Ve406 Lecture 15

Jing Liu

UM-SJTU Joint Institute

July 9, 2018

Q: What happens if the normality assumption is violated when others are fine?

- Since central limit theorem ensures the asymptotic distribution of $\hat{\boldsymbol{\beta}}$ to be

$$\text{Normal}\left(\boldsymbol{\beta}, \sigma^2\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\right)$$

which is identical to the sampling distribution of $\hat{\boldsymbol{\beta}}$ under normality

$$\varepsilon_i \sim \text{Normal}(0, \sigma^2) \qquad \text{for} \quad i = 1, 2, \ldots, n$$

- So normality is the least important assumption if n is sufficiently large.
- However, we need a systemic approach to fix normality if n is small.
- When $y_i > 0$ for all i, we can use a special power transformation known as

Box-Cox transformation

which can normalising the error distribution, stabilising the error variance and straightening the relationship between the response and the predictors.

- We assume the data is not conditionally normal, but the transformed data

$$\mathbf{y}^*_{(\lambda)}$$

is conditionally normal, that is, there exists λ such that

$$\mathbf{y}^*_{(\lambda)} \sim \text{Normal}\left(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I}\right)$$

where the transformation takes the following form

$$y^*_{i(\lambda)} = \begin{cases} \dfrac{y_i^\lambda - 1}{\lambda}, & \text{if} \quad \lambda \neq 0; \\ \ln y_i, & \text{if} \quad \lambda = 0. \end{cases}$$

- Box and Cox also proposed the following for y_i that might be negative

$$y^*_{i(\lambda_1,\lambda_2)} = \begin{cases} \dfrac{(y_i + \lambda_2)^{\lambda_1} - 1}{\lambda_1}, & \text{if} \quad \lambda \neq 0; \\ \ln\left(y_i + \lambda_2\right), & \text{if} \quad \lambda = 0. \end{cases}$$

- Box-Cox transformation is widely used partly because its form offers a simple way to choose the parameter λ using the maximum likelihood principle.
- Since $\mathbf{y}^*$ is assumed to be normal, its density is readily available

$$f(\mathbf{y}^*) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{(\mathbf{y}^* - \mathbf{X}\boldsymbol{\beta})^{\mathrm{T}}(\mathbf{y}^* - \mathbf{X}\boldsymbol{\beta})}{2\sigma^2}\right)$$

- The density of $\mathbf{y}$ can be found using the change of variable technique

$$\begin{aligned} f(\mathbf{y}) &= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{\left(\mathbf{y}^*_{(\lambda)} - \mathbf{X}\boldsymbol{\beta}\right)^{\mathrm{T}}\left(\mathbf{y}^*_{(\lambda)} - \mathbf{X}\boldsymbol{\beta}\right)}{2\sigma^2}\right) |\mathbf{J}(\lambda, \mathbf{y})| \\ &= \mathcal{L}\left(\lambda, \boldsymbol{\beta}, \sigma^2; \mathbf{y}, \mathbf{x}\right) \end{aligned}$$

where $|\mathbf{J}(\lambda, \mathbf{y})|$ is the Jacobian of changing from $\mathbf{y}$ to $\mathbf{y}^*_{(\lambda)}$.

- Notice the likelihood function $\mathcal{L}\left(\lambda, \boldsymbol{\beta}, \sigma^2; \mathbf{y}, \mathbf{x}\right)$ is proportional to $f(\mathbf{y}^*)$ for any given value of λ, thus the MLE for $(\boldsymbol{\beta}, \sigma^2)$ takes the usual form

$$\tilde{\boldsymbol{\beta}}(\lambda) = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}^*$$
$$\tilde{\sigma}^2(\lambda) = \frac{1}{n}\left(\mathbf{y}^*\right)^{\mathrm{T}}\left(\mathbf{I}-\mathbf{P}\right)\mathbf{y}^*$$

where $\mathbf{P} = \mathbf{X}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}$ is the projection matrix.

- Since the form that Box and Cox have chosen, the Jacobian is very simple

$$|\mathbf{J}(\lambda, \mathbf{y})| = \prod_{i=1}^{n} y_i^{\lambda-1}$$

- Using the above, we have the following profile log likelihood function

$$\begin{aligned}\ell(\lambda) &= \ln\left(\mathcal{L}(\lambda; \mathbf{y}, \mathbf{x}, \tilde{\boldsymbol{\beta}}(\lambda), \tilde{\sigma}^2(\lambda))\right) \\ &= -\frac{n}{2}\left(1+\ln 2\pi\right) - \frac{n}{2}\ln\left(\tilde{\sigma}^2(\lambda)\right) + (\lambda-1)\sum_{i=1}^{n}\ln y_i\end{aligned}$$

- Arranging and dropping constants, we see λ should be chosen to maximise

$$-\frac{n}{2}\ln\left(\sum_{i=1}^{n}\left(y_i^*(\lambda)-\hat{y}_i^*(\lambda)\right)^2\right)+(\lambda-1)\sum_{i=1}^{n}\ln y_i$$

- Consider the following simple example to understand Box-Cox transformation

`year`	between 1952 and 1980
`price`	in 1980 US dollars, converted to an index with 1961=100
`temp`	average temperature during the growing season
`h.rain`	total rainfall during the harvest period
`w.rain`	total rainfall during the preceding winter

- It was attempted to assess the quality of various Bordeaux wines.

- Anyone know something about wine and regression would be surprised to see

```
> summary(wine.LM)
```

```
Call:
lm(formula = price ~ temp + h.rain + w.rain + year, data = wine.df)

Residuals:
    Min      1Q  Median      3Q     Max
-14.077  -9.040  -1.018   3.172  26.991

Coefficients:
              Estimate Std. Error t value Pr(>t)
(Intercept) 1305.52761  597.31137   2.186  0.03977 *
temp          19.25337    3.92945   4.900 6.72e-05 ***
h.rain        -0.10121    0.03297  -3.070  0.00561 **
w.rain         0.05704    0.01975   2.889  0.00853 **
year          -0.82055    0.29140  -2.816  0.01007 *
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

Residual standard error: 11.69 on 22 degrees of freedom
Multiple R-squared:  0.7369,    Adjusted R-squared:  0.6891
F-statistic: 15.41 on 4 and 22 DF,  p-value: 3.806e-06
```

- It is surprising that we can explain so much of the variation in something as complex as wine quality using three simple climatic variables and the year.

- Of course, we cannot completely trust the results before looking it residuals
- First look at the residual plots to check the linearity and equal variance

```
> plot(wine.LM, which = 1)
```

- Then look at Residuals Vs Previous Residuals and acf to check independence

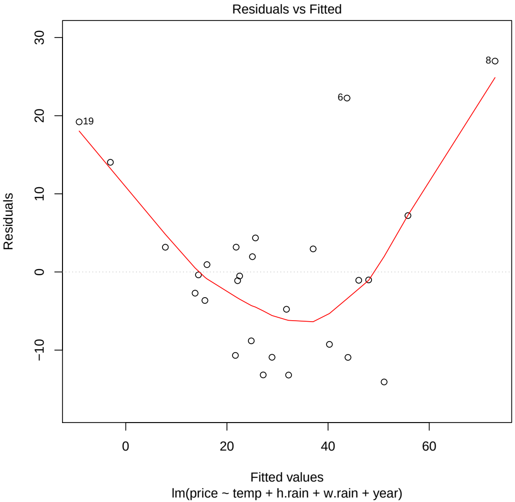
```
> res = wine.LM$residuals
> plot(res[-length(res)], res[-1], ylab = "Residual",
+       xlab = "Previous Residual",
+       main = "Residuals Vs Previous Residuals")
> acf(res)
```

- Lastly look at QQ-normal plot, and use Shapiro-Wilk when n is small.

```
> plot(wine.LM, which = 2)
> shapiro.test(res)
```

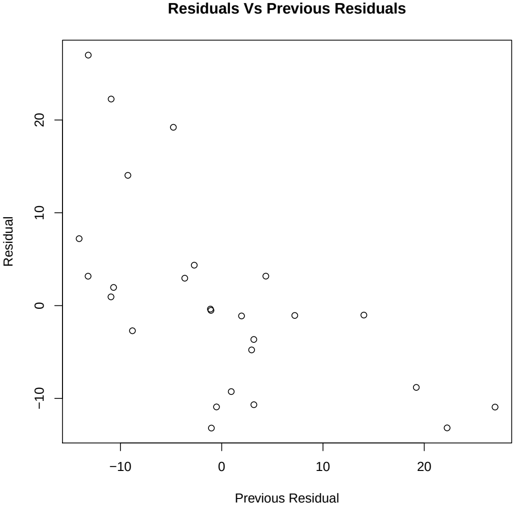
```
        Shapiro-Wilk normality test

data:  res
W = 0.91142, p-value = 0.02464
```

- It is possible that either the linearity or the equal variance is violated, or both



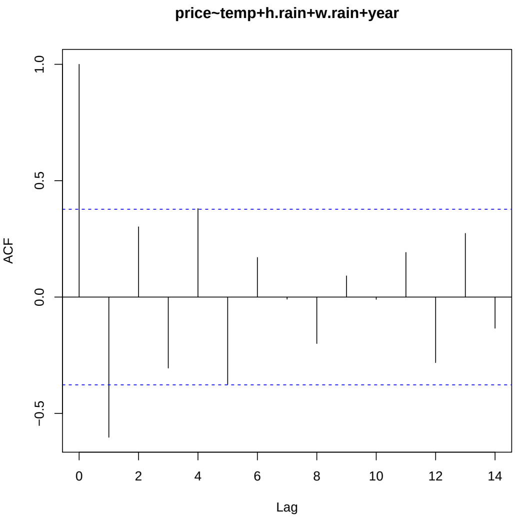
- But it could be also to due outliers, and outliers here are always positive.

- It is clear that the model lack independence,

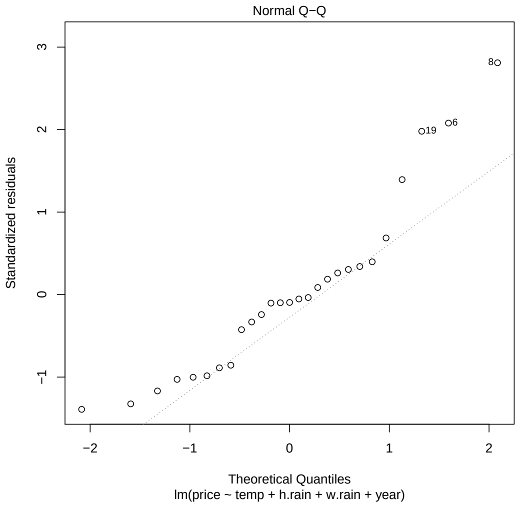


which is not surprising, since the data points are ordered according to `year`.

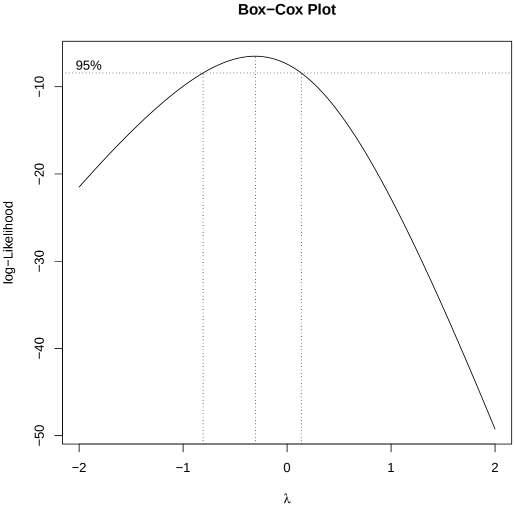
- There seems to exist higher order dependence structure in errors



- The normal plot of the residuals shows that the residuals are right skewed.



- Usually one wants to fix the linearity first by transforming the predictors, but when all assumptions are badly violated, we start with Box-Cox transform.



- The Box-Cox plot is implemented in R

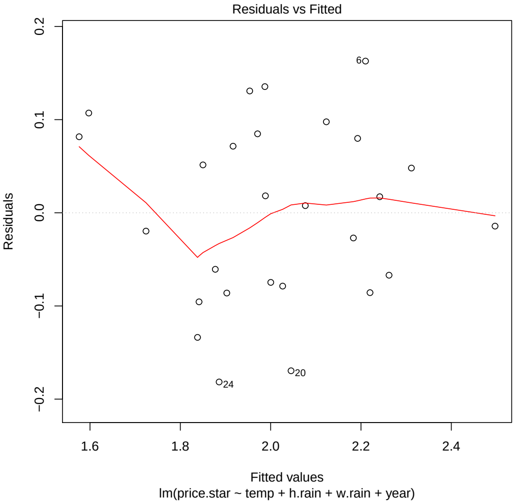
```
> library(MASS)
> bc = boxcox(wine.LM)
> title("Box-Cox Plot")
```

- Using the optimal λ to transform,

```
> lambda = bc$x[which.max(bc$y)]
>
> price.star = (wine.df$price^lambda - 1 ) / lambda
>
> wine.opt.LM =
+   lm(price.star~temp+h.rain+w.rain+year,
+      data = wine.df)

> shapiro.test(wine.opt.LM$residuals)
```

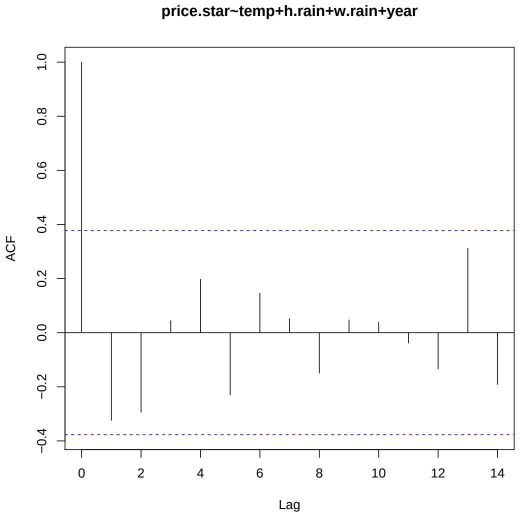
```
        Shapiro-Wilk normality test

data:  wine.opt.LM$residuals
W = 0.96602, p-value = 0.5009
```

- The assumptions of linearity and equal variance are improved drastically.



- Polynomial terms were considered, but none are significant.

- The independence assumption seems to be OK as well.



- This optimal transformation seems to fix all the issues!

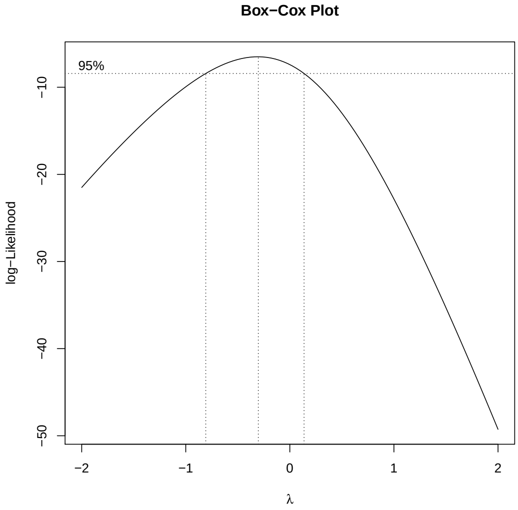
- One downside of doing an arbitrary power transform is interpretation

```
> summary(wine.opt.LM)
```

```
Call:
lm(formula = price.star ~ temp + h.rain + w.rain + year, data = wine.df)

Residuals:
      Min        1Q    Median        3Q       Max
-0.181583 -0.076683  0.007709  0.080743  0.162856

Coefficients:
              Estimate Std. Error t value Pr(>t)
(Intercept) 15.3772414  5.3246430   2.888  0.00854 **
temp         0.2328954  0.0350285   6.649 1.10e-06 ***
h.rain      -0.0014567  0.0002939  -4.956 5.86e-05 ***
w.rain       0.0003881  0.0001760   2.205  0.03824 *
year        -0.0087589  0.0025977  -3.372  0.00275 **
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

Residual standard error: 0.1042 on 22 degrees of freedom
Multiple R-squared:  0.8334,    Adjusted R-squared:  0.8031
F-statistic: 27.52 on 4 and 22 DF,  p-value: 2.784e-08
```

- We prefer integer power or log transformation over arbitrary real power.

- Notice we used maximum likelihood theory to obtain the estimated λ



- So the usual interpretation of a confidence interval applies, we could also use

```
> wine.log.LM =
+    lm(log(price)~temp+h.rain+w.rain+year,
+       data = wine.df)
```

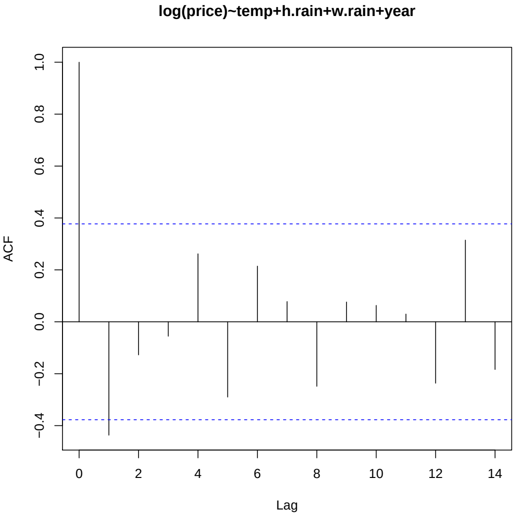
```
> shapiro.test(wine.log.LM$residuals)
```

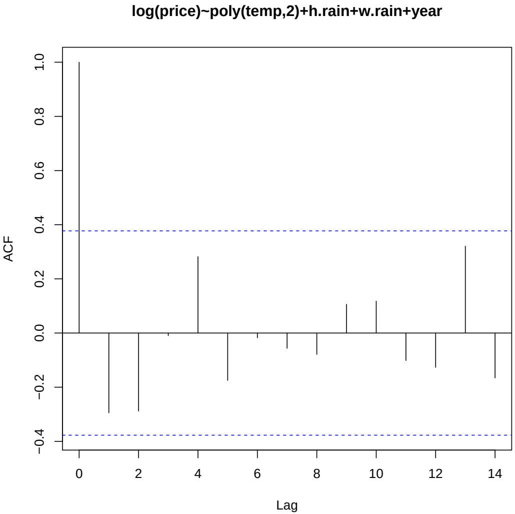
```
        Shapiro-Wilk normality test

data:  wine.log.LM$residuals
W = 0.95907, p-value = 0.352
```

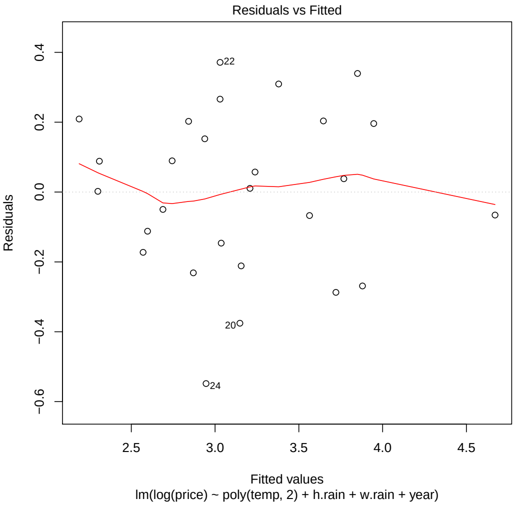
- Except for the independence assumption, using $\lambda = 0$ is just as good.
- However, when we improve the linearity assumption using a quadratic term

```
> wine.final.LM =
+    lm(log(price)~poly(temp,2)+h.rain+w.rain+year,
+       data = wine.df)
```

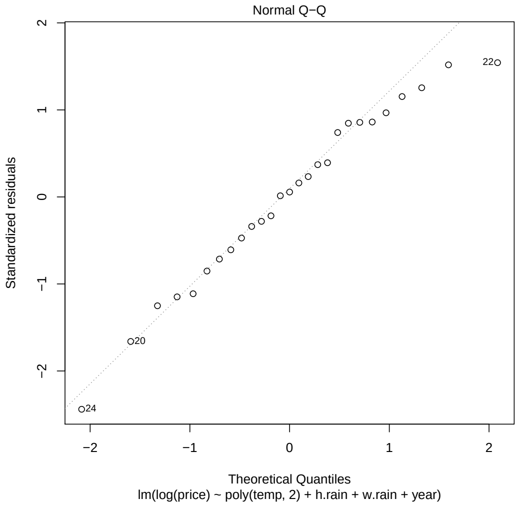
all the assumptions seem to be satisfied.


log(price)~temp+h.rain+w.rain+year
ACF
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
0
2
4
6
8
10
12
14
Lag


log(price)~poly(temp,2)+h.rain+w.rain+year
ACF
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
0
2
4
6
8
10
12
14
Lag



Residuals vs Fitted
Residuals
0.4
0.2
0.0
−0.2
−0.4
−0.6
22
20
24
2.5
3.0
3.5
4.0
4.5
Fitted values
lm(log(price) ~ poly(temp, 2) + h.rain + w.rain + year)


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(log(price) ~ poly(temp, 2) + h.rain + w.rain + year)
22
20
24
−2
−1
0
1
2

```
> summary(wine.final.LM)
```

```
Call:
lm(formula = log(price) ~ poly(temp, 2) + h.rain + w.rain + year,
    data = wine.df)

Residuals:
     Min       1Q   Median       3Q      Max
-0.54831 -0.15940  0.01057  0.19933  0.37131

Coefficients:
                 Estimate Std. Error t value Pr(>t)
(Intercept)    52.4701332 12.7072508   4.129 0.000477 ***
poly(temp, 2)1  2.1172509  0.2897324   7.308 3.41e-07 ***
poly(temp, 2)2  0.6161551  0.2755380   2.236 0.036323 *
h.rain         -0.0038060  0.0007285  -5.225 3.53e-05 ***
w.rain          0.0015263  0.0004602   3.317 0.003277 **
year           -0.0252639  0.0064470  -3.919 0.000789 ***
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

Residual standard error: 0.258 on 21 degrees of freedom
Multiple R-squared:  0.8634,    Adjusted R-squared:  0.8308
F-statistic: 26.54 on 5 and 21 DF,  p-value: 2.08e-08
```

- Because we have only a small number of observations, we will not consider removing any point from the dataset to improve the stability.

- In the regression setting, we assume the response is given by

$$Y_i = \mathbb{E}\left[Y_i \mid X_i = x_i\right] + \varepsilon_i$$

where the true conditional mean is a function of x_i

$$\mathbb{E}\left[Y_i \mid X_i = x_i\right] = f(x_i)$$

- So far, when we often model the non-linear relationship using a polynomial

$$f(x_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \cdots + \beta_d x_i^d$$

which leads polynomial regression

$$y_i = \hat{\beta}_0 + \hat{\beta}_1 x_i + \hat{\beta}_2 x_i^2 + \hat{\beta}_3 x_i^3 + \cdots + \hat{\beta}_d x_i^d + \hat{e}_i$$

and β_j are estimated using least squares or maximum likelihood principle.

- For large enough d, the polynomial

$$f(x) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \cdots + \beta_d x_i^d$$

allows us to model an extremely non-linear relationship.

- However, its weakness is the fact it imposes a global structure.
- The idea of spline is assume piecewise polynomials for the conditional mean,

$$g(x)$$

that is, dividing the region of x into contiguous intervals,

$$[\xi_{\ell-1}, \xi_\ell] \qquad \text{for} \quad \ell = 1, 2, \ldots, L$$

and model $\mathbb{E}[Y \mid X]$ by a separate polynomial in each interval.

- The points ξ_ℓ, which split the region, are known as knots.

- We prefer the conditional mean to be a smooth function, thus we require

$$g(x)$$

to be twice continuously differentiable, which can be done by using cubics

$$s_\ell(x) = \beta_{0\ell} + \beta_{1\ell}x + \beta_{2\ell}x^2 + \beta_{3\ell}x^3$$

which leads to the so-called cubic spline, with the following the constraints

$$s_{\ell-1}(\xi_\ell) = s_\ell(\xi_\ell); \qquad s'_{\ell-1}(\xi_\ell) = s'_\ell(\xi_\ell); \qquad s''_{\ell-1}(\xi_\ell) = s''_\ell(\xi_\ell)$$

thus the number of parameters of cubic spline with L knots is $4 + L$.

- The parameters in the estimated $\hat{g}$ are estimated by solving

$$\min \sum_{i=1}^{n} (y_i - \hat{g}(x_i))^2$$

- So, the model becomes the following, which is not that different from SLR

$$y_i = \hat{g}(x_i) + \hat{e}_i$$

  where the $4 + L$ number parameters in $\hat{g}(x)$ are found by minimising the sum of squared residuals once knots are decided.

- Consider the following dataset for an illustration of cubic spline in this setup

wage	Raw wage in the Mid-Atlantic region
age	Age of the worker

```
> summary(age)
```

```
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
18.00   33.75   42.00   42.41   51.00   80.00
```

- Ideally, we would like to place more knots at where the conditional mean is rapidly changing, of course, we don't unusually have that information.

- In practice, equally spaced knots are often used, or where we suspect changes

```
> library(splines)
>
> wage_cubic_spline.LM =
+   lm(wage ~ bs(age, knots = c(25,40,60)),
+      data = Wage)
```

```
> summary(wage_cubic_spline.LM)
```

```
Call:
lm(formula = wage ~ bs(age, knots = c(25, 40, 60)), data = Wage)

Coefficients:
                                Estimate Std. Error t value Pr(>t)
(Intercept)                       60.494      9.460   6.394 1.86e-10 ***
bs(age, knots = c(25, 40, 60))1    3.980     12.538   0.317 0.750899
bs(age, knots = c(25, 40, 60))2   44.631      9.626   4.636 3.70e-06 ***
bs(age, knots = c(25, 40, 60))3   62.839     10.755   5.843 5.69e-09 ***
bs(age, knots = c(25, 40, 60))4   55.991     10.706   5.230 1.81e-07 ***
bs(age, knots = c(25, 40, 60))5   50.688     14.402   3.520 0.000439 ***
bs(age, knots = c(25, 40, 60))6   16.606     19.126   0.868 0.385338
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

Residual standard error: 39.92 on 2993 degrees of freedom
Multiple R-squared:  0.08642,   Adjusted R-squared:  0.08459
F-statistic: 47.19 on 6 and 2993 DF,  p-value: < 2.2e-16
```

- For convenience and numerical reasons, R uses the following bases for $g(x)$

$$1,\ x,\ x^2,\ x^3,\ h(x,\xi_1),\ h(x,\xi_2),\ldots,h(x,\xi_L)$$

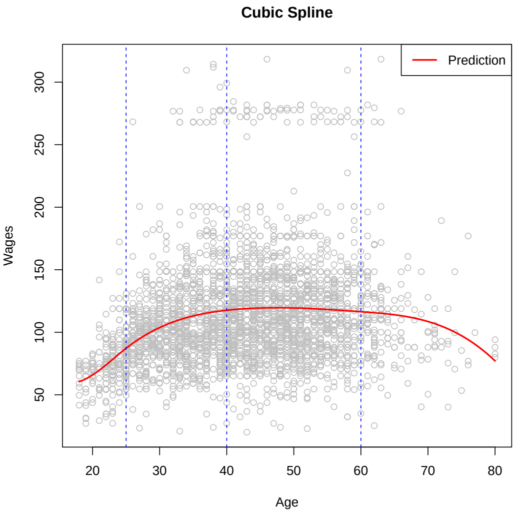
where

$$h(x,\xi)=\begin{cases}(x-\xi)^3, & \text{if} \quad x>\xi;\\ 0, & \text{otherwise.}\end{cases}$$

- So the summary gives the coefficients for each of the followings respectively

$$1,\ x,\ x^2,\ x^3,\ h(x,25),\ h(x,40),\ h(x,60)$$

```
> age.pred = seq(min(age),
+                max(age), length.out = 100)
>
> wage.pred =
+   predict(wage_cubic_spline.LM,
+           newdata = data.frame(age=age.pred))
```


Cubic Spline
Prediction
Wages
300
250
200
150
100
50
20
30
40
50
60
70
80
Age

- Since cubic splines are extremely flexible when L is large,

  overfitting is a serious concern

- The smoothing spline in R

  `smooth.spline`

  we used from time to time is a penalised spline for a given parameter $\lambda > 0$

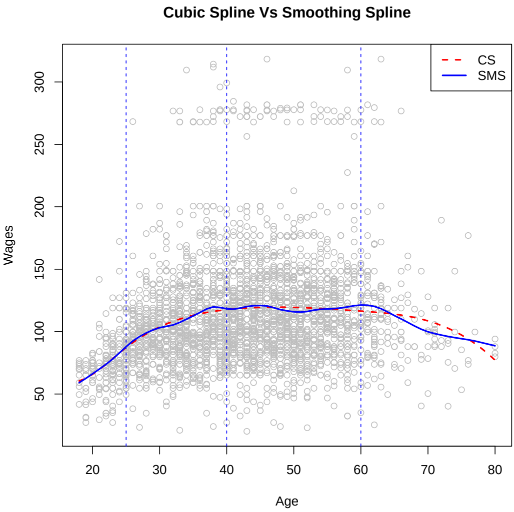
$$Y_i = g(x_i) + \varepsilon_i;$$

  where $\hat{g}(x)$ is chosen to minimise the following

$$\sum_{i=1}^{n} \left(y_i - \hat{g}(x_i)\right)^2 + \boxed{\lambda \int \left(g''(x)\right)^2 dx}$$

Q: Why will smoothing spline tackle the overfitting problem?

```
> wage_cubic_spline.smooth =
+   smooth.spline(age, wage, df=16)

> plot(age, wage, col="grey",
+      xlab="Age", ylab="Wages",
+      main = "Cubic Spline Vs Smoothing Spline")
>
> points(age.pred, wage.pred,
+        col="red", lwd=2, lty = 2, type="l")
>
> lines(wage_cubic_spline.smooth, col="blue",lwd=2)
>
> abline(v=c(25,40,60), lty=2, col="blue")
>
> legend("topright", c("CS","SMS"), lwd =2 ,
+        col = c(2,4), lty = c(2,1))
```


Cubic Spline Vs Smoothing Spline
CS
SMS
Wages
300
250
200
150
100
50
20
30
40
50
60
70
80
Age

- Note having a larger df of 16 leads a more wiggly smoothing spline than the cubic spline with only 3 knots, which is equivalent to only df of 7.
- The overfitting problem can be addressed by using a larger λ

```
> wage_cubic_spline_lambda.smooth =
+   smooth.spline(age, wage, cv = TRUE)
>
> wage_cubic_spline_lambda.smooth
```

```
Call:
smooth.spline(x = age, y = wage, cv = TRUE)

Smoothing Parameter  spar= 0.6988943  lambda= 0.02792303 (12 iterations)
Equivalent Degrees of Freedom (Df): 6.794596
Penalized Criterion (RSS): 75215.9
PRESS(l.o.o. CV): 1593.383
```

```
> wage_cubic_spline.smooth
```

```
Call:
smooth.spline(x = age, y = wage, df = 16)

Smoothing Parameter  spar= 0.4732071  lambda= 0.0006537868 (13 iterations)
Equivalent Degrees of Freedom (Df): 16.00237
Penalized Criterion (RSS): 61597.01
GCV: 1599.69
```

- The penalty parameter λ is chosen according to cross-validation.