Data Preprocessing

Why data preprocessing?

The raw format of real data tend to be incomplete, noise, and inconsistent.

- Incomplete (missing), e.g. lacking attribute values.
- noisy, e.g. containing errors or outliers.
- inconsistent, e.g. containing discrepancies in codes or names.

Data preparation is a big issue for data mining.

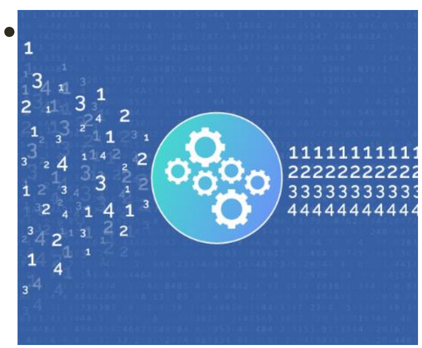
"No quality data, no quality mining results!"

Major Tasks in Data Pre-processing:

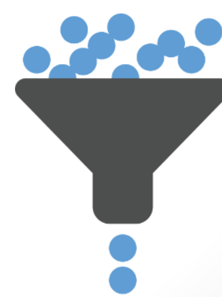
1. Data cleaning.

2. Data integration.

3. Data transformation.



4. Data reduction



1. Data Cleaning:

Data cleaning or data cleansing is the process of identifying and removing (or correcting) inaccurate records from a dataset, table, or database and refers to recognising unfinished, unreliable, inaccurate or non-relevant parts of the data and then restoring, remodelling, or removing the dirty or crude data.

- Data scientists spend about 80% of their time cleaning and manipulating data and only 20% of their time actually analyzing it.

1.1 Missing Data:

Missing values are a common occurrence in real data sets.

- It is important to understand *why* the values are missing.
- Before you can use data with missing data fields, you need to deal with those fields so they can be used for analysis and modeling.

1.1.1 Types of Missing Data:

a. Missing at Random (MAR):

The missingness is not really at random. There is a relationship between the missing value and some of the observed data, but not with the missing data.

Complete data	
Age	IQ score
25	133
26	121
29	91
30	105
30	110
31	98
44	118
46	93
48	141
51	104
51	116
54	97

Incomplete data	
Age	IQ score
25	
26	
29	
30	
30	
31	
44	118
46	93
48	141
51	104
51	116
54	97

1.1.1 Types of Missing Data:

b. Missing Completely at Random (MCAR):

The missingness are randomly distributed across all data (there is no pattern).

Complete data	
Age	IQ score
25	133
26	121
29	91
30	105
30	110
31	98
44	118
46	93
48	141
51	104
51	116
54	97

Incomplete data	
Age	IQ score
25	
26	121
29	91
30	
30	110
31	
44	118
46	93
48	
51	
51	116
54	

1.1.1 Types of Missing Data:

c. Missing Not at Random (nonignorable) (MNAR):

The missingness depends not only on the observed data but also on the unobserved (missing) values.

Complete data	
Age	IQ score
25	133
26	121
29	91
30	105
30	110
31	98
44	118
46	93
48	141
51	104
51	116
54	97

Incomplete data	
Age	IQ score
25	133
26	121
29	
30	
30	110
31	
44	118
46	
48	141
51	
51	116
54	

1.1.2 How Missing Data are Represented in Data set:

- Blank (empty)
- Numbers (999, 99, -1, -99, -9999)
- Codes (NA, N/A, NaN, None, Null, -, .)

Note: R represents missing values using the symbols (NA, NaN, Null)

1.1.3 Detecting the Missing Values in R:

The is.na() function indicates which elements are missing.

- It returns a vector of the same length as the input vector, with a TRUE for any elements that are missing, and a FALSE for non-missing elements.
- is.na() can detect only the blank cells.

The sum() function can then be used to count all of the missing elements.

- We can write it as sum(is.na()).

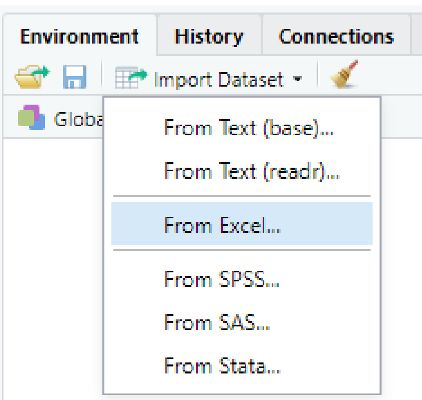
The which() function will return the position of the elements.

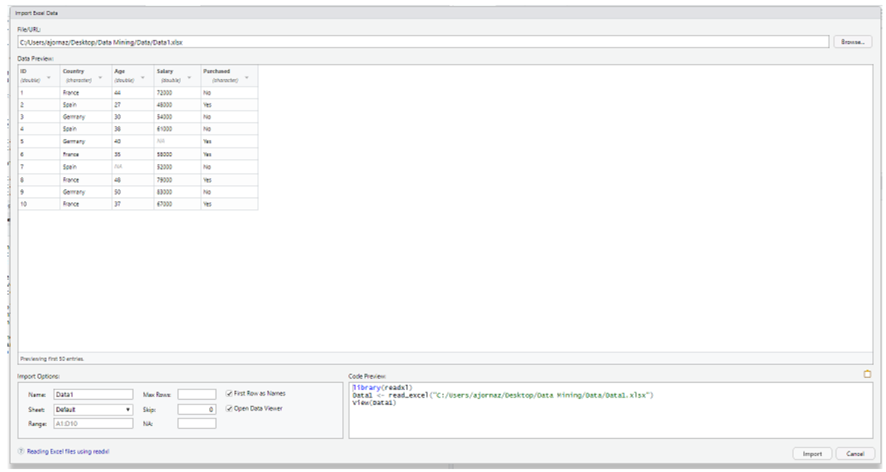
Example:

Dataset "*Data1.xlsx*" is available on canvas.

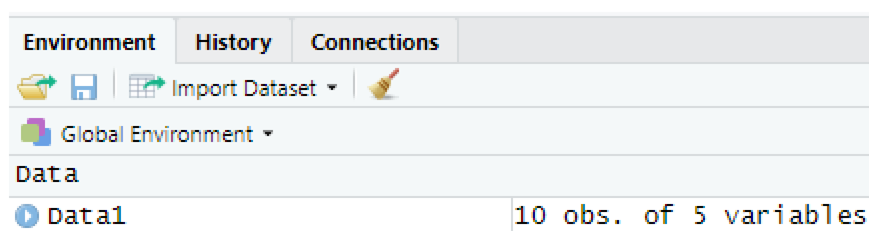
1. Import this file to R.

Go to workspace area > Import Dataset > From Excel > Browse > Import





	ID	Country	Age	Salary	Purchased
1	1	France	44	72000	No
2	2	Spain	27	48000	Yes
3	3	Germany	30	54000	No
4	4	Spain	38	61000	No
5	5	Germany	40	NA	Yes
6	6	France	35	58000	Yes
7	7	Spain	NA	52000	No
8	8	France	48	79000	Yes
9	9	Germany	50	83000	No
10	10	France	37	67000	Yes



Example:

We can observe that there are 2 missing values.

2. Detect the missing values in Data1.

```
> is.na(Data1)
         ID Country   Age Salary Purchased
 [1,] FALSE   FALSE FALSE  FALSE     FALSE
 [2,] FALSE   FALSE FALSE  FALSE     FALSE
 [3,] FALSE   FALSE FALSE  FALSE     FALSE
 [4,] FALSE   FALSE FALSE  FALSE     FALSE
 [5,] FALSE   FALSE FALSE   TRUE     FALSE
 [6,] FALSE   FALSE FALSE  FALSE     FALSE
 [7,] FALSE   FALSE  TRUE  FALSE     FALSE
 [8,] FALSE   FALSE FALSE  FALSE     FALSE
 [9,] FALSE   FALSE FALSE  FALSE     FALSE
[10,] FALSE   FALSE FALSE  FALSE     FALSE
```

```
> sum(is.na(Data1))
[1] 2
```

- sum() function showed that there were 2 missing values.
- We can use summary() function to calculate the summary statistics and the number of missing values.

```
> summary(Data1$Age)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  27.00   35.00   38.00   38.78   44.00   50.00       1
```

Example:

3. Find the position of the missing elements.

```
> which(is.na(Data1))
[1] 27 35
```

```
> which(is.na(Data1$Age))
[1] 7
```

```
> which(is.na(Data1$Salary))
[1] 5
```

We need to specify the attribute (variable).

- The values number 7 under age was missing.
- The values number 5 under salary was missing.

We can determine the percentage of missing values by using,

```
> p = function(Data1) {sum(is.na(Data1))/length(Data1)*100}
> apply(Data1, 2, p)
       ID   Country       Age    Salary Purchased
        0         0        10        10         0
```

10% of age and 10% of salary are missing.

Example:

4. Find the parentage of the missing elements.

```
> p = function(Data1) {sum(is.na(Data1))/length(Data1)*100}
> apply(Data1, 2, p)
       ID   Country       Age    Salary Purchased 
        0         0        10        10         0 
```

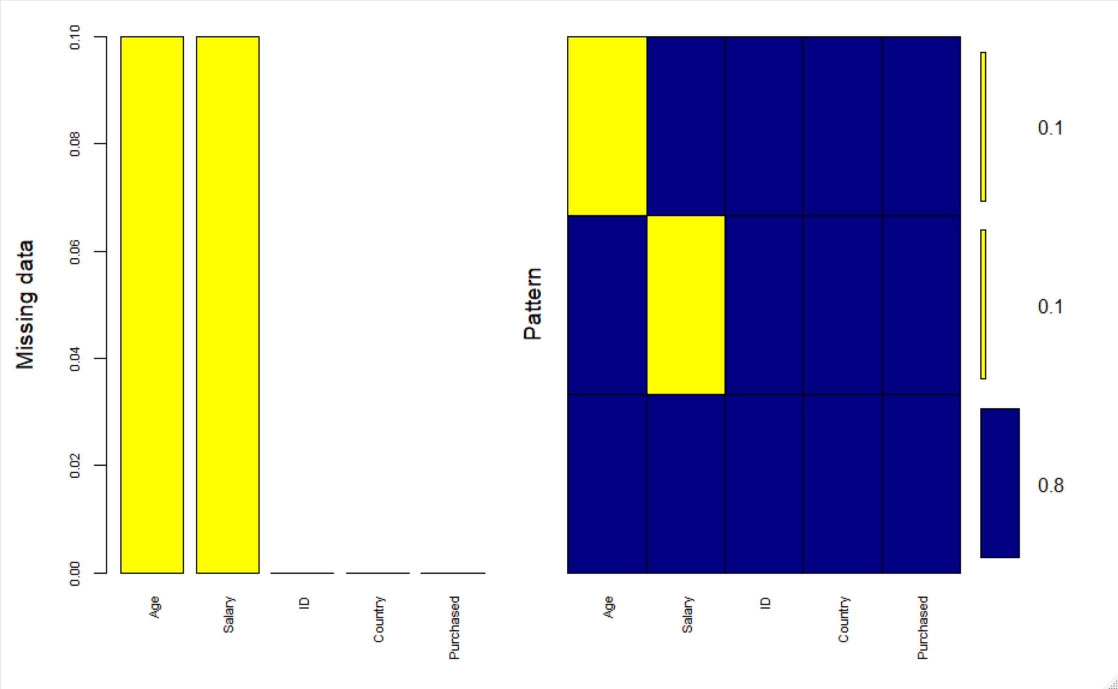
10% of age and 10% of salary are missing.

We can also use library(VIM)

```
> library(VIM)
```

```
> aggr(Data1, col=c('navyblue','yellow'),
+      numbers=TRUE, sortVars=TRUE,
+      labels=names(Data1), cex.axis=.7,
+      gap=3, ylab=c("Missing data","Pattern"))

 Variables sorted by number of missings: 
  Variable Count
       Age   0.1
    Salary   0.1
        ID   0.0
   Country   0.0
 Purchased   0.0
```



Example:

"*Data2.xlsx*" contains some missing values with different symbols.

	ID	Country	Age	Salary	Purchased
1	1	France	44	72000	No
2	2	Spain	27	48000	Yes
3	3	Germany	30	54000	No
4	4	Spain	38	61000	No
5	5	Germany	40	N/A	NA
6	6	France	35	58000	Yes
7	7	Spain	-	52000	No
8	8	France	48	79000	NA
9	9	Germany	50	83000	No
10	10	France	37	67000	Yes

```
> is.na(Data2)
         ID Country   Age Salary Purchased
 [1,] FALSE   FALSE FALSE  FALSE     FALSE
 [2,] FALSE   FALSE FALSE  FALSE     FALSE
 [3,] FALSE   FALSE FALSE  FALSE     FALSE
 [4,] FALSE   FALSE FALSE  FALSE     FALSE
 [5,] FALSE   FALSE FALSE  FALSE     FALSE
 [6,] FALSE   FALSE FALSE  FALSE     FALSE
 [7,] FALSE   FALSE FALSE  FALSE     FALSE
 [8,] FALSE   FALSE FALSE  FALSE      TRUE
 [9,] FALSE   FALSE FALSE  FALSE     FALSE
[10,] FALSE   FALSE FALSE  FALSE     FALSE
```

```
> sum(is.na(Data2))
[1] 1
```

is.na() function couldn't detect the missing values with symbols.

In this case we can use data[data == " "] <- NA.

Example:

	ID	Country	Age	Salary	Purchased
1	1	France	44	72000	No
2	2	Spain	27	48000	Yes
3	3	Germany	30	54000	No
4	4	Spain	38	61000	No
5	5	Germany	40	N/A	NA
6	6	France	35	58000	Yes
7	7	Spain	-	52000	No
8	8	France	48	79000	NA
9	9	Germany	50	83000	No
10	10	France	37	67000	Yes

```
> Data2[Data2 == "-"] <- NA
> Data2[Data2 == "N/A"] <- NA
> Data2[Data2 == "NA"] <- NA
> Data2
# A tibble: 10 x 5
      ID Country Age   Salary Purchased
   <dbl> <chr>   <chr> <chr>  <chr>
 1     1 France  44    72000  No
 2     2 Spain   27    48000  Yes
 3     3 Germany 30    54000  No
 4     4 Spain   38    61000  No
 5     5 Germany 40    NA     NA
 6     6 France  35    58000  Yes
 7     7 Spain   NA    52000  No
 8     8 France  48    79000  NA
 9     9 Germany 50    83000  No
10    10 France  37    67000  Yes
```

1.1.4 Handling Missing Data:

a. Deletion:

- Listwise Deletion: an entire record is excluded from the data if any single value is missing (< 5%).
  - It does not introduce bias if the data are missing completely at random (MCAR).
  - It removes all data for a case that has one or more missing values.

- Pairwise Deletion: Unlike listwise, pairwise deletion only removes the specific missing values from the analysis (not the entire case).

Example:

Subject	Age	Gender	Income
1	29	M	$40,000
2	45	M	$36,000
3	81	M	N/A
4	22	N/A	$16,000
5	41	M	N/A

Listwise Deletion

Subject	Age	Gender	Income
1	29	M	$40,000
2	45	M	$36,000

Subject	Age	Gender	Income
1	29	M	$40,000
2	45	M	$36,000
3	81	M	N/A
4	22	N/A	$16,000
5	41	M	N/A

Pairwise Deletion

Subject	Age	Gender	Income
1	29	M	$40,000
2	45	M	$36,000
3	81	M	
4	22		$16,000
5	41	M	

Listwise Deletion in R:

The na.omit() function removes all incomplete cases (tuple) of a dataset.

Untitled1 × Data1 ×

Filter

	ID	Country	Age	Salary	Purchased
1	1	France	44	72000	No
2	2	Spain	27	48000	Yes
3	3	Germany	30	54000	No
4	4	Spain	38	61000	No
5	5	Germany	40	NA	Yes
6	6	France	35	58000	Yes
7	7	Spain	NA	52000	No
8	8	France	48	79000	Yes
9	9	Germany	50	83000	No
10	10	France	37	67000	Yes

```
> na.omit(Data1)
# A tibble: 8 x 5
     ID Country   Age Salary Purchased
  <dbl> <chr>   <dbl>  <dbl> <chr>
1     1 France     44  72000 No
2     2 Spain      27  48000 Yes
3     3 Germany    30  54000 No
4     4 Spain      38  61000 No
5     6 France     35  58000 Yes
6     8 France     48  79000 Yes
7     9 Germany    50  83000 No
8    10 France     37  67000 Yes
```

The cases (rows) 5 and 7 have been removed.

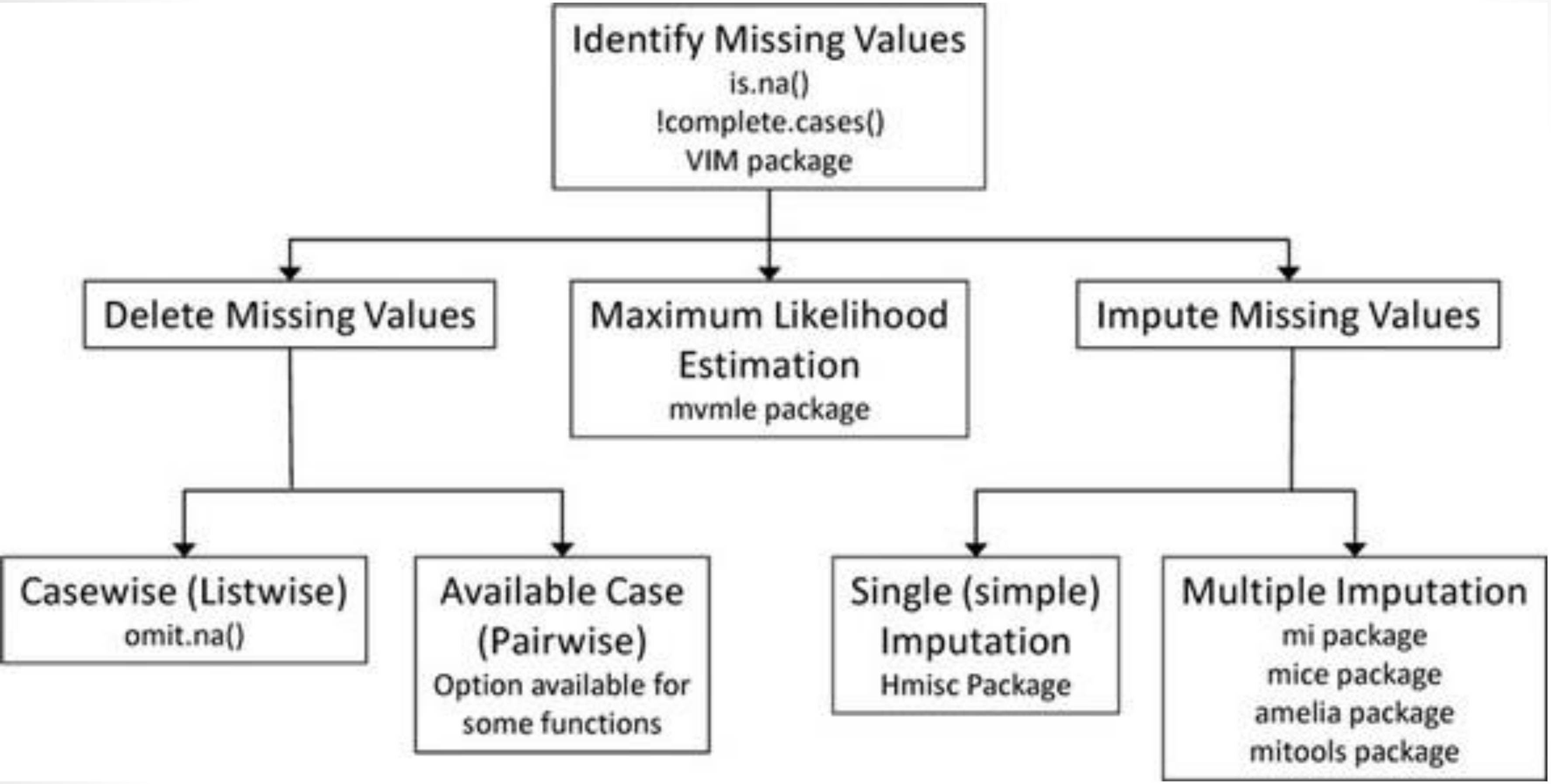
1.1.4 Handling Missing Data:

b. **Imputation:** is the process of replacing missing data with substituted values.

- Use a variety of methods to impute these values varying in accuracy and complexity

Imputation Techniques:

- Mean imputation
- Hot-deck imputation
- Cold-deck imputation
- Regression imputation
- KNN (K-Nearest Neighbors)
- Maximum likelihood
- Multiple imputation
- Other Techniques



https://livebook.manning.com/book/r-in-action/chapter-15/25

1.1.4 Handling Missing Data:

i. Mean (or Median) substitution:

it involves replacing any missing value with the mean (or the median) of that attribute (variable) for all other cases.

Note: In case of missing values in time series data (a series of data points indexed in time order), we can substitute the missing value with the mean of the prior and posterior values.

1.1.4 Handling Missing Data:

Mean Imputation of One Column in R:

data$attribute[is.na(data$attribute)] <- mean(data$attribute, na.rm = TRUE)

Example:

Use mean substitution to replace the missings in *Data1*.

Untitled1 × Data1 ×

Filter

	ID	Country	Age	Salary	Purchased
1	1	France	44	72000	No
2	2	Spain	27	48000	Yes
3	3	Germany	30	54000	No
4	4	Spain	38	61000	No
5	5	Germany	40	NA	Yes
6	6	France	35	58000	Yes
7	7	Spain	NA	52000	No
8	8	France	48	79000	Yes
9	9	Germany	50	83000	No
10	10	France	37	67000	Yes

```
> Data1$Age[is.na(Data1$Age)] <- mean(Data1$Age, na.rm = TRUE)
> Data1$Salary[is.na(Data1$Salary)] <- mean(Data1$Salary, na.rm = TRUE)
> Data1
# A tibble: 10 x 5
      ID Country   Age Salary Purchased
   <dbl> <chr>   <dbl>  <dbl> <chr>
 1     1 France   44    72000 No
 2     2 Spain    27    48000 Yes
 3     3 Germany  30    54000 No
 4     4 Spain    38    61000 No
 5     5 Germany  40    63778. Yes
 6     6 France   35    58000 Yes
 7     7 Spain    38.8  52000 No
 8     8 France   48    79000 Yes
 9     9 Germany  50    83000 No
10    10 France   37    67000 Yes
```

1.1.4 Handling Missing Data:

Median Imputation of One Column in R:

data$attribute[is.na(data$attribute)] <- median(data$attribute, na.rm = TRUE)

Example:

Use Median substitution to replace the missingness in *Data1*.

Untitled1 × | Data1 ×

Filter

	ID	Country	Age	Salary	Purchased
1	1	France	44	72000	No
2	2	Spain	27	48000	Yes
3	3	Germany	30	54000	No
4	4	Spain	38	61000	No
5	5	Germany	40	NA	Yes
6	6	France	35	58000	Yes
7	7	Spain	NA	52000	No
8	8	France	48	79000	Yes
9	9	Germany	50	83000	No
10	10	France	37	67000	Yes

```
> Data1$Age[is.na(Data1$Age)] <- median(Data1$Age, na.rm = TRUE)
> Data1$Salary[is.na(Data1$Salary)] <- median(Data1$Salary, na.rm = TRUE)
> Data1
# A tibble: 10 x 5
      ID Country   Age Salary Purchased
   <dbl> <chr>   <dbl>  <dbl> <chr>
 1     1 France     44  72000 No
 2     2 Spain      27  48000 Yes
 3     3 Germany    30  54000 No
 4     4 Spain      38  61000 No
 5     5 Germany    40  61000 Yes
 6     6 France     35  58000 Yes
 7     7 Spain      38  52000 No
 8     8 France     48  79000 Yes
 9     9 Germany    50  83000 No
10    10 France     37  67000 Yes
```

1.1.4 Handling Missing Data:

List of R Packages:

The most common R packages that handle multiple imputation are:

- MICE
- Amelia
- missForest
- Hmisc
- Mi

https://medium.com/coinmonks/dealing-with-missing-data-using-r-3ae428da2d17

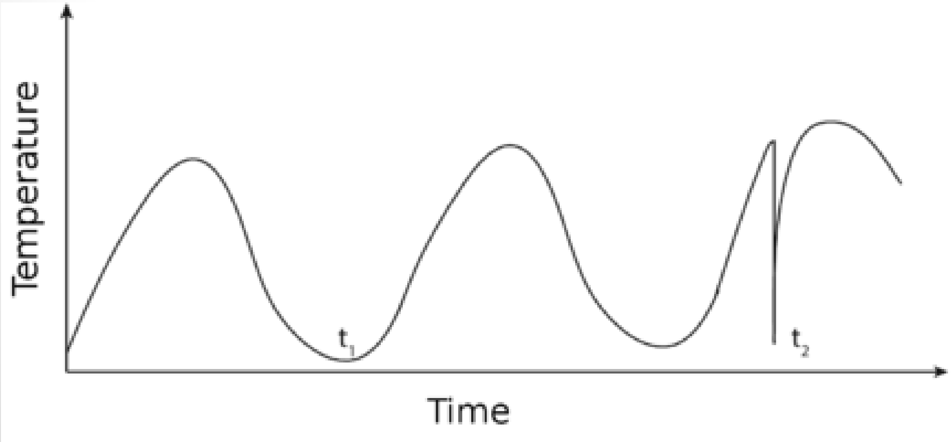
1.2 Noisy Data:

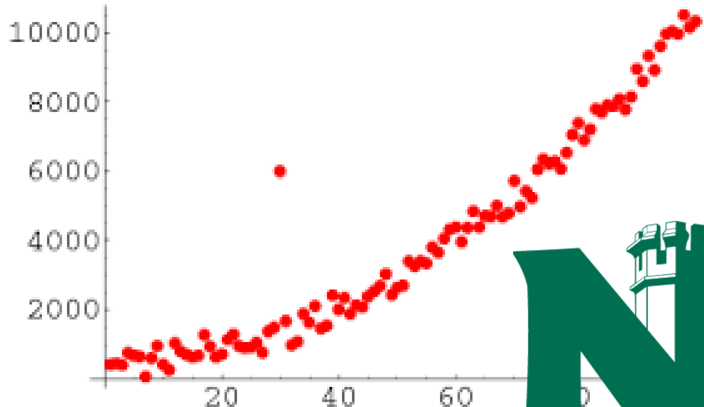
Noise is a random error or variance in a measured attribute. Noisy data comes from:

- Faulty data collection (human or equipment error)
- Data entry mistake
- Data transmission problem

Example:

- Observing a student's height as 6 meters instead of 6 feet
- Entering $500 instead of $5.00

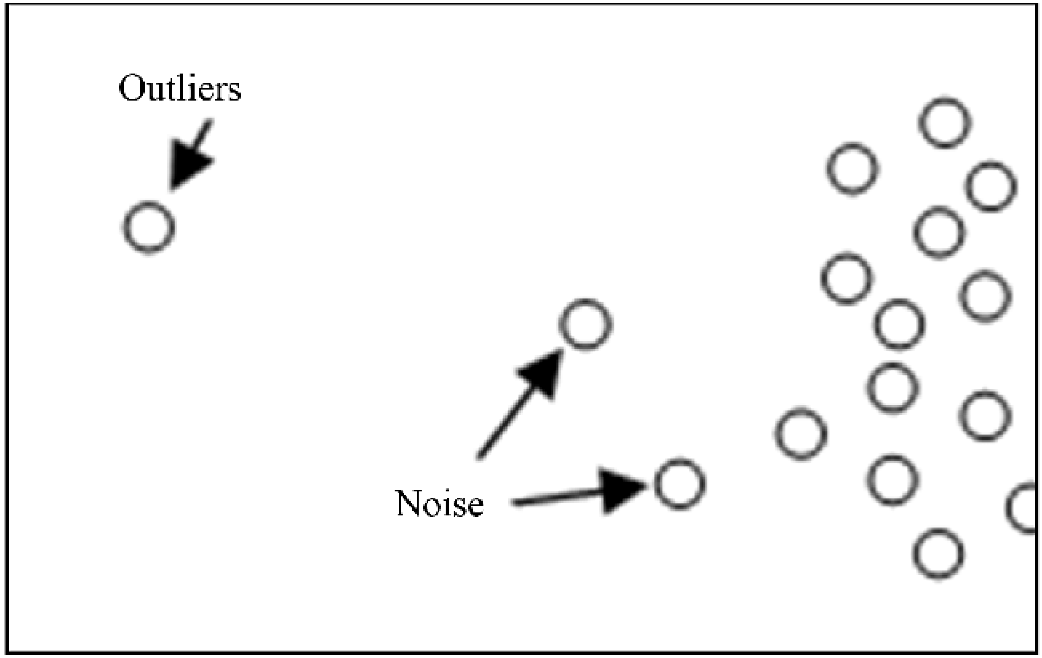




1.2.2 Noise vs. Outlier:

Noise may appear randomly in a dataset, but outliers are the once which are significantly different from the remaining dataset.

- Outliers are supposed to be rare.



1.2.3 Handling Noisy Data:

The easy way is collecting more data to identify the underlying phenomenon that is generating the data.

The common techniques are:

1. Binning Method (Smoothing)
2. Clustering Method
3. Regression Method

1. Binning Method:

Binning methods smooth a sorted data value by consulting the values around it.

- Sort data and partition into (equi-depth) bins
- In smoothing by bin boundaries (minimum and maximum values), each bin value is then replaced by the closest boundary value.
- Similarly, smoothing by bin mean or median, can be employed.

Note: equi-depth means that dividing the range into *N* intervals of equal size (uniform grid), and width of the intervals is $\frac{Maximum - Minimum}{N}$

Sorted data for *price* **(in dollars)**: 4, 8, 15, 21, 21, 24, 25, 28, 34

Partition into (equal-frequency) bins:

Bin 1: 4, 8, 15
Bin 2: 21, 21, 24
Bin 3: 25, 28, 34

Smoothing by bin means:

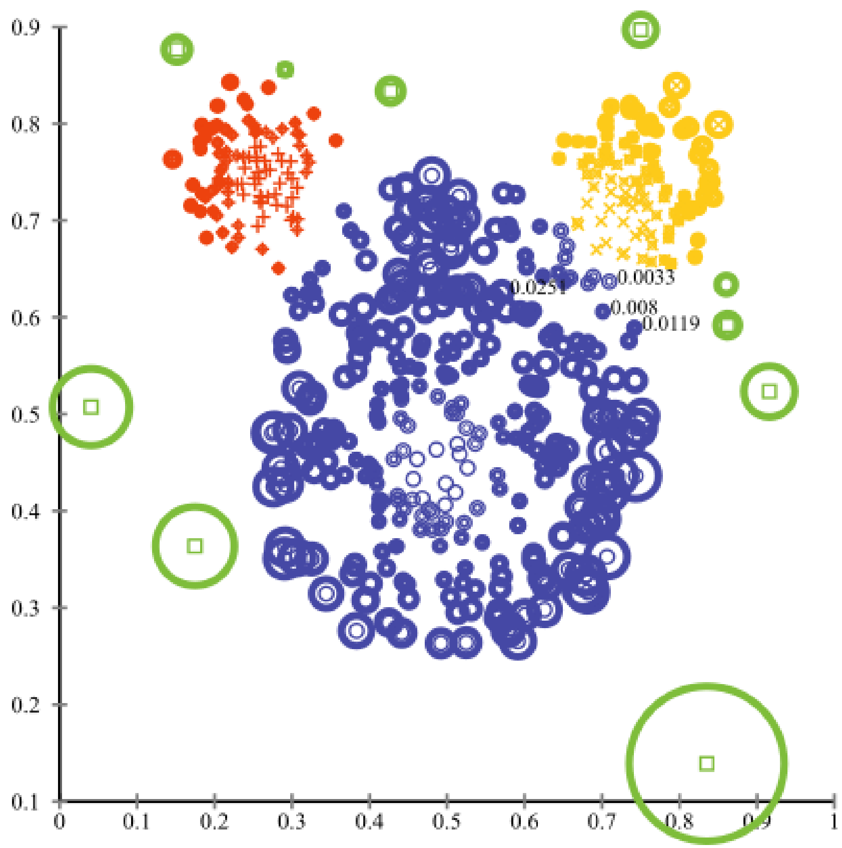
Bin 1: 9, 9, 9
Bin 2: 22, 22, 22
Bin 3: 29, 29, 29

Smoothing by bin boundaries:

Bin 1: 4, 4, 15
Bin 2: 21, 21, 24
Bin 3: 25, 25, 34

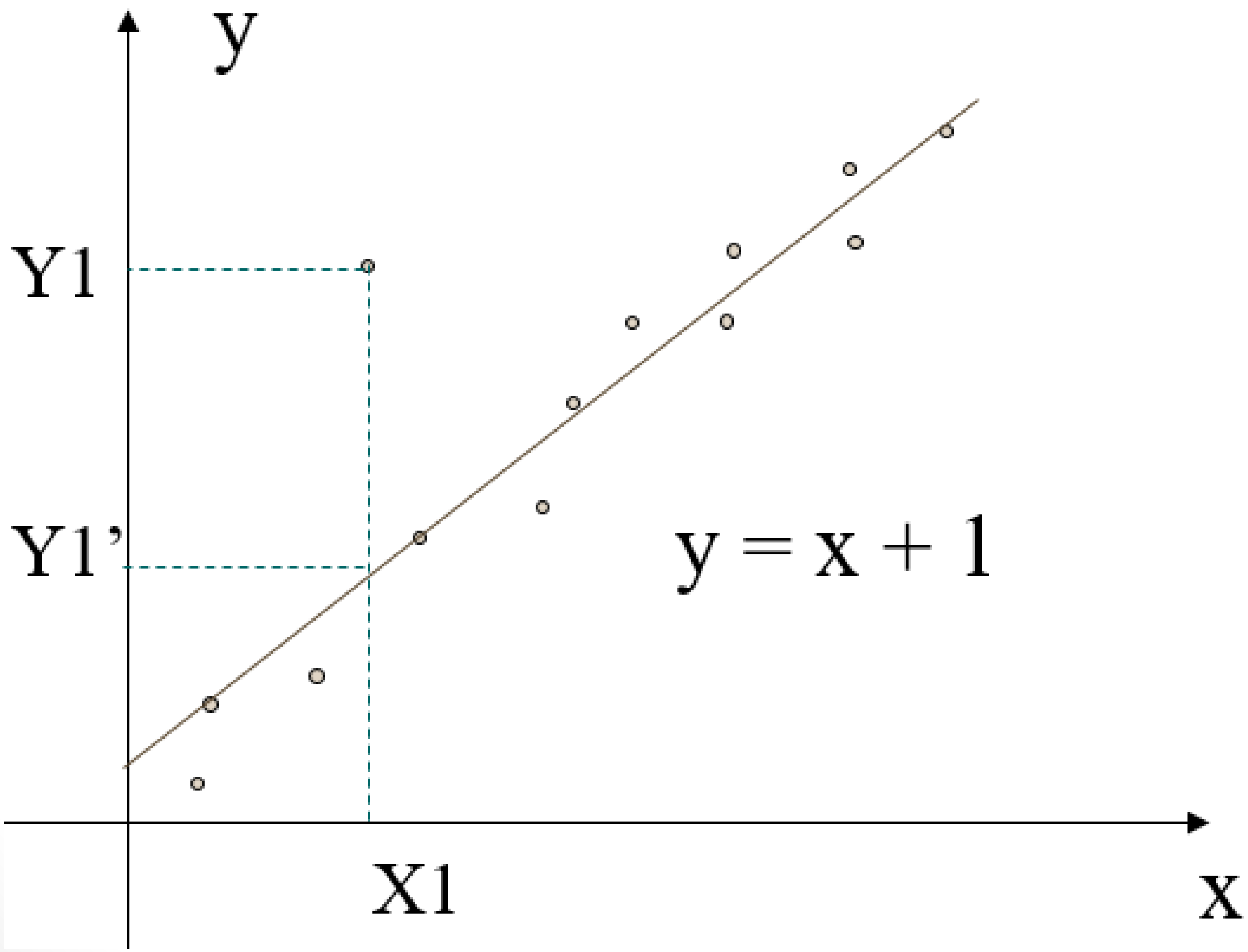
2. Clustering Method:

Outliers may be detected by clustering, where similar values are organized into groups (clusters). Values that fall outside of the set of clusters may be considered outliers.



3. Regression Method:

Data can be smoothed by fitting the data into a linear regression function.



1.3 Inconsistent Data:

In cases where the same information is available from multiple sources (data integration), inconsistencies may be detected.

Example: Similar data are kept in different formats in two file.

- It is important to match the data between files.

1.3.1 Handling Inconsistent Data:

- Manual correction using external references
- Semi-automatic using various tools
  - To detect violation of known functional dependencies and data constraints
  - To correct redundant data

1.4 Duplicate Data:

Data set may include data objects that are duplicates, or almost duplicates of one another.

Example:

Same person with multiple email addresses

- Removing duplicate records is the easiest way to handle the duplicate data.