Storing values *(Waarden opslaan)*

Een enkele waarde opslaan

Bij R-programmering is een "variabele" een container waarin een waarde kan worden opgeslagen voor later gebruik door het programma.

Een variabele wordt gemaakt in R Script door een unieke identificatienaam naar keuze in de Code-Editor te schrijven en vervolgens een initiële waarde toe te wijzen die binnen de variabele moet worden opgeslagen.

De waarde kan worden toegewezen aan een variabele in R-programmering met behulp van de <- toewijzingsoperator. Als u bijvoorbeeld een nummer wilt toewijzen aan een variabele met de naam "dozen", gaat u als volgt te werk:

dozen <- 12

Variabele namen worden gekozen door de programmeur, maar moeten voldoen aan bepaalde naamgevingsconventies. De naam van de variabele mag alleen beginnen met een letter, of een punt gevolgd door een letter, en mag vervolgens alleen letters, cijfers, punten of onderstrepingstekens bevatten. Namen zijn hoofdlettergevoelig, dus "var" en "Var" zijn duidelijk verschillende namen en spaties zijn niet toegestaan in namen.

Variabele namen moeten ook de gereserveerde woorden vermijden, die in de onderstaande tabel worden vermeld, omdat deze een speciale betekenis hebben in de **R**-taal.

if	**else**	**repeat**	**while**
function	for	in	next
break	TRUE	FALSE	NULL
Inf	NaN	NA	NA_integer
NA_real	NA_complex	NA_character	return

Voer op elk gewenst moment de opdracht **?reserved** in de console in om de lijst met gereserveerde woorden te zien verschijnen op het tabblad Help in het **Notebook-venster**.

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. Typ in de Code Editor **name** als de naam van de variabele.

3. Typ <- of druk op **Alt + -** om de toewijzingsoperator toe te voegen.

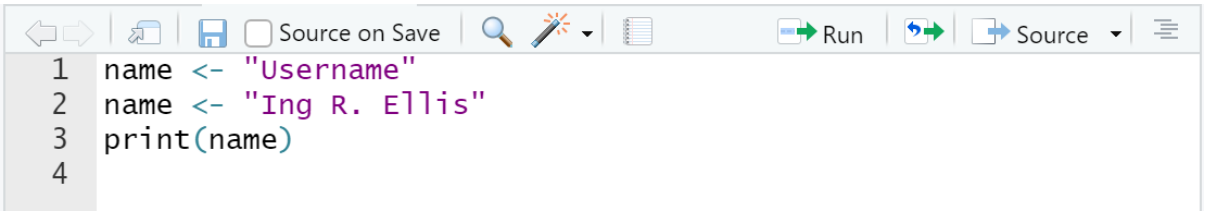
4. Druk vervolgens op de toets " om twee dubbele aanhalingstekens toe te voegen en typ vervolgens **Username** tussen de aanhalingstekens.

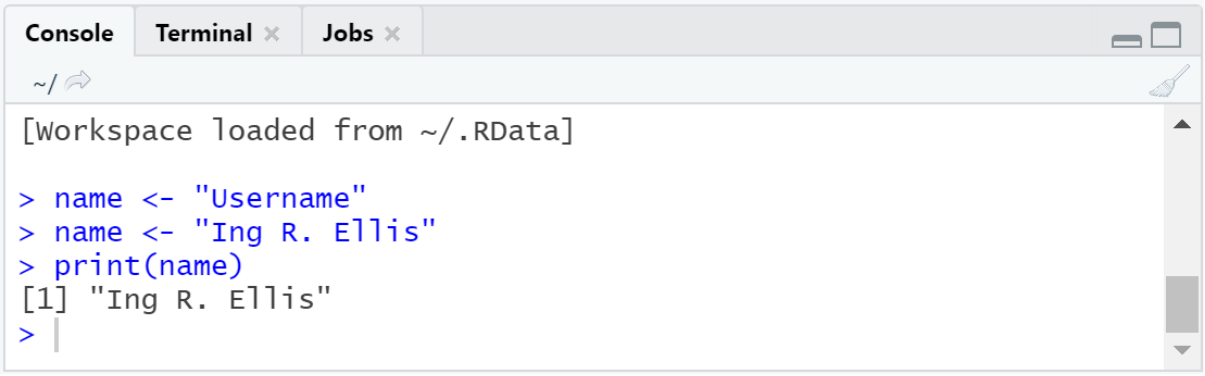
5. Zorg ervoor dat de cursor op dezelfde regel staat als de code en klik vervolgens op **Run** of druk op **Ctrl + Enter** - zie de variabele en de waarde ervan verschijnt nu op het tabblad Environment.

6. Terug in de code-editor, ga naar de volgende regel en schrijf **name** <- ""

7. Plaats je eigen naam tussen de aanhalingstekens en klik vervolgens op **Run** om een nieuwe waarde aan de variabele toe te wijzen - zie de waardeverandering direct op het tabblad Omgeving.

8 Ga naar de volgende regel en schrijf **print(name)**, klik vervolgens op **Run** om de variabele waarde in de console uit te voeren.





8 Klik op **File** in het **RStudio** menu. Kies de optie **Save as...** Ge naar waar je het bestand wil opslaan en sla de **R Script** op met de naam **EersteVariabele.R**.

Commentaar toevoegen

In R Script-programmering kunnen opmerkingen worden toegevoegd door een regel te beginnen met het # hekje. Alle volgende tekens op die regel worden volledig genegeerd door R-interpreter. Er is geen optie voor opmerkingen met meerdere regels in R Script.

Als de R Script met anderen wordt gedeeld, is het een goed idee om de code te documenteren door een koptekst (header) opmerking op te maken. Dit moet details bevatten zoals:

- De naam van het script
- De datum waarop het script gemaakt is
- De auteur(s) van het script
- Het doel van het script
- De geschiedenis van de herzieningen van het script

1 Begin in RStudio Code Editor een R-script door regels met koptekstinformatie te typen.

```
Script name:    Comment.R
Created on:     September 21, 2021
Author:         ing. R. Ellis
Purpose:        Echo user input
Version:        1.0
Execution:      Must be run as source to wait user input
```

2 Sleep de cursor over de hele koptekst om deze te selecteren en druk vervolgens op **Ctrl + Shift + C** om alle geselecteerde regels te verwijderen.

3 Voeg vervolgens een opmerking en instructie toe om gebruikersinvoer te vragen

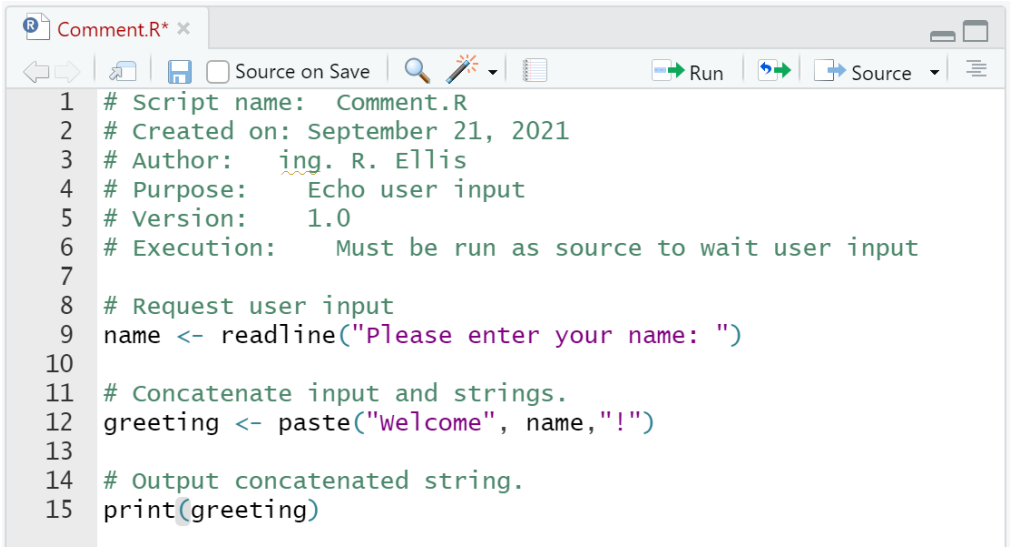
```
# Request user input
name <- readline("Please enter your name: ")
```

4 Voeg nu een opmerking en instructie toe om de gebruikersinvoer in een tekenreeks te plakken

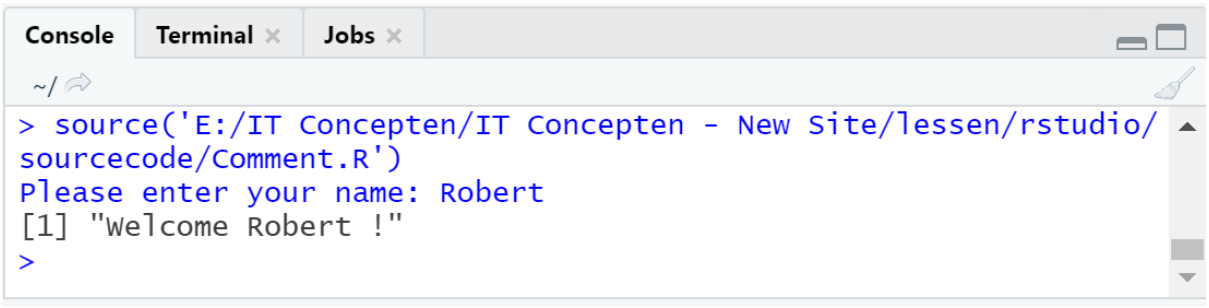
```
# Concatenate input and strings.
greeting <- paste("Welcome", name,"!")
```

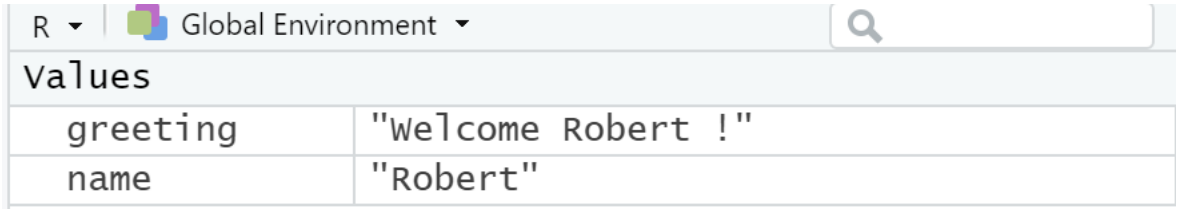
5 Voeg ten slotte een opmerking en instructie toe om de hele string af te drukken

```
# Output concatenated string.
print(greeting)
```



6 Volg de header-instructie, klik op de source knop in de Code Editor of druk op **Ctrl + Shift + S** om het script uit te voeren en voer vervolgens invoer in wanneer daarom wordt gevraagd.





7 Klik op **File** in het **RStudio** menu. Kies de optie **Save as...** Ge naar waar je het bestand wil opslaan en sla de **R Script** op met de naam **Comentaar.R**.

Gegevenstypen herkennen

Variabelen in **R** kunnen verschillende typen bevatten. De meest gebruikte datatypes van variabelen in R-programmering staan in de onderstaande tabel, samen met een korte beschrijving:

Data type:	Omschrijving:	Voorbeeld:
Character	Een tekstteken of tekenreeks	"R" "R string
Double	Een decimaal getal	3.14
Integer	Een geheel getal	5
Boolean	Een logische waarde	TRUE

R bepaalt automatisch het variabele gegevenstype op basis van de waarde die het bevat. Het gegevenstype van een variabele kan worden onthuld door de naam ervan op te geven als argument voor de ingebouwde functie **typeof()**.

Het is belangrijk te beseffen dat numerieke variabelen standaard altijd worden gemaakt als een dubbel gegevenstype, tenzij een toegewezen geheel getal wordt achtervoegt door een letter **L**. Bijvoorbeeld, **number**

= 5L creëert een integer gegevenstype, maar **number = 5** creëert een dubbel gegevenstype.

R biedt verschillende ingebouwde functies om het gegevenstype van een variabele te testen. De naam van de variabele kan worden opgegeven als argument voor de functie **is.character()**, die de Booleaanse waarde **TRUE** of **FALSE** retourneert, afhankelijk van het gegevenstype van de variabele. Er zijn ook **is.double()**, **is.integer()** en **is.logical()** functies die op een vergelijkbare manier kunnen worden gebruikt om het gegevenstype van een variabele te testen.

Booleaanse waarden kunnen aan een variabele worden toegewezen met de trefwoorden **TRUE** of **FALSE**, of gewoon door de letters **T** en **F** te gebruiken.

1. Open de RStudio Code Editor en maak een variabele die een string waarde bevat

```
title <- "R for Data Analysis"
```

2. Een tekenreeks en gegevenstype toewijzen aan een tweede variabele

```
result <- paste("Type of title:", typeof(title))
```

3. Voer de gecombineerde tekenreeks uit om het gegevenstype van de variabele te zien

```
print(result)
```

4. Maak vervolgens een variabele met een dubbele waarde en een variabele met een geheel getal.

```
pi <- 3.14159265
dozen <- 12L
```

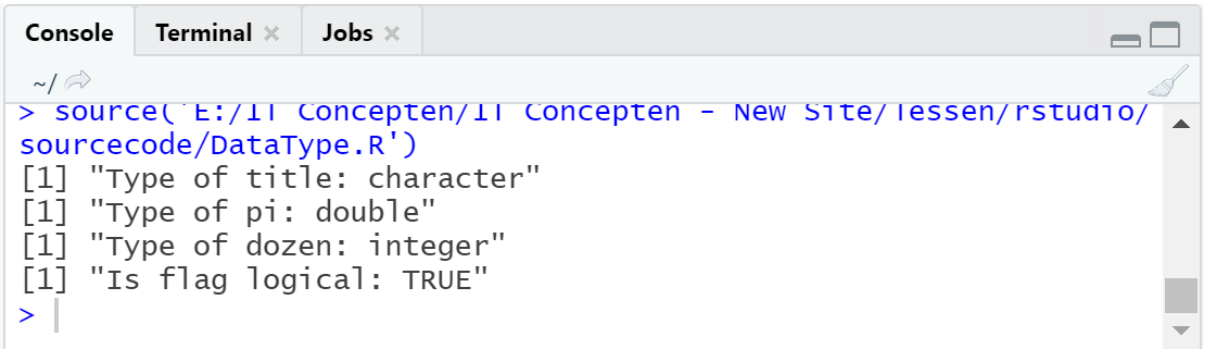
5. Voer het gegevenstype van elke variabele uit in de vorige stap.

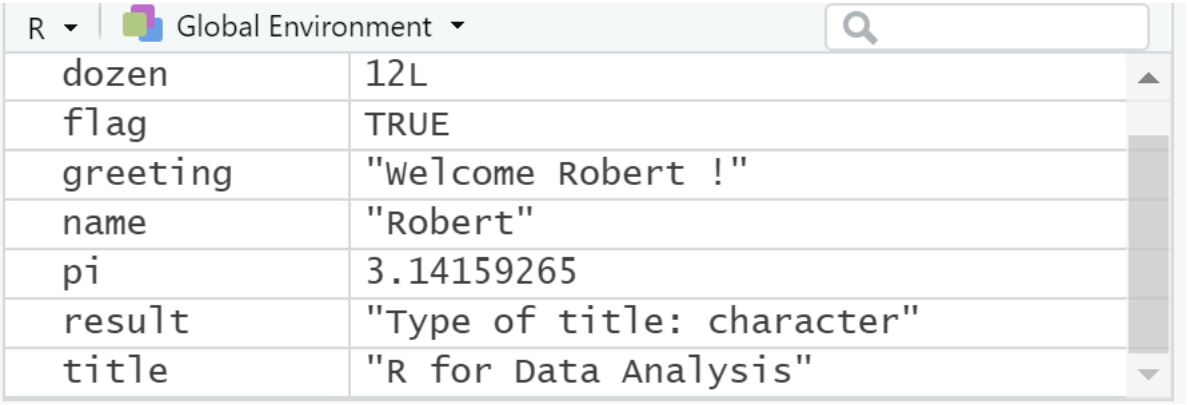
```
print(paste("Type of pi:", typeof(pi)))
print(paste("Type of dozen:", typeof(dozen)))
```

6. Maak nu een variabele met een logische waarde en voer het resultaat uit van een gegevenstypetest op deze variabele.

```
flag <- T
print(paste("Is flag logical:", is.logical(flag)))
```

7 klik op de  source knop in de Code Editor of druk op **Ctrl + Shift + S** om het script uit te voeren





8 Sla het **R-script** op als een bestand met de naam **DataType.R** in de **huidige werkmap**.

Meerdere waarden opslaan

Omdat de programmeertaal R is ontworpen om gegevenssets te verwerken, is een variabele eigenlijk een "vector" die meerdere waarden kan bevatten. Elke waarde bevindt zich in een "element" van de vector.

Meerdere waarden worden toegewezen aan een variabele met behulp van de ingebouwde combineerfunctie **c()** die een door komma's gescheiden lijst met waarden accepteert die aan de vectorelementen moeten worden toegewezen. Als u bijvoorbeeld drie waarden wilt toewijzen met **month = c("Jan", "Feb", "Mar")**.

Vectoren in **R** zijn ingesloten vanaf één, dus de eerste waarde die in de vector is opgeslagen, bevindt zich in element één, de tweede waarde bevindt zich in element twee, enzovoort. In code worden de vectorelementen geadresseerd door het gewenste indexnummer tussen een paar **[ ]** vierkante haken, *brackets,* achter de variabel naam te

plaatsen. Bijvoorbeeld, zou **month[1]** de waarde ophalen die is opgenomen in het eerste element van de variabele **month** - in dit geval de tekenreekswaarde "Jan".

Nieuwe waarden kunnen worden toegewezen aan afzonderlijke elementen met behulp van de variabelenaam en het indexnummer. Om bijvoorbeeld de waarde in het derde element te vervangen door **month[3] = "March"**.

De lengte van een vector kan worden gevonden door de variabele naam op te geven als argument voor de ingebouwde functie **length()**. Bijvoorbeeld, **lengte(month)** zou een lengte van drie elementen tonen.

Vectoren zijn flexibel en kunnen dus automatisch uitbreiden wanneer een waarde wordt toegewezen aan een indexnummer dat groter is dan de huidige lengte van de vector. Bijvoorbeeld, **month[4] = "Apr"** zou automatisch de vector uitbreiden en **length(month)** zou nu een lengte van vier elementen tonen.

Het is belangrijk om te erkennen dat elke vector alleen waarden van hetzelfde gegevenstype kan bevatten. Als je een mengsel van integers (gehele getallen) en doubles (decimale getallen) toewijst, zullen alle elementen doubles bevatten (geconverteerde gehele getallen). Als u een combinatie van cijfers en tekens toewijst, zullen alle elementen tekens bevatten (geconverteerd cijfers). De ingebouwde functie **typeof()** kan worden gebruikt om het gegevenstype van alle elementen vast te stellen.

R biedt verschillende andere structuren waarin naast de vector gegevens kunnen worden opgeslagen. De naam van het object kan worden opgegeven als het argument voor de functie **is.vector()**, die een Booleaanse waarde van **TRUE** of **FALSE** retourneert, afhankelijk van of het object inderdaad een vector is of niet.

1. Open RStudio Code Editor en maak een variabele die meerdere tekst waarden bevat.
   alphabet <- c("Alpha", "Bravo", "Charlie")

2. Toon de volledige inhoud van alle elementen van de variabele uit.
   print (alphabet)

3. Toon een tekst string en de waarde in één element.
   print(paste("2nd Element: ",alphabet[2]))

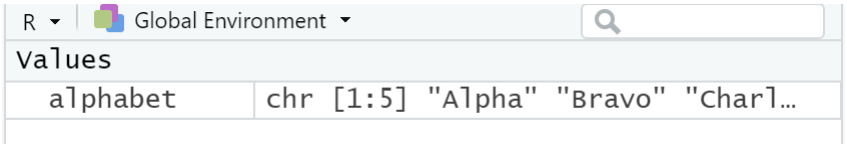
4. Toon een string uit en het aantal elementen in de vector.
   print(paste("Vector Length: ", length(alphabet)))

5 Wijs een andere waarde toe om de vector uit te breiden en voer vervolgens de volledige inhoud en lengte nogmaals uit.

```
alphabet[5] <- "Echo"
print(alphabet)
print(paste("Vektor Length Now: ", length(alphabet)))
```

6 Sla het **R-script** op als een bestand met de naam **DataType.R** in de **huidige werkmap**.

7 klik op de  source knop in de Code Editor of druk op **Ctrl + Shift + S** om het script uit te voeren



Gemengde gegevenstypen opslaan

Omdat items van verschillende datatypes niet kunnen worden opgeslagen in een enkele vector, biedt R een bruikbare alternatieve "lijst"-structuur, waarvan de elementen elk waarden van elk datatype kunnen bevatten.

Lijsten worden geïndexeerd vanaf één, net als vectoren, en waarden worden toegewezen aan lijstelementen door ze op te geven als een door komma's gescheiden lijst met argumenten voor de ingebouwde functie **list()**. U kunt bijvoorbeeld een lijst maken met waarden van elk hoofdgegevenstype, als volgt:

data <- list(12,13.14,"Robert", TRUE)

De lijstlengte en het structuurtype kunnen worden onthuld met behulp van de functies **length()** en **typeof()** zoals bij vectoren, en er is een **is.list()**-functie om vast te stellen of een object een lijst is.

Net als vectoren kunt u elk afzonderlijk lijstelement adresseren door het indexnummer op te geven tussen [ ] vierkante haken. Bijvoorbeeld **data[3]** om de string op te halen in de lijst die hierboven is gemaakt. In tegenstelling tot vectoren zijn lijsten niet flexibel, wat betekent dat u geen waarde kunt toekennen aan een indexnummer dat groter is dan de huidige lengte van de lijst. U kunt echter de functie **c()** gebruiken om een bestaande lijst te combineren met extra waarden, of een andere lijst, om de lijst langer te maken.

Het belangrijkste is dat u optioneel elk element in een lijst een naam kunt geven door **key=value**-paren op te geven als een door komma's gescheiden lijst met argumenten voor de ingebouwde **list()-**functie, zoals deze:

Data <- list(dozen=12, pi=3.14, user="Robert", flag=TRUE)

Met een benoemd element kunt u de waarde ophalen door de lijstnaam en de elementnaam op te geven, gescheiden door de **$** dollar-operator. Bijvoorbeeld **data$user** om de string in de bovenstaande lijst op te halen.

R biedt twee ingebouwde functies speciaal voor lijsten die **key=value**-paren bevatten. De functie **names()** haalt alle sleutels op in de volgorde waarin ze in de lijst voorkomen. De functie **unlist()** retourneert een vector van alle sleutels en waarden in volgorde, maar de namen kunnen expliciet worden genegeerd door een **use.names=FALSE** argument op te nemen.

De functie **sum()** kan worden gebruikt om de numerieke waarden in een vector bij elkaar op te tellen, en de functie **mean()** kan worden gebruikt om een gemiddelde te berekenen van de numerieke waarden in een vector.

1 Open RStudio Code Editor en maak een list welke meerdere key=value-paren bevat.

```
sales <- list(Jan=1500, Feb=1300, Mar=2400)
```

2 Combineer de lijst met een extra **key=value pair list** om de lengte van de originele lijst uit te breiden en voer vervolgens alle paren uit.

```
sales <- c(sales, list(Apr=1800))
print(unlist(sales))
```

3 Wijs de lijstwaarden alleen toe aan een vectorvariabele

```
monthly.sales <- unlist(sales, use.names=FALSE)
```

4 Wijs vervolgens het berekende totaal van de lijstwaarden toe aan een variabele en voer vervolgens de totale waarde uit.

```
total.sales <- sum(monthly.sales)
print(paste("Total Sales: ", total.sales))
```

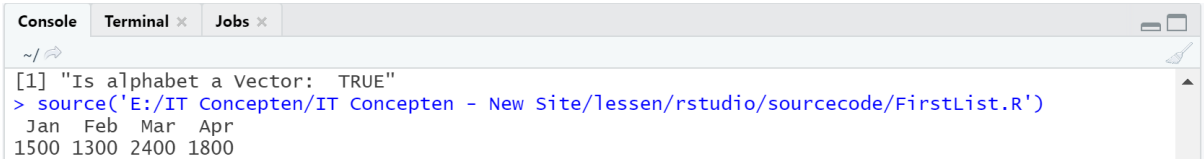
5 Wijs nu het berekende gemiddelde van de lijstwaarden toe aan een variabele en voer vervolgens de gemiddelde waarde uit.

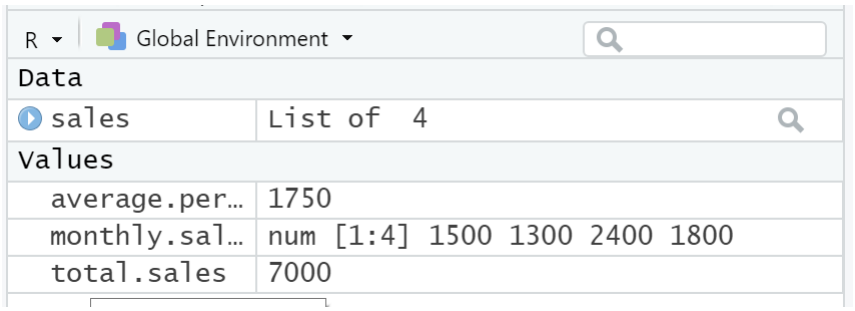
```
average.per.month <- mean(monthly.sales)
print(paste("Monthly Average: ", average.per.month))
```

6 Voer het resultaat van een gegevenstypetest uit in de lijst.

```
print(paste("Is sales a List: ", is.list(sales)))
```

7 klik op de  source knop in de Code Editor of druk op **Ctrl + Shift + S** om het script uit te voeren.





8 Sla het **R-script** op als een bestand met de naam **EersteLijst.R** in de **huidige werkmap**.

Opgeslagen waarden plotten

De kracht van **R**-programmering ligt in het vermogen om gemakkelijk grafische afbeeldingen te geven van de gegevens die zijn opgeslagen in **R** Script-structuren. U kunt een vectorargument opgeven voor de ingebouwde functie **plot()** om een spreidingsplot te maken die de gegevensomvang versus de index weergeeft. Meer typisch kunt u twee vectorargumenten opgeven voor de functie **plot()** die moet worden weergegeven op de **X-** en **Y-assen** van de plot, en een derde argument **type="o"** kan worden opgenomen om punten en lijnen te over plotten:

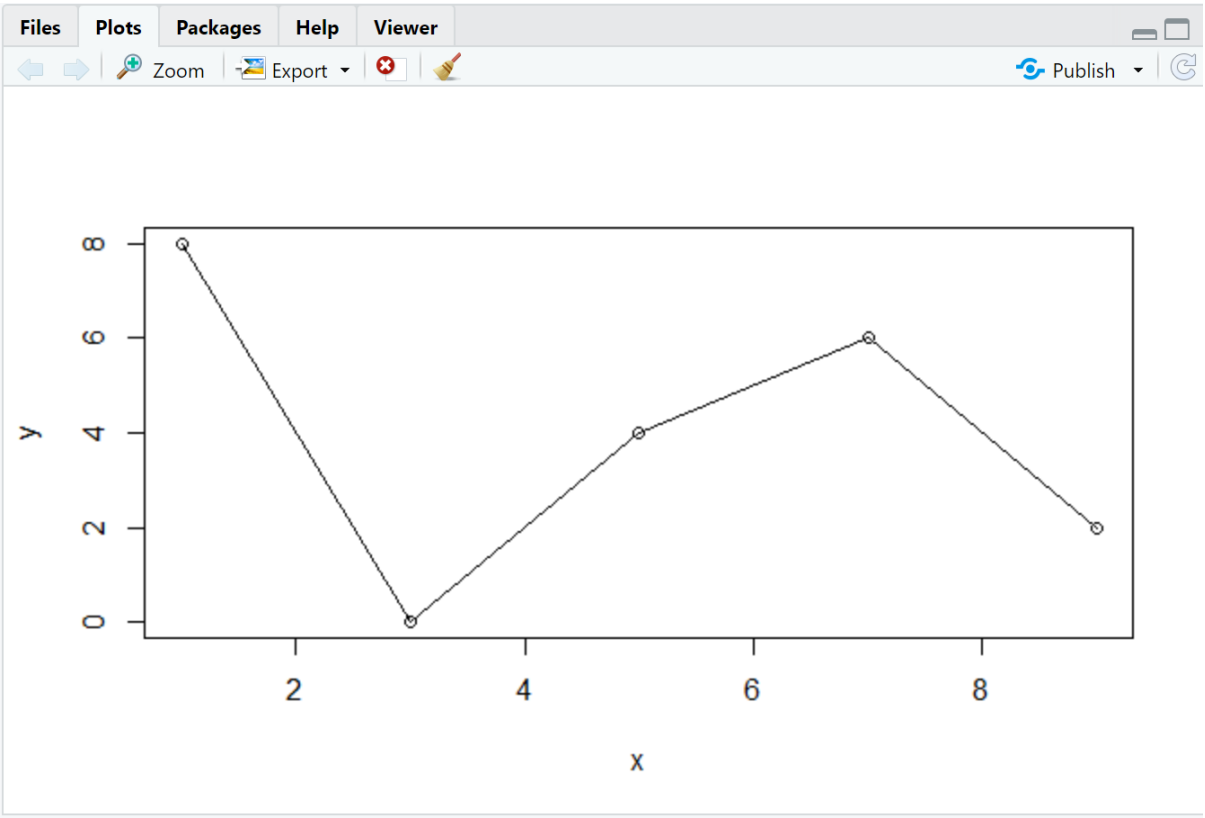
1. Open de RStudio Code Editor en maak twee vectoren.

```
x <- c(1, 3, 5, 7, 9)
y <- c(8, 0, 4, 6, 2)
```

2. Voeg vervolgens een instructie toe om de vectorwaarden weer te geven en selecteer vervolgens alle drie de regels in de code-editor.

```
plot(x, y, type="o")
```

3. klik op de  source knop in de Code Editor of druk op **Ctrl + Shift + S** om het script uit te voeren - zie een grafiek verschijnen op het tabblad Plots.



Andere mogelijke waarden voor het type-argument zijn

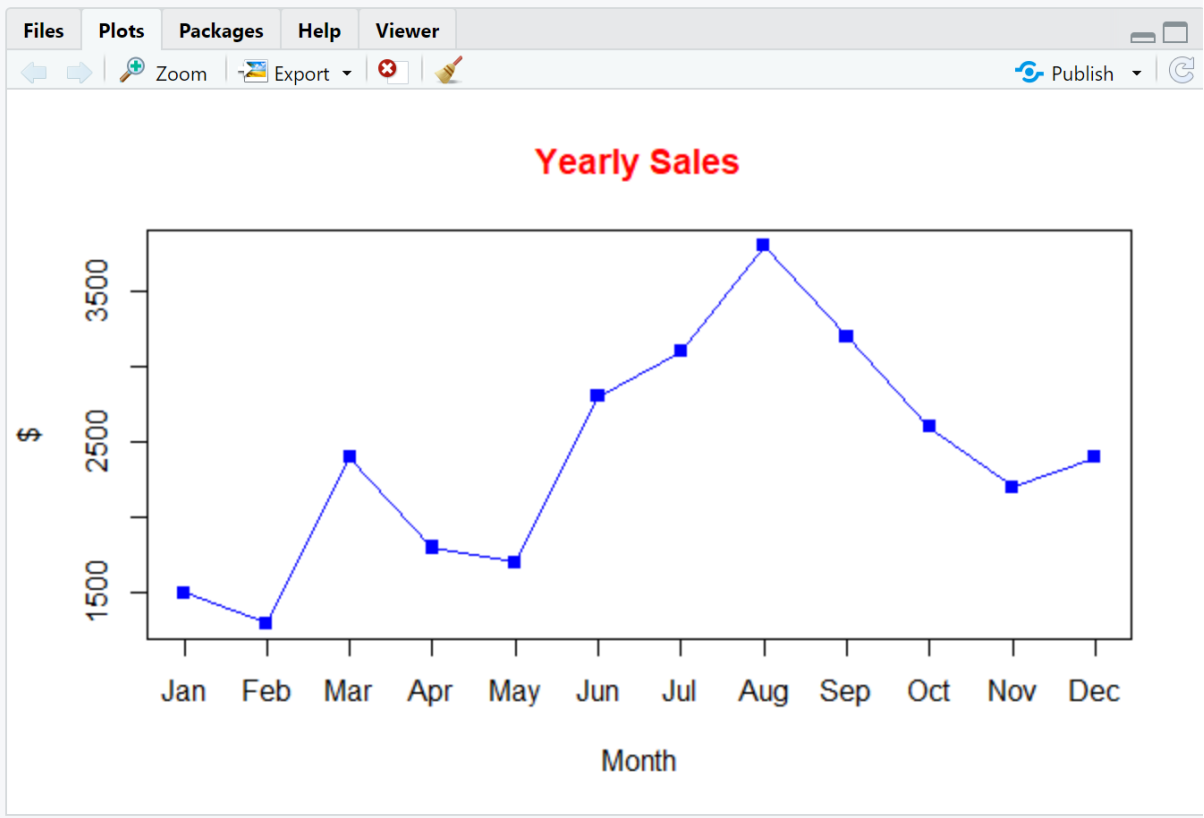
- alleen "p"-punten,
- alleen "l"-lijnen,
- "b" zowel punten als lijnen,
- "S"-stappen,
- "h" histogramachtige verticale lijnen.

De grafiek toont annotaties op basis van het bereik van de vectorwaarden, en de assen labels zijn gewoon de namen van de variabele-ID's. Maar u kunt dit beter doen door de controle over annotaties, aslabels, titel, puntteken en kleuren te nemen. Verdere argumenten kunnen aan de functie **plot()** worden toegevoegd om de lijnkleur en het puntteken te specificeren, en de ingebouwde functies **title()** en **axis()** kunnen worden gebruikt om de hoofdtitel, annotatie en aslabels te specificeren.

1 Open RStudio Code Editor en neem de onderstaande broncode over. Neem het commentaar bij de regels code goed door.

```
qtr.1 <- list(Jan=1500, Feb=1300, Mar=2400)
qtr.2 <- list(Apr=1800, May=1700, Jun=2800)
qtr.3 <- list(Jul=3100, Aug=3800, Sep=3200)
qtr.4 <- list(Oct=2600, Nov=2200, Dec=2400)

#Combineer de vier bovenstaande lijsten
year <- unlist(c(qtr.1, qtr.2, qtr.3, qtr.4))

#Plot de vector die type, kleur en puntkarakter specificeert, en
#schakel automatische annotatie en as-labels uit
plot(year, type="o", col="Blue", pch=15, ann=FALSE, axes=FALSE)

#Geef het bereik en de annotatie voor de X-as op,
#maar laat R automatisch de Y-as annoteren.
axis(1, at=1:12, lab=c(names(year)))
axis(2)

#Voeg labels toe voor elke as en een grafisch hoofdlabel en
#teken vervolgens een kader rond de grafiek.
title(xlab="Month", ylab="$", main="Yearly Sales", col.main="Red")
box()
```

2 klik op de  source knop in de Code Editor of druk op **Ctrl + Shift + S** om het script uit te voeren - zie een grafiek verschijnen op het tabblad Plots.



3 Experimenteer met het puntteken door een numerieke waarde op te geven in het bereik 0-25 in het **pch**-argument.

4 Sla het **R Script** op met de naam **CustomPlot.R**

Objecten besturen

Wanneer je **R**-code uitvoert om gegevens in een willekeurige structuur op te slaan, zoals een variabele of lijst, wordt een data structure object gemaakt in de **RStudio**-omgeving. Deze objecten worden weergegeven op het tabblad **Environment** in het **Workspace** deelvenster in één van de twee mogelijke weergaven: **List view** of **Grid view**. Grote datastructuren worden ingeklapt om ruimte te besparen in de lijstweergave, maar u kunt ze uitvouwen om hun inhoud weer te geven. In de **Grid view** kunnen grote datastructuren worden onderzocht door een uitgebreide lijst te maken in het deelvenster Code-editor.

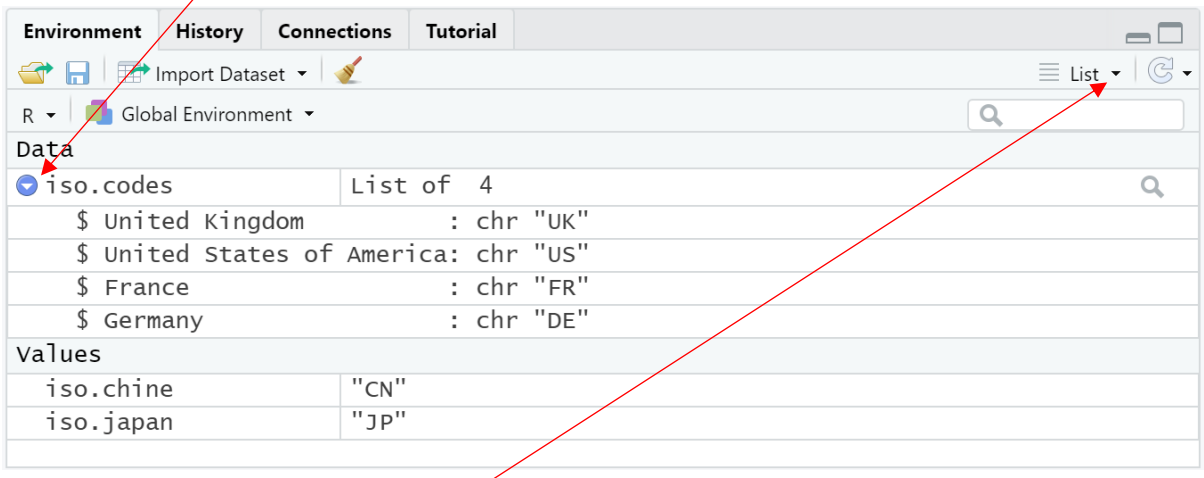
Je kunt de ingebouwde functie **ls()** aanroepen om alle objecten in de huidige omgeving in de console weer te geven. Individuele objecten kunnen uit de omgeving worden verwijderd door hun naam op te geven

als een door komma's gescheiden argument voor de ingebouwde **rm()**-functie, of alle objecten kunnen worden verwijderd door een **list=ls()**-argument op te geven.

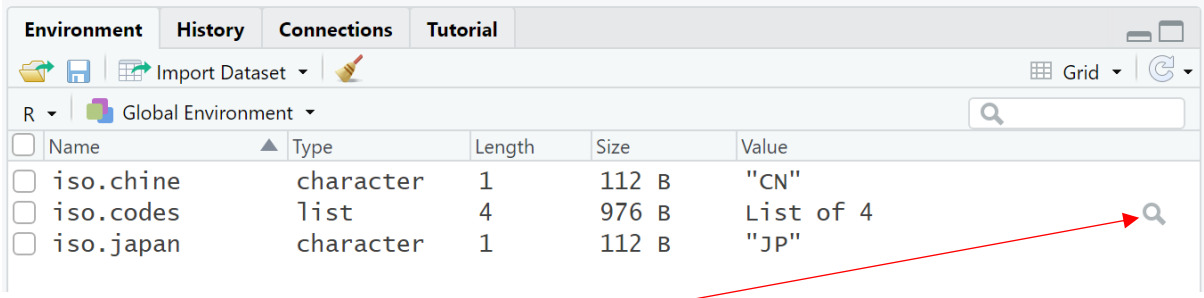
1. Open RStudio Code Editor en maak een lijst en twee variabelen.

2. Druk vervolgens op **Ctrl + A** om alle code te selecteren en druk vervolgens op **Ctrl + Enter** om de code uit te voeren en de objecten te maken.

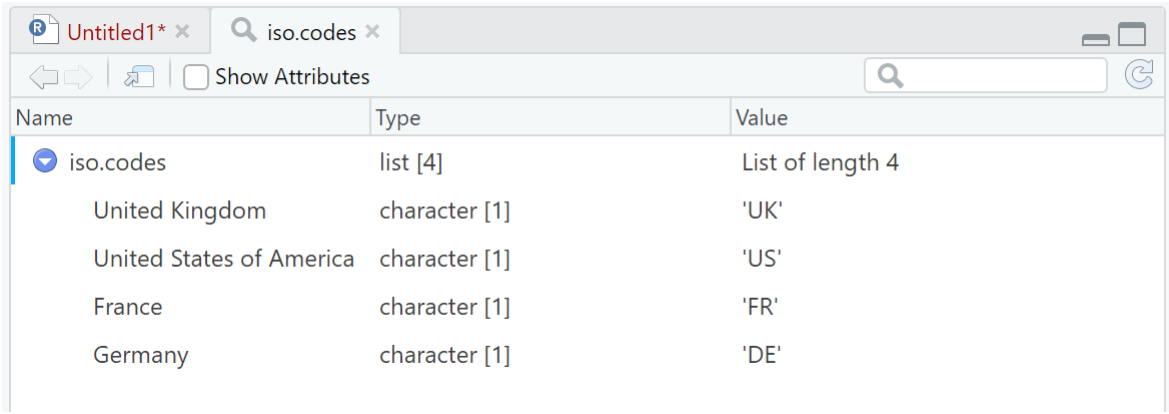
3. Open het tabblad **Environment** in de **List view** en klik vervolgens op de knop om het lijstobject uit te vouwen en de **key=value pairs** te zien.



4. Klik nu op de pijlknop op de menubalk van de tabbladen en schakel over naar de **Grid view**.



5. Klik op het vergrootglas pictogram naast de lijst in de **Grid view** om een uitgevouwen lijst te maken in het deelvenster Code-editor.



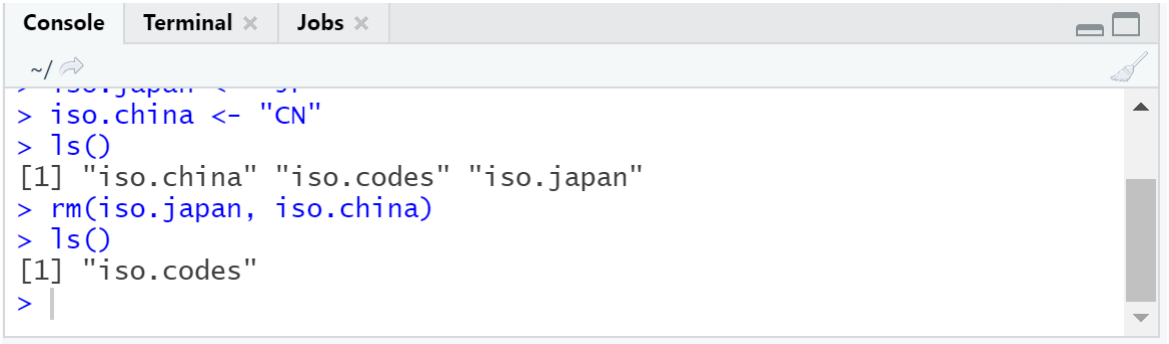
6 Keer terug naar de Code Editor en voer een instructie in om de huidige omgevingsobjecten weer te geven.

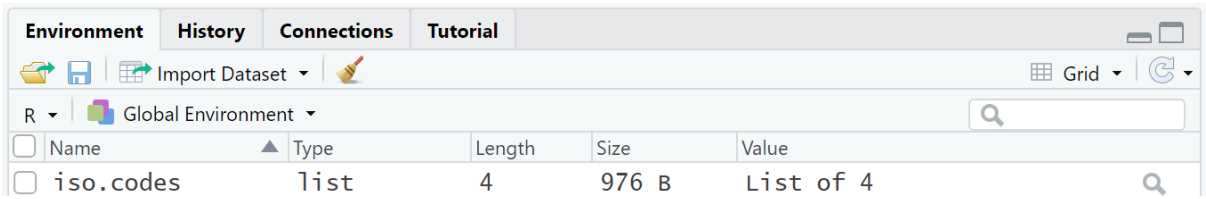
```
ls()
```

7 Voeg een instructie toe om beide variabele objecten te verwijderen en toon vervolgens de huidige omgevingsobjecten (environment objects) om de verwijdering te bevestigen.

```
rm(iso.japan, iso.china)
ls()
```

8 Selecteer de drie instructies en klik vervolgens op de knop Uitvoeren of druk op **Ctrl + Enter** om de code uit te voeren.





9 Sla het **R Script** op met de naam **Environment.R**