Terabytes aan wegdata

Wat doe je ermee?

Outline

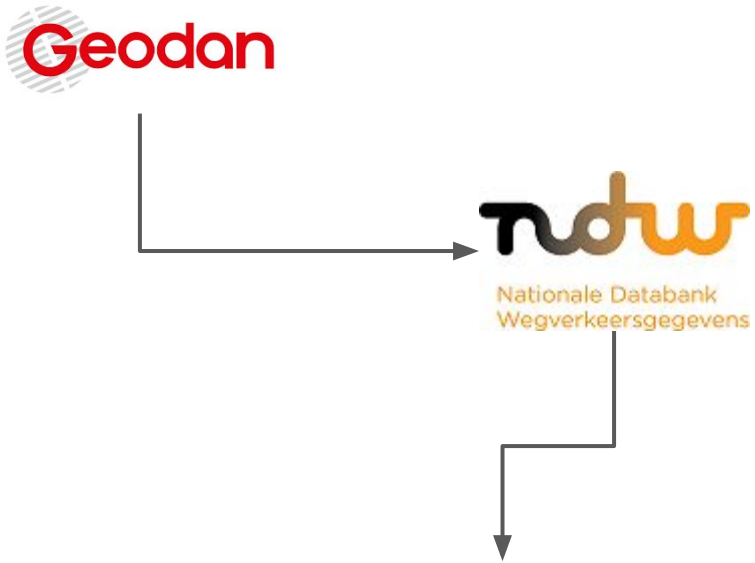
Veel wegdata (en meer)

Hoe sla je dat op?

Wat kan je ermee in R?

Context

Tom van Tilburg



19 overheden
inwinnen, opslaan en distribueren
van wegverkeersgegevens.
tbv. verkeersmanagement,
gerichte verkeersinformatie en
verkeerskundige analyses.

https://www.ndw.nu/

NDSS
proeftuin voor nieuwe technieken

Wat is wegdata?

Gebeurtenissen
(ongelukken, brugopeningen, evenementen etc.)

Meetpunten
(snelheid, intensiteit, voertuigtype, reistijd)

Floating Car Data
(snelheid)

Meetpunten

SensorData

14.00 meetlocaties

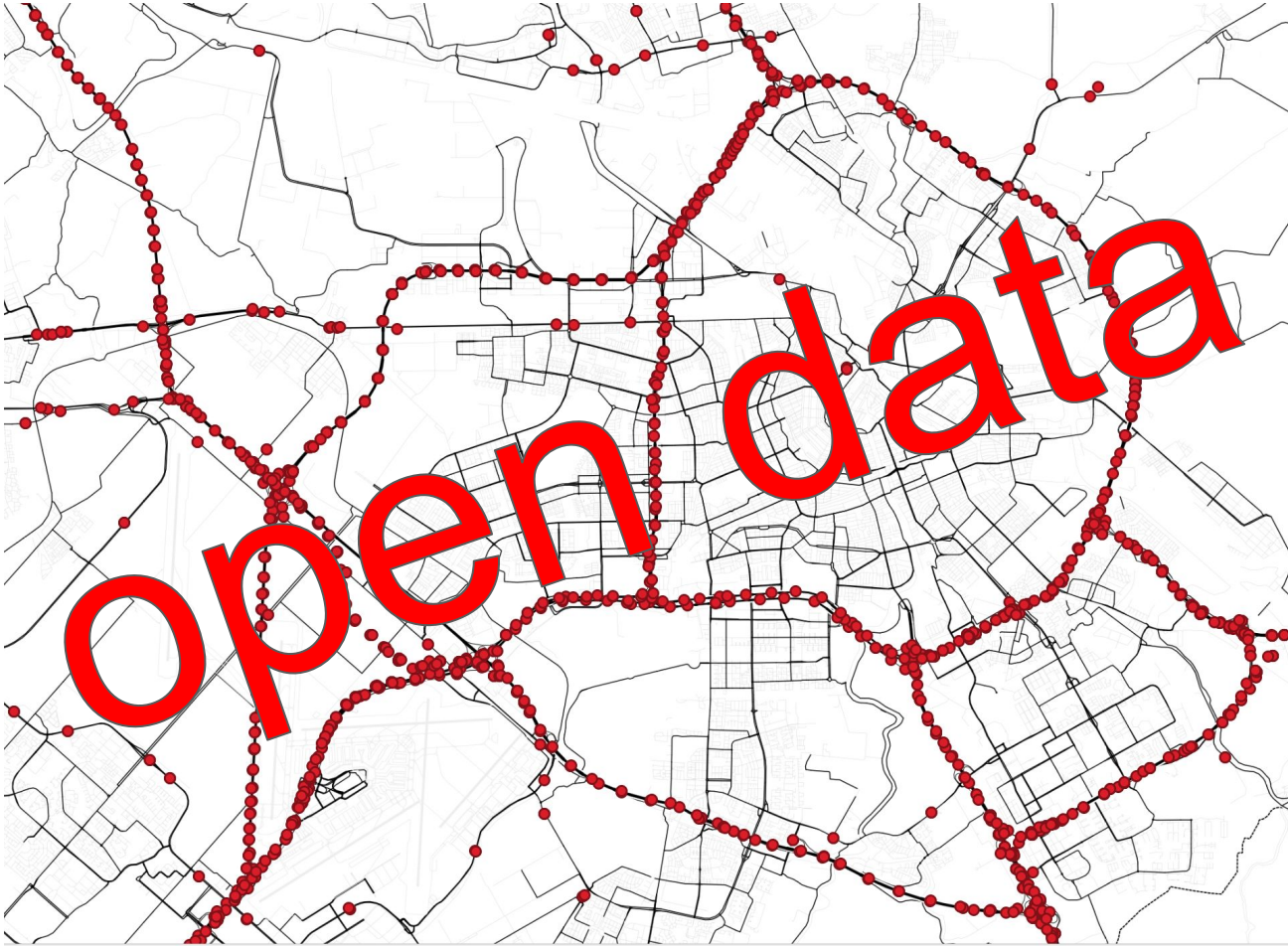
N rijstroken per punt

snelheid, intensiteit

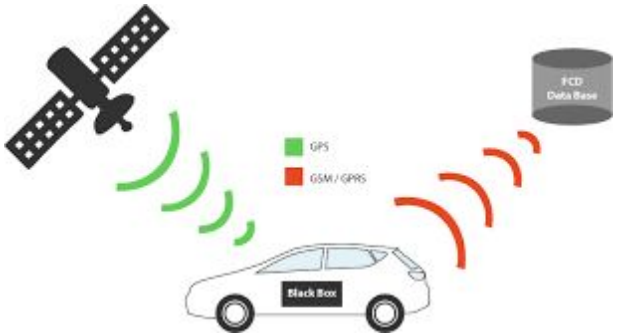
per minuut

alleen hoofdwegen

Nationale Databank
Wegverkeersgegevens



Floating Car Data



Elke stukje weg in Nederland een actuele snelheid

400.000 km

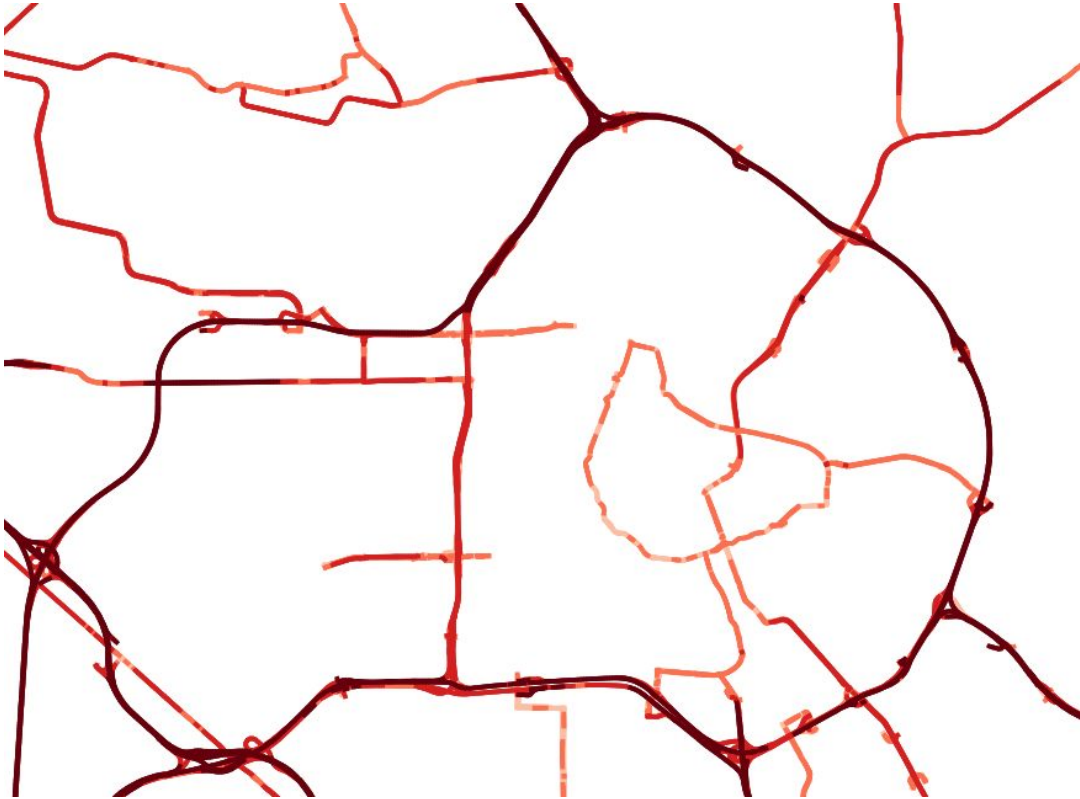
10 miljoen wegsegmenten

van ~50 meter

elk data per minuut

Nationale Databank Wegverkeersgegevens

niet open...



Wat zit er in Floating Car Data?

Elke minuut 750.000 records

Elke dag 1 miljard records

Elke maand 250 gb

Elk jaar 3 Terabyte

Elke maand een nieuwe shapefile

10 miljoen segmenten per shape

Routeerbaar netwerk

Hoe sla je dat op?

1. Relational database (Postgres)

Kennen we al

Postgis !

PGRouting

Diverse handige indexen

Niet handig als je index groter is dan je geheugen

2. Column storage (Clickhouse)

Nieuw van Yandex

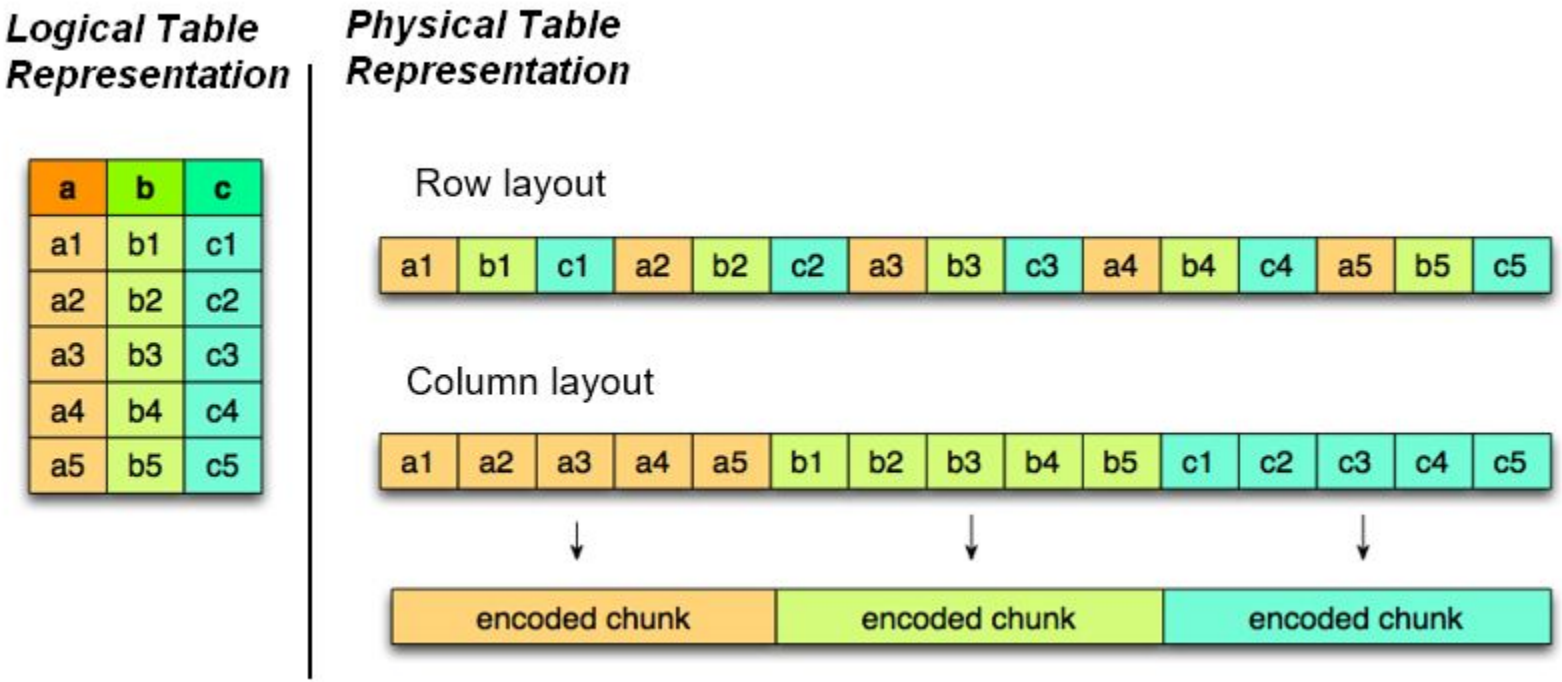
Columnar storage

Niet geschikt voor transacties en updates

Wel geschikt voor analyses op grote homogene datasets

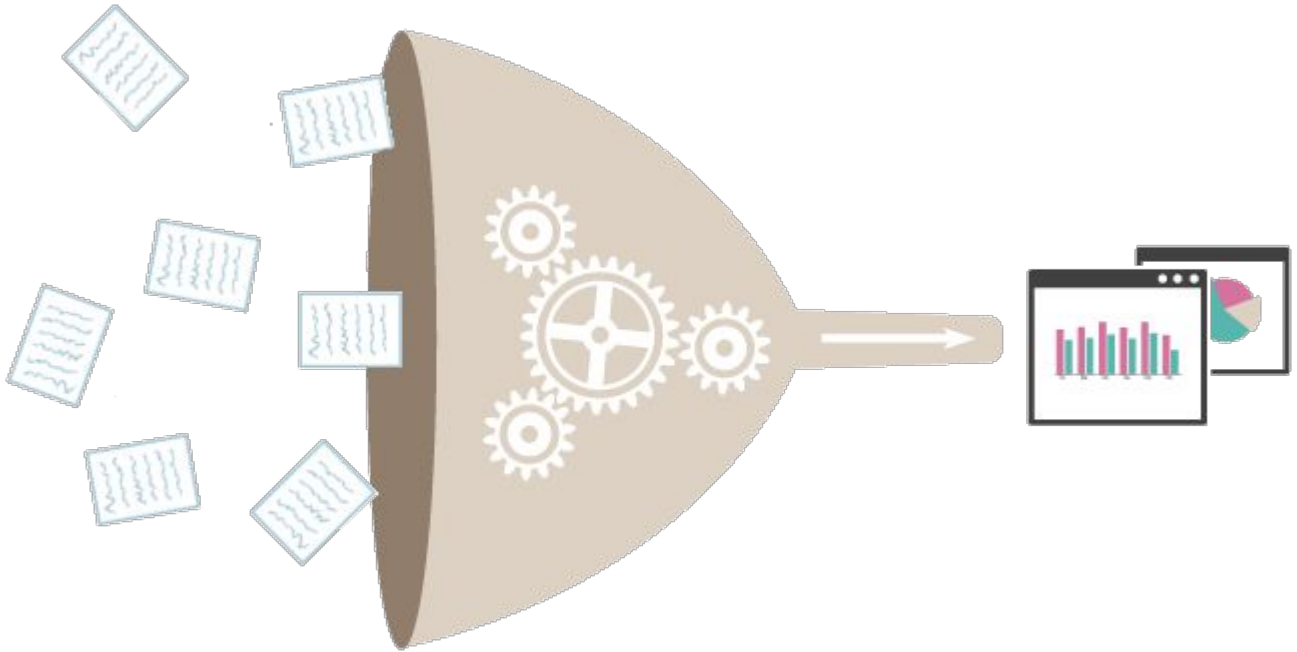
Comprimeert !

Column storage??



https://gerardnico.com/_detail/data/type/relation/structure/columnar_physical_table_representation.png

Ingesting data



```
ogr2ogr -f 'PostgreSQL' PG:"dbname=ndw" \
   -nln wegvakken \
   -a_srs EPSG:28992 \
   /vsizip/vsicurl/https://ndw.nu/Wegvakken1234.zip/Wegvakken/Wegvakken.shp


curl -s -L 'https://ndw.nu/fcddata/1234/20181205?op=OPEN' | \
   gunzip - | \
   awk -F';' '$6>2 {print}' | \
   clickhouse-client \
    -d ndw \
    -q "INSERT INTO fcd_1234 FORMAT CSV" \
    --format_csv_delimiter=";"
```

Wat zit er in de database?

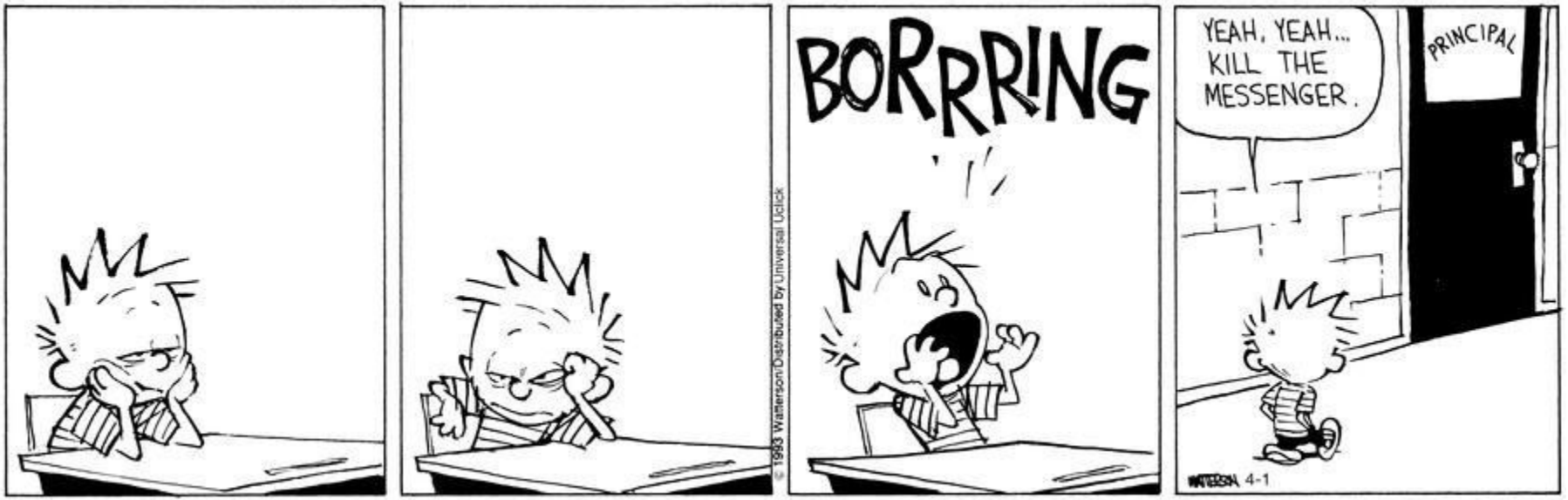
table	size	rows	days	avgDaySize
fcd_14159	246.44 GiB	32.350.290.953	0	inf YiB
fcd_14148	85.96 GiB	18.942.687.219	0	inf YiB
regendata	921.86 MiB	414.790.063	0	inf YiB
gridsegments_14148	46.66 MiB	10.112.685	0	inf YiB
gridsegments	10.24 MiB	1.992.374	0	inf YiB

5 rows in set. Elapsed: 0.047 sec. Processed 4.28 thousand rows, 1.58 MB (91.15 thousand rows/s., 33.67 MB/s.)

```
SELECT
    avg(speedkph) AS avgspeed,
    toStartOfHour(toDateTime(timest)) AS time,
    bar(avgspeed, 0, 80, 20) AS bar
FROM ndw.fcd_14148
WHERE toDate(timest) = '2018-12-05'
GROUP BY time
ORDER BY time ASC
```

avgspeed	time	bar
87.10533005456999	2018-12-05 00:00:00	
89.81221667226542	2018-12-05 01:00:00	
89.09388986318768	2018-12-05 02:00:00	
88.96518946165361	2018-12-05 03:00:00	
79.4539578549339	2018-12-05 04:00:00	
66.82501420057436	2018-12-05 05:00:00	
58.65694159554462	2018-12-05 06:00:00	
55.26684159186408	2018-12-05 07:00:00	
57.24654746469568	2018-12-05 08:00:00	
57.818275225686854	2018-12-05 09:00:00	
57.68039539406487	2018-12-05 10:00:00	
57.048633996351576	2018-12-05 11:00:00	
56.597612816817005	2018-12-05 12:00:00	
55.85356930248396	2018-12-05 13:00:00	
54.44810871029282	2018-12-05 14:00:00	
51.893994720772234	2018-12-05 15:00:00	
51.91768708012431	2018-12-05 16:00:00	
56.80484213246235	2018-12-05 17:00:00	
62.51154537937126	2018-12-05 18:00:00	
66.51300019066049	2018-12-05 19:00:00	
68.99154099514067	2018-12-05 20:00:00	
70.58476744657058	2018-12-05 21:00:00	
74.66897724445963	2018-12-05 22:00:00	
82.06142198407879	2018-12-05 23:00:00	

```
24 rows in set. Elapsed: 7.573 sec. Processed 1.37 billion rows, 7.31 GB (180.46 mill
ion rows/s., 965.01 MB/s.)
```


BORRRING
YEAH, YEAH... KILL THE MESSENGER.
PRINCIPAL
WATTERSON 4-1
© 1993 Watterson/Distributed by Universal Uclick

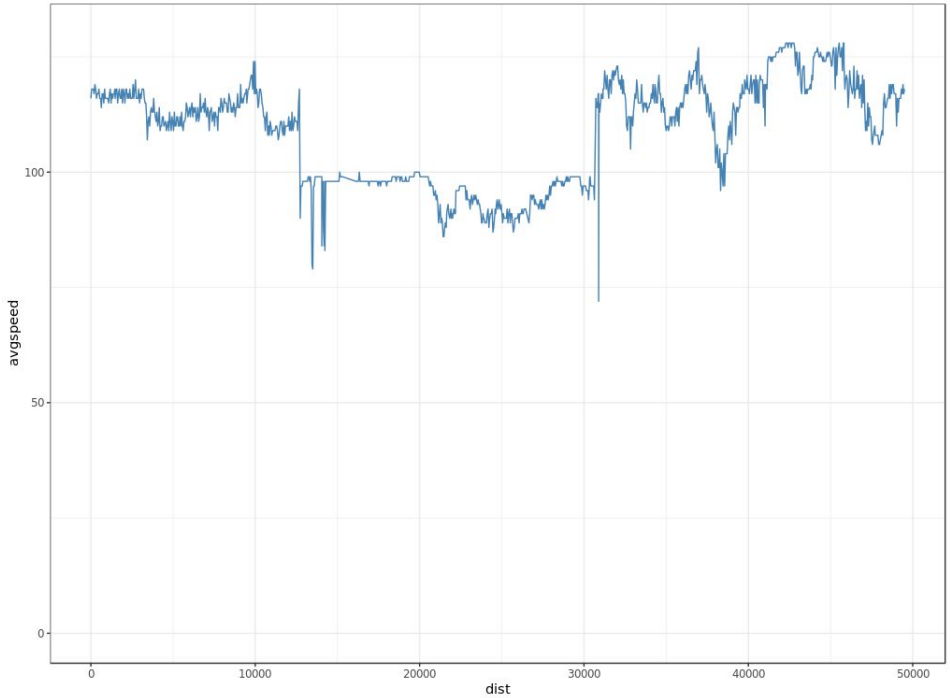
Routing

```
SELECT r.* , s.id, s.geom
FROM pgr_bddijkstra(
    'SELECT id, source, target, cost, reverse_cost FROM
basemaps.segments WHERE source Is Not Null',
        463497, -- Houten
        301109, --Purmerend
        true
    ) as r
```

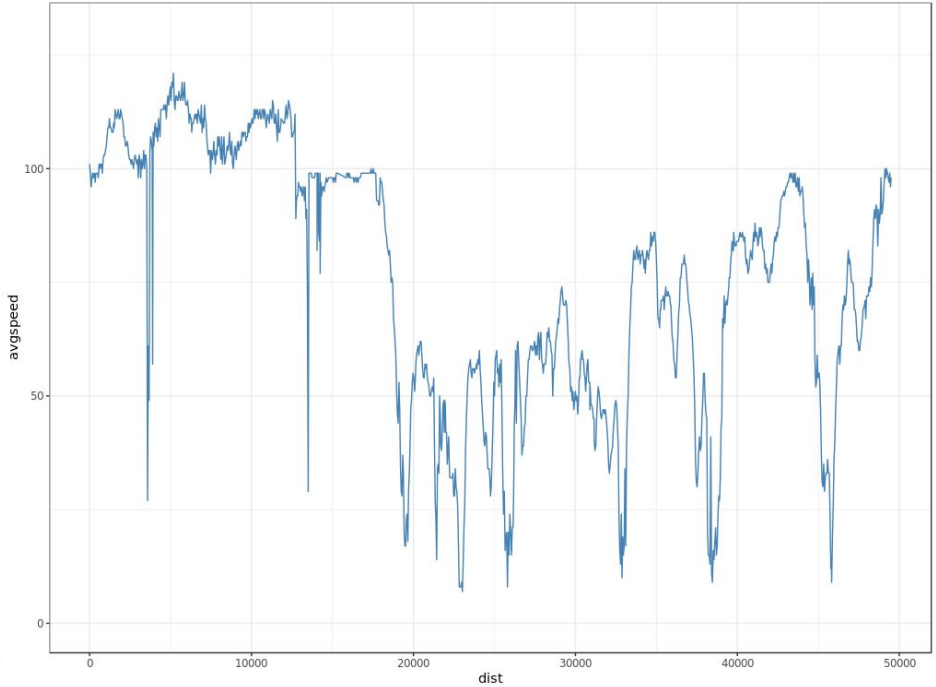
Hoe zie je een file in je data?

Snelheid op A12

13:00

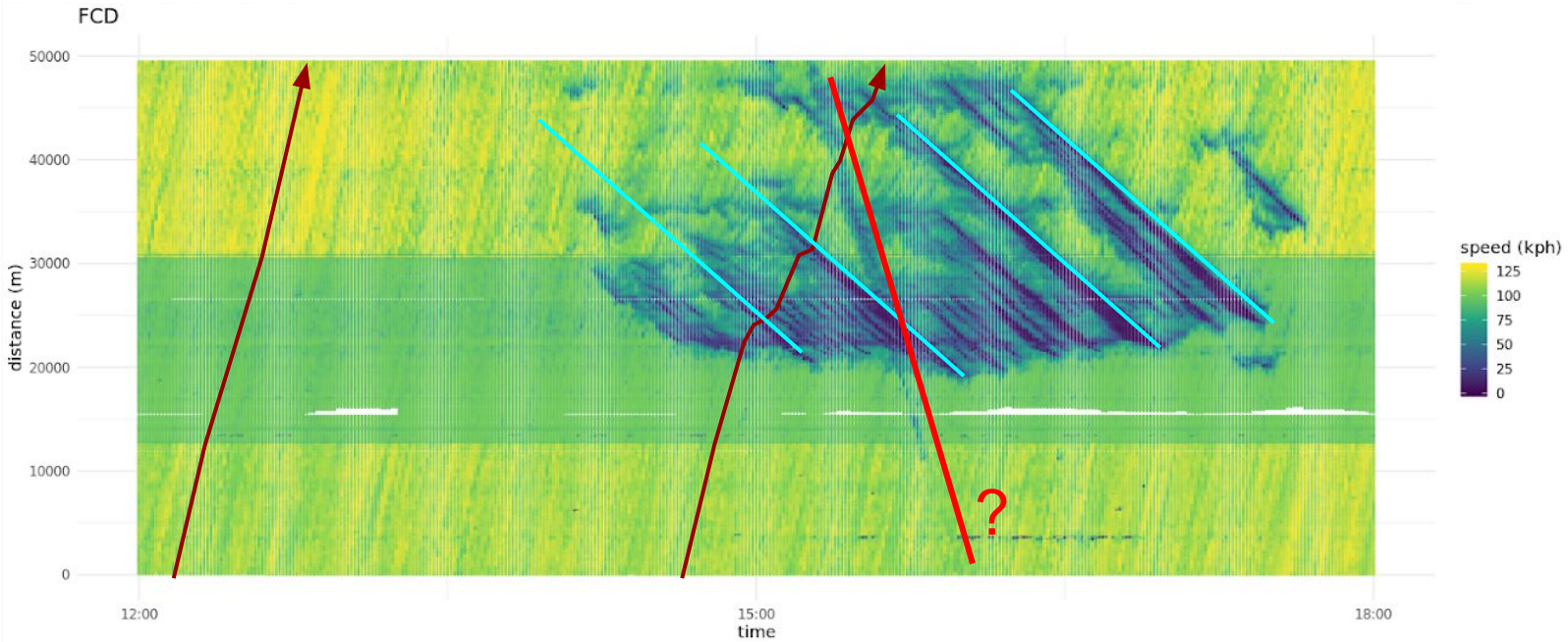


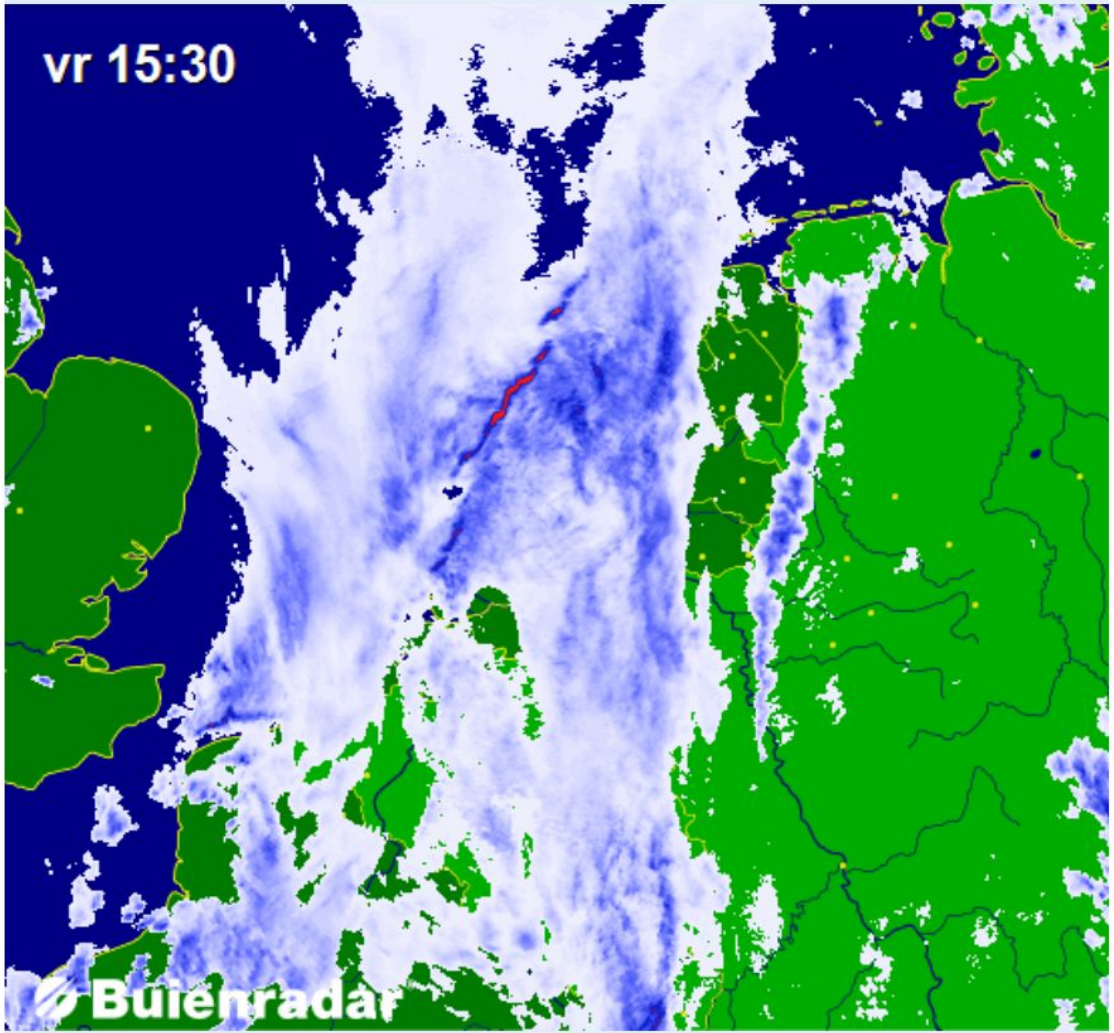
16:00




The Mathematical Society of Traffic Flow

Tijdweg diagram




vr 15:30
Buienradar

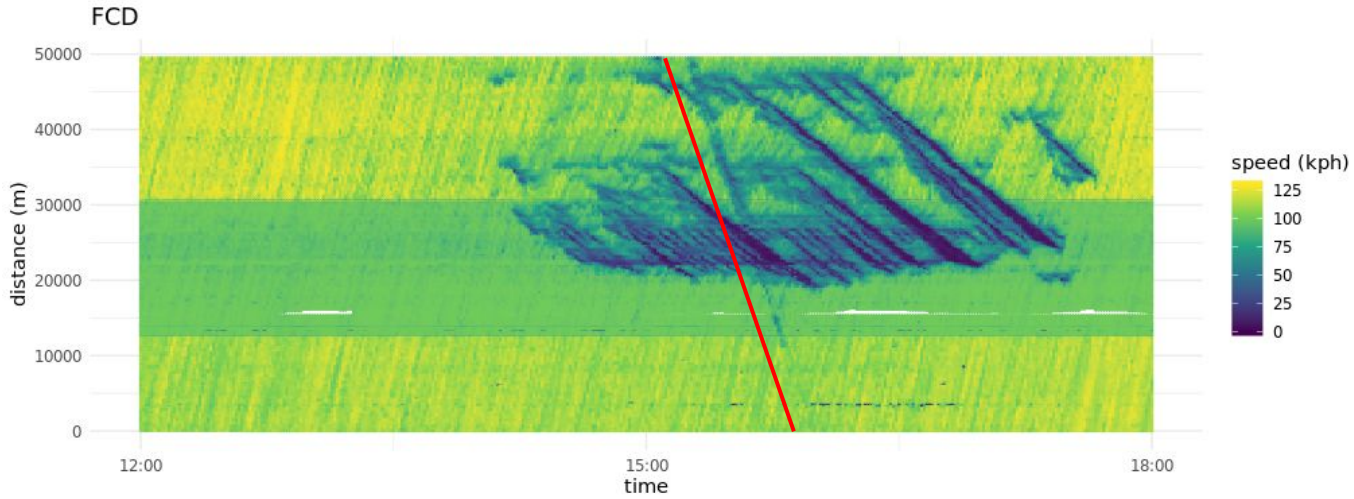
```
#/bin/bash

wget -qO-
https://data.knmi.nl/download/radar_tar_refl_composites/1.0/0001/2018/12/05/RAD25_OPER_R___TARPCP__L2__2018120
5T000000_20181206T000000_0001.tar  | \
    tar xv

for file in *.h5; do
    DATE=${file:17:8}
    TIME=${file:25:4}
    TIMEST=`date -d "$DATE $TIME" +"%s"`
    gdal_translate $file -sds -f XYZ /vsistdout/ -a_srs '+proj=stere +lat_0=90 +lon_0=0 +lat_ts=60 +a=6378.14
+b=6356.75 +x_0=0 y_0=0' | \
        awk '($3>0 && $3<32768) {printf("%d,%d,%d,%d\n"), $1, $2, $3, '$TIMEST'}' | \
        clickhouse-client -d ndw "INSERT INTO regendata FORMAT CSV"
done
rm *.h5
```

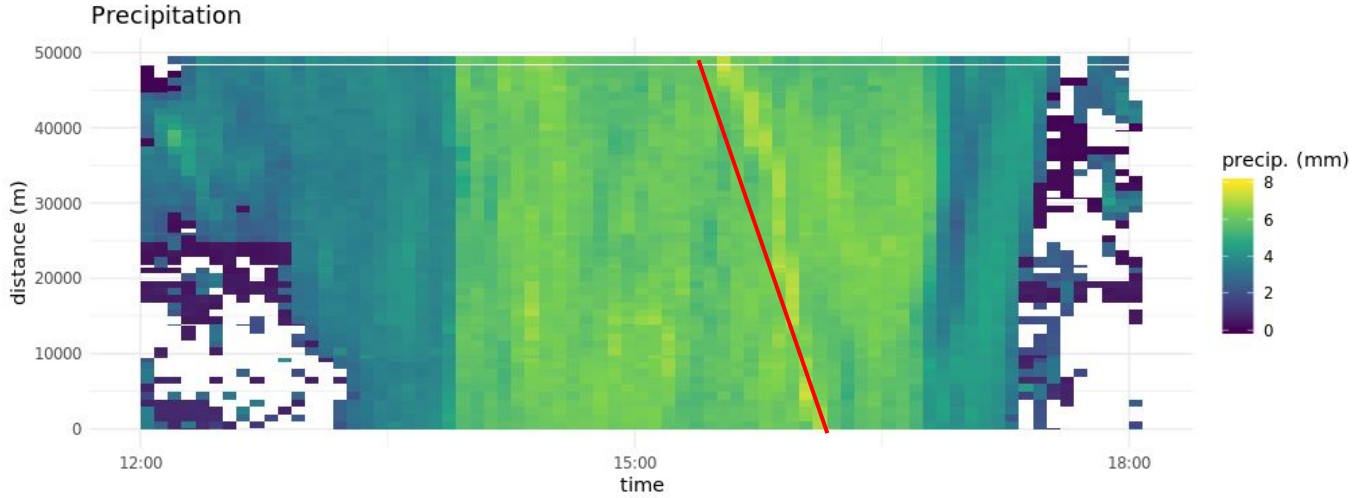
2 maanden regendata, elke 5 min elke km^2

```
SELECT
    sum(p) / 100000 AS p,
    toDate(timest) AS date,
    bar(p, 0, 30000, 20) AS bar
FROM ndw.regendata
GROUP BY date
```

Koppel aan wegsegmenten

```
regen_sql <- paste0("
  SELECT timest + 3600 as timest, segmentid, coalesce(p/20,0) as p
  FROM (
      SELECT * FROM ndw.gridsegments_14148 WHERE segmentid IN (", paste0(route1$bm0_id, collapse = ","), ")
  ) as a
  INNER JOIN (
        SELECT *
        FROM ndw.regendata
        WHERE timest BETWEEN ",starttime - 3600," AND ",endtime - 3600,"
  ) as b
  ON (a.y = b.y AND a.x = b.x)
")
```


FCD
50000
40000
30000
20000
10000
0
distance (m)
12:00
15:00
18:00
time
speed (kph)
125
100
75
50
25
0


Precipitation
50000
40000
30000
20000
10000
0
distance (m)
12:00
15:00
18:00
time
precip. (mm)
8
6
4
2
0

Live demonstratie

Einde

Zelf spelen:

Verkeersdata: http://opendata.ndw.nu/

KNMI data: https://data.knmi.nl/datasets?q=radar

Clickhouse: https://clickhouse.yandex/

Blog: https://carto.com/blog/geospatial-processing-with-clickhouse/