Regressieanalyse en regressiebomen

In dit hoofdstuk bespreken we regressieanalyse, en de aan regressieanalyse verwante regressiebomen (*regression trees*).

Het belang van regressieanalyse voor het analyseren van gegevens is niet te onderschatten. Het is één van de meestgebruikte technieken in statistische analyse van gegevens. Het principe van eenvoudige (lineaire) regressie staat aan de basis van een uitgebreide 'familie' van technieken die geschikt zijn voor een enorme hoeveelheid aan praktische statistische uitdagingen. In deze les beperken we ons tot de basis.

Eenvoudige regressieanalyse veronderstelt een lineair verband tussen een 'te verklaren' variabele en één of meer 'verklarende' numerieke variabelen (variabelen die getalsmatig kunnen worden gemeten). Maar er zijn veel varianten van regressieanalyse die het mogelijk maken de techniek toe te passen op situaties die niet voldoen aan dit ideaalbeeld (bijvoorbeeld niet-lineaire verbanden, en/of categoriale te verklaren en/of verklarende variabelen). Voor wie de basisprincipes van eenvoudige regressieanalyse onder de knie heeft, gaat een wereld aan analysetechnieken open!

Waar regressieanalyse uitgaat van een door de analist/onderzoeker gespecificeerd model, zijn regressiebomen meer beschrijvend en explorerend. Rregressiebomen hebben als voordeel dat ze gemakkelijker te visualiseren en te interpreteren zijn, en dat ze niet aan de strikte veronderstellingen van regressieanalyse hoeven te voldoen.

Regressieanalyse: het algoritme

Analisten denken vaak in relaties die in wiskundige modellen kunnen worden uitgedrukt.

Dit geldt ook voor tal van bedrijfskundige vragen.

> *Voorbeelden:*
>
> *De vraag naar veel producten varieert in de tijd. Deze variatie is deels afhankelijk van allerlei toevallige factoren, maar deels ook van factoren die we kennen, en soms zelf kunnen beïnvloeden.*
>
> *In wiskundige termen is de vraag naar het product de te verklaren variabele, en is er een diversiteit aan omgevingsfactoren (zoals het weer, de stand van de economie en de prijs van het product) die de vraag verklaren. We spreken dan ook van verklarende variabelen. In wiskundige termen willen we bijvoorbeeld weten, hoeveel meer ijs wordt er geconsumeerd als de temperatuur met een graad stijgt.*

Om die laatste vraag te beantwoorden hebben we (historische) gegevens nodig. Een ijsverkoper kan gedurende de zomermaanden de gemiddelde dagtemperatuur bijhouden, en het aantal verkochte ijsjes. Als hij dat doet voor een steekproef van 20 dagen, dan zouden zijn data er als volgt kunnen uitzien:

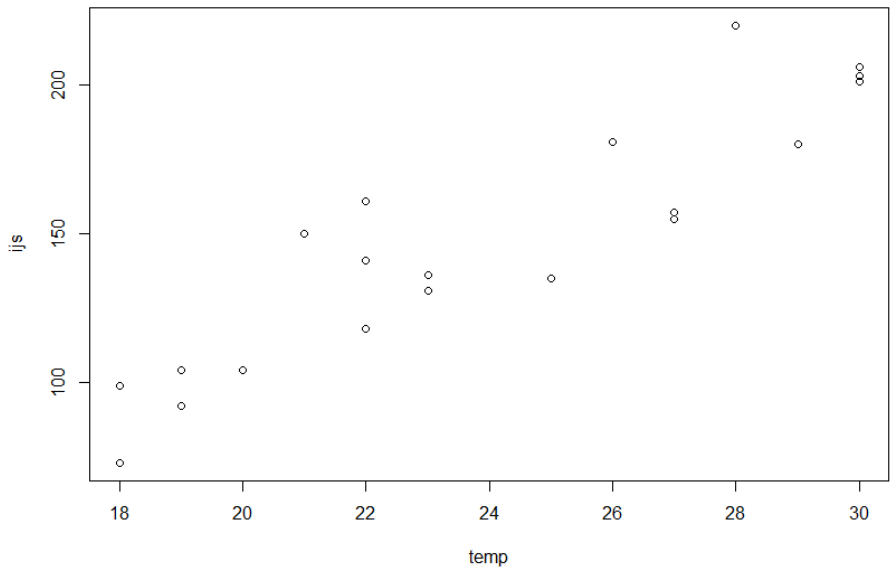
	A	B	C
1	ijs	temp	evenement
2	99	18	3
3	73	18	2
4	92	19	3
5	104	19	4
6	104	20	3
7	150	21	5
8	161	22	5
9	141	22	3
10	118	22	3
11	131	23	2
12	136	23	3
13	135	25	1
14	181	26	4
15	157	27	1
16	155	27	2
17	220	28	5
18	180	29	2
19	206	30	3
20	203	30	2
21	201	30	2

Figuur 1. Voorbeelddata. *Bron: Excel/ OGN.*

Omdat de data zijn gesorteerd op oplopende temperatuur, zien we gemakkelijk dat er een positieve relatie lijkt te zijn: hoe warmer het is, des te meer ijsjes worden er verkocht.

Echter, de temperatuur is geen perfecte voorspeller van de ijsverkoop. Immers, voor de drie dagen waarop het 30 graden was, loopt de ijsverkoop uiteen van 201 tot 206. De ijsverkoper denkt dat deze variatie deels toevallig is, maar ook te maken heeft met de drukte in zijn stad. Voor elk van de 20 dagen heeft hij vastgelegd, op een 5-puntsschaal, hoeveel evenementen er in de stad waren die zorgden voor extra drukte (1=heel weinig evenementen; 5=heel veel evenementen, of koopzondag).

Als we ons eerst beperken tot de relatie tussen ijsverkoop en temperatuur, dan kunnen we de relatie tussen de “te verklaren” variabele (ijsverkoop) en de “verklarende” variabele (temperatuur) grafisch weergeven:



Figuur 2: Spreidingsdiagram van ijsverkopen naar temperatuur. *Bron: OGN.*

Er is sprake van een positief verband: hoe hoger de temperatuur, des te meer ijsjes er worden verkocht.

Het is gebruikelijk de "te verklaren" variabele (vaak aangeduid als de y-variabele) op de verticale (y-)as weer te geven, en de "verklarende" (x-)variabele op de horizontale (x-)as.

Vaak veronderstellen we dat het verband tussen twee variabelen (bij benadering) lineair is. Ons doel is dan een rechte lijn te trekken in de grafiek die zo goed mogelijk aansluit bij de waarnemingen. We spreken dan van "lineaire regressie".

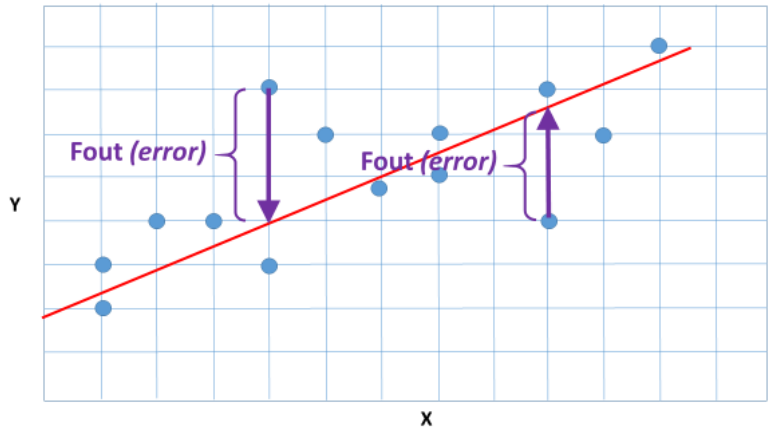
> *Wat betekent "regressie"? De term "regressie" is gebaseerd op het fenomeen dat variabelen die extreme waarden vertonen in een eerste meting, minder extreme waarden hebben in een tweede meting. Als bijvoorbeeld studenten een test maken met alleen ja-neevragen, en het antwoord op veel vragen wordt gegokt, dan zal de gemiddelde score rond de 50% liggen. Sommige studenten zullen een veel lagere score hebben, en andere een veel hogere. Als we nog een keer zo'n test doen, dan is het waarschijnlijk dat de studenten die eerder hoog scoorden, nu lager (dichterbij het gemiddelde) scoren.*
>
> *Galton (1822-1911) noemde dit verschijnsel "regression toward the mean". In één van zijn onderzoeken concludeerde hij dat extreme lange vaders, zonen hadden die (i) kleiner waren dan hun vader, maar (ii) langer dan gemiddeld; en het omgekeerde gold voor extreem kleine vaders.*

In de bovenstaande grafiek is het positieve verband onmiskenbaar. U zou "op het oog" een rechte opgaande lijn kunnen trekken. Maar dat is nog niet zo gemakkelijk! Als we meer analisten zo'n lijn laten trekken dan krijgen we lijnen die allemaal net iets anders zijn. We hebben daarom een eenduidig criterium nodig om die lijn te bepalen.

Het criterium dat we gebruiken is gebaseerd op de *errors* (de fouten die we maken in de voorspelling: het verschil tussen de werkelijke waarde en de voorspelling). Een lijn die zo goed mogelijk aansluit bij de werkelijke waarde, minimaliseert de fouten. Omdat sommige fouten positief en andere negatief zijn, kwadrateren we de fouten zodat ze altijd positief zijn.

In principe zouden we in plaats van gekwadrateerde fouten, de absolute waarden van de fouten kunnen nemen. Kwadrateren heeft twee voordelen ten opzichte van absolute waarden. Het eerste

praktische voordeel is dat het wiskundig gemakkelijker is om met gekwadrateerde waarden te werken. Een tweede voordeel is dat bij kwadrateren grote fouten zwaarder wegen.



Figuur 3: De lineaire regressielijn, en afwijkingen (fouten). *Bron: OGN.*

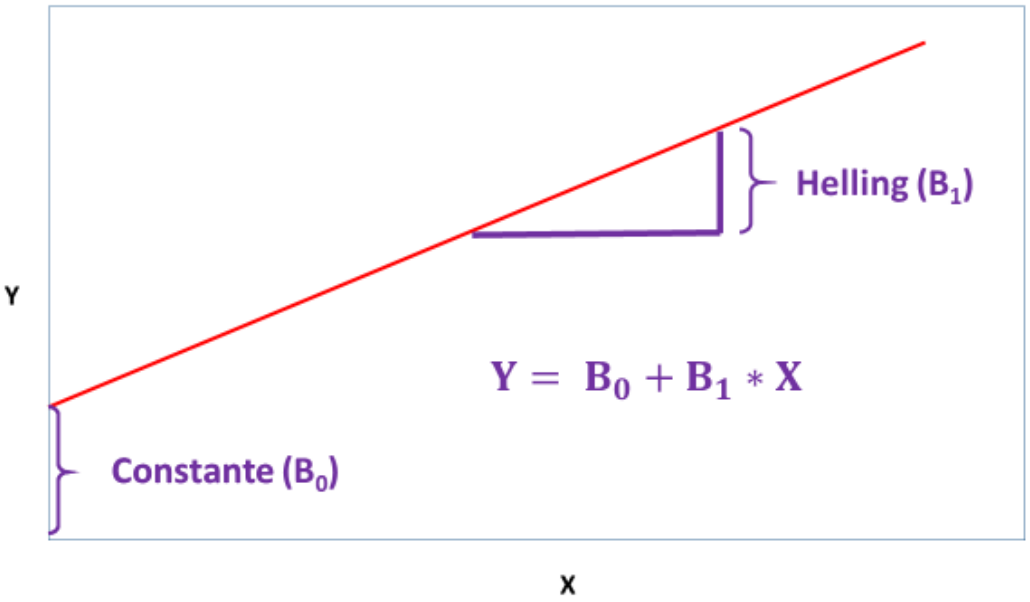
Het schatten van de regressielijn door het minimaliseren van de gekwadrateerde fouten, noemen we de methode van *ordinary least squares* (OLS).

Een interessant aspect van de OLS-methode is dat we de voorspelling kunnen ontleden in componenten. De redenering is als volgt. Stel dat u de waarde van **X** (de verklarende variabele) niet zou kennen. In dat geval is de beste schatting die u kunt maken voor **Y** (de te verklaren variabele), het gemiddelde van **Y**. De variantie van de fouten die u maakt door altijd het gemiddelde als voorspeller te gebruiken, is per definitie gelijk aan de variantie van de variabele **Y**. Na het schatten van een (in dit voorbeeld, stijgende) lineaire regressielijn, zzal u voorspelling voor **Y** groter zijn naarmate **X** toeneemt. Omdat de regressielijn niet perfect aansluit bij de werkelijke waarden, maakt u nog steeds fouten in uw voorspelling, maar die zijn kleiner dan zonder voorkennis over **X**. Een statistiek, de zogeheten R^2 ("*R-kwadraat*"), geeft aan hoeveel procent van de variantie in **Y** door het model wordt verklaard (R^2 ligt tussen 0 en 1, of 0% en 100%).

Het schatten van de rechte lijn die het beste aansluit bij de werkelijke waarden van de **Y** en **X**, resulteert in de twee coëfficiënten die de lijn duiden:

- De **constante** (ook wel *intercept*) genoemd
- De *richtingscoëfficiënt* (ook wel **helling** of hellingshoek genoemd, of in het Engels *slope).*

Wij zullen verder de termen 'constante' en 'helling' gebruiken.



Figuur 4: De constante en de helling, in lineaire regressie. *Bron: OGN.*

Met de coëfficiënten B_0 en B_1 ligt de lijn vast. De constante (B_0) is de waarde van **Y** die hoort bij een waarde **X**=0. Het kan, voor onze ijsverkoper, zijn dat hij toch wel een aantal ijsjes verkoopt, ook al is het geen weer voor ijs. In de praktijk is het niet altijd gemakkelijk om de constante juist te interpreteren, zoals we zullen zien!

De helling (B_1) is meestal interessanter, omdat die het verband aangeeft tussen veranderingen in **Y** die samenhangen met veranderingen in **X**. In ons voorbeeld: hoeveel extra ijsjes worden er verkocht bij een stijging in de temperatuur van één graad?

In een model met slechts één verklarende variabele (hier: temperatuur) zijn B_0 en B_1 gemakkelijk te berekenen. Het valt buiten het bestek van deze module om het bewijs te bespreken, maar voor het minimaliseren van de gekwadrateerde fouten geldt dat:

$$B_1 = \frac{Cov(X,Y)}{Var(X)}$$

En:

$$B_0 = \bar{Y} - B_1 * \bar{X}$$

In deze formules zijn:

Cov(X,Y)	De covariantie tussen X en Y
Var(X)	De variantie van X
$\bar{Y}$; $\bar{X}$	De gemiddeldes van X en Y

Aangezien voor al deze statistieken standaardformules bestaan in **R**, is het gemakkelijk om de coëfficiënten te schatten.

Ook **Excel** heeft standaardfuncties voor lineaire regressieanalyse (in de Nederlandse versie van **Excel**, zijn dit de functies **SNIJPUNT()** en **RICHTING()**.

Er is een **Excel**-bestand (**ijsverkoop_2.xlsx**) bijgevoegd waarin de berekeningen worden gemaakt. De onderstaande tabel is een gedeelte van dit **Excel**-bestand. Zoals u ziet is de constante -70.99, en de helling 9.12.

Tabel 5.1: Regressieanalyse in Excel

ijs	Temp		
99	18	**Constante (B_0):**	**-70.99**
73	18	**Coefficient (B_1):**	**9.12**
..	..		
203	30		
201	30		

Of in formulevorm:

$$IJs = -70.99 + 9.12 * Temp$$

De interpretatie van de helling is duidelijk, en informatief. Bij elke stijging van de temperatuur met één graad, verkopen we gemiddeld 9.12 ijsjes extra.

De interpretatie van de constante is dat bij een temperatuur van 0 graden (**X**=0) we -71 ijsjes verkopen. Dat is natuurlijk onzinnige informatie: het aantal verkochte ijsjes kan laag, en zelfs 0 zijn, maar niet negatief! Deze uitkomst is een gevolg van het feit dat de waarde **X**=0 een irrelevante waarde is: we hebben alleen gegevens over ijsverkoop bij temperaturen tussen 18 en 30 graden (en misschien is de winkel wel dicht als het kouder wordt.). Het is riskant om uitspraken te doen over waarden die buiten het bereik van **X** liggen (*extrapoleren*).

Laten we de regressielijn nu schatten met **R**. Ook gaan we in **R** kijken wat we kunnen doen aan het probleem van de moeilijk te interpreteren constante. En we gaan de analyse uitbreiden naar een model met twee verklarende variabelen.

Na het inlezen van het bestand, en het maken van de gebruikelijke overzichten, maken we gebruik van de standaardfuncties in **R**: **`cov()`**, **`var()`** en **`sd()`**, voor de covariantie, variantie, en standaarddeviatie. De met de bovenstaande formules berekende coëfficiënten drukken we vervolgens overzichtelijk af in de console, met de **`cat()`** functie.

```
> ijsverkoop <- read.csv("ML5_OEFEN1.csv", header=T,sep=",")
> head(ijsverkoop)
  ijs temp evenement
1  99   18         3
2  73   18         2
3  92   19         3
4 104   19         4
5 104   20         3
6 150   21         5
```

```
> b1 <- cov(ijs, temp) / var(temp)
> b0 <- mean(ijs) - b1* mean(temp)
> r  <- cov(ijs, temp) / (sd(ijs)*sd(temp)); r
[1] 0.9105663
> cat("Constante (b0) = ", round(b0,2),
+     "\nCoefficient (b1) = ", round(b1,2),
+     "\nR-square =", round(r^2, 2))
Constante (b0) =  -70.99
Coefficient (b1) =  9.12
R-square = 0.83
```

Bron: OGN.

De coëfficiënten zijn hetzelfde als volgens **Excel**.

We hebben de correlatiecoëfficiënt berekend tussen **X** en **Y**, maar voor de beoordeling van de voorspelkracht van het model hebben de R^2 nodig (de gekwadrateerde correlatiecoëfficiënt; of r^2 in de programmaregel). De waarde geeft aan dat 83% van de variantie van de te verklaren variabele wordt verklaard door het model. Dat betekent dat we met voorkennis over de temperatuur een veel betere schatting kunnen maken van het aantal verkochte ijsjes.

Om het probleem van de moeilijk te interpreteren constante op te lossen, maken we een nieuwe variabele, **temp2**, die het aantal graden voorstelt ten opzichte van de laagste temperatuur (18 graden) waarbij we ijsjes gaan verkopen. We herhalen vervolgens de analyse met **temp2**.

```
> ijsverkoop$temp2 <- ijsverkoop$temp - 18
> head(ijsverkoop, 4)
  ijs temp evenement temp2
1  99   18         3     0
2  73   18         2     0
3  92   19         3     1
4 104   19         4     1

> attach(ijsverkoop)
> b1 <- cov(ijs, temp2) / var(temp2)
> b0 <- mean(ijs) - b1* mean(temp2)
> r  <- cov(ijs, temp2) / (sd(ijs)*sd(temp2)); r

> cat("Constante (b0) = ", round(b0,2),
+     "\nCoefficient (b1) = ", round(b1,2),
+     "\nR-square =", round(r^2, 2))
Constante (b0) =  93.11
Coefficient (b1) =  9.12
R-square = 0.83
```

Het is belangrijk in te zien dat het model niet wezenlijk verandert. De coëfficiënt B_1 is ongewijzigd, en de voorspelkracht ook. Het enige dat verandert is de constante (B_0). De interpretatie van de nieuwe constante is nu het aantal ijsjes dat wordt verkocht bij de laagste temperatuur (18 graden) waarbij we ijsjes gaan verkopen. Dat is een duidelijkere interpretatie.

Een regressiemodel met één verklarende variabele is een bijzondere vorm van een algemeen regressiemodel met één te verklaren variabele en één of meerdere verklarende variabelen.

Het principe van de manier van schatting (OLS) verandert niet bij het toevoegen van verklarende variabelen. In plaats van een ééndimensionale lijn gaan we met twee of meer verklarende variabelen nu meerdimensionale meetkundige vormen schatten (bij twee verklarende variabelen, een vlak in plaats van een lijn). Natuurlijk moeten we ook meer coëfficiënten gaan schatten, en de formules worden een stuk ingewikkelder.

Vóór de uitvinding van de computer waren de berekeningen tijdrovend en foutgevoelig, maar met de komst van computers is het een koud kunstje om bewerkingen uit te voeren op *data matrices*. Het gaat het bestek van deze les te buiten, maar voor wie het zelf wil proberen hebben we het script opgenomen voor de schatting van de coëfficiënten met behulp van matrixalgebra.

Aangetoond kan worden dat voor het minimaliseren van de gekwadrateerde fouten, de oplossing voor de waarden van de coëfficiënten de volgende is:

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

In deze formule zijn:

$\hat{\beta}$	Een vector met geschatte *k+1* coëfficiënten, voor de constante en *k* verklarende variabelen
X	De data matrix met enen in de eerste kolom, en *k* kolommen met gegevens van de verklarende variabelen, en *n* rijen voor het aantal observaties
X^T	De getransponeerde matrix X
Y	Een vector (met lengte *n*) met de gegevens voor de te verklaren variabele.

Toelichting:

*Voor het gebruiken van de matrixfuncties moeten we de kolommen van het data frame opslaan als een matrix (voor **X**) en een vector (voor **Y**). Ook moeten we aan **X** een kolom toevoegen met de waarden 1, voor de constante. De standaardfuncties zijn dan eenvoudig toe te passen. Met **`t()`** transponeren we een matrix. De **`solve()`** functie berekent de inverse van een matrix.*

Het gemakkelijkste is zelf een functie te maken met als input de y- en x-variabelen. Hieronder hebben we de functie **`multreg()`** gedefinieerd, en toegepast. Omdat een lineaire regressie met een verklarende variabele een bijzonder geval is, kunnen we de functie ook gebruiken op ons voorbeeld.

```
> multreg <- function(y,x) {
+   x <- as.matrix(x)
+   x <- cbind(Constante = 1,x)
+   b <- solve(t(x) %*% x) %*% t(x) %*% y
+   colnames(b) <- "Coefficient"
+   print(b)
+ }

> multreg(y=ijsverkoop$ijs,x=ijsverkoop[2])
          Coefficient
Constante  -70.990990
temp         9.116534
```

Zoals u ziet zijn de uitkomsten weer hetzelfde!

Met onze zelfgeschreven functie kunnen we een model schatten met één of meer verklarende variabelen. We kunnen nu eenvoudig de variabele **evenement** erbij betrekken.

```
> multreg(y=ijsverkoop$ijs,x=ijsverkoop[2:3])
          Coefficient
Constante  -136.40701
temp         10.18991
evenement    13.69268
```

Natuurlijk hebben we hier weer dezelfde uitdaging bij het interpreteren van de constante, maar u weet intussen hoe u dit kunt oplossen! Omdat we meestal niet erg geïnteresseerd zijn in de constante laten we het verder voor wat het is.

De constante is hier het aantal verkochte ijsjes als alle verklarende variabelen 0 zijn; maar **evenement** is gemeten op een 5-puntsschaal (van 1 tot 5) en kan geen 0 zijn! De coëfficiënt voor evenement geeft aan dat het aantal verkochte ijsjes met 13.69 stijgt als de drukte in de stad één punt, op de 5-puntsschaal, stijgt.

Ook ziet u dat de coëfficiënt voor **temperatuur** verandert, van 9.12 naar 10.19. Dit komt doordat nu wordt "gecontroleerd" voor de waarde van **evenement**. De nieuwe schatting is een betere schatting omdat het model nu completer is. De eerdere coëfficiënt werd naar beneden gedrukt omdat er kennelijk een (toevallige) negatieve relatie is tussen de twee verklarende variabelen: het was gemiddeld iets kouder op drukke dagen. We kunnen dit testen met de **`cor()`** functie, voor correlaties.

```
> cor(ijsverkoop)
                ijs       temp  evenement      temp2
ijs       1.0000000  0.9105663  0.1190785  0.9105663
temp      0.9105663  1.0000000 -0.2712868  1.0000000
evenement 0.1190785 -0.2712868  1.0000000 -0.2712868
temp2     0.9105663  1.0000000 -0.2712868  1.0000000
```

De correlatie tussen **temperatuur** en **evenement** is inderdaad negatief (-.27). We zien ook de eerder berekende correlatie tussen **ijs** en **temperatuur**!

Gelukkig beschikt **R** (en *packages* binnen **R**) over uitgebreide mogelijkheden om een regressieanalyse uit te voeren. Vooral het **cars** *package* mag niet onvermeld blijven. Het **cars** *package* biedt ook elegante mogelijkheden om het model te testen, en te visualiseren.

Dummyvariabelen

Regressieanalyse lijkt beperkt te zijn omdat de te verklaren en verklarende variabelen moeten worden gemeten op een interval- of ratioschaal (zoals temperatuur; inkomen; lengte; enzovoorts). Dat is echter niet het geval.

Er zijn ook varianten voor regressieanalyse waarbij **de te verklaren variabele** categoriaal is (twee uitkomsten, zoals ja/nee; of een ordinale uitkomst, zoals een 5-puntsschaal). In geval van een *binaire* uitkomst (0/1, of ja/nee) spreken we van *logistische regressie*. Dat zullen we hier niet bespreken.

Wel bespreken we het gebruik van categoriale **verklarende variabelen** (groepen). In het simpelste geval hebben we twee groepen, zoals studenten die voor een examen wel of niet een aparte training hebben gevolgd. Het is belangrijk (vooral voor de interpretatie van de constante) om die twee groepen te coderen als 0 en 1. We noemen dit *dummycodering*. Voor de interpretatie is het verstandig een referentiegroep te kiezen, en die te coderen als 0. In dit voorbeeld kunnen we de groep die geen training heeft gevolgd, de referentiegroep noemen (met code 0). De "interessante" of experimentele groep die wel de training heeft gevolgd, coderen we dan als 1.

In ons uitgebreidere voorbeeld gaan we regressieanalyse toepassen op een bestand van 200 studenten waarvan er 50 een speciale examentraining hebben gevolgd. Het bestand bevat de behaalde cijfers, het aantal studie-uren en een indicatie (0/1) voor het volgen van de examentraining.

Stap 1: inlezen van de data

Eerst lezen we de data in, die in *csv*-formaat in de file **cijfer.csv** zijn opgeslagen. Het is een goed gebruik met de **`summary()`** en **`str()`** commando's de data te inspecteren.

```
> cijfer <- read.csv("ML5_OEFEN2.csv", header=TRUE)
> summary(cijfer)
     cijfer           uren          training
 Min.   :  5.00   Min.   :20.00   Min.   :0.00
 1st Qu.: 35.00   1st Qu.:34.00   1st Qu.:0.00
 Median : 51.00   Median :51.00   Median :0.00
 Mean   : 50.63   Mean   :49.74   Mean   :0.25
 3rd Qu.: 66.25   3rd Qu.:64.25   3rd Qu.:0.25
 Max.   :100.00   Max.   :79.00   Max.   :1.00
> str(cijfer)
'data.frame':  200 obs. of  3 variables:
 $ cijfer  : int  45 25 32 23 86 48 24 17 48 53 ...
 $ uren    : int  40 36 28 21 72 41 24 39 53 72 ...
 $ training: int  0 0 0 0 0 0 0 0 0 0 ...
```

Uit het overzicht zien we dat de waardes voor **cijfer** uiteenlopen van 5 tot 100. Het gemiddelde cijfer is 50.63. Omdat de mediaan (51) ongeveer gelijk is aan het gemiddelde, verwachten we een verdeling die redelijk symmetrisch is.

Het aantal studie-uren loopt van 20 tot 79, met een gemiddelde van 49.74. Het al of niet volgen van een training is gecodeerd als een *dummyvariabele* met de waarden 0 en 1. De variabele wordt bij het inlezen in **R** gezien als een *integer*, een geheel getal. En van getallen kan een gemiddelde worden berekend. Veel onderzoekers en analisten geven er de voorkeur aan om dergelijke variabelen te beschouwen als een nominale variabele. De waarden 0 en 1 zijn immers willekeurige waarden, en hebben op zich geen betekenis. Zoals hierboven uitgelegd, is het voor het opnemen van nominale variabelen in een regressieanalyse nodig groepen te coderen met

nullen en enen. In **R** hoeven we die codering niet zelf aan te brengen omdat het programma dat voor ons doet: factorvariabelen (bijvoorbeeld gecodeerd als “ja” en “nee”) worden door **R** zelf gecodeerd als 0 en 1. We laten voor de volledigheid zien hoe dit werkt, door de variabele **training** nog eens te definiëren, als *factor*.

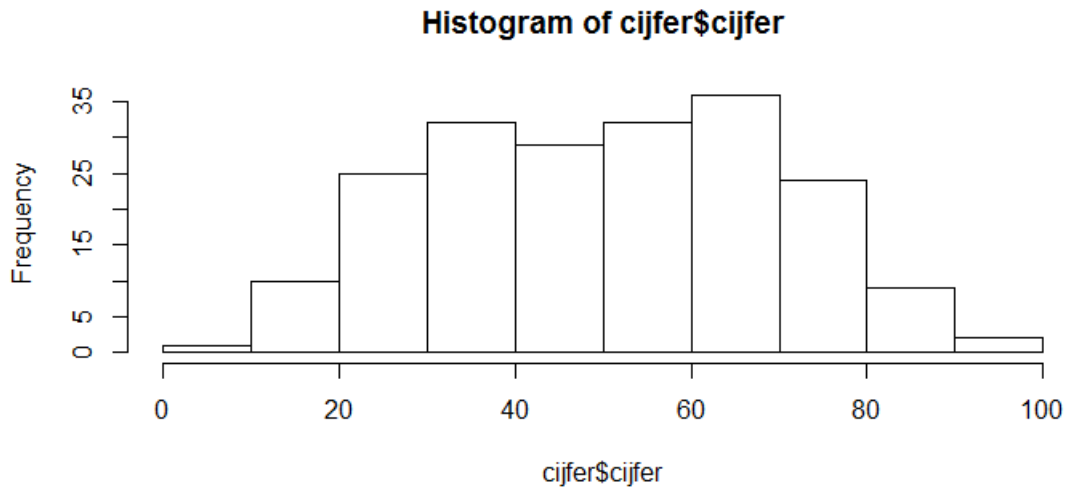
Let op de verschillen in de weergave van de samenvattende informatie, voor de twee variabelen!

```
> cijfer$trainfac <- as.factor(cijfer$training) # training as factor
> summary(cijfer)
     cijfer           uren          training    trainfac
 Min.   :  5.00   Min.   :20.00   Min.   :0.00   0:150
 1st Qu.: 35.00   1st Qu.:34.00   1st Qu.:0.00   1: 50
 Median : 51.00   Median :51.00   Median :0.00
 Mean   : 50.63   Mean   :49.74   Mean   :0.25
 3rd Qu.: 66.25   3rd Qu.:64.25   3rd Qu.:0.25
 Max.   :100.00   Max.   :79.00   Max.   :1.00
> str(cijfer)
'data.frame':  200 obs. of  4 variables:
 $ cijfer  : int  45 25 32 23 86 48 24 17 48 53 ...
 $ uren    : int  40 36 28 21 72 41 24 39 53 72 ...
 $ training: int  0 0 0 0 0 0 0 0 0 0 ...
 $ trainfac: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

Stap 2: verkennen van de data

Voor regressieanalyse is het belangrijk dat de te verklaren variabele, hier **cijfer** (bij benadering) normaal is verdeeld. We kunnen kijken of dit het geval is met een histogram van deze variabele. De functie **`hist()`** geeft een histogram.

```
> hist(cijfer$cijfer)
```



Figuur 5: Histogram van behaalde cijfers. *Bron: OGN.*

Het histogram laat een redelijk symmetrische verdeling zien. De verdeling heeft de piek niet in het midden zoals in een normale verdeling (de verdeling is bimodaal, met pieken tussen 30 en 40, en tussen 60 en 70. Maar in de praktijk is geen enkele verdeling perfect normaal, en de analyse is vrij robuust voor kleine afwijkingen van de normale verdeling, zeker als – zoals hier – de verdeling symmetrisch is.

In ons model willen we achterhalen of het behaalde cijfer, als te verklaren variabele, afhangt van het aantal studie-uren en van het volgen van een training. Voor een eerste inzicht kunnen we de samenhangen tussen de variabelen laten zien in een correlatiematrix, met het **`cor()`** commando.

Dit commando werkt echter niet op factorvariabelen!

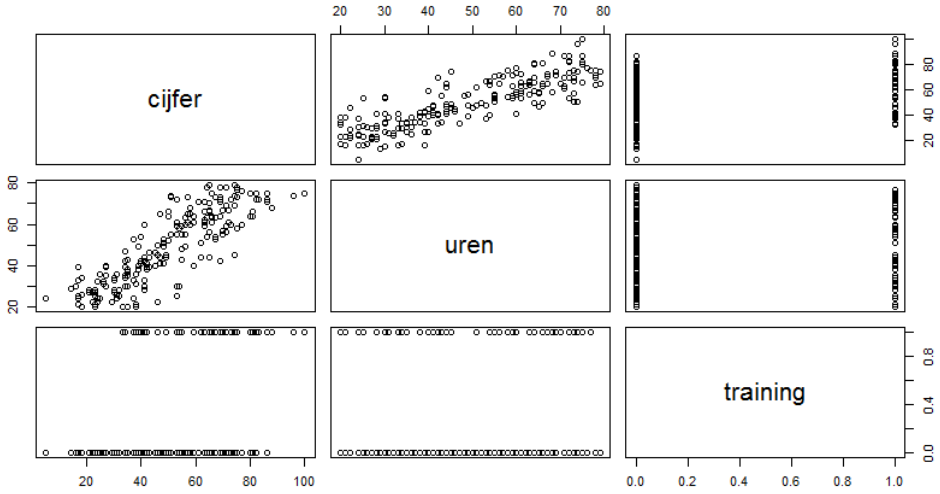
```
> cor(cijfer) # Foutmelding: Error in cor(cijfer) : 'x' must be numeric
Error in cor(cijfer) : 'x' must be numeric
> cor(cijfer[,1:3])
             cijfer       uren   training
cijfer   1.0000000 0.84780706 0.31006045
uren     0.8478071 1.00000000 0.02072542
training 0.3100605 0.02072542 1.00000000

> round(cor(cijfer[,1:3]),2)
         cijfer uren training
cijfer     1.00 0.85     0.31
uren       0.85 1.00     0.02
training   0.31 0.02     1.00
```

In de matrix zijn de waarden op de diagonaal (de correlatie van een variabele met zichzelf) altijd gelijk aan 1, en dus niet interessant. Verder is de matrix symmetrisch: de correlatie tussen uren en cijfer, is gelijk aan de correlatie tussen cijfer en uren. We hoeven dus slechts op één kant van de diagonaal te letten! In dit geval kijken we naar de geel gemarkeerde getallen aan de onderkant van de diagonaal.

De correlatie tussen **cijfer** en uren, is hoog: 0.85. De gekwadrateerde waarde (0.85^2=0.72) geeft aan dat 72% van de variantie in **cijfer**, wordt verklaard door de variantie in het aantal studie-uren. Voor **training** is dat een stuk lager (de correlatie is 0.31). Maar daarbij moet worden bedacht dat **training** een *dummy-variabele* is die niet vrijelijk kan variëren: de variabele is of 0 of 1! Zelfs al zouden alle hoge cijfers worden behaald door studenten die getraind zijn, dan nog kan de correlatie een stuk lager zijn dan 1, doordat **cijfer** wel, en **training** niet kan variëren!

Ondanks de positieve correlatie tussen **cijfer** en **training**, kan het zijn dat **training** niet bijdraagt aan de verklaring van **cijfer**. Dit zal het geval zijn als de correlatie tussen de twee verklarende variabelen erg hoog is: diegenen die training hebben genoten, spenderen ook veel meer uren aan de voorbereiding. Dat lijkt echter niet het geval te zijn, want de correlatie tussen **uren** en **training** is gering (0.02).

Met het **`pairs()`** commando kunnen we alle gepaard gaande relaties tussen de variabelen grafisch weergeven, in een *scatter-diagram*.

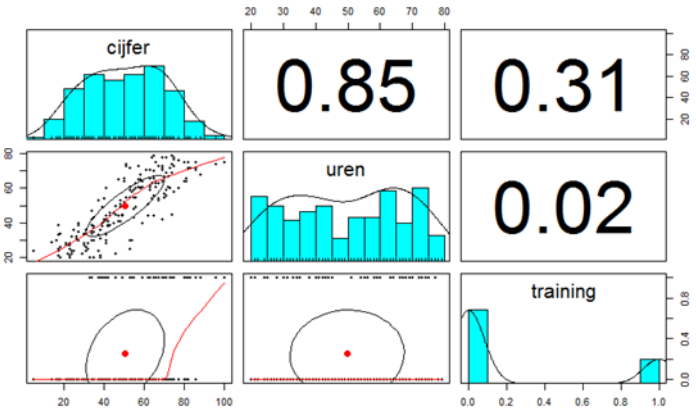


Figuur 6. Scatter-diagram matrix. *Bron: OGN.*

Omdat we **cijfer**, als de te verklaren variabele, op de y-as willen hebben, zijn de twee grafieken in de eerste rij het meest geschikt.

In de grafiek met cijfer op de y-as en uren op de x-as, blijkt duidelijk het positieve verband tussen de twee variabelen. Het verband tussen **cijfer** en **training**, in het grafiekje rechtsboven, is lastiger te zien. De verdeling van cijfers is gesplitst in een verdeling voor de groep zonder training (**training**=0) en een verdeling voor de groep met training (**training**=1). Omdat we maar 50 personen met training hebben zijn de punten aan de rechterkant nog goed te zien. Maar met 150 personen zonder training aan de linkerkant, worden de punten vrijwel een dikke streep! U kunt zich voorstellen dat met grote datasets dit soort grafieken lastig te lezen zal zijn. In dit geval kunnen we nog wel zien dat de verdeling van cijfers voor personen met training hoger ligt (hoger minimum; en hoger maximum).

Een handige uitbreiding van de **`pairs()`** plot is te verkrijgen met het **`pairs.panel()`** commando in het **psych** *package*.

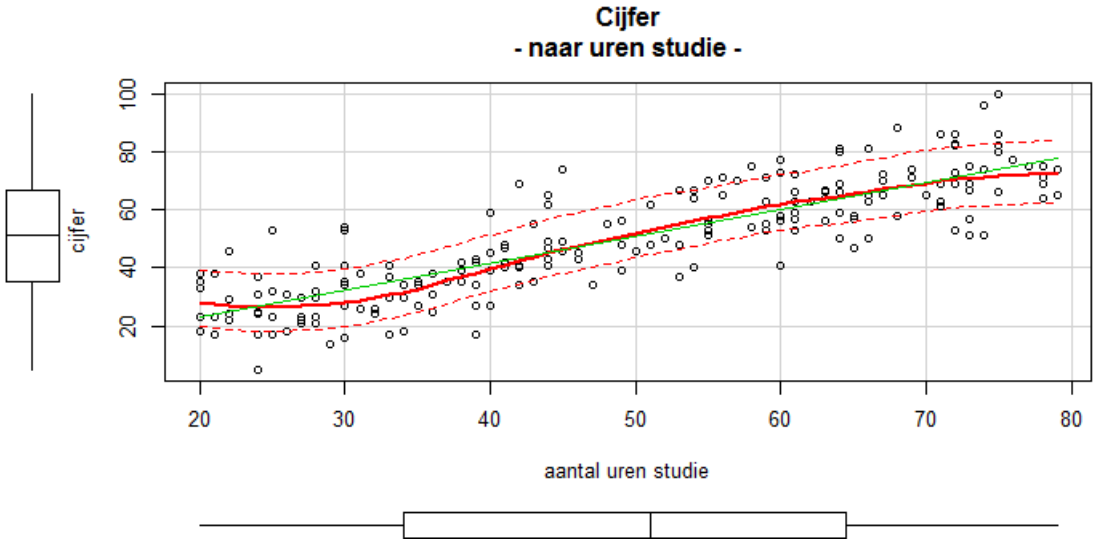


Figuur 7. Scatter-diagram matrix (psych package). *Bron: OGN.*

Rechts boven in de grafiek worden de correlaties tussen de variabelen weergegeven, en linksonder de *scatter*-plots en de *correlatie ellipsen*. Hoe ronder de ellips is, des te lager de correlatie. Hoge correlaties worden gekenmerkt door een "sigaarvormige" ellips.

Voor het verkennen van de data, kan ook gebruik worden gemaakt van de uitgebreide (grafische) opties in het **cars** *package*. Voor de relatie tussen **cijfers** en **uren**, geven we hieronder de *scatterplot*. In die plot wordt op de assen de verdeling (als *boxplot*) gegeven van de variabelen. In de grafiek worden naast punten, ook de geschatte lineaire regressielijn (de groene lijn) weergegeven en de zogenoemde *lowess*, een niet-rechte lijn die zo goed mogelijk aansluit bij de punten. Deze lijn geeft soms goede informatie voor het verbeteren van het model, bijvoorbeeld als het verband niet lineair is. Uit de curve (de rode lijn) ziet u dat er van een zwakke s-vorm sprake is: aanvankelijk stijgt het cijfer nauwelijks met het toenemen van het aantal studie-uren. Pas bij 30 studie-uren is er sprake van een positief verband. Na 70 studie-uren vlakt de lijn weer af: elk uur boven de 70 studie-uren leidt nauwelijks tot een toename in het cijfer.

```
> library(car)
> scatterplot(cijfer ~ uren, data=cijfer,
+             xlab="aantal uren studie", ylab="cijfer",
+             main="Cijfer\n- naar uren studie -")
```



Figuur 8. Scatterplot (cars package). *Bron: OGN.*

Stap 3: het trainen van het model

Voor het trainen van het model kunnen we gebruik maken van diverse *packages*. Voor uitgebreide en geavanceerde regressieanalyse is het **cars** *package* een absolute aanrader. In de meeste gevallen zijn echter de functies in het **stats** *package* toereikend. Het **stats** *package* bevindt zich in de *system library* van **R**, en hoeft dus niet apart te worden geïnstalleerd.

In de notatie van **R** wordt de relatie tussen te verklaren, en verklarende variabelen weergegeven met een tilde ("~"). De betekenis van de tilde is 'wordt verklaard door'. In die logica komt de te verklaren variabele aan de linkerkant van de tilde, en de verklarende variabele(n) aan de rechterkant.

Je zult de notatie met de tilde telkens terugzien bij aan regressieanalyse gerelateerde technieken (bijvoorbeeld de formules voor neurale netwerken, in het volgende hoofdstuk).

We zullen onze analyse opbouwen in een aantal stappen. Eerst schatten we een model met alleen **uren** als verklarende variabele. Voor dit eenvoudige model bestuderen we de output. Daarna kijken we naar een model dat is uitgebreid met **training** als extra verklarende variabele. We bespreken dan ook een test om te beoordelen of het uitgebreide model beter is dan het simpele model. Ten slotte laten we zien dan **R** inderdaad geen moeite heeft met opnemen van factorvariabelen in het model.

Het schatten van het model maakt gebruik van de **`lm()`** functie. De letters "lm" staan voor *linear model* (lineair model). Tussen haakjes staat de relatie tussen de variabelen, gevolgd door de naam van het data frame waarin de variabelen staan. De resultaten worden weggeschreven naar een object (hier **model1**). We kunnen het object afdrukken in de console, maar de informatie is summier; alleen de coëfficiënten worden weergegeven. Meer informatie wordt verkregen met het **`summary()`** commando.

```
> model1 <- lm(cijfer ~ uren, data=cijfer)
> model1

Call:
lm(formula = cijfer ~ uren, data = cijfer)

Coefficients:
(Intercept)         uren
     4.2377       0.9326

> summary(model1)

Call:
lm(formula = cijfer ~ uren, data = cijfer)

Residuals:
    Min      1Q  Median      3Q     Max
-23.609  -6.300  -1.250   5.126  27.795

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.23768    2.18876   1.936   0.0543 .
uren         0.93260    0.04146  22.496   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.37 on 198 degrees of freedom
Multiple R-squared:  0.7188,   Adjusted R-squared:  0.7174
F-statistic: 506.1 on 1 and 198 DF,  p-value: < 2.2e-16
```

De output van het **`summary()`** commando is omvangrijk. Laten we er stap voor stap naar kijken.

De residuals

In de output wordt, na het herhalen van het model, onder "*call*", informatie gegeven over de *residuals*.

```
Residuals:
    Min      1Q  Median      3Q     Max
-23.609  -6.300  -1.250   5.126  27.795
```

De *residuals* zijn de afwijkingen tussen de werkelijke waarden van **cijfer**, en de door het model voorspelde waarden (zoals in figuur 2). De afwijkingen bevinden zich in het interval van -23.6 tot +27.8. Ook met voorkennis over het aantal studie-uren kunt u voorspelling er dus nog flink naast zitten! De kwartielen (1Q en 3Q) geven aan dat in de helft van de gevallen de fout zich beweegt tussen -6.3 en +5.1.

Dat klinkt niet slecht. Maar om te beoordelen hoe goed dit is, hebben we een *benchmark* nodig. De logische benchmark is een model zonder verklarende variabelen!

De *benchmark* kan worden verkregen met het onderstaande model, **model0**. Na de tilde geven we een 1 aan (alleen een constante, en geen verklarende variabelen). De constante (*intercept*) is het gemiddelde van **cijfer**. Anders gezegd, zonder enige voorinformatie over verklarende variabelen, is de beste schatting van het cijfer van een student, het gemiddelde cijfer van alle studenten!

De *residuals* lopen nu uiteen van -45.6 tot +49.4. Dat is simpel te begrijpen. Onze beste voorspelling voor **cijfer** - zonder verdere voorkennis - is het gemiddelde. De maximale overschatting is dan gelijk aan het gemiddelde minus het minimum van **cijfer**: 50.63 – 5 = 45.63. De maximale onderschatting is 100 – 50.63 = 49.37.

```
> model0 <- lm(cijfer ~ 1, data=cijfer) # alleen constante;
>                                       # geen verklarende variabelen
> summary(model0)
```

```
Call:
lm(formula = cijfer ~ 1, data = cijfer)

Residuals:
   Min     1Q Median     3Q    Max
-45.63 -15.63   0.37  15.62  49.37

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   50.630      1.379    36.7   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 19.51 on 199 degrees of freedom
```

```
# zelf berekende benchmark
> (max.onderschatting  <- mean(cijfer$cijfer) - min(cijfer$cijfer))
[1] 45.63
> (max.overschatting   <- max(cijfer$cijfer)  - mean(cijfer$cijfer))
[1] 49.37
> (spreiding.schatting <- max(cijfer$cijfer)  - min(cijfer$cijfer))
[1] 95
```

De coëfficiënten

De interpretatie van de coëfficiënten is eenvoudig.

Zoals we hebben uitgelegd is de waarde van de constante (*intercept*) niet altijd zinnig. Het is het verwachte cijfer bij een aantal studie-uren van nul. Maar er is (althans in onze data set) geen student die minder uren studeert dan 20!

Belangrijker is de coëfficiënt (de helling) voor **uren**. Het geeft aan met hoeveel het cijfer stijgt door één extra uur te studeren.

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.23768    2.18876   1.936   0.0543 .
uren         0.93260    0.04146  22.496   <2e-16 ***
```

Voor elke extra uur studie gaat het cijfer naar verwachting 0.93 omhoog. Omdat het verband lineair is, geldt ook dat het cijfer met 10*0.03 = 9.3 punten omhoog gaat als 10 extra uren wordt gestudeerd. De gemiddelde student studeert 49.475 uur; het verwachte cijfer is dan 4.23768 + 0.93260 * 49.745 = 50.63 (het gemiddelde van **cijfer**!).

Het is overigens niet nodig om hiervoor een rekenmachine te gebruiken. De twee coëfficiënten (de constante en de helling) van **model1** zijn opgeslagen in **`model1$coefficients`**. Zij kunnen worden opgehaald met **`model1$coefficients[1]`** en **`model1$coefficients[2]`**. Het gemiddelde aantal studie-uren is te berekenen met **`mean()`**. We kunnen een eenvoudige functie **`voorspel()`** schrijven.

```
> model1$coefficients
(Intercept)        uren
  4.2376847   0.9326026

voorspel <-function(x){
  y<-model1$coefficients[1] + model1$coefficients[2] * x
  cat("Voorspeld cijfer bij", x, "studie-uren",round(y,2))
}
```

```
> voorspel(0)
Voorspeld cijfer bij 0 studie-uren 4.24

> voorspel(20)
Voorspeld cijfer bij 20 studie-uren 22.89

> voorspel(mean(cijfer$uren)) # gemiddeld aantal studie-uren
Voorspeld cijfer bij 49.745 studie-uren 50.63

> voorspel(60)
Voorspeld cijfer bij 60 studie-uren 60.19
```

De Goodness-of-Fit

Het laatste deel van de output laat zien hoe goed het model aansluit bij de data. In het statistisch jargon, gebruiken we hiervoor *goodness-of-fit* indicatoren. De R^2 is een populaire *goodness-of-fit* indicator.

```
Residual standard error: 10.37 on 198 degrees of freedom
Multiple R-squared:  0.7188,   Adjusted R-squared:  0.7174
F-statistic: 506.1 on 1 and 198 DF, p-value: < 2.2e-16
```

De *residual standard error (RSE)* geeft de standaardfout aan van de *residuals*. In een normale verdeling ligt 95% van de residuals in het bereik 0 ± 1.96*RSE. In dit geval is dat het bereik van -20.33 tot +20.33 (dat is, zoals verwacht, iets kleiner dan de maximale fouten die we maken).

De R^2 (*multiple R-squared*) geeft het percentage van de door het model verklaarde variantie in **cijfer** aan. Dit model verklaart bijna 72% van de variantie.

Ten slotte is er de *F-statistic* die weergeeft of een model met verklarende variabelen significant beter is dan een model zonder verklarende variabelen (de benchmark). De *p*-waarde is de kans op deze *F*-statistic, onder de veronderstelling dat in werkelijkheid de coëfficiënten nul zijn. Bij een lage *p*-waarde (meestal gebruiken we p-value<0.05) concluderen we dat de benchmark kan worden verworpen: het model met verklarende variabelen is beter dan de benchmark. De *p*-waarde is nagenoeg gelijk aan 0 ($2.2*10^{-16}$).

In een model met slechts één verklarende variabele voegt de *F-statistic* overigens niets toe: de F-waarde is dan gelijk aan het kwadraat van de t-waarde van de coëfficiënt van de enige verklarende variabele (**uren**): $22.496^2 = 506.1$.

Twee verklarende variabelen

We gaan nu het model uitbreiden met een extra verklarende variabele, namelijk de dummyvariabele voor training. Wij kunnen in de **`lm()`** functie verklarende variabelen toevoegen met "+".

```
> # twee verklarende variabelen
>
> model2 <- lm(cijfer ~ uren + training, data=cijfer)
> summary(model2)

Call:
lm(formula = cijfer ~ uren + training, data = cijfer)

Residuals:
    Min      1Q  Median      3Q     Max
-24.432  -5.468   0.346   5.264  24.347
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.28220    1.85768   0.690    0.491
uren         0.92593    0.03467  26.705   <2e-16 ***
training    13.14938    1.41643   9.283   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 8.672 on 197 degrees of freedom
Multiple R-squared:  0.8044,   Adjusted R-squared:  0.8024
F-statistic:   405 on 2 and 197 DF,  p-value: < 2.2e-16
```

Het interpreteren van de coëfficiënt voor **uren** voor dit met **training** uitgebreide model, is hetzelfde als in het model met alleen **uren** als verklarende variabele. De waarde is iets anders, omdat het effect van het aantal studie-uren op het cijfer nu gecorrigeerd wordt voor het effect van training. Omdat er een licht positieve correlatie is tussen **training** en **uren**, bevatte de iets hogere coëfficiënt in **model1** ook een deel van het effect van **training**. Nu **training** expliciet is opgenomen in **model2** is het “netto-effect” van **uren** iets lager.

De interpretatie van de dummyvariabele is, dat voor elke verandering met één eenheid van **training**, het cijfer 13.1 punt hoger wordt. Omdat de dummyvariabele maar twee mogelijke waarden heeft (0 en 1), komt het erop neer dat studenten die de training hebben gevolgd gemiddeld ruim 13 punten hoger scoren dan de groep die geen training heeft gevolgd.

> *Zelftest: probeer nu zelf de functie te schrijven die het cijfer voorspelt op basis van waarden van **uren** en **training**!*

Regressie met een factorvariabele

Om te laten zien dat **R** op een slimme manier omgaat met factorvariabelen, herhalen we het voorgaande model, met **trainfac** in plaats van **training**.

We hebben eerder al de **`as.factor()`** functie gebruikt om van de (0/1) dummyvariabele een factor te maken. Op het oog verandert er niet veel. Waar nullen en enen staan voor de variabele **training**, staan er nog steeds nullen en enen in **trainfac**. Alleen met de **`str()`** zien we dat de ene variabele een numerieke variabele is, en de andere een factorvariabele.

Het is instructiever om nog een extra factorvariabele te maken, met de waarden “nee” en “ja”, in plaats van “0” en “1”. Omdat de records zijn gesorteerd op **training**, laten we de eerste en de laatste records zien.

De twee factorvariabelen wekken waarschijnlijk verwarring.

- De factorvariabele **trainfac** heeft 2 niveaus, gelabeld “0” en “1”. De waarde die alfabetisch het eerste komt (“0”), krijgt code 1. De waarden die volgen (hier alleen “1”) krijgt code 2. Omdat het bestand begint met waarden “0”, zien we de codes 1.
- De factorvariabele **train_jn** heeft ook 2 niveaus, gelabeld “ja” en “nee”. De waarde die alfabetisch het eerste komt (“ja”), krijgt de code 1, en “nee” de code 2. Het bestand begint dus met een reeks codes 2.

```
> str(cijfer)
'data.frame':   200 obs. of  5 variables:
 $ cijfer  : int  45 25 32 23 86 48 24 17 48 53 ...
 $ uren    : int  40 36 28 21 72 41 24 39 53 72 ...
 $ training: int  0 0 0 0 0 0 0 0 0 0 ...
 $ trainfac: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ train_jn: Factor w/ 2 levels "ja","nee": 2 2 2 2 2 2 2 2 2 2 ...
```

```
> head(cijfer[,3:5], 3); tail(cijfer[,3:5], 3)
  training trainfac train_jn
1        0        0      nee
2        0        0      nee
3        0        0      nee
    training trainfac train_jn
198        1        1       ja
199        1        1       ja
200        1        1       ja
```

We schatten nu het model met **uren** en de factorvariabelen. In **model3a** gebruiken we **trainfac**, en in **model3b train_jn**.

De output van **model3a** is identiek aan de eerdere output (**model2**).

```
> summary(model3a)

[output weggelaten]

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.28220    1.85768   0.690    0.491
uren         0.92593    0.03467  26.705   <2e-16 ***
trainfac1   13.14938    1.41643   9.283   <2e-16 ***

Multiple R-squared:  0.8044,   Adjusted R-squared:  0.8024
```

In **model3b** ziet de output er echter anders uit!

```
> summary(model3b)

[output weggelaten]

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  14.43157    2.13433   6.762 1.51e-10 ***
uren          0.92593    0.03467  26.705  < 2e-16 ***
train_jnnee -13.14938    1.41643  -9.283  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Multiple R-squared:  0.8044,   Adjusted R-squared:  0.8024
```

De coëfficiënt voor **uren** is hetzelfde. En ook de R^2, en de significantie van de variabelen veranderen niet. Maar de constante, en het teken van de variabele voor training, veranderen!

Bij dummyvariabelen wordt gebruikgemaakt van een basis (of referentie-)categorie. Het is aan de analist om de keuze van de referentiecategorie te bepalen. Omdat we willen weten wat het effect is van training, gebruiken we “geen training” als referentiecategorie. Bij factorvariabelen echter maakt het programma de keuze voor ons. De *default* referentiecategorie is de waarde die alfabetisch het eerste komt (en alfabetisch komt “ja” voor “nee”).

De oplossing is zelf referentiecategorie te bepalen met het **`relevel()`** commando. Tussen haakjes moeten we de factorvariabele aangeven, gevolgd door de gewenste referentiecategorie. Daarna herhalen we regressieanalyse met de nieuwe referentiecategorie (**model3c**).

```
> cijfer$train_jn <- relevel(cijfer$train_jn, "nee")
> model3c <- lm(cijfer ~ uren + train_jn, data=cijfer)
> summary(model3c)
```

…

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.28220    1.85768   0.690    0.491
uren         0.92593    0.03467  26.705   <2e-16 ***
train_jnja  13.14938    1.41643   9.283   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

…

Step 4: het verbeteren van het model

Hogere orde termen (polynomiaal)

Het model is vaak te verbeteren. Hoewel "lineaire" regressieanalyse uitgaat van de veronderstelling dat het verband tussen de verklarende variabele en de te verklaren variabele rechtlijnig is. Maar er zijn allerlei andere relaties denkbaar. Veel voorkomende niet-lineaire verbanden zijn kromlijnig; parabolisch (U-vormig); S-vormig; enzovoorts.

Deze stereotype verbanden kunnen worden geschat door het model uit te breiden. Bijvoorbeeld, in het geval van kromlijnige en U-vormige verbanden kan het model worden uitgebreid met een term X^2.

$$Y = b_0 + b_1 * X + b_2 * X^2$$

Bij het verkennen van de gegevens zagen we een rechtlijnig verband, en geen kromlijnig. De *lowess* suggereerde een licht S-vormig verband.

De uitkomsten van **model4**, met een term voor het kwadraat van **uren**, leidt inderdaad niet tot betere uitkomsten. De coëfficiënt van de nieuwe term is niet significant anders dan 0, en kan dus net zo goed achterwege worden gelaten.

```
> cijfer$uren2 <- cijfer$uren * cijfer$uren
> head(cijfer[,c(2,6)], 3)
  uren uren2
1   40  1600
2   36  1296
3   28   784

> model4 <- lm(cijfer ~ uren + uren2 + train_jn, data=cijfer)
> summary(model4)
```

…

```
Residuals:
     Min       1Q   Median       3Q      Max
-24.6293  -5.3470   0.3943   5.4022  24.6451

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.2100704  5.2898327  -0.040    0.968
uren         0.9958892  0.2347052   4.243  3.4e-05 ***
uren2       -0.0007121  0.0023626  -0.301    0.763
train_jnja  13.1377561  1.4202288   9.250  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.692 on 196 degrees of freedom
Multiple R-squared:  0.8045,   Adjusted R-squared:  0.8015
F-statistic: 268.8 on 3 and 196 DF,  p-value: < 2.2e-16
```

Interactie

Ook kan het helpen een zogenoemde *interactie*-term op te nemen. Deze term veronderstelt dat het effect van studie-uren verschilt, voor de verschillende niveaus van **training**. Het is voorstelbaar dat het effect van extra studie-uren groter is, na deelname aan de training.

Het interactie-effect kan worden getest door het opnemen van het product van de interacterende variabelen. Het is niet nodig eerst een nieuwe variabele aan te maken; het interactie-effect kan worden toegevoegd aan het model, als **uren:training**.

Ook het interactie-effect is niet significant, en kan dus uit het model worden weggelaten.

In het algemeen geldt dat de verklaarde variantie (R^2) altijd toeneemt bij het toevoegen van variabelen (inclusief polynomialen en interactie-termen). Check voor jezelf dat de R^2 inderdaad iets groter is dan in de voorgaande modellen! Het principe van spaarzaamheid zegt dat het de voorkeur verdient om niet-significante termen niet in het uiteindelijke model op te nemen.

```
> # interactie
> model5 <- lm(cijfer ~ uren + training + uren:training, data=cijfer)
> summary(model5)

…
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.74168    2.09631   0.354 0.723866
uren          0.93684    0.03982  23.524  < 2e-16 ***
training     15.43681    4.32385   3.570 0.000449 ***
uren:training -0.04559    0.08140  -0.560 0.576091
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Multiple R-squared:  0.8047,   Adjusted R-squared:  0.8017
```

Regressiebomen: het algoritme

Regressieanalyse kan ook worden ingezet bij het maken van beslissingsbomen. De combinatie van regressieanalyse en het idee van beslissingsbomen, leidt tot zogenoemde *regression trees* (“regressiebomen”). Het principe van deze techniek is ingewikkelder dan regressieanalyse.

In regressiebomen maken we numerieke voorspellingen, voor elke vertakking in de boom. Het voordeel van deze techniek is dat het inzichtelijker is dan een regressiemodel, en dat het niet nodig is om al vooraf een model te specificeren (het model, of de boom, groeit op basis van de waargenomen data). In die zin is de techniek meer beschrijvend, dan de “toetsende” regressieanalyse. Als verdere voordelen worden genoemd dat het meer geschikt is voor taken waarin veel variabelen een rol spelen, en waarin de verbanden niet-lineair zijn. Bovendien worden er minder strenge eisen gesteld aan de verdeling van de data (zoals de eis van een normaal verdeelde te verklaren variabele).

In het algoritme worden de data gesplitst op basis van kenmerken (variabelen) in de data set. Aangezien we te maken met numerieke data, maken we nu voor het meten van de homogeniteit van de gevormde groepen gebruik van de standaarddeviatie.

Algoritmes voor regressiebomen zijn er in twee varianten. In de eerste variant – anders dan de naam “regressiebomen” doet vermoeden – maken we geen gebruik van regressieschatting. In de tweede variant is dat wel het geval. Voor de uiteindelijke “bladen van de boom”, wordt een regressieschatting gemaakt op basis van de voor die tak van de boom geldende variabelen.

In ons voorbeeld zullen we eerst gebruikmaken van de eerste, eenvoudigere, techniek met het **rpart** *package*. We zullen de tweede, moeilijkere variant (met het **M5P** *package*) gebruiken voor een mogelijke modelverbetering.

Als simpel voorbeeld kunnen we ons een kleine dataset voorstellen, met gegevens over de winstgevendheid van 10 bedrijven. We kunnen de data splitsen naar twee regio's (noord en zuid), of naar de leeftijd van het bedrijf (korter of langer dan 5 jaar).

We lezen de gegevens in uit het bestand **ML5_OEFEN3.csv**. Voor 10 bedrijven zijn de winsten (in bijvoorbeeld duizenden euro's) oplopend gesorteerd weergegeven. Regio is gecodeerd als 0/1, maar voor het gemak hebben we er 1 en 2 van gemaakt, door er 1 bij op te tellen. Leeftijd is ook gecodeerd als 1 en 2 (voor jonger of ouder dan 5 jaar). De lengte van het bestand (10 records) is opgeslagen in **vl**.

```
> # voorbeeld
>
> (v <- read.csv("ML5_OEFEN3.csv", header=TRUE))
   bedrijf winst regio leeftijd
1        1     4     0        1
2        2    10     1        1
3        3    18     0        1
4        4    27     1        1
5        5    35     0        1
6        6    42     1        2
7        7    45     0        2
8        8    45     0        1
9        9    72     1        2
10      10    75     1        2
> v$regio <- v$regio + 1; v
   bedrijf winst regio leeftijd
1        1     4     1        1
2        2    10     2        1
3        3    18     1        1
4        4    27     2        1
5        5    35     1        1
6        6    42     2        2
7        7    45     1        2
8        8    45     1        1
9        9    72     2        2
10      10    75     2        2
> (vl <- length(v$winst))
[1] 10
```

In het bestand ziet u dat regio=1 over de hele linie van de op winst gesorteerde records voorkomt. De winst van bedrijven in regio 1 loopt uiteen van 4 tot 45; en van bedrijven in regio 2 van 10 tot 72). Voor leeftijd is er een duidelijker patroon: oudere bedrijven zitten vooral aan de bovenkant van de verdeling (winsten van 42 tot 75), en jongere bedrijven aan de onderkant (winsten van 4 tot 45). Een splitsing naar leeftijd zal daarom leiden tot homogenere groepen, dan een splitsing naar regio!

In het algoritme wordt gezocht naar die splitsingen in de data set die leiden tot homogene groepen. De splitsingen (of vertakkingen van de boom) stoppen als vrijwel geen winst meer te behalen is.

Het criterium voor homogeniteit is gebaseerd op de standaarddeviatie. Hoe lager de standaarddeviatie binnen een groep is, des te homogener de groep. Het stuurcriterium voor splitsing is de vermindering van de standaarddeviatie (of SDR, *Standard Deviation Reduction*). In een formule:

$$SDR = sd(X) - \sum_i \frac{N_i}{N} * sd(X_i)$$

In deze formule geldt:

SDR	*Standard Deviation Reduction*
sd()	Standaarddeviatie
N; N_i	Het aantal waarnemingen (totaal; in groep *i*)
X	De vector met waarnemingen van de ongesplitste groep
X_i	De vector met waarnemingen van groep *i*

De formule ziet er natuurlijk erg ingewikkeld uit, maar is eigenlijk eenvoudig uit te leggen.

Aan de rechterkant van het "is gelijk" teken staat de standaarddeviatie van de ongesplitste data set. Daarvan trekken we af de som (Σ) van de gewogen standaarddeviaties van de groepen. De gewichten zijn het aantal waarnemingen in iedere groep ten opzichte van het totale aantal waarnemingen.

- Als de gevormde groepen perfect homogeen zijn, dan is de standaarddeviatie binnen die groepen gelijk aan 0. De vermindering in de standaarddeviatie is dan volledig: alle variantie (namelijk de variantie van de ongesplitste data set) is verdwenen!
- Als de gevormde groepen even heterogeen zijn als de ongesplitste data set, dan zal de variantie nagenoeg even groot zijn in de nieuwe als in de oude situatie, en is de vermindering (SDR) ongeveer gelijk aan 0.

In ons voorbeeld zal de splitsing naar leeftijd te leiden tot homogenere groepen, en zal de SDR dus groter zijn.

Het aantal waarnemingen in de ongesplitste data set en in de groepen, wijze we toe aan **vl**, **r1l**, **r2l**, **l1l** en **l2l** (bijvoorbeeld, het aantal waarnemingen in de eerste leeftijdscategorie (**l**=1), is **l1l**=6.

We kunnen de berekeningen zelf uitvoeren met de standaardfuncties van **R**: de **`sd()`**, voor standaarddeviatie, en de **`length()`** functie, voor het aantal waarnemingen.

De vermindering van de standaarddeviatie is, zoals verwacht, aanzienlijk groter bij een splitsing naar leeftijd!

```
> (v <- read.csv("ML5_OEFEN3.csv", header=TRUE))
   bedrijf winst regio leeftijd
```

[output weggelaten]

```
> v$regio <- v$regio + 1; v
   bedrijf winst regio leeftijd
```

[output weggelaten]

```
> (vl <- length(v$winst))
[1] 10
> (r1 <- v[v$regio==1,"winst"]); (r1l <- length(r1))
[1]  4 18 35 45 45
[1] 5
> (r2 <- v[v$regio==2,"winst"]); (r2l <- length(r2))
[1] 10 27 42 72 75
[1] 5
>
> (l1 <- v[v$leeftijd==1,"winst"]); (l1l <- length(l1))
[1]  4 10 18 27 35 45
[1] 6
> (l2 <- v[v$leeftijd==2,"winst"]); (l2l <- length(l2))
[1] 42 45 72 75
[1] 4
> sdr.r <- sd(v$winst) - (r1l/vl * sd(r1) + r2l/vl * sd(r2))
> sdr.l <- sd(v$winst) - (l1l/vl * sd(l1) + l2l/vl * sd(l2))
```

```
> sdr.r
[1] 0.7111182
> sdr.l
[1] 7.56093
```

Stap 1 en 2: het inlezen en verkennen van de data

Als voorbeeld gaan we gebruikmaken van de gegevens van een vereniging van kleine ondernemingen. De vereniging heeft als doel om de prestaties van haar leden te bevorderen, onder andere door optimaal gebruik te maken van de mogelijkheden van mobiele technologie en van Internet. Veel van de leden zijn al van middelbare leeftijd, en hebben niet altijd de tijd en de mogelijkheden om bij te blijven op het gebied van technologie. De vereniging beschikt over de volgende gegevens:

- Een indicatie van de relatieve prestatie van de onderneming (de prestatie van het bedrijf ten opzichte van concurrerende bedrijven): **busperf**. Deze variabele is gemeten op een 5-puntsschaal (1=zeer slecht, 5=zeer goed);
- Indicatoren voor het gebruik van mobiele technologie en Internet (gemeten op een 5-puntsschaal, 1=zeer weinig; 5=zeer veel)
  - **calls** (aantal gesprekken per mobiele telefoon);
  - **sms** (aantal verstuurde en ontvangen SMS-berichten);
  - **telebank** (intensiteit van ontvangen en betaalde bedragen, met mobiel telebankieren);
  - **internet** (intensiteit van gebruik van mobiel Internet)
- Regio (**region;** 0=Oost; 1=West)
- Werkgebied (**internat**; 0=lokaal; 1=(inter)nationaal);
- Hoogst genoten opleiding (**educ;** 1=lagere school; 2=middelbaar onderwijs; 3=universitair);
- Inkomensindicatie (**incgroup**; 1=laagste 20%, 5=hoogste 20%).

Met de **`summary()`** en **`str()`** functies vatten we de informatie over de 200 ondernemers in de data set samen.

```
> busperf <- read.csv("ML5_OEFEN4.csv", header=TRUE)
> summary(busperf)
      rec              region       internat         calls      
 Min.   :  1.00   Min.   :0.0   Min.   :0.000   Min.   :1.000  
 1st Qu.: 51.75   1st Qu.:0.0   1st Qu.:0.000   1st Qu.:1.000  
 Median :101.50   Median :0.5   Median :0.000   Median :1.000  
 Mean   :101.42   Mean   :0.5   Mean   :0.435   Mean   :1.485  
 3rd Qu.:151.25   3rd Qu.:1.0   3rd Qu.:1.000   3rd Qu.:2.000  
 Max.   :201.00   Max.   :1.0   Max.   :1.000   Max.   :5.000  
      sms            telebank        internet        busperf     
 Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000  
 1st Qu.:2.000   1st Qu.:2.000   1st Qu.:1.000   1st Qu.:3.000  
 Median :2.000   Median :4.000   Median :1.000   Median :3.000  
 Mean   :1.885   Mean   :3.175   Mean   :1.535   Mean   :3.145  
 3rd Qu.:2.000   3rd Qu.:4.000   3rd Qu.:2.000   3rd Qu.:4.000  
 Max.   :5.000   Max.   :5.000   Max.   :4.000   Max.   :5.000  
      educ          incgroup    
 Min.   :1.000   Min.   :1.000  
 1st Qu.:1.000   1st Qu.:2.000  
 Median :2.000   Median :3.000  
 Mean   :1.875   Mean   :3.005  
 3rd Qu.:3.000   3rd Qu.:4.000  
 Max.   :3.000   Max.   :5.000  
> str(busperf)
```

```
'data.frame':   200 obs. of  10 variables:
 $ rec     : int  1 2 3 4 5 6 7 8 9 10 ...
 $ region  : int  1 0 0 1 0 1 0 0 0 1 ...
 $ internat: int  0 0 1 1 0 0 0 1 1 1 ...
 $ calls   : int  1 3 1 2 1 1 2 3 1 3 ...
 $ sms     : int  1 1 2 2 2 2 2 2 2 2 ...
 $ telebank: int  3 1 5 4 2 4 2 5 5 4 ...
 $ internet: int  1 1 1 2 1 2 1 1 3 2 ...
 $ busperf : int  3 3 3 3 3 2 2 3 1 1 ...
 $ educ    : int  2 1 1 3 1 3 1 1 3 1 ...
 $ incgroup: int  2 3 2 5 1 3 4 1 5 5 ...
```

Ons doel is om met behulp van een regressieboom inzichtelijk te maken wat het verband is tussen de verklarende variabelen, en **busperf** als de te verklaren variabele.

Stap 3: het trainen van het model

We willen aan de hand van een trainingset een model (regressieboom) bouwen, en vervolgens met een testset kijken hoe goed het model werkt.

We splitsen de data in een groep van 80% (160 records) voor training, en 20% (40 records) voor de test. Omdat we er op hebben toegezien dat de records in willekeurige volgorde zijn gesorteerd, kunnen we de data splitsen in de eerste 160 records voor de training set, en de laatste 40 records voor de test set.

We hebben 10 variabelen in het databestand. De eerste kolom bevat een volgnummer voor de onderneming, en is voor de analyse niet van belang. Met “2:10” lezen we alleen de overige 9 variabelen in.

```
> bp_train <- busperf[1:160,2:10]
> bp_test  <- busperf[161:200,2:10]
> head(bp_train); tail(bp_test)
  region internat calls sms telebank internet busperf educ incgroup
1      1        0     1   1        3        1       3    2        2
2      0        0     3   1        1        1       3    1        3
3      0        1     1   2        5        1       3    1        2
4      1        1     2   2        4        2       3    3        5
5      0        0     1   2        2        1       3    1        1
6      1        0     1   2        4        2       2    3        3
    region internat calls sms telebank internet busperf educ incgroup
195      0        0     2   2        5        1       3    1        2
196      1        1     3   3        4        4       4    3        4
197      0        0     1   2        1        1       3    1        1
198      0        0     1   2        1        1       3    1        2
199      0        0     1   2        4        3       3    1        3
200      1        0     2   2        4        1       2    3        5
```

Bron: OGN.

Voor het bouwen en het testen van het model, gaan we gebruikmaken van een tweetal *packages*.

Het eerste *package* is **rpart**, en heeft functies voor het bouwen en weergeven van de regressieboom. Voor mooiere grafische weergave gebruiken we het *package* **rpart.plot**.

Het installeren van de *packages*, en het aanroepen van de functies van deze *packages*, zal geen probleem meer vormen!

```
# install.packages("rpart") [eenmalig!]
library(rpart)
# install.packages("rpart.plot")
library(rpart.plot)
```

Met de **rpart()** functie bouwen we de regressieboom. Merk op dat de structuur erg lijkt op die voor een regressieanalyse. Met de tilde "~" geven we aan welke variabele (aan de linkerkant van de tilde) wordt verklaard uit welke andere variabelen. Aan de rechterkant van de tilde kunnen we alle verklarende variabelen aangeven (net zoals bij regressieanalyse gescheiden door een "+"). We kunnen kortweg met een punt (".") aangeven, dat "alle andere variabelen" (dat wil zeggen, alle variabelen in de data set behalve de te verklaren variabele aan de linkerkant van de tilde, **busperf**) moeten worden gebruikt.

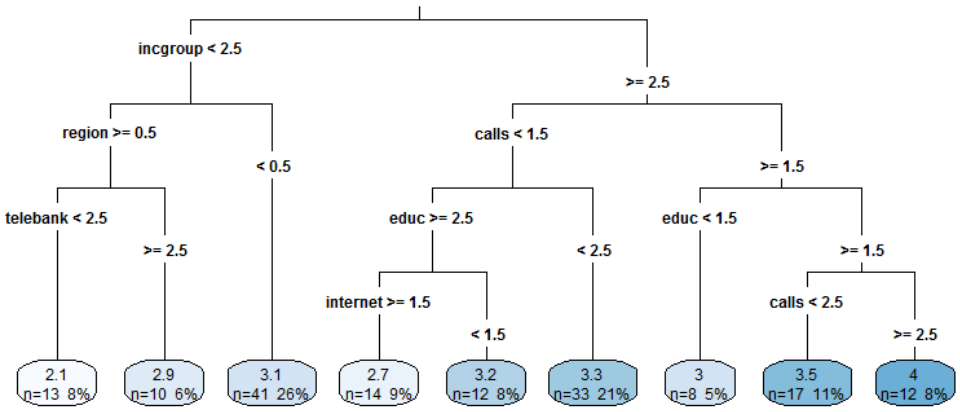
```
bp.model <- rpart(busperf ~ ., data = bp_train)
bp.model
summary(bp.model)
```

```
> bp.model
n= 160

node), split, n, deviance, yval
      * denotes terminal node

 1) root 160 144.243800 3.131250
   2) incgroup< 2.5 64  53.734380 2.859375
     4) region>=0.5 23  13.652170 2.434783
       8) telebank< 2.5 13   6.923077 2.076923 *
       9) telebank>=2.5 10   2.900000 2.900000 *
     5) region< 0.5 41  33.609760 3.097561 *
   3) incgroup>=2.5 96  82.625000 3.312500
     6) calls< 1.5 59  53.627120 3.152542
      12) educ>=2.5 26  24.961540 2.961538
        24) internet>=1.5 14  14.857140 2.714286 *
        25) internet< 1.5 12   8.250000 3.250000 *
      13) educ< 2.5 33  26.969700 3.303030 *
     7) calls>=1.5 37  25.081080 3.567568
      14) educ< 1.5 8   8.000000 3.000000 *
      15) educ>=1.5 29  13.793100 3.724138
        30) calls< 2.5 17   8.235294 3.529412 *
        31) calls>=2.5 12   4.000000 4.000000 *
> summary(bp.model)

[output omitted]
```

De samenvattende informatie (zeker die uit de **summary()**) functie is niet gemakkelijk te lezen. De belangrijkste informatie is de informatie die ook met **rpart.plot()** kan worden verkregen, en daarom gaan we meteen naar de visualisatie.

De functie heeft veel handige opties. Met "**fallen.leaves=TRUE**" verschijnen de bladeren van de boom op overzichtelijke manier aan de onderkant van de grafiek (dat is soms niet handig bij complexe modellen met veel vertakkingen). Met "**extra=101**" vragen we om zowel aantallen respondenten als om percentages (bij grote bestanden is het beter alleen het percentage weer te geven).

```
rpart.plot(bp.model, fallen.leaves=TRUE, type=3, extra=101)
```



Figuur 9. Illustratie van de bladeren van de boom. *Bron: OGN.*

De grafiek is goed te interpreteren.

Vanaf boven beginnend, wordt eerst (op grond van het SDR-algoritme), de dataset gesplitst naar inkomen. De ondernemers met lage inkomens hebben verwachte prestaties tussen de 2.1 en 3.1 (op een schaal van 1 tot 5). Ondernemers in de regio "<0.5" (dat kan alleen 0, dus het oosten zijn!) presteren beter. In de regio west wordt nog een verdere uitsplitsing gemaakt naar telebankieren: ondernemers die gebruik maken van telebankieren, scoren 0.8 punt hoger.

Aan de rechterkant van de grafiek zien we dat het profiel van de best presterende ondernemers is samen te vatten als: hoger inkomen, intensief gebruik van de mobiele telefoon, en hoger opgeleid. Een opvallende uitkomst is dat intensief gebruik van internet nauwelijks in de beslissingsboom voorkomt. We zien het alleen terug in de midden-categorieën, en daar gaat intensief gebruik van internet juist gepaard met lagere prestaties.

Stap 4: het testen van het model

Anders dan bij regressieanalyse, testen we hier geen vooraf gespecificeerd model. Ons model is exploratief en beschrijvend. Of het model ook goed werkt, willen we uitzoeken aan de hand van hetzelfde model maar dan toegepast op de testset. We gebruiken hiervoor het **`predict()`** commando.

```
> bp.pred <- predict(bp.model, bp_test)
> summary(bp.pred)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.077   3.000   3.098   3.153   3.303   4.000
> summary(bp_test$busperf)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    1.0     3.0     3.0     3.2     4.0     5.0
> cor(bp.pred, bp_test$busperf)
[1] 0.5347431
```

Het **`predict()`** commando vraagt als enige input, tussen haakjes, om het voor de trainingset gemaakte model, en om de naam van de test set (40 records). De **`summary()`** van het model geeft informatie over de verdeling van de voorspelde waarde van **busperf**. Die informatie is niet nieuw, want ook uit de grafiek weten we al dat de voorspelde waren voor de trainingset uiteenlopen van 2.1 tot 4! Aangezien de testset een willekeurige trekking is van 40 waarden uit de dataset, is het waarschijnlijk dat de voorspellingen in hetzelfde bereik liggen.

Een veel voorkomende uitkomst is dat regressiemodellen en regressiebomen moeite hebben om de extreme waardes aan de onderkant en de bovenkant van de verdeling te voorspellen. Het bereik

van de voorspelde waardes loopt van 2.1 tot 4, maar in de werkelijke verdeling komen waardes voor van 1 tot 5. Het verbeteren van het model zou zich mede kunnen richten op extra variabelen die de extreme waardes bepalen.

Omdat onze te verklaren variabele een numerieke variabele is, kunnen we de correlatie tussen de werkelijke waarde en de voorspelde waarde gebruiken als maatstaf voor de voorspelkracht van het model. De correlatie is een getal tussen -1 en +1. Een waarde van 0 betekent dat er geen enkel verband is tussen de twee variabelen, en een waarde van +1 (-1) duidt op een perfect positieve (negatieve) correlatie. De correlatie op basis van de voorspelde en werkelijke waarden in de test set bedraagt 0.53. Dat niet heel goed maar ook niet heel slecht.

Stap 5: het verbeteren van het model

Een mogelijke verbetering voor regressieanalyses in het algemeen, en dus ook voor regressiebomen, is het toevoegen van meer variabelen. Omdat het toevoegen van variabelen altijd leidt tot een verbetering van het model, is het verstandig het principe van spaarzaamheid te respecteren: voeg geen variabelen toe die tot marginale verbeteringen leiden, en waarvan het onduidelijk is waarom ze gerelateerd zijn aan de te verklaren variabele!

Een tweede optie is gebruik te maken van meer geavanceerde algoritmes. In het algoritme dat we hierboven hebben gebruikt, hebben we de dataset op een slimme manier uitgesplitst, en aan de "bladeren van de boom" een schatting gegeven van de gemiddeldes van de groep.

In plaats van een gemiddelde van de groep, kunnen we voor het blad van de boom ook een regressieschatting maken, aan de hand van de waardes van de verklarende variabelen. We maken dan in feite gebruik van meer informatie: waar we, bijvoorbeeld, in de grafiek hierboven, de gemiddelde prestatie (2.1) berekenen van ondernemers met een inkomen<2.5, in regio=1, en met telebankieren<2.5, schatten we nu een regressiemodel met de exacte waarde voor inkomen en telebankieren (regio kan maar een waarde hebben, voor deze groep).

Gebruik van meer informatie zal leiden tot betere resultaten.

We maken gebruik van het **RWeka** *package*. Omdat de procedure nagenoeg hetzelfde is, lopen we er snel doorheen. We concentreren ons op de samenvattende statistieken van de voorspelling en op de correlatie. De functie die we gebruiken is **`M5P()`**. **M5P** is de naam van een algoritme.

```
# install.packages("RWeka")
library(RWeka)
bp.model2 <- M5P(busperf ~ ., data = bp_train)
bp.pred2  <- predict(bp.model2, bp_test)
summary(bp.pred2)
cor(bp.pred2, bp_test$busperf)
```

```
> bp.model2 <- M5P(busperf ~ ., data = bp_train)
> bp.pred2  <- predict(bp.model2, bp_test)
> summary(bp.pred2)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.689   3.006   3.168   3.159   3.330   3.895
> cor(bp.pred2, bp_test$busperf)
[1] 0.5659919
```

Hoewel de correlatie tussen de voorspelde en werkelijke waarde in de test set iets hoger is dan voor het **rpart** algoritme, zien we ook dat de voorspellingen zich nu in een kleinere range bevinden (van 2.69 tot 3.90). Het was al moeilijk om extreme waarden te voorspellen, maar met het nieuwe algoritme geldt dat nog sterker! Welk model wordt gekozen, hangt af van wat u als analist belangrijker vindt. In dit geval blijkt het eenvoudigere model te leiden tot goed interpreteerbare resultaten die niet veel onderdoen voor die van een complexere methode. Om die reden is het beter om het bij het eerste model te houden.