Is de bandit-methode een verbetering op de klassieke methode van A/B-testen?

Are bandits an improvement for A/B testing compared to the classic method?

door

F. I. M. Lückerath

ter verkrijging van de graad van Bachelor of Science
aan de Technische Universiteit Delft,
in het openbaar te verdedigen op dinsdag 30 juli om 11:00 uur.

Studienummer:	4551680	
Project duur:	23 April, 2019 – 30 Juli, 2019	
Beoordelingscommissie:	Dr. J. Söhl,	TU Delft, begeleider
	Dr. D. Kurowicka,	TU Delft
	Drs. E. M. van Elderen,	TU Delft

TU Delft Delft University of Technology

Voorwoord

Voor u ligt mijn onderzoek ter afronding van de Bachelor Technische Wiskunde aan de Technische Universiteit Delft. Het onderzoek bestudeert de verbetering van A/B-testen met behulp van de bandit-methode.

Gedurende mijn studie heb ik vaak gezocht naar hoe de opgedane wiskundige kennis toe te passen is op economische onderwerpen. Dit eindproject was voor mij het perfecte antwoord op deze zoektocht. Het bevatte een combinatie van statistiek, data analyse en economie (marketing). Dit zijn voor mij de richtingen die ik na mijn bachelor ook graag op zou willen.

Ik wil graag mijn begeleider, Jakob Söhl, bedanken voor het altijd beantwoorden van mijn vragen en de deskundige feedback. De gedeelde interesse in het onderwerp maakte onze besprekingen boeiend en behulpzaam. Zonder zijn enthousiasme voor de statistiek was dit project niet mogelijk geweest.

Als laatste wil ik graag mijn dank uitbrengen naar Dorota Kurowicka en Emiel van Elderen voor het beoordelen van dit project.

Femke Lückerath
Juli 2019

Samenvatting

In dit onderzoek zijn twee manieren van A/B-testen met elkaar vergeleken. A/B-testen is het vergelijken van verschillende website versies om te achterhalen welke versie voor een hogere opbrengt zorgt. Consumenten krijgen afzonderlijk meerdere versies van een website te zien: versie A, versie B, versie C, et cetera. De websites verschillen op basis van één onderscheidend kenmerk van elkaar.
De twee manieren van A/B-testen zijn de klassieke methode en de meer recent ontwikkelde bandit-methode. In de klassieke methode van A/B-testen krijgen meerdere groepen van consumenten de verschillende versies van de website te zien en wordt achteraf bepaald welke website versie de betere versie is op basis van het aantal conversies (conversies zijn bijvoorbeeld aankopen, clicks, et cetera). Het nadeel van deze test is dat je pas achteraf weet welke website de hoogste opbrengst oplevert, terwijl deze uitkomst misschien al tijdens de steekproef duidelijk wordt.
De bandit-methode heeft meerdere varianten, waarvan vijf varianten in dit onderzoek zijn onderzocht en vergeleken. De bandit-methode van A/B-testen blijft gedurende de steekproef de groep consumenten die naar de verschillende versies van de website gestuurd worden aanpassen zodat al tijdens het testen zo min mogelijk consumenten naar de slechter presterende website gestuurd worden. Dit betekent namelijk misgelopen opbrengsten, ook wel spijt in A/B-testen. In dit onderzoek wordt daarom de klassieke methode vergeleken met een vijftal bandit-methoden.

Aan de hand van een gesimuleerde steekproef en drie fictieve versies van een website (A, B en C) worden de uitkomsten van de zes methoden op basis van statistische analyses vergeleken. De analyses zijn in RStudio geprogrammeerd en uitgevoerd. Hieruit blijkt dat, alhoewel alle zes de methoden uiteindelijk de best presterende versie kunnen aanwijzen, er duidelijke verschillen zichtbaar zijn in de totale conversies en de spijt. De bandit-methode verbetert op die onderdelen de klassieke methode. Daarnaast zijn er aanwijzingen dat de bandit-methoden de verschillende conversieratio's van de website versies eerder statistisch significant kunnen aantonen.

Inhoudsopgave

1 Inleiding

De online markt groeit. Volgens Thuiswinkel.org heeft in 2018 96% van de Nederlandse consumenten van 15 jaar en ouder een of meerdere online aankopen van producten en diensten gedaan [7]. Het is als bedrijf noodzakelijk mee te groeien in deze wereld en ervoor te zorgen dat hun websites pakkend zijn én dit ook blijven bij consumenten. Maar hoe zorg je hiervoor?

Door middel van A/B-testen kunnen bedrijven onderzoeken welke versie van hun website (versie A of versie B) zorgt voor meer aankopen en hierdoor een hogere omzet genereert. Zo leidt bijvoorbeeld het tonen van het Thuiswinkel Waarborg logo op een website (website versie *B*) tot 10.70% meer aankopen dan als dit logo niet getoond wordt (website versie A). [1] Het A/B-testen geeft dan relevante informatie voor de onderneming. Door de hoeveelheid data, die er verzameld wordt over het surfgedrag van de consumenten, wordt het gemakkelijker A/B-testen uit te voeren. Bedrijven merken op bij welke pagina de consument de website verlaat en wanneer een consument een aankoop doet.

Een bijzondere toepassing van A/B-testen is een bandit-test. Bandits zijn eenarmige gokmachines. Oorspronkelijk werd de bandit-test gebruikt voor verschillende statistische problemen. In 1952 beschreef Robbins het idee van 'sequential design' (continue aanpassingen), een nieuwe manier van het oplossen van statistische problemen waarbij je door onderzoek een betere, en snellere keuze kon maken tussen een of meerdere opties [5]. Dit was een voorloper van de bandit-test. 30 jaar later beschreven Berry en Fristedt (1985) [2] het bandit-probleem. In 2002 schreven Auer, Cesa-Bianchi en Fischer over bandit algoritmen, in die tijd gericht op het kiezen van de juiste gokmachines [1]. Pas later werden bandits geassocieerd met websites en A/B-testen, door onder andere White (2012) [8].

In deze tijd van online winkelen wordt in de zoektocht naar het verbeteren van A/B-testen onderzocht hoe bandit-testen toegepast kunnen worden op de keuze van consumenten op websites en hoe deze methode daardoor het A/B-testen kan verbeteren.

In dit onderzoek wordt er onderzocht of de bandit-methode inderdaad een verbetering is van de klassieke methode van A/B-testen. Daarnaast wordt aan de hand van gesimuleerde data, verschillende bandit-methoden vergeleken. Om de klassieke A/B-testen en de bandit-test te vergelijken zijn in RStudio de testen geprogrammeerd. Uit de resultaten volgt een duidelijk analyse. Het onderzoek legt een duidelijke brug tussen de klassieke methode en de bandit-methode van A/B-testen.

Dit onderzoek is als volgt opgebouwd. Hoofdstuk 2 beschrijft de opzet van A/B-testen en de aannames die hierbij gemaakt worden. Hoofdstuk 3 typeert twee verschillende toepassingen van de statistiek: statische en dynamische statistiek. In Hoofdstuk 4 en 5 worden de verschillende methoden van A/B-testen uitgelegd en geanalyseerd: in Hoofdstuk 4 de klassieke methode en in Hoofdstuk 5 de vijf varianten van de bandit-methode. Hoofdstuk 6 geeft de vergelijking tussen de klassieke methode en de bandit-methode. Tot slot volgt er een Conclusie en een Discussie met suggesties voor verder onderzoek.

[1] https://www.ism.nl/presentaties/de-aanpak-voor-succesvolle-ab-testen/, slide 7

2

A/B Testen

In dit onderzoek worden twee methoden van A/B-testen vergeleken, de klassieke methode en de bandit-methode. Voordat deze methoden in het volgende hoofdstuk nader onderzocht worden, beschrijft dit hoofdstuk de opzet van A/B-testen en de aannames die gemaakt worden voorafgaand aan het testen van verschillende versies.

A/B-testen wordt uitvoerig gebruikt bij het vergelijken van verschillende website versies. In dit onderzoek wordt bij het vergelijken van versies van een website eenzelfde webpagina bedoeld met één duidelijk verschil. Dit zou bijvoorbeeld bij de bestelpagina de kleur van de bestelknop kunnen zijn. Er wordt onderzocht of de kleur van deze knop een significante invloed heeft op de conversieratio van de website. Een conversie is alles wat een bedrijf voor de website als winstgevend beschouwt, denk aan aankopen, downloads of clicks. De conversieratio is vervolgens het aantal conversies ten opzichte van alle websitebezoekers. In dit onderzoek ligt de focus op aankopen.

A/B testen is in eerste instantie het testen van de invloed op de conversieratio van de onderscheidende eigenschappen in versie A en B van dezelfde website. Echter kunnen het aantal versies uitgebreid worden (tot n-websites) zodat A/B-testen vervangen wordt door A/B/n-testen. Hierbij worden meerdere versies getest met nog steeds maar één onderscheidend verschil in versies, bijvoorbeeld een groene, oranje of rode bestelknop.

A/B-testen kan op verschillende manieren uitgevoerd worden. In de volgende hoofdstukken worden de klassieke methode en verschillende varianten van de bandit-methode nader uitgelegd. Het doel van de A/B-testen is om via deze methoden erachter te komen welke versie van de website de meest gunstige invloed heeft op het conversieratio, waarbij de hogere conversieratio statistisch significant moet zijn om duidelijke uitspraken te doen.

Door middel van een simulatie worden de verwachte conversieratio's bepaald. Vervolgens wordt gekeken naar het aantal conversies per steekproefomvang per website versie.

Voordat de twee methoden uitgelegd, toegepast en vergeleken kunnen worden, zijn er een viertal aannames voor het vraagstuk gemaakt die voor beide methoden gelden. Hierdoor kunnen de methoden eenvoudig vergeleken worden onder dezelfde omstandigheden.

Deze aannames zijn als volgt:

1. **Onafhankelijke websitebezoekers.** Websitebezoekers beïnvloeden elkaar niet. Zij en hun keuzes zijn onafhankelijk van elkaar.
2. **Conversies hebben gelijke waarde.** Er wordt geen onderscheid gemaakt tussen een aankoop van 100 € en van 50 €. Elke conversie is evenveel waard.
3. **Unieke websitebezoeker op een uniek tijdstip.** Websitebezoekers zitten niet tegelijk op de website, ook al zien ze verschillende versies. Een websitebezoeker heeft daarmee zijn eigen tijdstip.

4. **Meerdere bezoeken per websitebezoeker.** Dezelfde websitebezoeker wordt opnieuw geregistreerd wanneer deze meerdere keren de website bezoekt.

3

Statistiek

Door middel van statistische technieken kunnen gegevens verzameld en onderzocht worden om hier vervolgens een conclusie uit te kunnen trekken. Uit een aselecte, gesimuleerde steekproef wordt data over het websitebezoek verzameld, bijvoorbeeld wat de gemiddelde conversieratio is bij de steekproeven van verschillende versies. Wanneer de verzamelde data een statistisch betrouwbare conclusie geeft over bijvoorbeeld de gemiddelde conversieratio, dan kan de gevonden uitslag gelden voor de gehele populatie behorende bij de steekproef.

Dit onderzoek gaat over (simulaties van-) steekproeven van conversies binnen verschillende versies van websites, waarna de uitkomsten van de verschillende methoden vergeleken worden.
In dit hoofdstuk worden twee verschillende vormen van de statistiek behandeld, namelijke statische en dynamische statistiek. Dit is voor dit onderzoek mede van belang omdat de statische en dynamische statistiek anders de omvang en de verdeling van een steekproef bepalen.

3.1. Statische Statistiek

Een model is statisch wanneer een model niet in beweging is en dus niet afhangt van de tijd. Bij statische statistiek is de steekproef datgene dat niet in beweging is. Daarmee wordt bedoeld dat de steekproefomvang en verdeling vooraf is vastgesteld en door het testen heen ook niet veranderd zal worden.
Voorafgaand aan een statistische test worden hypothesen opgesteld en bij elke hypothese wordt bepaald hoeveel data er nodig is om deze hypothesen tegen elkaar te kunnen testen. Het einddoel van een statistische test is het toetsen van de hypothesen. Verdere informatie over een hypothese toets is te vinden in Sectie 4.1. Hypothesen worden dan ook niet door de test heen veranderd.

3.2. Dynamische Statistiek

Dynamische statistiek houdt in dat de steekproefverdeling verandert gedurende de analyse van de steekproef. Robbins gaf dit idee aan in zijn artikel over 'sequential design' [5]. Volgens Robbins is de intentie van deze vorm van statistiek een vermindering van de gemiddelde steekproefomvang om zodoende de kans op een verkeerde beslissing terug te brengen naar een gewenst niveau. Het idee hierachter is dat als je sneller leert van de data die je verzameld, je eerder weet wat de juiste keuze is.
Belangrijk van dynamische statistiek is dat het model leert van eerdere waargenomen data uit de test. De statistische test werkt voortdurend de gemiddelde kansen bij. Hierdoor verandert ook de manier waarop de steekproef uitgevoerd moet worden.

Een dynamisch statistisch probleem definieert Robbins (p. 528) [5] als volgt "In het algemeen kunnen we methoden overwegen waarin waarnemingen één voor één worden gedaan, met de beslissing over uit welke populatie elke waarneming moet komen met de mogelijkheid om af te hangen van alle eerdere waarnemingen; de totale steekproefomvang N zou kunnen worden vastgesteld of zou een willekeurige variabele kunnen zijn, afhankelijk van de waarnemingen".

In dit onderzoek wordt er uitgegaan van een vooraf vastgestelde steekproefomvang, maar met een dynamische verdeling van deze steekproef.

4

De Klassieke Methode

In dit hoofdstuk zal A/B-testen aan de hand van de klassieke methode toegelicht worden. Met de klassieke methode wordt een statistische hypothese-test bedoeld, waarbij steekproefomvang en -verdeling vooraf vastgesteld zijn. Het model gaat uit van een nulhypothese en alternatieve hypothesen. In dit hoofdstuk worden deze hypothesen verduidelijkt en getoetst.

4.1. Hypothesen Testen

Het klassieke A/B-testen kan als een statistische hypothese toets gezien worden. Bij een statistische hypothese toets spreekt men van een nulhypothese, H_0, en een alternatieve hypothese, H_1. De nulhypothese is een stelling, iets wat men aanneemt als vanzelfsprekend. De alternatieve hypothese gaat deze stelling proberen te verwerpen. Het vraagstuk luidt ALS dit gebeurt, gebeurt DAN dat? En berust deze relatie op toeval of is deze statistisch significant?
Websites kunnen op deze manier ook getest worden. Zo kan er getest worden of de tijd dat mensen op een beginpagina blijven langer wordt als de achtergrondkleur groen is in plaats van blauw. Is de relatie tussen een groene achtergrondkleur en tijdsduur, D, significant? De hypothesen worden aldus:

H_0 : Er is geen verschil in tijdsduur tussen twee achtergrondkleuren, $D_{blauw} = D_{groen}$
H_1 : Een groene achtergrondkleur zorgt voor een langere tijdsduur, $D_{groen} > D_{blauw}$

Dit is een continu probleem, namelijk de tijdsduur, D, is een continue stochast.
Ook kan er op een website onderzocht worden of een foto toevoegen aan de beginpagina beduidend effect heeft op de conversieratio, k.

H_0 : Er is geen verschil tussen conversieratio wanneer er wel of geen extra foto wordt toegevoegd aan de beginpagina, $k_{geenfoto} = k_{welfoto}$
H_1 : Een extra foto op de beginpagina geeft een hogere conversieratio, $k_{welfoto} > k_{geenfoto}$

Dit is een binair probleem, omdat een testpersoon er alleen voor kan kiezen om 'wel' of 'geen' aankoop te doen.

Twee hypothesen worden tegen elkaar getest aan de hand van een steekproef. Bij een steekproef worden selecties van de steekproefomvang gestuurd naar de verschillende versies van een website. De steekproefomvang N en de manier waarop deze omvang verdeeld wordt, 50/50 of 30/70, worden vooraf bepaald.
Bij A/B-testen worden dan bijvoorbeeld 500 websitebezoekers eerlijk verdeeld over de twee verschillende beginpagina's. Ook kan men ervoor kiezen 30/70 te verdelen, als er al meer vertrouwen is in versie B. Dit is om het verlies te minimaliseren.

Een van de belangrijkste onderdelen van hypothese toetsen is het significantieniveau, α. Alleen bij statistische significantie kan de nulhypothese verworpen of geaccepteerd worden. Als het verschil in conversieratio na de steekproef tussen versie A en B niet significant is, dan wordt de nulhypothese aangenomen. Wanneer de uitkomst van de test wel onder het gekozen significantieniveau ligt, wordt de nulhypothese verworpen. De

steekproef geeft dan aan dat de kans op geen verschil (weergegeven als p) te klein is.

Voorafgaand aan een hypothese test wordt het betrouwbaarheidsinterval bepaald. Dit interval bepaalt het significantieniveau.
Omdat de steekproef werkt met een deel van de populatie, kan deze nooit 100% betrouwbaar zijn. En zelfs al zou er met een hele populatie getest worden, zijn er nog steeds kans op afwijkingen. Het betrouwbaarheidsinterval is het interval waarin betrouwbare schattingen van de uitkomst liggen. Meestal wordt er gekozen voor een 95 of 90 procent betrouwbaarheidsinterval.
Welk significantieniveau hoort bij welk betrouwbaarheidsinterval ligt aan het type hypothese-test. In dit onderzoek zal er gebruik gemaakt worden van de Z-test. Deze test is gebaseerd op de normale verdeling. Wanneer een steekproef mogelijk een andere verdeling heeft, zoals binomiaal, dan wordt deze met behulp van de centrale limietstelling omgeschreven naar een normale verdeling met een gemiddelde en een standaardafwijking. Deze stelling zegt dat voor een rij onafhankelijke stochasten X_i, met gemiddelde μ en standaardafwijking σ^2, de som $\frac{1}{n}\sum_{i=1}^{n} X_i$ naar de normale verdeling met gemiddelde μ en standaardafwijking $\frac{\sigma}{\sqrt{n}}$ convergeert [4].

Nadat de steekproef is uitgevoerd, kan de p-waarde berekend worden. De p-waarde is de kans dat de steekproef een extreme uitslag heeft, gegeven dat de nulhypothese waar is. Wanneer deze p-waarde onder het significantieniveau ligt, mag de nulhypothese verworpen worden.

In de volgende paragrafen worden het model en de resultaten van de klassieke methode van A/B-testen beschreven.

4.2. Model Beschrijving Klassieke Methode

Laat $X_{i,n}$ discrete stochasten zijn met i de versie en n de websitebezoeker. Er geldt

$$X_{i,n} = \begin{cases} 1 & \text{als persoon } n \textbf{ een} \text{ aankoop doet op website } i, \\ 0 & \text{als persoon } n \textbf{ geen} \text{ aankoop doet op website } i. \end{cases}$$

De stochastische variabelen $X_{i,j}$ zijn Bernoulli verdeeld en onafhankelijk. De totale successen van een versie i is gedefinieerd als $Y_i = \sum_{j=1}^{n_i} X_{i,j}$ en deze stochast is binomiaal verdeeld.
Stel er zijn twee versies van de beginpagina van een website: versie A en versie B, dan $i \in \{A, B\}$. Laat de foto het punt zijn waarop deze versies van elkaar verschillen. Versie B moet de conversieratio verbeteren ten opzichte van versie A. Zoals in de vorige sectie is gezien, worden de volgende hypothesen opgesteld:

H_0 : Er is geen verschil tussen conversieratio wanneer er wel of geen extra foto wordt toegevoegd aan de beginpagina, $k_A = k_B$
H_1 : Een extra foto op de beginpagina geeft een hogere conversieratio, $k_B > k_A$

Hierin is k_i de conversieratio voor versie i.

Voor het betrouwbaarheidsinterval wordt een interval van 95% gekozen met het bijbehorende significantieniveau $\alpha = 0.05$.

Wanneer meerdere versies van websites tegen elkaar getest worden, zijn er ook meerdere hypothese-testen. Er kunnen immers niet bij één hypothese-test meerdere alternatieve hypothesen opgesteld worden. Bovendien kan versie A niet alleen tegen versie B getest worden als ook versie C bij de steekproef hoort. Als gevolg zijn er dan n verschillende hypothese-testen. De hypothese-testen zijn bijvoorbeeld bij drie verschillende website versies

1. $H_0 : k_A = k_{BC}$
   $H_1 : k_A > k_{BC}$
2. $H_0 : k_B = k_{AC}$
   $H_1 : k_B > k_{AC}$

3. $H_0 : k_C = k_{AB}$
   $H_1 : k_C > k_{AB}$

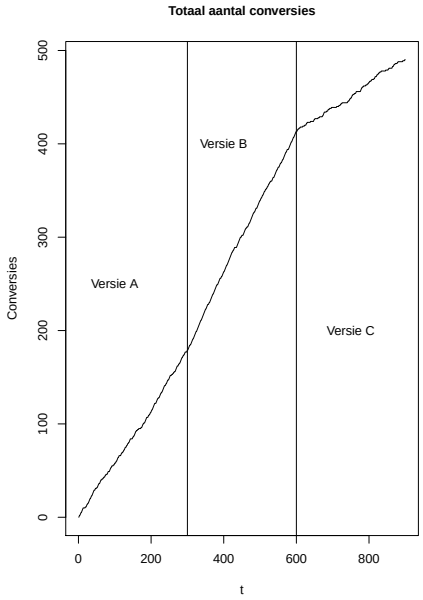
Hier staat k_{AC} voor de conversieratio als de conversies en websitebezoekers van versie A en C samengenomen worden.
Omdat het meerdere hypothese-testen zijn en de kans op onterecht de nulhypothese verwerpen groter is, wordt de Bonferroni correctie toegepast op het significantieniveau. Dit betekent dat het significantieniveau gedeeld wordt door het aantal hypothese-testen, in dit geval $\alpha = \frac{0.05}{3} = 0.01667$.

4.3. Resultaten Klassieke Methode

Het model beschreven in Sectie 4.2 is in RStudio geprogrammeerd en resulteert met werkelijke conversieratio's $k_A = 0.6$, $k_B = 0.8$ en $k_C = 0.3$ in de volgende verdeling

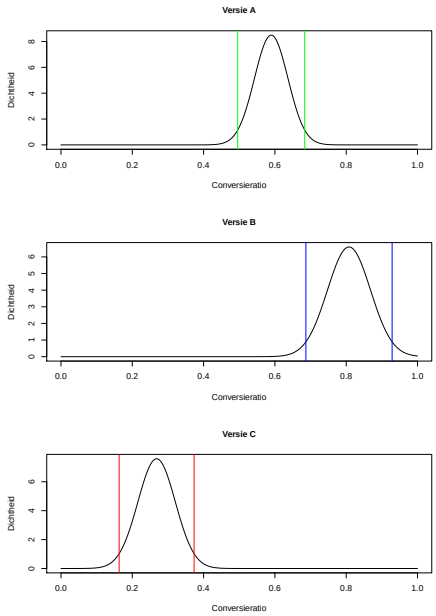
Conversie: Ja of Nee?	Versie A	Versie B	Versie C
Ja	179	234	77
Nee	121	66	223
Totaal	300	300	300
Ratio	$\hat{k}_A = 0.5967$	$\hat{k}_B = 0.78$	$\hat{k}_C = 0.2567$

waarbij $\hat{k}$ staat voor de geschatte conversieratio.
Figuur 4.1 toont het verloop van het totaal aantal conversies. De scheiding tussen de drie versies is goed waarneembaar door het verschil in helling van de grafiek.



Figuur 4.1: Totaal aantal conversies

Versie B geeft de meest steile lijn en dus de meeste conversies tijdens de steekproef.
De verdeling van alle drie de conversieratio's hebben een normale verdeling, zie Figuur 4.2. De bovengrens van versie A is gelijk aan 0.6836, terwijl de ondergrens van B gelijk is aan 0.6869. Hieruit blijkt dat versie B een hogere conversieratio heeft dan versie A. Het betrouwbaarheidsinterval van versie C ligt ook geheel buiten de intervallen van versie A en B. De conversieratio van versie C is hierdoor met 95% betrouwbaarheid lager dan de conversieratio's van A en B.



Figuur 4.2: Normale verdeling conversieratio's: Klassieke Methode

Om te testen of versie B significant voor meer conversies zorgt, worden de p-waarden van de drie hypothesentesten in RStudio berekend met de functie *prop.test()*.

1. $H_0 : k_A = k_{BC}$
   $H_1 : k_A > k_{BC}$
2. $H_0 : k_B = k_{AC}$
   $H_1 : k_B > k_{AC}$
3. $H_0 : k_C = k_{AB}$
   $H_1 : k_C > k_{AB}$

H_0 wordt verworpen als de p-waarde onder 0.01667 ligt.

Test nummer	p-waarde	H_0 verwerpen?
1	0.01564	Ja
2	$1.112 \cdot 10^{-23}$	Ja
3	≈ 1	Nee

Tabel 4.1: p-waarden Klassieke Methode

Uit Tabel 4.1 is te concluderen dat versie C met statistische significantie zorgt voor de minste conversies ten opzichte van versie A en B.

5

De Bandit-Methode

Met behulp van de bandit-methode, kunnen verschillende versies van websites tegen elkaar getest worden. In dit hoofdstuk zal deze methode toegelicht worden. Deze methode schakelt verschillende algoritmen in die trachten een optimale verdeling van de steekproef te bereiken. In dit hoofdstuk worden vijf van dit soort algoritmen behandeld en met elkaar vergeleken.

5.1. Definitie Bandit-methode

Bandits zijn eenarmige gokmachines, die ieder een eigen onafhankelijke uitbetalingsratio hebben. Dit kan vergeleken worden met versies van een website en de bijbehorende conversieratio. Een K-armig bandit probleem probeert de uitbetalingsratio's van K eenarmige gokmachines te achterhalen. Daarnaast probeert het probleem de totale opbrengsten van alle gokmachines zo hoog mogelijk te maken. Dit gebeurt door de juiste balans te vinden tussen het onderzoeken van alle gokmachines en het benutten van de opgedane kennis om zo snel mogelijk de opbrengst te optimaliseren. Er mag namelijk maar aan één hendel tegelijk getrokken worden. Om te achterhalen welke gokkast het beste presteert **moet** elke gokkast getest worden. Aan de andere kant moet, om een zo hoog mogelijke winst te krijgen, zo vaak mogelijk gekozen worden voor de betere kast. Wanneer namelijk zo vaak mogelijk voor de betere machine gekozen wordt, zal de spijt minimaal zijn. De spijt is het verlies (misgelopen opbrengsten) dat geleden wordt door niet de gehele tijd de beste machine te kiezen. De beste machine heeft namelijk een grotere kans op hogere opbrengsten en deze opbrengsten worden misgelopen. Hoe vindt het K-armige bandit-probleem de goede balans tussen **onderzoeken en benutten** om een zo hoog mogelijke winst te behalen?

5.2. Model Beschrijving Bandit-methode

In het geval van de websites wordt het probleem uitgedrukt in K website versies en de bijbehorende conversieratio. Deze ratio is voor iedereen onbekend, in tegenstelling tot een gokkast waarbij de uitbetalingsratio ingesteld kan worden. Het probleem zal vanaf hier uitgewerkt worden aan de hand van website versies en conversieratio's.
Laat $X_{i,n} \in \{0,1\}$ discrete stochasten zijn, waarbij $1 \leq i \leq K$ en $n \geq 1$. De variabele $X_{i,n}$ definieert de opbrengst van website versie i voor persoon n. Hierbij is persoon n voor versie i ongelijk aan persoon n voor versie j. Een persoon kan maar één versie van een website te zien krijgen. Omdat de conversieratio bepalend is voor de opbrengst, wordt met opbrengst wel of geen conversie bedoeld. Zodanig

$$X_{i,n} = \begin{cases} 1 & \text{als bij persoon } n \textbf{ een} \text{ conversie heeft plaatsgenomen op website } i, \\ 0 & \text{als bij persoon } n \textbf{ geen} \text{ conversie heeft plaatsgenomen op website } i. \end{cases}$$

Duidelijk te zien is dat $X_{i,n}$ en $X_{j,n}$ onafhankelijk zijn voor verschillende i en j. Het gaat namelijk om verschillende versies i en j.
Eveneens geldt dat $X_{i,n}$ en $X_{i,m}$ onafhankelijk zijn voor verschillende personen n en m. Deze stochasten zijn

voor vaste i Bernoulli verdeeld met kans k_i op een conversie. Dit wordt ook wel de conversieratio genoemd. Dit houdt in dat de verdeling van $Y_i = \sum_{n=1}^{n_i} X_{i,n}$ binomiaal is met dezelfde kans k_i. Hier is n_i het aantal keer dat versie i bezocht wordt. De conversieratio van website i, k_i, is in het begin nog onbekend, maar zal in de testperiode steeds meer uitgesproken worden.

De vraag is nu hoeveel personen naar elke website versie i gestuurd worden en in welke volgorde deze bezoekers naar de versies worden gestuurd.

Definieer de verzameling $N = \{\text{personen die de website bezoeken}\} \subseteq \mathbb{N}$. Er wordt verondersteld dat dit een stijgende verzameling is en dat personen niet tegelijk in de test zitten ofwel verschillende versies krijgen te zien. Er volgt $\#N = \sum_{i=1}^{K} n_i$. Voor elk persoon $m \in N$ wordt er gekozen naar welke versie i deze zal gaan met als opbrengst

$$X_{i,m} = \begin{cases} 1 & \text{met kans } k_i \\ 0 & \text{met kans } 1 - k_i \end{cases}$$

Het doel is een maximale winst, $\sum_{i=1}^{K} \sum_{n \leq n_i} X_{i,n}$.
Dat betekent ook het minimaliseren van de spijt. Dit is het verlies dat geleden wordt doordat niet continu de beste website wordt getoond. De spijt is als volgt gedefinieerd

$$\text{Spijt} = k^* \cdot N - \sum_{i=1}^{K} k_i * n_i$$

waarbij $k^* = max_{1 \leq i \leq K} k_i$ de beste conversieratio is.

Om de verdeling van N over de websites te bepalen zijn er algoritmen ontworpen, met het oog op een zo hoog mogelijk aantal conversies en een zo laag mogelijke spijt.

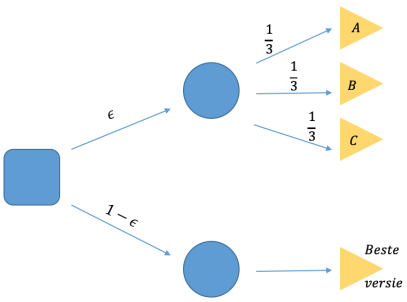
5.3. Bandit Algoritmen

Zoals beschreven in Sectie 5.1 maken bandit-methoden gebruik van eerdere uitkomsten, om zodoende gebruikers op een andere manier naar verschillende website versies te leiden.
Gebaseerd op bezoeken van eerdere gebruikers bepalen de volgende vijf bandit-algoritmen in welke volgorde de versies van de websites getoond zullen worden. Deze vijf algoritmen worden allereerst toegelicht, waarna ze op basis van de verdeling, de spijt en de betrouwbaarheid onderzocht worden.
De vijf algoritmen zijn het ϵ-greedy, ϵ_n-greedy, Upper Confidence Bound 1, Upper Confidence Bound 2 en Thompson Sampling algoritme.

Bij ieder algoritme is er een test uitgevoerd op drie versies met $N = 1000$ website bezoekers en conversieratio's $k_A = 0.6$, $k_B = 0.8$ en $k_C = 0.3$. De conversieratio's zijn alleen gebruikt om conversies te kunnen simuleren en zijn dus nog onbekend voor het algoritme.
De algoritmen kunnen ook uitgevoerd worden met onbekend totaal aantal websitebezoekers N. Deze moet dan voldoen aan een aantal voorwaarden, bijvoorbeeld wanneer de test minimaal 95% betrouwbaarheid is of wanneer het totaal aantal conversies boven een bepaald percentage ligt.

5.3.1. ϵ-greedy

Een greedy algoritme kiest voor elk moment de dan beste keuze. De afzonderlijke voorkeur wordt benut in de hoop op een algehele optimale opbrengst. Er zijn twee greedy algoritmen: ϵ-greedy (Sectie 5.3.1) en ϵ_n-greedy (Sectie 5.3.2).
Het ϵ-greedy algoritme kiest voor een ϵ onderzoeken en $1-\epsilon$ benutten verhouding. Er wordt met een kans ϵ onderzocht. Dit houdt in dat ϵ procent van de tijd een onderzoekende fase is. Wanneer het algoritme kiest om te onderzoeken, hebben alle K versies een even grote kans, $1/K$, om aan de betreffende persoon te worden getoond. Met een kans van $1-\epsilon$ wordt de versie met de op dat moment hoogste conversieratio, $\hat{k}^*(t) = \max_{1 \leq i \leq K} \hat{k}_i(t)$ met $\hat{k}_i(t)$ de geschatte conversieratio van versie i bij persoon t, getoond.
Bij zowel de onderzoeks- als de benuttingsfase wordt de conversieratio aangepast, wanneer bij een bezoeker wel of geen conversie plaatsvindt. Als tijdens de test duidelijk wordt dat de hoogste conversieratio gehaald wordt bij een andere versie, stapt het algoritme over om met een kans van $1-\epsilon$ deze versie te kiezen. In een beslissingsboom met $K = 3$ ziet dit er bijvoorbeeld als volgt uit:



Figuur 5.1: Beslissingsboom ϵ-greedy

Algoritme Kies een $0 < \epsilon < 1$.

Stap 1: Stuur een willekeurig aantal bezoekers naar elke versie als initialisatie stap. Hieruit kan voor elke versie i een conversieratio, $\hat{k}_i$, worden bepaald

$$\hat{k}_i = \#\{\text{conversies versie i}\}/n_i \tag{5.1}$$

Stap t: Laat δ een random getal zijn in $(0,1)$. En laat $\hat{k}^* = \max_{1\le i\le K} \hat{k}_i$. Wanneer

- $\delta < \epsilon$. Stuur persoon t met een kans van $1/K$ naar een willekeurige versie.
- $\delta \ge \epsilon$. Stuur persoon t naar de versie met $\hat{k}_i = \hat{k}^*$.

Update, nadat er wel of geen conversie is geweest, n_i, het aantal conversies en $\hat{k}_i$ voor de gekozen versie i.
Neem nu $t = t+1$ en herhaal stap t. Stop wanneer $t = N$.

5.3.2. ϵ_n-greedy

Het ϵ_n-greedy algoritme is een verbetering op het ϵ-greedy algoritme met het oog op een lagere spijt. Het idee is namelijk dat de waarde van ϵ_n daalt naarmate de waardes van de conversieratio's duidelijker worden. Op deze manier wordt de kans $1-\epsilon_n$ om de beste versie te kiezen groter en zal er dus ook een grotere kans zijn op meer conversies, bijgevolg een lagere spijt.

Algoritme Kies een $c > 0$.

Stap 1: Stuur n_i voor $1 \le i \le K$ bezoekers naar de versie i als initialisatie stap. Hieruit kan voor elke versie i een conversieratio, $\hat{k}_i$, worden bepaald.

$$\hat{k}_i = \#\{\text{conversies versie i}\}/n_i \tag{5.2}$$

Laat dan $\hat{k}^* = \max_{1\le i\le K} \hat{k}_i$.

Stap t: Laat $0 < d \le \min_{i:\hat{k}_i<\hat{k}^*}(\hat{k}^* - \hat{k}_i)$ en definieer

$$\epsilon_t = \min\{1, \frac{cK}{d^2 t}\} \tag{5.3}$$

.
Laat δ een random getal zijn in $(0,1)$. Wanneer

- $\delta < \epsilon_t$. Stuur persoon t met een kans van $1/K$ naar een willekeurige versie.
- $\delta \ge \epsilon_t$. Stuur persoon t naar de versie met $\hat{k}_i = \hat{k}^*$.

Update, nadat er wel of geen conversie is geweest, n_i, het aantal conversies en $\hat{k}_i$ voor de gekozen versie i. En $\hat{k}^* = \max_{1\le i\le K} \hat{k}_i$.
Neem nu $t = t+1$ en herhaal stap t. Stop wanneer $t = N$.

5.3.3. Upper Confidence Bound 1

Het Upper Confidence Bound 1 (UCB1) algoritme selecteert een versie op basis van de verwachte conversieratio plus een bovengrens. De vergelijking tussen versies is gebaseerd op $\hat{k}_i + b_i$. De bovengrens b_i voor versie i wordt bepaald door de volgende vergelijking

$$b_i = \sqrt{\frac{2 \cdot \ln(n)}{n_i}}, \tag{5.4}$$

waar n het totaal aantal bezoekers is en n_i het aantal bezoekers op website versie i.
Doordat de bovengrens afhangt van het totaal aantal pogingen krijgt uiteindelijk elke versie een kans om zich te bewijzen. Namelijk wanneer versie i vaker getest wordt, zal n_i groter worden en de bovengrens dus kleiner. Ook worden de bovengrenzen van de andere versies groter, omdat n groter wordt. Hierdoor wordt de grens voor een minder onderzochte versie groter en zal deze dus ook gekozen worden. Echter omdat in de UCB1 formule ook de verwachte conversieratio zit, zal de versie met de laagste ratio minder vaak gekozen worden al is de bovengrens, b_i, groter.
De natuurlijke logaritme in de bovengrens is een afnemende stijging als t groter wordt. Daarmee heeft de bovengrens een toenemende daling, immers $\ln(n)$ stijgt langzamer dan n_i. Op deze manier krijgen versies die in het begin weinig onderzocht zijn later meer kans.

Algoritme

Stap 1: Probeer elke versie 1 keer en bereken de conversieratio's $\hat{k}_i$.

Stap 2: Kies de versie met de hoogste waarde voor $\hat{k}_i + b_i = \hat{k}_i + 0$ voor de volgende stap, want de initialisatie stap wordt maar één keer meegenomen in n.

Stap t: Stuur persoon t naar versie i en kijk of er een conversie plaatsvindt. Update n, n_i en $\hat{k}_i$.
Voor alle $0 \leq i \leq K$, bereken de nieuwe bovengrens, b_i. Deze grens verandert namelijk voor alle versies, want n is groter geworden.
Kies de nieuwe versie i, waarvoor geldt $\hat{k}_i + b_i = \max_{0 \leq j \leq K}(\hat{k}_j + b_j)$. Neem nu $t = t + 1$ en herhaal stap t totdat $t = N$.

5.3.4. Upper Confidence Bound 2

Ook het Upper Confidence Bound 2 (UCB2) algoritme bepaalt de verdeling op basis van de verwachte conversieratio plus een bovengrens. Echter waar het UCB1 algoritme met het aantal websitebezoekers per versie i, n_i, werkt, gebruikt UCB2 algoritme het aantal tijdvakken per versie i, r_i. De tijdvakken zijn tijdvakken waarin een of meerdere websitebezoekers naar dezelfde versie worden gestuurd. Naar welke versie wordt bepaald door de verwachte conversieratio en een bovengrens. In het UCB2 algoritme is de bovengrens, b_i, gedefinieerd als

$$b_i = \sqrt{\frac{(1+\alpha)\ln(\frac{\exp(1)\cdot n}{\tau(r_i)})}{2 \cdot \tau(r_i)}} \tag{5.5}$$

waar n het totaal aantal bezoekers is.
In één tijdvak wordt de gekozen versie i $\tau(r_{i+1}) - \tau(r_i)$ keer getest. De τ functie hangt af van de gekozen $\alpha \in (0,1)$ en is aangeduid met

$$\tau(r) = \lceil (1+\alpha)^r \rceil \tag{5.6}$$

De ln en de τ functie zorgen er in de bovengrens voor dat vaak geteste versies een kleinere bovengrens krijgen en dus, tenzij ze naar verhouding een hoge conversieratio hebben, minder vaak getest zullen worden. De bovengrens wordt namelijk kleiner omdat $\ln(\tau(r_i))$ aanzienlijk langzamer stijgt dan $\tau(r_i)$, waardoor de breuk kleiner wordt.

Algoritme Kies een α met $0 < \alpha < 1$.

Stap 1: Laat $r_i = 0$ voor alle i, probeer elke versie één keer en bereken de conversieratio's $\hat{k}_i$.

Stap 2: Bereken $\hat{k}_i + b_i = \hat{k}_i + \sqrt{\frac{(1+\alpha)\ln(\frac{\exp(1)\cdot 1}{\tau(0)})}{2\cdot\tau(0)}}$, want de initialisatie stap wordt maar één keer meegenomen in n. Kies de versie met de hoogste waarde voor $\hat{k}_i + b_i$ voor de volgende stap.

Stap t: Stuur $\tau(r_i+1) - \tau(r_i)$ personen naar versie i en kijk of er conversies plaatsvinden. Update n en $\hat{k}_i$ en doe $r_i + 1$.
Voor alle $0 \leq i \leq K$, bereken de nieuwe bovengrens, b_i. Deze grens verandert namelijk voor alle versies, want n is groter geworden.
Kies de nieuwe versie i, waarvoor geldt $\hat{k}_i + b_i = \max_{0\leq j\leq K}(\hat{k}_j + b_j)$. Neem nu $t = t+1$ en herhaal stap t totdat $t = N$.

5.3.5. Thompson Sampling

Thompson Sampling is gebaseerd op de Bayesiaanse statistiek, waarbij wordt gewerkt met een 'prior' verdelingsfunctie en een 'posterior' verdelingsfunctie.
De 'prior' functie is een beta verdeling met variabelen α en β en ziet er als volgt uit

$$\mathbb{P}(k) = \frac{k^{\alpha-1}\cdot(1-k)^{\beta-1}}{B(\alpha,\beta)}$$

waarbij $B(\alpha,\beta)$ de beta functie is.
De variabele α en β staan respectievelijk voor het aantal wel of geen conversies. Zo geeft bijvoorbeeld $\alpha = 2$ en $\beta = 5$ dat er van de 7 websitebezoekers 2 mensen wel een conversie plaatsen en 5 niet.
De α en β in de 'prior' functie zijn geschatte waarden en de bedoeling is dat deze variabelen aan de hand van geteste data aangepast worden zodat het representatieve waarden worden. Dit wordt ook wel de 'posterior' functie genoemd.
De 'prior' verdelingsfunctie bij Bernoulli bandits wordt als volgt aangepast [6]:

$$\alpha_i = \alpha_i + \begin{cases} 1 & \text{als er } \textbf{een} \text{ conversie is geplaatst} \\ 0 & \text{als er } \textbf{geen} \text{ conversie is geplaatst} \end{cases}$$

$$\beta_i = \beta_i + \begin{cases} 0 & \text{als er } \textbf{een} \text{ conversie is geplaatst} \\ 1 & \text{als er } \textbf{geen} \text{ conversie is geplaatst} \end{cases}$$

Aan het eind van de test zullen de α en β representatieve waarden zijn.

Algoritme

Stap 1: Schat α_i en β_i voor elke versie i.

Stap t: Bepaal voor elke versie i aan de hand van de beta verdeling met bijbehorende α_i en β_i een willekeurige conversieratio $\hat{k}_i$. Stuur persoon t naar versie i waarvoor geldt $\hat{k}_i = \max_{1\leq j\leq N}\hat{k}_j$. Update, nadat er wel of geen conversie is geweest, de variabelen op de volgende manier

$$\alpha_i = \alpha_i + \begin{cases} 1 & \text{als er } \textbf{een} \text{ conversie is geplaatst} \\ 0 & \text{als er } \textbf{geen} \text{ conversie is geplaatst} \end{cases}$$

$$\beta_i = \beta_i + \begin{cases} 0 & \text{als er } \textbf{een} \text{ conversie is geplaatst} \\ 1 & \text{als er } \textbf{geen} \text{ conversie is geplaatst} \end{cases}$$

Neem nu $t = t+1$ en herhaal stap t totdat $t = N$.

5.4. Resultaten Bandit Algoritmen

Om een goede interpretatie te krijgen van de vijf algoritmen zijn deze geprogrammeerd in RStudio.
Aan alle vijf de algoritmen is dezelfde vaste steekproefomvang $N = 1000$ toegewezen, evenals het aantal versies (A, B, C) en de conversieratio's van de versies (respectievelijk $0.6, 0.8, 0.3$).
Voor een geschikte vergelijking tussen de algoritmen zijn de vectoren O_i voor elke versie i geconstrueerd. De vector O_i geeft N observaties bij versie i weer. De observaties betekenen of er wel of geen conversie heeft plaatsgevonden. Op deze manier krijgt elk algoritme dezelfde observaties en zal de fout in het vergelijken van algoritmen verkleind worden. De vergelijking wordt na de resultaten geanalyseerd in Sectie 5.5.

De algoritmen zullen op de volgende factoren onderzocht worden:

- De verdeling van de websitebezoekers over de drie versies

- De spijt (misgelopen conversies)

- De betrouwbaarheid

Bij de verdeling van de websitebezoekers van elk algoritme kan er gekeken worden naar welke versie de t'de websitebezoeker wordt gestuurd. Vervolgens wordt er waargenomen of er wel of geen conversie wordt gedaan. Hieruit kunnen het aantal conversies afgeleid worden na t websitebezoekers. Evenals de spijt na t websitebezoekers die gedefinieerd is als

$$k^* - \sum_{i=1}^{3} (k_i \cdot n_i).$$

De spijt is een theoretische waarde en is in de praktijk niet bekend. Het is een vermoeden van het aantal misgelopen conversies. In het onderzoek van de algoritmen zijn de bekende conversieratio's k_i gebruikt om de spijt te berekenen.
Om de betrouwbaarheid van de algoritmen te testen zijn de veranderingen van de conversieratio's opgeslagen na t websitebezoekers. Er is immers meer uit te spreken over de conversieratio's wanneer er meer websitebezoekers t naar de versies zijn gestuurd. Met de data die verzameld is tijdens de algoritmen kunnen er normale verdelingen voor elke conversieratio opgesteld worden. De data heeft namelijk een gemiddelde, μ, en een standaardafwijking, σ^2. Om 95% betrouwbaarheidsintervallen vast te stellen zijn er in onderstaande grafieken de lijnen voor twee standaarddeviaties ($\mu \pm 2 * \sigma^2$) toegevoegd.
Voor de betrouwbaarheid van de testen met betrekking tot de conversieratio's zijn ook p-waarden bepaald van drie hypothesen testen. (Voor toelichting over hypothese-testen zie Sectie 4.1.) De drie hypothese-testen zijn

1. $H_0 : k_A = k_{BC}$
   $H_1 : k_A > k_{BC}$

2. $H_0 : k_B = k_{AC}$
   $H_1 : k_B > k_{AC}$

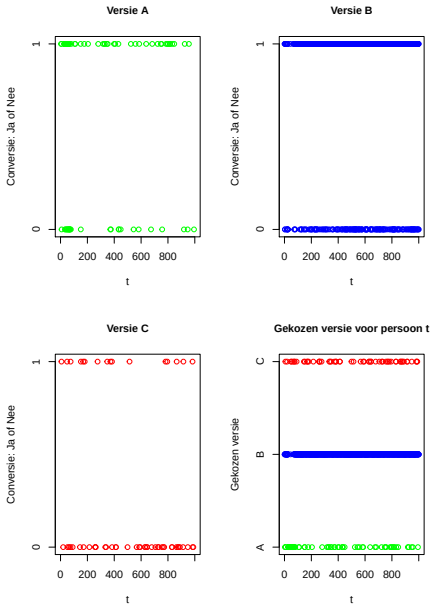
3. $H_0 : k_C = k_{AB}$
   $H_1 : k_C > k_{AB}$

Er is dus getest voor drie verschillende versies met voor versie A een conversieratio van 0.6, voor versie B 0.8 en voor versie C 0.3. De gesimuleerde steekproef van alle algoritmen bestaat uit 1000 website bezoekers.

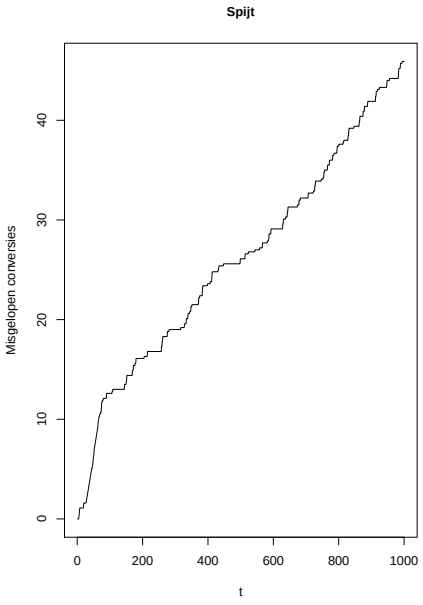
5.4.1. ϵ-greedy

Voor dit algoritme is er gekozen voor een ϵ met waarde van 0.15. Dit betekent dat er 15 procent van de tijd onderzocht zal worden.

Verdeling Het ϵ-greedy algoritme is tot onderstaande verdeling, zie Figuur 5.2, gekomen met bijbehorende conversies.
Dit figuur (ook bij de andere vier algoritmen) toont steeds allereerst per website het aantal keer conversie ja of nee en vervolgens (rechtsonder) naar welke versie websitebezoeker t toe is gestuurd.
In het figuur is rechtsonder te zien dat versie B het vaakst gekozen wordt. De versie heeft namelijk de meeste websitebezoekers.
Omdat het algoritme met een kans van 0.85 de betere versie kiest, is te concluderen dat het algoritme versie B als betere versie beschouwt.



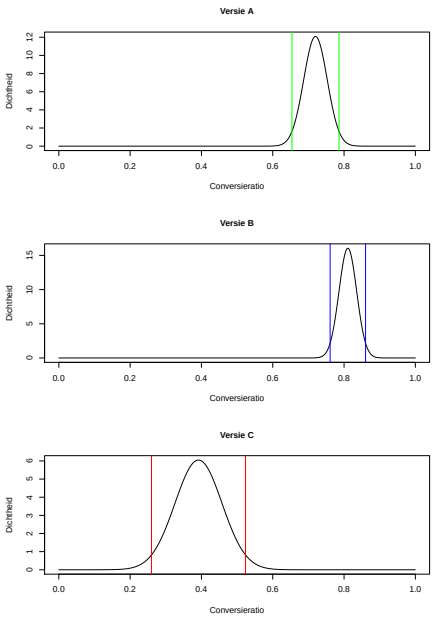
Figuur 5.2: Gekozen versie met bijbehorende conversie: ϵ-greedy



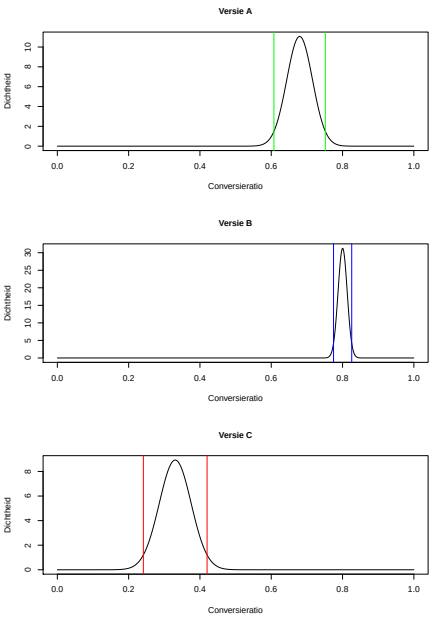
Figuur 5.3: Spijt: ϵ-greedy

Spijt Het ϵ-greedy algoritme geeft een nagenoeg lineaire spijt, Figuur 5.3. Dit komt door de ϵ procent onderzoeksfase. In deze onderzoeksfase wordt namelijk maar met een kans van $\frac{1}{3}$ de betere versie gekozen.

Betrouwbaarheid In Figuur 5.4 is te zien dat de conversieratio's van versie A en B duidelijk verschillen met de conversieratio van versie C, aangezien het betrouwbaarheidsinterval van C geheel buiten de intervallen van A en B vallen. Echter is ook te zien dat de intervallen van A en B overlappen. Er is nog niet met zekerheid te zeggen dat A statistisch significant beter of slechter werkt dan B.



Figuur 5.4: Normale verdeling conversieratio's $N = 1000$: ϵ-greedy



Figuur 5.5: Normale verdeling conversieratio's $N = 2000$: ϵ-greedy

Wanneer de steekproefomvang verhoogd wordt naar $N = 2000$ is er met meer betrouwbaarheid te zeggen dat versie B een hogere conversieratio heeft dan versie A en C, zie figuur 5.5.

De ondergrens van de conversieratio verdeling van versie B, 0.774697, ligt boven de bovengrens van de conversieratio verdeling van versie A, 0.7516991.

Ook achteraf aan de bandit-methode kunnen de p-waarden door RStudio berekend worden. Tabel 5.1 toont de p-waarden van de drie hypothese-testen en hoort bij de steekproefomvang $N = 1000$.

Test nummer	p-waarde	H_0 verwerpen?
1	0.8996	Nee
2	$1.679 \cdot 10^{-11}$	Ja
3	≈ 1	Nee

Tabel 5.1: p-waarden: ϵ-greedy

Hieruit volgt dat alleen de nulhypothese $H_0 : k_B = k_{AC}$ verworpen mag worden en dat er dus significant verschil is tussen versie B en $A + C$.

5.4.2. ϵ_n-greedy

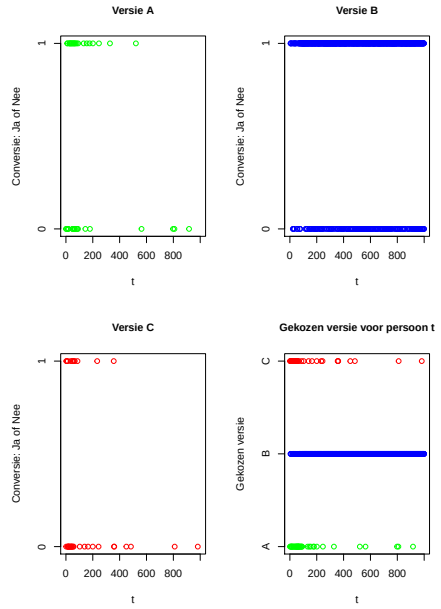
Bij het ϵ_n-greedy algoritme hangt de ϵ_t af van de variabele c en t, namelijk

$$\epsilon_t = \min\{1, \frac{cK}{d^2 t}\} \tag{5.7}$$

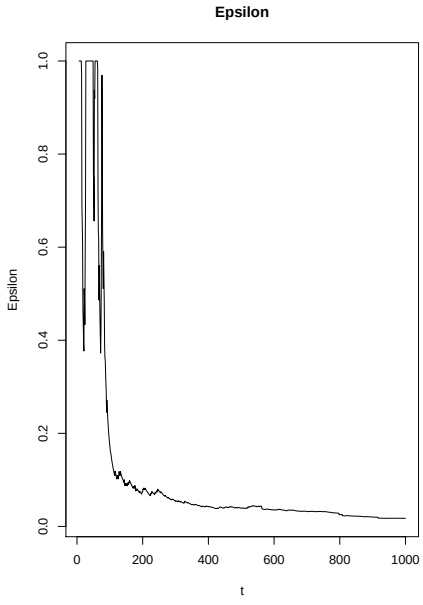
Zie Sectie 5.3.2 voor de uitleg van de vergelijking.

Er is gekozen voor c gelijk aan 0.3 en d gelijk aan $\min_{1 \leq j \leq 3}(\hat{k} * - \hat{k}_i)$. Deze d verandert dus bij elke stap t, omdat de verwachte conversieratio's $\hat{k}_i$ door de test heen veranderen.

Verdeling Het ϵ_n-greedy algoritme zorgt voor de verdeling, die is afgebeeld in Figuur 5.6, van 1000 website bezoekers.



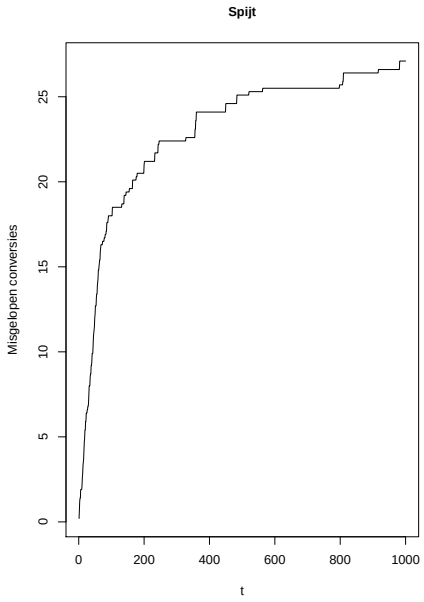
Figuur 5.6: Gekozen versie met bijbehorende conversie: ϵ_n-greedy



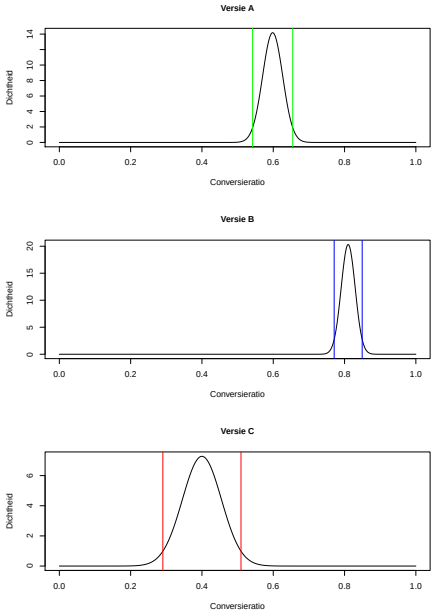
Figuur 5.7: Waarde van ϵ_n-greedy

Het verschil in gekozen versies tijdens de eerste 200 bezoekers bevestigt de werking van het algoritme. In het begin van de test is de ϵ_n namelijk groter, waardoor er met een grotere kans onderzocht wordt en er dus verschillende versies bekeken worden. Bij de 150e websitebezoeker heeft ϵ_n een waarde van 0.180068. Het verloop van ϵ_n is weergegeven in Figuur 5.7.

Spijt De spijt van het ϵ_n-greedy algoritme is in Figuur 5.8 afgebeeld. Er is te zien dat tot websitebezoeker 150 de spijt een lineaire groei heeft. Wanneer de ϵ_n zodanig klein begint te worden, neemt de helling van de spijt af. De afname van de spijt komt overeen met het verloop van de ϵ_n in Figuur 5.7.



Figuur 5.8: Spijt: ϵ_n-greedy



Figuur 5.9: Normale verdeling conversieratio's: ϵ_n-greedy

Betrouwbaarheid De normale verdeling van de data over de conversieratio's verzameld door het ϵ_n-greedy algoritme geeft de grafieken, weergegeven in Figuur 5.9. Het figuur geeft duidelijk aan dat de conversieratio van versie B met 95% betrouwbaarheid hoger is dan de conversieratio van versie A en van versie C.

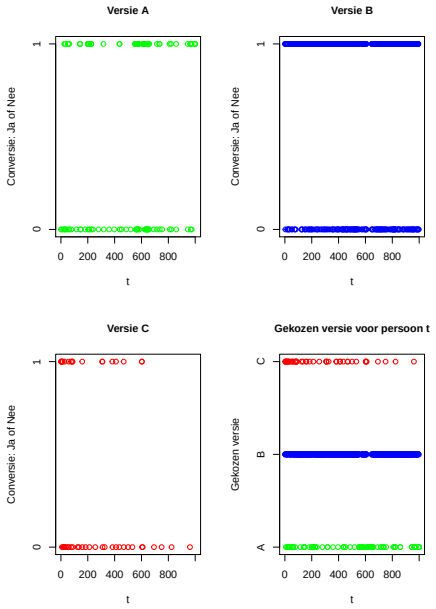
De p-waarden behorende bij de hypothese-testen van het ϵ_n-greedy algoritme staan in Tabel 5.2. De tabel steunt de conclusie van de normale verdelingen uit Figuur 5.9. Namelijk dat alleen $H_0 : k_B = k_{AC}$ verworpen mag worden, en dat er significant verschil is tussen de versies B en $A + C$.

Test nummer	p-waarde	H_0 verwerpen?
1	0.9997	Nee
2	$1.734 \cdot 10^{-13}$	Ja
3	≈ 1	Nee

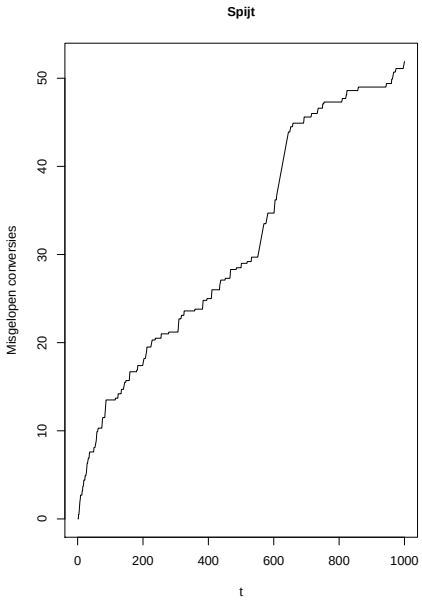
Tabel 5.2: p-waarden: ϵ_n-greedy

5.4.3. Upper Confidence Bound 1

Verdeling Hoewel het algoritme duidelijk maakt dat het versie B het vaakst als de betere versie ziet, kan uit onderstaand figuur ook opgemaakt worden dat versie A vaak getest is. Dit kan verklaard worden doordat de conversieratio's relatief dichtbij elkaar liggen. Wanneer het totaal aantal bezoekers van versie B, n_B, groter wordt, wordt de bovengrens b_B kleiner ten opzichte van de bovengrens van versie A. Hierdoor zal versie A ook snel gekozen worden, totdat het verschil in conversieratio's tussen versie A en B te groot wordt en B duidelijk een hogere ratio geeft.



Figuur 5.10: Gekozen versie met bijbehorende conversie: UCB1



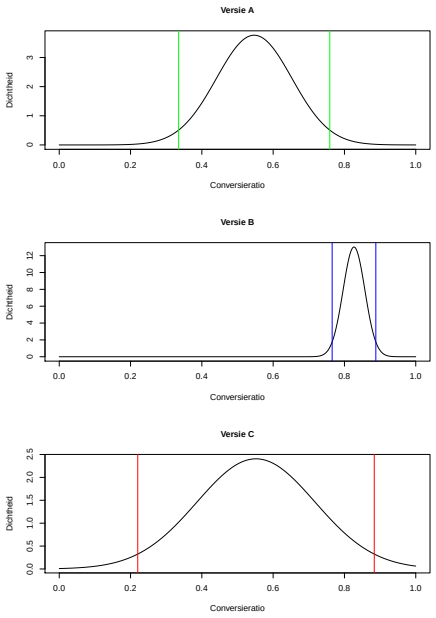
Figuur 5.11: Spijt: UCB1

Spijt De spijt van het Upper Confidence Bound 1 algoritme heeft het verband gegeven in Figuur 5.11. De bovengrens van de spijt volgens P. Auer [1] wordt gegeven door

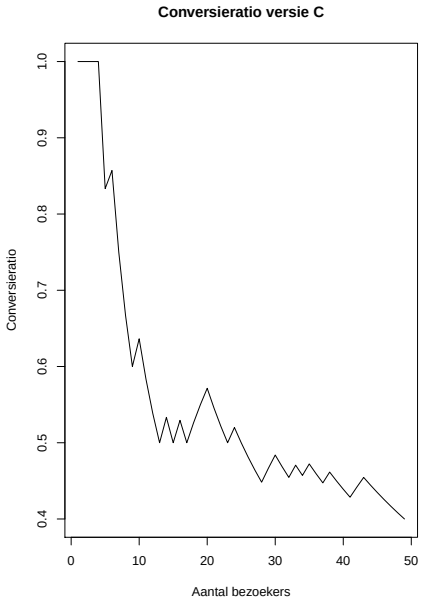
$$[8 \cdot \sum_{i:k_i<k^*} (\frac{\ln(n)}{(k^*-k_i)})] + (1+\frac{\pi^2}{3})(\sum_{j=1}^{K}(k^*-k_j)) \tag{5.8}$$

De spijt van het UCB1 algoritme met deze data ligt inderdaad onder deze bovengrens.

Betrouwbaarheid De bovengrens van het 95% betrouwbaarheidsinterval van versie A is 0.7588 en de ondergrens van de conversieratio van versie B is 0.7655. Deze grenzen zijn te herkennen in Figuur 5.12.



Figuur 5.12: Normale verdeling conversieratio's: UCB1



Figuur 5.13: Conversieratio versie C

Er kan met statistische significantie gezegd worden dat versie B een hogere conversieratio heeft dan versie A. Over de conversieratio van versie C kunnen nog weinig sterke uitspraken gedaan worden. Het interval is nog te groot. Figuur 5.13 laat zien dat de conversieratio van versie C convergeert naar 0.4, wat de reden is dat versie C in de loop van het algoritme minder vaak gekozen wordt.

De p-waarden van de hypothese-testen van UCB1, Tabel 5.3, geven hetzelfde resultaat als bij ϵ_n-greedy en ondersteunen de conclusie getrokken uit Figuur 5.12. Met statistische significante mag het verschil in conversieratio's tussen versie B en $A+C$ vastgesteld worden.

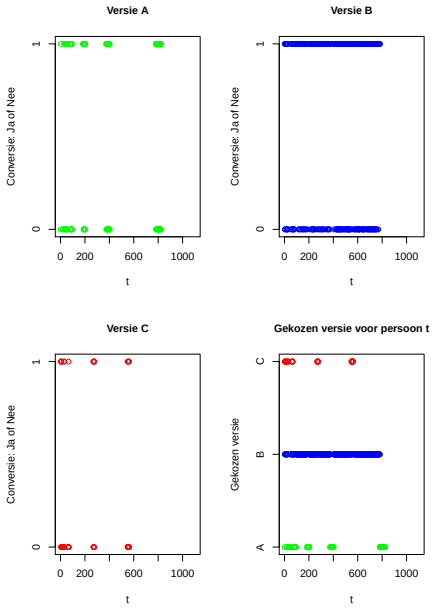
Test nummer	p-waarde	H_0 verwerpen?
1	0.9999	Nee
2	$2.217 \cdot 10^{-11}$	Ja
3	≈ 1	Nee

Tabel 5.3: p-waarden: UCB1

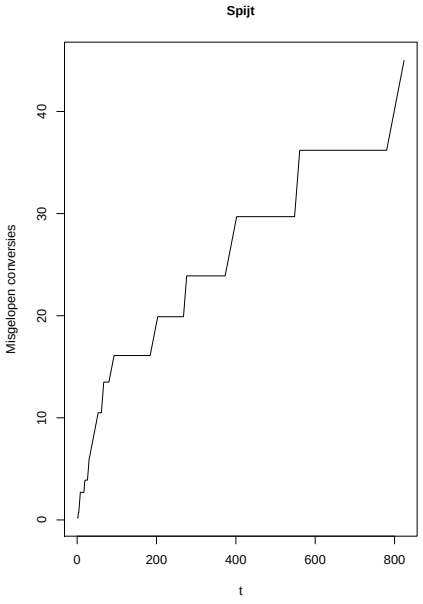
5.4.4. Upper Confidence Bound 2

Voor het Upper Confidence Bound 2 algoritme is er gekozen voor een α gelijk aan 0.5.

Verdeling In Figuur 5.14 is de verdeling van 824 websitebezoekers weergegeven. Omdat het laatste tijdvak meer dan 176 bezoekers zou tellen en dus over de 1000 bezoekers gaan, is deze niet in de data meegenomen. In het figuur zijn de tijdvakken goed op te merken, namelijk er is weinig wisseling tussen de versies.



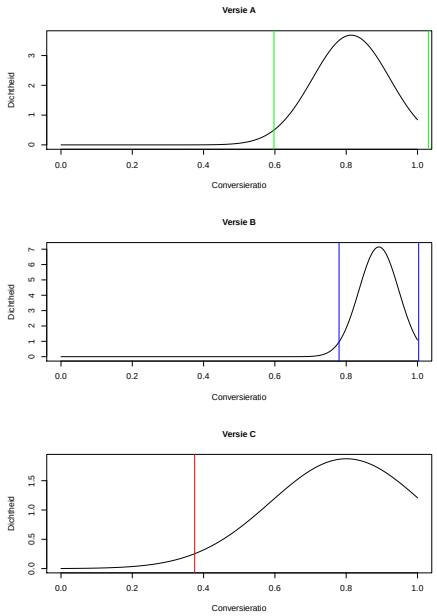
Figuur 5.14: Gekozen versie met bijbehorende conversie: UCB2



Figuur 5.15: Spijt: UCB2

Spijt Wanneer de spijt van het UCB2 algoritme in kaart gebracht wordt, zijn de tijdvakken duidelijk te herkennen, zie Figuur 5.15. De tijdperken van versie B zijn goed waar te nemen in Figuur 5.15, namelijk de horizontale stukken van de spijt. Er is ook te zien dat deze horizontale stukken langer worden naarmate er meer websitebezoekers zijn getest. Dit komt omdat het verschil tussen $\tau(r_i+1)$ en $\tau(r_i)$ van versie B steeds groter wordt en versie B daarom langer getest wordt.

Betrouwbaarheid Over de betrouwbaarheid van de conversieratio's is weinig te zeggen. In Figuur 5.16 hebben alle drie de versies alleen een ondergrens, waarbij de ondergrenzen dichtbij de daadwerkelijke conversieratio's liggen.



Figuur 5.16: Normale verdeling conversieratio's: UCB2

Bijbehorende p-waarden zijn weergegeven in Tabel 5.4. Uit deze p-waarden is af te lezen dat de conversieratio van versie B statistisch significant verschilt van de conversieratio's van versie A en versie C.

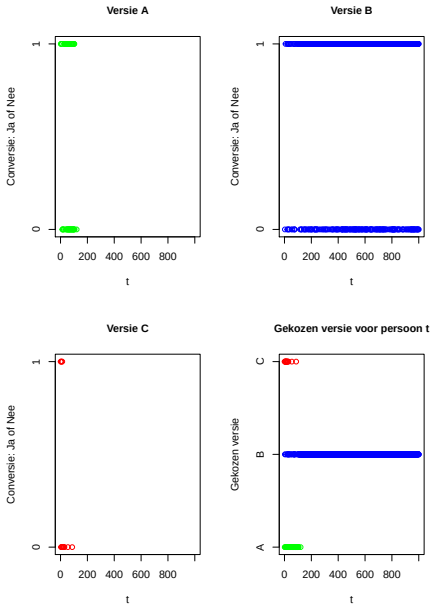
Test nummer	p-waarde	H_0 verwerpen?
1	0.9999	Nee
2	$1.365 \cdot 10^{-9}$	Ja
3	≈ 1	Nee

Tabel 5.4: p-waarden: UCB2

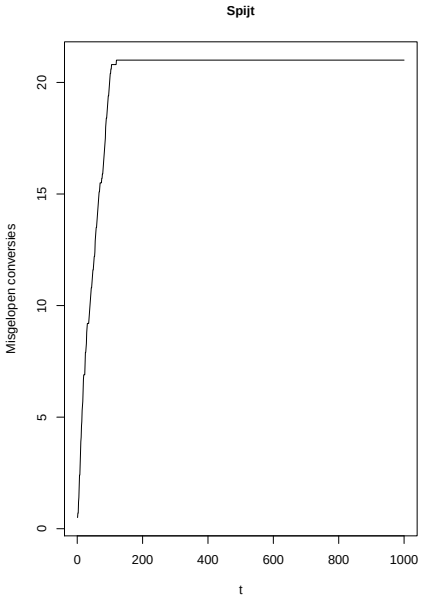
5.4.5. Thompson Sampling

α en β hebben voor alle drie de versies dezelfde waarde, te weten $\alpha_i = 1$ en $\beta_i = 5$ voor alle i. Op deze manier hebben de drie versies dezelfde kans in het begin gekozen te worden.

Verdeling Na minder dan 200 website bezoekers kiest het algoritme al voor versie B. De α_B en β_B zijn dan al zodanig hoog dat deze waarden de beta verdelingen van versie A en C overschaduwen.



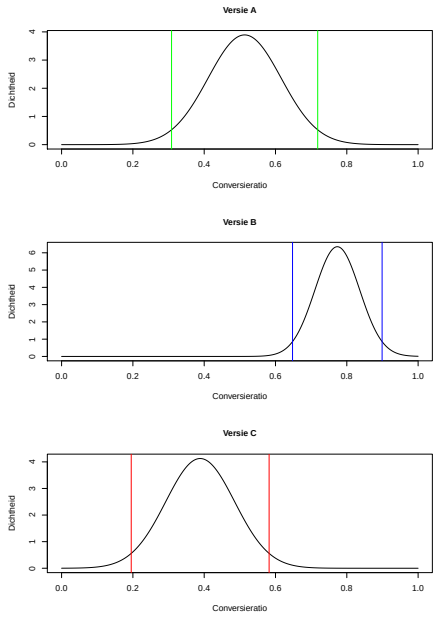
Figuur 5.17: Gekozen versie met bijbehorende conversie: Thompson Sampling



Figuur 5.18: Spijt: Thompson Sampling

Spijt Uit Figuur 5.17 is de spijt af te lezen, omdat duidelijk is dat versie B al snel voor de gehele tijd gekozen wordt. Figuur 5.18 bevestigt de gedachte dat de spijt na ongeveer 200 bezoekers niet meer zal stijgen.

Betrouwbaarheid Figuur 5.19 toont de betrouwbaarheid van de conversieratio's. Hieruit kan geconcludeerd worden dat versie B met 95% zekerheid een hogere conversieratio heeft dan versie C. Over het verschil tussen de conversieratio's van versie A en versie B of over het verschil tussen versie A en C is nog zekerheid te geven.



Figuur 5.19: Normale verdeling conversieratio's: Thompson Sampling

Uit Tabel 5.5 kan wel met zekerheid gezegd worden dat de conversieratio B significant verschilt van de conversieratio's van versie A en C, namelijk $1.948 \cdot 10^{-10} < 0.01667$.

Test nummer	p-waarde	H_0 verwerpen?
1	0.9999	Nee
2	$1.948 \cdot 10^{-10}$	Ja
3	≈ 1	Nee

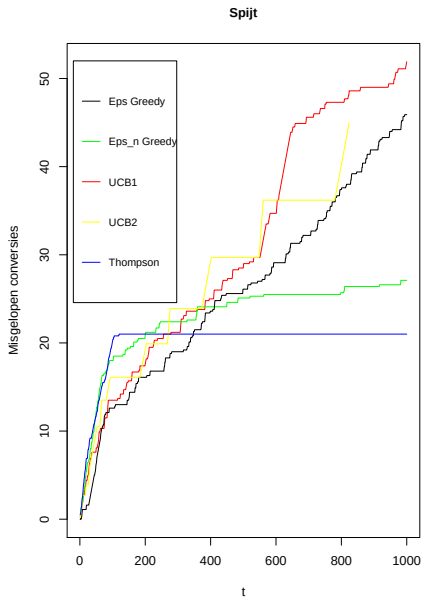
Tabel 5.5: p-waarden: Thompson Sampling

5.5. Vergelijking Bandit Algoritmen

In Secties 5.3 en 5.4 zijn de verschillende bandit-algoritmen uitgelegd en bestudeerd. Nu zullen de algoritmen met elkaar vergeleken worden. Ze worden geanalyseerd aan de hand van spijt en de werking van het algoritme.

5.5.1. Spijt

In Figuur 5.20 zijn alle spijt figuren samengevoegd om een goede vergelijking te kunnen maken.



Figuur 5.20: Spijt $N = 1000$: alle algoritmen

De laagste spijt is behaald door het Thompson Sampling algoritme. Dit algoritme heeft namelijk vanaf websitebezoeker 121 alleen maar versie B gekozen. Hierdoor is de spijt alleen toegenomen bij de eerste 120 websitebezoekers. Echter was de betrouwbaarheid van het Thompson Sampling algoritme moeilijk te bepalen na 1000 bezoekers. Dit komt omdat de α en β van een goede versie met gehele getallen stijgen en dus ook ver kunnen uitlopen van de slechtere versies. De eind α's en β's van de versies waren:

Versie	α	β
A	35	36
B	723	202
C	5	17

Hierna bereikt het ϵ_n-greedy een stabiele ontwikkeling. Figuur 5.7 bekrachtigt de stabiliteit van de spijt, want in het figuur is te zien dat de waarde van ϵ_n dicht naar de 0 gaat, wat betekent dat er voor 100% van de tijd websitebezoekers naar de beste versie gestuurd zullen worden.
Bovendien kon uit de verdelingen van de conversieratio's opgemaakt worden met 95% betrouwbaarheid dat versie B daadwerkelijk de beste versie was.

Wat opvalt in Figuur 5.20 is dat beide Upper Confidence Bound algoritmen de slechtste spijt geven. Dit komt

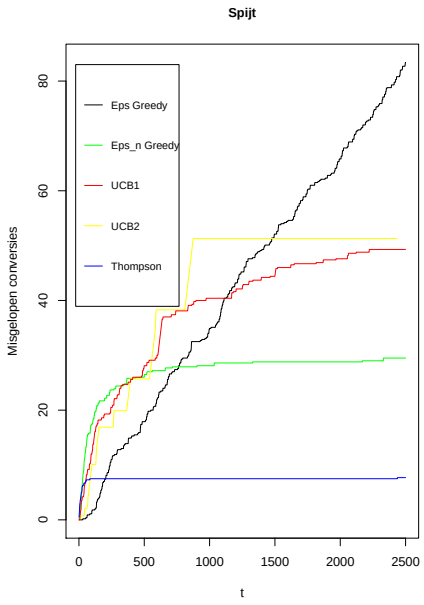
omdat de algoritmen beide rekening houden met de slechtere conversieratio's. Omdat de algoritmen de bovengrenzen van veel geteste versies omlaag brengen, kunnen minder vaak geteste versies ook getest worden, mits ze niet een relatief lage verwachte conversieratio hebben. In Figuren 5.10 en 5.14 is op te merken dat alle drie de versies vaak getest worden.

Algoritme	A	B	C	$A+C$
ϵ-greedy	87	856	57	144
ϵ_n-greedy	43	920	37	80
UCB1	137	814	49	186
UCB2	130	656	38	168
Thompson Sampling	65	919	16	81

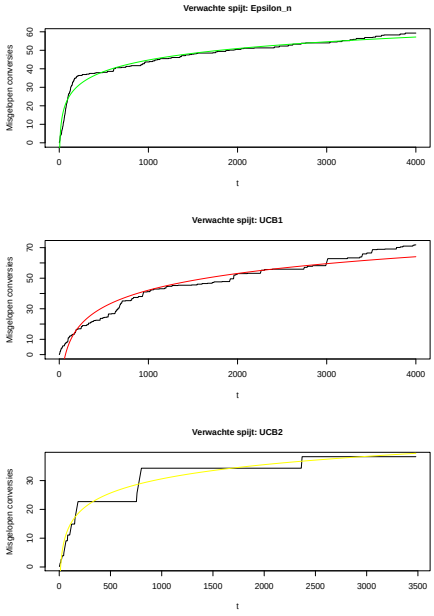
Tabel 5.6: Aantal bezoekers per website versie per algoritme

Tabel 5.6 geeft ook aan dat UCB1 en UCB2 het meeste versie A en C onderzoeken, waarbij UCB2 zelfs maar een totaal van 824 websitebezoekers had. Dit verklaart de hogere spijt.

Echter wanneer de steekproefgrootte verhoogd wordt naar 2500 websitebezoekers, stijgt de spijt van het ϵ-greedy algoritme steiler dan het UCB1 en UCB2 algoritme, zie Figuur 5.21.



Figuur 5.21: Spijt $N = 2500$: alle algoritmen



Figuur 5.22: Verwachte spijt $N = 4000$: ϵ_n-greedy, UCB1 en UCB2

Lai en Robbins [3] bewezen deze logaritmische groei al in 1985, namelijk dat de spijt een logaritmisch verband heeft met steekproefomvang. Auer et al. [1] versterkten dit inzicht en toonden aan dat dit geldt voor ϵ_n-greedy, UCB1 en UCB2.
Met behulp van linear models in RStudio kan het verband tussen spijt en aantal websitebezoekers van elk algoritme achterhaald worden.
Omdat er al geconcludeerd was dat het ϵ-greedy algoritme een lineaire spijt geeft en dat het Thompson Sampling algoritme al de optimale versie benut, ligt de interesse nog bij het ϵ_n-greedy, UCB1 en UCB2 algoritme en hun logaritmische groei. Figuur 5.22 geeft de verwachte groei van de spijt en de feitelijke spijt van de drie algoritmen bij een steekproefomvang van $N = 4000$ websitebezoekers. In het figuur zijn de gekleurde lijnen de gepaste spijt. De drie ontwikkelingen van de spijt zijn in de volgende vergelijkingen vast te leggen:

$$\text{Spijt } (\epsilon_n\text{-greedy})(t) = -17.236 + 8.971 \cdot \log(t)$$
$$\text{Spijt (UCB1)}(t) = -66.99 + 15.80 \cdot \log(t)$$
$$\text{Spijt (UCB2)}(t) = -18.487 + 7.107 \cdot \log(t)$$

Hieruit is te concluderen dat van de drie algoritmen het UCB1 algoritme de meest steile logaritmische groei heeft en het UCB2 algoritme de laagst verwachte spijt, hoewel deze niet veel verschilt van de verwachte spijt van het ϵ_n-greedy algoritme.

5.5.2. Algoritme

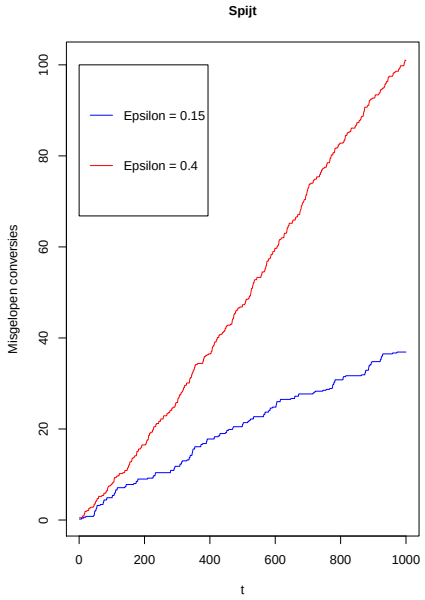
Eigenschappen die een algoritme maken zoals het algoritme is, zijn de variabelen die voorafgaan aan het runnen van het algoritme bepaald worden. Deze variabelen, Figuur 5.7, kunnen een algoritme zodanig beïnvloeden dat deze ongunstige uitkomsten geeft.

De algoritmen hebben de volgende variabelen die vooraf bepaald moeten worden

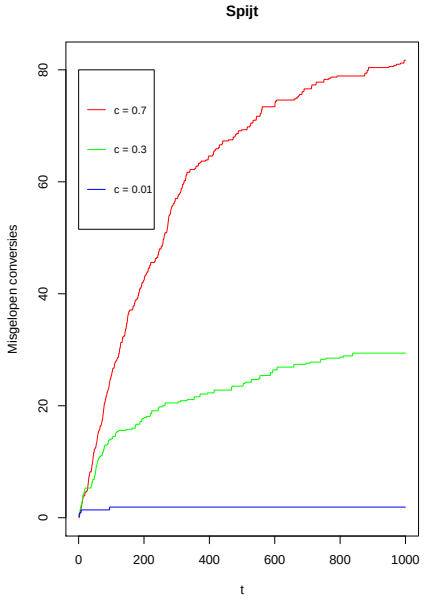
Algoritme	Variabelen	Domein
ϵ-greedy	ϵ	$(0,1)$
ϵ_n-greedy	c	$(0,1)$
	d (per t)	$(0, \min_{i:\hat{k}_i<\hat{k}^*}(\hat{k}^* - \hat{k}_i)]$
UCB1	**geen**	**n.v.t.**
UCB2	α	$(0,1)$
Thompson Sampling	α_i en β_i voor alle i	$(0,1)$ voor alle i

Tabel 5.7: Gekozen variabelen met domein per algoritme

Bij het ϵ-greedy algoritme bepaalt de ϵ alleen hoeveel procent je van de tijd wilt onderzoeken. Deze keuze beïnvloedt het algoritme niet zodanig dat het onjuiste uitkomsten zou geven. De keuze beïnvloedt wel de spijt en het totaal aantal conversies. Dit is het resultaat van een langere of kleinere onderzoeksfase. In Figuur 5.23 is deze invloed goed waar te nemen.



Figuur 5.23: Spijt: verschillende waardes ϵ



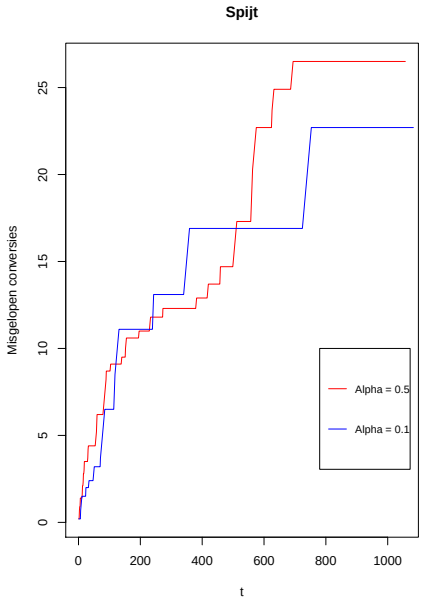
Figuur 5.24: Spijt: verschillende waardes c

De waarde van ϵ is interessanter bij het ϵ_n-greedy algoritme. Hierbij hangt de variabele d af van t, want d verandert met het algoritme mee. De variabele is namelijk gelijk aan $\min_{i:\hat{k}_i<\hat{k}^*}(\hat{k}^* - \hat{k}_i)$, waarbij $\hat{k}$ de

geschatte conversieratio's zijn bij websitebezoeker t. Deze variabele d kan dus niet fout gekozen worden. De variabele c beïnvloedt wel het algoritme, en daarmee ook de variabele d.
In Figuur 5.24 is duidelijk waar te nemen dat het verschil in spijt tussen de verschillende waardes voor c voldoende groot is om stil te staan bij de waardes van c. Het verschil in spijt tussen deze verschillende waardes van c is makkelijk te verklaren door vergelijking (5.3) te bekijken. Immers K en t zijn voor elke c hetzelfde. Wanneer c groter is, is ϵ_t gedurende de test groter en dus ook de kans op onderzoeken, waardoor de spijt weer hoger wordt.

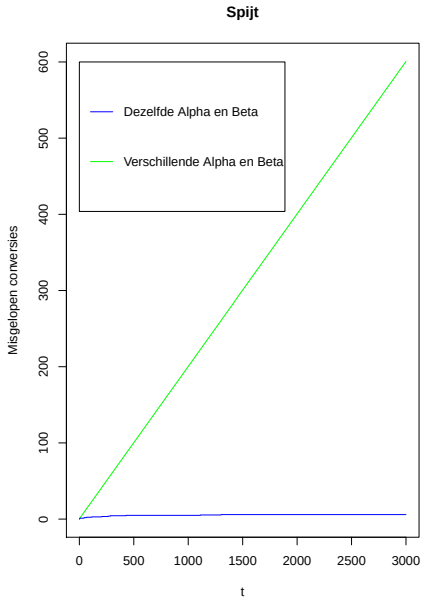
Het Upper Confidence Bound 1 algoritme heeft geen vooraf gekozen variabelen. Dit is een groot voordeel, aangezien er niks fout kan gaan voorafgaand aan de test. Het algoritme is zodanig ontworpen dat het algoritme zichzelf ontwikkelt aan de hand van de tijd. Het algoritme werkte namelijk met bovengrenzen van versies die afhingen van het totaal aantal websitebezoekers en de websitebezoekers van de betreffende versie.

Het Upper Confidence Bound 2 algoritme vraagt in tegenstelling tot het Upper Confidence Bound 1 algoritme wel een variabele die vooraf bepaald moet worden, namelijk $\alpha \in (0,1)$. Figuur 5.25 laat het verschil in spijt zien voor de waardes $\alpha = 0.5$ en $\alpha = 0.1$. Wat opvalt is dat wanneer $\alpha = 0.1$ de spijt lager is voor $t \geq 550$ en dat de tijdperken waarin de betere versie gekozen wordt langer zijn. Vergelijking (5.5) leidt deze opmerking af, want wanneer α kleiner is, wordt de bovengrens b_i kleiner en dus wordt de versie met de hoogste conversieratio vaker gekozen.

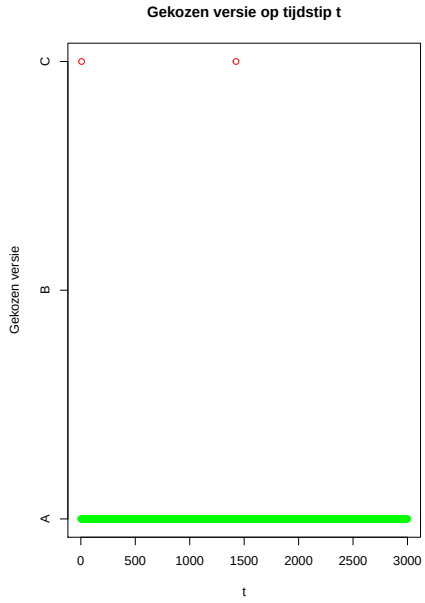


Figuur 5.25: Spijt: verschillende waardes α

Bij Thompson Sampling worden er vooraf drie keer 2 variabelen gekozen, namelijk (α_A, β_A), (α_B, β_B) en (α_C, β_C). Figuur 5.26 bewijst het nadeel van Thompson Sampling. Het figuur weergeeft de spijt van verschillende waardes voor α_i en β_i voor $i \in \{A, B, C\}$ in tegenstelling tot alle $\alpha_i = 1$ en $\beta_i = 5$. De alternatieve α_i's en β_i's zijn $\alpha_i = 1$ voor A, B en C en β_i zijn respectievelijk 4, 9 en 7.



Figuur 5.26: Spijt: verschillende waardes α_i en β_i



Figuur 5.27: Gekozen versie: verschillende waardes α_i en β_i

Figuur 5.26 is een belangrijke aantekening op Thompson Sampling, namelijk het grote verschil in spijt wanneer verkeerde beginwaarden worden gekozen. Het grote verschil is te verklaren door de relatief hoge inschatting over de conversies van versie A. Hierdoor ziet het algoritme al snel versie A als beste versie, hoewel dat de inschatting is van de programmeur. Figuur 5.27 toont de veronderstelling dat het algoritme versie A gelijk als beste versie ziet.

5.5.3. Conclusie

Als we in de voorgaande analyses kijken naar de omvang van de spijt, vallen het Thompson Sampling en het ϵ_n-greedy algoritme het meest op. De spijt van de twee algoritmen namen al snel af, ook wanneer de steekproefomvang vergroot werd naar 2500 websitebezoekers. Echter de twee algoritmen presteren slechter als vooraf de verkeerde waarden aan de variabelen worden toegekend. Figuren 5.24 en 5.26 geven een groter verschil in spijt dan de Figuren 5.23 en 5.25 die de spijt voor verschillende variabelen van het ϵ-greedy en UCB2 algoritme tonen.

Hoewel het verschil in spijt van verschillende ϵ klein is, is de gehele spijt van het ϵ-greedy algoritme relatief hoog. Bovendien is dit het enige algoritme met een lineair verband. Dit wees Figuur 5.21 duidelijk uit.
Beide Upper Confidence Bound algoritmen produceerden een hoge spijt bij 1000 websitebezoekers. Dit is een gevolg van het onderzoeken van de slechtere twee versies. Desondanks daalde de spijt bij 2500 websitebezoekers en toonde het een logaritmische groeicurve.
Hoewel de UCB algoritmen voor de langste periode onderzoeken, heeft het ϵ_n-greedy algoritme wel meer duidelijkheid wat betreft de betrouwbaarheid.
Hoewel Thompson Sampling de laagste spijt geeft, is het algoritme een van de minst betrouwbare algoritmen. Dit komt vooral doordat het algoritme het meest afhangt van de vooraf gekozen variabelen.
Het ϵ-greedy algoritme blijft door het lineaire verband vanaf een zeker punt de hoogste spijt geven. Wanneer veel informatie over de verschillende versies verkregen moet worden, zorgt het ϵ-greedy algoritme hier wel voor. De onderzoeksfase blijft namelijk altijd ϵ% van de tijd.
Een verschillende keuze voor de variabele α bij het Upper Confidence Bound 2 algoritme geeft een klein verschil in spijt. Echter is de verdeling van de conversieratio's van het algoritme nog te onzeker om een conclusie uit te trekken.
Het grootste voordeel van het Upper Confidence Bound 1 algoritme is dat deze geen gekozen variabele als input nodig heeft. De spijt zal dus voor elke simulatie hetzelfde logaritmische verband hebben. Daarentegen heeft het ϵ_n-greedy algoritme een lagere spijt en zijn de conversieratio's van de versies duidelijker af te lezen dan bij het UCB1 algoritme.

Het ϵ_n-greedy algoritme lijkt de meeste voordelen te hebben. De spijt van het algoritme is laag, er is weinig verschil tussen verschillende vooraf gekozen variabelen en de betrouwbaarheid is goed na te gaan.
Dit algoritme zal in de vergelijking met de klassieke methode gebruikt worden als voorbeeld van de bandit-methode.

6

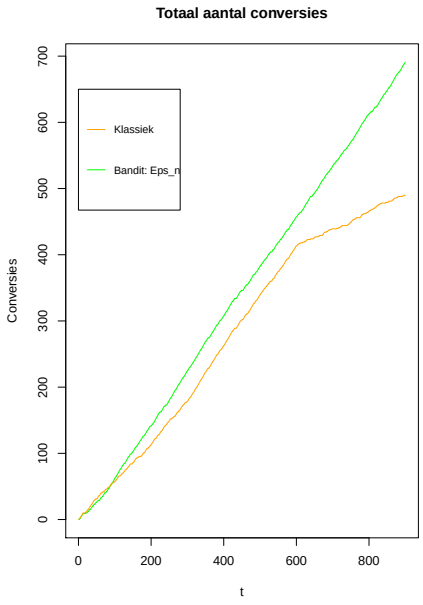
Vergelijking Klassieke en Bandit-methode

In dit hoofdstuk worden de twee methoden naast elkaar beoordeeld. Nadat de twee methoden zijn uitgelegd in Hoofdstukken 4 en 5, worden nu de methoden vergeleken op basis van totaal aantal conversies, spijt, betrouwbaarheid en uitvoering.

In Hoofdstuk 5 werd het ϵ_n-greedy algoritme als meest geschikt algoritme aangewezen. Voor de vergelijking tussen de twee A/B-test methoden zal het ϵ_n-greedy algoritme de leidraad voor de bandit-methode zijn.

6.1. Totaal aantal conversies

Voor een bedrijf is het aantal conversies een van de belangrijkste factoren bij de keuze voor de optimale website, aangezien ze voor hogere opbrengsten zorgen. Het is dan ook noodzakelijk dat de A/B-methoden voor een zo groot mogelijk aantal conversies zorgen. Het totaal aantal conversies van de resultaten is per methode in Figuur 6.1 opgesteld. Hierbij is bij de resultaten van het ϵ_n-greedy naar de eerste 900 websitebezoekers gekeken zodat beide methoden een evengrote steekproefomvang hebben.



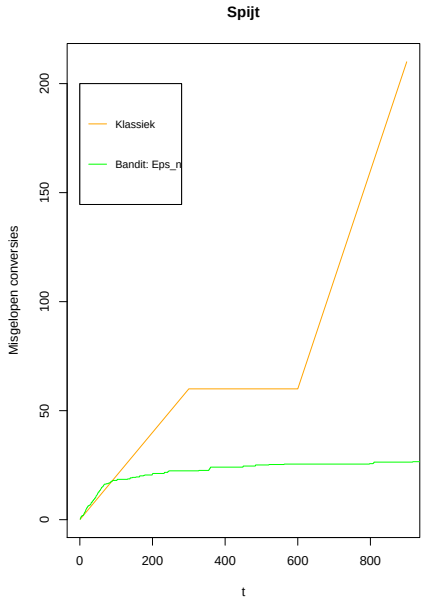
Figuur 6.1: Totaal aantal conversies: Klassiek vs. Bandit

Wanneer de versies A en B getest worden bij de klassieke methode is er nog geen groot verschil tussen de

klassieke en bandit-methode, wanneer echter ook versie C getest wordt, neemt het totaal aantal conversies af. Na 900 websitebezoekers is het verschil tussen de klassieke methode en het ϵ_n-greedy algoritme, de bandit-methode, al 201 conversies.
Ter illustratie: in 2017 had online warenhuis Coolblue 180 miljoen websitebezoekers.[1]
Wanneer de klassieke methode op alle 180 miljoen websitebezoekers was gebruikt in plaats van het ϵ_n-greedy algoritme, had dit tot een verschil van 40 miljoen conversies kunnen leiden.

6.2. Spijt

De spijt hangt samen met het totaal aantal conversies en is ook een belangrijke maatstaf voor de kwaliteit van de methoden. Het is een begrip waar niet vaak rekening mee wordt gehouden, omdat het geen conversies zijn maar eerder misgelopen opbrengsten. Elke keer dat een bezoeker de minst presterende website getoond wordt, loopt het bedrijf opbrengsten mis. De bandit-methode houdt echter wel rekening met het begrip spijt. De methode wil sneller overstappen op de betere versie, omdat er anders conversies misgelopen kunnen worden.
Hoewel de spijt alleen berekend is voor de bandit-methode, is voor een goede vergelijking ook de spijt voor de klassieke methode bepaald. In Figuur 6.2 is de spijt van de klassieke methode tegenover de spijt van het ϵ_n-greedy algoritme gezet.



Figuur 6.2: Spijt: Klassiek vs Bandit

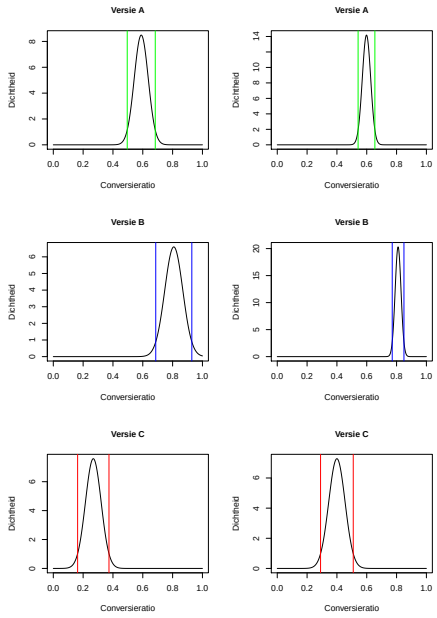
Het grote verschil in spijt komt doordat de klassieke methode vooraf het aantal websitebezoekers per website versie vaststelt en pas achteraf bepaalt dat een van de websites minder presteert. Het ϵ_n-greedy algoritme stuurt bijvoorbeeld totaal 822 van de 900 bezoekers naar website versie B. Het verschil in spijt bij $t = 900$ tussen de twee methoden is een behoorlijk verschil, namelijk $210 - 26.4 = 183.6$ misgelopen conversies.
Bij een steekproefomvang van 2400 is het verschil in spijt al opgelopen tot $560 - 18.3 = 541.7$ misgelopen conversies.
Het voordeel van de bandit-methode ten opzichte van de klassieke methode is dat de bandit-methode de onderzoeksfase van data verzamelen ook als kosten ziet en hierop inspeelt door eerder over te stappen op de betere versie.

6.3. Betrouwbaarheid

Terwijl het totaal aantal conversies en de spijt zich richten op een zo hoog mogelijke opbrengst, is het belangrijk de (statistische) betrouwbaarheid van de methode te toetsen. Anders zou een willekeurige versie tijdens

[1] Jaarverslag 2017 Coolblue

een test met een beperkte steekproefomvang een toevallig hoge opbrengst kunnen geven.
De normale verdelingen van de conversieratio's van de klassieke methode en het ϵ_n-greedy algoritme zijn in Figuur 6.3 samengevoegd. Links is de klassieke methode afgebeeld en rechts de bandit-methode.



Figuur 6.3: Normale verdeling conversieratio's: Klassiek vs Bandit

Beide methoden geven met 95% betrouwbaarheid antwoord op de vraag welke conversieratio het hoogst is. Hieruit is geen conclusie te trekken over welke methode meer statistisch significant is.
In dit onderzoek zijn alle zes methoden getest met een vaste steekproef van 900 of 1000 websitebezoekers. De uitkomsten van deze vergelijkingen geven echter tevens aanwijzingen dat de 95% betrouwbaarheid van de methoden eerder bereikt kan worden met een kleinere steekproefomvang. In de vraagstelling van dit onderzoek is niet opgenomen om aan te tonen bij welke steekproefomvang statistisch betrouwbare uitspraken gedaan kunnen worden. Hiervoor is eerst een grondige analyse van meervoudig hypothese-testen nodig.

6.4. Uitvoering

Met de uitvoering van de methode wordt bedoeld wat nodig is om de methode uit te voeren en hoeveel tijd het kost.
Hier zit namelijk ook een verschil in tussen de twee methoden.
Nadat de klassieke methode is geïmplementeerd en de steekproef is uitgevoerd, wordt er gekeken naar wat de opbrengsten per website zijn. Hieruit wordt dan een significante conclusie getrokken en zal de beste versie online komen te staan.
De bandit-methode is een methode die voor langere tijd online kan blijven staan nadat deze is geïmplementeerd. De bandit-algoritmen passen zich namelijk aan aan de verzamelde data en stappen zelf al sneller over op de betere website versie. Wanneer er geen vaste steekproefomvang is vastgesteld, zal het algoritme dus de websitebezoekers naar de betere versie blijven sturen. Dit onderzoek is echter niet uitvoerig ingegaan op de bandit-algoritmen zonder vaste steekproefomvang. Dit zou de bandit-methode nog beter maken.

7

Conclusie

In dit onderzoek zijn twee verschillende methoden van A/B-testen vergeleken: de klassieke methode en de bandit-methode. Van de bandit-methoden zijn vijf varianten getest. Om de vergelijking te kunnen maken zijn de steekproefomvang en de conversieratio's van drie website versies vooraf bepaald en is een simulatie geprogrammeerd.

Het grote verschil tussen de twee methoden is dat de klassieke methode achteraf besluit wat de beste versie is, waarbij de bandit-methode gedurende de test sneller websitebezoekers stuurt naar de betere versie. Dit zorgt ervoor dat bij de bandit-methode het totaal aantal conversies hoger is en de spijt lager dan bij de klassieke methode. Dit geldt voor alle vijf de geteste varianten van de bandit-methode, waarbij het ϵ_n-greedy algoritme de beste resultaten gaf.
Beide methoden gaven na de steekproef 95% betrouwbare resultaten. Over het verloop van de statistische significantie bij de twee methoden is niet voldoende te concluderen.
Met het grote verschil in het totaal aantal conversies en de spijt en het vermoeden dat statistische significantie eerder bereikt kan worden, maakt de bandit-methode duidelijk een goede verbetering te zijn op het klassieke A/B-testen.

In het volgende hoofdstuk worden suggesties voor verder onderzoek gedaan om deze conclusie te verfijnen.

Discussie en suggesties voor verder onderzoek

De vergelijking tussen de klassieke-methode en de bandit-methode in dit onderzoek is vooral gebaseerd op het totaal aantal conversies en de spijt. Dit is voor de marketingafdeling van een bedrijf een goede strategie, want dit optimaliseert de opbrengst.

Echter de vergelijking mist nog een nadere analyse van de statistische betrouwbaarheid, de meer wiskundige aanpak. De nauwkeurigheid van de betrouwbaarheid, en de daaraan gekoppelde noodzakelijk steekproefomvang, is voor vervolgonderzoek essentieel. Met deze nauwkeurigheid kan er onderzocht worden hoe de bandit-algoritmen nog nauwkeuriger toegepast kunnen worden, zodat ook de steekproefomvang aansluit bij de minimaal gewenste betrouwbaarheid.

Daarnaast zijn voor vervolgonderzoek de verschillende aannames, Hoofdstuk 2, die gemaakt zijn interessant om nader te onderzoeken. Conversies kunnen namelijk wel degelijk verschillende (economische) waarden hebben (een aankoop van 100 € levert een hogere opbrengst dan een aankoop van 20 €) en beïnvloeden de uitkomsten van de methoden. Daarnaast is een nadeel van beide testen dat één persoon meerdere keer een versie gezien kan hebben of zelfs verschillende versies. Dit kan de keuze van de consumenten beïnvloeden en daarmee ook de betrouwbaarheid van de methoden.

De laatste opmerking op de vergelijking is de implementatie van de bandit-algoritmen. Aangezien de algoritmen zichzelf continu aanpassen zijn ze ook moeilijk te controleren. Dit blijkt ook uit andere toepassingen in de praktijk waarbij algoritmen een blackbox zijn en dus interessant is om te onderzoeken. In dit onderzoek was de steekproef simulatie en de implementatie van algoritmen goed te controleren omdat de conversieratio's bekend waren.

Al met al was het onderzoek goed uit te voeren wat betreft de implementatie van de methoden, waardoor er een duidelijke vergelijking gemaakt kon worden tussen de methoden. Echter op de nauwkeurigheid van de statistische significantie is het onderzoek niet dieper ingegaan. Dit is voor vervolgonderzoek interessant omdat het verloop van de steekproefomvang hierdoor aangepast kan worden.

Bibliografie

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47:235–256, 2002.

[2] D. Berry and B. Fristedt. *Bandit problems: sequential allocation of experiments*. London; New York: Chapman and Hall, 1985. ISBN 0412248107 9780412248108.

[3] T. Lai and H. Robbins. Asympotically Efficient Adaptive Allocation Rules. *ADVANCES IN APPLIED MATHEMATICS*, 6:4–22, 1985. URL `https://core.ac.uk/download/pdf/82425825.pdf`.

[4] J. Rice. *Mathematical Statistics and Data Analysis*. Thomson Brooks/Cole, 3 edition, 2007.

[5] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

[6] D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen. A Tutorial on Thompson Sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018. URL `https://web.stanford.edu/~bvr/pubs/TS_Tutorial.pdf`.

[7] Thuiswinkel Waarborg. Nederlandse consumenten besteedden € 23,7 miljard online in 2018. URL `https://www.thuiswinkel.org/nieuws/3992/nederlandse-consumenten-besteedden-23-7-miljard-online-in-2018`.

[8] J. White. *Bandit Algorithms for Website Optimization*. O'Reilly Media, 1 edition, 2012.

R Code

```
library('Rlab')

#Alle methoden samen:

# Definieren ----------------------------------------------------------------

N <- 1000 #Steekproefomvang
convratios <- c(0.6, 0.8, 0.3); #Conversieratio's

O <- matrix(NA,3,N) #Matrix O, dezelfde observaties voor elke test

O[1,] <- rbern(N, convratios[1])
O[2,] <- rbern(N, convratios[2])
O[3,] <- rbern(N, convratios[3])

#Geschatte conversieratio's per versie, per test
k1 <- c(0,0,0)
k2 <- c(0,0,0)
k3 <- c(0,0,0)
k4 <- c(0,0,0)
k5 <- c(0,0,0)
k6 <- c(0,0,0)

#Totaal aantal observaties per versie, per test
n1 <- c(0,0,0)
n2 <- c(0,0,0)
n3 <- c(0,0,0)
n4 <- c(0,0,0)
n5 <- c(0,0,0)
n6 <- c(0,0,0)

#Totaal aantal conversies per versie, per test
counts1 <- c(0,0,0)
counts2 <- c(0,0,0)
counts3 <- c(0,0,0)
counts4 <- c(0,0,0)
counts5 <- c(0,0,0)
counts6 <- c(0,0,0)

#Matrix met versie keuze per website bezoeker t, bijbehorende conversie
#en geschatte conversieratio van gekozen versie, per test
keuze1 <- matrix(NA, 5, N)
keuze2 <- matrix(NA, 5, N)
keuze3 <- matrix(NA, 5, N)
keuze4 <- matrix(NA, 5, N)
keuze5 <- matrix(NA, 5, N)
keuze6 <- matrix(NA, 5, 900) #Klassieke methode: N = 900
```

```
#Spijt per test
tr1 <- 1:N
tr2 <- 1:N
tr3 <- 1:N
tr4 <- 1:N
tr5 <- 1:N
tr6 <- 1:N

#Versie 1, 2 en 3
x <- c(1,2,3)

# Epsilon Greedy ----------------------------------------------------------

#Initialisatie stap: 10 observaties
counts1[1] <- count(O[1,][1:10])
counts1[2] <- count(O[2,][1:10])
counts1[3] <- count(O[3,][1:10])
k1[1] <- count(O[1,][1:10])/10
k1[2] <- count(O[2,][1:10])/10
k1[3] <- count(O[3,][1:10])/10

n1 <- c(10,10,10)

t <- 1
epsilon <- 0.15

while (t <= N) {
  p <- runif(1,0,1)
  if (p < epsilon) {
    i <- sample(x, 1, replace = FALSE, prob = c(1/3,1/3,1/3))
    n1[i] <- n1[i] + 1
    z <- O[i,t]
    keuze1[4,t] <- i
    keuze1[i,t] <- z
    counts1[i] <- counts1[i] + count(z)
    k1[i] <- counts1[i]/n1[i]
    keuze1[5,t] <- k1[i]
    tr1[t] <- 0.8*t - length(keuze1[5,][1:t][keuze1[4,][1:t] == 1])*0.6 -
    length(keuze1[5,][1:t][keuze1[4,][1:t] == 2])*0.8 -
    length(keuze1[5,][1:t][keuze1[4,][1:t] == 3])*0.3
    t <- t+1
  } else {
    i <- which(k1 == max(k1))[1]
    n1[i] <- n1[i] + 1
    z <- O[i,t]
    keuze1[4,t] <- i
    keuze1[i,t] <- z
    counts1[i] <- counts1[i] + count(z)
    k1[i] <- counts1[i]/n1[i]
    keuze1[5,t] <- k1[i]
    tr1[t] <- 0.8*t - length(keuze1[5,][1:t][keuze1[4,][1:t] == 1])*0.6 -
    length(keuze1[5,][1:t][keuze1[4,][1:t] == 2])*0.8 -
    length(keuze1[5,][1:t][keuze1[4,][1:t] == 3])*0.3
    t <- t+1
  }
}
```

```
# Epsilon_N Greedy ------------------------------------------------------------

#Initialisatie stap: 10 observaties
counts2[1] <- count(O[1,][1:10])
counts2[2] <- count(O[2,][1:10])
counts2[3] <- count(O[3,][1:10])
k2[1] <- count(O[1,][1:10])/10
k2[2] <- count(O[2,][1:10])/10
k2[3] <- count(O[3,][1:10])/10

n2 <- c(10,10,10)

t <- 1
c <- 0.3
l <- c(0,0,0) #Verschillen tussen conversieratio's opslaan

dfunc <- function(k) {
  for (i in 1:3) {
    l[i] <- max(k) - k[i]
  }
  l[l==0] <- 1
  d <- min(l)
  return(d)
}

inid <- dfunc(k2)

epsfunc <- function(x,y) {
  epsilon <- min(1, (c*3)/(y^2*x))
  return(epsilon)
}

tr2[1] <- 0

while (t <= N) {
  p <- runif(1,0,1)
  if (p < epsfunc(t, inid)) {
    i <- sample(x, 1, replace = FALSE, prob = c(1/3,1/3,1/3))
    n2[i] <- n2[i] + 1
    z <- O[i,t]
    keuze2[4,t] <- i
    keuze2[i,t] <- z
    counts2[i] <- counts2[i] + count(z)
    k2[i] <- counts2[i]/n2[i]
    keuze2[5,t] <- k2[i]
    tr2[t] <- 0.8*t - length(keuze2[5,][1:t][keuze2[4,][1:t] == 1])*0.6 -
    length(keuze2[5,][1:t][keuze2[4,][1:t] == 2])*0.8 -
    length(keuze2[5,][1:t][keuze2[4,][1:t] == 3])*0.3
    inid <- dfunc(k2)
    t <- t+1
    #print("Explore")
  } else {
    i <- which(k2 == max(k2))[1] #beste ratio
    n2[i] <- n2[i] + 1
    z <- O[i,t]
```

```
    keuze2[4,t] <- i
    keuze2[i,t] <- z
    counts2[i] <- counts2[i] + count(z)
    k2[i] <- counts2[i]/n2[i]
    keuze2[5,t] <- k2[i]
    tr2[t] <- 0.8*t - length(keuze2[5,][1:t][keuze2[4,][1:t] == 1])*0.6 -
    length(keuze2[5,][1:t][keuze2[4,][1:t] == 2])*0.8 -
    length(keuze2[5,][1:t][keuze2[4,][1:t] == 3])*0.3
    inid <- dfunc(k2)
    t <- t+1
    #print("Exploit")
  }
}

# UCB1 ----------------------------------------------------------------

#Initialisatiestap: 1 observatie
z1 <- O[1,1]
z2 <- O[2,1]
z3 <- O[3,1]

counts3[1] <- z1
counts3[2] <- z2
counts3[3] <- z3

k3 <- counts3
bounds3 <- counts3

n3 <- c(1,1,1)

keuze3[1,1] <- z1
keuze3[2,1] <- z2
keuze3[3,1] <- z3
keuze3[4,1] <- which(bounds3 == max(bounds3))[1]
keuze3[5,1] <- max(z1,z2,z3)[1]

t <- 2
tr3[1] <- 0

while (t <= N) {
  i <- which(bounds3 == max(bounds3))[1]
  n3[i] <- n3[i] + 1
  z <- O[i,t]
  keuze3[4,t] <- i
  keuze3[i,t] <- z
  counts3[i] <- counts3[i] + count(z)
  k3[i] <- counts3[i]/n3[i]
  keuze3[5,t] <- k3[i]
  for (i in 1:3) {
    bounds3[i] <- k3[i] + sqrt(2*log(t)/n3[i])
  }
  tr3[t] <- max(convratios)*t - convratios[1]*length(keuze3[5,][1:t][keuze3[4,][1:t]==1]) -
  convratios[2]*length(keuze3[5,][1:t][keuze3[4,][1:t]==2]) -
  convratios[3]*length(keuze3[5,][1:t][keuze3[4,][1:t]==3])
  t <- t + 1
}
```

```
# UCB2 ----------------------------------------------------------------------

#Initialisatiestap: 1 observatie
z1 <- O[1,1]
z2 <- O[2,1]
z3 <- O[3,1]

counts4[1] <- z1
counts4[2] <- z2
counts4[3] <- z3

alphaUCB <- 0.5

tau <- function(x) {
  valtau <- ceiling((1+alphaUCB)^x)
  return(valtau)
}

anr <- function(n,r) {
  vala <- sqrt((1+alphaUCB)*log(exp(1)*n/tau(r))/(2*tau(r)))
  return(vala)
}

k4 <- counts4
bounds4 <- rep(anr(1, tau(0)), 3)

n4 <- c(1,1,1) #aantal keer versie geprobeerd
keuze4 <- matrix(NA, 5, N+100) #Keuze groter want (tau functie + t) kan groter dan N worden

t4 <- 1
tr4[1] <- 0

r <- c(0,0,0) #Aantal tijdvakken per versie

while (t4 <= N) {
  i <- which(bounds4 == max(bounds4))[1]
  taui <- tau(r[i]+1) - tau(r[i])
  if(taui > 0){
    ttaui <- t4 + taui
    z <- O[i,][t4:ttaui]
    keuze4[4,][t4:ttaui] <- rep(i)
    keuze4[i,][t4:ttaui] <- z
    for (j in t4:ttaui) {
      keuze4[5,j] <- (counts4[i] + count(z[t4:j]))/(n4[i]+length(z[t4:j]))
      tr4[j] <- max(convratios)*j - convratios[1]*length(keuze4[4,][1:j][keuze4[4,][1:j]==1]) -
      convratios[2]*length(keuze4[4,][1:j][keuze4[4,][1:j]==2]) -
      convratios[3]*length(keuze4[4,][1:j][keuze4[4,][1:j]==3])
    }
    n4[i] <- n4[i] + taui
    counts4[i] <- counts4[i] + count(z) #nieuw aantal successen
    k4[i] <- counts4[i]/n4[i] #nieuw conversieratio
  }
  r[i] <- r[i] + 1
  t4 <- t4 + taui
  for (i in 1:3) {
```

```
    bounds4[i] <- k4[i] + anr(t4, r[i])
  }
}

# Thompson Sampling ————————————————————————————————————————

#Vectoren voor alpha en beta per versie
alpha <- 1:3
beta <- 1:3

#Begin alpha en beta
alpha[1] <- 1
beta[1] <- 5
alpha[2] <- 1
beta[2] <- 5
alpha[3] <- 1
beta[3] <- 5

t <- 1
tr5[1] <- 0

while (t <= N) {
  for (i in 1:3) {
    k5[i] <- rbeta(1, alpha[i], beta[i])
  }
  vk <- which(k5 == max(k5))
  conv <- O[vk,t]
  keuze5[4,t] <- vk
  keuze5[vk,t] <- conv
  counts5[vk] <- counts5[vk] + conv
  alpha[vk] <- alpha[vk] + conv
  beta[vk] <- beta[vk] + 1 - conv
  keuze5[5,t] <- k5[vk]
  tr5[t] <- max(convratios)*t - convratios[1]*length(keuze5[4,][1:t][keuze5[4,][1:t]==1]) -
  convratios[2]*length(keuze5[4,][1:t][keuze5[4,][1:t]==2]) -
  convratios[3]*length(keuze5[4,][1:t][keuze5[4,][1:t]==3])
  t <- t + 1
}

# Klassiek ————————————————————————————————————————————————

N <- 900

#N opdelen in 3
N1 <- N/3
N11 <- N1+1
N2 <- 2*N/3
N22 <- N2+1

keuze6[1,1:N1] <- O[1,][1:N1]
keuze6[2,N11:N2] <- O[2,][N11:N2]
keuze6[3,N22:N] <- O[3,][N22:N]
keuze6[4,1:N1] <- 1
keuze6[4,N11:N2] <- 2
keuze6[4,N22:N] <- 3
```

```
#Geschatte conversieratio's van versie per website bezoeker
for (n1 in 1:N1) {
  keuze6[5,n1] <- count(keuze6[1,][1:n1])/n1
}
for (n2 in N11:N2) {
  keuze6[5,n2] <- count(keuze6[2,][N11:n2])/(n2-N1)
}
for (n3 in N22:N) {
  keuze6[5,n3] <- count(keuze6[3,][N22:n3])/(n3-N2)
}

#Spijt
for (t in 1:N) {
  tr6[t] <- max(convratios)*t - convratios[1]*length(keuze6[4,][1:t][keuze6[4,][1:t]==1]) -
  convratios[2]*length(keuze6[4,][1:t][keuze6[4,][1:t]==2]) -
  convratios[3]*length(keuze6[4,][1:t][keuze6[4,][1:t]==3])
}

# Vergelijken ------------------------------------------------------------

N = 1000

#Plot: Spijt alle testen samen
par(mfrow = c(1,1))
plot(1:N, tr1, type = 'l', xlab = 't', ylab = 'Misgelopen conversies', main =
'Spijt', col = "black")
lines(1:N, tr2, type = 'l', col = 'green')
lines(1:t4, tr4[1:t4], type = 'l', col = 'yellow')
lines(1:N, tr3, type = 'l', col = 'red')
lines(1:N, tr5, type = 'l', col = 'blue')

legend(-25, 83, legend=c("Eps Greedy", "Eps_n Greedy", "UCB1", "UCB2", "Thompson"),
col=c("black", "green", "red", "yellow", "blue"), lty = 1, cex = 0.8)

#Plot: Totaal aantal conversies alle testen samen
plot(1:N, convt1, type = 'l', xlab = 't', ylab = 'Conversies', main = 'Totaal
aantal conversies')
lines(1:N, convt2, type = 'l', col = 'green')
lines(1:N, convt3, type = 'l', col = 'red')
lines(1:t4, convt4[1:t4], type = 'l', col = 'yellow')
lines(1:N, convt5, type = 'l', col = 'blue')

legend(700, 400, legend=c("Eps Greedy", "Eps_n Greedy", "UCB1", "UCB2",
"Thompson"), col=c("black", "green", "red", "yellow", "blue"), lty = 1, cex = 0.8)


# Plotten ----------------------------------------------------------------

#Voor elke methode zijn de plots hetzelfde, verander alleen de extra 1 naar een
#extra 2,3,4,5 of 6 voor de bijbehorende methode
#bijvoorbeeld keuze1 wordt keuze3 of verd21 wordt verd24

#Plot: keuze per website bezoeker met bijbehorende conversie
par(mfrow = c(2,2), cex.main = 1, cex.lab = 1, cex.axis = 1)
plot(keuze1[1,], col = 'green', ylab = 'Conversie: Ja of Nee', xlab = 't', main =
'Versie A', yaxt = 'none')
```

```
axis(2, seq(0,1,1))
plot(keuze1[2,], col = 'blue', ylab = 'Conversie: Ja of Nee', xlab = 't', main =
'Versie B', yaxt = 'none')
axis(2, seq(0,1,1))
plot(keuze1[3,], col = 'red', ylab = 'Conversie: Ja of Nee', xlab = 't', main =
'Versie C', yaxt = 'none')
axis(2, seq(0,1,1))

cols <- c('green', 'blue', 'red')
cols_k <- cols[keuze1[4,]]
plot(keuze1[4,],col=cols_k, ylab = 'Gekozen versie', xlab = 't', main = 'Gekozen
versie voor persoon t', yaxt = 'none')
axis(2, at = seq(1,3,1), labels = c("A", "B", "C"))

#Plot: Het verloop van de conversieratio's per versie
par(mfrow = c(3,1))
plot(keuze1[5,][keuze1[4,] == 1], type = 'l', xlab = 't', ylab = 'Conversieratio',
main = 'Versie A')
plot(keuze1[5,][keuze1[4,] == 2], type = 'l', xlab = 't', ylab = 'Conversieratio',
main = 'Versie B')
plot(keuze1[5,][keuze1[4,] == 3], type = 'l', xlab = 't', ylab = 'Conversieratio',
main = 'Versie C')

#Verdeling van de conversieratio's per versie
verd11 <- na.omit(keuze1[5,][keuze1[4,] == 1])
verd21 <- keuze1[5,][keuze1[4,] == 2]
verd31 <- keuze1[5,][keuze1[4,] == 3]

#Bijbehorende plot van de verdelingen
y <- seq(0, 1, 0.001)
y11 <- dnorm(y, mean(verd11), sd(verd11))
y21 <- dnorm(y, mean(verd21), sd(verd21))
y31 <- dnorm(y, mean(verd31), sd(verd31))
par(mfrow = c(3,1))
plot(y, y11, type = 'l', main = 'Versie A', xlab = 'Conversieratio', ylab =
'Dichtheid')
abline(v = mean(verd11) + 2*sd(verd11), col = 'green')
abline(v = mean(verd11) - 2*sd(verd11), col = 'green')
plot(y, y21, type = 'l', main = 'Versie B', xlab = 'Conversieratio', ylab =
'Dichtheid')
abline(v = mean(verd21) + 2*sd(verd21), col = 'blue')
abline(v = mean(verd21) - 2*sd(verd21), col = 'blue')
plot(y, y31, type = 'l', main = 'Versie C', xlab = 'Conversieratio', ylab =
'Dichtheid')
abline(v = mean(verd31) + 2*sd(verd31), col = 'red')
abline(v = mean(verd31) - 2*sd(verd31), col = 'red')

#Plot: Spijt
par(mfrow = c(1,1))
plot(1:N, tr1, type = 'l', xlab = 't', ylab = 'Misgelopen conversies', main = 'Spijt')

#Totaal aantal conversies
convt1 <- 1:N
for (conv in 1:N) {
  convt1[conv] <- sum(keuze1[1,][1:conv], keuze1[2,][1:conv], keuze1[3,][1:conv],
  na.rm = TRUE)
```

```
}

#Plot: totaal aantal conversies
par(mfrow = c(1,1))
plot(convt1, type = 'l', main = 'Totaal aantal conversies', xlab = 't', ylab =
'Conversies')

#Voor de vergelijkingen van variabelen binnen de methoden zijn bovenstaande secties
#van de algoritmen meerdere keren gerund voor verschillende variabele en geplot.

#Groei van de Spijt
xt <- 1:N
lm2 <- lm(tr2 ~ log(xt))
lm3 <- lm(tr3 ~ log(xt))
lm4 <- lm(tr4[1:t4] ~ log(xt[1:t4]))

par(mfrow = c(3,1))
plot(xt, tr2, type = 'l', main = 'Verwachte spijt: Epsilon_n',
ylab = 'Misgelopen conversies', xlab = 't')
lines(xt, lm2$coefficients[1]+lm2$coefficients[2]*log(xt), col =
'green')

plot(xt, tr3, type = 'l', main = 'Verwachte spijt: UCB1', ylab =
'Misgelopen conversies', xlab = 't')
lines(xt, lm3$coefficients[1]+lm3$coefficients[2]*log(xt), col =
'red')

plot(xt[1:t4], tr4[1:t4], type = 'l', main = 'Verwachte spijt:
UCB2', ylab = 'Misgelopen conversies', xlab = 't')
lines(xt[1:t4], lm4$coefficients[1]+lm4$coefficients[2]*log(xt[1
:t4]), col = 'yellow')
```