ÜBE
R

GEO-DATEN IN PYTHON

AGENDA

1. Grundlegende Pakete: GeoPandas, Shapely und CRS
2. Beispielanalyse: Kundennachfrage nach PLZ und wo der optimale, nächste Standort zu suchen ist
3. Zusammenfassung

PYTHON STELLT EIN BREITES TOOLSET FÜR DIE ANALYSE VON GEODATEN

Packet	Nutzen	Beispiel

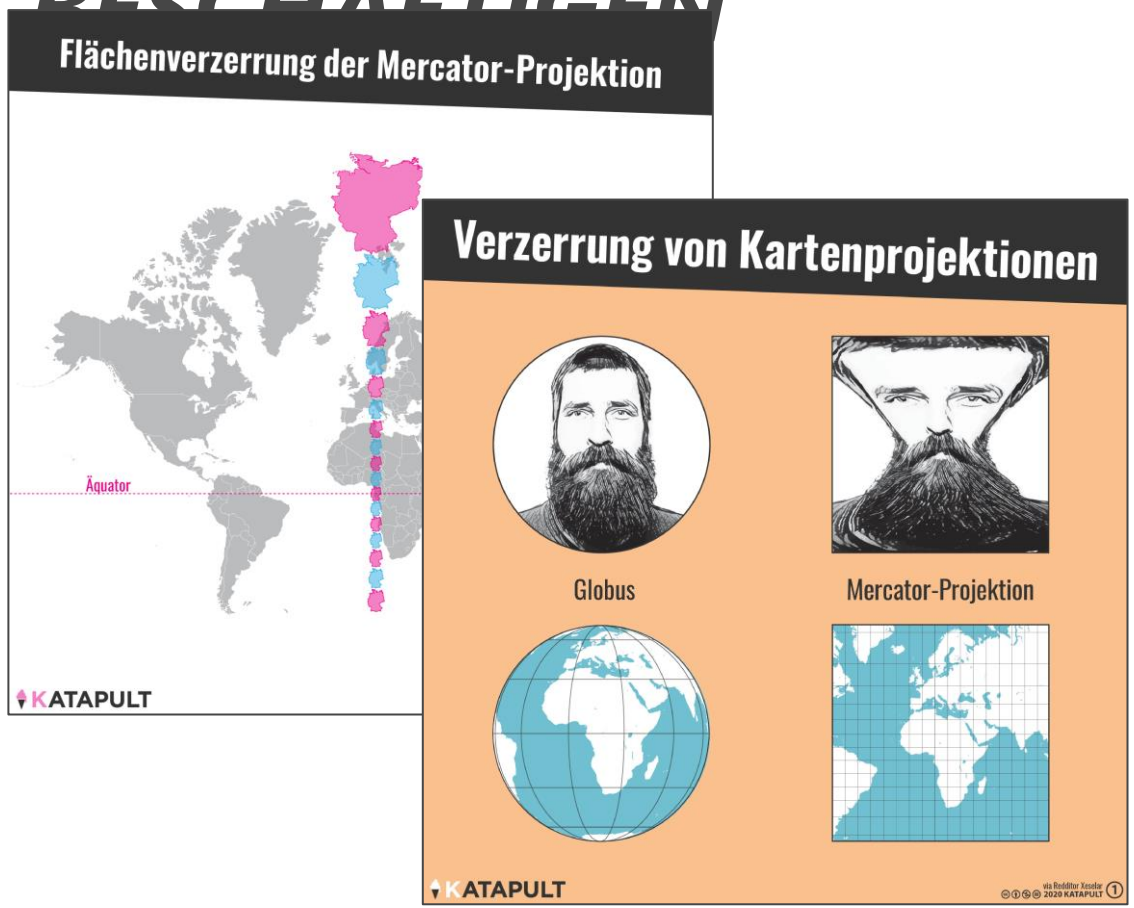
Shapely
https://shapely.readthedocs.io
 	
Geopandas
https://geopandas.org
	▪ Stellt die Geometrischen Grundlagen (Polygone, Punkte etc.)
	
	▪ Pandas mit Geometrie Erweiterung
	▪ Lesen und Schreiben von Geodaten
	▪ Joinen, Analysieren und Plotten
		`from shapely.geometry import Point`
		`import pandas as pd`
		`import geopandas as gpd`
		`df = pd.read_csv("filename.csv")`
		`gdf = gpd.GeoDataFrame(df,`
		`  geometry=df[["lat", "lng"]].apply(Point, axis=1), crs="epsg:4326")`
		`gdf.to_crs(epsg=3395, inplace=True)`
		`gdf["d_nxt_point"] = gdf.distance(gdf.shift())`
		`gdf.to_crs(epsg=4326, inplace=True)`
		`gdf.to_postgis("table_name", engine, if_exists="append")`

Folium
https://python-visualization.github.io/folium/
	▪ Komplexe Visualisierungen
	▪ HTML Export, Mouseover, PopUp etc.
	▪ Einbinden von „Hintergrundkarten“ (zB Mapbox)
		`import folium`
		`center = df[["lat", "lng"]].mean()`
		`m = folium.Map(location=center, tiles="CartoDB positron", zoom_start=6)`
		`for row in df.itertuples():`
		`  folium.CircleMarker(location=[row.lat, row.lng], opacity=0.7).add_to(m)`
		`m.save("map_name.html")`
		`m # displays the map in jupyter`

Osmnx
https://osmnx.readthedocs.io
	▪ Netzwerk Analyse
	▪ Routing Algorithmen
	▪ OSM als Datenbasis
		`import networkx as nx`
		`import osmnx as ox`
		`G = ox.graph_from_place("Hamburg")`
		`start = ox.get_nearest_node(G, (53.5445857,9.9964945)) # Appanion`
		`end = ox.get_nearest_node(G, (53.5664297,9.9855702)) # University HH`
		`route = nx.shortest_path(G, source=start, target=end, weight="length")`
		`fig, ax = ox.plot_graph_route(G, route, node_size=0)`

APPANION

WER MIT GEOKOORDINATEN ARBEITET MUSS SICH MIT KOORDINATENSYSTEMEN BESCHÄFTIGEN



- Zum Darstellen spherischer Daten (Kugeln) auf 2D Objekten (Bildschirmen, Karten), sind Projektionen/ Transformationen nötig
- Es gibt je nach Anwendungsfall unterschiedliche Projektionen und kein allgemeines „richtig“ oder „falsch“
- Üblich ist das Angeben von Geokoordinaten im spherischen „WGS84“ System (=„EPSG:4326“)
- Die meisten Webtools nutzen EPSG:3857 zur Darstellung der WGS84 Daten

Lat, Lon: EPSG4326/ WGS84
Diese sind Flächentreu
Für Distanzberechnung: EPSG 3395*

*Insbesondere bei transkontinentalen Distanzen (Asien – Europe – Amerika) ist hier Vorsicht geboten, da “linksrum” und “rechtsrum” in der kartesischen 2D Darstellung nicht berücksichtigt werden

APPANION

WO ERÖFFNEN WIR AUF BASIS UNSERER NACHFRAGE EIN NÄCHSTES DISTRIBUTIONSZENTRUM?

- Problem:
- Wir haben Distributionszentren und Verteilfahrten zu mehreren Kunden
- Es wird an einem zentralen Standort geladen und zu Kundenstandorten gefahren um zu entladen.
- **Frage: Wo wäre ein Standort rein logistisch (kurze Wege) am sinnvollsten?**
- Es liegen Nachfragedaten der Kunden aus dem vergangenen Jahr nach PLZ vor (Menge und Produkt), bei mehreren Lieferungen bestehen mehrere Einträge

	A	B	C	D	E	F
1	Kunde	Plz. + Ort	Auftrags-Nr.	Produkt	Bestellte Menge	Gepl. Liefertermin
2						
3	Kunde_563	37603 Holzminden	2220284061	Produkt_001	600	03.01.2019
4						
5	Kunde_087	01109 Dresden	2220284144	Produkt_002	1200	03.01.2019
6	Kunde_032	09600 Weißenborn-Berthelsdorf	2220284195	Produkt_003	700	03.01.2019
7	Kunde_481	A-4240 Freistadt	2220284059	Produkt_004	600	04.01.2019
8	Kunde_376	1120 Wien	2220284145	Produkt_005	900	04.01.2019
9	Kunde_774	A-5301 Eugendorf	2220284040	Produkt_006	600	04.01.2019
10	Kunde_484	97539 wonfurt	2220284018	Produkt_006	1000	07.01.2019
11						
12	Kunde_601	68199 Mannheim-Neckarau	2220284143	Produkt_002	800	02.01.2019
13	Kunde_601	68199 Mannheim-Neckarau	2220284143	Produkt_007	500	02.01.2019
14	Kunde_073	08209 Auerbach	2220284477	Produkt_008	1000	03.01.2019
15	Kunde_494	29525 Uelzen	2220284178	Produkt_009	1000	03.01.2019
16	Kunde_794	75223 Niefern-Öschelbronn	2220284298	Produkt_007	600	03.01.2019
17	Kunde_567	74076 Heilbronn	2220283828	Produkt_006	1000	04.01.2019
18						
19	Kunde_084	29525 Uelzen	2220284262	Produkt_010	1000	03.01.2019
20						
21						
22						
23						
24	Kunde_648	85356 Freising	2220284616	Produkt_009	1000	08.01.2019
25	Kunde_648	85356 Freising	2220284616	Produkt_011	600	08.01.2019
26	Kunde_460	18273 Güstrow	2220284626	Produkt_012	900	08.01.2019
27	Kunde_709	97539 Wonfurt	2220284619	Produkt_013	1000	09.01.2019
28	Kunde_709	97539 Wonfurt	2220284619	Produkt_014	1000	09.01.2019

APPANION

VOR DER UMSETZUNG KOMMEN DIE ANNAHMEN UND DAS GROBE KONZEPT

Annahmen	Annahme: Kosten ~ Menge * Distanze
	Welche DCs bedienen zur Zeit welche Kunden (wahrscheinlich)
Status Quo	Wie sehen die Kosten aktuell aus?
Konstruieren der Lösung	Wenn wir in einzelnen PLZ Gebieten DCs hinzufügen, wie sähen die Kosten dann aus?
	Suche nach den PLZs mit den niedrigsten Kosten

APPANION

DAS VORGEHEN BESTEHT AUS VORBEREITUNG/ DATENANREICHERUNG UND ANALYSE

Vorbereitung und explorative Datenanalyse (EDA)

1. Data cleaning: Leerzeilen, Typos, Auftrennen der Adresse, …)
2. Umwandeln der PLZ Daten in Geo-Koordinaten („Senken“ der Transporte)
   nach Recherche zB hier: http://opengeodb.giswiki.org/wiki/OpenGeoDB_Downloads
3. Importieren der eigenen Distributionszentren (DC) („Quellen“ der Transporte)
   Hier ist manuelles Geocoding nötig (z.B. über Google Maps)
4. Plotten der Ergebnisse (immer wieder empfehlenswert um Fehler vorzubeugen

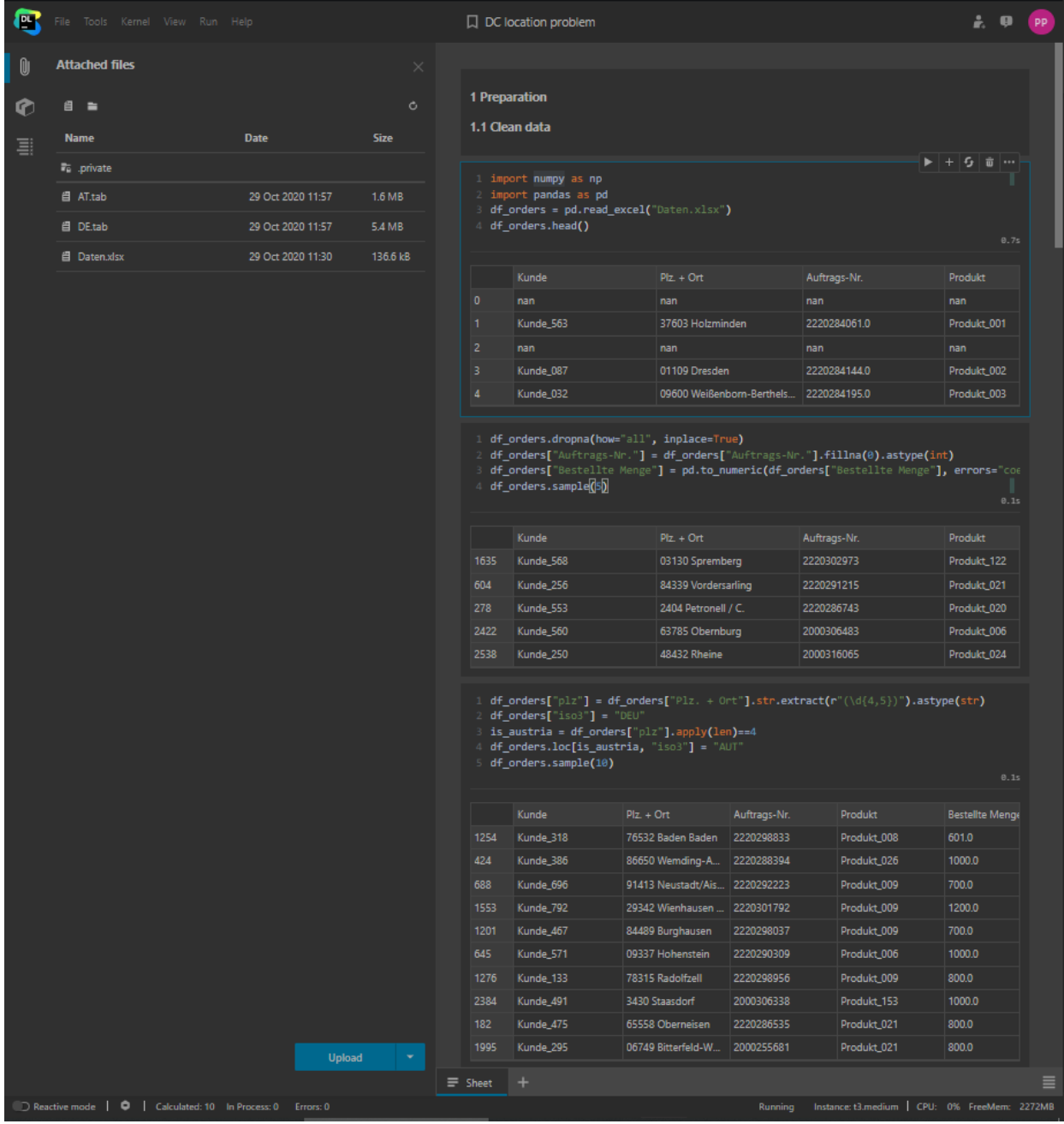
Eigentlicher Berechnungs-Algorithmus

1. Berechnen der nächsten DCs für jeden Kunden
2. Berechnen der Gesamtkosten
3. „Was wäre wenn“: Berechnen der Gesamtkosten, wenn ein zusätzliches DC in einem PLZ-Bereich stünde. Für jeden PLZ Bereich durchführen
4. Ergebnisse sortieren und ggf. plotten

APPANION

DATALORE ALS IDE IN DER CLOUD

- Datalore: JetBrains Alternative zu Jupyter Notebook
- Berechnungen in der Cloud (4GB Maschine Kostenfrei)
- Env in der Cloud
- Publishing & Kollaboration direkt möglich
- https://view.datalore.jetbrains.com/notebook/miZA6n6yui4seavQWP7oiy



APPANION

1 Preparation

1.1 Clean data

```
import numpy as np
import pandas as pd
df_orders = pd.read_excel("Daten.xlsx")
df_orders.head()
```

0.7s

	Kunde	Plz. + Ort	Auftrags-Nr.	Produkt	Bestellte Menge	Gepl. Liefertermin
0	nan	nan	nan	nan	nan	NaT
1	Kunde_563	37603 Holzminden	2220284061.0	Produkt_001	600	2019-01-03 00:00:00
2	nan	nan	nan	nan	nan	NaT
3	Kunde_087	01109 Dresden	2220284144.0	Produkt_002	1200	2019-01-03 00:00:00
4	Kunde_032	09600 Weißenborn-Berthels...	2220284195.0	Produkt_003	700	2019-01-03 00:00:00

```
df_orders.dropna(how="all", inplace=True)
df_orders["Auftrags-Nr."] = df_orders["Auftrags-Nr."].fillna(0).astype(int)
df_orders["Bestellte Menge"] = pd.to_numeric(df_orders["Bestellte Menge"], errors="coerce")
df_orders.sample(5)
```

0.1s

	Kunde	Plz. + Ort	Auftrags-Nr.	Produkt	Bestellte Menge	Gepl. Liefertermin
1635	Kunde_568	03130 Spremberg	2220302973	Produkt_122	500.0	2019-08-21 00:00:00
604	Kunde_256	84339 Vordersarling	2220291215	Produkt_021	600.0	2019-03-28 00:00:00
278	Kunde_553	2404 Petronell / C.	2220286743	Produkt_020	1000.0	2019-02-12 00:00:00
2422	Kunde_560	63785 Obernburg	2000306483	Produkt_006	700.0	2019-12-02 00:00:00
2538	Kunde_250	48432 Rheine	2000316065	Produkt_024	700.0	2019-12-17 00:00:00

```
df_orders["plz"] = df_orders["Plz. + Ort"].str.extract(r"(\d{4,5})").astype(str)
df_orders["iso3"] = "DEU"
is_austria = df_orders["plz"].apply(len)==4
df_orders.loc[is_austria, "iso3"] = "AUT"
df_orders.sample(10)
```

0.1s

	Kunde	Plz. + Ort	Auftrags-Nr.	Produkt	Bestellte Menge	Gepl. Liefertermin	plz	iso3
1254	Kunde_318	76532 Baden Baden	2220298833	Produkt_008	601.0	2019-06-27 00:00:00	76532	DEU
424	Kunde_386	86650 Wemding-A...	2220288394	Produkt_026	1000.0	2019-02-28 00:00:00	86650	DEU
688	Kunde_696	91413 Neustadt/Ais...	2220292223	Produkt_009	700.0	2019-04-04 00:00:00	91413	DEU
1553	Kunde_792	29342 Wienhausen ...	2220301792	Produkt_009	1200.0	2019-08-07 00:00:00	29342	DEU
1201	Kunde_467	84489 Burghausen	2220298037	Produkt_009	700.0	2019-06-18 00:00:00	84489	DEU
645	Kunde_571	09337 Hohenstein	2220290309	Produkt_006	1000.0	2019-03-27 00:00:00	09337	DEU
1276	Kunde_133	78315 Radolfzell	2220298956	Produkt_009	800.0	2019-07-03 00:00:00	78315	DEU
2384	Kunde_491	3430 Staasdorf	2000306338	Produkt_153	1000.0	2019-11-27 00:00:00	3430	AUT
182	Kunde_475	65558 Oberneisen	2220286535	Produkt_021	800.0	2019-01-30 00:00:00	65558	DEU
1995	Kunde_295	06749 Bitterfeld-W...	2000255681	Produkt_021	800.0	2019-10-08 00:00:00	06749	DEU

APPANION

1.2 Join Geocoordinates

```python
df_plz = pd.DataFrame()
for country in [["AT.tab", "AUT"], ["DE.tab", "DEU"]]:
    df_country = pd.read_csv(country[0], sep="\t", usecols=["name", "plz", "lat", "lon"])
    df_country["plz"] = df_country["plz"].astype(str)
    df_country["iso3"] = country[1]
    if country[1]=="AUT":
        df_country["lat"] = df_country["lat"].str.replace(" ", "")
        df_country["lon"] = df_country["lon"].str.replace(" ", "")
    df_plz = pd.concat([df_plz,df_country])

df_plz["lat"] = pd.to_numeric(df_plz["lat"])
df_plz["lon"] = pd.to_numeric(df_plz["lon"])
df_plz["plz"] = df_plz["plz"].apply(lambda x: x.split(","))
df_plz = df_plz.explode("plz")

df_plz = df_plz.groupby(["iso3", "plz"], as_index=False).agg({"lat":"mean", "lon":"mean", "name":"first"})
df_plz.dropna(how="any", inplace=True)
print(df_plz.sample(10))
```

2.2s

```
     iso3    plz        lat        lon                      name
6967  DEU  66879  49.485089   7.520175                Kollweiler
3821  DEU  24214  54.402541   9.971958                   Gettorf
4429  DEU  29568  52.907898  10.685904                    Wieren
1242  AUT   5580  47.145452  13.814714                   Lessach
8549  DEU  83679  47.806255  11.643358                Sachsenkam
7901  DEU  76770  49.111100   8.245280                Hatzenbühl
7255  DEU  70184  48.773040   9.190860                 Stuttgart
6094  DEU  54498  49.880550   6.916667                  Piesport
3951  DEU  24876  54.471851   9.346850  Hollingstedt bei Schleswig
445   AUT   2871  47.505113  16.134435                    Zöbern
```

1.3 Own locations

```python
distribution_centers = {
    "Hamburg": {"lat":53.5445082, "lon":9.9959688},
    "Frankfurt": {"lat":50.11245, "lon":8.672099},
    "Berlin":{"lat":52.5208045, "lon":13.4092927},
    "Essen":{"lat":51.4518119,"lon":7.0135793},
    "Stuttgart":{"lat":48.7785873,"lon":9.1797865},
    "Muenchen":{"lat":48.1418032, "lon":11.5793887}
}
df_dcs = pd.DataFrame(distribution_centers).T
df_dcs
```

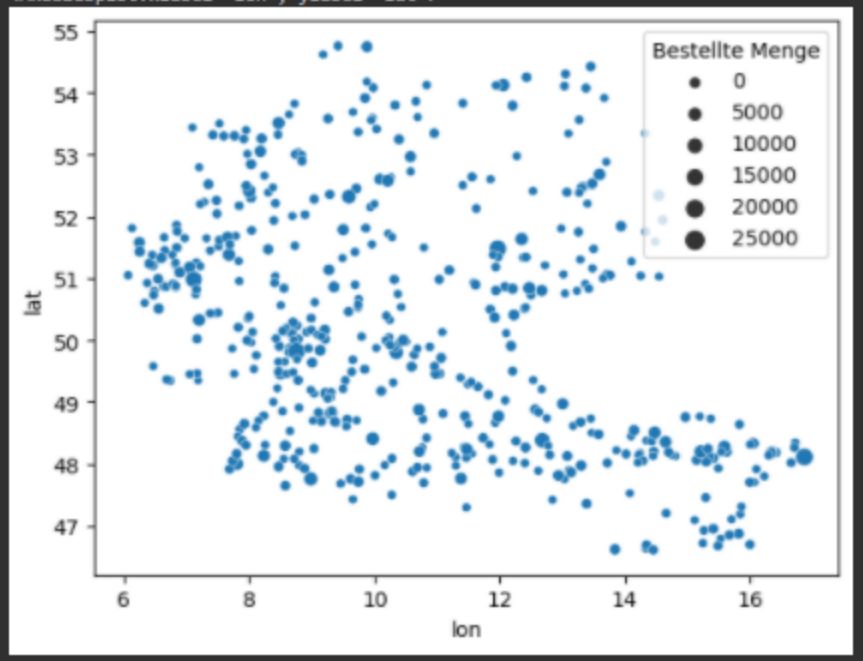
	lat	lon
Ham...	53.5445082	9.9959688
Frank...	50.11245	8.672099
Berlin	52.5208045	13.4092927
Essen	51.4518119	7.0135793
Stutt...	48.7785873	9.1797865
Muen...	48.1418032	11.5793887

APPANION

1.4 Plot

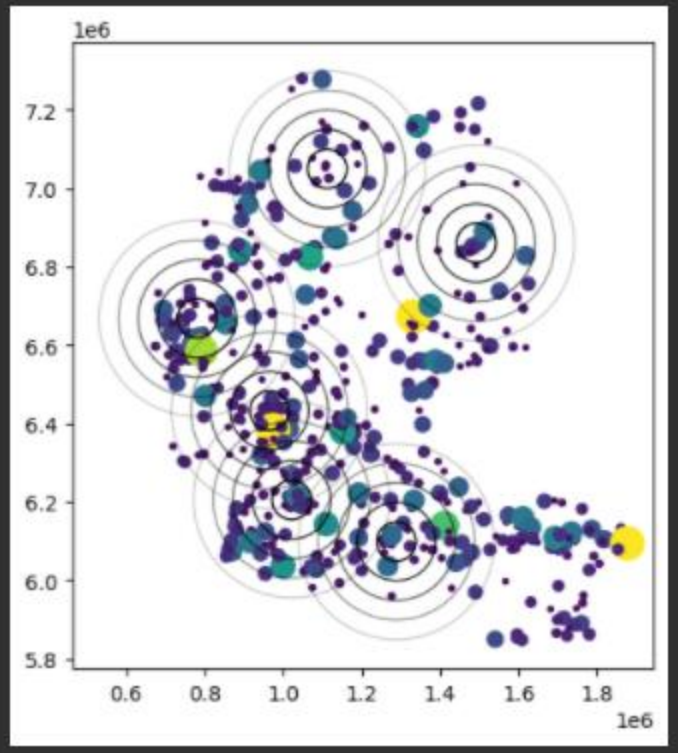
```
import seaborn as sns
df_quantities = df_orders.groupby(["plz", "iso3"], as_index=False)["Bestellte Menge
df_quantities = df_quantities.merge(df_plz, on=["iso3", "plz"])
sns.scatterplot(x="lon", y="lat", size="Bestellte Menge", data=df_quantities)
```

```
<AxesSubplot:xlabel='lon', ylabel='lat'>
```




```
import matplotlib.pyplot as plt
from shapely.geometry import Point
import geopandas as gpd
gdf_quantities = gpd.GeoDataFrame(df_quantities, geometry=df_quantities[["lon","lat"]].apply(Point, axis=1), crs="epsg:4326").to_crs(epsg=3395)

gdf_dcs = gpd.GeoDataFrame(df_dcs, geometry=df_dcs[["lon","lat"]].apply(Point, axis=1), crs="epsg:4326").to_crs(epsg=3395)

ax = gdf_quantities.plot(markersize=gdf_quantities["Bestellte Menge"]/100, column="Bestellte Menge",
                         legend_kwds={"label": "Bestellte Menge"},
                         figsize=(5, 10)
                         )

rads = [50000, 100000, 150000, 200000, 250000]
for i, buff in enumerate(rads):
    gdf_dcs.buffer(buff).plot(ax=ax, facecolor="none", edgecolor="black", alpha=(len(rads)-i)/len(rads))

plt.show()
```

1.6s



2. Computation

```python
def calculate_total_costs(gdf_quantities, gdf_dcs):
    dist = pd.DataFrame()
    for dc in gdf_dcs.itertuples():
        dist[dc.Index] = gdf_quantities.distance(dc.geometry)/1000
    nearest_dc = dist.idxmin(axis=1)

    costs = 0
    for dc in gdf_dcs.itertuples():
        mask = nearest_dc==dc.Index
        costs += (dist.loc[mask, dc.Index]*gdf_quantities.loc[mask,"Bestellte Menge"]).sum()

    return costs/1000

print("Current costs, all dcs: ", calculate_total_costs(gdf_quantities, gdf_dcs))
for dc in gdf_dcs.itertuples():
    print(f"without {dc.Index}: ", calculate_total_costs(gdf_quantities, gdf_dcs.drop(dc.Index)))
```

0.4s

```
Current costs, all dcs:  282452.0621702564
without Hamburg:  314396.7495219027
without Frankfurt:  306881.63020130317
without Berlin:  324963.0485194885
without Essen:  319678.5429218514
without Stuttgart:  309836.23662141577
without Muenchen:  380940.72011354886
```

```python
gdf_plz = gpd.GeoDataFrame(df_plz, geometry=df_plz[["lon", "lat"]].apply(Point, axis=1), crs="epsg:4326").to_crs(epsg=3395)

gdf_plz["total_costs"] = 9999999
for idx, row in gdf_plz.iterrows():
    one_more_dc = gdf_dcs.append(row)
    gdf_plz.loc[idx, "total_costs"] = calculate_total_costs(gdf_quantities, one_more_dc)

gdf_plz = gdf_plz.sort_values("total_costs")
print(gdf_plz[["iso3", "plz", "name", "total_costs"]].head(10))
```

7:03m

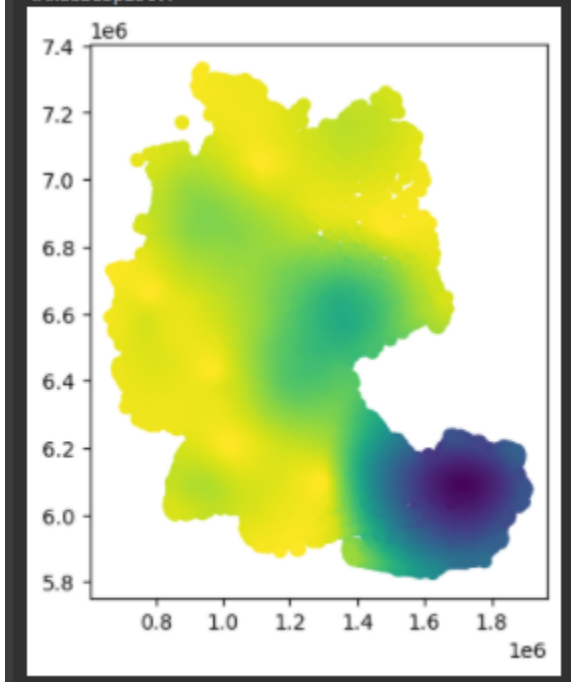
	iso3	plz	name	total_costs
523	AUT	3240	Kilb	212623.735876
522	AUT	3233	Bischofstetten	212636.347178
586	AUT	3383	Hürm	212724.227533
526	AUT	3243	Kirnberg an der Mank	212763.951682
521	AUT	3232	Bischofstetten	212764.508877
524	AUT	3241	Kirnberg an der Mank	212764.886997
520	AUT	3231	Sankt Margarethen an der Sierning	212896.290979
509	AUT	3202	Hofstetten-Grünau	212901.059788
510	AUT	3203	Hofstetten-Grünau	212901.817323
527	AUT	3244	Ruprechtshofen	212904.591380

APPANION

```
1 print(gdf_plz[["iso3", "plz", "name", "total_costs"]].head())
2 print(gdf_plz[["iso3", "plz", "name", "total_costs"]].tail())
3 gdf_plz.plot(column="total_costs")
```

1.7s

```
    iso3   plz                 name    total_costs
523  AUT  3240                 Kilb  212623.735876
522  AUT  3233        Bischofstetten  212636.347178
586  AUT  3383                 Hürm  212724.227533
526  AUT  3243  Kirnberg an der Mank  212763.951682
521  AUT  3232        Bischofstetten  212764.508877
     iso3    plz       name    total_costs
3075  DEU  13442     Berlin  282434.978649
3087  DEU  13589     Berlin  282434.978649
3089  DEU  13593     Berlin  282434.978649
7249  DEU  70173  Stuttgart  282444.854873
7254  DEU  70182  Stuttgart  282444.854873
<AxesSubplot:>
```



APPANION

MIT DEN RICHTIGEN TOOLS IST DIE ANALYSE VON GEODATEN IN PYTHON EFFIZIENT UND EFFEKTIV

- Es gibt zahlreiche Pakete zur Verarbeitung von Geodaten in Python
  - Shapely für Geometrie-Objekte allgemein
  - Geopandas (Pandas mit „geometry“-Spalte)
  - Folium zum Plotten hochqualitativer Grafiken
  - Osmnx für Netzwerkanalysen
  - …
  - Plotten funktioniert wie gewohnt mit Matplotlib, Seaborn, …

- Wichtig ist die Wahl des „Coordinate Reference Systems“ (CRS):
  - EPSG: 4326 – Mercator
  - EPSG: 3395 – DE-zentristisch für Distanzen

Philipp Huhn
Chief Data Scientist
philipp@appanion.com

APPANION

BACKUP

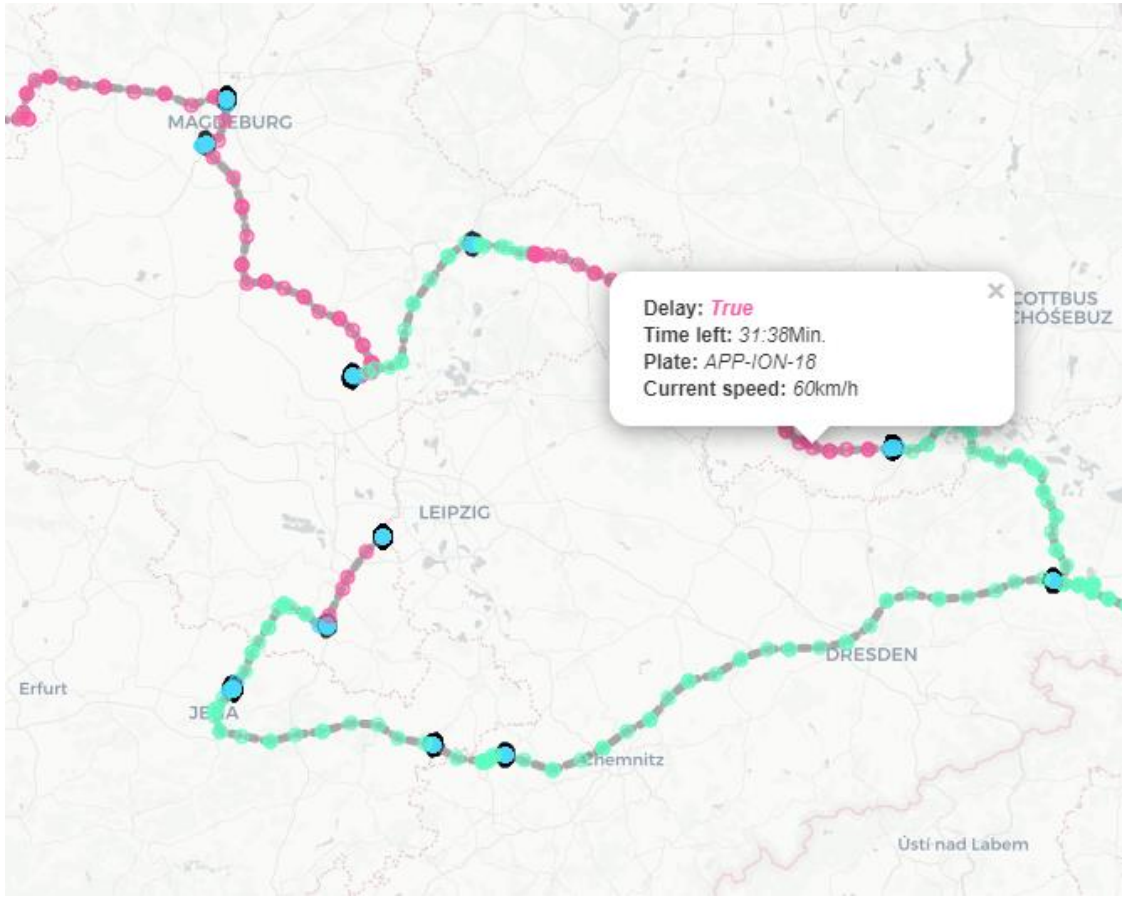
BEISPIEL 1: INTEGRIERTES GEOTRACKING

2 DATA PRODUCTS & SERVICES

Use Case: Geolocation und -fencing

- Nutzung von Telematik- und Tracking-Daten
- Abgleich von Trackingdaten und dynamischen Geozonen (Standorte, Lieferorte, Wegpunkte)
- Fortlaufende Informationen:
  - Wo sind Unregelmäßigkeiten vorgefallen?
  - Wo sind Güter aktuell?
  - Was ist der Status-Quo der Ware?
  - Wurden Güter pünktlich versandt?
  - Wurden Fahr- und Bremsregularien eingehalten?
  - Wird es eine Verpätung bei der Lieferung geben?

Erhöhung von Schnelligkeit und Effizienz durch Verknüpfung von bereits vorhandenen Daten



APPANION