Performing operations *(Bewerkingen uitvoeren)*

Rekenen

Rekenkundige operatoren, vermeld in de onderstaande tabel, worden gebruikt om expressies te maken in R Script-programma's die een enkele resulterende waarde retourneren.

Operator	Bewerking:
+	Optelling
-	Aftrekking
*****	Vermenigvuldiging
/	Deling **Reële** getallen
%/%	Deling **Gehele** getallen
^	Machtsverheffing
%%	Modulus

1. Open RStudio en klik vervolgens op **File**, **New File**, **R Script** of druk op de toetsen **Ctrl + Shift + N** om een nieuw Code Editor venster te openen.

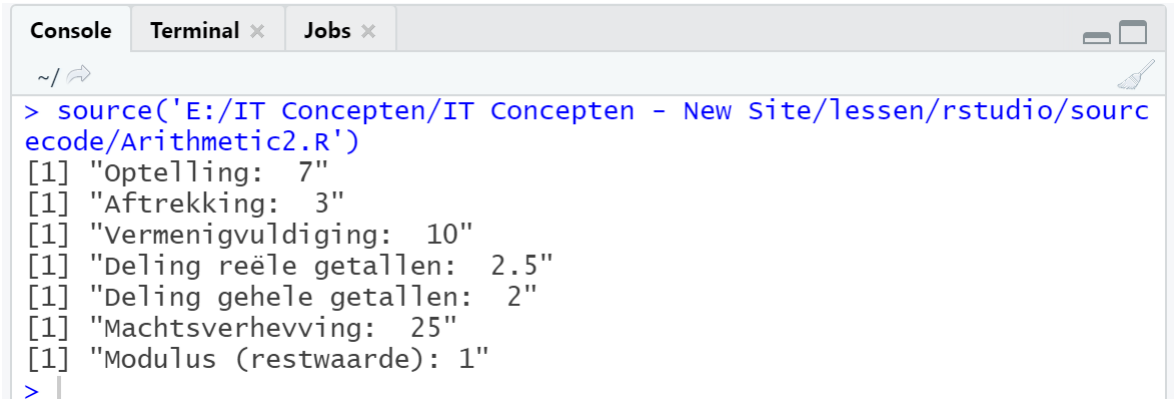
2. Voer de onderstaande broncode in de Code Editor in. Laat de groene commentaarregels weg.

Arithmetic.R*

Source on Save | Run | Source

```
1  #Maak twee integer variabelen aan met waarden voor bewerking
2  large <- 5
3  small <- 2
4
5  #Voeg instructies toe om het resultaat van enkele rekenkundige
6  #basisbewerkingen uit te voeren
7  print(paste("Optelling: ",large + small))
8  print(paste("Aftrekking: ",large - small))
9  print(paste("Vermenigvuldiging: ", large * small))
10
11 #Voeg instructies toe om het resultaat van de twee soorten
12 #delingsbewerkingen uit te voeren.
13 print(paste("Deling reële getallen: ", large / small))
14 print(paste("Deling gehele getallen: ", large %/% small))
15
16 #Voeg een statement toe om het resultaat van een
17 #machtsverheffing uit te voeren
18 print(paste("Machtsverhevving: ", large ^ small))
19
20 #Voeg een instructie toe om de rest uit te voeren na het
21 #uitvoeren van een delingsbewerking.
22 print(paste("Modulus (restwaarde):", large %% small))
23
```

3 Sla het **R Script** op met de naam **Arithmetic.R** en klik vervolgens op de knop **Source** of druk op **Ctrl + Shift + S** om de rekenkundige uitvoer te zien.



Vergelijkingen maken

Vergelijkingsoperatoren die in de onderstaande tabel worden vermeld, worden gebruikt om twee waarden in n-expressie te vergelijken en een enkele Booleaanse waarde TRUE of FALSE te retourneren, waarmee het resultaat van die vergelijking wordt beschreven.

Operator:	Vergelijking:
==	Gelijkwaardigheid
!=	Ongelijkheid
>	Groter dan
>=	Groter dan of gelijk aan
<	Kleiner dan
<=	Kleiner dan of gelijk aan

1 Open RStudio en klik vervolgens op **File**, **New File**, **R Script** of druk op de toetsen **Ctrl + Shift + N** om een nieuw Code Editor venster te openen.

2 Voer de onderstaande broncode in de Code Editor in. Laat de groene commentaarregels weg.

Comparison.R

Source on Save | Run | Source

```
#Maak drie variabelen met gehele waarden voor vergelijking
nil <- 0
num <- 0
max <-1

#Maak twee variabelen met karakterwaarden ter vergelijking
cap <-"A"
low <- "a"

#Voeg instructies toe om het resultaat van een gelijkheids
#vergelijking van integer- en karakterwaarden uit te voeren
print(paste("0 == 0 Gelijkheid: ", nil == num))
print(paste("A == a Gelijkheid: ", cap == low))

#Voeg instructies toe om het resultaat van de ongelijkheidsvergelijking
#van gehele waarden uit te voeren
print(paste("0 != 1 Ongelijkheid:", nil != max))

#Voeg instructies toe om de resultaten van groter dan en kleiner dan
#vergelijkingen van gehele waarden uit te voeren
print(paste("0 > 1 Groter dan:", nil > max))
print(paste("0 < 1 Kleiner dan:", nil < max))

#voeg instructies toe om de resultaten van grotere of gelijke en kleinere
#of gelijke vergelijkingen van gehele waarden uit te voeren
print(paste("0 >= 1 Groter dan of gelijk: "), nil >= num)
print(paste("1 <= 0 Kleiner dan of gelijk: ", max <= nil))
```

Spellcheck

3 Sla het **R Script** op met de naam **Comparison.R** en klik vervolgens op de knop  **Source** of druk op **Ctrl + Shift + S** om de rekenkundige uitvoer te zien.

Console | Terminal | Jobs

```
> source('E:/IT Concepten/IT Concepten - New Site/lessen/rstudio/sourcecode/Compariso
n.R')
[1] "0 == 0 Gelijkheid:  TRUE"
[1] "A == a Gelijkheid:  FALSE"
[1] "0 != 1 Ongelijkheid: TRUE"
[1] "0 > 1 Groter dan: FALSE"
[1] "0 < 1 Kleiner dan: TRUE"
[1] "0 >= 1 Groter dan of gelijk: "
[1] "1 <= 0 Kleiner dan of gelijk:  FALSE"
>
```

Logica beoordelen

Logische operators, vermeld in de onderstaande tabel, kunnen worden gebruikt om meerdere expressies te combineren die elk een Booleaanse waarde retourneren in een expressie die één enkele Booleaanse waarde retourneert.

Operator:	Bewerking:
!	Logische NIET
&&	Logische EN
&	Element-gewijze Logische EN
||	Logische OF
|	Element-gewijze Logische OF

Logische operators worden gebruikt met operandi die de Booleaanse waarden **TRUE** of **FALSE** hebben, of waarden die kunnen worden geconverteerd naar **TRUE** of **FALSE**. Bij **R**-programmering wordt **nul** als **FALSE** beschouwd en worden alle andere getallen als **TRUE** beschouwd.

Het logische **! NOT**-operator is een "unaire" operator die vóór een enkele operand wordt gebruikt. Het retourneert de inverse Booleaanse waarde van de gegeven operand - waarbij **TRUE** wordt omgekeerd in **FALSE** en **FALSE** in **TRUE**.

De logische **&&** AND-operator evalueert het eerste element van twee operanden en retourneert alleen TRUE als beide operanden zelf **TRUE** zijn. Anders retourneert de logische operator **&& FALSE**. De elementaire logische **&** operator voert dezelfde bewerking uit, maar op alle elementen van de operanden.

In tegenstelling tot de logische operator **&&**, die twee operanden nodig heeft om **TRUE** te zijn, is de logische **||** De **OR**-operator evalueert het eerste element van zijn twee operanden en retourneert **TRUE** als een van de operanden **TRUE** is - hij retourneert alleen **FALSE** als geen van beide operands **TRUE** is. De element-gewijze logische **|** operator voert dezelfde bewerking uit, maar op alle elementen van de operanden.

Als de twee operanden een verschillend aantal elementen hebben, zal het resultaat even lang zijn als de operand met de meeste elementen.

1. Open RStudio en klik vervolgens op **File**, **New File**, **R Script** of druk op de toetsen **Ctrl + Shift + N** om een nieuw Code Editor venster te openen.

2 Voer de onderstaande broncode in de Code Editor in. Laat de groene commentaarregels weg.

Logic.R

```
1  #Maak een variable die een Boolean waarde bevat
2  active <- TRUE
3
4  #Voeg een instructie toe om de inverse van de opgeslagen
5  #Booleaanse waarde uit te voeren.
6  print(paste("NOT logic !active:", !active))
7
8  #Voeg een instructie toe om de inverse van de opgeslagen
9  #Booleaanse waarde uit te voeren.
10 flags <- c(TRUE, TRUE, FALSE, (1>0),0)
11 marks <- c(FALSE,TRUE, TRUE, 16, FALSE)
12
13 #Voeg een instructie toe om het resultaat van de logische
14 #EN- en logische OF-evaluatie van alleen de eerste elementen
15 #uit te voeren.
16 print(paste("AND logic:", flags && marks))
17 print(paste("OR logic:", flags || marks))
18
19 #Wijs het resultaat van de logische EN- en OF-evaluatie van
20 #alle elementen twee andere variabelen toe.
21 and.result <- flags & marks
22 or.result <- flags | marks
23
```

3 Sla het **R Script** op met de naam **Logic.R** en klik vervolgens op de knop **Source** of druk op **Ctrl + Shift + S** om de rekenkundige uitvoer te zien.

```
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> source('E:/IT Concepten/IT Concepten - New Site/lessen/rstudio/so
urcecode/Logic2.R')
[1] "NOT logic !active: FALSE"
[1] "AND logic: FALSE"
[1] "OR logic: TRUE"
>
```

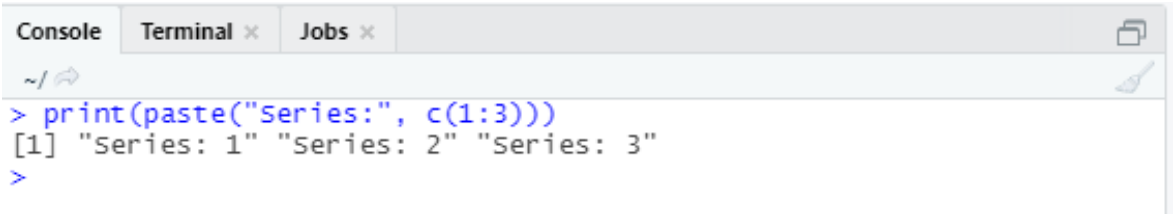
Name	Type	Len...	Size	Value
active	logic...	1	56 B	TRUE
and.res...	logic...	5	80 B	logi [1:5] FA...
flags	numer...	5	96 B	num [1:5] 1 1...
marks	numer...	5	96 B	num [1:5] 0 1...
or.resu...	logic...	5	80 B	logi [1:5] TR...

Werken op elementen

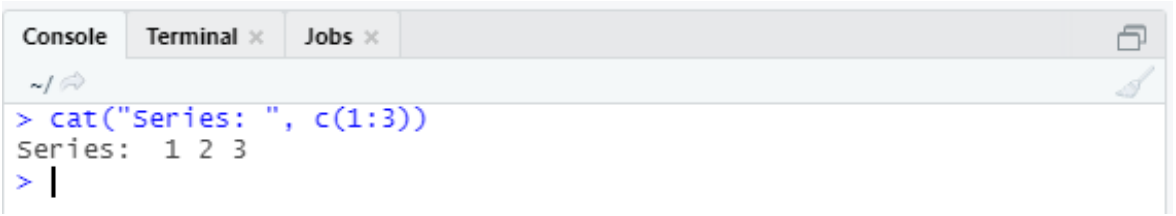
De elementen van een vector kunnen eenvoudig worden gevuld met een numerieke reeks met behulp van de **:** dubbele punt operator (*colon*). Een numerieke "***from***"-operand wordt gevraagd vóór de **:**-operator om een bereik op te geven. De bewerking genereert een inclusieve reeks, in stappen van één, tussen het opgegeven bereik. De gegenereerde reeks zal oplopend of aflopend zijn volgens de gespecificeerde "***from***" en "***to***" waarden.

Een numerieke reeks kan worden opgegeven als argument voor de combinatiefunctie **c( )** om de elementen van een vector te vullen. Bovendien kan een numerieke reeks worden gespecificeerd tussen **[ ]** vierkante haken om een indexbereik te specificeren - om een "***slice***" ofwel een stuk van een vector te kopiëren.

Als je alle elementen van een vector naar de console wilt uitvoeren, wordt de combinatie van de functies **print( )** en **paste( )** voor elk element aangeroepen. Beschouw bijvoorbeeld deze instructie en de uitvoer ervan:



Dit is misschien niet wat je wilt, maar R biedt een handige ingebouwde **cat()**-functie die zijn argumenten samenvoegt (*joins*) en ze vervolgens als volgt naar de console uitvoert:



Een van de grote voordelen van **R**-variabelen is de mogelijkheid om vectorberekeningen uit te voeren op al hun numerieke elementwaarden door simpelweg een rekenkundige operator tussen variabele namen te plaatsen. Als de vectoren een gelijk aantal elementen hebben, wordt de bewerking uitgevoerd tussen het overeenkomstige element in elke vector. Als de vectoren een ongelijk aantal elementen hebben, wordt de kortere "gerecycleerd" om overeen te komen met de langere vectorlengte. Als de grootte van de langere vector geen exact veelvoud is van de kortere-vector, wordt de bewerking uitgevoerd, maar geeft de R Interpreter een waarschuwing.

1 Open RStudio en klik vervolgens op **File**, **New File**, **R Script** of druk op de toetsen **Ctrl + Shift + N** om een nieuw Code Editor venster te openen.

2 Voer de onderstaande broncode in de Code Editor in. Laat de groene commentaarregels weg.

VectorArithmethic.R

Source on Save | Run | Source

```
#Maak een variabele met een numerieke reeks
#van één tot negen.
series <- c(1:9)

#Voeg een instructie toe om een tekenreeks,
#de numerieke reeks en een nieuwe regel uit te voeren.
cat("Series: ",series, "\n")

#Maak een tweede variabele met een deel van de
#elementwaarden van de eerste variabele en voer
#die reeks vervolgens uit.
slice <- series[1:3]
cat("Slice:", slice, "\n")

#Maak een derde variabele die het totaal van elementwaarden
#in de andere twee variabelen bevat en voer vervolgens de
#totalen uit.
totals <- series + slice
cat("Totals:", totals, "\n")

#Verleng het segment en voer vervolgens die reeks uit.
slice <- series[1:4]
cat("New Slice:", slice, "\n")

#Bereken het totaal van de elementwaarden die zich nu in de
#andere twee variabelen bevinden en voer deze totalen uit.
totals <- series + slice
cat("New Totals:", totals, "\n")
```

3 Sla het **R Script** op met de naam **VectorArithmetic.R** en klik vervolgens op de knop **Source** of druk op **Ctrl + Shift + S** om de rekenkundige uitvoer te zien.

Console | Terminal | Jobs

```
> source('E:/IT Concepten/IT Concepten - New Site/lessen/rstudio/so
urcecode/VectorArithmethic2.R')
Series:  1 2 3 4 5 6 7 8 9
Slice: 1 2 3
Totals: 2 4 6 5 7 9 8 10 12
New Slice: 1 2 3 4
New Totals: 2 4 6 8 6 8 10 12 10
Warning message:
In series + slice :
  longer object length is not a multiple of shorter object length
>
```

Elementen vergelijken

Net zoals rekenkundige operatoren kunnen worden gebruikt om berekeningen uit te voeren op elementen binnen twee vectoren, zo kunnen vergelijkingsoperatoren worden gebruikt om elementen binnen twee vectoren te vergelijken. Er kan een vergelijking worden gemaakt van zowel numerieke als tekstreekswaarden. Dit resultaat wordt geretourneerd als een vector van Booleaanse **TRUE**- en **FALSE**-waarden die elke overeenkomstige elementvergelijking aangeven.

Bij het vergelijken van korte vectoren van slechts een paar elementen is het gemakkelijk om het resultaat te bepalen door de geretourneerde vector van Booleans te onderzoeken, maar dit wordt moeilijker bij het vergelijken van grotere vectoren. R biedt hiervoor een ingebouwde **which( )** functie. Deze functie accepteert een Booleaanse vector als argument en retourneert een lijst met indexnummers die een **TRUE**-waarde bevatten.

Vergelijking van tekstreeksen om overeenkomende waarden te ontdekken binnen corresponderende elementen van twee vectoren kan worden gemaakt met de **==** gelijkheidsoperator, maar het hoofdlettergebruik en de volgorde moeten precies identiek zijn om de vergelijking een **TRUE**-waarde te laten opleveren.

Vergelijking van tekstreeksen om overeenkomende waarden te ontdekken binnen elk element van twee vectoren kan worden gemaakt met de ingebouwde **intersect( )**-functie. Dit accepteert de namen van twee vectorvariabelen als een door komma's gescheiden lijst en retourneert alle waarden die precies overeenkomen:

1. Open RStudio en klik vervolgens op **File**, **New File**, **R Script** of druk op de toetsen **Ctrl + Shift + N** om een nieuw Code Editor venster te openen.

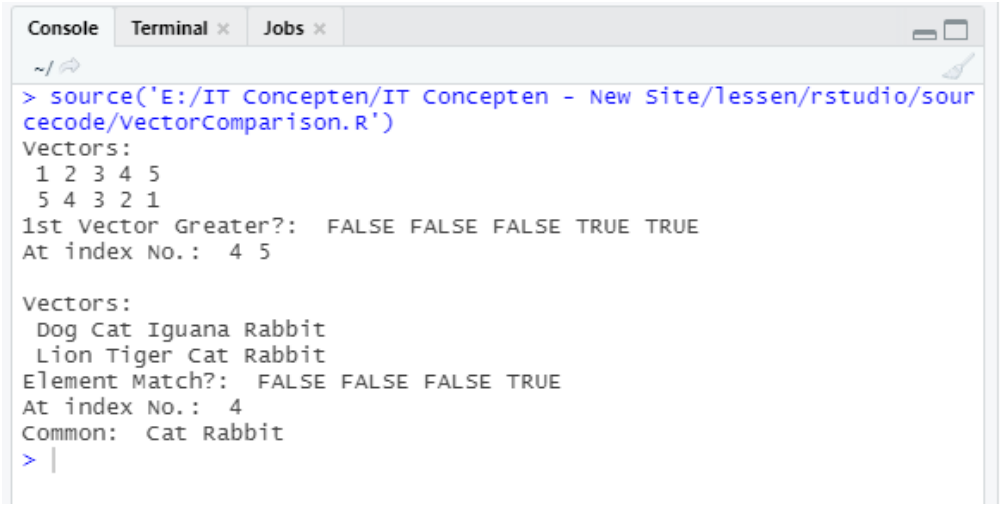
2. Voer de onderstaande broncode in de Code Editor in. Laat de groene commentaarregels weg.

VectorComparison.R

Source on Save | Run | Source

```r
#Maak twee variabelen die elk een numerieke reeks bevatten.
ascend <- c(1:5)
descend <- c(5:1)

#Voeg een instructie toe om een tekstreeks en de numerieke
#reeksen uit te voeren, opgemaakt met nieuwe regels.
cat("Vectors: \n", ascend, "\n", descend)

#Vergelijk de numerieke waarden binnen elk corresponderend
#element van de twee vectoren.
result <- ascend > descend

#Voer de retourvector van Booleaanse waarden uit.
cat("\n1st Vector Greater?: ", result)

#Voer de indexnummers uit die een TRUE-waarde bevatten.
cat("\nAt index No.: ", which(result))

#Maak twee variabelen die elk tekenreekswaarden bevatten
#binnen elk element.
pets <- c("Dog", "Cat", "Iguana", "Rabbit")
animals <- c("Lion", "Tiger", "Cat", "Rabbit")

#Voeg een instructie toe om een tekstreeks en de
#elementwaarden uit te voeren, opgemaakt met nieuwe regels.
cat("\n\nVectors: \n", pets, "\n", animals)

#Vergelijk de tekenreekswaarden binnen elk corresponderend
#element van de twee vectoren.
result <- pets == animals

#Voer de retourvector van Booleaanse waarden uit.
cat("\nElement Match?: ", result)

#Voer de indexnummers uit die een TRUE-waarde bevatten.
cat("\nAt index No.: ", which(result))

#Toon de overeenkomende waarden uit binnen alle elementen
#van de twee vectoren.
cat("\nCommon: ", intersect(pets, animals))
```

3 Sla het **R Script** op met de naam **VectorComparison.R** en klik vervolgens op de knop **Source** of druk op **Ctrl + Shift + S** om de rekenkundige uitvoer te zien.



Voorrang herkennen

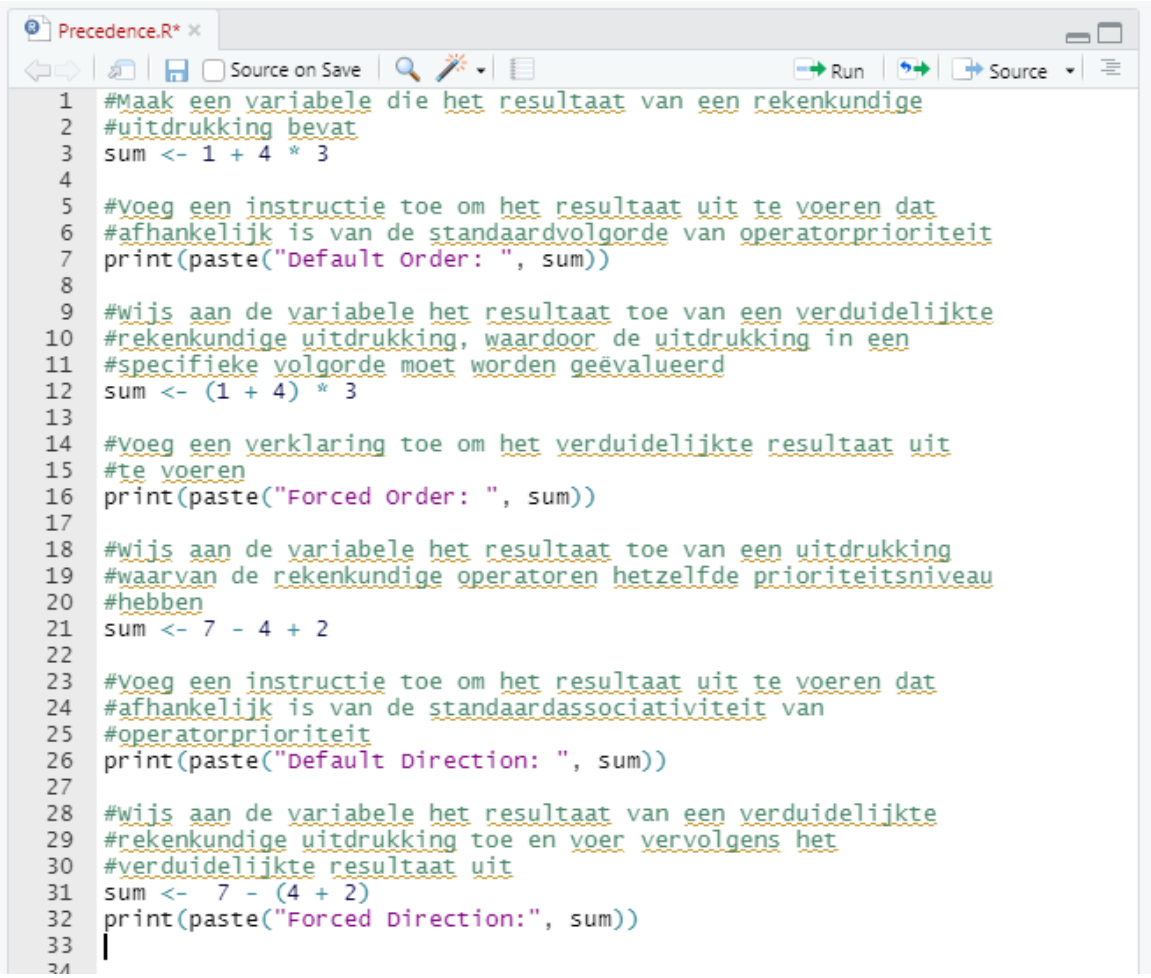
De operatorprioriteit bepaalt de volgorde waarin R uitdrukkingen evalueert.

Wanneer bewerkingen dezelfde prioriteit hebben, bepaalt hun "associativiteit" hoe expressies worden gegroepeerd. De -aftrekken-operator is bijvoorbeeld links-associatief, groepeert van links naar rechts (LTR), dus **8 - 4 - 2** wordt gegroepeerd als **(8 - 4) - 2** en komt dus uit op 2. Andere operatoren zijn rechts-associatief, groeperen van rechts naar links (RTL).
In de onderstaande tabel staan veelvoorkomende operators in volgorde van prioriteit, met de hoogste prioriteit bovenaan. Operators op dezelfde regel hebben dezelfde prioriteit, dus de associativiteit van operators bepaalt hoe expressies worden gegroepeerd en geëvalueerd.

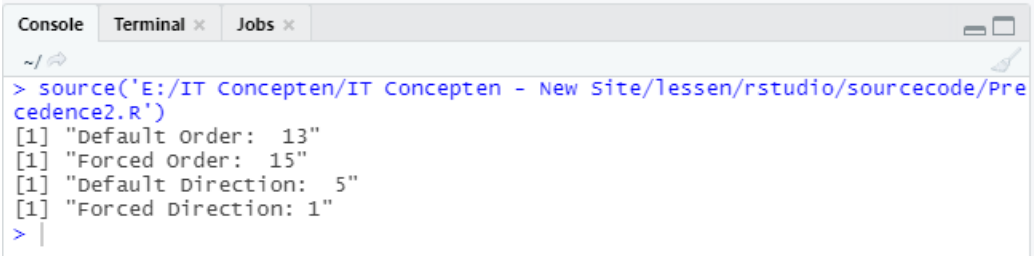
Categorie:	Operator:	Associativiteit:
Subset	$	LTR ☞
Exponent	^	☜ RTL
Sign (unary)	+ -	LTR ☞
Sequence	:	LTR ☞
Modulus	%% (and %/%)	LTR ☞
Mulyiplicative	• /	LTR ☞
Additive	+ -	LTR ☞
Comparative	< <= > >= == !=	LTR ☞
Logical NOT	!	LTR ☞
Logical AND	&& &	LTR ☞
Logical OR	|| |	LTR ☞
Assignment	=	☜ RTL
Assignment	<-	☜ RTL
Help	? ??	LTR ☞

1 Open RStudio en klik vervolgens op **File**, **New File**, **R Script** of druk op de toetsen **Ctrl + Shift + N** om een nieuw Code Editor venster te openen.

2 Voer de onderstaande broncode in de Code Editor in. Laat de groene commentaarregels weg.



3 Sla het **R Script** op met de naam **Precedence.R** en klik vervolgens op de knop **Source** of druk op **Ctrl + Shift + S** om de rekenkundige uitvoer te zien.



Elementen manipuleren

Naast de verschillende operatoren die in dit hoofdstuk worden beschreven, biedt R een aantal ingebouwde functies die kunnen worden gebruikt om de elementen binnen vectorvariabelen te manipuleren. Elk van de onderstaande functies accepteert een vector als argument:

- **De sort( ) functie**
  - Sorteert de elementwaarden in numerieke of alfabetische volgorde. Standaard worden de elementwaarden in oplopende volgorde gesorteerd, maar ze kunnen in aflopende volgorde worden gesorteerd door een argument **decreasing=TRUE** op te nemen in de functieaanroep. Elementen met ontbrekende waarden, in **R** aangeduid met een **NA**-constante waarde, worden standaard automatisch verwijderd, maar kunnen aan het einde van de opdracht worden behouden door een argument **na.last=TRUE** op te nemen in de functieaanroep.
- **De rev( ) functie**
  - Keert de volgorde van alle elementen binnen de vectorvariabele om.
- **De unique( ) functie**
  - Verwijdert elementen met dubbele waarden uit de vectorvariabele.

1 Open RStudio en klik vervolgens op **File**, **New File**, **R Script** of druk op de toetsen **Ctrl + Shift + N** om een nieuw Code Editor venster te openen.

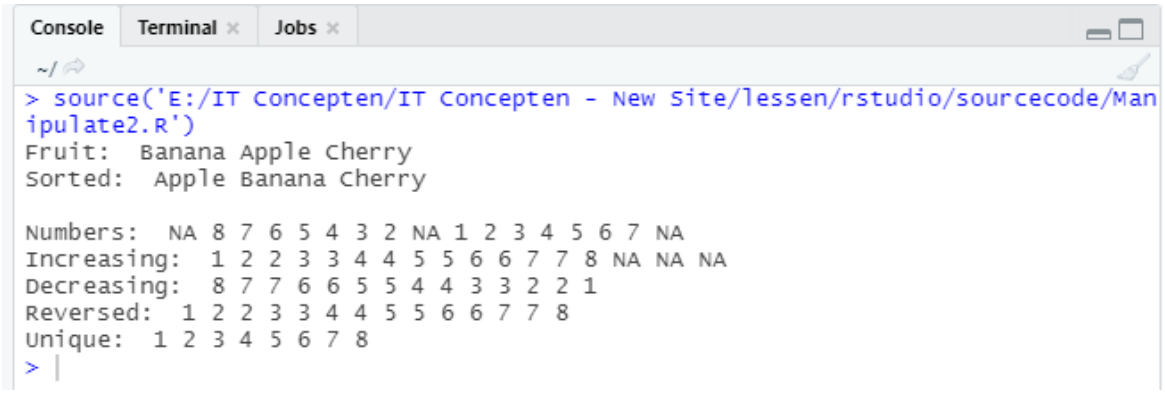
2 Voer de onderstaande broncode in de Code Editor in. Laat de groene commentaarregels weg.

Manipulate.R

Source on Save | Run | Source

```r
#Maak een vectorvariabele met drie tekenreekswaarden
fruit <- c("Banana", "Apple", "Cherry")

#Voeg een instructie toe om de waarden in elk element
#uit te voeren - in hun huidige volgorde
cat("Fruit: ", fruit, "\n")

#Wijs een gesorteerde rangschikking van de elementen
#toe aan de vectorvariabele, met behulp van de standaard
#oplopende volgorde
fruit <- sort(fruit)

#Voer de waarden in elk element uit - in hun nieuwe
#gesorteerde volgorde.
cat("Sorted: ", fruit, "\n\n")

#Maak een tweede vectorvariabele met numerieke waarden
#en enkele elementen met ontbrekende waarden
nums <- c(NA, 8:2, NA, 1:7, NA)

#Voer de waarden in elk element van de tweede variabele
#uit - in hun huidige volgorde
cat("Numbers: ", nums, "\n")

#Wijs een gesorteerde rangschikking van de elementen
#toe aan de variabele, gebruik de standaard oplopende
#volgorde en plaats aan het einde elementen met
#ontbrekende waarden.
nums <- sort(nums, na.last=TRUE)

#Voer de waarden in elk element uit - in hun nieuwe
#gesorteerde volgorde
cat("Increasing: ", nums, "\n")

#Wijs een gesorteerde rangschikking van de elementen
#toe aan de variabele in aflopende volgorde en voer
#vervolgens de waarden uit
nums <- sort(nums, decreasing=TRUE)
cat("Decreasing: ", nums, "\n")

#Assign a reversed arrangement of the elements to the
#variable, then output the values
nums <- rev(nums)
cat("Reversed: ", nums, "\n")

#Wijs alleen elementen met unieke waarden toe aan de
#vectorvariabele en voer vervolgens de waarden uit
nums <- unique(nums)
cat("Unique: ", nums, "\n")
```

3 Sla het **R Script** op met de naam **Manipulate.R** en klik vervolgens op de knop **Source** of druk op **Ctrl + Shift + S** om de rekenkundige uitvoer te zien.



De functie **sort()** kan ook een argument **na.last=FALSE** bevatten om elementen met ontbrekende waarden aan het begin van de volgorde te behouden.

Als de aanroep van de functie **sort()** geen argument **na.last** specificeert, worden elementen met ontbrekende waarden automatisch verwijderd.