Examen Niet-parametrische Statistische Methoden

Bart Verweire

2018 M08 2

Contents

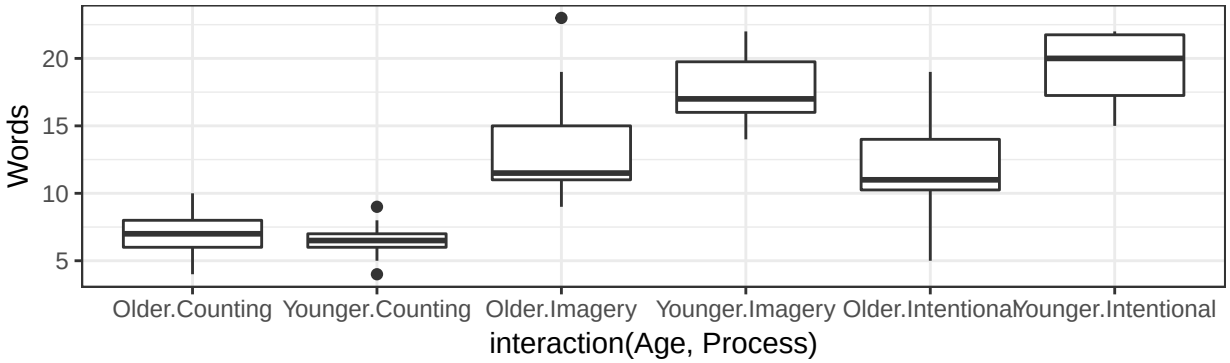
Vraag 1 - Geheugentest

Vraagstelling

Er wordt een theorie mbt. het geheugen getest. In het onderzoek wordt het aantal woorden geteld dat een proefpersoon zich kan herinneren. Het experiment voorziet in 3 verschillende processen en 2 leeftijdscategorieën.

De onderzoeksvraag is de volgende: *Is er een effect van de graad van verwerking op het aantal woorden dat de proefpersonen kunnen reproduceren en hangt dit effect mogelijks af van de leeftijd?*

Data Visualisatie



Uit de boxplots van het aantal gereproduceerde woorden voor de verschillende combinaties van Age en Process, kan afgeleid worden dat er een verschil is tussen het Counting proces en de processen Imagery en Intentional. Voor de processen Imagery en Intentional lijkt er ook een verschil te zijn tussen jongeren en ouderen. We bekijken dit probleem via Regressie, en via rang-gebaseerde methodes.

Regressie

Bij een lineaire regressie zijn de veronderstellingen :

- de observaties zijn onafhankelijk (in orde door random toewijzing)
- de residuelen zijn normaal verdeeld
- de variantie van de residuelen is onafhankelijk van de groep (voor categorieke variabelen)

De lineaire regressie is uitgevoerd op verschillende wijzen :

- onafhankelijk van de leeftijd (mem.lm.proc)
- afhankelijk van proces en leeftijd, met (mem.lm.int) en zonder interactie (mem.lm)

De modellen kunnen vergeleken worden via de functie AIC (An Information Criterion). Het model met de interactie term levert de laagste AIC, en is dus te verkiezen.

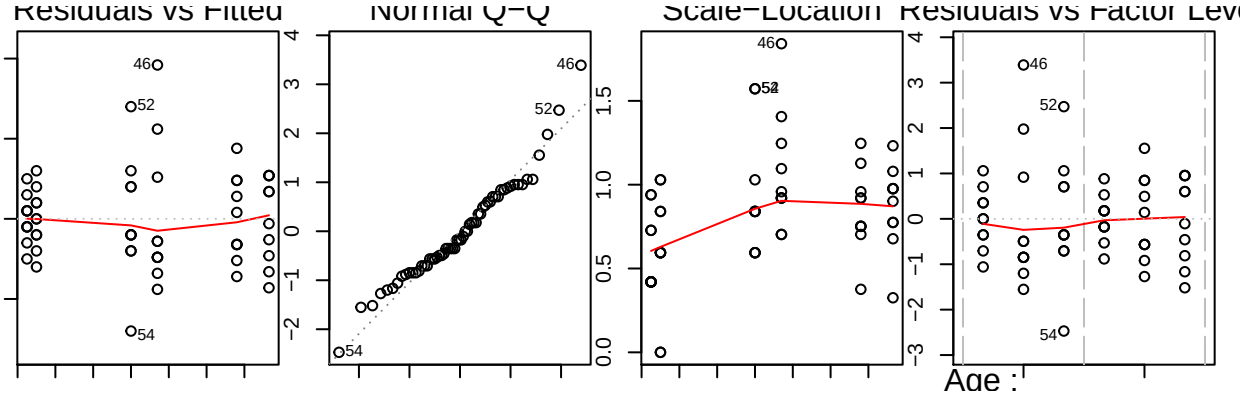
```
##             df      AIC
## mem.lm.proc  4 336.4229
## mem.lm       5 321.8893
## mem.lm.int   7 309.2139
```

De functie anova geeft in alle gevallen aan dat de regressie parameters significant zijn. Bv. voor het model met interactie :

```
##             Df    Sum Sq    Mean Sq   F value       Pr(>F)
## Age          1  201.6667 201.666667 22.621521 1.509712e-05
## Process      2 1038.6333 519.316667 58.253220 3.293292e-14
## Age:Process  2  154.2333  77.116667  8.650395 5.509147e-04
## Residuals   54  481.4000   8.914815        NA           NA
```

Er is een significant verschil tussen het Counting proces enerzijds, en de processen Imagery en Intentional anderzijds (van een significant verschil tussen Imagery en Intentional kunnen we via deze modellen niet spreken). In het algemeen is er ook een significante invloed van de leeftijd : jongeren onthouden meer woorden dan ouderen (maar niet voor het Counting proces).

In de plot van het model (bv. voor het model met interactie) zien we dat de voorwaarden rond normaliteit en gelijkheid van de variantie niet perfect, maar toch redelijk goed voldaan zijn.



Rang-gebaseerde methodes

We gebruiken :

- de Kruskal-Wallis test, op verschillende subsets (per Process, per Process beperkt tot een specifieke leeftijdscategorie, en per combinatie van Process en Age), om uit te maken of er al dan niet een subset met afwijkende distributie.
- De Wilcoxon-Mann-Whitney two-sample test voor elke combinatie van 2 subsets volgens de hierboven vermelde verdeling.
- In de veronderstelling van locatie-shift gebruiken we de Hodges-Lehman schatter om het verschil in gemiddelde van de subsets te bepalen.

De rang-gebaseerde methodes veronderstellen geen normaliteit of gelijke distributies, maar als we de Hodges-Lehman schatter gebruiken om de locatie-shift te berekenen, dan wordt natuurlijk wel verondersteld dat de distributies gelijkvormig zijn, en dus dat varianties van de verschillende distributies gelijk zijn.

Kruskal-Wallis testen

De p-waarde van de kruskal Wallis tests, uitgevoerd tov. de verschillend subsets, toont telkens aan dat er minstens 1 subset is met afwijkende distributie. De p-waarde is het laagst voor de test op alle combinaties van Process en Age.

```
##           test.name      p.value
## 1  Process (no Age) 1.891602e-08
## 2 Process (younger) 3.739886e-05
## 3   Process (older) 4.012503e-04
## 4     Process + Age 1.556013e-08
```

Wilcoxon-Mann-Whitney testen

Uitgevoerd voor alle subsets per Process en Age, kunnen we hieruit de Hodges-Lehmann schatter, het betrouwbaarheids interval en de p-waarde berekenen. Onderstaande tabel bevat de resultaten, samen met deze van het lineair model met interactie (kolom diff).

Table 1: Samenvattende tabel Lineair model + Wilcoxon test

subset1	subset2	diff	p.value	lower	upper	location.shift	p.value.adj	significant
Older.Counting	Younger.Counting	0.5	0.5380792	-1	2	0.500	1.0000000	FALSE
Older.Counting	Older.Imagery	-6.4	0.0003131	-9	-3	-5.000	0.0046961	TRUE
Older.Counting	Younger.Imagery	-10.6	0.0001746	-13	-8	-10.000	0.0026194	TRUE
Older.Counting	Older.Intentional	-5.0	0.0023768	-8	-2	-5.000	0.0356517	TRUE
Older.Counting	Younger.Intentional	-12.3	0.0001746	-15	-10	-12.749	0.0026194	TRUE
Younger.Counting	Older.Imagery	-6.9	0.0001963	-10	-4	-5.000	0.0029448	TRUE
Younger.Counting	Younger.Imagery	-11.1	0.0001697	-13	-9	-11.000	0.0025459	TRUE
Younger.Counting	Older.Intentional	-5.5	0.0018274	-8	-3	-5.000	0.0274113	TRUE
Younger.Counting	Younger.Intentional	-12.8	0.0001697	-15	-10	-13.000	0.0025459	TRUE
Older.Imagery	Younger.Imagery	-4.2	0.0248079	-8	-1	-5.000	0.3721191	FALSE
Older.Imagery	Older.Intentional	1.4	0.6175675	-3	5	1.000	1.0000000	FALSE
Older.Imagery	Younger.Intentional	-5.9	0.0109634	-10	-3	-6.484	0.1644506	FALSE
Younger.Imagery	Older.Intentional	5.6	0.0020693	2	9	5.000	0.0310402	TRUE
Younger.Imagery	Younger.Intentional	-1.7	0.1695524	-5	1	-2.000	1.0000000	FALSE
Older.Intentional	Younger.Intentional	-7.3	0.0007237	-11	-4	-7.000	0.0108553	TRUE

De kolom **p.value.adj** is de bonferroni-aangepaste p-waarde, en de kolom **significant** geeft aan of deze aangepaste p-waarde < 0.05. We zien dat het via de lineaire regressie bepaalde verschil steeds binnen het betrouwbaarheidsinterval van de Wilcoxon test valt.

Conclusie

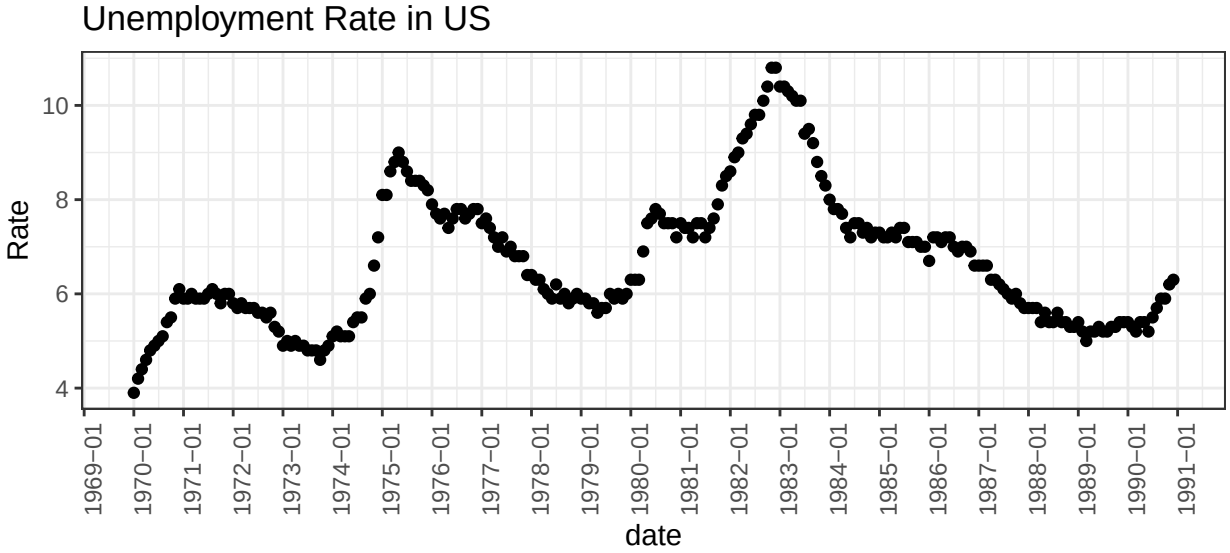
Beide methodes geven aan dat er een significant effect is van het proces, en van de leeftijd.
Het Counting proces is significant verschillend van de Imagery en Intentional processen, voor elke leeftijdscategorie.
Het Intentional proces voor ouderen is significant minder efficiënt dan wat de jongeren presteren in zowel het Imagery en Intentional proces. De methode via lineaire regressie laat enkel toe om na te gaan of er een significant effect is tov. een referentietoestand. De Wilcoxon-Mann-Whitney tests daarentegen laten toe om alle subsets met elkaar te vergelijken.

Vraag 2 - Werkloosheidsgraad

Vraagstelling

Op basis van de cijfers van de werkloosheidsgraad wordt gevraagd om na te gaan of er een langetermijnseffect en een seizoenseffect is.

Data Visualisatie



We zien in bovenstaande figuur een duidelijk niet-lineair verloop. Er lijkt een periodieke component te zijn, maar de periode is groter dan 1 jaar.

Uitwerking

Gezien de niet-lineariteit is een gam-model aangewezen. Er zijn 2 onafhankelijke variabelen, namelijk Year en Monthly, maar we kunnen die combineren in 1 tijdsvariabele. We kunnen dus een aantal mogelijke modellen opstellen

- met Year als onafhankelijke variabele
- met Year en Monthly als onafhankelijke variabelen
- met Tijd (of numeriek : Year + Monthly/12) als onafhankelijke variabele

De functie anova geeft voor elk model aan of de coëfficienten significant zijn.

```
##          model.name    coefficient       p-value
## 1     Year, Monthly        s(Year) 2.415320e-200
## 2     Year, Monthly     s(Monthly)  2.914169e-01
## 3              Year        s(Year) 2.421312e-200
## 4 Year + Monthly/12 s(date.number) 1.747472e-252
## 5     Year (linear)           Year  2.119049e-02
```

Er is een significant niet-lineair verband met Year, maar niet met Monthly (p-value = 0.29). In het zuiver lineaire model zijn de coëfficiënten zwak significant. Maar, zoals visueel al duidelijk was, kunnen we er beter van uitgaan dat de afhankelijkheid niet-lineair is.

Het model op basis van tijd (Year + Monthly / 12) toont eveneens een significant niet-lineair verband.

Vergelijking

De modellen kunnen vergeleken worden via AIC.

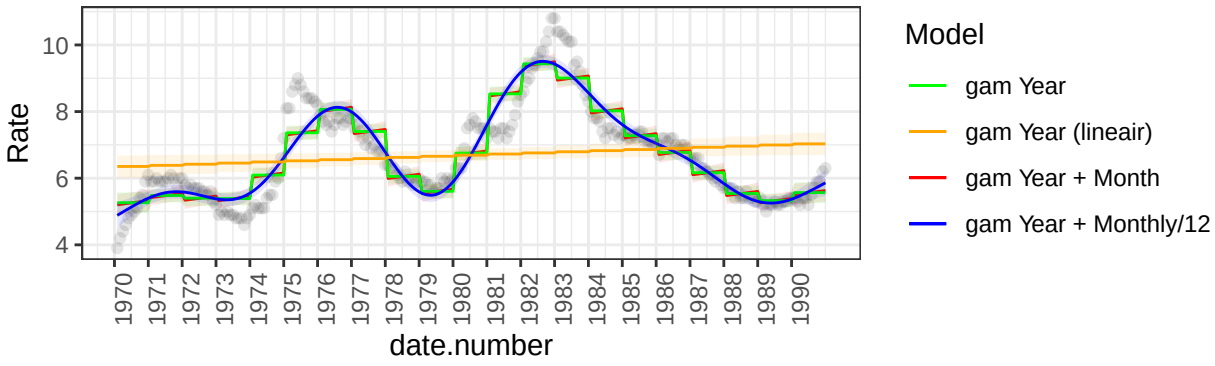
```
##                        df      AIC
## ue.mod           11.95173 465.0867
```

```
## ue.mod.datenr   10.96919 416.1111
## ue.mod.year     10.95171 464.2527
## ue.mod.year.lin  3.00000 890.9637
```

Op basis van de AIC, is het niet-lineair model op basis van Year + Monthly /12 de beste keuze.

Visualisatie van de modellen

Zetten we alle resultaten in één grafiek:



Conclusie

- Er is een significant niet-lineair verband tussen werkloosheidgraad en de tijd
- De invloed van de jaarcomponent is veel belangrijker dan die van de maandcomponent.
- De periode van de fluctuaties is groter dan 1 jaar. Er is geen seizoensinvloed.

Vraag 3 - Monte Carlo simulatie

Vraagstelling

Opzetten van een Monte-Carlo simulatie voor een two-sample t-test evalueert onder de nulhypothese H_0 : $\mu_1 = \mu_2$

Uitwerking

Er wordt gevraagd om Monte-Carlo simulaties uit te voeren in een aantal verschillende situaties. Hiervoor definiëren we een functie, met volgende parameters (algemener dan in de vraagstelling)

- dist (waarden **norm** of **lnorm**) : Data uit een normale/lognormale verdeling
- n.total : totaal aantal samples voor steekproef 1 en steekproef 2 samen
- sample.ratio (default 1) : verhouding van het aantal samples in steekproef 1 tov. steekproef 2, bv.
  - n.total = 200, sample.ratio = 3 leidt tot 150 samples in steekproef 1 en 50 in steekproef 2
  - de default waarde leidt tot een gelijk aantal samples
- mu : gemiddelde waarde voor beide steekproeven
- sd : standaardafwijking voor steekproef 1
- sd.ratio : verhouding van standaardafwijking voor steekproef 1 tov. die van steekproef 2
  - sd.ratio = 5 betekent $\sigma_1 = 1$ en $\sigma_2 = 1/5$

- test.type (waarden **pooled** of **Welch**) : geeft aan of de t-test gebruik maakt van de "pooled" variantie, of van de Welch benadering voor het aantal vrijheidsgraden.
- R (default 10000) : aantal Monte-Carlo simulaties in elk scenario
- alpha (default 0.05) : p-value drempelwaarde

Om alle scenario's uit te voeren, bouwen we eerst een data frame op met alle combinaties van voorwaarden. Via lapply kunnen we de Monte-Carlo simulatiefunctie toepassen met de parameters in dit data frame.

Onderstaande tabel geeft het resultaat weer :

Table 2: Monte-Carlo simulatie voor 32 Scenario's

dist	sd.ratio	n.total	sample.ratio	test.type	p.val
norm	1	20	0.3333333	pooled	0.0504
norm	1	20	0.3333333	Welch	0.0576
norm	1	20	1.0000000	pooled	0.0510
norm	1	20	1.0000000	Welch	0.0517
norm	1	200	0.3333333	pooled	0.0505
norm	1	200	0.3333333	Welch	0.0492
norm	1	200	1.0000000	pooled	0.0496
norm	1	200	1.0000000	Welch	0.0532
norm	5	20	0.3333333	pooled	0.2802
norm	5	20	0.3333333	Welch	0.0515
norm	5	20	1.0000000	pooled	0.0633
norm	5	20	1.0000000	Welch	0.0508
norm	5	200	0.3333333	pooled	0.2370
norm	5	200	0.3333333	Welch	0.0507
norm	5	200	1.0000000	pooled	0.0545
norm	5	200	1.0000000	Welch	0.0505
lnorm	1	20	0.3333333	pooled	0.0466
lnorm	1	20	0.3333333	Welch	0.0625
lnorm	1	20	1.0000000	pooled	0.0371
lnorm	1	20	1.0000000	Welch	0.0322
lnorm	1	200	0.3333333	pooled	0.0458
lnorm	1	200	0.3333333	Welch	0.0564
lnorm	1	200	1.0000000	pooled	0.0467
lnorm	1	200	1.0000000	Welch	0.0468
lnorm	5	20	0.3333333	pooled	0.3188
lnorm	5	20	0.3333333	Welch	0.1413
lnorm	5	20	1.0000000	pooled	0.1261
lnorm	5	20	1.0000000	Welch	0.1216
lnorm	5	200	0.3333333	pooled	0.2627
lnorm	5	200	0.3333333	Welch	0.0869
lnorm	5	200	1.0000000	pooled	0.0661
lnorm	5	200	1.0000000	Welch	0.0659

Maar het is overzichtelijker om de invloed van elk van de parameters te bekijken, door voor elke parameter een succes percentage te berekenen, dwz. het aantal gevallen waarbij de Monte-Carlo simulatie een "aanvaardbare" p-waarde oplevert.

Voor een "aanvaardbare" p-waarde van 0.055 :

Table 3: Succes percentage voor p-waarde <= 0.055

parameter	value	n	success.pct
dist	norm	12	75
dist	lnorm	6	38
sd.ratio	1	13	81
sd.ratio	5	5	31
n.total	20	8	50
n.total	200	10	62
sample.ratio	0.333333333333333	7	44
sample.ratio	1	11	69
test.type	pooled	9	56
test.type	Welch	9	56

We kunnen bv. ook de invloed bekijken van het type test, voor de verschillende combinaties van de andere parameters. De kolom winner bevat de test met de laagste p-waarde, en de kolom diff_pct het procentueel verschil tov. de kleinste waarde.

Table 4: Invloed van test type onder verschillende scenario's

dist	sd.ratio	n.total	sample.ratio	pooled	Welch	winner	diff_pct
norm	1	20	0.3333333	0.0504	0.0576	pooled	14.3
norm	1	20	1.0000000	0.0510	0.0517	pooled	1.4
norm	1	200	0.3333333	0.0505	0.0492	Welch	-2.6
norm	1	200	1.0000000	0.0496	0.0532	pooled	7.3
norm	5	20	0.3333333	0.2802	0.0515	Welch	-444.1
norm	5	20	1.0000000	0.0633	0.0508	Welch	-24.6
norm	5	200	0.3333333	0.2370	0.0507	Welch	-367.5
norm	5	200	1.0000000	0.0545	0.0505	Welch	-7.9
lnorm	1	20	0.3333333	0.0466	0.0625	pooled	34.1
lnorm	1	20	1.0000000	0.0371	0.0322	Welch	-15.2
lnorm	1	200	0.3333333	0.0458	0.0564	pooled	23.1
lnorm	1	200	1.0000000	0.0467	0.0468	pooled	0.2
lnorm	5	20	0.3333333	0.3188	0.1413	Welch	-125.6
lnorm	5	20	1.0000000	0.1261	0.1216	Welch	-3.7
lnorm	5	200	0.3333333	0.2627	0.0869	Welch	-202.3
lnorm	5	200	1.0000000	0.0661	0.0659	Welch	-0.3
## Concl	usie						

De Monte-Carlo simulatie is efficiënter

- voor een normale distributie dan voor een lognormale
- wanneer de standaardafwijkingen van beide populaties gelijk zijn
- wanneer de samples even groot zijn, en voldoende groot
- voor gelijke standaardafwijkingen tov. verschillende standaardafwijkingen
- een Welch t-test geeft betere resultaten dan een pooled-variance, wanneer de test voorwaarden minder goed voldaan zijn

Appendix 1 - Geheugentest - Gedetailleerde Uitwerking

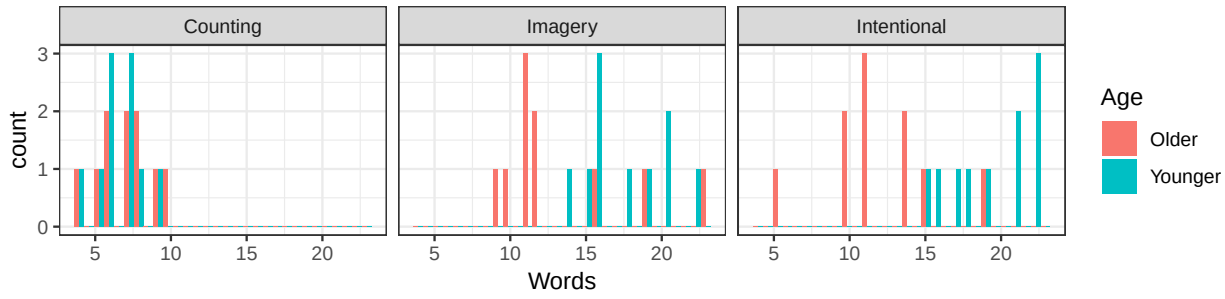
Data Visualisatie

Opladen data

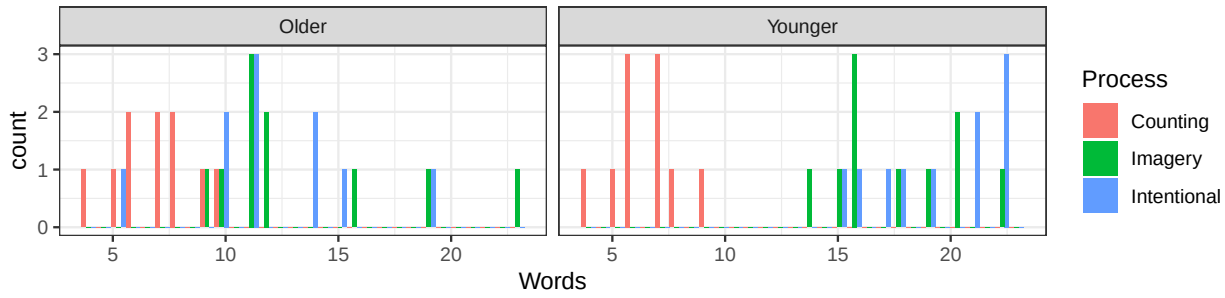
```
load(file = "memory.rda")
memory.younger <- memory %>%
  filter(Age == "Younger")
memory.older <- memory %>%
  filter(Age == "Older")
```

Visualisatie

```
# Visualiseer data
ggplot(memory, aes(Words, fill = Age)) +
  geom_histogram(position = "dodge") +
  facet_wrap(~ Process)
```




```
ggplot(memory, aes(Words, fill = Process)) +
  geom_histogram(position = "dodge") +
  facet_wrap(~ Age)
```




```
ggplot(memory, aes(interaction(Age, Process), Words)) +
  geom_boxplot()
```

Uit de figuren kan afgeleid worden dat er een verschil is in het aantal gereproduceerde woorden tussen het Counting proces en de andere processen (Imagery en Intentional). Voor de processen Imagery en Intentional lijkt er ook een verschil te zijn tussen jongeren en ouderen.

We onderzoeken nu of het aantal gereproduceerde woorden afhankelijk is van het proces, en of dit effect afhankelijk is van de leeftijd.

Regressie

We voeren een lineaire regressie uit. Merk op dat de onafhankelijke variabelen alle categorische variabelen zijn.

Bij een lineaire regressie zijn de veronderstellingen :

- de observaties zijn onafhankelijk : we gaan ervan uit dat dit klopt, aangezien de onderzoekers de personen random in groepen verdeeld hebben.
- de residuelen zijn normaal verdeeld
- de variantie van de residuelen is onafhankelijk van de groep

We kunnen de lineaire regressie uitvoeren op verschillende wijzen :

- onafhankelijk van de leeftijd
- afhankelijk van proces en leeftijd, zonder interactie
- afhankelijk van proces en leeftijd, met en zonder interactie tussen de variabelen Age en Process.

Test functie

Om de resultaten van het lineaire model eenvoudiger te vergelijken met de resultaten van de Wilcoxon testen (zie verder), definiëren we een functie om de resultaten van de lineaire regressie in een data.frame te bewaren.

```
lm.tests <- function(model, test.data) {
  pred <- predict(model, newdata = test.data, se.fit = TRUE)
  res.table <- test.data %>%
    bind_cols(value = pred$fit,
              lower = pred$fit - 1.96*pred$se.fit,
              upper = pred$fit + 1.96*pred$se.fit) %>%
    mutate(subset = id)

  ind.comb <- combn(1:length(res.table), 2)

  pairwise.diffs <- lapply(1:ncol(ind.comb), function(i) {
    diff = res.table[ind.comb[1,i],"value"] - res.table[ind.comb[2,i],"value"]

    lm.val <- data.frame(
      subset1 = res.table[ind.comb[1,i],"subset"],
```

```
      subset2 = res.table[ind.comb[2,i],"subset"],
      diff = round(diff, 3)
    )

    lm.val
  })

  do.call("bind_rows", pairwise.diffs)
}
```

We zullen de test functie uitvoeren op de resultaten van de predict functie, met als nieuwe data een data frame met de mogelijke combinaties van Age en Process, of van Process (afhankelijk van het model).

```
test.data.age.process <- expand.grid(Age = c("Older","Younger"),
                                     Process = c("Counting","Imagery","Intentional")) %>%
  mutate(id = paste(Age, Process, sep = "."))
test.data.process <- data.frame(Process = c("Counting","Imagery","Intentional")) %>%
  mutate(id = Process)
test.data.process.younger <- data.frame(Process = c("Counting","Imagery","Intentional"),
                                        Age = "Younger") %>%
  mutate(id = Process)
test.data.process.older <- data.frame(Process = c("Counting","Imagery","Intentional"),
                                      Age = "Older") %>%
  mutate(id = Process)

test.data.age.process
```

```
##       Age     Process                  id
## 1   Older    Counting      Older.Counting
## 2 Younger    Counting    Younger.Counting
## 3   Older     Imagery       Older.Imagery
## 4 Younger     Imagery     Younger.Imagery
## 5   Older Intentional   Older.Intentional
## 6 Younger Intentional Younger.Intentional
```

```
test.data.process.younger
```

```
##       Process     Age          id
## 1    Counting Younger    Counting
## 2     Imagery Younger     Imagery
## 3 Intentional Younger Intentional
```

```
test.data.process.older
```

```
##       Process   Age          id
## 1    Counting Older    Counting
## 2     Imagery Older     Imagery
## 3 Intentional Older Intentional
```

Effect van het proces

```
mem.lm.proc = lm(Words ~ Process, data = memory)
```

```
summary(mem.lm.proc)
```

```
##
## Call:
## lm(formula = Words ~ Process, data = memory)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.650  -2.000   0.250   2.388   7.500
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)           6.750      0.857   7.876 1.10e-10 ***
## ProcessImagery        8.750      1.212   7.219 1.37e-09 ***
## ProcessIntentional    8.900      1.212   7.343 8.52e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.833 on 57 degrees of freedom
## Multiple R-squared:  0.5537, Adjusted R-squared:  0.538
## F-statistic: 35.35 on 2 and 57 DF,  p-value: 1.036e-10
```

```
anova(mem.lm.proc)
```

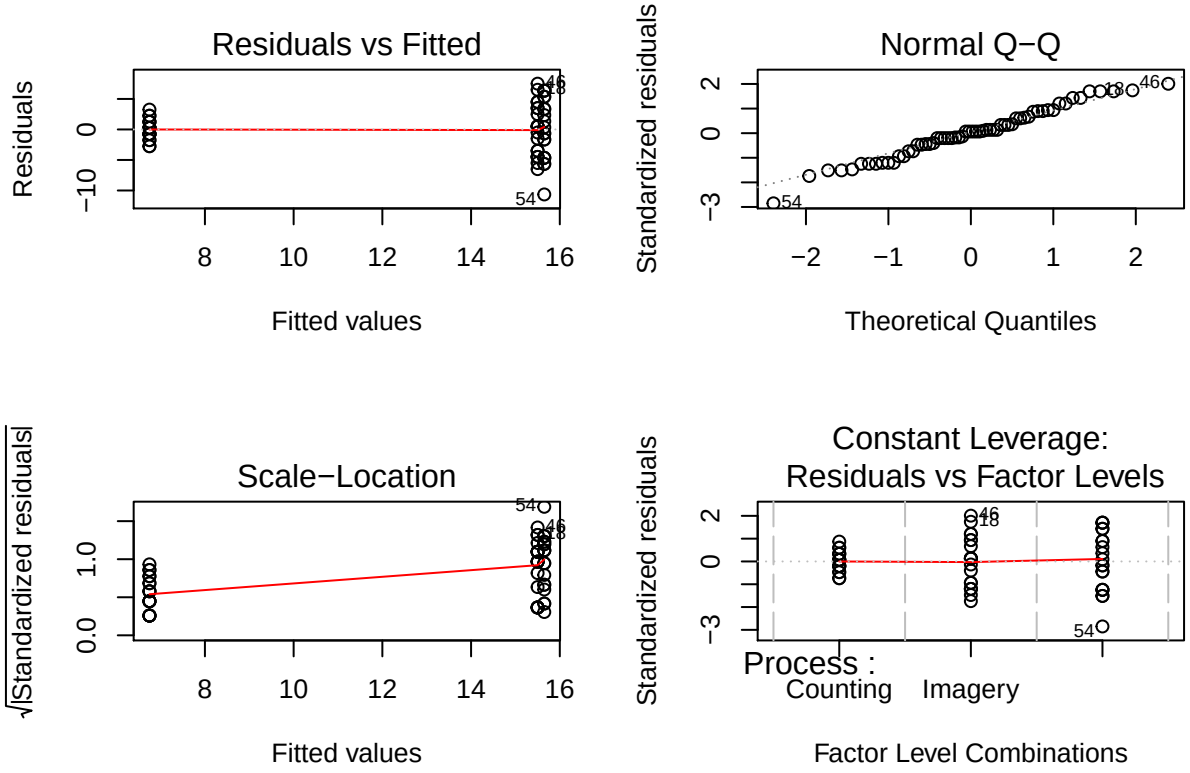
```
## Analysis of Variance Table
##
## Response: Words
##           Df Sum Sq Mean Sq F value    Pr(>F)
## Process    2 1038.6  519.32  35.353 1.036e-10 ***
## Residuals 57  837.3   14.69
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Er is een significant verband tussen het aantal gereproduceerde woorden en het proces.

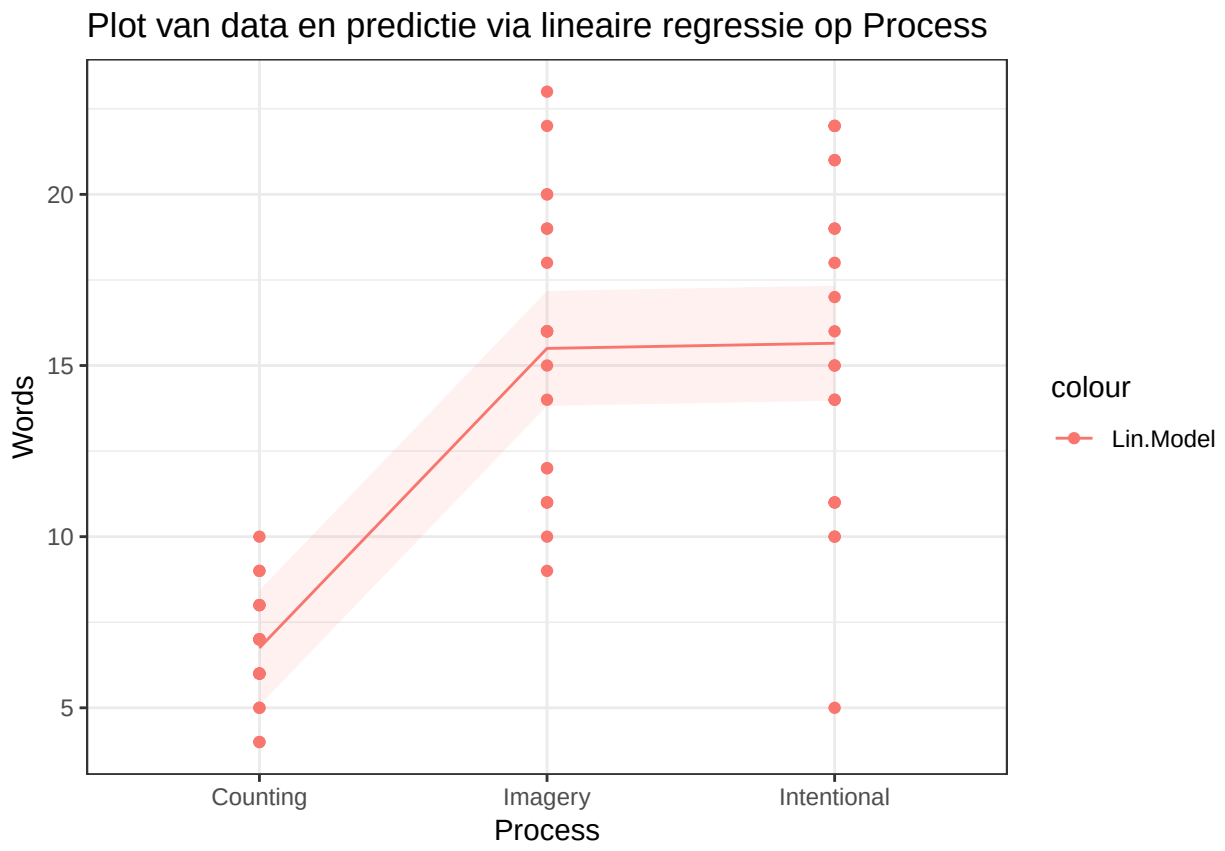
Zowel de t-test als de F-test wijzen op een significant verband is tussen het proces Counting enerzijds, en de processen Imagery en Intentional anderzijds (van een significant verschil tussen Imagery en Intentional kunnen we via dit model niet spreken).

In de plot van het model zien we dat de voorwaarden rond normaliteit en gelijkheid van de variantie redelijk goed voldaan zijn.
De variantie niet onafhankelijk van de gefitte waarde, en de residuals zijn niet perfect normaal verdeeld, maar het verschil is aanvaardbaar.

```
par(mfrow=c(2,2))
plot(mem.lm.proc)
```




```
mem.lm.proc.pred <- predict(mem.lm.proc, se.fit = TRUE)
mem.lm.proc.data <- memory %>%
  mutate(fit = mem.lm.proc.pred$fit,
         l = mem.lm.proc.pred$fit - 1.96*mem.lm.proc.pred$se.fit,
         u = mem.lm.proc.pred$fit + 1.96*mem.lm.proc.pred$se.fit)

ggplot(mem.lm.proc.data, aes(Process, Words, color = "Lin.Model", fill = "Lin.Model")) +
  geom_point() +
  geom_line(aes(x = as.numeric(Process), y = fit)) +
  geom_ribbon(aes(as.numeric(Process), ymin = l, ymax = u),
              size = 0, linetype = 2, alpha = 0.1, color = NA) +
  ggtitle("Plot van data en predictie via lineaire regressie op Process") +
  guides(fill = FALSE)
```



Effect van leeftijd, zonder interactie

We definiëren het volgende model

```
mem.lm <- lm(Words ~ Age + Process, data = memory)
```

```
summary(mem.lm)
```

```
##
## Call:
## lm(formula = Words ~ Age + Process, data = memory)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.8167 -2.5833 -0.5333  2.4167  9.3333
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)          4.9167     0.8699   5.652 5.56e-07 ***
## AgeYounger           3.6667     0.8699   4.215 9.18e-05 ***
## ProcessImagery       8.7500     1.0654   8.213 3.45e-11 ***
## ProcessIntentional   8.9000     1.0654   8.354 2.03e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.369 on 56 degrees of freedom
```

```
## Multiple R-squared:  0.6612, Adjusted R-squared:  0.643
## F-statistic: 36.42 on 3 and 56 DF,  p-value: 3.431e-13
```

```
anova(mem.lm)
```

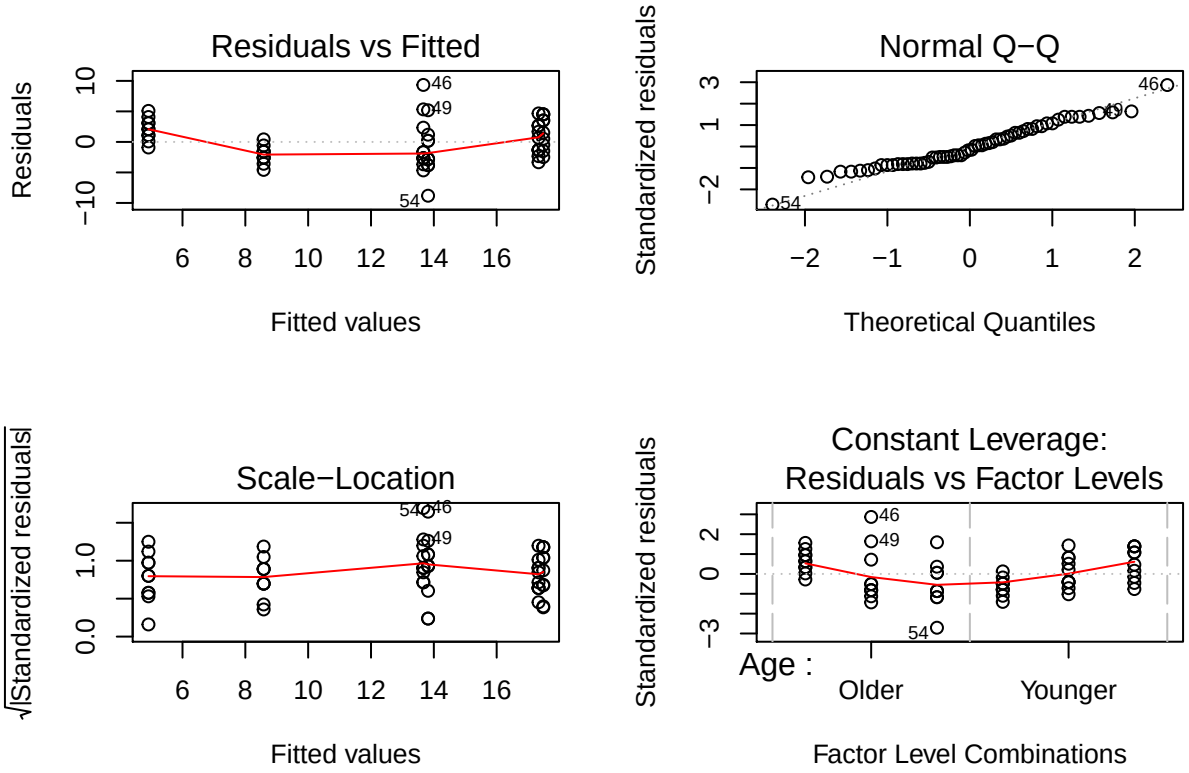
```
## Analysis of Variance Table
##
## Response: Words
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## Age        1  201.67  201.67  17.767 9.184e-05 ***
## Process    2 1038.63  519.32  45.752 1.670e-12 ***
## Residuals 56  635.63   11.35
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Zowel de t-test als de F-test wijzen op een significant verband is tussen het proces Counting enerzijds, en de processen Imagery en Intentional anderzijds (van een significant verschil tussen Imagery en Intentional kunnen we via dit model niet spreken). In het algemeen is er ook een significante invloed van de leeftijd : jongeren onthouden meer woorden dan ouderen.

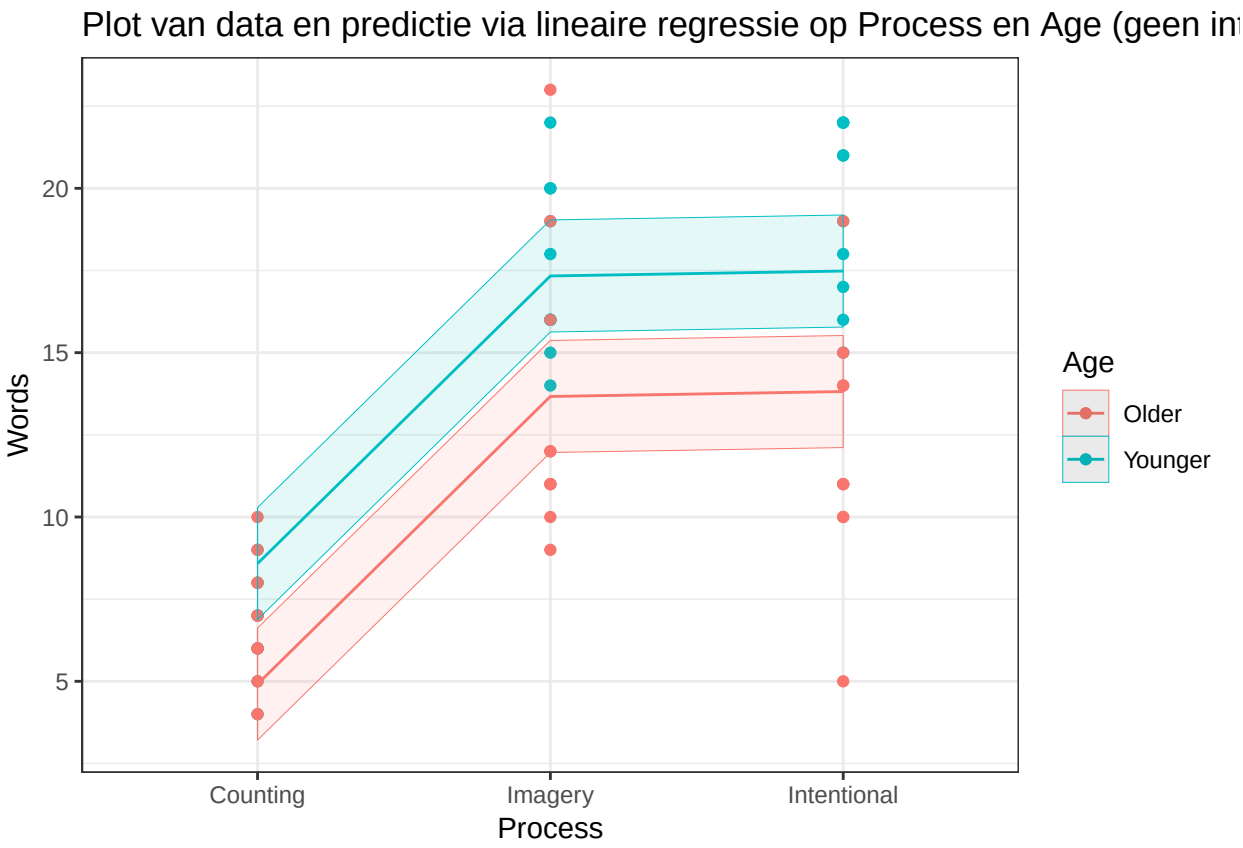
In de plot van het model zien we dat de voorwaarden rond normaliteit en gelijkheid van de variantie redelijk goed voldaan zijn.

De variantie niet onafhankelijk van de gefitte waarde, en de residuals zijn niet perfect normaal verdeeld, maar het verschil is aanvaardbaar.

```
par(mfrow=c(2,2))
plot(mem.lm)
```



```r
mem.lm.pred <- predict(mem.lm, se.fit = TRUE)
mem.lm.data <- memory %>%
  mutate(fit = mem.lm.pred$fit,
         l = mem.lm.pred$fit - 1.96*mem.lm.pred$se.fit,
         u = mem.lm.pred$fit + 1.96*mem.lm.pred$se.fit)

ggplot(mem.lm.data, aes(Process, Words, color = Age, fill = Age)) +
  geom_point() +
  geom_line(aes(x = as.numeric(Process), y = fit)) +
  geom_ribbon(aes(as.numeric(Process), ymin = l, ymax = u), size = 0, linetype = 2, alpha = 0.1) +
  ggtitle("Plot van data en predictie via lineaire regressie op Process en Age (geen interactie)") +
  guides(fill = FALSE)
```



Met interactie

We kijken nu naar de invloed van de interactie tussen Age en Process, via volgend model :

```r
mem.lm.int <- lm(Words ~ Age * Process, data = memory)
```

```r
summary(mem.lm.int)
```

```
##
## Call:
## lm(formula = Words ~ Age * Process, data = memory)
##
## Residuals:
```

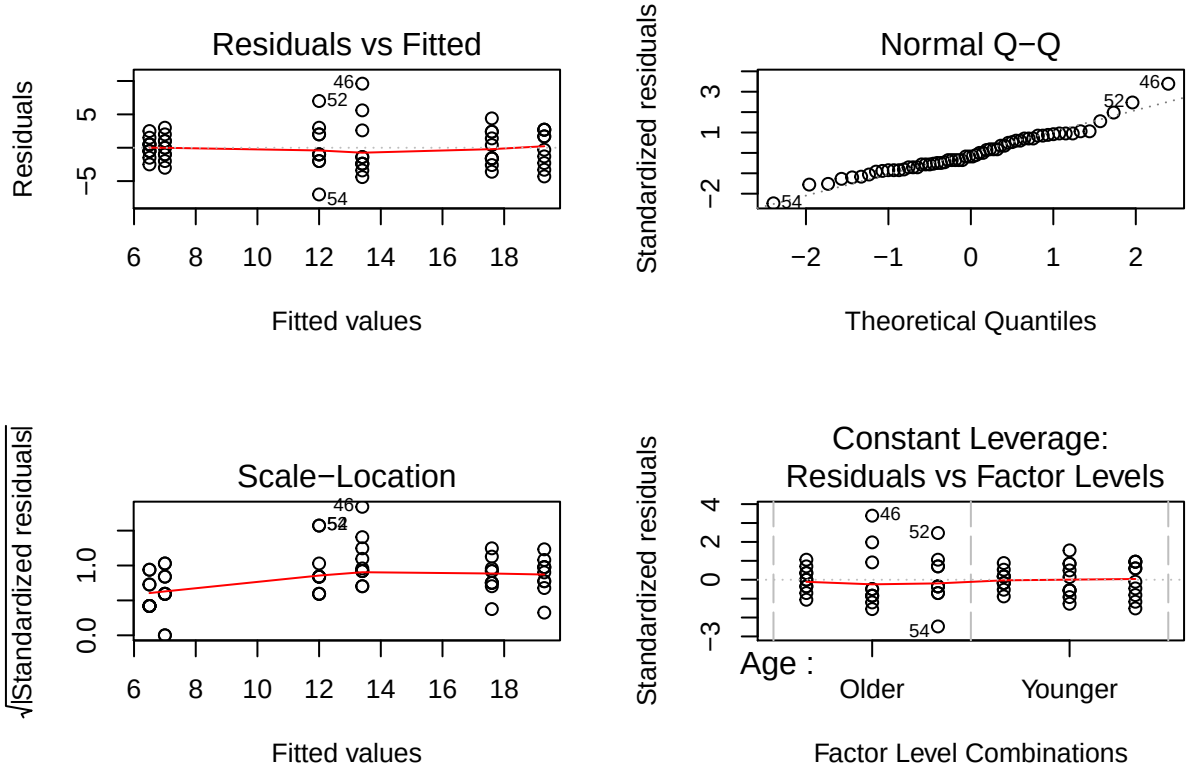
```
##    Min     1Q Median     3Q    Max
##   -7.0   -2.0   -0.5    2.0    9.6
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    7.0000     0.9442   7.414 8.77e-10 ***
## AgeYounger                    -0.5000     1.3353  -0.374 0.709533
## ProcessImagery                 6.4000     1.3353   4.793 1.33e-05 ***
## ProcessIntentional             5.0000     1.3353   3.745 0.000440 ***
## AgeYounger:ProcessImagery      4.7000     1.8884   2.489 0.015927 *
## AgeYounger:ProcessIntentional  7.8000     1.8884   4.131 0.000127 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.986 on 54 degrees of freedom
## Multiple R-squared:  0.7434, Adjusted R-squared:  0.7196
## F-statistic: 31.29 on 5 and 54 DF,  p-value: 8.288e-15
```

```
anova(mem.lm.int)
```

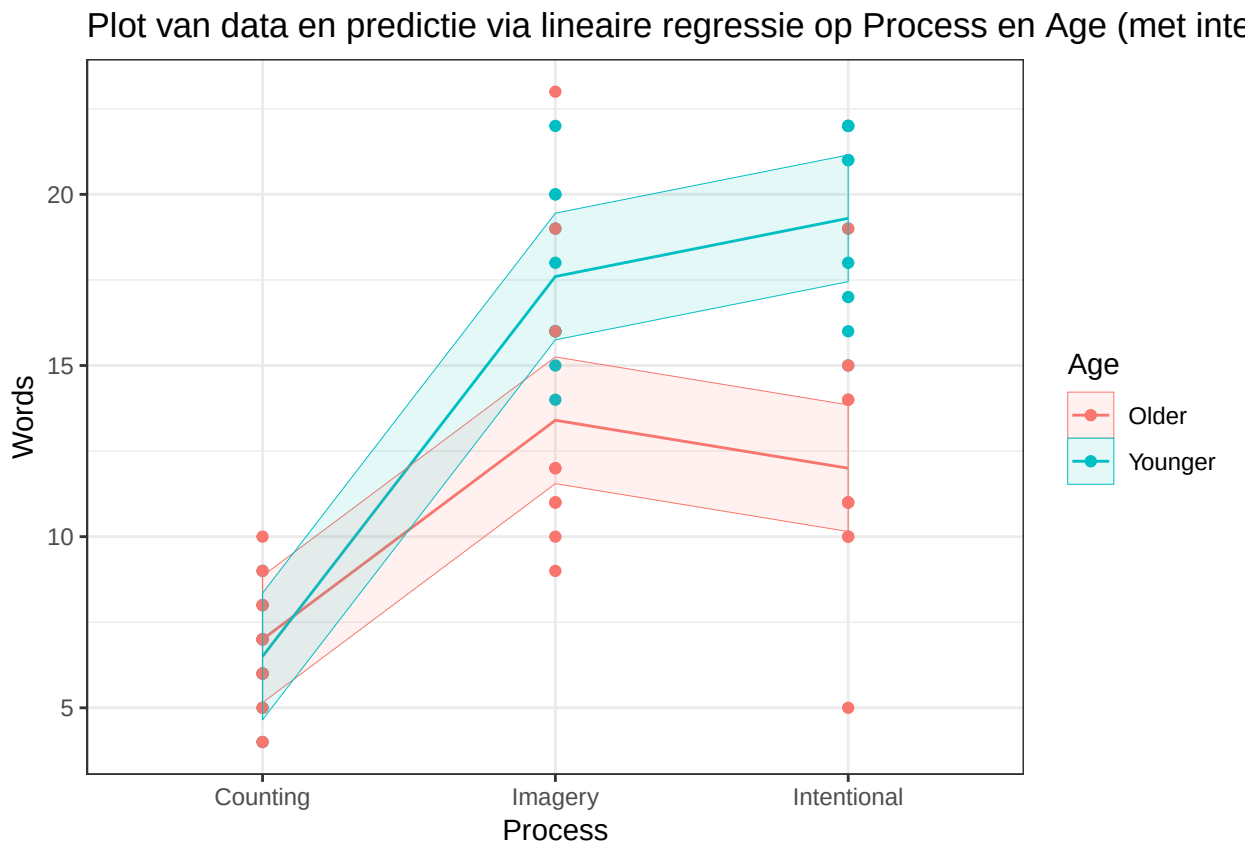
```
## Analysis of Variance Table
##
## Response: Words
##             Df  Sum Sq Mean Sq F value    Pr(>F)
## Age          1  201.67  201.67 22.6215 1.510e-05 ***
## Process      2 1038.63  519.32 58.2532 3.293e-14 ***
## Age:Process  2  154.23   77.12  8.6504 0.0005509 ***
## Residuals   54  481.40    8.91
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Ook de interactieterm is dus significant, vooral dan in geval van het Intentional Proces. Dit gaat ten koste van de algemene leeftijdsfactor, die nu niet significant is.

```
par(mfrow=c(2,2))
plot(mem.lm.int)
```




```
mem.lm.int.pred <- predict(mem.lm.int, se.fit = TRUE)
mem.lm.int.data <- memory %>%
  mutate(fit = mem.lm.int.pred$fit,
         l = mem.lm.int.pred$fit - 1.96*mem.lm.int.pred$se.fit,
         u = mem.lm.int.pred$fit + 1.96*mem.lm.int.pred$se.fit)

ggplot(mem.lm.int.data, aes(Process, Words, color = Age, fill = Age)) +
  geom_point() +
  geom_line(aes(as.numeric(Process), fit)) +
  geom_ribbon(aes(x = as.numeric(Process), ymin = l, ymax = u), size = 0, linetype = 2, alpha = 0.1) +
  ggtitle("Plot van data en predictie via lineaire regressie op Process en Age (met interactie)")
```



Merk op dat het model met interactie er op neer komt dat voor elke combinatie (Age, Process) het gemiddelde genomen wordt.

We gebruiken het model om het aantal woorden te voorspellen.

```
pred.lm.int <- predict(mem.lm.int, newdata = test.data.age.process)
res.table <- test.data.age.process %>%
  bind_cols(value = pred.lm.int)
```

En dit resultaat vergelijken we met een tabel met de gemiddelden :

```
avg.table.age.process <- memory %>%
  group_by(Age, Process) %>%
  summarize(mean.age.process = mean(Words))
```

Beide resultaten kunnen vergeleken worden door de resultaten te joinen

```
comp.table <- res.table %>%
  inner_join(avg.table.age.process, by = c("Age","Process"))
comp.table
```

```
##       Age     Process                  id value mean.age.process
## 1   Older    Counting      Older.Counting   7.0              7.0
## 2 Younger    Counting    Younger.Counting   6.5              6.5
## 3   Older     Imagery       Older.Imagery  13.4             13.4
## 4 Younger     Imagery     Younger.Imagery  17.6             17.6
## 5   Older Intentional   Older.Intentional  12.0             12.0
## 6 Younger Intentional Younger.Intentional  19.3             19.3
```

Keuze van het model

```
AIC(mem.lm, mem.lm.int)
```

```
##            df      AIC
## mem.lm      5 321.8893
## mem.lm.int  7 309.2139
```

Het model met de interactie term levert de laagste AIC, en is dus te verkiezen.

Rang-gebaseerde methodes

De Kruskal-Wallis test laat toe om te kijken of er een significant verschil is in het aantal gereproduceerde woorden. We kunnen de test uitvoeren op

- de subsets per Process
- de subsets per Process en Age

Deze test zal aanduiden of er een significant verschil is in het gemiddeld aantal gereproduceerde woorden, maar leert ons niets over het effectieve verschil. Hiervoor moeten we dan de Wilcoxon-Mann-Whitney two-sample tests uitvoeren per 2 subsets.
Aan de hand van de Hodges-Lehman schatter kunnen we dan bekijken wat het verschil is in het aantal gereproduceerde woorden tussen de verschillende subsets.

De rank-gebaseerde methodes veronderstellen geen normaliteit. Als we de Hodges-Lehman schatter gebruiken, dan is er uiteraard wel een veronderstelling van locatie-shift, dus in dit geval wordt wel verondersteld dat de varianties van de verschillende subsets gelijk zijn.

Voorbereiding - data splitsen

We kunnen de Kruskal-Wallis en Wilcoxon tests uitvoeren op verschillende subsets. Bv.

- subsets per proces
- subsets per leeftijd (invloed van process, voor een specifieke leeftijdscategorie)
- subsets per proces en leeftijd (invloed van proces en leeftijd)

Daarvoor definiëren we de subsets als volgt :

```
memory.by.age.process <- lapply(split(memory, list(memory$Age, memory$Process)), '[[', "Words")
memory.by.age <- lapply(split(memory, memory$Age), '[[', "Words")
memory.by.process <- lapply(split(memory, memory$Process), '[[', "Words")
memory.by.process.older <- lapply(split(memory.older, memory.older$Process), '[[', "Words")
memory.by.process.younger <- lapply(split(memory.younger, memory.younger$Process), '[[', "Words")
```

Om efficiënter de verschillende Wilcoxon-Mann-Whitney testen te kunnen uitvoeren, en de resultaten in een data.frame terug te krijgen, definiëren we de functie :

```
wc.tests <- function(subsets, adjust.method = "bonferroni") {
  ind.comb <- combn(1:length(subsets), 2)

  pw.result <- data.frame()

  pairwise.tests <- lapply(1:ncol(ind.comb), function(i) {
    wc.test = wilcox.test(subsets[[ind.comb[1,i]]],
                          subsets[[ind.comb[2,i]]], conf.int = TRUE)
```

```
    pw.test <- data.frame(
      subset1 = names(subsets)[ind.comb[1,i]],
      subset2 = names(subsets)[ind.comb[2,i]],
      p.value = wc.test$p.value,
      lower = round(wc.test$conf.int[1],3),
      upper = round(wc.test$conf.int[2],3),
      location.shift = round(wc.test$estimate,3)
    )

    pw.test
  })

  # Add adjusted p-values, based on the chosen method
  pw.df <- do.call("bind_rows", pairwise.tests)
  pw.df$p.value.adj <- p.adjust(pw.df$p.value, adjust.method)

  pw.df
}
```

Kruskal-Wallis testen

Per process

De Kruskal-Wallis test voor de subsets per Process

```
kruskal.test(memory.by.process)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  memory.by.process
## Kruskal-Wallis chi-squared = 35.567, df = 2, p-value = 1.892e-08
```

Zonder rekening te houden met leeftijd, is er een significant bewijs dat minstens 1 process een verschillend gemiddelde heeft voor het aantal gereproduceerde woorden.

Effect van leeftijd

Om te controleren of er een effect is van leeftijd, doen we dezelfde test opnieuw, maar eens voor de data overeenkomend met jongeren, en de tweede keer voor de data van de oudere testpersonen.

```
kruskal.test(Words ~ Process, data = memory.younger)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  Words by Process
## Kruskal-Wallis chi-squared = 20.388, df = 2, p-value = 3.74e-05
```

```
kruskal.test(Words ~ Process, data = memory.older)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  Words by Process
## Kruskal-Wallis chi-squared = 15.642, df = 2, p-value = 0.0004013
```

Het effect is meer uitgesproken voor jongeren, maar in beide gevallen significant.

Effect van proces en leeftijd

Overeenkomend met de interactie term in het lineaire model, kunnen we de Kruskal-Wallis test ook uitvoeren op de subsets per combinatie van Process en Age.

```
kruskal.test(memory.by.age.process)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  memory.by.age.process
## Kruskal-Wallis chi-squared = 44.851, df = 5, p-value = 1.556e-08
```

Wilcoxon-Mann-Whitney testen

Uitgevoerd voor alle combinaties van Process en Age :

```
wc.df.age.process <- wc.tests(memory.by.age.process) %>%
  mutate(significant = p.value.adj < 0.05 )
wc.df.age.process%>%
  kable(caption = "Wilcoxon-Mann-Whitney tests voor combinaties (Age, Process)")
```

Table 5: Wilcoxon-Mann-Whitney tests voor combinaties (Age, Process)

subset1	subset2	p.value	lower	upper	location.shift	p.value.adj	significant
Older.Counting	Younger.Counting	0.5380792	-1	2	0.500	1.0000000	FALSE
Older.Counting	Older.Imagery	0.0003131	-9	-3	-5.000	0.0046961	TRUE
Older.Counting	Younger.Imagery	0.0001746	-13	-8	-10.000	0.0026194	TRUE
Older.Counting	Older.Intentional	0.0023768	-8	-2	-5.000	0.0356517	TRUE
Older.Counting	Younger.Intentional	0.0001746	-15	-10	-12.749	0.0026194	TRUE
Younger.Counting	Older.Imagery	0.0001963	-10	-4	-5.000	0.0029448	TRUE
Younger.Counting	Younger.Imagery	0.0001697	-13	-9	-11.000	0.0025459	TRUE
Younger.Counting	Older.Intentional	0.0018274	-8	-3	-5.000	0.0274113	TRUE
Younger.Counting	Younger.Intentional	0.0001697	-15	-10	-13.000	0.0025459	TRUE
Older.Imagery	Younger.Imagery	0.0248079	-8	-1	-5.000	0.3721191	FALSE
Older.Imagery	Older.Intentional	0.6175675	-3	5	1.000	1.0000000	FALSE
Older.Imagery	Younger.Intentional	0.0109634	-10	-3	-6.484	0.1644506	FALSE
Younger.Imagery	Older.Intentional	0.0020693	2	9	5.000	0.0310402	TRUE
Younger.Imagery	Younger.Intentional	0.1695524	-5	1	-2.000	1.0000000	FALSE
Older.Intentional	Younger.Intentional	0.0007237	-11	-4	-7.000	0.0108553	TRUE

Leeftijd is volgens deze test niet altijd een significante parameter.

Voor alle Processen, gelimiteerd tot de jongeren

```
wc.df.process.younger <- wc.tests(memory.by.process.younger) %>%
  mutate(significant = p.value.adj < 0.05)
wc.df.process.younger%>%
  kable(caption = "Wilcoxon-Mann-Whitney tests voor processen, beperkt tot jongeren")
```

Table 6: Wilcoxon-Mann-Whitney tests voor processen, beperkt tot jongeren

subset1	subset2	p.value	lower	upper	location.shift	p.value.adj	significant
Counting	Imagery	0.0001697	-13	-9	-11	0.0005092	TRUE
Counting	Intentional	0.0001697	-15	-10	-13	0.0005092	TRUE
Imagery	Intentional	0.1695524	-5	1	-2	0.5086571	FALSE

Voor alle Processen, gelimiteerd tot de ouderen

```
wc.df.process.older <- wc.tests(memory.by.process.older) %>%
  mutate(significant = p.value.adj < 0.05)
wc.df.process.older %>%
  kable(caption = "Wilcoxon-Mann-Whitney tests voor processen, beperkt tot ouderen")
```

Table 7: Wilcoxon-Mann-Whitney tests voor processen, beperkt tot ouderen

subset1	subset2	p.value	lower	upper	location.shift	p.value.adj	significant
Counting	Imagery	0.0003131	-9	-3	-5	0.0009392	TRUE
Counting	Intentional	0.0023768	-8	-2	-5	0.0071303	TRUE
Imagery	Intentional	0.6175675	-3	5	1	1.0000000	FALSE

Er is zowel voor jongeren als voor ouderen enkel een significant verschil tussen het Counting proces en de twee andere processen, maar niet tussen de Imagery en Intentional processen onderling.

Vergelijking van de methodes

We kunnen dit vergelijken met de resultaten van het lineaire model :

```
lm.df.age.process <- lm.tests(mem.lm.int, test.data.age.process)
lm.df.process.younger <- lm.tests(mem.lm.int, test.data.process.younger)
lm.df.process.older <- lm.tests(mem.lm.int, test.data.process.older)
```

via een inner join met het resultaat van de Wilcoxon-Mann-Whitney tests :

Voor de jongeren :

```
comp.df.process.younger <- lm.df.process.younger %>%
  inner_join(wc.df.process.younger, by = c("subset1","subset2"))
comp.df.process.younger %>%
  kable(caption = "Vergelijking lm met wilcoxon test, jongeren")
```

Table 8: Vergelijking lm met wilcoxon test, jongeren

subset1	subset2	diff	p.value	lower	upper	location.shift	p.value.adj	significant
Counting	Imagery	-11.1	0.0001697	-13	-9	-11	0.0005092	TRUE
Counting	Intentional	-12.8	0.0001697	-15	-10	-13	0.0005092	TRUE
Imagery	Intentional	-1.7	0.1695524	-5	1	-2	0.5086571	FALSE

Voor de ouderen

```
comp.df.process.older <- lm.df.process.older %>%
  inner_join(wc.df.process.older, by = c("subset1","subset2"))
comp.df.process.older %>%
  kable(caption = "Vergelijking lm met wilcoxon test, ouderen")
```

Table 9: Vergelijking lm met wilcoxon test, ouderen

subset1	subset2	diff	p.value	lower	upper	location.shift	p.value.adj	significant
Counting	Imagery	-6.4	0.0003131	-9	-3	-5	0.0009392	TRUE
Counting	Intentional	-5.0	0.0023768	-8	-2	-5	0.0071303	TRUE
Imagery	Intentional	1.4	0.6175675	-3	5	1	1.0000000	FALSE

Voor de combinaties (Age, Process)

```
comp.df.age.process <- lm.df.age.process %>%
  inner_join(wc.df.age.process, by = c("subset1","subset2"))
comp.df.age.process %>%
  kable(caption = "Vergelijking lm met wilcoxon test, combinaties Age, Process")
```

Table 10: Vergelijking lm met wilcoxon test, combinaties Age, Process

subset1	subset2	diff	p.value	lower	upper	location.shift	p.value.adj	significant
Older.Counting	Younger.Counting	0.5	0.5380792	-1	2	0.500	1.0000000	FALSE
Older.Counting	Older.Imagery	-6.4	0.0003131	-9	-3	-5.000	0.0046961	TRUE
Older.Counting	Younger.Imagery	-10.6	0.0001746	-13	-8	-10.000	0.0026194	TRUE
Older.Counting	Older.Intentional	-5.0	0.0023768	-8	-2	-5.000	0.0356517	TRUE
Older.Counting	Younger.Intentional	-12.3	0.0001746	-15	-10	-12.749	0.0026194	TRUE
Younger.Counting	Older.Imagery	-6.9	0.0001963	-10	-4	-5.000	0.0029448	TRUE
Younger.Counting	Younger.Imagery	-11.1	0.0001697	-13	-9	-11.000	0.0025459	TRUE
Younger.Counting	Older.Intentional	-5.5	0.0018274	-8	-3	-5.000	0.0274113	TRUE
Younger.Counting	Younger.Intentional	-12.8	0.0001697	-15	-10	-13.000	0.0025459	TRUE
Older.Imagery	Younger.Imagery	-4.2	0.0248079	-8	-1	-5.000	0.3721191	FALSE
Older.Imagery	Older.Intentional	1.4	0.6175675	-3	5	1.000	1.0000000	FALSE
Older.Imagery	Younger.Intentional	-5.9	0.0109634	-10	-3	-6.484	0.1644506	FALSE
Younger.Imagery	Older.Intentional	5.6	0.0020693	2	9	5.000	0.0310402	TRUE
Younger.Imagery	Younger.Intentional	-1.7	0.1695524	-5	1	-2.000	1.0000000	FALSE
Older.Intentional	Younger.Intentional	-7.3	0.0007237	-11	-4	-7.000	0.0108553	TRUE

Deze tabel geeft weer voor welke subsets een significant verschil kan worden aangeduid (in dit geval op basis van de bonferroni-aangepaste p-waarde).
Het Counting proces is significant verschillend van de Imagery en Intentional processen, voor elke leeftijdscategorie.
Het Intentional proces voor ouderen is significant minder efficiënt dan wat de jongeren presteren in voor het Imagery en Intentional proces.

Conclusie

Beide methodes geven aan dat er een significant effect is van het proces, en van de leeftijd.
De methode via lineaire regressie laat enkel toe om na te gaan of er een significant effect is tov. een

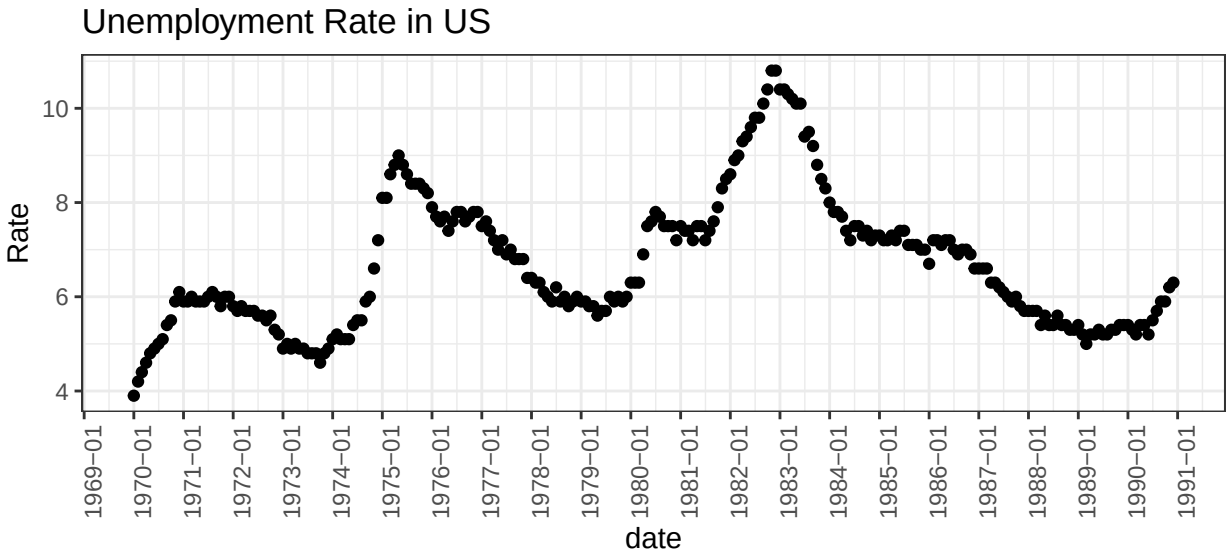
referentietoestand. De Wilcoxon-Mann-Whitney tests daarentegen laten toe om alle subsets met elkaar te vergelijken.

Appendix 2 - Werkloosheid - Gedetailleerde uitwerking

Data Visualisatie

```
load(file = "unemploymentUS.rda")

unemploymentUS <- unemploymentUS %>%
  mutate(date = ymd(paste(Year, Monthly, "01", sep="/")),
         date.number = Year + Monthly / 12)
```

```
ggplot(unemploymentUS, aes(date, Rate)) +
  geom_point() +
  scale_x_date(labels = date_format("%Y-%m"), date_breaks = "year") +
  theme(axis.text.x = element_text(angle = 90)) +
  ggtitle("Unemployment Rate in US")
```



We zien een duidelijk niet-lineair verloop. Er lijkt een periodieke component te zijn, maar de periode is groter dan 1 jaar.

Gezien de niet-lineariteit is een gam-model aangewezen. Er zijn 2 onafhankelijke variabelen, namelijk Year en Monthly. We kunnen een aantal verschillende modellen opstellen

- Year
- Year en Monthly als onafhankelijke variabelen
- effectieve datum (of numeriek : Year + Monthly/12) als onafhankelijke variabele

De modellen kunnen vergeleken worden via AIC.

gam model met splines voor Year en Monthly

```
ue.mod <- gam(Rate ~ s(Year) + s(Monthly), data = unemploymentUS)
summary(ue.mod)
```

```
##
## Family: gaussian
```

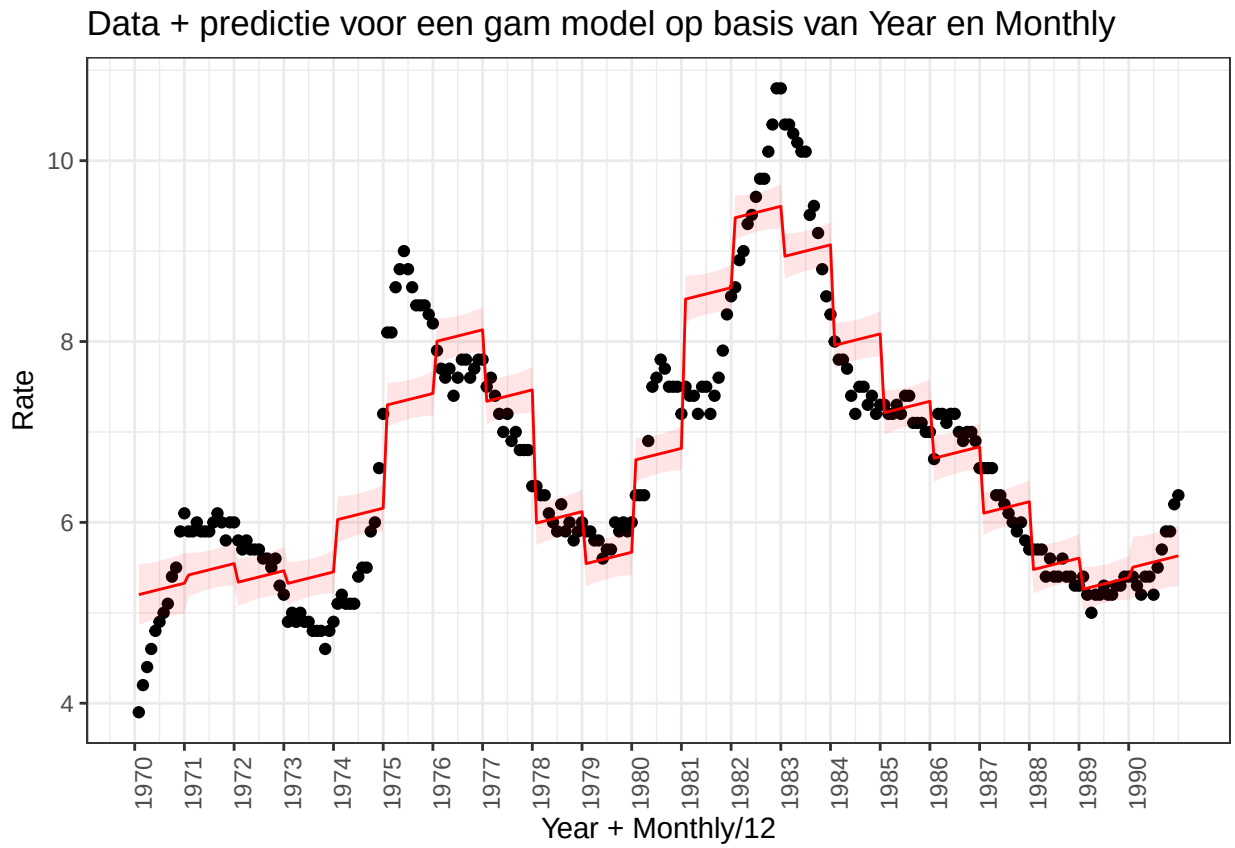
```
## Link function: identity
##
## Formula:
## Rate ~ s(Year) + s(Monthly)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.6913     0.0374   178.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df       F p-value
## s(Year)    8.952  8.999 132.011  <2e-16 ***
## s(Monthly) 1.000  1.000   1.118   0.291
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.825   Deviance explained = 83.2%
## GCV = 0.36851  Scale est. = 0.3525    n = 252
```

```
anova(ue.mod)
```

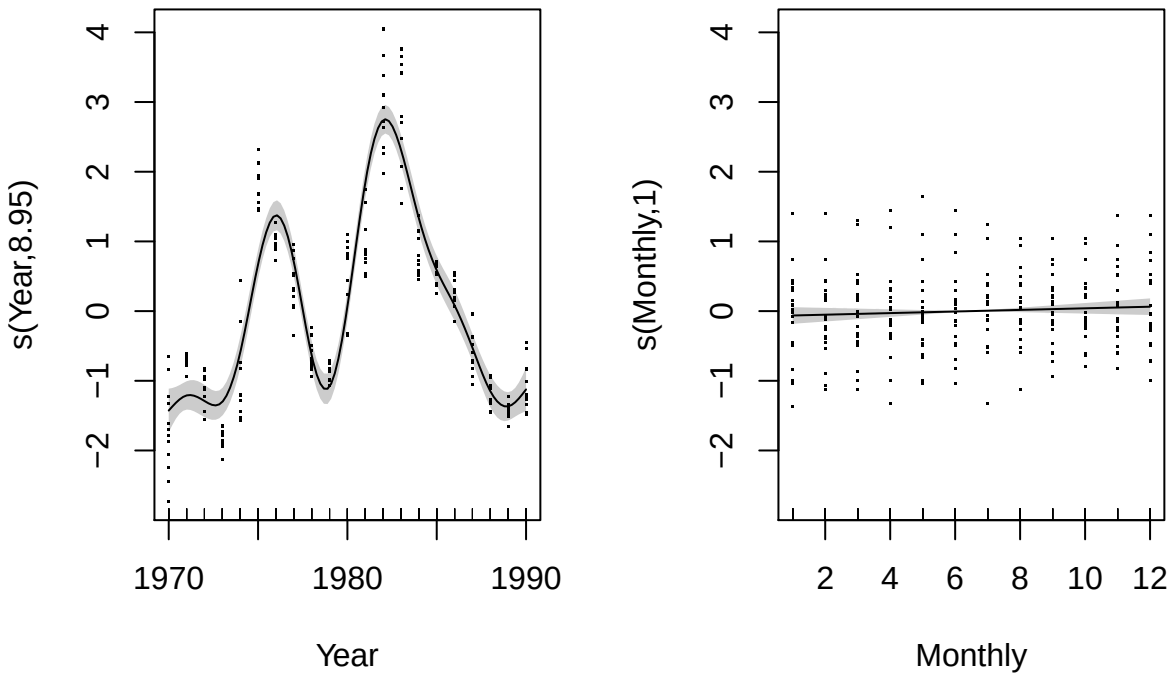
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Rate ~ s(Year) + s(Monthly)
##
## Approximate significance of smooth terms:
##              edf Ref.df       F p-value
## s(Year)    8.952  8.999 132.011  <2e-16
## s(Monthly) 1.000  1.000   1.118   0.291
```

Er is een significant verband met Year, maar niet met Monthly.

```
ue.mod.pred = predict(ue.mod, se.fit = TRUE)
ue.mod.data <- unemploymentUS %>%
  mutate(fit = ue.mod.pred$fit,
         l = ue.mod.pred$fit - 1.96 * ue.mod.pred$se.fit,
         u = ue.mod.pred$fit + 1.96 * ue.mod.pred$se.fit,
         resid = residuals(ue.mod))

ggplot(ue.mod.data, aes(Year + Monthly / 12)) +
  geom_point(aes(y = Rate)) +
  geom_line(aes(y = fit), color = "red") +
  geom_ribbon(aes(ymin = l, ymax = u), fill = "red", alpha = 0.1) +
  scale_x_continuous(breaks = 1970:1990) +
  theme(axis.text.x = element_text(angle = 90)) +
  ggtitle("Data + predictie voor een gam model op basis van Year en Monthly")
```




```
par(mfrow=c(1,2))
plot(ue.mod, residuals = TRUE, select = 1, shade = TRUE)
plot(ue.mod, residuals = TRUE, select = 2, shade = TRUE)
```



gam model met enkel Year

```
ue.mod.year <- gam(Rate ~ s(Year), data = unemploymentUS)
summary(ue.mod.year)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Rate ~ s(Year)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.69127    0.03741   178.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##           edf Ref.df     F p-value
## s(Year) 8.952  8.999 131.9  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.825   Deviance explained = 83.1%
```

```
## GCV = 0.36717  Scale est. = 0.35267   n = 252
```

```
anova(ue.mod.year)
```

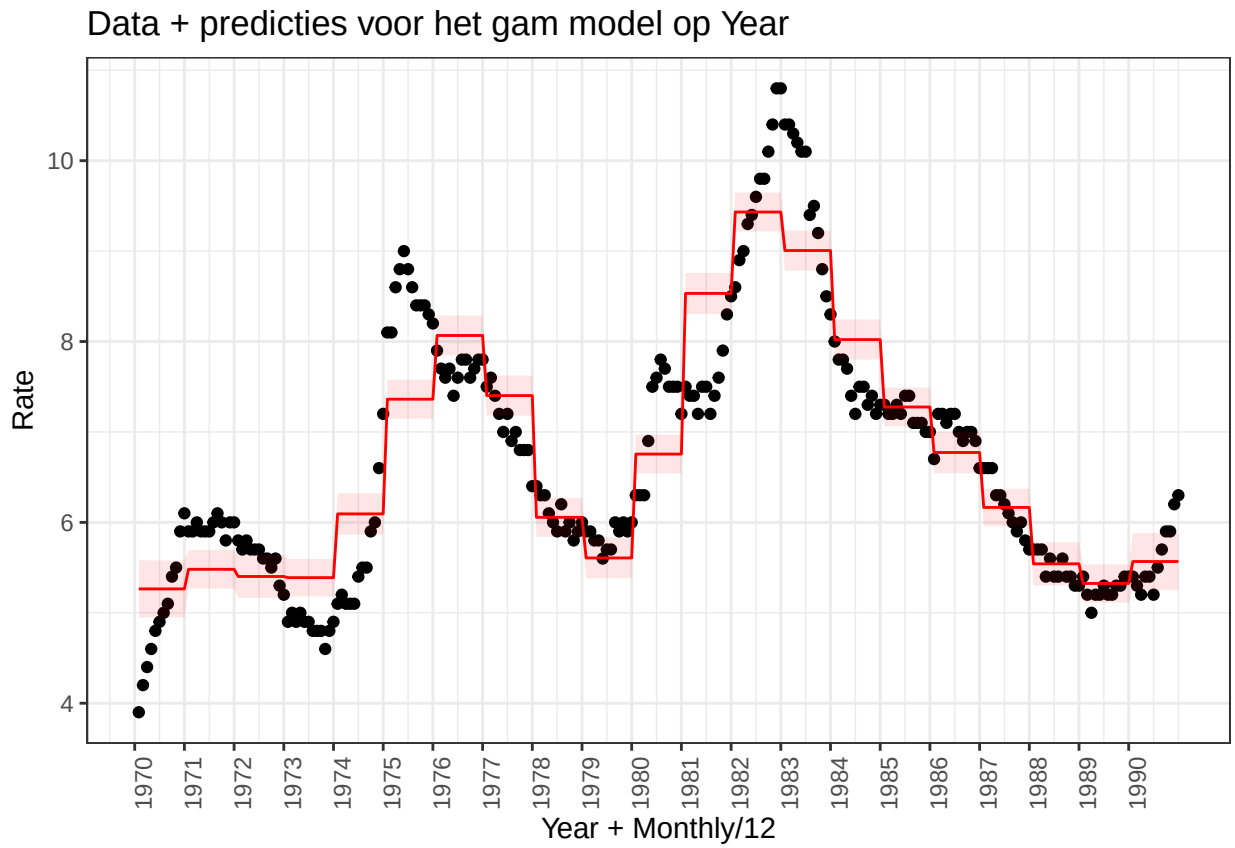
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Rate ~ s(Year)
##
## Approximate significance of smooth terms:
##           edf Ref.df     F p-value
## s(Year) 8.952  8.999 131.9  <2e-16
```

De invloed van Year is significant. Het model heeft een iets lagere AIC dan het model met Year en Monthly.

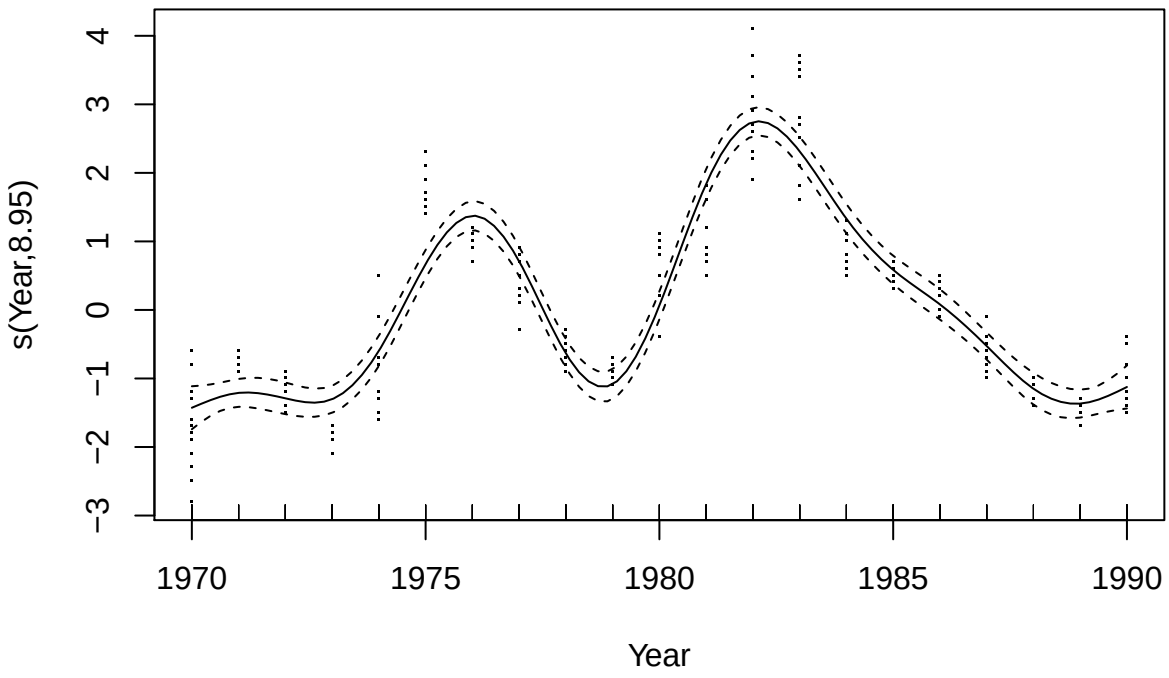
```
AIC(ue.mod, ue.mod.year)
```

```
##                    df      AIC
## ue.mod      11.95173 465.0867
## ue.mod.year 10.95171 464.2527
```

```
ue.mod.year.pred = predict(ue.mod.year, se.fit = TRUE)

ue.mod.year.data <- unemploymentUS %>%
  mutate(fit = ue.mod.year.pred$fit,
         l = ue.mod.year.pred$fit - 1.96 * ue.mod.year.pred$se.fit,
         u = ue.mod.year.pred$fit + 1.96 * ue.mod.year.pred$se.fit,
         resid = residuals(ue.mod.year))

ggplot(ue.mod.year.data, aes(Year + Monthly / 12)) +
  geom_point(aes(y = Rate)) +
  geom_line(aes(y = fit), color = "red") +
  geom_ribbon(aes(ymin = l, ymax = u), fill = "red", alpha = 0.1) +
  scale_x_continuous(breaks = 1970:1990) +
  theme(axis.text.x = element_text(angle = 90)) +
  ggtitle("Data + predicties voor het gam model op Year")
```




```
plot(ue.mod.year, residuals = TRUE)
```



Is het model significant niet-lineair ?

```
ue.mod.year.lin <- gam(Rate ~ Year, data = unemploymentUS)
summary(ue.mod.year.lin)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Rate ~ Year
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -60.48730   28.96633  -2.088   0.0378 *
## Year          0.03393    0.01463   2.319   0.0212 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## R-sq.(adj) =  0.0171   Deviance explained = 2.11%
## GCV = 1.9934  Scale est. = 1.9775    n = 252
```

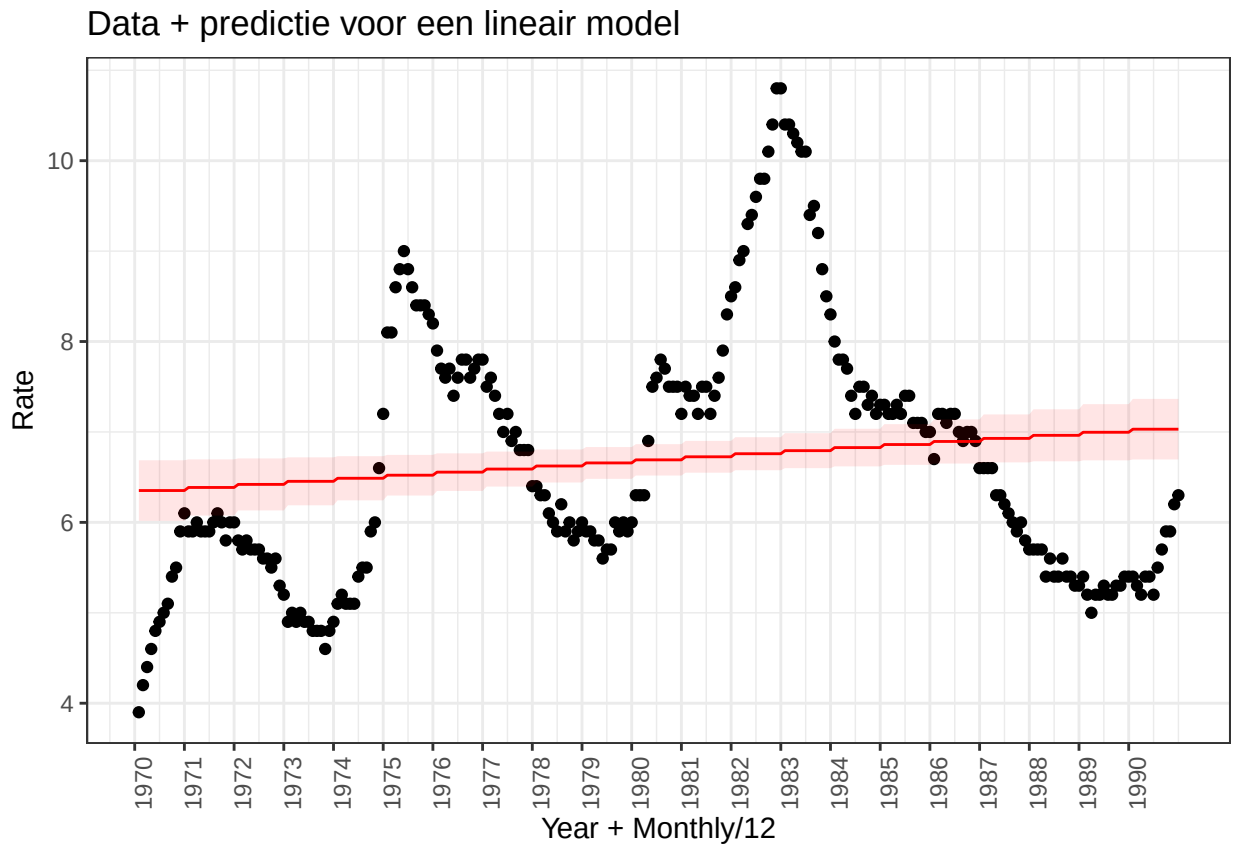
```
anova(ue.mod.year.lin)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Rate ~ Year
##
## Parametric Terms:
##      df     F p-value
## Year  1 5.379  0.0212
```

In het zuiver lineaire model zijn de coëfficiënten zwak significant.

Zoals visueel al duidelijk was, kunnen we er beter van uitgaan dat de afhankelijkheid niet-lineair is.

```
ue.mod.year.lin.pred = predict(ue.mod.year.lin, se.fit = TRUE)
ue.mod.year.lin.data <- unemploymentUS %>%
  mutate(fit = ue.mod.year.lin.pred$fit,
         l = ue.mod.year.lin.pred$fit - 1.96 * ue.mod.year.lin.pred$se.fit,
         u = ue.mod.year.lin.pred$fit + 1.96 * ue.mod.year.lin.pred$se.fit)

ggplot(ue.mod.year.lin.data, aes(Year + Monthly / 12)) +
  geom_point(aes(y = Rate)) +
  geom_line(aes(y = fit), color = "red") +
  geom_ribbon(aes(ymin = l, ymax = u), fill = "red", alpha = 0.1) +
  scale_x_continuous(breaks = 1970:1990) +
  theme(axis.text.x = element_text(angle = 90)) +
  ggtitle("Data + predictie voor een lineair model")
```



Gam model op tijd

Aan gezien de periode groter is dan een jaar, kan het effect van maand misschien beter inbegrepen worden door niet te fitten op Year, maar op Year + Monthly / 12

```
ue.mod.datenr <- gam(Rate ~ s(date.number), data = unemploymentUS)
summary(ue.mod.datenr)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Rate ~ s(date.number)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    6.691      0.034   196.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                   edf Ref.df     F p-value
## s(date.number) 8.969      9 165.5  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

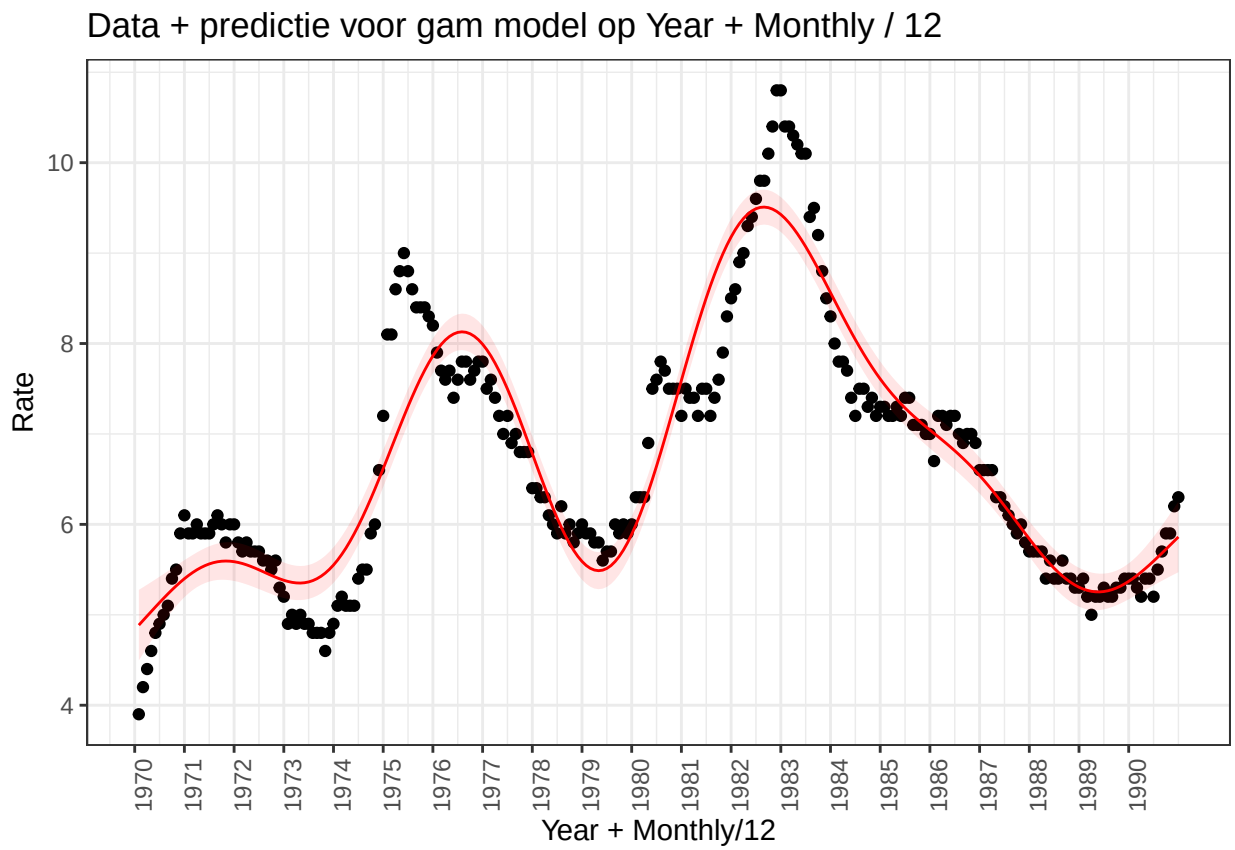
```
##
## R-sq.(adj) =  0.855   Deviance explained =   86%
## GCV = 0.30332  Scale est. = 0.29132   n = 252
```

```
anova(ue.mod.datenr)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Rate ~ s(date.number)
##
## Approximate significance of smooth terms:
##                   edf Ref.df     F p-value
## s(date.number) 8.969  9.000 165.5  <2e-16
```

De coëfficiënten in dit model zijn eveneens significant

```
ue.mod.datenr.pred = predict(ue.mod.datenr, se.fit = TRUE)

ue.mod.datenr.data <- unemploymentUS %>%
  mutate(fit = ue.mod.datenr.pred$fit,
         l = ue.mod.datenr.pred$fit - 1.96 * ue.mod.datenr.pred$se.fit,
         u = ue.mod.datenr.pred$fit + 1.96 * ue.mod.datenr.pred$se.fit)

ggplot(ue.mod.datenr.data, aes(Year + Monthly / 12)) +
  geom_point(aes(y = Rate)) +
  geom_line(aes(y = fit), color = "red") +
  geom_ribbon(aes(ymin = l, ymax = u), fill = "red", alpha = 0.1) +
  scale_x_continuous(breaks = 1970:1990) +
  theme(axis.text.x = element_text(angle = 90)) +
  ggtitle("Data + predictie voor gam model op Year + Monthly / 12")
```




```
# plot(ue.mod.datenr.pred)
```

Vergelijking

```
AIC(ue.mod, ue.mod.datenr, ue.mod.year)
```

```
##                     df      AIC
## ue.mod        11.95173 465.0867
## ue.mod.datenr 10.96919 416.1111
## ue.mod.year   10.95171 464.2527
```

Het model op basis van Year + Monthly /12 is het beste.

Conclusie

- Er is een niet-lineair verband tussen werkloosheidgraad en de tijd
- De periode van de fluctuaties is groter dan 1 jaar, dus de invloed van het jaar is significant groter dan deze van de maand

Appendix 3 - Monte Carlo simulatie - Gedetailleerde uitwerking

Voorbereiding: Lognormale verdeling

Voor de lognormale verdeling zullen we een functie definiëren die gebruik maakt van de functie rlnorm. Ik had ook de in de opgave voorgestelde functie kunnen gebruiken, maar om beter het verband tussen normale en lognormale te begrijpen ik ervoor gekozen om de functie rlnorm wat dieper te onderzoeken.

```
rlnorm(n, meanlog = 0, sdlog = 1)
```

Hierin zijn meanlog en sdlog het gemiddelde en de standaardafwijking van de normale verdeling die de basis vormt voor de lognormale verdeling, d.w.z.

als

$$X \sim N(\mu, \sigma^2)$$

dan is

$$Y = e^X \sim LogN(\mu, \sigma^2)$$

Hierbij zijn μ en σ de parameters van de geassocieerde normale verdeling.

Het zou natuurlijk interessanter zijn om een functie te hebben met als parameters het gemiddelde en de standaardafwijking van de log-normale verdeling zelf. Daartoe moeten de parameters van de normale verdeling uitgedrukt worden in functie van deze van de lognormale.

Als we het gemiddelde en de standaardafwijking van de normale verdeling aanduiden als μ_N en σ_N, dan zijn gemiddelde en standaardafwijking van de lognormale verdeling gegeven door

$$\begin{aligned} \mu_L &= e^{\mu_N + \frac{\sigma_N^2}{2}} \\ \sigma_L^2 &= \left(e^{\sigma_N^2} - 1\right) e^{2\mu_N + \sigma_N^2} \end{aligned}$$

De laatste formule kan herwerkt worden tot

$$e^{\mu_N} = \mu_L e^{-\frac{\sigma_N^2}{2}}$$

en

$$\sigma_L^2 = \left(e^{\sigma_N^2} - 1\right) \mu_L^2$$

Dit laat toe om de parameters van de normale distributie te schrijven in functie van deze van de log-normale distributie :

$$\begin{aligned} e^{\sigma_N^2} &= 1 + \frac{\sigma_L^2}{\mu_L^2} \\ e^{\mu_N} &= \frac{\mu_L}{\sqrt{1 + \frac{\sigma_L^2}{\mu_L^2}}} \end{aligned}$$

en finaal

$$\sigma_N = \sqrt{ln\left(1 + \frac{\sigma_L^2}{\mu_L^2}\right)}$$

$$\mu_N = ln\left(\frac{\mu_L}{\sqrt{1 + \frac{\sigma_L^2}{\mu_L^2}}}\right)$$

Aan de hand hiervan definiëren we de volgende functies

```
# Random generator voor de Log-normale distributie
rlnorm2 <- function(n, mu = 1, sd = 1) {
  exp_sdlog_sq <- 1 + sd^2/mu^2
  sdlog <- sqrt(log(exp_sdlog_sq))
  meanlog <- log(mu/sqrt(exp_sdlog_sq))

  rlnorm(n, meanlog, sdlog)
}

# Density functie voor de Log-normale distributie
dlnorm2 <- function(n, mu = 1, sd = 1) {
  exp_sdlog_sq <- 1 + sd^2/mu^2
  sdlog <- sqrt(log(exp_sdlog_sq))
  meanlog <- log(mu/sqrt(exp_sdlog_sq))

  dlnorm(n, meanlog, sdlog)
}
```

We kunnen dit testen op een aantal verschillende combinaties voor de parameters μ_L en σ_L. Eerst maken we een data frame met verschillende combinaties voor μ_L en σ_L. Dan gebruiken we de functie sapply om de gemiddelde waarden te berekenen voor een sample. Als de functie goed gedefinieerd is, moeten de gemeten gemiddelde waarde en standaardafwijking overeenkomen met de waarden van de parameters.

Het resultaat van deze berekening is een matrix, waarbij de eerste 2 rijen de parameters weergeven, en de laatste 2 rijen de (afgeronde) gemeten waarden.

```
test.params <- expand.grid(c(1,2,3),c(0.5,1,2))

sapply(1:nrow(test.params), function(i) {
  mu <- test.params[i,1]
  sd <- test.params[i,2]
  test.data <- rlnorm2(100000, mu,sd)
  list(mu = mu, sd = sd, sample_mu = round(mean(test.data),1), sample_sd = round(sd(test.data),1))
})
```

```
##           [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
## mu        1    2    3    1    2    3    1    2    3
## sd        0.5  0.5  0.5  1    1    1    2    2    2
## sample_mu 1    2    3    1    2    3    1    2    3
## sample_sd 0.5  0.5  0.5  1    1    1    2    2    2
```

De gemeten waarde komt inderdaad overeen met de waarden van de parameters, dus de functie werkt zoals het hoort.

Monte-Carlo simulatie functie

Er wordt gevraagd om Monte-Carlo simulaties uit te voeren in een aantal verschillende situaties. Hiervoor definiëren we een functie, met volgende parameters (algemener dan in de vraagstelling)

- dist (waarden **norm** of **lnorm**) : Data uit een normale/lognormale verdeling
- n.total : totaal aantal samples voor steekproef 1 en steekproef 2 samen
- sample.ratio (default 1) : verhouding van het aantal samples in steekproef 1 tov. steekproef 2, bv.
  - n.total = 200, sample.ratio = 3 leidt tot 150 samples in steekproef 1 en 50 in steekproef 2
  - de default waarde leidt tot een gelijk aantal samples
- mu : gemiddelde waarde voor beide steekproeven
- sd : standaardafwijking voor steekproef 1
- sd.ratio : verhouding van standaardafwijking voor steekproef 1 tov. die van steekproef 2
  - sd.ratio = 5 betekent $\sigma_1 = 1$ en $\sigma_2 = 1/5$
- test.type (waarden **pooled** of **Welch**) : geeft aan of de t-test gebruik maakt van de "pooled" variantie, of van de Welch benadering voor het aantal vrijheidsgraden.
- R (default 10000) : aantal Monte-Carlo simulaties in elk scenario
- alpha (default 0.05) : p-value drempelwaarde

De functie geeft een list terug met daarin de gebruikte (afgeleide) parameterwaarden en de berekende waarde voor de typeI fouten tov. drempelwaarde alpha. De list bevat daarnaast ook het laatste paar steekproeven, voor controle.

```r
sim.mctest <- function(dist = c("norm", "lnorm"),
                       n.total,
                       sample.ratio = 1,
                       mu = 1,
                       sd = 1,
                       sd.ratio = 1,
                       test.type = c("pooled","Welch"),
                       R = 10000,
                       alpha = 0.05) {

  # calculate sample sizes
  n1 <- n.total / (1 + 1/sample.ratio)
  n2 <- n.total / (1 + sample.ratio)

  test.type <- test.type[1]
  if(!(test.type %in% c("pooled","Welch"))) {
    stop(paste("Invalid test type", test.type))
  }
  # valid test type
  t.test.var.equal <- test.type == "pooled"

  mu1 <- mu
  mu2 <- mu
  sd1 <- sd
  sd2 <- sd / sd.ratio

  dist <- dist[1]
  if (!(dist %in% c("norm", "lnorm"))) {
    stop("invalid distribution")
  }

  # valid distribution type
```

```
  if (dist == "norm") {
    dist.fun <- rnorm
  } else {
    dist.fun <- rlnorm2
  }

  cnt.typeI <- 0
  for (i in 1:R) {
    d1 <- dist.fun(n1, mu1, sd1)
    d2 <- dist.fun(n2, mu2, sd2)

    pval <- t.test(d1, d2, var.equal = t.test.var.equal)$p.value
    if (pval < alpha) {
      cnt.typeI <- cnt.typeI + 1
    }
  }

  # output a list with all simulation parameters
  list(
    n.total = n.total,        # total sample size
    n1 = n1,      # sample 1 size
    n2 = n2,      # sample 2 size,
    dist = dist,  # distribution type
    test.type = test.type, # test type
    d1 = d1,      # last d1 sample
    d2 = d2,      # last d2 sample
    mu1 = mu1,
    mu2 = mu2,
    sd1 = sd1,
    sd2 = sd2,
    typeI.fout.pct = cnt.typeI / R
  )

}
```

Test functie

Om deze functie te testen definiëren we volgende functie. Deze functie resulteert in een ggplot object met daarin de (density) histogrammen van de laatste steekproeven, de theoretische density functie, en de berekende p-value.

```
eval.mctest <- function(test) {
  plot_data <- rbind(data.frame(sample = "sample 1", x = test$d1), data.frame(sample = "sample 2", x = t

  # valid distribution type
  if (test$dist == "norm") {
    dist.fun <- dnorm
  } else {
    dist.fun <- dlnorm2
  }

  dist_data_1 <- tibble(dist = "dist 1",
                        x = seq(test$mu1 - 3*test$sd1, test$mu1 + 3 * test$sd1, by = 0.1),
```

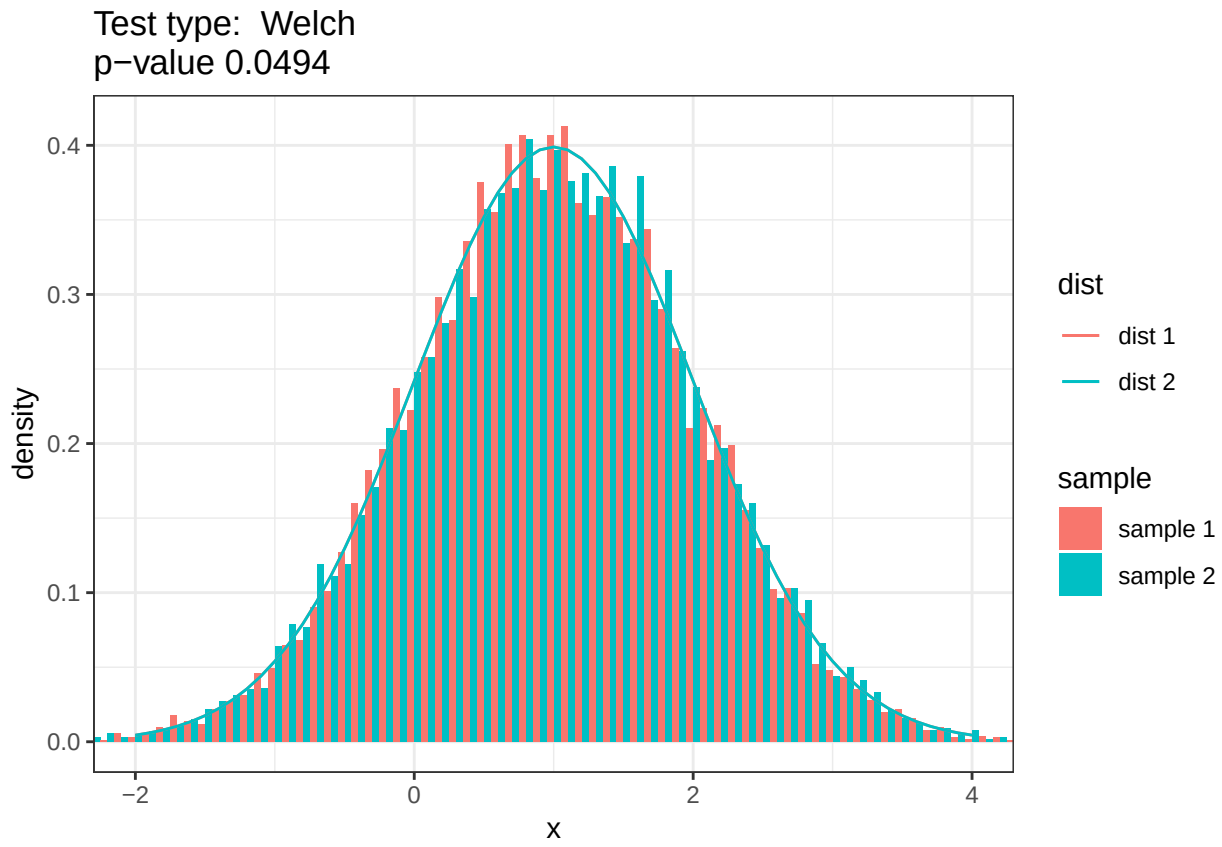
```
                        y = dist.fun(x, test$mu1, test$sd1))
  dist_data_2 <- tibble(dist = "dist 2",
                        x = seq(test$mu2 - 3*test$sd2, test$mu2 + 3 * test$sd2, by = 0.1),
                        y = dist.fun(x, test$mu2, test$sd2))

  display_range <- c(min(c(min(dist_data_1$x), min(dist_data_2$x))),
                     max(c(max(dist_data_1$x), max(dist_data_2$x))))

  p <- ggplot(plot_data, aes(x)) +
    geom_histogram(aes(y = ..density.., fill = sample), position = "dodge", binwidth = 0.1, stat) +
    geom_line(aes(x, y, color = dist), data = dist_data_1) +
    geom_line(aes(x, y, color = dist), data = dist_data_2) +
    coord_cartesian(xlim = display_range) +
    ggtitle(paste("Test type: ", test$test.type, "\np-value", test$typeI.fout.pct))

  p
}
```

Tests

We voeren een aantal tests uit, met grote getallen, omdat in dit geval de berekende p-waarde de verwachte p-waarde van 0.05 goed moet benaderen.

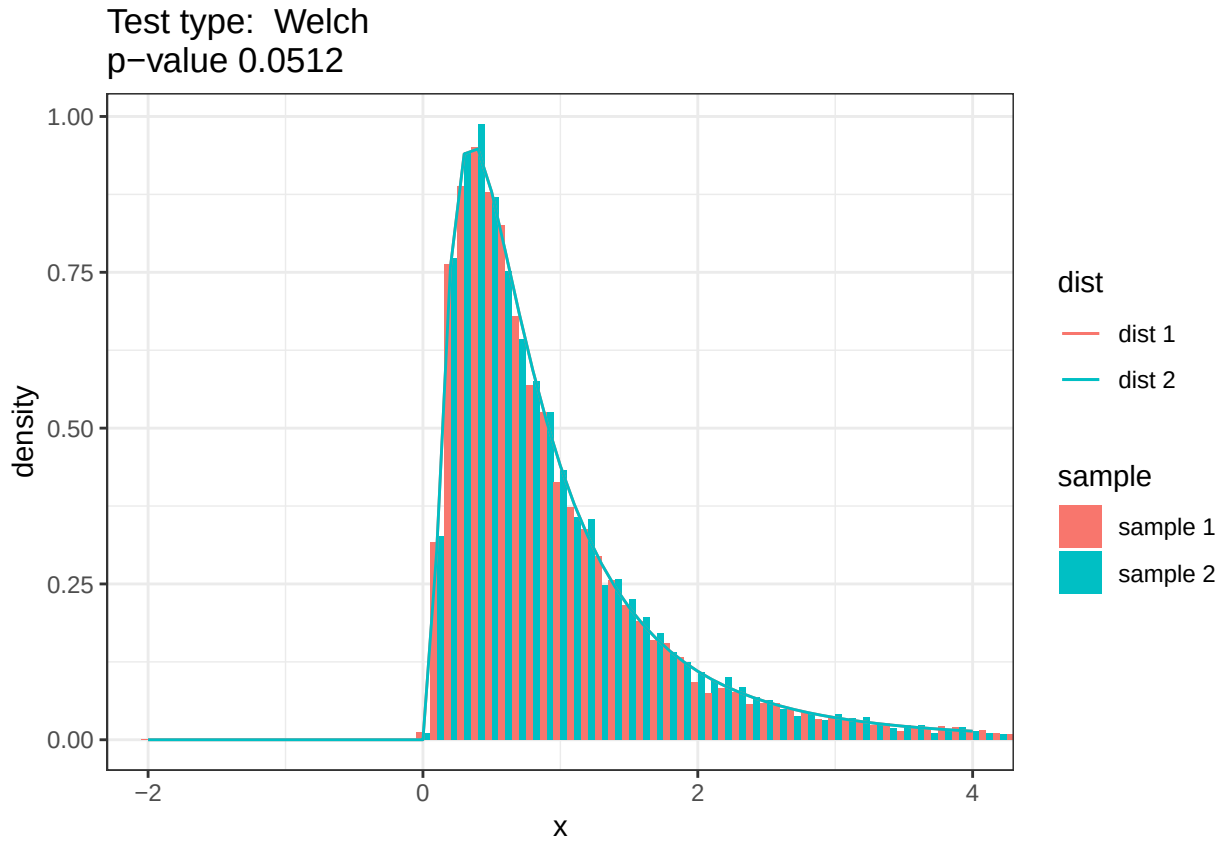
Een eerste test, met de normale verdeling

```
test1 <- sim.mctest(n.total = 20000,
                    sample.ratio = 1,
                    dist = "norm",
                    mu = 1,
                    sd = 1,
                    sd.ratio = 1,
                    test.type = "Welch",
                    R = 10000,
                    alpha = 0.05)
eval.mctest(test1)
```



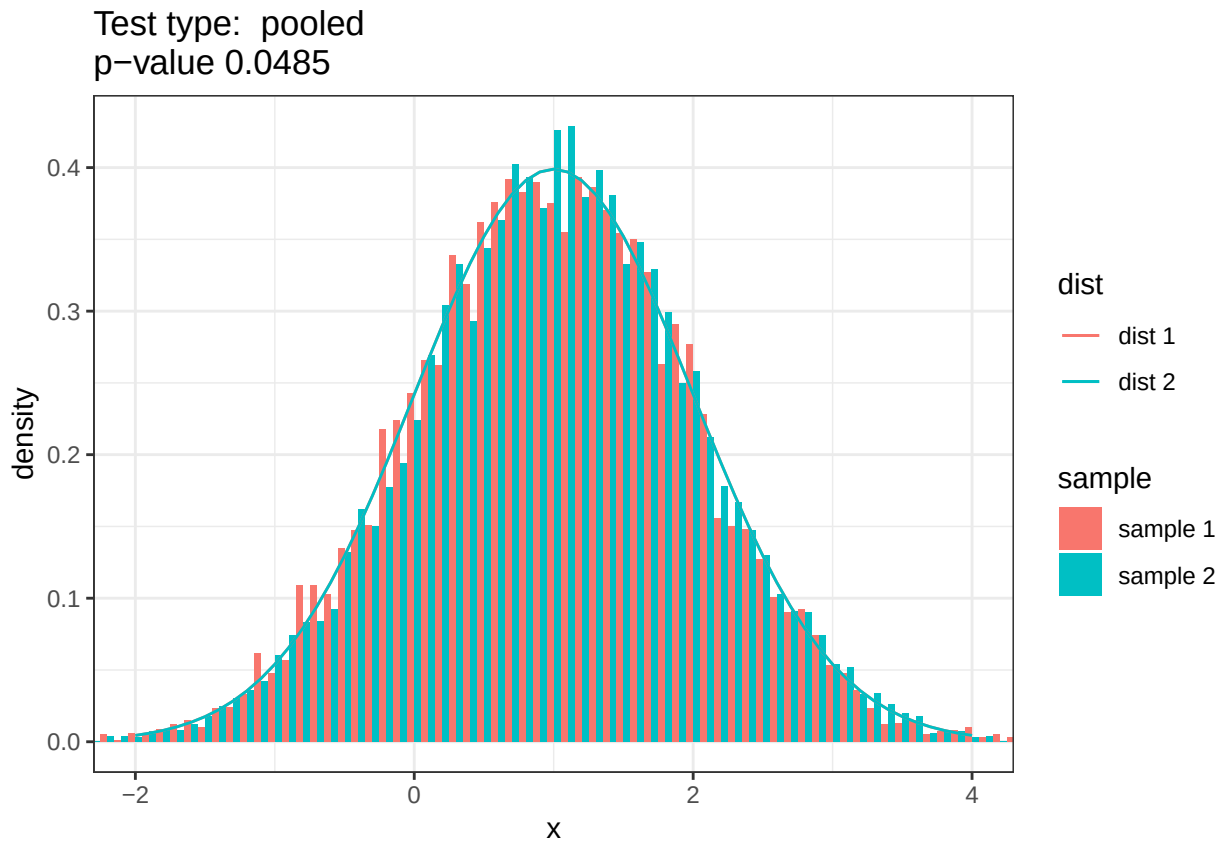
Een tweede test, met de log-normale verdeling

```
test2 <- sim.mctest(n.total = 20000,
                    sample.ratio = 1,
                    dist = "lnorm",
                    mu = 1,
                    sd = 1,
                    sd.ratio = 1,
                    test.type = "Welch",
                    R = 10000,
                    alpha = 0.05)
eval.mctest(test2)
```



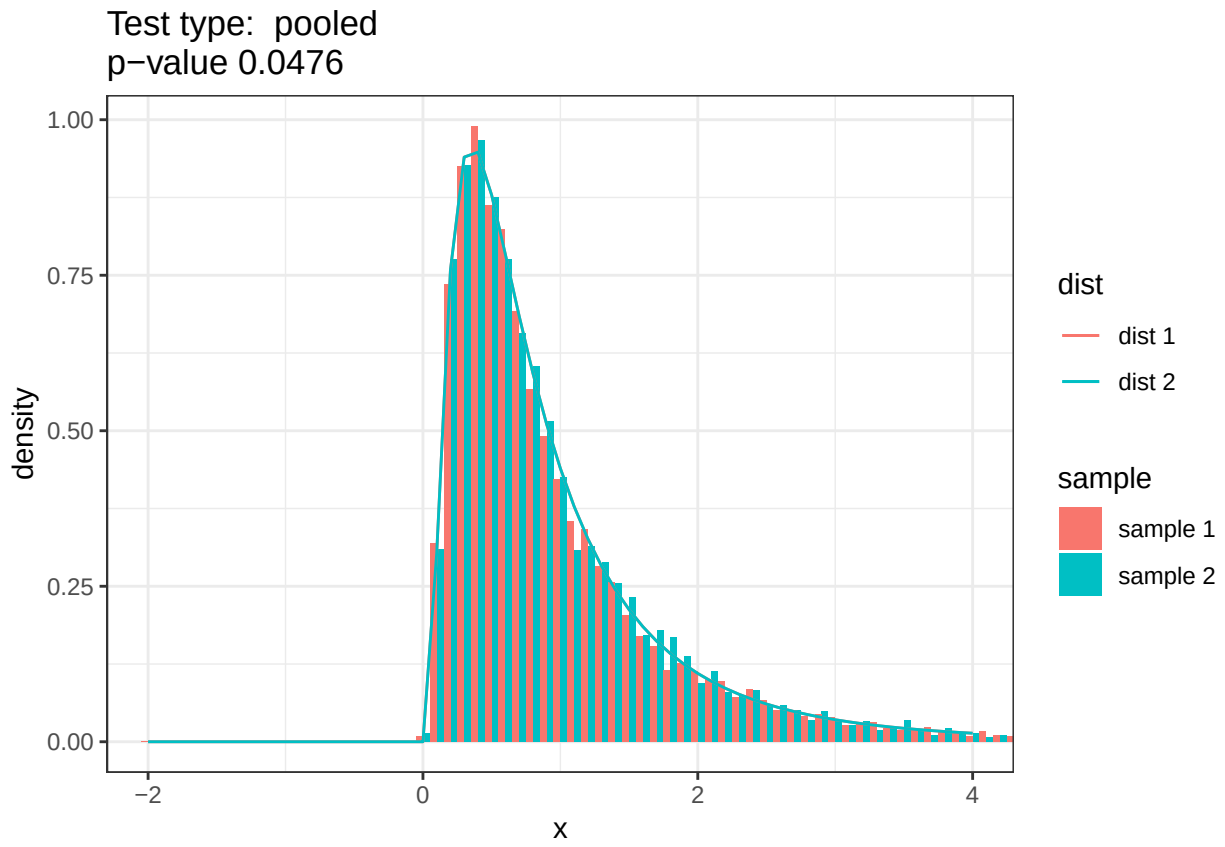
Een test, met de normale verdeling en de pooled-variance two-sample t-test.

```
test3 <- sim.mctest(n.total = 20000,
                    sample.ratio = 1,
                    dist = "norm",
                    mu = 1,
                    sd = 1,
                    sd.ratio = 1,
                    test.type = "pooled",
                    R = 10000,
                    alpha = 0.05)
eval.mctest(test3)
```



Een test, met de normale verdeling en de pooled-variance two-sample t-test en een log-normale verdeling.

```
test4 <- sim.mctest(n.total = 20000,
                    sample.ratio = 1,
                    dist = "lnorm",
                    mu = 1,
                    sd = 1,
                    sd.ratio = 1,
                    test.type = "pooled",
                    R = 10000,
                    alpha = 0.05)
eval.mctest(test4)
```



Scenario's

Om alle scenario's uit te voeren, bouwen we eerst een data frame op met alle combinaties van voorwaarden :

```
params <- expand.grid(
  dist = c("norm","lnorm"),
  sd.ratio = c(1, 5),
  n.total = c(20, 200),
  sample.ratio = c(1, 1/3),
  test.type = c("pooled","Welch")
)
```

En via lapply kunnen we de Monte-Carlo simulatie toepassen met de parameters in dit data frame.

```
sims <- sapply(1:nrow(params), function(i) {
  p <- params[i,]

  p.val <- sim.mctest(n.total = p$n.total,
             sample.ratio = p$sample.ratio,
             dist = p$dist,
             mu = 1,
             sd = 1,
             sd.ratio = p$sd.ratio,
             test.type = p$test.type,
             R = 10000
            )$typeI.fout.pct
```

```
})

result <- params %>%
  mutate(p.val = sims)
```

```
result %>%
  arrange(sd.ratio, sample.ratio, test.type, dist, n.total) %>%
  kable(caption = "Monte-Carlo simulatie voor 32 Scenario's")
```

Table 11: Monte-Carlo simulatie voor 32 Scenario's

dist	sd.ratio	n.total	sample.ratio	test.type	p.val
norm	1	20	0.3333333	pooled	0.0506
norm	1	200	0.3333333	pooled	0.0497
lnorm	1	20	0.3333333	pooled	0.0476
lnorm	1	200	0.3333333	pooled	0.0421
norm	1	20	0.3333333	Welch	0.0538
norm	1	200	0.3333333	Welch	0.0473
lnorm	1	20	0.3333333	Welch	0.0649
lnorm	1	200	0.3333333	Welch	0.0599
norm	1	20	1.0000000	pooled	0.0513
norm	1	200	1.0000000	pooled	0.0517
lnorm	1	20	1.0000000	pooled	0.0375
lnorm	1	200	1.0000000	pooled	0.0459
norm	1	20	1.0000000	Welch	0.0455
norm	1	200	1.0000000	Welch	0.0528
lnorm	1	20	1.0000000	Welch	0.0305
lnorm	1	200	1.0000000	Welch	0.0462
norm	5	20	0.3333333	pooled	0.2731
norm	5	200	0.3333333	pooled	0.2314
lnorm	5	20	0.3333333	pooled	0.3200
lnorm	5	200	0.3333333	pooled	0.2590
norm	5	20	0.3333333	Welch	0.0494
norm	5	200	0.3333333	Welch	0.0502
lnorm	5	20	0.3333333	Welch	0.1395
lnorm	5	200	0.3333333	Welch	0.0784
norm	5	20	1.0000000	pooled	0.0634
norm	5	200	1.0000000	pooled	0.0552
lnorm	5	20	1.0000000	pooled	0.1235
lnorm	5	200	1.0000000	pooled	0.0711
norm	5	20	1.0000000	Welch	0.0509
norm	5	200	1.0000000	Welch	0.0448
lnorm	5	20	1.0000000	Welch	0.1210
lnorm	5	200	1.0000000	Welch	0.0677

We kunnen de invloed van elk van de parameters bekijken door voor elke parameter een succes percentage te berekenen, dwz. het aantal gevallen waarbij de Monte-Carlo simulatie een "aanvaardbare" p-waarde oplevert.

```
success.rate <- function(p.val.threshold) {
  do.call("bind_rows", lapply(names(params), function(p) {
    res.table <- result %>%
      mutate(p.val.ok = p.val < p.val.threshold) %>%
      filter(p.val.ok == TRUE) %>%
```

```
      group_by_at(.vars = c(p, "p.val.ok")) %>%
      count() %>%
      ungroup() %>%
      mutate(parameter = p,
             success.pct = round(100 * n / 16, 0)) %>%
      rename_at(.vars = p, funs(paste0("value"))) %>%
      mutate(value = as.character(value)) %>%
      select(parameter, value, n, success.pct)


    res.table
  })) %>%
    kable(caption = paste("Succes percentage voor p-waarde <= ", p.val.threshold))
}
```

Voor een "aanvaardbare" p-waarde van 0.05 :

```
success.rate(0.05)
```

Table 12: Succes percentage voor p-waarde <= 0.05

parameter	value	n	success.pct
dist	norm	5	31
dist	lnorm	6	38
sd.ratio	1	9	56
sd.ratio	5	2	12
n.total	20	5	31
n.total	200	6	38
sample.ratio	0.333333333333333	5	31
sample.ratio	1	6	38
test.type	pooled	5	31
test.type	Welch	6	38

Voor een "aanvaardbare" p-waarde van 0.055 :

```
success.rate(0.055)
```

Table 13: Succes percentage voor p-waarde <= 0.055

parameter	value	n	success.pct
dist	norm	12	75
dist	lnorm	6	38
sd.ratio	1	14	88
sd.ratio	5	4	25
n.total	20	9	56
n.total	200	9	56
sample.ratio	0.333333333333333	8	50
sample.ratio	1	10	62
test.type	pooled	8	50
test.type	Welch	10	62

Voor een "aanvaardbare" p-waarde van 0.06 :

```
success.rate(0.06)
```

Table 14: Succes percentage voor p-waarde <= 0.06

parameter	value	n	success.pct
dist	norm	13	81
dist	lnorm	7	44
sd.ratio	1	15	94
sd.ratio	5	5	31
n.total	20	9	56
n.total	200	11	69
sample.ratio	0.333333333333333	9	56
sample.ratio	1	11	69
test.type	pooled	9	56
test.type	Welch	11	69

De invloed van het type test, voor de verschillende combinaties van de andere parameters.

```
result %>%
  spread(test.type, p.val) %>%
  mutate(winner = if_else(pooled < Welch, "pooled","Welch"),
         diff = round(100 * (Welch - pooled) / if_else(pooled < Welch, Welch, pooled),1)) %>%
  kable(caption = "Vergelijking pooled-variance en Welch voor verschillende combinaties")
```

Table 15: Vergelijking pooled-variance en Welch voor verschillende combinaties

dist	sd.ratio	n.total	sample.ratio	pooled	Welch	winner	diff
norm	1	20	0.3333333	0.0506	0.0538	pooled	5.9
norm	1	20	1.0000000	0.0513	0.0455	Welch	-11.3
norm	1	200	0.3333333	0.0497	0.0473	Welch	-4.8
norm	1	200	1.0000000	0.0517	0.0528	pooled	2.1
norm	5	20	0.3333333	0.2731	0.0494	Welch	-81.9
norm	5	20	1.0000000	0.0634	0.0509	Welch	-19.7
norm	5	200	0.3333333	0.2314	0.0502	Welch	-78.3
norm	5	200	1.0000000	0.0552	0.0448	Welch	-18.8
lnorm	1	20	0.3333333	0.0476	0.0649	pooled	26.7
lnorm	1	20	1.0000000	0.0375	0.0305	Welch	-18.7
lnorm	1	200	0.3333333	0.0421	0.0599	pooled	29.7
lnorm	1	200	1.0000000	0.0459	0.0462	pooled	0.6
lnorm	5	20	0.3333333	0.3200	0.1395	Welch	-56.4
lnorm	5	20	1.0000000	0.1235	0.1210	Welch	-2.0
lnorm	5	200	0.3333333	0.2590	0.0784	Welch	-69.7
lnorm	5	200	1.0000000	0.0711	0.0677	Welch	-4.8

Conclusie

De Monte-Carlo simulatie is efficiënter

- voor een normale distributie dan voor een lognormale
- wanneer de standaardafwijkingen van beide populaties gelijk zijn

- wanneer de samples even groot zijn, en voldoende groot
- voor gelijke standaardafwijkingen tov. verschillende standaardafwijkingen
- een Welch t-test geeft betere resultaten dan een pooled, zeker wanneer de standaardafwijkingen verschillend zijn