Basic Data Handling

Agenda

If Else

For Loop

Switch

Functions

Missing Values

subset

Apply family

Outlier

Sort, order, rank

If else

```
big <- 20
small <- 5
ifelse(big > 10, "Yes", "No")
ifelse(small > 10,"Yes","No")
```

```
# You can also create more
complex if statements
if(big ==15){
  tiny <- 2
} else {
  tiny <- 3
}
tiny
```

For Loop

```
x <- NULL
for(i in 1:5){
  x[i] <- i * 2
}
x
#[1]  2  4  6  8 10
```

```
for ( i in 1:10)
{  print(i)
}
#print 1 to 10 in each row
```

```
#substitute 2 and 3 position with
zero values
for(i in 2:3){
  x[i] <- 0
}
x
#[1]  2  0  0  8 10
```

```
x = c(1,2,3)
for (v in c(4:6))
{
  print(c(x, v))
}
```

```
[1] 1 2 3 4
[1] 1 2 3 5
[1] 1 2 3 6
```

switch

`switch (expression, list)`

tests an expression against elements of a list. If the value evaluated from the expression matches item from the list, the corresponding value is returned.

```
switch(2,"red","green","blue")
#[1] "green"
switch(1,"red","green","blue")
#[1] "red"
```

```
use.switch <- function(x)
{
  switch(x,'a' = 'First','b' = 'Second','c' =  'Third','other')
}

use.switch('a')
use.switch('b')
use.switch('other')
use.switch('6')
use.switch(6)  # nothing returned
```

Functions

```
# This function will take two inputs, x and y
my.function <- function(x,y){
  5*x+y
}
my.function(2,1)
```

Missing Values

```
library(VIM)
df4 = sleep
head(df4)
dim(df4)

#row-wise delete missing values in your dataset
na.omit(df4)
na.exclude(df4)

#will keep an object only if no missing values
are present
na.fail(df4)
```

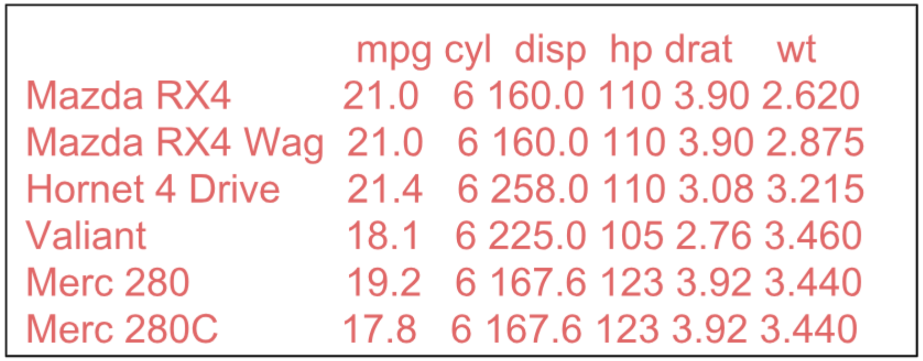
Adding & Keeping Variables

```
# Basic Data Manipulation in mtcars
df3 = mtcars
head(df3)
#adding variables
df3$mpgplus = df3$mpg + 2
head(df3)


#remove the column
df3$mpgplus = NULL
head(df3)
```

```
#selected variables
varnames = c('mpg', 'wt', 'cyl')
df3[varnames]
#columns to be from any of these
(selected = names(df3) %in% varnames)
df3[selected]

#other than selected columns
df3[-selected]
```

Subset

```
#subset( ) function is the easiest way to select variables and observations.

mtcars
# using subset function
(newdata = subset(mtcars, mpg >=20 & mpg < 30))
(newdata = subset(mtcars, mpg >=20 & mpg < 30, select=c(mpg, disp)))
(newdata <- subset(mtcars, cyl == 6 & disp > 150,select=mpg:wt))
```



```
               mpg cyl  disp  hp drat    wt
Mazda RX4      21.0   6 160.0 110 3.90 2.620
Mazda RX4 Wag  21.0   6 160.0 110 3.90 2.875
Hornet 4 Drive 21.4   6 258.0 110 3.08 3.215
Valiant        18.1   6 225.0 105 2.76 3.460
Merc 280       19.2   6 167.6 123 3.92 3.440
Merc 280C      17.8   6 167.6 123 3.92 3.440
```

Apply : alternative to for loop

```
(m1 = matrix(1:20, nrow=4)) # matrix
#mean of each row: manually
mean(m1[1,]); mean(m1[2,]); mean(m1[3,]);
mean(m1[4,])
#mean of each row: for loop
for (i in 1:nrow(m1)) {
  print(mean(m1[i,]))
}

#apply command
apply(m1,1,mean)
#apply for columns
apply(m1, MARGIN = 2, mean)
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
```

```
[1] 9
[1] 10
[1] 11
[1] 12
```

```
[1]  9 10 11 12
```

```
[1]  2.5  6.5 10.5 14.5 18.5
```

Number Formatting

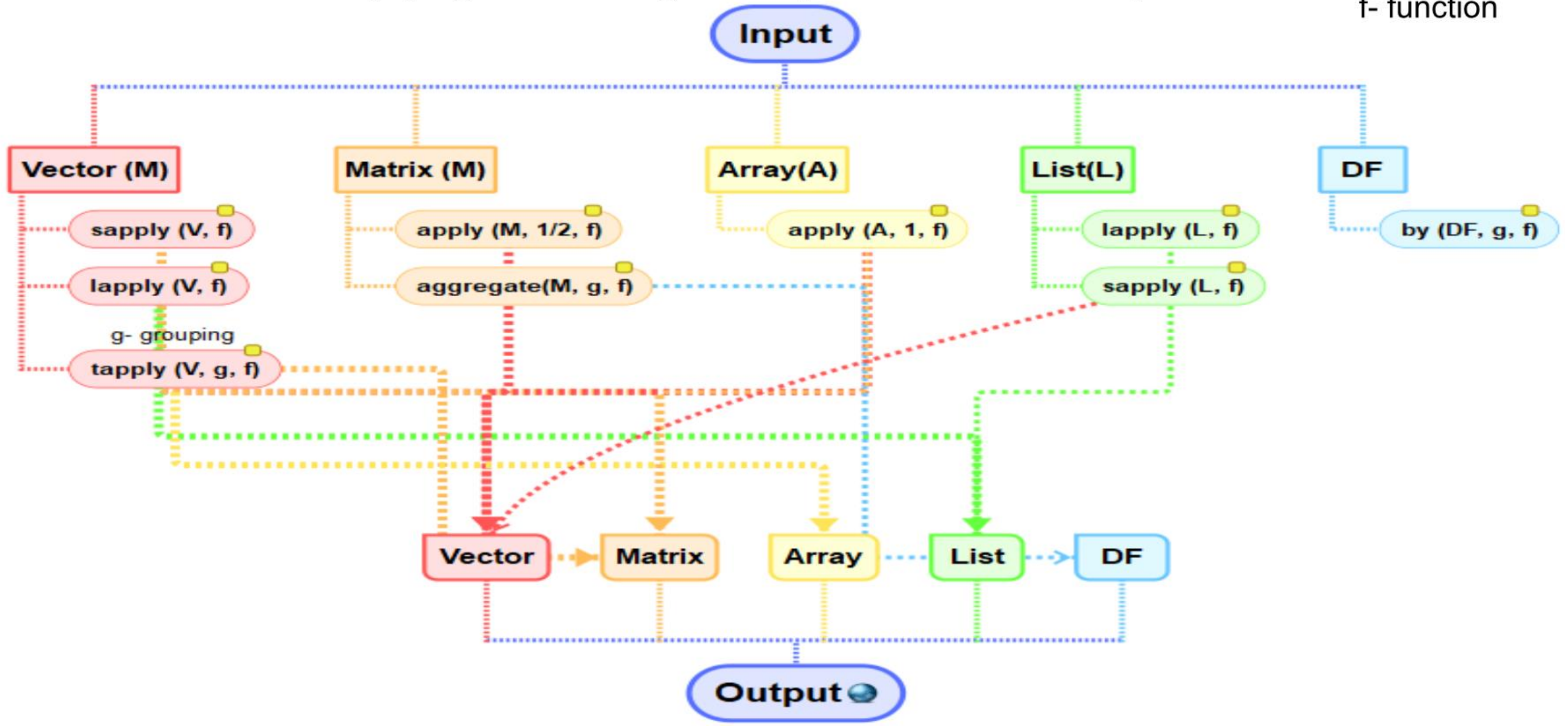
```
x = c(23.3, 34.742)

floor(x); ceiling(x); trunc(x); round(x,1)

options(digits=2) # will change display for future to 2 decimal places
```

Apply : family of commands

Base::apply	Apply Functions Over Array Margins
base::by	Apply a Function to a Data Frame Split by Factors
Base::eapply	Apply a Function Over Values in an Environment
base::lapply	Apply a Function over a List or Vector
base::mapply	Apply a Function to Multiple List or Vector Arguments
base::rapply	Recursively Apply a Function to a List
base::tapply	Apply a Function Over a Ragged Array

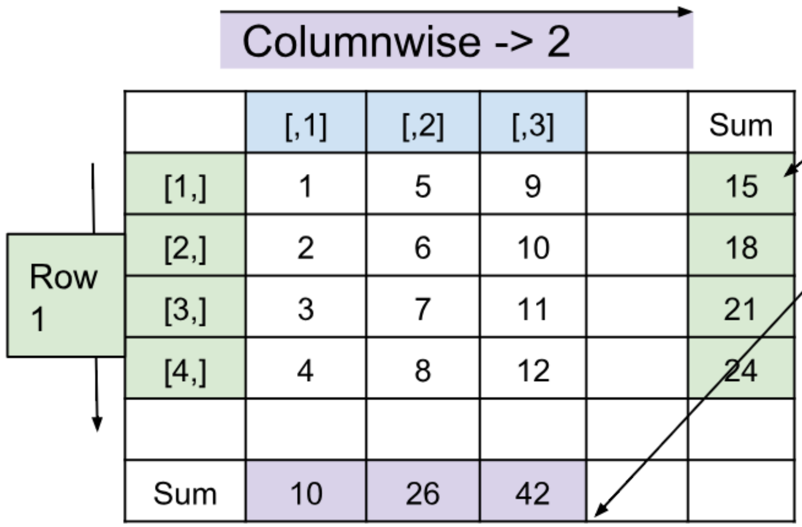
Apply Family - Input and Output

g-group
f- function



apply

```
apply(X, MARGIN, FUN, ...)
```



	[,1]	[,2]	[,3]		Sum
[1,]	1	5	9		15
[2,]	2	6	10		18
[3,]	3	7	11		21
[4,]	4	8	12		24
Sum	10	26	42		

```
m1 =
matrix(1:12,ncol=3)
apply(m1,1,sum)
apply(m1,2,sum)
```

```
apply(m1,c(1,2), function(x) x^2)
```

```
     [,1] [,2] [,3]
[1,]    1   25   81
[2,]    4   36  100
[3,]    9   49  121
[4,]   16   64  144
```

```
sapply(1:3, function(x) x^2)
unlist(sapply(1:3, function(x) x^2))
```

```
[1] 1 4 9
```

list/ vector -> Vector

Sort , Order & Rank

```
set.seed(123)
#Vector
(marks = ceiling(runif(11,5,10)))

sort(marks)
sort(marks, decreasing = TRUE)
rev(sort(marks))

order(marks) #index values
marks[order(marks)] #this is marks
marks[order(-marks)] #this is marks

#rank
rank(marks)
```

```
#DF
(df1=mtcars)
df[order(mtcars$mpg),c(1:5)]
df[order(mtcars$mpg,
decreasing=T),c(1:5)]
df[order(mtcars$cyl, -mtcars$mpg),
c('cyl','mpg','wt','hp')]
```

Thanks