Voorspelling van de verspreiding van het COVID-19 virus in Nederland met het Ensemble Kalman Filter

door

Ireen Fick

ter verkrijging van de graad van Bachelor of Science

aan de Technische Universiteit Delft,

Datum:	30 juli 2020	
Studentnummer:	4433963	
BEP-commissie:	Prof. dr. ir. A. Heemink,	TU Delft, supervisor
	Dr. ir. L. J. J. van Iersel,	TU Delft

TU Delft

Voorwoord

Het verslag dat voor u ligt, is geschreven ter verkrijging van de graad Bachelor of Science.

De afgelopen tijd heb ik onderzoek gedaan naar de verspreiding van het COVID-19 virus in Nederland. Ik heb hiervoor het SIR model en het Ensemble Kalman Filter bestudeerd en toegepast op bestaande data.

Mijn motivatie voor dit onderwerp komt voort uit twee aspecten.
Het eerste is dat het mij erg interesseert wiskunde te kunnen toepassen op de realiteit. Op papier kan je ontzettend veel met wiskunde, maar mijn passie ligt bij het begrijpelijk maken ervan voor ook niet-wiskundigen.
Het tweede aspect, wat goed aansluit bij het eerste, is dat het Corona virus het meest besproken onderwerp van 2020 is. De wereld is voortdurend bezig met het virus en de situatie verandert continu, waardoor het zeer actueel nieuws blijft. Belangrijke onderdelen van het bestrijden van het virus, zijn een goede voorspelling en analyse van het gedrag van het virus. Hier ben ik dan ook de afgelopen maanden druk mee geweest.

Graag wil ik A. Heemink hartelijk bedanken voor de kans dit onderzoek te doen, het continu begeleiden en motiveren tijdens het proces van mijn Bachelor Eind Project en zijn feedback op dit verslag. Hiernaast uit ik ook mijn dank aan L. van Iersel als lid van mijn Bachelor Eind Project beoordelingscommissie.

Inhoudsopgave

1 Introductie

Al ruim een half jaar houdt het de hele wereld in zijn greep: het Corona virus. Waar we dachten dat het allemaal begon als een onschuldige griep, werd al snel duidelijk dat er meer aan de hand was. Wuhan kampte al een tijdje met dit onbekende virus en op 23 januari 2020 werd het zelfs zo serieus dat de Chineese overheid een lockdown afkondigde voor Wuhan en andere steden in de provincie Hubei. Winkels, openbare plekken en -vervoersmiddelen werden gesloten, kinderen zouden vanuit huis geschoold gaan worden en waar mogelijk werd opgedragen thuis te werken. Zoiets hadden we nog nooit meegemaakt.
Toch leek dit voor menig Nederlander één grote ver-van-mijn-bed-show, tot het nieuws ons bereikte dat ook in Italië besmettingsgevallen van het COVID-19 virus werden geconstateerd. Langzaam maar zeker kroop het virus verder naar ons kikkerlandje en op 23 maart 2020 was ook in Nederland het virus zo serieus aanwezig dat een lockdown werd aangekondigd.

Nu, zo'n 4 maanden verder, weten we niet meer beter. Iedereen is in bezit van één of meerdere mondkapjes, op het verplichte gebruik van een winkelkar in de supermarkten wordt niet meer gemopperd en *anderhalve-meter-samenleving* maakt grote kans om de Woord van het Jaar-verkiezing 2020 te winnen.
Toch blijft iedereen het spannend vinden hoe deze crisis zal aflopen. De voortgang van en nieuwe informatie over het virus zijn al tijden *hot topic* op social media en bijna dagelijks verschijnen er nieuwe wiskundige modellen met aanvullende informatie in de krant. Deze modellen geven ontzetten veel informatie over onder andere de verspreiding en het verloop van het Corona virus.

In dit verslag bekijken we met statistische methoden hoe we ziektes kunnen voorspellen. Daarnaast gebruiken we een methode om voorspellingen met data te combineren. Dankzij het RIVM zijn er veel gegevens[1] over de huidige Corona situatie beschikbaar. We gebruiken deze data om het verloop van het virus te modelleren, rekening houdend met een modelfout en een meetfout. Hiermee brengen we in kaart hoe het virus zich de afgelopen tijd heeft gedragen en wat we in de toekomst kunnen verwachten van een vergelijkbare ziekte.

Alle berekeningen en plots die zijn gebruikt, zijn gemaakt in *Rstudio*[2], een statistisch computerprogramma.

[1] https://www.rivm.nl/coronavirus-covid-19/actueel
[2] https://rstudio.com/

2 Het SIR model

In deze sectie wordt beschreven hoe we het COVID-19 virus kunnen bestuderen en voorspellen aan de hand van het SIR model[3], het model voor verspreiding van ziekten. We kijken naar het basis model en de mogelijkheden om deze uit te breiden. Vervolgens kijken we naar het SIR model gemaakt aan de hand van de Euler methode.

2.1 Het SIR model

In tabel 2.1 zien we het aantal doden in Nederland van de afgelopen paar weken en het deel hiervan veroorzaakt door het Corona virus.[4]

Week in 2020	Overledenen totaal	Overledenen door Corona	% Overledenen door Corona
24	2.686	46	1,71 %
25	2.684	31	1,15 %
26	2.653	15	0,57 %
27	2.623	22	0,84 %
28	2.581	10	0,39 %

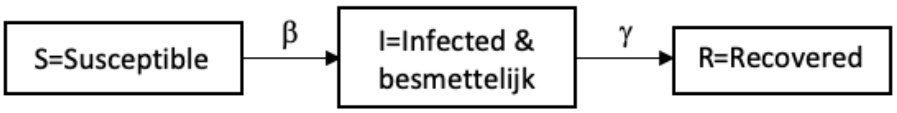
Figuur 2.1: Aantal overledenen per week Nederland

Deze aantallen staan waarschijnlijk in verhouding met het aantal nieuwe ziektegevallen een bepaalde tijd eerder. Dit is belangrijke informatie om te kunnen bepalen hoeveel geïnfecteerden we op bepaalde momenten hadden. Op deze manier kan achteraf een verloop geschetst worden van het virus.

Het SIR model wordt gebruikt om dit soort ziektes te modelleren, om in de toekomst het verloop van dit virus en vergelijkbare virussen te kunnen voorspellen.
Er worden 3 compartimenten onderscheiden: S: vatbaar/susceptible, I: besmettelijk / infectious en R: genezen/recovered. Onder de genezen individuen vallen ook de mensen die gestorven zijn aan de ziekte, immers kunnen ook zij de ziekte niet meer overdragen. Tussen de verschillende compartimenten zitten overgangen, zoals weergegeven in figuur 2.2.

[3] https://www.maa.org/press/periodicals/loci/joma/the-sir-model-for-spread-of-disease-the-differential-equation-model

[4] https://allecijfers.nl/nieuws/statistieken-over-het-corona-virus-en-covid19/#Corona_-kerncijfers



Figuur 2.2: De compartimenten van het SIR model

We gebruiken deze letters ook om twee verschillende tijdsafhankelijke verzamelingen aan te duiden.
De eerste verzameling telt het aantal personen in elke groep:

$S(t)$: het aantal vatbare individuen
$I(t)$: het aantal geïnfecteerde & besmettelijke individuen
$R(t)$: het aantal genezen individuen

De tweede verzameling representeert het deel van de bevolking in elk van de groepen. Dus als N de totale populatie is, hebben we:

$s(t) = S(t)/N$: het vatbare deel van de bevolking
$i(t) = I(t)/N$: het geïnfecteerde & besmettelijke deel van de bevolking (hierna te noemen: geïnfecteerde óf besmettelijke deel van de bevolking)
$r(t) = R(t)/N$: het genezen deel van de bevolking

2.1.1 Aannames voor het SIR model

Om het model te kunnen gebruiken, moeten we een aantal aannames doen.

1. We nemen aan dat $s(t) + i(t) + r(t) = 1$, aangezien deze groepen samen de hele bevolking moeten weergeven.

2. We negeren in het basismodel de nieuw-geborenen, de overledenen door andere oorzaken dan de ziekte en emigranten en immigranten.

3. De groep vatbare individuen wordt niet groter, we negeren geboorte en immigratie in het basis model. Deze groep kan wel kleiner worden, doordat mensen besmet raken. Het aantal vatbare personen hangt af van het aantal mensen dat al vatbaar was, het aantal mensen dat besmet is en de hoeveelheid contact tussen vatbare en besmettelijke individuen.
   Neem aan dat elk besmettelijk individu een vast aantal contacten heeft per dag dat kans heeft besmet te raken. Hier komt de parameter β in beeld, die de overgangssnelheid van de groep vatbare mensen naar de geïnfecteerden aangeeft, oftewel de kans op overdracht van de ziekte tussen mensen in de groep S en mensen in de groep I. We nemen aan dat het deel van deze contacten dat met vatbare individuen in contact staat, $s(t)$ is. Dus gemiddeld veroorzaakt elk besmet persoon $\beta * s(t)$ nieuwe besmettingen.

4. We kunnen ervan uitgaan dat er elke dag een vast deel γ van de bevolking geneest van de ziekte. Als een persoon bijvoorbeeld 10 dagen besmettelijk is van het virus, geneest als het ware elke dag $1/10$ van de besmette personen. Deze personen behoren hierna tot de groep genezen individuen. γ

geeft dus ook wel de overgangssnelheid van het I naar het R compartiment aan.

Nu we deze aannames hebben gedaan, komen we veel te weten over de afgeleiden van onze variabelen.

2.1.2 De vergelijking voor de vatbare bevolking

We krijgen de volgende vergelijkingen voor de afgeleiden van het aantal vatbare individuen en het deel van de bevolking dat vatbaar is:

$$\frac{ds}{dt} = -\beta * s(t)i(t)$$

Het $-$-teken volgt uit het feit dat het aantal vatbare personen afneemt als er meer mensen besmet raken.
Als iedereen in de populatie vatbaar is, dan zijn er per dag $\beta * I(t)$ nieuwe geïnfecteerden. Als echter een deel van de bevolking niet meer vatbaar is, wordt effectief het aantal contacten dat tot een besmetting kan leiden minder. Immers een deel van de contacten kan niet meer tot een besmetting leiden omdat mensen al besmet zijn of inmiddels immuun zijn. Dus per dag zijn er dan $\beta * S(t)/N = \beta * s(t)$ nieuwe geïnfecteerden. Daaruit volgt dat als $s(t)$ heel klein wordt, de besmettelijkheid van het virus afneemt. Dit noemen we groepsimmuniteit. Bij een β van 2 moet dus meer dan 50% van de bevolking niet meer vatbaar zijn voordat $\beta * s(t)$ onder de 1 komt.

2.1.3 De vergelijking voor de genezen bevolking

$$\frac{dr}{dt} = \gamma * i(t).$$

De coëfficiënt γ geeft aan welk deel van de geïnfecteerde bevolking i(t) per dag immuun wordt en gaat behoren tot r(t). Deze coëfficiënt hangt van de ziekte af, waarbij vooral van belang is hoe lang iemand besmettelijk is voor zijn omgeving.

2.1.4 De vergelijking voor de besmette bevolking

Deze vergelijking maken we op uit de afgeleide van $s(t) + i(t) + r(t) = 1$:

$$\frac{ds}{dt} + \frac{di}{dt} + \frac{dr}{dt} = 0,$$

zodat

$$\frac{di}{dt} = \beta * s(t)i(t) - \gamma * i(t).$$

2.1.5 De parameters

Allereerst moeten we de totale bevolking N in Nederland weten. We houden hierbij de schatting van het CBS aan op 5 juni 2020: 17.425.000 (afgerond)[5].

[5] https://www.cbs.nl/nl-nl/visualisaties/bevolkingsteller

We weten nu nog weinig over de exacte waarden van γ en β, maar deze kunnen we schatten en later aanpassen aan de meetgegevens. Wel weten we een aantal andere dingen.

1. γ hangt af van de duur van de ziekte. Mensen met weinig klachten hebben hier vaak maar een paar dagen last van. Voor ernstigere gevallen is dit aantal dagen hoger [6]. We schatten dat gemiddeld gezien, men 10 dagen klachten van het COVID-19 virus heeft. Hierin is meegenomen dat sommige mensen heel weinig tot geen klachten hebben, terwijl anderen 14 dagen in quarantaine moeten. We nemen dus aan dat $\gamma = 1/10$.

2. De waarde van β hebben we zelf in de hand, door onze contacten met de buitenwereld zo veel mogelijk te beperken. Voordat het Corona virus in Nederland uitbrak was de waarde van β rond de 2. Na/gedurende de quarantaine periode zal dit waarschijnlijk afgenomen zijn tot iets onder de 1, zeg 0,9.

Benadrukt moet worden dat dit schattingen zijn.

2.1.6 Analyse van de vergelijkingen

Onze SIR vergelijkingen worden dus als volgt:

$$\frac{ds}{dt} = -\beta * s(t)i(t)$$

$$\frac{di}{dt} = \beta * s(t)i(t) - \gamma * i(t)$$

$$\frac{dr}{dt} = \gamma * i(t).$$

en we gebruiken hiervoor de algemene notatie

$$f(\vec{x_t}, t) = \begin{bmatrix} -\beta * s(t)i(t) \\ \beta * s(t)i(t) - \gamma * i(t) \\ \gamma * i(t) \end{bmatrix}$$

Laten we kijken naar de stabiliteit[7] van het systeem. Hiervoor bepalen we de evenwichtspunten door de differentiaalvergelijkingen gelijk te stellen aan 0. We zien dat $i = 0$ en $0 \leq s \leq N$, met N de constante bevolking. Bekijk de eerder genoemde vector

$$f(\vec{x_t}, t) = \begin{bmatrix} -\beta * s(t)i(t) \\ \beta * s(t)i(t) - \gamma * i(t) \\ \gamma * i(t) \end{bmatrix},$$

y is dan een evenwichtspunt van $f(\vec{x_t}, t)$ als $f(\vec{y}, t) = 0$. Vullen we dit evenwichtspunt in in de Jacobi-matrix[8] van dit stelsel:

[6] https://www.zorgvoorbeter.nl/nieuws/corona-veelgestelde-vragen

[7] http://homepage.tudelft.nl/11r49/onderw0102/diffvglwb/college/week41.pdf

[8] https://www.imedpub.com/articles/study-of-simple-sir-epidemic-model.pdf

$$J(s,i,r) = \begin{bmatrix} \frac{\partial}{\partial s}\frac{ds}{dt} & \frac{\partial}{\partial i}\frac{ds}{dt} & \frac{\partial}{\partial r}\frac{ds}{dt} \\ \frac{\partial}{\partial s}\frac{di}{dt} & \frac{\partial}{\partial i}\frac{di}{dt} & \frac{\partial}{\partial r}\frac{di}{dt} \\ \frac{\partial}{\partial s}\frac{dr}{dt} & \frac{\partial}{\partial i}\frac{dr}{dt} & \frac{\partial}{\partial r}\frac{dr}{dt} \end{bmatrix}$$
$$= \begin{bmatrix} -\beta i & -\beta s & 0 \\ \beta i & \beta s - \gamma & 0 \\ 0 & \gamma & 0 \end{bmatrix},$$

krijgen we:

$$J(s_0,0,r) = \begin{bmatrix} 0 & -\beta s & 0 \\ 0 & \beta s - \gamma & 0 \\ 0 & \gamma & 0 \end{bmatrix}.$$

De karakteristieke vergelijking wordt nu:

$$-\lambda(\beta s - \gamma - \lambda)(-\lambda),$$

waaruit we de eigenwaarden krijgen:

$$-\lambda(\beta s - \gamma - \lambda)(-\lambda) = 0$$

$$\lambda^2(\beta s - \gamma - \lambda) = 0$$

$$\lambda_{1,2} = 0, \lambda_3 = \beta s - \gamma$$

λ_3 is positief als $\beta s > \gamma$ en negatief als $\beta s < \gamma$. Als alle eigenwaarden een negatief reëel deel hebben is het systeem stabiel, als dat niet zo is, is het systeem instabiel.

Om te bepalen met welke situatie we te maken hebben, kijken we eerst naar de vergelijking voor i, afgeleid naar s, waaruit de we de kritieke punten van de functie i kunnen bepalen:

$$\frac{di}{ds} = \frac{\beta * s(t)i(t) - \gamma * i(t)}{-\beta * s(t)i(t)}$$
$$= \frac{\beta * s(t)i(t)}{-\beta * s(t)i(t)} + \frac{-\gamma * i(t)}{-\beta * s(t)i(t)}$$
$$= -1 + \frac{\gamma}{\beta * s(t)}$$

We bepalen de kritieke punten door $\frac{di}{ds} = 0$ te stellen. We vinden dan $s = \frac{\gamma}{\beta}$ en gebruiken nu de tweede afgeleide om te bepalen of dit punt een minimum, maximum of zadelpunt is:

$$\frac{d^2i}{ds^2} = -\frac{\gamma}{\beta * s^2}$$

γ, β en S zijn allemaal groter of gelijk aan 0, dus volgt dat $\frac{d^2i}{ds^2} \leq 0$, wat betekent dat dit kritiek punt een maximum is. Hieruit weten we dus dat als er een epidemie uitbreekt, $\frac{di}{dt} > 0$ als $s > \frac{\gamma}{\beta}$. Dit betekent dat $\beta s > \gamma$ en dat λ_3 dus een positieve eigenwaarde is, wat het systeem instabiel maakt.

2.2 Het reproductie getal

2.2.1 Het contact getal

Om een goede afspiegeling te maken van een epidemie, moet het duidelijk zijn in welke mate mensen contact met elkaar hebben. Het virus kan zich gemakkelijk verspreiden als de bevolking constant met elkaar in contact is, bij bijvoorbeeld werkplekken, scholen & universiteiten, openbare ruimtes en uitgaansgelegenheden. Wanneer er een epidemie uitbreekt, neemt de overheid maatregelen om het contact getal te verkleinen. Het contact getal blijft dus niet constant, maar is tijdsafhankelijk en wordt bepaald door:

$$c(t) = (c_0 - c_b)e^{-r_1 t} + c_b$$

waarbij c_0 het contact getal op het eerste moment is, c_b het minimum contact getal onder de huidige maatregelen en r_1 de mate waarin het contact getal exponentieel afneemt.[9] Er geldt $c_b < c_0$, om aan te nemen dat de mate van contact afneemt als de maatregelen worden nagestreefd.

2.2.2 R_0 en R_{eff}

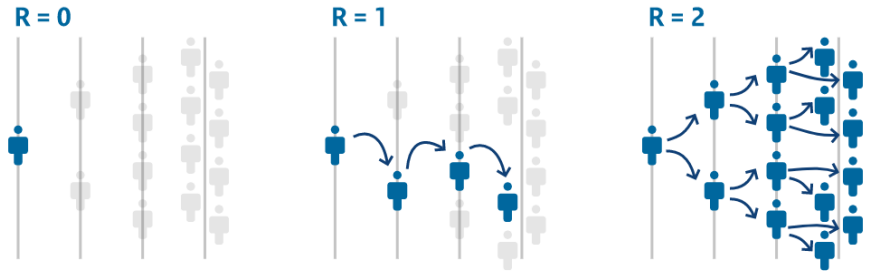
Een belangrijk getal dat afhankelijk is van het contact getal is het reproductie getal R, niet te verwarren met de R van de recovered uit het SIR model. Dit zogenaamde reproductie getal R is een van de belangrijkste getallen in elke epidemie en geeft het aantal personen aan dat een besmet individu gemiddeld infecteert.[10]. Hierin onderscheiden we twee groepen, R_{eff} en R_0. R_{eff} is het effectief reproductie getal, R_0 het basaal reproductie getal. Deze laatste gaat uit van een bevolking die nog niet bekend is met de ziekte en waarin nog niemand immuun is.

In afbeelding 2.3 wordt grafisch weergegeven hoe de ziekte zich verspreidt bij verschillende waarden van R.[11] In een normale samenleving is R gewoonlijk rond de 2. Dat betekent dat als een persoon besmet is, deze persoon gemiddeld weer twee andere personen besmet. Deze twee personen besmetten weer allebei twee mensen en zo ontstaat er een exponentiële groei van infecties. Is R gelijk aan 1, dan komen er geen nieuwe besmettingen bij, echter nemen ze ook niet af. Om voor een daling van het aantal besmettingen te zorgen, moet de R dus onder 1 komen te liggen. Op deze manier besmet een geïnfecteerd persoon gemiddeld minder dan 1 extra persoon en zal de verspreiding van het virus dus afnemen. Dit is dus ook waar het Outbreak Management Team (OMT) op mikt sinds de huidige corona uitbraak. Als R lange tijd onder 1 blijft, dooft de ziekte langzaam uit en wordt het systeem *stabiel*.

[9] An updated estimation of the risk of transmission of the novel coronavirus (2019-nCov)

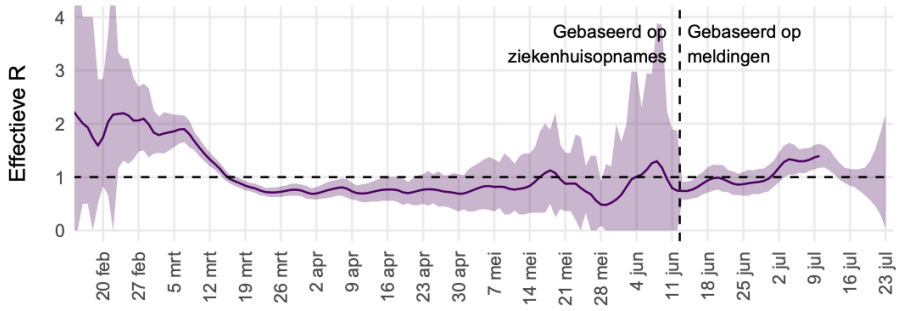
[10] https://decorrespondent.nl/11149/dit-is-het-belangrijkste-getal-van-deze-epidemie-hoe-wordt-het-berekend/162248208386-a0a5fdf8

[11] https://coronadashboard.rijksoverheid.nl/



Figuur 2.3: Ziekteverspreiding bij verschillende R waarden

In grafiek 2.4 zien we de effective waarde van R in Nederland van 17 februari tot 23 juli 2020.[12] Zoals te zien is R sinds half maart lange tijd onder 1 geweest. Dat is ook logisch, omdat er toen strenge maatregelen werden ingesteld en iedereen in Nederland voornamelijk thuis bleef. Zodra de maatregelen een beetje werden versoepeld, werd men lakser en we zien dus ook dat er rond 18 mei en 8 juni pieken zijn geweest waarbij R weer even boven 1 was.



Figuur 2.4: Effectieve R waarde in Nederland

De waarde van R wordt beïnvloed door drie factoren:

1. De kans dat iemand besmet wordt bij contact met een geïnfecteerd persoon $(\beta * s(t))$
2. De mate waarin mensen contact hebben met elkaar per tijdseenheid $(c(t))$
3. Hoe lang de ziekte besmettelijk is (γ)

Vermenigvuldig deze drie met elkaar en je krijgt het reproductie getal:

$$R_0 = \frac{\beta * c(0)}{\gamma} * s(0)$$

$$R_t = \frac{\beta * c(t)}{\gamma} * s(t)$$

[12] https://coronadashboard.rijksoverheid.nl/

We zagen een dergelijke vergelijking al eerder, namelijk bij de eigenwaarden van het systeem. Hiervoor gold

$$s > \frac{\gamma}{\beta}$$

$$\beta s > \gamma$$

$$\frac{\beta s}{\gamma} > 1$$

en dus geldt

$$\frac{R_0}{c(0)} > 1.$$

R_0 ontstaat dus als instabiele eigenwaarde van het systeem.

Voor een deel hebben we de voortgang van R zelf in de hand. Zo hebben we een bepaalde invloed op bovengenoemde factoren.
De kans dat iemand besmet kan worden door een geïnfecteerd persoon, kunnen we verkleinen door afstand van elkaar te houden, een beperkt aantal mensen toe te laten in winkels en openbare plekken, maar ook door bijvoorbeeld goed je handen te wassen en te niesen in je elleboog.
Ook de mate waarin mensen contact hebben met elkaar is goed te controleren, door thuis te werken, scholen en sociale verzamelplekken te sluiten en feesten & festivals af te gelasten.
Als laatste kunnen we de duur van de besmettelijkheid verkorten door snel op de hoogte te zijn van ziektegevallen en hierbij meteen actie te ondernemen.

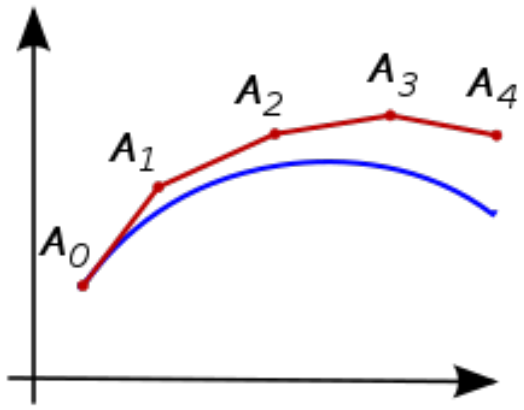
Echter moet vermeld worden dat R een gemiddelde is. Het kan zo zijn dat op bepaalde plekken geen extra personen besmet worden door de huidige geïnfecteerde individuen, terwijl op andere plekken een enkele zieke een groot aantal extra besmettingen veroorzaakt. Dat kan bijvoorbeeld veroorzaakt worden door grotere vatbaarheid als gevolg van leeftijd, zoals in een bejaardentehuis. In deze gebieden kunnen er toch ineens grote uitbraken ontstaan, terwijl op andere plekken het virus zich terug begint te trekken.

2.3 De Euler methode

De methode van Euler is de eenvoudigste methode om een numerieke oplossing te berekenen van een differentiaalvergelijking met beginvoorwaarden. De methode werd bedacht door Leonhard Euler en gepubliceerd in 1768 in zijn boek *Institutiones Calculi Integralis*. We willen de vorm van een onbekende curve bepalen, die start op een bepaald punt en voldoet aan de gegeven differentiaalvergelijking. De differentiaalvergelijking is als het ware een formule waarmee de helling van de raaklijn aan de curve op elk punt van de curve kan worden berekend, zodra de positie van dat punt is berekend.

We starten met een bekend uitgangspunt, A_0, om de nog onbekende curve te bepalen. Vervolgens berekenen we vanuit de differentiaalvergelijking de helling naar de curve en dus de raaklijn. Als we nu een volgend punt, A_1, op de raaklijn kiezen, dichtbij het vorige zodat de helling van de curve niet te veel verandert, kunnen we op dezelfde manier als voor A_0 de raaklijn aan dit punt bepalen. Door dit voor verschillende punten te doen, kunnen we de curve bepalen. Zie

figuur 2.5.[13] Hoe kleiner we de stappen maken, hoe meer de curve overeenkomt met de oorspronkelijke curve.



Figuur 2.5: Illustratie van de Euler methode

Deze methode is van de eerste orde, dit betekent dat de lokale fout, de fout per stap, evenredig is met het kwadraat van de stapgrootte en de algemene fout, de fout op een bepaald moment, evenredig is met de stapgrootte.

De methode van Euler houdt dus in dat als we een hellingsvergelijking hebben, waarmee we $\frac{dy}{dt}$ op elk punt (t, y) kunnen bepalen, we een rij $y-$waarden kunnen creëren, waarbij we starten bij een gegeven y_0 en steeds de helling berekenen, door

$$y_{n+1} = y_n + helling_n * \Delta t.$$

In het SIR model willen we de variabelen s, i en r gebruiken, dus worden de Euler vergelijkingen:

$$s_{n+1} = s_n + helling_n * \Delta t$$
$$i_{n+1} = i_n + helling_n * \Delta t$$
$$r_{n+1} = r_n + helling_n * \Delta t.$$

Wegens de SIR vergelijkingen:

$$\frac{ds}{dt} = -\beta * s(t)i(t)$$
$$\frac{di}{dt} = \beta * s(t)i(t) - \gamma * i(t)$$
$$\frac{dr}{dt} = \gamma * i(t)$$

worden de uiteindelijke Euler vergelijkingen:

$$s_{n+1} = s_n - \beta * s_n * i_n * \Delta t$$

[13] https://en.wikipedia.org/wiki/Euler_method

$$i_{n+1} = i_n + (\beta * s_n * i_n - \gamma * i_n) * \Delta t$$
$$r_{n+1} = r_n + \gamma * i_n * \Delta t.$$

Ook hier beginnen we met geschatte waarden. Voor β en γ hadden we dit al gedaan. Δt kunnen we variëren, we zullen beginnen met een waarde klein genoeg, zoals bijvoorbeeld 0,1.

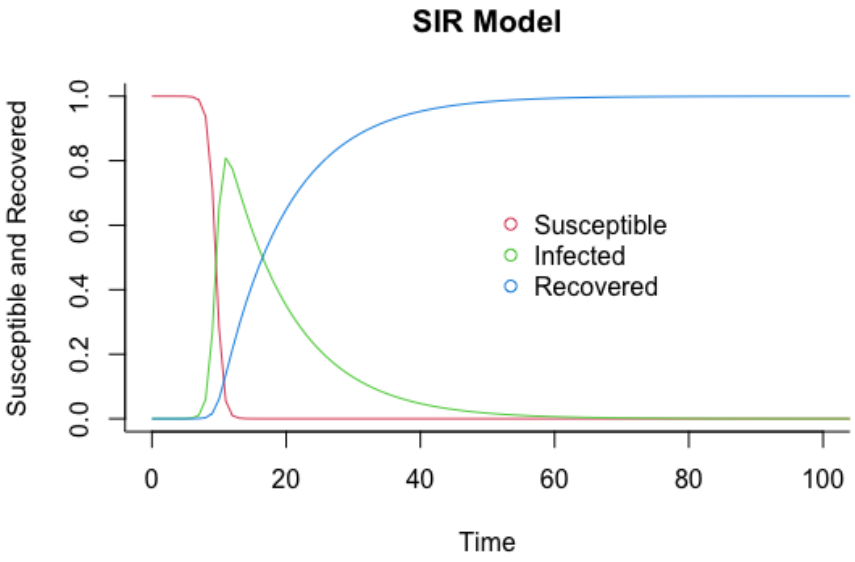
Ook moeten we de startwaarden $s(0), i(0), r(0)$ weten. Om te beginnen gaan we ervan uit dat het virus Nederland binnen kwam doordat er een enkel persoon besmet was. Om hier de fractie van de bevolking van te maken, delen we door de totale populatie. Nederland heeft 17.425.000 inwoners (5 juni 2020), wat een startwaarde voor i van $5,7e^{-8}$ oplevert. De grootte van het vatbare deel van de bevolking bij de start, is iedereen die niet tot $i(0)$ behoort. Trek van de totale bevolking het besmette deel van de bevolking af en we krijgen een groep van $1 - 5,7e^{-8}$ als startwaarde voor s. $r(0)$ heeft een startwaarde van 0, aangezien niemand nog immuun was, toen het virus ontstond.
Met deze startwaarden kunnen we dus alle volgende s, i en r bepalen.

De Euler methode is een van de simpelste methoden voor het oplossen van differentiaalvergelijkingen met beginvoorwaarden. Toch is de methode voor ons zeer geschikt om te gebruiken omdat we hier te maken hebben met een klein systeem, waarbij de snelheid van het systeem niet van het grootste belang is.

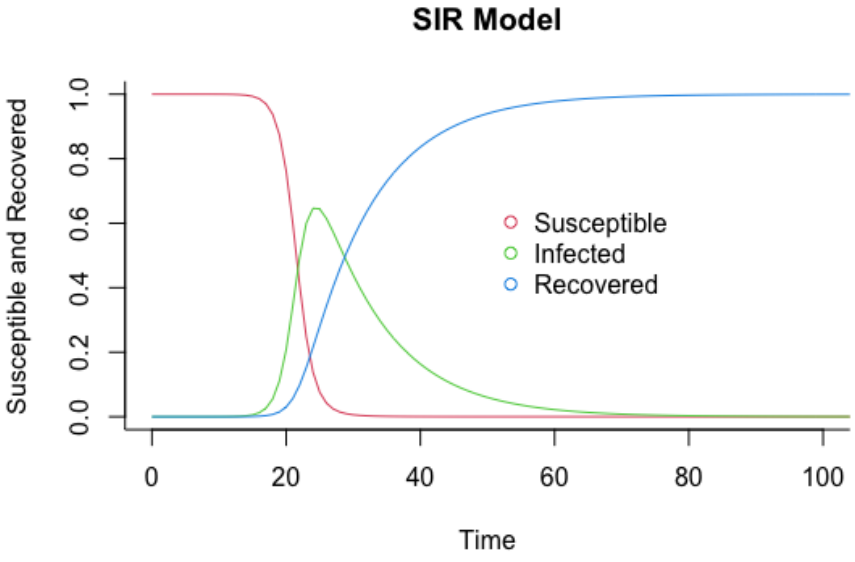
2.4 Modelleren van het SIR model

Nu we de differentiaalvergelijkingen hebben en de parameters hebben geschat, kunnen we een eerste plot maken van hoe het SIR model eruit komt te zien. We doen dit voor twee gevallen:

1. $\beta = 2$: hoe de situatie zou zijn zonder maatregelen te nemen tegen het verspreiden van het virus
2. $\beta = 0,9$: de geschatte waarde van β na een thuisquarantaineperiode.



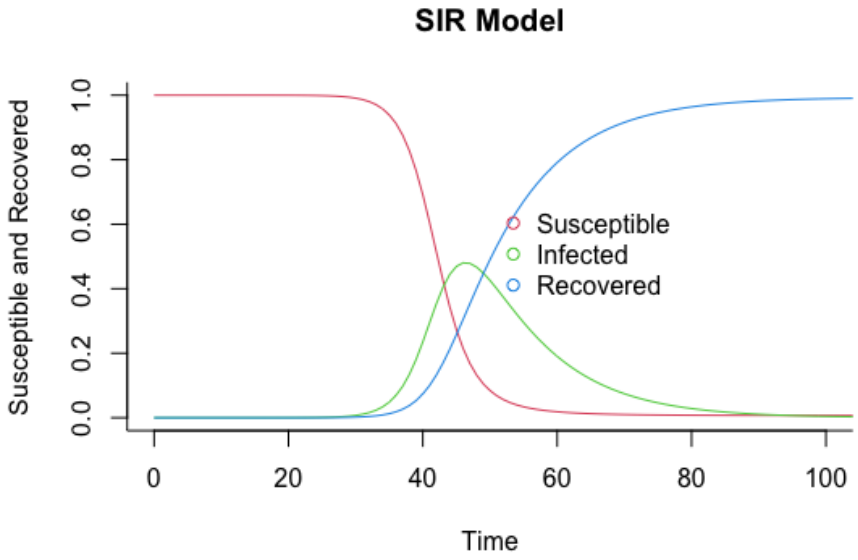
Figuur 2.6: Het SIR model met $\beta = 2$



Figuur 2.7: Het SIR model met $\beta = 0{,}9$

Een verschil is te zien in de hoogte en de plaats van de groene piek. Als β = 2, zien we dat zeer snel een heel groot deel van de bevolking besmet raakt. Dit zou betekenen dat 80% van de bevolking na zo'n 15 dagen besmet is met het virus, mt het gevolg dat ziekenhuizen deze piek niet aankunnen, waardoor niet

elk besmet persoon behandeld kan worden. Dit verhoogt de kans op snelle verspreiding van de ziekte.
Als $\beta = 0,9$ is deze piek iets later en minder hoog.
Opvallend is dat de voorspellingen voor het geïnfecteerde deel van de bevolking in beide gevallen niet veel van elkaar afwijken. Wellicht dat de schatting voor een van de parameters niet goed is. We bekijken daarom ook het model met $\beta = 0,5$:



Figuur 2.8: Het SIR model met $\beta = 0,5$

Hier zien we al veel meer verschil met $\beta = 2$ als uitgangspunt. De waarde van β is dus heel belangrijk voor het voorspellen van de epidemie, aangezien deze samenhangt met het reproductiegetal R en we willen deze daarom zo goed mogelijk benaderen. We corrigeren onze geschatte keuze voor β eventueel, nadat we verdere analyse hebben gedaan.

3 Het Ensemble Kalman Filter

In de volgende sectie wordt een uitleg gegeven over het ensemble Kalman filter. Daarnaast zien we hoe dit filter toegepast kan worden om een optimale voorspelling te maken voor de verspreiding van het Corona virus. Hiervoor gebruiken we zowel de geschatte meetgegevens uit het SIR model, als de daadwerkelijke data.

3.1 Introductie

Het Ensemble Kalman Filter is een geschatte filtermethode, geintroduceerd door Evensen in de geofysische literatuur[14]. Het filter is een Monte-Carlo implementatie van het Bayesiaanse update probleem, dat als volgt in zijn werk gaat. Als we een kansdichtheidsfunctie hebben van de toestand van het gemodelleerde systeem en gegeven de data likelihood, wordt de stelling van Bayes gebruikt om de kansdichtheidsfunctie te verkrijgen, nádat rekening is gehouden met de data likelihood. Ook wordt het model ontwikkeld op tijd, waarbij steeds nieuwe meetgegevens worden opgenomen, waarna dit gecombineerd wordt met de Bayesiaanse update.
We willen het model en de metingen combineren, omdat het model met geschatte waarden werkt, die we dus niet zeker weten, maar we aan de hand van de metingen alleen achteraf conclusies kunnen trekken. Door beide te combineren kunnen we een optimale voorspelling maken.

Het ensemble Kalman filter is in het leven geroepen toen een obstakel zich voordeed bij gebruik van het oorspronkelijke Kalman filter. Dit laatste neemt namelijk aan dat alle kansdichtheidsfuncties Gaussisch[15] zijn. Daarnaast biedt het Kalman filter algebraïsche formules voor de verandering van het gemiddelde en covariantie door de Bayesiaanse update, evenals een formule voor het op tijd bevorderen van de covariantiematrix. Echter is het gebruik van de covariantiematrix niet mogelijk voor hoogdimensionale systemen en zo werd het ensemble Kalman filter (EnKF) ontwikkeld.

EnKF's gebruiken een willekeurige steekproef (een ensemble) om zo de verdeling van het systeem te vertegenwoordigen en vervangen de covariantiematrix door de steekproefcovariantie berekend uit het ensemble. Als er nieuwe waarnemingen beschikbaar worden, wordt het ensemble bijgewerkt. Het EnKF is heel succesvol in veel extreem hoge dimensionale, niet-lineaire en niet-Gaussische data-assimilatie-applicaties. Voor veel realistische, zeer complexe systemen is de EnKF in wezen de enige manier om gevolgtrekking te doen, terwijl alternatieve exacte gevolgtrekkingstechnieken alleen kunnen worden toegepast op sterk vereenvoudigde versies van het probleem.[16]

[14] Evensen, Geir (2009). Data Assimilation. The Ensemble Kalman Filter
[15] https://nl.wikipedia.org/wiki/Normale_verdeling
[16] https://www.math.umd.edu/ slud/RITF17/enkf-tutorial.pdf

3.2 Het Kalman Filter

Allereerst bekijken we het originele Kalman filter[17]. Laat $\vec{x}$ de vector $\begin{bmatrix} S \\ I \\ R \end{bmatrix}$ zijn, met Gaussische kansverdeling, waarvan het gemiddelde μ is en de covariantie P^f. De kansdichtheidsfunctie hiervan wordt dan gegeven door:

$$p(\vec{x}) \propto exp(-\frac{1}{2}(\vec{x}-\mu)^T(P^f)^{-1}(\vec{x}-\mu)).$$

Deze kansverdeling ontstaat door het model uit te voeren over de tijd. We noemen dit de *prior*. Omdat we ook over data beschikken, moet de prior worden bijgewerkt om deze nieuwe gegevens te verwerken. We geven de data weer door $\vec{d}$, ook met Gaussische kansverdeling, met gemiddelde $M\vec{x}$ en covariantie R. R geeft hier de schatting van de meetfout aan. M is hierbij een vector gevuld met nullen, behalve op de plek van de variabele waar we de meting voor doen. Meten we bijvoorbeeld I, het aantal geïnfecteerde personen, dan ziet M eruit als $\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$. We noemen dit de observatie-matrix. $M\vec{x}$ geeft aan wat de waarde werkelijk zou moeten zijn. We kunnen dan de kansdichtheid opstellen voor de data $\vec{d}$, op voorwaarde dat $\vec{x}$ geldt. Dit heet de data likelihood:

$$p(\vec{d}|\vec{x}) \propto exp(-\frac{1}{2}(\vec{d}-M\vec{x})^T R^{-1}(\vec{d}-M\vec{x}).$$

We kunnen nu met de regel van Bayes[18] $p(\vec{d}|\vec{x})$ en $p(\vec{x})$ combineren en de kansdichtheidsfunctie opstellen voor $\vec{x}$ afhankelijk van de gegevens van $\vec{d}$, de *posterior* genoemd:

$$p(\vec{x}|\vec{d}) \propto p(\vec{d}|\vec{x})p(\vec{x}).$$

We krijgen dan:

$$p(\vec{x}|\vec{d}) \propto exp(-\frac{1}{2}(\vec{x}|\vec{d}-\hat{\mu})^T(\hat{P^f})^{-1}(\vec{x}|\vec{d}-\hat{\mu})),$$

met gemiddelde $\hat{\mu}$ en covariantie $(\hat{P^f})$, gegeven door de Kalman update vergelijkingen:

$$\hat{\mu} = \mu + K(\vec{d} - M\mu)$$

$$\hat{P^f} = (I - KM)P^f.$$

Hierbij is K de Kalman gain, die we terug gaan zien in het Ensemble Kalman Filter, en wordt gegeven door:

$$K = P^f M^T[MP^f M^T + R]^{-1}.$$

De Kalman gain is het gewicht dat wordt gegeven aan de metingen en de schatting. Met een hoge K legt het filter meer gewicht op de meest recente metingen en moeten de voorspelde waarden dus meer worden aangepast dan met een lage K.

[17] Mandel, Jan (2009). A Brief Tutorial on the Ensemble Kalman Filter

[18] https://nl.wikipedia.org/wiki/Theorema_van_Bayes

3.3 Ensemble methode voor het SIR model

Zoals eerder genoemd is het Kalman filter niet bruikbaar voor extreem hoge dimensionale, niet-lineaire en niet-Gaussische systemen. Om die reden werd het Ensemble Kalman Filter bedacht. Dit borduurt voort op het originele Kalman filter, maar vervangt de covariantiematrix door de covariantiematrix berekend uit het ensemble. Dit is ook wat we gaan doen als we het EnKF toepassen op het SIR model.

We willen dus een groot aantal ensembles genereren en die vergelijken met ons model. Uiteindelijk gebruikt het filter zowel de metingen als het model en combineert deze tot een optimale voorspelling.

Om te beginnen willen we een algemene vorm om de notatie gemakkelijk te maken:

$$\vec{x_{t+\Delta t}} = \vec{x}_t + f(\vec{x_t}, t) * \Delta t.$$

Hierbij is $\vec{x}_t$ de vector $\begin{bmatrix} S_t \\ I_t \\ R_t \end{bmatrix}$ en $f(\vec{x_t}, t)$ zoals gedefinieerd in sectie 2.1.6.

We gaan een ensemble van mogelijke uitkomsten modelleren. We kunnen dit doen door de startwaarden van S, I en R of de parameters te variëren. We vervangen in ons model bijvoorbeeld de waarde van β door de normaalverdeling met gemiddelde 1 en variantie 0,5 en laten hieruit voor elk ensemble member een random trekking doen. Uit het ensemble dat gegenereerd wordt, kunnen we de beste voorspelling bepalen door het gemiddelde te bepalen. De spreiding van de plots is een maat voor de onzekerheid.

We krijgen hieruit een ensemble voorspellingen $(x_t^f)^i$. Hierbij staat f voor forecast. t is de tijdsindex en i het aantal ensembles dat we willen genereren, bijvoorbeeld 100. Merk op dat deze voorspelling zonder data gegenereerd is.

Er zijn twee grootheden van belang nadat we dit ensemble hebben gegeneerd. De eerste is het gemiddelde van het ensemble, deze kunnen we op de volgende manier bepalen:

$$Gemiddelde : \vec{x_t^f} = \frac{1}{N} \sum_i (\vec{x_t^f})^i.$$

Dit gemiddelde hebben we nodig om voor elke tijdstap de covariantie matrix te bepalen:

$$Covariantiematrix : P^f = \frac{1}{N-1} \sum_i \left(\left[(x_t^f)^i - \vec{x_t^f} \right] * \left[(x_t^f)^i - \vec{x_t^f} \right]^T \right).$$

Hierbij is N het aantal ensembles en P_t^f een positief definiete matrix, met dimensie gelijk aan de lengte van de vector $\vec{x_t}$.

Deze grootheden hebben we nodig voor het toepassen van het ensemble Kalman filter, om onze voorspelling te combineren met de daadwerkelijke data.

Omdat we gebruik maken van een algemene notatie, kunnen we later gemakkelijk parameters aan de vectoren toevoegen, zoals β of γ. Voegen we bijvoorbeeld

β toe, ziet de vergelijking er hetzelfde uit, maar wordt $\vec{x}_t = \begin{bmatrix} S_t \\ I_t \\ R_t \\ \beta_t \end{bmatrix}$ en

$$f(\vec{x_t}, t) = \begin{bmatrix} -\beta * s(t)i(t) \\ \beta * s(t)i(t) - \gamma * i(t) \\ \gamma * i(t) \\ 0 \end{bmatrix}.$$

Door β toe te voegen en hier ook het EnKF op toe te passen, wordt ook de waarde van β bij elke analyse stap gecorrigeerd, waardoor we deze ook niet meer hoeven te schatten.

3.3.1 Illustratief voorbeeld

Om onze voorspellingen te combineren met de daadwerkelijke data, hebben we vergelijkingen nodig voor beide modellen. Voor onze voorspelling ziet deze eruit als:

$$x^f = x + w,$$

waarbij x^f het resultaat van het forecast model is, x de onbekende werkelijke waarde en w een onbekende random model fout. Deze model fout heeft een variantie P^f, die we net berekend hebben en een maat voor de onzekerheid van het model is. De model fout heeft een gemiddelde van 0.

De vergelijking voor de meting is:

$$z = x + v,$$

waarbij z de meting is, x weer de onbekende werkelijke waarde en v de meetfout die ontstaat door bijvoorbeeld een fout bij het registreren van het aantal overledenen of het ontbreken van gegevens. Deze meetfout heeft een gemiddelde van 0 en een variantie R, niet te verwarren met de R uit het SIR model of de R_0 waarde.

Om x, de werkelijke waarde die we zoeken, zo goed mogelijk te schatten, gebruiken we beide methoden. We kiezen x^a, waarbij de a staat voor analysis (= model + meting), als volgt:

$$x^a = (1 - K)x^f + Kz,$$

waarbij x^f en z, de voorspelde waarde en de meting, bekend zijn. K noemen we de Kalman gain zoals we die gezien hebben bij het originele Kalman filter.

We kunnen nu van x^a het gemiddelde en de variantie bepalen.

$$\begin{aligned} E(x^a) &= E((1 - K)x^f + Kz) \\ &= E((1 - K)(x + w) + K(x + v)) \\ &= E((1 - K)E(x) + KE(x)) \\ &= (1 - K)x + Kx \\ &= x. \end{aligned}$$

Hieruit blijkt dat x een zuivere schatter is, wat de slimme keuze voor x^a bevestigt.

Voor de variantie geldt:

$$\begin{aligned} Var(x^a) &= E((x^a - E(x^a))^2) \\ &= E((1-K)x^f + Kz - ((1-K)x + Kx)^2) \\ &= E(((1-K)x^f - (1-K)x + Kz - Kx)^2) \\ &= E(((1-K)(x^f - x) - K(z-x))^2 \\ &= E(((1-K)w - Kv)^2 \\ &= (1-K)^2 P^f + K^2 R. \end{aligned}$$

Deze variantie zal een betere benadering zijn dan de variantie van het model of de meting, omdat we beide nu mee wegen.

3.3.2 Het EnKF met het SIR model

De vergelijking voor onze meting wordt

$$\vec{z} = M\vec{x} + \vec{v},$$

waarbij $\vec{z}$ een vector is die de dagelijkse metingen voor I bevat. $\vec{x}$ is de vector $\begin{bmatrix} S \\ I \\ R \end{bmatrix}$ met de werkelijke waarden voor S, I en R, en M weer de observatie-matrix. De meetfout $\vec{v}$ heeft ook hier een gemiddelde van 0 en een variantie van R.

Vervolgens stellen we $\vec{x^a}$ op, om de metingen met het model te kunnen combineren. Deze zal er nu als volgt uitzien:

$$(\vec{x^a})^i = (\vec{x^f})^i + K(\vec{z} - M * (\vec{x^f})^i + v^i),$$

met i de index van het ensemble member en v^i een fout die random wordt gekozen voor elk ensemble member. De Kalman gain K kan op dezelfde manier als bij het originele Kalman filter bepaald worden door

$$K = P^f M^T [MP^f M^T + R]^{-1},$$

en de waarden van $\vec{x^f}$ en $\vec{z}$ bekend zijn.

De volgende stap is om de analyse daadwerkelijk te gaan toepassen en dus de meting met ons model te combineren. We beginnen hiervoor met onze voorspelde waarden, deze berekenen we tot en met de eerste tijdstap waarop een meting plaatsvindt, noem deze tijdstap t_1. Voor t_1 bepalen we $(\vec{x^a})^i$ voor alle i ensemble members en vervangen we de voorspelde waarde door de geanalyseerde waarden. Deze geanalyseerde waarden geven ons nieuwe informatie die we gebruiken om onze voorspelling beter te maken. Vanaf t_1 bepalen we nieuwe waarden voor $(\vec{x^f})^i$ tot de tweede tijdstap waarvoor een meting is, zeg t_2, met $(\vec{x^a})^i$ als beginwaarden. Op deze manier corrigeert de analyse niet alleen de

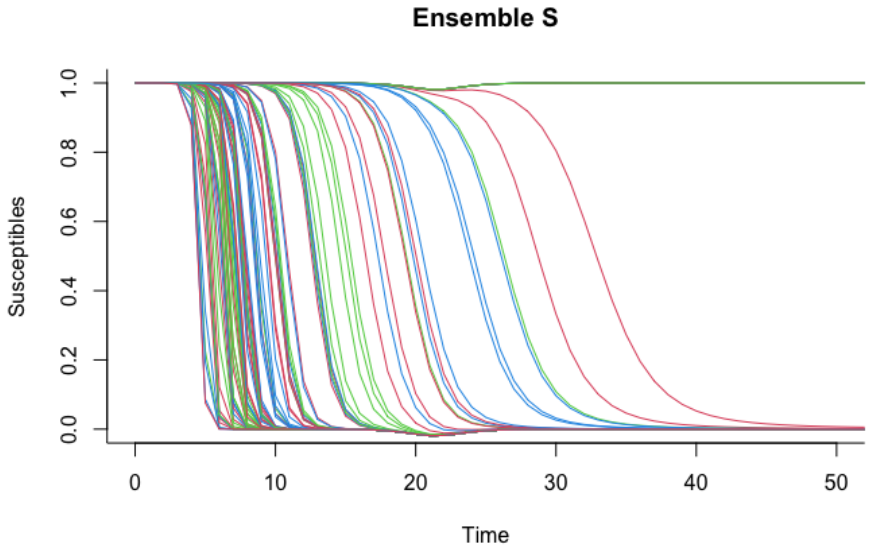
voorspelling voor I, maar ook voor S en R en komen voor alle drie de variabelen de ensemble members steeds dichter bij elkaar te liggen, wat uiteindelijk leidt tot de optimale voorspelling.

Later kan aan $\vec{x^f}$ bijvoorbeeld ook de parameter β worden toegevoegd, zodat ook deze steeds gecorrigeerd wordt en we hiervoor ook een optimale voorspelling vinden.

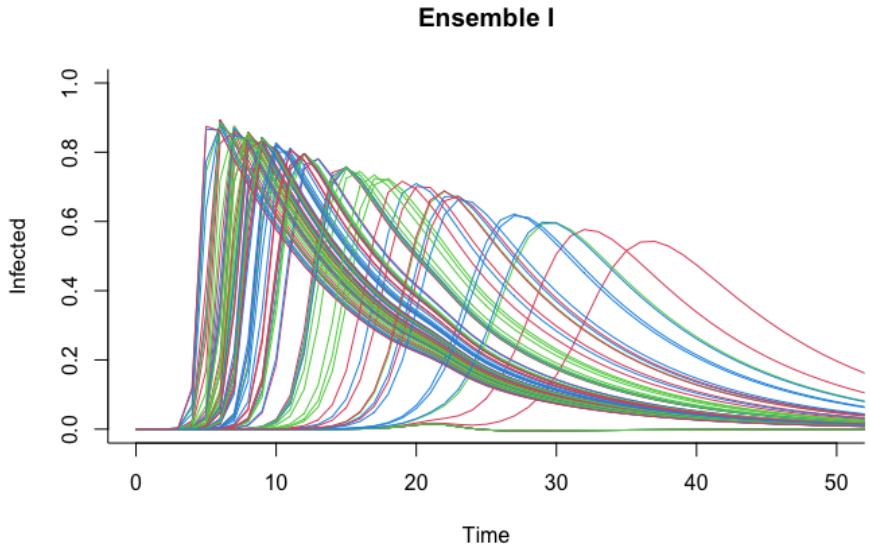
3.4 Modelleren van het EnKF

We modelleren het SIR model en het EnKF in Rstudio en doorlopen daarbij alle bovengenoemde stappen.

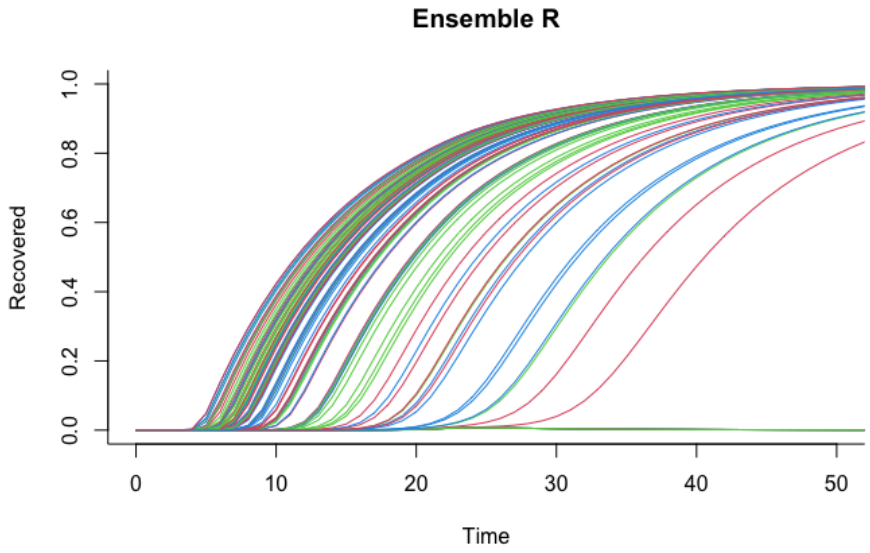
Allereerst creëren we voor S, I en R een ensemble, met i ensemble members, in dit geval 100. We doen dit door de parameter β te vervangen door de normaalverdeling met gemiddelde 1 en standaard deviatie 0.5. We krijgen dan voor S, I en R 100 mogelijke uitkomsten, geïllustreerd in de onderstaande plots. Hierbij is de tijd in dagen.



Figuur 3.1: Ensemble S



Figuur 3.2: Ensemble I



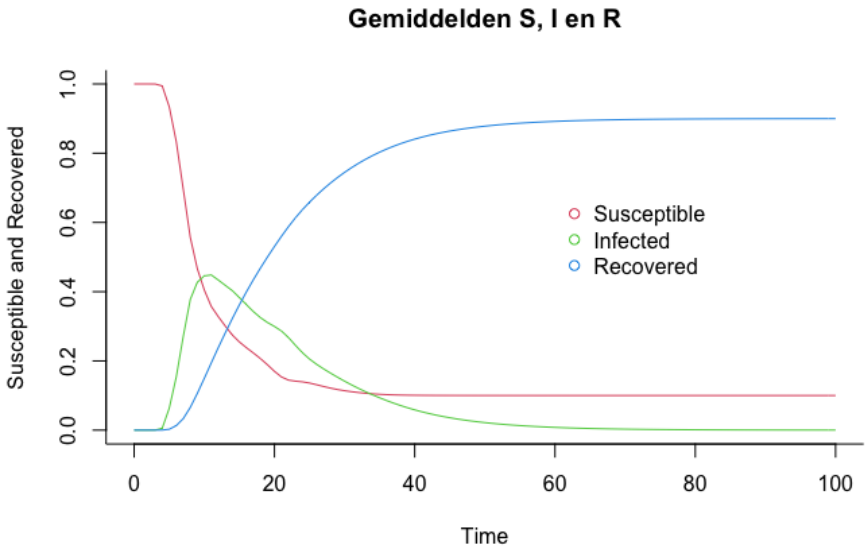
Figuur 3.3: Ensemble R

Uit deze ensembles kunnen we het gemiddelde en de covariantie matrix van S, I en R bepalen. Deze worden berekend door

$$Gemiddelde : \vec{x_t^f} = \frac{1}{N} \sum_i (\vec{x_t^f})^i,$$

$$Covariantiematrix : P^f = \frac{1}{N-1} \sum_i \left(\left[(x_t^f)^i - \vec{x_t^f} \right] * \left[(x_t^f)^i - \vec{x_t^f} \right]^T \right),$$

wat plot 3.4 oplevert voor de gemiddelden.

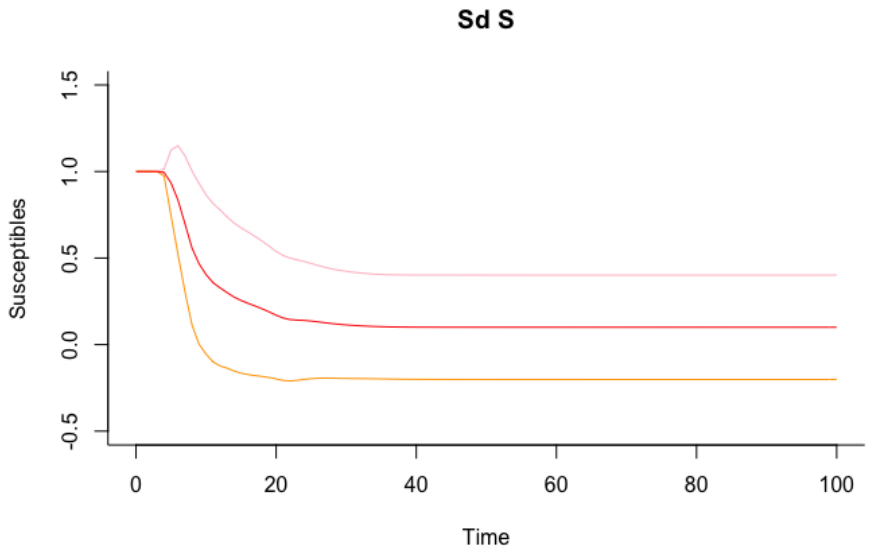


Figuur 3.4: Gemiddelden S, I en R

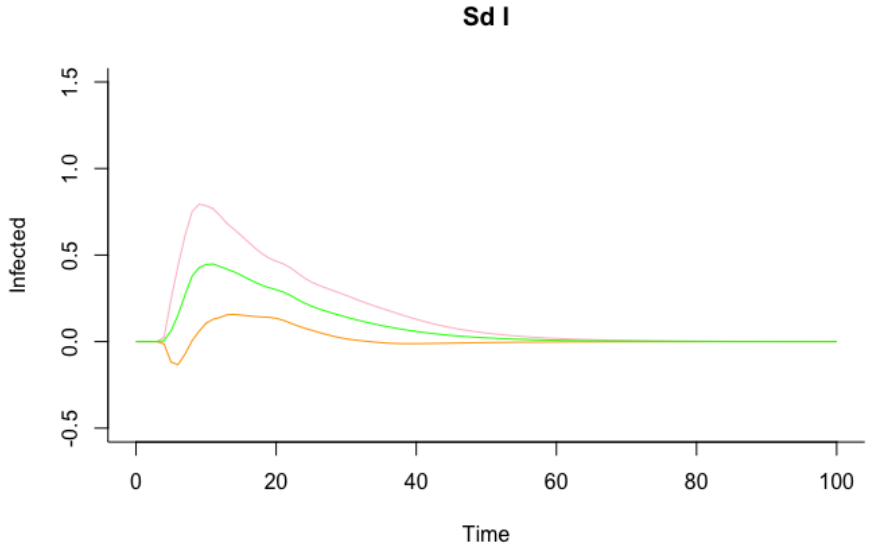
Uit de covariantie matrix kunnen we de standaard deviaties van S, I en R verkrijgen. Het kwadraat van de standaard deviatie van S staat op de eerste plek op de diagonaal van de covariantie matrix, die van I op de tweede en die van R op de derde:

$$Covariantiematrix : \begin{bmatrix} (sdS)^2 & ... & ... \\ ... & (sdI)^2 & ... \\ ... & ... & (sdR)^2 \end{bmatrix}$$

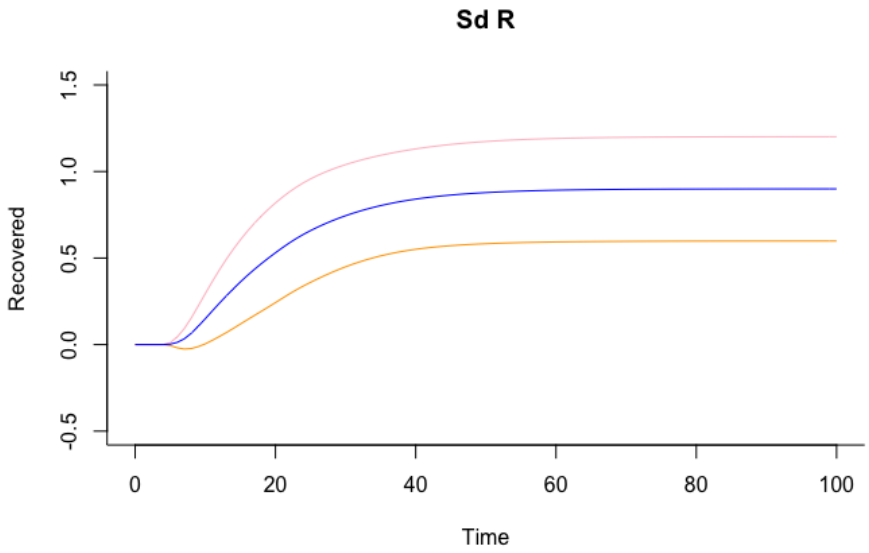
We kunnen de gemiddelden van S, I en R illustreren met de standaard deviaties. De standaard deviatie is niet op elk tijdstip hetzelfde. Het is goed voor te stellen dat op de momenten met veel metingen, de kans groter is dat er fouten gemaakt worden. We creëren met de standaard deviaties als het ware een 'band' om het gemiddelde heen, wat de range aangeeft waarin de waarden van S, I of R hoogstwaarschijnlijk liggen. Zie plots 3.5, 3.6 en 3.7, waarbij ook hier de tijd in dagen is uitgedrukt.



Figuur 3.5: Gemiddelde S met standaard deviatie



Figuur 3.6: Gemiddelde I met standaard deviatie



Figuur 3.7: Gemiddelde R met standaard deviatie

Als we de gegevens van x^f hebben berekend en op een rijtje gezet hebben, kunnen we ook de vergelijking voor de meting gaan opstellen:

$$\vec{z} = M\vec{x} + \vec{v}.$$

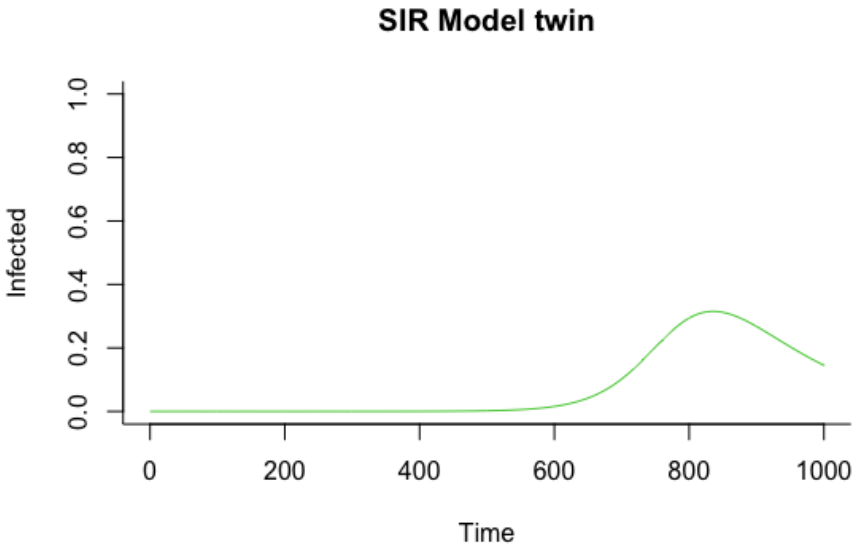
M wordt de vector $\begin{bmatrix}0 & 1 & 0\end{bmatrix}$, met 1 op de middelste plek, omdat we I meten. $\vec{v}$ is de meetfout, voor elk ensemble member verschillend.

3.4.1 Twin experiment

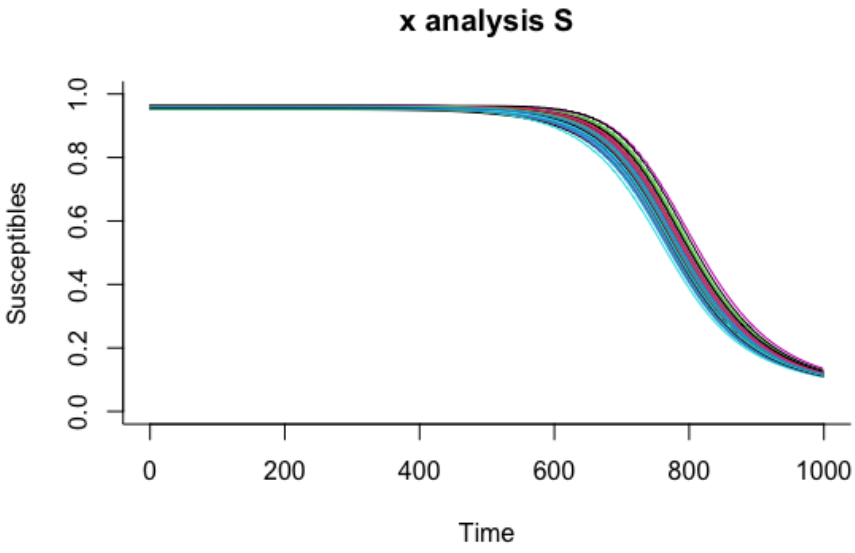
Allereerst doen we een zogenaamd twin experiment. Hierbij gebruiken we één van de ensemble members als data en combineren deze met onze voorspelde waarden. Op deze manier weten we zeker dat de metingen passen bij het model en kunnen we checken of het ensemble Kalman filter goed zijn werk doet. De figuur van de analyse moet dan vrijwel overeenkomen met de data.

In figuur 3.8 zien we de zelf gegenereerde data voor I uit het ensemble van I. In figuur 3.9, 3.10 en 3.11 zien we de analyse gedaan aan de hand van deze data.

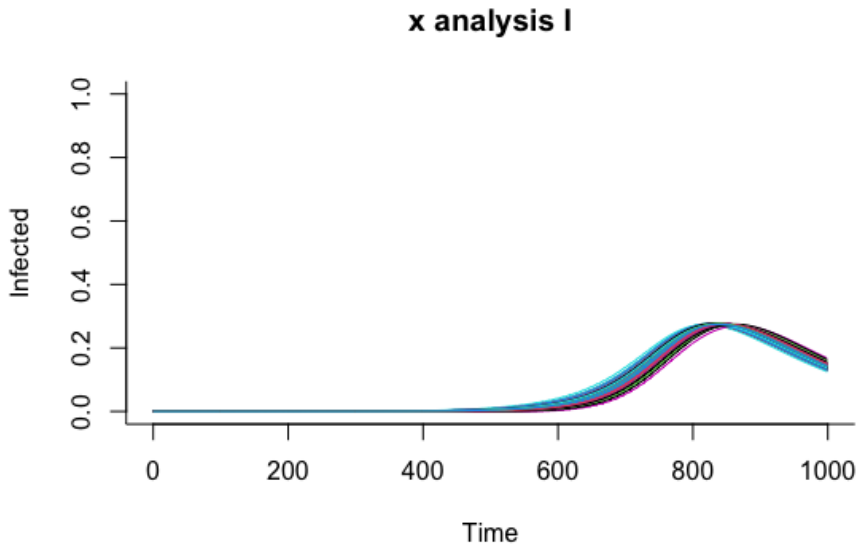
Vanaf hier gebruiken we tijdstappen van grootte 0,1, waardoor de tijd wordt uitgedrukt in tiende van dagen.



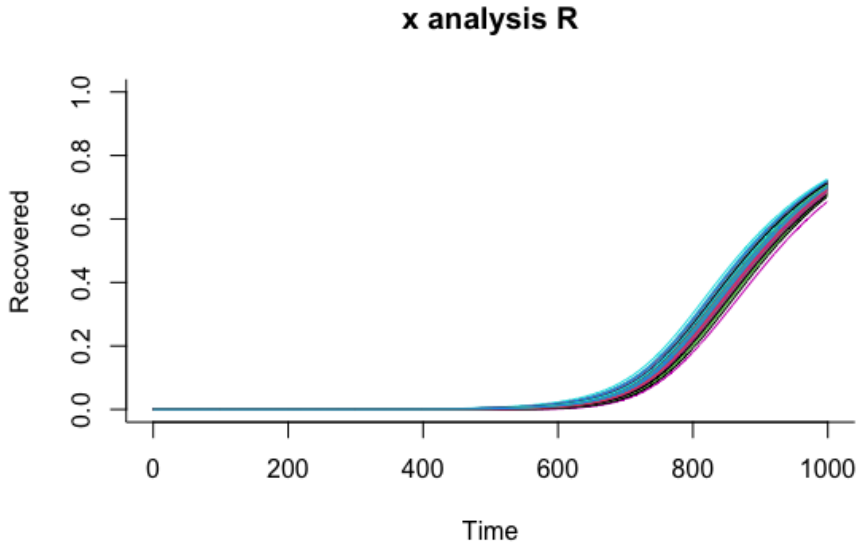
Figuur 3.8



Figuur 3.9: x-analysis S twin experiment



Figuur 3.10: x-analysis I twin experiment

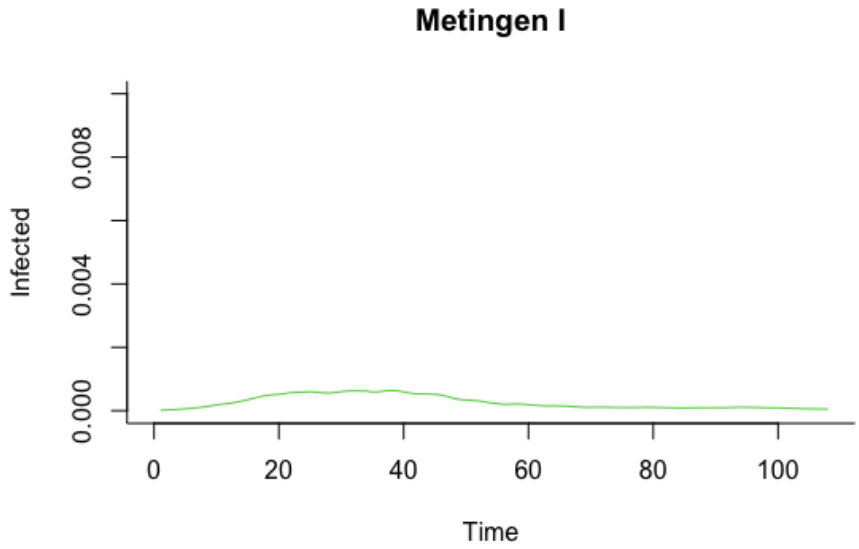


Figuur 3.11: x-analysis R twin experiment

We zien inderdaad dat de analyse de meting van I goed volgt en hier dus ook de analyses van S en R op aanpast. Hieruit kunnen we concluderen dat het filter zijn werk goed doet en dit ook zo zou moeten zijn als we het toepassen op realistische data.

3.4.2 De werkelijke data voor I

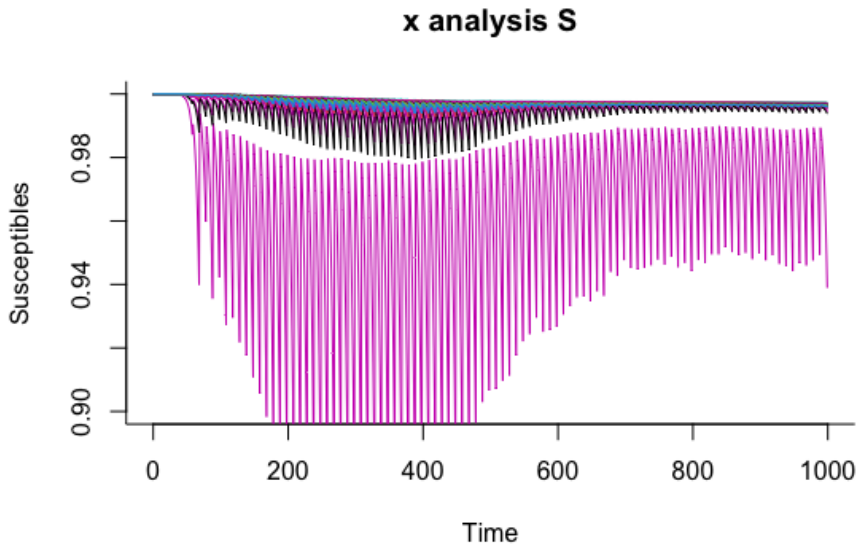
We meten I, dus we beginnen met de eerste 100 metingen van de bekende geïnfecteerden te importeren. Deze gegevens zijn verkregen van allecijfers.nl[19]. Dit wordt de vector $\vec{z}$. Omdat we nu alleen informatie hebben over het aantal nieuwe geïnfecteerden per dag, veranderen we de gegevens een beetje. We willen namelijk niet weten hoeveel nieuwe ziektegevallen er per dag zijn, maar hoeveel geïnfecteerden we totaal per dag hebben. Hiervoor gaan we uit van de gemiddelde duur van 10 dagen van het virus. Alle geïnfecteerden op dag 1 tellen we dus op bij het aantal geïnfecteerden van dag 2 t/m 10. Alle geïnfecteerden op dag 2 tellen we op bij het aantal geïnfecteerden van dag 3 t/m 11, enzovoorts. Uiteindelijk nemen we de metingen van de laatste 10 dagen niet mee in onze bepalingen, omdat we over deze data de aanvullende informatie over het totaal aantal geïnfecteerden niet compleet hebben. In figuur 3.12 is onze data weergegeven.



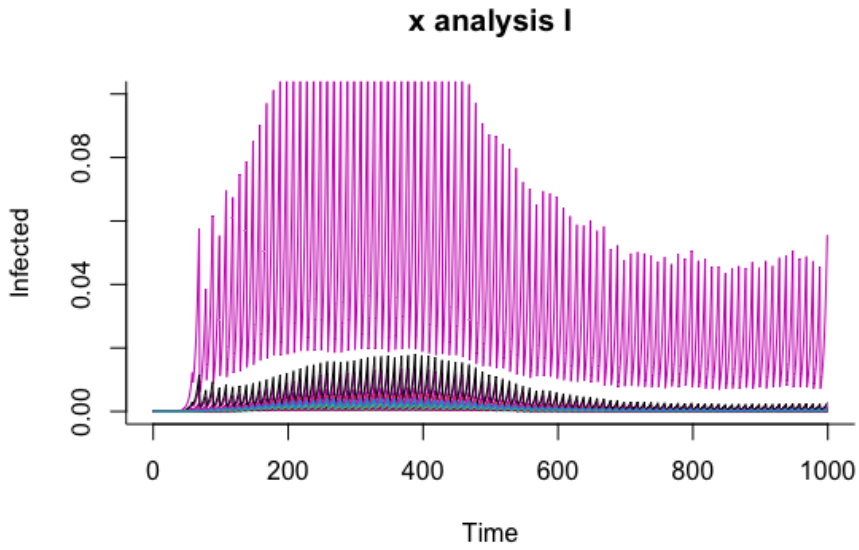
Figuur 3.12: Totaal aantal geïnfecteerden in Nederland per dag

Het ensemble Kalman filter combineert nu deze data met het ensemble, waaruit de analyses volgen, weergegeven in figuur 3.13, 3.14 en 3.15. Voor de precieze wijze van modelleren, zie Appendix A.

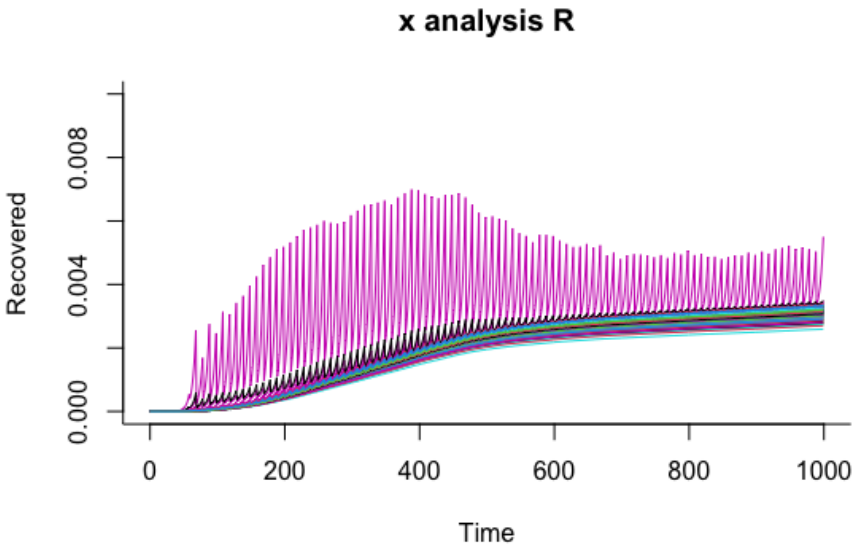
[19] https://allecijfers.nl/nieuws/statistieken-over-het-corona-virus-en-covid19/#Corona_-kerncijfers



Figuur 3.13: x-analysis S



Figuur 3.14: x-analysis I



Figuur 3.15: x-analysis R

Als we kijken naar de analyse van I, zien we dat het model de metingen goed volgt, op een enkel uitschietend ensemble member na.

Het valt direct op dat de metingen die we hier gebruiken erg verschilt van de voorspelde waarden. Het aantal geïnfecteerden ligt in werkelijkheid een stuk lager dan werd voorspeld aan de hand van het SIR model. We zien dit ook terug na het toepassen van het EnKF, de ensemble members schieten steeds omhoog, maar worden elke tijdstap dat er een meting plaatsvindt weer omlaag gehaald omdat deze meting een veel kleinere waarde is.

Het is duidelijk dat een of meerdere van onze geschatte waarden in het SIR model totaal niet overeenkomen met de werkelijk waarden. Het vermoeden is dat dit aan β ligt, aangezien dit de parameter is die op de minste informatie gebaseerd is. We voegen de parameter β nu toe aan ons model en passen hierop het EnKF toe, om erachter de komen of dit vermoeden juist is.

3.4.3 Toevoeging van de parameter β

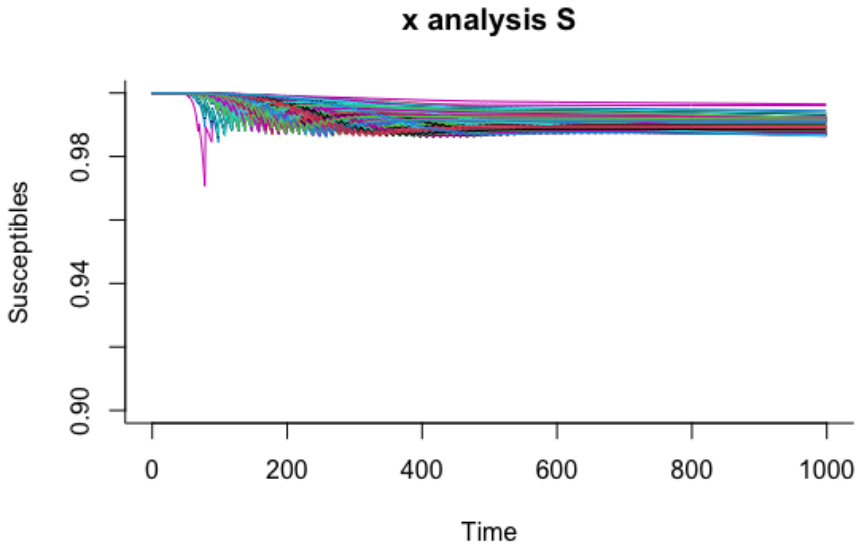
Zoals benoemd in sectie 3.3, komt ons model als we deze uitbreiden met β er als volgt uit te zien:

$$x_{t+\vec{\Delta t}} = \vec{x}_t + f(\vec{x_t}, t) * \Delta t,$$

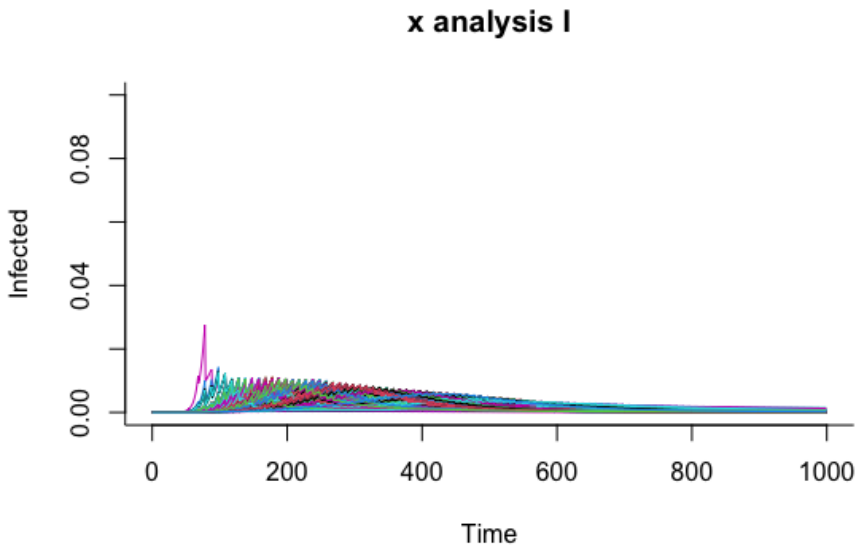
met $\vec{x}_t = \begin{bmatrix} S_t \\ I_t \\ R_t \\ \beta_t \end{bmatrix}$ en $f(\vec{x_t}, t) = \begin{bmatrix} -\beta * s(t)i(t) \\ \beta * s(t)i(t) - \gamma * i(t) \\ \gamma * i(t) \\ 0 \end{bmatrix}$, aangezien de afgeleide van β naar de tijd 0 is.

We beginnen met stellen dat β constant is, dat wil zeggen $\beta_{t+1} = \beta_t$. We nemen

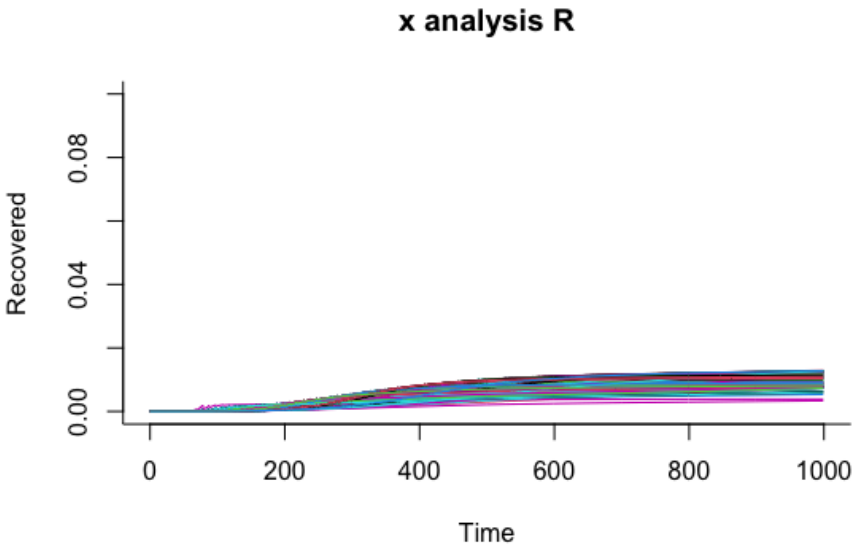
als beginwaarden voor β 100 random trekkingen met gemiddelde 1 en standaard deviatie 0.5. We krijgen dan figuren 3.16, 3.17, 3.18 en 3.19.



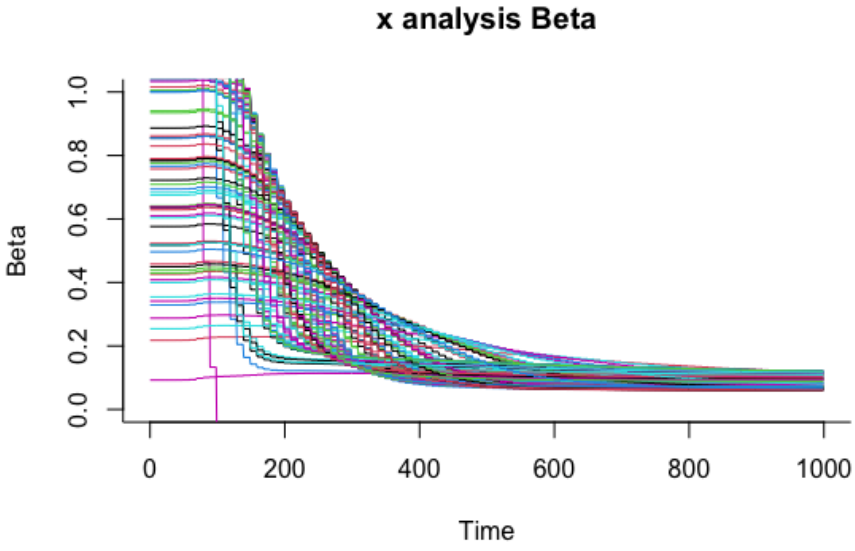
Figuur 3.16: x-analysis S na toevoeging β



Figuur 3.17: x-analysis I na toevoeging β



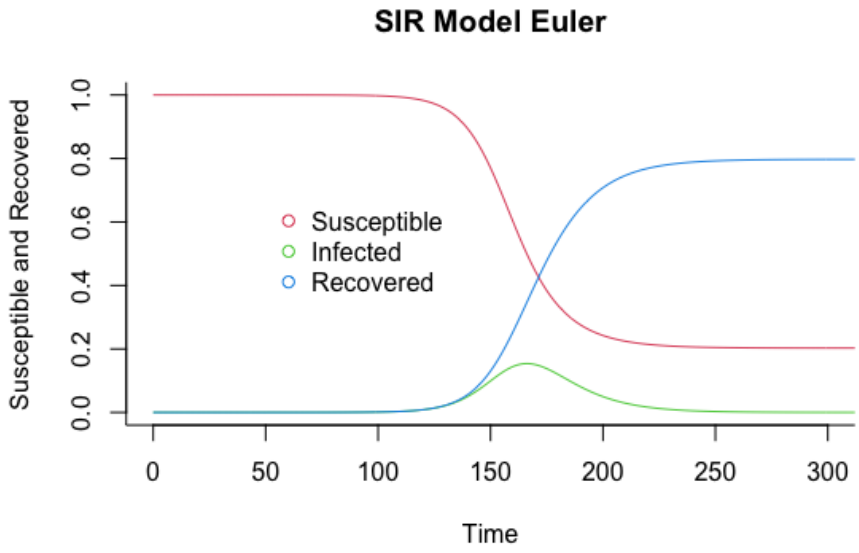
Figuur 3.18: x-analysis R na toevoeging β



Figuur 3.19: x-analysis constante β

Als we kijken naar figuur 3.19 zien we dat β na elke meting gecorrigeerd wordt en uiteindelijk convergeert naar iets onder de 0,2. De schatting van $\beta = 0,9$, zoals gedaan in sectie 2.1.5 lag te ver van de werkelijke waarde. Als we dit aanpassen in ons SIR model, krijgen we plot 3.20, wat al een stuk meer lijkt

op de daadwerkelijke data.



Figuur 3.20: Het SIR model met $\beta = 0,2$

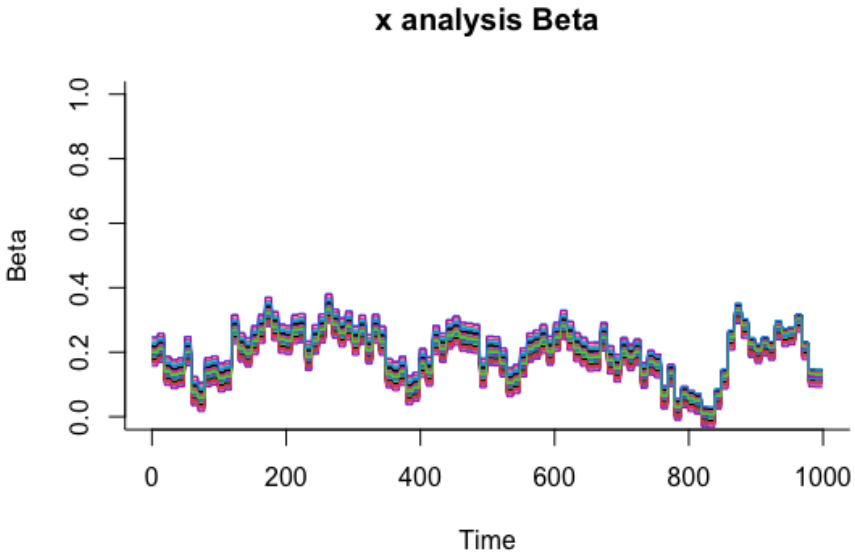
Ook zien we dat door β te corrigeren, de analyses voor S, I en R verbeterd worden. De ensemble members liggen dichter bij elkaar en er zijn minder uitschieters aanwezig.

In werkelijkheid is β natuurlijk niet constant, dus ook dat willen we modelleren om een nog realistischer weergave van de werkelijkheid te creëren. Dit doen we door na elke analyse stap een ruis toe te voegen aan β, zodat we krijgen $\beta_{t+1} = \beta_t + wk$ en onze model vergelijking wordt:

$$\vec{x_{t+\Delta t}} = \vec{x}_t + f(\vec{x_t}, t) * \Delta t + wk.$$

Hierbij is wk de fout van het systeem met gemiddelde 0 en standaard deviatie 0,05.

De analyse van β ziet er dan uit als in figuur 3.21. Voor de beginwaarden voor β nemen we 100 random trekkingen met gemiddelde 0,2 en standaard deviatie 0,02.



Figuur 3.21: x-analyse niet-constante β

β convergeert nu niet meer, maar blijft over de tijd schommelen rond 0,2. De grafiek van β ziet er nu een beetje uit zoals figuur 2.4, wat ook logisch is aangezien β samenhangt met het reproductiegetal.

3.4.4 Conclusie

In figuur 3.16, 3.17 en 3.18 is weergegeven hoe de voorspelling, die we in sectie 2 deden, wordt gecombineerd met de data over het deel van de bevolking dat in Nederland geïnfecteerd was met het virus. Als we kijken naar de analyse van I, het geïnfecteerde deel van de bevolking de eerste 100 dagen, zien we dat het aantal geïnfecteerden nauwelijks hoger dan 0,02 komt, wat ongeveer 2% van de bevolking vertegenwoordigt. Dit klopt redelijk, na 100 dagen lag het totaal aantal besmettingen in Nederland op 49.426, 0,28% van de bevolking. Hieruit kan worden afgeleid dat de gemaakte analyse klopt.

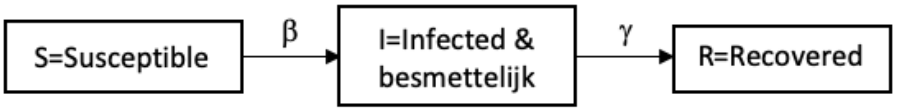
Dat het aantal geïnfecteerden in Nederland laag is gebleven tot nu toe, hangt samen met de waarde van β. Zoals te zien in figuur 3.21 blijft de β-waarde schommelen rond de 0,2, wat betekent dat de overgangssnelheid van het S compartiment naar het I compartiment 0,2 is, wat laag is. Als we dus willen voorkomen dat we het virus weer de vrije loop laten, moeten we zorgen dat β, en daarmee het reproductie getal R, laag blijft.

4 Uitbreidingen van het SIR model

In hoofdstuk 2 hebben we het SIR model besproken, waarbij we alleen de basis variabelen in acht namen. Dit waren de vatbare individuen S, de geïnfecteerden I, de genezen groep R en de parameters β en γ. Echter geeft dit slechts een heel basale voorspelling, ten opzichte van de werkelijkheid. Ook maakten we hier een aantal aannames, die het model vereenvoudigen. Maar we willen dat het model een zo realistisch mogelijke weergave van de werkelijkheid weergeeft. Hiervoor moeten we wat beter naar het model kijken en meer parameters in acht nemen. In deze sectie zullen we kijken naar de uitbreidingen van het SIR model en bespreken we experimenten die we hiermee kunnen uitvoeren.

4.1 Verschillende uitbreidingen

Het basis model bekijkt de groepen S, mensen die vatbaar zijn voor de ziekte, I, mensen die geïnfecteerd en dus besmettelijk zijn, en R, personen die hersteld zijn van het virus, met bepaalde overgangen tussen deze groepen:



Figuur 4.1: De compartimenten van het SIR model

Om een realistischer beeld van de werkelijkheid te schetsen, moeten we niet alleen kijken naar de vatbare, de geïnfecteerden en de genezen personen, maar moeten we ook een aantal andere groepen bekijken.[20]
Zo is een eerste voor de hand liggende uitbreiding het SIS model, waarbij niemand geneest na besmet te zijn geweest met het virus, maar iedereen opnieuw vatbaar wordt. Ook zijn er ziekten en virussen waarvoor bepaalde mensen al bij de geboorte immuun zijn. Dit heet maternale immuniteit en deze ziekten worden beschreven door het $MSIR$ model.

Bovengenoemde uitbreidingen voeren grote veranderingen door ten opzichte van het SIR model. Er zijn echter ook uitbreidingen die voortborduren op het SIR concept, maar alleen extra groepen toevoegen waar, naast S, I en R ook rekening mee gehouden moet worden. Voor het COVID-19 virus bekijken we het $SEIQHRF$ model, waarbij 4 extra compartimenten worden toegevoegd. Deze nieuwe compartimenten voegen nogal wat toe aan de eerdere parameters. Ook

[20] https://nl.wikipedia.org/wiki/Ziektecompartimentenmodel

doen we een aantal experimenten hiermee, om een idee te krijgen van het verloop van de verschillende compartimenten bij veranderingen in de parameters.

We bestuderen hiervoor de Health Data Sciences blog van Tim Churches, waarin hij dit model bespreekt en modelleert in Rstudio.[21]

4.2 Het SEIQHRF model

Om te beginnen kunnen we de groep geïnfecteerden opsplitsen in twee categorieën: personen die geïnfecteerd raken met het virus en daar klachten van ondervinden, en personen die geïnfecteerd raken, maar geen symptomen krijgen. Personen in deze laatste groep weten om deze reden vaak niet dat ze het virus hebben opgelopen. We geven deze groep aan met de variabele E. We nemen aan dat deze groep niet weet het virus bij zich te dragen, maar het wel kan overdragen op andere individuen. Mensen die wel symptomen hebben van het virus omschrijven we met de letter I.

Ook binnen de groep I kunnen we verschillende categorieën onderscheiden. Zo is er een deel van deze groep dat in thuis-quarantaine gaat en daarmee de kans op het overdragen van het virus sterk verkleint. We geven deze groep aan met de letter Q.

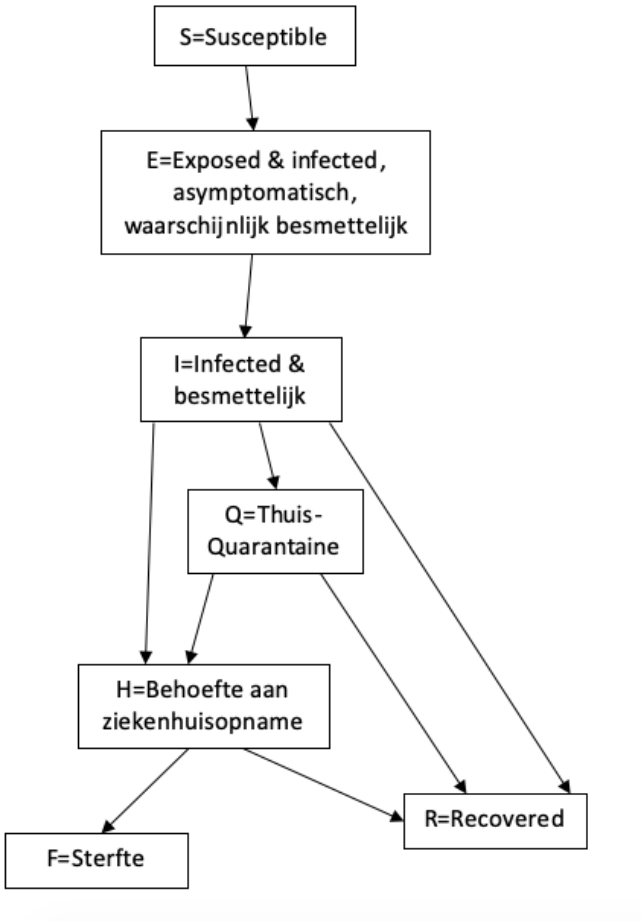
Daarnaast is er een deel van de groep I dat gezondheidszorg nodig heeft. We geven deze groep aan met de letter H. Dit zijn dus niet de personen die daadwerkelijk in het ziekenhuis liggen, maar de mensen waarbij het virus zo ernstig optreedt, dat een ziekenhuisopname nodig is. We nemen aan dat deze mensen opgenomen worden in een ziekenhuis als er voldoende ziekenhuis capaciteit is. In het $SEIQHRF$ model wordt de ziekenhuis capaciteit meegenomen.

Hierop inhakend geeft de letter F het deel van de geïnfecteerden aan voor wie het virus fataal was. Hierbij wordt ook rekening gehouden met andere sterfgevallen in de bevolking en negeren we deze niet langer, zoals we dat wel deden bij het SIR model.

De letters S en R geven nog steeds de vatbare en de genezen individuen aan.

De overgangen tussen deze groepen worden dan zoals in figuur 4.2.

[21] https://timchurches.github.io/blog/posts/2020-03-18-modelling-the-effects-of-public-health-interventions-on-covid-19-transmission-part-2/

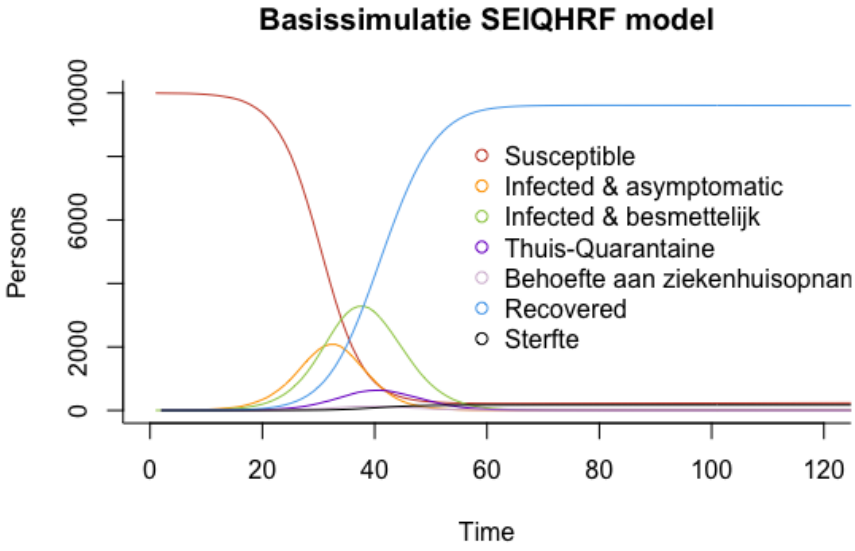


Figuur 4.2: De compartimenten van het $SEIQHRF$ model

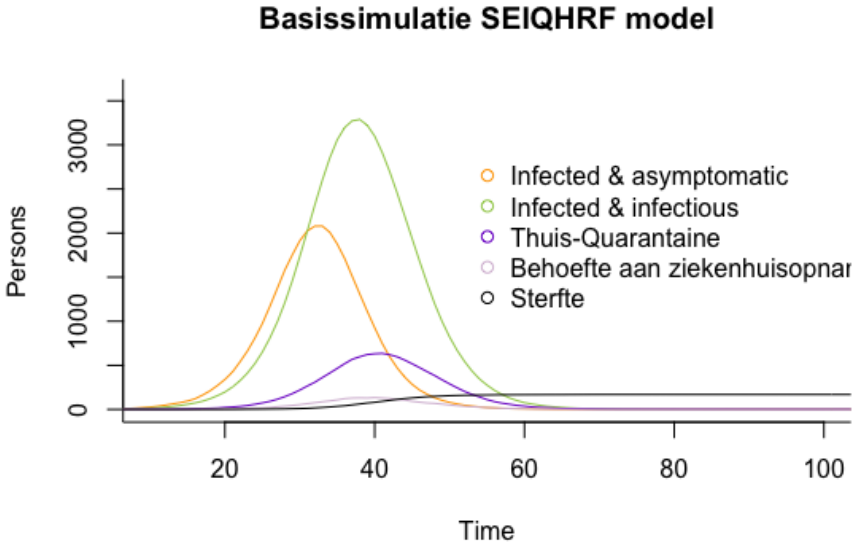
In figuren 4.3 en 4.4 wordt geïllustreerd hoe het $SEIQHRF$ model eruit zou komen te zien.
Dit model beschrijft een hypothetische bevolking van 10.000 personen. Verder nemen we een aantal dingen aan, zie hiervoor Appendix B. De belangrijkste aannames zijn dat:

1. Er 3 personen besmet waren met het virus op tijdstip 0,
2. Er in totaal 40 ziekenhuisbedden beschikbaar zijn,
3. Het aantal mogelijke overdrachtsmomenten tussen besmettelijke individuen in de groepen I,E en vatbare individuen in de groep S, 10 per dag is,
4. Er per dag 1/30 van de groep I in thuis-quarantaine gaat,
5. Er per dag 1/100 van de groepen I of Q verplaatsen naar de groep H,
6. 1/25 van de geïnfecteerde personen overlijdt aan de ziekte.

Om de toevoegingen duidelijker te laten zien, is de plot ook afgebeeld zonder de Susceptibles en de Recovered.



Figuur 4.3: Het $SEIQHRF$ model



Figuur 4.4: Het $SEIQHRF$ model ingezoomd

We kunnen nu een aantal dingen afleiden uit deze plots.

1. De epidemie is na 2 maanden voorbij,
2. Het aantal personen dat ziekenhuisopname nodig heeft, lijkt redelijk,
3. Het aantal sterfgevallen neemt monotoon toe.

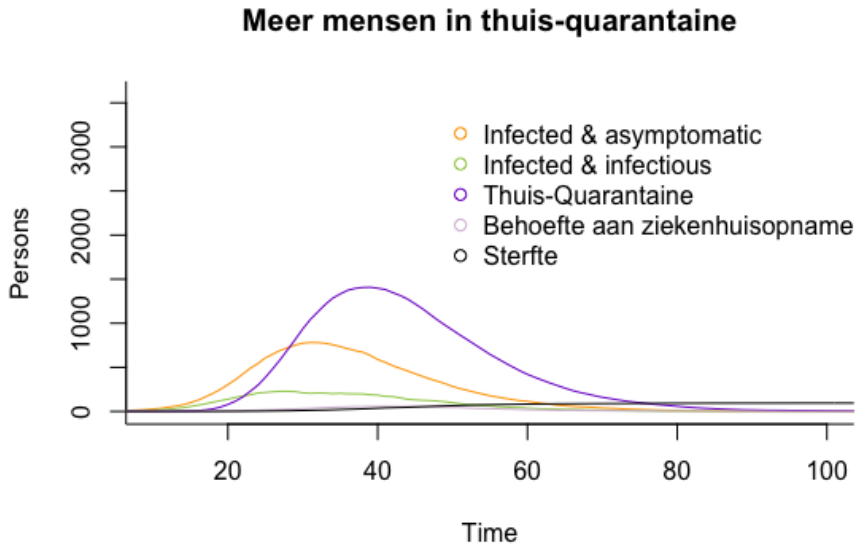
Omdat dit een hypothese is en niet de echte wereld weergeeft, geeft het model een indicatie van het gedrag van het virus in de werkelijkheid, maar kunnen de waarden enigszins verschillen. Wel kunnen we een aantal experimenten uitvoeren met het model, waardoor we bijvoorbeeld de effecten van maatregelen kunnen bestuderen.

4.3 Experimenten met het SEIQHRF model

Door experimenten toe te passen op het $SEIQHRF$ model bekijkt Tim Churches in zijn artikel de effecten van maatregelen die kunnen worden genomen tijdens een epidemie. Het is belangrijk dit soort experimenten te doen, om in de toekomst een idee te hebben van welke maatregelen het meeste effect hebben en welke daarnaast haalbaar zijn. We zullen er een aantal bespreken.

4.3.1 Thuis-quarantaine

Vrijwel direct nadat het virus in Nederland een aantal besmettingsgevallen had veroorzaakt, werd er een lockdown afgekondigd. Er werd opgedragen thuis te werken als die mogelijkheid er was, de scholen en universiteiten begonnen met online lessen en bijna alle sociale ontmoetingsplaatsen werden gesloten. Het is nogal wat gevraagd van een heel land om in quarantaine te gaan en ook de economie zou er zeker onder leiden. Om te besluiten tot een lockdown, moet dus vrij zeker zijn dat dir groot effect zal hebben. We bekijken de effecten van een thuis-quarantaine periode als we de isolatiesnelheid verhogen van 0,033 (zoals in de basissimulatie), beginnend op dag 15, tot 0,5 op dag 30:



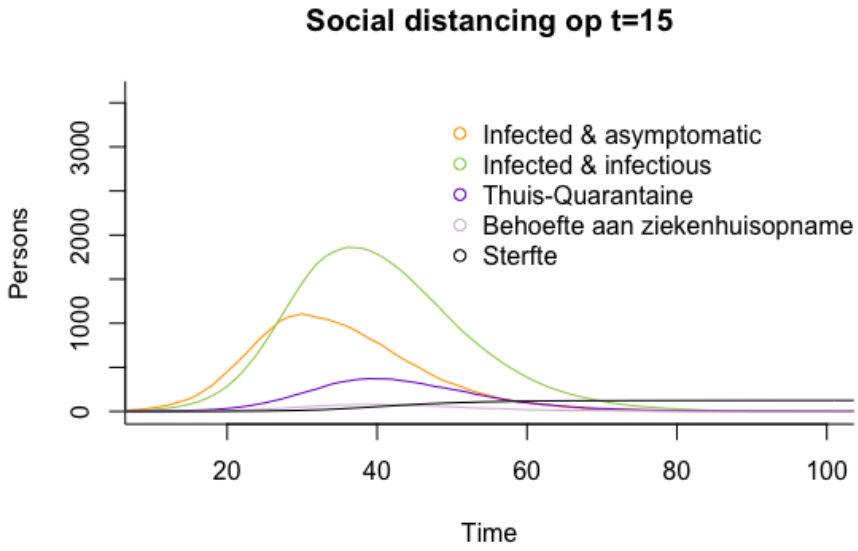
Figuur 4.5: Meer mensen in thuis-quarantaine

Zoals te zien in figuur 4.5 is het aantal geïnfecteerden, zowel met als zonder symptomen, in vergelijking met figuur 4.4 erg afgenomen. Thuis-quarantaine is dus inderdaad een hele goede manier om te voorkomen dat het virus zich snel verspreidt.

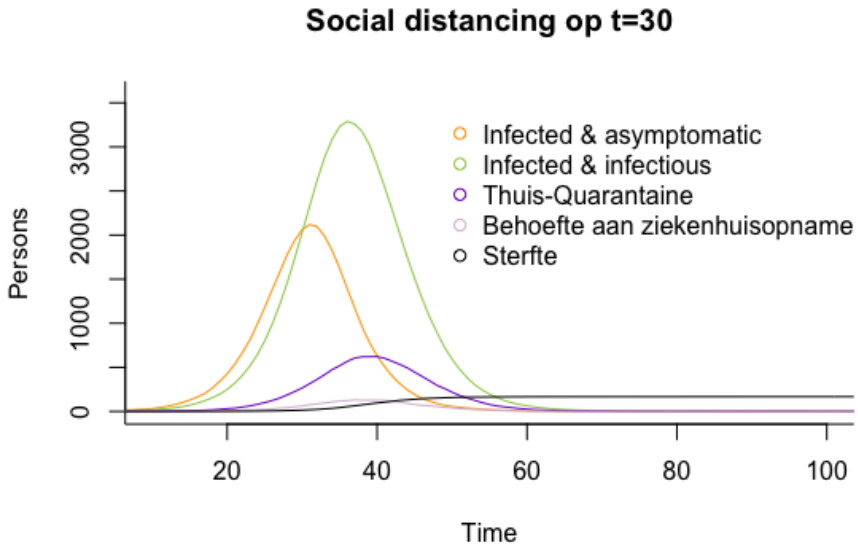
Toch is een land in quarantaine zetten een erg drastische maatregel. De overheid heeft waarschijnlijk liever een andere oplossing, waaronder de economie minder lijdt.

4.3.2 Social distancing

Een mogelijke alternatief is social distancing. We bekijken twee gevallen waarbij social distancing wordt ingevoerd, de eerste op dag 15 en de tweede op dag 30:



Figuur 4.6: Social distancing t=15



Figuur 4.7: Social distancing t=30

Het valt meteen op dat deze twee figuren heel verschillend zijn. Voeren we social distancing in op dag 15, dan maakt dit wel degelijk een groot verschil. De piek van de geïnfecteerden verplaatst niet, maar wordt wel $\frac{1}{3}$ keer kleiner. Dit is $\frac{1}{10}$ deel van de totale bevolking in dit voorbeeld en dus zeer de moeite waard.

Voeren we echter pas op dag 30 de social distancing in, verandert het verloop van de geïnfecteerden nauwelijks.
Het is dus zeker aan te raden bij een epidemie zo snel mogelijk social distancing in te voeren. Het is echter nog niet zó effectief als thuis-quarantaine, waarbij het geïnfecteerden aantal nog vele malen lager ligt.

De figuren die gegenereerd worden bij dit soort experimenten geven ons ontzettend waardevolle informatie. Deze informatie is zeer nuttig om een voorspelling te doen over het verloop van de huidige COVID-19 epidemie, maar kan ook in de toekomst gebruikt worden om bij vergelijkbare epidemieën maatregelen eerder toe te passen. We weten immers dan basaal wat te verwachten is.

Ook kunnen er op veel meer manieren experimenten gedaan worden. Zo kan gekeken worden naar het effect van het verhogen van de ziekenhuis capaciteit en kunnen we testen in hoeverre het dragen van mondkapjes helpt bij het bestrijden van het virus. Daarnaast kunnen er natuurlijk meerdere maatregelen tegelijk toegepast worden.

5 Conclusie en discussie

In dit verslag hebben we geprobeerd het verloop van het Corona virus in Nederland de afgelopen maanden in kaart te brengen. We gebruikten hiervoor het SIR model, een standaard model voor het voorspellen van ziekten.
Om een beter beeld te krijgen van de situatie in Nederland, verzamelden we informatie van het Rijksinstituut voor Volksgezondheid en Milieu, die veel data vrijgeven over de huidige situatie.
We gebruikten deze data om met het SIR model te combineren. Hiervoor bleek het Ensemble Kalman Filter een geschikte methode. De methode corrigeert de voorspelde waarden aan de hand van de meetgegevens.

Zoals genoemd in sectie 3.4.4 hebben we tot nu toe het aantal geïnfecteerden in Nederland redelijk laag weten te houden, maar moeten we vooral zorgen dat dit zo blijft.
In figuren 4.5 en 4.6 zijn de effecten van thuis-quarantaine en vroege social distancing duidelijk te zien. We kunnen dus concluderen dat Nederland er goed aan heeft gedaan vroegtijdig een *intelligente lockdown* af te kondigen.

Uiteraard is hier slechts een heel klein deel van het onderzoek naar epidemiologische modellen uitgelicht. Er zijn ontelbaar extra mogelijkheden om hier onderzoek naar te doen. Allereerst kan het SIR model of het $SEIQHRF$ model nog veel verder uitbreid worden. Zo kan het concept van dragers van het virus optreden, kan immuniteit een rol spelen en verandert er veel als er een vaccin voor het virus is uitgevonden.

Daarnaast hebben er nog veel meer factoren invloed op de verspreiding van het virus. Denk hierbij aan leeftijd, geslacht, onderliggende ziekten of zwangerschap en reisgedrag van de bevolking. Ook topografische aspecten spelen een rol.

Voor verder, realistischer onderzoek moet het model dus zo ver mogelijk uitbreid worden en moeten de waarden in het model worden aangepast naar de werkelijke waarden in Nederland. Dit vereist veel informatie over de Nederlandse ziekenhuis capaciteit, het aantal sterfgevallen door het virus, de mate waarin mensen zich aan de maatregelen houden et cetera.
Vervolgens zou ook op dit uitgebreider model het ensemble Kalman filter toegepast moeten worden. De uitkomsten hiervan zullen een stuk betrouwbaarder zijn dan de voorspellingen uit slechts het SIR model.

Met wiskundige modellen kunnen we dus de gezondheid van de Nederlandse bevolking verbeteren.
Ik heb dit onderzoek met plezier gedaan en ben erdoor overtuigd geraakt van het nut en de noodzaak van de verschillende genomen maatregelen.

Appendices

A Eerste appendix; Modelleren

A.1 Basis SIR model

Listing A.1: Code basis SIR model RStudio

```
#install.packages('EpiModel')

## Load deSolve package
library(deSolve)

## Create an SIR function
sir <- function(time, state, parameters) {

  with(as.list(c(state, parameters)), {

    dS <- -beta * S * I
    dI <-  beta * S * I - gamma * I
    dR <-                 gamma * I

    return(list(c(dS, dI, dR)))
  })
}

### Set parameters
## Startwaarden, als deel vd bevolking
init       <- c(S = 1-5.7e-8, I = 5.7e-8, R = 0.0)
## beta: infection parameter; gamma: recovery parameter
parameters <- c(beta = rnorm(1, mean = 2, sd = 1), gamma = 0
    .1)
## Time frame
times      <- seq(0, 150, by = 1)

## Solve using ode (General Solver for Ordinary Differential
     Equations)
out <- ode(y = init, times = times, func = sir, parms =
    parameters)
## change to data frame
out <- as.data.frame(out)
## Delete time variable
out$time <- NULL
## Show data
head(out, 10)


## Plot
matplot(x = times, y = out, type = "l",
        xlab = "Time", ylab = "Susceptible␣and␣Recovered",
        xlim=c(0, 100), ylim=c(0, 1),
        main = "SIR␣Model␣basic", lwd = 1,
        lty = 1, bty = "l", col = 2:4)
```

```
## Add legend
legend(60, 0.7, c("Susceptible", "Infected", "Recovered"),
       pch = 1, col = 2:4, bty = "n")
```

A.2 SIR model Euler's methode

Listing A.2: Code SIR model Euler's methode RStudio

```
euler_sir <- function(S0, I0, R0, beta, gamma, days){
  b <<- beta
  g <<- gamma
  delta_t <<- 0.1
  D <- days

  I <- numeric(D)
  S <- numeric(D)
  R <- numeric(D)

  I[1] <- I0
  S[1] <- S0
  R[1] <- R0

  n=1
  for (n in 1:(D)) {
    S[n+1] = S[n] + (-b*S[n]*I[n]*delta_t)
    I[n+1] = I[n] + (b*S[n]*I[n]*delta_t - g*I[n]*delta_t)
    R[n+1] = R[n] + g*I[n]*delta_t
  }
  return(data.frame(S, I, R))
}

dff <- euler_sir(1-0.000000057, 0.000000057, 0, 0.9, 0.1, 10
    000)
head(dff)

dagenx <- seq(0, 10000*delta_t, by = 1)
#aanpassen zodat het per dag geplot wordt
stap <- 1/delta_t
#print(stap)
plotlijst = dff[seq(1, nrow(dff), stap), ]
head(plotlijst)

matplot(x = dagenx, y = plotlijst,
        type = "l", xlab = "Time", ylab = "Susceptible␣and␣
            Recovered",
        xlim=c(0, 100), ylim=c(0, 1), main = "SIR␣Model␣
            Euler", lwd = 1,
        lty = 1, bty = "l", col = 2:4)
legend(70, 0.7, c("Susceptible", "Infected", "Recovered"),
    pch = 1,
       col = 2:4, bty = "n")
```

Listing A.3: Code vectorvorm Euler's methode RStudio

```
source("Euler␣Sir␣1.R")

xkfun <- function(S0, I0, R0, beta, gamma, tijd){
  matxt <- euler_sir(S0, I0, R0, beta, gamma, tijd)
  matxt <- as.matrix(matxt)

  matfxt <- matrix(data = 0, nrow = (tijd+1), ncol=3)
  for(t in 1:tijd){
    matfxt[t,1] = -b * matxt[t,1] * matxt[t,2] #dS
    matfxt[t,2] = b * matxt[t,1] * matxt[t,2] - g * matxt[t,
        2] #dI
    matfxt[t,3] = g * matxt[t,2] #dR
  }

  xkk <- matxt + matfxt*delta_t
  return(xkk)
}
```

Listing A.4: Code gemiddelde ensemble RStudio

```
gemfun <- function(s, i, r){
  gemlijstS = rowSums(s)/100
  gemlijstI = rowSums(i)/100
  gemlijstR = rowSums(r)/100
  return(data.frame(gemlijstS, gemlijstI, gemlijstR))
}
```

Listing A.5: Code covariantie matrix Rstudio

```
covarmatfun <- function(SS, II, RR, gemlijstSIR, tijd){
  matvarS <- sweep(SS, 1, gemlijstSIR[,1])
  matvarI <- sweep(II, 1, gemlijstSIR[,2])
  matvarR <- sweep(RR, 1, gemlijstSIR[,3])

  covarmatrix <<- list()

  for(t in 1:tijd){

    matvar <- matrix(0, 3, 1)
    varvec <- list()
    for(i in 1:100){
      matvar[1,] = matvarS[t,i]
      matvar[2,] = matvarI[t,i]
      matvar[3,] = matvarR[t,i]
      varvec[[i]] <- matvar
    }

    #vectoren keer getransponeerden doen
    #covarmatrix is nu een lijst van alle 3 bij 3
        covarmatrices
    varmatlist <- list()
    varmat3 <- matrix(0, 3, 3)
    #covarmatrix <- list()
```

```
    for(i in 1:100){
      varmat3 <- varvec[[i]] %*% t(varvec[[i]])
      varmatlist[[i]] <- varmat3
    }
    covarmatrix[[t]] <- (Reduce('+', varmatlist))/99
  }
  return(covarmatrix)
}
```

A.3 Ensembles en standaard deviaties

Listing A.6: Code ensembles en standaard deviaties Rstudio

```
source("SIR basic.R")
source("Euler Sir 1.R")
source("gemfun.R")
source("covarmatfun.R")
source("xkfun.R")

#basis plot met xk functie
xk <- xkfun(1-0.000000057, .000000057, 0, 0.9, 0.1, 1000)

dagenxk <- seq(0, 1000*delta_t, by = 1)
stap <- 1/delta_t
plotlijstxk = xk[seq(1, nrow(xk), stap), ]


matplot(x = dagenxk, y = plotlijstxk,
        type = "l", xlab = "Time", ylab = "Susceptible and
            Recovered",
        xlim=c(0, 100), ylim=c(0, 1),
        main = "SIR Model Euler vector vorm", lwd = 1, lty =
            1,
        bty = "l", col = 2:4)
legend(70, 0.7, c("Susceptible", "Infected", "Recovered"),
    pch = 1,
       col = 2:4, bty = "n")



#100x S, I en R plotten met xk funtie
SS <- matrix(data = 0, nrow = 1001, ncol=100)
II <- matrix(data = 0, nrow = 1001, ncol=100)
RR <- matrix(data = 0, nrow = 1001, ncol=100)

for(i in 1:100){
  xk <- xkfun(1-0.000000057, .000000057, 0,
              rnorm(1, mean = 1, sd = 0.5),0.1, 1000)
  SS[,i] = xk[,1]
  II[,i] = xk[,2]
  RR[,i] = xk[,3]
}

plotx <- seq(0, 1000*delta_t, by = 1)
```

```
stap <- 1/delta_t
plotlijstSS = SS[seq(1, nrow(SS), stap), ]
plotlijstII = II[seq(1, nrow(SS), stap), ]
plotlijstRR = RR[seq(1, nrow(SS), stap), ]

matplot(x = plotx, y = plotlijstSS,
        type = "l", xlab = "Time", ylab = "Susceptibles",
            xlim=c(0, 100),
        ylim=c(0, 1), main = "Ensemble␣S", lwd = 1, lty = 1,
             bty = "l",
        col = 2:4)
matplot(x = plotx, y = plotlijstII,
        type = "l", xlab = "Time", ylab = "Infected", xlim=c
            (0, 100),
        ylim=c(0, 1), main = "Ensemble␣I", lwd = 1, lty = 1,
             bty = "l",
        col = 2:4)
matplot(x = plotx, y = plotlijstRR,
        type = "l", xlab = "Time", ylab = "Recovered", xlim=
            c(0, 100),
        ylim=c(0, 1), main = "Ensemble␣R", lwd = 1, lty = 1,
             bty = "l",
        col = 2:4)



#gemiddelden berekenen met gemfun functie
gemlijstSIR <- gemfun(SS, II, RR)
plotgemSIR = gemlijstSIR[seq(1, nrow(gemlijstSIR), stap), ]

#plot gemiddelden
xrow <- seq(0, 100, by = 1)
matplot(x = xrow, y = plotgemSIR, type = "l",
        xlab = "Time", ylab = "Susceptible␣and␣Recovered",
            xlim=c(0, 100),
        ylim=c(0, 1), main = "Gemiddelden␣S,␣I␣en␣R", lwd =
            1, lty = 1,
        bty = "l", col = 2:4)
legend(60, 0.7, c("Susceptible", "Infected", "Recovered"),
    pch = 1, col = 2:4,
       bty = "n")



#covarmatrix en sd's berekenen met covarmat functie
covarmatrix <- covarmatfun(SS, II, RR, gemlijstSIR, 1000)

sdSlist <- matrix(data = 0, nrow = 1001, ncol=1)
sdIlist <- matrix(data = 0, nrow = 1001, ncol=1)
sdRlist <- matrix(data = 0, nrow = 1001, ncol=1)

for(t in 1:1000){
```

```
  #standaard deviaties van SIR berekenen
  sdS = sqrt(covarmatrix[[t]][1,1])
  sdI = sqrt(covarmatrix[[t]][2,2])
  sdR = sqrt(covarmatrix[[t]][3,3])

  #alles van s samenvoegen voor plot later
  sdSlist[[t]] <- sdS
  sdIlist[[t]] <- sdI
  sdRlist[[t]] <- sdR
}


#plot sd's
#Varianties bij gemiddelde optellen en aftrekken
#moet dit ten opzichte van gemiddelde zoals ik hieronder doe
    ??? JA
sdSmin <- gemlijstSIR[,1] - sdSlist
sdSplus <- gemlijstSIR[,1] + sdSlist
sdImin <- gemlijstSIR[,2] - sdIlist
sdIplus <- gemlijstSIR[,2] + sdIlist
sdRmin <- gemlijstSIR[,3] - sdRlist
sdRplus <- gemlijstSIR[,3] + sdRlist

#per dag laten lopen en gem met var plotten
plotsdSmin <- sdSmin[seq(1, nrow(sdSmin), stap), ]
plotsdSplus <- sdSplus[seq(1, nrow(sdSplus), stap), ]
plotsdImin <- sdImin[seq(1, nrow(sdImin), stap), ]
plotsdIplus <- sdIplus[seq(1, nrow(sdIplus), stap), ]
plotsdRmin <- sdRmin[seq(1, nrow(sdRmin), stap), ]
plotsdRplus <- sdRplus[seq(1, nrow(sdRplus), stap), ]

plot(x = plotx, y = plotsdSplus,
     type = "l", xlab = "Time", ylab = "Susceptibles", xlim=
         c(0, 100),
     ylim=c(-0.5, 1.5), main = "Sd␣S", lwd = 1, lty = 1, bty
          = "l",
     col = 'pink')
lines(x = plotx, y = plotsdSmin, col='orange')
lines(x = plotx, y = plotgemSIR[,1], col='red')
###
plot(x = plotx, y = plotsdIplus,
     type = "l", xlab = "Time", ylab = "Infected", xlim=c(0,
          100),
     ylim=c(-0.5, 1.5), main = "Sd␣I", lwd = 1, lty = 1, bty
          = "l",
     col = 'pink')
lines(x = plotx, y = plotsdImin, col='orange')
lines(x = plotx, y = plotgemSIR[,2], col='green')
###
plot(x = plotx, y = plotsdRplus,
     type = "l", xlab = "Time", ylab = "Recovered", xlim=c(0
         , 100),
     ylim=c(-0.5, 1.5), main = "Sd␣R", lwd = 1, lty = 1, bty
          = "l",
```

```
      col = 'pink')
lines(x = plotx, y = plotsdRmin, col='orange')
lines(x = plotx, y = plotgemSIR[,3], col='blue')
```

A.4 Uitbreidingen SIR model

Listing A.7: Het SEIQHRF model

```
source("_icm.mod.status.seiqhrf.R")
source("_icm.control.seiqhrf.R")
source("_icm.icm.seiqhrf.R")
source("_icm.mod.init.seiqhrf.R")
source("_icm.mod.vital.seiqhrf.R")
source("_icm.saveout.seiqhrf.R")
source("_icm.utils.seiqhrf.R")

# function to set-up and run the baseline simulations
simulate <- function(# control.icm params
  type = "SEIQHRF",
  nsteps = 366,
  nsims = 8,
  ncores = 4,
  prog.rand = FALSE,
  rec.rand = FALSE,
  fat.rand = TRUE,
  quar.rand = FALSE,
  hosp.rand = FALSE,
  disch.rand = TRUE,
  infection.FUN = infection.seiqhrf.icm,
  recovery.FUN = progress.seiqhrf.icm,
  departures.FUN = departures.seiqhrf.icm,
  arrivals.FUN = arrivals.icm,
  get_prev.FUN = get_prev.seiqhrf.icm,
  # init.icm params
  s.num = 9997, #S0
  e.num=0,
  i.num = 3, #I0
  q.num=0,
  h.num=0,
  r.num = 0, #R0
  f.num = 0,
  # param.icm params
  inf.prob.e = 0.02,
  act.rate.e = 10,
  inf.prob.i = 0.05,
  act.rate.i = 10,
  inf.prob.q = 0.02,
  act.rate.q = 2.5,
  quar.rate = 1/30,
  hosp.rate = 1/100,
  disch.rate = 1/15,
  prog.rate = 1/10,
  prog.dist.scale = 5,
  prog.dist.shape = 1.5,
```

```
  rec.rate = 1/20,
  rec.dist.scale = 35,
  rec.dist.shape = 1.5,
  fat.rate.base = 1/50,
  hosp.cap = 40,
  fat.rate.overcap = 1/25,
  fat.tcoeff = 0.5,
  vital = TRUE,
  a.rate = (10.5/365)/1000,
  a.prop.e = 0.01,
  a.prop.i = 0.001,
  a.prop.q = 0.01,
  ds.rate = (7/365)/1000,
  de.rate = (7/365)/1000,
  di.rate = (7/365)/1000,
  dq.rate = (7/365)/1000,
  dh.rate = (20/365)/1000,
  dr.rate = (7/365)/1000,
  out="mean"
) {

  control <- control.icm(type = type,
                         nsteps = nsteps,
                         nsims = nsims,
                         ncores = ncores,
                         prog.rand = prog.rand,
                         rec.rand = rec.rand,
                         infection.FUN = infection.FUN,
                         recovery.FUN = recovery.FUN,
                         arrivals.FUN = arrivals.FUN,
                         departures.FUN = departures.FUN,
                         get_prev.FUN = get_prev.FUN)

  init <- init.icm(s.num = s.num,
                   e.num = e.num,
                   i.num = i.num,
                   q.num = q.num,
                   h.num = h.num,
                   r.num = r.num,
                   f.num = f.num)

  param <-  param.icm(inf.prob.e = inf.prob.e,
                      act.rate.e = act.rate.e,
                      inf.prob.i = inf.prob.i,
                      act.rate.i = act.rate.i,
                      inf.prob.q = inf.prob.q,
                      act.rate.q = act.rate.q,
                      quar.rate = quar.rate,
                      hosp.rate = hosp.rate,
                      disch.rate = disch.rate,
                      prog.rate = prog.rate,
                      prog.dist.scale = prog.dist.scale,
                      prog.dist.shape = prog.dist.shape,
                      rec.rate = rec.rate,
```

```
                        rec.dist.scale = rec.dist.scale,
                        rec.dist.shape = rec.dist.shape,
                        fat.rate.base = fat.rate.base,
                        hosp.cap = hosp.cap,
                        fat.rate.overcap = fat.rate.overcap,
                        fat.tcoeff = fat.tcoeff,
                        vital = vital,
                        a.rate = a.rate,
                        a.prop.e = a.prop.e,
                        a.prop.i = a.prop.i,
                        a.prop.q = a.prop.q,
                        ds.rate = ds.rate,
                        de.rate = de.rate,
                        di.rate = di.rate,
                        dq.rate = dq.rate,
                        dh.rate = dh.rate,
                        dr.rate = dr.rate)

  sim <- icm.seiqhrf(param, init, control)
  sim_df <- as.data.frame(sim, out=out)

  return(list(sim=sim, df=sim_df))
}

baseline_sim <- simulate(ncores = 4)


compcols <- c(s.num = "tomato3", e.num = "orange",
              i.num = "darkolivegreen3", q.num = "purple3",
              h.num = "thistle", r.num = "steelblue2",
              f.num = "black")
compcols2 <- c(e.num = "orange", i.num = "darkolivegreen3",
              q.num = "purple3", h.num = "thistle",
              f.num = "black")

plotS <- baseline_sim[["df"]][2]
plotE <- baseline_sim[["df"]][17]
plotI <- baseline_sim[["df"]][3]
plotQ <- baseline_sim[["df"]][19]
plotH <- baseline_sim[["df"]][20]
plotR <- baseline_sim[["df"]][18]
plotF <- baseline_sim[["df"]][21]


plotbs <- seq(1, 366, by = 1)

#plot met uitbreidingen
matplot(x=plotbs, y=plotS, type = 'l', xlab = "Time",
        ylab = "Persons", xlim=c(0, 120), ylim=c(0, 10000),
        main = "Basissimulatie␣SEIQHRF␣model",
        col = "tomato3", lwd = 1, lty = 1, bty = "l")
matlines(x=plotbs, y=plotE, lwd = 1, col="orange")
matlines(x=plotbs, y=plotI, lwd = 1, col="darkolivegreen3")
matlines(x=plotbs, y=plotQ, lwd = 1, col="purple3")
```

```
matlines(x=plotbs, y=plotH, lwd = 1, col="thistle")
matlines(x=plotbs, y=plotR, lwd = 1, col="steelblue2")
matlines(x=plotbs, y=plotF, lwd = 1, col="black")


legend(55, 9000, c("Susceptible", "Infected␣&␣asymptomatic",
                   "Infected␣&␣besmettelijk", "Thuis-
                      Quarantaine",
                   "Behoefte␣aan␣ziekenhuisopname",
                   "Recovered", "Sterfte"),
       pch = 1, col = compcols, bty = "n")



#plot ingezoomd op onderste deel

plotE2 <- baseline_sim[["df"]][17]
plotI2 <- baseline_sim[["df"]][3]
plotQ2 <- baseline_sim[["df"]][19]
plotH2 <- baseline_sim[["df"]][20]
plotF2 <- baseline_sim[["df"]][21]

matplot(x=plotbs, y=plotE2, type = 'l', xlab = "Time",
        ylab = "Persons", xlim=c(10, 100), ylim=c(0, 3600),
        main = "Basissimulatie␣SEIQHRF␣model", lwd = 1,
        lty = 1, bty = "l", col = "orange")
matlines(x=plotbs, y=plotI2, lwd = 1, col="darkolivegreen3")
matlines(x=plotbs, y=plotQ2, lwd = 1, col="purple3")
matlines(x=plotbs, y=plotH2, lwd = 1, col="thistle")
matlines(x=plotbs, y=plotF2, lwd = 1, col="black")


legend(52, 3000, c("Infected␣&␣asymptomatic",
                   "Infected␣&␣infectious", "Thuis-
                      Quarantaine",
                   "Behoefte␣aan␣ziekenhuisopname",
                   "Sterfte"),
       pch = 1, col = compcols2, bty = "n")


#Meer quarantaine
quar_rate_ramp <- function(t) {
  ifelse(t < 15, 0.0333, ifelse(t <= 30, 0.0333 + (t - 15) *
                                  (0.5 - 0.0333)/15, 0.5))
}

ramp_quar_rate_sim <- simulate(quar.rate =
                                 quar_rate_ramp(1:366))

plotEq <- ramp_quar_rate_sim[["df"]][17]
plotIq <- ramp_quar_rate_sim[["df"]][3]
plotQq <- ramp_quar_rate_sim[["df"]][19]
plotHq <- ramp_quar_rate_sim[["df"]][20]
plotFq <- ramp_quar_rate_sim[["df"]][21]
```

```
matplot(x=plotbs, y=plotEq, type = 'l', xlab = "Time",
        ylab = "Persons", xlim=c(10, 100), ylim=c(0, 3600),
        main = "Meer mensen in thuis-quarantaine",
        col = "orange", lwd = 1, lty = 1, bty = "l")
matlines(x=plotbs, y=plotIq, lwd = 1, col="darkolivegreen3")
matlines(x=plotbs, y=plotQq, lwd = 1, col="purple3")
matlines(x=plotbs, y=plotHq, lwd = 1, col="thistle")
matlines(x=plotbs, y=plotFq, lwd = 1, col="black")

legend(48, 3500, c("Infected & asymptomatic",
                   "Infected & infectious", "Thuis-
                       Quarantaine",
                   "Behoefte aan ziekenhuisopname",
                   "Sterfte"),
       pch = 1, col = compcols2, bty = "n")



#Social distancing
social_distancing_day15_ramp <- function(t) {
  ifelse(t < 15, 10, ifelse(t <= 30, 10 - (t - 15) *
                              (10 - 5)/15, 5))
}
t15_social_distancing_sim <- simulate(act.rate.i =
          social_distancing_day15_ramp(1:366),
          act.rate.e = social_distancing_day15_ramp(1:366))


social_distancing_day30_ramp <- function(t) {
  ifelse(t < 30, 10, ifelse(t <= 45, 10 - (t - 30) *
                              (10 - 5)/15, 5))
}
t30_social_distancing_sim <- simulate(act.rate.i =
          social_distancing_day30_ramp(1:366),
          act.rate.e = social_distancing_day30_ramp(1:366))


plotEs <- t15_social_distancing_sim[["df"]][17]
plotIs <- t15_social_distancing_sim[["df"]][3]
plotQs <- t15_social_distancing_sim[["df"]][19]
plotHs <- t15_social_distancing_sim[["df"]][20]
plotFs <- t15_social_distancing_sim[["df"]][21]


matplot(x=plotbs, y=plotEs, type = 'l', xlab = "Time",
        ylab = "Persons", xlim=c(10, 100), ylim=c(0, 3600),
        main = "Social distancing op t=15", col = "orange",
        lwd = 1, lty = 1, bty = "l")
matlines(x=plotbs, y=plotIs, lwd = 1, col="darkolivegreen3")
matlines(x=plotbs, y=plotQs, lwd = 1, col="purple3")
matlines(x=plotbs, y=plotHs, lwd = 1, col="thistle")
```

```
matlines(x=plotbs, y=plotFs, lwd = 1, col="black")

legend(48, 3500, c("Infected␣&␣asymptomatic",
                   "Infected␣&␣infectious", "Thuis-
                      Quarantaine",
                   "Behoefte␣aan␣ziekenhuisopname",
                   "Sterfte"),
       pch = 1, col = compcols2, bty = "n")



plotEss <- t30_social_distancing_sim[["df"]][17]
plotIss <- t30_social_distancing_sim[["df"]][3]
plotQss <- t30_social_distancing_sim[["df"]][19]
plotHss <- t30_social_distancing_sim[["df"]][20]
plotFss <- t30_social_distancing_sim[["df"]][21]


matplot(x=plotbs, y=plotEss, type = 'l', xlab = "Time",
        ylab = "Persons", xlim=c(10, 100), ylim=c(0, 3600),
        main = "Social␣distancing␣op␣t=30", col = "orange",
        lwd = 1, lty = 1, bty = "l")
matlines(x=plotbs, y=plotIss, lwd = 1, col="darkolivegreen3"
    )
matlines(x=plotbs, y=plotQss, lwd = 1, col="purple3")
matlines(x=plotbs, y=plotHss, lwd = 1, col="thistle")
matlines(x=plotbs, y=plotFss, lwd = 1, col="black")

legend(48, 3500, c("Infected␣&␣asymptomatic",
                   "Infected␣&␣infectious", "Thuis-
                      Quarantaine",
                   "Behoefte␣aan␣ziekenhuisopname",
                   "Sterfte"),
       pch = 1, col = compcols2, bty = "n")
```

B Tweede appendix; Bijlagen

B.1 Data totaal aantal geïnfecteerden I per dag in Nederland

Totaal aantal zieken	Als fractie van de bevolking
117	6,71449E-06
221	1,26829E-05
395	2,26686E-05
578	3,31707E-05
754	4,32712E-05
1032	5,92253E-05
1324	7,59828E-05
1670	9,58393E-05
2079	0,000119311
2613	0,000149957
3133	0,000179799
3602	0,000206714
3973	0,000228006
4601	0,000264046
5277	0,000302841
6018	0,000345366
6898	0,000395868
7711	0,000442525
8406	0,00048241
8756	0,000502496
8964	0,000514433
9410	0,000540029
9948	0,000570904
10163	0,000583242
10215	0,000586227
10420	0,000597991
10200	0,000585366
9818	0,000563443
9683	0,000555696
10012	0,000574577
10502	0,000602697
10799	0,000619742
10890	0,000624964
10828	0,000621406
10792	0,00061934
10302	0,00059122
10411	0,000597475
10869	0,000623759
11040	0,000633572

10893	0,000625136
10308	0,000591564
9721	0,000557877
9255	0,000531133
9178	0,000526714
9116	0,000523156
9037	0,000518623
8631	0,000495323
7796	0,000447403
6827	0,000391793
6147	0,000352769
5911	0,000339225
5657	0,000324648
5394	0,000309555
4842	0,000277877
4235	0,000243042
3897	0,000223644
3474	0,000199369
3529	0,000202525
3677	0,000211019
3580	0,000205452
3311	0,000190014
2997	0,000171994
2748	0,000157704
2640	0,000151506
2711	0,000155581
2594	0,000148867
2551	0,000146399
2221	0,000127461
2048	0,000117532
1867	0,000107145
1820	0,000104448
1912	0,000109727
1904	0,000109268
1853	0,000106341
1755	0,000100717
1764	0,000101234
1708	9,80201E-05
1773	0,00010175
1809	0,000103816
1877	0,000107719
1810	0,000103874

1742	9,99713E-05
1657	9,50933E-05
1583	9,08465E-05
1497	8,5911E-05
1497	8,5911E-05
1574	9,033E-05
1567	8,99283E-05
1624	9,31994E-05
1613	9,25681E-05
1646	9,4462E-05
1645	9,44046E-05
1706	9,79053E-05
1814	0,000104103
1907	0,00010944
1841	0,000105653
1796	0,00010307
1752	0,000100545
1630	9,35438E-05
1580	9,06743E-05
1523	8,74032E-05
1415	8,12052E-05
1342	7,70158E-05
1197	6,86944E-05
1082	6,20947E-05
1021	5,8594E-05
966	5,54376E-05
918	5,26829E-05
870	4,99283E-05

Figuur B.1: Metingen I

B.2 Aannames $SEIQHRF$ model

DiagramRef	Parameter	Default	Explanation
	type	SEIQHRF	Type of model: SI, SIR, SIS, SEIR, SEIQHR and SEIQHRF available, but only SEIQHRF is likely to work in the current version of the code.
	nsteps	366	Number of days for simulation. Note that day 1 is for initialisation, day 2 is the first day of the simulation, hence default of 366 for 1 year.
	nsims	10	Number of simulations to run and then average.
	ncores	10	Number of CPU cores to use for parallel execution.
b	prog.rand	FALSE	Method for progression from E compartment to I. If TRUE, random binomial draws at `prog.rate`, if FALSE, random draws from a Weibull distribution (yes, I know it should be a discrete Weibull distribution but it makes little difference and speed of computation matters), with parameters `prog.dist.scale` and `prog.dist.shape`
d,g,h	rec.rand	FALSE	Method for recovery transition from I, Q or H to R. If TRUE, random binomial draws at `prog.rate`, if FALSE, random draws from a random draws from a Weibull distribution, with parameters `rec.dist.scale` and `rec.dist.shape`
f	fat.rand	FALSE	Method for case fatality transition from H to F. If TRUE, random binomial draws at `fat.rate.base`, if FALSE, random sample with a sample fraction also given by `fat.rate.base`. However, if the current number of patients in the H (needs hospitalisation) compartment is above a hospital capacity level specified by `hosp.cap`, then the fatality rate is the mean of the base fatality rate weighted by the hospital capacity, plus a higher rate, specified by `fat.rate.overcap`, weighted by the balance of those requiring hospitalisation (but who can't be accommodated). By setting `fat.rate.overcap` higher, the effect of exceeding the capacity of the health care system can be simulated. There is also a coefficient `fat.tcoeff` for the fatality rates that increases them as a linear function of the number of days the individual has been in the H compartment. Use of the co-efficient better approximates the trapezoid survival time distribution typical of ICU patients.
c	quar.rand	FALSE	Method for self-isolation transition from I to Q. If TRUE, random binomial draws at `quar.rate`, if FALSE, random sample with a sample fraction also given by `quar.rate.

e,i	hosp.rand	FALSE	Method for transition from I or Q to H -- that is, from infectious or from self-isolated to requiring hospitalisation. If TRUE, random binomial draws at `hosp.rate`, if FALSE, random sample with a sample fraction also given by `hosp.rate.
e,i	disch.rand	FALSE	Method for transition from H to R -- that is, from requiring hospitalisation to recovered. If TRUE, random binomial draws at `disch.rate`, if FALSE, random sample with a sample fraction also given by `disch.rate`. Note that the only way out of the **H** compartment is recovery or death.
	infection.FUN	infection.seiqhrf.icm	No, being infected with SARS-CoV2 is **not** fun. Rather this is the the name of the function to implement infection processes. Use the default.
	departures.FUN	departures.seiqhrf.icm	Handles background demographics, specifically departures (deaths not due to the virus, and emigration). Use the default.
	arrivals.FUN	arrivals.icm	Handles background demographics, specifically arrivals (births and immigration). Uses the original EpiModel code currently. A replacement that implements modelling the arrival of infected individuals is under development -- but for now, all arrivals go into the **S** compartment.
	get_prev.FUN	get_prev.seiqhrf.icm	Utility function that collects prevalence and transition time data from each run and stores it away in the simulation result object. Use the default.
	s.num	9997	Initial number of **S* compartment individuals in the simulated population. An overall population of 10,000 is a good compromise. A set of models will still take several minutes or more to run, in parallel.
	e.num	0	Initial number of **E** compartment individuals in the simulated population.
	i.num	3	Initial number of **I** compartment individuals in the simulated population.
	q.num	0	Initial number of **Q** compartment individuals in the simulated population.
	h.num	0	Initial number of **H** compartment individuals in the simulated population.
	r.num	0	Initial number of **R** compartment individuals in the simulated population.
	f.num	0	Initial number of **F** compartment individuals in the simulated population.
x	act.rate.i	10	The number of exposure events (*acts*) between infectious individuals in the **I** compartment and susceptible individuals in the **S** compartment, per day. It's stochastic, so the rate is an average, some individuals may have more or less. Note that not every exposure event results in infection - that is governed by the `inf.prob.i` parameters (see below). Reducing `act.rate.i` is equivalent to increasing social distancing by people in the **I** compartment.
x	inf.prob.i	0.05	Probability of passing on infection at each exposure event for interactions between infectious people in the **I** compartment and susceptibles in **S**. Reducing `inf.prob.i` is equivalent to increasing hygiene measures, such as not putting hands in eyes, nose or moth, use of hand sanitisers, wearing masks by the infected, and so on.

y	act.rate.e	10	The number of exposure events (*acts*) between infectious individuals in the **E** compartment and susceptible individuals in the **S** compartment, per day. Otherwise as for `act.rate.i`.
y	inf.prob.e	0.02	Probability of passing on infection at each exposure event for interactions between infectious people in the **E** compartment and susceptibles in **S**. Note the default is lower than for `inf.prob.i` reflecting the reduced infectivity of infected but asymptomatic people (the **E** compartment). Otherwise as for `inf.exp.i`.
z	act.rate.q	2.5	The number of exposure events (*acts*) between infectious individuals in the **Q** compartment (isolated, self or otherwise) and susceptible individuals in the **S** compartment, per day. Note the much lower rate than for the **I** and **E** compartments, reflecting the much greater degree of social isolation for someone in (self-)isolation. The exposure event rate is not zero for this group, just much less. Otherwise as for `act.rate.i`.
z	inf.prob.q	0.02	Probability of passing on infection at each exposure event for interactions between infectious people in the **Q** compartment and susceptibles in **S**. Note the default is lower than for `inf.prob.i` reflecting the greater care that self-isolated individuals will, on average, take regarding hygiene measures, such as wearing masks, to limit spread to others. Otherwise as for `inf.exp.i`.
c	quar.rate	1/30	Rate per day at which symptomatic (or tested positive), infected **I** compartment people enter self-isolation (**Q** compartment). Asymptomatic **E** compartment people can't enter self-isolation because they don't yet know they are infected. Default is a low rate reflecting low community awareness or compliance with self-isolation requirements or practices, but this can be tweaked when exploring scenarios.
e,i	hosp.rate	1/100	Rate **per day** at which symptomatic (or tested positive), infected **I** compartment people or self-isolated **Q** compartment people enter the state of requiring hospital care -- that is, become serious cases. A default rate of 1% per day with an average illness duration of about 10 days means a bit less than 10% of cases will require hospitalisation, which seems about right (but can be tweaked, of course).
g	disch.rate	1/15	Rate per day at which people needing hospitalisation recover.
b	prog.rate	1/10	Rate per day at which people who are infected but asymptomatic (**E** compartment) progress to becoming symptomatic (or test-positive), the **I** compartment. See `prog.rand` above for more details.
b	prog.dist.scale	5	Scale parameter for Weibull distribution for progression, see `prog.rand` for details.
b	prog.dist.shape	1.5	Shape parameter for Weibull distribution for progression, see `prog.rand` for details. Read up on the Weibull distribution before changing the default.

d	rec.rate	1/20	Rate per day at which people who are infected and symptomatic (**I** compartment) recover, thus entering the **R** compartment. See `rec.rand` above for more details.
d	rec.dist.scale	35	Scale parameter for Weibull distribution for recovery, see `rec.rand` for details.
d	rec.dist.shape	1.5	Shape parameter for Weibull distribution for recovery, see `rec.rand` for details. Read up on the Weibull distribution before changing the default.
f	fat.rate.base	1/50	Baseline mortality rate per day for people needing hospitalisation (deaths due to the virus). See `fat.rand` for more details.
f	hosp.cap	40	Number of available hospital beds for the modelled population. See `fat.rand` for more details.
f	fat.rate.overcap	1/25	Mortality rate per day for people needing hospitalisation but who can't get into hospital due to the hospitals being full (see `hosp.cap` and `fat.rand`). The default rate is twice that for those who do get into hospital.
f	fat.tcoeff	0.5	Time co-efficient for increasing mortality rate as time in the **H** compartment increases for each individual in it. See `fat.rand` for details.
	vital	TRUE	Enables demographics, that is, arrivals and departures, to and from the simulated population.
	a.rate	(10.5/365)/1000	Background demographic arrival rate. Currently all arrivals go into the **S** compartment, the default is approximately the daily birth rate for Australia. Will be extended to cover immigration in future versions.
	ds.rate, de.rate, de.rate, dq.rate, dh.rate, dr.rate	various rates	Background demographic departure (death not due to virus) rates. Defaults based on Australian crude death rates. Can be used to model emigration as well as deaths.
	out	mean	Summary function for the simulation runs. median is also available, or percentiles, see the `EpiModel` documentation.

Figuur B.2: Aannames $SEIQHRF$ model

Literatuur

Rijksinstituut voor Volksgezondheid en Milieu,
https://www.rivm.nl/coronavirus-covid-19/actueel

Rstudio,
https://rstudio.com/

Het SIR model,
https://www.maa.org/press/periodicals/loci/joma/the-sir-model-for-spread-of-disease-the-differential-equation-model

Data Coronavirus,
https://allecijfers.nl/nieuws/statistieken-over-het-corona-virus-en-covid19/#Corona_kerncijfers

Bevolingsteller Nederland,
https://www.cbs.nl/nl-nl/visualisaties/bevolkingsteller

Data coronavirus,
https://www.zorgvoorbeter.nl/nieuws/corona-veelgestelde-vragen

Stabiliteit en evenwichtspunten,
http://homepage.tudelft.nl/11r49/onderw0102/diffvglwb/college/week41.pdf

Jacobi-matrix,
https://www.imedpub.com/articles/study-of-simple-sir-epidemic-model.pdf

Tang, B., Bragazzi, N.L., Qian Li, Q., Tang, S., Xiao, Y. & Wu, J., (2019). *An updated estimation of the risk of transmission of the novel coronavirus*,
https://www.sciencedirect.com/science/article/pii/S246804272030004X

Blauw, S., (2020). *Dit is het belangrijkste getal van deze epidemie en hoe wordt het berekend*,
https://decorrespondent.nl/11149/dit-is-het-belangrijkste-getal-van-deze-epidemie-hoe-wordt-het-berekend/162248208386-a0a5fdf8

Corona Dashboard RIVM,
https://coronadashboard.rijksoverheid.nl

De Euler methode,
https://en.wikipedia.org/wiki/Eulermethod

Evensen, Geir (2009). *Data Assimilation. The Ensemble Kalman Filter.*

De Gaussische verdeling,
https://nl.wikipedia.org/wiki/Normaleverdeling

Katzfuss, M., Stroud, J.R. & Wiklec, C.K., (2016). *Understanding the Ensemble Kalman Filter.*
https://www.math.umd.edu/slud/RITF17/enkf-tutorial.pdf

Mandel, J., (2009). *A Brief Tutorial on the Ensemble Kalman Filter.*

De stelling van Bayes,
https://nl.wikipedia.org/wiki/TheoremavanBayes

Het ziektecompartimentenmodel,
https://nl.wikipedia.org/wiki/Ziektecompartimentenmodel

Churches, T., (2020). *Modelling the effects of public health interventions on COVID-19 transmission using R - part 2*,
https://timchurches.github.io/blog/posts/2020-03-18-modelling-the-effects-of-public-health-interventions-on-covid-19-transmission-part-2/