A statistical approach to predicting a tsunami's wave height

Can the height of a tsunami's wave be predicted based on the distance from and the magnitude of a preceding earthquake?

Word count: TBD

Contents

1 Introduction

1.1 Background

Naturally occurring disasters such as tsunamis are devastating for people and resources residing in risk areas. These tsunamis are caused by many different types of natural phenomena such as landslides, collapse of seamount, or more rarely, the impact of a meteorite. This paper however will focus on one of the more common cases of tsunami; one caused by the earthquakes. This is due to earthquakes causing around 72% of earthquakes [1], and as the time between an earthquake event and its tsunami is relatively high compared to events such as landslides [2], leaving adequate time for prediction and correct preparation of the events properties, such as the wave height, as discussed in this paper.

1.2 Personal Engagement

I have chosen to write about this subject as I have recently become interested by the statistics and its implications in real-life matters. This research has proved very interesting for me as it has abstracted the field of statistics for me. Furthermore, as I enjoy programming, the research I have done has helped me learn about the tools and technologies used by data analysts and other professionals of the matter.

2 Statistical Model

2.1 Processing the dataset for use

The dataset, procured from the NOAA website [3], includes many tsunami runups, going back before the year 0. Due to this, the NOAA states uncertainties that can occur in its data. For example, reports that were recorded before the 20th century tend to be based on written accounts, and not recorded seismic activity. Furthermore, the installation of a system called the Worldwide Standardized Seismograph Network in the 1960s allowed for more accurate data and for that reason any recorded activity before this date will be discarded from the dataset. The dataset itself can be accessed through two different methods; searching the database using the online tool or downloading it in a somewhat obscure Google Earth format. The online tool supports the download of search results in a much more human-readable `.csv` format, so I search for all entries after the year 1900, and downloaded the result.

The data is cleaned of all "dubious" reports, reports before the year 1960 (see above), and of all irrelevant fields such as "Injuries" or "Damage $Mil". This preliminary data cleanup is done through the use of the pandas Python library [4][5],

The data must be then correctly split into a training and an test dataset. The training dataset will be used to generate a regression model while the test dataset will be used to calculate the general accuracy of the model, by predicting values and comparing those to the known value as found in the dataset. The dataset is split based on a decimal number using the `df.sample` method found in the pandas library. A split of 70% for training and 30% for test has been chosen. A `random_state` argument is set to an arbitrary value 42, as if it weren't, the dataset would be split in a different manner at each execution of the program.

2.2 Regression

Regression analysis is a very large portion of statistics, and regression models can be found in numerous use cases of the field. Regression models consist of estimating the relationship between one (or more) independent variables and a dependent variable. A simple example of regression can be seen in the linear relationship between an independent variable x_i, and the dependent variable y. The following expression shows this relationship in what is called linear regression, as it models a straight line.

$y = a + \beta x + e$

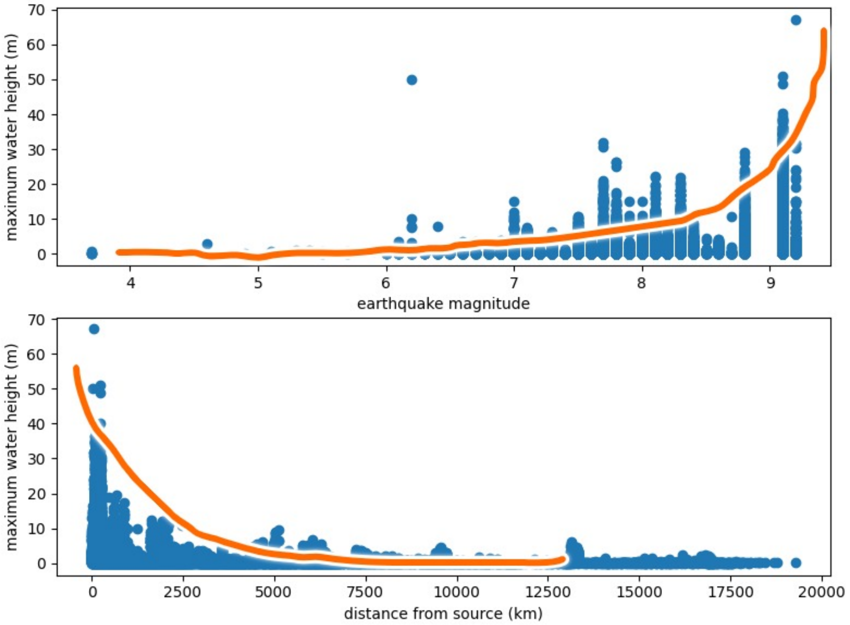
Where the term a represents the y-intercept of the line and β represents the coefficient of x_i. The final term e is what is known as the error or residual term, it is an estimate of the amount by which the predictions differ from the real value.

However, the relationship between the distance from an event and the wave height is not necessarily linear, especially if it is considered that the travel of sound in a fluid obeys the inverse square law.

This already tells us a simple linear regression model cannot be used for this research. If we want a second degree polynomial regression formula, a new x^2 term can be added to the previous formula and we get:

$y = a + \beta_1 x + \beta_2 x^2 + e$

Even though the right side of this equation is quadratic in x, this is still linear regression as the right side is linear in β (if we were to let $x = 1$, we would be left with $y = a + \beta_1 + \beta_2 + e$, which is linear). A second degree polynomial regression would best fit our model for both independent variables, as seen in the graphs below, where a theoretical regression line is drawn by hand:



In addition, our dataset has two independent variables: the distance from a particular earthquake event and the magnitude of that earthquake.

. . .

So the relationship between the two independent variables and the wave height can be estimated with a multivariate, polynomial regression model. We will use the Python scikit-learn library to aid in creating this model.

. . .

The model's accuracy is then calculated, with the mentioned before "test" dataset. The `accuracy_score` method from the `sklearn.metrics` class is used for this task.

. . .

2.3 Ease of access

For an easy access to the predictions generated by the program, I implemented a simple function to fetch new earthquake events from the https://earthquake.usgs.gov/earthquakes/feed/v1.0/geojson.php live earthquake feed. These earthquake events are quantified into a magnitude `mag` and a distance `dist` variable, which are supplied to the statistical model. This information is then easily accessible from a human-readable report exported to a plaintext file. Following is an example report generated by the tool:

3 Uses of this research

The research included in this paper could be useful for many life-saving scenarios, notably ones in which, for example, high quality and large-scale tsunami alert systems are not available.

4 Conclusion

4.1 Evaluation of the method used

This method, like any other statistical approximation, is by no means perfect and for that reason should not be used in any real-world application. The results procured by the program are subject to many uncertainties, one of which is described below.

When attempting to predict the maximum wave height of a tsunami such as the one caused by the 2011 earthquake off the Pacific coast of Tōhoku, approximately 70km away from shore, which produced a 38.9m wave [6]. The model in this paper approximates this same event as only a 8m wave, when supplied the 9.0 magnitude of the earthquake and the 70km distance. These types of errors can be attributed to the fact that there are many more independent variables, such as the shape of the seafloor for example, that can amplify or potentially change the height of a tsunami's wave.

Works Cited

[1] *What Causes a Tsunami?* URL: https://tsunami.org/what-causes-a-tsunami/.

[2] Langford P Sue, Roger I Nokes, and Roy A Walters. *Modelling of Tsunami Generated By Underwater Landslides*, p. 5. URL: https://www.eqc.govt.nz/sites/public_files/1596-Modelling-tsunami-generated-by-underwater-landslides.pdf.

[3] National Geophysical Data Center / World Data Service. *NCEIWDS Global Historical Tsunami Database*. https://dx.doi.org/10.7289/V5PN93H7. DOI: 10.7289/V5PN93H7.

[4] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[5] Wes McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[6] Akio Okayasu. *Tsunami of the great earthquake, 38.9m in Miyako ... surpasses Meiji Sanriku*. Apr. 2011. URL: https://web.archive.org/web/20110418144715/http://www.yomiuri.co.jp/science/news/20110415-OYT1T00389.htm.

A Appendix

A.1 Python program

Listed here is the Python program through which the dataset was cleaned and predictions based on the statistical model were made. Run using 64-bit Python version 3.9.1.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score

def filter_time(df, threshold):
    filter_df = df.loc[df.year > threshold].copy()
    return filter_df



def filter_doubtful(df):
    filter_df = df.loc[df['doubtful␣runup'] != 'y'].copy()
    return filter_df



def get_droppable(df, whitelist):
    to_drop = []
    for (columnName, columnData) in df.iteritems():
        if columnName not in whitelist:
            to_drop.append(columnName)
    return to_drop



if __name__ == "__main__":
    df = pd.read_table("dataset/runups.tsv", sep='\t')
    df.columns = df.columns.str.lower()

    # Filter data points before installation of WWSSN
```

```
filter_df = filter_time(df, threshold=1960)

# Doubtful runups to be removed
filter_df = filter_doubtful(filter_df)

# Drop all columns except the ones needed for X1, X2 and the Y variables.
filter_df = filter_df.drop(
    get_droppable(filter_df,
                  ['earthquake␣magnitude',
                   'distance␣from␣source␣(km)',
                   'maximum␣water␣height␣(m)']), axis=1)

# Drop all rows with a NaN value
filter_df = filter_df.dropna(how='any')

# Dataset split:
# The dataset is split into two parts, a "training" df_train dataset and
# an "test" df_eval dataset. The "training" will be used for the
# generation of a regression model and the "test" will
# be used for the testing of this model, to calculate its accuracy.
df_train, df_test = train_test_split(
    filter_df, test_size=0.30, random_state=42)

# For the train dataset,
# Set X to the independent variables (equivalent to dropping the dependent)
# Set y to our output column: maximum water height
X_train = df_train.drop('maximum␣water␣height␣(m)', 1)
y_train = df_train['maximum␣water␣height␣(m)']
```

```
# The test dataset, idem to the train dataset
X_test = df_test.drop('maximum␣water␣height␣(m)', 1)
y_test = df_test['maximum␣water␣height␣(m)']

# Fit the values to
poly = PolynomialFeatures(degree=2)
X_ = poly.fit_transform(X_train)
predict_ = poly.fit_transform(X_test)

# Predict
mlr_model = LinearRegression()
mlr_model.fit(X_, y_train)
y_pred = mlr_model.predict(predict_)

print(y_test, y_pred)
print(mlr_model.score(predict_, y_test))

# Print the coefficients and intercept
theta0 = mlr_model.intercept_
print("Intercept:␣", theta0, "Coefficients:␣", mlr_model.coef_)

# Code for generating 3d plot with prediction scatter
"""
fig = plt.figure()
plt.clf()
ax = Axes3D(fig)
ax = fig.gca(projection='3d')

ax.scatter(X_test['distance from source (km)'],
           X_test['earthquake magnitude'], y_test)

ax.scatter(X_test['distance from source (km)'],
           X_test['earthquake magnitude'], y_pred)

plt.show()
"""

#TODO: add GeoJSON report thingy https://earthquake.usgs.gov/earthquakes/feed/
print(mlr_model.predict(poly.fit_transform([[4.3, 41]])))
```