

Rechthoeken met allemaal puntjes erin

Dennis, Fons, Matthijs,
Rick & Sebastiaan



Opdracht

Wie zijn wij?

- Bachelorstudenten Wiskunde aan de Radboud Universiteit
- Vak Modellenpracticum
- Opdracht: vind clusters in een dataset



Wat is een SCG?

- SCG, **Smart Cable Guard** is een systeem om locaties en tijden van ontladingen in een draad te meten
- Veel ontladingen? ➔ Er is iets mis met die plek in de kabel
- Fouten in de kabel zitten meestal bij **moffen**
- Moffen zijn plekken waar twee stukken kabel aan elkaar zijn **verbonden**
- SCG meet ook de **ladingssterkte** in pc

De data

-Drie kommagescheiden bestanden per circuit:

2063-pd.csv

2063-cableconfig.csv

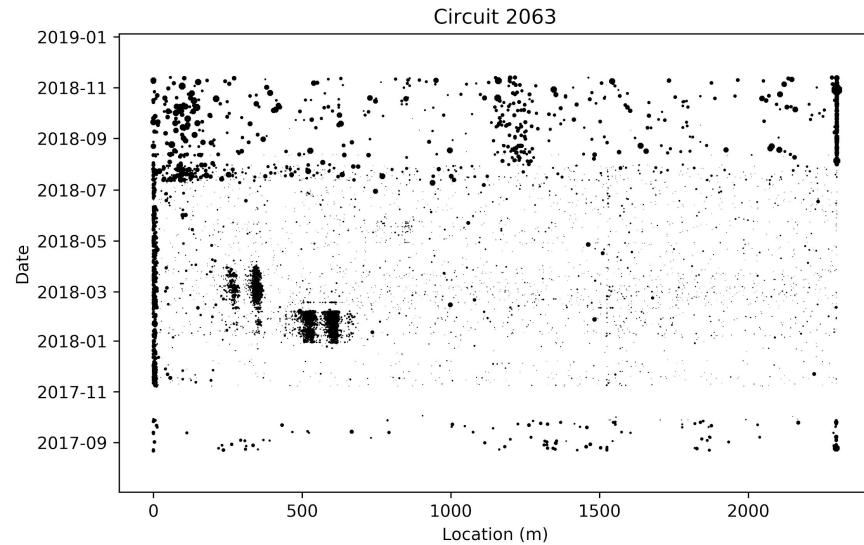
2063-warning.csv

-Geanonimiseerd



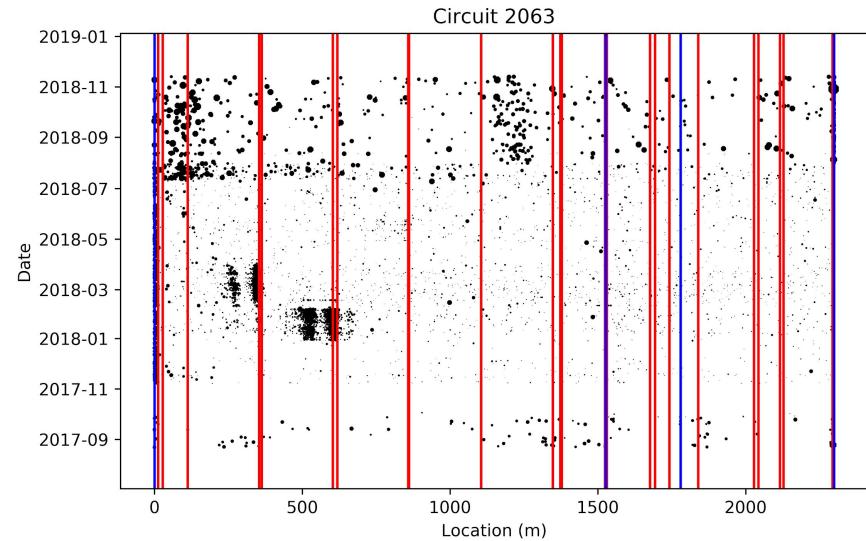
2063-pd.csv

```
2063-pd - Notepad
File Edit Format View Help
Date/time (UTC);Location in meters (m);Charge (picocoulomb)
2017-08-21 14:39:00;;
2017-08-21 14:40:00;;
2017-08-21 14:41:00;;
.
.
.
2017-10-03 05:09:00;;
2017-10-03 05:10:00;;
2017-10-03 05:11:00;;
2017-10-03 05:12:00;;
2017-10-03 05:13:00;;
2017-10-03 05:14:00;;
2017-10-03 05:15:00;;
2017-10-03 05:16:00;905.105796009168;5201.5
2017-10-03 05:17:00;;
2017-10-03 05:18:00;;
2017-10-03 05:19:00;;
2017-10-03 05:20:00;;
2017-10-03 05:21:00;;
2017-10-03 05:22:00;;
2017-10-03 05:23:00;;
2017-10-03 05:24:00;;
2017-10-03 05:25:00;;
2017-10-03 05:26:00;;
```

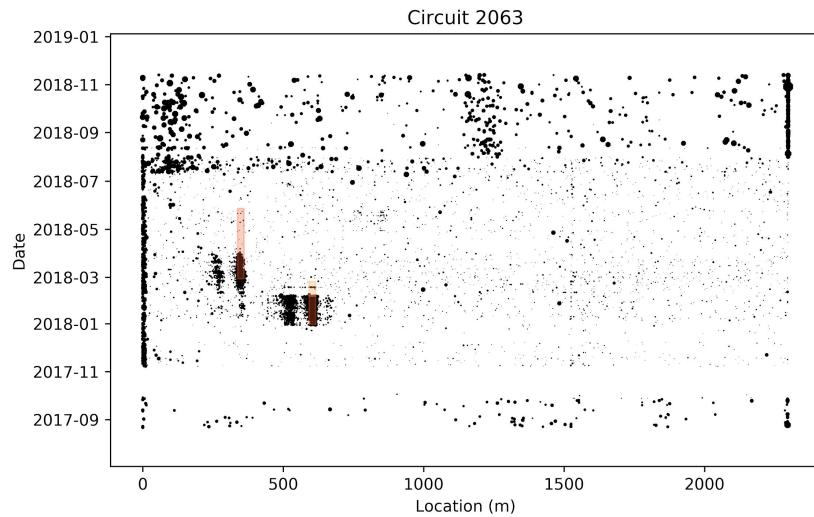


2063-cableconfig.csv

```
2063-cableconfig - Notepad
File Edit Format View Help
Component type;Length (m);Cumulative length (m)
RMU;;0.0
Termination (unknown);;0.0
Cable (PILC, 3 cores, belted);12.0;12.0
Joint (resin);;12.0
Cable (PILC, 3 cores, belted);16.0;28.0
Joint (oil);;28.0
Cable (PILC, 3 cores, belted);84.0;112.0
Joint (oil);;112.0
Cable (PILC, 3 cores, belted);242.0;354.0
Joint (resin);;354.0
Cable (PILC, 3 cores, belted);9.0;363.0
Joint (unknown);;363.0
Cable (PILC, 3 cores, belted);240.0;603.0
Joint (cold shrink);;603.0
Cable (XLPE, 3 cores, common earth screen);15.0;618.0
Joint (cold shrink);;618.0
Cable (PILC, 3 cores, belted);240.0;858.0
Joint (cold shrink);;858.0
Cable (XLPE, 3 cores, common earth screen);3.0;861.0
Joint (cold shrink);;861.0
Cable (PILC, 3 cores, belted);244.0;1105.0
Joint (oil);;1105.0
Cable (PILC, 3 cores, belted);242.0;1347.0
Joint (oil);;1347.0
```



2063-warnings.csv



2063-warning - Notepad

File Edit Format View Help

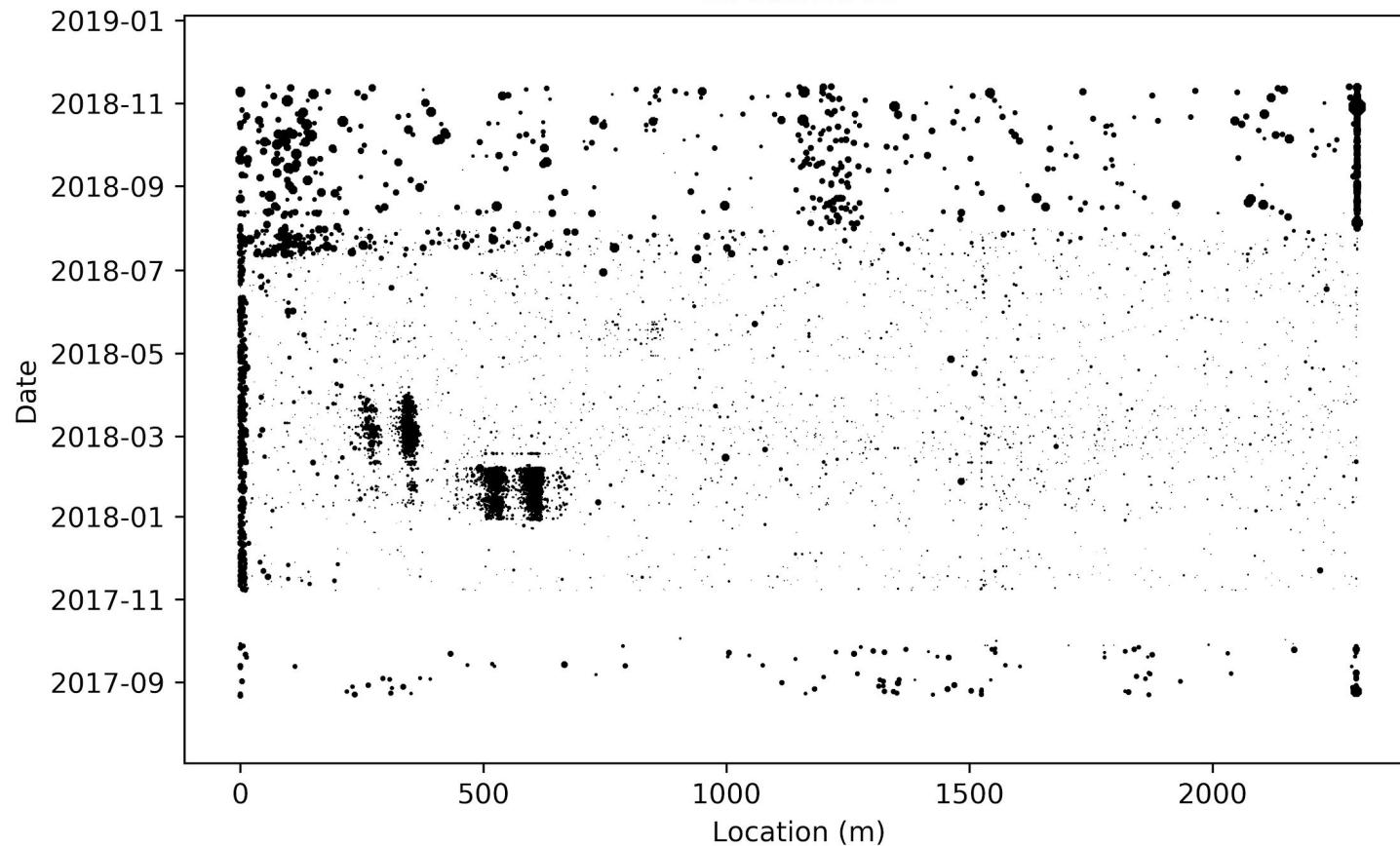
Location in meters (m);SCG warning level (1 to 3 or Noise);Start Date/time (UTC);End Date/time (UTC)

```
349;3;2018-02-26 22:40:58;2018-05-27 08:45:43
604;2;2018-02-05 00:03:25;2018-02-23 16:08:12
604;3;2017-12-30 19:48:43;2018-02-05 00:03:25
```

Wat is onze opdracht?

- Vinden verdachte gebieden, aka clusters
- Precieze definitie van zo'n cluster

Circuit 2063

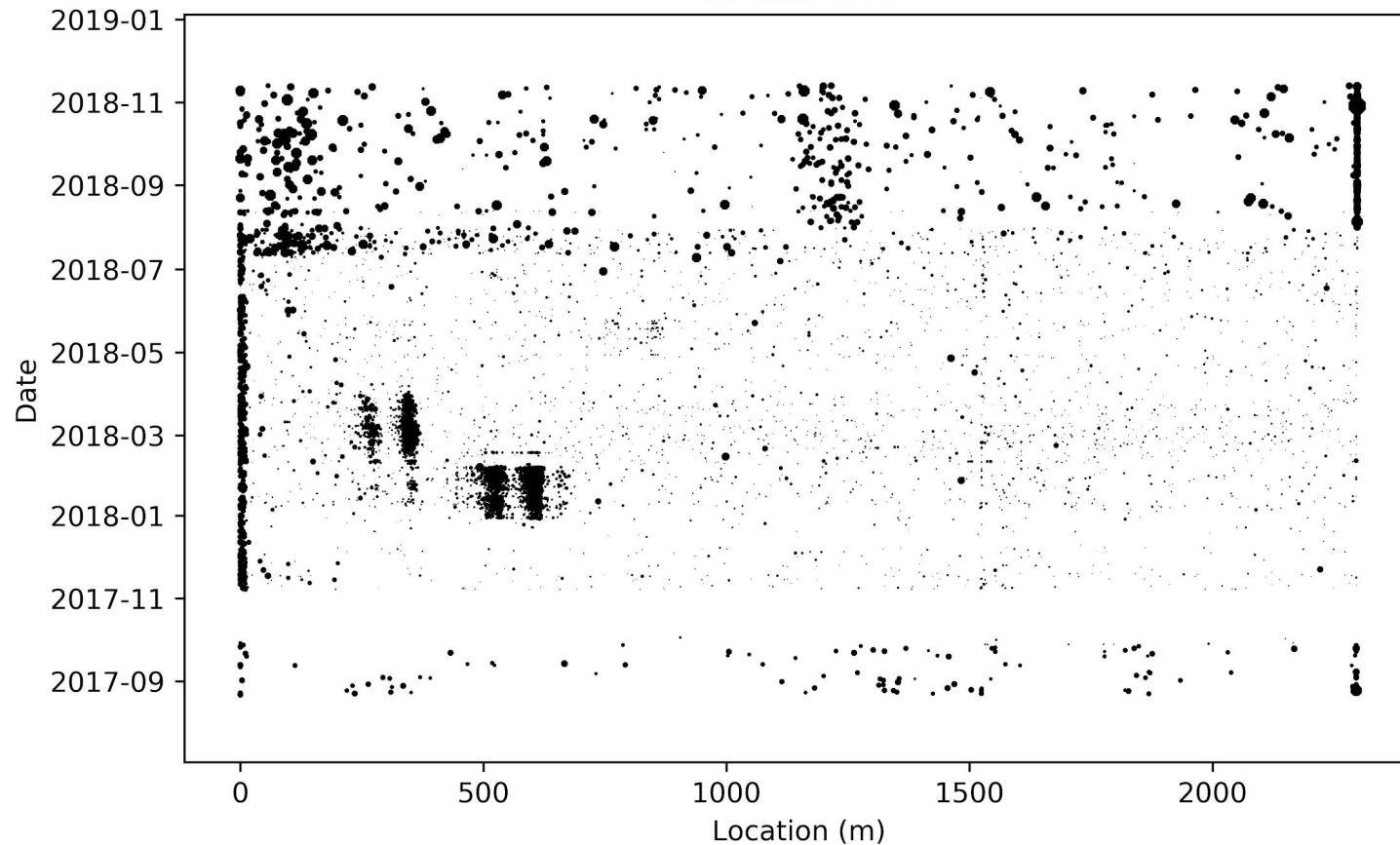


Onze aanpak

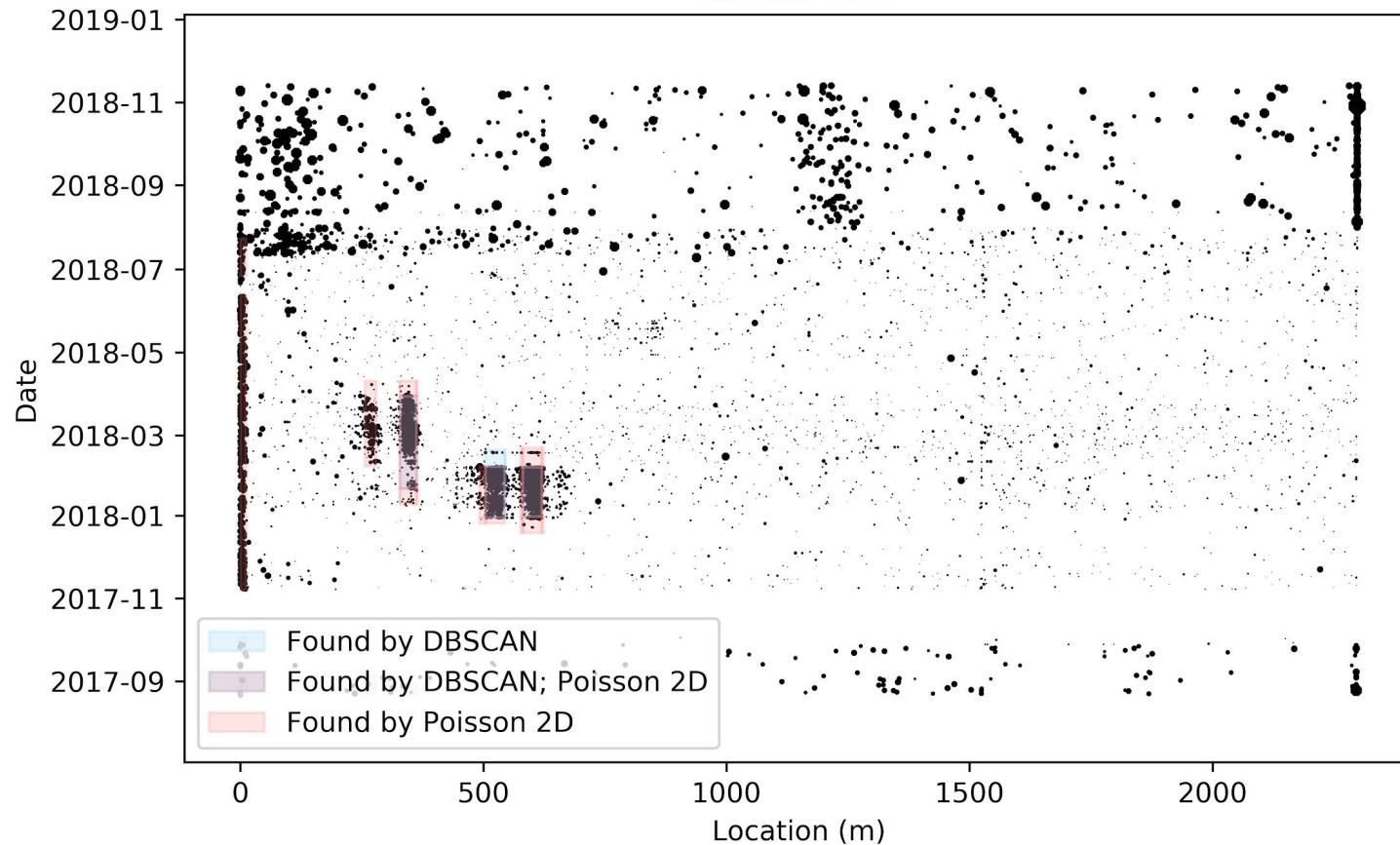
- Sommigen uit groepje veel programmeerervaring
- Tools: Python, pandas, Jupyter Notebook, Github
- Elke week samenkommen
- Data bestuderen bijv. met verdelingen



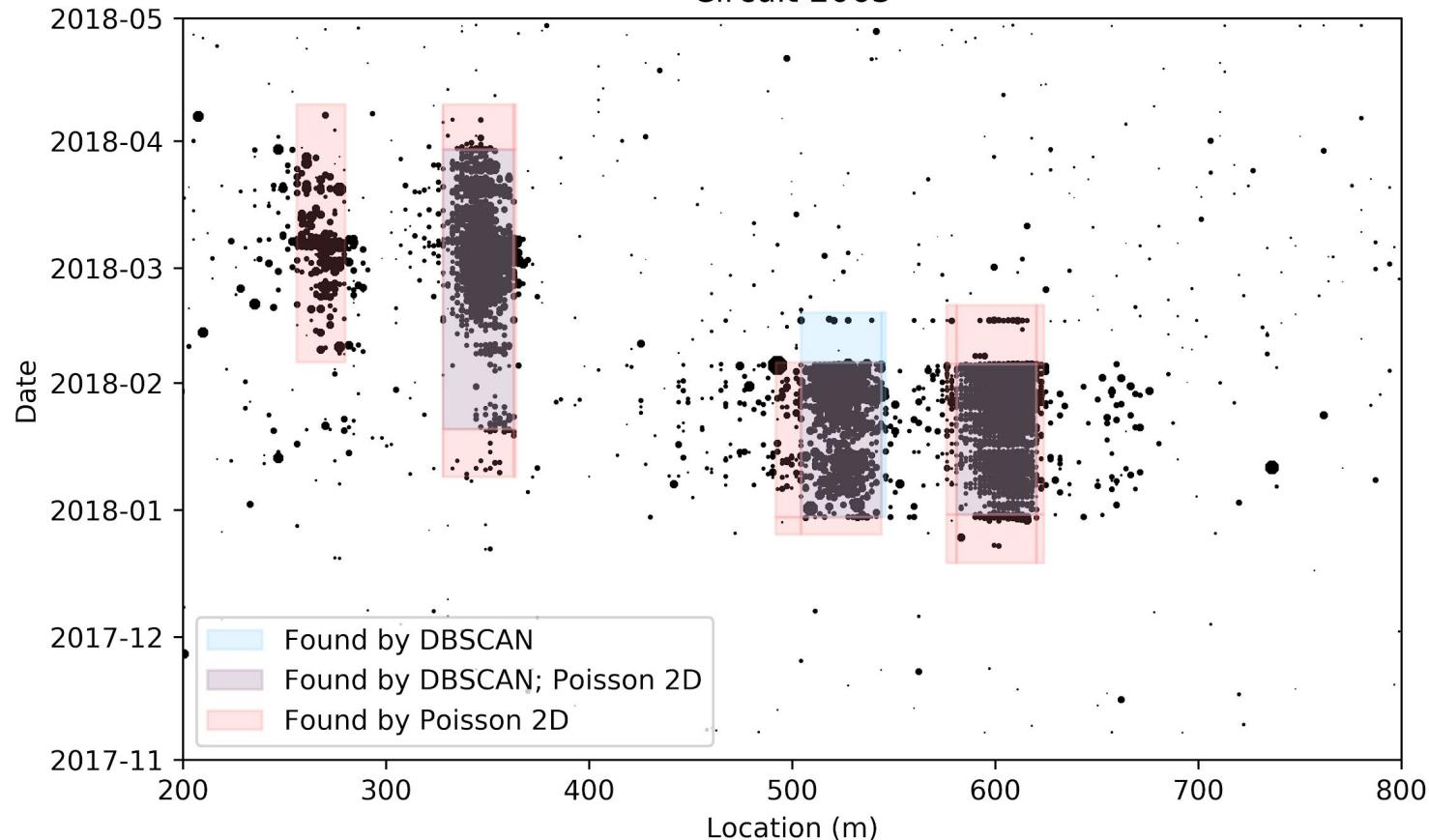
Circuit 2063



Circuit 2063



Circuit 2063





Algoritmen

Wat is een cluster?

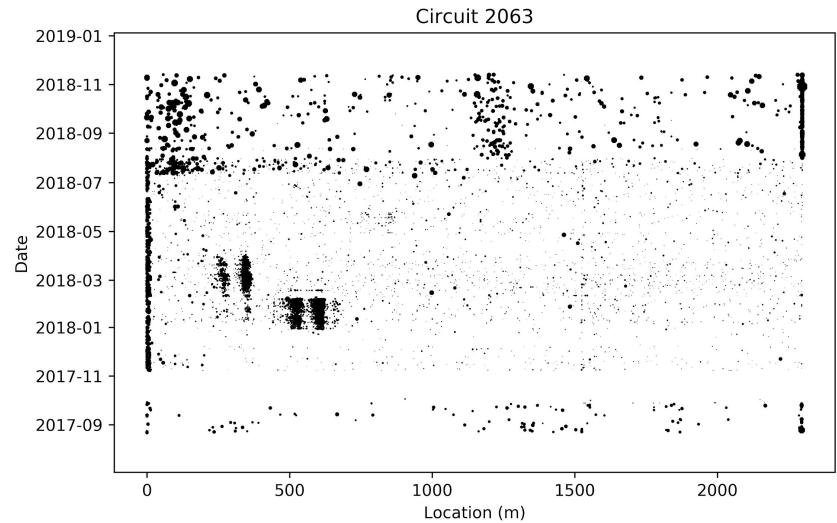
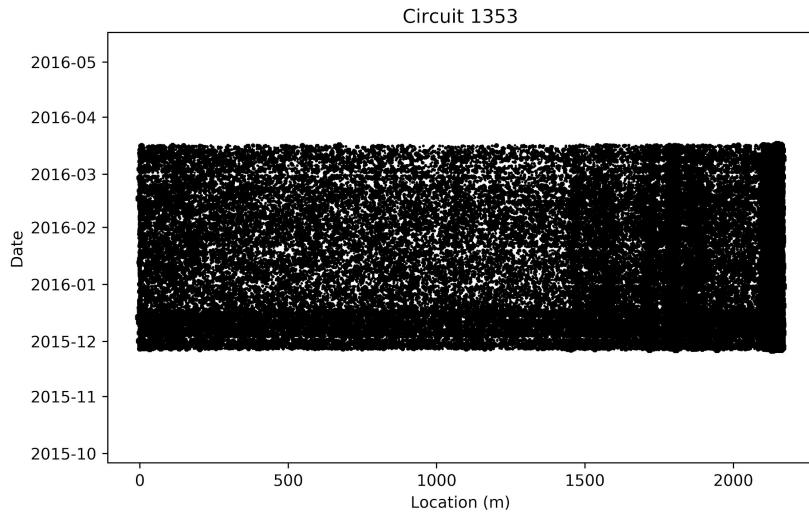
- In principe verzameling punten of aaneengesloten gebied met uitzonderlijk veel punten
- Onze definitie: Rechthoeken
- Draait om lokatie mof
- Resultaten verschillende algoritmen combineren



Voorwaarden algoritmen

- Snel genoeg
- Zo min mogelijk parameters
- Parameters onafhankelijk van circuit
- Standaardwaardes
- Returned specifiek type object: Cluster

Ter vergelijking



Algoritmen

- Poisson
- DBSCAN
- Pinta

Combineren:

- Ensemble

Bonus:

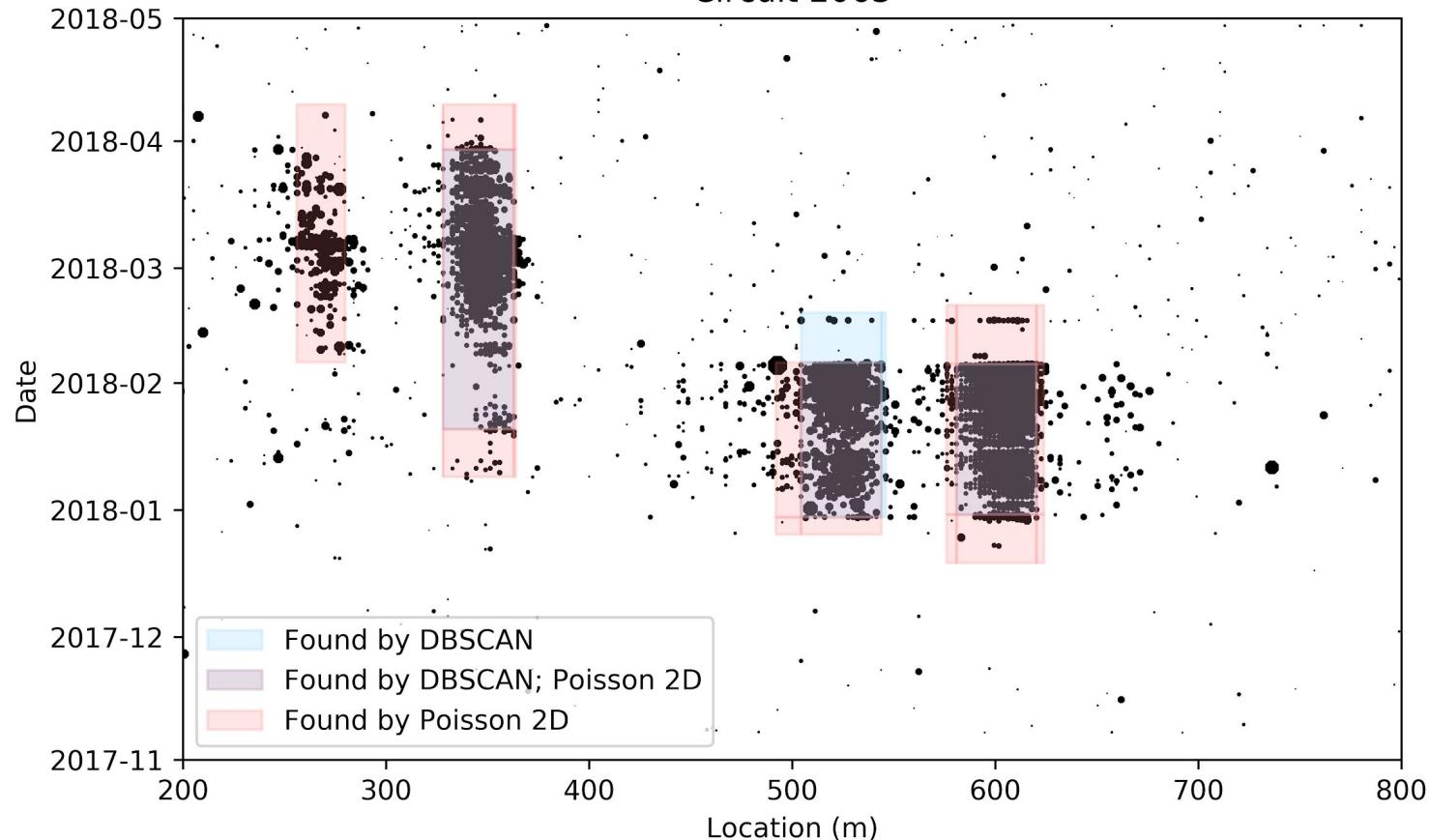
- Monte Carlo



Nog een keer teaser, na het tegenwoordig zodat je weer ziet dat de verschillende algoritmen ongeveer hetzelfde vinden



Circuit 2063







Poisson-algoritme

Poisson

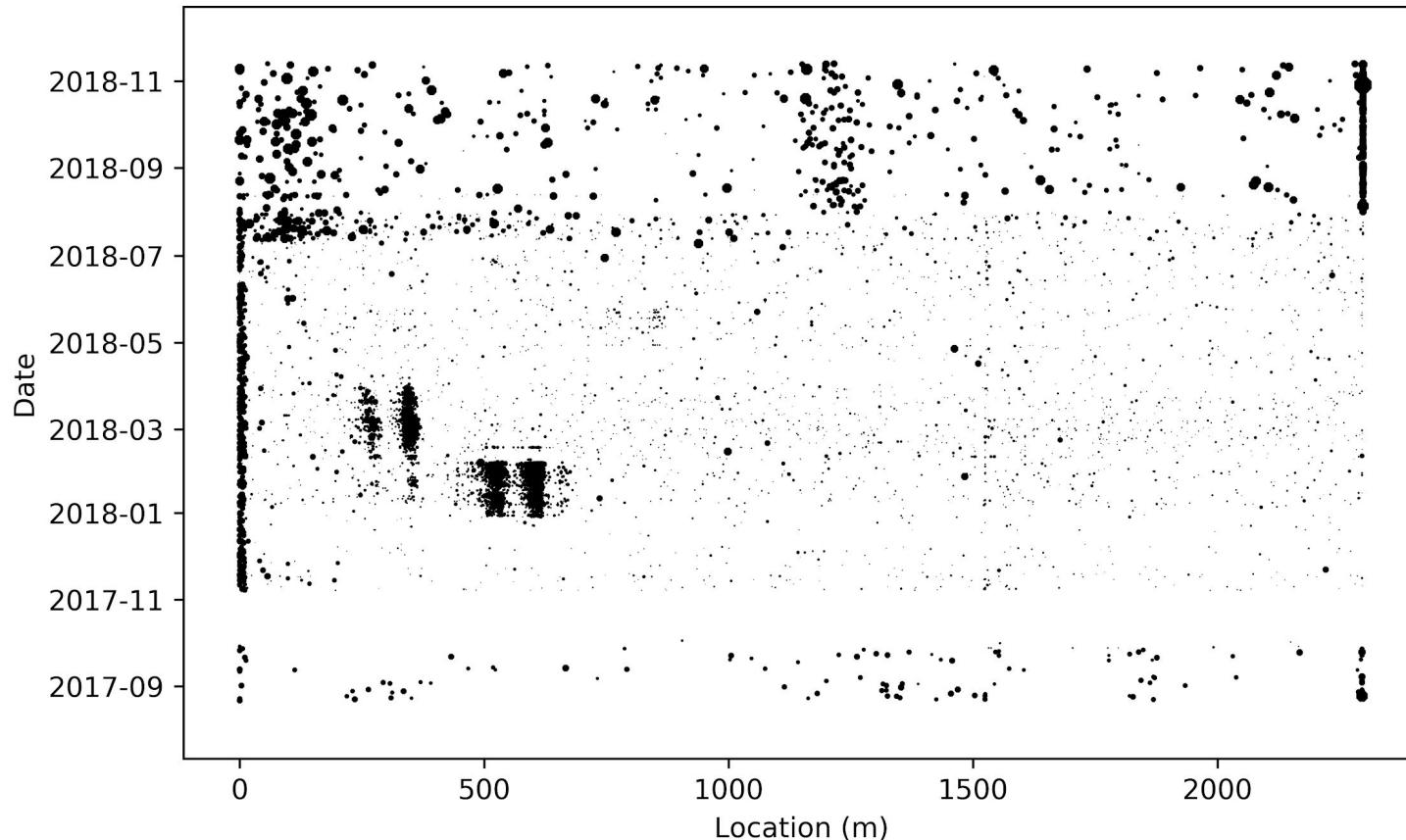
Algoritme werkt in **twee delen**:

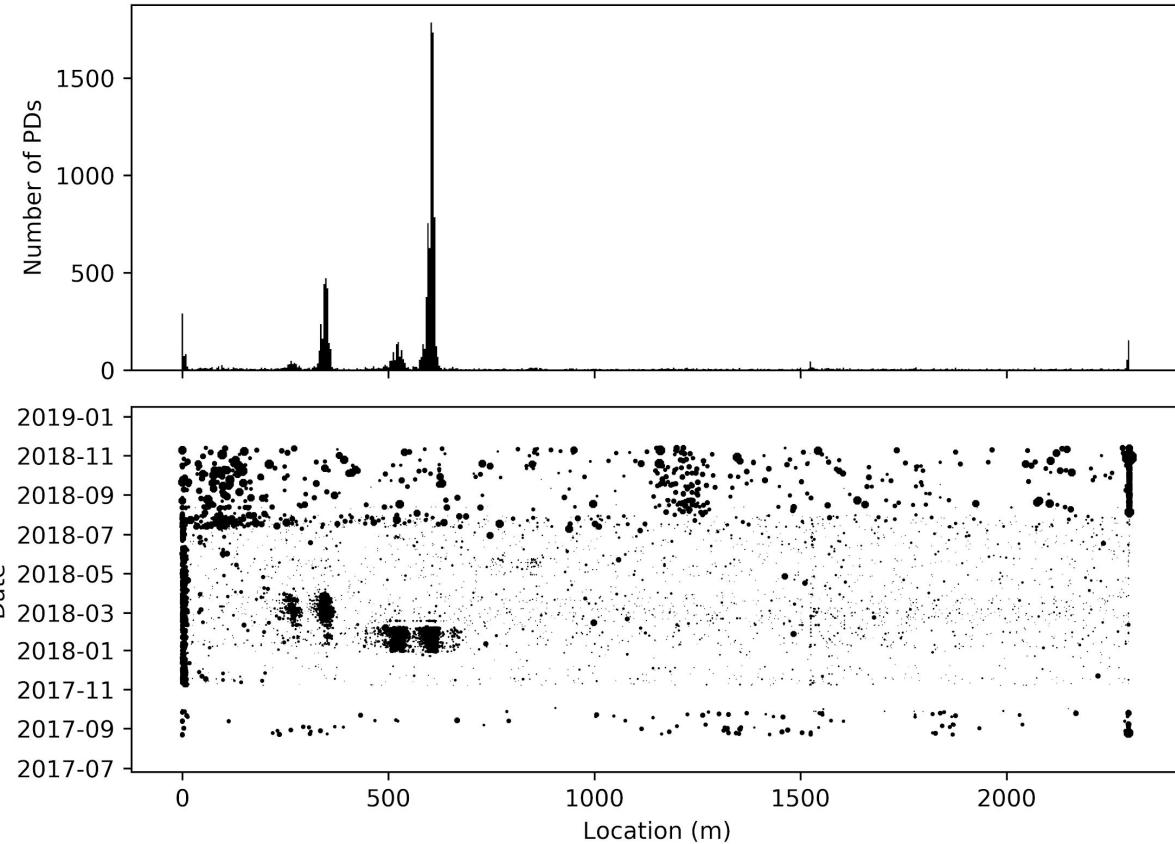
1. Zoek verdachte **lijnsegmenten** (1D)
2. Analyseer de **tijdverloop** van elk segment (2D)

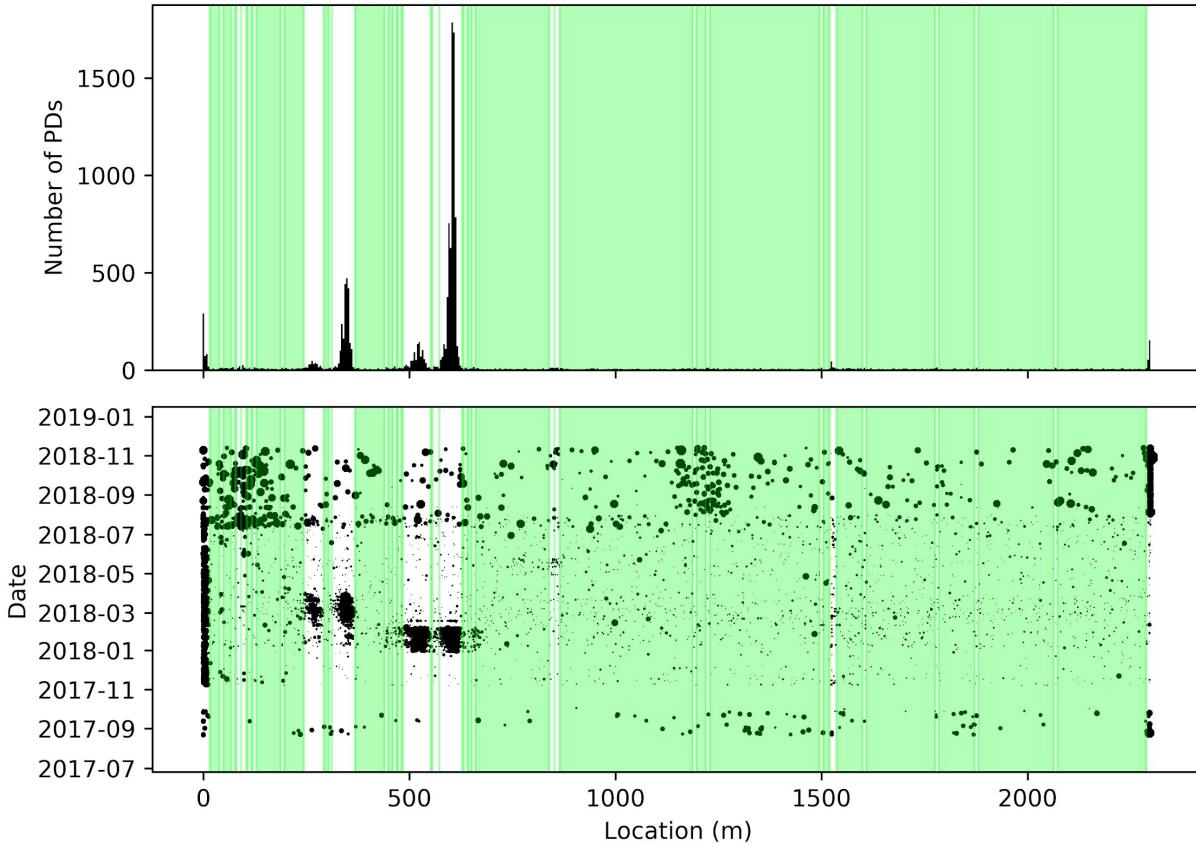
Circuit 2063

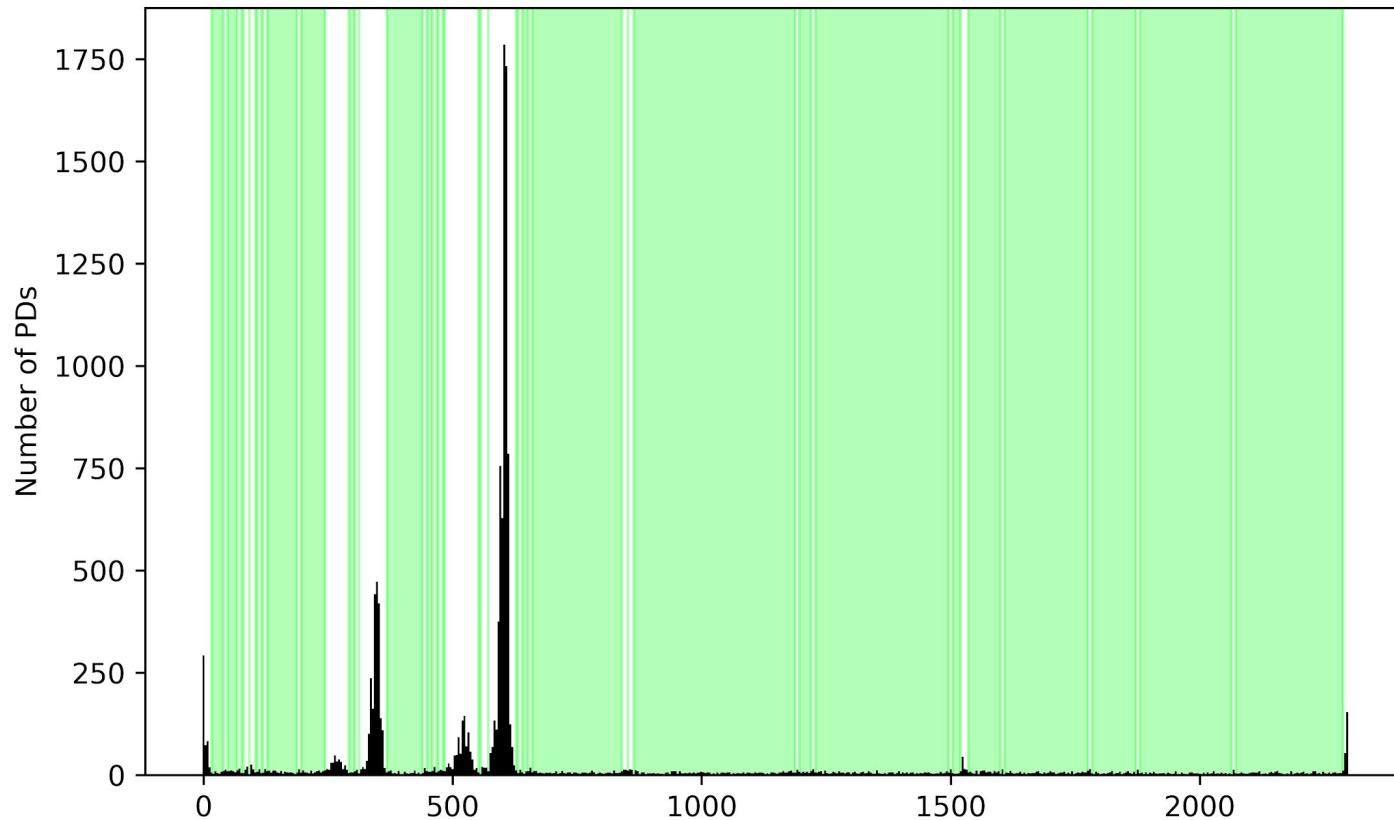
1D

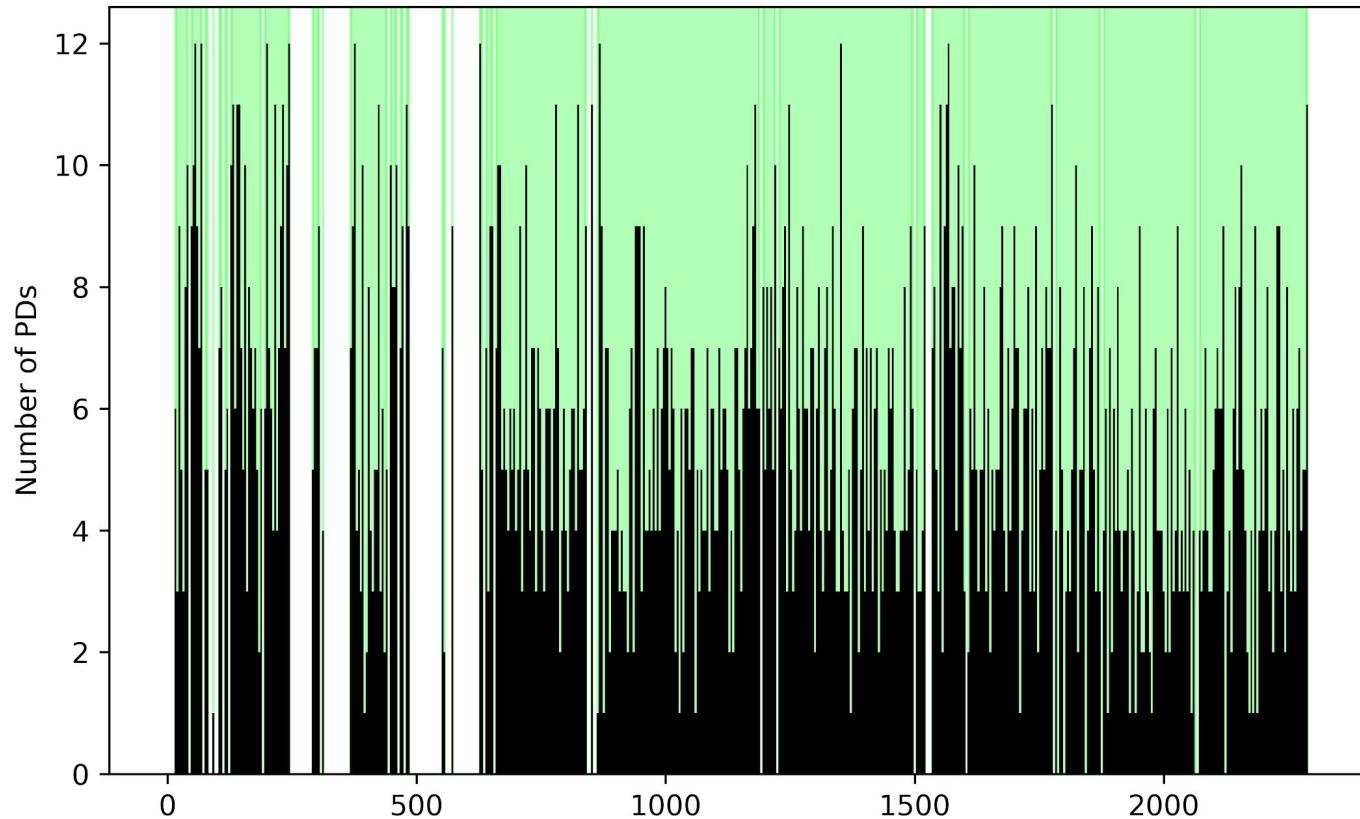
Circuit 2063

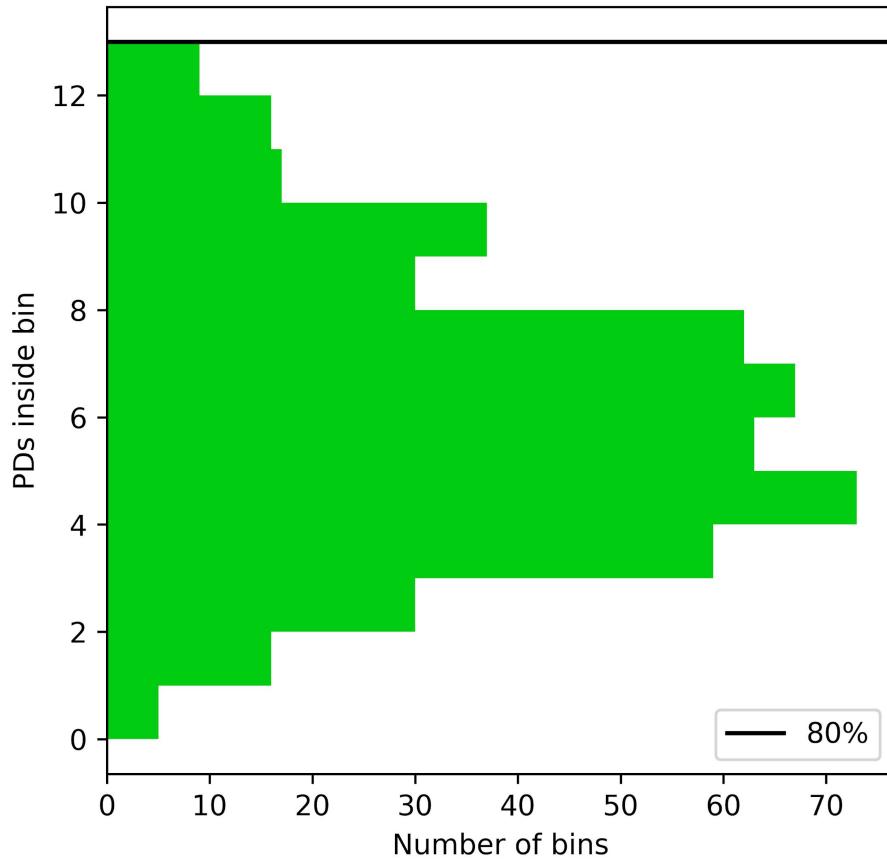


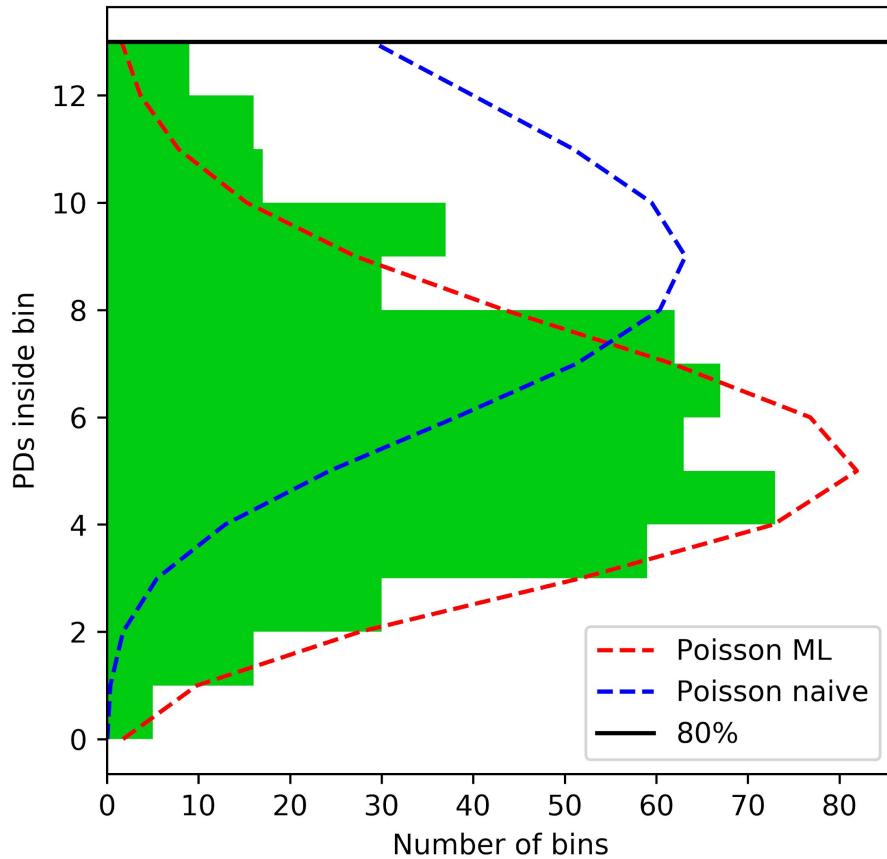


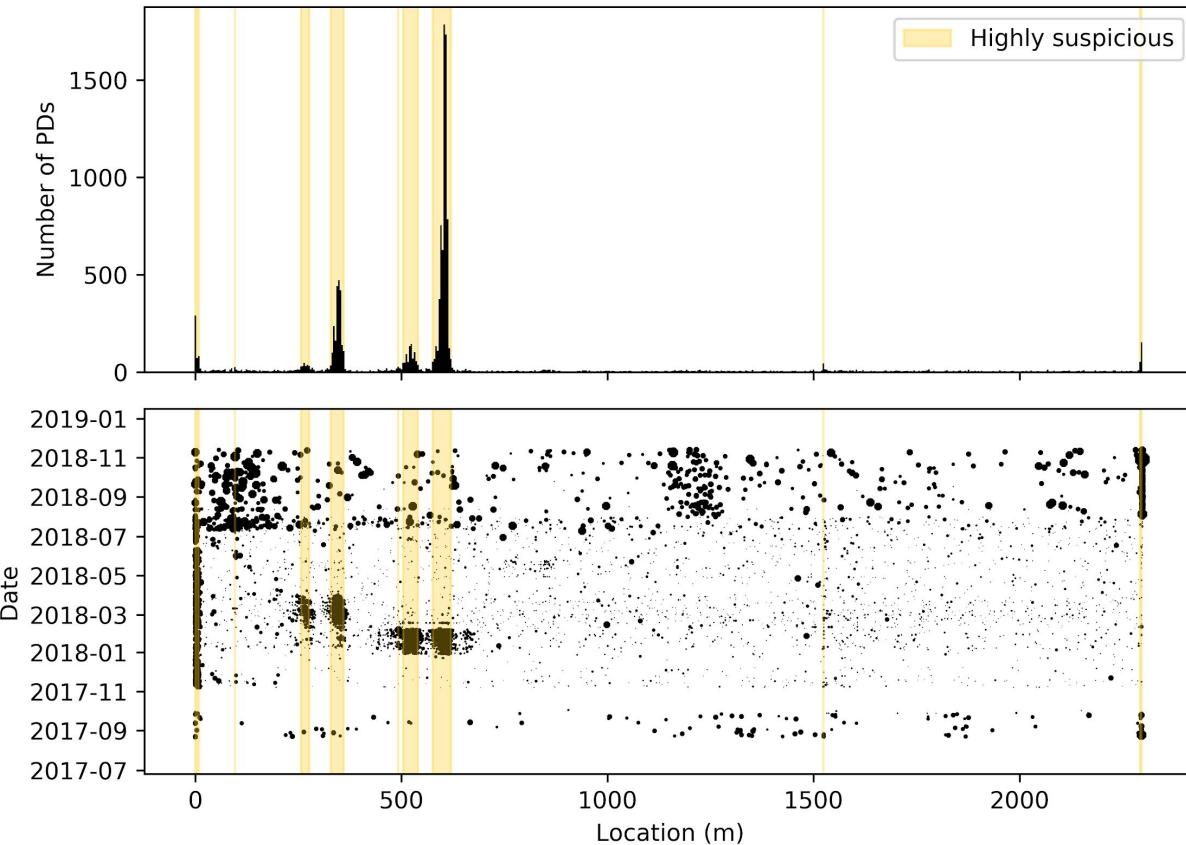




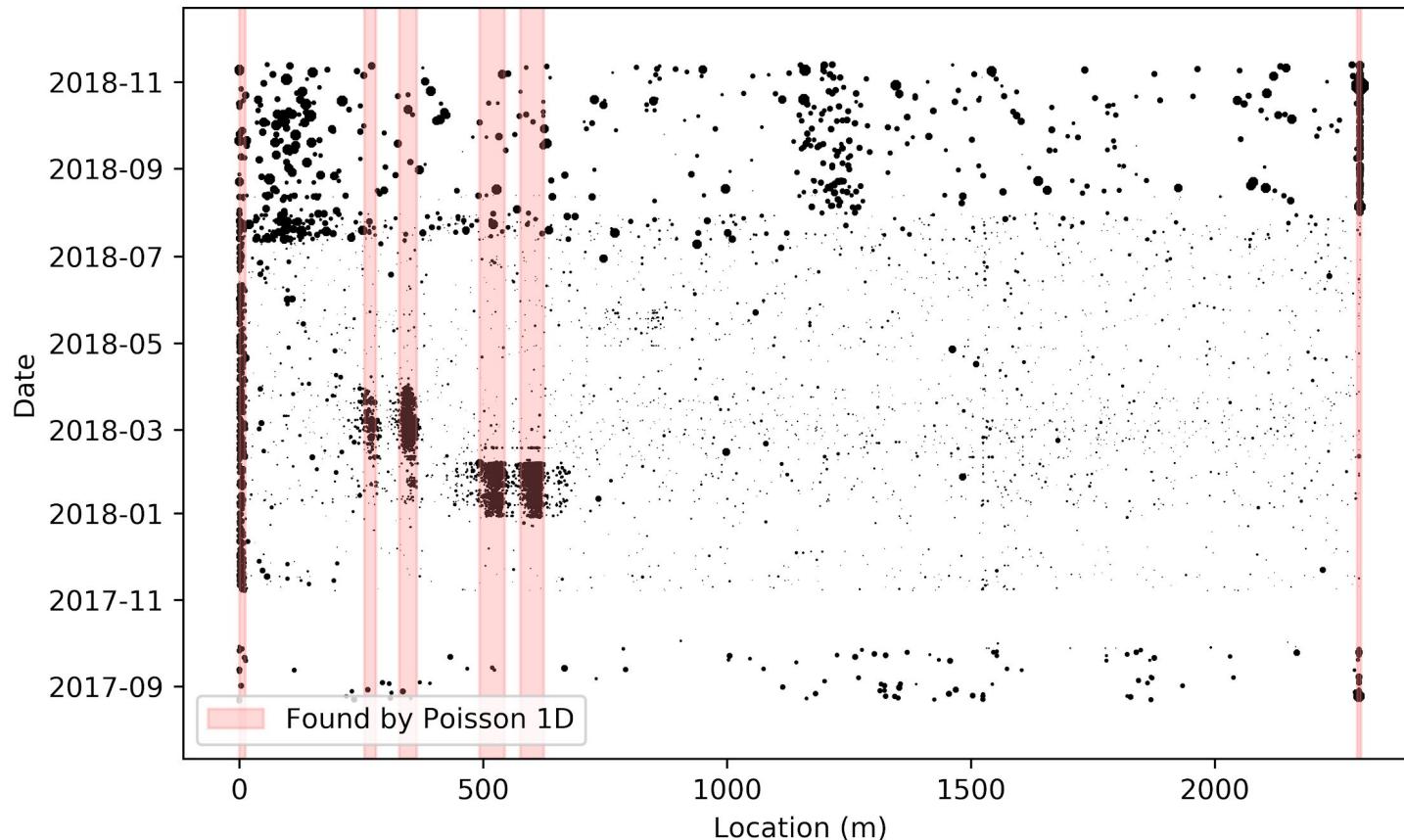








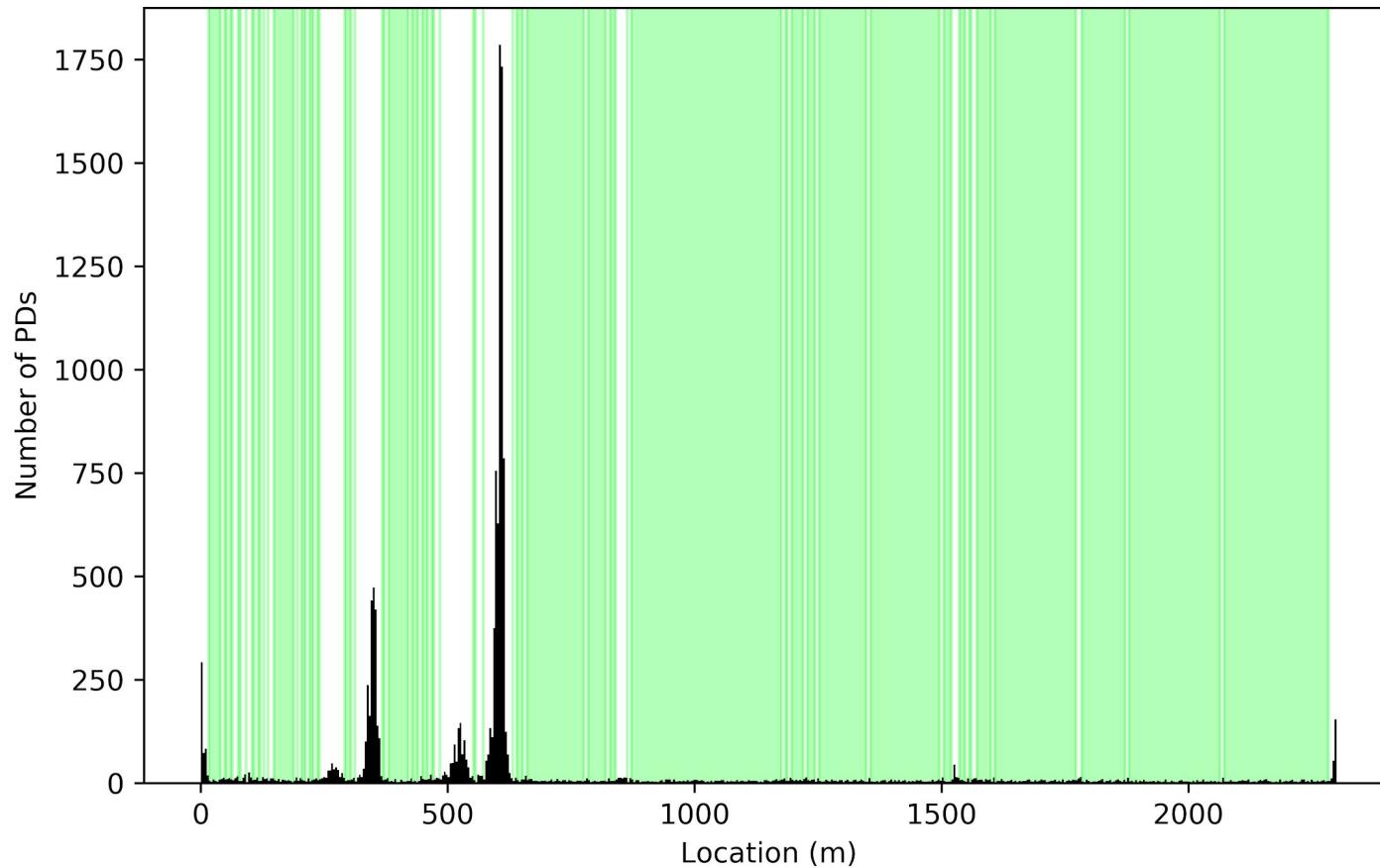
Circuit 2063



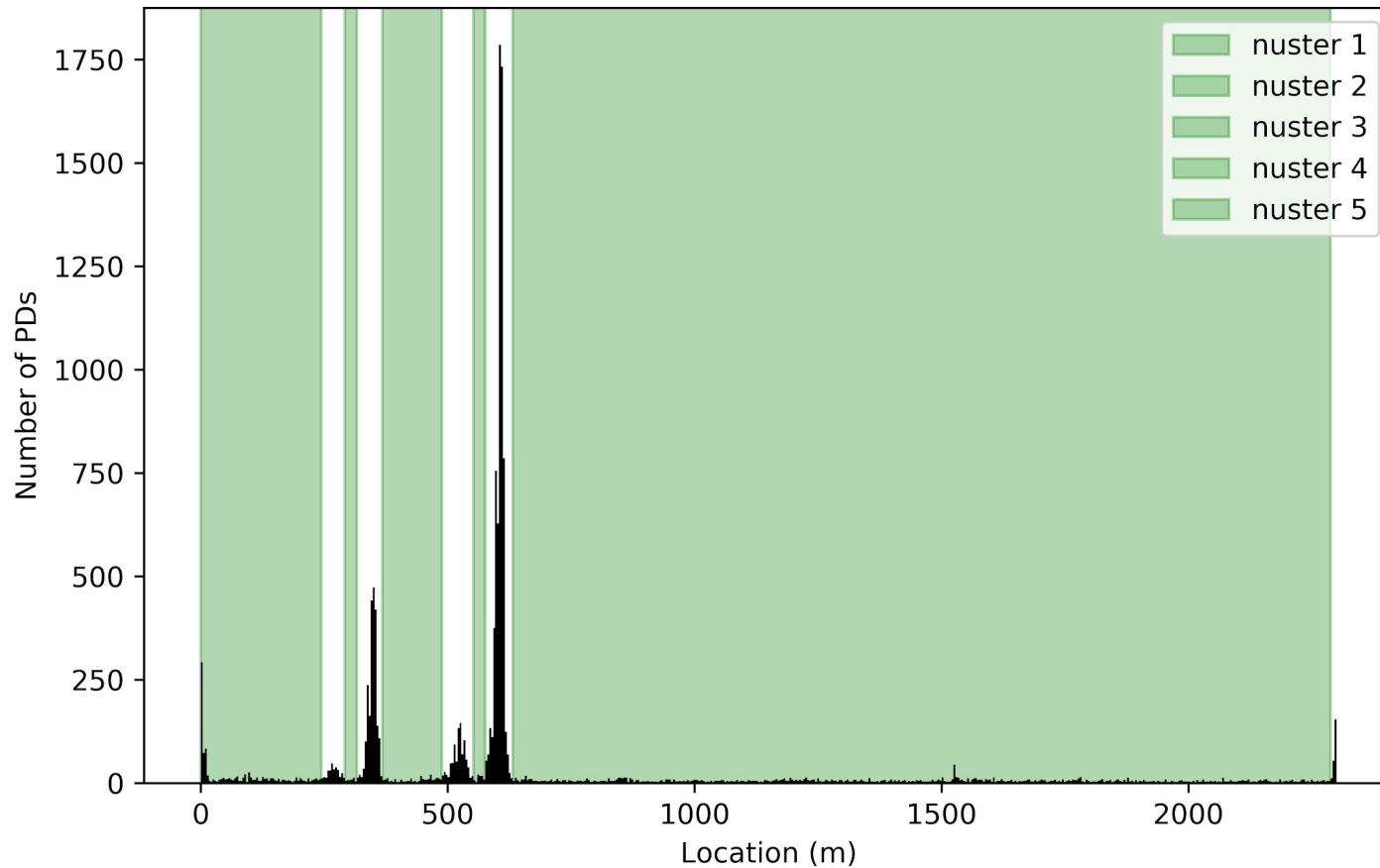
2D

*Elk 1D-cluster wordt
'verfijnd' tot 2D-cluster*

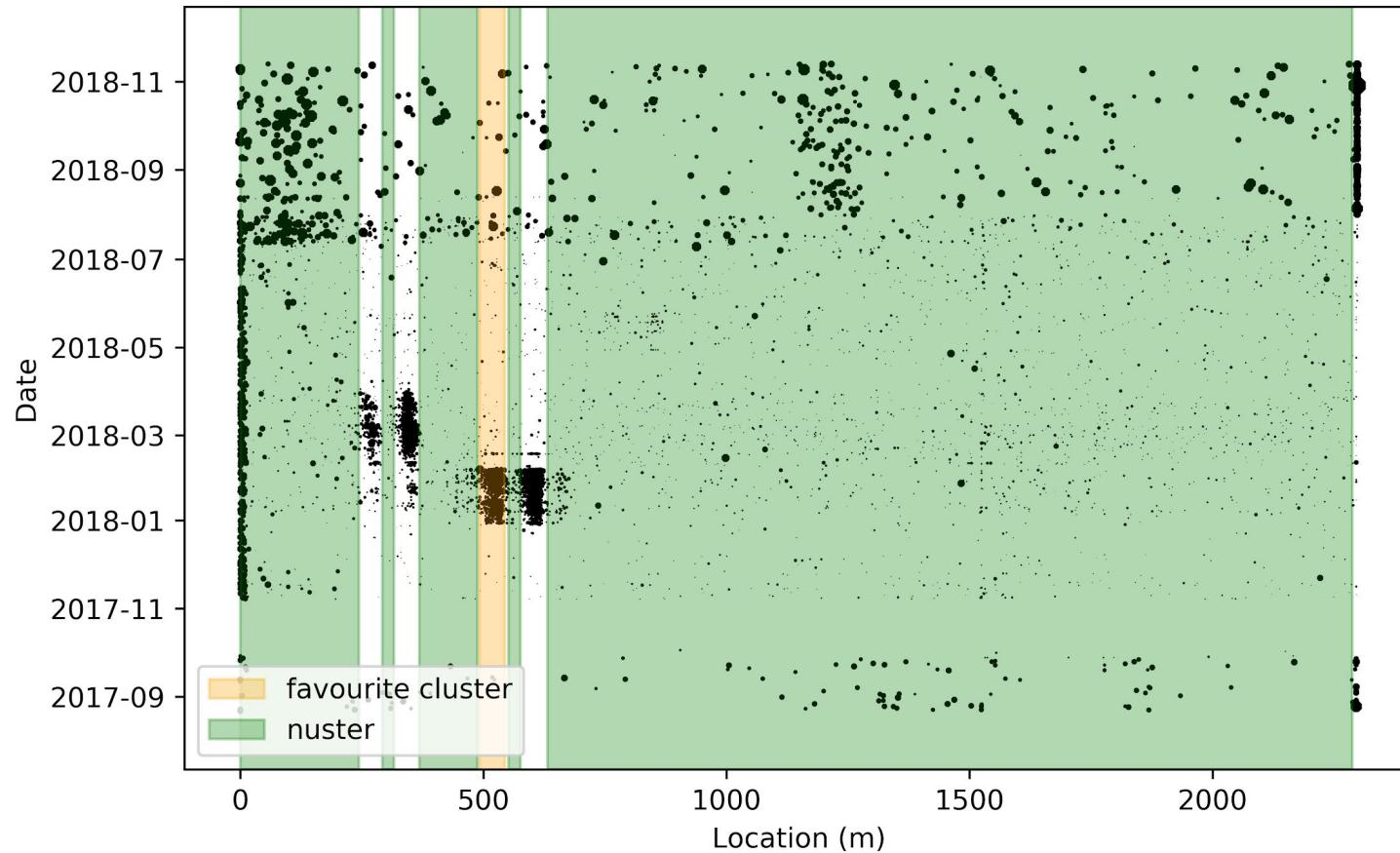
Circuit 2063



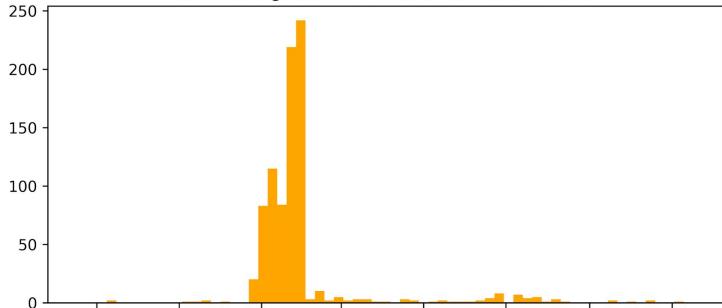
Circuit 2063



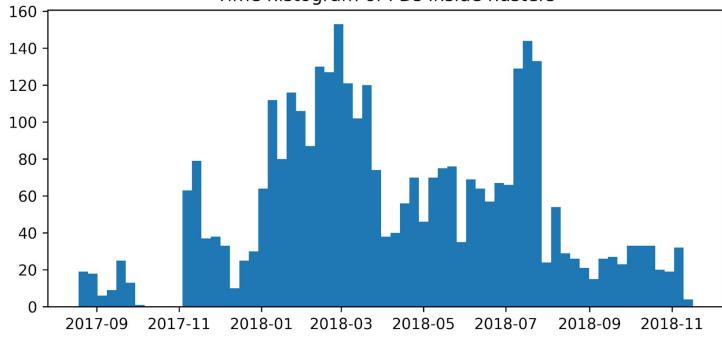
Circuit 2063



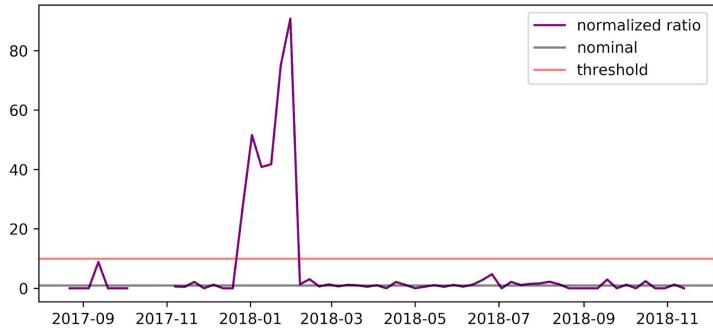
Time histogram of PDs inside favourite cluster



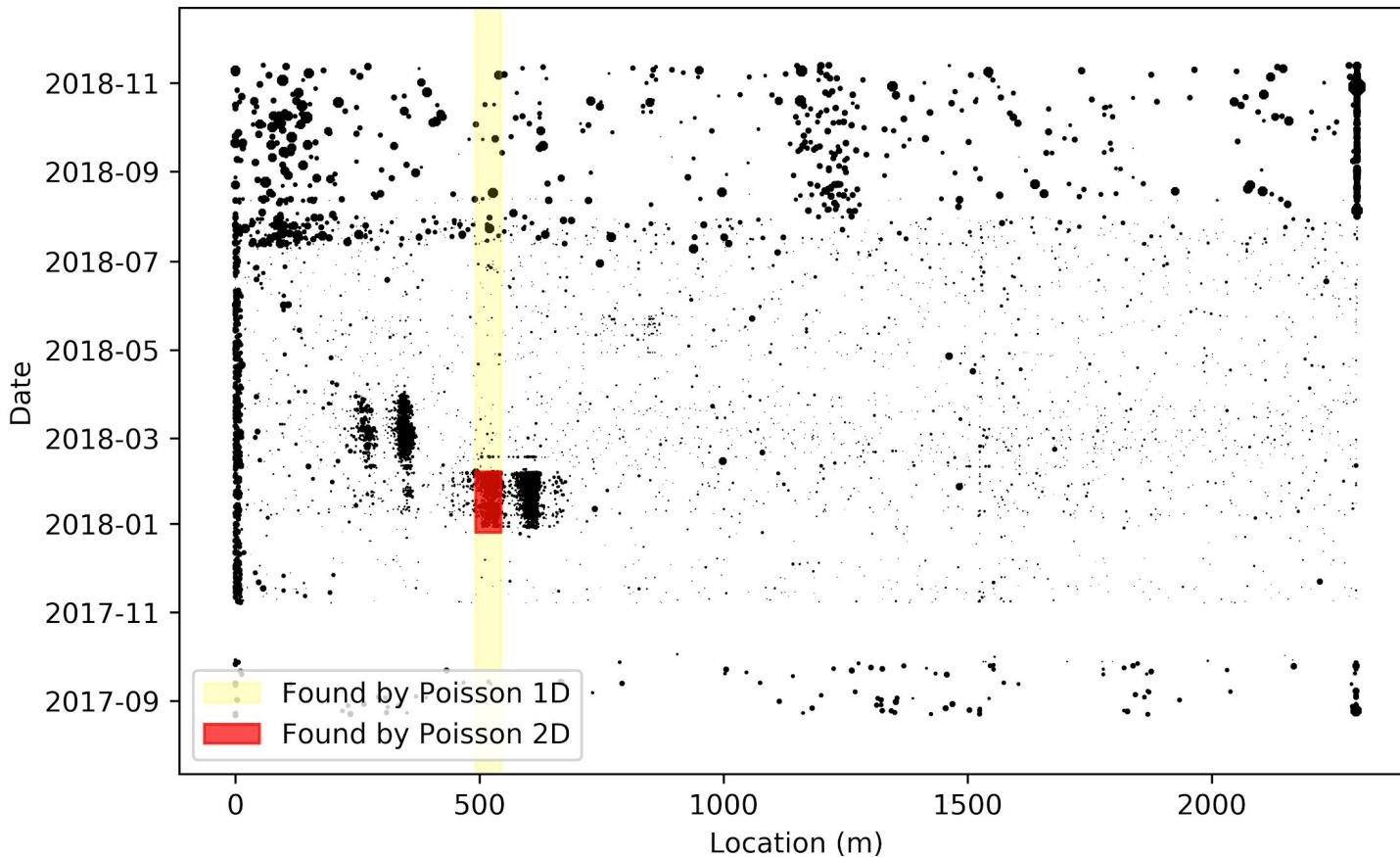
Time histogram of PDs inside nusters



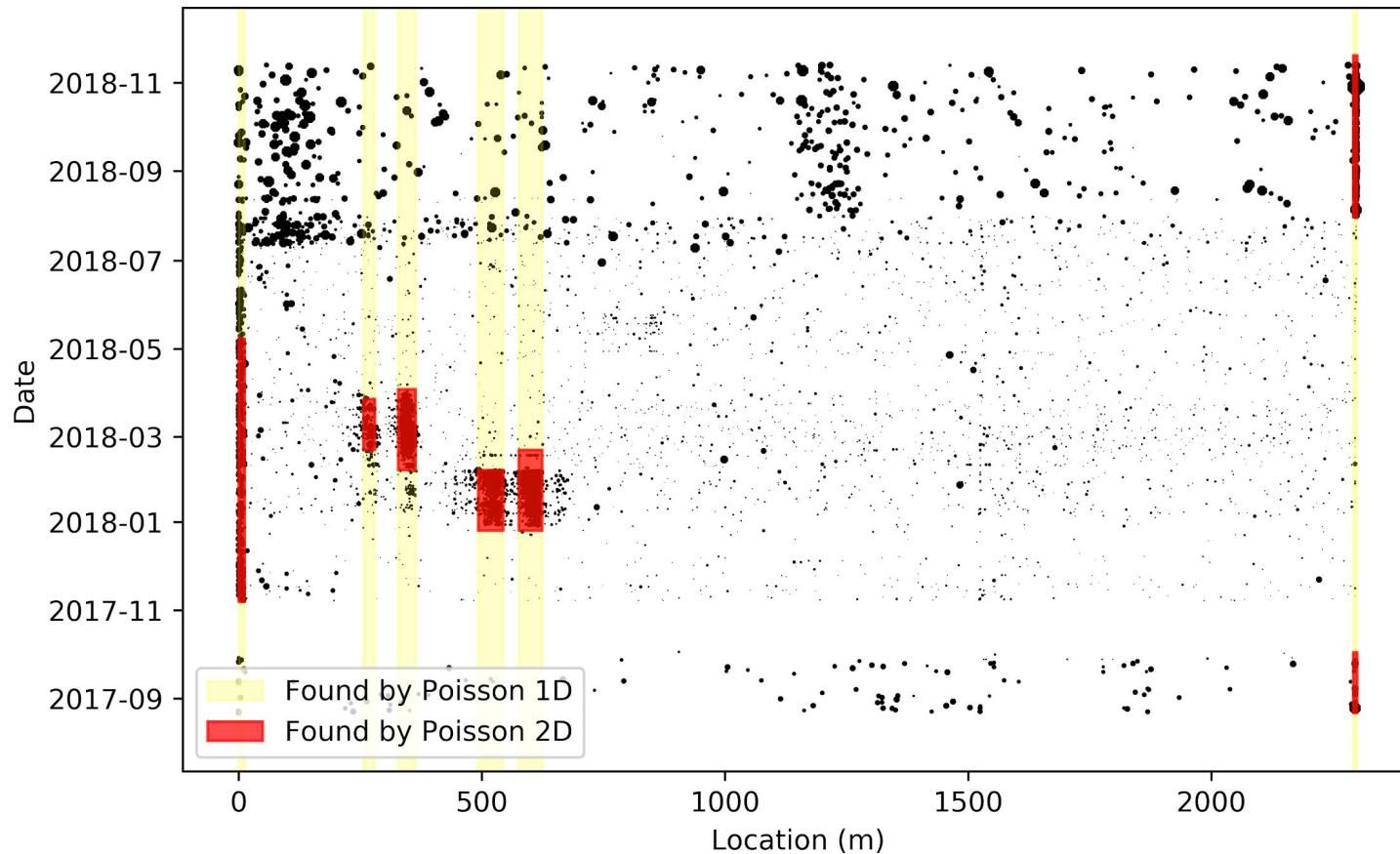
Ratio between two previous histograms



Circuit 2063



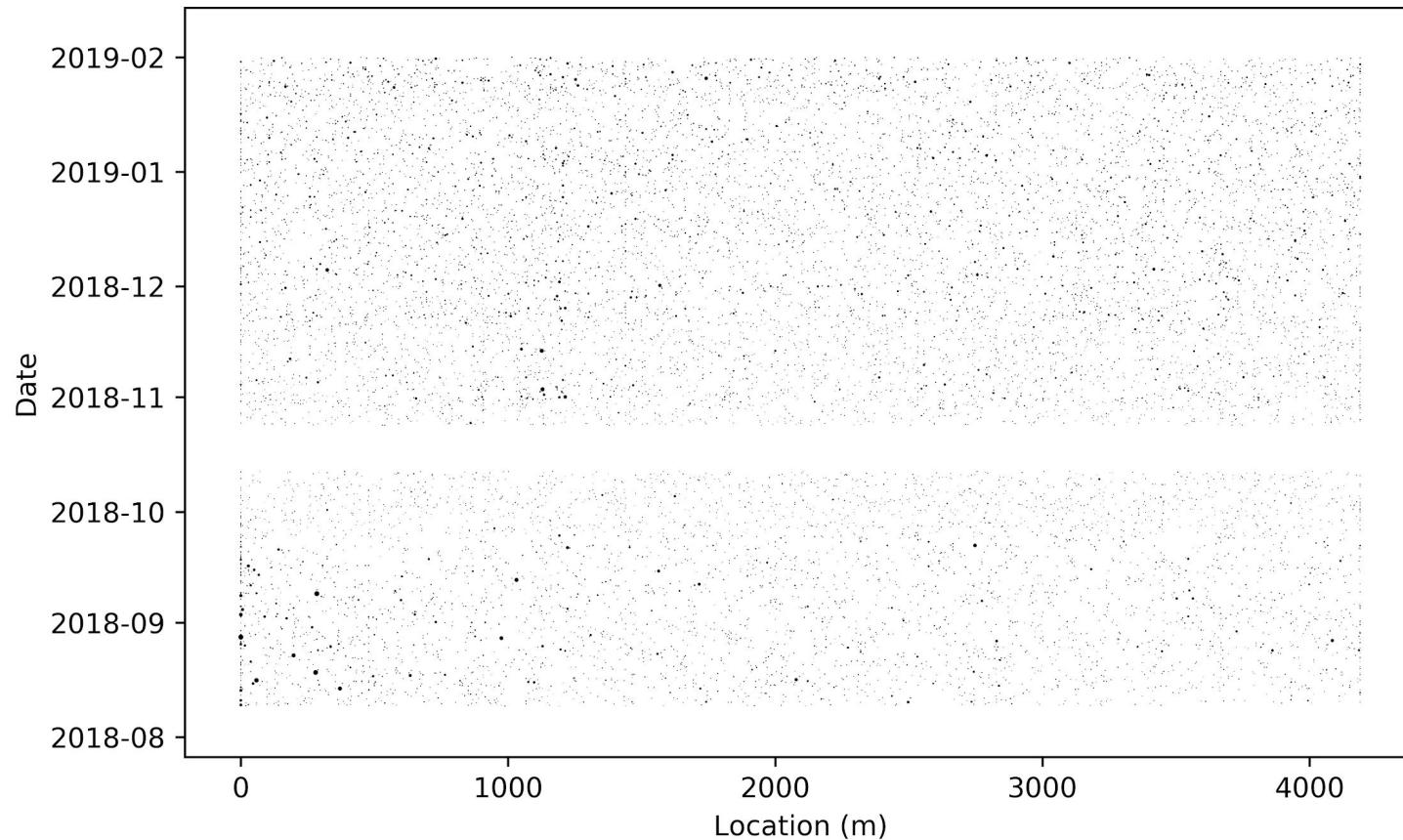
Circuit 2063

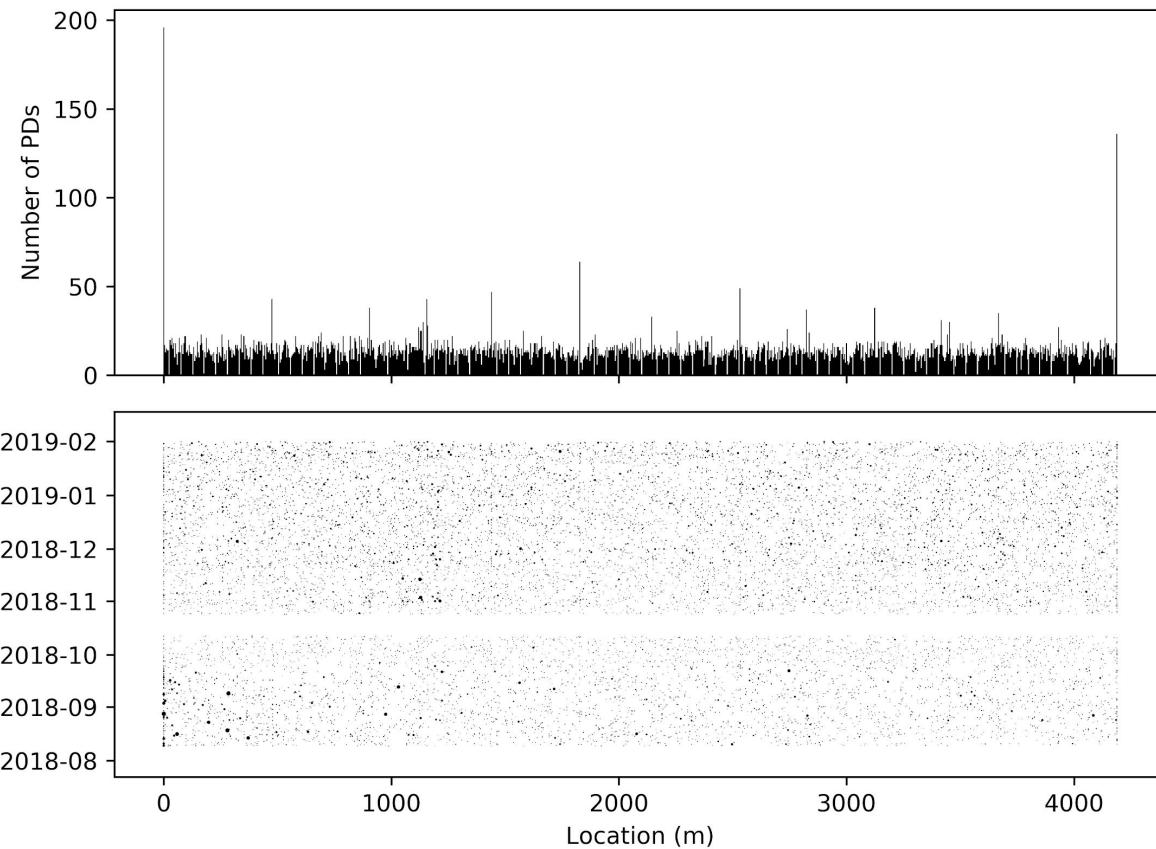


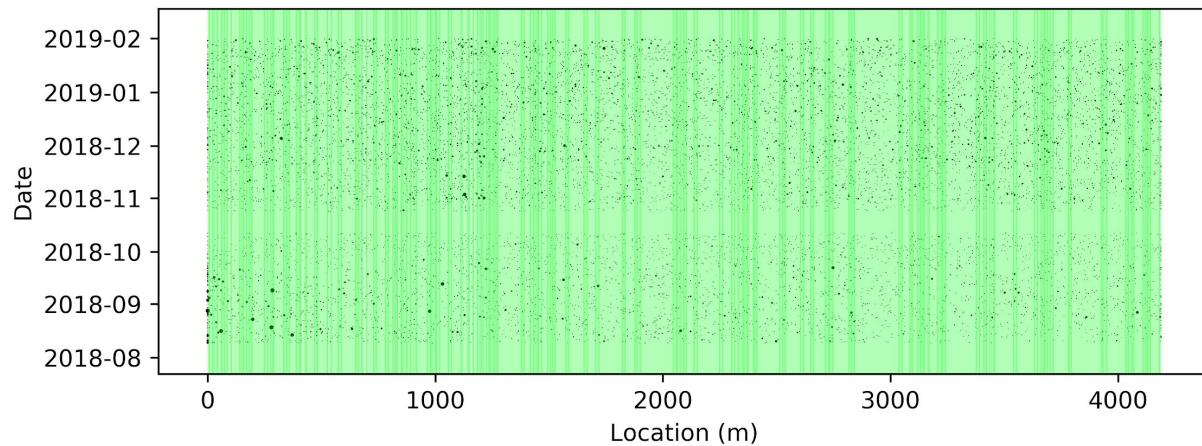
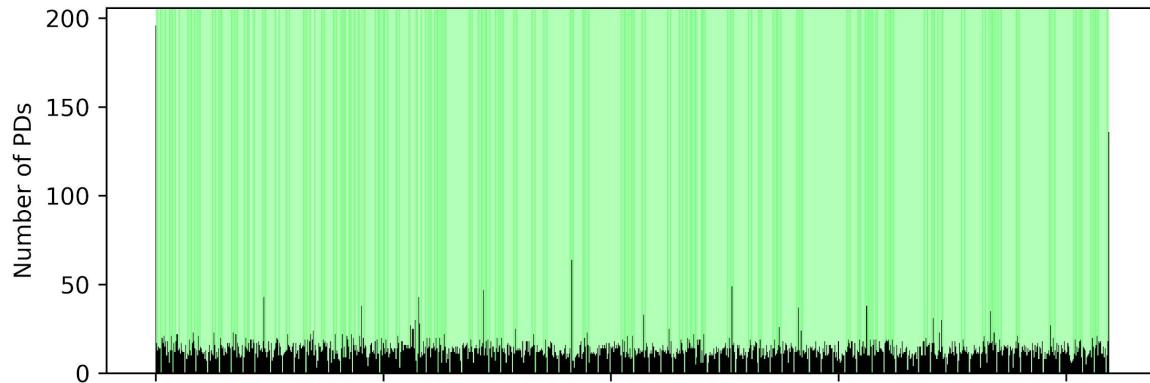
Circuit 3010

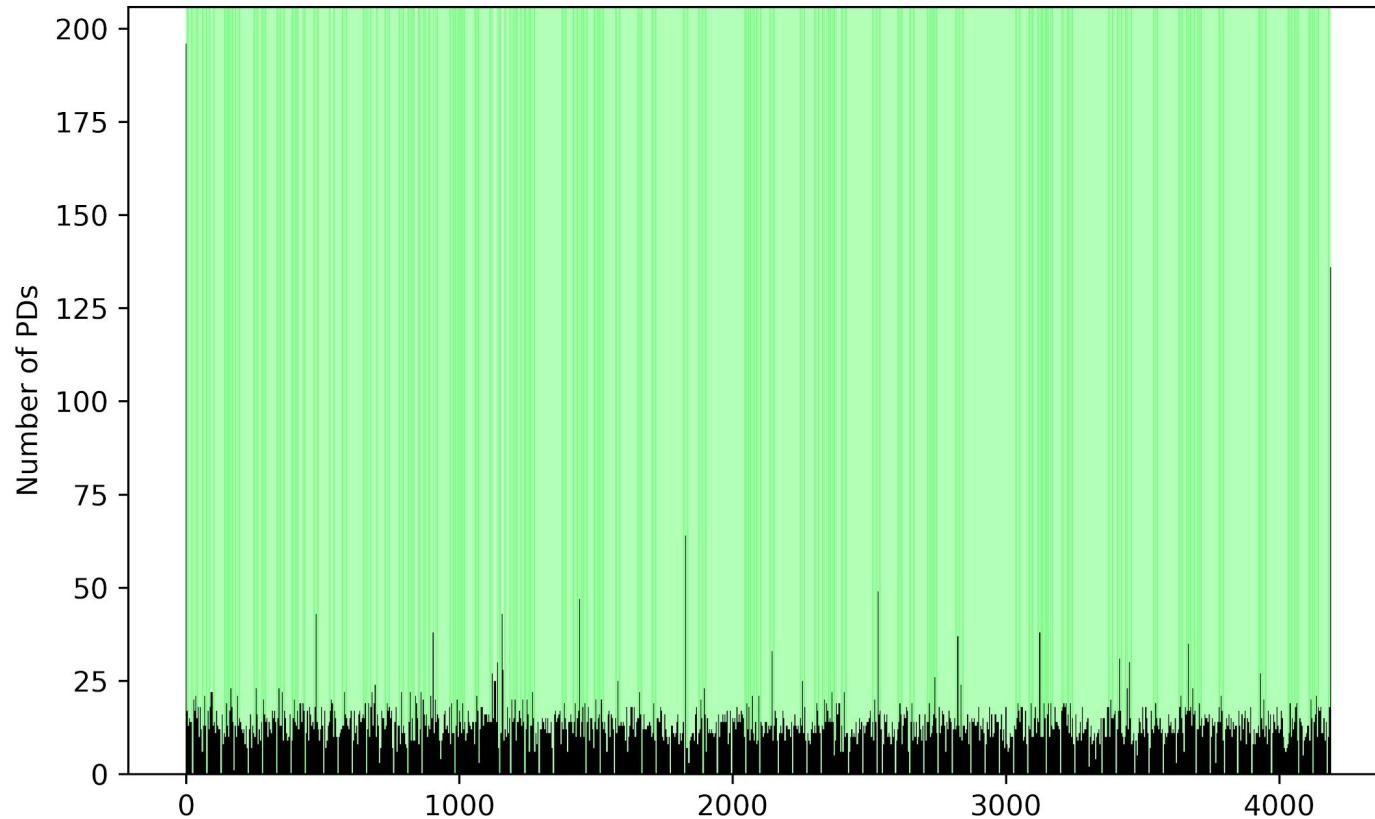
(zonder clusters)

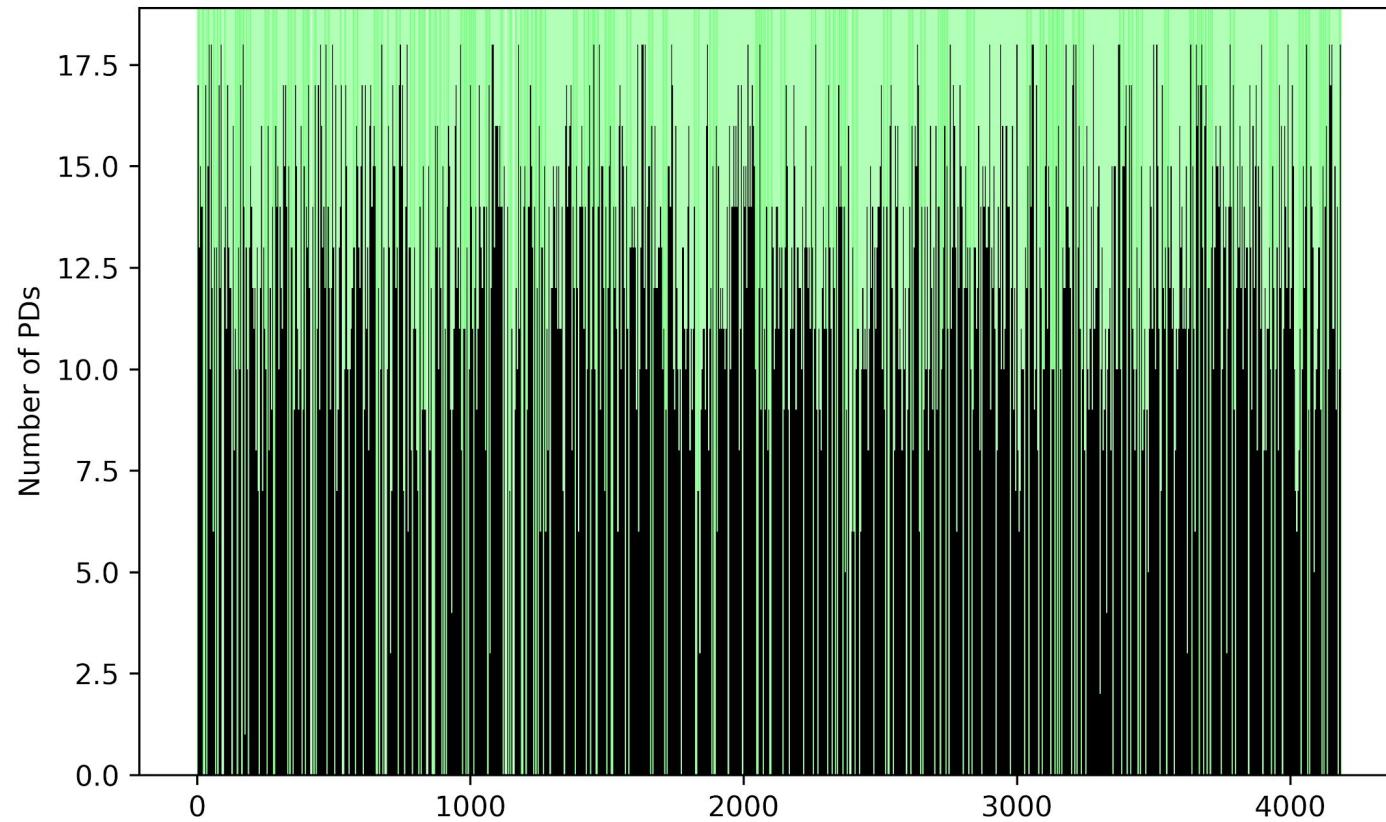
Circuit 3010

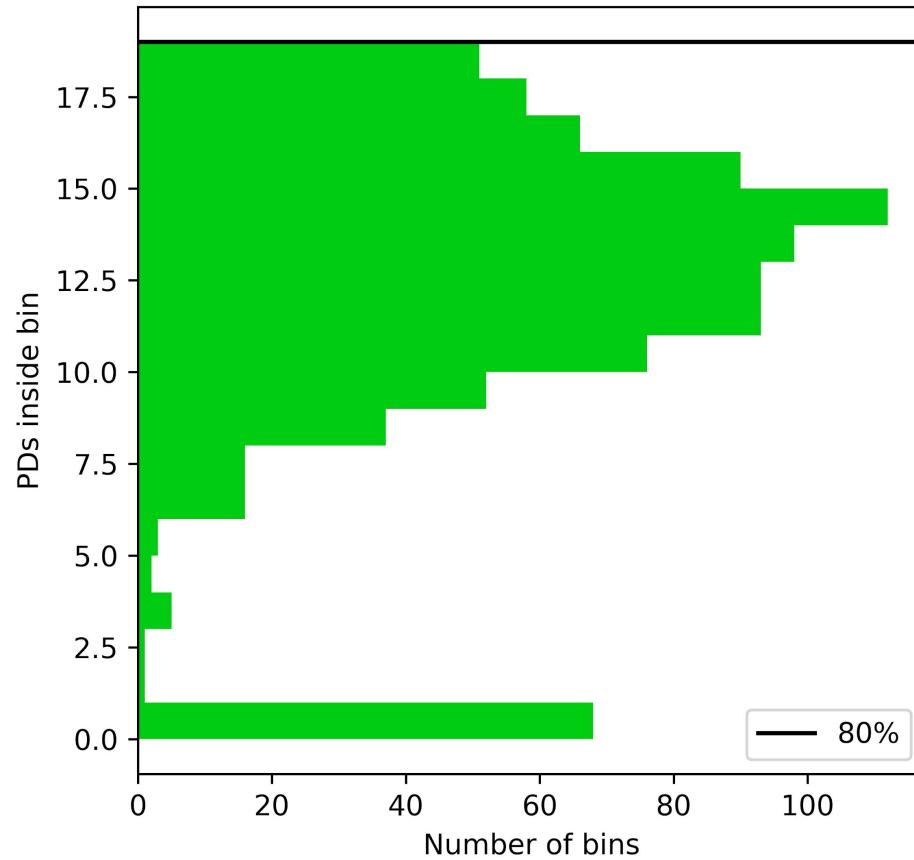


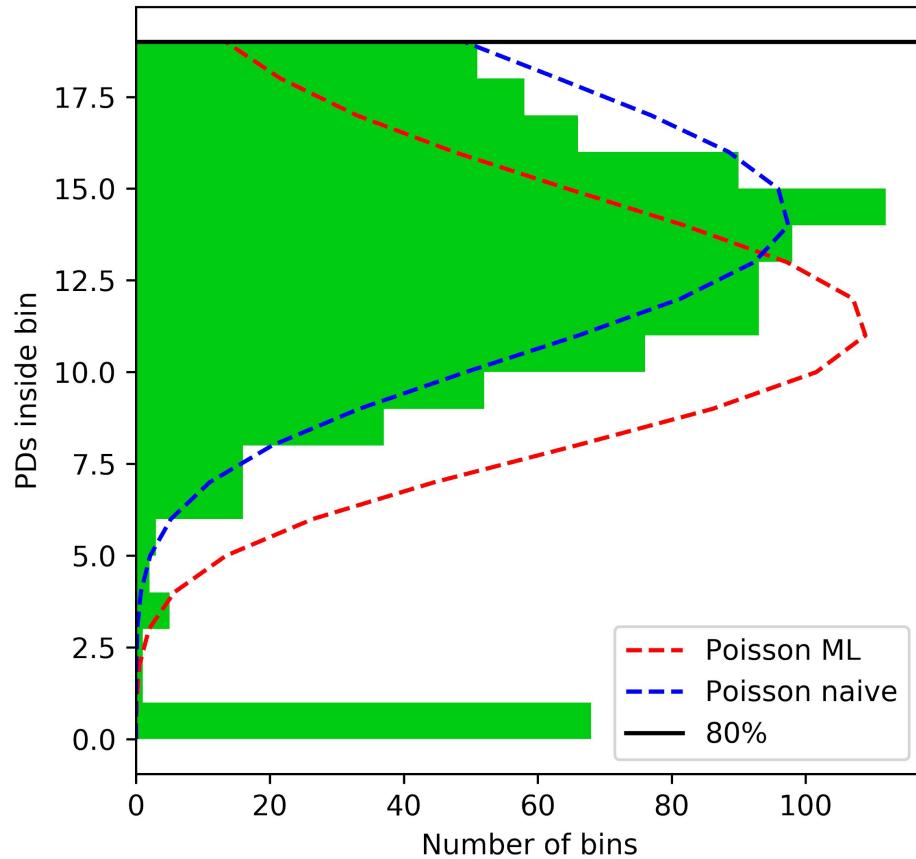


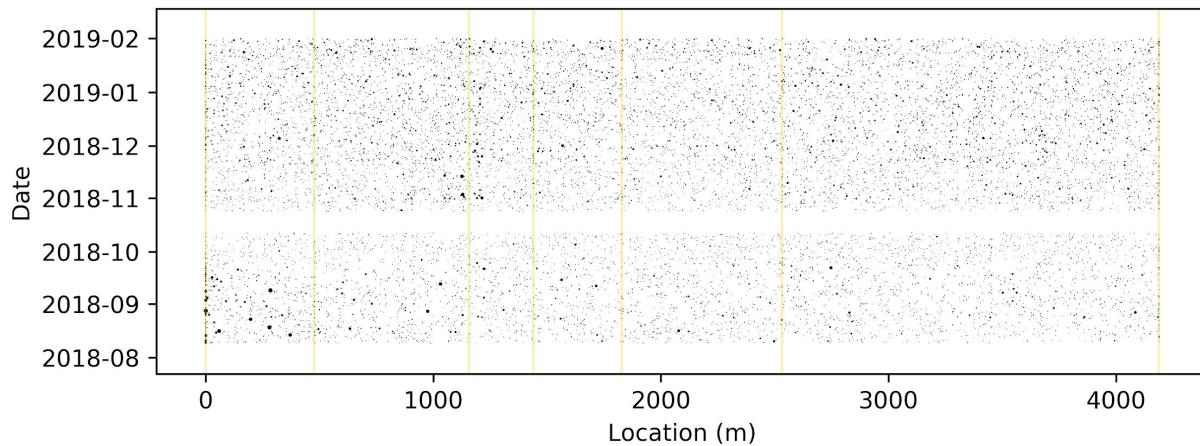
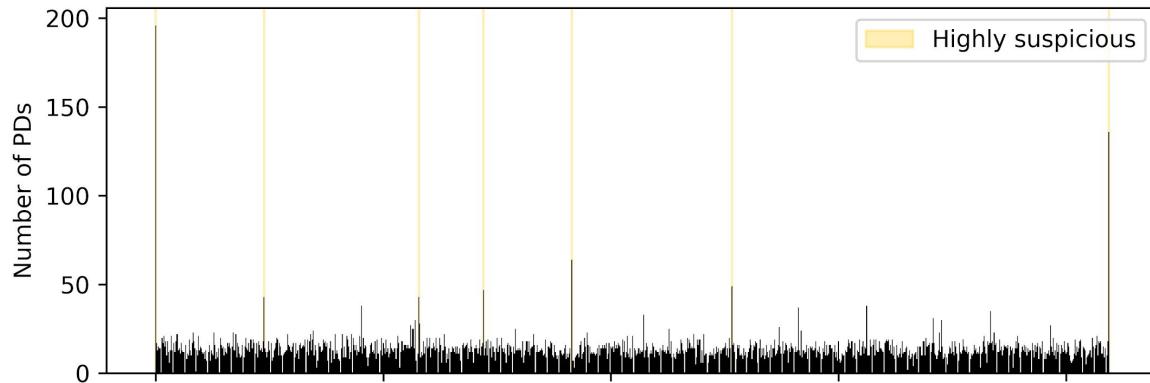




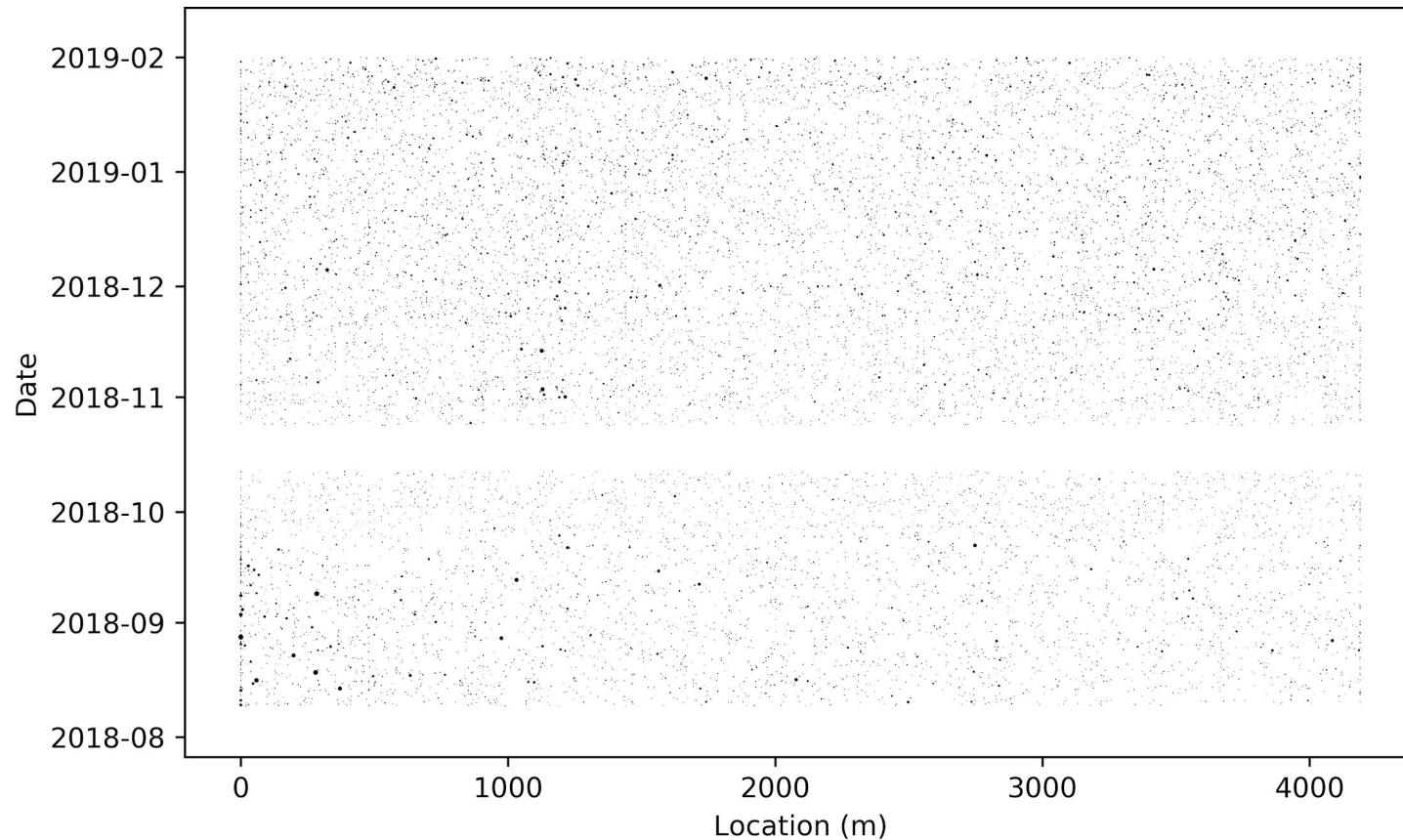








Circuit 3010



Klaar!

*(geen 2D: er zijn geen
1D-clusters)*

Runtime

	 1D	 2D		
Circuit 2063 (gemiddeld)	6.8 ms	9.9 ms	155 ms	12.4 ms
Circuit 3010 (geen clusters)	4.8 ms	4.8 ms	380 ms	29 ms
Circuit 2145 (veel PD's)	17 ms	21 ms	2070 ms	85 ms



DBSCAN-algoritme

Waarom DBSCAN?

- Aantal clusters is **geen** parameter
- Clusters van **verschillende vormen**
- **Ruis** wordt gemakkelijk gedetecteerd



Concept

Het algoritme bestaat uit 4 stappen

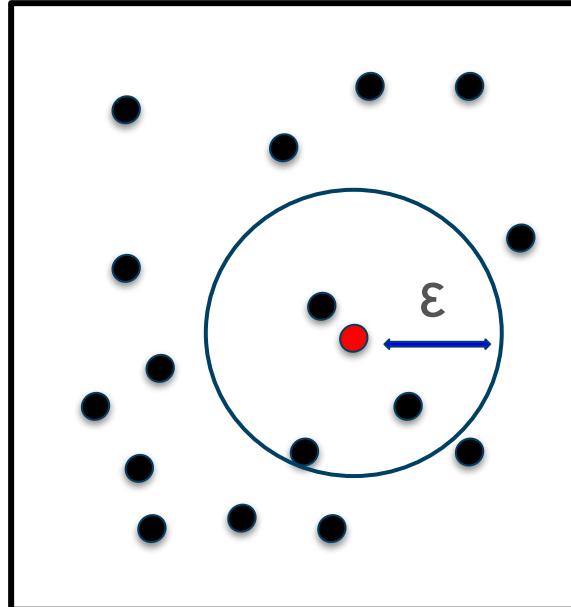
- 1) Zoek een **kernpunt** van een cluster
- 2) Voeg punten die **dichtbij** liggen toe
- 3) Zijn deze punten kernpunten? Zo ja, **breid het cluster uit**
- 4) Herhaal stappen 2 en 3 tot het cluster af is
- 5) Ga terug naar stap 1, of stop als alle punten een **label** hebben

Concept

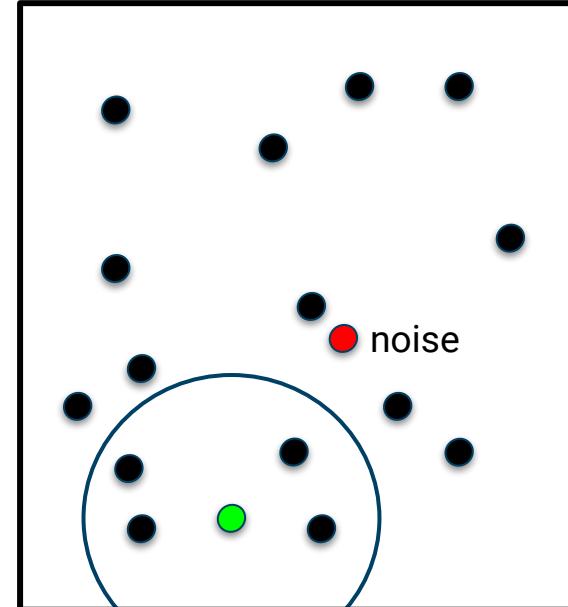
Stap 1

- Bekijk de **epsilon**-omgeving van een willekeurig punt
- Genoeg punten → kernpunt, begin een nieuw cluster
- Parameters: epsilon, MinPts, metriek

niet genoeg punten



wel genoeg punten!



nog een keer

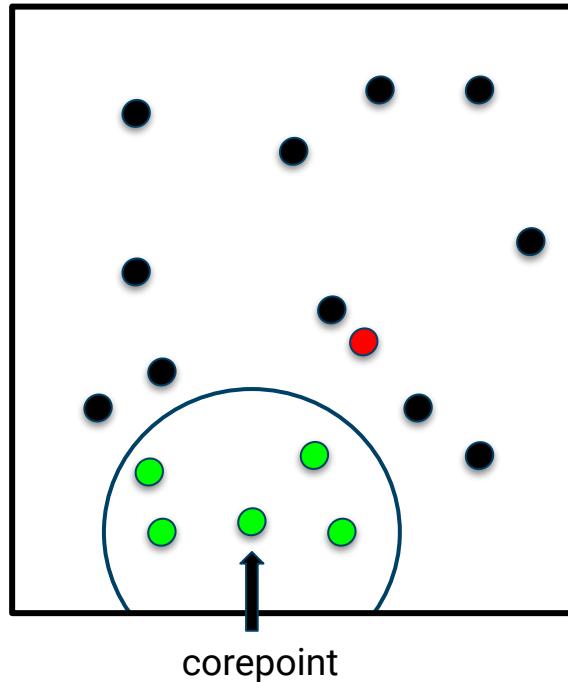
MinPts = 5

Concept

Stap 2

- Voeg alle punten binnen de epsilon-omgeving toe aan het cluster

nieuw cluster begonnen

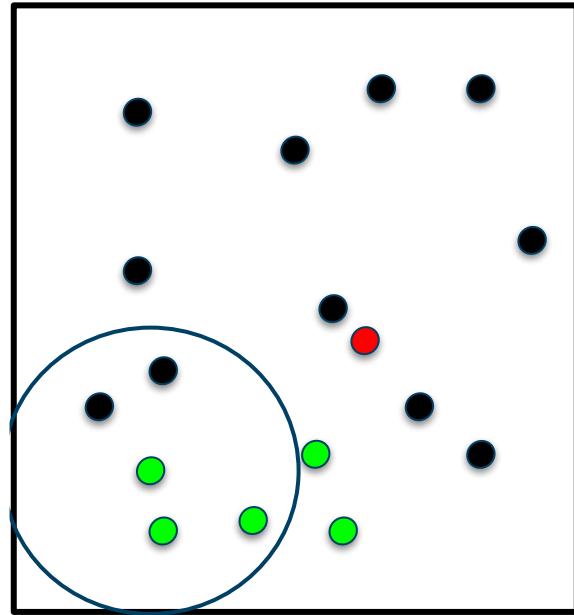


Concept

Stap 3

- Bekijk de epsilon-omgeving van een punt in het cluster
- **Kernpunt of randpunt?**
- Breid het cluster uit

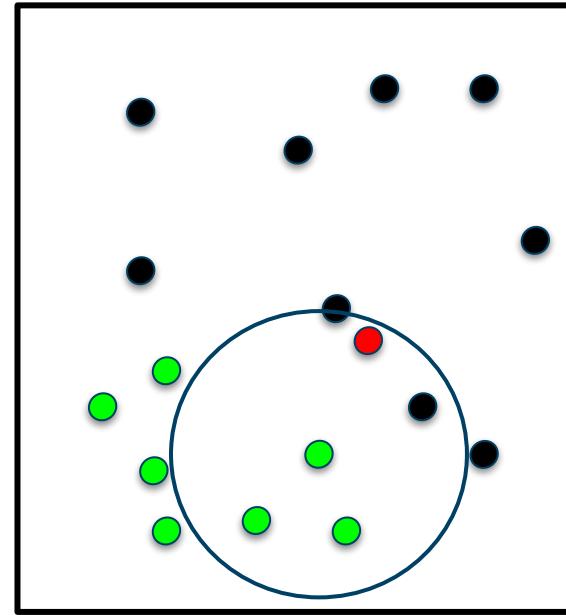
cluster wordt uitgebreid



volgende punt

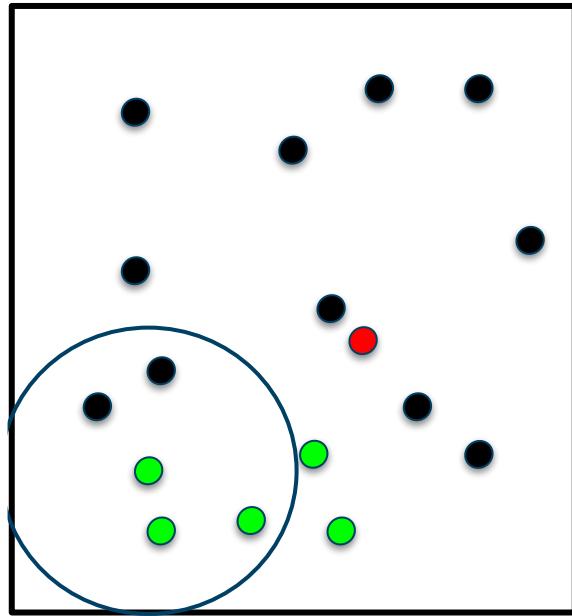


wel of geen clusterpunt?

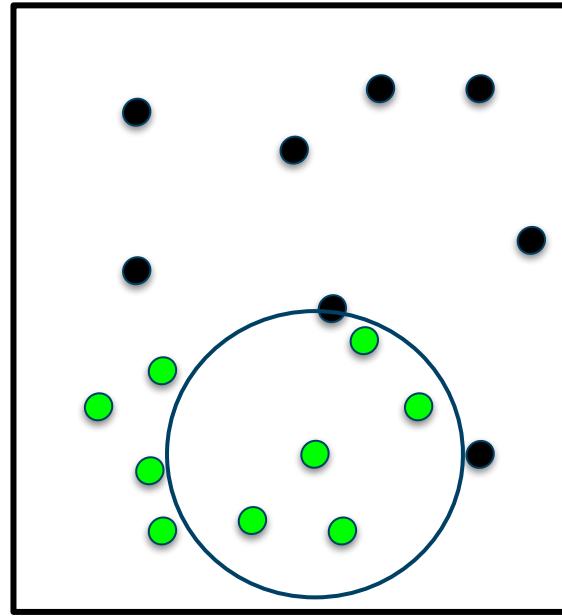


cluster wordt uitgebreid

wel of geen clusterpunt?



volgende punt
→

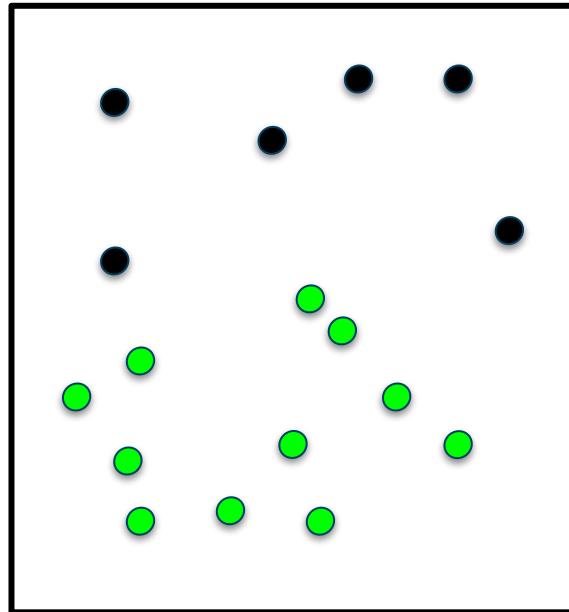


Concept

Stap 4

- Zoek naar een punt in het cluster **zonder label**
- Niet mogelijk? Dan is het cluster af

Het cluster is af!

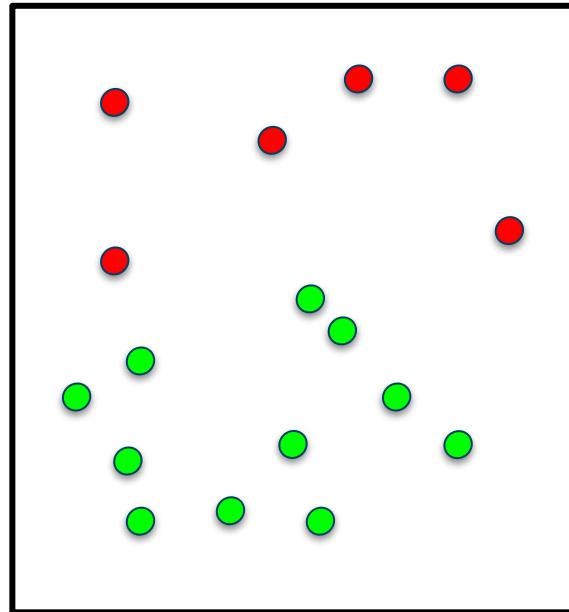


Concept

Stap 5

- Vind een punt zonder label
- Ieder punt gelabeld?
- Labels: **kernpunt, randpunt, ruis**

Alle punten gelabeld



Concept



gebaseerd op: <https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/>

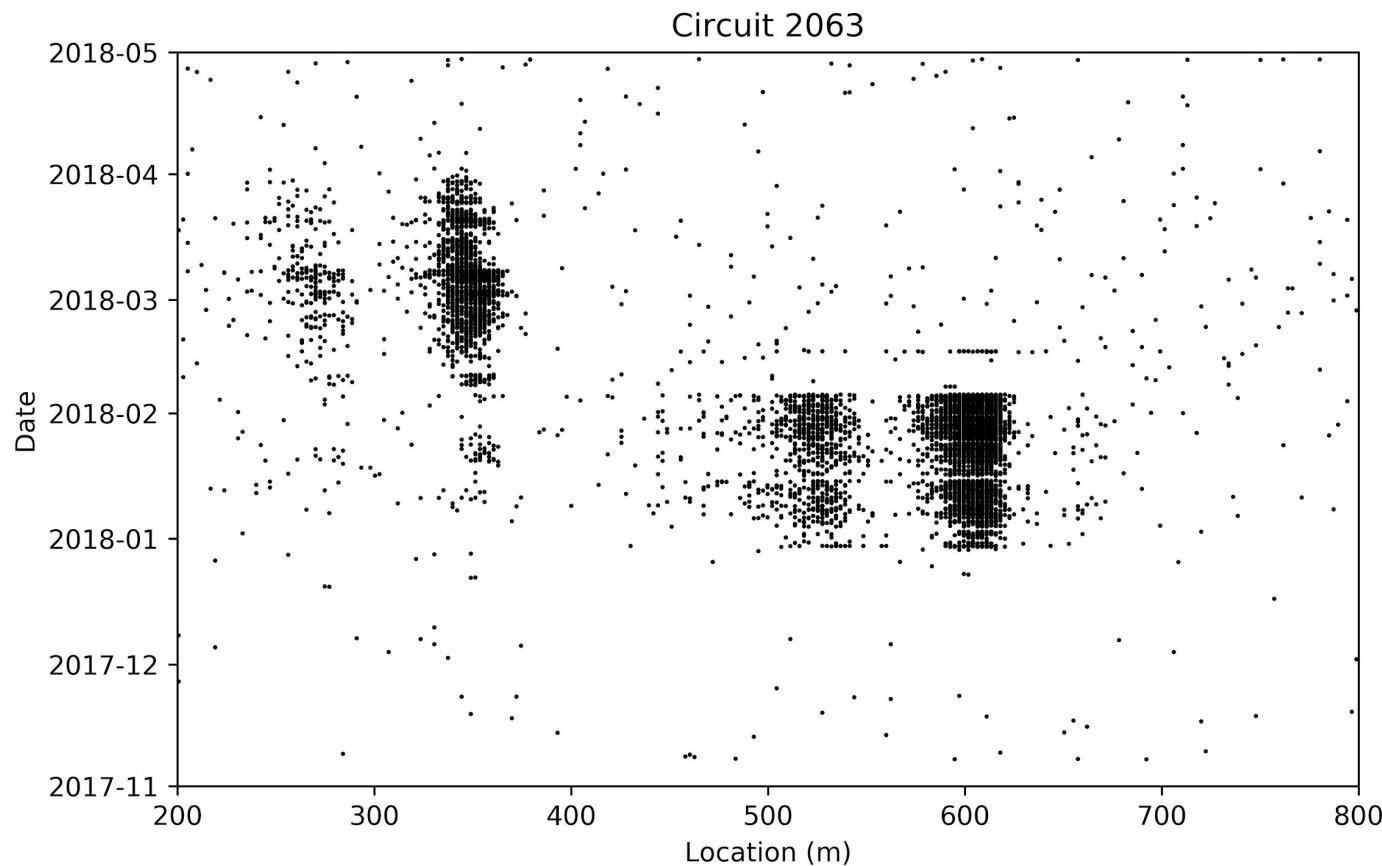
Radboud University



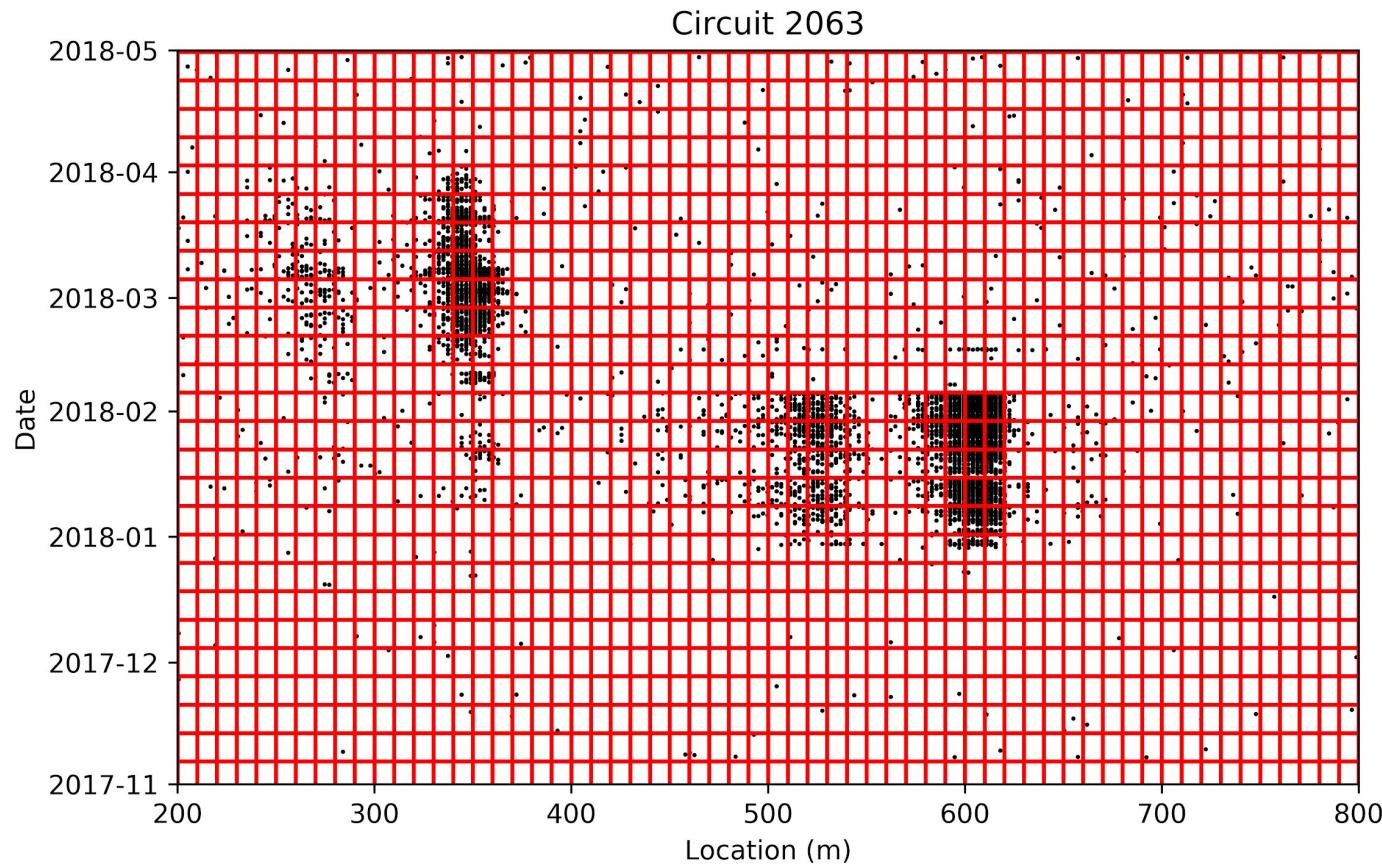
Implementatie

- Hoe passen we dit algoritme toe op onze data?
- Eerst **discretiseren** voor betere runtime

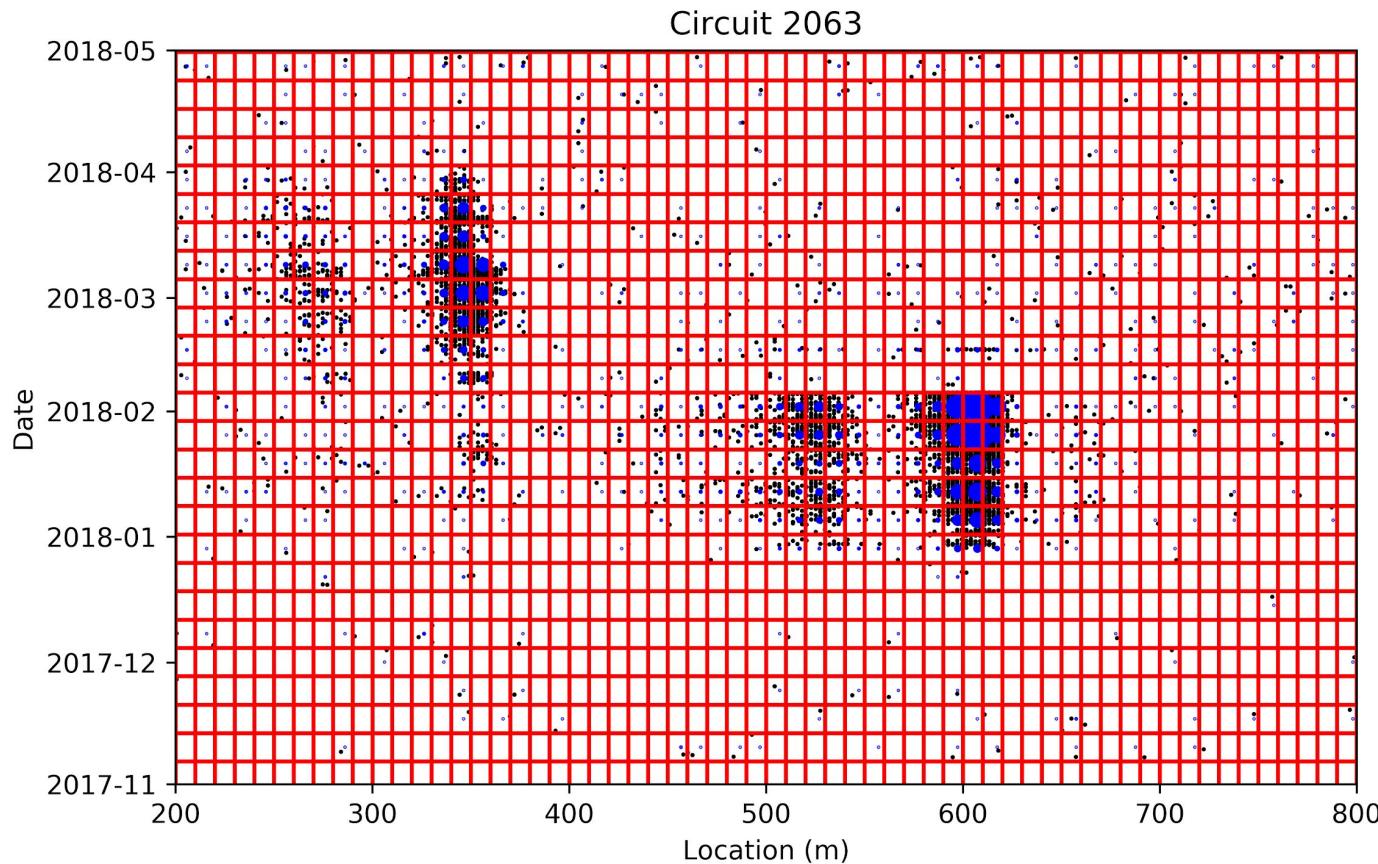
De data



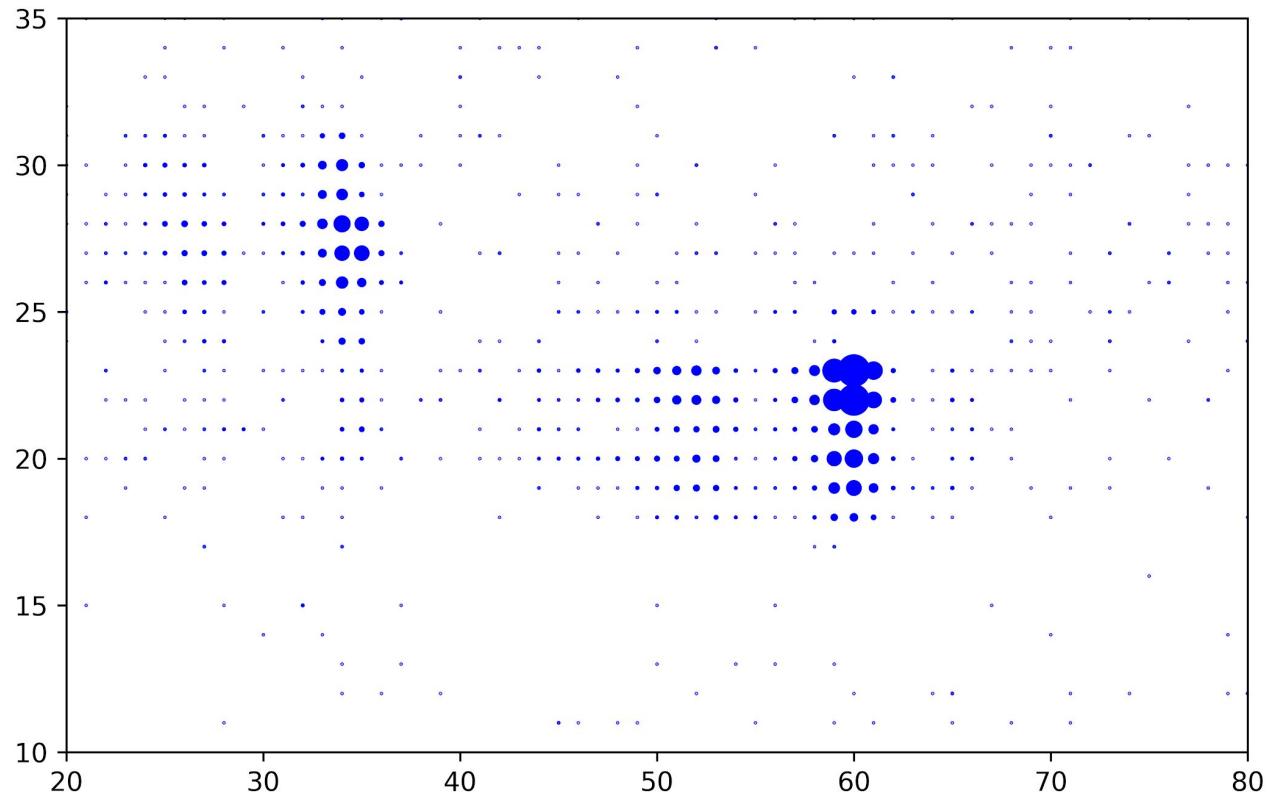
Discretiseren



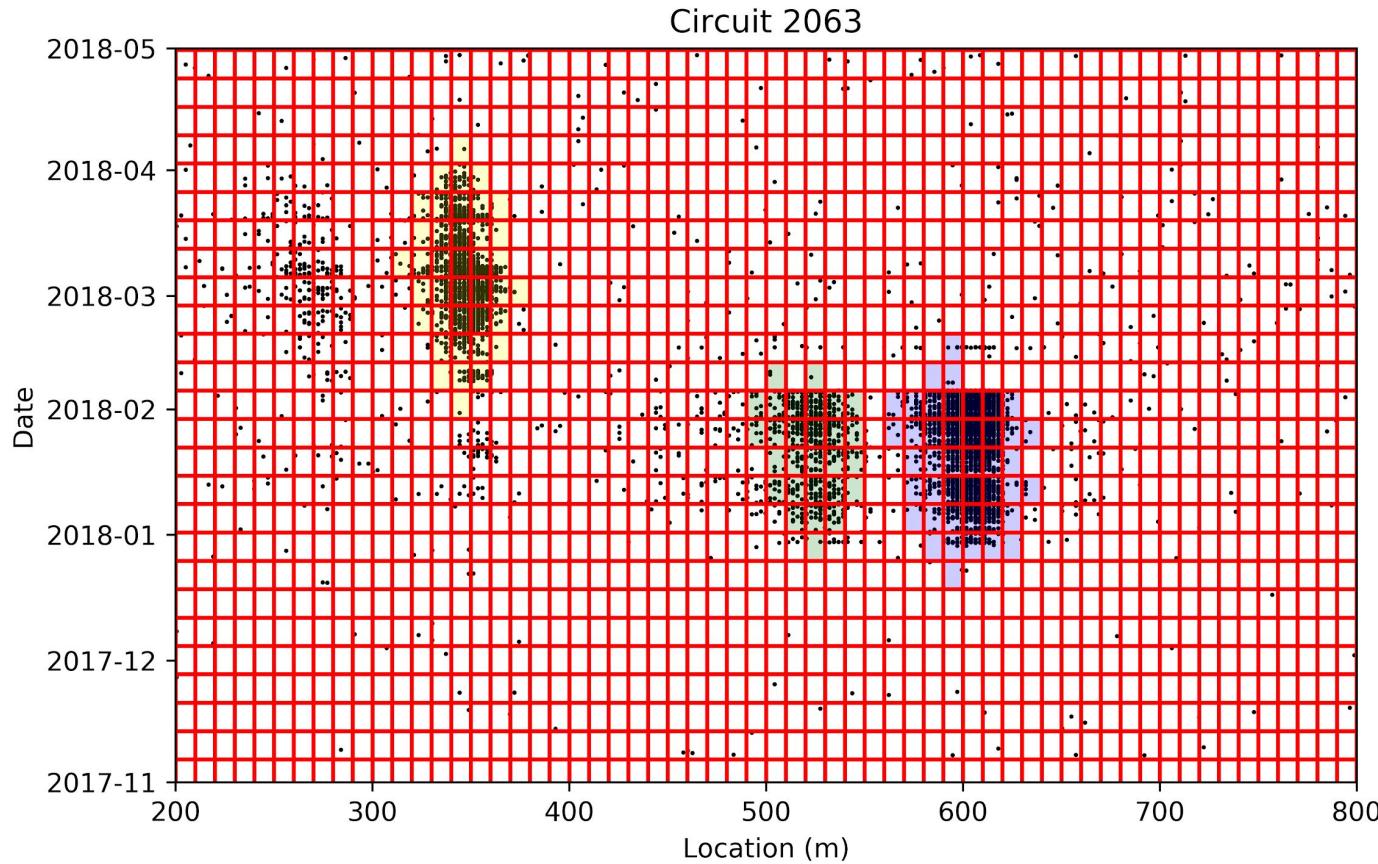
Van vakjes naar punten



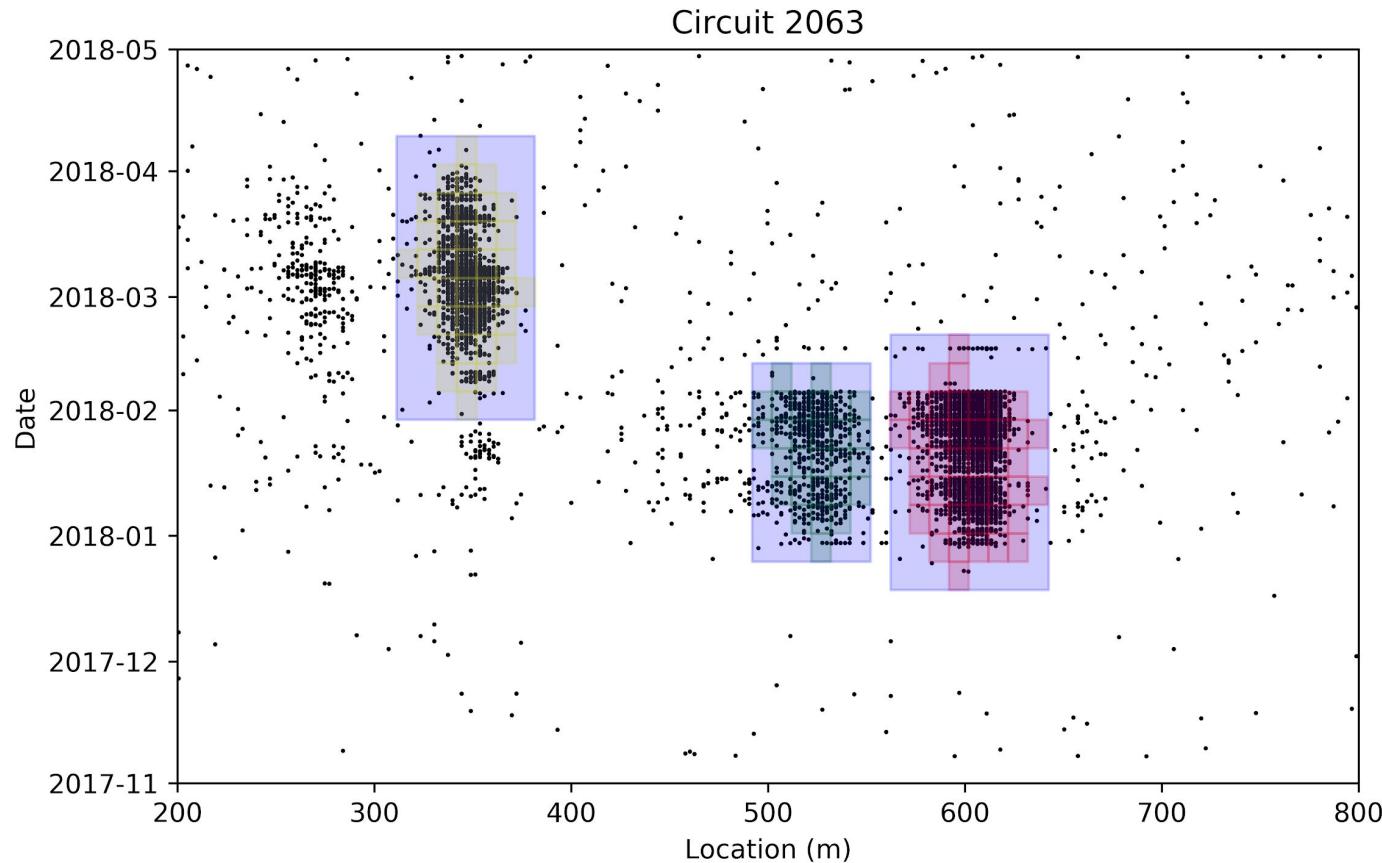
Herschalen



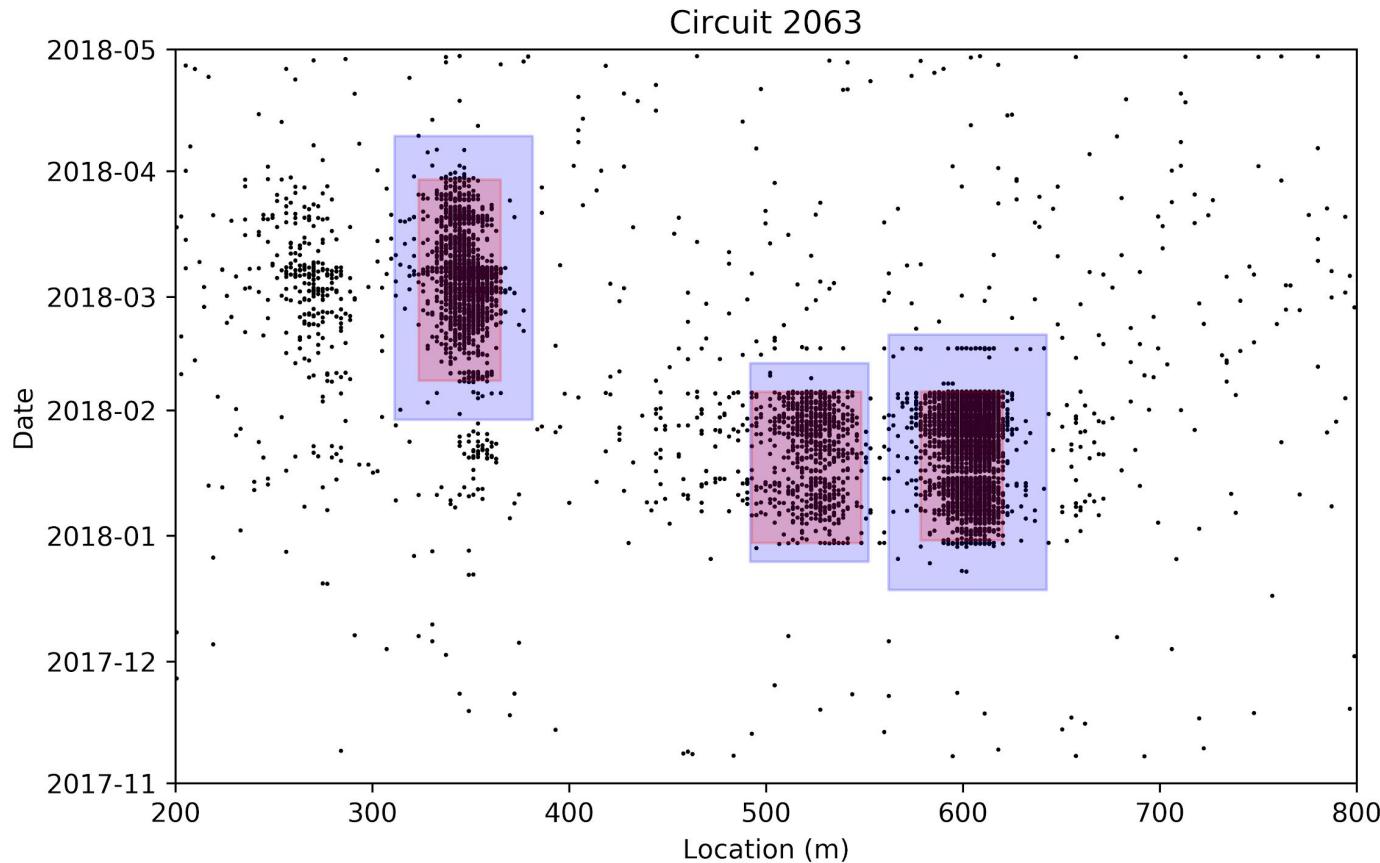
DBSCAN uitvoeren



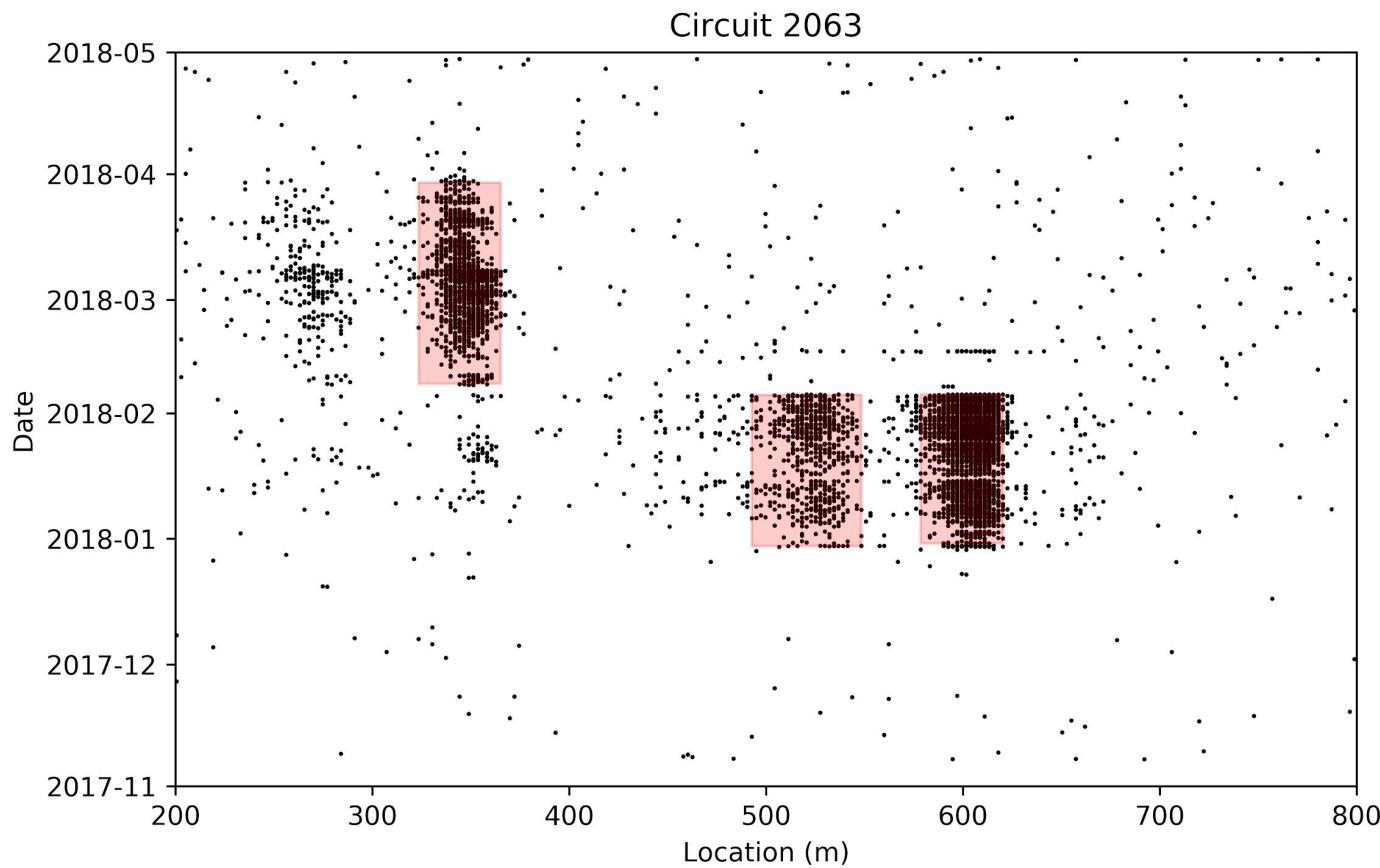
De ‘bounding box’



Clusters bijschaven



Het resultaat



Parameters

- Epsilon = 3
- MinPts = 125
- Metriek = Euclidian
- binLengthX = 2
- binLengthY = 1
- Shave = 0.01
- Circuit

Runtime

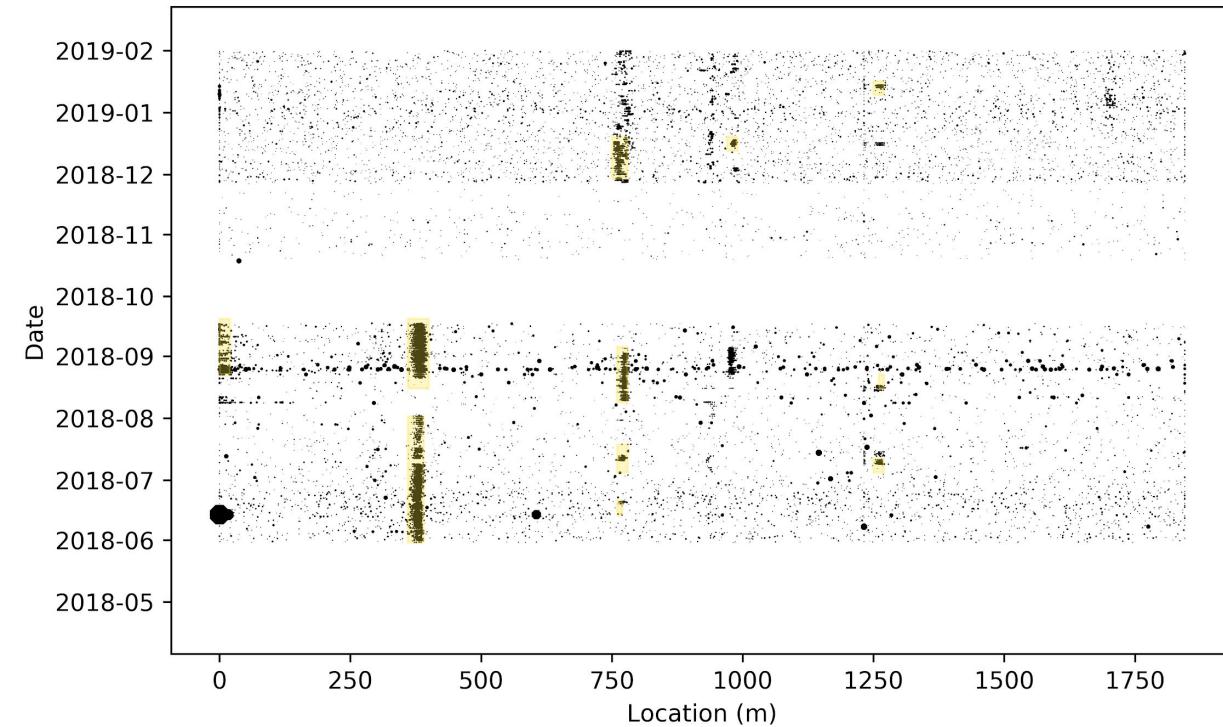
			
Circuit 2063 (gemiddeld)	9 . 9 ms	155 ms	12 . 4 ms
Circuit 3010 (geen clusters)	4 . 8 ms	380 ms	29 ms
Circuit 2145 (veel PD's)	21 ms	2070 ms	85 ms



Pinta-Algoritme

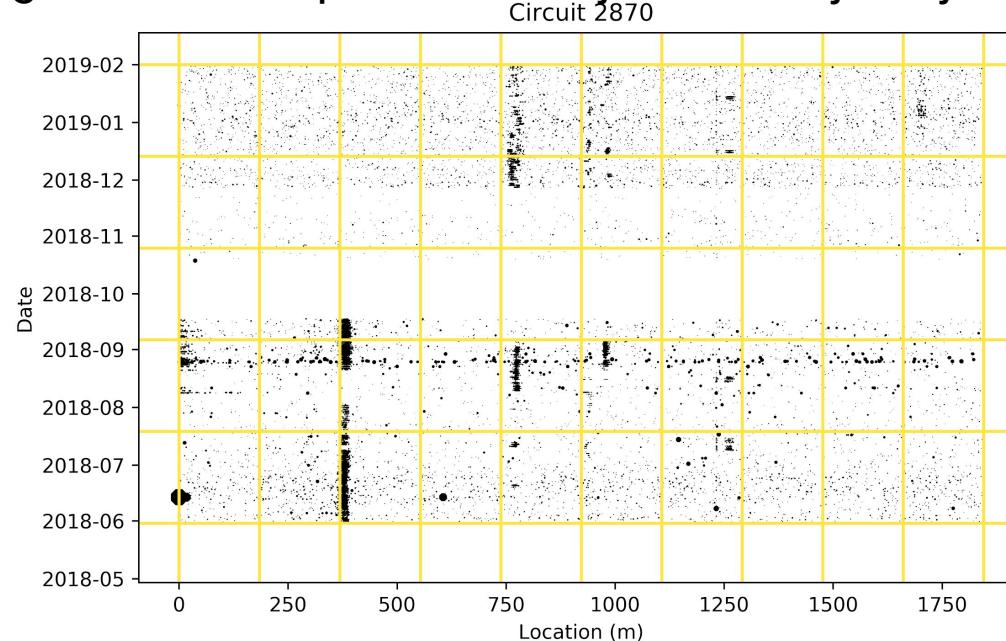
Voorbeeld

Circuit 2870



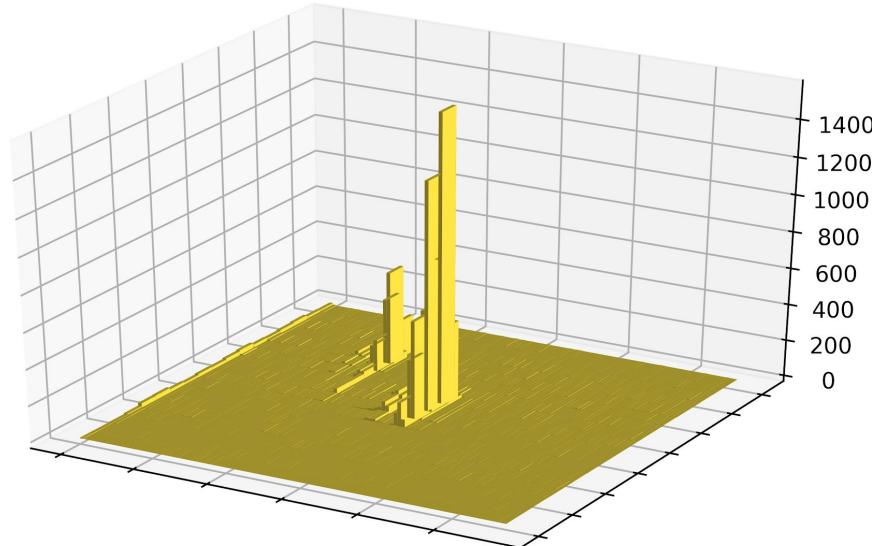
Hoe gaat Pinta te werk?

- Begin met data opdelen in hokjes. De hokjes zijn over het algemeen kleiner.



Hoe gaat Pinta te werk?

- Tel vervolgens alle partial discharges in een hokje.



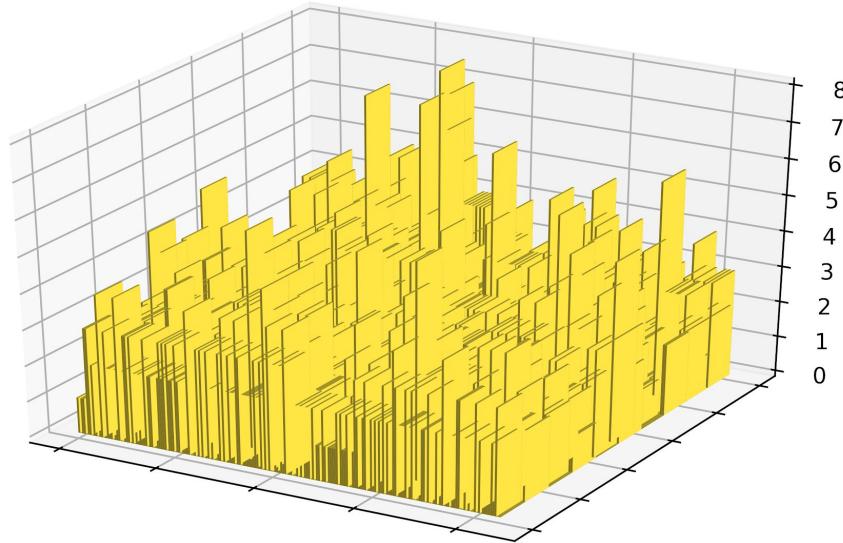
Hoe gaat Pinta te werk?

- Meeste hokjes: weinig PD's
- Groot verschil in aantal PD's in andere hokjes.
- Geen cluster? Dan ook minder verschil in aantal!



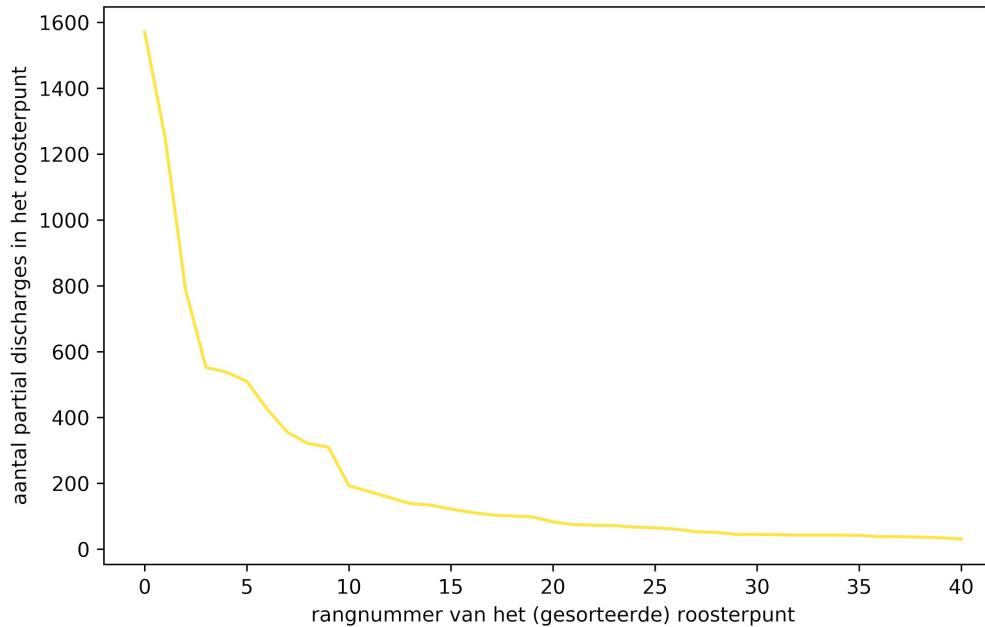
Hoe gaat Pinta te werk?

- 3D plot van een kabel zonder clusters:



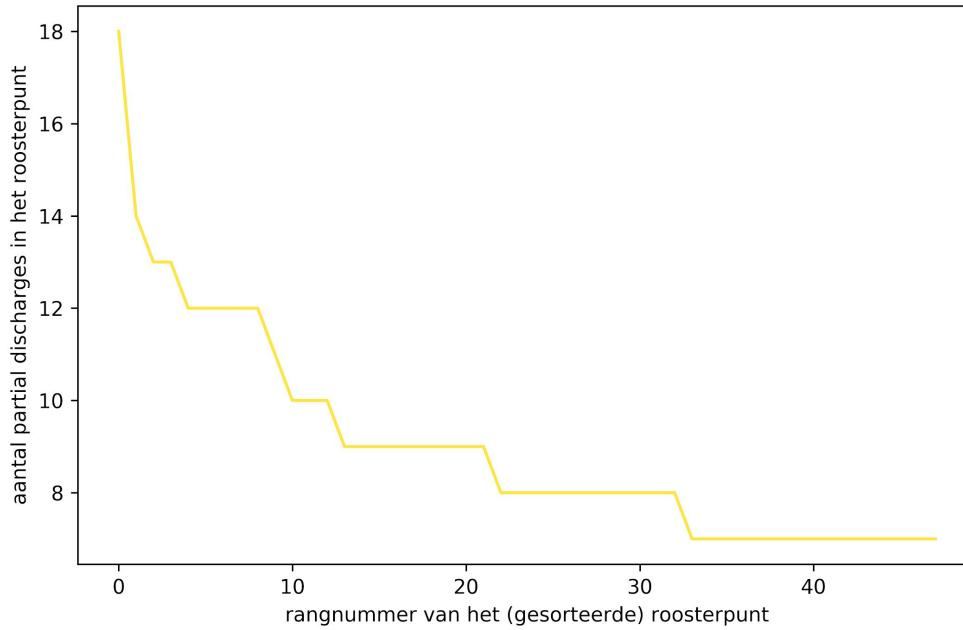
Hoe gaat Pinta te werk?

- Pinta sorteert de hokjes van veel PD's naar weinig PD's:



Hoe gaat Pinta te werk?

- Grafiek van een kabel waar niets mee aan de hand is:

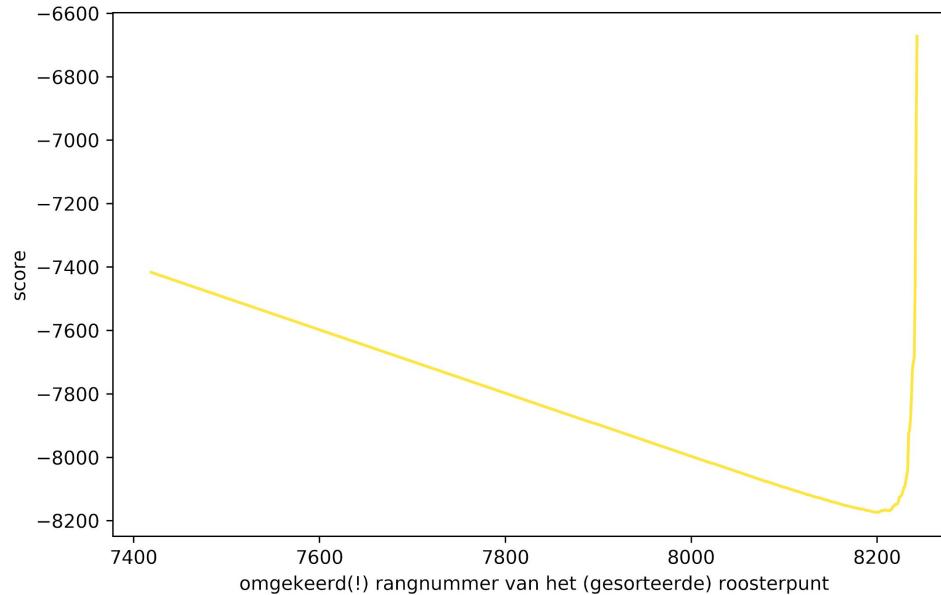


Hoe gaat Pinta te werk?

- Vraag: vanaf hoeveel PD's is een hokje "abnormaal"?
- Mogelijk antwoord: helling van de grafiek. Op plekken met een steile helling zit waarschijnlijk een cluster.
- Elk paar opeenvolgende hokjes krijgt een "score".
- Pinta telt alle scores bij elkaar op.

Hoe gaat Pinta te werk?

Hoog tijd voor een plotje.

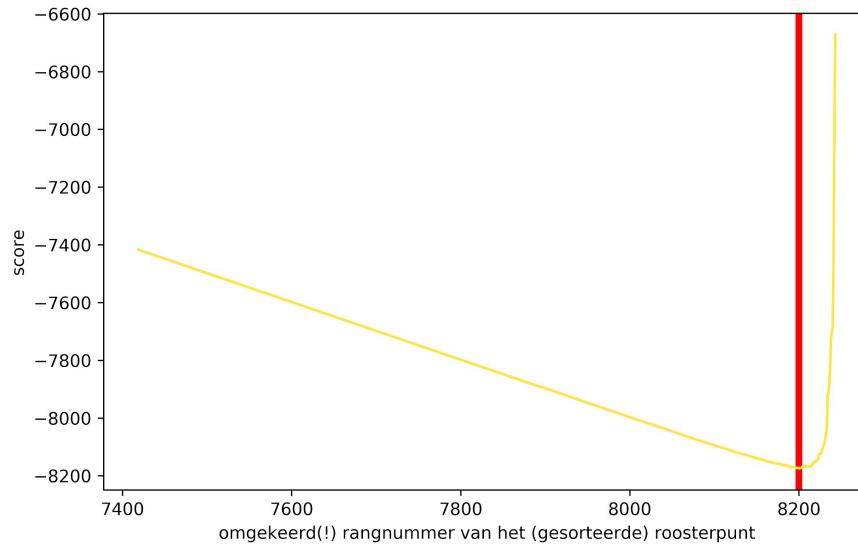


Hoe gaat Pinta te werk?

- Er is iets aan de hand op het minimum
- Kantelpunt! Van veel dezelfde PD's naar veel verschillende PD's

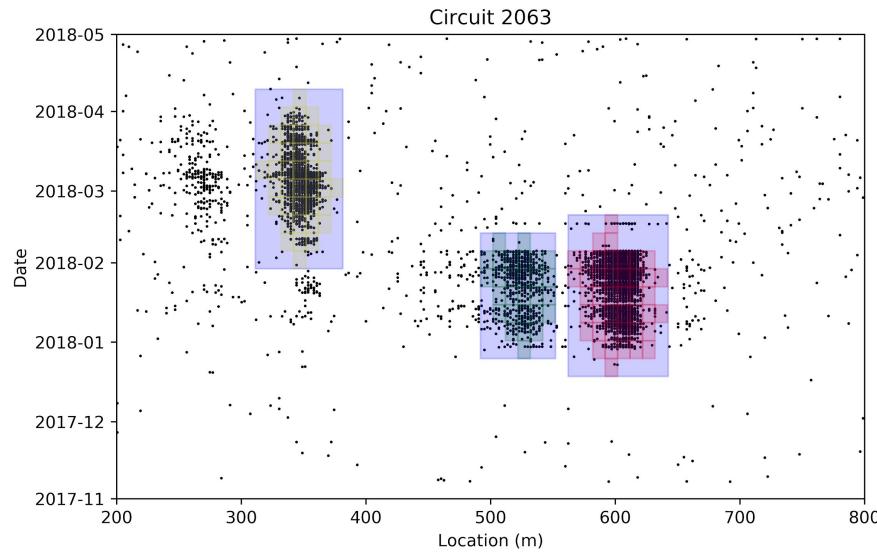
Hoe gaat Pinta te werk?

- We zien dat er iets mis is met de top-44 hokjes!



Hoe gaat Pinta te werk?

- De rest gaat ongeveer zoals de laatste stappen van het DB-scan algoritme:



Voordelen van Pinta

- Erg snel
- Het algoritme geeft geen resultaten als er niets aan de hand is
- Men kan de gevoelighedsparameter flink veranderen zonder dat het algoritme ongewenste resultaten geeft
- Werpt een andere blik op de data



Nadelen van Pinta

- Maakt bijzondere aannames over de PDs
- De gevoelighedsparameter heeft geen concrete betekenis
- De gevoelighedsparameter moet opgeschaald worden als de hokjes groter of kleiner gekozen worden
- Slechte naamgeving
 - De naam geeft niet aan hoe verdachte rechthoekjes gevonden worden

Runtime

			
Circuit 2063 (gemiddeld)	9 . 9 ms	155 ms	12 . 4 ms
Circuit 3010 (geen clusters)	4 . 8 ms	380 ms	29 ms
Circuit 2145 (veel PD's)	21 ms	2070 ms	85 ms



Ensemble-algoritme

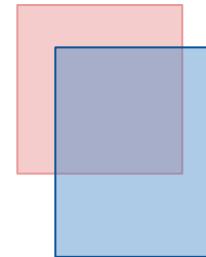
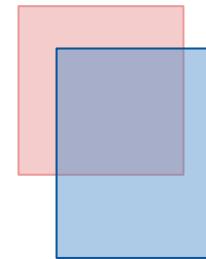
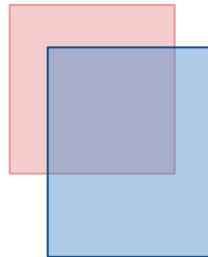
Samenvatting algoritmen

Wow dat was even interessant zeg!

Meneer op de voorste rij, weet jij nog welke algoritmes we hebben bedacht?

Laat handen zien: welke was jullie favoriet? (*Iedereen steekt bij alledrie beide handen op; wild gejuich*) Precies! Ze zijn allemaal fantastisch! Daarom gaan we ze combineren

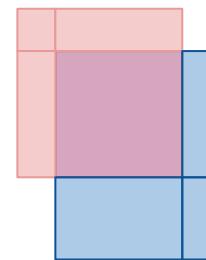
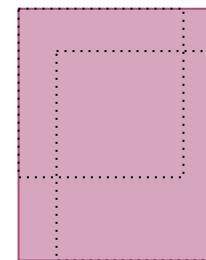
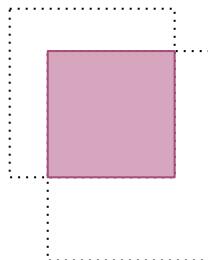
Combineren van Clusters



&

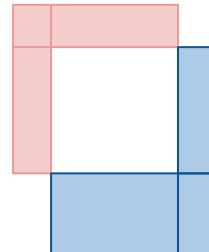
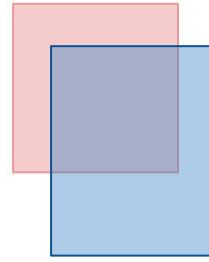
|

+



Implementatie Clusters

- Operator overloading om Clusters samen te voegen
 - $\&$, $|$, $+$
- Hulpfuncties:
 - $-$
 - $disjunct$
- Extra snufjes:
 - Ordening ($<$, $==$)
 - Hashable



Ensemble model

- Resultaten van andere algoritmen combineren
- Gelaagde opzet
 - Rectangle
 - Cluster
 - ClusterEnsemble
- Nodig voor behoud van structuur bij +
 - Bij & en | is de uitkomst 1 Rectangle per Cluster
 - Bij + kunnen dit er wel 7 zijn!



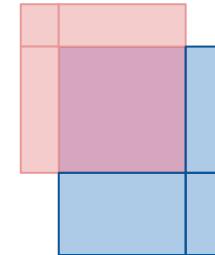
Rectangle

- Een rechthoek
- Twee dimensies:
 - Location_range
 - Time_range
- Found_by: Door wie is deze Rectangle gevonden?
- Gebruik &, |, + en - om ze te combineren

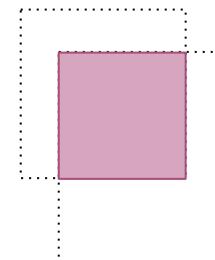


Cluster

- Een verzameling Rectangles
- Bevat gewoonlijk 1 Rectangle
 - Meerdere Rectangles als + wordt gebruikt
- Gebruik &, | en + om ze te combineren
 - Rectangle doet het zware werk
- Most confident
 - Op basis van meeste Rectangle.found_by
- Iterable
 - `for r in my_cluster.rectangles`
 - `for r in my_cluster`

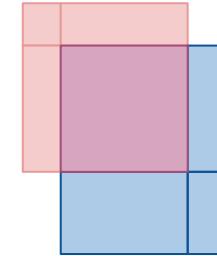
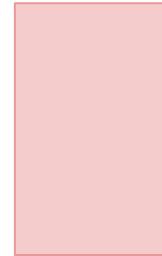


most_confident

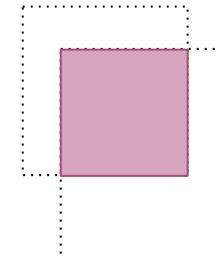


ClusterEnsemble

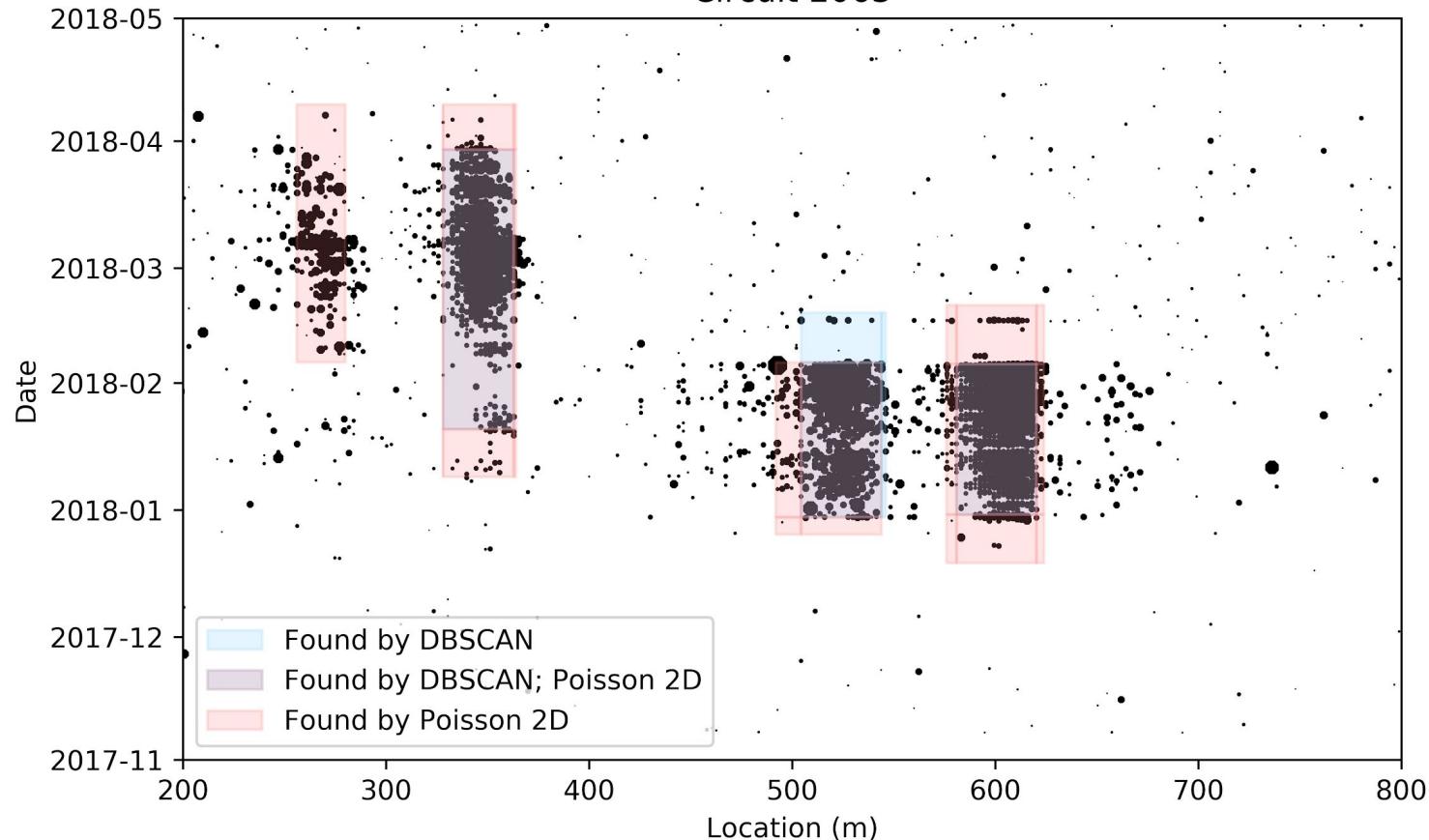
- Een verzameling Clusters
- Ieder cluster is disjunct
- Gebruik &, | en + om ze te combineren
 - Cluster doet het zware werk
- Most confident
 - Cluster doet het zware werk
 - Ieder cluster kan een andere grens hebben
 - Niet hetzelfde als &
- Iterable
 - `for r in my_ensemble.clusters`
 - `for r in my_ensemble`



most_confident



Circuit 2063



Runtime

- Combineren van clusters kan erg snel

&	$533 \mu\text{s} \pm 9.73 \mu\text{s}$
	$2.19 \text{ ms} \pm 58.8 \mu\text{s}$
+	$28 \text{ ms} \pm 856 \mu\text{s}$

Runtime

- Totale runtime hangt vooral af van de algoritmes

Poisson & Pinta	$16.2 \text{ ms} \pm 255 \text{ } \mu\text{s}$
Poisson & DBSCAN	$165 \text{ ms} \pm 1.08 \text{ ms}$
DBSCAN & Pinta	$180 \text{ ms} \pm 3.72 \text{ ms}$
Poisson & DBSCAN & Pinta	$189 \text{ ms} \pm 4.17 \text{ ms}$

Voor- en nadelen

- &
 - Voordeel: Snelst ($533 \mu\text{s}$)
 - Voordeel: Maakt te grote clusters kleiner
 - Nadeel: Verlies van informatie
- |
 - Voordeel: Snel (2.19 ms)
 - Voordeel: Slecht algoritme wordt verbeterd door de rest
 - Nadeel: Clusters worden soms te groot
- +
 - Voordeel: Geen verlies van informatie
 - Nadeel: Langzamer dan de rest (28 ms)
 - Nadeel: Erg veel rechthoekjes

Maatstaven

Behalve kijken hebben we nu ook nieuwe maatstaven:

- Hoeveel algoritmes hebben het gevonden?
- Verhouding (#PD binnen cluster / oppervlakte cluster) / (#PD totaal / oppervlakte totaal)
- aka
- Verhouding PD-dichtheid binnen cluster / PD-dichtheid totaal





Technische details

Python module

- Duidelijke verdeling in bestanden
 - algorithms.py
 - cluster.py
- Alle functies hebben uitgebreide documentatie
- plot.py
 - Handige plot functies voor circuits, clusters, en meer!
- Veel dingen kunnen in één regel code



Conclusie

Geslaagd!

- Clusters worden vaak gevonden door ieder algoritme
- Clusters worden altijd gevonden door één algoritme
- Runtime is goed



Verslag

- Verantwoording van keuzes
- Gedetailleerde informatie algoritmen

Toekomst module

We hebben al:

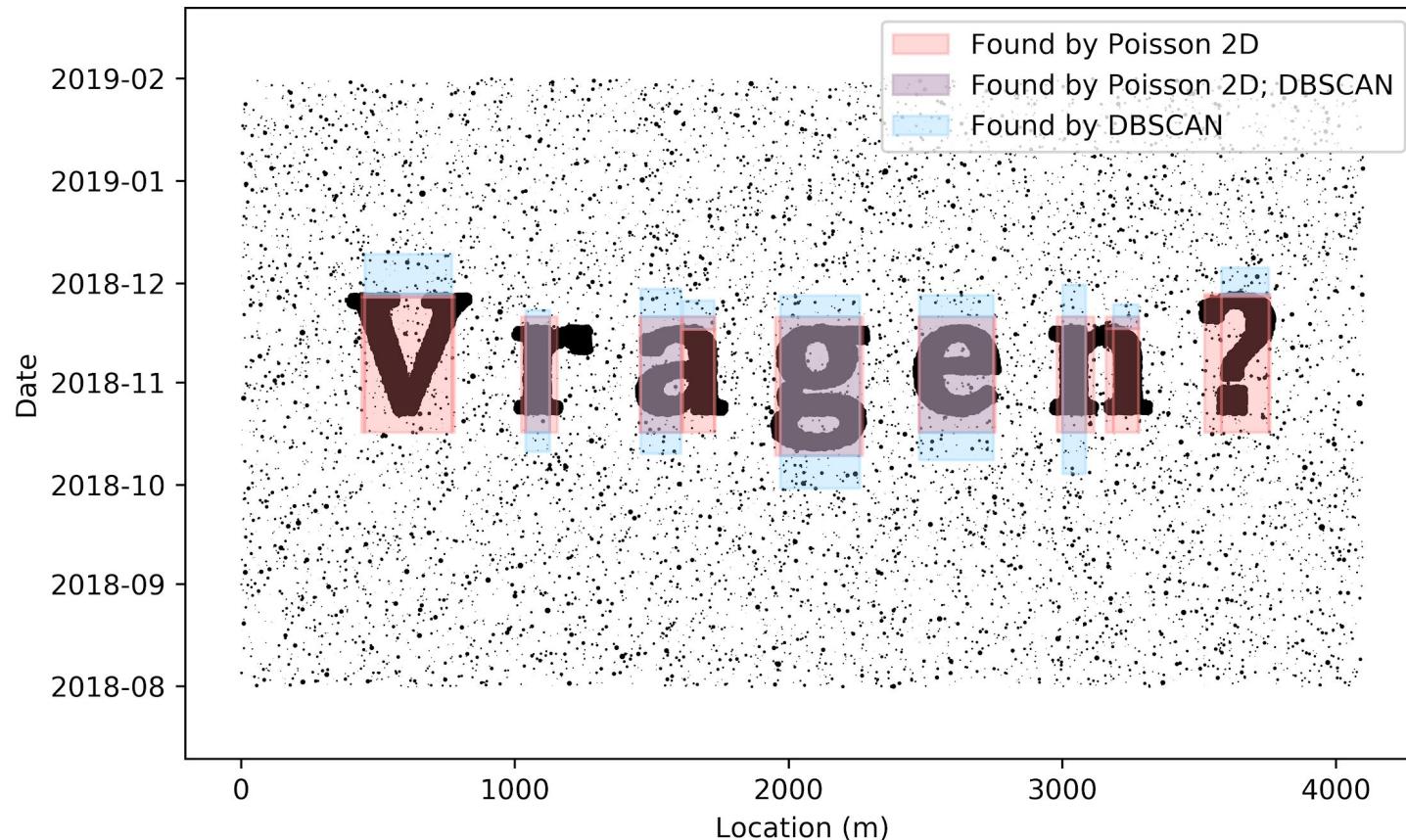
- De module is al bruikbaar
- Leesbaar geschreven

Toekomst module

In de toekomst:

- Nieuwe algoritmen zijn gemakkelijk toe te voegen
- Moet nog real-time worden toegepast
- De algoritmen kunnen nog worden aangepast zodat ze ook de grootte van de lading van de PD's meenemen
- Er kan nog een drempelwaarde worden toegevoegd voor de ladingsgrootte

Circuit 99999999





EXTRA TIJD

Inhoudsopgave (Choose your own adventure)

- [Monte Carlo clustering](#)
- [Bandenafstand](#)
- [Clusterbreedte](#)
- [Variërend nominaal PD gedrag](#)
- [Reflecties](#)
- [DNV GL warnings](#)
- [Drempel voor de lading](#)





Monte Carlo

Randomized clusteren

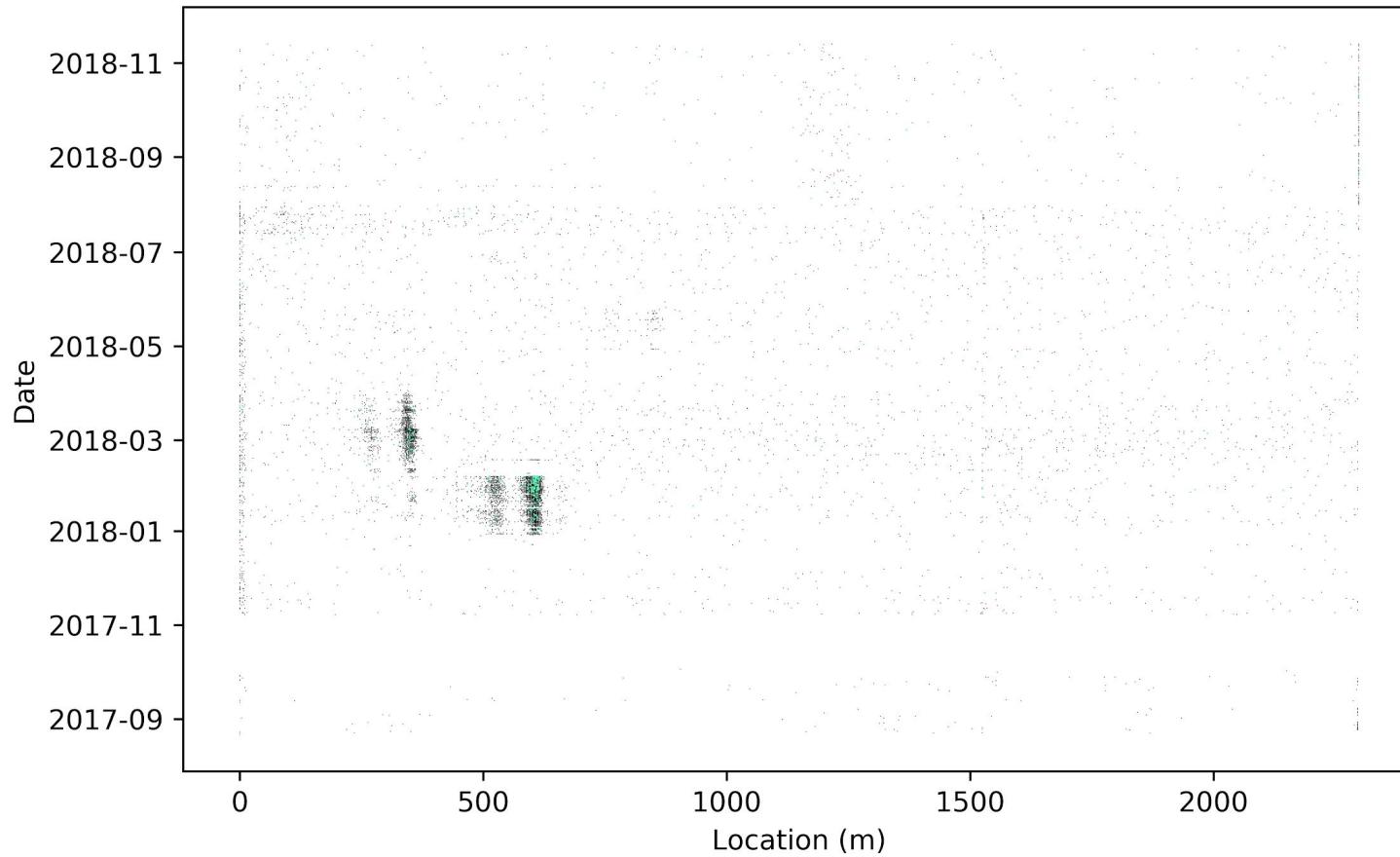
- In een cluster zitten veel PDs
- **Idee:** Een willekeurige PD zit vaak in een cluster
- Kies een heleboel willekeurige PDs
- Kijk of ze dicht bij elkaar liggen



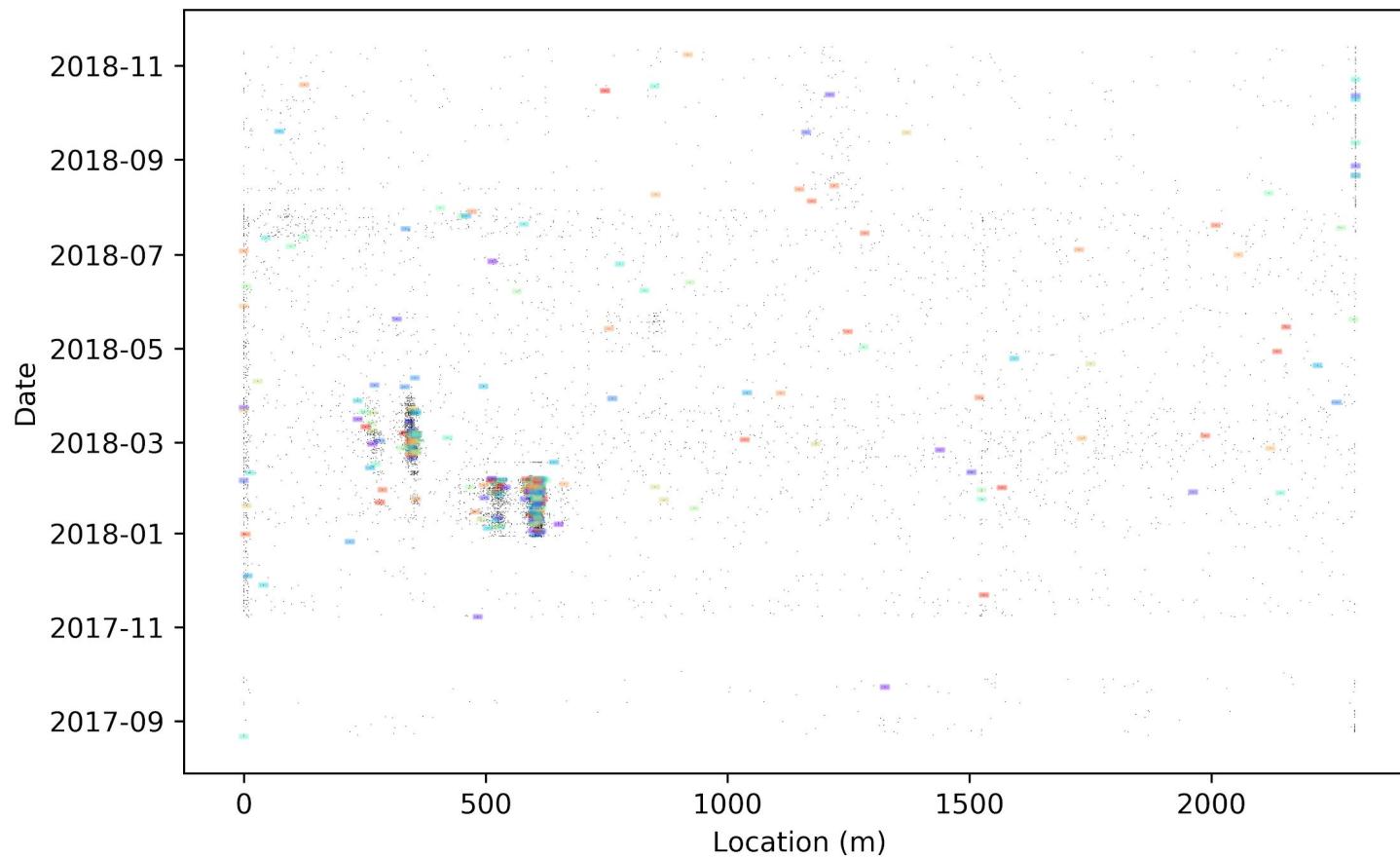
Implementatie Monte Carlo

- Kies een heleboel willekeurige PDs
- Teken om iedere PD een rechthoek
- I (or) alle rechthoeken die overlappen
- Filter rechthoeken
 - We willen alleen rechthoeken die door veel willekeurige PDs gegenereerd zijn
 - $\text{len}(\text{cluster}.\text{found_by}) > \text{num}$
- Klaar!

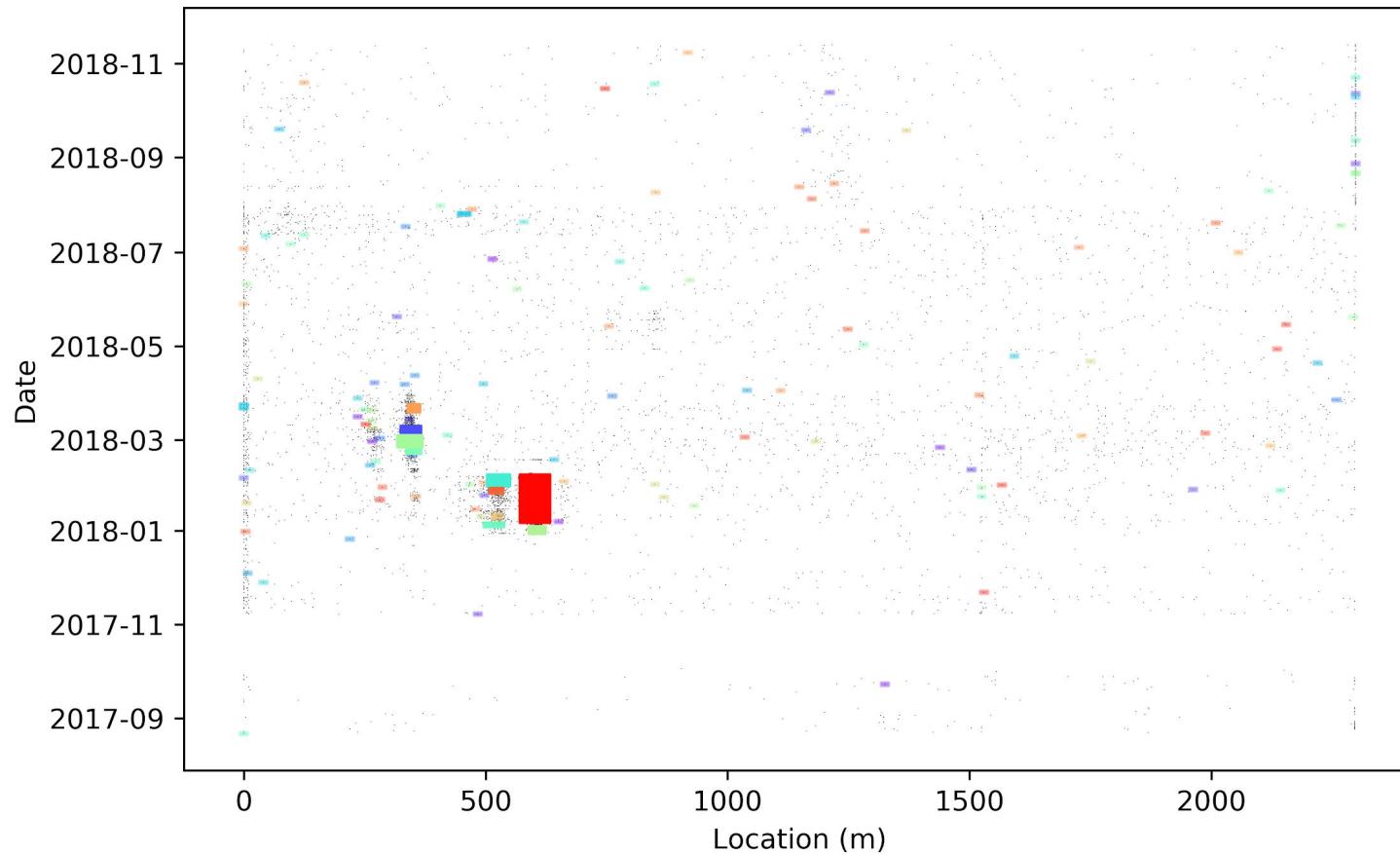
Circuit 2063



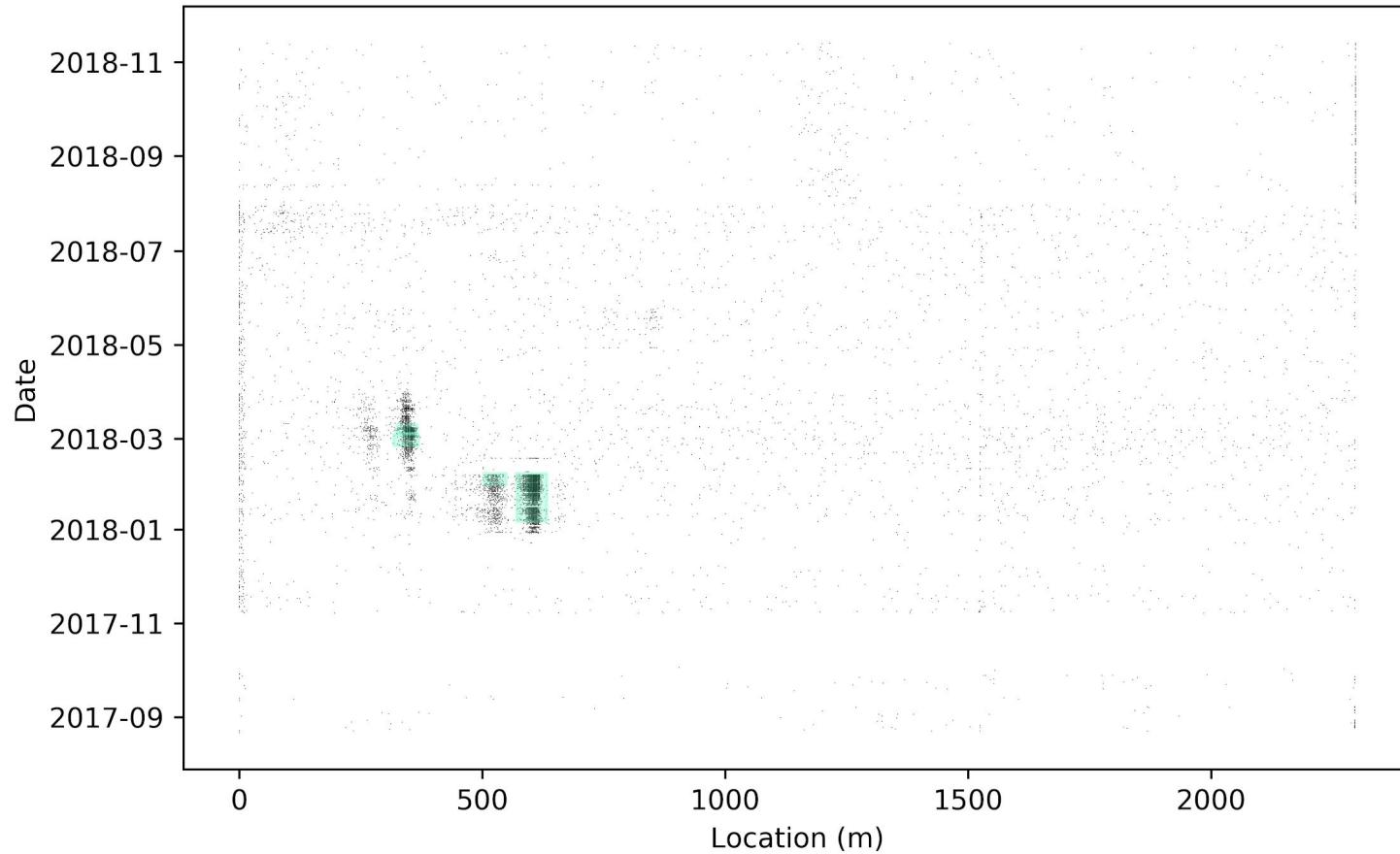
Circuit 2063



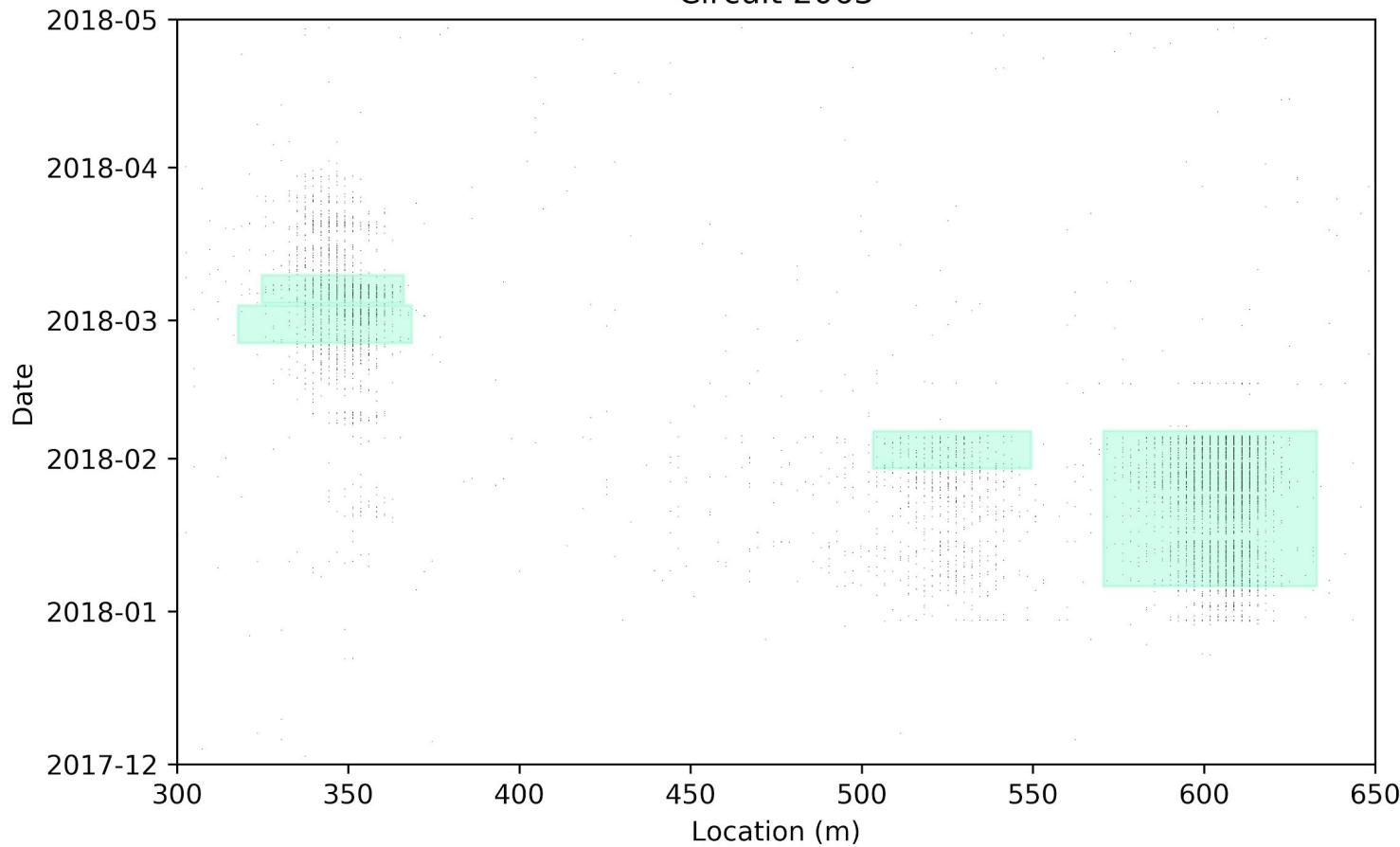
Circuit 2063



Circuit 2063



Circuit 2063



Analyse Monte Carlo

- Runtime
 - Hangt erg af van het aantal punten dat willekeurig gekozen wordt
- 100 punten: $70 \text{ ms} \pm 1.93 \text{ ms}$
- 300 punten: $520 \text{ ms} \pm 29.6 \text{ ms}$
- 500 punten: $1.23 \text{ s} \pm 95.4 \text{ ms}$
- 1000 punten: $5.12 \text{ s} \pm 424 \text{ ms}$
- Reden: Rechthoekjes | kost veel tijd als je veel rechthoekjes hebt
 - Dit is geen probleem bij normale Ensembles, want dan hebben we maar ~ 20 rechthoekjes

Analyse Monte Carlo

- Voordelen:
 - Output is willekeurig (Slecht resultaat? Probeer opnieuw!)
 - Weinig rechthoekjes: Snel
 - Veel rechthoekjes: Goed resultaat
 - Vindt niets bij ruis
- Nadelen:
 - Output is willekeurig (soms slecht resultaat door pech)
 - Veel rechthoekjes: Langzaam
 - Weinig rechthoekjes: Slecht resultaat
 - Een groot cluster kan een kleiner cluster verbergen
 - Splitst clusters soms op in twee delen omdat rechthoekjes slecht vallen





*Monte Carlo
Finished*

Bandenafstand

- PDs lijken niet continu over de locatie verdeeld
- Er zijn zogenaamde “banden”; verticale lijtjes PDs
- De banden zijn minder dan 30 centimeter breed
- De afstand tussen de verschillende banden is variabel
 - Zelfs binnen een circuit!

Clusterbreedte

- Het gemiddelde cluster is 30 meter breed
- Standaarddeviatie: 6.8 meter
- Standaarddeviatie is erg groot en verschilt per circuit
- Truncated normal distribution: Standaarddeviatie verandert
 - Dit is wat je verwacht!
- Het lijkt erop dat de clusterbreedte normaal verdeeld is
- Niet onderzocht: Hangt de clusterbreedte af van de circuitlengte?

Variërend nominaal PD gedrag

- Soms verandert het gedrag van een circuit plotseling
- Dit kan meerdere redenen hebben:
 - SCG is verplaatst naar een ander circuit maar dit is niet doorgegeven
 - De parameters zijn aangepast: gevoeliger of minder gevoelig gezet
 - Uitgezet: de kabel wordt gerepareerd

Reflecties

- Soms worden clusters op magische wijze gereflecteerd
- Dan vind je ze twee keer
- Een van de twee is nep!
- De neppe is te herkennen aan een kleiner aantal PDs
- Hypothese: 30%



DNV GL warnings

- Er zit een “algoritme” in de module die de DNV GL warnings als Clusters geeft
- Hiermee kan je ze makkelijk vergelijken met de echte algoritmes
- Doen speciale dingen:
 - Reflecties negeren
- Doen vreemde dingen:
 - Warnings gaan door terwijl het probleem al opgelost lijkt

Drempel voor de lading

- Meeste algoritmes kijken alleen naar aantallen pds
- Niet naar de lading
- Drempel: Alle pds met kleinere lading weggooien
- Ruis heeft vaak een kleinere lading