The impact of roadside construction on freight transport within Germany

This study examines how roadside construction activities influence the choice between rail and road for freight transport. We hope to reveal a significant shift towards rail transport during intensive roadside construction periods, with a notable return to road transport post-construction.

Motivation:

With the ascend of just-in-time production and efficient warehouses, the punctual transportation of goods has become a matter of ever greater economic importance. Besides airways or shipments, which are largely responsible for global distribution, local transportation mainly occurs in two ways: By train or by truck. In the case of Germany particularly, via the Autobahn or the German railway system. Both systems supplement each other, trains leading goods to central distribution networks within the country, where they are then picked up and driven to an individual location via the German motorway. But next to said supplementation, a significant substitution effect, contingent on route-specific travel time and punctuality, can also be assumed. What happens to a specific train connection, if the substituting autobahn route is beset by construction and repair works, and thus becomes unreliable and congested, for instance? Do actors react and place more goods from the truck onto the train?

The transportation of goods via railways and the state of autobahns are pivotal elements of a nation's logistics and economic vitality. Understanding how these aspects interplay, particularly concerning the repercussions of autobahn repair works on railway traffic for goods, can help actors make better decisions in the future.

Methods:

Data Sources

The following data sources are used for this project:

Datasource1: Beförderte Gütermenge und Beförderungsleistung (Straßengüterverkehr): Deutschland, Jahre, Verkehrswege.

* Metadata URL: https://mobilithek.info/offers/-2366581057346576431

* Data URL: https://www-genesis.destatis.de/genesis/downloads/00/tables/46231-0001_00.csv

* Data Type: CSV

This DataSource holds information about the freight transport (on road) within Germany).

Datasource2: Beförderte Güter, Beförderungsleistung(Eisenbahngüterverkehr): Deutschland, Jahre

* URL: Metadata https://mobilithek.info/offers/-2816139595278720568

* Data URL: https://www-genesis.destatis.de/genesis/downloads/00/tables/46131-0001_00.csv

* Data Type: XML

This DataSource holds information about the freight transport (on rail) within Germany).

Datasource3: Aggregierte Arbeitsstellen längerer Dauer 2020 bis 2022

* Metadata URL: https://mobilithek.info/offers/520200225602543616

* Data URL: https://maps.infoware.de/opendata/roadworks.geojson

* Data Type: JSON

This DataSource holds information about roadside construction in Germany.

Installation of requirements

All requirements are installed via the requirements.txt file, which looks as follows:

```
attrs==22.2.0
greenlet==2.0.2
iniconfig==2.0.0
numpy==1.24.2
packaging==23.0
pandas==1.5.3
pluggy==1.0.0
python-dateutil==2.8.2
pytz==2022.7.1
six==1.16.0
SQLAlchemy==1.4.46
typing_extensions==4.5.0
geopandas==0.14.2
matplotlib==3.8.2
scikit-learn==1.3.2
statsmodels==0.14.1
```

Figure 1 - requirements.txt

Loading Data

The Data is loaded via the pipeline.py script. This script consumes all three datapoints and inserts them into a *sqlite* database. Additionally, it executes some data cleaning to get rid of corrupted data and ensure the right format.

```python
import pandas as pd
import geopandas as gpd
from shapely import wkt
import csv
import os
from sqlalchemy import create_engine

#set data target
script_dir = os.path.dirname(__file__)
parent_dir = os.path.dirname(script_dir)
folder_path = os.path.join(parent_dir, "data")
data_file = "projdata.sqlite"
full_path = os.path.join(folder_path, data_file)


# Declare dataset urls
datasource1 = "https://www-genesis.destatis.de/genesis/downloads/00/tables/46231-0001_00.csv"
datasource2 = "https://www-genesis.destatis.de/genesis/downloads/00/tables/46131-0001_00.csv"
datasource3 = "https://maps.infoware.de/opendata/roadworks.geojson"

# Create SQLLite DB
engine = create_engine(f"sqlite:///{full_path}")

#read in datasource1
column_names = ['Year','Gewerblicher Verkehr und Werkverkehr insgesamt Beförderte Gütermenge','Gewerblicher Verkehr und Werkverkehr insgesamt Beförderungsleistung','Binnenverkehr Beför
try:
    df = pd.read_csv(datasource1, sep=";", encoding='ISO-8859-1', on_bad_lines='skip', engine='python', header=None, names=column_names, skiprows=9,skipfooter=4)
except UnicodeDecodeError:
    df = pd.read_csv(datasource1, sep=";", encoding='cp1252', on_bad_lines='skip', engine='python', header=None, names=column_names, skiprows=9, skipfooter=4)
df.to_sql(name="straße", con=engine, if_exists="replace", index=False)

#read in datasource2
column_names = ['Year','Beförderte Güter','Veränderung zum Vorjahr BefGü','Beförderungsleistung','Veränderung zum Vorjahr BefLst']
try:
    df = pd.read_csv(datasource2, sep=";", encoding='ISO-8859-1', on_bad_lines='skip', engine='python', header=None, names=column_names, skiprows=7, skipfooter=3)
except UnicodeDecodeError:
    df = pd.read_csv(datasource2, sep=";", encoding='cp1252', on_bad_lines='skip', engine='python', header=None, names=column_names, skiprows=7, skipfooter=3,)

df.to_sql(name="eisenbahn", con=engine, if_exists="replace", index=False)

#read in datasource3
gdf = gpd.read_file(datasource3)
gdf.drop('geometry', axis=1, inplace=True)
gdf.to_sql(name="baustellen", con=engine, if_exists="replace", index=False)
```

Figure 2 - pipeline.py

This tool some effort to make this run everywhere and independently, as when integrating GitHub Actions, multiple errors arose (which is due to the "./main" path when executing on GitHub). To solve this a combination of the folder path and tools from the os library were used to ensure a truly dynamic pathing when referencing from the projects folder to the data folder.

Additionally, to ensure that csvs' are getting read in correctly a try/catch construct is used to ensure some independency from the encoding.

Finally, as sqlite doesn't comply with the geometry datatype of geopandas, the choice was made to remove the geometry column from the roadside construction dataset. This was due to the fact, that for this analysis the exact coordinates aren't important but it's just about the number of constructions taking place.

Results:

Data:

For each of the three data sources a quick sneak-peak is shown below. This is achieved using the df.head() function of the python framework pandas.

Datasource 1:

	Year	Gewerblicher Verkehr und Werkverkehr insgesamt Beförderte Gütermenge	...	Kabotage Beförderte Gütermenge	Kabotage Beförderungsleistung
0	1995	3169559	...	-	-
1	1996	3014946	...	-	845
2	1997	2980981	...	/	718
3	1998	2968023	...	/	1016
4	1999	3181363	...	6756	1533

Figure 3 - Datasource 1

Datasource 2:

	Year	Beförderte Güter	Veränderung zum Vorjahr BefGü	Beförderungsleistung	Veränderung zum Vorjahr BefLst
0	2005	315575665	.	95420493145	.
1	2006	344269654	+9,1	107007169751	+12,1
2	2007	359299060	+4,4	114614826747	+7,1
3	2008	369524594	+2,8	115651864816	+0,9
4	2009	311013540	-15,8	95834460393	-17,1

Figure 4 - Datasource 2

Datasource 3:

	Year	Gewerblicher Verkehr und Werkverkehr insgesamt Beförderte Gütermenge	...	Kabotage Beförderte Gütermenge	Kabotage Beförderungsleistung
0	1995	3169559	...	-	-
1	1996	3014946	...	-	845
2	1997	2980981	...	/	718
3	1998	2968023	...	/	1016
4	1999	3181363	...	6756	1533

Figure 5 - Datasource 3

Analysis:

To conduct an analysis the three data sources are combined and a new table (years_df) is generated, which aggregates interesting information from the sources by years to be able to analyze.

This is achieved like shown below:

```
min_year = df_bau['startDate'].dt.year.min()
max_year = df_bau['endDate'].dt.year.max()
years = range(min_year, max_year)
years_df = pd.DataFrame(years, columns=['Year'])

df_bau['startDate'] = pd.to_datetime(df_bau['startDate'])
df_bau['endDate'] = pd.to_datetime(df_bau['endDate'])

def count_events(year):
    return ((df_bau['startDate'].dt.year <= year) & (df_bau['endDate'].dt.year >= year)).sum()

years_df['Event_Count'] = years_df['Year'].apply(count_events)
years_df = years_df.rename(columns={'Event_Count': 'Baustellen'})


years_df = years_df.merge(df_eis[['Year', 'Beförderte Güter']], on='Year', how='left')
years_df = years_df.rename(columns={'Beförderte Güter': 'Beförderte Güter Eisenbahn'})
years_df['Beförderte Güter Eisenbahn'] = round(years_df['Beförderte Güter Eisenbahn'] / 1000)

years_df = years_df.merge(df_str[['Year', 'Gewerblicher Verkehr und Werkverkehr insgesamt Beförderte Gütermenge']], on='Year', how='left')
years_df = years_df.rename(columns={'Gewerblicher Verkehr und Werkverkehr insgesamt Beförderte Gütermenge': 'Beförderte Güter Straße'})

years_df.to_sql(name="baustellen_per_year", con=engine, if_exists="replace", index=False)
```

Figure 6 - Creation of the analytics dataframe (years_df)

First, the boundaries of years are calculated based on the data from roadside constructions. Afterward, the amount of roadside construction within each year is calculated (as each construction has a start and end date) and is added as a separate column to the years_df. Finally, the values from rail and street data are merged into the table and adjusted to be ready for analysis.

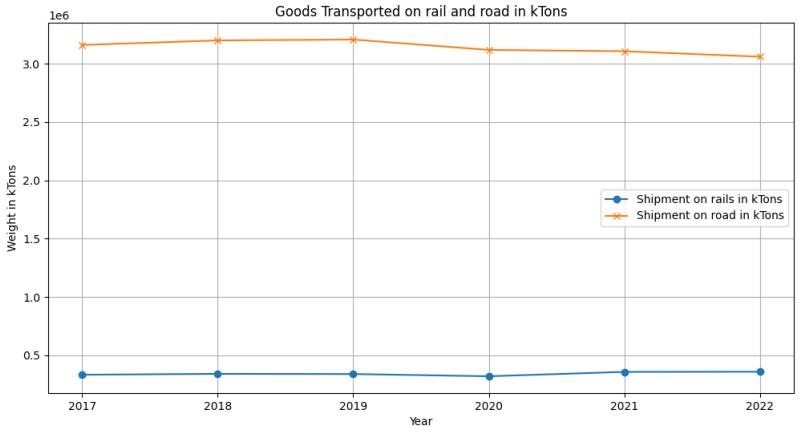

Figure 7 - Transportation by rail and road

The picture above shows a chart of the transportation data over the years by transportation vehicle.

Afterwards further statistical tools (scikit-learn and `statsmodels`) are used to conduct statistical research within the data.

Then the effect of roadside construction on the goods transported via train are investigated. This is achieved via the following code:

```
#regression analysis on trains
X = years_df[['Baustellen']]
y = years_df['Beförderte Güter Eisenbahn']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"Coefficients: {model.coef_}")
print(f"Intercept: {model.intercept_}")

X = sm.add_constant(X)
model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

Figure 8 - Statistical Analysis code for rail transport

The results of this analysis are shown below:

```
Dep. Variable:     Beförderte Güter Eisenbahn   R-squared:                       0.401
Model:                                    OLS   Adj. R-squared:                  0.251
Method:                         Least Squares   F-statistic:                     2.675
Date:                        Wed, 10 Jan 2024   Prob (F-statistic):              0.177
Time:                                14:40:19   Log-Likelihood:                -64.061
No. Observations:                           6   AIC:                             132.1
Df Residuals:                               4   BIC:                             131.7
Df Model:                                   1
Covariance Type:                    nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       3.367e+05   6003.752     56.084      0.000     3.2e+05    3.53e+05
Baustellen     3.7136      2.271      1.636      0.177      -2.591      10.018
==============================================================================
```

Figure 9 - Statistical Analysis results for rail transport

Afterwards the effect of roadside construction on the goods transported on road are also investigated. This is done via a similar code as before, looking like this:

```
#regression analysis on streets
X = years_df[['Baustellen']]
y = years_df['Beförderte Güter Straße']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"Coefficients: {model.coef_}")
print(f"Intercept: {model.intercept_}")

X = sm.add_constant(X)
model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

Figure 10 - Statistical Analysis code for road transport

The results of the second analysis can be seen below:

```
Dep. Variable:     Beförderte Güter Straße   R-squared:                       0.578
Model:                                 OLS   Adj. R-squared:                  0.472
Method:                      Least Squares   F-statistic:                     5.470
Date:                     Wed, 10 Jan 2024   Prob (F-statistic):             0.0795
Time:                             14:43:39   Log-Likelihood:                -71.126
No. Observations:                        6   AIC:                             146.3
Df Residuals:                            4   BIC:                             145.8
Df Model:                                1
Covariance Type:                 nonrobust
================================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const       3.165e+06   1.95e+04    162.413      0.000    3.11e+06    3.22e+06
Baustellen   -17.2402      7.371     -2.339      0.079     -37.706       3.225
================================================================================
```

Figure 11 - Statistical Analysis results for road transport

Discussion/Conclusion:

When taking the results into account, the thesis of this project unfortunately cannot be confirmed to a satisfying degree of certainty. While the regression analyzing the relationship between roadwork and train usage showed a positive coefficient, the results cannot be said to be statistically relevant at the five- or 10-percent level (p=0.177). This could have multiple reasons. For one, the dataset lacks significant control variables, like overall transportation volume or train capacities and delays. Additionally, upon closer inspection it becomes evident that the roadside construction data itself is not sufficient to satisfy the needs for this analysis, either. The dataset seems to be too shallow, lacking the quantitative mass required for a representative study. It also only observes long-term construction works, disregarding short-term blockages or quick repairs.

Nevertheless, it seems solid enough to at least showcase the proposed negative relationship between road transportation of goods and construction work. With a coefficient of –17.24 and p=0.079 it confirms that actors react to roadblocks within their transportation systems, relying less on trucks when the amount of construction work rises. Since it seems logical to assume a direct substitutionary relationship between Autobahn transportation and train transportation (as Autobahns also only connect the big traffic arteries to each other), we can thus tentatively confirm our hypothesis, whilst remaining mindful methodological shortcomings.

To solve the previously stated issues, the roadside dataset could be replaced by a more potent data source of higher quality (and especially quantity). This should allow us to craft a meaningful analysis. Another improvement could be the inclusion of locale data (for example federal state). This would require acquiring transportation data for all (or a couple) of the federal states of Germany and conducting a more detailed analysis. Additionally, and to contextualize the analysis, macro trends of the freight transportation industry should be considered (to have a baseline for the analysis of changes).