

# Ve406 Applied Regression Analysis using R

## 1 Introduction

### 1.1 Course Profile

#### 1.1.1 Course Information

- **Course Description:**

This course provides an introduction to the process and procedures of statistical modelling. We will explore real data sets, examine various models for the data, assess the validity of their assumptions, and determine which conclusions we can make, if any. In this course you will learn how to program in R and how to use R for effective data analysis.

- **Learning Outcomes:**

After successful completion of this course, you should be able to

1. Explore data graphically.
2. Select appropriate models.
3. Implement those models in R.
4. Examine those models critically.
5. Interpret the results of those models to non-statisticians.

- **Who should take this class?**

The prerequisite for this class is computer/programming knowledge at the level of Vg101 (or above), and statistics knowledge at the level of Ve401 (or above). Both undergraduates and graduate students are welcome to take the course.

#### 1.1.2 Contact Information

- **Instructor:**

Tong Zhu

- **Lectures:**

Tuesday (06:20pm – 08:55pm) in **E3-102**  
Thursday (06:20pm – 08:00pm) in **E3-102**

- **Office Hours:**

Thursday (12:00pm – 14:00pm) in **JI-Building 442**

- **Email:**

tong.zhu@protonmail.com

When sending an email related to this course please include the tag [Ve406] in the subject e.g. Subject: [Ve406] Special Request

- Teaching Assistant/s:

Wu Zheng

### 1.1.3 Grading Policy

- Assignment:

15% There will be 5 assignments in the form of problem sets.

- Lab/Workshop:

10% There will be 4 labs/Workshops.

- Project:

25% There will be a project in the form of a challenge.

- Exam:

50%	There will be two exams:	Midterm	Final
		25%	25%

- Quiz (Optional):

15% Quizzes will be given frequently in class.

- For those who attempt all quizzes, their grade is whichever is the higher of:

0. 25% ALW + 25% Proj + 0% Quiz + 50% Exam
1. 25% ALW + 10% Proj + 15% Quiz + 50% Exam
2. 25% ALW + 40% Proj + 15% Quiz + 20% Exam

- For this course, the grade will be curved to achieve a median grade of “B+”.

### 1.1.4 Project

- Each of you need to be in one and only one 3-member team for the project.

- The project will be graded according to the following three aspects:

1. Oral Presentation of your model
2. Poster Presentation of your model
3. Prediction Accuracy of your model

each of those three aspects has an equal weight.

- Each member of the same team will receive the same project mark.

- You will be working on real data (~~T E S T L A B~~), some of part 1 is [here]



but part 2 will not materialise before the due date.

### 1.1.5 Honour Code

- Honesty and trust are important. Students are responsible for familiarising themselves with what is considered as a violation of honour code.
- Assignments/projects are to be solved by each student individually. You are encouraged to discuss problems with other students, but you are advised not to show your written work to others. Copying someone else's work is a very serious violation of the honour code.
- Students may read resources on the Internet, such as articles on Wikipedia, Wolfram MathWorld or any other forums, but you are not allowed to post the original assignment question online and ask for answers. It is regarded as a violation of the honour code.
- Since it is impossible to list all conceivable instance of honour code violations, the students has the responsibility to always act in a professional manner and to seek clarification from appropriate sources if their or another student's conduct is suspected to be in conflict with the intended spirit of the honour code.

### 1.1.6 Teaching Schedule

Week	Topics	Others
1	Introduction Simple Linear Regression Least-Squares and Maximum Likelihood	A1R
2	Diagnostics Transformation	W1R
3	Workshop 1 Multiple Linear Regression Diagnostics	A1D A2R
4	National Holiday	
5	Inference Categorical regressors Collinearity	W1D W2R
6	Variable Selections Influential points and Outliers	A2D A3R

	Workshop 2	W2D
7	Heteroskedasticity Correlated Noise	W3R
8	Nonlinear Regression <b>Midterm Exam</b>	A3D A4R
	Workshop 3	W3D
9	Logistic Regression Poisson Regression	W4R
10	Nonparametric Regression Generalized Linear Model	A4D A5R
	Workshop 4	W4D
11	Estimating Equations Generalised Additive Models	W5R
12	Principal Component Analysis Factor Analysis	A5D Poster
	Mixture Models Survival Analysis (Optional) Project Presentation	
14	<b>Final Exam</b>	

### 1.1.7 Textbook

- Simon J. Sheather (2010)  
A Modern Approach to Regression with R
- Gromlund and Wickham (2016)  
R for Data Science
- Myers et al. (2010)  
Generalized Linear Models with Applications in Engineering and the Sciences
- Fox. (2015)  
Applied Regression Analysis and Generalized Linear Models

## 1.2 R language

### 1.2.1 Python Vs R

#### Data Science Wars: Python Vs R

Q: Python probably needs no introduction to this audience, however, what is R?



- There are two unusual things regarding its origin.
- Firstly, it also comes from a quiet and small place,  
University of Auckland, New Zealand
- Secondly, it was created by two statisticians instead of a computer scientist  
Ross Ihaka and Robert Gentleman

It is a programming language for statistics/data science.



- Release Year  
1991
- Inspiration  
C
- Purpose  
Emphasises productivity and code readability.



- Release Year  
1995
- Inspiration  
S
- Purpose  
Focuses on better, user friendly data analysis, statistics and graphical models.

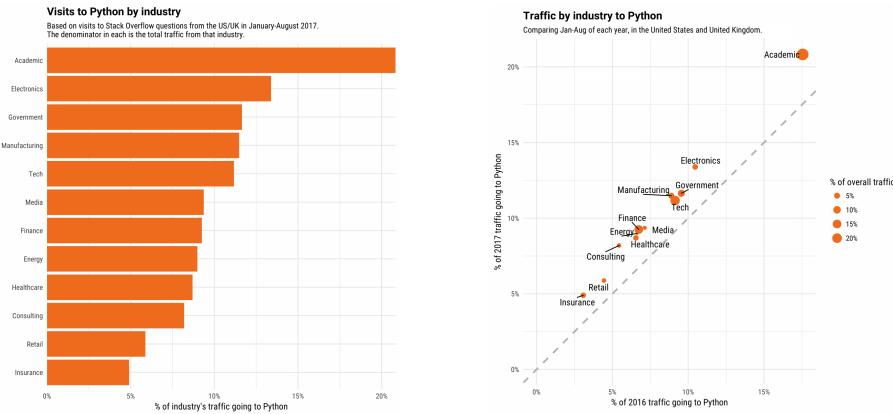
The S language was a commercial software developed at the famous Bell Laboratories by John Chambers and his collaborators Rick Becker, Allan Wilks and Duncan Temple Lang over the years from 1975 to 1998.



- Usability  
Coding and debugging is easier to do. Any piece of functionality is always written the same way.
- Ease of Learning  
Python's focus on readability and simplicity makes that its learning curve is relatively low and gradual.
- Usability  
Statistical models can be written with only a few lines. The same piece of functionality can be written in several ways.
- Ease of Learning  
Require only a minimum knowledge of computing at start, then a steep learning curve later on. However, R is easy to learn for experience programmers in C.

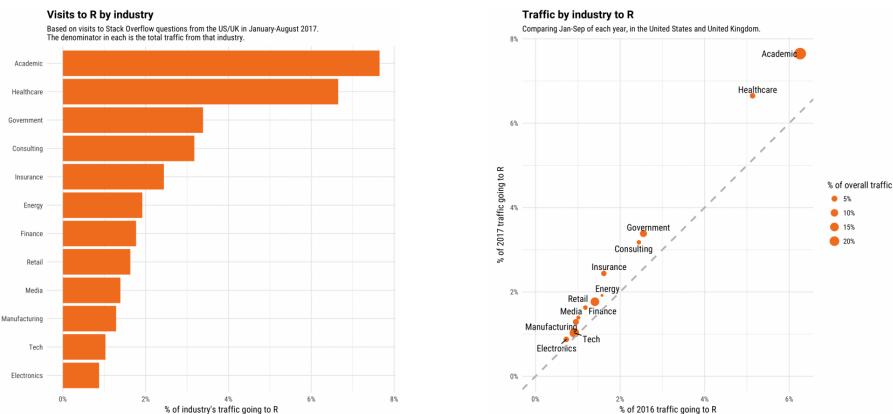


- Community  
Python is used by programmers that want to delve into data analysis or apply statistical techniques, and by developers that turn to data scientists.



- Community

Traditionally, R had been used primarily in academic and research institutes. However, R has expanded into the enterprise market.



In 2015, Microsoft closed its acquisition of Revolution Analytics, a commercial provider of and services for R. Revolution has made R enterprise-ready with speed and scalability for the largest data warehouses and Hadoop systems.  
<https://www.cio.com/article/2906456/data-analytics/microsoft-closes-acquisition-of-r-software-and-services-provider.html>



- Usage

Python is generally used when the data analysis tasks need to be integrated with web apps or incorporated into a production database.

- Good for

Implementing algorithms for production use.  
Easy to share your work with other developers.



- Usage

R is mainly used when the data analysis tasks require standalone computing

- Good for

Beginners in statistics.  
Easy to interact with.

- So if you are good at programming and you want to learn statistics,

R

is the way to go, especially, if you begin with something standard.

- However, if you are a member of a research and development team,

Python

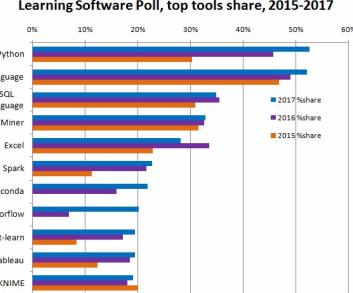
is the way to go when integrating with web apps is important.

Q: Who is winning at the moment?

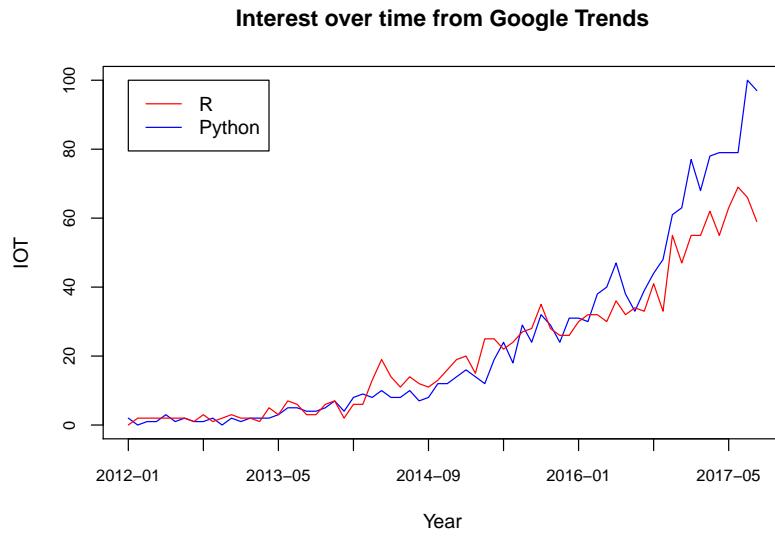
2017 IEEE top programming languages

1. Python	🌐	💻	100.0
2. C	💻	⌚	99.7
3. Java	🌐	💻	99.4
4. C++	💻	⌚	97.2
5. C#	🌐	💻	88.6
6. R	🌐	💻	88.1
7. JavaScript	🌐	💻	85.5
8. PHP	🌐	💻	81.4
9. Go	🌐	💻	76.1
10. Swift	💻	🌐	75.3

KDnuggets Analytics, Data Science, Machine Learning Software Poll, top tools share, 2015-2017



Often R is used to conduct statistical analysis, graph data, and inspect large data sets, and Python is used for machine learning/mining algorithms that are only a portion of a large project.

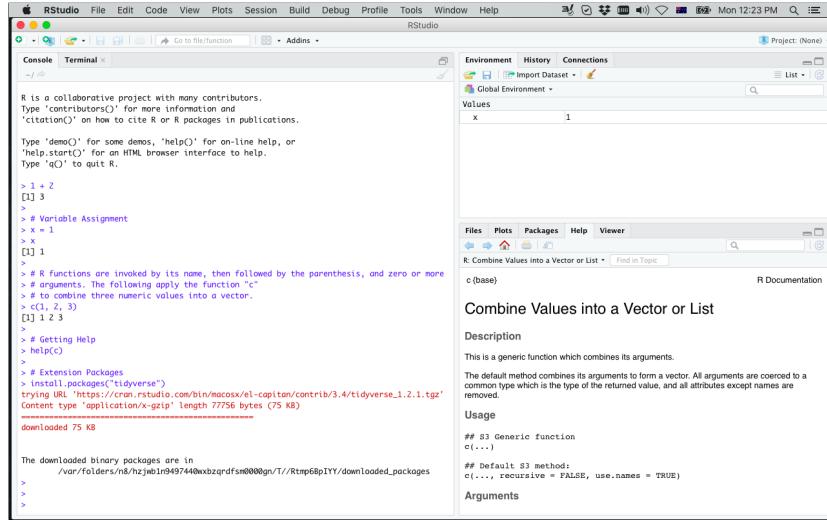


Q: Why using R in this course?

Numbers, IOT, represent search interest relative to the highest point on the chart for the given region and time. A value of 100 is the peak popularity for the term. A value of 50 means that the term is half as popular. A score of 0 means there was not enough data for this term.

- New to data science and statistics in general.
- Working independently
- More resources when comes to regression analysis

- R can be downloaded from [HERE](#)
- RStudio is an IDE for R, and can be downloaded from [HERE](#)



IDE stands for integrated development environment.

## 1.2.2 Data Format

### Data Type

- In contrast to C, R sets the type based on the given value or expression.

```
> 3.14      # numeric, double-precision real
```

```
[1] 3.14
```

```
> TRUE      # logical, TRUE/FALSE
```

```
[1] TRUE
```

```
> "Hello" # Character
```

```
[1] "Hello"
```

```
> is.logical(TRUE); is.logical("TRUE")
```

```
[1] TRUE
[1] FALSE
```

This setup is similar to Python.

- Other data type exists, but often not explicitly specified or used

```
> 7L          # integer
```

```
[1] 7

> is.integer(7L); is.integer(7)
[1] TRUE
[1] FALSE

> 2+3i                                # complex
[1] 2+3i

> sqrt(-1); sqrt(as.complex(-1))      # R in R
[1] NaN
# Warning message:
# In sqrt(-1) : NaNs produced
[1] 0+1i
```

NaN stands for Not a Number

## Data Structure

- Assignment

```
> x = 1

> x <- 1

> 1 -> x

note all of above statements store the value 1 under the name x.

> y = TRUE
> z = "TRUE"

> class(x); class(y); class(z)      # Object Classes
[1] "numeric"
[1] "logical"
[1] "character"
```

The “=” assignment is preferred when specifying names.

- Vector

```
> c(1, 2, 3)
```

```
[1] 1 2 3
```

```
> x.vec = c(.Last.value, x)      # Special Variable  
> x.vec
```

```
[1] 1 2 3 1
```

```
> 1:12                      # Sequence
```

```
[1] 1 2 3 4 5 6 7 8 9 10 11 12
```

Q: What do you think the result of the following statement is?

```
> v.vec = 5:-5
```

```
> v.vec
```

```
[1] 5 4 3 2 1 0 -1 -2 -3 -4 -5
```

- A more general sequence can be generated

```
> y.vec = seq(  
+   from = 1,                  # starting  
+   by = 1,                    # increment  
+   length.out = 16)          # number of elements
```

Q: What do you think the result of the following statement is?

```
> tmp = seq(as.Date('2018-09-10'),  
+           by = 14, length.out = 7)
```

```
> tmp[1:3] # assess elements of a vector by indexing
```

```
[1] "2018-09-10" "2018-09-24" "2018-10-08"
```

```
> lab = tmp[-2] # deselecting an element  
>  
> lab
```

```
[1] "2018-09-10" "2018-10-08" "2018-10-22"  
[4] "2018-11-05" "2018-11-19" "2018-12-03"
```

## Watch out for the recycling rule in R

Q: What do you think the result of the following statement is?

```
> c(1, 2, 3, 4) + c(1, 2)
```

### Recycling Rule

Vectors occurring in the same expression need not all be of the same length. If they are not, the value of the expression is a vector with the same length as the longest vector which occurs in the expression. Shorter vectors in the expression are recycled as often as need be (perhaps fractionally) until they match the length of the longest vector. In particular a constant is simply repeated.

---

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}$$

- Categorical Variable

```
> z.vec = c("hard",
+           "really-hard",
+           "extremely-hard",
+           "bite-your-head-off-hard",
+           "extremely-hard", "really-hard", "hard")

> z.fac = factor(z.vec, order = TRUE,
+                  levels = c(
+                    "hard",
+                    "really-hard",
+                    "extremely-hard",
+                    "bite-your-head-off-hard"))

> class(z.fac); mode(z.fac)      # The storage mode
```

```
[1] "ordered" "factor"
[1] "numeric"
```

- By running the factor statement, R has assigned integers to each level

```
> str(z.fac)          # Structure
```

```
Ord.factor w/ 4 levels "hard" <"really-hard" <...:
 1 2 3 4 3 2 1
```

```
> z.fac[1] >= z.fac[2]    # Comparison of 1st & 2nd
```

```
[1] FALSE
```

```
> table(z.vec)
```

z.vec	
bite-your-head-off-hard	1
extremely-hard	2
hard	2
really-hard	2

```
> table(z.fac)
```

z.fac	
hard	2
really-hard	2
extremely-hard	2
bite-your-head-off-hard	1

Q: What happens when you mix types inside a vector?

```
> x = 1; y = TRUE; z = "TRUE";  
> u.vec = c(x,y); class(u.vec)  
> u.vec = c(x,z); class(u.vec)  
> u.vec = c(x,z.fac); class(u.vec)  
> u.vec = c(y,z.fac); class(u.vec)  
> u.vec = c(y,z.fac); class(u.vec)
```

- R does so-called implicit coercion to mixed types, the coercion rule goes

logical → integer → numeric → complex → character

```
> u.vec = c(x,y); class(u.vec)
```

```
[1] "numeric"
```

```
> u.vec = c(x,z); class(u.vec)
```

```
[1] "character"
```

logical →  $\underbrace{\text{factor}}_{\text{integer}}$  → numeric → complex → character

- R effectively assigns an integer to each level of a factor,

```
> u.vec = c(x,z.fac); class(u.vec)
```

```
[1] "numeric"
```

```
> u.vec = c(y,z.fac); class(u.vec)
```

```
[1] "integer"

> u.vec = c(z,z.fac); class(u.vec)
[1] "character"
```

- Matrix

```
> A = matrix(
+     y.vec,                      # the data elements
+     nrow = 4,                    # number of rows
+     byrow = TRUE)               # fill matrix by rows
```

- Matrices are special vectors in R.

```
> class(A)
[1] "matrix"

> mode(A)
[1] "numeric"

> attributes(A)

$dim
[1] 4 4
```

- A matrix is stored as a vector with a dimension specification.

Matrices are a special vectors in R. They are stored as vector with a dimension specification.

- Indexing

```
> A
      [,1]  [,2]  [,3]  [,4]
[1,]    1     2     3     4
[2,]    5     6     7     8
[3,]    9    10    11    12
[4,]   13    14    15    16

> A[2,4]
[1] 8

> A[3,]
```

```

[1] 9 10 11 12

> A[c(1,3),c(2,4)]
     [,1] [,2]
[1,]    2    4
[2,]   10   12

> # Empty matrix
> B = matrix(nrow = 2, ncol = 3)
>
> # Single indexing in matrix
> B[1:5] = c("a", "p", "p", "l", "e")
>
> # Default is to fill B by row
> B
     [,1] [,2] [,3]
[1,] "a"  "p"  "e"
[2,] "p"  "l"  NA

> B[6] = 17
>
> B # No mixed types in matrix, coerced accordingly
     [,1] [,2] [,3]
[1,] "a"  "p"  "e"
[2,] "p"  "l"  "17"

```

- List

```

> xyzlist = list(x.vec=x.vec, y=y, z.fac=z.fac)
> xyzlist
$x.vec
[1] 1 2 3 1

$y
[1] TRUE

$z.fac
[1] hard
[2] really-hard
[3] extremely-hard
[4] bite-your-head-off-hard
[5] extremely-hard
[6] really-hard
[7] hard
4 Levels: hard < ... < bite-your-head-off-hard

```

- List is a special vector, each element of which can be a different class.

```
> class(xyzlist)
[1] "list"

> attributes(xyzlist)
$names
[1] "x.vec"   "y"        "z.fac"

> class(xyzlist$x.vec); class(xyzlist$y)
[1] "numeric"
[1] "logical"

> xyzlist$x.vec[1]; xyzlist[[1]][1]    # 1st of 1st
[1] 1
[1] 1
```

Q: What do you think the result of the following statement is?

```
> xyzlist[["y"]]; xyzlist[[y]]
● Data frame

> # Vector by crated by replication statement
> manufacturer = rep("audi", 6)

> displ = c(1.80, 1.80, 2.00, 2.80, 3.10, 1.80)
> cyl = as.integer(c(rep(4,4), rep(6,2)))
> cty = c(18, 21, 20, 21, 16, 18)

> mpg.df = data.frame(manufacturer, displ, cyl, cty)
> attributes(mpg.df)

$names
[1] "manufacturer"    "displ"      "cyl"       "cty"

$row.names
[1] 1 2 3 4 5 6

$class
[1] "data.frame"
```

- `displ` stands for engineer displacement/size, volume of all the pistons inside the cylinders.
- `cyl` stands for the number of cylinders.
- `cty` miles per gallon in city

- For this tiny data set, asking R to display it explicitly makes sense

```
> mpg.df
```

	manufacturer	displ	cyl	cty
1	audi	1.8	4	18
2	audi	1.8	4	21
3	audi	2.0	4	20
4	audi	2.8	4	21
5	audi	3.1	6	16
6	audi	1.8	6	18

```
> mpg.df[["cty"]]; mpg.df$cty[5]; mpg.df[[4]][5]
```

[1] 18 21 20 21 16 18
[1] 16
[1] 16

- Notice the indexing is very similar to list indexing.

```
> is.data.frame(mpg.df)
```

[1] TRUE
----------

```
> is.list(mpg.df)
```

[1] TRUE
----------

- Notice data frame can be indexed as if they are matrices, but...

```
> mpg.df[3,1]
```

[1] audi
Levels: audi

```
> is.matrix(mpg.df)
```

[1] FALSE
-----------

Q: What is the difference between list and data.frame?

Data frames are lists as well, but they have a few restrictions:

- All elements of a data frame are vectors
- All elements of a data frame have an equal length
- Cannot have the same name for two different variables

So it is like a matrix that has more than numerical columns.

## Conditional and Repetitive Execution

```
• > if (is.matrix(mpg.df)) {  
+   print("Data frame and Matrix are the same in R")  
+ } else {  
+   print("A data frame is not a matrix")  
+ }  
  
[1] "A data frame is not a matrix"  
  
> is.data.frame(mpg.df) & is.list(mpg.df) # and  
[1] TRUE  
  
> is.data.frame(mpg.df) | is.matrix(mpg.df) # or  
[1] TRUE  
  
> ! is.matrix(mpg.df) # not  
[1] TRUE
```

Of course, you need to know the above.

- The following functions are often useful

```
> x = 1:10  
> all(x>0)  
  
[1] TRUE  
  
> any(x>9)  
  
[1] TRUE  
  
> x = 1:10; (x = ifelse(x > 5, x, -x))  
  
[1] -1 -2 -3 -4 -5  6  7  8  9 10
```

Q: What is s after the following for loop statement?

```
> s = 0  
> for (eps in x) {           # Repetitive execution  
+   s = s + eps  
+ }
```

- Notice R does not need to use an integer loop variable.

s = 25

```

> myfunc =
+   function(x) {      # Define myfunc
+     s = 0
+     for (eps in x) {
+       if (eps <= 0) {
+         next          # jump to the end of the loop
+       }
+       s = s + eps
+       if (s > 20) {
+         break        # stop the loop
+       }
+     }
+     s               # output value
+   }

```

Q: What do you think the result of the following statement is?

```
> x = 1:10; x = ifelse(x > 5, x, -x); myfunc(x)
```

## 2 Simple Linear Regression

### 2.1 Overview

- Regression analysis is about statistical models that involve only one dependent variable  $Y$

and one or more independent variables  $X_j$ .

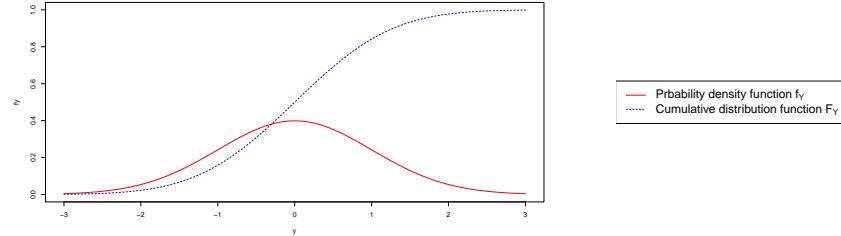
Q: What is the difference between a statistical model and mathematical model?

- It focuses on modelling the distribution of the dependent variable

$$f_Y(y)$$

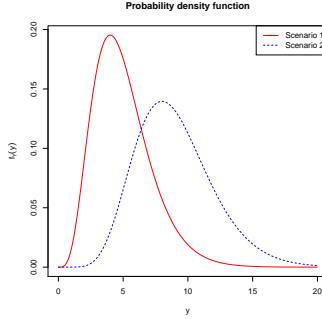
using the independent variables  $X_j$ .

- Recall the distribution of a random variable is usually defined by



It summarises relationships between variables. It is a special kind of mathematical model.

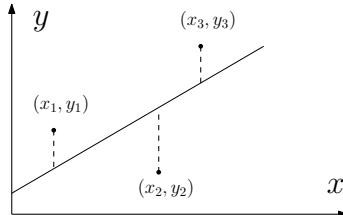
- Non-deterministic, that is, instead of assigning a specific value for some of those variables, it will have probability distributions.
- It describes a *simplified* version of the data-generating process.
- Depending on the values that the independent variables take,



$Y$  may follow a different distribution. e.g.

the distribution of height depends on gender

- Roughly speaking, regression is about finding a rule of picking distributions for  $Y$  from a space of infinitely many distributions that agrees with the data.
- This view of regression models is probably different from your understanding



- You will see that the two views are equivalent, but one is easier to generalise.
- Regression models usually attempt to address one of two questions:
  - **Explaining** the dependencies between

$\underbrace{\text{the explanatory variables}}_{\text{the independent variables}}$       and       $\underbrace{\text{the response}}_{\text{the dependent variable}}$

- **Predicting** the range of response values given a set of values for

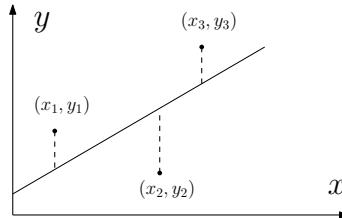
$\underbrace{\text{the predictors}}_{\text{the independent variables}}$

- Independent variables
- Explanatory variables
- predictors
- regressors

- A simple linear model, where both  $X$  and  $Y$  are continuous

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

is particularly easy to understand and implement



Q: Can you recall how to find the best line? In other words, the estimated

$$\beta_0 \quad \text{and} \quad \beta_1$$

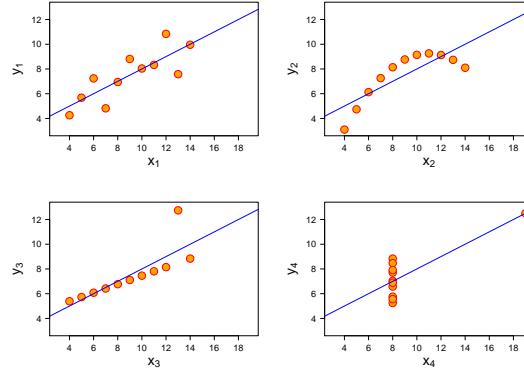
- It is even not hard to imagine how one would proceed with finding

$$f_Y$$

$$b_1 = \frac{\sum_i^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_i^n (x_i - \bar{x})^2}$$

$$b_0 = \bar{y} - b_1 \bar{x}$$

Q: But why a linear model? What is exactly complicated with nonlinearity?



The datasets have approximately the same least squares line, as well as nearly identical means, standard deviations, and correlations.

- Note the least squares principle can also be used for

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \varepsilon_i \quad \text{or} \quad y_i = \frac{\beta_1 x_i}{\beta_2 + x_i} + \varepsilon_i$$

To understand why the linear model, we consider one more approach.

- Consider a random variable  $Y$  that follows some known distribution.

$$f_Y$$

Q: Suppose we would like to predict the value of  $Y$ . What is the "best" guess?

- Let us denote our guess by

$$m$$

Q: What is a sensible and common way to measure the how good  $m$  is ?

- If we don't care about positive more than negative errors, then

$$\text{MSE}(m) = \mathbb{E}[(Y - m)^2]$$

is the traditional choice, which can be decomposed into

$$\text{MSE}(m) = \text{Var}[Y] + (\mathbb{E}[Y] - m)^2$$

Q: Can you recall why the above is true and what does it mean?

MSE stands for mean square error. And

$$\begin{aligned} \mathbb{E}[(Y - m)^2] &= \mathbb{E}[(Y - \mathbb{E}[Y] + \mathbb{E}[Y] - m)^2] \\ &= \mathbb{E}[(Y - \mathbb{E}[Y])^2 + 2(Y - \mathbb{E}[Y])(\mathbb{E}[Y] - m) + (\mathbb{E}[Y] - m)^2] \\ &= \mathbb{E}[(Y - \mathbb{E}[Y])^2] + 2(\mathbb{E}[Y] - m)\mathbb{E}[Y - \mathbb{E}[Y]] + \mathbb{E}[(\mathbb{E}[Y] - m)^2] \\ &= \text{Var}[Y] + 0 + (\mathbb{E}[Y] - m)^2 \\ &= \text{Var}[Y] + (\mathbb{E}[Y] - m)^2 \end{aligned}$$

Q: What is bias in statistics?

$$\text{Bias}(\hat{\theta}, \theta) = \mathbb{E}[\hat{\theta} - \theta]$$

where  $\hat{\theta}$  is often an estimator, and  $\theta$  is a population parameter, thus fixed.

- So we have the simplest form of the bias-variance decomposition,

$$\text{MSE}(m) = \text{Var}[Y] + (\mathbb{E}[Y] - m)^2$$

which confirms the expected value is the best single number for predicting  $Y$

$$m = \mathbb{E}[Y] = \int_{-\infty}^{\infty} y f_Y(y) dy$$

- Note changing our prediction,  $m$ , does nothing to the true distribution of  $Y$ .

When other criterion is used, similar things can be done but we might need a bit more calculus

- Now imagine we have two random variables, say

$$X \quad \text{and} \quad Y$$

- Suppose we also know  $X$  in the sense we know

$$f_X,$$

and we would like to use this knowledge to improve our prediction of  $Y$ .

Q: What is the “best” guess  $m$  now?

- It is clear that our guess should be a function of  $x$ ,

$$m(x)$$

- Again we would like to make MSE as small as possible

$$\mathbb{E} [(Y - m(X))^2]$$

- We use conditional expectations to reduce

$$\mathbb{E} [(Y - m(X))^2]$$

back to the problem we already solved

$$\mathbb{E} [(Y - m(X))^2] = \mathbb{E}_X [\mathbb{E}_Y [(Y - m(X))^2 | X]]$$

- For each possible value  $x$ , the best value is just the conditional mean

$$m(x) = \mu(x) = \mathbb{E}[Y | X = x]$$

which is known as the true [regression function](#).

- Now we have reached the part that requires us to use a linear approximation,

$$\begin{aligned} f(x) &= f(a) + f'(a)(x - a) + \dots \\ &= f(a) - f'(a) + f'(a)x + \dots \end{aligned}$$

- We restrict the regression function to have the linear form

$$m(x) = \beta_0 + \beta_1 x$$

and find the choice of  $\beta_0$  and  $\beta_1$  so that it is the best under this restriction.

- This is achieved by simply minimising MSE with respect to  $\beta_0$  and  $\beta_1$

$$\text{MSE}(\beta_0, \beta_1) = \mathbb{E} [(Y - (\beta_0 + \beta_1 X))^2]$$

Q: Can you see why the following choice is the best?

$$\beta_0 = \mathbb{E}[Y] - \beta_1 \mathbb{E}[X] \quad \text{where} \quad \beta_1 = \frac{\text{Cov}[X, Y]}{\text{Var}[X]}$$

- Notice means play no role in  $\beta_1$ , only the variance and covariance mattered.

It is clear from the law of total expectation

$$\mathbb{E}[Y] = \mathbb{E}_X[\mathbb{E}_Y[Y | X]] = \mathbb{E}_X[\beta_0 + \beta_1 X] = \beta_0 + \beta_1 \mathbb{E}[X]$$

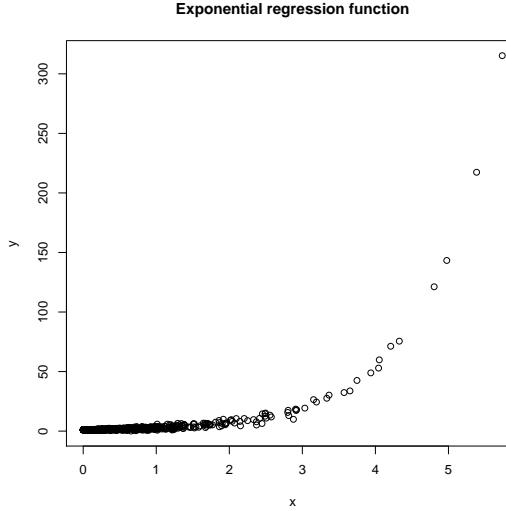
However,  $\beta_1$  is less clear, using the above

$$\begin{aligned} \text{MSE}(\beta_1) &= \mathbb{E}[(Y - (\mathbb{E}[Y] - \beta_1 \mathbb{E}[X] + \beta_1 X))^2] \\ &= \mathbb{E}[(Y - \mathbb{E}[Y])^2 + \beta_1^2 \mathbb{E}[(X - \mathbb{E}[X])^2] \\ &\quad + 2\beta_1 \mathbb{E}[(Y - \mathbb{E}[Y])(X - \mathbb{E}[X])]]) \\ &= \text{Var}[Y] + 2\beta_1 \text{Cov}[X, Y] + \beta_1^2 \text{Var}[X] \end{aligned}$$

This means a big slope  $\beta_1$  leads  $X$  and  $Y$  fluctuate together for a fixed  $\text{Var}[X]$ .

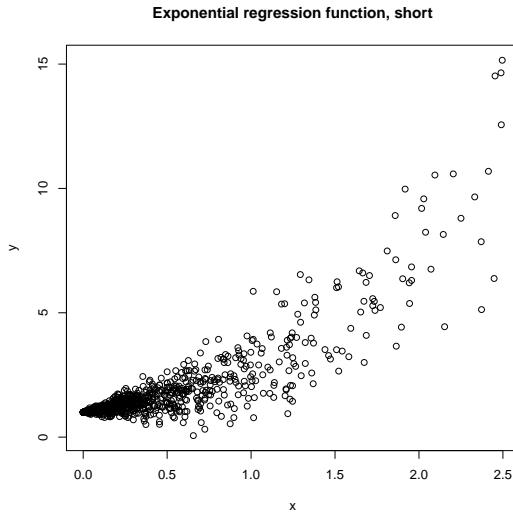
- It is the best amongst linear predictors, but it can be far from the truth. e.g.

$$\mathbb{E}[Y | X] = e^x$$



- We did not assume the relationship between  $X$  and  $Y$  is linear, but

$$\mu(x) = \mathbb{E}[Y | X] = \mu(x_0) + (x - x_0)\mu'(x_0) + \frac{1}{2}(x - x_0)^2\mu''(x_0) + \dots$$



- So using linear approximation as a justification for teaching and using linear model globally is a weak argument.
- So it was done in the past mainly for computational reasons.
- That said, linear models are very useful for illustrative purposes.

- Recall regression is about finding a rule of picking distributions for  $Y$  from a space of infinitely many distributions that agrees with the data.
- If we know everything about  $X$  and  $Y$ , then the “best” guess is

$$m = \beta_0 + \beta_1 x$$

where

$$\beta_0 = \mathbb{E}[Y] - \beta_1 \mathbb{E}[X] \quad \text{and} \quad \beta_1 = \frac{\text{Cov}[X, Y]}{\text{Var}[X]}$$

Q: What do we do when we don't have  $f_X$  and  $f_Y$ ?

- It is natural to consider the unbiased estimates for the expectations, variance and covariance to obtain estimates for  $\beta_0$  and  $\beta_1$

$$b_0 = \bar{y} - b_1 \bar{x} \quad \text{where} \quad b_1 = \frac{c_{xy}}{s_x^2}$$

Q: What do you notice?

This is the same as OLS estimates.

## 2.2 Basics

Q: What does it mean when we say a sequence of random variables

$$\{X_1, X_2, \dots, X_n\}$$

is **independent and identically distributed** (i.i.d.)?

- We will treat the data you have on the response,

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

as just one realisation of the sequence

$$\{Y_1, Y_2, \dots, Y_n\}$$

Q: Does this sequence necessarily satisfy the i.i.d. condition?

$$\{Y_1, Y_2, \dots, Y_n\}$$

- Until the end of part I, we will only consider independent  $Y_i$ .
- It was said that some men are more likely to have daughters than others e.g.

a deep-sea diver, a fighter pilot and a heavy smoker

- If you prefer sons, easy! just become US president.



### The Facts

- The 45 US presidents from George Washington to Donald Trump have had a total of 158 children, comprising 91 sons and only 67 daughters.

about 1.4 sons for every daughter

- Two studies of deep-sea divers revealed that the men had a total of 190 children, comprising 65 sons and 125 daughters:

about 1.9 daughters for every son

- Let consider testing the two hypotheses one at time.

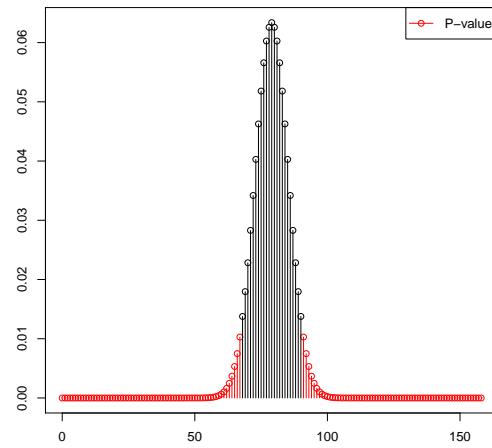
```
> # President -----
> rm(list = ls()) # Clean environment
> # Suppose the chance is 50-50
> # for a president as well
> p = 1/2
>
> # Total number of trials
> n = 158
>
> # All possible values 0 to 158 daughters
```

```

> x = 0:n
>
> # Binomial density/mass function
> fX = dbinom(x, size = n, prob = p)
> # Produce a plot of the density function
> # With the p-value region highlighted in red

```

- Recall  $P$ -value is the probability of getting a result at least as extreme as 65 daughters under  $H_0 : p = 0.5$ .



```

> p.lowerTail = pbinom(67, size = n, prob = p)
> 2 * p.lowerTail          # P-value
[1] 0.06694264

> # Lower tail
> tmpL = fX[x <= 67] # As extreme as 67

> # Upper tail probability
> x.upperTail = 1 + qbinom(
+   p.lowerTail, size = n, prob = p,
+   lower.tail = FALSE)

> tmpU = fX[x >= x.upperTail]

> # Middle
> M = x[ x>67 & x<x.upperTail]
> tmpM = fX[ x>67 & x<x.upperTail ]

```

```

> # Actual plotting
> plot(x, fX, type = "n", xlab = "", ylab = "")

> lines(0:67, tmpL, type = "h", col = "red")

> points(0:67, tmpL, col = "red")

> lines(x.upperTail:n, tmpU,
+        type = "h", col = "red")
> points(x.upperTail:n, tmpU, col = "red")

> lines(M,tmpM, type = "h")
> points(M, tmpM)

> legend("topright", legend = "P-value",
+        col ="red", lty = 1, pch = 1)

```

- If we do the same to

```

> # Deep-sea divers -----
> 2 * pbinom(65, size = 190, prob = 0.5)

```

```
[1] 1.603136e-05
```

- This is a little more than one chance in 100 thousand.
- The following is what we can say:

We conclude that it is extremely unlikely that this observation could have occurred by chance, if the deep-sea divers had equal probabilities of having sons and daughters. The data are not compatible with  $H_0$ .

Q: How about the president case?

```
> 2 * p.lowerTail
```

```
[1] 0.06694264
```

- For the president case, the following is what we can say:

We conclude that there is no real evidence that presidents are more likely to have sons than daughters or vice versa. The observations are compatible with the possibility that there is no difference.

Q: Does this mean presidents are equally likely to have sons and daughters?

No, the observations are also compatible with the possibility that there is a difference. We just don't have enough evidence either way.



Q: Back to the deep-sea divers case, does p-value of 1.603136e-05 say about the actual probability of a diver having a daughter instead of a son?

- Let  $p$  denote the probability of a deep-sea diver has a daughter, and  $X$  be the number of daughters out of 190 children that deep-sea divers have

$$X \sim \text{Binomial}(190, p)$$

Q: Which single value would you say  $p$  is equal to?

Common-sense suggests to use

$$p = \frac{\text{number of daughters observed}}{\text{total number of children}} = \frac{125}{190} = 0.658$$

### Definition

The process of using data to suggest a value for a parameter is called [estimation](#).

---

- The value suggested is called the [estimate](#) of the parameter. It is clear that the estimate should depend on the data.
- An [estimator](#) is a function of the data, that gives estimates of the parameter

$$\hat{\theta} = h(X)$$

- Notice  $\hat{\theta}$  is also random, its randomness is inherited from the data.
- The distribution for  $\hat{\theta}$  is called the [sampling distribution](#) of the estimator.
- Consider the following

```
> # Function -----
> myp_func = function(p){
+   n = 190
+   p125 = pbinom(125, size = n, prob = p)
+   p124 = pbinom(124, size = n, prob = p)
+   p125 - p124 # dbinom(125, size = n, prob = p)
+ }
```

```
> myp_func(p = 0.5); myp_func(p = 0.6)
```

```
[1] 3.972689e-06
[1] 0.01576121
```

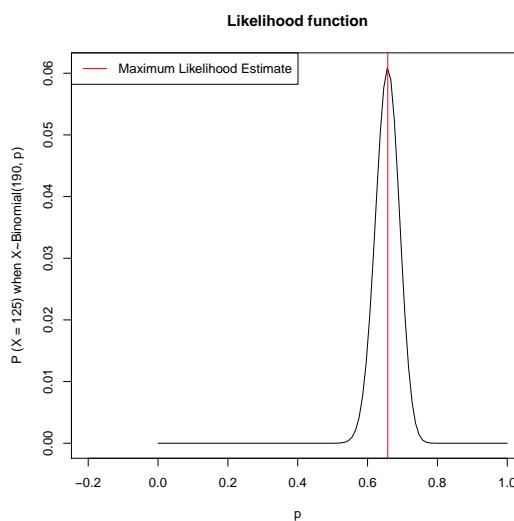
- Having  $p = 0.6$  still looks unlikely, but it is almost 4000 times more likely.

```
> n
```

```
[1] 158
```

- It is not hard to understand the best choice of  $p$  in this case is MLE.

Q: What is MLE?



```
> # many functions in R can take vector input
> pbinom(125, size = n, prob = c(0.5, 0.6))
[1] 0.9999960 0.9567501
```

Thus the tidy function can actually take vector input as well

```
> myp_func = function(p){
+   n = 190
+   p125 = pbinom(125, size = n, prob = p)
+   p124 = pbinom(124, size = n, prob = p)
+   p125 - p124 # dbinom(125, size = n, prob = p)
+ }

> # Create a vector contains a sequence of numbers
> pvec = seq(0, 1, length.out = 100)
>
> lvec = myp_func(pvec)
```

```

> plot(pvec, lvec,
+       type = "l",
+       xlab = "p",
+       ylab = "P(X = 125) when X~Binomial(190, p)",
+       main = "Likelihood function",
+       xlim = c(-0.2,1)
+     )

> phat = 125/190 # common-sense

> abline(v = phat, col = "red") # MLE
>
> legend("topleft",
+         legend = "Maximum Likelihood Estimate",
+         lty = 1, col = 2)

```

Q: Is there any way to attach some kind of “strength” to our estimate?

$$\hat{p} = \frac{125}{190}$$

- Because there clearly is a difference between the following scenarios
  - 125 daughters out of 190 children
  - 1250 daughters out of 1900 children
 despite of having the same MLE  $\hat{p} = \frac{25}{28}$ .
- I have been hiding things from you.

```

> # confidence interval: Clopper-Pearson
> binom.test(125, n = 190, p = 0.5)

```

```

Exact binomial test

data: 125 and 190
number of successes = 125, number of trials = 190, p-value = 1.603e-05
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
0.5857408 0.7250337
sample estimates:
probability of success
0.6578947

```

- When people say

We are 95% confident that the true probability of having a daughter is between (0.586, 0.725) for a male deep-sea diver.

they actually meant

The method used to obtain the interval

$$(0.586, 0.725)$$

will contain the true probability 95% of time if it is to be repeated.

```

> install.packages("binom")
> binom::binom.confint(125, n = 190)

      method   x   n     mean    lower    upper
1 agresti-coull 125 190 0.6578947 0.5878351 0.7216963
2 asymptotic 125 190 0.6578947 0.5904374 0.7253521
3 bayes 125 190 0.6570681 0.5895875 0.7236296
4 cloglog 125 190 0.6578947 0.5857332 0.7205325
5 exact 125 190 0.6578947 0.5857408 0.7250337
6 logit 125 190 0.6578947 0.5876377 0.7218476
7 probit 125 190 0.6578947 0.5882529 0.7225372
8 profile 125 190 0.6578947 0.5886447 0.7229128
9 lrt 125 190 0.6578947 0.5886470 0.7229437
10 prop.test 125 190 0.6578947 0.5852078 0.7240821
11 wilson 125 190 0.6578947 0.5879068 0.7216245

```

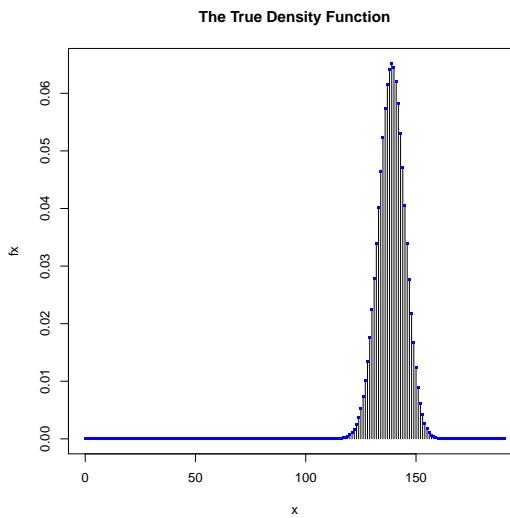
Q: Can you understand what the following piece of R code is about?

```

> rm(list = ls())
> # Simulation study on CI, estimation, and testing
> num = 1e3      # number of repetition
> n = 190        # number of trial
> # Generate a true parameter use uniform(0,1)
> p = runif(1)
> p
[1] 0.730913

> x = 0:190
> fx = dbinom(x, size = n, prob = p)
>
> plot(x, fx, type = "h",
+       main = "The True Density Function")
>
> points(x, fx, pch = ".", col = "blue", cex = 3)

```



- Let us indicate unusual events under the true  $p$  on the graph.

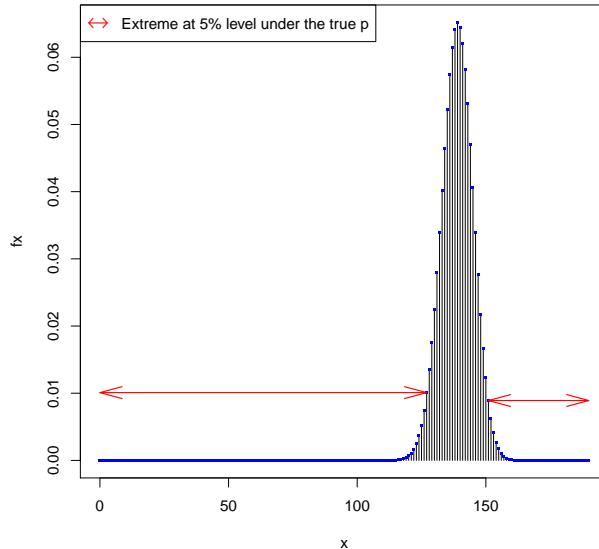
```
> # Indicate unlikely events at 5% level under true
> x_unlikely_lower =
+   qbinom(0.025, size = n, prob = p)

> yL = dbinom(x_unlikely_lower,
+               size = n, prob = p)

> arrows(x0 = 0, y0 = yL,
+         x1 = x_unlikely_lower, y1 = yL,
+         angle = 15, col = 2, code = 3, lwd = 1.2)

> x_unlikely_upper =
+   qbinom(0.025, size = n, prob = p,
+           lower.tail = FALSE)
> yU = dbinom(x_unlikely_upper, size = n, prob = p)
> arrows(x0 = x_unlikely_upper, y0 = yU,
+         x1 = n, y1 = yU,
+         angle = 15, col = 2, code = 3, lwd = 1.2)
```

The True Density Function



```
> legend("topleft", lwd = NA, lty = NA, legend =
+         "Extreme at 5% level under the true p",
+         x.intersp = 0)
```

```

> par(font = 5) #change font to get arrows

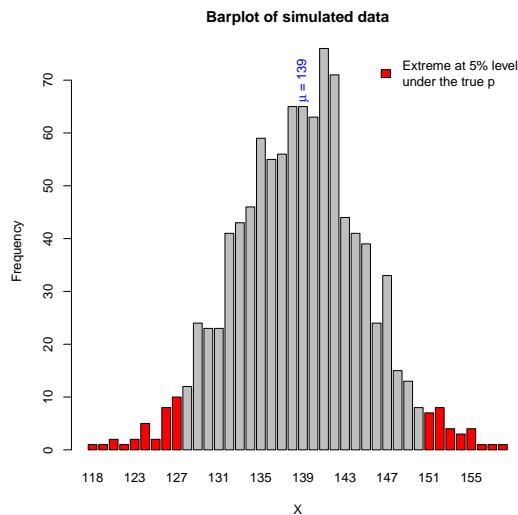
> legend("topleft", legend = NA,
+         lwd = 1, lty = NA,
+         pch = 171,
+         col = 2,
+         bty = "n",
+         pt.cex=1.3)

> par(font = 1) #change back to common font

```

Q: A **confidence interval** is a type of interval estimate that may fail with a given probability, what exactly is this probability? What is actually to be repeated?

```
> X = rbinom(num, size = n, prob = p)
```



```

> # Sorting the data into a table of counts
> counts = table(X)

> nonzero.counts = names(counts)
> head(nonzero.counts)

[1] "118" "120" "121" "122" "123" "124"

```

---

```

> tmp = as.character(x_unlikely_lower)
> xL.index = which(nonzero.counts == tmp)

```

```

> tmp = as.character(x_unlikely_upper)
> xU.index = which(nonzero.counts == tmp)

> tmp = length(counts)
> index.vec = rep(1, tmp)

> index.vec[1:xL.index] = 2
> index.vec[xU.index:tmp] = 2

> cols.vec = c("grey", "red")[index.vec]

> x.mean = round(n*p)
> xm.index = which(nonzero.counts == x.mean)

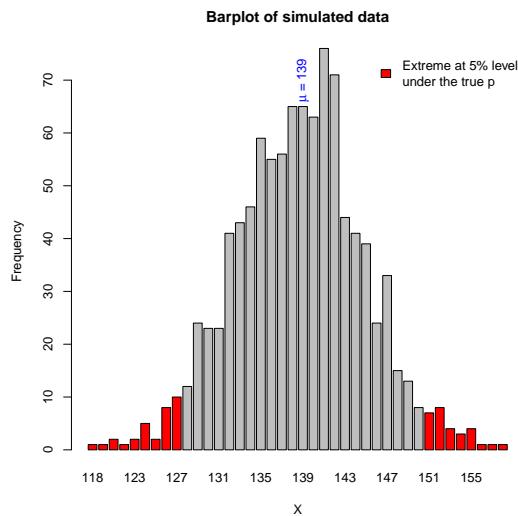
> barpos =
+   barplot(counts, col = cols.vec,
+           xlab = "X", ylab = "Frequency",
+           main = "Barplot of simulated data")

> text(barpos[xm.index],
+       counts[[xm.index]] + 5,
+       bquote(mu ~ "=" ~ .(x.mean)),
+       col = 4, srt = 90)

> legend("topright", legend =
+         "Extreme at 5% level\nunder the true p",
+         fill = 2, bty = "n")

```

Q: What do you think will happen to the C.I. based on those  $x$  in the red bars?



```

> res = binom.test(X[1], n = n, p = p)
>
> res; p

```

---

```

Exact binomial test

data: X[1] and n
number of successes = 124, number of trials = 190, p-value = 0.01733
alternative hypothesis: true probability of success is not equal to 0.7315841
95 percent confidence interval:
0.5803205 0.7200957
sample estimates:
probability of success
0.6526316
[1] 0.7315841

```

---

```

> names(res)

```

---

```

[1] "statistic"      "parameter"      "p.value"
[4] "conf.int"        "estimate"       "null.value"
[7] "alternative"    "method"         "data.name"

```

---

```

> res$p.value

```

---

```

[1] 0.01732913

```

---

```

> res$conf.int[1:2]

```

---

```

[1] 0.5803205 0.7200957

```

---

```

> res.df = data.frame(
+   CIL = double(), CIU = double())

> for (i in 1:num){
+   res = binom.test(X[i], n = n, p = p)
+   res.df[i,1:2] = res$conf.int[1:2]
+ }

> head(res.df, 2)

```

---

	CIL	CIU
1	0.5803205	0.7200957
2	0.6184566	0.7544657

---

```

> res.df[1,1] < p

```

---

```

[1] TRUE

```

---

```

> res.df[1,1] < p & res.df[1,2] > p

```

```
[1] FALSE
```

---

```
> n_ci_contain_true =
+   sum(res.df[, 1] < p & res.df[, 2] > p)

> (rate = 1 - n_ci_contain_true/num)
[1] 0.044
```

---

- What does the law of large numbers say?

### Law of Large Numbers

Suppose that  $X_1, X_2, \dots, X_n$  all have the same expected value

$$\mathbb{E}[X_i] = \mu,$$

the same variance

$$\text{Var}[X_i] = \sigma^2,$$

and zero covariance with each other

$$\text{Cov}[X_i, X_j] = 0 \quad \text{where } i \neq j$$

In particular, if the  $X_i$  are i.i.d., then the following holds

$$\bar{X}_n = \frac{1}{n} \sum_{i=1}^n X_i \rightarrow \mu \quad \text{when } n \rightarrow \infty$$


---

- The law of large number is the justification for using the following earlier

$$\hat{\beta}_0 = \bar{Y}_n - \hat{\beta}_1 \bar{X}_n$$

- It is easy to understand intuitively, but it was often misunderstood.
- Consider the following pieces of R code.

```
> # LLN -----
> rm(list=ls())
> n = 1e4 # final sample size
> lambda = 3

> # Consider Poisson random variables
> xpois = rpois(n, lambda)
> xexp = lambda # True mean for Poisson
```

- Let us investigate  $\bar{X}$  as  $n$  increases

```
> # sample size at various stages
> nseq = 1:n

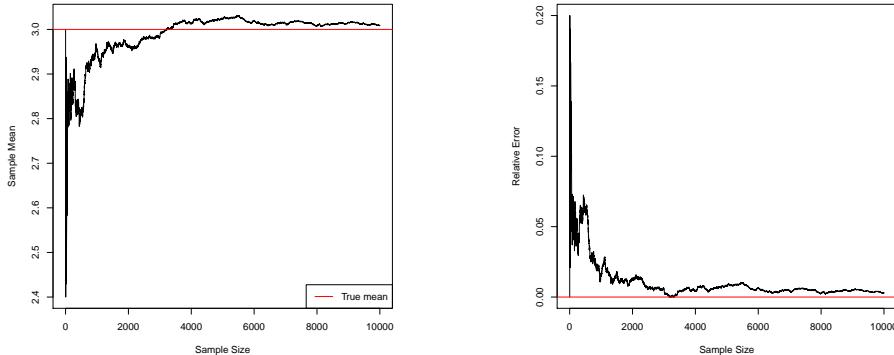
> # sample mean at various stage
> xcbar = cumsum(xpois) / nseq

> # Relative error
> error = abs(xcbar-xexp) / abs(xexp)

> plot(nseq, xcbar, type = "l",
+       xlab = "Sample Size", ylab = "Sample Mean")
>
> abline(h = xexp, col = 2)
>
> legend("bottomright", legend = "True mean",
+        lty = 1, col = 2)

> plot(nseq, error, type = "l",
+       xlab = "Sample Size", ylab = "Relative Error")
>
> abline(h = 0, col = 2)
```

Q: What do you expect to see?



Q: Do you think it is due to not having a large enough sample size?

```
> # A better look at LLN -----
> sample.size.vec = c(5, 10, 50, 100, 500)
> ncases = length(sample.size.vec)           # 5 cases
> num = 1000                                # number of repetitions
> xbar.vec = double()      # x bar for all simulations
> error.vec = double()    # error for all simulations
> n.vec = integer()      # sample size for each case
```

```

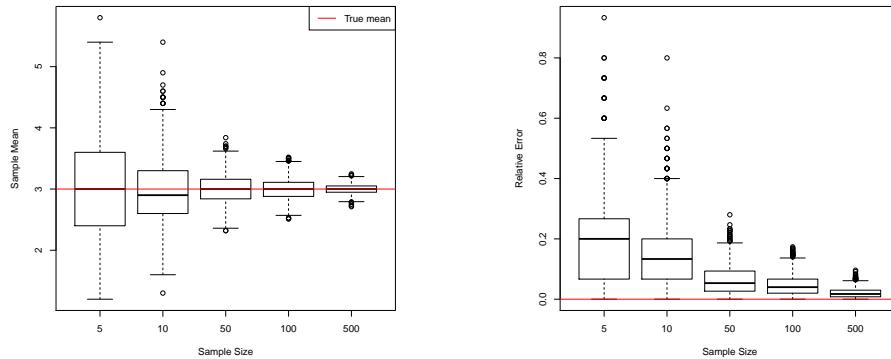
> for(j in 1:ncases){
+   # Current sample size
+   n = sample.size.vec[j]

+   s.vec = double(num) # all x bar for this n
+   e.vec = double(num) # all error for this n

+   # Repeat num number of times
+   for (i in 1:num){
+     x = rpois(n, lambda)
+     s.vec[i] = sum(x) / n
+     e.vec[i] = abs(s.vec[i] - xexp) / abs(xexp)
+   }

+   # Store simulation results
+   xbar.vec = c(xbar.vec, s.vec)
+   error.vec = c(error.vec, e.vec)
+   n.vec = c(n.vec, rep(sample.size.vec[j], num))
+
+ }

```



- Note how we manage to reduce the spread of the distribution to 0 effectively.

$$\lim_{n \rightarrow \infty} \Pr(|\bar{X}_n - \mu| > \varepsilon) = 0 \quad \text{for any } \varepsilon > 0$$

```

> x.df =
+   data.frame(xbar = xbar.vec,
+             error = error.vec, n = n.vec)
>
> boxplot(xbar~n, data = x.df,
+           xlab = "Sample Size",
+           ylab = "Sample Mean")
>

```

```

> abline(h = xexp, col = 2)
>
> legend("topright", legend = "True mean",
+         lty = 1, col = 2)
>
> boxplot(error~n, data = x.df,
+           xlab = "Sample Size",
+           ylab = "Relative Error")
>
> abline(h = 0, col = 2)

```

Q: What does the central limit theorem say?

### Central Limit Theorem

Suppose  $X_1, X_2, \dots, X_n$  are i.i.d. with mean  $\mu$  and variance  $\sigma^2$ , then

$$\sqrt{n} \frac{\bar{X}_n - \mu}{\sigma}$$

becomes more and more likely a standard normal random variable as  $n \rightarrow \infty$

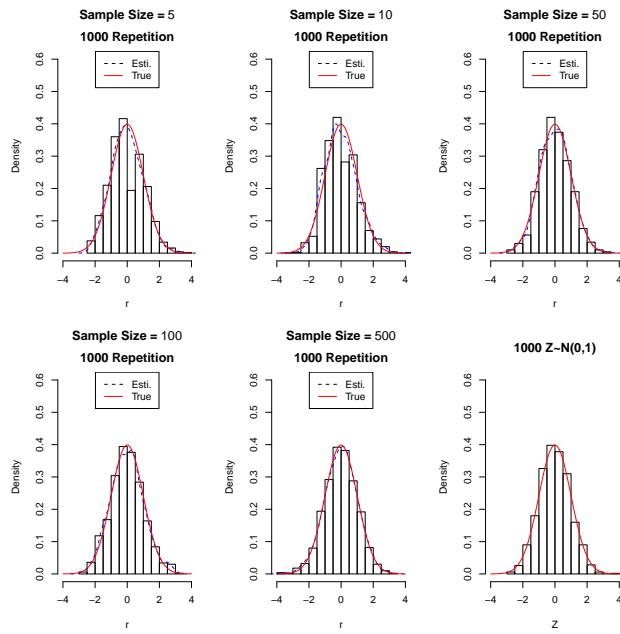
$$Z \sim \text{Normal}(0, 1)$$

in the sense that for every real number  $z$

$$\lim_{n \rightarrow \infty} \Pr \left( \sqrt{n} \frac{\bar{X}_n - \mu}{\sigma} \leq z \right) = \Pr(Z \leq z)$$


---

- Intuitively, it means the sequence of distributions can be better and better described by a normal distribution as  $n$  becomes bigger and bigger.



```
> # CLT -----
> sample.size.vec
```

[1]	5	10	50	100	500
-----	---	----	----	-----	-----

```
> lambda; xexp
```

[1]	3
[1]	3

```
> xvar = lambda # True variance for Poisson

> sqrt.n.vec = sqrt(n.vec) # last for loop

> top = xbar.vec - xexp
> bottom = sqrt(xvar)
> r.vec = sqrt.n.vec * top / bottom

> x.hist = seq(-4, 4, length.out = 100)
> par(mfrow = c(2,3))

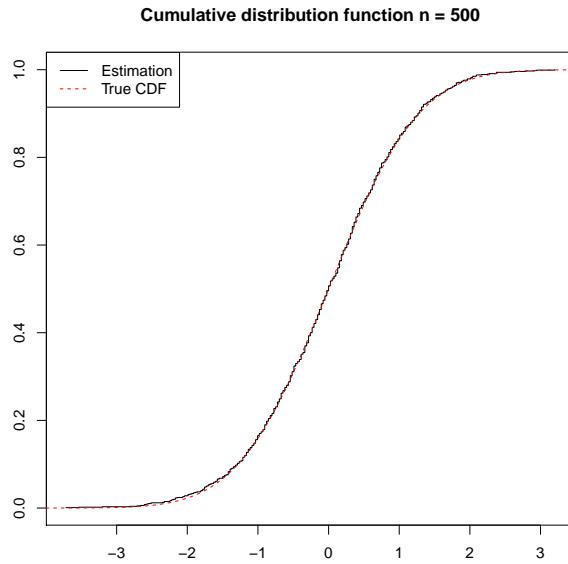
> for (eps in sample.size.vec){
+   ss = r.vec[n.vec == eps] # subset by n
+
```

```

+   tname =                               # title
+   bquote(bold(atop(
+     "Sample Size ="~.(eps), "1000 Repetition")))
+
+   hist(ss, freq = FALSE, xlab = "r", main = tname,
+         ylim = c(0, 0.6), xlim = c(-4, 4))
+
+   # Kernel Density Estimation
+   lines(density(ss), col = 4, lty = 2)
+   # True Density function
+   lines(x.hist, dnorm(x.hist), col = 2, lty = 1)
+
+   legend("top", col = c(1,2), lty = c(2,1),
+          legend = c("Estimation","True "))
}

```

- We can ask R to use sample quantiles to estimate the distribution function.



```

> true.norm = rnorm(num, mean = 0, sd = 1)
>
> hist(true.norm, freq = FALSE,
+       ylim = c(0, 0.6), xlim = c(-4, 4),
+       xlab = "Z", main = "1000 Z~N(0,1)")
>
> lines(x.hist, dnorm(x.hist), col = 2, lty = 1)
>
> par(mfrow = c(1,1))

```

```

> sample_quantile = sort(r.vec[n.vec == 500])
> sample_cdf = (1:num) / num
>
> plot(sample_quantile, sample_cdf, type = "s",
+       xlab = "", ylab = "", main =
+         "Cumulative distribution function n = 500")
> lines(x.hist, pnorm(x.hist), col = 2, lty = 2)
> legend("topleft", lty = c(1, 2), col = c(1, 2),
+         legend = c("Estimation", "True CDF"))

```

### Definition

An estimator is **consistent** if

$$\hat{\theta}_n \rightarrow \theta \quad \text{when } n \rightarrow \infty$$


---

- The law of large numbers states  $\bar{X}_n = \frac{1}{n} \sum_{i=1}^n X_i$  is a **consistent** estimator of

$$\mathbb{E}[X_i] \quad \text{where } X_i \text{ are i.i.d.}$$

as well as being an unbiased estimator of it.

- The central limit theorem states the sampling distribution is asymptotically normal with the variance  $\frac{\sigma^2}{n}$ , thus the **standard error** of  $\hat{\theta} = \bar{X}$ ,

$$\text{SE} = \frac{\sigma}{\sqrt{n}} \rightarrow 0 \quad \text{when } n \rightarrow \infty$$

## 2.3 LSE and MLE

- Consider two continuous random variables  $X$  and  $Y$ , e.g. height and weight.
- Suppose we have obtained  $n$  data points

$$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$$

and would like to construct a simple linear regression model

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

where  $\beta_0$  and  $\beta_1$  are fixed but unknown, and  $\varepsilon$  is the random error term.

- For notational reasons, it is better to treat the observations

$$x_1, x_2, \dots, x_n$$

as realisations of  $n$  random variables, one for each observation

$$X_1, X_2, \dots, X_n$$

- Similarly,  $Y_1, Y_2, \dots, Y_n$  denote random variables for  $y_1, y_2, \dots, y_n$ .
- Note  $\beta_0$  and  $\beta_1$  are unknown to us, thus the realised errors are not observed,

$$e_i = y_i - (\beta_0 + \beta_1 x_i)$$

despite having observed  $x_i$  and  $y_i$  for all  $i$ .

- However, if we have some estimates of  $\beta_0$  and  $\beta_1$ , then we can estimate it by

$$\hat{e}_i = y_i - (b_0 + b_1 x_i)$$

where  $b_0$  and  $b_1$  are estimates of  $\beta_0$  and  $\beta_1$ , respectively.

- The estimate  $\hat{e}_i$  is known as the **residual**, and the term

$$\hat{y}_i = b_0 + b_1 x_i$$

is known as the **predicted/fitted value**.

- Therefore, the observed  $y_i$  is the sum of the two components

$$y_i = \hat{y}_i + \hat{e}_i$$

- Recall either by the least square principle

$$(b_0, b_1) = \underset{(\beta_0, \beta_1)}{\text{Arg min}} \sum_{i=1}^n e_i^2$$

or by MSE criterion and using unbiased sample estimates, we have

$$b_0 = \bar{y} - b_1 \bar{x} \quad \text{and} \quad b_1 = \frac{c_{xy}}{s_x^2}$$

as the estimates of  $\beta_0$  and  $\beta_1$ , respectively.

- In order to determine the properties of the two estimators, i.e.

unbiasedness and consistency

we have to refine our assumptions.

- Three common model assumptions for simple linear regression are:

1. The conditional mean is linear for all  $i$ , i.e.

$$\mathbb{E}[Y_i | X_i = x_i] = \beta_0 + \beta_1 x_i$$

2. The error term

$$\varepsilon_i = Y_i - \beta_0 - \beta_1 X_i$$

has a zero mean and a constant but unknown variance for all  $i$ , i.e.

$$\mathbb{E}[\varepsilon_i | X_i] = 0 \quad \text{and} \quad \text{Var}[\varepsilon_i | X_i] = \sigma^2$$

3. The error terms are uncorrelated with  $X_i$  for all  $i$ , i.e.

$$\text{Cov} [\varepsilon_i, X_i] = 0$$

and are uncorrelated to each other for all  $i \neq j$ , i.e.

$$\text{Cov} [\varepsilon_i, \varepsilon_j] = 0$$

### Definition

The **bias** of an estimator  $\hat{\theta}$  for a fixed unknown parameter  $\theta$  is given by

$$\text{Bias} (\hat{\theta}, \theta) = \mathbb{E} [\hat{\theta} - \theta]$$

and the estimator is said to be **unbiased** if its bias is always zero.

---

- Consider our estimator for  $\beta_1$ ,

$$\hat{\beta}_1 = \frac{\frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2} \quad \text{where} \quad \begin{aligned} \bar{X} &= \frac{1}{n} \sum_{i=1}^n X_i \\ \bar{Y} &= \frac{1}{n} \sum_{i=1}^n Y_i \end{aligned}$$

Q: How can we show it is unbiased?

It can be shown that  $\hat{\beta}_1$  is a random variable in terms of  $X_i$  and  $\varepsilon_i$  for all  $i$ , thus

$$\mathbb{E} [\hat{\beta}_1] = \mathbb{E}_X \left[ \mathbb{E}_{\varepsilon} [\hat{\beta}_1 | X_1 = x_1, \dots, X_n = x_n] \right]$$

Specifically, we will show in a minute that

$$\begin{aligned} \mathbb{E} [\hat{\beta}_1] &= \beta_1 + \mathbb{E} \left[ \frac{\sum_{i=1}^n (X_i - \bar{X}) \varepsilon_i}{W} \right] \quad \text{where} \quad W = \sum_{i=1}^n (X_i - \bar{X})^2 \\ &= \beta_1 + \mathbb{E}_X \left[ \sum_{i=1}^n \frac{X_i - \bar{X}}{W} \mathbb{E}_{\varepsilon} [\varepsilon_i | X_1 = x_1, \dots, X_n = x_n] \right] = \beta_1 \end{aligned}$$

$$\begin{aligned}
\hat{\beta}_1 &= \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^n (X_i - \bar{X})^2} \\
&= \frac{\sum_{i=1}^n (X_i - \bar{X}) \left( \beta_0 + \beta_1 X_i + \varepsilon_i - \frac{1}{n} \sum_{i=1}^n \beta_0 + \beta_1 X_i + \varepsilon_i \right)}{\sum_{i=1}^n (X_i - \bar{X})^2} \\
&= \frac{\sum_{i=1}^n (X_i - \bar{X}) [\beta_1 (X_i - \bar{X}) + (\varepsilon_i - \bar{\varepsilon})]}{\sum_{i=1}^n (X_i - \bar{X})^2} \\
&= \frac{\beta_1 \sum_{i=1}^n (X_i - \bar{X})^2 + \sum_{i=1}^n (X_i - \bar{X})(\varepsilon_i - \bar{\varepsilon})}{\sum_{i=1}^n (X_i - \bar{X})^2} \\
&= \beta_1 + \frac{\sum_{i=1}^n (X_i - \bar{X}) \varepsilon_i - \bar{\varepsilon} \sum_{i=1}^n (X_i - \bar{X})}{\sum_{i=1}^n (X_i - \bar{X})^2} \\
&= \beta_1 + \frac{\sum_{i=1}^n (X_i - \bar{X}) \varepsilon_i - \bar{\varepsilon} (n\bar{X} - n\bar{X})}{\sum_{i=1}^n (X_i - \bar{X})^2} = \beta_1 + \frac{\sum_{i=1}^n (X_i - \bar{X}) \varepsilon_i}{\sum_{i=1}^n (X_i - \bar{X})^2}
\end{aligned}$$

### Definition

An estimator  $\hat{\theta}_n$  for a fixed unknown parameter  $\theta$  is said to be **consistent** if

$$\hat{\theta}_n \rightarrow \theta \quad \text{as} \quad n \rightarrow \infty$$

- Consider our estimator for  $\beta_1$  again and notice it depends on  $n$ ,

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (X_i - \bar{X}_n)(Y_i - \bar{Y}_n)}{\sum_{i=1}^n (X_i - \bar{X}_n)^2} \quad \text{where} \quad \begin{aligned} \bar{X}_n &= \frac{1}{n} \sum_{i=1}^n X_i \\ \bar{Y}_n &= \frac{1}{n} \sum_{i=1}^n Y_i \end{aligned}$$

Q: How can we show it is consistent?

Consider the sample variance for  $n$  observations,

$$\begin{aligned}s_x^2 &= \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x}_n)^2 \\&= \frac{1}{n-1} \sum_{i=1}^n (x_i^2 - 2x_i\bar{x}_n + (\bar{x}_n)^2) \\&= \frac{n}{n-1} \left( \frac{1}{n} \sum_{i=1}^n x_i^2 - 2\bar{x}_n \frac{1}{n} \sum_{i=1}^n x_i + (\bar{x}_n)^2 \right) = \frac{n}{n-1} \left( \bar{x}_n^2 - (\bar{x}_n)^2 \right)\end{aligned}$$

Consider the sample covariance

$$\begin{aligned}c_{xy} &= \frac{1}{n-1} \sum_i^n (x_i - \bar{x}_n)(y_i - \bar{y}_n) \\&= \frac{1}{n-1} \left( \sum_i^n x_i y_i - n\bar{x}_n \bar{y}_n \right) \\&= \frac{1}{n-1} \left( \sum_i^n x_i y_i - \bar{x}_n \sum_i^n y_i \right) \\&= \frac{1}{n-1} \left( \sum_i^n x_i(\beta_0 + \beta_1 x_i + r_i) - \bar{x}_n \sum_i^n (\beta_0 + \beta_1 x_i + r_i) \right) \\&= \frac{n}{n-1} \left( \beta_0 \bar{x}_n + \beta_1 \bar{x}_n^2 + \overline{(xr)}_n - \beta_0 \bar{x}_n - \beta_1 (\bar{x}_n)^2 - \bar{x}_n \bar{r}_n \right) \\&= \beta_1 \frac{n}{n-1} \left( \bar{x}_n^2 - (\bar{x}_n)^2 \right) + \frac{n}{n-1} \overline{(xr)}_n - \frac{n}{n-1} \bar{x}_n \bar{r}_n\end{aligned}$$

According to model assumptions errors have constant  $\mathbb{E}[\varepsilon_i] = 0$  and variance, and are not correlated with each other, so the law of large numbers says

$$\bar{r}_n \rightarrow \mathbb{E}[\varepsilon_i] = 0 \quad \text{when } n \rightarrow \infty$$

And  $\varepsilon_i$  is not correlated with  $X_i$  either, so  $X_i \varepsilon_i$  are uncorrelated with  $X_j \varepsilon_j$

$$\begin{aligned}\text{Cov}[X_i \varepsilon_i, X_j \varepsilon_j] &= \mathbb{E}[X_i \varepsilon_i X_j \varepsilon_j] - \mathbb{E}[X_i \varepsilon_i] \mathbb{E}[X_j \varepsilon_j] \\&= \mathbb{E}[X_i \varepsilon_i X_j] \mathbb{E}[\varepsilon_j] - \mathbb{E}[X_i] \mathbb{E}[\varepsilon_i] \mathbb{E}[X_j] \mathbb{E}[\varepsilon_i] = 0 \quad \text{for } i \neq j\end{aligned}$$

It is not difficult to see the it has zero mean and a constant variance,

$$\begin{aligned}\mathbb{E}[X_i \varepsilon_i] &= \mathbb{E}[X_i] \cdot \mathbb{E}[\varepsilon_i] = 0 \\ \text{Var}[X_i \varepsilon_i] &= \text{Var}[X_i] \text{Var}[\varepsilon_i] + \text{Var}[X_i] \mathbb{E}[\varepsilon_i]^2 + \text{Var}[\varepsilon_i] \mathbb{E}[X_i]^2\end{aligned}$$

So the law of large numbers guarantees the following, as  $n \rightarrow \infty$ , thus  $\hat{\beta}_1 \rightarrow \beta_1$

$$\overline{(xr)}_n \rightarrow 0 \quad \text{when } n \rightarrow \infty$$

It follows that  $\hat{\beta}_0 = \bar{y}_n - \hat{\beta}_1 \bar{x}_n = \beta_0 + \beta_1 \bar{x}_n + \bar{r}_n - \hat{\beta}_1 \bar{x}_n \rightarrow \beta_0 \quad \text{when } n \rightarrow \infty$

- In order to do inferences, i.e.

hypothesis testing, confidence interval, and prediction interval

we have to make slightly stronger assumptions about the error terms  $\varepsilon_i$ :

1. The conditional mean is linear for all  $i$ , i.e.

$$\mathbb{E}[Y_i | X_i = x_i] = \beta_0 + \beta_1 x_i$$

2. The error term

$$\varepsilon_i = Y_i - \beta_0 - \beta_1 X_i$$

has a zero mean and a constant but unknown variance for all  $i$ , i.e.

$$\mathbb{E}[\varepsilon_i | X_i] = 0 \quad \text{and} \quad \text{Var}[\varepsilon_i | X_i] = \sigma^2$$

3. The error terms  $\varepsilon_i$  are independent of  $X_i$ , and of each other, for all  $i$ .
4. The error terms  $\varepsilon_i$  follow a normal distribution.

Q: Why do you think we need the additional assumption?

$$\begin{aligned} f_{\varepsilon_i}(e_i) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{e_i^2}{2\sigma^2}\right) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - (\beta_0 + \beta_1 x_i))^2}{2\sigma^2}\right) \end{aligned}$$

Q: What is the significance of modifying assumption 3.?

- Being independent means the likelihood function given the observed data is

$$\mathcal{L}(\beta_0, \beta_1, \sigma^2) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - (\beta_0 + \beta_1 x_i))^2}{2\sigma^2}\right)$$

from which we can derive the maximum likelihood estimates of  $(\beta_0, \beta_1, \sigma^2)$

$$\boldsymbol{\theta}_{MLE} = \operatorname{Arg} \max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$$

where  $\boldsymbol{\theta}$  denotes a vector of all unknown parameters.

- It is usually easier to work with the negative log-likelihood,

$$\ell(\boldsymbol{\theta}) = -\ln \mathcal{L}(\boldsymbol{\theta})$$

for which we minimise instead of maximising with respect to  $\boldsymbol{\theta}$  to obtain

$$\boldsymbol{\theta}_{MLE}$$

- In this case, it can be shown that the negative log-likelihood is given by

$$\begin{aligned} \ell(\beta_0, \beta_1, \sigma^2) &= \sum_{i=1}^n \left( \frac{1}{2} \ln(2\pi) + \ln \sigma + \frac{(y_i - \beta_0 - \beta_1 x_i)^2}{2\sigma^2} \right) \\ &= \frac{n}{2} \ln(2\pi) + n \ln \sigma + \frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2 \end{aligned}$$

Q: What are the maximum likelihood estimates of  $\beta_0$  and  $\beta_1$ ?

- The maximum likelihood estimates of  $\beta_0$  and  $\beta_1$  are identical to the previous estimates we had, for which we have shown them to be unbiased and consistent.
- By differentiating  $\ell$  with respect to  $\sigma$  and setting it to zero, we have

$$\frac{n}{\sigma} - \frac{1}{\sigma^3} \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2 = 0$$

solving and using the estimates for  $\beta_0$  and  $\beta_1$ , we have the following estimate

$$\hat{\sigma}_{MLE}^2 = \frac{1}{n} \sum_{i=1}^n \hat{\epsilon}_i^2$$

- However, the estimator that gives  $\hat{\sigma}_{MLE}^2$  can be shown to have a small bias,

$$\mathbb{E} [\hat{\sigma}_{MLE}^2] = \frac{n-2}{n} \sigma^2$$

- So the following estimate is unbiased and can also be shown to be consistent

$$\hat{\sigma}_u^2 = \frac{n}{n-2} \hat{\sigma}_{MLE}^2 = \frac{1}{n-2} \sum_{i=1}^n \hat{\epsilon}_i^2$$

- Recall we have shown the estimator for  $\beta_1$  can be written as

$$\hat{\beta}_1 = \beta_1 + \frac{\sum_{i=1}^n (X_i - \bar{X}_n) \varepsilon_i}{\sum_{i=1}^n (X_i - \bar{X}_n)^2}$$

Q: Can you see  $\hat{\beta}_1$  follows a normal distribution with mean of  $\beta_1$  and variance of

$$\frac{\sigma^2}{(n-1)s_x^2} \quad \text{where } s_x^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

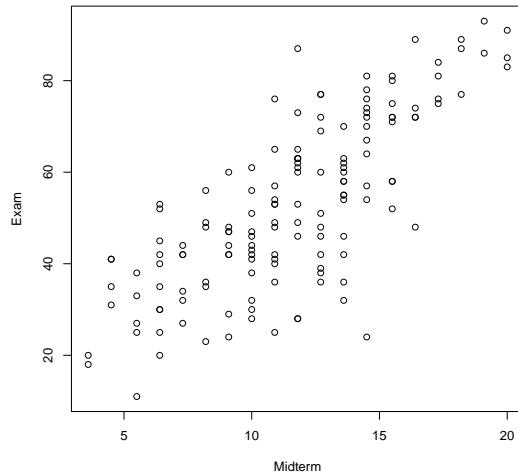
conditional on the observed data  $X_1 = x_1, X_2 = x_2, \dots, X_n = x_n$ .

It is essentially a sum of normal random variables, thus normal, for the variance,

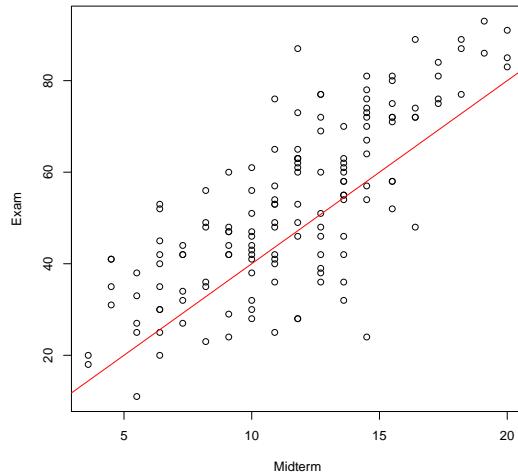
$$\begin{aligned} \text{Var} [\hat{\beta}_1 \mid X_1 = x_1, \dots, X_n = x_n] &= \text{Var} \left[ \beta_1 + \frac{\sum_{i=1}^n (X_i - \bar{X}_n) \varepsilon_i}{\sum_{i=1}^n (X_i - \bar{X}_n)^2} \mid X_1 = x_1, \dots, X_n = x_n \right] \\ &= \text{Var} \left[ \sum_{i=1}^n \frac{(X_i - \bar{X}_n)}{W} \varepsilon_i \mid X_1 = x_1, \dots, X_n = x_n \right] \\ &= \sum_{i=1}^n \frac{(X_i - \bar{X}_n)^2}{W^2} \text{Var} [\varepsilon_i \mid X_1 = x_1, \dots, X_n = x_n] \\ &= \frac{\sigma^2}{W} \end{aligned}$$

- This means, if  $\sigma^2$  is somehow known, we can use the normal distribution to do hypothesis testing and construct confidence interval for  $\beta_1$ , if  $\sigma^2$  is not known, then we can use a  $t$ -distribution to do so.
- We have only discussed  $\beta_1$ , however, similarly ideas can be applied to  $\beta_0$ .

Q: What do you think the relationship between Midterm and Final Exam is?



- The straight line  $y = 4x$  seems to be reasonable to my eye.



```
> # Load data locally
```

```

> course.df = read.table("~/Desktop/course.txt",
+                         header = TRUE)
>
> plot(Exam~Midterm, data = course.df)
>
> abline(a = 0, b = 4, col = "red")

```

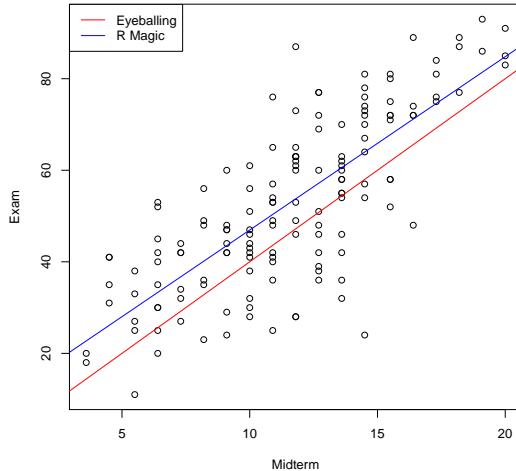
- Of course, R can “automatically” construct the simple linear model for us

```

> course.lm = lm(Exam~Midterm, data = course.df)
>
> abline(course.lm, col = "blue")
>
> legend("topleft", legend =
+         c("Eyeballing", "R Magic"),
+         lty = 1, col = c(2, 4))

```

- Eyeballing is not too far from the truth!



- Of course, we can confirm what R did by manually computing the estimates.

```

> x = course.df$Midterm
>
> y = course.df$Exam

> beta_1_hat = cov(x, y) / var(x)

> xbar = mean(x); ybar = mean(y)

> beta_0_hat = ybar - beta_1_hat * xbar

```

```

> beta_0_hat; beta_1_hat
[1] 9.084463
[1] 3.785924

> beta_0_hat; beta_1_hat
[1] 9.084463
[1] 3.785924

> summary(course.lm)

```

---

```

Call:
lm(formula = Exam ~ Midterm, data = course.df)

Residuals:
    Min      1Q  Median      3Q     Max 
-39.980 -6.471  0.826  8.575 33.242 

Coefficients:
            Estimate Std. Error t value Pr(>t)    
(Intercept) 9.0845    3.2204   2.821  0.00547 ** 
Midterm      3.7859    0.2647  14.301 < 2e-16 *** 
---
Signif. codes:  0 `***` 0.001 `**` 0.01 `*` 0.05 `.` 0.1 ` ` 1

Residual standard error: 12.05 on 144 degrees of freedom
Multiple R-squared:  0.5868, Adjusted R-squared:  0.5839 
F-statistic: 204.5 on 1 and 144 DF,  p-value: < 2.2e-16

```

---

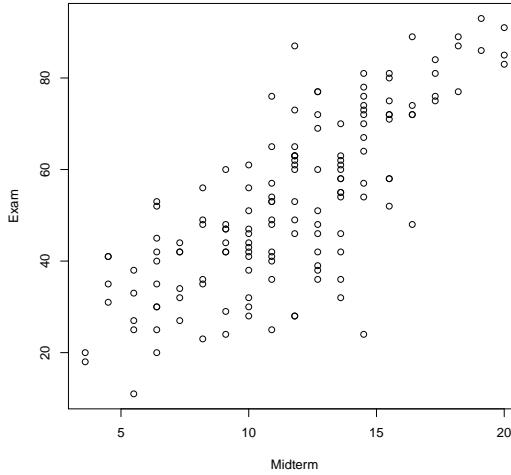
Q: Why can we not trust the above results at this stage?

## 2.4 Diagnostics

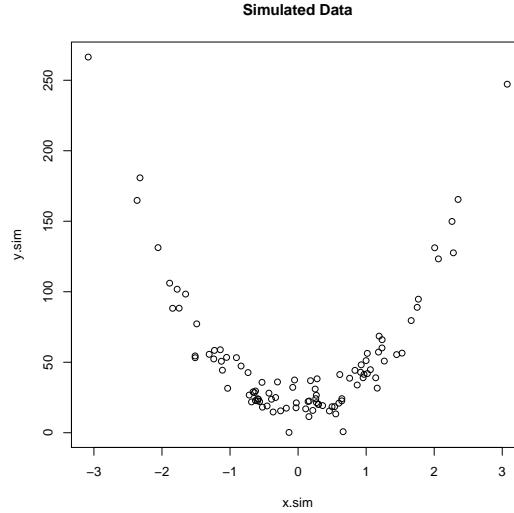
Q: How can we check assumption 1. graphically? What do you expect to see?

1. The conditional mean of the response is linear for all  $i$ , i.e.

$$\mathbb{E}[Y_i | X_i = x_i] = \beta_0 + \beta_1 x_i$$



- The following plot of response and regressor provides evidence of nonlinearity



Q: However, do we know for sure that assumption 1. is not adequate?

- Alternatively, nonlinearity is also evident in a plot of

$$y_i \quad \text{Vs} \quad \hat{y}_i$$

Q: What do you expect to see in such plots if assumption 1. is fine?

But we again expect points to be scattered around the diagonal line instead of forming a pattern that is systematically off the diagonal line if the linearity assumption is fine.

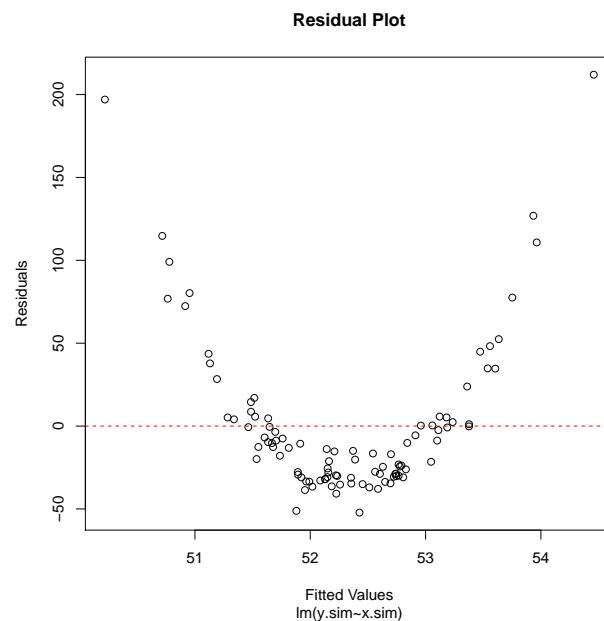
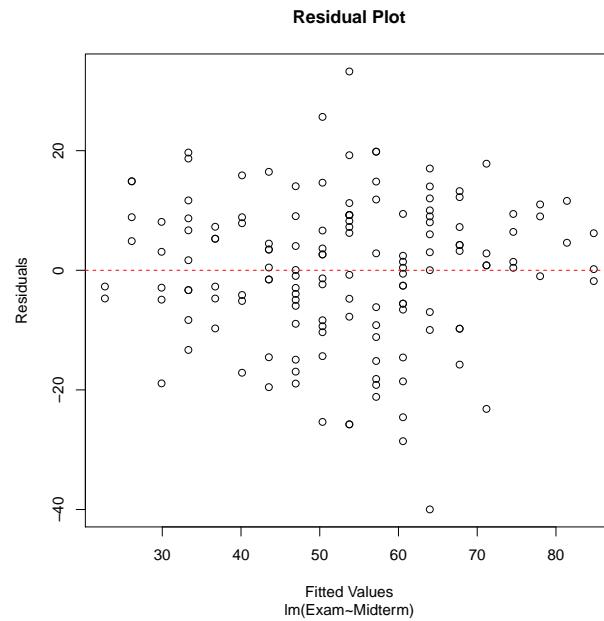
- In practice, to avoid the visual distraction of a sloping pattern, we plot

$$\hat{e}_i \quad \text{Vs} \quad \hat{y}_i$$

Q: What do you expect to see in such plots if assumption 1. is fine?

- Having non-flat band of points suggests that 1. might be inadequate.

```
> plot(fitted.values(course.lm),
+       residuals(course.lm),
+       xlab = "Fitted Values", ylab = "Residuals",
+       main = "Residual Plot",
+       sub = "lm(Exam~Midterm)")
>
> abline(a = 0, b = 0, lty = 2, col = "red")
```



Q: How can we check assumption 2. graphically? What do you expect to see?

- 2. The error has a zero mean and a constant but unknown variance for all  $i$ , i.e.

$$\mathbb{E} [\varepsilon_i | X_i] = 0 \quad \text{and} \quad \text{Var} [\varepsilon_i | X_i] = \sigma^2$$

Q: If assumption 2. is satisfied, what do you expect to see in a plot of

$$e_i \quad \text{Vs} \quad \hat{y}_i$$

- However,  $e_i$  are not directly observed, we have to consider the residuals

$$\hat{e}_i = y_i - \hat{y}_i = \beta_0 + \beta_1 x_i + e_i - b_0 - b_1 x_i = (\beta_0 - b_0) + (\beta_1 - b_1) x_i + e_i$$

- Notice  $e_i$  and  $\hat{e}_i$  are not the same, however, it can be shown that

$$\text{Var} [Y_i - \hat{Y}_i \mid X_i] = \sigma^2 \left[ 1 - \frac{1}{n} - \frac{(X_i - \bar{X}_n)^2}{\sum_{i=1}^n (X_i - \bar{X}_n)^2} \right]$$

Q: So what do you expect to see in a plot of  $\hat{e}_i$  and  $x_i$  if assumption 2. is fine?

Given the followings

$$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X} \quad \text{and} \quad \hat{\beta}_1 = \beta_1 + \sum_{i=1}^n \frac{(X_i - \bar{X}) \varepsilon_i}{\sum_{i=1}^n (X_i - \bar{X})^2}$$

Since  $Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$ , we have

$$\begin{aligned}\hat{\beta}_0 &= \bar{Y} - \hat{\beta}_1 \bar{X} = \frac{1}{n} \sum_{i=1}^n (\beta_0 + \beta_1 X_i + \varepsilon_i) - \bar{X} \left( \beta_1 + \sum_{i=1}^n \frac{(X_i - \bar{X}) \varepsilon_i}{\sum_{i=1}^n (X_i - \bar{X})^2} \right) \\ &= \beta_0 + \bar{\varepsilon} - \sum_{i=1}^n \frac{\bar{X} (X_i - \bar{X}) \varepsilon_i}{\sum_{i=1}^n (X_i - \bar{X})^2}\end{aligned}$$

from which, it is clear that

$$\mathbb{E} [\hat{\beta}_0 | X] = \beta_0 + \frac{1}{n} \sum_{i=1}^n \mathbb{E} [\varepsilon_i | X] - \sum_{i=1}^n \frac{\bar{X} (X_i - \bar{X})}{\sum_{i=1}^n (X_i - \bar{X})^2} [\varepsilon_i | X] = \beta_0$$

According to the following property of variance

$$\text{Var} \left[ \sum_i^n a_i X_i \right] = \sum_i^n a_i^2 \text{Var} [X_i] + \sum_{1 \leq i < j \leq n} 2a_i a_j \text{Cov} [X_i, X_j]$$

we have,

$$\begin{aligned}\text{Var} [\hat{\beta}_0 | X] &= \frac{1}{n^2} \sum_{i=1}^n \text{Var} [\varepsilon_i | X] + \sum_{i=1}^n \frac{\bar{X}^2 (X_i - \bar{X})^2}{\left( \sum_{i=1}^n (X_i - \bar{X})^2 \right)^2} \text{Var} [\varepsilon_i | X] \\ &\quad - \frac{2}{n} \sum_{i=1}^n \frac{\bar{X} (X_i - \bar{X})}{\sum_{i=1}^n (X_i - \bar{X})^2} \text{Cov} [\varepsilon_i, \varepsilon_j] \\ &= \frac{\sigma^2}{n} + \frac{\bar{X}^2 \sigma^2}{\sum_{i=1}^n (X_i - \bar{X})^2} - \frac{2\bar{X} \sigma^2}{n \sum_{i=1}^n (X_i - \bar{X})^2} \sum_{i=1}^n (X_i - \bar{X}) \\ &= \sigma^2 \left( \frac{1}{n} + \frac{\bar{X}^2}{\sum_{i=1}^n (X_i - \bar{X})^2} \right)\end{aligned}$$

and

$$\begin{aligned}\text{Var} [Y_i - \hat{Y}_i | X] &= \text{Var} \left[ (\beta_0 - \hat{\beta}_0) + (\beta_1 - \hat{\beta}_1) X_i + \varepsilon_i | X \right] \\ &= \text{Var} [\hat{\beta}_0 | X] + X_i^2 \text{Var} [\hat{\beta}_1 | X] + \text{Var} [\varepsilon_i | X] \\ &\quad + 2X_i \text{Cov} [\hat{\beta}_0, \hat{\beta}_1 | X] \\ &\quad + 2 \text{Cov} \left[ (\beta_0 - \hat{\beta}_0) + (\beta_1 - \hat{\beta}_1) X_i, \varepsilon_i | X \right]\end{aligned}$$

For the covariances, recall the following properties of covariance

$$\begin{aligned}\text{Cov}[X + a, Y + b] &= \text{Cov}[X, Y] \\ \text{Cov}[aX + bY, cW + dV] &= ac \text{Cov}[X, W] + ad \text{Cov}[X, V] \\ &\quad + bc \text{Cov}[Y, W] + bd \text{Cov}[Y, V]\end{aligned}$$

Substituting  $\hat{\beta}_0$  and  $\hat{\beta}_1$  into the covariance  $\text{Cov}[\hat{\beta}_0, \hat{\beta}_1 | X]$ , we have

$$\begin{aligned}&\text{Cov}\left[\left(\beta_0 + \bar{\varepsilon} - \sum_{i=1}^n \frac{\bar{X}(X_i - \bar{X})\varepsilon_i}{\sum_{i=1}^n (X_i - \bar{X})^2}\right), \left(\beta_1 + \sum_{i=1}^n \frac{(X_i - \bar{X})\varepsilon_i}{\sum_{i=1}^n (X_i - \bar{X})^2}\right) | X\right] \\&= \text{Cov}\left[\left(\frac{1}{n} \sum_{i=1}^n \varepsilon_i - \sum_{i=1}^n \frac{\bar{X}(X_i - \bar{X})\varepsilon_i}{\sum_{i=1}^n (X_i - \bar{X})^2}\right), \left(\sum_{i=1}^n \frac{(X_i - \bar{X})\varepsilon_i}{\sum_{i=1}^n (X_i - \bar{X})^2}\right) | X\right] \\&= \frac{1}{n} \text{Cov}\left[\left(\sum_{i=1}^n \varepsilon_i\right), \left(\sum_{i=1}^n \frac{(X_i - \bar{X})\varepsilon_i}{\sum_{i=1}^n (X_i - \bar{X})^2}\right) | X\right] \\&\quad - \text{Cov}\left[\left(\sum_{i=1}^n \frac{\bar{X}(X_i - \bar{X})\varepsilon_i}{\sum_{i=1}^n (X_i - \bar{X})^2}\right), \left(\sum_{i=1}^n \frac{(X_i - \bar{X})\varepsilon_i}{\sum_{i=1}^n (X_i - \bar{X})^2}\right) | X\right] \\&= \frac{\sigma^2}{n} \sum_{i=1}^n \frac{(X_i - \bar{X})}{\sum_{i=1}^n (X_i - \bar{X})^2} - \bar{X}\sigma^2 \sum_{i=1}^n \frac{(X_i - \bar{X})^2}{\left(\sum_{i=1}^n (X_i - \bar{X})^2\right)^2} \\&= \frac{\sigma^2}{n} \cdot 0 - \frac{\bar{X}\sigma^2}{\sum_{i=1}^n (X_i - \bar{X})^2} = \frac{-\bar{X}\sigma^2}{\sum_{i=1}^n (X_i - \bar{X})^2}\end{aligned}$$

Using the property  $\text{Cov}[X, Y] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$  and  $\mathbb{E}[\varepsilon_i | X] = 0$ , thus

$$\begin{aligned}&\text{Cov}\left[\left(\left(\beta_0 - \hat{\beta}_0\right) + \left(\beta_1 - \hat{\beta}_1\right)X_i\right), \varepsilon_i | X\right] \\&= \mathbb{E}\left[\varepsilon_i \left(\left(\beta_0 - \hat{\beta}_0\right) + \left(\beta_1 - \hat{\beta}_1\right)X_i\right) | X\right] \\&= -\mathbb{E}\left[\varepsilon_i \hat{\beta}_0 | X\right] - X_i \mathbb{E}\left[\varepsilon_i \hat{\beta}_1 | X\right] \\&= -\mathbb{E}\left[\varepsilon_i (\bar{Y} - \hat{\beta}_1 \bar{X}) | X\right] - X_i \mathbb{E}\left[\varepsilon_i \hat{\beta}_1 | X\right] \\&= -\mathbb{E}\left[\varepsilon_i \bar{Y} | X\right] - (X_i - \bar{X}) \mathbb{E}\left[\varepsilon_i \hat{\beta}_1 | X\right]\end{aligned}$$

Continuing, we have

$$\begin{aligned}\mathbb{E} [\varepsilon_i \bar{Y} | X] &= \mathbb{E} [\varepsilon_i (\beta_0 + \beta_1 \bar{X} + \bar{\epsilon}) | X] \\ &= 0 + 0 + \frac{1}{n} \mathbb{E} \left[ \varepsilon_i \sum_{i=1}^n \varepsilon_i \right] \\ &= \frac{\sigma^2}{n}\end{aligned}$$

since

$$\begin{aligned}\text{Var} [\varepsilon_i | X] &= \mathbb{E} [\varepsilon_i^2 | X] - (\mathbb{E} [\varepsilon_i | X])^2 \\ &= \mathbb{E} [\varepsilon_i^2 | X] \\ &= \sigma^2\end{aligned}$$

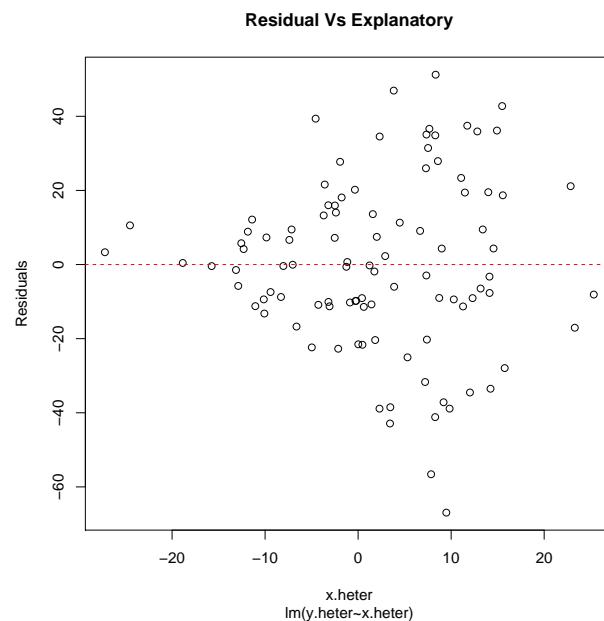
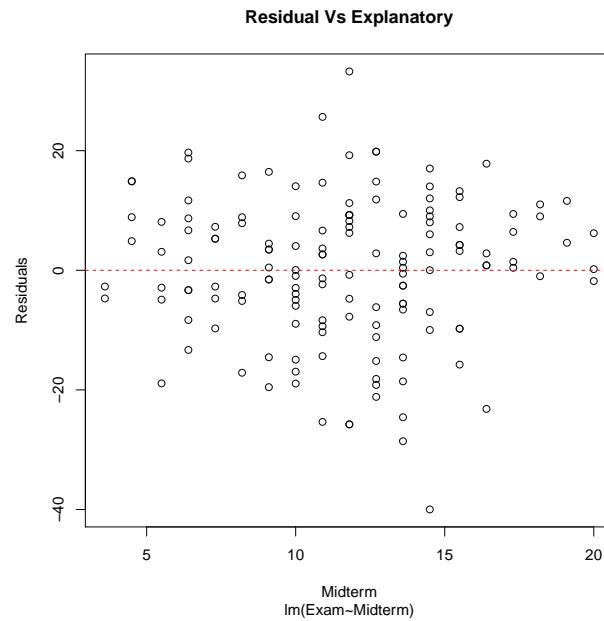
$$\begin{aligned}\text{Cov} [\varepsilon_i, \varepsilon_j | X] &= \mathbb{E} [\varepsilon_i \varepsilon_j | X] - \mathbb{E} [\varepsilon_i | X] \mathbb{E} [\varepsilon_j | X] \\ &= \mathbb{E} [\varepsilon_i \varepsilon_j | X] \\ &= 0 \quad \text{for } i \neq j\end{aligned}$$

and

$$\mathbb{E} [\varepsilon_i \hat{\beta}_1 | X] = \mathbb{E} \left[ \varepsilon_i \beta_1 + \varepsilon_i \sum_{i=1}^n \frac{(X_i - \bar{X}) \varepsilon_i}{\sum_{i=1}^n (X_i - \bar{X})^2} | X \right] = \frac{\sigma^2 (X_i - \bar{X})}{\sum_{i=1}^n (X_i - \bar{X})^2}$$

Now putting everything together, we have the stated result,

$$\begin{aligned}\text{Var} [Y_i - \hat{Y}_i | X] &= \text{Var} [\hat{\beta}_0 | X] + X_i^2 \text{Var} [\hat{\beta}_1 | X] + \text{Var} [\varepsilon_i | X] \\ &\quad + 2X_i \text{Cov} [\hat{\beta}_0, \hat{\beta}_1 | X] - 2\mathbb{E} [\varepsilon_i \bar{Y} | X] \\ &\quad - 2(X_i - \bar{X}) \mathbb{E} [\varepsilon_i \hat{\beta}_1 | X] \\ &= \sigma^2 \left( \frac{1}{n} + \frac{\bar{X}^2}{\sum_{i=1}^n (X_i - \bar{X})^2} \right) + \frac{\sigma^2 X_i^2}{\sum_{i=1}^n (X_i - \bar{X})^2} + \sigma^2 \\ &\quad - \sigma^2 \frac{2X_i \bar{X}}{\sum_{i=1}^n (X_i - \bar{X})^2} - \sigma^2 \frac{2}{n} - \sigma^2 \frac{2(X_i - \bar{X})^2}{\sum_{i=1}^n (X_i - \bar{X})^2} \\ &= \sigma^2 \left( 1 - \frac{1}{n} - \frac{(X_i - \bar{X})^2}{\sum_{i=1}^n (X_i - \bar{X})^2} \right)\end{aligned}$$



Q: How can we check the third assumption?

- 3. The errors are independent of  $X_i$ , and of each other.

- You might be tempted to use the fact the following must be zero if 3. is true

$$\text{Cov} [X_i, \varepsilon_i] = 0$$

and thus the sample covariance shall not be too far away from zero, that is,

$$\sum_{i=1}^n (e_i - \bar{e}_n)(x_i - \bar{x}_n) = \sum_{i=1}^n e_i(x_i - \bar{x}_n) \approx 0$$

which is correct, however, you might wrongly jump to the conclusion to test

$$\sum_{i=1}^n (\hat{e}_i - \bar{\hat{e}}_n)(x_i - \bar{x}_n) = \sum_{i=1}^n \hat{e}_i(x_i - \bar{x}_n)$$

Q: Why is this not going to provide any information about  $\text{Cov} [X_i, \varepsilon_i]$ ?

- Recall  $b_0$  and  $b_1$  are solutions to the following estimation equations

$$\begin{aligned} \sum_{i=1}^n (y_i - b_0 - b_1 x_i) &= 0 \implies \sum_{i=1}^n \hat{e}_i = 0 \\ \sum_{i=1}^n (y_i - b_0 - b_1 x_i)(x_i) &= 0 \implies \sum_{i=1}^n \hat{e}_i x_i = 0 \end{aligned}$$

Q: Why does the sample covariance between residual and  $X$  is always 0?

$$\sum_{i=1}^n \hat{e}_i x_i - \bar{x} \sum_{i=1}^n \hat{e}_i - \bar{\hat{e}} \sum_{i=1}^n (x_i - \bar{x}) = 0 \implies \sum_{i=1}^n (\hat{e}_i - \bar{\hat{e}})(x_i - \bar{x}_n) = 0$$

so checking the sample covariance is not a detection tool for the dependence.

- In practice, the assumption on the dependence of  $\varepsilon$  on  $X$  is intimately linked to assumption 1., so we don't run additional diagnostics for it.

$$y_i = \mathbb{E}[Y_i | X_i = x_i] + \varepsilon_i = \beta_0 + \beta_1 x_i + \varepsilon_i$$

Q: How about the part says that errors are independent?

Q: Can we rely on residuals to detect correlated errors?

Q: Will residuals be independent if errors are independent?

$$\sum_{i=1}^n \hat{e}_i = 0$$

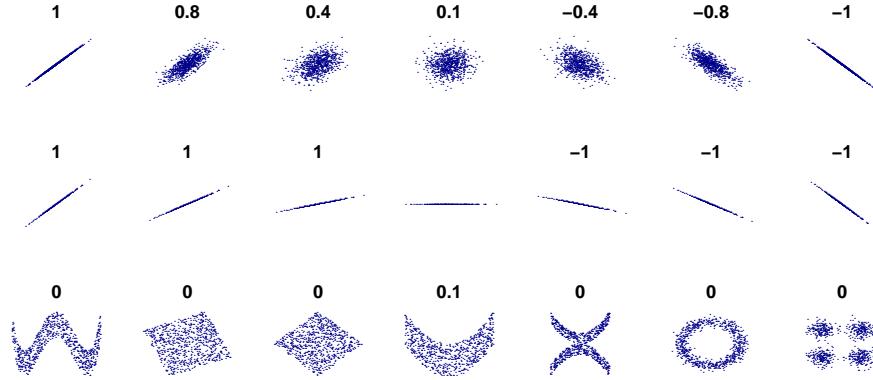
- We expect the sample covariance between  $\hat{e}_i$  and  $\hat{e}_j$  to be small but nonzero.
- Given all the residuals except one, we can always work out the last one, thus they must be correlated. However, the correlation will disappear as  $n \rightarrow \infty$ .
- So, when there is enough data, we do not expect to see a large correlation

between residual  $\hat{e}_i$  and previous residual  $\hat{e}_{i-1}$

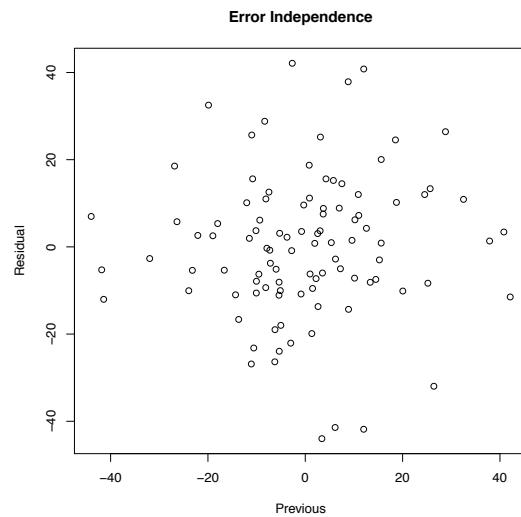
Q: How can we detect excessive correlation in the residuals, thus dependence?

Q: How about nonlinear dependence between  $\hat{e}_i$  and  $\hat{e}_{i-1}$ ?

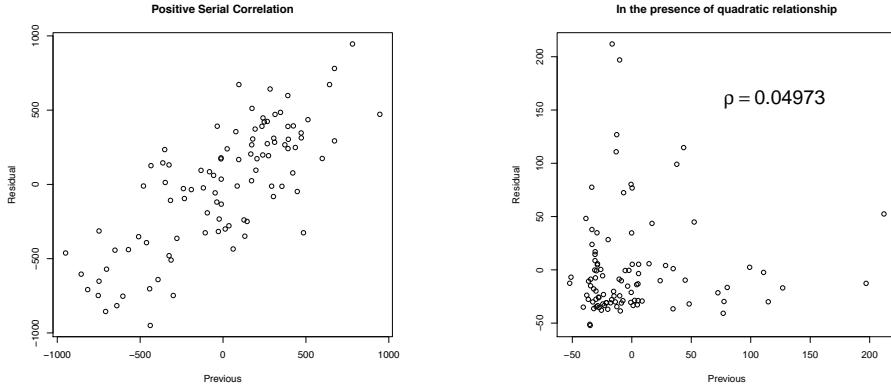
- Plotting residuals is a way to detect severe dependence, especially nonlinear.



- Recall Pearson correlation coefficient is only a measure of linear correlation.
- We are hoping for random scatter, i.e. no particular linear/nonlinear pattern.



- A clear pattern in the plot indicates 1. or 3. is violated.



- Of course, we need to check not only just neighbouring residuals in general,

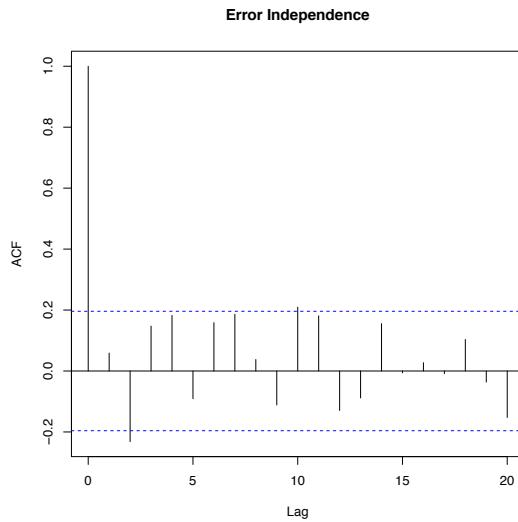
$$\hat{R}(k) = \frac{(n-1) \sum_{i=1}^{n-k} \left( \hat{e}_i - \frac{1}{n} \sum_{i=1}^n \hat{e}_i \right) \left( \hat{e}_{i+k} - \frac{1}{n} \sum_{i=1}^n \hat{e}_i \right)}{(n-k) \sum_{i=1}^n \left( \hat{e}_i - \frac{1}{n} \sum_{i=1}^n \hat{e}_i \right)^2}$$

which is an estimate of **autocorrelation function**.

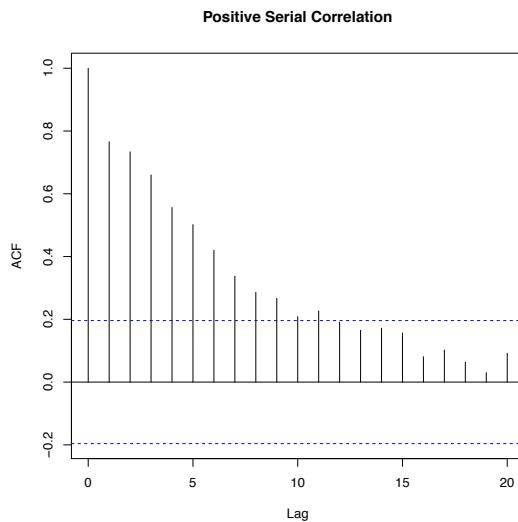
- Under our choice of  $\hat{\beta}_0$  and  $\hat{\beta}_1$ , this estimate can be simplified to

$$\hat{R}(k) = \frac{(n-1) \sum_{i=1}^{n-k} \hat{e}_i \hat{e}_{i+k}}{(n-k) \sum_{i=1}^n \hat{e}_i^2}$$

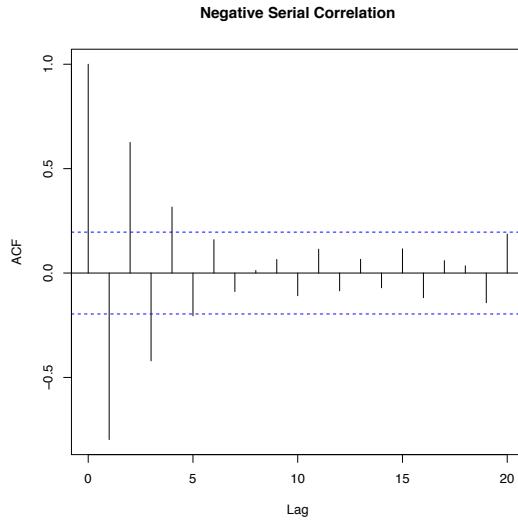
- R can easily compute and plot this estimate for various  $k$ ,



- We expect the estimates to be small and patternless if errors are independent



- If not, then we have evidence that errors are not independent.



```

> # Generating autocorrelated data
> y.psc.vec = double(n)
> y.nsc.vec = double(n)
>
> y.psc.vec[1] = rnorm(1, mean = msl[1])
> y.nsc.vec[1] = rnorm(1, mean = msl[1])
>
> for (i in 2:n) {
+   y.psc.vec[i] =
+     rnorm(1, mean = msl[i] + 0.8 * y.psc.vec[i-1])
>
+   y.nsc.vec[i] =
+     rnorm(1, mean = msl[i] - 0.8 * y.nsc.vec[i-1])
+ }
>
> psc.lm = lm(y.psc.vec~x.new.vec)
> nsc.lm = lm(y.nsc.vec~x.new.vec)

# Plotting ehat_i against ehat_{i-1}, and acf
> plot(psc.lm$residuals[-n], psc.lm$residuals[-1],
+       xlab = "Previous", ylab = "Residual",
+       main = "Positive Serial Correlation")
> plot(lmlist[[1]]$residuals[-n],
+       lmlist[[1]]$residuals[-1],
+       xlab = "Previous", ylab = "Residual",
+       main = "Error Independence")
> plot(nsc.lm$residuals[-n], nsc.lm$residuals[-1],
+       xlab = "Previous", ylab = "Residual",
+       main = "Negative Serial Correlation")

```

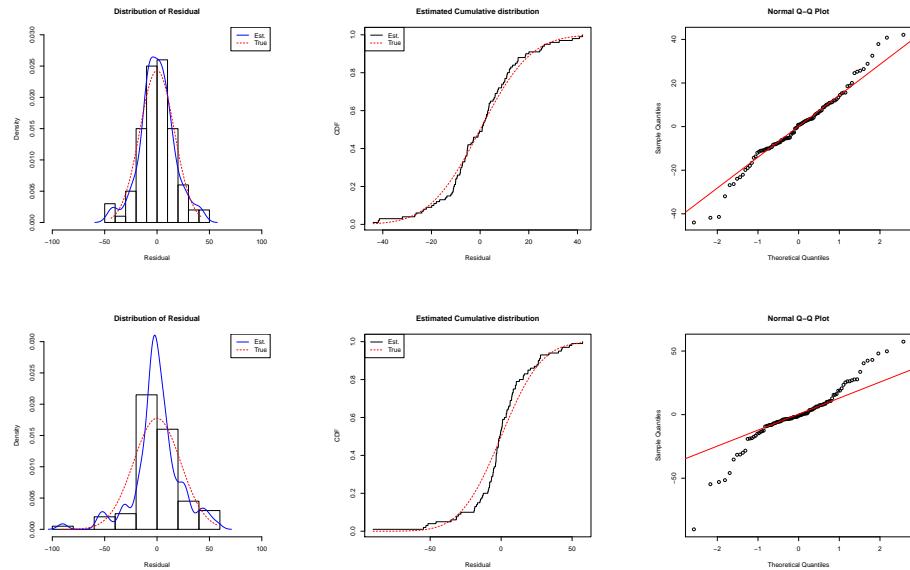
```

>
> acf(psc.lm$residuals,
+       main = "Positive Serial Correlation")
> acf(lmlist[[1]]$residuals,
+       main = "Error Independence")
> acf(nsc.lm$residuals,
+       main = "Negative Serial Correlation")

```

- Normality can be checked by plotting the estimated Vs assumed distribution.

4. The errors follow the normal distribution of  $N(0, \sigma^2)$ .



```

> tmp = seq(-100, 100, length.out = 100)
> res = residuals(lmlist[[1]])
> # res = residuals(msl.lm)
>
> sigma = sqrt(sum(res^2) / (n-2))
> # Density function
> hist(res, probability = TRUE,
+       xlim = c(-100, 100), ylim = c(0, 0.03),
+       xlab = "Residual",
+       main = "Distribution of Residual")
>
> lines(density(res), col = "blue")
>
> lines(tmp, dnorm(tmp, sd = sigma),
+       col = "red", lty = 2)
>

```

```

> legend("topright", legend = c("Est.", "True"),
+         lty = c(1,2), col = c(4, 2))

> # Distribution function
> sample_quantile = sort(res); sample_cdf = (1:n)/n
>
> tmp = seq(min(sample_quantile),
+             max(sample_quantile), length.out = 100)
>
> plot(sample_quantile, sample_cdf,
+       xlab = "Residual", ylab = "CDF",
+       main = "Estimated Cumulative distribution",
+       type = "s")
>
> lines(tmp, pnorm(tmp, sd = sigma),
+        col = 2, lty = 2)
>
> legend("topleft", legend = c("Est.", "True"),
+         lty = c(1, 2), col = c(1, 2))

> # Quantile-Quantile Normal plot
> qqnorm(res)
> qqline(res, col = "red")

```

- In general, we can compare the quantiles of two arbitrary distributions

```

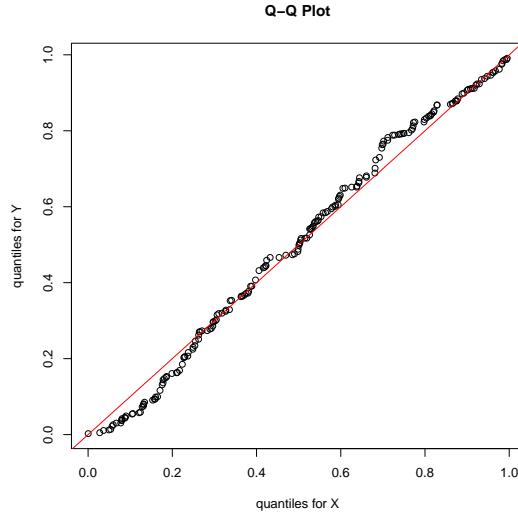
> num = 200 # number of observation

> # Compare samples from the same distribution
> x = runif(num, min = 0, max = 1)
> y = runif(num, min = 0, max = 1)

> qqplot(x, y, main = "Q-Q Plot",
+          xlab = "quantiles for x",
+          ylab = "quantiles for y")
>
> abline(a = 0, b = 1, col = 2)

```

Q: What do you expect to see if  $X$  and  $Y$  follow the same distribution?



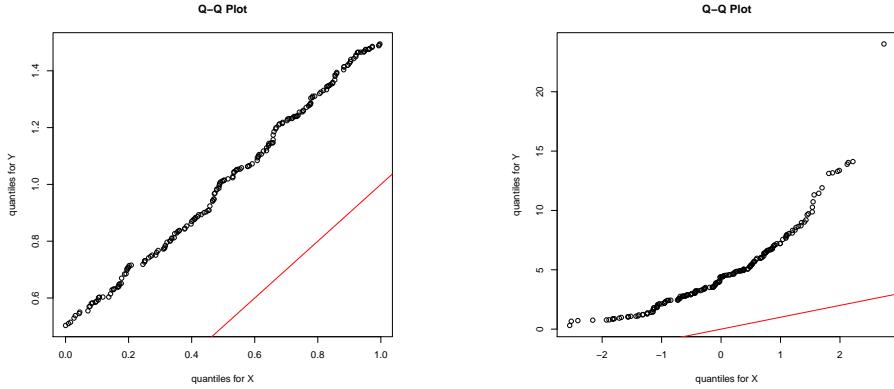
Q: What do you expect to see in the Q-Q plot if we have the following instead?

```

> # The same shape, but different center
> x = runif(num, min = 0, max = 1)
> y = runif(num, min = 0.5, max = 1.5)
> qqplot(x, y, main = "Q-Q Plot",
+         xlab = "quantiles for X",
+         ylab = "quantiles for Y")
> abline(a = 0, b = 1, col = 2)

> # One is Skewed to the right, one symmetric
> x = rnorm(num)
> y = rchisq(num, df = 5)
> qqplot(x, y, main = "Q-Q Plot",
+         xlab = "quantiles for X",
+         ylab = "quantiles for Y")
> abline(a = 0, b = 1, col = 2)

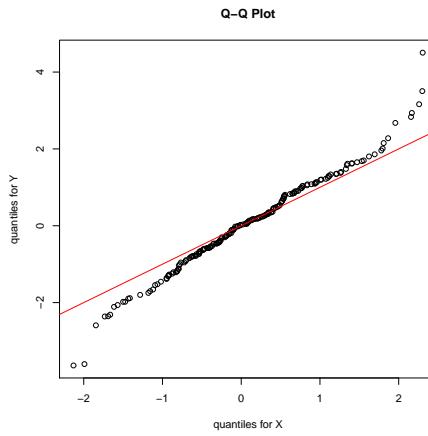
```

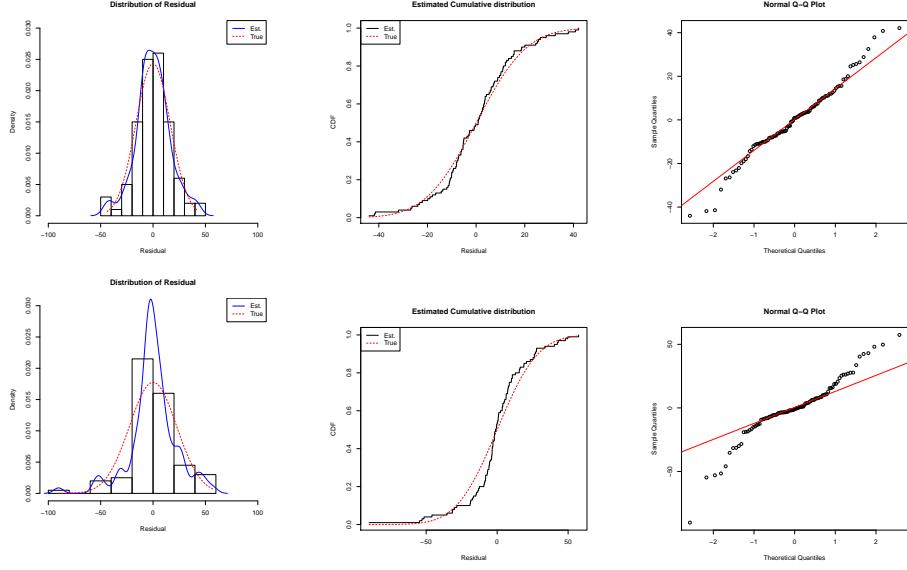


Q: How about the following?

```
> # One has longer tail than the other
> x = rnorm(num)
> y = rt(num, df = 5)

> qqplot(x, y, main = "Q-Q Plot",
+         xlab = "quantiles for X",
+         ylab = "quantiles for Y")
> abline(a = 0, b = 1, col = 2)
```





## 2.5 Transformation

Q: How should we proceed if one of assumptions is violated?

1. The conditional mean is linear for all  $i$ , i.e.

$$\mathbb{E}[Y_i | X_i = x_i] = \beta_0 + \beta_1 x_i$$

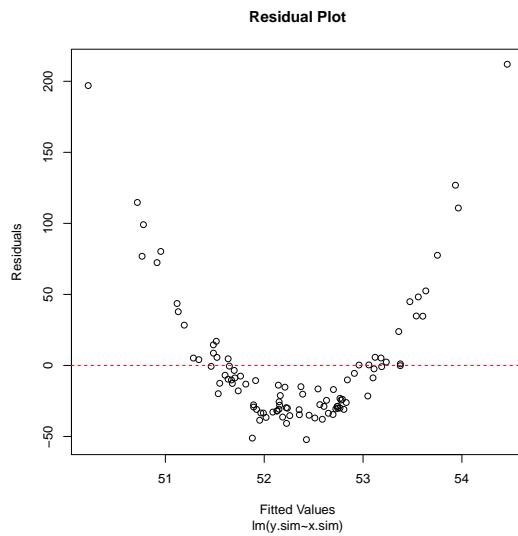
2. The error term

$$\varepsilon_i = Y_i - \beta_0 - \beta_1 X_i$$

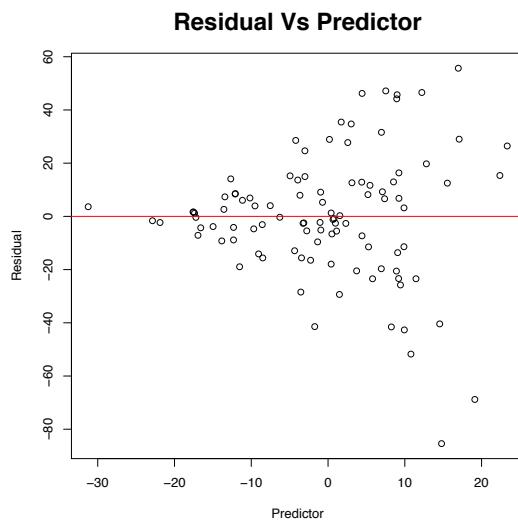
has a zero mean and a constant but unknown variance for all  $i$ , i.e.

$$\mathbb{E}[\varepsilon_i | X_i] = 0 \quad \text{and} \quad \text{Var}[\varepsilon_i | X_i] = \sigma^2$$

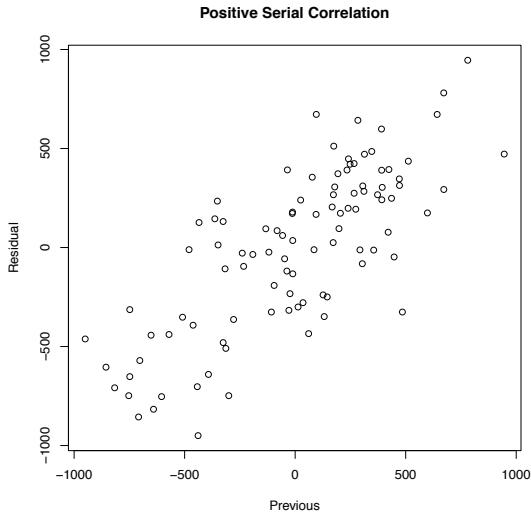
3. The error terms  $\varepsilon_i$  are independent of  $X_i$ , and of each other, for all  $i$ .
  4. The error terms  $\varepsilon_i$  follow a normal distribution.
- Nonlinearity, e.g.



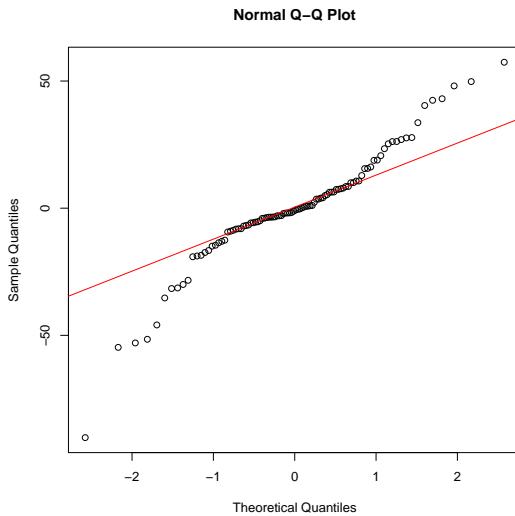
- Heteroskedasticity, e.g.



- Autocorrelation, e.g.



- Nonnormality , e.g.



- There are three approaches:

  1. Transformation on the variables
  2. Switch to advanced Models
  3. Do nothing!

- Transformation is often used when assumption 1., 2. or 4. are in doubt.
- Consider the following dataset,

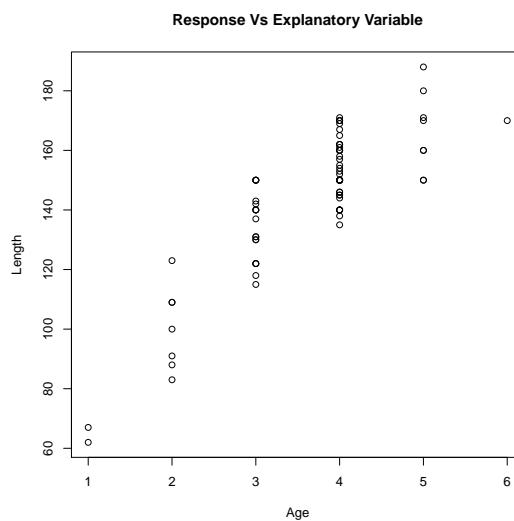
Age      Integer age of the fish  
 Length    Integer length of the fish in mm

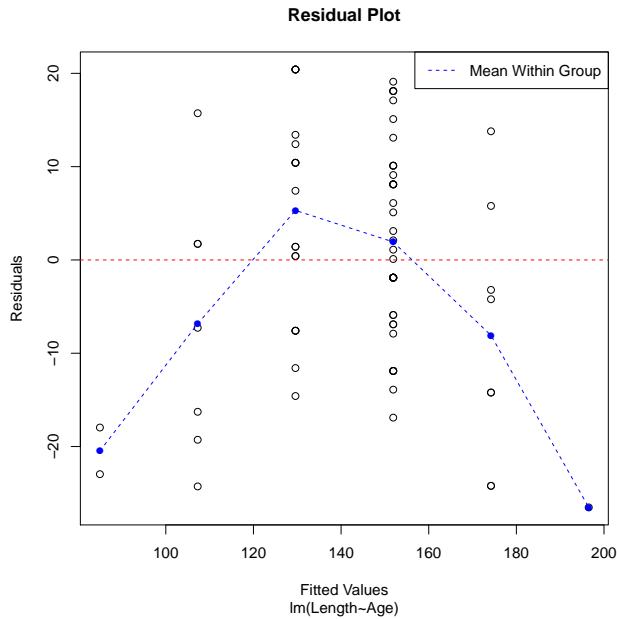
which is about 78 bluegills that were captured from Lake Mary, Minnesota.

```

> # Load data locally
> bluegill.df = read.table("~/Desktop/bluegill.txt",
+                               header = TRUE)

> plot(Length~Age, data = bluegill.df)
> title(main = "Response Vs Explanatory Variable")
  
```





```

> model = bluegill.LM # For reusing the code
> # Basic residual Plot
> plot(fitted.values(model), residuals(model),
+       xlab = "Fitted Values", ylab = "Residuals",
+       main = "Residual Plot",
+       sub = "lm(Length~Age)")
> abline(a = 0, b = 0, lty = 2, col = "red")
> # Computing mean residual within group
> FV.group = factor(bluegill.df$Age)
> res.mean = tapply(residuals(model), FV.group, mean)
> age.unique = as.numeric(levels(FV.group))
> tmp = data.frame(Age = age.unique)
> x.tmp = predict(model, tmp)
> y.tmp = as.numeric(res.mean)
> # Adding the means, and a legend
> points(x.tmp, y.tmp, col = "blue", pch = 16)
> lines(x.tmp, y.tmp, col = "blue", lty = 2)
> legend("topright", lty = 2, col = 4,
+        legend = c("Mean Within Group"))

```

- The curvature suggests to us that the conditional mean might be

$$\mathbb{E}[Y_i | X_i = x_i] = \beta_0 + \beta_1 x_i + \beta_2 x_i^2$$

```

> bluegill.quad.LM = lm(Length~Age+I(Age^2),
+                        data = bluegill.df)

```

```

> model = bluegill.quad.LM # for reusing the code
> summary(model)

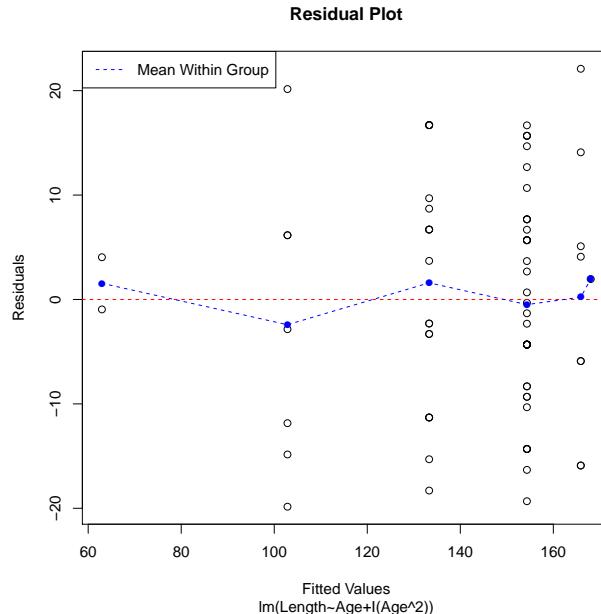
Call:
lm(formula = Length ~ Age + I(Age^2), data = bluegill.df)

Residuals:
    Min      1Q  Median      3Q     Max 
-19.846 -8.321 -1.137  6.698 22.098 

Coefficients:
            Estimate Std. Error t value Pr(>t)    
(Intercept) 13.622     11.016   1.237   0.22    
Age         54.049      6.489   8.330 2.81e-12 ***
I(Age^2)   -4.719      0.944  -4.999 3.67e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Residual standard error: 10.91 on 75 degrees of freedom
Multiple R-squared:  0.8011, Adjusted R-squared:  0.7958 
F-statistic: 151.1 on 2 and 75 DF,  p-value: < 2.2e-16

```



- Consider the following dataset,

year	Year of manufacture
price	Price in Australian dollars

which is about 123 Mazda cars taken from a Melbourne newspaper in 1991.

```

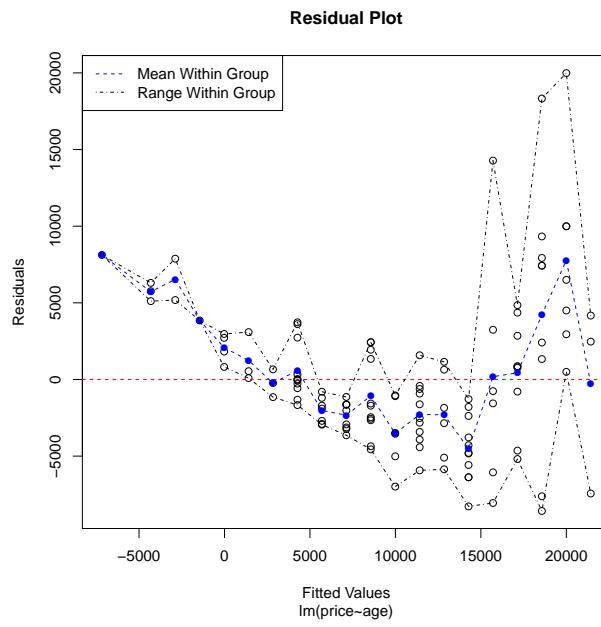
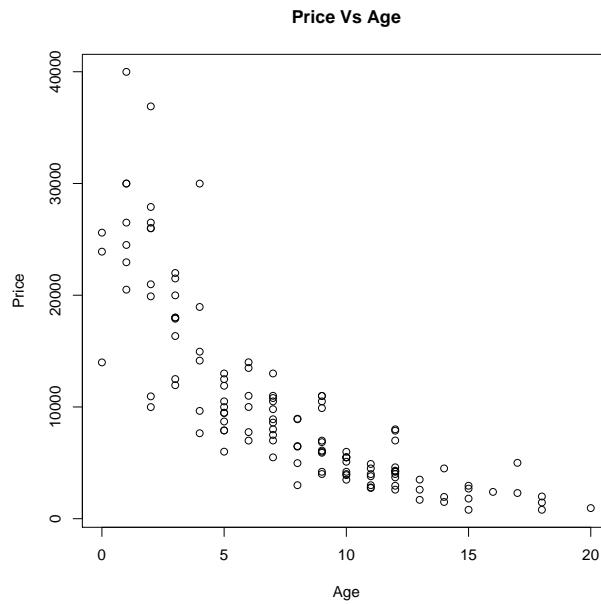
> oc.df = read.table("~/Desktop/old_car.txt",
+                     header = TRUE)
> str(oc.df)

'data.frame':   123 obs. of  2 variables:
 $ year : int  79 82 83 ...

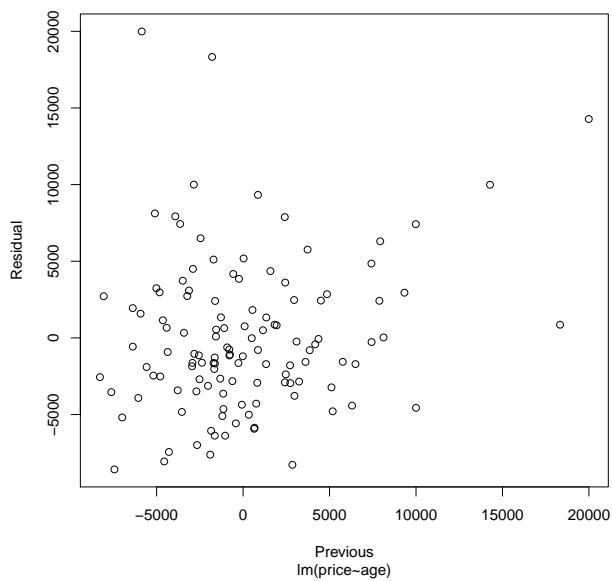
```

```
$ price: int 2950 5900 2999 ...
```

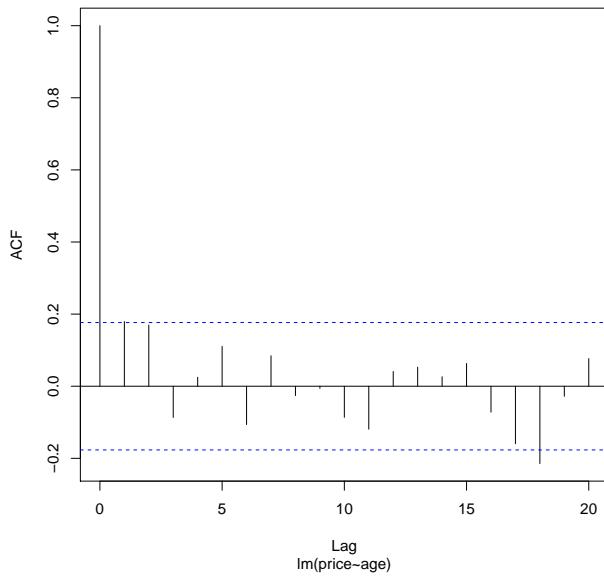
```
> # Create a new variable  
> age = 91 - oc.df$year
```

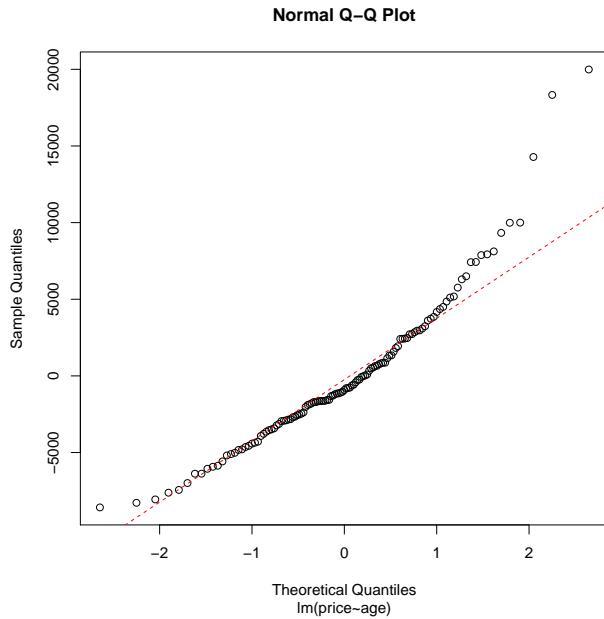


**Residual Vs Previous Residual**



**Residual Autocorrelation**





```

> ## Construct a preliminary model
> oc.LM = lm(price~age, data = oc.df)
> model = oc.LM
>
> ## Basic Residual Plot
> plot(fitted.values(model), residuals(model),
+       xlab = "Fitted Values", ylab = "Residuals",
+       main = "Residual Plot",
+       sub = "lm(price~age)")
> abline(a = 0, b = 0, lty = 2, col = "red")
>
> # Create groups according to fitted values
> FV.group = factor(age)
> age.unique = as.numeric(levels(FV.group))
>
> # Estimated Conditional Mean
> res.mean = tapply(residuals(model), FV.group, mean)

> # Adding Points and Lines
> tmp = data.frame(age = age.unique)
> x.tmp = predict(model, tmp)
> y.tmp = as.numeric(res.mean)
>
> points(x.tmp, y.tmp, col = "blue", pch = 16)
> lines(x.tmp, y.tmp, col = "blue", lty = 2)
>
```

```

> # Max and Min values
> res.max = tapply(residuals(model), FV.group, max)
> res.min = tapply(residuals(model), FV.group, min)
>
> # Adding Lines
> y.tmp = as.numeric(res.max)
> lines(x.tmp, y.tmp, lty = 4)
>
> y.tmp = as.numeric(res.min)
> lines(x.tmp, y.tmp, lty = 4)

> ## Residual Vs Previous Residual
> res = residuals(model)
> n = length(res)
>
> plot(res[-n], res[-1],
+       xlab = "Previous", ylab = "Residual",
+       main = "Residual Vs Previous Residual",
+       sub = "lm(price~age)")
>
> ## ACF Plot
> acf(res, main = "Residual Autocorrelation",
+       sub = "lm(price~age)")
>
> ## QQ Normal Plot
> qqnorm(res, sub = "lm(price~age)")
> qqline(res, col = 2, lty = 2)

```

Q: Why is using p-value to judge whether polynomial is needed a bad idea?

```

> oc.quad.LM = lm(price~age+I(age^2), data = oc.df)
> model = oc.quad.LM
> summary(model)

```

---

```

Call:
lm(formula = price ~ age + I(age^2), data = oc.df)

Residuals:
    Min      1Q      Median      3Q      Max 
-12974.9 -1442.4   -187.3   1241.6  16200.0 

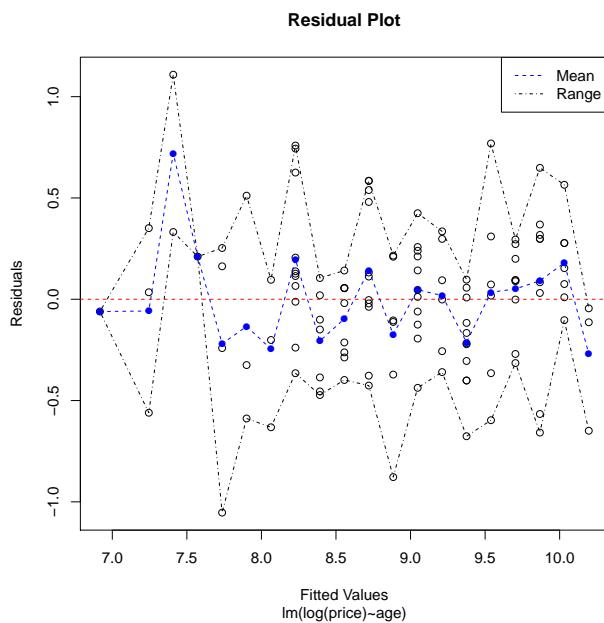
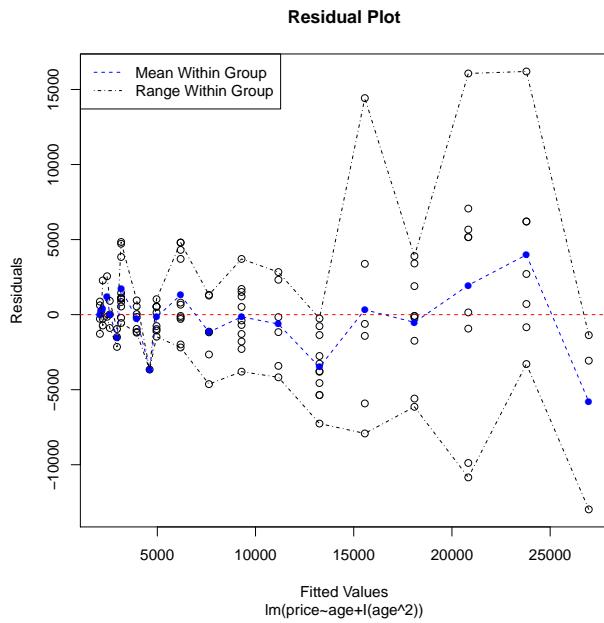
Coefficients:
            Estimate Std. Error t value Pr(>t)    
(Intercept) 26964.90     1066.35  25.287 < 2e-16 ***
age         -3283.16     271.81  -12.079 < 2e-16 ***
I(age^2)     108.27      15.16   7.142 7.68e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Residual standard error: 4164 on 120 degrees of freedom
Multiple R-squared:  0.7514,    Adjusted R-squared:  0.7473 
F-statistic: 181.4 on 2 and 120 DF,  p-value: < 2.2e-16

```

---

Q: What should we look at instead?



```

> oc.log.LM = lm(log(price)~age, data = oc.df)
> model = oc.log.LM # For reusing the code
>
> plot(fitted.values(model), residuals(model),

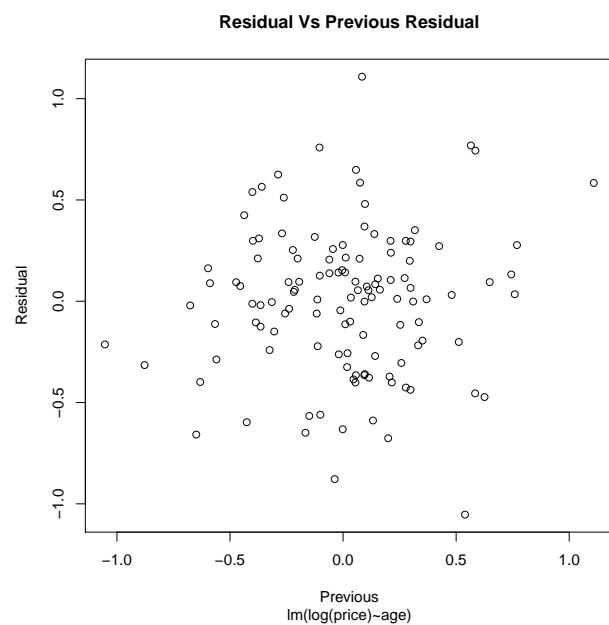
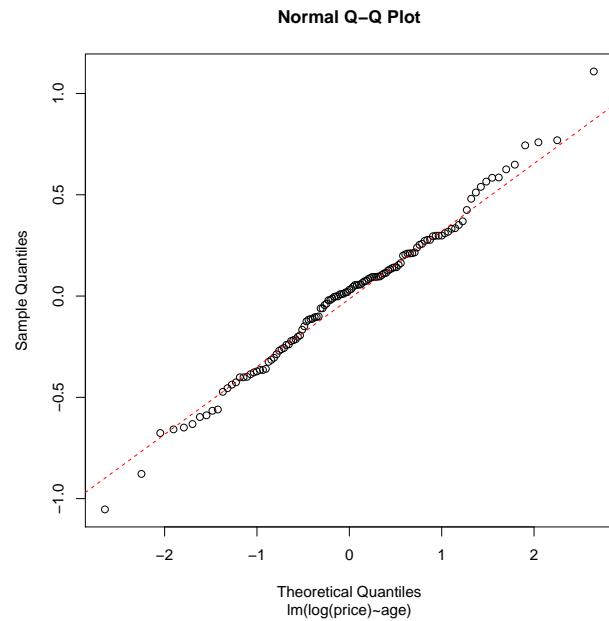
```

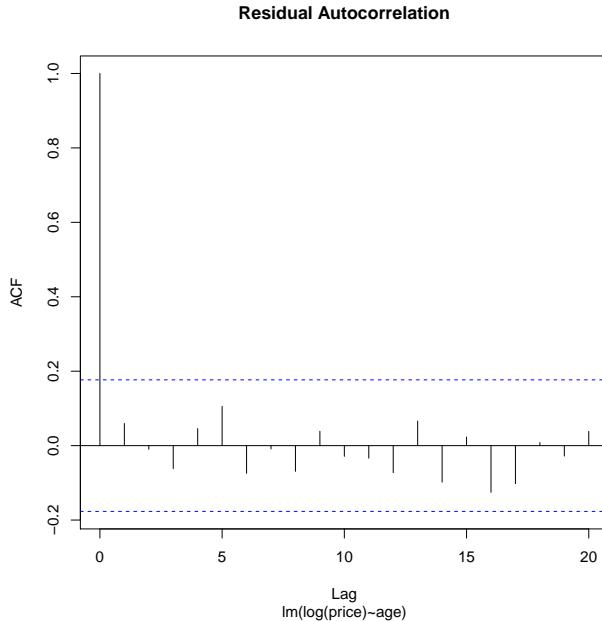
```

+      xlab = "Fitted Values", ylab = "Residuals",
+      main = "Residual Plot",
+      sub = "lm(log(price)~age)")
> abline(a = 0, b = 0, lty = 2, col = "red")
>
> # Computing mean residual within group
> FV.group = factor(age)
> res.mean = tapply(residuals(model), FV.group, mean)
> age.unique = as.numeric(levels(FV.group))
> tmp = data.frame(age = age.unique)
> x.tmp = predict(model, tmp)
> y.tmp = as.numeric(res.mean)
>
> # Adding the means, and a legend
> points(x.tmp, y.tmp, col = "blue", pch = 16)

> x.sort = sort(x.tmp) # To have not backward lines
> index = as.integer(names(sort(x.tmp)))
> lines(x.sort, y.tmp[index], col = "blue", lty = 2)
>
> # Computing residual range within group
> res.max = tapply(residuals(model), FV.group, max)
> res.min = tapply(residuals(model), FV.group, min)
>
> y.tmp = as.numeric(res.max)
> lines(x.sort, y.tmp[index], lty = 4)
>
>
> y.tmp = as.numeric(res.min)
> lines(x.sort, y.tmp[index], lty = 4)
>
>
> legend("topright", lty = c(2,4), col = c(4,1),
+        legend = c("Mean", "Range"))

```



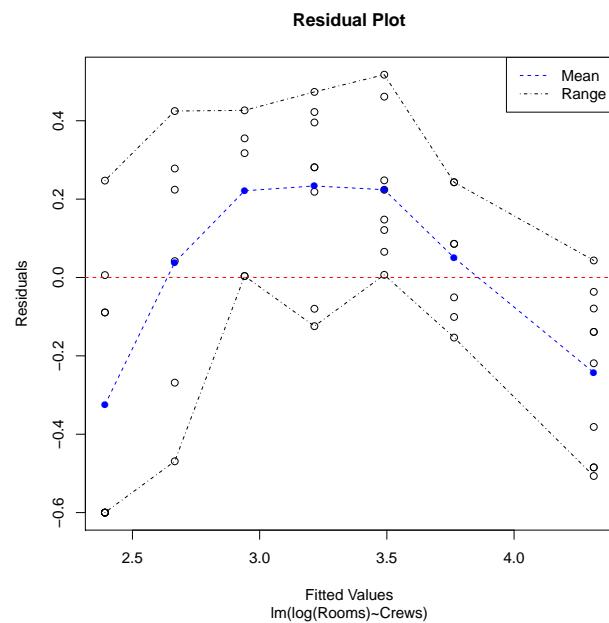
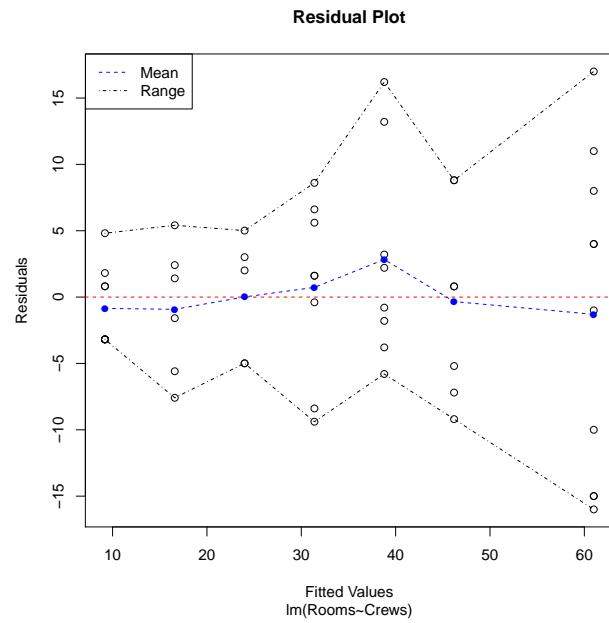


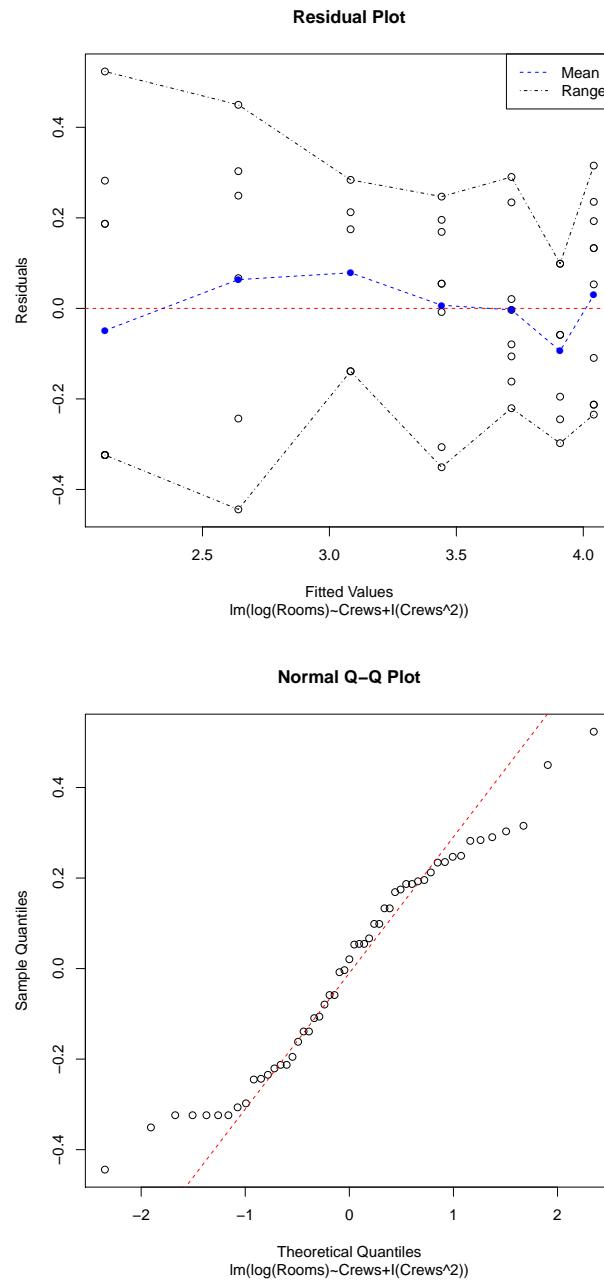
- In this case, it seems the natural logarithmic transformation not only solves the nonlinearity issue but solves the heteroskedasticity/nonnormality issue
- In general, when the variance is greater for larger fitted values than it is for smaller fitted values, a natural logarithmic transformation of the response variable often solves the heteroskedasticity issue.
- However, we cannot possibly expect log-transform or polynomial terms to always be the remedy, we need a more general and systematic method.
- To illustrate what can go wrong, consider the following dataset,

Crews    Number of works on the job  
 Rooms    Number of offices need to be cleaned

which is about 53 cases that a maintenance company did in the past.

```
> cleaning.df = read.table("~/Desktop/cleaning.txt",
+                           header = TRUE)
>
> cleaning.LM = lm(Rooms~Crews, data = cleaning.df)
```





1. The conditional mean is linear for all  $i$ , i.e.

$$\mathbb{E}[Y_i \mid X_i = x_i] = \beta_0 + \beta_1 x_i$$

2. The error has a zero mean and a constant but unknown variance for all  $i$ , i.e.

$$\mathbb{E}[\varepsilon_i | X_i] = 0 \quad \text{and} \quad \text{Var}[\varepsilon_i | X_i] = \sigma^2$$

4. The error terms  $\varepsilon_i$  follow a normal distribution.

- Recall the response is given by

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

- Therefore, a compact way to state the assumptions is

$$Y_i | X_i \sim \text{Normal}(\beta_0 + \beta_1 x_i, \sigma^2)$$

for all  $i$ .

- When we detect a possible violation of those three assumptions,

$$Y_i | X_i \sim \text{Normal}(\beta_0 + \beta_1 x_i, \sigma^2)$$

it means we have evidence that  $Y | X$  does not follow a normal distribution.

- The idea is to assume the response can be transformed by

$$Y_i^*(\lambda) = \begin{cases} \frac{Y_i^\lambda - 1}{\lambda}, & \text{if } \lambda \neq 0; \\ \ln Y_i, & \text{if } \lambda = 0. \end{cases}$$

so that the new response is conditionally normal, that is

$$Y_i^*(\lambda) | X_i \sim \text{Normal}(\beta_0 + \beta_1 x_i, \sigma^2)$$

Q: Can you see the transformation when  $\lambda = 0$  is the limiting case of  $\lambda \neq 0$ ?

Q: Do you notice it implicitly assumes  $Y$  is positive?

- For responses that might be negative, the following was proposed

$$Y_{i(\lambda_1, \lambda_2)}^* = \begin{cases} \frac{(Y_i + \lambda_2)^{\lambda_1} - 1}{\lambda_1}, & \text{if } \lambda \neq 0; \\ \ln(Y_i + \lambda_2), & \text{if } \lambda = 0. \end{cases}$$

the idea is essentially the same, so we will consider the positive case here.

Q: How can we determine which  $\lambda$  value to use?

- Since  $Y_i^* | X_i$  is assumed to be normal,

$$f_{Y_i^* | X_i}(y_i^* | x_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i^* - \beta_0 - \beta_1 x_i)^2}{2\sigma^2}\right)$$

with the independence assumption, the joint density is readily available, and

$$\mathcal{L}(\beta_0, \beta_1, \sigma^2) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^n (y_i^* - \beta_0 - \beta_1 x_i)^2\right)$$

Q: Do you remember the change of variable technique?

$$\mathcal{L}(\lambda, \beta_0, \beta_1, \sigma^2) = \left( \frac{1}{\sqrt{2\pi\sigma^2}} \right)^n \exp \left( -\frac{\sum_{i=1}^n (y_i^*(\lambda) - \beta_0 - \beta_1 x_i)^2}{2\sigma^2} \right) |J(\mathbf{y}; \lambda)|$$

where  $J(\mathbf{y}; \lambda)$  is the Jacobian of the transformation in terms of  $\mathbf{y}$ .

Q: What is the Jacobian in this case?

- Notice the likelihood function with  $\lambda$  is proportional to the one without  $\lambda$ .

$$\mathcal{L}(\beta_0, \beta_1, \sigma^2) = \left( \frac{1}{\sqrt{2\pi\sigma^2}} \right)^n \exp \left( -\frac{1}{2\sigma^2} \sum_{i=1}^n (y_i^* - \beta_0 - \beta_1 x_i)^2 \right)$$

- So the maximum likelihood estimates of  $\beta_0$ ,  $\beta_1$  and  $\sigma^2$  takes the usual form.
- Thus, for a given  $\lambda$  value, we can compute

$$b_0^* = \bar{y}^* - b_1^* \bar{x} \quad \text{where} \quad b_1^* = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i^* - \bar{y}^*)}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

and

$$\sigma_u^2 = \frac{n}{n-2} \sigma_{MLE}^2 = \frac{1}{n-2} \sum_{i=1}^n (y_i^* - \hat{y}_i^*)^2$$

where

$$\hat{y}_i^* = b_0^* + b_1^* x_i$$

Q: How can we find  $\lambda$ ?

- The negative log-likelihood function is given by

$$\ell(\lambda) = \frac{n}{2} \ln(2\pi) + \frac{n}{2} \log \sigma_{MLE}^2 + \frac{1}{2\sigma_{MLE}^2} \sum_{i=1}^n (y_i^* - \hat{y}_i^*)^2 - (\lambda - 1) \sum_{i=1}^n \ln y_i$$

- Notice terms in red also depend on  $\lambda$ , and no closed form exists for MLE of

$$\lambda$$

however, numerically we can determine  $\lambda$  reasonably well.

- Using the  $\lambda$  value that maximises the likelihood to transform  $Y$  is known as

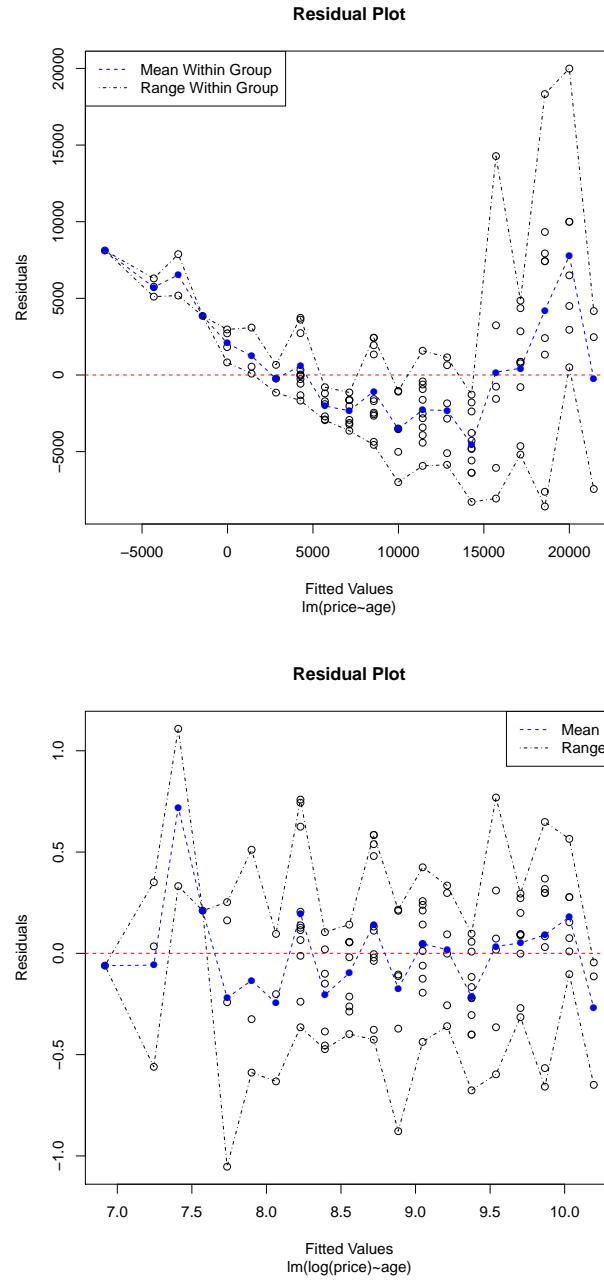
Box-Cox transformation

- Recall the following two models for the old Mazda cars dataset

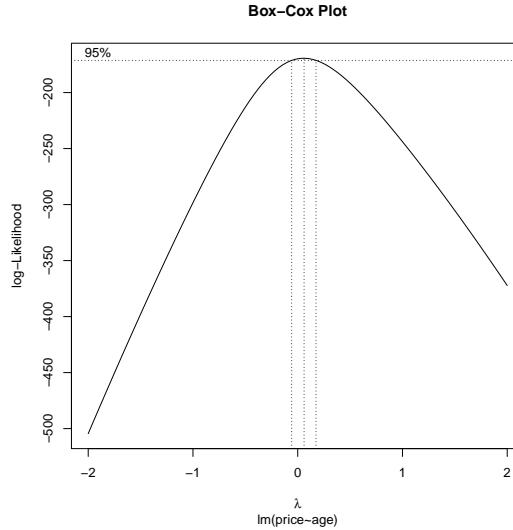
```

> oc.LM = lm(price~age, data = oc.df)
> oc.log.LM = lm(log(price)~age, data = oc.df)

```



Q: What does the following Box-Cox plot suggest?



- Imagine cases in which both variables,  $X$  and  $Y$ , need to be transformed.

$$Y_i^* \mid X_i^* \sim \text{Normal}(\beta_0 + \beta_1 x_i^*, \sigma^2)$$

that is, there are surely cases that the Box-Cox procedure cannot handle.

Q: Given two normal random variables,  $X$  and  $Y$ , what is the distribution of

$$Y \mid X$$

- It can be shown the conditional distribution is always normal,

$$Y \mid X \sim \text{Normal}\left(\mu_Y + \rho_{XY} \frac{\sigma_Y}{\sigma_X} (x - \mu_X), \sigma_Y^2 (1 - \rho_{XY}^2)\right)$$

Q: What does it mean in terms of simple linear regression and transformation?

- Combining this with the Box-Cox procedure, we have a systematic way to fix assumption 1, 2 and 4 by transforming both  $X$  and  $Y$ .
- Let us illustrate that by applying it to the following dataset

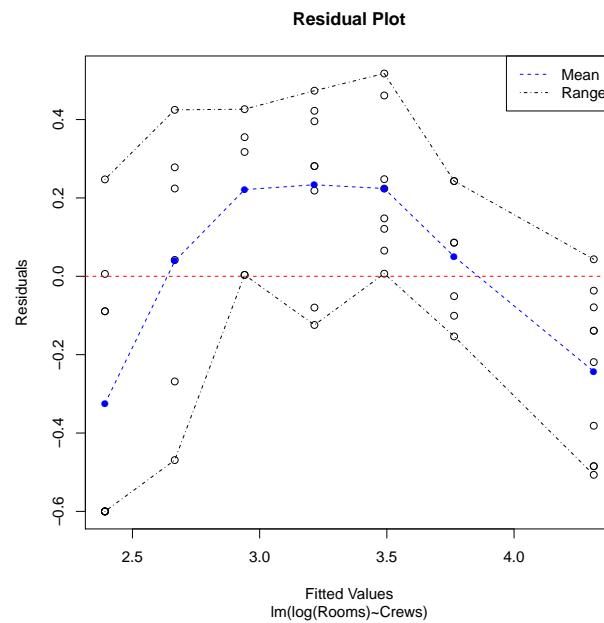
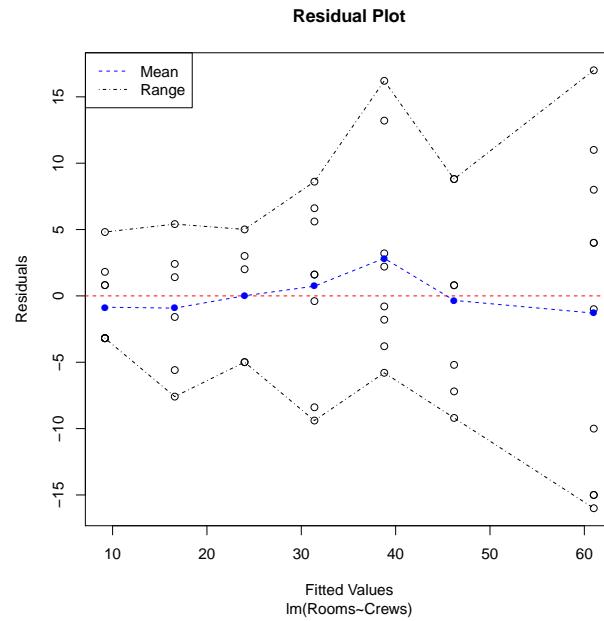
Crews Number of works on the job

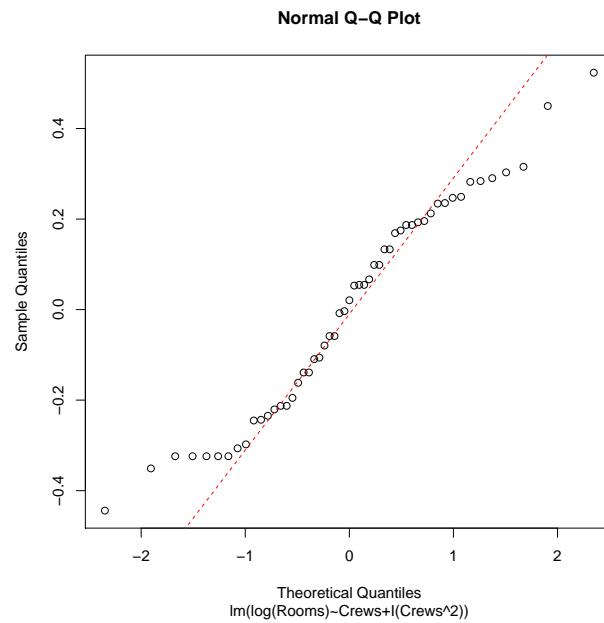
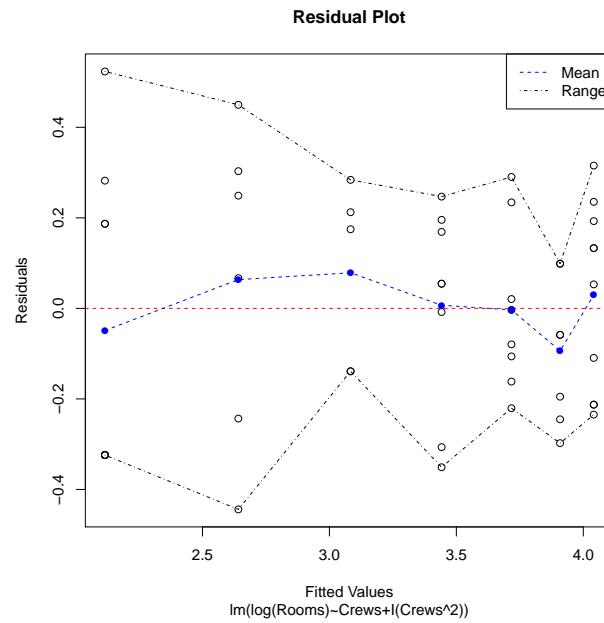
Rooms Number of offices need to be cleaned

- Recall we considered the following three models for this dataset,

```
> cleaning.LM = lm(Rooms~Crews, data = cleaning.df)
>
> cleaning.log.LM =
+   lm(log(Rooms)~Crews, data = cleaning.df)
>
> cleaning.log.sq.LM =
+   lm(log(Rooms)~Crews+I(Crews^2),
+       data = cleaning.df)
```

none of which is satisfactory!





- The numerical optimisation and the Box-Cox plot were done by

```
> # Box-Cox procedure is provided by this library
> library(MASS)
> bc = boxcox(oc.LM)
```

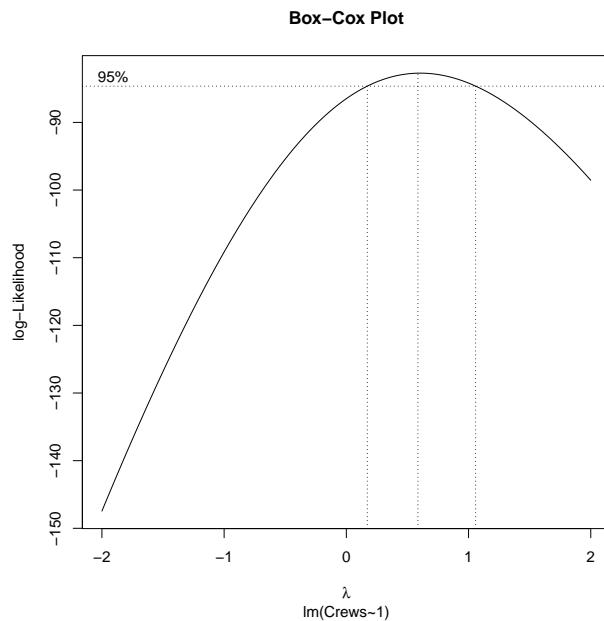
```
> title("Box-Cox Plot", sub = "lm(price~age)")
```

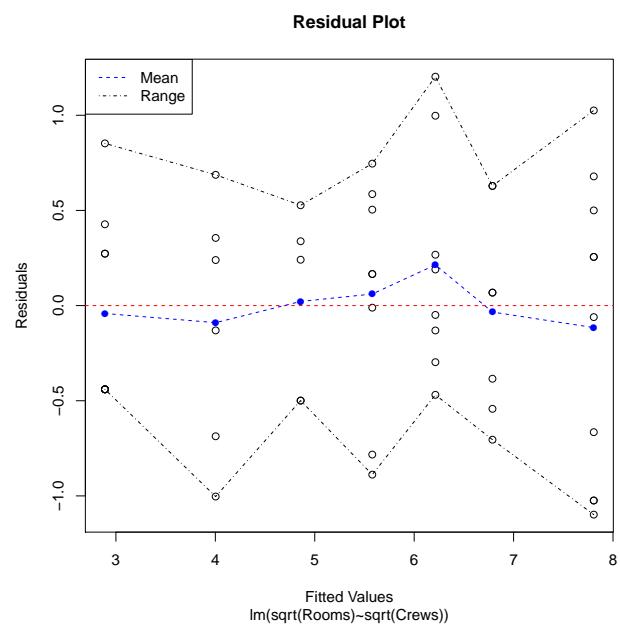
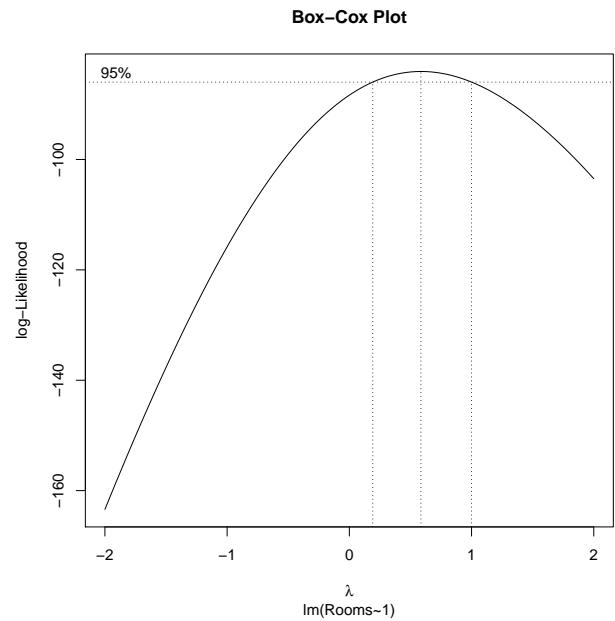
- For the cleaning dataset, we consider the best transformation for Crews

```
> x.LM = lm(Crews~1, data = cleaning.df)
> bc = boxcox(x.LM)
> title("Box-Cox Plot", sub = "lm(Crews~1)")
```

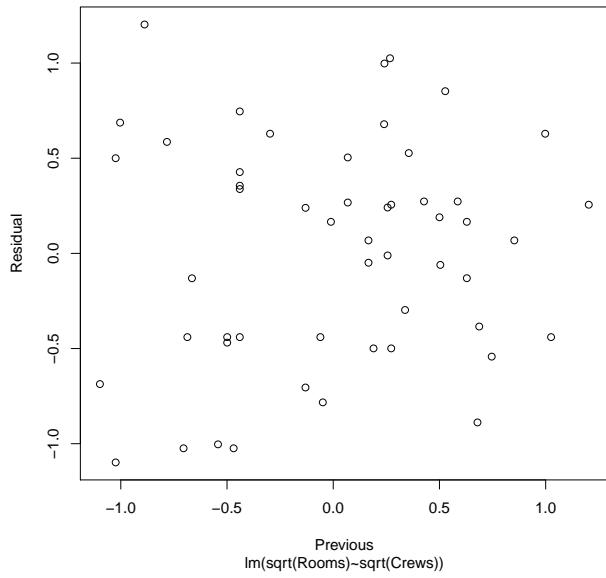
as well considering the best transformation for Rooms

```
> y.LM = lm(Rooms~1, data = cleaning.df)
> bc = boxcox(y.LM)
> title("Box-Cox Plot", sub = "lm(Rooms~1)")
```

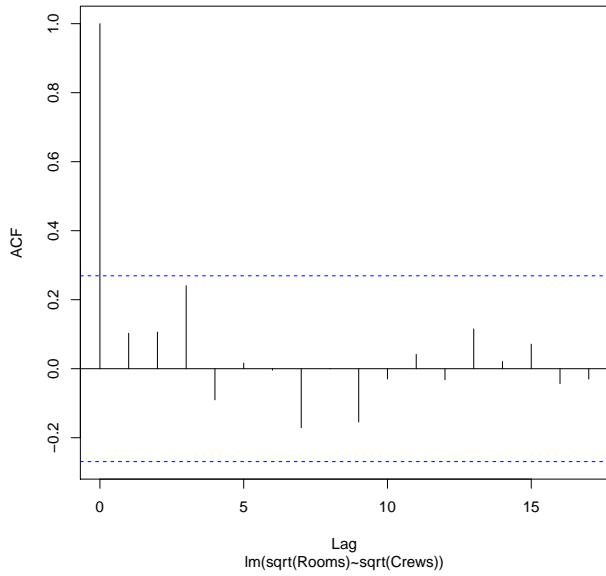


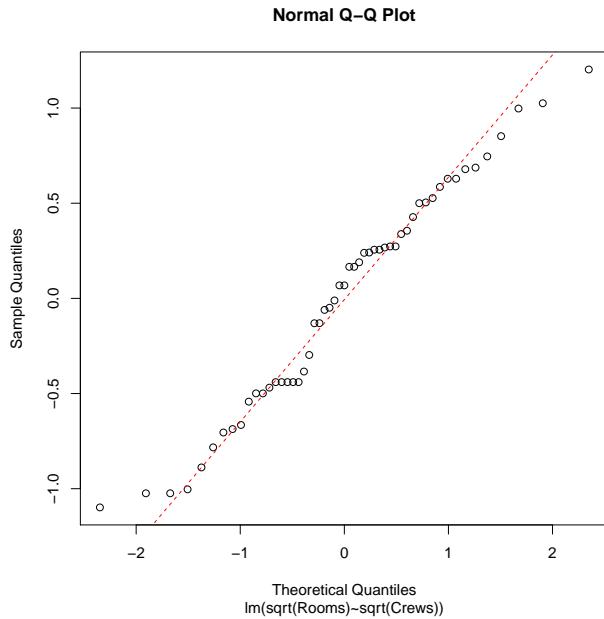


**Residual Vs Previous Residual**



**Residual Autocorrelation**





## 2.6 Inference

### A request from an insurance Company

- An insurance company wants to predict the damage (in \\$) to a home in a particular area if a fire occurs. The damage, and distance (in miles) from the fire station were recorded for 15 house fires in the area of interest.

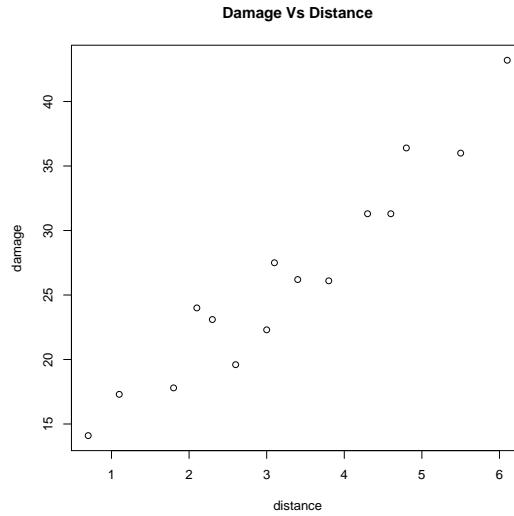
```
> fire.df = read.table("~/Desktop/fire.txt",
+                      header = TRUE)
```

```
> str(fire.df)
```

```
'data.frame': 15 obs. of 2 variables:
 $ distance: num 3.4 2.6 1.8 4.3 4.6 ...
 $ damage   : num 26.2 19.6 17.8 31.3 31.3 ...
```

- We are required to predict the damage for house fires that are 1 and 4 miles from the fire station.
- Visualisation

```
> with(fire.df, plot(distance, damage,
+                     main = "Damage Vs Distance"))
```



- Running a simple linear regression model

```
> fire.LM = lm(damage~distance, data = fire.df)
```

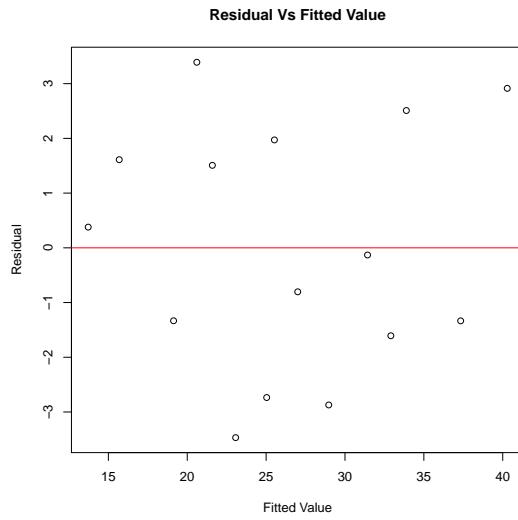
- Checking assumptions by looking at residuals

```
> fit = fire.LM$fitted.values
> res = fire.LM$residuals

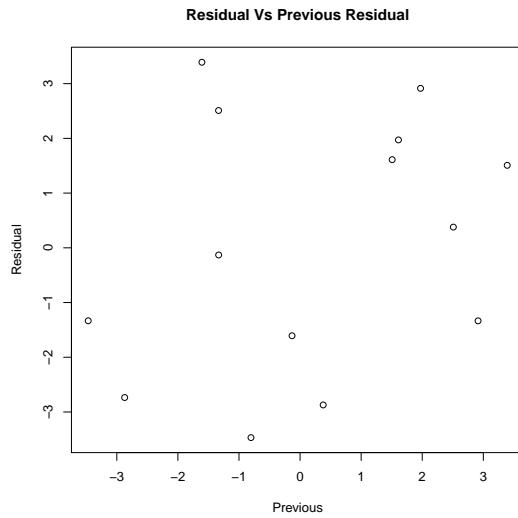
> # Residual Plot
> plot(fit, res, main = "Residual Vs Fitted Value",
+       xlab = "Fitted Value", ylab = "Residual")
> abline(h = 0, col = "red")

> # Correlation Plot
> plot(res[-nrow(fire.df)], res[-1],
+       xlab = "Previous", ylab = "Residual",
+       main = "Residual Vs Previous Residual")

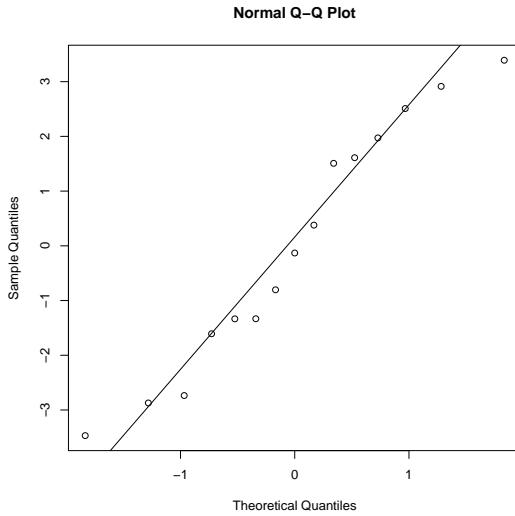
> # QQ plot
> qqnorm(res); qqline(res)
```



- There is no indication of nonlinearity, correlation or non-constant variance.



- There is no indication of autocorrelation.



- There is no indication of non-normality.
- Since the data size  $n$  is really small, we have to be careful with normality

```
> # Shapiro-Wilk
> shapiro.test(res)
```

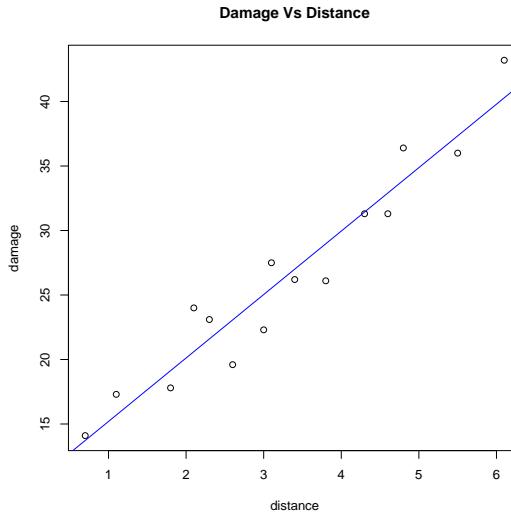
```
Shapiro-Wilk normality test

data: res
W = 0.94671, p-value = 0.4743
```

- Notice formal tests on normality are rather strict, use them if  $n$  is small.
- Since all the assumptions seem to be satisfied, we can now do inference.

```
> with(fire.df, plot(distance, damage,
+                      main = "Damage Vs Distance"))
> abline(fire.LM, col = "blue")
```

- So not only we can trust the line, but we can do other meaningful things...



- In general, the distribution of an estimator  $\hat{\theta}$  is known as the sampling distribution of  $\hat{\theta}$  to emphasize the fact that it is a random variable due to resampling, and the standard deviation of the estimator is known as the standard error.
- Consider a sample of normal random variable  $X$  with mean  $\mu$  and variance  $\sigma^2$ ,

$$\{x_1, x_2, \dots, x_n\}$$

then it can be shown, where s.e. =  $\sigma/\sqrt{n}$  is the standard error,

$$\frac{\bar{X} - \mu}{\sigma/\sqrt{n}} \sim \text{Normal}(0, 1)$$

- Suppose we would like to construct a

$$(1 - \alpha)100\%$$

confidence interval for  $\mu$ , and  $z^*$  is the value such that

$$\Pr[Z \leq z^*] = 1 - \alpha/2 \quad \text{where } Z \sim \text{Normal}(0, 1)$$

- Then, we have

$$\Pr \left[ -z^* \leq \frac{\bar{X} - \mu}{\sigma/\sqrt{n}} \leq z^* \right] = 1 - \alpha$$

$$\Pr \left[ \bar{X} - z^* \sigma / \sqrt{n} \leq \mu \leq \bar{X} + z^* \sigma / \sqrt{n} \right] = 1 - \alpha$$

from which the  $(1 - \alpha)100\%$  confidence interval for  $\mu$  in this case should be

$$(\bar{X} - z^* \sigma / \sqrt{n}, \bar{X} + z^* \sigma / \sqrt{n})$$

Of course, in practice,  $\sigma^2$  is not known, and is estimated using the following

$$S_X^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2$$

which can be shown to be unbiased. When  $S_X$  is used instead of the true  $\sigma$ ,

$$\frac{\bar{X} - \mu}{S_X / \sqrt{n}} \sim t_{n-1}$$

In a similar fashion,

$$\begin{aligned} \Pr \left[ -t^* \leq \frac{\bar{X} - \mu}{S_x / \sqrt{n}} \leq t^* \right] &\implies \Pr \left[ \bar{X} - t^* S_x / \sqrt{n} \leq \mu \leq \bar{X} + t^* S_x / \sqrt{n} \right] \\ &\implies (\bar{X} - t^* S_x / \sqrt{n}, \bar{X} + t^* S_x / \sqrt{n}) \end{aligned}$$

- Now back to simple linear regression, let us denote

$$s_x^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

you should have shown

$$\hat{\beta}_1 \sim \text{Normal} \left( \beta_1, \frac{\sigma^2}{(n-1)s_x^2} \right) \implies \frac{\hat{\beta}_1 - \beta_1}{\sigma / \sqrt{(n-1)s_x^2}} \sim \text{Normal}(0, 1)$$

when  $\sigma^2$  is not known, then it is usually estimated by

$$\sigma_u^2 = \frac{1}{n-2} \sum_{i=1}^n \hat{e}_i^2$$

and it can be shown that

$$\frac{\hat{\beta}_1 - \beta_1}{\sigma_u / \sqrt{(n-1)s_x^2}} \sim t_{n-2}$$

Unless otherwise specified, i.e.  $\sigma^2$  is known, the standard error of  $\beta_1$  referrs to

$$\text{se} [\hat{\beta}_1] = \frac{\sigma_u}{\sqrt{(n-1)s_x^2}}$$

Similarly, we have

$$\frac{\hat{\beta}_0 - \beta_0}{\sigma \sqrt{\frac{1}{n} + \frac{\bar{x}^2}{(n-1)s_x^2}}} \sim \text{Normal}(0, 1)$$

and

$$\frac{\hat{\beta}_0 - \beta_0}{\text{se}[\hat{\beta}_0]} \sim t_{n-2}$$

where

$$\text{se}[\hat{\beta}_0] = \sigma_u \sqrt{\frac{1}{n} + \frac{\bar{x}^2}{(n-1)s_x^2}}$$

```
> (fire.sm = summary(fire.LM))
```

```
Call:
lm(formula = damage ~ distance, data = fire.df)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.4682 -1.4705 -0.1311  1.7915  3.3915 

Coefficients:
            Estimate Std. Error t value Pr(>t)    
(Intercept) 10.2779    1.4203   7.237 6.59e-06 ***
distance    4.9193    0.3927  12.525 1.25e-08 *** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Residual standard error: 2.316 on 13 degrees of freedom
Multiple R-squared:  0.9235, Adjusted R-squared:  0.9176 
F-statistic: 156.9 on 1 and 13 DF,  p-value: 1.248e-08
```

```
> names(fire.sm)
```

```
[1] "call"          "terms"        "residuals"      "coefficients" "aliased"      
[6] "sigma"         "df"           "r.squared"     "adj.r.squared" "fstatistic"  
[11] "cov.unscaled"
```

Residual standard error is given by

$$\sigma_u = \sqrt{\frac{1}{n-2} \sum_{i=1}^n \hat{e}_i^2}$$

```
> (coeff.m = fire.sm$coefficients)
```

```
            Estimate Std. Error t value Pr(>t)    
(Intercept) 10.277929  1.4202778  7.236562 6.585564e-06
distance    4.919331  0.3927477 12.525421 1.247800e-08
```

```
> class(coeff.m)
```

```
[1] "matrix"
```

- Confidence interval for the parameters, e.g. for  $\hat{\beta}_1$

```
> b1_hat = coeff.mat[2,1]
>
> b1_se = coeff.mat[2,2]
>
> (b1_ci = b1_hat + c(-1, 1) * b1_se *
+   qt(0.975, fire.LM$df.residual))

[1] 4.070851 5.767811
```

- In practice, we use the following to obtain the confidence interval

```
> confint(fire.LM, "(Intercept)", level = 0.95)
```

2.5 %	97.5 %
(Intercept)	7.209605 13.34625

```
> confint(fire.LM, "distance", level = 0.95)
```

2.5 %	97.5 %
distance	4.070851 5.767811

- The following gives the predictions according to our model.

```
> pred.df = data.frame(distance = c(1,4))
> predict(fire.LM, pred.df)
```

1	2
15.19726	29.95525

- Recall the optimal prediction of a random variable is the mean. Thus

$$\hat{y}_{n+1}^* \mid X_{n+1} = b_0 + b_1 x_{n+1} \quad \text{where} \quad b_0 = \bar{y} - b_1 \bar{x}$$

$$b_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})}$$

the star is here to distinguish it from the fitted value that uses  $n+1$  observations.

## 2.7 Prediction

Q: How can we construct a “confidence interval” for our prediction?

$$(\hat{y}_{n+1}^* - c, \hat{y}_{n+1}^* + c)$$

Q: How can we find the constant  $c$  for a given significant level?

Q: Why R has two types of “confidence interval” for prediction?

```
> pred.df = data.frame(distance = c(1,4))
> predict(fire.LM, pred.df, interval = "confidence")
```

	fit	lwr	upr
1	15.19726	12.87092	17.52360
2	29.95525	28.52604	31.38446

```
> predict(fire.LM, pred.df, interval = "predict")
```

	fit	lwr	upr
1	15.19726	9.67879	20.71573
2	29.95525	24.75100	35.15951

- Now consider the random variable conditioning on  $X_1, X_2, \dots, X_{n+1}$ ,

$$\hat{Y}_{n+1} = \hat{\beta}_0 + \hat{\beta}_1 X_{n+1}$$

where  $\hat{\beta}_0$  and  $\hat{\beta}_1$  are the sources of randomness.

- Recall we have derived that

$$\hat{\beta}_1 = \beta_1 + \frac{\sum_{i=1}^n (X_i - \bar{X}_n) \varepsilon_i}{\sum_{i=1}^n (X_i - \bar{X}_n)^2}$$

and

$$\begin{aligned} \hat{\beta}_0 &= \bar{Y}_n - \hat{\beta}_1 \bar{X}_n \\ &= \beta_0 + \beta_1 \bar{X}_n + \bar{\varepsilon}_n - \beta_1 \bar{X}_n - \frac{\bar{X}_n \sum_{i=1}^n (X_i - \bar{X}_n) \varepsilon_i}{\sum_{i=1}^n (X_i - \bar{X}_n)^2} \\ &= \beta_0 + \bar{\varepsilon}_n - \frac{\bar{X}_n \sum_{i=1}^n (X_i - \bar{X}_n) \varepsilon_i}{\sum_{i=1}^n (X_i - \bar{X}_n)^2} \end{aligned}$$

- Therefore, the estimated conditional mean is given by

$$\hat{Y}_{n+1} = \beta_0 + \beta_1 X_{n+1} + \bar{\varepsilon}_n + \frac{(X_{n+1} - \bar{X}_n) \sum_{i=1}^n (X_i - \bar{X}_n) \varepsilon_i}{\sum_{i=1}^n (X_i - \bar{X}_n)^2}$$

from which, we can conclude  $\hat{Y}_{n+1}$  is a normal random variable with

$$\mathbb{E} [\hat{Y}_{n+1} | X_1, \dots, X_n, X_{n+1}] = \beta_0 + \beta_1 X_{n+1}$$

- Recall the formula for the variance of linear combination of random variables

$$\text{Var} \left[ \sum_{i=1}^n a_i W_i \right] = \sum_{i=1}^n a_i^2 \text{Var} [W_i] + 2 \sum_{1 \leq i < j \leq n} a_i a_j \text{Cov} [W_i, W_j]$$

Using which, we have

$$\begin{aligned} & \text{Var} [\hat{Y}_{n+1} | X_1, \dots, X_n, X_{n+1}] \\ &= \sum_{i=1}^n \frac{1}{n^2} \sigma^2 + \frac{(X_{n+1} - \bar{X}_n)^2 \sum_{i=1}^n (X_i - \bar{X}_n)^2 \sigma^2}{\left( \sum_{i=1}^n (X_i - \bar{X}_n)^2 \right)^2} \\ & \quad + 2 \sum_{1 \leq i < j \leq n} \frac{1}{n} \frac{(X_{n+1} - \bar{X}_n) (X_j - \bar{X}_n)}{\sum_{\ell=1}^n (X_\ell - \bar{X}_n)^2} \text{Cov} [\varepsilon_i, \varepsilon_j] \\ &= \frac{\sigma^2}{n} + \frac{(X_{n+1} - \bar{X}_n)^2 \sigma^2}{\sum_{i=1}^n (X_i - \bar{X}_n)^2} + \frac{2\sigma^2}{n} \frac{\sum_{i=1}^n (X_i - \bar{X}_n)}{\sum_{i=1}^n (X_i - \bar{X}_n)^2} \\ &= \sigma^2 \left( \frac{1}{n} + \frac{(X_{n+1} - \bar{X}_n)^2}{\sum_{i=1}^n (X_i - \bar{X}_n)^2} \right) + 0 \\ &= \frac{\sigma^2}{n-1} \left( \frac{n-1}{n} + \frac{(X_{n+1} - \bar{X}_n)^2}{\frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X}_n)^2} \right) \\ &= \frac{\sigma^2}{n-1} \left( \frac{n-1}{n} + \frac{(X_{n+1} - \bar{X}_n)^2}{S_X^2} \right) \end{aligned}$$

- The variance of  $\hat{Y}_{n+1}^*$  can be derived based on the variance of  $\hat{\beta}_1$ ,

$$\text{Var} [\hat{Y}_{n+1}^* | X_1, X_2, \dots, X_n, X_{\textcolor{red}{n+1}}] = \frac{\sigma^2}{n-1} \left( \frac{n-1}{n} + \frac{(x_{n+1} - \bar{x}_n)^2}{s_x^2} \right)$$

and the sampling distribution of  $\hat{Y}_{n+1}^*$  is normal given  $\sigma^2$ .

- Assuming our model is correct,

$$Y = \beta_0 + \beta_1 X + \epsilon$$

the variance of  $Y_{n+1}$  conditional on  $X_{n+1}$  is simply

$$\text{Var} [Y_{n+1} | X_{\textcolor{red}{n+1}}] = \sigma^2$$

Q: What does the following variance capture?

$$\text{Var} [\hat{Y}_{n+1}^* + \varepsilon_{n+1} | X_1, X_2, \dots, X_{\textcolor{red}{n+1}}] = \frac{\sigma^2}{n-1} \left( n - \frac{1}{n} + \frac{(x - \bar{x})^2}{s_x^2} \right)$$

Q: Can you distinguish various random processes in simple linear regression?

Q: Can you distinguish the two classes of variability in the response variable?

$$\text{TSS} = \sum_{i=1}^n (y_i - \bar{y})^2 = \underbrace{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2}_{\text{ESS}} + \underbrace{\sum_{i=1}^n \hat{e}_i^2}_{\text{RSS}}$$

- Subtract and add a term of  $\hat{y}_i$

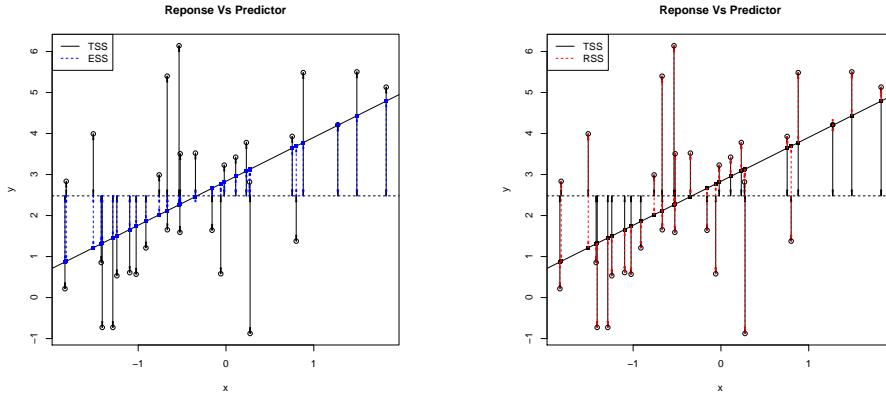
$$\begin{aligned} \sum_{i=1}^n (y_i - \bar{y})^2 &= \sum_{i=1}^n (y_i - \hat{y}_i + \hat{y}_i - \bar{y})^2 \\ &= \sum_{i=1}^n \left( (y_i - \hat{y}_i)^2 + 2(y_i - \hat{y}_i)(\hat{y}_i - \bar{y}) + (\hat{y}_i - \bar{y})^2 \right) \\ &= \sum_{i=1}^n (\hat{y}_i - \bar{y})^2 + 2 \sum_{i=1}^n \hat{e}_i (b_0 + b_1 x_i - \bar{y}) + \sum_{i=1}^n \hat{e}_i^2 \\ &= \sum_{i=1}^n (\hat{y}_i - \bar{y})^2 + \sum_{i=1}^n \hat{e}_i^2 \\ &\quad + 2 \left( b_0 \sum_{i=1}^n \hat{e}_i + b_1 \sum_{i=1}^n \hat{e}_i x_i - \bar{y} \sum_{i=1}^n \hat{e}_i \right) \end{aligned}$$

- Recall the two equations from setting the first derivatives equal to zero means

$$\sum_{i=1}^n \hat{e}_i = 0 \quad \text{and} \quad \sum_{i=1}^n \hat{e}_i x_i = 0$$

thus

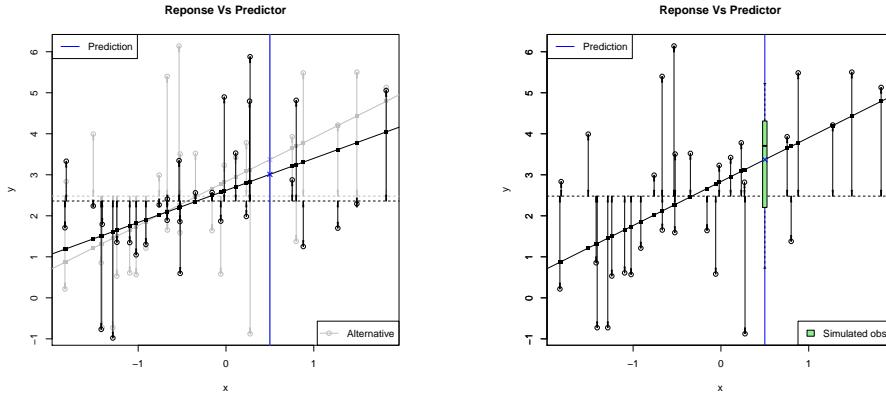
$$\text{TSS} = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2 + \sum_{i=1}^n \hat{e}_i^2$$



- The coefficient of determination

$$R^2 = \frac{\text{ESS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}}$$

- The variability in the value of  $Y_{n+1}^*$  has two layers given  $\{x_1, x_2, \dots, x_{n+1}\}$ .



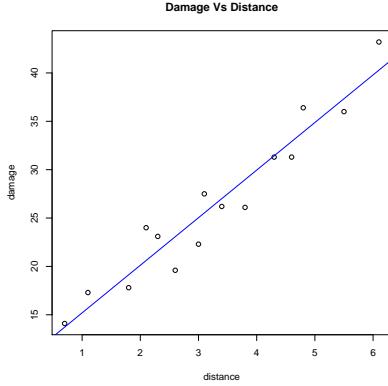
## Reporting

- When using regression in a project, you report should consists of two sections
  1. Technical Notes
    - Exploratory Analysis
    - Model Specification
    - Checking Assumptions
    - Statistical Inference
  2. Executive Summary
    - Overall quality of the model
    - Explaining the relationship between the response and the predictors
    - Making predictions
    - Addressing specific questions the project is about

## Technical Notes

- **Exploratory Analysis:**

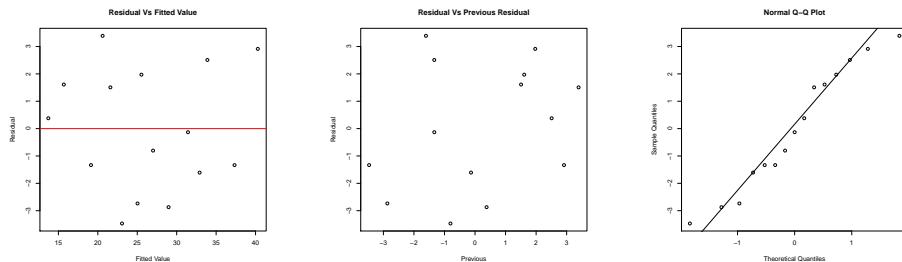
The scatter plot of fire damage versus distance shows a strong, increasing, linear relationship. The greater the distance from the fire station, the greater the mean amount of damage that is caused by the house fire.



## Technical Notes

- **Checking Assumptions:**

The observations appear to be independent. The plot of residuals versus fitted values shows random scatter about 0. The Normal Q-Q plot shows the points lying close to the straight line indicating that the errors could have a normal distribution. The Shapiro-Wilk test provides no evidence against the hypothesis that the errors have a normal distribution (P-value = 0.4743).



## Technical Notes

- **Statistical Inference:**

The F-test for regression provides extremely strong evidence against the hypothesis that distance from the fire station is not related to the amount of damage ( $P$ -value =  $1.248 \times 10^{-8}$ ). The Multiple  $R^2$  is 0.9235 indicating that 92% of the variation in damage is explained by the variation in distance from the fire station, so the model should be very accurate for prediction. We have extremely strong evidence against the hypothesis that the intercept is equal to 0 ( $P$ -value =  $6.59 \times 10^{-6}$ ). We have extremely strong evidence against the hypothesis that the slope coefficient associated with distance is equal to 0 ( $P$ -value =  $1.25 \times 10^{-8}$ ).

Coefficients:					
	Estimate	Std. Error	t value	Pr(>t)	
(Intercept)	10.2779	1.4203	7.237	6.59e-06 ***	
distance	4.9193	0.3927	12.525	1.25e-08 ***	
---					
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1					
Residual standard error: 2.316 on 13 degrees of freedom					
Multiple R-squared: 0.9235, Adjusted R-squared: 0.9176					
F-statistic: 156.9 on 1 and 13 DF, p-value: 1.248e-08					

## Executive Summary

Our model explains 92% of the variation in house fire damage and should therefore be a very accurate model for prediction. We have extremely strong evidence that as the distance from the fire station increases, the average amount of damage increases. We estimate that if the house next to the fire station catches fire, the mean fire damage will be between \$7,200 and \$13,300. We estimate that for each additional mile from the fire station, the mean fire damage increases by between \$4,100 and \$5,800. Using our model, we predict that if a new fire occurs in a house that is 1 mile from the fire station, the damage will be between \$9,700 and \$20,700. The mean damage for house fires that are 1 mile from the fire station will be between \$12,900 and \$17,500. For a house that is 4 miles from the fire station, we predict the damage will be between \$24,800 and \$35,200. The mean damage for house fires that are 4 miles from the fire station will be between \$28,500 and \$31,400.

## 3 Multiple Linear Regression

### 3.1 Matrix Form

- In multiple linear regression, we explore the relation between the response

$$Y_i \quad i = 1, 2, \dots, n$$

and two or more predictors/explanatory variables,  $k$  of them in general,

$$X_{i1}, \quad X_{i2}, \quad \dots \quad X_{ij}, \quad \dots \quad X_{ik} \quad i = 1, 2, \dots, n$$

where we are assuming that we observed  $n$  cases, and

1. The conditional mean of the response is given by the following for all  $i$

$$\mathbb{E}[Y_i | X_{i1}, X_{i2}, \dots, X_{ik}] = \mathbb{E}[Y_i | \mathbf{X}_i] = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_k x_{ik}$$

2. The errors have zero mean and constant variance

$$\mathbb{E}[\varepsilon_i | \mathbf{X}] = 0 \quad \text{and} \quad \text{Var}[\varepsilon_i | \mathbf{X}] = \sigma^2 \quad \text{where} \quad \varepsilon_i = Y_i - \mathbb{E}[Y_i | \mathbf{X}_i]$$

3. The errors are independent of  $\mathbf{X}_i$ , and of each other, for all  $i$ .

4. The errors follow a normal distribution.

- In matrix form, we have

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1k} \\ 1 & x_{21} & \cdots & x_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nk} \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}, \quad \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

Q: How to interpret the regression coefficients  $\beta_0, \beta_1, \dots, \beta_n$  now?

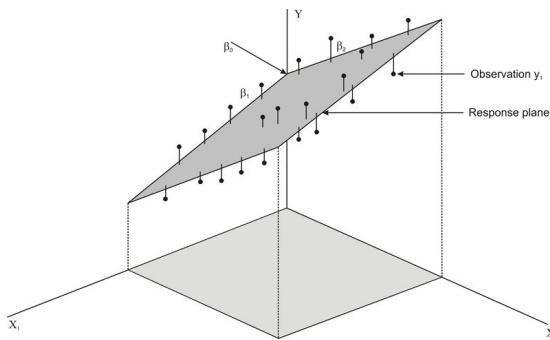
- Suppose there are only two predictors

$$m(x_{i1}, x_{i2}) = \mathbb{E}[Y_i | X_{i1} = x_{i1}, X_{i2} = x_{i2}] = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}$$

$$\Rightarrow \frac{\partial m}{\partial x_{i1}} = \beta_1; \quad \frac{\partial m}{\partial x_{i2}} = \beta_2; \quad m(0, 0) = \beta_0$$

The regression coefficient represents the increase in the mean response associated with a unit increase in the predictor variable, *provided other predictor variables are held fixed.*

- The regression coefficients in multiple regression are sometimes called partial regression coefficients to emphasise that their interpretation requires that the other variables should be held fixed.



- Notice the essential idea is the same, but we are fitting a plane/hyperplane instead of a line to the data by estimating the parameters in

$$\mathbb{E}[Y | \mathbf{X}]$$

- Most of what we have done can be extended with a bit of linear algebra

$$\mathbf{y} = \mathbf{X}\hat{\boldsymbol{\beta}} + \hat{\mathbf{e}}$$

we choose  $\hat{\boldsymbol{\beta}} = \mathbf{b}$  that minimises residual sum of squares, which is given by

$$\begin{aligned} f(b_0, b_1, \dots, b_k) &= \hat{\mathbf{e}}^T \hat{\mathbf{e}} = (\mathbf{y} - \mathbf{X}\mathbf{b})^T (\mathbf{y} - \mathbf{X}\mathbf{b}) \\ &= (\mathbf{y}^T - \mathbf{b}^T \mathbf{X}^T) (\mathbf{y} - \mathbf{X}\mathbf{b}) \\ &= \mathbf{y}^T \mathbf{y} - \mathbf{y}^T \mathbf{X}\mathbf{b} - \mathbf{b}^T \mathbf{X}^T \mathbf{y} + \mathbf{b}^T \mathbf{X}^T \mathbf{X}\mathbf{b} \end{aligned}$$

- Notice the all terms are scalars, thus equal

$$\mathbf{b}^T \mathbf{X}^T \mathbf{y} = (\mathbf{y}^T \mathbf{X}\mathbf{b})^T = \text{scalar} = \mathbf{y}^T \mathbf{X}\mathbf{b}$$

- Hence the function we need to minimise is given by

$$f(\mathbf{b}) = \mathbf{y}^T \mathbf{y} - 2\mathbf{b}^T \mathbf{X}^T \mathbf{y} + \mathbf{b}^T \mathbf{X}^T \mathbf{X}\mathbf{b}$$

- Recall to minimise a function,

$$f(\mathbf{b}) = \mathbf{y}^T \mathbf{y} - 2\mathbf{b}^T \mathbf{X}^T \mathbf{y} + \mathbf{b}^T \mathbf{X}^T \mathbf{X}\mathbf{b}$$

we set the gradient to zero,

$$\nabla f = 0 - 2\mathbf{X}^T \mathbf{y} + 2\mathbf{X}^T \mathbf{X}\mathbf{b}$$

Setting this to zero, we have

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

- Hence, the **fitted value** is given by

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = \mathbf{P}\mathbf{y}$$

and the residual can be found using

$$\hat{\mathbf{e}} = \mathbf{y} - \hat{\mathbf{y}} = (\mathbf{I} - \mathbf{P})\mathbf{y}$$

- With more linear algebra, we have

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}) = \boldsymbol{\beta} + (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \boldsymbol{\varepsilon}$$

which means it is unbiased as expected,

$$\mathbb{E} [\hat{\boldsymbol{\beta}} | \mathbf{X}] = \boldsymbol{\beta} + (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbb{E} [\boldsymbol{\varepsilon} | \mathbf{X}] = \boldsymbol{\beta}$$

- The variance is given by

$$\begin{aligned} \text{Var} [\hat{\boldsymbol{\beta}} | \mathbf{X}] &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \text{Var} [\boldsymbol{\varepsilon} | \mathbf{X}] ((\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T)^T \\ &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \sigma^2 \mathbf{I} \mathbf{X} (\mathbf{X}^T \mathbf{X})^{-1} = \sigma^2 (\mathbf{X}^T \mathbf{X})^{-1} \end{aligned}$$

- With the normal assumption, we see

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

- For the estimation  $\sigma^2$ , we only need adjust the scalar a bit

$$\hat{\sigma}_u^2 = \frac{1}{n - k - 1} \hat{\mathbf{e}}^T \hat{\mathbf{e}} \quad \text{where } \hat{\mathbf{e}} = (\mathbf{I} - \mathbf{P}) \mathbf{y}$$

so that it is unbiased as well as being consistent as before.

- Considering the residual under resampling, it can be shown

$$\hat{\mathbf{e}} = (\mathbf{I} - \mathbf{P}) \mathbf{Y} = (\mathbf{I} - \mathbf{P}) (\mathbf{X} \beta + \varepsilon)$$

is an unbiased and consistent estimator of the error  $\mathbf{e}$ , and the variance is

$$\begin{aligned} \text{Var} [\hat{\mathbf{e}} | \mathbf{X}] &= \text{Var} [(\mathbf{I} - \mathbf{P}) (\mathbf{X} \beta + \varepsilon) | \mathbf{X}] \\ &= (\mathbf{I} - \mathbf{P}) \text{Var} [\varepsilon | \mathbf{X}] (\mathbf{I} - \mathbf{P})^T \\ &= (\mathbf{I} - \mathbf{P}) \sigma^2 \mathbf{I} (\mathbf{I} - \mathbf{P})^T = \sigma^2 (\mathbf{I} - \mathbf{P}) \end{aligned}$$

- Thus with the normal assumption, we have

$$\hat{\mathbf{e}} \sim \text{Normal} (\mathbf{0}, \sigma^2 (\mathbf{I} - \mathbf{P}))$$

## 3.2 Diagnostics

Q: How can we check assumption 1. graphically? What do you expect to see?

- The conditional mean of the response is given by the following for all  $i$

$$\begin{aligned} \mathbb{E}[Y_i | X_{i1}, X_{i2}, \dots, X_{ik}] &= \mathbb{E}[Y_i | \mathbf{X}_i] \\ &= \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_k x_{ik} \end{aligned}$$

- Since the conditional mean is not available, but the fitted value

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + \dots + \hat{\beta}_k x_{ik}$$

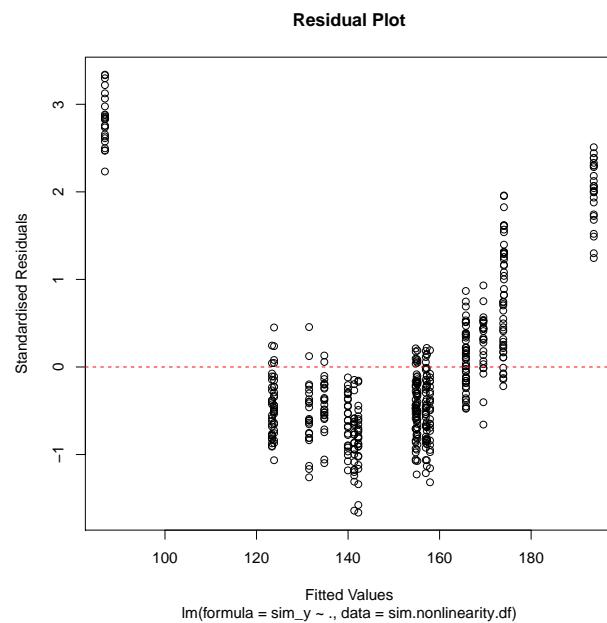
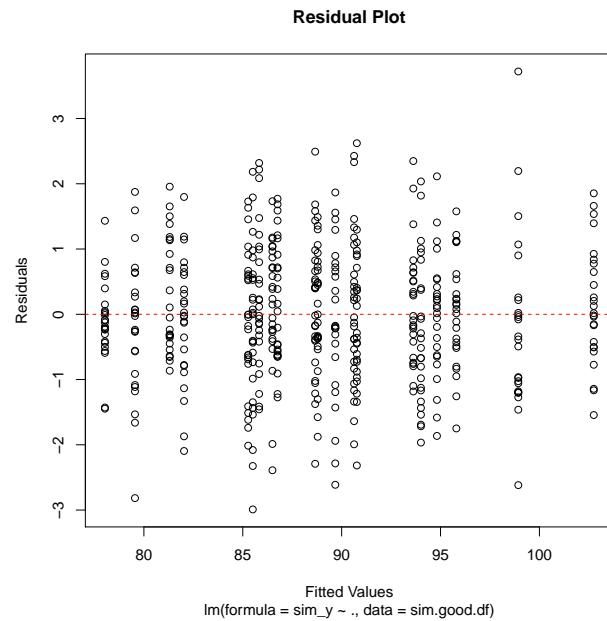
our estimate of it is, thus  $y_i$  is expected to be scattered around it if 1. holds.

- This is hardly different from the case of simple linear regression, we look at

the residual plot:  $\hat{e}_i$  Vs  $\hat{y}_i$

instead of  $y_i$  Vs  $\hat{y}_i$  to avoid the visual distraction of a sloping line.

- Having a random scatter of points round  $x$ -axis means we have no evidence against 1.. If there is a pattern, then there is evidence against 1..



Q: How can we check assumption 2. graphically? What do you expect to see?

- 2. The errors have zero mean and constant variance for all  $i$

$$\mathbb{E} [\varepsilon_i | \mathbf{X}] = 0 \quad \text{and} \quad \text{Var} [\varepsilon_i | \mathbf{X}] = \sigma^2$$

- Recall for simple linear regression, we have

$$\text{Var} [Y_i - \hat{Y}_i | X_i] = \sigma^2 \left( 1 - \frac{1}{n} - \frac{(x_i - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \right)$$

- For multiple linear regression, we have derived the following earlier

$$\text{Var} [\mathbf{Y} - \hat{\mathbf{Y}} | \mathbf{X}] = \sigma^2 (\mathbf{I} - \mathbf{P})$$

Q: What does this formula suggest?

- Internally studentised residual/standardised residual is given by

$$\hat{e}'_i = \frac{\hat{e}_i}{\hat{\sigma}_u \sqrt{1 - p_{ii}}}$$

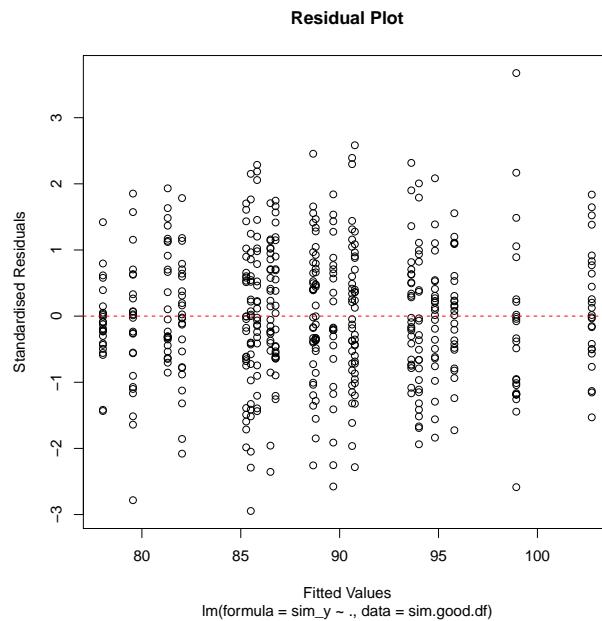
where  $p_{ii}$  is the  $i$ th diagonal element of  $\mathbf{P}$  and

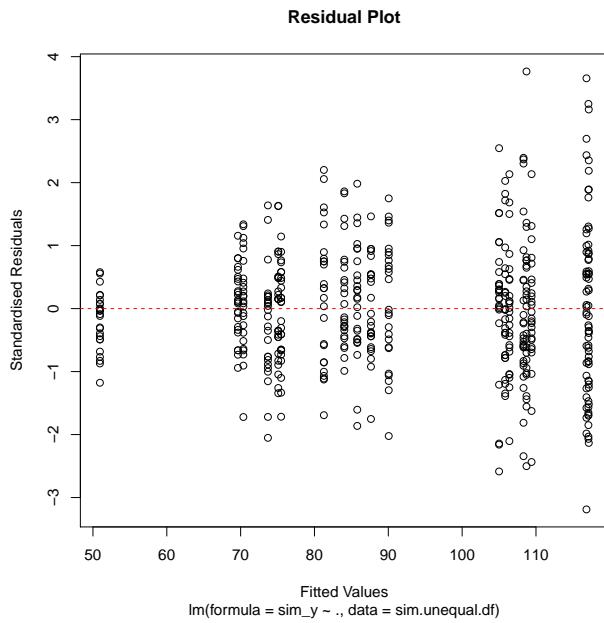
$$\begin{aligned} \hat{\sigma}_u^2 &= \frac{1}{n - k - 1} \hat{\mathbf{e}}^T \hat{\mathbf{e}} \\ &= \frac{1}{n - k - 1} \mathbf{y}^T (\mathbf{I} - \mathbf{P}) \mathbf{y} \end{aligned}$$

Q: What do you expect to see in a plot of

$$\hat{e}'_i \quad \text{VS} \quad \hat{y}_i$$

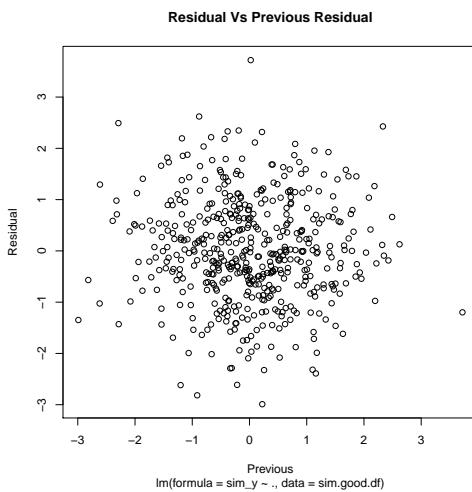
if assumption 1. and 2. are satisfied?

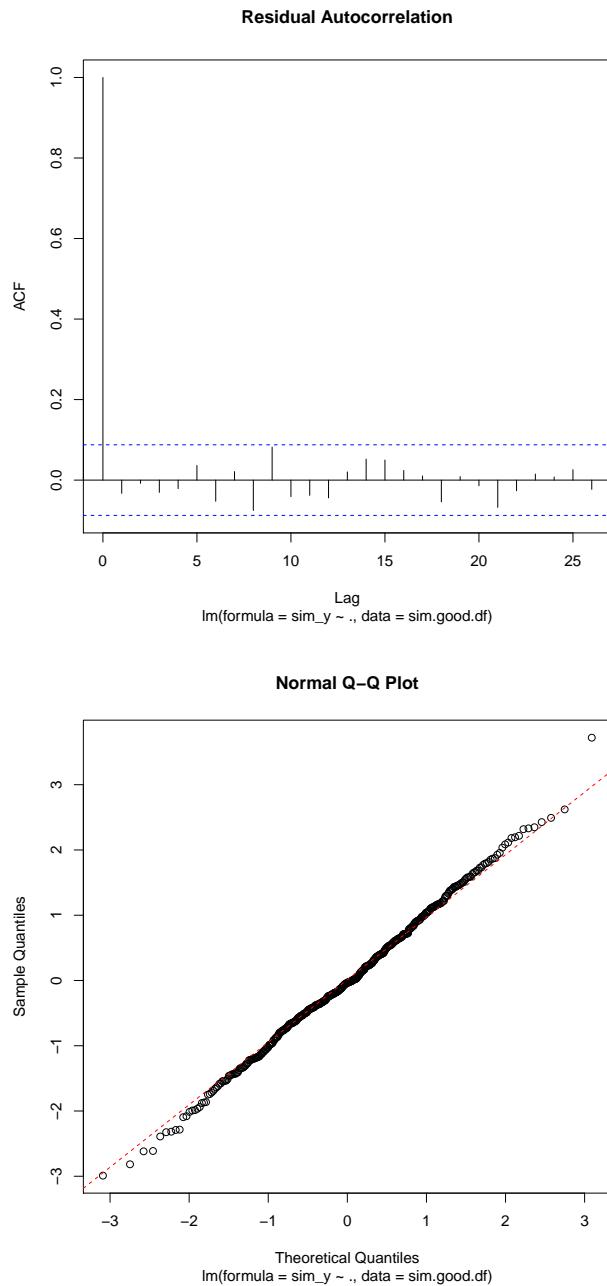




Q: How about assumption 3. and 4.?

- It is clear that we can reply on the same diagnostic tools to for 3. and 4.





### Transistor Study

- A study on transistor gain between emitter and collector in an integrated circuit device was done in early days, the following variables are collected.

**dit**      Drive-in time in minutes  
**dose**     ions  $\times 10^{14}$   
**gain**    measured in hFE

- It is interested to determine whether
  1. drive-in time or dose influences gain linearly
  2. drive-in time influences gain for a given dose linearly
  3. dose influences gain for a given drive-in time linearly
  4. there is any reason to use both drive-in time and dose to explain gain
- After loading the data

```

> # install.packages("xlsx")
> library(xlsx)
> trans.df = read.xlsx(
+   file = "~/Desktop/transistor.xlsx",
+   sheetIndex = 1)
  
```

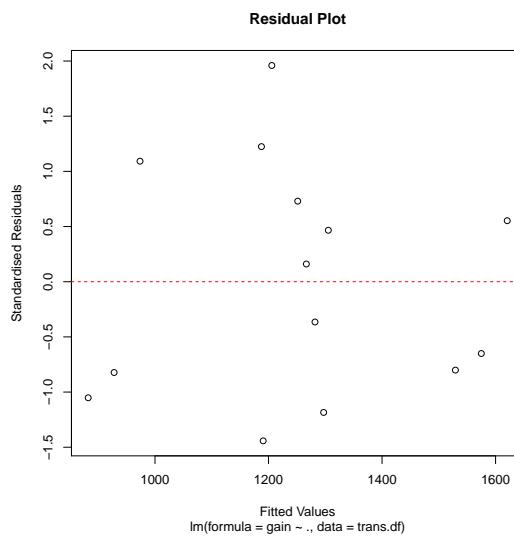
- We consider the full model, that is, including all the regression variables

```

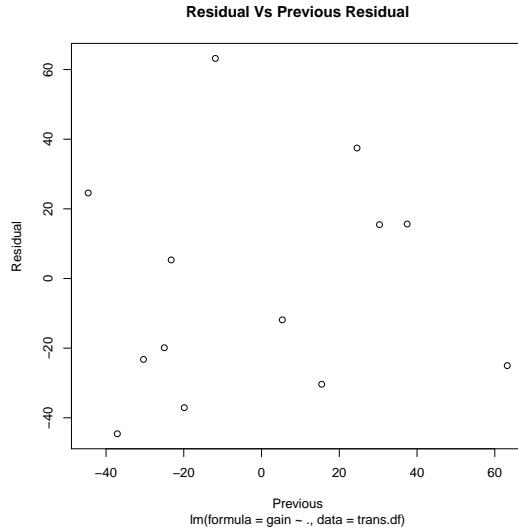
> trans.LM = lm(gain ~ ., data = trans.df)

> model = trans.LM           # Reusable
>
> fvs = fitted.values(model) # fitted values
> res = residuals(model)    # residuals
> sres = rstandard(model)   # standardised
  
```

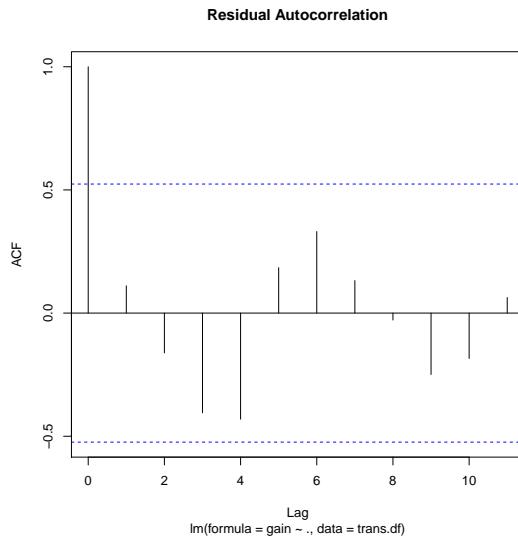
- We must check assumptions before we can use it to answer those questions.
- There is no indication of nonlinearity, correlation or non-constant variance.



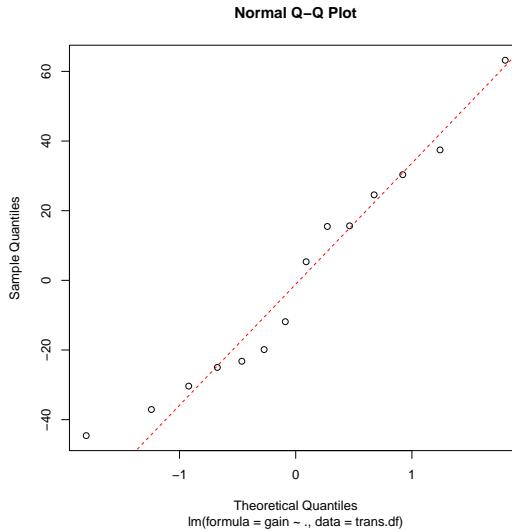
- There is no indication that the residuals are correlated.



- There is no indication that the residuals are correlated.



- There is no clear indication that normality is violated.



- Given there is no indication of any violation of the assumptions, we proceed

```
> summary(model)
```

---

```
Call:
lm(formula = gain ~ ., data = trans.df)

Residuals:
    Min      1Q  Median      3Q     Max 
-44.584 -24.565 -3.266  22.330  63.201 

Coefficients:
            Estimate Std. Error t value Pr(>t)    
(Intercept) -520.0767   192.1071 -2.707  0.02039 *  
dit          10.7812    0.4743  22.730 1.35e-10 *** 
dose        -152.1489   36.6754 -4.149  0.00162 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Residual standard error: 34.93 on 11 degrees of freedom
Multiple R-squared:  0.9798,    Adjusted R-squared:  0.9762 
F-statistic: 267.2 on 2 and 11 DF,  p-value: 4.742e-10
```

---

Q: What is the base for answering question 1.?

Does drive-in time or dose influence gain linearly?

Q: Any evidence against the claim that neither **dit** nor **dose** influences **gain**?

Q: Any one remember how F-test works?

- Recall the underlying hypothesis of F-statistics is that all slopes are zero

$$\beta_1 = \beta_2 = \dots = \beta_k = 0$$

- If the p-value corresponding to a F-statistics is small, then we have evidence against the regression coefficients being **all zero**.

Q: Can you remember how to derive F-test?

- Under the above hypothesis and the model assumptions, F-statistics follows

$$F(d_1 = k, d_2 = n - k - 1)$$

where  $k$  is the number of predictors, and  $n$  is the number of observations.

- We can work out the F-statistics manually by using R as calculator

```
> n = nrow(trans.df)
> n
```

```
[1] 14
```

```
> y.bar = mean(trans.df$gain)

> tmp = trans.df$gain - y.bar

> yss = sum(tmp^2)

> rse = sigma(trans.LM)^2

> f0 = (yss - (n - 2 - 1) * rse) / (2 * rse)
>
> f0
```

```
[1] 267.177
```

```
> 1 - pf(f0, 2, n - 2 - 1)
```

```
[1] 4.741632e-10
```

- It verifies the F-statistics and the extremely small p-value in the summary,

```
> summary(model)
```

```
Call:
lm(formula = gain ~ ., data = trans.df)

Residuals:
    Min      1Q  Median      3Q     Max 
-44.584 -24.565 -3.266  22.330  63.201 

Coefficients:
            Estimate Std. Error t value Pr(>t)
(Intercept) -520.0767   192.1071 -2.707  0.02039 *
dit          10.7812    0.4743  22.730 1.35e-10 ***
dose        -152.1489   36.6754 -4.149  0.00162 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 34.93 on 11 degrees of freedom
Multiple R-squared:  0.9798,    Adjusted R-squared:  0.9762 
F-statistic: 267.2 on 2 and 11 DF,  p-value: 4.742e-10
```

- Hence we have extremely strong evidence against the claim that neither `dit` nor `dose` influences `gain` in this case, therefore, we favour the conclusion either drive-in time or dose, or both influence gain linearly.

Q: What is the base for answering question 2. and 3.?

Does drive-in time influence gain for a given dose linearly?  
Does dose influence gain for a given drive-in time linearly?

- Recall we have derived the following

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

from which t-test can be derived to be used as the bases.

- Individually, for the  $j$ th predictor, the t-test has the hypothesis that

$$\beta_j = 0$$

- If the p-value corresponding to a t-statistics is small, then we have evidence against the corresponding predictor not influencing the response linearly.

```
> summary(model)
```

```
Call:
lm(formula = gain ~ ., data = trans.df)

Residuals:
    Min      1Q  Median      3Q     Max 
-44.584 -24.565 -3.266  22.330  63.201 

Coefficients:
            Estimate Std. Error t value Pr(>t)    
(Intercept) -520.0767   192.1071 -2.707  0.02039 *  
dit          10.7812    0.4743  22.730 1.35e-10 *** 
dose        -152.1489   36.6754 -4.149  0.00162 ** 
---
Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Residual standard error: 34.93 on 11 degrees of freedom
Multiple R-squared:  0.9798,    Adjusted R-squared:  0.9762 
F-statistic: 267.2 on 2 and 11 DF,  p-value: 4.742e-10
```

- We have extremely strong evidence against the claim that `dit` does not influence `gain` for a fixed value of `dose`, therefore, we favour the conclusion the drive-in time influence gain linearly while holding the dose constant.
- Similarly, we have very strong evidence against the claim that `dose` does not influence `gain` for a fixed value of `dit`, therefore, we favour the conclusion dose influence gain linearly while holding the drive-in time constant.

Q: What is the base for answering question 4.?

Is there any reason to use both drive-in time and dose to explain gain?

Q: Can we rely on F-test?

Q: How about t-test?

Q: Is there anything else in summary that can be used to address this question?

```
> summary(model)
```

```

Call:
lm(formula = gain ~ ., data = trans.df)

Residuals:
    Min      1Q  Median      3Q     Max 
-44.584 -24.565 -3.266  22.330  63.201 

Coefficients:
            Estimate Std. Error t value Pr(>t)    
(Intercept) -520.0767   192.1071 -2.707 0.02039 *  
dit          10.7812    0.4743  22.730 1.35e-10 *** 
dose        -152.1489   36.6754 -4.149 0.00162 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Residual standard error: 34.93 on 11 degrees of freedom
Multiple R-squared:  0.9798, Adjusted R-squared:  0.9762 
F-statistic: 267.2 on 2 and 11 DF,  p-value: 4.742e-10

```

- Essentially, this question could be answered by comparing the submodel

```
> trans.dose.LM = lm(gain~dose, data = trans.df)
```

and the submodel

```
> trans.dit.LM = lm(gain~dit, data = trans.df)
```

with the full model

```
> trans.LM = lm(gain~., data = trans.df)
```

- The Adjusted R-squared in the summary is a crude to do the comparison.
- Recall R-squared, which is also known as the coefficient of determination, is

$$R^2 = \frac{\text{ESS}}{\text{TSS}} = \frac{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^n (y_i - \bar{y})^2} = 1 - \frac{\sum_{i=1}^n \hat{e}_i^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

which gives the proportional of variability explained by the model.

- The Adjusted R-squared is given by

$$\bar{R}^2 = 1 - (1 - R^2) \frac{n - 1}{n - k - 1}$$

Q: Given R-squared can be understood as

$$R^2 = 1 - \frac{\sum_{i=1}^n \hat{e}_i^2}{\sum_{i=1}^n (y_i - \bar{y})^2} = 1 - \frac{\frac{1}{n} \sum_{i=1}^n \hat{e}_i^2}{\frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2}$$

why adjusted R-squared is simply correcting the biased MLE.

- This alteration allows  $\bar{R}^2$  to take model complexity into account.
- A submodel might have a larger  $\bar{R}^2$ , if so, it is preferred over the full model.

Q: What is the difference between  $R^2$  and the adjusted  $R^2$ ?

- $R^2$  is a measure of goodness of fit when all the assumptions are satisfied.
- However, large  $R^2$  does not mean:
  - the assumptions are satisfied.
  - a better predictive model.
  - a better model across different datasets.
  - a better model when the models have different number of parameters.
- The adjusted  $R^2$  is a relative measure to address 4., and it cannot be
  - interpreted alone.
  - used for two models that have different responses.
- Like F-test and t-test in the summary, both  $R^2$  and  $\bar{R}^2$  are only meaningful if all model assumptions are satisfied.
- However, it can be shown there is evidence against the two submodels for `trans.df` satisfying the models assumptions, thus we prefer the full model.

### 3.3 Polynomial

- Under multiple linear regression,
- 1. The conditional mean of the response is given by

$$\begin{aligned}\mathbb{E}[Y_i | X_{i1}, X_{i2}, \dots, X_{ik}] &= \mathbb{E}[Y_i | \mathbf{X}_i] \\ &= \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_k x_{ik}\end{aligned}$$

which means there is no curvature in conditional mean.

- For a general region, there is no reason to expect it is a flat hyperplane.
- Using polynomials of  $x_{*j}$  is the easiest way to model nonlinearity, we assume

$$\mathbb{E}[Y_i | \mathbf{X}_i] = \beta_0 + P_1^{d_1} + P_2^{d_2} + \dots + P_k^{d_k}$$

where  $P_j^{d_j} = \sum_{\ell=1}^{d_j} \beta_{j\ell} x_{ij}^\ell$  is a  $d_j$ th degree polynomial of the predictor  $x_{*j}$ .

- This is often known as **polynomial regression**, which is linear in parameters.
- In terms of estimation and inference, we only modify the matrix  $\mathbf{X}_{n \times (k+1)}$  to

$$\mathbf{X}_{n \times (m+1)}^* = \begin{bmatrix} 1 & x_{11} & x_{11}^2 & \cdots & x_{1k}^{d_k} \\ 1 & x_{21} & x_{21}^2 & \cdots & x_{2k}^{d_k} \\ \vdots & \vdots & \ddots & & \vdots \\ 1 & x_{n1} & x_{n1}^2 & \cdots & x_{nk}^{d_k} \end{bmatrix}, \quad \text{where } m = \sum_{j=1}^k d_j,$$

the rest remains the same

$$\mathbf{y} = \mathbf{X}^* \boldsymbol{\beta} + \mathbf{e}$$

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_m \end{bmatrix}, \text{ and } \mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix}$$

- The following illustrates when and how to used polynomial regression,

```
> sim.df = read.csv(
+   file = "~/Desktop/sim_poly_class.csv.bz2")

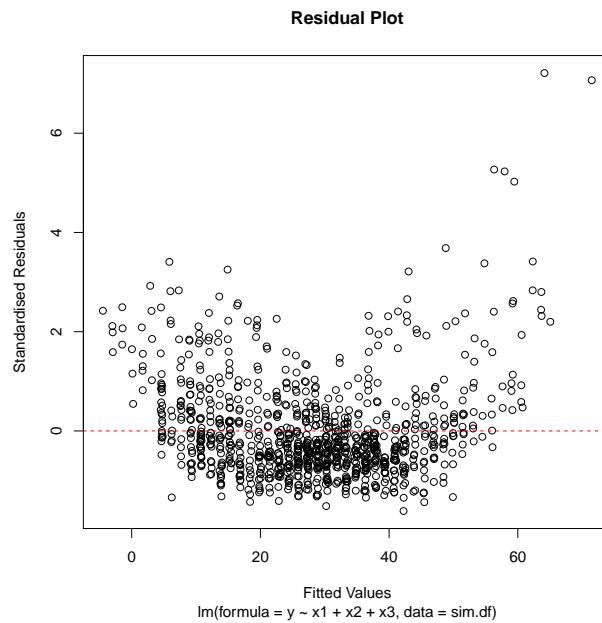
> str(sim.df)
```

```
'data.frame': 1000 obs. of 4 variables:
$ y : num 4.31 16.09 10.67 3.24 33.5 ...
$ x1: int 1 1 1 1 1 1 1 1 1 ...
$ x2: int 3 6 5 3 9 9 3 7 5 5 ...
$ x3: num -0.3706 -1.5564 -0.5157 -1.5024 -0.0638
```

- Since there are only 3 regressors, we construct the full multiple linear model

```
> sim.LM = lm(y ~ x1 + x2 + x3, data = sim.df)
```

- The residual plot indicates nonlinearity or error is correlated with fitted value.

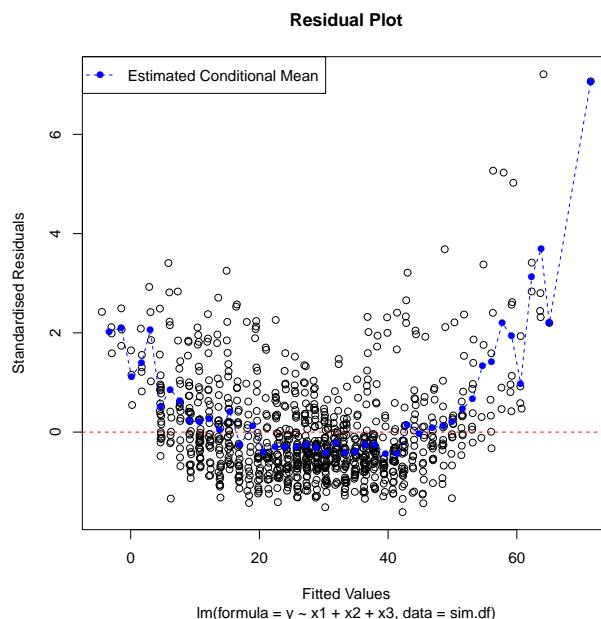


- The nonlinearity is clearer if we add estimated conditional means to the plot

```

> c.mean.plot =
+   function(tmp.x, tmp.y, n.each, pos = "topleft"){
+
+   n.x = round(length(unique(tmp.x)) / n.each)
+   x.group = cut(tmp.x, breaks = n.x, labels = FALSE)
+
+   x.vec = tapply(tmp.x, x.group, mean)
+   y.vec = tapply(tmp.y, x.group, mean)
+
+   points(x.vec, y.vec, col = "blue", pch = 16)
+   lines(x.vec, y.vec, col = "blue", lty = 2)
+
+   legend(pos, "Estimated Conditional Mean",
+         col = "blue", lty = 2, pch = 16)
+
+ }

```



- The residual plot only indicates a potential problem,

$$\hat{e}'_i \quad \text{Vs} \quad \hat{y}_i$$

it provides no information regarding the exact nature of the problem.

Q: Suppose the conditional mean is given by or can be approximated by

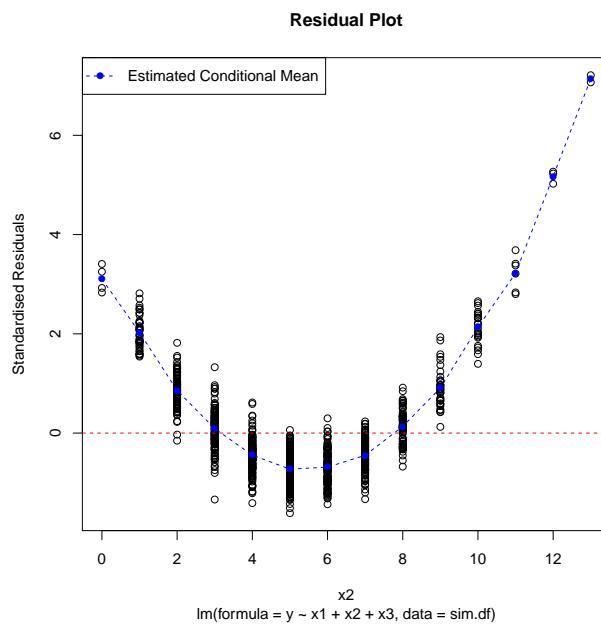
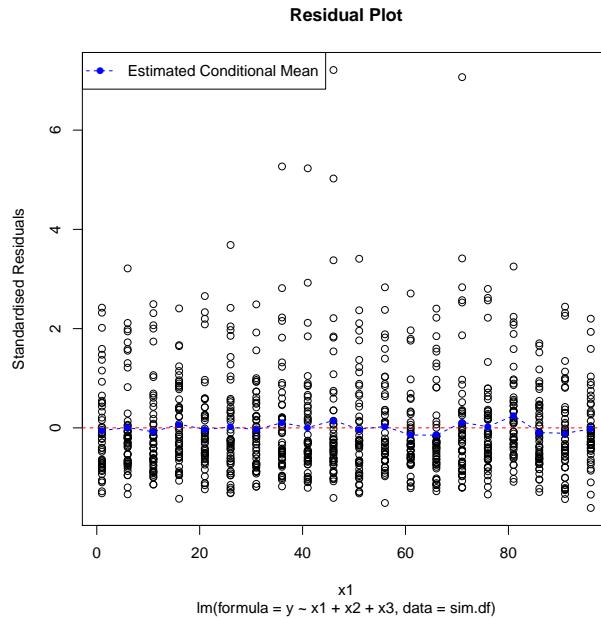
$$\mathbb{E}[Y_i | \mathbf{X}_i] = \beta_0 + P_1^{d_1} + P_2^{d_2} + \cdots + P_k^{d_k}$$

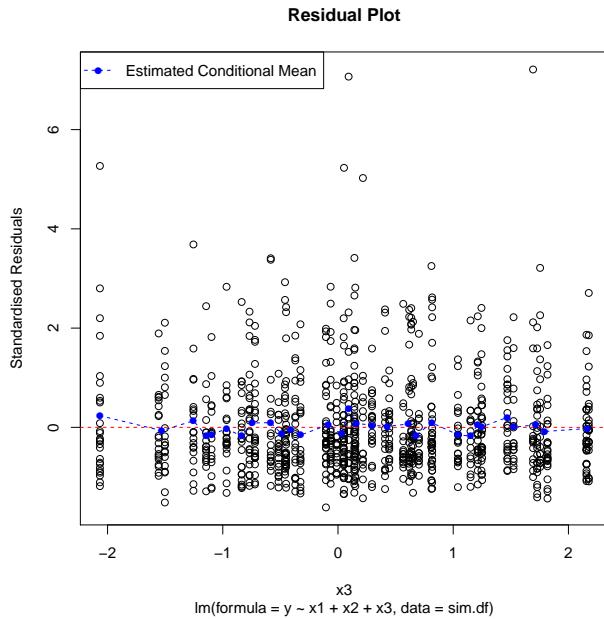
what do you expect to see in the residual plot of

$$\hat{e}'_i \quad \text{Vs} \quad x_{ij}$$

that is, the standardised residual against one of the regressors?

- This is the simplest way to determine whether a polynomial term is needed.
- So we usually produce residual plots of both kinds as a standard practice.



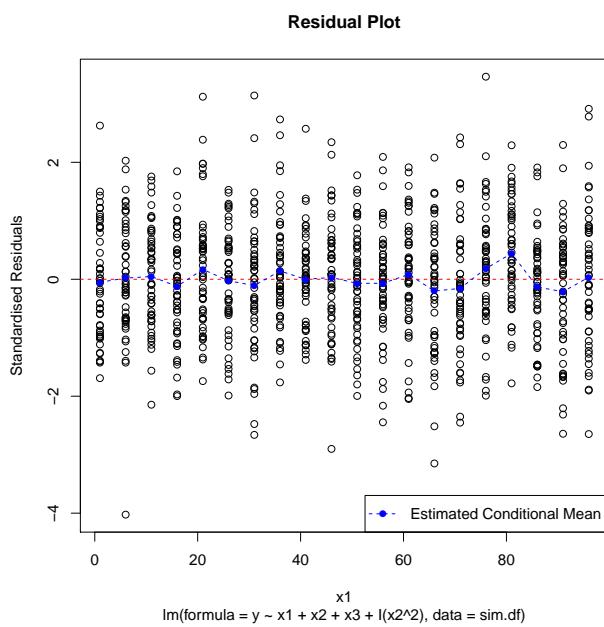
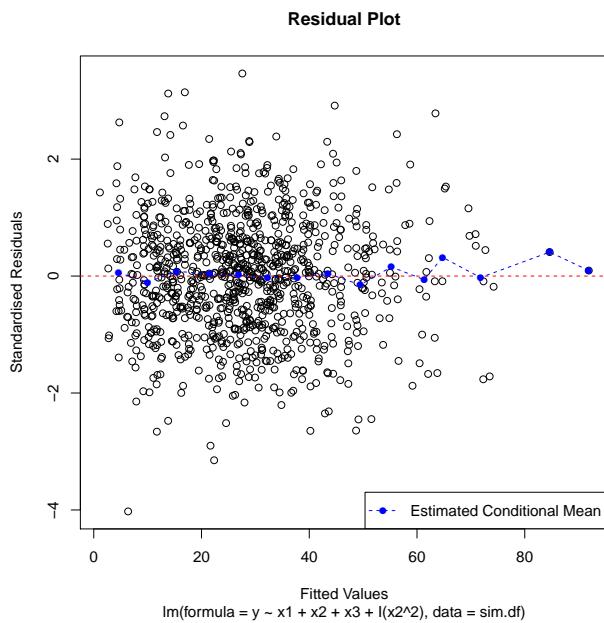


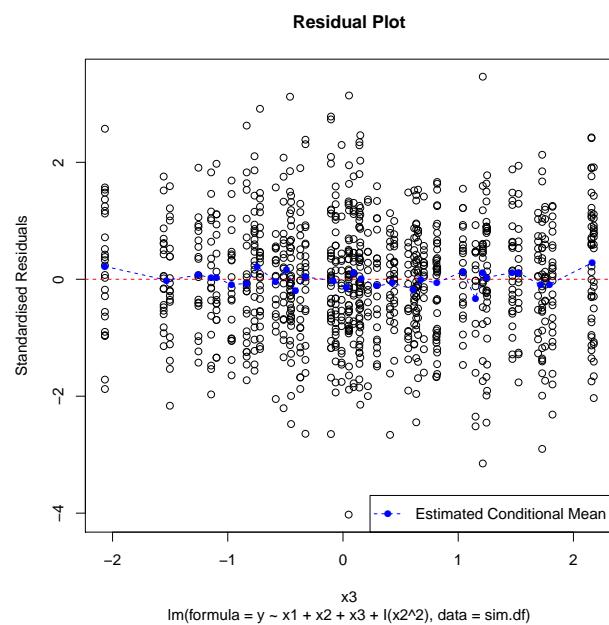
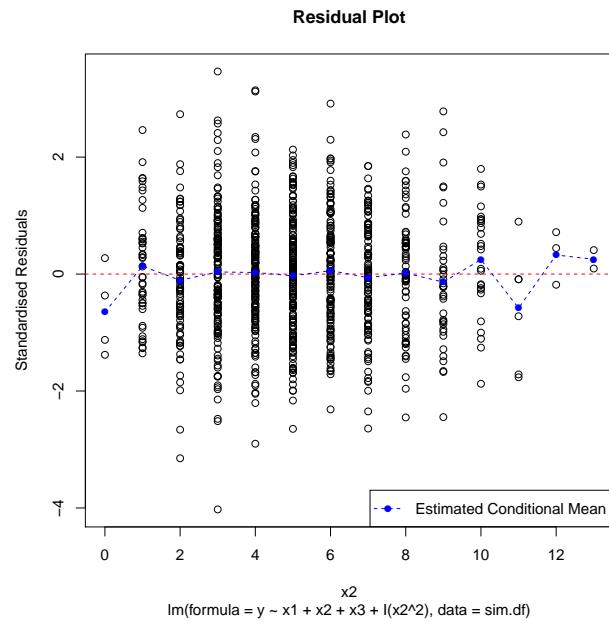
Q: What do those residual plots show us?

- It seems there is nothing left in the residual that can be explained by  $x_1$  and  $x_3$ .
- However, the residual is clearly not independent of  $x_2$ .
- One of many possibilities for seeing such residual plots is

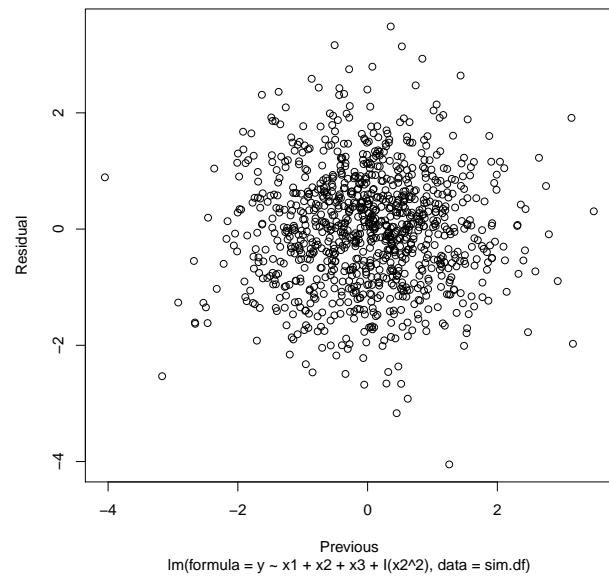
$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \underbrace{\beta_4 X_2^2}_{\varepsilon} + \varepsilon^*$$

- Based on those residual plots, we consider the following model next
- ```
> sim.quad.LM = lm(y ~ x1 + x2 + x3 + I(x2^2), data = sim.df)
```
- which seems to satisfy all the model assumptions.

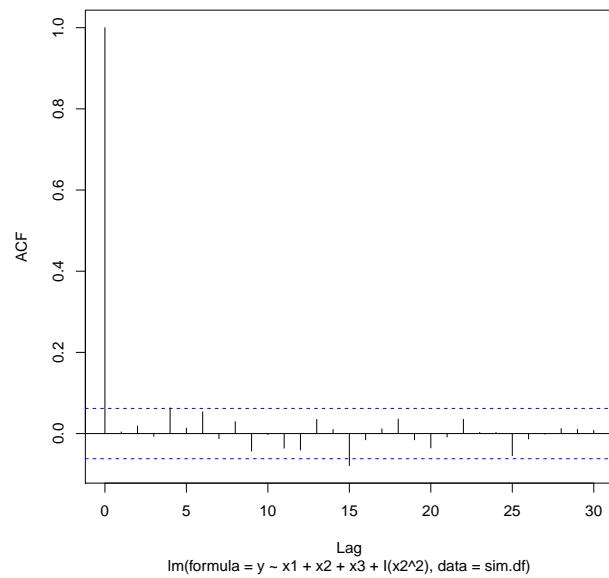


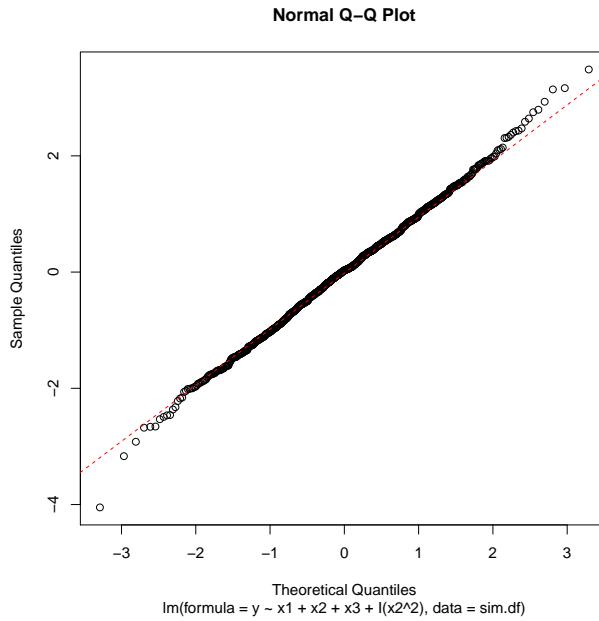


**Residual Vs Previous Residual**



**Residual Autocorrelation**





- Since we have no evidence against all assumptions being satisfied, we trust

```
> summary(sim.quad.LM)
```

```
Call:
lm(formula = y ~ x1 + x2 + x3 + I(x2^2), data = sim.df)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.0491 -0.6691  0.0280  0.6326  3.4861 

Coefficients:
            Estimate Std. Error t value Pr(>t)
(Intercept) 0.277649  0.146033  1.901   0.0576 .
x1          0.300215  0.001106 271.321 < 2e-16 ***
x2          0.238286  0.052575  4.532 6.54e-06 ***
x3          0.155625  0.030654  5.077 4.58e-07 ***
I(x2^2)     0.397486  0.004636 85.741 < 2e-16 ***

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.008 on 995 degrees of freedom
Multiple R-squared:  0.9948,    Adjusted R-squared:  0.9947 
F-statistic: 4.73e+04 on 4 and 995 DF,  p-value: < 2.2e-16
```

Q: What can you learn from comparing this with the output of

```
> summary(sim.LM)
```

```
> summary(sim.LM)
```

```
Call:
lm(formula = y ~ x1 + x2 + x3, data = sim.df)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.6987 -1.8885 -0.7724  1.0437 20.8883 

Coefficients:
            Estimate Std. Error t value Pr(>t)
```

```

(Intercept) -9.322246   0.271399 -34.349   <2e-16 ***
x1          0.298398   0.003203  93.176   <2e-16 ***
x2          4.585967   0.040203 114.071   <2e-16 ***
x3          0.066189   0.088685   0.746    0.456
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

Residual standard error: 2.918 on 996 degrees of freedom
Multiple R-squared:  0.9561,   Adjusted R-squared:  0.9556
F-statistic: 7233 on 3 and 996 DF,  p-value: < 2.2e-16

```

- We could wrongly drop **x3** if we didn't do residual analysis first!
- Transformations can be used in as before to alleviate non-constant variance and non-normality in multiple linear regression or polynomial models.
- Consider the following dataset,

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <b>Temperature</b> | temperature standard deviation during production |
| <b>Density</b>     | density of the final product                     |
| <b>Rate</b>        | rate of production                               |
| <b>Defective</b>   | average number of defects per 1000 produced      |

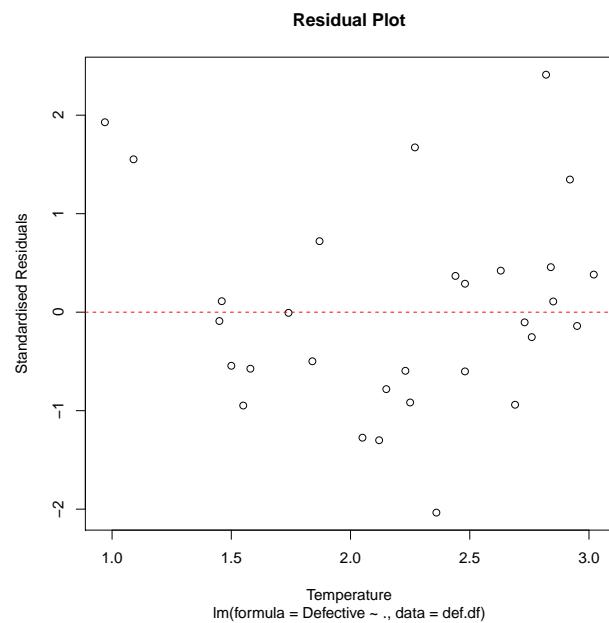
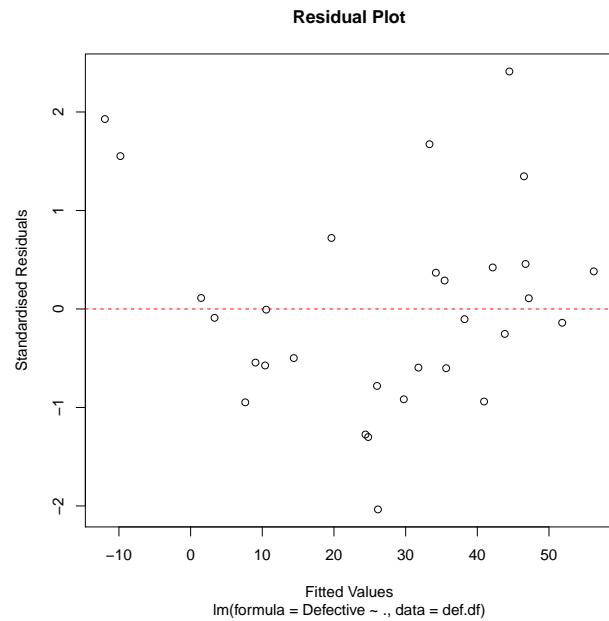
which is gathered for quality control in production.

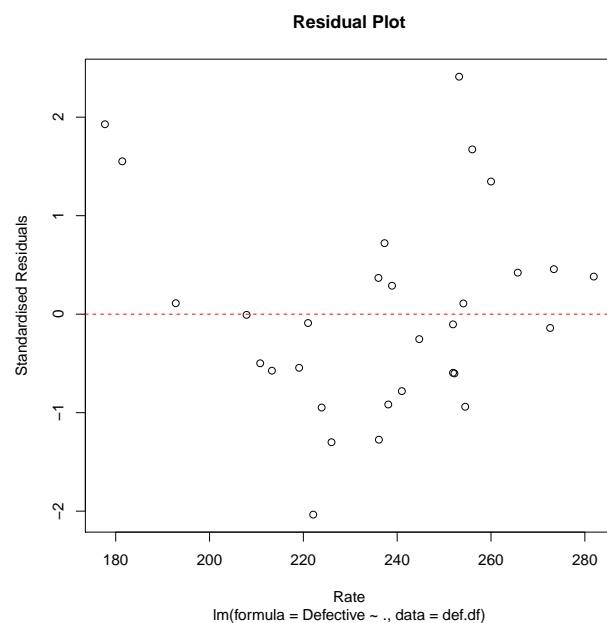
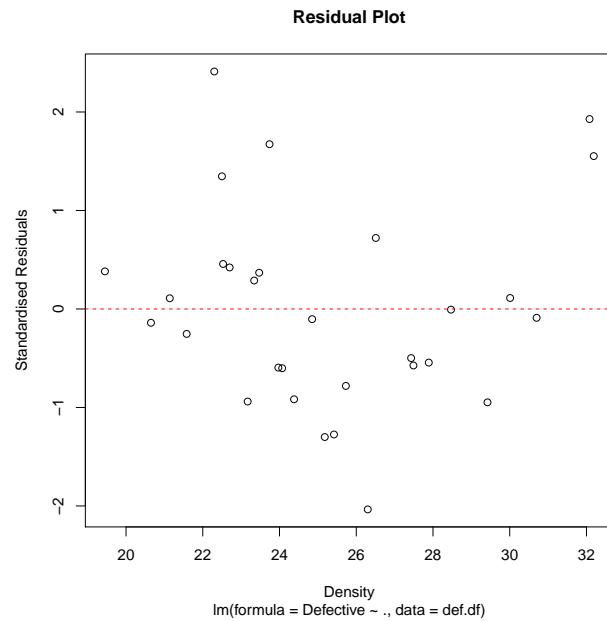
```
> str(def.df)
```

```
'data.frame': 30 obs. of 4 variables:
$ Temperature: num 0.97 2.85 2.95 2.84 1.84 2.05 1.5 2.48 2.23 3.02 ...
$ Density    : num 32.1 21.1 20.6 22.5 27.4 ...
$ Rate       : num 178 254 273 273 211 ...
$ Defective  : num 0.2 47.9 50.9 49.7 11 15.6 5.5 37.4 27.8 58.7 ...
```

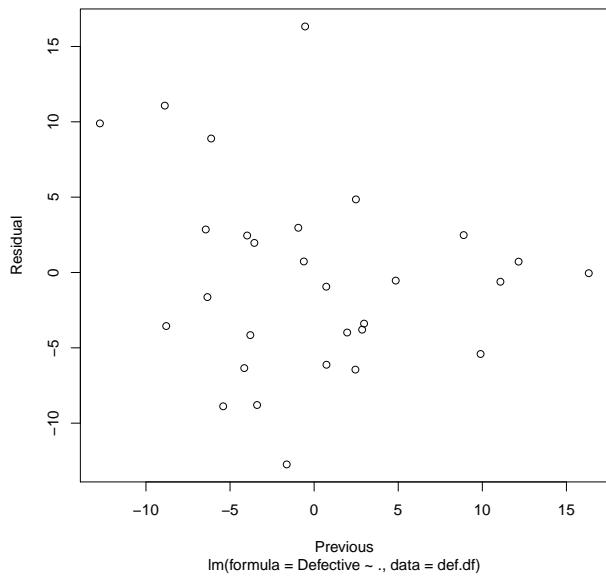
- We start with the multiple linear regression model with all the variables

```
> def.LM = lm(Defective~., data = def.df)
```

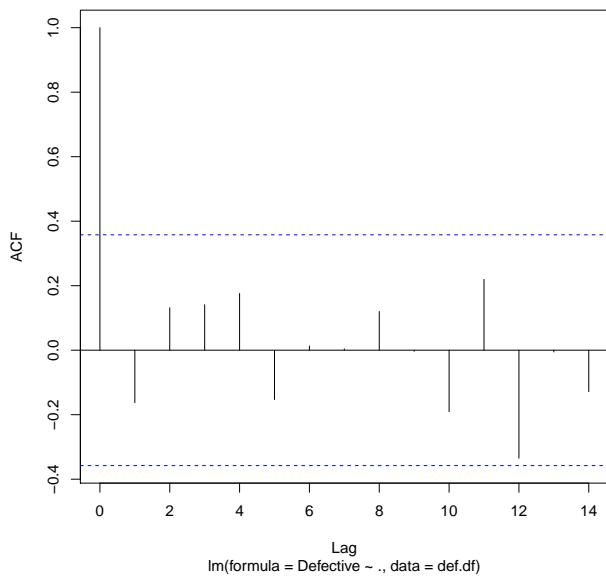


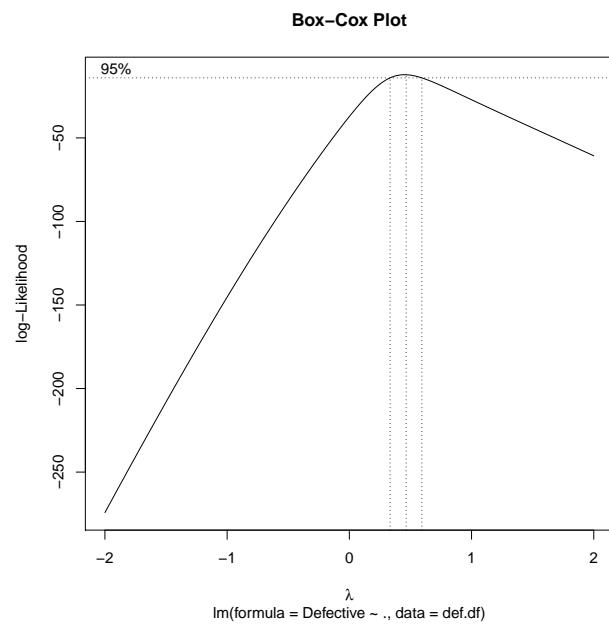
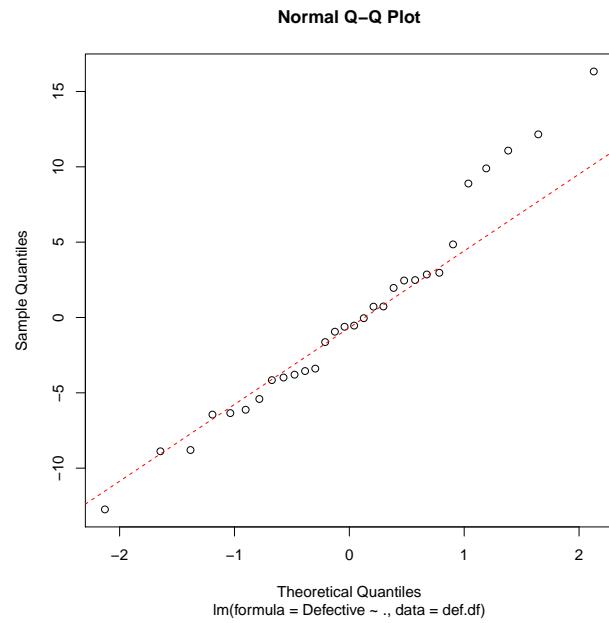


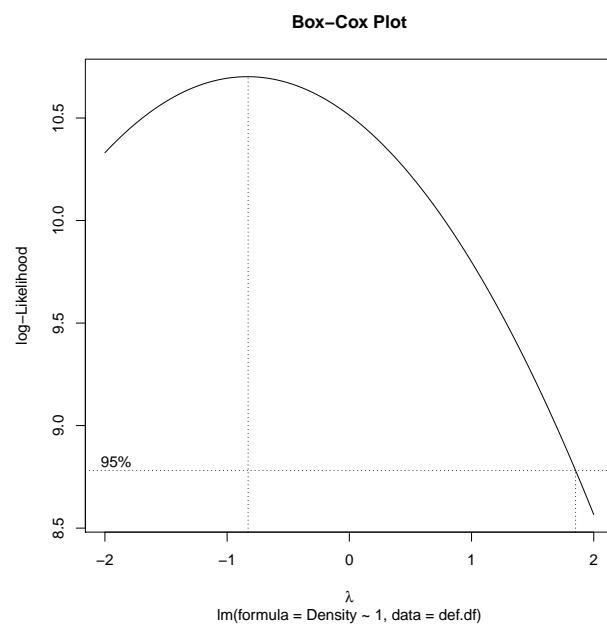
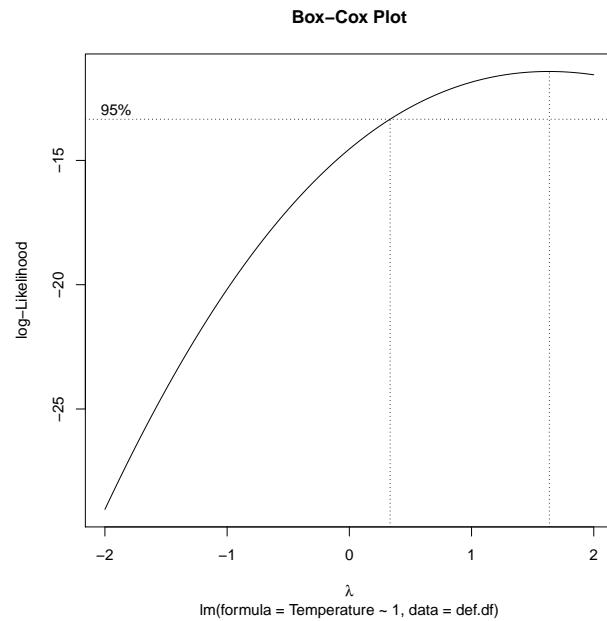
**Residual Vs Previous Residual**

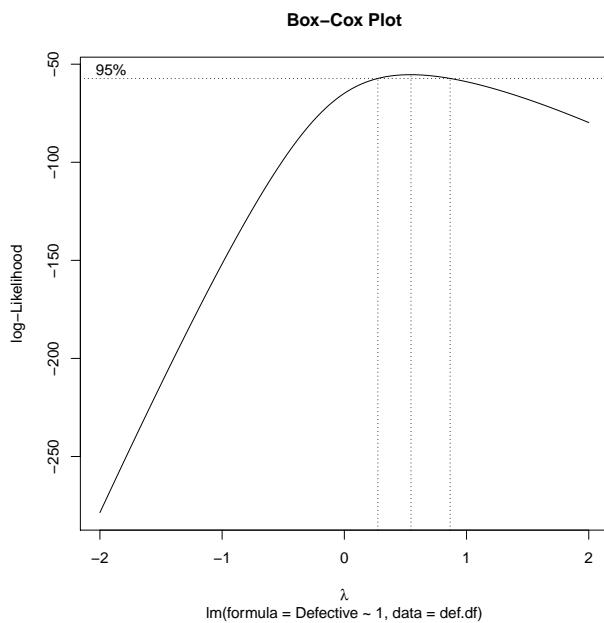
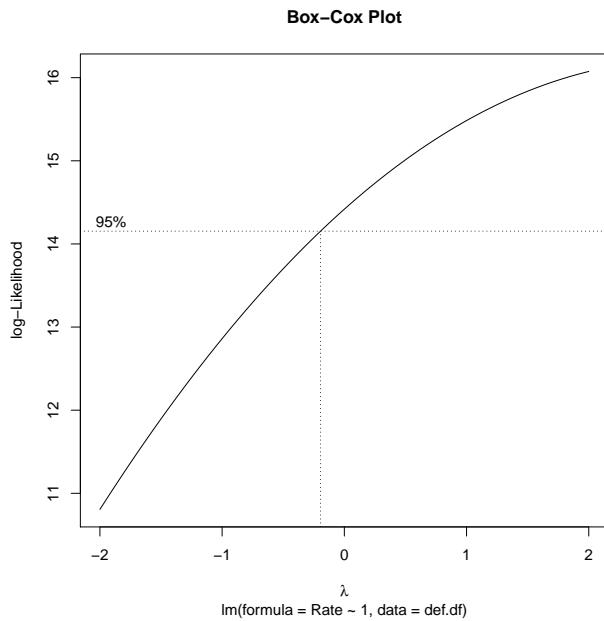


**Residual Autocorrelation**









Q: We don't have any serious issue, but what do those Box-Cox plots say?

- Therefore, we consider the following model instead

```
> def.bc.y.LM=lm(2*sqrt(Defective)-2~, data=def.df)
```

- Running the residual analysis again reveals no evidence of any violation, thus

```
> summary(def.bc.y.LM)
```

```
Call:
lm(formula = 2 * sqrt(Defective) ~ 2 ~ ., data = def.df)

Residuals:
    Min      1Q  Median      3Q     Max 
-2.2029 -0.5700 -0.1543  0.6828  2.2790 

Coefficients:
            Estimate Std. Error t value Pr(>t)
(Intercept) 9.18594   10.52802   0.873   0.3909
Temperature  3.13033    1.32451   2.363   0.0259 *
Density     -0.58333    0.23907  -2.440   0.0218 *
Rate        0.02580    0.02086   1.237   0.2273 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.135 on 26 degrees of freedom
Multiple R-squared:  0.943,    Adjusted R-squared:  0.9365 
F-statistic: 143.5 on 3 and 26 DF,  p-value: 2.713e-16
```

```
> summary(def.bc.y.no.rate.LM)
```

```
Call:
lm(formula = 2 * sqrt(Defective) ~ 2 ~ . ~ Rate, data = def.df)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.96888 -0.86248 -0.02937  0.57578  2.58125 

Coefficients:
            Estimate Std. Error t value Pr(>t)
(Intercept) 17.1226    8.4271   2.032   0.0521 .
Temperature  3.5661    1.2892   2.766   0.0101 *
Density     -0.6939    0.2239  -3.099   0.0045 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.147 on 27 degrees of freedom
Multiple R-squared:  0.9397,    Adjusted R-squared:  0.9352 
F-statistic: 210.3 on 2 and 27 DF,  p-value: < 2.2e-16
```

Q: Given the above model also satisfies all assumptions, should we retain **Rate**?

- Suppose there is actually one more variable collected by the study

**am\_pm** Morning is indicated by 1, and Afternoon by 0.

Q: How can we interpret each of the following parameters?

$$\sqrt{\text{Defective}} = \beta_0 + \beta_1 \text{Temperature} + \beta_2 \text{Density} + \beta_3 \text{Rate} + \beta_4 \text{am_pm} + \varepsilon$$

Q: How can we interpret  $\beta_5^*$ ?

$$\begin{aligned} \sqrt{\text{Defective}} = & \beta_0^* + \beta_1^* \text{Temperature} + \beta_2^* \text{Density} + \beta_3^* \text{Rate} \\ & + \beta_4^* \text{am_pm} + \beta_5^* \text{am_pm} * \text{Temperature} + \varepsilon \end{aligned}$$

- Suppose there is another categorical variable collected by the study

**location** CHN, UK and USA

- Let us just investigate the two categorical variables,

```
> def.interaction.LM =
+ lm(Defective ~ am_pm * location, data = def.df)
```

Q: What do you think the following coefficients represent?

```
> summary(def.interaction.LM)
```

| Call:                                                     |         |        |        |            |  |
|-----------------------------------------------------------|---------|--------|--------|------------|--|
| lm(formula = Defective ~ am_pm * location, data = def.df) |         |        |        |            |  |
| Coefficients:                                             |         |        |        |            |  |
| (Intercept)                                               | 33.750  | 9.892  | 3.412  | 0.00229 ** |  |
| am_pmPM                                                   | -5.467  | 12.771 | -0.428 | 0.67243    |  |
| locationUK                                                | -6.710  | 13.272 | -0.506 | 0.61777    |  |
| locationUSA                                               | -14.517 | 12.771 | -1.137 | 0.26689    |  |
| am_pmPM:locationUK                                        | -1.973  | 17.879 | -0.110 | 0.91303    |  |
| am_pmPM:locationUSA                                       | 26.483  | 18.061 | 1.466  | 0.15554    |  |

Q: Can you construct a two-way table for the 6 categories from the output?

- The `interaction` is the difference between column/row differences

$$(\mu_{ij} - \mu_{1j}) - (\mu_{i1} - \mu_{11}) = \mu_{ij} - \mu_{i1} - \mu_{1j} + \mu_{11}$$

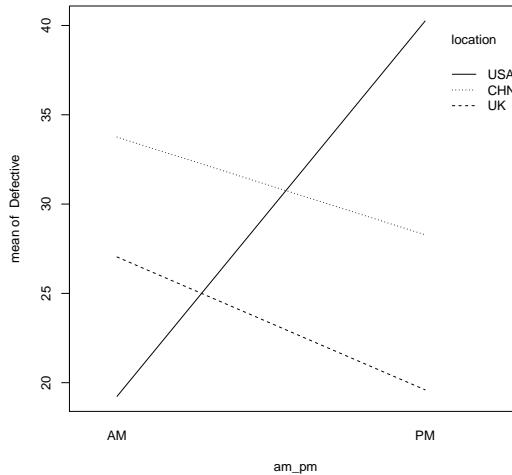
Q: Can you see the R summary output naturally decomposes the mean  $\mu_{ij}$ ?

|    |                     |                     |                     |
|----|---------------------|---------------------|---------------------|
|    | CHN                 | UK                  | USA                 |
| AM | $\mu_{11} = 33.750$ | $\mu_{12} = 27.040$ | $\mu_{13} = 19.233$ |
| FM | $\mu_{21} = 28.283$ | $\mu_{22} = 19.600$ | $\mu_{23} = 40.249$ |

The conditional mean can be decomposed into the following

$$\mu_{ij} = \mu_{11} + \underbrace{(\mu_{i1} - \mu_{11})}_{\text{row mean effect}} + \underbrace{(\mu_{1j} - \mu_{11})}_{\text{col mean effect}} + \underbrace{(\mu_{ij} - \mu_{i1} - \mu_{1j} + \mu_{11})}_{\text{interaction}}$$

```
> # A graphical representation of the two-way table
> with(def.df, interaction.plot(
+   am_pm, location, Defective))
```



- Of course, the full model now will include all 5 regressors

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <b>Temperature</b> | temperature standard deviation during production |
| <b>Density</b>     | density of the final product                     |
| <b>Rate</b>        | rate of production                               |
| <b>Defective</b>   | average number of defects per 1000 produced      |
| <b>am_pm</b>       | Morning is indicated by 1, and Afternoon by 0.   |
| <b>location</b>    | CHN, UK and USA                                  |

- In R, this can be specified as the following

```
> def.full.LM = lm(Defective~am_pm*location*Rate
+                   +am_pm*location*Density
+                   +am_pm*location*Temperature ,
+                   data = def.df)
```

Q: What does the above model correspond to?

This model is the model that has different slopes and different intercepts for each one of the 6 categories.

### 3.4 Multicollinearity

- Let us go back to simple linear regression for a moment,

$$y_i = \beta_0 + \beta_1 x_i + e_i$$

- Recall the standard error of  $\hat{\beta}_1$  is given by

$$\begin{aligned} \text{Var} [\hat{\beta}_1 | x_1, x_2, \dots, x_n] &= \frac{\sigma^2}{(n-1)s_x^2} \\ \implies \text{SE} (\hat{\beta}_1) &= \frac{\sigma}{\sqrt{(n-1)s_x^2}} = \frac{\sigma}{\left( \sum_{i=1}^n (x_i - \bar{x})^2 \right)^{1/2}} \end{aligned}$$

from which we see the “accuracy” of our model is partly determined by the amount of scatter about the true regression line, measured by  $\sigma$ , but also by

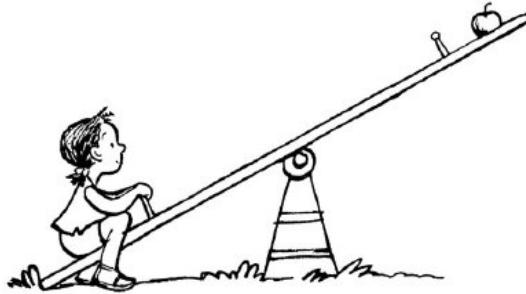
“configuration”

of observed  $x_i$ , that is, the spread of the observed  $x_i$ .

- If the  $x_i$  values are spread out, the estimated regression line is well supported

and changes a little under resampling.

- On the other hand, if  $x_i$ ’s are bunched up, then it is not well supported,



and the regression line is “unstable” very much like a seesaw.

- Consider the following simulation to see the stability problem

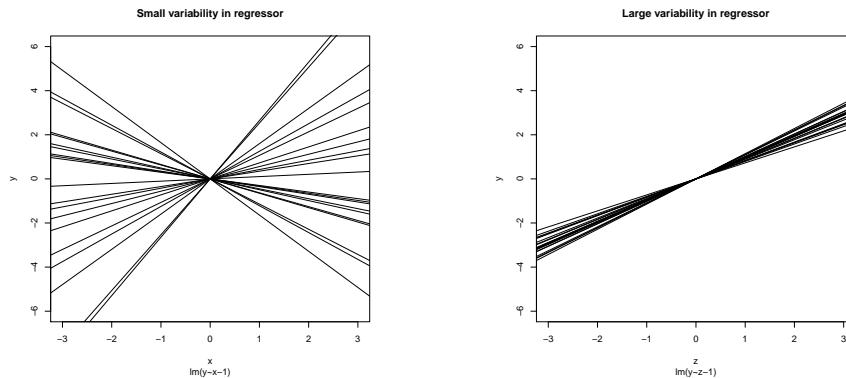
```
> set.seed(1)
> x.vec = rnorm(10, 0, 1)
> z.vec = 10*x.vec

> tmp = seq(-3, 3, length.out = 10)
> plot(tmp, 2*tmp, type = "n")

> replicate(20, {
+   y.vec = rnorm(10, mean = x.vec, sd = 3)
+   abline(lm(y.vec~x.vec-1))
+ })

> plot(tmp, 2*tmp, type = "n")
>
> replicate(20, {
+   y.vec = rnorm(10, mean = z.vec, sd = 3)
+   abline(lm(y.vec~z.vec-1))
+ })
```

- On the left, we have  $\sum_{i=1}^{10} (x_i - \bar{x})^2 = 60.9$ , and 20 simulated sets of  $\{y_i\}$ ,



while  $\sum_{i=1}^{10} (z_i - \bar{z})^2 = 0.609$  on the right.

Q: What happens the stability problem when having two predictors  $X_1$  and  $X_2$ ?

- Intuitively, the spread are still important in the stability of the regression,

$$\sum_{i=1}^n (x_{i1} - \bar{x}_1)^2 \quad \text{and} \quad \sum_{i=1}^n (x_{i2} - \bar{x}_2)^2$$

Q: Can you see why the correlation between  $X_1$  and  $X_2$  also matters?

- If there is a strong linear relationship between  $X_1$  and  $X_2$ , we tends to have "a knife edge" support for the regression plane, which is also not stable intuitively.
- On the other hand, if the predictors are uncorrelated (or orthogonal), then they will be well spread out and support the fitted plane better.
- This intuition can be confirmed by explicitly working out

$$\text{Var} [\hat{\beta}_j | \mathbf{X}] \quad \text{for } j = 1, 2$$

- When having only two predictors  $X_1$  and  $X_2$ ,

$$y = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\epsilon}$$

the variance of the estimator  $\hat{\beta}_j$  can be shown to take the following form

$$\text{Var} [\hat{\beta}_j | \mathbf{X}] = \frac{1}{1 - r^2} \cdot \frac{\sigma^2}{(n - 1)s_{x_j}^2} \quad \text{for } j = 1, 2$$

where  $r$  is the correlation coefficient between the two predictors, and

$$s_{x_j}^2 = \frac{1}{n - 1} \sum_{i=1}^n (x_{ij} - \bar{x}_{ij})^2$$

denotes the sample variance of  $X_j$  as usual.

- The term  $\frac{1}{1 - r^2}$  in the variance can be used to detect the stability problem.

Q: Can you see that this term is a ratio between two related variances?

- In general,

$$y = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_k x_k + \hat{\epsilon}$$

the variance can be shown to take the following form

$$\text{Var} [\hat{\beta}_j | \mathbf{X}] = \frac{1}{1 - R_j^2} \cdot \frac{\sigma^2}{(n - 1)s_{x_j}^2} \quad \text{for } j = 1, 2, \dots, k$$

where  $R_j^2$  is the Multiple R-squared obtained from the regression

$$x_j = \hat{\gamma}_0 + \sum_{\ell \neq j} \hat{\gamma}_{\ell} x_{\ell} + \hat{\nu}$$

- The term  $\frac{1}{1 - R_j^2}$ , which is a generalisation of  $\frac{1}{1 - r^2}$ , is used to detect the stability problem, and is known as the  $j$ th **variance inflation factor** (VIF).
- To fully understand the stability problem, consider the data matrix

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1k} \\ 1 & x_{21} & \cdots & x_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nk} \end{bmatrix} = [\mathbf{1} \quad \mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_k]$$

- **Multicollinearity** occurs when the columns of the data matrix are **almost linearly dependent**

that is, for  $\alpha_i \in \mathbb{R}$  that are not simultaneously zero, the following is true

$$\alpha_0 \mathbf{1} + \alpha_1 \mathbf{x}_1 + \alpha_2 \mathbf{x}_2 + \cdots + \alpha_k \mathbf{x}_k = \mathbf{0}$$

- In the presence of numerical errors, it can occur if
  1. One or more of the predictors has/have very little variation.
  2. One or more of the predictors has/have very large mean.
  3. Two or more of the predictors have a linear relationship.
- We can think of multicollinearity caused by the first two ways as inessential,
  1. One or more of the predictor variables has/have very little variation.
  2. One or more of the predictor variables has/have very large mean.

since we can remove it by standardising the data

$$x_{ij}^c = \frac{x_{ij} - \bar{x}_j}{\left( \frac{1}{n-1} \sum_{i=1}^n (x_{ij} - \bar{x}_j)^2 \right)^{1/2}} \quad \text{or} \quad x_{ij}^c = \frac{x_{ij} - \bar{x}_j}{\left( \sum_{i=1}^n (x_{ij} - \bar{x}_j)^2 \right)^{1/2}}$$

- The “Essential” multicollinearity is what remains after standardisation, and is caused by almost linear relationships between the predictor variables.

$$\alpha_0 \mathbf{1} + \alpha_1 \mathbf{x}_1^c + \alpha_2 \mathbf{x}_2^c + \cdots + \alpha_k \mathbf{x}_k^c = \mathbf{0}$$

Q: How can we detect multicollinearity and identify the source of the problem?

$$\text{Var} [\hat{\beta} | \mathbf{X}] = \sigma^2 (\mathbf{X}^T \mathbf{X})^{-1}$$

- Eigenvalue and eigenvector problem

$$\mathbf{A}_{n \times n} \mathbf{x} = \lambda \mathbf{x} \quad \text{where } \mathbf{x} \neq \mathbf{0}$$

- Recall if  $\mathbf{A}$  possesses  $n$  linearly independent eigenvectors, then

$$\mathbf{A} = \mathbf{P} \mathbf{D} \mathbf{P}^{-1}$$

- It can be shown if  $\mathbf{A}$  is a normal, that is

$$\mathbf{A}^T \mathbf{A} = \mathbf{A} \mathbf{A}^T$$

then the matrix can be decomposed as

$$\mathbf{A} = \mathbf{Q} \mathbf{D} \mathbf{Q}^T \quad \text{where} \quad \mathbf{Q}^T \mathbf{Q} = \mathbf{I} = \mathbf{Q} \mathbf{Q}^T$$

- It is clear that  $\mathbf{X}^T \mathbf{X}$  is normal, thus

$$\mathbf{X}^T \mathbf{X} = \mathbf{Q} \mathbf{D} \mathbf{Q}^T$$

- Now consider the inverse

$$(\mathbf{X}^T \mathbf{X})^{-1} = (\mathbf{Q} \mathbf{D} \mathbf{Q}^T)^{-1} = \mathbf{Q} \mathbf{D}^{-1} \mathbf{Q}^T =$$

- Therefore, the  $j$ th diagonal element of  $(\mathbf{X}^T \mathbf{X})^{-1}$  is  $\sum_{\ell=1}^{k+1} q_{j\ell}^2 / \lambda_\ell$  and

$$\text{Var} [\hat{\beta}_j | \mathbf{X}] = \sigma^2 \sum_{\ell=1}^{k+1} q_{j\ell}^2 / \lambda_\ell \quad \text{where} \quad \sum_{\ell=1}^{k+1} q_{j\ell}^2 = 1$$

which means it can only be usually big if one of  $\lambda_\ell$  is usually small.

- To identify which the source of the problem, consider

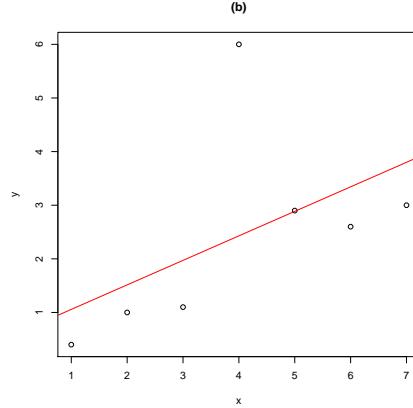
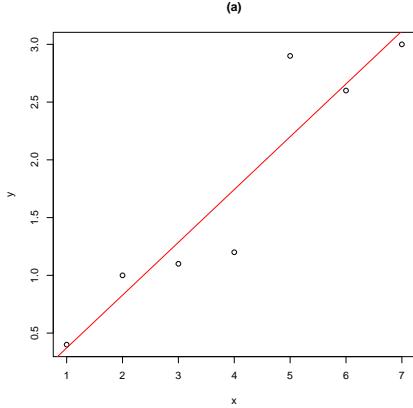
$$\begin{aligned} \mathbf{X}^T \mathbf{X} \mathbf{q}_\ell &= \lambda_\ell \mathbf{q}_\ell \\ \mathbf{q}_\ell^T \mathbf{X}^T \mathbf{X} \mathbf{q}_\ell &= \mathbf{q}_\ell^T \lambda_\ell \mathbf{q}_\ell \\ (\mathbf{X} \mathbf{q}_\ell)^T \mathbf{X} \mathbf{q}_\ell &= \lambda_\ell \mathbf{q}_\ell^T \mathbf{q}_\ell \\ \|\mathbf{X} \mathbf{q}_\ell\|^2 &= \lambda_\ell \approx 0 \end{aligned}$$

which means the eigenvector corresponding to a very small eigenvalue tell us which variables are almost linearly dependent.

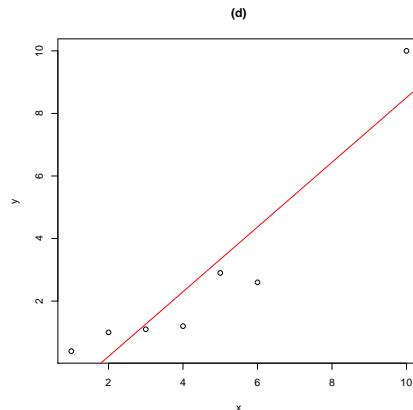
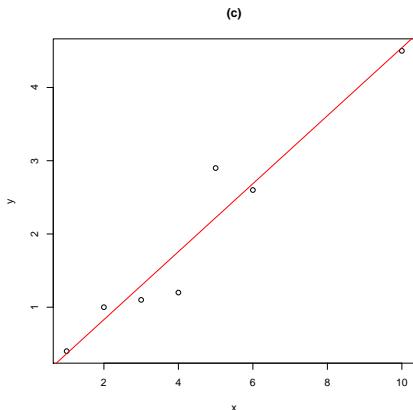
### 3.5 Unusual points

- **Outliers** are points with extreme response values  $y_i | x_i$ , so possibly large  $\hat{e}_i$ .
- **High leverage points** are points with extreme  $x_{ij}$ -values relative to others.

Q: Do we have any outlier or high leverage point in the following?



- Notice the difference between the following two cases.

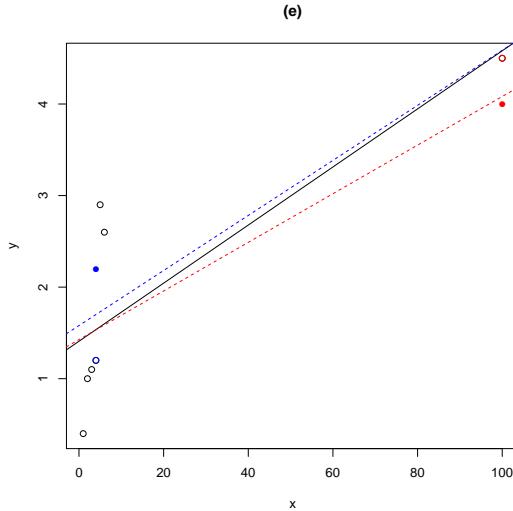


Thus no large residuals.

- High leverage points do not necessarily have large residuals, so that it is occasionally difficult to recognise them from a residual plot.

- Outliers need to be discussed because outliers can distort the regression.

Q: Why do we care about high leverage points? Consider the following



- Having different response values for high leverage points

$$y_i$$

will alter the regression surface more in comparison to low leverage points.

- This observation leads to a common detection and quantification method.
- Notice the fitted values are always on the regression surface,

$$\hat{y}_i$$

so if they change values, regression surface will change.

- Recall we derived the following when we started multiple regression

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{P}\mathbf{y}$$

where  $\mathbf{P}$  is known as the project or hat matrix.

- The leverage score is defined as the partial derivative

$$\frac{\partial \hat{y}_i}{\partial y_i} = [\mathbf{P}]_{ii} = p_{ii} \quad \text{since} \quad \hat{y}_i = \sum_{j=1}^n p_{ij} y_j$$

- The leverage score is entirely depended on the design matrix  $\mathbf{X}$ , not on  $\mathbf{y}$ .
- For the model having one predictor  $x_i$  and no intercept, it can be shown

$$\frac{\partial \hat{y}_i}{\partial y_i} = p_{ii} = \frac{x_i^2}{\sum_{j=1}^n x_j^2}$$

- When we add back the intercept, we have

$$\frac{\partial \hat{y}_i}{\partial y_i} = p_{ii} = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum_{j=1}^n (x_j - \bar{x})^2}$$

- Recall residuals have the following variance formula

$$\text{Var} [\hat{e}_i | \mathbf{X}] = (1 - p_{ii})\sigma^2$$

based on which we use the following

$$\hat{e}'_i = \frac{\hat{e}_i}{\hat{\sigma}\sqrt{1 - p_{ii}}}$$

is known as the **Internally studentised residual/standardised residual**.

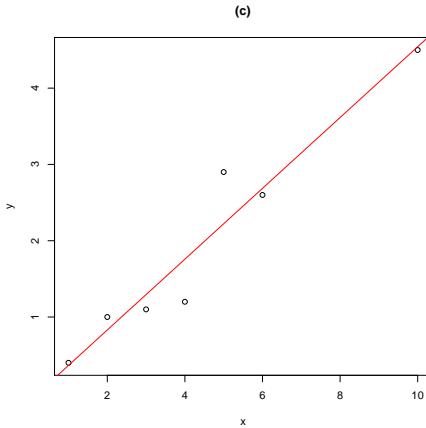
- The implication is that high leverage points tend to have smaller variances.
- **Externally studentised residuals/studentised residual** is defined as

$$\hat{e}^*_i = \frac{\hat{e}_i}{\hat{\sigma}(-i)\sqrt{1 - p_{ii}}}$$

which offers some protection from the case that  $i$ th point is an outlier.

Q: Can you figure out why  $p_{ii}$  is between 0 and 1?

- Intuitively, the leverage score  $p_{ii}$  measures the “the amount of support” that the  $i$ th data point provided to the regression surface.
- A large leverage score for the  $i$ th point means the  $i$ th point is high leverage, and regression surface alters significantly if  $y_i$  takes a different value.
- A high leverage point is not necessarily **influential**, or an **influential point**



- An **influential point** is a point whose deletion would significantly alter the regression surface. There are several detection or quantification methods:

1. Standardised difference in coefficients

$$\frac{\hat{\beta}_j - \hat{\beta}_j(-i)}{\text{SE}(\hat{\beta}_j)}$$

where  $\hat{\beta}_j(-i)$  is the estimate of  $\beta_j$  after the  $i$ th data point has been deleted.

2. Standardised difference in fitted values

$$\frac{\hat{Y}_i - \hat{Y}_i(-i)}{\text{SE}(\hat{Y}_i)}$$

- The standard errors are based on an estimate of  $\sigma$  without the  $i$ th data.

3. Cook's distance,  $D_i$ , is based on the idea of **confidence ellipsoid**

$$\left\{ \boldsymbol{\beta}: \frac{(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})^T \mathbf{X}^T \mathbf{X} (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})}{(k+1)\hat{\sigma}^2} \leq F_{k+1, n-k-1}(\alpha) \right\}$$

which gives  $100 \cdot (1 - \alpha)\%$  confidence ellipsoid for  $\boldsymbol{\beta}$ .

- The idea is to define

$$D_i = \frac{(\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}(-i))^T \mathbf{X}^T \mathbf{X} (\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}(-i))}{(k+1)\hat{\sigma}^2}$$

to quantify the whether  $\hat{\boldsymbol{\beta}}(-i)$  is essentially the same as  $\hat{\boldsymbol{\beta}}$  by comparing

$$D_i \quad \text{with} \quad F_{k+1, n-k-1}(\alpha)$$

## 3.6 Case Study

### Yield Study of a chemical process

- The following data were gathered in the course of studying the yield of a particular chemical process.

**yield** measures the effectiveness of a synthetic procedure

**conversion** a percentage of desired over undesired product(s)

**flow** measures the speed of the reaction process

**ratio** a ratio of between two chemicals

- Chemical theory predicts **yield** is related to the reciprocal of **ratio**.
- Theory also predicts **yield** is related to **conversion\*flow**.
- We are interested in the relation between the yield and the other variables.
- In particular, we would like to see whether the data support the two theories.

- In practice, after loading the data

```
> chem_pro.df = read.table("~/Desktop/chem_pro.csv",
+                           sep = ",", header = TRUE)
```

we should always check the data frame for issues

```
> str(chem_pro.df)
```

```
'data.frame': 44 obs. of 4 variables:
$ yield   : num  55.5 54.8 52.2 50.4 49.3 ...
$ conversion: num  11.8 11.9 12.1 12.1 12 ...
$ flow     : num  119 105 97 101 44 ...
$ ratio    : Factor w/ 40 levels "0.036","0.089",...
```

- In this case, somehow `ratio` is a `Factor`,

```
> class(chem_pro.df$ratio)
```

```
[1] "factor"
```

- It is often a result of typos or having inconsistent data format in the data file.
- We look into the values that this factor variable can take

```
> levels(chem_pro.df$ratio)
```

```
[1] "0.036" "0.089" "0.094" "0.097"
[5] "0.1"   "0.108" "0.113" "0.116"
[9] "0.123" "0.126" "0.135" "0.136"
[13] "0.143" "0.152" "0.153" "0.155"
[17] "0.16"  "0.161" "0.164" "0.166"
[21] "0.169" "0.17"  "0.18"  "0.183"
[25] "0.184" "0.188" "0.192" "0.194"
[29] "0.195" "0.197" "0.201" "0.211"
[33] "0.215" "0.221" "0.222" "0.223"
[37] "0.225" "0.229" "0.233" "0>163"
```

- Identifying the observation, which is clearly a typo,

```
> ratio_typo = which(chem_pro.df$ratio == "0>163")
> ratio_typo
```

```
[1] 32
```

- Correcting the typo by first converting it into a character variable

```
> chem_pro.df$ratio = as.character(chem_pro.df$ratio)
```

- Reassigning the value 0.163 instead of "0>163"

```
> chem_pro.df$ratio[ratio_typo] = "0.163"
```

- Coercing it into a double precision numeric variable

```

> chem_pro.df$ratio = as.double(chem_pro.df$ratio)
> chem_pro.df$ratio

[1] 0.155 0.089 0.094 0.108 0.100 0.036 0.113
[8] 0.123 0.135 0.183 0.166 0.221 0.192 0.188
[15] 0.201 0.153 0.194 0.097 0.136 0.143 0.116
[22] 0.195 0.160 0.164 0.197 0.233 0.211 0.222
[29] 0.223 0.229 0.170 0.163 0.153 0.180 0.126
[36] 0.152 0.184 0.225 0.169 0.161 0.197 0.201
[43] 0.221 0.215

```

Q: Is there any more obvious error in the dataset?

```
> summary(chem_pro.df)
```

|         | yield  | conversion     | flow          | ratio          |
|---------|--------|----------------|---------------|----------------|
| Min.    | :40.05 | Min. :-11.12   | Min. : 30.0   | Min. :0.0360   |
| 1st Qu. | :48.05 | 1st Qu.: 11.19 | 1st Qu.:221.5 | 1st Qu.:0.1358 |
| Median  | :51.28 | Median : 11.89 | Median :325.0 | Median :0.1675 |
| Mean    | :50.68 | Mean : 11.11   | Mean :291.9   | Mean :0.1658   |
| 3rd Qu. | :53.91 | 3rd Qu.: 12.04 | 3rd Qu.:380.0 | 3rd Qu.:0.1980 |
| Max.    | :59.34 | Max. : 12.36   | Max. :523.0   | Max. :0.2330   |

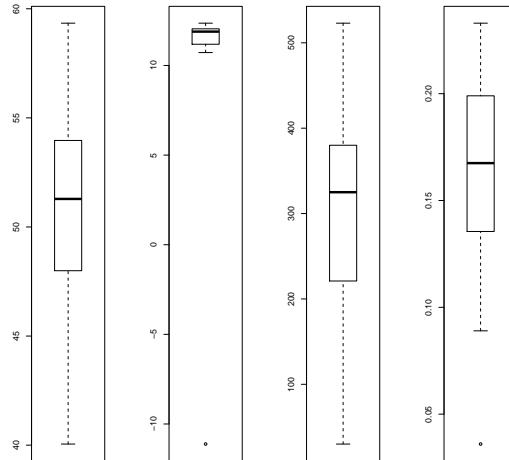
- Boxplot as well as summary is particularly useful to identify unusual values

```

> par(mfrow = c(1,4))
> lapply(chem_pro.df, boxplot) # do for every column
> par(mfrow = c(1,1))

```

- We should take a note and report later when there is any suspicion of error in the data, since every data point matters when there are only 44 of them.
- This procedure of correcting obvious typos and reporting potential errors in the dataset is often known as [data cleaning](#).



- There seems to be 1 unusually small value in variable 2 and one in variable 4.

- Identifying both of them

```
> names(chem_pro.df)
```

|             |              |        |         |
|-------------|--------------|--------|---------|
| [1] "yield" | "conversion" | "flow" | "ratio" |
|-------------|--------------|--------|---------|

```
> conversion_typo = which(
+   chem_pro.df$conversion <= -10)
>
> ratio_unusual = which(
+   chem_pro.df$ratio <= 0.05)
>
> conversion_typo; ratio_unusual
```

|        |
|--------|
| [1] 44 |
|--------|

|       |
|-------|
| [1] 6 |
|-------|

- From the description of the data, `conversion` can only be between 0 and 1, but observation 6, the unusually small `ratio` value, might just be unusual.

- According to the summary,

```
> summary(chem_pro.df$conversion)
```

| Min.   | 1st Qu. | Median | Mean  | 3rd Qu. | Max.  |
|--------|---------|--------|-------|---------|-------|
| -11.12 | 11.19   | 11.89  | 11.11 | 12.04   | 12.36 |

it is likely that the minus sign is a typo.

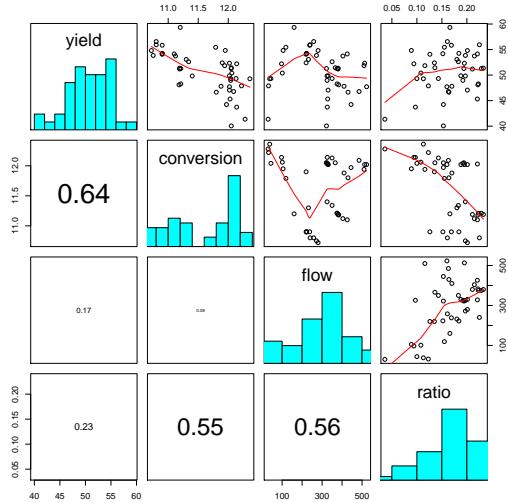
- So we modify the 44th observation and keep record of it

```
> chem_pro.df$conversion[conversion_typo] =
+   - chem_pro.df$conversion[conversion_typo]
```

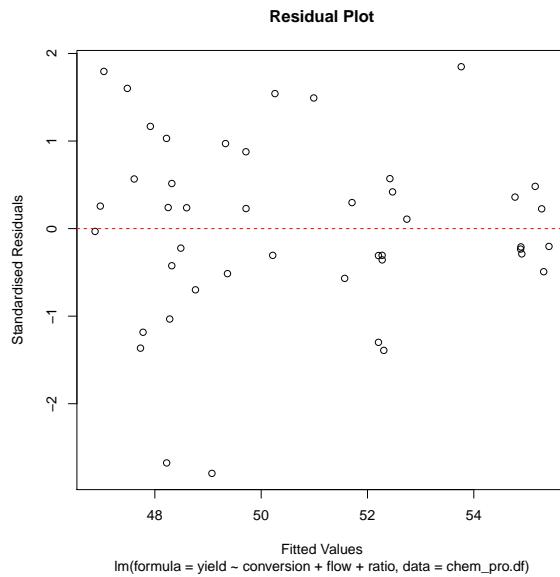
- We have to keep observation 6 on watch list, it may be a high leverage point.

- Observations can be unusual in terms of relationships amongst variables, so plotting two variables at a time is useful, which is known as a pairs plot

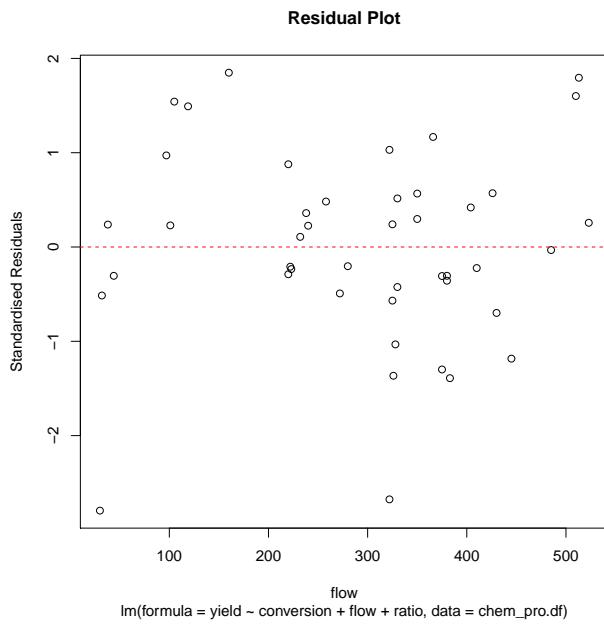
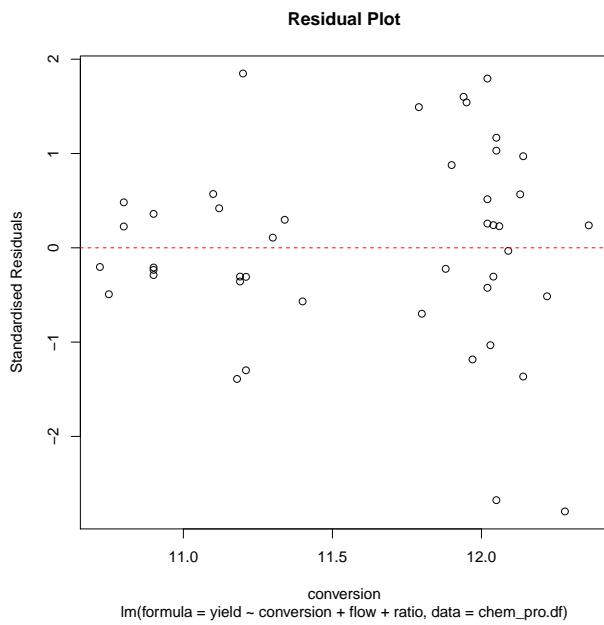
```
> ?pairs # See the help document for how to use it
```

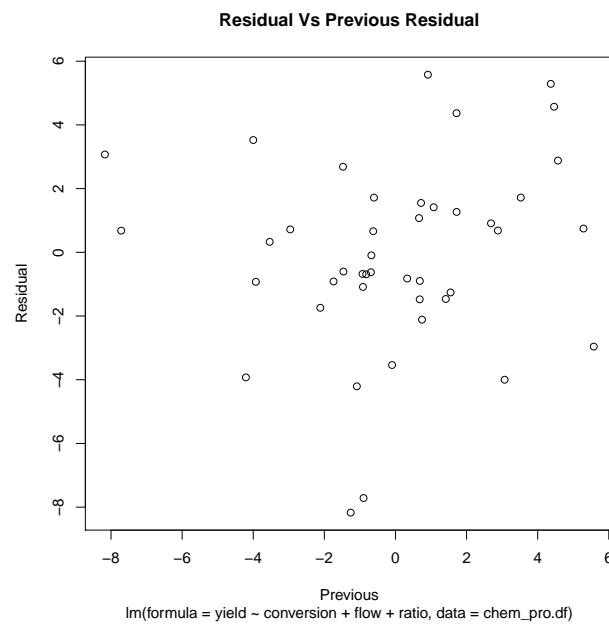
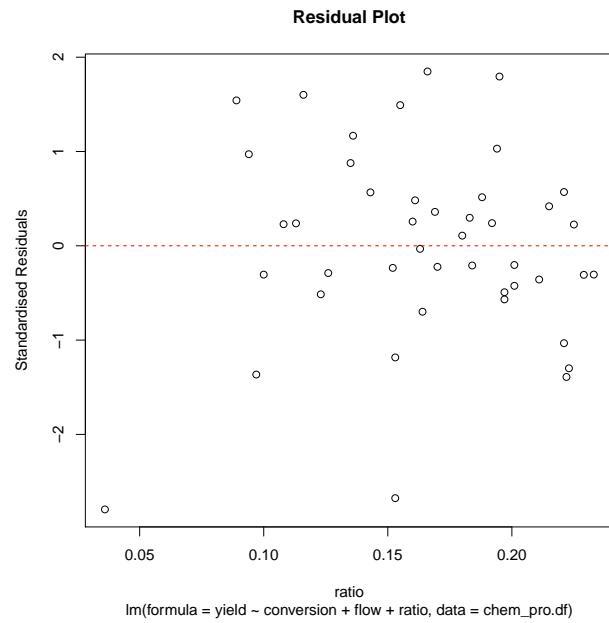


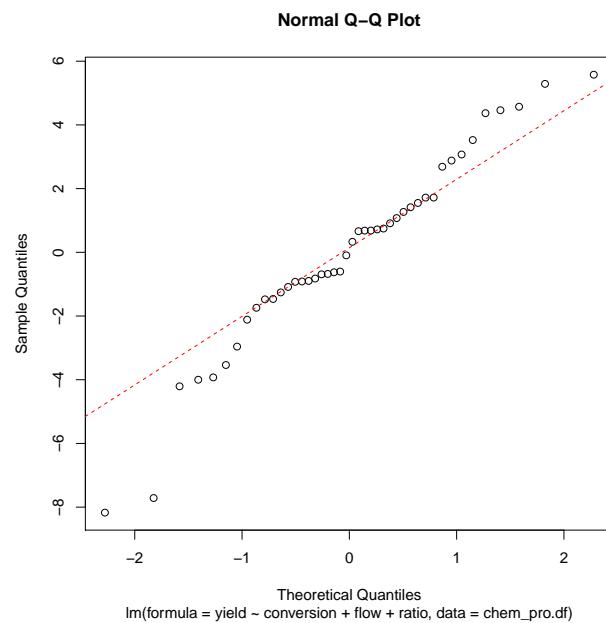
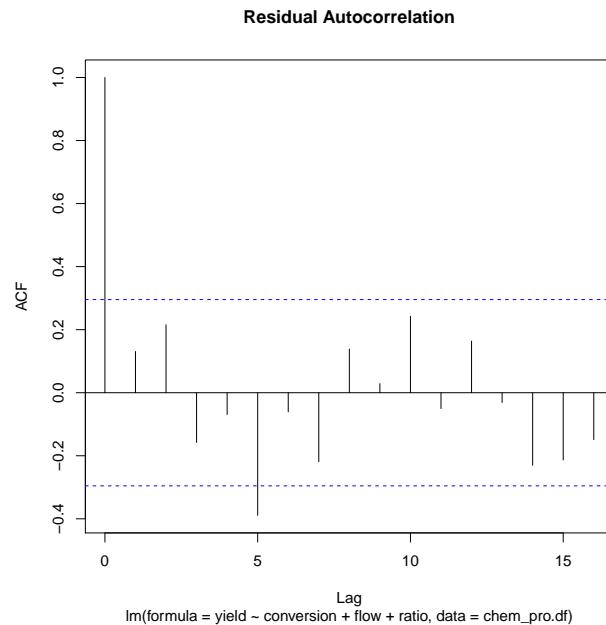
- No more unusual point other than the 44th, and 6th seems to be consistent.
- It is also useful for seeing the spread and detecting strong linear relationships.
- We have no evidence against the multiple linear regression model being valid.



This simple model will be used as a yardstick for comparison if it is valid.







- The normal QQ plot is the only plot that deserves a second look since

```
> nrow(chem_pro.df)
```

```
[1] 44
```

```

> model

Call:
lm(formula = yield ~ conversion + flow + ratio, data = chem_pro.df)

Coefficients:
(Intercept) conversion      flow        ratio      
 113.011270   -5.189435    -0.006983    -0.055762

> res = residuals(model)
>
> shapiro.test(res)

Shapiro-Wilk normality test

data: res
W = 0.96238, p-value = 0.1597

```

Q: Given we have no evidence against the model being valid,

```

> summary(model)

Call:
lm(formula = yield ~ conversion + flow + ratio, data = chem_pro.df)

Residuals:
    Min      1Q  Median      3Q      Max 
-8.1715 -1.3101  0.1171  1.5926  5.5769 

Coefficients:
            Estimate Std. Error t value Pr(>t)
(Intercept) 113.011270 14.631935 7.724 1.88e-09 ***
conversion  -5.189435  1.149651 -4.514 5.49e-05 ***
flow        -0.006983  0.004467 -1.563  0.126  
ratio       -0.055762 15.755755 -0.004  0.997  
---
Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.118 on 40 degrees of freedom
Multiple R-squared:  0.46,    Adjusted R-squared:  0.4195 
F-statistic: 11.36 on 3 and 40 DF,  p-value: 1.591e-05

```

what can we conclude from the above summary output?

- Of course, being a valid model doesn't mean it is the best model.
- One aspect of a model that is not addressed by its validity is the "stability" of the regression surface.
- This can be partly addressed by looking at the variance inflation factor (VIF)

$$\frac{1}{1 - R_j^2}$$

- If it is big, e.g. greater than 10, then we will have to reconsider our model.

Q: What does it mean to have a VIF bigger than 10?

- It can be shown the  $j$ th VIF is just the  $j$ th diagonal element of

$$\mathbf{R}^{-1}$$

where  $\mathbf{R}$  is the sample correlation matrix for the  $k$  predictors.

- Recall the  $j$ th variance inflation factor is given by

$$\frac{1}{1 - R_j^2}$$

which is the first factor of

$$\text{Var} [\hat{\beta}_j \mid \mathbf{X}] = \frac{1}{1 - R_j^2} \frac{\sigma^2}{(n - 1)s_{x_j}^2}$$

which is given by the  $j$ th diagonal element of the following

$$\text{Var} [\hat{\boldsymbol{\beta}} \mid \mathbf{X}] = \sigma^2 (\mathbf{X}^T \mathbf{X})^{-1}$$

- Recall the sample Pearson's correlation is given by

$$\begin{aligned}
r_{ij} &= \frac{\sum_{\ell=1}^n (x_{\ell i} - \bar{x}_i)(x_{\ell j} - \bar{x}_j)}{\sqrt{\sum_{\ell=1}^n (x_{\ell i} - \bar{x}_i)^2} \sqrt{\sum_{\ell=1}^n (x_{\ell j} - \bar{x}_j)^2}} \\
&= \frac{1}{n-1} \frac{\sum_{\ell=1}^n (x_{\ell i} - \bar{x}_i)(x_{\ell j} - \bar{x}_j)}{\sqrt{\frac{1}{n-1} \sum_{\ell=1}^n (x_{\ell i} - \bar{x}_i)^2} \sqrt{\frac{1}{n-1} \sum_{\ell=1}^n (x_{\ell j} - \bar{x}_j)^2}} \\
&= \frac{1}{n-1} \sum_{\ell=1}^n \frac{(x_{\ell i} - \bar{x}_i)}{s_i} \frac{(x_{\ell j} - \bar{x}_j)}{s_j}
\end{aligned}$$

- In matrix notation, suppose the usual data matrix is partitioned such that

$$\mathbf{X} = [\mathbf{1} \quad \mathbf{X}_{-1}]$$

then

$$\mathbf{R} = \frac{1}{n-1} \mathbf{X}_{cs}^T \mathbf{X}_{cs}$$

where  $\mathbf{X}_{cs} = \mathbf{C}\mathbf{X}_{-1}\mathbf{D}^{-1}$  with  $\mathbf{C} = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T$  and  $\mathbf{D} = \text{diag}(s_1, s_2, \dots, s_k)$ .

- Instead of the original model,

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_k x_{ik} + e_i$$

consider  $\alpha_0 = \beta_0 + \beta_1 \bar{x}_1 + \beta_2 \bar{x}_2 + \cdots + \beta_k \bar{x}_k$ , then

$$\begin{aligned}
y_i &= \alpha_0 + \beta_1 (x_{i1} - \bar{x}_1) + \beta_2 (x_{i2} - \bar{x}_2) + \cdots + \beta_k (x_{ik} - \bar{x}_k) + e_i \\
&= \alpha_0 + \beta_1 \tilde{x}_{i1} + \beta_2 \tilde{x}_{i2} + \cdots + \beta_k \tilde{x}_{ik} + e_i
\end{aligned}$$

and scaling is done by considering the following

$$x_{ij}^* = \frac{x_{ij} - \bar{x}_j}{s_j} = \frac{\tilde{x}_{ij}}{s_j}$$

which means  $\gamma_j = \beta_j s_j$  and  $\hat{\gamma}_j = \hat{\beta}_j s_j$  and

$$y_i = \alpha_0 + \gamma_1 x_{i1}^* + \gamma_2 x_{i2}^* + \cdots + \gamma_k x_{ik}^* + e_i$$

- It can be proved by considering the relationship between

$$\text{Var}[\hat{\beta} | \mathbf{X}] \quad \text{and} \quad \text{Var}[\hat{\gamma} | \mathbf{X}]$$

the later variance which can be written in terms of the correlation matrix  $\mathbf{R}$ .

- Thus the easiest way to obtain VIF is the following

```
> Xminus1 = chem_pro.df[, -1]
> VIF = diag(solve(cor(Xminus1)))
> VIF
```

| conversion | flow     | ratio    |
|------------|----------|----------|
| 1.580323   | 1.606860 | 2.276409 |

which shows we have no multicollinearity problem.

- Unusual points also effect the “stability”, so need to be checked/reported.
- It can be shown the sum of leverage scores

$$\frac{\partial \hat{y}_i}{\partial y_i} = p_{ii}, \quad \text{where } \mathbf{P} = \mathbf{X} \left( \mathbf{X}^T \mathbf{X} \right)^{-1} \mathbf{X}^T$$

is given by the rank of  $\mathbf{X}$ , i.e. it is the number of linearly independent columns in  $\mathbf{X}$  when  $n \geq k + 1$ , which means the sum is equal to  $k + 1$ .

- Ideally we wish this “total leverage” of  $k + 1$  to be shared equally, that is,

$$\frac{k+1}{n}$$

by the  $n$  observations, any particular observation having more than

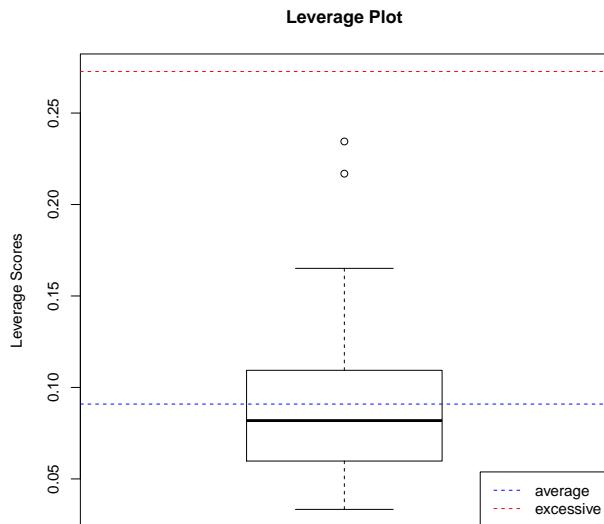
$$\frac{3(k+1)}{n}$$

is deemed to have “excessive leverage” and has the potential of exerting a very large influence over the position of the fitted regression surface.

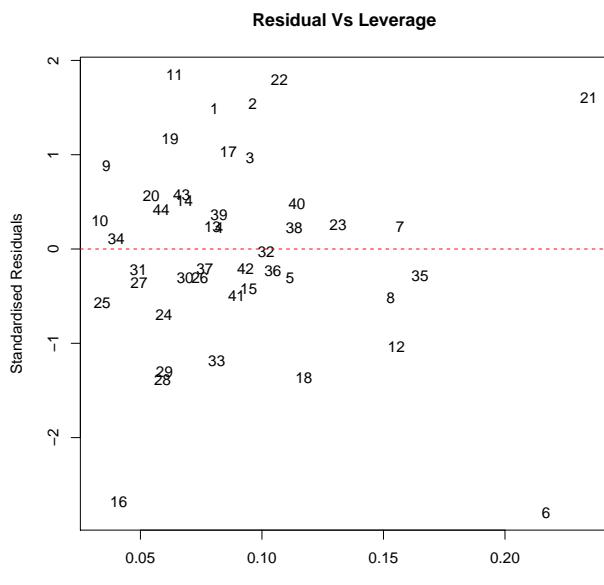
- We need to report if there is any observation that has “excessive leverage”, especially so if it is also an outlier, the regression plane is unstable due to it.

```
> pii.vec = hatvalues(model)
> order(pii.vec, decreasing = TRUE)
```

|      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| [1]  | 21 | 6  | 35 | 7  | 12 | 8  | 23 | 18 | 40 | 38 | 5  | 22 | 36 | 32 | 2  |
| [16] | 3  | 15 | 42 | 41 | 17 | 4  | 39 | 33 | 1  | 13 | 37 | 26 | 30 | 14 | 43 |
| [31] | 11 | 19 | 29 | 24 | 28 | 44 | 20 | 27 | 31 | 16 | 34 | 9  | 25 | 10 |    |



```
lm(formula = yield ~ conversion + flow + ratio, data = chem_pro.df)
```



```
lm(formula = yield ~ conversion + flow + ratio, data = chem_pro.df)
```

- Another way to determine whether our model is excessively influenced by a point is to explicitly consider “leave-one-out” influence measures

```
> im = influence.measures(model)
> im # Report and possibly delete influential points
```

```
Influence measures of
  lm(formula = yield ~ conversion + flow + ratio, data = chem_pro.df) :

      dfb.1_   dfb.cnvr dfb.flow dfb.ratidffit cov.r    cook.d     hat inf
1 -0.15379  0.171205 -0.37075  0.21414  0.4483 0.957 4.87e-02 0.0804
2  0.10348 -0.044898 -0.11423 -0.24833  0.5123 0.958 6.33e-02 0.0962
3 -0.02789  0.062869 -0.12017 -0.07752  0.3146 1.111 2.48e-02 0.0951
4 -0.01039  0.017236 -0.03453 -0.00626  0.0678 1.200 1.18e-03 0.0823
5  0.01094 -0.021314  0.06432  0.00689 -0.1070 1.234 2.93e-03 0.1116
6 -0.31180  0.115894  0.17695  0.97151 -1.6193 0.592 5.41e-01 0.2169 *
7 -0.05076  0.058095 -0.07330  0.03086  0.1015 1.305 2.64e-03 0.1569 *
8  0.10158 -0.115249  0.17135 -0.08036 -0.2165 1.272 1.19e-02 0.1530
:
35 -0.11592  0.110300 -0.03173  0.09279 -0.1270 1.314 4.13e-03 0.1651 *
36 -0.06936  0.067273 -0.00626  0.04110 -0.0790 1.229 1.60e-03 0.1045
37 -0.03747  0.038115  0.01600  0.00120 -0.0594 1.193 9.04e-04 0.0765
38  0.01881 -0.023149 -0.03880  0.03878  0.0797 1.242 1.63e-03 0.1133
39  0.08636 -0.085527 -0.00198 -0.03544  0.1066 1.190 2.91e-03 0.0823
40  0.15475 -0.152393  0.03017 -0.08905  0.1717 1.221 7.51e-03 0.1144
41 -0.10567  0.110095  0.01319  0.00863 -0.1529 1.186 5.96e-03 0.0896
42 -0.04383  0.046000  0.00491  0.00218 -0.0645 1.215 1.06e-03 0.0933
43  0.03744 -0.047960  0.04730  0.02774  0.1516 1.148 5.85e-03 0.0670
44  0.02885 -0.035479  0.02799  0.01741  0.1034 1.155 2.73e-03 0.0585
```

Q: Does the data support the idea that yield depends on the reciprocal of ratio?

```
> recip.LM = lm(yield ~ conversion + flow +
+                   I(1/ratio), data = chem_pro.df)

> summary(recip.LM)
```

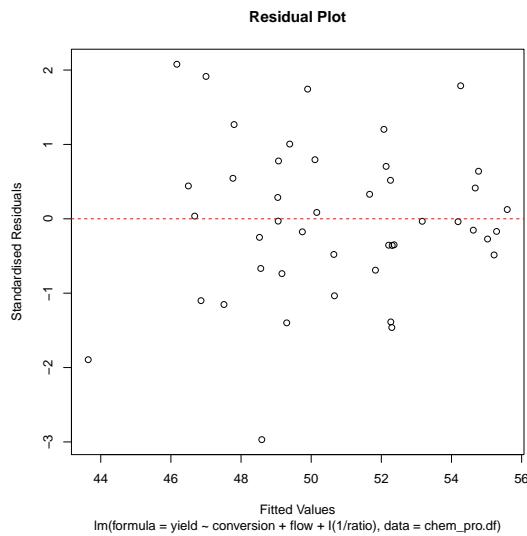
```
Call:
lm(formula = yield ~ conversion + flow + I(1/ratio), data = chem_pro.df)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.5413 -1.5016 -0.1017  1.6203  5.6798 

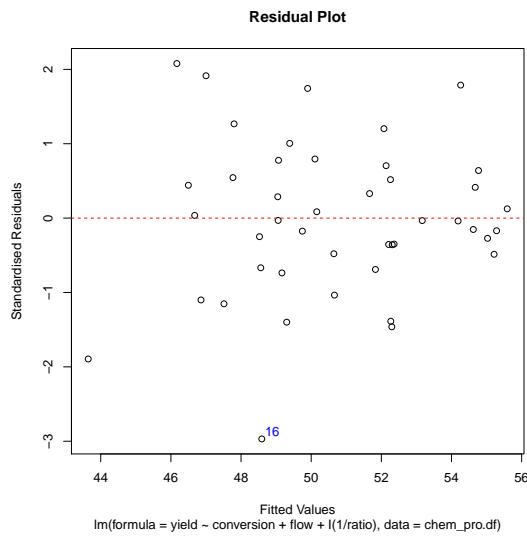
Coefficients:
            Estimate Std. Error t value Pr(>t)
(Intercept) 105.935729 10.694444  9.906 2.54e-12 ***
conversion  -4.263058  0.961839 -4.432 7.08e-05 ***
flow        -0.011542  0.003921 -2.944 0.00537 ** 
I(1/ratio) -0.345478  0.155987 -2.215 0.03253 *  
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

Residual standard error: 2.943 on 40 degrees of freedom
Multiple R-squared:  0.519,    Adjusted R-squared:  0.4829 
F-statistic: 14.39 on 3 and 40 DF,  p-value: 1.663e-06
```

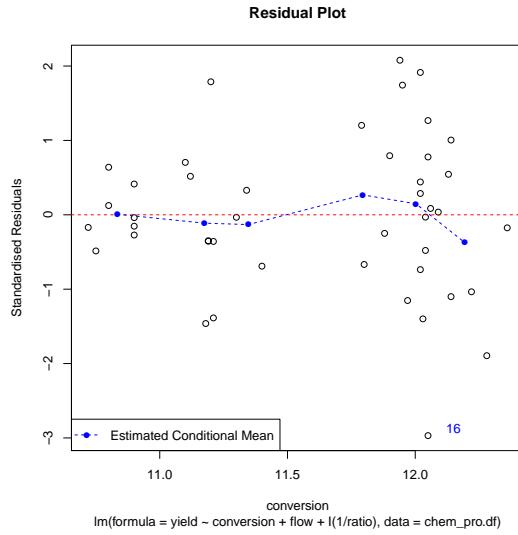
Q: Can we trust this model? Of course, we have to check the residuals first.



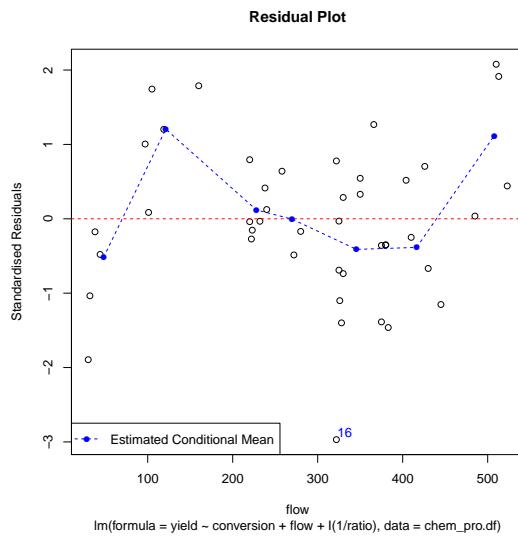
```
> recip_outlier_index = which( sres < -2.5 )
```



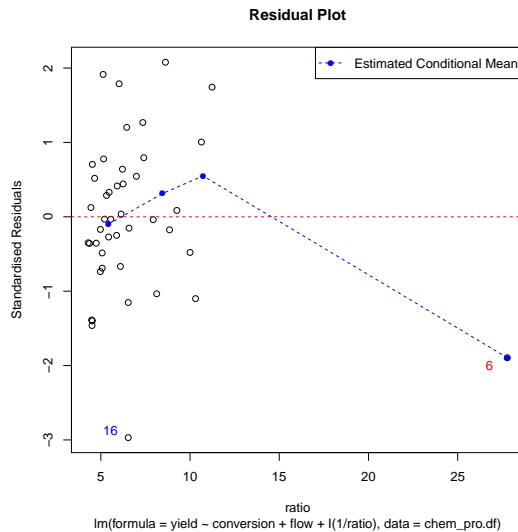
- Notice the outlier, but it seems no transformation is needed for **conversion**.



- It is a different story for **flow**, there seems to be a cubic relationship.



- The curvature is clearly due to the high leverage point.



- Identifying the high leverage point,

```
> recip_hl_index = which(1/chem_pro.df$ratio > 25)
> recip_hl_index
```

```
[1] 6
```

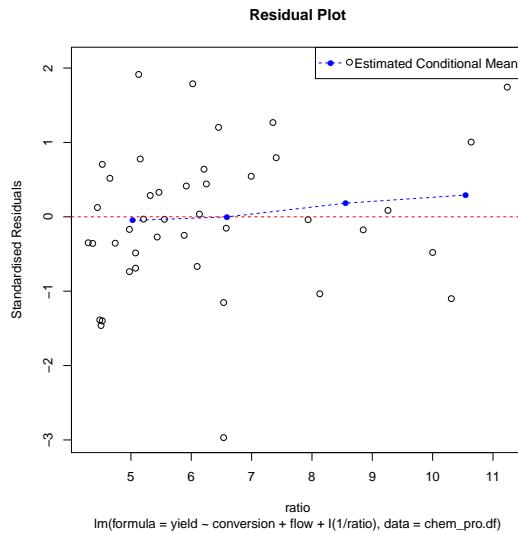
```
> ratio_unusual
```

```
[1] 6
```

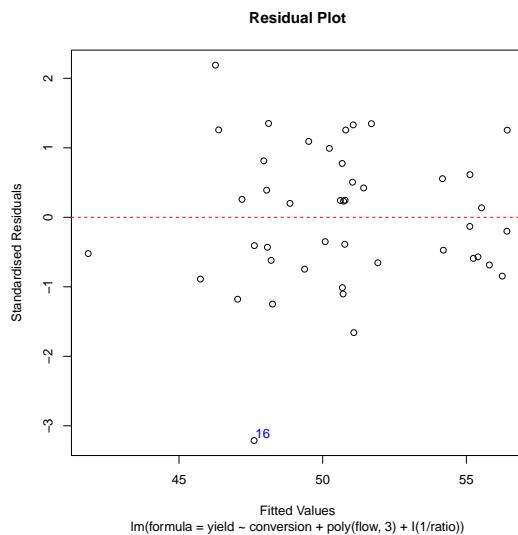
```
> chem_pro.df[ratio_unusual ,]
```

```
yield conversion flow ratio
6 41.36      12.28    30 0.036
```

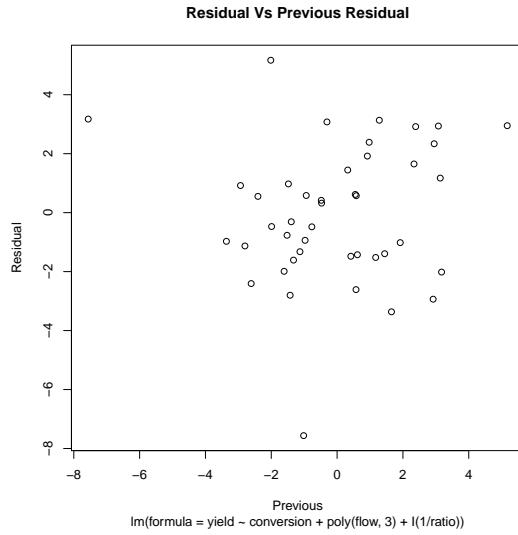
- We can see the effect this point on the trend line by leaving it out.
- No more transformation is need for **ratio**.



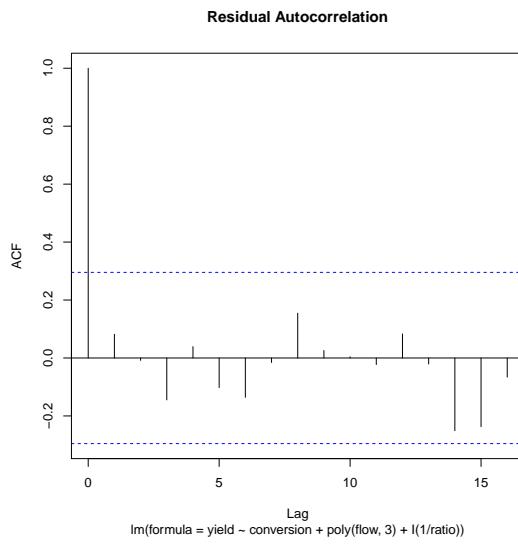
```
> cubic.LM = lm(yield ~ conversion + poly(flow, 3)
+ I(1/ratio), data = chem_pro.df)
```



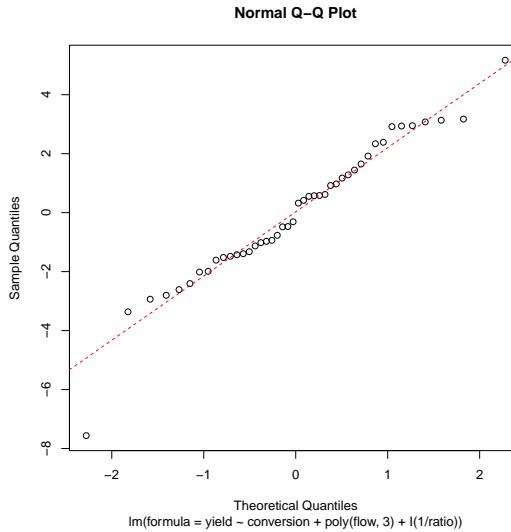
- Other than the outlier, it looks fine.



- We have no evidence against the independence assumption.



- Other than the outlier, QQ-normal plot seems to be fine.



```
> summary(cubic.LM)
```

```
Call:
lm(formula = yield ~ conversion + poly(flow, 3) + I(1/ratio),
    data = chem_pro.df)

Residuals:
    Min      1Q  Median      3Q     Max 
-7.562 -1.440  0.007  1.497  5.170 

Coefficients:
            Estimate Std. Error t value Pr(>t)
(Intercept) 101.6345   10.4408  9.734 7.18e-12 ***
conversion  -4.1717    0.9105 -4.582 4.86e-05 ***
poly(flow, 3)1 -10.4209   3.0096 -3.462 0.001340 ** 
poly(flow, 3)2   3.7437   3.1710  1.181 0.245090  
poly(flow, 3)3  10.0417   2.5762  3.898 0.000382 *** 
I(1/ratio)   -0.3644    0.1404 -2.594 0.013386 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Residual standard error: 2.469 on 38 degrees of freedom
Multiple R-squared:  0.6783, Adjusted R-squared:  0.6359 
F-statistic: 16.02 on 5 and 38 DF,  p-value: 1.742e-08
```

- This model is reasonably good, it explains 67.83% of variability in `yield`.
- The data support the claim that `yield` relates to `1/ratio` under this model.
- We press on and exam the other claim, that is, `yield` is related to

conversion\*flow

```
> prod.LM =
+   lm(yield ~ conversion + poly(flow, 3) +
+        I(1/ratio) + I(flow*conversion),
+        data = chem_pro.df)
```

- After examining the residuals, it seems all assumptions are satisfied, but

```
> summary(prod.LM)
```

```

Call:
lm(formula = yield ~ conversion + poly(flow, 3) + I(1/ratio) +
    I(flow * conversion), data = chem_pro.df)

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 99.42924   10.67628   9.313 3.08e-11 ***
conversion  -0.18663    4.11546  -0.045  0.96407
poly(flow, 3)1 127.62434  139.06123   0.918  0.36469
poly(flow, 3)2  1.73942   3.75947   0.463  0.64631
poly(flow, 3)3 13.30024   4.17243   3.188  0.00291 **
I(1/ratio)   -0.35397   0.14086  -2.513  0.01646 *
I(flow * conversion) -0.01305   0.01314  -0.993  0.32720

```

- Notice `conversion` and `flow` were highly significant

```

> conversion = chem_pro.df$conversion
>
> flow = poly(chem_pro.df$flow, 3)
>
> recip_ratio = 1/chem_pro.df$ratio
>
> prod = chem_pro.df$flow * chem_pro.df$conversion
>
> Xtd = cbind(conversion, flow, recip_ratio, prod)

```

- Recall variance inflation factor of more than 10 is considered to be large,

```

> VIF = diag(solve(cor(Xtd)))
>
> VIF

```

|            | 1         | 2           | 3        | recip_ratio | prod        |
|------------|-----------|-------------|----------|-------------|-------------|
| conversion | 32.276768 | 3169.973656 | 2.316849 | 2.853786    | 1.932279    |
|            |           |             |          |             | 3132.705871 |

- Eigenvalues of  $\tilde{\mathbf{X}}^T \tilde{\mathbf{X}}$  confirms, we definitely have collinearity problem

```

> eigen.stuff = eigen(cor(Xtd))
>
> eigen.stuff$values

```

|     |              |              |              |              |              |              |
|-----|--------------|--------------|--------------|--------------|--------------|--------------|
| [1] | 2.4837616083 | 1.7021510928 | 1.0013419291 | 0.4480934145 | 0.3644941045 | 0.0001578508 |
|-----|--------------|--------------|--------------|--------------|--------------|--------------|

```
> round(eigen.stuff$vectors, 2)
```

|      | [,1]  | [,2]  | [,3]  | [,4]  | [,5]  | [,6]  |
|------|-------|-------|-------|-------|-------|-------|
| [1,] | 0.28  | -0.56 | 0.03  | 0.76  | 0.16  | 0.07  |
| [2,] | -0.55 | -0.35 | 0.00  | -0.17 | 0.20  | 0.71  |
| [3,] | 0.23  | -0.57 | -0.35 | -0.36 | -0.61 | -0.01 |
| [4,] | -0.09 | 0.21  | -0.94 | 0.18  | 0.20  | 0.02  |
| [5,] | 0.52  | -0.19 | -0.04 | -0.48 | 0.68  | 0.00  |
| [6,] | -0.53 | -0.40 | -0.02 | -0.08 | 0.24  | -0.70 |

- It seems we have to go without `conversion*flow`.
- It seems that `conversion*flow` is unnecessary according to this dataset.

- We move back to the cubic model,

```
> cubic.LM = lm(yield ~ conversion + poly(flow, 3)
+                   + I(1/ratio), data = chem_pro.df)
```

and investigate the effect of unusual points on the model `cubic.LM`.

```
> model$call
```

```
lm(formula = yield ~ conversion + poly(flow, 3) + I(1/ratio), data = chem_pro.df)
```

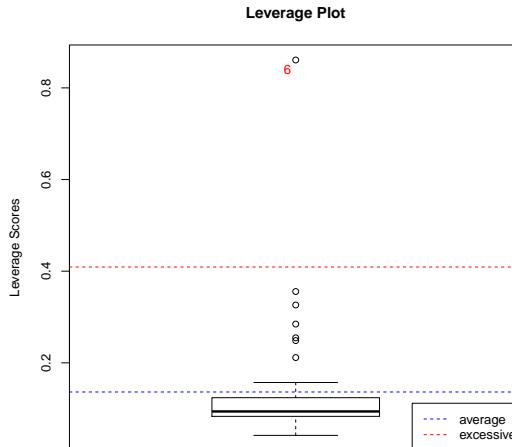
```
> pii.vec = hatvalues(model)
> order(pii.vec, decreasing = TRUE)
```

```
[1] 6 8 23 7 22 21 5 18 32 1 9 43 3 4 35 2 11 44 20 36 38 12
[23] 42 17 37 16 13 40 41 15 19 14 39 28 26 27 30 33 29 34 24 31 10 25
```

```
> influence.measures(model)
```

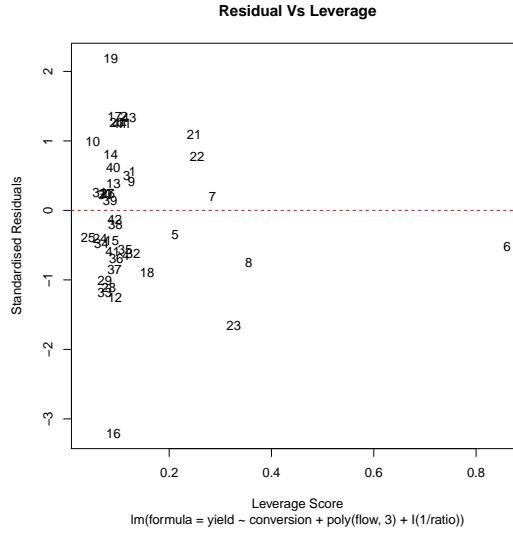
|    | dfb.1.    | dfb.cnvr | dfb.p..3.1 | dfb.p..3.2 | dfb.p..3.3 | dfb.I.1. | dffit   | cov.r | cook.d   |
|----|-----------|----------|------------|------------|------------|----------|---------|-------|----------|
| 6  | -0.184018 | 0.27195  | -0.26389   | 4.31e-02   | 0.21870    | -1.07461 | -1.2852 | 8.077 | 0.280687 |
| 7  | -0.009520 | 0.01655  | -0.09130   | 6.73e-02   | -0.05045   | -0.06502 | 0.1246  | 1.630 | 0.002654 |
| 8  | -0.029285 | -0.00373 | 0.40018    | -3.36e-01  | 0.26729    | 0.30531  | -0.5513 | 1.667 | 0.051257 |
| 16 | 0.833470  | -0.83617 | -0.23336   | 8.30e-01   | -0.01424   | -0.08045 | -1.1740 | 0.193 | 0.171884 |

- Point 6 definitely has excessive leverage.



`lm(formula = yield ~ conversion + poly(flow, 3) + I(1/ratio))`

- Point 16 is an outlier



- But 6 is NOT an outlier despite being picked up by influential measures.
- Recall the following are the rows that are picked up by influential measures

```

      dfb.1_ dfb.cnvr dfb.p..3.1 dfb.p..3.2 dfb.p..3.3 dfb.I.1.   dfit cov.r cook.d
6   -0.184018  0.27195  -0.26389  4.31e-02  0.21870 -1.07461 -1.2852 8.077 0.280687
7    0.009520  0.01655  -0.09130  6.73e-02  -0.05045 -0.06502 0.1246 1.630 0.002654
8   -0.029285 -0.00373  0.40018 -3.36e-01  0.26729  0.30531 -0.5513 1.667 0.051257
16   0.833470 -0.83617  -0.23336  8.30e-01  -0.01424 -0.08045 -1.1740 0.193 0.171884

```

- So point 6 is a high leverage influential point, and 16 is an influential outlier.

```
> recip_outlier_index
```

```
16
```

- Refit the model without the two observations,

```

> cubic.rm.LM =
+   lm(yield~conversion
+       + poly(flow,3) + I(1/ratio),
+       data = chem_pro.df[-c(6,16),])
>
> summary(cubic.rm.LM)

```

- The current cubic model and the previous model are shown below

```

Call:
lm(formula = yield ~ conversion + poly(flow, 3) + I(1/ratio),
    data = chem_pro.df[-c(6, 16), ])

Coefficients:
            Estimate Std. Error t value Pr(>t)
(Intercept)  96.0543   9.8763   9.726  1.3e-11 ***
conversion   -3.7628   0.9155  -4.110  0.000218 ***
poly(flow, 3)1 -9.2321   2.7876  -3.312  0.002117 **
poly(flow, 3)2  2.0092   2.6767   0.751  0.457747
poly(flow, 3)3  9.0940   2.3355   3.894  0.000410 ***

```

```

I(1/ratio)      -0.2025     0.2828   -0.716  0.478691
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?. 0.1 ? ? 1

Residual standard error: 2.155 on 36 degrees of freedom
Multiple R-squared:  0.6727,    Adjusted R-squared:  0.6272
F-statistic: 14.8 on 5 and 36 DF,  p-value: 6.765e-08

```

---

| Coefficients:  |          |            |         |              |  |
|----------------|----------|------------|---------|--------------|--|
|                | Estimate | Std. Error | t value | Pr(>t)       |  |
| (Intercept)    | 101.6345 | 10.4408    | 9.734   | 7.18e-12 *** |  |
| conversion     | -4.1717  | 0.9105     | -4.582  | 4.86e-05 *** |  |
| poly(flow, 3)1 | -10.4209 | 3.0096     | -3.462  | 0.001340 **  |  |
| poly(flow, 3)2 | 3.7437   | 3.1710     | 1.181   | 0.245090     |  |
| poly(flow, 3)3 | 10.0417  | 2.5762     | 3.898   | 0.000382 *** |  |
| I(1/ratio)     | -0.3644  | 0.1404     | -2.594  | 0.013386 *   |  |

- It seems observation 6 and 16 are also problematic for prod.LM, however,

```

Call:
lm(formula = yield ~ conversion + poly(flow, 3) + I(1/ratio) +
  I(flow * conversion), data = chem_pro.df[-c(6, 16), ])

Coefficients:
              Estimate Std. Error t value Pr(>t)
(Intercept)      94.57208   9.93354  9.520 3.02e-11 ***
conversion        0.15272   3.63512   0.042   0.9667
poly(flow, 3)1    118.75295  115.04943   1.032   0.3091
poly(flow, 3)2     0.42501   3.02399   0.141   0.8890
poly(flow, 3)3    12.19298   3.62966   3.359   0.0019 **
I(1/ratio)       -0.20802   0.28194   -0.738   0.4655
I(flow * conversion) -0.01273  0.01144  -1.113   0.2734
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?. 0.1 ? ? 1

Residual standard error: 2.148 on 35 degrees of freedom
Multiple R-squared:  0.6839,    Adjusted R-squared:  0.6297
F-statistic: 12.62 on 6 and 35 DF,  p-value: 1.567e-07

```

---

and eigenvalues/eigenvectors give similar conclusions.

```
> VIF = diag(solve(cor(Xtd[-c(6,16),]))); VIF
```

|            |             |          |               |                      |
|------------|-------------|----------|---------------|----------------------|
| conversion | 1           | 2        | 3 recip_ratio | prod                 |
| 31.416465  | 2889.153756 | 2.177048 | 2.895338      | 2.507123 2872.891795 |

---

- It seems 1/ratio is only significant because of those two points!

```

> chem_pro_final.LM =
+   lm(yield ~ conversion + poly(flow, 3),
+       data = chem_pro.df[-c(6, 16), ])
> summary(chem_pro_final.LM)

Call:
lm(formula = yield ~ conversion + poly(flow, 3), data = chem_pro.df[-c(6, 16), ])

Residuals:
    Min      1Q  Median      3Q      Max 
-4.1800 -1.4444  0.2131  1.7172  4.0394 

Coefficients:
              Estimate Std. Error t value Pr(>t)
(Intercept)      98.4299   9.2406  10.652 7.98e-13 ***
conversion       -4.0787   0.7968  -5.119 9.77e-06 ***
poly(flow, 3)1   -7.9682   2.1429  -3.718 0.000662 ***
poly(flow, 3)2    1.5336   2.5758   0.595 0.555213
poly(flow, 3)3    8.5939   2.2138   3.882 0.000412 ***
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?. 0.1 ? ? 1

Residual standard error: 2.141 on 37 degrees of freedom
Multiple R-squared:  0.668,    Adjusted R-squared:  0.6322
F-statistic: 18.61 on 4 and 37 DF,  p-value: 1.844e-08

```

---

### 3.7 Heteroskedasticity

- So far, when we have evidence against the assumption of equal variance,
- 2. The errors have zero mean and constant variance

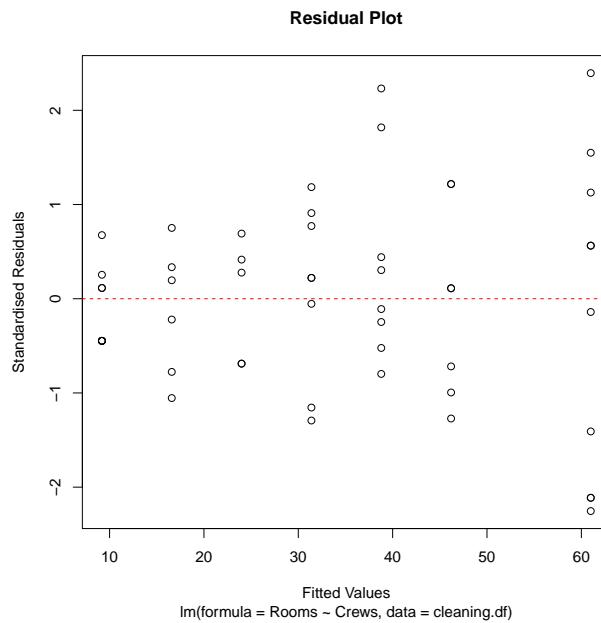
$$\mathbb{E}[\varepsilon_i | \mathbf{X}_i] = 0 \quad \text{and} \quad \text{Var}[\varepsilon_i | \mathbf{X}_i] = \sigma^2 \quad \text{where} \quad \varepsilon_i = Y_i - \mathbb{E}[Y_i | \mathbf{X}_i]$$

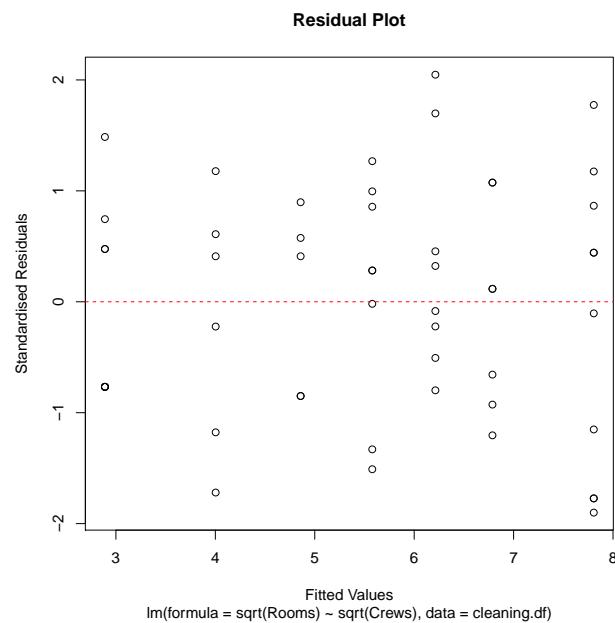
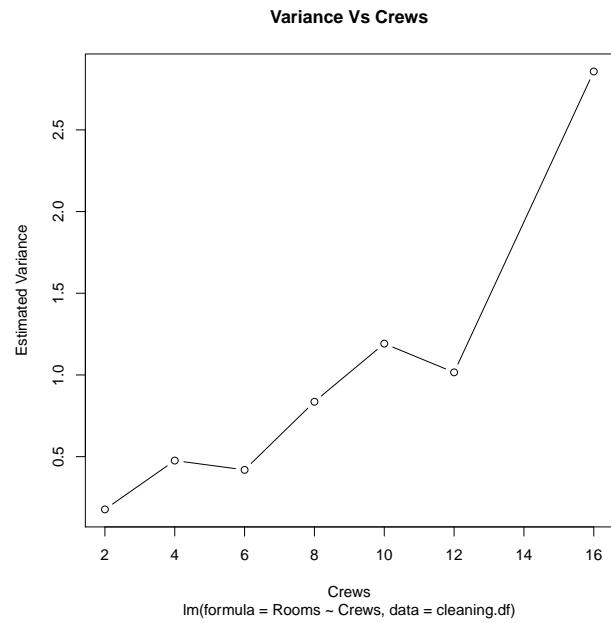
we consider transforming the data. For example, recall the following dataset

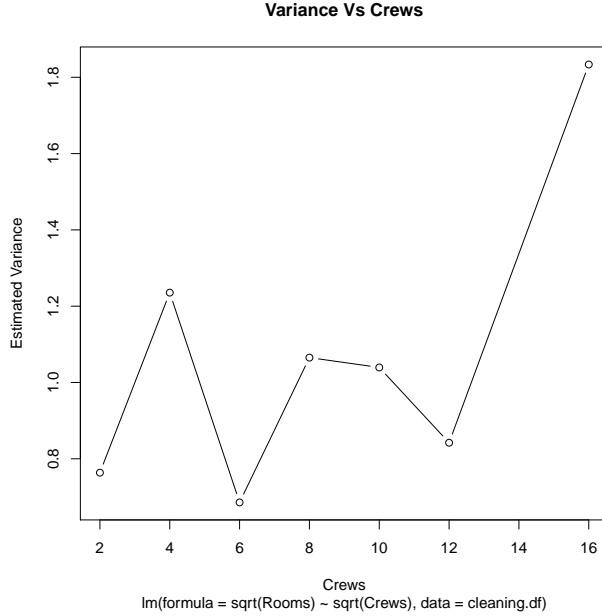
Crews    Number of works on the job  
 Rooms    Number of offices need to be cleaned

which is about 53 cases that a maintenance company did in the past.

```
> cleaning.df = read.table("~/Desktop/cleaning.txt",
+                               header = TRUE)
>
> cleaning.LM = lm(Rooms ~ Crews, data = cleaning.df)
```







- In the presence of heteroskedasticity, the estimator

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

is still unbiased and consistent, but is no longer efficient.

- However, the variance of the estimator loses the consistency property as well

$$\begin{aligned} \hat{\text{Var}}[\hat{\beta} | \mathbf{X}] &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \hat{\text{Var}}[\boldsymbol{\varepsilon} | \mathbf{X}] \left( (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \right)^T \\ &= \hat{\sigma}_u^2 (\mathbf{X}^T \mathbf{X})^{-1} \quad \text{where } \hat{\sigma}_u^2 = \frac{1}{n - k - 1} \hat{\mathbf{e}}^T \hat{\mathbf{e}} \end{aligned}$$

- This is particularly problematic if the purpose of the model is to explain since

$$t_j = \frac{\hat{\beta}_j}{\text{SE}(\hat{\beta}_j)}$$

where  $\text{SE}(\hat{\beta}_j)$  is the  $j$ th main diagonal element of  $\hat{\text{Var}}[\hat{\beta} | \mathbf{X}]$ .

- Recall we have the following under equal variance

$$Y_i = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_k X_{ik} + \varepsilon_i \quad \text{where } \varepsilon_i \sim N(0, \sigma^2)$$

and the estimate  $\hat{\beta} = \mathbf{b}$  is found by minimising

$$f(\mathbf{b}) = \hat{\mathbf{e}}^T \hat{\mathbf{e}} = \sum_{i=1}^n (y_i - b_0 - b_1 x_{i1} - \cdots - b_k x_{ik})^2$$

which is given by founding the gradient of the following and setting it to  $\mathbf{0}$

$$f(\mathbf{b}) = (\mathbf{y} - \mathbf{X}\mathbf{b})^T (\mathbf{y} - \mathbf{X}\mathbf{b}) = \mathbf{y}^T \mathbf{y} - 2\mathbf{b}^T \mathbf{X}^T \mathbf{y} + \mathbf{b}^T \mathbf{X}^T \mathbf{X} \mathbf{b}$$

$$\implies \hat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

- We could have written the equation differently without affecting  $\hat{\boldsymbol{\beta}} = \mathbf{b}$

$$Y_i = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_k X_{ik} + \sigma \varepsilon_i \quad \text{where } \varepsilon_i \sim N(0, 1)$$

- Now with heteroskedasticity, we have

$$Y_i = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_k X_{ik} + \sigma_i \varepsilon_i \quad \text{where } \varepsilon_i \sim N(0, 1)$$

- If we scale our variables according to the true values of  $\sigma_i$ , we have

$$\frac{Y_i}{\sigma_i} = \beta_0 \frac{1}{\sigma_i} + \beta_1 \frac{X_{i1}}{\sigma_i} + \cdots + \beta_k \frac{X_{ik}}{\sigma_i} + \varepsilon_i$$

$$Y_i^* = \beta_0 \frac{1}{\sigma_i} + \beta_1 X_{i1}^* + \cdots + \beta_k X_{ik}^* + 1 \varepsilon_i \quad \text{where } \varepsilon_i \sim N(0, 1)$$

from which we see all the nice properties will be back if  $y_i^*$  and  $x_i^*$  are used

$$\hat{\mathbf{e}}_w^T \hat{\mathbf{e}}_w = \sum_{i=1}^n (y_i^* - b_0/\sigma_i - b_1 x_{i1}^* - \cdots - b_k x_{ik}^*)^2$$

$$= \sum_{i=1}^n \frac{1}{\sigma_i^2} (y_i - b_0 - b_1 x_{i1} - \cdots - b_k x_{ik})^2 = \hat{\mathbf{e}}^T \mathbf{W} \hat{\mathbf{e}}$$

where  $\mathbf{W}$  denote the diagonal matrix containing the scaling factor  $\frac{1}{\sigma_i^2}$ .

- In terms of original response, the objective function now is given by

$$f(\mathbf{b}) = \hat{\mathbf{e}}_w^T \hat{\mathbf{e}}_w = \hat{\mathbf{e}}^T \mathbf{W} \hat{\mathbf{e}} = (\mathbf{y} - \mathbf{X}\mathbf{b})^T \mathbf{W} (\mathbf{y} - \mathbf{X}\mathbf{b})$$

$$= \mathbf{y}^T \mathbf{W} \mathbf{y} - 2\mathbf{b}^T \mathbf{X}^T \mathbf{W} \mathbf{y} + \mathbf{b}^T \mathbf{X}^T \mathbf{W} \mathbf{X} \mathbf{b}$$

- Differentiating with respect to  $\mathbf{b}$ , we have the following gradient,

$$\nabla f = -2\mathbf{X}^T \mathbf{W} \mathbf{y} + 2\mathbf{X}^T \mathbf{W} \mathbf{X} \mathbf{b}$$

- Hence the following estimator will have the same nice properties as before

$$\hat{\boldsymbol{\beta}}_w = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{y}$$

which is known as the [weighted least squares estimator](#).

- Of course, the true  $\sigma_i^2$  are unknown in practice, thus we have estimate

$$\mathbf{W}$$

Q: What is the intuitive reason behind the weighted least squares estimator?

- For the maintenance company data,

```
> crews.group = factor(cleaning.df$Crews)
>
> tapply(cleaning.df$Rooms, crews.group, length)

 2   4   6   8  10  12  16
 9   6   5   8   8   7  10
```

which has many repeated observations, one clearly choice of estimator for

$$\mathbf{W} = \text{diag}\left(\frac{1}{\sigma_1^2}, \frac{1}{\sigma_2^2}, \dots, \frac{1}{\sigma_i^2}, \dots, \frac{1}{\sigma_n^2}\right)$$

is based on the conditional sample variance of the response, that is,

```
> convar = tapply(cleaning.df$Rooms, crews.group, var)
> convar
```

|         |          |          |          |          |          |           |
|---------|----------|----------|----------|----------|----------|-----------|
| 2       | 4        | 6        | 8        | 10       | 12       | 16        |
| 9.00000 | 24.66667 | 22.00000 | 44.12500 | 62.83929 | 53.14286 | 144.01111 |

```
> # vector of corresponding conditional variance
> v.vec = convar[crews.group]

> w.vec = 1/v.vec # vector of diagonal elements of w

> cleaning.WLS = lm(Rooms~Crews, weights = w.vec,
+                      data = cleaning.df)
```

- The above syntax leads to the desired estimate of

$$\hat{\beta}_w = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{y}$$

and as before the fitted value is given by

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

- However, the residual analysis should be done using the following residual

$$\hat{\mathbf{e}}_w = \mathbf{W}^{1/2} \hat{\mathbf{e}} \quad \text{where} \quad \hat{\mathbf{e}} = \mathbf{y} - \hat{\mathbf{y}}$$

- It is the residual  $\hat{\mathbf{e}}_w$ , not  $\hat{\mathbf{e}}$ , that should be used as the estimate of the error

$$\varepsilon$$

that satisfies the model assumptions, i.e.

$$Y_i = \mathbb{E}[Y_i | \mathbf{X}_i] + \sigma_i \varepsilon_i \quad \text{where} \quad \varepsilon_i \sim N(0, 1)$$

$$\implies \varepsilon_i = \frac{Y_i - \mathbb{E}[Y_i | \mathbf{X}_i]}{\sigma_i}$$

- In terms of our current model, we obtain  $\hat{e}_w$  by running the following

```
> model = cleaning.WLS
>
> res = residuals(model)
>
> resw = sqrt(w.vec) * res
```

- Using conditional sample variance to estimate  $\sigma_i^2$ , thus the weight matrix

**W**

is common but could fail badly when the groups of similar or identical points

```
> tapply(cleaning.df$Rooms, crews.group, length)
```

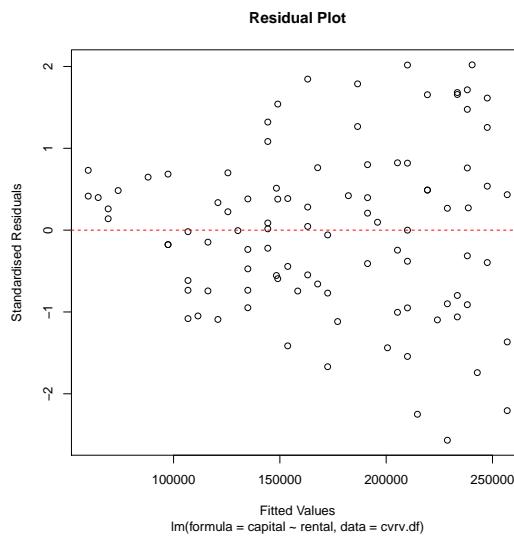
|   |   |   |   |    |    |    |
|---|---|---|---|----|----|----|
| 2 | 4 | 6 | 8 | 10 | 12 | 16 |
| 9 | 6 | 5 | 8 | 8  | 7  | 10 |

are too small, say less than 10. So it actually not reliable for this dataset.

- This approach is certainly not reliable for the following dataset

```
> cvrv.df =
+   read.table("~/Desktop/crvv.csv",
+             sep = ",", header = TRUE)
>
> cvrv.LM = lm(capital~rental, data = cvrv.df)
```

for which the variance is clearly unequal, and **rental** varies continuously.



- Notice the following equality

$$\begin{aligned}\sigma_i^2 &= \text{Var} [\varepsilon_i | \mathbf{X}] = \mathbb{E} [(\varepsilon_i - \mathbb{E} [\varepsilon_i | \mathbf{X}])^2 | \mathbf{X}] \\ &= \mathbb{E} [\varepsilon_i^2 | \mathbf{X}]\end{aligned}$$

- This leads us to the following estimate for  $\sigma_i^2$ ,

$$\hat{\sigma}_i^2 = \mathbb{E} [\hat{\varepsilon}_i^2 | \mathbf{X}]$$

which is a conditional mean of a random variable, that can be observed, and can in turn be estimated by the fitted value of another linear regression.

- Since  $\hat{\varepsilon}_i^2$  might be really small/large in practice, thus people often work with

$$z_i = \ln (\hat{\varepsilon}_i^2) = 2 \ln |\hat{\varepsilon}_i| \implies \hat{\varepsilon}_i^2 = \exp(z_i)$$

the log-scale provides extra numerical stability.

- Thus  $z$  is the auxiliary response,

```
> z = 2 * log(abs(cvr.v.LM$residuals))

> # Perform the auxiliary regression
> auxiliary.LM = lm(z~rental, data = cvrv.df)
```

- Transform back to obtain the estimated  $\sigma_i^2$

```
> v.vec = exp(auxiliary.LM$fitted.values)
>
> w.vec = 1/v.vec
```

- Specify the weights according to the estimated  $\sigma_i^2$

```
> cvrv.WLS = lm(capital~rental, weights = w.vec,
+                 data = cvrv.df)
```

- Of course, this can also be performed for the maintenance dataset, and WLS can be used when we have  $k$  predictors in general.

### 3.8 Correlated Errors

- Consider the following simple example to start our discussion on

lack of independence

- The data is about sales and advertising expenditure

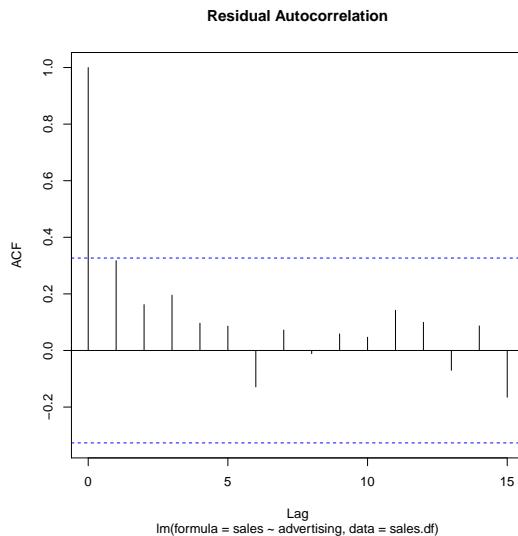
|                          |                                        |
|--------------------------|----------------------------------------|
| <code>sales</code>       | Monthly sales of a retailer            |
| <code>advertising</code> | Amount spent on advertising this month |

```

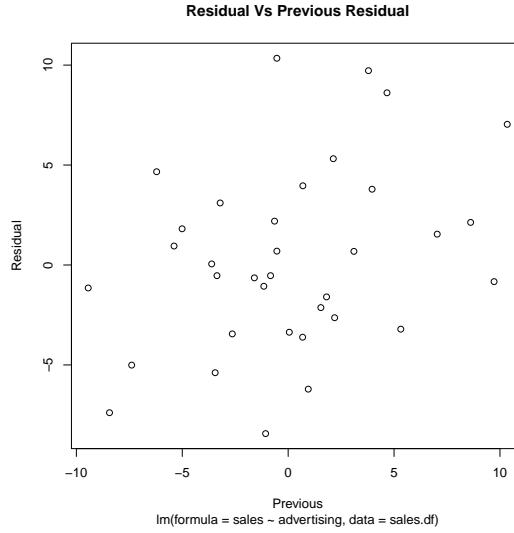
> sales.df =
+   read.table("~/Desktop/sales.txt",
+             sep = "", header = TRUE)
>
> sales.LM = lm(sales~advertising, data = sales.df)

```

- It is expected that advertising might take a while to come in effect.
- Although we don't have clear evidence from acf that the errors are correlated



- Extra care should be taken when data is collected over time.



- For a simple linear regression model,

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

where  $\varepsilon_i$  is independent from  $\varepsilon_j$  for  $i \neq j$ , and independent from  $X_j$  for all  $j$ .

- One of many possibilities that errors can lack independence is the following

$$\mathbb{E}[Y_i | X_1, X_2, \dots, X_n] = \beta_0 + \beta_1 x_i + \beta_2 x_{i-1} + \beta_3 x_{i-2} + \dots$$

which is known as a **distributed lag model** since the following no longer holds

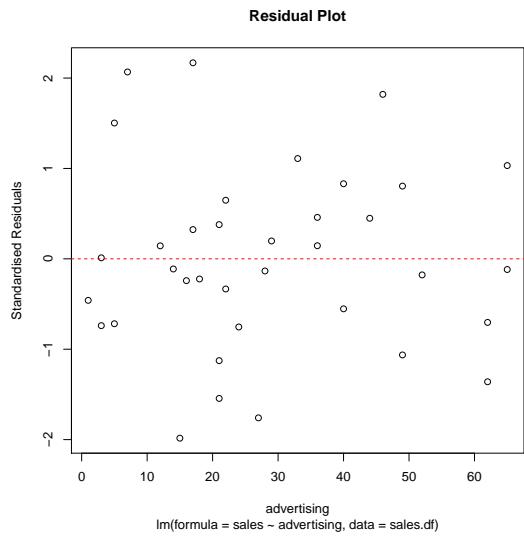
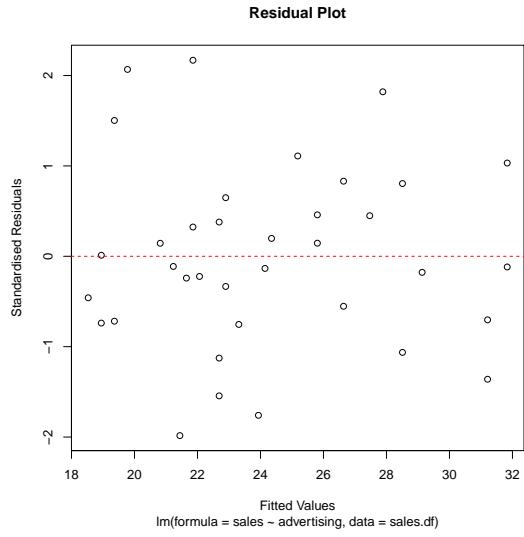
$$\mathbb{E}[Y_i | X_1, X_2, \dots, X_n] = \beta_0 + \beta_1 x_i$$

- Let  $\nu_i$  be independent from  $\nu_j$  for  $i \neq j$ , and independent from  $X_j$  for all  $j$ ,

$$\begin{aligned} Y_i &= \beta_0 + \beta_1 X_i + \beta_2 X_{i-1} + \nu_i \\ &= \beta_0 + \beta_1 X_i + \varepsilon_i \quad \text{where } \varepsilon_i = \beta_2 X_{i-1} + \nu_i \end{aligned}$$

Q: Why is this a problem? How can we detect it? What do we expect to see?

- In addition to the following two residual plots,



- The lagged predictors should be plotted with the standardised residual,

```
> lag = 1 # This might take other natural numbers

> index = 0:(lag-1) - nrow(sales.df)

> x = sales.df$advertising[index]

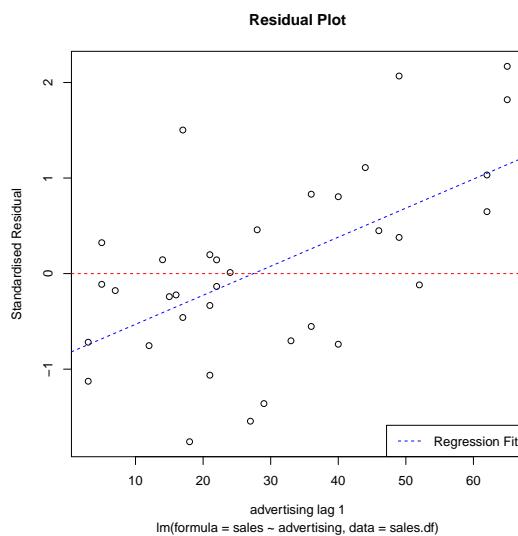
> y = rstandard(sales.LM)[-c(1:lag)]
```

```

> plot(x, y, xlab = bquote("advertising lag"~.(lag)),
+       ylab = "Standardised Residual",
+       main = "Residual Plot",
+       sub = deparse(model$call))
>
> abline(a = 0, b = 0, lty = 2, col = "red")
> abline(lm(y~x), lty = 2, col = "blue")
> legend("bottomright", "Regression Fit",
+         lty = 2, col = 4)

```

- We can expect a nonrandom pattern in the plot below if  $\varepsilon_i$  depends on  $X_{i-1}$



- The plot provides evidence that the error is not independent of the predictor,

$$\begin{aligned}
 Y_i &= \beta_0 + \beta_1 X_i + \beta_2 X_{i-1} + \nu_i \\
 &= \beta_0 + \beta_1 X_i + \varepsilon_i \quad \text{where } \varepsilon_i = \beta_2 X_{i-1} + \nu_i
 \end{aligned}$$

- If the above is a valid model, then having the first order lag of the predictor in the model will satisfy independence assumption.

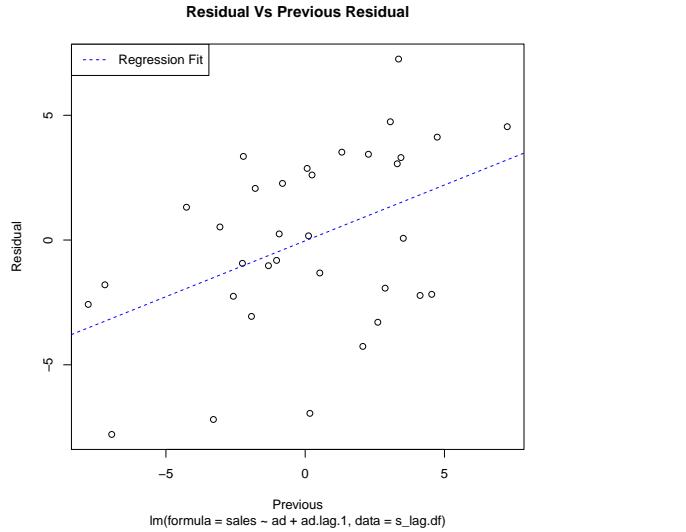
```

> sales = sales.df$sales[-(1:lag)]
>
> ad = sales.df$advertising[-(1:lag)]
>
> ad.lag.1 = sales.df$advertising[index]

> s_lag.df = data.frame(sales = sales,
+                        ad = ad,
+                        ad.lag.1 = ad.lag.1)

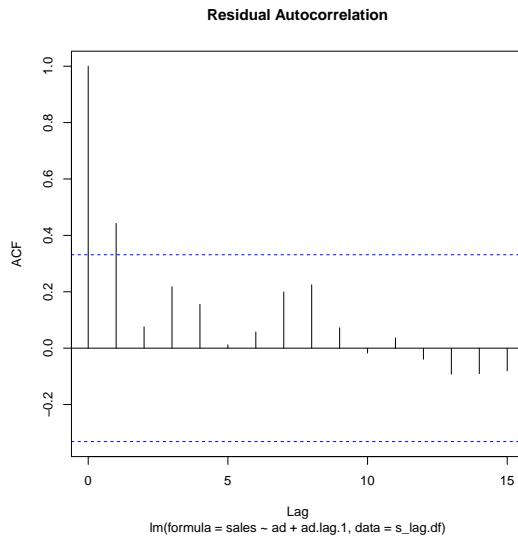
```

- However, it does not solve the lack of independence problem for this dataset.

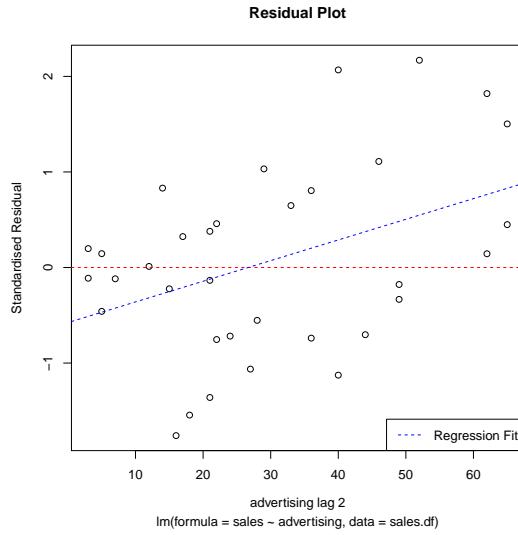


There is a slight but unmistakable linear relationship

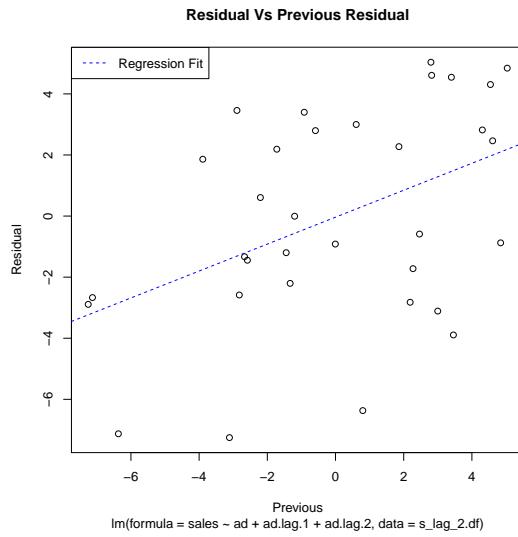
- The linear relationship between residual and previous residual is confirmed by



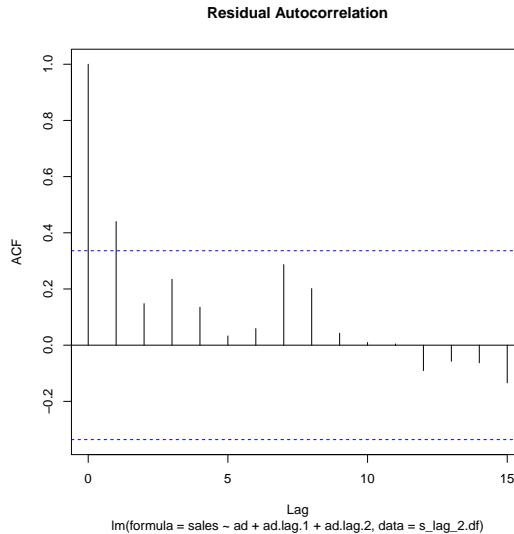
- Looking at the second order lagged values,  $X_{i-2}$ , you might be tempted to...



- However, including higher order lagged terms is not helping at all!



- The slight but unmistakable linear trend still presents.



- Another possible and simple way that error can lack independence is having

$$\varepsilon_i = \rho \varepsilon_{i-1} + \nu_i$$

where  $\nu_i$  are independent and identically distributed for all  $i$

$$\nu_i \sim N(0, \sigma^2)$$

- This structure in errors is called [first-order autoregressive process or AR\(1\)](#).

Q: How can detect the presence of such structure?

```
> s_lag.LM$call
```

```
lm(formula = sales ~ ad + ad.lag.1, data = s_lag.df)
```

```
> res = residuals(s_lag.LM)
>
> cor(res[-1], res[-length(res)])
```

```
[1] 0.4450734
```

- Of course, we have seen this value

```
> cor(res[-1], res[-length(res)])
```

```
[1] 0.4450734
```

Q: However, can this value be due to chance while the true  $\rho$  is zero?

$$\varepsilon_i = \rho \varepsilon_{i-1} + \nu_i$$

- There are 3 tools, the 1st you know, using which depends on the data size.

```
> res.lag.df = data.frame(x = res[-length(res)],
+                           y = res[-1])

> auxiliary.LM = lm(y~x, data = res.lag.df)
```

- Of course, we have to ask whether we can trust the output of

```
> summary(auxiliary.LM)
```

- It can be shown we have no evidence against the auxiliary model being valid

```
> summary(auxiliary.LM)
```

---

```
Call:
lm(formula = y ~ x, data = res.lag.df)

Residuals:
    Min      1Q  Median      3Q     Max 
-6.9895 -2.0044  0.8178  2.5104  5.7853 

Coefficients:
            Estimate Std. Error t value Pr(>t)
(Intercept) -0.03219   0.56092 -0.057  0.95460
x            0.44763   0.15921   2.812  0.00835 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.27 on 32 degrees of freedom
Multiple R-squared:  0.1981, Adjusted R-squared:  0.173 
F-statistic: 7.905 on 1 and 32 DF,  p-value: 0.00835
```

---

- The data size of our current example is arguably too small,

```
> lmtest::dwtest(s_lag.LM, alternative = "two.sided")
```

---

```
Durbin-Watson test

data: s_lag.LM
DW = 1.1039, p-value = 0.004222
alternative hypothesis: true autocorrelation is not 0
```

---

- All three methods indicate we need AR(1) for our current example.

- In its simplest form, AR(1) is given by the following

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

$$\varepsilon_i = \rho \varepsilon_{i-1} + \nu_i \quad \text{where } \nu_i \sim N(0, \sigma^2)$$

where the errors are usually assumed to be stationary, that is

$$\mathbb{E}[\varepsilon_i] = 0 \quad \text{and} \quad \text{Var}[\varepsilon_i] = \sigma_\varepsilon^2 \quad \text{for all } i$$

under which, it can be shown the correlation coefficient is given by

$$r_{i,i-r} = \text{Corr}[\varepsilon_i, \varepsilon_{i-r}] = \rho^r$$

for  $i = 1, 2, \dots, n$  and  $r = 1, 2, \dots, n-1$ , where  $i > r$ .

Q: What is the distribution of  $\varepsilon_i | \varepsilon_{i-1}$ ?

Q: What is this simple AR(1) different from simple linear regression?

The covariance is given by

$$\begin{aligned}\text{Cov}[\varepsilon_i, \varepsilon_{i-1}] &= \mathbb{E}[\varepsilon_i \varepsilon_{i-1}] - \mathbb{E}[\varepsilon_i] \mathbb{E}[\varepsilon_{i-1}] = \mathbb{E}[(\rho \varepsilon_{i-1} + \nu_i) \varepsilon_{i-1}] \\ &= \rho \mathbb{E}[\varepsilon_{i-1}^2] + \mathbb{E}[\nu_i] \mathbb{E}[\varepsilon_{i-1}] = \rho \sigma_\varepsilon^2\end{aligned}$$

- When there are  $k$  predictors, and  $n$  observations, in general matrix notation,

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

however, instead of assuming that the errors are independent, we assume

$$\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \boldsymbol{\Sigma})$$

where  $\boldsymbol{\Sigma}$  is the covariance matrix, and  $\mathbf{R}$  is the correlation coefficient matrix

$$\boldsymbol{\Sigma} = \sigma_\varepsilon^2 \mathbf{R} = \frac{\sigma^2}{1 - \rho^2} \begin{bmatrix} 1 & \rho & \cdots & \rho^{n-1} \\ \rho & 1 & \cdots & \rho^{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ \rho^{n-1} & \rho^{n-2} & \cdots & 1 \end{bmatrix}$$

- If  $\sigma^2$  and  $\rho$  are given, the estimation problem can be solved similarly to WLS,

$$\hat{\boldsymbol{\beta}}_g = \left( \mathbf{X}^T \boldsymbol{\Sigma}^{-1} \mathbf{X} \right)^{-1} \mathbf{X}^T \boldsymbol{\Sigma}^{-1} \mathbf{y}$$

which is known as [generalised least squares](#) (GLS) estimator of  $\boldsymbol{\beta}$ .

- Combing AR(1) together with our early distributed lag model,

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_{i-1} + \varepsilon_i$$

$$\varepsilon_i = \rho \varepsilon_{i-1} + \nu_i \quad \text{where } \nu_i \sim N(0, \sigma^2)$$

we have the following model,

$$Y_i = \beta_0(1 - \rho) + \beta_1 X_i + (\beta_2 - \rho \beta_1) X_{i-1} - \beta_2 \rho X_{i-2} + \rho Y_{i-1} + \nu_i$$

- When  $\sigma^2$  and  $\rho$  are not given, this is a nonlinear optimisation problem

$$\begin{aligned}\ell(\boldsymbol{\beta}, \rho, \sigma^2; \mathbf{X}^*, \mathbf{y}) &= -\ln(L(\boldsymbol{\beta}, \rho, \sigma^2; \mathbf{X}^*, \mathbf{y})) \\ &= \frac{n}{2} \ln(2\pi) + \frac{1}{2} \ln(\det \boldsymbol{\Sigma}) \\ &\quad + \frac{1}{2} (\mathbf{y} - \mathbf{X}^* \boldsymbol{\beta})^T \boldsymbol{\Sigma}^{-1} (\mathbf{y} - \mathbf{X}^* \boldsymbol{\beta})\end{aligned}$$

which has too be solved numerically.

- R can construct AR models, and solve the optimisation without the details

```
> with(s_lag.df,
+       {s_lag.AR1 = arima(
+         sales, order = c(1, 0, 0),
+         xreg = cbind(ad, ad.lag.1))
+       })
> s_lag.AR1
```

```

Call:
arima(x = sales, order = c(1, 0, 0), xreg = cbind(ad, ad.lag.1))

Coefficients:
            ar1  intercept      ad  ad.lag.1
            0.4966    16.9080   0.1218   0.1391
s.e.    0.1580     1.6716   0.0308   0.0316

sigma^2 estimated as 9.476:  log likelihood = -89.16,  aic = 188.32

```

> s\_lag.LM

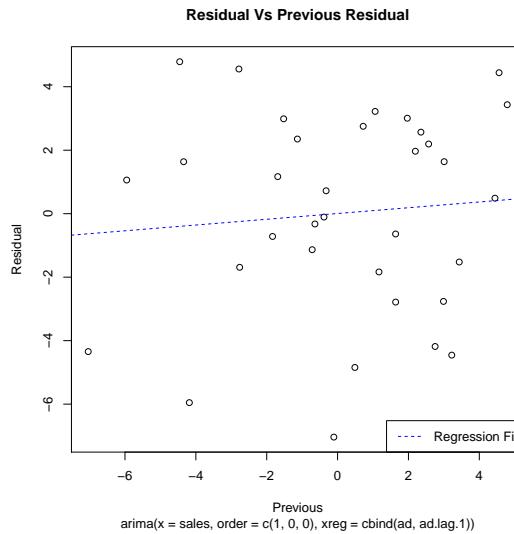
```

Call:
lm(formula = sales ~ ad + ad.lag.1, data = s_lag.df)

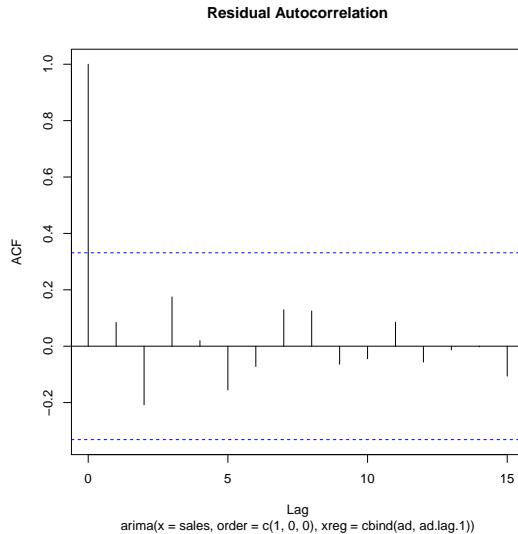
Coefficients:
(Intercept)      ad      ad.lag.1
15.6036       0.1424     0.1665

```

- It seems the new error  $\nu_i$  are independent and identically distributed,



- It seems the new error  $\nu_i$  are independent and identically distributed,



- The rest of assumptions seem to be fine, however, the model equation is

$$Y_i = \beta_0(1 - \rho) + \beta_1 X_i + (\beta_2 - \rho\beta_1)X_{i-1} - \beta_2\rho X_{i-2} + \rho Y_{i-1} + \nu_i$$

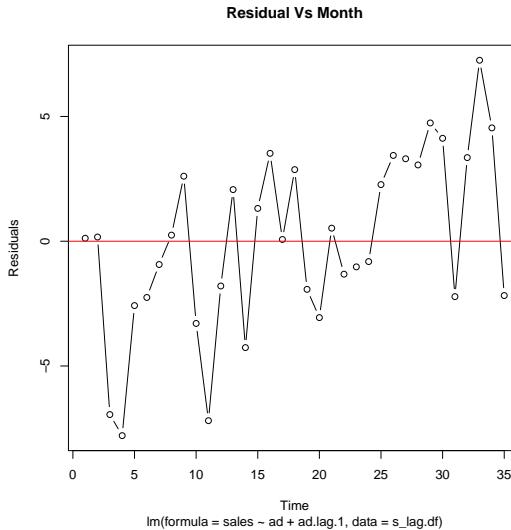
more advanced models are harder to interpret, should be avoided if possible.

- In this case, there is actually a simpler model that is reasonable good.
- Consider the following model again

```
> sales_pre.LM = lm(sales ~ ad + ad1, data = s_pre.df)
>
> res = sales_pre.LM$residuals

> plot(res, type = "b",
+       xlab = "Time", ylab = "Residuals",
+       main = "Residual Vs Month")
>
> abline(h = 0, col = "red")
```

- Notice there seems to be an weak increasing trend



- It is very likely that the data is recorded/sorted according to time, the plot suggests that time might help to explain the error, thus the sales number.

```
> s_lag_m.df = cbind(s_lag.df, m = 1:nrow(s_lag.df))

> s_f.LM = lm(sales~ad+ad.lag.1+m, data = s_lag_m.df)

> summary(s_f.LM) # A peek before checking residuals
```

```
Call:
lm(formula = sales ~ ad + ad.lag.1 + m, data = s_lag_m.df)

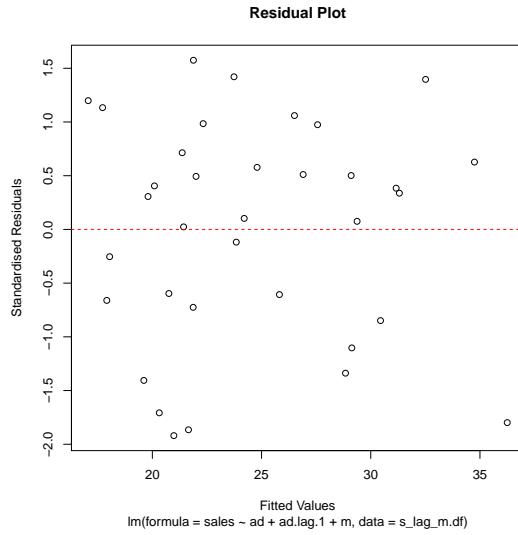
Residuals:
    Min      1Q  Median      3Q     Max 
-5.4840 -2.0409  0.8971  1.9423  4.5192 

Coefficients:
            Estimate Std. Error t value Pr(>t)
(Intercept) 12.03345   1.48653   8.095 3.84e-09 ***
ad          0.15308   0.02984   5.129 1.48e-05 ***
ad.lag.1    0.15914   0.03052   5.214 1.16e-05 ***
m           0.19323   0.05189   3.724 0.000781 ***

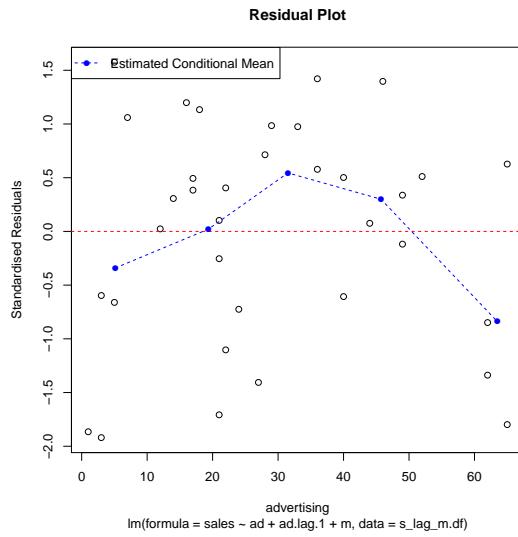
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.084 on 31 degrees of freedom
Multiple R-squared:  0.7507,    Adjusted R-squared:  0.7266 
F-statistic: 31.12 on 3 and 31 DF,  p-value: 1.769e-09
```

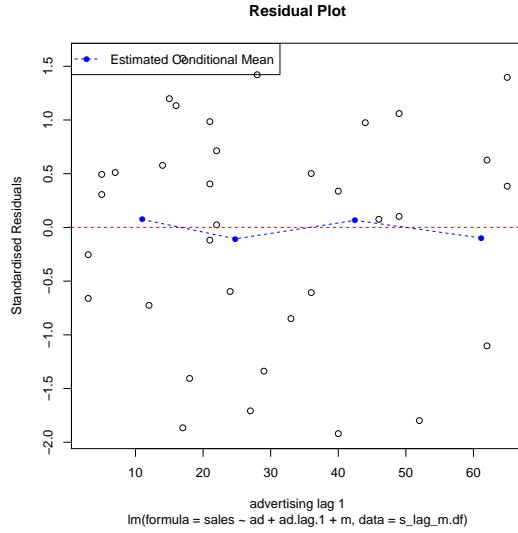
- It seems time, m, is highly significant, but we have to check residuals first,



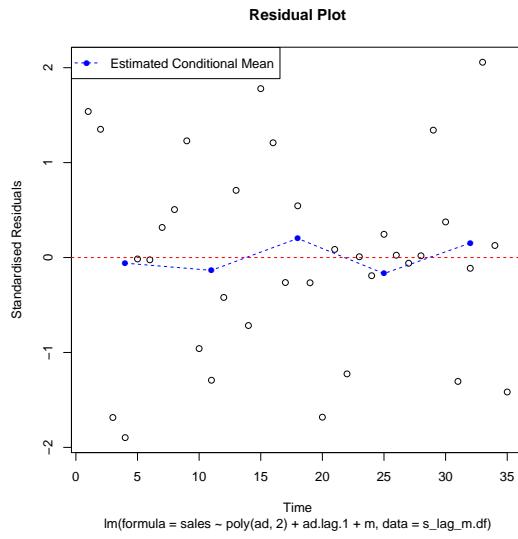
- It seems that we need a polynomial term for `ad`,



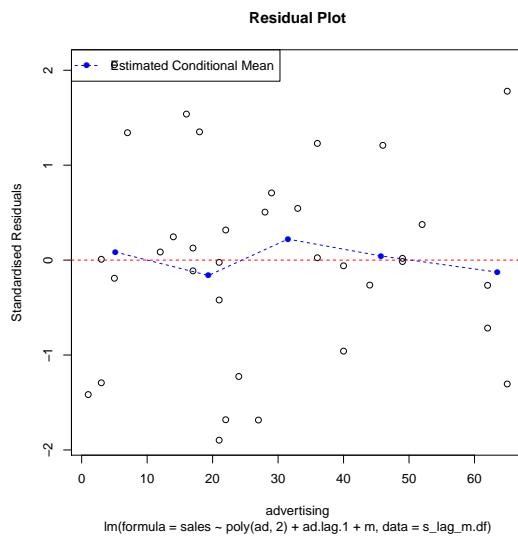
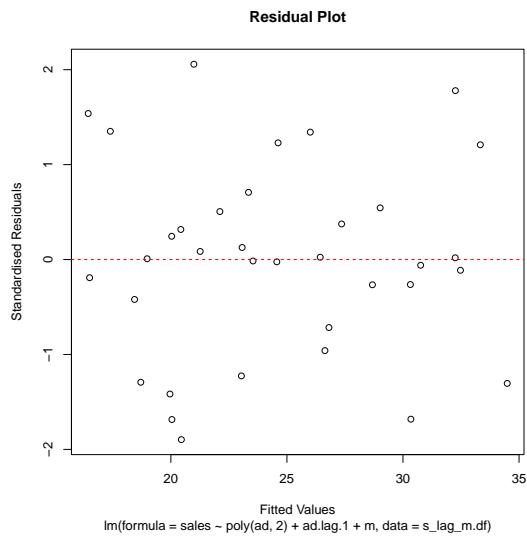
- The lag 1 term seems to be OK.

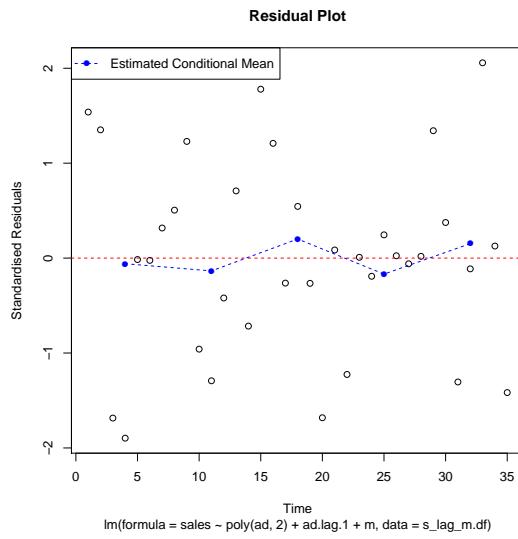
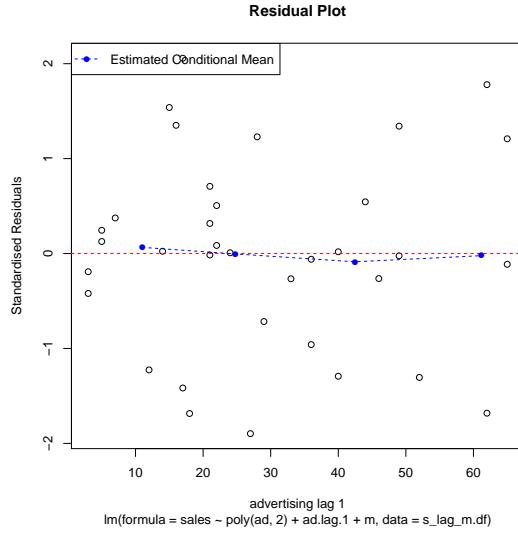


- We also have no evidence against time being linear.

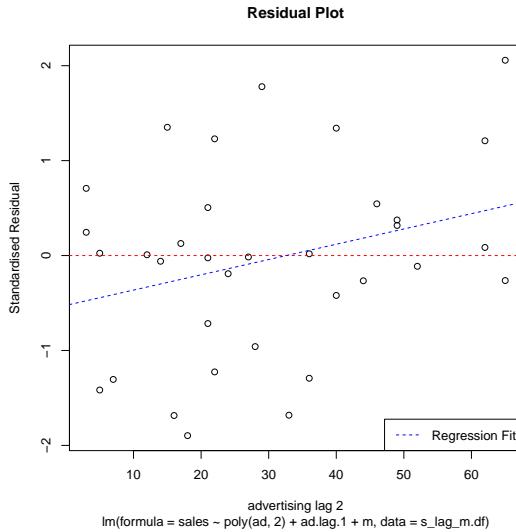


- We have no evidence against linearity from the next 4 plots.





- And we have no evidence that errors depend on higher order lagged `ad`



- There are no evidence against the following auxiliary regression being valid

```

> lag = 1
> index = 0:(lag-1) - nrow(s_lag_m.df)
>
> x = s_lag_m.df$ad.lag.1[index]
> y = rstandard(sales_final.LM)[-c(1:lag)]
>
> auxiliary.LM = lm(y~x)

> summary(auxiliary.LM)

```

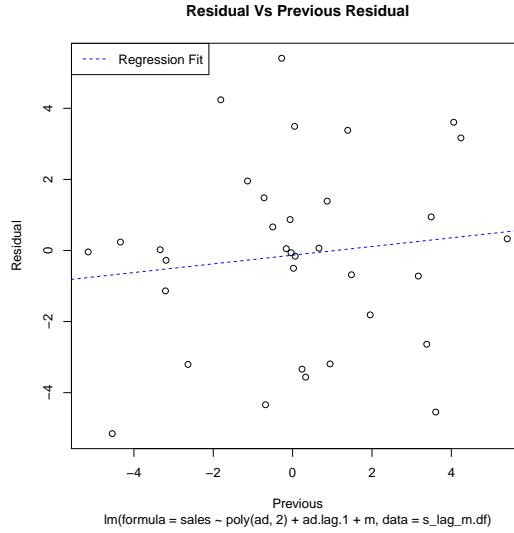
| Coefficients: | Estimate  | Std. Error | t value | Pr(>t)   |
|---------------|-----------|------------|---------|----------|
| (Intercept)   | -0.524933 | 0.317745   | -1.652  | 0.1083   |
| x             | 0.016100  | 0.009111   | 1.767   | 0.0868 . |

Signif. codes: 0 \*\*\* 0.001 \*\* 0.01 \* 0.05 . 0.1 ? ? 1

Residual standard error: 0.9731 on 32 degrees of freedom  
Multiple R-squared: 0.0889, Adjusted R-squared: 0.06043  
F-statistic: 3.123 on 1 and 32 DF, p-value: 0.08675

based on the  $t$ -test, we do not pursue a higher oder distributed lag model.

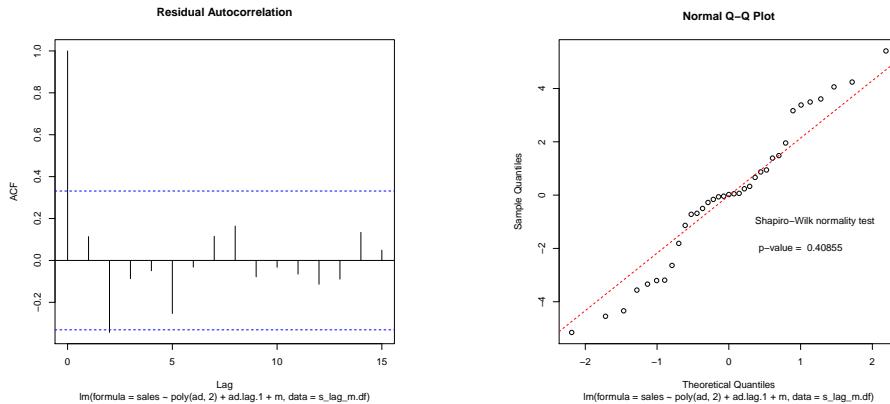
- We have no evidence of more first order autocorrelation,



```
> lmtest::dwtest(s_f.LM, alternative = "two.sided")
```

```
Durbin-Watson test
data: s_f.LM
DW = 1.6193, p-value = 0.1615
alternative hypothesis: true autocorrelation is not 0
```

- However, the fix for independence might not be as good as the AR(1) model.



- However, given we have only a relatively small number of data points, and we probably should not push for AR(2) without a significant AR(1) term.
- Since the normality assumption seems to be reasonable as well, we proceed

```
> summary(sales_final.LM)
```

```

Call:
lm(formula = sales ~ poly(ad, 2) + ad.lag.1 + m, data = s_lag_m.df)

Residuals:
    Min      1Q  Median      3Q     Max 
-5.1523 -1.4739  0.0221  1.4357  5.4090 

Coefficients:
            Estimate Std. Error t value Pr(>t)
(Intercept) 15.40536   1.32144 11.658 1.15e-12 ***
poly(ad, 2)1 16.90116   3.04137  5.557 4.83e-06 ***
poly(ad, 2)2 -7.75778   3.09681 -2.505  0.0179 *  
ad.lag.1     -0.16483   0.02831  5.823 2.29e-06 ***
m            0.24254   0.05185  4.678 5.77e-05 *** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.851 on 30 degrees of freedom
Multiple R-squared:  0.7939, Adjusted R-squared:  0.7664 
F-statistic: 28.88 on 4 and 30 DF,  p-value: 6.653e-10

```

Q: If you are the manager, what conclusions can you draw from this model?

### 3.9 Variable Selection

- So far we have only a handful of regressors/predictors/explanatory variables.

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_k X_{ik} + \varepsilon_i$$

that is, the above **full model** has a relatively small  $k$  value.

- Variable selection aims to choose a subset of  $p$  regressors that is

“best”

in some sense out of  $k$  regressors, it often deals with a reasonably large  $k$

$$5 \leq k \leq 30$$

- Notice there are various ways to define the “best” model.
- Often the following criteria are used **wrongly**!

$$\text{RSS}, \quad R^2, \quad R_{\text{adj}}^2, \quad \text{Mallow's } C_p, \quad t\text{-test}, \quad \text{partial } F\text{-test}, \quad \text{AIC}, \quad \text{BIC}$$

Q: What can go wrong with variable selection?

- If we introduce too few, we are **underfitting**, and if we introduce too many, we are **overfitting**, both of which lead to unpleasant consequences.
- The type of consequences that we would try to avoid depends on the reason for building a regression model, there are two purposes for building a model:

1. To explain
2. To predict

Q: Why is it unlikely to find a model that fulfil both purposes?

- There is a bias-variance trade-off,

$$\mathbb{E} \left[ \underbrace{\left( Y_{n+1} - \hat{Y}_{n+1}^* \right)^2}_{\text{mean squared error}} \mid \cdot \right] = \sigma^2 + \underbrace{\text{Var} \left[ \hat{Y}_{n+1}^* \mid \cdot \right]}_{\text{variance}} + \underbrace{\mathbb{E} \left[ \hat{Y}_{n+1}^* - \mu_{n+1} \mid \cdot \right]^2}_{\text{bias}}$$

however, the trade-off is not necessarily one-for-one.

- In both cases, we would like to select a model that has a small MSE.

Q: How can we get some idea on MSE?

- **Data splitting**

If we have a lot of data, we can divide the data into two parts, the **training set** and the **test set**. We use the training set to build models and test set to estimate the MSE of prediction.

- **Cross-validation**

If there is not sufficient data, we split the data into several parts. Treating one of them as the test set and estimate MSE, and the rest as the training set. Repeat by treating each of other parts as the test set and the rest as the training set. Take the mean of all of those estimates.

- **Bootstrap**

Sample the data with replacement to create a training set, and use the original data set as the test set and estimate the MSE. Repeat this many times and take the mean of all of those estimates.

- The last two are computationally intense, thus were avoided in the old days.
- So when we have the “luxury”, use the last three. However, traditionally,
- **Residual sum of squares**

$$\begin{aligned} \text{RSS} &= (n - p - 1) \sigma_u^2 = \hat{\mathbf{e}}^T \hat{\mathbf{e}} = \mathbf{y}^T (\mathbf{I} - \mathbf{P}) \mathbf{y} \\ &= \mathbf{y}^T \left( \mathbf{I} - \mathbf{X} \left( \mathbf{X}^T \mathbf{X} \right)^{-1} \mathbf{X}^T \right) \mathbf{y} \end{aligned}$$

This simply uses the residual sum of squares as the estimate for MSE. This is usually too optimistic and underestimates the prediction error.

- **Coefficient of determination**

$$R^2 = \frac{\text{ESS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}}$$

where ESS is the **explained sum of squares** and TSS the **total sum of squares**

$$\text{TSS} = \sum_{i=1}^n (y_i - \bar{y})^2 = \left( \mathbf{y} - \frac{\mathbf{1}^T \mathbf{y} \mathbf{1}}{n} \right)^T \left( \mathbf{y} - \frac{\mathbf{1}^T \mathbf{y} \mathbf{1}}{n} \right)$$

This is only valid if the two models have the same number of regressors.

- **Adjusted coefficient of determination**

$$R_{\text{adj}}^2 = 1 - \frac{\text{RSS}/(n - p - 1)}{\text{TSS}/(n - 1)}$$

This can be argued to correct the bias in  $R^2$  and can be used when the two models have different number of regressors. Notice this equivalent to using

$$\sigma_u^2 = \frac{1}{n - p - 1} \hat{\mathbf{e}}^T \hat{\mathbf{e}}$$

which tends to select a model that suffers from over-fitting.

- Mallow's  $C_p$

$$C_p = \frac{\text{RSS}_p}{\hat{\sigma}_{ku}^2} - n + 2p$$

Before cross-validation/ bootstrap was realistic, it was used as an estimate of MSE and address the issue of overfitting without regression assumptions.

- **t-test**

This removes 1 regressor at a time from the model according the  $p$ -value.

- **Partial F-test**

This checks the adequacy of the submodel according to the  $p$ -value.

- **Akaike information criterion**

$$\text{AIC} = 2p^* - 2 \ln \left( L \left( \hat{\beta}, \hat{\sigma}^2; \mathbf{y}, \mathbf{X} \right) \right)$$

where  $\hat{\beta}$  and  $\hat{\sigma}^2$  denote the maximum likelihood estimates of  $\beta$  and  $\sigma^2$ , and

$$p^* = p + 2$$

is the number of parameters estimated in the model.

Q: Can you work out whether a better model is indicated by a small or big AIC?

- AIC is based on Kullback-Leibler divergence,

$$I(f, g) = \int f(z) \ln \left( \frac{f(z)}{g(z; \theta)} \right) dz$$

which is a distance measure between two distributions.

- In some sense, it measures the amount information lost when distribution

$$g(z; \theta)$$

is used to model distribution

$$f(z)$$

- Since  $f(z)$  is a density function, we have

$$\begin{aligned} I(f, g) &= \int f(z) \ln(f(z)) dz - \int f(z) \ln(g(z; \theta)) dz \\ &= \mathbb{E} [\ln(f(z))] - \mathbb{E} [\ln(g(z; \theta))] \end{aligned}$$

- Akaike found the likelihood function  $L$  of the model can be used to estimate

$$\mathbb{E} [\ln(g(z; \theta))]$$

and the bias of using  $\ln(L(\hat{\beta}, \hat{\sigma}^2; \mathbf{y}, \mathbf{X}))$  is approximately  $p^*$ , thus

$$\hat{I}(f, g) = \mathbb{E} [\ln(f(z))] - \left( \ln(L(\hat{\beta}, \hat{\sigma}^2; \mathbf{y}, \mathbf{X})) - p^* \right)$$

is approximately unbiased.

- Since the first expectation is a constant with respect to model selection,

$$AIC = 2p^* - 2 \ln \left( L(\hat{\beta}, \hat{\sigma}^2; \mathbf{y}, \mathbf{X}) \right)$$

along with the multiple of 2 was proposed by Akaike historically.

- Bayesian information criterion

$$BIC = p^* \ln n - 2 \ln \left( L(\hat{\beta}, \hat{\sigma}^2; \mathbf{y}, \mathbf{X}) \right)$$

Similar to AIC, but penalises the number of parameters more severely, since  $n$  is the number of observations. Having a small value of BIC means better.

- For variable/model selection purposes, there is no clear choice between

AIC and BIC.

- BIC is asymptotically consistent as a selection criterion, that is, given a set of models that include the true model, the probability of that BIC will select the correct model approaches 1 as  $n \rightarrow \infty$ , which is not the case for AIC, which tends to choose models that are too complex as  $n \rightarrow \infty$ .
- However, for finite samples, BIC often chooses models that are too simple, because of the heavy penalty on complexity.

Q: So what should we use for variable selection? Do you notice any problem?

- If there are  $k$  regressors, there are

$$2^k$$

possible multiple regression models in total if assume the intercept is needed.

- Traditionally, the following procedures were popular

Forward selection, Backward elimination, and Stepwise regression

- However, an efficient algorithm (the leaps and bounds procedure) together with modern computing power, means an exhaustive approach is possible.
- If an exhaustive approach cannot find all possible regressions efficiently, we really need to question whether we want to do what we do in the first place.
- Best subset regression finds for each

$$p \in \{0, 1, 2, \dots, k\}$$

the subset of size  $p$  according to a certain selection criterion.

- The following is a slightly bigger dataset,

```
> evap.df = read.table("~/Desktop/evap.txt",
+ header = TRUE)
```

```
'data.frame': 46 obs. of 11 variables:
$ avst : int 84 84 79 81 84 74 73 75 84 86 ...
$ minst: int 65 65 66 67 68 66 66 67 68 72 ...
$ maxst: int 147 149 142 147 167 131 131 134 161 169 ...
$ avat : int 85 86 83 83 88 77 78 84 89 91 ...
$ minat: int 59 61 64 65 69 67 69 68 71 76 ...
$ maxat: int 151 159 152 158 180 147 159 159 195 206 ...
$ avh : int 95 94 94 94 93 96 96 95 95 93 ...
$ minh : int 40 28 41 50 46 73 72 70 63 56 ...
$ maxh : int 398 345 388 406 379 478 462 464 430 406 ...
$ wind : int 273 140 318 282 311 446 294 313 455 604 ...
$ evap : int 30 34 33 26 41 4 5 20 31 38 ...
```

the exact nature of study that generated the dataset is not important.

- R has an implementation of the leaps and bounds algorithm

```
> # install.packages("leaps")
> library(leaps)

> regsubsets.out = regsubsets(
+   evap~.,
+   data = evap.df,
+   nbest = 1,          # 1 best model for each p
+   nvmax = NULL,       # NULL for no limit on p
+   method = "exhaustive")
```

- Regressor(s) needed is/are indicated with "\*".

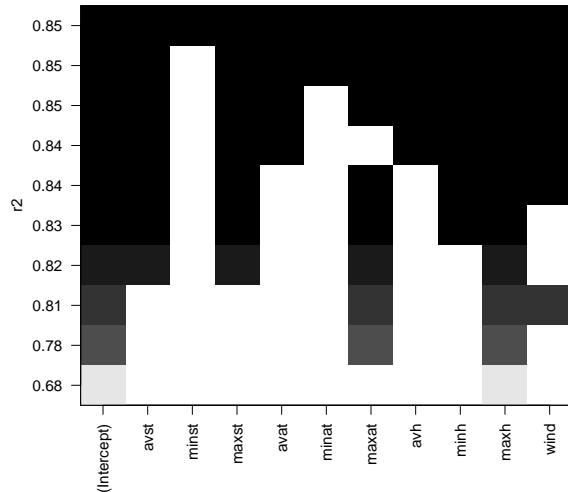
```
> summary(regsubsets.out)
```

|    | Selection Algorithm: | exhaustive | avst | minst | maxst | avat | minat | maxat | avh | minh | maxh | wind |
|----|----------------------|------------|------|-------|-------|------|-------|-------|-----|------|------|------|
| 1  | ( 1 )                | " "        | " "  | " "   | " "   | " "  | " "   | " "   | " * | " *  | " "  | " "  |
| 2  | ( 1 )                | " "        | " "  | " "   | " "   | " "  | " *   | " "   | " * | " *  | " "  | " "  |
| 3  | ( 1 )                | " "        | " "  | " "   | " "   | " "  | " *   | " "   | " " | " *  | " "  | " "  |
| 4  | ( 1 )                | " *"       | " "  | " *"  | " "   | " "  | " *   | " "   | " " | " *  | " "  | " "  |
| 5  | ( 1 )                | " *        | " "  | " *   | " "   | " "  | " *   | " "   | " * | " *  | " "  | " "  |
| 6  | ( 1 )                | " *        | " "  | " *   | " "   | " "  | " *   | " "   | " * | " *  | " "  | " *  |
| 7  | ( 1 )                | " *        | " "  | " *   | " *   | " "  | " "   | " *   | " * | " *  | " *  | " *  |
| 8  | ( 1 )                | " *        | " "  | " *   | " *   | " "  | " *   | " *   | " * | " *  | " *  | " *  |
| 9  | ( 1 )                | " *        | " "  | " *   | " *   | " *  | " *   | " *   | " * | " *  | " *  | " *  |
| 10 | ( 1 )                | " *"       | " *  | " *   | " *   | " *  | " *   | " *   | " * | " *  | " *  | " *  |

Q: Why would RSS,  $R^2$ ,  $R_{\text{adj}}^2$ , Mallow's  $C_p$ , AIC and BIC give the same output?

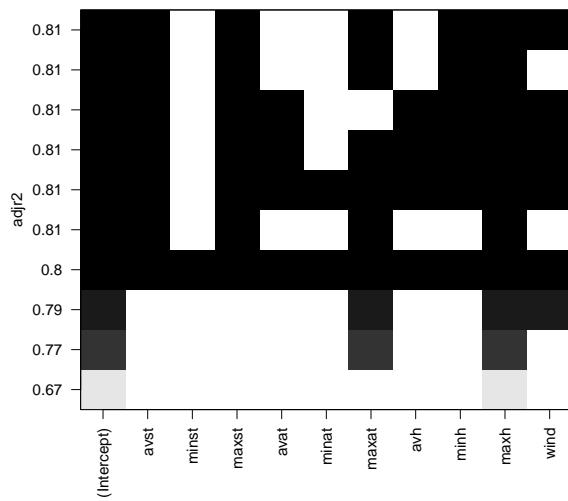
```
> plot(regsubsets.out, scale = "r2", main = "R^2")
```

**R<sup>2</sup>**



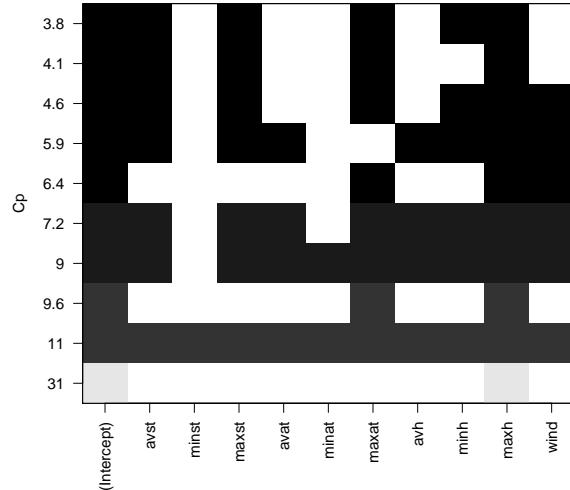
```
> plot(regsubsets.out, scale = "adjr2", main = "Adjusted R^2")
```

**Adjusted R<sup>2</sup>**



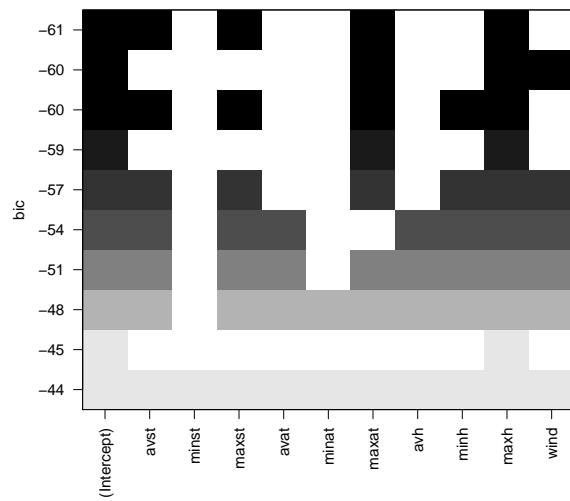
```
> plot(regsubsets.out, scale = "Cp", main = "Mallow's Cp")
```

Mallow's Cp



```
> plot(regsubsets.out, main = "BIC")
```

BIC



- We should also check residuals for each one of those models, then using

$$R_{\text{adj}}^2, \quad \text{Mallow's } C_p, \quad t\text{-test}, \quad \text{partial } F\text{-test}, \quad \text{AIC}, \quad \text{BIC}$$

to pick the “best” model out of the valid models.

- If some transformation on regressors is used, then

$$t\text{-test}, \quad \text{or} \quad \text{partial } F\text{-test},$$

should NOT be used, since they only work on nested models.

- If the response needs to be transformed, we need redo the variable selection!
- The validity of the model is a priority especially when the goal is to explain.
- For predictive models, the modern approach is very different!

1. First of all, we tends to have plenty of data, and powerful computers, so

data splitting, cross-validation or bootstrapping

2. By keeping a subset of the predictors and discarding the rest, it leads us to a model that is interpretable and possibly has a lower MSE than the full model.

- However, it is a discrete process, i.e.

variables are either kept or discarded

so it often does not leads to a smaller MSE than the full model.

- Also for predictive models, we don’t care about interpretability of the model.

Q: Can we drop half of a variable from the full model?

- This can be achieved if we deviate from the traditional least squares estimate

$$\hat{\beta} = \underset{\mathbf{b}}{\operatorname{argmin}} \left\{ (\mathbf{y} - \mathbf{X}\mathbf{b})^T (\mathbf{y} - \mathbf{X}\mathbf{b}) \right\} = \underset{\mathbf{b}}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \left( y_i - \sum_{j=0}^k x_{ij} b_j \right)^2 \right\}$$

which have the smallest variance amongst all linear unbiased estimators.

- But modern approaches argue having unbiased estimators is not necessary!

- Shrinkage methods are more continuous in nature, often lead to estimators that are biased but don’t suffer as much from high variability.

- Ridge regression shrinks the regression coefficients by imposing a penalty

$$\hat{\beta}^{\text{ridge}} = \underset{\mathbf{b}}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \left( y_i - \hat{b}_0 - \sum_{j=1}^k x_{ij}^* b_j \right)^2 + \lambda \sum_{j=1}^k b_j^2 \right\}$$

and effectively “dropping” variables partly to achieve a small MSE.

Q: Why is this penalty of sum of squares of the partial slope parameters useful?

- Here  $\lambda \geq 0$  is a complexity parameter that controls the amount of shrinkage.
- Notice  $\mathbf{X}^*$  is different from the original  $\mathbf{X}$ , where  $\mathbf{X}^*$  has no unit column for the  $b_0$ , and are centred and scaled to have zero mean and unit variance, i.e.

$$x_{ij}^* = \frac{x_{i,j+1} - \bar{x}_{j+1}}{s_{j+1}} \quad \text{for } j = 1, 2, \dots, k$$

Q: Have you seem something like in other courses?

- According to Lagrange's multiplier, the minimisation problem

$$\hat{\beta}^{\text{ridge}} = \underset{\mathbf{b}}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \left( y_i - \hat{b}_0 - \sum_{j=1}^k x_{ij}^* b_j \right)^2 + \lambda \sum_{j=1}^k b_j^2 \right\}$$

can be shown to be equivalent to

$$\begin{aligned} \hat{\beta}^{\text{ridge}} &= \underset{\mathbf{b}}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \left( y_i - \hat{b}_0 - \sum_{j=1}^k x_{ij}^* b_j \right)^2 \right\} \\ \text{subject to } &\sum_{j=1}^k b_j^2 \leq c_\lambda \end{aligned}$$

- Notice the minimisation is done with respect to  $\mathbf{b}$ , which does not include,

$$\hat{b}_0 = \bar{y} = \frac{1}{n} \sum_i^n y_i$$

- Comparing it with least squares problem,

$$\hat{\beta} = \underset{\mathbf{b}}{\operatorname{argmin}} \left\{ (\mathbf{y} - \mathbf{X}\mathbf{b})^T (\mathbf{y} - \mathbf{X}\mathbf{b}) \right\} = \underset{\mathbf{b}}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \left( y_i - \sum_{j=0}^k x_{ij} b_j \right)^2 \right\}$$

and dropping  $\hat{b}_0$  since it does not affect  $\mathbf{b}$ , we have the following

$$\begin{aligned} \hat{\beta}^{\text{ridge}} &= \underset{\mathbf{b}}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \left( y_i - \hat{b}_0 - \sum_{j=1}^k x_{ij}^* b_j \right)^2 + \lambda \sum_{j=1}^k b_j^2 \right\} \\ &= \underset{\mathbf{b}}{\operatorname{argmin}} \left\{ (\mathbf{y} - \mathbf{X}^* \mathbf{b})^T (\mathbf{y} - \mathbf{X}^* \mathbf{b}) + \lambda \mathbf{b}^T \mathbf{b} \right\} \\ \implies \mathbf{0} &= -2\mathbf{X}^{*\top} \mathbf{y} + 2\mathbf{X}^{*\top} \mathbf{X}^* \mathbf{b} + 2\lambda \mathbf{b} \\ \implies \hat{\beta}^{\text{ridge}} &= (\mathbf{X}^{*\top} \mathbf{X}^* + \lambda \mathbf{I})^{-1} \mathbf{X}^{*\top} \mathbf{y} \end{aligned}$$

- We will come back to this later on when we talk about SVD and PCA, and show why  $\hat{\beta}^{\text{ridge}}$  has a smaller variance in return for allowing some bias.

- Another popular shrinkage method in modern era is known as [Lasso](#),

$$\begin{aligned} \hat{\beta}^{\text{lasso}} &= \underset{\mathbf{b}}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \left( y_i - \hat{b}_0 - \sum_{j=1}^k x_{ij}^* b_j \right)^2 \right\} \\ \text{subject to } &\sum_{j=1}^k |b_j| \leq c_\lambda \end{aligned}$$

in Lagrangian form, we have

$$\hat{\beta}^{\text{lasso}} = \underset{\mathbf{b}}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \left( y_i - \hat{b}_0 - \sum_{j=1}^k x_{ij}^* b_j \right)^2 + \lambda \sum_{j=1}^k |b_j| \right\}$$

- It can be shown  $\lambda$  or  $c_\lambda$  in ridge or Lasso affect the MSE and variance, and are chosen to minimise MSE for a given variance in practice.

$$\underbrace{\mathbb{E} \left[ (Y_{n+1} - \hat{Y}_{n+1}^*)^2 \mid \cdot \right]}_{\text{mean squared error}} = \sigma^2 + \underbrace{\text{Var} \left[ \hat{Y}_{n+1}^* \mid \cdot \right]}_{\text{variance}} + \underbrace{\mathbb{E} \left[ (\hat{Y}_{n+1}^* - \mu_{n+1})^2 \mid \cdot \right]}_{\text{bias}}$$

## 4 Nonlinear Regression

- There are many situations where the response and regressors are related through a known **nonlinear function**. It leads to a **nonlinear regression model**.
- We have largely focus on the multiple linear regression model

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + \varepsilon$$

but linear regression models in general include polynomial models and others

$$Y = \beta_0 + \beta_1 Z_1 + \beta_2 Z_2 + \cdots + \beta_k Z_k + \varepsilon$$

where  $Z_j$  represents any function of the regressors  $X_1, X_2, \dots, X_k$ , e.g.

$$X_j^2, \quad \sqrt{X_j}, \quad \exp(X_j), \quad \ln(X_j),$$

- The reason that those models are considered to be linear is that they are linear in the unknown parameters  $\beta_j$  for all  $j = 1, 2, \dots, k$ .
- That is, the regression function  $m$  is always linear in terms of  $\beta$

$$m(\mathbf{X}, \beta) = \mathbb{E}[Y \mid \mathbf{X}]$$

- Of course, this might not always be appropriate, e.g.

$$Y = \beta_1 e^{\beta_2 X} + \varepsilon$$

- In practice, the true nonlinear relationship might be known to be governed

$$Y = m(\mathbf{X}, \beta) + \varepsilon$$

by a differential equation up to some unknown parameters,

$$\beta$$

- Often an uncorrelated normally distributed random error is assumed with

$$\mathbb{E}[\varepsilon \mid \mathbf{X}] = 0 \quad \text{Var}[\varepsilon \mid \mathbf{X}] = \sigma^2$$

- So for a typical nonlinear regression model,

$$Y = m(\mathbf{X}, \boldsymbol{\beta}) + \varepsilon$$

we essentially retain all the assumptions from linear regression model except

$$m(\mathbf{X}, \boldsymbol{\beta})$$

is no longer assumed to be a linear function of  $\boldsymbol{\beta}$ , that is, at least one of

$$\frac{\partial m}{\partial \beta_j}$$

depends on at least one of the element of the parameter vector  $\boldsymbol{\beta}$ , e.g.

$$\begin{aligned} m(\mathbf{X}, \boldsymbol{\beta}) = \beta_1 e^{\beta_2 X} &\implies \frac{\partial m}{\partial \beta_1} = e^{\beta_2 x} \\ &\implies \frac{\partial m}{\partial \beta_2} = \beta_1 x e^{\beta_2 x} \end{aligned}$$

- Occasionally, it is possible to **transform** a nonlinear into a linear model e.g.

$$Y = \beta_1 e^{\beta_2 X} + \varepsilon$$

- It is tempting to taking the logarithmic transformation ,

$$\ln(m) = \ln(\beta_1) + \beta_2 X$$

and considering the following, which is a linear model in terms of  $\alpha_0$  and  $\alpha_1$

$$\begin{aligned} \ln(Y) &= \ln(\beta_1) + \beta_2 X + \varepsilon^* \\ &\implies Y^* = \alpha_0 + \alpha_1 X + \varepsilon^* \end{aligned}$$

which can be solved by least squares/maximum likelihood estimation.

- Care must be taken since different error assumptions are being made here.

$$\begin{aligned} Y = \beta_1 e^{\beta_2 X} \cdot \varepsilon^{**} &\implies \ln(Y) = \ln(\beta_1) + \beta_2 X + \ln(\varepsilon^{**}) \\ &\implies Y^* = \alpha_0 + \alpha_1 X + \varepsilon^* \end{aligned}$$

- Given a sample of  $n$  observations on the response and the regressors, say

$$y_i, x_{i1}, x_{i2}, \dots, x_{ik}$$

**least squares estimation** for a linear regression model involves solving

$$\hat{\boldsymbol{\beta}} = \underset{\mathbf{b}}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \left( y_i - \sum_{j=0}^k x_{ij} b_j \right)^2 \right\} = \underset{\mathbf{b}}{\operatorname{argmin}} \left\{ \sum_{i=1}^n (y_i - m(\mathbf{x}_i, \mathbf{b}))^2 \right\}$$

- For a nonlinear regression model, the problem can be stated exactly the same

$$\hat{\boldsymbol{\beta}} = \underset{\mathbf{b}}{\operatorname{argmin}} \left\{ \sum_{i=1}^n (y_i - m(\mathbf{x}_i, \mathbf{b}))^2 \right\}$$

- The difference is the minimisation for the nonlinear regression model has no closed-form solution. In general, it needs to be solved numerically.
- Given the errors are independent and identically normally distributed, then

$$\begin{aligned}\mathcal{L}(\boldsymbol{\beta}, \sigma^2) &= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left[-\frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - m(\mathbf{x}_i, \boldsymbol{\beta}))^2\right] \\ \implies \ell(\boldsymbol{\beta}, \sigma^2) &= -\ln(\mathcal{L}) = \frac{n}{2} \ln(2\pi\sigma^2) + \frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - m(\mathbf{x}_i, \boldsymbol{\beta}))^2\end{aligned}$$

- As before, it is clear that the MLE of  $\boldsymbol{\beta}$  is identical to the LSE of  $\boldsymbol{\beta}$ ,

$$\frac{\partial \ell}{\partial \sigma^2} = 0 \quad \text{and} \quad \frac{\partial \ell}{\partial \boldsymbol{\beta}} = \mathbf{0}$$

- This identical estimate  $\hat{\boldsymbol{\beta}} = \mathbf{b}$  is obtained by solving the following

$$\frac{1}{\sigma^2} \sum_{i=1}^n (y_i - m(\mathbf{x}_i, \mathbf{b})) \left( \frac{\partial}{\partial b_j} m(\mathbf{x}_i, \mathbf{b}) \right) = 0 \quad \text{for } j = 1, 2, \dots, k$$

- We will come back to MLE since treating  $\hat{\boldsymbol{\beta}}$  as MLE provides something else.
- With the usual vector notations for  $\mathbf{y}$  and  $\hat{\mathbf{y}}$ , that is

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \hat{\mathbf{y}} = \begin{bmatrix} m(\mathbf{x}_1, \hat{\boldsymbol{\beta}}) \\ m(\mathbf{x}_2, \hat{\boldsymbol{\beta}}) \\ \vdots \\ m(\mathbf{x}_n, \hat{\boldsymbol{\beta}}) \end{bmatrix} \quad \text{and} \quad \mathbf{J} = \begin{bmatrix} m'_{11} & m'_{12} & \cdots & m'_{1k} \\ m'_{21} & m'_{22} & \ddots & m'_{2k} \\ \vdots & \ddots & \ddots & \vdots \\ m'_{n1} & \cdots & \cdots & m'_{nk} \end{bmatrix}$$

where  $m'_{ij} = \frac{\partial m}{\partial b_j}$  denotes the partial derivative w.r.t to  $\beta_j$  evaluated at  $\mathbf{x}_i$ .

- Using this notation, the nonlinear equations

$$\frac{1}{\sigma^2} \sum_{i=1}^n (y_i - m(\mathbf{x}_i, \mathbf{b})) m'_{ij} = 0 \quad \text{for } j = 1, 2, \dots, k$$

that we have to solve in order to solve for MLE can be written as

$$\frac{1}{\sigma^2} \mathbf{J}^T (\mathbf{y} - \hat{\mathbf{y}}) = \mathbf{0} \implies \mathbf{J}^T (\mathbf{y} - \hat{\mathbf{y}}) = \mathbf{0}$$

- This is the **score equation**, and reduces to the **normal equation** when linear.
- A widely used numerical method for solving the estimation is to expand

$$m(\mathbf{x}_i, \boldsymbol{\beta}) \approx m(\mathbf{x}_i, {}^0\mathbf{b}) + \sum_{j=1}^k \left[ \frac{\partial m(\mathbf{x}_i, \boldsymbol{\beta})}{\partial \beta_j} \right]_{{}^0\boldsymbol{\beta}} (\beta_j - {}^0\beta_j)$$

using the Taylor approximation around some initial guess  ${}^0\mathbf{b}$ , then consider

$$y_i \approx m(\mathbf{x}_i, {}^0\mathbf{b}) + \sum_{j=1}^k \left[ \frac{\partial m(\mathbf{x}_i, \boldsymbol{\beta})}{\partial \beta_j} \right]_{\boldsymbol{\beta}={}^0\mathbf{b}} (\beta_j - {}^0b_j) + \varepsilon_i$$

$$\underbrace{y_i - m(\mathbf{x}_i, {}^0\mathbf{b})}_{{}^0y_i} \approx \sum_{j=1}^k \underbrace{\left[ \frac{\partial m(\mathbf{x}_i, \boldsymbol{\beta})}{\partial \beta_j} \right]_{\boldsymbol{\beta}={}^0\mathbf{b}}}_{{}^0J_{ij}} (\beta_j - {}^0b_j) + \varepsilon_i$$

- Linearisation of  $m(\mathbf{x}, \boldsymbol{\beta})$  at  ${}^0\mathbf{b}$  leads us to an linear auxiliary regression

$${}^0y_i = {}^0\gamma_1 {}^0z_{i1} + {}^0\gamma_2 {}^0z_{i2} + \dots + {}^0\gamma_k {}^0z_{ik} + {}^0\varepsilon_i \iff {}^0\mathbf{y} = {}^0\mathbf{J} {}^0\boldsymbol{\gamma} + {}^0\boldsymbol{\varepsilon}$$

Q: What is the significance of the auxiliary regression?

$${}^0\mathbf{y} = {}^0\mathbf{J} {}^0\boldsymbol{\gamma} + {}^0\boldsymbol{\varepsilon}$$

- If the linearisation is reasonable, then we expect

$$(\beta_j - {}^0b_j) \approx {}^0\gamma_j$$

- LSE/MLE of  ${}^0\boldsymbol{\gamma}$  is used to update the initial guess  ${}^0\mathbf{b}$ ,

$${}^0\hat{\boldsymbol{\gamma}} = \left( {}^0\mathbf{J}^T {}^0\mathbf{J} \right)^{-1} {}^0\mathbf{J}^T {}^0\mathbf{y} \implies {}^1\mathbf{b} = {}^0\mathbf{b} + {}^0\hat{\boldsymbol{\gamma}}$$

- In general, the following is used to update the  $\ell$ th estimation of  $\boldsymbol{\beta}$ ,

$${}^\ell\mathbf{b} = {}^{\ell-1}\mathbf{b} + {}^{\ell-1}\hat{\boldsymbol{\gamma}} = {}^{\ell-1}\mathbf{b} + \left( {}^{\ell-1}\mathbf{J}^T {}^{\ell-1}\mathbf{J} \right)^{-1} {}^{\ell-1}\mathbf{J}^T {}^{\ell-1}\mathbf{y}$$

until convergence, this is known as [Gauss-Newton algorithm](#).

- Consider the following dataset,

**Concentration** Substrate Concentration in ppm

**Velocity** Rate of formation in counts per min

which is about an chemical reaction for an antibiotic, known as puromycin.

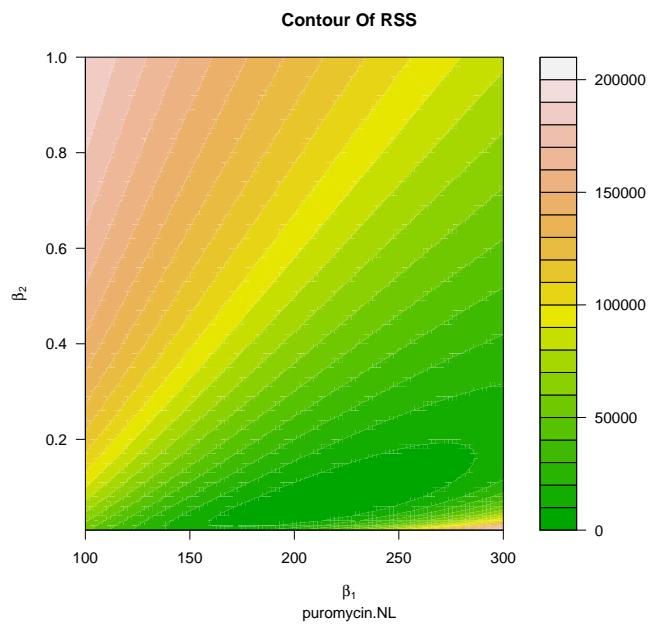
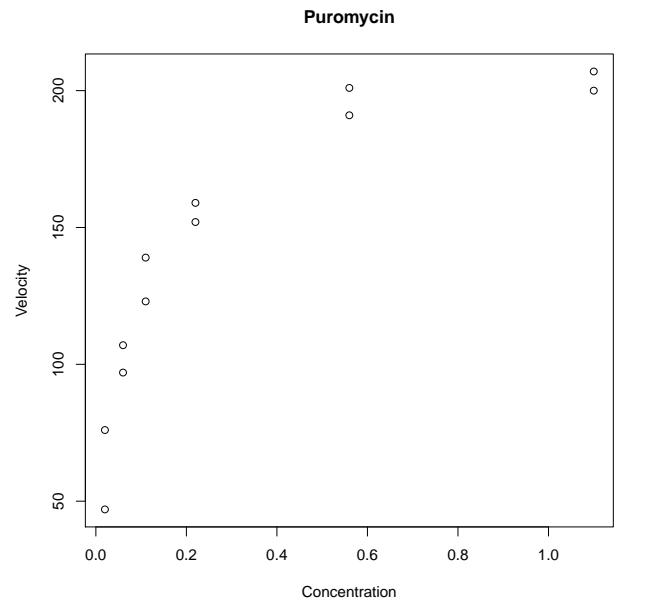
```
> puromycin.df = read.table(
+   "~/Desktop/puromycin.txt", header = TRUE)
>
> str(puromycin.df)
```

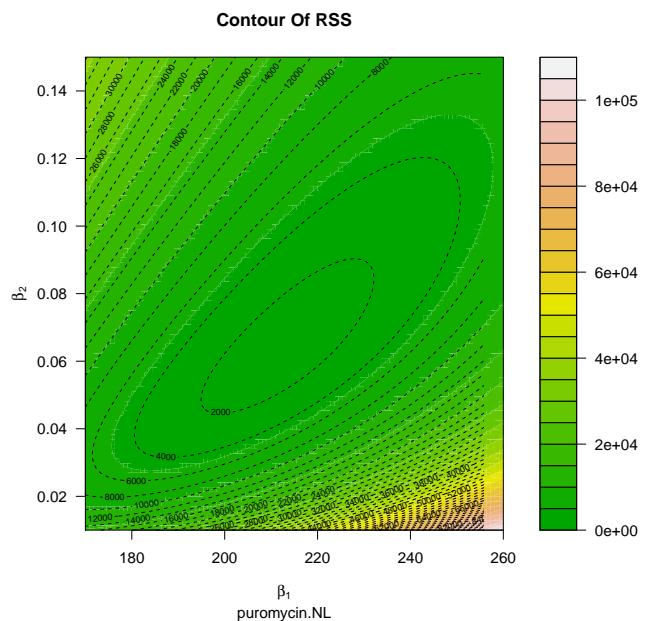
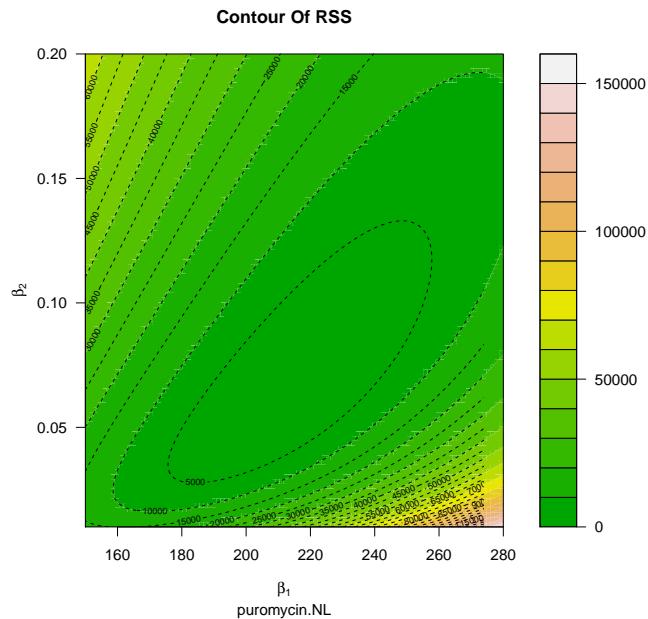
```
'data.frame': 12 obs. of 2 variables:
$ Concentration: num 0.02 0.02 0.06 0.06 0.11 0.11 0.22 0.22 0.56 0.56 ...
$ Velocity     : int 47 76 97 107 123 139 152 159 191 201 ...
```

- In biochemistry, enzyme kinetics is expect to obey the following

$$\mathbb{E}[\text{Velocity} | \text{Concentration}] = \frac{\beta_1 \text{Concentration}}{\beta_2 + \text{Concentration}}$$

with some unknown parameters,  $\beta_1$  and  $\beta_2$ .





```
> # Create a grid
> p1n = 100; p2n = 100
> par1.vec = seq(170, 260, length = p1n)
> par2.vec = seq(0.01, 0.15, length = p2n)
```

```

>
> grid = expand.grid(par1.vec, par2.vec)
>
> # RSS function
> my.func = function(x){
+   fvs = x[1]*puromycin.df[,1]/
+         (x[2]+puromycin.df[,1])
+   res = puromycin.df[,2] - fvs
+   sum(res^2)
+ }
>
> # Values of the Objective function
> RSS = matrix(
+   apply(grid, MARGIN = 1, FUN = my.func),
+   nrow = p1n, ncol = p2n)

> # Contour base
> filled.contour(par1.vec, par2.vec, RSS,
+                  color.palette=terrain.colors)
>
> # Align contour lines with the base
> mar.orig = par("mar")
> w = (3 + mar.orig[2]) * par("csi") * 2.54
> layout(matrix(c(2,1), nc = 2), widths = c(1,1cm(w)))
>
> # Add Contour lines to the base
> lev = pretty(range(RSS), 50)
> contour(par1.vec,par2.vec,RSS, levels = lev,
+          drawlabels = TRUE, lty = 2, axes = FALSE,
+          frame.plot = FALSE, add = TRUE)
>
> title(xlab = expression(beta[1]),
+        ylab = expression(beta[2]),
+        main = "Contour Of RSS",
+        sub = "puromycin.NL")

> # Initial guess based on the contour plot
> initial = list(par1=210, par2=0.07)

> # Gauss-Newton is the default algorithm in R
> parameter_each = capture.output({
+   puromycin.NL = nls(
+     Velocity~
+       par1*Concentration/(par2+Concentration),
+     start = initial, data = puromycin.df,
+     trace = TRUE)
+ })

```

```

> model = puromycin.NL
> model

Nonlinear regression model
  model: Velocity ~ par1 * Concentration/(par2 + Concentration)
    data: puromycin.df
      par1      par2
 212.68385   0.06412
  residual sum-of-squares: 1195

Number of iterations to convergence: 4
Achieved convergence tolerance: 5.97e-06

```

- Using `capture.output`, we will have RSS and parameters at each iteration.

```

> parameter_each

[1] "1480.134 : 210.00 0.07"           "1195.633 : 212.95331922 0.06431621"
[3] "1195.449 : 212.69569008 0.06413995" "1195.449 : 212.68489661 0.06412308"
[5] "1195.449 : 212.68385411 0.06412145"

> # Split according to a single white space
> tmp = strsplit(parameter_each, " ")
> str(tmp)

List of 5
$ : chr [1:7] "1480.134" ":" "" "210.00" ...
$ : chr [1:7] "1195.633" ":" "" "212.95331922" ...
$ : chr [1:7] "1195.449" ":" "" "212.69569008" ...
$ : chr [1:7] "1195.449" ":" "" "212.68489661" ...
$ : chr [1:7] "1195.449" ":" "" "212.68385411" ...

> # Coerce into a vector
> pei.vec = unlist(tmp)

> pei.vec

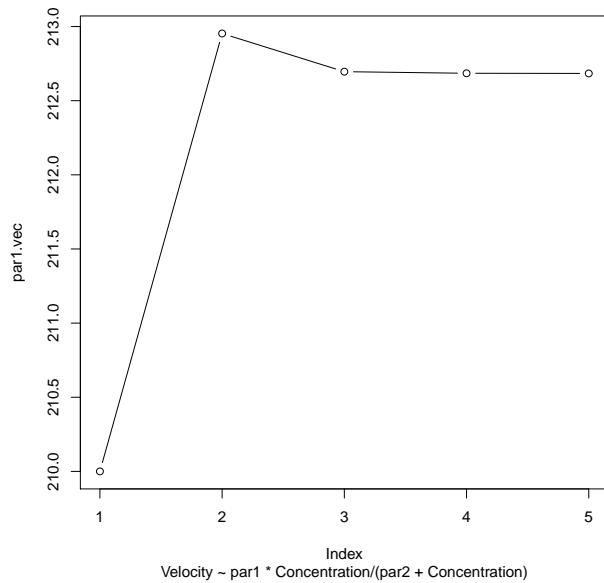
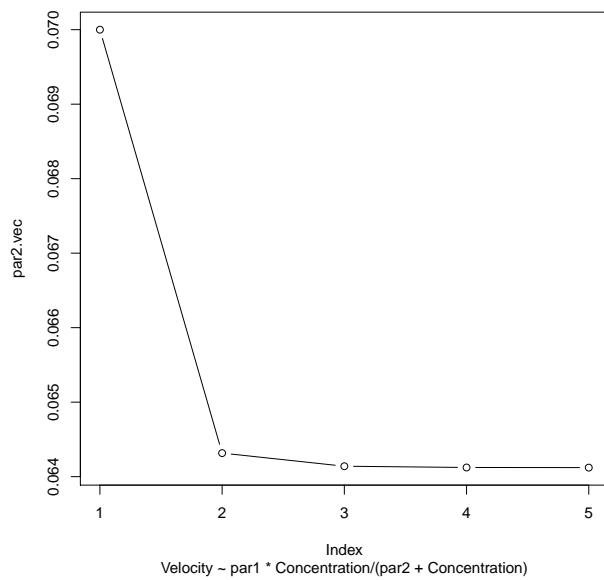
[1] "1480.134"      ":"      ""      "210.00"
[5] ""              ""      "0.07"   "1195.633"
[9] ":"              ""      "212.95331922" ""
[13] ""              "0.06431621" "1195.449" ":" 
[17] ""              "212.69569008" ""      ""
[21] "0.06413995"  "1195.449" ":"      ""
[25] "212.68489661" ""              ""      "0.06412308"
[29] "1195.449"     ":"      ""      "212.68385411"
[33] ""              ""      "0.06412145"

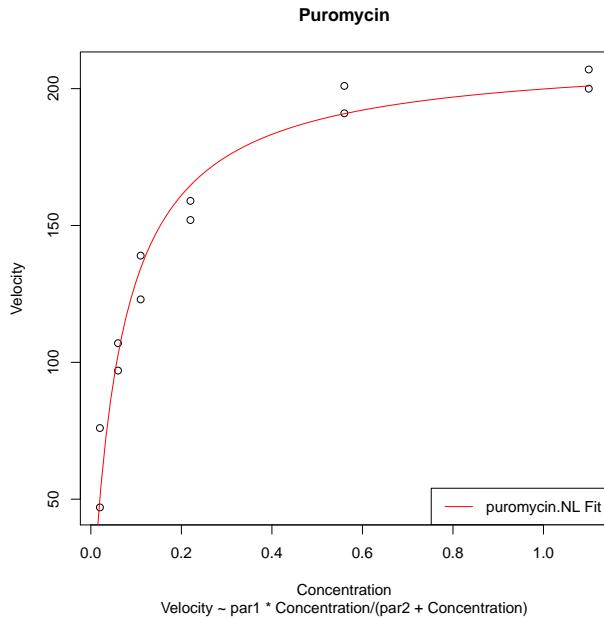
> n.itera = length(tmp); n.element = length(tmp[[1]])

> index_1 = seq(4, by = n.element, length = n.itera)
>
> index_2 = seq(7, by = n.element, length = n.itera)

> par1.vec = as.numeric(pei.vec[index_1])
>
> par2.vec = as.numeric(pei.vec[index_2])

```

$\beta_1$  $\beta_2$ 



- For this nonlinear model, it is possible to transform into a linear model easily

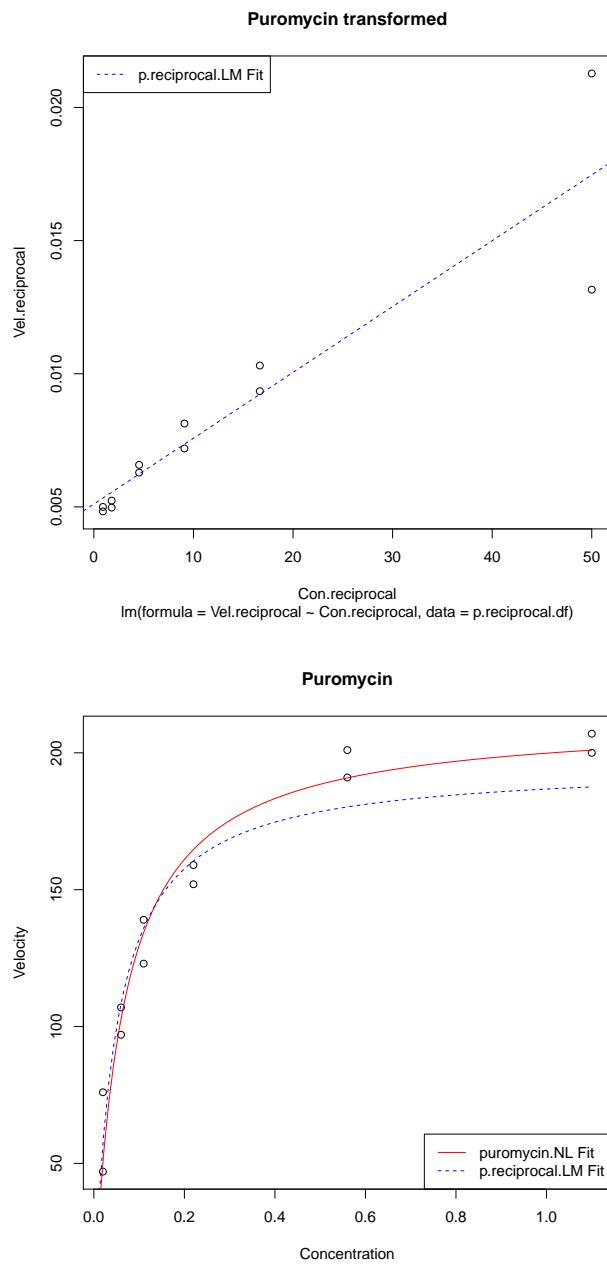
```

> Con.reciprocal = 1/puromycin.df$Concentration
>
> Vel.reciprocal = 1/puromycin.df$Velocity
>
> p.reciprocal.df = data.frame(
+   Con.reciprocal = Con.reciprocal,
+   Vel.reciprocal = Vel.reciprocal)

> p.reciprocal.LM = lm(Vel.reciprocal~Con.reciprocal,
+                       data = p.reciprocal.df)

> # Back Transform
> a.vec = p.reciprocal.LM$coefficients
>
> paras.t.vec = c(1/a.vec[1], a.vec[2]/a.vec[1])

```



- Of course, we cannot proceed further without diagnostics,

$$Y_i = m(\mathbf{X}_i, \boldsymbol{\beta}) + \varepsilon_i, \quad i = 1, 2, \dots, n$$

where  $\varepsilon_i \stackrel{i.i.d.}{\sim} \text{Normal}(0, \sigma^2)$ , we could define the residual as before

$$\hat{e}_i = y_i - \hat{y}_i = y_i - m(\mathbf{x}_i, \hat{\beta})$$

- However, diagnostic tools based on those ordinary residuals, which are useful for linear regression, can produce misleading results in nonlinear models.
- Various other types of residuals and diagnostics measures exist, e.g.

#### Projected Residuals

however, the usefulness of such residuals is not as great as ordinary residuals to linear regression, and it will involve way too much linear algebra if it is to be presented in this course, thus omitted here.

Q: The sampling distribution of  $\hat{\beta}_j$  is not available in general, since the relation between  $m$  and  $\beta_j$  is nonlinear. Can you see why that is a problem?

- Without a sampling distribution, we cannot do any statistical inference.
- However, in advanced likelihood theory, it can be shown asymptotically

$$\hat{\beta} \stackrel{a}{\sim} \text{Normal}\left(\beta, \sigma^2 \left(\mathbf{J}^T \mathbf{J}\right)^{-1}\right)$$

that is, intuitively, the sampling distribution approaches normal as  $n \rightarrow \infty$ .

- So if  $n$  is very large, it is used approximately as the sampling distribution.
- This large-sample inference for hypothesis testing and confidence interval

$$T = \frac{\hat{\beta}_j}{\hat{\sigma} \left[ (\mathbf{J}^T \mathbf{J})^{-1} \right]_{jj}^{1/2}} \sim t_{n-k} \quad \text{and} \quad \hat{\beta}_j \pm t_{\alpha/2, n-k} \hat{\sigma} \left[ \left( \mathbf{J}^T \mathbf{J} \right)^{-1} \right]_{jj}^{1/2}$$

where  $\hat{\sigma}^2$  denotes the following asymptotically unbiased estimator of  $\sigma^2$ ,

$$\hat{\sigma}^2 = \frac{1}{n-k} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

is widely used, and are known as [Wald test](#) and [Wald interval](#), respectively.

```
> summary(puromycin.NL)
```

```
Formula: Velocity ~ par1 * Concentration/(par2 + Concentration)

Parameters:
            Estimate Std. Error t value Pr(>t)
par1    2.127e+02  6.947e+00 30.614 3.24e-11 ***
par2    6.412e-02  8.281e-03   7.743 1.57e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '?' 0.1 '?' 1

Residual standard error: 10.93 on 10 degrees of freedom

Number of iterations to convergence: 4
Achieved convergence tolerance: 5.97e-06
```

- Note the t-statistics and the corresponding p-values are those for Wald test.
- Of course, it only meant to be used for a really big sample, 12 is not big!

- For most of dataset that your laptop can handle, `bootstrap` should be used at least to check whether large-sample inference is appropriate.
- Bootstrap is used to construct the sampling distribution of the estimators in a nonlinear regression model, which then can be used to compute estimated standard errors, thus produce confidence intervals and prediction intervals.
- The idea here is to take the estimated

$$\hat{\beta} \quad \text{and} \quad \hat{\sigma}^2$$

as the true underlying population parameters, and see how

$$\text{variability/uncertainty}$$

is propagated if the proposed model is valid

$$Y_i = m(\mathbf{X}_i, \hat{\beta}) + \delta_i \quad \text{where } \delta_i \stackrel{i.i.d.}{\sim} \text{Normal}(0, \hat{\sigma}^2).$$

- Specifically, we generate  $M$  sets of size- $n$  samples of  $Y_i$  based on the above model using the original  $\mathbf{X}$ , each of those simulated datasets gives one  $\tilde{\beta}$

$$y_i = m(\mathbf{x}_i, \tilde{\beta}) + \delta_i$$

then the  $M$  samples of  $\tilde{\beta}$  are used to approx. the sampling distribution of  $\hat{\beta}$ .

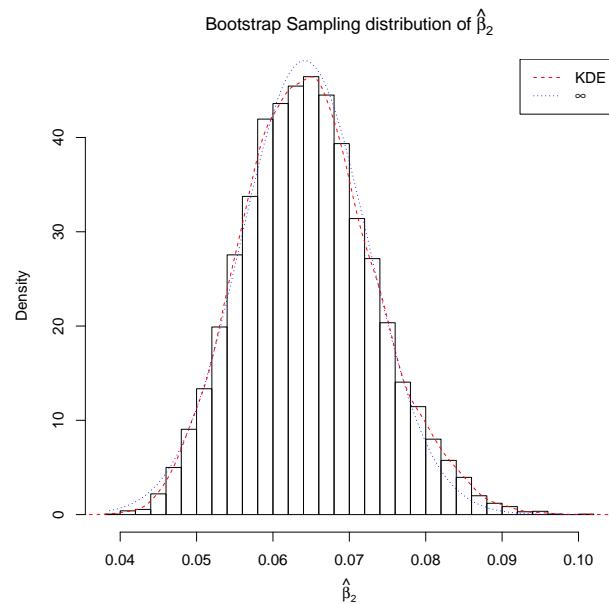
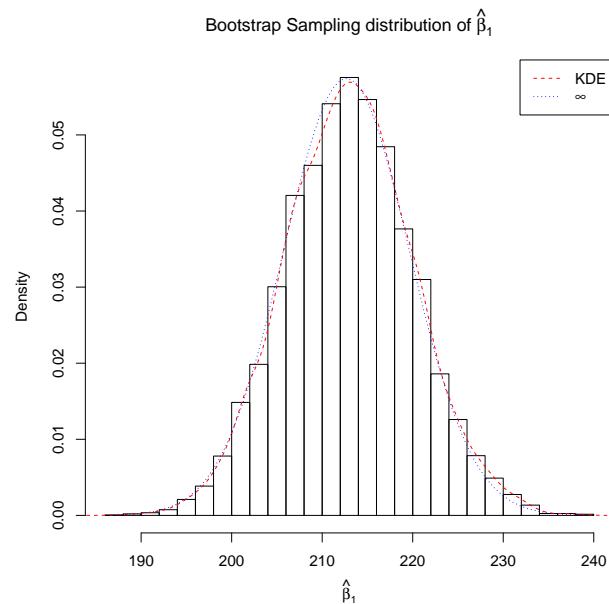
```
> M = 10000 ## Bootstrapping
> n = nrow(puromycin.df)
> fvs = fitted.values(puromycin.NL)

> vel.vec = fvs + # Generate the responses
+   rnorm(M*n, 0, summary(puromycin.NL)$sigma)

> vel.mat = matrix(vel.vec, nrow = M,
+                   ncol = n, byrow = TRUE)

> my.func = function(x){
+   tmp.NL = nls(
+     x~par1*Concentration/(par2+Concentration),
+     start = list(par1=210, par2=0.07),
+     data = puromycin.df)
+   tmp.NL$m$getPars()
+ }

> parameter.sim = apply(
+   vel.mat, MARGIN = 1, FUN = my.func)
```



```

> p = 1 # which parameter 1 or 2 in this case
>
> hist(parameter.sim[p,], probability = TRUE,
+       xlab = bquote(hat(beta)[.(p)]), breaks = 30,

```

```

+     main = bquote(paste(
+         "Bootstrap Sampling distribution of ",
+         hat(beta)[.(p)])))
>
> tmp.x = seq(min(parameter.sim[p,]), 
+               max(parameter.sim[p,]), length = 100)
>
> tmp.y = dnorm(tmp.x, puromycin.NL$m$getPars()[p],
+                 sqrt(vcov(puromycin.NL)[p,p]))
>
> lines(density(parameter.sim[p,]), col = 2, lty = 2)
> lines(tmp.x, tmp.y, col = 4, lty = 3)
>
> legend("topright", lty = 2:3, col = c(2, 4),
+         legend = c("KDE", expression(infinity)))

> # C.I. for parameters
> quantile(parameter.sim[1,], probs=c(0.025, 0.975))

2.5%    97.5%
199.4794 227.2073
```

---

```

> con.pred = 0.8 # C.I for conditional mean
>
> fvs.sim = parameter.sim[1,]*con.pred/
+   (parameter.sim[2,]+con.pred)
>
> quantile(fvs.sim, probs=c(0.025, 0.975))

2.5%    97.5%
186.8908 207.1992
```

---

```

> v.sim = fvs.sim + # Prediction interval
+   rnorm(M, 0, summary(puromycin.NL)$sigma)
>
> quantile(v.sim, probs=c(0.025, 0.975))

2.5%    97.5%
173.5875 220.4332
```

---

?

## 5 Logistic

- So far, we have only considered the following class of regression models

$$Y_i = m(\mathbf{X}, \boldsymbol{\beta}) + \varepsilon_i \quad \text{where } \varepsilon_i \sim \text{Normal}(0, \sigma^2)$$

where the errors following a normal distribution has been assumed so far.

- Consider the following dataset to see why normality is not always appropriate

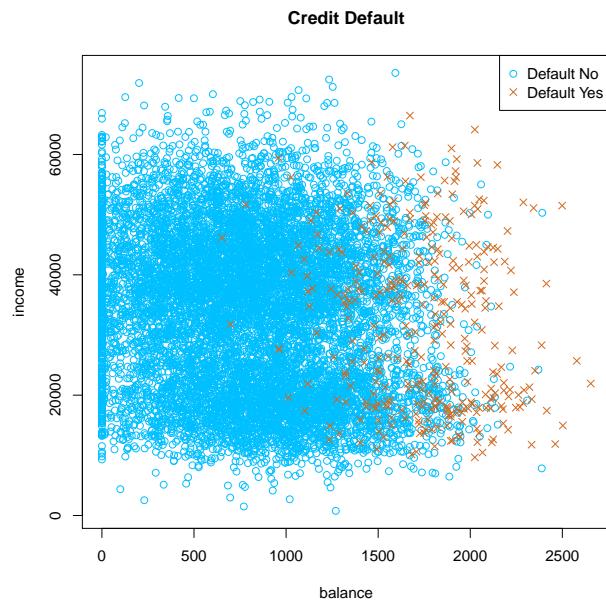
**Income** Annual income  
**Balance** Credit card balance  
**Default** Whether the card holder has defaulted  
**Student** Whether the card holder is a student

this is a kind of dataset that banks typically use to predict credit card default.

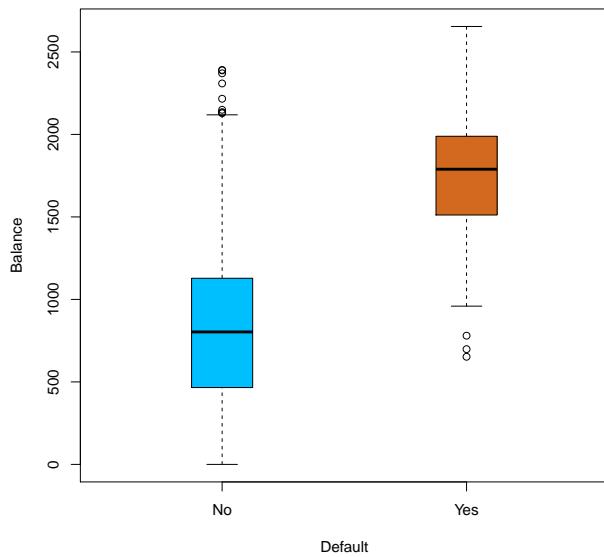
```

> credit.df = read.table(
+   "~/Desktop/credit.csv", header = TRUE, sep = ",")
> str(credit.df)

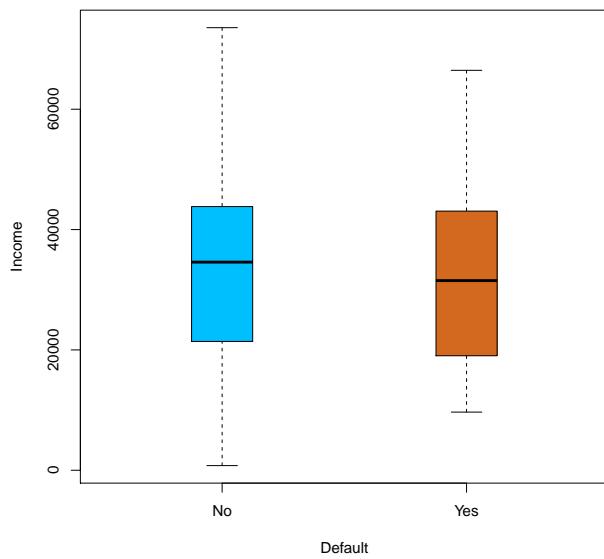
'data.frame': 10000 obs. of 4 variables:
$ default: int 0 0 0 0 0 0 0 0 ...
$ student: Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 1 ...
$ balance: num 730 817 1074 529 786 ...
$ income : num 44362 12106 31767 35704 38463 ...
  
```

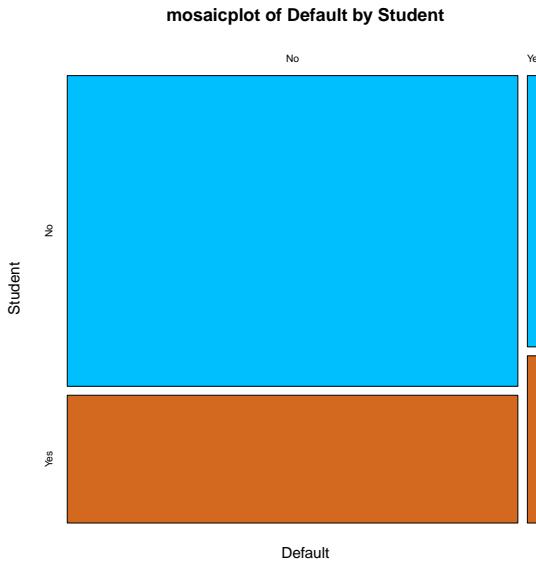


**Balance by Default**



**Income by Default**





```

> # Scatter plot
> plot(income~balance, type="n", data = credit.df,
+       xlab = "balance", ylab = "income",
+       main = "Credit Default")
>
> index = (credit.df$default == 0L)
> points(credit.df[index, 3], credit.df[index, 4],
+         col = "deepskyblue", pch = 1)
>
> index = (credit.df$default == 1L)
> points(credit.df[index, 3], credit.df[index, 4],
+         col = "chocolate", pch = 4)
>
> legend("topright", c("Default No", "Default Yes"),
+        col = c("deepskyblue", "chocolate"),
+        pch = c(1, 4))

> # Boxplot
> boxplot(balance~default, data = credit.df,
+           col = c("deepskyblue", "chocolate"),
+           boxwex = 0.25, xlab = "Default",
+           ylab = "Balance", names = c("No", "Yes"),
+           main = "Balance by Default")

> # Mosaicplot
> mosaicplot(

```

```

+   table(list(
+     Default = factor(credit.df$default,
+                       labels = c("No", "Yes")),
+     Student = credit.df$student)),
+     col = c("deepskyblue", "chocolate"),
+     main = "mosaicplot of Default by Student"
+   )

```

Q: Why a linear regression is not going to be useful/meaningful here? e.g.

$$y_i = \beta_0 + \beta_1 x_i + e_i$$

where  $X$  corresponds to `Balance` and  $Y$  corresponds `Default`

$$Y = \begin{cases} 0 & \text{if } \text{Default} = \text{No}; \\ 1 & \text{if } \text{Default} = \text{Yes}. \end{cases}$$

- Recall a simple linear regression is about the conditional mean

$$m(x, \beta) = \mathbb{E}[Y | X = x] = \beta_0 + \beta_1 x$$

- After finding  $\hat{\beta}_0$  and  $\hat{\beta}_1$  for  $\beta_0$  and  $\beta_1$ , and we use the following to estimate

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

the conditional mean  $m(x, \beta)$  which in turn is the best single predictor of  $Y$ .

- However, here `Default` is a categorical variable, which takes only

$$Y = \begin{cases} 0 & \text{if } \text{Default} = \text{No}; \\ 1 & \text{if } \text{Default} = \text{Yes}. \end{cases}$$

- Base on the simple linear regression model,  $\hat{y}_{n+1}^*$ , can take any value in  $\mathbb{R}$ ,

$$\hat{y}_{n+1}^* = \hat{\beta}_0 + \hat{\beta}_1 x_{n+1}$$

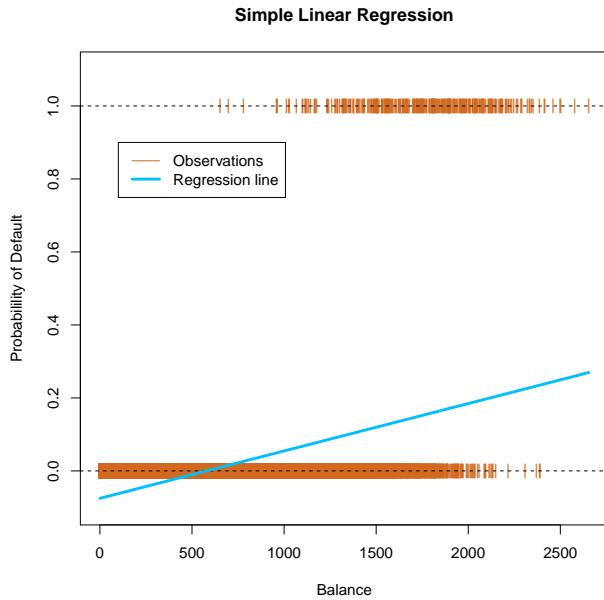
thus it will not give a meaningful prediction for the value of  $Y_{n+1}$ .

- However, it is meaningful to estimate the probability of  $Y_{n+1} = 1$  and

$$\mathbb{E}[Y | X = x] = \Pr(Y = 1 | X = x)$$

Q: Do you notice any problem even if we adjust to this interpretation of  $\hat{y}_{n+1}^*$ ?

$$\hat{\Pr}(Y_{n+1} = 1 | X_{n+1} = x_{n+1}) = \hat{y}_{n+1}^* = \hat{\beta}_0 + \hat{\beta}_1 x_{n+1}$$

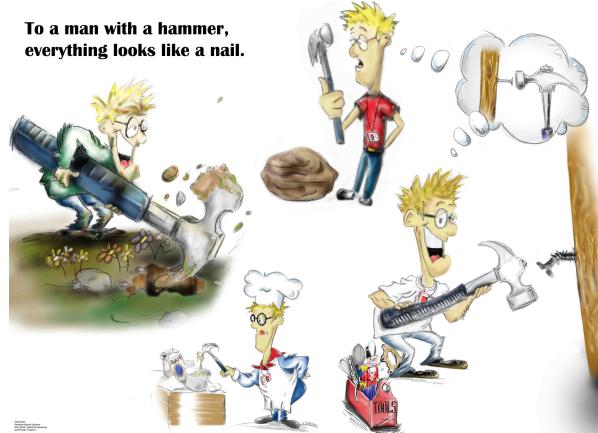


```

> x.plot = seq(
+   min(credit.df$balance), max(credit.df$balance),
+   length.out = 200)
>
> credit.LM = lm(default~balance, data = credit.df)
>
> plot(default~balance, data = credit.df, type = "n",
+       ylim = c(-0.1, 1.1), xlab = "Balance",
+       ylab = "Probability of Default",
+       main = "Simple Linear Regression")
>
> points(credit.df[,3], credit.df[,1],
+         pch = "|", col = "chocolate")
>
> abline(h = 1, lty = 2); abline(h = 0, lty = 2)
>
> n.df = data.frame(balance = x.plot)
> pred = predict(credit.LM, n.df, type = "response")
> lines(x.plot, pred, col = "deepskyblue", lwd = 3)

```

- We could indulge in a discussion on constrained least square problems, but...



- The real problem is not we don't have a more sophisticated hammer, it is we do not have a nail to start with, i.e. linear regression is not for binary  $Y$ .
- To avoid having an estimated probability outside of  $[0, 1]$ , we model

$$\Pr(Y = 1 | X = x)$$

using a function that has the range  $[0, 1]$  instead of using

$$\Pr(Y = 1 | X = x) = \mathbb{E}[Y | X = x] = m(x, \beta) = \beta_0 + \beta_1 x$$

- There are many functions that meet this requirement, if the [logistic](#) function

$$\Pr(Y = 1 | X = x) = m(x, \beta) = \frac{\exp(\beta_0 + \beta_1 x)}{1 + \exp(\beta_0 + \beta_1 x)}$$

is used, then we will end up with so-called [logistic regression](#).

Q: The conditional probability  $m$  is nonlinear in terms of  $\beta$ , how about [odds](#)?

$$o_s = \frac{p}{1 - p}$$

Q: The above ensures  $m$  is between 0 and 1, how about the error assumption?

To understand why the logistic function is a natural choice,

$$\begin{aligned} p &= \Pr(Y = 1 | X = x) = \frac{\exp(\beta_0 + \beta_1 x)}{1 + \exp(\beta_0 + \beta_1 x)} \\ &\implies \log\left(\frac{p}{1 - p}\right) = \beta_0 + \beta_1 x \end{aligned}$$

thus we are essentially making a linear assumption to the [log-odds/logit](#) of

$$y = 1$$

- In simple linear regression, the response is modelled as

$$Y_i | X_i \sim \text{Normal}(\text{mean} = m, \text{variance} = \sigma^2)$$

where the mean is assumed to be

$$m_i = \beta_0 + \beta_1 x_i$$

- Simple logistic regression can be thought as modelling the response as

$$Y_i \sim \text{Binomial}(\text{prob} = m, \text{size} = 1)$$

where the probability of success is assumed to be

$$m(x, \beta) = \frac{\exp(\beta_0 + \beta_1 x)}{1 + \exp(\beta_0 + \beta_1 x)}$$

- Recall the mean and variance of a binomial random variable, so now we have

$$\mathbb{E}[Y | X] = m \quad \text{and} \quad \text{Var}[Y | X] = m(1 - m)$$

Q: How can we estimate  $\beta_0$  and  $\beta_1$  in simple logistic regression?

- Once the parametric form is decided,  $\beta_0$  and  $\beta_1$  can be estimated by MLE

$$\hat{\beta} = \arg \max_{\mathbf{b}} \{\mathcal{L}(\mathbf{b})\}$$

- Using the maximum likelihood estimate (MLE) of  $\beta_0$  and  $\beta_1$ , we have

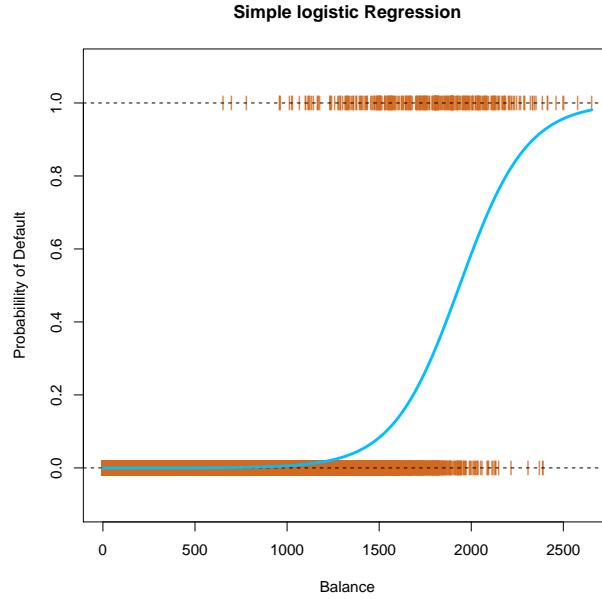
$$\hat{m} = m(x, \hat{\beta}) = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 x)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 x)}$$

which is an estimate of

$$\Pr(Y = 1 | X = x)$$

and can be used to indicate the risk of defaulting in this example, e.g.

$$\Pr(\text{default=Yes} | \text{balance}=1000)$$



Q: How to obtain maximum likelihood estimates  $\hat{\beta}_0$  and  $\hat{\beta}_1$ ?

$$f_Y(y) = \binom{1}{y} p^y (1-p)^{1-y}$$

- Under the assumption of independence and

$$p = \Pr(Y = 1 | X = x) = \frac{\exp(\beta_0 + \beta_1 x)}{1 + \exp(\beta_0 + \beta_1 x)}$$

the negative log-likelihood function is given by

$$\begin{aligned} \ell(b_0, b_1) &= -\ln(\mathcal{L}(b_0, b_1)) \\ &= -\ln \left( \prod_{i=1}^n (\Pr(Y = 1 | X = x_i))^{y_i} (\Pr(Y = 0 | X = x_i))^{1-y_i} \right) \\ &= \sum_{i=1}^n \left( \ln \left( 1 + e^{b_0 + b_1 x_i} \right) - y_i (b_0 + b_1 x_i) \right) \end{aligned}$$

- We obtain two nonlinear equations when setting the first derivatives to zero

$$\begin{aligned} \sum_{i=1}^n \left( y_i - \frac{\exp(b_0 + b_1 x_i)}{1 + \exp(b_0 + b_1 x_i)} \right) &= 0 \iff \sum_{i=1}^n (y_i - m(\mathbf{b})) = 0 \\ \sum_{i=1}^n x_i \left( y_i - \frac{\exp(b_0 + b_1 x_i)}{1 + \exp(b_0 + b_1 x_i)} \right) &= 0 \iff \sum_{i=1}^n x_i (y_i - m(\mathbf{b})) = 0 \end{aligned}$$

- Of course, we have to rely on R to solve the nonlinear equations numerically

```

> credit.LG = glm(default~balance, family = binomial,
+                     data = credit.df)
>
> coef(credit.LG)

(Intercept)      balance
-10.651330614   0.005498917

```

and we will not discuss the numerical aspect of it in this course.

- In this case, our prediction for the risk based on our estimates is given by

$$\begin{aligned}\hat{P}_r(Y = 1 | X = 1000) = \hat{m} &= \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 x)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 x)} \\ &= \frac{\exp(-10.6513 + 0.0055 \cdot 1000)}{1 + \exp(-10.6513 + 0.0055 \cdot 1000)} \\ &= 0.00576\end{aligned}$$

```

> newdata.df = data.frame(balance = 1000)
>
> mhat = predict(credit.LG, newdata = newdata.df,
+                 type = "response")
> mhat

1
0.005752145

```

```

> predict(credit.LG, newdata = newdata.df) # logit

1
-5.152414

```

```
> (os = mhat/(1-mhat)) # odds
```

```
1
0.005785424
```

```
> log(os) # log odds
```

```
1
-5.152414
```

- Of course, logistics regression can be used for more than one predictor.

$$m(\mathbf{X}, \boldsymbol{\beta}) = \frac{\exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3)}{1 + \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3)}$$

```

> credit.all3.LG = glm(
+   default~balance+income+student,
+   family = binomial, data = credit.df)

> summary(credit.all3.LG)

Call:
glm(formula = default ~ balance + income + student, family = binomial,
     data = credit.df)

Deviance Residuals:
    Min      1Q  Median      3Q      Max 
-2.4691 -0.1418 -0.0557 -0.0203  3.7383 

Coefficients:
            Estimate Std. Error z value Pr(>z)    
(Intercept) -1.087e+01  4.923e-01 -22.080 < 2e-16 ***
balance     5.737e-03  2.319e-04  24.738 < 2e-16 ***
income      3.033e-06  8.203e-06  0.370  0.71152    
studentYes -6.468e-01  2.363e-01 -2.738  0.00619 **  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 2920.6 on 9999 degrees of freedom
Residual deviance: 1571.5 on 9996 degrees of freedom
AIC: 1579.5

Number of Fisher Scoring iterations: 8

```

- Note the sampling distributions of  $\hat{\beta}$  is based on asymptotic approximations.
- Similar to nonlinear regression models with normality, it can be shown MLE,

$$\hat{\beta}$$

for a logistic regression also follows a normal distribution asymptotically

$$\hat{\beta} \stackrel{a}{\sim} \text{Normal}(\beta, \mathbf{H}^{-1})$$

where  $\mathbf{H}$  is the Hessian matrix of the negative log-likelihood function

$$\ell(\beta; \mathbf{X})$$

evaluated at  $\hat{\beta}$ , that is,

$$h_{ij} = \left. \frac{\partial^2 \ell}{\partial \beta_i \partial \beta_j} \right|_{\beta=\hat{\beta}} \quad \text{for } i, j = 0, 1, \dots, k$$

- Of course, keep in mind this large-sample inference is not always appropriate.
- For a large sample, a likelihood ratio is useful to compare two nested models,

$$\Lambda = \frac{\mathcal{L}_r(\hat{\beta}_r; \mathbf{X})}{\mathcal{L}(\hat{\beta}; \mathbf{X})}$$

where  $\mathcal{L}(\hat{\beta}; \mathbf{X})$  denotes the likelihood function of a logistic model,

$$A: m(\mathbf{X}, \beta) = \frac{\exp(\beta_0 + \beta_1 x_1 + \dots + \beta_k x_k)}{1 + \exp(\beta_0 + \beta_1 x_1 + \dots + \beta_k x_k)}$$

which is known as the **full/saturated** model, evaluated at its MLE value  $\hat{\beta}$ ,

$$\mathcal{L}_r(\hat{\beta}_r)$$

is the likelihood function of the reduced model of  $A$  evaluated at its MLE  $\hat{\beta}_r$ .

- It is “reduced” in the sense that some of the  $\beta_j$  is/are forced to be zero.
- The following can be shown to be true when the reduced model is valid,

$$-2 \ln(\Lambda) \stackrel{a}{\sim} \chi^2_{df}$$

that is, the  $\beta_j$  that is/are forced to be zero is/are indeed zero.

- The degrees of freedom  $df$  is the number of  $\beta_j$  that is forced to be zero.
- The test based on this is known as the **likelihood ratio test**.

```
> # LR test of significance, similar to F-test
> credit.null.LG = glm(default~1, family = binomial,
+                         data = credit.df)
>
> LR.test = 2*(logLik(credit.all3.LG)[1]
+               -logLik(credit.null.LG)[1])
>
> 1-pchisq(LR.test, 3)
```

---

[1] 0

---

- Of course, likelihood ratio test can be used to do variable selection

```
> # LR test for reduced, similar to partial F-test
> credit.no.income.LG =
+   glm(default~-income, family = binomial,
+        data = credit.df)
>
> LR.test = 2*(logLik(credit.all3.LG)[1]
+               -logLik(credit.no.income.LG)[1])
>
> 1-pchisq(LR.test, 1)
```

---

[1] 0.7115139

---

- Since we don’t have efficient and effective methods to check the validity of the model, and appropriateness of using the asymptotic approximation, care must be taken if  $n$  is small.
- If we have a grouped dataset, then we can try checking the validity.
- Consider the following dataset

```
> ingots.df
```

|    | heat | soak | notready | total |
|----|------|------|----------|-------|
| 1  | 7    | 1.0  | 0        | 10    |
| 2  | 14   | 1.0  | 0        | 31    |
| 3  | 27   | 1.0  | 1        | 56    |
| 4  | 51   | 1.0  | 3        | 13    |
| 5  | 7    | 1.7  | 0        | 17    |
| 6  | 14   | 1.7  | 0        | 43    |
| 7  | 27   | 1.7  | 4        | 44    |
| 8  | 51   | 1.7  | 0        | 1     |
| 9  | 7    | 2.2  | 0        | 7     |
| 10 | 14   | 2.2  | 2        | 33    |
| 11 | 27   | 2.2  | 0        | 21    |
| 12 | 51   | 2.2  | 0        | 1     |
| 13 | 7    | 2.8  | 0        | 12    |
| 14 | 14   | 2.8  | 0        | 31    |
| 15 | 27   | 2.8  | 1        | 22    |
| 16 | 51   | 2.8  | 0        | 0     |
| 17 | 7    | 4.0  | 0        | 9     |
| 18 | 14   | 4.0  | 0        | 19    |
| 19 | 27   | 4.0  | 1        | 16    |
| 20 | 51   | 4.0  | 0        | 1     |

which is a grouped dataset, "grouped" in the sense that there are groups of data points that have the same  $\mathbf{X}$ , so there are 20 binomial distributions,

$$S_i \sim \text{Binomial}(\text{prob} = m, \text{size} = n_i) \quad \text{for } i = 1, 2, \dots, 20$$

- Diagnostics for linearity and independence can be done using Pearson residuals

$$\hat{e}_i = \frac{s_i - n_i \hat{m}_i}{\sqrt{n_i \hat{m}_i (1 - \hat{m}_i)}}$$

where  $\hat{m}_i$  is the "fitted valued", that is, the estimated probability of

$$\Pr[Y_i = 1 \mid \mathbf{X}_i = \mathbf{x}_i]$$

and  $s_i$  is the number of observations belonging to  $Y = 1 \cap \mathbf{X} = \mathbf{x}_i$  and  $n_i$  is the number of observations belonging to  $\mathbf{X} = \mathbf{x}_i$ .

- Pearson's is similar to standardised residual in multiple linear regression.

Q: Can you see the similarity?

$$\hat{e}'_i = \frac{y_i - \hat{y}_i}{\hat{\sigma} \sqrt{1 - p_{ii}}}$$

Q: What do you expect to see in  $\hat{e}_i$  Vs  $\text{logit}$  if the assumptions are valid?

- Fitting logistic regression to this dataset, we have

```
> ingots.LG = glm(
+   notready/total~heat+soak, # notice the syntax
+   family=binomial, data=ingots.df, weight=total)

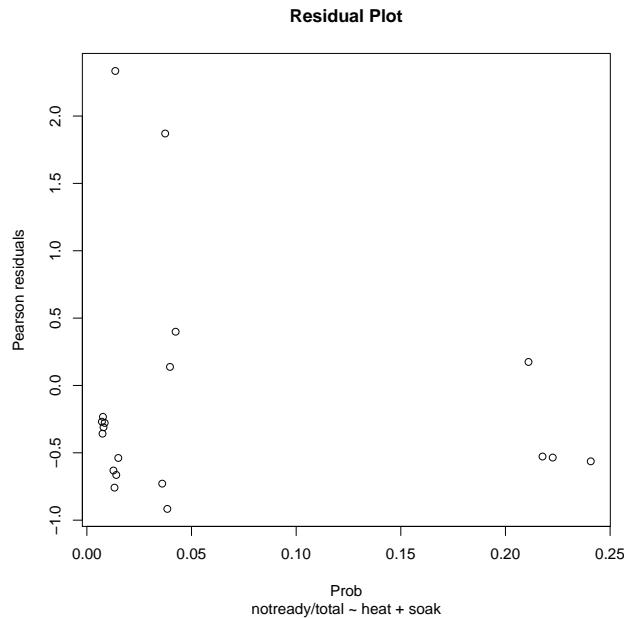
> summary(ingots.LG)
```

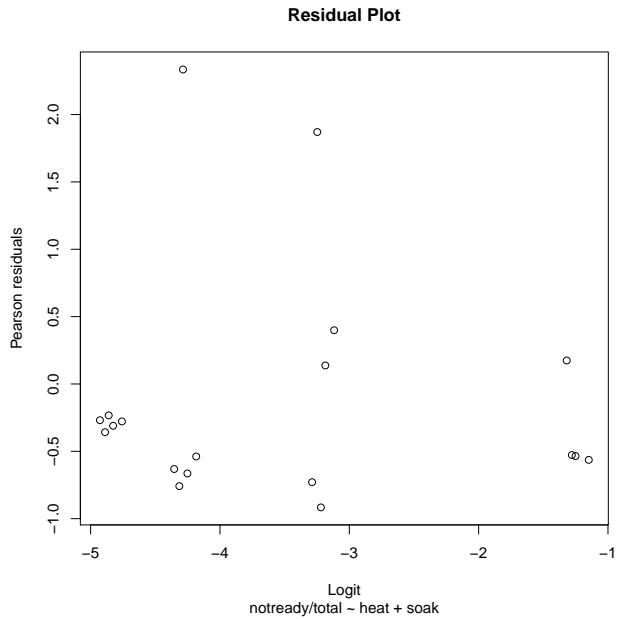
```
Call:
glm(formula = notready/total ~ heat + soak, family = binomial,
     data = ingots.df, weights = total)

Coefficients:
            Estimate Std. Error z value Pr(>z)
(Intercept) -5.55917    1.11969 -4.965 6.87e-07 ***
heat         0.08203    0.02373  3.456 0.000548 ***
soak         0.05677    0.33121  0.171 0.863906
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- In this case, we can try checking the residuals and get some feedback

```
> pres = residuals(ingots.LG, type = "pearson")
>
> fvs = predict(ingots.LG, type = "response")
```



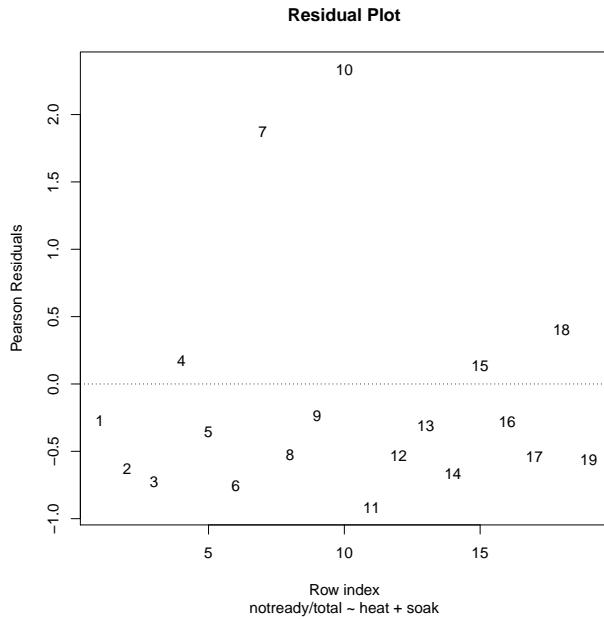


```

> plot(fvs, pres,
+       xlab="Fitted Conditional Probability",
+       ylab = "Pearson residuals",
+       main="Residual Plot",
+       sub = deparse(ingots.LG$formula))
>
> fvs = predict(ingots.LG, type = "link") # logit
>
> plot(fvs, pres, xlab="Logit",
+       ylab = "Pearson residuals",
+       main="Residual Plot",
+       sub = deparse(ingots.LG$formula))

```

- Since the dataset is very small, the residual plot is not very informative in terms of linearity or independence.
- However, in this case, we can use it to identify outliers, high leverage points, and influential points. e.g. 7 and 10 are two possible outliers that distorted our regression fit.



- Consider the following simulation study

```

> set.seed(1)
> n = 100
>
> x1 = rnorm(n, mean = 0, sd = 6)
> x2 = rt(n, df = 1)
> x3 = rbinom(n, size = 1, prob = 0.45)

> beta0 = 0.6
> beta1 = 0.3
> beta2 = 0.1
> beta3 = 0.1
>
> logit = beta0 + beta1*x1 + beta2*x2 + beta3*x3

> true.pi = exp(logit)/(1+exp(logit))

> m = 100
>
> y = integer(n)
> for (i in 1:n){
+   y[i] = rbinom(1, size = m, prob = true.pi[i])
+ }

```

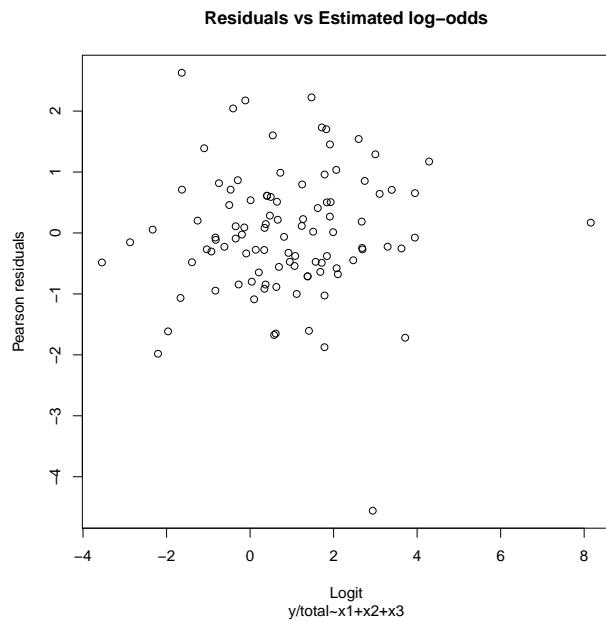
```

> sim.df = data.frame(
+   y = y, x1 = x1, x2 = x2, x3 = x3,
+   total = rep(m, n))

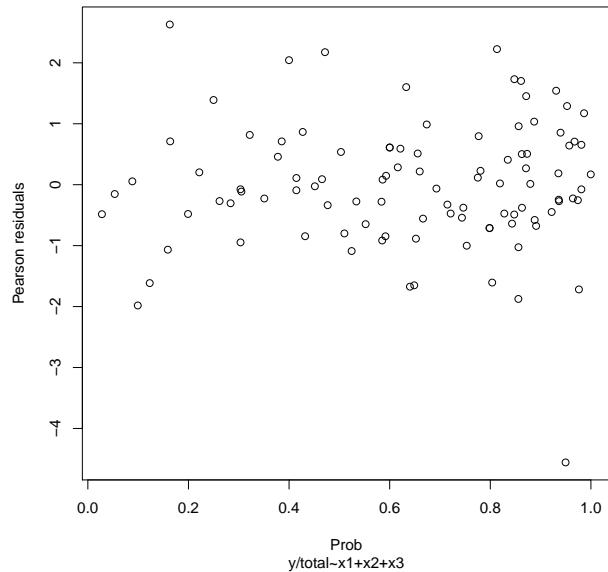
> sim.LG = glm(y/total~x1+x2+x3, family = binomial,
+                 weights = total, data = sim.df)

> res = residuals(sim.LG, type="pearson")
>
> logit = predict(sim.LG) # default logit
>
> prob = predict(sim.LG, type = "response") # Prob

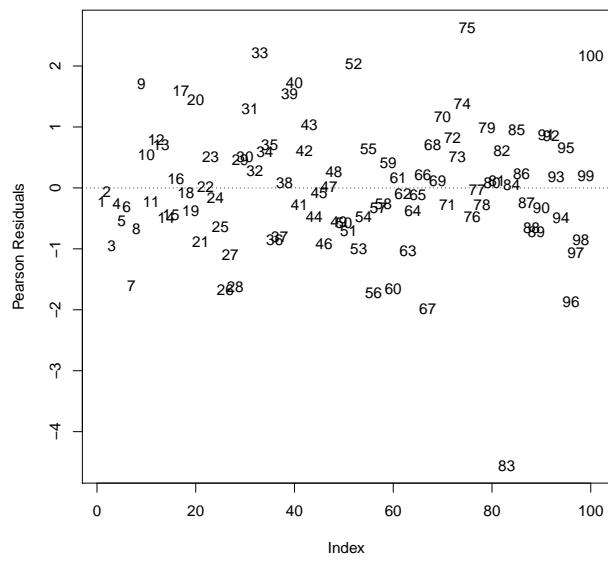
```

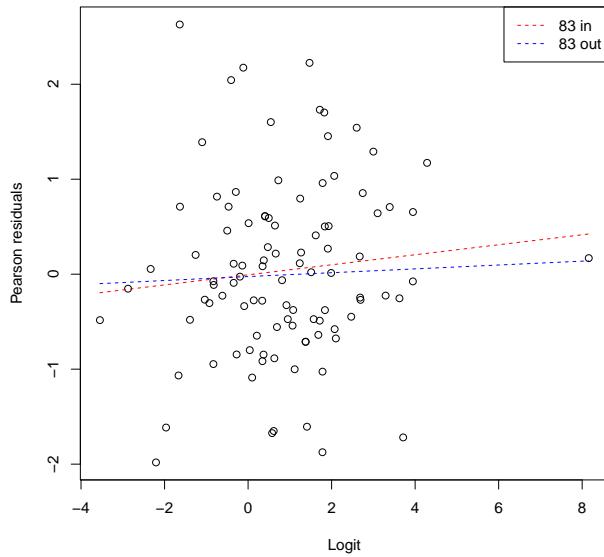


**Residuals vs Estimated  $\Pr(Y=1|X)$**



**y/total~x1+x2+x3**





```

> plot(res, type="n",
+       ylab = "Pearson Residuals",
+       main="y/total~x1+x2+x3")
>
> text(logit); abline(h=0,lty=3)
>
> plot(logit[-c(83)], res[-c(83)],
+       xlab="Logit", ylab="Pearson residuals")
>
> lines(smooth.spline(logit[-c(83)], res[-c(83)]),
+        col = 2, lty = 2)
>
> lines(smooth.spline(logit, res), col = 4, lty = 2)
>
> legend("topright", c("83 in", "83 out"),
+        col = c(2, 4), lty = 2)

```

- Checking high leverage points

```
> head(sort(hatvalues(sim.LG), decreasing = TRUE))
```

|            |            |            |            |            |            |
|------------|------------|------------|------------|------------|------------|
| 21         | 46         | 62         | 36         | 71         | 45         |
| 0.37676714 | 0.15760469 | 0.13174284 | 0.07640166 | 0.06673067 | 0.06273533 |

- Check influential points

```
> head(sort(cooks.distance(sim.LG),
+           decreasing = TRUE))
```

|            |            |            |            |            |            |
|------------|------------|------------|------------|------------|------------|
| 21         | 83         | 75         | 67         | 52         | 46         |
| 0.19014483 | 0.11674260 | 0.08128077 | 0.06087118 | 0.05277872 | 0.04652510 |

- 83 is an influential outlier and 21 is a high leverage influential point, thus

```
> sim.final.LG =
+   glm(y/total ~ x1+x2+x3, family=binomial,
+       weights = total, data=sim.df[-c(83, 21),])
```

```
> summary(sim.final.LG)
```

```
Call:
glm(formula = y/total ~ x1 + x2 + x3, family = binomial, data = sim.df[-c(83,
21), ], weights = total)

Deviance Residuals:
    Min      1Q  Median      3Q     Max
-2.06696 -0.59007 -0.07006  0.57910  2.56599

Coefficients:
            Estimate Std. Error z value Pr(>z)
(Intercept) 0.593201  0.033575 17.668 <2e-16 ***
x1          0.303756  0.006812 44.594 <2e-16 ***
x2          0.100487  0.010559  9.517 <2e-16 ***
x3          0.129673  0.051550  2.515  0.0119 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 3474.622 on 97 degrees of freedom
Residual deviance:  87.842 on 94 degrees of freedom
AIC: 521.32

Number of Fisher Scoring iterations: 4
```

```
> c(beta0, beta1, beta2, beta3)
```

|     |     |     |     |     |
|-----|-----|-----|-----|-----|
| [1] | 0.6 | 0.3 | 0.1 | 0.1 |
|-----|-----|-----|-----|-----|

- Another thing that becomes available when we have the luxury of a grouped dataset is a generalisation of multiple  $R^2$  in multiple linear regression.

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}}$$

- The **deviance** of a model, say  $A$ , is defined to be

$$d_A = 2 \left( \ln(\mathcal{L}_{max}) - \ln(\mathcal{L}_A) \right)$$

where  $\mathcal{L}_A$  is the likelihood function of model  $A$  evaluated at the MLE value,

$$\begin{aligned} \mathcal{L}_A &= \prod_{i=1}^n \frac{n_i!}{s_i!(n_i - s_i)!} \hat{m}_i^{s_i} (1 - \hat{m}_i)^{n_i - s_i} \\ \mathcal{L}_{max} &= \prod_{i=1}^n \frac{n_i!}{s_i!(n_i - s_i)!} \left( \frac{s_i}{n_i} \right)^{s_i} \left( 1 - \left( \frac{s_i}{n_i} \right) \right)^{n_i - s_i} \end{aligned}$$

and  $\mathcal{L}_{max}$  is the maximum possible value of likelihood if the model is valid.

- The deviance plays a similar role that RSS has in a multiple linear regression.

- Therefore, the deviance of logistic regression model  $A$  explicitly is given by

$$\begin{aligned} d_A &= 2 \left( \ln(\mathcal{L}_{max}) - \ln(\mathcal{L}_A) \right) \\ &= 2 \sum_{i=1}^n \left( s_i \ln \left( \frac{s_i}{\hat{s}_i} \right) + (n_i - s_i) \ln \left( \frac{n_i - s_i}{n_i - \hat{s}_i} \right) \right) \quad \text{where } \hat{s}_i = n_i \hat{m}_i. \end{aligned}$$

- Using this form, we can conclude that deviances are always  $\geq 0$  and that the larger the deviance the worse the fit, 0 deviance indicates a perfect fit.
- The deviance  $R^2$  for a model  $A$  is defined to be

$$R_d^2 = 1 - \frac{d_A}{d_0}$$

where  $d_0$  is the deviance of the null model, that is, the logistics regression

$$m(\mathbf{X}, \beta_0) = \frac{\exp(\beta_0)}{1 + \exp(\beta_0)}$$

that includes only the intercept.

```
> summary(sim.final.LG)
```

| Coefficients:  |          |            |         |            |      |
|----------------|----------|------------|---------|------------|------|
|                | Estimate | Std. Error | z value | Pr(>z)     |      |
| (Intercept)    | 0.593201 | 0.033575   | 17.668  | <2e-16 *** |      |
| x1             | 0.303756 | 0.006812   | 44.594  | <2e-16 *** |      |
| x2             | 0.100487 | 0.010559   | 9.517   | <2e-16 *** |      |
| x3             | 0.129673 | 0.051550   | 2.515   | 0.0119 *   |      |
| ---            |          |            |         |            |      |
| Signif. codes: | 0        | ***        | 0.001   | **         | 0.05 |
| ?              | ?        | ?          | ?       | ?          | ?    |
| 1              |          |            |         |            |      |

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 3474.622 on 97 degrees of freedom  
Residual deviance: 87.842 on 94 degrees of freedom  
AIC: 521.32

Number of Fisher Scoring iterations: 4

- Note Null deviance is  $d_0$ , and Residual deviance is  $d_A$ , thus  $R_d^2$

```
> 1 - sim.final.LG$deviance / sim.final.LG>null.deviance
```

|               |
|---------------|
| [1] 0.9747189 |
|---------------|

indicates our model capture most of deviation in the data.

## 6 Poisson

- So far, we have considered two kinds of regressions:

1. Regressions where the response are

continuous.

- We assumed that the responses were normally distributed,

$$Y_i \mid \mathbf{X}_i \sim \text{Normal}(\text{mean} = m_i, \text{var} = \sigma^2)$$

with the mean depending on the regressors/predictors.

2. Regressions where the response are  
 binary, or the number “successes” out of a fixed number of “trials”.

- We assumed that the responses were binomially distributed,

$$Y_i | \mathbf{X}_i \sim \text{Binomial}(\text{prob} = m_i, \text{size} = 1)$$

$$S_i | \mathbf{X}_i \sim \text{Binomial}(\text{prob} = m_i, \text{size} = n_i)$$

with the success probability depending on the regressors/predictors.

- Now we examine a third type of regression,

[Poisson Regression](#),

where the responses are

counts

of some relatively rare event, e.g.

defects in a manufactured product, errors or bugs in software.

- We will assume each individual response  $Y_i$  follows a Poisson distribution,

$$Y_i | \mathbf{X}_i \sim \text{Poisson}(\text{mean} = m_i)$$

whose mean depends on the regressors/predictors.

Q: What are the reasons for not using the first two kinds of regressions?

- Consider the following dataset

|        |                                    |
|--------|------------------------------------|
| Damage | The number damages on the aircraft |
| Type   | Types of aircraft                  |
| Month  | Total months of aircrew experience |
| Load   | Bomb load in tons                  |

```
> bomber.df = read.table("~/Desktop/aircraft.csv",
+                         header = TRUE, sep = ",")
>
> str(bomber.df)

'data.frame':   30 obs. of  4 variables:
$ Damage: int  0 1 0 0 0 0 1 0 0 2 ...
$ Type  : int  0 0 0 0 0 0 0 0 0 0 ...
$ Month : int  4 4 4 5 5 5 6 6 6 7 ...
$ Load   : num  91.5 84 76.5 69 61.5 80 72.5 65 57.5 50 ...
```

- We are interested in finding a predictive model with `Damage` as the response.
- As was the case in the logistic regression, we do NOT write the model as

$$Y_i = \mathbb{E}[Y_i | \mathbf{X}_i; \beta] + \varepsilon_i$$

but rather, we link  $\beta$  to a parameter of the distribution the response.

- In multiple linear regression model, we could perceive it as

$$Y_i | X_i \sim \text{Normal}(\text{mean} = m_i, \text{var} = \sigma^2)$$

where the mean is linked by

$$m_i = \mathbf{x}_i^T \boldsymbol{\beta}$$

- In multiple logistic regression model, we have

$$Y_i | X_i \sim \text{Binomial}(\text{prob} = m_i, \text{size} = 1)$$

where the probability of success, which is also the mean, is linked by

$$m_i = \frac{\exp(\mathbf{x}_i^T \boldsymbol{\beta})}{1 + \exp(\mathbf{x}_i^T \boldsymbol{\beta})}$$

- Recall the probability mass function of a Poisson random variable is given by

$$\Pr[Y = y] = \frac{e^{-\lambda} \lambda^y}{y!}$$

- Notice any Poisson random variable has the following property

$$\mathbb{E}[Y] = \text{Var}[Y] = \lambda$$

- Since  $\lambda$  must be positive, the following is often used in Poisson regression

$$Y_i | X_i \sim \text{Poisson}(\text{mean} = \text{var} = m_i)$$

where the mean/variance is linked by

$$m_i = \exp(\mathbf{x}_i^T \boldsymbol{\beta})$$

- With the independence assumption, then negative log-likelihood function is

$$\mathcal{L}(\boldsymbol{\beta}; \mathbf{X}) = \prod_{i=1}^n \frac{e^{-m_i} m_i^{y_i}}{y_i!} \implies \ell(\boldsymbol{\beta}; \mathbf{X}) = \sum_{i=1}^n (m_i + \ln(y_i!) - y_i \ln(m_i))$$

- As before, the MLE of  $\boldsymbol{\beta}$  can be found numerically,

```
> bomber.PS = glm(Damage ~ Type + Month + Load,
+                     family=poisson, data=bomber.df)
> bomber.PS
```

```
Call: glm(formula = Damage ~ Type + Month + Load, family = poisson,
          data = bomber.df)

Coefficients:
(Intercept)      Type       Month      Load
-0.40602     0.56877    0.16543   -0.01352

Degrees of Freedom: 29 Total (i.e. Null); 26 Residual
Null Deviance: 53.88
Residual Deviance: 25.95      AIC: 87.65
```

- The deviances for Poisson regression models,

$$d_A = 2(\ln(\mathcal{L}_{max}) - \ln(\mathcal{L}_A)) \quad \text{and} \quad d_0 = 2(\ln(\mathcal{L}_{max}) - \ln(\mathcal{L}_0))$$

are interpreted as for grouped logistic regression.

```
> 1 - bomber.PS$deviance / bomber.PS>null.deviance
```

```
[1] 0.5183429
```

- Recall a likelihood ratio is based on the asymptotic result

$$-2 \ln \left( \frac{\mathcal{L}_r}{\mathcal{L}_A} \right) \xrightarrow{a} \chi^2_{df}$$

where  $\mathcal{L}_A$  is the likelihood function of a model  $A$  evaluated at its MLE and  $\mathcal{L}_r$  is the likelihood function of a reduced model of  $A$  evaluated at its MLE.

Q: Do you see the connection between LR-test and deviances?

```
> 1 - pchisq(bomber.PS$deviance, bomber.PS$df.residual)
```

```
[1] 0.4656818
```

- The large  $p$ -value above means we have no evidence of lack of fit.
- The small  $p$ -value below means at least one of the regressors is needed.

```
> 1 - pchisq(bomber.PS>null.deviance, bomber.PS$df.null)
```

```
[1] 0.003337052
```

- Again, we should check the validity of using asymptotic approximation,

```
> M = 100000 # bootstrap simulation

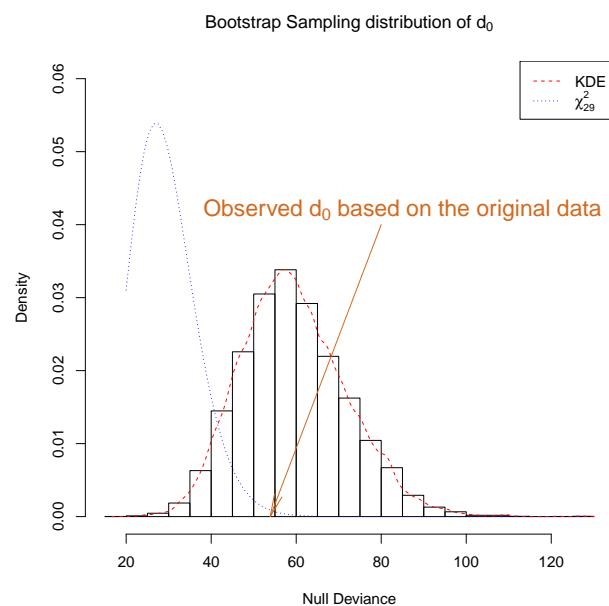
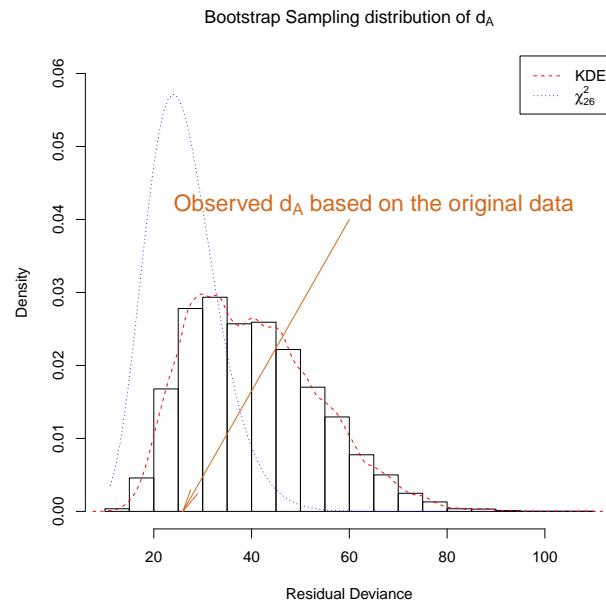
> n = nrow(bomber.df)

> fvs = fitted.values(bomber.PS) # mhat = mean

> dam.vec = rpois(M, lambda = fvs)

> dam.mat = matrix(dam.vec, nrow = M,
+                     ncol = n, byrow = TRUE)

> my.func = function(x){
+   tmp.PS = glm(x ~ Type + Month + Load,
+                 family=poisson, data=bomber.df)
+   return(c(tmp.PS$deviance, tmp.PS>null.deviance))
+ }
```



```
> summary(bomber.PS)
```

```
Call:
glm(formula = Damage ~ Type + Month + Load, family = poisson,
     data = bomber.df)
```

```

Deviance Residuals:
    Min      1Q  Median      3Q     Max
-1.6418 -1.0064 -0.0180  0.5581  1.9094

Coefficients:
            Estimate Std. Error z value Pr(>z)
(Intercept) -0.406023  0.877489 -0.463  0.6436
Type         0.568772  0.504372  1.128  0.2595 *
Month        0.165425  0.067541  2.449  0.0143 *
Load        -0.013522  0.008281 -1.633  0.1025
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 53.883 on 29 degrees of freedom
Residual deviance: 25.953 on 26 degrees of freedom
AIC: 87.649

Number of Fisher Scoring iterations: 5

```

- Notice those are Wald z-tests which are based on asymptotic results as well.
- As before, Wald test relies on MLE has the following asymptotic property

$$\hat{\beta} \stackrel{a}{\sim} \text{Normal}(\beta, \mathbf{H}^{-1})$$

where  $\mathbf{H}$  is the Hessian matrix of the negative log-likelihood function,

$$\mathbf{H} = \mathbf{X}^T \mathbf{V} \mathbf{X}$$

where  $\mathbf{V}$  is the diagonal matrix with the exponential of each element of

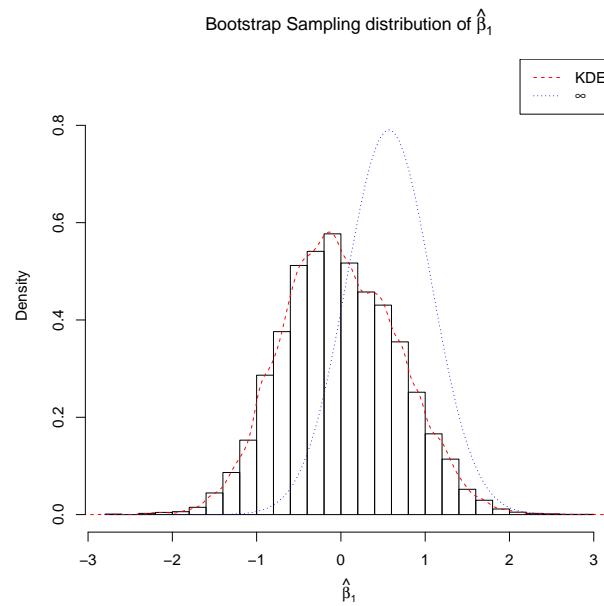
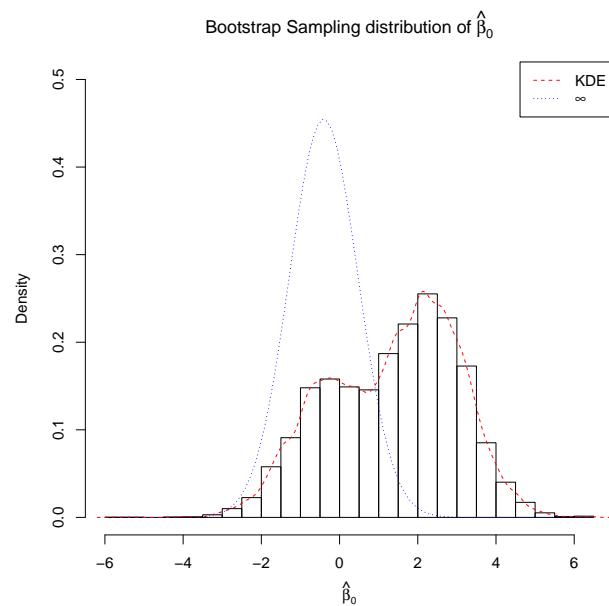
$$\mathbf{X}\hat{\beta}$$

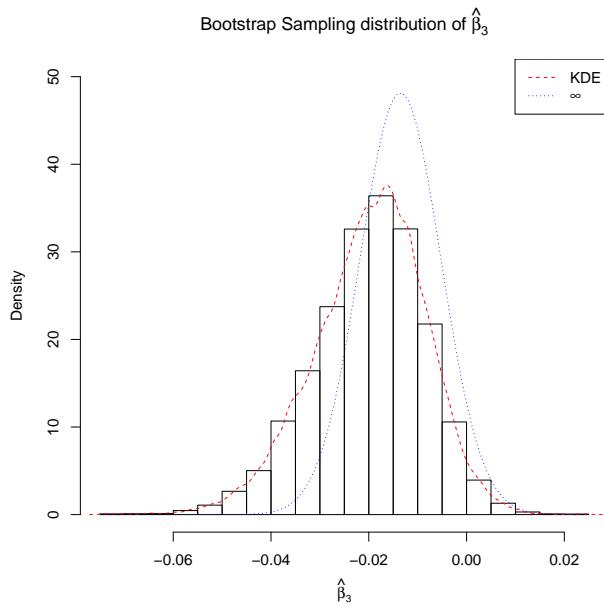
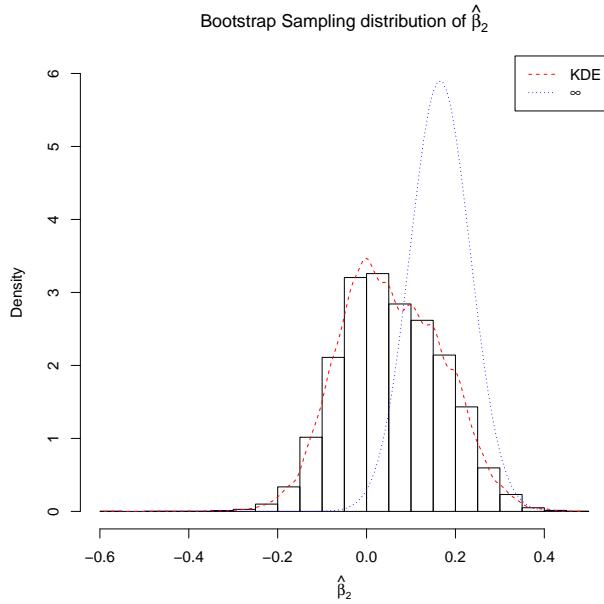
as its diagonal elements.

- The standard error of  $\hat{\beta}_j$  is approximated using

$$[\mathbf{H}^{-1}]_{jj}^{1/2}$$

- So the quality of the  $p$ -value depends on the quality of this approximation.
- Given we have only 30 observation in this dataset, we really should check it.





- Based on what we have seen, we should really use simulation to estimate C.I.

```
> quantile(coeff.sim[1,], probs=c(0.025, 0.975))
```

|      |       |
|------|-------|
| 2.5% | 97.5% |
|------|-------|

```

- 1.841330  4.126274

> quantile(coeff.sim[2,], probs=c(0.025, 0.975))
  2.5%      97.5%
-1.266146  1.352075

> quantile(coeff.sim[3,], probs=c(0.025, 0.975))
  2.5%      97.5%
-0.1481764  0.2750278

> quantile(coeff.sim[4,], probs=c(0.025, 0.975))
  2.5%      97.5%
-0.0442088047  0.0005977476

```

- Suppose we would like to select amongst the following models,

```

> bomber.PS = glm(Damage ~ Type + Month + Load,
+                   family=poisson, data=bomber.df)
>
> bomber.Month.PS =
+   glm(Damage ~ Month,
+       family=poisson, data=bomber.df)
>
> bomber.no.Type.PS =
+   glm(Damage ~ Month + Load,
+       family=poisson, data=bomber.df)
>
> bomber.sq.PS =
+   glm(Damage ~ Type + Month + poly(Load,2),
+       family=poisson, data=bomber.df)

```

- Of course, you would need to be more systematic about it in practice.

```

> varname = names(bomber.df)

> r.vec = varname[1] # You might have transformed it
>
> p.vec = varname[-1]
>
> k = length(p.vec)

> my.func = function(x){
+   # Generate all combinations
+   combn(1:k , x, simplify = FALSE)
+ }
```

```

> # Indices for all possible sub-models
> index = unlist(
+   lapply(1:k, FUN = my.func),
+   recursive = FALSE)

> my.func = function(x){
+   # Create a formula
+   as.formula(
+     paste(r.vec[1],
+           paste(p.vec[x], collapse = "+"),
+           sep = "~"))
+   )
+ }

> # Generate all the formulas for linear terms
> formula.data = lapply(index, FUN = my.func)

> poly = # Create the formula for the poly term
+   update(formula.data[[4]], ~.+poly(Load, 2))
>

> # Final formula set
> formula.data = c(formula.data[c(7,2,6)], poly)

> n.model = length(formula.vec) # number of models

> my.func = function(x){
+   # Run each model on bootstrap data
+   tmp.mat = matrix(nrow = n, ncol = n.model)
+
+   for (i in 1:n.model){
+
+     tmp.PS = glm(formula.data[[i]], family=poisson,
+                   data=bomber.df[,])
+
+     tmp.mat[, i] =
+       predict(tmp.PS, type = "response",
+               newdata = bomber.df)
+   }
+
+   # Return mean square error for each model
+   return(colMeans(
+     (tmp.mat - bomber.df[, "Damage"])^2))
+ }

> M = 10000
>
> n = nrow(bomber.df)

```

```

> index.mat =
+   matrix(sample(1:n, size = M*n, replace = TRUE),
+         nrow = M, ncol = n, byrow = TRUE)

> mse.sim = apply(
+   index.mat, MARGIN = 1, FUN = my.func)

> rowMeans(mse.sim)
[1] 1.440796 1.536930 1.385622 1.532743

```

- Based this the estimated MSE of this bootstrap simulation, we would prefer

bomber.no.Type.PS

## 7 Generalised Linear Model

- There are many similarities in the way that logistic and Poisson regression

$$Y_i \mid \mathbf{X}_i \sim \text{Binomial}(\text{mean} = \text{prob} = m_i, \text{size} = 1)$$

$$Y_i \mid \mathbf{X}_i \sim \text{Poisson}(\text{mean} = \text{var} = m_i)$$

are constructed, in both cases, a parameter of the distribution is linked to

$$m_i = \begin{cases} \frac{\exp(\mathbf{x}_i^T \boldsymbol{\beta})}{1 + \exp(\mathbf{x}_i^T \boldsymbol{\beta})} & \text{for Logistic} \\ \exp(\mathbf{x}_i^T \boldsymbol{\beta}) & \text{for Poisson} \end{cases}$$

predictors via some kind of function so that  $m_i$  is in the right interval, i.e.

$$m_i \in \begin{cases} (0, 1) & \text{Logistic} \\ (0, \infty) & \text{Poisson} \end{cases}$$

- In fact, the two have a deeper connection, and can be unified under one roof.
- The unifying theme is the exponential family of distributions, which leads to  
[generalised linear model \(GLM\)](#)
- By definition, members of the exponential family of distributions all have

$$f_Y(y; \theta, \phi) = \exp \left\{ \frac{y \cdot \theta - b(\theta)}{a(\phi)} + c(y, \phi) \right\}$$

as their probability mass/density function for the random variable  $Y$ , where  
the parameter  $\theta$  is known as the [location parameter](#),  
and  $\phi$  is known as the [dispersion parameter](#).

- The functions  $a$ ,  $b$ , and  $c$  are different for different members.

- The most famous one is when  $\theta = \mu$ ,  $\phi = \sigma^2$ ,  $a(\phi) = \phi$ ,  $b(\theta) = \frac{1}{2}\theta^2$ , and

$$c(y, \phi) = -\frac{1}{2} \left[ \frac{y^2}{\phi^2} + \ln(2\pi\phi^2) \right]$$

Q: What is this famous distribution?

- Now consider the Poisson distribution,

$$f_Y(y; \lambda) = \frac{e^{-\lambda} \lambda^y}{y!}$$

which can be easily put into the following form

$$f_Y(y; \theta, \phi) = \exp \left\{ \frac{y \cdot \theta - b(\theta)}{a(\phi)} + c(y, \phi) \right\} = \exp \left\{ \frac{y \cdot \ln \lambda - \lambda}{1} - \ln y! \right\}$$

- As a result, Poisson can be treated a member of the exponential family with

$$\theta = \ln \lambda, \quad \phi = 1, \quad a(\phi) = 1, \quad b(\theta) = \exp \theta, \quad c(y, \phi) = -\ln y!$$

- One can verify binomial with parameters  $p$  and  $n$  is also a member with

$$\theta = \ln \frac{p}{1-p}, \quad \phi = 1, \quad a(\phi) = 1, \quad b(\theta) = n \ln(1 + e^\theta), \quad c(y, \phi) = -\ln \binom{n}{y}$$

- In general, **GLM** is a model when the following model structure is assumed:

1. There are  $n$  independent responses  $Y_1, Y_2, \dots, Y_n$  with the following means,

$$\mu_1, \mu_2, \dots, \mu_n$$

2. The distributions of  $Y_i$  belong to the same member of the exponential family.

3. The model involves only the **linear predictor**

$$\eta_i = \mathbf{x}_i^T \boldsymbol{\beta} = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_k x_{ik}$$

4. The mean is linked to  $\eta$  via a **strictly monotonic differentiable link function**

$$\eta_i = \mathbf{m}^{-1}(\mu_i) \iff \mu_i = \mathbb{E}[Y_i | \mathbf{X}_i] = m(\mathbf{x}_i^T \boldsymbol{\beta})$$

5. The variance  $\sigma_i^2$  is a function of the mean  $\mu_i$ .

Q: Do you see that linear regression, logistics and Poisson regressions are GLM?

- Amongst the members of the exponential family, the followings

Gamma and Inverse Gaussian,

are often used in regression in addition to normal, binomial and Poisson.

- Gamma regression is originally proposed to model inter-arrival time.

- If some rare event occurs  $\beta$  times on average in some unit time interval, then

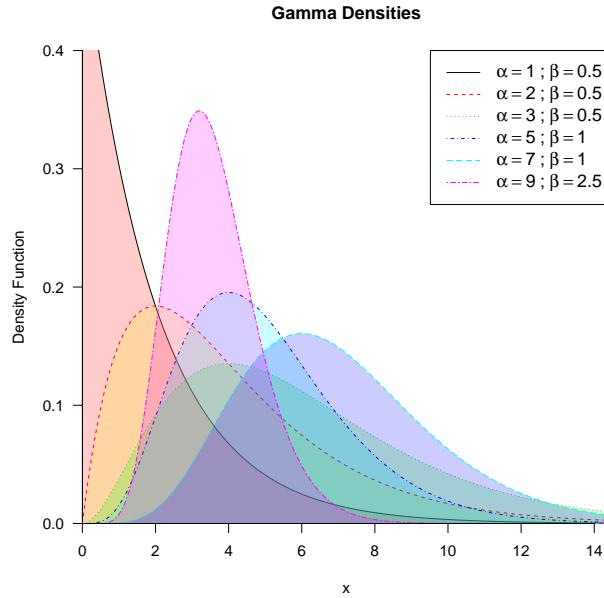
$$y$$

the time interval with  $\alpha$  times occurrence of the same event is known to have

$$Y \sim \text{Gamma}(\alpha, \beta)$$

where  $\alpha$  is known to be the **shape parameter**, and  $\beta$  the **rate parameter**

$$f_Y(y; \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} y^{\alpha-1} e^{-\beta y} \quad \text{for } y, \alpha, \beta > 0$$



- Gamma distribution is a member of the exponential family with

$$\theta = -\frac{\beta}{\alpha}, \quad \phi = \frac{1}{\alpha}, \quad a(\phi) = \phi, \quad b(\theta) = -\ln(-\theta)$$

and

$$c(y, \phi) = \frac{1}{\phi} \ln \frac{1}{\phi} - \ln \Gamma \left( \frac{1}{\phi} \right) + \left( \frac{1}{\phi} - 1 \right) \ln y$$

- It can be shown a gamma distribution comes with the followings

$$\mathbb{E}[Y] = \frac{\alpha}{\beta} = -\frac{1}{\theta} \quad \text{and} \quad \text{Var}[Y] = \frac{\alpha}{\beta^2} = \frac{\phi}{\theta^2} = \phi \mu^2$$

- So, in addition to inter-arrival time, Gamma regression is also used to model  $y \in (0, \infty)$  where the variance is not constant but is proportional to  $\mu^2$ .

- Here  $\alpha$  is assumed to be fixed, which implies the mean is varying through  $\beta$ .
- Inverse Gaussian is also used originally to model time, so similar to Gamma, we have  $y \in (0, \infty)$  but its variance is proportional to the cube of the mean.
- Clearly the choice of the response distribution and the link function is crucial.

### Generalised Linear Model

| Distribution     | Support                      | Canonical Link                                        | Mean function                                   |
|------------------|------------------------------|-------------------------------------------------------|-------------------------------------------------|
| Normal           | $y_i \in (-\infty, \infty)$  | $\eta_i = \mu_i$                                      | $\mu_i = \eta_i$                                |
| Binomial         | $y_i \in \{0, 1\}$           | $\eta_i = \ln \left( \frac{\mu_i}{1 - \mu_i} \right)$ | $\mu_i = \frac{\exp(\eta_i)}{1 + \exp(\eta_i)}$ |
| Poisson          | $y_i \in \{0, 1, 2, \dots\}$ | $\eta_i = \ln(\mu_i)$                                 | $\mu_i = \exp(\eta_i)$                          |
| Gamma            | $y_i \in (0, \infty)$        | $\eta_i = -\mu_i^{-1}$                                | $\mu_i = -\eta_i^{-1}$                          |
| Inverse Gaussian | $y_i \in (0, \infty)$        | $\eta_i = \mu_i^{-2}$                                 | $\mu_i = \eta_i^{-1/2}$                         |

- The choice of the response distribution is often made in terms of what  $Y$  is.

Q: Why generalised linear model? Why exponential family? Why those links?

- Consider the maximum likelihood estimation applied to GLM,

$$\mathcal{L}(\theta; y, \phi) = \exp \left\{ \frac{y \cdot \theta - b(\theta)}{a(\phi)} + c(y, \phi) \right\}$$

and suppose we use the link functions given on the previous slides where

$$\eta_i = \mathbf{x}_i^T \boldsymbol{\beta} = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_k x_{ik}$$

has been linked to  $\mu_i$  via the location parameter  $\theta_i$ , in fact, we have used

$$\eta_i = \theta_i$$

for every case given in the table, which is known as the [canonical link](#).

- This is always possible since the following holds for the exponential family

$$\mu = \mathbb{E}[Y] = b'(\theta) \quad \text{and} \quad \sigma^2 = \text{Var}[Y] = b''(\theta) a(\phi) = \mu' a(\phi)$$

Q: Can you show why using the canonical link simplifies the optimisation

$$\mathbf{X}^T (\mathbf{y} - \boldsymbol{\mu}) = \mathbf{0}$$

where  $\mathbf{X}$  and  $\mathbf{y}$  are the usual data matrix and response vector, and

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix} = \begin{bmatrix} b'(\theta_1) \\ b'(\theta_2) \\ \vdots \\ b'(\theta_n) \end{bmatrix} = \begin{bmatrix} b'(\mathbf{x}_1^T \boldsymbol{\beta}) \\ b'(\mathbf{x}_2^T \boldsymbol{\beta}) \\ \vdots \\ b'(\mathbf{x}_n^T \boldsymbol{\beta}) \end{bmatrix}$$

- Furthermore, the Hessian matrix, which is used for asymptotic inference, is

$$\mathbf{H} = \frac{\mathbf{X}^T \mathbf{V} \mathbf{X}}{[a(\phi)]^2}$$

where  $\mathbf{V} = \text{diag}\{\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2\}$  is a function of  $\mu_i$ .

- If we do not use the canonical link, then the score equation becomes

$$\mathbf{X}^T \mathbf{U} (\mathbf{y} - \boldsymbol{\mu}) = \mathbf{0}$$

where  $\mathbf{U} = \text{diag}\left\{\frac{d\theta_1}{d\eta_1}, \frac{d\theta_2}{d\eta_2}, \dots, \frac{d\theta_n}{d\eta_n}\right\}$ , and the Hessian becomes

$$\mathbf{H} = \frac{\mathbf{X}^T \mathbf{U} \mathbf{V} \mathbf{U} \mathbf{X}}{[a(\phi)]^2}$$

Q: Why do you think we prefer using the canonical link?

- However, there are cases that we have to use a non-canonical link and solve

$$\mathbf{X}^T \mathbf{U} (\mathbf{y} - \boldsymbol{\mu}) = \mathbf{0}$$

Q: Do you see any issue regarding using the canonical link for Gamma?

$$\eta_i = \theta_i \implies \eta_i = -\mu_i^{-1} \implies \mathbf{x}_i^T \boldsymbol{\beta} = -\frac{\beta}{\alpha}$$

- Usually, the canonical link for binomial and Poisson are used, however,

$$\eta_i = \mu_i^{-1} \quad \text{or} \quad \eta_i = \ln(\mu_i)$$

is often used as the link function for Gamma instead of the canonical link

$$\eta_i = -\mu_i^{-1}$$

- Consider the following, where  $Y$  is inverse linked with the linear predictor

```
> N = 100
> x = runif(N, 0, 1)
> a = 0.5
> b = 1.2
> eta = a + b * x
> shape = 10

> rate = shape*eta # This is for inverse link
> y = rgamma(N, rate = rate, shape = shape)
```

- In R, the default link function is the inverse link, so `family = Gamma` is used

$$\eta_i = \mu_i^{-1}$$

```
> m_glm = glm(y ~ x, family = Gamma) # default link
> coef(m_glm)
```

| (Intercept) | x         |
|-------------|-----------|
| 0.5368359   | 1.1758975 |

- The following  $Y$  is log linked with the linear predictor, so we must specify it

```
> rate = shape / exp(eta)
> y = rgamma(N, rate = rate, shape = shape)
> m_glm = glm(y ~ x, family = Gamma(link = "log"))
> coef(m_glm)
```

| (Intercept) | x         |
|-------------|-----------|
| 0.4823127   | 1.2165194 |

- Whether to use the inverse link or the log link

$$\begin{cases} \mu_i = (\mathbf{x}_i^T \boldsymbol{\beta})^{-1} & \text{inverse link} \\ \mu_i = \exp(\mathbf{x}_i^T \boldsymbol{\beta}) & \text{log link} \end{cases}$$

in a gamma regression depends on the data

- If the inverse link is appropriate, then a plot of

$$x_{ij} \text{ Vs } y_i^{-1}$$

should be roughly linear.

- If the log link is appropriate, then a plot of

$$x_{ij} \text{ Vs } \ln(y_i)$$

should be roughly linear.

Q: What is the difference between a linear regression with  $z_i = \ln(y_i)$  being the response and a gamma regression (log link) with  $y_i$  being the response?

## 8 Generalised Estimating Equation

- One of the important assumptions in GLM is independence:

- There are  $n$  independent responses  $Y_1, Y_2, \dots, Y_n$  with the following means,

$$\mu_1, \mu_2, \dots, \mu_n$$

However, there are many situations in which repeated response observations are made, and duplication is valuable but can badly violate independence.

- The clustering of response observations that results in the violation of the independence assumption can come from various sources. e.g.

1. The amount of wearing and tearing in tires from the same vehicle
2. The height/weight of the same person over time
3. The price of some identical object
4. Repeated measurements on the same object in the same experiment

- For example, consider the following dataset,

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <b>Ltemperature</b> | Lamination temperature choice                   |
| <b>Ltime</b>        | Lamination time choice                          |
| <b>Lpressure</b>    | Lamination pressure choice                      |
| <b>Ftemperature</b> | Firing temperature choice                       |
| <b>Ftime</b>        | Firing cycle time choice                        |
| <b>Fpoint</b>       | Firing dew point choice                         |
| <b>Run</b>          | 1 of 4 repeated measurements on the same device |
| <b>Camber</b>       | Curvature of the substrate devices produced     |

which is from an experiment on devices produced in a semiconductor plant.

- It is clear that the measurement taken on the same device may be correlated.

```
> semiconductor.df = read.table(header = TRUE,
+   file = "~/Desktop/semiconductor.txt",
+   colClasses = c(rep("factor", 7), "numeric"))
>
> str(semiconductor.df, vec.len = 3)
```

```
'data.frame': 64 obs. of 8 variables:
$ Ltemperature: Factor w/ 2 levels "0","1": 1 2 1 2 1 2 1 2 ...
$ Ltime       : Factor w/ 2 levels "0","1": 1 1 2 2 1 1 2 2 ...
$ Lpressure   : Factor w/ 2 levels "0","1": 1 1 1 1 2 2 2 ...
$ Ftemperature: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 ...
$ Ftime       : Factor w/ 2 levels "0","1": 1 2 2 1 2 1 1 2 ...
$ Fpoint      : Factor w/ 2 levels "0","1": 1 2 1 2 2 1 2 1 ...
$ Run         : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 ...
$ Camber      : num  0.0167 0.0062 0.0041 0.0073 0.0047 0.0219 0.0121 0.0255 ...
```

```
> head(dplyr::arrange_all(semiconductor.df), 8)
```

|   | Ltemperature | Ltime | Lpressure | Ftemperature | Ftime | Fpoint | Run | Camber |
|---|--------------|-------|-----------|--------------|-------|--------|-----|--------|
| 1 | 0            | 0     | 0         | 0            | 0     | 0      | 1   | 0.0167 |
| 2 | 0            | 0     | 0         | 0            | 0     | 0      | 2   | 0.0128 |
| 3 | 0            | 0     | 0         | 0            | 0     | 0      | 3   | 0.0149 |
| 4 | 0            | 0     | 0         | 0            | 0     | 0      | 4   | 0.0185 |
| 5 | 0            | 0     | 0         | 1            | 0     | 1      | 1   | 0.0032 |
| 6 | 0            | 0     | 0         | 1            | 0     | 1      | 2   | 0.0023 |
| 7 | 0            | 0     | 0         | 1            | 0     | 1      | 3   | 0.0077 |
| 8 | 0            | 0     | 0         | 1            | 0     | 1      | 4   | 0.0069 |

- In general, we are dealing with the following data structure,

Cluster

$$\begin{array}{ccccccc}
 1 & \mathbf{y}_1 & \mathbf{x}_{11} & \mathbf{x}_{12} & \cdots & \mathbf{x}_{1k} \\
 2 & \mathbf{y}_2 & \mathbf{x}_{21} & \mathbf{x}_{22} & \cdots & \mathbf{x}_{2k} \\
 \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
 s & \mathbf{y}_s & \mathbf{x}_{s1} & \mathbf{x}_{s2} & \cdots & \mathbf{x}_{sk}
 \end{array}$$

where the vector  $\mathbf{x}_{\ell j} \in \mathbb{R}^n$  consists of  $n$  values of the  $j$ th predictor on the  $\ell$ th cluster, and  $\mathbf{y}_\ell \in \mathbb{R}^n$  consists of  $n$  response values on the  $\ell$ th cluster.

- So it can be thought that the dataset has two layers, with  $n \cdot s$  number of observations in total, independence holds in the 1st layer, but not the 2nd.
- The response value is assumed to follow a distribution from the exponential family, but unlike GLM, now we have to model the within-cluster correlation.
- Let us start with normal distribution with identity link  $\eta_i = \mu_i$ , then

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_s \end{bmatrix} = \begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \\ \vdots \\ \mathbf{X}_s \end{bmatrix} \boldsymbol{\beta} + \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \vdots \\ \mathbf{e}_s \end{bmatrix} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}, \quad \text{where } \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}$$

and  $\mathbf{X}_\ell$  is an  $n \times (k+1)$  matrix consists of  $\mathbf{X}_\ell = [\mathbf{1} \ \mathbf{x}_{\ell 1} \ \mathbf{x}_{\ell 2} \ \cdots \ \mathbf{x}_{\ell k}]$ .

- Essentially, within each cluster, the response vector

$$\mathbf{y}_\ell = \mathbf{X}_\ell \boldsymbol{\beta} + \mathbf{e}_\ell$$

is assumed to follow  $\text{Normal}(\mathbf{X}_\ell \boldsymbol{\beta}, \mathbf{V}_\ell)$ , where  $\mathbf{V}_\ell$  is an  $n \times n$  matrix.

- Since independence is assumed between clusters, the 1st layer, we have

$$\mathbf{V} = \text{diag}\{\mathbf{V}_1, \mathbf{V}_2, \dots, \mathbf{V}_s\}$$

a block diagonal matrix, as the variance-covariance matrix for  $\mathbf{y}$  overall.

- Typically, the same within-cluster correlation structure is assumed for all  $\ell$ .
- To understand this assumption, recall a correlation matrix of consists of

$$r_{ii'} = \text{corr}([\mathbf{Z}]_i, [\mathbf{Z}]_{i'}) = \frac{\text{Cov}([\mathbf{Z}]_i, [\mathbf{Z}]_{i'})}{\sigma_i \sigma_{i'}} = \frac{\sigma_{ii'}}{\sigma_i \sigma_{i'}}$$

for any random vector  $\mathbf{Z}$ , and can be written using matrix notation

$$\mathbf{R} = \text{Corr}(\mathbf{Z}) = [\text{diag}(\boldsymbol{\Sigma})]^{-1/2} \boldsymbol{\Sigma} [\text{diag}(\boldsymbol{\Sigma})]^{-1/2}$$

- Thus the variance-covariance matrix of  $\mathbf{Z}$  in general can be defined as

$$\boldsymbol{\Sigma} = [\text{diag}(\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2)]^{1/2} \mathbf{R} [\text{diag}(\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2)]^{1/2}$$

that is, the correlation structure is defined by the correlation matrix  $\mathbf{R}$ .

- So the same within-cluster correlation structure means having the same  $\mathbf{R}_*$

$$\mathbf{V}_\ell = \mathbf{A}_\ell^{1/2} \mathbf{R}_* \mathbf{A}_\ell^{1/2} \quad \text{where } \mathbf{A}_\ell = \text{diag}(\sigma_{\ell 1}^2, \sigma_{\ell 2}^2, \dots, \sigma_{\ell n}^2)$$

- When the response is assumed to follow a normal distribution, then

$$\mathbf{A}_\ell = \text{diag}(\sigma^2, \sigma^2, \dots, \sigma^2) = \sigma^2 \mathbf{I} \implies \mathbf{V}_\ell = \mathbf{A}_\ell^{1/2} \mathbf{R}_* \mathbf{A}_\ell^{1/2} = \sigma^2 \mathbf{R}_*$$

since the variance is a constant across all  $n \cdot s$  observations in this case.

- Therefore, we essentially have the following model,

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \quad \text{where } \boldsymbol{\varepsilon} \sim \text{Normal}(\mathbf{0}, \mathbf{V})$$

and the overall variance-covariance matrix is given by

$$\mathbf{V} = \text{diag}\{\mathbf{V}_1, \mathbf{V}_2, \dots, \mathbf{V}_s\} = \sigma^2 \text{diag}\{\mathbf{R}_*, \mathbf{R}_*, \dots, \mathbf{R}_*\} = \sigma^2 \mathbf{R}$$

Q: Have you seen something similar to this model before?

Q: How did we estimate  $\boldsymbol{\beta}$  in those models similar to the one above?

Q: Can we use the same idea here? How about nonnormal cases?

- In general, a normal likelihood function in matrix notation is given by

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = ((2\pi)^n \det \boldsymbol{\Sigma})^{-1/2} \exp \left[ -\frac{1}{2} (\mathbf{y} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{y} - \boldsymbol{\mu}) \right]$$

- To model heteroskedasticity, we have considered

$$\begin{aligned} \hat{\boldsymbol{\beta}} &= \arg \max_{\mathbf{b}} \{\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma})\} \\ &= \arg \min_{\mathbf{b}} \left\{ (\mathbf{y} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{y} - \boldsymbol{\mu}) \right\} \\ &= \arg \min_{\mathbf{b}} \left\{ (\mathbf{y} - \mathbf{X}\mathbf{b})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{y} - \mathbf{X}\mathbf{b}) \right\} \\ &= (\mathbf{X}^\top \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{W} \mathbf{y} \end{aligned}$$

where  $\boldsymbol{\Sigma}^{-1} = \mathbf{W} = \text{diag}(1/\sigma_1^2, 1/\sigma_2^2, \dots, 1/\sigma_n^2)$  is a known matrix.

Q: How about AR(1), do you remember what we did?

- We essentially face the same optimisation problem without independence

$$\hat{\boldsymbol{\beta}} = \arg \max_{\mathbf{b}} \{\mathcal{L}(\mathbf{X}\mathbf{b}, \boldsymbol{\Sigma})\}$$

but the variance-covariance matrix is no longer a diagonal matrix, e.g. AR(1)

$$\boldsymbol{\Sigma} = \frac{\sigma^2}{1 - \rho^2} \begin{bmatrix} 1 & \rho & \cdots & \rho^{n-1} \\ \rho & 1 & \cdots & \rho^{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ \rho^{n-1} & \rho^{n-2} & \cdots & 1 \end{bmatrix}$$

- If  $\sigma^2$  and  $\rho$  are given, the optimisation can be solved directly,

$$\begin{aligned}\hat{\beta} &= \arg \max_{\mathbf{b}} \{\mathcal{L}(\mathbf{Xb}, \Sigma)\} = \arg \min_{\mathbf{b}} \left\{ (\mathbf{y} - \mathbf{Xb})^T \Sigma^{-1} (\mathbf{y} - \mathbf{Xb}) \right\} \\ &= \left( \mathbf{X}^T \Sigma^{-1} \mathbf{X} \right)^{-1} \mathbf{X}^T \Sigma^{-1} \mathbf{y}\end{aligned}$$

if not, then an iterative optimisation procedure is needed to find MLE.

- The same can be said regarding this newish model under GEE,

$$\mathbf{Y} = \mathbf{X}\beta + \varepsilon \quad \text{where} \quad \varepsilon \sim \text{Normal}(\mathbf{0}, \mathbf{V})$$

- When the overall variance-covariance matrix is known

$$\mathbf{V} = \sigma^2 \mathbf{R}$$

that is,  $\sigma^2$  and  $\mathbf{R}$  are known or predetermined, then

$$\begin{aligned}\hat{\beta} &= \arg \max_{\mathbf{b}} \{\mathcal{L}(\mathbf{Xb}, \mathbf{V})\} = \arg \min_{\mathbf{b}} \left\{ (\mathbf{y} - \mathbf{Xb})^T \mathbf{V}^{-1} (\mathbf{y} - \mathbf{Xb}) \right\} \\ &= \left( \mathbf{X}^T \mathbf{R}^{-1} \mathbf{X} \right)^{-1} \mathbf{X}^T \mathbf{R}^{-1} \mathbf{y}\end{aligned}$$

- Of course, one usually does not know what the true correlation structure is, thus some structure needs to be specified, and optimisation done iteratively.
- There is no question the correlation structure matters, however, it turns out GEE provides a consistent estimator for  $\beta$  even if the structure is incorrect.
- The followings are some common specifications:

$$\text{Independent} \quad \mathbf{R} = \mathbf{I}$$

$$\text{Exchangeable} \quad [\mathbf{R}]_{ii'} = \begin{cases} 1 & i = i', \\ r & i \neq i'. \end{cases}$$

$$\text{1-dependent} \quad [\mathbf{R}]_{ii'} = \begin{cases} 1 & i = i', \\ r & |i - i'| = 1, \\ 0 & \text{otherwise.} \end{cases}$$

$$\text{AR}(1) \quad [\mathbf{R}]_{ii'} = r^{|i - i'|} \quad \text{for all } i, i'$$

- GEE for normally distributed responses is special in the sense that its estimating equations

are not usually emphasised

$$\mathbf{X}^T \mathbf{V}^{-1} (\mathbf{y} - \boldsymbol{\mu}) = \mathbf{0}$$

since it is understood as a result of solving the optimisation iteratively.

Q: The formulation for the use of GEE for nonnormal  $Y_i$  is very similar, but the estimation philosophy is very different. Can you see the potential difference?

- The first difference is the variance-covariance matrix

$$\mathbf{V} = \text{diag}\{\mathbf{V}_1, \mathbf{V}_2, \dots, \mathbf{V}_s\}$$

although it is still block diagonal,  $\mathbf{V}_\ell$ 's are different from cluster to cluster.

$$\mathbf{V}_\ell = \mathbf{A}_\ell^{1/2} \mathbf{R}_* \mathbf{A}_\ell^{1/2} \quad \text{where } \mathbf{A}_\ell = \text{diag}(\sigma_{\ell 1}^2, \sigma_{\ell 2}^2, \dots, \sigma_{\ell n}^2)$$

- Note  $\mathbf{A}_\ell$  depends on the relationship between the variance and the mean,

$$\sigma_{\ell i}^2 = \begin{cases} \mu_{\ell i} (1 - \mu_{\ell i}) & \text{Binomial} \\ \mu_{\ell i} & \text{Poisson} \\ \phi \mu_{\ell i}^2 & \text{Gamma} \end{cases}$$

- As before, once we have to specify a form for  $\mathbf{R}_*$ , then we have

$$\mathbf{V}_\ell = \mathbf{A}_\ell^{1/2} \mathbf{R}_* \mathbf{A}_\ell^{1/2} \implies \mathbf{V} = \begin{bmatrix} \mathbf{V}_1 & & & \\ & \mathbf{V}_2 & & \\ & & \ddots & \\ & & & \mathbf{V}_s \end{bmatrix}$$

which now not only depends on the unknown parameter  $r$ , but also the mean

$$\boldsymbol{\mu} = m(\mathbf{x}^T \boldsymbol{\beta})$$

Q: How to find an estimate of  $\boldsymbol{\beta}$  under this model? What would you do next?

- If a nonnormal marginal distribution is assumed for  $Y_i$ , the joint distribution for  $\mathbf{Y}$  without independence is not available, so MLE cannot be found here!
- GEE in general on the other hand solve the following iteratively instead

$$\hat{\boldsymbol{\beta}} = \arg \min_{\mathbf{b}} \left\{ (\mathbf{y} - \boldsymbol{\mu})^T \mathbf{V}^{-1} (\mathbf{y} - \boldsymbol{\mu}) \right\}$$

which leads to the following estimating equation

$$\mathbf{J}^T \mathbf{V}^{-1} (\mathbf{y} - \boldsymbol{\mu}) = \mathbf{0}$$

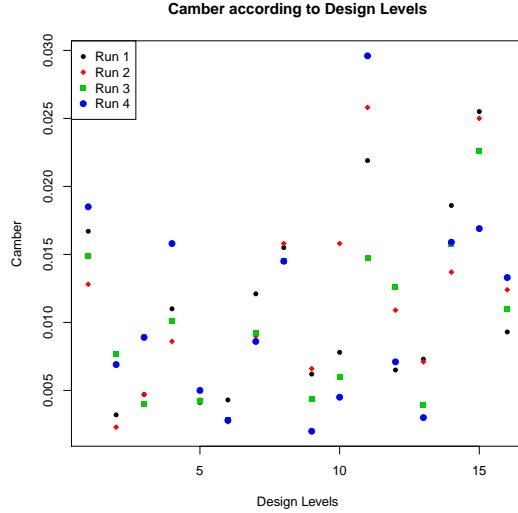
where elements of the vector  $\boldsymbol{\mu}$  and the matrix  $\mathbf{J}$  are given by the followings

$$[\boldsymbol{\mu}]_i = m(\mathbf{x}_i^T \mathbf{b}) \quad \text{and} \quad [\mathbf{J}]_{ij} = \frac{\partial m}{\partial b_j} \Big|_{\mathbf{x}_i, \mathbf{b}} \quad \text{for} \quad \begin{array}{ll} i = 1, 2, \dots, n \cdot s \\ j = 0, 1, \dots, k \end{array}$$

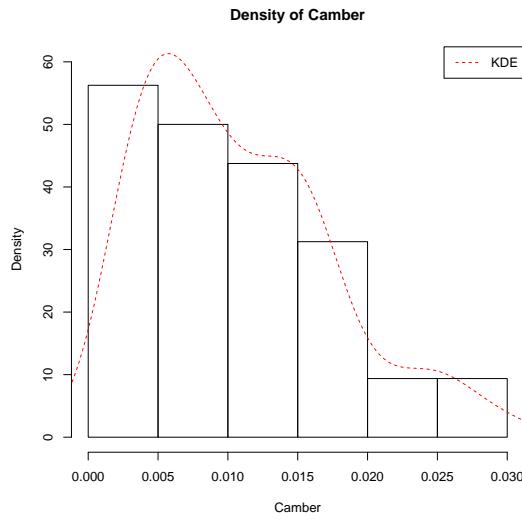
- The function  $m(\eta)$  denotes the mean function in the corresponding GLM.

Q: Have you seen something similar to this in this course?

- As we suspected, the data indicates within-cluster correlation.



- It seems  $Y_i$  is nonnormal with a heavy right-tailed distribution.



- Based on what we have seen, a gamma response is assumed with a log link,

$$\begin{aligned} \ln \mu = & \beta_0 + \beta_1 L_{\text{temperature}} + \beta_2 L_{\text{time}} + \beta_3 L_{\text{pressure}} \\ & + \beta_4 F_{\text{temperature}} + \beta_5 F_{\text{time}} + \beta_6 F_{\text{point}} \end{aligned}$$

- In R, generalised estimating equations are solved numerically by the following

```
> library(gee)
>
> semi.GEE =
+   gee(Camber~.-Run,
+         family = Gamma(link = "log"),
+         id = Run,
+         corstr = "AR-M", M = 1,
+         data = semiconductor.df)
```

| Coefficients: |               |             |            |
|---------------|---------------|-------------|------------|
| (Intercept)   | Ltemperature1 | Ltime1      | Lpressure1 |
| -4.57245347   | 0.35862963    | 0.01178501  | 0.59939347 |
| Ftemperature1 | Ftime1        | Fpoint1     |            |
| -0.04090419   | -0.39396896   | -0.76706214 |            |

```
> semi.GEE
```

```
Model:
Link: Logarithm
Variance to Mean Relation: Gamma
Correlation Structure: AR-M , M = 1

Call:
gee(formula = Camber ~ . - Run, id = Run, data = semiconductor.df,
      family = Gamma(link = "log"), corstr = "AR-M", Mv = 1)

Coefficients:
(Intercept) Ltemperature1 Ltime1 Lpressure1
-4.57245347 0.35862963 0.01178501 0.59939347

Ftemperature1 Ftime1 Fpoint1
-0.04090419 -0.39396896 -0.76706214

Estimated Scale Parameter: 0.1381516
Number of Iterations: 7

Working Correlation[1:4,1:4]
 [,1]      [,2]      [,3]      [,4]
 [1,] 1.000000000 -0.12867472 0.01655718 -0.002130491
 [2,] -0.128674720 1.00000000 -0.12867472 0.016557184
 [3,] 0.016557184 -0.12867472 1.00000000 -0.128674720
 [4,] -0.002130491 0.01655718 -0.12867472 1.000000000
```

- It turns out the correlation is not that strong!

```
> semi.GLM =
+   glm(Camber~.-Run,
+         family = Gamma(link = "log"),
+         data = semiconductor.df)
>
> summary(semi.GLM)
```

| Coefficients:                                                |                                               |            |         |              |  |
|--------------------------------------------------------------|-----------------------------------------------|------------|---------|--------------|--|
|                                                              | Estimate                                      | Std. Error | t value | Pr(>t)       |  |
| (Intercept)                                                  | -4.57340                                      | 0.12203    | -37.478 | < 2e-16 ***  |  |
| Ltemperature1                                                | 0.36117                                       | 0.09225    | 3.915   | 0.000244 *** |  |
| Ltime1                                                       | 0.01262                                       | 0.09225    | 0.137   | 0.891702     |  |
| Lpressure1                                                   | 0.59995                                       | 0.09225    | 6.504   | 2.13e-08 *** |  |
| Ftemperature1                                                | -0.04702                                      | 0.09225    | -0.510  | 0.612188     |  |
| Ftime1                                                       | -0.39334                                      | 0.09225    | -4.264  | 7.63e-05 *** |  |
| Fpoint1                                                      | -0.75052                                      | 0.09225    | -8.136  | 4.09e-11 *** |  |
| ---                                                          |                                               |            |         |              |  |
| Signif. codes:                                               | 0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?. 0.1 ? ? 1 |            |         |              |  |
| (Dispersion parameter for Gamma family taken to be 0.136147) |                                               |            |         |              |  |

```

Null deviance: 26.5188 on 63 degrees of freedom
Residual deviance: 8.1949 on 57 degrees of freedom
AIC: -540.04

Number of Fisher Scoring iterations: 8

```

## 9 Nonparametric

### 9.1 Simple Smoothing

- Suppose we have  $n$  observations of

$$(x_i, y_i)$$

and we return to our assumption of additive relationship,

$$Y_i = m(x_i) + \varepsilon_i$$

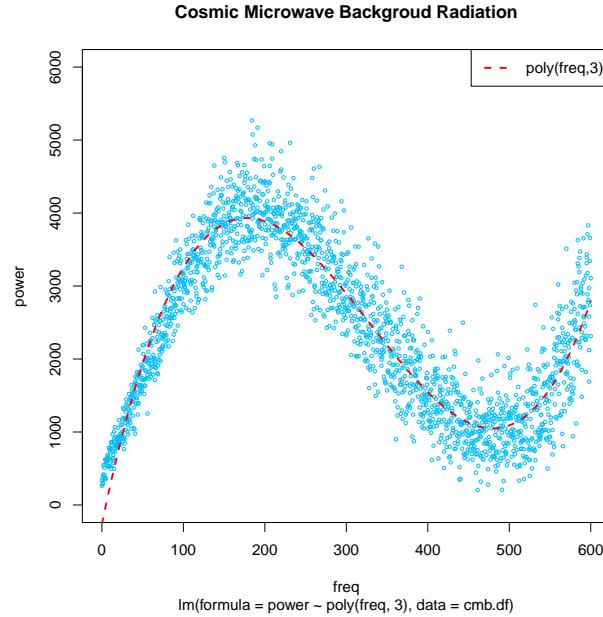
where the error has zero mean, that is, the conditional mean of  $Y$  is given by

$$\mathbb{E}[Y | X = x] = m(x)$$

- This might seem to be reminiscent of nonlinear regression model,

$$Y_i = m(\mathbf{x}_i, \boldsymbol{\beta}) + \varepsilon_i \quad \text{where } \varepsilon_i \sim \text{Normal}(0, \sigma^2)$$

- However, there are two differences in setup to nonlinear regression:
  1. Modelling only one independent variable instead of  $k$  of them
  2. The form of  $m(x)$  not being assumed or given beforehand



- The goal is to gather information from the data to estimate the form of

$$\mathbb{E}[Y | X = x] = m(x)$$

because there is no clear reason why a cubic polynomial should be chosen.

- So it is not about estimating parameters in some parametric form of  $m$ , e.g.

$$m(x; \beta_1, \beta_2) = \frac{\beta_1 + \beta_2 x}{\exp(\beta_1 + \beta_2 x)}$$

Q: How can we estimate the form of association between  $x$  and  $m(x)$ ?

- Since no parametric form is assumed, the function

$$\hat{m}(x)$$

which is used to estimate the  $m(x)$  cannot possess a closed form.

- Usually  $\hat{m}(x)$  is in the form of an algorithm, nevertheless a proper function!
- One naive way of constructing a nonparametric estimator

$$\hat{m}(x)$$

is to use step-functions, i.e. allowing only a finite number of values for  $m(x)$ .

- That is, without loss of generality, assume  $X$  takes values over  $[0, 1]$ , which can be partitioned into  $L$  subintervals of the same length for a chosen  $L$ :

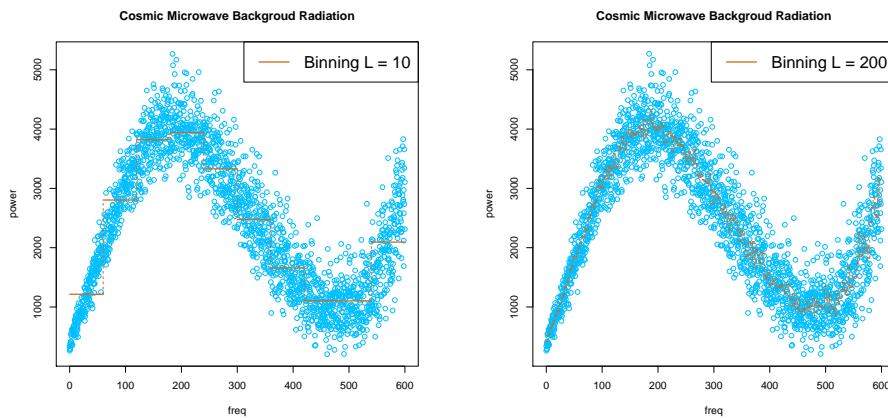
$$\mathcal{B}_1 = \left[0, \frac{1}{L}\right), \mathcal{B}_2 = \left[\frac{1}{L}, \frac{2}{L}\right), \dots, \mathcal{B}_L = \left[\frac{L-1}{L}, 1\right]$$

- We estimate  $m(x)$  for  $x \in \mathcal{B}_\ell$  using the corresponding sub-sample mean of  $Y$

$$\hat{m}_L(x) = \frac{\sum_{i=1}^n y_i \cdot \mathbb{I}(x_i \in \mathcal{B}_\ell)}{\sum_{i=1}^n \mathbb{I}(x_i \in \mathcal{B}_\ell)}$$

this is known as [binning](#) or [bin-smoothing](#).

- This is a very simple  $\hat{m}(x)$  but can cope with complicated relationships,



however, it always has discontinuities at the boundaries.

- Another simple construction for  $\hat{m}(x)$  that involves only sub-sample means is **simple moving averages** (SMA), which also has discontinuities.
- Without loss of generality, suppose the data  $(x_i, y_i)$  are sorted such that

$$x_{i+1} \geq x_i \quad \text{for all } i$$

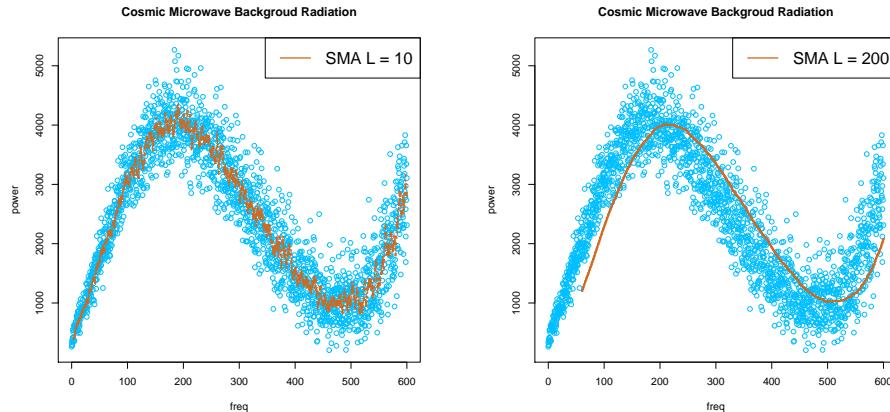
if there is a tie, then  $y$  are used to break the tie according to some order.

- For a chosen  $L \geq 1$ , we estimate  $m(x)$  using

$$\hat{m}_L(x) = \frac{1}{L} \sum_{\ell=0}^{L-1} y_{i-\ell}, \quad \text{for } x \in [x_i, x_{i+1}) \text{ where } L \leq i \leq n-1$$

that is, the sub-sample mean of  $Y$  corresponding to  $x_i, x_{i-1}, \dots, x_{i-L+1}$ .

- Note SMA is typically used for times series, it forms  $n - L$  bins, each has a fixed number of observations, rather than fixed-width bins with a variable number of observations.
- SMA can produce a more smooth looking curve than binning:



however, SMA results a shift in  $x$  for a large  $L$  due to using only "past" data.

- For non-time series data, often **simple central moving averages** (SCMA)

$$\hat{m}_L(x) = \frac{1}{2L+1} \left( y_i + \sum_{\ell=1}^L y_{i-\ell} + y_{i+\ell} \right)$$

is used to estimate  $m(x)$  for

$$x \in \left[ \frac{x_{i-1} + x_i}{2}, \frac{x_i + x_{i+1}}{2} \right]$$

where

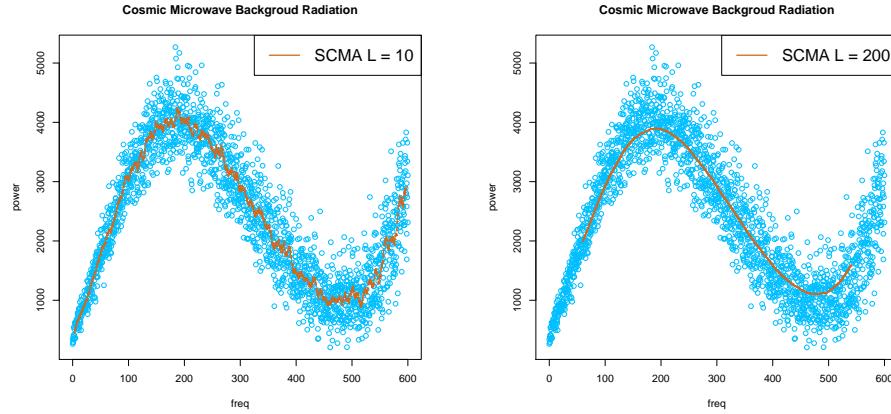
$$L+1 \leq i \leq n-L-1$$

- Notice  $\hat{m}_L(x)$  is undefined for  $x < x_L$  or  $x > x_n$  when SMA is used, while

$$\frac{x_L + x_{L+1}}{2} \leq x < \frac{x_{n-L-1} + x_{n-L}}{2}$$

is the domain of  $\hat{m}_L(x)$  when SCMA is used, and which is in  $n - 2L$  bins.

- SCMA avoids the shift in  $x$  by using data on both sides



however, SCMA provides no estimated value of  $m(x)$  at two endpoints of  $x$ .

- More sophisticated moving averages methods address the values of  $m(x)$  at the two ends by defining a neighbourhood  $N_i$  around  $x_i$ , e.g., a simple one is

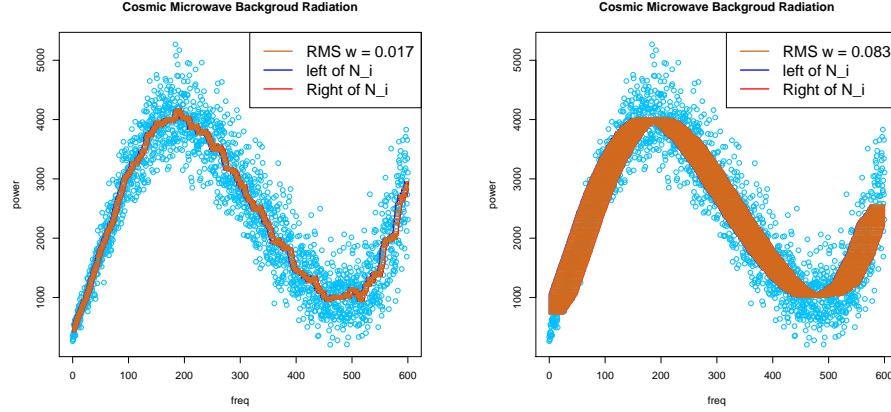
$$N_i = \left\{ \max \left( i - \frac{\lfloor wn \rfloor - 1}{2}, 1 \right), \dots, i - 1, i, i + 1, \dots, \min \left( i + \frac{\lfloor wn \rfloor - 1}{2}, n \right) \right\}$$

where the number  $w \in (0, 1)$  is chosen such that the floor  $\lfloor wn \rfloor$  is odd.

- It extends "coverage" to the two ends by truncating the number of points in the bins near the two ends, this is known as [running mean smoothing](#) (RMS).

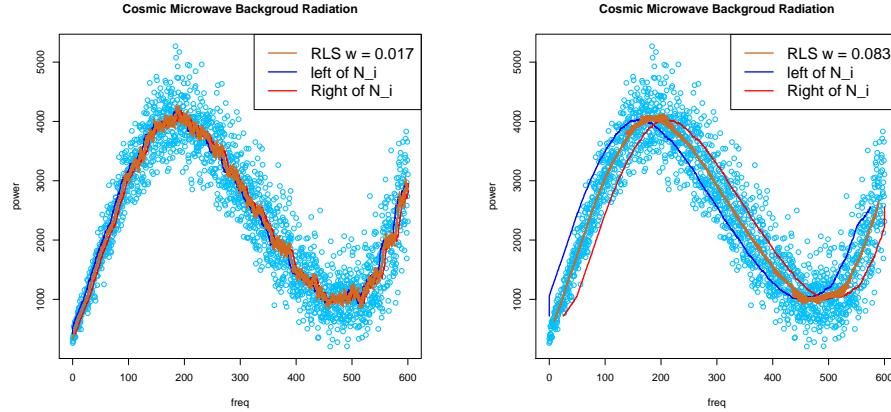
Q: Suppose  $n = 10$  and  $w = 0.5$ , how many data points are in  $N_1$ ,  $N_2$  and  $N_3$ ?

- RMS tends to wiggle near the center of the data,



however, flatten out near the boundary of the data.

- We could improve RMS by fitting a sloping line rather than a horizontal line to  $N_i$



this is known as [running line smoothing](#) (RLS).

- A common issue with those simple smoothers is that they are not smooth!
- One way to address is to use a [linear spline](#) rather than a least squares line.
- A spline function consists of piecewise polynomial on subintervals

$$S(x) = \begin{cases} p_0(x) = a_0 + b_0x & \text{for } \xi_0 \leq x < \xi_1 \\ \vdots & \vdots \\ p_i(x) = a_i + b_i x & \text{for } \xi_i \leq x < \xi_{i+1} \\ \vdots & \vdots \\ p_\ell(x) = a_\ell + b_\ell x & \text{for } \xi_\ell \leq x \leq \xi_{\ell+1} \end{cases}$$

where the coefficients are chosen such that the function  $S(x)$  is continuous.

Q: What is the form of the coefficients? How to construct a quadratic spline?

|   |    |   |   |
|---|----|---|---|
| x | -1 | 0 | 1 |
| y | 0  | 1 | 3 |

- Alternatively, this arbitrary function  $m$  can be modelled using [cubic spline](#),

$$\hat{m}(x) = \begin{cases} \hat{m}_0(x) = a_0 + b_0x + c_0x^2 + d_0x^3 & \text{for } \xi_0 \leq x < \xi_1 \\ \vdots & \vdots \\ \hat{m}_i(x) = a_i + b_ix + c_ix^2 + d_ix^3 & \text{for } \xi_i \leq x < \xi_{i+1} \\ \vdots & \vdots \\ \hat{m}_\ell(x) = a_\ell + b_\ell x + c_\ell x^2 + d_\ell x^3 & \text{for } \xi_\ell \leq x \leq \xi_{\ell+1} \end{cases}$$

where  $\xi_0 \leq \xi_1 \leq \dots \leq \xi_\ell \leq \xi_{\ell+1}$  is a set of ordered points, so-called [knots](#).

- In general, a cubic spline is a continuous function in  $(\xi_0, \xi_{\ell+1})$ , that is

$$\hat{m}_i(\xi_i) = \lim_{x \rightarrow \xi_i^-} \hat{m}_{i-1}(x) = \hat{m}_{i-1}(\xi_i^-) \quad \text{for } i = 1, 2, \dots, \ell$$

and has continuous first and second derivatives, that is,

$$\hat{m}'_i(\xi_i^+) = \hat{m}'_{i-1}(\xi_i^-) \quad \text{and} \quad \hat{m}''_i(\xi_i^+) = \hat{m}''_{i-1}(\xi_i^-)$$

- Given  $n$  observations of

$$(x_i, y_i) \quad i = 1, 2, \dots, n$$

- The coefficients  $\{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}\}$  are chosen at the observed data only, i.e.

$$\xi_0 = x_1, \quad \xi_i = x_{i+1} \quad \text{for } i = 1, 2, \dots, n-2, \quad \xi_{\ell+1} = x_n$$

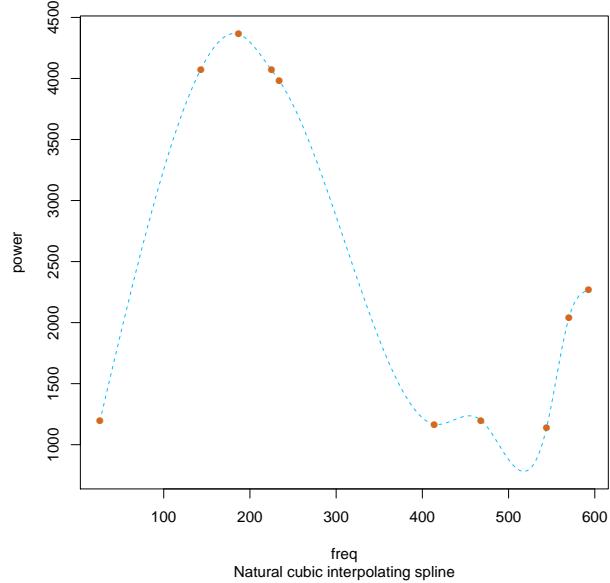
and obtain the set of coefficients by solving the following set of equations,

$$\begin{aligned} a_i + b_i x_i + c_i x_i^2 + d_i x_i^3 &= y_i \\ a_{i-1} + b_{i-1} x_i + c_{i-1} x_i^2 + d_{i-1} x_i^3 &= y_i \\ b_i + 2c_i x_i + 3d_i x_i^2 &= b_{i-1} + 2c_{i-1} x_i + 3d_{i-1} x_i^2 \\ 2c_0 + 6d_0 x_1 &= 0, \quad 2c_i + 6d_i x_i = 2c_{i-1} + 6d_{i-1} x_i, \quad 2c_{n-1} + 6d_{n-1} x_n = 0 \end{aligned}$$

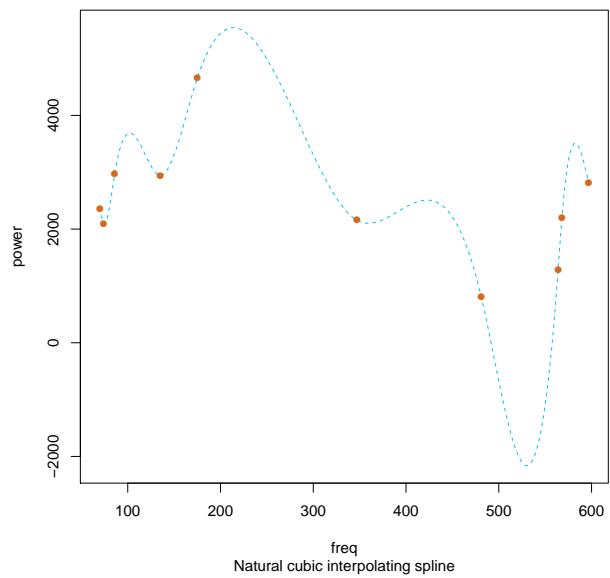
for  $i = 1, \dots, n-1$ , which is known as the [natural cubic interpolating](#) spline.

Q: Do you notice a problem with simple splines above?

A Sample of Cosmic Microwave Backgroud Radiation



Another Sample of Cosmic Microwave Backgroud Radiation



```
> library(splines)
>
> n.samp = 10
>
```

```

> index = sample(nrow(cmb.df), n.samp)
>
> cmb_train = cmb.df[index, ]
>
> cmb_train = cmb_train[order(cmb_train$freq), ]
>
> ispl = interpSpline(cmb_train$freq,
+                      cmb_train$power)
>
> pred.x = seq(min(cmb_train$freq),
+               max(cmb_train$freq),
+               length.out = 200)
>
> pred.vec = predict(ispl, pred.x)

```

- Kernel smoothing can be used to address continuity of  $m(x)$

$$\hat{m}(x) = \sum_{j=1}^n \left( \frac{w_j}{w} \right) y_j \quad \text{where } w = \sum_{j=1}^n w_j$$

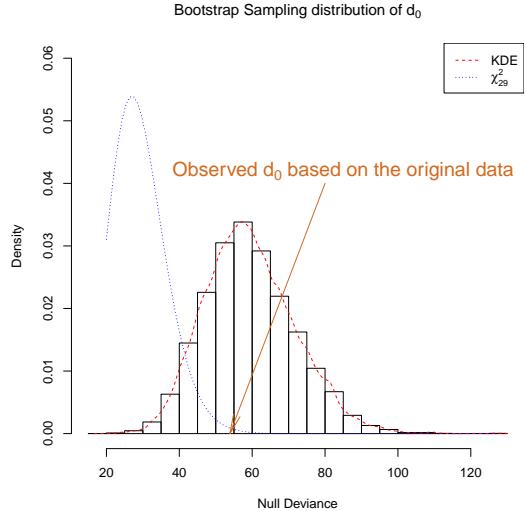
for a given  $h \in \mathbb{R}$  and some kernel function  $K(\cdot)$ ,  $w_j$  depends on  $x$  via  $K$

$$w_j = w_j(x) = K\left(\frac{x - x_j}{h}\right)$$

- Thus a kernel smoother estimates  $Y$  at  $x$  by a weighted mean of the data  $y_j$ .
- If being continuous/differentiable is important, Gaussian kernel is often used

$$K(z) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}z^2\right) \text{ for } z \in \mathbb{R}$$

- The choice of  $h$ , which is known as the **bandwidth**, is crucial.
- Kernel functions are often used in density estimation (KDE)



## 9.2 Kernel Smoothing

- In general, a kernel is a **non-negative** integrable function  $K(z)$  such that

$$K(z) = K(-z); \quad \int_{-\infty}^{\infty} K(z) dz = 1; \quad \lim_{z \rightarrow -\infty} K(z) = \lim_{z \rightarrow \infty} K(z) = 0$$

- Other common kernel functions include

Rectangular  $K(z) = \begin{cases} 1/2 & \text{for } |z| \leq 1, \\ 0 & \text{otherwise.} \end{cases}$

Triangular  $K(z) = \begin{cases} 1 - |z| & \text{for } |z| \leq 1, \\ 0 & \text{otherwise.} \end{cases}$

Parabolic  $K(z) = \begin{cases} 3(1 - z^2)/4 & \text{for } |z| \leq 1, \\ 0 & \text{otherwise.} \end{cases}$

- Gaussian kernel is often used for cases that need a high level of smoothness.
- It can be shown a kernel smoother

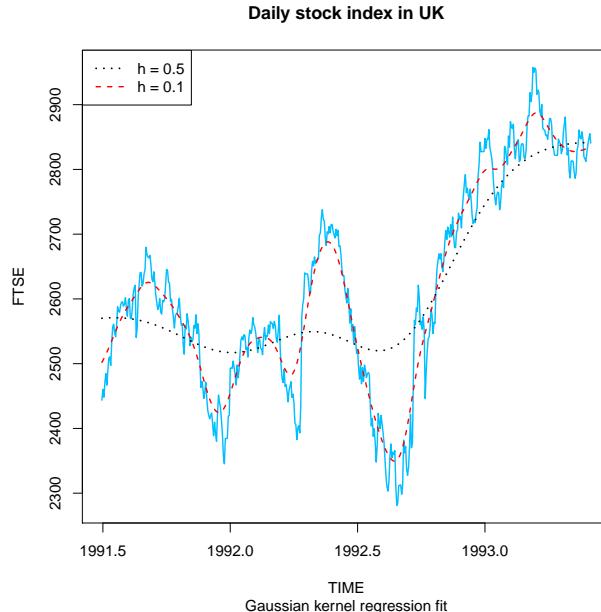
$$\hat{m}(x) = \sum_{i=1}^n \left( \frac{K\left(\frac{x-x_i}{h}\right)}{\sum_{j=1}^n K\left(\frac{x-x_j}{h}\right)} \right) y_i$$

is actually the solution to the following

$$\hat{m}(x) = \arg \min_{m_x \in \mathbb{R}} \sum_{j=1}^n K\left(\frac{x - x_j}{h}\right) (y_j - m_x)^2$$

which is a weighted/local least squares problem according to the kernel, note the difference between this optimisation and ordinary least squares.

- Due to this connection, kernel smoothing is often known as [kernel regression](#).
- The bandwidth  $h$  serves as a tuning parameter, thus often  $\hat{m}_h(x)$  is used.



```

> ftse = EuStockMarkets[1:500,4]
> times = time(EuStockMarkets)[1:500]
> plot(times, ftse, type="l",
+       xlab = "TIME", ylab = "FTSE",
+       lwd = 1.5, col = "deepskyblue",
+       main = "Daily stock index in UK",
+       sub = "Gaussian kernel regression fit")

> kh05 = ksmooth(times, ftse, kernel = "normal",
+                  bandwidth=0.5)
> lines(kh05, col = "black", lwd = 2, lty = 3)

> kh01 = ksmooth(times, ftse, kernel = "normal",
+                  bandwidth=0.1)
> lines(kh01, col = 2, lwd = 1.5, lty = 2)
>

```

```

> legend("topleft", c("h = 0.5", "h = 0.1"),
+         col = c("black", "red"),
+         lwd = c(2, 1.5), lty = 3:2)

```

- Similar to KDE, it can be shown for sufficiently small  $h$ , we have

$$\text{bias} [\hat{m}_h(x^*)] \approx \frac{h^2}{2} \int_{-\infty}^{\infty} z^2 K(z) dz \left( \underbrace{m''(x^*)}_{\text{curvature}} + 2 \underbrace{\frac{m'(x^*) f'_X(x^*)}{f_X(x^*)}}_{\text{design}} \right)$$

$$\text{Var} [\hat{m}_h(x^*)] \approx \frac{\sigma^2}{f_X(x)} \cdot \frac{1}{nh} \int_{-\infty}^{\infty} K^2(z) dz \quad \text{where } \sigma^2 = \text{Var} [\epsilon_i]$$

$$\text{MSE} [\hat{m}_h(x^*)] \approx (\text{bias} [\hat{m}_h(x^*)])^2 + \text{Var} [\hat{m}_h(x^*)]$$

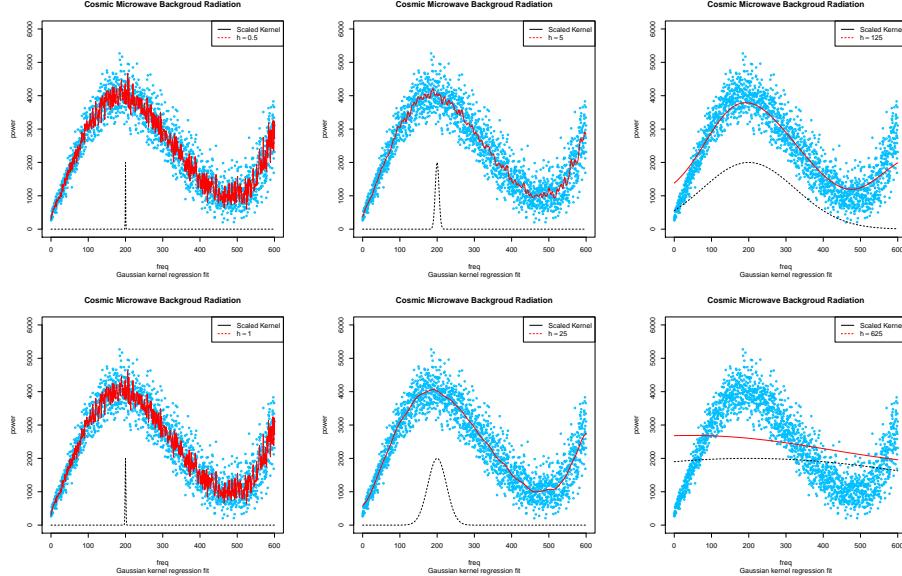
Q: Can you thus identify the sources of bias and variance?

- Theoretically,  $h$  can be chosen according to the mean integrated square error

$$h_{\text{opt}} = \arg \min_h \left\{ \int_{-\infty}^{\infty} \text{MSE} [\hat{m}_h(x^*)] dx^* \right\} = C \cdot \frac{1}{n^5}$$

where  $C$  is a constant depending on  $f_x$  and  $K$ .

- It seems  $h = 125$  is the best choice out of those  $h$  values by “eyeballing”.



Q: How can we choose the bandwidth  $h$  more precisely in practice?

- With enough computing power, we can assess the performance of a kernel regression model by estimating the MSE of the regression model using

### Leave-one-out Cross Validation (LOO-CV).

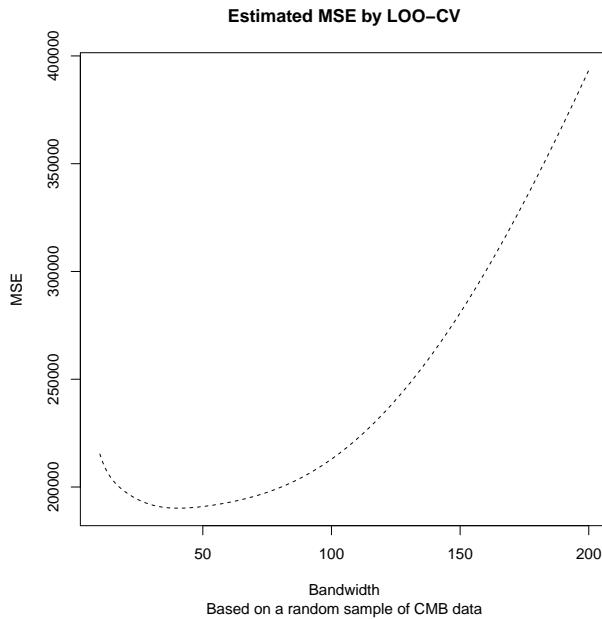
which involves computing  $\hat{m}(x)$  without the  $i$ th point for various  $h$  values,

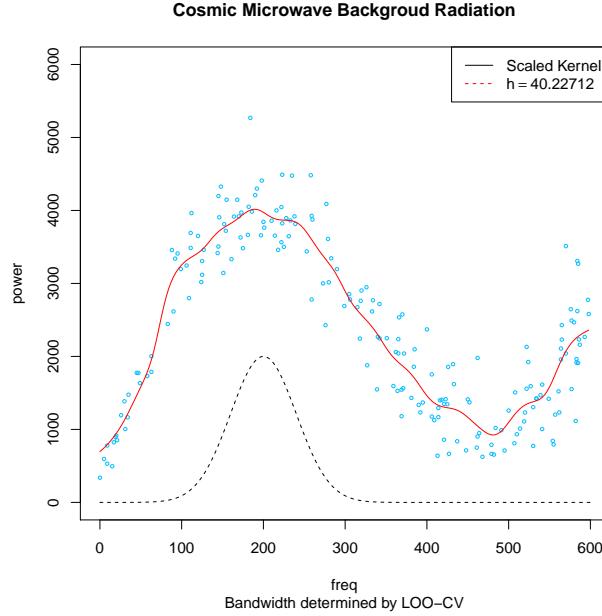
$$\hat{m}_h^{(-i)}(x)$$

- For a given  $h$ , the MSE =  $\nu$  is estimated by

$$\begin{aligned}\hat{\nu}(h) &= \frac{1}{n} \sum_{i=1}^n \left( y_i - \hat{m}_h^{(-i)}(x_i) \right)^2 \\ &= \frac{1}{n} \sum_{i=1}^n \left( \frac{y_i - \hat{m}_h(x_i)}{1 - s_{ii}} \right)^2 \quad \text{where} \quad s_{ij} = \frac{K\left(\frac{x_i - x_j}{h}\right)}{\sum_{\ell=1}^n K\left(\frac{x_i - x_\ell}{h}\right)}\end{aligned}$$

- The bandwidth  $h$  is chosen to minimise  $\hat{\nu}(h)$





- Before discussing more nonparametric estimators, let us introduce the notion of [linear smoother](#), which is an estimator that can be written as

$$\hat{m}(x) = \sum_{j=1}^n S_j(x)y_j$$

where  $S_i$  is some function depending on  $x_1, x_2, \dots, x_n$  but not on any of  $y_i$ .

- It means the fitted values under a linear smoother is simply given by

$$\hat{y}_i = \hat{m}(x_i) = \sum_{j=1}^n S_j(x_i)y_j = \sum_{j=1}^n s_{ij}y_j \implies \hat{\mathbf{y}} = \mathbf{S}\mathbf{y}$$

where the matrix  $\mathbf{S}$  is known as the [smoothing matrix](#).

- It is linear because the fitted value is a linear combination of the observed  $Y$ .
- Notice our linear model can also be considered as a linear smoother,

$$\hat{\mathbf{y}} = \mathbf{X} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = \mathbf{P}\mathbf{y}$$

so the goal here is to generalise linear mdoel along another line of thinking.

- It is clear that the kernel regression is a linear smoother since the fitted value

$$\hat{y}_i = \hat{m}(x_i)$$

can be expressed as linear combination of other observed  $Y$ ,

$$\hat{y}_i = \sum_{j=1}^n s_{ij}y_j \quad \text{where} \quad s_{ij} = \frac{K\left(\frac{x_i - x_j}{h}\right)}{\sum_{\ell=1}^n K\left(\frac{x_i - x_\ell}{h}\right)}$$

- Being a linear smoother allows us to estimate  $\sigma^2$  easily, so we know roughly

$$\text{bias}[\hat{m}_h(x^*)] \approx \frac{h^2}{2} \int_{-\infty}^{\infty} z^2 K(z) dz \left( m''(x^*) + 2 \frac{m'(x^*) f'_X(x^*)}{f_X(x^*)} \right)$$

$$\text{Var}[\hat{m}_h(x^*)] \approx \frac{\sigma^2}{f_X(x)} \cdot \frac{1}{nh} \int_{-\infty}^{\infty} K^2(z) dz \quad \text{where } \sigma^2 = \text{Var}[\epsilon_i]$$

since terms in red is readily available and  $f_x$  can be estimated using KDE.

- For a matrix  $\mathbf{A}$ , and a random vector  $\mathbf{y}$ , the following holds

$$\text{Var}[\mathbf{Ay}] = \mathbf{A} \text{Var}[\mathbf{y}] \mathbf{A}^T$$

using it on the variance-covariance matrix of the residual vector, we have

$$\text{Var}[\hat{\mathbf{e}}] = \text{Var}[(\mathbf{I} - \mathbf{S})\mathbf{y}] = (\mathbf{I} - \mathbf{S}) \text{Var}[\mathbf{y}] (\mathbf{I} - \mathbf{S})^T$$

- Since  $Y_i$  are assumed to be i.i.d., thus

$$\text{Var}[\mathbf{y}] = \sigma^2 \mathbf{I} \implies \text{Var}[\hat{\mathbf{e}}] = \sigma^2 (\mathbf{I} - \mathbf{S} - \mathbf{S}^T + \mathbf{S}\mathbf{S}^T)$$

- Taking matrix trace, where  $\nu = \text{Trace}(\mathbf{S})$  and  $\eta = \text{Trace}(\mathbf{S}\mathbf{S}^T)$ , we have

$$\sum_{i=1}^n \text{Var}[\hat{e}_i] = \sigma^2 (n - 2\nu + \eta) \implies \hat{\sigma}^2 = \frac{1}{n - 2\nu + \eta} \sum_{i=1}^n \hat{e}_i^2$$

for the residual sum of squares  $\sum_{i=1}^n \hat{e}_i^2$  is a consistent estimator of  $\sum_{i=1}^n \text{Var}[\hat{e}_i]$

### 9.3 Penalised Regression

- Recall the kernel regression is solution to the following WLS problem

$$\hat{m}(x) = \arg \min_{m_x \in \mathbb{R}} \sum_{j=1}^n K\left(\frac{x - x_j}{h}\right) (y_j - m_x)^2$$

that is, given a value of  $x$ , the optimisation returns the fitted value  $m_x$  at  $x$ .

- Now consider replacing this single fitted value  $m_x$  by a polynomial,

$$P_x(u; \beta) = \beta_0 + \beta_1(u - x) + \frac{\beta_2}{2!}(u - x)^2 + \cdots + \frac{\beta_p}{p!}(u - x)^p$$

then the optimisation becomes

$$\arg \min_{\beta \in \mathbb{R}^{p+1}} \sum_{j=1}^n K\left(\frac{x - x_j}{h}\right) (y_j - P_x(x_j; \beta))^2$$

- A local polynomial regression chooses the estimate to be

$$\hat{m}(x) = \beta_0(x)$$

- Let  $\mathbf{W}_x = \text{diag} \left( K \left( \frac{x - x_1}{h} \right), K \left( \frac{x - x_2}{h} \right), \dots, K \left( \frac{x - x_j}{h} \right) \right)$ , and

$$\mathbf{X}_x = \begin{bmatrix} 1 & x_1 - x & \cdots & (x_1 - x)^p/p! \\ 1 & x_2 - x & \cdots & (x_2 - x)^p/p! \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n - x & \cdots & (x_n - x)^p/p! \end{bmatrix}$$

then the optimisation for local polynomial regression

$$\arg \min_{\boldsymbol{\beta} \in \mathbb{R}^{p+1}} \sum_{j=1}^n K \left( \frac{x - x_j}{h} \right) (y_j - P_x(x_i; \boldsymbol{\beta}))^2$$

has the following usual WLS solution

$$\boldsymbol{\beta}_x = (\mathbf{X}_x^T \mathbf{W}_x \mathbf{X}_x)^{-1} \mathbf{X}_x^T \mathbf{W}_x \mathbf{y}$$

- Thus the fitted value is the first element of the solution

$$\hat{m}(x) = [\boldsymbol{\beta}_x]_1$$

- The local polynomial regression is a linear smoother as well,

$$\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}$$

since the fitted values of a local polynomial regression is given by

$$\hat{y}_i = [\boldsymbol{\beta}_{x_i}]_1 = \left[ (\mathbf{X}_{x_i}^T \mathbf{W}_{x_i} \mathbf{X}_{x_i})^{-1} \mathbf{X}_{x_i}^T \mathbf{W}_{x_i} \right]_1 \mathbf{y}$$

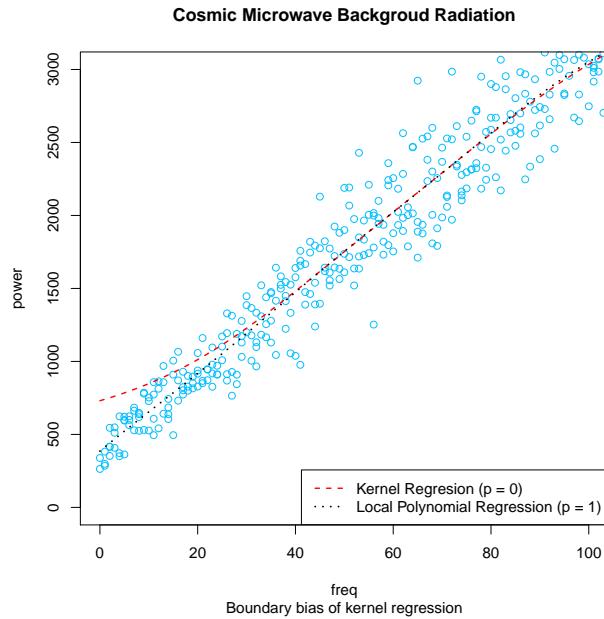
- The smoothing matrix for local polynomial regression is given by

$$s_{ij} = \left[ (\mathbf{X}_{x_i}^T \mathbf{W}_{x_i} \mathbf{X}_{x_i})^{-1} \mathbf{X}_{x_i}^T \mathbf{W}_{x_i} \right]_{1j}$$

- Although only the intercept of the polynomial is used, it is different from

$$\beta_0(x) \neq \arg \min_{m_x \in \mathbb{R}} \sum_{j=1}^n K \left( \frac{x - x_j}{h} \right) (y_j - \beta_0)^2$$

- Kernel regression can be thought as local polynomial regression with  $p = 0$ .



```

> kreg = ksmooth(cmb.df$freq, cmb.df$power,
+                  kernel = 'normal', bandwidth = 44)

> library(KernSmooth)
>
> lreg = locpoly(cmb.df$freq, cmb.df$power, degree=1,
+                  bandwidth=dpline(cmb.df$freq, cmb.df$power))

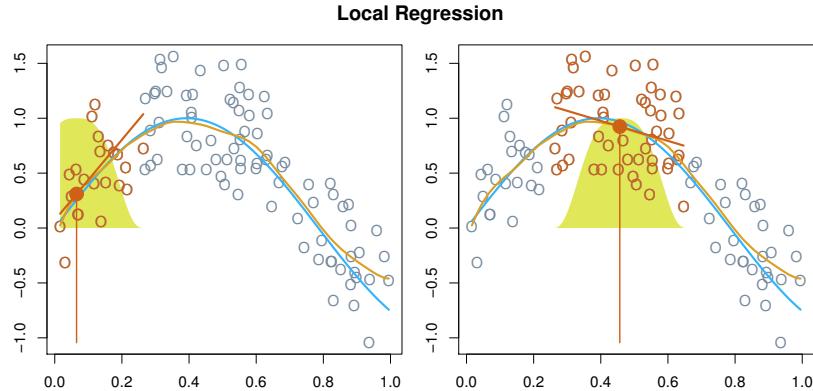
> plot(power~freq, data = cmb.df, xlim=c(0,100),
+       ylim = c(0, 3000), col = "deepskyblue",
+       main ="Cosmic Microwgroud Radiation",
+       sub = "Boundary bias of kernel regression")
>
> lines(kreg, col=2, lwd = 1.5, lty = 2)
> lines(lreg, col= 1, lwd = 2, lty = 3)
>
> legend("bottomright",
+         c("Kernel Regresion (p = 0)",
+           "Local Polynomial Regression (p = 1)"),
+         col = c(2,1), lwd = c(1.5, 2), lty = 2:3)

```

- Recall our objective is to have some model for

$$Y_i = m(x_i) + \varepsilon_i$$

where  $m$  is an ‘arbitrary’ function, local regression achieves that via WLS.



- Recall **natural cubic interpolating spline**, NCIS, and notice it is a solution to

$$\arg \min_{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}} \sum_{i=1}^n (y_i - \hat{m}(x_i))^2$$

- The word **interpolating** is used because like

interpolating polynomial

NCIS is forced to pass through the data values, which results overfitting.

Q: How can we adjust the amount of “alignment” between spline and data?

- Recall the concavity of a curve is given by the sign of its second derivative

$$\hat{m}''(x)$$

- So the relative extent of overfitting on an interval  $[a, b]$  can be quantified by

$$I[\hat{m}] = \int_a^b (\hat{m}''(x))^2 dx$$

- To see what I meant by the relative extent, consider the Doppler function

$$m(x) = \sqrt{x(1-x)} \sin\left(\frac{2.1\pi}{x+0.05}\right) \quad \text{for } 0 \leq x \leq 1$$

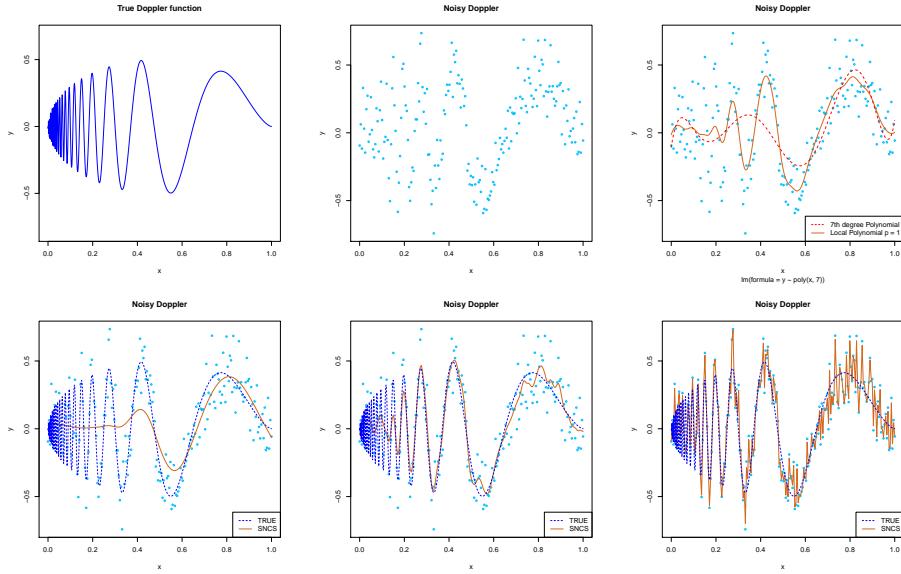
being the true unknown mean function of the model

$$Y_i = m(x_i) + \varepsilon_i$$

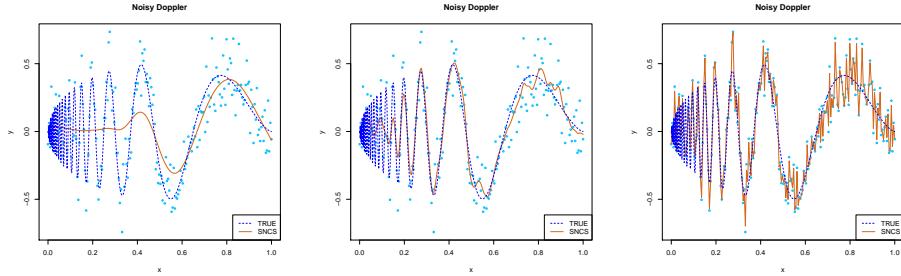
where  $\varepsilon_i$  is assumed to be normal.

```
> n = 200
> x = seq(0, 1, length = n)
> m = sqrt(x*(1-x)) * sin(2.1*pi/(x+0.05))
> y = m + 0.15*rnorm(n)
```

- It is a very challenging dataset to work with since error will dominate small  $x$



- Notice as the squared error decreases from the left to the right,



while the relative measure of overfitting

$$I[\hat{m}] = \int_a^b (\hat{m}''(x))^2 dx$$

which is known as the **roughness** increases from the left to the right.

- Therefore,  $\hat{m}$  should be chosen so that the following is minimised,

$$\sum_{i=1}^n (y_i - \hat{m}(x_i))^2 + \lambda \int_a^b (\hat{m}''(x))^2 dx$$

where  $\lambda$  is known as the **smoothing parameter**, which controls the roughness

- When  $\lambda \rightarrow 0$ , then the solutions converge to the interpolating spline.
- When  $\lambda \rightarrow \infty$ , then the solutions converge to the linear regression fit.

- Typically, we also require  $\hat{m}$  to have the following properties:

1. It passes through a certain set of points,

$$\hat{m}(x_j) = f_j, \quad j = 1, \dots, n$$

2. It is continuous, and has continuous first and second derivatives.
3. It has vanishing second derivatives at the two boundary points

$$\hat{m}''(x_1) = \hat{m}''(x_n) = 0$$

- It can be shown that out of all functions that satisfy all three conditions,

$$\hat{m}(x) = \begin{cases} \hat{m}_1(x) = a_1 + b_1x + c_1x^2 + d_1x^3, & x \in [x_1, x_2), \\ \vdots & \vdots \\ \hat{m}_i(x) = a_i + b_ix + c_ix^2 + d_ix^3, & x \in [x_i, x_{i+1}), \\ \vdots & \vdots \\ \hat{m}_{n-1}(x) = a_{n-1} + b_{n-1}x + c_{n-1}x^2 + d_{n-1}x^3, & x \in [x_{n-1}, x_n]. \end{cases}$$

which is known as the [smoothing spline](#), is the solution that minimises

$$\sum_{i=1}^n (y_i - \hat{m}(x_i))^2 + \lambda \int_{x_1}^{x_n} (\hat{m}''(x))^2 dx$$

since the smoothing spline has the smallest roughness value

$$I[\hat{m}] = \int_{x_1}^{x_n} (\hat{m}''(x))^2 dx$$

- It turns out, if we use a special basis, the solution that minimises

$$\sum_{i=1}^n (y_i - \hat{m}(x_i))^2 + \lambda \int_{x_1}^{x_n} (\hat{m}''(x))^2 dx$$

can be conveniently written despite the piecewise nature of the solution.

- Consider a knot sequence  $\{\xi\}$  with  $L$  [interior knots](#),

$$\xi_1 \leq \xi_2 \leq \xi_3 \leq \dots \leq \xi_L$$

- Let  $\xi_0 < \xi_1$  and  $\xi_L < \xi_{L+1}$  be two [boundary knots](#), which define the interval over which our spline is evaluated, and  $\{\tau\}$  be the [augmented knot sequence](#),

$$\tau_0 \leq \tau_1 \leq \tau_2 \leq \tau_3 \leq \dots \leq \tau_Q \leq \xi_0;$$

$$\tau_{\ell+Q} = \xi_\ell, \quad \text{for } \ell = 1, \dots, L;$$

$$\xi_{L+1} \leq \tau_{L+Q+1} \leq \tau_{L+Q+2} \leq \dots \leq \tau_{L+2Q}.$$

- In general, a spline function of order  $Q$  is a piecewise polynomial function of degree  $Q-1$  in a variable, thus a cubic spline is a  $Q=4$ th order spline.

- The  $p$ th spline basis function of order  $Q$ ,  $B_{p,Q}$ , can be defined recursively,

$$B_{p,1}(x) = \begin{cases} 1 & \text{if } \tau_p \leq x < \tau_{p+1}, \\ 0 & \text{otherwise.} \end{cases} \quad \text{for } p = 1, \dots, L + 2Q - 1.$$

$$\begin{aligned} B_{p,q}(x) &= \frac{x - \tau_p}{\tau_{p+q-1} - \tau_p} B_{p,q-1}(x) && \text{for } q \leq Q \\ &+ \frac{\tau_{p+q} - x}{\tau_{p+q} - \tau_{p+1}} B_{p+1,q-1}(x) && \text{for } p = 1, \dots, L + 2Q - q. \end{aligned}$$

- This “convenient” basis is known as **B-spline**. Since  $\hat{m}$  is a cubic spline,

$$\hat{m}(x) = \sum_{p=1}^{n+4} \beta_p B_p(x) \quad \text{where } B_p(x) = B_{p,Q}(x)$$

- Substituting the representation of  $\hat{m}$  in B-spline basis into the following

$$\sum_{i=1}^n (y_i - \hat{m}(x_i))^2 + \lambda \int_{\xi_1}^{\xi_n} (\hat{m}''(x))^2 dx$$

then it becomes clear that we need to minimise with respect to  $\beta_p$ ,

$$\sum_{i=1}^n \left( y_i - \sum_{p=1}^{n+4} \beta_p B_p(x_i) \right)^2 + \lambda \sum_{p=1}^{n+4} \sum_{p'=1}^{n+4} \beta_p \beta_{p'} \int_{\xi_1}^{\xi_n} B_p''(x) B_{p'}''(x) dx$$

- Let  $\mathbf{y}$  be the usual response vector, and  $\mathbf{B}_{n \times (n+4)}$  denote the matrix of

$$[\mathbf{B}]_{ip} = B_p(x_i)$$

and  $\mathbf{\Lambda}_{(n+4) \times (n+4)}$  be the matrix of

$$[\mathbf{\Lambda}]_{pp'} = \int_{\xi_1}^{\xi_n} B_p''(x) B_{p'}''(x) dx$$

- Using B-spline basis and the notations defined above, we have

$$\hat{\boldsymbol{\beta}} = \arg \min_{\boldsymbol{\beta}} \left\{ (\mathbf{y} - \mathbf{B}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{B}\boldsymbol{\beta}) - \lambda \boldsymbol{\beta}^T \mathbf{\Lambda} \boldsymbol{\beta} \right\}$$

- Differentiating and setting the gradient equal to zero, we have

$$\hat{\boldsymbol{\beta}} = (\mathbf{B}^T \mathbf{B} + \lambda \mathbf{\Lambda})^{-1} \mathbf{B}^T \mathbf{y}$$

- Notice similarities between linear ridge regression and smoothing spline,

$$\hat{\boldsymbol{\beta}}^{\text{ridge}} = (\mathbf{X}^{*\top} \mathbf{X}^* + \lambda \mathbf{I})^{-1} \mathbf{X}^{*\top} \mathbf{y}$$

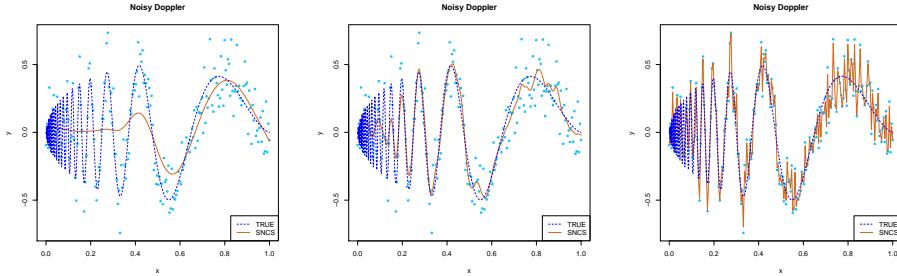
- It is clear that smoothing spline is also a linear smoother,

$$\hat{\mathbf{y}} = \mathbf{B} \hat{\boldsymbol{\beta}} = \mathbf{S} \mathbf{y} \quad \text{where } \mathbf{S} = \mathbf{B} (\mathbf{B}^T \mathbf{B} + \lambda \mathbf{\Lambda})^{-1} \mathbf{B}^T$$

- Unlike local polynomial regression, prediction requires no optimisation again.

Q: How can we pick the roughness parameter  $\lambda$ ?

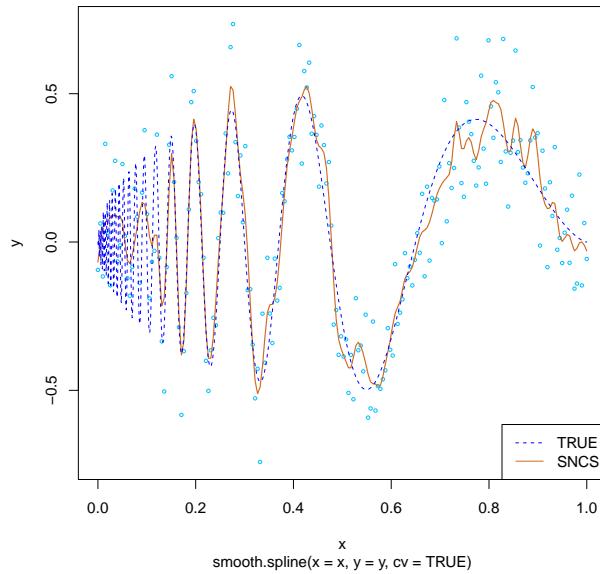
$$\hat{\beta} = (\mathbf{B}^T \mathbf{B} + \lambda \Lambda)^{-1} \mathbf{B}^T \mathbf{y}$$



- Alternative to eyeballing, LOO-CV can be used to choose  $\lambda$ ,

```
> sm.spl = smooth.spline(x, y, cv = TRUE)
>
> lines(sm.spl, col = "chocolate", lwd = 1.2)
```

Smoothing Spline where  $\lambda$  is chosen according to LOO-CV



## 10 Generalised Additive Model

- Recall the nonparametric model we have considered takes the following form

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

which is essentially an generalisation of SLR to model non-linear relationships

$$y_i = \hat{\beta}_0 + \hat{\beta}_1 x_i + \hat{e}_i$$

Q: How can we do something similar for multiple linear regression?

$$y_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + \cdots \hat{\beta}_k x_{ik} + \hat{e}_i$$

- It is natural to propose the following

$$y_i = \hat{m}_1(x_{i1}) + \hat{m}_2(x_{i2}) + \cdots \hat{m}_k(x_{ik}) + \hat{e}_i$$

where  $\hat{m}_j$  is a piecewise polynomial.

- However, there is a problem without additional restrictions.
- Consider the simple case where  $k = 2$ , i.e. the conditional mean is given by

$$\mathbb{E}[Y_i | X_{i1} = x_{i1}, X_{i2} = x_{i2}] = m(x_{i1}) + m(x_{i2})$$

- Now imagine we add a constant  $c$  to  $g_1$  and subtract a constant  $c$  from  $g_2$ ,

$$m_1(x_{i1}) + c \quad \text{for all } x_{i1}$$

$$m_2(x_{i1}) - c \quad \text{for all } x_{i2}$$

then nothing observable has changed about the model,

$$\mathbb{E}[Y_i | X_{i1} = x_{i1}, X_{i2} = x_{i2}] = m(x_{i1}) + m(x_{i2})$$

this is known as a **non-identifiable** model in statistics.

- It is similar to when we have perfect multicollinearity in MLR, i.e. singular

$$\mathbf{X}^T \mathbf{X}$$

- Practically, it just means the optimisation procedure for finding the set of

$$\hat{m}_j$$

will fail since there are infinitely many solutions.

- The non-identifiable part of model can be eliminated by further restrictions.
- The standard convention in this case is to require

$$\sum_{i=1}^n m_j(x_{ij}) = 0 \quad \text{for all } j$$

and adding a constant to the conditional mean independent of any  $g_j$

$$Y_i = \beta_0 + m_1(x_{i1}) + m_2(x_{i2}) + \cdots m_k(x_{ik}) + \varepsilon_i$$

which then is known as **additive model** if we assume

$$\varepsilon_i \stackrel{i.i.d.}{\sim} \text{Normal}(0, \sigma^2)$$

- To illustrate additive model, consider the following dataset

|           |                                            |
|-----------|--------------------------------------------|
| prestige  | Pineo-Porter prestige score for occupation |
| income    | Average income                             |
| education | Average number of years of education       |

```
> library(carData) # The dataset is a part of it
> attach(Prestige) # Variables become global
```

```
> sapply(list(prestige, income, education), summary)
```

|         | [,1]     | [,2]      | [,3]     |
|---------|----------|-----------|----------|
| Min.    | 14.80000 | 611.000   | 6.38000  |
| 1st Qu. | 35.22500 | 4106.000  | 8.44500  |
| Median  | 43.60000 | 5930.500  | 10.54000 |
| Mean    | 46.83333 | 6797.902  | 10.73804 |
| 3rd Qu. | 59.27500 | 8187.250  | 12.64750 |
| Max.    | 87.20000 | 26879.000 | 15.97000 |

```
> pre.LM = lm(prestige ~ income + education)
```

```
> summary(pre.LM)
```

```
Call:
lm(formula = prestige ~ income + education)

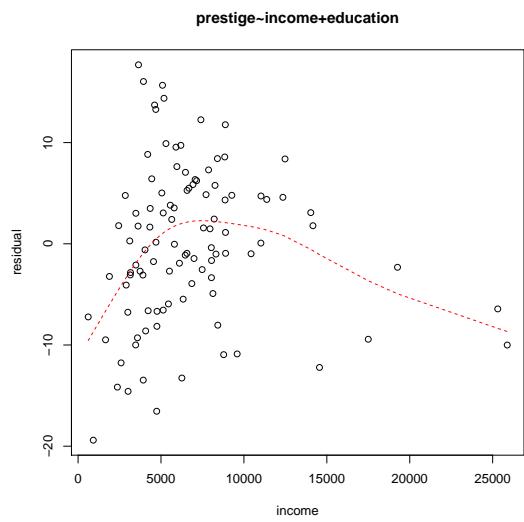
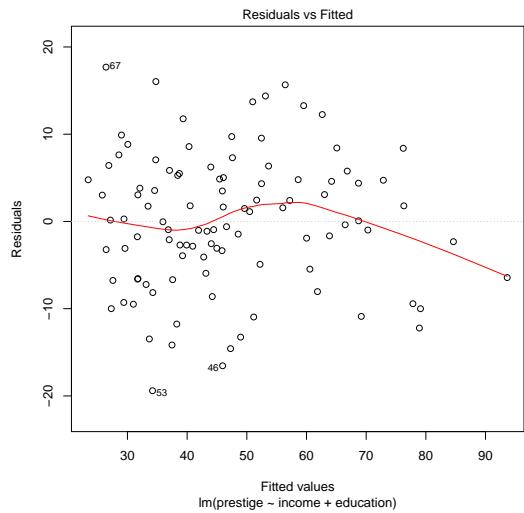
Residuals:
    Min      1Q  Median      3Q     Max 
-19.4040 -5.3308  0.0154  4.9803 17.6889 

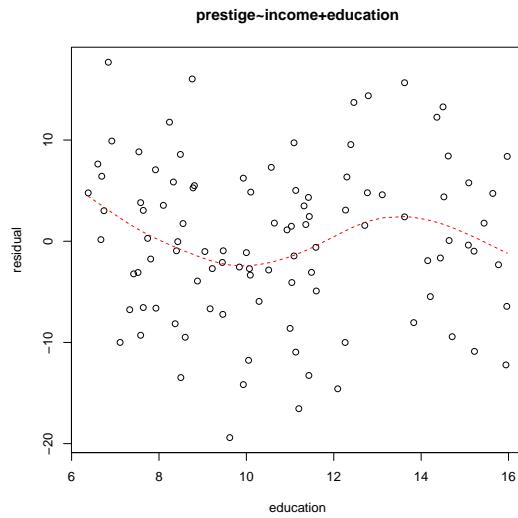
Coefficients:
            Estimate Std. Error t value Pr(>t)
(Intercept) -6.8477787  3.2189771 -2.127  0.0359 *
income       0.0013612  0.0002242  6.071 2.36e-08 ***
education    4.1374444  0.3489120 11.858 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.81 on 99 degrees of freedom
Multiple R-squared:  0.798,   Adjusted R-squared:  0.7939 
F-statistic: 195.6 on 2 and 99 DF,  p-value: < 2.2e-16
```

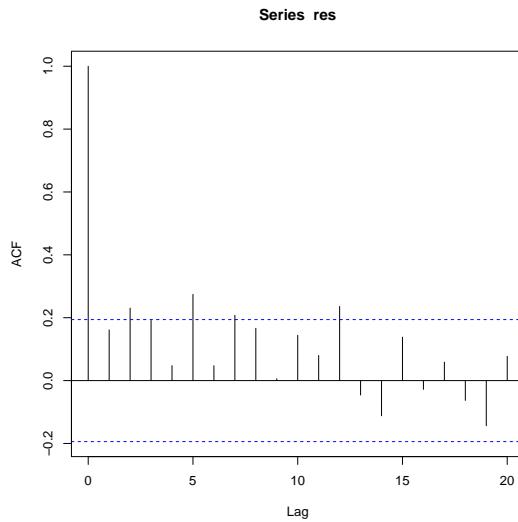
- Running the diagnostics shows we might have non-linearity issue,

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

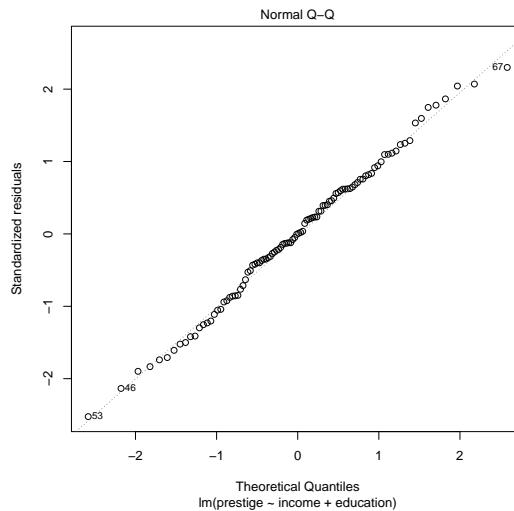




- And the errors might lack independence as well



- However, normality seems to be OK



- So we will try to first linearity instead of transforming the response, and we can do so using try regression spline instead of polynomial regression.

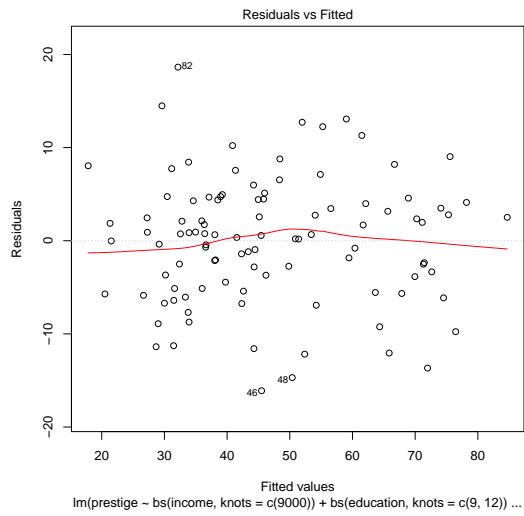
```
> pre.spline.LM =
+   lm(prestige~bs(income, knots = c(9000)
+                 +bs(education, knots = c(9, 12)))
```

where the choices of knots are made largely based on residual plots and the knowledge regarding education system, your guess is just as good as mine.

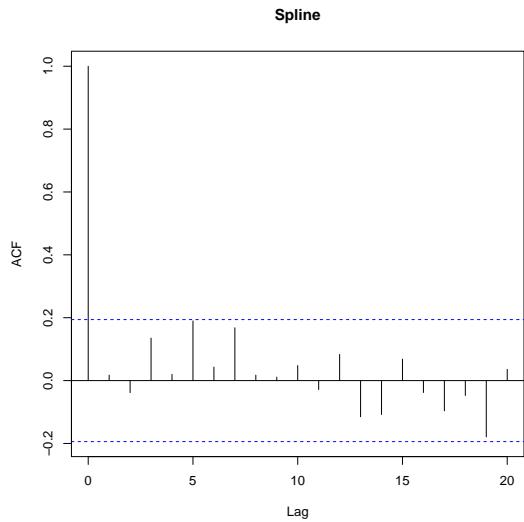
- Looking at diagnostics again

```
> plot(pre.spline.LM, which = 1)
>
> plot(pre.spline.LM, which = 2)
>
> res = pre.spline.LM$residuals
>
> acf(res)
```

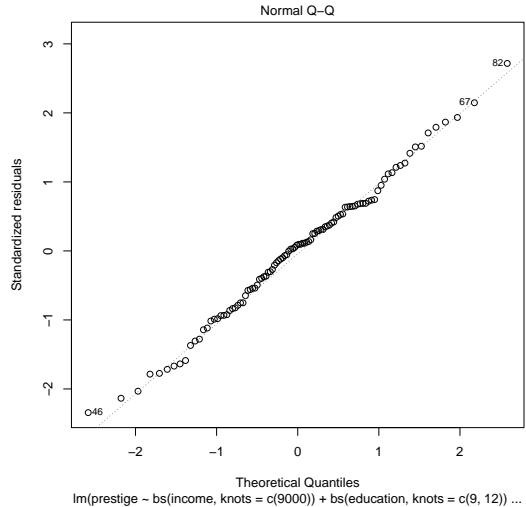
- It seems that having one knot for `income` and two knots for `education` is enough to fix the non-linearity problem.



- It seems also alleviate the independence problem



- And normality seems to be fine for this model as well



- So far we have only considered additive models that use splines, that is,

$$m_j(x_{ij})$$

is modelled by a piecewise polynomial of  $x_{ij}$ .

- In fact, any other type of functions can be used in additive models as long as

$$Y_i = \mathbb{E}[Y_i | \mathbf{X}] + \varepsilon_i$$

where the conditional mean is modelled by

$$\mathbb{E}[Y_i | \mathbf{X}] = \beta_0 + m_1(x_{i1}) + m_2(x_{i2}) + \dots + m_k(x_{ik})$$

- Thus technically your polynomial regression model is also an additive model.
- And using smoothing splines and a linear function are certainly allowed.
- To illustrate such additive model, consider the following dataset again

|           |                                             |
|-----------|---------------------------------------------|
| wage      | Raw wage in the Mid-Atlantic region         |
| age       | Age of the worker                           |
| year      | The year that wage information was recorded |
|           | 1. < HS Grad                                |
|           | 2. HS Grad                                  |
| education | A factor with levels:                       |
|           | 3. Some College                             |
|           | 4. College Grad                             |
|           | 5. Advanced Degree                          |

```
> library(ISLR) # The Wage dataset is a part of it
> attach(Wage) # Variables in Wage become global
>
> wage.LM = lm(wage ~ age + year + education)
```

```
> summary(wage.LM)
```

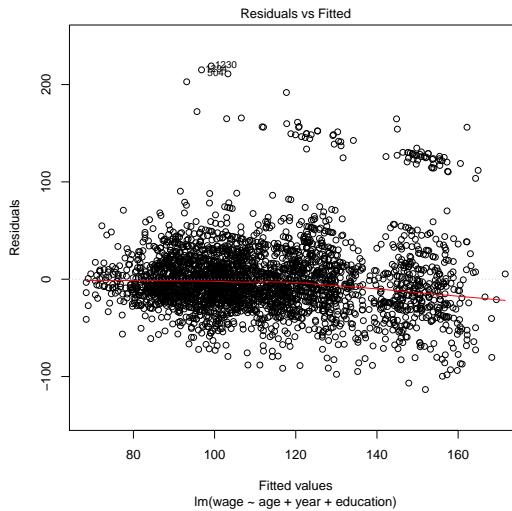
```
Call:
lm(formula = wage ~ age + year + education)

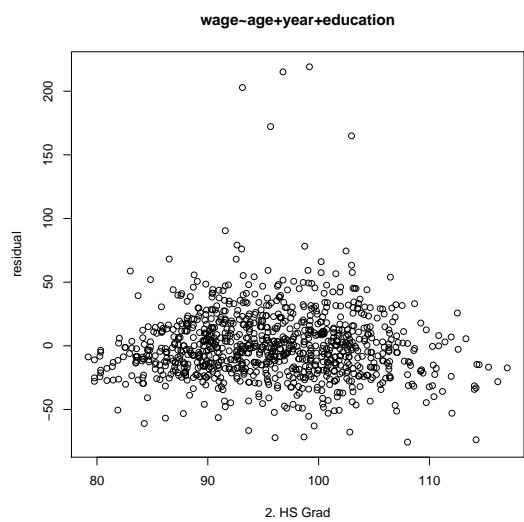
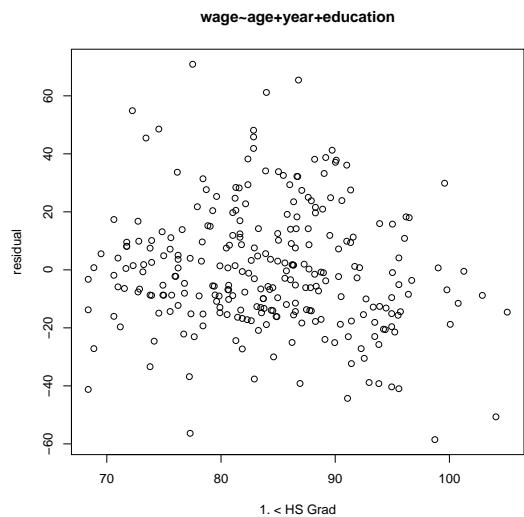
Residuals:
    Min      1Q  Median      3Q     Max 
-113.323 -19.521 -3.964  14.438 219.172 

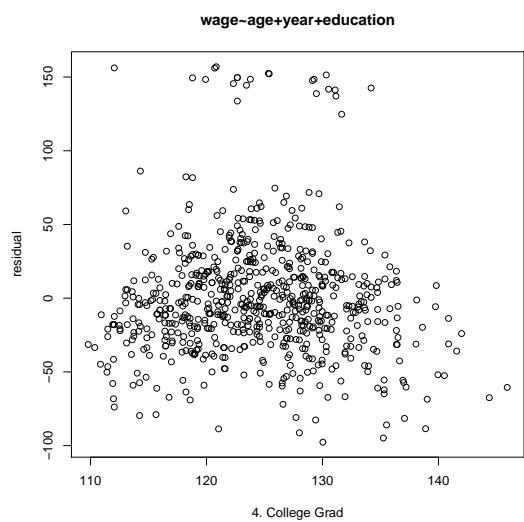
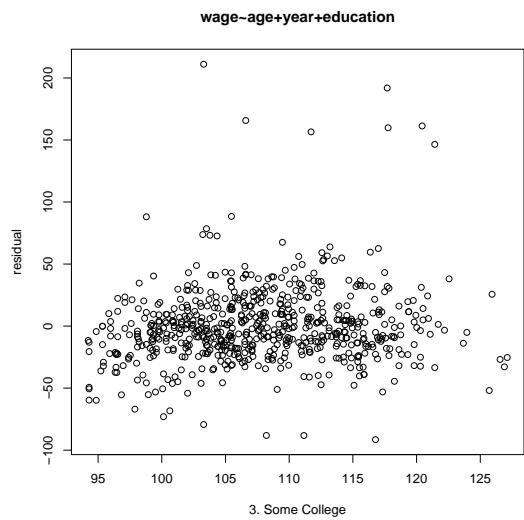
Coefficients:
            Estimate Std. Error t value Pr(>t)
(Intercept) -2.058e+03 6.493e+02 -3.169 0.00154 ** 
age          5.621e-01 5.714e-02  9.838 < 2e-16 *** 
year         1.056e+00 3.238e-01  3.262 0.00112 ** 
education2. HS Grad 1.140e+01 2.476e+00  4.603 4.34e-06 *** 
education3. Some College 2.423e+01 2.606e+00  9.301 < 2e-16 *** 
education4. College Grad 3.974e+01 2.586e+00 15.367 < 2e-16 *** 
education5. Advanced Degree 6.485e+01 2.804e+00 23.128 < 2e-16 *** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

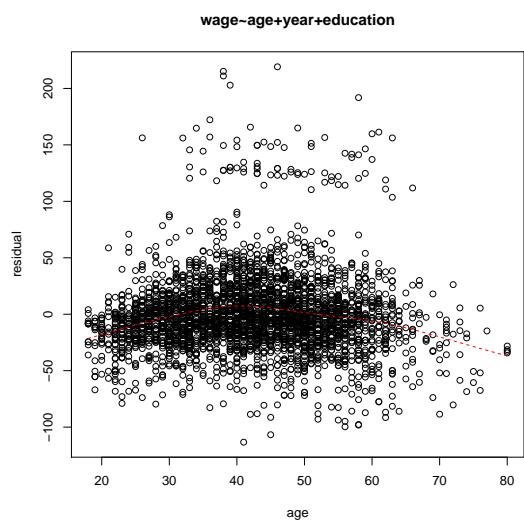
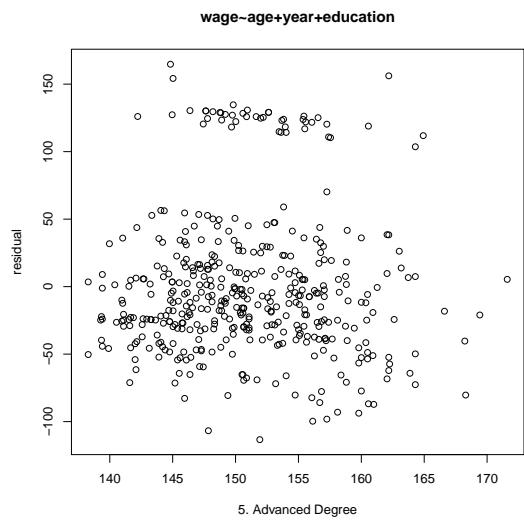
Residual standard error: 35.89 on 2993 degrees of freedom
Multiple R-squared:  0.2619,   Adjusted R-squared:  0.2604 
F-statistic: 177 on 6 and 2993 DF,  p-value: < 2.2e-16
```

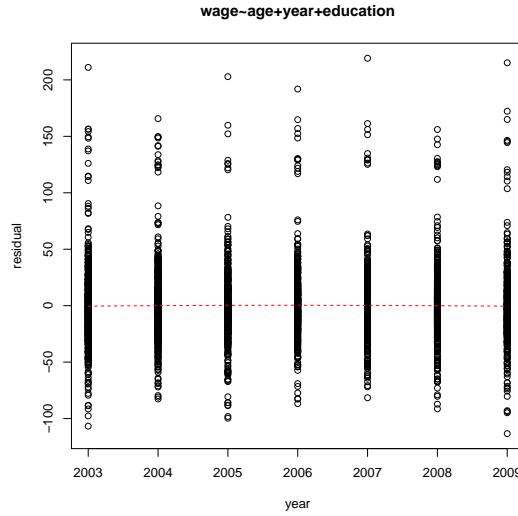
- The model seems to be very promising, but running the diagnostics, we got a rather ugly residual plot, which suggests we should investigate further.
- Initially, I thought it must be due to **education**.





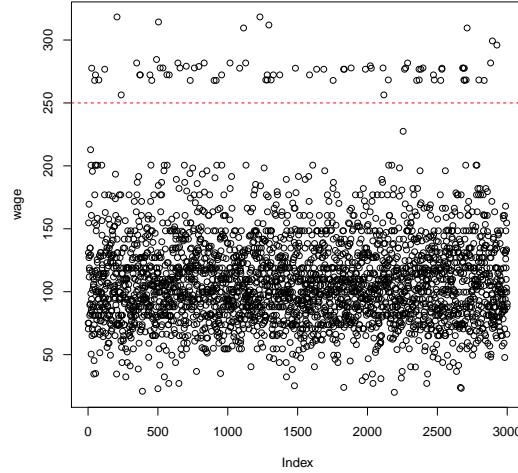






- It seems there is no variable inside the dataset can be used to explain the group of usually hight wages. So we have to treat them as outliers.

```
> plot(wage); abline(h=250, col = "red", lty = 2)
```



- Splitting the data, and investigate them individually,

```
> wage.1250.df =
+   subset(Wage, wage<250,
+         select = c(wage, age, year, education))
```

```

>
> attributes(wage.1250.df)$row.names =
+   1:nrow(wage.1250.df)
>
> wage.g250.df =
+   subset(Wage, wage>250,
+          select = c(wage, age, year, education))
>
> attributes(wage.g250.df)$row.names =
+   1:nrow(wage.g250.df)

```

- In practice, we do that to compare the differences between the two portions of the dataset, and in the hope that more information regarding the dataset become available in the future, and allows to explain the difference.

- Fit the linear model again,

```

> wage.1250.LM = lm(wage~age+year+education,
+                     data = wage.1250.df)

```

it seems the model assumptions are reasonably good except normality.

```
> shapiro.test(wage.1250.LM$residuals)
```

```

Shapiro-Wilk normality test

data: wage.1250.LM$residuals
W = 0.99288, p-value = 8.495e-11

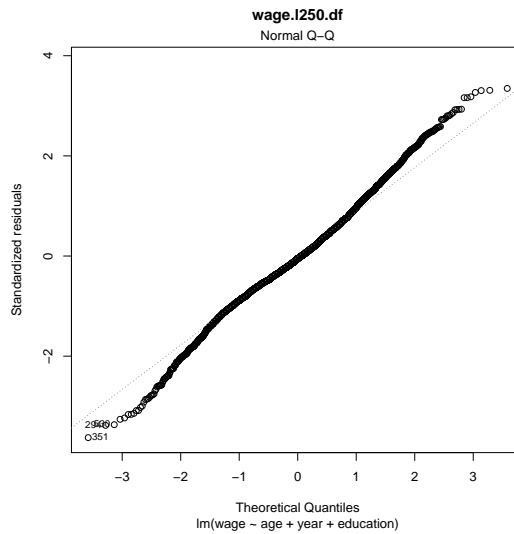
```

- Because the dataset is still reasonably large,

```
> nrow(wage.1250.df)
```

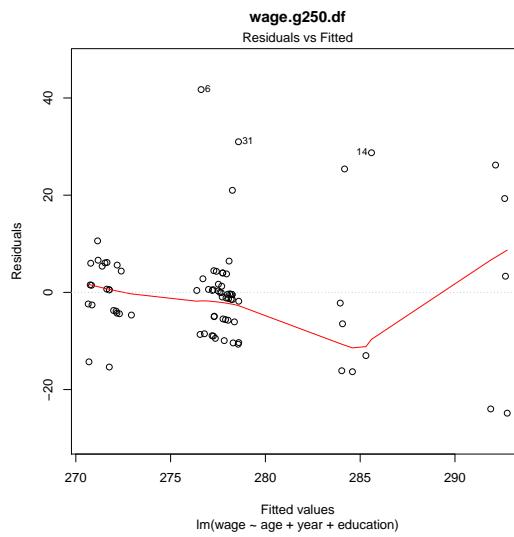
```
[1] 2921
```

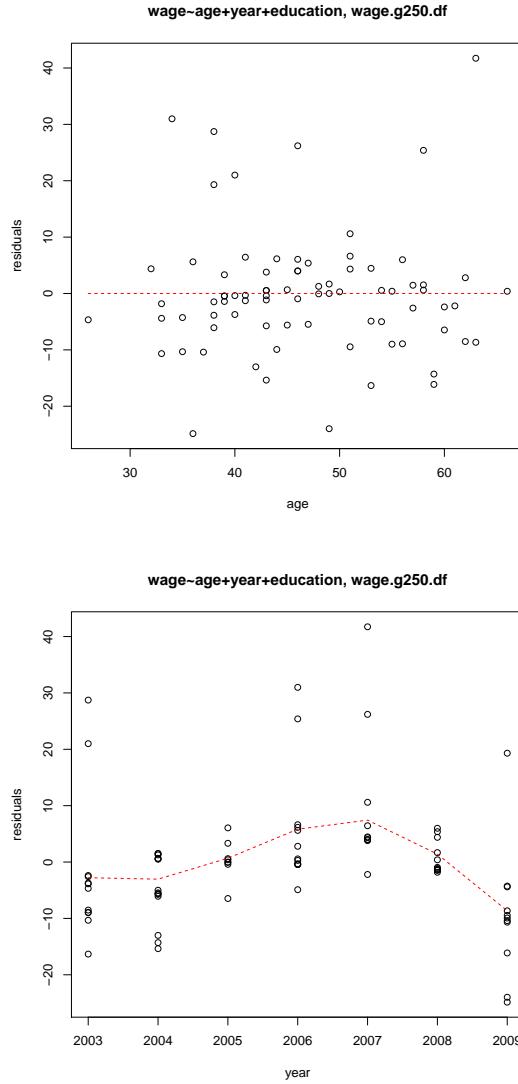
and QQ-normal plot reveals the distribution is fairly symmetric, so we will reply on the central limit theorem and not consider further transformation.



- For the other portion, additive model with smoothing spline is worth trying

```
> wage.g250.LM = lm(wage ~ age + year + education,
+ data = wage.g250.df)
```





- It seems the following additive model is reasonable but far from perfect
 

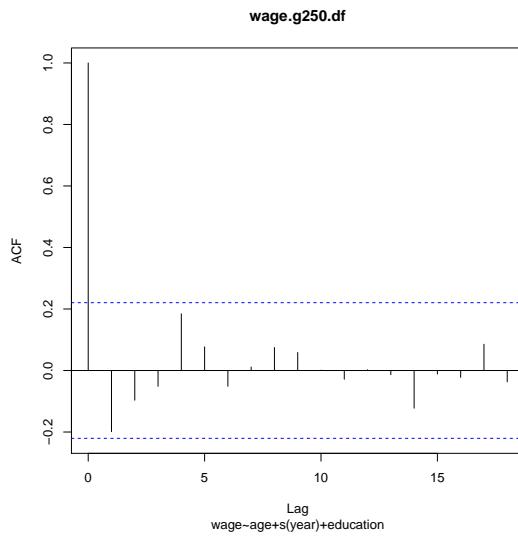
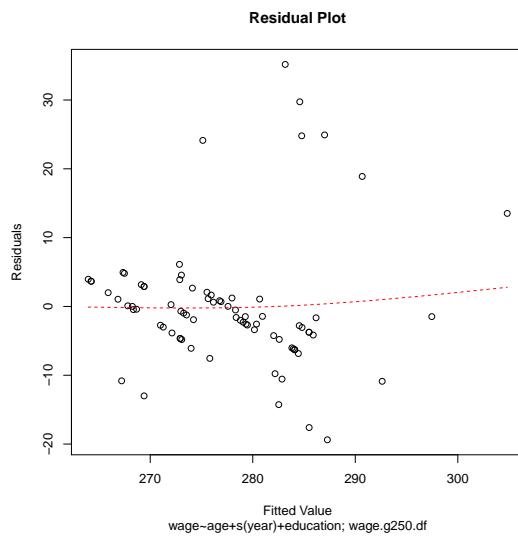
```
> library(gam)
> wage.g250.SMS = gam(wage~age+s(year)+education,
+                         data = wage.g250.df)
```
- The Residual plot shows the linearity assumption is OK, but the amount of variability in `year` is not great, and the variance might not be equal.
- Together with the ACF plot, it seems the independence assumption is fine.
- It is clear that normality is violated, but I could not fix it after trying a few common transformations. I decided to move on which means we should not use

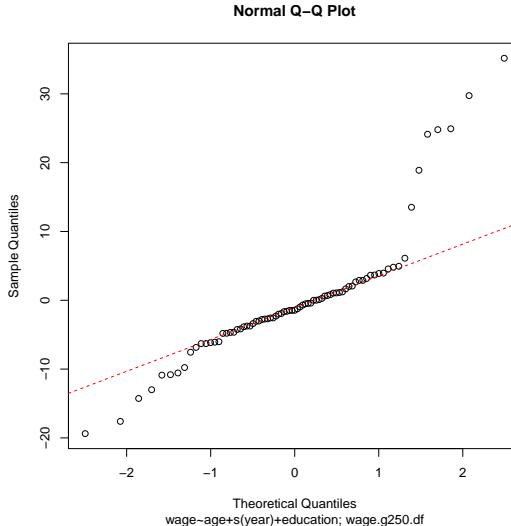
any results that are based on normality, that includes  $t$ -test, etc.

```
> shapiro.test(wage.g250.SMS$residuals)
```

```
Shapiro-Wilk normality test

data: wage.g250.SMS$residuals
W = 0.82966, p-value = 4.131e-08
```





```
> summary(wage.g250.SMS)
```

```

Call: gam(formula = wage ~ age + s(year) + education, data = wage.g250.df)
Deviance Residuals:
    Min      1Q  Median      3Q      Max 
-19.378 -4.198 -1.445  2.034  35.176 

(Dispersion Parameter for gaussian family taken to be 96.4459)

Null Deviance: 11930.06 on 78 degrees of freedom
Residual Deviance: 6751.217 on 70.0001 degrees of freedom
AIC: 595.5866

Number of Local Scoring Iterations: 2

Anova for Parametric Effects
   Df Sum Sq Mean Sq F value    Pr(>F)    
age     1  110.6  110.59  1.1467    0.2879    
s(year) 1  100.3  100.35  1.0404    0.3112    
education 3 2641.1  880.37  9.1282 3.545e-05 ***
Residuals 70 6751.2   96.45                              

Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

Anova for Nonparametric Effects
   Npar Df F       Pr(F)    
(Intercept) 1 10.085 1.324e-05 *** 
age          1 110.6  110.59  1.1467    0.2879    
s(year)      1 100.3  100.35  1.0404    0.3112    
education    3 2641.1  880.37  9.1282 3.545e-05 *** 
Residuals    70 6751.2   96.45                              

Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

```

- Recall GLMs assumes  $Y$  follows a distribution from the exponential family

$$\mu_i = \mathbb{E}[Y_i | \mathbf{X}_i]$$

where  $\mu_i$  is modelled by a linear predictor via  $\eta_i = m^{-1}(\mu_i)$  and

$$\eta_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_k x_{ik}$$

- **Generalised Additive Models** (GAMs) are an extension of GLMs by having

$$\eta_i = \beta_0 + m_1(x_{i1}) + m_2(x_{i2}) + \cdots + m_k(x_{ik})$$

where  $m_j$  are often piecewise smooth functions of  $x_{ij}$ .

- GAMs are an extension of additive models (AMs) which assume

$$Y_i \stackrel{i.i.d.}{\sim} \text{Normal}(\beta_0 + m_1(x_{i1}) + m_2(x_{i2}) + \dots + m_k(x_{ik}), \sigma^2)$$

- GAMs provide a powerful data-driven of models, especially, predictive models
- Recall we struggled to obtain a satisfactory predictive model for the following

wage            Raw wage in the Mid-Atlantic region

age            Age of the worker

year            The year that wage information was recorded

education      A factor with levels:

1. < HS Grad

2. HS Grad

3. Some College

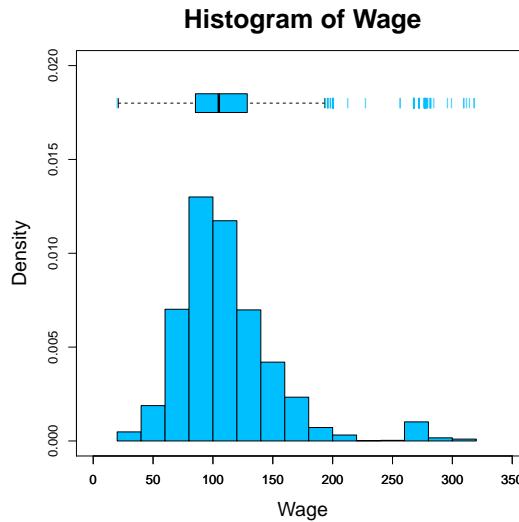
4. College Grad

5. Advanced Degree

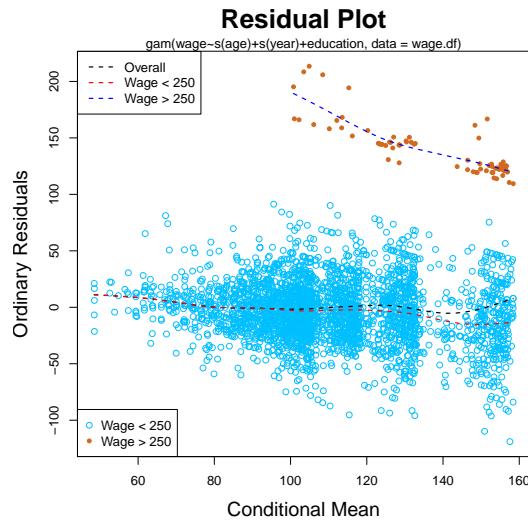
```
> library(ISLR); wage.df =
+   Wage[, c("year", "age", "education", "wage")]
> str(wage.df)
```

```
'data.frame': 3000 obs. of 4 variables:
$ year    : int 2006 2004 2003 2003 2005 2008 2009 2008 2006 2004 ...
$ age     : int 18 24 45 43 50 54 44 30 41 52 ...
$ education: Factor w/ 5 levels "1. < HS Grad",...
$ wage    : num 75 70.5 131 154.7 75 ...
```

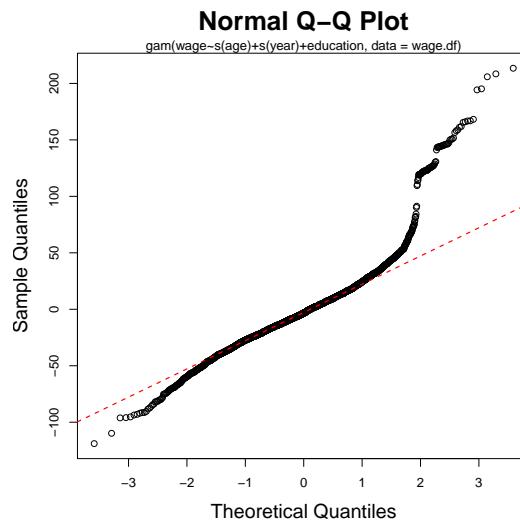
- Recall the outliers/extreme values in the wage was one of the concerns.



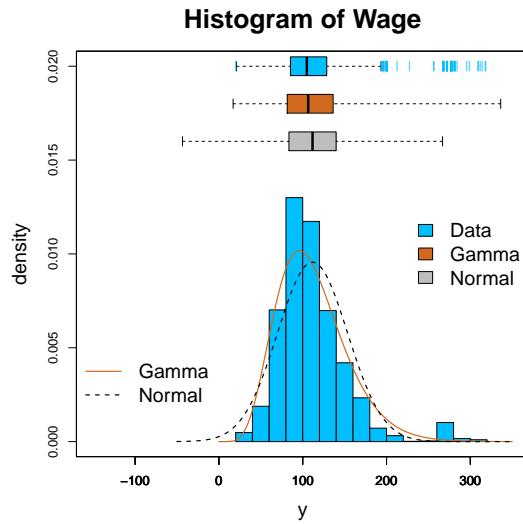
- An ordinary additive model is far from satisfactory.



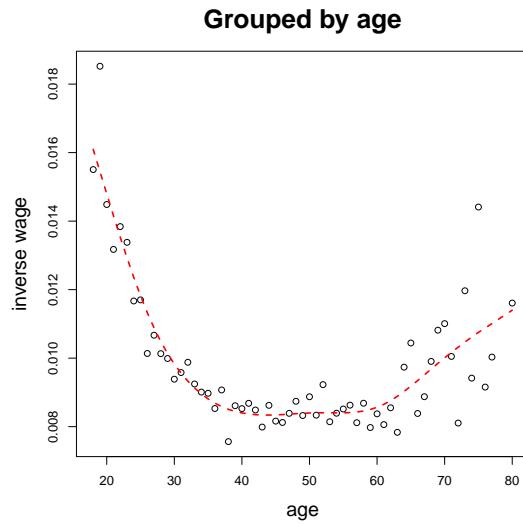
- It is clear the errors are not normally distributed, and severely right skewed.



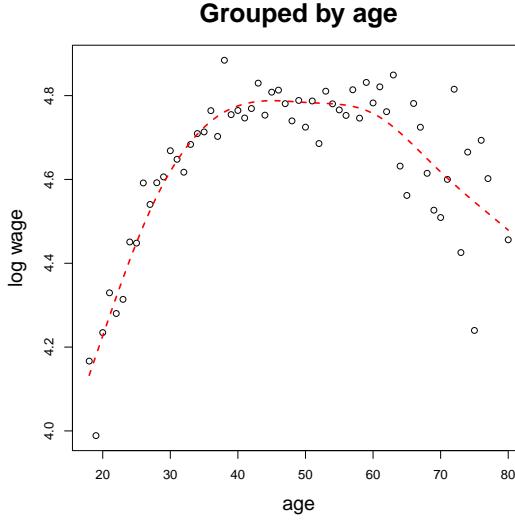
- Gamma is more flexible and more suitable for modelling wage.



- It seems gamma regression under GLM with inverse link is not appropriate.



- It seems gamma regression under GLM with inverse link is not appropriate.



- Instead of trying various transformation on  $x_{ij}$ , let us use smoothing spline

```
> library(gam)
>
> wage.inv.GAM =
+   gam(wage~s(age)+s(year)+education,
+        family = Gamma(link = "inverse"),
+        data = wage.df)
>
> wage.log.GAM =
+   gam(wage~s(age)+s(year)+education,
+        family = Gamma(link = "log"),
+        data = wage.df)
```

- Just like GLMs, GAMs do not possess additive residuals to the predictor

$$y_i \neq \beta_0 + \hat{m}_1(x_{i1}) + \hat{m}_2(x_{i2}) + \cdots + \hat{m}_k(x_{ik}) + \hat{\epsilon}_i$$

- In general, [Pearson residuals](#) for GLM and GAM are defined as

$$\hat{\epsilon}_i = \frac{y_i - \hat{\mu}_i}{\sqrt{\text{Var}[\hat{\mu}_i]}}$$

which should approximately have zero mean and constant variance.

```
> res.inv.df = data.frame(
+   cond_m = fitted(wage.inv.GAM),
+   pear_r = residuals(wage.inv.GAM, type = "pearson"))

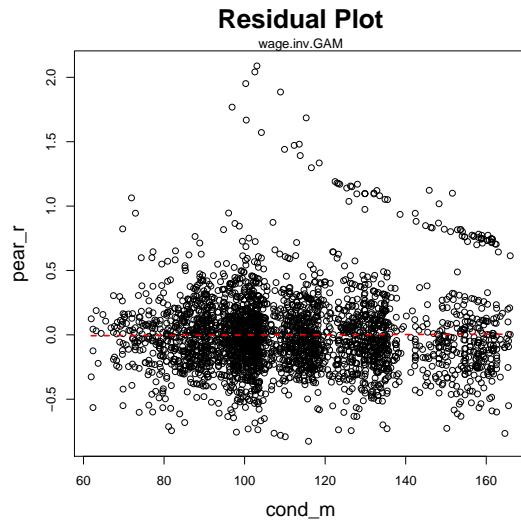
> with(res.inv.df, plot(
+   cond_m, pear_r, main = "Residual Plot",
```

```

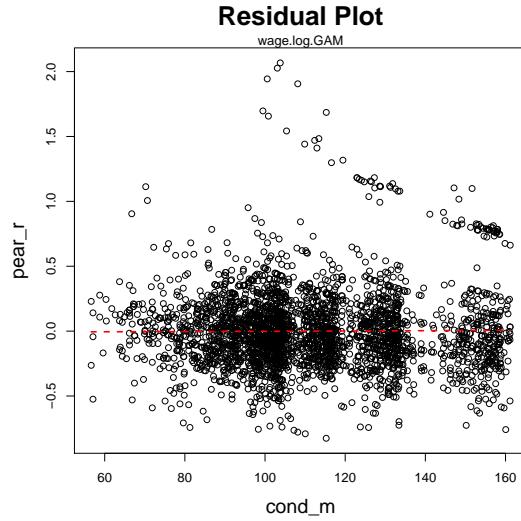
+   cex.lab = 1.5, cex.main = 2))
>
> with(res.inv.df, lines(smooth.spline(
+   cond_m, pear_r), col = "red", lty = 2, lwd = 2))
>
> mtext("wage.inv.GAM")

```

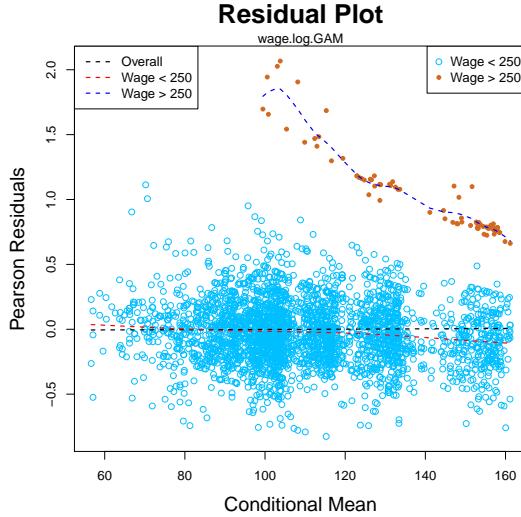
- The residual plot indicates no problem other than the outliers



- We obtain a similar residual plot, so we expect similar results from both



- You might think the residual plot didn't improve much from our first model



- However, both Gamma regression model is far better than the original model

```
> sum(residuals(wage.normal.GAM, type="pearson")^2)
```

```
[1] 3692824
```

```
> sum(residuals(wage.inv.GAM, type="pearson")^2)
```

```
[1] 260.4095
```

```
> sum(residuals(wage.log.GAM, type="pearson")^2)
```

```
[1] 261.5848
```

- Like logistic and Poisson regression, deviance can be used to check goodness of fit, but unlike logistic and Poisson, we have to use the **scaled deviance**

$$D^* = \frac{D}{\hat{\phi}} \sim \chi_{n-(k+1)}^2 \quad \text{where} \quad D = 2(\ell_{sat} - \ell_{prop})$$

and  $\hat{\phi}$  is the dispersion parameter which is given by  $\phi = \frac{1}{\alpha}$  for Gamma.

```
> summary(wage.inv.GAM)
```

```
(Dispersion Parameter for Gamma family taken to be 0.0872)
Null Deviance: 371.6636 on 2999 degrees of freedom
Residual Deviance: 248.1586 on 2987 degrees of freedom
```

```
> 1 - pchisq(248.1586/0.0872, 2987)
```

```
[1] 0.9676302
```

```
> summary(wage.log.GAM)
```

```
(Dispersion Parameter for Gamma family taken to be 0.0876)
Null Deviance: 371.6636 on 2999 degrees of freedom
Residual Deviance: 248.9206 on 2987 degrees of freedom
```

```
> 1 - pchisq(248.9206/0.0876, 2987)
```

```
[1] 0.9715768
```

- However, since the two models are not nested, we CANNOT use deviance based test, Likelihood ratio test, to judge which model is a better one.
- We could still consider AIC of the three models

```
> AIC(wage.normal.GAM, wage.inv.GAM, wage.log.GAM);
```

|                 | df | AIC      |
|-----------------|----|----------|
| wage.normal.GAM | 8  | 29888.23 |
| wage.inv.GAM    | 8  | 29029.51 |
| wage.log.GAM    | 8  | 29038.84 |

which seems to prefer the gamma regression with the inverse link.

- We could also use cross-validation to judge the quality of our models

```
> k = 100 # number of subsamples
> n = nrow(wage.df)
> n.test = n/k
> row.index = sample(1:n, n)
> pred.nor = matrix(0, nrow = n.test, ncol = k)
> pred.inv = matrix(0, nrow = n.test, ncol = k)
> pred.log = matrix(0, nrow = n.test, ncol = k)

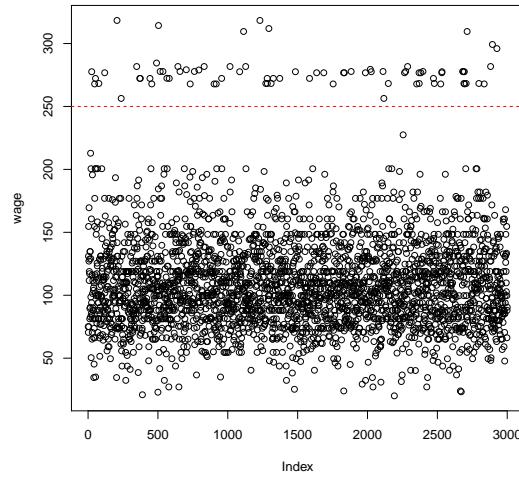
> for (i in 1:k){
+   start = (i-1)*n.test+1; end = i*n.test
+   index = row.index[start:end]
+   wage.inv.GAM = gam(wage~s(age)+s(year)+education,
+                      data = wage.df[-index,])
+   pred.inv[, i] = predict(wage.inv.GAM,
+                          wage.df[index,], type = "response")
}

> tmp = pred.nor[1:n] - wage.df[row.index, "wage"]
> mse.nor = mean((tmp)^2)
> tmp = pred.inv[1:n] - wage.df[row.index, "wage"]
> mse.inv = mean((tmp)^2)
> tmp = pred.log[1:n] - wage.df[row.index, "wage"]
> mse.log = mean((tmp)^2)
```

Q: Which one do you think is the best model in terms of MSE estimated by CV?

```
> c(mse.nor, mse.inv, mse.log)
[1] 1241.127 1231.781 1234.101
```

- Recall we didn't find any single variable that can be used to explain



- We could use logistic regression under GAM to see whether collectively

age, year and education

can explain this apparent separation in wage.

```
> wage.LG.GAM =
+   gam(I(wage > 250) ~ s(age) + s(year) + education,
+       data = Wage, family = binomial)

> 1 - pchisq(wage.LG.GAM$deviance,
+             wage.LG.GAM$df.residual)
```

```
[1] 1
```

which means there is no indication of lack of fit.

- This indicates that we probably should consider using mixture models, which will not be covered, but I will upload the slides if you are interested!

## 11 Principal Component Analysis

- For datasets with a large number of independent variables,  $k$ , we often need to “reduce” this number  $k$  before selecting variables for our model.
- For the extreme case, if the number of observations is too small, that is,

$$n < k$$

then it is simply impossible to include all  $k$  of them in the model.

- In general, consider some **centred data** in a matrix of  $n \times k$ ,

$$\mathbf{X}_{n \times k} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1k} \\ x_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ x_{n1} & \cdots & \cdots & x_{nk} \end{bmatrix}$$

that is, the column means are all zero.

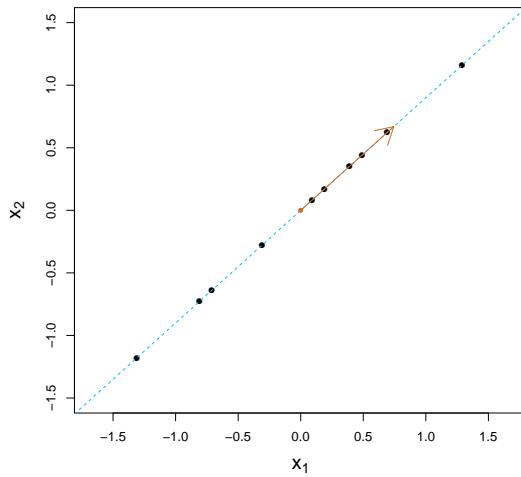
Q: How can we reduce the number  $k$  without losing too much information?

The simplest way of reducing  $k$  is to take just some columns and discard all others.

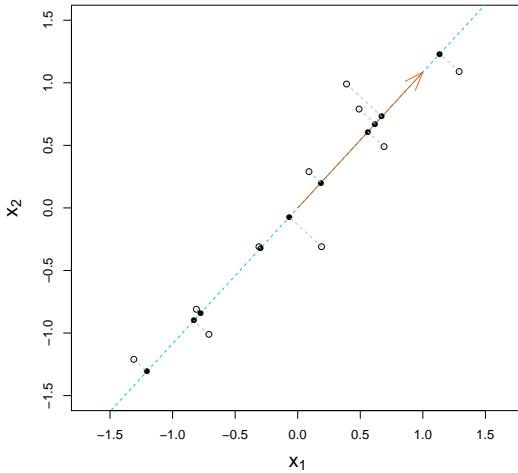
- If there is a perfect multicollinearity, discarding columns is appropriate

$$\alpha_1 x_{i1} + \alpha_2 x_{i2} + \cdots + \alpha_k x_{ik} = 0 \quad \text{for all } i.$$

where  $\alpha_j$  are not all zeros. When  $k = 2$ , another choice seems reasonable.



Q: What if they were not perfect multicollinear? e.g.



- In general, imagine the extreme case of reducing  $k$  to just 1, i.e. replacing

$$\mathbf{X}_{n \times k}$$

by a single vector in  $\mathbf{z} \in \mathbb{R}^n$ , which can be thought as a new variable.

- The idea of PCA originated from identifying the [principal component](#)

$$\mathbf{w}$$

which is a [unit](#) vector in  $\mathbb{R}^k$  that minimises

$$\sum_{i=1}^n \|\mathbf{r}_i - (\mathbf{r}_i^T \mathbf{w}) \mathbf{w}\|^2, \quad \text{where } \mathbf{r}_i^T \text{ denote rows of } \mathbf{X}.$$

and use the following weighted average as the “reduced” data

$$\mathbf{z} = \mathbf{X}\mathbf{w}$$

- It can be show that the minimisation problem is equivalent to

$$\max_{\{\mathbf{w}: \|\mathbf{w}\|=1\}} f(\mathbf{w})$$

where the objective function is given by

$$f(\mathbf{w}) = \|\mathbf{X}\mathbf{w}\|^2$$

- Using Lagrange multiplier, we have a unconstrained optimisation problem

$$\mathcal{L}(\mathbf{w}, \lambda) = \mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} - \lambda (\mathbf{w}^T \mathbf{w} - 1)$$

- Setting the first derivatives to zero, we have the following equations

$$\begin{aligned}\mathbf{X}^T \mathbf{X} \mathbf{w} &= \lambda \mathbf{w} \\ \mathbf{w}^T \mathbf{w} &= 1\end{aligned}$$

Q: Have you seen the first equation before?

- Hence the principal component(s) is a unit eigenvector of

$$\mathbf{X}^T \mathbf{X}$$

- Since the objective function at local optimiser  $\mathbf{w}^*$  can be written as

$$f(\mathbf{w}^*) = \lambda$$

the eigenvector associated to the largest eigenvalue  $\lambda$  is global maximiser.

- Since  $\mathbf{X}^T \mathbf{X}$  is symmetric, there are  $k$  orthonormal eigenvectors for  $\mathbf{X}^T \mathbf{X}$  which in turn means all eigenvalues are non-negative.
- This gives us a way to “reduce”  $\mathbf{X}$  to two columns, or three columns, etc.

$$\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \dots$$

where  $\mathbf{w}_1$  a unit eigenvector associated to the largest eigenvalue, and  $\mathbf{w}_2$  is a unit eigenvector associated to the second largest eigenvalue, etc. They are known as the **first, second, third principal components**, etc.

- Consider the following dataset which is about Swiss bank notes

|          |                                |
|----------|--------------------------------|
| Y        | Genuine (0) or counterfeit (1) |
| Length   | Length mm                      |
| Left     | Width of left edge mm          |
| Right    | Width of right edge mm         |
| Bottom   | Bottom margin width mm         |
| Top      | Top margin width mm            |
| Diagonal | Length of diagonal mm          |

```
> sm.df = read.table("~/Desktop/swiss_money.txt",
+                     sep = "", header = TRUE)

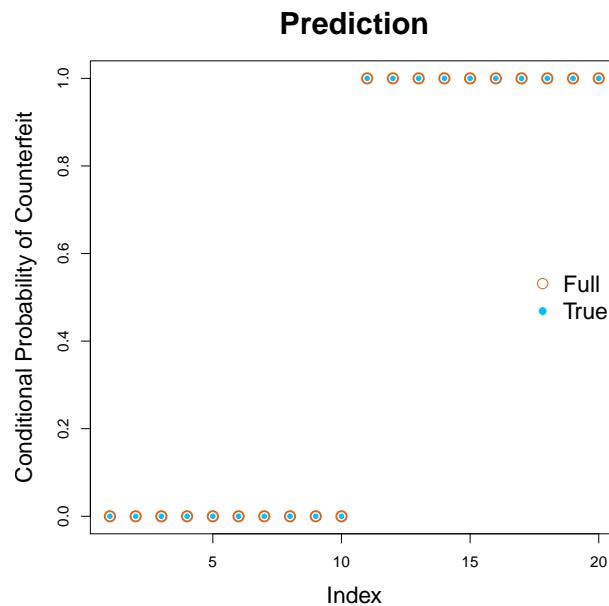
> # Identify rows corresponding to genuine
> row_id_0 = which(sm.df$Y == 0)
> row_id_1 = which(sm.df$Y == 1)

> n = 10 # leave out 10 genuine and 10 counterfeit
>
> # randomly generate those 20 observations
> set.seed(1)
>
> index_0 = sample(row_id_0, size = n)
> index_1 = sample(row_id_1, size = n)
```

```

> library(gam)
> # Full model
> sm.LG =
+   gam(Y~s(Length) + s(Left) + s(Right) + s(Bottom)
+       + s(Top) + s(Diagonal), family = binomial,
+       data = sm.df[-c(index_0, index_1),])
> # Prediction for those 20 observations
> sm_full =
+   predict(sm.LG, sm.df[c(index_0, index_1),],
+           type = "response")

```



```

> # models with one variable at a time
> vname = names(sm.df)
> m = 6 # number of independent variables
> pred = matrix(nrow = 2*n, ncol = m)

> for (i in 1:m){
+   f = as.formula(paste("Y~", vname[i], sep = ""))
+
+   sm.tmp.LG = gam(f, family = binomial,
+                   data = sm.df[-c(index_0, index_1),])
+
+   tmp = predict(
+     sm.tmp.LG, sm.df[c(index_0, index_1),],
+     type = "response")
+
}

```

```

+     pred[, i] = tmp
+
+ }

> # Centering the data matrix
> X = scale(sm.df[, -7], scale = FALSE)

> e = eigen(t(X) %*% X) # solve eigen problem

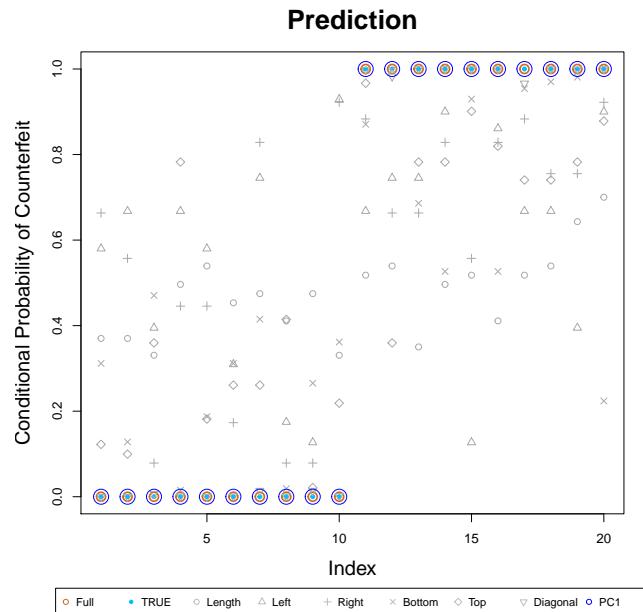
> z1 = X %*% e$vectors[,1] # first principal comp

> # remove testing cases from the ``new data''
> z1.m = z1[-c(index_0, index_1),]

> sm.pc1.LG = gam(
+   Y~s(z1.m), family = binomial,
+   data = sm.df[-c(index_0, index_1),])

> z1.m = z1[c(index_0, index_1),]
> sm_pc1 =
+   predict(sm.pc1.LG, data.frame(z1.m = z1.m),
+           type = "response")

```



- Notice we have converted the minimisation problem

$$\min_{\{\mathbf{w}: \|\mathbf{w}\|=1\}} \sum_{i=1}^n \|\mathbf{r}_i - (\mathbf{r}_i^T \mathbf{w}) \mathbf{w}\|^2$$

to the maximisation problem merely for computation reasons

$$\max_{\{\mathbf{w}: \|\mathbf{w}\|=1\}} \|\mathbf{X}\mathbf{w}\|^2$$

- However, it can be understood as maximising the sample variance of  $z$  since

$$\frac{1}{n-1} \|\mathbf{X}\mathbf{w}\|^2 = s_z^2$$

where  $s_z^2$  denotes the sample variance of  $z$ .

- Note we have centred the data before PCA is performed, when the variables are measured in different units, we need to scaled the data as well.
- To illustrate the scaling effect, consider the the dataset for the 50 US states

|          |                                  |             |
|----------|----------------------------------|-------------|
| Murder   | numeric murder arrests           | per 100,000 |
| Assault  | numeric assault arrests          | per 100,000 |
| UrbanPop | numeric percent urban population | %           |
| Rape     | numeric Rape arrests             | per 100,000 |

```
> var(USArrests) # Covariance matrix
```

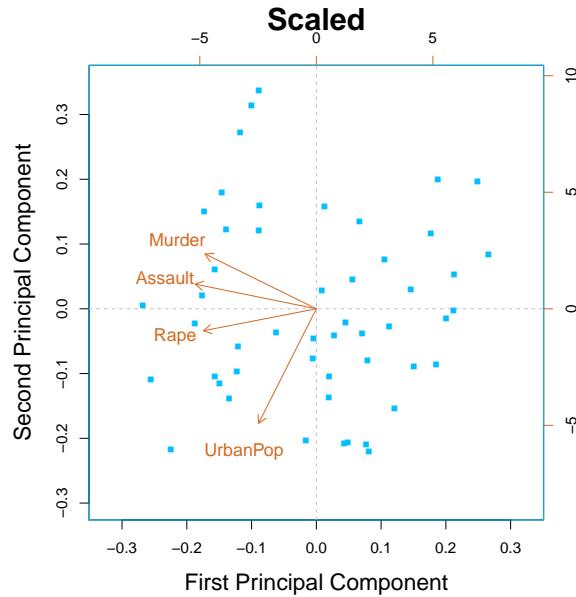
|          | Murder     | Assault   | UrbanPop   | Rape      |
|----------|------------|-----------|------------|-----------|
| Murder   | 18.970465  | 291.0624  | 4.386204   | 22.99141  |
| Assault  | 291.062367 | 6945.1657 | 312.275102 | 519.26906 |
| UrbanPop | 4.386204   | 312.2751  | 209.518776 | 55.76808  |
| Rape     | 22.991412  | 519.2691  | 55.768082  | 87.72916  |

- Notice the variance of **Assault** is dominantly large, followed by **UrbanPop**.
- Of course, R has functions for PCA

```
> USArrests.PCA = prcomp(
+   USArrests, center = TRUE, scale. = TRUE)
```

- Typically, PCA is visualised using a biplot, which depict the plane defined by the first and second eigenvectors with projections of the original data points.

```
> biplot(USArrests.PCA, main = "Scaled",
+         xlab = "First Principal Component",
+         ylab = "Second Principal Component",
+         asp = TRUE, cex.lab = 1.5,
+         cex.main = 2, cex=c(3, 1.2),
+         xlim = c(-0.3, 0.3), ylim = c(-0.3, 0.35),
+         col=c("deepskyblue", "chocolate"),
+         xlabs=rep("ü", nrow(USArrests)))
>
> abline(h = 0, lty = 2, col = "grey")
> abline(v = 0, lty = 2, col = "grey")
```



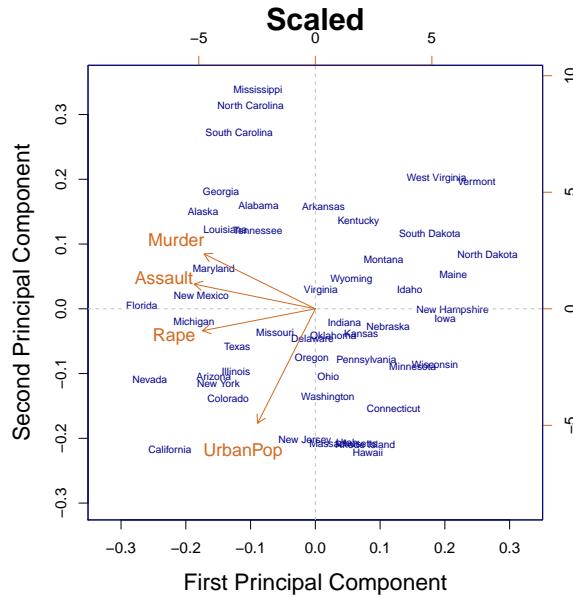
- According to the biplot, the first eigenvector places roughly equal weight on **Assault, Murder and Rape** with less weight on **UrbanPop**.
- So the 1st-pc roughly corresponds to a measure of overall serious crimes.
- The 2nd eigenvector places most of its weight on

**UrbanPop,**

so the 2nd-pc roughly corresponds to the level of urbanisation.

- Furthermore, we see that the crime-related variables are positioned close to each other, which indicates crime-related variables are correlated, that is, states with high murder rates tend to have high assault and rape rates.
- By default, row names are used for each observations in biplot.

Q: What can you conclude?



- If we don't scale, that is, we don't divide the data by the standard deviations

```
> USArrests.u.PCA = prcomp(
```

```
+     USArrests, center = TRUE, scale. = FALSE)
```

```
> sapply(USArrests, var)
```

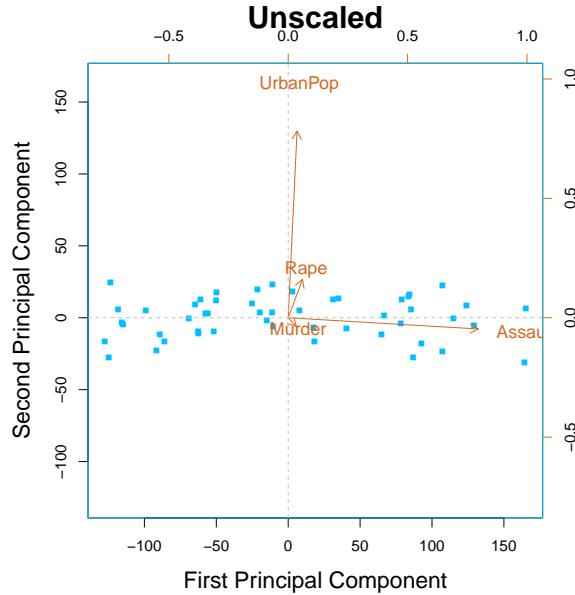
| Murder   | Assault    | UrbanPop  | Rape     |
|----------|------------|-----------|----------|
| 18.97047 | 6945.16571 | 209.51878 | 87.72916 |

- If we put it in the context of maximising variance, it is not surprising that

Assault and UrbanPop

will have the largest weights.

- So we usually scale our data as well as centring them unless they are all measured in the same units, like our Swiss bank note example.



- It is natural to ask how much of the information in a given dataset is lost by using the first  $\ell$  principal components, because it is usually used to decide how many principal components should be included in the investigation.
- Since  $\mathbf{X}^T \mathbf{X}$  is symmetric, there exists  $k$  orthonormal eigenvectors in  $\mathbb{R}^k$

$$\mathcal{B} = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_k\}$$

so the data matrix of all principal components captures all information

$$\mathbf{Z} = \mathbf{XW}$$

where  $\mathbf{W}$  is a matrix with vectors in  $\mathcal{B}$  as its columns.

- The proportion of variability explained by the first  $\ell$ -pc is given

$$R^2 = \frac{\sum_{j=1}^{\ell} s_{z_j}^2}{\sum_{j=1}^k s_{z_j}^2} = \frac{\sum_{j=1}^{\ell} \lambda_j}{\sum_{j=1}^k \lambda_j}$$

- Consider the following data which contains expression levels on 6830 genes from 64 cancer cell lines. Cancer type is also recorded.

```
> gene.df = read.table("~/Desktop/NCI60.csv",
+ sep = ",", header = TRUE)

> dim(gene.df)
```

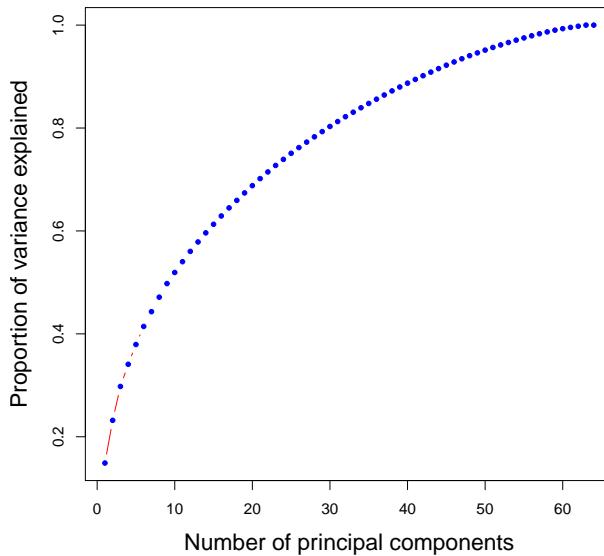
```
[1] 64 6832
```

```
> class.vec = sapply(gene.df, class)
> table(class.vec)
```

```
class.vec
factor numeric
2      6830
```

```
> index.rm = which(class.vec == "factor")
```

### NCI microarray data



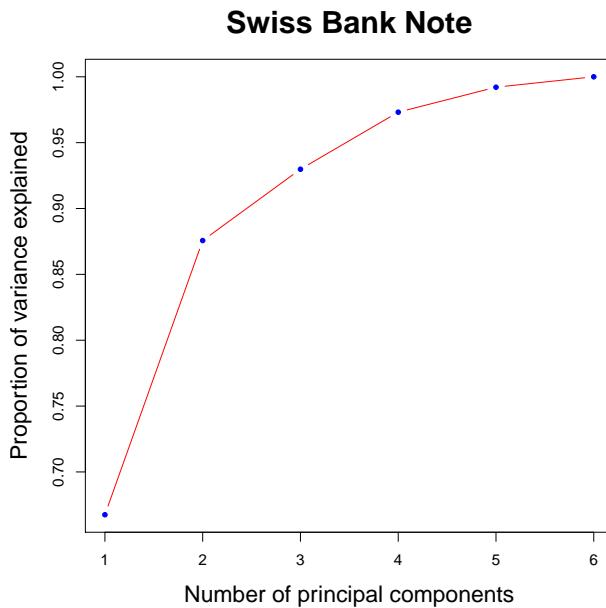
```
> # Remove factor or character variables
> gene.PCA = prcomp(
+   gene.df[, -index.rm], center = TRUE)
>
>
> prop_variance =
+   cumsum(gene.PCA$sdev^2)/sum(gene.PCA$sdev^2)

> plot(prop_variance, col = 2,
+       type = "c", main = "NCI microarray data",
+       xlab = "Number of principal components",
+       ylab = "Proportion of variance explained",
+       cex.lab = 1.5, cex.main = 2)
```

```

>
>
> points(prop_variance, pch = 20, col = 4)

```



## 12 Factor Analysis

- Principal component analysis is about replacing a **centred** and **scaled**  $\mathbf{X}$  by

$$\mathbf{Z}_{n \times \ell} = \mathbf{X}_{n \times k} \mathbf{W}_{k \times \ell} \quad \text{where } \ell = 1, \dots, k$$

and  $\mathbf{W}$  has orthonormal eigenvectors of  $\mathbf{X}^T \mathbf{X}$  as its columns.

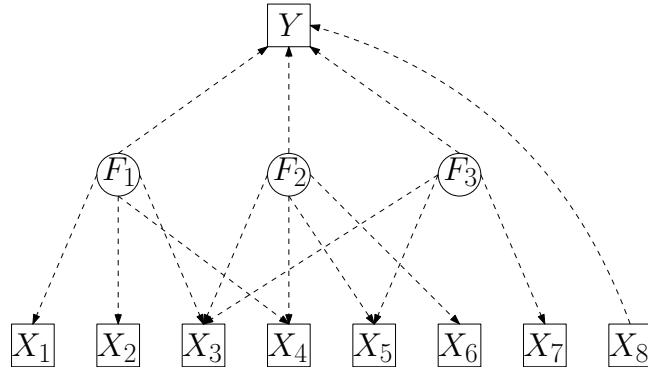
- In reverse, the observed data can be completely reconstructed if  $\ell = k$

$$\mathbf{X}_{n \times k} = \mathbf{Z}_{n \times k} \mathbf{W}_{k \times k}^T$$

- If only the first few eigenvectors are used, that is  $\ell < k$ , then

$$\mathbf{X}_{n \times k} \approx \mathbf{X}_{n \times k}^* = \mathbf{Z}_{n \times \ell} \mathbf{W}_{k \times \ell}^T$$

- Note PCA involves absolutely no probabilistic assumptions at all, thus it says nothing about the stochastic processes that generated the data.
- If we assume there is some **hidden structure** from which the observed  $\mathbf{X}$  are generated, then other dimension reduction methods might be reasonable.



- A **factor model** is used to model such hidden structures

$$\mathbf{X}_{n \times k} = \mathbf{F}_{n \times \ell} \mathbf{W}_{\ell \times k}^T + \boldsymbol{\epsilon}_{n \times k} \quad \text{where } \boldsymbol{\epsilon} \text{ is an error term.}$$

- The observed variables in  $\mathbf{X}$  are known as **manifest** variables and the hidden variables, now denoted collectively as  $\mathbf{F}$  are known as **latent** factors.
- The idea of factor analysis is the variability in a large number of observed variables could be due to a few unobserved factors and unobserved errors.

$$\mathbf{X}_{n \times k} = \mathbf{F}_{n \times \ell} \mathbf{W}_{\ell \times k}^T + \boldsymbol{\epsilon}_{n \times k} \quad \text{where } \boldsymbol{\epsilon} \text{ is an error term.}$$

- By first identify  $\mathbf{W}$  then  $\mathbf{F}$ , dimension reduction is achieved since  $\ell < k$ .

- The following specifications are used for a typical factor model:

1. The manifest variables have zero mean and unit variance.
2. The latent factors have zero mean and unit variance.
3. The error terms have zero mean.
4. The latent factors are uncorrelated

across observations and with other latent factors.

5. The error terms are uncorrelated

across observations, across manifest variables and across latent factors.

- Under the assumptions 4-5 of the factor model, the manifest variables are correlated with each other solely due to correlation with the latent factors.

$$\text{Cov}[X_{i1}, X_{i2}] = \sum_{p=1}^{\ell} w_{1p} w_{2p}$$

where  $w_{1p}$  and  $w_{2p}$  are elements in the first and the second row of  $\mathbf{W}$ , since weights now are the covariance/correlation of manifest and factor variables

$$\text{Cov}[X_{ij}, F_{iq}] = w_{jq}$$

- This leads us to consider the covariance matrix  $\Sigma$  in order to compute  $\mathbf{W}$ ,

$$\Sigma = \mathbb{E}[\mathbf{C}] = \frac{1}{n-1} \mathbb{E}[\mathbf{X}^T \mathbf{X}] = \psi + \mathbf{W} \mathbf{W}^T$$

where  $\mathbf{C}$  is the sample covariance matrix and  $\psi$  is a diagonal matrix.

- A natural step next is to use the sample covariance matrix,

$$\mathbf{C} = \psi + \mathbf{W} \mathbf{W}^T$$

and try to solve the system of equations obtained from the above equality.

- For example, consider the case  $k = 3$  and  $\ell = 1$ , which corresponds

$$\begin{bmatrix} 1 & c_{12} & c_{13} \\ c_{21} & 1 & c_{23} \\ c_{31} & c_{32} & 1 \end{bmatrix} = \begin{bmatrix} \psi_1 & 0 & 0 \\ 0 & \psi_2 & 0 \\ 0 & 0 & \psi_3 \end{bmatrix} + \begin{bmatrix} w_{11} \\ w_{21} \\ w_{31} \end{bmatrix} \begin{bmatrix} w_{11} & w_{21} & w_{31} \end{bmatrix}$$

which from the weights/covariances/correlations  $w_{ji}$  are given by

$$w_{11}^2 = \frac{c_{12}c_{13}}{c_{23}}; \quad w_{21}^2 = \frac{c_{12}c_{23}}{c_{13}}; \quad w_{31}^2 = \frac{c_{13}c_{23}}{c_{12}}$$

and

$$\psi_1 = 1 - w_{11}^2; \quad \psi_2 = 1 - w_{21}^2; \quad \psi_3 = 1 - w_{31}^2$$

- Once  $\mathbf{W}_{3 \times 1}$  is available, we can estimate  $\mathbf{F}_{n \times 1}$  using “regression”

$$\mathbf{X}_{n \times 3} \mathbf{W}_{3 \times 1} = \hat{\mathbf{F}}_{n \times 1} + \epsilon_{n \times 3} \mathbf{W}_{3 \times 1}$$

it can be shown the optimal linear solution which minimises

$$\|\epsilon_{n \times 3} \mathbf{W}_{3 \times 1}\|^2$$

is given by

$$\hat{\mathbf{F}}_{n \times 1} = \mathbf{X}_{n \times 3} \mathbf{W}_{3 \times 1}$$

- However, in general it is not that simple, notice for an orthogonal matrix  $\mathbf{Q}$ ,

$$\begin{aligned} \mathbf{X}_{n \times k} &= \mathbf{F}_{n \times \ell} \mathbf{W}_{\ell \times k}^T + \epsilon_{n \times k} \\ \mathbf{X}_{n \times k} &= (\mathbf{F}_{n \times \ell} \mathbf{Q}_{\ell \times \ell}) (\mathbf{W}_{\ell \times k} \mathbf{Q}_{\ell \times \ell})^T + \epsilon_{n \times k} \end{aligned}$$

which means neither  $\mathbf{W}$  nor  $\mathbf{F}$  is unique when we have more than one factor.

- Having multiple solutions is numerically problematic, but it can be avoided by looking at it slightly differently and employ a numerical stable method

$$\mathbf{C} - \psi = \mathbf{W} \mathbf{W}^T$$

which naturally adds constraints to avoid multiple solutions from rotation.

- For a given  $\psi$ , the left-hand side is surely orthogonally diagonalizable

$$\mathbf{C} - \boldsymbol{\psi} = \mathbf{U}\mathbf{D}\mathbf{U}^T$$

since it is real symmetric, and we can reduce to  $\ell$  factors by finding the first  $\ell$  largest eigenvalues and the corresponding eigenvectors to approximate

$$\mathbf{C} - \boldsymbol{\psi} \approx \tilde{\mathbf{A}}_{k \times k} = \mathbf{U}_{k \times \ell} \mathbf{D}_{\ell \times \ell} \mathbf{U}_{\ell \times k}^T$$

from which we obtain

$$\mathbf{W}_{k \times \ell} = \mathbf{U}_{k \times \ell} \mathbf{D}_{\ell \times \ell}^{1/2}$$

- Once we have  $\mathbf{W}_{k \times \ell}$ , by making a more assumption on data

$$\mathbf{x}_i \sim \text{Normal}(\mathbf{0}, \boldsymbol{\psi} + \mathbf{W}\mathbf{W}^T)$$

we can update  $\boldsymbol{\psi}$  according maximum likelihood estimation, then we iterate by computing  $\mathbf{W}$  again and then  $\boldsymbol{\psi}$  again and again until it converges.

- To illustrate factor analysis, consider the following dataset

```
> ct.df = read.table("~/Desktop/cities.csv",
+                     sep = ",", header = TRUE)
>
> dim(ct.df)
```

```
[1] 21 12
```

```
> names(ct.df)
```

```
[1] "City"      "Area"       "Pop.1980"   "Pop.1990"
[5] "Pop.2000"   "Growth"     "Food"       "PersRoom"
[9] "Water"      "Elec"       "Phones"     "Vehicles"
```

```
> str(ct.df)
```

```
'data.frame': 21 obs. of 12 variables:
$ City      : Factor w/ 21 levels "Beijing","Bombay",...
$ Area      : num 700 100 650 800 1680 ...
$ Pop.1980: int 9918 3290 8789 12101 9029 11739 7268 6852 8067 9030 ...
$ Pop.1990: int 11448 6578 10948 18119 10867 13447 9249 8633 12223 10741 ...
$ Pop.2000: int 12822 11511 12162 22552 14366 17407 12508 10761 18142 12675 ...
$ Growth    : num 1.4 7.2 2.2 4.1 1.9 ...
$ Food      : int 40 63 26 50 52 55 52 47 57 60 ...
$ PersRoom: num 1.3 2.4 0.8 0.8 1.2 ...
$ Water     : int 80 60 88 100 88 95 80 91 92 51 ...
$ Elec      : int 91 85 98 100 90 95 84 98 78 63 ...
$ Phones    : int 14 2 8 16 2 4 4 4 5 2 ...
$ Vehicles: int 1000 600 4000 4000 308 148 200 939 588 500 ...
```

```
> # remove city names
> X = data.matrix(ct.df[, 2:12])

> # use city names as row names
> rownames(X) = ct.df[, 1]
```

```
> ct.FA = factanal(X, factors = 2, rotation="none",
+ scores="regression")
```

```
> ct.FA
```

```
Call:
factanal(x = X, factors = 2, scores = "regression", rotation = "none")

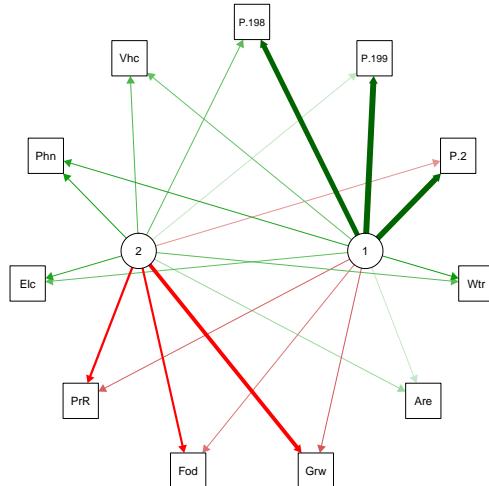
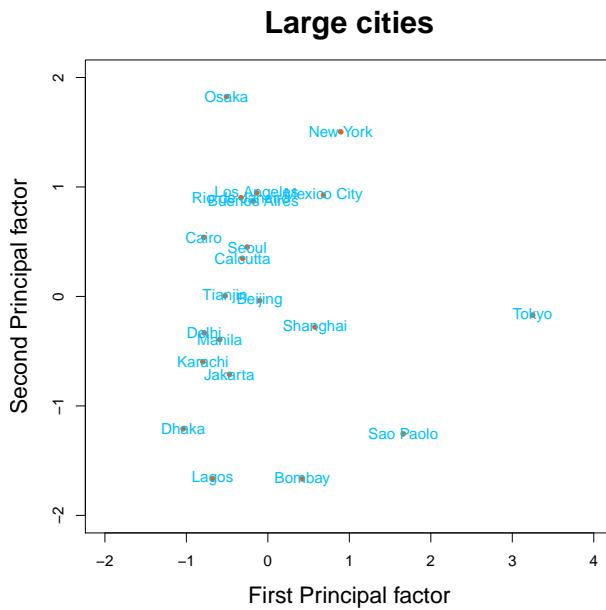
Uniquenesses:
      Area Pop.1980 Pop.1990 Pop.2000   Growth    Food PersRoom     Water
0.933    0.022   0.005   0.005    0.179   0.437   0.387   0.541
      Elec  Phones Vehicles
0.605    0.426   0.706

Loadings:
          Factor1 Factor2
Area        0.129   0.225
Pop.1980    0.914   0.377
Pop.1990    0.988   0.136
Pop.2000    0.967  -0.243
Growth     -0.386  -0.820
Food       -0.316  -0.681
PersRoom   -0.368  -0.691
Water      0.558   0.384
Elec       0.366   0.510
Phones     0.518   0.553
Vehicles   0.356   0.409
```

- The output **Uniquenesses** tells us what fraction of the variance in each observable comes from its own error/noise, that is, the diagonals of  $\psi$ .

```
> # The two factors are stored under scores
> ct.FA$scores
```

```
          Factor1 Factor2
Buenos Aires -0.17889040  0.871144421
Dhaka        -1.03956967 -1.209158145
Rio de Janeiro -0.32938587  0.902073847
Sao Paolo     1.66151441 -1.255368200
Beijing       -0.09985343 -0.037292081
Shanghai      0.57846585 -0.279218251
Tianjin        -0.52473954  0.005131718
Cairo          -0.78794455  0.540001986
Bombay         0.42386495 -1.666913485
Calcutta      -0.30956389  0.348401654
Delhi          -0.78137675 -0.333084139
Jakarta        -0.47221026 -0.711526317
Osaka          0.50879573  1.824775539
Tokyo          3.25066807 -0.170749103
Seoul          -0.25369592  0.450506136
Mexico City    0.68217916  0.925132375
Lagos          -0.67890390 -1.663405574
Karachi        -0.79825159 -0.596300773
Manila         -0.59030193 -0.393923970
Los Angeles    -0.13777899  0.946137238
New York       0.89456997  1.503635125
```



- Notice the first factor from the following two models are not the same,

```

> ct.FA = factanal(X, factors = 2, rotation="none",
+ scores="regression")
> ct.1.FA = factanal(X,factors = 1,rotation="none",

```

```

+           scores="regression")
> cbind(ct.FA$loadings, ct.1.FA$loadings)

          Factor1    Factor2    Factor1
Area      0.1288805  0.2253654  0.1840646
Pop.1980  0.9142481  0.3771544  0.9959482
Pop.1990  0.9884345  0.1356698  0.9589555
Pop.2000  0.9673852 -0.2433919  0.7995058
Growth    -0.3856607 -0.8198362 -0.6800145
Food      -0.3156484 -0.6809724 -0.5238999
PersRoom  -0.3677294 -0.6914914 -0.5697535
Water     0.5581297  0.3836780  0.6317289
Elec      0.3662733  0.5103482  0.4915450
Phones    0.5177603  0.5533497  0.6758071
Vehicles  0.3558366  0.4094097  0.4306993

```

- Similar to what we have defined for PCA, we can define

$$R^2 = \frac{\sum_j \sum_q w_{jq}^2}{k} = \frac{\sum_j k \lambda_j}{k}$$

- However,  $R^2$  should not be used to assess the fit of a factor model
- A likelihood ratio test is provided by the output

```
> ct.FA
```

```

Call:
factanal(x = X, factors = 2, scores = "regression", rotation = "none")

Test of the hypothesis that 2 factors are sufficient.
The chi square statistic is 83.44 on 34 degrees of freedom.
The p-value is 4.86e-06

```

which is similar to the deviance  $\chi^2$  test we had for glm, and can be used to test the fit of a factor model.

## 13 Mixture Model

- In factor analysis, we have implicitly assumed the latent factor is continuous

$$\mathbf{X}_{n \times k} = \mathbf{F}_{n \times 1} \mathbf{W}_{k \times 1}^T + \boldsymbol{\epsilon}_{n \times k}$$

- What if the latent variables are not continuous but categorical? e.g.

$$\begin{aligned} Z &\sim \text{Binomial}(1, 0.2) \\ X | z = 0 &\sim \text{Normal}(0, 9) \\ X | z = 1 &\sim \text{Normal}(1, 9) \end{aligned}$$

which corresponds to

$$\mathbf{X}_{n \times 1} = \mathbf{Z}_{n \times 1} \mathbf{W}_{1 \times 1}^T + \boldsymbol{\epsilon}_{n \times 1}$$

- More generally, the following is known as a [Gaussian mixture model](#).

$$Z \sim \text{Multinomial}(1, \mathbf{p})$$

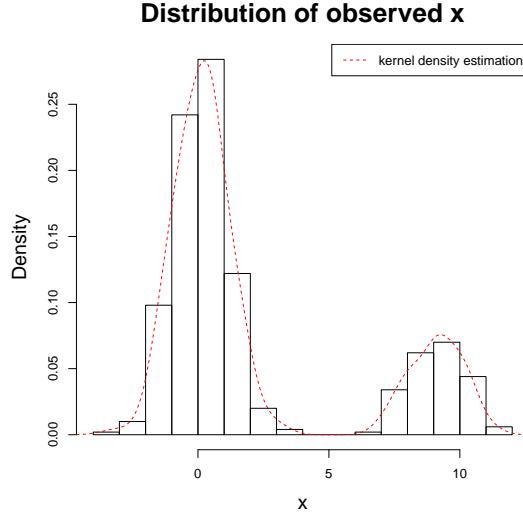
$$X | z = k \sim \text{Normal}(\mu_k, \sigma_k^2)$$

where  $\mathbf{p}$  is a vector probabilities,  $\mathbf{1}^T \mathbf{p} = 1$ , known as the [mixing proportions](#).

- In general, a mixture model assumes  $x_i$  are generated in the following fashion

```
> set.seed(2)
> n = 500
> z.vec = rbinom(n, size = 1, prob = 0.2)
> # z is latent, thus not observed in practice

> # x is observed data
> x.vec = double(n)
>
> for (i in 1:n){
+   if (z.vec[i] == 0) {
+     x.vec[i] = rnorm(1, mean = 0, sd = 1)
+   } else {
+     x.vec[i] = rnorm(1, mean = 9, sd = 1)
+   }
+ }
```



- In terms of probability density function, it means

$$f_{X,Z}(x, z) = f_Z(z)f_{X|Z}(x | z)$$

- Consider the marginal distributions,

$$f_X(x) = \sum_z f_Z(z)f_{X|Z}(x | z) = \sum_z \Pr(Z = z)f_{X|Z}(x | z)$$

- Since the following formula is also true for the joint density

$$f_{X,Z}(x,z) = f_Z(z)f_{X|Z}(x|z) = f_X(x)f_{Z|X}(z|x)$$

we have

$$f_{Z|X}(z|x) = \frac{f_Z(z)f_{X|Z}(x|z)}{f_X(x)} = \frac{f_Z(z)f_{X|Z}(x|z)}{\sum_z \Pr(Z=z)f_{X|Z}(x|z)}$$

Q: What is the significance of this formula?

- Of course, the parameters are unknown and need to be estimated in practice,

$$\begin{aligned} Z &\sim \text{Binomial}(1, p) \\ X | z=0 &\sim \text{Normal}(\mu_0, \sigma_0^2) \\ X | z=1 &\sim \text{Normal}(\mu_1, \sigma_1^2) \end{aligned}$$

- The MLE of  $\theta^T = [p, \mu_0, \mu_1, \sigma_0, \sigma_1]$  cannot be computed in a closed form.

$$\begin{aligned} f_X(x; p, \mu_0, \mu_1, \sigma_0, \sigma_1) &= \sum_z \Pr(Z=z)f_{X|Z}(x|z) \\ &= (1-p) \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(-\frac{(x_i - \mu_0)^2}{2\sigma_0^2}\right) \\ &\quad + p \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{(x_i - \mu_1)^2}{2\sigma_1^2}\right) \end{aligned}$$

Q: What is the likelihood function if we assume observations are independent?

- We can numerically find MLE,

```
> obj_func = function(theta, x) {
+   p      = theta[1]
+   p0    = 1 - p
+   mu0   = theta[2]
+   mu1   = theta[3]
+   sigma0 = theta[4]
+   sigma1 = theta[5]
+
+   pdf    = p0 * dnorm(x, mu0, sigma0) +
+             p * dnorm(x, mu1, sigma1)
+
+   res    = -sum(log(pdf))
+   return(res)
+
+ }

> res.nlm = nlm( # Newton based numerical methods
+   obj_func, c(.25, 10, 10, 10, 10), x.vec)
```

```

> res.nlm # True parameters [0.2, 0, 9, 1, 1]

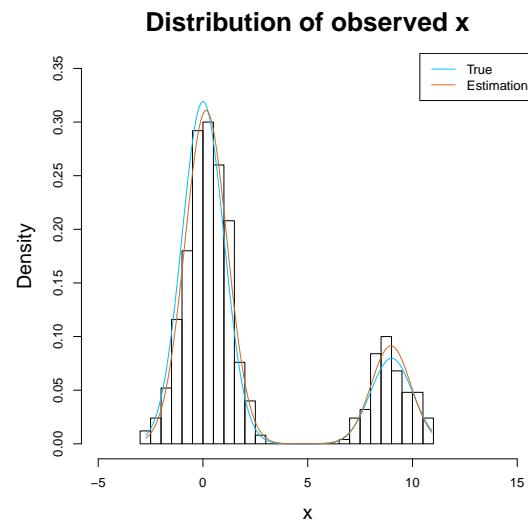
$estimate
[1] 0.2159996 0.1487675 8.9709822 1.0059598 0.9431263

> p      = res.nlm$estimate[1]; p0 = 1 - p
> mu0    = res.nlm$estimate[2]
> mu1    = res.nlm$estimate[3]
> sigma0 = res.nlm$estimate[4]
> sigma1 = res.nlm$estimate[5]

> x.p = seq(min(x.vec), max(x.vec), length = 200)
>
> est.pdf = p0 * dnorm(x.p, mu0, sigma0) +
+   p * dnorm(x.p, mu1, sigma1)

> true.pdf = 0.8 * dnorm(x.p, 0, 1) +
+   0.2 * dnorm(x.p, 9, 1)

```



- Of course, R has a package and a function for mixture models

```

> library(mixtools)
> res = normalmixEM(x.vec)
> res

$lambda
[1] 0.784 0.216

$mu
[1] 0.148769 8.970986

```

```
$sigma
[1] 1.0059601 0.9431269
```

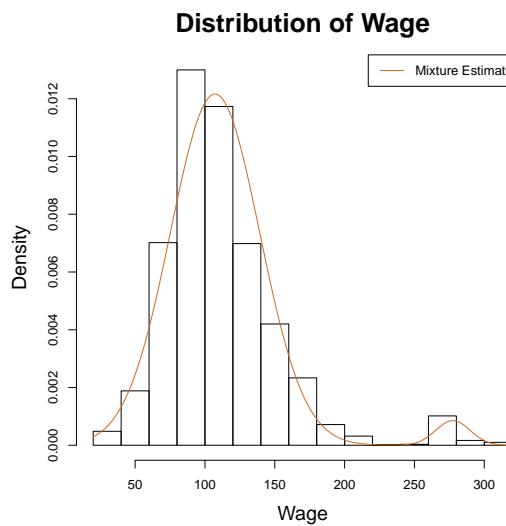
which is more stable and faster than general purpose minimisation routines.

Q: Now we know how to estimate parameters in a mixture model, what is the significance of it in terms of regression?

- Let us finish lectures for this semester by looking at my archenemy again!

|           |                                             |
|-----------|---------------------------------------------|
| wage      | Raw wage in the Mid-Atlantic region         |
| age       | Age of the worker                           |
| year      | The year that wage information was recorded |
|           | 1. < HS Grad                                |
|           | 2. HS Grad                                  |
| education | A factor with levels:                       |
|           | 3. Some College                             |
|           | 4. College Grad                             |
|           | 5. Advanced Degree                          |

```
> library(ISLR);
> wage.df =
+   Wage[, c("year", "age", "education", "wage")]
>
> res = normalmixEM(wage.df$wage)
```



```
> str(wage.df)
```

```
'data.frame': 3000 obs. of 4 variables:  
$ year : int 2006 2004 2003 2003 2005 2008 2009 2008 2006 2004 ...  
$ age : int 18 24 45 43 50 54 44 30 41 52 ...  
$ education: Factor w/ 5 levels "1. < HS Grad",...: 1 4 3 4 2 4 3 3 2 ...  
$ wage : num 75 70.5 131 154.7 75 ...
```

```
> X = as.matrix(wage.df[, 1:2])  
>  
> X = cbind(X, as.integer(wage.df$education))  
>  
> wage.gauss.MM = regmixEM(wage.df[,4], X, k=2)  
  
> summary(wage.gauss.MM)
```

```
summary of regmixEM object:  
            comp 1          comp 2  
lambda 7.48706e-02      0.925129  
sigma  5.42237e+01     25.622410  
beta1  -2.99259e+03   -2120.478107  
beta2  1.48038e+00     1.081827  
beta3  1.72743e+00     0.484611  
beta4  3.92691e+01    12.019221  
loglik at estimate: -14510.38
```

The end of Ve406!