R Functions: Table of Contents

General:

dim, length.....1
help.....2
paste.....3
read.csv.....4
round.....5
sort.....6
table.....7

Vector Functions:

c.....8
cbind.....9
rbind.....10
seq.....11

Subsetting:

all.....12
any.....13
unique.....14
which.....15

Plotting:

barplot.....16
boxplot.....17
hist.....18
mosaicplot.....19
plot.....20

Statistics:

cumprod.....21
cumsum.....22
mean.....23
median.....24
quantile.....25
sample.....26
sd.....27
sum.....28
var.....29
weighted.mean.....30

dim(), length()

Get dimensions (# of rows/ cols) of data.frame

Sample Code input:

```
data <- read.csv('sample.csv', header =
TRUE)
dim(data)
```

Code Output: (given code above what's executed when it's run?)

```
> dim(data)
[1] 40  2
```

Examples of Use

- `dim(dataframe)`

Function Inputs

- `dataframe` – a variable of type dataframe

Function Output

- Returns number of rows and columns. First value represents rows and second represents columns

Additional Notes

- Use `length()` for variable of type vector (one dimensional)

help(), ?()

Obtain documentation for a given R command

Examples of Use

- `help(function)`

Function Inputs

- `function` – function name

Function Output

- Description of how to use function

Additional Notes

- Help content appears on the right side of RStudio

```
Sample Code input:
help(seq)
```

```
Code Output: (given code above what's executed when it's run?)
>
Description
Generate regular sequences. seq is a standard generic with a default method. seq.int
is a primitive which can be much faster but has a few restrictions. seq_along and
seq_len are very fast primitives for two common cases.

Usage
seq(...)

## Default S3 method:
seq(from = 1, to = 1, by = ((to - from)/(length.out - 1)),
    length.out = NULL, along.with = NULL, ...)
>
```

paste

Concatenate character vectors

Function Inputs

- Vectors to be concatenated
- `sep` - Separator between vectors in the result

Function Output

- Vector of concatenated values

Additional Notes

- Default separator is blank

```
Sample Code input:

paste("I", "Love", "R")
paste("Mean:", mean(1:10))
paste("One", "Two", "Three", "Four", sep="-")
```

```
Code Output: (given code above what's executed when it's run?)

> paste("I", "Love", "R")
[1] "I Love R"

> paste("Mean:", mean(1:10))
[1] "Mean: 5.5"

> paste("One", "Two", "Three", "Four", sep="-")
[1] "One-Two-Three-Four"
```

read.csv(), read.table()

Load data file into a data.frame

Examples of Use

- `read.csv(filename, header=TRUE)`

Function Inputs

- `filename` – csv file
- `header` = boolean value if column/header name required

Function Output

- Data frame containing csv file data

Additional Notes

- `read.table()` is used to read txt files. See `help()`

Sample Code input:

```
read.csv('sample.csv', header = TRUE)
```

Code Output: (given code above what's executed when it's run?)

```
  Gender Height
1      M   62.5
2      M   64.6
3      M   69.1
4      M   73.9
5      M   67.1
6      M   64.4
7      M   71.1
8      M   71.0
```

round

Rounds a number

Function Inputs

- `x` – vector
- `digits` – num digits round to

Function Output

- Rounded result

```
Sample Code input:

x = c(1/3, 2/3, 30/75)
x
round(x)     #nearest integer
round(x, 3)
```

```
Code Output: (given code above what's executed when it's run?)

> x
[1] 0.3333333 0.6666667 0.4000000

> round(x)
[1] 0 1 0

> round(x, 3)
[1] 0.333 0.667 0.400
```

sort()

Sort the data in descending or ascending order

Examples of Use

- `sort(variable)`
- `sort(variable, decreasing = TRUE)`

Function Inputs

- Numeric, character variable of type vector

Function Output

- Returns the vector in order (default is ascending)

Additional Notes

- See `help(sort)` to use sort in matrices and data frames

```
Sample Code input:

num <- c(5,6,3,7,2,2,8)
sort(num)

color <-
c('blue','white','blue','green','red','red
','blue','white')
sort(color)
```

```
Code Output: (given code above what's executed when it's run?)

> sort(num)
[1] 2 2 3 5 6 7 8
> sort(color)
[1] "blue"  "blue"  "blue"  "green" "red"   "red"   "white" "white"
```

table()

List all unique values of a variable with frequencies

Examples of Use

- `table(variable)`

Function Inputs

- `variable` – a vector of numbers, characters ect

Function Output

- Counts of each unique value in variable

Additional Notes

- Try using variable with numeric values

```
Sample Code input:

color <-
c('blue','white','blue','green','red','red
','blue','white')
table(color)
```

```
Code Output: (given code above what's executed when it's run?)

> table(color)
color
 blue green   red white
    3     1     2     2
```

c

Combine vectors or lists

Function Inputs

- Vectors/lists to be combined

Function Output

- Vector/list of inputs combined

Additional Notes

- Can combine > 2 vectors/lists

```
Sample Code input:

v1 = 1:5; v2 = 6:10; v3 = 11:15
v4 = c(v1, v2, v3)
v4
list1 = list(v1)
list2 = list(v2)
list3 = c(list1, list2)
list3
```

```
Code Output: (given code above what's executed when it's run?)

> v4
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15

> list3
[[1]]
[1] 1 2 3 4 5

[[2]]
[1]  6  7  8  9 10
```

cbind

Combines vectors/matrices/ data frames by *columns*

Function Inputs

- Vectors/matrices/data frames to be combined

Function Output

- Combined result

```
Sample Code input:

v1 = 1:5; v2 = 6:10; v3 = 11:15; v4 = 16:20
m1 = rbind(v1,v2); m2 = rbind(v2,v1)
d1 = data.frame(v1,v2); d2 = data.frame(v1,v2)
m3 = rbind(m1, m2)
d3 = rbind(d1, d2)
```

```
Code Output: (given code above what's executed when it's run?)

> m3
     v1 v2 v2 v1
[1,]  1  6  6  1
[2,]  2  7  7  2
[3,]  3  8  8  3
[4,]  4  9  9  4
[5,]  5 10 10  5

> d3
   v1 v2 v3 v4
1   1  6 11 16
2   2  7 12 17
3   3  8 13 18
4   4  9 14 19
5   5 10 15 20
```

rbind

Combines vectors/matrices/ data frames by *rows*

Function Inputs

- Vectors/matrices/data frames to be combined

Function Output

- Combined result

```
Sample Code input:

v1 = 1:3; v2 = 4:6; v3 = 7:10; v4 = 11:14
m1 = cbind(v1,v2); m2 = cbind(v2,v1)
d1 = data.frame(v1,v2); d2 = data.frame(v3,v4)
m3 = cbind(m1, m2)
d3 = cbind(d1, d2)
```

```
Code Output: (given code above what's executed when it's run?)

> m3
   [,1] [,2] [,3]
v1    1    2    3
v2    4    5    6
v2    4    5    6
v1    1    2    3

> d3
  v1 v2
1  1  4
2  2  5
3  3  6
4  1  4
5  2  5
6  3  6
```

seq

Generate sequence of numbers

Function Inputs

- `from` – start value of sequence
- `end` – end value of sequence
- `by` – increment value
- `length` – length of desired sequence

Function Output

- Generated sequence

Additional Notes

- See ?seq for other forms of this function

```
Sample Code input:

seq(1, 10, length = 10)
seq(1, 100, length = 10)
seq(1, 100, by = 10)  # ends before 100
seq(5,1)
seq(5)
```

```
Code Output: (given code above what's executed when it's run?)

> seq(1, 10, length = 10)
 [1]  1  2  3  4  5  6  7  8  9 10

> seq(1, 100, length = 10)
 [1]  1  12  23  34  45  56  67  78  89 100

> seq(1, 100, by = 10)
 [1]  1 11 21 31 41 51 61 71 81 91

> seq(5,1)
[1] 5 4 3 2 1

> seq(5)
[1] 1 2 3 4 5
```

all

Are all values TRUE?

Function Inputs

- Logical vector

Function Output

- `TRUE` if all values are `TRUE` in the input

```
Sample Code input:

x = 1:10
all(x > 5)
all(x < 0)
all(x > 0)
```

```
Code Output: (given code above what's executed when it's run?)

> all(x > 5)
[1] FALSE

> all(x < 0)
[1] FALSE

> all(x > 0)
[1] TRUE
```

any

Is there a TRUE value?

Function Inputs

- Logical vector

Function Output

- `TRUE` if `TRUE` value exists in the input

```
Sample Code input:

x = 1:10
any(x > 5)
any(x < 0)
```

```
Code Output: (given code above what's executed when it's run?)

> > any(x > 5)
[1] TRUE

> any(x < 0)
[1] FALSE
```

unique

Remove duplicate elements

Function Inputs

- `x` - Vector/data frame/array

Function Output

- Input with duplicate elements removed

Additional Notes

- For a data frame, removes duplicate rows

```
Sample Code input:

unique(c(1,4,7,4,1,9))
d1 = data.frame(c(1,3,3,4), c(2,5,5,5))
colnames(d1) = c("f1", "f2")
d1
unique(d1)
```

```
Code Output: (given code above what's executed when it's run?)

> unique(c(1,4,7,4,1,9))
[1] 1 4 7 9

> d1
  f1 f2
1  1  2
2  3  5
3  3  5
4  4  5

> unique(d1)
  f1 f2
1  1  2
2  3  5
4  4  5
```

which

Return TRUE *indices* of logical vector

Function Inputs

- Logical vector

Function Output

- Indices which are TRUE

```
Sample Code input:

v1 = c(7, 20, 15, 19, 30, 9)
which(v1 > 10)
which(v1 %% 2 == 0) # which are even
v1[which(v1 > 10)]
```

```
Code Output: (given code above what's executed when it's run?)

> which(v1 > 10)
[1] 2 3 4 5

> which(v1 %% 2 == 0)
[1] 2 5

> v1[which(v1 > 10)]
[1] 20 15 19 30
```

barplot()

Creates a bar plot of different styles

Examples of Use

- `barplot(count)`

Function Inputs

- `count` – a table contains counts of values

Function Output

- Bar plot

Additional Notes

- See `help(barplot)` for single variable, multiple variable and different styles of barplot

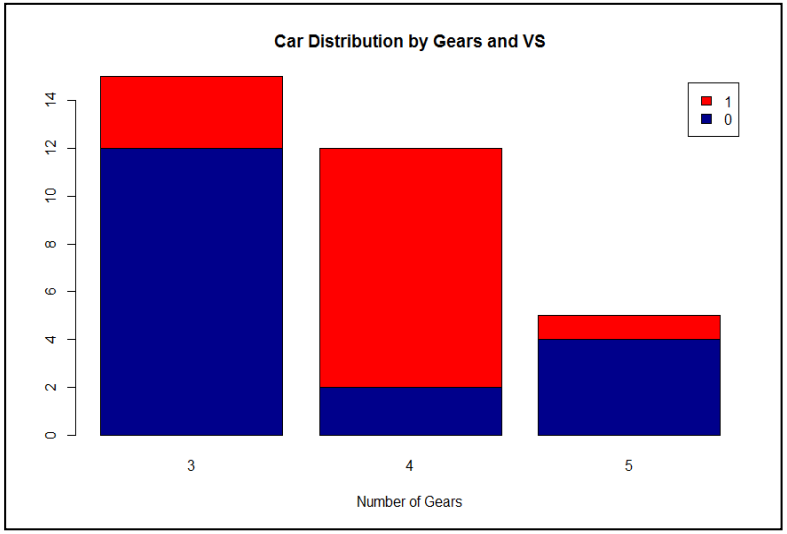
```
Sample Code input:

# Stacked Bar Plot with Colors and Legend
data(mtcars)
counts <- table(mtcars$vs, mtcars$gear)
barplot(counts, main="Car Distribution by
Gears and VS", xlab="Number of Gears",
col=c("darkblue","red"),
legend=rownames(counts))
```

```
Code Output: (given code above what's
executed when it's run?)

> counts

     3  4  5
  0 12  2  4
  1  3 10  1
> barplot(counts, main="Car Distribution
by Gears and VS", xlab="Number of Gears",
col=c("darkblue","red"),
legend=rownames(counts))
```



boxplot()

Produce box-and-whisker plot

```
Sample Code input:

data(iris)
boxplot(Sepal.Length ~ Species, data=iris,
col="gold", ylab="Sepal length",main="Iris
Sepal Length by Species")
```

```
Code Output: (given code above what's
executed when it's run?)

> boxplot(Sepal.Length ~ Species,
data=iris, col="gold", ylab="Sepal
length",main="Iris Sepal Length by
Species"
```

Examples of Use

- `boxplot(object, data)`

Function Inputs

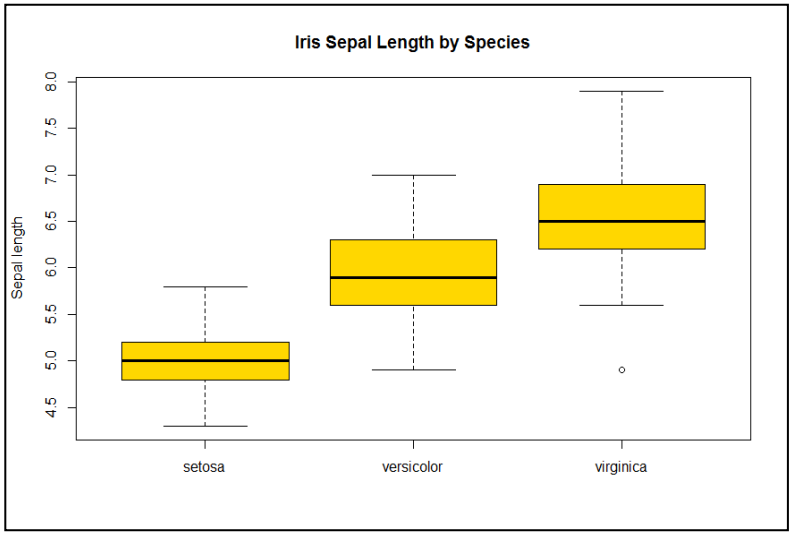
- `object` – a vector or list of numeric values. Relation between two vectors is created using " ~ " symbol
- `data` – dataset name

Function Output

- Box plot. Sepal.Length ~ Species means to calculate range of sepal.length in each category of Species. A small circle in the boxplot represents outliers/noise in the data

Additional Notes

- See `help(boxplot)` for more input options and styles



hist()

Computes and plots the histogram (based on frequency count)

```
Sample Code input:

data(faithful)
hist(faithful$eruptions, breaks=25,
main="Histogram of Old Faithful Geyser
Data", xlab="Duration (mins)",
col='orange')
```

```
Code Output: (given code above what's
executed when it's run?)

> hist(faithful$eruptions, breaks=25,
main="Histogram of Old Faithful Geyser
Data", xlab="Duration (mins)",
col='orange')
>
```

Examples of Use

- `hist(object, breaks, main, xlab, col)`

Function Inputs

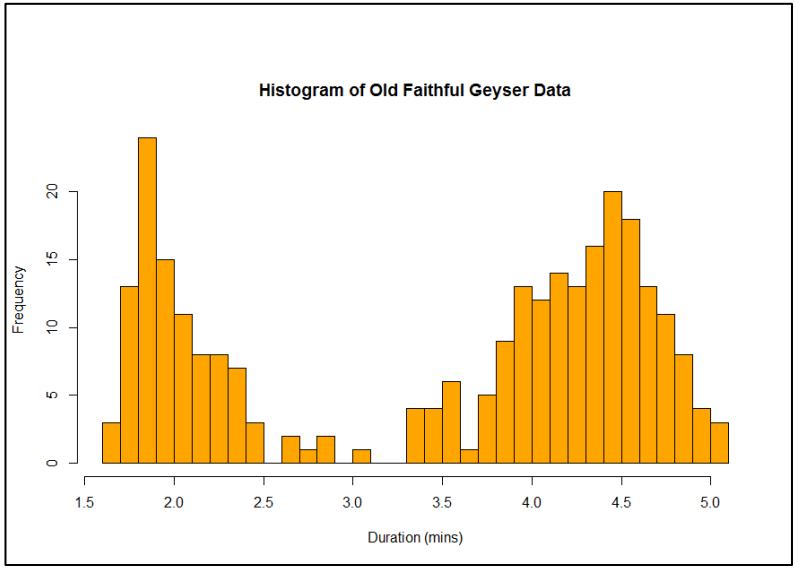
- `object` – a vector of numeric values
- `breaks` – number of cells required in the histogram
- `main` – title of the histogram
- `xlab` – label on the x-axis
- `col` – color of the histogram

Function Output

- Histogram plot

Additional Notes

- See `help(hist)` for more input choices
- Try `col=rainbow(16)`



mosaicplot()

creates a mosaic plot

Sample Code input:

```
data(HairEyeColor)
mosaicplot(HairEyeColor, shade = TRUE,
color =TRUE)
```

Code Output: (given code above what's executed when it's run?)

```
> data(HairEyeColor)
> mosaicplot(HairEyeColor, shade = TRUE,
color =TRUE)
>
```

Examples of Use

- `mosaicplot(data, shade, color)`

Function Inputs

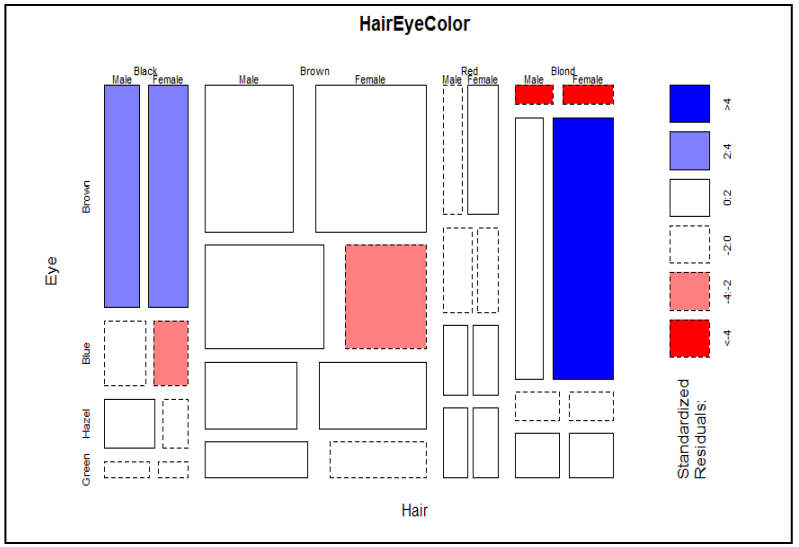
- `data` – table, list or data frame
- `shade` – boolean value to include shade on right side
- `color` – boolean or integer value to set plot color

Function Output

- Mosaic plot

Additional Notes

- See `help(mosaicplot)` for more input options



plot()

Produces a scatter plot

Sample Code input:

```
data(faithful)
plot(waiting~eruptions, data=faithful,
cex=1, pch=19, col="blue", main="Old
Faithful Geyser Eruptions", ylab="Wait
time between eruptions", xlab="Duration of
eruption")
```

Code Output: (given code above what's executed when it's run?)

```
> plot(waiting~eruptions, data=faithful,
cex=1, pch=19, col="blue", main="Old
Faithful Geyser Eruptions", ylab="Wait
time between eruptions", xlab="Duration of
eruption")
```

Examples of Use

- `plot( object, data)`
- `plot(object, data, cex, pch)`

Function Inputs

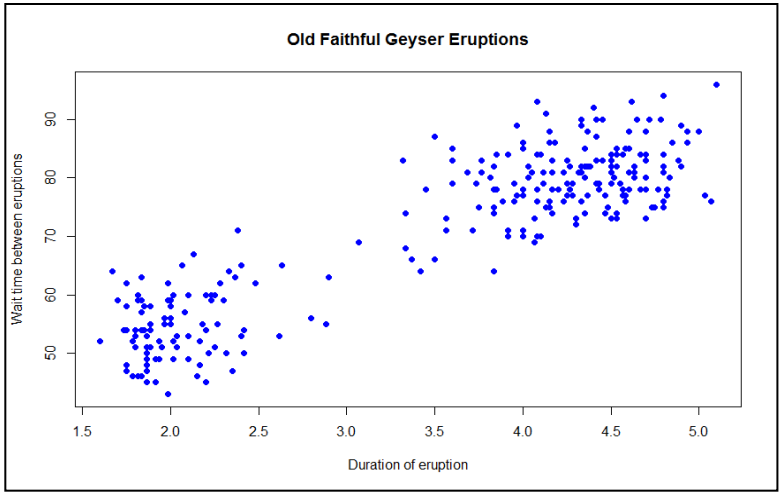
- `object` – an object of two variables. Relation between two vectors is created using " ~ " symbol
- `data` – dataset name
- `cex` – size of the data points in plot
- `pch` – shape of the data point in plot

Function Output

- Scatter plot showing relationship between two vectors

Additional Notes

- See `help(plot)` for more options and styles
- Try changing `cex` and `pch` values



cumprod

Compute cumulative product

Function Inputs

- `x` – vector

Function Output

- Computed cumulative product

```
Sample Code input:

x = 1:10
cumprod(x)
```

```
Code Output: (given code above what's executed when it's run?)

> cumprod(x)
 [1]       1       2       6      24     120     720    5040   40320  362880 3628800
```

cumsum

Compute cumulative sum

Function Inputs

- `x` – vector

Function Output

- Computed cumulative sum

```
Sample Code input:

x = 1:10
cumsum(x)
```

```
Code Output: (given code above what's executed when it's run?)

> cumsum(x)
 [1]  1  3  6 10 15 21 28 36 45 55
```

mean

Arithmetic/trimmed mean

```
Sample Code input:

x = c(1:100, 434:589)
mean(x)
mean(x, trim=.1)
mean(x > 500)     #proportion > 500
```

Function Inputs

- `x` – Vector
- `trim` – prop. of obs. to trim (for trimmed mean) from each end
- `na.rm` – remove missing values?

Function Output

- Computed mean

Additional Notes

- If any missing present, result is missing unless `na.rm = T`

```
Code Output: (given code above what's executed when it's run?)

> mean(x)
[1] 331.4219

> mean(x, trim=.1)
[1] 340.2621

> mean(x > 500)
[1] 0.3476562
```

median

Compute median

Function Inputs

- `x` – Vector
- `na.rm` – remove missing values?

Function Output

- Computed median

Additional Notes

- If any missing present, result is missing unless `na.rm = T`

```
Sample Code input:

x = c(1:100, 434:589)
median(x)
y = c(1:10, NA)
median(y)
median(y, na.rm=TRUE)
```

```
Code Output: (given code above what's executed when it's run?)

> median(x)
[1] 461.5

> median(y)
[1] NA

> median(y, na.rm=TRUE)
[1] 5.5
```

quantile

Sample quantiles

```
Sample Code input:

x = c(1:100, 434:589)
quantile(x, .25)
quantile(x, .5)        #median
quantile(x, .9)
```

Function Inputs

- `x` – Vector
- `probs` – probability for quantile computation
- `na.rm` – remove missing values?

Function Output

- Computed quantile

Additional Notes

- If any missing present, result is missing unless `na.rm = T`

```
Code Output: (given code above what's executed when it's run?)

> quantile(x, .25)
  25%
64.75

> quantile(x, .5)
  50%
461.5

> quantile(x, .9)
  90%
563.5
```

sample

Generate random samples or permutations

Function Inputs

- `x` – vector to choose the sample from or integer
- `size` – size of the generated random sample
- `replace` – Should sampling be done with replacement?
- `prob`– probabilities for weighted sampling

Function Output

- Generated random sample or permutation

Additional Notes

- Default is sampling without replacement
- `prob` – no need to sum to 1

```
Sample Code input:

sample(1:10, 2)
sample(1:10, 10, replace=TRUE)
sample(8)    #random permutation
sample(1:5, 5, replace = TRUE,
          prob = c(.7, .1, .1, .05, .05))
```

```
Code Output: (given code above what's executed when it's run?)

> sample(1:10, 2)
[1] 1 8

> sample(1:10, 10, replace=TRUE)
 [1]  1  1  2  4 10  1  7  8  4  2

> sample(8)
[1] 4 8 2 6 7 1 3 5

> sample(1:5, 5, replace = TRUE, prob = c(.7, .1, .1, .05, .05))
[1] 1 1 1 1 3
```

sd

Standard deviation

Sample Code input:

```
x = 1:10
sd(x)
```

Function Inputs

- `x` – vector
- `na.rm` – remove missing values?

Function Output

- Computed standard deviation

Additional Notes

- If any missing present, result is missing unless `na.rm = T`

Code Output: (given code above what's executed when it's run?)

```
> sd(x)
[1] 3.02765
```

sum

Compute sum

Function Inputs

- `x` – Vectors to sum

Function Output

- Computed sum

Additional Notes

- Can pass more than one vector

```
Sample Code input:

x = 1:10
y = 1:10
z = 1:10
sum(x,y,z)
```

```
Code Output: (given code above what's executed when it's run?)

> sum(x,y,z)
[1] 165
```

var

Compute variance

```
Sample Code input:

x = 1:10; y = seq(.5,1.5,len=10)
var(x)
var(x,y)
```

Function Inputs

- `x` – vector/matrix
- `y` – vector/matrix
- `na.rm` – remove missing values?

Function Output

- Computed variance/covariance

Additional Notes

- If only x is present, then variance of x is returned
- If x, y present, covariance of x,y is returned
- If x,y are matrices, covariance of cols of x,y is returned

```
Code Output: (given code above what's executed when it's run?)

> var(x)
[1] 9.166667

> var(x,y)
[1] 1.018519
```

weighted.mean

Compute weighted mean

```
Sample Code input:

x = 1:5
w = c(.2,.2,.3,.1,.2)
weighted.mean(x)  # same as mean(x)
weighted.mean(x,w)
```

Function Inputs

- `x` – vector
- `w` – vector of weights same length as x
- `na.rm` – remove missing values?

Function Output

- Computed weighted mean

Additional Notes

- If any missing present, result is missing unless `na.rm = T`

```
Code Output: (given code above what's executed when it's run?)

> weighted.mean(x)  # same as mean(x)
[1] 3

> weighted.mean(x,w)
[1] 2.9
```