Kansverdelingen en objecten

drs. Sytse Knypstra *

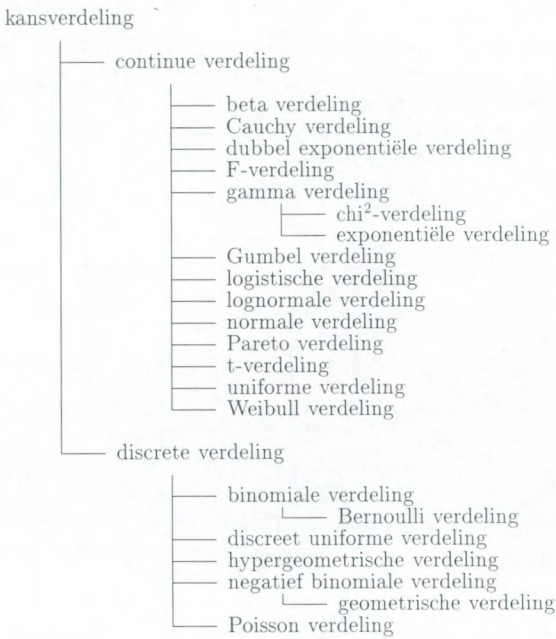
Samenvatting

Computerprogramma's waarin de eigenschappen van kansverdelingen worden vastgelegd, kunnen elegant worden geprogrammeerd in een programmeertaal die *Object Oriented Programming* (OOP) toelaat. In dit artikel wordt beschreven wat de uitgangspunten van OOP zijn en waarom in het geval van kansverdelingen de aanpak met OOP aanbeveling verdient. De auteur stelt een Borland Pascal *unit* beschikbaar waarin 22 kansverdelingen zijn opgenomen.

* Vakgroep Econometrie, Rijksuniversiteit Groningen, postbus 800, 9700 AV Groningen, email: S.Knypstra@eco.rug.nl

1 Inleiding

Sinds 1985 is er bij de Vakgroep Econometrie van de RuG computerondersteund onderwijs ontwikkeld voor het vak statistiek. De bijbehorende computerprogramma's zijn geprogrammeerd met behulp van *Object Oriented Programming* (OOP) technieken in Turbo/Borland Pascal. Zij bevatten onder meer functies waarmee men kansen en kwantielen van kansverdelingen kan bepalen en aselecte trekkingen uit verdelingen kan genereren. Hierbij gaat het om de verzameling verdelingen uit figuur 1.



Figuur 1: Boomstruktuur kansverdelingen.

In paragraaf 2 beschrijven we hoe men kansverdelingen kan representeren met behulp van objecten en in paragraaf 3 bespreken we de belangrijkste eigenschappen en voordelen van OOP. De voorbeelden worden geformuleerd in Borland Pascal.

2 Kansverdelingen en objecten

In figuur 1 zijn de kansverdelingen weergegegeven in de vorm van een boomstruktuur. Het meest algemene element, links bovenaan, is 'kansverdeling'. Als we een kansverdeling iets meer specificeren, dan onderscheiden we discrete en (absoluut) continue verdelingen.

Binnen ieder van deze twee kategoriën kunnen we een reeks verdelingen opnoemen, waarbij sommige weer speciale gevallen van andere verdelingen zijn. Zo zijn bijvoorbeeld de chi^2- en de exponentiële verdeling speciale gevallen van de gamma-verdeling.

Gegevens en functies. Vanaf versie 5.5 kan men in Turbo/Borland Pascal technieken gebruiken die zijn gebaseerd op objecten; dit zijn strukturen die zowel *gegevens* bevatten als *functies en procedures* die op die gegevens werken. De functies en procedures van een object worden ook wel zijn *methoden* genoemd. Ook bij kansverdelingen kunnen we gegevens (bijvoorbeeld de parameterwaarden) en functies (kans, kansdichtheid, verdelingsfunctie, enzovoorts) onderscheiden. Kansverdelingen kunnen dus gedefinieerd worden als objecten. De gamma-verdeling en de exponentiële verdeling zouden dan als volgt kunnen worden gedeclareerd:

```
{1} gamma_verdeling = object(continue_verdeling)        { nieuw object }
{2}   alpha,lambda:real;                                    { gegevens }
{3}   constructor init(newalpha,newlambda:real);       { initialisatie }
{4}   function dichtheid(x:real):real;          { kansdichtheid f in x }
{4}   function cum_verd(x:real):real;       { verdelingsfunctie F in x }
{4}   function trekking:real;                { geeft aselecte trekking }
    end;

{1} exponentiele_verdeling = object(gamma_verdeling)    { nieuw object }
{2}   theta:real;                                           { gegeven }
{3}   constructor init(newtheta:real);                 { initialisatie }
    end;
```

In de regels {1} definiëren we een nieuw object dat een speciaal geval is van het één niveau hoger geplaatste object uit de hiërarchie. In {2} zijn de gegevens gedefinieerd die bij het nieuwe object horen, en in {3} en {4} de methoden. Hiervan is {3} een zogenaamde constructor procedure die men doorgaans de naam `init` geeft en die het object initialiseert.

Boomstruktuur en erfelijkheid. Een afgeleid object zoals `exponentiele_verdeling`, neemt in principe alle gegevens en methoden over van zijn voorganger (`gamma_verdeling`). Zonder dat de functie `cum_verd` expliciet voor `exponentiele_verdeling` is gedefinieerd, kan hij toch worden gebruikt:

```
expon.init(2);      { initialiseer exponentiele verdeling met theta=2 }
writeln('P(X>3)=',1-expon.cum_verd(3));           { bereken een kans }
```

De relaties tussen verdelingen uit figuur 1 kunnen we dus overbrengen naar relaties tussen objecten, waarbij een vertakking betekent: 'speciaal geval van'. Omdat een boomstruktuur lijkt op een familiestamboom worden in de OOP-wereld vaak termen gebruikt die aan familierelaties zijn ontleend. De eigenschap van objecten om gegevens en methoden over te nemen van hun 'ouders' heet dan ook wel 'erfelijkheid'. Een 'kind' kan echter ook de geërfde eigenschappen veranderen en/of nieuwe gegevens en methoden toevoegen.

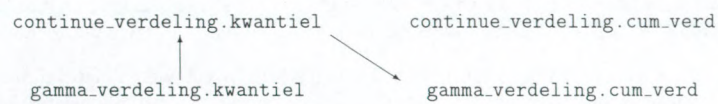
Kwantiel. Nog niet genoemd is de `function kwantiel(q:real):real`. Deze berekent voor een continue kansverdeling op iteratieve wijze bij een gegeven kans `q` het nulpunt van de niet-dalende functie `G(x) = cum_verd(x)-q`. Net als bij de functie `rtsafe` in [PRESS et al.] worden de Newton-Raphson- en de bisectiemethode gecombineerd (zie figuur 2). Voor de Newton-Raphsonmethode hebben we behalve de functie `G(x)` ook nog haar afgeleide nodig; dit is de kansdichtheid `dichtheid(x)`. Het algoritme werkt voor

```
{  eerst startwaarden bepalen: voor x, x_onder en x_boven geldt steeds }
{  dat x_onder <= x <= x_boven   en   G(x_onder) < 0 < G(x_boven)       }
  repeat
    G:=cum_verd(x)-q; pdf:=dichtheid(x);
    if G<=0 then x_onder:=x; if G>=0 then x_boven:=x;
    if (x_boven-x_onder)*pdf<=abs(G)    { nieuwe punt buiten interval? }
    then
      begin
        dx:=(x_boven-x_onder)/2; x:=x_onder+dx;              { bisectie }
      end
    else
      begin
        dx:=G/pdf; x:=x-dx;                                { Newton-Raphson }
      end;
  until abs(dx) < eps;
  kwantiel:=x;
```

Figuur 2: Algoritme voor de bepaling van het kwantiel van een continue verdeling.

iedere continue verdeling, mits de functies `cum_verd` en `dichtheid` correct gedefinieerd zijn. Het ligt dus voor de hand de functie `kwantiel` reeds te definiëren bij het object `continue_verdeling` en alle nakomelingen deze functie te laten erven.

Virtuele methoden. Bij aanroep van bijvoorbeeld `gamma_verdeling.kwantiel` wordt dan de geërfde functie `continue_verdeling.kwantiel` toegepast (zie figuur 3). Op zijn beurt heeft deze de functies `cum_verd` en `dichtheid` nodig, maar dan wel van `gamma_verdeling`! Om er voor te zorgen dat `continue_verdeling.kwantiel` altijd



Figuur 3: Referentieschema bij virtuele functies.

de juiste functies gebruikt, moeten we `dichtheid` en `cum_verd` definiëren als *virtual*. Niet-virtuele methoden heten statisch.

3 Voordelen van OOP

Het gebruik van OOP bij het programmeren van kansverdelingen heeft een aantal belangrijke voordelen, met name als er later nieuwe verdelingen moeten worden toegevoegd.
Uitbreidingen. Doordat afgeleide objecten in principe de eigenschappen van hun ouder erven, kan men het systeem van kansverdelingen gemakkelijk uitbreiden. Als je een nieuwe kansverdeling definieert als 'kind' van `continue_verdeling` hoef je slechts enkele gegevens aan te passen en methoden te herschrijven (de functie `cum_verd` bijvoorbeeld).
Onderhoud. Bij verschillende objecten kun je methoden met dezelfde naam gebruiken (polymorfisme): `verdeling.trekking` levert altijd een trekking uit de juiste verdeling op, of `verdeling` nu van het type `gamma_verdeling` of `Poisson` is. Eenvoud in naamgeving, maar flexibiliteit van de inhoud van methoden bevordert de leesbaarheid, de consistentie en daardoor een gemakkelijker onderhoud van de programmatuur.
Overdraagbaarheid. Ook als men niet over de brontekst, maar slechts over een gecompileerde versie van de verzameling kansverdelingen/objecten beschikt, kan men toch op eenvoudige wijze zelf nieuwe kansverdelingen toevoegen. Een nieuwe verdeling definieert men bijvoorbeeld als een speciaal geval van `continue_verdeling` en daarmee erft men een hele infrastruktuur.
Robuustheid. OOP maakt het mogelijk om data te 'verpakken' in objecten. De toegang tot de data is dan alleen mogelijk via de methoden van het bijbehorende object. Kiest men men op een bepaald moment een andere datastruktuur, dan heeft dit alleen konsekwenties voor de methoden die de toegang tot de data regelen. Voor andere delen van de programmatuur blijft de wijziging 'onzichtbaar'. Deze aanpak heet data inkapseling of data abstraktie. Het maakt het wijzigen van de programma's gemakkelijker.

4 Tenslotte

In dit artikel wordt getoond hoe men OOP technieken kan toepassen op het programmeren van (eigenschappen van) kansverdelingen. Daarbij biedt OOP enkele voordelen boven de traditionele manier van programmeren. OOP laat een struktuur toe die overeenkomt met de boomstruktuur (figuur 1) die tussen kansverdelingen bestaat. De eigenschappen van erfelijkheid, polymorfisme en de mogelijkheid van data inkapseling maken dat de programmatuur gemakkelijk is te onderhouden en uit te breiden, dit laatste ook als men slechts over een gecompileerde *unit* beschikt.
De auteur stelt twee met Borland Pascal for Windows (versie 7.0) gecompileerde *units* beschikbaar met kansverdelingen die veel meer methoden bevatten dan de voorbeelden in dit artikel. Zo zijn er functies die de verwachting en de variantie opleveren, die vaststellen of een punt tot de drager behoort, enzovoorts. Ze zijn, samen met een beschrijving en drie programma's (PCalc, Sila en PQRS) die erop zijn gebaseerd, beschikbaar via de URL: `http://www.eco.rug.nl/medewerk/knypstra/knypnl.html`.

Referenties

DEVROYE, L. (1986) *Non-Uniform Random Variate Generation.* Springer, New York.
PRESS, W.H., B.P.FLANNERY, S.A. TEUKOLSKY, W.T.VETTERLING (1989) *Numerical Recipes in Pascal: the art of scientific programming.* Cambridge University Press, Cambridge.

Ontvangen: 22-12-1995
Geaccepteerd: 4-9-1996