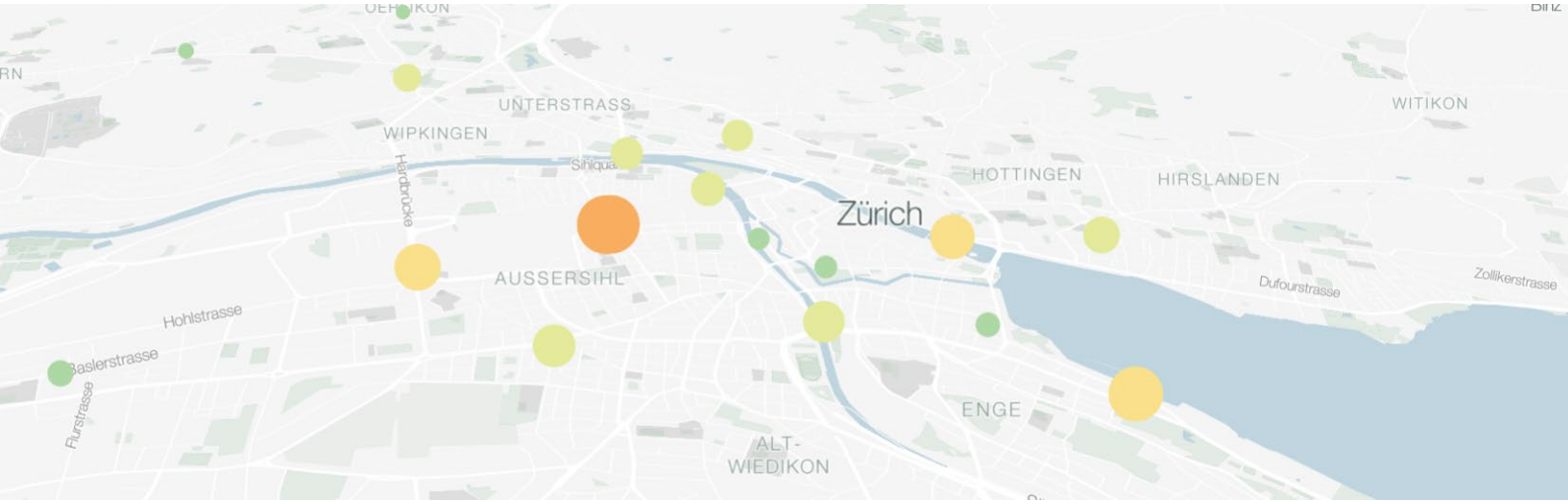
Mobilität in Zürich im COVID-19-Pandemie-Jahr 2020



GEO372 Vertiefung Geographische Informationswissenschaft
HS 2020

Universität Zürich
Geographisches Institut

Ueli Isenschmid

Webkarte unter
https://bountan.github.io/mobilitycovid/

Zürich, 21. Dezember 2020

Inhaltsverzeichnis

Einleitung

Ziel der vorliegenden Arbeit ist die Visualisierung der Veränderung von Mobilität in der Stadt Zürich im Verlauf der verschiedenen Massnahmen zur Eindämmung der COVID-19-Epidemie im Jahr 2020. Datenbasis sollen Open Data der Stadt Zürich sein. Ausgewählt werden zwei Datensätze aus der Verkehrszählung der Stadt Zürich, jener zum motorisierten Individualverkehrs sowie jener zum Velo- und Fussgängerverkehr. Aufgrund der grossen Anzahl Messstationen, der verschiedenen Verkehrsmodi und der hohen Messfrequenz wird gegen eine statische Darstellung des Materials entschieden, um die Detailgenauigkeit der Datengrundlage nicht übermässig einschränken zu müssen. Im Sinne der Herkunft der Datengrundlage aus Open Data wird versucht, eine Visualisierung mithilfe von freien Werkzeugen zu erstellen. Die Wahl fällt auf die Datenaufbereitung mittels Python-Modulen und dem Angebot der Resultate durch eine interaktive Webkarte. So soll die detailgenaue Verortung der Messstationen mit einer guten zeitlichen und räumlichen Lesbarkeit verbunden werden. Durch die interaktive Komponente kann das Resultat unter Umständen auch ein interessiertes Laienpublikum interessieren.

Methoden

Die Daten beider Verkehrszählungen wurden durch das Tiefbauamt durch automatische Messungen erhoben Die Fussgänger- und Velofrequenz wird durch passive Infrarotstrahlung gezählt, Autoverkehr durch Induktionsschlaufen im Strassenbelag (Stadt Zürich 2020a, Stadt Zürich 2020b). Autozählungen finden an anderen Standorten statt als Fussgänger- und Velozählung. Bei Stationen der Fussgänger- und Velozählung werden bei hohem Veloanteil eine eigene Velozählstation installiert. Beide Messwerte werden durch manuell erhobene Korrekturwerte angepasst. Die Fussgänger- und Velozählung ist viertelstundenscharf aufsummiert, die Autozählung stundenscharf. Die Datensätze werden täglich aktualisiert, zum Zeitpunkt der Erstellung dieser Arbeit wird also der Zeitraum vom 1. Januar 2020 bis 16. Dezember 2020 abgedeckt.

Die Daten liegen als kommagetrennte Tabellen vor und enthalten jeweils Attribute der geografischen Länge und Breite im Format LV95.

Fussgänger- und Velozählung

Die Daten werden durch das Pythonmodul Pandas importiert. Eine Datenzeile repräsentiert jeweils eine Messung einer Station mit codiertem Standort, kann aber Velo- oder Fussgängerzählung repräsentieren. Die beiden Zählungen für unterschiedliche Verkehrsrichtungen werden zunächst aufsummiert und es liegen nun zwei Attribute «Veloverkehr total» und «Fussverkehr total» vor. Jeweils ein Attribut bleibt leer, mittels einer Logikschleife wird dadurch der Verkehrsmodus als neues Attribut generiert und die Zählungstotale in ein Attribut «Total Zählung» kombiniert. Aus dem Attribut Zählungsdatum wird ein Attribut «Kalenderwoche» erstellt. Der Datensatz wird aggregiert nach den Attributen Standort, Kalenderwoche und Verkehrsmodus und das Attribute «Total Zählung» jeweils aufsummiert. Die zwei Koordinatenattribute werden entsprechend dem Standort jeweils erhalten. Alle nicht mehr benötigten Attribute werden gelöscht.

Autozählung

Die Daten liegen in gleichem Format vor und werden mit gleichem Vorgehen verarbeitet. Das Attribut «Verkehrsmodus» wird jedoch uniform allen Datenzeilen zugeschrieben. Dieser Datensatz enthält weitere Attribute, die aber allesamt nicht benötigt werden.

Datenausgabe

Die beiden Datensätze haben nun die gleichen Attribute und werden zusammengefügt. Eine Datenzeile repräsentiert eine Messung mit klar zugeordnetem und geocodiertem Standort, der Kalenderwoche der Messung, dem Verkehrsmodus, und der Gesamtzahl Messungen gemäss diesen Attributen. Aus den Koordinatenattributen wird ein geocodiertes Punktelement erstellt. Der Datensatz wird in eine geocodierte JSON-Datei (.geojson) ausgegeben.

	loc	wk	kind	dtstr	n	geometry
0	2	1	1	01.01.2020	5288	POINT (8.48750 47.39099)
1	2	2	1	06.01.2020	13469	POINT (8.48750 47.39099)
2	2	3	1	13.01.2020	14155	POINT (8.48750 47.39099)
3	2	4	1	20.01.2020	13787	POINT (8.48750 47.39099)
4	2	5	1	27.01.2020	13789	POINT (8.48750 47.39099)
...	...	...	...	...	...	...
8293	3927	47	2	16.11.2020	5481	POINT (8.53562 47.35866)
8294	3927	48	2	23.11.2020	4932	POINT (8.53562 47.35866)
8295	3927	49	2	30.11.2020	3370	POINT (8.53562 47.35866)
8296	3927	50	2	07.12.2020	3191	POINT (8.53562 47.35866)
8297	3927	51	2	14.12.2020	657	POINT (8.53562 47.35866)

8298 rows × 6 columns

Tabelle 1: Datenformat nach Aufarbeitung

Darstellung

Als Grundlage der Darstellung dient die Javascript-Bibliothek von Mapbox GL. Sie erlaubt das Erstellen von interaktiven Karten und Steuerelementen auf Javascriptbasis und wird in eine HTML-basierte Webseite integriert. Teile der HTML-Struktur wurden vom Autor bereits für frühere Arbeiten genutzt. Als Basiskarte dient eine kachelbasierte Karte, die ebenfalls mithilfe von Mapbox erstellt wird, in Farbe, Typographie und Detailgrad frei anpassbar ist und von jedem Mapbox-basierten Kartenelement mittels eines privaten Tokens abgerufen werden kann. Das Werkzeug ist gratis im Rahmen der nichtgewerblichen Nutzung. Die Seite wird auf Github.com gehostet, da bereits in der Erstellung als Repository so gespeichert wurde und über «Github Pages» die Möglichkeit zur Verfügung steht, die Seite direkt aus dem Repository abzurufen und öffentlich anzubieten.

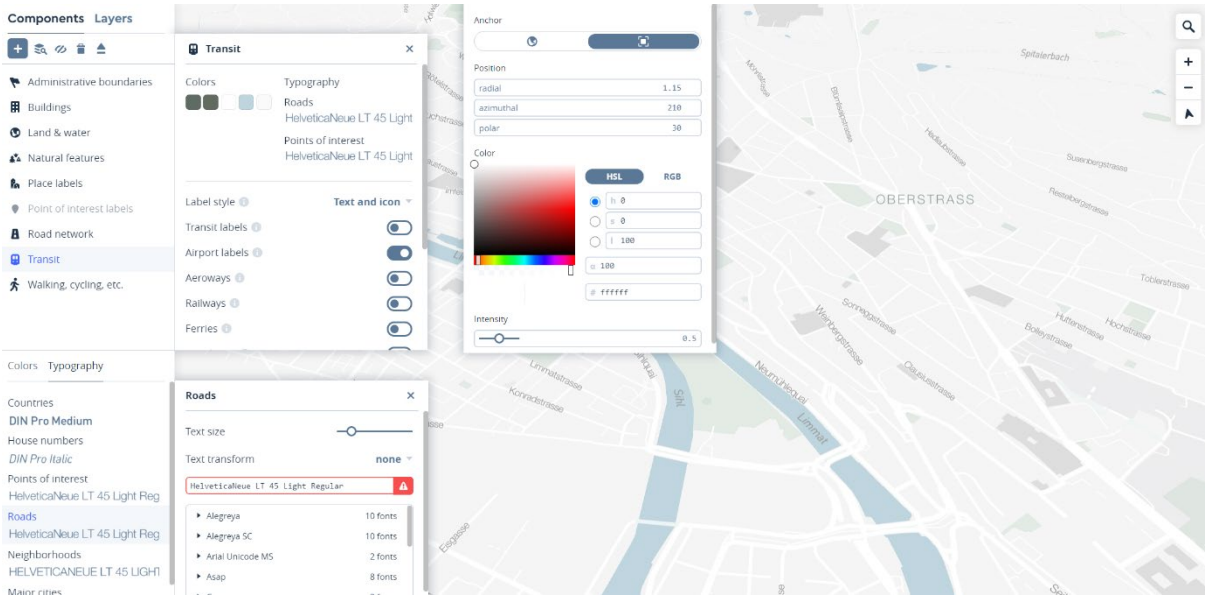


Abbildung 1: Benutzeroberfläche Mapbox Studio

Die Karte wird möglichst kartografisch vollständig ausgestattet mit Titel, Begleittext, Legende, Massstabsanzeige sowie Quellen- und Autorenangaben. Die Steuerelemente bestehen aus einem Schieberegler zur Auswahl der Kalenderwoche und einer Schaltfläche zur Auswahl des darzustellenden Verkehrsmodus. Das geojson-Element mit den geocodierten Zählungsresultaten wird per der Javascript-Bibliothek abgerufen. Die Darstellung der Zählwerte erfolgt durch die grafischen Variablen von Farbe und Grösse, die jedoch redundant auf den gleichen Wert verweisen, dies aus Überlegung der visuellen Attraktivität und Verdeutlichung.

Datenqualität

Die Qualität der verwendeten Daten ist im Allgemeinen hoch. Die Darstellung als Webkarte scheint für diese und viele weitere Anwendungen von zeitlich und räumlich basierten Daten optimal. Die Messungen scheinen weitgehend vollständig, Lücken werden plausibel erklärt, beispielsweise wie im Falle der Zeitumstellung Stundenlücken entstehen. Die Messungen sind durch fixierte Messstationen natürlich räumlich sehr exakt, die Zählung an sich ist in ihrer Exaktheit und Zuverlässigkeit durch technische Gründe limitiert, was aber annehmbar durch Abgleich mit manuellen Zählungen und Einführung von mittleren Korrekturwerten begegnet wird (Stadt Zürich 2020a). Die Daten werden täglich aktualisiert, eine zeitnahe Frequenz. Die Daten sind relevant und wertvoll für Anwendungen in Verkehrs- und Raumplanung, dies natürlich auch im Rahmen von plötzlich sich änderndem Verhalten vieler Menschen im öffentlichen Raum. Die Daten erfüllen die selbstgesetzten Anforderungen von Open Data Stadt Zürich und sind frei und relativ einfach zugänglich und sehr gut dokumentiert.

Resultat und Diskussion

Das Resultat der interaktiven Webkarte ist abrufbar unter https://bountan.github.io/mobilitycovid/. Durch die Aufbereitung der Daten mit freier Software entstehen Datenmaterialien mit Zwischenschritten, die jeweils wieder für neue Anwendungen dienlich sein können und auch nach Fertigstellung des Kartenmaterials jederzeit und einfach den Rückgriff auf statistisch auswertbare Daten ermöglichen. Die Gebundenheit an kommerzielle Anbieter und Plattformen bleibt aus.

Abbildung 2: Benutzeroberfläche Webkarte

Die Zusammenfassung der zeitlichen Frequenz in Kalenderwochen scheint ein guter Kompromiss zwischen Detailtreue und Übersichtlichkeit und die sich aus der COVID-19-Pandemie ergebenden Änderungen von Bewegungsmustern scheint wochenscharf annehmbar abbildbar.

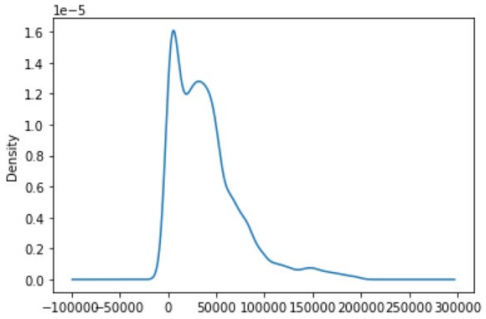


Abbildung 3: Verteilung der wochensummierten Zählwerte mit Peaks im Bereich Grössenordnung 100 (Langsamverkehr) und Peak im Bereich Grössenordnung 40'000 (Autoverkehr)

Die Karte müsste für eine weitere Verwendung in Hinblick auf ihre grafischen Variablen weiterentwickelt werden. Die Parametrisierung von Farb- und Grössenwahl der Kartenelement ist ungenügend angepasst für die grosse Bandbreite an Zählwerten, die sich durch Zusammenfassung von Auto-, Fussgänger- und Velofrequenz ergibt, mit Werten der Grössenordnung 100'000 bei Ausfallstrassen bis zu Werten der Grössenordnung 100 bei Fussgängerpassagen. Die zweite grafische Variable könnte wertvoller eingesetzt werden als nur zur Betonung der bereits bestehenden. Die aktuellen Massnahmen bezüglich des Pandemie-Status sollten dominanter und grafisch ansprechender unterlegt werden.

Grundsätzlich scheint die erhaltene Webkarte aber einen effizienten und attraktiven Einblick in die Bewegungsmuster im Jahresverlauf zu geben, die durch eine statische Karte so nicht möglich ist. Ebenfalls ist die Verwendung von kostenfreien und frei anpassbaren Werkzeugen wertvoll und hat sich im Rahmen dieser Arbeit als leistungsfähigeren GIS-Werkzeugen nicht unterlegen gezeigt.

Verzeichnisse

Quellenverzeichnis

Institut für Verkehrsplanung ETH Zürich (2020): Mobilitätsverhalten in der Schweiz - Coronavirus-Studie. https://ivtmobis.ethz.ch/mobis/covid19/reports/latest_de. Abgerufen am 16.12.2020.

Stadt Zürich Open Data (2020a). Daten der automatischen Fussgänger- und Velozählung – Viertelstundenwerte. https://data.stadt-zuerich.ch/dataset/ted_taz_verkehrszaehlungen_werte_fussgaenger_velo. Abgerufen am 16.12.2020.

Stadt Zürich Open Data (2020b). Daten der Verkehrszählung zum motorisierten Individualverkehr (Stundenwerte). https://data.stadt-zuerich.ch/dataset/sid_dav_verkehrszaehlung_miv_od2031. Abgerufen am 16.12.2020.

Bildverzeichnis

Tabellenverzeichnis

Anhänge

Anhang 1: Jupyter Notebook Datenbearbeitung mit Python

```
import os
from IPython.display import display, HTML
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import geopandas as gpd
from shapely.geometry import Point
from shapely.geometry import LineString
import shapely.geometry as geom
import datetime as dt
import csv
import json
pd.set_option("display.max_rows", 20, "display.max_columns", None)
```

```
data import
In [94]:
#data directory
datadir = os.path.join(os.path.abspath(''),'data')
outdir = os.path.join(os.path.abspath(''),'out')

#counting data
#nrows limits data amount during code test , nrows=10000
bikped = pd.read_csv(os.path.join(datadir, 'bikped.csv'), encoding = "utf-8", sep=',')
miv = pd.read_csv(os.path.join(datadir, 'miv.csv'), encoding = "utf-8", sep=',')

#id of counting machine irrelevant, drop
bikped=bikped.drop(columns=["FK_ZAEHLER"])
```

```
extract and aggregate relevant data bike and pedestrian data
In [95]:
agg=bikped.copy()
agg.shape
#get day as int from datetime (gejson export doesnt accept dt-format) and create attribute that contai
ns "readable" date
agg["DATUM"]=pd.to_datetime(agg["DATUM"])


#readable date
agg["dtstr"] = agg["DATUM"].dt.strftime("%d.%m.%Y")
agg["DATUM"]=agg["DATUM"].dt.isocalendar().week.astype(int)

#create attribute for means of transport: bike and pedestrian
def assignkind(row):
    if pd.isna(row['VELO_IN']) & pd.isna(row['VELO_OUT']):
        val = 1
    elif pd.isna(row['FUSS_IN']) & pd.isna(row['FUSS_OUT']):
        val = 2
    else:
        val = 0
    return val

agg['kind'] = agg.apply(assignkind, axis=1)

#sum counts to kind, location, date, and keep coords and station id
agg = agg.groupby(['FK_STANDORT', 'DATUM', 'kind']).agg(
                bik=('VELO_IN', 'sum'),
                ped=('FUSS_IN', 'sum'),
                dtstr=('dtstr', 'first'),
                lon=('OST', 'first'),
                lat=('NORD', 'first'))

agg['bik']=agg['bik'].astype(int)
agg['ped']=agg['ped'].astype(int)
display(agg)
#reset index
```

```
agg['i'] = np.arange(len(agg))
agg = agg.reset_index()
agg.set_index('i')
agg=agg.drop(columns=['i'])

#rename columns
agg=agg.rename(columns={
    "FK_STANDORT" : "loc",
    "DATUM" : "wk",

    })
display(agg)
```

			bik	ped	dtstr	lon	lat
FK_STANDORT	**DATUM**	**kind**					
2	**1**	**1**	0	5288	01.01.2020	2679190	1249436
	2	**1**	0	13469	06.01.2020	2679190	1249436
	3	**1**	0	14155	13.01.2020	2679190	1249436
	4	**1**	0	13787	20.01.2020	2679190	1249436
	5	**1**	0	13789	27.01.2020	2679190	1249436
...	...	...	...	...	...	...	...
3927	**47**	**2**	5481	0	16.11.2020	2682873	1245891
	48	**2**	4932	0	23.11.2020	2682873	1245891
	49	**2**	3370	0	30.11.2020	2682873	1245891
	50	**2**	3191	0	07.12.2020	2682873	1245891
	51	**2**	657	0	14.12.2020	2682873	1245891

1895 rows × 5 columns

	loc	wk	kind	bik	ped	dtstr	lon	lat
0	2	1	1	0	5288	01.01.2020	2679190	1249436
1	2	2	1	0	13469	06.01.2020	2679190	1249436
2	2	3	1	0	14155	13.01.2020	2679190	1249436
3	2	4	1	0	13787	20.01.2020	2679190	1249436
4	2	5	1	0	13789	27.01.2020	2679190	1249436
...	...	...	...	...	...	...	...	...
1890	3927	47	2	5481	0	16.11.2020	2682873	1245891
1891	3927	48	2	4932	0	23.11.2020	2682873	1245891
1892	3927	49	2	3370	0	30.11.2020	2682873	1245891
1893	3927	50	2	3191	0	07.12.2020	2682873	1245891
1894	3927	51	2	657	0	14.12.2020	2682873	1245891

1895 rows × 8 columns

extract and aggregate relevant car data

In [96]:

```
car = miv.copy()


#get day as string from datetime (gejson export doesnt accept dt-format) and create attribute that con
tains "readable" date
car["MessungDatZeit"]=pd.to_datetime(car["MessungDatZeit"])

#readable date
car["dtstr"] = car["MessungDatZeit"].dt.strftime("%d.%m.%Y")

car["MessungDatZeit"]=car["MessungDatZeit"].dt.strftime("%U").astype(int)

#set kind to c for "car"
car['kind'] = 3


#sum counts to kind, location, date, and keep coords and station id
car = car.groupby(['Knummer', 'MessungDatZeit', 'EKoord', 'NKoord']).agg(
                car=('AnzFahrzeuge', 'sum'),
                dtstr=('dtstr', 'first'),
                kind=('kind', 'first'))
car['car']=car['car'].astype(int)

#reset index
car['i'] = np.arange(len(car))
car = car.reset_index()
car.set_index('i')
car=car.drop(columns=['i'])

#rename columns
car=car.rename(columns={
    "Knummer" : "loc",
    "MessungDatZeit" : "wk",
    "EKoord" : "lon",
    "NKoord" : "lat",
    })
```

merge miv and bike/pedestrian

In [97]:

```
biped =agg.copy()

fulldata=biped.append(car)

fulldata = fulldata.groupby(['loc', 'wk', 'kind', 'lon', 'lat']).agg(
                car=('car', 'sum'),
                ped=('ped', 'sum'),
                bik=('bik', 'sum'),
                dtstr=('dtstr', 'first'))
fulldata['n']=(fulldata['ped']+fulldata['car']+fulldata['bik']).astype(int)


#reset index
fulldata['i'] = np.arange(len(fulldata))
fulldata = fulldata.reset_index()
fulldata.set_index('i')
fulldata=fulldata.drop(columns=['i'])

fulldata['lontemp']=fulldata['lat']
fulldata['lat']=fulldata['lon']
fulldata['lon']=fulldata['lontemp']
fulldata=fulldata.drop(columns=['lontemp', 'bik', 'ped', 'car'])
fulldata['kind']=fulldata['kind'].astype(int)
display(fulldata)
```

	loc	wk	kind	lon	lat	dtstr	n
0	2	1	1	1249436.0	2679190.0	01.01.2020	5288
1	2	2	1	1249436.0	2679190.0	06.01.2020	13469
2	2	3	1	1249436.0	2679190.0	13.01.2020	14155

	loc	wk	kind	lon	lat	dtstr	n
3	2	4	1	1249436.0	2679190.0	20.01.2020	13787
4	2	5	1	1249436.0	2679190.0	27.01.2020	13789
...	...	...	...	...	...	...	...
8293	3927	47	2	1245891.0	2682873.0	16.11.2020	5481
8294	3927	48	2	1245891.0	2682873.0	23.11.2020	4932
8295	3927	49	2	1245891.0	2682873.0	30.11.2020	3370
8296	3927	50	2	1245891.0	2682873.0	07.12.2020	3191
8297	3927	51	2	1245891.0	2682873.0	14.12.2020	657

8298 rows × 7 columns

write geojson file

In [98]:

```
#output geojson file

proctable=fulldata.copy()

zuri = gpd.GeoDataFrame(proctable, geometry=gpd.points_from_xy(proctable.lat, proctable.lon))


zuri = zuri.set_crs(epsg=2056, allow_override=True)
zuri = zuri.to_crs(epsg=4326)

zuri=zuri.drop(columns=["lon","lat"])
display(zuri.crs)
display(zuri)

zuri.to_file(os.path.join(outdir,'zuri.geojson'), driver='GeoJSON')
```

```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

	loc	wk	kind	dtstr	n	geometry
0	2	1	1	01.01.2020	5288	POINT (8.48750 47.39099)
1	2	2	1	06.01.2020	13469	POINT (8.48750 47.39099)
2	2	3	1	13.01.2020	14155	POINT (8.48750 47.39099)
3	2	4	1	20.01.2020	13787	POINT (8.48750 47.39099)
4	2	5	1	27.01.2020	13789	POINT (8.48750 47.39099)
...	...	...	...	...	...	...
8293	3927	47	2	16.11.2020	5481	POINT (8.53562 47.35866)
8294	3927	48	2	23.11.2020	4932	POINT (8.53562 47.35866)
8295	3927	49	2	30.11.2020	3370	POINT (8.53562 47.35866)

	loc	wk	kind	dtstr	n	geometry
8296	3927	50	2	07.12.2020	3191	POINT (8.53562 47.35866)
8297	3927	51	2	14.12.2020	657	POINT (8.53562 47.35866)

8298 rows × 6 columns

Anhang 2: HTML-Code Webseite

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Mobilität ZH COVID19</title>

<!--- libraries -->
  <script src='https://api.mapbox.com/mapbox-gl-js/v1.11.0/mapbox-gl.js'></script>
  <link href='https://api.mapbox.com/mapbox-gl-js/v1.11.0/mapbox-gl.css' rel='stylesheet' />
  <link rel="stylesheet" type="text/css" href="https://bountan.github.io/mobilitycovid/styles/styles-covid.css">
</head>
<body>

<!--- banner-->
  <div class="bannerimage">
    <div class="bannertext">
      <h1>Mobilität in Zürich während COVID-19  </h1>
    </div>
  </div>

<!-- introtext -->
<div id='headtext'>
Diese Karte stellt die Frequenz von Fussgängern, Velofahrern und motorisierten Fahrzeugen an verschiedenen automatischen Zählstationen in der Stadt Zürich im Jahr 2020
dar. Sie wurden in verschiedenen Intervallen gezählt, zum Zweck dieser Darstellung aber auf ganze Wochenzahlen
aufsummiert. Ziel ist ein Überblick über die Frequenz des Individualverkehrs in Hinblick auf das Verhalten während verchiedener Massnahmen zur Eindämmung der COVID-19-Pandemie.
 Die Darstellung erlaubt die Auswahl von Kalenderwochen und Verkehrsmodus und informiert jeweils kurz über den aktuellen Massnahmenkatalog in der Stadt Zürich.

<br><br>Die Werte basieren auf automatisierten Zählungen durch die Stadt Zürich, die von ebenjener auf statistische Fehlerquellen hin geprüft und korrigiert wurden. Es besteht kein Anspruch auf Vollständigkeit oder Gültigkeit der publizierten Daten seitens des Autors dieser Darstellung.
  <br><br>Datenherkunft: Open Data Stadt Zürich <a href='https://data.stadt-zuerich.ch/dataset/sid_dav_verkehrszaehlung_miv_od2031/resource/44607195-a2ad-4f9b-b6f1-d26c003d85a2' target="_blank">Verkehrszählung MIV</a>, <a href='https://data.stadt-zuerich.ch/dataset/ted_taz_verkehrszaehlungen_werte_fussgaenger_velo/resource/b9308f85-9066-4f5b-8eab-344c790a6982' target="_blank">Verkehrszählung Fussgänger und Veloverkehr</a>
  <br>Kontakt Autor: Ueli Isenschmid   <a href='mailto:ueli.isenschmid@gmail.com'>Email</a>, <a href='https://www.linkedin.com/in/ueli-isenschmid/' target="_blank">Linkedin</a>

  </div>

<!--- map object-->

<div id='map'></div>

<!-- console controls -->
<div id='console'>

<!-- legend -->
  <div class='session'>
    <h2>Total Zählungen</h2>
    <div class='row colors'></div>

    <div class='row' >
    <div class='label vol'>   &nbsp0 -<br>200</div>
    <div class='label vol'> &nbsp200 -<br>1000</div>
    <div class='label vol'> &nbsp1000 -<br>10000</div>
    <div class='label vol'> &nbsp10000 -<br>75000</div>
    <div class='label vol'>75000+</div>
  </div>



<!-- slider modetime -->
    <div class='session' id='sliderbar'><br>
<h2 class = 'paddingtop'>Kalenderwoche: <label id='active-week' >1</label></h2>
<h2 class = 'paddingtop'><label id='lockdownstat' > </label></h2>

  <input id='slider' class='row' type='range' min='1' max='52' step='1' value='1' />
</div>
  </div>

<!-- buttons weekmode kind -->
  <div class='session'>
    <h2>Verkehrsmodus</h2>
    <div class='row' id='filters'>
      <input id='car' type='radio' name='toggle' value=3>
      <label for='car'>Motorisierter Verkehr</label><br>
      <input id='ped' type='radio' name='toggle' value=1>
      <label for='ped'>Fussgänger</label><br>
      <input id='bike' type='radio' name='toggle' value=2>
      <label for='bike'>Velo</label>
    </div>
  </div>

</div>

<!-- map controls -->
<script>

// basemap from mapbox
    mapboxgl.accessToken = 'pk.eyJ1IjoiYm91bnRhbiIsImEiOiJpSXVpaU1BIn0.pzuLQeLdLb3bRoYeQNc9jA';
```

```
    var map = new mapboxgl.Map({
      container: 'map',
      center: [8.539261, 47.376166],
      zoom: 13,
      minZoom: 10,
      maxZoom: 16,
      style: 'mapbox://styles/bountan/ckc9d897z432e1imtuzvbx928',
      maxBounds: [[8.236389, 47.300210], [8.696031, 47.461794]]
    });

// map controls
    var scale = new mapboxgl.ScaleControl({
    maxWidth: 200,
    unit: 'metric'
    });
    map.addControl(scale);

    map.addControl(new mapboxgl.FullscreenControl());

// load map
map.on('load', function() {

// variables filter what to display
  var filterWeek = ['==', ['get', 'wk'], 1];
  var filterMode = ['==', ['get', 'kind'], 1];
  var modetext = "Fussgänger"

// add sections of traffic layer
  map.addLayer

//listener for slider change
  document.getElementById('slider').addEventListener('input', function(e) {
  var week = parseInt(e.target.value);



  filterWeek = ["==", ['get', 'wk'], week]

  map.setFilter('location', ['all', filterWeek, filterMode]
  );

  // update text in the UI
  document.getElementById('active-week').innerText = week;


  if (week < 12){
      lockdown = " "
    } else if (week < 20) {
      lockdown = "Lockdown 16. März"
    } else if (week < 28) {
      lockdown = "Aufgehoben 11. Mai"
    } else if (week < 43) {
      lockdown = "Maskenpflicht 6. Juli"
    } else {
      lockdown = "Zweite Welle 19. Oktober"
    }

document.getElementById('lockdownstat').innerText = lockdown;
});


// listener for button change

document.getElementById('filters').addEventListener('change', function(e) {
  var mode = parseInt(e.target.value);
  if (mode == 1){
    modetext = "Fussgänger"
  } else if (mode == 2) {
    modetext = "Velofahrer"
  } else if (mode == 3){
    modetext = "motorisierte Fahrzeuge"
  }
  else{
    modetext = ""
  }




  filterMode = ["==", ['get', 'kind'], mode]
  map.setFilter('location', ['all', filterWeek, filterMode]);
});

//layer location, pretty static layer
map.addLayer({
  id: 'location',
  type: 'circle',
  source: {
    type: 'geojson',
    data: 'https://bountan.github.io/mobilitycovid/data/zuri.geojson'
    },
```

```
  paint: {
    'circle-color': [
    'step',
    ['get', 'n'],
    '#2d8ac0', 200,
    '#559783', 1000,
    '#add8a4', 5000,
    '#e4ea9a', 10000,
    '#fae08a', 15000,
    '#f9ae60', 40000,
    '#f26b44', 75000,
    '#d83d51'
    ],

    'circle-radius': [
      'interpolate', ['linear'], ['zoom'],
        11, ['/',   ['step',
            ['get', 'n'],

            2, 200,
            4, 1000,
            6, 5000,
            10, 10000,
            12, 15000,
            16, 40000,
            20, 75000,
            25
          ], 2],
        13, ['/', ['step',
            ['get', 'n'],

            2, 200,
            4, 1000,
            6, 5000,
            10, 10000,
            12, 15000,
            16, 40000,
            20, 75000,
            25
          ], 0.7],
          15, ['/',   ['step',
              ['get', 'n'],
              2, 200,
              4, 1000,
              6, 5000,
              10, 10000,
              12, 15000,
              16, 40000,
              20, 75000,
              25
            ], 0.3],
      ],

}
});
// popup stops
map.on('click', 'location', function(e) {
new mapboxgl.Popup()
.setLngLat(e.lngLat)
.setHTML("Total "+modetext+" in Kalenderwoche "+e.features[0].properties.wk+": <b>"+e.features[0].properties.n+"</b>")
.addTo(map);
});
// change cursor on hover sections
// change cursor on hover stops
map.on('mouseenter', 'location', function() {
map.getCanvas().style.cursor = 'pointer';
});

// change cursor back stops
map.on('mouseleave', 'location', function() {
map.getCanvas().style.cursor = '';
});
});

</script>

</body>

</html>
```