Big Data Analysis HW1

2020270026 王姿文

4/11/2021

1 OLS

1.1 OLS结果

首先创建data，下表列出前几笔data的数据：

```
#create data
set.seed(2020270026)
n <- 1000
x1 <- rnorm(n,0,3)
x2 <- rbinom(n, size=1, prob=0.47)  #0,1
x3 <- runif(n, 18, 60)
a <- 2
b1 <- 2
b2 <- 0.4
b3 <- 0.02
e <- runif(n,-1,1)
y <- a + b1*x1 + b2*x2 + b3*x3 + e
dat <- cbind.data.frame(y, x1, x2, x3)
kbl(head(dat)) %>%
  kable_styling(bootstrap_options = c("striped", "hover"), full_width = F, font_si
ze = 12)
```

y	x1	x2	x3
-4.7857978	-3.600183	0	52.93018
10.3307274	3.909391	0	27.96269
8.2063853	3.008760	1	31.09358
6.2430445	1.129340	1	34.38349
0.4290972	-1.801032	1	44.52318
8.7295701	2.758932	1	44.76994

接着依照下列公式求出$\mathbf{X}$的估计参数β和预测值$\hat{\mathbf{y}}$：

$$\beta = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

$$\hat{\mathbf{y}} = \mathbf{X}\beta$$

$$\epsilon = \mathbf{y} - \hat{\mathbf{y}}$$

$$V(\hat{\beta}) = \frac{\epsilon' \epsilon}{n-k}(\mathbf{X}'\mathbf{X})^{-1}$$

得到β：

```
#ols
x <- data.matrix(dat[2:4])
y_m <- data.matrix(dat[1])
xx_inv <- solve(t(x) %*% x)

#beta
B <- xx_inv %*% t(x) %*% y_m
kbl(B) %>%
  kable_styling(bootstrap_options = c("striped", "hover"), full_width = F, font_si
ze = 12)
```

	y
x1	1.9971965
x2	0.7564111
x3	0.0630761

下表列出前几笔数据，分别为原始数据y，预测值$\hat{y}$，和残差ϵ：

```
y_pre <- x %*% B
eps <- y_pre-y_m
result1_1 <- cbind.data.frame(y, y_pre, eps,x1)
colnames(result1_1) <- c("y", "y_pre", "residual", 'x1')

kbl(head(result1_1[,1:3])) %>%
  kable_styling(bootstrap_options = c("striped", "hover"), full_width = F, font_si
ze = 12)
```

y	y_pre	residual
-4.7857978	-3.8516422	0.9341555
10.3307274	9.5715999	-0.7591274
8.2063853	8.7267585	0.5203732
6.2430445	5.1807015	-1.0623429
0.4290972	-0.0322541	-0.4613514
8.7295701	9.0904545	0.3608844

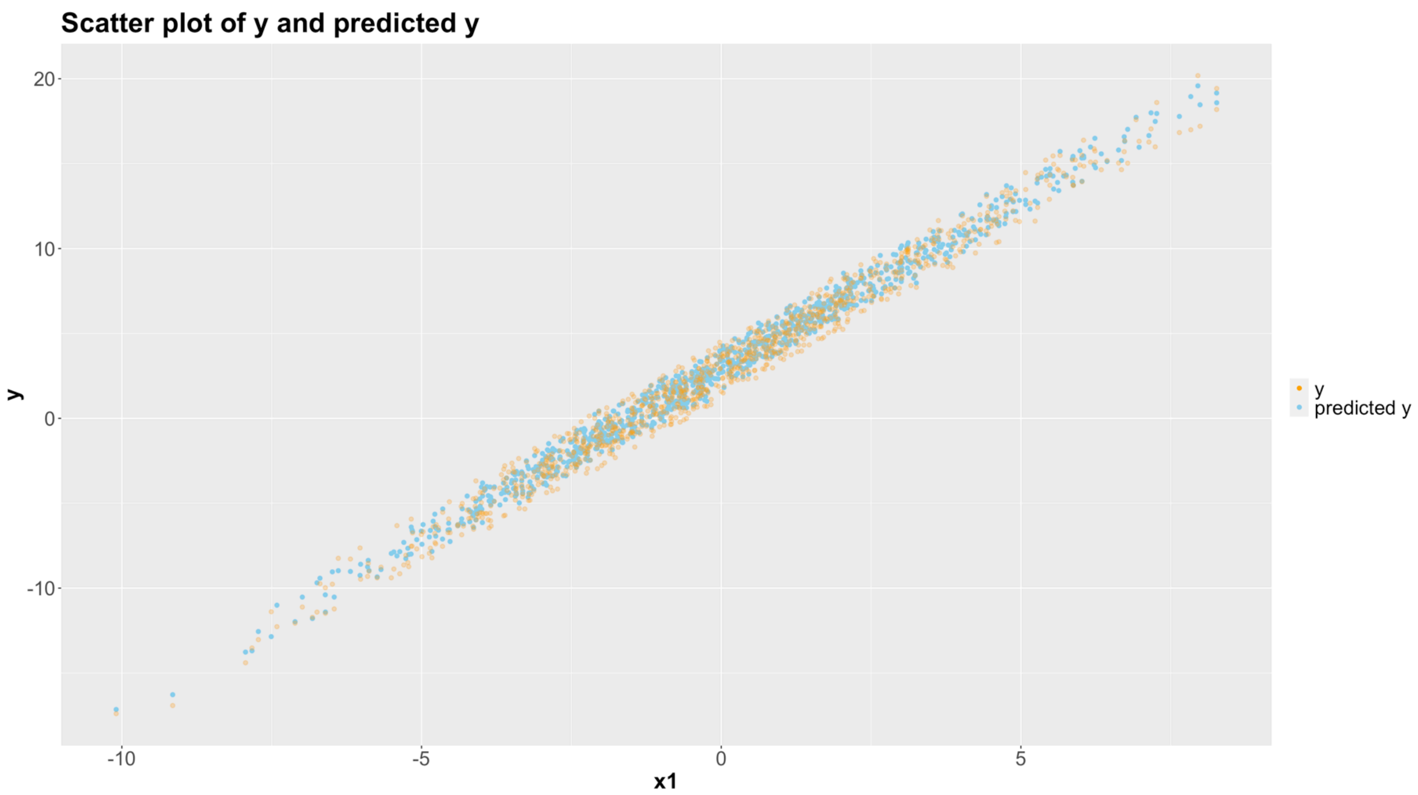
斜方差矩阵为：

```
ee <- t(eps) %*% eps
v <- (ee[1]/n-3)*xx_inv
v
```

```
##                    x1             x2                x3
## x1 -0.0002701770444 -0.00003173506  0.0000007536283
## x2 -0.0000317350642 -0.00846398476  0.0000989761733
## x3  0.0000007536283  0.00009897617 -0.0000026149505
```

下图为OLS配适结果：

```
ggplot(result1_1, aes(x=x1, y=y)) +
  geom_point(aes(color='skyblue'),show.legend = T) +
  geom_point(aes(x=x1, y=y_pre,color = 'orange'), alpha = 0.3,show.legend = T) +
  scale_colour_manual(values = c("orange","skyblue"), labels = c('y','predicted y'
),name = ' ')+
  theme(plot.title = element_text(size=25, face="bold"),
        axis.title = element_text(size=20, face="bold"),
        axis.text = element_text(size=18),
        legend.title=element_blank(),
        legend.text = element_text(size=18)) +
  labs(title =paste('Scatter plot of y and predicted y'))
```



1.2 比较不同sample number的OLS结果

首先编写一个boot()函数，分别重复抽样出两个数据集，以比较OLS估计结果。
下表为β：

```
boot <- function(i){sample_n(dat,i,replace=TRUE)}
dat_50 <- boot(50)
dat_5000 <- boot(5000)

#dat_50
x <- data.matrix(dat_50[2:4])
y_m <- data.matrix(dat_50[1])
xx_inv <- solve(t(x) %*% x)
B_50 <- xx_inv %*% t(x) %*% y_m

#dat_5000
x <- data.matrix(dat_5000[2:4])
y_m <- data.matrix(dat_5000[1])
xx_inv <- solve(t(x) %*% x)
B_5000 <- xx_inv %*% t(x) %*% y_m

#compare
result1_21 <- cbind.data.frame(B_50,B_5000)
colnames(result1_21) <- c("n=50", 'n=5000')
kbl(result1_21) %>%
  kable_styling(bootstrap_options = c("striped", "hover"), full_width = F, font_si
ze = 12)
```

	n=50	n=5000
x1	1.959593	1.9932060
x2	0.933731	0.7530717
x3	0.063100	0.0632635

接著比較兩者β均值會發覺均值不同（因為β），不同的原因在于，OLS使用LSE来估计β，亦即 $min \sum_{i=1}^{n}(\hat{y_i} - y_i)^2$ 来求解。已知两者的n不同，故求出的解也会不同，一般来说，sample数越大估计结果越准确。

```
result1_22 <- cbind.data.frame(mean(B_50),mean(B_5000))
colnames(result1_22) <- c("n=50", 'n=5000')
kbl(result1_22) %>%
  kable_styling(bootstrap_options = c("striped", "hover"), full_width = F, font_si
ze = 12)
```

n=50	n=5000
0.9854748	0.9365137

2 Plot

- Residual Plot：
  左上图，采用$n = 1000$的原始数据来估计，可以看出残差没有特别趋势，且大多分布在$[-2 * sd(\epsilon), 2 * sd(\epsilon)]$，因此估计结果是好的。

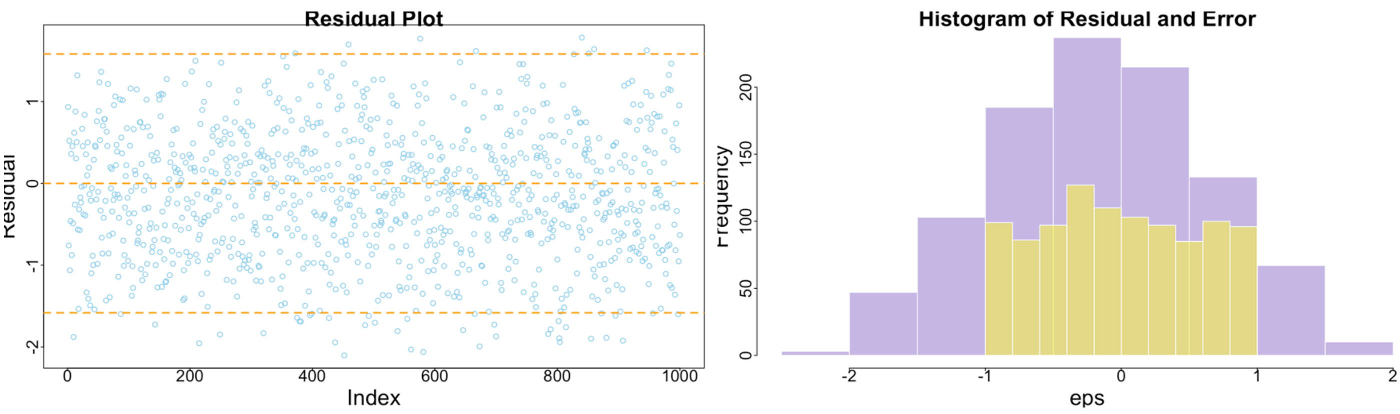
- Histogram of Residual and Error：
  右上图，采用$n = 1000$的原始数据来估计，两个histogram都是设定8个条图，其中紫色为ϵ，黄色为e，可以看出误差error集中在$[-1, 1]$，且$e \sim U(-1, 1)$，而residualϵ集中在$[-2, 2]$，为正态分布，$\epsilon \sim Normal\ Distribution$。

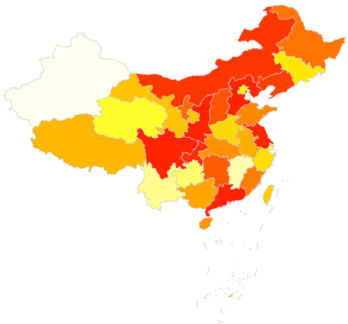
```
layout(matrix(c(1,2,3,3), 2, 2, byrow = TRUE))
par(mar=c(3,3,1,1), mgp=c(2,0.2,0), tcl=-0.2)
#####2.1######
x <- data.matrix(dat[2:4])
y_m <- data.matrix(dat[1])
xx_inv <- solve(t(x) %*% x)
B <- xx_inv %*% t(x) %*% y_m
y_pre <- x %*% B
eps <- y_pre-y_m
plot(c(1:1000), eps, main="Residual Plot",
     xlab="Index", ylab='Residual', col = 'skyblue',
     cex.lab=2, cex.axis=1.5, cex.main=2)
abline(h=0, col="orange",lwd=2,lty=2)
abline(h=2*sd(eps), col="orange",lwd=2,lty=2)
abline(h=(-2)*sd(eps), col="orange",lwd=2,lty=2)
#####2.2######
t_col <- function(color, percent = 50, name = NULL) {
  rgb.val <- col2rgb(color)

  t.col <- rgb(rgb.val[1], rgb.val[2], rgb.val[3],
               max = 255,
               alpha = (100 - percent) * 255 / 100,
               names = name)

  invisible(t.col)
}

hist(eps, main='Histogram of Residual and Error'
     ,col=t_col("mediumpurple3", perc = 50, name = ""), border=F, breaks=8,
     cex.lab=2, cex.axis=1.5, cex.main=2)
hist(e, col=t_col("yellow", perc = 50, name = ""), add=T, border=F, breaks=8)
#####2.3######
set.seed(2020270026)
GDP <- rnorm(35, mean=5000, sd=50)
cols <- heat.colors(35, alpha = 1)[order(GDP)]
cnMap <- read_sf('https://geo.datav.aliyun.com/areas_v2/bound/100000_full.json')
st_crs(cnMap) <- '+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs'
mapObj <- st_geometry(cnMap)
ext <- extent(cnMap)
plot(mapObj, xlim=c(ext[1],ext[2]), ylim=c(ext[3], ext[4]), border="gray80", col=c
ols, lwd=0.5)
```





3 Multiple Regression

首先来看这笔数据ISLR的结构，只有 `name` 不是continuous variable：

```
attach(Auto)
str(Auto)
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 1
4 161 141 54 223 241 2 ...
```

去除 `name` 来看correlation和distribution，发现其中有些variable相关性明显，恐怕有共线性问题：

```
#corr
wrap_1<-wrap(ggally_points,size=2,color="mediumpurple2",alpha=0.3)
wrap_2<-wrap(ggally_densityDiag,size=2,color="skyblue")
wrap_3 <- wrap(ggally_cor, size = 40, color = "darkgrey", fontface = "bold")
ggpairs(Auto[,1:8],
        lower = list(continuous = wrap_1),
        diag = list(continuous = wrap_2),
        higher = list(continuous = wrap_3))
```


```
cor(Auto[,1:8])
```

```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
## cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
## displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
## horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377
## weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000
## acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392
## year          0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199
## origin        0.5652088 -0.5689316   -0.6145351 -0.4551715 -0.5850054
##              acceleration       year     origin
## mpg             0.4233285  0.5805410  0.5652088
## cylinders      -0.5046834 -0.3456474 -0.5689316
## displacement   -0.5438005 -0.3698552 -0.6145351
## horsepower     -0.6891955 -0.4163615 -0.4551715
## weight         -0.4168392 -0.3091199 -0.5850054
## acceleration    1.0000000  0.2903161  0.2127458
## year            0.2903161  1.0000000  0.1815277
## origin          0.2127458  0.1815277  1.0000000
```

接着设定 `y=mpg` 来做回归，由于此题不需要剔除共线性高的变量，因此全部带入回归，并配合所有量变与 `mpg` 的correlation来解释结果。
首先 `weight` 、 `cylinders` 、 `acceleration` 的估参数不显著，比对correlation可以得知不是correlation越高，该变量的估计结果就会越显著。 虽然有些变量不显著，然而此model的R^2 高达0.8，且整个model十分显著，整体表现仍然很好，但一定要记得此模型有共线性问题，因此model的结果仍有问题存在。

```
lm.fit <- lm(mpg~.,data=Auto[,1:8])
summary(lm.fit)
```

```
##
## Call:
## lm(formula = mpg ~ ., data = Auto[, 1:8])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5903 -2.1565 -0.1169  1.8690 13.0604
##
## Coefficients:
##                Estimate Std. Error t value             Pr(>|t|)
## (Intercept)  -17.218435   4.644294  -3.707              0.00024 ***
## cylinders     -0.493376   0.323282  -1.526              0.12780
## displacement   0.019896   0.007515   2.647              0.00844 **
## horsepower    -0.016951   0.013787  -1.230              0.21963
## weight        -0.006474   0.000652  -9.929 < 0.0000000000000002 ***
## acceleration   0.080576   0.098845   0.815              0.41548
## year           0.750773   0.050973  14.729 < 0.0000000000000002 ***
## origin         1.426141   0.278136   5.127          0.000000467 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.328 on 384 degrees of freedom
## Multiple R-squared:  0.8215, Adjusted R-squared:  0.8182
## F-statistic: 252.4 on 7 and 384 DF,  p-value: < 0.00000000000000022
```

correlation：

```
cor(Auto[,1:8])[1,]
```

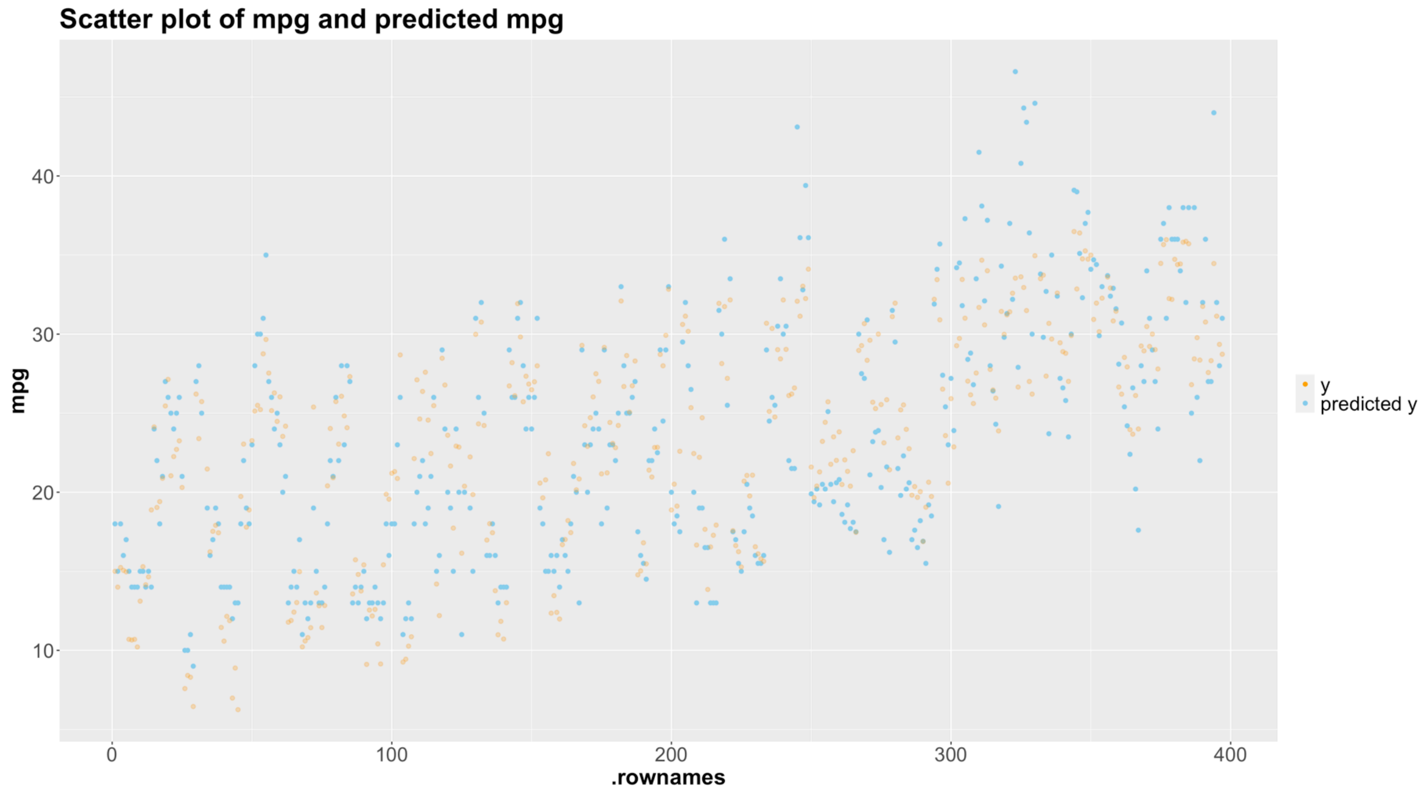
```
##          mpg    cylinders displacement   horsepower       weight acceleration
##    1.0000000   -0.7776175   -0.8051269   -0.7784268   -0.8322442    0.4233285
##         year       origin
##    0.5805410    0.5652088
```

下表列出前几笔 `mpg` 与 `mpg` 预测值的数值，以及残差...等：

```
model.diag.metrics <- augment(lm.fit)
model.diag.metrics <- as.data.frame(model.diag.metrics)
model.diag.metrics$.rownames <- as.numeric(model.diag.metrics$.rownames)
kbl(head(model.diag.metrics[,c(2,10:15)])) %>%
  kable_styling(bootstrap_options = c("striped", "hover"), full_width = F, font_si
ze = 12)
```

mpg	.fitted	.resid	.hat	.sigma	.cooksd	.std.resid
18	15.00096	2.9990413	0.0232314	3.328414	0.0024722	0.9118948
15	13.99930	1.0007008	0.0181167	3.331624	0.0002124	0.3034817
18	15.24045	2.7595530	0.0215582	3.328973	0.0019357	0.8383577
16	15.06191	0.9380941	0.0222674	3.331671	0.0002314	0.2850982
17	14.96718	2.0328224	0.0258697	3.330361	0.0012717	0.6189407
15	10.69562	4.3043766	0.0344457	3.324497	0.0077273	1.3163763

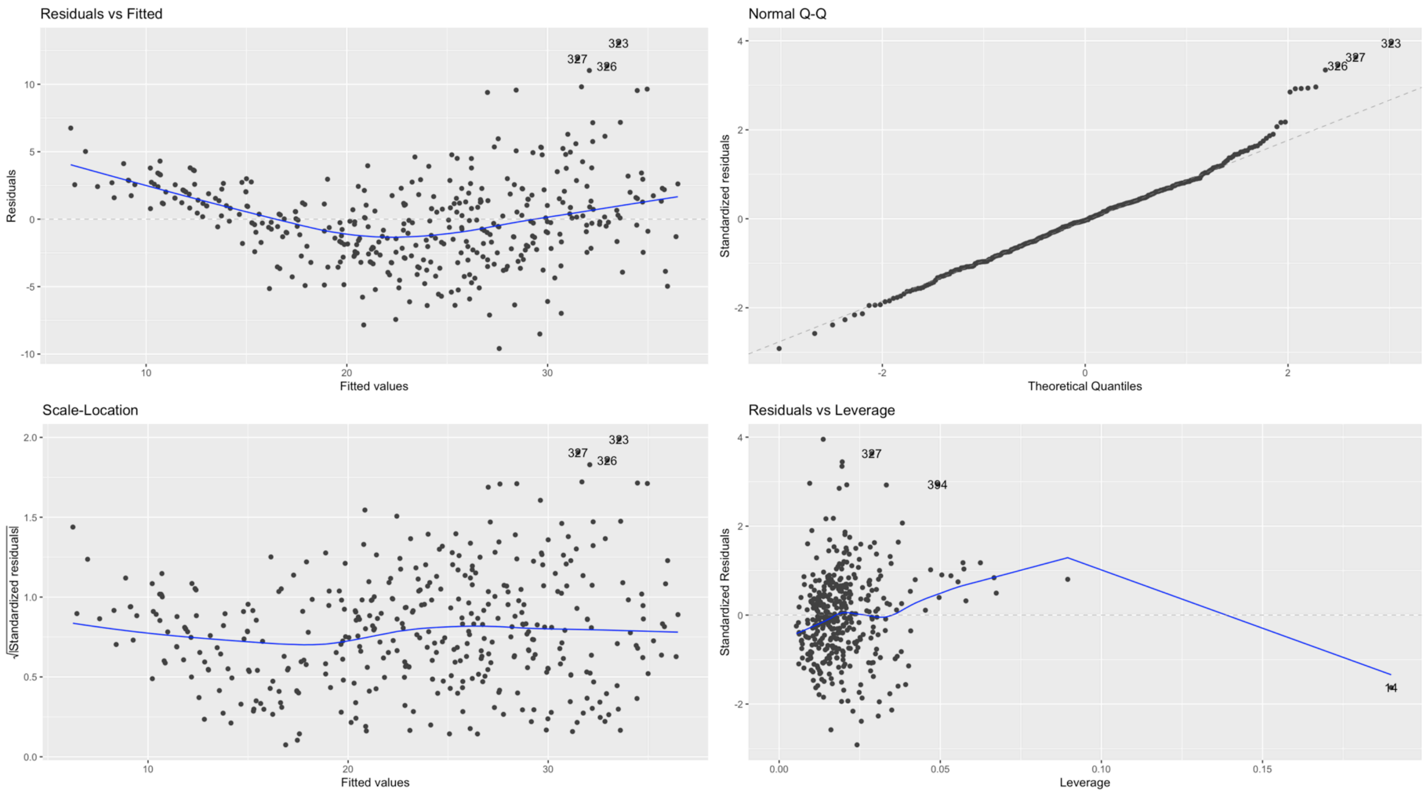
```
ggplot(model.diag.metrics, aes(x=.rownames, y=mpg)) +
  geom_point(aes(color='skyblue'),show.legend = T) +
  geom_point(aes(x=.rownames, y=.fitted,color = 'orange'), alpha = 0.3,show.legend
= T) +
  scale_colour_manual(values = c("orange","skyblue"), labels = c('y','predicted y'
),name = ' ')+
  theme(plot.title = element_text(size=25, face="bold"),
        axis.title = element_text(size=20, face="bold"),
        axis.text = element_text(size=18),
        legend.title=element_blank(),
        legend.text = element_text(size=18)) +
  labs(title =paste('Scatter plot of mpg and predicted mpg'))
```



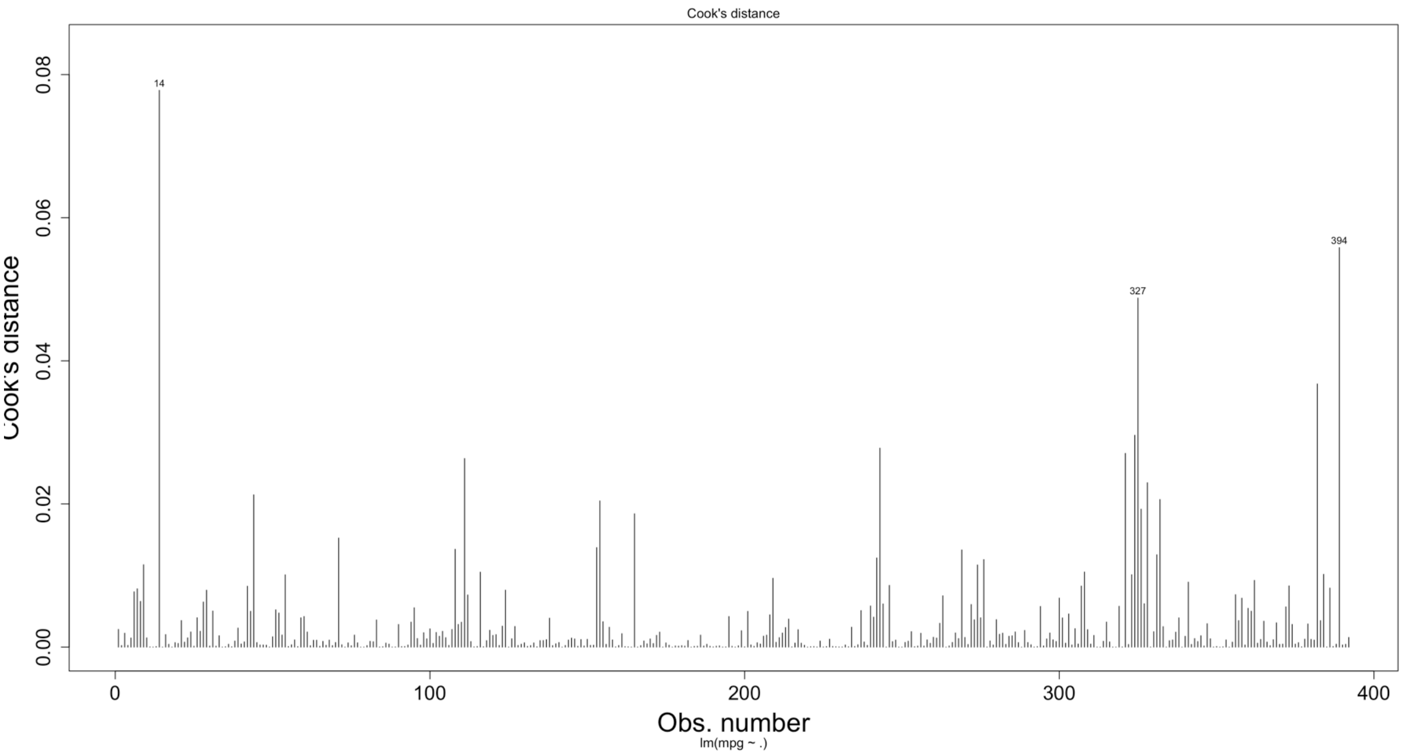
接着来看diagnostic plots，下为详细解释，总归来说，此model的拟合诊断结果很不错： - residual：希望分布呈现正太，且与fitted value间的scatter plot没明显形状（配适越接近水平线越佳），此图可以看出分布呈现正态但超略微右偏，且其与fitted value间的scatter plot没明显形状，只有三笔数据的residual是比较异常的，但整体而言表现极好。

- leverage：
  - Leverage point：
    这样的点拥有很极端的x值。比如其他样本点的x值都只有几十，而有一个样本点的x值超过了100，这时候这个点将会极大地影响回归模型。就像物理中的力矩一样，这种样本点有很高的leverage，所以叫做leverage points。有的leverage points可以很好地融入模型中，不会对模型造成很大的影响，通常也叫做good leverage points。有时，这种good leverage points证明了模型的普适性；但是它们同时会增大R^2 ，使对模型过度自信。所以good leverage points也并不是完全没有弊端。
  - Influential point：
    有good leverage points自然有bad leverage points。既是leverage points又是outliers的点就被称为bad leverage points，也就是influential points。它们的存在极大地影响了模型的可靠性，因为它们会把回归直线向自己的方向“拉扯”。判断influential points的方式可以用图上的Cook‘s distance，若大于1，就认为这个点是异常点。
  - 此数据：
    虽有Leverage point但不是Influential point，所以虽对模型产生影响但不至于到太大，且有才三笔数据罢了。

```
#Diagnostic plots
autoplot(lm.fit)
```




```
# Cook's distance
plot(lm.fit, 4,cex.lab=2, cex.axis=1.5, cex.main=2)
```



4 Polynomial Regression

首先读取 Boston 数据，并决定 x = dis , y = nox 。

```r
attach(Boston)
str(Boston)
```

```
## 'data.frame':	506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ..
.
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

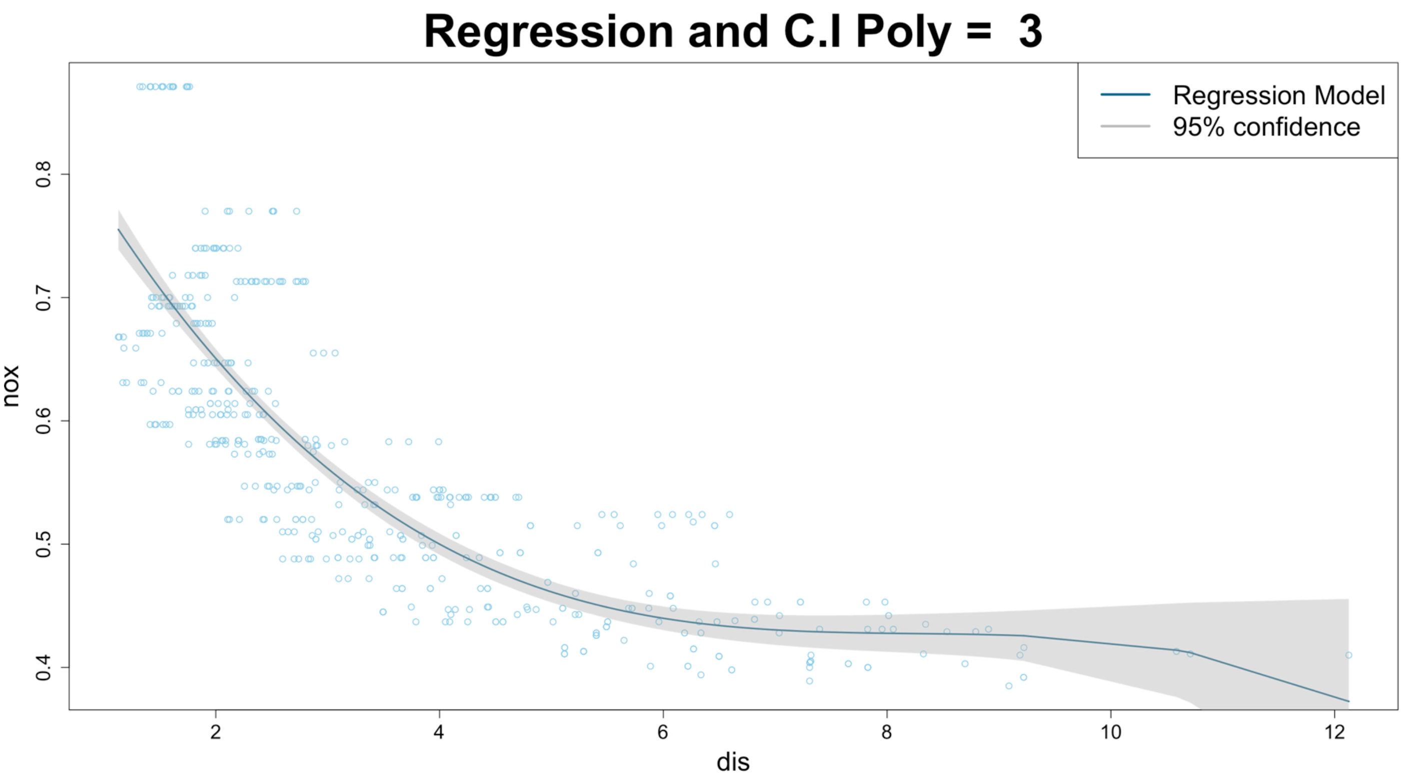
接着使用poly()来做回归，设定为3阶，配适结果极佳，不仅估计参数全都显著，model也显著，且R^2高达0.7。

```r
dataB <- Boston[,c(5,8)]
model <- lm(nox ~ poly(dis,3),data = dataB)
summary(model)
```

```
## 
## Call:
## lm(formula = nox ~ poly(dis, 3), data = dataB)
## 
## Residuals:
##       Min        1Q    Median        3Q       Max 
## -0.121130 -0.040619 -0.009738  0.023385  0.194904 
## 
## Coefficients:
##                Estimate Std. Error t value             Pr(>|t|)    
## (Intercept)    0.554695   0.002759 201.021 < 0.0000000000000002 ***
## poly(dis, 3)1 -2.003096   0.062071 -32.271 < 0.0000000000000002 ***
## poly(dis, 3)2  0.856330   0.062071  13.796 < 0.0000000000000002 ***
## poly(dis, 3)3 -0.318049   0.062071  -5.124          0.000000427 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.06207 on 502 degrees of freedom
## Multiple R-squared:  0.7148, Adjusted R-squared:  0.7131 
## F-statistic: 419.3 on 3 and 502 DF,  p-value: < 0.00000000000000022
```

下图为 `nox` 和预测值 `nox` ，以及预测值 `nox` 的95%C.I：

```r
lmpoly <- function(model,k){
  plot(dataB$dis, dataB$nox, main=paste("Regression and C.I Poly = ", k),
       xlab="dis", ylab='nox', col = 'skyblue',
       cex.lab=2, cex.axis=1.5, cex.main=3.5)
  myPredict <- predict(model , interval="confidence",level=0.95)
  ix <- sort(dis,index.return=T)$ix
  lines(dis[ix], myPredict[ix , 1], col='deepskyblue4', lwd=2 )
  polygon(c(rev(dis[ix]), dis[ix]), c(rev(myPredict[ ix,3]), myPredict[ ix,2]), co
l = rgb(0.7,0.7,0.7,0.4) , border = NA)
  legend('topright',c("Regression Model","95% confidence"),
         col=c("deepskyblue4","gray"), lwd=3, cex=2)
}
lmpoly(model,3)
```

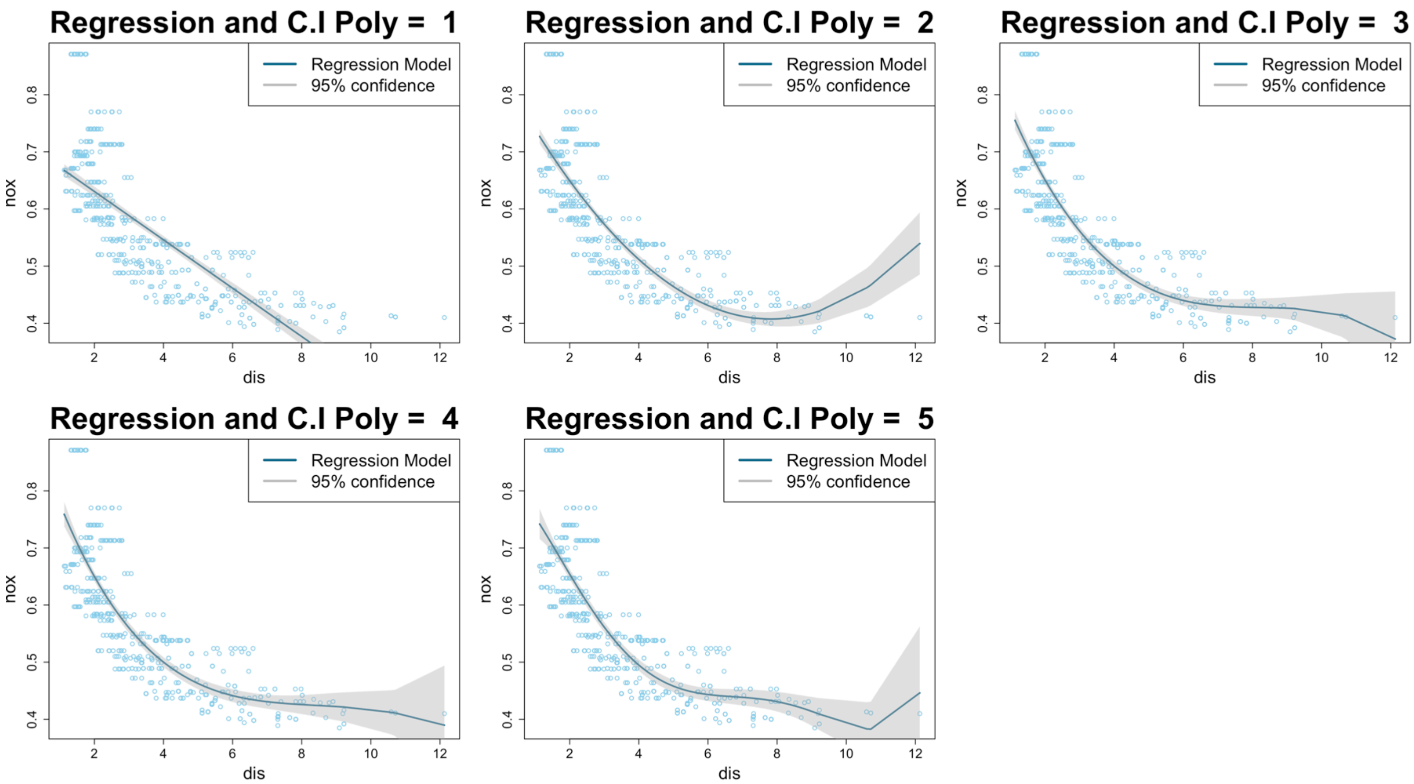


接着配饰1~5阶的Polynomial Regression，虽然阶次越高RSS越低，照理说RSS越低模型越好，但要考虑到overfitting的问题，所以不是RSS越低越好。、

```
#1~5
par(mfrow=c(2,3))
model1 <- lm(nox ~ poly(dis,1),data = dataB)
model2 <- lm(nox ~ poly(dis,2),data = dataB)
model3 <- lm(nox ~ poly(dis,3),data = dataB)
model4 <- lm(nox ~ poly(dis,4),data = dataB)
model5 <- lm(nox ~ poly(dis,5),data = dataB)

lmpoly <- function(model,k){
  plot(dataB$dis, dataB$nox, main=paste("Regression and C.I Poly = ", k),
       xlab="dis", ylab='nox', col = 'skyblue',
       cex.lab=2, cex.axis=1.5, cex.main=3.5)
  myPredict <- predict(model , interval="confidence",level=0.95)
  ix <- sort(dis,index.return=T)$ix
  lines(dis[ix], myPredict[ix , 1], col='deepskyblue4', lwd=2 )
  polygon(c(rev(dis[ix]), dis[ix]), c(rev(myPredict[ ix,3]), myPredict[ ix,2]), co
l = rgb(0.7,0.7,0.7,0.4) , border = NA)
  legend('topright',c("Regression Model","95% confidence"),
         col=c("deepskyblue4","gray"), lwd=3, cex=2)
}

lmpoly(model1,1)
lmpoly(model2,2)
lmpoly(model3,3)
lmpoly(model4,4)
lmpoly(model5,5)
```



```
#RSS
RSS_d <- cbind.data.frame(deviance(model1),deviance(model2),deviance(model3),
                          deviance(model4),deviance(model5))
colnames(RSS_d) <- c("1階", "2階","3階", "4階","5階")
kbl(RSS_d) %>%
  kable_styling(bootstrap_options = c("striped", "hover"), full_width = F, font_si
ze = 12)
```

1階	2階	3階	4階	5階
2.768563	2.035262	1.934107	1.932981	1.91529

为了以防overfitting，因此改用LOOCV来判断哪个模型最好，综合比对RMSE和R^2后，判定poly=3的model最合适，因为其RMSE最小且R^2最大。

```
train.control <- trainControl(method = "LOOCV")
model1 <- train(nox ~ poly(dis,1), data = dataB, method = "lm",
               trControl = train.control)
model2 <- train(nox ~ poly(dis,2), data = dataB, method = "lm",
                trControl = train.control)
model3 <- train(nox ~ poly(dis,3), data = dataB, method = "lm",
                trControl = train.control)
model4 <- train(nox ~ poly(dis,4), data = dataB, method = "lm",
                trControl = train.control)
model5 <- train(nox ~ poly(dis,5), data = dataB, method = "lm",
                trControl = train.control)
CV_d <- bind_rows(model1$results, model2$results,model3$results,
          model4$results,model5$results)[,2:4]
CV_d$poly <- c(1:5)
kbl(CV_d) %>%
  kable_styling(bootstrap_options = c("striped", "hover"), full_width = F, font_si
ze = 12)
```

RMSE	Rsquared	MAE	poly
0.0743227	0.5878228	0.0572823	1
0.0638706	0.6955946	0.0485290	2
0.0622476	0.7108755	0.0466460	3
0.0623500	0.7099383	0.0467063	4
0.0645358	0.6897521	0.0474606	5

最后再用smooth spline分别在df=12, 13, 14做拟合，三个拟合回归线差异不大，不过也较高阶，因此可能有overfitting问题。

```
#sp
fit1<-smooth.spline(dataB$dis,dataB$nox,df=12)
fit2<-smooth.spline(dataB$dis,dataB$nox,df=13)
fit3<-smooth.spline(dataB$dis,dataB$nox,df=14)
plot(dataB$dis, dataB$nox, main=paste('Smoothing Spline with different df'),
     xlab="dis", ylab='nox', col = 'skyblue',
       cex.lab=2, cex.axis=1.5, cex.main=3.5)
lines(fit3,col="brown",lwd=2)
lines(fit2,col="darkgreen",lwd=2)
lines(fit1,col="deepskyblue4",lwd=2)
legend("topright",c("df=12",'df=13','df=14'),col=c("deepskyblue4","darkgreen",'bro
wn'),lwd=2, cex=3)
```