1 一元线性回归

2 多元线性回归

Lab: 线性回归

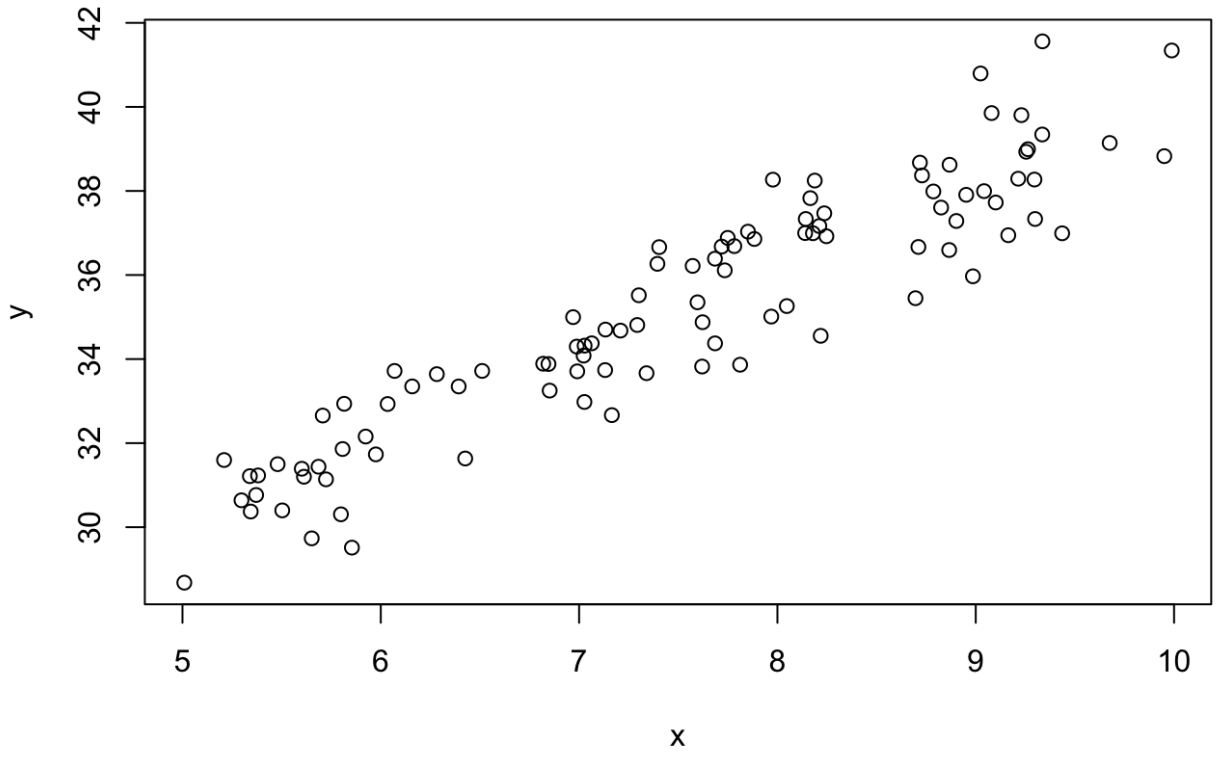
李文东

最后编译于 10/10/2022

1 一元线性回归

1.1 产生模拟数据

```
set.seed(2022)
x <- runif(100,5,10)
y <- 20+2*x+rnorm(100)
plot(x,y)
```



1.2 最小二乘估计

我们首先使用 `lm()` 函数来拟合一元线性回归模型，其中 `x` 作为特征， `y` 作为响应变量。

基本语法是 lm(y~x,data=) ，其中 data 代表了储存两个变量的数据集。若 x 与 y 不在数据集中，则可以直接调用。

```
lm.fit <- lm(y~x)
```

我们来看一个特征和响应变量在数据集中的例子： ISLR2 包中 Boston 数据集中的变量 medv 与 lstat 。下面第一行代码报错，原因是R不知道去哪里找这两个变量。第二行代码正常运行，因为我们告诉了R变量在 Boston 数据集中。

```
library(ISLR2)
##
## 载入程辑包：'ISLR2'
## The following object is masked from 'package:MASS':
##
##     Boston
lm.tem <- lm(medv ~ lstat)
## Error in eval(predvars, data, env): 找不到对象'medv'
lm.tem <- lm(medv ~ lstat,data=Boston)
```

回到 lm.fit 。如果我们输入 lm.fit ，一些关于模型的基本信息就会输出。如果需要更加详细的信息，我们可以输入 summary(lm.fit) ，这会给我们提供系数/参数的p值与标准误、R^2统计量等等。

```
lm.fit
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##      19.942        2.032
```

```
summary(lm.fit)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.32603 -0.69970  0.09282  0.83237  2.64476
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 19.94178    0.62378   31.97   <2e-16 ***
## x            2.03223    0.08192   24.81   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.09 on 98 degrees of freedom
## Multiple R-squared:  0.8626, Adjusted R-squared:  0.8612
## F-statistic: 615.4 on 1 and 98 DF,  p-value: < 2.2e-16
```

我们可以利用 names() 函数来看看在 lm.fit 模型中存储了哪些信息。

尽管我们可以通过名字提取这些信息，比如 lm.fit$coefficients ，往往更加安全的方式是利用像 coef() 一样的提取函数。

```
names(lm.fit)
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
coef(lm.fit)
## (Intercept)           x
##   19.941778    2.032228
lm.fit$coefficients
## (Intercept)           x
##   19.941778    2.032228
```

函数 predict() 可以用来产生样本内/样本外的预测值/置信区间等等。如果不给定 newdata= ，则会默认输出样本内预测。

```
predict(lm.fit)
```

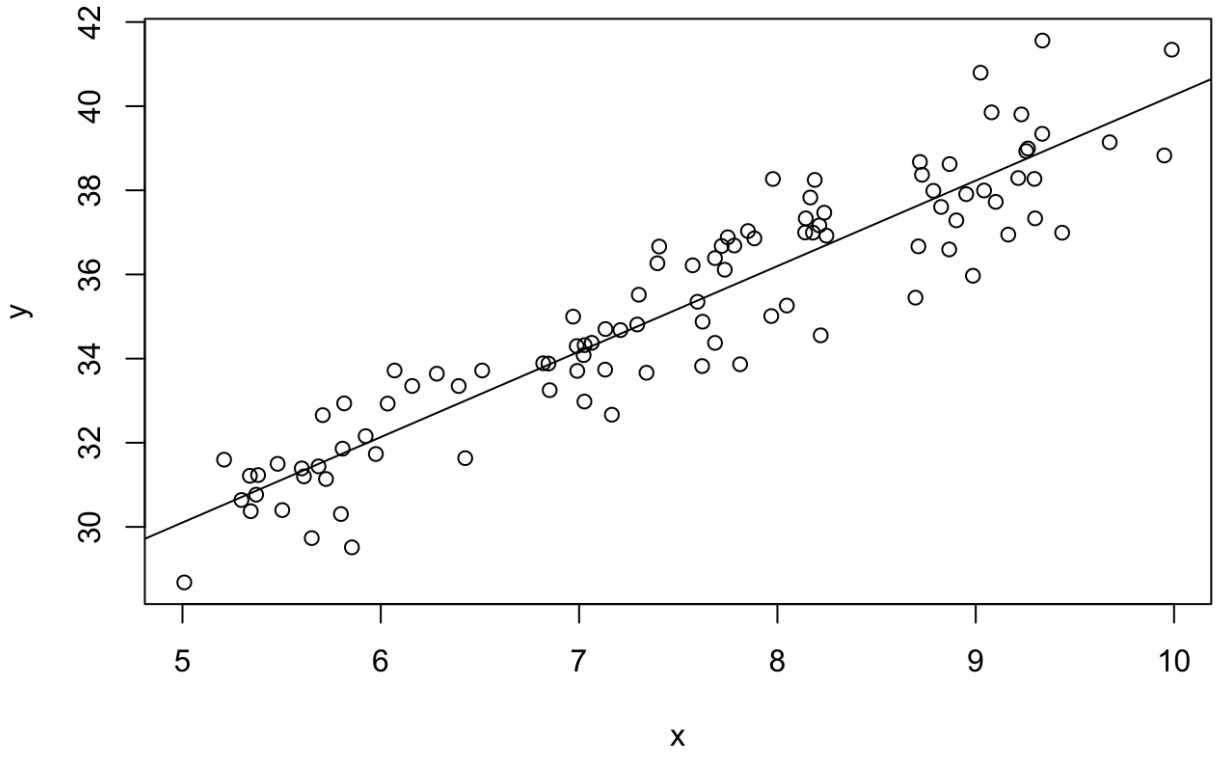
```
##        1        2        3        4        5        6        7        8
## 38.39418 36.67981 31.32559 35.62854 31.97998 36.56327 30.85788 30.52944
##        9       10       11       12       13       14       15       16
## 33.86576 37.79747 30.12184 31.72666 31.57366 35.38058 36.29589 31.34776
##       17       18       19       20       21       22       23       24
## 37.95969 35.33091 36.15335 38.74913 38.66710 35.89808 34.43722 31.49708
##       25       26       27       28       29       30       31       32
## 31.76012 39.60441 35.55956 40.24022 34.96961 37.96367 34.85821 36.58199
##       33       34       35       36       37       38       39       40
## 32.70948 34.22188 31.83928 30.79363 38.69956 35.69030 38.28103 37.87684
##       41       42       43       44       45       46       47       48
## 36.62884 34.59109 36.70137 35.43279 38.91499 31.12559 39.11737 36.48365
##       49       50       51       52       53       54       55       56
## 33.79952 38.76683 32.08452 33.00038 37.64413 33.85358 36.48987 35.96339
##       57       58       59       60       61       62       63       64
## 33.17410 34.43404 34.22386 31.54046 35.65982 35.56023 38.20296 38.43646
##       65       66       67       68       69       70       71       72
## 30.70758 38.56504 34.29676 34.76314 32.20534 36.13648 38.83324 38.91310
##       73       74       75       76       77       78       79       80
## 38.84001 38.03279 35.75770 30.80251 35.81723 37.65971 34.50277 34.98801
##       81       82       83       84       85       86       87       88
## 34.10590 34.21349 38.13416 37.61438 31.42747 34.14832 30.87766 32.45706
##       89       90       91       92       93       94       95       96
## 35.42800 38.31707 32.27632 31.74390 36.53670 34.77923 34.14336 32.93324
##       97       98       99      100
## 31.07810 40.16387 36.64291 37.68103
```

```
predict(lm.fit,newdata=data.frame(x = c(0,10,100)))
```

```
##         1         2         3
##  19.94178  40.26405 223.16453
```

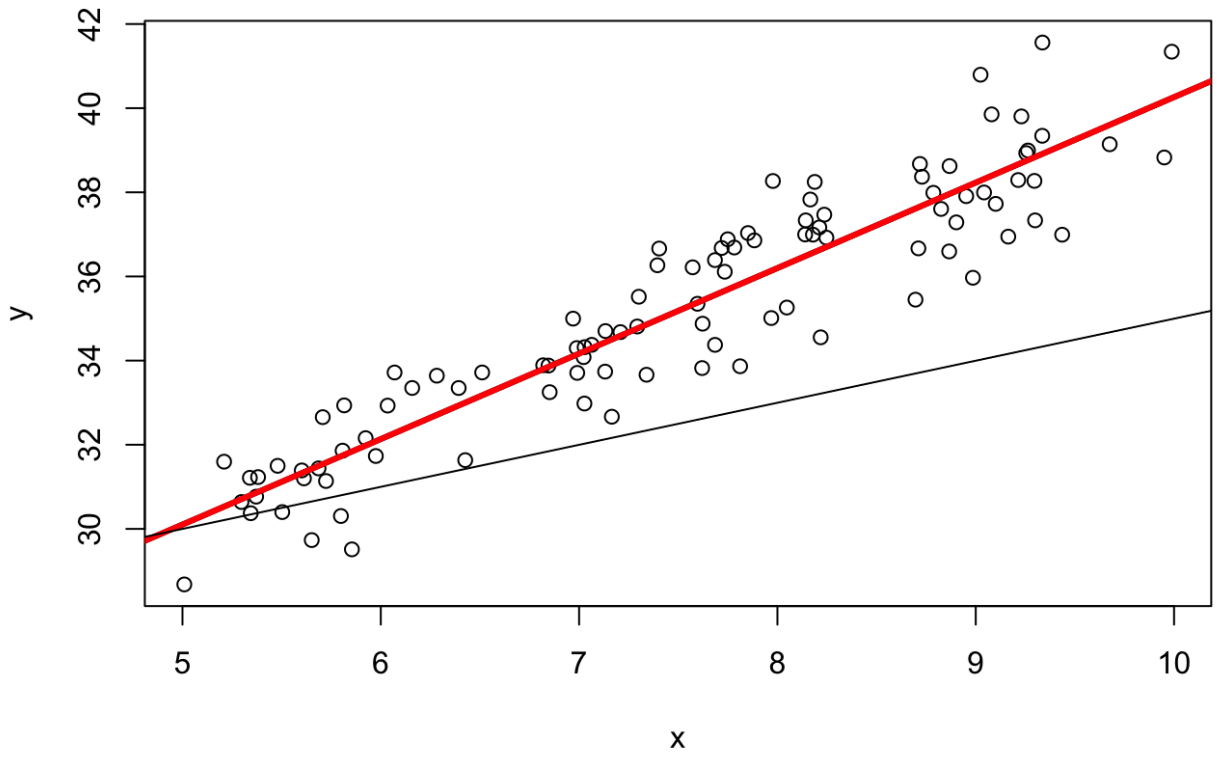
现在我们利用 plot() 和 abline() 函数画出 x 和 y 以及最小二乘回归线。

```
plot(x,y)
abline(lm.fit)
```

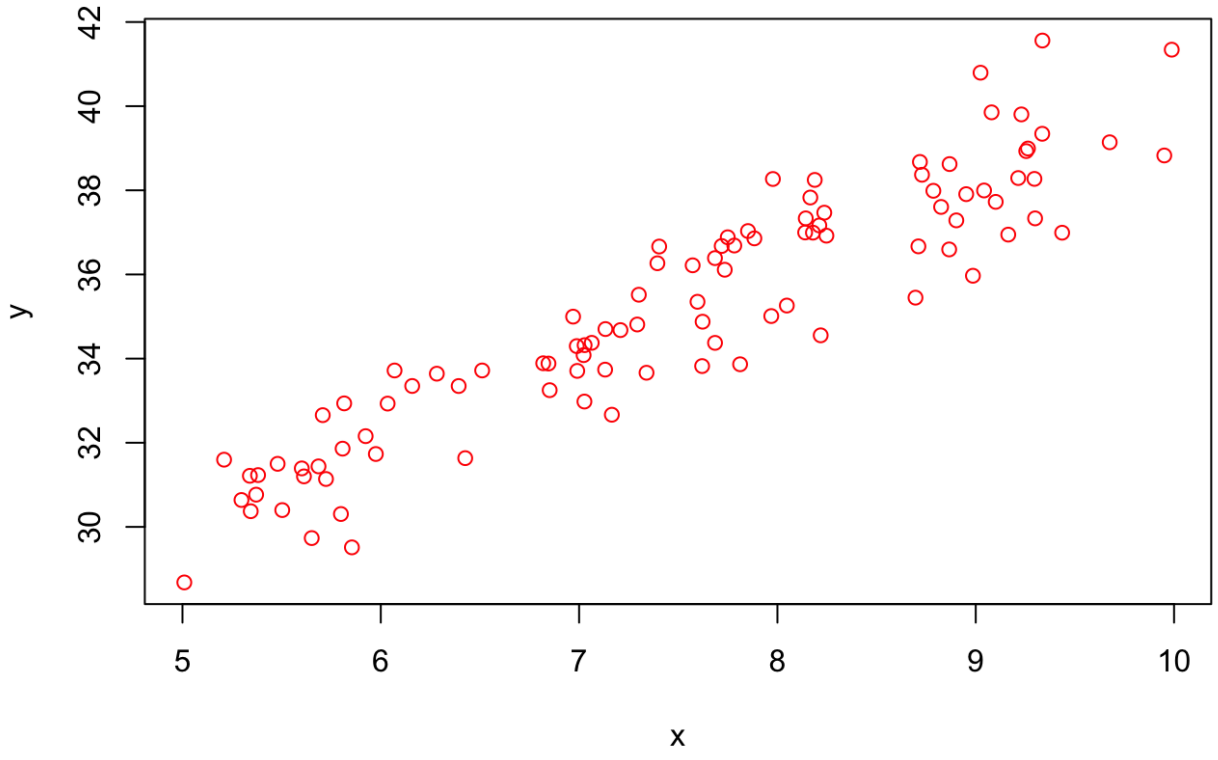


abline() 函数可以用来画任何线，不仅仅是最小二乘回归线。输入 abline(a,b) 可以画出截距项为 a 斜率为 b 的直线。下面我们给出多个画图参数示例。

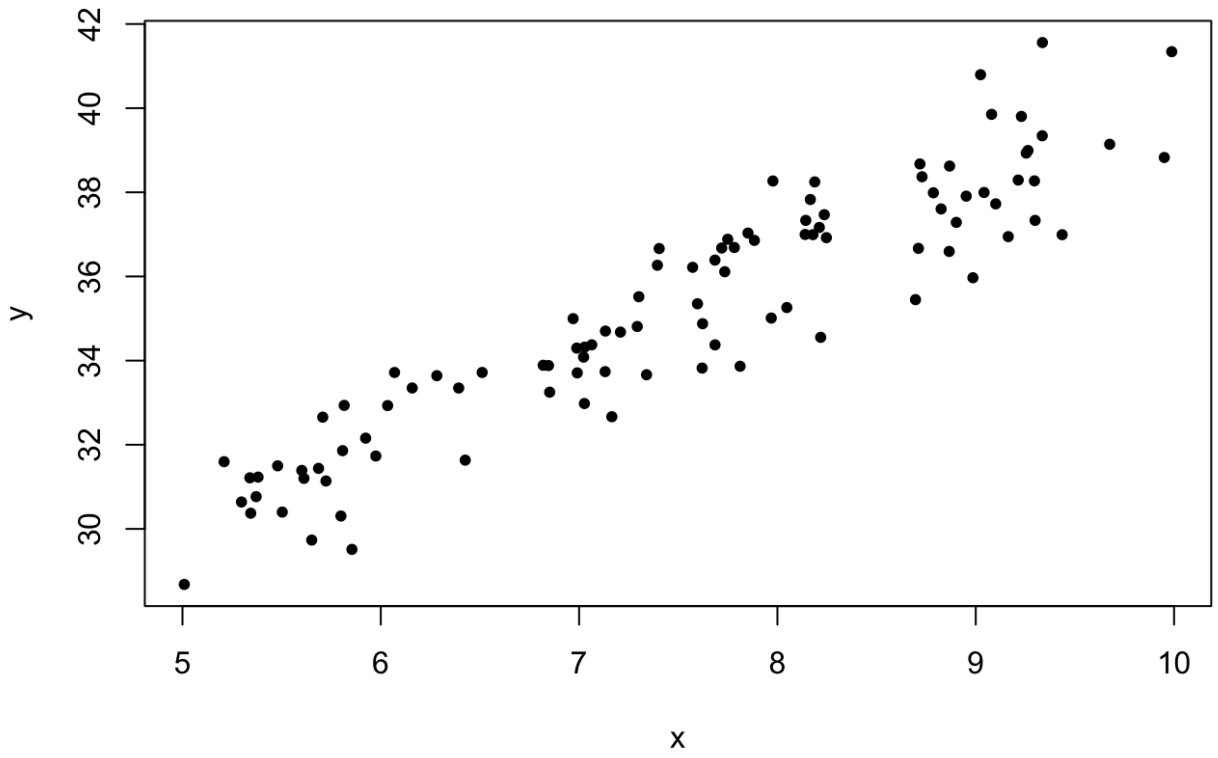
```
plot(x,y)
abline(lm.fit, lwd = 3)
abline(lm.fit, lwd = 3, col = "red")
abline(25,1)
```



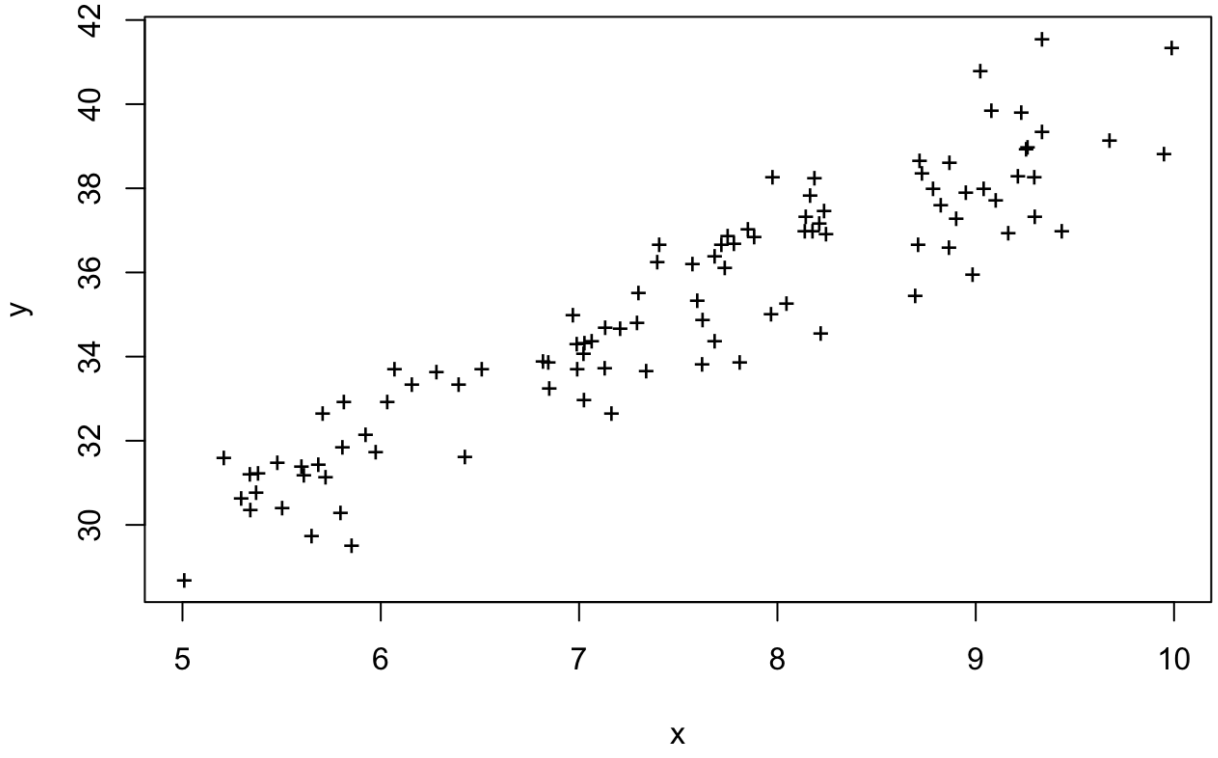

```
plot(x,y, col = "red")
```



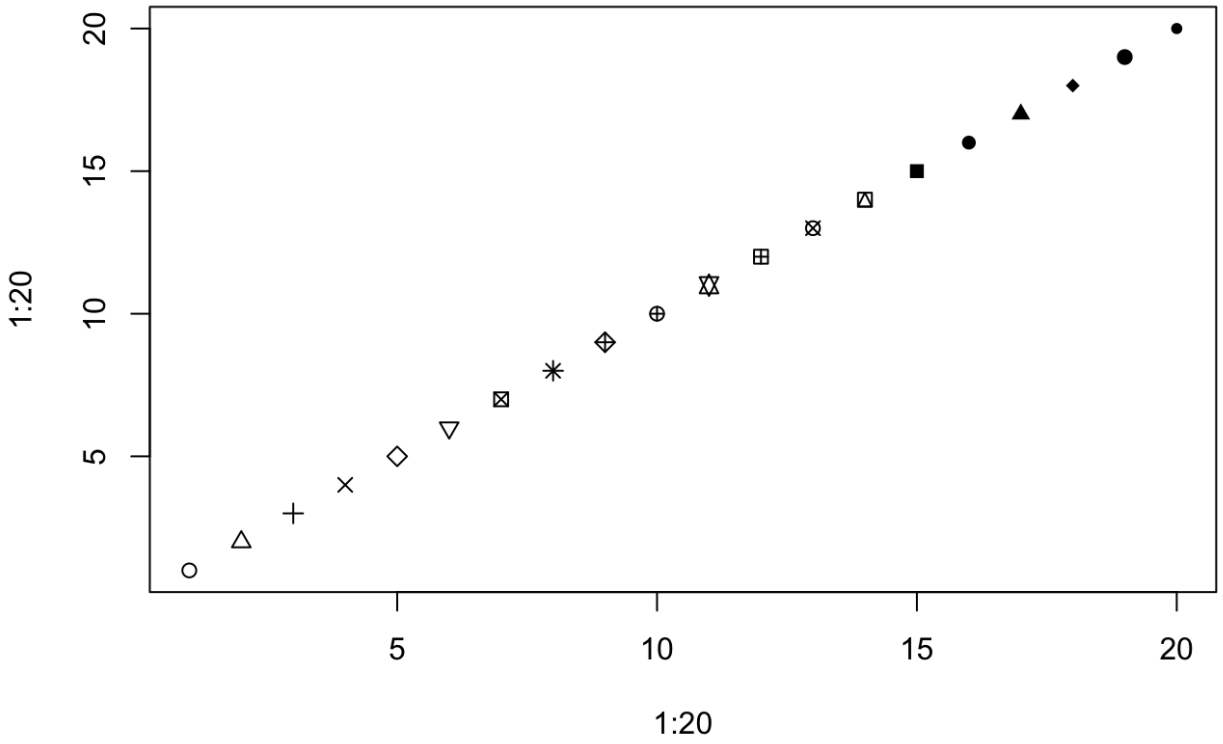

```
plot(x,y, pch = 20)
```




```
plot(x,y, pch = "+")
```




```
plot(1:20, 1:20, pch = 1:20)
```



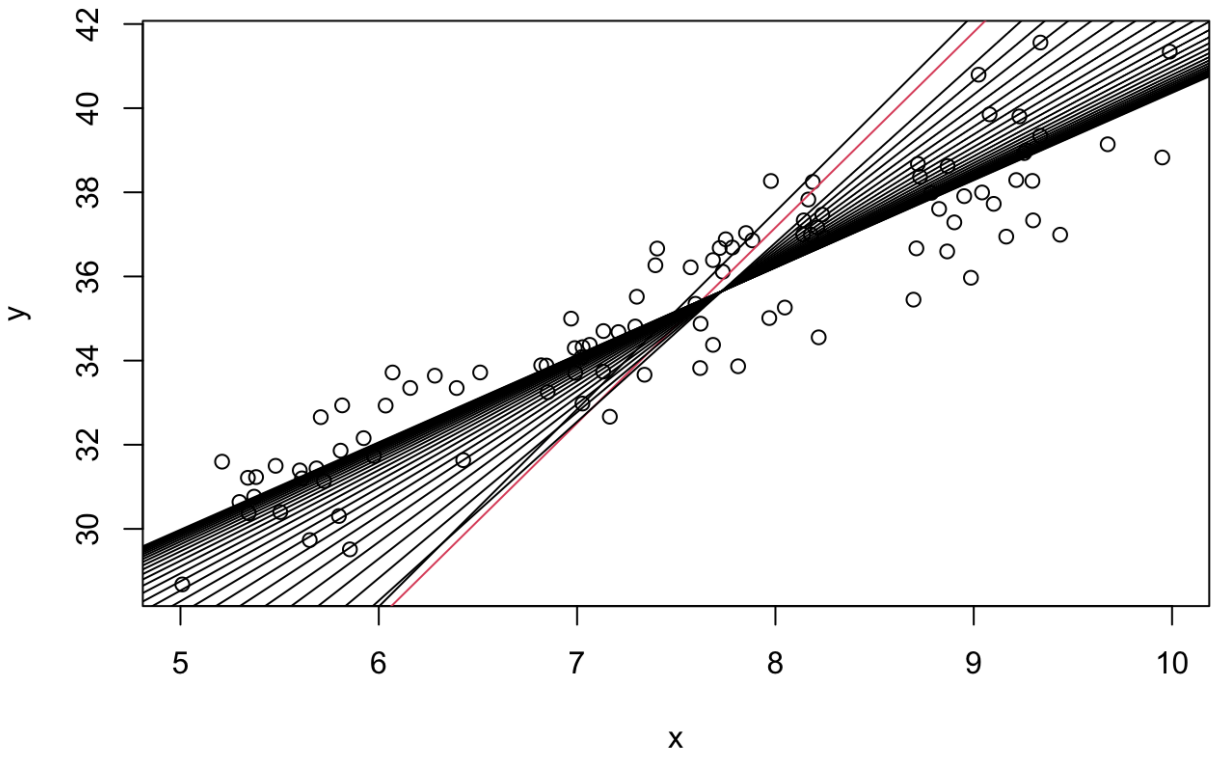
1.3 梯度下降算法

我们首先定义损失函数，方便调用。

```
cost <- function(x,y,beta){
  return(sum((y-cbind(1,x) %*% beta)^2)/(2*length(y)))
}
```

接下来我们编写梯度下降算法来实现线性回归。

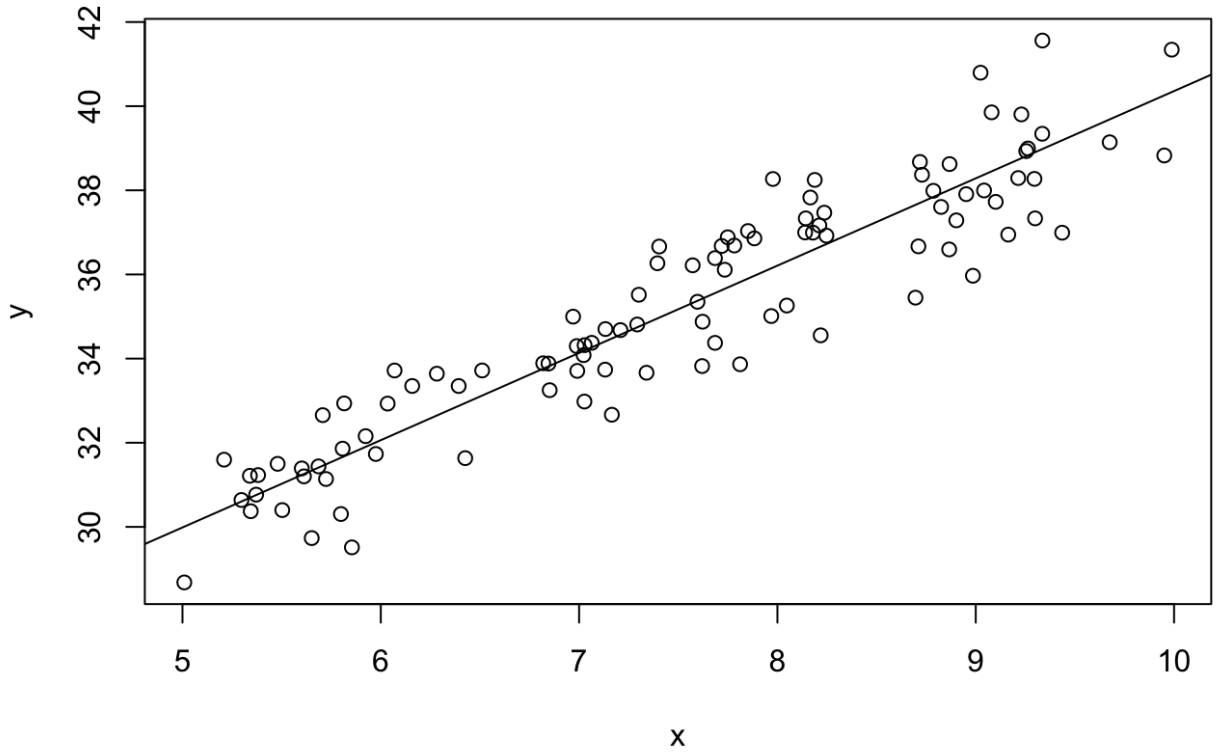
```
alpha <- 0.01   #learning rate
n <- length(y)    #training sample size

beta <- matrix(0,2,1)
beta[1] <- 0
beta[2] <- mean(y)/mean(x)
plot(x,y)
abline(beta[1],beta[2])
cost.history <- cost(x,y,beta)

tem.beta1 <- beta[1]+alpha*(1/n)*sum(y-beta[1]-beta[2]*x)
tem.beta2 <- beta[2]+alpha*(1/n)*sum((y-beta[1]-beta[2]*x)*x)
beta[1] <- tem.beta1; beta[2] <- tem.beta2
abline(beta[1],beta[2],col=2)
cost.history <- c(cost.history,cost(x,y,beta))

index <- 1
while ( abs(cost.history[index+1]-cost.history[index])>0.000001) {
  index <- index+1
  tem.beta1 <- beta[1]+alpha*(1/n)*sum(y-beta[1]-beta[2]*x)
  tem.beta2 <- beta[2]+alpha*(1/n)*sum((y-beta[1]-beta[2]*x)*x)
  beta[1] <- tem.beta1; beta[2] <- tem.beta2
  if (index%%500==0) {
    abline(beta[1],beta[2])
  }
  cost.history <- c(cost.history,cost(x,y,beta))
}
```



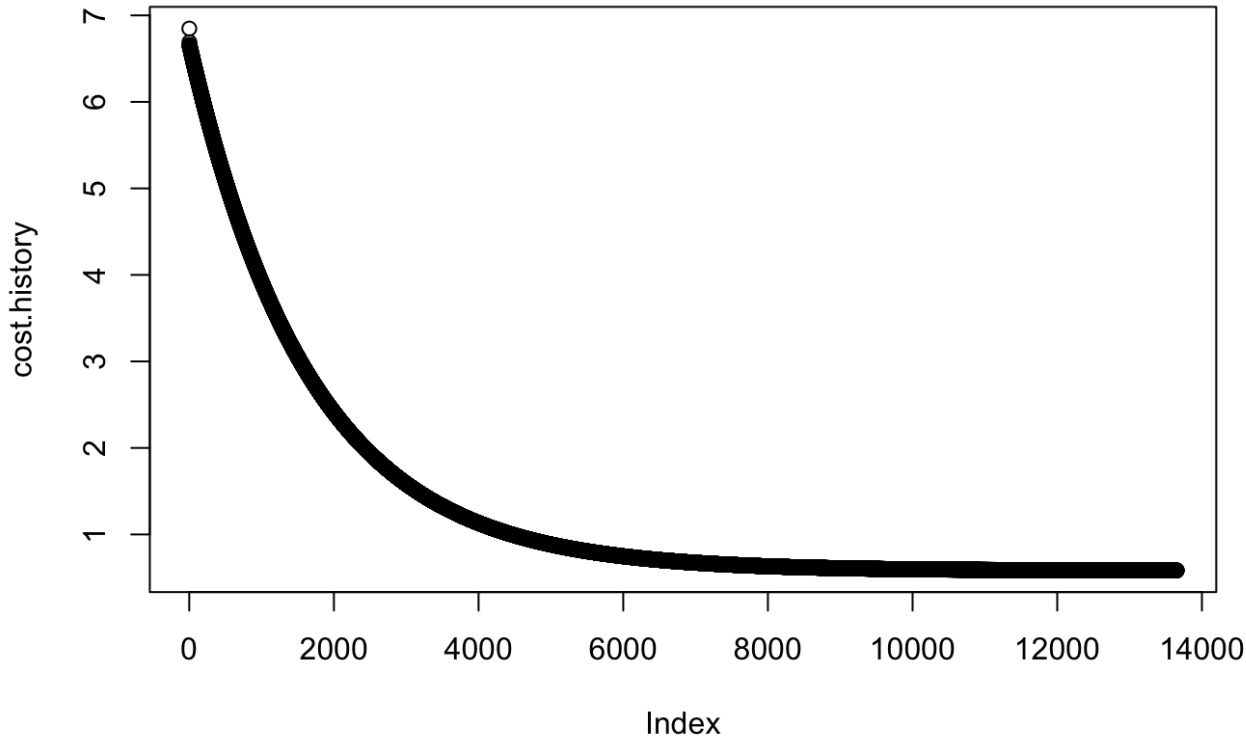
如果你掌握了 `shiny` 包的使用方式，上面的图可以做成动态交互的形式，非常的fancy。

下面我们来看一些迭代结束后的结果：

```
index #迭代次数
## [1] 13651
beta  #最终参数
##           [,1]
## [1,] 19.611799
## [2,]  2.074919
cost.history[length(cost.history)] #最终损失函数
## [1] 0.5840761
plot(x,y)
abline(beta[1],beta[2])
```




```
plot(cost.history)
```



练习

除了上面的batch梯度下降算法，尝试编写随机梯度下降和mini-batch梯度下降算法求解上面的线性回归问题，并展示三个梯度下降算法的不同（如迭代次数，参数值，损失函数变化趋势等等）。

2 多元线性回归

2.1 最小二乘估计

为了实现多元线性回归的最小二乘估计，我们仍然使用 `lm()` 函数，其语法为 `lm(y~x1+x2+x3, data=)` 。

这次我们采用一个在 `ISLR2` 包中的 `Boston` 实际数据集，其中记录了波士顿506个人口普查街区的 `medv` (房屋价值中位数)。我们尝试利用12个特征比如 `rm` (平均房间数)、 `age` (平均房龄)、和 `lstat` (低社会地位房屋比例)来对 `medv` 进行预测。

```
library(ISLR2)
head(Boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio lstat medv
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3  4.98 24.0
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8  9.14 21.6
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8  4.03 34.7
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7  2.94 33.4
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7  5.33 36.2
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7  5.21 28.7
```

```
lm.fit <- lm(medv ~ lstat + age, data = Boston)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat + age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.981  -3.978  -1.283   1.968  23.158
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 33.22276    0.73085  45.458  < 2e-16 ***
## lstat       -1.03207    0.04819 -21.416  < 2e-16 ***
## age          0.03454    0.01223   2.826  0.00491 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.173 on 503 degrees of freedom
## Multiple R-squared:  0.5513, Adjusted R-squared:  0.5495
## F-statistic:   309 on 2 and 503 DF,  p-value: < 2.2e-16
```

Boston 数据集包含了12个特征，为了使用所有特征构造线性回归，将他们全部打出来会过于繁琐。为此，我们可以使用下面的简单形式：

```
lm.fit <- lm(medv ~ ., data = Boston)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ ., data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.1304  -2.7673  -0.5814   1.9414  26.2526
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  41.617270   4.936039   8.431 3.79e-16 ***
## crim         -0.121389   0.033000  -3.678 0.000261 ***
## zn            0.046963   0.013879   3.384 0.000772 ***
## indus         0.013468   0.062145   0.217 0.828520
## chas          2.839993   0.870007   3.264 0.001173 **
## nox         -18.758022   3.851355  -4.870 1.50e-06 ***
## rm            3.658119   0.420246   8.705  < 2e-16 ***
## age           0.003611   0.013329   0.271 0.786595
## dis          -1.490754   0.201623  -7.394 6.17e-13 ***
## rad           0.289405   0.066908   4.325 1.84e-05 ***
## tax          -0.012682   0.003801  -3.337 0.000912 ***
## ptratio      -0.937533   0.132206  -7.091 4.63e-12 ***
## lstat        -0.552019   0.050659 -10.897  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.798 on 493 degrees of freedom
## Multiple R-squared:  0.7343, Adjusted R-squared:  0.7278
## F-statistic: 113.5 on 12 and 493 DF,  p-value: < 2.2e-16
```

如果我们想要利用除了某一个特征之外的全部特征构造线性回归呢？例如，在上面的回归模型的输出中， `age` 的p值很大，代表其不显著。所以我们可能想要去掉这个特征：

```
lm.fit1 <- lm(medv ~ . - age, data = Boston)
summary(lm.fit1)
```

```
##
## Call:
## lm(formula = medv ~ . - age, data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.1851  -2.7330  -0.6116   1.8555  26.3838
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  41.525128   4.919684   8.441 3.52e-16 ***
## crim         -0.121426   0.032969  -3.683 0.000256 ***
## zn            0.046512   0.013766   3.379 0.000785 ***
## indus         0.013451   0.062086   0.217 0.828577
## chas          2.852773   0.867912   3.287 0.001085 **
## nox         -18.485070   3.713714  -4.978 8.91e-07 ***
## rm            3.681070   0.411230   8.951  < 2e-16 ***
## dis          -1.506777   0.192570  -7.825 3.12e-14 ***
## rad           0.287940   0.066627   4.322 1.87e-05 ***
## tax          -0.012653   0.003796  -3.333 0.000923 ***
## ptratio      -0.934649   0.131653  -7.099 4.39e-12 ***
## lstat        -0.547409   0.047669 -11.483  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.794 on 494 degrees of freedom
## Multiple R-squared:  0.7343, Adjusted R-squared:  0.7284
## F-statistic: 124.1 on 11 and 494 DF,  p-value: < 2.2e-16
```

除此之外，`update()` 函数也可以使用。

```
lm.fit2 <- update(lm.fit1, ~ . - indus)
summary(lm.fit2)
```

```
##
## Call:
## lm(formula = medv ~ crim + zn + chas + nox + rm + dis + rad +
##     tax + ptratio + lstat, data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.1814  -2.7625  -0.6243   1.8448  26.3920
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  41.451747   4.903283   8.454 3.18e-16 ***
## crim         -0.121665   0.032919  -3.696 0.000244 ***
## zn            0.046191   0.013673   3.378 0.000787 ***
## chas          2.871873   0.862591   3.329 0.000935 ***
## nox         -18.262427   3.565247  -5.122 4.33e-07 ***
## rm            3.672957   0.409127   8.978  < 2e-16 ***
## dis          -1.515951   0.187675  -8.078 5.08e-15 ***
## rad           0.283932   0.063945   4.440 1.11e-05 ***
## tax          -0.012292   0.003407  -3.608 0.000340 ***
## ptratio      -0.930961   0.130423  -7.138 3.39e-12 ***
## lstat        -0.546509   0.047442 -11.519  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.789 on 495 degrees of freedom
## Multiple R-squared:  0.7342, Adjusted R-squared:  0.7289
## F-statistic: 136.8 on 10 and 495 DF,  p-value: < 2.2e-16
```

多元线性回归的梯度下降算法实现和一元类似，引入一些矩阵运算即可，感兴趣的同学可以自己尝试，我们不再展开。