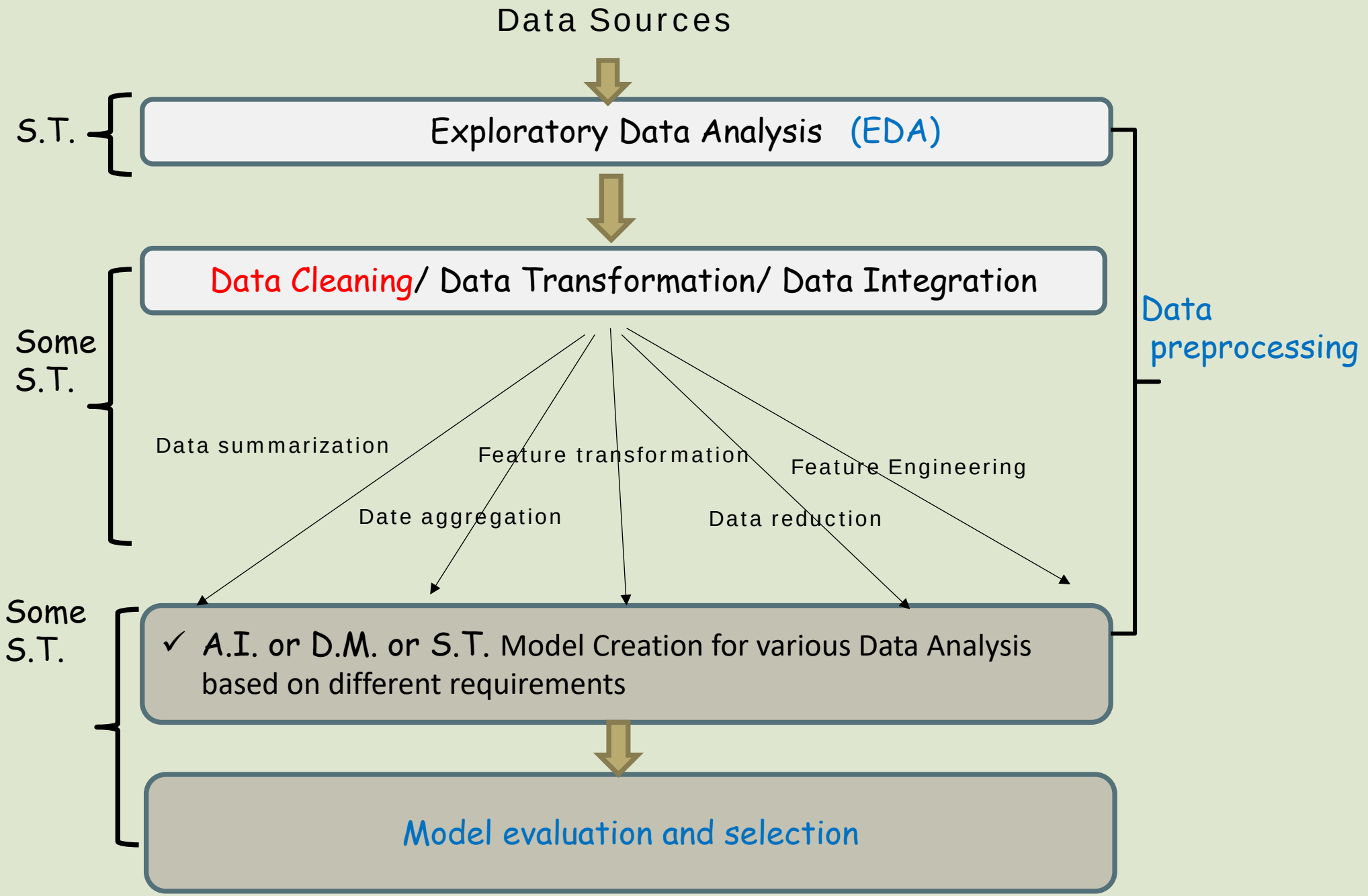
DATA CLEANING


Data Sources
S.T.
Exploratory Data Analysis (EDA)
Some S.T.
Data Cleaning/ Data Transformation/ Data Integration
Data summarization
Date aggregation
Feature transformation
Data reduction
Feature Engineering
Data preprocessing
Some S.T.
✓ A.I. or D.M. or S.T. Model Creation for various Data Analysis based on different requirements
Model evaluation and selection

資料清除（Data Cleaning）

- 資料清除大概是每個資料分析員最痛苦的一件事，整個分析專案的時間大概一半以上都消耗在資料清洗的環節
- 當然現在的統計軟體對於這一塊的處理越來越強大
- 資料有誤，未清除．即使建完模，也不正確
- 另外，我們需知道資料收集的目的,方法,和途徑，這些資訊可以增加對資料集的了解．這了解可以增強建模前的構思，以及模型含意的了解
- Example,慢性阻塞性肺病(Chronic Obstructive Pulmonary Disease，COPD)是一種高致殘率和高致死率的慢性呼吸系統疾病。根據歷史資料建立模型，此模型可協助臨床醫師面對手術病患時，解釋發生術後併發症的風險。根據此模型，臨床醫師於術前面對病患進行風險評估時，將有實證醫學的參考依據，醫護人員並可事先提高警覺，採行必要措施來減少或避免重大術後併發症的發生，對於病患整體的醫療照護，可提供相當大的助益。

資料前置處理

- 所謂的資料前置處理, 如 data cleaning. 是指在進行資料探勘或是人工智慧建模之前，為了讓資料更適合進行建模的工作，對於資料所做的預先處理動作。
- 資料前置處理的主要目的就是解決資料品質不良的問題，使得建模的品質得以提升。
- 在真實的世界中，資料往往沒有想像中的「乾淨」。在實務中，資料會有資料缺失（Imcomplete/ Missing data）、雜訊（Noise）、離異值等等的問題。
- 資料前置處理泛指的是在分析演算法/ 建模之前，先對資料進行處理，讓資料在格式上比較標準一致。為的是讓演算法/ 建模，不會因為資料產生的瑕疵而誤判。
- 資料前處理可以分為以下幾種不同類型的工作：資料清理 （Data Cleaning）、„ 資料整合 （Data Integration）„、資料轉換 （Data Transformation）

Data Cleaning

- 資料清理是資料前處理的第一步，需要先將資料中的問題處理。主要的工作包涵：填補遺失值，處理 Outliers 及 Noise 問題，修正資料的不一致。
- 資料中可能會一些因素造成某些資料不見或不存在，我們會把這樣的缺失稱為遺失值（Missing Data）。
- Generally speaking, 面對 **Missing Data重點處理**:
  - ✓ 可以選擇丟棄或是採用眾數/平均數來取代。
  - ✓ Outlier 及 Noise 的話，必須要先找出哪些點可能是 Outlier/Noise ，然後透過叢集方法/回歸方法來進行**平滑化**。

Data Cleaning

Data Cleaning

- **數據清除**是指發現並糾正資料中可識別的錯誤的方法，包括檢查數據一致性，處理無效值和缺失值等。一般而言，是由電腦而不是人工完成。
- 一般所處理的項目:
  - ✓ **一致性檢查:**一致性檢查(consistency check)是根據每個變數的合理取值範圍，和相互關係，檢查數據是否合乎要求，發現超出正常範圍、邏輯上不合理或者相互矛盾的數據。例如，用1-7級量表測量的變數出現了0值，體重出現了負數，都應視為超出正常值域範圍。另外，如具有邏輯上不一致性的答案可能以多種形式出現：例如，許多調查對象說自己開車上班，又報告沒有汽車；或者調查對象報告自己是某品牌的重度購買者和使用者，但同時又在熟悉程度量表上給了很低的分值。發現不一致時，要列出問卷序號、記錄序號、變數名稱、錯誤類別等，便於進一步核對和糾正

Data Cleaning

- 一般所處理的項目:
  - ✓ 無效值和缺失值的處理:由於調查、編碼和登錄時的誤差，發生數據中可能存在一些無效值和缺失值，需要給予適當的處理。常用的處理方法有：估算，整例刪除，變數刪除和成對刪除。
    - 估算(estimation)。最簡單的辦法就是用某個變數的樣本均值、中位數或眾數代替無效值和缺失值。這種辦法簡單，但沒有充分考慮數據中已有的資料訊息，誤差可能較大。另一種辦法就是根據調查對象對其他問題的答案，通過變數之間的相關分析或邏輯推論進行估計。例如，某一產品的擁有情況可能與家庭收入有關，可以根據調查對象的家庭收入推算擁有這一產品的可能性。
    - 整例刪除(casewise deletion)是剔除含有缺失值的樣本。由於很多問卷都可能存在缺失值，這種做法的結果可能導致有效樣本量大大減少，無法充分利用已經收集到的數據。因此，只適合關鍵變數缺失，或者含有無效值或缺失值的樣本比重很小的情況。
    - 變數刪除(variable deletion)。如果某一變數的無效值和缺失值很多，而且該變數對於所研究的問題不是特別重要，則可以考慮將該變數刪除。這種做法減少了供分析用的變數數目，但沒有改變樣本量。
    - 成對刪除(pairwise deletion)是用一個特殊碼(通常是9、99、999等)代表無效值和缺失值，同時保留數據集中的全部變數和樣本。但是，在具體計算時只採用有完整答案的樣本，因而不同的分析因涉及的變數不同，其有效樣本量也會有所不同。這是一種保守的處理方法，最大限度地保留了數據集中的可用信息。
    - 採用不同的處理方法可能對分析結果產生影響，尤其是當缺失值的出現並非隨機且變數之間明顯相關時。因此，在調查中應當儘量避免出現無效值和缺失值，保證數據的完整性。

Data Cleaning

- 一般所處理的項目:
  - ✓ 無效值和缺失值的處理:由於調查、編碼和登錄時的誤差，發生數據中可能存在一些無效值和缺失值，需要給予適當的處理。常用的處理方法有：估算，整例刪除，變數刪除和成對刪除。
    - 數據清洗原理:數據清洗原理:利用有關技術如數理統計、資料探勘或預定義的清理規則將臟數據轉化為滿足數據質量要求的數據
- 數據清理屬於一個較新的研究領域,直接針對這方面的研究並不多,中文資料清理方面的研究更少。現在的主要為解決兩個問題:發現異常、清理重覆記錄。

Data Cleaning

- 數據清洗原理:數據清洗原理:利用有關技術如數理統計、資料探勘或預定義的清理規則將臟數據轉化為滿足數據質量要求的數據

Data Cleaning

- 數據清洗的方法:一般來說，資料清理從數據的準確性、完整性、一致性、惟一性、適時性、有效性幾個方面來處理數據的丟失值、越界值、不一致代碼、重覆數據等問題。數據清理一般針對具體應用，因而難以歸納統一的方法和步驟，但是根據數據不同可以給出相應的數據清理方法。
  - **解決不完整數據（ 即值缺失）的方法:**大多數情況下，缺失的值必須手工填入（ 即手工清理）。當然，某些缺失值可以從本數據源或其它數據源推導出來，這就可以用平均值、最大值、最小值或更為複雜的概率估計代替缺失的值，從而達到清理的目的。
  - **錯誤值的檢測及解決方法:**用統計分析的方法識別可能的錯誤值或異常值，如偏差分析、識別不遵守分佈或回歸方程的值，也可以用簡單規則庫（ 常識性規則、業務特定規則等）檢查數據值，或使用不同屬性間的約束、外部的數據來檢測和清理數據。
  - **重覆記錄的檢測及消除方法:**資料庫中屬性值相同的記錄被認為是重覆記錄，通過判斷記錄間的屬性值是否相等來檢測記錄是否相等，相等的記錄合併為一條記錄（即合併/清除）。合併/清除是消重的基本方法。
  - **不一致性（ 數據源內部及數據源之間）的檢測及解決方法:**從多數據源集成的數據可能有語義衝突，可定義完整性約束用於檢測不一致性，也可通過分析數據發現聯繫，從而使得數據保持一致。目前開發的數據清理工具大致可分為三類。數據遷移工具允許指定簡單的轉換規則，如：將字元串gender替換成sex。sex公司的PrismWarehouse是一個流行的工具，就屬於這類。
  - 數據清洗工具使用領域特有的知識（ 如，郵政地址）對數據作清洗。它們通常採用語法分析和模糊匹配技術完成對多數據源數據的清理。某些工具可以指明源的“ 相對清潔程度” 。工具Integrity和Trillum屬於這一類。 數據審計工具可以通過掃描數據發現規律和聯繫。因此，這類工具可以看作是數據挖掘工具的變形。

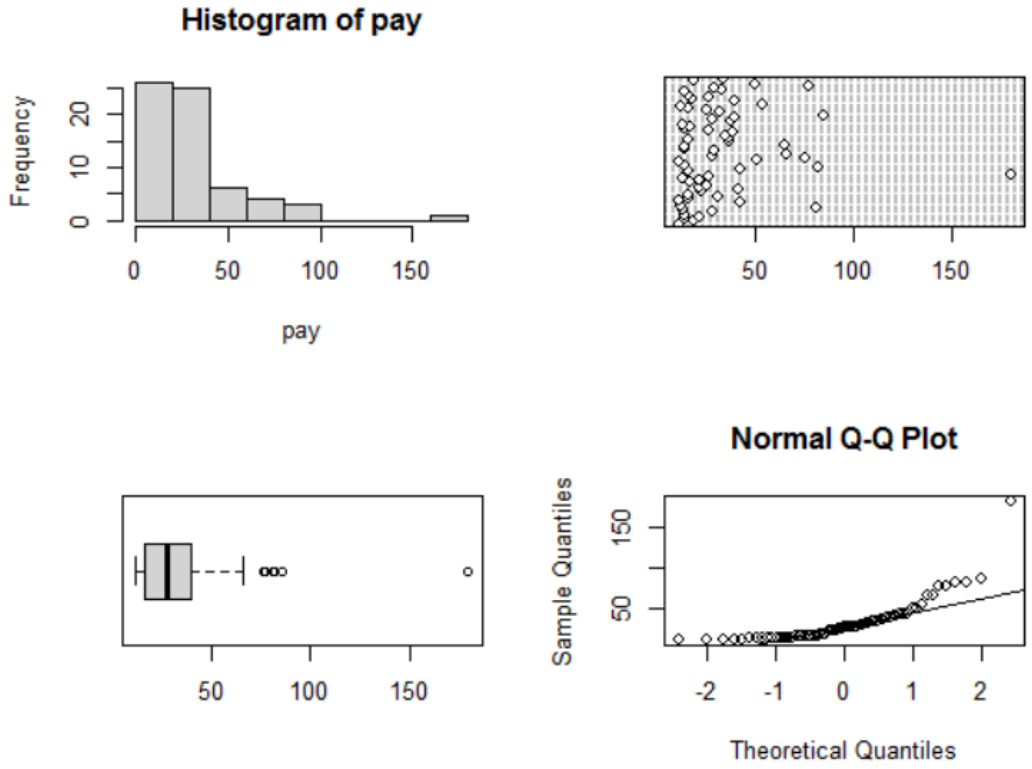
Data Cleaning

- 資料清理,建議可使用底下幾個方法 :

Data Cleaning 處理方式

- 資料清理,建議可使用底下幾個方法 :☺
- Example for Data Cleaning
  - ✓ 對底下66筆資料,作EDA.
  - ✓ 23th 筆資料”180”遠離其他值, 可認為其為異常值(可用描述統計量、Outlier 、Quartile等方法來檢測).
  - ✓ 於EDA中, 對任何異常值皆須解釋兩處理.

```
# example
pay=c(11,19,14,22,14,28,13,81,12,43,
      11,16,31,16,23,42,22,26,17,22,
      13,27,180,16,43,82,14,11,51,76,
      28,66,29,14,14,65,37,16,37,35,
      39,27,14,17,13,38,28,40,85,32,
      25.26,16,12,54,40,18,27,16,14,
      33,29,77,50,19,34)
par(mfrow=c(2,2))
hist(pay)
dotchart(pay)
boxplot(pay,horizontal=T)
qqnorm(pay);qqline(pay)
```



Situation 1:遺漏值的處理(Missing Data Processing) ☺

- 在R語言中的遺漏值,通常以NA表示,可以使用 is.na 函數判斷遺漏值
- 計算遺漏值的數量

```
> summary(nhanes2)
    age          bmi          hyp          chl       
 20-39:12   Min.   :20.40   no  :13   Min.   :113.0  
 40-59: 7   1st Qu.:22.65   yes : 4   1st Qu.:185.0  
 60-99: 6   Median :26.75   NA's: 8   Median :187.0  
            Mean   :26.56             Mean   :191.4  
            3rd Qu.:28.93             3rd Qu.:212.0  
            Max.   :35.30             Max.   :284.0  
            NA's   :9                 NA's   :10     
> sum(is.na(nhanes2))
[1] 27
```

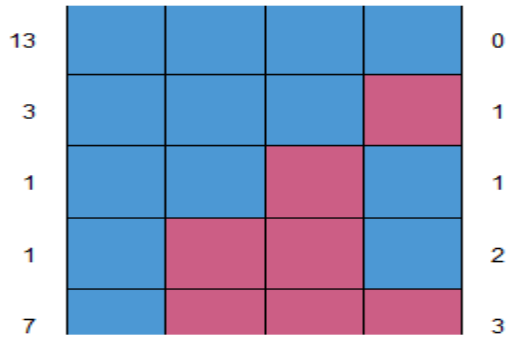
- 計算觀測樣本是否完整, 所以函數 complete.cases()檢測後,可得13筆完整資料. 另外, 25-13=12 可得遺漏值的資料筆數

```
> nrow(nhanes2)
[1] 25
> sum( complete.cases(nhanes2))
[1] 13
```

遺漏值的處理（Missing Data Processing）

- 在存在遺漏值的情況下，需要進一步對資料缺失狀況進行觀察與判斷，判斷缺失資料是否為隨機. Output資訊中，1表示無缺失資料，0表示存在缺失資料. 最左邊一行，表示所對應右邊columns遺失資料狀況的 “頻率”. 而最右邊一行，表示所對應左邊columns遺失資料的 “個數”. 而最下一列，表示所對應該column遺失資料的 “頻率”.
  - ✓ 對於缺失的資料的處理，最直接的處理方法是 移除它們. 但此情況僅適合較少的缺失的資料，並且缺失的資料是隨機出現的狀況的時候. 此時使用此方法，比較不會影響分析的結果.
  - ✓ 對於缺失的資料的處理，也可使用變數平均數或中位數的方式，來替補遺漏的數值，其優點在於不會減少 樣本資訊，缺點是，當遺失資料不是隨機出現時，分析會產商偏差
  - ✓ 對於缺失的資料的處理，也可使用多重插補法(Multivariate Impluation)透過變數間的關係對缺師資料進行預測. 再利用 蒙特卡洛方法(Monte Carlo method)產生多個完整資料集，進而對這些資料集進行分析，最後對分析結果進行處理. ← Main Key

```
> md.pattern(nhanes2)
   age hyp bmi chl
13   1   1   1   1  0
3    1   1   1   0  1
1    1   1   0   1  1
1    1   1   0   0  2
7    1   0   0   0  3
     0   8   9  10 27
```



遺漏值的處理

- Example for mice() - 因nhanes2 data set資料量少，且有遺漏值，無法直接用來建立模型. 但可用mice()產生多個完整資料集，且產生線性迴歸模型. 無欄位 age，因age無遺漏值
- with()函式可依次對每個完整資料集應用統計模型
- pool()函式將這些單獨的分析結果整合為一組結果。最終模型的標準誤和p值都將準確地反映出由於缺失值和多重插補而產生的不確定性。

```
> imp=mice(nhanes2,m=4)

 iter imp variable
  1   1  bmi  hyp  chl
  1   2  bmi  hyp  chl
  1   3  bmi  hyp  chl
  1   4  bmi  hyp  chl
  2   1  bmi  hyp  chl
  2   2  bmi  hyp  chl
  2   3  bmi  hyp  chl
  2   4  bmi  hyp  chl
  3   1  bmi  hyp  chl
  3   2  bmi  hyp  chl
  3   3  bmi  hyp  chl
  3   4  bmi  hyp  chl
  4   1  bmi  hyp  chl
  4   2  bmi  hyp  chl
  4   3  bmi  hyp  chl
  4   4  bmi  hyp  chl
  5   1  bmi  hyp  chl
  5   2  bmi  hyp  chl
  5   3  bmi  hyp  chl
  5   4  bmi  hyp  chl
```

```
> pooled
Class: mipo    m = 4
         term m   estimate         ubar            b           t dfcom
1 (Intercept) 4  35.737563 3019.815272 3782.550654 7748.003589    20
2    age40-59 4  39.346204  368.590940  580.173505 1093.807822    20
3    age60-99 4  60.177987  450.118209 1399.295244 2199.237264    20
4      hypyes 4 -17.247560  382.069256  298.797975  755.566725    20
5         bmi 4   5.273349    3.674973    4.006327    8.682881    20
         df       riv    lambda       fmi
1 3.778759 1.5657210 0.6102460 0.7252387
2 3.235826 1.9675385 0.6630204 0.7710990
3 2.090250 3.8859105 0.7953299 0.8757464
4 5.270135 0.9775648 0.4943276 0.6166164
5 4.162002 1.3627063 0.5767565 0.6949479
```

```
imp=mice(nhanes2,m=4)
fit=with(imp,lm(chl~age+hyp+bmi))
pooled=pool(fit)
summary(pooled)
```

```
         term   estimate std.error  statistic       df   p.value
1 (Intercept)  35.737563 88.022745  0.4060037 3.778759 0.7066714
2    age40-59  39.346204 33.072766  1.1896859 3.235826 0.3140837
3    age60-99  60.177987 46.896026  1.2832215 2.090250 0.3232744
4      hypyes -17.247560 27.487574 -0.6274675 5.270135 0.5565369
5         bmi   5.273349  2.946673  1.7895943 4.162002 0.1452248
```

遺漏值的處理

- 遺漏值的常見處理方法:

```
> sub=which(is.na(nhanes2[,4])==TRUE)
> dataTR=nhanes2[-sub,]
> dataTE=nhanes2[sub,]
> dataTE
     age  bmi  hyp chl
1  20-39   NA <NA>  NA
4  60-99   NA <NA>  NA
10 40-59   NA <NA>  NA
11 20-39   NA <NA>  NA
12 40-59   NA <NA>  NA
15 20-39 29.6   no  NA
16 20-39   NA <NA>  NA
20 60-99 25.5  yes  NA
21 20-39   NA <NA>  NA
24 60-99 24.9   no  NA
```

遺漏值的處理

- 刪除法：若資料足夠建模，可以刪除遺漏值，或直接跳過去計算．例如，

```
> x <- c(12,7,3,4.2,18,2,54,-21,8,-5,NA)
> result.mean <-  mean(x)
> print(result.mean)
[1] NA
> result.mean <-  mean(x,na.rm = TRUE)
> print(result.mean)
[1] 8.22
```

遺漏值的處理

- 差補法：1）隨機取樣本替代遺漏的欄位值

```
> sub=which(is.na(nhanes2[,4])==TRUE)
> dataTR=nhanes2[-sub,]
> dataTE=nhanes2[sub,]
> dataTE
     age  bmi  hyp chl
1  20-39   NA <NA>  NA
4  60-99   NA <NA>  NA
10 40-59   NA <NA>  NA
11 20-39   NA <NA>  NA
12 40-59   NA <NA>  NA
15 20-39 29.6   no  NA
16 20-39   NA <NA>  NA
20 60-99 25.5  yes  NA
21 20-39   NA <NA>  NA
24 60-99 24.9   no  NA
```

```
> #sample non-missing data. Use them as new missing data instead
> dataTE[,4]=sample(dataTR[,4],length(dataTE[,4]),replace=T)
> dataTE
     age  bmi  hyp chl
1  20-39   NA <NA> 184
4  60-99   NA <NA> 118
10 40-59   NA <NA> 187
11 20-39   NA <NA> 199
12 40-59   NA <NA> 113
15 20-39 29.6   no 284
16 20-39   NA <NA> 238
20 60-99 25.5  yes 206
21 20-39   NA <NA> 284
24 60-99 24.9   no 218
```

遺漏值的處理

- 差補法：2)計算所在欄位的非遺漏值的平均值，來替代遺漏的欄位值

```
> sub=which(is.na(nhanes2[,4])==TRUE)
> dataTR=nhanes2[-sub,]
> dataTE=nhanes2[sub,]
> dataTE
     age  bmi  hyp chl
1  20-39   NA <NA>  NA
4  60-99   NA <NA>  NA
10 40-59   NA <NA>  NA
11 20-39   NA <NA>  NA
12 40-59   NA <NA>  NA
15 20-39 29.6   no  NA
16 20-39   NA <NA>  NA
20 60-99 25.5  yes  NA
21 20-39   NA <NA>  NA
24 60-99 24.9   no  NA
```

```
> sub=which(is.na(nhanes2[,4])==TRUE)
> dataTR=nhanes2[-sub,]
> dataTE=nhanes2[sub,]
> dataTE[,4]=mean(dataTR[, 4])
> dataTE  # The 10 missing data in the column chl
     age  bmi  hyp   chl
1  20-39   NA <NA> 191.4
4  60-99   NA <NA> 191.4
10 40-59   NA <NA> 191.4
11 20-39   NA <NA> 191.4
12 40-59   NA <NA> 191.4
15 20-39 29.6   no 191.4
16 20-39   NA <NA> 191.4
20 60-99 25.5  yes 191.4
21 20-39   NA <NA> 191.4
24 60-99 24.9   no 191.4
```

遺漏值的處理

- 差補法: 3) 使用迴歸模型法(linear regression)產生新值 ，來替代遺漏的欄位值

```
> sub=which(is.na(nhanes2[,4])==TRUE)
> dataTR=nhanes2[-sub,]
> dataTE=nhanes2[sub,]
> dataTE
     age  bmi  hyp chl
1  20-39   NA <NA>  NA
4  60-99   NA <NA>  NA
10 40-59   NA <NA>  NA
11 20-39   NA <NA>  NA
12 40-59   NA <NA>  NA
15 20-39 29.6   no  NA
16 20-39   NA <NA>  NA
20 60-99 25.5  yes  NA
21 20-39   NA <NA>  NA
24 60-99 24.9   no  NA
```

```
> lm=lm(chl~age,data=dataTR)
> nhanes2[sub,4]=round(predict(lm,dataTE))
> head(nhanes2)
    age  bmi  hyp chl
1 20-39   NA <NA> 169
2 40-59 22.7   no 187
3 20-39   NA   no 187
4 60-99   NA <NA> 225
5 20-39 20.4   no 113
6 60-99   NA <NA> 184
```

遺漏值的處理

- 差補法：4） 熱平台差補法 在非遺漏值的資料中，找到一個與遺漏值所在樣本相似的樣本(比對樣本)，利用其中的關測值對遺漏值進行插補

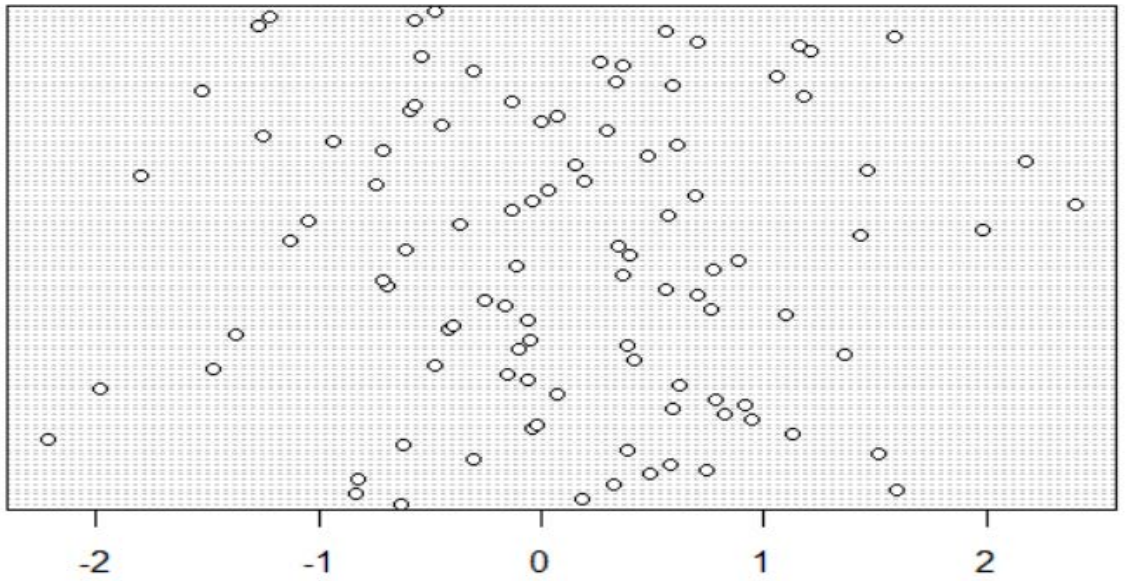
```
> sub=which(is.na(nhanes2[,4])==TRUE)
> dataTR=nhanes2[-sub,]
> dataTE=nhanes2[sub,]
> dataTE
     age  bmi  hyp chl
1  20-39   NA <NA>  NA
4  60-99   NA <NA>  NA
10 40-59   NA <NA>  NA
11 20-39   NA <NA>  NA
12 40-59   NA <NA>  NA
15 20-39 29.6   no  NA
16 20-39   NA <NA>  NA
20 60-99 25.5  yes  NA
21 20-39   NA <NA>  NA
24 60-99 24.9   no  NA
```

```
> #with missing values
> accept=nhanes2[which(apply(is.na(nhanes2),1,sum)!=0),]
> #without missing values
> donate=nhanes2[which(apply(is.na(nhanes2),1,sum)==0),]
> accept[1,]
    age bmi  hyp chl
1 20-39  NA <NA>  NA
> donate[1,]
    age  bmi hyp chl
2 40-59 22.7  no 187
```

Situation 2: 雜訊資料處理

- 尋找最大點的異常值

```
> library(outliers)
> set.seed(1); s1=.Random.seed
> y=rnorm(100)
> outlier(y)
[1] -2.2147
> outlier(y,opposite=TRUE)
[1] 2.401618
> dotchart(y)
```

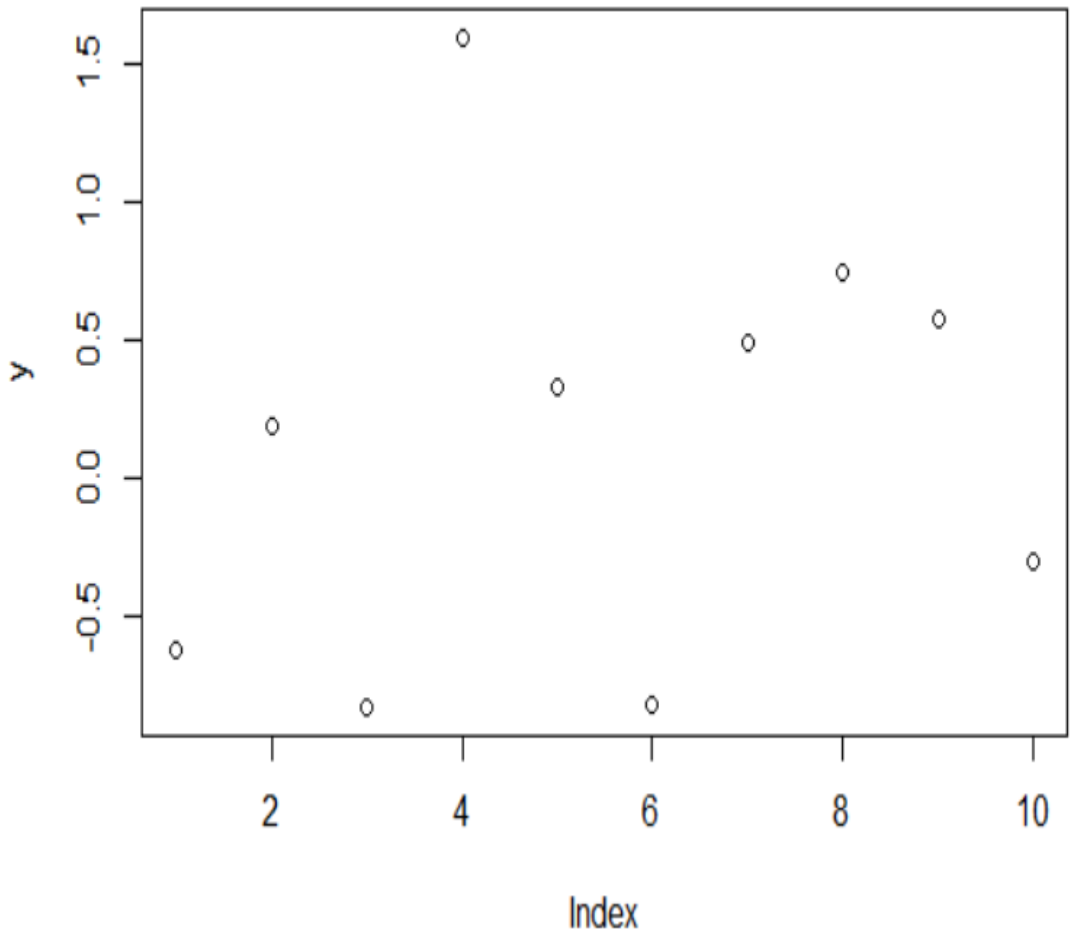



```
> dim(y) <- c(20,5)
> outlier(y)
[1] -2.214700 -1.989352  1.980400  2.401618 -1.523567
```

Situation 2: 雜訊資料處理

- 尋找最大點的異常值

```
> outlier(y,opposite=TRUE)
[1] 2.401618
> set.seed(1); s1=.Random.seed
> y=rnorm(10)
> outlier(y,logical=TRUE)
 [1] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
> plot(y)
```



Situation 2: 雜訊資料處理

- 尋找最大點的異常值

```
> set.seed(1); s1=.Random.seed
> x=rnorm(12)
> x=sort(x)
> dim(x)=c(3,4)
> x[1,]=apply(x,1,mean)[1]
> x[2,]=apply(x,1,mean)[2]
> x[3,]=apply(x,1,mean)[3]
> x
             [,1]         [,2]         [,3]         [,4]
[1,] -0.003212265 -0.003212265 -0.003212265 -0.003212265
[2,]  0.340596290  0.340596290  0.340596290  0.340596290
[3,]  0.468529029  0.468529029  0.468529029  0.468529029
```

Situation 3: 資料不一致的處理

```
> x <- list(a=1:10, beta=exp(-3:3), logic=c(TRUE, FALSE, FALSE, TRUE))
> x
$a
 [1]  1  2  3  4  5  6  7  8  9 10

$beta
[1]  0.04978707  0.13533528  0.36787944  1.00000000  2.71828183  7.38905610
[7] 20.08553692

$logic
[1]  TRUE FALSE FALSE  TRUE

> probs <- c(1: 3/4)
> rt.value <- c(0,0,0)
> vapply(x, quantile, FUN.VALUE=rt.value, probs=probs)
       a      beta logic
25% 3.25 0.2516074   0.0
50% 5.50 1.0000000   0.5
75% 7.75 5.0536690   1.0
```

Situation 3: 資料不一致的處理

```
> probs <- c(1: 4/4)
> vapply(x, quantile, FUN.VALUE=rt.value, probs=probs)
Error in vapply(x, quantile, FUN.VALUE = rt.value, probs = probs) :
  值的長度必須是 3，
但是 FUN(X[[1]]) 的結果長度是 4
```

```
> rt.value <- c(0,0,0, 0)
> vapply(x, quantile, FUN.VALUE=rt.value, probs=probs)
           a        beta logic
25%    3.25   0.2516074   0.0
50%    5.50   1.0000000   0.5
75%    7.75   5.0536690   1.0
100% 10.00 20.0855369   1.0
```

```
> rt.value <- c(0,0,0, "")
> vapply(x, quantile, FUN.VALUE=rt.value, probs=probs)
Error in vapply(x, quantile, FUN.VALUE = rt.value, probs = probs) :
  值的類型必須是 'character'，
但是 FUN(X[[1]]) 的結果類型是 'double'
```

The End