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

3 Statistical Model

3.1 Cleaning the dataset

The dataset, [3]

3.2 Regression

3.3 Ease of access of data

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.

4 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.

5 Conclusion

5.1 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

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`.

A Appendix

A.1 Python program

qsd

```
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



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



def filter_distance(df, threshold):
    filter_df = df.loc[df['distance␣from␣source␣(km)'] < threshold].copy()
    return filter_df



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



def calculate_mean(x):
    x['weight'] * x['maximum␣water␣height␣(m)']
```

```
def weighted_mean_height(df):
    return df.groupby('earthquake␣magnitude', as_index=True).apply(lambda x: (x['w


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
    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')

    # Set X to the independent variables (equivalent to dropping the dependent)
    X = filter_df.drop('maximum␣water␣height␣(m)', 1)

    # Set y to our output column: maximum water height
    y = filter_df['maximum␣water␣height␣(m)']

    # The values to predict y with
    pred = [[5.2, 225]]

    # Fit the values
    poly = PolynomialFeatures(degree=2)
    X_ = poly.fit_transform(X)
    predict_ = poly.fit_transform(pred)

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

    # Print the coefficients and intercept
    theta0 = mlr_model.intercept_
    theta1, theta2, theta3, theta4, theta5, theta6 = mlr_model.coef_
    print(theta0, theta1, theta2, theta3, theta4, theta5, theta6)
```