奧迪中古車資料分析

專題研究

Python 網路爬蟲與迴歸分析

暨 R 語言資料分析實作

目錄

壹、研究摘要 ---------- p.3

貳、研究動機 ---------- p.3

參、研究方法與資料來源 ---------- p.4

肆、基本資料分析----------p.6

伍、模型建立 ---------- p.7

陸、結論 ---------- p.12

柒、程式碼 ---------- p.14

捌、原始資料 ---------- P.17

玖、參考資料 ---------- P.19

壹、 研究摘要

本次研究目的是要瞭解奧迪中古車的定價情況，找到一些直觀且易取得的資訊和汽車折舊間的關聯性，並以迴歸分析建構模型，幫助量化數字間的關係。

最初從中古車網站上選取較有可能有線性關係的兩個變數車齡與里程數建立迴歸，再以 partial F -test、殘差三大假設來對模型修正。最終採用 58 筆官方網站的汽車資訊，並以車齡為自變數，建立最終的迴歸模型：Depreciation Percentage = 0.02906 * Days + 10.19442。

貳、研究動機

汽車乃是生活中人們代步的重要工具之一，而新車與中古車的買賣亦是常常在我們周遭發生的事。今年過年期間，親戚相聚時，我恰巧聽到舅舅說誇著他新買的奧迪，車況多好、價格又多實惠。一旁的叔公則潑了一盆冷水說，這價格買貴了，應該要能買到更新、更好的。見兩個人爭的面紅耳赤、好氣又好笑。此時，我靈光一線，想到自己在學了這麼多程式、分析方法，是不是也能用在這個問題的探討上呢? 沒想到一搜尋奧迪中古車的資訊，便發現其官方網站分別有兩區在介紹新車與中古車，除了價格外，也提供掛牌日期、里程數及排氣量等資訊。為了解答我心中的疑惑，亦給親戚們一把明確的尺衡量中古車的價格，我決定著手研究中古車價格及其他車況相關資訊的關係。

參、研究方法與資料來源

奧迪販售中古車的官方網站提供了汽車的車款、掛牌日期、排氣量、驅動模式、販售地點、售價、耗用能源及里程數等資訊。我推測其中**掛牌日期**與**里程數**會與**折舊量**有較直接的關係，因此我將最後爬取資料的日期（2018/2/27）-掛牌日期，轉換為天數後，視為車齡，並初步選取車齡與里程數，作為回歸模型中的自變數，再以（原價-二手價）/原價來衡量折舊量（應變數）。

X1 = Days **車齡**：計算公式為資料取得日期減去掛牌日期，並轉換為天數。一般來說，汽車開越久，耗損程度越大，車齡為相當直觀的衡量方式。

X2 = Mileage（Km）：行駛量越高通常對汽車的耗損越大。因此，里程數亦為相當直觀的衡量方式。

Y = Depreciation Percentage **折舊率**(%)：計算公式為 100 *（原價-二手價）/原價。這裡折舊計算的依據是**會計理論**中企業對於資產折舊攤提的方式 ，視折舊量為資產可使用價值的耗損量。

註：因報廢價格極低，此處殘值視為零處理。

決定 Days 與 Mileage（Km）兩個自變數為我初步分析的對象後，我將會對資料做基本的分析，運用統計檢定來刪除不適合的變數及其資料，再以最新的資料配適迴歸模型的結果來解釋變數間的關係。

為了分析網站上資料的關係，我必須搜集資料並將其整理成易於分析的表格形式。由奧迪福斯官方網站的資料格式不易複製，也未提供 Excel 檔案，若要手動複製所需資訊後整理成表格，將會耗費大量時間。因此，我決定撰寫爬蟲程式抓取 2018 年 2 月 27 日奧迪福斯官網上的新車與中古車資

訊，並輸出成 CSV 檔（2018/2/27 為建立統計模型當日，無特別意義）。

爬蟲部分，我以 python 撰寫，分別抓取新車與中古車販售網站上所有車款、車齡（中古車網站才有）與價格資訊後，將資料互相配對。最後成功配對 58 筆資料，並將車款、車齡、里程數、二手價與原價資料以表格方式呈現。

至於回歸模型與分析，原本亦以 python 撰寫。但初步建立迴歸模型後，我希望能以更嚴謹的統計方法去驗證與修正之。然而，我上網搜尋以後，發現大多數以 python 實作機器學習-線性回歸的寫手都在建立初步的迴歸模型並畫出迴歸線後止步，而以 R、SAS 或 SPSS 撰寫程式的分析者才會以嚴謹的統計假設去修正迴歸模型。這部分推測是因為上述統計相關的程式語言都有很強大的統計套件，只需要短短幾行程式碼就能完成 Python 好幾行程式碼才能做到的統計驗證。因此我決定改以 R 來建立並修正統計模型。

肆、基本資料分析

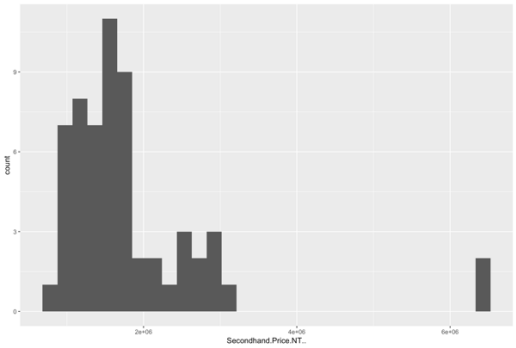
Days	Mileage.km.	Secondhand.Price.NT..	Original.Price.NT..
Min. : 61.0	Min. : 250	Min. : 750000	Min. :1170000
1st Qu.: 275.0	1st Qu.: 2259	1st Qu.:1245000	1st Qu.:1499000
Median : 426.0	Median : 4760	Median :1500000	Median :1990000
Mean : 468.9	Mean : 9198	Mean :1810780	Mean :2406390
3rd Qu.: 518.0	3rd Qu.:12060	3rd Qu.:1930000	3rd Qu.:2600000
Max. :1583.0	Max. :59000	Max. :6400000	Max. :7410000

（表一）

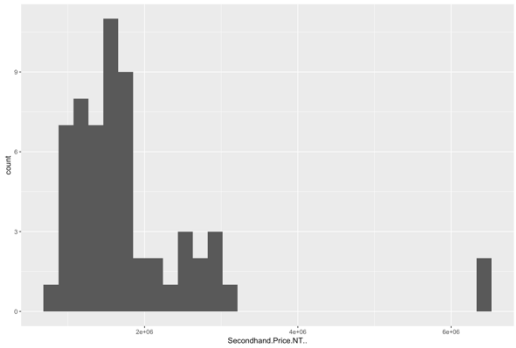
（表一）為 58 種車款的應變數及自變數之統計資料。觀察表中數字，首先，我們可以看到二手價格的各項數據都較原價低，符合預想中二手車會比較便宜的情況。而二手價格的四分位距小於原價，代表二手價格的分佈較為集中，可以推估原價低的車款降價幅度沒有大幅高於原價高的車款。

另外，二手車車齡的第三四分位差為 518 天，也就是說有 75%以上的二手車車齡都在一年半以內。和業界中古車行相比，這樣的車齡可說是相當年輕，推測奧迪的中古車販售網對品質把關較為嚴格。

註：畫出直方圖後，可發現二手車價格（圖一）與原價（圖二）分佈皆為右偏分配，因此用四分位距，而非標準差，來解釋資料分散程度。



（圖一）



（圖二）

伍、模型建立

$Y=\boldsymbol{\beta_0}+\boldsymbol{\beta_1 x_1}+\boldsymbol{\beta_2 x_2}$

註：以下檢定過程，均考慮信心水準為 95%。

（一）做顯著性檢定 F-test 探討模型

一般來說，迴歸模型要越簡單越好，若一變數對於迴歸模型的預測幫助不大，我們便應將其刪除。因此我們以 partial f-test 互相比較模型，找出其中最簡單且預測力夠強者。

虛無假設 A：

H0: 完整模型和簡化模型（車齡為自變數）的殘差平方和 SSE 沒有顯著差異

HA: 完整模型的殘差平方和 SSE 顯著低於簡化模型（車齡為自變數）的殘差平方和 SSE

```
Model 1: Depreciation_percentage ~ Days
Model 2: Depreciation_percentage ~ Days + Mileage.km.
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     57 3146.9
2     56 3094.8  1    52.157 0.9438 0.3355
```

結果：p-value > 0.05，不拒絕虛無假設，推論兩模型的殘差平方和沒有顯著差異。考慮模型的簡單程度，推論只有車齡為自變數的模型較佳。

虛無假設 B:

H0: 完整模型和簡化模型（里程數為自變數）的殘差平方和 SSE 沒有顯著差異

HA: 完整模型的殘差平方和 SSE 顯著低於簡化模型（里程數為自變數）的殘差平方和 SSE

```
Model 1: Depreciation_percentage ~ Mileage.km.
Model 2: Depreciation_percentage ~ Days + Mileage.km.
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1     57 5023.5
2     56 3094.8  1    1928.8 34.901 2.147e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

結果: p-value<0.05，拒絕虛無假設，推論完整模型的殘差平方和顯著低於簡化模型的殘差平方和，不應刪減車齡這個自變數。

模型優劣比較: 簡化模型，刪去自變數 Mileage(Km) > 完整模型 > 簡化模型，刪除自變數 Days

⇨ 刪除自變數 Mileage，建立新的回歸模型: $Y=\boldsymbol{\beta_0}+\boldsymbol{\beta_1 x_1}$

刪除自變數 Mileage(Km)後，形成一個新的簡單回歸模型。而一般來說，建立模型後，我們必須確認其殘差符合三項假設:

1. **常態性**
2. **獨立性**
3. **變異數同值性**

（二）檢驗模型是否符合殘差三大假設

1. 檢驗殘差的常態性

虛無假設：

H_0: 殘差符合常態分配

H_A: 殘差不符合常態分配

```
        Shapiro-Wilk normality test

data:  Rl_Days$residual
W = 0.95887, p-value = 0.04424
```

結果：p-value < 0.05，推論殘差不服從常態分配。一般來說，我們建立模型時會要求其殘差服從常態分配，但在某些情況下，殘差服從常態分配不是必要的。若此處的迴歸模型係屬此類情況，我們才可以繼續使用。

首先，根據高斯-馬可夫定理(Gauss-Markov Theorem)，以最小平方法(Least Squares Method)計算線性迴歸參數 $\boldsymbol{\beta_0}$ 、$\boldsymbol{\beta_1}$ 將有最佳線性不偏估計量(BLUE, Best Linear Unbiased Estimator)性質的前提，要求殘差符合以下條件：

i. 殘差期望值為 0

ii. 殘差具有同質變異，變異數為一固定常數

iii. 殘差間沒有自相關(Autocorrelation)

iv. 自變數與殘差無關，即正交性(Orthogonality)

也就是說，因為此模型是以**最小平方法**建立，**其殘差不必服從常態分配**。

再者，統計上針對殘差假設為常態分配的意義有三:

一、迴歸是需要相對大樣本才較有意義的方法。特別是多元變數的複迴歸，對樣本的需求量很大，很自然會符合中央極限定理。實務上是 300-500 個樣本或更多時才適用。

二、統計推論常見的 Z、T、Chi-squared、F 基本上都是跟常態的機率分布性質(Normal Distribution)有關。光有殘差，但無法對殘差進行推論也是不夠力的。

三、係數檢定用的 T 分配及類 T 統計量都是對偏離常態不太敏感的統計量，因為它們本身就是常態 Z 統計量的近似，因此近似又近似的結果就是，除非是殘差真實分配遠離常態，不然影響非常有限。在稍大的樣本條件下更是如此(理由同第一點)。

因此，此處樣本數不足使殘差不服從常態分配係屬正常。而且若沒有要用到 Z、T、Chi-squared、F 等相關統計推論，殘差服從常態分配也是不必要的。

綜合以上兩點，雖然模型未通過此假設檢定，但經細究殘差需符合常態分配之情況與原因後，**推論此模型殘差雖不服從常態分配，仍可以被有效使用**。

2. 檢驗殘差的獨立性

虛無假設:

H0: 殘差間相互獨立

HA: 殘差間非相互獨立

```
 lag Autocorrelation D-W Statistic p-value
   1       0.1949312       1.60964   0.126
 Alternative hypothesis: rho != 0
```

結果: p-value > 0.05，不拒絕 H0，推論殘差間相互獨立。

3. 檢驗殘差變異數的同質性

虛無假設:

H0: 殘差間變異數具有同質性

HA: 殘差間變異數不具有同質性

```
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 1.152253, Df = 1, p = 0.28308
```

結果: p-value > 0.05，不拒絕 H0，推論殘差變異數具有同質性。

陸、結論

做完顯著性檢定 F-test 刪減自變數，再確認新的模型可通過殘差假設後，我們完成了模型的建立，並推論此線性模型可以被使用。

另外，在建立模型前，我原本以為 Mileage 和車子本身耗損會有較直接的關係，也因此的解釋力會比 Days 強，沒想到最後的結果是只有 Day 為自變數的簡單迴歸為最適模型。推測是 Mileage 和車子耗損間的關係不如我想像中的直接，亦有可能是因為市場上的消費者對於 Day 這樣的資訊較為敏感，才產生了這樣的定價情形。這樣的結果也展現出我們平時的直覺和真實情況的差異，凸顯了資料分析的重要性。

最終模型：Depreciation Percentage = 10.19442 + 0.0296 * Days

```
lm(formula = Depreciation_percentage ~ Days)

Residuals:
    Min      1Q  Median      3Q     Max
-15.283  -4.629  -1.812   4.130  20.185

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 10.19442    1.86868   5.455 1.10e-06 ***
Days         0.02906    0.00341   8.524 9.34e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.43 on 57 degrees of freedom
Multiple R-squared:  0.5604,    Adjusted R-squared:  0.5527
F-statistic: 72.65 on 1 and 57 DF,  p-value: 9.338e-12
```

從模型的**判定係數** 0.5604 我們可以推估**相關係數約為** 0.7486，而一般來說，當相關係數高於 0.7 時，我們可以推論變數的相關程度為**高度相關**，也就是**車齡增加與價格折舊這兩件事的相關性很高**。除了透過相關係數瞭解單一模型變數的相關程度外，未來建立類似迴歸模型時，我們也可以將兩模型相關係數比較，判斷哪個模型對變數間關係的解釋力較強。

模型應用：從 Audi 中古車網站使用者的角度來看，賣方想要販售中古車卻不了解市場行情時，可利用此迴歸模型代入車齡得到折舊量來推估自己目前的開價是否合理，可避免價格太低被揩油，或價格過高，導致汽車難以售出。同理，買方也可以比較不同中古車，推估哪一台最划算，作為購車的參考依據；作為 Audi 官方，建立模型除了可以比較和其他同業的差異外，也可以利用單筆資料的殘差去檢驗是否有價格誤植、系統設定錯誤等問題（類似審計實務中比率分析的應用）。

柒、程式碼

(A) 網路爬蟲

```
import requests
from lxml import etree
import pandas as pd
import datetime
import requests
from lxml import etree
import pdfkit

href1_list =[]
Audilist = []
pricelist = []
Audipricelist = []

def appendlist( hrefx ): # 資料清洗，把型號跟價格分別裝成list
    if hrefx != []:
            for i in range( len( hrefx ) ):
                if 'Audi' in hrefx[i] and hrefx[i][0] != 'A': # 有些車款資料，前面有多5個字元，有些前面多6個
                    Audilist.append( hrefx[i][6:] )
                if 'Audi' in hrefx[i] and hrefx[i][0] == 'A': # 多5個字元
                    Audilist.append( hrefx[i][5:] )
                if 'NT' in hrefx[i]: # 價格
                    pricelist.append( hrefx[i][3:] )

headers = { "User-Agent" : "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.0 Safari/605.1.15" }
# 用自身瀏覽器的網頁代理，假裝不是爬蟲
response = requests.get( 'http://www.audi.com.tw/tw/web/zh.html?ds_rl=1256997&ds_rl=1257015&ds_rl=1253959&ds_rl=1253959&ds_rl=1257015', headers = headers )
# 取得網址，reponse為200代表取得成功
content = response.content.decode() # 單純印出content為unicode的形式，因此用decode解碼，()內沒輸入時預設為utf-8
html = etree.HTML( content ) # 使用 xpath 選擇器來擷取數據
for i in range(1,14,1): # 不同車款
    for j in range(1,6,1): # 不同型號
        href = html.xpath('/html/body/div[1]/div[1]/div[2]/div[2]/div/div[1]/ul/li/a/@href') # 取得各類車型的網址
for i in range( len( href) - 1 ):
        response1 = requests.get( 'http://www.audi.com.tw' + href[i][0:-5] + '/price.html', headers = headers ) # 建議售價的網址
        content1 = response1.content.decode() # 解碼
        html1 = etree.HTML( content1 ) # 4種xpath的呈現方式
        href1 = html1.xpath( '/html/body/div[1]/div[2]/div/div[7]/div[2]/div/div[2]/table/tbody/tr//text()')
        href2 = html1.xpath( '/html/body/div[1]/div[2]/div/div[7]/div[2]/div[2]/table/tbody/tr//text()')
        href3 = html1.xpath( '/html/body/div[1]/div[2]/div/div[6]/div[2]/div[2]/table/tbody/tr//text()' )
        href4 = html1.xpath( '/html/body/div[1]/div[2]/div/div[7]/div[2]/div/div[3]/table/tbody/tr//text()')
        appendlist( href1 )
        appendlist( href2 )
        appendlist( href3 )
        appendlist( href4 )
for i in range( len(Audilist) ):
    Audilist[i] = Audilist[i].replace(u'\xa0', u' ') # 清洗資料中無用的資訊
    Audipricelist.append( (Audilist[i].strip(), pricelist[i].strip()) ) # 新車網站所有車款的型號與原價
```

```
href_list =[]
information_list =[]
headers = { "User-Agent" : "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.0 Safari/605.1.15" }
# 用自身瀏覽器的網頁代理，假裝不是爬蟲
for i in range( 1, 12, 1 ): # 1到11頁
    response = requests.get( 'http://approvedplus.audi.com.tw/?lm=r&p='+str(i), headers = headers ) # 取得網址，reponse為200代表取得成功
    content = response.content.decode() # 單純印出content為unicode的形式，因此用decode解碼，()內沒輸入時預設為utf-8
    html = etree.HTML( content ) # 使用 xpath 選擇器來擷取數據
    href = html.xpath('//*[@id="srch-results"]/div/div[1]/div/div/div[2]/div/h3/a/@href') # 辨認每台車的網址
    for j in range(12):
        try: # 每頁至多12筆資料，用try才不會 runtime error
            if href[j] not in href_list:
                href_list.append( href[j] )
        except:
            pass
# 目前已取得126台中古車的網址（此數字會不斷變動）

for i in href_list: # 透過中古車型號，配對車相同車款的原價
    response = requests.get( 'http://approvedplus.audi.com.tw'+str(i), headers = headers ) # 取得網址，reponse為200代表取得成功
    content = response.content.decode() # 單純印出content為unicode的形式，因此用decode解碼，()內沒輸入時預設為utf-8
    html = etree.HTML( content ) # 使用 xpath 選擇器來擷取數據
    model = html.xpath('/html/body/div[1]/div[2]/section[2]/div/h1/text()') # 型號
    original_price = 'None' # 初始設定
    for j in range( len(Audipricelist) ):
        if model[0] == Audipricelist[j][0]: # 若車款相同
            original_price = Audipricelist[j][1].strip() # 找到原價
    years = html.xpath('/html/body/div[1]/div[2]/section[2]/div/div[2]/div[1]/ul/li[1]/span[2]/text()') # 年份
    year = years[0][0:4] # 年份
    month = years[0][5:] # 月份
    date = datetime.datetime(int(year),int(month),1,0,0,0) # 日期
    today = datetime.datetime(2019,1,1,0,0,0) # 現在的年份與月份
    days = today - date # 已掛牌幾天
    days = days.days # 轉換格式
    secondhand_price = html.xpath('/html/body/div[1]/div[2]/section[2]/div/div[2]/div[1]/ul/li[2]/span[2]/text()') # 價格
    secondhand_price[0] = float( secondhand_price[0].strip()[0:-2] ) * 10000 # 網站以中文字萬元計價，在此將其轉換為數字
    mileage = html.xpath('/html/body/div[1]/div[2]/section[2]/div/div[2]/div[1]/ul/li[10]/span[2]/text()') # 里程
    if mileage[0].strip() == '<500': # 若里程數低於500，不會顯示實際里程數
        mileage[0] = '250' # 假設為250
    if original_price != "None": # 有配對到資料才能跑迴歸（新車與中古車網站上有部分車款資訊詳細程度不一，無法確認為相同車款，因此不予配對）
        information = [model[0],days, mileage[0].strip(), secondhand_price[0], original_price.strip()] # 標題依序為 車款、車齡、里程數、二手價、原價
        information_list.append(information)

column_name = ['Model', 'Days', 'Mileage(km)', 'Secondhand Price(NT$)', 'Original Price(NT$)']
test = pd.DataFrame(columns = column_name, data = information_list) # 轉換為矩陣形式
test.to_csv('/Users/lijicheng/Downloads/履歷表/奧迪福斯資料分析/result.csv', encoding = 'utf_8_sig') # 存成csv檔，輸出結果會有千分位，要用excel解決
```

(B) 以 Python 初步建立迴歸模型（機器學習之監督式學習）

```python
import csv
import matplotlib.pyplot as plt
import numpy as np # 科學計算庫

fn1 = '/Users/lijicheng/Downloads/履歷表/奧迪福斯資料分析/result.csv' # 檔案名稱
fh1 = open(fn1, 'r', newline = '', encoding = 'utf-8') # 開啟檔案
csv1 = csv.DictReader(fh1) # 以dictionary的形式讀取檔案
cname1 = csv1.fieldnames # 標題
dayslist = [] # 車齡
depreciationlist = [] # 折舊百分比
mileagelist = [] # 里程數

def model( Xvariable, 自變數 )
    X = Xvariable # 自變數
    Y = depreciationlist # 應變數
    Xsum = 0 # 設立起點值
    X2sum = 0
    Ysum = 0
    XY = 0
    n = len( X ) # 資料比數

    for i in range( n ): # 自變數為車齡的簡單迴歸
        Xsum += X[i] # 總天數
        Ysum += Y[i] # 總折舊百分比
        XY += X[i] * Y[i] # 相乘
        X2sum += X[i] ** 2 # 平方和
    a = ( Xsum * Ysum / n - XY ) / ( Xsum ** 2 / n - X2sum )
    b = ( Ysum - a * Xsum ) / n
    print( 自變數 + '與折舊的關係=%f*x+%f' % (a,b) )
    SSR = 0
    SSE = 0
    for i in range( n ):
        SSR += ( ( a * X[i] + b ) - ( Ysum / n ) ) ** 2 # 估計值 - 平均值 的平方
        SSE += ( Y[i] - ( a * X[i] + b ) ) ** 2 # 實際值 - 平均值 的平方
    SST = SSR + SSE # 總評方和 = 可被解釋平方和 + 殘差平方和
    Rsquare = SSR / SST # 判定係數
    print( "判定係數為" + str( Rsquare ) )

for aline in csv1:
    dayslist.append( int( aline[cname1[2]] ) ) # 車齡
    depreciationpercentage = 100 * ( int( aline[cname1[5]] ) - int( aline[cname1[4]] ) ) / int( aline[cname1[5]] ) # 折舊百分比
    depreciationlist.append( depreciationpercentage )
    mileagelist.append( int( aline[ cname1[3] ] ) ) # 里程數

model( dayslist, '車齡')
model( mileagelist, '里程數')
```

(C) 以 R 驗證並調整迴歸模型

```r
library(utils) # 為了讀csv
Audi_data = read.csv('/Users/lijicheng/Downloads/履歷表/奧迪福斯資料分析/result.csv')
Audi_data_summary = summary(Audi_data) # 基本資料
library(magrittr) # 才能用 %>%
library(dplyr) # 才能用mutate()

attach(Audi_data) # 為了讓系統讀得到Audi_data中的每一行標題
Audi_data <- Audi_data %>%
  mutate( Depreciation_percentage = 100 * (Original.Price.NT.. - Secondhand.Price.NT..)/Original.Price.NT..
# 新增折舊率
Rl_2v = lm( Depreciation_percentage ~  Days + Mileage.km. ) # 2個自變數的複回歸
Rl_Days = lm( Depreciation_percentage ~  Days ) # 自變數為天數的簡單迴歸
Rl_Mileage = lm( Depreciation_percentage ~  Mileage.km. ) # 自變數為里程數的簡單迴歸
anova(Rl_Days,Rl_2v) # p-value > 0.05，不能拒絕虛無假設，只有車齡為自變數的模型較好
anova(Rl_Mileage,Rl_2v) # p-value < 0.05，拒絕虛無假設，full model較只有里程數為自變數的模型好
# 模型優劣比較： 車齡為自變數 > full model > 里程數為自變數
Rl_Days_summary = summary(Rl_Days) # p-value < 0.05，拒絕虛無假設，迴歸係數顯著
# 註：簡單迴歸中，f-test和t-test意義相同，因此不用再做t-test

library(ggplot2) # 載入畫圖套件
ggplot(Audi_data, aes(Days,Depreciation_percentage))+ # x軸為車齡，y軸為折舊比率
   geom_smooth(method="lm") + # 畫出回歸線
   geom_line()+ # 點與點之間以直線連接
   geom_point() # 標出各點

#畫出回歸線後，可看到有些許殘差較大的離群值。
#但除了車齡以外，還有其他與車況相關之因素會影響折舊率，因此判斷這些離群值係屬正常，不予以刪除

# 建立迴歸模型時，必須確認其殘差(residual)符合三個假設： 1.常態性假設 2.獨立性假設 3.同質性假設
shapiro.test(Rl_Days$residual) # 檢驗常態性假設
# 虛無假設H0:殘差服從常態分配，因為p-value < 0.05，拒絕H0，推論殘差不服從常態分配
# 殘差的常態假設並非必要，通常是較大的樣本（300~500）才會服從，一般來說以最小平方法建立回歸模型時無須符合
# 若要做進一步檢定才必須符合
require(car) # 載入套件（為了檢驗獨立性與同質性）
durbinWatsonTest(Rl_Days) # 檢驗獨立性假設
# 虛無假設H0:殘差間相互獨立，因為p-value > 0.05，不拒絕H0，推論殘差間相互獨立
ncvTest(Rl_Days)
# 虛無假設H0:殘差變異數具有同質性，因為p-value > 0.05，不拒絕H0，推論殘差變異數具有同質性
# 三項假設均符合，迴歸模型可以使用
```

捌、原始資料

	Model	Days	Mileage(km)	Secondhand Price(NT$)	Original Price(NT$)
0	A5 Coup_ 40 TFSI Sport	275	2500	1880000	2270000
1	A6 Sedan 35 TDI	640	15950	1480000	2490000
2	A1 Sportback 25 TFSI	153	2000	980000	1170000
3	S5 Sportback	975	27800	1980000	3510000
4	Q3 30 TFSI	334	6486	1330000	1710000
5	A6 Avant 35 TFSI	1188	59000	1498000	2660000
6	TT Coup_ 45 TFSI quattro	1096	21630	1588000	2500000
7	Q7 40 TFSI quattro Luxury	275	2058	3010000	3100000
8	S5 Coup_	365	7350	2880000	3470000
9	A7 Sportback 40 TFSI quattro	275	250	2680000	3410000
10	Q7 40 TFSI quattro Luxury	1005	14700	2210000	3100000
11	A1 Sportback 25 TFSI	396	11519	890000	1170000
12	A3 Sportback 35 TFSI Premium	396	11273	1190000	1499000
13	A1 Sportback 25 TFSI	275	4900	950000	1170000
14	A1 Sportback 25 TFSI	275	3900	950000	1170000
15	Q2 35 TFSI Luxury	518	12900	1160000	1430000
16	Q3 35 TFSI quattro	275	2058	1710000	1990000
17	A8 50 TFSI quattro	518	9700	2390000	4380000
18	Q2 35 TFSI Sport	518	12600	1280000	1540000
19	A8 50 TFSI quattro	518	3850	2490000	4380000
20	A6 Sedan 35 TFSI	457	2058	1710000	2540000
21	A6 Sedan 35 TDI	457	2058	1670000	2490000
22	A6 Sedan 35 TFSI	640	5600	1560000	2540000
23	Q3 35 TFSI quattro	579	7100	1570000	1990000
24	A8 50 TDI quattro	457	1550	2490000	3890000

25	A6 Sedan 35 TFSI	852	12950	1480000	2540000
26	A6 Sedan 35 TDI	457	2058	1670000	2490000
27	Q3 30 TFSI	275	2058	1430000	1710000
28	A1 Sportback 25 TFSI	275	2068	980000	1170000
29	A1 Sportback 25 TFSI	275	2068	1000000	1170000
30	RS 6 Avant performance	275	2068	6400000	7410000
31	RS 6 Avant performance	275	2058	6400000	7410000
32	Q3 35 TFSI quattro	852	8000	1470000	1990000
33	Q2 35 TFSI Sport	518	3650	1230000	1540000
34	Q7 40 TFSI quattro Luxury	275	1500	2920000	3100000
35	A3 Sedan 35 TFSI Premium	61	4220	1360000	1560000
36	Q3 30 TFSI	426	11356	1320000	1710000
37	Q5 35 TDI quattro	61	4530	2150000	2320000
38	A3 Sedan 35 TFSI Premium	396	12803	1190000	1560000
39	A1 Sportback 30 TFSI	1583	54924	750000	1420000
40	A3 Sportback 35 TFSI Premium	365	6441	1280000	1499000
41	A3 Sportback 35 TFSI Premium	184	2876	1380000	1499000
42	Q7 40 TFSI quattro Luxury	426	9600	2480000	3100000
43	Q2 35 TFSI Luxury	153	2008	1260000	1430000
44	A6 Sedan 35 TDI	457	2500	1780000	2490000
45	A6 Sedan 35 TFSI	671	19074	1500000	2540000
46	A6 Sedan 35 TFSI	791	23907	1500000	2540000
47	Q2 35 TFSI Luxury	487	21750	1140000	1430000
48	Q2 35 TFSI Luxury	487	16750	1160000	1430000
49	Q7 45 TFSI quattro	852	7000	2650000	3880000
50	Q7 45 TFSI quattro	334	4760	3200000	3880000
51	Q3 35 TFSI quattro	457	7435	1480000	1990000
52	Q3 35 TFSI quattro	396	4210	1650000	1990000

53	Q3 35 TFSI quattro	334	4730	1680000	1990000
54	Q3 35 TFSI quattro	275	3160	1730000	1990000
55	Q2 35 TFSI Luxury	487	20403	1140000	1430000
56	A6 Sedan 35 TFSI	457	2450	1780000	2540000
57	A6 Sedan 35 TDI	184	3290	1780000	2490000
58	A1 Sportback 25 TFSI	153	3210	990000	1170000

玖、參考資料

奧迪官方汽車販售網站:

http://www.audi.com.tw/tw/web/zh.html?ds_rl=1256997&ds_rl=1257015&ds_rl=1253959&ds_rl=1253959&ds_rl=1257015

奧迪官方中古車販售網站:

http://approvedplus.audi.com.tw

註: 網站資料會隨時間變動，不會和此份研究所採之統計資料完全相同