

Foundations of R Software

Lecture 11

Basics of Calculations

⋮⋮⋮

Built in Functions and Assignments

Shalabh

Department of Mathematics and Statistics

Indian Institute of Technology Kanpur

R as a calculator

R can perform all standard calculations like addition, subtraction, multiplication, division etc.

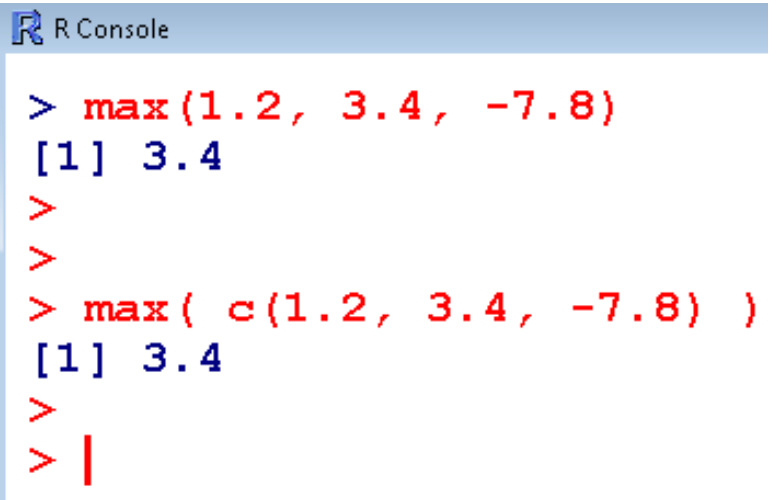
R has built in functions for computations.

Maximum

Operator: **max**

```
> max(1.2, 3.4, -7.8)
[1] 3.4
```

```
> max( c(1.2, 3.4, -7.8) )
[1] 3.4
```

A screenshot of an R console window with a blue header bar containing the R logo and the text "R Console". The console shows the following commands and output in red text: the command `> max(1.2, 3.4, -7.8)` followed by the output `[1] 3.4`; three empty lines; the command `> max( c(1.2, 3.4, -7.8) )` followed by the output `[1] 3.4`; and two more empty lines, with the second one ending in a vertical bar.

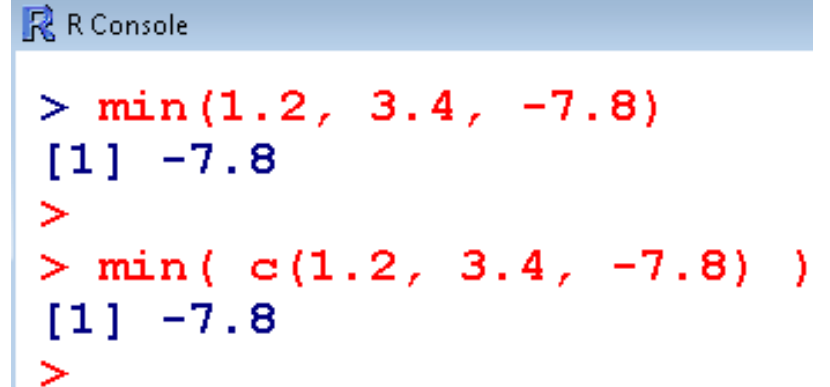
```
R Console
> max(1.2, 3.4, -7.8)
[1] 3.4
>
>
> max( c(1.2, 3.4, -7.8) )
[1] 3.4
>
> |
```

See the difference in use of **c** command.

Minimum

Operator: `min`

```
> min(1.2, 3.4, -7.8)
[1] -7.8
```

A screenshot of an R console window with a blue header bar containing the R logo and the text "R Console". The console shows two lines of code and their output. The first line is "> min(1.2, 3.4, -7.8)" in red, followed by the output "[1] -7.8" in blue. The second line is "> min(c(1.2, 3.4, -7.8))" in red, followed by the output "[1] -7.8" in blue. The prompt ">" is shown at the end of the second line.

```
> min(1.2, 3.4, -7.8)
[1] -7.8
>
> min( c(1.2, 3.4, -7.8) )
[1] -7.8
>
```

```
> min( c(1.2, 3.4, -7.8) )
[1] -7.8
```

See the difference in use of `c` command.

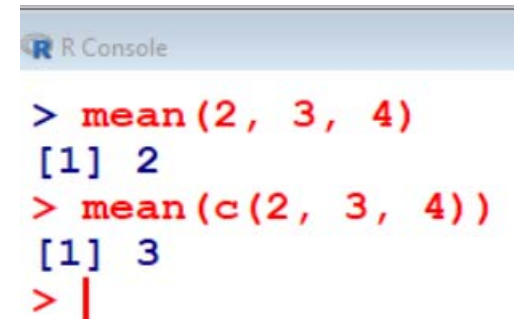
Arithmetic mean

Operator: **mean**

> mean(2, 3, 4) $\text{mean} = \frac{2+3+4}{3} = 3$

[1] 2

> mean(c(2, 3, 4))
[1] 3



```
R Console
> mean(2, 3, 4)
[1] 2
> mean(c(2, 3, 4))
[1] 3
> |
```

See the difference in use of **c** command.

Suggestion: Always use **c** command to input more than one data value.

Overview Over Further Functions

<code>abs()</code>	Absolute value
<code>sqrt()</code>	Square root
<code>round()</code> , <code>floor()</code> , <code>ceiling()</code>	Rounding, up and down
<code>sum()</code> , <code>prod()</code>	Sum and product
<code>log()</code> , <code>log10()</code> , <code>log2()</code>	Logarithms
<code>exp()</code>	Exponential function
<code>sin()</code> , <code>cos()</code> , <code>tan()</code> , <code>asin()</code> , <code>acos()</code> , <code>atan()</code>	Trigonometric functions
<code>sinh()</code> , <code>cosh()</code> , <code>tanh()</code> , <code>asinh()</code> , <code>acosh()</code> , <code>atanh()</code>	Hyperbolic functions

Examples

Operator: **abs** for finding the absolute values

```
> abs(-4)
[1] 4
```

```
> abs(c(-1, -2, -3, 4, 5))
[1] 1 2 3 4 5
```

 R Console

```
> abs(-4)
[1] 4
>
> abs(c(-1, -2, -3, 4, 5))
[1] 1 2 3 4 5
>
```

Examples

Operator: **sqrt** for finding the square root of values

```
> sqrt(4)
[1] 2
```

```
> sqrt(c(4, 9, 16, 25))
[1] 2 3 4 5
```

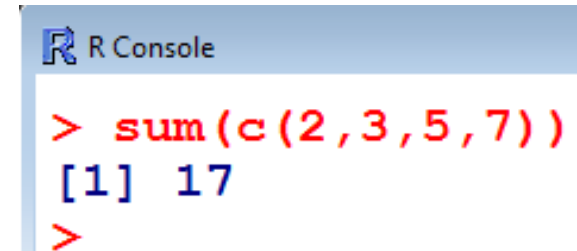
 R Console

```
> sqrt(4)
[1] 2
>
> sqrt(c(4, 9, 16, 25))
[1] 2 3 4 5
>
```

Examples

Operator: **sum** for finding the sum of values

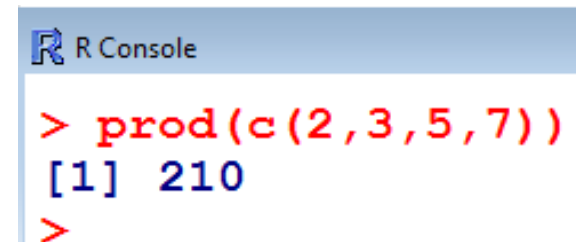
```
> sum(c(2, 3, 5, 7))  
[1] 17
```

A screenshot of an R console window. The title bar says "R Console". The prompt ">" is followed by the command "sum(c(2, 3, 5, 7))" in red text. The output "[1] 17" is shown in blue text. The prompt ">" is shown again in red text.

```
R Console  
> sum(c(2, 3, 5, 7))  
[1] 17  
>
```

Operator: **prod** for finding the product of values

```
> prod(c(2, 3, 5, 7))  
[1] 210
```

A screenshot of an R console window. The title bar says "R Console". The prompt ">" is followed by the command "prod(c(2, 3, 5, 7))" in red text. The output "[1] 210" is shown in blue text. The prompt ">" is shown again in red text.

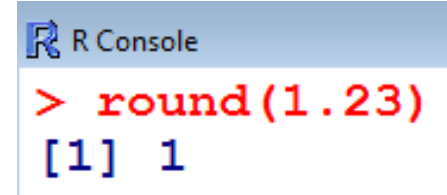
```
R Console  
> prod(c(2, 3, 5, 7))  
[1] 210  
>
```

Examples

Operator: **round** for finding the round off values

```
> round(1.23)
```

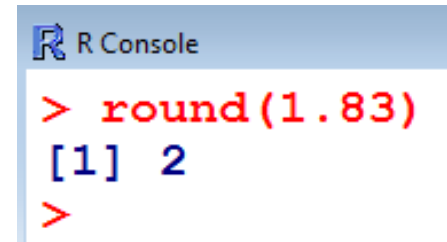
```
[1] 1
```



```
R Console  
> round(1.23)  
[1] 1
```

```
> round(1.83)
```

```
[1] 2
```



```
R Console  
> round(1.83)  
[1] 2  
>
```

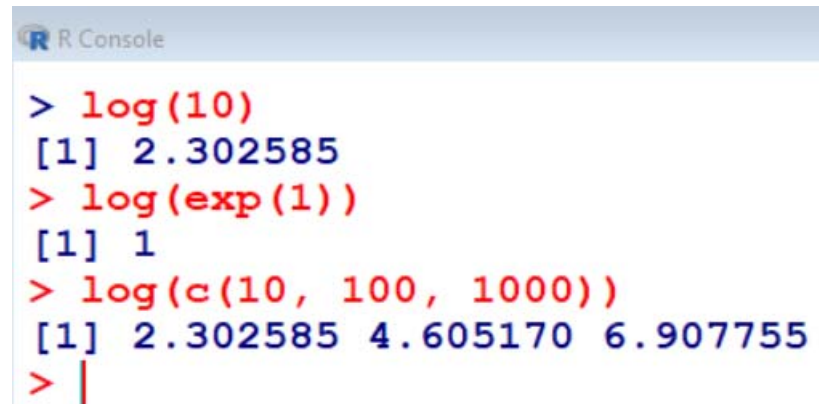
Examples

Operator: **log** for finding the natural log (ln) of values

```
> log(10)
[1] 2.302585
```

```
> log(exp(1))
[1] 1
```

```
> log(c(10, 100, 1000))
[1] 2.302585 4.605170 6.907755
```

A screenshot of an R console window with a light blue header bar containing the R logo and the text "R Console". The console shows four lines of code and their corresponding outputs. The first line is "> log(10)" in red, followed by "[1] 2.302585" in blue. The second line is "> log(exp(1))" in red, followed by "[1] 1" in blue. The third line is "> log(c(10, 100, 1000))" in red, followed by "[1] 2.302585 4.605170 6.907755" in blue. The fourth line is "> |" in red, with a vertical cursor bar at the end.

```
> log(10)
[1] 2.302585
> log(exp(1))
[1] 1
> log(c(10, 100, 1000))
[1] 2.302585 4.605170 6.907755
> |
```

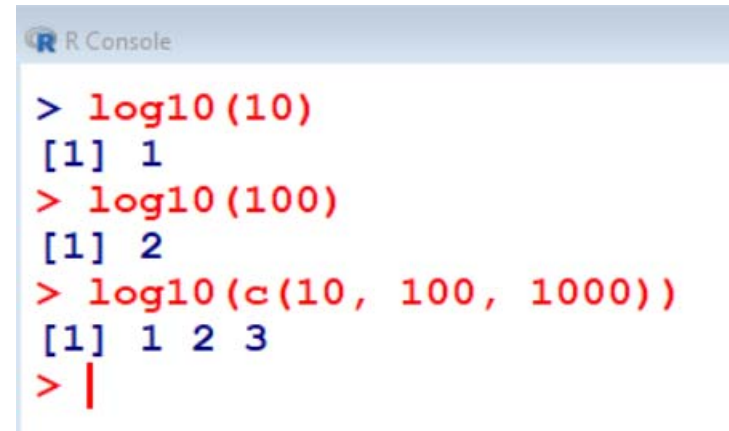
Examples

Operator: **log10** for finding the log (with base 10) of values

```
> log10(10)
[1] 1
```

```
> log10(100)
[1] 2
```

```
> log10(c(10, 100, 1000))
[1] 1 2 3
```

A screenshot of an R Console window. The title bar says "R Console". The console shows three lines of input and output. The first line is "> log10(10)" followed by "[1] 1". The second line is "> log10(100)" followed by "[1] 2". The third line is "> log10(c(10, 100, 1000))" followed by "[1] 1 2 3". The prompt "> |" is visible at the bottom.

```
> log10(10)
[1] 1
> log10(100)
[1] 2
> log10(c(10, 100, 1000))
[1] 1 2 3
> |
```

Assignments

An assignment can also be used to save values in variables:

```
> x1 = c(1,2,3,4)
```

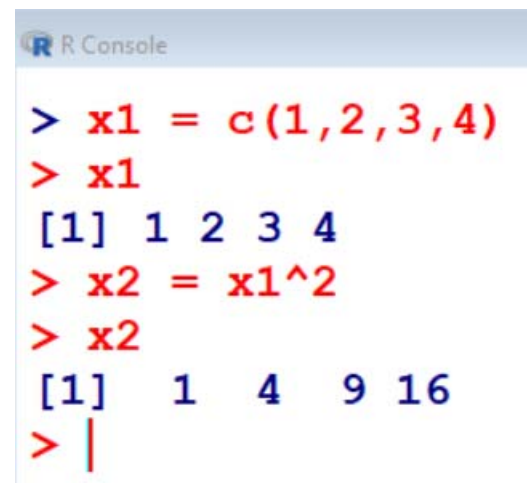
```
> x1
```

```
[1] 1 2 3 4
```

```
> x2 = x1^2
```

```
> x2
```

```
[1] 1 4 9 16
```

A screenshot of an R console window showing the execution of R code. The code includes creating a vector x1, printing it, creating a new vector x2 from x1 squared, and printing x2. The console output shows the values of the vectors.

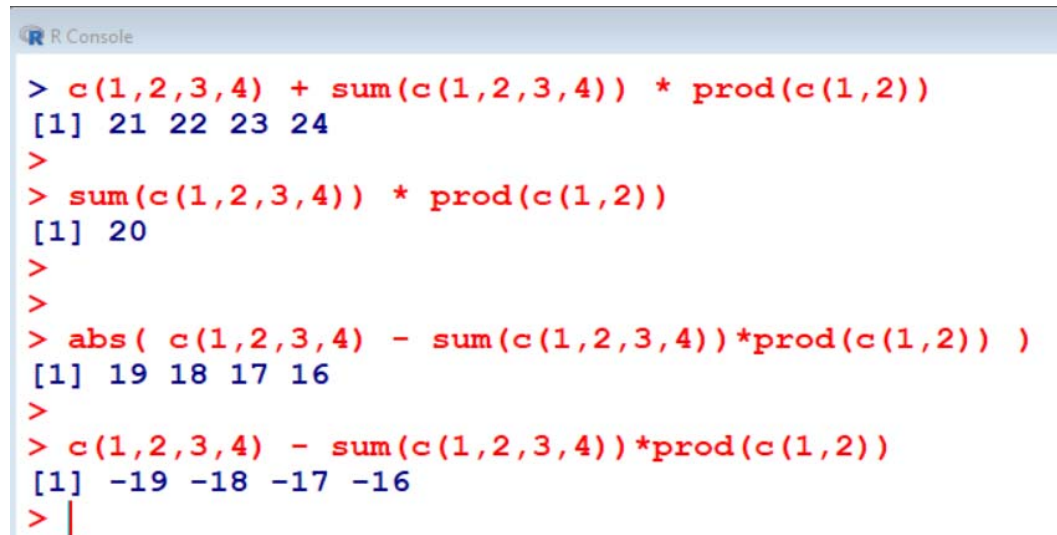
```
R Console  
> x1 = c(1,2,3,4)  
> x1  
[1] 1 2 3 4  
> x2 = x1^2  
> x2  
[1] 1 4 9 16  
> |
```

Assignments

Calculations with scalar, data vector and built in functions also work.

```
> c(1,2,3,4) + sum(c(1,2,3,4)) * prod(c(1,2))  
[1] 21 22 23 24
```

```
> abs( c(1,2,3,4) - sum(c(1,2,3,4))*prod(c(1,2)) )  
[1] 19 18 17 16
```



A screenshot of an R console window. The title bar says 'R Console'. The console shows the following code and output:

```
> c(1,2,3,4) + sum(c(1,2,3,4)) * prod(c(1,2))  
[1] 21 22 23 24  
>  
> sum(c(1,2,3,4)) * prod(c(1,2))  
[1] 20  
>  
>  
> abs( c(1,2,3,4) - sum(c(1,2,3,4))*prod(c(1,2)) )  
[1] 19 18 17 16  
>  
> c(1,2,3,4) - sum(c(1,2,3,4))*prod(c(1,2))  
[1] -19 -18 -17 -16  
> |
```