Zorgverzekeraars Nederland

Bijlage 7: Onderbouwing vertrektarieven Wlz 2027

Juni 2026

1. Landelijk tariefmodel als gemeenschappelijke basis

Waarom een landelijk tariefmodel?

Om de toegang tot kwalitatief goede langdurige zorg op een doelmatige manier te waarborgen gebruiken zorgkantoren een landelijk tariefmodel. Aan de hand van dit landelijke tariefmodel wordt een vertrektarief voor clusters berekend, welke dient als uitgangspunt voor de zorgkantoren in hun tariefafspraak met zorgaanbieders. Door te kiezen voor een landelijk tariefmodel dragen zorgkantoren bij aan duidelijkheid en transparantie over de gemaakte modelkeuzes. Het landelijke tariefmodel biedt daarmee een onderbouwd vertrektarief dat als uitgangspunt dient voor zorgkantoren om binnen de regio tot definitieve tariefafspraken te komen met zorgaanbieders.

Welke bronnen zijn gebruikt?

Voor het bepalen van de vertrektarieven per cluster gebruiken de zorgkantoren drie bestaande databronnen:

1. De jaarverslagen die eind juli jaar t-2 beschikbaar waren in het Intrakoop bestand van zorgaanbieders binnen de Wlz.
2. Declaratiegegevens van alle zorgkantoren over jaar t-2[1] met daarin per zorgaanbieder het aantal verrichtingen per prestatie dat is ingediend bij het voor hem verantwoordelijke zorgkantoor en het door de zorgaanbieder totaal gedeclareerde bedrag per prestatie[2]. Voor zorgaanbieders die bovenregionaal werken (d.w.z. met meerdere zorgkantoren) wordt gerekend met het totaal aantal verrichtingen per prestatie dat door de zorgaanbieder is ingediend en het door de zorgaanbieder totaal gedeclareerde bedrag per prestatie. Er wordt in een dergelijk geval niet gekeken per regio of per zorgkantoor.
3. De NZa maximumtarieven (uit de beleidsregels) van jaar t-2.

Er is door de zorgkantoren geen aanvullende uitvraag gedaan bij zorgaanbieders om de administratieve lasten te beperken. Zorgkantoren controleerden de data op basis van een outlier analyse. De outlier analyse wordt standaard uitgevoerd en afwijkende waardes (>2 * standaarddeviaties van het gemiddelde) zijn handmatig gecontroleerd door de zorgkantoren[3]. Indien de bevindingen daar aanleiding toe geven (als er bijzondere uitgaven of inkomsten aan ten grondslag liggen) worden

[1] Deze gegevens zijn tussen zorgkantoren uitgevraagd middels het uitvraagformulier zoals opgenomen in bijlage 7.c
[2] Op basis van de AVG zijn declaratiegegevens alleen geaggregeerd op prestatieniveau gedeeld tussen zorgkantoren.
[3] Vanaf 2027 wordt een aanbieder als outlier beschouwd wanneer hij meer dan twee standaarddeviaties van het gemiddelde afwijkt. Voor 2027 gold dat een aanbieder als outlier werd aangemerkt bij een afwijking van meer dan twee standaarddeviaties ten opzichte van de mediaan. Deze wijziging is doorgevoerd omdat de standaarddeviatie rekenkundig is gekoppeld aan het gemiddelde en niet aan de mediaan.

correcties doorgevoerd. Outliers die na twee correctierondes buiten de gestelde grens vielen zijn uitgesloten bij de bepaling van het vertrektarief (deel 3 model).

2. De onderbouwing van de vertrektarieven per cluster

Indeling en opzet van tariefmodel

Het tariefmodel bestaat uit drie onderdelen. Het eerste deel wijst zorgaanbieders op basis van ingediende declaraties toe aan een cluster. Het tweede deel bepaalt per zorgaanbieder een kostenpercentage op basis van de kosten die zij hebben gemaakt voor de levering van Wlz-zorgprestaties. Het derde deel rekent met behulp van deze kostenpercentages een vertrektarief uit voor ieder cluster. We lichten per onderdeel het model toe[4].

Deel 1: Bepalen cluster van individuele zorgaanbieder

Het clusteren van zorgaanbieders stelt zorgkantoren in staat meer gelijksoortige zorgaanbieders met elkaar te vergelijken, om zo tot een reëler vertrektarief te komen voor elk cluster van zorgaanbieders. In totaal zijn er zeven clusters waarin een zorgaanbieder ingedeeld kan worden. We clusteren zorgaanbieders in drie stappen:

1. Voor de clustering worden zorgaanbieders eerst ingedeeld naar sector (V&V, GZ of GGZ).
2. Vervolgens worden zorgaanbieders voor alle sectoren onderverdeeld in een volledig extramuraal en een cluster waarbij aanbieders ook intramuraal aanbod hebben (gedeeltelijk intramuraal). Dit doen we voor alle sectoren.
3. Voor de V&V sector wordt binnen het volledig extramurale cluster nog onderscheid gemaakt naar hoofdzakelijk VPT en extramuraal divers. Dit doen we niet voor de GGZ en GZ.

Hieronder wordt de methodiek van de clustering toegelicht. Voor de vaststelling van de clustering en de indeling over de clusters wordt gebruik gemaakt van de declaratiedata jaar t-2. Alle aanbieders worden ingedeeld in een cluster, ongeacht de beschikbaarheid van de jaarrekeningen in het Intrakoopbestand op 1 juli 2026 van de betreffende zorgaanbieder.

Stap 1: Hoofdsector zorgaanbieder bepalen

De sector waarin een zorgaanbieder hoofdzakelijk werkzaamheden verricht (o.b.v. totale omzet) in de Wlz is maatgevend voor de sector waarin deze zorgaanbieder wordt ingedeeld, namelijk GGZ, GZ of V&V. Bij zorgaanbieders die zorg leveren aan cliënten vanuit meerdere sectoren wordt de totale omzet in de Wlz onderverdeeld naar sector. Voor deze indeling wordt voor de toewijzing van declaraties aan een bepaalde sector aansluiting gezocht bij zorgprofielen, zoals genoemd in Bijlage A. bij artikel 2.1 van de Regeling langdurige zorg. De sector waarin een zorgaanbieder hoofdzakelijk werkzaamheden verricht

[4] Zie bijlage 7.b voor de gebruikte python code waarmee het model uitgevoerd is.

(o.b.v. hoogste sectorale omzet) in de Wlz is maatgevend voor de sector waarin deze zorgaanbieder wordt ingedeeld.

Stap 2: Onderscheid tussen volledig extramurale zorgaanbieders en gedeeltelijk intramurale aanbieders

In elke sector (V&V, GZ en GGZ) wordt onderscheid gemaakt naar (i) volledig extramurale zorgaanbieders en (ii) aanbieders die ook intramuraal aanbod hebben. Indien uit de declaratiegegevens van jaar t-2 van een zorgaanbieder blijkt dat de zorgaanbieder intramurale prestatiecodes, zoals aangegeven in de tabel in bijlage 7a, heeft gedeclareerd, wordt de betreffende zorgaanbieder beschouwd als intramurale zorgaanbieder. Echter, indien de totale Wlz-omzet van een zorgaanbieder voor minder dan 1% uit intramurale prestatiecodes bestaat, wordt deze zorgaanbieders alsnog aangemerkt als ‘extramuraal’.

Stap 3: Onderscheid tussen zorgaanbieders met hoofdzakelijk VPT en divers extramurale aanbieders

Voor de V&V-sector is vervolgens voor extramurale zorgaanbieders nog een onderscheid te maken, namelijk naar (i) zorgaanbieders die voornamelijk VPT leveren en (ii) zorgaanbieders die een divers extramuraal aanbod leveren. Ook hiervoor baseren de zorgkantoren zich op de declaratiegegevens van jaar t-2 van de zorgaanbieder. Indien een zorgaanbieder 85% of meer van zijn totale omzet in jaar t-2 heeft gerealiseerd met VPT-prestaties, dan zal de zorgaanbieders in het cluster ‘voornamelijk VPT’ worden ingedeeld. Zorgaanbieders die een divers extramuraal aanbod declareerden in jaar t-2, zullen in het cluster ‘extramuraal divers’ worden ingedeeld.

Deze clustering leidt per sector tot de volgende clusters:

V&V:

Cluster A: extramuraal divers
Cluster B: hoofdzakelijk VPT
Cluster C: gedeeltelijk intramuraal

GZ:

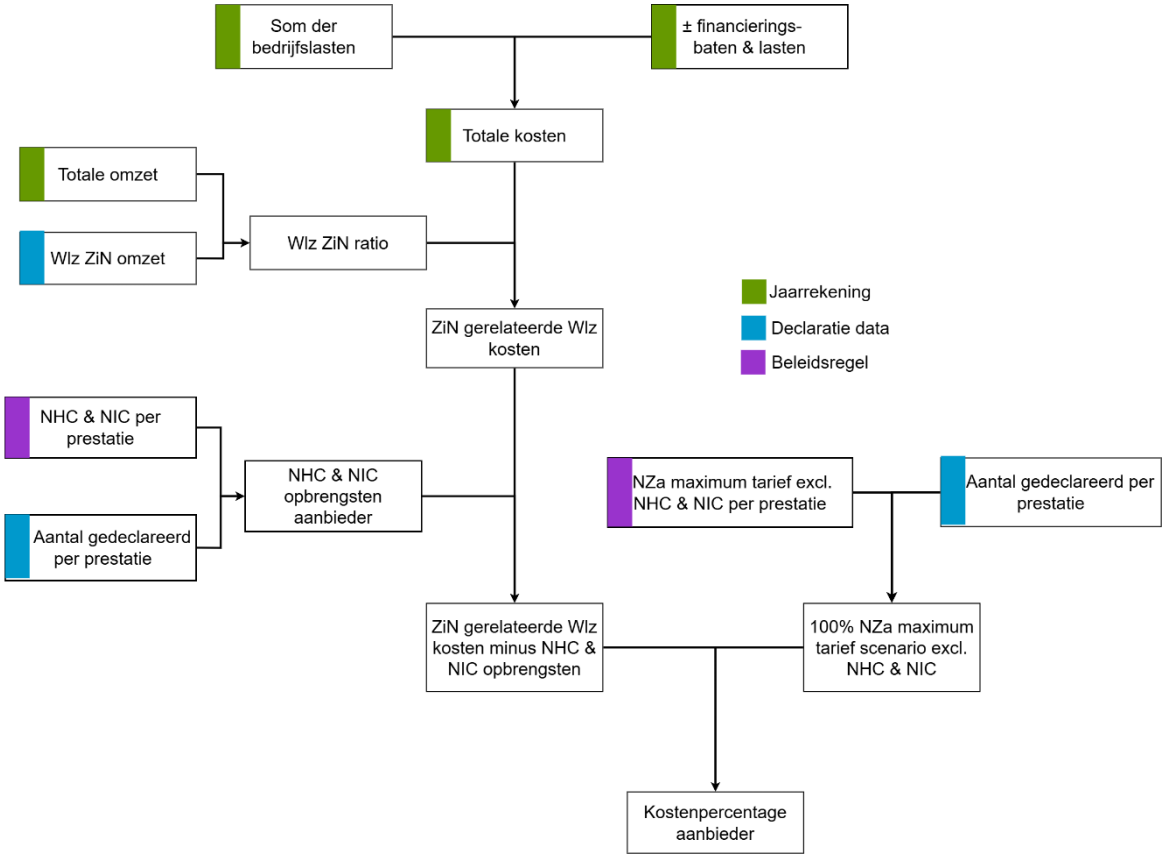
Cluster A: volledig extramuraal
Cluster B: gedeeltelijk intramuraal

GGZ:

Cluster A: volledig extramuraal
Cluster B: gedeeltelijk intramuraal

Deel 2: Bepalen zorgaanbieder percentage (kostenpercentage)

In dit deel wordt per zorgaanbieder een kostenpercentage bepaald op basis van de kosten die zij hebben gemaakt voor de levering van Wlz-zorgprestaties. Als bronnen hebben de zorgkantoren daarbij gebruik gemaakt van de bronnen als genoemd op pagina 2 hiervoor, namelijk jaarrekeningdata van zorgaanbieders, declaratiedata van zorgaanbieders en tarieven uit de geldende beleidsregel van de NZa, allemaal voor het jaar t-2. Schematisch ziet de berekening eruit als in figuur 1 waarbij stap 1, 3 en 4 identiek zijn aan voorgaande jaren en hierna een toelichting voor elk van de 4 stappen volgt.



Figuur 1. Schematische weergave van berekening om tot kostenpercentage per zorgaanbieder te komen

Stap 1: bepalen Wlz ZiN kosten

Om tot de 'totale kosten' van een aanbieder te komen worden eerst de 'som der bedrijfslasten' en de 'financieringsbaten en lasten' bij elkaar opgeteld.

Vanuit de 'totale kosten' worden vervolgens de 'ZiN gerelateerde Wlz kosten' bepaald. Dit betreffen kosten die specifiek aan 'zorg in natura' (ZiN) gerelateerd kunnen worden. Nu vanuit jaarrekeningen geen onderscheid is te maken naar (i) de kosten die aan de Wlz gerelateerd zijn en (ii) kosten uit andere domeinen (Zvw, Wmo en Jeugdwet), wordt gebruik gemaakt van 'Wlz ZiN ratio': 'de Wlz ZIN omzet' (o.b.v. declaratiegegevens) afgezet tegen 'de totale omzet' van een zorgaanbieder.[5] Hieraan ligt de

[5] De post 'som der bedrijfsopbrengsten' in de jaarrekeningen.

aanname ten grondslag dat de verhouding tussen de 'Wlz ZiN omzet' en 'de totale omzet' van een zorgaanbieder een adequate, en bovendien objectieve en uniforme afspiegeling is voor de verhouding tussen de 'ZiN gerelateerde Wlz kosten' en de 'totale kosten'. De NZa hanteert deze aanname ook voor kosten die zij, o.b.v. gegevens die ze uitvraagt bij zorgaanbieders, niet kan toewijzen aan de Wlz.

Alle Wlz zorgaanbieders, waarvan de gegevens beschikbaar zijn in het Intrakoop bestand (versie 1 juli 2026), zijn meegenomen in de berekening, ongeacht de hoogte van de omzet die zij genereren binnen de Wlz.

Definities stap 1:

Som der bedrijfslasten	**De totale kosten die een zorgaanbieder in het betreffende jaar gemaakt heeft, exclusief het saldo van de financieringsbaten en -lasten, zoals die in de jaarrekening zijn opgenomen.**
Saldo van financieringsbaten & -lasten	Het saldo van de opbrengst van vorderingen die tot de vaste activa behoren en van effecten', andere rentebaten en soortgelijke opbrengsten, waardeverandering van vorderingen die tot de vaste activa behoren en van effecten en rentelasten en soortgelijke kosten, zoals die in de jaarrekening zijn opgenomen
Totale kosten	Totale kosten die een zorgaanbieder in het betreffende jaar heeft (Som der bedrijfslasten - Saldo van financieringsbaten & -lasten)
Totale omzet	Totale omzet die een zorgaanbieder in het betreffende jaar gedraaid heeft, zoals in de jaarrekening opgenomen onder Som der bedrijfsopbrengsten.
Wlz ZiN omzet	Wlz omzet in Zorg in Natura die een zorgaanbieder in het betreffende jaar heeft, bestaande uit de som van de aangeleverde declaratiebedragen
Wlz ZiN ratio	Verhouding tussen de Wlz ZiN omzet en totale omzet om kosten toe te kunnen kennen aan het Wlz zorgdomein (Wlz ZiN omzet/ Totale omzet)
ZiN gerelateerde Wlz kosten	De kosten van een zorgaanbieder die gemaakt zijn voor het leveren van de zorgprestaties (ZiN) in het zorgdomein Wlz o.b.v. de totale kosten vermenigvuldigd met de Wlz-ZiN ratio (Totale kosten * Wlz ZiN ratio)

Stap 2: Schonen van NHC en NIC vergoeding in ZiN gerelateerde Wlz kosten

Het NZa-maximumtarief bestaat in beginsel uit (i) een zorgcomponent (loon en materieel), (ii) een NHC-component en (iii) een NIC-component. De zorgkantoren zullen 100% van het tarief voor de NHC/NIC-componenten vergoeden, en hebben daarom het model geschoond voor de NIC/NHC-componenten. Dit

betekent concreet dat de ‘ZiN gerelateerde Wlz kosten’ worden geschoond door de ‘NHC & NIC opbrengsten aanbieder’ te berekenen en deze post in mindering te brengen op de post ‘ZiN gerelateerde Wlz kosten’.

Voor het bepalen van de ‘NHC & NIC opbrengsten aanbieder’ is gekeken naar de NHC en NIC vergoeding zoals gepubliceerd in de beleidsregels van de NZa in jaar t-2 (‘NHC & NIC per prestatie’) en die vergoeding te vermenigvuldigen met het ‘aantal gedeclareerd per prestatie’. Uitkomst van deze stap is de ‘ZiN gerelateerde Wlz kosten minus NHC & NIC opbrengsten’.

Definities stap 2:

NHC & NIC per prestatie	De vergoeding van de NHC en NIC component per prestatie zoals gepubliceerd in de NZa beleidsregels in jaar t-2
Aantal gedeclareerd per prestatie	Het aantal keer dat een bepaalde prestatie gedeclareerd is in t-2 door een zorgaanbieder
NHC & NIC opbrengsten zorgaanbieder	De vergoeding van de NHC en NIC component vermenigvuldigd met het aantal prestaties dat gedeclareerd is in jaar t-2
ZiN gerelateerde Wlz kosten minus NHC en NIC opbrengsten	Na vaststelling van de ‘NHC & NIC opbrengsten aanbieder’ worden deze post in mindering gebracht op de post ‘ZiN gerelateerde Wlz kosten’

Stap 3: 100% NZa maximumtarief scenario

In deze stap wordt het kostenpercentage als ratio berekend, waarbij de daadwerkelijke kosten van een zorgaanbieder worden afgezet tegen de hypothetische omzet bij vergoeding tegen het NZa-maximumtarief zonder NHC- en NIC-component met als uitkomst het ‘100% NZa maximumtarief scenario excl. NHC & NIC’. Deze omzet in het hypothetische 100% NZa-maximumtarief-scenario berekenen we door het ‘NZa maximumtarief excl. NHC &NIC per prestatie’ te vermenigvuldigen met het aantal keer dat een bepaalde prestatie in jaar t-2 door een zorgaanbieder is gedeclareerd (‘Aantal gedeclareerd per prestatie’).

Definities stap 3:

Aantal gedeclareerd per prestatie	**Het aantal keer dat een bepaalde prestatie is gedeclareerd in jaar t-2 door een zorgaanbieder**
NZa maximumtarieven excl. NHC en NIC per prestatie	Het maximumtarief per prestatie zoals opgenomen in de beleidsregels van de NZa zonder de NHC en NIC component
100% NZa maximumtarief excl. NHC en NIC scenario	Omzet in het hypothetische scenario dat de zorgaanbieder een tariefpercentage van 100% had gekregen na schoning op NHC

	en NIC component (∑(NZa maximumtarief excl. NHC en NIC per prestatie * aantal gedeclareerd per prestatie))

Stap 4: bepalen kostenpercentage zorgaanbieder

Het kostenpercentage wordt bepaald door de ZiN gerelateerde Wlz kosten minus NHC & NIC opbrengsten af te zetten tegen het 100% NZa maximumtarief scenario.

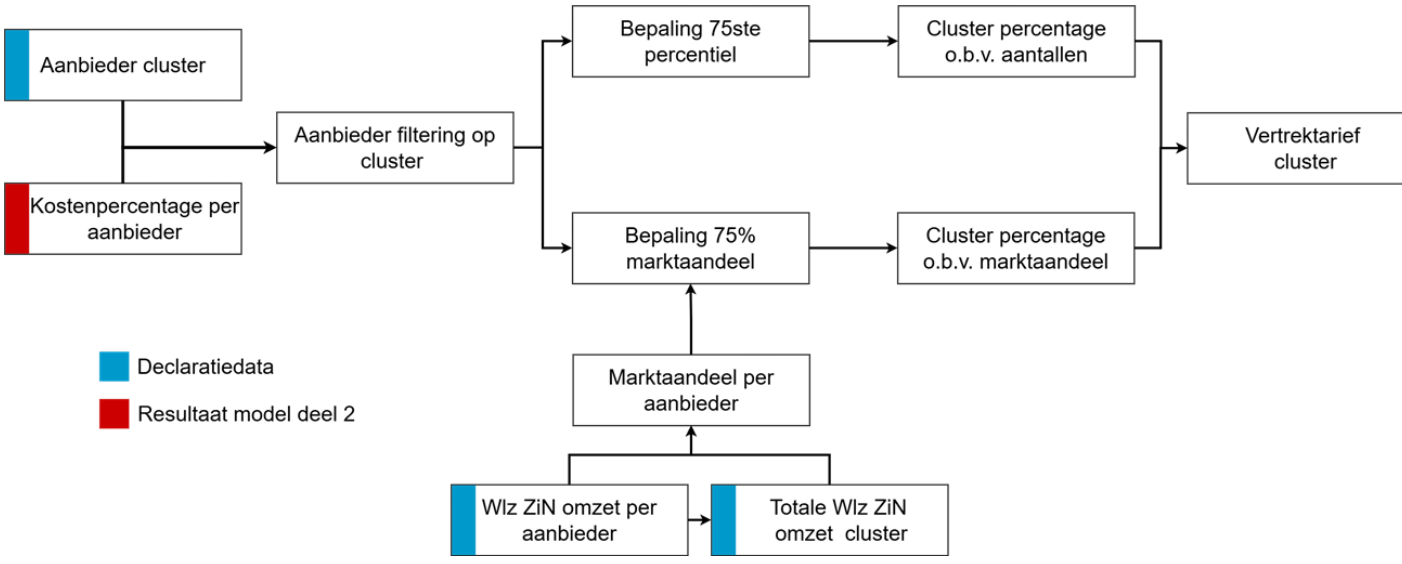
Definities stap 4:

Kostenpercentage Zorgaanbieder	**Uitdrukking van de 'de 'ZiN gerelateerde Wlz kosten minus NHC & NIC opbrengsten' als percentage van het '100% NZa maximumtarief scenario excl. NHC & NIC'. (ZiN gerelateerde Wlz kosten minus NHC & NIC opbrengsten/ 100% NZa maximumtarief excl. NHC en NIC scenario).**

Deel 3: Bepalen vertrektarief

Voor het bepalen van het vertrektarief van een cluster wordt op twee manieren een clusterpercentage vastgesteld: (i) naar aantallen zorgaanbieders: op het niveau waarbij het tarief voor 75% van de zorgaanbieders binnen een cluster exclusief outliers kostendekkend voor de Wlz is (stap 1) en (ii) naar marktaandeel: op het niveau waarbij het tarief voor minimaal 75% van de markt binnen een cluster kostendekkend voor de Wlz is (stap 2). Deze twee clusterpercentages worden met elkaar vergeleken. Het hoogste van deze twee sectorpercentage wordt gehanteerd als vertrektarief van een bepaald cluster (stap 3). Dit is schematisch weergegeven in figuur 2.

Hierna volgt een toelichting en op pagina 11 is een rekenvoorbeeld opgenomen.



Figuur 2 schematische weergave model voor het bepalen van het vertrektarief per cluster.

Stap 1: Bepaling 75ste percentiel aantallen

Bij stap 1 wordt het percentage vastgesteld op het niveau waarbij het tarief voor 75% van de zorgaanbieders uit het cluster, exclusief outliers, kostendekkend voor de Wlz is (zie ook het rekenvoorbeeld op pagina 11). De zorgkantoren houden in lijn met eerdere jaren vast aan het 75ste percentiel, nu het 75ste percentiel in rechtspraak rechtmatig is bevonden[6] voor het merendeel van de zorgaanbieders voldoende moet zijn om goede zorg te kunnen verlenen, en ook om doelmatigheid te stimuleren.

Stap 2: Bepaling 75% marktaandeel

Bij stap 2 wordt het percentage vastgesteld op het niveau waarbij het tarief voor minimaal 75% van de zorgaanbieders in een cluster naar marktaandeel kostendekkend voor de Wlz is. Met marktaandeel wordt bedoeld de Wlz ZiN omzet van alle zorgaanbieders die meegenomen zijn in het betreffende cluster, exclusief de outliers.

Stap 3: Bepaling vertrektarief cluster

In de laatste stap worden de twee clusterpercentages, op basis van aantallen en marktaandeel, met elkaar vergeleken. Het hoogste percentage wordt gehanteerd als het vertrektarief voor het betreffende cluster.

---

[6] Rechtbank Den Haag, 19 oktober 2021, ECLI:NL:RBDHA:2021:11368; Rechtbank Den Haag 31 oktober 2023, ECLI:NL:RBDHA:2023:16336; Gerechtshof Den Haag 13 februari 2024, ECLI:NL:GHDHA:2024:199.

Een illustratie van het model aan de hand van een fictieve zorgaanbieder en voorbeeld data

Aan onderstaand voorbeeld kunnen zorgaanbieders geen rechten ontlenen, en is slechts bedoeld ter illustratie van de berekening van het vertrektarief van een bepaald cluster.

Gegevens zorgaanbieder A

Onderdeel	Waarde	Bron
Totale omzet	€1000,-	Jaarrekening
Som der bedrijfslasten	€850,-	Jaarrekening
Saldo van financieringsbaten & -lasten	-€50,-	Jaarrekening
Wlz ZiN omzet	€700,-	Declaratiedata
Aantal gedeclareerd per prestatie		Declaratiedata
Prestatie 1	30	
Prestatie 2	25	
Prestatie 3	60	
NZa maximum tarief per prestatie		Beleidsregel NZa
Prestatie 1	€ 10,-	
Prestatie 2	€ 8,-	
Prestatie 3	€ 5,-	
NHC en NIC component per prestatie		
Prestatie 1	€ 1,-	
Prestatie 2	€ 0,20	
Prestatie 3	-	

Deel 2 van het model– bepaling kostenpercentage

Toelichting: In deel 2 van het model wordt een kostenpercentage per zorgaanbieder bepaald.
Som der bedrijfslasten - saldo financieringsbaten & -lasten = Totale kosten
850 –(-50) = 900

Wlz ZiN omzet / Totale omzet = Wlz ZiN ratio
700/1000 = 0.7

Totale kosten * Wlz ZiN ratio = ZiN gerelateerde Wlz kosten 900 * 0.7 = 630

ZiN gerelateerde Wlz kosten – NHC & NIC opbrengsten zorgaanbieder = ZiN gerelateerde Wlz kosten minus NHC & NIC opbrengsten:
630 – (30*1) – (25*0,20) = 595

(∑(NZa maximumtarief excl NHC en NIC per prestatie * aantal gedeclareerd per prestatie) = 100% NZa maximum scenario excl. NHC en NIC:
(30*9) + (25*7,80) + (60*5) = 765

ZiN gerelateerde Wlz kosten minus NHC & NIC/ 100% NZa maximumtarief excl. NHC & NIC = Kostenpercentage zorgaanbieder:
595/765 = 0,778 =77,8%

Deel 3 – bepaling cluster percentage/vertrektarief

Toelichting: In deel 3 van het model wordt op basis van de kostenpercentages per zorgaanbieder per cluster het vertrektarief bepaald door vast te stellen of het percentage naar aantallen zorgaanbieders of naar marktaandeel hoger is. Onderstaand een illustratief voorbeeld aan de hand van fictieve zorgaanbieders.

Zorgaanbieder	Cluster	Kostenpercentage	Wlz ZiN omzet
A	V&V C	77,8%	€700,-
B	V&V B	90,1%	€1200,-
C	V&V C	93,4%	€8000,-
D	V&V A	88,7%	€450,-
E	V&V B	98,3%	€800,-
F	V&V C	97,6%	€950,-
G	V&V C	96,1%	€1050,-

Filteren op cluster (V&V C)

Zorgaanbieder	Cluster	Kostenpercentage	Wlz ZiN omzet
A	V&V C	77,8%	€700,-
C	V&V C	93,4%	€8000,-
F	V&V C	97,6%	€950,-
G	V&V C	96,1%	€1050,-

Bepaling 75ste percentiel na sortering van laag naar hoog

Zorgaanbieder	Cluster	Kostenpercentage	
A	V&V C	77,8%	
C	V&V C	93,4%	50ste percentiel
G	**V&V C**	**96,1%**	**75ste percentiel**

F	V&V C	97,6%	

Bepaling 75% marktaandeel na sortering van laag naar hoog

Zorgaanbieder	Cluster	Kosten-percentage	Marktaandeel	Marktaandeel cumulatief
A	V&V C	77,8%	6,5%	6,5%
C	**V&V C**	**93,4%**	**74,8%**	**81,3%**
G	V&V C	96,1%	8,9%	90,2%
F	V&V C	97,6%	9,8%	100%

Vergelijking vertrektarieven cluster 75ste percentiel aantallen & 75% marktaandeel:

Zorgaanbieder	vertrektarief
75ste percentiel	96,1%
75% Marktaandeel	93,4%
Vertrektarief V&V	**96,1%**

3. Het vertrektarief per cluster voor 2027

Hier volgt bij de publicatie van de vertrektarieven op 4 september een overzicht van de vertrektarieven per sector per cluster.

4. Resultaten per cluster voor 2027

Voor alle clusters geven we bij de publicatie van de vertrektarieven op 4 september 2026 een overzicht van de belangrijkste resultaten. Hiervoor presenteren we achtereenvolgens:

a. De verdeling van het kostenpercentage per zorgaanbieder;
b. Het resultaat per zorgaanbieder: geeft inzicht in de marge van zorgaanbieders bij het vertrektarief. Hiervoor is de omzet die een zorgaanbieder behaalt op basis van het vertrektarief vergeleken met het kostenpercentage van zorgaanbieders;
c. De verdeling van het resultaat over alle zorgaanbieders.

Het vertrektarief van ieder cluster is bepaald op basis van de zorgaanbieders die meegenomen zijn in deel 2 van het model. Dit is exclusief de outliers.

Bijlage 7.a: onderscheid per prestatie leveringsvorm

Voor bepaling vertrektarief van 2027

Zie bijgevoegd Excel document “leveringsvorm prestaties 2025 Wlz.xslx”.

Bijlage 7.b: Python code

Inlezen_data

```
# Databricks notebook source
# MAGIC %md
# MAGIC # Voorbereiding
# MAGIC

# COMMAND ----------

pip install openpyxl

# COMMAND ----------

# %reset -f

## packages inladen
import numpy as np
import pandas as pd

## gebruiker en werkmap definiëren
gebruiker = <Bestandspad>

filepath_input = gebruiker + "Input/"
filepath_output = gebruiker + "Output/"

# COMMAND ----------

# MAGIC %md
# MAGIC # Bestanden inlezen

# COMMAND ----------

# MAGIC %md
# MAGIC ### Ruwe data

# COMMAND ----------
```

```
# we lezen per zorgkantoor de drie tabbladen uit de Excel-bestanden in
dict_cz = pd.read_excel(<Bestandspad CZ>, skiprows = [1], sheet_name = None)
dict_dsw = pd.read_excel(<Bestandspad DSW>, skiprows = [1], sheet_name = None)
dict_menzis = pd.read_excel(<Bestandspad Menzis>, skiprows = [1], sheet_name = None)
dict_salland = pd.read_excel(<Bestandspad Salland>, skiprows = [1], sheet_name = None)
dict_vgz = pd.read_excel(<Bestandspad VGZ>, skiprows = [1], sheet_name = None)
dict_zk = pd.read_excel(<Bestandspad ZK>, skiprows = [1], sheet_name = None)
dict_zz = pd.read_excel(<Bestandspad ZZ>, skiprows = [1], sheet_name = None)

# COMMAND ----------

# we lezen het Intrakoop-bestand in
df_intrakoop = pd.read_excel(filepath_input + <Bestandspad intrakoop>)

# COMMAND ----------

# MAGIC %md
# MAGIC ### Koppeltabellen

# COMMAND ----------

# we gebruiken een koppeltabel per prestatie. doel hiervan is om declaratie-eenheden om te zetten van alle prestaties
# om te zetten naar 24 uur, voor mogelijk later gebruik. we maakten een tabel waarin per prestatie en voor zorgkantoorcategorieën
# een vermenigvuldigingsfactor per prestatie beschikbaar is.

# eerst laden we de koppeltabel in.
df_kpl_prestaties = pd.read_excel(filepath_input + "Koppeltabel prestaties - ZN Spoor 2.xlsx",
sheet_name = "Koppeltabel prestatiecodes")
df_kpl_zorgkantoren = pd.read_excel(filepath_input + "Koppeltabel prestaties - ZN Spoor 2.xlsx",
sheet_name = "Koppeltabel zorgkantoren ehd")

# vervolgens maken we een lange koppeltabel waarin we ook onderscheid maken naar zorgkantoorcategorieën
# deze groepen zorgkantoren hadden een verschil in declaratie-eenheid. daar corrigeren we later voor,
# maar dat willen we ook in deze tabel meenemen.
df_tmp1 = df_kpl_prestaties[["Prestatiecode", "Declaratieeenheid","Factor_ZZ_Sal",
"Factor_DSW_CZ_Menzis"]].melt(
   id_vars = ["Prestatiecode", "Declaratieeenheid"],
```

```
  var_name = "Kolomnaam",
  value_name = "conversiefactor"
)

# we hebben nu een tabel voor de zorgkantoorcategorieën, maar dit willen we aanpassen naar een tabel per zorgkantoor
# we nemen daarom een tabel als basis en koppelen daaraan de waarden per zorgkantoorcategorie
df_kpl_prestaties_lang = pd.merge(df_kpl_zorgkantoren, df_tmp1, on = "Kolomnaam", how = "left").drop(columns = ["Kolomnaam"]).rename(columns = {
  "Zorgkantoor" : "zorgkantoor",
  "Prestatiecode" : "prestatiecode",
})

# COMMAND ----------

# we maken daarnaast een tabel met de max-tarieven per prestatiecode
df_nza_max = df_kpl_prestaties[["Prestatiecode", "nza_max_tarief_2024", "nza_max_tarief_2025"]]

# COMMAND ----------

# MAGIC %md
# MAGIC # Een tabel maken per type uitvraag (Excel-sheet)

# COMMAND ----------

# MAGIC %md
# MAGIC ### Sheets combineren

# COMMAND ----------

# lege dictionary maken
dict_sheets = {}

# loop maken over de dictionary per zorgkantoor
# vanuit iedere dictionary halen we dezelfde sheets
# we voegen vervolgens toe van welk zorgkantoor de data komen
# daarna verbinden we de sheets
for naam in dict_menzis.keys():
  df_cz = dict_cz[naam].copy()
  df_dsw = dict_dsw[naam].copy()
```

```
    df_menzis = dict_menzis[naam].copy()
    df_salland = dict_salland[naam].copy()
    df_vgz = dict_vgz[naam].copy()
    df_zk = dict_zk[naam].copy()
    df_zz = dict_zz[naam].copy()

    df_cz["Zorgkantoor"] = "cz"
    df_dsw["Zorgkantoor"] = "dsw"
    df_menzis["Zorgkantoor"] = "menzis"
    df_salland["Zorgkantoor"] = "salland"
    df_vgz["Zorgkantoor"] = "vgz"
    df_zk["Zorgkantoor"] = "zk"
    df_zz["Zorgkantoor"] = "zz"

    dict_sheets[naam] = pd.concat([df_cz, df_dsw, df_menzis, df_salland, df_vgz, df_zk, df_zz],
ignore_index = True)

# we halen de tabellen uit de dictionary en slaan deze op als bron
df_prestaties_bron = dict_sheets["Prestaties aanbieders"]
df_lumpsum_bron = dict_sheets["Lumpsum gegevens aanbieders"]

# COMMAND ----------

# vanaf hier gaan we verder obv een kopie
df_prestaties = df_prestaties_bron.copy()
df_lumpsum = df_lumpsum_bron.copy()

# COMMAND ----------

# MAGIC %md
# MAGIC # Checkpoint: verschillende statistieken

# COMMAND ----------

# aantal rijen per zorgkantoor
som1 = df_prestaties.groupby("Zorgkantoor").size()
som2 = df_lumpsum.groupby("Zorgkantoor").size()
pd.concat(
    [som1, som2],
    axis = 1,
```

```
    keys = ["pres", "lumpsum"]
)

# COMMAND ----------

# aantal unieke kvk's per zorgkantoor
som1 = df_prestaties.groupby("Zorgkantoor")["KVK"].nunique()
som2 = df_lumpsum.groupby("Zorgkantoor")["KVK"].nunique()
pd.concat(
    [som1, som2],
    axis = 1,
    keys = ["pres", "lumpsum"]
)

# COMMAND ----------

# aantal unieke kvk-nummers in totaal
som1 = df_prestaties["KVK"].nunique()
som2 = df_lumpsum["KVK"].nunique()
print("prestaties:", som1, "lumpsum:", som2)

# COMMAND ----------

# totaal vergoed bedrag per zorgkantoor (prestatie- en lumpsumtabblad)
som1 = df_prestaties[pd.to_numeric(df_prestaties['Totaal vergoed bedrag'],
errors='coerce').notna()].groupby("Zorgkantoor")["Totaal vergoed bedrag"].sum()
som2 = df_lumpsum[pd.to_numeric(df_lumpsum['Totaal vergoed bedrag'],
errors='coerce').notna()].groupby("Zorgkantoor")["Totaal vergoed bedrag"].sum()
pd.concat(
    [som1, som2],
    axis = 1,
    keys = ["pres", "lumpsum"]
).round(2)

# COMMAND ----------

# MAGIC %md
# MAGIC # Opschonen van tabellen

# COMMAND ----------
```

```
# MAGIC %md
# MAGIC ### Kolomnamen opschonen

# COMMAND ----------

# eerst zetten we alle kolomnamen om naar kleine letters
df_prestaties.columns = df_prestaties.columns.str.lower()
df_lumpsum.columns = df_lumpsum.columns.str.lower()

# we passen de kolomnamen aan
df_prestaties = df_prestaties.rename(columns = {"nza nummer" : "nza_nummer",
                                    "totaal aantal verrichtingen" : "aantal_verrichtingen",
                                    "totaal vergoed bedrag" : "vergoed_bedrag",
                                    })
df_lumpsum = df_lumpsum.rename(columns = {"nza nummer" : "nza_nummer",
                                "totaal vergoed bedrag" : "vergoed_bedrag",
                                "totaalbedrag correcties" : "correctiebedrag",
                                "totaal bedrag uit vv-indicaties":"vv_bedrag",
                                "totaal bedrag uit gz-indicaties":"gz_bedrag",
                                "totaal bedrag uit ggz-indicaties":"ggz_bedrag"})

# COMMAND ----------

# MAGIC %md
# MAGIC ### Kolommen achterwege laten

# COMMAND ----------

# we laten in alle gevallen de kolom variabele buiten beschouwing - was bedoeld voor toelichting, niet voor analyse
# voor de lumpsumtabel laten we daarnaast de zorgkantoorcodekolom achterwege - geen onderdeel uitvraag, enkel door z&z ingevuld
df_prestaties = df_prestaties.drop(columns = ["variabele"])
df_lumpsum = df_lumpsum.drop(columns = ["variabele"])

# COMMAND ----------

# MAGIC %md
# MAGIC ### Aantal verrichtingen omzetten in prestatietabel
```

```
# COMMAND ----------

# we vervangen het aantal verrichtingen dat kleiner is dan 10 door 5
df_prestaties["aantal_verrichtingen"] = df_prestaties["aantal_verrichtingen"].replace(["<10", "< 10"], 5)

# we koppelen de conversiefactoren aan de prestaties
df_prestaties = pd.merge(df_prestaties, df_kpl_prestaties_lang, on = ["zorgkantoor", "prestatiecode"],
how = "left")

display(df_prestaties)

# COMMAND ----------

# we voeren hier bewerkingen uit
# aantal verrichtingen omzetten naar 24-uurs-declaratie-eenheid mbv eerder gemaakte koppeltabel
# bij het ene zorgkantoor en per minuut bij het andere zorgkantoor voor eenzelfde prestatie)

# eerst zetten we het aantal verrichtingen om naar numerieke waarden
df_prestaties["aantal_verrichtingen"] = pd.to_numeric(df_prestaties["aantal_verrichtingen"], errors =
"coerce")

# aantal verrichitngen omzettenmet koppeltabel
df_prestaties["aantal_verrichtingen_in_etmaal"] = df_prestaties["aantal_verrichtingen"] *
df_prestaties["conversiefactor"].fillna(1)

# COMMAND ----------

# we controleren het aantal verrichtingen kleiner dan 10
# 5 komt vaak voor omdat we dit als vervangende waarde gebruiken van < 10
# andere waarden onder 10 komen voor door bewerkingen op de data
df_prestaties_klein = df_prestaties[df_prestaties["aantal_verrichtingen"] < 10]
df_prestaties_klein["aantal_verrichtingen"].plot.hist(bins = 10, edgecolor = "black")

# COMMAND ----------

# ter controle: nieuw overzicht van kvk-nummers geven
som1 = df_prestaties.groupby("zorgkantoor")["kvk"].nunique()
som2 = df_lumpsum.groupby("zorgkantoor")["kvk"].nunique()
pd.concat(
```

```
  [som1, som2],
  axis = 1,
  keys = ["pres", "lumpsum"]
)

# COMMAND ----------

# MAGIC %md
# MAGIC ### Aanpassen van KVK-codes

# COMMAND ----------

# er zijn drie aanbieders zonder KVK-nummer
df_prestaties = df_prestaties[((df_prestaties["kvk"].notna()) & (df_prestaties["kvk"] != 0))]
df_lumpsum = df_lumpsum[((df_lumpsum["kvk"].notna()) & (df_lumpsum["kvk"] != 0))]

# KVK-nummers met minder dan 7 of meer dan 8 cijfers worden als ongeldig beschouwd. om te controleren of deze er zijn moeten we de waarden eerst omzetten naar een string.
# daarvoor moeten we de waarden omzetten naar een geheel getal, om te voorkomen dat punt 0 wordt meegenomen in de string.
df_prestaties["kvk"] = df_prestaties["kvk"].astype('Int64').astype(str)
df_lumpsum["kvk"] = df_lumpsum["kvk"].astype('Int64').astype(str)

# COMMAND ----------

# vervolgens kunnen we het KVK-nummer omzetten naar een achtcijferige code
df_prestaties["kvk"] = df_prestaties["kvk"].str.zfill(8)
df_lumpsum["kvk"] = df_lumpsum["kvk"].str.zfill(8)

# COMMAND ----------

# check: aantal unieke KVK's per zorgkantoor
som1 = df_prestaties.groupby("zorgkantoor")["kvk"].nunique()
som2 = df_lumpsum.groupby("zorgkantoor")["kvk"].nunique()
pd.concat(
  [som1, som2],
  axis = 1,
  keys = ["pres", "lumpsum"]
)
```

```
# COMMAND ----------

# MAGIC %md
# MAGIC ### Aanpassen van NZa-nummers

# COMMAND ----------

# we zien dat een deel van de NZa-nummers een liggend streepje heeft na de eerste drie cijfers. deze halen we weg
df_prestaties["nza_nummer"] = df_prestaties["nza_nummer"].astype("string").str.replace("-", "", regex = False).astype("float").astype("Int64")
df_lumpsum["nza_nummer"] = df_lumpsum["nza_nummer"].astype("string").str.replace("-", "", regex = False).astype("float").astype("Int64")

# COMMAND ----------

# MAGIC %md
# MAGIC # Checkpoint: controle op juiste declaratie-eenheid

# COMMAND ----------

# we voerden een controle uit om te bekijken of het vergoed bedrag per verrichting per prestatie t.o.v. nza-max ongeveer uitkwam op logische
# waarden (ongeveer richttariefpercentages). daarvoor gebruikten we onderstaande berekeningen

# we onderzoeken 2025 en nemen het aantal_verrichtingen dat gelijk is aan 5 (merendeel daarvan volgt uit handmatige aanpassing door onszelf)
df_pres_agg = df_prestaties[(df_prestaties["aantal_verrichtingen"] != 5)].groupby(["zorgkantoor", "prestatiecode"], as_index = False).agg(
   {"aantal_verrichtingen" : "sum",
    "vergoed_bedrag" : "sum"}
)

# berekenen "afgesproken tarief"
df_pres_agg["afgesproken_tar"] = df_pres_agg["vergoed_bedrag"] /
df_pres_agg["aantal_verrichtingen"]

# tabel pivoten en nza-max koppelen
df_pres_agg_wd = df_pres_agg.pivot(index = "prestatiecode", columns = "zorgkantoor", values = "afgesproken_tar")
```

```
df_kpl_prestaties_tmp = df_kpl_prestaties[["Prestatiecode", "nza_max_tarief_2025"]].rename(columns = {"Prestatiecode" : "prestatiecode"})
df_delta_tar = pd.merge(df_pres_agg_wd, df_kpl_prestaties_tmp, on = "prestatiecode", how = "left")

# tabel maken waarin we "richttariefpercentage" per zorgkantoor berekenen
df_delta = df_delta_tar.copy()
df_delta["rtp_cz"] = df_delta["cz"] / df_delta["nza_max_tarief_2025"]
df_delta["rtp_dsw"] = df_delta["dsw"] / df_delta["nza_max_tarief_2025"]
df_delta["rtp_menzis"] = df_delta["menzis"] / df_delta["nza_max_tarief_2025"]
df_delta["rtp_salland"] = df_delta["salland"] / df_delta["nza_max_tarief_2025"]
df_delta["rtp_vgz"] = df_delta["vgz"] / df_delta["nza_max_tarief_2025"]
df_delta["rtp_zk"] = df_delta["zk"] / df_delta["nza_max_tarief_2025"]
df_delta["rtp_zz"] = df_delta["zz"] / df_delta["nza_max_tarief_2025"]

# COMMAND ----------

# MAGIC %md
# MAGIC # NZa-max berekenen

# COMMAND ----------

# MAGIC %md
# MAGIC ### Jaar selecteren

# COMMAND ----------

jaar_num = 2025
jaar_txt = "2025"

# COMMAND ----------

# MAGIC %md
# MAGIC ### Prestaties filteren op relevantie

# COMMAND ----------

# codes uit analyse inladen
df_codes_basis = pd.read_excel(filepath_input + "Prestaties (incorrect NZa-max).xlsx", sheet_name = "meerzorg")
```

```
# uitbreiden met prestaties zonder max-tarief
df_nza_max_0 = df_nza_max[(df_nza_max["nza_max_tarief_" + jaar_txt] == 0) |
(df_nza_max["nza_max_tarief_" + jaar_txt].isna())]
df_codes = pd.concat([df_codes_basis["Prestatiecode"], df_nza_max_0["Prestatiecode"]], ignore_index
= True).drop_duplicates()

# prestatietabel filteren op de relevante prestaties
df_prestaties_tmp = df_prestaties[(~df_prestaties["prestatiecode"].isin(df_codes)) &
                        (df_prestaties["vergoed_bedrag"].notna()) &
                        (df_prestaties["vergoed_bedrag"] != 0)]

# COMMAND ----------

# MAGIC %md
# MAGIC ### Tariefpercentage berekenen

# COMMAND ----------

# nza-max koppelen
df_kpl_prestaties_tmp = df_kpl_prestaties[["Prestatiecode", "nza_max_tarief_" +
jaar_txt]].rename(columns = {"Prestatiecode" : "prestatiecode"})
df_prestaties_tmp = pd.merge(df_prestaties_tmp, df_kpl_prestaties_tmp, on = "prestatiecode", how =
"left")

# nza-max-totaal per prestatie berekenen
df_prestaties_tmp["nza_max"] = df_prestaties_tmp["aantal_verrichtingen"] *
df_prestaties_tmp["nza_max_tarief_" + jaar_txt]

# aggregeren naar aanbieder-niveau
df_prestaties_tmp_agg = df_prestaties_tmp.groupby(["kvk", "zorgkantoor"], as_index = False).agg(
   {"vergoed_bedrag" : "sum",
    "nza_max" : "sum"}
)

# tariefpercentage berekenen
df_prestaties_tmp_agg["tar_pct"] = df_prestaties_tmp_agg["vergoed_bedrag"] /
df_prestaties_tmp_agg["nza_max"]

# tabel kvk-tariefpercentage maken
df_prestaties_tmp_agg = df_prestaties_tmp_agg[["kvk", "zorgkantoor", "tar_pct"]]
```

```
# COMMAND ----------

# MAGIC %md
# MAGIC ### NZa-max totaal berekenen

# COMMAND ----------

# tariefpercentage koppelen
df_nza_max_exp = pd.merge(df_lumpsum, df_prestaties_tmp_agg, on = ["kvk", "zorgkantoor"], how =
"left")

# NZa-max-totaal berekenen
df_nza_max_exp["nza_max"] = df_nza_max_exp["vergoed_bedrag"] / df_nza_max_exp["tar_pct"]

# groeperen per aanbieder
df_nza_max_agg = df_nza_max_exp.groupby(["kvk"], as_index = False).agg(
   {"vergoed_bedrag" : "sum",
   "nza_max" : "sum"}
)

# COMMAND ----------

df_nza_max_agg["check"] = df_nza_max_agg["vergoed_bedrag"] / df_nza_max_agg["nza_max"]

# COMMAND ----------

# MAGIC %md
# MAGIC # Checkpoint: verschillende statistieken

# COMMAND ----------

# aantal rijen per zorgkantoor
som1 = df_prestaties.groupby("zorgkantoor").size()
som2 = df_lumpsum.groupby("zorgkantoor").size()
pd.concat(
   [som1, som2],
   axis = 1,
   keys = ["pres", "lumpsum"]
)
```

```
# aantal unieke kvk's per zorgkantoor
som1 = df_prestaties.groupby("zorgkantoor")["kvk"].nunique()
som2 = df_lumpsum.groupby("zorgkantoor")["kvk"].nunique()
pd.concat(
   [som1, som2],
   axis = 1,
   keys = ["pres", "lumpsum"]
)

# aantal unieke kvk's
som1 = df_prestaties["kvk"].nunique()
som2 = df_lumpsum["kvk"].nunique()
print("prestaties:", som1, "lumpsum:", som2)

# vergoed bedrag per zorgkantoor
som1 = df_prestaties.groupby("zorgkantoor")["vergoed_bedrag"].sum()
som3 = df_lumpsum.groupby("zorgkantoor")["vergoed_bedrag"].sum()
pd.concat(
   [som1, som3],
   axis = 1,
   keys = ["pres", "lumpsum"]
).round(2)

# COMMAND ----------

# MAGIC %md
# MAGIC # Tabellen omzetten naar analyseformat

# COMMAND ----------

df_zk_temp = df_lumpsum[['kvk','organisatienaam', 'zorgkantoor']].groupby(['kvk', 'zorgkantoor']).agg({
   'organisatienaam': 'first'
}).reset_index()
display(df_zk_temp)

df_zk_temp.to_excel(filepath_output + "kvk_per_zk.xlsx")

# COMMAND ----------
```

```
# prestaties
df_prestaties_eind = df_prestaties.groupby(
   ["kvk", "prestatiecode"], as_index = False, dropna = False
)[["aantal_verrichtingen","aantal_verrichtingen_in_etmaal", "vergoed_bedrag"]].sum()

# lumpsum
df_lumpsum_eind = df_lumpsum.groupby(
   ["kvk"], as_index = False, dropna = False
)[["vergoed_bedrag", "vv_bedrag", "gz_bedrag", "ggz_bedrag"]].sum()

# nza-max
df_nza_max_eind = df_nza_max_agg.copy()

# COMMAND ----------

# MAGIC %md
# MAGIC ### Checkpoint: verschillende statistieken

# COMMAND ----------

# aantal unieke kvk-nummers
som1 = df_prestaties_eind["kvk"].nunique()
som2 = df_lumpsum_eind["kvk"].nunique()
print("prestaties:", som1,"lumpsum:", som2)

# vergoed bedrag (op prestatie- en lumpsumtabblad)
som1 = df_prestaties_eind["vergoed_bedrag"].sum()
som2 = df_lumpsum_eind["vergoed_bedrag"].sum()
print(som1, som2)

# COMMAND ----------

# MAGIC %md
# MAGIC # Gegevens wegschrijven

# COMMAND ----------

# bestand met alle drie de tabellen
with pd.ExcelWriter(filepath_output + "totale_datasets_2025.xlsx") as writer:
   df_prestaties_eind.to_excel(writer, sheet_name = "prestaties", index = False)
```

```
    df_lumpsum_eind.to_excel(writer, sheet_name = "lumpsum", index = False)

# COMMAND ----------

# tabel met nza-max waarden
# LET OP: vul hier het juiste jaar in
with pd.ExcelWriter(filepath_output + "nza_max_tabel_" + jaar_txt + ".xlsx") as writer:
    df_nza_max_eind.to_excel(writer, sheet_name = "nza_max", index = False)

data verwerken
# Databricks notebook source
# MAGIC %md
# MAGIC # Spoor 2 - Verwerken data
# MAGIC

# COMMAND ----------

# MAGIC %md
# MAGIC ## Voorbereiding

# COMMAND ----------

pip install openpyxl

# COMMAND ----------

# DBTITLE 1,Imports & Magic numbers
#Clear variabelen
# %reset -f

# Imports
import numpy as np
import pandas as pd
import warnings
warnings.simplefilter(action='ignore', category=pd.errors.SettingWithCopyWarning)

## gebruiker en werkmap definiëren
gebruiker = <Bestandspad>

filepath_input = gebruiker + "Input/"
```

```
filepath_output = gebruiker + "Output/"

# COMMAND ----------

# MAGIC %md
# MAGIC ### Importeren van bestanden

# COMMAND ----------

# Prestaties
df_wlz_declaraties = pd.read_excel(filepath_output + 'totale_datasets_2025.xlsx',
sheet_name='prestaties')

# Totaal bedragen
df_wlz_totalen = pd.read_excel(filepath_output + 'totale_datasets_2025.xlsx',
sheet_name='lumpsum').rename(columns = {"vergoed_bedrag" : "vergoed_bedrag_totaal"})

# NZa-max
df_nza_max = pd.read_excel(filepath_output + 'nza_max_tabel_2025.xlsx').rename(columns =
{"vergoed_bedrag" : "vergoed_bedrag_totaal"})

# COMMAND ----------

# aanbieder met geen NZa-max eruit laten
df_nza_max = df_nza_max[df_nza_max["nza_max"] != 0]

# COMMAND ----------

# Intrakoop - lees relevante kolommen in met directe conversie van bedragen
intrakoop_path = filepath_input + <Bestandspad intrakoop>

intrakoop_kolommen = [
    'KVK',
    'Organisatie',
    'Som der bedrijfsopbrengsten Verslagjaar',
    'Som der bedrijfsopbrengsten Vergelijkingsjaar',
    'Som der bedrijfslasten Verslagjaar',
    'Som der bedrijfslasten Vergelijkingsjaar',
    'Totaal financiële baten en lasten Verslagjaar',
    'Totaal financiële baten en lasten Vergelijkingsjaar'
```

```
]

df_intrakoop = pd.read_excel(
   intrakoop_path,
   usecols=intrakoop_kolommen,
   dtype={'KVK': str},
   thousands='.',
   decimal=','
)

# Hernoemen van kolommen naar snake_case
df_intrakoop.columns = [
   'organisatie',
   'kvk',
   'som_der_bedrijfsopbrengsten_verslagjaar',
   'som_der_bedrijfsopbrengsten_vergelijkingsjaar',
   'som_der_bedrijfslasten_verslagjaar',
   'som_der_bedrijfslasten_vergelijkingsjaar',
   'totaal_financiële_baten_en_lasten_verslagjaar',
   'totaal_financiële_baten_en_lasten_vergelijkingsjaar'
]

df_intrakoop['kvk'] = pd.to_numeric(df_intrakoop['kvk'], errors='coerce').astype('Int64')

# Dubbele regels verwijderen (op alle kolommen)
df_intrakoop.drop_duplicates(inplace=True)

# COMMAND ----------

# we willen het Intrakoop-bestand koppelen aan de prestaties. het bestand bevat per rij gegevens over de zorgaanbieders.
# we willen dit omvormen naar een tabel waarin de rijen unieke combinaties zijn van aanbieder en jaartal zodat we deze kunnen
# koppelen.

# organisatienaam achterwege laten
df_intrakoop_copy = df_intrakoop.drop(columns= {"organisatie"})

# Intrakoop-tabel pivoten voor eerste stap
```

```
df_intrakoop_lang = df_intrakoop_copy.melt(id_vars = "kvk", var_name = "variabele", value_name =
"waarde")

# liggend streepjes weghalen (is missende waarde)
df_intrakoop_lang["waarde"] = df_intrakoop_lang["waarde"].replace("-", np.nan)

# kolom toevoegen voor een jaartal
df_intrakoop_lang["jaar"] = np.where(df_intrakoop_lang["variabele"].str.contains("_verslagjaar", na =
False),
                                     2025,
                                     2024)

# naam van variabelen aanpassen (verslagjaar en vergelijkingsjaar verwijderen)
df_intrakoop_lang["variabele"] = (
   df_intrakoop_lang["variabele"]
   .str.replace("_verslagjaar", "")
   .str.replace("_vergelijkingsjaar", "")
)

# Intrakoop-tabel weer wijd maken
df_intrakoop_wijd = (
   df_intrakoop_lang
   .pivot(index = ["kvk", "jaar"], columns = "variabele", values = "waarde")
   .reset_index()
)

# COMMAND ----------

# Koppeltabel prestaties
df_koppel_prestaties = pd.read_excel(filepath_input + 'Koppeltabel prestaties - ZN Spoor 2.xlsx',
sheet_name='Koppeltabel prestatiecodes')
df_koppel_prestaties =
df_koppel_prestaties[['Prestatiecode','Zorgsoort2','Declaratieeenheid','nza_max_tarief_2025']]
df_koppel_prestaties.columns = ['prestatiecode',
'zorgsoort2','declaratieeenheid','nza_max_tarief_2025']

# intramurale indicatorkolom obv sectoroverstijgende aanduidingen per prestatie
df_koppel_zzp_intramuraal = pd.read_excel(filepath_input +
'Koppeltabel_Intramuraal_Extramuraal.xlsx').rename(columns= {"code" : "prestatiecode",
"Intramuraal" : "intramuraal"})
```

```
df_koppel_prestaties = pd.merge(
   df_koppel_prestaties, df_koppel_zzp_intramuraal, on = "prestatiecode", how = "left"
)

df_koppel_zzp_intramuraal = df_koppel_zzp_intramuraal.drop_duplicates()

# COMMAND ----------

# Koppeltabel organisatienamen
df_koppel_organisatienaam = pd.read_excel(filepath_input + 'koppeltabel_organisatienamen.xlsx')

# COMMAND ----------

# MAGIC %md
# MAGIC ### Mergen tabellen

# COMMAND ----------

# MAGIC %md
# MAGIC #### Prestaties

# COMMAND ----------

# DBTITLE 1,Data import & Preparatie
# Mergen van tabellen
df_wlz_declaraties_full = df_wlz_declaraties.merge(df_koppel_prestaties, how='left',
on='prestatiecode')

# kopie maken
df_wlz_declaraties_full_copy = df_wlz_declaraties_full.copy()

# COMMAND ----------

# MAGIC %md
# MAGIC ### Toevoegen variabelen

# COMMAND ----------

# MAGIC %md
# MAGIC #### Prestaties
```

```
# COMMAND ----------

# vpt en mpt onderscheiden
df_wlz_declaraties_full['vpt_verricht'] = np.where(df_wlz_declaraties_full['zorgsoort2'] == 'VPT',
df_wlz_declaraties_full['aantal_verrichtingen'], 0)
df_wlz_declaraties_full['mpt_verricht'] = np.where(df_wlz_declaraties_full['zorgsoort2'] == 'MPT',
df_wlz_declaraties_full['aantal_verrichtingen'], 0)
df_wlz_declaraties_full['overige_zorg_verricht'] = np.where(df_wlz_declaraties_full['zorgsoort2'] ==
'Overig', df_wlz_declaraties_full['aantal_verrichtingen'], 0)
df_wlz_declaraties_full['vpt_bedrag'] = np.where(df_wlz_declaraties_full['zorgsoort2'] == 'VPT',
df_wlz_declaraties_full['vergoed_bedrag'], 0)
df_wlz_declaraties_full['mpt_bedrag'] = np.where(df_wlz_declaraties_full['zorgsoort2'] == 'MPT',
df_wlz_declaraties_full['vergoed_bedrag'], 0)
df_wlz_declaraties_full['overige_zorg_bedrag'] = np.where(df_wlz_declaraties_full['zorgsoort2'] ==
'Overig', df_wlz_declaraties_full['vergoed_bedrag'], 0)

# intramuraal
df_wlz_declaraties_full['zzp_intramuraal_verricht'] = np.where(df_wlz_declaraties_full['intramuraal'] ==
'Ja', df_wlz_declaraties_full['aantal_verrichtingen'], 0)
df_wlz_declaraties_full['zzp_intramuraal_bedrag'] = np.where(df_wlz_declaraties_full['intramuraal'] ==
'Ja', df_wlz_declaraties_full['vergoed_bedrag'], 0)

# COMMAND ----------

# MAGIC %md
# MAGIC ### Benchmarkingtabel maken

# COMMAND ----------

# benchmarkingtabel initiëren
df_benchmarking = df_nza_max.copy()

# kvk-nummers numeriek maken
df_benchmarking["kvk"] = pd.to_numeric(df_benchmarking["kvk"], errors='coerce').astype('Int64')

# COMMAND ----------

# joinen van relevante tabellen
df_intrakoop_wijd = df_intrakoop_wijd[df_intrakoop_wijd['jaar'] == 2025]
```

```
df_benchmarking = pd.merge(df_benchmarking, df_intrakoop_wijd, on = ["kvk"], how = "left")

# COMMAND ----------

# omzetten van liggende streepjes in Intrakoop-data naar missende waarden
df_benchmarking["som_der_bedrijfsopbrengsten"] =
df_benchmarking["som_der_bedrijfsopbrengsten"].replace("-", np.nan)
df_benchmarking["som_der_bedrijfslasten"] = df_benchmarking["som_der_bedrijfslasten"].replace("-",
np.nan)
df_benchmarking["totaal_financiële_baten_en_lasten"] =
df_benchmarking["totaal_financiële_baten_en_lasten"].replace("-", np.nan)

# COMMAND ----------

# MAGIC %md
# MAGIC #### Kostenpercentages berekenen

# COMMAND ----------

# functie kostenpercentageberekening
def kostenberekening(df: pd.DataFrame) -> pd.DataFrame:
    """
    Bereken WLZ-kosten en ratio's.
    """
    # Zorg dat alle relevante kolommen numeriek zijn
    num_cols = [
        'vergoed_bedrag_totaal',
        'nza_max',
        'som_der_bedrijfsopbrengsten',
        'som_der_bedrijfslasten',
        'totaal_financiële_baten_en_lasten'
    ]
    for col in num_cols:
        df[col] = pd.to_numeric(df[col], errors='coerce')

    # Berekeningen
    df['wlz_ratio'] = df['vergoed_bedrag_totaal'] / df['som_der_bedrijfsopbrengsten']
    df['Totale_kosten'] = (
        df['som_der_bedrijfslasten'] - df['totaal_financiële_baten_en_lasten']
```

```
    )
    df['wlz_kosten'] = df['Totale_kosten'] * df['wlz_ratio']
    df['kosten_ratio'] = df['wlz_kosten'] / df['nza_max']

    return df

# COMMAND ----------

# functie toepassen
df_benchmarking_kpct = kostenberekening(df_benchmarking)

# COMMAND ----------

# MAGIC %md
# MAGIC ### Filters

# COMMAND ----------

jaar = 2025
jaar_txt = '2025'

df_wlz_declaraties_full_2025 = df_wlz_declaraties_full
df_wlz_totalen_2025 = df_wlz_totalen


# COMMAND ----------

# MAGIC %md
# MAGIC ## Wegschrijven bestanden
# MAGIC
# MAGIC Noot: vergeet niet de bestanden die je overschrijft eerst in het archief op te slaan met datum.

# COMMAND ----------

df_wlz_declaraties_full_2025.to_csv(filepath_output + 'df_wlz_declaraties_full_'+ jaar_txt + '.csv',
index=False)
df_wlz_totalen_2025.to_csv(filepath_output + 'df_wlz_totalen_'+ jaar_txt + '.csv', index=False)
df_benchmarking_kpct.to_csv(filepath_output + 'df_benchmarking_'+ jaar_txt + '.csv', index=False)

sector bepaling
```

```
# Databricks notebook source
# MAGIC %md
# MAGIC # Sector bepaling aanbieders o.b.v. vergoed bedrag

# COMMAND ----------

pip install openpyxl

# COMMAND ----------

#Clear variabelen

# Imports
import pandas as pd
import warnings
warnings.simplefilter(action='ignore', category=pd.errors.SettingWithCopyWarning)

## gebruiker en werkmap definiëren
gebruiker = <Bestandspad>

filepath_input = gebruiker + "Input/"
filepath_output = gebruiker + "Output/"

# COMMAND ----------

# Import
df_lumpsum_2025 = pd.read_excel(filepath_output + 'totale_datasets_2025.xlsx', sheet_name =
'lumpsum')

display(df_lumpsum_2025)

# COMMAND ----------

# Assuming the DataFrame with the amount columns is df_lumpsum_2025
# Determine the column (vv_bedrag, gz_bedrag, ggz_bedrag) that has the highest value per row
df_lumpsum_2025 = df_lumpsum_2025.copy()  # avoid SettingWithCopyWarning
df_lumpsum_2025["sector"] = (
   df_lumpsum_2025[["vv_bedrag", "gz_bedrag", "ggz_bedrag"]]
   .idxmax(axis=1)
)
```

```
# Map column names to sector codes
sector_map = {"vv_bedrag": "VV", "gz_bedrag": "GZ", "ggz_bedrag": "GGZ"}
df_lumpsum_2025["sector"] = df_lumpsum_2025["sector"].map(sector_map)

display(df_lumpsum_2025)

# COMMAND ----------

sector_per_kvk = df_lumpsum_2025[['kvk', 'sector']].copy()
sector_per_kvk.to_csv(filepath_output + 'koppeltabel_segment_per_kvk.csv', index = False)

GGZ clustering
# Databricks notebook source
# MAGIC %md
# MAGIC # Spoor 2 - clustering
# MAGIC

# COMMAND ----------

# MAGIC %md
# MAGIC ## Voorbereiding

# COMMAND ----------

pip install openpyxl

# COMMAND ----------

# DBTITLE 1,Imports & Magic numbers
# Clear variabelen
# %reset -f

# Imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('seaborn-v0_8')
import warnings
warnings.simplefilter(action = 'ignore', category = pd.errors.SettingWithCopyWarning)
```

```
# gebruiker en werkmap definiëren
gebruiker = <Bestandspad>

filepath_input = gebruiker + "Input/"
filepath_output = gebruiker + "Output/"

# COMMAND ----------

jaar = '2025'

# COMMAND ----------

# MAGIC %md
# MAGIC ### Importeren van bestanden

# COMMAND ----------

# de bestanden die volgen uit de directory 'filepath_input' zijn brongegevens. deze vormen de basis van de analyses.
# de bestanden die volgen uit de directory filepath_output' zijn uitdraaien van de eerdere codes.

# inladen prestaties
df_wlz_declaraties_full_2025 = pd.read_csv(filepath_output + 'df_wlz_declaraties_full_' + jaar + '.csv')

# inladen sectorbepaling
df_koppel_segment_per_kvk = pd.read_csv(filepath_output + 'koppeltabel_segment_per_kvk.csv')

# inladen lumpsumbedragen
df_wlz_totalen_2025 = pd.read_csv(filepath_output + 'df_wlz_totalen_' + jaar + '.csv')

# inladen koppeltabel organisatienaam
df_koppel_organisatienaam = pd.read_excel(filepath_input + 'koppeltabel_organisatienamen.xlsx')
df_koppel_organisatienaam["kvk"] = pd.to_numeric(df_koppel_organisatienaam["kvk"], errors = "coerce")

# COMMAND ----------

# MAGIC %md
# MAGIC ### Kolommen toevoegen en aggregratie per kvk
```

```
# COMMAND ----------

# Inladen data
df_base = df_wlz_declaraties_full_2025.copy()

# Filter sector
df_base = pd.merge(df_base, df_koppel_segment_per_kvk, on = 'kvk', how = 'left')
df_base = df_base[df_base['sector'] == 'GGZ'] # Nieuwe filter op GGZ- organisaties

# COMMAND ----------

# Groeperen per aanbieder
keys = ["kvk"]

agg = {
   "aantal_verrichtingen": "sum",
   "zzp_intramuraal_verricht": "sum"
}

df_base = (
   df_base.groupby(keys, as_index = False)
      .agg(agg)
   )

# COMMAND ----------

#Toevoegen omzet (wlz_totalen)
df_base = pd.merge(df_base, df_wlz_totalen_2025[['kvk', 'vergoed_bedrag_totaal']], on = 'kvk', how =
'left')



# COMMAND ----------

#Percentage intramuraal berekenen
df_base["intramuraal_aandeel"] = df_base["zzp_intramuraal_verricht"] /
df_base["aantal_verrichtingen"]

# COMMAND ----------
```

```
# MAGIC %md
# MAGIC ## Check grenswaardes

# COMMAND ----------

# MAGIC %md
# MAGIC ### Intramuraal

# COMMAND ----------

# Check clustering grens
s = pd.to_numeric(df_base["intramuraal_aandeel"], errors="coerce").dropna()

plt.figure(figsize=(8,4))
plt.hist(s, bins=100, color = "#0E3E37")  # eenvoudig: automatische bin-keuze
plt.ylabel("Aantal")
plt.title("Verhouding van intramurale prestaties [intramurale verrichtingen t.o.v. totaal aantal]")
plt.tight_layout()
plt.axvline(x = 0.01, color = "#E87200")
plt.show()

# COMMAND ----------

# MAGIC %md
# MAGIC ## Clustering

# COMMAND ----------

# MAGIC %md
# MAGIC ### 2 clusters binnen de GGZ

# COMMAND ----------

#clustering benchmarking

# - Extramuraal = A
# - (Deels) intramuraal = B

# cluster-kolom als object om problemen met NaN/strings te vermijden
df_base["cluster"] = pd.Series(pd.NA, dtype = "object")
```

```
# A - extramuraal
df_base["cluster"] = np.where(
   (df_base["intramuraal_aandeel"] < 0.01),   # Geen intramuraal ZZP
   "A",
   df_base["cluster"]
)

# B - intramuraal
df_base["cluster"] = np.where((
   df_base["cluster"].isna() &          #Niet cluster A
   (df_base["intramuraal_aandeel"] >= 0.01)),   #Wel intramuraal ZZP
   "B",
   df_base["cluster"])

# Voeg een beschrijvende kolom toe voor de clusterindeling
df_base["cluster_omschrijving"] = df_base["cluster"].map({
   "A": "Extramuraal",
   "B": "(Deels) Intramuraal"
})

assert df_base["cluster"].notna().all(), "Niet alle aanbieders zijn ingedeeld in een cluster"

# COMMAND ----------

# MAGIC %md
# MAGIC ### Output wegschrijven

# COMMAND ----------

# Kopieën
# Enkel clusterindeling
df_clusters_GGZ = df_base.copy()
df_clusters_GGZ = df_clusters_GGZ[['kvk', 'cluster', 'cluster_omschrijving']]

# Bredere informatieset
df_clusters_GGZ_output = df_base.copy()

# Join organisatienaam
```

```
df_clusters_GGZ_output = pd.merge(df_clusters_GGZ_output, df_koppel_organisatienaam, on = 'kvk', how = 'left')

# COMMAND ----------

df_kvk_per_zk = pd.read_excel(filepath_output + 'kvk_per_zk.xlsx')
df_kvk_per_zk = df_kvk_per_zk[['kvk', 'zorgkantoor', 'organisatienaam']]

display(df_kvk_per_zk)

# COMMAND ----------

# Left join df_clusters_GGZ with df_kvk_per_zk on 'kvk'
df_clusters_GGZ = df_clusters_GGZ.merge(df_kvk_per_zk, on="kvk", how="left")
df_clusters_GGZ['sector'] = 'GGZ'
display(df_clusters_GGZ)

for zk in df_clusters_GGZ['zorgkantoor'].unique():
   df_clusters_GGZ_zk = df_clusters_GGZ[df_clusters_GGZ['zorgkantoor'] == zk]
   df_clusters_GGZ_zk = df_clusters_GGZ_zk[['kvk', 'organisatienaam', 'sector', 'cluster', 'cluster_omschrijving']]
   df_clusters_GGZ_zk.to_excel(filepath_output + 'clusters/clusters_GGZ_' + jaar + '_' + zk + '.xlsx', index=False)

df_clusters_GGZ.to_csv(filepath_output + 'df_clusters_GGZ_' + jaar + '.csv',index=False)

# COMMAND ----------

#Voor juridische onderbouwing dataset
df_clusters_GGZ_output_juridische_onderbouwing = df_base.copy()

df_clusters_GGZ_output_juridische_onderbouwing['sector'] = 'GGZ'

df_clusters_GGZ_output_juridische_onderbouwing = df_clusters_GGZ_output_juridische_onderbouwing[['kvk', 'sector', 'cluster', 'cluster_omschrijving', 'intramuraal_aandeel']]

# COMMAND ----------

display(df_clusters_GGZ_output_juridische_onderbouwing)
```

```
# COMMAND ----------

df_clusters_GGZ_output_juridische_onderbouwing.to_excel(filepath_output +
'df_clusters_GGZ_output_' + jaar + '.xlsx',index=False)



GZ clustering
# Databricks notebook source
# MAGIC %md
# MAGIC # Spoor 2 - clustering
# MAGIC

# COMMAND ----------

# MAGIC %md
# MAGIC ## Voorbereiding

# COMMAND ----------

pip install openpyxl

# COMMAND ----------

# DBTITLE 1,Imports & Magic numbers
#Clear variabelen
# %reset -f

# Imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('seaborn-v0_8')
import warnings
warnings.simplefilter(action='ignore', category=pd.errors.SettingWithCopyWarning)

## gebruiker en werkmap definiëren
gebruiker = <Bestandspad>

filepath_input = gebruiker + "Input/"
```

```
filepath_output = gebruiker + "Output/"

# COMMAND ----------

jaar = '2025'

# COMMAND ----------

# MAGIC %md
# MAGIC ### Importeren van bestanden

# COMMAND ----------

# de bestanden die volgen uit de directory 'filepath_input' zijn brongegevens. deze vormen de basis van de analyses.
# de bestanden die volgen uit de directory filepath_output' zijn uitdraaien van de eerdere codes.

# inladen prestaties
df_wlz_declaraties_full_2025 = pd.read_csv(filepath_output + 'df_wlz_declaraties_full_' + jaar + '.csv')

# inladen sectorbepaling
df_koppel_segment_per_kvk = pd.read_csv(filepath_output + 'koppeltabel_segment_per_kvk.csv')

# inladen lumpsumbedragen
df_wlz_totalen_2025 = pd.read_csv(filepath_output + 'df_wlz_totalen_' + jaar + '.csv')

# inladen koppeltabel organisatienaam
df_koppel_organisatienaam = pd.read_excel(filepath_input + 'koppeltabel_organisatienamen.xlsx')
df_koppel_organisatienaam["kvk"] = pd.to_numeric(
   df_koppel_organisatienaam["kvk"], errors="coerce"
)


# COMMAND ----------

# MAGIC %md
# MAGIC ### Kolommen toevoegen en aggregratie per kvk

# COMMAND ----------
```

```
# Inladen data
df_base = df_wlz_declaraties_full_2025.copy()

# Filter sector
df_base = pd.merge(df_base, df_koppel_segment_per_kvk, on = 'kvk', how = 'left')
df_base = df_base[df_base['sector'] == 'GZ'] # Filter op GZ- organisaties

# COMMAND ----------

# Groeperen per aanbieder
keys = ["kvk"]

agg = {
   "aantal_verrichtingen": "sum",
   "zzp_intramuraal_verricht": "sum"
}

df_base = (
   df_base.groupby(keys, as_index=False)
      .agg(agg)
   )

# COMMAND ----------

#Toevoegen omzet (wlz_totalen)
df_base = pd.merge(df_base, df_wlz_totalen_2025[['kvk','vergoed_bedrag_totaal']], on = 'kvk', how =
'left')
df_base['vergoed_bedrag'] = df_base['vergoed_bedrag_totaal']



# COMMAND ----------

#Percentage intramuraal berekenen
df_base["intramuraal_aandeel"] = df_base["zzp_intramuraal_verricht"] /
df_base["aantal_verrichtingen"]

# COMMAND ----------

# MAGIC %md
# MAGIC ## Check grenswaardes
```

```
# COMMAND ----------

# MAGIC %md
# MAGIC ### Intramuraal

# COMMAND ----------

# Check clustering grens
s = pd.to_numeric(df_base["intramuraal_aandeel"], errors="coerce").dropna()

plt.figure(figsize=(8,4))
plt.hist(s, bins=100, color = "#0E3E37")  # eenvoudig: automatische bin-keuze
plt.ylabel("Aantal")
plt.title("Verhouding van intramurale prestaties [intramurale verrichtingen t.o.v. totaal aantal]")
plt.tight_layout()
plt.axvline(x = 0.01, color = "#E87200")
plt.show()

# COMMAND ----------

# MAGIC %md
# MAGIC ## Clustering

# COMMAND ----------

# MAGIC %md
# MAGIC ### 2 clusters

# COMMAND ----------

# - Extramuraal = A
# - (Deels) intramuraal = B

# cluster-kolom als object om problemen met NaN/strings te vermijden
df_base["cluster"] = pd.Series(pd.NA, dtype = "object")

# A Alleen extramuraal (≈ 100% extramuraal)
df_base["cluster"] = np.where(
   df_base["intramuraal_aandeel"] < 0.01,
```

```
    "A",
    df_base["cluster"]
)

# B (Deels) intramuraal
df_base["cluster"] = np.where(
    (   df_base["cluster"].isna() &                                 # niet A
        df_base["intramuraal_aandeel"] >= 0.01

    ),
    "B",
    df_base["cluster"]
)

# Voeg een beschrijvende kolom toe voor de clusterindeling
df_base["cluster_omschrijving"] = df_base["cluster"].map({
    "A": "Extramuraal",
    "B": "(Deels) Intramuraal"
})

assert df_base["cluster"].notna().all(), "Niet alle aanbieders zijn ingedeeld in een cluster"

# COMMAND ----------

# Kopieën
# Enkel clusterindeling
df_clusters_GZ = df_base.copy()
df_clusters_GZ = df_clusters_GZ[['kvk', 'cluster', 'cluster_omschrijving']]

# Bredere informatieset
df_clusters_GZ_output = df_base.copy()

# Join organisatienaam
df_clusters_GZ_output = pd.merge(df_clusters_GZ_output, df_koppel_organisatienaam, on = 'kvk', how
= 'left')

# Label leveringsvorm
df_clusters_GZ_output['label_leveringsvorm'] = np.where(
    df_clusters_GZ_output['intramuraal_aandeel'] < 0.01,
    '100% extramuraal',
```

```
    '>0% intramuraal'
)

df_clusters_GZ_output['sector'] = 'GZ'

df_clusters_GZ_output_juridische_onderbouwing = df_clusters_GZ_output[['kvk',  'organisatienaam',
'sector', 'cluster', 'intramuraal_aandeel']]

# COMMAND ----------

# MAGIC %md
# MAGIC ### Wegschrijven

# COMMAND ----------

df_kvk_per_zk = pd.read_excel(filepath_output + 'kvk_per_zk.xlsx')
df_kvk_per_zk = df_kvk_per_zk[['kvk', 'zorgkantoor' , 'organisatienaam']]

display(df_kvk_per_zk)

# COMMAND ----------

# Left join df_clusters_GZ with df_kvk_per_zk on 'kvk'
df_clusters_GZ = df_clusters_GZ.merge(df_kvk_per_zk, on="kvk", how="left")
df_clusters_GZ['sector'] = 'GZ'

for zk in df_clusters_GZ['zorgkantoor'].unique():
    df_clusters_GZ_zk = df_clusters_GZ[df_clusters_GZ['zorgkantoor'] == zk]
    df_clusters_GZ_zk = df_clusters_GZ_zk[['kvk', 'organisatienaam', 'sector', 'cluster',
'cluster_omschrijving']]
    df_clusters_GZ_zk.to_excel(filepath_output + 'clusters/clusters_GZ_' + jaar + '_' + zk + '.xlsx',
index=False)

df_clusters_GZ.to_csv(filepath_output + 'df_clusters_GZ_' + jaar + '.csv',index=False)

# COMMAND ----------

df_clusters_GZ_output_juridische_onderbouwing.to_excel(filepath_output + 'df_clusters_GZ_output_'
+ jaar + '.xlsx',index=False)
```

VV clustering

```
 # Databricks notebook source
# MAGIC %md
# MAGIC # Spoor 2 - clustering
# MAGIC

# COMMAND ----------

# MAGIC %md
# MAGIC ## Voorbereiding

# COMMAND ----------

pip install openpyxl

# COMMAND ----------

# DBTITLE 1,Imports & Magic numbers
#Clear variabelen
# %reset -f

# Imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('seaborn-v0_8')
import warnings
warnings.simplefilter(action = 'ignore', category = pd.errors.SettingWithCopyWarning)

## gebruiker en werkmap definiëren
gebruiker = <Bestandspad>

filepath_input = gebruiker + "Input/"
filepath_output = gebruiker + "Output/"

# COMMAND ----------

jaar = '2025'

# COMMAND ----------
```

```
# MAGIC %md
# MAGIC ### Importeren van bestanden

# COMMAND ----------

# de bestanden die volgen uit de directory 'filepath_input' zijn brongegevens. deze vormen de basis van de analyses.
# de bestanden die volgen uit de directory filepath_output' zijn uitdraaien van de eerdere codes.

# inladen prestaties
df_wlz_declaraties_full_2025 = pd.read_csv(filepath_output + 'df_wlz_declaraties_full_' + jaar + '.csv')

# inladen sectorbepaling
df_koppel_segment_per_kvk = pd.read_csv(filepath_output + 'koppeltabel_segment_per_kvk.csv')

# inladen lumpsumbedragen
df_wlz_totalen_2025 = pd.read_csv(filepath_output + 'df_wlz_totalen_' + jaar + '.csv')

# inladen koppeltabel organisatienaam
df_koppel_organisatienaam = pd.read_excel(filepath_input + 'koppeltabel_organisatienamen.xlsx')
df_koppel_organisatienaam["kvk"] = pd.to_numeric(df_koppel_organisatienaam["kvk"],
errors="coerce")



# COMMAND ----------

# MAGIC %md
# MAGIC ### Kolommen toevoegen en aggregratie per kvk

# COMMAND ----------

# Inladen data
df_base = df_wlz_declaraties_full_2025.copy()

# Filter sector
df_base = pd.merge(df_base, df_koppel_segment_per_kvk, on = 'kvk', how = 'left')
df_base = df_base[df_base['sector'] == 'VV'] # Filter op VV- organisaties

# COMMAND ----------
```

```
# Groeperen per aanbieder
keys = ["kvk"]

agg = {
   "aantal_verrichtingen": "sum",
   "zzp_intramuraal_verricht": "sum",
   "vpt_verricht": "sum",
   "vpt_bedrag":"sum",
   "mpt_bedrag":"sum",
   "mpt_verricht": "sum"
}

df_base = (
   df_base.groupby(keys, as_index=False)
      .agg(agg)
   )

# COMMAND ----------

# Toevoegen omzet (wlz_totalen)
df_base = pd.merge(df_base, df_wlz_totalen_2025[['kvk','vergoed_bedrag_totaal']], on = 'kvk', how =
'left')



# COMMAND ----------

# Intramuraal verhouding toevoegen
df_base['intramuraal_aandeel'] = df_base['zzp_intramuraal_verricht'] / df_base['aantal_verrichtingen']

# COMMAND ----------

# MAGIC %md
# MAGIC ### VPT aandeel

# COMMAND ----------

df_extramuraal = df_base[df_base["intramuraal_aandeel"] < 0.01]
```

```
df_extramuraal['vpt_aandeel'] = df_base["vpt_bedrag"] / (df_extramuraal["vpt_bedrag"] +
df_extramuraal["mpt_bedrag"])

# Bins van 0 tot 1 in stapjes van 0.01
bins = np.linspace(0, 1, 101)

# Histogram gericht
plt.figure(figsize=(8,4))
plt.hist(df_extramuraal['vpt_aandeel'], bins=bins, edgecolor="white", alpha=0.7, color="#0E3E37")
plt.xlim(0, 1)
plt.ylabel("Aantal aanbieders")
plt.title("Verhouding VPT, extramurale aanbieders [omzet VPT t.o.v. totale omzet]")
plt.tight_layout()
plt.vlines(x = 0.85, ymin = 0, ymax = 46, colors="#E87200")
plt.ylim(0,50)
plt.show()

# COMMAND ----------

# MAGIC %md
# MAGIC ## Clustering

# COMMAND ----------

# MAGIC %md
# MAGIC ### 3 clusters binnen VV

# COMMAND ----------

#clustering benchmarking

# - Extramuraal + minder dan 85% VPT = A
# - Extramuraal + meer dan 85% VPT = B
# - Intramuraal = C

# cluster-kolom als object om problemen met NaN/strings te vermijden
df_base["cluster"] = pd.Series(pd.NA, dtype = "object")

# A minder dan 85% VPT
df_base["cluster"] = np.where(
```

```
    df_base["cluster"].isna() &
    ((df_base["intramuraal_aandeel"] < 0.01) &
        (df_base["vpt_bedrag"] < 0.85 * (df_base["vpt_bedrag"] + df_base["mpt_bedrag"]))
    ),
    "A",
    df_base["cluster"]
)

# B meer dan 85% VPT
df_base["cluster"] = np.where(
    ((df_base["intramuraal_aandeel"] < 0.01) &
        (df_base["vpt_bedrag"] >= 0.85 * (df_base["vpt_bedrag"] + df_base["mpt_bedrag"]))
    ),
    "B",
    df_base["cluster"]
)

# C (deels) intramuraal
df_base["cluster"] = np.where((
    df_base["cluster"].isna() &
    (df_base["intramuraal_aandeel"] >= 0.01)),
    "C",
    df_base["cluster"])

df_base["cluster_omschrijving"] = df_base["cluster"].map({
    "A": "Extramuraal",
    "B": "VPT" ,
    "C": "(Deels) Intramuraal"
})

assert df_base["cluster"].notna().all(), "Niet alle aanbieders zijn ingedeeld in een cluster"

# COMMAND ----------

# MAGIC %md
# MAGIC ### Output wegschrijven

# COMMAND ----------

# Kopieën
```

```
# Enkele clusterindeling
df_clusters_VV = df_base.copy()
df_clusters_VV = df_clusters_VV[['kvk','cluster', 'cluster_omschrijving']]

# Bredere informatieset
df_clusters_VV_output = df_base.copy()

# Join organisatienaam
df_clusters_VV_output = pd.merge(df_clusters_VV_output, df_koppel_organisatienaam, on = 'kvk', how
= 'left')

# Label leveringsvorm
df_clusters_VV_output['label_leveringsvorm'] = np.where(
   df_clusters_VV_output['intramuraal_aandeel'] < 0.01,
   '100% extramuraal',
   '>0% intramuraal'
)

# COMMAND ----------

df_kvk_per_zk = pd.read_excel(filepath_output + 'kvk_per_zk.xlsx')
df_kvk_per_zk = df_kvk_per_zk[['kvk', 'zorgkantoor', 'organisatienaam']]

display(df_kvk_per_zk)

# COMMAND ----------

# Left join df_clusters_VV with df_kvk_per_zk on 'kvk'
df_clusters_VV = df_clusters_VV.merge(df_kvk_per_zk, on="kvk", how="left")
df_clusters_VV['sector'] = 'VV'

for zk in df_clusters_VV['zorgkantoor'].unique():
   df_clusters_VV_zk = df_clusters_VV[df_clusters_VV['zorgkantoor'] == zk]
   df_clusters_VV_zk = df_clusters_VV_zk[['kvk', 'organisatienaam', 'sector', 'cluster',
'cluster_omschrijving']]
   df_clusters_VV_zk.to_excel(filepath_output + 'clusters/clusters_VV_' + jaar + '_' + zk + '.xlsx',
index=False)

df_clusters_VV.to_csv(filepath_output + 'df_clusters_VV_' + jaar + '.csv',index=False)
```

```
# COMMAND ----------

df_clusters_VV_output_juridische_onderbouwing = df_base.copy()

df_clusters_VV_output_juridische_onderbouwing['sector'] = 'VV'

df_clusters_VV_output_juridische_onderbouwing =
df_clusters_VV_output_juridische_onderbouwing[['kvk', 'sector',
'cluster','intramuraal_aandeel','vpt_bedrag', 'mpt_bedrag']]

df_clusters_VV_output_juridische_onderbouwing.to_excel(filepath_output + 'df_clusters_VV_output_'
+ jaar + '.xlsx',index=False)
```

NHC/NIC

```
# Databricks notebook source
# MAGIC %md
# MAGIC # Juridische onderbouwing: NHC-NIC

# COMMAND ----------

# MAGIC %md
# MAGIC ## Voorbereiding

# COMMAND ----------

pip install openpyxl

# COMMAND ----------

#Clear variabelen
# %reset -f

# Imports
import pandas as pd
import warnings
from scipy import stats
warnings.simplefilter(action='ignore', category=pd.errors.SettingWithCopyWarning)

## gebruiker en werkmap definiëren
gebruiker = <Bestandspad>
```

```
filepath_input = gebruiker + "Input/"
filepath_output = gebruiker + "Output/"

# COMMAND ----------

# MAGIC %md
# MAGIC ## Importeren van bestanden

# COMMAND ----------

# koppeltabel NHC-NIC

# mpt-sheet
df_koppel_nhcnic_mpt = pd.read_excel(filepath_input + 'Koppeltabel_NZamax_nhcnic.xlsx',
sheet_name = "MPT")

# vpt-sheet
df_koppel_nhcnic_vpt = pd.read_excel(filepath_input + 'Koppeltabel_NZamax_nhcnic.xlsx', sheet_name
= "VPT").rename(columns = {
   "NHC + Compensatie i.r.t NHC Wlz n.a.v. aanwijzing/amendement" : "NHC",
   "NIC + Compensatie i.r.t NIC Wlz n.a.v. aanwijzing/amendement" : "NIC",
   "dagbehandeling vg emg" : "dagbehandeling"
}
)

# zzp-sheet
df_koppel_nhcnic_zzp = pd.read_excel(filepath_input + 'Koppeltabel_NZamax_nhcnic.xlsx', sheet_name
= "ZZP").rename(columns = {
   "NHC + Compensatie i.r.t NHC Wlz n.a.v. aanwijzing/amendement" : "NHC",
   "NIC + Compensatie i.r.t NIC Wlz n.a.v. aanwijzing/amendement" : "NIC"
}
)

# koppeltabel maken (combineren van de drie sheets)
df_koppel_nhcnic = pd.concat([df_koppel_nhcnic_mpt,
                      df_koppel_nhcnic_vpt,
                      df_koppel_nhcnic_zzp],
                      ignore_index = True)
```

```
# COMMAND ----------

# declaratiegegevens op prestatieniveau
df_prestaties = pd.read_excel(filepath_output + 'totale_datasets_2025.xlsx', sheet_name = 'prestaties')

# COMMAND ----------

# clusterindeling
# we gebruiken deze in later stadium om de NHC- en NIC-inkomsten per cluster te berekenen
# we laden de clusteringsoutputs ins
df_clustering_vv_2025 = pd.read_excel(filepath_output + 'df_clusters_VV_output_2025.xlsx')
df_clustering_gz_2025 = pd.read_excel(filepath_output + 'df_clusters_GZ_output_2025.xlsx')
df_clustering_ggz_2025 = pd.read_excel(filepath_output + 'df_clusters_GGZ_output_2025.xlsx')

clusters = pd.concat([df_clustering_vv_2025,
                      df_clustering_gz_2025,
                      df_clustering_ggz_2025],
                      ignore_index = True)


# COMMAND ----------

# MAGIC %md
# MAGIC ## Berekening van NHC- en NIC-inkomsten per aanbieder

# COMMAND ----------

# selecteren van juiste gegevens uit koppeltabel
df_koppel_nhcnic_an = df_koppel_nhcnic[["Code", "NHC", "NIC"]].rename(columns = {"Code" :
"prestatiecode",
                                                                                 "NHC" : "nhc",
                                                                                 "NIC" : "nic"})

# koppelen van NHC- en NIC-bedragen aan geleverde prestaties
df_prestaties_nhcnic = pd.merge(df_prestaties, df_koppel_nhcnic_an, on = ["prestatiecode"], how =
"left")

# COMMAND ----------

# we vervangen de missende waarden van NHC/NIC in de tabel nu door 0
```

```
df_prestaties_nhcnic["nhc"] = df_prestaties_nhcnic["nhc"].fillna(0)
df_prestaties_nhcnic["nic"] = df_prestaties_nhcnic["nic"].fillna(0)

# COMMAND ----------

# voor onze analyse filteren we de rijen met minder dan 10 verrichtingen eruit
# dit doen we omdat we daarvoor geen vergoed bedrag hebben. hierdoor ontstaan anders niet kloppende geschoonde resultaten.
# (we zouden dan wel NHC-NIC-inkomsten berekenen voor deze verrichtingen terwijl er geen bedrag is vergoed voor de verrichtingen)
df_prestaties_nhcnic = df_prestaties_nhcnic[df_prestaties_nhcnic["vergoed_bedrag"] != 0]

# berekenen geaggregeerde NHC/NIC-waarden
df_prestaties_nhcnic["nhc_agg"] = df_prestaties_nhcnic["aantal_verrichtingen"] *
df_prestaties_nhcnic["nhc"]
df_prestaties_nhcnic["nic_agg"] = df_prestaties_nhcnic["aantal_verrichtingen"] *
df_prestaties_nhcnic["nic"]
df_prestaties_nhcnic["nhcnic_agg"] = df_prestaties_nhcnic["nhc_agg"] +
df_prestaties_nhcnic["nic_agg"]

# geschoond vergoed bedrag berekenen
df_prestaties_nhcnic["vergoed_bedrag_schoon"] = df_prestaties_nhcnic["vergoed_bedrag"] -
df_prestaties_nhcnic["nhcnic_agg"]

# we houden hier rijen over waarbij geschoond vergoed bedrag kleiner is dan 0
# het gaat om een klein aantal rijen (379 / 129183 - onderaan printen we deze waarden)
# we filteren deze rijen eruit omdat dit niet aannemelijk is
print("Rijen verwijderd vanwege vergoed bedrag kleiner dan 0:")
print(df_prestaties_nhcnic[df_prestaties_nhcnic["vergoed_bedrag_schoon"] < 0].shape[0])
print(df_prestaties_nhcnic.shape[0])
df_prestaties_nhcnic = df_prestaties_nhcnic[df_prestaties_nhcnic["vergoed_bedrag_schoon"] >= 0]

# COMMAND ----------

# vervolgens aggreren we de relevante kolommen (waaronder NHC- en NIC-bedragen) per KVK-nummer en jaar
df_vergoed_bedrag = df_prestaties_nhcnic.groupby(["kvk"], as_index = False).agg(
   {"aantal_verrichtingen" : "sum",
    "nhc" : "mean",
    "nic" : "mean",
```

```
    "nhc_agg" : "sum",
    "nic_agg" : "sum",
    "nhcnic_agg" : "sum",
    "vergoed_bedrag_schoon" : "sum"}
).rename(columns = {
   "nhc" : "nhc_gem",
   "nic" : "nic_gem"
})

# COMMAND ----------

# MAGIC %md
# MAGIC ## Berekening van NHC- en NIC-inkomsten per sector en cluster

# COMMAND ----------

# NHC- en NIC-bedragen koppelen aan de clusterindelingen
df_clusters = pd.merge(clusters, df_vergoed_bedrag, on = ["kvk"], how = "left")

# NHC- en NIC-inkomsten per sector en cluster berekenen
df_nhcnic_tot = (
   df_clusters
   .groupby(["sector", "cluster"], as_index=False)[["nhcnic_agg"]]
   .sum()
)

# COMMAND ----------

# MAGIC %md
# MAGIC ### Wegschrijven gegevens

# COMMAND ----------

# tabel met NHC-NIC-bedragen wegschrijven
df_vergoed_bedrag.to_excel(filepath_output + 'geschoond_vergoed_bedrag.xlsx')

# tabel met NHC-NIC-bedragen per cluster wegschrijven
df_nhcnic_tot.to_excel(filepath_output + 'nhcnic_per_cluster.xlsx', index = False)
```

Benchmarking

```
# Databricks notebook source
# MAGIC %md
# MAGIC # Spoor 2 - Benchmarking op alleen zorgdeel (geen nhc nic)

# COMMAND ----------

# MAGIC %md
# MAGIC ## Voorbereiden

# COMMAND ----------

pip install openpyxl

# COMMAND ----------

#Clear variabelen
# %reset -f

# Imports
import numpy as np
import pandas as pd
import warnings
warnings.simplefilter(action='ignore', category=pd.errors.SettingWithCopyWarning)

## gebruiker en werkmap definiëren
gebruiker = <Bestandspad>

filepath_input = gebruiker + "Input/"
filepath_output = gebruiker + "Output/"

# COMMAND ----------

# MAGIC %md
# MAGIC ## Variabelen kiezen

# COMMAND ----------

# jaar selecteren
jaar_txt = "2025"
jaar = 2025
```

```
# COMMAND ----------

# kostenpercentages
df_kostenpercentages = pd.read_csv(filepath_output + 'df_benchmarking_' + jaar_txt + '.csv')

# clusters
df_clusters = pd.read_csv(filepath_output + 'df_clusters_VV_' + jaar_txt + '.csv')
# df_clusters = pd.read_csv(filepath_output + 'df_clusters_GZ_' + jaar_txt + '.csv')
# df_clusters = pd.read_csv(filepath_output + 'df_clusters_GGZ_' + jaar_txt + '.csv')

# selecteer clusteringoptie
df_clusters = df_clusters[['kvk','cluster']]

#Definieer output bestand (pas sectornaam aan)
#Aanpassen bij verandering instellingen
naam_output_bestand_rtp = "VV nieuw 2025 (huidige methode, outliers per cluster)"
#naam_output_bestand_rtp = "GZ nieuw 2025 (huidige methode, outliers per cluster)"
#naam_output_bestand_rtp = "GGZ nieuw 2025 (huidige methode, outliers per cluster)"



#clusters eruit halen. Niet meer van toepassing (dus altijd leeg laten).
clusters_uitsluiten = [""]

# COMMAND ----------

#Kies een grens:
q_bound = 0.75

# COMMAND ----------

# clusters aan kostenpercentages koppelen
df_benchmark = pd.merge(df_kostenpercentages, df_clusters, on = "kvk", how = "left")

# voor berekening aanbieders zonder cluster eruit laten (anders wel in berekening van
sectorpercentage, maar niet in cluster)
# let op: hier vallen aanbieders eruit die we niet konden indelen in een cluster (en in andere sector
vallen)
df_benchmark = df_benchmark[df_benchmark["cluster"].notna()]
```

```
# tabel maken voor analyse
df_benchmark_nhc_nic_correctie = df_benchmark.copy()

# aanbieders zonder kostenratio weglaten
df_benchmark_nhc_nic_correctie =
df_benchmark_nhc_nic_correctie[df_benchmark_nhc_nic_correctie["kosten_ratio"].notna()]

# COMMAND ----------

# MAGIC %md
# MAGIC ## NHC NIC schonen

# COMMAND ----------

# we lezen het bestand in waarin geaggregeerd NHC-NIC-bedragen beschikbaar zijn per aanbieder
df_nhc_nic = pd.read_excel(filepath_output + 'geschoond_vergoed_bedrag.xlsx')
df_nhc_nic = df_nhc_nic.drop(columns = "Unnamed: 0")

# COMMAND ----------

# we koppelen de NHC-NIC bedragen aan de benchmarkinformatie
df_benchmark_nhc_nic_correctie = pd.merge(df_benchmark_nhc_nic_correctie, df_nhc_nic, on =
["kvk"], how = "left")

# Ook voor df-benchmark nhc nic nodig (we gebruiken dit DataFrame in een later stadium)
df_benchmark = pd.merge(df_benchmark, df_nhc_nic, on = ["kvk"], how = "left")
df_benchmark['nza_max_min_nhc_nic'] = df_benchmark['nza_max'] - df_benchmark['nhcnic_agg']

# COMMAND ----------

# we willen kostenpercentages berekenen zonder de NHC-NIC-bedragen
# daarom berekenen we hier de Nza-max-bedragen zonder de NHC-NIC-bedragen
# omdat we de Nza-max-bedragen op geaggregeerd niveau hebben, trekken we de NHC-NIC-bedragen
ook op geaggregeerd niveau hiervan af
df_benchmark_nhc_nic_correctie['nza_max_min_nhc_nic'] =
df_benchmark_nhc_nic_correctie['nza_max'] - df_benchmark_nhc_nic_correctie['nhcnic_agg']
df_benchmark_nhc_nic_correctie['wlz_kosten_min_nhc_nic'] =
df_benchmark_nhc_nic_correctie['wlz_kosten'] - df_benchmark_nhc_nic_correctie['nhcnic_agg']

# COMMAND ----------
```

```
# we berekenen de kostenratio o.b.v. geschoonde Wlz-kosten en Nza-max (voor de NHC-NIC-bedragen)
df_benchmark_nhc_nic_correctie['kosten_ratio'] =
df_benchmark_nhc_nic_correctie['wlz_kosten_min_nhc_nic'] /
df_benchmark_nhc_nic_correctie['nza_max_min_nhc_nic']

# COMMAND ----------

# we maken een kopie van het dataframe om de analyses op uit te voeren
df_benchmark_tmp = df_benchmark_nhc_nic_correctie.copy()

# COMMAND ----------

# MAGIC %md
# MAGIC ## Outliers bepalen

# COMMAND ----------

df_copy = df_benchmark_tmp.copy()

df_temp = df_copy[['kvk', 'cluster', 'kosten_ratio']]

stats_per_cluster = (df_temp.groupby('cluster')['kosten_ratio']
                .agg(median = 'median', sd = lambda s: s.std(skipna = True, ddof = 1))
                .reset_index())

df_temp = df_temp.merge(stats_per_cluster, on = 'cluster', how = 'left')

mask_outlier = (
   (df_temp['kosten_ratio'] <= df_temp['median'] - 2 * df_temp['sd']) |
   (df_temp['kosten_ratio'] >= df_temp['median'] + 2 * df_temp['sd'])
)

df_temp['outlier_cluster'] = 0
df_temp.loc[mask_outlier, 'outlier_cluster'] = 1

df_temp = df_temp.drop(columns = ['median', 'sd'])

df_benchmark_tmp = pd.merge(df_benchmark_tmp, df_temp, on = ['kvk', 'cluster', 'kosten_ratio'])
```

```
# COMMAND ----------

df_benchmark_tmp = df_benchmark_tmp[df_benchmark_tmp["outlier_cluster"] == 0]

# COMMAND ----------

# MAGIC %md
# MAGIC ## Functie marktaandeel

# COMMAND ----------

def kostenratio_marktaandeel(df, q):

    # als dataframe leeg is, komt er geen kostenpercentage
    if df.empty:
        return float('nan')

    # dataframe sorteren op kostenratio, zodat cumulatieve omzet berekend kan worden
    df_sorted = df.sort_values('kosten_ratio').copy()

    # cumulatieve omzet berekenen
    df_sorted['cumul_omzet'] = df_sorted['vergoed_bedrag_totaal'].cumsum()

    # totale omzet berekenen
    totaal_omzet = df_sorted['vergoed_bedrag_totaal'].sum()

    # marktaandeel berekenen
    df_sorted['cumul_pct'] = df_sorted['cumul_omzet'] / totaal_omzet

    # kostenratio bij q%-marktaandeel retourneren
    return df_sorted[df_sorted['cumul_pct'] >= q].iloc[0]['kosten_ratio']

# COMMAND ----------

# MAGIC %md
# MAGIC ## Benchmark per cluster

# COMMAND ----------

# leeg dataframe maken
```

```
rt_cl = pd.DataFrame(columns = ["cluster", "rtp_cluster"])

# for loop voor alle clusters starten
for c in df_benchmark_tmp['cluster'].unique():


    # dataframe filteren op cluster
    df_cluster = df_benchmark_tmp[df_benchmark_tmp['cluster'] == c]

    q_cl = np.nanquantile(df_cluster['kosten_ratio'], q_bound, method = "higher")
    ma_cl = kostenratio_marktaandeel(df_cluster, q_bound)
    max_cl = max(q_cl, ma_cl)

    print(round(q_cl,3))
    print(round(ma_cl,3))



    # resultaten opslaan in rij (voor het cluster)
    tmp_rij = pd.DataFrame({"cluster" : [c], "rtp_cluster" : [max_cl]})

    # rij toevoegen aan tabel
    rt_cl = pd.concat([rt_cl, tmp_rij], ignore_index = True)

rt_cl = rt_cl.sort_values("cluster")
display(rt_cl.reset_index())

# COMMAND ----------

rt_cl.to_excel(filepath_output + naam_output_bestand_rtp + '.xlsx',index=False)
```

Bijlage 7.c: uitvraagformulier declaratiegegevens

Zie bijgevoegd Excel document "*Uitvraagformat declaratiegegevens 2025 Wlz.xslx*".