Data Management-I

What will we learn

- Importance of data management
- First look at the data

- Creating sub sets

using Index

using subset function

First look at the data

Import

- Typically data is imported from excel or other databases such as Oracle, MySql etc. Always check dimensions, variable types and first few rows of data after import.

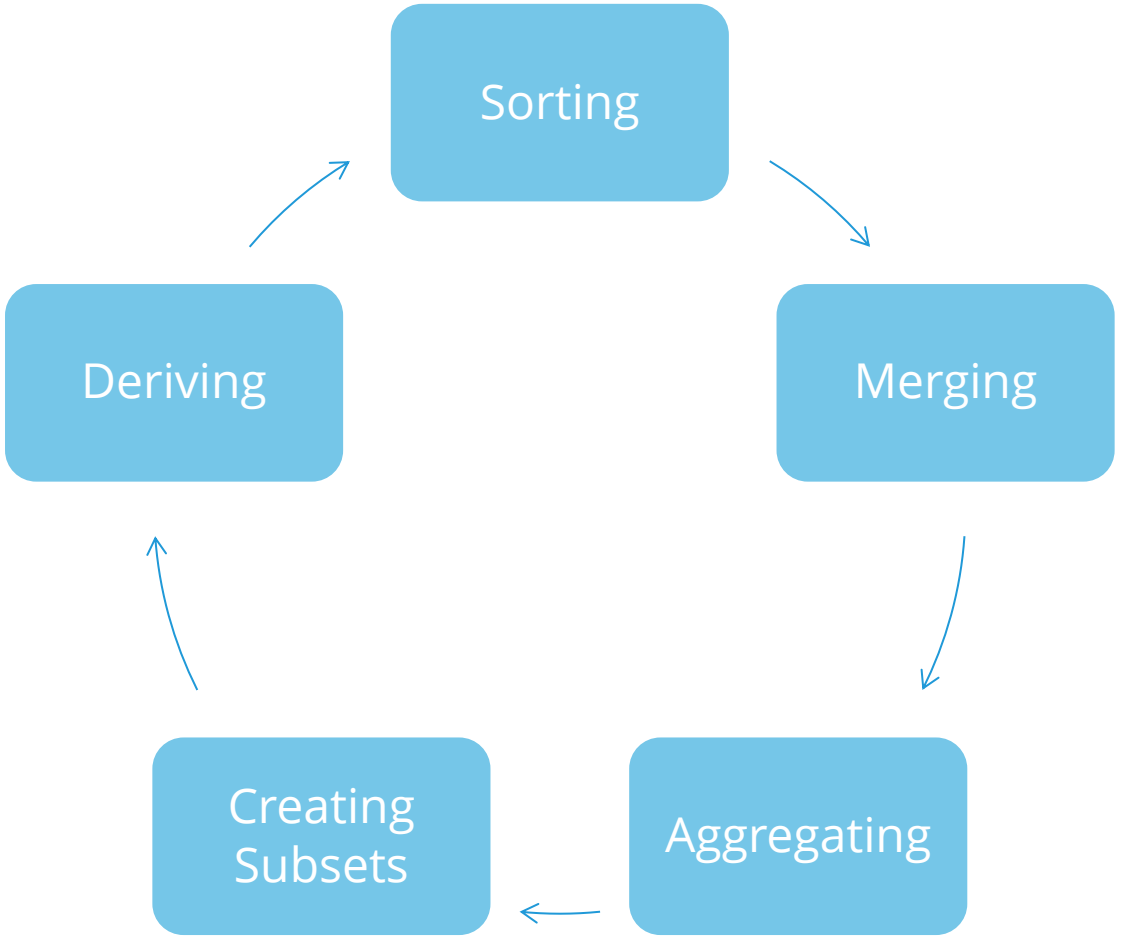
Data Health Check

- Check each variable for missing data, inconsistent data and incorrect data.
- 3 C's of good data: Complete, Correct and Consistent

Data Cleaning

- Wherever possible, clean the data before starting statistical analysis.

Data Management Tasks



Understand Data Using summary() Function

```
#Import basic_salary2 data from day 2 folder
salary<-read.csv(file.choose(),stringsAsFactors =T)
summary(salary)                    #Variables summarized based on
```

```
First_Name   Last_Name   Grade      Location       Function        ba              ms
Kavita : 2   Joshi  : 2   GR1 :23   DELHI :17   FINANCE  :13   Min.   :10940   Min.   : 2700
Mahesh : 2   Shah   : 2   GR2 :17   MUMBAI:21   SALES    :15   1st Qu.:13785   1st Qu.:10450
Nishi  : 2   Singh  : 2   NA's: 1   NA's  : 3   TECHNICAL:11   Median :16230   Median :12420
Priya  : 2   Arora  : 1                        NA's     : 2   Mean   :17210   Mean   :11939
Ajit   : 1   Bhide  : 1                                       3rd Qu.:19305   3rd Qu.:14200
Ameet  : 1   Bhutala: 1                                       Max.   :29080   Max.   :16970
(Other):31   (Other):32                                       NA's   :2       NA's   :4
```

#The summary function displays count of NA's which are interpreted as missing values
(NA stands for Not Available)

DATA SCIENCE INSTITUTE

Check Missing Observations

```
nmiss<-sum(is.na(salary$ba))
nmiss
[1] 2
                                    #Number of missing observations

mean(salary$ba)
[1] NA
                                    #NA output due to missing observations

mean(salary$ba, na.rm=TRUE)
[1] 17209.74
                                    #Removes missing observations and calculates
                                    mean
```

Extracting Complete Cases

- Extract complete cases using any one of following two functions
- It removes rows with at least one NA value

```
d1<-na.omit(salary)
d1
sum(is.na(d1))
[1] 0o
```

- Note that instead of na.omit, na.exclude() is also a valid syntax.

Handling Missing Values (Blank Entries) While Importing

- Sometimes, missing values are kept blank in the data.
- In case of numeric variables, read.csv() replaces blank cells with NA by default.

```
> data<-read.csv(file.choose())
> head(data,n=5)
```

```
  First_Name Last_Name Grade Location  Function    ba    ms
1     Mahesh      Joshi   GR1    DELHI     SALES 17990 16070
2     Rajesh      Kolte          DELHI   FINANCE 19250 14960
3       Neha        Rao   GR1    DELHI   FINANCE 19235 15200
4      Priya       Jain   GR1    DELHI     SALES    NA 13490
5      Sneha      Joshi   GR1    DELHI   FINANCE 20660 15660
```

Handling Missing Values (Blank Entries) While Importing

- In case of categorical variables, NAs (<NA>) are coded by adding the following argument

```
> data<-read.csv(file.choose(),na.strings="")
> head(data,n=5)
```

```
 First_Name Last_Name Grade Location  Function    ba    ms
1     Mahesh     Joshi   GR1    DELHI     SALES 17990 16070
2     Rajesh     Kolte  <NA>    DELHI   FINANCE 19250 14960
3       Neha       Rao   GR1    DELHI   FINANCE 19235 15200
4      Priya      Jain   GR1    DELHI     SALES    NA 13490
5      Sneha     Joshi   GR1    DELHI   FINANCE 20660 15660
```

Subset Using Selected Rows

- Display rows from 5th to 10th and all columns

```
> salary[c(5:10), ]
```

	First Name	Last Name	Grade	Location	Function	ba	ms
5	Sneha	Joshi	GR1	DELHI	FINANCE	20660	15660
6	Mahesh	Rane	GR1	DELHI	TECHNICAL	23160	14200
7	Ram	Kanade	GR1	DELHI	TECHNICAL	20160	15850
8	Nishi	Honrao	GR1	DELHI	TECHNICAL	20460	15880
9	Nishi	Kulkarni	GR1	<NA>	SALES	22620	16150
10	Hameed	Singh	GR1	DELHI	SALES	23720	15120

Subset Using Selected Rows...

- Display only selected rows

```
> salary[c(1,3,5,8), ]
```

	First Name	Last Name	Grade	Location	Function	ba	ms
1	Mahesh	Joshi	GR1	DELHI	SALES	17990	16070
3	Neha	Rao	GR1	DELHI	FINANCE	19235	15200
5	Sneha	Joshi	GR1	DELHI	FINANCE	20660	15660
8	Nishi	Honrao	GR1	DELHI	TECHNICAL	20460	15880

Subset Using Selected Columns

- Object salary1 has columns 1 and 2

```
> salary1<-salary[   , c(1,2)]
> head(salary1)
```

```
  First_Name Last_Name
1     Mahesh     oshi
2     Rajesh     Kolte
3     Neha       Rao
4     Priya      Jain
5     Sneha    Joshi
6     Mahesh   Rane
```

Selected Rows for Selected Columns

- Object salary2 has rows 1,5,8,4 and columns 1 and 2

```
> salary2<-salary[c(1,5,8,4), c(1,2)]
> salary2
```

```
  First_Name Last_Name
1     Mahesh     Joshi
5      Sneha     Joshi
8      Nishi    Honrao
4      Priya      Jain
```

Subset Function

- Condition on observations
- Condition on variable names
- Condition on observations and variable names

Condition on Observations

- All details of employees of DELHI with ba more than 20000

```
> salary3 <-subset(salary, Location=="DELHI" & ba > 20000)
> head(salary3)
```

	First Name	Last Name	Grade	Location	Function	ba	ms
4	Priya	Jain	GR1	DELHI	SALES	23280	13490
5	Sneha	Joshi	GR1	DELHI	FINANCE	20660	15660
6	Mahesh	Rane	GR1	DELHI	TECHNICAL	23160	14200
7	Ram	Kanade	GR1	DELHI	TECHNICAL	20160	15850
8	Nishi	Honrao	GR1	DELHI	TECHNICAL	20460	15880
10	Hameed	Singh	GR1	DELHI	SALES	23720	15120

Condition on Variable Names

- Only First name and Last name of previous data

```
> salary4<-subset(salary3,select=c(First_Name,Last_Name))
> head(salary4)
```

```
  First_Name Last_Name
4      Priya      Jain
5      Sneha     Joshi
6     Mahesh      Rane
7        Ram    Kanade
8      Nishi    Honrao
10    Hameed     Singh
```

Condition on Observations and Variables

- Select details of specific set of employees

```
> salary5<-subset(salary, Grade=="GR1" & ba>15000,
                  select= c(First_Name,Grade, Location))
> head(salary5)
```

```
  First_Name Grade Location
1     Mahesh   GR1    DELHI
2     Rajesh   GR1    DELHI
3      Neha    GR1    DELHI
4      Priya   GR1    DELHI
5      Sneha   GR1    DELHI
6     Mahesh   GR1    DELHI
```

A Quick Recap

- is.na()
- mean() function to treat missing values
- Sub-setting using indexing
- subset() function

Appendix-Operators

Arithmetic Operators

Operator	Description
+	addition
-	subtraction
*	multiplication
/	division
^ or **	exponentiation
x %% y	modulus (x mod y) 5%%2 is 1
x %/% y	integer division 5%/%2 is 2

Logical Operators

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x | y	x OR y
x & y	x AND y
isTRUE(x)	test if X is TRUE

Data Management -II

What will we learn

- Sorting of data set
- Merging data sets
- Aggregating to get sum

Data Sorting in R

- Sorting data is one of the common activities in preparing data for analysis
- Sorting is storage of data in sorted order, it can be in ascending or descending order.
- We will be exploring all the ways in which sorting can be done.

```
#Import and attach basic_salary2 data from day 2 folder
```

```
salary<-read.csv(file.choose())
```

```
attach(salary)
```

attach() attaches the database to the R search path, so the variables in the database can be accessed by simply giving their names

Data Sorting in R (Ascending)

- Sort salary by ba in ascending order
- order() sorts in ascending order by default

```
> ba_sorted<-salary[order(ba), ]
> head(ba_sorted)
```

```
   First_Name Last_Name Grade Location  Function    ba    ms
37      Archa  Narvekar   GR2   MUMBAI TECHNICAL 10940 11160
32       Anup      Save   GR2   MUMBAI     SALES 11960  7880
33     Yogesh    Lonkar   GR2   MUMBAI TECHNICAL 12390  6630
38      Shiva    Jathar   GR2   MUMBAI   FINANCE 12860 10940
41      Ketan   Kharkar   GR2   MUMBAI     SALES 13140  9800
34      Sagar    Chavan   GR2   MUMBAI   FINANCE 13390  6700
```

Data Sorting in R (Descending)

- Sort salary by ba in descending order

```
> ba_sorted_2<-salary[order(-ba), ]
> head(ba_sorted_2)
```

```
   First_Name Last_Name Grade Location  Function    ba    ms
12     Yogita      Raje   GR1    DELHI     SALES 29080  8795
11        Raj    Mohite   GR1    DELHI   FINANCE 26080 16970
10     Hameed     Singh   GR1    DELHI     SALES 23720 15120
4       Priya      Jain   GR1    DELHI     SALES 23280 13490
6      Mahesh      Rane   GR1    DELHI TECHNICAL 23160 14200
9       Nishi  Kulkarni   GR1     <NA>     SALES 22620 16150
```

- The '-' sign sorts numeric columns in descending order. Alternatively you can use **decreasing=TRUE**

Data Sorting in R (Using Factor Variable)

- Sort data by column with characters / factors

#Sort salary by Grade

```
> gr_sorted<-salary[order(Grade), ]
> head(gr_sorted)
```

```
  First_Name Last_Name Grade Location  Function    ba    ms
1     Mahesh     Joshi   GR1    DELHI     SALES 17990 16070
2     Rajesh     Kolte   GR1    DELHI   FINANCE 19250 14960
3       Neha       Rao   GR1    DELHI   FINANCE 19235 15200
4      Priya      Jain   GR1    DELHI     SALES 23280 13490
5      Sneha     Joshi   GR1    DELHI   FINANCE 20660 15660
6     Mahesh      Rane   GR1    DELHI TECHNICAL 23160 14200
```

- Note that by default order() sorts in ascending order

Data Sorting in R (Using Factor Variable)

- Sort data by column with characters / factors

#Sort salary by Grade in descending order

```
> gr_sorted_2<-salary[order(Grade, decreasing=TRUE), ]
> head(gr_sorted_2)
```

```
   First_Name Last_Name Grade Location  Function    ba    ms
25      Priya    Mittal   GR2    DELHI TECHNICAL 15000 10680
26     Naresh     Sinha   GR2    DELHI TECHNICAL 13810 11540
27     Jivesh      Shah   GR2     <NA>   FINANCE 16000 13730
28      Jigar      Shah   GR2    DELHI   FINANCE 16230    NA
29     Gaurav     Singh   GR2    DELHI     SALES 13760 13220
30       Amit     Mehta   GR2    DELHI TECHNICAL 13660  6840
```

- For reversing the sorting order for factor variables, include logical argument decreasing=TRUE

Sorting Data by Multiple Variables

- Sort data by giving multiple columns; one column with characters / factors and one with numerals

#Sort salary_data by Grade and ba

```
> grba_sorted<-salary[order(Grade, ba), ]
> head(grba_sorted)
```

```
   First_Name Last_Name Grade Location  Function    ba    ms
13     Anjali     Sonar   GR1   MUMBAI      <NA> 14410 10450
15      Rahul    Potdar   GR1   MUMBAI     SALES 15125    NA
14      Bipin     Bhide   GR1   MUMBAI   FINANCE 15230 11010
17    Mangesh       Oak   GR1   MUMBAI     SALES 15800 12420
18      Anand     Soman   GR1     <NA>   FINANCE 16540 12780
19     Malhar    Jadhav   GR1   MUMBAI TECHNICAL 17240 13220
```

- Here, data is first sorted in increasing order of Grade then by increasing order of ba within Grade

Merging by Variables

- #Import following 2 data sets

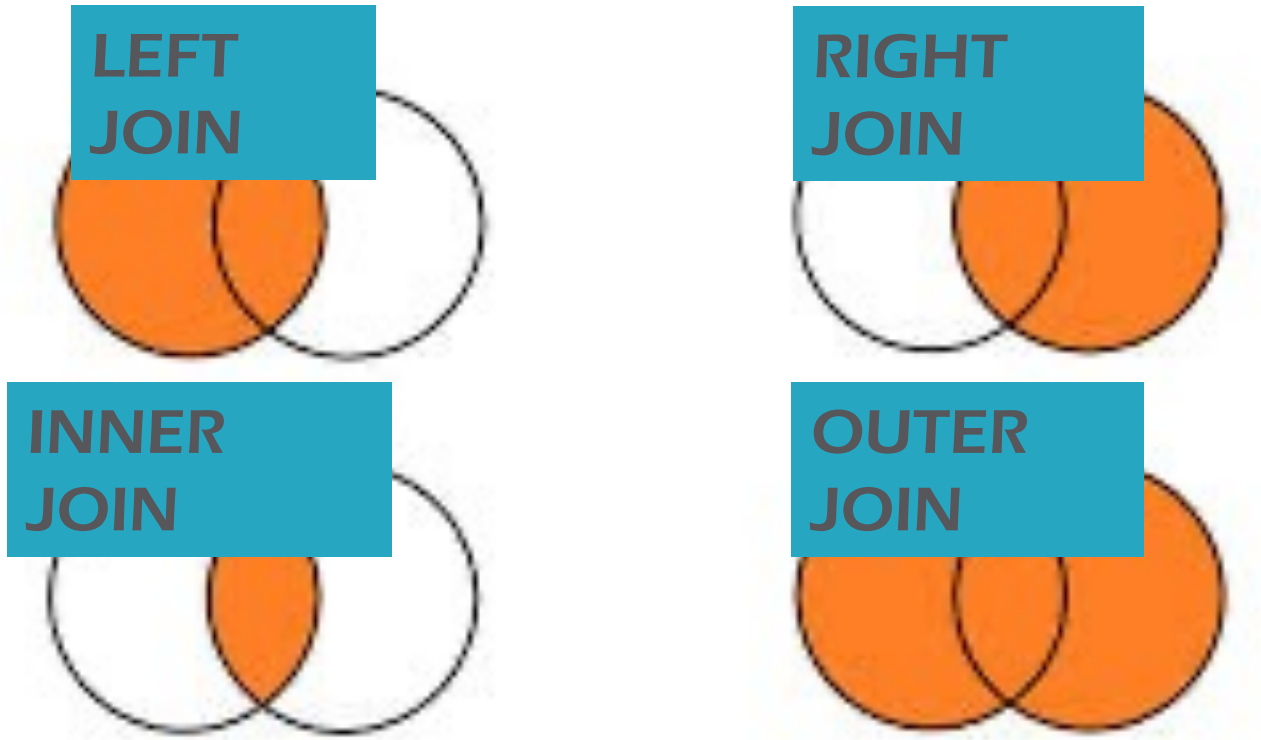
sal_data

	Employee_ID	First_Name	Last_Name	Basic_Salary
1	E-1001	Mahesh	Joshi	16070
2	E-1002	Rajesh	Kolte	14960
3	E-1004	Priya	Jain	13490
4	E-1005	Sneha	Joshi	15660
5	E-1007	Ram	Kanade	15850
6	E-1008	Nishi	Honrao	15880

bonus_data

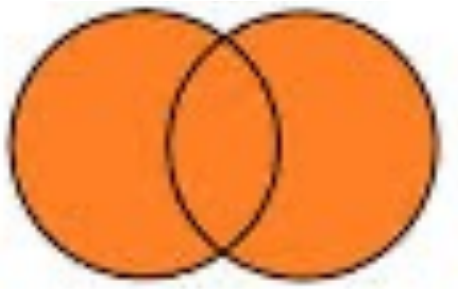
	Employee_ID	Bonus
1	E-1001	12050
2	E-1003	11400
3	E-1004	10110
4	E-1006	10650
5	E-1008	11910
6	E-1010	11340

Types of Joins



DATA SCIENCE INSTITUTE

Outer Joins



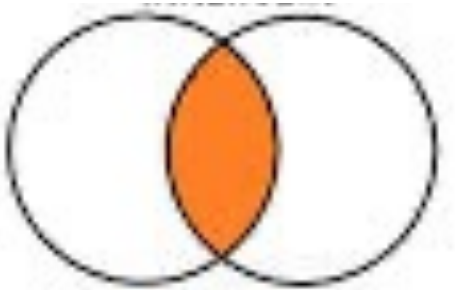
- Outer Join includes all employee ID's from both data sets

```
> outerjoin<- merge(sal_data,bonus_data,
                    by=c("Employee_ID"), all=TRUE)

> outerjoin
```

	Employee_ID	First_Name	Last_Name	Basic_Salary	Bonus
1	E-1001	Mahesh	Joshi	16070	12050
2	E-1002	Rajesh	Kolte	14960	NA
3	E-1004	Priya	Jain	13490	10110
4	E-1005	Sneha	Joshi	15660	NA
5	E-1007	Ram	Kanade	15850	NA
6	E-1008	Nishi	Honrao	15880	11910
7	E-1009	Hameed	Singh	15120	NA
8	E-1003	<NA>	<NA>	NA	11400
9	E-1006	<NA>	<NA>	NA	10650
10	E-1010	<NA>	<NA>	NA	11340

Inner Join



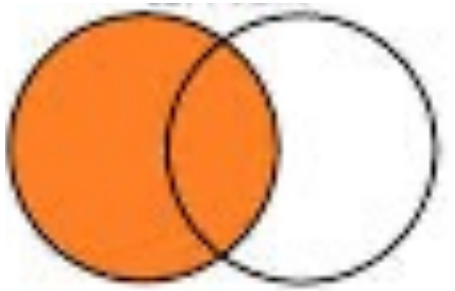
- Inner Join includes employee ID only if present in both data sets

```
> innerjoin<- merge(sal_data,bonus_data,
                    by=c("Employee_ID"))

> innerjoin
```

	Employee_ID	First_Name	Last_Name	Basic_Salary	Bonus
1	E-1001	Mahesh	Joshi	16070	12050
2	E-1004	Priya	Jain	13490	10110
3	E-1008	Nishi	Honrao	15880	11910

Left Join

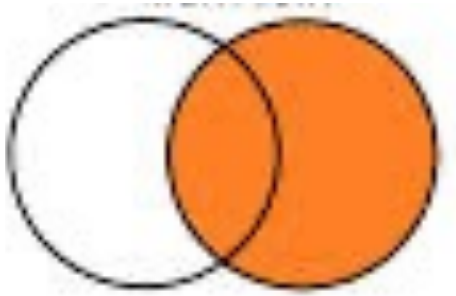


- Left Join includes all employee ID's from first data set

```
> leftjoin<-merge(sal_data,bonus_data,
                  by=c("Employee_ID"), all.x=TRUE)
> leftjoin
```

	Employee_ID	First_Name	Last_Name	Basic_Salary	Bonus
1	E-1001	Mahesh	Joshi	16070	12050
2	E-1002	Rajesh	Kolte	14960	NA
3	E-1004	Priya	Jain	13490	10110
4	E-1005	Sneha	Joshi	15660	NA
5	E-1007	Ram	Kanade	15850	NA
6	E-1008	Nishi	Honrao	15880	11910
7	E-1009	Hameed	Singh	15120	NA

Right Join



- Right Join includes all employee ID's from second data set

```
> rightjoin<-merge(sal_data,bonus_data,
                   by=c("Employee_ID"), all.y=TRUE)

> rightjoin
```

	Employee_ID	First_Name	Last_Name	Basic_Salary	Bonus
1	E-1001	Mahesh	Joshi	16070	12050
2	E-1004	Priya	Jain	13490	10110
3	E-1008	Nishi	Honrao	15880	11910
4	E-1003	<NA>	<NA>	NA	11400
5	E-1006	<NA>	<NA>	NA	10650
6	E-1010	<NA>	<NA>	NA	11340

Merging Cases (Append)

- Appending two datasets using *rbind* function requires both the datasets with exactly the same number of variables with exactly the same names.

- If datasets do not have the same number of variables, variables can be either dropped or created so both match.

Merging Cases (Append)...

```
#Import new_emp dataset

new_emp<-read.csv(file.choose(),header=T)

#Append datasets

sal_data<-rbind(sal_data,new_emp)

sal_data
```

Aggregate Function≈

```
#To calculate mean for variable 'ba' by Location variable
```

```
A<-aggregate(ba ~ Location, data = salary, FUN = mean )
```

```
A
```

```
  Location       ba
1    DELHI 19430.29
2   MUMBAI 15037.11
```

#Aggregate function by default ignores the missing data values.

Therefore, na.rm=TRUE is not required in mean function.