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?

Väinö-verneri Kauppila

February 2021

Word count: TBD

Note: diagrams reserve a large amount of space, and cannot be placed in the appendix for the sake of clarity

Contents

1 Introduction

1.1 Background

Naturally occurring disasters, such as tsunamis, are devastating for people, human settlements, buildings and other resources residing in risk areas. 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, focuses on one of the more common causes of tsunamis; earthquakes. Earthquakes cause around 72% of tsunamis [1]. The time between an earthquake and its tsunami is relatively long compared to events such as landslides [2], leaving adequate time to prepare for impact. Predicting wave height, as discussed in this paper, can further help to assess the seriousness of the tsunami and provide help to prepare for the event.

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 made the field of statistics more concrete, because it provides a real world use case. Furthermore, as I enjoy programming, the research I have done has helped me learn about the tools and techniques used by data analysts and other professionals.

2 Statistical Model

2.1 Processing the dataset for use

The dataset, procured from the National Oceanic and Atmospheric Administration 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. Only since the 1960s, the installation of the Worldwide Standardized Seismograph Network system allowed for more accurate data in this paper. For that reason, any recorded activity before this date was discarded from the dataset for this paper. The dataset itself can be accessed through two different methods; searching the database using the online tool or downloading it in a somewhat obscure compressed Google Earth `.kmz` format. The online tool supports the download of search results in a much more human-readable `.tsv` (tab-separated) format. For this paper, all entries after the year 1900 were searched and downloaded.

The data is cleaned of data points marked as "dubious" (there is an column for this parameter), 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], which is commonly used for the handling of large datasets for its efficiency.

The data was split into a training and a test dataset. The training dataset was used to generate a regression model while the test dataset was 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 was split based on a fraction using the `df.sample` method found in the pandas library. A split of 70% for training and 30% for test was 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. The program can be found in Appendix A.2.

2.2 Regression

2.2.1 Linear model

Regression analysis is a commonly used tool in statistics, and regression models can be found in numerous use cases. 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 that can be used for prediction of values:

$$y = \beta_0 a + \beta_1 x + e,$$

where the term β_0 represents the y-intercept of the line and β represents the coefficient of x. 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. An example of linear regression can be found in Appendix A.1, where a set of data points is presented on a scatter plot, and the regression line is found by finding the individual coefficients β_n of x (see equation above), like such:

$\hat{\beta}_1 = \frac{\sum(x_i-\bar{x})(y_i-\bar{y})}{\sum(x_i-\bar{x})^2}$, where $\hat{\beta}_n$ represents an estimated value and $\bar{x}$ represents the mean of the set x and the $\hat{\beta}_0$ value can be approximated as $\hat{\beta}_0 = \bar{y} - \hat{\beta}_1\bar{x}$

For the dataset in A.1, the results to these equations are:

$\hat{\beta}_1 = \frac{1806757.14}{322280.86} = 5.60$ and $\hat{\beta}_0 = 2257.14 - 5.61 \cdot 512.86 = -618.01$

So the regression line can be expressed as $\hat{y} = 5.60x - 618.01$

2.2.2 Special cases

However, the relationship between the distance between an earthquake and its tsunami and the wave height is not necessarily linear, especially if it is considered that the travel of sound (which is a wave, or the displacement of a fluid that can travel) in a fluid obeys the inverse square law. Therefore, a simple linear regression model cannot be used for this research. A second degree polynomial regression formula would be more appropriate, and a new x^2 term can be added to the previous formula. We get:

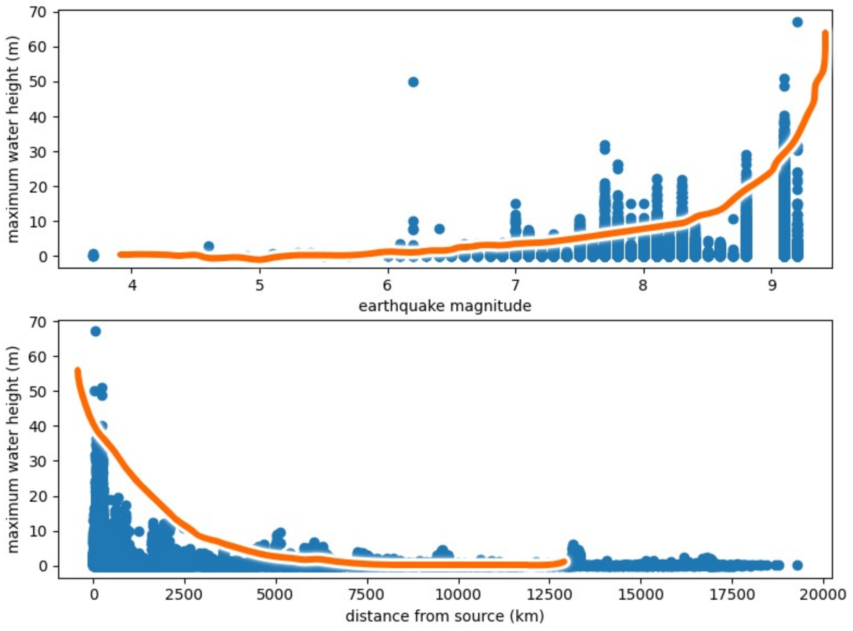
$y = a + \beta_1 x + \beta_2 x^2 + e$

Even though the right side of this equation is quadratic in x, this is still linear regression as the right side is linear in the β terms (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 curve of regression is drawn by hand:

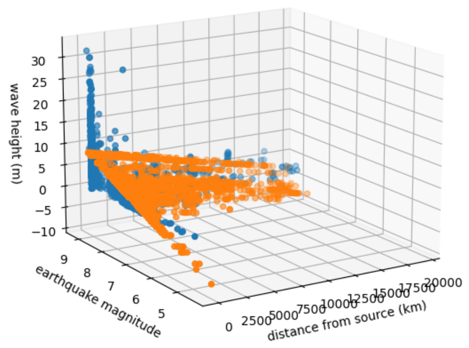
In addition, our dataset has two independent variables: the distance from a particular earthquake event and the magnitude of that earthquake. Models such as the one explored in this paper, where multiple independent variables are present, are called multivariate and can be modeled by adding the terms of a new independent variable, as such for a linear model with two independent variables:

$a_1 + \beta_1 x_1 + a_2 + \beta_2 x_2 + e,$

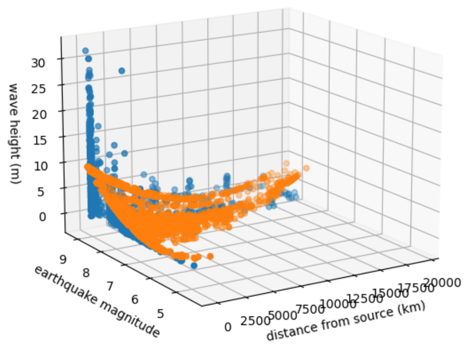
where the terms a_n are the y-intercepts of the nth independent variable,



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.



(a) Linear model



(b) Polynomial model

Figure 1: 2 multivariate (hence the 3 dimensions) regression models

. . .

The model's fit is then calculated, with the test dataset. The `accuracy_score` method from the `sklearn.metrics` class was used for this task. This function calculates what is called the R^2 value, which explains the extent at which the variance (the amount by which numbers are spread from the average value) in the dependent variable can be explained by the independent variable(s). An R^2 value of 0.6 indicates that 60% of the model's results match those of the dataset. This value is defined as:

$R^2 = 1 - \frac{SS_{res}}{SS_{total}}$,

where SS_{res} is the sum of the error terms, or the residual sum of squares, in a dataset of n values $y_1, \ldots, y_i$, with their corresponding predicted values $p_1, \ldots, p_i$ defined by

$SS_{res} = \sum_i e_i^2 = \sum_i (y_i - p_i)^2$, [1] where e_i is the error term, equal to the difference of a data point y_i and its corresponding predicted value p_i.

and SS_{tot} is the total sum of squares, which is proportional to the variance in the data, defined as:

$SS_{tot} = \sum_i (y_i - \bar{y})^2$, where $\bar{y}$ is the mean of the data $y_1, \ldots, y_i$, or $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$

3 Conclusion

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.

3.1 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

[1] $\sum_{i=1}^{n} x_i$ means that all values of x are summed, beginning from x_1 and ending at x_n

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.

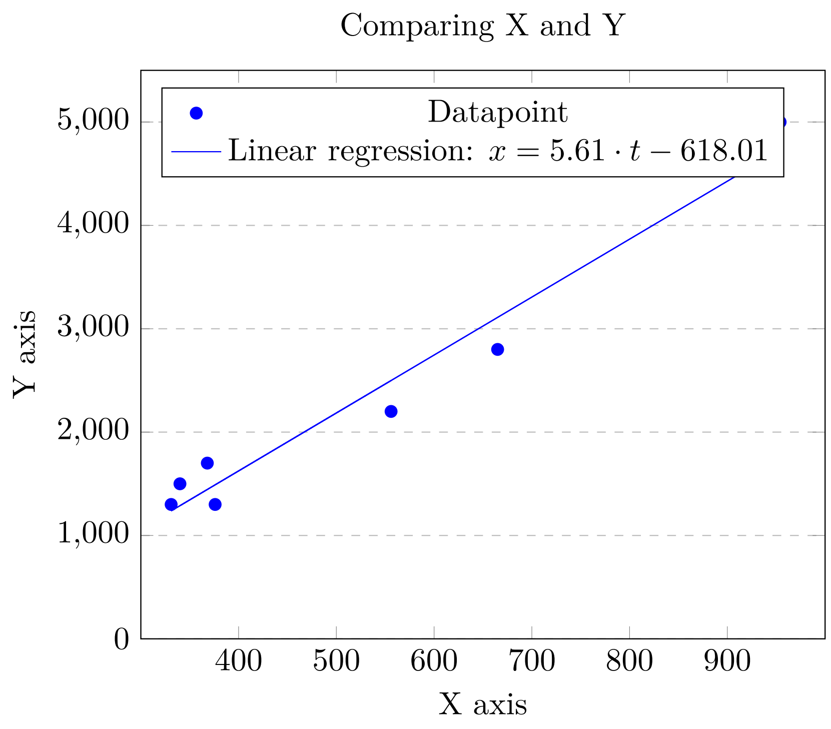
A Appendix

A.1 Example linear regression

Below is an example of a dataset with x values and their corresponding Y values, the data points chosen are purely arbitrary.

X values	Y values
368	1700
340	1500
665	2800
954	5000
331	1300
556	2200
376	1300

Below is the dataset plotted, along with a line of regression.



A.2 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
regressor = LinearRegression()
regressor.fit(X_, y_train)
y_pred = regressor.predict(predict_)

#print(regressor.score(predict_, y_test))

# Print the coefficients and intercept
print('Coefficients:␣\n', regressor.coef_)
# The mean squared error
print("Mean␣squared␣error:␣%.2f" % np.mean((y_pred - y_test) ** 2))
# Explained variance score: 1 is perfect prediction
print('Variance␣score:␣%.2f' % regressor.score(predict_, y_test))

# Code for generating 3d plot with prediction scatter
# only generates one graph, rerun with PolynomialFeatures(degree=n)
"""
fig = plt.figure()
```

```
plt.clf()
ax = Axes3D(fig)
ax = fig.gca(projection='3d')
ax.view_init(elev=15, azim=-122)

ax.set_title("2nd degree polynomial model")
ax.set_xlabel('distance from source (km)')
ax.set_ylabel('earthquake magnitude')
ax.set_zlabel('wave height (m)')

ax.scatter(X_test['distance from source (km)'].sample(frac=0.2, random_state=4
           X_test['earthquake magnitude'].sample(frac=0.2, random_state=42), y_

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

plt.show()
"""
```