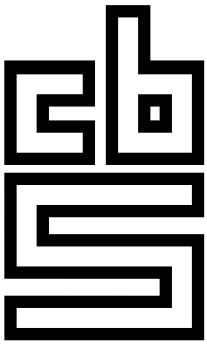
Automatische foutlocalisatie met harde en zachte restricties op het CBS

Sevinç Gökşen

Stageverslag[1]

[1]Kennisgeving: De in dit rapport weergegeven opvattingen zijn die van de auteur en komen niet noodzakelijk overeen met het beleid van het Centraal Bureau voor de Statistiek.



Automatische foutlocalisatie met harde en zachte restricties op het CBS

Stageverslag

Auteur:
Sevinç Gökşen

Begeleiders CBS:
Drs. Sander Scholtus
Dr. Jeroen Pannekoek

Begeleiders VU:
Dr. Marianne Jonker
Dr. René Bekker

Vrije Universiteit Amsterdam
Faculteit der Exacte Wetenschappen
Studierichting Business Mathematics and Informatics
De Boelelaan 1081a
1081 HV Amsterdam

Stagebedrijf:
Centraal Bureau voor de Statistiek
Henri Faasdreef 312
2492 JP Den Haag

22 februari 2012

Samenvatting

Eén van de problemen bij statistisch onderzoek is dat de data van de respondenten nog fouten kunnen bevatten. Om de data geschikt te maken voor analyse-doeleinden, vindt daarom eerst een controle- en correctieproces plaats, waarbij foute en ontbrekende waarden worden gedetecteerd en vervangen door imputaties (geschatte waarden uit een model). Traditioneel voerden inhoudelijke experts dit werk met de hand uit. Tegenwoordig wordt het controle- en correctieproces steeds meer geautomatiseerd. Hierbij spelen restricties waaraan de data moeten voldoen een belangrijke rol. Een dataset bevat records en elk record stelt een ingevulde enquête voor. Een record dat niet voldoet aan de restricties bevat blijkbaar fouten. De eerste stap van het controle- en correctieproces bestaat uit het zogenaamde foutlocalisatieprobleem, d.w.z. het aanwijzen van de foute waarden die de schendingen van de restricties veroorzaken. Het is gebruikelijk om hierbij aan te nemen dat de data voldoen aan het principe van 'Fellegi en Holt': de foute waarden vormen de kleinste verzameling waarden die kan worden aangewezen zodat het mogelijk is om nieuwe waarden te imputeren die voldoen aan alle restricties. Gegeven dit principe is het foutlocalisatieprobleem te formuleren als een minimalisatieprobleem met binaire doelvariabelen.

Een belangrijke beperking van het bestaande algoritme voor het oplossen van het foutlocalisatieprobleem, is dat ze ervan uitgaan dat alle restricties hard zijn, in de zin dat elke schending veroorzaakt wordt door een fout. Echter hoeft dat niet het geval te zijn, omdat ook zachte restricties worden gebruikt, die soms geschonden worden door correcte waarden. Tijdens dit onderzoek is gekeken naar een nieuwe aanpak, die wel onderscheid maakt tussen harde en zachte edits. Om rekening te houden met de schending van de zachte edits geven we elke zachte edit een schendingsgewicht mee. Voor het localiseren van fouten onderscheiden we twee situaties: in de eerste variant krijgt elke zachte restrictie een statisch schendingsgewicht en in de tweede variant laten we elk schendingsgewicht afhangen van de grootte van de bijbehorende schending.

Voor de eerste variant hebben we ruim 25 verschillende methoden bedacht en uitgeprobeerd op twee datasets uit het jaar 2007, zodat we de verkregen resultaten kunnen vergelijken met elkaar. In de meeste methoden zijn

referentiedatasets gebruikt om de schendingsgewichten te bepalen. De referentiedataset is de gaaf gemaakte dataset door gaafmakers van het jaar ervoor. De formulering van de edits en aanpassingen van de doelfunctie zijn enkele methoden die toegepast zijn. De formulering van de edits hebben goede resultaten gegeven en zijn dus toepasbaar voor automatische foutlocalisatie.

Voor de tweede variant hebben we ruim 5 verschillende methoden bedacht en uitgevoerd, waarbij onder andere de schendingsgrootte een rol speelt in het bepalen van een passend schendingsgewicht voor de zachte edits. De resultaten van deze tweede variant zijn echter niet beter in vergelijking met de statische schendingsgewichten. Dus wordt de eerste variant verkozen boven de tweede variant, omdat deze eenvoudiger is en tot betere resultaten leidt. Tevens localiseren we door het meenemen van de zachte edits meer fouten in de ruwe dataset.

Voorwoord

De master Business Mathematics and Informatics, met de nieuw naam Business Analytics aan de Vrije Universiteit Amsterdam wordt afgesloten met een afstudeerstage. Het doel van deze stage is om een product op te leveren dat van waarde is voor de organisatie, de afstudeerscriptie. Ik heb onderzoek gedaan op het Centraal Bureau voor de Statistiek naar nieuwe mogelijkheden tot het automatisch opsporen van fouten in datasets die van groot belang zijn voor de schattingen van de publicatiecijfers. Het onderzoek heb ik uitgevoerd binnen de divisie Methodologie en Kwaliteit, die per 1 januari 2012 de Hoofddirectie P.I.M. (Procesontwikkeling, IT en Methodologie) heet.

In eerste instantie wil ik graag het CBS bedanken voor de mogelijkheid die ze mij hebben geboden om mijn onderzoek uit te voeren. Daarnaast wil ik graag mijn begeleiders vanuit het CBS Jeroen Pannekoek en in het bijzonder Sander Scholtus bedanken voor de begeleiding, de inzichten in de foutlocalisatiemethoden en adviezen gedurende dit onderzoek. Graag wil ik mijn begeleider vanuit de VU, Marianne Jonker bedanken voor alle feedback tijdens het uitvoeren van het onderzoek. Ook wil ik graag Rene Bekker bedanken voor het herzien van mijn afstudeerscriptie. Tenslotte wil ik alle collega's van de divisie Methodologie en Kwaliteit bedanken voor de gezellige periode die ik heb gehad tijdens mijn afstudeerstage en de interessante gesprekken in de pauzes.

Sevinç Gökşen
Februari 2012

Inhoudsopgave

Hoofdstuk 1

Inleiding

1.1 Het Centraal Bureau voor de Statistiek

De oprichting van het Centraal Bureau voor de Statistiek vond plaats op 9 januari 1899. Het publiceren van betrouwbare en samenhangende statistische informatie en de productie van cijfers over allerlei uiteenlopende aspecten van de samenleving is de taak van het bureau. Het CBS heeft toegang tot bestaande administratieve bestanden van de overheid en buiten de overheid. Dit is de secundaire gegevensverzameling. Het CBS houdt enquêtes onder bedrijven en burgers ter aanvulling op de bestaande registers en deze vorm van dataverzameling heet de primaire gegevensverzameling. De ingevulde enquêtes kunnen per post, per email, via een persoonlijk of telefonisch interview worden verkregen. Het CBS is verplicht tot geheimhouding van de individuele gegevens. Zowel de data uit deze eigen waarneming als de data uit de bestaande databestanden bevatten meestal fouten. Deze fouten kunnen zowel ontstaan tijdens de waarneming als tijdens het proces van verwerking. Indien deze fouten leiden tot vertekening in schattingen van publicatiecijfers is het voor het CBS van groot belang om deze fouten op te sporen en te verbeteren [1]. De divisie Methodologie en Kwaliteit is binnen het CBS de afdeling waar onderzoek wordt gedaan naar nieuwe statistische methoden en naar de toepasbaarheid van reeds ontwikkelde methoden in het statistische proces.

1.2 Het probleem

Een van de problemen bij statistisch onderzoek is dat de data van de respondenten nog fouten kunnen bevatten. Dit kan komen doordat respondenten de vraag niet goed hebben gelezen of niet hebben begrepen. Om de data geschikt te maken voor analysedoeleinden, vindt daarom eerst een

controle- en correctieproces plaats, waarbij foute en ontbrekende waarden worden gedetecteerd en vervangen door imputaties (geschatte waarden uit een model). Het foutlocalisatieprobleem is het bepalen van de foute waarden en het imputatieprobleem is het bepalen van de juiste waarden voor de gelocaliseerde foute waarden [2]. Het opsporen en corrigeren van fouten wordt **gaafmaken** genoemd. Traditioneel voerden inhoudelijke experts dit werk met de hand uit. Omdat dit handmatige controle- en correctieproces tijdrovend en relatief duur is, wordt het controle- en correctieproces tegenwoordig steeds meer geautomatiseerd. Gaafmaken is van groot belang voor de publicatie van betrouwbare waarden en het verbeteren van de enquêtes. Bij de foutlocalisatie spelen de controleregels waaraan de data moeten voldoen een belangrijke rol. Deze controleregels heten ook wel restricties of edits en worden opgesteld door de experts. Restricties geven aan waaraan de variabelen in de data moeten voldoen. Voorbeelden van zulke restricties zijn: *winst=omzet−kosten* en *omzet* ≥ 0 bij bedrijfseconomische datasets of ALS *geslacht= 'man'* DAN *zwanger ='nee'* bij demografische datasets. Bij numerieke gegevens heeft de lineaire restrictie een algemene vorm [1]:

$$\sum_{j=1}^{n} a_{kj} x_j \geq b_k \quad \text{of} \quad \sum_{j=1}^{n} a_{kj} x_j = b_k. \tag{1.1}$$

Hierbij staat de a_{kj} voor de lineaire coëfficiënten van de k^{de} edit en de b_k staat voor de constante van de k^{de} edit. De x_j $(j = 1, \ldots, n)$ zijn de variabelen die voorkomen in de data.

De restricties worden opgedeeld in twee deelverzamelingen, de harde en de zachte restricties. De harde restricties, ook wel de harde edits genoemd, mogen niet geschonden blijven. De zachte edits mogen soms geschonden blijven. Een voorbeeld van een zachte edit is: *winst≤0,6 × omzet*. Deze restrictie wordt gehanteerd, omdat bedrijven waarvan de winst meer dan 60% van de omzet bedraagt zeldzame bedrijven zijn. Een dergelijke combinatie van waarden moet daarom kritisch worden bekeken, maar is niet per se fout. Elke dataset bevat records met waarden. Een record bij productiestatistieken (PS) staat voor één bedrijf. De productiestatistieken geven een beschrijving van de werkgelegenheid in en het financiële reilen en zeilen van een bedrijfstak [1]. De PS worden samengesteld voor de volgende bedrijfstakken: landbouw, groothandel en detailhandel, horeca, vervoer, zakelijke en persoonlijke dienstverlening. Een record dat niet voldoet aan de restricties bevat blijkbaar fouten. De eerste stap van het controle- en correctieproces bestaat uit het zogenaamde foutlocalisatieprobleem, het aanwijzen van de foute waarden die de schendingen van de restricties veroorzaken. De foutlocalisatie gebeurt volgens **"het principe van Fellegi en Holt"**, dat luidt:

[1] *CBS.nl, onderdeel Thema's, handel en ondernemen.*

wijs zo min mogelijk foute waarden aan, maar wel zodanig dat het veranderen van deze waarden leidt tot een record dat voldoet aan alle harde edits [1]. Een belangrijke beperking van het bovengenoemde principe van Fellegi en Holt voor het oplossen van het foutlocalisatieprobleem, is dat het ervan uitgaat dat alle restricties "hard" zijn, in de zin dat elke schending veroorzaakt wordt door een fout. In het traditionele, handmatige controle- en correctieproces worden echter ook "zachte" restricties gebruikt, die soms geschonden worden door correcte waarden. Daarom is een verbeterd foutlocalisatiealgoritme interessant voor het CBS. Het zou ertoe kunnen leiden dat meer data automatisch worden gecorrigeerd. Een mogelijke verbetering is het uitbreiden van het bestaande algoritme zodat het ook zachte restricties kan verwerken.

1.3 Het doel

We beschrijven eerst het huidige foutlocalisatieprobleem. Om de meest geschikte oplossing te bepalen die voldoet aan alle harde edits krijgen de variabelen (x_j) betrouwbaarheidsgewichten (w_j) mee die net zoals de edits bepaald zijn door de experts. De variabelen y_j ($j = 1, \ldots, n$) definiëren we als volgt:

$$y_j = \begin{cases} 1 \text{ als variabele } x_j \text{wordt aangepast} \\ 0 \text{ anders.} \end{cases} \tag{1.2}$$

Het is de bedoeling dat er zo min mogelijk variabelen worden aangepast onder de voorwaarde dat er kan worden voldaan aan alle harde edits [1]:

$$\left\{ \sum_{j=1}^{n} w_j y_j \right\}. \tag{1.3}$$

De formule (1.3) wordt geminimaliseerd en als er geen variabelen moeten worden aangepast, is de som van de formule (1.3) gelijk aan nul.

Het zoeken naar een mogelijke verbetering van het bovenstaande foutlocalisatieprobleem is het doel van dit onderzoek. Het idee is om te zoeken naar verzamelingen van variabelen die kunnen worden aangepast, zodat voldaan wordt aan alle harde edits. Zodra er wordt voldaan aan alle harde edits, hebben we een toegelaten oplossing voor het foutlocalisatieprobleem. Vervolgens bepalen we voor elk van deze toegelaten oplossingen welke zachte edits geschonden zijn. Deze informatie gebruiken we om een keuze te maken uit de toegelaten oplossingen die voldoen aan de harde edits. Hierbij is het idee van het CBS om elke zachte restrictie een "schendingsgewicht" (s_k) te

geven met $k = 1, \ldots, K$. Om rekening te houden met de schendingen van de zachte edits voegen we een term D toe aan (1.3):

$$D_{zacht} = \sum_{k=1}^{K} s_k z_k \tag{1.4}$$

met

$$z_k = \begin{cases} 1 \text{ als zachte edit } k\text{wordt geschonden} \\ 0 \text{ anders.} \end{cases} \tag{1.5}$$

Structureel komt dit proces op het volgende neer:

- We bepalen eerst alle toegelaten oplossingen die voldoen aan alle harde edits.
- Voor elk van deze toegelaten oplossingen bepalen we de minimale D_{zacht}. Dit is dus een kleine minimalisatie binnen het grote minimalisatie probleem.
- Met behulp van de formule hieronder (1.6) bepalen we de totale kosten van een oplossing en we minimaliseren over alle toegelaten oplossingen die voldoen aan alle harde edits.

$$\left\{ \lambda \sum_{j=1}^{n} w_j y_j + (1-\lambda) \sum_{k=1}^{K} s_k z_k \right\}. \tag{1.6}$$

De λ kiezen we in bovenstaande formule in eerste instantie gelijk aan 0,5. Afhankelijk van de gekozen parameters in formule (1.6) (de λ, de betrouwbaarheidsgewichten en de schendingsgewichten) wordt een afweging gemaakt tussen het aanpassen van zo min mogelijk variabelen en het voldoen aan zo veel mogelijk zachte edits. In paragraaf 2.4 gaan we D_{zacht} en de oplossing van dit probleem verder toelichten.

In de eerste, meest eenvoudige variant van het probleem krijgt elke zachte restrictie een **statisch schendingsgewicht** s_k. Statische schendingsgewichten zijn voor alle records in een dataset gelijk en worden éénmalig aan het begin van de foutlocalisatie bepaald. Voor deze variant moet onderzocht worden wat een goede manier is om deze statische gewichten automatisch te kiezen, gegeven een bepaalde dataset.

De inhoudelijke experts besteden bij de handmatige foutlocalisatie veel aandacht aan de grootte van de schendingen (een kleine schending is eenvoudiger te negeren dan een grote schending) en de statische schendingsgewichten

kunnen geen rekening houden met de grootte van de schendingen, waardoor deze eerste variant niet zo aantrekkelijk is. Een tweede variant is daarom dat er onderzocht moet worden wat een goede manier is om deze gewichten automatisch te bepalen, gegeven de schendingen, waarbij elk schendingsgewicht afhangt van de grootte van de bijbehorende schending en van de mate waarin een edit belangrijk is, deze schendingsgewichten noemen we de **dynamische schendingsgewichten**. De dynamische schendingsgewichten zijn voor alle records in een dataset verschillend en kunnen gedurende de foutlocalisatie veranderen. Overigens zou het kunnen dat de eerste variant in de praktijk toch wordt verkozen boven de tweede variant: de eerste variant is namelijk een stuk eenvoudiger, en de tweede variant is daarom alleen nuttig als hij leidt tot duidelijk betere resultaten. Of dat laatste inderdaad het geval is, is een centraal thema van dit onderzoek.

De kwaliteit van de gevonden oplossingen hangt af van de mate waarin de resulterende data verschillen van de bijbehorende handmatig behandelde data. Het idee is, om dezelfde dataset zowel handmatig als automatisch gaaf te maken. De aanname die we maken is als volgt: hoe meer de automatisch gaafgemaakte dataset lijkt op de bijbehorende handmatig gaafgemaakte dataset, des te beter de kwaliteit van het automatisch gaafmaken is. Hierbij kan worden gekeken naar de foutlocalisatie zelf (aantal ten onrechte wel/niet als fout aangewezen variabelen), maar ook naar de statistische eigenschappen van de behandelde data, of er bijvoorbeeld sprake is van een systematische vertekening.

1.4 De structuur

Dit verslag geeft het onderzoek weer naar automatisch gaafmaken van productiestatistieken, waarbij een databestand van de groothandelbranche als testbestand wordt gebruikt.

In het volgende hoofdstuk wordt het probleem geanalyseerd en uitgelegd. In de eerste paragraaf wordt uitgelegd hoe de data worden verzameld en in de tweede paragraaf hoe de data verwerkt worden. Tijdens deze verwerkingsfase worden de soorten fouten en de foutlocalisatie nader toegelicht. In de laatste paragraaf gaan we het hebben over de mogelijkheden tot het meten van de kwaliteit van de foutlocalisatie. Dus we bespreken twee manieren die als maatstaf kunnen dienen bij de beoordeling van de gevonden resultaten.

In hoofdstuk drie bespreken we de verschillende modellen van de eerste variant die gebruikt zijn tijdens het verbeterproces van automatisch gaafmaken met zachte edits. In de eerste paragraaf worden verschillende methoden met een statisch schendingsgewicht behandeld en in de tweede paragraaf creëren we nieuwe edits, namelijk de kwantieledits. In de derde paragraaf wordt een nieuwe methode toegelicht waarbij er in plaats van de somfunctie (1.4) de

maximale schending wordt toegepast. In de vierde paragraaf wordt de nearest neighbormethode behandeld waarbij we bij het ruwe record het meest overeenkomende referentierecord zoeken. Voor de selectie van het referentierecord, wordt de referentiedataset gebruikt. De referentiedataset is de gaaf gemaakte dataset door gaafmakers van het jaar ervoor. In de vijfde paragraaf van hoofdstuk drie behandelen we de chi-kwadraatmethode.

In hoofdstuk vier worden de resultaten van de benoemde methoden in hoofdstuk drie weergegeven.

In hoofdstuk vijf behandelen we een drietal geavanceerdere methoden, waaronder de tweede variant. Bij deze tweede variant wordt er met verschillende modellen weergegeven hoe dynamische schendingsgewichten automatisch bepaald kunnen worden.

Hoofdstuk zes geeft de resultaten weer van de geavanceerdere methoden.

In hoofdstuk zeven betrekken we de betrouwbaarheidsgewichten bij de gevonden resultaten uit hoofdstuk drie en in hoofdstuk acht geven we de conclusie van het gehele onderzoek. In het laatste hoofdstuk worden de aanbevelingen beschreven.

Hoofdstuk 2

Analyse van het probleem

In dit hoofdstuk wordt het probleem uitgespit. Dit doen we in de eerste paragraaf door in te gaan op de verzameling van data. In de volgende paragraaf gaan we in op de verwerking van data en in de laatste paragraaf worden twee manieren weergeven die gebruikt kunnen worden bij het beoordelen van de resultaten. In hoofdstuk drie zullen we de verwerkingfase benutten voor het localiseren van fouten door middel van verschillende methoden.

2.1 Dataverzameling

De data die binnenkomen op het CBS kunnen worden verkregen via mondeling enquêteren bij de respondent thuis, telefonisch enquêteren of schriftelijk enquêteren waarbij de respondent de vragenlijst invult en retourneert. De vragenlijsten worden ontworpen door het CBS en met behulp van een steekproef kan worden vastgesteld naar wie ze worden toegezonden. Het mondeling enquêteren is een dure vorm om aan gegevens te komen. Er is namelijk een enquêteur die langs gaat bij de respondent en de enquêteurs moeten allen opgeleid en betaald worden. Daarentegen levert het in vergelijking met de andere vormen van enquêteren een betere respons op en is de verkregen informatie van hoge kwaliteit. Telefonisch enquêteren betekent dat de respondenten via de telefoon worden benaderd. Hierbij is het van groot belang dat het gesprek kort duurt en alle vragen die gesteld worden duidelijk zijn. Het voordeel is dat er niet gereisd hoeft te worden naar de respondent, echter een nadeel is dat niet alle telefoonnummers te achterhalen zijn. Bij schriftelijk enquêteren krijgen de geselecteerde respondenten een vragenlijst opgestuurd en die moet dan ingevuld geretourneerd worden. Hierbij worden er geen enquêteurs ingezet en dit betekent dat er geen rechtstreekse controle is bij het invullen van de enquête. Hierdoor kan de respondent de vraag anders interpreteren en daardoor verkeerde gegevens invullen. Tevens wekt het schriftelijk enquêteren vaak de indruk, dat het vrijblijvend is en

dit zorgt voor hoge non-respons [4]. Het is echter niet altijd vrijblijvend, omdat er vaak een selectie van bedrijven of personen wordt gemaakt met verplichte deelname aan de enquête. Tegenwoordig wordt steeds meer gebruik gemaakt van electronische vragenlijsten (via het internet). Dit lijkt heel veel op schriftelijke enquêteren. Echter heeft deze vorm ook zijn voor- en nadelen. Een voordeel is dat het goedkoop is, maar nog belangrijker is dat er automatische controles ingebouwd kunnen worden. Een nadeel is, net zoals bij schriftelijk enquêteren, dat men denkt dat het vrijblijvend is, terwijl dat niet het geval blijkt te zijn. Een ander nadeel is dat we hiermee ons beperken tot de internetgebruikers en bijvoorbeeld ouderen, die minder actief zijn op het internet, niet goed meenemen in de steekproef. Naast het enquêteren kan het CBS ook gebruik maken van door andere instanties verzamelde data, bijvoorbeeld het CWI of de belastingdienst.

Er zijn drie verschillende vormen van data: numerieke data, categoriale data en gemengde data. Numerieke data is een databestand dat bestaat uit alleen cijfers en er kunnen rekenkundige bewerkingen worden uitgevoerd met de numerieke variabelen, terwijl categoriale data een databestand is met zowel cijfers als letters, zonder de rekenkundige bewerkingen. Bijvoorbeeld de provincie Noord-Holland wordt gecodeerd als een '1', de provincie Zuid-Holland als een '2', etc. Een voorbeeld van numerieke data is de financiële administratie van een bedrijf. Een voorbeeld van categoriale data is de burgerlijke stand van een gemeente. Daarnaast bestaat er meestal een combinatie van deze twee vormen, gemengde data. Een voorbeeld hiervan zijn de inkomsten in combinatie met de burgerlijke stand.

2.2 Dataverwerking

Als een groot deel van de respons binnen is begint de verwerkingsfase en in deze paragraaf behandelen we deze fase. Een heel belangrijk onderdeel van de verwerkingsfase is dat de verzamelde data worden gecontroleerd op foute waarden en ontbrekende waarden. Dit is noodzakelijk om betrouwbare gegevens te kunnen publiceren. Bij dataverzameling kan het voorkomen dat de respondent de vraag niet goed begrijpt of juist overslaat, terwijl een antwoord noodzakelijk is. Als deze fouten niet worden gecontroleerd en in de data blijven, zullen er bij de analyse mogelijk verkeerde conclusies worden afgeleid. Om dit te voorkomen worden de data bij de verwerking gecontroleerd op mogelijke foute waarden en ontbrekende waarden. De gevonden foute waarden en ontbrekende waarden worden vervangen door imputaties. De imputaties zijn geschatte waarden uit een model. Het localiseren van de fouten en vervolgens imputeren van de juiste waarden wordt gaafmaken genoemd. In dit verslag wordt de nadruk gelegd op diverse methoden om data automatisch gaaf te maken.

Het gaafmaken houdt dus in dat fouten in een verzameling gegevens worden opgespoord en gecorrigeerd. Het gaafmaken gebeurt aan de hand van een aantal controleregels, ook wel edits genoemd, waaraan de data moeten voldoen. Deze edits worden opgesteld door experts die het gaafmaken handmatig uitvoeren. Deze manier van gaafmaken wordt **interactief gaafmaken** genoemd. Een record staat voor de waarden van één bedrijf. De experts gaan per record na of de data voldoen aan de edits. De experts doen dit met behulp van een computerprogramma dat controles uitvoert en toont welke variabelen er zijn betrokken bij de geschonden edit(s). Het programma dat ze hiervoor gebruiken heet Blaise en is ontwikkeld op het CBS. De experts worden ook wel gaafmakers genoemd. De gaafmaker besluit welke variabelen moeten worden gecorrigeerd en welke waarden deze variabelen moeten krijgen. De gaafmakers stellen vaak eerst een gaafmaakinstructie op om de kwaliteit te waarborgen. In deze instructies worden o.a. de controleregels, edits vastgelegd.

Omdat het handmatige proces tijdrovend is en een erg dure manier betreft, zijn er alternatieve manieren van gaafmaken ontwikkeld, **selectief gaafmaken** en **automatisch gaafmaken**. Automatisch gaafmaken betekent dat de records in een dataset automatisch worden gaafgemaakt door een computerprogramma. Voor het automatisch gaafmaken worden de edits gebruikt die zijn verkregen van de gaafmakers. Selectief gaafmaken is dat de data worden gesplitst in records waar waarschijnlijk invloedrijke fouten in voorkomen, die handmatig moeten worden gaafgemaakt en records zonder invloedrijke fouten die automatisch gaafgemaakt kunnen worden [1].

2.3 Soorten fouten

Een dataset kan verschillende soorten fouten bevatten. Om te beginnen waardefouten, relatiefouten en routefouten. Een voorbeeld van een waardefout is dat een ondervraagde als antwoord bij zijn leeftijd 199 jaar invult. Een voorbeeld van een relatiefout is dat een ondervraagde bij zijn leeftijd 19 jaar antwoordt en vermeldt dat hij gepensioneerd is. Dus de relatie leeftijd en gepensioneerd speelt hier een belangrijke rol. Routefouten zijn vragen die beantwoord hadden moeten worden tijdens het invullen van de enquête, maar die zijn overgeslagen door de ondervraagde. Naast deze fouten bestaan er ook **systematische fouten, random fouten, invloedrijke fouten en uitbijters**. Een systematische fout houdt in dat een vraag op dezelfde manier verkeerd is beantwoord door meerdere respondenten. Dit kan er dus op wijzen dat een vraagstelling niet duidelijk genoeg is geweest en dit kan een aanleiding zijn tot een mogelijk herontwerp van een vragenlijst. Bij systematische fouten worden dezelfde edits vaak op dezelfde manier geschonden. Dit soort fouten kunnen op een deductieve manier worden opgelost [1].

Voorbeelden van deductief op te lossen fouten zijn:

- Duizendfout
- Minteken
- Tekenfout
- Doortelfout
- Eenvoudige schrijffout

Eén van de meest voorkomende fouten die deductief kan worden opgelost is de duizendfout. De duizendfout treedt op als een bedrijf bijvoorbeeld in plaats van duizenden euro's het antwoord in euro's opgeeft, dus met een factor duizend. Dit soort fouten worden gedetecteerd door referentiewaarden te gebruiken. Referentiewaarden zijn waarden van dezelfde ofwel soortgelijke respondenten van voorgaande jaren. Indien er geen data van de voorgaande jaren beschikbaar zijn, wordt er gekeken naar de mediaan van een desbetreffende variabele over de huidige periode in het stratum van de respondent.

Een andere systematische fout die deductief kan worden opgelost is het "onterecht geplaatste minteken". Meestal staat er in de vragenlijst al een minteken genoteerd indien bijvoorbeeld de kosten moeten worden afgetrokken van de omzet. Respondenten noteren soms nog een minteken en daardoor krijg je in plaats van een positief getal een negatief getal. Deze fout wordt opgelost door de absolute waarde te nemen van bijvoorbeeld de ingevulde kosten.

De tekenfout houdt in dat in vragenlijsten die meestal worden gebruikt in de PS een rekening is opgebouwd uit saldi die vervolgens moeten worden opgeteld tot een eindsaldo. De deductieve oplossing van dit soort fouten is gebaseerd op enkel (min)tekens aan te passen en variabelen in een record te verwisselen. Het kan namelijk zijn dat de respondent deze twee antwoorden door elkaar gehaald zou kunnen hebben. De doortelfout betekent dat de respondent tussendoor de berekeningen uitvoert en achteraf deze tussenstappen betrekt in het eindantwoord. Een gedetaileerde beschrijving van de tekenfout en de doortelfout is te vinden in [5].

De eenvoudige schrijffout houdt in dat er twee opeenvolgende cijfers per ongeluk verwisseld zijn, bijvoorbeeld in plaats van 843 is er 483 genoteerd. Het kan ook zo zijn dat er een cijfer is toegevoegd. Een voorbeeld daarvan is in plaats van 962 is er 9621 genoteerd. Het tegenovergestelde kan ook voorkomen, namelijk dat er een cijfer is weggelaten. Tevens komt het ook voor dat er een cijfer is vervangen door een ander cijfer, bijvoorbeeld in plaats van 963 is er 973 genoteerd. Dit soort fouten kunnen opgelost worden als er controleregels zijn in de vorm van lineare gelijkheden.

Een uitgebreid onderzoek naar deze verschillende deductieve fouten is verricht door methodoloog Sander Scholtus [5] en [6].

Een random fout is niet een systematische tekortkoming, maar eerder een fout die per ongeluk is opgetreden. Een fout is invloedrijk als de fout leidt tot een significante afwijking in schattingen van publicatiecijfers en een uitbijter is vaak gerelateerd aan een invloedrijke waarde. Een invloedrijke fout kan zowel random als systematisch zijn. Indien dit soort fouten voorkomen in een record wordt er voor gekozen om ze interactief gaaf te maken.

2.4 De foutlocalisatie

Aan het begin van het controle- en correctieproces vindt eerst de deductieve foutlocalisatie plaats. Vervolgens kan de dataset interactief of automatisch worden gaafgemaakt. Bij het gaafmaken moeten eerst de fouten worden gelocaliseerd en vervolgens worden de gelocaliseerde fouten gecorrigeerd. Dit gebeurt op basis van controleregels die edits worden genoemd. Dus een foute waarde wordt aangewezen indien een controleregel/edit wordt geschonden. We kennen twee verschillende edits. De harde edits zijn edits die zeggen dat elke schending is veroorzaakt door een fout. Er moet altijd worden voldaan aan de harde edits en harde edits mogen niet geschonden blijven in de data. Aan de zachte edits hoeft niet altijd te worden voldaan en de zachte edits mogen geschonden blijven in de data. Een schending van een zachte edit hoeft niet te betekenen dat de schending wordt veroorzaakt door een fout. De verzameling edits worden dus onderverdeeld in twee disjuncte deelverzamelingen: $\Psi = \Psi^H \bigcup \Psi^Z$. Elke ψ_k^H uit de verzameling Ψ^H is een harde edit, hiervoor geldt dus dat elk record moet voldoen aan deze edit. Elke ψ_k^Z uit de verzameling Ψ^Z is een zachte edit, hiervoor geldt dus dat er aan deze edit niet verplicht hoeft te worden voldaan [2]. Om te kunnen voldoen aan alle harde edits krijgen de variabelen in het record (x_j) betrouwbaarheidsgewichten (w_j) mee die net zoals de edits bepaald zijn door de experts. We definiëren de doelvariabelen als volgt:

$$y_j = \begin{cases} 1 & \text{als variabele } x_j \text{ wordt aangepast.} \\ 0 & \text{anders.} \end{cases} \tag{2.1}$$

Op dit moment vindt de foutlocalisatie in de praktijk plaats volgens **"het principe van Fellegi en Holt"**, dat luidt: pas zo min mogelijk variabelen aan, maar wel zodanig dat het veranderen van deze variabelen leidt tot een record dat voldoet aan alle harde edits [1]. In feite gebruiken we een generalisatie van dit principe, waarbij we de som van de betrouwbaarheidsgewichten minimaliseren. Doordat we zo min mogelijk aanpassingen willen doen, wordt de formule (2.2) geminimaliseerd met variabelen $j = 1, \ldots, n$:

$$\left\{ \sum_{j=1}^{n} w_j y_j \right\}. \tag{2.2}$$

Op het CBS is een algoritme ontwikkeld om dit probleem op te kunnen lossen en dit algoritme is geprogrammeerd in het computerprogramma SLICE [7]. Het doel is het zoeken naar een mogelijke verbetering van het foutlocalisatieprobleem. Bij het automatisch gaafmaken is er op het CBS tot nu toe gekozen om ofwel de zachte edits op te vatten als harde edits, ofwel de zachte edits weg te laten. Volgens [2] zijn beide keuzes onbevredigend, omdat sommige waarden ten onrechte als fout worden gezien bij het opvatten van de zachte edits als harde edits en bij het weglaten van deze zachte edits worden sommige fouten gemist. Daarom is een verbeterd foutlocalisatie-algoritme interessant voor het CBS. Hierbij is het idee van het CBS om elke zachte edit een "schendingsgewicht" (s_k) te geven. Om rekening te houden met de schending van de zachte edits voegen we een term D toe. Deze toevoeging geven we weer in formule (2.3). Dus als we de zachte edits willen betrekken bij de foutlocalisatie wordt aan de formule van (2.2) de D functie toegevoegd:

$$\left\{ \sum_{j=1}^{n} w_j y_j + D \right\} \tag{2.3}$$

waarbij de D staat voor

$$D(B_m) = \sum_{k \in B_m} s_k. \tag{2.4}$$

Hierbij is B_m de verzameling van geschonden zachte edits, waarbij $m = 1, \ldots, M$ verschillende voorgestelde verzamelingen geschonden zachte edits nummert. Bij een toegelaten oplossing voor het foutlocalisatieprobleem (d.w.z. een gegeven combinatie van aan te passen variabelen) bestaan er vaak verschillende verzamelingen zachte edits die geschonden kunnen blijven, waardoor er verschillende keuzes voor B_m mogelijk zijn, waarvan we er één moeten kiezen. De doelfunctie D uit formule (2.4) is de som van de schendingsgewichten van de zachte edits die op dat moment geschonden zijn. We kiezen de verzameling B_m waarvoor D(B_m) minimaal is gegeven de combinatie van aan te passen variabelen en deze minimale som noemen we $D(B_m^*)$.

We kunnen deze formule veralgemeniseren door een $\lambda \in (0, 1)$ toe te voegen, waarbij we formule (2.5) minimaliseren:

$$\left\{ \lambda \sum_{j=1}^{n} w_j y_j + (1-\lambda) D(B_m^*) \right\}. \tag{2.5}$$

Formule (2.3) is equivalent aan een speciaal geval van (2.5) door $\lambda = 0,5$ te stellen, zodat beide kanten even zwaar wegen. De B_m stelt de minimale verzameling edits voor die geschonden mogen blijven, gegeven de keuze voor $y_1, \ldots, y_n$. Er wordt gekozen voor de minimale som van schendingen en de minimale $D(B_m)$ noemen we $D(B_m^*)$. Met behulp van een voorbeeld lichten we dit toe:

Stel dat we de volgende verzameling met k edits hebben: {1, 2, 3, 4} en we hebben de volgende verzamelingen met voorgestelde edits B_m die we geschonden kunnen laten:

$B_1 = \{4\}$, $B_2 = \{1, 2, 3\}$ en $B_3 = \{1, 2\}$.

De schendingsgewichten van de zachte edits zijn {0,3; 0,7; 0,8; 0,9}. Hiermee berekenen we de som van de schendingsgewichten, zodat we de minimale schending kunnen nemen.

$$\begin{aligned} D(B_1) &= 0,9 \\ D(B_2) &= 0,3 + 0,7 + 0,8 = 1,8 \\ D(B_3) &= 0,3 + 0,7 = 1,0 \end{aligned}$$

We kiezen de verzameling met minimale som van schendingsgewichten, de B_m^*. In dit voorbeeld wordt er dus gekozen om B_1 geschonden te laten, omdat B_1 de minimale som van schendingsgewichten heeft. De zachte edits die niet in B_m^* zitten, moet aan worden voldaan, dus aan de edits 1, 2 en 3 in het voorbeeld.

Het algoritme van SLICE is uitgebreid om het probleem "minimaliseer (2.5), gegeven dat moet worden voldaan aan alle harde edits", op te kunnen lossen. Dit algoritme is als prototype geprogrammeerd in de taal R [8]. Voor het testen van de methoden in dit verslag hebben we gebruik gemaakt van het prototype-algoritme in R.

Elke zachte edit heeft een schendingsgewicht en zoals gegeven in het voorbeeld moeten deze schendingsgewichten eerst worden bepaald. Eén van de problemen is ook hoe deze schendingsgewichten het beste kunnen worden gemodelleerd, zodat het geautomatiseerde proces in grote mate overeenkomt met de traditionele uitvoering van de inhoudelijke experts die dit werk met de hand doen en de verschillen beperkt worden tot een minimum. De inhoudelijke experts besteden bij de handmatige foutlocalisatie veel aandacht aan de grootte van de schendingen, dus hoe groter de schending des te kleiner

de kans is om deze geschonden edit ook daadwerkelijk geschonden te laten. In eerste instantie nemen wij statische schendingsgewichten. De statische schendingsgewichten zijn eenvoudiger te bepalen en de methoden om deze schendingsgewichten te bepalen, behandelen we in hoofdstuk drie. Echter kunnen deze statische schendingsgewichten niet direct rekening houden met de grootte van de schendingen. Hierdoor gaan we in hoofdstuk vijf een andere methode uitproberen waarbij de schendingsgewichten niet vast worden gekozen. De schendingsgewichten worden dan automatisch bepaald, gegeven de schendingen, waarbij elk schendingsgewicht afhangt van de grootte van de bijbehorende schending en van de mate waarin een restrictie belangrijk is. In hoofdstuk acht wordt antwoord gegeven op de vraag welke van deze twee methoden het beste resultaat geeft.

2.5 De succesmaten

In deze paragraaf behandelen we twee manieren die gebruikt kunnen worden bij het beoordelen van de resultaten. Dus we gaan twee manieren toelichten die als maatstaf kunnen dienen bij de beoordeling van de resultaten verkregen uit bovenstaande methoden.

2.5.1 De succesmaten voor foutlocalisatie

Maat 1
Om de prestatie van de foutlocalisator te meten gebruiken we de alpha (α), de beta (β) en de gamma (γ) uit het boek van de Waal, Pannekoek en Scholtus:[7].

$$\alpha = \frac{\text{de gemiste fouten}}{\text{alle werkelijke fouten}} = \frac{FN}{TP + FN} \tag{2.6}$$

$$\beta = \frac{\text{de onterechte fouten}}{\text{alle werkelijke niet fouten}} = \frac{FP}{FP + TN} \tag{2.7}$$

$$\gamma = \frac{\text{de gemiste fouten + de onterechte fouten}}{\text{alle data}} = \frac{FN + FP}{TP + FN + FP + TN} \tag{2.8}$$

$$\delta = \text{voor hoeveel procent van de records hij de juiste fouten aanwijst.} \tag{2.9}$$

We hechten de meeste waarde aan de δ, omdat het de toegevoegde waarde weergeeft van de foutlocalisatie op record niveau.

De betekenis van de TP, FN, FP en de TN geven we hieronder:

- TP = true positive (de waarde is als fout aangewezen en was daadwerkelijk fout)
- FN = false negative (de waarde is als goed aangewezen en was eigenlijk fout)
- FP = false positive (de waarde is als fout aangewezen en was eigenlijk goed)
- TN = true negative (de waarde is als goed aangewezen en was daadwerkelijk goed)

De fracties α, β, γ en δ zullen we in het vervolg van dit verslag weergeven als percentages.

De TP, FN, FP en TN geven we weer in tabel 2.1

	Aangewezen foute waarden	Aangewezen goede waarden
Werkelijke foute waarden	TP	FN
Werkelijke goede waarden	FP	TN

Tabel 2.1: De gevonden fouten uitgezet tegen de werkelijke fouten

Hierbij moeten α, β en γ geminimaliseerd worden en de δ moet gemaximaliseerd worden. Bij de α gaat het om het minimaliseren van het aantal gemiste fouten de onterechte fouten doen er niet toe. Zo wordt er dan gezocht naar de meeste fouten. Bij de β is het van belang dat het aantal onterechte fouten wordt geminimaliseerd en de gemiste fouten hebben verder geen invloed. Zo wordt er dan gezocht naar de minste fouten. Bij de γ is het van belang dat het aantal verkeerde beslissingen wordt geminimaliseerd. Dus zowel de gemiste fouten als de onterechte fouten. Bij de δ willen we juist weten in hoeveel procent het de juiste fouten aanwijst. Uiteindelijk moeten de gevonden fouten ook de daadwerkelijke fouten zijn en daarom is dit percentage van groot belang en streven we naar een maximum van de δ.

Om een uitspraak over het geheel te kunnen doen, gebruiken we het percentage records waarin geen onterechte en gemiste fouten voorkomen, dus records waarin exact de juiste variabelen als fout worden aangewezen. Dit percentage noemen we δ.

Maat 2

Twee andere maten die lijken op bovenstaande maten zijn de sensitiviteit en specificiteit. Deze termen worden veel gebruikt in de geneeskunde en worden weergegeven in een ROC-curve waarbij de sensitiveit wordt uitgezet tegenover (1 -specificiteit). De sensitiviteit staat voor de gevoeligheid van een methode en de specificiteit voor hoe specifiek de verkregen resultaten van een methode zijn.

$$\text{De sensitiviteit} = \frac{TP}{TP + FP} \tag{2.10}$$

$$\text{De specificiteit} = \frac{TN}{TN + FN} \tag{2.11}$$

De sensitiviteit is het aantal gevonden terechte fouten, gedeeld door het totale aantal ruwe waarden dat als fout wordt aangewezen. De specificiteit is het aantal gevonden juiste waarden gedeeld door het totale aantal ruwe waarden dat als goed wordt aangewezen. Hoe hoger de sensitiviteit en specificiteit des te beter het verkregen resultaat is.

2.5.2 De succesmaten voor geïmputeerde waarden

Het succes van het geïmputeerde resultaat kunnen we meten met de dL1, m1 en de rdm volgens het boek van de Waal, Pannekoek en Scholtus: [7] (blz.410).

- dL1 = De gemiddelde absolute afstand tussen de geïmputeerde waarden en de correcte waarden.
- m1 = De absolute waarde van de gemiddelde afstand tussen de geïmputeerde waarden en de correcte waarden.
- rdm = De relatieve gemiddelde imputatie fout.

Hier neem je geen absolute waarden, je deelt het verschil tussen de geïmputeerde waarden en de oorspronkelijke waarde door de som van de correcte waarden per record. Hierbij is de $\hat{y}_i$ de geïmputeerde waarde in record i en y_i is de corresponderende correcte waarde. De M staat voor de verzameling records waarbij de waarden van $\hat{y}_i$ anders zijn dan de correcte waarden van y_i. De #M staat voor het aantal records in de verzameling M.

Een kleine waarde voor de dl1, m1 en de rdm duidt op goede imputatiewaarden.

$$d_{L1} = \frac{\sum_{i \in M} |\hat{y}_i - y_i|}{\#M} \tag{2.12}$$

$$m_1 = \left| \frac{\sum_{i \in M} (\hat{y}_i - y_i)}{\#M} \right| \tag{2.13}$$

$$rdm = \frac{\frac{1}{\#M} \sum_{i \in M} \hat{y}_i - \frac{1}{\#M} \sum_{i \in M} y_i}{\frac{1}{\#M} \sum_{i \in M} y_i} = \frac{\sum_{i \in M} (\hat{y}_i - y_i)}{\sum_{i \in M} y_i} \tag{2.14}$$

Hoofdstuk 3

Methoden met statische schendingsgewichten

In het vorige hoofdstuk hebben we kunnen lezen dat er verschillende controleregels, edits, bestaan om de fouten uit de data te halen. Zo hebben we de harde edits die altijd moeten gelden en de zachte edits die soms geschonden mogen blijven. Een voorbeeld van een harde edit dat al eerder gebruikt is, zullen we hier ook aanhalen: *omzet = kosten + winst*. Deze regel is altijd van toepassing en daardoor ook hard. Een voorbeeld van een zachte edit is: $winst \leq 0.6 \times omzet$ [3] Dit hoeft niet voor elk bedrijf te gelden. In uitzonderlijke gevallen kan het voorkomen dat de winst meer dan 60% van de omzet bedraagt. Daardoor mag deze edit geschonden blijven en hoeft hij dus niet in alle gevallen te gelden. Op het CBS wordt momenteel bij het automatisch gaafmaken van data alleen gebruik gemaakt van harde edits ofwel van zachte edits die als harde edits worden aangenomen. De zachte edits worden niet als zacht meegenomen en hierdoor worden sommige waarden ten onrechte als fout aangewezen of de fouten worden gemist. In dit hoofdstuk worden de zachte edits meegenomen als zacht en elke zachte edit krijgt een schendingsgewicht zoals aangekondigd in hoofdstuk één. Het bepalen van deze schendingsgewichten wordt in de eerste paragraaf behandeld. Om aan te kunnen tonen wat het effect is om de zachte edits op te vatten als zacht, nemen we de momenteel gebruikte methoden op het CBS bij het automatisch gaafmaken mee. Deze methoden zijn hieronder benoemd. Dit doen we zodat de verkregen resultaten van onderstaande drie methoden met elkaar kunnen worden vergeleken en zichtbaar gemaakt kan worden welke methode de meeste fouten localiseert.

- Alleen de harde edits, de zachte edits worden hier buiten beschouwing gelaten.

- De zachte edits als harde edits, waarbij dus alle edits (zowel harde als zachte edits) als hard worden beschouwd.
- Harde edits als hard en zachte edits als zacht met schendingsgewichten weergegeven in onderstaande methoden.

De eerste en tweede methoden werken zonder een schendingsgewicht, want bij de eerste methode zijn er geen zachte edits en bij de tweede methode worden de zachte edits als harde edits ingevoerd. De harde edits hebben geen schendingsgewichten. De variabelen hebben betrouwbaarheidsgewichten die worden bepaald door de experts. Echter houden we in eerste instantie de betrouwbaarheidsgewichten gelijk aan één, zodat we de toegevoegde waarde van de zachte edits goed zichtbaar kunnen maken.

In dit hoofdstuk onderscheiden we de volgende methoden:

1. De methoden voor het bepalen van de schendingsgewichten.
2. De formulering van de edits.

   We focussen op de coëfficiënten die voorkomen in de zachte edits door middel van de verdeling van de variabelen in de dataset en bepalen de kwantielen waarmee de kwantieledits ontstaan.
3. De verschillende vormen van de doelfunctie.

   Eén daarvan is dat in plaats van de som van schendingen zoals weergegeven in hoofdstuk één (1.4) de maximale schending wordt toegepast.
4. De nearest neighbormethode.

   De nearest neighbormethode zoekt in de referentiedata naar een gelijkwaardig record, waarmee vervolgens vergeleken wordt.
5. De chi-kwadraatmethode.

3.1 De statische schendingsgewichten voor de zachte edits

In hoofdstuk één hebben we kunnen lezen dat we gebruik maken van het principe van “Fellegi en Holt” bij het localiseren van fouten in data. Volgens dit principe mogen zo min mogelijk waarden in een record worden aangepast om te kunnen voldoen aan alle edits. Het meenemen van zachte edits zoals in formule (2.5) is in principe een uitbreiding op het principe van Fellegi en Holt. Hierbij is B_m de verzameling van voorgestelde zachte edits die geschonden mogen blijven met verzameling $m = 1, \ldots, M$.

$$D(B_m) = \sum_{k \in B_m} s_k. \tag{3.1}$$

Het schendingsgewicht s_k geeft aan hoe belangrijk het is dat een record voldoet aan deze edit [3]. In deze paragraaf nummeren we de diverse methoden die gebruikt kunnen worden bij de bepaling van het schendingsgewicht s_k.

A: Elke zachte edit krijgt een schendingsgewicht $s_k^A = 1$.

B: Het schendingsgewicht s_k^B is de fractie records in de referentiedata die de k^{de} edit niet schendt. In dit geval is de referentiedata de handmatig gaafgemaakte data van vorig jaar. We gaan kijken in de referentiedata hoe vaak een zachte edit niet wordt geschonden. Het aantal niet schendingen per edit gedeeld door het totale aantal records, is de fractie records die de edit niet schendt. Deze fractie noemen we s_k^B. Per edit bekijken we dus het aantal niet schendingen in de referentiedata. Als er weinig schendingen zijn in de referentiedata dan willen we na het automatisch gaafmaken van de ruwe data ook weinig schendingen overhouden. Hoe hoger het schendingsgewicht van een zachte edit, des te betrouwbaarder de zachte edit is. Met formule (3.2) bepalen we de schendingsgewichten s_k^B.

$$s_k^B = \left\{ \frac{1}{T} \sum_{i=1}^{T} z_{ki} \right\} \tag{3.2}$$

$$z_{ki} = \begin{cases} 1 & \text{als referentierecord } i \text{ voldoet aan de zachte edit } k. \\ 0 & \text{anders.} \end{cases} \tag{3.3}$$

T = het totale aantal referentierecords.

C: Het schendingsgewicht s_k^C verkrijgen we door de verkregen schendingsgewichten s_k^B te verdelen tussen $\{0{,}5;\ 1;\ 1{,}5\}$. De verdeling hebben we zo gekozen omdat de betrouwbaarheidsgewichten gelijk aan één zijn. Deze verdeling is gebaseerd op de verkregen schendingsgewichten s_k^B, omdat we de fractie niet schendingen in de referentiedata willen gebruiken. Daarbij kiezen we de verdeling bewust rondom de één, omdat de betrouwbaarheidsgewichten van de variabelen gelijk aan één zijn. We kiezen twee waarden $\hat{a}$ en $\hat{b}$ uit s_k^B waarbij $k = 1, \ldots, K$ en bepalen de nieuwe schendingsgewichten als volgt:

De $\hat{a}$ is de ondergrens en de $\hat{b}$ is de bovengrens van schendingsgewichten s_k^B, waarvoor geldt dat $\hat{a} \leq \hat{b}$. In het geval dat $\hat{a} = \hat{b}$, kunnen we de ondergrens en de bovengrens bepalen door het gemiddelde te nemen

van de schendingsgewichten s_k^B. Indien $\hat{a} \neq \hat{b}$ hebben we twee waarden uit de verdeling nodig.

$$\begin{aligned}
&\text{Als } \; s_k^B < \hat{a} \; \text{ is het schendingsgewicht } \; s_k^C = 0,5 \\
&\text{Als } \; \hat{a} \leq s_k^B \leq \hat{b} \; \text{ is het schendingsgewicht } \; s_k^C = 1 \\
&\text{Als } \; s_k^B > \hat{b} \; \text{ is het schendingsgewicht } \; s_k^C = 1,5
\end{aligned}$$

Als voorbeeld nemen we een dataset die we hebben gebruikt bij het vinden van een geschikte methode voor het bepalen van de schendingsgewichten. Hierbij hadden we dezelfde $\hat{a} = \hat{b} = 0,95$. Dat betekent dat we de volgende verdeling van schendingsgewichten s_k^C hebben:

$$\begin{aligned}
&\text{Voor } \; s_k^B < 0,95 \; \text{ geldt } \; s_k^C = 0,5 \\
&\text{Voor } \; s_k^B = 0,95 \; \text{ geldt } \; s_k^C = 1 \\
&\text{Voor } \; s_k^B > 0,95 \; \text{ geldt } \; s_k^C = 1,5
\end{aligned}$$

De reden dat we hier kiezen voor een 1:2:3 verhouding ofwel de gewichten verdelen naar aanleiding van een s_k^C van 0,95 heeft te maken met de schending van de edits in de referentiedataset. We hebben deze verdeling gekozen, omdat we 0,95 in deze dataset kunnen beschouwen als een tussenliggende waarde. Eventueel kan ook het gemiddelde van de schendingsgewichten s_k^B worden gebruikt, om de tussenliggende waarde te bepalen. Het feit dat we het onderverdelen in drie klassen, heeft te maken met de betrouwbaarheidsgewichten, die in deze dataset allen gelijk aan één zijn. Daardoor hebben we voor de schendingsgewichten s_k^C een gewicht erboven, een schendingsgewicht gelijk aan het betrouwbaarheidgewicht en een schendingsgewicht eronder gekozen. Dit lichten we hieronder met behulp van (3.4) en (3.5) verder toe. Een hoog schendingsgewicht betekent weinig schendingen en een laag schendingsgewicht betekent veel schendingen. Dus als de edit in minder dan 5% van de referentierecords wordt geschonden, is hij betrouwbaarder dan wanneer hij in meer dan 5% van de referentierecords wordt geschonden en daarom krijgt hij een laag schendingsgewicht bij een $s_k^B < 0,95$. Er worden verzamelingen van geschonden edits weergegeven, zoals aangekondigd in hoofdstuk 1 formule (1.4). Uiteindelijk wordt er gekozen voor de minimale som van de schendingen om de edits geschonden te laten. De minimale som functie van de edits die geschonden mogen blijven, noemen we $D(B_m^*)$. Hiermee is onderstaande formule (3.5) minimaal. Hierbij is $\lambda = 0,5$, zodat beide kanten even zwaar wegen (harde- en zachte edits). Hierbij zijn de betrouwbaarheidsgewichten $w_j = 1$ met variabelen $j = 1, \ldots, n$ zo danig dat de variabele y_j gelijk is aan:

$$y_j = \begin{cases} 1 & \text{als variabele } x_j \text{ wordt aangepast.} \\ 0 & \text{anders.} \end{cases} \tag{3.4}$$

$$\left\{ \lambda \sum_{j=1}^{n} w_j y_j + (1-\lambda) D(B_m^*) \right\}. \tag{3.5}$$

Dit houdt in dat er na de minimale schending van de zachte edits, wordt gekeken naar de totale kosten, dus inclusief de betrouwbaarheidsgewichten. Dit is tevens uitgebreid beschreven in paragraaf 2.4. Vandaar dat we bij deze methode kijken of het overschrijden van de betrouwbaarheidsgewichten door de schendingsgewichten s_k^C wel of geen effect heeft op de totale kosten.

Een ander voorbeeld is dat we de schendingsgewichten van s_k^C bepalen door $\hat{b} = 0,99$ als bovengrens te kiezen en $\hat{a} = 0,97$ als ondergrens. Dat betekent dat we de volgende verdeling van schendingsgewichten s_k^C krijgen:

$$\begin{aligned} &\textit{Als } s_k^B < 0,97 \textit{ is het schendingsgewicht } s_k^C = 0,5 \\ &\textit{Als } 0,97 \leq s_k^B \leq 0,99 \textit{ is het schendingsgewicht } s_k^C = 1 \\ &\textit{Als } s_k^B > 0,99 \textit{ is het schendingsgewicht } s_k^C = 1,5 \end{aligned}$$

D: De schendingsgewichten s_k^D verkrijgen we door de edits met de laagste schendingsgewichten s_k^B te verwijderen. Er hoeft niet meer te worden voldaan aan deze verwijderde edits. Dus alle edits met een schendingsgewicht kleiner dan â verwijderen we, omdat de edits met de laagste waarden van s_k^B het minst betrouwbaar zijn. Alle edits met een schendingsgewicht groter of gelijk aan â geven we de schendingsgewichten s_k^B mee.

E: De schendingsgewichten s_k^E berekenen we met behulp van de onderstaande tabel die per edit wordt gecreëerd. Dit lichten we met een voorbeeld toe. Stel dat we edit $\sum_{j=1}^{n} a_{kj} x_j \geq b_k$ meekrijgen voor een dataset. De resultaten van deze edit bij toepassing op een referentiedataset zetten we uit in tabel 3.1. Hierbij horen de gave en de ruwe referentiedataset bij elkaar: elk record heeft een ruwe en een gave versie. In de tabel staat dus steeds het aantal records waarvan de ruwe versie wel of niet voldoet aan een edit en de gave versie wel of niet voldoet aan dezelfde edit.

De betekenis van de getallen in de tabel zijn als volgt: stel dat de zachte edit in de gave dataset en in de ruwe dataset 8 keer wordt geschonden.

Deze edit voldoet 12 keer aan de gave dataset en schendt de ruwe data. Het aantal keer dat deze edit voldoet aan de ruwe dataset, maar de gave dataset schending is 5 en het aantal keer dat hij voldoet aan beide datasets is 539 keer.

	Data gaaf schendt	Data gaaf voldoet
Data ruw schendt	8	12
Data ruw voldoet	5	539

Tabel 3.1: edit 1 toegepast op de referentiedata.

In totaal hebben we 580 records. We nemen het totaal aantal foute waarden in de gave dataset en delen deze door het totaal aantal records, waarbij we de ontbrekende waarden buiten beschouwing laten. We delen (8+5) door (580-16) en dat is gelijk aan $\frac{13}{564}$. Dit geeft ongeveer 0,02, echter betekent een groot schendingsgewicht dat hij betrouwbaar is. Dus we moeten dan één min het verkregen antwoord doen, willen we schendingsgewicht s_k^E krijgen. De ontbrekende waarden zijn in dit geval gelijk aan (580-(539+8+12+5)= 16).

We bekijken bovenstaande voorbeeld nu in het algemeen:

	Data gaaf schendt	Data gaaf voldoet
Data ruw schendt	a	b
Data ruw voldoet	c	d

Tabel 3.2: Algemene naamgeving voor schendingen en voldoeningen van gave en ruwe data.

De verklaring van de naamgevingen:

- a is het aantal schendingen van de edit door de ruwe data en de gave data.
- b is het aantal schendingen van de edit door de ruwe waarden, terwijl hij voldoet aan de gave data.
- c is het aantal schendingen van de edit door de gave data, terwijl hij voldoet aan de ruwe data.
- d is het aantal dat de edit voldoet aan zowel de ruwe data als de gave data.

De schendingsgewichten s_k^E berekenen we met:

$$s_k^E = 1 - \frac{a + c}{(T - NA)}. \tag{3.6}$$

De "NA" waarden staan voor de ontbrekende waarden en de T staat voor het totale aantal records in de dataset.

F: De schendingsgewichten s_k^F zijn het totale aantal schendingen in de gave dataset gedeeld door het totale aantal niet schendingen in de gave dataset. De formule voor de schendingsgewichten is:

$$s_k^F = 1 - \frac{a+c}{b+d}. \tag{3.7}$$

Volgens tabel 3.1 en met behulp van formule (3.7) krijgen we $\frac{(8+5)}{(12+539)} = \frac{13}{551}$. Dit geeft ongeveer $0,02$, echter betekent een groot schendingsgewicht dat hij betrouwbaar is en in dit geval moeten we dus $(1-0,02) = s_k^F = 0,98$ als schendingsgewicht meegeven.

G: De schendingsgewichten s_k^G berekenen we door het aantal records waarbij een schending in de ruwe data is opgelost tijdens het gaafmaken te delen door het totale aantal schendingen in de ruwe dataset.

$$s_k^G = \frac{b}{a+b}. \tag{3.8}$$

Volgens tabel 3.1 en met formule (3.8) geeft dit als schendingsgewicht: $\frac{12}{(8+12)} = \frac{12}{20} = 0,60 = s_k^G$ We kunnen dit ook opvatten als een kans. Hierbij willen we weten wat de kans is voor het geval dat de gave data geschonden is, gegeven dat de ruwe data ook is geschonden. Laten we aannemen dat A staat voor de ruwe data en de B staat voor de gave data. We kunnen deze kans met behulp van de "regel van Bayes" als volgt uitrekenen:

$$s_k^G = 1 - \hat{P}(B|A) = 1 - \frac{\hat{P}(A \wedge B)}{\hat{P}(A)}. \tag{3.9}$$

Volgens tabel 3.1 en met formule (3.9) geeft dit ook als schendingsgewicht: $s_k^G = 1 - \hat{P}(B|A) = 1 - \frac{8}{20} = 0,60$.

H: De schendingsgewichten s_k^H verkrijgen we met behulp van de schendingsgewichten s_k^G, volgens dezelfde werkwijze als bij het bepalen van de schendingsgewichten s_k^C, waarbij we kiezen voor een boven- of ondergrens. Dit kan bijvoorbeeld met het gemiddelde van de schendingsgewichten s_k^G. Het verschil met methode C is dat we de verdeling van de schendingsgewichten s_k^H baseren op s_k^G en niet zoals bij de schendingsgewichten van s_k^C op s_k^B. Vervolgens verdelen we de schendingsgewichten s_k^G als volgt om de schendingsgewichten s_k^H te verkrijgen:

$$s_k^G < \hat{a} \text{ geven } \; s_k^H = 0,5$$
$$\hat{a} \leq s_k^G \leq \hat{b} \text{ geven } \; s_k^H = 1$$
$$s_k^G > \hat{b} \text{ geven } \; s_k^H = 1,5$$

I: De schendingsgewichten s_k^I verkrijgen we door de kans uit te rekenen dat de ruwe data (A) geschonden is, gegeven dat de gave data (B) ook is geschonden. We kunnen deze kans berekenen met behulp van de "regel van Bayes" en de formule:

$$s_k^I = \hat{P}(A|B) = \frac{\hat{P}(A \wedge B)}{\hat{P}(B)}. \tag{3.10}$$

Volgens tabel 3.1 en met formule (3.10) geeft dit als schendingsgewicht:

$$s_k^I = \hat{P}(A|B) = \frac{8}{13} \approx 0,615 \tag{3.11}$$

Deze methode zou zinvol kunnen zijn, omdat we rekening houden met de schendingen van de zachte edit in de referentiedata en gegeven dat proberen te bepalen wat de kans is dat de edit ook wordt geschonden in de ruwe data.

J: De verschilmethode:

We beschikken over een ruwe referentiedataset en een gave referentiedataset met edits. Stel dat edit k gelijk is aan $x_1 \geq x_2$ die hoort bij deze dataset. Dan nemen we het verschil tussen x_1 en x_2 in de gave referentiedata voor record i en dit noemen we g_i. Het verschil tussen de x_1 en de x_2 in de ruwe dataset voor record i noemen we r_i. Hierbij laten we de ontbrekende waarden in de ruwe referentiedataset achterwege, omdat we anders geen verschil kunnen berekenen tussen het ruwe record en het gave record. Vervolgens delen we het absolute verschil van g_i en r_i door het absolute verschil van g_i en nemen daarvan de som. Dit geeft formule (3.12):

$$v_k = \sum_{i=1}^{T} \frac{|r_i - g_i|}{|g_i|}. \tag{3.12}$$

De x_1 en de x_2 zijn de variabelen die voorkomen in een record en we berekenen de v_k op basis van de ruwe en de gave dataset. Vervolgens gaan we met behulp van de verkregen v_k de schendingsgewichten bepalen, zoals bij methode C, waarbij we de verkregen verschillen

verdelen. Dit kan met behulp van een boven- en/of ondergrens gedaan worden. Echter hebben wij ervoor gekozen om dit anders te verdelen, omdat de berekende verschillen (v_k) zodanig klein zijn. De verschillen (v_k) zijn opgenomen in bijlage D.

Een voorbeeld uit een dataset zijn onderstaande schendingsgewichten die met behulp van de berekende verschillen (v_k) zijn toegekend. Een v_k van nul betekent dat het ruwe record niet afwijkt van het gave record in de referentiedata en daardoor krijgt deze edit een hoog schendingsgewicht. Een v_k dat een groot verschil heeft, betekent dat het ruwe record veel afwijkt van het gave referentierecord en is dus redelijk onbetrouwbaar. Deze edit krijgt daarom een laag schendingsgewicht. Volgens de berekende verschillen in bijlage D verdelen we de schendingsgewichten als volgt:

Indien $v_k = 0$, geeft dit als schendingsgewicht $s_k^J = 2$.
Indien $0 < v_k \leq 0,001$, geeft dit als schendingsgewicht $s_k^J = 1,5$
Indien $0,001 < v_k \leq 0,01$, geeft dit als schendingsgewicht $s_k^J = 1,0$
Indien $0,01 < v_k \leq 1$, geeft dit als schendingsgewicht $s_k^J = 0,5$
Indien $1 < v_k \leq 10$, geeft dit als schendingsgewicht $s_k^J = 0,2$.
Indien $10 < v_k$, geeft dit als schendingsgewicht $s_k^J = 0,1$.

De reden dat we de ontbrekende waarden niet meenemen bij het berekenen van het verschil heeft te maken met het feit dat het invullen van een nul op de ontbrekende waarden niet een betrouwbare manier is om het verschil uit te rekenen. Toch hebben we dit uitgeprobeerd en het resultaaat is opgenomen in bijlage F, waarbij er een niet groot verschil is ontstaan en de resultaten hiervan worden besproken in hoofdstuk vier.

3.2 De kwantieledits

Het voorbeeld dat we al eerder hebben gezien van een zachte edit: $winst \leq 0,6 \times omzet$, gebruiken we hier om deze paragraaf toe te lichten. De reden dat deze zachte edit wordt gehanteerd is, omdat bedrijven waarvan de winst meer dan 60% van de omzet bedraagt worden gekenmerkt als zeldzame bedrijven. Echter is het zo dat in enkele situaties een gaafmaker 62% van de omzet bij dat bedrijf een goede winst vindt en weer bij een ander bedrijf zal dat 57% van de omzet zijn. Dus dan hebben we bij het ene bedrijf niet de zachte edit: $winst \leq 0,6 \times omzet$, maar een zachte edit: $winst \leq 0,62 \times omzet$. Bij het andere bedrijf hebben we dan

de zachte edit: $winst \leq 0,57 \times omzet$. Doordat er aan een zachte edit niet altijd hoeft te worden voldaan, zijn bovenstaande aanpassingen niet beslist fout. De vraag die hieruit voortvloeit, is welke waarde je dan moet nemen voor de coëfficiënt(en) in de zachte edit. De kwantieledits zijn een manier om dit soort coëfficiënten te kiezen op een systematische manier, namelijk gebaseerd op de verdeling in een eerder gaafgemaakte dataset van de verhouding tussen de variabelen in de edit. In deze paragraaf wordt behandeld hoe deze coëfficiënten bepaald kunnen worden.

Als voorbeeld nemen we edit: $x_2 - 0,5x_3 \geq 0$. Hier is niet te zeggen voor welke waarden van de coëfficiënt er minder zachte edits geschonden zullen worden. De coëfficiënt van 0,5 is niet een exact bepaalde waarde. Daarom is het in eerste instantie van belang om de verhouding te bepalen tussen de x_2 en de x_3, waarna we in de referentiedata kunnen kijken naar de kwantielen. Voor dit onderzoek hebben we drie verschillende datasets gebruikt van de PS en daarbij hebben we met name gebruik gemaakt van het 1% kwantiel, het 5% kwantiel en het 10% kwantiel. Indien de oorspronkelijk edits bijvoorbeeld $x_2 - 0,5x_3 \leq 0$ hebben in plaats van $x_2 - 0,5x_3 \geq 0$, is er gekeken naar het 99% kwantiel, 95% kwantiel en naar het 90% kwantiel. De verkregen waarde bij het 1% kwantiel is ingevuld op de plaats van de coëfficiënt 0,5 en daarmee is er een nieuwe edit ontstaan die we de 1% kwantieledit noemen.

Bij het bepalen van de kwantieledits houden we rekening met:

1. De edit: $x_2 - 0,5x_3 \geq 0$, waarbij de x_2 en de x_3 niet de waarde nul mogen aannemen. Als x_2 de waarde nul heeft, zal dit een negatief antwoord geven, wat niet toegestaan is, aangezien we als harde edit hebben dat x_3 niet kleiner mag zijn dan nul. Als x_3 de waarde nul heeft, zal de coëfficiënt geen rol meer spelen en dit willen wij voorkomen, omdat we juist opzoek zijn naar een geschikte coëfficiënt voor deze edit. Dus die gevallen laten we buiten beschouwing in het bepalen van het 10% kwantiel, 5% kwantiel en het 1% kwantiel.

2. De edits van de vorm $x_5 + x_6 - x_7 \geq 0$ kunnen niet meegenomen worden bij het bepalen van de kwantielen. Dit komt doordat er meer dan twee varaiabelen zijn en de coëfficiënt van deze edit in dit geval gelijk zijn aan één. Daarom hanteren we voor deze gevallen schendingsgewichten s_k^B.

De 10%, de 5% en de 1% kwantielen in combinatie met een schendingsgewicht geven goede resultaten. Dit zullen we hieronder verder toelichten.

Stel dat we de volgende zachte edits hebben met daarbij de 10%, de 5% en de 1% kwantielen.

Naar aanleiding van de berekende kwantielen zoals bovenstaand zijn de kwantieledits ontstaan en deze worden weergegeven in de tabel hieronder.

De oorspronkelijke edits	10% kwantiel	5% kwantiel	1% kwantiel
$x_2 - 0,5x_3 \geq 0$	0,75	0,60	0,10
$x_3 - 0,9x_9 \geq 0$	0,98	0,93	0,62
$x_5 + x_6 - x_7 \geq 0$	-	-	-
$x_9 - 50x_{12} \geq 0$	110	85	40
$5000x_{12} - x_9 \geq 0$	2126	5228	15347
$0,4x_9 - x_{11} \geq 0$	0,19	0,27	0,52
$10x_{11} + x_9 \geq 0$	10	42	334

Tabel 3.3: De oorspronkelijke zachte edits en de berekende kwantielen

Oorspronkelijke edits	**10% kwantiel-edits**	**5% kwantiel-edits**	**1% kwantiel-edits**
$x_2 - 0,5x_3 \geq 0$	$x_2 - 0,75x_3 \geq 0$	$x_2 - 0,6x_3 \geq 0$	$x_2 - 0,1x_3 \geq 0$
$x_3 - 0,9x_9 \geq 0$	$x_3 - 0,98x_9 \geq 0$	$x_3 - 0,93x_9 \geq 0$	$x_3 - 0,62x_9 \geq 0$
$x_5 + x_6 - x_7 \geq 0$	$x_5 + x_6 - x_7 \geq 0$	$x_5 + x_6 - x_7 \geq 0$	$x_5 + x_6 - x_7 \geq 0$
$x_9 - 50x_{12} \geq 0$	$x_9 - 110x_{12} \geq 0$	$x_9 - 85x_{12} \geq 0$	$x_9 - 40x_{12} \geq 0$
$5000x_{12} - x_9 \geq 0$	$2126x_{12} - x_9 \geq 0$	$5228x_{12} - x_9 \geq 0$	$15347x_{12} - x_9 \geq 0$
$0,4x_9 - x_{11} \geq 0$	$0,19x_9 - x_{11} \geq 0$	$0,27x_9 - x_{11} \geq 0$	$0,52x_9 - x_{11} \geq 0$
$10x_{11} + x_9 \geq 0$	$10x_{11} + x_9 \geq 0$	$42x_{11} + x_9 \geq 0$	$334x_{11} + x_9 \geq 0$

Tabel 3.4: De kwantieledits

Echter kunnen we naar aanleiding van dit voorbeeld de algemene vorm voor deze methode als volgt beschrijven:

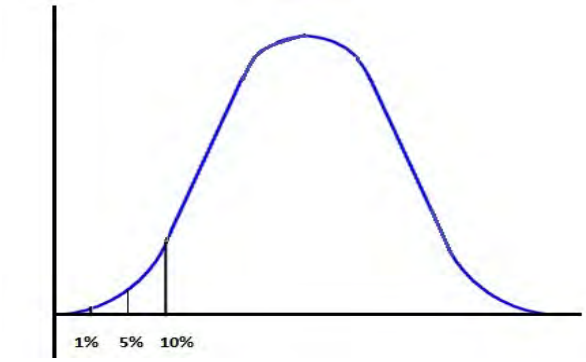
1. We hebben een zachte edit van de vorm $x_{j1} - cx_{j2} \geq 0$. Hierbij zijn de x_{j1} en de x_{j2} variabelen van de edit en de c is coëfficiënt in deze edit waarvoor we een geschikte waarde willen bepalen.

2. Bereken de verhouding tussen variabelen van de zachte edit, dus de verhouding tussen de x_{j1} t.o.v. de x_{j2} voor alle records in een dataset met gave referentiedata. Noteer de 10%, de 5% en de 1% kwantielen van de verdeling van deze verhouding als $\hat{c}_{10}, \hat{c}_5, \hat{c}_1$

3. Formuleer de volgende edits met behulp van de verkregen kwantielen: $x_{j1} - \hat{c}_{10}x_{j2} \geq 0$, $x_{j1} - \hat{c}_5x_{j2} \geq 0$ en $x_{j1} - \hat{c}_1x_{j2} \geq 0$

4. De 10% kwantieledit $x_{j1} - \hat{c}_{10}x_{j2} \geq 0$ is geschonden in 10% van de gave referentierecords. De 5% kwantieledit $x_{j1} - \hat{c}_5x_{j2} \geq 0$ is geschonden in 5% van de gave referentierecords en de 1% kwantieledit $x_{j1} - \hat{c}_1x_{j2} \geq 0$ is geschonden in 1% van de gave referentierecords. Als de 5% kwantieledit wordt geschonden, dan is de 10% kwantieledit ook geschonden. We kunnen dit ook interpreteren als een dichtheid

zoals weergegeven in figuur 3.1. Hierin kunnen we zien dat als de 1% kwantieledits geschonden worden dat het valt buiten de marge van de 5% en de 10% edits. Indien we de drie edits (10%, 5% en de 1% kwantieledits) tegelijk meenemen, kan er bij het automatisch gaafmaken onderscheid worden gemaakt tussen de verschillende soorten schendingen, zoals weergegeven in figuur 3.1. Eventueel zou dit uitgebreid kunnen worden met meerdere kwantieledits, maar dit zal het foutlocalisatieprobleem moeilijker oplosbaar maken. Hoe meer edits, hoe complexer het probleem wordt. Daardoor beperken we ons tot de drie kwantieledits.



Figuur 3.1: Schematische weergave van de kwantielen.

Stel dat we de eerste kwantieledits hebben uit tabel 3.4 :

$x_2 - 0,75x_3 \geq 0$ als de 10% kwantieledit, $x_2 - 0,6x_3 \geq 0$ als de 5% kwantieledit en $x_2 - 0,1x_3 \geq 0$ als de 1% kwantieledit. Laten we aannemen dat $x_2 = 1$ en dat $x_3 = 11$. Tevens hebben we de harde edits: $x_2 \geq 0$ en $x_3 \geq 0$, waaraan voldaan is. Echter wordt de 1% kwantieledit geschonden, omdat $1 - 0,1 * 11 = -0,1$. De 5% kwantieledit wordt ook geschonden, omdat $1 - 0,6 * 11 = -5,6$ en de 10% kwantieledit wordt ook geschonden, want $1 - 0,75 * 11 = -7,25$. Dus als de 1% kwantieledit geschonden wordt, worden ook de 5% en de 10% kwantieledits geschonden, omdat de schendingsgrootte steeds groter wordt.

5. Bij een edit van de vorm $cx_{j1} - x_{j2} \geq 0$, gaan we analoog te werk, maar dan gebruiken we het 90%, 95% of 99% kwantiel in plaats van het 10%, 5% of 1% kwantiel van de verdeling van x_{j1} t.o.v. x_{j2} in de referentiedata.

Met behulp van de verkregen kwantieledits kunnen we nu de foutlocalisatie uitvoeren. Echter zijn deze nieuwe edits ook zachte edits en krijgen allen

een schendingsgewicht. Mogelijke schendingsgewichten zullen we hieronder beschrijven.

Type kwantieledits:

I De 10%, 5% en 1% kwantieledits apart waarbij elk soort kwantieledit afzonderlijk met schendingsgewicht $s_k^A = 1$ wordt uitgevoerd.

II De kwantieledits met schendingsgewicht 1: De 10%, 5% en 1% kwantieledits tegelijkertijd meenemen en alle een schendingsgewicht $s_k^A = 1$ meegeven.

III De kwantieledits met wegingsom 1: We gebruiken alle drie de kwantieledits, waarbij de edits als som een schendingsgewicht 1 krijgen, per edit wordt een weging van 0,33 gegeven.

IV De kwantieledits met wegingsom 3: We gebruiken alle drie de kwantieledits, waarbij de edits als som een schendingsgewicht 3 krijgen. De 10% kwantieledits krijgen een schendingsgewicht 0,5. De 5% kwantieledits krijgen een schendingsgewicht 1,0 en de 1% kwantieledits krijgen een schendingsgewicht 1,5.

V De kwantieledits met weging 0,9 - 0,05 - 0,05: In stap 4 van de beschrijving van de algemene vorm hebben we kunnen lezen dat de schending van de 1% kwantieledit, de 5% en 10% kwantieledits ook geschonden worden, op basis hiervan kiezen we de drie schendingsgewichten van 0,9-0,05-0,05. Dit betekent dat we alle drie de kwantieledits meenemen, waarbij de 10% kwantieledits een schendingsgewicht van 0,9 krijgen, de 5% kwantieledits een schendingsgewicht van 0,05 krijgen en de 1% kwantieledits een schendingsgewicht van 0,05 krijgen. Indien alleen de 10% en de 5% kwantieledits worden geschonden zal het totale schendingsgewicht van (0,9+0,05 =) 0,95 worden. Als de 1% kwantieledits geschonden worden, dan zijn de 5% en de 10% kwantieledits ook geschonden, waardoor er dus een totale schending met schendingsgewicht één ontstaat. Dit zal in de doelfunctie D(B) dan een totaal schendingsgewicht van (0,9+0,05+0,05 =) 1 geven. Bij een maximale schending zal het totale gewicht niet boven de één uitkomen en daarmee de betrouwbaarheidsgewichten niet overschrijden. In de volgende methode kiezen we de betrouwbaarheidsgewichten zodanig dat ze de betrouwbaarheidsgewichten overschrijden bij een totale schending. Dit wordt verder uitgelegd in methode VI.

VI De kwantieledits met weging 0,9 - 0,05 - 0,10: Bij deze methode gebruiken we alle drie de kwantieledits, waarbij de 10% kwantieledits een gewicht van 0,90 krijgen, de 5% kwantieledits een gewicht van 0,05 en de 1% kwantieledits een gewicht van 0,10. Op deze manier

worden de betrouwbaarheidsgewichten overschreden in geval dat de 1% kwantieledit is geschonden. Indien alleen de 10% en 5% kwantieledits worden geschonden, hebben we een totaal schendingsgewicht van 0,95 en overschrijdt het schendingsgewicht niet de betrouwbaarheidsgewichten. Het totale schendingsgewicht wordt 1,05 in geval van schending van de 1% kwantieledit. Er is geen direct verband met de betrouwbaarheidsgewichten, maar uit formule (3.5) blijkt dat het van belang is voor de totale kosten van de gevonden oplossing.

VII De schendingsgewichten s_k^K verkrijgen we door de 1% kwantieledits te vergelijken met de 5% kwantieledits. Dit doen we omdat we weten dat als de 1% kwantieledit wordt geschonden, dat dan ook het 5% en 10% kwantieledit zijn geschonden. Tevens geldt voor de 5% kwantieledit dat dan ook de 10% kwantieledit is geschonden. Daarom vergelijken we het 1% kwantiel met het 5% kwantiel. Als het 1% kwantiel minder dan twee keer verschilt van het 5% kwantiel in de referentiedata, is het gewicht van de zachte edit 0,75 . Als het ongeveer twee keer verschilt, is het gewicht van de zachte edit 0,50 en als het ongeveer drie keer verschilt, is het gewicht van de zachte edit 0,25. Hoe hoger het gewicht, des te betrouwbaarder de edit is. Afhankelijk van de schendingsfractie van de kwantieledits in de referentiedata hebben deze edits gewichten tussen de 0,25 en 0,75 gekregen voor de edit $k = 1, \ldots, K$. Op deze manier overschrijden ze

$$\begin{array}{lll} \text{Als} & \frac{\hat{c}_5}{\hat{c}_1} < 2 & \text{dan } s_k^K = 0,75 \\ \text{Als} & 2 \leq \frac{\hat{c}_5}{\hat{c}_1} \leq 3 & \text{dan } s_k^K = 0,50 \\ \text{Als} & \frac{\hat{c}_5}{\hat{c}_1} > 3 & \text{dan } s_k^K = 0,25 \end{array}$$

VIII De schendingsgewichten s_k^Lverkrijgen we door de schendingsgewichten s_k^Kte verdubbelen. Op deze manier hebben we grotere schendingsgewichten en de resultaten hiervan worden besproken in paragraaf 4.2. De verdubbeling van de schendingsgewichten van s_k^K heeft te maken met de betrouwbaarheidsgewichten die we in eerste instantie gelijk aan één hebben gekozen en verdubbelen om te zien of een groter schendingsgewicht invloed heeft op de foutlocalisatie.

3.3 De verschillende vormen van de doelfunctie

In deze paragraaf behandelen we een nieuwe methode, waarbij we in plaats van de som van de schendingen zoals weergegeven in formule (3.1) gaan kijken naar het maximum van de schendingen. In hoofdstuk twee hebben

we met een voorbeeld laten zien dat de $D(B_m^*)$ uit formule (2.5) de minimale som van de schendingen is. De $D(B_m)$ is de doelfunctie voor de m^{de} verzameling met de som van de schendingsgewichten van de voorgestelde edits, zoals weergegeven in formule (3.1). In deze paragraaf nemen we voor de $D(B_m^*)$ niet de minimale som, maar passen het minimax criterium toe, waarbij we het maximum bepalen voor de $D(B_m)$ en daar vervolgens de verzameling met de minimale schending uit kiezen.

Vervolgens gaan we de λ uit formule (3.4) aanpassen. Tot nu toe hebben we aangenomen dat de λ gelijk is aan 0,5, zodat beide kanten (harde en de zachte edits) even zwaar wegen bij het bepalen van de totale kosten van de gevonden oplossing. De verschillende methoden waarbij we de som en/of de λ aanpassen zijn hieronder weergegeven.

1. De maximale weging: Tot nu hebben we gezien dat er wordt gekozen voor de som uit de verzameling voorgestelde edits om geschonden te laten (3.1). In plaats van de som nemen we bij deze methode het maximum van de schendingsgewichten. Voor de maximale weging krijgen we:

$$D(B_m) = \max_{k \in B_m} s_k. \tag{3.13}$$

   Met een voorbeeld geven we het verschil tussen de oorspronkelijke formule (3.1) en de maximale weging (3.13) hieronder weer:

   Stel dat we de volgende k edits hebben: {1, 2, 3, 4} en we hebben de volgende verzamelingen voorgestelde edits B_m om geschonden te laten:

   $B_1 = \{4\}$, $B_2 = \{2, 3\}$ en $B_3 = \{1, 2\}$.

   We kunnen met de schendingsgewichten van de zachte edits {0,3; 0,7; 0,8; 0,9} berekenen wat de som wordt om de edits geschonden te laten.
   $D(B_1) = 0,9$
   $D(B_2) = 0,7 + 0,8 = 1,5$
   $D(B_3) = 0,3 + 0,7 = 1,0$

   Er wordt gekozen voor de minimale som, dus er zou in dit voorbeeld gekozen worden om de B_1 geschonden te laten, omdat deze de minimale som heeft. Echter kijken we nu niet meer naar de som van de schendingsgewichten, maar naar het maximum. Dus met de methode "de maximale weging" krijgen we:

   $D(B_1) = \max(0,9) = 0,9$
   $D(B_2) = \max(0,3; 0,7; 0,8) = 0,8$
   $D(B_3) = \max(0,3; 0,7) = 0,7$

Hier heeft B_3 het minste schendingsgewicht en wordt er gekozen om B_3 geschonden te laten. Dus bij deze weging kijken we naar de voorgestelde edits die geschonden mogen blijven en daarvan nemen we degene met het kleinste maximum. We kijken naar een minimax criterium. De reden dat we hebben gekozen om het maximale gewicht te nemen tussen de voorgestelde edits is omdat we op deze manier meer waarde hechten aan een hoger gewicht en juist een edit met een hoog gewicht niet geschonden willen laten. Bij het minimax-criterium willen we liever B_3 geschonden laten dan B_1, ook al bevat B_3 twee edits en B_1 maar één edit. Het gewicht van die ene edit in B_1(0,9) is namelijk hoger dan de gewichten in B_3(0.3 en 0.7). Als we daarentegen het somcriterium gebruiken, dan willen we liever B_1 geschonden laten dan B_3, want de som van de gewichten van de edits in B_3(1.0) is hoger dan het gewicht van de edit in B_1(0.9). Hoe hoger het gewicht, des te betrouwbaarder de edit is, dus die kunnen we dan beter niet geschonden laten.

2. De λ: we hebben gezien dat de minimale verzameling geschonden edits $D(B_m^*)$ uiteindelijk mee weegt in de totale kosten met betrouwbaarheidsgewichten, waarbij formule (3.5) minimaal is met een $\lambda = 0,5$. Echter gaan we hier de $\lambda \neq 0,5$ kiezen. De λ hebben we bij alle schendingsgewichten die we tot nu hebben besproken wel gelijk aan 0,5 gekozen. Bij de NIM (Nearest Neighbor Methodology) [7] (blz.145) wordt in een vergelijkbare formule de λ gelijk aan 0,75 of 0,90 gekozen, omdat dit in praktijk bij de Canadese volkstelling het beste resultaat heeft gegeven. Daarom stellen wij de λ bij het bepalen van de totale kosten voor een oplossing ook gelijk aan 0,75 en aan 0,90.

3. De λ en max: voor het bepalen van de totale kosten gaan we de maximale weging zoals bij 1. in deze paragraaf combineren met de lambda bij 2. in deze paragraaf. Dus we passen zowel de lambda aan van 0,5 naar 0,75 of 0,90 en we passen de som aan naar het maximum. Waarbij we dus onderstaande twee formules gebruiken, afgeleid uit (3.5) en (3.13)

$$\left\{ 0,75 \sum_{j=1}^{n} w_j y_j + (0,25) D(B_m^*) \right\}$$

of

$$\left\{ 0,90 \sum_{j=1}^{n} w_j y_j + (0,10) D(B_m^*) \right\}$$

z.d.d.

$$D(B_m) = \max_{k \in B_m} s_k.$$

4. De $\lambda < 0,5$: In de methoden hierboven hebben we $\lambda > 0,5$ uitgeprobeerd en zijn bij deze methode benieuwd wat er gebeurt als we voor de $\lambda < 0,5$ nemen. In dit geval wordt de nadruk gelegd op de $D(B_m^*)$ in plaats van op de kosten die veroorzaakt zijn door de schending van de harde edits. We gebruiken hierbij dus de formules (3.1) en (3.4) waarin we $\lambda = 0,3$ invullen:

$$\left\{ 0,3 \sum_{j=1}^{n} w_j y_j + (0,7) D(B_m^*) \right\}$$

z.d.d.

$$D(B_m) = \sum_{k \in B_m} s_k.$$

5. De $\lambda < 0,5$ en de max-methode: In deze methode nemen we de $\lambda = 0,3$ zoals in de methode hierboven. Dus wordt er ook in deze methode meer gelet op de kosten van de geschonden zachte edits dan op de kosten die zijn veroorzaakt door de schendingen van de harde edits. Een verschil met de vorige methode is dat we voor de doelfunctie het maximum gebruiken in plaats van de som zoals weergegeven in formule(3.13), namelijk:

$$D(B_m) = \max_{k \in B_m} s_k.$$

3.4 De Nearest Neighbormethode

In deze methode gaan we op zoek naar schendingsgewichten die worden bepaald aan de hand van de referentiedata. Hierbij wordt er bij elk ruw record een nearest-neighborrecord gezocht uit de referentiedata. Het zoeken naar een nearest-neighborrecord betekent dat we op zoek gaan naar een eerder gaafgemaakt record dat het meest lijkt op de oorspronkelijke ruwe record. Als het gevonden nearest-neighborrecord een zachte edit schendt, dan mag het ruwe record deze edit ook schenden. Deze geschonden edit krijgt dan voor dit record een laag schendingsgewicht. Als het nearest-neighborrecord een zachte edit niet schendt, dan mag het ruwe record deze edit ook minder snel schenden. Dus krijgt deze edit voor dit record een hoog schendingsgewicht. Een verschil met de tot nu toe behandelde methoden is dus dat de schendingsgewichten bij de nearest-neighbormethode niet gelijk zijn voor alle records. Het nearest-neighborrecord kan op een aantal manieren worden gevonden, namelijk:

1. Met behulp van de gegeven ruwe records gaan we per record na in de referentiedata wat een passende nearest-neighbor record is. Dit doen we stapgewijs:

   - De referentiedata zetten we naast de ruwe data in een werkblad.
   - We bepalen van elke variabele in de referentiedata (x_{ij}^{ref}) het gemiddelde ($\bar{x}_j$) en de steekproef standaarddeviatie (σ_j).
   - In een nieuwe werkmap bepalen we voor elke waarde het verschil tussen de oorspronkelijke referentiewaarde min de gemiddelde waarde en dat gedeeld door de standaarddeviatie. Dit noemen we het referentieverschil r_{ij}^{ref}. Dit doen we ook voor met de oorspronkelijke ruwe waarden min de gemiddelde waarde en dat gedeeld door de standaarddeviatie en dit noemen het ruweverschil r_{ij}^{ruw}.

   $$r_{ij}^{ref} = \frac{x_{ij}^{ref} - \bar{x}_j}{\sigma_j} \tag{3.14}$$

   $$r_{ij}^{ruw} = \frac{x_{ij}^{ruw} - \bar{x}_j}{\sigma_j} \tag{3.15}$$

   - Daarna nemen we van elke waarde het absolute verschil met het eerste ruwe record. Voor de ontbrekende waarden nemen we in de ruwe data voor het verschil nul.
   - Per record nemen we de som van de verschillen.
   - We zoeken het record erbij met de minimale som en dit is het nearest-neighbor record voor het **eerste** ruwe record. Die we hebben bepaald met behulp van formule (3.16)

   $$\arg\min_i \sum_{j=1}^{J} \left| r_{1j}^{ruw} - r_{ij}^{ref} \right| \tag{3.16}$$

   - Dit herhalen we voor de overige ruwe records ($i = 2, \ldots, T$) en zo kunnen we voor elk record een nearest-neighborrecord bepalen.

   We gaan kijken welke zachte edits er geschonden worden in het gevonden nearest-neighborrecord, daarbij onderscheiden we twee situaties zoals eerder aangegeven:

   (a) Als de zachte edit is geschonden in het nearest-neighborrecord, dan mag hij ook geschonden blijven in het ruwe record en krijgt de edit een laag schendingsgewicht

(b) Als de zachte edit niet is geschonden in het nearest-neighborrecord, dan mag hij ook niet geschonden blijven in het ruwe record en krijgt een hoog schendingsgewicht.

In bijlage E is de gebruikte R-code voor het vinden van de nearest-neighborrecord vermeld. Deze methode noemen we de NN-methode. In tabel 3.5 is er een voorbeeld opgenomen van een ruw record, het bijbehorende nearest-neighborrecord en het gave referentierecord voor de variabelen x_1 tot en met x_{12}. De NA waarden staan voor de ontbrekende waarden. Opmerkelijk aan dit voorbeeld is dat de variabele x_2 ontbreekt en dat er geen nul wordt ingevuld op de plaats van de NA zoals in het nearest-neighborrecord. Terwijl er voor de NA op de overige plaasten in het ruwe record wel een nul moet staan volgens de gekozen nearest-neighborrecord. In bijlage A zijn de harde en de zachte edits die horen bij deze dataset 'test-data' vermeld en daaruit kunnen we opmaken dat er een harde wordt geschonden, namelijk de harde edit: $x_2 - x_4 = 0$. Daardoor wordt edit x_2 aangepast naar 23010. Hierdoor is het de moeite waard om de nearest-neighborrecord te baseren op de ongeschonden variabelen en dit wordt besproken in methode 2.

Variabelen:	**ruwe record**	**NN-record**	**gave record**
x_1	23010	24740	0
x_2	NA	0	23010
x_3	23010	24740	23010
x_4	23010	0	23010
x_5	NA	0	0
x_6	NA	0	0
x_7	NA	0	0
x_8	NA	0	0
x_9	23010	24740	23010
x_{10}	23224	18532	23224
x_{11}	-214	6208	-214
x_{12}	31	32	33

Tabel 3.5: Voorbeeld ruw record met bijbehorende NN-record

2. In het foutlocalisatieproces zoeken we naar combinaties van variabelen die kunnen worden aangepast, zodanig dat er kan worden voldaan aan alle harde edits. In deze methode gaan we daarom eerst kijken welke variabelen er in de toegelaten oplossing worden aangepast en bij de onaangepaste variabelen gaan we een nearest-neighborrecord zoeken. Het verschil van deze methode met de vorige methode is dat we niet één nearest-neighborrecord hebben dat we steeds gebruiken, maar verschil-

lende nearest-neighborrecords. Bij de NN-methode hoort bij elk gaaf te maken record één nearest-neighborrecord dat steeds gebruikt wordt. Bij deze methode kunnen bij een gaaf te maken record in de loop van het foutlocalisatie-algoritme verschillende nearest-neighborrecords worden gekozen, afhankelijk van welke variabelen zijn aangewezen als fout. Een voordeel van deze methode is dat de nearest-neigborrecord wordt gekozen op slechts de variabelen die niet worden aangepast door de harde edits, waardoor we het nearest-neigborrecord niet baseren op foute waarden uit de ruwe data. Als er veel schendingen zijn van de harde edits, houden we niet veel variabelen over om onze nearest-neigborrecord op te baseren, dit is een nadeel van deze methode. In hoofdstuk 4 behandelen we de resultaten van deze methoden. Deze methode noemen we de FNN-methode.

Er zijn verschillende afstandmaten die men kan gebruiken om de nearest-neigbor aan te wijzen. Zo is een afstandmaat die we kunnen gebruiken voor het aanwijzen van de nearest-neighbor dat sommige variabelen bij voorkeur exact moeten overeenkomen met het gaaf te maken record. Dus we kiezen een aantal variabelen uit die hetzelfde moeten zijn voor de nearest-neighborrecord als voor het ruwe record. In de groothandeldata waren dat de volgende drie variabelen: GrootteKlasse, Rechtsvorm en de WND-ID. De grootteklasse zegt hoeveel mensen er werken in een groothandel en geeft dat in een grootteklasse weer. Zo is bijvoorbeeld een groothandel met werknemers tussen de 10 en de 20 een grootteklasse van vier. De rechtsvorm zegt wat voor bedrijf het is, dus of het een eenmanszaak is of een Naamloze Vennootschap. De WND-ID geeft weer in welke goederen de groothandel handelt. Door de WND-ID vast te kiezen voor de nearest neighbor kunnen we ervoor zorgen dat we een nearest-neighborrecord aanwijzen dat handelt in dezelfde goederen. Voor het geval dat er geen enkele referentierecord voldoet aan dezelfde grootteklasse om een nearest-neighborrecord te zijn, hebben we een stap ingebouwd, dat er in dat geval alleen wordt gekeken naar de rechtsvorm en naar de WND-ID. Als daar ook niet een nearest-neighborrecord in zit, dan wordt er alleen gekeken naar de WND-ID. Om het verschil weer te kunnen geven tussen het wel of niet kiezen van exact te overeenkomen variabelen, noemen dit de GNN-methode.

3.5 De chi-kwadraatmethode

We beschikken over een ruwe referentiedataset en een gave referentiedataset, dit betekent dat we voor de ruwe referentiedata na kunnen gaan welke waarden fout zijn. Per edit creëren we een twee-bij-twee tabel waarin we ‘de wel geschonden edit en de niet geschonden edit’ uitzetten tegen over ‘de waarden die voorkomen in de edit bevatten wel fouten en de waarden die

voorkomen in de edit bevatten geen fouten'. Dit lichten we toe met behulp van een voorbeeld:

Stel we hebben een edit $x_2 - 0,5x_3 \geq 0$ en bekijken voor deze edit hoeveel records er van de ruwe referentiedata hem schenden en vergelijken vervolgens de waarden van x_2 en x_3 in de ruwe data met de waarden uit de gave data om te kunnen zeggen welke waarden in de ruwe data fout zijn. Dit zetten we uit in een tabel waarbij we precies kunnen zien hoe vaak de edit geschonden wordt en hoeveel waarden die horen bij de x_2 en de x_3 nog fouten bevatten.

	Zit er een fout in de x_2 of x_3?		
	Ja	Nee	Totaal
Edit 1 is geschonden	31	56	87
Edit 1 is niet geschonden	330	311	641
Totaal	361	367	728

Tabel 3.6: Edit 1 in dataset 'test-data'.

Op de kruistabel passen we de χ^2-methode toe. De χ^2-methode die we in deze paragraaf bespreken is niet precies een χ^2-toets zoals we die kennen uit de statistiek. Echter bereken we wel de χ^2. De bijbehorende formule wordt gegeven in (3.17).

$$\chi^2 = \sum \frac{(f-e)^2}{e}. \tag{3.17}$$

De som loopt over alle cellen in het binnenwerk van de kruistabel, zoals weergegeven in tabel 3.6. De "f" staat voor de waargenomen waarden en de "e" staat voor de verwachte waarden, als we aannemen dat er geen samenhang is tussen enerzijds het geschonden zijn van een edit en anderzijds de aanwezigheid van fouten in de variabelen die voorkomen in een edit. Voor een betrouwbare zachte edit verwachten we dat die samenhang wel aanwezig is en verwachten we een hoge waarde van χ^2.

Met behulp van tabel 3.6 kunnen we de verwachte waarde "e" uitrekenen en deze berekening geven we weer in tabel 3.7.

	Ja	Nee
Edit 1 is geschonden	$\frac{87}{728} \times 361 = 43,1$	$\frac{87}{728} \times 367 = 43,9$
Edit 1 is niet geschonden	$\frac{641}{728} \times 361 = 317,9$	$\frac{641}{728} \times 367 = 323,1$

Tabel 3.7: De berekening van de verwachte waarde "e" in dataset 'test-data'.

Als de aanname van geen samenhang tussen het wel of niet geschonden zijn van edit 1 en het wel of niet voorkomen van fouten in x_2 en x_3 waar is, dan heeft χ^2 een χ^2-verdeling. Voor de χ^2-toets hebben we de vrijheidsgraden

nodig. De vrijheidsgraden kunnen bepaald worden met de formule (rijen-1)(kolommen-1). Dit voorbeeld heeft als vrijheidsgraden (2-1)(2-1)=1.

Het 95% kwantiel van de χ^2-verdeling met vrijheidsgraad één is ongeveer gelijk aan 3,84. Dit houdt in dat alles boven de 3,84 significant is. Dus we hebben voldoende reden om aan te nemen dat er wel samenhang is tussen het geschonden zijn van de edit en het wel of niet fout zijn van de x_2 en x_3. Tevens kunnen we met behulp van het uitgerekende χ^2 nagaan wat het kwantiel is van de χ^2-verdeling dat erbij hoort. De berekening van de chi-kwadraat kan dan met behulp van formule (3.17):

$$\chi^2 = \frac{(31-43,1)^2}{43,1}+\frac{(56-43,9)^2}{43,9}+\frac{(330-317,9)^2}{317,9}+\frac{(311-323,1)^2}{323,1} \approx 7,645$$

Het bijbehorende kwantiel van de chi-kwandraatverdeling met vrijheidsgraad één is afgerond 1. Naar aanleiding van bovenstaand voorbeeld kunnen we dit in het algemeen formuleren:

	Zit er een fout in de variabelen van de edit?		
	Ja	**Nee**	**Totaal**
De edit is geschonden	k	l	k+l
De edit is niet geschonden	m	n	m+n
Totaal	k+m	l+n	k+l+m+n

Tabel 3.8: Algemene weergave voor de berekening van de χ^2-methode

$$\begin{aligned}
e_k &= \left(\frac{k+l}{k+l+m+n}\right) \times (k+m) \\
e_l &= \left(\frac{k+l}{k+l+m+n}\right) \times (l+n) \\
e_m &= \left(\frac{m+n}{k+l+m+n}\right) \times (k+m) \\
e_n &= \left(\frac{m+n}{k+l+m+n}\right) \times (l+n) \\
\chi^2 &= \frac{(k-e_k)}{e_k} + \frac{(l-e_l)}{e_l} + \frac{(m-e_m)}{e_m} + \frac{(n-e_n)}{e_n}.
\end{aligned}$$

Op deze manier kunnen we voor elke edit de χ^2 en het bijbehorende kwantiel berekenen. De kwantielen kunnen gebruikt worden als schendingsgewicht s_k^M. Hoe groter de χ^2, des te dichter het schendingsgewicht bij 1 ligt.

Dit betekent een betrouwbaardere edit wegens samenhang tussen editschendingen en aanwezigheid van fouten.

In hoofdstuk vier bespreken we de resultaten van de toegepaste methoden. Bij deze resultaten zijn de betrouwbaarheidsgewichten of gelijk aan één ofwel gelijk aan de aangeleverd betrouwbaarheidsgewichten door de experts.

Hoofdstuk 4

De resultaten van de methoden met statische schendingsgewichten

In dit hoofdstuk gaan we in op de methoden die zijn besproken in hoofdstuk drie. In de eerste paragraaf laten we zien wat het gebruik van de zachte edits met de verschillende schendingsgewichten voor resultaten heeft gegeven. Hierbij passen we de verschillende schendingsgewichten uit paragraaf 3.1 toe op twee verschillende datasets: de test-dataset en de WP-blok-dataset. Beide datasets zijn van de productiestatistieken (PS) met data van groothandelaars. De edits die we hebben gebruikt voor deze datasets zijn opgenomen in bijlage A. Van elke dataset beschikken we over een gave dataset en een ruwe dataset. De gave dataset is de gaafgemaakte dataset door experts. De ruwe dataset is de oorspronkelijke dataset, de dataset die binnenkomt nadat de enquêtes zijn verwerkt en nog foutvrij gemaakt moet worden. Verder beschikken we ook over een ruwe en gave referentiedataset. Dit zijn de datasets van het jaar ervoor en deze gebruiken we voor het bepalen van de schendingsgewichten. Aan de hand van deze gegevens gaan we laten zien wat de resultaten zijn van de verschillende methoden.

De test-dataset bestaat uit twaalf numerieke variabelen van de groothandel dataset uit 2007, waarbij records van groothandels met 10 tot 100 werknemers in dienst zijn geselecteerd. Deze dataset is vervolgens gesplitst in tweeën. Zo hebben we een referentiedataset en een gave dataset elk met 728 records verkregen. Vervolgens zijn er in de eerste helft van de dataset random fouten en edits gecreëerd door Sander Scholtus, methodoloog bij het CBS. De orginele eerste helft van de dataset is gebruikt als gave data en de tweede helft van de dataset als referentiedata, omdat we voor deze dataset niet beschikken over een oudere dataset. De edits die horen bij deze dataset zijn aan de hand van de vragenlijst ook gekozen door Sander Scholtus. In de

'test-data' set zijn er geen ontbrekende waarden, alle waarden zijn ingevuld.

De dataset 'WP-blok' is een bestaande dataset uit het jaar 2007. De WP staat voor het aantal werkzame personen. In bijlage H is de vragenlijst van deze dataset opgenomen. Van deze dataset is een ruwe dataset en een gaafgemaakte dataset aanwezig. De referentiedata is de dataset van het jaar ervoor. De dataset 'WP-blok' bevat ook waarden die niet ingevuld zijn door de respondenten. Deze waarden zijn de ontbrekende waarden, die we de 'NA'-waarden noemen. Door de zachte edits mee te nemen hebben we geprobeerd om de ruwe data automatisch gaaf te maken met de verschillende schendingsgewichten uit paragraaf 3.1. Hierbij zullen we elke keer nagaan in hoeverre de verkregen resultaten overeenkomen met de gave dataset. In alle resultaten onderscheiden we daarom drie situaties:

1. 'hard': Alleen de harde edits, de zachte edits worden hier buiten beschouwing gelaten.
2. 'alles': De zachte edits als harde edits, waarbij dus alle edits (zowel harde als zachte edits) als hard worden beschouwd.
3. 'zacht': De zachte edits als zacht met schendingsgewichten afhankelijk van de methoden.

In de laatste paragraaf van hoofdstuk twee zijn er een drietal maten gegeven om het succes te meten van de gelocaliseerde fouten. Dat zijn de α, β en de γ ((2.6) t/m (2.8)), waarbij we streven naar lage waarden voor de α, β en de γ. Het percentage records waarin geen onterechte- en gemiste fouten in voorkomen, dus records waarin exact de juiste variabelen als fout worden aangewezen, noemen we δ. Voor 'hard' geven we de resultaten van beide datasets hieronder in de tabellen 4.1 en 4.2 weer. Deze hangen niet af van de keuze s_k, omdat de harde edits geen schendingsgewichten hebben. Tevens kijken we voor dataset 'WP-blok' niet naar de situatie waarbij alle edits als hard worden aangezen, omdat de zachte edits soms in tegenspraak met elkaar kunnen zijn. In dat geval is het onmogelijk om alle zachte edits als hard mee te nemen en het foutlocalisatie-algoritme geeft dan in sommige gevallen geen oplossing.

α	β	γ	δ
36,4	4,7	11,5	40,2

Tabel 4.1: De succesmaten voor dataset 'test-data' met alleen 'hard' in procenten.

α	β	γ	δ
55,1	0,3	13,1	58,4

Tabel 4.2: De succesmaten voor dataset 'WP-blok' met alleen 'hard' in procenten.

In de eerste paragraaf worden de resultaten besproken van de schendingsgewichten uit paragraaf 3.1. In de tweede paragraaf bespreken we de resultaten van de verkregen kwantieledits met de bijbehorende schendingsgewichten. In paragraaf drie worden de resultaten gegeven van de verschillende vormen van de doelfunctie. In de vierde paragraaf bespreken we de resultaten van de nearest neighbormethode. In de vijfde paragraaf geven we de resultaten van de chi-kwadraatmethode en in de laatste paragraaf wordt er een overzicht gegeven van de beste verkregen resultaten met de statische schendingsgewichten.

4.1 De resultaten van de statische schendingsgewichten

In paragraaf 3.1 hebben we gezien dat het schendingsgewichten s_k volgens verschillende methoden bepaald kunnen worden. In deze paragraaf laten we de resultaten zien van de benoemde methoden in paragraaf 3.1 in dezelfde volgorde.

A: De resultaten van methode A met de dataset 'test-data' worden gegeven in tabel 4.3. Uit deze resultaten kunnen we opmaken dat het gebruik van zachte edits een positieve bijdrage leveren, omdat we door middel van het gebruik van zachte edits als zacht meer fouten localiseren dan wanneer we alleen de harde edits meenemen en ook wanneer we de zachte edits als hard meenemen in onze foutlocalisatie. Tevens zien we dat zowel de α, β als de γ minder zijn dan wanneer we alleen harde edits gebruiken en ook minder zijn dan wanneer we alles als hard meenemen. Dit betekent dat we de foutlocalisatie kunnen verbeteren wanneer we de zachte edits meenemen als zacht met een schendingsgewicht s_k^A. Opmerkelijk is dat wanneer we alle edits als hard (alles) meenemen we een lagere percentage fouten vinden dan wanneer we alleen de harde edits (hard) meenemen. Dit komt doordat bij alles als hard de onterechte fouten erg hoog zijn en daardoor het percentage juist aangewezen fouten minder is.

De resultaten van methode met de dataset 'WP-blok' worden gegeven in tabel 4.4. Hieruit kunnen we opmaken dat de α lager is geworden, terwijl de β juist hoger is geworden. Dit betekent dat er meer

onterechte fouten worden aangewezen door het gebruik van de schendingsgewichten s_k^A. Toch geeft dit een lagere γ voor de zachte edits. Aan de hand van de δ kunnen we concluderen dat het meenemen van de zachte edits leidt tot een hogere foutlocalisatie.

	α		β		γ		δ	
	alles	zacht	alles	zacht	alles	zacht	alles	zacht
s_k^A	23,2	22,7	13,1	6,0	15,3	9,6	36,8	47,3
s_k^B	23,2	25,3	13,1	3,7	15,3	8,3	36,8	52,1
s_k^C	23,2	22,4	13,1	5,3	15,3	8,9	36,8	50,0
s_k^D-1 edit	25,1	25,7	9,4	7,4	12,8	11,3	42,9	46,0
s_k^D-3 edits	26,0	25,8	8,1	6,9	11,9	11,0	45,6	47,7
s_k^E	22,3	33,6	14,9	3,9	16,5	10,3	34,8	43,1
s_k^F	22,3	33,4	14,9	3,9	16,5	10,2	34,8	43,1
s_k^G	22,3	33,2	14,9	3,8	16,5	10,1	34,8	43,3
s_k^H	22,3	33,3	14,9	3,8	16,5	10,1	34,8	43,1
s_k^I	22,3	33,5	14,9	3,9	16,5	10,2	34,8	42,7

Tabel 4.3: De succesmaten voor dataset 'test-data' in procenten.

	α	β	γ	δ
s_k^A	43,5	1,4	12,1	63,4
s_k^B	50,8	0,4	12,5	60,9
s_k^C	43,2	0,9	11,6	64,5
s_k^E	50,2	0,6	12,5	60,9
s_k^F	50,6	0,5	12,6	60,9
s_k^G	52,4	0,3	12,6	60,7
s_k^H	44,4	0,7	11,7	64,5
s_k^I	49,6	0,5	12,4	61,4
s_k^J	43,5	1,1	12,3	62,9

Tabel 4.4: De succesmaten voor dataset 'WP-blok' in procenten voor 'zacht'.

Voor beide datasets kunnen we concluderen dat het meenemen van zachte edits als zacht kan bijdragen aan een verbeterde foutlocalisatie.

B: In bijlage B zijn de gebruikte schendingsgewichten s_k^B voor beide datasets per edit opgenomen. De resultaten van deze schendingsgewichten s_k^B met dataset 'test-data' worden gegeven in tabel 4.3. Uit deze tabel kunnen we opmaken dat het gebruik van zachte edits een positieve bijdrage levert, omdat we door middel van het gebruik van zachte edits als zacht met een schendingsgewicht s_k^B een hogere δ vinden. Tevens zien we dat alleen de α niet laag is voor de zachte edits en dat betekent dus dat we meer gemiste fouten hebben bij deze situatie dan bij de

situatie alles. Daarentegen zien we dat zowel de β als de γ het laagste zijn bij deze methode. We kunnen dus concluderen dat we voor deze dataset met schendingsgewichten s_k^B meer fouten localiseren dan de schendingsgewichten s_k^A.

Hetzelfde schendingsgewicht s_k^B gaan we nu toepassen op de dataset 'WP-blok'. In tabel 4.4 staan de resultaten van de schendingsgewichten s_k^B voor deze dataset. Uit deze tabel kunnen we opmaken dat we een hogere α, maar lagere β vinden bij deze schendingsgewichten. We hebben dus meer gemiste fouten en minder onterechte fouten. Hierdoor hebben we ook een lagere γ en δ. We kunnen dus concluderen dat de zachte edits als zacht positief bijdragen aan het localiseren van fouten, echter dat de schendingsgewichten s_k^B niet meer fouten kunnen localiseren dan de schendingsgewichten s_k^A.

Opmerkelijk is dat we bij de dataset 'WP-blok' met de schendingsgewichten s_k^B een lagere δ en een hogere γ vinden dan de schendingsgewichten s_k^A, terwijl dat bij dataset 'test-data' juist andersom is.

C: In bijlage B zijn de zachte edits die gebruikt zijn in beide datasets met de bijbehorende schendingsgewichten s_k^B en de verkregen schendingsgewichten s_k^C opgenomen. De resultaten van dataset 'test-data' staan in tabel 4.3. Het gemiddelde van de schendingsgewichten s_k^B in deze dataset geeft 0,94. Er is echter geen schendingsgewicht s_k^B dat gelijk is hieraan en daarom hebben we 0,95 als boven/ondergrens gekozen. Vervolgens hebben we het als volgt onderverdeeld:

$$\begin{aligned}
&\text{voor } s_k^B < 0,95 \text{ geven we een schendingsgewicht } s_k^C = 0,5\\
&\text{voor } s_k^B = 0,95 \text{ geven we een schendingsgewicht } s_k^C = 1,0\\
&\text{voor } s_k^B > 0,95 \text{ geven we een schendingsgewicht } s_k^C = 1,5
\end{aligned}$$

Uit tabel 4.3 kunnen we opmaken dat de α en de γ laag zijn in vergelijking met situatie 'hard' en 'alles'. De β is het laagst in de situatie alleen 'hard'. Dus we hebben meer onterechte fouten dan bij 'hard'. Echter kunnen we aan de δ zien dat we met 'zacht' meer fouten localiseren, meer dan de schendingsgewichten s_k^A, echter minder dan s_k^B. Dit betekent dat de schendingsgewichten s_k^C in de 'test-data' niet hebben geleid tot verbetering in het localiseren van fouten.

Hetzelfde schendingsgewicht s_k^C gaan we nu toepassen op de dataset 'WP-blok', waarvan de resultaten zijn opgenomen in tabel 4.4. We leggen eerst uit hoe we de schendingsgewichten voor deze dataset hebben gevonden. We hebben het gemiddelde van de schendingsgewichten s_k^B in de dataset 'WP-blok' berekend, dat geeft 0,976. Aan de schendingsgewichten s_k^B zien we dat zowel $s_k^B = 0,97$ voorkomt als

$s_k^B = 0,98$. Dus hebben we $s_k^B = 0,98$ als bovengrens en $s_k^B = 0,97$ als ondergrens gekozen. Vervolgens hebben we het als volgt onderverdeeld:

voor $s_k^B < 0,97$ geven we een schendingsgewicht $s_k^C = 0,5$
voor $0,97 \leq s_k^B \leq 0,98$ geven we een schendingsgewicht $s_k^C = 1,0$
voor $s_k^B > 0,98$ geven we een schendingsgewicht $s_k^C = 1,5$

In tabel 4.4 zien we weer dat de α bij zacht lager is, terwijl de β juist hoger is. Toch geeft dit een lagere γ en een hogere δ voor de zachte edits. Dus ook hier kunnen we concluderen dat het meenemen van de zachte edits als zacht positief bijdraagt aan het localiseren van meer fouten en dat de bijdrage van de schendingsgewichten s_k^C hoger zijn dan de schendingsgewichten s_k^A en s_k^B voor de dataset 'WP-blok'.

D: In de dataset 'test-data' verwijderen we de edit waarvan de s_k^B het laagst is om te kijken wat het effect is op foutlocalisatie. Vervolgens voeren we het nog eens uit maar dan verwijderen we drie edits met het laagste s_k^B. In bijlage B zijn de schendingsgewichten s_k^B opgenomen en daaruit blijkt dat edit $x_2 - 0.5x_3 \geq 0$ in 12% van de records wordt geschonden, dus we verwijderen eerst deze edit en het resultaat hiervan is opgenomen in tabel 4.3. Vervolgens verwijderen we de edits met schendingsgewichten $s_k^B < 0.95$, dat zijn de edits $x_5 + x_6 - x_7 \geq 0$ en $5000x_{12} - x_9 \geq 0$ en de overige edits geven we een schendingsgewicht s_k^B. Uit de tabel kunnen we dit zien bij de resultaten van s_k^D-3 edits. Tevens kunnen we opmaken dat zowel de α, β als de γ groter zijn geworden en aan de δ kunnen we zien dat we minder fouten localiseren, dus we kunnen concluderen dat dit geen goede methode is voor dataset 'test-data'.

De zachte edits in dataset 'test-data' zijn verzonnen edits en daardoor hebben we deze methode alleen toegepast op deze dataset.

E: De berekende schendingsgewichten s_k^E voor beide datasets zijn opgenomen in bijlage C. Uit tabel 4.3 kunnen we opmaken dat dit geen goede methode is voor de dataset 'test-data', omdat er heel weinig fouten worden gelocaliseerd. Dus gaan we ook kijken wat deze schendingsgewichten bij de dataset 'WP-blok' doen. De resultaten hiervan zijn opgenomen in tabel 4.4. De β is lager in vergelijking met methode C, waar we de meeste fouten hebben gelocaliseerd, maar de α is hoog en dat betekent dat we fouten mislopen. We kunnen dus concluderen dat deze methode in beide datasets niet de meeste fouten localiseren.

F: In bijlage C staan de berekende schendingsgewichten s_k^F voor beide datasets. Opvallend is dat deze zeer dicht bij de s_k^E liggen en dit komt doordat er bij s_k^E wordt gedeeld door het totaal aantal records, zonder

de ontbrekende waarden en bij het bepalen van de schendingsgewichten s_k^F wordt er gedeeld door alle goede waarden en deze liggen dus bij elkaar in de buurt. Daardoor zien we in beide datasets niet een groot verschil in de gevonden resultaten van deze schendingsgewichten. Dus we kunnen concluderen dat het verschil tussen de twee schendingsgewichten s_k^E en s_k^F minimaal is en aan de resultaten kunnen we zien dat er geen sprake is van verbetering.

G: In bijlage C zijn de schendingsgewichten s_k^G voor beide datasets opgenomen en deze verschillen behoorlijk van de schendingsgewichten s_k^E en s_k^F. In de tabellen 4.3 en 4.4 staan de resultaten. Hieruit kunnen we opmaken dat we een laag percentage aan onterechte fouten hebben. Dit is terug te zien aan de β. Voor deze schendingsgewichten en beide datasets kunnen we concluderen dat deze methode met schendingsgewichten s_k^G niet heeft bijgedragen aan een verbetering van de foutlocalisatie, omdat we voor dataset 'test-data' minder fouten localiseren dan met de schendingsgewichten s_k^B en voor dataset 'WP-blok' minder fouten localiseren dan met de schendingsgewichten s_k^C.

H: De schendingsgewichten s_k^H zijn verkregen door de schendingsgewichten van s_k^G te verdelen zoals bij de schendingsgewichten s_k^C en dat is gedaan door eerst het het gemiddelde van de schendingsgewichten te berekenen. Het gemiddelde in dataset 'WP-blok' is gelijk aan 0,75. Vervolgens hebben we de schendingsgewichten s_k^G voor deze dataset als volgt verdeeld:

$$\begin{aligned}
&\text{voor } s_k^G < 0,75 \text{ geven we een schendingsgewicht } s_k^H = 0,5\\
&\text{voor } s_k^G = 0,75 \text{ geven we een schendingsgewicht } s_k^H = 1,0\\
&\text{voor } s_k^G > 0,75 \text{ geven we een schendingsgewicht } s_k^H = 1,5
\end{aligned}$$

De schendingsgewichten s_k^H voor beide datasets zijn opgenomen in bijlage C. Op dezelfde manier hebben we de schendingsgewichten s_k^H ook voor dataset 'test-data' bepaald, echter zijn de verkregen resultaten slecht. Voor dataset 'WP-blok' geeft deze methode een vergelijkbaar resultaat als bij de schendingsgewichten s_k^C, echter zijn er in dit geval veel minder onterechte fouten en dat kunnen we zien aan de lagere β. De α is hoger dan de verkregen α uit s_k^C, maar doordat de β kleiner is, hebben we een vergelijkbare δ. We kunnen dus concluderen dat we met de schendingsgewichten s_k^H zoals bij s_k^C in dataset 'WP-blok' het hoogste percentage fouten localiseren. Voor dataset 'test-data' is dit niet een goede methode.

Bij de methoden E,F,G en H in dataset 'WP-blok' hebben we de ontbrekende waarden vervangen door nullen en bekeken of dit betere resultaten oplevert, maar dan vinden we een veel lagere δ, wat duidt

op een verslechtering van de methode. Het invullen van nullen op de plaatsen van de ontbrekende waarden heeft geen positief effect op de foutlocalisatie.

I: Met de schendingsgewichten s_k^I berekenen we de kans dat de ruwe data geschonden is, gegeven dat de gave data ook is geschonden. De resultaten van deze methode zijn opgenomen in de tabellen 4.3 en 4.4. Tot hetgeen wat we in paragraaf 3.1 beschreven, zijn de resultaten van deze methode voor beide dataset niet beter dan de resultaten van de vorige methoden uit paragraaf 3.1. Voor deze methode kunnen we dus concluderen dat het geen goede methode is voor foutlocalisatie.

J: De verschilmethode: de verschillen tussen de variabelen per edit met de bijbehorende schendingsgewichten s_k^J zijn opgenomen in bijlage D. Een verschil van nul betekent geen verschil tussen het gave- en het ruwe record en is dus een hoog schendingsgewicht, een verschil dat groter is dan 1, is redelijk onbetrouwbaar en krijgt een laag schendingsgewicht. We hebben deze methode enkel toegepast op dataset 'WP-blok', omdat de edits van deze dataset de daadwerkelijk gebruikte edits van de gaafmakers zijn. De schendingsgewichten voor deze dataset zijn als volgt bepaald:

$$
\begin{aligned}
&v_k = 0, \text{ geeft } s_k^J = 2,0 \\
&0,001 < v_k \leq 0,01, \text{ geeft } s_k^J = 1,5 \\
&0,01 < v_k \leq 1, \text{ geeft } s_k^J = 1,0 \\
&v_k > 1, \text{ geeft } s_k^J = 0,5
\end{aligned}
$$

De resultaten hiervan zijn opgenomen in tabel 4.4. Hieruit kunnen we concluderen dat deze methode een slechter resultaat geeft dan de schendingsgewichten s_k^A en geen goede methode is voor foutlocalisatie.

4.2 De resultaten van de kwantieledits

In deze paragraaf geven we de resultaten van de kwantieledits. Voor dataset 'test-data' zijn de resultaten weergegeven in tabel 4.5 en voor dataset 'WP-blok' in tabel 4.6.

I In tabel 4.5 zijn de resultaten van de 10%, de 5% en de 1% kwantieledits met als schendingsgewicht s_k^A gegeven voor dataset 'test-data'. Uit deze tabel kunnen we opmaken dat er minder fouten worden gevonden bij de 10% kwantieledit in vergelijking met de andere kwantieledits, omdat de δ bij deze methode het laagst is. Daarnaast zijn bij de

	α		β		γ		δ	
	alles	zacht	alles	zacht	alles	zacht	alles	zacht
$\hat{c}_{10}$-edits $+s_k^A$	22,4	21,7	17,6	7,4	18,7	10,5	29,1	43,1
$\hat{c}_5$-edits $+s_k^A$	22,3	21,0	14,9	6,3	16,5	9,5	34,8	47,9
$\hat{c}_1$-edits $+s_k^A$	22,1	20,7	12,8	5,6	14,7	8,8	38,5	50,8
II	19,2	18,9	18,4	11,5	18,6	13,1	28,7	35,4
III	19,2	24,7	18,4	3,2	18,6	7,8	28,7	54,4
IV	19,2	19,5	18,4	9,2	18,6	11,4	28,7	42,2
V	19,2	21,4	18,4	3,7	18,6	7,5	28,7	56,5
VI	19,2	21,5	18,4	4,2	18,6	7,9	28,7	55,9
$\hat{c}_1$-edits $+s_k^K$	22,1	26,1	12,8	3,6	14,7	8,4	38,5	52,9
$\hat{c}_5$-edits $+s_k^K$	22,3	25,8	14,9	3,5	16,5	8,3	34,8	52,7
$\hat{c}_5$-edits $+s_k^L$	22,3	23,0	14,9	4,8	16,5	8,7	34,8	49,7

Tabel 4.5: De succesmaten voor dataset 'test-data' met de kwantieledits in procenten. De kwantieledits geven we in paragraaf 3.2 weer met II,III,IV,V en VI. De $\hat{c}_{10}$, $\hat{c}_5$, $\hat{c}_1$ geven het 10%, 5% en 1% kwantieledit weer.

	α	β	γ	δ
$\hat{c}_{10}$-edits $+s_k^A$	42,8	1,8	12,3	62,2
$\hat{c}_5$-edits $+s_k^A$	41,5	2,4	12,6	62,4
$\hat{c}_1$-edits $+s_k^A$	45,1	0,5	11,6	65,3
III	48,1	0,5	12,1	63,4
IV	30,3	7,2	14,0	54,7
V	46,6	0,5	11,9	63,8
VI	46,7	0,5	11,9	63,8
$\hat{c}_1$-edits $+s_k^K$	54,1	0,2	12,8	59,7
$\hat{c}_5$-edits $+s_k^K$	54,2	0,3	12,9	59,5
$\hat{c}_5$-edits $+s_k^L$	48,1	1,2	12,7	60,2

Tabel 4.6: De succesmaten en juiste fouten voor dataset 'WP-blok' met 'zacht' en de kwantieledits in procenten. De kwantieledits geven we in paragraaf 3.2 weer met III,IV,V en VI. De $\hat{c}_{10}$, $\hat{c}_5$, $\hat{c}_1$ geven het 10%, 5% en 1% kwantieledit weer.

10% kwantieledits de α en de β groter dan bij de andere twee kwantieledits. Bij de 1% kwantieledits is de δ het grootst, met minder gemiste fouten en onterechte fouten, dus een kleinere α en β. Dus we kunnen concluderen dat we meer fouten localiseren bij de 1% kwantieledits dan bij de 10% kwantieledits. Voor deze dataset localiseren we met deze methode echter nog niet meer fouten dan de geleverde edits door de experts en schendingsgewicht s_k^B uit paragraaf 4.1.

De resultaten van dataset 'WP-blok' staan in tabel 4.6. Volgens δ geven de 1% kwantieledits het beste resultaat. Met de 5% kwantiel edits worden er minder fouten gemist, maar meer onterechte fouten gevonden. De α is in alle gevallen lager bij 'zacht' en de β is juist hoger bij 'zacht' in vergelijking met 'hard'. We kunnen concluderen dat we voor dataset 'WP-blok' met de 1% kwantieledits de meeste fouten localiseren, dit overtreft de schendingsgewichten s_k^C.

II Hierbij worden alle 3 de kwantieledits gebruikt (10%, 5% en 1%) en krijgt ieder kwantieledit een schendingsgewicht gelijk aan één. Deze methode hebben we enkel toegepast op dataset 'test-data', omdat het een erg slecht resultaat heeft gegeven voor deze dataset, hebben we het niet toegepast op dataset 'WP-blok'. Dus we kunnen opmaken dat dit niet een goede methode is voor foutlocalisatie.

III In tabel 4.5 staan de resultaten van deze methode met dataset 'test-data' waaruit we kunnen opmaken dat we een hoge percentage aan gemiste fouten hebben, een hoge α en juist een zeer laag percentage aan onterechte fouten, een lage β. Tevens kunnen we opmaken dat we een hoog percentage bij de δ hebben in vergelijking met de methoden uit paragraaf 4.1 en we kunnen concluderen dat deze methode voor deze dataset het beste resultaat geeft en een goede methode is voor foutlocalisatie.

In tabel 4.6 staan de resultaten van dataset 'WP-blok', waaruit we kunnen opmaken dat deze methode een redelijk hoge α en een gelijkwaardige β aan de resultaten van de 1% kwantieledits heeft. Dit is niet een slechte benadering, echter blijven we met de 1% kwantieledits met schendingsgewicht één de meeste fouten localiseren. Dus we kunnen concluderen dat dit een redelijke methode is, maar dat we met de 1% kwantieledits en de schendingsgewichten gelijk aan één meer fouten localiseren voor deze dataset.

IV De resultaten van dataset 'test-data' zijn opgenomen in tabel 4.5 en van dataset 'WP-blok' zijn opgenomen in tabel 4.6. Met deze methode hebben we in beide datasets aanzienlijk veel onterechte fouten, waardoor we uiteindelijk een lage δ hebben. We kunnen dus concluderen dat dit een slechte methode is voor foutlocalisatie voor beide dataset.

Echter is deze methode wel beter dan wanneer we de kwantieledits alle drie tegelijk meenemen en elk een schendingsgewicht één (II). Dus voor beide datasets is dit niet een geschikte methode voor foutlocalisatie.

V De kwantieledits met schendingsgewichten 0,9-0,05-0,05: De resultaten van de datatset 'test-data' zijn opgenomen in tabel 4.5 en hieruit kunnen we opmaken dat we wel fouten missen, doordat we een hoge percentage voor de α hebben, maar daarentegen hebben we een zeer kleine β en dus minder onterechte fouten. We kunnen voor deze methode concluderen dat het de meeste juiste fouten heeft gelocaliseerd tot nu toe voor dataset 'test-data'.

De resultaten voor dataset 'WP-blok' staan in tabel 4.6. Doordat we bij deze methode meer gemiste fouten hebben in vergelijking met de methode 1% kwantieledits met schendingsgewicht gelijk aan één, kunnen we concluderen dat deze methode voor dataset 'WP-blok' niet meer fouten localiseert dan de 1% kwantieledits met schendingsgewichten gelijk aan één. Voor beide datasets is dit een goede methode voor foutlocalisatie.

VI De kwantieledits met schendingsgewichten 0,9-0,05-0,10: Voor dataset 'test-data' hebben we meer gemiste en meer onterechte fouten dan in methode V, waardoor we in deze dataset een lagere δ hebben. We kunnen dus nog steeds concluderen dat methode V de meeste juiste fouten heeft gelocaliseerd voor dataset 'test-data'. In dataset 'WP-blok' geeft deze methode bijna dezelfde resultaten als in methode V. Voor dataset 'WP-blok' is dit samen met methode V, na de 1% kwantieledits met schendingsgewichten gelijk aan één, de methode waarmee we de meeste juiste fouten aanwijzen.

VII De resultaten voor de dataset 'test-data' zijn opgenomen in tabel 4.5, waaruit we kunnen opmaken dat de resultaten van de 1% kwantieledits + s_k^K redelijk overeenkomen met de 5% kwantieledits + s_k^K. Hierbij is s_k^K de vergelijking van de 1% kwantieledit met de 5% kwantieledit. Bij de 5% kwantieledits zijn er iets minder gemiste- en onterechte fouten. Echter zijn de resultaten niet beter dan methoden V en VI. Voor de dataset 'WP-blok' zijn de verkregen resultaten vergelijkbaar met de dataset 'test-data'.

VIII Deze methode geeft voor dataset 'test-data' meer onterechte fouten, dus een zeer hoge β. Waardoor de verkregen resultaten bij deze methode met de 5% kwantieledits en schendingsgewichten s_k^L minder zijn dan bij de schendingsgewichten s_k^K. Voor dataset 'WP-blok' is het juist het percentage gemiste fouten dat aanziendelijk daalt, waardoor deze methode een betere resultaat geeft dan de methode met de 5% kwantieledits + s_k^K.

4.3 De resultaten van de verschillende vormen van de doelfunctie

In deze paragraaf beschrijven we zoals we in paragraaf 3.3 hebben beschreven de aanpassingen van het 'maximum' en van de 'λ'. Voor dataset 'test-data' hebben we in deze paragraaf de beschikking over de oorspronkelijke ruwe dataset, waarin wel ontbrekende waarden voorkomen. Dit is dus de dataset zoals die is binnengekomen bij de gaafmakers. Dit is namelijk ook het geval voor dataset 'WP-blok' en daardoor kunnen we de verkregen resultaten van beide dataset beter met elkaar vergelijken. De resultaten van 'hard' voor deze methode in dataset 'test-data' geven we eerst hieronder weer in tabel 4.7, omdat deze voor de methoden die worden beschreven in deze paragraaf niet zullen veranderen.

α	β	γ	δ
31,3	1,1	8,9	58,7

Tabel 4.7: De succesmaten voor dataset 'test-data' met ontbrekende waarden en alleen 'hard' in procenten.

De methoden waarbij er wordt voldaan aan de formule hieronder, duiden we aan met 'MAX'. De resultaten van dataset 'test-data' staan in tabel 4.8 en de resultaten van dataset 'WP-blok' staan in tabel 4.9.

$$D(B_m) = \max_{k \in B_m} s_k.$$

1. Het maximum: Uit de resultaten van dataset 'test-data' kunnen we opmaken dat we een hoog percentage onterechte fouten vinden voor de 'MAX' met schendingsgewichten gelijk aan één. Echter is het opmerkelijk dat alleen de methoden met 'MAX' + s_k^B een hogere δ heeft dan de resultaten alleen 'hard' uit tabel 4.7. Dus we kunnen concluderen dat het gebruik van het maximum voor deze dataset geen positieve bijdrage levert aan het foutlocalisatieprobleem. Uit de resultaten van dataset 'WP-blok' kunnen we opmaken dat de verkregen resultaten voor deze dataset verslechterd zijn. We localiseren minder fouten die overeenkomen met de fouten in de gave dataset voor deze dataset. Voor de dataset 'WP-blok' geeft deze methode ook slechtere resultaten.

2. De λ: De resultaten van deze methoden worden in de resultaten aangeduid met $\lambda = 0,75$ of $\lambda = 0,90$. Hieruit kunnen we opmaken dat het aantal onterechte fouten, de β sterk verminderd is. Dit levert als resultaat voor de δ betere resultaten op als bij de oorspronkelijke resultaten

	α		β		γ		δ	
	alles	zacht	alles	zacht	alles	zacht	alles	zacht
$MAX + s_k^A$	13,2	28,1	10,4	2,2	12,8	9,3	42,0	52,1
$MAX + s_k^B$	13,2	30,7	10,4	0,9	12,8	8,6	42,0	60,0
$MAX + s_k^C$	13,2	28,8	10,4	1,8	12,8	9,0	42,0	55,8
$s_k^A + (\lambda = 0,75)$	13,2	29,8	10,4	0,9	12,8	8,4	42,0	60,6
$s_k^B + (\lambda = 0,75)$	13,2	30,4	10,4	0,8	12,8	8,5	42,0	60,7
$s_k^C + (\lambda = 0,75)$	13,2	29,8	10,4	0,9	12,8	8,4	42,0	60,7
$s_k^A + (\lambda = 0,90)$	13,2	30,3	10,4	0,8	12,8	8,4	42,0	60,7
$s_k^B + (\lambda = 0,90)$	13,2	30,4	10,4	0,8	12,8	8,5	42,0	60,7
$s_k^C + (\lambda = 0,90)$	13,2	30,4	10,4	0,8	12,8	8,5	42,0	60,7
$MAX + s_k^A + (\lambda = 0,90)$	13,2	30,7	10,4	0,9	12,8	8,6	42,0	60,0
$s_k^B + (\lambda = 0,30)$	13,2	18,0	10,4	7,5	12,8	11,7	42,0	43,1
$s_k^C + (\lambda = 0,30)$	13,2	20,9	10,4	6,4	12,8	11,5	42,0	44,1
$MAX + s_k^B + (\lambda = 0,30)$	13,2	23,1	10,4	4,2	12,8	10,0	42,0	48,5
$MAX + s_k^C + (\lambda = 0,30)$	13,2	24,2	10,4	3,8	12,8	10,0	42,0	48,6

Tabel 4.8: De resultaten van dataset 'test-data' met aanpassingen van de doelfunctie in procenten.

met een $\lambda = 0,50$. Voor de resultaten met $\lambda = 0,90$ is het opmerkelijk dat ze bijna allemaal op elkaar lijken. We krijgen zelfs het hoogste resultaat, hoogste δ.

De verklaring voor het feit dat de $\lambda = 0,90$ betere resultaten geeft, is doordat de zachte edits in dataset 'test-data' niet de zachte edits zijn van de gaafmakers.

Dit is het tegenovergestelde effect van wat we bij dataset 'WP-blok' in tabel 4.9 vinden. In dataset 'WP-blok' zijn de zachte edits de oorspronkelijk geleverde edits door de gaafmakers. In dataset 'WP-blok' localiseren we met $\lambda = 0,90$ voor schendingsgewichten s_k^A en s_k^C minder fouten dan voor $\lambda = 0,75$. Echter bij de 1% kwantieledits maakt het geen verschil of we de lambda aanpassen. Dus het aanpassen van de λ naar boven is geen goede methode voor foutlocalisatie voor de dataset 'WP-blok' en wel een goede methode voor de dataset 'test-data'.

3. De λ en het maximum: De resultaten van deze methode voor beide datasets zijn opgenomen in de tabellen 4.8 en 4.9 met $\lambda = 0,75$ en $\lambda = 0,90$.

   De resultaten van 'test-data' uit tabel 4.8 voor het maximum als doelfunctie en de schendingsgewichten s_k^A of s_k^B of s_k^C of de 1% kwantieledits met een $\lambda = 0,75$ en met een $\lambda = 0,90$ zijn allen aan elkaar gelijk. Dus in al deze methoden vinden we dezelfde resultaten, waardoor we deze resultaten niet allemaal opnemen in de tabel, maar

	α	β	γ	δ
$MAX + s_k^A$	51,0	0,6	12,7	61,2
$MAX + s_k^B$	48,7	0,5	12,2	63,3
$MAX + \hat{c}_1$-edits	48,9	0,5	12,3	63,3
$s_k^A + (\lambda = 0,75)$	54,8	0,3	13,0	58,6
$s_k^B + (\lambda = 0,75)$	54,2	0,2	12,9	59,5
$s_k^C + (\lambda = 0,75)$	54,9	0,2	13,0	58,8
$s_k^A + (\lambda = 0,90)$	55,2	0,3	13,1	58,4
$s_k^B + (\lambda = 0,90)$	55,0	0,3	13,0	58,6
$s_k^C + (\lambda = 0,90)$	54,9	0,2	13,0	58,8
$MAX + s_k^A + (\lambda = 0,75)$	55,1	0,3	13,1	58,4
$MAX + s_k^B + (\lambda = 0,75)$	55,0	0,3	13,0	58,6
$MAX + \hat{c}_1$-edits $+ (\lambda = 0,75)$	54,9	0,2	13,0	58,8
$MAX + s_k^A + (\lambda = 0,90)$	55,2	0,3	13,1	58,4
$MAX + s_k^B + (\lambda = 0,90)$	55,1	0,3	13,1	58,6
$MAX + \hat{c}_1$-edits $+ (\lambda = 0,90)$	54,9	0,2	13,0	58,8
$s_k^A + (\lambda = 0,30)$	30,3	5,7	12,9	55,3
$s_k^C + (\lambda = 0,30)$	29,8	4,3	11,7	60,2
$\hat{c}_1$-edits $+ (\lambda = 0,30)$	35,4	4,1	12,7	59,1
$MAX + s_k^A + (\lambda = 0,30)$	42,0	4,0	14,0	54,5
$MAX + s_k^C + (\lambda = 0,30)$	34,7	3,4	12,0	60,5
$MAX + \hat{c}_1$-edits $+(\lambda = 0,30)$	42,5	2,3	12,7	59,8

Tabel 4.9: De resultaten van dataset 'WP-blok' met 'zacht' en aanpassingen van de doelfunctie in procenten. Hierbij staat de $\hat{c}_1$-edits voor de 1% kwantieledits.

slechts één resultaat vermelden. Deze wordt in de tabel aangeduid met $MAX + s_k^A + \lambda = 0,90$. Voor de dataset 'test-data' localiseren we met deze methode iets minder fouten dan wanneer we voor de doelfunctie de som nemen en $\lambda > 0,50$. Echter geeft deze methode dezelfde resultaten als we eerder hebben gevonden met het maximum als doelfunctie + s_k^B. Dus voor deze dataset geeft het maximum geen verschil tussen een $\lambda = 0,75$ en $\lambda = 0,90$.

De resultaten voor dataset 'WP-blok' zijn niet allemaal aan gelijk en daardoor hebben we deze wel opgenomen in de tabel 4.9. Een $\lambda > 0,50$ voor deze dataset geeft slechtere resultaten dan de $\lambda = 0,50$. Voor deze dataset zijn de gevonden resultaten voor $\lambda = 0,75$ voor de schendingsgewichten gelijk aan dezelfde schendingsgewichten met een $\lambda = 0,90$. We kunnen voor deze dataset concluderen dat deze methode voor de dataset 'WP-blok' minder fouten localiseert. We hebben namelijk in de vorige paragraaf meer fouten gelocaliseerd dan we bij deze methode localiseren. Dus we kunnen voor beide datasets concluderen dat het gebruik van deze methode niet leidt tot een verbetering van de foutlocalisatie.

4. De λ kleiner dan 0,5:

   Voor de dataset 'test-data' geeft het gebruik van deze methode een verslechtering voor de foutlocalisatie. Er worden veel meer onterechte fouten aangewezen, waarbij we een hoge percentage voor de β hebben. We kunnen dus concluderen dat deze methode niet veel fouten localiseert voor deze dataset.

   Voor de dataset 'WP-blok' geeft het gebruik van deze methode ook slechtere resultaten dan de methoden die we eerder hebben gebruikt voor de foutlocalisatie. Dus het gebruik van een $\lambda < 0,5$ localiseert bij deze dataset minder fouten dan wanneer we de $\lambda = 0,5$ kiezen in de doelfunctie.

5. De λ kleiner dan 0,5 en het maximum: Bij deze methode is de $\lambda = 0,3$ zoals bij methode 4, echter is het verschil dat we hier voor de doelfunctie het maximum hanteren, zoals in formule (3.13). Deze methode wordt in de tabellen aangeduid met $MAX + \lambda = 0,30$.

   Voor de dataset 'test-data' hebben we een hogere δ dan wanneer we voor de doelfunctie de som nemen. Dit komt door het gebruik van het maximum waarmee we minder onterechte fouten localiseren. Dus voor de dataset 'test-data' is het gebruik van de maximum en een lage λ beter dan wanneer we alleen een lage λ gebruiken. Echter is deze methode niet zo goed in het localiseren van fouten als de methode waarbij we de $\lambda \geq 0,50$ kiezen. In tegenstelling tot de resultaten van 'test-data' zijn de resultaten veel slechter in dataset 'WP-blok'. Over

het algemeen leidt het maximum voor de doelfunctie in combinatie met een $\lambda < 0,50$ niet tot een verbetering in de foutlocalisatie.

4.4 De resultaten van de nearest neighbormethode

In deze paragraaf behandelen we de nearest neighbormethode, waarbij we twee situaties onderscheiden:

1. NN: We zoeken eerst een nearest neighborrecord en vervolgens passen we foutlocalisatie toe.

   Als laag schendinggewicht kiezen we verschillende waarden en als hoog schendingsgewicht kiezen we ook verschillende waarden. Deze verschillende hoge en lage schendingsgewichten zijn opgenomen in tabel 4.10 voor de dataset 'test-data' waarbij we de methode met de bijbehorende hoog- en laag gewicht aanduiden als: methode(hoog gewicht; laag gewicht). De R-code voor het bepalen van de NN is opgenomen in bijlage E.

	α		β		γ		δ	
	alles	zacht	alles	zacht	alles	zacht	alles	zacht
NN(5;0,5)	13,2	23,3	10,4	4,9	12,8	10,7	42,0	51,5
NN(3;0,03)	13,2	24,1	10,4	4,5	12,8	10,5	42,0	51,9
NN(1,5;0,25)	13,2	28,2	10,4	1,9	12,8	9,0	42,0	57,3
NN(1;0,1)	13,2	28,4	10,4	1,7	12,8	8,9	42,0	57,3
GNN(1,5;0,1)	13,2	28,2	10,4	1,9	12,8	9,0	42,0	57,3

Tabel 4.10: De resultaten van dataset 'test-data' met de nearest neighbormethode in procenten, waarbij (hoog gewicht; laag gewicht).

Uit tabel 4.10 kunnen we opmaken dat een hoog gewicht als 5 erg hoog is en niet zorgt voor betere resultaten dan wanneer we als hoog gewicht 1,5 nemen. Dit is te verklaren aan de hand van de betrouwbaarheidsgewichten die 1 zijn. De nearest neighbor methode hebben we ook toegepast op de dataset 'WP-blok' en de resultaten zijn opgenomen in tabel 4.11. Opmerkelijk is dat de resultaten van de nearest neighbormethode redelijk in de buurt liggen van de voorheen gevonden resultaten uit paragraaf 4.1. De λ en het maximum geven bij de dataset 'WP-blok' slechtere resultaten dan bij de 'test-data', zoals in paragraaf 4.3. Verder is het opmerkelijk dat er slechtere resultaten worden gevonden wanneer we een hoog gewicht kiezen dat niet in de buurt van de betrouwbaarheidsgewichten ligt. Dus als we bijvoorbeeld in plaats van een 2 een 1,25 kiezen als hoog gewicht, krijgen we betere resultaten, omdat de betrouwbaarheidsgewichten gelijk aan één zijn.

	α	β	γ	δ
NN(2;0,1)	35,1	2,4	11,3	63,1
NN(1,5;0,1)	43,2	0,8	11,6	64,3
NN(1;0,1)	45,5	0,8	11,9	63,6
NN(1,25;0,1)	44,3	0,7	11,7	64,5
$\hat{c}_1$-edits+NN(2;0,1)	35,1	2,6	11,5	63,6
$\hat{c}_1$-edits+NN(1,5,0,1)	43,5	0,9	11,7	64,7
$\hat{c}_1$-edits+NN(1,25;0,1)	44,6	0,8	11,8	64,8
MAX+NN(1,25;0,1)	48,9	0,3	12,2	62,8
$(\lambda = 0.75)$+NN(1,5;0,1)	54,7	0,2	13,0	58,8
$(\lambda = 0,90)$+NN(1.5;0.1)	55,1	0,3	13,1	58,6
$\hat{c}_1$-edits+s_k^C+NN(1,25;0,1)	43,5	0,8	11,6	64,7
FNN(1,25;0,1)	43,6	0,9	11,7	62,9
FNN(2;0,1)	34,7	2,6	11,4	61,6
$\hat{c}_1$-edit+FNN(1,25;0,1)	44,2	0,9	11,8	64,1
GNN(1,25;0,1)	39,1	1,6	11,4	63,3

Tabel 4.11: De resultaten van dataset 'WP-blok' met de nearest neighbormethode voor 'zacht' in procenten, waarbij (hoog gewicht; laag gewicht) en de $\hat{c}_1$-edits staan voor de 1% kwantieledits.

Wanneer we $\lambda = 0,90$ kiezen, worden er bijna evenveel fouten gevonden als wanneer we alleen de harde edits gebruiken. Dus de zachte edits hebben een belangrijke invloed bij het vinden van de juiste fouten bij de dataset 'WP-blok'.

2. FNN: Bij deze methode gaan we eerst kijken welke variabelen er door de harde edits worden aangepast en bij de onaangepaste variabelen gaan we een nearest neighbor record zoeken.

Uit tabel 4.11 kunnen we opmaken dat de resultaten zijn verslechterd door het gebruik van de FNN-methode ten opzichte van de resultaten van de NN-methode.

Zoals we paragraaf 4.4 hebben benoemd is de GNN naar aanleiding van een gesprek met de expert ontstaan waarbij we zelf extra edits toevoegen, door de verhoudingen tussen de edits te bepalen. Er zit een verband tussen:

1. de brutolonen en het aantal personen op de loonlijst
2. de kosten uitzendkrachten en het aantal uitzendkrachten
3. de kosten overige inleen en het aantal overig ingeleend personeel
4. de totale personeelskosten en het totaal aantal werkzame personen

5. de vergoedingen uitleen en het aantal uitgeleend personeel

Door in de referentiedataset te kijken naar het 5% kwantiel en 95% kwantiel tussen deze verbanden vinden we een aantal edits.

We bepalen voor de dataset 'WP-blok' een nearest neighborrecord door de volgende variabelen vast te kiezen:

1. GK
2. SBI
3. Rechtsvorm

Dit betekent dat deze waarden in de ruwe record gelijk moeten zijn aan de waarden van de nearest neighbor record. Uit de tabellen 4.10 en 4.11 zien we dat de toevoeging van de GNN geen grote invloed heeft op de foutlocalisatie.

4.5 De chi-kwadraatmethode

In deze paragraaf beschrijven we de resultaten van de methoden waarbij de χ^2 gebruiken om foutlocalisatie toe te passen. Voor dataset 'test-data' hebben we de dataset gebruikt, waarbij er geen ontbrekende waarden voorkomen. Dit is dezelfde dataset als in paragraaf 4.1 en 4.2.

	α		β		γ		δ	
	alles	zacht	alles	zacht	alles	zacht	alles	zacht
kwantiel χ^2	22,3	33,6	14,9	3,9	16,5	10,3	34,8	42,9

Tabel 4.12: De succesmaten van de dataset 'test-data' met de χ^2-methode in procenten.

	α	β	γ	δ
kwantiel χ^2	47,5	0,8	12,3	62,4

Tabel 4.13: De succesmaten van de dataset 'WP-blok' met de χ^2-methode voor 'zacht' in procenten.

De resultaten van de dataset 'test-data' zijn opgenomen in tabel 4.12. De berekende kwantielen zijn per edit opgenomen in bijlage F. We concluderen dat deze methode met de kwantielen van de χ^2 voor deze dataset minder fouten localiseert ten opzichte van de andere methoden. De resultaten van dataset 'WP-blok' zijn te vinden in tabel 4.13. Voor deze dataset zijn de berekende kwantielen ook opgenomen in bijlage F. Uit de resultaten van deze dataset kunnen we opmaken dat de χ^2 methode niet leidt tot het localiseren van meer fouten. Dus dit is geen goede methode voor foulocalisatie.

4.6 De beste resultaten met de statische schendingsgewichten

Naar aanleiding van de resultaten uit dit hoofdstuk kunnen we een top vijf samenstellen van de beste resultaten voor dataset 'test-data'. We baseren deze selectie op de δ, omdat we daaruit kunnen opmaken wat het percentage is dat we localiseren, zonder dat we de onterechte fouten en gemiste fouten meenemen en controleren op de fouten die overeenkomen met de fouten in de gave dataset. De top vijf voor dataset 'test-data' geven we weer in tabel 4.14.

	δ
$(\lambda = 0,90) + s_k^A$	60,7%
maximum + s_k^B	60,0%
Kwantieledits V	56,5%
Kwantieledits VI	55,9%
maximum + s_k^C	55,8%

Tabel 4.14: De top vijf van de schendingsgewichten voor foutlocalisatie in dataset 'test-data' voor δ in procenten.

Op dezelfde manier hebben we voor dataset 'WP-blok' ook een top vijf samengesteld met de beste resultaten. Dit wordt weergegeven in tabel 4.15.

	δ
1% kwantieledit met s_k^A	65,3
1%-NN(1,5;0,1)	64,8
NN(1,25;0,1)	64,5
s_k^H	64,5
Kwantieledits V	63,8

Tabel 4.15: De top vijf van de schendingsgewichten voor foutlocalisatie in dataset 'WP-blok' voor δ in procenten.

Uit deze tabellen kunnen we opmerken dat de verschillen erg klein zijn en dat de kwantieledits er het beste uitkomen voor de statische schendingsgewichten. De kwantieledits komen namelijk voor in beide datasets en geven in beide datasets één van de beste resultaten.

Hoofdstuk 5

Dynamische schendingsgewichten

In hoofdstuk drie hebben we methoden beschreven voor foutlocalisatie waarbij we de schendingsgewichten s_k vast kiezen voor de zachte edits. Deze schendingsgewichten zijn gekozen volgens één van de beschreven methoden en voor elk record in de dataset is het gekozen schendingsgewicht per edit hetzelfde (behalve bij de nearest-neighbormethode).

Stel we hebben een record met $x_1 = 5$ en een tweede record met $x_1 = 50$. Stel dat we een zachte edit: $x_1 \leq 4$ hebben. De behandelde schendingsgewichten in hoofdstuk drie geven een gelijk schendingsgewicht aan deze twee records. De grootte van de schending van een zachte edit speelt geen rol. Terwijl de edit door het eerste record wordt geschonden met één en door het tweede record met 46. De grootte van de schending van de edit noemen we de schendingsgrootte. Het eerste record heeft een kleinere schendingsgrootte dan het tweede record.

De inhoudelijke experts besteden bij de handmatige foutlocalisatie veel aandacht aan de grootte van de schendingen (een kleine schending is eenvoudiger te negeren dan een grote schending) en de vaste gewichten kunnen geen rekening houden met de grootte van de schendingen, waardoor we in dit hoofdstuk methoden beschrijven om deze schendingsgewichten automatisch te bepalen, gegeven de schendingen, waarbij elk schendingsgewicht afhangt van de grootte van de bijbehorende schending en van de mate waarin een edit belangrijk is. Als namelijk de schending van een zachte edit groter is, is de kans dat de opgegeven waarde in het record een fout bevat ook groter en daarom willen we de grootte van de schending betrekken in de schendingsgewichten. Hiermee bedoelen we dat de waarde van s_k zou moeten toenemen met de grootte van de schending van de k^{de} zachte edit. Tevens kan gedurende het doorlopen van de foutlocalisatie de schendingsgrootte veranderen, omdat de schending van de ene edit bijvoorbeeld wordt opgelost

ten koste van een andere edit. Een mooi voorbeeld hiervan wordt gegeven in het artikel van methodoloog Sander Scholtus paragraaf 7.1 [8].

In de eerste paragraaf van dit hoofdstuk worden er een aantal methoden beschreven die de schendingsgrootte van zachte edits meenemen bij het automatisch bepalen van de schendingsgewichten. In de tweede paragraaf bespreken we de Mahalanobis-afstand waarbij de afstand afhangt van de correlaties tussen de editschendingen in de dataset. In de derde paragraaf passen we logistische regressie toe om de kans te voorspellen dat een record fouten bevat. De resultaten van de beschreven methoden in dit hoofdstuk worden weergegeven in hoofdstuk 6.

5.1 De schendingsgrootte

Een schendingsgrootte gelijk aan nul betekent dat er geen edit wordt geschonden. Een schending ongelijk aan nul betekent daarentegen een schending van één of meerdere edits. De grootte van de schendingen noemen we ε_{ki}:

$$\varepsilon_{ki} = \begin{cases} 0 & \text{als record } i \text{ voldoet aan de } k^{de} \text{ zachte edit.} \\ > 0 & \text{anders.} \end{cases} \tag{5.1}$$

In hoofdstuk 1 hebben we de algemene vorm (1.1) van de lineaire edits gegeven, namelijk:

$$\sum_{j=1}^{n} a_{kj} x_j \geq b_k \quad \text{of} \quad \sum_{j=1}^{n} a_{kj} x_j = b_k.$$

De vorm van de edit is van groot belang bij het bepalen van de schendingsgrootte en daarom onderscheiden we twee situaties:

1. Voor de zachte edits in de vorm $\sum_{j=1}^{n} a_{kj} x_j \geq b_k$ kunnen we de ε_{ki} als volgt bepalen:

$$\varepsilon_{ki} = \max(0, -(a_{k1}x_1 + a_{k2}x_2 + \ldots + a_{kn}x_n - b_k)). \tag{5.2}$$

2. Voor zachte edits in de vorm $\sum_{j=1}^{n} a_{kj} x_j = b_k$ kunnen we de ε_{ki} als volgt bepalen:

$$\varepsilon_{ki} = |a_{k1}x_1 + a_{k2}x_2 + \ldots + a_{kn}x_n - b_k| \tag{5.3}$$

We noemen de grootte van de schendingen in de ruwe data ε_{ki}^{ruw}. De ε_{ki}^{ruw} zouden we als schendingsgewichten kunnen gebruiken, maar dat heeft vervelende eigenschappen:

- De ε_{ki}^{ruw} kan veel te groot zijn in vergelijking met de betrouwbaarheidsgewichten.

- De ε_{ki}^{ruw}wordt groter of juist kleiner als we een edit anders formuleren. Daarbij kunnen we denken aan de edit $3x - y \geq 10$ en de edit $6x - 2y \geq 20$ die een grotere ε_{ki}^{ruw} zal geven.

Daarom gaan we de ε_{ki}^{ruw} herschalen ofwel standaardiseren. In de referentiedata wordt voor elke edit k en alle records i met behulp van formule (5.2) of (5.3) de ε_{ki} berekend afhankelijk van de vorm van de edit. De standaarddeviatie van ε_{ki} in L, waarbij L de referentiedata van alle records met minimaal één geschonden edit is, noemen we σ_k:

$$\sigma_k = \sqrt{\frac{1}{\#L - 1} \sum_{i \in L} (\varepsilon_{ki} - \bar{\varepsilon}_k)^2}. \tag{5.4}$$

Hierbij is $\bar{\varepsilon}_k$ het gemiddelde van de schendingen van edit k over de records i, zoals gegeven in formule (5.5)

$$\bar{\varepsilon}_k = \frac{1}{\#L} \sum_{i \in L} \varepsilon_{ki}. \tag{5.5}$$

M: Een methode die we gebruiken om de schendingsgewichten dynamisch per edit (k) voor elk record (i) te bepalen is met behulp van formule (5.6).

$$s_{ki} = \frac{\varepsilon_{ki}^{ruw}}{\sigma_k}. \tag{5.6}$$

N: Een tweede methode om de schendingsgewichten dynamisch met behulp van de schendingsgrootte te bepalen is dat we in plaats van te delen door de standaarddeviatie, delen door het gemiddelde in de referentiedata. Deze schendingsgewichten noemen we s_{ki}^{β}. De schendingsgewichten s_{ki}^{β} worden berekend met behulp van formule (5.7).

$$s_{ki}^{\beta} = \frac{\varepsilon_{ki}^{ruw}}{\bar{\varepsilon}_k}. \tag{5.7}$$

De resultaten van deze twee methoden (5.6) en (5.7) worden beschreven in hoofdstuk 6.

5.2 De Mahalanobis-afstand

De Mahalanobis-afstand is een maat die gebruikt wordt om de grootte van de schendingen in de dataset te bepalen [8], [9]. Er wordt eerst gekeken of er sprake is van een schending en vervolgens wordt er gekeken naar de grootte van de schending ten opzichtte van de referentiedata.

De schendingen in record i noemen we ε_{ki}. De waarden ε_i worden gedefiniëerd als:

$$\varepsilon_i = (\varepsilon_{1i}, \varepsilon_{2i}, \ldots, \varepsilon_{Ki}). \tag{5.8}$$

De covariantiematrix van ε_i wordt bepaald op basis van de gave referentiedata met alleen de records die één of meer zachte edits schenden. De records die geen enkele edit schenden laten we buiten beschouwing. Dit doen we omdat we uiteindelijk vergelijken met de geschonden edits uit de referentiedata. De niet geschonden edits geven een nul als verschil in afstand en deze nullen beïnvloeden de standaarddeviatie en dus ook de covariantiematrix. De covariantiematrix noemen we S. De dynamische schendingsgewichten kunnen we bepalen met behulp van de schendingsgrootte per record. In het algemeen komt het er op neer dat de edits in elk record een schendingsgewicht krijgen dat groot is naarmate de Mahanalobis-afstand ook groot is. Voordat we in gaan op de toepassing van de schendingsgrootte geven we de algemene formule voor het bepalen van de Mahanalobis-afstand.

De Mahalanobis-afstand kan worden berekend met:

$$D_M(\varepsilon_i, 0) = \sqrt{(\varepsilon_i)' S^{-1} (\varepsilon_i)}. \tag{5.9}$$

De ideale situatie voor ons is als de schendingsgrootte gelijk is aan nul en bij de Mahalanobis-afstand wordt er geen schendingsgewicht per edit en per record bepaald, maar één gewicht en dat overeenkomt met de extra term $D(B)$. De $D(B)$ hebben we in het eerste hoofdstuk toegevoegd om rekening te houden met de zachte edits, namelijk: $D(B) = \sum_{k \in B} s_k$.

Hierbij is B de verzameling van geschonden zachte edits. De extra term in de doelfunctie D is de gewogen minimale som van de schendingsgewichten van alle zachte edits die op dat moment geschonden zijn. In deze methode nemen we niet de gewogen minimale som, maar gaan we de Mahalanobis-afstand gebruiken en nemen uiteindelijk de minimale afstand om de edit geschonden te laten. Dit lichten we met het voorbeeld toe, dat we eerder hebben gebruikt in hoofdstuk 3.3:

Stel dat we de volgende k edits hebben: {1, 2, 3, 4} en we hebben de volgende verzamelingen voorgestelde edits B_m om geschonden te laten:

$B_1 = \{4\}$, $B_2 = \{2,3\}$, $B_3 = \{1,2\}$ en covariantiematrix S is een matrix met 4 rijen en 4 kolommen.

We kunnen ε_i en de Mahalanobis-afstand $D_M(\varepsilon_i(B_m))$ berekenen, zoals we in paragraaf 3.3 de som van de schendingsgewichten namen om de edits geschonden te laten.

Voor $B_1 = \{4\}$:
Stel dat de berekende $\varepsilon_{4i} = 13$, geeft $\varepsilon_i = (0,0,0,13)$ en stel dat de $D_M(0,0,0,13) = 0,3$

Voor $B_2 = \{2,3\}$:
Stel dat de berekende $\varepsilon_{2i} = 5$ en $\varepsilon_{3i} = 6$, geeft $\varepsilon_i = (0,5,6,0)$ en stel dat de $D_M(0,5,6,0) = 0,5$

Voor $B_3 = \{1,2\}$:
Stel dat de berekende $\varepsilon_{1i} = 4$ en $\varepsilon_{2i} = 5$, geeft $\varepsilon_i = (4,5,0,0)$ en stel dat de $D_M(4,5,0,0) = 0,4$

Het uitkomst hiervan is één getal en we kiezen uiteindelijk voor de minimale (dat is in dit geval B_1), de $D_M(\varepsilon_i(B_m^*))$ en deze gebruiken we in de algemene formule uit hoofdstuk 1 (1.6):

$$\left\{ \lambda \sum_{j=1}^{n} w_j y_j + (1-\lambda) D_M(\varepsilon_i(B_m^*)) \right\}. \tag{5.10}$$

5.3 Logistische regressie

In deze paragraaf gaan we logistische regressie gebruiken bij het bepalen van de dynamische schendingsgewichten. Logistische regressie gebruiken wij als een techniek om te voorspellen wat de kans is dat een record fouten bevat.

In het algemeen hebben we een binaire doelvariabele $q \in \{0,1\}$. Met behulp van logistische regressie kunnen we $p = \Pr(q = 1)$ voorspellen met behulp van het model (5.11) met voorspellende variabelen $(t_1, t_2, ..., t_n)$ en storingsterm φ, is een stochastische variabele met verwachting nul.

$$\log\left(\frac{p}{1-p}\right) = b_0 + b_1 t_1 + b_2 t_2 + \ldots + b_n t_n + \varphi \tag{5.11}$$

Vervolgens fitten we het model met behulp van de referentiedata waarvoor $(t_1, t_2, ..., t_n)$ en q bekend is: dit geeft de parameterschattingen $(\hat{b}_0, \hat{b}_1, ..., \hat{b}_n)$.

Daarna berekenen we voor een record met $(t_1, t_2, ..., t_n)$ bekend en q onbekend:

$$\hat{p} = \hat{P}r(q = 1) = \left(\frac{1}{1 + e^{-\hat{\eta}}} \right) \tag{5.12}$$

z.d.d.

$$\hat{\eta} = \hat{b}_0 + \hat{b}_k t_1 + ... + \hat{b}_n t_n. \tag{5.13}$$

In ons geval gaan we dit als volgt toepassen: per record i gaan we voorspellen of het fouten bevat. De binaire doelvariabele wordt:

$$q_i = \begin{cases} 1 \text{ als record } i \text{ ten minste één fout bevat} \\ 0 \text{ anders.} \end{cases}$$

Vervolgens gaan we na of er zachte edits worden geschonden. De voorspellende variabele z_{ki} kunnen we definiëren als:

$$z_{ki} = \begin{cases} 1 \text{ als edit k geschonden is door record i} \\ 0 \text{ anders.} \end{cases} \tag{5.14}$$

Naar aanleiding hiervan hebben we drie vormen van logistische regressie geprobeerd, die we hieronder beschrijven in de modellen 1, 2 en 3. Op de plaats van de t_k in formule (5.11) vullen we in model 1 de s_{ki} uit formule (5.6) in, voor model 2 vullen we de z_k in uit formule (5.14) en voor model 3 vullen we zowel de z_k in uit formule (5.14) als de Mahalanobis-afstand uit (5.9) in.

In model 1 hanteren we de gestandaardiseerde schendingsgrootte als voorspellende variabelen.

$$\log \left(\frac{p_i}{1 - p_i} \right) = b_0 + b_1 s_{1i} + ... + b_K s_{Ki} + \varphi \tag{5.15}$$

In model 2 hanteren we het wel of niet geschonden zijn van de edit als voorspellende variabelen.

$$\log \left(\frac{p_i}{1 - p_i} \right) = b_0 + b_1 z_{1i} + ... + b_K z_{Ki} + \varphi \tag{5.16}$$

In model 3 hanteren we het wel of niet geschonden zijn van de edit als voorspellende variabele met een extra variabele, namelijk de Mahalanobis-afstand van de schendingen.

$$\log \left(\frac{p_i}{1 - p_i} \right) = b_0 + b_1 z_{1i} + ... + b_K z_{Ki} + b_{DM} D_M(\varepsilon_i, 0) + \varphi \tag{5.17}$$

Ter illustratie, gegeven model 1, krijgen de formules (5.12) en (5.13) de volgende vorm:

$$\hat{p}_i = \left(\frac{1}{1 + e^{-\eta_i}} \right)$$

$$\eta_i = \hat{b}_0 + \sum_{k=1}^{K} (\hat{b}_k s_{ki})$$

We schatten met behulp van de referentiedata $(\hat{b}_0, \ldots, \hat{b}_n)$. Vervolgens gebruiken we bovenstaande om $\hat{p}$ te berekenen. Gegeven de toegelaten oplossing van het foutlocalisatieprobleem is $\hat{p}^*$ net zoals in het voorbeeld in paragraaf 5.2 de minimale $\hat{p}$, die we vervolgens gebruiken om de totale kosten te berekenen:

$$\left\{ \lambda \sum_{j=1}^{n} w_j y_j + (1 - \lambda) \hat{p}^* \right\}.$$

De resultaten van de methoden die we in dit hoofdstuk hebben beschreven zullen we weergeven in hoofdstuk 6 en tevens zullen we aan de hand van de verkregen resultaten concluderen of deze tweede variant leidt tot duidelijk betere resultaten.

Hoofdstuk 6

De resultaten van de methoden met dynamische schendingsgewichten

In dit hoofdstuk geven we de resultaten weer van de dynamische schendingsgewichten die we hebben besproken in hoofdstuk 5. De methoden zijn alleen toegepast op de dataset: 'test-data', waarbij de dataset geen ontbrekende waarden bevat. Dit hebben we gedaan gezien de beperkte tijd en omdat de edits in dataset 'WP-blok' gecompliceerder zijn dan in dataset 'test-data'. De methoden uit hoofdstuk 5 kunnen in principe ook toegepast worden op dataset 'Wp-blok'.

6.1 De resultaten van de schendingsgrootte

De schendingsgewichten s_{ki} zijn afhankelijk van edit k en record i. De resultaten van de s_{ki} met de dataset 'test-data' staan in tabel 6.1.

	α		β		γ		δ	
	alles	zacht	alles	zacht	alles	zacht	alles	zacht
M	23,2	27,3	13,1	5,0	15,3	9,8	36,8	49,2
M, eerste oplossing	23,2	21,2	13,1	4,7	15,3	8,2	36,8	56,2
N	23,2	24,7	13,1	8,7	15,3	12,1	36,8	43,0

Tabel 6.1: De succesmaten van de s_{ki} in procenten.

Uit deze tabel kunnen we opmaken dat we een lagere δ vinden in vergelijking met de resultaten van de methoden met statische schendingsgewichten uit hoofdstuk 3. In het algoritme dat we gebruiken, wordt er voor de foutlocalisatie bij meerdere gelijkwaardige oplossingen gekozen voor de laatste

gevonden oplossing. Daarom is het interessant om te kijken wat de resultaten van de andere oplossingen worden. Dit hebben we ontdekt doordat het algoritme in een bepaalde dataset vaker kiest om de variabele x_{10} aan te passen, terwijl er in de gave data wordt gekozen om vaker de x_{11} aan te passen. De x_{10} komt alleen voor in de harde edits en de x_{11} komt zowel voor in de harde- als in de zachte edits. Daardoor zijn we gaan kijken naar de mogelijk gevonden oplossingen met dezelfde totale kosten. In geval van meerdere oplossingen met dezelfde totale kosten, kiest het algoritme de laatste oplossing. We hebben bij een aantal records gekeken naar de mogelijke oplossingen en dit verwerkt in tabel 6.2. Hierin geven we een voorbeeld van de variabelen één tot en met twaalf. Het overeenkomstige record met het gave record zit er tussen, maar er is niet eenduidig te zeggen welke dat is. De totale kosten voor de vier oplossingen in tabel 6.2 zijn even groot.

Variabelen	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
Oplossing 1	0	0	0	0	0	0	0	1	0	0	1	1
Oplossing 2	0	0	0	0	0	0	0	1	1	1	0	1
Oplossing 3	0	0	0	0	1	0	0	0	0	0	1	1
Oplossing 4	0	0	0	0	1	0	0	0	0	1	0	1
Oplossing Gaaf	0	0	0	0	1	0	0	0	0	0	1	1

Tabel 6.2: Verschillende oplossingen voor record i uit dataset 'test-data' met s_{ki}.

In dit voorbeeld komt de derde oplossing overeen met de gave oplossing, maar dit kan voor een ander record anders zijn.

Opmerkelijk is dat het kiezen van de eerste oplossing in de dataset 'test-data' een beter resultaat geeft. Het feit dat het algoritme de laatste oplossing neemt, hoeft dus niet te betekenen dat het ook de best passende oplossing is. Daardoor hebben we de foutlocalisatie zodanig aangepast dat het niet de laatste oplossing geeft in geval van meerdere gelijkwaardige oplossingen, maar de eerste oplossing geeft.

Hieruit kunnen we concluderen dat dit veel meer fouten localiseert dan de laatste oplossing. Dit betekent dat een verder onderzoek naar de keuze van de gelijkwaardige oplossingen één en ander kan verbeteren aan de foutlocalisatie.

6.2 De Mahalanobis-afstand

Het uitvoeren van de Mahalanobis-afstand duurde erg lang, omdat het algoritme veel vastloopt doordat hij geen oplossing kan vinden voor het foutlocalisatieprobleem. Dit komt doordat het algoritme dan moeite heeft met het bepalen van de $D_M(\varepsilon_i(B_m^*))$ uit formule (5.10) [8].

De Mahalanobis-afstand die we hebben toegepast op de dataset 'test-data' localiseert niet meer fouten dan het percentage gevonden fouten met de methoden uit hoofdstuk 3. Tevens kunnen we in tabel 6.3 opmaken dat we een redelijk hoge α, β en γ hebben en een hele lage δ. Deze methode localiseert daarentegen wel meer fouten dan de methode in paragraaf 5.1 s^{β}_{ki}, maar minder dan s^{α}_{ki}. Dus we kunnen bij deze methode vermelden dat het geen goede methode is voor foutlocalisatie.

α		β		γ		δ	
alles	zacht	alles	zacht	alles	zacht	alles	zacht
23,2	32,8	13,1	4,9	15,3	10,9	36,8	46,8

Tabel 6.3: De succesmaten van de Mahalanobis-afstand in procenten.

6.3 Logistische regressie

Het resultaat van logistische regressie met de modellen 1,2 en 3 zijn opgenomen in tabel 6.4. Hieruit kunnen we opmaken dat de δ in dataset 'test-data' een laag percentage voor foutlocalisatie heeft en daarom voeren we dezelfde methode opnieuw uit, waarbij we λ gelijk aan 0,4 en 0,3 en ook 0,2 kiezen. De keuze van de λ heeft te maken met de gevonden resultaten uit hoofdstuk (4.3) waar een lage λ de nadruk legt op de zachte edits. We kunnen aan tabel 6.4 zien dat dit een veel beter resultaat geeft voor de foutlocalisatie.

	α		β		γ		δ	
	alles	zacht	alles	zacht	alles	zacht	alles	zacht
Model 1	23,2	30,7	13,1	3,4	15,3	9,2	36,8	45,9
Model 1, $\lambda = 0,3$	23,2	22,8	13,1	3,7	15,3	7,8	36,8	54,3
Model 1, $\lambda = 0,2$	23,2	22,2	13,1	5,4	15,3	9,0	36,8	50,0
Model 1, $\lambda = 0,4$	23,2	25,3	13,1	3,4	15,3	8,1	36,8	53,3
Model 2	23,2	31,9	13,1	3,5	15,3	9,5	36,8	43,4
Model 2, $\lambda = 0,3$	23,2	24,6	13,1	4,4	15,3	8,7	36,8	51,1
Model 3	23,2	34,4	13,1	4,0	15,3	10,5	36,8	42,6
Model 3, $\lambda = 0,3$	23,2	27,9	13,1	4,1	15,3	9,2	36,8	51,1

Tabel 6.4: De succesmaten van logistische regressie in procenten.

Bij model 2 vinden een lagere δ dan bij het eerste model van logistische regressie waarbij we de schendingsgrootte meenemen. Dus als we slechts zeggen of het wel/niet geschonden is, zoals in model 2 vinden we minder fouten.

De resultaten van het derde model met logistische regressie is weergegeven in tabel 6.4, waaruit we kunnen opmaken dat er met dit model ook minder

fouten worden gelocaliseerd dan het eerste model. Dit kunnen we zien aan de lagere δ. Het verlagen van de λ heeft bij alle modellen een beter resultaat dan wanneer we $\lambda = 0,50$ nemen.

We kunnen dus concluderen dat de gevonden resultaten in dit hoofdstuk niet meer fouten localiseren dan de gevonden resultaten in hoofdstuk 4.

In hoofdstuk 7 zullen we de betrouwbaarheidsgewichten die we hebben gekregen van de gaakmakers gebruiken en de geselecteerde beste methoden uit paragraaf 4.6 opnieuw uitvoeren met de betrouwbaarheidsgewichten, waarbij we de schendingsgewichten met een aantal methoden aanpassen aan de betrouwbaarheidsgewichten. Gezien de beperkte tijd was het niet mogelijk om alle methoden opnieuw uit te voeren met de betrouwbaarheidsgewichten van de gaafmakers. In het laatste hoofdstuk zijn de conclusie en aanbevelingen te vinden.

Hoofdstuk 7

De betrouwbaarheidsgewichten

In de voorgaande hoofdstukken hebben we de betrouwbaarheidsgewichten steeds gelijk aan één gekozen. Dit hebben we gedaan, omdat we de invloed van de betrouwbaarheidsgewichten van de variabelen wilden beperken tot een minimum en zo de bijdrage van de schendingsgewichten beter zichtbaar konden maken. Aangezien we nu een aantal methoden hebben gevonden die toepasbaar zijn voor de foutlocalisatie met zachte edits, kunnen we de betrouwbaarheidsgewichten van de gaafmakers uitproberen op onze datasets in combinatie met de methoden om de schendingsgewichten te bepalen. In praktijk worden namelijk de betrouwbaarheidsgewichten gebruikt bij de foutlocalisatie en daarom zullen wij in dit hoofdstuk de foutlocalisatie toepassen met de betrouwbaarheidsgewichten van de gaafmakers.

7.1 De statische schendingsgewichten met de betrouwbaarheidsgewichten

In hoofdstuk 4 zijn de resultaten gegeven van beide datasets: 'test-data' en 'WP-blok' waarin de betrouwbaarheidsgewichten gelijk aan één waren gekozen. In dit hoofdstuk gebruiken we de dataset 'WP-blok' omdat deze dataset werkelijke fouten bevat die door groothandelaars zijn gemaakt. In dataset 'WP-Blok' hebben we de volgende variabelen met betrouwbaarheidsgewichten:

Uit tabel 7.1 blijkt dat de variabele x_1 een groot betrouwbaarheidsgewicht heeft in vergelijking met bijvoorbeeld x_4. Dit betekent dat de variabele x_1 betrouwbaarder is dan de andere variabelen in deze dataset. Indien een harde edit wordt geschonden door deze variabele met een betrouwbaarheidsgewicht 5, zal deze variabele niet snel worden aangepast. Het algoritme

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
w_j	5	2	2	1	2	2	3	2	3	3

Tabel 7.1: Betrouwbaarheidsgewichten bij de variabelen van dataset 'WP-blok'.

werkt namelijk volgens het gegeneraliseerde principe van "Fellegi en Holt" [1]. Dit doen we met formule (2.2) en daardoor zal een variabele met een laag betrouwbaarheidsgewicht eerder aangepast worden dan een variabele met een hoog betrouwbaarheidsgewicht. Dit is in paragraaf 2.4 voor de schendingsgewichten toegelicht met een voorbeeld en dit wordt op dezelfde manier toegepast voor de betrouwbaarheidsgewichten.

De schendingsgewichten die wij hebben gebruikt in hoofdstuk 3 zijn naar verhouding veel kleiner dan de betrouwbaarheidsgewichten in tabel 7.1 en daarom passen we de schendingsgewichten aan. Dit doen we in vier varianten:

1. We nemen het gemiddelde van de betrouwbaarheidsgewichten van de variabelen die voorkomen in een edit en vermenigvuldigen de schendingsgewichten met dit gemiddelde. Deze variant noemen we "gem".

2. We nemen het gewogen gemiddelde van de betrouwbaarheidsgewichten van de variabelen die voorkomen in een edit en vermenigvuldigen de schendingsgewichten met dit gewogen gemiddelde. Deze variant noemen we "gew". Hierbij gebruiken we de reciproke waarden van de betrouwbaarheidsgewichten als wegingsfactoren.

3. We nemen de mediaan van de betrouwbaarheidsgewichten van de variabelen die voorkomen in een edit en vermenigvuldigen de schendingsgewichten met deze mediaan. Deze variant noemen we "med".

4. We nemen het maximum van de betrouwbaarheidsgewichten van de variabelen die voorkomen in een edit en vermenigvuldigen de schendingsgewichten met dit maximum. Deze variant noemen we "max".

Om de vier varianten beter toe te lichten geven we hieronder een voorbeeld van elke variant en voor de schendingsgewichten s_k kunnen de methoden uit hoofdstuk 3 worden gebruikt.

Stel we hebben een edit met betrouwbaarheidsgewichten $w_1 = 5$, $w_2 = 2$ en $w_3 = 4$.

1. Het gemiddelde van deze betrouwbaarheidsgewichten is $\frac{11}{3}$ en dit geeft voor zachte edit k een schendingsgewicht van $\frac{11}{3} \times s_k$.

2. Het gewogen gemiddelde berekenen we als volgt:

$$\frac{1}{5}+\frac{1}{2}+\frac{1}{4}=\frac{19}{20}$$

$$5\times\frac{\frac{1}{5}}{\frac{19}{20}}+2\times\frac{\frac{1}{2}}{\frac{19}{20}}+4\times\frac{\frac{1}{4}}{\frac{19}{20}}=3\times\frac{1}{\frac{19}{20}}=3\times\frac{20}{19}=\frac{60}{19}=3\frac{3}{19}$$

   Dit geeft voor zachte edit k een schendingsgewicht van $3\frac{3}{19}\times s_k$.

3. De mediaan van de betrouwbaarheidsgewichten (5,2,4) is 4 en dit geeft voor zachte edit k een schendingsgewicht van $4\times s_k$.

4. Het maximum van de betrouwbaarheidsgewichten (5,2,4) is en dit geeft voor zachte edit k een schendingsgewicht van $5\times s_k$.

Deze 4 varianten gaan we toepassen op de volgende methoden uit hoofdstuk 3 met dataset 'WP-Blok': s_k^A, s_k^C, 1% kwantieledits, V=0,9-0,05-0,05, VI=0,9-0,05-0,10, s_k^H en de 1% kwantieledits met s_k^C. In hoofdstuk 4 zijn de resultaten van deze methoden met de betrouwbaarheidsgewichten gelijk aan één te vinden ($w_j = 1$). De betrouwbaarheidsgewichten $w_j \in W$, waarin W = (5,2,2,1,2,2,3,2,3,3) betekent dat we de betrouwbaarheidsgewichten uit tabel 7.1 hanteren. In bijlage G en tabel G.1 zijn de succesmaten met $w_j \in W$ in combinatie met de statische schendingsgewichten uit paragraaf 4.1 te vinden. Hieruit kunnen we opmaken dat we door het gebruik van de originele betrouwbaarheidsgewichten in combinatie met de schendingsgewichten die bepaald zijn in paragraaf 3.1, zorgen voor een vermindering van de δ. Daardoor passen we de vier varianten toe op deze zelfde methoden.

In tabel 7.2 staan de resultaten van alleen de harde edits met de betrouwbaarheidsgewichten voor dataset 'WP-blok'. In alle overige tabellen nemen we dit dus niet meer op en tonen de resultaten waarbij we zachte edits als 'zacht' meenemen.

α	β	γ	δ
55,7	0,5	13,3	57,8

Tabel 7.2: De resultaten van alleen hard in 'WP-blok' in procenten met $w_j \in W$.

In tabel 7.3 staan de resultaten van het gemiddelde. In tabel 7.4 staan de resultaten van het gewogen gemiddelde. In tabel 7.5 staan de resultaten van de mediaan en in tabel 7.6 staan de resultaten van het maximum. Uit al deze tabellen kunnen we opmaken dat ze een beter resultaat geven als we de schendingsgewichten schalen naar de betrouwbaarheidsgewichten. De resultaten die hieronder staan zijn immers beter dan de resultaten uit bijlage

G tabel G.1. Tevens is het zo dat we met alleen 'hard' en betrouwbaarheidsgewichten gelijk aan één meer fouten localiseren dan wanneer we de betrouwbaarheidsgewichten van de gaafmakers meenemen. De reden voor een lagere δ kan komen doordat we de methoden hebben geselecteerd op basis van de betrouwbaarheidsgewichten die gelijk zijn aan één.

	α	β	γ	δ
s_k^A	45,0	1,1	12,1	63,3
s_k^C	43,4	1,0	11,8	64,3
$\hat{c_1}$-edits	46,2	0,7	12,0	64,1
V	46,4	0,6	11,9	63,8
VI	46,4	0,6	11,9	63,8
$s_k H$	45,0	0,8	11,8	64,5
$\hat{c_1}$-edits s_k^C	43,9	1,0	11,8	64,1

Tabel 7.3: De resultaten van het gemiddelde in 'WP-blok' in procenten met $w_j \in W$. Hierbij zijn de $\hat{c}_1$-edits de 1% kwantieledits en de V, VI de methoden uit paragraaf 3.2.

	α	β	γ	δ
s_k^A	44,9	1,0	12,0	63,3
s_k^C	43,5	1,0	11,7	64,5
$\hat{c_1}$-edits	46,8	0,7	12,1	64,1
V	46,4	0,6	11,9	63,8
VI	46,4	0,6	11,9	63,8
s_k^H	44,9	0,8	11,8	64,5
$\hat{c_1}$-edits s_k^C	45,5	0,9	11,8	64,1

Tabel 7.4: De resultaten van het gewogen gemiddelde in 'WP-blok' in procenten met $w_j \in W$. Hierbij zijn de $\hat{c}_1$-edits de 1% kwantieledits en de V, VI de methoden uit paragraaf 3.2.

Indien we het gemiddelde van de betrouwbaarheidsgewichten nemen zien we dat we dezelfde δ vinden voor s_k^H als met betrouwbaarheidsgewichten gelijk aan één. Dit geldt trouwens ook voor de overige varianten.

Voor schendingsgewichten s_k^A, s_k^C, V ($= 0,9 - 0,05 - 0,05$) en VI ($= 0,9 - 0,05 - 0,10$) geeft de variant maximum de laagste δ en voor de 1% kwantieledits en de 1% kwantieledits met schendingsgewichten s_k^C benaderen we met geen enkele variant een $\delta = 65,3\%$ noch een $\delta = 65,2\%$, wat wel het geval was in paragraaf 4.1.

	α	β	γ	δ
s_k^A	45,0	1,1	12,1	63,3
s_k^C	43,4	1,0	11,8	64,3
$\hat{c_1}$-edits	46,2	0,7	12,0	64,1
V	46,4	0,6	11,9	63,8
VI	46,4	0,6	11,9	63,8
s_k^H	45,0	0,8	11,8	64,5
$\hat{c_1}$-edits s_k^C	43,9	1,0	11,8	64,1

Tabel 7.5: De resultaten van de mediaan in 'WP-blok' in procenten met $w_j \in W$. Hierbij zijn de $\hat{c}_1$-edits de 1% kwantieledits en de V, VI de methoden uit paragraaf 3.2.

	α	β	γ	δ
s_k^A	43,8	1,6	12,4	62,6
s_k^C	42,8	1,4	11,9	63,8
$\hat{c_1}$-edits	45,9	0,9	12,1	64,0
V	46,0	1,1	12,3	62,8
VI	46,0	1,1	12,3	62,8
s_k^H	44,1	0,8	11,7	64,5
$\hat{c_1}$-edits s_k^C	43,9	1,1	11,9	64,1

Tabel 7.6: De resultaten van het maximum in 'WP-blok' in procenten met $w_j \in W$. Hierbij zijn de $\hat{c}_1$-edits de 1% kwantieledits en de V, VI de methoden uit paragraaf 3.2.

7.2 De betrouwbaarheidsgewichten en de aanpassing van de doelfunctie

Uit de bovenstaande varianten kiezen we het gewogen gemiddelde, omdat we daarmee de meeste fouten localiseren. Bij deze paragraaf passen we de λ aan en het maximum zoals in paragraaf 3.3 en gebruiken de betrouwbaarheidsgewichten van 'WP-blok' uit tabel 7.1. De resultaten hiervan zijn opgenomen in de tabellen 7.7.

De resultaten met betrouwbaarheidsgewicht gelijk aan één en deze methode zijn te vinden in paragraaf 4.3.

Uit tabel 7.7 kunnen we opmaken dat de resultaten van het gewogen gemiddelde redelijk in de buurt komen van de foutlocalisatie die we ondervonden toen we de betrouwbaarheidsgewichten gelijk aan één hadden gesteld.

	α	β	γ	δ
s_k^A	55,4	0,5	13,3	58,3
s_k^C	55,3	0,4	13,2	58,6
$\hat{c_1}$-edits	55,2	0,4	13,2	58,6
V	55,5	0,4	13,2	58,3

Tabel 7.7: De resultaten $\lambda = 0,75$ + MAX in dataset 'WP-blok' in procenten met $w_j \in W$. Hierbij zijn de $\hat{c}_1$-edits de 1% kwantieledits en de V de methode uit paragraaf 3.2.

De resultaten met $\lambda = 0,90$ en het maximum zijn opgenomen in bijlage G en tabel G.2, met de betrouwbaarheidsgewichten zoals uit tabel 7.1. Met variant: het gewogen gemiddelde en de betrouwbaarheidsgewichten benaderen we helaas niet de δ zoals in paragraaf 4.3. Tevens zijn de gevonden resultaten in tabel 7.7 bijna gelijk aan de resultaten van 'hard'. Dit wordt veroorzaakt doordat we de $\lambda = 0,75$ ofwel $\lambda = 0,90$ hebben gekozen.

7.3 De nearest neighbormethode met betrouwbaarheidsgewichten

In deze paragraaf passen we de nearest neighbormethode toe in combinatie met de betrouwbaarheidsgewichten. Hierbij houden we rekening met het feit dat als het ruwe record een schending heeft in de orde van grootte van de schending in het nearest neighborrecord, dan kunnen we als schendingsgewicht een nul geven. Dus hij mag dan zonder kosten geschonden blijven. Als het ruwe record niet geschonden is en het nearest neighborrecord wel wordt geschonden, maakt het in principe niet uit, omdat het dan geen schendingsgewicht toegekend krijgt.

De resultaten van de NN-methode met betrouwbaarheidsgewichten $w_j = 1$ en $w_j \in W$ zijn weergegeven in tabel 7.8, waarbij W = (5,2,2,1,2,2,3,2,3,3). Dit is de verzameling van schendingsgewichten uit tabel 7.1.

	α	β	γ	δ
NN(1.25;0.1), $w_j = 1$	44,3	0,2	11,7	60,5
NN(2;0.1), $w_j = 1$	35,1	0,4	11,3	64,7
NN(5;0.1), $w_j = 1$	38,1	1,1	11,3	63,6
NN(1.25;0.1), $w_j \in W$	53,3	0,3	12,9	60,2
NN(2;0.1), $w_j \in W$	46,1	0,5	12,0	64,0
NN(5;0.1), $w_j \in W$	37,7	1,3	11,5	62,8

Tabel 7.8: De resultaten van de NN-methode met $w_j = 1$ en $w_j \in W$ en dataset 'WP-blok' in procenten.

Bij deze methode kunnen we duidelijk zien dat er een samenhang bestaat tussen de $w_j = 1$ en de $w_j \in W$. Voor $w_j = 1$ localiseren we met de nearest-neighbormethode de meeste fouten met een hoog gewicht van 2. Voor $w_j \in W$ localiseren we de meeste fouten ook met 2 en dat is de mediaan in $w_j \in W$.

We kunnen dus opmerken dat er een samenhang kan bestaan tussen de betrouwbaarheidsgewichten en de schendingsgewichten. Dit is goed redeneerbaar met behulp van formule (3.5) die we steeds gebruiken om de totale kosten te bepalen. Dit kan dus een interessant onderwerp zijn voor nader onderzoek.

Hoofdstuk 8

De conclusie en aanbevelingen

In deze nota zijn een aantal methoden beschreven die ertoe kunnen leiden dat meer data automatisch kunnen worden gaafgemaakt door onderscheid te maken tussen de harde en zachte edits. De oplossingen die we gedurende dit onderzoek hebben gevonden door de zachte edits mee te nemen als zacht in de foutlocalisatie hebben te allen tijde geleid tot een betere foutlocalisatie. Dit is in tegenstelling tot de momenteel gebruikte methoden op het CBS, waarbij de zachte edits helemaal niet worden gebruikt ofwel de zachte edits als 'hard' worden aangenomen.

We hebben twee varianten van het algoritme beschreven. In de eerste, meest eenvoudige variant van het probleem kreeg elke zachte restrictie een **statisch schendingsgewicht** s_k. Voor deze variant hebben we een aantal methoden onderzocht om deze statische gewichten automatisch te kiezen, gegeven twee datasets: 'test-data' en 'WP-blok'. Doordat we van beide datasets beschikken over de oorspronkelijke ruwe dataset en de reeds gaafgemaakte dataset, konden we de gelocaliseerde fouten per methode vergelijken met de gave dataset. Hierbij zijn we uitgegaan van het feit dat de referentiedataset en de gave dataset geen fouten bevatten en in beide datasets de fouten goed zijn gelocaliseerd en geïmputeerd. Echter hoeft dit niet het geval te zijn en zit er dus een risico aan het gebruik van de referentiedata.

Eén van de methoden die voor de eerste variant goede resultaten heeft gegeven is de methode waarbij de we de kwantieledits gebruiken in plaats van de oorspronkelijke edits. Vooral de drie kwantieledits (10%, 5% en de 1%) die we tegelijk gebruiken, geeft goede resultaten voor de foutlocalisatie. Doordat we de kwantieledits (10%, 5% en de 1%) tegelijk meenemen, kan er bij het automatisch gaafmaken onderscheid worden gemaakt tussen de verschillende soorten schendingen, zoals beschreven in paragraaf 3.2. In dit onderzoek hebben wij steeds slechts een deel van de volledige dataset

gaafgemaakt. Indien de gehele dataset gaafgemaakt moet worden kunnen de kwantieledits, het foutlocalisatieprobleem moeilijker oplosbaar maken, omdat we dan drie keer zo veel zachte edits hebben, want hoe meer edits we hebben, hoe complexer het probleem zal worden.

Andere methoden die goede resultaten hebben opgeleverd met statische schendingsgewichten zijn de methoden waarbij we de fractie niet-fouten berekenen in de referentiedata s_k^B en de fractie onterechte fouten berekenen s_k^G, vervolgens beide methoden verdelen tussen (0,5; 1; 1,5) door middel van een boven- ($\hat{a}$) en/of ondergrens ($\hat{b}$). Deze verdeling is gebaseerd op de betrouwbaarheidsgewichten die wij in een groot deel van dit onderzoek gelijk aan één hebben gesteld. Indien de betrouwbaarheidsgewichten van de gaafmakers worden gebruikt, zouden deze verdelingen daarop kunnen worden gebaseerd met behulp van de beschreven methoden in paragraaf 7.1, waarbij het gewogen gemiddelde het beste resultaat geeft.

De aanpassingen van de doelfunctie hebben over het algemeen niet geleid tot betere resultaten, behalve voor dataset 'test-data' waarbij de zachte edits niet de oorspronkelijke zachte edits van de gaafmakers zijn. Echter kan verder onderzocht worden of een aanpassing van de $0,50 \leq \lambda \leq 0,75$ kan bijdragen aan een betere foutlocalisatie. Wij hebben in dit onderzoek enkel de aanpassingen van $\lambda = 0,75$ en $\lambda = 0,90$ toegepast. De aanpassing van de doelfunctie naar het maximum en het gebruik van de chi-kwadraatmethode hebben beide niet geleid tot een betere foutlocalisatie. We kunnen concluderen dat we door het gebruik van de minimale som (formule 3.1) in de doelfunctie meer fouten localiseren.

Een geheel ander methode die vaak wordt gebruikt binnen de statistiek voor de imputatie van ontbrekende waarden is de nearest neighbormethode die we in dit onderzoek hebben gebruikt voor foutlocalisatie. Het gebruik van deze methode geeft geen betere resultaten voor de foutlocalisatie dan de verkregen resultaten voor de kwantieledits, maar is een redelijk toepasbare methode voor foutlocalisatie.

In de tweede variant hebben we onderzocht wat een goede manier is om de schendingsgewichten automatisch te bepalen, gegeven de schendingen, waarbij elk schendingsgewicht afhangt van de grootte van de bijbehorende schending en van de mate waarin een edit belangrijk is, de **dynamische schendingsgewichten**. Hierbij hebben we gekeken naar de grootte van de schendingen van de zachte edits en deze schendingsgrootte gestandaardiseerd en het als schendingsgewicht gebruikt voor de zachte edits. Dit heeft niet geleid tot een betere foutlocalisatie in vergelijking met de statische schendingsgewichten. Een reden hiervoor kan zijn dat het foutlocalisatieprobeem niet opgelost kan worden voor sommige records. Er zou nog verder onderzoek hiernaar gedaan kunnen worden. Tevens hebben we tijdens het uitvoeren van de foutlocalisatatie voor deze methode ontdekt dat er een verschil

zit bij het localiseren van fouten indien er meer dan één oplossing wordt gevonden met dezelfde totale kosten. Dus verder onderzoek voor oplossingen met gelijke kosten is nog voor verbetering vatbaar. Bij het onderzoek naar automatische foutlocalisatie hebben we naast de statische schendingsgewichten en dynamische schendingsgewichten ook gebruik gemaakt van wat geavanceerdere methoden, namelijk de Mahalanobis-afstand en logistische regressie. Echter kunnen we concluderen dat beide methoden niet hebben geleid tot een verbetering voor automatische foutlocalisatie met zachte edits.

Alhoewel we niet één methode hebben die voor beide toegepaste datasets het beste resultaat geeft, kunnen we toch concluderen dat naar aanleiding van de toegepaste methoden in dit onderzoek de eerste variant in de praktijk leidt tot een betere foutlocalisatie dan de tweede variant. Hierbij is de eerste variant dat elke zachte edit een statisch schendingsgewicht s_k krijgt en de tweede variant is dat elke zachte edit een schendingsgewicht s_k krijgt die afhangt van de grootte van de bijbehorende schending en de mate waarin een edit belangrijk is. Met de methoden die zijn beschreven in de eerste variant localiseren we echter nog niet alle fouten die een dataset bevatten, maar wel ruim 65% van alle aanwezige fouten in een dataset. De belangrijkste records kunnen nog steeds door gaafmakers gelocaliseerd en gecorrigeerd worden, omdat de kans groot is dat deze fouten kunnen leiden tot een vertekening in schattingen van publicatiecijfers. Tevens kan de handmatig gaafgemaakte dataset met de belangrijkste bedrijven en enkele minder belangrijke bedrijven gebruikt worden als referentie data voor het automatisch gaafmaken. Hierbij is wel de vraag of gaafmakers alle fouten die de dataset bevat kunnen localiseren en corrigeren, zodanig dat er geen vertekend beeld ontstaat. In de praktijk blijkt dat gaafmakers meer restricties toepassen bij het gaafmaken dan dat ze in de gaafmaakinstructies vermelden. Indien deze restricties niet worden gehandhaafd bij de automatische foutlocalisatie is de kans dat de automatische foutlocalisatie lijkt op de handmatig foutlocalisatie kleiner. Daarom is het van groot belang dat de gaafmakers goede gaafmaakinstructies opstellen. Een aanbeveling is hierbij ook dat alle mogelijk restricties worden vastgelegd in de gaafmaakinstructies, waarbij de gaafmakers zelfs kunnen aangeven welke zachte restrictie ze vaker hebben toegepast dan de andere en dit is weer goed bruikbaar voor de automatische foutlocalisatie. In dit onderzoek is voornamelijk gebruik gemaakt van de succesmaten α, β, γ en de δ. Verder onderzoek zou gedaan kunnen worden naar de succesmaten: sensitiviteit, specificiteit en de geïmputeerde waarden.

Ten slotte willen we benadrukken dat dit onderzoek ook laat zien dat er mogelijk een verband kan zitten tussen de betrouwbaarheidsgewichten en de schendingsgewichten. Dit kan een interessant onderwerp zijn voor nader onderzoek. Voor dataset 'test-data' zijn de fouten random verdeeld over de variabelen, waarbij de betrouwbaarheidsgewichten geen toegevoegde waarde hebben. Voor dataset 'WP-blok' hebben de betrouwbaarheidsgewichten wel

een toegevoegde waarde, maar omdat we in de eerste instantie de toegevoegde waarde van de zachte edits wilden onderzoeken, hebben we de betrouwbaarheidsgewichten gelijk aan één gekozen. Dit heeft als voordeel dat we bij de keuze van de schendingsgewichten (s_k) geen rekening hoefden te houden met de betrouwbaarheidsgewichten (w_j) van de variabelen die van elkaar verschillen. In hoofdstuk 7 is dit verband tussen de s_k en de w_j weergegeven.

Bijlage A

De edits van beide datasets

De harde edits in de dataset ‘test-data’:

$$
\begin{aligned}
&x_1 + x_2 = x_3 \\
&x_2 = x_4 \\
&x_5 + x_6 + x_7 = x_8 \\
&x_3 + x_8 = x_9 \\
&x_9 - x_{10} = x_{11} \\
&x_1 \geq 0 \\
&x_2 \geq 0 \\
&x_3 \geq 0 \\
&x_4 \geq 0 \\
&x_5 \geq 0 \\
&x_6 \geq 0 \\
&x_7 \geq 0 \\
&x_8 \geq 0 \\
&x_9 \geq 0 \\
&x_{10} \geq 0 \\
&x_{11} \geq 0 \\
&x_{12} \geq 0
\end{aligned}
$$

De zachte edits in de dataset ‘test-data’:

$$
\begin{aligned}
&x_2 - 0.5x_3 \geq 0\\
&x_3 - 0.9x_9 \geq 0\\
&x_5 + x_6 - x_7 \geq 0\\
&x_9 - 50x_{12} \geq 0\\
&5000x_{12} - x_9 \geq 0\\
&0.4x_9 - x_{11} \geq 0\\
&10x_{11} + x_9 \geq 0\\
&x_{12} - 1 \geq 0\\
&x_{12} - 5 \geq 0\\
&100 - x_{12} \geq 0
\end{aligned}
$$

De harde edits in de dataset ‘WP-blok’:

$$
\begin{aligned}
&x_1 = x_4 - x_7 + x_8 + x_6 + x_5\\
&x_9 = x_4 - x_7\\
&x_{10} = x_8 + x_6 + x_5\\
&x_1 \geq 0\\
&x_2 \geq 0\\
&x_3 \geq 0\\
&x_4 \geq 0\\
&x_5 \geq 0\\
&x_6 \geq 0\\
&x_7 \geq 0\\
&x_8 \geq 0\\
&x_9 \geq 0\\
&x_{10} \geq 0\\
&x_1 \geq x_2\\
&x_4 \geq x_3\\
&\text{ALS } (h_2 > 5 \text{ OF } x_3 > 2) \text{ DAN } h_1 > 0\\
&\text{ALS } x_4 > 0 \text{ DAN } h_2 > 0
\end{aligned}
$$

De zachte edits in de dataset 'WP-blok':

$x_1 \geq x_4$
$x_2 \geq x_3$
$x_2 \geq 0.001$
ALS $h_3 \neq 0$ DAN $(x_8) \neq 0$
ALS $x_7 \neq 0$ DAN $h_6 \neq 0$ OF $h_7 \neq 0)$
ALS $(x_1 \neq 0$ OF $x_2 \neq 0)$ DAN $h_5 \neq 0$
ALS $(x_1 \neq 0$ OF $x_2 \neq 0)$ DAN $h_8 \neq 0$
ALS $h_2 \neq 0$ DAN $(x_4 \neq 0$ EN $x_3 \neq 0)$
ALS $h_4 \neq 0$ DAN $x_6 \neq 0$
ALS $h_6 \neq 0$ DAN $x_7 \neq 0$
ALS $x_8 \neq 0$ DAN $h_3 \neq 0$
ALS $x_6 \neq 0$ DAN $h_4 \neq 0$
ALS GK $= 4$ DAN $(x_1 \geq 10$ EN $x_1 < 20)$
ALS GK $= 5$ DAN $(x_1 \geq 20$ EN $x_1 < 50)$
ALS GK $= 6$ DAN $(x_1 \geq 50$ EN $x_1 < 100)$

De variabelen h_1 t/m h_8 zijn de hulpvariabelen die in de dataset voorkomen, maar die we beschouwen als foutloos.

Bijlage B

De schendingsgewichten s_k^B en s_k^C

Edit	% fout in dataset	s_k^B	s_k^C
$x_2 - 0,5x_3 \geq 0$	12	0,88	0,50
$x_3 - 0,9x_9 \geq 0$	4	0,96	1,50
$x_5 + x_6 - x_7 \geq 0$	8	0,92	0,50
$x_9 - 50x_{12} \geq 0$	2	0,98	1,50
$5000x_{12} - x_9 \geq 0$	7	0,93	0,50
$0,4x_9 - x_{11} \geq 0$	2	0,98	1,50
$10x_{11} + x_9 \geq 0$	5	0,95	1,00
$x_{12} - 1 \geq 0$	2	0,88	0,50
$x_{12} - 5 \geq 0$	5	0,95	1,00
$100 - x_{12} \geq 0$	3	0,97	1,50

Tabel B.1: Het percentage fout in 'test-data' met bijbehorende s_k^B en s_k^C

Edit	s_k^B	s_k^C
1 $x_1 \geq x_4$	0,98	1,00
2 $x_2 \geq x_3$	0,97	1,00
3 $x_2 \geq 0,001$	0,98	1,00
4 *ALS* $h_3 \neq 0$ *DAN* $(x_8) \neq 0$	0,98	1,00
5 *ALS* $x_7 \neq 0$ *DAN* $(h_6 \neq 0$ *OF* $h_7 \neq 0)$	1,00	1,50
6 *ALS* $x_7 \neq 0$ *DAN* $(h_6 \neq 0$ *OF* $h_7 \neq 0)$	1,00	1,50
7 *ALS* $(x_1 \neq 0$ *OF* $x_2 \neq 0)$ *DAN* $h_5 \neq 0$	1,00	1,50
8 *ALS* $(x_1 \neq 0$ *OF* $x_2 \neq 0)$ *DAN* $h_5 \neq 0$	1,00	1,50
9 *ALS* $(x_1 \neq 0$ *OF* $x_2 \neq 0)$ *DAN* $h_8 \neq 0$	1,00	1,50
10 *ALS* $(x_1 \neq 0$ *OF* $x_2 \neq 0)$ *DAN* $h_8 \neq 0$	1,00	1,50
11 *ALS* $h_2 \neq 0$ *DAN* $(x_4 \neq 0$ *EN* $x_3 \neq 0)$	1,00	1,50
12 *ALS* $h_2 \neq 0$ *DAN* $(x_4 \neq 0$ *EN* $x_3 \neq 0)$	1,00	1,50
13 *ALS* $h_4 \neq 0$ *DAN* $x_6 \neq 0$	0,99	1,50
14 *ALS* $h_4 \neq 0$ *DAN* $x_6 \neq 0$	1,00	1,50
15 *ALS* $h_6 \neq 0$ *DAN* $x_7 \neq 0$	1,00	1,50
16 *ALS* $x_8 \neq 0$ *DAN* $h_3 \neq 0$	1,00	1,50
17 *ALS* $x_6 \neq 0$ *DAN* $h_4 \neq 0$	1,00	1,50
18 *ALS* $x_6 \neq 0$ *DAN* $h_4 \neq 0$	1,00	1,50
19 *ALS GK = 4 DAN* $(x_1 \geq 10)$	0,89	0,50
20 *ALS GK = 4 DAN* $(x_1 < 20)$	0,86	0,50
21 *ALS GK = 5 DAN* $(x_1 \geq 20$	0,94	0,50
22 *ALS GK = 5 DAN* $(x_1 < 50)$	0,88	0,50
23 *ALS GK = 6 DAN* $(x_1 \geq 50)$	0,98	1,00
24 *ALS GK = 6 DAN* $(x_1 < 100)$	0,98	1,00

Tabel B.2: Dataset 'WP-blok' met bijbehorende s_k^B en s_k^C

Bijlage C

De schendingsgewichten s_k^E, s_k^F, s_k^G, s_k^H en s_k^I

Edit	**a**	**c**	**b**	**d**	**NA**	s_k^E	s_k^F	s_k^G	s_k^H	s_k^I
1	75	12	132	509	0	0,88	0,86	0,64	0,50	0,86
2	25	6	143	554	0	0,96	0,96	0,85	1,50	0,81
3	53	7	31	637	0	0,92	0,91	0,37	0,50	0,88
4	13	0	109	606	0	0,98	0,98	0,89	1,50	1,00
5	37	12	90	589	0	0,93	0,93	0,71	1,00	0,76
6	12	6	99	611	0	0,98	0,97	0,89	1,50	0,67
7	27	8	42	651	0	0,95	0,95	0,61	0,50	0,77
8	14	0	61	653	0	0,98	0,98	0,81	1,00	1,00
9	34	3	78	613	0	0,95	0,95	0,70	0,50	0,92
10	16	3	60	649	0	0,97	0,97	0,79	1,00	0,84

Tabel C.1: De schendingsgewichten s_k^E, s_k^F, s_k^G, s_k^H en s_k^I in dataset 'test-data'

Edit	**a**	**c**	**b**	**d**	**NA**	s_k^E	s_k^F	s_k^G	s_k^H	s_k^I
1	8	5	12	539	16	0,98	0,98	0,60	0,50	0,62
2	10	7	61	389	113	0,96	0,96	0,86	1,00	0,59
3	9	2	10	451	108	0,98	0,98	0,53	0,50	0,82
4	4	0	25	517	34	0,99	0,99	0,86	1,00	1,00
5	0	0	0	311	269	1,00	1,00	1,00	1,50	1,00
6	2	0	10	299	269	0,99	0,99	1,00	1,50	1,00
7	0	0	0	580	0	1,00	1,00	1,00	1,50	1,00
8	0	0	1	579	0	1,00	1,00	1,00	1,50	1,00
9	0	0	0	578	2	1,00	1,00	1,00	1,50	1,00
10	0	0	0	578	2	1,00	1,00	1,00	1,50	1,00
11	0	0	2	572	6	1,00	1,00	1,00	1,50	1,00
12	0	0	6	549	25	1,00	1,00	1,00	1,50	1,00
13	1	0	8	554	17	1,00	1,00	0,89	1,00	1,00
14	1	1	2	570	6	1,00	1,00	0,67	0,50	0,50
15	0	0	0	445	135	1,00	1,00	1,00	1,50	0,00
16	2	0	16	427	135	1,00	1,00	0,89	1,00	1,00
17	0	0	0	284	296	1,00	1,00	1,00	1,50	0,00
18	1	0	17	266	296	1,00	1,00	0,94	1,00	1,00
19	55	3	9	508	4	0,89	0,90	0,14	0,50	0,95
20	77	1	4	494	4	0,84	0,86	0,05	0,50	0,99
21	35	0	8	529	8	0,93	0,94	0,19	0,50	1,00
22	60	4	3	505	8	0,87	0,89	0,05	0,50	0,94
23	11	0	3	566	0	0,98	0,98	0,21	0,50	1,00
24	9	1	0	570	0	0,98	0,98	1,00	1,50	0,90

Tabel C.2: De schendingengewichten s_k^E, s_k^F, s_k^G, s_k^H en s_k^I in 'WP-blok'.

Bijlage D

De verschilmethode met s_k^J

De verschillen (v_k) voor dataset 'test-data' zijn opgenomen in bijlage C. De verschillen (v_k) voor dataset 'WP-blok' zijn opgenomen in de tabel hieronder.

Edit	**Verschil I**	**Verschil II (zonder ontbrekende waarden)**	s_k^J
1.	0,33	0,3084	0,50
2.	1,59	1,642	0,20
3.	17,43	19,895	0,10
4.	0,00001	0,000007	1,50
5.	0,000001	0,000002	1,50
6.	0,000001	0,000002	1,50
7.	0,000001	0,0000002	1,50
8.	0,000001	0,0000002	1,50
9.	0	0	2,00
10.	0	0	2,00
11.	0,002	0,0013	1,00
12.	0,23	0,23	0,50
13.	0,000001	0,000003	1,50
14.	0,000001	0,0000004	1,50
15.	0,000001	0,0000008	1,50
16.	0,000001	0,0000008	1,50
17.	0,000001	0,000006	1,50
18.	0,000001	0,000006	1,50
19.	0,000001	0,000009	1,50
20.	0,000001	0,000009	1,50
21.	0,000001	0,00004	1,50
22.	0,000001	0,00004	1,50
23.	0,000001	0,000007	1,50
24.	0,000001	0,000007	1,50

Tabel D.1: De resultaten van de verschilmethode in 'WP-blok' per edit en de bijbehorende s_k^J.

Bijlage E

De nearest neighbormethode

De R-code voor het bepalen van de NN-record:

record ¡- 1

nearestNeighbor ¡- function(dataRuw, dataRef, record){

#gemidelde van de eerste kolom is met gem[1] of gem[“x1”] (of misschien gem$x1)

gem ¡- apply(X=dataRef[,-1], MARGIN=2, FUN=mean)

#standaarddeviatie

stdev ¡- apply(X=dataRef[,-1], MARGIN=2, FUN=sd)

#(elke ref waarde in de dataRef - mean)/ stdev

verschilRef ¡- scale(dataRef[,-1], center = gem, scale = stdev)

#(elke ruwe waarde in de dataRuw - mean)/ stdev

verschilRuw ¡- scale(dataRuw[record,-1], center = gem, scale = stdev)

besteTotNuToe ¡- 1000000

besteI ¡- 0

for(i in 1:dim(dataRef)[1]){

absoluteVerschil ¡- abs(verschilRuw-verschilRef[i,])

somVerschillen ¡- sum(absoluteVerschil, na.rm=TRUE)

if(somVerschillen ¡ besteTotNuToe){

besteTotNuToe ¡- somVerschillen

besteI ¡- i }

if(besteI == 0){

stop(“Er is geen nearest neighbor record gevonden voor deze ruwe record”)
}

return(besteI) }

De gecreëerde extra edits naar aanleiding van een gesprek met de gaafmaker:

$$\begin{aligned}
&\textbf{extra edits 1:} && \frac{h_2}{x_4} \geq 14 \\
& && \frac{h_2}{x_4} \geq 81 \\
&\textbf{extra edits 2:} && \frac{h_3}{x_8} \geq 4,7 \\
& && \frac{h_3}{x_8} \geq 89,5 \\
&\textbf{extra edits 3:} && \frac{h_4}{x_6} \geq 1,9 \\
& && \frac{h_4}{x_6} \geq 171,3 \\
&\textbf{extra edits 4:} && \frac{h_8}{x_1} \geq 4,6 \\
& && \frac{h_8}{x_1} \geq 144,7 \\
&\textbf{extra edits 5:} && \frac{h_6}{x_7} \geq 28,6 \\
& && \frac{h_6}{x_7} \geq 128
\end{aligned}$$

Per extra edit geven we twee waarden weer. Hierbij is de eerste het 5% kwantiel en de tweede is het 95% kwantiel. Deze gevonden extra edits kunnen we meegeven aan onze errorlocalisator. Daarbij is de rechtsvorm ook een andere extra edit die we mee kunnen gebruiken. Als de rechtsvorm een eenmanszaak (EZ) of een Cooperatieve Vereniging (CV) is, geldt dat het aantal werkzame eigenaren gelijk is aan één. Als de rechtsvorm een maatschap of een Vennootschap Onder Firma (VoF) is, dan zijn het aantal werkzame personen gelijk aan twee. Als de rechtsvorm een Besloten Vennootschap (BV) is, dan zijn het aantal werkzame personen gelijk aan nul. Hiermee vinden we als extra edits:

$$\begin{aligned}
&ALS\ RECHTSVORM = 61|22|65|1|21\ DAN\ x_5 = 1 \\
&ALS\ RECHTSVORM = 11\ DAN\ x_5 = 2 \\
&ALS\ RECHTSVORM = 41|42\ DAN\ x_5 = 0
\end{aligned}$$

Bijlage F

De chi-kwadraatmethode

zachte edits	k	l	m	n	χ^2	Kwantiel χ^2
1	28	8	135	409	46,9	1,00
2	176	8	128	268	202	1,00
3	119	8	185	268	111,1	1,00
4	59	4	161	356	93,2	1,00
5	269	0	72	239	351,6	1,00
6	10	2	331	237	3	0,92
7	0	0	134	446	0	0,00
8	133	443	1	3	0,008	0,07
9	2	0	279	299	2,14	0,86
10	281	297	0	2	1,89	0,83
11	8	0	67	505	54,6	1,00
12	31	0	90	459	124,2	1,00
13	25	1	316	238	15,7	1,00
14	8	1	333	238	3,4	0,94
15	135	0	85	360	287,9	1,00
16	204	205	16	155	84,1	1,00
17	296	0	45	239	423,7	1,00
18	324	42	17	197	361,9	1,00
19	33	36	308	203	3,9	0,95
20	279	213	62	26	5,8	0,98
21	28	23	313	216	0,35	0,45
22	307	207	34	32	1,6	0,80
23	7	7	334	232	0,46	0,50
24	335	235	6	4	0,006	0,06

Tabel F.1: De chi-kwadraat op dataset 'WP-BLOK'.

zachte edits:	χ^2	**Kwantiel** χ^2
$x_2 - 0,5x_3 \geq 0$	63,10	1,00
$x_3 - 0,9x_9 \geq 0$	152,70	1,00
$x_5 + x_6 - x_7 \geq 0$	57,10	1,00
$x_9 - 50x_{12} \geq 0$	136,10	1,00
$5000x_{12} - x_9 \geq 0$	72	1,00
$0,4x_9 - x_{11} \geq 0$	79,40	1,00
$10x_{11} + x_9 \geq 0$	12,80	1,00
$x_{12} - 1 \geq 0$	0,04	0,15
$x_{12} - 5 \geq 0$	0,26	0,39
$100 - x_{12} \geq 0$	0,0035	0,05

Tabel F.2: De chi-kwadraat op dataset ‘test-data’.

Bijlage G

De betrouwbaarheidsgewichten

In deze bijlage zijn de betrouwbaarheidsgewichten gebruikt, zoals aangekondigd in hoofdstuk 7.

	α	β	γ	δ
s_k^A	53,4	0,3	12,8	60,5
s_k^C	53,4	0,3	12,8	60,5
$\hat{c}_1$-edits	53,8	0,3	12,9	60,0
V	54,8	0,5	13,0	59,7
VI	54,3	0,5	12,9	59,8
s_k^H	54,1	0,2	12,9	59,7
$\hat{c}_1$-edits s_k^C	53,8	0,3	12,9	60,0

Tabel G.1: De succesmaten van de statische schendingsgewichten met $w_j \in W$ in dataset 'WP-blok' in procenten.

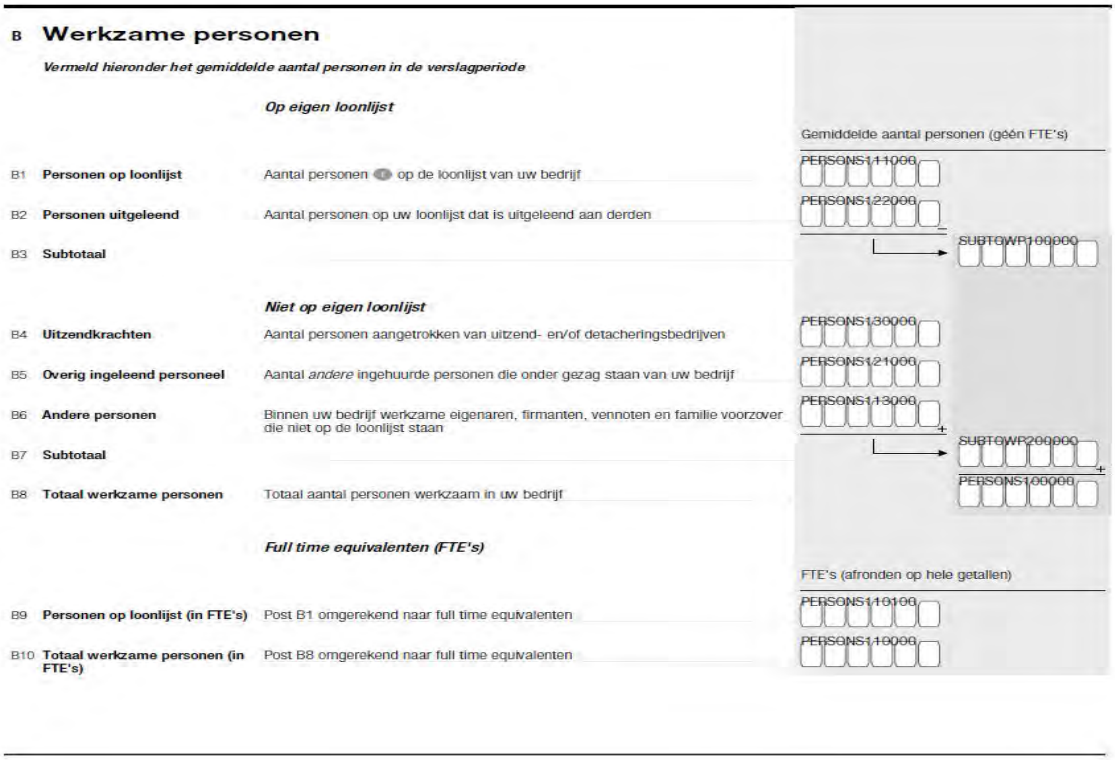
	α	β	γ	δ
s_k^A	55,7	0,5	13,1	57,8
s_k^C	55,5	0,5	13,1	58,1
$\hat{c}_1$-edits	55,6	0,5	13,1	57,9
V	55,7	0,5	13,1	57,8
VI	55,7	0,5	13,1	57,8
s_k^H	55,5	0,5	13,1	58,1
$\hat{c}_1$-edits s_k^C	55,5	0,5	13,1	58,1

Tabel G.2: De succesmaten met $\lambda = 0,90$ + MAX en $w_j \in W$ in dataset 'WP-blok' in procenten.

Bijlage H

De vragenlijst bij dataset ‘WP-blok’

Van boven naar beneden zijn de variabelen als volgt gekozen: x_4, x_7, x_9, x_8, x_6, x_5, x_{10}, x_1, x_3 en x_2.



B Werkzame personen

Vermeld hieronder het gemiddelde aantal personen in de verslagperiode

			Gemiddelde aantal personen (géén FTE's)
		Op eigen loonlijst	
B1	**Personen op loonlijst**	Aantal personen op de loonlijst van uw bedrijf	PERSONS141000
B2	**Personen uitgeleend**	Aantal personen op uw loonlijst dat is uitgeleend aan derden	PERSONS122000
B3	**Subtotaal**		SUBTOWP100000
		Niet op eigen loonlijst	
B4	**Uitzendkrachten**	Aantal personen aangetrokken van uitzend- en/of detacheringsbedrijven	PERSONS130000
B5	**Overig ingeleend personeel**	Aantal *andere* ingehuurde personen die onder gezag staan van uw bedrijf	PERSONS121000
B6	**Andere personen**	Binnen uw bedrijf werkzame eigenaren, firmanten, vennoten en familie voorzover die niet op de loonlijst staan	PERSONS113000
B7	**Subtotaal**		SUBTOWP200000
B8	**Totaal werkzame personen**	Totaal aantal personen werkzaam in uw bedrijf	PERSONS100000
		Full time equivalenten (FTE's)	FTE's (afronden op hele getallen)
B9	**Personen op loonlijst (in FTE's)**	Post B1 omgerekend naar full time equivalenten	PERSONS110100
B10	**Totaal werkzame personen (in FTE's)**	Post B8 omgerekend naar full time equivalenten	PERSONS110000

Bibliografie

[1] J. Hoogland, M. van der Loo, J. Pannekoek en S. Scholtus (2010), *Methodenreeks: Controle en correctie*. Rapport 25-01-2010, Centraal Bureau voor de Statistiek (zie ook op www.cbs.nl)

[2] S. Scholtus (2010), *Betrouwbaarheidsgewichten voor het automatisch gaafmaken bij de productiestatistieken*. Rapport 04-05-2010, Centraal Bureau voor de Statistiek.

[3] S. Scholtus (2010), *Automatisch gaafmaken met zachte edits*. Rapport 20-07-2010, Centraal Bureau voor de Statistiek.

[4] J. Bethlehem (Module 1), *Startcursus Methodologie, Module 1: Inleiding statistisch proces*, Centraal Bureau voor de Statistiek.

[5] S. Scholtus (2008), *Algorithms for correcting some obvious inconsistencies and rounding errors in business survey data*, Discussie Paper 08015, Centraal Bureau voor de Statistiek (zie ook op www.cbs.nl)

[6] S. Scholtus (2009), *Automatic correction of simple typing errors in numerical data with balance edits*. Discussie Paper 09046, Centraal Bureau voor de Statistiek (zie ook op www.cbs.nl)

[7] T. de Waal, J. Pannekoek en S. Scholtus (2011), *Handbook of Statistical Data Editing and Imputation*. John Wiley & Sons, Inc., Hoboken, New Jersey.

[8] S. Scholtus (2011), *Automatic editing with soft edits*. Discussie Paper 201130, Centraal Bureau voor de Statistiek (zie ook op www.cbs.nl)

[9] D. Hedlin (2003), *Score functions to reduce business survey editing at the U.K. office for national statistics*. Journal of Official Statistics, Vol. 19, No. 2, 2003, pp. 177-199.