Zelfstudieopdracht - R, RStudio en RMarkdown

In deze zelfstudieopgave raak je bekend met R en R Markdown. Je hoeft de opdracht niet in te leveren.

1. Van start gaan

Alvorens te beginnen, moet je een RStudio Cloud account hebben gemaakt, of R en RStudio gedownload en geïnstalleerd hebben op je eigen computer. Zie Brightspace voor instructies.

Open nu het bestand Introductie_R.Rmd in RStudio, als je dat nog niet gedaan had. Als je vervolgens op "Knit PDF" in de balk boven de file klikt (misschien staat er aanvankelijk "Knit HMTL"; je kunt "PDF" kiezen met de muis), zet RStudio de .Rmd om in een .pdf file. Dit is precies de file die je nu leest.

2. Voorbeelden

Ter introductie een aantal voorbeelden. Voor de meeste bekende verdelingen is in R alvast de *dichtheidsfunctie* f, de *cumulatieve verdelingsfunctie* $x \mapsto F(x) = \mathbb{P}(X \leq x)$, en de *kwantielfunctie* $\alpha \mapsto F^{-1}(\alpha)$ geimplementeerd. Het betreffende commando begint respectievelijk met een `d`, `p` of `q`. We illustreren dit nu voor een normale verdeling met parameters $\mu = 2$ en $\sigma = 3$. Het eerste argument is altijd het argument van de functie, gevolgd door de parameters. Hieronder staat wat R code in grijze blokken, zogeheten 'chunks'.

```
dnorm(0.5, 2, 3)
```

geeft de dichtheid in $x = 0.5$, en is dus gelijk aan

$$\frac{1}{\sqrt{2\pi 3^2}} e^{-\frac{(0.5-2)^2}{2\cdot 3^2}}.$$

Inderdaad vinden we dat

```
dnorm(0.5, 2, 3)-1/sqrt(2*pi*3^2)*exp(-(0.5-2)^2/(2*3^2))
```

```
## [1] 0
```

```
pnorm(0.5, 2, 3)
```

geeft $\mathbb{P}(X \leq 0.5)$ voor $X \sim \mathcal{N}(2, 9)$.

```
pnorm(0.5, 2, 3, lower.tail = FALSE)
```

geeft $\mathbb{P}(X \geq 0.5)$, wat gelijk is aan $1 - \mathbb{P}(X \leq 0.5)$.

```
qnorm(0.5, 2, 3)
```

geeft de waarde van x zodanig dat $\mathbb{P}(X \leq x) = 0.5$. In dit geval is deze waarde gelijk aan de verwachtingswaarde, dus twee.

Je kunt een steekproef genereren door `r` voor de naam van de verdeling te zetten, bijvoorbeeld

```
x <- runif(10,0,1)
```

maakt een steekproef met tien trekkingen uniform uit het interval $[0, 1]$, of

```r
x <- rnorm(100, 2, 3)
```

maakt een steekproef met honderd trekkingen uit een normale verdeling met verwachting $\mu = 2$ en standaarddeviatie $\sigma = 3$. De steekproef bevindt zich nu in de vector `x`, en je kan de waarden zien door te typen

```r
x
```

```
##   [1] -1.16592565  2.70205105 -2.39052007  0.47884388  0.22451767  2.24653855
##   [7]  3.00992874  1.09935214  4.47572461 -0.99709453  1.67013861  6.11680678
##  [13] -1.35706984 -2.26665913  5.15330028  3.58570496 -0.57252546  0.96526137
##  [19] -0.35826109  2.65938489  3.42937097  4.70439770  3.54954116 -2.29478375
##  [25] -2.11277236  1.95359068 -4.26951703  5.45091561  0.16452116  2.24871149
##  [31] -0.01924206  4.58760444  0.60639478  2.03177302  4.54839178  2.12787375
##  [37] -0.54917624  3.12266994  3.99602528 -1.74961305 -0.04329414  5.75849504
##  [43]  1.97224520 -0.13961007  1.76432261 -2.45133016  0.86164015  1.23146998
##  [49] -0.43400503 -3.50427317  7.78336050  0.96352613  0.16312946  0.15042523
##  [55]  2.97028433 -1.60801630  3.28152376 -1.62118923  3.55330076 -5.12014904
##  [61]  0.48559713  6.28358581 -1.42254190  4.53160692  3.84799265  1.72936673
##  [67]  2.76596474 -1.61675843 -0.99531188  1.54605598 -2.05245986 -3.11636923
##  [73]  1.69523597  1.66120055  0.15521174  3.03859971  4.71067944  6.69736992
##  [79]  0.11597058  3.09515794  2.16593109 -0.21955403  0.00365424  2.13320920
##  [85]  2.35122865 -1.41622250  0.60586494  6.21955112 -0.67527261  6.09911353
##  [91]  2.36243445  4.13708064  5.75065406 -0.57134530  0.52984259  6.14072776
##  [97]  3.64004842 -1.22627148  2.16736443  4.85996906
```

wat gegevens verkrijgen door te typen

```r
summary(x)
```

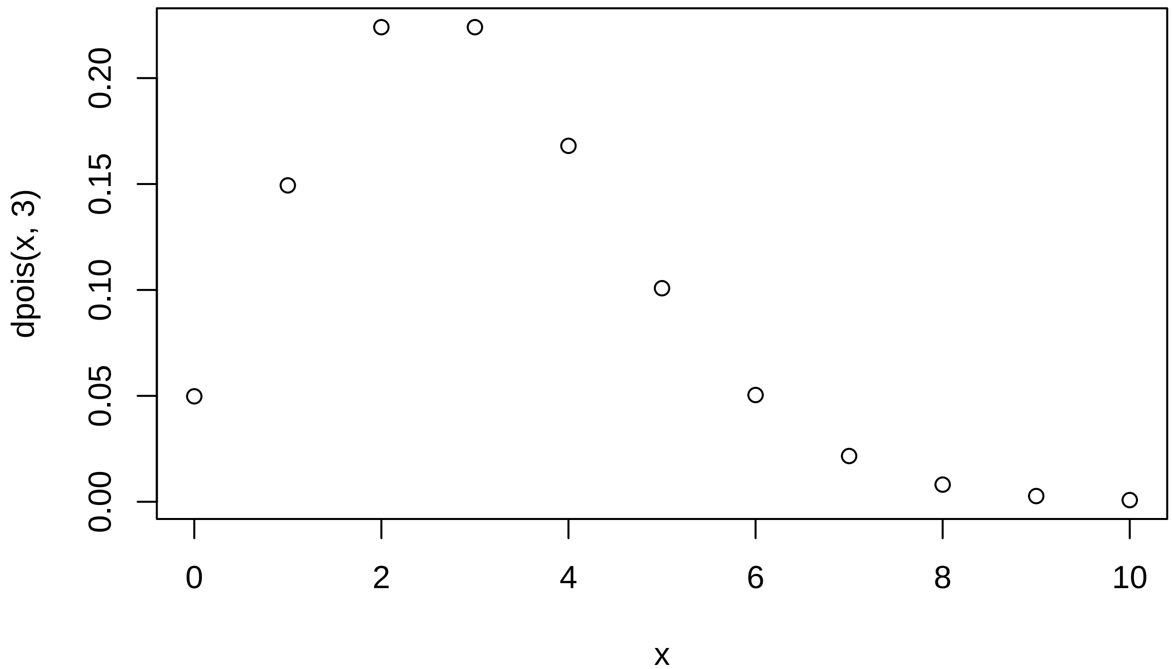
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -5.1201 -0.4628  1.6657  1.5055  3.4594  7.7834
```

en een histogram maken door te typen

```r
hist(x)
```

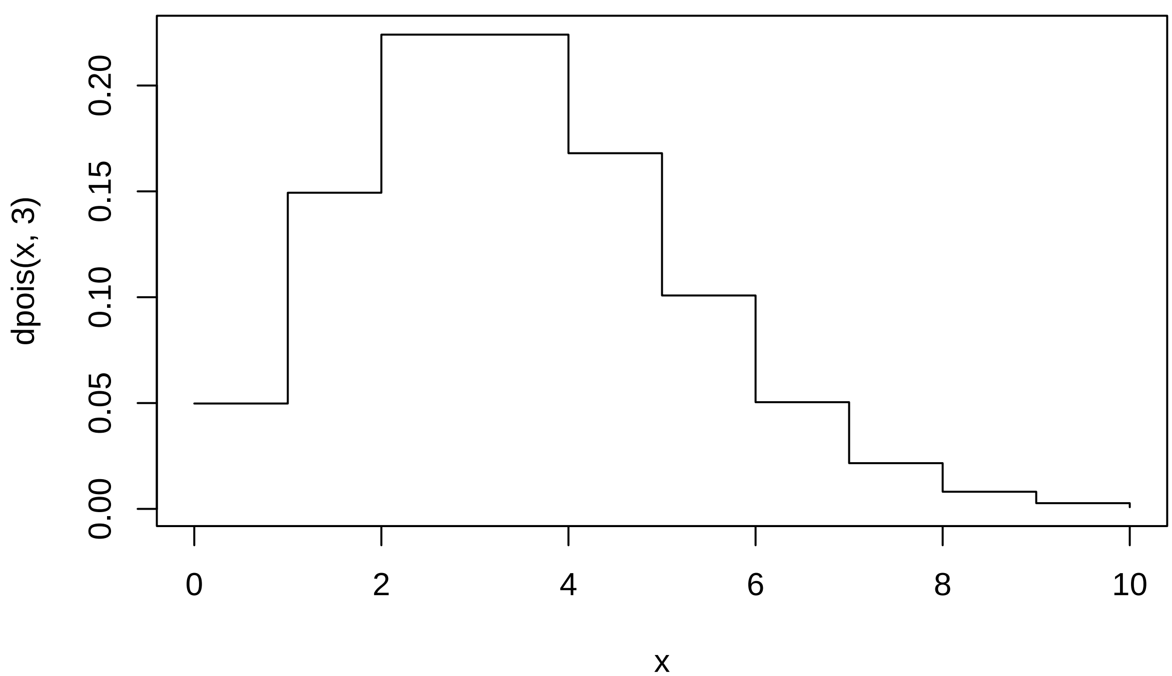
Met R kun je ook plots van functies maken maken. Je kunt bijvoorbeeld eerst een vector `x` maken die de waarden $0, 1, \ldots, 10$ bevat en dan een plot van de Poisson dichtheid met parameter $\lambda = 3$ met

```r
x <- seq(0, 10, by = 1)
plot(x, dpois(x, 3), type = "p")
```



Of misschien is deze mooier?

```
plot(x, dpois(x,3), type = "s")
```



Tot slot laten we zien hoe je een barplot kunt maken. Eerst maken we een steekproef met 100 waarnemingen uit een binomiale verdeling.

```
binom <- rbinom(100, size = 5, p = 0.6)
binom
```

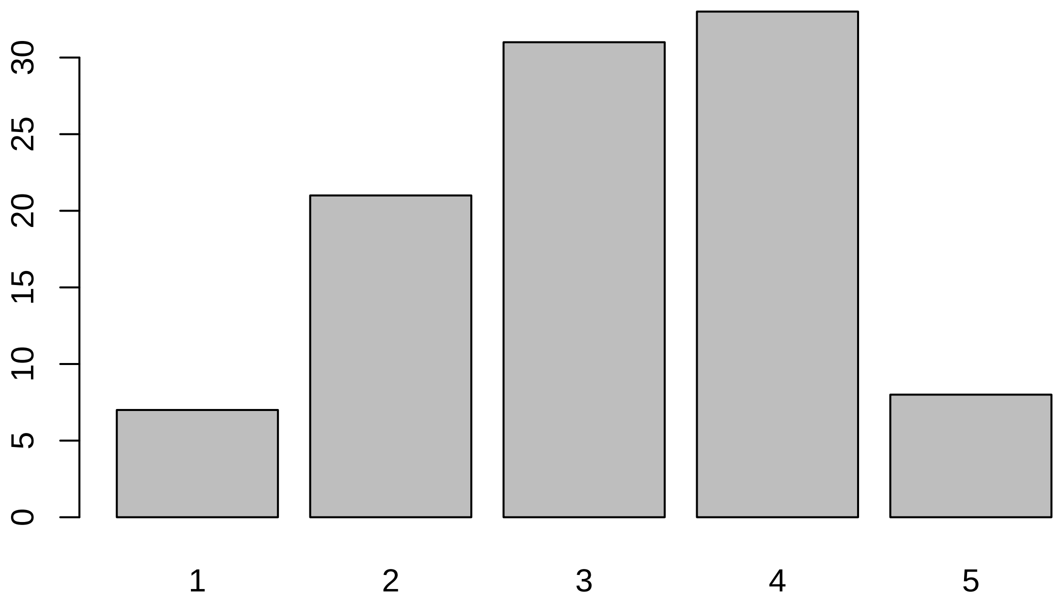
```
##   [1] 3 1 3 4 3 2 4 3 2 2 4 4 3 2 4 4 3 2 3 3 1 4 4 3 3 4 3 2 4 1 5 3 3 5 3 5 3
##  [38] 5 3 4 4 3 2 5 5 3 5 5 3 3 2 4 3 2 3 4 4 2 2 3 2 3 4 4 4 2 2 3 4 1 4 2 4 4
##  [75] 4 3 4 3 4 3 1 3 1 1 4 4 3 4 2 2 4 2 2 4 3 4 4 4 2 2
```

```r
table(binom)
```

```
## binom
##  1  2  3  4  5
##  7 21 31 33  8
```

De tabel geeft aan hoe vaak elke uitkomst is geobserveerd. We visualiseren dit met een barplot.

```r
barplot(table(binom))
```



3. Vragen

Typ je antwoorden op onderstaande vragen onder elke vraag in de .Rmd-file. Niet alle benodigde informatie staat in deze opdracht gegeven, voor een aantal dingen zul je op internet op zoek moeten. Je zult zien dat er veel online hulp voor R beschikbaar is. Daarnaast kan het nuttig zijn om de documentatie te bekijken van R functies die je gaat gebruiken. Typ bijvoorbeeld

```r
help("hist")
```

of

```r
?hist
```

om meer over `hist()` te lezen.

Vraag 1 De toevoeging 'eval=FALSE' die in enkele van bovenstaande chunks staat, is een zogeheten *chunk option*. Wat is het effect van het toevoegen van 'eval=FALSE' tussen de chunk options?

Vraag 2 Zoek op welke chunk options er verder nog zijn, en beschrijf er twee die je handig lijken.

Vraag 3 Voeg hieronder een nieuwe chunk in, maak een histogram van 500 trekkingen uit een standaard normale verdeling, en zorg dat in het geknitte document de R-code niet zichtbaar is, maar het histogram wel.

Vraag 4 Maak drie vectoren met 100 trekkingen uit een normale verdeling met verwachtingswaarde gelijk aan respectievelijk 1, 2, en 3, en standaarddeviatie ook gelijk aan respectievelijk 1, 2 en 3. Bereken het gemiddelde en de variantie van elk van de vectoren.

Vraag 5 Wat is de theoretische verdeling van de som van de drie vectoren uit de vorige vraag?

Vraag 6 Bereken nu met R het gemiddelde en de variantie van de som van de drie vectoren uit vraag 5.

Vraag 7 Maak een vector van lengte 50, gevuld met alleen maar enen. Gebruik hiervoor `rep()`.

Vraag 8 Maak een vector `x` van lengte 50, gevuld met de getallen 1 t/m 50, en een vector `y` die gelijk is aan tweemaal `x`. Maak met `plot()` een scatterplot van `x` en `y`.

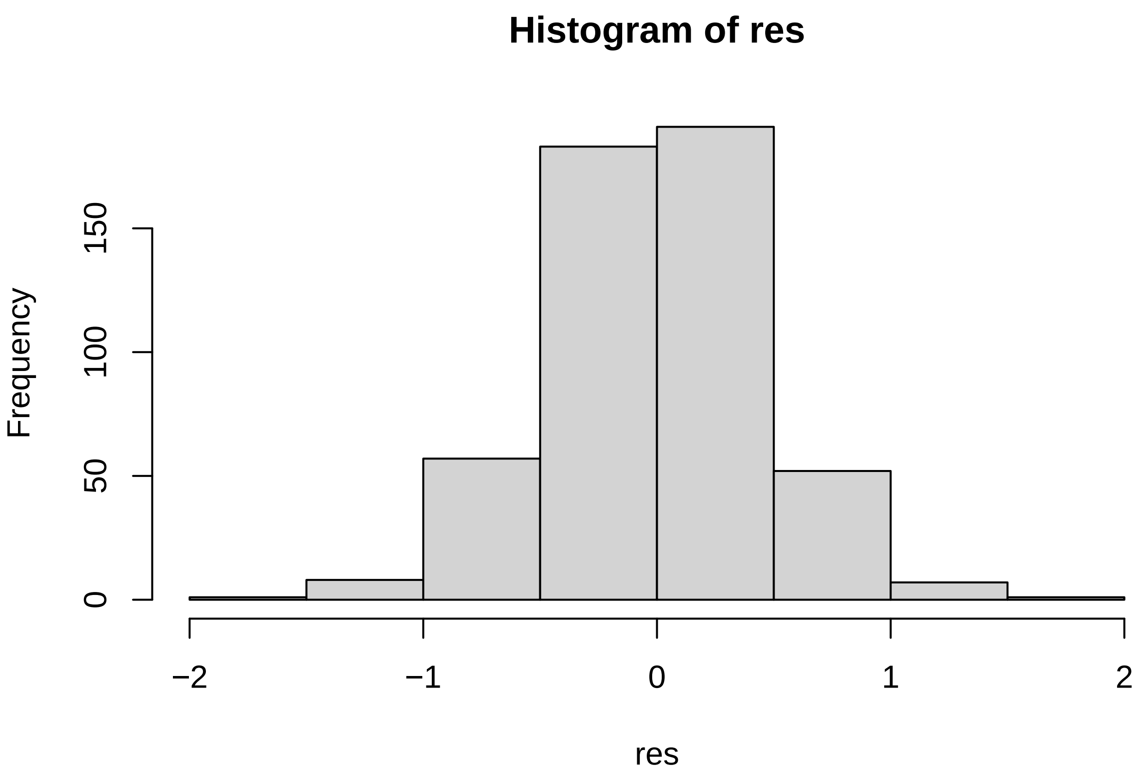
Vraag 9 Als vraag 8, maar trek nu een lijn door de punten heen.

Vraag 10 Bereken de correlatie tussen `x` en `y` met behulp van R.

Vraag 11 Als vraag 8, maar tel nu een vector standaard normaal verdeelde variabelen op bij `y`. Bereken tevens de correlatie tussen `x` en de nieuwe vector. Waarom is het antwoord anders dan bij vraag 9?

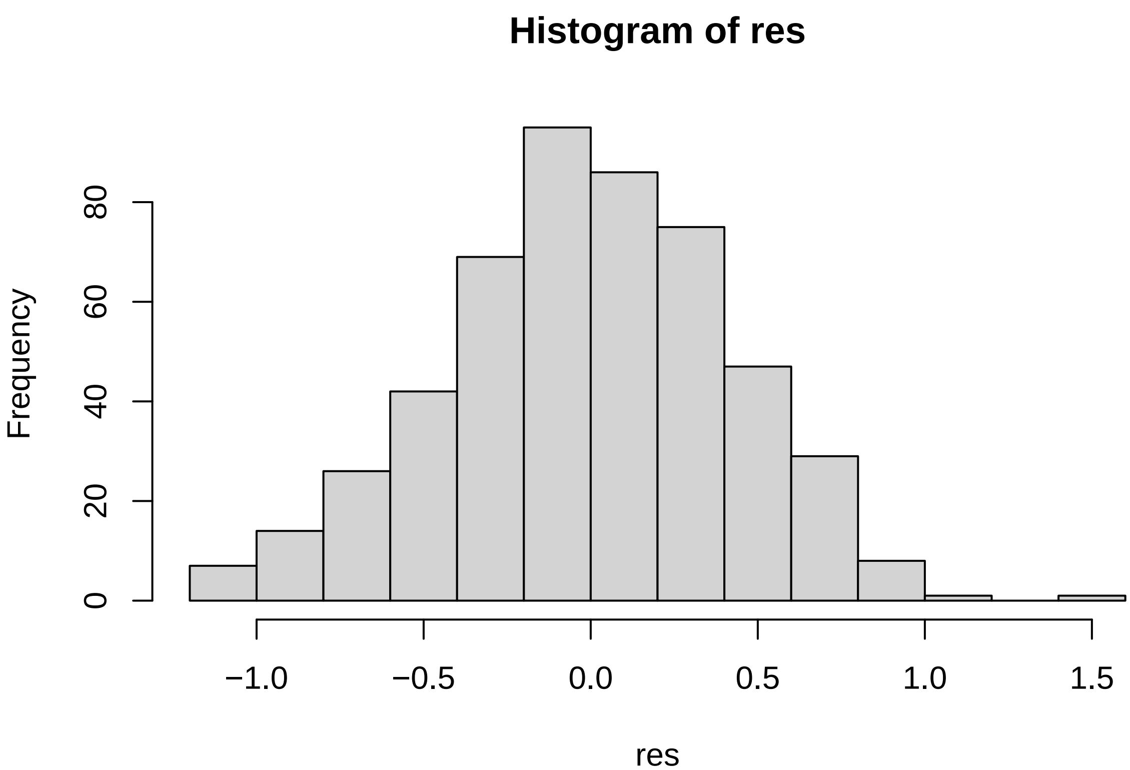
Vraag 12 Beschrijf in woorden wat er in onderstaande code gebeurt.

```
N <- 500
n <- 5
res <- rep(0, N)

for(i in 1:N){
  vector <- rnorm(n)
  res[i] <- mean(vector)
}

hist(res)
```



NB Een andere manier om hetzelfde te bereiken is:

```
N <- 500
n <- 5

obs <- matrix(rnorm(N*n), nrow = N)
res <- apply(obs, 1, mean)
hist(res)
```



Vraag 13 Maak het histogram uit bovenstaande vraag opnieuw, nu zo dat het histogram precies 20 bins heeft.

Vraag 14 Beschrijf wat in `pnorm()` het effect is van de keuze `lower.tail = TRUE` of juist `lower.tail = FALSE`.

NB In de help is te zien dat `lower.tail = TRUE` de 'default' is. Als je die optie wil, hoef je dat niet expliciet te vermelden wanneer je de functie aanroept.

Vraag 15 Zij $q_{0.3}$ de waarde zodanig dat de kans dat een standaard normaal verdeelde variabele kleiner dan $q_{0.3}$ is, gelijk is aan 0.3. Gebruik `qnorm()` om de waarde $q_{0.3}$ te vinden.

Vraag 16 We maken een vector met 20 uniform verdeelde waarnemingen.

```
obs <- runif(20)
obs
```

```
##  [1] 0.0689497269 0.2948936061 0.6127378687 0.9030575999 0.0002731008
##  [6] 0.2308603502 0.1817636313 0.7365311028 0.7996524740 0.6789397849
## [11] 0.3423936104 0.4793452781 0.9476468603 0.3755175697 0.6954132749
## [16] 0.6997398175 0.9703127663 0.5020238520 0.3415208487 0.9182093658
```

Bekijk wat de volgende functies doen:

```
min(obs)
```

```
## [1] 0.0002731008
```

```
max(obs)
```

```
## [1] 0.9703128
```

```
which.min(obs)
```

```
## [1] 5
```

```r
which.max(obs)
```

```
## [1] 17
```

```r
obs[which.min(obs)]
```

```
## [1] 0.0002731008
```

```r
obs[which.max(obs)]
```

```
## [1] 0.9703128
```

Gebruik nu de functie `which()` om de indices te vinden van de waarnemingen die groter dan 0.5 zijn. Gebruik deze indices vervolgens om de waarnemingen die groter dan 0.5 zijn te selecteren.