A.C. van Oorschot

Virtuele Wachtrijen in de Efteling

Bachelorscriptie

2 oktober 2025

Scriptiebegeleider: prof.dr. F.M. Spieksma

Universiteit Leiden
Mathematisch Instituut

Inhoudsopgave

1 Inleiding **4**

2 Basismodel **6**
2.1 Attractie 6
2.2 Model 7

3 Data **9**

4 Parameters **13**

5 Basismodel Simulatie **16**
5.1 Simulatie 16
5.2 Resultaten 17

6 Droomvlucht Model **18**
6.1 Virtuele Wachtrij 18
6.2 Model 19

7 Droomvlucht Parameters **21**
7.1 Methode 0 22
7.2 Methode 1 22
7.3 Methode 2 22
7.4 Overige Parameters 23

8 Droomvlucht Simulatie **24**
8.1 Simulatie 24
8.2 Resultaten 26

9 Danse Macabre Model **29**
9.1 Attractie 29
9.2 Model 31

10 Danse Macabre Data **33**

11 Danse Macabre Parameters **36**
11.1 Eigen Metingen 36
11.2 No-shows 38
11.3 Overige Parameters 39

12 Danse Macabre Simulatie **41**
12.1 Simulatie 41
12.2 Resultaten 42

13 Test **46**

14 Discussie **48**

14.1 Data 48
14.2 Parameters 48
14.3 Model 49
14.4 Vervolgonderzoek 50

A Code Basismodel **55**

B Code Droomvlucht Methode 1 **58**

C Code Droomvlucht Methode 2 **63**

D Code Danse Macabre Methode 1 **68**

E Code Danse Macabre Methode 2 **72**

1 Inleiding

De Efteling is het meest bezochte attractiepark van Nederland [14]. Het park streeft ernaar om in 2030 geen wachttijden langer dan 20 minuten te hebben [9]. Om dit te bereiken, experimenteert het park met verschillende methodes om de wachttijden te verkorten. Eén van deze methodes is het laten zien van wachttijdsvoorspellingen bij iedere attractie [6] in de officiële Efteling-app. Hiermee krijgen bezoekers in een grafiek een schatting te zien van de wachttijden op die dag. Dit moet ertoe leiden dat men op piekmomenten voor een andere attractie kiest, waardoor de drukte meer verspreid wordt.

Een andere methode waarmee de Efteling experimenteert is het gebruik van een virtuele wachtrij. Bezoekers in een virtuele wachtrij krijgen een tijdslot toegewezen waarin ze worden toegelaten tot de attractie. Het grootste deel van de wachttijd kan hierdoor buiten de fysieke wachtrij doorgebracht worden, bijvoorbeeld bij een andere attractie of op een terras.

Eén van deze experimenten heeft in de zomer van 2024 plaatsgevonden bij de darkride Droomvlucht. In tegenstelling tot voorgaande testen, waren er tegelijkertijd een virtuele wachtrij en een reguliere fysieke wachtrij actief [11]. Hierdoor konden de bezoekers kiezen welke wachtrij ze willen gebruiken.

Op 22 april 2025 opende de Efteling een virtuele wachtrij bij haar nieuwste attractie, Danse Macabre [18]. Hierbij wordt gebruik gemaakt van hetzelfde systeem als eerder bij Droomvlucht, met de keuze tussen een virtuele en een reguliere rij. Hier gaat het echter om een permanente implementatie, en niet om een tijdelijk experiment.

De combinatie van een reguliere wachtrij en een virtuele wachtrij maakt dit een interessant systeem om te bestuderen. Het doel van deze scriptie is dan ook om de wachtrijsystemen van Droomvlucht en Danse Macabre te modelleren en vervolgens te simuleren. We doen dit door eerst een basismodel op te stellen voor een reguliere wachtrij bij de attractie Droomvlucht, zonder virtuele rij. We kunnen dit model testen met behulp van een simulatie, en maken hierbij gebruik van historische data. Door de resultaten te vergelijken met de historische data kunnen we beoordelen hoe realistisch het model is. Vervolgens breiden we dit model uit naar een model dat zowel de reguliere als de virtuele wachtrij bij Droomvlucht omvat. Ook dit model simuleren we aan de hand van historische data. Tot slot passen we het tweede model aan naar de situatie bij Danse Macabre, en voeren we ook bij dit laatste model een simulatie uit aan de hand van historische data. Omdat deze virtuele wachtrij op het moment van schrijven ook actief is, testen we dit model ook in de praktijk.

Wanneer de simulaties goed werken, kunnen deze dienen als een soortgelijke wachttijdsvoorspelling als die de Efteling al laat zien in haar app. Voor virtuele wachtrijen zijn deze grafieken namelijk nog niet beschikbaar.

Om aan data en informatie over de werking van de wachtrijen te komen, hebben we geprobeerd om hierover contact te leggen met de Efteling. Het park gaf

echter aan dat het te veel verzoeken van studenten krijgt om hieraan te kunnen meewerken. Hierdoor moest ik zelf de werking van de virtuele wachtrijen reconstrueren, hetgeen veel tijd heeft gekost. Ook besloeg een groot deel van dit onderzoek het verzamelen en ordenen van data. Deze bestaan uit historische wachttijden en gegevens afkomstig van verschillende websites, in combinatie met eigen metingen die ik ter plekke heb verricht. Ik wil medestudent Liam Elderbroek bedanken voor het schrijven van een Python script en een Windows Powershell script, waarmee ik de data overzichtelijk kon samenvoegen in een Excel bestand. Ik wil ook Efteling-podcast Kleine Boodschap bedanken voor het beantwoorden van mijn vragen over de virtuele rij bij Droomvlucht. Tenslotte wil ik Floske Spieksma bedanken voor de begeleiding bij het schrijven van deze scriptie.

2 Basismodel

We zullen eerst een simpel wachtrijmodel opstellen waarbij we alleen de reguliere wachtrij bekijken. Dit kunnen we gebruiken als basis voor een model dat zowel de reguliere als virtuele wachtrij omvat. Omdat bij de aanvang van dit project nog niet bekend was dat er een virtuele wachtrij bij Danse Macabre zou komen, is ervoor gekozen om dit model te baseren op de situatie bij Droomvlucht. Dit basismodel kan indien gewenst eenvoudig aangepast worden naar attracties met hetzelfde bedieningsproces, zoals Fata Morgana, door andere waardes voor de parameters te kiezen.

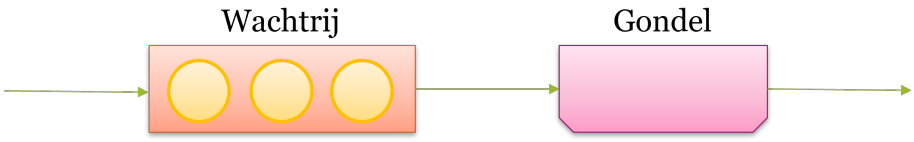
2.1 Attractie

De attractie Droomvlucht is een darkride voor het hele gezin, en werd geopend in 1993. In een hangende gondel wordt men door een sprookjeswereld met elfjes en trollen gevoerd. Bij het betreden van de attractie belandt men in een standaard, meanderende wachtrij. Aan het einde van de wachtrij stappen bezoekers op een loopband, waarna ze in één van de voertuigen kunnen stappen. Er zijn 28 voertuigen met ieder twee gondels, en iedere gondel biedt plaats aan maximaal drie volwassenen of vier kinderen [27]. Doordat de voertuigen op dezelfde snelheid als de loopband bewegen, kunnen bezoekers instappen zonder dat het voertuig stilgezet moet worden. Het uitstapproces werkt op dezelfde manier. Het instap- en uitstapstation bevinden zich tegenover elkaar, waardoor de gondels via een bocht meteen weer in het instapstation belanden. We hebben gemeten dat een gondel 6 minuten en 47 seconden over één ronde doet.

Mensen die niet zelfstandig kunnen lopen, kunnen Droomvlucht niet bezoeken. Voor hen is een aparte ervaring gerealiseerd, waarbij men met een 3d-bril en effecten de attractie kan ervaren. De attractie wordt dus niet stilgezet voor deze mensen.

De wachttijd wordt weergegeven op een bord bij de ingang van de attractie, en is ook te vinden in de Efteling-app. Deze wachttijd is een inschatting van de tijd dat men moet wachten als men nu aansluit in de rij, en wordt afgerond op vijftallen minuten. Het is onduidelijk of deze wachttijd door een medewerker wordt ingeschat of automatisch bepaald wordt, bijvoorbeeld met behulp van een sensor.

Van 1 juli 2024 tot en met 31 augustus 2024 vond een experiment met een virtuele wachtrij plaats bij Droomvlucht. Bezoekers konden kiezen of ze in de fysieke, reguliere rij wilden gaan staan, of wilden aansluiten in de virtuele rij. Men kon in de virtuele rij aansluiten door in de Efteling-app de attractie te selecteren. Zowel de virtuele als de reguliere wachttijd zijn in dit scherm zichtbaar. Door op de knop “Aansluiten in de virtuele wachtrij” te klikken en de grootte van het gezelschap in te voeren, krijgt men een tijdslot te zien. Tot dit tijdslot is aangebroken, kan het gezelschap bijvoorbeeld op een terras gaan zitten of een andere attractie bezoeken. Wanneer het tijdslot is aangebroken,



Figuur 1: Schematische weergave van het wachtrijproces bij Droomvlucht.

meldt de groep zich bij een aparte ingang van de attractie. Hier belandt het gezelschap in een kleine, aparte wachtrij waar het mogelijk nog een paar minuten moet wachten. Daarna kan men meteen instappen.

2.2 Model

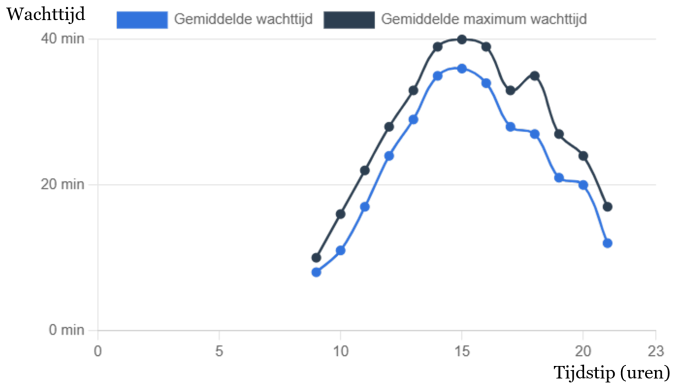
Schematisch kunnen we het wachtrijproces weergeven als in Figuur 1. We nemen aan dat er geen mensen zijn die tijdens het wachten besluiten om de wachtrij te verlaten.

Merk op dat het aantal personen dat in een gondel zit variabel is; personen uit verschillende gezelschappen worden namelijk niet in dezelfde gondel geplaatst. Daarom werken we met groepen personen in plaats van individuen, waarbij we aannemen dat iedere groep in één gondel plaatsneemt. De wachttijd hangt immers af van het aantal gondels dat nodig is om alle wachtende bezoekers te vervoeren, en het is hiervoor niet relevant hoeveel mensen er in elke gondel zitten. In de praktijk kan het voorkomen dat een gezelschap over meerdere gondels gesplitst wordt, bijvoorbeeld bij een gezin met twee ouders en meerdere kinderen. Dit wordt gemodelleerd als twee aparte groepen die vlak achter elkaar bij de attractie aankomen.

In Figuur 2 is de gemiddelde wachttijd van Droomvlucht in 2024 zichtbaar. Merk op dat de aankomstsnelheid van bezoekers duidelijk door de dag heen varieert: gedurende het eerste deel van de dag komen er meer mensen aan dan de attractie aankan, waardoor de wachttijd oploopt, terwijl tijdens het laatste deel van de dag de wachttijd juist afneemt. We zullen het aankomstproces dus niet kunnen weergeven met een Poissonproces met een vaste parameter λ, zoals gebruikelijk is, maar met een niet-stationair Poissonproces met parameter $\lambda(t)$ die afhangt van de tijd t.

Zij T_0 de openingstijd van de attractie en T_N de sluitingstijd van het park, en zij $N(t)$ het aantal groepen dat is aangesloten in de wachtrij in het tijdsinterval $[T_0, T_0 + t]$ voor $t \in [0, T_N - T_0]$. We definiëren het aankomstproces in ons wachtrijmodel als een niet-stationair Poissonproces $\{N(t), t \in [T_0, T_N]\}$ met parameter $\lambda(t)$, zoals gedefinieerd in [1, Sectie 1.3]. Het is tot de sluitingstijd van het park mogelijk om in de wachtrij aan te sluiten, de wachtrij wordt dus niet al eerder gesloten. Daarom stopt het proces wanneer de sluitingstijd van het park bereikt wordt.

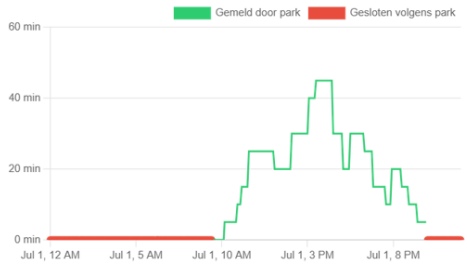
We kunnen het instapproces modelleren als een bedieningsproces met 1 be-



Figuur 2: Gemiddelde wachttijd (minuten) bij Droomvlucht in 2024, grafiek afkomstig van [25], aslabels eigen toevoeging.

diende. Iedere gondel is hierbij dezelfde bediende die een nieuwe groep in laat stappen. De gondels hangen in groepjes van twee bij elkaar, en de afstand tussen twee van deze groepjes is groter; het duurt steeds een paar seconden voordat een nieuw tweetal gondels voorbij komt. In ons model zullen we echter aannemen dat de tijd tussen twee aangrenzende gondels overal hetzelfde is, en de gondels dus niet per tweetal bij elkaar hangen. Dit doen we om te voorkomen dat we steeds na twee groepen een paar seconden extra wachttijd moeten rekenen. Omdat we geïnteresseerd zijn in de gemiddelde wachttijd, en het hier gaat om hoogstens enkele seconden, zal dit de gesimuleerde wachttijden niet minder realistisch maken, terwijl het model er wel eenvoudiger van wordt. De bedieningsduur is hierdoor namelijk deterministisch en constant. Zij g het aantal gondels en T_R de ritduur van de attractie. Dan hebben we een bedieningssnelheid μ van $\mu = \frac{g}{T_R}$ gondels per minuut. Omdat in iedere gondel één groep zit, is dit equivalent aan μ groepen per minuut.

Samengevat is ons basismodel dus een $M_t/D/1$ proces, met een niet-stationair Poissonproces met parameter $\lambda(t)$, en een bedieningsproces met één bediende en een bedieningssnelheid van μ groepen per minuut.



Figuur 3: De wachttijden bij Droomvlucht op 1 juli 2023. Afbeelding afkomstig van [22].

3 Data

We zullen zowel het basismodel als het uitgebreidere Droomvlucht-model simuleren aan de hand van de historische wachttijden van Droomvlucht. Deze gegevens zijn afkomstig van de website queue-times.com [22]. Op deze website worden sinds 2019 de actuele wachttijden van alle Efteling attracties opgeslagen en weergegeven. Die wachttijden worden overgenomen uit de officiële Efteling-app [26]. Dit gebeurt elke vijf minuten.

Het experiment met de virtuele wachtrij bij Droomvlucht heeft plaatsgevonden van 1 juli tot en met 31 augustus 2024. Voor het basismodel zullen we voor consistentie de gegevens van diezelfde periode in 2023 gebruiken.

De wachttijden zijn per dag weergegeven in grafieken zoals die in Figuur 3. De grafieken van de reguliere wachtrij in de periode van het experiment zijn te vinden op [23], de wachttijden van de virtuele rij en van de reguliere rij buiten het experiment om op [22]. Deze wachttijden hebben we opgeslagen in een Excel bestand met behulp van een Python script. In dit bestand konden we op ieder tijdstip de gemiddelde wachttijd van de twee maanden berekenen.

Dit heeft drie datasets opgeleverd. De eerste noteren we met DR, en bestaat uit alle wachttijden bij Droomvlucht in juli en augustus 2023. Als we de wachttijden van een specifieke dag d bedoelen, zullen we dit noteren als $DR[d]$. De gemiddelde wachttijden van de hele periode noteren we met $DR[gem]$. De dataset met de virtuele wachttijden bij Droomvlucht in juli en augustus 2024 wordt weergegeven met DR_v, en de reguliere wachttijden van Droomvlucht in diezelfde periode worden weergegeven met DR_r. Ook bij deze twee datasets zullen we een specifieke dag d aanduiden door er $[d]$ achter te schrijven, of de gemiddelde wachttijden van die periode door er $[gem]$ achter te zetten.

Op queue-times.com staat ook een kalender, met voor iedere dag in zowel het verleden als in de toekomst de openingstijden van de Efteling en (een voorspelling van) het druktepercentage [24]. Hierbij stelt 100% de drukste dag ooit in het park voor, en 0% de rustigste. We zullen het druktepercentage van dag d noteren als $c[d]$, naar het Engelse “crowdedness”.

(a) Grote storing (1) bij Droomvlucht, 9 juli 2023, afbeelding afkomstig van [22].

(b) Storing (2), fout (3) en "onmogelijke"daling (4) bij Droomvlucht, 1 juli 2024, reguliere rij, afbeelding afkomstig van [23].

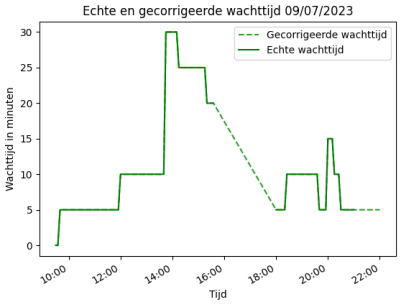
Figuur 4: Voorbeelden van dataproblemen. De cijfers 1 tot en met 4 en de bijbehorende pijlen zijn zelf toegevoegd om deze problemen aan te duiden.

Op alle dagen in juli en augustus van zowel 2023 als 2024 is de officiële openingstijd om 10 uur 's ochtends, en de officiële sluitingstijd wisselt tussen 21:00 en 22:00. Een half uur voor de officiële openingstijd van het park, dus om half 10, opent Droomvlucht al voor verblijfsgasten, die van een andere parkingang gebruikmaken [5]. De reguliere wachtrij sluit pas om sluitingstijd. Alle bezoekers die op dat moment nog in de rij staan, worden toegelaten tot de attractie [3]. Het is onduidelijk of dit ook geldt voor de virtuele rij van Droomvlucht, en wat voor gevolgen dit heeft voor de toegewezen tijdsloten.

De gegevens in het Excel bestand bevatten oorspronkelijk ook de wachttijden voor en na sluitingstijd, die werden weergegeven als 0 minuten. We hebben het bestand ingekort naar alleen de data van 9:30 tot en met 22:00. Als het park om 9 uur al sluit, stond er in de data dat de wachttijd tussen 9 en 10 uur 's avonds 0 minuten was. Omdat we willen voorkomen dat deze dagen de gemiddelde wachttijd na 9 uur beïnvloeden, vervangen we deze cellen door lege cellen, die geen cijfer bevatten en dus niet meetellen bij het gemiddelde.

De data brengen problemen met zich mee. Eén daarvan zijn storingen bij de attractie; deze kunnen variëren van een probleem dat na enkele minuten weer opgelost is tot een storing die uren duurt. Voorbeelden hiervan zijn te zien in Figuur 4a bij (1), van 15:40 tot en met 17:55, en in Figuur 4b bij (2), om 19:45. In de data stonden deze wachttijden met nul aangeduid. Om onderscheid te maken tussen echte wachttijden van nul minuten en storingen, hebben we ook hier deze cellen vervangen door lege cellen in ons Excel bestand.

Ook staan er soms duidelijk fouten in de data. Zo is in Figuur 4b bij (3) te zien dat er om 8:50 en 8:55 al mensen in de wachtrij zouden moeten staan, terwijl dit pas vanaf 9:30 mogelijk is. In dit geval was dit de eerste dag van de proef met de virtuele wachtrij, waardoor er mogelijk een test van het systeem is weergegeven. Hier valt deze fout buiten de tijden in onze dataset, waardoor



Figuur 5: Gecorrigeerde wachttijd bij Droomvlucht, 9 juli 2023.

dit niet voor problemen zorgt. Bij iedere fout die we tegenkomen, zullen we individueel moeten kijken hoe we hier het beste mee om kunnen gaan.

Ook zijn er “onmogelijke” dalingen zichtbaar in de data. Dit zijn dalingen in de wachttijd die theoretisch gezien niet mogelijk zijn. Stel immers dat vanaf een bepaald moment niemand meer aansluit in de wachtrij. Dan kan de wachttijd niet sneller dalen dan met één minuut per minuut. In de data komen echter situaties voor waarin de wachttijd een stuk sneller daalt, zoals in Figuur 4b bij (4) van 60 minuten om 17:20 naar 40 minuten om 17:25. Voor dit soort snelle dalingen zijn verschillende mogelijke verklaringen.

De eerste is dat groepen die al in de wachtrij staan, de rij vroegtijdig verlaten. Dit zou echter op grote schaal moeten gebeuren om de wachttijd te beïnvloeden, terwijl de ervaring leert dat dit slechts incidenteel voorkomt. Een betere verklaring is dat de wachttijden afgerond worden op vijftallen minuten. Zo wordt een werkelijke wachttijd van 7 minuten afgerond op 5 minuten, terwijl een wachttijd van 8 minuten al wordt afgerond op 10 minuten. Een werkelijk wachttijdsverschil van één minuut levert zo in de data al een verschil van vijf minuten op. Daarnaast blijft de weergegeven wachttijd een inschatting die kan afwijken van de werkelijkheid. Het is niet bekend hoe de Efteling de wachttijden berekent of hoe het aantal wachtenden geteld wordt. Als de geschatte wachttijd bij nader inzien te hoog of te laag blijkt te zijn, is het dus goed mogelijk dat deze enkele minuten later wordt bijgesteld, met plotselinge stijgingen of dalingen tot gevolg. Deze combinatie van factoren kan het voorkomen van dit soort theoretisch “onmogelijke” dalingen veroorzaken.

Om te zorgen dat storingen geen invloed hebben op onze simulatie, maken we gebruik van een gecorrigeerde wachttijd. Dit houdt in dat we met behulp van interpolatie de laatste wachttijd voor de storing en de eerste wachttijd na de storing met elkaar verbinden door een lijn, die de gecorrigeerde wachttijd moet voorstellen. Dit is te zien in Figuur 5. Deze gecorrigeerde wachttijd kunnen we vervolgens gebruiken om de benodigde parameters te berekenen, zoals we zullen uitleggen in Hoofdstuk 4.

(a) Wachttijd virtuele rij Droomvlucht, 2 juli 2024, afbeelding afkomstig van [22].

(b) Wachttijd virtuele rij Droomvlucht, 27 juli 2024, afbeelding afkomstig van [22].

Figuur 6: Twee dagen met drukteniveau 47% [24].

Wanneer er "onmogelijke" dalingen in de data voorkomen, kunnen we deze niet nabootsen in de simulatie. Het Poissonproces in ons $M_t/D/1$ model kan immers geen negatieve groepsaantallen genereren, wat wel nodig zou zijn om deze data te volgen. We verwachten daarom dat onze simulatie in deze gevallen hogere wachttijden zal genereren dan de werkelijke, historische wachttijd. Omdat een oplossing van dit probleem vergaande manipulatie van de data of de simulatie zou vereisen, hebben we besloten om deze verschillen te negeren.

Een andere factor waar we rekening mee moeten houden, is dat we niet weten hoe de Efteling de geschatte wachttijden berekent. Dit is met name relevant voor de data tijdens het experiment met de virtuele wachtrij. Het is immers mogelijk dat de Efteling gedurende het experiment merkt dat haar geschatte virtuele of fysieke wachttijd niet accuraat is, en dus deze berekeningen aanpast. Ook kan ze bepaalde strategiën veranderen, zoals de verdeling van groepen uit de twee rijen over de gondels. Op dagen met hetzelfde drukteniveau kunnen dit soort veranderingen er mogelijk voor zorgen dat de wachttijden toch drastisch van elkaar verschillen. Een mogelijk voorbeeld hiervan is weergegeven in Figuur 6. Door deze mogelijke verschillen heeft het simuleren van een gemiddelde dag in juli en augustus 2024 weinig betekenis. Dit zullen we dan ook niet doen.

4 Parameters

In dit hoofdstuk zullen we twee schatters definiëren voor de parameter $\lambda(t)$ van ons aankomstproces, de aankomstsnelheid. Omdat deze schatters ook voor latere modellen gebruikt zullen worden, zullen we dit doen voor een algemene dataset W. In onze simulaties zal W bestaan uit de gekozen virtuele of reguliere wachttijden van een specifieke dag, of een gemiddelde dag. Gegeven deze dataset W en een te simuleren dag d, introduceren we de volgende notatie.

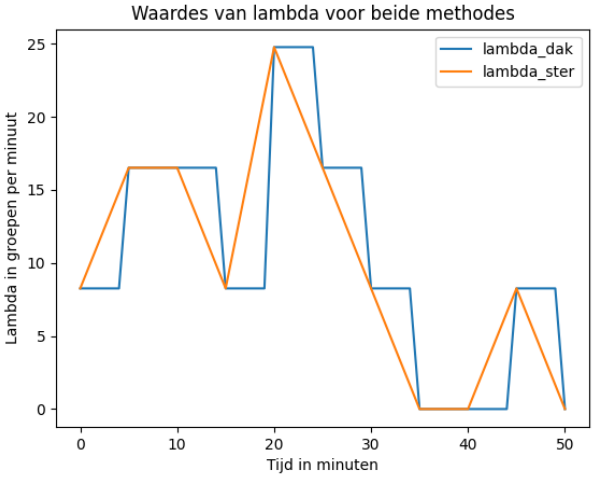
- Zij N het aantal minuten dat de attractie open is gedeeld door 5.
- Zij $T_n = 5n$ voor alle $n \in \{0, 1, \ldots, N\}$. T_0 representeert dan de openingstijd van de attractie, en T_N de sluitingstijd. Tijdstip T_n correspondeert met het n-de vijftal minuten na openingstijd.
- Zij $T_W[d](T_n)$ de historische wachttijd op tijdstip T_n op dag d, voor $n \in \{0, 1, \ldots, N\}$. We zullen $[d]$ alleen schrijven als we duidelijk willen maken welke dag we bedoelen. In de rest van dit hoofdstuk zal deze notatie niet gebruikt worden. Hetzelfde geldt voor $[gem]$ voor de gemiddelde wachttijd van de periode van W.
- Zij $\Delta T_W(n) = T_W(n) - T_W(n-1)$ voor $n \in \{1, 2, \ldots, N\}$ het verschil tussen de n-de en $(n-1)$-ste historische wachttijd in de dataset.
- Zij $g = 2 \cdot 28 = 56$ het aantal gondels.
- Zij $T_R = 6\frac{47}{60}$ de ritduur in minuten.
- Zij $\mu \coloneqq \frac{g}{T_R} \approx 8$ de bedieningssnelheid in groepen per minuut. Merk op dat dit in groepen per minuut kan worden uitgedrukt door de aanname dat één groep in één gondel plaatsneemt.

Omdat we alleen de historische wachttijd op tijdstippen T_n, $n \in \{0, 1, \ldots, N\}$ weten, zullen we eerst een inschatting maken van de aankomstsnelheden op die tijdstippen, die we noteren met $\lambda_{W,n}$. Vervolgens zullen we ook voor de tussengelegen tijdstippen een uitdrukking voor de aankomstsnelheid definiëren, wat een schatter voor $\lambda(t)$ op ieder tijdstip $t \in [T_0, T_N]$ op dag d oplevert.

Merk op dat als $\Delta T_W(n) = 0$ voor een $n \in \{1, 2, \ldots, N\}$, dat er dan evenveel groepen bij de wachtrij aankomen als groepen die de rij verlaten. De aankomstsnelheid is dan dus gelijk aan de bedieningssnelheid, oftewel $\lambda(T_n) = \mu$.

Wanneer $\Delta T_W(n) > 0$ voor een $n \in \{1, 2, \ldots, N\}$, dan komen er bovenop deze μ groepen per minuut nog extra gezelschappen bij de attractie aan. In één minuut worden μ groepen geholpen, dus als de wachttijd met één minuut toeneemt betekent dit dat er μ groepen extra zijn aangesloten. Een toename van $\Delta T_W(n)$ betekent dan dat er $\frac{\Delta T_W(n)}{5}\mu$ groepen per minuut extra de wachtrij hebben betreden.

Voor een negatieve $\Delta T_W(n)$ zijn er juist $\frac{\Delta T_W(n)}{5}\mu$ groepen minder dan μ die zijn aangesloten. In het geval van "onmogelijke"dalingen zou dit tot negatieve



Figuur 7: De twee verschillende schatters van $\lambda(t)$, voor de fictieve dataset $W = [0, 0, 5, 10, 10, 20, 25, 25, 20, 15, 15]$.

waardes van $\lambda(t)$ kunnen leiden. Daarom definiëren we $\lambda_{W,n}$ zodanig dat die nooit negatief kan worden. Dit geeft de volgende getallen $\lambda_{W,n}$:

$$\lambda_{W,n} = \max\left(\mu\left(\frac{\Delta T_W(n+1)}{5} + 1\right), 0\right) \text{ voor } n \in \{0, 1, \ldots, N-1\}. \tag{1}$$

Merk op dat we voor $\lambda_{W,n}$ naar het verschil in wachttijd tussen T_n en T_{n+1} kijken. We willen namelijk dat $\lambda(t)$ correspondeert met hoe de wachttijd de komende 5 minuten gaat veranderen, en niet de afgelopen 5 minuten is veranderd. Dit voorkomt dat de simulatie achter gaat lopen op de werkelijkheid. Omdat er na sluitingstijd geen bezoekers meer aansluiten in de wachtrij, definiëren we $\lambda_{W,N} = 0$.

We definiëren twee verschillende schatters voor $\lambda(t)$, voor $t \in [T_0, T_N]$. De eerste noteren we met $\hat{\lambda}_W(t)$, en houdt in dat we voor $t \in [T_n, T_{n+1})$ kiezen voor $\hat{\lambda}_W(t) = \lambda_{W,n}$, zodat voor elk tijdstip in het interval we dezelfde waarde hebben. Dit geeft

$$\hat{\lambda}_W(t) = \sum_{n=0}^{N-1} \lambda_{W,n} \mathbf{1}_{t \in [T_n, T_{n+1})}. \tag{2}$$

De tweede schatter gebruikt een lineair interpolatiepolynoom, zoals in [4]. Dit geeft

$$\lambda^\star_W(t) = \sum_{n=0}^{N-1} \left[\lambda_{W,n} + \left(\lambda_{W,n+1} - \lambda_{W,n}\frac{t - T_n}{5}\right)\right] \mathbf{1}_{t \in [T_n, T_{n+1})}. \tag{3}$$

Een fictief voorbeeld van de twee schatters is weergegeven in Figuur 7. Bij de simulatie van het basismodel zullen we beide schatters gebruiken, om te kijken welke beter is.

5 Basismodel Simulatie

In dit hoofdstuk voeren we de simulatie van het basismodel uit. Eerst leggen we uit hoe deze simulatie in zijn werk gaat. Vervolgens laten we de resultaten zien van de simulatie van een gemiddelde dag in juli en augustus 2023. Dit doen we zowel voor één simulatieronde als voor het gemiddelde van 10.000 simulaties.

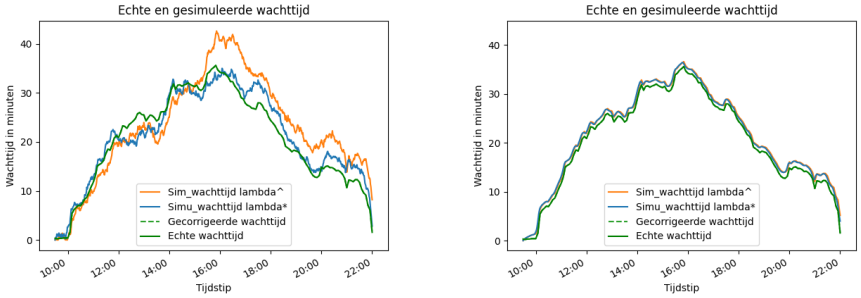
5.1 Simulatie

Voor de simulatie van ons basismodel zullen we de dataset DR gebruiken. Zoals beschreven in Hoofdstuk 3 bestaat deze dataset uit de historische wachttijden van Droomvlucht in juli en augustus 2024, op iedere vijfde minuut n tussen 9:30 en 21:00 of 22:00 (afhankelijk van de sluitingstijd op de desbetreffende dag). Ook de gemiddelde wachttijd op ieder tijdstip is opgenomen in de dataset. De simulatie wordt uitgevoerd met Python. In de simulatie wordt eerst uit de data een specifieke dag d gekozen die we willen simuleren. Daarna wordt de gecorrigeerde wachttijd bepaald, zoals uitgelegd in Hoofdstuk 3. Deze wachttijden worden vervolgens gebruikt om voor iedere minuut dat het park open is de waarde van zowel $\hat{\lambda}_{DR[d]}(t)$ als $\lambda^{\star}_{DR[d]}(t)$ te bepalen, zoals gedefinieerd in Vergelijking 2 en Vergelijking 3. De hiervoor gebruikte ritduur is, zoals gemeten, $T_R = 6\frac{47}{60}$ minuten en het aantal gondels is $g = 2 \cdot 28 = 56$ [27].

Vervolgens worden in de simulatie de volgende arrays gemaakt en gevuld.

- Lambda$[i]$ van lengte $T_N + 1$, met de aankomstsnelheid op ieder tijdstip $i \in [T_0, T_N]$;
- Sim$[i]$ van lengte T_N, met het aantal groepen dat aankomt in tijdsinterval $[i, i+1]$. Dit wordt bepaald door een getal uit de Poissonverdeling met parameter $\lambda(i)$ te trekken;
- Groepen$[i]$ van lengte $T_N + 1$, met het aantal groepen dat op tijdstip i in de wachtrij staat. We nemen daarom Groepen[0]=0, en berekenen de overige getallen door Groepen$[i+1] = \max(\text{Groepen}[i] - \mu + \text{Sim}[i], 0)$;
- Wachttijd_simulatie$[i]$ van lengte T_N+1, met de gesimuleerde wachttijd op tijdstip i. Dit wordt berekend door Wachttijd_simulatie$[i] = \text{Groepen}[i]/\mu$.

Dit wordt eerst uitgevoerd voor $\hat{\lambda}_{DR[d]}(t)$ en daarna nog een keer voor $\lambda^{\star}_{DR[d]}(t)$. Wanneer we meerdere simulaties uitvoeren, wordt voor ieder tijdstip het gemiddelde van alle gesimuleerde wachttijden op dat tijdstip genomen. Dit doen we om te kijken hoe goed de simulatie gemiddeld is; omdat de simulatie stochastisch is kan één enkele simulatie flinke uitschieters hebben. We verwachten dat het gemiddelde nauwkeuriger is wegens de wet van de grote aantallen. Vervolgens worden beide simulaties samen met de historische en gecorrigeerde wachttijden afgedrukt in een grafiek. De gebruikte code staat in Bijlage A.



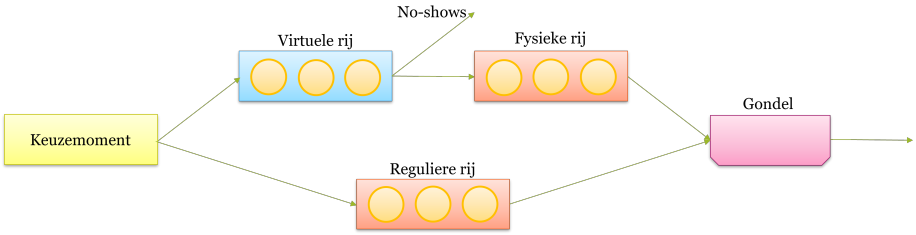
(a) 1 keer uitgevoerd.

(b) 10.000 keer uitgevoerd.

Figuur 8: Simulatie gemiddelde dag juli en augustus 2023.

5.2 Resultaten

In Figuur 8 is te zien hoe we de simulatie uitgevoerd hebben voor het gemiddelde van de data uit juli en augustus 2023. We hebben de simulatie eerst één keer uitgevoerd (Figuur 8a), en daarna het gemiddelde van 10.000 simulaties genomen (Figuur 8b). Bij één simulatie wordt de historische wachttijd al erg redelijk benaderd, al zijn er wat uitschieters. Bij 10.000 simulaties wordt de vorm van de historische wachttijd bijna perfect gevolgd. Echter ligt de gesimuleerde wachttijd ongeveer een minuut boven de werkelijke wachttijd. Dit verschil lijkt in het eerste half uur te ontstaan. Bovendien lijkt er vrijwel geen verschil te zijn tussen de schatters $\hat{\lambda}_{DR}(t)$ en $\lambda^{\star}_{DR}(t)$. We nemen aan dat dit altijd het geval is, en niet alleen bij deze specifieke simulatie. Omdat $\hat{\lambda}_{DR}(t)$ de eenvoudigste methode is, kiezen we ervoor om deze voor de andere simulaties te gebruiken.



Figuur 9: Schematische weergave van wachtrijproces bij Droomvlucht, juli en augustus 2024.

6 Droomvlucht Model

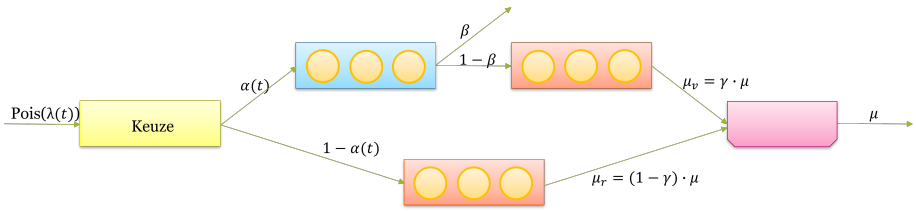
In dit hoofdstuk wordt het basismodel uit Hoofdstuk 2 uitgebreid naar een model dat ook de virtuele wachtrij omvat. Eerst wordt uitgebreid uitgelegd hoe de virtuele wachtrij werkt, en vervolgens bekijken we hoe we dit kunnen implementeren in ons model.

6.1 Virtuele Wachtrij

Het wachtrijproces bij Droomvlucht inclusief de virtuele wachtrij is weergegeven in Figuur 9. Elk gezelschap dat de attractie wil bezoeken maakt de keuze of het in de reguliere of de virtuele rij wil gaan staan. In het geval van de reguliere rij belandt men in een fysieke wachtrij, waar men wacht tot men aan de beurt is. Als er voor de virtuele rij wordt gekozen, kan het gezelschap via de Efteling-app aansluiten in de virtuele wachtrij. De groep krijgt dan een tijdslot toegewezen.

Ieder tijdslot bestaat uit een begintijd en een eindtijd 15 minuten later. Een mogelijk tijdslot is bijvoorbeeld 11:00-11:15. Merk op dat hoewel dit tijdslot een kwartier lijkt te duren, er 16 minuten zijn waarin de groep zich bij de attractie kan melden: het eerste tijdstip is namelijk 11:00:00, en het laatste tijdstip 11:15:59. De tijdsloten van verschillende groepen kunnen met elkaar overlappen; het is dus mogelijk dat de ene groep 11:00-11:15 als tijdslot heeft, en een andere groep 11:01-11:16. De begintijd van het tijdslot is gelijk aan het huidige tijdstip plus de huidige virtuele wachttijd. Als mensen niet binnen hun tijdslot komen opdagen, worden ze niet meer toegelaten tot de attractie. Deze groepen noemen we no-shows.

Uit privé-correspondentie met Efteling-podcast Kleine Boodschap, gemaakt door Eftelingliefhebbers, bleek dat het mogelijk is dat tijdsloten vervroegd worden. Dit konden grote verschuivingen zijn, van bijvoorbeeld 45 minuten of meer. We nemen aan dat dit komt door de no-shows; als mensen niet op komen dagen kunnen andere groepen eerder instappen, wat voor een domino-effect zorgt. Hierdoor kunnen de tijdsloten steeds verder vervroegen. We hebben echter geen gegevens over hoe vaak dit voorkwam en hoe groot deze verschuivingen waren. In ons model zullen we de kans dat een groep niet op tijd voor zijn tijdslot komt



Figuur 10: Model van wachtrijproces bij Droomvlucht, juli en augustus 2024.

opdagen noteren als β. We zullen hier echter geen expliciete uitdrukking voor proberen te vinden, omdat we hier geen informatie over hebben.

Wanneer het tijdslot is aangebroken, krijgt men een melding in de app dat de groep zich bij de attractie kan melden, en verschijnt er een qr-code. Wanneer men zich meldt bij de ingang van de attractie, kan de groep deze qr-code laten scannen om binnen te komen. De groep belandt dan in een korte fysieke wachtrij, die gescheiden is van de reguliere wachtrij. Aan het einde komen beide rijen uit bij een medewerker, die groepen uit beide rijen in laat stappen. De virtuele rij bestaat dus eigenlijk uit twee wachtrijen: de werkelijke virtuele rij waar groepen niet fysiek in staan, en de korte fysieke rij voordat men de attractie kan betreden. Merk op dat de volgorde van groepen in de twee rijen kan verschillen: als één groep achter een tweede groep in de eerste, virtuele rij staat, is het mogelijk dat deze groep zich eerder dan die tweede groep bij de attractie meldt, en dus eerder de tweede, fysieke rij en de attractie betreedt.

Anders dan bij het basismodel, moeten er hier bij het instappen twee wachtrijen samengevoegd worden, namelijk de mensen uit de virtuele en uit de reguliere rij. Volgens de Efteling-podcast Kleine Boodschap wordt er steeds 1 groepje uit de virtuele rij toegelaten en dan weer 2 of 3 uit de reguliere rij [12].

6.2 Model

Als aankomstproces nemen we weer een niet-stationair Poissonproces met parameter $\lambda(t)$. Wanneer de groepen aankomen bij de attractie, moeten ze kiezen of ze in de virtuele rij of de reguliere rij aansluiten. De fractie van hoeveel mensen er voor de virtuele rij kiezen, kan door de dag heen veranderen. We nemen daarom aan dat de groepen op tijdstip $t \in [T_0, T_N]$ kiezen voor de virtuele rij met kans $\alpha(t) \in [0, 1]$, en met kans $1-\alpha(t)$ voor de reguliere rij. In Hoofdstuk 7 wordt dieper in gegaan op de keuze om α van t af te laten hangen. De groepen worden op tijdstip t dus verdeeld over de rijen volgens een Bernoulliproces met parameter $\alpha(t)$.

In de eerste virtuele rij wacht iedere groep tot zijn toegewezen tijdslot aangebroken is. We nemen aan dat een groep met kans $p_0 = \beta \in [0, 1]$ niet komt opdagen voor zijn tijdslot, en dat met kans $p_i = \frac{1-\beta}{16}$ de groep in de i-de minuut van zijn tijdslot aansluit in de tweede rij, voor $i \in \{1, 2, \ldots, 15, 16\}$. Met kans

$1 - \beta$ komt de groep dus wel op tijd bij de attractie aan.

Bij het instappen moeten groepen uit zowel de virtuele als de reguliere rij over de gondels verdeeld worden. Uit privé-correspondentie met Efteling-podcast Kleine Boodschap blijkt dat het waarschijnlijk is dat dit altijd met grofweg dezelfde verhouding gebeurt. We nemen daarom aan dat een fractie $\gamma \in [0, 1]$ van alle voertuigen gevuld wordt met groepen uit de virtuele rij, en de rest met groepen uit de reguliere rij. De virtuele rij volgt dan dus een bedieningsproces met één bediende en bedieningssnelheid $\mu_v = \gamma\mu$, en de reguliere rij een bedieningsproces met één bediende en bedieningssnelheid $\mu_r = (1-\gamma)\mu$, met μ zoals in Hoofdstuk 2.2. Het gehele model staat weergegeven in Figuur 10.

7 Droomvlucht Parameters

In dit hoofdstuk zullen we veel subscripts en de notatie $[d]$ en $[gem]$ gebruiken, om het onderscheid tussen de verschillende methodes en datasets duidelijk te maken. In de rest van deze scriptie zullen we deze notaties weglaten indien duidelijk is wat we bedoelen.

Om met ons model de wachttijden van een specifieke dag d te simuleren, hebben we een schatter voor onze parameter $\lambda(t)$ nodig. Voor het basismodel hebben we deze schatter bepaald aan de hand van de wachttijden van dag d. We gebruikten dus data om diezelfde data te reproduceren. Dit kan nuttig zijn om te testen hoe goed het model werkt, maar in de praktijk is het zinloos. Wanneer men een voorspelling wil maken van de toekomst, zijn die data immers nog niet beschikbaar. Voor ons Droomvlucht model zullen we daarom, naast deze manier (Methode 0), nog twee andere methodes om een schatter voor $\lambda(t)$ te definiëren ontwikkelen. Deze methodes zullen ook gebaseerd zijn op historische data, maar gebruiken de gemiddelde wachttijden over een langere periode, in plaats van de wachttijden op één specifieke dag.

Omdat de aankomststroom op een bepaalde dag d afhangt van hoe druk het op die dag is, zullen we hier rekening mee moeten houden. Dit doen we door de gemiddelde wachttijden van de dataset die we gebruiken te schalen voor de drukte. Hoewel het op het eerste gezicht misschien logischer lijkt om onze schatter voor de aankomstsnelheid $\hat{\lambda}_{W[gem]}(t)$ te schalen, kiezen we ervoor om in plaats daarvan de wachttijden te schalen. Dit is omdat de drukteniveaus gebaseerd zijn op de wachttijden in het park [26].

Bij beide nieuwe methodes nemen we aan dat we de wachttijden op een dag d met druktepercentage $c[d]$ kunnen bepalen door de gemiddelde wachttijd hiernaar te schalen. We doen dit door de gemiddelde wachttijd te delen door het gemiddelde druktepercentage, en dit te vermenigvuldigen met het druktepercentage van dag d. In formuleform levert dit de volgende definitie op. Voor een dataset W, gemiddelde wachttijd van die dataset $W[gem]$, en gemiddeld druktepercentage $c_{W,g}$, definiëren we de geschaalde wachttijd $T_{W,s[d]}$ op tijdstip n door

$$T_{W,s[d]}(n) \coloneqq \frac{c[d]}{c_{W,g}} T_{W[gem]}(n). \tag{4}$$

De druktepercentages zijn afkomstig van de website queue-times.com [24], zoals besproken in Hoofdstuk 3.

We zullen de drie methodes hieronder uitwerken voor de virtuele rij bij Droomvlucht. De datasets die we gebruiken zijn $DRv[d]$ en $DRr[d]$ voor Methode 0, $DRv[gem]$ en $DRr[gem]$ voor Methode 1, en $DR[gem]$ voor Methode 2. We kunnen deze methodes ook voor andere attracties gebruiken door een vergelijkbare dataset te nemen.

7.1 Methode 0

Zoals gezegd gebruiken we bij Methode 0 alleen de wachttijden van de dag d die we willen simuleren. We hebben echter twee datasets, $DRv[d]$ en $DRr[d]$, terwijl we één totale aankomstsnelheid willen weten. We kunnen dit oplossen door de aankomststromen van de virtuele en de reguliere rij te berekenen, en deze bij elkaar op te tellen. We definiëren daarom

$$\hat{\lambda}_0^{tot}(t) \coloneqq \hat{\lambda}_{DRv[d]}(t) + \hat{\lambda}_{DRr[d]}(t) \tag{5}$$

voor een gegeven attractie (in dit geval Droomvlucht) en dag d. Merk op dat het voor deze methode niet nodig is om te schalen voor drukte, aangezien we alleen de wachttijden van dag d gebruiken.

7.2 Methode 1

Bij Methode 1 gebruiken we de virtuele en reguliere wachttijden uit de periode dat de virtuele rij actief was. Bij ons huidige model zijn dit de datasets DRv en DRr. Het voordeel hiervan is dat een eventueel veranderd aankomstpatroon door toevoeging van de virtuele rij geen probleem is, omdat we alleen dit nieuwe aankomstpatroon bekijken. Wel moeten er eerst voldoende gegevens verzameld worden.

Bij deze methode bepalen we eerst de gemiddelde geschaalde aankomstsnelheid bij de virtuele en de reguliere wachtrij, en tellen we deze vervolgens bij elkaar op om de totale aankomstsnelheid te vinden. We schalen dus eerst $T_{DRv[gem]}$ en $T_{DRr[gem]}$ met behulp van Vergelijking 4, wat $T_{DRv[gem],s[d]}$ en $T_{DRr[gem],s[d]}$ oplevert. Vervolgens definiëren we

$$\hat{\lambda}_1^{tot}(t) \coloneqq \hat{\lambda}_{DRv[gem],s[d]}(t) + \hat{\lambda}_{DRr[gem],s[d]}(t) \tag{6}$$

als schatter van de aankomstsnelheid $\lambda(t)$, met schatters $\hat{\lambda}_{DRv[gem],s[d]}(t)$ en $\hat{\lambda}_{DRr[gem],s[d]}(t)$ zoals gedefinieerd in Vergelijking 2. Bij het bepalen van deze laatste twee schatters met behulp van Vergelijking 1 nemen we $\gamma\mu$ in plaats van μ voor $DRv[gem]$, en $(1-\gamma)\mu$ in plaats van μ voor $DRr[gem]$. Hierbij is γ de fractie van het aantal gondels dat door de virtuele rij wordt opgevuld.

7.3 Methode 2

We kunnen ook data gebruiken uit de tijd dat er nog geen virtuele rij was. In dit geval is dat de dataset DR. In het algemeen is het voordeel van deze methode dat deze methode meteen geïmplementeerd kan worden als een nieuwe virtuele rij opent, zonder te moeten wachten tot er genoeg data van deze virtuele rij beschikbaar zijn. Men zal in deze situatie wel een andere schatter voor $\alpha(t)$ moeten vinden, omdat wij deze wel laten afhangen van de virtuele wachttijd.

Voor deze methode nemen we aan dat de totale aankomststroom bij de attractie niet verandert als er een virtuele wachtrij wordt toegevoegd. Zij d de dag die

we simuleren, en zij $c_{DRv,g}$ de gemiddelde drukte in juli en augustus 2023. We kunnen de wachttijden uit dataset $DR[gem]$ schalen met behulp van Vergelijking 4, wat de dataset $DR[gem], s[d]$ oplevert. Voor een gegeven attractie (in dit geval Droomvlucht) en dag (d) definiëren we vervolgens

$$\hat{\lambda}_2^{tot}(t) \coloneqq \hat{\lambda}_{DR[gem],s[d]}(t), \tag{7}$$

met $\hat{\lambda}_{DR[gem],s[d]}(t)$ zoals gedefinieerd in Vergelijking 2.

7.4 Overige Parameters

De waarde van de kans α zal in de praktijk afhangen van veel factoren, waaronder de virtuele en reguliere wachttijd op dat moment. Het opstellen van een goede uitdrukking hiervoor valt echter buiten het doel van deze scriptie. Om het model eenvoudig te houden nemen wij daarom aan dat de kans $\alpha(t)$ alleen afhangt van de tijd. We doen dit door de gemiddelde aankomststroom bij de virtuele wachtrij te delen door de totale aankomststroom, voor ieder tijdstip t. Zo kiezen we onze schatter $\hat{\alpha}(t)$ van $\alpha(t)$ op basis van wat bezoekers in het verleden op datzelfde tijdstip gekozen hebben. We definiëren dus de schatter

$$\hat{\alpha}(t) \coloneqq \frac{\hat{\lambda}_{DRv[gem]}}{\hat{\lambda}_{DRv[gem]} + \hat{\lambda}_{DRr[gem]}}. \tag{8}$$

Merk op dat $\hat{\alpha}(t)$ voor iedere dag die we simuleren hetzelfde is.

Zoals eerder gezegd hebben we geen informatie over hoe vaak no-shows voorkomen, en kunnen we onze keuze voor de parameter β dus nergens op baseren. Voor de simulatie van het model hebben we deze parameter echter niet nodig. Onze simulatie wordt immers gebaseerd op de historische wachttijden zoals de Efteling ze heeft ingeschat. We hebben aangenomen dat door de no-shows de wachttijden van de virtuele rij in de praktijk vaak korter waren dan de Efteling inschatte. Wanneer we wel rekening houden met de no-shows, zal de simulatie dus een stuk lager uitvallen dan de bedoeling is. Dit zou goed zijn als we het verschil tussen de werkelijke en geschatte wachttijden wilden onderzoeken, maar voor nu willen we alleen de door de Efteling geschatte wachttijden benaderen.

8 Droomvlucht Simulatie

In dit hoofdstuk voeren we de simulatie van het basismodel uit. We hebben één willekeurige dag gekozen, 17 juli 2024, en zullen deze simuleren met de drie methodes die we in het vorige hoofdstuk besproken hebben. Deze simulatie zullen we steeds eerst één keer en vervolgens 10.000 keer uitvoeren. We leggen eerst uit welke stappen de simulatie volgt, en vervolgens laten we de resultaten zien.

8.1 Simulatie

We zullen steeds de virtuele en reguliere wachttijden van een specifieke dag d simuleren. Zoals gedefinieerd in Hoofdstuk 7 hebben we drie manieren om een schatter voor de parameter $\lambda(t)$ te definiëren, namelijk Methode 0, 1 en 2. Dit geeft respectievelijk de schatters $\hat{\lambda}_0^{tot}(t)$, $\hat{\lambda}_1^{tot}(t)$ en $\hat{\lambda}_2^{tot}(t)$. De getallen $c[d]$ en $c_{W,g}$ zijn afkomstig van de website queue-times.com [24]. Voor methode 1 vonden we een gemiddeld druktepercentage van $c_{DRv,g} = c_{DRr,g} = 50\%$, voor methode 2 een gemiddeld druktepercentage van $c_{DR,g} = 51\%$.

We gebruiken de schatter $\hat{\alpha}(t)$ zoals gedefinieerd in 7. Net als in Hoofdstuk 5 nemen we ritduur $T_R = 6\frac{47}{60}$ en $g = 56$ gondels. Daarnaast nemen we $\gamma = \frac{1}{4}$. Parameter β wordt in de simulatie niet gebruikt, zoals uitgelegd in Hoofdstuk 7.

In de simulatie wordt eerst alle benodigde data ingelezen. Wanneer het park om 9 uur sluit wordt de simulatie een uur eerder afgekapt. Ook wordt er gecorrigeerd voor storingen, zoals uitgelegd in Hoofdstuk 3. Vervolgens worden de volgende stappen uitgevoerd.

1. De gekozen schatter voor $\lambda(t)$ (corresponderend met Methode 0, Methode 1 of Methode 2) wordt bepaald. Dit wordt gedaan op iedere minuut $n \in \{0, 1, \ldots, T_N\}$.
2. De schatter $\hat{\alpha}(n)$ wordt berekend.
3. Voor iedere minuut $n \in \{0, 1, \ldots, T_N\}$ wordt er met de Poissonverdeling met parameter $\lambda(n)$ een getal getrokken, dat het aantal aangekomen groepen in die minuut voorstelt.
4. Voor elke gesimuleerde groep wordt er met de Bernoulliverdeling met parameter $\alpha(n)$ een 0 of een 1 getrokken. Als dit een 1 is, wordt deze groep toegekend aan de virtuele rij. Als dit een 0 is, wordt de groep toegekend aan de reguliere rij.
5. In twee arrays, Simvirt en Simreg, wordt voor iedere minuut opgeslagen hoeveel groepen er aan respectievelijk de virtuele en de reguliere rij zijn toegekend.
6. In de arrays Groepenvirt en Groepenreg wordt berekend hoeveel mensen er iedere minuut in respectievelijk de virtuele en de reguliere wachtrij staan.

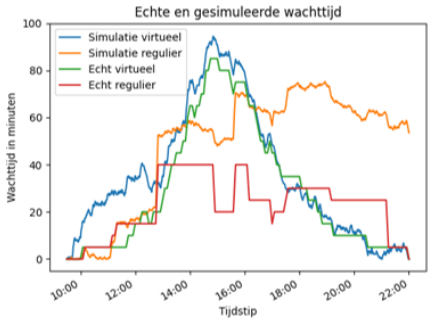
Op tijdstip 0 staan er 0 mensen in iedere rij. Vervolgens definiëren we Groepenvirt$[n+1] = \max(\text{Groepenvirt}[n] - \gamma\mu + \text{Simvirt}[n], 0)$ en Groepenreg$[n+1] = \max(\text{Groepenreg}[n] - (1-\gamma)\mu + \text{Simreg}[n], 0)$, voor $n \in \{0, 1, \ldots, T_N\}$.

7. Vervolgens worden de wachttijden berekend en opgeslagen in de arrays Wachttijdvirt en Wachttijdreg. Dit doen we met
Wachttijdvirt$[n] = \text{Groepenvirt}[n]/(\gamma\mu)$ en
Wachttijdreg$[n] = \text{Groepenreg}[n]/((1-\gamma)\mu)$.

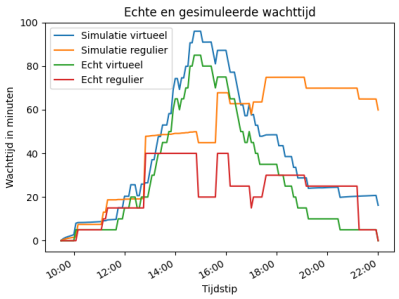
Bij meerdere simulatierondes herhalen we stap 3 tot en met 7, en wordt het puntsgewijs gemiddelde bepaald van de gesimuleerde wachttijden. Vervolgens worden de gesimuleerde wachttijden samen met de gecorrigeerde historische wachttijden afgedrukt in een grafiek. Hierna wordt het absolute verschil tussen de simulatie en de historische wachttijden berekend, en wordt deze fout weergegeven in een grafiek.

De gebruikte code is te vinden in Bijlage B en Bijlage C. Voor Methode 0 gebruiken we dezelfde code als voor Methode 1, alleen passen we dan de gebruikte datasets aan van $DRv[gem]$ en $DRr[gem]$ naar respectievelijk $DRv[d]$ en $DRr[d]$.

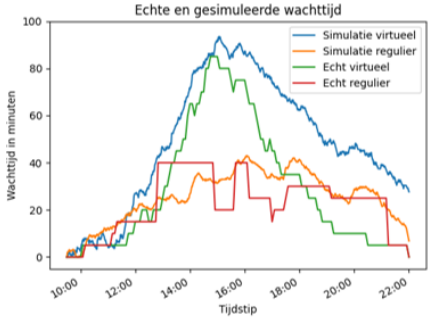
8.2 Resultaten



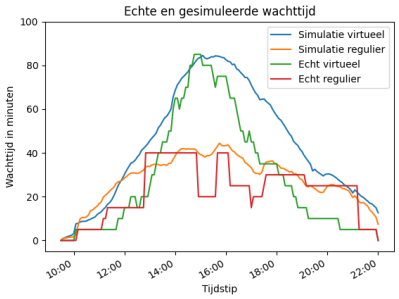
(a) Methode 0, 1 keer uitgevoerd.



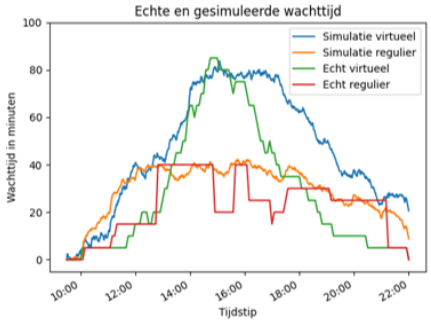
(b) Methode 0, 10.000 keer uitgevoerd.



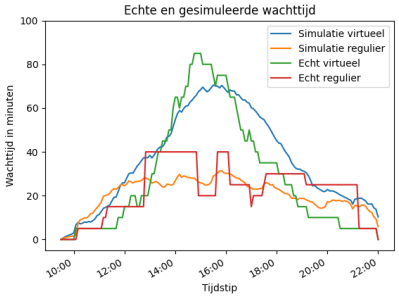
(c) Methode 1, 1 keer uitgevoerd.



(d) Methode 1, 10.000 keer uitgevoerd.



(e) Methode 2, 1 keer uitgevoerd.



(f) Methode 2, 10.000 keer uitgevoerd.

Figuur 11: Resultaten simulatie 17 juli 2024 Droomvlucht.

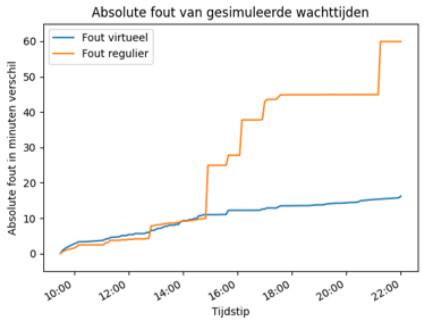
In Figuur 11 zijn de resultaten te zien van alle drie de simulatiemethodes voor 17 juli 2024. Dit is een willekeurig gekozen dag. Op deze dag was het druktepercentage 57% [24]. Elke simulatie hebben we eerst 1 keer uitgevoerd, en daarna 10.000 keer. Voor Figuren 11a en 11b hebben we Methode 0 gebruikt. We zien

dat de simulatie van de reguliere rij tot 3 uur gemiddeld iets hoger lijkt te liggen dan de virtuele data, maar wel heel erg dezelfde vorm lijkt te hebben. Na 3 uur wordt er nog steeds gepiekt en gedaald op dezelfde momenten, maar doordat er op 17 juli veel "onmogelijke dalingen"plaatsvinden, komt de simulatie veel hoger te liggen dan de werkelijke wachttijden. Bij de virtuele rij vinden er geen onmogelijke dalingen plaats, maar eindigt de simulatie ook een stuk hoger.

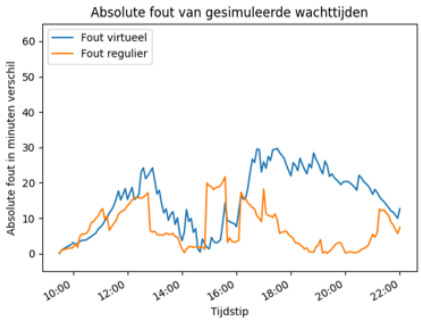
In Figuur 12a is de absolute fout te zien van deze simulatie. Deze fout is gedefinieerd als het absolute verschil tussen de gesimuleerde wachttijd en de werkelijke wachttijd op ieder tijdstip T_n voor $n \in [0, N]$. Hier komt inderdaad in terug dat de fout van de reguliere rij erg groot wordt, en dat de fout van de virtuele rij ook door de dag heen stijgt. De simulatie lijkt deels iets te hoog uit te vallen, hoewel het grootste deel van de reguliere fout veroorzaakt wordt door onmogelijke dalingen. Een methode die hier geen problemen mee heeft zou dus beter zijn.

De resultaten van Methode 1 voor 17 juli 2024 zijn weergegeven in Figuren 11c en 11d. De simulatie volgt niet precies de pieken en dalen van de historische wachttijden, maar volgt wel enigszins hetzelfde patroon. In Figuur 12b zien we dat de absolute fout van de virtuele rij meestal een stuk hoger uitvalt, maar dat de fout van de reguliere rij het tweede deel van de dag een stuk lager ligt dan bij Methode 0. Dit zien we ook terug in Figuur 12d: de gemiddelde absolute reguliere fout ligt veel lager dan bij Methode 0, hoewel de virtuele fout juist wat hoger ligt.

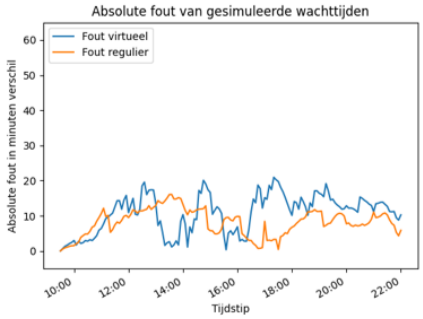
Voor Methode 2 gebruiken we zoals gezegd de gemiddelde wachttijden van juli en augustus 2023, toen er nog geen virtuele wachtrij was. We gebruiken deze data om onze parameter $\lambda(t)$ te bepalen, maar hebben nog steeds de data uit 2024 nodig om $\alpha(t)$ te bepalen. Dit geeft vergelijkbare resultaten als met Methode 1, al lijken zowel de virtuele als reguliere simulaties wat lager te liggen. Ook de fouten zijn enigszins vergelijkbaar met die van Methode 1. In Figuren 12c en 12d zien we dat de virtuele wachttijden iets accurater lijken, en de reguliere rij iets minder accuraat dan Methode 1.



(a) Methode 0.



(b) Methode 1.



(c) Methode 2.

	Gemiddelde absolute fout virtueel	Gemiddelde absolute fout regulier
Methode 0	10.2	26.4
Methode 1	15.2	7.0
Methode 2	10.7	8.2

(d) Gemiddelde fouten.

Figuur 12: Absolute fout bij 10.000 simulaties en gemiddelde van de absolute fouten, 17 juli 2024 Droomvlucht.

Samengevat lijkt Methode 0 de beste resultaten te geven voor de virtuele rij, maar veruit de slechtste voor de reguliere rij. Naast het feit dat we deze manier niet kunnen gebruiken om voorspellingen te doen, geven de slechte resultaten bij de reguliere wachttijd een extra reden om deze methode niet verder te bekijken. Methode 1 geeft de beste resultaten voor de reguliere wachtrij, maar de slechtste voor de virtuele wachtrij. Methode 2 zit steeds tussen de andere twee manieren in, maar zit zowel bij de virtuele als bij de reguliere rij erg dicht bij de beste methode. Daarom lijkt Methode 2 over het algemeen de beste resultaten te geven. Dit is onverwacht, omdat het voor de hand lag dat het gemiddelde van 2024 dichter in de buurt zou liggen bij wachttijden van 2024.

Figuur 13: Plattegrond van omgeving Danse Macabre, met in blauw de route van de reguliere wachtrij, rood het fysieke stukje van de virtuele wachtrij (Rij 3), en geel de drie poorten. Afbeelding afkomstig van [7], routes en poorten zijn een eigen toevoeging.

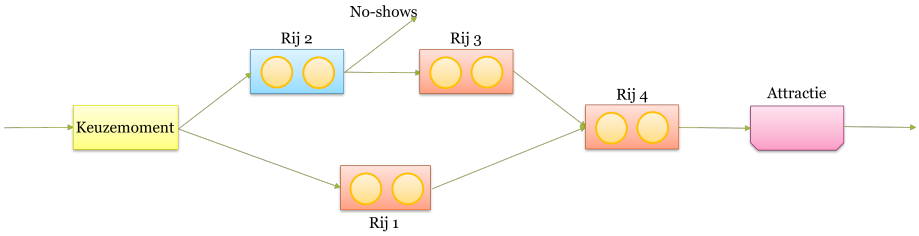
9 Danse Macabre Model

Op 22 april 2025 kondigde de Efteling aan dat er per direct een virtuele wachtrij geopend werd bij Danse Macabre. Dit gebeurde terwijl het onderzoek naar de virtuele rij bij Droomvlucht in volle gang was, en bood de kans om een model te ontwikkelen dat ook in de praktijk getest kon worden. Daarom hebben we besloten om het model voor Droomvlucht aan te passen naar de situatie bij Danse Macabre. Dit bracht de nodige uitdagingen met zich mee. Het wachtrijsysteem bij Danse Macabre is namelijk een stuk ingewikkelder, en ook het bedieningsproces verloopt anders. We zullen in dit hoofdstuk meer vertellen over de attractie Danse Macabre, en daarna het opgestelde model beschrijven.

9.1 Attractie

Danse Macabre opende op 31 oktober 2024. Het attractiegebouw is vormgegeven als een oude, vervallen abdij en heeft een griezelthema. De attractie bestaat uit een grote kantelende draaischijf met daarop zes koorbanken, die ook allemaal kunnen draaien. Iedere koorbank bestaat uit drie banken van zes zitplekken. In totaal kan de attractie dus 108 personen per keer bedienen.

Danse Macabre heeft een single rider wachtrij. Dit is een wachtrij waarin mensen kunnen aansluiten die het niet erg vinden om niet bij hun gezelschap in de attractie te zitten, of mensen die in hun eentje de attractie bezoeken. Mensen uit deze wachtrij worden gebruikt om eventuele lege plekken in de koorbanken op te vullen. Er is geen aparte wachtrij voor invalide bezoekers; mensen met



Figuur 14: Schematische weergave van wachtrijproces bij Danse Macabre.

een beperking kunnen de hele wachtrij tegelijk met de rest van de bezoekers afleggen. Wanneer men niet in staat is om de attractie op de gebruikelijke manier te bezoeken, wordt men in het gebouw naar een andere ruimte geleid om daar een gefilmde versie te bekijken.

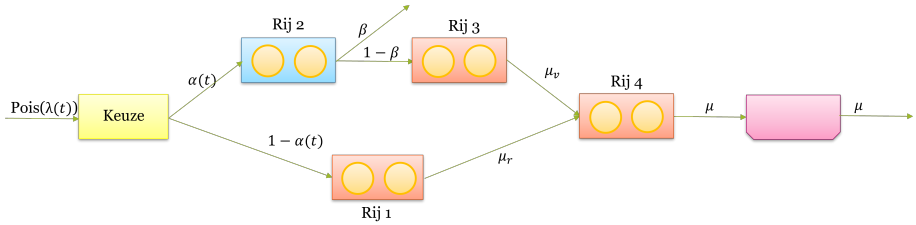
Het volledige wachtrijproces bij Danse Macabre bestaat eigenlijk uit 4 wachtrijen. Wanneer men kiest voor de reguliere wachtrij wordt men door verschillende gethematiseerde gebieden gevoerd, weergegeven in Figuur 13. De bezoekers lopen door de kruisgang en kruidentuin, waarna men tot stilstand komt bij Poort 1. Dit gedeelte noemen we Rij 1.

De virtuele rij wordt Rij 2 genoemd. Deze rij werkt volgens hetzelfde principe als de virtuele rij bij Droomvlucht; men sluit aan in de Efteling-app en krijgt een tijdslot van een kwartier toegewezen. De tijdsloten kunnen weer overlappen. Wanneer het tijdslot van een groep aanbreekt, kan men zich melden bij een speciale ingang van de attractie. Hier belandt men in een korte rij, Rij 3 genoemd. In Figuur 13 is deze weergegeven door het rode lijntje.

Op het kerkhof wordt Rij 4 gevormd. Hierbij wordt steeds eerst iedereen die op dat moment in Rij 3 staat via Poort 3 toegelaten op de begraafplaats. Vervolgens wordt het aantal mensen op het kerkhof aangevuld tot ongeveer 108, door reguliere bezoekers uit Rij 1 door Poort 1 te laten. Hoeveel mensen er precies op het kerkhof komen kan variëren, en is een inschatting van de medewerker die de poorten opent [20, minuut 34].

Vervolgens vindt er een kleine voorshow plaats in de vorm van een verteld verhaal, waarna Poort 2 opent en de groep verder kan doorlopen. Hier bereikt men het attractiegebouw, een vervallen abdij. Bij de ingang van het gebouw wordt iedereen alvast verdeeld over één van zes gangen, die elk naar een andere koorbank leiden. Als de vorige show voorbij is en iedereen de zaal verlaten heeft, gaan de deuren van de catacomben open en kan iedereen uit de volgende groep plaatsnemen.

Merk op dat virtuele bezoekers van Danse Macabre op vier momenten nog fysiek moeten wachten: in de rij voor het kerkhof, op het kerkhof zelf, voor ze het gebouw betreden, en in het gebouw. Dit kan bij elkaar al snel oplopen tot 20 minuten. Dit wordt echter niet meegerekend in de wachttijd die de Efteling-app



Figuur 15: Model van wachtrijproces bij Danse Macabre.

weergeeft; dat is namelijk enkel de tijd tot het aanbreken van het tijdslot.

We kunnen deze wachtrijen schematisch weergeven zoals in Figuur 14. Het belangrijkste verschil met Figuur 9 is dat de virtuele en reguliere rij al eerder samengevoegd worden, en samen nog door een stuk fysieke wachtrij moeten.

Door zelf naar de Efteling te gaan, hebben we ontdekt dat de tijdsloten vervroegd kunnen worden, net als eerder bij de virtuele rij van Droomvlucht. We nemen aan dat dit komt doordat mensen niet op tijd bij hun tijdslot komen opdagen ("no-shows"). De plekken die eigenlijk voor hen bedoeld waren, worden dan opgevuld door de tijdsloten daarna. Hierdoor schuiven de tijdsloten steeds verder op. De gegevens van de geobserveerde verschuivingen staan in Hoofdstuk 11.

Bij de virtuele rij van Droomvlucht was het onduidelijk wat er gebeurde als de sluitingstijd van het park naderde, maar bij Danse Macabre konden we dit zelf onderzoeken. Er bleek dat de virtuele rij sluit zodra de virtuele wachttijd hoger wordt dan de tijd tot sluitingstijd. Dit voorkomt dat er tijdsloten ontstaan die pas na sluitingstijd zijn. Doordat de tijdsloten ondertussen naar voren blijven schuiven, komen er aan het einde van de dag weer tijdsloten vrij. Daarom wordt aan het einde van de dag de virtuele rij soms opnieuw open gezet. Dit is te zien in Figuur 16 in Hoofdstuk 10.

9.2 Model

Hoewel het wachtrijproces van Danse Macabre een stuk ingewikkelder is dan dat van Droomvlucht, start ook dit proces met een niet-stationair Poissonproces met parameter $\lambda(t)$, $t \in [T_0, T_N]$. Elke gesimuleerde persoon krijgt een groepsgrootte van minstens 1 en maximaal 6 toegewezen. Dit gebeurt met een kansverdeling π. We nemen aan dat deze kansverdeling voor de virtuele en reguliere rij hetzelfde is. Iedere groep kiest vervolgens voor de virtuele rij met kans $\alpha(t)$, en voor de reguliere rij met kans $1 - \alpha(t)$, voor $t \in [T_0, T_N]$.

Iedere groep in de virtuele rij is een no-show met kans $\beta(T_W(t))$, die afhangt van de virtuele wachttijd op tijdstip $t \in [T_0, T_N]$ waarop de groep in de virtuele rij aansluit. In de praktijk zal de kans op no-shows van meer en complexere factoren afhangen, maar dit is te complex voor deze scriptie. Mensen uit de

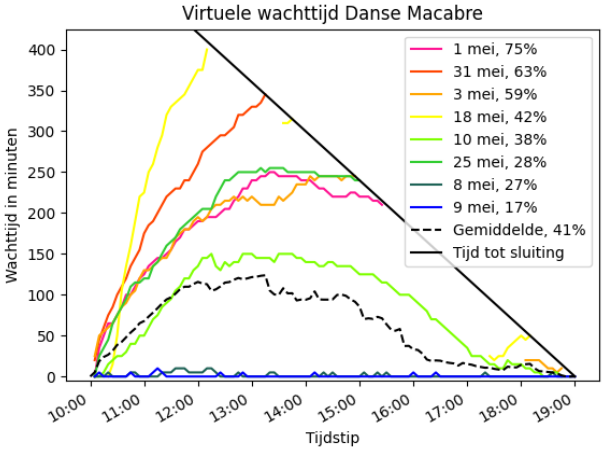
virtuele rij melden zich in minuut i van hun tijdslot met kans $\frac{1}{16}$.

We nemen aan dat de tijdsduur van iedere ronde op het kerkhof een normaalverdeling volgt met parameters μ_{show} en σ^2. We nemen aan dat de hoofdshow synchroon loopt met de show op het kerkhof, en dat één ronde op het kerkhof dus even lang duurt als de hoofdshow die op dat moment bezig is. Aan het begin van iedere ronde wordt iedereen die zich op dat moment bij de virtuele rij gemeld heeft toegelaten op het kerkhof. Deze mensen worden geteld, en aangevuld tot 108 mensen uit de reguliere rij. Hierbij worden er geen groepen opgesplitst. In de praktijk kan het aantal mensen op het kerkhof variëren, maar in ons model nemen we aan dat het exact 108 bezoekers zijn. Dit doen we omdat de koorbanken vrijwel altijd helemaal gevuld zullen zijn, door het plaatsen van bezoekers uit de single rider rij. We nemen daarom aan dat het altijd mogelijk is om precies 108 mensen op het kerkhof toe te laten, eventueel met toevoegingen uit de single rider rij. We hebben verder geen informatie over hoe vaak mensen voor de single rider rij kiezen en hoe lang deze wachtrij is, dus we verwerken de rij op deze manier in het model.

Wanneer de show op het kerkhof voorbij is, loopt de groep van maximaal 108 mensen door naar de abdij. Hier worden de bezoekers in de catacomben verdeeld over de koorbanken. Dit gebeurt terwijl de show van de voorgaande groep aan het spelen is, en heeft dus weer een tijdsduur die de normaalverdeling met parameters μ_{show} en σ^2 volgt.

Wanneer de voorgaande groep de attractie verlaten heeft, wordt de huidige groep toegelaten tot de attractie. Na nog een keer een normaalverdeelde tijdsduur met dezelfde parameters, verlaat de groep de attractie. Het model is samengevat in Figuur 15.

Bezoekers met een beperking worden niet meegenomen in het model. Een deel van deze mensen neemt plaats in de aparte filmzaal en neemt dus geen capaciteit van de attractie in, maar wordt wel meegerekend in de wachttijd. We gaan er echter vanuit dat het aantal mensen dat dit doet verwaarloosbaar is.



Figuur 16: De virtuele historische wachttijden bij Danse Macabre op verschillende dagen in mei 2025, met de bijbehorende drukteniveaus. De legenda is gesorteerd op drukteniveau.

10 Danse Macabre Data

Bij Danse Macabre hebben we, naast de problemen die in Hoofdstuk 3 worden genoemd, nog meer moeilijkheden. Deze problemen, en de gebruikte oplossingen, worden in dit hoofdstuk besproken.

Voor onze simulatie van het Danse Macabre model zullen we drie datasets gebruiken, namelijk DM, DMv en DMr. Vergelijkbaar met de datasets die we gebruikt hebben voor Droomvlucht, bevat DM wachttijden uit de periode dat er nog geen virtuele rij was, terwijl DMv en DMr respectievelijk de virtuele en reguliere wachttijden bevatten terwijl de virtuele rij wel actief was.

Een deel van de problemen die we met Danse Macabre hebben hangt samen met de nieuwigheid van de attractie: Danse Macabre is geopend op 31 oktober 2024, en op het moment van schrijven dus nog geen jaar oud [10]. De virtuele wachtrij is zonder aankondiging vooraf geopend op 22 april 2025, wat halverwege dit onderzoek is [18]. We kunnen dus niet, zoals bij Droomvlucht, gegevens van dezelfde periode een jaar eerder gebruiken. In dezelfde periode van het jaar hanteert de Efteling meestal dezelfde openingstijden, maar in verschillende periodes kan dit anders zijn. Zo opende het park in november 2024 regelmatig pas om 11 uur, terwijl dit in mei 2025 nooit voorkwam. Dit levert niet direct problemen op, maar kan het gemiddelde van de wachttijden in die periodes wel beïnvloeden.

De wachttijden van de reguliere rij halen we weer van de website queue-times.com [21]. Deze website heeft de virtuele wachtrij van Danse Macabre echter niet toe-

gevoegd. Hierdoor waren er geen data direct beschikbaar om voor onze dataset DMv te gebruiken. Er zijn wel websites die de huidige wachttijden weergeven, zoals Looopings [28], maar deze websites slaan de gegevens niet op. We hebben daarom zelf iedere 5 minuten de virtuele wachttijd van Danse Macabre gekopieerd van [28]. Dit doen we met behulp van een Windows Powershell script. Dit hebben we van 1 tot en met 31 mei gedaan. Op 4 mei is het niet gelukt om dit te doen, deze dag ontbreekt daarom in DMv. Omdat we daardoor niets hebben aan de reguliere wachttijden op 4 mei, hebben we deze dag ook uit DMr verwijderd.

Een paar van deze virtuele wachttijden zijn in Figuur 16 weergegeven, met hun drukteniveaus. Ook is de gemiddelde virtuele wachtrij van mei 2025 weergegeven, net als het gemiddelde drukteniveau (41%), en de tijd tot sluiting. We zien in deze figuur duidelijk terug dat de virtuele wachtrij sluit als de wachttijden te hoog worden. Ook zijn er een aantal dagen waarop de wachtrij aan het einde van de dag weer open ging. Ook bij heropeningen sluit de virtuele rij meestal enkele minuten voor sluitingstijd.

In mei 2025 zijn er 2 dagen waarop het park tot 8 uur open is, namelijk 29 en 30 mei. Op 29 mei was de virtuele wachtrij open tot en met 19:40, op 30 mei tot en met 15:45. Hierdoor hebben we in $DMv[gem]$ geen gemiddelde wachttijden voor de periode 19:45-20:00, er was immers geen dag waarop de attractie op deze tijdstippen open was. We hebben die gegevens echter wel nodig om de simulatie uit te kunnen voeren. Daarom plaatsen we nullen in $DMv[gem]$ van 19:45 tot en met 20:00.

Wat opvalt in Figuur 16 is dat er grote verschillen in de wachttijden zitten. Op sommige dagen liep de virtuele wachttijd op tot bijna 400 minuten, terwijl er ook dagen waren waar de wachttijd een groot deel van de tijd gelijk aan 0 minuten is. Ook valt op dat een rustigere dag niet automatisch betekent dat de wachttijden lager zijn. Zo heeft 18 mei een druktepercentage van 42%, maar vallen de virtuele wachttijden hier veel hoger uit dan op 1 mei, toen er een druktepercentage van 75% was. Deze verschillen kunnen ertoe leiden dat het moeilijker wordt om een accurate voorspelling te geven.

Aan de data die we kunnen gebruiken voor dataset DM kleven ook de nodige problemen. Zo zijn er dagen die niet representatief zijn voor een normale dag, zoals de periode dat de attractie net geopend was. Op de openingsdag (31 oktober 2024) haalde de attractie wachttijden van meer dan 4 uur, terwijl het die dag in het park met een druktepercentage van 71% niet extreem druk was [10, 24]. Ter vergelijking: op 30 december 2024 was het druktepercentage 98%, en kwamen de wachttijden bij Danse Macabre niet boven de 80 minuten uit [24, 21]. Om de grootste uitschieters weg te filteren, zullen we daarom de eerste 4 dagen van de attractie, donderdag 31 oktober tot en met zondag 3 november, niet opnemen in onze dataset. Op 31 december is de Efteling wegens oud en nieuw ook 's nachts open [8]. Ook deze dag zullen we daarom niet opnemen in DM.

Een ander probleem is dat op rustige dagen de attractie maar op halve capaciteit draait. Dit gebeurt sinds 1 januari 2025 [17]. Dit zorgt ervoor dat de wachttijd op deze dagen twee keer zo hoog wordt. Er zijn ook gevallen geweest waarop er halverwege de dag besloten werd om af te schalen naar halve capaciteit [20]. We weten echter niet op welke dagen dit het geval is geweest. Om dit probleem te omzeilen, zullen we voor dataset DM alleen dagen vóór 1 januari 2025 gebruiken. Concreet komt dit neer op 4 november tot en met 30 december. Voor de periode met de virtuele wachtrij hebben we echter geen manier om dit te omzeilen. Het is echter mogelijk dat de Efteling alleen besloot om de halve capaciteit te gebruiken in de periodes dat het standaard echt rustig is, zoals in januari en februari buiten de vakanties. In mei is het over het algemeen drukker dan in de wintermaanden. We nemen daarom aan dat de attractie heel mei op volle capaciteit gedraaid heeft.

De Efteling maakt graag gebruik van "easter eggs", verstopte grapjes. Een van deze easter eggs vindt men terug in de wachttijd: waar andere attracties hun wachttijd afronden op vijftallen, schrijft de Efteling 13 minuten in plaats van 10 minuten, als verwijzing naar het ongeluksgetal 13. De overige wachttijden komen wel voor. Dit gebeurt alleen bij de reguliere wachttijden, en niet bij de virtuele rij. Dit kan een vertekend beeld geven. We zullen daarom in DM en DMr de wachttijden die met 13 minuten staan aangegeven, vervangen door 10 minuten.

Sinds 15 mei heeft Danse Macabre een nieuw ritprogramma. Sinds dit moment is het aantal storingen bij de attractie drastisch toegenomen [16]. Op dagen met grote storingen volgen de wachttijden niet de patronen van een "normale"dag. Daarom zullen we dagen waarop de attractie in totaal meer dan een uur in storing is, niet meenemen in DMv en DMr. Dit is het geval op zes dagen, namelijk 11, 16, 17, 21, 22 en 24 mei. Ook in november en december waren er enkele grote storingen, namelijk op 4 en 30 november, en op 9, 10 en 11 december. Deze dagen zullen we dus ook niet meenemen in DM.

Samengevat hebben we dus de volgende datasets: DM bestaat uit de wachttijden van 5 november tot en met 29 november, 1 tot en met 9 december en 12 tot en met 30 december 2024, en de gemiddelde wachttijden op deze dagen. DMv bestaat uit de virtuele wachttijden op 1 tot en met 3 mei, 5 tot en met 10 mei, 12 tot en met 15 mei, 18 tot en met 20 mei, 23 mei en 25 tot en met 31 mei 2025, en de gemiddelde wachttijden op deze dagen, en DMr uit de reguliere wachttijden op dezelfde dagen als DMv.

Ronde	Groepsgroottes	Totaal # mensen	Showduur
1	2 5 7(3+4) 5 9(4+5) 5 5 4	42	6:22
2	4 4 1 4 2 2 3 3 4 1 3 6 5	42	8:15
3	3 5 3 4 2 8(4+4) 2 2	29	7:11
4	5 3 2 2 1 2 4 4 2 4 2 2	33	9:46
5	3 3 2 4 2 2 4 2 2 2 1 10(5+5) 1	38	8:39
6	4 1 3 3 2 2 4 3 3 4 4 3 1 2 6	45	9:22
7	2 2 4 4 4 2 2 1 2 3 6 1 2 2	37	8:04
8	2 2 3 6 2 4 2 2 2 3 4 3	35	8:13
Totaal		301	65:52

(a) Voorshow

Ronde	Showduur
1	7:48
2	6:56
3	7:41
4	6:51
5	7:28
6	7:09
7	7:46
8	7:58
Totaal	59:37

(b) Hoofdshow

Tabel 1: Gemeten showduur van voorshow en hoofdshow van Danse Macabre in minuten:seconden, en de getelde groepsgroottes bij de ingang van de virtuele wachtrij. De metingen bij de voorshow en de hoofdshow vonden niet gelijktijdig plaats.

11 Danse Macabre Parameters

In dit hoofdstuk leggen we uit welke waardes we voor alle parameters in het model kiezen. Een deel van de parameters is gebaseerd op gegevens die we zelf ter plekke gemeten hebben. Eerst bespreken we deze waardes. Vervolgens definiëren we een schatter voor de parameter β. Hierna definiëren we de overige parameters.

11.1 Eigen Metingen

Om een beter beeld te krijgen van de virtuele rij en de benodigde parameters, ben ik zelf naar Danse Macabre gegaan. Hier heb ik bijgehouden uit hoeveel mensen iedere groep die zich bij de ingang van de virtuele rij meldde bestaat. Deze gegevens zijn weergegeven in Tabel 1a. Omdat het bij het aanmelden voor de virtuele rij alleen mogelijk is om een groepsgrootte van 1 tot en met 6 op te geven, zijn dit de enige getallen die ik in de tabel heb gezet. Het gebeurde echter een paar keer dat grotere groepen zich bij de ingang meldden. Dit is mogelijk als meerdere mensen uit het gezelschap zich tegelijk inschrijven

Groepsgrootte	1	2	3	4	5	6	Totaal
Frequentie	9	34	18	24	10	4	99
Kans π	$\frac{9}{99}$	$\frac{34}{99}$	$\frac{18}{99}$	$\frac{24}{99}$	$\frac{10}{99}$	$\frac{4}{99}$	1

Tabel 2: Frequentie van groepsgroottes bij de virtuele rij van Danse Macabre, en de kans π_i dat een willekeurige groep de groepsgrootte i heeft.

	Eerste keer	Tweede keer
Tijdstip aangesloten	11:47	14:19
Voorspelde wachttijd	153 min	235 min
Voorspelde tijdslot	14:20-14:35	18:11-18:26
Werkelijke wachttijd	120 min	182 min
Werkelijk tijdslot	13:47-14:02	17:21-17:36
Te vroeg	33 min	53 min

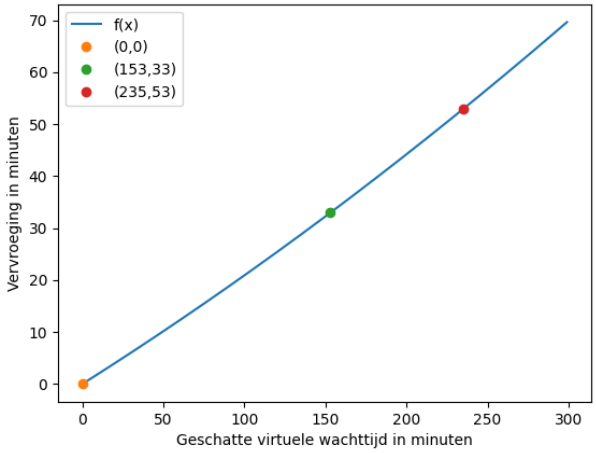
Tabel 3: Details van het opschuiven van tijdsloten van Danse Macabre.

voor de virtuele rij; zo kan een gezelschap van 10 mensen zich inschrijven als twee groepen van 5, of een groep van 6 en een groep van 4. Het was in deze situaties echter niet mogelijk om uit te vinden hoe deze grote gezelschappen zich daadwerkelijk hebben ingeschreven. Daarom hebben we ieder te groot gezelschap zelf opgedeeld in twee groepen, waarbij beide groepsgroottes zo dicht mogelijk bij elkaar zaten. Gezelschappen die bestonden uit meer dan 12 mensen zijn niet voorgekomen.

Ook hebben we bijgehouden hoe lang de voorshows en hoofdshows duurden. Om de duur van de hoofdshow te meten, hebben we bij de uitgang van de attractie de tijd tussen opeenvolgende groepen die naar buiten kwamen gemeten. Bij de voorshow hebben we de tijden tussen het sluiten van de deur naar het kerkhof gemeten. Bij zowel de voor- als de hoofdshow hebben we dus de duur van een complete cyclus van binnenkomst, show, verlaten van de ruimte en het resetten van de show gemeten, in plaats van de show zelf. De metingen aan de voorshow en de hoofdshow vonden niet gelijktijdig plaats. Het meten van de duur van de voorshow en het tellen van de groepsgroottes werd wel tegelijk gedaan. De resultaten zijn weergegeven in Tabel 1.

Bij iedere groepsgrootte hebben we geteld hoe vaak die is voorgekomen, en daarmee uitgerekend hoe groot de kans π_i op groepsgrootte i is. Dit is weergegeven in Tabel 2.

Ik ben ook twee keer zelf aangesloten in de virtuele rij, om te kijken hoeveel de tijdsloten opschoven. De eerste keer was dit 33 minuten op een voorspelde wachttijd van 153 minuten. De tweede keer was dit 53 minuten op een voorspelde wachttijd van 182 minuten. De complete gegevens staan in Tabel 3.



Figuur 17: Vervroeging van de virtuele wachttijd. Gemarkeerd zijn de punten waarvan de vervroeging bekend is.

11.2 No-shows

Om een uitdrukking voor β te vinden, moeten we weten hoeveel mensen niet komen opdagen voor hun tijdslot. Het is echter aannemelijk dat dit onder andere afhangt van zowel de virtuele wachttijd als de verschuiving van die wachttijd. Als de wachttijd erg hoog is, is het namelijk een stuk waarschijnlijker dat een groep zich op een andere plek in het park bevindt als zijn tijdslot aanbreekt, en dus niet op tijd is. En als het tijdslot veel naar voren verschuift, kunnen mensen die dit niet doorhebben hierdoor overvallen worden, en ook te laat zijn. Hoe groter de wachttijd en verschuiving, hoe groter dus ook het aantal mensen dat niet op komt dagen. Dit beïnvloedt echter weer opnieuw de verschuiving: als meer mensen niet op komen dagen, zullen de tijdsloten nog verder naar voren schuiven. Er is dus een recursief verband tussen de verschuiving en het aantal no-shows. Een uitdrukking vinden voor dit verband is echter te complex voor deze scriptie. Daarom zullen we een eenvoudigere uitdrukking proberen te vinden. Hiertoe proberen we eerst een verband te vinden tussen de virtuele wachttijd en de vervroeging. Dit doen we door gebruik te maken van de gegevens uit Tabel 3: We weten dat een voorspelde wachttijd van 153 minuten resulteerde in een vervroeging van 33 minuten, en dat bij een voorspelde wachttijd van 235 minuten het tijdslot 53 minuten vervroegde. Ook kan bij een wachttijd van 0 minuten het tijdslot niet vervroegen, men kan dan immers al meteen aansluiten.

We hebben dus 3 wachttijden waarvan we de vervroeging weten. We kunnen een kwadratische functie opstellen die door deze 3 punten heen gaat. Zij x de virtuele wachttijd op een bepaald tijdstip, dan is $f(x)$ het aantal minuten dat

het bijbehorende tijdslot zal vervroegen, met $f(x)$ gedefinieerd door

$$f(x) = 0.000120x^2 + 0.197x. \tag{9}$$

De grafiek van deze functie is weergegeven in Figuur 17. Merk op dat deze functie bijna lineair lijkt, maar dat niet is. Een lineaire functie door de punten (153,33) en (235,53) zou namelijk de $x - as$ snijden in het punt (14,4 , 0), wat zou betekenen dat voor wachttijden van 14 minuten en lager de tijdsloten juist worden uitgesteld. We gaan er vanuit dat dit niet gebeurt, aangezien er niet meer mensen kunnen aansluiten dan er zich al hebben aangemeld.

Met de gegevens die we verzameld hebben, kunnen we ook een inschatting maken van het aantal mensen dat overeenkomt met 1 minuut aan virtuele wachttijd, zeg M. We weten namelijk dat in 65 minuten en 52 seconden er 99 groepen het kerkhof op gelaten zijn (Tabel 1a), wat neerkomt op $M \approx 1,5$ groepen per minuut. We definiëren vervolgens

$$g(x) = \frac{f(x)}{x - f(x)} M \tag{10}$$

als het aantal groepen dat in één minuut niet op komt dagen, als hun voorspelde wachttijd x was. Merk op dat $0 \leq \frac{f(x)}{x-f(x)} \leq 1$ voor alle relevante wachttijden, en dat $g(x)$ dus niet negatief of groter dan M zal worden. Vervolgens definiëren we $\beta(0) = 0$ en

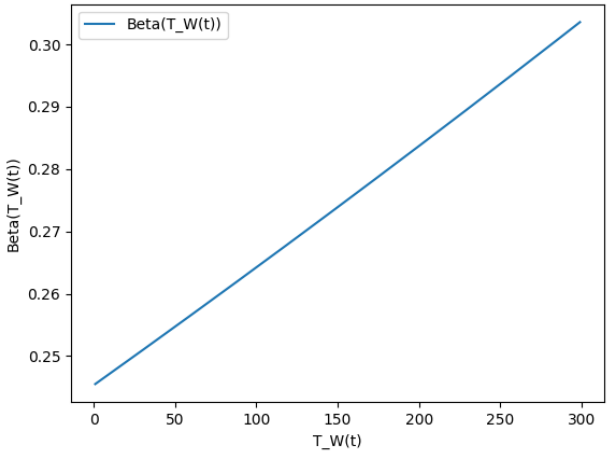
$$\beta(T_W(t)) = \frac{g(T_W(t))}{M} = \frac{f(T_W(t))}{T_W(t) - f(T_W(t))} \tag{11}$$

voor $T_W(t) > 0$ als de kans dat een groep niet op tijd komt opdagen.

11.3 Overige Parameters

We introduceren verder de volgende parameters.

- Zij T_S de gemiddelde duur van één showcyclus (de tijd tussen de start van twee voor- of hoofdshows). Deze berekenen we door de gemiddelde showduur van Tabel 1 te bepalen.
- Zij M_K het aantal mensen dat gemiddeld per voorshow het kerkhof op gaat vanuit de virtuele rij. Dit kunnen we berekenen met de gegevens uit Tabel 2.
- Zij G_{gem} de gemiddelde groepsgrootte. Dit kunnen we berekenen met de gegevens uit Tabel 2.
- Definieer $\mu_{gem} = \frac{108}{G_{gem} \cdot T_S}$ als het aantal groepen dat gemiddeld per minuut door de attractie verwerkt wordt.
- Definieer $\mu_v = \frac{M_K}{T_S}$ als het aantal mensen dat gemiddeld per minuut het kerkhof op gelaten wordt vanuit de virtuele rij.



Figuur 18: De kans op no-shows als functie van de virtuele wachttijd $T_W(t)$ op tijdstip t.

- Definieer $\mu_r = \frac{108}{T_S} - \mu_v$ als het aantal mensen dat gemiddeld per minuut het kerkhof op gelaten wordt vanuit de reguliere rij.
- Definieer $\gamma = \frac{M_K}{108}$ als de gemiddelde fractie van mensen op het kerkhof die uit de virtuele rij afkomstig zijn.

Omdat we in de simulatie iedere minuut de wachttijd uitrekenen, drukken we deze parameters uit in mensen of groepen per minuut. We zullen μ_{gem} gebruiken voor μ in Vergelijking 1. Om de wachttijden in de virtuele rij te berekenen gebruiken we μ_v, en voor de wachttijden in de reguliere rij gebruiken we μ_r. Dit wordt uitgebreider beschreven in Hoofdstuk 12. We gebruiken in de simulatie dus niet de normaalverdeling met parameters μ en σ^2. Dit kan echter wel van pas komen wanneer men de wachttijden van individuele groepen wil simuleren, in plaats van gemiddelde wachttijden. In dat geval is het mogelijk om μ_{show} en σ^2 te schatten met behulp van de gegevens in Tabel 1, door het steekproefgemiddelde en de steekproefvariantie te berekenen [2]. Wij zullen deze berekening niet uitvoeren.

We definiëren onze schatters voor $\alpha(t)$ en $\lambda(t)$ op dezelfde manier als in Hoofdstuk 7. Hierbij vervangen we de datasets DR, DRv en DRr door respectievelijk DM, DMv en DMr. We gebruiken zowel Methode 1 als Methode 2 voor de aankomstsnelheid. Bij het bepalen van $\hat{\lambda}_{DMv[gem],s[d]}(t)$ en $\hat{\lambda}_{DMr[gem],s[d]}(t)$ gebruiken we respectievelijk $\gamma\mu_{gem}$ en $(1-\gamma)\mu_{gem}$ in plaats van μ. Voor het bepalen van $\hat{\lambda}_{2,DM[gem],s[d]}(t)$ gebruiken we μ_{gem} in plaats van μ. Dit geeft twee schatters voor de aankomstsnelheid, $\hat{\lambda}_1^{tot}(t)$ en $\hat{\lambda}_2^{tot}(t)$, zoals in respectievelijk Vergelijking 6 en Vergelijking 7, en een schatter $\hat{\alpha}(t)$ zoals in Vergelijking 8.

12 Danse Macabre Simulatie

In dit hoofdstuk voeren we de simulatie van het Danse Macabre model uit. Eerst leggen we uit hoe deze simulatie in zijn werk ging. Vervolgens bespreken we de resultaten.

12.1 Simulatie

Om de wachttijden zoals die worden weergegeven in de Efteling-app te simuleren, is het niet nodig om het complete model te simuleren. We zijn immers op zoek naar de gemiddelde wachttijden, zoals de Efteling deze ook in haar app laat zien, en niet de tijden die individuele bezoekers moeten wachten. Daarom hoeven we bijvoorbeeld niet aan iedere groep een individueel tijdslot toe te kennen, of de showtijden te bepalen met de normaalverdeling.

Net als bij Droomvlucht zijn er verschillende manieren waarop we de simulatie kunnen uitvoeren. Voor Methode 1 gebruiken we de datasets DMv en DMr, met respectievelijk de virtuele en reguliere wachttijden van bepaalde dagen in mei 2025. Bij Methode 2 gebruiken we de DM dataset, met data uit de tijd dat er nog geen virtuele wachttijd was. Bij beide methodes hebben we sowieso zowel DMv als DMr nodig om de waardes van parameter $\alpha(t)$ te bepalen, zoals gedefinieerd in Vergelijking 8. Verder gebruiken we de parameters $M_K = 37.625$, $T_S = 7.843$ en $G_{gem} = 3$, en μ_{gem}, μ_v, μ_r en γ zoals gedefinieerd in Hoofdstuk 11.3.

De simulatie volgt grotendeels dezelfde stappen als die van Droomvlucht. We kiezen eerst een dag die we simuleren, en laden de benodigde datasets. Vervolgens worden op iedere minuut $n \in \{0, 1, \ldots, T_N\}$ de benodigde waardes van de schatters van $\lambda(t)$ berekend, zowel voor de berekening van de schatter van $\alpha(t)$ als voor de simulatie. Hiervoor nemen we $\mu = \mu_{gem}$. We berekenen $\hat{\alpha}(n)$, en voeren vervolgens de simulatie zelf uit. Deze simulatie volgt op ieder tijdstip $n \in \{0, 1, \ldots, T_N\}$ de volgende stappen.

1. Er wordt een Poissonverdeling met parameter $\lambda_{gem,tot}(n)$ ($\hat{\lambda}_1^{tot}(n)$, Methode 1) of $\lambda_{aankomst}(n)$ ($\hat{\lambda}_2^{tot}(n)$, Methode 2) getrokken.
2. Voor elke gesimuleerde groep wordt groepsgrootte i toegekend met kans π_i, en wordt de keuze tussen de virtuele en de reguliere rij gemaakt met behulp van een Bernoulliverdeling met kans $\hat{\alpha}(n)$.
3. Er wordt geteld hoeveel mensen er iedere minuut in de virtuele rij bijkomen.
4. Het aantal wachtenden in de virtuele rij op tijdstip $n + 1$ is het aantal mensen dat op tijdstip n was aangesloten, plus het aantal mensen dat in deze minuut erbij kwam, min μ_v mensen.
5. Het aantal wachtenden in de reguliere rij op tijdstip $n + 1$ is het aantal mensen dat op tijdstip n was aangesloten, plus het aantal mensen dat in

deze minuut erbij kwam, min μ_r mensen.

6. De gesimuleerde virtuele wachttijd op tijdstip $n+1$ is vervolgens het aantal virtuele wachtenden gedeeld door μ_v.
7. De gesimuleerde reguliere wachttijd op tijdstip $n+1$ is het aantal reguliere wachtenden gedeeld door μ_r, plus 13.
8. Als de virtuele wachttijd hoger wordt dan de tijd tot sluitingstijd, slaan we het tijdstip waarop dit gebeurt op in de float afbreektijdstip.

Wanneer we meer dan één simulatieronde uitvoeren, nemen we weer de gemiddelde wachttijden en het gemiddelde afbreektijdstip. De virtuele wachttijden worden na dit gemiddelde afbreektijdstip afgekapt. Vervolgens wordt hier Vergelijking 9 op toegepast, en worden de wachttijden geplot in een grafiek.

Merk op dat we wel de virtuele rij sluiten wanneer deze wachttijden te hoog worden, maar deze niet later weer opnieuw openen. Dit doen we omdat we niet genoeg informatie hebben over wanneer deze rij heropend wordt.

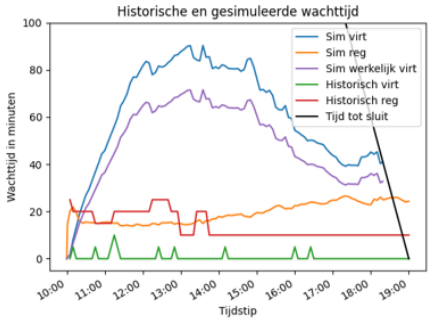
We tellen 13 minuten bij de gesimuleerde reguliere wachttijd op, omdat er altijd wachttijd is, ook als er niemand in de rij staat. Men moet namelijk sowieso wachten op het kerkhof tot de voorshow voorbij is, en vervolgens ook in de catacomben van de abdij tot men daadwerkelijk kan plaatsnemen en de show begint. Bij de ingang van de virtuele wachtrij (Poort 2) staat een bord waarop vermeld wordt dat de wachttijd vanaf dat punt nog ongeveer 13 minuten is [13]. Dit zal dus voor de reguliere rij ook gelden vanaf Poort 1.

De gebruikte code is weergegeven in Bijlage D en Bijlage E. Ook hier kunnen we Methode 0 gebruiken door $DMv[d]$ en $DMr[d]$ te gebruiken in plaats van $DMv[gem]$ en $DMr[gem]$.

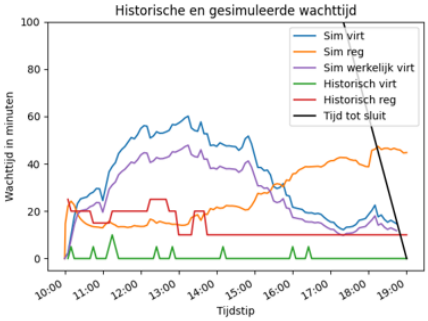
12.2 Resultaten

In Figuur 19 worden de resultaten van de simulatie getoond voor Methode 1 en Methode 2 op drie dagen met erg verschillende drukteniveaus: een rustige, een normale en een drukke dag. Op iedere dag en met iedere methode is de simulatie uitgevoerd. We geven ook weer wat de wachttijd inclusief de vervroeging zou zijn. Dit doen we door op ieder tijdstip de vervroeging te berekenen met behulp van Vergelijking 9, en deze af te trekken van de virtuele wachttijd. We noemen dit de werkelijke virtuele wachttijd. Dit is dus iets anders dan de historische wachttijd, die bestaat uit de verzamelde data van de corresponderende dag. Iedere simulatie is 10.000 keer uitgevoerd.

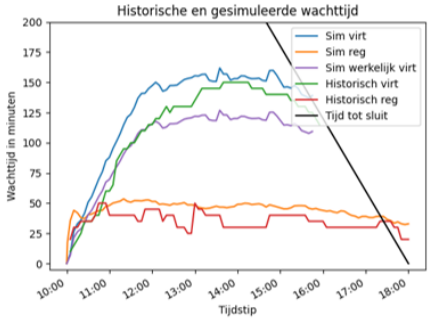
Als rustige dag hebben we 9 mei gekozen. Op deze dag was er een drukte-percentage van 17%, en was het park open van 10:00 tot 19:00. We zien dat de gesimuleerde virtuele rij (zowel de normale simulatie als de voor no-shows gecorrigeerde simulatie) een stuk hoger uitvalt dan de historische wachttijd, die een groot deel van de tijd gelijk is aan 0 minuten. Beide methodes volgen de reguliere rij wat beter, al ontstaan er in de tweede helft van de dag ook grotere



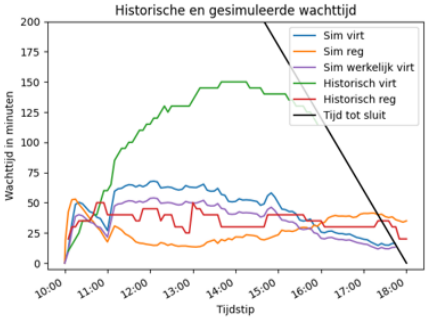
(a) 9 mei, Methode 1, 10.000 keer uitgevoerd. Druktepercentage 17%.



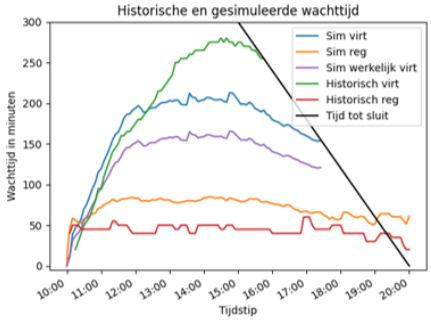
(b) 9 mei, Methode 2, 10.000 keer uitgevoerd. Druktepercentage 17%.



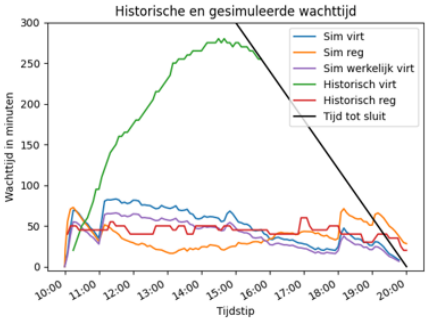
(c) 26 mei, Methode 1, 10.000 keer uitgevoerd. Druktepercentage 60%.



(d) 26 mei, Methode 2, 10.000 keer uitgevoerd. Druktepercentage 60%.



(e) 30 mei, Methode 1, 10.000 keer uitgevoerd. Druktepercentage 88%.



(f) 30 mei, Methode 2, 10.000 keer uitgevoerd. Druktepercentage 88%.

Figuur 19: Resultaten simulatie Danse Macabre. De "werkelijke" virtuele wachttijd is hier de gesimuleerde virtuele wachttijd minus de berekende vervroeging.

afwijkingen. Methode 1 lijkt het beste te kloppen voor de reguliere rij. Methode 2 zit voor de virtuele rij het meest in de buur, maar valt veel te hoog uit.

Voor de normale dag hebben we 26 mei genomen, met een druktepercentage van 60%. De openingstijden op deze dag waren 10 uur tot 6 uur. Methode 1 geeft hier voor zowel de reguliere als de virtuele wachttijden de beste resultaten, met voorspellingen die dicht in de buurt van de werkelijkheid zitten. Methode 2 blijft bij de virtuele wachttijden enorm achter bij de historische wachttijd. Ook komt hier rond 11 uur een opvallende dip voor. De werkelijke virtuele wachttijd ligt een flink stuk lager dan de originele voorspelling, met op sommige tijdstippen verschillen van ongeveer een half uur.

Hier zien we dat bij Methode 2 de virtuele wachttijden enorm achterblijven bij de historische wachttijd. Ook komt hier rond 11 uur een opvallende dip voor. Methode 1 doet dit een stuk beter. Wel gaat de reguliere wachttijd bij deze methode aan het einde van de dag weer veel te veel omhoog. Bij Methode 2 gaat dat beter, maar daar laat de reguliere simulatie weer rond 11 uur een dipje zien, juist terwijl de historische wachttijd daar een piek vertoont. De gecorrigeerde virtuele wachttijd valt een stuk lager uit dan de normale simulatie.

We hebben 30 mei gekozen voor de drukke dag, met openingstijden van 10 uur tot en met 8 uur. Op deze dag was er een druktepercentage van 88%. Methode 1 is hier nog steeds het beste, maar ligt bij de virtuele wachttijden meer dan een uur onder de historische wachttijd. Dit is echter een stuk beter dan Methode 2, die niet in de buurt komt bij de historische virtuele wachttijd. De reguliere wachttijd is redelijk, maar vertoont ook weer de opvallende piek tussen 10 en 11 uur. Ook om 6 uur is er hier een piek zichtbaar.

Hier blijft Methode 2 weer enorm achter bij de virtuele wachttijden. Methode 1 loopt op het hoogtepunt ook een flink stuk onder de historische wachttijd, maar komt veel meer in de buurt dan Methode 2. Ook hier is bij Methode 2 weer een dal om 11 uur en een piek om 6 uur zichtbaar.

De pieken en dalen aan het begin van de simulaties met Methode 2 zijn mogelijk te verklaren door de openingstijden van de Efteling in november en december 2024. Op een deel van deze dagen opende het park om 10 uur, en de rest van de periode pas om 11 uur. Het kan op deze dagen dus op verschillende momenten druk worden bij de attractie. Door de gemiddelde wachttijden van deze dagen te nemen, kunnen er op sommige plekken opvallende pieken of dalen ontstaan.

Wat bij de virtuele wachttijden opvalt is dat Methode 2 in alle simulaties amper boven de 50 minuten uit lijkt te komen, terwijl er meerdere dagen zijn waarop de historische wachttijden vele malen hoger zijn. Het lijkt erop dat het aankomstpatroon bij Danse Macabre tussen november en december 2024 en mei 2025 flink veranderd is, en er meer mensen in de (virtuele) rij gaan staan. Het ligt voor de hand dat dit door de toevoeging van de virtuele rij komt, die blijkbaar meer mensen trekt.

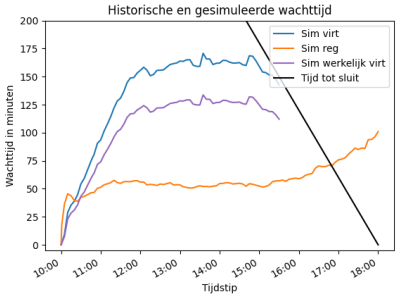
Op rustige dagen zijn zowel Methode 1 als Methode 2 voor de virtuele rij erg

slecht, en zou zelfs een schatting die constant 0 minuten is beter zijn. Voor drukkere dagen komt Methode 1 een stuk dichter in de buurt dan Methode 2. Methode 1 valt voor de reguliere wachttijden op drukke dagen juist te hoog uit, en schiet aan het einde ook onverklaarbaar omhoog. Wel volgt dit een logischer patroon dan de pieken en dalen die als gevolg van openings- en sluitingstijden voorkomen bij Methode 2.

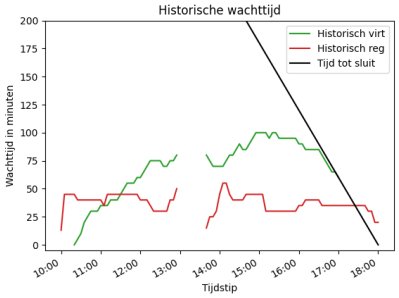
We zien in Figuur 16 dat de wachttijden nogal extreem van elkaar verschillen; een deel van de wachttijden is bijna overal gelijk aan 0, terwijl de wachttijden op andere dagen erg hoog zijn. De simulatie van zo'n extreme dag zal dus niet erg realistisch zijn, omdat we deze baseren op de gemiddelde wachttijden. Het ligt dus voor de hand dat op een dag met extreem lage wachttijden de simulatie te hoog uitpakt, terwijl op een dag met extreem hoge wachttijden de simulatie te laag uitvalt. We verwachten dat de simulatie het beste werkt op dagen met een middelmatige drukte.

In alle simulaties zien we een groot verschil tussen de werkelijke virtuele wachttijd en de originele virtuele simulatie. De vervroeging kan dus behoorlijk oplopen.

Samengevat lijkt Methode 1 in alle drie de situaties beter te zijn. Wel valt de virtuele simulatie op rustige dagen veel te hoog uit, en op drukke dagen juist flink te laag. Bij de simulatie van Droomvlucht leek Methode 2 over het algemeen juist beter te zijn, dus dit is opvallend. Vooral bij de virtuele Droomvluchtrij was Methode 2 beter dan Methode 1, terwijl bij Danse Macabre Methode 2 juist erg slechte resultaten geeft voor de virtuele rij.



(a) De gesimuleerde wachttijden bij Danse Macabre voor 24 juni 2025, 10.000 keer uitgevoerd.



(b) Historische wachttijden 24 juni 2025.

Figuur 20: Gesimuleerde en werkelijke wachttijden 24 juni 2025. Voorspeld drukteniveau: 65%.

13 Test

Om te testen of het model in de praktijk gebruikt kan worden om een inschatting te maken van de wachttijden bij Danse Macabre, ben ik op dinsdag 24 juni naar de Efteling gegaan. De dag van tevoren was het geschatte druktepercentage voor 24 juni 65% [24, geraadpleegd op 24-06-2025]. Achteraf is dit bijgesteld naar 58% [24, geraadpleegd op 27-06-2025]. Omdat we gezien hebben dat Methode 1 betere resultaten lijkt te geven dan Methode 2, kiezen we Methode 1.

In Figuur 20a zijn de voorspelde wachttijden te zien voor 24 juni. Hierbij hebben we dus een druktepercentage van 65% en de datasets DRv en DRr genomen, en Methode 1 gebruikt. In Figuur 20b zijn de werkelijke wachttijden te zien. Tussen 12:55 en 13:40 is er een storing geweest. Wat opvalt is dat de gesimuleerde virtuele wachttijden veel te hoog uitvallen. Dit is opvallend, omdat we op basis van het drukteniveau een soortgelijke situatie als in Figuur 19c verwachtten. De historische virtuele wachttijden vallen echter lager uit. De gesimuleerde reguliere wachttijden lijken beter te kloppen, al loopt deze te ver op aan het einde van de dag.

We zijn drie keer aangesloten in de virtuele wachtrij. De details van de tijdsloten en de voorspelde vervroegingen staan in Tabel 4. Hier valt op dat er bij de eerste ronde sprake was van een negatieve vervroeging, het tijdslot werd dus later in plaats van eerder. Dit is niet iets dat we eerder voor hebben zien komen. Verder vallen de voorspelde vervroegingen veel hoger uit dan de werkelijke vervroegingen. Blijkbaar treden grote vervroegingen pas op wanneer de virtuele wachttijden een stuk hoger zijn. Een andere mogelijke verklaring is dat de virtuele wachtrij in de week voor ons bezoek heeft uitgestaan wegens werkzaamheden [15]. Het is mogelijk dat de Efteling zelf een uitdrukking heeft gevonden voor

	Eerste keer	Tweede keer	Derde keer
Tijdstip aangesloten	10:29	11:10	14:10
Voorspelde wachttijd Efteling x	10 min	36 min	75 min
Voorspelde tijdslot	10:39-10:54	11:46-12:01	15:25-15:40
Werkelijk tijdslot	10:40-10:55	11:44-11:59	15:23-15:38
Vervroeging	-1 min	2 min	2 min
Voorspelde vervroeging $f(x)$	2 min	7 min	15 min

Tabel 4: Het gedrag van de tijdsloten bij Danse Macabre op 24 juni 2025.

de no-shows, en die voorspelling meeneemt in de voorspelde tijdsloten. Dit kan ook de negatieve vervroeging verklaren.

14 Discussie

Het valt op dat Methode 2 bij de Droomvlucht-simulatie de beste resultaten leek te geven, terwijl Methode 1 bij Danse Macabre veruit de beste was. Men zou misschien eerder het tegenovergestelde verwachten: bij Droomvlucht leek het voor de hand liggend dat de data van de gesimuleerde periode de beste resultaten zou geven, terwijl bij Danse Macabre er misschien nog niet genoeg data beschikbaar was om Methode 1 een succes te maken. Het tegendeel blijkt echter waar.

Bij Droomvlucht leken de virtuele en reguliere wachttijden een enigszins vergelijkbaar patroon te volgen. Bij Danse Macabre worden de virtuele wachttijden echter zo hoog, dat deze een compleet ander patroon hebben dan de reguliere wachttijden. Dit kan een van de redenen zijn dat Methode 2 bij de Danse Macabre simulatie zo slecht presteerde: als het aankomstpatroon van de reguliere rij in november en december 2024 vergelijkbaar is met dat in mei 2025, komt dit niet bij de virtuele aankomststromen in de buurt.

Er is een hoop ruimte voor verbetering bij onze simulaties, ook bij de beste methode. Bij sommige dagen komen de resultaten redelijk in de buurt van de werkelijkheid, maar op andere dagen zitten ze er volledig naast. Hoewel het in de praktijk niet mogelijk zal zijn om perfecte voorspellingen te maken, zijn er wel veel zaken die verbeterd of grondiger onderzocht kunnen worden.

14.1 Data

Voor de simulaties van toekomstige dagen gebruiken we de druktepercentages van queue-times.com [24]. Deze percentages zijn echter ook voorspellingen. We weten niet precies hoe deze percentages tot stand komen, en hebben geen onderzoek gedaan naar hoe betrouwbaar deze voorspellingen zijn. Het is dus aan te raden om dit nader te onderzoeken.

Bij de data van de Droomvlucht-simulatie zaten, net als bij Danse Macabre, ook dagen met grote storingen. Bij de Droomvlucht-simulatie hebben we die er echter nog niet uit gefilterd. Dit hebben we bij Danse Macabre wel gedaan, omdat deze storingen na de introductie van het nieuwe ritprogramma opvallend vaak in korte tijd voorkwamen, terwijl de grote storingen bij Droomvlucht slechts incidenteel plaats vonden.

Bij Methode 1 en Methode 2 maken we gebruik van de gemiddelde wachttijden en gemiddelde drukte over een bepaalde periode. Het is echter ook mogelijk om een gewogen gemiddelde wachttijd te gebruiken. Het zou interessant zijn om te onderzoeken wat het effect hiervan op de simulaties is.

14.2 Parameters

Zoals eerder gezegd zal de kans α dat een groep voor de virtuele rij van meer factoren dan alleen tijd afhangen. Denk aan de reguliere en virtuele wachttijden

op dat moment, persoonlijke voorkeur, en de leeftijden van de groepsleden. Bij Danse Macabre zouden zelfs de weersomstandigheden een rol kunnen spelen, omdat deze wachtrij niet overdekt is. Het is dus aan te raden om meer onderzoek te doen naar de factoren die deze kans α beïnvloeden, en om een betere uitdrukking voor deze kans te vinden.

In de simulaties is er nog een extra probleem met α: de schatter $\hat{\alpha}(t)$ hangt af van de waarde van γ. Dit komt doordat we $\hat{\lambda}_{DRv[gem]}(t)$ en $\hat{\lambda}_{DRr[gem]}(t)$ gedefinieerd hebben aan de hand van γ, net als de corresponderende schatters bij de Danse Macabre simulatie. Dit verband zorgt ervoor dat de simulatieresultaten nauwelijks veranderen wanneer we γ aanpassen. In werkelijkheid zal dit niet zo zijn: als er meer voertuigen worden toegewezen aan de virtuele rij, betekent dit niet automatisch dat er ook meer mensen voor de virtuele rij kiezen.

Hoewel we zelf gegevens hebben verzameld bij Danse Macabre, was dit te weinig data om zekerheid te hebben over de waardes van bepaalde parameters. Het is daarom aan te raden om deze metingen nog een keer uit te voeren met een grotere steekproefgrootte, zodat dit nauwkeurigere resultaten oplevert voor bijvoorbeeld de showduur en de groepssamenstelling.

Zoals eerder gezegd is het goed mogelijk dat de Efteling haar strategiën en parameters van de virtuele rijen aanpast. Dit kan als gevolg hebben dat de waardes van de parameters in onze simulatie opeens niet meer kloppen. Wanneer de Efteling zelf mee zou werken aan eventueel toekomstig onderzoek, kunnen werkelijke gegevens over bijvoorbeeld de no-shows of de waarde van γ gebruikt worden. Dit kan leiden tot accuratere voorspellingen, die de Efteling zelf zou kunnen gebruiken in haar app.

14.3 Model

Volgens de data van Droomvlucht kwam het regelmatig voor dat de wachttijd 0 minuten bedroeg. Wanneer de wachttijd gelijk is aan 0, neemt dit model aan dat er precies evenveel mensen aankomen als er geholpen worden. Bij Droomvlucht is het in de praktijk echter ook mogelijk, en waarschijnlijk, dat er minder mensen aankomen, en er dus soms lege voertuigen vertrekken. Doordat de simulatie stochastisch is, is het daarom goed mogelijk dat het aantal gesimuleerde mensen te hoog uitvalt, en er dus extra wachttijd ontstaat. Dit zorgt ervoor dat de simulatie te hoog uitvalt. In de toekomst kan dit dus beter op een andere manier gedefinieerd worden. Hetzelfde geldt voor het model van Danse Macabre.

Bij de test van het Danse Macabre model bleek dat de vervroeging veel lager uitviel dan verwacht. Ook werd een van de toegewezen tijdsloten zelfs een minuut vertraagd, wat we hiervoor niet hebben zien gebeuren. Zoals gezegd is een mogelijke verklaring hiervoor dat de Efteling zelf de no-shows in de virtuele wachttijden heeft verwerkt, waardoor ze niet meer hoeven te verschuiven. Wanneer er minder no-shows zijn dan verwacht, zou dit als gevolg kunnen hebben dat de tijdsloten inderdaad later worden. Het is echter ook mogelijk dat dit toeval is: zowel de test zelf als de gegevens waarmee de formule voor vervroeging

is opgesteld zijn een momentopname. Hoe dan ook is het aan te raden om meer onderzoek te doen naar deze verschuivingen.

Ook bleek tijdens de test dat het model vooral voor de virtuele wachtrij nog niet zo goed werkt als we zouden willen. We verwachten dat dit voor een groot deel te maken heeft met de inconsistentie van de drukteniveaus in Figuur 16. Een lager drukteniveau betekent blijkbaar niet automatisch een rustigere dag. Het is daarom een goed idee om te proberen een verband te vinden tussen de algemene drukteniveaus en de drukte bij Danse Macabre.

We zien in Figuur 16 dat er een aantal dagen zijn waarop de virtuele wachttijden bij Danse Macabre voor een groot deel van de dag gelijk aan nul zijn. Als het lukt om een grens te bepalen vanaf welke drukte dit gebeurt, is de beste voorspelling waarschijnlijk die waarop de virtuele wachttijd standaard op nul staat.

We hebben de simulaties vaak maar voor enkele dagen uitgevoerd. Dit is het geval omdat deze scriptie voor een groot deel bestond uit het opstellen van het model en het vinden van data, waardoor er geen ruimte was om de resultaten heel uitgebreid te analyseren. Wanneer hier in het vervolg meer tijd voor genomen wordt, kan dit meer inzicht geven in hoe goed of slecht de modellen zijn.

14.4 Vervolgonderzoek

Waar we in deze scriptie niet naar gekeken hebben, is het verband tussen de virtuele en reguliere wachttijden. Figuur 20b geeft echter de indruk dat de virtuele wachttijd stijgt als de reguliere wachttijd daalt, en andersom. Het kan interessant zijn om te kijken of er inderdaad een verband tussen deze wachttijden te ontdekken is.

De Efteling geeft in haar app steeds een schatting van de wachttijd op dit moment. Wij hebben in dit onderzoek geprobeerd om deze schattingen te reconstrueren. Het zou echter ook interessant zijn om te onderzoeken of de werkelijke wachttijden van individuele groepen overeenkomen met deze schattingen. Hierbij kunnen individuele groepen gesimuleerd worden, die een verschuivend tijdslot toegewezen krijgen. Dit kan ook meer inzicht geven in het voorkomen van de verschuivingen. Hiervoor is het aan te raden om meer onderzoek te doen naar hoe groot de verschuivingen vaak zijn, en kan het ook interessant zijn om toch de groepsgroottes bij Droomvlucht te bekijken.

Hoewel dit niet direct voor de hand ligt bij een wiskundescriptie, zou het ook erg interessant zijn om onderzoek te doen naar de psychologische kant van virtuele wachtrijen. Inzicht krijgen in de beweegredenen van bezoekers om wel of niet voor de virtuele wachtrij te kiezen, kan ook voor een beter model zorgen.

Met name het Danse Macabre model combineert een aantal interessante aspecten die nog niet veel onderzocht zijn, zoals het principe van tijdsloten en no-shows, en de manier waarop Rij 2 en Rij 3 worden samengevoegd. Dit maakt het erg interessant om dit model theoretisch te analyseren.

De Efteling heeft inmiddels aangekondigd dat in juli en augustus 2025 de virtuele wachtrij bij Droomvlucht zal terugkeren [19]. Voor verder onderzoek zou het daarom erg interessant zijn om dat model te perfectioneren, zodat dit in de praktijk gebruikt kan worden. Dit kan een mooie toevoeging zijn op de voorspellingen van wachttijden die de Efteling bij andere attracties al laat zien in de Efteling-app.

Referenties

[1] Henk C. Tijms. *A First Course in Stochastic Models*. 2de ed. John Wiley & Sons, 2003.

[2] F. Bijma, M. Jonker en A. Van der Vaart. *Inleiding in de Statistiek*. 3de ed. Epsilon uitgaven, 2018.

[3] Looopings. *Efteling gaat overstag: attracties voortaan open tot sluitingstijd park*. Mrt 2018. URL: https://www.looopings.nl/weblog/9398/Efteling-gaat-overstag-attracties-voortaan-open-tot-sluitingstijd-park.html.

[4] C. Vuik e.a. *Numerical Methods for Ordinary Differential Equations*. 2de ed. Delft Academic Press / VSSD, 2018. Hfdstk. 2: Interpolation.

[5] Looopings. *Verblijfsgasten Efteling mogen half uur eerder in acht grote attracties*. Jun 2019. URL: https://www.looopings.nl/weblog/12393/Verblijfsgasten-Efteling-mogen-half-uur-eerder-in-acht-grote-attracties.html.

[6] Looopings. *Efteling-app voorspelt nu wachttijden bij attracties voor de rest van de dag*. Mei 2022. URL: https://www.looopings.nl/weblog/18933/Efteling-app-voorspelt-nu-wachttijden-bij-attracties-voor-de-rest-van-de-dag.html.

[7] Efteling. *Efteling onthult invulling themagebied rond Danse Macabre*. Mrt 2023. URL: https://www.efteling.com/nl/pers/efteling-onthult-invulling-themagebied-rond-danse-macabre/.

[8] Looopings. *Efteling maakt programma oudejaarsavond bekend: acrobatiek, bingo met grootmoeder en aftellen met Pardoes*. Nov 2024. URL: https://www.looopings.nl/weblog/27292/Efteling-maakt-programma-oudejaarsavond-bekend-acrobatiek--bingo-moet-grootmoeder-en-aftellen-met-Pardoes.html.

[9] Looopings. *Efteling wil wachttijden aanpakken: in 2030 maximaal twintig minuten wachten*. Mei 2024. URL: https://www.looopings.nl/weblog/25654/Efteling-wil-wachttijden-aanpakken-in-2030-maximaal-twintig-minuten-wachten.html.

[10] Looopings. *Video: wachtrij voor nieuwe Efteling-attractie Danse Macabre begint bij Aquanura-vijver*. Okt 2024. URL: https://www.looopings.nl/weblog/27183/Video-wachtrij-voor-nieuwe-Efteling-attractie-Danse-Macabre-begint-bij-de-Aquanura-vijver.html.

[11] Looopings. *Zo werkt de nieuwe virtuele wachtrij bij Droomvlucht in de Efteling*. Jul 2024. URL: https://www.looopings.nl/weblog/26051/Zo-werkt-de-nieuwe-virtuele-wachtrij-bij-Droomvlucht-in-de-Efteling.html.

[12] Paul Sprangers en Tim Hinssen. *399: Persvak*. Kleine Boodschap, jul 2024. URL: https://kleineboodschap.com/afleveringen/2024/7/15/399-persvak.

[13] Eftelwesley. *[#Efteling] Nieuw: Virtuele wachtrij bij Danse Macabre*. Apr 2025. URL: https://www.youtube.com/watch?v=w-BIxm5XylM.

[14] Adviesbureau Ginder. *Top 50 dagrecreatie over 2024 is weer verschenen.* Mei 2025. URL: https://pretwerk.nl/recreatie-actueel/top-50-dagrecreatie-over-2024-is-weer-verschenen/93505/.

[15] Looopings. *Efteling zet virtuele wachtrij bij Danse Macabre uit vanwege werkzaamheden.* Jun 2025. URL: https://www.looopings.nl/weblog/29326/Efteling-zet-virtuele-wachtrij-bij-Danse-Macabre-uit-vanwege-werkzaamheden.html.

[16] Looopings. *Nieuw ritprogramma bij Efteling-attractie Danse Macabre zorgt voor storingen.* Mei 2025. URL: https://www.looopings.nl/weblog/29050/Nieuw-ritprogramma-bij-Efteling-attractie-Danse-Macabre-zorgt-voor-storingen.html.

[17] Looopings. *Nieuwe Efteling-attractie Danse Macabre draait op rustige dagen op halve capaciteit.* Jan 2025. URL: https://www.looopings.nl/weblog/27766/Nieuwe-Efteling-attractie-Danse-Macabre-draait-op-rustige-dagen-op-halve-capaciteit.html.

[18] Looopings. *Vanaf vandaag: virtuele wachtrij actief bij Danse Macabre in de Efteling.* Apr 2025. URL: https://www.looopings.nl/weblog/28700/Vanaf-vandaag-virtuele-wachtrij-actief-bij-Danse-Macabre-in-de-Efteling.html.

[19] Looopings. *Virtuele wachtrij bij Efteling-attractie Droomvlucht komende zomer terug.* Jun 2025. URL: https://www.looopings.nl/weblog/29147/Virtuele-wachtrij-bij-Efteling-attractie-Droomvlucht-komende-zomer-terug.html.

[20] Paul Sprangers en Tim Hinssen. *434: De operatie en showtechniek van Danse Macabre.* Kleine Boodschap, feb 2025. URL: https://kleineboodschap.com/afleveringen/2025/2/10/434-de-operatie-en-showtechniek-van-danse-macabre.

[21] Zachary Bull. *Danse Macabre in Efteling wachttijden.* URL: https://queue-times.com/nl/parks/160/rides/14114. Eigen wijziging bij overgenomen grafieken: legenda-element 'Gemeld door gebruiker' verwijderd ter verduidelijking.

[22] Zachary Bull. *Droomvlucht in Efteling wachttijden.* URL: https://queue-times.com/nl/parks/160/rides/6165. Eigen wijziging bij overgenomen grafieken: legenda-element 'Gemeld door gebruiker' verwijderd ter verduidelijking.

[23] Zachary Bull. *Droomvlucht Regular Queue in Efteling wachttijden.* URL: https://queue-times.com/nl/parks/160/rides/13744. Eigen wijziging bij overgenomen grafieken: legenda-element 'Gemeld door gebruiker' verwijderd ter verduidelijking.

[24] Zachary Bull. *Efteling druktekalender.* URL: https://queue-times.com/nl/parks/160/calendar. Geraadpleegd op 19-06-2025.

[25] Zachary Bull. *Gemiddelde wachttijd per uur (2024).* URL: https://queue-times.com/nl/parks/160/rides/6165/2024%5C#dropdown.

[26] Zachary Bull. *Over Queue Times.* URL: https://queue-times.com/nl/pages/about.

[27] Eftepedia. *Droomvlucht*. URL: https://www.eftepedia.nl/lemma/Droomvlucht. Geraadpleegd op 08-02-2025.
[28] Looopings. *Wachttijden Efteling*. URL: https://www.looopings.nl/wachten/efteling.

A Code Basismodel

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from numpy import random
import pandas as pd
import datetime as dt

data2023 = pd.read_excel('C:\\Users\\annev\\Scriptie\\
    Droomvluchtdata.xlsx', sheet_name = "Normaal 2023 storingen")
dag = 62
aantal_simulaties = 1
T_R = 6+47/60 # ritduur in minuten
g = 2*28 # aantal voertuigen
mu = g/T_R # aantal groepen per minuut
print(mu)
datadag = data2023.iloc[dag] # Data inlezen uit Excel file
wachttijd = datadag.values
wachttijd = wachttijd[1:] # Eerste kolom bevat de datum, die
    verwijderen

vroegsluit = False # Om simulatie eerder te stoppen als het park
    een uur eerder sluit
for i in range(12):
    if not wachttijd[len(wachttijd)-1-i] >= 0:
        if i == 0:
            vroegsluit = True
    else:
        vroegsluit = False
if vroegsluit == True:
    wachttijd = wachttijd[0:len(wachttijd)-12]

N = len(wachttijd)-1 # Aantal vijftallen minuten
T = np.zeros(N+1)
for i in range(N+1):
    T[i] = i*5 # Vijftallen minuten

nepdata = np.zeros(N+1) # Wachttijd gecorrigeerd voor storingen etc
storing = False
tijdeen = -1
tijdtwee = -1
for i in range(N+1):
    nepdata[i] = wachttijd[i] # Kopieren echte wachttijd naar
    nepdata
for i in range(N+1): # Gecorrigeerde wachttijd bepalen
    if (not wachttijd[i] >= 0) and storing == False: # Checkt op
    begin van storing
        tijdeen = i-1
        storing = True
    if wachttijd[i] >= 0 and storing == True: # Checkt op einde van
     storing
        tijdtwee = i
        storing = False
    if i == N and storing == True: # Einde van dag
        tijdtwee = N
        storing = False
    if tijdeen > 0 and tijdtwee > 0: # Begin en einde storing zijn
```

```
    bekend
        if tijdtwee == N: # Einde van dag
            for j in range(tijdtwee-tijdeen+1):
                nepdata[tijdeen+j]=wachttijd[tijdeen]
        else: # Interpolatie tussen begin en einde storing
            for j in range(tijdtwee-tijdeen+1):
                nepdata[tijdeen+j] = wachttijd[tijdeen]+(wachttijd[
    tijdtwee]-wachttijd[tijdeen])*(j)/(tijdtwee-tijdeen)
        tijdeen = -1
        tijdtwee = -1

DT_W = np.zeros(len(wachttijd))
lambdas = np.zeros(len(DT_W))
for i in range(N): # Delta T_W en lambda_n bepalen
    DT_W[i+1] = nepdata[i+1]-nepdata[i]
    lambdas[i] = max(mu*(DT_W[i+1]/5+1),0)

def lambda_ster (t): # lambda*(t)
    n = 0
    T_1 = 0
    for i in range(N):
        if t >= T[i] and t < T[i+1]: # Kijken in welk tijdsinterval
     t ligt
            n = i
            T_1 = T[i]
        elif t == T[N]: # Als t = T_N
            return 0
    return lambdas[n]+(lambdas[n+1]-lambdas[n])*(t-T_1)/5

def lambda_dak (t): # lambda^(t)
    for j in range(N):
        if t >= T[j] and t < T[j+1]:
            return(lambdas[j])
        elif t == T[N]:
            return(lambdas[N])


def simulatie(lambdaversie):
    sim = np.zeros(int(T[N])) # Aantal groepen dat aankomt in
    interval [i-1,i], i-de minuut
    groepen = np.zeros(int(T[N])+1) # Aantal groepen dat in de rij
    staat op tijdstip i
    wachttijd_simulatie = np.zeros(int(T[N])+1) # Gesimuleerde
    wachttijd op tijdstip i
    lambda_t = np.zeros(int(T[N])+1) # Waardes van lambda(t)
    if lambdaversie == 0: # lambda^, constant op intervallen
        for i in range(int(T[N])+1):
            lambda_t[i] = lambda_dak(i)
    elif lambdaversie == 1: # lambda*, interpolatie
        for i in range(int(T[N])+1):
            lambda_t[i] = lambda_ster(i)

    for i in range(int(T[N])):
        sim[i] = random.poisson(lam=lambda_t[i]) # Aantal groepen
    dat aankomt in interval [i-1,i], i-de minuut
        groepen[i+1] = max(groepen[i]-mu+sim[i],0) # Aantal groepen
     dat er op tijdstip i+1 in rij staat = aantal groepen dat op
```

```
    tijdstip i in rij staat
100                                                     # + aantal
    groepen dat er in minuut i+1 is bij gekomen - aantal groepen
    dat in minuut i+1 ingestapt is,
101                                                     # negatief
    aantal groepen kan niet
102         wachttijd_simulatie[i+1] = groepen[i+1]/mu # Gesimuleerde
    wachttijd op tijdstip i
103     return(wachttijd_simulatie)
104
105 wachttijdsimulatie_dak = np.zeros(int(T[N])+1)
106 wachttijdsimulatie_ster = np.zeros(int(T[N])+1)
107 for i in range(aantal_simulaties):
108     wachttijdsimulatie_dak = wachttijdsimulatie_dak + simulatie(0)
109     wachttijdsimulatie_ster = wachttijdsimulatie_ster + simulatie
    (1)
110 wachttijdsimulatie_dak = wachttijdsimulatie_dak/aantal_simulaties #
     Gemiddelde gesimuleerde wachttijd met lambda^
111 wachttijdsimulatie_ster = wachttijdsimulatie_ster/aantal_simulaties
     # Gemiddelde gesimuleerde wachttijd met lambda*
112
113 openingstijd = dt.datetime(year = 2025, month = 3, day = 3, hour =
    9, minute = 30)
114 xas1min = [openingstijd + dt.timedelta(minutes=i) for i in range(
    len(wachttijdsimulatie_dak))]
115 xas5min = [openingstijd + 5*dt.timedelta(minutes=i) for i in range(
    len(wachttijd))]
116
117 fig, ax = plt.subplots()
118 ax.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M'))
119 plt.plot(xas1min, wachttijdsimulatie_dak, label = "Sim_wachttijd
    lambda^", color = 'tab:orange')
120 plt.plot(xas1min, wachttijdsimulatie_ster, label = "Simu_wachttijd
    lambda*", color = 'tab:blue')
121 plt.plot(xas5min, nepdata, label = "Gecorrigeerde wachttijd",
    linestyle = 'dashed', color = 'tab:green')
122 plt.plot(xas5min, wachttijd, label = "Echte wachttijd", color = '
    green')
123 plt.gcf().autofmt_xdate()
124 plt.legend()
125 ax.set_ylim([-2, 45])
126 plt.xlabel("Tijdstip")
127 plt.ylabel("Wachttijd in minuten")
128 plt.title("Echte en gesimuleerde wachttijd")
129 plt.show()
```

B Code Droomvlucht Methode 1

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from numpy import random
import pandas as pd
import datetime as dt
import statistics

datavirt = pd.read_excel('C:\\Users\\annev\\Scriptie\\
    Droomvluchtdata.xlsx', sheet_name = "Virtueel 2024 storingen")
datareg = pd.read_excel('C:\\Users\\annev\\Scriptie\\
    Droomvluchtdata.xlsx', sheet_name = "Normaal 2024 storingen")
drukte = pd.read_excel('C:\\Users\\annev\\Scriptie\\Droomvluchtdata
    .xlsx', sheet_name = "Druktepercentage 2024")

dag2024 = 0 # Dag die we simuleren
aantal_simulaties = 1
T_R = 6+47/60 # Ritduur in minuten
g = 2*28 # Aantal voertuigen
mu = g/T_R # Aantal groepen per minuut
gamma = 1/4 # Fractie van gondels die aan virtuele rij worden
    toegewezen

# Virtuele wachttijden op dag die we simuleren, alleen gebruikt
    voor plot
wachttijdvirt = datavirt.iloc[dag2024]
wachttijdvirt = wachttijdvirt.values
wachttijdvirt = wachttijdvirt[1:]

# Virtuele wachttijden op dag die we simuleren, alleen gebruikt
    voor plot
wachttijdreg = datareg.iloc[dag2024]
wachttijdreg = wachttijdreg.values
wachttijdreg = wachttijdreg[1:]

# Drukte op dag die we simuleren
drukte2024 = drukte.iloc[dag2024]
drukte2024 = drukte2024.values
drukte2024 = drukte2024[1]

# Gemiddelde virtuele wachttijd, voor Methode 0 vervangen we "62"
    door "dag2024"
wachttijdgemvirt = datavirt.iloc[62]
wachttijdgemvirt = wachttijdgemvirt.values
wachttijdgemvirt = wachttijdgemvirt[1:]

# Gemiddelde reguliere wachttijd, voor Methode 0 vervangen we "62"
    door "dag2024"
wachttijdgemreg = datareg.iloc[62]
wachttijdgemreg = wachttijdgemreg.values
wachttijdgemreg = wachttijdgemreg[1:]

vroegsluit = False # Om simulatie eerder te stoppen als het park
    een uur eerder sluit
for i in range(12):
    if not wachttijdvirt[len(wachttijdvirt)-1-i] >= 0:
```

```
        if i == 0:
            vroegsluit = True
    else:
        vroegsluit = False
if vroegsluit == True:
    wachttijdvirt = wachttijdvirt[0:len(wachttijdvirt)-12]
    wachttijdreg = wachttijdreg[0:len(wachttijdreg)-12]
    wachttijdgemvirt = wachttijdgemvirt[0:len(wachttijdgemvirt)-12]
    wachttijdgemreg = wachttijdgemreg[0:len(wachttijdgemreg)-12]

N = len(wachttijdvirt)-1 # Aantal vijftallen minuten
T = np.zeros(N+1)
for i in range(N+1):
    T[i] = i*5 # Vijftallen minuten

# Functie om te corrigeren voor storingen
def storingen(wachttijd):
    nepwachttijd = np.zeros(N+1)
    storing = False
    tijdeen = -1
    tijdtwee = -1
    for i in range(N+1):
        nepwachttijd[i] = wachttijd[i] # Kopieren echte wachttijd
    naar nepwachttijd
    for i in range(N+1): # Gecorrigeerde wachttijd bepalen
        if (not wachttijd[i] >= 0) and storing == False: # Checkt
    op begin van storing
            tijdeen = i-1
            storing = True
        if wachttijd[i] >= 0 and storing == True: # Checkt op einde
     van storing
            tijdtwee = i
            storing = False
        if i == N and storing == True: # Einde van dag
            tijdtwee = N
            storing = False
        if tijdeen > 0 and tijdtwee > 0: # Begin en einde storing
    zijn bekend
            if tijdtwee == N: # Einde van dag
                for j in range(tijdtwee-tijdeen+1):
                    nepwachttijd[tijdeen+j]=wachttijd[tijdeen]
            else: # Interpolatie tussen begin en einde storing
                for j in range(tijdtwee-tijdeen+1):
                    nepwachttijd[tijdeen+j] = wachttijd[tijdeen]+(
    wachttijd[tijdtwee]-wachttijd[tijdeen])*(j)/(tijdtwee-tijdeen)
            tijdeen = -1
            tijdtwee = -1
    return (nepwachttijd)

nepdatagemvirt = storingen(wachttijdgemvirt)
nepdatagemreg = storingen(wachttijdgemreg)

lambda_dak = np.zeros(int(T[N])+1)
lambda_virt = np.zeros(int(T[N])+1)
lambda_reg = np.zeros(int(T[N])+1)


```

```
def lambda_bepalen(data, type):
    DT_W = np.zeros(len(data))
    lambdas = np.zeros(len(DT_W))
    lambda_t = np.zeros(int(T[N])+1)
    for i in range(N): # Delta T_W en lambda_n bepalen
        DT_W[i+1] = data[i+1]-data[i]
        if type == 0:
            lambdas[i] = max(gamma*mu*(DT_W[i+1]/5+1),0)
        elif type == 1:
            lambdas[i] = max((1-gamma)*mu*(DT_W[i+1]/5+1),0)
        for t in range(int(T[N])+1):
            if  t>= T[i] and t < T[i+1]:
                lambda_t[t]=lambdas[i]
            elif i == T[N]:
                lambda_t[t]=lambdas[N]
    return(lambda_t)


lambda_virt = lambda_bepalen(nepdatagemvirt, 0)
lambda_reg = lambda_bepalen(nepdatagemreg, 1)
geschaaldreg = nepdatagemreg/49.6*drukte2024
geschaaldvirt = nepdatagemvirt/49.6*drukte2024
lambdagem_virt = lambda_bepalen(geschaaldvirt, 0)
lambdagem_reg = lambda_bepalen(geschaaldreg, 1)
lambda_tot = lambda_virt+lambda_reg
lambdagem_tot = lambdagem_virt + lambdagem_reg
alpha = np.zeros(len(lambdagem_tot))
for i in range(len(lambda_tot)):
    if lambda_tot[i] <= 0:
        alpha[i] = alpha[i-1]
    else:
        alpha[i] = lambda_virt[i]/lambda_tot[i]

def simulatie_1poisproces():
    sim = np.zeros(int(T[N])) # Aantal groepen dat aankomt in
    interval [i-1,i], i-de minuut
    wachtendenvirt = np.zeros(int(T[N])+1) # Aantal wachtenden op
    tijdstip i in virtuele rij
    wachtendenreg = np.zeros(int(T[N])+1) # Aantal wachtenden op
    tijdstip i in reguliere rij
    groepenvirt = np.zeros(int(T[N])+1) # Aantal groepen dat in de
    virtuele rij aankomt in interval [i-1,i], i-de minuut
    groepenreg = np.zeros(int(T[N])+1) # Aantal groepen dat in de
    reguliere rij aankomt in interval [i-1,i], i-de minuut
    wachttijdvirt_sim = np.zeros(int(T[N])+1) # Gesimuleerde
    wachttijd op tijdstip i in virtuele rij
    wachttijdreg_sim = np.zeros(int(T[N])+1) # Gesimuleerde
    wachttijd op tijdstip i in reguliere rij
    for i in range(int(T[N])):
        sim[i] = random.poisson(lam=lambdagem_tot[i]) # Aantal
    groepen dat aankomt in interval [i-1,i], i-de minuut
        bernoulli = random.binomial(n=1, p = alpha[i], size = int(
    sim[i]))
        for j in range(int(sim[i])):
            if bernoulli[j] == 1:
                groepenvirt[i] = groepenvirt[i] + 1
            else:
```

```
                groepenreg[i] = groepenreg[i] + 1
        wachtendenvirt[i+1] = max(wachtendenvirt[i]-gamma*mu+
    groepenvirt[i],0)
        wachtendenreg[i+1] = max(wachtendenreg[i]-(1-gamma)*mu+
    groepenreg[i],0)
        wachttijdvirt_sim[i+1] = wachtendenvirt[i+1]/(gamma*mu) #
    Gesimuleerde wachttijd op tijdstip i
        wachttijdreg_sim[i+1] = wachtendenreg[i+1]/((1-gamma)*mu)
    return(wachttijdvirt_sim, wachttijdreg_sim)

wachttijdvirt_sim = np.zeros(int(T[N])+1)
wachttijdreg_sim = np.zeros(int(T[N])+1)

for i in range(aantal_simulaties):
    simulatie = simulatie_1poisproces()
    wachttijdvirt_sim = wachttijdvirt_sim + simulatie[0]
    wachttijdreg_sim = wachttijdreg_sim + simulatie[1]
wachttijdvirt_sim = wachttijdvirt_sim/aantal_simulaties #
    Gemiddelde gesimuleerde wachttijd met lambda^
wachttijdreg_sim = wachttijdreg_sim/aantal_simulaties

openingstijd = dt.datetime(year = 2025, month = 3, day = 3, hour =
    9, minute = 30)
xas1min = [openingstijd + dt.timedelta(minutes=i) for i in range(
    len(wachttijdvirt_sim))]
xas5min = [openingstijd + 5*dt.timedelta(minutes=i) for i in range(
    len(wachttijdvirt))]

fig, ax = plt.subplots()
ax.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M'))
plt.plot(xas1min, wachttijdvirt_sim, label = "Simulatie virtueel")
plt.plot(xas1min, wachttijdreg_sim, label = "Simulatie regulier")
plt.plot(xas5min, wachttijdvirt, label = "Echt virtueel")
plt.plot(xas5min, wachttijdreg, label = "Echt regulier")
plt.gcf().autofmt_xdate()
ax = plt.gca()
ax.set_ylim([-5, 100])
plt.legend()
plt.xlabel("Tijdstip")
plt.ylabel("Wachttijd in minuten")
plt.title("Echte en gesimuleerde wachttijd")
plt.show()

foutvirt = np.zeros(N+1)
foutreg = np.zeros(N+1)

def fout():
    for i in range(len(wachttijdvirt)):
        foutvirt[i] = wachttijdvirt_sim[5*i] - wachttijdvirt[i]
        foutreg[i] = wachttijdreg_sim[5*i] - wachttijdreg[i]
    return(foutvirt, foutreg)

fouten = fout()
foutenvirt = fouten[0]
foutenreg = fouten[1]

foutenvirt = abs(foutenvirt)
```

```
foutenreg = abs(foutenreg)
totaalfoutvirt = 0
aantalfoutvirt = 0
totaalfoutreg = 0
aantalfoutreg = 0
for i in range(len(foutenvirt)):
    if foutenvirt[i] >= 0:
        totaalfoutvirt = totaalfoutvirt + foutenvirt[i]
        aantalfoutvirt = aantalfoutvirt + 1
for i in range(len(foutenreg)):
    if foutenreg[i] >= 0:
        totaalfoutreg = totaalfoutreg + foutenreg[i]
        aantalfoutreg = aantalfoutreg + 1
gemfoutvirt = totaalfoutvirt/aantalfoutvirt
gemfoutreg = totaalfoutreg/aantalfoutreg
print("Gemiddelde fout virtueel = ", gemfoutvirt)
print("Gemiddelde fout regulier = ",gemfoutreg)

fig, ax = plt.subplots()
ax.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M'))
plt.plot(xas5min, foutenvirt, label = "Fout virtueel")
plt.plot(xas5min, foutenreg, label = "Fout regulier")
plt.gcf().autofmt_xdate()
ax.set_ylim([-5, 65])
plt.legend(loc = 'upper left')
plt.xlabel("Tijdstip")
plt.ylabel("Absolute fout in minuten verschil")
plt.title("Absolute fout van gesimuleerde wachttijden")
plt.show()
```

C Code Droomvlucht Methode 2

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from numpy import random
import pandas as pd
import datetime as dt

data2023 = pd.read_excel('C:\\Users\\annev\\Scriptie\\
    Droomvluchtdata.xlsx', sheet_name = "Normaal 2023 storingen")
datavirt = pd.read_excel('C:\\Users\\annev\\Scriptie\\
    Droomvluchtdata.xlsx', sheet_name = "Virtueel 2024 storingen")
datareg = pd.read_excel('C:\\Users\\annev\\Scriptie\\
    Droomvluchtdata.xlsx', sheet_name = "Normaal 2024 storingen")
drukte = pd.read_excel('C:\\Users\\annev\\Scriptie\\Droomvluchtdata
    .xlsx', sheet_name = "Druktepercentage 2024")

dag2024 = 0 # Dag die we simuleren
aantal_simulaties = 1
T_R = 6+47/60 # Ritduur in minuten
g = 2*28 # Aantal voertuigen
mu = g/T_R # Aantal groepen per minuut
gamma = 1/4 # Fractie van gondels die aan virtuele rij worden
    toegewezen

# Virtuele wachttijden op dag die we simuleren, alleen gebruikt
    voor plot
wachttijdvirt = datavirt.iloc[dag2024]
wachttijdvirt = wachttijdvirt.values
wachttijdvirt = wachttijdvirt[1:]

# Virtuele wachttijden op dag die we simuleren, alleen gebruikt
    voor plot
wachttijdreg = datareg.iloc[dag2024]
wachttijdreg = wachttijdreg.values
wachttijdreg = wachttijdreg[1:]

# Drukteniveau op dag die we simuleren
drukte2024 = drukte.iloc[dag2024]
drukte2024 = drukte2024.values
drukte2024 = drukte2024[1]

# Gemiddelde virtuele wachttijden
wachttijdgemvirt = datavirt.iloc[62]
wachttijdgemvirt = wachttijdgemvirt.values
wachttijdgemvirt = wachttijdgemvirt[1:]

# Gemiddelde reguliere wachttijden
wachttijdgemreg = datareg.iloc[62]
wachttijdgemreg = wachttijdgemreg.values
wachttijdgemreg = wachttijdgemreg[1:]

# Gemiddelde reguliere wachttijden 2023
data2023 = data2023.iloc[62]
wachttijd2023 = data2023.values
wachttijd2023 = wachttijd2023[1:]

```

```
drukte2023 = 51.2 # Gemiddeld drukteniveau juli en augustus 2023

vroegsluit = False # Om simulatie eerder te stoppen als het park een uur eerder sluit
for i in range(12):
    if not wachttijdvirt[len(wachttijdvirt)-1-i] >= 0:
        if i == 0:
            vroegsluit = True
    else:
        vroegsluit = False
if vroegsluit == True:
    wachttijdvirt = wachttijdvirt[0:len(wachttijdvirt)-12]
    wachttijdreg = wachttijdreg[0:len(wachttijdreg)-12]
    wachttijd2023 = wachttijd2023[0:len(wachttijd2023)-12]


N = len(wachttijdvirt)-1 # Aantal vijftallen minuten
T = np.zeros(N+1)
for i in range(N+1):
    T[i] = i*5 # Vijftallen minuten

# Functie om te corrigeren voor storingen
def storingen(wachttijd):
    nepdata = np.zeros(N+1)
    storing = False
    tijdeen = -1
    tijdtwee = -1
    for i in range(N+1):
        nepdata[i] = wachttijd[i] # Kopieren echte wachttijd naar nepwachttijd
    for i in range(N+1): # Gecorrigeerde wachttijd bepalen
        if (not wachttijd[i] >= 0) and storing == False: # Checkt op begin van storing
            tijdeen = i-1
            storing = True
        if wachttijd[i] >= 0 and storing == True: # Checkt op einde van storing
            tijdtwee = i
            storing = False
        if i == N and storing == True: # Einde van dag
            tijdtwee = N
            storing = False
        if tijdeen > 0 and tijdtwee > 0: # Begin en einde storing zijn bekend
            if tijdtwee == N: # Einde van dag
                for j in range(tijdtwee-tijdeen+1):
                    nepdata[tijdeen+j]=wachttijd[tijdeen]
            else: # Interpolatie tussen begin en einde storing
                for j in range(tijdtwee-tijdeen+1):
                    nepdata[tijdeen+j] = wachttijd[tijdeen]+(wachttijd[tijdtwee]-wachttijd[tijdeen])*(j)/(tijdtwee-tijdeen)
            tijdeen = -1
            tijdtwee = -1
    return (nepdata)

nepdatagemvirt = storingen(wachttijdgemvirt)
nepdatagemreg = storingen(wachttijdgemreg)
```

```
101
102 lambda_dak = np.zeros(int(T[N])+1)
103 lambda_virt = np.zeros(int(T[N])+1)
104 lambda_reg = np.zeros(int(T[N])+1)
105
106 def lambda_bepalen(data, type):
107     DT_W = np.zeros(len(data))
108     lambdas = np.zeros(len(DT_W))
109     lambda_t = np.zeros(int(T[N])+1)
110     for i in range(N): # Delta T_W en lambda_n bepalen
111         DT_W[i+1] = data[i+1]-data[i]
112         if type == 0:
113             lambdas[i] = max(gamma*mu*(DT_W[i+1]/5+1),0)
114         elif type == 1:
115             lambdas[i] = max((1-gamma)*mu*(DT_W[i+1]/5+1),0)
116         elif type == 2:
117             lambdas[i] = max(mu*(DT_W[i+1]/5+1),0)
118         for t in range(int(T[N])+1):
119             if  t>= T[i] and t < T[i+1]:
120                 lambda_t[t]=lambdas[i]
121             elif i == T[N]:
122                 lambda_t[t]=lambdas[N]
123     return(lambda_t)
124
125 wachttijd2023_drukte = wachttijd2023/drukte2023*drukte2024
126 lambda_dak = lambda_bepalen(wachttijd2023_drukte, 2)
127 lambda_virt = lambda_bepalen(nepdatagemvirt,0)
128 lambda_reg = lambda_bepalen(nepdatagemreg,1)
129 lambda_tot = lambda_virt+lambda_reg
130 alpha = np.zeros(len(lambda_tot))
131 for i in range(len(lambda_tot)):
132     if lambda_tot[i] <= 0:
133         alpha[i] = alpha[i-1]
134     else:
135         alpha[i] = lambda_virt[i]/lambda_tot[i]
136
137 def simulatie_poisproces():
138     sim = np.zeros(int(T[N])) # Aantal groepen dat aankomt in
    interval [i-1,i], i-de minuut
139     wachtendenvirt = np.zeros(int(T[N])+1) # Aantal wachtenden op
    tijdstip i in virtuele rij
140     wachtendenreg = np.zeros(int(T[N])+1) # Aantal wachtenden op
    tijdstip i in reguliere rij
141     groepenvirt = np.zeros(int(T[N])+1) # Aantal groepen dat in de
    virtuele rij aankomt in interval [i-1,i], i-de minuut
142     groepenreg = np.zeros(int(T[N])+1) # Aantal groepen dat in de
    reguliere rij aankomt in interval [i-1,i], i-de minuut
143     wachttijdvirt_sim = np.zeros(int(T[N])+1) # Gesimuleerde
    wachttijd op tijdstip i in virtuele rij
144     wachttijdreg_sim = np.zeros(int(T[N])+1) # Gesimuleerde
    wachttijd op tijdstip i in reguliere rij
145     for i in range(int(T[N])):
146         sim[i] = random.poisson(lam=lambda_dak[i]) # Aantal groepen
     dat aankomt in interval [i-1,i], i-de minuut
147         bernoulli = random.binomial(n=1, p = alpha[i], size = int(
    sim[i]))
148         for j in range(int(sim[i])):
```

```
            if bernoulli[j] == 1:
                groepenvirt[i] = groepenvirt[i] + 1
            else:
                groepenreg[i] = groepenreg[i] + 1
        wachtendenvirt[i+1] = max(wachtendenvirt[i]-gamma*mu+groepenvirt[i],0)
        wachtendenreg[i+1] = max(wachtendenreg[i]-(1-gamma)*mu+groepenreg[i],0)
        wachttijdvirt_sim[i+1] = wachtendenvirt[i+1]/(gamma*mu) # Gesimuleerde wachttijd op tijdstip i
        wachttijdreg_sim[i+1] = wachtendenreg[i+1]/((1-gamma)*mu)
    return(wachttijdvirt_sim, wachttijdreg_sim)

wachttijdvirt_sim = np.zeros(int(T[N])+1)
wachttijdreg_sim = np.zeros(int(T[N])+1)

for i in range(aantal_simulaties):
    simulatie = simulatie_poisproces()
    wachttijdvirt_sim = wachttijdvirt_sim + simulatie[0]
    wachttijdreg_sim = wachttijdreg_sim + simulatie[1]
wachttijdvirt_sim = wachttijdvirt_sim/aantal_simulaties # Gemiddelde gesimuleerde wachttijd met lambda^
wachttijdreg_sim = wachttijdreg_sim/aantal_simulaties

openingstijd = dt.datetime(year = 2025, month = 3, day = 3, hour = 9, minute = 30)
xas1min = [openingstijd + dt.timedelta(minutes=i) for i in range(len(wachttijdvirt_sim))]
xas5min = [openingstijd + 5*dt.timedelta(minutes=i) for i in range(len(wachttijdvirt))]

fig, ax = plt.subplots()
ax.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M'))
plt.plot(xas1min, wachttijdvirt_sim, label = "Simulatie virtueel")
plt.plot(xas1min, wachttijdreg_sim, label = "Simulatie regulier")
plt.plot(xas5min, wachttijdvirt, label = "Echt virtueel")
plt.plot(xas5min, wachttijdreg, label = "Echt regulier")
plt.gcf().autofmt_xdate()
ax = plt.gca()
ax.set_ylim([-5, 100])
plt.legend()
plt.xlabel("Tijdstip")
plt.ylabel("Wachttijd in minuten")
plt.title("Echte en gesimuleerde wachttijd")
plt.show()

foutvirt = np.zeros(N+1)
foutreg = np.zeros(N+1)

def fout():
    for i in range(len(wachttijdvirt)):
        foutvirt[i] = wachttijdvirt_sim[5*i] - wachttijdvirt[i]
        foutreg[i] = wachttijdreg_sim[5*i] - wachttijdreg[i]
    return(foutvirt, foutreg)

fouten = fout()
foutenvirt = fouten[0]
```

```
foutenreg = fouten[1]

foutenvirt = abs(foutenvirt)
foutenreg = abs(foutenreg)
totaalfoutvirt = 0
aantalfoutvirt = 0
totaalfoutreg = 0
aantalfoutreg = 0
for i in range(len(foutenvirt)):
    if foutenvirt[i] >= 0:
        totaalfoutvirt = totaalfoutvirt + foutenvirt[i]
        aantalfoutvirt = aantalfoutvirt + 1
for i in range(len(foutenreg)):
    if foutenreg[i] >= 0:
        totaalfoutreg = totaalfoutreg + foutenreg[i]
        aantalfoutreg = aantalfoutreg + 1
gemfoutvirt = totaalfoutvirt/aantalfoutvirt
gemfoutreg = totaalfoutreg/aantalfoutreg
print("Gemiddelde fout virtueel = ", gemfoutvirt)
print("Gemiddelde fout regulier = ",gemfoutreg)

fig, ax = plt.subplots()
ax.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M'))
plt.plot(xas5min, foutenvirt, label = "Fout virtueel")
plt.plot(xas5min, foutenreg, label = "Fout regulier")
plt.gcf().autofmt_xdate()
ax.set_ylim([-5, 65])
plt.legend(loc = 'upper left')
plt.xlabel("Tijdstip")
plt.ylabel("Absolute fout in minuten verschil")
plt.title("Absolute fout van gesimuleerde wachttijden")
plt.show()
```

D Code Danse Macabre Methode 1

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from numpy import random
import pandas as pd
import datetime as dt
import math

datavirtmei = pd.read_excel('C:\\Users\\annev\\Scriptie\\Danse Macabre Data\\Danse Macabre Data.xlsx', sheet_name = "Alles")
dataregmei = pd.read_excel('C:\\Users\\annev\\Scriptie\\Danse Macabre regulier data.xlsx', sheet_name = "Methode 2")
drukteniveau = pd.read_excel('C:\\Users\\annev\\Scriptie\\Danse Macabre regulier data.xlsx', sheet_name = "Drukte mei")

dag = 0 # Dag die we simuleren
aantal_simulaties = 1

T_S = 7.843 # Gemiddelde duur van 1 showcyclus
M = 37.625 # Aantal mensen dat gemiddeld per voorshow het kerkhof op gaat vanuit de virtuele rij
G_gem = 3 # Gemiddelde groepsgrootte
mu_gem = 108/(G_gem*T_S) # Aantal groepen dat gemiddeld per minuut door de attractie geholpen wordt
mu_v = M/T_S # Aantal mensen dat gemiddeld per minuut het kerkhof op mag vanuit de virtuele rij
mu_r = 108/T_S-mu_v # Aantal mensen dat gemiddeld per minuut het kerkhof op mag vanuit de reguliere rij
gamma = M/108 # Gemiddelde fractie van mensen op het kerkhof dat uit de virtuele rij afkomstig is

# Gemiddelde virtuele wachttijden mei 2025
wachttijdvirtgem = datavirtmei.iloc[24]
wachttijdvirtgem = wachttijdvirtgem.values
wachttijdvirtgem = wachttijdvirtgem[1:]

# Gemiddelde reguliere wachttijden mei 2025
wachttijdreggem = dataregmei.iloc[24]
wachttijdreggem = wachttijdreggem.values
wachttijdreggem = wachttijdreggem[1:]

# Gemiddelde drukte mei 2025
druktegem = drukteniveau.iloc[24]
druktegem = druktegem.values
druktegem = druktegem[1]

# Drukte en openings- en sluitingstijd op te simuleren dag
drukte = drukteniveau.iloc[dag]
drukte = drukte.values
openingsuur = drukte[3] # Uur waarop het park opent
sluitingsuur = drukte[4] # Uur waarop het park sluit
sluitingstijd = (sluitingsuur - openingsuur)*60 # Aantal minuten dat het park open is
sluitingstijd = sluitingstijd.item()
openingsuur = openingsuur.item()
openingsuur = int(openingsuur)
```

```
drukte = drukte[1] # Drukte van te simuleren dag

# Historische virtuele wachttijden op gesimuleerde dag, gebruikt voor plot
wachttijdvirt = datavirtmei.iloc[dag]
wachttijdvirt = wachttijdvirt.values
wachttijdvirt = wachttijdvirt[1:]

# Historische reguliere wachttijden op gesimuleerde dag, gebruikt voor plot
wachttijdreg = dataregmei.iloc[dag]
wachttijdreg = wachttijdreg.values
wachttijdreg = wachttijdreg[1:]

if sluitingstijd == 480: # Past lengte van dataset aan als deze te groot is
    wachttijdvirtgem = wachttijdvirtgem[0:len(wachttijdvirtgem)-24]
    wachttijdreggem = wachttijdreggem[0:len(wachttijdreggem)-24]
    wachttijdvirt = wachttijdvirt[0:len(wachttijdvirt)-24]
    wachttijdreg = wachttijdreg[0:len(wachttijdreg)-24]

if sluitingstijd == 540: # Past lengte van dataset aan als deze te groot is
    wachttijdvirtgem = wachttijdvirtgem[0:len(wachttijdvirtgem)-12]
    wachttijdreggem = wachttijdreggem[0:len(wachttijdreggem)-12]
    wachttijdvirt = wachttijdvirt[0:len(wachttijdvirt)-12]
    wachttijdreg = wachttijdreg[0:len(wachttijdreg)-12]

N = len(wachttijdreggem)-1 # Aantal vijftallen minuten
T = np.zeros(N+1)
for i in range(N+1):
    T[i] = i*5 # Vijftallen minuten

def lambda_bepalen(data, type):
    DT_W = np.zeros(len(data))
    lambda_n = np.zeros(len(DT_W))
    lambda_t = np.zeros(int(T[N])+1)
    for i in range(N): # Delta T_W en lambda_n bepalen
        DT_W[i+1] = data[i+1]-data[i]
        if type == 0: # Virtueel
            lambda_n[i] = max(gamma*mu_gem*(DT_W[i+1]/5+1),0) # lambda_n bepalen
        elif type == 1: # Regulier
            lambda_n[i] = max((1-gamma)*mu_gem*(DT_W[i+1]/5+1),0) # lambda_n bepalen
        for t in range(int(T[N])+1): # lambda(t) vullen
            if  t>= T[i] and t < T[i+1]:
                lambda_t[t]=lambda_n[i]
            elif i == T[N]:
                lambda_t[t]=lambda_n[N]
    return(lambda_t)

lambda_virt = lambda_bepalen(wachttijdvirtgem,0) # Nodig voor alpha(t)
lambda_reg = lambda_bepalen(wachttijdreggem,1) # Nodig voor alpha(t)
lambda_tot = lambda_virt+lambda_reg # Nodig voor alpha(t)
```

```
geschaaldreg = wachttijdreggem/druktegem*drukte
geschaaldvirt = wachttijdvirtgem/druktegem*drukte
lambdagem_virt = lambda_bepalen(geschaaldvirt, 0)
lambdagem_reg = lambda_bepalen(geschaaldreg, 1)
lambdagem_tot = lambdagem_virt + lambdagem_reg #\hat{\lambda}(t)

alpha = np.zeros(len(lambda_tot))
for i in range(len(lambda_tot)): # alpha(t) bepalen
    if lambda_tot[i] <= 0: # Geval van een storing
        alpha[i] = alpha[i-1]
    else:
        alpha[i] = lambda_virt[i]/lambda_tot[i]

groepsgroottes = [1, 2, 3, 4, 5, 6] # Mogelijke groepsgroottes
groepskansen = [14/156, 57/156, 27/156, 35/156, 18/156, 5/156] #
    Kans op iedere groepsgrootte

def simulatie_poisproces(): # Simulatie
    afbreektijdstip = sluitingstijd # Tijd die tijdsloten niet
    mogen overschrijden
    sim = np.zeros(int(T[N])) # Aantal groepen dat aankomt in
    interval [i-1,i], i-de minuut
    wachtendenvirt = np.zeros(int(T[N])+1) # Aantal wachtenden op
    tijdstip i in virtuele rij
    wachtendenreg = np.zeros(int(T[N])+1) # Aantal wachtenden op
    tijdstip i in reguliere rij
    mensenvirt = np.zeros(int(T[N])+1) # Aantal groepen dat in de
    virtuele rij aankomt in interval [i-1,i], i-de minuut
    mensenreg = np.zeros(int(T[N])+1) # Aantal groepen dat in de
    reguliere rij aankomt in interval [i-1,i], i-de minuut
    wachttijdvirt_sim = np.zeros(int(T[N])+1) # Gesimuleerde
    wachttijd op tijdstip i in virtuele rij
    wachttijdreg_sim = np.zeros(int(T[N])+1) # Gesimuleerde
    wachttijd op tijdstip i in reguliere rij
    virtgesloten = False
    for i in range(int(T[N])):
        if virtgesloten == False:
            if wachttijdvirt_sim[i] > sluitingstijd - i:
                afbreektijdstip = i
                virtgesloten = True
        sim[i] = random.poisson(lam=lambdagem_tot[i]) # Aantal
    groepen dat aankomt in interval [i-1,i], i-de minuut
        bernoulli = random.binomial(n=1, p = alpha[i], size = int(
    sim[i]))
        groepsgrootte = np.random.choice(groepsgroottes, p=
    groepskansen, size = int(sim[i]))
        for j in range(int(sim[i])):
            if bernoulli[j] == 1:
                mensenvirt[i] = mensenvirt[i] + groepsgrootte[j]
            else:
                mensenreg[i] = mensenreg[i] + groepsgrootte[j]
        wachtendenvirt[i+1] = max(wachtendenvirt[i]-mu_v+mensenvirt
    [i],0)
        wachttijdvirt_sim[i+1] = wachtendenvirt[i+1]/(mu_v)
        wachtendenreg[i+1] = max(wachtendenreg[i]-mu_r+mensenreg[i
    ],0)
        wachttijdreg_sim[i+1] = wachtendenreg[i+1]/(mu_r)+13
```

```
    return(wachttijdvirt_sim, wachttijdreg_sim, afbreektijdstip)


wachttijdvirt_sim = np.zeros(int(T[N])+1)
wachttijdreg_sim = np.zeros(int(T[N])+1)

afbreektijdstip = np.zeros(aantal_simulaties)
for i in range(aantal_simulaties):
    simulatie = simulatie_poisproces()
    for j in range(len(simulatie[0])):
        wachttijdvirt_sim[j] = wachttijdvirt_sim[j] + simulatie[0][
    j]
    wachttijdreg_sim = wachttijdreg_sim + simulatie[1]
    afbreektijdstip[i] = simulatie[2]
wachttijdvirt_sim = wachttijdvirt_sim/aantal_simulaties #
    Gemiddelde gesimuleerde wachttijd met lambda^
wachttijdreg_sim = wachttijdreg_sim/aantal_simulaties
afbreektijd = np.mean(afbreektijdstip)
wachttijdvirt_sim = wachttijdvirt_sim[0:math.ceil(afbreektijd)]

openingstijd = dt.datetime(year = 2025, month = 3, day = 3, hour =
    openingsuur, minute = 0)
xas1min = [openingstijd + dt.timedelta(minutes=i) for i in range(
    len(wachttijdreg_sim))]
xas5min = [openingstijd + 5*dt.timedelta(minutes=i) for i in range(
    len(wachttijdreg))]
xasvirt = [openingstijd + dt.timedelta(minutes=i) for i in range(
    len(wachttijdvirt_sim))]
xas2punten = [openingstijd+i*dt.timedelta(minutes = sluitingstijd)
    for i in range(2)]
tijden = [sluitingstijd,0]

werkelijk_sim = wachttijdvirt_sim - (0.00012*wachttijdvirt_sim*
    wachttijdvirt_sim+0.197*wachttijdvirt_sim)

fig, ax = plt.subplots()
ax.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M'))
plt.plot(xasvirt, wachttijdvirt_sim, label = "Sim virt", color = "
    tab:blue")
plt.plot(xas1min, wachttijdreg_sim, label = "Sim reg", color = "tab
    :orange")
plt.plot(xasvirt, werkelijk_sim, label = "Sim werkelijk virt",
    color = "tab:purple")
plt.plot(xas5min, wachttijdvirt, label = "Historisch virt", color =
     "tab:green")
plt.plot(xas5min, wachttijdreg, label = "Historisch reg", color = "
    tab:red")
plt.plot(xas2punten, tijden, label = "Tijd tot sluit", color = "
    black")
plt.gcf().autofmt_xdate()
ax = plt.gca()
ax.set_ylim([-5, 300])
plt.legend(loc = 'upper right')
plt.xlabel("Tijdstip")
plt.ylabel("Wachttijd in minuten")
plt.title("Historische en gesimuleerde wachttijd")
plt.show()
```

E Code Danse Macabre Methode 2

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from numpy import random
import pandas as pd
import datetime as dt
import math

datanov = pd.read_excel('C:\\Users\\annev\\Scriptie\\Danse Macabre regulier data.xlsx', sheet_name = "Methode 1")
datavirtmei = pd.read_excel('C:\\Users\\annev\\Scriptie\\Danse Macabre Data\\Danse Macabre Data.xlsx', sheet_name = "Alles")
dataregmei = pd.read_excel('C:\\Users\\annev\\Scriptie\\Danse Macabre regulier data.xlsx', sheet_name = "Methode 2")

drukteniveau_nov = pd.read_excel('C:\\Users\\annev\\Scriptie\\Danse Macabre regulier data.xlsx', sheet_name = "Drukte november")
drukteniveau_mei = pd.read_excel('C:\\Users\\annev\\Scriptie\\Danse Macabre regulier data.xlsx', sheet_name = "Drukte mei")

dag = 0 # dag die we simuleren
aantal_simulaties = 1

T_S = 7.843 # Gemiddelde duur van 1 showcyclus
M_K = 37.625 # Aantal mensen dat gemiddeld per show vanuit de virtuele rij het kerkhof op mag
G_gem = 3 # Gemiddelde groepsgrootte
mu_gem = 108/(G_gem*T_S) # Aantal groepen dat gemiddeld per minuut door de attractie geholpen wordt
mu_v = M_K/T_S # Aantal mensen dat gemiddeld per minuut het kerkhof op mag vanuit de virtuele rij
mu_r = 108/T_S-mu_v # Aantal mensen dat gemiddeld per minuut het kerkhof op mag vanuit de reguliere rij
gamma = M_K/108 # Gemiddelde fractie van mensen op het kerkhof dat uit de virtuele rij afkomstig is

# Gemiddelde virtuele wachttijden mei 2025
wachttijdvirtgem = datavirtmei.iloc[24]
wachttijdvirtgem = wachttijdvirtgem.values
wachttijdvirtgem = wachttijdvirtgem[1:]

# Gemiddelde reguliere wachttijden mei 2025
wachttijdreggem = dataregmei.iloc[24]
wachttijdreggem = wachttijdreggem.values
wachttijdreggem = wachttijdreggem[1:]

# Gemiddelde reguliere wachttijden november en december 2024
data = datanov.iloc[52]
data = data.values
data = data[1:]

# Gemiddelde drukte november en december 2024
druktegem = drukteniveau_nov.iloc[52]
druktegem = druktegem.values
druktegem = druktegem[1]

```

```
drukte = drukteniveau_mei.iloc[dag] # Bevat drukte en
    openingstijden van te simuleren dag
drukte = drukte.values
openingsuur = drukte[3] # Openingstijd van park in uren
sluitingsuur = drukte[4] # Openingstijd van park in uren
sluitingstijd = (sluitingsuur - openingsuur)*60 # Aantal minuten
    dat het park open is
sluitingstijd = sluitingstijd.item()
openingsuur = openingsuur.item()
openingsuur = int(openingsuur)
drukte = drukte[1] # Drukteniveau van te simuleren dag

# Historische virtuele wachttijden op gesimuleerde dag, alleen
    gebruikt voor plot
wachttijdvirt = datavirtmei.iloc[dag]
wachttijdvirt = wachttijdvirt.values
wachttijdvirt = wachttijdvirt[1:]

# Historische reguliere wachttijden op gesimuleerde dag, alleen
    gebruikt voor plot
wachttijdreg = dataregmei.iloc[dag]
wachttijdreg = wachttijdreg.values
wachttijdreg = wachttijdreg[1:]

if sluitingstijd == 480: # Past lengte van dataset aan als deze te
    groot is
    wachttijdvirtgem = wachttijdvirtgem[0:len(wachttijdvirtgem)-24]
    wachttijdreggem = wachttijdreggem[0:len(wachttijdreggem)-24]
    data = data[0:len(data)-24]
    wachttijdvirt = wachttijdvirt[0:len(wachttijdvirt)-24]
    wachttijdreg = wachttijdreg[0:len(wachttijdreg)-24]

if sluitingstijd == 540: # Past lengte van dataset aan als deze te
    groot is
    wachttijdvirtgem = wachttijdvirtgem[0:len(wachttijdvirtgem)-12]
    wachttijdreggem = wachttijdreggem[0:len(wachttijdreggem)-12]
    data = data[0:len(data)-12]
    wachttijdvirt = wachttijdvirt[0:len(wachttijdvirt)-12]
    wachttijdreg = wachttijdreg[0:len(wachttijdreg)-12]

N = len(data)-1 # Aantal vijftallen minuten
T = np.zeros(N+1)
for i in range(N+1):
    T[i] = i*5 # Vijftallen minuten

lambda_aankomst = np.zeros(int(T[N])+1)
lambda_virt = np.zeros(int(T[N])+1)
lambda_reg = np.zeros(int(T[N])+1)

def lambda_bepalen(data, type):
    DT_W = np.zeros(len(data))
    lambdas = np.zeros(len(DT_W))
    lambda_t = np.zeros(int(T[N])+1)
    for i in range(N): # Delta T_W en lambda_n bepalen
        DT_W[i+1] = data[i+1]-data[i]
        if type == 0:
            lambdas[i] = max(gamma*mu_gem*(DT_W[i+1]/5+1),0)
```

```
        elif type == 1:
            lambdas[i] = max((1-gamma)*mu_gem*(DT_W[i+1]/5+1),0)
        elif type == 2:
            lambdas[i] = max(mu_gem*(DT_W[i+1]/5+1),0)
        for t in range(int(T[N])+1):
            if  t>= T[i] and t < T[i+1]:
                lambda_t[t]=lambdas[i]
            elif i == T[N]:
                lambda_t[t]=lambdas[N]
    return(lambda_t)

data_drukte = data/druktegem*drukte
lambda_aankomst = lambda_bepalen(data_drukte, 2)
lambda_virt = lambda_bepalen(wachttijdvirtgem,0)
lambda_reg = lambda_bepalen(wachttijdreggem,1)
lambda_tot = lambda_virt+lambda_reg
alpha = np.zeros(len(lambda_tot))
for i in range(len(lambda_tot)):
    if lambda_tot[i] <= 0:
        alpha[i] = alpha[i-1]
    else:
        alpha[i] = lambda_virt[i]/lambda_tot[i]

groepsgroottes = [1, 2, 3, 4, 5, 6] # Mogelijke groepsgroottes
groepskansen = [14/156, 57/156, 27/156, 35/156, 18/156, 5/156] #
    Kans op iedere groepsgrootte

def simulatie_poisproces():
    afbreektijdstip = sluitingstijd
    sim = np.zeros(int(T[N])) # Aantal groepen dat aankomt in
    interval [i-1,i], i-de minuut
    wachtendenvirt = np.zeros(int(T[N])+1) # Aantal wachtenden op
    tijdstip i in virtuele rij
    wachtendenreg = np.zeros(int(T[N])+1) # Aantal wachtenden op
    tijdstip i in reguliere rij
    mensenvirt = np.zeros(int(T[N])+1) # Aantal groepen dat in de
    virtuele rij aankomt in interval [i-1,i], i-de minuut
    mensenreg = np.zeros(int(T[N])+1) # Aantal groepen dat in de
    reguliere rij aankomt in interval [i-1,i], i-de minuut
    wachttijdvirt_sim = np.zeros(int(T[N])+1) # Gesimuleerde
    wachttijd op tijdstip i in virtuele rij
    wachttijdreg_sim = np.zeros(int(T[N])+1) # Gesimuleerde
    wachttijd op tijdstip i in reguliere rij
    virtgesloten = False
    for i in range(int(T[N])):
        if virtgesloten == False:
            if wachttijdvirt_sim[i] > sluitingstijd - i:
                afbreektijdstip = i
                virtgesloten = True
        sim[i] = random.poisson(lam=lambda_aankomst[i]) # Aantal
    groepen dat aankomt in interval [i-1,i], i-de minuut
        bernoulli = random.binomial(n=1, p = alpha[i], size = int(
    sim[i]))
        groepsgrootte = np.random.choice(groepsgroottes, p=
    groepskansen, size = int(sim[i]))
        for j in range(int(sim[i])):
            if bernoulli[j] == 1:
```

```
                mensenvirt[i] = mensenvirt[i] + groepsgrootte[j]
            else:
                mensenreg[i] = mensenreg[i] + groepsgrootte[j]
        wachtendenvirt[i+1] = max(wachtendenvirt[i]-mu_v+mensenvirt[i],0)
        wachtendenreg[i+1] = max(wachtendenreg[i]-mu_r+mensenreg[i],0)
        wachttijdvirt_sim[i+1] = wachtendenvirt[i+1]/(mu_v) # Gesimuleerde wachttijd op tijdstip i
        wachttijdreg_sim[i+1] = wachtendenreg[i+1]/(mu_r)+13
    return(wachttijdvirt_sim, wachttijdreg_sim, afbreektijdstip)

wachttijdvirt_sim = np.zeros(int(T[N])+1)
wachttijdreg_sim = np.zeros(int(T[N])+1)
afbreektijdstip = np.zeros(aantal_simulaties)
for i in range(aantal_simulaties):
    simulatie = simulatie_poisproces()
    wachttijdvirt_sim = wachttijdvirt_sim + simulatie[0]
    wachttijdreg_sim = wachttijdreg_sim + simulatie[1]
    afbreektijdstip[i] = simulatie[2]
wachttijdvirt_sim = wachttijdvirt_sim/aantal_simulaties # Gemiddelde gesimuleerde wachttijd met lambda^
wachttijdreg_sim = wachttijdreg_sim/aantal_simulaties
afbreektijd = np.mean(afbreektijdstip)
wachttijdvirt_sim = wachttijdvirt_sim[0:math.ceil(afbreektijd)]

openingstijd = dt.datetime(year = 2025, month = 3, day = 3, hour = openingsuur, minute = 0)
xas1min = [openingstijd + dt.timedelta(minutes=i) for i in range(len(wachttijdreg_sim))]
xas5min = [openingstijd + 5*dt.timedelta(minutes=i) for i in range(len(wachttijdreg))]
xasvirt = [openingstijd + dt.timedelta(minutes=i) for i in range(len(wachttijdvirt_sim))]
xas2punten = [openingstijd+i*dt.timedelta(minutes = sluitingstijd) for i in range(2)]
tijden = [sluitingstijd,0]

werkelijk_sim = wachttijdvirt_sim - (0.00012*wachttijdvirt_sim*wachttijdvirt_sim+0.197*wachttijdvirt_sim)

fig, ax = plt.subplots()
ax.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M'))
plt.plot(xasvirt, wachttijdvirt_sim, label = "Sim virt", color = "tab:blue")
plt.plot(xas1min, wachttijdreg_sim, label = "Sim reg", color = "tab:orange")
plt.plot(xasvirt, werkelijk_sim, label = "Sim werkelijk virt", color = "tab:purple")
plt.plot(xas5min, wachttijdvirt, label = "Historisch virt", color = "tab:green")
plt.plot(xas5min, wachttijdreg, label = "Historisch reg", color = "tab:red")
plt.plot(xas2punten, tijden, label = "Tijd tot sluit", color = "black")
plt.gcf().autofmt_xdate()
ax = plt.gca()
```

```
185 ax.set_ylim([-5, 300])
186 plt.legend(loc = 'upper right')
187 plt.xlabel("Tijdstip")
188 plt.ylabel("Wachttijd in minuten")
189 plt.title("Historische en gesimuleerde wachttijd")
190 plt.show()
```