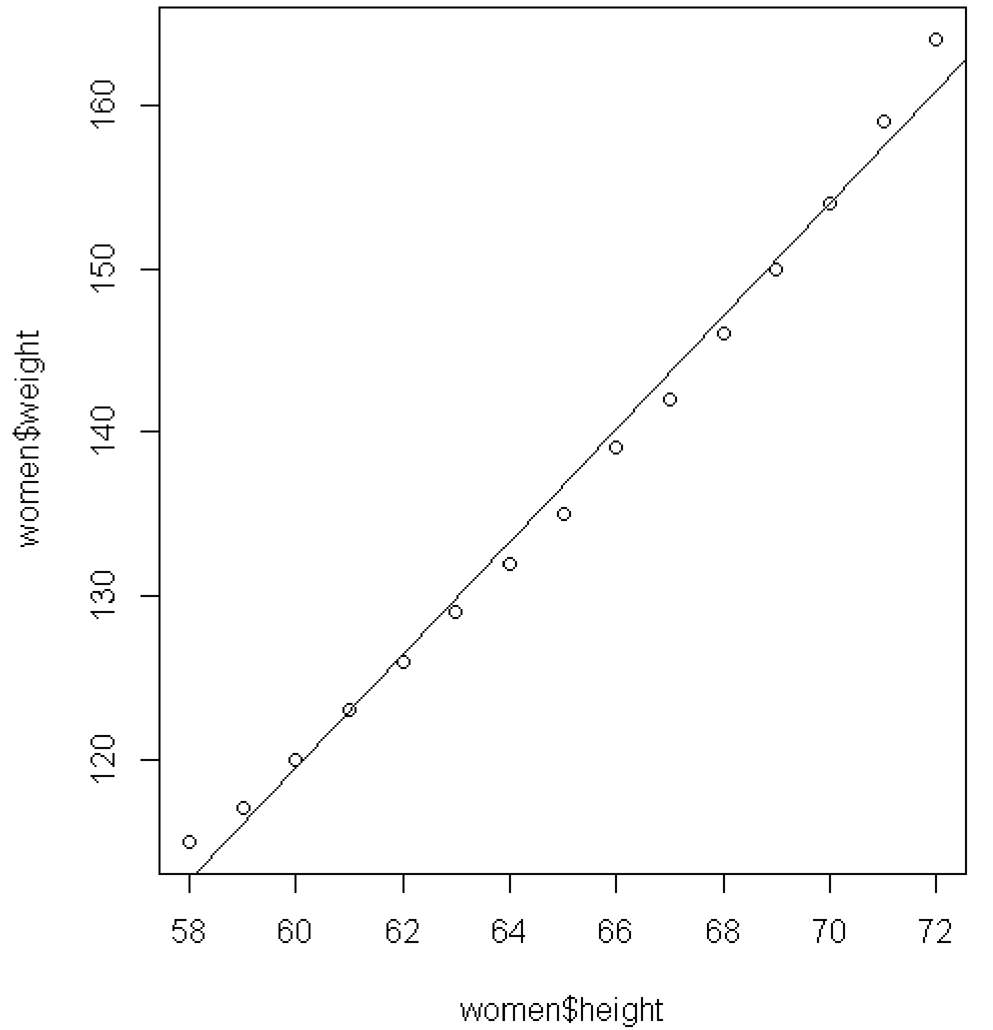
- 以R为例

```
> fit<-lm(weight~height, data=women)
> summary(fit)

Call:
lm(formula = weight ~ height, data = women)

Residuals:
    Min      1Q  Median      3Q     Max
-1.7333 -1.1333 -0.3833  0.7417  3.1167

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -87.51667    5.93694  -14.74 1.71e-09 ***
height        3.45000    0.09114   37.85 1.09e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.525 on 13 degrees of freedom
Multiple R-squared:  0.991,     Adjusted R-squared:  0.9903
F-statistic:  1433 on 1 and 13 DF,  p-value: 1.091e-14
```

回归分析

- 以R为例

```
> fitted(fit)
       1        2        3        4
112.5833 116.0333 119.4833 122.9333
      13       14       15
153.9833 157.4333 160.8833
> residuals(fit)
          1           2           3
 2.41666667  0.96666667  0.51666667
         10          11          12
-1.63333333 -1.08333333 -0.53333333
> plot(women$height, women$weight)
> abline(fit)
> coefficients(fit)
(Intercept)      height
  -87.51667     3.45000
> 
```



回归分析

- **stats.linregress**
  - from scipy import stats
  - np.random.seed(12345678)
  - x = np.random.random(10)
  - y = np.random.random(10)
  - slope, intercept, r_value, p_value, std_err = stats.linregress(x,y)

Examples

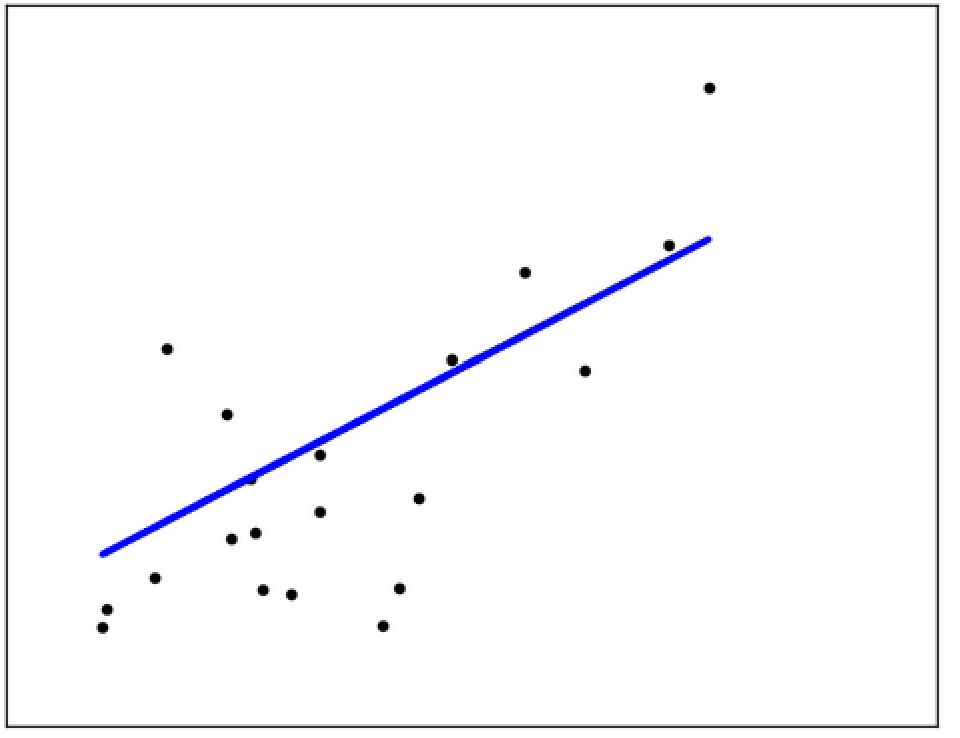
```
>>> from scipy import stats
>>> np.random.seed(12345678)
>>> x = np.random.random(10)
>>> y = np.random.random(10)
>>> slope, intercept, r_value, p_value, std_err = stats.linregress(x,y)
```

To get coefficient of determination (r_squared)

```
>>> print("r-squared:", r_value**2)
('r-squared:', 0.080402268539028335)
```

回归分析

```python
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model

# Load the diabetes dataset
diabetes = datasets.load_diabetes()


# Use only one feature
diabetes_X = diabetes.data[:, np.newaxis, 2]

# Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]

# Split the targets into training/testing sets
diabetes_y_train = diabetes.target[:-20]
diabetes_y_test = diabetes.target[-20:]

# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)

# The coefficients
print('Coefficients: \n', regr.coef_)
# The mean square error
print("Residual sum of squares: %.2f"
      % np.mean((regr.predict(diabetes_X_test) - diabetes_y_test) ** 2))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % regr.score(diabetes_X_test, diabetes_y_test))

# Plot outputs
plt.scatter(diabetes_X_test, diabetes_y_test,  color='black')
plt.plot(diabetes_X_test, regr.predict(diabetes_X_test), color='blue',
         linewidth=3)

plt.xticks(())
plt.yticks(())
```

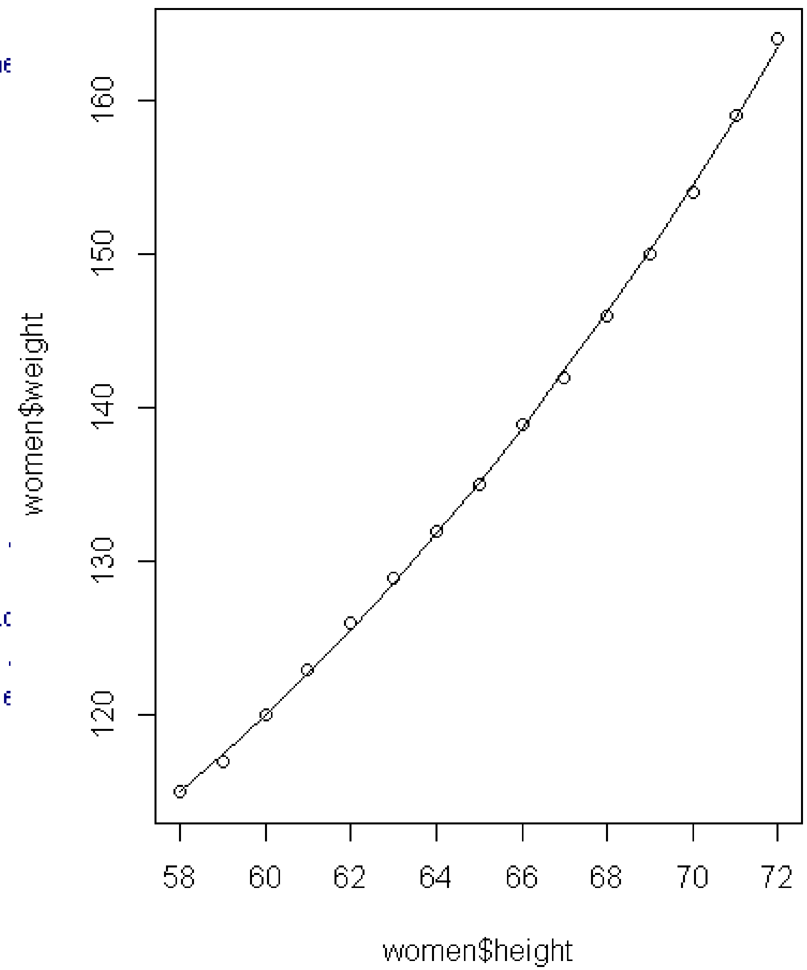


回归分析

小象学院 ChinaHadoop.cn

- 多项式回归，以R为例

```
> fit2<-lm(weight~height+I(height^2), data=women)
> summary(fit2)

Call:
lm(formula = weight ~ height + I(height^2), data = wome

Residuals:
     Min       1Q   Median       3Q      Max
-0.50941 -0.29611 -0.00941  0.28615  0.59706

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 261.87818   25.19677  10.393 2.36e-07 ***
height       -7.34832    0.77769  -9.449 6.58e-07 ***
I(height^2)   0.08306    0.00598  13.891 9.32e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.

Residual standard error: 0.3841 on 12 degrees of freedo
Multiple R-squared:  0.9995,    Adjusted R-squared:  0.
F-statistic: 1.139e+04 on 2 and 12 DF,  p-value: < 2.2e

> plot(women$height, women$weight)
> lines(women$height,fitted(fit2))
>
```



回归分析

➢ 多元线性回归，以R为例

```
> fit<-lm(Murder~Population+Illiteracy+Income+Frost,data=states)
> summary(fit)

Call:
lm(formula = Murder ~ Population + Illiteracy + Income + Frost,
    data = states)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.7960 -1.6495 -0.0811  1.4815  7.6210 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 1.235e+00  3.866e+00   0.319   0.7510    
Population  2.237e-04  9.052e-05   2.471   0.0173 *  
Illiteracy  4.143e+00  8.744e-01   4.738 2.19e-05 ***
Income      6.442e-05  6.837e-04   0.094   0.9253    
Frost       5.813e-04  1.005e-02   0.058   0.9541    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.535 on 45 degrees of freedom
Multiple R-squared:  0.567,     Adjusted R-squared:  0.5285 
F-statistic: 14.73 on 4 and 45 DF,  p-value: 9.133e-08
```

回归分析

➢ 多元线性回归，以R为例

```
> fit<-lm(mpg~hp+wt+hp:wt,data=mtcars)
> summary(fit)

Call:
lm(formula = mpg ~ hp + wt + hp:wt, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.0632 -1.6491 -0.7362  1.4211  4.5513 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 49.80842    3.60516  13.816 5.01e-14 ***
hp          -0.12010    0.02470  -4.863 4.04e-05 ***
wt          -8.21662    1.26971  -6.471 5.20e-07 ***
hp:wt        0.02785    0.00742   3.753 0.000811 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.153 on 28 degrees of freedom
Multiple R-squared:  0.8848,	Adjusted R-squared:  0.8724 
F-statistic: 71.66 on 3 and 28 DF,  p-value: 2.981e-13
```

回归分析

- **线性回归：最小二乘法**
  - 回归参数的最小二乘估计：误差项平方和最小

$$S(\beta)=\sum_{i=1}^{n}\epsilon_i^2$$

$$\hat{\beta}=(\hat{\beta}_0,\hat{\beta}_1,\ldots,\hat{\beta}_{p-1})^T=(X^TX)^{-1}X^TY$$

- **经验回归方程（回归方程）**

$$\hat{Y}=\hat{\beta}_0+\hat{\beta}_1X_1+\ldots+\hat{\beta}_{p-1}X_{p-1}$$

  - 可用来预测Y

回归分析

- **线性回归：误差方差的估计**
  - 残差向量：

$$\hat{\epsilon} = Y - \hat{Y}$$

  - 残差平方和

$$SSE = \sum_{i=1}^{n} \epsilon_i^2$$

  - 无偏估计

$$\hat{\sigma}^2 = \frac{SSE}{n-p}$$

回归分析

- **线性回归：对参数的估计**
- **估计量的基本性质1**

$$E(\hat{\beta}) = \beta$$

$$Cov(\hat{\beta}) = \sigma^2 (X^T X)^{-1}$$

$$E(\hat{\sigma}^2) = \sigma^2$$

回归分析

- 线性回归：对参数的估计
- 估计量的基本性质2

$$If \quad \epsilon \sim N(0, \sigma^2 I)$$

$$\hat{\beta} \sim N(\beta, \sigma^2 (X^T X)^{-1})$$

$$\frac{1}{\sigma^2} SSE = \frac{n-p}{\sigma^2} \hat{\sigma}^2 \sim \chi^2_{n-p}$$

$\hat{\beta} \ and \ SSE(or \sigma^2)$ 相互独立

回归分析

- 线性回归：对参数的估计
- 估计量的基本性质3

$$If \quad \epsilon \sim N(0, \sigma^2 I)$$

$$E(\hat{\epsilon}) = 0, \quad Cov(\hat{\epsilon}) = \sigma^2 (I - H)$$

$$\hat{\epsilon} \sim N(0, \sigma^2 (I - H))$$

回归分析

小象学院 ChinaHadoop.cn

- **线性回归：拟合度**

- **回归方程的显著性检验：因变量和自变量之间是否存在显著的线性关系？**
  - 1. 总离差平方和 $$SST=\sum_{i=1}^{n}(y_i-\bar{y})^2$$
  - 2. 残差(误差)平方和 $$SSE=\sum_{i=1}^{n}(y_i-\hat{y}_i)^2$$
  - 3. 回归平方和 $$SSR=\sum_{i=1}^{n}(\hat{y}_i-\bar{y})^2$$
  - 4. 复相关系数

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

越大越显著

回归分析

小象学院 ChinaHadoop.cn

- **线性回归：线性回归关系的显著性检验：**

$$H_0: \beta_1 = \beta_2 = \ldots = \beta_{p-1} = 0 \longleftrightarrow$$

$$H_1: \exists 1 \leq i \leq p-1 \ s.t. \ \beta_i \neq 0$$

$$F = \frac{SSR/(p-1)}{SSE/(n-p)} = \frac{MSR}{MSE}$$

$$F \sim F(p-1, n-p)$$

回归分析

小象学院 ChinaHadoop.cn

- **线性回归：回归系数的统计推断**
  - 若回归关系显著性H0被拒绝，仍需对每一自变量做显著性检验

$$H_{0k} : \beta_k = 0 \leftrightarrow H_{1k} : \beta_k \neq 0$$

回归分析

- 多元线性回归，以R为例

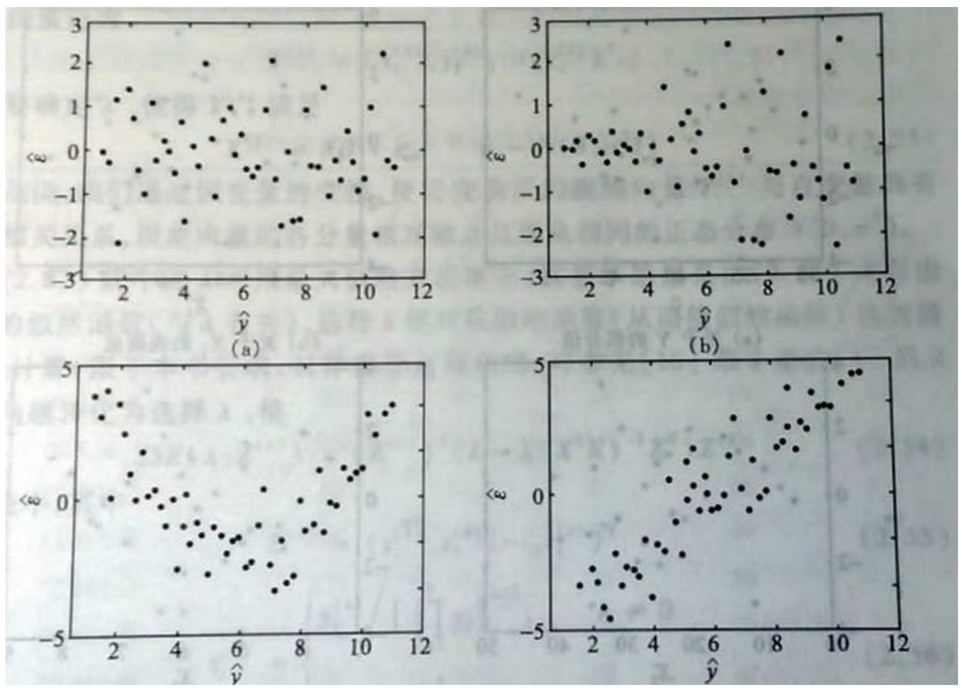
```
> fit<-lm(mpg~hp+wt+hp:wt,data=mtcars)
> summary(fit)

Call:
lm(formula = mpg ~ hp + wt + hp:wt, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-3.0632 -1.6491 -0.7362  1.4211  4.5513

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 49.80842    3.60516  13.816 5.01e-14 ***
hp          -0.12010    0.02470  -4.863 4.04e-05 ***
wt          -8.21662    1.26971  -6.471 5.20e-07 ***
hp:wt        0.02785    0.00742   3.753 0.000811 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.153 on 28 degrees of freedom
Multiple R-squared:  0.8848,    Adjusted R-squared:  0.8724
F-statistic: 71.66 on 3 and 28 DF,  p-value: 2.981e-13
```

回归分析

- **线性回归：分析结论的前提：数据满足统计假设**
  - 正态性：预测变量固定时，因变量正态分布
  - 独立性：因变量互相独立
  - 线性：因变量与自变量线性（残差白噪声）
  - 同方差性：残差方差均匀

A

B

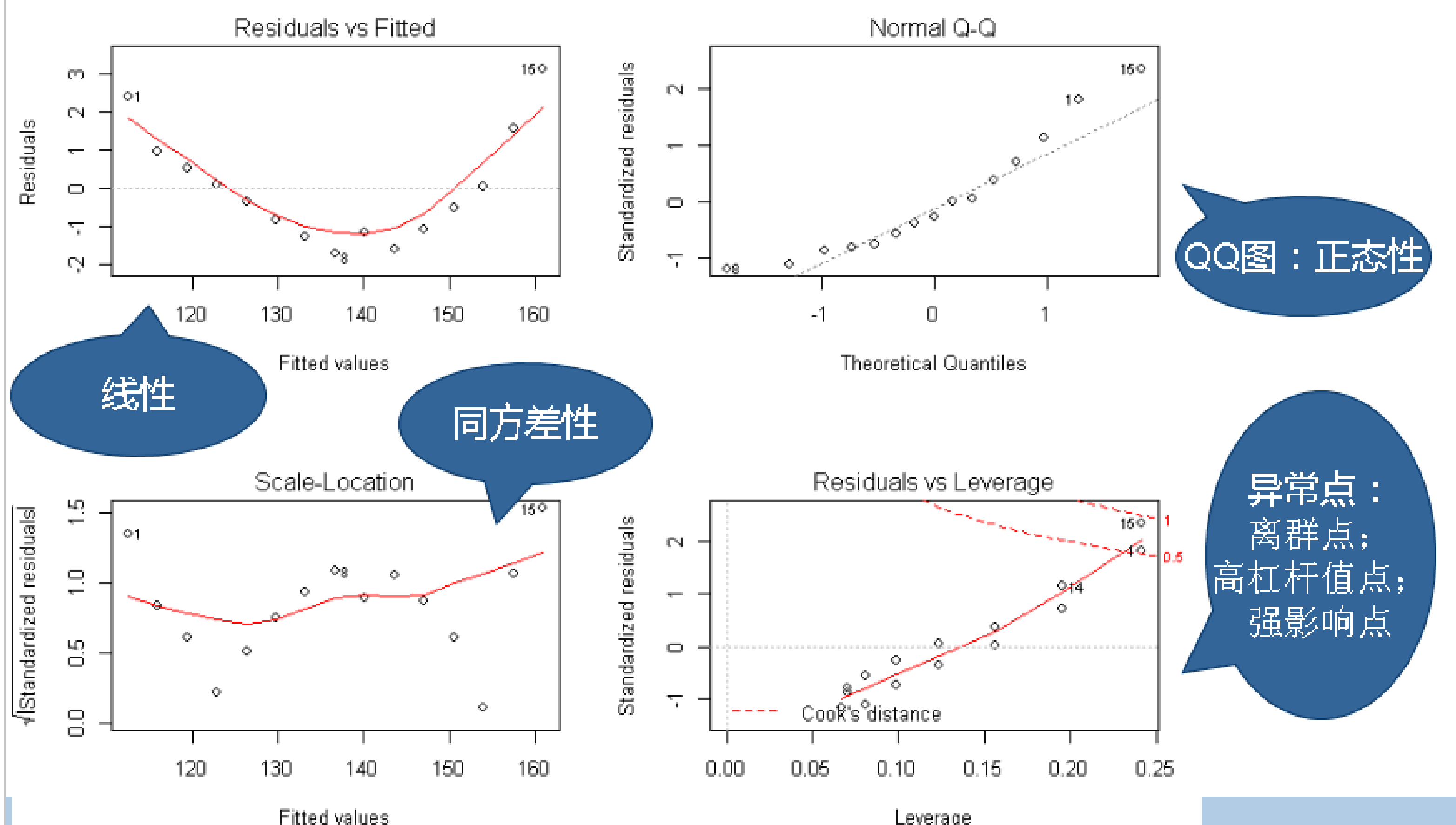


C

D

回归诊断：标准方法

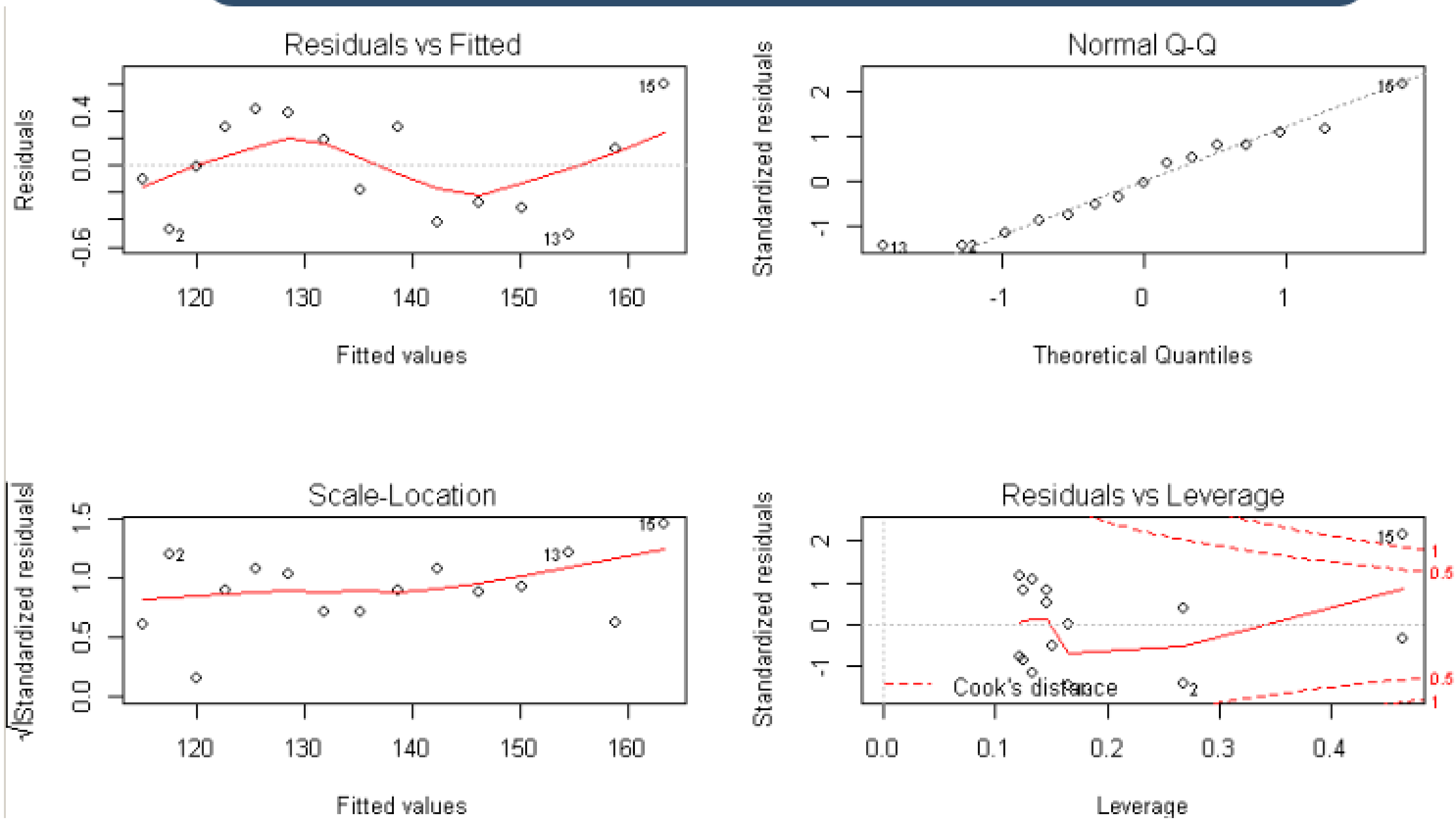
```
> opar<-par(no.readonly=TRUE)
> fit<-lm(weight~height,data=women)
> opar<-par(no.readonly=TRUE)
> par(mfrow=c(2,2))
> plot(fit)
```



回归分析

- 回归诊断

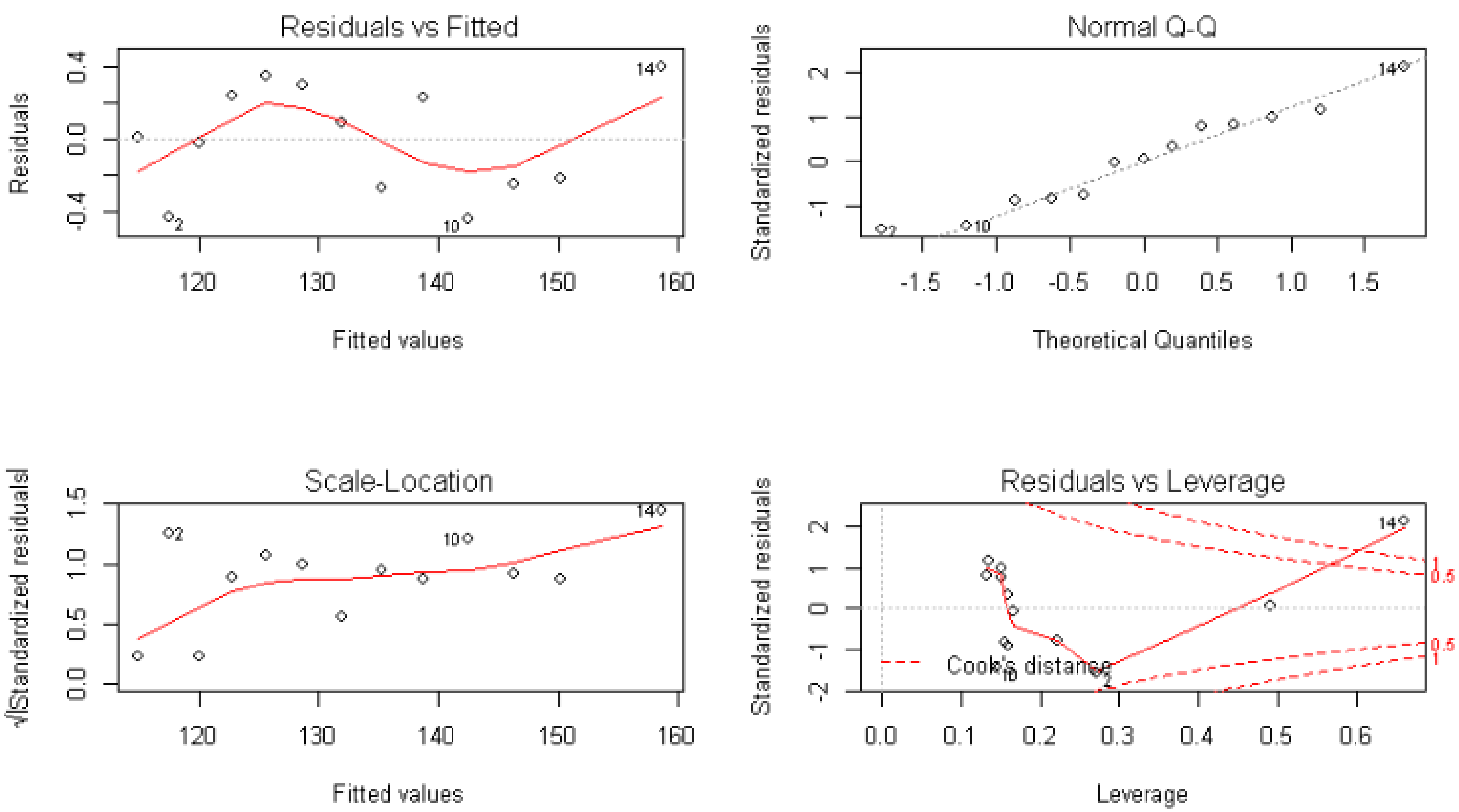
```
> fit<-lm(weight~height+I(height^2),data=women)
> plot(fit)
```



回归分析

➢ **回归诊断：剔除数据**

```
> fit<-lm(weight~height+I(height^2),data=women[-c(13,15),])
> plot(fit)
```



回归分析

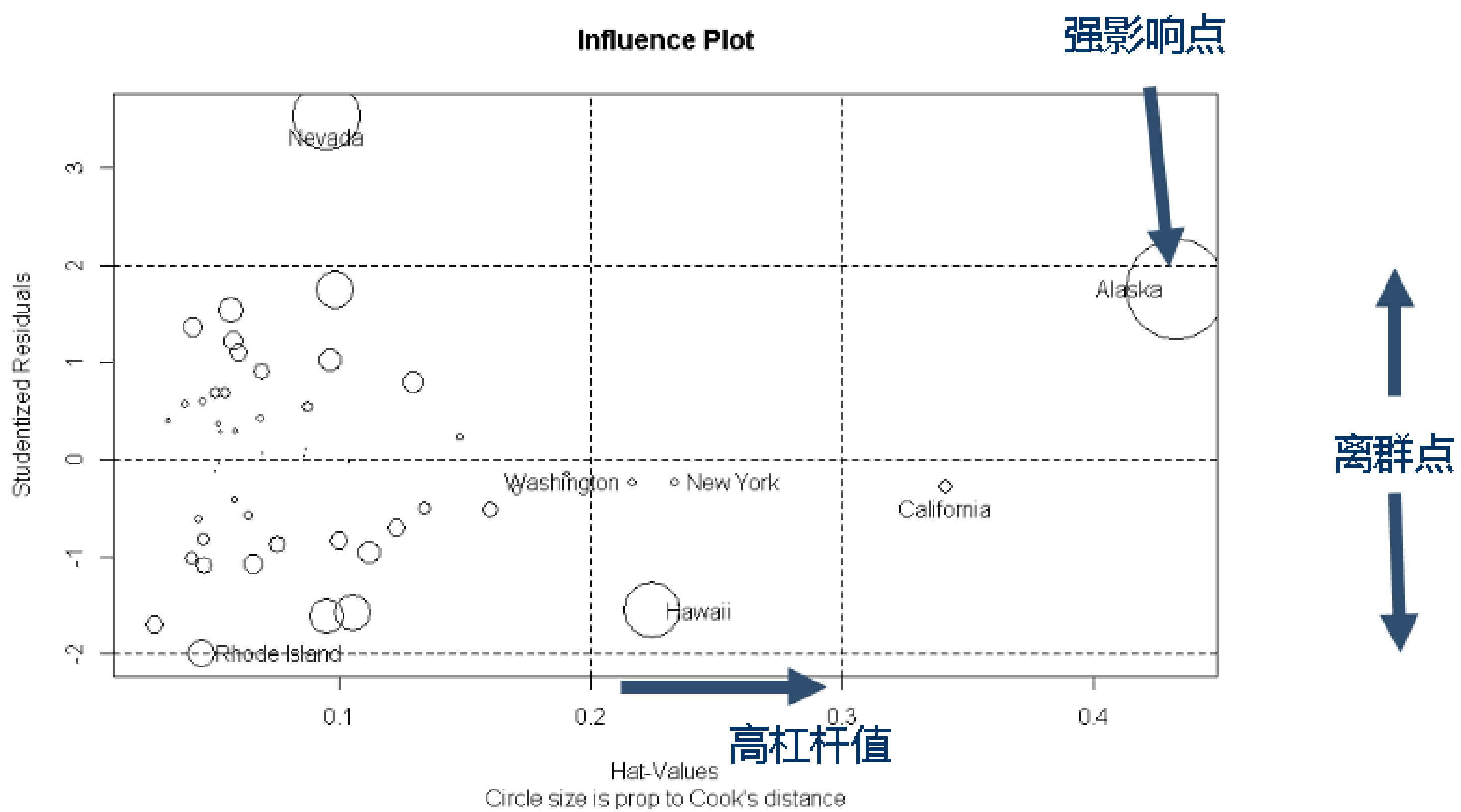
- **回归诊断**
- **更多改进的方法**
  - 正态性的诊断
  - 误差的独立性 Durbin-Watson自相关检验
  - 线性：成分残差图（偏残差图）
  - 同方差性
  - 多重共线性
    - 是指线性回归模型中的解释变量之间由于存在**精确相关关系或高度相关关系**而使模型估计失真或难以估计准确
    - 导致模型参数的置信区间过大
  - 异常点观测值
    - 离群点、高杠杆值点、强影响点

回归分析

➢ 回归诊断

➢ 异常点观测值

- 离群点、高杠杆值点、强影响点



回归分析

小象学院 ChinaHadoop.cn

- 1. 总离差平方和
- 2. 残差(误差)平方和
- 3. 回归平方和
- 4. 复相关系数

$$SST = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$SSR = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2$$

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

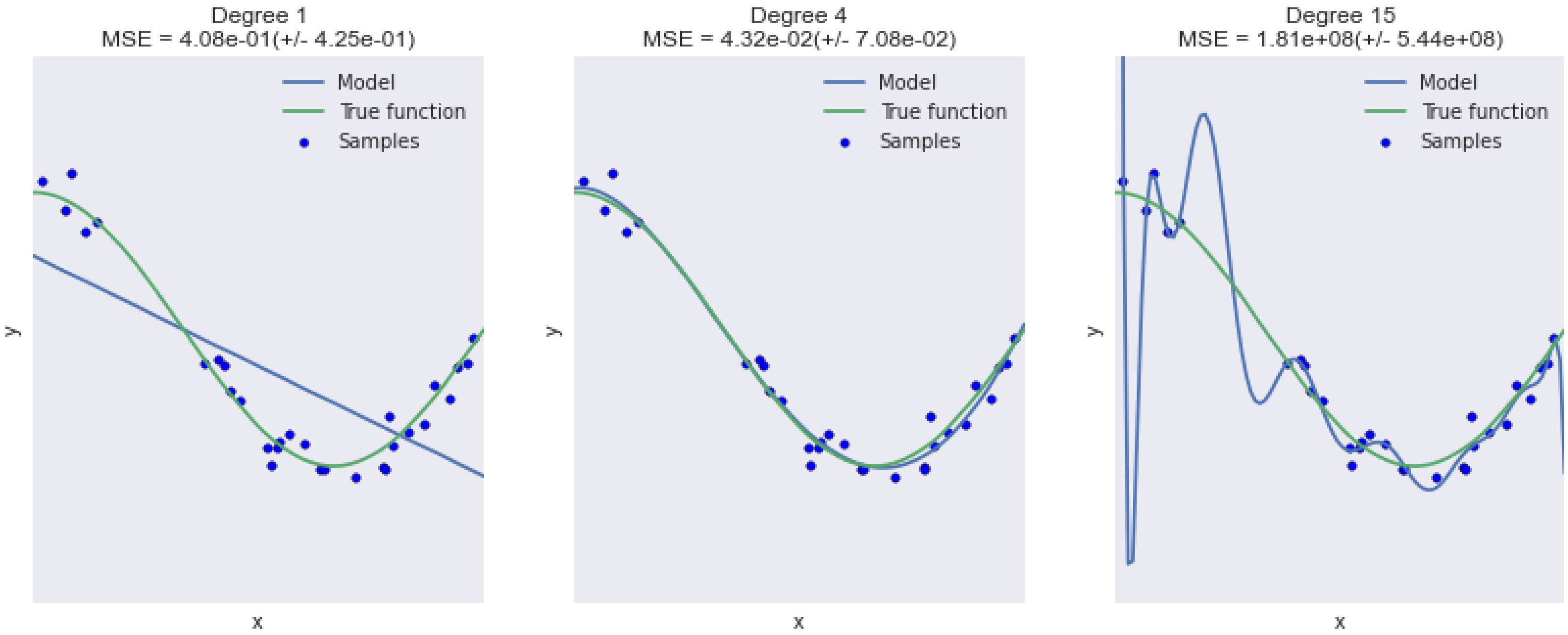
越大越显著

$$R_a^2(p) = 1 - (\frac{n-1}{n-p})\frac{SSE_p}{SST} = 1 - \frac{MSE_p}{SST/(n-1)}$$

- p增加时，SSEp减少，R^2_p增大，因此需要修正

回归分析

- 线性回归：模型选择
- 欠拟合 vs 过拟合



```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn import cross_validation

np.random.seed(0)

n_samples = 30
degrees = [1, 4, 15]

true_fun = lambda X: np.cos(1.5 * np.pi * X)
X = np.sort(np.random.rand(n_samples))
y = true_fun(X) + np.random.randn(n_samples) * 0.1

plt.figure(figsize=(14, 5))
for i in range(len(degrees)):
    ax = plt.subplot(1, len(degrees), i + 1)
    plt.setp(ax, xticks=(), yticks=())

    polynomial_features = PolynomialFeatures(degree=degrees[i],
                                             include_bias=False)
    linear_regression = LinearRegression()
    pipeline = Pipeline([("polynomial_features", polynomial_features),
                         ("linear_regression", linear_regression)])
    pipeline.fit(X[:, np.newaxis], y)

    # Evaluate the models using crossvalidation
    scores = cross_validation.cross_val_score(pipeline,
        X[:, np.newaxis], y, scoring="mean_squared_error", cv=10)

    X_test = np.linspace(0, 1, 100)
    plt.plot(X_test, pipeline.predict(X_test[:, np.newaxis]), label="Model")
    plt.plot(X_test, true_fun(X_test), label="True function")
    plt.scatter(X, y, label="Samples")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.xlim((0, 1))
    plt.ylim((-2, 2))
    plt.legend(loc="best")
    plt.title("Degree {}\nMSE = {:.2e}(+/- {:.2e})".format(
        degrees[i], -scores.mean(), scores.std()))
plt.show()
```

Python source code: plot_underfitting_overfitting.py


```python
print(__doc__)

import numpy as np
import matplotlib.pyplot as plt
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn import cross_validation

np.random.seed(0)

n_samples = 30
degrees = [1, 4, 15]

true_fun = lambda X: np.cos(1.5 * np.pi * X)
X = np.sort(np.random.rand(n_samples))
y = true_fun(X) + np.random.randn(n_samples) * 0.1

plt.figure(figsize=(14, 5))
for i in range(len(degrees)):
    ax = plt.subplot(1, len(degrees), i + 1)
    plt.setp(ax, xticks=(), yticks=())

    polynomial_features = PolynomialFeatures(degree=degrees[i],
                                             include_bias=False)
    linear_regression = LinearRegression()
    pipeline = Pipeline([("polynomial_features", polynomial_features),
                         ("linear_regression", linear_regression)])
    pipeline.fit(X[:, np.newaxis], y)

    # Evaluate the models using crossvalidation
    scores = cross_validation.cross_val_score(pipeline,
        X[:, np.newaxis], y, scoring="mean_squared_error", cv=10)

    X_test = np.linspace(0, 1, 100)
    plt.plot(X_test, pipeline.predict(X_test[:, np.newaxis]), label="Model")
    plt.plot(X_test, true_fun(X_test), label="True function")
    plt.scatter(X, y, label="Samples")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.xlim((0, 1))
    plt.ylim((-2, 2))
    plt.legend(loc="best")
    plt.title("Degree {}\nMSE = {:.2e}(+/- {:.2e})".format(
        degrees[i], -scores.mean(), scores.std()))
plt.show()
```

重抽样：基于模拟的分析方法

- **当数据抽样存在下列情况时，用什么方法？**
  - 来自未知或混合分布
  - 样本量过小
  - 存在离群点
  - 基于理论分布设计合适的统计检验过于复杂
- **置换检验**
- **自助法**

重抽样：置换检验

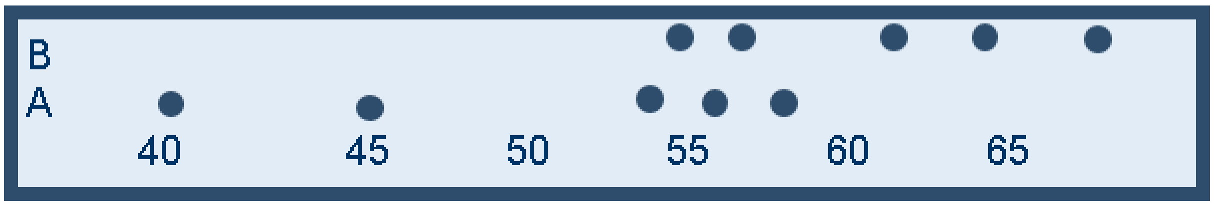
- 举例：10个受试者随机分两组进行A或B实验，结果是否有不同？
- 假设检验：t-test
  - 假设数据抽样自等方差的正态分布
  - 独立分组的双尾t检验
  - 观测t值是否极端

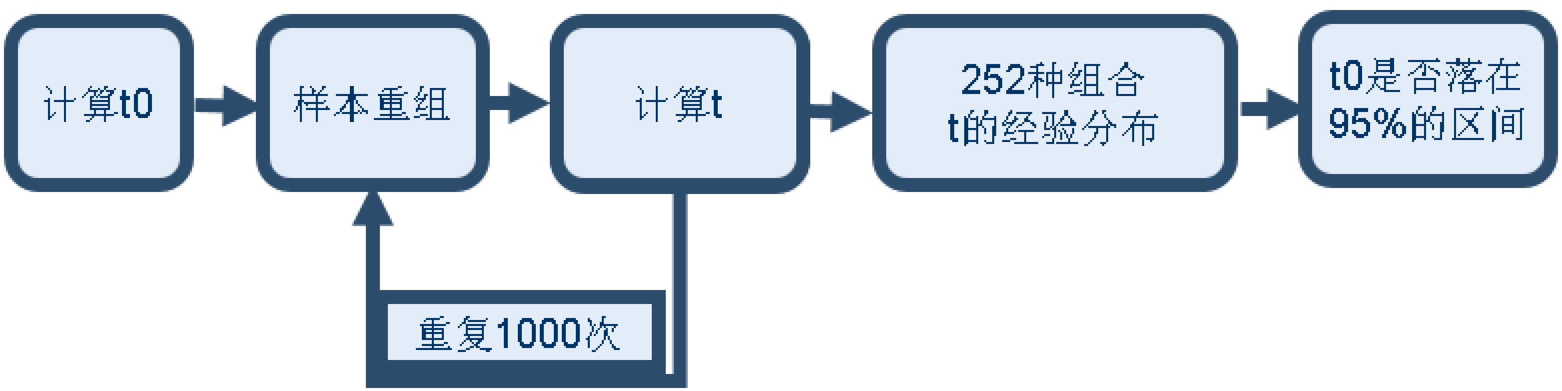
重抽样：置换检验

小象学院 ChinaHadoop.cn

➢ 置换检验：生成零假设的p值

- 置换检验的思路:
  - 如果A，B两种方式真的等价，AB的结果应该是任意的



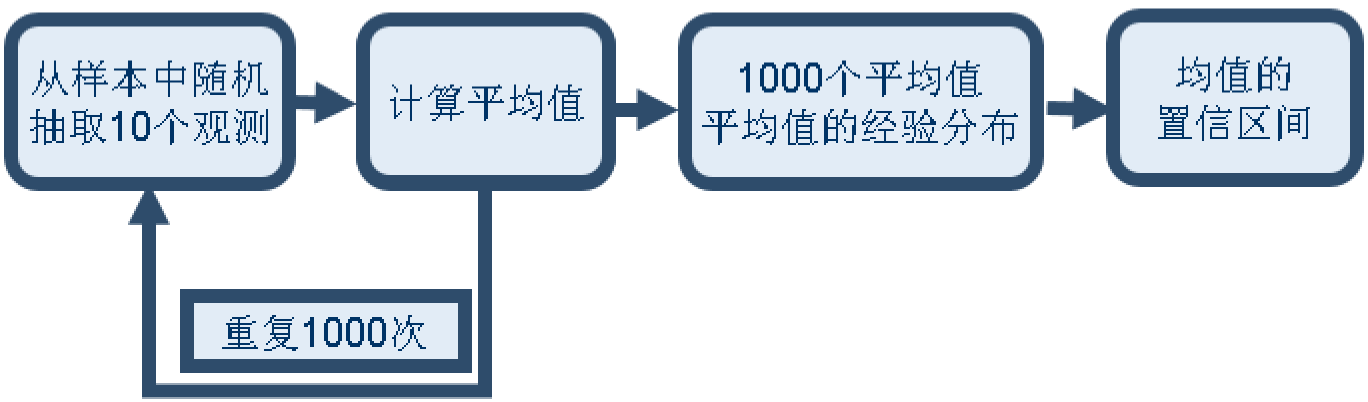


重抽样：自助法

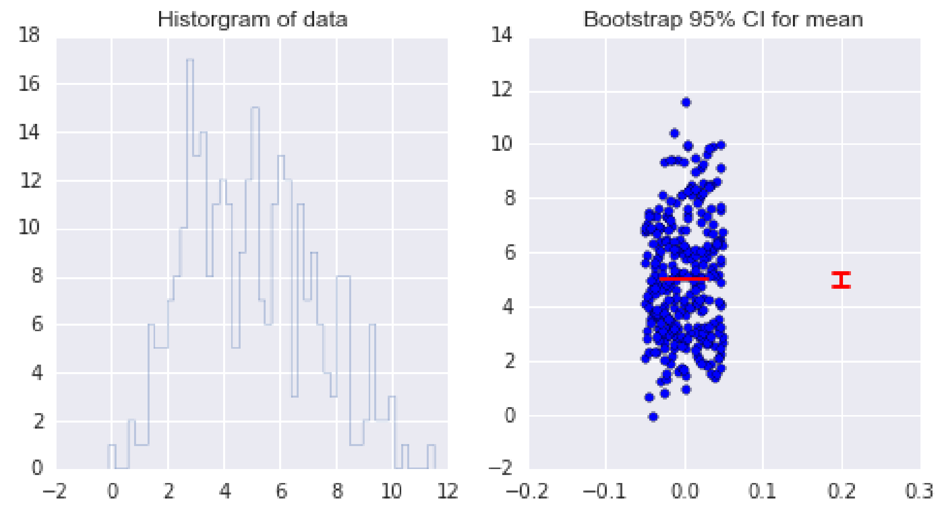
- 自助法：获取置信区间和估计测量精度
- 从初始样本重复随机替换抽样，生成待检验统计量的经验分布，生成统计量的置信区间
- 例：

$$\overline{X} - t\frac{s}{\sqrt{n}} < \mu < \overline{X} + t\frac{s}{\sqrt{n}}$$

t为分位数



重抽样：自助法




```python
# -*- coding: utf-8 -*-

import numpy as np
import numpy.random as npr
import pylab

def bootstrap(data, num_samples, statistic, alpha):
    """Returns bootstrap estimate of 100.0*(1-alpha) CI for statistic."""
    n = len(data)
    idx = npr.randint(0, n, (num_samples, n))
    samples = data[idx]
    stat = np.sort(statistic(samples, 1))
    return (stat[int((alpha/2.0)*num_samples)],
            stat[int((1-alpha/2.0)*num_samples)])


# data of interest is bimodal and obviously not normal
x = np.concatenate([npr.normal(3, 1, 100), npr.normal(6, 2, 200)])

# find mean 95% CI and 100,000 bootstrap samples
low, high = bootstrap(x, 100000, np.mean, 0.05)

# make plots
pylab.figure(figsize=(8,4))
pylab.subplot(121)
pylab.hist(x, 50, histtype='step')
pylab.title('Historgram of data')
pylab.subplot(122)
pylab.plot([-0.03,0.03], [np.mean(x), np.mean(x)], 'r', linewidth=2)
pylab.scatter(0.1*(npr.random(len(x))-0.5), x)
pylab.plot([0.19,0.21], [low, low], 'r', linewidth=2)
pylab.plot([0.19,0.21], [high, high], 'r', linewidth=2)
pylab.plot([0.2,0.2], [low, high], 'r', linewidth=2)
pylab.xlim([-0.2, 0.3])
pylab.title('Bootstrap 95% CI for mean')
```

by郭鹏程（绿树@小象）

重抽样：基于模拟的分析

- **自助法：**
  - 初始样本 20~30
  - 重复次数 >~ 1000

- **重抽样与自助法:**
  - 数据不满足标准的假设时
  - 非万能：无法将烂数据转化为好数据

回归分析

- 练习
- 回归分析：
  - tip~ total_bill+size

联系我们：

– 新浪微博：ChinaHadoop
– 微信公号：ChinaHadoop
– 网站：http://chinahadoop.cn
– 问答社区：http://wenda.ChinaHadoop.cn