The code used in the calculations was implemented in the latest version of Matlab (R2016b at the time of writing), however most of the code should work well in the previous versions, like 2016a, 2015b and 2015a. All files are well commented. If you find any mistakes, please let me know. The table below lists and describes the files:

Table 1: List of the Files in the jump-regression Repository and their Descriptions

File Name	Description
main.m	Implements the hypothesis test, efficient jump beta estimation, and confidence interval calculation for a single stock, using the full sample period (2007-2015). Then plots all results and save them on the 'figures/' folder.
main_sample.m	Same as 'main.m', but uses the sample data provided in the 'data/' folder of the repository. No changes are needed to run the code, simply download everything in the repository and run this script in Matlab.
fullsample.m	Same as 'main.m' but for all stocks. Uses the full sample period (2007-2015).
yearbyyear.m	Same as 'main.m' but for all stocks. The calculations are done considering year by year data separately (2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015).
simulate.m	Simulates a Gaussian diffusion process..
scripts/clearWorkspace.m	Clears Matlab workspace, and creates 'figures/' folder if it does not exist. All plots are saved in this folder by default.
script/getReturnAndDate.m	Extracts return data and serial dates (Matlab format for dates) from the raw data.
script/getSerialDate.m	Helper function used in 'getReturnAndDate.m'. Extracts year, month, day, hour, minutes, and seconds information from a dates matrix.
script/getBV.m	Calculates the bipower variance estimator for the realized variance.
script/getTOD.m	Calculates the time of day factor based on the bipower variance.
script/getCUT.m	Calculates the jump threshold for all returns, one threshold per return. If the absolute value of a return is higher than the threshold, then it is considered a jump.
script/separateReturns.m	Separates diffusive from jump returns given the jump threshold.
script/getJumpCov.m	Calculates the estimator of the jump covariance matrix $\mathcal{Q}_n$.
script/getSpotCov.m	Estimates the pre-jump and post-jump spot covariance of the bivariate process: $\hat{c}_{n,i-}, \hat{c}_{n,i+}$.
script/getSpotVol.m	Estimates the spot volatility at a jump time without assuming that the underlying volatility process is continuous.
script/jumpReg.m	Estimates the jump beta using the efficient estimator.
script/jumpRegCI.m	Calculates the confidence interval for the efficient jump beta estimator using Monte Carlo simulation.
script/jumpRegHT.m	Calculates the critical value of the hypothesis test for the jump beta being a constant.
script/plotJumpReg.m	Plots the result from 'jumpReg.m', 'jumpRegCI.m', and 'jumpRegHT.m' in a single figure. It does not automatically save the figure.
script/plotPrice.m	Plots the price of the stock against the price of the market in a single figure. It does not automatically save the figure.
script/simGeoPrice.m	Simulates geometric stock prices (log price) according to a constant coefficients gaussian diffusion process.
script/simPureJump.m	Simulates a pure jump process based on the compound poisson.

Unfortunately, I could not make the stock data available in the repository since it is attached to a license. However, the data can be bought from TickData, and I also included a simulated stock and market data in the 'data/' folder. The data we use in the project are stock prices sampled at a 5 minute interval, resulting in 78 price observations per day, during a period of 9 years. The total number of price observations is

175110, and the total number of days is 2245. Throughout the project we considered a sampling interval of $\Delta_n = 1/77$, a significance level of 5%, $\alpha = 4$ for the jump threshold calculations, and $k_n = 11$ for the spot covariance calculations. Also, the number of simulations used to calculate the confidence intervals and the hypothesis test was 10000.

If you want to test the code as fast as possible, download the files to your local machine using the big green button in the repository page ('Clone or download', then 'Download ZIP'). Run the script 'main_sample.dat', which uses the sample data provided in the 'data/' folder. The script will do all estimations and provide plots with the results. The plots will be saved in the 'figures/' folder, which is automatically created when you run the script for the first time.

If you have data that you want to analyze within the jump regression framework, the simplest way to do it is to run the 'main.m' script (see Script 1). The assumptions on the data structure are:

1. The raw data is assumed to be comprised of 3 columns:
   (a) The first column contains date in the format "yyyymmdd", representing the year, month, and day of the trade.
   (b) The second column contains time stored as "hhnn", representing hours and minutes of the trade.
   (c) The third column stores actual prices.
   (d) The same structure is assumed to hold for all stocks, including the market.
   (e) The sample size is assumed to be the same across all stocks, including the market.
   (f) The first two columns are assumed to be identical for all stocks, including the market. This means that our data is sampled at the exact same time for all stocks, and the market.

And there are a few modifications that need to be made to the code:

1. Identify how many returns per day you have in your data, and update the variable n in line 6.
2. Identify the total number of observations and update T in line 7, where T is the total number of observations divided by $n + 1$.
3. Create a folder 'data/' in your project folder and add your data to it.
4. Update the variable 'filename' in line 15 to load your data for the market. Update the ticker 'tkr' in line 16 if necessary.
5. Update the line 29 to load your stock data. Update the stock ticker 'stkr' in line 28 if necessary.

Now you can run the script and you will be presented with 2 plots: a price plot of your stock price, and another plot with the jump regression results. These plots will be saved in the 'figures/' folder.

If you want to run the 'main.m' script over multiple files, you can use the 'fullsample.m' script instead. In addition to the modifications to run the 'main.m' script, you will need to:

1. Modify the cell variable 'stocks' to contain the tickers of your stocks. For example, if you have the data for the AAPL stock in your 'data/' folder, then let 'stocks = 'AAPL''.
2. Change line 32 to accommodate for any differences in the naming of your data files. For example, my data files always ended up in '_5min.dat', so line 32 adds that to the stock ticker in order to load the data correctly.

Now you can run the script and you will be presented with several plots, depending on the number of stocks you use. All plots will be saved to the 'figures/' folder.

If you want to analyze your data on a year by year basis, you have to run the 'yearbyyear.m' script instead. The modifications follow those from 'fullsample.m':

1. Change lines 38 and 43 to account for your data file names, as done in the 'fullsample.m' case.
2. Change lines 8 and 9 to load your market data.

3. Lines 13 to 16 find the last index of the data for each year:
   (a) In line 7 we have a for-loop on the years, starting at 2007 and ending at 2015. Change these values to reflect your data sample.
   (b) In line 13 we compare the vector of dates of 'raw_full' (first column vector) to the date given by the year 'y' and the last day of December. For example, if $y = 2010$, then we try to find the last date of that year: 20101231. If that is not available, we move the second-last day: 20101230. If your data does not end on December, you will need to change this part of the code to correctly find it.

Now you can run the script and you will be presented with a lot of plots, depending on the number of stocks you use. All plots are saved to the 'figures/' folder.