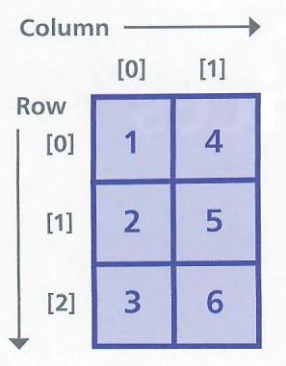
Building matrices *(Matrices maken)*

Building a matrix



In R-programmering is een "matrix" een tweedimensionale structuur die gegevens opslaat in een tabelindeling van cel rijen en cel kolommen. Net als bij vectorstructuren moeten de waarden die in een matrix zijn opgeslagen allemaal van hetzelfde gegevenstype zijn.

Een matrix kan worden gemaakt in **R** Script door een unieke identificatienaam naar keuze in de code-editor te schrijven en vervolgens waarden toe te wijzen met behulp van de ingebouwde **matrix()**-functie. Deze functie vereist een vector met de waarden als argument, gevolgd door **nrow=** en **ncol=** argumenten om het gewenste aantal rijen en kolommen op te geven dat u wilt maken. Het aantal rijen en kolommen moet overeenkomen met de lengte van de toegewezen vector of een exact veelvoud zijn van de lengte, anders verschijnt er een waarschuwingsbericht. In dit geval wordt de matrix nog steeds gemaakt, maar worden de vectorwaarden in extra cellen gerecycled. Een matrix kan worden gebruikt om voor elke dag van een jaar een waarde vast te leggen in een tabel van 52 rijen (*één per week*) en 7 kolommen (*één per dag*), zoals deze:

daily.record <- matrix(vector, nrow=52, ncol=7)

Individuele waarden worden in een matrix geadresseerd met behulp van de juiste inde-nummers van de rij en kolom. In dit geval kunt u bijvoorbeeld de waarde voor de tweede dag in de derde week ophalen met **daily.record[3, 2]**. Nieuwe waarden kunnen ook worden toegewezen aan de afzonderlijke cellen met behulp van hun indexnummer van rij en kolom. Bijvoorbeeld voor de eerste dag van de zesde week, als volgt:

daily.rocord[6,1] <- value

Om te bevestigen dat een structuur inderdaad een matrix is, biedt **R** een ingebouwde functie **is.matrix()** die **TRUE** retourneert wanneer het opgegeven argument een matrixobject is.

Het is handig om een waarde in een matrix te zoeken met behulp van de ingebouwde functie **which()**. Dit vereist een testuitdrukking als argument, beginnend met de matrixnaam en de te zoeken waarde. De functie **which()** onderzoekt de cellen alsof ze een vector zijn en geeft, als de gezochte waarde bestaat, de indexnummers terug waarop de waarde

in de vector wordt gevonden. Als de gezochte waarde niet wordt gevonden, retourneert de functie nul. Optioneel kunt je een argument **arr.ind=TRUE** toevoegen aan de functie **which()**, zodat het de rij- en kolom-indexnummer van elke cel die de gezochte waarde bevat, retourneert.

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.

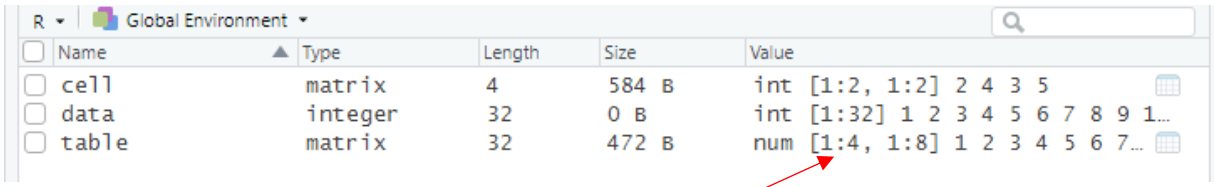
2 Voer de onderstaande broncode in de Code Editor in. Laat de groene commentaarregels weg, maar lees ze wel goed.

FirstMatrix.R

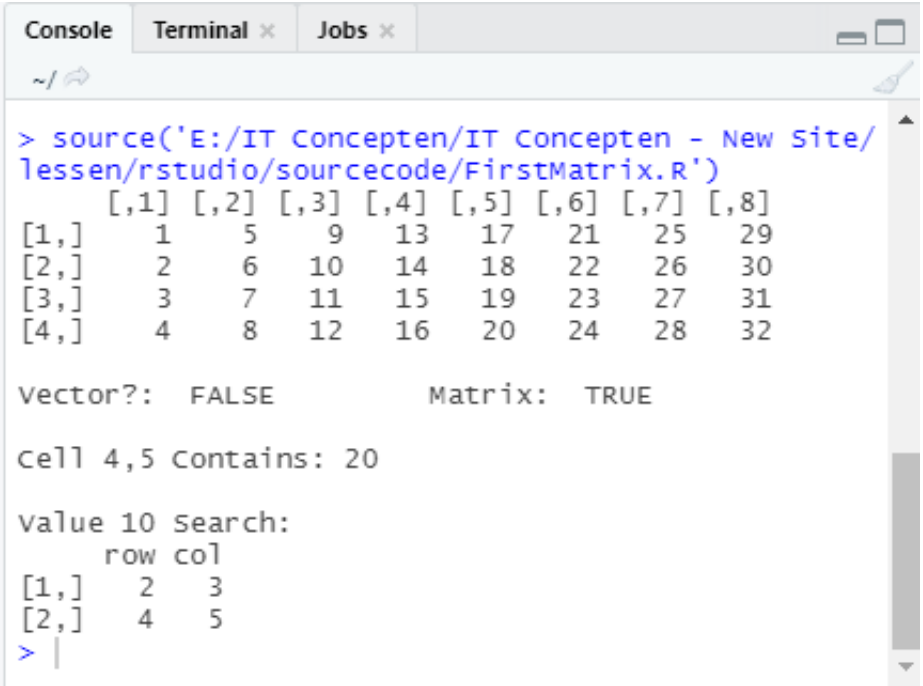
Source on Save | Run | Source

```
1  #Maak een vector met een reeks van 32 gehele waarden
2  data <- seq(1:32)
3
4  #Maak een matrix die de vectorwaarden in tabelvorm
5  #opslaat en voer de matrix vervolgens uit om de cellen te zien.
6  table <- matrix(data, nrow=4, ncol=8)
7  print(table)
8
9  #Bevestig het type structuur waarin de waarden zijn opgeslagen.
10 cat("\nVector?: ", is.vector(table),
11     "\tMatrix: ", is.matrix(table))
12
13 #Haal een celwaarde op en wijs vervolgens een nieuwe waarde toe
14 #aan die cel.
15 cat("\n\nCell 4,5 Contains:", table[4,5])
16 table[4,5] <- 10
17
18 #Zoek ten slotte in alle cellen naar een specifieke waarde en
19 #identificeer de locatie van cellen die die waarde wel bevatten.
20 cell <- which(table == 10, arr.ind=TRUE)
21 cat("\n\nValue 10 Search: \n")
22 print(cell)
```

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



Merk op dat de matrix is gemaakt met twee indices - eerst de rijen en daarna de kolommen.



Transposing data

Bij het maken van een matrix zal de functie **matrix()** standaard de gegevens die u invoert invoegen in cellen die zijn gerangschikt op kolomvolgorde. Dit betekent dat cellen in de eerste kolom worden gevuld met gegevens van elementen aan het begin van de opgegeven vector voordat de tweede kolom wordt gevuld, vervolgens de derde kolom, enzovoort.

Als je liever bepaalt hoe de cellen worden gevuld met gegevens, kunt u een **byrow=**-argument opnemen in de aanroep van de functie **matrix()**. Wanneer hieraan een **TRUE**-waarde wordt toegewezen, voegt de functie de door u verstrekte gegevens in cellen in, gerangschikt op rijvolgorde. Cellen in de eerste rij worden nu gevuld met gegevens van de elementen aan het begin van de opgegeven vector, voordat de tweede rij wordt gevuld, dan de derde rij, enzovoort. Als u een **FALSE**-waarde toewijst aan het argument **byrow=**, worden de gegevens die u invoert in cellen ingevoegd, gerangschikt op kolomvolgorde - de standaardvolgorde.

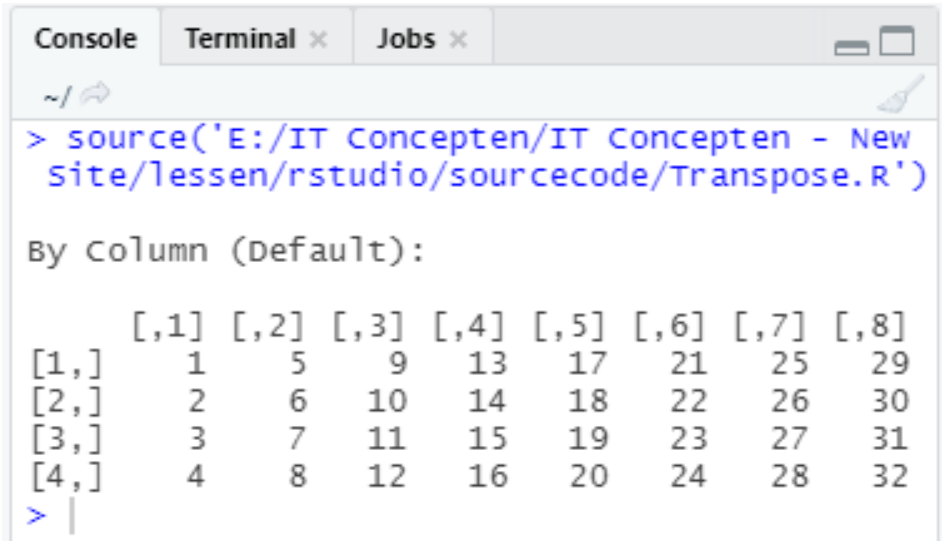
De rangschikking van cellen in een matrix kan ook worden getransponeerd, zodat de rijen kolommen worden en de kolommen rijen, simpelweg door de matrixnaam op te geven voor de ingebouwde **t()**-functie.

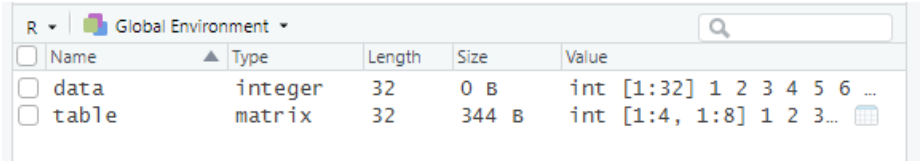
1 Begin een **R** Script door een vector te maken die een reeks van 32 gehele waarden bevat.

```
data <- seq(1:32)
```

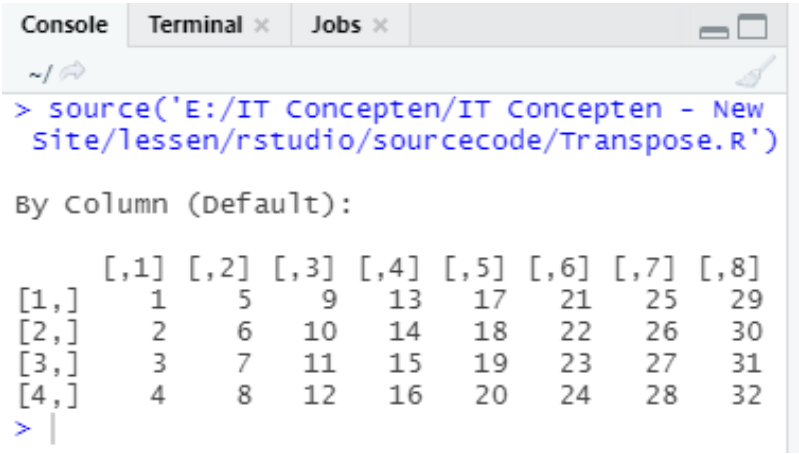
2 Maak vervolgens een matrix die de vectorwaarden in kolomvolgorde opslaat en voer vervolgens de matrix uit om de cellen te zien.

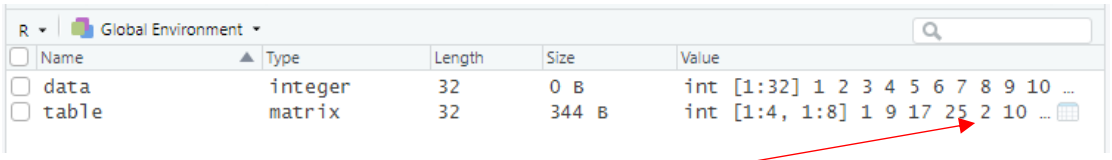
```
table <- matrix(data,nrow=4, ncol=8)
cat("\nBy Column (Default): \n\n")
print(table)
```





3 Maak nu de matrix opnieuw om de vectorwaarden in rijvolgorde op te slaan, en voer de matrix uit om de cellen te zien.

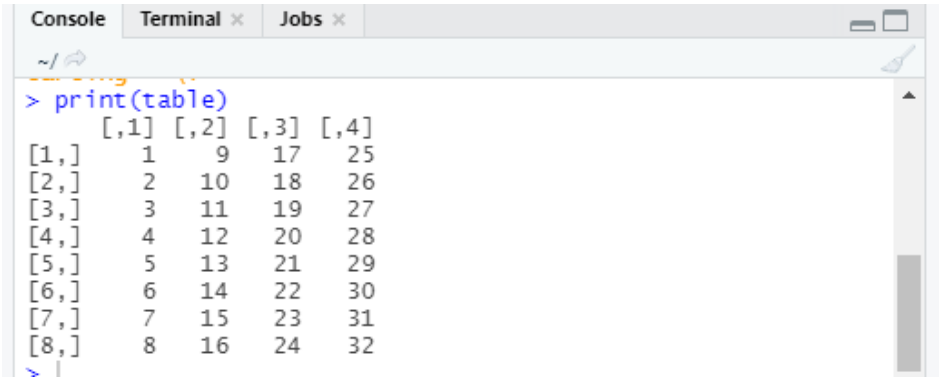


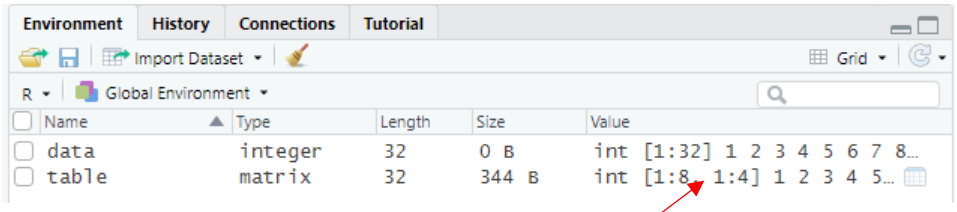


Merk op dat de cel gegevens worden herschikt wanneer ze per rij worden opgeslagen.

4 Transponeer ten slotte de matrix om rijen en kolommen uit te wisselen, en voer de matrix uit om de cellen te zien.

```
table <- t(table)
cat("\Transposed (Rows to Columns): \n\n")
print(table)
```





Merk op dat de matrixelementen worden herschikt wanneer het wordt getransponeerd.

5 Sla het **R Script** op met de naam **Transpose.R.**

Binding vectors

Wanneer je gegevens van meerdere vectoren in een enkele matrix moet combineren, biedt de R-programmeertaal twee mogelijkheden:

1. Een reeks vectornamen kan worden opgegeven als argumenten voor de ingebouwde functie **rbind()** om een matrix te maken die de gegevens van elke vector in *afzonderlijke rijen* zal bevatten.
2. Een reeks vectornamen kan worden opgegeven als argumenten voor de ingebouwde functie **cbind()** om een matrix te maken die de gegevens van elke vector in *afzonderlijke kolommen* zal bevatten.

De lengte van alle gespecificeerde vectoren moet identiek zijn, anders verschijnt er een waarschuwingsbericht. In dit geval wordt de matrix nog steeds gemaakt, maar worden de vectorwaarden gerecycleerd in extra cellen.

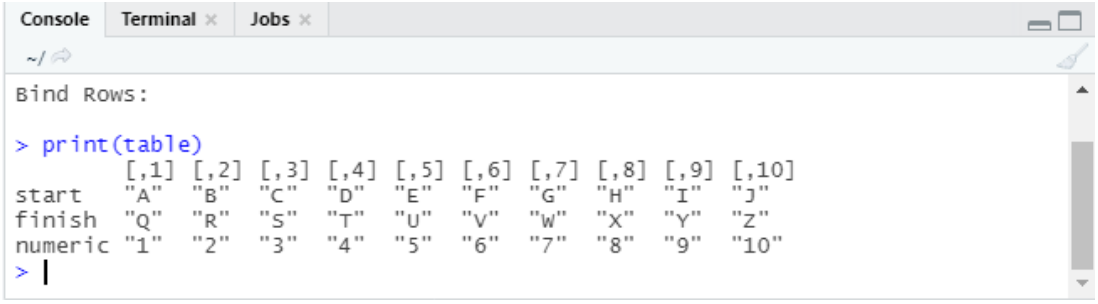
De gegevens in de te combineren vectoren moeten van hetzelfde gegevenstype zijn, anders kan het worden geconverteerd zodat de matrix cel gegevens van slechts één gegevenstype bevat.

1 Begin een **R** Script door drie vectoren van identieke lengte te maken, die verschillende gegevenstypen bevatten.

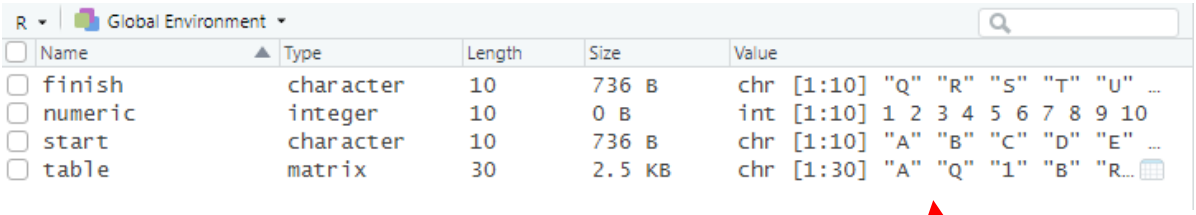
```
start <- LETTERS[1:10]
finish <- LETTERS[17:26]
numeric <- seq(1:10)
```

2 Maak vervolgens een matrix die de vectorwaarden op een afzonderlijke rij opslaat en voer de matrix vervolgens uit om de cellen te zien.

```
table <- rbind(start, finish, numeric)
cat("\nBind Rows: \n\n")
print(table)
```



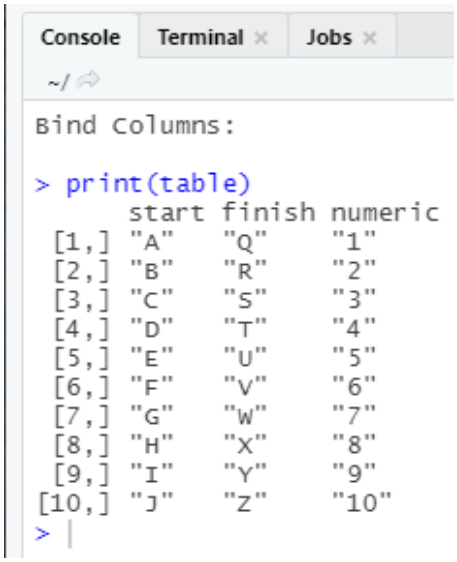
3 Open het tabblad Omgeving in het deelvenster Werkruimte om te zien dat de vectorgegevenstypen verschillen, maar zorg ervoor dat de matrix alleen waarden van het tekengegevenstype bevat.



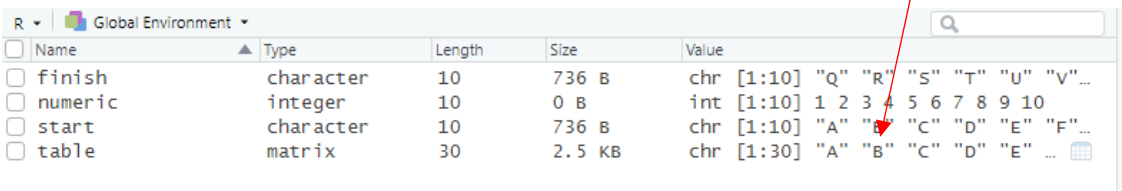
Maak nu een matrix die de vectorwaarden in afzonderlijke kolommen opslaat en voer de matrix vervolgens uit om de cellen te zien.

4

```
table <- cbind(start, finish, numeric)
cat("\nBind Columns: \n\n")
print(table)
```



Merk op dat de matrixelementen anders zijn gerangschikt bij het binden op rijen of op kolommen.



Het gegevenstype van elk object kan worden onderzocht door dat object op te geven als argument voor het ingebouwde **type()** van de functie. Met de bovenstaande matrix bevestigt **typeof(tabel[1,3])** bijvoorbeeld dat de cel de waarde "1" van het tekengegevenstype bevat.

Tekengegevens kunnen worden geconverteerd naar numerieke gegevens voor gebruik in een R-script door de ingebouwde **as.numeric()**-functie. **as.numeric(table[1,3])** converteert bijvoorbeeld naar het dubbele gegevenstype.

Naming rows and columns

De **R**-interpreter geeft automatisch rij-labels en kolomkoppen weer wanneer een matrix wordt uitgevoerd in het consolevenster. Deze kunnen eenvoudigweg het indexnummer van elke rij ([1,],[2,],[3,],etc.) en elke kolom ([,1],[,2],[,3], enz. aangeven.) als de functie **matrix()** is gebruikt om de matrix te maken.

Matrices die zijn gemaakt met de functie **rbind()** zullen bij uitvoer automatisch de naam van de vectorvariabele weergeven in plaats van het indexnummer voor elk rijlabel. Evenzo zullen matrices die zijn gemaakt met de functie **cbind()** automatisch de naam van de vectorvariabele weergeven in plaats van het indexnummer voor elke kolomkop wanneer ze worden uitgevoerd.

Betekenisvolle namen kunnen worden gegeven door de matrixnaam op te geven als argument voor de functies **rownames()** en **colnames()**. Deze kunnen vervolgens elk worden toegewezen aan een door komma's gescheiden lijst met namen door de functie **c()** voor rij-labels en kolomkoppen. Uiteraard moet de lengte van elke lijst overeenkomen met het aantal rijen en kolommen binnen de matrix.

Afzonderlijke rijen of kolommen kunnen worden geadresseerd met hun indexnummer of opgegeven naam. Als u gegevens kopieert van een matrix met benoemde rijen en kolommen naar een vector, worden de namen ook gekopieerd om een benoemde vector te maken met key=value-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, maar lees ze wel goed.

Name.R

```r
#Maak drie vectoren van identieke lengte, die elk gegevens
#van het dubbele gegevenstype bevatten.
ny <- c(3.8, 5.5, 9.9, 15.7, 21.5, 26.3)
la <- c(10.5, 19.4, 19.7, 20.8, 21.3, 22.7)
fw <- c(13.7,15.4, 20.0,24.6,28.5, 32.7)

#Maak een matrix die de vectorwaarden op afzonderlijke
#rijen opslaat en voer de matrix vervolgens uit om de
#cellen te zien.
table <- rbind(ny, la, fw)
print(table)

#Wijs betekenisvolle namen toe aan de rij-labels en
#kolomkoppen.
rownames(table) <- c("New York",
                     "Los Angeles",
                     "Forth Worth")

colnames(table) <- month.abb[1:6]

#Voer een teksregel en herziene matrix uit
cat("\nAverage High Temperature (°C):\n\n")
print(table)

#Maak een nieuwe vector om de gegevens van een enkele rij
#van de matrix op te slaan, met behulp van de rijnaam of
#het indexnummer.
nyc <- table["New York", ]        # Or table[1,]

#Geef vervolgens de gegevens weer die zijn opgeslagen in
#de nieuwe vectorvariabele
print(nyc)
```

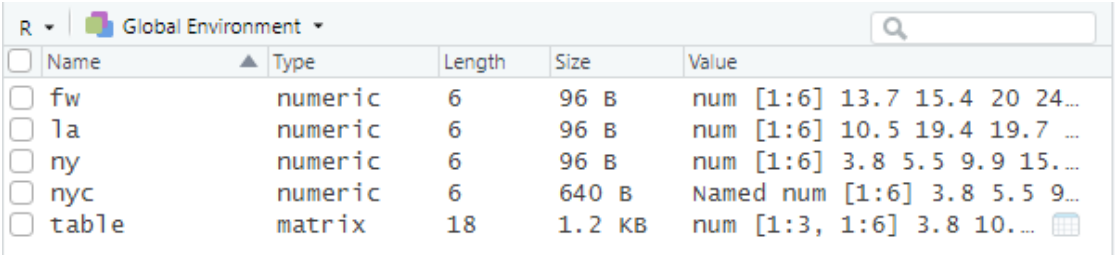
3 Sla het **R Script** op met de naam **Name.R** en klik vervolgens op de knop **Source** of druk op **Ctrl + Shift + S** om de rekenkundige uitvoer te zien.

Console | Terminal | Jobs

```
> source('E:/IT Concepten/IT Concepten - New Site/lessen/rstudio/so
urcecode/Name2.R')
   [,1] [,2] [,3] [,4] [,5] [,6]
ny  3.8  5.5  9.9 15.7 21.5 26.3
la 10.5 19.4 19.7 20.8 21.3 22.7
fw 13.7 15.4 20.0 24.6 28.5 32.7

Average High Temperature (°C):

             Jan  Feb  Mar  Apr  May  Jun
New York     3.8  5.5  9.9 15.7 21.5 26.3
Los Angeles 10.5 19.4 19.7 20.8 21.3 22.7
Forth Worth 13.7 15.4 20.0 24.6 28.5 32.7
 Jan  Feb  Mar  Apr  May  Jun
 3.8  5.5  9.9 15.7 21.5 26.3
>
```



Merk op hoe, in regel 19, de R-constante **month.abb** wordt gebruikt om namen van kolomkoppen op te geven.

Merk op hoe de functie **cat()**, in regel 22, hier alleen de opgeslagen gegevens op haalt, terwijl, in regel 23, de functie **print()** ook de kolomkoppen ophaalt.

De functie **names()** kan worden gebruikt om de namen binnen een benoemde vector op te halen, **names(nyc[1])** haalt in dit voorbeeld bijvoorbeeld "Jan" op, terwijl **nyc[1]** de waarde 3.8 ophaalt.

Plotting matrices

De programmeertaal R biedt een **matplot()**-functie waarmee u gemakkelijk grafische visualisaties kunt maken van gegevens die zich in een matrixstructuur bevinden. Deze functie vereist de matrixnaam als argument, plus verschillende andere argumenten om aan te geven hoe u de visualisatie wilt laten verschijnen:

- **type=** - het type plot dat moet worden getekend. Opties zijn onder meer "p" voor alleen punten, "l" voor alleen lijnen en "b" voor zowel plots als lijnen.
- **pch=** - het type plot dat moet worden getekend. Opties zijn numeriek gespecificeerd voor een of meer van de onderstaande symbolen:

Meerdere opties kunnen worden opgegeven als een door komma's gescheiden lijst met argumenten voor de functie **c()** of als een reeks met de **: dubbele punt**-operator (*colon*).

- **Col=** - de te gebruiken plotkleur. Opties worden gespecificeerd op naam of numeriek voor een of meer kleuren, en omvatten het onderstaande basispalet:

Color:	Name:	Code:	Hexadecimal:
	Black	1	#000000
	Red	2	#FF0000
	Green	3	#00FF00
	Blue	4	#0000FF
	Cyan	5	#00FFFF
	Magenta	6	#FF00FF
	Yellow	7	#FFFF00
	Gray	8	#BEBEBE

Hexadecimale kleurwaarden moeten worden opgegeven als tekenreeksen tussen aanhalingstekens.

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.

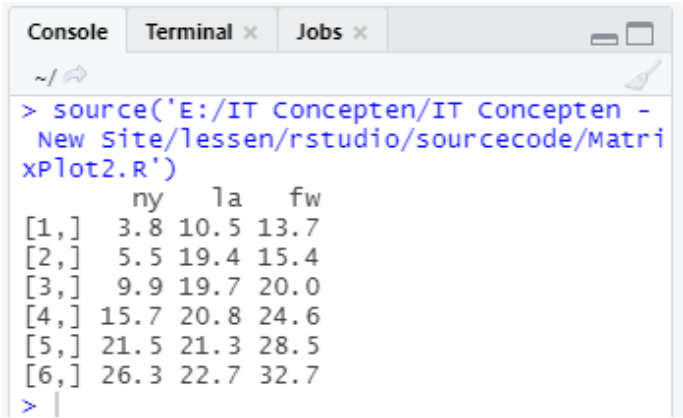
2 Voer de onderstaande broncode in de Code Editor in. Laat de groene commentaarregels weg, maar lees ze wel goed.

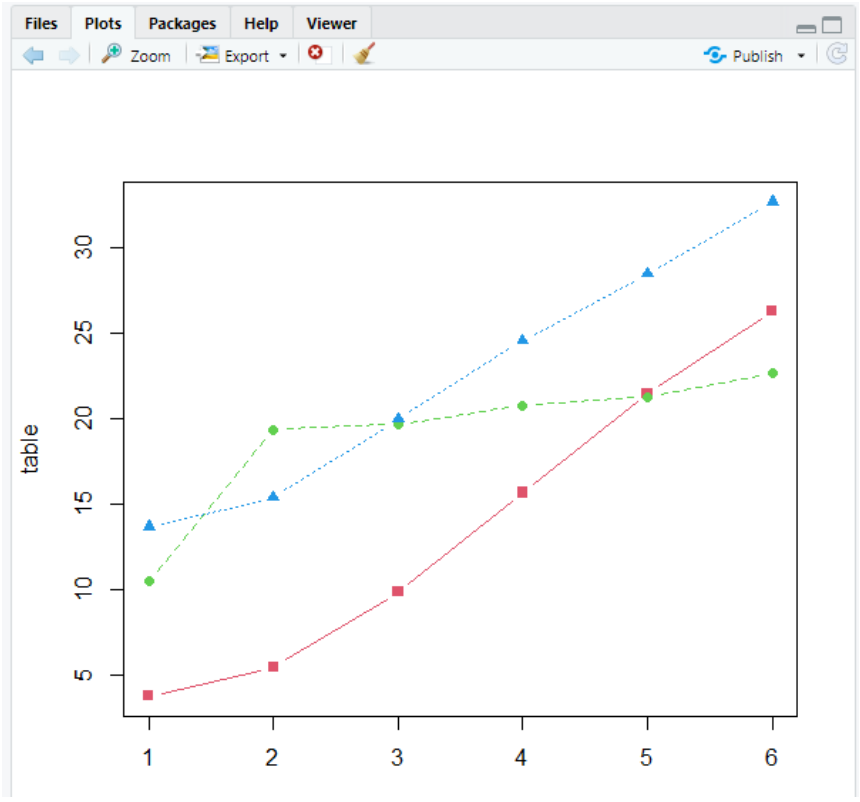
MatrixPlot2.R | MatrixPlot.R

Source on Save | Run | Source

```
1  #Maak drie vectoren van identieke lengte, die elk gegevens van
2  #het dubbele gegevenstype bevatten.
3  ny <- c(3.8, 5.5, 9.9, 15.7, 21.5, 26.3)
4  la <- c(10.5, 19.4, 19.7, 20.8, 21.3, 22.7)
5  fw <- c(13.7,15.4, 20.0,24.6,28.5, 32.7)
6
7  #Maak een matrix die de vectorwaarden op afzonderlijke
8  #rijen opslaat en voer de matrix vervolgens uit om de
9  #cellen te zien.
10 table <- cbind(ny, la, fw)
11 print(table)
12
13 #Voeg een instructie toe om een grafische visualisatie
14 #van de gegevens te maken - met punten en lijnen, drie
15 #verschillende plot-tekens te gebruiken en in drie
16 #verschillende kleuren te tekenen.
17 matplot(table, type="b", pch=15:17, col=2:4)
18
```

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





In regel 17 kunnen meerdere opties als alternatief worden gespecificeerd als **pch=c(15, 16, 17)** en **col=c(2, 3, 4)**

Adding labels

De R-programmeertaal **matplot()**-functie, geïntroduceerd in het vorige voorbeeld, kan verdere argumenten accepteren om labels voor de plot te specificeren en om het bereik langs elke as te regelen:

- **xlab=**, **ylab=** - titel voor de x-as en y-as.
- **xlim=**, **ylim=** titel voor de x-as en y-as.
- **main=**, - koptitel voor de plot

De R-interpreter zal plot-assen automatisch voorzien van gelabelde maatstreepjes, maar je kunt deze onderdrukken om je eigen assen te specificeren. De eerste vereist dat je een **axes=FALSE** argument opneemt in de **matplot()** functieaanroep om de automatische assen te onderdrukken, dan kun je de ingebouwde **axis()** functie gebruiken om elke vereiste as te specificeren.
Deze functie vereist een eerste geheel getal om aan te geven aan welke kant van de plot de as moet worden getekend:

- onder (1),
- links (2),
- boven (3) of
- rechts (4).

Als er geen verdere argumenten zijn opgenomen, worden het bereik en de labels automatisch toegevoegd. Om je eigen bereik en labels te leveren, is een **at=**-argument vereist om de punten te specificeren waarop maatstreepjes moeten worden getekend, en een **labels=**-argument om labelnamen op te geven. Labelnamen worden toegewezen als een vector van door komma's gescheiden tekenreeksen waarvan de lengte moet overeenkomen met het opgegeven vmaatstreepje. Meestal gebruikt dit de functies **rownames()** of **colnames()** om de matrix ro- of kolomnamen als aslabels toe te wijzen.

Legenden

Met de functie R-programmeertaal **legend()** kunt u eenvoudig een beschrijvende legenda aan een plot toevoegen. Deze functie vereist een eerste argument om een positie te specificeren waarop de legenda moet worden getekend.
Hiervoor kunnen speciale trefwoorden, zoals "*topleft*" worden gebruikt. De positie kan verder worden aangepast door een **inset=**-argument op te nemen om de legenda weg te houden van de plotmarges.
Het belangrijkste is dat de **legend()**-functie **pch=**- en **col=**-argumenten moet bevatten waarvan de toegewezen waarden precies moeten overeenkomen met die gespecificeerd voor de **matplot()**-functie voor

correcte identificatie van de punttekens en kleuren van de plot. Ten slotte moet de functie **legend()** een argument **legend=** bevatten om de plotcomponent te beschrijven door een vector van door komma's gescheiden tekenreeksen toe te wijzen. Meestal kan dit de functie **rownames()** of **colnames()** gebruiken om de matrixrij- of kolomnamen toe te wijzen.

1 Kopieer het vorige **MatrixPlot.R** scriptbestand en sla het op met de nieuwe naam **Label.R.** Voer de onderstaande aanpassingen aan de code uit.

2 Wijs rij- en kolomnamen toe voordat u de functie **print()** aanroept.

```
colnames(table) <- c("New York", "Los Angeles", "Forth Worth")
rownames(table) <- month.abb[1:6]
```

3 Wijzig de functieaanroep matplot() om argumenten toe te voegen

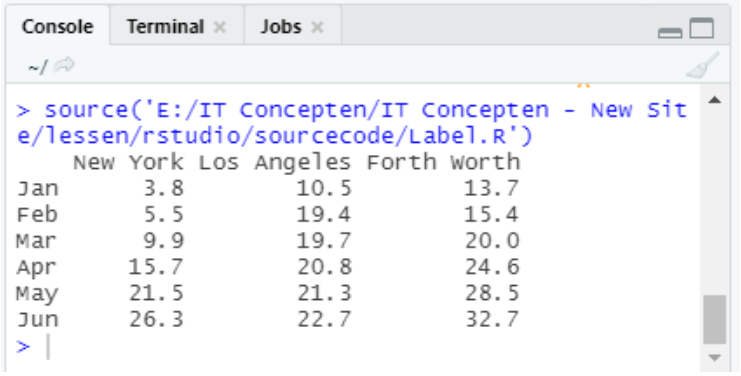
```
matplot(table, type="b", pch=15:17, col=2:4,
        xlab="Months", ylab="Average High (°C)",
        xlim=c(1,6), ylim=c(0, 35), axes=FALSE,
        main="City Temprature Comparison")
```

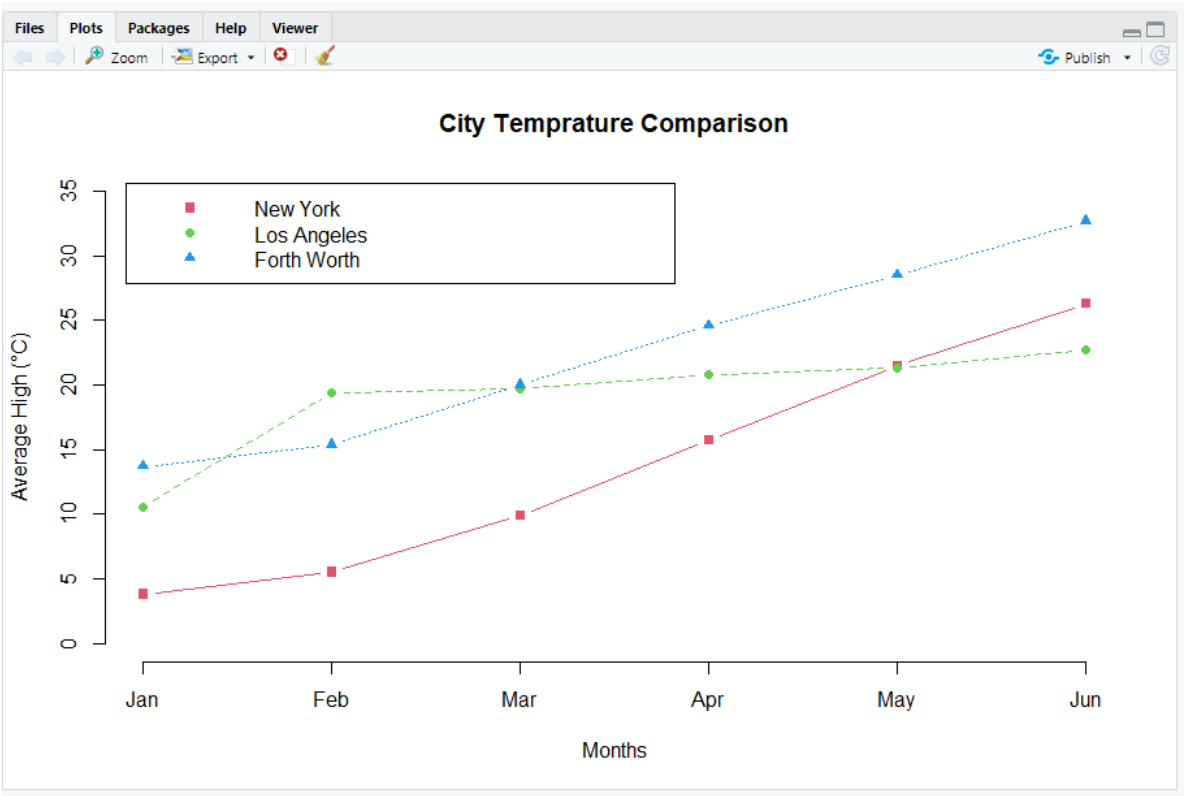
4 Maak as-labels en voeg een beschrijvende legenda toe

```
axis(1, at=1:6, labels=rownames(table))
axis(2)
legend("topleft", inset=0.02, pch=15:17, col=2:4,
       legend=colnames(table))
```

De waarde die is toegewezen aan het argument **inset=** is een fractie van het plotgebied, geen absolute lengtemaat.

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





Inzichten uit deze plot: Los Angeles heeft de meest constante hoge temperatuur, Forth Worth bereikt de temperatuur van Los Angeles in maart, maar New York bereikt de temperatuur van Los Angeles pas in mei.

Extracting matrix subsets

Een "*subset*" *(subgroep)* is gewoon een groep gegevenswaarden die deel uitmaken van een andere grotere set gegevenswaarden. Bij R-programmering is het vaak handig om een *subset* te extraheren voor vergelijking van specifieke interessegebieden. Met vectorvariabelen kan een *subset* worden geëxtraheerd door de indexnummers van specifieke elementen op te geven, zoals deze:

alphabet <- [LETTERS[]
vowel.subset <- alphabet[1, 5, 9,15, 21]

Als alternatief kan met benoemde vectorvariabelen een subset worden geëxtraheerd door de indexnamen van specifieke elementen op te geven:

nato <- c(A=”Alpha”, B=”Bravo”, C=”Charlie”, D=”Delta”)
abc.subset <- nato[c(A, B, C)]

Merk op dat de functie c() tussen vierkante haken moet worden opgenomen om de indexnamen van te extraheren elementen te specificeren.

Subsets die uit een vector worden geëxtraheerd, hebben één dimensie en worden geretourneerd als een vectorgegevensstructuur.
Met matrices kan een subset worden geëxtraheerd door de indexnummers van specifieke cellen op te geven, zoals deze:

```
table <- matrix(1:60, nrow=12, ncol=5)
table.subset <- table[1:3, 1:5]
```

De rijen en kolommen van een matrix kunnen een naam krijgen met de functies **rownames()** en **colnames()** of door een **dimnames=**-argument op te nemen in de aanroep van de functie **matrix()**. Dit argument vereist een lijst met lengte één om alleen de rijen een naam te geven, of een lijst met lengte twee om zowel rijen als kolommen een naam te geven, zoals deze:

```
table <- matrix(1:60, nrow=12, ncol=5,
          dimnames=list(month.abb[], LETTERS[1:5]))
```

Een subset kan vervolgens worden geëxtraheerd door de rij- en kolomnamen van specifieke cellen op te geven, zoals deze:

```
table.subset <- table[month.abb[1:3],LETTERS[1:5]]
```

Wanneer een hele rij of kolom moet worden geëxtraheerd, kan het indexnummer of de naam volledig worden weggelaten uit de vierkante haken. Bovenstaande opdrachten kunnen bijvoorbeeld als volgt worden gemaakt:

```
table.subset <- table[1:3, ]
table.subset <- table[c("Jan", "Feb", "Mar"), ]
```

Een komma is nog steeds vereist bij het weglaten van een indexnaam of nummer tussen vierkante haken.

Geëxtraheerde subsets die meer dan één dimensie bevatten (d.w.z. meer dan één rij of kolom) worden geretourneerd als een matrixgegevensstructuur.

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, maar lees ze wel goed.

Subset2.R | Subset.R

Source on Save | Run | Source

```
1  #Maak drie vectoren van identieke lengte, die elk gegevens van
2  #het dubbele gegevenstype bevatten.
3  ny <- c(3.8, 5.5, 9.9, 15.7, 21.5, 26.3)
4  la <- c(10.5, 19.4, 19.7, 20.8, 21.3, 22.7)
5  fw <- c(13.7,15.4, 20.0,24.6,28.5, 32.7)
6
7  #Maak een matrix die de vectorwaarden in afzonderlijke kolommen
8  #opslaat en geef de rijen en kolommen een naam.
9  table <- cbind(ny, la, fw)
10 rownames(table) <- month.name[1:6]
11 colnames(table) <- c("New York",
12                        "Los Angeles", "Forth Worth")
13
14 #Voeg statements toe om de hele matrix uit te voeren
15 cat("\nMatrix...\n")
16 print(table)
17
18 #Maak een subset die alleen waarden opslaat uit de eerste drie
19 #rijen van slechts twee kolommen van de matrix.
20 table.q1 <- table[1:3, c(1, 3)]
21
22 #Voeg instructies toe om de volledige subset uit te voeren.
23 cat("\nSubset...\n")
24 print(table.q1)
25
```

In regel 20 van de broncode wordt de operator **:** dubbele punt *(colon)* gebruikt om rijen **1-TO-3** te selecteren en wordt de operator **,** komma gebruikt in de functie **c()** om kolommen **1-AND-3** te selecteren.

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

Maintaining dimensions

Bij het extraheren van een subset uit een matrix is het belangrijk om het type gegevensstructuur te herkennen waarin de gegevens worden geretourneerd - standaard gedrag is misschien niet wat u nodig heeft!

Als de subset meer dan één dimensie heeft, wordt de zonsondergang geretourneerd in een matrixgegevensstructuur, maar als de subset slechts één dimensie heeft, wordt de subset standaard altijd geretourneerd in een vectorgegevensstructuur. Dit komt omdat R probeert te anticiperen op uw vereisten door automatisch dimensies te laten vallen (verwijderen) die het beschouwt als overbodige informatie.

Het standaardgedrag is verstandig, omdat u over het algemeen wilt dat gegevens uit enkele rijen of kolommen van een matrix als vector worden geretourneerd.

Waar de subset gegevens uit een enkele rij haalt, wordt de dimensie van de rij naam verwijderd en worden de kolomnamen gebruikt als elementnamen in de geretourneerde vector. Omgekeerd, waar de subset gegevens extraheert uit een enkele kolom, wordt de dimensie van de kolomnaam verwijderd en worden de rijnamen gebruikt als elementnamen in de geretourneerde vector.

Het standaardgedrag kan worden overschreven door een laatste argument **drop=FALSE** op te nemen tussen de **[ ]** vierkante haken die de rijen of kolommen specificeren die moeten worden geëxtraheerd. Dit betekent dat subsets die gegevens uit een enkele rij of uit een enkele kolom halen, nu worden geretourneerd in een matrixgegevensstructuur.

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, maar lees ze wel goed.

Dimension.R

Source on Save | Run | Source

```
1  #Maak een vector met een numerieke reeks.
2  data <- 1:28
3
4  #Maak een matrix die de vectorwaarden in
5  #rijen opslaat en noem de rijen en kolommen alfabetisch.
6  table <- matrix(data, nrow=4, ncol=7, byrow=TRUE,
7                  dimnames=list(letters[1:4], LETTERS[1:7]))
8
9  #Voeg instructies toe om de hele matrix uit te voeren.
10 cat("\nMatrix...\n")
11 print(table)
12
13 #Maak een subset die gegevens extraheert uit een enkele rij
14 #van de matrix - met behulp van het standaardgedrag.
15 tier <- table[2, ]
16
17 #Voeg instructies toe om de subset uit te voeren.
18 cat("\nSubset...\n\nRow #2 (Default)...\n")
19 print(tier)
20
21 #Voeg instructies toe om de gegevensstructuur van de subset
22 #te identificeren.
23 cat("\nMatrix?: ", is.matrix(tier))
24 cat("\tVector?: ", is.vector(tier),"\n")
25
```

3 Sla het **R Script** op met de naam **Dimension.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/Dimension2.
R')

Matrix...
   A  B  C  D  E  F  G
a  1  2  3  4  5  6  7
b  8  9 10 11 12 13 14
c 15 16 17 18 19 20 21
d 22 23 24 25 26 27 28

Subset...

Row #2 (Default)...
 A  B  C  D  E  F  G
 8  9 10 11 12 13 14

Matrix?:  FALSE Vector?:  TRUE
>
```

Door het argument **drop=FALSE** toe te voegen, zorgt u ervoor dat gegevens altijd worden geretourneerd in dezelfde klasse van objecten als die waaruit ze zijn opgehaald.