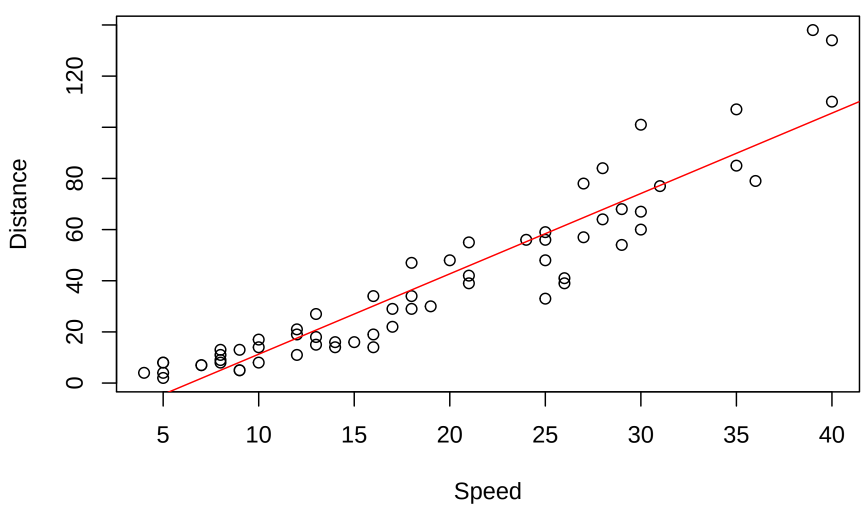
应用回归分析第六章

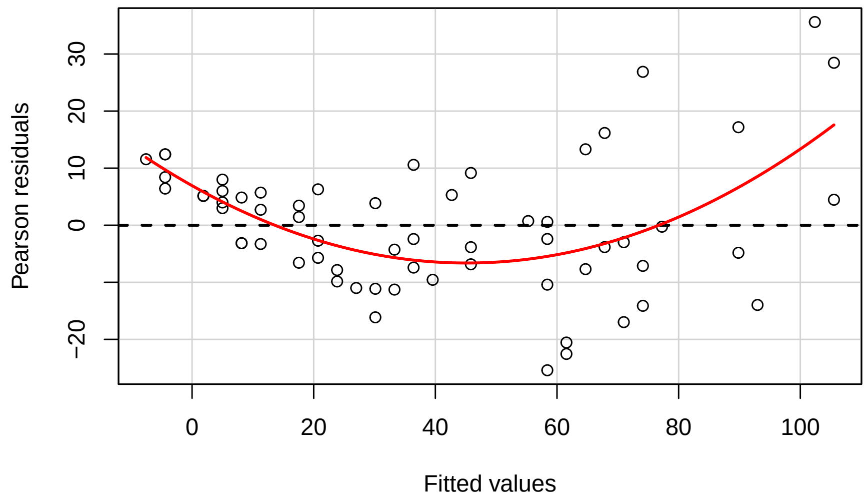
邵智轩　　1400012141　　物理学院

6.4.1

```r
library(alr4)
attach(stopping)
# 6.4.1
plot(Distance ~ Speed)  #Distance 关于 Speed 的散点图
lm.0 <- lm(Distance ~ Speed)
abline(lm.0, col = "red")
```



```r
# summary(lm.0)
residualPlot(lm.0)
```



残差图有明显的 U 型 pattern，暗示非线性；而且残差的方差随拟合值增大。

下面计算拟合失真的 F 检验。

```r
SSE <- function(X, Y) {
    SUM <- 0
    df <- 0
    for (x in unique(X)) {
        Yi <- Y[X == x]
        n <- length(Yi)
        SUM<-SUM+sum((Yi-mean(Yi))^2)
        df <- df + n - 1
    }
    c(SUM, df)
}
SSpe <- SSE(Speed, Distance)
```

```
SSreg <- sum(lm.0$residuals ^ 2)
F.stat <- ((SSreg - SSpe[1]) / (lm.0$df.residual - SSpe[2])) / (SSpe[1] / SSpe[2])
p.stat <- pf(F.stat, df1 = lm.0$df.residual - SSpe[2], df2 = SSpe[2],lower.tail = FALSE
p.stat
```

```
## [1] 0.02529981
```

在 $\alpha = 0.05$ 的水平下是显著的，认为确实有拟合失真。

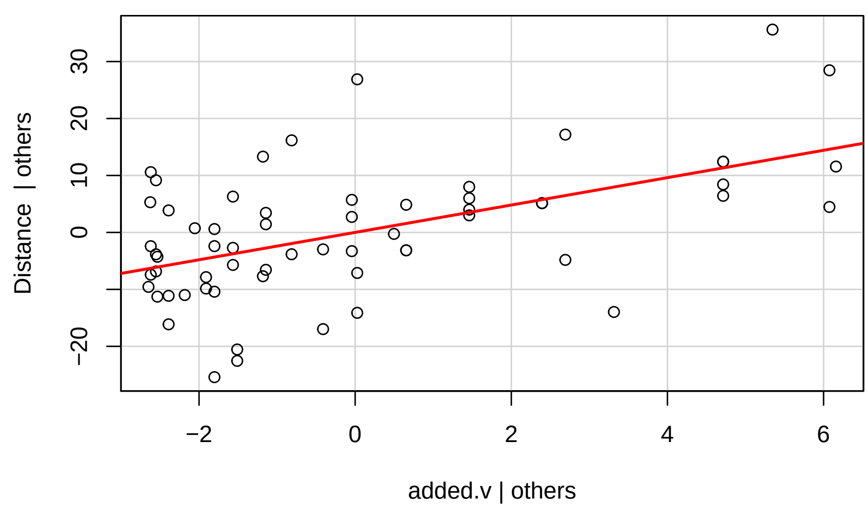
6.4.2

```
added.v <- Speed * log(Speed)
lm.1 <- lm(Distance ~ Speed + added.v)
summary(lm.1)
```

```
##
## Call:
## lm(formula = Distance ~ Speed + added.v)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21.9323  -4.8918  -0.6625   4.4106  26.8194
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  19.2820     8.5872   2.245   0.0285 *
## Speed        -6.3430     1.9615  -3.234   0.0020 **
## added.v       2.4030     0.4959   4.846 9.51e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.04 on 59 degrees of freedom
## Multiple R-squared:  0.9125, Adjusted R-squared:  0.9096
## F-statistic: 307.7 on 2 and 59 DF,  p-value: < 2.2e-16
```

$X \ln X$ 的系数 η 的 t 检验显著，有理由认为变换 X 是必要的,，可按照公式 (6.18)：$\hat{\alpha} = \frac{\hat{\eta}}{\hat{\beta}_1} + 1$ 估计 α。

```
avPlots(lm.1, terms = "added.v") # 附加变量图
```

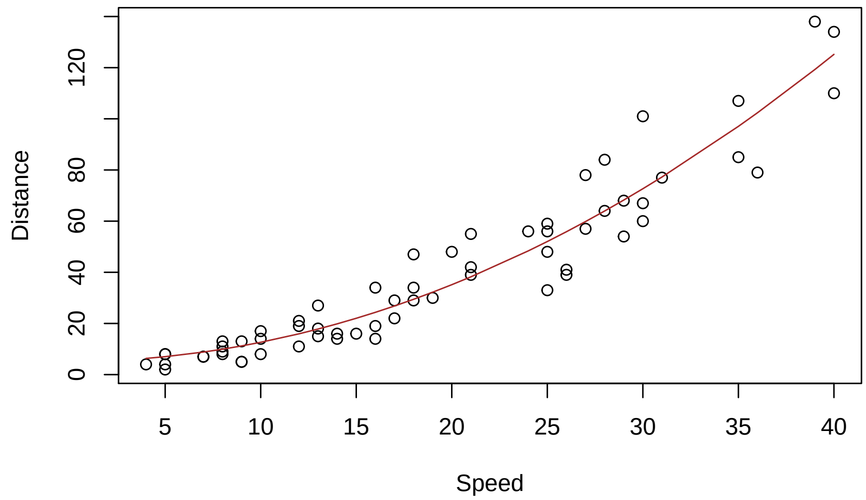



```
alpha <- lm.1$coef["added.v"] / lm.0$coef["Speed"] + 1
alpha
```

```
##  added.v
## 1.764879
```

α 接近 2，表明我们应对 X 做平方变换。

```
Speed.squared <- Speed ^ 2
lm.2 <- lm(Distance ~ Speed.squared)
plot(Distance ~ Speed)
points(Speed, lm.2$fitted.values, type = 'l',col="brown")
```



通过与 6.4.1 的比较，这一模型（$\mathrm{E}[Y|X] = \beta_0 + \beta_1 X^2$）的残差更小，而且直观来看也对数据点拟合得更好。

6.4.3 Atkinson 得分方法变换 Y

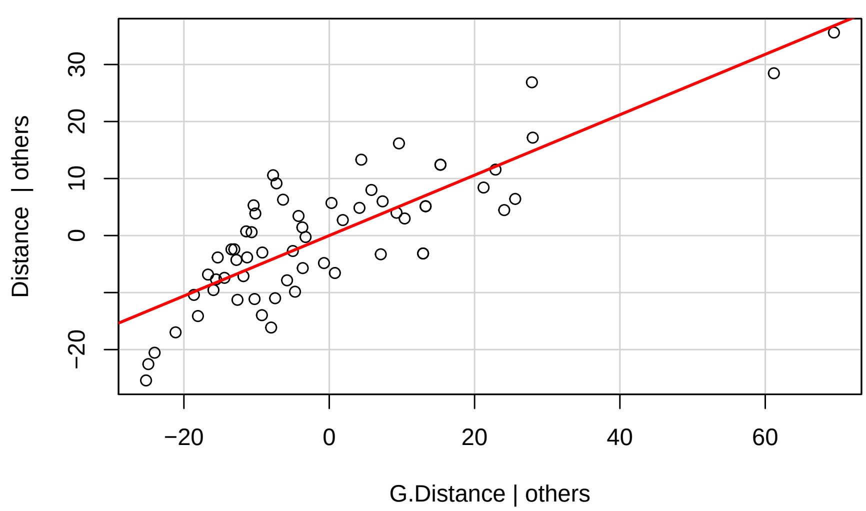
```
GM <- function(y) exp(mean(log(y)))
G <- function(y) {
    GM.y <- GM(y)
    y * (log(y / GM.y) - 1) + log(GM.y) + 1
}
G.Distance <- G(Distance)
lm.atk <- lm(Distance ~ Speed + G.Distance)
summary(lm.atk)
```

```
##
## Call:
## lm(formula = Distance ~ Speed + G.Distance)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -12.0543  -4.7286  -0.3004   4.3153  14.6747
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.70130    2.51889  -0.278    0.782
## Speed        2.27132    0.11562  19.644  < 2e-16 ***
## G.Distance   0.52975    0.04703  11.264 2.47e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.686 on 59 degrees of freedom
## Multiple R-squared:  0.9612, Adjusted R-squared:  0.9599
## F-statistic: 730.4 on 2 and 59 DF,  p-value: < 2.2e-16
```

G 的系数 ϕ 是显著的。$\hat{\lambda} = 1 - \tilde{\phi}$

```
avPlots(lm.atk,terms = "G.Distance")#Atkinson 得分方法的附加变量图
```

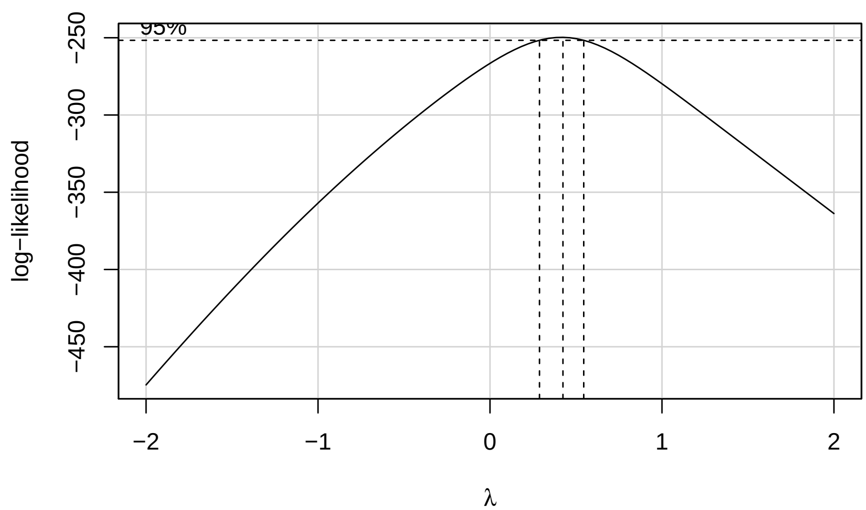


```
lambda <- 1 - lm.atk$coef["G.Distance"]
lambda
```

```
## G.Distance
##  0.4702549
```

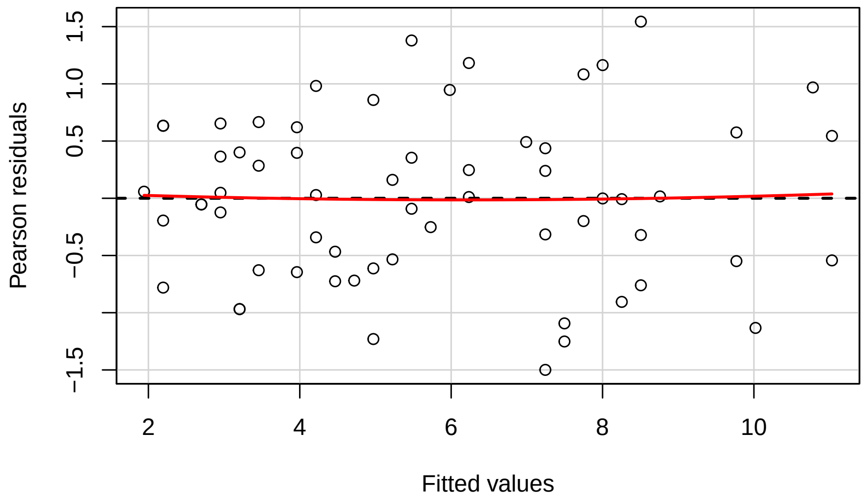
$\lambda \approx 0.5$，应对 Y 做根号变换。下面用 Box-Cox 的似然方法，求估计的变换

```
boxCox(lm.0)
```



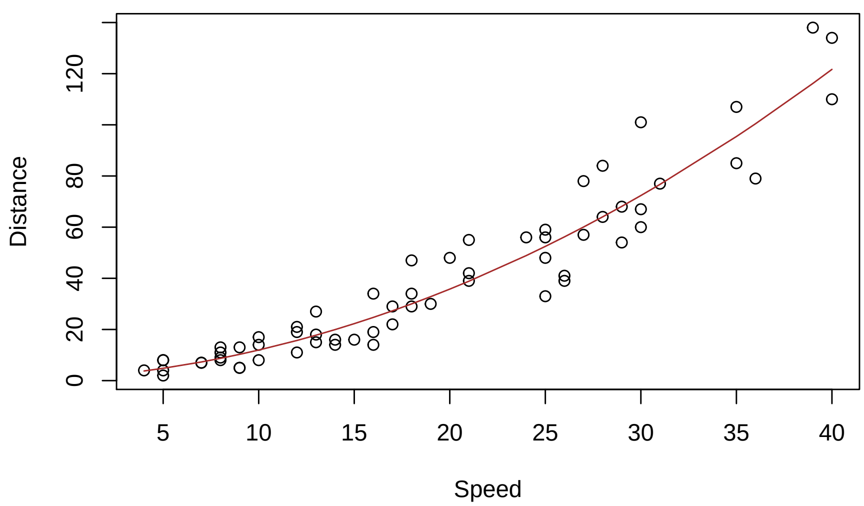
$L(\lambda)$ 是单峰的，最大值及 95% 置信区间都在 0.5 附近，同样暗示我们对 Y 做根号变换，与 Atkinson 得分方法是一致的。

```
Distance.sqrt <- sqrt(Distance)
lm.sqrty <- lm(Distance.sqrt ~ Speed)
residualPlot(lm.sqrty)
```



可以看到，对响应变量 Y 变换后残差已不再有明显的 Pattern。

```
plot(Distance ~ Speed)
points(Speed, lm.sqrty$fitted.values ^ 2, type = 'l', col = "brown")
```



拟合的曲线与 6.4.2 中的拟合曲线很相似。

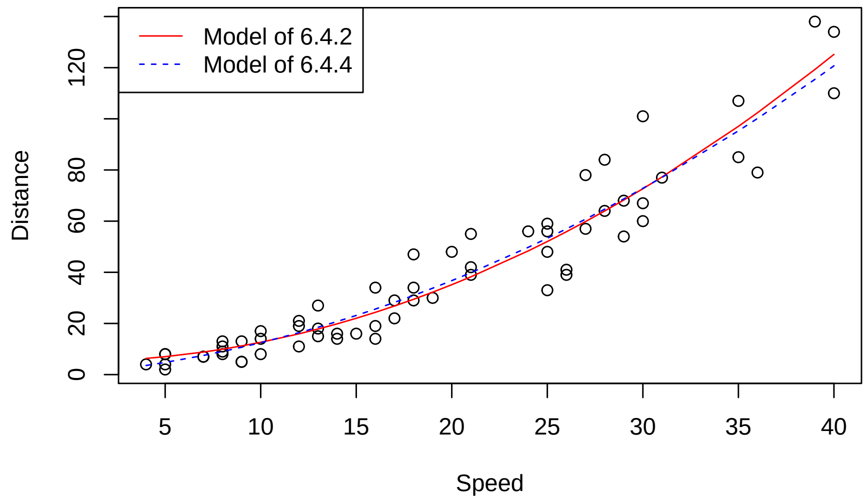
6.4.4

在 6.4.2 中的模型为 $Y=\beta_0+\beta_1x^2+e,\quad \mathrm{Var}(e)=\sigma^2$，拟合结果已储存在 lm.2 中。下面拟合 6.4.4 的模型：$Y=\beta_1x+\beta_2x^2+e,\quad \mathrm{Var}(e)=\sigma^2x^2$

```
lm.Hald <- lm(Distance ~ Speed + Speed.squared - 1,weights = 1/Speed.squared)
summary(lm.Hald)
```

```
##
## Call:
## lm(formula = Distance ~ Speed + Speed.squared - 1, weights = 1/Speed.squared)
##
## Weighted Residuals:
##      Min       1Q   Median       3Q      Max
## -0.81244 -0.35957 -0.03149  0.31400  0.93905
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## Speed         0.656539   0.123447   5.318 1.63e-06 ***
## Speed.squared 0.059036   0.005785  10.205 9.84e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4497 on 60 degrees of freedom
## Multiple R-squared:  0.9468, Adjusted R-squared:  0.9451
## F-statistic: 534.3 on 2 and 60 DF,  p-value: < 2.2e-16
```

```
plot(Distance ~ Speed)
points(Speed, lm.2$fitted.values, type = 'l',lty=1, col = "red")
points(Speed, lm.Hald$fitted.values, type = 'l', lty = 2, col = "blue")
legend('topleft', legend = c('Model of 6.4.2', 'Model of 6.4.4'), col = c('red', 'blue'
```



可以看到，X 在 0 到 40 之间时，两个模型对 Y 的拟合值是很接近的。比如，我们可以考察它们最大的拟合值之差：

```
max(abs(lm.Hald$fitted.values - lm.2$fitted.values))
```

```
## [1] 4.473463
```

对于 6.4.2 中的**预测值**，我们可以用公式

$$\mathrm{Var}(\hat{y}|\mathbf{x}_*) = \sigma^2 \mathbf{x}_*'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_*$$

（求它的估计只需把 σ^2 换成 $\hat{\sigma}^2$）。对 $p' = p + 1 = 2$ 可进一步化简：

$$\mathrm{Var}(\hat{y}|x_*) = \sigma^2 \left(\frac{1}{n} + \frac{(x_* - \bar{x})^2}{\mathrm{SXX}} \right), \quad \mathbf{X} = (\mathbf{1}, x)$$

则模型 6.4.2 的**预测值** (fitted values) 的标准误的为

$$\mathrm{sefit}(\hat{y}|x_*) = \hat{\sigma} \sqrt{\frac{1}{n} + \frac{(x_* - \bar{x})^2}{\mathrm{SXX}}}$$

预测 (prediction) 的标准误为

$$\mathrm{sepred}(\tilde{y}_*|x_*) = \hat{\sigma} \sqrt{1 + \frac{1}{n} + \frac{(x_* - \bar{x})^2}{\mathrm{SXX}}}$$

可用如下程序计算 6.4.2 模型在任意点 x_* 的**预测值**的标准误以及**预测**的标准误。

```
se.lm.2 <- function(x_new) {
    sefit <- sigma(lm.2) * sqrt(1 / length(Speed) +
                  (x_new - mean(Speed)) ^ 2 / sum((Speed - mean(Speed)) ^ 2))
    sepred <- sqrt(sefit ^ 2 + sigma(lm.2) ^ 2)
    c(sefit,sepred)
}
```

模型 6.4.4 是加权最小二乘，**预测值**的方差为

$$\mathrm{Var}(\hat{y}|\mathbf{x}_*) = \sigma^2\mathbf{x}_*'(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{x}_*, \quad \mathbf{X} = (x, x^2), \quad \mathbf{W}_{ij} = \delta_{ij}\frac{1}{x_i^2}$$

预测值的标准误：

$$\mathrm{sefit}(\hat{y}|\mathbf{x}_*) = \hat{\sigma}\sqrt{\mathbf{x}_*'(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{x}_*}$$

预测的标准误：

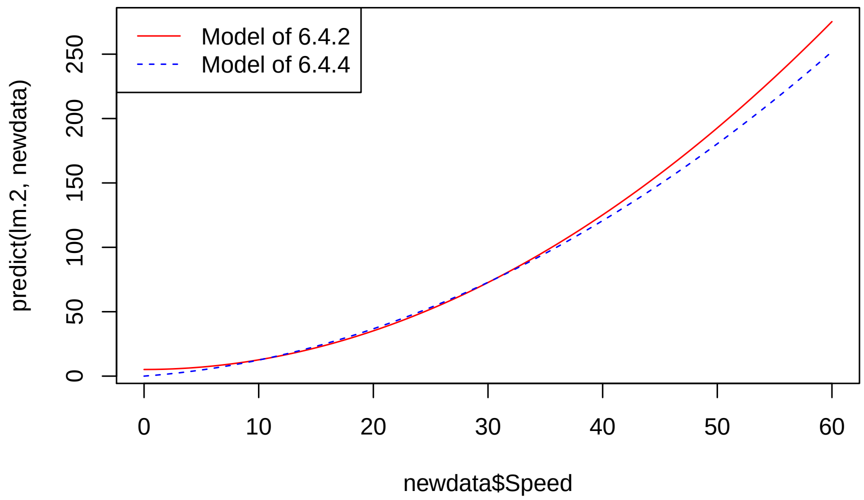
$$\mathrm{sepred}(\tilde{y}_*|\mathbf{x}_*) = \hat{\sigma}\sqrt{\mathbf{x}_*'(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{x}_*}$$

可用如下程序计算 6.4.4 模型在任意点 $\mathbf{x}_*' = (x_*, x_*^2)$ 的**预测值**的标准误以及**预测**的标准误。

```
se.lm.Hald <- function(Speed_new) {
    x_new <- c(Speed_new, Speed_new ^ 2)
    sefit <- sqrt(c(x_new %*% vcov(lm.Hald) %*% x_new))
    sepred <- sqrt(sefit ^ 2 + sigma(lm.Hald) ^ 2 * x_new[2])
    c(sefit,sepred)
}
```

我们试图将模型外推到 $X = 60$mph

```
newdata <- data.frame(Speed = seq(0, 60, 0.5), Speed.squared = (seq(0, 60, 0.5)) ^ 2)
plot(newdata$Speed, predict(lm.2,newdata), type = 'l', lty = 1, col = "red")
points(newdata$Speed, predict(lm.Hald, newdata), type = 'l', lty = 2, col = "blue")
legend('topleft', legend = c('Model of 6.4.2', 'Model of 6.4.4'),
       col = c('red', 'blue'), lty = c(1, 2))
```




```
diff.of.pred <- predict(lm.2, newdata) - predict(lm.Hald, newdata)
diff.of.pred[length(diff.of.pred)]
```

```
##      121
## 23.34299
```

从图中可以看到，在 $X > 40$ 的区域，两个模型的预测差别逐渐增大。在 $X = 60\text{mph}$ 处，两个模型对 Y 的预测值相差 23.34 英尺。然而我们若分别计算两个模型下在 $X = 60$ 处 Y 的**预测值**和**预测**的标准误：

```
se.lm.2(60)# 模型 6.4.2 下 X=60 的预测值和预测的标准误
```

```
## [1]  5.375664 11.256923
```

```
se.lm.Hald(60)# 模型 6.4.4 下 X=60 的预测值和预测的标准误
```

```
## [1] 14.66436 30.70794
```

可以看到，此时的预测值之差甚至已经大于两者的预测值标准误之和，两个模型在外推区域会给出明显不同的结论。

6.4.5

```r
lm.Hald.cons <- lm(Distance ~ Speed + Speed.squared - 1) # 拟合常数方差的 Hald 模型
```

NH: $Y = \beta_1 x + \beta_2 x^2 + e, \quad \mathrm{Var}(e) = \sigma^2$

AH: $Y = \beta_1 x + \beta_2 x^2 + e, \quad \mathrm{Var}(e_i) = \sigma^2 \exp(\lambda^T \mathbf{z}_i)$，其中 $\mathbf{z}_i' = (x_i, x_i^2)$

按照教材 141 页的步骤，编写如下程序进行得分检验：

```r
e <- lm.Hald.cons$residuals # 计算模型中 Y 关于所有 X 的回归，保存残差
u <- e ^ 2 / mean(e ^ 2) # 计算比例平方残差 u
lm.Hald.score <- lm(u ~ Speed + Speed.squared + 1) # 计算 u 关于 z 包含截距的回归
SSreg <- sum((lm.Hald.score$fitted.values - mean(u)) ^ 2) # 计算回归平方和，自由度（不含
S <- SSreg / 2 # 在 NH 下渐进分布为 chisquared(2)
S
```

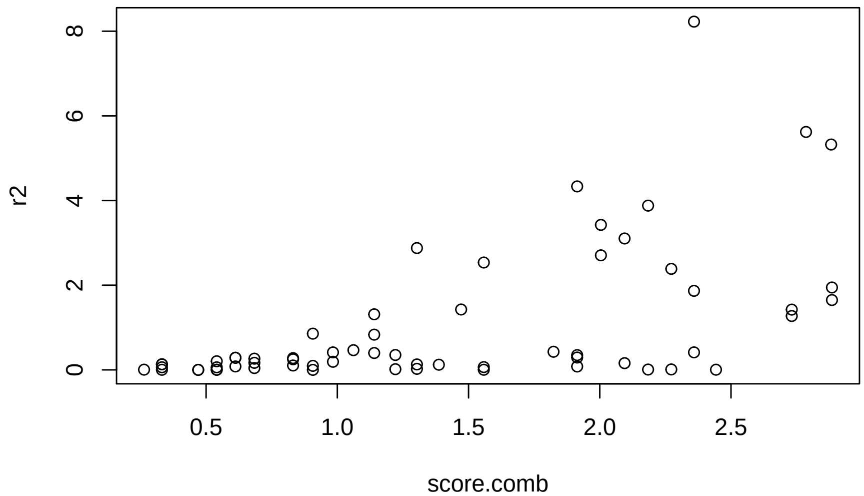
```
## [1] 23.57714
```

```r
pchisq(S, df = 2, lower.tail = FALSE)
```

```
## [1] 7.590831e-06
```

所以我们相当肯定有非常数方差。进一步通过图形考察。作 r_i^2 关于 $(1-h_{ii})\lambda^T \mathbf{z}_i$ 的图。

```r
r2 <- (lm.Hald.cons$residuals / sigma(lm.Hald.cons)) ^ 2 / (1 - hatvalues(lm.Hald.cons)
score.comb <- (1 - hatvalues(lm.Hald.cons)) * (lm.Hald.score$coefficients[2] * Speed
    + lm.Hald.score$coefficients[3] * Speed.squared)
plot(score.comb, r2)
```



可以看到明显的楔形形状，应拒绝 NH。